# TMS320C55x DSP Algebraic Instruction Set Reference Guide

Literature Number: SPRU375G October 2002



#### **IMPORTANT NOTICE**

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of that third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments Post Office Box 655303 Dallas, Texas 75265

Copyright © 2002, Texas Instruments Incorporated

#### Preface

## **Read This First**

#### About This Manual

The TMS320C55x<sup>™</sup> is a fixed-point digital signal processor (DSP) in the TMS320<sup>™</sup> DSP family, and it can use either of two forms of the instruction set: a mnemonic form or an algebraic form. This book is a reference for the algebraic form of the instruction set. It contains information about the instructions used for all types of operations. For information on the mnemonic instruction set, see *TMS320C55x DSP Mnemonic Instruction Set Reference Guide*, SPRU374.

#### Notational Conventions

This book uses the following conventions.

In syntax descriptions, the instruction is in a **bold typeface**. Portions of a syntax in **bold** must be entered as shown. Here is an example of an instruction syntax:

Ims(Xmem, Ymem, ACx, ACy)

**Ims** is the instruction, and it has four operands: *Xmem*, *Ymem*, *ACx*, and *ACy*. When you use **Ims**, the operands should be actual dual datamemory operand values and accumulator values. A comma and a space (optional) must separate the four values.

□ Square brackets, [ and ], identify an optional parameter. If you use an optional parameter, specify the information within the brackets; do not type the brackets themselves.

#### **Related Documentation From Texas Instruments**

The following books describe the C55x<sup>™</sup> devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477-8924. When ordering, please identify the book by its title and literature number.

- **TMS320C55x Technical Overview** (SPRU393). This overview is an introduction to the TMS320C55x<sup>™</sup> digital signal processor (DSP). The TMS320C55x is the latest generation of fixed-point DSPs in the TMS320C5000<sup>™</sup> DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features of the TMS320C55x.
- **TMS320C55x DSP CPU Reference Guide** (literature number SPRU371) describes the architecture, registers, and operation of the CPU for the TMS320C55x<sup>™</sup> digital signal processors (DSPs).
- **TMS320C55x DSP Mnemonic Instruction Set Reference Guide** (literature number SPRU374) describes the mnemonic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.
- **TMS320C55x Programmer's Guide** (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x<sup>™</sup> DSPs and explains how to write code that uses special features and instructions of the DSP.
- **TMS320C55x Optimizing C Compiler User's Guide** (literature number SPRU281) describes the TMS320C55x<sup>™</sup> C Compiler. This C compiler accepts ANSI standard C source code and produces assembly language source code for TMS320C55x devices.
- TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x<sup>™</sup> devices.

#### Trademarks

TMS320, TMS320C54x, TMS320C55x, C54x, and C55x are trademarks of Texas Instruments.

1	Lists	and def	bols, and Abbreviations fines the terms, symbols, and abbreviations used in the TMS320C55 ruction set summary and in the individual instruction descriptions.	
	1.1	Instruc	tion Set Terms, Symbols, and Abbreviations	1-2
	1.2		tion Set Conditional (cond) Fields	
	1.3		of Status Bits	
		1.3.1	Accumulator Overflow Status Bit (ACOVx)	1-9
		1.3.2	C54CM Status Bit	
		1.3.3	CARRY Status Bit	
		1.3.4	FRCT Status Bit	
		1.3.5	INTM Status Bit	
		1.3.6	M40 Status Bit	
		1.3.7	RDM Status Bit	1-12
		1.3.8	SATA Status Bit	1-12
		1.3.9	SATD Status Bit	
		1.3.10	SMUL Status Bit	
		1.3.11	SXMD Status Bit	1-13
		1.3.12	Test Control Status Bit (TCx)	1-13
	1.4	Instruc	tion Set Notes and Rules	
		1.4.1	Notes	
		1.4.2	Rules	1-14
	1.5	Nonrep	peatable Instructions	1-20
2	Paral	lelism F	eatures and Rules	2-1
_			parallelism features and rules of the TMS320C55x DSP algebraic instruction	
	2.1	Paralle	lism Features	2-2
	2.2	Paralle	lism Basics	2-3
	2.3	Resour	rce Conflicts	2-4
		2.3.1	Operators	
		2.3.2	Address Generation Units	
		2.3.3	Buses	
	2.4		ual Parallelism	
		2.4.1	Soft-Dual Parallelism of MAR Instructions	
	2.5		e Conditionally Instructions	
	2.6		Exceptions	
			•	

3	Introduction to Addressing Modes Provides an introduction to the addressing modes of the TMS320C55x DSP.	3-1
	3.1 Introduction to the Addressing Modes	3-2
	3.2 Absolute Addressing Modes	3-3
	3.2.1 k16 Absolute Addressing Mode	
	3.2.2 k23 Absolute Addressing Mode	3-3
	3.2.3 I/O Absolute Addressing Mode	3-3
	3.3 Direct Addressing Modes	3-4
	3.3.1 DP Direct Addressing Mode	3-4
	3.3.2 SP Direct Addressing Mode	3-5
	3.3.3 Register-Bit Direct Addressing Mode	
	3.3.4 PDP Direct Addressing Mode	3-5
	3.4 Indirect Addressing Modes	3-6
	3.4.1 AR Indirect Addressing Mode	3-6
	3.4.2 Dual AR Indirect Addressing Mode	3-14
	3.4.3 CDP Indirect Addressing Mode	3-16
	3.4.4 Coefficient Indirect Addressing Mode	3-18
	3.5 Circular Addressing	3-20
4	Instruction Set Summary	1-1
-	Provides a summary of the TMS320C55x DSP algebraic instruction set.	·····
5	Instruction Set Descriptions	5-1
	Detailed information on the TMS320C55x DSP algebraic instruction set.	
	Absolute Distance (abdst)	
	Absolute Value	
	Addition	
	Addition with Absolute Value	
	Addition with Parallel Store Accumulator Content to Memory	
	Addition or Subtraction Conditionally (adsc)	
	Addition or Subtraction Conditionally with Shift (ads2c)	
	Addition, Subtraction, or Move Accumulator Content Conditionally (adsc)	
	Bitwise AND	
	Bitwise AND Memory with Immediate Value and Compare to Zero	
	Bitwise OR	
	Bitwise Exclusive OR (XOR)	
	Branch Conditionally (if goto)	5-66
	Branch Unconditionally (goto)	
	Branch on Auxiliary Register Not Zero (if goto)	5-74
	Call Conditionally (if call)	
	Call Unconditionally (call)	5-83
	Circular Addressing Qualifier (circular)	
	Clear Accumulator, Auxiliary, or Temporary Register Bit	
	Clear Memory Bit	
	Clear Status Register Bit	
	Compare Accumulator, Auxiliary, or Temporary Register Content	
	Compare Accumulator, Auxiliary, or Temporary Register Content with AND	
	Compare Accumulator, Auxiliary, or Temporary Register Content with OR	

Compare Accumulator, Auxiliary, or Temporary Register Content Maximum (max)	
Compare Accumulator, Auxiliary, or Temporary Register Content Minimum (min)	
Compare and Branch (compare goto)	5-111
Compare and Select Accumulator Content Maximum (max_diff)	5-114
Compare and Select Accumulator Content Minimum (min_diff)	5-120
Compare Memory with Immediate Value	5-126
Complement Accumulator, Auxiliary, or Temporary Register Bit (cbit)	5-128
Complement Accumulator, Auxiliary, or Temporary Register Content	5-129
Complement Memory Bit (cbit)	5-130
Compute Exponent of Accumulator Content (exp)	
Compute Mantissa and Exponent of Accumulator Content (mant, exp)	
Count Accumulator Bits (count)	
Dual 16-Bit Additions	
Dual 16-Bit Addition and Subtraction	
Dual 16-Bit Subtractions	
Dual 16-Bit Subtraction and Addition	
Execute Conditionally (if execute)	
Expand Accumulator Bit Field (field_expand)	
Extract Accumulator Bit Field (field_extract)	
Finite Impulse Response Filter, Antisymmetrical (firsn)	
Finite Impulse Response Filter, Symmetrical (firs)	
Idle	
Least Mean Square (Ims)	
Linear Addressing Qualifier (linear)	
Load Accumulator from Memory	
Load Accumulator from Memory with Parallel Store Accumulator Content to Memory	
Load Accumulator Pair from Memory	
Load Accumulator with Immediate Value	
Load Accumulator, Auxiliary, or Temporary Register from Memory	
Load Accumulator, Auxiliary, or Temporary Register with Immediate Value	
Load Auxiliary or Temporary Register Pair from Memory	
Load CPU Register from Memory	
Load CPU Register with Immediate Value	
Load Extended Auxiliary Register from Memory	
Load Extended Auxiliary Register with Immediate Value	
Load Memory with Immediate Value	
Memory Delay (delay)	
Memory-Mapped Register Access Qualifier (mmap)	5-213
Modify Auxiliary Register Content (mar)	0 - 10
Modify Auxiliary Register Content with Parallel Multiply	
Modify Auxiliary Register Content with Parallel Multiply and Accumulate	
Modify Auxiliary Register Content with Parallel Multiply and Subtract	
Modify Auxiliary or Temporary Register Content (mar)	
Modify Auxiliary or Temporary Register Content by Addition (mar)	
Modify Auxiliary or Temporary Register Content by Subtraction (mar)	
Modify Data Stack Pointer	
Modify Extended Auxiliary Register Content (mar)	5-232
	5 200

vii

Move Accumulator Content to Auxiliary or Temporary Register	5-239
Move Accumulator, Auxiliary, or Temporary Register Content	5-240
Move Auxiliary or Temporary Register Content to Accumulator	5-242
Move Auxiliary or Temporary Register Content to CPU Register	
Move CPU Register Content to Auxiliary or Temporary Register	5-245
Move Extended Auxiliary Register Content	5-247
Move Memory to Memory	5-248
Multiply	5-255
Multiply with Parallel Multiply and Accumulate	
Multiply with Parallel Store Accumulator Content to Memory	5-269
Multiply and Accumulate (MAC)	5-271
Multiply and Accumulate with Parallel Delay	5-286
Multiply and Accumulate with Parallel Load Accumulator from Memory	5-288
Multiply and Accumulate with Parallel Multiply	5-290
Multiply and Accumulate with Parallel Store Accumulator Content to Memory	5-292
Multiply and Subtract	5-294
Multiply and Subtract with Parallel Load Accumulator from Memory	5-302
Multiply and Subtract with Parallel Multiply	5-304
Multiply and Subtract with Parallel Multiply and Accumulate	5-306
Multiply and Subtract with Parallel Store Accumulator Content to Memory	5-311
Negate Accumulator, Auxiliary, or Temporary Register Content	5-313
No Operation (nop)	5-315
Parallel Modify Auxiliary Register Contents (mar)	5-316
Parallel Multiplies	5-317
Parallel Multiply and Accumulates	5-319
Parallel Multiply and Subtracts	5-326
Peripheral Port Register Access Qualifiers	5-328
Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers	
(popboth)	
Pop Top of Stack (pop)	
Push Accumulator or Extended Auxiliary Register Content to Stack Pointers (pshboth)	
Push to Top of Stack (push)	
Repeat Block of Instructions Unconditionally	
Repeat Single Instruction Conditionally (while/repeat)	
Repeat Single Instruction Unconditionally (repeat)	
Repeat Single Instruction Unconditionally and Decrement CSR (repeat)	
Repeat Single Instruction Unconditionally and Increment CSR (repeat)	
Return Conditionally (if return)	
Return Unconditionally (return)	
Return from Interrupt (return_int)	
Rotate Left Accumulator, Auxiliary, or Temporary Register Content	
Rotate Right Accumulator, Auxiliary, or Temporary Register Content	
Round Accumulator Content (rnd)	
Saturate Accumulator Content (saturate)	
Set Accumulator, Auxiliary, or Temporary Register Bit	
Set Memory Bit	
Set Status Register Bit	5-386

	Shift Accumulator Content Conditionally (sftc)	
	Shift Accumulator Content Logically	
	Signed Shift of Accumulator Content	
	Signed Shift of Accumulator, Auxiliary, or Temporary Register Content	
	Software Interrupt (intr)	
	Software Reset (reset)	
	Soliware map (itap)	
	Square and Accumulate	
	Square and Subtract	
	Square Distance (sqdst)	
	Store Accumulator Content to Memory	
	Store Accumulator Pair Content to Memory	
	Store Accumulator, Auxiliary, or Temporary Register Content to Memory	
	Store Auxiliary or Temporary Register Pair Content to Memory	
	Store CPU Register Content to Memory	
	Store Extended Auxiliary Register Content to Memory Subtract Conditionally (subc)	
	Subtraction	
	Subtraction with Parallel Store Accumulator Content to Memory	
	Swap Accumulator Content (swap)	
	Swap Accumulator Pair Content (swap)	
	Swap Accumulator Pair Content (swap)	
	Swap Auxiliary Register Pair Content (swap)	
	Swap Auxiliary and Temporary Register Content (swap)	
	Swap Auxiliary and Temporary Register Pair Content (swap)	
	Swap Auxiliary and Temporary Register Pairs Content (swap)	
	Swap Temporary Register Content (swap)	
	Swap Temporary Register Pair Content (swap)	
	Test Accumulator, Auxiliary, or Temporary Register Bit	
	Test Accumulator, Auxiliary, or Temporary Register Bit Pair	
	Test Memory Bit Test and Clear Memory Bit	
	,	
	Test and Complement Memory Bit	
	Test and Set Memory Bit	5-513
6	Instruction Opcodes in Sequential Order	6-1
	The opcode in sequential order for each TMS320C55x DSP instruction syntax.	
	6.1 Instruction Set Opcodes	
	6.2 Instruction Set Opcode Symbols and Abbreviations	. 6-16
7	Cross-Reference of Algebraic and Mnemonic Instruction Sets Cross-Reference of TMS320C55x DSP Algebraic and Mnemonic Instruction Sets.	7-1

# Figures

	Status Registers Bit Mapping Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat)	5-92
	Instruction	5-350
5–3	Status Registers Bit Mapping	5-388
5–4	Effects of a Software Reset on Status Registers	5-416

# Tables

1_1	Instruction Set Terms, Symbols, and Abbreviations	1-2
1–2	Operators Used in Instruction Set	
1_3	Instruction Set Conditional (cond) Field	
1-4	Nonrepeatable Instructions	
3–1	Addressing-Mode Operands	
3–2	Absolute Addressing Modes	
3–3	Direct Addressing Modes	
3–4	Indirect Addressing Modes	
3–5	DSP Mode Operands for the AR Indirect Addressing Mode	
3–6	Control Mode Operands for the AR Indirect Addressing Mode	
3–7	Dual AR Indirect Operands	
3–8	CDP Indirect Operands	
3–9	Coefficient Indirect Operands	
3–10	Circular Addressing Pointers	
4–1	Algebraic Instruction Set Summary	
5–1	Opcodes for Load CPU Register from Memory Instruction	
5–2	Opcodes for Load CPU Register with Immediate Value Instruction	5-208
5–3	Opcodes for Move Auxiliary or Temporary Register Content to CPU Register	
5–4	Opcodes for Move CPU Register Content to Auxiliary or Temporary Register	5-246
5–5	Effects of a Software Reset on DSP Registers	5-414
5–6	Opcodes for Store CPU Register Content to Memory Instruction	5-461
6–1	Instruction Set Opcodes	
6–2	Instruction Set Opcode Symbols and Abbreviations	6-16
7–1	Cross-Reference of Algebraic and Mnemonic Instruction Sets	7-2

### **Chapter 1**

# Terms, Symbols, and Abbreviations

This chapter lists and defines the terms, symbols, and abbreviations used in the TMS320C55x<sup>™</sup> DSP algebraic instruction set summary and in the individual instruction descriptions. Also provided are instruction set notes and rules and a list of nonrepeatable instructions.

# TopicPage1.1Instruction Set Terms, Symbols, and Abbreviations1-21.2Instruction Set Conditional (cond) Fields1-71.3Affect of Status Bits1-91.4Instruction Set Notes and Rules1-141.5Nonrepeatable Instructions1-20

#### 1.1 Instruction Set Terms, Symbols, and Abbreviations

Table 1–1 lists the terms, symbols, and abbreviations used and Table 1–2 lists the operators used in the instruction set summary and in the individual instruction descriptions.

Symbol Meaning [] **Optional operands** ACB Bus that brings D-unit registers to A-unit and P-unit operators ACOVx Accumulator overflow status bit: ACOV0, ACOV1, ACOV2, ACOV3 ACw, ACx, Accumulator: ACy, ACz AC0, AC1, AC2, AC3 Content of selected auxiliary register (ARn) is premodified or postmodified in the address ARn\_mod generation unit. ARx, ARy Auxiliary register: AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 AU A unit Baddr Register bit address BitIn Shifted bit in: Test control flag 2 (TC2) or CARRY status bit BitOut Shifted bit out: Test control flag 2 (TC2) or CARRY status bit BORROW Logical complement of CARRY status bit C, Cycles Execution in cycles. For conditional instructions, x/y field means: x cycle, if the condition is true. y cycle, if the condition is false. CA Coefficient address generation unit CARRY Value of CARRY status bit Cmem Coefficient indirect operand referencing a 16-bit or 32-bit value in data space Condition based on accumulator value (ACx), auxiliary register (ARx) value, temporary cond register (Tx) value, test control (TCx) flag, or CARRY status bit. See section 1.2. CR Coefficient Read bus CSR Computed single-repeat register

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations

1-2 Terms, Symbols, and Abbreviations

Symbol	Meaning
DA	Data address generation unit
DR	Data Read bus
dst	Destination accumulator (ACx), lower 16 bits of auxiliary register (ARx), or temporary register (Tx): AC0, AC1, AC2, AC3 AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
DU	D unit
DW	Data Write bus
Dx	Data address label coded on x bits (absolute address)
E	Indicates if the instruction contains a parallel enable bit.
KAB	Constant bus
KDB	Constant bus
kx	Unsigned constant coded on x bits
Kx	Signed constant coded on x bits
Lmem	Long-word single data memory access (32-bit data access). Same legal inputs as Smem.
lx	Program address label coded on x bits (unsigned offset relative to program counter register)
Lx	Program address label coded on x bits (signed offset relative to program counter register)
M40	If the optional M40 keyword is applied to the instruction, the instruction provides the option to locally set M40 to 1 for the execution of the instruction
Operator	Operator(s) used by an instruction.
Pipe, Pipeline	Pipeline phase in which the instruction executes: AD Address D Decode R Read X Execute
Px	Program or data address label coded on x bits (absolute address)

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning
RELOP	Relational operators:
	<pre>== equal to &lt; less than &gt;= greater than or equal to != not equal to</pre>
rnd	If the optional rnd keyword is applied to the instruction, rounding is performed in the instruction
RPTC	Single-repeat counter register
S, Size	Instruction size in bytes.
SA	Stack address generation unit
saturate	If the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated
SHFT	4-bit immediate shift value, 0 to 15
SHIFTW	6-bit immediate shift value, -32 to +31
Smem	Word single data memory access (16-bit data access)
SP	Data stack pointer
SIC	Source accumulator (ACx), lower 16 bits of auxiliary register (ARx), or temporary register (Tx): AC0, AC1, AC2, AC3 AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
SSP	System stack pointer
STx	Status register: ST0, ST1, ST2, ST3
TAx, TAy	Auxiliary register (ARx) or temporary register (Tx): AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7 T0, T1, T2, T3
ТСх, ТСу	Test control flag: TC1, TC2
TRNx	Transition register: TRN0, TRN1
Tx, Ty	Temporary register (Tx): T0, T1, T2, T3

Table 1–1. Instruction Set Terms, Symbols, and Abbreviations (Continued)

Symbol	Meaning	
uns	If the optional uns keyword is applied to the input operand, the operand is zero extended	
XAdst	Destination extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7	
XARx	All 23 bits of extended auxiliary register: XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7	
XAsrc	Source extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7	
xdst	Accumulator: AC0, AC1, AC2, AC3	
	Destination extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7	
xsrc	Accumulator: AC0, AC1, AC2, AC3	
	Source extended register: All 23 bits of data stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7	
Xmem, Ymem	Indirect dual data memory access (two data accesses)	

Symbols			Operators	Evaluation
+	_	~	Unary plus, minus, 1s complement	Right to left
*	/	%	Multiplication, division, modulo	Left to right
+		_	Addition, subtraction	Left to right
<<		>>	Signed left shift, right shift	Left to right
< < <		>>>	Logical left shift, logical right shift	Left to right
<		<=	Less than, less than or equal to	Left to right
>		>=	Greater than, greater than or equal to	Left to right
==		!=	Equal to, not equal to	Left to right
&			Bitwise AND	Left to right
I			Bitwise OR	Left to right
^			Bitwise exclusive OR (XOR)	Left to right

Table 1–2. Operators Used in Instruction Set

Note: Unary +, –, and \* have higher precedence than the binary forms.

#### 1.2 Instruction Set Conditional (cond) Fields

Table 1–3 lists the testing conditions available in the cond field of the conditional instructions.

Bit or Register	Condition (cond) Field	For Condition to be True	
Accumulator	Tests the accumulator (ACx) content against 0. The comparison against 0 depends on M40 status bit:		
	☐ If M40 = 0, ACx(31–0)	is compared to 0.	
	☐ If M40 = 1, ACx(39–0)	is compared to 0.	
	ACx == #0	ACx content is equal to 0	
	ACx < #0	ACx content is less than 0	
	ACx > #0	ACx content is greater than 0	
	ACx != #0	ACx content is not equal to 0	
	ACx <= #0	ACx content is less than or equal to 0	
	ACx >= #0	ACx content is greater than or equal to 0	
Accumulator Overflow Status Bit	Tests the accumulator overflow status bit (ACOVx) against 1; when the optional ! symbol is used before the bit designation, the bit can be tested against 0. When this condition is used, the corresponding ACOVx is cleared to 0.		
	overflow(ACx)	ACOVx bit is set to 1	
	!overflow(ACx)	ACOVx bit is cleared to 0	
Auxiliary Register	Tests the auxiliary register (ARx) content against 0.		
	ARx == #0	ARx content is equal to 0	
	ARx < #0	ARx content is less than 0	
	ARx > #0	ARx content is greater than 0	
	ARx != #0	ARx content is not equal to 0	
	ARx <= #0	ARx content is less than or equal to 0	
	ARx >= #0	ARx content is greater than or equal to 0	
CARRY Status Bit		against 1; when the optional ! symbol is used he bit can be tested against 0.	
	CARRY	CARRY bit is set to 1	
	!CARRY	CARRY bit is cleared to 0	

Table 1–3. Instruction Set Conditional (cond) Field

SPRU375G

Terms, Symbols, and Abbreviations 1-7

Bit or Register	Condition (cond) Field	For Condition to be True
Temporary Register	Tests the temporary register (Tx) content against 0.	
	Tx == #0	Tx content is equal to 0
	Tx < #0	Tx content is less than 0
	Tx > #0	Tx content is greater than 0
	Tx != #0	Tx content is not equal to 0
	Tx <= #0	Tx content is less than or equal to 0
	Tx >= #0	Tx content is greater than or equal to 0
Test Control Flags	Tests the test control flags (TC1 and TC2) independently against 1; when the optional ! symbol is used before the flag designation, the flag can be tested independently against 0.	
	TCx	TCx flag is set to 1
	!TCx	TCx flag is cleared to 0
	TC1 and TC2 can be combined with an AND (&), OR ( ), and XOR (^) logical bit combinations:	
	TC1 & TC2	TC1 AND TC2 is equal to 1
	!TC1 & TC2	TC1 AND TC2 is equal to 1
	TC1 & !TC2	TC1 AND $\overline{\text{TC2}}$ is equal to 1
	!TC1 & !TC2	TC1 AND TC2 is equal to 1
	TC1   TC2	TC1 OR TC2 is equal to 1
	!TC1   TC2	TC1 OR TC2 is equal to 1
	TC1   !TC2	TC1 OR $\overline{\text{TC2}}$ is equal to 1
	!TC1   !TC2	$\overline{\text{TC1}}$ OR $\overline{\text{TC2}}$ is equal to 1
	TC1 ^ TC2	TC1 XOR TC2 is equal to 1
	!TC1 ^ TC2	TC1 XOR TC2 is equal to 1
	TC1 ^ !TC2	TC1 XOR $\overline{\text{TC2}}$ is equal to 1
	!TC1 ^ !TC2	TC1 XOR TC2 is equal to 1

Table 1–3. Instruction Set Conditional (cond) Field (Continued)

#### 1.3 Affect of Status Bits

#### **1.3.1** Accumulator Overflow Status Bit (ACOVx)

The ACOV[0–3] depends on M40:

- $\Box$  When M40 = 0, overflow is detected at bit position 31
- $\Box$  When M40 = 1, overflow is detected at bit position 39

If an overflow is detected, the destination accumulator overflow status bit is set to 1.

#### 1.3.2 C54CM Status Bit

- When C54CM = 0, the enhanced mode, the CPU supports code originally developed for a TMS320C55x<sup>™</sup> DSP.
- When C54CM = 1, the compatible mode, all the C55x CPU resources remain available; therefore, as you translate code, you can take advantage of the additional features on the C55x DSP to optimize your code. This mode must be set when you are porting code that was originally developed for a TMS320C54x<sup>™</sup> DSP.

#### 1.3.3 CARRY Status Bit

- $\Box$  When M40 = 0, the carry/borrow is detected at bit position 31
- □ When M40 = 1, the carry/borrow is detected at bit position 39

When performing a logical shift or signed shift that affects the CARRY status bit and the shift count is zero, the CARRY status bit is cleared to 0.

#### 1.3.4 FRCT Status Bit

- ☐ When FRCT = 0, the fractional mode is OFF and results of multiply operations are not shifted.
- When FRCT = 1, the fractional mode is ON and results of multiply operations are shifted left by 1 bit to eliminate an extra sign bit.

#### 1.3.5 INTM Status Bit

The INTM bit globally enables or disables the maskable interrupts. This bit has no effect on nonmaskable interrupts (those that cannot be blocked by software).

- $\Box$  When INTM = 0, all unmasked interrupts are enabled.
- $\Box$  When INTM = 1, all maskable interrupts are disabled.

#### SPRU375G

#### 1.3.6 M40 Status Bit

□ When M40 = 0:

- overflow is detected at bit position 31
- the carry/borrow is detected at bit position 31
- saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- TMS320C54x<sup>TM</sup> DSP compatibility mode
- for conditional instructions, the comparison against 0 (zero) is performed on 32 bits, ACx(31–0)
- □ When M40 = 1:
  - overflow is detected at bit position 39
  - the carry/borrow is detected at bit position 39
  - saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)
  - for conditional instructions, the comparison against 0 (zero) is performed on 40 bits, ACx(39–0)

#### 1.3.6.1 M40 Status Bit When Sign Shifting

In D-unit shifter:

- U When shifting to the LSBs:
  - when M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted according to the shift quantity:
    - if SXMD = 0, 0 is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
    - if SXMD = 1, bit 31 of the source operand is substituted for the guard bits (39–32) as the input, instead of ACx(39–32), to the shifter
  - bit 39 is extended according to SXMD
  - the shifted-out bit is extracted at bit position 0
- U When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - if M40 = 0, the shifted-out bit is extracted at bit position 31
  - if M40 = 1, the shifted-out bit is extracted at bit position 39

- $\Box$  After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (if an overflow is detected, the destination ACOVx bit is set)
  - the carry/borrow is detected at bit position 31
  - if SATD = 1, when an overflow is detected, ACx saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
  - TMS320C54x<sup>™</sup> DSP compatibility mode
- $\Box$  After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVx bit is set)
  - the carry/borrow is detected at bit position 39
  - if SATD = 1, when an overflow is detected, ACx saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

In A-unit ALU:

- When shifting to the LSBs, bit 15 is sign extended
- When shifting to the MSBs, 0 is inserted at bit position 0
- After shifting, unless otherwise noted:
  - overflow is detected at bit position 15 (if an overflow is detected, the destination ACOVx bit is set)
  - if SATA = 1, when an overflow is detected, register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

#### 1.3.6.2 M40 Status Bit When Logically Shifting

In D-unit shifter:

- U When shifting to the LSBs:
  - if M40 = 0, 0 is inserted at bit position 31 and the guard bits (39–32) of the destination accumulator are cleared
  - if M40 = 1, 0 is inserted at bit position 39
  - the shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit

SPRU375G

U When shifting to the MSBs:

- 0 is inserted at bit position 0
- if M40 = 0, the shifted-out bit is extracted at bit position 31 and stored in the CARRY status bit, and the guard bits (39–32) of the destination accumulator are cleared
- if M40 = 1, the shifted-out bit is extracted at bit position 39 and stored in the CARRY status bit

In A-unit ALU:

- U When shifting to the LSBs:
  - 0 is inserted at bit position 15
  - the shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit
- U When shifting to the MSBs:
  - 0 is inserted at bit position 0
  - the shifted-out bit is extracted at bit position 15 and stored in the CARRY status bit

#### 1.3.7 RDM Status Bit

When the optional rnd or R keyword is applied to the instruction, then rounding is performed in the D-unit shifter. This is done according to RDM:

- □ When RDM = 0, the biased rounding to the infinite is performed. 8000h  $(2^{15})$  is added to the 40-bit result of the shift result.
- When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit result of the shift result, 8000h (2<sup>15</sup>) is added:
  - if( 8000h < bit(15-0) < 10000h)
     add 8000h to the 40-bit result of the shift result.
    else if( bit(15-0) == 8000h)
     if( bit(16) == 1)
     add 8000h to the 40-bit result of the shift result.</pre>

If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.

#### 1.3.8 SATA Status Bit

This status bit controls operations performed in the A unit.

- $\Box$  When SATA = 0, no saturation is performed.
- When SATA = 1 and an overflow is detected, the destination register is saturated to 7FFFh (positive overflow) or 8000h (negative overflow).

#### 1.3.9 SATD Status Bit

This status bit controls operations performed in the D unit.

- $\Box$  When SATD = 0, no saturation is performed.
- □ When SATD = 1 and an overflow is detected, the destination register is saturated.

#### 1.3.10 SMUL Status Bit

- $\Box$  When SMUL = 0, the saturation mode is OFF.
- ❑ When SMUL = 1, the saturation mode is ON. When SMUL = 1, FRCT = 1, and SATD = 1, the result of 18000h × 18000h is saturated to 00 7FFF FFFFh (regardless of the value of the M40 bit). This forces the product of the two negative numbers to be a positive number. For multiply-and-accumulate/subtract instructions, the saturation is performed after the multiplication and before the addition/subtraction.

#### 1.3.11 SXMD Status Bit

This status bit controls operations performed in the D unit.

- $\Box$  When SXMD = 0, input operands are zero extended.
- $\Box$  When SXMD = 1, input operands are sign extended.

#### 1.3.12 Test Control Status Bit (TCx)

The test/control status bits (TC1 or TC2) hold the result of a test performed by the instruction.

#### 1.4 Instruction Set Notes and Rules

#### 1.4.1 Notes

Algebraic syntax keywords and operand modifiers are case insensitive. You can write:

abdst(\*AR0, \*ar1, AC0, ac1)
or
aBdST(\*ar0, \*aR1, aC0, Ac1)

- Operands for commutative operations (+, \*, &, |, ^) can be arranged in any order.
- Expression qualifiers can be specified in any order. For example, these two instructions are equivalent:

```
AC0 = m40(rnd(uns(*AR0) * uns(*AR1)))
AC0 = rnd(m40(uns(*AR0) * uns(*AR1)))
```

Algebraic instructions must use parenthesis in the exact form shown in the instruction set. For example, this instruction is legal:

AC0 = AC0 + (AC1 << T0)

while both of these instructions are illegal:

AC0 = AC0 + ((AC1 << T0)) AC0 = AC0 + AC1 << T0

#### 1.4.2 Rules

Simple instructions are not allowed to span multiple lines. One exception, single instructions that use the "," notation to imply parallelism. These instructions may be split up following the "," notation.

The following example shows a single instruction (dual multiply) occupying two lines:

```
ACx = m40(rnd(uns(Xmem) * uns(coef(Cmem)))),
ACy = m40(rnd(uns(Ymem) * uns(coef(Cmem))))
```

□ User-defined parallelism instructions (using || notation) are allowed to span multiple lines. For example, all of the following instructions are legal:

```
AC0 = AC1 || AC2 = AC3
AC0 = AC1 ||
AC2 = AC3
AC0 = AC1
|| AC2 = AC3
AC0 = AC1
||
AC2 = AC3
```

□ The block repeat syntax uses braces to delimit the block that is to be repeated:

```
blockrepeat {
    instr
    instr
    instr
    }
localrepeat {
    instr
    instr
    instr
    instr
    instr
    instr
    }
}
```

The left opening brace must appear on the same line as the repeat keyword. The right closing brace must appear alone on a line (trailing comments allowed).

Note that a label placed just inside the closing brace of the loop is effectively outside the loop. The following two code sequences are equivalent:

```
localrepeat {
    instr1
    instr2
Label:
    }
    instr3
and
localrepeat {
    instr1
    instr2
    }
    Label:
    instr3
```

A label is the address of the first construct following the label that gets assembled into code in the object file. A closing brace does not generate any code and so the label marks the address of the first instruction that generates code, that is, instr3.

In this example, "goto Label" exits the loop, which is somewhat unintuitive:

```
localrepeat {
    goto Label
    instr2
    Label:
    }
    instr3
```

SPRU375G

#### 1.4.2.1 Reserved Words

Register names and algebraic syntax keywords are reserved. They may not be used as names of identifiers, labels, etc.

#### 1.4.2.2 Literal and Address Operands

Literals in the algebraic strings are denoted as K or k fields. In the Smem address modes that require an offset, the offset is also a literal (K16 or k3). 8-bit and 16-bit literals are allowed to be linktime-relocatable; for other literals, the value must be known at assembly time.

Addresses are the elements of the algebraic strings denoted by P, L, and I. Further, 16-bit and 24-bit absolute address Smem modes are addresses, as is the dma Smem mode, denoted by the '@' syntax. Addresses may be assembly-time constants or symbolic linktime-known constants or expressions.

Both literals and addresses follow syntax rule 1. For addresses only, rules 2 and 3 also apply.

#### Rule 1

A valid address or literal is a # followed by one of the following:

- **a number** (#123)
- □ an identifier (#FOO)
- □ a parenthesized expression (#(FOO + 2))

Note that # is not used inside the expression.

#### Rule 2

When an address is used in a dma, the address does not need to have a leading #, be it a number, a symbol or an expression. These are all legal:

```
@#123
@123
@#foo
@foo
@#(foo+2)
@(foo+2)
```

#### Rule 3

When used in contexts other than dma (such as branch targets or Smemabsolute address), addresses generally need a leading #. As a convenience, the # may be omitted in front of an identifier. These are all legal:

Brand	ch	Absolute Address
goto	#123	*(#123)
goto	#foo	*(#foo)
goto	foo	*(foo)
goto	#(foo+2)	*(#(foo+2))
These are	e illegal:	
goto	123	*(123)
goto	(foo+2)	*((foo+2))

#### 1.4.2.3 Memory Operands

- Syntax of Smem is the same as that of Lmem or Baddr.
- In the following instruction syntaxes, Smem cannot reference to a memory-mapped register (MMR). No instruction can access a byte within a memory-mapped register. If Smem is an MMR in one of the following syntaxes, the DSP sends a hardware bus-error interrupt (BERRINT) request to the CPU.

```
dst = uns(high_byte(Smem))
dst = uns(low_byte(Smem))
ACx = low_byte(Smem) << #SHIFTW
ACx = high_byte(Smem) << #SHIFTW
high_byte(Smem) = src
low_byte(Smem) = src</pre>
```

Syntax of Xmem is the same as that of Ymem.

Syntax of coefficient operands, Cmem:

```
*CDP
*CDP+
*CDP-
*(CDP + T0), when C54CM = 0
*(CDP + AR0), when C54CM = 1
```

When an instruction uses a Cmem operand with paralleled instructions, the pointer modification of the Cmem operand must be the same for both instructions of the paralleled pair or the assembler generates an error. For example:

AC0 = AC0 + (\*AR2+ \* coef(\*CDP+)), AC1 = AC1 + (\*AR3+ \* coef(\*CDP+)) □ An optional mmr prefix is allowed to be specified for indirect memory operands, for example, mmr (\*AR0). This is an assertion by you that this is an access to a memory-mapped register. The assembler checks whether such access is legal in given circumstances.

The mmr prefix is supported for Xmem, Ymem, indirect Smem, indirect Lmem, and Cmem operands. It is not supported for direct memory operands; it is expected that an explicit mmap() parallel instruction is used in conjunction with direct memory operands to indicate MMR access.

Note that the mmr prefix is part of the syntax. It is an implementation restriction that mmr cannot exchange positions with other prefixes around the memory operand, such as dbl or uns. If several prefixes are specified, mmr must be the innermost prefix. Thus, uns(mmr(\*AR0)) is legal, but mmr(uns(\*AR0)) is not legal.

- ☐ The following indirect operands **cannot** be used for accesses to I/O space. An instruction using one of these operands requires a 2-byte extension for the constant. This extension would prevent the use of the port() qualifier needed to indicate an I/O-space access.
  - \*ARn(#K16) \*+ARn(#K16) \*CDP(#K16) \*+CDP(#K16)

Also, the following instructions that include the delay operation cannot be used for accesses to I/O space:

```
delay(Smem)
ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem],
delay(Smem)
```

Any illegal access to I/O space will generate a hardware bus-error interrupt (BERRINT) to be handled by the CPU.

#### 1.4.2.4 Operand Modifiers

Operand modifiers look like function calls on operands. Note that uns is an operand modifier and an instruction modifier meaning unsigned. The operand modifier uns is used when the operand is modified on the way to the rest of the operation (multiply-and-accumulate). The instruction modifier uns is used when the whole operation is affected (multiply, register compare, compare and branch).

Modifier	Meaning
dbl	Access a true 32-bit memory operand
dual	Access a 32-bit memory operand for use as two independent 16-bit halves of the given operation
HI	Access upper 16 bits of the accumulator
high_byte	Access the high byte of the memory location
LO	Access lower 16 bits of the accumulator
low_byte	Access the low byte of the memory location
pair	Dual register access
rnd	Round
saturate	Saturate
uns	Unsigned operand

When an instruction uses a Cmem operand with paralleled instructions and the Cmem operand is defined as unsigned (uns), both Cmem operands of the paralleled pair must be defined as unsigned (and reciprocally).

When an instruction uses both Xmem and Ymem operands with paralleled instructions and the Xmem operand is defined as unsigned (uns), Ymem operand must also be defined as unsigned (and reciprocally).

#### 1.4.2.5 Operator Syntax Rules

Instructions that read and write the same operand can also be written in op-assign form. For example:

AC0 = AC0 + \*AR4

can also be written:

AC0 += \*AR4

This form is supported for these operations: +=, -=, &=, |=, ^=

Note that in certain instances use of op-assign notation results in ambiguous algebraic assembly. This happens if the op-assign operator is not delimited by white space, for example:

```
*AR0+=#4 is ambiguous, is it *AR0 += #4 or *AR0+ = #4?
```

The assembler always parses adjacent += as plus-assign; therefore, this instructions is parsed as \*AR0 += #4.

```
*AR0+=*AR1 is ambiguous, is it *AR0 += *AR1 or *AR0+ =*AR1?
```

Once again, the first form, \*AR0 += \*AR1, is used. This is not a valid instruction — an error is printed.

SPRU375G

#### **1.5 Nonrepeatable Instructions**

Table 1–4 lists the instructions that cannot be used in a repeatable instruction.

Instruction Description	Algebraic Syntax That Cannot Be Repeated
Addition <sup>†</sup>	ACy = ACx + ( <mark>uns(</mark> Smem) << #SHIFTW)
	Smem = Smem + K16
Bitwise AND <sup>†</sup>	Smem = Smem & k16
Bitwise OR <sup>†</sup>	Smem = Smem   k16
Bitwise Exclusive OR (XOR) <sup>†</sup>	Smem = Smem ^ k16
Bitwise AND Memory with Immediate Value and Compare to Zero <sup>†</sup>	TCx = Smem & k16
Branch Conditionally	if (cond) goto I4
	if (cond) goto L8
	if (cond) goto L16
	if (cond) goto P24
Branch Unconditionally	goto ACx
	goto L7
	goto L16
	goto P24
Branch on Auxiliary Register Not Zero	if (ARn_mod != #0) goto L16
Call Conditionally	if (cond) call L16
	if (cond) call P24
Call Unconditionally	call ACx
	call L16
	call P24
Clear Status Register Bit	bit(STx, k4) = #0
Compare and Branch	compare (uns(src RELOP K8)) goto L8
Compare Memory with Immediate Value <sup>†</sup>	TCx = (Smem == K16)

Table 1–4.	Nonrepeatable	Instructions
------------	---------------	--------------

<sup>†</sup> This instruction may not be repeated when using the \*(#k23) absolute addressing mode to access the memory operand Smem.

Instruction Description	Algebraic Syntax That Cannot Be Repeated
Execute Conditionally	if (cond) execute(AD_Unit)
	if (cond) execute(D_Unit)
Idle	idle
Load Accumulator from Memory <sup>†</sup>	ACx = uns(Smem) << #SHIFTW
Load CPU Register from Memory	DP = Smem
	RETA = dbl(Lmem)
Load CPU Register with Immediate Value	DP = k16
Load Memory with Immediate Value <sup>†</sup>	Smem = K16
Move CPU Register Content to Auxiliary or Temporary Register	TAx = RPTC
Multiply <sup>†</sup>	ACx = rnd(Smem * K8)[, T3 = Smem]
Multiply and Accumulate <sup>†</sup>	ACy = rnd(ACx + (Smem * K8))[, T3 = Smem ]
Repeat Block of Instructions Unconditionally	localrepeat{}
	blockrepeat{}
Repeat Single Instruction Conditionally	while (cond && (RPTC < k8)) repeat
Repeat Single Instruction Unconditionally	repeat(k8)
	repeat(k16)
	repeat(CSR)
Repeat Single Instruction Unconditionally and Decrement CSR	repeat(CSR), CSR -= k4
Repeat Single Instruction Unconditionally and	repeat(CSR), CSR += TAx
Increment CSR	repeat(CSR), CSR += k4
Return Conditionally	if (cond) return
Return Unconditionally	return
Return from Interrupt	return_int
Round Accumulator Content	ACy = rnd(ACx)

Table 1-4. Nonrepeatable Instructions (Continued)

<sup>†</sup> This instruction may not be repeated when using the \*(#k23) absolute addressing mode to access the memory operand Smem.

Instruction Description	Algebraic Syntax That Cannot Be Repeated
Set Status Register Bit	bit(STx, k4) = #1
Software Interrupt	intr(k5)
Software Reset	reset
Software Trap	trap(k5)
Store Accumulator Content to Memory <sup>†</sup>	Smem = HI(rnd(ACx << #SHIFTW))
	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))
Store CPU Register Content to Memory	dbl(Lmem) = RETA
Subtraction <sup>†</sup>	ACy = ACx - (uns(Smem) << #SHIFTW)

Table 1-4. Nonrepeatable Instructions (Continued)

<sup>†</sup> This instruction may not be repeated when using the \*(#k23) absolute addressing mode to access the memory operand Smem.

## Chapter 2

# **Parallelism Features and Rules**

This chapter describes the parallelism features and rules of the TMS320C55x<sup>™</sup> DSP algebraic instruction set.

#### Topic

#### Page

2.1	Parallelism Features 2-2
2.2	Parallelism Basics 2-3
2.3	Resource Conflicts 2-4
2.4	Soft-Dual Parallelism 2-5
2.5	Execute Conditionally Instructions 2-6
2.6	Other Exceptions

#### 2.1 Parallelism Features

The C55x<sup>™</sup> DSP architecture enables you to execute two instructions in parallel within the same cycle of execution. The types of parallelism are:

Built-in parallelism within a single instruction.

Some instructions perform two different operations in parallel. A comma is used to separate the two operations. This type of parallelism is also called implied parallelism. For example:

```
AC0 = *AR0 * coef(*CDP),
AC1 = *AR1 * coef(*CDP)
AC1 = *AR1 * coef(*CDP) * coef
```

User-defined parallelism between two instructions.

Two instructions may be paralleled by you or the C compiler. The parallel bars, ||, are used to separate the two instructions to be executed in parallel. For example:

AC1 = \*AR1- \* \*AR2+The first instruction performs a<br/>multiplication in the D-unit. The second<br/>instruction performs a logical operation in<br/>the A-unit ALU.

□ Built-in parallelism can be combined with user-defined parallelism. Parenthesis separators can be used to determine boundaries of the two instructions. For example:

(AC2 = *AR3+ * AC1,	The first instruction includes implied
T3 = *AR3+)	parallelism. The second instruction is
AR1 = #5	paralleled by you.

#### 2.2 Parallelism Basics

In the parallel pair, all of these constraints must be met:

- Total size of both instructions may not exceed 6 bytes.
- □ No resource conflicts as detailed in section 2.3.
- One instruction must have a parallel enable bit or the pair must qualify for soft-dual parallelism as detailed in section 2.4.
- ☐ No memory operand may use an addressing mode that requires a constant that is 16 bits or larger:
  - \*abs16(#k16)
  - \*(#k23)
  - \*port(#k16)
  - \*ARn(K16)
  - \*+ARn(K16)
  - \*CDP(K16)
  - \*+CDP(K16)

The following instructions cannot be in parallel:

- if (cond) goto P24
- if (cond) call P24
- idle
- intr(k5)
- reset
- trap(k5)
- Neither instruction in the parallel pair can use any of these instruction or operand modifiers:
  - circular()
  - linear()
  - mmap()
  - readport()
  - writeport()
- ☐ A particular register or memory location can only be written once per pipeline phase. Violations of this rule take many forms. Loading the same register twice is a simple case. Other cases include:
  - Conflicting address mode modifications (for example, \*AR2+ versus \*AR2-)
  - Combining a SWAP instruction (modifies all of its registers) with any other instruction that writes one of the same registers

- Modifying the data stack pointer (SP) or system stack pointer (SSP) in combination with:
  - all Push to Top of Stack (push) instructions
  - all Pop Top of Stack (pop) instructions
  - all Call Conditionally, if (cond) call; and Call Unconditionally, call, instructions
  - all Return Conditionally, if (cond) return; Return Unconditionally, return; and Return from Interrupt, return\_int, instructions
  - trap and intr instructions
- When both instructions in a parallel pair modify a status bit, the value of that status bit becomes undefined.

#### 2.3 Resource Conflicts

Every instruction uses some set of operators, address generation units, and buses, collectively called resources, while executing. To determine which resources are used by a specific instruction, see Table 4–1. Two instructions in parallel use all the resources of the individual instructions. A resource conflict occurs when two instructions use a combination of resources that is not supported on the C55x device. This section details the resource conflicts.

#### 2.3.1 Operators

You may use each of these operators only once:

- D Unit ALU
- D Unit Shift
- D Unit Swap
- A Unit Swap
- A Unit ALU
- 🗋 P Unit

For an instruction that uses multiple operators, any other instruction that uses one or more of those same operators may not be placed in parallel.

#### 2.3.2 Address Generation Units

You may use no more than the indicated number of data address generation units:

- **2** Data Address (DA) Generation Units
- 1 Coefficient Address (CA) Generation Unit
- 1 Stack Address (SA) Generation Unit

#### 2.3.3 Buses

You may use no more than the indicated number of buses:

- 2 Data Read (DR) Buses
- 1 Coefficient Read (CR) Bus
- 2 Data Write (DW) Buses
- □ 1 ACB Bus brings D-unit registers to A-unit and P-unit operators
- 1 KAB Bus Constant Bus
- 1 KDB Bus Constant Bus

# 2.4 Soft-Dual Parallelism

Instructions that reference memory operands do not have parallel enable bits. Two such instructions may still be combined with a type of parallelism called soft-dual parallelism. The constraints of soft-dual parallelism are:

Both memory operands must meet the constraints of the dual AR indirect addressing mode (Xmem and Ymem), as described in section 3.4.2. The operands available for the dual AR indirect addressing mode are:

- \*ARn
- \*ARn+
- \*ARn-
- \*(ARn + AR0)
- \*(ARn + T0)
- \*(ARn AR0)
- \*(ARn T0)
- \*ARn(AR0)
- \*ARn(T0)
- \*(ARn + T1)
- \*(ARn T1)

Neither instruction can contain any of the following:

- Instructions embedding high\_byte(Smem) and low\_byte(Smem).
  - dst = uns(high\_byte(Smem))
  - dst = uns(low\_byte(Smem))
  - ACx = low\_byte(Smem) << #SHIFTW</pre>
  - ACx = high\_byte(Smem) << #SHIFTW</p>
  - high\_byte(Smem) = src
  - low\_byte(Smem) = src

■ These instructions that read and write the same memory location:

- cbit(Smem, src)
- bit(Smem, src) = #0
- bit(Smem, src) = #1
- TCx = bit(Smem, k4), bit(Smem, k4) = #1
- TCx = bit(Smem, k4), bit(Smem, k4) = #0
- TCx = bit(Smem, k4), cbit(Smem, k4)
- □ With regard to soft-dual parallelism, the mar(Smem) instruction has the same properties as any memory reference instruction.

#### 2.4.1 Soft-Dual Parallelism of MAR Instructions

Although the following modify auxiliary register (MAR) instructions do not reference memory and do not have parallel enable bits, they may be combined together or with any other memory reference instructions (not limited to Xmem/ Ymem) to form soft-dual parallelism.

mar(TAy + TAx)
mar(TAx + k8)
mar(TAy = TAx)
mar(TAx = k8)
mar(TAy - TAx)
mar(TAx - k8)

Note that this is not the full list of MAR instructions; instructions mar(TAx = D16) and mar(Smem) are not included.

## 2.5 Execute Conditionally Instructions

The parallelization of the execute conditionally, if (cond) execute, instructions does not adhere to the descriptions in this chapter. All of the specific instances of legal parallelism are covered in the execute conditionally descriptions in Chapter 5.

# 2.6 Other Exceptions

The following are other exceptions not covered elsewhere in this chapter.

- □ These instructions, when k4 is a value of 0–8, change the value of the XDP register:
  - bit(ST0, k4) = #1
    bit(ST0, k4) = #0

Therefore, they may not be combined with any of these load-the-DP instructions:

- DP = Smem
- XDP = dbl(Lmem)
  XDP = popboth()
- An instruction that reads the repeat counter register (RPTC) may not be combined with any single-repeat instruction:
  - repeat()
  - repeat(CSR)
  - while (cond) repeat

# Chapter 3

# **Introduction to Addressing Modes**

This chapter provides an introduction to the addressing modes of the TMS320C55x<sup>TM</sup> DSP.

#### Topic

#### Page

3.1	Introduction to the Addressing Modes 3-2
3.2	Absolute Addressing Modes 3-3
3.3	Direct Addressing Modes 3-4
3.4	Indirect Addressing Modes 3-6
3.5	Circular Addressing 3-20

# 3.1 Introduction to the Addressing Modes

The TMS320C55x DSP supports three types of addressing modes that enable flexible access to data memory, to memory-mapped registers, to register bits, and to I/O space:

- ☐ The absolute addressing mode allows you to reference a location by supplying all or part of an address as a constant in an instruction.
- ☐ The direct addressing mode allows you to reference a location using an address offset.
- The indirect addressing mode allows you to reference a location using a pointer.

Each addressing mode provides one or more types of operands. An instruction that supports an addressing-mode operand has one of the following syntax elements listed in Table 3–1.

Table 3–1. Addressing-Mode Operands

Syntax Element(s)	Description
Baddr	When an instruction contains Baddr, that instruction can access one or two bits in an accumulator (AC0–AC3), an auxiliary register (AR0–AR7), or a temporary register (T0–T3). Only the register bit test/set/clear/complement instructions support Baddr. As you write one of these instructions, replace Baddr with a compatible operand.
Cmem	When an instruction contains Cmem, that instruction can access a single word (16 bits) of data from data memory. As you write the instruction, replace Cmem with a compatible operand.
Lmem	When an instruction contains Lmem, that instruction can access a long word (32 bits) of data from data memory or from a memory-mapped registers. As you write the instruction, replace Lmem with a compatible operand.
Smem	When an instruction contains Smem, that instruction can access a single word (16 bits) of data from data memory, from I/O space, or from a memory-mapped register. As you write the instruction, replace Smem with a compatible operand.
Xmem and Ymem	When an instruction contains Xmem and Ymem, that instruction can perform two simultaneous 16-bit accesses to data memory. As you write the instruction, replace Xmem and Ymem with compatible operands.

# 3.2 Absolute Addressing Modes

Table 3–2 lists the absolute addressing modes available.

Table 3–2. Absolute Addressing Modes

Addressing Mode	Description
k16 absolute	This mode uses the 7-bit register called DPH (high part of the extended data page register) and a 16-bit unsigned constant to form a 23-bit data-space address. This mode is used to access a memory location or a memory-mapped register.
k23 absolute	This mode enables you to specify a full address as a 23-bit unsigned constant. This mode is used to access a memory location or a memory-mapped register.
I/O absolute	This mode enables you to specify an I/O address as a 16-bit unsigned constant. This mode is used to access a location in I/O space.

#### 3.2.1 k16 Absolute Addressing Mode

The k16 absolute addressing mode uses the operand \*abs16(#k16), where k16 is a 16-bit unsigned constant. DPH (the high part of the extended data page register) and k16 are concatenated to form a 23-bit data-space address. An instruction using this addressing mode encodes the constant as a 2-byte extension to the instruction. Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction.

#### 3.2.2 k23 Absolute Addressing Mode

The k23 absolute addressing mode uses the \*(#k23) operand, where k23 is a 23-bit unsigned constant. An instruction using this addressing mode encodes the constant as a 3-byte extension to the instruction (the most-significant bit of this 3-byte extension is discarded). Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction.

Instructions using the operand (#k23) to access the memory operand Smem cannot be used in a repeatable instruction. See Table 1–4 for a list of these instructions.

## 3.2.3 I/O Absolute Addressing Mode

The I/O absolute addressing mode uses the \*port(#k16) operand, where k16 is a 16-bit unsigned constant. An instruction using this addressing mode encodes the constant as a 2-byte extension to the instruction. Because of the extension, an instruction using this mode cannot be executed in parallel with another instruction. The delay() instruction cannot use this mode.

# 3.3 Direct Addressing Modes

Table 3–3 lists the direct addressing modes available.

Table 3–3. Direct Addressing Modes

Addressing Mode	Description
DP direct	This mode uses the main data page specified by DPH (high part of the extended data page register) in conjunction with the data page register (DP). This mode is used to access a memory location or a memory-mapped register.
SP direct	This mode uses the main data page specified by SPH (high part of the extended stack pointers) in conjunction with the data stack pointer (SP). This mode is used to access stack values in data memory.
Register-bit direct	This mode uses an offset to specify a bit address. This mode is used to access one register bit or two adjacent register bits.
PDP direct	This mode uses the peripheral data page register (PDP) and an offset to specify an I/O address. This mode is used to access a location in I/O space.
	The DD disect and CD direct addressing reader are restually evolveing. The

The DP direct and SP direct addressing modes are mutually exclusive. The mode selected depends on the CPL bit in status register ST1\_55:

CPL	Addressing Mode Selected
0	DP direct addressing mode
1	SP direct addressing mode

The register-bit and PDP direct addressing modes are independent of the CPL bit.

#### 3.3.1 DP Direct Addressing Mode

When an instruction uses the DP direct addressing mode, a 23-bit address is formed. The 7 MSBs are taken from DPH that selects one of the 128 main data pages (0 through 127). The 16 LSBs are the sum of two values:

- ☐ The value in the data page register (DP). DP identifies the start address of a 128-word local data page within the main data page. This start address can be any address within the selected main data page.
- □ A 7-bit offset (Doffset) calculated by the assembler. The calculation depends on whether you are accessing data memory or a memory-mapped register (using the mmap() qualifier).

The concatenation of DPH and DP is called the extended data page register (XDP). You can load DPH and DP individually, or you can use an instruction that loads XDP.

## 3.3.2 SP Direct Addressing Mode

When an instruction uses the SP direct addressing mode, a 23-bit address is formed. The 7 MSBs are taken from SPH. The 16 LSBs are the sum of the SP value and a 7-bit offset that you specify in the instruction. The offset can be a value from 0 to 127. The concatenation of SPH and SP is called the extended data stack pointer (XSP). You can load SPH and SP individually, or you can use an instruction that loads XSP.

On the first main data page, addresses 00 0000h–00 005Fh are reserved for the memory-mapped registers. If any of your data stack is in main data page 0, make sure it uses only addresses 00 0060h–00 FFFFh on that page.

#### 3.3.3 Register-Bit Direct Addressing Mode

In the register-bit direct addressing mode, the offset you supply in the operand, @bitoffset, is an offset from the LSB of the register. For example, if bitoffset is 0, you are addressing the LSB of a register. If bitoffset is 3, you are addressing bit 3 of the register.

Only the register bit test/set/clear/complement instructions support this mode. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).

#### 3.3.4 PDP Direct Addressing Mode

When an instruction uses the PDP direct addressing mode, a 16-bit I/O address is formed. The 9 MSBs are taken from the 9-bit peripheral data page register (PDP) that selects one of the 512 peripheral data pages (0 through 511). Each page has 128 words (0 to 127). You select a particular word by specifying a 7-bit offset (Poffset) in the instruction. For example, to access the first word on a page, use an offset of 0.

You must use a readport() or writeport() instruction qualifier to indicate that you are accessing an I/O-space location rather than a data-memory location. You place the readport() or the writeport() instruction qualifier in parallel with the instruction that performs the I/O-space access.

# 3.4 Indirect Addressing Modes

Table 3–4 list the indirect addressing modes available. You may use these modes for linear addressing or circular addressing.

Table 3–4. Indirect Addressing Modes

Addressing Mode	Description
AR indirect	This mode uses one of eight auxiliary registers (AR0–AR7) to point to data. The way the CPU uses the auxiliary register to generate an address depends on whether you are accessing data space (memory or memory-mapped registers), individual register bits, or I/O space.
Dual AR indirect	This mode uses the same address-generation process as the AR indirect addressing mode. This mode is used with instructions that access two or more data-memory locations.
CDP indirect	This mode uses the coefficient data pointer (CDP) to point to data. The way the CPU uses CDP to generate an address depends on whether you are accessing data space (memory or memory-mapped registers), individual register bits, or I/O space.
Coefficient indirect	This mode uses the same address-generation process as the CDP indirect addressing mode. This mode is available to support instructions that can access a coefficient in data memory at the same time they access two other data-memory values using the dual AR indirect addressing mode.

#### 3.4.1 AR Indirect Addressing Mode

The AR indirect addressing mode uses an auxiliary register ARn (n = 0, 1, 2, 3, 4, 5, 6, or 7) to point to data. The way the CPU uses ARn to generate an address depends on the access type:

For An Access To	ARn Contains
Data space (memory or registers)	The 16 least significant bits (LSBs) of a 23-bit address. The 7 most significant bits (MSBs) are supplied by ARnH, which is the high part of extended auxiliary register XARn. For accesses to data space, use an instruction that loads XARn; ARn can be individually loaded, but ARnH cannot be loaded.
A register bit (or bit pair)	A bit number. Only the register bit test/set/clear/com- plement instructions support AR indirect accesses to register bits. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).
I/O space	A 16-bit I/O address.

The AR indirect addressing-mode operand available depends on the ARMS bit of status register ST2\_55:

ARMS	DSP Mode or Control Mode
0	DSP mode. The CPU can use the list of DSP mode operands (Table 3–5), which provide efficient execution of DSP-intensive applications.
1	Control mode. The CPU can use the list of control mode operands (Table 3–6), which enable optimized code size for control system applications.

Table 3–5 (page 3-8) introduces the DSP operands available for the AR indirect addressing mode. Table 3–6 (page 3-12) introduces the control mode operands. When using the tables, keep in mind that:

- Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the appropriate 16-bit buffer start address register (BSA01, BSA23, BSA45, or BSA67) is added only if circular addressing is activated for the chosen pointer.
- All additions to and subtractions from the pointers are done modulo 64K. You cannot address data across main data pages without changing the value in the extended auxiliary register (XARn).

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn+	ARn is incremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn–	ARn is decremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*+ARn	ARn is incremented before the address is generated: If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*–ARn	ARn is decremented before the address is generated: If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after	Data-memory (Smem, Lmem)
	the address is generated: ARn = ARn + AR0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode

Operand	Pointer Modification	Supported Access Types
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn	Data-memory (Smem, Lmem)
	after the address is generated: ARn = ARn – T0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in AR0 is used as an offset from that base pointer. This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*ARn(T1)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T1 is used as an offset from that base pointer.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn + T1)	The 16-bit signed constant in T1 is added to ARn after the address is generated: ARn = ARn + T1	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn – T1)	The 16-bit signed constant in T1 is subtracted from ARn	Data-memory (Smem, Lmem)
	after the address is generated: ARn = ARn – T1	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + AR0B)	The 16-bit signed constant in AR0 is added to ARn after	Data-memory (Smem, Lmem)
	the address is generated: ARn = ARn + AR0 (The addition is done with reverse carry propagation) This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time. Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + T0B)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0 (The addition is done with reverse carry propagation) This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
	Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*(ARn – AR0B)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = $ARn - AR0$ (The subtraction is done with reverse carry propagation)	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr) I/O-space (Smem)
	Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	
*(ARn – T0B)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: ARn = ARn $-$ T0 (The subtraction is done with reverse carry propagation)	Data-memory (Smem, Lmem) Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	Register bit (Baddr) I/O-space (Smem)
	Note: When this bit-reverse operand is used, ARn cannot be used as a circular pointer. If ARn is configured in ST2_55 for circular addressing, the corresponding buffer start address register value (BSAxx) is added to ARn, but ARn is not modified so as to remain inside a circular buffer.	
*ARn(#K16)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that	Data-memory (Smem, Lmem) Memory-mapped register
	base pointer. Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Be- cause of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	(Smem, Lmem) Register bit (Baddr)
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before the address is generated: ARn = ARn + K16	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Be- cause of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register bit (Baddr)

Table 3–5. DSP Mode Operands for the AR Indirect Addressing Mode (Continued)

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn+	ARn is incremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn + 1 If 32-bit/2-bit operation: ARn = ARn + 2	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*ARn–	ARn is decremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: ARn = ARn – 1 If 32-bit/2-bit operation: ARn = ARn – 2	Memory-mapped register Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after	Data-memory (Smem, Lmem)
	the address is generated: ARn = ARn + AR0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	Register bit (Baddr)
		I/O-space (Smem)
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after	Data-memory (Smem, Lmem)
	the address is generated: ARn = ARn + T0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from	Data-memory (Smem, Lmem)
	ARn after the address is generated: ARn = ARn – AR0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)

Table 3–6. Control Mode Operands for the AR Indirect Addressing Mode

Operand	Pointer Modification	Supported Access Types
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn	Data-memory (Smem, Lmem)
	after the address is generated: ARn = ARn – T0	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant in AR0 is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant in T0 is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at	Register bit (Baddr)
	assembly time.	I/O-space (Smem)
*ARn(#K16)	ARn is not modified. ARn is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant (K16) is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register bit (Baddr)
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn	Data-memory (Smem, Lmem)
	before the address is generated: ARn = ARn + K16	Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register bit (Baddr)
*ARn(short(#k3))	ARn is not modified. ARn is used as a base pointer. The	Data-memory (Smem, Lmem)
	3-bit unsigned constant (k3) is used as an offset from that base pointer. k3 is in the range 1 to 7.	Memory-mapped register (Smem, Lmem)
		Register bit (Baddr)
		I/O-space (Smem)

Table 3–6. Control Mode Operands for the AR Indirect Addressing Mode (Continued)

#### 3.4.2 Dual AR Indirect Addressing Mode

The dual AR indirect addressing mode enables you to make two data-memory accesses through the eight auxiliary registers, AR0–AR7. As with single AR indirect accesses to data space, the CPU uses an extended auxiliary register to create each 23-bit address. You can use linear addressing or circular addressing for each of the two accesses.

You may use the dual AR indirect addressing mode for:

Executing an instruction that makes two 16-bit data-memory accesses. In this case, the two data-memory operands are designated in the instruction syntax as Xmem and Ymem. For example:

ACx = (Xmem << #16) + (Ymem << #16)

Executing two instructions in parallel. In this case, both instructions must each access a single memory value, designated in the instruction syntaxes as Smem or Lmem. For example:

```
dst = Smem
|| dst = src & Smem
```

The operand of the first instruction is treated as an Xmem operand, and the operand of the second instruction is treated as a Ymem operand.

The available dual AR indirect operands are a subset of the AR indirect operands. The ARMS status bit does not affect the set of dual AR indirect operands available.

Note:

The assembler rejects code in which dual operands use the same auxiliary register with two different auxiliary register modifications. You can use the same ARn for both operands, if one of the operands is \*ARn or \*ARn(T0); neither modifies ARn.

Table 3–7 (page 3-15) introduces the operands available for the dual AR indirect addressing mode. Note that:

- Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the appropriate 16-bit buffer start address register (BSA01, BSA23, BSA45, or BSA67) is added only if circular addressing is activated for the chosen pointer.
- All additions to and subtractions from the pointers are done modulo 64K. You cannot address data across main data pages without changing the value in the extended auxiliary register (XARn).

Operand	Pointer Modification	Supported Access Types
*ARn	ARn is not modified.	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn+	ARn is incremented after the address is generated: If 16-bit operation: ARn = ARn + 1 If 32-bit operation: ARn = ARn + 2	Data-memory (Smem, Lmem, Xmem, Ymem)
*ARn–	ARn is decremented after the address is generated: If 16-bit operation: ARn = ARn – 1 If 32-bit operation: ARn = ARn – 2	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn + AR0)	The 16-bit signed constant in AR0 is added to ARn after the address is generated: ARn = ARn + AR0	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*(ARn + T0)	The 16-bit signed constant in T0 is added to ARn after the address is generated: ARn = ARn + T0	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	
*(ARn – AR0)	The 16-bit signed constant in AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*(ARn – T0)	The 16-bit signed constant in T0 is subtracted from ARn after the address is generated: $ARn = ARn - T0$	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in AR0 is used as an offset from that base pointer.	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	

Table 3–7. Dual AR Indirect Operands

Operand	Pointer Modification	Supported Access Types
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. The 16-bit signed constant in T0 is used as an offset from that base pointer.	Data-memory (Smem, Lmem, Xmem, Ymem)
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	
*(ARn + T1)	The 16-bit signed constant in T1 is added to ARn after the address is generated: ARn = ARn + T1	Data-memory (Smem, Lmem, Xmem, Ymem)
*(ARn – T1)	The 16-bit signed constant in T1 is subtracted from ARn after the address is generated: $ARn = ARn - T1$	Data-memory (Smem, Lmem, Xmem, Ymem)

# 3.4.3 CDP Indirect Addressing Mode

The CDP indirect addressing mode uses the coefficient data pointer (CDP) to point to data. The way the CPU uses CDP to generate an address depends on the access type:

For An Access To	CDP Contains
Data space (memory or registers)	The 16 least significant bits (LSBs) of a 23-bit address. The 7 most significant bits (MSBs) are supplied by CDPH, the high part of the extended coefficient data pointer (XCDP).
A register bit (or bit pair)	A bit number. Only the register bit test/set/clear/com- plement instructions support CDP indirect accesses to register bits. These instructions enable you to access bits in the following registers only: the accumulators (AC0–AC3), the auxiliary registers (AR0–AR7), and the temporary registers (T0–T3).
I/O space	A 16-bit I/O address.

Table 3–8 (page 3-17) introduces the operands available for the CDP indirect addressing mode. Note that:

Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the 16-bit buffer start address register BSAC is added only if circular addressing is activated for CDP. □ All additions to and subtractions from CDP are done modulo 64K. You cannot address data across main data pages without changing the value of CDPH (the high part of the extended coefficient data pointer).

Operand	Pointer Modification	Supported Access Types
*CDP	CDP is not modified.	Data-memory (Smem, Lmem)
		Memory-mapped register (Smem, Lmem)
		Register-bit (Baddr)
		I/O-space (Smem)
*CDP+	CDP is incremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: $CDP = CDP + 1$ If 32-bit/2-bit operation: $CDP = CDP + 2$	Memory-mapped register (Smem, Lmem)
		Register-bit (Baddr)
		I/O-space (Smem)
*CDP-	CDP is decremented after the address is generated:	Data-memory (Smem, Lmem)
	If 16-bit/1-bit operation: $CDP = CDP - 1$ If 32-bit/2-bit operation: $CDP = CDP - 2$	Memory-mapped register (Smem, Lmem)
		Register-bit (Baddr)
		I/O-space (Smem)
*CDP(#K16)	CDP is not modified. CDP is used as a base pointer. The	Data-memory (Smem, Lmem)
	16-bit signed constant (K16) is used as an offset from that base pointer.	Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register-bit (Baddr)
*+CDP(#K16)	The 16-bit signed constant (K16) is added to CDP before	Data-memory (Smem, Lmem)
	the address is generated: CDP = CDP + K16	Memory-mapped register (Smem, Lmem)
	Note: When an instruction uses this operand, the constant is encoded in a 2-byte extension to the instruction. Be- cause of the extension, an instruction using this operand cannot be executed in parallel with another instruction.	Register-bit (Baddr)

Table 3–8. CDP Indirect Operands

#### 3.4.4 Coefficient Indirect Addressing Mode

The coefficient indirect addressing mode uses the same address-generation process as the CDP indirect addressing mode for data-space accesses. The coefficient indirect addressing mode is supported by select memory-to-memory move and memory initialization instructions and by the following arithmetical instructions:

- Dual multiply (accumulate/subtract)
- Finite impulse response filter
- Multiply
- Multiply and accumulate
- Multiply and subtract

Instructions using the coefficient indirect addressing mode to access data are mainly instructions performing operations with three memory operands per cycle. Two of these operands (Xmem and Ymem) are accessed with the dual AR indirect addressing mode. The third operand (Cmem) is accessed with the coefficient indirect addressing mode. The Cmem operand is carried on the BB bus.

Keep the following facts about the BB bus in mind as you use the coefficient indirect addressing mode:

- The BB bus is not connected to external memory. If a Cmem operand is accessed through the BB bus, the operand must be in internal memory.
- Although the following instructions access Cmem operands, they do not use the BB bus to fetch the 16-bit or 32-bit Cmem operand.

Instruction Syntax	Description of Cmem Access	Bus Used to Access Cmem
Smem = Cmem	16-bit read from Cmem	DB
Cmem = Smem	16-bit write to Cmem	EB
Lmem = dbl(Cmem)	32-bit read from Cmem	CB for most significant word (MSW) DB for least significant word (LSW)
dbl(Cmem) = Lmem	32-bit write to Cmem	FB for MSW EB for LSW

Consider the following instruction syntax. In one cycle, two multiplications can be performed in parallel. One memory operand (Cmem) is common to both multiplications, while dual AR indirect operands (Xmem and Ymem) are used for the other values in the multiplication.

ACx = Xmem \* Cmem, ACy = Ymem \* Cmem

To access three memory values (as in the above example) in a single cycle, the value referenced by Cmem must be located in a memory bank different from the one containing the Xmem and Ymem values.

Table 3–9 introduces the operands available for the coefficient indirect addressing mode. Note that:

- Both pointer modification and address generation are linear or circular according to the pointer configuration in status register ST2\_55. The content of the 16-bit buffer start address register BSAC is added only if circular addressing is activated for CDP.
- All additions to and subtractions from CDP are done modulo 64K. You cannot address data across main data pages without changing the value of CDPH (the high part of the extended coefficient data pointer).

Operand	Pointer Modification	Supported Access Type
*CDP	CDP is not modified.1	Data-memory
*CDP+	CDP is incremented after the address is generated: If 16-bit operation: CDP = CDP + 1 If 32-bit operation: CDP = CDP + 2	Data-memory
*CDP-	CDP is decremented after the address is generated: If 16-bit operation: CDP = CDP $- 1$ If 32-bit operation: CDP = CDP $- 2$	Data-memory
*(CDP + AR0)	The 16-bit signed constant in AR0 is added to CDP after the address is generated: CDP = CDP + AR0	Data-memory
	This operand is available when C54CM = 1. This operand is usable when .c54cm_on is active at assembly time.	
*(CDP + T0)	The 16-bit signed constant in T0 is added to CDP after the address is generated: CDP = CDP + T0	Data-memory
	This operand is available when C54CM = 0. This operand is usable when .c54cm_off is active at assembly time.	

Table 3–9. Coefficient Indirect Operands

#### 3.5 Circular Addressing

Circular addressing can be used with any of the indirect addressing modes. Each of the eight auxiliary registers (AR0–AR7) and the coefficient data pointer (CDP) can be independently configured to be linearly or circularly modified as they act as pointers to data or to register bits, see Table 3–10. This configuration is done with a bit (ARnLC) in status register ST2\_55. To choose circular modification, set the bit.

Pointer	Linear/Circular Configuration Bit	Supplier of Main Data Page	Buffer Start Address Register	Buffer Size Register
AR0	ST2_55(0) = AR0LC	AR0H	BSA01	BK03
AR1	ST2_55(1) = AR1LC	AR1H	BSA01	BK03
AR2	ST2_55(2) = AR2LC	AR2H	BSA23	BK03
AR3	ST2_55(3) = AR3LC	AR3H	BSA23	BK03
AR4	ST2_55(4) = AR4LC	AR4H	BSA45	BK47
AR5	ST2_55(5) = AR5LC	AR5H	BSA45	BK47
AR6	ST2_55(6) = AR6LC	AR6H	BSA67	BK47
AR7	ST2_55(7) = AR7LC	AR7H	BSA67	BK47
CDP	ST2_55(8) = CDPLC	CDPH	BSAC	BKC

Table 3–10. Circular Addressing Pointers

Each auxiliary register ARn has its own linear/circular configuration bit in ST2\_55:

ARnLC	ARn Is Used For
0	Linear addressing
1	Circular addressing

The CDPLC bit in status register ST2\_55 configures the DSP to use CDP for linear addressing or circular addressing:

CDPLC	CDP Is Used For
0	Linear addressing
1	Circular addressing

You can use the circular addressing instruction qualifier, circular(), if you want every pointer used by the instruction to be modified circularly, just add the circular() qualifier in parallel with the instruction. The circular addressing instruction qualifier overrides the linear/circular configuration in ST2\_55.

# **Chapter 4**

# **Instruction Set Summary**

This chapter provides a summary of the TMS320C55x<sup>™</sup> DSP algebraic instruction set (Table 4–1). With each instruction, you will find the availability of a parallel enable bit, word count (size), cycle time, what pipeline phase the instruction executes, in what operator unit the instruction executes, how many of each address generation unit is used, and how many of each bus is used.

Table 4–1 does not list all of the resources that may be used by an instruction, it only lists those that may result in a resource conflict, and thus prevent two instructions from being in parallel. If an instruction lists nothing in a particular column, it means that particular resource will never be in conflict for that instruction.

The column heads of Table 4–1 are:

- Instruction: In cases where the resource usage of an instruction varies with the kinds of registers, you see the notation <name>-AU for A-unit registers and <name>-DU for D-unit registers. So, dst-AU is a destination that is an A-unit register and src-DU is a source that is a D-unit register. In the few cases where that notation is insufficient, you see the cases listed in the Notes column.
- E: Whether that instruction has a parallel enable bit
- S: The size of the instruction in bytes
- C: Number of cycles required for the instruction
- Pipe: The pipeline phase in which the instruction executes:

Name	Phase
AD	Address
D	Decode
R	Read
Х	Execute

Operator: Which operator(s) are used by this instruction. When an instruction uses multiple operators, any other instruction that uses one or more of those same operators may not be placed in parallel.

Address Generation Unit: How many of each address generation unit is
used. The address generation units are:

Name	Unit
DA	Data Address Generation Unit
CA	Coefficient Address Generation Unit
SA	Stack Address Generation Unit

Buses: How many of each bus is used. The buses are:

Name	Bus
DR	Data Read
CR	Coefficient Read
DW	Data Write
ACB	Brings D unit registers to A unit and P unit operators
KAB	Constants
KDB	Constants

Table 4–1	Algebraic	Instruction	Set Summary
	7 ligebiule	1130 00001	Oct Ourminary

								ddres					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Ab	solute Distance (page 5-2)															
	abdst(Xmem, Ymem, ACx, ACy)	Ν	4	1	х	DU_ALU	2			2						
Ab	solute Value (page 5-4)									-						
	dst-AU =  src-AU	Y	2	1	х	AU_ALU				.						1
	dst-AU =  src-DU	Y	2	1	х	AU_ALU							1			
	dst-DU =  src	Y	2	1	х	DU_ALU										See Note
Ad	dition (page 5-7)	•					•			•						
[1]	dst-AU = dst-AU + src-AU	Y	2	1	х	AU_ALU										
	dst-AU = dst-AU + src-DU	Y	2	1	х	AU_ALU							1			
	dst-DU = dst-DU + src	Y	2	1	х	DU_ALU										See Note
[2]	dst-AU = dst-AU + k4	Y	2	1	х	AU_ALU									1	
	dst-DU = dst-DU + k4	Y	2	1	х	DU_ALU									1	
[3]	dst-AU = src-AU + K16	Ν	4	1	х	AU_ALU									1	
	dst-AU = src-DU + K16	Ν	4	1	х	AU_ALU		•					1		1	
	dst-DU = src + K16	Ν	4	1	х	DU_ALU									1	See Note
[4]	dst-AU = src-AU + Smem	Ν	3	1	х	AU_ALU	1			1						
	dst-AU = src-DU + Smem	Ν	3	1	х	AU_ALU	1	•		1			1	•		
	dst-DU = src + Smem	Ν	3	1	х	DU_ALU	1	•		1	•	•				See Note
[5]	ACy = ACy + (ACx << Tx)	Y	2	1	х	DU_ALU + DU_SHIFT		•			·					
[6]	ACy = ACy + (ACx << #SHIFTW)	Y	3	1	х	DU_ALU + DU_SHIFT		•								
[7]	ACy = ACx + (K16 << #16)	Ν	4	1	х	DU_ALU									1	
[8]	ACy = ACx + (K16 << #SHFT)	Ν	4	1	х	DU_ALU + DU_SHIFT		•			·			·	1	
[9]	ACy = ACx + (Smem << Tx)	N	3	1	х	DU_ALU + DU_SHIFT	1			1				·		

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Instruction	
Set	
Summary	

								Addres eration					Buses			
No.	Instruction	E	S	с	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Note
[10]	ACy = ACx + (Smem << #16)	Ν	3	1	х	DU_ALU	1			1	·					
[11]	ACy = ACx + uns(Smem) + CARRY	Ν	3	1	х	DU_ALU	1			1						
[12]	ACy = ACx + uns(Smem)	N	3	1	х	DU_ALU	1			1						
[13]	ACy = ACx + (uns(Smem) << #SHIFTW)	Ν	4	1	х	DU_ALU + DU_SHIFT	1			1						
[14]	ACy = ACx + dbl(Lmem)	N	3	1	х	DU_ALU	1			2			•	•		
[15]	ACx = (Xmem << #16) + (Ymem << #16)	Ν	3	1	х	DU_ALU	2			2						
[16]	Smem = Smem + K16	Ν	4	1	х	DU_ALU	1			1		1			1	
Ado	dition with Absolute Value (page 5-27)	•					•									
	ACy = rnd(ACy +  ACx )	Y	2	1	х	DU_ALU										
Ado	dition with Parallel Store Accumulator Co	ntent to Me	mor	<b>y</b> (pa	ige 5-2	29)				•						•
	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	N	4	1	х	DU_ALU + DU_SHIFT	2			2		2				
Ado	dition or Subtraction Conditionally (page 5	5-31)														
[1]	ACy = adsc(Smem, ACx, TC1)	Ν	3	1	х	DU_ALU	1			1						
[2]	ACy = adsc(Smem, ACx, TC2)	Ν	3	1	х	DU_ALU	1			1						
Ado	dition or Subtraction Conditionally with S	hift (page 5-	33)				1			I						1
	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	Ν	3	1	х	DU_ALU + DU_SHIFT	1			1						
Ado	dition, Subtraction, or Move Accumulator	Content Co	ndi	tiona	<b>illy</b> (pa	age 5-36)				•						•
	ACy = adsc(Smem, ACx, TC1, TC2)	N	3	1	х	DU_ALU	1			1						

2) dst-DU, src-AU or dst-AU, src-DU

Table 4–1. Algebraic Instruction Set Summary (Continued)

								Addres eration					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Bitv	wise AND (page 5-38)															
[1]	dst-AU = dst-AU & src-AU	Y	2	1	х	AU_ALU				.						
	dst-AU = dst-AU & src-DU	Y	2	1	х	AU_ALU							1			
	dst-DU = dst-DU & src	Y	2	1	х	DU_ALU										See Note 1
[2]	dst-AU = src-AU & k8	Y	3	1	х	AU_ALU									1	
	dst-AU = src-DU & k8	Y	3	1	х	AU_ALU							1		1	
	dst-DU = src & k8	Y	3	1	х	DU_ALU									1	See Note 1
[3]	dst-AU = src-AU & k16	Ν	4	1	х	AU_ALU									1	
	dst-AU = src-DU & k16	Ν	4	1	х	AU_ALU							1		1	
	dst-DU = src & k16	Ν	4	1	х	DU_ALU									1	See Note 1
[4]	dst-AU = src-AU & Smem	Ν	3	1	х	AU_ALU	1			1						
	dst-AU = src-DU & Smem	Ν	3	1	х	AU_ALU	1			1			1			
	dst-DU = src & Smem	Ν	3	1	х	DU_ALU	1			1						See Note 1
[5]	ACy = ACy & (ACx <<< #SHIFTW)	Y	3	1	х	DU_ALU + DU_SHIFT	•	•	•			•				
[6]	ACy = ACx & (k16 <<< #16)	N	4	1	х	DU_ALU									1	
[7]	ACy = ACx & (k16 <<< #SHFT)	Ν	4	1	х	DU_ALU + DU_SHIFT			•						1	
[8]	Smem = Smem & k16	Ν	4	1	х	AU_ALU	1			1		1			1	
Bitv	wise AND Memory with Immediate Valu	le and Compare	e to	Zere	<b>o</b> (pag	e 5-47)				•						
[1]	TC1 = Smem & k16	Ν	4	1	х	AU_ALU	1			1					1	
[2]	TC2 = Smem & k16	Ν	4	1	х	AU_ALU	1			1					1	

# Table 4-1 Algebraic Instruction Set Summary (Continued)

dst-DU, src-AU or dst-DU, src-DU Notes: 1)

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Bit	wise OR (page 5-48)															
[1]	dst-AU = dst-AU   src-AU	Y	2	1	х	AU_ALU										
	dst-AU = dst-AU   src-DU	Y	2	1	х	AU_ALU							1			
	dst-DU = dst-DU   src	Y	2	1	х	DU_ALU										See Note
[2]	dst-AU = src-AU   k8	Y	3	1	х	AU_ALU									1	
	dst-AU = src-DU   k8	Y	3	1	х	AU_ALU							1		1	
	dst-DU = src   k8	Y	3	1	х	DU_ALU									1	See Note
[3]	dst-AU = src-AU   k16	Ν	4	1	х	AU_ALU									1	
	dst-AU = src-DU   k16	Ν	4	1	х	AU_ALU							1		1	
	dst-DU = src   k16	Ν	4	1	х	DU_ALU									1	See Note
[4]	dst-AU = src-AU   Smem	Ν	3	1	х	AU_ALU	1			1						
	dst-AU = src-DU   Smem	Ν	3	1	х	AU_ALU	1			1			1			
	dst-DU = src   Smem	Ν	3	1	х	DU_ALU	1			1						See Note
[5]	ACy = ACy   (ACx <<< #SHIFTW)	Y	3	1	х	DU_ALU + DU_SHIFT										
[6]	ACy = ACx   (k16 <<< #16)	Ν	4	1	х	DU_ALU									1	
[7]	ACy = ACx   (k16 <<< #SHFT)	Ν	4	1	х	DU_ALU + DU_SHIFT		·			·	·	·	·	1	
[8]	Smem = Smem   k16	Ν	4	1	х	AU_ALU	1			1		1			1	

# Table 4–1. Algebraic Instruction Set Summary (Continued)

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Bitv	wise Exclusive OR (XOR) (page 5-57)															
[1]	dst-AU = dst-AU ^ src-AU	Y	2	1	х	AU_ALU				.						
	dst-AU = dst-AU ^ src-DU	Y	2	1	х	AU_ALU							1			
	dst-DU = dst-DU ^ src	Y	2	1	х	DU_ALU										See Note 1
[2]	dst-AU = src-AU ^ k8	Y	3	1	х	AU_ALU									1	
	dst-AU = src-DU ^ k8	Y	3	1	х	AU_ALU							1		1	
	dst-DU = src ^ k8	Y	3	1	х	DU_ALU									1	See Note 1
[3]	dst-AU = src-AU ^ k16	N	4	1	х	AU_ALU									1	
	dst-AU = src-DU ^ k16	N	4	1	х	AU_ALU							1		1	
	dst-DU = src ^ k16	N	4	1	х	DU_ALU									1	See Note 1
[4]	dst-AU = src-AU ^ Smem	N	3	1	х	AU_ALU	1			1						
	dst-AU = src-DU ^ Smem	N	3	1	х	AU_ALU	1			1			1			
	dst-DU = src ^ Smem	N	3	1	х	DU_ALU	1			1						See Note 1
[5]	ACy = ACy ^ (ACx <<< #SHIFTW)	Y	3	1	х	DU_ALU + DU_SHIFT	•									
[6]	ACy = ACx ^ (k16 <<< #16)	N	4	1	х	DU_ALU									1	
[7]	ACy = ACx ^ (k16 <<< #SHFT)	Ν	4	1	х	DU_ALU + DU_SHIFT	•								1	
[8]	Smem = Smem ^ k16	N	4	1	х	AU_ALU	1			1		1			1	
Bra	Inch Conditionally (page 5-66)	•														•
[1]	if (cond) goto I4	N	2	6/5†	R	P_UNIT	.			.						
[2]	if (cond) goto L8	Y	3	6/5†	R	P_UNIT										
[3]	if (cond) goto L16	N	4	6/5†	R	P_UNIT										
[4]	if (cond) goto P24	N	5	5/5†	R	P_UNIT	Ι.			Ι.						

#### Table 4–1. Algebraic Instruction Set Summary (Continued)

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratio					Buses			
No.	Instruction	E	s	с	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Bra	nch Unconditionally (page 5-70)															
[1]	goto ACx	N	2	10	х	P_UNIT							1			
[2]	goto L7	Y	2	6†	AD	P_UNIT										
[3]	goto L16	Y	3	6†	AD	P_UNIT										
[4]	goto P24	N	4	5	D	P_UNIT										
† The	se instructions execute in 3 cycles if the addressed instruction is in	the instruction bu	ffer u	nit.												
Bra	nch on Auxiliary Register Not Zero (page	5-74)														
	if (ARn_mod != #0) goto L16	Ν	4	6/5†	AD	P_UNIT	1			.						
† x/y d	cycles: x cycles = condition true, y cycles = condition false	I					I			1					I	
Call	Conditionally (page 5-77)															
[1]	if (cond) call L16	Ν	4	6/5†	R	P_UNIT	1		1			2				
[2]	if (cond) call P24	N	5	5/5†	R	P_UNIT P_UNIT	1 1		1			2				
† x/y a	cycles: x cycles = condition true, y cycles = condition false	I					1			1					I	
Call	Unconditionally (page 5-83)															
[1]	call ACx	N	2	10	х	P_UNIT	1		1			2	1			
[2]	call L16	Y	3	6	AD	P_UNIT	1		1			2				
[3]	call P24	N	4	5	D	P_UNIT	1		1			2				
Circ	ular Addressing Qualifier (page 5-87)	·														
	circular()	Ν	1	1	AD		.			.						
Clea	ar Accumulator, Auxiliary, or Temporary R	egister Bit	(pa	ge 5-	88)		I			1					1	
	bit(src-AU, Baddr) = #0	-	3	1	x	AU_ALU	1			.						
	bit(src-DU, Baddr) = #0	N	3	1	х	DU_ALU	1									
Clea	ar Memory Bit (page 5-89)	I					I			I					I	
	bit(Smem, src) = #0	N	3	1	х	AU ALU	1			1		1				

Instruction Set Summary

SPRU375G

2) dst-DU, src-AU or dst-AU, src-DU

	. Instruction						Address Generation Unit						Buses			
No.		E	s	с	Pipe	Operator	DA	CA		DR	CR	DW	ACB	KAB	KDB	Notes
Cle	ear Status Register Bit (page 5-90)	•														
[1]	bit(ST0, k4) = #0	Y	2	1	х	AU_ALU	.			.					1	
[2]	bit(ST1, k4) = #0	Y	2	1	х	AU_ALU									1	
[3]	bit(ST2, k4) = #0	Y	2	1	х	AU_ALU									1	
[4]	bit(ST3, k4) = #0	Y	2	1†	х	AU_ALU									1	
t wi	hen this instruction is decoded to modify status bit CAFRZ (15), C	AEN (14), or CACLR	(13),	the CF	U pipelir	ne is flushed a	nd the i	nstruct	ion is e	xecute	d in 5 c	ycles re	egardless	s of the in	struction	context.
Со	mpare Accumulator, Auxiliary, or Tempo	rary Register	Со	ntent	t (pag	e 5-93)										
[1]	TC1 = uns(src-AU RELOP dst-AU)	Y	3	1	x	AU_ALU	.			.						.
	TC1 = uns(src RELOP dst)	Y	3	1	х	AU_ALU							1			See Note
	TC1 = uns(src-DU RELOP dst-DU)	Y	3	1	х	DU_ALU										
[2]	TC2 = uns(src-AU RELOP dst-AU)	Y	3	1	х	AU_ALU										
	TC2 = uns(src RELOP dst)	Y	3	1	х	AU_ALU							1			See Note
	TC2 = uns(src-DU RELOP dst-DU)	Y	3	1	х	DU_ALU										
Со	mpare Accumulator, Auxiliary, or Tempo	rary Register	Со	nten	t with	AND (pa	ge 5-	95)		1						1
[1]	TCx = TCy & uns(src-AU RELOP dst-AU)	Y	3	1	х	AU_ALU				.						
	TCx = TCy & uns(src RELOP dst)	Y	3	1	х	AU_ALU							1			See Note
	TCx = TCy & uns(src-DU RELOP dst-DU)	Y	3	1	х	DU_ALU										
[2]	TCx = !TCy & uns(src-AU RELOP dst-AU)	Y	3	1	х	AU_ALU										
	TCx = !TCy & uns(src RELOP dst)	Y	3	1	х	AU_ALU							1			See Note
	TCx = !TCy & uns(src-DU RELOP dst-DU)	Y	3	1	х	DU_ALU										
Со	mpare Accumulator, Auxiliary, or Tempo	rary Register	Со	nten	t with	OR (page	י 9 5-1	00)		I						I
[1]	TCx = TCy   uns(src-AU RELOP dst-AU)	ү (у. 1997) Тү	3	1	x	AU_ALU	.	-,		.						
	TCx = TCy   uns(src RELOP dst)	Y	3	1	х	AU_ALU							1			See Note
	TCx = TCy   uns(src-DU RELOP dst-DU)	Y	3	1	x	DU ALU										

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

							4	Addre	ss							
									n Unit				Buses			
No.		E	<b>S</b> 3	<b>C</b>	Pipe	Operator AU_ALU	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
[2]	TCx = !TCy   uns(src-AU RELOP dst-AU)				X		•	•	•	•	•	•	•			On a Nata O
	TCx = !TCy   uns(src RELOP dst)	Y Y	3	1	X	AU_ALU	•	•	•	•	•	•	1			See Note 2.
_	TCx = !TCy   uns(src-DU RELOP dst-DU)	I	3		X	DU_ALU	·			·	•		•	•	·	
Co	npare Accumulator, Auxiliary, or Temporary Regis	ter	Coi	nter	t Max		ge 5-	105	)	1						1
	dst-AU = max(src-AU, dst-AU)	Y	2	1	Х	AU_ALU	•			•			•		·	
	dst-AU = max(src-DU, dst-AU)	Y	2	1	Х	AU_ALU	•	·	·	•	•	·	1	·	·	
	dst-DU = max(src, dst-DU)	Y	2	1	Х	DU_ALU					•					See Note 1.
Со	mpare Accumulator, Auxiliary, or Temporary Regis	ter	Со	nter	t Mini	<b>mum</b> (pag	e 5-'	108)								
	dst-AU = min(src-AU, dst-AU)	Y	2	1	х	AU_ALU	.			.						
	dst-AU = min(src-DU, dst-AU)	Y	2	1	х	AU_ALU							1			
	dst-DU = min(src, dst-DU)	Y	2	1	х	DU_ALU										See Note 1.
Со	mpare and Branch (page 5-111)	•					•			•						
	compare (uns(src-AU RELOP K8)) goto L8	N	4	7/6†	х	AU_ALU + P_UNIT				.					1	
	compare (uns(src-DU RELOP K8)) goto L8	N	4	7/6†	х	DU_ALU + P_UNIT			·		•		·		1	
† x/y	cycles: x cycles = condition true, y cycles = condition false	•					•			•						•
Со	npare and Select Accumulator Content Maximum (	ра	ge 5	5-11-	4)											
[1]	max_diff(ACx, ACy, ACz, ACw)	Y	3	1	х	DU_ALU	.			.						
[2]	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Y	3	1	х	DU_ALU										
Со	npare and Select Accumulator Content Minimum (	pag	ge 5	-120	))											
[1]	min_diff(ACx, ACy, ACz, ACw)	Y	3	1	х	DU_ALU	.			.						
[2]	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	Y	3	1	х	DU_ALU										
Со	npare Memory with Immediate Value (page 5-126)	•														1
[1]	TC1 = (Smem == K16)	N	4	1	х	AU_ALU	1			1					1	
[2]	TC2 = (Smem == K16)	N	4	1	х	AU_ALU	1			1					1	

Table 4–1. Algebraic Instruction Set Summary (Continued)

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

	Instruction						Address Generation Unit						Buses			
No.		E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB KD	KDB	Notes
Со	mplement Accumulator, Auxiliary, or Tem	porary Regis	ste	r Bit	(page	5-128)										
	cbit(src-AU, Baddr)	Ν	3	1	х	AU_ALU	1									
	cbit(src-DU, Baddr)	Ν	3	1	х	DU_ALU	1									
Со	mplement Accumulator, Auxiliary, or Tem	porary Regis	ste	r Cor	ntent (	page 5-12	9)									
	dst-AU = ~src-AU	Y	2	1	х	AU_ALU	.			.						
	dst-AU = ~src-DU	Y	2	1	х	AU_ALU						•	1			
	dst-DU = ~src	Y	2	1	х	DU_ALU										See Note 1
Со	mplement Memory Bit (page 5-130)						•			•						
	cbit(Smem, src)	Ν	3	1	х	AU_ALU	1			1		1				
Со	mpute Exponent of Accumulator Content	(page 5-131)					1			1						•
	Tx = exp(ACx)		3	1	х	DU_ALU + DU_SHIFT + AU_ALU				.			1			
Со	mpute Mantissa and Exponent of Accum	ulator Conter	nt (	page	5-132	2)	1			I						I
	ACy = mant(ACx), Tx = exp(ACx)	Y	3	1	х	DU_ALU + DU_SHIFT + AU_ALU	.	·	·		·	·	1	·		
Со	unt Accumulator Bits (page 5-134)	I					1			1						I
[1]	Tx = count(ACx, ACy, TC1)	Y	3	1	х	DU_ALU + DU_SHIFT + AU_ALU		·		.	·	·	1	·		
[2]	Tx = count(ACx, ACy, TC2)	Y	3	1	х	DU_ALU + DU_SHIFT + AU_ALU	.	·			·	·	1	·		
Du	al 16-Bit Additions (page 5-135)	I					I			I						I
[1]	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	Ν	3	1	х	DU_ALU	1			2						
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	Ν	3	1	х	DU_ALU	1			2						

#### 10

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

			Address Generation Unit								Buses					
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Du	al 16-Bit Addition and Subtraction (page 5-140)															
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem - Tx	Ν	3	1	х	DU_ALU	1	·		1	•		•	·		
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) - Tx	N	3	1	х	DU_ALU	1			2						
Du	al 16-Bit Subtractions (page 5-145)						•									
[1]	HI(ACy) = HI(ACx) - HI(Lmem), LO(ACy) = LO(ACx) - LO(Lmem)	N	3	1	х	DU_ALU	1			2						
[2]	$ \begin{array}{l} HI(ACy) = HI(Lmem) - HI(ACx), \\ LO(ACy) = LO(Lmem) - LO(ACx) \end{array} $	N	3	1	х	DU_ALU	1			2			•	·		
[3]	$ \begin{array}{l} HI(ACx) = Tx - HI(Lmem), \\ LO(ACx) = Tx - LO(Lmem) \end{array} $	N	3	1	х	DU_ALU	1			2						
[4]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx	N	3	1	х	DU_ALU	1			2						
Du	al 16-Bit Subtraction and Addition (page 5-154)															
[1]	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	N	3	1	х	DU_ALU	1	·		1	•		•	·		
[2]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	N	3	1	х	DU_ALU	1			2						
Exe	ecute Conditionally (page 5-159)															
[1]	if (cond) execute(AD_Unit)	Ν	2	1	AD	P_UNIT	.			.						
[2]	if (cond) execute(D_Unit)	N	2	1	х	P_UNIT				.						
Exp	pand Accumulator Bit Field (page 5-166)															
	dst-AU = field_expand(ACx, k16)	N	4	1	х	DU_ALU + DU_SHIFT + AU_ALU						•	1	·	1	
	dst-DU = field_expand(ACx, k16)	N	4	1	х	DU_ALU + DU_SHIFT									1	

# Table 4–1. Algebraic Instruction Set Summary (Continued)

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			
No.	Instruction	E	s	с	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Ext	tract Accumulator Bit Field (page 5-167)															
	dst-AU = field_extract(ACx, k16)	Ν	4	1	х	DU_ALU + DU_SHIFT + AU_ALU							1		1	
	dst-DU = field_extract(ACx, k16)	N	4	1	х	DU_ALU + DU_SHIFT							•	·	1	
Fin	ite Impulse Response Filter, Antisymmetric	cal (page 5-	168	)			•			•						
	firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)	Ν	4	1	х	DU_ALU	2	1		2	1					
Fin	ite Impulse Response Filter, Symmetrical (	page 5-170)	)													
	firs(Xmem, Ymem, coef(Cmem), ACx, ACy)	Ν	4	1	х	DU_ALU	2	1		2	1					
Idle	<b>e</b> (page 5-172)	·					•			•						
	idle	Ν	4	?	D	P_UNIT				.						
Lea	<b>ast Mean Square (LMS)</b> (page 5-173)															
	Ims(Xmem, Ymem, ACx, ACy)	Ν	4	1	х	DU_ALU	2			2						
Lin	ear Addressing Qualifier (page 5-175)						-									
	linear()	Ν	1	1	AD											
Loa	ad Accumulator from Memory (page 5-176)									-						
[1]	ACx = rnd(Smem << Tx)	Ν	3	1	х	DU_ALU + DU_SHIFT	1		·	1			•	•		
[2]	ACx = low_byte(Smem) << #SHIFTW	Ν	3	1	х	DU_ALU + DU_SHIFT	1		·	1	•					
[3]	ACx = high_byte(Smem) << #SHIFTW	Ν	3	1	х	DU_ALU + DU_SHIFT	1		·	1	•		•	·		
[4]	ACx = Smem << #16	Ν	2	1	х	DU_ALU	1			1						
[5]	ACx = uns(Smem)	Ν	3	1	х		1			1						
[6]	ACx = uns(Smem) << #SHIFTW	Ν	4	1	х	DU_ALU + DU_SHIFT	1		·	1	•					
[7]	ACx = M40(dbl(Lmem))	Ν	3	1	х		1			2						

#### Table 4–1. Algebraic Instruction Set Summary (Continued)

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Instruction Set Summary

								ddres eration					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
[8]	LO(ACx) = Xmem, HI(ACx) = Ymem	Ν	3	1	х		2			2			·	·		
Loa	ad Accumulator Pair from Memory (pa	ge 5-187)														
[1]	pair(HI(ACx)) = Lmem	Ν	3	1	х		1			2						
[2]	pair(LO(ACx)) = Lmem	Ν	3	1	х		1			2						
Loa	ad Accumulator with Immediate Value	(page 5-190)														
[1]	ACx = K16 << #16	Ν	4	1	х	DU_ALU									1	
[2]	ACx = K16 << #SHFT	N	4	1	х	DU_ALU + DU_SHIFT							·	·	1	
Loa	ad Accumulator from Memory with Pa	allel Store Accu	umu	lato	Con	tent to Me	emor	<b>y</b> (pa	age 5	5-185	5)					
	ACy = Xmem << #16, Ymem = HI(ACx << T2)	N	4	1	х	DU_ALU + DU_SHIFT	2			2		2	·	·		
Loa	ad Accumulator, Auxiliary, or Tempora	ry Register from	n M	emoi	<b>'y</b> (pa	ge 5-193)	•			-						
[1]	dst = Smem	Ν	2	1	х		1			1					.	
[2]	dst = uns(high_byte(Smem))	Ν	3	1	х		1			1						
[3]	dst = uns(low_byte(Smem))	Ν	3	1	х		1			1						
Loa	ad Accumulator, Auxiliary, or Tempora	ry Register with	n Im	med	iate V	alue (page	e 5-1	99)								
[1]	dst = k4	Y	2	1	х		.								1	
[2]	dst = -k4	Y	2	1	х										1	
[3]	dst = K16	Ν	4	1	х										1	
Loa	ad Auxiliary or Temporary Register Pa	ir from Memory	(pa	ge 5-	203)		•			•					I	
	pair(TAx) = Lmem	N	3	1	х		1			2					.	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			I
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Loa	d CPU Register from Memory (page 5-204)															
[1]	BK03 = Smem	Ν	3	1	х		1			1						I
[2]	BK47 = Smem	Ν	3	1	х		1			1						l
[3]	BKC = Smem	Ν	3	1	х		1			1						l
[4]	BSA01 = Smem	Ν	3	1	х		1			1						l
[5]	BSA23 = Smem	Ν	3	1	х		1			1						1
[6]	BSA45 = Smem	N	3	1	х		1			1						1
[7]	BSA67 = Smem	Ν	3	1	х		1			1						1
[8]	BSAC = Smem	Ν	3	1	х		1			1						1
[9]	BRC0 = Smem	Ν	3	1	х		1			1						1
10]	BRC1 = Smem	Ν	3	1	х		1			1						1
[11]	CDP = Smem	Ν	3	1	х		1			1						l
12]	CSR = Smem	Ν	3	1	х		1			1						1
13]	DP = Smem	Ν	3	1	х		1			1						1
14]	DPH = Smem	Ν	3	1	х		1			1						I
15]	PDP = Smem	Ν	3	1	х		1			1						I
16]	SP = Smem	Ν	3	1	х		1			1						I
17]	SSP = Smem	Ν	3	1	х		1			1						I
18]	TRN0 = Smem	Ν	3	1	х		1			1						I
19]	TRN1 = Smem	Ν	3	1	х		1			1			•	•		1
20]	RETA = dbl(Lmem)	Ν	3	5	х		1			2						

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								ddres eration					Buses			
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Loa	d CPU Register with Immediate Value (page 5-207)															
[1]	BK03 = k12	Υ	3	1	AD									1		
[2]	BK47 = k12	Y	3	1	AD									1		
[3]	BKC = k12	Y	3	1	AD									1		
[4]	BRC0 = k12	Y	3	1	AD									1		
[5]	BRC1 = k12	Y	3	1	AD									1		
[6]	CSR = k12	Y	3	1	AD									1		
[7]	DPH = k7	Y	3	1	AD									1		
[8]	PDP = k9	Y	3	1	AD									1		
[9]	BSA01 = k16	Ν	4	1	AD									1		
[10]	BSA23 = k16	Ν	4	1	AD		•			•	•		•	1	•	
[11]	BSA45 = k16	Ν	4	1	AD		•			•	•		•	1	•	
[12]	BSA67 = k16	Ν	4	1	AD		•			•	•		•	1	•	
[13]	BSAC = k16	Ν	4	1	AD		•			•	•		•	1	•	
[14]	CDP = k16	Ν	4	1	AD		•			•	•		•	1	•	
[15]	DP = k16	Ν	4	1	AD									1		
[16]	SP = k16	Ν	4	1	AD									1		
[17]	SSP = k16	Ν	4	1	AD									1		
Loa	d Extended Auxiliary Register from Memory (page	5-2	209)													
	XAdst = dbl(Lmem)	Ν	3	1	х		1			2						
Loa	d Extended Auxiliary Register with Immediate Valu	ie (	pac	e 5-2	210)											I
	XAdst = k23	Ν	6	1	AD		1			1						
Loa	d Memory with Immediate Value (page 5-211)	I					I			I						l
_0a	Smem = K8	N	3	1	х		1			Ι.		1			1	
[2]	Smem = K16		4	1	x		1	•	•	·		1			1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

# Instruction Set Summary

								Addres eratio					Buses			
No.	Instruction	E	s	с	Pipe	Operator	DA	СА	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Me	mory Delay (page 5-212)															
	delay(Smem)	Ν	2	1	х		2	1		1	1	1				
Me	mory-Mapped Register Access Qualifier (page 5-	213)					-			-						
	mmap()	Ν	1	1	D					.						
Мо	dify Auxiliary Register Content (page 5-214)									•						
	mar(Smem)	Ν	2	1	AD		1			1						
Мо	dify Auxiliary Register Content with Parallel Mul	tiply	(pag	e 5-2	216)					•					ľ	
	mar(Xmem), ACx = M40(md(uns(Ymem) * uns(coef(Cmem))))	N	4	1	х	DU_ALU	2	1		2	1					
Мо	dify Auxiliary Register Content with Parallel Mul	tiply	and	Acc	umula	ate (page	5-218	B)							-	
[1]	mar(Xmem), ACx = <mark>M40(rnd(</mark> ACx + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1		2	1					
[2]	mar(Xmem), ACx = <mark>M40(md((</mark> ACx >> #16) + ( <u>uns(</u> Ymem) * <u>uns(</u> coef(Cmem)))))	N	4	1	х	DU_ALU	2	1		2	1					
Мо	dify Auxiliary Register Content with Parallel Mul	tiply	and	Sub	tract	(page 5-2	23)									
	mar(Xmem), ACx = <mark>M40(rnd(</mark> ACx - (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1		2	1					
Мо	dify Auxiliary or Temporary Register Content (pa	ige 5-	225	)												
[1]	mar(TAy = TAx)	N	3	1	AD		1									
[2]	mar(TAx = P8)	N	3	1	AD		1							1		
[3]	mar(TAx = D16)	Ν	4	1	AD		1			.				1		
Мо	dify Auxiliary or Temporary Register Content by	Addi	tion	(pa	ge 5-2	29)										
[1]	mar(TAy + TAx)	Ν	3	1	AD		1			.						
[2]	mar(TAx + P8)	N	3	1	AD		1							1		

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Мо	dify Auxiliary or Temporary Register Content by Su	ıbtı	ract	ion	(page	5-233)										
[1]	mar(TAy – TAx)	Ν	3	1	AD		1									
[2]	mar(TAx – P8)	Ν	3	1	AD		1							1		
Мо	dify Data Stack Pointer (SP) (page 5-237)	•					•			•						
	SP = SP + K8	Y	2	1	AD									1		
Мо	dify Extended Auxiliary Register Content (page 5-2	38)														1
	XAdst = mar(Smem)	1 <sup>-</sup>	3	1	AD		1			1						
Мо	ve Accumulator Content to Auxiliary or Temporary	Re	ais	ter (	page (	5-239)	I			I						I
	TAx = HI(ACx)	i i	-	1		AU_ALU							1			
Мо	ve Accumulator, Auxiliary, or Temporary Register (	l Cor	nten	t (p	ade 5-	240)	I			I						I
	dst-AU = src-AU	1	2	1	X	AU_ALU	.			.						ĺ
	dst-AU = src-DU	Y	2	1	х	AU_ALU							1			
	dst-DU = src	Y	2	1	х	DU_ALU										See Note
Мо	ve Auxiliary or Temporary Register Content to Acc	um	ula	tor (	page (	5-242)	1			1						1
	HI(ACx) = TAx	1		1	. с х	, DU_ALU	.			.						l
Mo	ve Auxiliary or Temporary Register Content to CPL	I R	eais	ster	(page	5-243)	I			I						I
[1]	BRC0 = TAx	i i	2	1	X X	AU_ALU	Ι.			Ι.						1
[2]	BRC1 = TAx			1	X	AU_ALU										
[3]	CDP = TAx	Y	2	1	х	AU_ALU				Ι.						
[4]	CSR = TAx	Y	2	1	х	AU_ALU										
[5]	SP = TAx	Y	2	1	х	AU_ALU										
[6]	SSP = TAx	Y	2	1	х	AU_ALU										

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratio					Buses			
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Мо	ve CPU Register Content to Auxiliary or Temp	orary Re	egis	ster	(page	5-245)										
[1]	TAx = BRC0	Y	2	1	х	AU_ALU										
[2]	TAx = BRC1	Y	2	1	х	AU_ALU										
[3]	TAx = CDP	Y	2	1	х	AU_ALU										
[4]	TAx = RPTC	Y	2	1	х	AU_ALU										
[5]	TAx = SP	Y	2	1	х	AU_ALU										
[6]	TAx = SSP	Y	2	1	х	AU_ALU										
Мо	ve Extended Auxiliary Register Content (page	5-247)					-									
	xdst-AU = xsrc-AU	N	2	1	х	AU_ALU				.						
	xdst-AU = xsrc-DU	N	2	1	х	AU_ALU							1			
	xdst-DU = xsrc	N	2	1	х	DU_ALU										See Note
Мо	ve Memory to Memory (page 5-248)	I					•									1
[1]	Smem = coef(Cmem)	Ν	3	1	х		2			1		1				
[2]	coef(Cmem) = Smem	N	3	1	х		2			1		1				
[3]	Lmem = dbl(coef(Cmem))	N	3	1	х		2			2		2				
[4]	dbl(coef(Cmem)) = Lmem	N	3	1	х		2			2		2				
[5]	dbl(Ymem) = dbl(Xmem)	N	3	1	х		2			2		2				
[6]	Ymem = Xmem	N	3	1	х		2			2		2				
Mu	Itiply (MPY) (page 5-255)															
[1]	ACy = rnd(ACy * ACx)	Y	2	1	х	DU_ALU	.			.						
[2]	ACy = rnd(ACx * Tx)	Y	2	1	х	DU_ALU										
[3]	ACy = rnd(ACx * K8)	Y	3	1	х	DU_ALU									1	
[4]	ACy = rnd(ACx + K16)	N	4	1	х	DU_ALU									1	
[5]	ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	N	3	1	х	DU_ALU	1	1		1	1					
[6]	ACy = rnd(Smem * ACx)[, T3 = Smem]	N	3	1	х	DU_ALU	1			1						

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

4-19

Instruction Set Summary

								Addres eration				l	Buses			
lo.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Note
7]	ACx = rnd(Smem * K8)[, T3 = Smem]	Ν	4	1	х	DU_ALU	1		•	1			·	•	1	
8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	Ν	4	1	х	DU_ALU	2			2						
9]	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
۸u	tiply with Parallel Multiply and Accumulate (page s	5-26	67)							-						
	ACx = M40(md(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(md((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1	•	2	1					
Nu	tiply with Parallel Store Accumulator Content to N	lem	ory	(pa	ge 5-2	69)	•			•					-	
	ACy = md(Tx * Xmem), Ymem = HI(ACx << T2) [, T3 = Xmem]	N	4	1	х	DU_ALU + DU_SHIFT	2			2	•	2				
۸u	tiply and Accumulate (MAC) (page 5-271)									-						
[1]	ACy = rnd(ACy + (ACx * Tx))	Y	2	1	х	DU_ALU				.						
[2]	ACy = rnd((ACy * Tx) + ACx)	Y	2	1	х	DU_ALU										
[3]	ACy = rnd(ACx + (Tx * K8))	Y	3	1	х	DU_ALU									1	
4]	ACy = rnd(ACx + (Tx * K16))	Ν	4	1	х	DU_ALU									1	
[5]	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1	1		1	1					
6]	ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
7]	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
[8]	ACy = rnd(ACx + (Smem * K8))[, T3 = Smem ]	Ν	4	1	х	DU_ALU	1			1					1	
[9]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	Ν	4	1	х	DU_ALU	2			2						
10]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	Ν	4	1	х	DU_ALU	2			2						
٧u	tiply and Accumulate with Parallel Delay (page 5-2	86)														
	ACx = md(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)	Ν	3	1	Х	DU_ALU	2	1		1	1	1				
Иu	tiply and Accumulate with Parallel Load Accumula	ator	fro	om N	lemor	<b>y</b> (page 5-	·288)									
	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	N	4	1	х	DU_ALU	2			2						

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eration					Buses			
No.	Instruction	Е	s	с	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Mu	Itiply and Accumulate with Parallel Multiply (page	e 5-29	90)													
	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	х	DU_ALU	2	1		2	1					
Mu	tiply and Accumulate with Parallel Store Accum	ulato	r Co	onte	nt to I	Memory (p	bage	5-29	2)							
	ACy = md(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	N	4	1	х	DU_ALU + DU_SHIFT	2			2		2				
Mu	Itiply and Subtract (MAS) (page 5-294)															
[1]	ACy = rnd(ACy - (ACx * Tx))	Y	2	1	х	DU_ALU										
[2]	ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1	1		1	1					
[3]	ACy = rnd(ACy - (Smem * ACx))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
[4]	ACy = rnd(ACx - (Tx * Smem))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
[5]	ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	Ν	4	1	х	DU_ALU	2			2						
Mu	Itiply and Subtract with Parallel Load Accumulat	or fro	om I	Mem	ory (p	age 5-302	2)			•						
	ACx = md(ACx - (Tx * Xmem)), ACy = Ymem << #16 [, T3 = Xmem]	N	4	1	х	DU_ALU	2	•		2		•			•	
Mu	tiply and Subtract with Parallel Multiply (page 5-	304)														
	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	N	4	1	х	DU_ALU	2	1		2	1		•	•	•	
Mu	Itiply and Subtract with Parallel Multiply and Acc	umu	late	(pag	ge 5-3	06)										
[1]	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1		2	1					
[2]	ACx = M40(rnd(ACx – (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1		2	1					
Mu	Itiply and Subtract with Parallel Store Accumulat	or Co	onte	ent to	o Men	n <b>ory</b> (page	e 5-3	11)		•						
	ACy = md(ACy - (Tx * Xmem)), Ymem = HI(ACx << T2) [, T3 = Xmem]	Ν	4	1	х	DU_ALU + DU_SHIFT	2			2	·	2			·	

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Neg	gate Accumulator, Auxiliary, or Temporary Regi	ster Co	onte	ent (	page !	5-313)										
	dst-AU = -src-AU	Y	2	1	х	AU_ALU										
	dst-AU = -src-DU	Y	2	1	х	AU_ALU			·				1	·	·	
	dst-DU = -src	Y	2	1	Х	DU_ALU	.	·		.		·				See Note
No	Operation (NOP) (page 5-315)															
[1]	nop	Y	1	1	D											
[2]	nop_16	Y	2	1	D								·	·		
Par	allel Modify Auxiliary Register Contents (page s	5-316)														
	mar(Xmem) , mar(Ymem) , mar(coef(Cmem))	Ν	4	1	х		2	1		2	1					
Par	allel Multiplies (page 5-317)															
	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	Ν	4	1	х	DU_ALU	2	1	·	2	1					
Par	allel Multiply and Accumulates (page 5-319)									-						
[1]	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1	·	2	1				·	
[2]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	х	DU_ALU	2	1		2	1					
[3]	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	N	4	1	Х	DU_ALU	2	1	·	2	1	·				
Par	allel Multiply and Subtracts (page 5-326)															
	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy - (uns(Ymem) * uns(coef(Cmem)))))	Ν	4	1	х	DU_ALU	2	1	•	2	1				·	
Per	ipheral Port Register Access Qualifiers (page 5	-328)														
[1]	readport()	Ν	1	1	D					.						
[2]	writeport()	Ν	1	1	D											
Pop	o Accumulator or Extended Auxiliary Register (	Conten	t fr	om S	Stack	Pointers	(page	e 5-3	30)							
	xdst = popboth()	Y	2	1	х		1		1	2						1

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eration					Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Pop	<b>5 Top of Stack</b> (page 5-331)															
[1]	dst1, dst2 = pop()	Y	2	1	х		1		1	2						
[2]	dst = pop()	Y	2	1	х		1		1	1						
[3]	dst, Smem = pop()	Ν	3	1	Х		1		1	2		1				
[4]	ACx = dbl(pop())	Y	2	1	Х		1		1	2						
[5]	Smem = pop()	Ν	2	1	Х		1		1	1		1				
[6]	dbl(Lmem) = pop()	Ν	2	1	Х		1		1	2		2				
Pus	sh Accumulator or Extended Auxiliary R	egister Conte	ent t	o St	ack P	ointers (p	age :	5-338	3)							
	pushboth(xsrc)	Y	2	1	х		1		1			2				
Pus	sh to Top of Stack (page 5-339)	Ĩ														
[1]	push(src1, src2)	Y	2	1	х		1		1	.		2				
[2]	push(src)	Y	2	1	х		1		1			1				
[3]	push(src, Smem)	Ν	3	1	х		1		1	1		2				
[4]	dbl(push(ACx))	Y	2	1	х		1		1			2				
[5]	push(Smem)	Ν	2	1	х		1		1	1		1				
[6]	push(dbl(Lmem))	Ν	2	1	х		1		1	2		2				
Rep	peat Block of Instructions Unconditional	ly (page 5-340	6)				-			-						
[1]	localrepeat{ }	Y	2	1	AD	P_UNIT				.						
[2]	blockrepeat{ }	Y	3	1	AD	P_UNIT										
Rep	peat Single Instruction Conditionally (page	ge 5-357)					•			•						
	while (cond && (RPTC < k8)) repeat	Y	3	1	AD	P_UNIT	.			.				1		
Rei	peat Single Instruction Unconditionally (	page 5-360)					1									
[1]	repeat(k8)	Y	2	1	AD	P_UNIT	.			.				1		
[2]	repeat(k16)	Y	3	1	AD	P_UNIT								1		
[3]	repeat(CSR)	Y	2	1	AD	P_UNIT	Ι.									

### + C $(\mathbf{c})$ ~*1*:. ۷۲ -. . . . . . . . ... 0

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Instruction Set Summary

							Addres eration					Buses			
No. Instruction	E	S	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Repeat Single Instruction Unconditionally a	nd Decreme	nt C	CSR	(page	5-365)										
repeat(CSR), CSR -= k4	Y	2	1	х	AU_ALU + P_UNIT	.								1	
Repeat Single Instruction Unconditionally a	nd Incremen	it C	SR (	page 5	5-367)										
[1] repeat(CSR), CSR += TAx	Y	2	1	х	AU_ALU + P_UNIT						·	•	•	·	
[2] repeat(CSR), CSR += k4	Y	2	1	х	AU_ALU + P_UNIT				•	·	·	•	•	1	
Return Conditionally (page 5-370)															
if (cond) return	Y	3	5/5†	R	P_UNIT	1		1	2						
† x/y cycles: x cycles = condition true, y cycles = condition false															
Return Unconditionally (page 5-372)															
return	Y	2	5	D	P_UNIT	1		1	2				•		
Return from Interrupt (page 5-374)															
return_int	Y	2	5	D	P_UNIT	1		1	2						
Rotate Left Accumulator, Auxiliary, or Temp	orary Regist	er (	Cont	ent (pa	age 5-376)				-						
dst-AU = BitOut \\ src-AU \\ BitIn	Y	3	1	х	AU_ALU				.						
dst-AU = BitOut \\ src-DU \\ BitIn	Y	3	1	х	AU_ALU							1			
dst-DU = BitOut \\ src \\ BitIn	Y	3	1	х	DU_ALU + DU_SHIFT			•		·					See Note 1
Rotate Right Accumulator, Auxiliary, or Tem	porary Regi	ster	r Coi	ntent (	page 5-37	8)									
dst-AU = BitIn // src-AU // BitOut	Y	3	1	х	AU_ALU								•		
dst-AU = BitIn // src-DU // BitOut	Y	3	1	х	AU_ALU							1			
dst-DU = BitIn // src // BitOut	Y	3	1	х	DU_ALU + DU_SHIFT			•	•	·	·				See Note 1
Round Accumulator Content (page 5-380)															
ACy = rnd(ACx)	Y	2	1	х	DU_ALU	.			.						

4-24

								ddres				I	Buses			
No. Ins	struction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Satura	ate Accumulator Content (page 5-382)															
AC	Cy = saturate <mark>(rnd</mark> (ACx))	Y	2	1	х	DU_ALU										
Set Ac	ccumulator, Auxiliary, or Temporary Registe	er Bit (pa	age	5-38	64)		-			•						
bit	s(src-AU, Baddr) = #1	Ν	3	1	х	AU_ALU	1									
bit	:(src-DU, Baddr) = #1	N	3	1	х	DU_ALU	1									
Set Me	emory Bit (page 5-385)	·					•			•						•
bit	(Smem, src) = #1	Ν	3	1	х	AU_ALU	1			1		1				
Set St	atus Register Bit (page 5-386)	Ĭ								1						Ĩ
[1] bit	s(ST0, k4) = #1	Y	2	1	х	AU_ALU				.					1	
[2] bit	:(ST1, k4) = #1	Y	2	1	х	AU_ALU									1	
[3] bit	t(ST2, k4) = #1	Y	2	1	х	AU_ALU									1	
[4] bit	t(ST3, k4) = #1	Y	2	1†	х	AU_ALU									1	
† When tl	his instruction is decoded to modify status bit CAFRZ (15), CAEN (14	), or CACLR	(13),	the CF	PU pipelii	ne is flushed an	d the in	structi	on is e	xecuted	d in 5 c	ycles re	egardless	of the in	struction	context.
Shift A	Accumulator Content Conditionally (page 5	389)														
[1] AC	Cx = sftc(ACx, TC1)	Y	2	1	х	DU_ALU + DU_SHIFT										
[2] AC	Cx = sftc(ACx, TC2)	Y	2	1	х	DU_ALU + DU_SHIFT										
Shift A	Accumulator Content Logically (page 5-391)	)														
[1] AC	Cy = ACx <<< Tx	Y	2	1	х	DU_ALU + DU_SHIFT						•	·	•	•	
[2] AC	Cy = ACx <<< #SHIFTW	Y	3	1	х	DU_ALU + DU_SHIFT										

dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								ddres eratior					Buses			
No.	Instruction	E	S	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Note
Shi	ft Accumulator, Auxiliary, or Temporary	/ Register Cont	tent	t Log	gically	/ (page 5-3	394)									
[1]	dst-AU = dst-AU <<< #1	Y	2	1	х	AU_ALU										
	dst-DU = dst-DU <<< #1	Y	2	1	х	DU_ALU + DU_SHIFT	•			•			•		·	
[2]	dst-AU = dst-AU >>> #1	Y	2	1	Х	AU_ALU										
	dst-DU = dst-DU >>> #1	Y	2	1	х	DU_ALU + DU_SHIFT	•	•	•		·					
Sig	ned Shift of Accumulator Content (pag	e 5-397)														
[1]	ACy = ACx << Tx	Y	2	1	х	DU_ALU + DU_SHIFT	.	•	•							
[2]	ACy = ACx << #SHIFTW	Y	3	1	х	DU_ALU + DU_SHIFT										
[3]	ACy = ACx < <c td="" tx<=""><td>Y</td><td>2</td><td>1</td><td>х</td><td>DU_ALU + DU_SHIFT</td><td>•</td><td></td><td></td><td>•</td><td></td><td></td><td>•</td><td></td><td>·</td><td></td></c>	Y	2	1	х	DU_ALU + DU_SHIFT	•			•			•		·	
[4]	ACy = ACx < <c #shiftw<="" td=""><td>Y</td><td>3</td><td>1</td><td>х</td><td>DU_ALU + DU_SHIFT</td><td></td><td>•</td><td>•</td><td></td><td>·</td><td>•</td><td></td><td></td><td></td><td></td></c>	Y	3	1	х	DU_ALU + DU_SHIFT		•	•		·	•				
Sig	ned Shift of Accumulator, Auxiliary, or	Temporary Reg	gist	er C	onten	<b>t</b> (page 5-	406)									
[1]	dst-AU = dst-AU >> #1	Y	2	1	х	AU_ALU										
	dst-DU = dst-DU >> #1	Y	2	1	х	DU_ALU + DU_SHIFT										
[2]	dst-AU = dst-AU << #1	Y	2	1	х	AU_ALU										
	dst-DU = dst-DU << #1	Y	2	1	х	DU_ALU + DU_SHIFT		·	·		·	·				
Sof	tware Interrupt (page 5-411)															
	intr(k5)	Ν	2	3	D	P_UNIT	1		1	.		2				
Sof	itware Reset (page 5-413)															
	reset	Ν	2	?	D	P_UNIT	.	·	·	.	·	·	•	·	•	
Sof	f <b>tware Trap</b> (page 5-417)									_						
	trap(k5)	N	2	?	D	P_UNIT	1		1			2				

otes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eratior					Buses			
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Sqı	<b>Jare</b> (page 5-419)															
[1]	ACy = rnd(ACx * ACx)	Y	2	1	х	DU_ALU				.						
[2]	ACx = md(Smem * Smem)[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
Sqı	uare and Accumulate (page 5-422)	•														
[1]	ACy = rnd(ACy + (ACx * ACx))	Y	2	1	х	DU_ALU										
[2]	ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
Sqı	uare and Subtract (page 5-425)	•														
[1]	ACy = rnd(ACy - (ACx * ACx))	Y	2	1	х	DU_ALU				.						
[2]	ACy = rnd(ACx - (Smem * Smem))[, T3 = Smem]	Ν	3	1	х	DU_ALU	1			1						
Sqı	uare Distance (page 5-428)	I					1			1						
	sqdst(Xmem, Ymem, ACx, ACy)	Ν	4	1	х	DU_ALU	2			2						
Sto	re Accumulator Content to Memory (page 5-	430)					I			1					I	
[1]	Smem = HI(ACx)	N	2	1	х		1			.		1				
[2]	Smem = HI(rnd(ACx))	N	3	1	х	DU_SHIFT	1					1				
[3]	Smem = LO(ACx << Tx)	Ν	3	1	х	DU_SHIFT	1					1				
[4]	Smem = HI(rnd(ACx << Tx))	Ν	3	1	х	DU_SHIFT	1					1				
[5]	Smem = LO(ACx << #SHIFTW)	Ν	3	1	х	DU_SHIFT	1					1				
[6]	Smem = HI(ACx << #SHIFTW)	Ν	3	1	х	DU_SHIFT	1					1				
[7]	Smem = HI(rnd(ACx << #SHIFTW))	Ν	4	1	х	DU_SHIFT	1					1				
[8]	Smem = HI(saturate(uns(rnd(ACx))))	Ν	3	1	х	DU_SHIFT	1					1				
[9]	Smem = HI(saturate(uns(rnd(ACx << Tx))))	Ν	3	1	х	DU_SHIFT	1					1	•	·		
[10]	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	Ν	4	1	х	DU_SHIFT	1					1	•	·		
[11]	dbl(Lmem) = ACx	Ν	3	1	х		1		·	•		2		·		
[12]	dbl(Lmem) = saturate(uns(ACx))	N	3	1	х	DU_SHIFT	1					2				

SPRU375G

4-27

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

Instruction Set Summary

								Addres eratior					Buses			
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
[13]	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	N	3	1	Х	DU_SHIFT	1		•	•	·	2				
[14]	Xmem = LO(ACx), Ymem = HI(ACx)	N	3	1	х		2					2	·	·	·	
Sto	re Accumulator Pair Content to Memory (page 5-45	50)					•			•						•
[1]	Lmem = pair(HI(ACx))	Ν	3	1	х		1			.		2				
[2]	Lmem = pair(LO(ACx))	Ν	3	1	х		1					2				
Sto	re Accumulator, Auxiliary, or Temporary Register (	Cor	nten	t to	Memo	ory (page s	5-45	3)		•						•
[1]	Smem = src	Ν	2	1	х		1			.		1				
[2]	high_byte(Smem) = src	Ν	3	1	х		1					1				
[3]	low_byte(Smem) = src	Ν	3	1	х		1					1				
Sto	re Auxiliary or Temporary Register Pair Content to	Me	emo	ry (	page 5	5-457)	•			•						•
	Lmem = pair(TAx)	Ν	3	1	х		1			.		2				
Sto	re CPU Register Content to Memory (page 5-458)						•			•						•
[1]	Smem = BK03	Ν	3	1	х		1			.		1				
[2]	Smem = BK47	Ν	3	1	х		1					1				
[3]	Smem = BKC	Ν	3	1	х		1					1				
[4]	Smem = BSA01	Ν	3	1	х		1					1				
[5]	Smem = BSA23	Ν	3	1	Х		1			•		1				
[6]	Smem = BSA45	Ν	3	1	Х		1			•		1				
[7]	Smem = BSA67	Ν	3	1	Х		1					1				
[8]	Smem = BSAC	Ν	3	1	х		1					1				
[9]	Smem = BRC0	Ν	3	1	х		1			.		1				
[10]	Smem = BRC1	Ν	3	1	Х		1					1				

N 3

N 3 1

1

Х

Х

1

1

.

.

1

1

### Table 4–1. Algebraic Instruction Set Summary (Continued)

Instruction Set Summary

4-28

[11] Smem = CDP

[12] Smem = CSR

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eration				I	Buses			
No.	Instruction	E	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
[13]	Smem = DP	N	3	1	х		1	·	·	·	·	1	·	·	·	
[14]	Smem = DPH	N	3	1	х		1					1				
[15]	Smem = PDP	Ν	3	1	х		1					1				
[16]	Smem = SP	Ν	3	1	х		1	•	•		•	1	·	·		
[17]	Smem = SSP	Ν	3	1	х		1					1				
[18]	Smem = TRN0	Ν	3	1	х		1				•	1				
[19]	Smem = TRN1	Ν	3	1	х		1					1				
[20]	dbl(Lmem) = RETA	Ν	3	5	х		1					2				
Sto	re Extended Auxiliary Register Content to I	<b>lemory</b> (pa	age	5-46	62)											
	dbl(Lmem) = XAsrc	N	3	1	х		1			.		2				ĺ
Sut	otract Conditionally (page 5-463)	I					I			I						I
	subc(Smem, ACx, ACy)	N	3	1	х	DU_ALU	1			1						I
Ck	ptraction (page 5-465)	I				_	I			l						]
	dst-AU = dst-AU - src-AU	Y	2	1	v	AU_ALU	I			I						1
[1]		Y			X		•	•	•	•	•	•		·	•	
	dst-AU = dst-AU - src-DU			1	X	AU_ALU	•	•	•	•	•	•	I	·	•	0
101	dst-DU = dst-DU – src	Y		1	X	DU_ALU	•	•	•	•	•	·			•	See Note
[2]	dst-AU = dst-AU - k4	Y		1	х	AU_ALU	•	•	·	·			·	·	1	
	dst-DU = dst-DU - k4		2	1	Х	DU_ALU	•	•	•	•	•	•	·	·	1	
[3]	dst-AU = src-AU – K16	N	4	1	х	AU_ALU	•	•	•	•	•	·	•		1	
	dst-AU = src-DU – K16	N		1	х	AU_ALU	•	•	•	·			1		1	
	dst-DU = src – K16	N	4	1	х	DU_ALU	•	·	•	·			·	·	1	See Note
[4]	dst-AU = src-AU - Smem	N	3	1	х	AU_ALU	1	·	·	1	·	·	·	·	•	
	dst-AU = src-DU – Smem	N	3	1	х	AU_ALU	1	•		1			1	·		
	dst-DU = src - Smem	N	3	1	Х	DU_ALU	1			1						See Note 1

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

								Addres eration				I	Buses			
No.	Instruction	Е	s	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
[5]	dst-AU = Smem – src-AU	Ν	3	1	Х	AU_ALU	1	•	•	1						
	dst-AU = Smem – src-DU	Ν	3	1	х	AU_ALU	1			1	•		1			
	dst-DU = Smem – src	Ν	3	1	х	DU_ALU	1			1						See Note 1.
[6]	ACy = ACy - (ACx << Tx)	Y	2	1	х	DU_ALU + DU_SHIFT		•	•		•	·				
[7]	ACy = ACy – (ACx << #SHIFTW)	Y	3	1	х	DU_ALU + DU_SHIFT	•	•	•	•	•	·				
[8]	ACy = ACx - (K16 << #16)	Ν	4	1	х	DU_ALU				•	•				1	
[9]	ACy = ACx – (K16 << #SHFT)	Ν	4	1	х	DU_ALU + DU_SHIFT									1	
[10]	ACy = ACx - (Smem << Tx)	N	3	1	х	DU_ALU + DU_SHIFT	1			1						
[11]	ACy = ACx - (Smem << #16)	Ν	3	1	х	DU_ALU	1			1						
[12]	ACy = (Smem << #16) – ACx	Ν	3	1	х	DU_ALU	1			1						
[13]	ACy = ACx - uns(Smem) - BORROW	Ν	3	1	х	DU_ALU	1			1						
[14]	ACy = ACx - uns(Smem)	Ν	3	1	х	DU_ALU	1			1						
[15]	ACy = ACx - (uns(Smem) << #SHIFTW)	N	4	1	х	DU_ALU + DU_SHIFT	1			1						
[16]	ACy = ACx - dbl(Lmem)	Ν	3	1	х	DU_ALU	1			2	•					
[17]	ACy = dbl(Lmem) - ACx	Ν	3	1	х	DU_ALU	1			2	•					
[18]	ACx = (Xmem << #16) - (Ymem << #16)	Ν	3	1	х	DU_ALU	2			2						
Sub	otraction with Parallel Store Accumulator Content t	o N	/lem	ory	(page	5-490)										
	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	N	4	1	х	DU_ALU + DU_SHIFT	2	•	·	2	•	2	•		•	
Sw	ap Accumulator Content (page 5-492)															
[1]	swap(AC0, AC2)	Y	2	1	х	DU_SWAP	.			.						

Y 2 1

X DU\_SWAP

### Table 4–1. Algebraic Instruction Set Summary (Continued)

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

[2] swap(AC1, AC3)

								Addres eration					Buses			
No.	Instruction	Е	S	С	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Sw	ap Accumulator Pair Content (page 5-493)															
	swap(pair(AC0), pair(AC2))	Y	2	1	х	DU_SWAP										
Sw	ap Auxiliary Register Content (page 5-494)						-			-						
[1]	swap(AR0, AR1)	Y	2	1	AD	AU_SWAP										
[2]	swap(AR0, AR2)	Y	2	1	AD	AU_SWAP										
[3]	swap(AR1, AR3)	Y	2	1	AD	AU_SWAP										
Sw	ap Auxiliary Register Pair Content (page 5-495)															
	swap(pair(AR0), pair(AR2))	Y	2	1	AD	AU_SWAP	.			.						
Sw	ap Auxiliary and Temporary Register Content (pag	e 5-	496	)			•			•						
[1]	swap(AR4, T0)	Y	2	1	AD	AU_SWAP	.			.						
[2]	swap(AR5, T1)	Y	2	1	AD	AU_SWAP										
[3]	swap(AR6, T2)	Y	2	1	AD	AU_SWAP										
[4]	swap(AR7, T3)	Y	2	1	AD	AU_SWAP										
Sw	ap Auxiliary and Temporary Register Pair Content	(pa	ge 5	5-49	3)											
[1]	swap(pair(AR4), pair(T0))	Y	2	1	AD	AU_SWAP										
[2]	swap(pair(AR6), pair(T2))	Y	2	1	AD	AU_SWAP					•					
Sw	ap Auxiliary and Temporary Register Pairs Conten	<b>t</b> (p	age	5-5	00)		-			-						
	swap(block(AR4), block(T0))	Y	2	1	AD	AU_SWAP										
Sw	ap Temporary Register Content (page 5-502)	'														I
[1]	swap(T0, T2)	Y	2	1	AD	AU_SWAP										
[2]	swap(T1, T3)	Y	2	1	AD	AU_SWAP										
Sw	ap Temporary Register Pair Content (page 5-503)	1														I
	swap(pair(T0), pair(T2))	Y	2	1	AD	AU_SWAP	.			.						

2) dst-DU, src-AU or dst-AU, src-DU

Instruction Set Summary

								Addres eratior					Buses			
No.	Instruction	Е	s	с	Pipe	Operator	DA	CA	SA	DR	CR	DW	ACB	KAB	KDB	Notes
Tes	t Accumulator, Auxiliary, or Temporary Register B	it (j	bage	e 5-5	604)											
[1]	TC1 = bit(src-AU, Baddr)	N	3	1	х	AU_ALU	1			.						
	TC1 = bit(src-DU, Baddr)	N	3	1	х	DU_ALU	1									
[2]	TC2 = bit(src-AU, Baddr)	Ν	3	1	х	AU_ALU	1									
	TC2 = bit(src-DU, Baddr)	Ν	3	1	х	DU_ALU	1									
Tes	t Accumulator, Auxiliary, or Temporary Register B	it P	air	(pag	e 5-50	6)										
	bit(src-AU, pair(Baddr))	N	3	1	х	AU_ALU	1			.						
	bit(src-DU, pair(Baddr))	N	3	1	х	DU_ALU	1									
Tes	t Memory Bit (page 5-508)	•								•					I	
[1]	TCx = bit(Smem, src)	N	3	1	х	AU_ALU	1			1					.	
[2]	TCx = bit(Smem, k4)	N	3	1	х	AU_ALU	1			1					1	
Tes	t and Clear Memory Bit (page 5-511)	•								•					·	
[1]	TC1 = bit(Smem, k4), bit(Smem, k4) = #0	N	3	1	х	AU_ALU	1			1		1	·	·	1	
[2]	TC2 = bit(Smem, k4), bit(Smem, k4) = #0	N	3	1	х	AU_ALU	1			1		1			1	
Tes	t and Complement Memory Bit (page 5-512)	-					-			-					-	
[1]	TC1 = bit(Smem, k4), cbit(Smem, k4)	N	3	1	х	AU_ALU	1			1		1			1	
[2]	TC2 = bit(Smem, k4), cbit(Smem, k4)	N	3	1	х	AU_ALU	1			1	·	1	·	•	1	
Tes	t and Set Memory Bit (page 5-513)															
[1]	TC1 = bit(Smem, k4), bit(Smem, k4) = #1	N	3	1	х	AU_ALU	1			1		1			1	
[2]	TC2 = bit(Smem, k4), bit(Smem, k4) = #1	N	3	1	х	AU_ALU	1			1		1			1	

Notes: 1) dst-DU, src-AU or dst-DU, src-DU

2) dst-DU, src-AU or dst-AU, src-DU

# Chapter 5

# **Instruction Set Descriptions**

This chapter provides detailed information on the TMS320C55x<sup>™</sup> DSP algebraic instruction set.

See Section 1.1, *Instruction Set Terms, Symbols, and Abbreviations*, for definitions of symbols and abbreviations used in the description of each instruction. See Chapter 4 for a summary of the instruction set.

### Absolute Distance

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	abdst(Xmem, `	Ymem, ACx, ACy <b>)</b>	No	4	1	Х
Opcod	e	1000 0110 XXXX	M MMYY YMM	im di	DD 111	1 xxn%
Operar	nds	ACx, ACy, Xmem, Ymem	·		·	
Descri	ption	This instruction executes two operation one in the D-unit ALU:	ons in parallel:	one in	the D-uni	t MAC and
		ACy = ACy +  HI(ACx)  ACx = (Xmem << #16) - (Ymem <<	< #16)			
		The absolute value of accumulator accumulator ACy content through detected according to M40:				
		the destination accumulator ove	rflow status bit	(ACC	Vy) is se	t
		the destination register (ACy) is	saturated acco	ording	to SATD	
		The Ymem content shifted left 16 bi shifted left 16 bi shifted left 16 bits in the D-unit ALU.		d from	the Xme	em content
		Input operands (Xmem and Yme to SXMD.	m) are sign ext	ended	to 40 bits	according
		CARRY status bit depends on M CARRY status bit. It is the logical				•
		When an overflow is detected a	ccording to M4	0:		
		the destination accumulator	overflow statu	s bit (/	ACOVx) i	s set
		the destination register (AC	<li>k) is saturated</li>	accore	ding to SA	ATD
		Compatibility with C54x devices (	C54CM = 1)			
		When this instruction is executed wi	th M40 = 0, co	mpatik	oility is en	sured.
		When C54CM = 1, the subtract of detection, report, and saturation after	•		-	y overflow
Status	Bits	Affected by C54CM, FRCT, M40	), SATD, SXMI	D		
		Affects ACOVx, ACOVy, CA	ARRY			
Repeat	t	This instruction can be repeated.				

### See Also

See the following other related instructions:

### Square Distance

Syntax			Descript	ion			
abdst(*AR0	+, *AR1, A	AC0, AC1)	AC1 and subtracte	the res ed from	ult is s the co	stored in ntent a	tent of AC0 is added to the content of n AC1. The content addressed by AR1 is ddressed by AR0 and the result is stored s incremented by 1.
Before			After				
AC0	00 000	0 0000	AC0	00	4500	0000	
AC1	00 E80	0 0000	AC1	00	E800	0000	
AR0		202	AR0			203	
AR1		302	AR1			302	
202		3400	202			3400	
302		EF00	302			EF00	
ACOV0		0	ACOV0			0	
ACOV1		0	ACOV1			0	
CARRY		0	CARRY			0	
M40		1	M40			1	
SXMD		1	SXMD			1	

### Absolute Value

No.	Syntax																													-													E							lle e			t		s	iz	ze	9	(	С	у	cl	е	s			Р	iŗ	)(	el	ir	٦e
[1]	dst =  src																																														١	Y	e	s						2						1							)	<		
Opcod	e																																																	1	0	)(	)1	.1	_		0	0	1	E	1	1	F	S	S	55	5		F	D	ΡĽ	DI
Operai	nds	dst, s	rc																																																																					
Descri	ption	This i	nstru	ruc	IC	cl	ct	;1	:1	:1	ct	;t	ti	i	С	DI	n	n	C	C	:0	or	m	пр	bu	ıt	e	98	s	5	t	th	ne	2	a	b	s	0	lu	J	te	е	١	V	2	al	lι	Je	е	С	f	t	h	е	s	0	u	rc	ce	<b>;</b>	re	эĉ	lis	st	e	er	(	sı	ſĊ	;)		
		D W	Vhen	n t	tł	tŀ	th	ł	ł	ł	ł	h	n	ie	e	)	(	d	de	e	S	sti	ir	าอ	at	ic	DI	r	n	1	re	е	g	is	st	e	er	(	d	ls	st	:)	i	is	5	6	a	n	1	a	C	CI	JI	n	u	la	1te	ıc														
				Γhe	ne	e	e	е	Э	Э	э	Э	è	(	0	p	p	ЭС	e	ər	ra	at	tio	01	n	i	s	;	F	p	e	e	rfo	DI	rr	n	ne	ec	ł	0	or	n	2	4	С	)	k	зi	it	s	ir	n	tł	١e	<b>)</b>	D	)-1	Jr	٦İ	t	Α	L	U	١.								
			in	fa ns ext	st	sti	tı	tı	tı	tı	tı	tr	r	ι	u	10	C	ct	tio	0	n	٦,	, 1	th	ne	)	1	6	6	5	L	_3	SI	B	s	. (	of	ft	h	ie	Э	а	a	u	D	¢	ili	ia	aı	ry	C	10																				
			3	f N 31. he	۱.	.	.				l	I	ľ	f	F	S	SI	r	С	c(	(3	3	1)	) :	=	1	1,	,	t	th	h	e	9 8	sc	וכ	J	rc	ce	9	re	e	g	ji:	s	st	te	Э	r	С	0	n	te	er	nt	is	SI	n	еą	ga	at	e	d	.	lf	S	sr	С	(3	81	)	=	=
			39	f N 39. he	Э.	.	. I				l	I	ľ	f	f	S	SI	r	С	)(	(3	39	9	) :	=	1	1,	,	t	th	n	e	9 5	SC	ונ	J	rc	Ce	)	re	e	g	ji	s	st	e	Э	r	С	0	n	te	er	nt	is	SI	n	еą	ga	at	e	d	.	lf	S	sr	С	(3	36	))	=	=
				Du are													-																				-								٦,	, ;	а	ar	n	0	v	e	rf	lc	V	v	a	n	d	0	2	41	R	F	<i>'</i> `	Y	b	it	S	sta	a	tı
			•										-	Т	Г	ŀ	h	١e	е	• (	d	le	95	sti	in	าอ	al	ti	ic	0	or	n	а	C	:0	χ	Jr	n	u	I	a	t	0	or	ſ	С	D١	V	е	rf	lc	)\	N	s	ta	at	u	s	b	it		Ά	C	;(	С	٧	/x	:)	is	5	s	e
			-										-	Т	Г	ŀ	h	١e	е	• (	d	le	95	sti	in	าอ	al	ti	ic	0	or	n	r	e	g	is	st	e	er	i	s	5 :	s	66	a	tι	u	ır	а	te	ЭС	t	а	C	C	IC	ſĊ	lir	١Ç	)	to	)	S	Α			).					
			-										(	С	D	p	p	e	eı	ra	a	ıti	ic	or	RF n e,	s	st	c	C	r	е	ЭС	d	ir	n	t	th	e	¢	d	le	98	s	ti	ir	n	а																									
			Vhen	n t	tł	tŀ	th	ł	ł	ł	ł	h	h	ie	e	9	C	d	de	e	S	sti	ir	าอ	at	ic	DI	r	n	I	re	е	g	is	st	e	er	(	d	ls	st	:)	i	is	5	6	a	n	1	a	u	x	li	a	ſy	' (	DI	• t	e	n	٦I	20	or	а	۱r	y	r	e	gi	s	ste	e
				Γhe	٦e	e	e	e	Э	Э	Э	Э	è	C	0	D	p	Э	e	er	ra	at	tio	01	n	i	s	;	F	p	e	er	rfo	DI	rr	n	e	ec	ł	0	or	า	1	1	6	5	k	ci	it	s	ir	n	tł	ne	),	A	-l	Jr	nit	t.	A	L	U	•								
				fa oft																																								•																			n,	, 1	th	١e	) <sup>,</sup>	16	5	L	S	SE
		-	sr so	The src sou s c	c( bu	c( u	:( 	) 1	) ג	) ג	) וו	( 1	(1	1	c	5	5) e	) )	= r	= e	= eg	1 gi	, is	tl ste	he e	e r	c	s	0	o or	n n	ur nte	rc e	e n	e t	re is	eę s	gi m	is no	st C	e v	er	. (	С	c	D	n	nt	e	n	t	is	; 1	ne	eč	ja	at	e	d.		f	s	rc	;(	1	5	5)	=	: (	Э,	, †	tł
				Γhe	٦e	e	e	e	Э	Э	Э	Э	è	C	d	b	e	Э	s	sti	ir	n	а	ti	0	n	۱	r	ſ	e	eg	gi	is	te	e	ſ	is	; ;	s	a	t	u	I	6	at	te	е	d	ł	a	C	C	SI	d	ir	١Ç	J	tc	) (	S	Α	Т	A									

### Compatibility with C54x devices (C54CM = 1)

When C54CM =1, this instruction is executed as if M40 status bit was locally set to 1. To ensure compatibility versus overflow detection and saturation of destination accumulator, this instruction must be executed with M40 = 0.

Status Bits	Affected by	C54CM, M40, SATA, SATD, SXMD
	Affects	ACOVx, CARRY
Repeat	This instruction	can be repeated.
See Also	See the following	ng other related instructions:
	Addition wi	th Absolute Value

### Example 1

•					
Syntax		Desc	ription		
AC1 =  AC0		The a	absolut	e value of the cor	ntent of AC0 is stored in AC1.
Before				After	
AC1	00	0000	2000	AC1	7D FFFF EDCC
AC0	82	0000	1234	AC0	82 0000 1234
M40			1	M40	1

### Example 2

<u> </u>					
Syntax		Desc	ription		
AC1 =  AR1	The a	absolute v	alue of the con	tent of AR1 is stored in AC1.	
Before				After	
AC1	00	0000	2000	AC1	00 0000 0000
AR1			0000	AR1	0000
CARRY			0	CARRY	1

### Example 3

Syntax		Desc	ription								
AC1 =  AR1		The absolute value of the content of AR1 is stored in AC1. Since SXMD = 1, AR1 content is sign extended. The resulting 40-bit data is negated since M40 = 0 and AR1(31) = 1.									
Before				Afte	er						
AC1	00	0000	2000	AC1	00 0000 7900						
AR1			8700	AR1	8700						
M40			0	M40	0						
SXMD			1	SXMI	D 1						

### SPRU375G

Instruction Set Descriptions 5-5

### Example 4

Syntax	Descr	Description							
T1 =  AC0		The absolute value of the content of AC0(15–0) is stored in T1. The sign bit is extracted at AC0(15). Since AC0(15) = 0, T1 = AC0(15–0).							
Before			After						
T1	2	2000	Τ1		1234				
AC0	80 0002 1	L234	AC0	80 0002	1234				

Syntax	Description
T1 =  AC0	The absolute value of the content of AC0(15–0) is stored in T1. The sign bit is extracted at AC0(15). Since AC0(15) = 1, T1 equals the negated value of AC0(15–0).
Before	After

Before			After		
Т1		2000	Т1		6DCC
AC0	80 0002	9234	AC0	80 0002	9234

### **Syntax Characteristics**

		Darallal			
No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst + src	Yes	2	1	Х
[2]	dst = dst + k4	Yes	2	1	Х
[3]	dst = src + K16	No	4	1	Х
[4]	dst = src + Smem	No	3	1	Х
[5]	ACy = ACy + (ACx << Tx)	Yes	2	1	Х
[6]	ACy = ACy + (ACx << #SHIFTW)	Yes	3	1	Х
[7]	ACy = ACx + (K16 << #16)	No	4	1	Х
[8]	ACy = ACx + <b>(</b> K16 <b>&lt;&lt; #</b> SHFT <b>)</b>	No	4	1	Х
[9]	ACy = ACx + (Smem << Tx)	No	3	1	Х
[10]	ACy = ACx + (Smem << #16)	No	3	1	Х
[11]	ACy = ACx + uns(Smem) + CARRY	No	3	1	Х
[12]	ACy = ACx + uns(Smem)	No	3	1	Х
[13]	ACy = ACx + <b>(</b> uns(Smem) << <b>#</b> SHIFTW <b>)</b>	No	4	1	Х
[14]	ACy = ACx + <b>dbl(</b> Lmem <b>)</b>	No	3	1	Х
[15]	ACx = (Xmem << #16) + (Ymem << #16)	No	3	1	Х
[16]	Smem = Smem + K16	No	4	1	Х

### Description

These instructions perform an addition operation.

Status Bits

. . .

Affected by CARRY, C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

See Also	See the following other related instructions:				
	Addition or Subtraction Conditionally				
	Addition or Subtraction Conditionally with Shift				
	Addition with Absolute Value				
	Addition with Parallel Store Accumulator Content to Memory				
	Addition, Subtraction, or Move Accumulator Content Conditionally				
	Dual 16-Bit Additions				
	Dual 16-Bit Addition and Subtraction				
	Dual 16-Bit Subtraction and Addition				
	Subtraction				

No.	Syntax									allel ole Bit	Size	Cycles	Pipe	line
[1]	dst = dst + src									es	2	1	×	
Opcode										001	10 01	OE FSS	SS FI	DDD
Operan		dst	, src											
Descrip				uction	n perfo	orms a	an ado	dition	operat	ion be <sup>.</sup>	tween	two regis	ters.	
•			When						•			•		
		· <b></b>					. ,	•				nit ALU.		
												ording to	SXME	D.
			in	struc	•	he 16	LSBs	of the	auxilia			(src) ope ary regis		
				verflo	ow det	tectio	n and	CAR	RY sta	tus bit	depen	ds on M4	10.	
				/hen a		erflow	is det	ected,	, the ac	cumul	ator is	saturated	lacco	rdin
			When	the c	destin	ation	(dst) d	operai	nd is a	n auxil	liary or	tempora	ry reg	iste
			∎ Tł	he op	peratio	on is p	perforr	ned o	n 16 b	its in tł	ne A-u	nit ALU.		
									•	· ·		the instr the oper		
			■ Ao	dditio	on ove	rflow	detec	tion is	done	at bit p	oositio	n 15.		
					an ov ling to			etecte	d, the	destin	ation r	egister is	s satu	rate
		Co	mpatik	bility	with	C54x	devid	es (C	54CM	= 1)				
		Wh	en this	s instr	ructior	n is ex	ecute	d with	י M40	= 0, cc	mpatil	oility is er	nsurec	ł.
Status	Bits	Aff	ected b	у	M40	), SAT	A, SA	TD, S	XMD					
		Aff	ects		ACC	OVx, C	CARR	Y						
		Th	s instru	uction	n can I	be rep	peated	ł.						
Repeat														
Repeat Examp	le													

Addition

No Suntay		Parallel	Size	Cuoles	Dinalina
No. Syntax		Enable Bit		Cycles	Pipeline
[2] dst = dst + k4		Yes	2	1	Х
Opcode		010	00 00	0E kkk	k FDDD
Operands	dst, k4				
Description	This instruction performs an addition a 4-bit unsigned constant, k4.	operation betw	veen a	register c	ontent and
	When the destination (dst) oper	and is an accu	umulate	or:	
	The operation is performed	on 40 bits in t	he D-u	nit ALU.	
	<ul> <li>Overflow detection and CAI</li> </ul>	RRY status bit	depen	ds on M4	0.
	When an overflow is detecter to SATD.	d, the accumul	lator is	saturatec	laccording
	When the destination (dst) oper	and is an auxi	liary or	tempora	ry register
	The operation is performed	on 16 bits in t	he A-u	nit ALU.	
	<ul> <li>Addition overflow detection</li> </ul>	is done at bit p	oositio	n 15.	
	When an overflow is detect according to SATA.	ted, the destin	ation r	egister is	saturate
	Compatibility with C54x devices	(C54CM = 1)			
	When this instruction is executed with	ith M40 = 0, co	ompatil	oility is er	sured.
Status Bits	Affected by M40, SATA, SATD				
	Affects ACOVx, CARRY				
Repeat	This instruction can be repeated.				
Example					

Syntax	Description
AC0 = AC0 + k4	The content of AC0 is added to an unsigned 4-bit value and the result is stored in AC0.

						Parallel			
No.	Syntax					Enable Bit	Size	Cycles	Pipeline
[3]	dst = src + K16					No	4	1	Х
Opcod	9			0111	1011 ккк	к кккк кк	кк ки	KK FDI	DD FSSS
Operar	nds	dst, K1	I6, src						
Descri	otion		structior it signec	•		operation bet	ween a	register c	content and
		D W	hen the	destinat	ion (dst) opei	and is an acc	umulate	or:	
			The op	peration	is performed	on 40 bits in t	he D-u	nit ALU.	
		•	instruc	tion, the		register is the ne auxiliary or ID.		• • •	
			The 10 SXMD		nstant, K16, i	s sign extend	ed to 4	l0 bits ad	ccording to
			Overflo	ow dete	ction and CA	RRY status bit	depen	ds on M4	40.
			When to SAT		low is detecte	d, the accumu	lator is	saturated	laccording
		D W	hen the	destinat	ion (dst) opei	and is an aux	liary or	tempora	ry register
			The op	peration	is performed	on 16 bits in t	he A-u	nit ALU.	
		-				urce (src) ope r are used to p			
			Additic	on overfl	ow detection	is done at bit	positio	า 15.	
		-		an over ling to S		ted, the destir	nation r	egister is	saturated
		Сотр	atibility	with C	54x devices	(C54CM = 1)			
		When	this inst	ruction is	s executed w	ith M40 = 0, c	ompatil	oility is er	nsured.
Status	Bits	Affecte	ed by	M40, \$	SATA, SATD,	SXMD			
		Affects	6	ACOV	′x, CARRY				
Repeat	:	This in	structior	n can be	repeated.				
Examp	le								
Syntax	,	Deser	iption						

AC1 = AC0 + #2E00h	The content of AC0 is added to the signed 16-bit value (2E00h) and the result is
	stored in AC1.

Addition

<u> </u>					Parallel	<u></u>	0	Dinelin
•							-	Pipeline
dst = src + Sme	m				No	3	1	Х
e				1101	0110 AA	AA AA	AI FDI	D FSSS
nds	dst, S	mem, sro	C					
ption						veena	register c	ontent and
		/hen the	destination (	lst) operar	nd is an accu	umulate	or:	
		The op	peration is pe	rformed or	n 40 bits in tl	he D-u	nit ALU.	
		instruc	ction, the 16 L	SBs of the	auxiliary or t		• • •	
					location is	sign e	extended	to 40 bits
		Overflo	ow detection	and CARF	RY status bit	depen	ds on M4	0.
				detected,	the accumul	ator is	saturateo	laccording
		/hen the	destination (d	lst) operan	nd is an auxi	liary or	tempora	ry register:
		The op	peration is pe	rformed or	n 16 bits in tl	he A-u	nit ALU.	
		Additic	on overflow d	etection is	done at bit p	oositio	n 15.	
				is detected	d, the destin	ation r	egister is	saturated
	Com	oatibility	with C54x a	levices (C	54CM = 1)			
	Wher	this inst	ruction is exe	cuted with	M40 = 0, co	ompatil	oility is er	sured.
Bits	Affect	ed by	M40, SATA	, SATD, SX	XMD			
	Affect	S	ACOVx, CA	ARRY				
	Syntax dst = src + Sme e nds ption Bits	dst = src + Smem e nds dst, S ption This in the co W	dst = src + Smem e nds dst, Smem, sra ption This instruction the content of When the The op If an a instruct extend The op If an a instruct extend Overfil When the The op If an a instruct extend When the The op If an a 16 LSI Addition When this inst Bits Affected by	dst = src + Smem         e         nds       dst, Smem, src         ption       This instruction performs an the content of a memory (S         Image: Description       This instruction performs an the content of a memory (S         Image: Description       This instruction performs an the content of a memory (S         Image: Description       This instruction performs an the content of a memory (S         Image: Description       This instruction performs an the content of a memory (S         Image: Description       The operation is performed according to SXMD         Image: Description       The content of the according to SXMD         Image: Description       When the destination (c         Image: Description       The operation is performed according to SATD.         Image: Description       Image: Description operformed according to SATA.         Image: Description       The operation is performed according to SATA.         Image: Description       The operation is exerced according to SATA.         Image: Description       The operation is exerced according to SATA.         Image: Description       The operation is exerced according to SATA.         Image: Description       The operation is exerced according to SATA.         Image: Description       The operation is exerced according to SATA.         Image: Description       The	dst = src + Smem       1101         nds       dst, Smem, src         ption       This instruction performs an addition of the content of a memory (Smem) loca         Image: When the destination (dst) operar         Image: The operation is performed on         Image: The operation is detected acc	Syntax         Enable Bit           dst = src + Smem         No           e          1101_0110_AA           nds         dst, Smem, src           ption         This instruction performs an addition operation betwee the content of a memory (Smem) location.           Image: The operation is performed on 40 bits in the operation is performed on 40 bits in the farma auxiliary or temporary register is the second according to SXMD.           Image: The content of the memory location is according to SXMD.           Image: The operation is performed on 40 bits in the testination (dst) operand is an auxiliary or textended according to SXMD.           Image: The content of the memory location is according to SXMD.           Image: The operation is detected, the accumulation overflow detection and CARRY status bit.           Image: When the destination (dst) operand is an auxiliary or the operation is performed on 16 bits in the testination (dst) operand is an auxiliary to sATD.           Image: The operation is performed on 16 bits in the farma accumulator is the source (src) operation is performed on 16 bits in the farma according to SATA.           Image: Compatibility with C54x devices (C54CM = 1)           When this instruction is executed with M40 = 0, colored according to SATA.           Compatibility with C54x devices (C54CM = 1)           When this instruction is executed with M40 = 0, colored according to SATA.	Syntax         Enable Bit         Size           dst = src + Smem         No         3           e          1101         0110         AAAA         AA           nds         dst, Smem, src         Indiana         AAAA         AA           ption         This instruction performs an addition operation between a the content of a memory (Smem) location.         Image: Size of the accumulate           Image: Display the destination (dst) operand is an accumulate         The operation is performed on 40 bits in the D-u         If an auxiliary or temporary register is the source instruction, the 16 LSBs of the auxiliary or tempor extended according to SXMD.         Image: Display the destination (dst) operand is an auxiliary or extended according to SXMD.         Image: Display the destination (dst) operand is an auxiliary or extended according to SXMD.         Image: Display the destination (dst) operand is an auxiliary or extended on the destination (dst) operand is an auxiliary or extended according to SXMD.         Image: Display the destination (dst) operand is an auxiliary or extended according to SXMD.         Image: Display the destination (dst) operand on the bits in the A-u         Image: Display the destination (dst) operand is an auxiliary or extended according to SATD.           Image: Display the destination (dst) operand on the bits in the A-u         Image: Display the destination (dst) operand on the bits in the A-u           Image: Display the destination of the accumulator is the source (src) operand of the LSBs of the accumulator is the source (src) operand of the LSBs of the accumulator is detec	Syntax         Enable Bit         Size         Cycles           dst = src + Smem         No         3         1           e          1101         0110         AAAA         AAAI         FDE           Inds         dst, Smem, src         India         AAAI         FDE         India         AAAI         FDE           Inds         dst, Smem, src         This instruction performs an addition operation between a register of the content of a memory (Smem) location.         India         When the destination (dst) operand is an accumulator:         India         The operation is performed on 40 bits in the D-unit ALU.         If an auxiliary or temporary register is the source (src) ope instruction, the 16 LSBs of the auxiliary or temporary register extended according to SXMD.         Inthe content of the memory location is sign extended according to SXMD.         Overflow detection and CARRY status bit depends on M4         When an overflow is detected, the accumulator is saturated to SATD.         Inthe operation is performed on 16 bits in the A-unit ALU.         If an accumulator is the source (src) operand of the instruct 16 LSBs of the accumulator are used to perform the oper           In the operation is performed on 16 bits in the A-unit ALU.         If an accumulator is the source (src) operand of the instruct 16 LSBs of the accumulator are used to perform the oper         Addition overflow detection is done at bit position 15.         When an overflow is detected, the destination register is according to SATA.           Compatibility with C54x

## **Repeat** This instruction can be repeated.

Syntax		Description	Description					
T1 = T0 + *AR3+ stored in T1. AR3 is increme				ent addressed by AR3 and the result is				
Before		After						
AR3	0302	AR3	0303					
302	EF00	302	EFOO					
т0	3300	Т0	3300					
Т1	0	T1	2200					
CARRY	0	CARRY	1					

### Addition

### Syntax Characteristics

No. Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
$[5] \qquad ACy = ACy + (A)$	ACx << Tx			Yes	2	1	X
Opcode				010	)1 10	1E DDS	S ss00
Operands	ACx, A	y, Tx					
Description		truction performs ACy and an accum					
	☐ The operation is performed on 40 bits in the D-unit shifter.						
	🗋 Inp	Input operands are sign extended to 40 bits according to SXMD.					
	The shift operation is equivalent to the signed shift instruction.						
	Overflow detection and CARRY status bit depends on M40.						
	D Wh	n an overflow is de D.	etected, the	accumulato	r is sat	urated ad	ccording to
	Compa	ibility with C54x o	devices (C5	54CM = 1)			
	When the C54CN	s instruction is exe = 1:	cuted with N	140 = 0, com	patibili	ty is ensu	ired. When
	no	termediary shift op verflow detection, ation.	•			-	
	Tx	6 LSBs of Tx are u efine a shift quantit 7, a modulo 16 op	y within –32	to +31. Whe	n the v	alue is be	tween –32
Status Bits	Affecte	by C54CM, M	140, SATD, S	SXMD			
	Affects	ACOVy, C	ARRY				
Repeat	This ins	ruction can be rep	eated.				

Syntax	Description
( /	The content of AC1 shifted by the content of T0 is added to the content of AC0 and the result is stored in AC0.

### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[6]	ACy = ACy + c	(ACx << #SHIFTW)		Yes	3	1	Х			
Opcod	e		0001	000E DDS	SS 00	11 xxS	SH IFTW			
Operar	nds	ACx, ACy, SH	IFTW							
Descrij	ption		This instruction performs an addition operation between an accumulator content ACy and an accumulator content ACx shifted by the 6-bit value, SHIFTW.							
		The operation is performed on 40 bits in the D-unit shifter.								
		Input operands are sign extended to 40 bits according to SXM								
		The shift o	peration is equivalent to	o the signed	shift ir	struction				
		Overflow of the second seco	detection and CARRY st	tatus bit dep	ends o	n M40.				
		When an o SATD.	overflow is detected, the	accumulato	r is sat	urated ad	ccording to			
		Compatibility	with C54x devices (C	54CM = 1)						
		C54CM = 1, a	ruction is executed with N n intermediary shift ope o overflow detection, re ion.	ration is per	formec	as if M4	0 is locally			
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD						
		Affects	ACOVy, CARRY							
Repeat	t	n can be repeated.								

Syntax	Description
· · · · · · · · · · · · · · · · · · ·	The content of AC1 shifted left by 31 bits is added to the content of AC0 and the result is stored in AC0.

### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx + 0	(K16 <b>&lt;&lt; #16)</b>		No	4	1	Х
Opcod	e		0111 1010 KKKK	KKKK KKF	кк кк	KK SSI	D 000x
Operar	nds	ACx, ACy, K16	3				
Descri	ption		n performs an additior nd a 16-bit signed const	•			
		The opera	tion is performed on 40	bits in the D	-unit A	LU.	
		Input oper	ands are sign extended	to 40 bits ad	ccordin	ig to SXN	ID.
		The shift o	peration is equivalent to	o the signed	shift ir	struction	
		Overflow of the second seco	detection and CARRY st	tatus bit dep	ends o	n M40.	
		When an o SATD.	overflow is detected, the	accumulato	r is sat	urated ad	ccording to
		Compatibility	with C54x devices (C	54CM = 1)			
		C54CM = 1, a	ruction is executed with N n intermediary shift ope o overflow detection, re ion.	ration is per	formed	as if M4	0 is locally
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD			
		Affects	ACOVy, CARRY				
Repeat	t	This instruction	n can be repeated.				

Syntax	Description
AC0 = AC1 + (#2E00h << #16)	A signed 16-bit value (2E00h) shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0.

### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[8]	ACy = ACx +	<b>(</b> K16 <b>&lt;&lt; #</b> SHFT <b>)</b>		No	4	1	Х	
Opcod	e		0111 0000 кккк	KKKK KKI	K KK	KK SSI	DD SHFT	
Operai	nds	ACx, ACy, K16	6, SHFT					
Descri	ption		n performs an addition nd a 16-bit signed const	•				
		The opera	tion is performed on 40	bits in the D	-unit s	hifter.		
		Input oper	ands are sign extended	to 40 bits a	ccordir	ig to SXN	/ID.	
		The shift o	peration is equivalent to	o the signed	shift ir	struction		
		Overflow of the second seco	detection and CARRY s	tatus bit dep	ends o	n M40.		
		When an o SATD.	overflow is detected, the	accumulato	r is sat	urated a	ccording to	
		Compatibility with C54x devices (C54CM = 1)						
		C54CM = 1, a	ruction is executed with I n intermediary shift ope o overflow detection, re ion.	ration is per	formec	as if M4	0 is locally	
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD				
		Affects	ACOVy, CARRY					
Repea	t	This instruction	n can be repeated.					
	_							

Syntax	Description
	A signed 16-bit value (2E00h) shifted left by 15 bits is added to the content of AC1 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit Size Cycles Pipeline						
[9]	ACy = ACx + (	Smem << Tx)	No 3 1 X						
Opcode	e	1	.101 1101 AAAA AAAI SSDD ss00						
Operar	nds	ACx, ACy, Tx, Smem	ACx, ACy, Tx, Smem						
Descriț	ption		This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted by the content of Tx.						
		☐ The operation is performed of	The operation is performed on 40 bits in the D-unit shifter.						
		Input operands are sign external	ended to 40 bits according to SXMD.						
		The shift operation is equival	lent to the signed shift instruction.						
		Overflow detection and CARRY status bit depends on M40.							
		When an overflow is detected, the accumulator is saturated according to SATD.							
		Compatibility with C54x devices (C54CM = 1)							
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1:							
		_ , ,	on is performed as if M40 is locally set to 1 and rt, and saturation is done after the shifting						
		Tx define a shift quantity with	o determine the shift quantity. The 6 LSBs of in –32 to +31. When the value is between –32 on transforms the shift quantity to within –16						
Status	Bits	Affected by C54CM, M40, S	ATD, SXMD						
		Affects ACOVy, CARRY	,						
Repeat	:	This instruction can be repeated							

5-18 Instruction Set Descriptions

Syntax		Description	
AC0 = AC1	+ (*AR1 << T0)		dressed by AR1 shifted left by the content of T0 is added to the and the result is stored in AC0.
Before		After	
AC0	00 0000 000	D AC0	00 2330 0000
AC1	00 2300 000	D AC1	00 2300 0000
Т0	000	с то	000C
AR1	020	) AR1	0200
200	030	200	0300
SXMD		) SXMD	0
M40		D M40	0
ACOV0		) ACOV0	0
CARRY		) CARRY	1

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[10]	ACy = ACx + (S	Smem <b>&lt;&lt; #16)</b>		No	3	1	Х	
Opcod	e		1101	1110 AAA	AA AA	AI SSD	D 0100	
Operar	nds	ACx, ACy, Sm	em					
Descrij	otion	This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location shifted left by 16 bits.						
		The opera	tion is performed on 40	bits in the D	-unit A	LU.		
		Input operation	ands are sign extended	to 40 bits a	ccordin	ig to SXM	ID.	
		The shift o	peration is equivalent to	o the signed	shift in	struction		
		of the addi	detection and CARRY s ition generates a carry, t Y status bit is not affect	the CARRY s				
		When an o SATD.	overflow is detected, the	accumulato	r is sat	urated ac	cording to	
		Compatibility	with C54x devices (C	54CM = 1)				
		C54CM = 1, a	ruction is executed with N n intermediary shift ope o overflow detection, re ion.	ration is per	formed	as if M4	0 is locally	
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD				
		Affects	ACOVy, CARRY					
Repeat	:	This instruction can be repeated.						

Syntax	Description
AC0 = AC1 + (*AR3 << #16)	The content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ACy = ACx + uns	<mark>s(</mark> Sme	em <mark>) + CAR</mark> I	RY		No	3	1	х
Opcode					1101	1111 AAA	AA AA	AI SSI	D 100u
Operand	s	ACx,	ACy, Sm	em					
Descript	ion	This instruction performs an addition operation of the accumulator content ACx, the content of a memory (Smem) location, and the value of the CARRY status bit.							
			The operat	tion is performe	d on 40	bits in the D	-unit A	LU.	
			nput opera	ands are extend	ed to 4	0 bits accord	ling to	uns.	
		I		ptional uns keyw memory locatior		••	•	•	he content
		I		optional uns key at of the memory		••		• •	-
			Overflow d	letection and CA	ARRY s	tatus bit dep	ends o	n M40.	
		_	When an c SATD.	overflow is detec	ted, the	e accumulato	or is sat	urated ac	ccording to
		Com	patibility	with C54x dev	ices (C	54CM = 1)			
		Whe	n this insti	ruction is execut	ed with	n M40 = 0, co	ompatik	oility is en	sured.
Status B	its	Affec	ted by	CARRY, M40,	SATD,	SXMD			
		Affec	sts	ACOVy, CARF	RY				
Repeat		This instruction can be repeated.							

Syntax	Description
	The CARRY status bit and the unsigned content addressed by AR3 are added to the content of AC1 and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[12]	ACy = ACx + t	u <mark>ns(</mark> Smem)		No	3	1	Х		
Opcod	e		1101	1111 AAA	AA AA	AI SSI	DD 110u		
Operar	nds	ACx, ACy, S	Smem						
Descri	ption		This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location.						
		📋 The op	eration is performed on 40	bits in the D	)-unit A	LU.			
		🗋 Input o	perands are extended to 4	0 bits accord	ling to	uns.			
			e optional uns keyword is a he memory location is zero		•	•	he content		
		cor	ne optional uns keyword is tent of the memory location MD.	••		• •	-		
		Overflo	w detection and CARRY s	tatus bit dep	ends c	on M40.			
		When a SATD.	an overflow is detected, the	accumulato	or is sa	turated a	ccording to		
		Compatibi	lity with C54x devices (C	54CM = 1)					
		When this i	nstruction is executed with	M40 = 0, co	ompatil	oility is er	sured.		
Status	Bits	Affected by	M40, SATD, SXMD						
		Affects	ACOVy, CARRY						
Repeat	t	This instruc	tion can be repeated.						
Evom	1.								

Syntax	Description
AC0 = AC1 + uns(*AR3)	The unsigned content addressed by AR3 is added to the content of AC1 and the result is stored in AC0.

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[13]		(uns(Smem) <-	< #SHIFTW)	No	4	1	X	
Opcod	9		1111 1001 AA	AAA AAAI ux	SH IE	TW SSI	D 00xx	
Operar	nds	ACx, ACy,	SHIFTW, Smem	, i		·		
Description			uction performs an ad Cx and the content of a r FTW.	•				
		🗋 The op	peration is performed o	n 40 bits in the D	)-unit s	hifter.		
		🗋 Input d	operands are extended	to 40 bits accord	ding to	uns.		
		If the optional uns keyword is applied to the input ope of the memory location is zero extended to 40 bits.						
		со	the optional uns keywc ntent of the memory loc (MD.			• •	-	
		🗋 The sh	nift operation is equivale	ent to the signed	shift ir	nstruction		
		🗋 Overfle	Overflow detection and CARRY status bit depends on M40.					
		When SATD.	an overflow is detected	l, the accumulato	or is sat	turated a	ccording t	
		Compatib	ility with C54x device	s (C54CM = 1)				
		C54CM =	instruction is executed v 1, an intermediary shift nd no overflow detection eration.	operation is per	formec	as if M4	0 is locall	
Status	Bits	Affected by	y C54CM, M40, SA	TD, SXMD				
		Affects	ACOVy, CARRY					
Repeat		This instruction cannot be repeated when using the *(#k23) absolute a ing mode to access the memory operand (Smem); when using other a ing modes, this instruction can be repeated.						

Syntax	Description
AC0 = AC1 + (uns(*AR3) << #31)	The unsigned content addressed by AR3 shifted left by 31 bits is added to the content of AC1 and the result is stored in AC0.

# Syntax Characteristics

Ne	Syntax					Parallel	Sino	Cycles	Dipolino
No.	Syntax	1/1				Enable Bit	Size	Cycles	Pipeline
[14]	ACy = ACx + dl	ol(Lm	iem)			No	3	1	Х
Opcode	9				1110	1101 AAA	AA AA	AI SSI	D 000n
Operands			x, ACy, Lm	nem					
Description			This instruction performs an addition operation between an accumulator content ACx and the content of data memory operand dbl(Lmem).						
			The data	memory operand	d dbl(Ln	nem) addres	ses are	e aligned:	:
				nem address is icant word = Lme		most signific	ant wo	ord = Ln	nem, least
				nem address is icant word = Lme		nost signific	ant wo	ord = Ln	nem, least
			The opera	ation is performe	d on 40	bits in the D	-unit A	LU.	
			Input ope	rands are sign ex	ktended	I to 40 bits a	ccordin	g to SXN	1D.
			Overflow	detection and CA	ARRY s	tatus bit dep	ends o	n M40.	
			When an SATD.	overflow is detec	ted, the	e accumulato	r is sat	urated ad	ccording to
		Со	mpatibilit	y with C54x dev	ices (C	54CM = 1)			
		Wh	en this ins	truction is execu	ted with	M40 = 0, co	mpatik	oility is en	sured.
Status	Bits	Affe	ected by	M40, SATD, S	SXMD				
		Affe	ects	ACOVy, CARI	۲Y				
Repeat		Thi	s instructio	on can be repeate	ed.				

Syntax	Description
	The content (long word) addressed by AR3 and AR3 + 1 is added to the content of AC1 and the result is stored in AC0. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[15]	ACx = (Xmem	<< <b>#16)</b> + <b>(</b> Ymem	<< #16)	No	3	1	Х			
Opcod	е		1000	0001 XXX	KM MM	IYY YMM	IM 00DD			
Operar	nds	ACx, Xmem, Y	'mem							
Descri	ption	This instruction performs an addition operation between the content of data memory operand Xmem shifted left 16 bits, and the content of data memory operand Ymem shifted left 16 bits.								
		The operation is performed on 40 bits in the D-unit ALU.								
		Input operands are sign extended to 40 bits according to SXMD.								
		The shift operation is equivalent to the signed shift instruction.								
		Overflow detection and CARRY status bit depends on M40.								
		When an overflow is detected, the accumulator is saturated according to SATD.								
		Compatibility								
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.								
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD						
		Affects	ACOVx, CARRY							
Repeat	t	This instruction	n can be repeated.							
_	_									

Syntax	Description
, , , , ,	The content addressed by AR3 shifted left by 16 bits is added to the content addressed by AR4 shifted left by 16 bits and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline				
[16]	Smem = Smer	n + K16		No	4	1	Х				
Opcod	e		1111 0111 AAAA	AAAI KKI	KK KK	кк   ккк	к кккк				
Operar	nds	K16, Smem									
Descri	ption		This instruction performs an addition operation between a 16-bit signed constant, K16, and the content of a memory (Smem) location.								
		The oper	The operation is performed on 40 bits in the D-unit ALU.								
		Input operands are sign extended to 40 bits according to SXMD and shifted by 16 bits to the MSBs before being added.									
		Addition overflow is detected at bit position 31. If an overflow is detected, accumulator 0 overflow status bit (ACOV0) is set.									
		□ Addition carry report in CARRY status bit is extracted at bit position 31.									
		☐ If SATD is 1 when an overflow is detected, the result is saturated before being stored in memory. Saturation values are 7FFFh or 8000h.									
		Compatibility with C54x devices (C54CM = 1)									
		When this ins	struction is executed with	M40 = 0, co	ompatik	oility is en	sured.				
Status	Bits	Affected by	SATD, SXMD								
		Affects	ACOV0, CARRY								
Repeat	t	ing mode to a	on cannot be repeated wh ccess the memory opera his instruction can be rep	nd (Smem);	•	,					

Syntax	Description
*AR3 = *AR3 + #2E00h	The content addressed by AR3 is added to a signed 16-bit value (2E00h) and the result is stored back into the location addressed by AR3.

## Addition with Absolute Value

Syntax (	Characteristics
----------	-----------------

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = rnd(ACy	+  A	Cx  <mark>)</mark>		Yes	2	1	Х	
Opcod	e				010	01 01	0E DDS	S 000%	
Opera	nds	AC	x, ACy						
Descri	ption			n computes the absolute nulator ACy. This instru					
				lute value of accumu 6) by 00001h or 1FFF or.		•			
			☐ If FRCT = 1, the absolute value is multiplied by 2.						
			Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.						
			Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.						
		When an addition overflow is detected, the accumul according to SATD.						saturated	
			The result part of AC	of the absolute value y.	of the higher	part of	ACx is ir	n the lower	
		Compatibility with C54x devices (C54CM = 1)							
		Wł	nen this inst	ruction is executed wit	h M40 = 0, co	ompatik	oility is er	sured.	
Status Bits		Affected by FRCT, M40, RDM, SATD, SMUL							
		Aff	ects	ACOVy					
Repea	t	Th	is instructio	n can be repeated.					

See Also	See the following other related instructions:					
	Absol	ute Value				
	Additi	on				
	🗋 Additi	on or Subtraction Conditionally				
	🗋 Additi	on or Subtraction Conditionally with Shift				
	🗋 Additi	on, Subtraction, or Move Accumulator Content Conditionally				
Example						

Syntax	Description
AC0 = AC0 +  AC1	The absolute value of AC1 is added to the content of AC0 and the result is stored in AC0.

# Addition with Parallel Store Accumulator Content to Memory

## **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline				
[1]	ACy = ACx + <b>(</b> Ymem = <b>HI(</b> AC	Xmem <b>&lt;&lt; #16),</b> Cy <b>&lt;&lt; T2)</b>		No	4	1	Х				
Opcod	e	1000	0111 XXXM	MMYY YMM	AM SS	DD 100	x xxxx				
Operar	nds	ACx, ACy, T2, Xmem, Y	ACx, ACy, T2, Xmem, Ymem								
Descri	ption	This instruction perform	is two operation	ns in paralle	I: addit	ion and s	tore.				
		The first operation perfo and the content of data									
		The operation is pe	☐ The operation is performed on 40 bits in the D-unit ALU.								
		Input operands are sign extended to 40 bits according to SXMD.									
		The shift operation is equivalent to the signed shift instruction.									
		Overflow detection C54CM = 1, an inter- locally set to 1 and after the shifting op	ermediary shift no overflow de	operation	is perf	ormed as	if M40 is				
		When an overflow i SATD.	s detected, the	accumulato	r is sat	urated ac	cording to				
		The second operation s stores ACy(31–16) to d is not within –32 to +31 performed with this value	ata memory op , the shift is sa	erand Yme	m. If th	e 16-bit v	alue in T2				
		The input operand	s shifted in the	D-unit shift	er acco	ording to S	SXMD.				
		After the shift, the h the memory locatio		accumulato	r, ACy(	31–16), is	s stored to				
		Compatibility with C5									
		When this instruction is this instruction is execu determine the shift quar to +31. When the 16-b operation transforms th	ited with C54C itity. The 6 LSBs it value in T2	M = 1, the s of T2 define is between	6 LSB e a shif –32 to	s of T2 a t quantity 9 –17, a r	re used to within–32				

Status Bits	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	n can be repeated.
See Also	See the followi	ng other related instructions:
	Addition	
	Store Accu	imulator Content to Memory
	Subtraction	n with Parallel Store Accumulator Content to Memory

Syntax	Description
AC0 = AC1 + (*AR3 << #16), *AR4 = HI(AC0 << T2)	Both instructions are performed in parallel. The content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. The content of AC0 is shifted by the content of T2, and AC0(31–16) is stored at the address of AR4.

# Addition or Subtraction Conditionally

# Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = <b>adsc(</b> Sm	nem,	ACx, <b>TC1)</b>			No	3	1	Х	
[2]	ACy = <b>adsc(</b> Sm	nem,	ACx, <b>TC2)</b>			No	3	1	х	
Opcod	e	т	C1		1101	1110 AA	AA A	AAI SSE	D 0000	
		Т	C2		1101	1110 AA	AA A	AAI SSE	D 0001	
Operar	nds	AC	x, ACy, Sme	em, TCx				·		
Descri	ption	This instruction evaluates the selected TCx status bit and based on the result of the test, either an addition or a subtraction is performed. Evaluation of the condition on the TCx status bit is performed during the Execute phase of the instruction.								
			TC1 or TC2	O	peration					
			0	ACy = ACx	•					
			1	ACy = ACx	+ (Smem	<< #16)				
		<ul> <li>TCx = 0, then ACy = ACx – (Smem &lt;&lt; #16): This instruction subtracts the content of a memory (Smem) location s left by 16 bits from accumulator ACx and stores the result in accumu ACy.</li> </ul>								
		■ The operation is performed on 40 bits in the D-unit ALU.								
					according to SXMD.					
		<ul> <li>The shift operation is equivalent to the signed shift instruction</li> <li>Overflow detection and CARRY status bit depends on M40.</li> </ul>								
				an overflow is c						
		<b>TCx = 1</b> , then ACy = ACx + (Smem << #16):								
				ulator ACx I6 bits and						
			The op	eration is perf	ormed o	n 40 bits in t	he D-ι	unit ALU.		
		Input operands are sign extended to 40 bits according to SXMD.								
000/10							0 ( 5			

		· · · · ·
	Overfl	ow detection and CARRY status bit depends on M40.
	When to SAT	an overflow is detected, the accumulator is saturated according D.
	Compatibility	with C54x devices (C54CM = 1)
	C54CM = 1, a	Function is executed with $M40 = 0$ , compatibility is ensured. When in intermediary shift operation is performed as if M40 is locally o overflow detection, report, and saturation is done after the ion.
Status Bits	Affected by	C54CM, M40, SATD, SXMD, TCx
	Affects	ACOVy, CARRY
Repeat	This instruction	n can be repeated.
See Also	See the follow	ing other related instructions:
	Addition o	r Subtraction Conditionally with Shift
	Addition, S	Subtraction, or Move Accumulator Content Conditionally

■ The shift operation is equivalent to the signed shift instruction.

## Example 1

Syntax	Description
AC0 = adsc(*AR3, AC1, TC1)	If $TC1 = 1$ , the content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. If TC1 = 0, the content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

Syntax			Descrip	Description					
AC1 = ad	sc(*AR1, AC0	, TC2)	added to	TC2 = 1, the content addressed by AR1 shifted left by 16 bits is added to the content of AC0 and the result is stored in AC1. The result generated an overflow and a carry.					
Before			After						
AC0	00 EC00	0000	AC0	00 EC00 0000					
AC1	00 0000	0000	AC1	01 1F00 0000					
AR1		0200	AR1	0200					
200		3300	200	3300					
TC2		1	TC2	1					
SXMD		0	SXMD	0					
М40		0	M40	0					
ACOV1		0	ACOV1	1					
CARRY		0	CARRY	1					

#### Addition or Subtraction Conditionally with Shift

#### Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ads2	<b>c(</b> Smem, ACx, Tx <b>, TC1, T</b>		No	3	1	Х	
Opcod	e		AA AA	AAI SSI	DD ss10			
Opera	nds	ACx, ACy, Tx, Sme	em, TC1, TC	2				
Description		This instruction eva either an addition o TC2 status bit and or the content of T status bits is perfor	r a subtraction based on the Tx is perform	on is p e resul ned. E	erformed; th t of the test, valuation of	is instr either the co	uction eva a shift left ondition o	aluates the t by 16 bits on the TCx
		TC1	TC2		Opera	tion		
		0	0	A	ACy = ACx – (Smem << Tx		< Tx)	
		0	1	AC	Cy = ACx - (S)	mem <-	< #16)	
		1	0	A	ACy = ACx + (Smem << Tx)			

**TC1 = 0 and TC2 = 0**, then ACy = ACx - (Smem << Tx):

1

1

This instruction subtracts the content of a memory (Smem) location shifted left by the content of Tx from an accumulator ACx and stores the result in accumulator ACy.

ACy = ACx + (Smem << #16)

**TC1 = 0 and TC2 = 1**, then ACy = ACx – (Smem << #16):

This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from an accumulator ACx and stores the result in accumulator ACy.

- The operation is performed on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is equivalent to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
- When an overflow is detected, the accumulator is saturated according to SATD.

	<b>TC1 = 1 and TC2 = 0</b> , then ACy = ACx + (Smem << Tx):
	This instruction performs an addition operation between an accumulator ACx and the content of a memory (Smem) location shifted left by the content of Tx and stores the result in accumulator ACy.
	<b>TC1 = 1 and TC2 = 1</b> , then ACy = ACx + (Smem << #16):
	This instruction performs an addition operation between an accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.
	The operation is performed on 40 bits in the D-unit shifter.
	Input operands are sign extended to 40 bits according to SXMD.
	The shift operation is equivalent to the signed shift instruction.
	Overflow detection and CARRY status bit depends on M40.
	When an overflow is detected, the accumulator is saturated according to SATD.
	Compatibility with C54x devices (C54CM = 1)
	When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:
	An intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.
	□ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the value is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.
Status Bits	Affected by C54CM, M40, SATD, SXMD, TC1, TC2
	Affects ACOVy, CARRY
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Addition or Subtraction Conditionally
	Addition, Subtraction, or Move Accumulator Content Conditionally

Syntax				Description					
AC2 = ads	s2c(*AR2, A	C0, T1, TC	1, TC2)	TC1 = 1 and $TC2 = 0$ , the content addressed by AR2 shifted left by the content of T1 is added to the content of AC0 and the result is stored in AC2. The result generated an overflow.					
Before			After						
AC0	00 EC0	0 0000	AC0	00 EC00 0000					
AC2	00 000	0 0000	AC2	00 EC00 CC00					
AR2		0201	AR2	0201					
201		3300	201	3300					
Т1		0002	т1	0002					
TC1		1	TC1	1					
TC2		0	TC2	0					
M40		0	M40	0					
ACOV2		0	ACOV2	1					
CARRY		0	CARRY	0					

## Addition, Subtraction, or Move Accumulator Content Conditionally

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline				
[1]	ACy = adsc	(Smem, ACx, TC1, TC2)	)	No	3	1	Х				
Opcod	le			1101 1110 AA	AA AA	AI SSI	D 0010				
Opera	nds	ACx, ACy, Smem	ACx, ACy, Smem, TC1, TC2								
Descri	ption	test, an addition,	a move, or a	TCx status bits and a subtraction is pe s is performed durir	erformed	l. Evalua	tion of th				
		TC1	TC2	Opera	ation						
		0	0	ACy = ACx - (S)		#16)					
		0	1	ACy =		-#4 C)					
		1	0 1	ACy = ACx + (S ACy =		#16)					
				/(C) =	, lox						
		_	<ul> <li>TC2 = 1, then ACy = ACx:</li> <li>This instruction moves the content of ACx to ACy.</li> <li>The 40-bit move operation is performed in the D-unit</li> </ul>								
			-	operation, an overf							
		■ the d	estination acc	cumulator overflow	status I	bit (ACO	√y) is set.				
		■ the d	estination reg	jister (ACy) is satu	rated ac	cording t	o SATD.				
		TC1 = 0 and	TC2 = 0, the	n ACy = ACx – (Sm	nem <<	#16):					
				ne content of a mem Ilator ACx and stor							
		The oper	ation is perfo	rmed on 40 bits in	the D-ur	nit ALU.					
		Input ope	erands are sig	n extended to 40 k	oits acco	ording to	SXMD.				
		The shift	operation is e	equivalent to the sig	gned sh	ift instruc	tion.				
		Overflow	detection and	d CARRY status bi	t depen	ds on M4	0.				
		■ When an	overflow is de	etected, the accumu	ulatoris	saturated	laccordin				

to SATD.

	This instruction performs an addition operation between accumulator ACx and the content of a memory (Smem) location shifted left by 16 bits and stores the result in accumulator ACy.							
	The operation is performed on 40 bits in the D-unit ALU.							
	Input operands are sign extended to 40 bits according to SXMD.							
	The shift operation is equivalent to the signed shift instruction.							
	Overflow detection and CARRY status bit depends on M40.							
	When an overflow is detected, the accumulator is saturated according to SATD.							
	Compatibility with C54x devices (C54CM = 1)							
	When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.							
Status Bits	Affected by C54CM, M40, SATD, SXMD, TC1, TC2							
	Affects ACOVy, CARRY							
Repeat	This instruction can be repeated.							
See Also	See the following other related instructions:							
	Addition or Subtraction Conditionally							
	Addition or Subtraction Conditionally with Shift							

**TC1 = 1 and TC2 = 0**, then ACy = ACx + (Smem << #16):

Syntax	Description
AC0 = adsc(*AR3, AC1, TC1, TC2)	If TC2 = 1, the content of AC1 is stored in AC0. If TC2 = 0 and TC1 = 1, the content addressed by AR3 shifted left by 16 bits is added to the content of AC1 and the result is stored in AC0. If TC2 = 0 and TC1 = 0, the content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst & src	Yes	2	1	Х
[2]	dst = src & k8	Yes	3	1	х
[3]	dst = src & k16	No	4	1	х
[4]	dst = src & Smem	No	3	1	х
[5]	ACy = ACy & <b>(</b> ACx <b>&lt;&lt;&lt; #</b> SHIFTW <b>)</b>	Yes	3	1	х
[6]	ACy = ACx & (k16 <<< #16)	No	4	1	х
[7]	ACy = ACx & (k16 <<< #SHFT)	No	4	1	х
[8]	Smem = Smem & k16	No	4	1	Х

Description These instructions perform a bitwise AND operation: In the D-unit, if the destination operand is an accumulator. In the A-unit ALU, if the destination operand is an auxiliary or temporary register. In the A-unit ALU, if the destination operand is the memory. Affected by **Status Bits** C54CM Affects none See Also See the following other related instructions: D Bitwise AND Memory with Immediate Value and Compare to Zero □ Bitwise OR Bitwise Exclusive OR (XOR)

## Syntax Characteristics

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst & src						Yes	2	1	X
Opcod	e						00	10 10	OE FSS	S FDDD
Operar	nds	dst	, src							
Descri	ption	This instruction performs a bitwise AND operation between two registers.								
			Whe	en the o	destinatior	n (dst) oper	and is an accu	umulato	or:	
				The op	peration is	performed	on 40 bits in t	he D-u	nit ALU.	
		Input operands are zero extended to 40 bits.								
			•		tion, the 10		register is the s ne auxiliary or t		· · ·	
			Whe	en the o	destinatior	n (dst) oper	and is an auxi	liary or	tempora	ry register:
				The op	peration is	performed	on 16 bits in t	he A-u	nit ALU.	
							urce (src) ope r are used to p			
Status	Bits	Affe	ectec	l by	none					
		Affe	ects		none					
Repeat	t	Thi	s ins	tructior	n can be re	epeated.				

AC1 = AC1 & AC0 The content	of AC0 is ANDed with the content of AC1 and the result is stored in AC1.

Before		After	
AC0	7E 2355 4FC0	AC0	7E 2355 4FC0
AC1	OF E340 5678	AC1	OE 2340 4640

## Syntax Characteristics

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline		
	dst = src & k8						Yes	3	1	X		
Opcode						0001	100E kkł	ck kk	kk FDI	DD FSSS		
Operand	ds	dst	, k8,	src								
Descript	tion	This instruction performs a bitwise AND operation between a source (src) register content and an 8-bit value, k8.										
			Wh	en the	destinatio	n (dst) operai	nd is an accu	imulato	or:			
				The o	peration is	performed o	n 40 bits in tł	ne D-u	nit ALU.			
		Input operands are zero extended to 40 bits.										
		If an auxiliary or temporary register is the instruction, the 16 LSBs of the auxiliary or extended.					•					
			Wh	en the	destinatio	n (dst) operai	nd is an auxil	iary or	tempora	ry register:		
				The o	peration is	performed o	n 16 bits in tł	ne A-u	nit ALU.			
			•			or is the sour	· · ·			-		
Status B	Bits	Aff	ecte	d by	none							
		Aff	ects		none							
Repeat		Thi	is ins	structio	n can be r	epeated.						
Evenue	_											

Syntax	Description
AC0 = AC1 & #FFh	The content of AC1 is ANDed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline
	dst = src & k16						No	4	1	Х
[3] 0	$usl = sic \alpha kib$						INU	4	I	^
Opcode					0111	1101   kkk	k kkkk kkl	ck kł	kk FDI	D FSSS
Operand	S	dst	, k1(	6, src						
Descripti	ion	This instruction performs a bitwise AND operation between a source (sre register content and a 16-bit unsigned constant, k16.							ource (src)	
			W	hen the	destina	tion (dst) ope	rand is an acc	umulat	tor:	
			The op	peration	is performed	on 40 bits in tl	he D-u	nit ALU.		
	Input operands are zero extended to 40 bits.									
			•		tion, the		egister is the s ne auxiliary or t		· / ·	
			Wh	en the	destinat	ion (dst) oper	and is an auxi	liary or	tempora	ry register:
				The op	peration	is performed	on 16 bits in tl	he A-u	nit ALU.	
							urce (src) oper are used to p			
Status B	its	Aff	ecte	d by	none					
		Aff	ects		none					
Repeat		Thi	s in	structior	n can be	e repeated.				
•										

Syntax	Description
AC0 = AC1 & #FFFFh	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

## Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
[4] dst = src & Sme	am	No	3	1	X		
	5111	NU	3	I	^		
Opcode	1101	1001 AAA	A AA	AI   FDE	D FSSS		
Operands	dst, Smem, src						
Description	This instruction performs a bitwise AND operation between a source (src) register content and a memory (Smem) location.						
	When the destination (dst) operar	nd is an accu	mulato	or:			
	The operation is performed or	n 40 bits in th	ne D-u	nit ALU.			
	Input operands are zero exter	nded to 40 bi	ts.				
	If an auxiliary or temporary registruction, the 16 LSBs of the extended.			· / ·			
	When the destination (dst) operar	nd is an auxil	iary or	tempora	ry register:		
	The operation is performed or	n 16 bits in th	ne A-u	nit ALU.			
	<ul> <li>If an accumulator is the source</li> <li>16 LSBs of the accumulator a</li> </ul>	. , .					
Status Bits	Affected by none						
	Affects none						
Repeat	This instruction can be repeated.						
Evenue							

Syntax	Description
AC0 = AC1 & *AR3	The content of AC1 is ANDed with the content addressed by AR3 and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline				
[5]	ACy = ACy &	(ACx <<< #SHIFTW)	Yes	3	1	Х				
Opcod	e	0.0	01 000E DD:	SS 00	00 xxs	SH IFTW				
Operar	nds	ACx, ACy, SHIFTW								
Descri	ption	This instruction performs a bitwise AND operation between an accumulator (ACy) content and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.								
		The shift and AND operations shifter.	e cycle in	the D-unit						
		Input operands are zero exten								
		The input operand (ACx) is shifted by a 6-bit immediate value shifter.								
		The CARRY status bit is not affected by the logical shift operative status by the logical shift								
		Compatibility with C54x devices	(C54CM = 1)							
		When C54CM = 1, the intermediary set to 1. The 8 upper bits of the 40	•			•				
Status	Bits	Affected by C54CM								
		Affects none								
Repeat	:	This instruction can be repeated.	This instruction can be repeated.							

Syntax	Description
AC0 = AC0 & (AC1 <<< #30)	The content of AC0 is ANDed with the content of AC1 logically shifted left by 30 bits and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[6]	ACy = ACx &	(k16 <b>&lt;&lt;&lt; #16)</b>			No	4	1	Х		
Opcode	e		0111	1010 kkkk	kkkk kkk	k kk	kk SSI	D 010x		
Operar	nds	ACx, ACy, k16								
Descrij	otion	This instruction performs a bitwise AND operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by 16 bits.								
		The operation	tion is p	erformed on 40	bits in the D	-unit A	LU.			
		Input operands are zero extended to 40 bits.								
		The input of	operand	(k16) is shifted	I 16 bits to th	e MSE	s.			
Status	Bits	Affected by	none							
		Affects	none							
Repeat	:	This instructior	This instruction can be repeated.							

Syntax	Description
AC0 = AC1 & (#FFFFh <<< #16)	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) logically shifted left by 16 bits and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[7]	ACy = ACx &	(k16 <<< #SHFT)			No	4	1	Х		
Opcod	e		0111	0010 kkkk	kkkk kkk	ck kk	kk SSI	DD SHFT		
Operar	nds	ACx, ACy, k16	6, SHFT							
Descri	ption	This instruction (ACx) content value, SHFT.	•		•					
		The shift and AND operations are performed in one cycle in the D-unit shifter.								
		Input operands are zero extended to 40 bits.								
		The input of shifter.	operand (ł	(16) is shifted	by a 4-bit imr	nediat	e value in	the D-unit		
		The CARF	RY status	bit is not affec	ted by the lo	gical s	hift opera	ation.		
Status	Bits	Affected by	Affected by none							
		Affects	none							
Repeat	t	This instruction	n can be r	epeated.						
Fxamn	le									

Syntax	Description
AC0 = AC1 & (#FFFFh <<< #15)	The content of AC1 is ANDed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.

## **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smen	n & k16			No	4	1	Х
Opcod	e		1111	0100   АААА	AAAI kkk	ck kk	kk kkk	k kkkk
Operar	nds	k16, Smem						
Descrij	ption		•	ns a bitwise ANI nsigned consta	•	etweer	n a memo	ory (Smem)
		The operation	ation is p	erformed on 16	bits in the A	-unit A	LU.	
		The resul	lt is stored	d in memory.				
Status	Bits	Affected by	none					
		Affects	none					
Repeat	:	ing mode to a	ccess the	be repeated wh memory opera tion can be rep	nd (Smem); v	•	•	

Syntax		Description	
*AR1 = *AR1	1 & #0FC0		ressed by AR1 is ANDed with the unsigned 16-bit value (FC0h) stored in the location addressed by AR1.
Before		After	
*AR1	5678	*AR1	0640

Bitwise AND Memory with Immediate Value and Compare to Zero

#### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<b>TC1</b> = Smem 8	k16			No	4	1	Х
[2]	<b>TC2</b> = Smem 8	k16			No	4	1	Х
Opcod	e	TC1	1111	0010   АААА	AAAI kkk	ck kk	kk kkk	k kkkk
		TC2	1111	0011 AAAA	AAAI kkk	k kk	kk kkk	k kkkk
Operar	nds	k16, Smem, T	Сх					
Descri	ption		6, is ANE	ns a bit field ma Ded with the me	•			
		if( ((Sr	nem) AN	D k16 ) == 0	)			
		TCx :	= 0					
		else						
		TCx :	= 1					
Status	Bits	Affected by	none					
		Affects	TCx					
Repeat	t	ing mode to a	ccess the	be repeated wh memory opera tion can be rep	nd (Smem);	•	,	
See Al	so	See the follow	ving othe	r related instruc	tions:			
		Bitwise Al	ND					

Syntax	Description
	The unsigned 16-bit value (0060h) is ANDed with the content addressed by AR0. The result is 1, TC1 is set to 1.

Before		After	
*AR0	0040	*AR0	0040
TC1	0	TC1	1

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst   src	Yes	2	1	Х
[2]	dst = src   k8	Yes	3	1	х
[3]	dst = src   k16	No	4	1	х
[4]	dst = src   Smem	No	3	1	х
[5]	ACy = ACy   <b>(</b> ACx <b>&lt;&lt;&lt; #</b> SHIFTW <b>)</b>	Yes	3	1	х
[6]	ACy = ACx   <b>(</b> k16 <b>&lt;&lt;&lt; #16)</b>	No	4	1	х
[7]	ACy = ACx   <b>(</b> k16 <b>&lt;&lt;&lt; #</b> SHFT <b>)</b>	No	4	1	х
[8]	Smem = Smem   k16	No	4	1	х

 Description
 These instructions perform a bitwise OR operation:

 In the D-unit, if the destination operand is an accumulator.

 In the A-unit ALU, if the destination operand is an auxiliary or temporary register.

 In the A-unit ALU, if the destination operand is the memory.

 Status Bits
 Affected by
 C54CM

 Affects
 none

 See Also
 See the following other related instructions:

 Bitwise AND
 Bitwise Exclusive OR (XOR)

#### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	dst = dst   src				Yes	2	1	Х	
Opcode	9				001	LO 10	)1E FSS	SS FDDD	
Operan	ds	dst	, src						
Descrip	otion	Thi	This instruction performs a bitwise OR operation between two registers.						
			When the	destination (dst) oper	and is an accu	imulato	or:		
			The o	peration is performed	on 40 bits in tl	ne D-u	nit ALU.		
			Input	operands are zero ext	ended to 40 b	its.			
				uxiliary or temporary r ction, the 16 LSBs of th ded.	-		. , .		
			When the	destination (dst) operation	and is an auxil	iary or	tempora	ry register:	
			The o	peration is performed	on 16 bits in tl	ne A-u	nit ALU.		
				accumulator is the sou Bs of the accumulator					
Status	Bits	Affe	ected by	none					
		Affe	ects	none					
Repeat		Thi	s instructio	n can be repeated.					
Exampl	le								
Syntax		Desc	ription						

The content of AC0 is ORed with the content of AC1 and the result is stored in AC0.

AC0 = AC0 | AC1

## Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline					
[2] dst = src   k8		Yes	3	1	X					
Opcode	0001 1	LO1E   kkk	k kk	kk FDE	D FSSS					
Operands	dst, k8, src									
Description	This instruction performs a bitwise OR operation between a source (src) register content and an 8-bit value, k8.									
	When the destination (dst) operand is an accumulator:									
	The operation is performed on 4	40 bits in th	e D-u	nit ALU.						
	Input operands are zero extended to 40 bits.									
	If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.									
	When the destination (dst) operand in the destinatin (dst) operand in the destination (dst) oper	is an auxili	ary or	tempora	ry register:					
	The operation is performed on 1	16 bits in th	e A-ur	nit ALU.						
	<ul> <li>If an accumulator is the source</li> <li>16 LSBs of the accumulator are</li> </ul>									
Status Bits	Affected by none									
	Affects none									
Repeat	This instruction can be repeated.									
Example										

Syntax	Description
AC0 = AC1   #FFh	The content of AC1 is ORed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

## Syntax Characteristics

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline	
[3]	dst = src   k16						No	4	1	X	
Opcod	e				0111	1110   kkk	k kkkk kk	kk kł	kk   FDI	DD FSSS	
Operar	nds	dst	, k16	s, src							
Descri	ption	This instruction performs a bitwise OR operation between a source (src) register content and a 16-bit unsigned constantk16.									
			When the destination (dst) operand is an accumulator:								
		The operation is performed on 40 bits in the D-unit ALU.									
		Input operands are zero extended to 40 bits.									
		If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.									
			Wh	en the d	lestinat	ion (dst) ope	rand is an auxi	liary or	r tempora	ry register:	
				The op	eration	is performed	l on 16 bits in t	he A-u	nit ALU.		
			•				ource (src) ope or are used to p				
Status	Bits	Aff	ecte	d by	none						
		Aff	ects		none						
Repeat	t	This instruction can be repeated.									

Syntax	Description
AC0 = AC1   #FFFFh	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

## Syntax Characteristics

			Parallel			
No. Syntax			nable Bit	Size	Cycles	Pipeline
[4] dst = src   Sme	n		No	3	1	Х
Opcode		1101 10	010 AAA	A AA	AI FDD	D FSSS
Operands						
Description	This instruction performs a bitwise OR operation between a source (src) register content and a memory (Smem) location.					
	When the destination (	dst) operand is	s an accu	mulato	or:	
	The operation is performed as a second se	erformed on 40	0 bits in th	e D-u	nit ALU.	
	Input operands are zero extended to 40 bits.					
	If an auxiliary or ten instruction, the 16 L extended.				· / ·	
	When the destination (	dst) operand is	s an auxili	ary or	tempora	ry register:
	The operation is performed as a second se	erformed on 16	6 bits in th	e A-ur	nit ALU.	
	If an accumulator in 16 LSBs of the accumulator		· · ·			-
Status Bits	Affected by none					
	Affects none					
Repeat	This instruction can be repeated.					
Example						

Syntax	Description
AC0 = AC1   *AR3	The content of AC1 is ORed with the content addressed by AR3 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ACy = ACy   (	ACx <<< #SHII	-TW <b>)</b>		Yes	3	1	Х
Opcod	е			0001	000E DDS	GS 00	001   xxS	SH IFTW
Operands ACx, ACy, SHIFTW								
Descri	ption	This instruction performs a bitwise OR operation between an accur (ACy) content and and an accumulator (ACx) content shifted by the 6-bit SHIFTW.						
		The sl shifter.		operations are	e performed	in one	cycle in	the D-unit
		🗋 Input d	Input operands are zero extended to 40 bits.					
		The in shifter.	• •	ACx) is shifted	by a 6-bit im	mediat	e value ir	n the D-unit
		The C.	ARRY status	bit is not affec	ted by the lo	gical s	hift opera	ation.
		Compatibility with C54x devices (C54CM = 1)						
		When C54CM = 1, the intermediary logical shift is performed as if M40 set to 1. The 8 upper bits of the 40-bit intermediary result are not cle				•		
Status Bits		Affected by	/ C54CN	1				
		Affects	none					
Repeat	t	This instruction can be repeated.						

Syntax	Description
AC1 = AC1   (AC0 <<< #4)	The content of AC1 is ORed with the content of AC0 logically shifted left by 4 bits and the result is stored in AC1.

Before		After	
AC0	7E 2355 4FC0	AC0	7E 2355 4FC0
AC1	OF E340 5678	AC1	OF F754 FE78

#### Bitwise OR

#### Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx   (	k16 <b>&lt;&lt;&lt; #16)</b>			No	4	1	X
Opcode	e		0111	1010 kkkk	kkkk kkk	k k:	kk SSI	D 011x
Operar	nds	ACx, ACy, k16	6					
DescriptionThis instruction performs a bitwise (ACx) content and a 16-bit unsigned					•			
The operation			ition is p	ion is performed on 40 bits in the D-unit ALU.				
Input operands are zero extended to 40 bits.								
		The input	operand	(k16) is shifted	16 bits to th	e MSE	s.	
Status	Bits	Affected by	none					
		Affects	none					
<b>Repeat</b> This instruction can be repeated.								

Syntax	Description
AC0 = AC1   (#FFFFh <<< #16)	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) logically shifted left by 16 bits and the result is stored in AC0.

#### Bitwise OR

#### Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ACy = ACx   <b>(</b> k	:16 <<< #SHFT)			No	4	1	Х
Opcod	e		0111	0011 kkkk	kkkk kkk	ck kk	kk SSI	D SHFT
Operar	nds	ACx, ACy, k1	6, SHFT					
Description		This instruction performs a bitwise OR operation between an accumulator (ACx) content and a 16-bit unsigned constant, k16, shifted left by the 4-bit value, SHFT.						
		The shift shifter.	and OR	operations are	performed i	in one	cycle in	the D-unit
		🗋 Input ope	erands are	e zero extendeo	to 40 bits.			
		The input shifter.	t operand	(k16) is shifted	by a 4-bit imr	mediat	e value in	the D-unit
		The CAR	RY status	s bit is not affec	ted by the lo	gical s	hift opera	ation
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instruction	on can be	repeated.				
Evamn								

Syntax	Description
AC0 = AC1   (#FFFFh <<< #15)	The content of AC1 is ORed with the unsigned 16-bit value (FFFFh) logically shifted left by 15 bits and the result is stored in AC0.

#### Bitwise OR

#### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smer	n k16			No	4	1	Х
Opcod	e		1111	0101   АААА	AAAI kkk	ck kk	kk kkk	k kkkk
Operar	nds	k16, Smem						
Description		This instruction performs a bitwise OR operation between a memory (Smem) location and a 16-bit unsigned constant, k16.						
		The operation	ation is p	erformed on 16	bits in the A	-unit A	LU.	
		The result	t is store	d in memory.				
Status	Bits	Affected by	none					
		Affects	none					
Repeat	epeat This instruction cannot be repeate ing mode to access the memory o ing modes, this instruction can be				nd (Smem); v	•	,	

Syntax		Description	
*AR1 = *AR1   #0FC0h		The content addressed by AR1 is ORed with the unsigned 16-bit valu and the result is stored in the location addressed by AR1.	
Before		After	
*AR1	5678	*AR1	5FF8

# Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst ^ src	Yes	2	1	Х
[2]	dst = src ^ k8	Yes	3	1	Х
[3]	dst = src ^ k16	No	4	1	х
[4]	dst = src ^ Smem	No	3	1	х
[5]	ACy = ACy ^ (ACx <<< #SHIFTW)	Yes	3	1	х
[6]	ACy = ACx ^ (k16 <<< <b>#16</b> )	No	4	1	х
[7]	ACy = ACx ^ (k16 <<< #SHFT)	No	4	1	х
[8]	Smem = Smem ^ k16	No	4	1	х

Description	These instructions perform a bitwise exclusive-OR (XOR) operation:					
	In the D-unit, if the destination operand is an accumulator.					
	In the A-unit ALU, if the destination operand is an auxiliary or temporary register.					
	In the A-unit ALU, if the destination operand is the memory.					
Status Bits	Affected by C54CM					
	Affects none					
See Also	See the following other related instructions:					
	Bitwise AND					
	Bitwise OR					

# Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst ^ src					Yes	2	1	Х
Opcod	e					001	LO 11	OE FSS	S FDDD
Operar	nds	dst, src							
Descri	ption	This instruction performs a bitwise exclusive-OR (XOR) operation between two registers.							
			When the	destination (d	lst) operar	nd is an accu	imulato	or:	
			The o	peration is per	rformed or	n 40 bits in tł	ne D-u	nit ALU.	
			Input	operands are	zero exter	nded to 40 bi	its.		
				auxiliary or tem ction, the 16 LS ded.		-		· / ·	
			When the	destination (d	lst) operar	nd is an auxil	iary or	tempora	ry register:
			The o	peration is per	rformed o	n 16 bits in th	ne A-u	nit ALU.	
				accumulator is Bs of the accu					
Status	Bits	Aff	ected by	none					
		Aff	ects	none					
Repeat	t	Thi	is instructio	n can be repe	ated.				

Description
The content of AC0 is XORed with the content of AC1 and the result is stored in AC1.

Before		After	
AC0	7E 2355 4FC0	AC0	7E 2355 4FC0
AC1	OF E340 5678	AC1	71 C015 19B8

#### Syntax Characteristics

No. Syntax	Parallel Enable Bit Size Cycles Pipeline						
[2] $dst = src^k k\delta$							
Opcode	0001 110E kkkk kkkk FDDD FSSS						
Operands	dst, k8, src						
Description	This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and an 8-bit value, k8.						
	When the destination (dst) operand is an accumulator:						
	The operation is performed on 40 bits in the D-unit ALU.						
	Input operands are zero extended to 40 bits.						
	If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.						
	When the destination (dst) operand is an auxiliary or temporary register:						
	The operation is performed on 16 bits in the A-unit ALU.						
	If an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.						
Status Bits	Affected by none						
	Affects none						
Repeat	This instruction can be repeated.						

Syntax	Description
AC0 = AC1 ^ #FFh	The content of AC1 is XORed with the unsigned 8-bit value (FFh) and the result is stored in AC0.

#### Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[3] dst = src ^ k16		No	4	1	X
Opcode	0111 1111 k	kkk kkkk kkł	ck kk	kk FDE	D FSSS
Operands	dst, k16, src				
Description	This instruction performs a bitwis source (src) register content and	•	,	•	
	When the destination (dst) or	perand is an accu	imulato	or:	
	The operation is perform	ed on 40 bits in tl	ne D-ui	nit ALU.	
	Input operands are zero extended to 40 bits.				
	If an auxiliary or temporal instruction, the 16 LSBs of extended.			· / ·	
	When the destination (dst) op	perand is an auxil	iary or	tempora	ry register:
	The operation is performed on 16 bits in the A-unit ALU.				
	<ul> <li>If an accumulator is the 16 LSBs of the accumula</li> </ul>	. , .			
Status Bits	Affected by none				
	Affects none				
Repeat	This instruction can be repeated.				
Evennle					

Syntax	Description
AC0 = AC1 ^ #FFFFh	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh) and the result is stored in AC0.

# Syntax Characteristics

-		Parallel				
No. Syntax		Enable Bit	Size	Cycles	Pipeline	
[4] dst = src ^ Sme	m	No	3	1	Х	
Opcode	1101	1011 AAA	A AA	AI   FDI	DD FSSS	
Operands	dst, Smem, src					
Description	This instruction performs a bitwise exc source (src) register content and a me		,	•	between a	
	When the destination (dst) operand is an accumulator:					
	The operation is performed on 40 bits in the D-unit ALU.					
	Input operands are zero extended to 40 bits.					
	If an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or temporary register are zero extended.					
	When the destination (dst) operand	nd is an auxil	iary or	tempora	ry register:	
	The operation is performed on 16 bits in the A-unit ALU.					
	If an accumulator is the source 16 LSBs of the accumulator a	· / ·				
Status Bits	Affected by none					
	Affects none					
Repeat	This instruction can be repeated.					

Syntax	Description
AC0 = AC1 ^ *AR3	The content of AC1 is XORed with the content addressed by AR3 and the result is stored in AC0.

#### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[5]	-	(ACx <<< #SHIFTV	V)	Yes	3	1	x	
Opcod	e		0001	000E DDS	SS 00	10 xxS	H IFTW	
Opera	nds	ACx, ACy, SH	IFTW					
Descri	ption	This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACy) content and an accumulator (ACx) content shifted by the 6-bit value, SHIFTW.						
		The shift and XOR operations are performed in one cycle in the D-unit shifter.						
		Input operands are zero extended to 40 bits.						
		The input operand (ACx) is shifted by a 6-bit immediate value in the D-unit shifter.						
		The CARRY status bit is not affected by the logical shift operation.						
		Compatibility with C54x devices (C54CM = 1)						
		When $C54CM = 1$ , the intermediary logical shift is performed as if M40 is locally set to 1. The 8 upper bits of the 40-bit intermediary result are not cleared.						
Status	Bits	Affected by	C54CM					
		Affects	none					
Repeat This instruction can be repeated.								

Syntax	Description
AC0 = AC0 ^ (AC1 <<< #30)	The content of AC0 is XORed with the content of AC1 logically shifted left by 30 bits and the result is stored in AC0.

#### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ACy = ACx ^	<b>(</b> k16 <b>&lt;&lt;&lt; #16)</b>		No	4	1	Х
Opcod	le		0111 1010 kkkk	kkkk kkł	ck kk	kk SSI	DD 100x
Opera	nds	ACx, ACy, k16					
Descri	ption	•	performs a bitwise exc Ex) content and a 16-b	•	<i>,</i> .		
		The operation	on is performed on 40	) bits in the D	-unit A	LU.	
		Input operar	nds are zero extendeo	d to 40 bits.			
		The input op	perand (k16) is shifted	d 16 bits to th	e MSE	Bs.	
Status	Bits	Affected by	none				
		Affects	none				
Repea	t	This instruction of	can be repeated.				
_	_						

Syntax	Description
AC0 = AC1 ^ (#FFFFh <<< #16)	The content of AC1 is XORed with the unsigned 16-bit value (FFFFh)
	logically shifted left by 16 bits and the result is stored in AC0.

#### Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[7]	-	k16 <<< #SHFT)	No	4	1	X
Opcod	e	0111 010	00 kkkk kkkk kkk	k k:	kk SSD	D SHFT
Operar	nds	ACx, ACy, k16, SHFT				
Descri	ption	This instruction performs a b accumulator (ACx) content the 4-bit value, SHFT.				
		The shift and XOR operations are performed in one cycle in the D-unit shifter.				
		Input operands are zero extended to 40 bits.				
		The input operand (k16 shifter.	) is shifted by a 4-bit imr	nediate	e value in	the D-unit
		The CARRY status bit i	s not affected by the lo	gical s	hift opera	ation.
Status	Bits	Affected by none				
		Affects none				
Repeat	t	This instruction can be repe	eated.			
Examp	le					

Syntax	Description
	The content of AC1 is XORed with the unsigned 16-bit value (FFFh) logically shifted left by 15 bits and the result is stored in AC0.

#### Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[8]	Smem = Smem	1 ^ k16			No	4	1	X
Opcod	e		1111	0110 AAAA	AAAI kkk	k k:	kk kkk	k kkkk
Operar	nds	k16, Smem						
Description			•	ns a bitwise exc ion and a 16-bit	•	,	•	between a
		The operation	tion is p	erformed on 16	bits in the A	-unit A	LU.	
		The result	is stored	d in memory.				
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	ing mode to ac	cess the	be repeated wh memory opera tion can be rep	nd (Smem); v	•		

Syntax	Description
*AR3 = *AR3 ^ #FFFFh	The content addressed by AR3 is XORed with the unsigned 16-bit value (FFFFh) and the result is stored in the location addressed by AR3.

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) goto I4	No	2	6/5	R
[2]	if (cond) goto L8	Yes	3	6/5	R
[3]	if (cond) goto L16	No	4	6/5	R
[4]	if (cond) goto P24	No	5	5/5	R

 $\frac{1}{x}$  y cycles: x cycles = condition true, y cycles = condition false

Description	These instructions evaluate a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into I4, Lx, or P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.			
	The instruction selection depends on the branch offset between the current PC value and the program branch address specified by the label.			
	These instructions cannot be repeated.			
Status Bits	Affected by ACOVx, CARRY, C54CM, M40, TCx			
	Affects ACOVx			
See Also	See the following other related instructions:			
	Branch Unconditionally			
	Branch on Auxiliary Register Not Zero			
	Call Conditionally			
	Compare and Branch			

#### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) goto	14		No	2	6/5	R
$\uparrow$ x/y cycles: x cycles = condition true, y cycles = condition false							
Opcod	e			01	10 0	111 <b> </b> 1CC	
Opera	nds	cond, l4					
<b>Description</b> This instruction evaluates a single condition defined by the oread phase of the pipeline. If the condition is true, a branch program address label assembled into I4. There is a 1-cycle condition setting. A single condition can be tested as determine field of the instruction. See Table 1–3 for a list of conditions.				oranch oco cycle later ermined b	curs to the ncy on the		
		Compatibility	/ with C54x devices (	C54CM = 1)			
		When C54CM was set to 1.	= 1, the comparison of	accumulators	s to 0 is	performe	d as if M40
Status	Bits	Affected by	ACOVx, CARRY, C	54CM, M40, 1	ГСх		
		Affects	ACOVx				
Repea	t	This instructio	n cannot be repeated.				
Examp							

#### Example

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

	if (AC0 != #0) goto branc	h		
			address:	004057
branch				00405A
:				
Before		After		
AC0	00 0000 3000	AC0	00 0000	3000
PC	004055	PC	00	405A

SPRU375G

#### **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[2]	if (cond) goto L	_8				Yes	3	6/5	R
[3]	if (cond) goto L	_16				No	4	6/5	R
†x/y cyo	† x/y cycles: x cycles = condition true, y cycles = condition false								
Opcod	e	L8			0000	010E xC	CC C	CCC LLL	L LLLL
		L16	0110	1101	xCCC	CCCC LL	LL L	LLL LLL	L LLLL
Operar	Operands cond, Lx								
Description		This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into Lx. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.							
		Compatibilit	y with C	54x devi	ices (C	54CM = 1)			
		When C54CM = 1, the comparison of was set to 1.			son of a	accumulators	s to 0 is	performe	d as if M40
Status	Bits	Affected by	ACOV	/x, CARF	RY, C54	4CM, M40, 1	ГСх		
		Affects	ACOV	/x					
Repeat	t	This instruction	on canno	t be repe	ated.				
Evamn									

Syntax		Description			
if (AC0 != #0) goto branch		The content of AC0 is not equal to 0, control is passed to the program add label defined by branch.			
branch :		00305A			
	if (AC0 != #0) goto b	pranch			
		address: 004057			

Before		After	
AC0	00 0000 3000	AC0	00 0000 3000
PC	004055	PC	00305A

#### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[4]	if (cond) goto	<b>P</b> 24		No	5	5/5	R
† x/y cy	cles: x cycles = co						
Opcod	e	0110 1000	XCCC CCCC PPPP	PPPP PP	PP P	PPP PPP	P PPPP
Opera	nds	cond, P24					
Description		read phase of program addre condition setti	This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a branch occurs to the program address label assembled into P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.				
		Compatibility	/ with C54x devices (C	C54CM = 1)			
		When C54CM was set to 1.	= 1, the comparison of a	accumulators	s to 0 is	performe	d as if M40
Status	Bits	Affected by	ACOVx, CARRY, C5	4CM, M40, 1	ГСх		
		Affects	ACOVx				
Repea	t	This instructio	n cannot be repeated.				
_							

#### Example

Syntax	Description
if (AC0 != #0) goto branch	The content of AC0 is not equal to 0, control is passed to the program address label defined by branch.

00F05A

branch :	.sect "code1" if (AC0 != #0) goto branc  .sect "code2" 	h	address: 004057 00F05A	
<b>Before</b> AC0	00 0000 3000	<b>After</b> ACO	00 0000 3000	

PC

004055

#### SPRU375G

PC

#### **Syntax Characteristics**

	_	Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	goto ACx	No	2	10	Х
[2]	goto L7	Yes	2	6†	AD
[3]	goto L16	Yes	3	6†	AD
[4]	goto P24	No	4	5	D

 $\ensuremath{^\dagger}$  This instruction executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

Description	This instruction branches to a 24-bit program address defined by the content of the 24 lowest bits of an accumulator (ACx), or to a program address defined by the program address label assembled into Lx or P24.					
	These instructions cannot be repeated.					
Status Bits	Affected by none					
	Affects none					
See Also	See the following other related instructions:					
	Branch Conditionally					
	Branch on Auxiliary Register Not Zero					
	Call Unconditionally					
	Compare and Branch					

#### Syntax Characteristics

No. Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1] goto ACx			No	2	10	Х
Opcode			10	01 00	001 xxx	x xxSS
Operands	ACx					
Description		on branches to a 24-bit est bits of an accumula		ress de	efined by t	the content
Status Bits	Affected by	none				
	Affects	none				
Repeat	This instruction	on cannot be repeated.				
Fremula						

Syntax		Description		
goto AC0		Program contro	ol is passed to	the program address defined by the content of AC0(23–0).
Before			After	
AC0	00	0000 403D	AC0	00 0000 403D
PC		001F0A	PC	00403D

# Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[2]	goto L7	Yes	2	6	AD
[3]	goto L16	Yes	3	6	AD

<sup>†</sup>Executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

Opcode	L7				0100	101E	OLLL	LLLL
	L16		0000	011E	LLLL	LLLL	LLLL	LLLL
Operands	Lx							
Description	This instruction label assemble		program	addres	s define	d by a p	rogram	address
Status Bits	Affected by	none						
	Affects	none						
Repeat	This instruction	n cannot be re	peated.					

Syntax	Description
goto branch	Program control is passed to the absolute address defined by branch.

	goto branch		
	AC0 = #1	address:	004044
branch:			006047
	AC0 = #0		

Before		After	
PC	004042	PC	006047
AC0	00 0000 0001	AC0	00 0000 0000

#### **Syntax Characteristics**

No. Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[4] goto P24				No	4	5	D
Opcode		0110	1010 PPPP	PPPP PP	PP PI	PPP PPP	PP PPPP
Operands	P24						
Description	This instructio label assembl		es to a program 24.	address def	ined by	y a progra	m address
Status Bits	Affected by	none					
	Affects	none					
Repeat	This instructio	n cannot	be repeated.				

Syntax	Description
goto branch	Program control is passed to the absolute address defined by branch.

	goto branch		
	AC0 = #1	address:	004044
branch:			006047
	AC0 = #0		

Before		After
PC	004042	PC 006047
AC0	00 0000 0001	AC0 00 0000 0000

# Branch on Auxiliary Register Not Zero

#### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (ARn_mod !=	= #0) 🤉	goto L16		No	4	6/5	AD
⁺x/y cyo	cles: x cycles = con	dition	true, y cycles = condition f	alse				
Opcod	e		1111	1100 AAAA	AAAI LL	LL L	lll   lll	L LLLL
Operar	nds	AR	n_mod, L16					
Descrij	ption	cor ado	s instruction perform ntent not equal to 0) dress is specified as truction to branch wir ue.	of the program	m counter ( d offset, L1	PC). 1 6, rela	The progra ative to PC	am branch C. Use this
		The	e possible addressing	g operands ca	n be groupe	ed into	three cate	gories:
			ARx not modified (A *AR1; No modificati *AR1(#15); Use 16- *AR1(T0); Use cont *AR1(short(#4)); Us	on or offset bit immediate ent of T0 as of	value (15) a ffset	as offse	et	
			ARx modified before *–AR1; Decrement *+AR1(#20); Add 16	by 1 before co	mparison		·	son
			ARx modified after t *AR1+; Increment b *(AR1 – T1); Subtra	y 1 after comp	arison			
		1)	The content of the s address generation		ary register	(ARn)	is premod	ified in the
		2)	The (premodified) co in the address phas		•	l to 0 a	nd sets the	e condition
		3)	If the condition is no instructions are exe			If the o	condition i	s true, the
		4)	The content of ARn	is postmodifie	ed in the add	lress g	eneration	unit.

	Compatibility with C54x devices (C54CM = 1)
	When C54CM = 1:
	The premodifier *ARn(T0) is not available; *ARn(AR0) is available.
	The postmodifiers *(ARn + T0) and *(ARn – T0) are not available; *(ARn + AR0) and *(ARn – AR0) are available.
	The legality of the modifier usage is checked by the assembler when using the .c54cm_on and .c54cm_off assembler directives.
Status Bits	Affected by C54CM
	Affects none
Repeat	This instruction cannot be repeated.
See Also	See the following other related instructions:
	Branch Conditionally
	Branch Unconditionally
	Compare and Branch

Syntax	Description
	The content of AR1 is compared to 0. The content is not 0, program control is passed to the program address label defined by branch.

	If (*AR1(#6) != #0) goto branch	address:	004004
		;	00400A
branch		;	00400C
:			

Before		After	
AR1	0005	AR1	0005
PC	004004	PC	00400C

Syntax	Description
if (*AR3– != #0) goto branch	The content of AR3 is compared to 0. The content is 0, program control is passed to the next instruction (the branch is not taken). AR3 is decremented by 1 after the comparison.

	If (*AR3– != #0) goto branch	address:	00400F
		;	004013
branch		;	004015
:			

Before		After	
AR3	0000	AR3	FFFF
PC	00400F	PC	004013

# Call Conditionally

# Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) call L16	No	4	6/5	R
[2]	if (cond) call P24	No	5	5/5	R

† x/y cycles: x cycles = condition true, y cycles = condition false

Description	read phase of the program a L16 or P24. T condition can b	ions evaluate a single condition defined by the cond field in the the pipeline. If the condition is true, a subroutine call occurs to ddress defined by the program address label assembled into There is a 1-cycle latency on the condition setting. A single be tested as determined by the cond field of the instruction. See a list of conditions.				
	of two internal The CPU can	Before beginning a called subroutine, the CPU automatically saves the value of two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the subroutine is done.				
	loop context bi from a subrout	In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.				
	In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.					
		selection depends on the branch offset between the current PC gram subroutine address specified by the label.				
	These instruct	ions cannot be repeated.				
Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx				
	Affects	ACOVx				

SPRU375G

See Also

See the following other related instructions:

- Branch Conditionally
- Call Unconditionally
- Return Conditionally
- Return Unconditionally

Call Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) call L16	No	4	6/5	R

t x/y cycles: x cycles = condition true, y cycles = condition false

Opcode	0110 1110 xCCC CCCC LLLL LLLL LLLL					
Operands	cond, L16					
Description	scriptionThis instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into L16. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.					
	When a subroutine call occurs in the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the <i>TMS320C55x DSP CPU Reference Guide</i> (SPRU371).					
	☐ The data stack pointer (SP) is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.					
	☐ The system stack pointer (SSP) is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.					
	The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.					
	System Stack (SSP) Data Stack (SP)					
$\begin{array}{rcl} \mbox{After} & \rightarrow & \mbox{SSP} = x - \\ \mbox{Save} & \end{array}$	1 (Loop bits):PC(23–16) After $\Rightarrow$ SP = y - 1 PC(15–0) PC(15–0)					
$\begin{array}{rll} \textbf{Before} & \to & \text{SSP} = x \\ \textbf{Save} & & \end{array}$	Previously saved dataBefore Save $SP = y$ Previously saved data					

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Call Conditionally (if call)

# Status BitsAffected byACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

Syntax	Description
if (AC1 >= #2000h) call (subroutine)	The content of AC1 is equal to or greater than 2000h, control is passed to the program address label, subroutine. The program counter (PC) is loaded with the subroutine program address.

Call Conditionally

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[2]	if (cond) call P24	No	5	5/5	R

† x/y cycles: x cycles = condition true, y cycles = condition false

Opcode Operands	0110 1001 XCCC CCCC PPPP PPPP PPPP PPPP PPPP PPP				
Description	This instruction evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a subroutine call occurs to the program address defined by the program address label assembled into P24. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.				
	When a subroutine call occurs in the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks. For fast-return mode operation, see the <i>TMS320C55x DSP CPU Reference Guide</i> (SPRU371).				
	□ The data stack pointer (SP) is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.				
	The system stack pointer (SSP) is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.				
	The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.				
	System Stack (SSP)     Data Stack (SP)				
$\begin{array}{rcl} \textbf{After} & \to & \text{SSP} = \textbf{x} - \\ \textbf{Save} & \end{array}$	1(Loop bits):PC(23-16)After Save $\Rightarrow$ SP = y - 1PC(15-0)				
$\begin{array}{rll} \textbf{Before} \\ \textbf{Save} \end{array} \rightarrow  \textbf{SSP} = \textbf{x} \end{array}$	$\begin{array}{c} \mbox{Previously saved data} \\ \mbox{Save} \end{array} \rightarrow \ \ \mbox{SP} = y \\ \mbox{Previously saved data} \end{array}$				

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

# Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

Syntax	Description
if (TC1) call FOO	If TC1 is set to 1, control is passed to the program address label (FOO) assembled into an absolute address defined by the 24-bit value. If TC1 is cleared to 0, the program counter is incremented by 6 and the next instruction is executed.

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	call ACx	No	2	10	X
[2]	call L16	Yes	3	6	AD
[3]	call P24	No	4	5	D

# **Description** This instruction passes control to a specified subroutine program address defined by the content of the 24 lowest bits of the accumulator, ACx, or a program address label assembled into L16 or P24.

Before beginning a called subroutine, the CPU automatically saves the value of two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the subroutine is done.

In the slow-return process (default), the return address (from the PC) and the loop context bits are stored to the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.

In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions.

These instructions cannot be repeated.

Status Bits	Affected by	none		
	Affects	none		
See Also	See the following other related instructions:			
	🗋 Branch U	nconditionally		

- Call Conditionally
- Return Conditionally
- Return Unconditionally

#### **Syntax Characteristics**

No.	Syntax					F	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	call ACx					-	No	2	10	X
Opcod	е						10	01 0	010 xxx	x xxSS
Opera	nds	AC	Cx							
Descri	ption		his instructio afined by the							
		lo	the slow-ret op context bi e <i>TMS320C5</i>	its are stor	ed to the	stacks	s. For fast-	return		
			phase of	the pipeli	ne. The	16 LS	Bs of the	returr	address	ne address s, from the e top of SP.
			phase of th	ne pipeline	. The loo	p conte		ncatena		he address he 8 MSBs
			The PC is flow exect					addres	s. The ac	tive contro
			System Sta	ck (SSP)				l	Data Stac	k (SP)
Afte Save	$\rightarrow$ SS	P = x - 1	(Loop bits):P	PC(23–16)	Af Sa	ter $ ightarrow$ ve $ ightarrow$	SP = y -	1	PC(15-	-0)
Before Save	$\rightarrow$ SS	SP = x	Previously sa	aved data	Befo Sa	ve $\rightarrow$	SP = y	Pre	eviously sa	ved data
Status	Bits		fected by fects	none none						
Repea	t	Th	nis instructio	n cannot b	e repeat	ed.				
Examp	ble									
Syntax	ĸ	Descriptio	on							

Program control is passed to the program address defined by the content of AC0(23–0).

call AC0

# Syntax Characteristics

No. Syntax		Parallel Enable B	t Size	Cycles	Pipeline
[2] call L16		Yes	3	6	AD
Opcode		0000 100E I	LLL L	LLL LLI	L LLL
Operands	L16				
Description	This instruction passes control to a specified subroutine program address defined by a program address label assembled into L16.				
	In the slow-return proces loop context bits are store the <i>TM</i> S320C55x DSP CF	ed to the stacks. For fas	t-return	mode ope	,
	phase of the pipelir	ter (SP) is decrementene. The 16 LSBs of the called subrouting (a) of the called subrouting (b) of the	ne returr	n address	, from the
	phase of the pipeline	inter (SSP) is decremen . The loop context bits c s are pushed to the top	oncaten		
	The PC is loaded with flow execution conte	n the subroutine program xt flags are cleared.	n addres	ss. The ac	tive contro
	System Stack (SSP)			Data Stacl	k (SP)
After $\rightarrow$ SSP = $\rightarrow$	x – 1 (Loop bits):PC(23–16)	$\begin{array}{rcl} \textbf{After} & \\ \textbf{Save} & \rightarrow & \text{SP} = y \end{array}$	- 1	PC(15-	-0)
$\begin{array}{rl} \text{Before} \\ \text{Save} \end{array} \rightarrow  \text{SSP} = \end{array}$	x Previously saved data	$\begin{array}{rcl} \textbf{Before} \\ \textbf{Save} \end{array} \rightarrow  \textbf{SP} = \textbf{y} \end{array}$	Pre	eviously sa	ved data
Status Bits	Affected by none				
	Affects none				

**Repeat** This instruction cannot be repeated.

Syntax	Description
call FOO	Program control is passed to the program address label (FOO) assembled into the signed
	16-bit offset value relative to the program counter register.

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size Cycles Pipeline		
[3]	call P24		No	4 5 D		
Opcode	e	0110 1100	PPPP PPPP PPP	P PPPP PPPP PPPP		
Operan	nds	P24				
Descrip	otion	This instruction passes control to a specified subroutine program address defined by a program address label assembled into P24.				
		In the slow-return process (defa loop context bits are stored to the the TMS320C55x DSP CPU Re	he stacks. For fast-re	eturn mode operation, see		
		The data stack pointer (S phase of the pipeline. Th program counter (PC), of th	e 16 LSBs of the	return address, from the		
		The system stack pointer ( phase of the pipeline. The le of the return address are p	pop context bits conc	atenated with the 8 MSBs		
		The PC is loaded with the s flow execution context flag		ddress. The active control		
		System Stack (SSP)		Data Stack (SP)		
After Save		-1 (Loop bite) $PC(23-16)$	$\begin{array}{rcl} \text{After} & \\ \text{Save} & \rightarrow & \text{SP} = y - 1 \end{array}$	PC(15–0)		
Before Save			efore $\rightarrow$ SP = y Save	Previously saved data		
Status	Bits	Affected by none				
		Affects none				

**Repeat** This instruction cannot be repeated.

Syntax	Description
	Program control is passed to the program address label (FOO) assembled into an absolute address defined by the 24-bit value.

# Circular Addressing Qualifier

# Syntax Characteristics

				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[1]	circular()			No	1	1	AD
Opcode						100	01 1101
Operands		none					
Description		This instruction is an instruction qualifier that can be paralleled only with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. This instruction cannot be executed in parallel with any other types of instructions and it cannot be executed as a stand-alone instruction (assembler generates an error message).					
		When this instruction is used in parallel, all modifications of ARx and CDP pointer registers used in the indirect addressing mode are done circularly (as if ST2_55 register bits 0 to 8 were set to 1).					
Status	Bits	Affected by	none				
		Affects	none				
Repeat		This instruction can be repeated.					

# Clear Accumulator, Auxiliary, or Temporary Register Bit

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	bit(src, Baddr)	= #0		No	3	1	Х
Opcode			1110	1100 AA	AA A	AAI FSS	S 001x
Opera	nds	Baddr, src					
Description		This instruction performs a bit manipulation:					
		🗋 In the D-u	unit ALU, if the source (s	src) register (	operan	d is an ac	cumulator.
			unit ALU, if the source ( y register.	(src) register	opera	nd is an a	auxiliary or
			n clears to 0 a single bit, source register.	as defined b	by the b	oit address	sing mode,
		The generate	ed bit address must be w	vithin:			
		bit addre	en accessing accumulators are used to determin s not within 0–39, the sele	ne the bit po	sition).	If the ge	nerated bit
			en accessing auxiliary or nerated address are use		-	· •	
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	on can be repeated.				
See Al	so	See the follow	wing other related instruc	ctions:			
		Clear Me	mory Bit				
		Clear Sta	tus Register Bit				
			nent Accumulator, Auxilia	ary, or Temp	orary F	Register B	it
		Set Accu	mulator, Auxiliary, or Ter	mporary Reg	jister B	it	
Examp	ble						

Syntax	Description	
bit(AC0, AR3) = #0	The bit at the position defined by the content of AR3(4–0) in AC0 is cleared to 0.	

Clear Memory Bit

# Syntax Characteristics

			Parallel			
No.	Syntax		Enable Bit	Size	Cycles	Pipeline
[1]	bit(Smem, src)	) = #0	No	3	1	Х
Opcode		1110	0011 AA	AA AZ	AAI FSS	S 1101
Operar	nds	Smem, src				
Descri	ption	This instruction performs a bit manipuclears to 0 a single bit, as defined by for a memory (Smem) location.				
		The generated bit address must be win are used to determine the bit position	•	ly the 4	LSBs of t	he register
Status	Bits	Affected by none				
		Affects none				
Repeat	t	This instruction can be repeated.				
See Als	SO	See the following other related instru	ctions:			
		Clear Accumulator, Auxiliary, or T	emporary R	egister	Bit	
		Clear Status Register Bit				
		Complement Memory Bit				
		Set Memory Bit				

Syntax	Description
bit(*AR3, AC0) = #0	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is cleared to 0.

# Clear Status Register Bit

## **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	bit(ST0, k4) = #0	Yes	2	1	Х
[2]	bit(ST1, k4) = #0	Yes	2	1	х
[3]	bit(ST2, k4) = #0	Yes	2	1	х
[4]	bit(ST3, k4) = #0	Yes	2	1†	Х

<sup>†</sup> When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

Opcode	ST0	0100	011E kkkk	0000			
	ST1	0100	011E kkkk	0010			
	ST2	0100	011E kkkk	0100			
	ST3	0100	011E kkkk	0110			
Operands	k4, STx						
Description	These instructions perform a bit manipulation in the A-unit ALU.						
	These instructions clear to 0 a single bit, as de k4, in the selected status register (ST0, ST1			e value,			
	Compatibility with C54x devices (C54CM = 1)						
	C55x DSP status registers bit mapping (Figure correspond to C54x DSP status register bits	-	, page 5-92) d	oes not			
Status Bits	Affected by none						
	Affects Selected status bits						
Repeat	This instruction cannot be repeated.						
See Also	See the following other related instructions:						
	Clear Accumulator, Auxiliary, or Tempora	ary Regist	ter Bit				
	Clear Memory Bit						
	Set Status Register Bit						

Syntax			Description
bit(ST2, #ST2_	_AR2LC) = #0; AF	R2LC = bit 2	The ST2 bit position defined by the label (ST2_AR2LC, bit 2) is cleared to 0.
Before		After	
ST2_55	0006	ST2_55	0002

ST0_55								
15	14	13		12		11	10	9
ACOV2 <sup>†</sup>	ACOV3 <sup>†</sup>	TC1	t	TC2	C2 CARRY		RRY ACOV0	
R/W-0	R/W–0	R/W-	1	R/W–1	R	/W–1	R/W–0	R/W–0
8								0
				DP				
				R/W–0				
ST1_55								
15	14	13	12	11	l	10	9	8
BRAF	CPL	XF	HM	INT	М	<b>M40</b> <sup>†</sup>	SATD	SXMD
R/W–0	R/W–0	R/W–1	R/W–0	R/W	/—1	R/W–0	R/W–0	R/W-1
7	6	5	4					0
C16	FRCT	C54CM <sup>†</sup>				ASM		
R/W–0	R/W-0	R/W-1				R/W-0		
ST2_55								
15	14	13	12	11	I	10	9	8
ARMS	Rese	erved	DBGM	EALL	.OW	RDM	Reserved	CDPLC
R/W–0			R/W-1	R/W	/—0	R/W-0		R/W–0
7	6	5	4	3		2	1	0
AR7LC	AR6LC	AR5LC	AR4LC	AR3	LC	AR2LC	AR1LC	AR0LC
R/W–0	R/W-0	R/W–0	R/W–0	R/W	/—0	R/W–0	R/W–0	R/W–0
ST3_55								
15	14	13	12	11				8
CAFRZ <sup>†</sup>	CAEN <sup>†</sup>	CACLR <sup>†</sup>	<b>HINT</b> ‡		R	eserved (alwa	ays write 1100	b)
R/W–0	R/W–0	R/W–0	R/W-1					
7	6	5	4		3	2	1	0
CBERR <sup>†</sup>	MPNMC§	SATA <sup>†</sup>	R	eserved		CLKOFF	SMUL	SST
R/W–0	R/W–pins	R/W–0				R/W-0	R/W–0	R/W–0

# Figure 5–1. Status Registers Bit Mapping

**Legend:** R = Read; W = Write; -n = Value after reset

<sup>†</sup> Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

<sup>‡</sup>The HINT bit is not used for all C55x host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

§ The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see the boot loader section of its data sheet.

5-92 Instruction Set Descriptions

# Compare Accumulator, Auxiliary, or Temporary Register Content

## **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	<b>TC1 = uns(</b> src F	RELOP dst	:)			Yes	3	1	Х	
[2]	<b>TC2</b> = uns(src F	RELOP dst	:)			Yes	3	1	Х	
Opcode	e	TC1			0001	001E FS	SS CO	c00 FDE	D xux0	
		TC2			0001	001E FS	SS C	c00 FDI	D xuxl	
Operan	ds	dst, RELOP, src, TCx								
		tempora A-unit A it is clea The cor accumu	ry regist LU. If the red to 0 nparisor lator co	er TAx, t e compa n deper mpariso	ccumulator A the 16 lowest b arison is true, t nds on the op ns. As the fol comparison and	bits of ACx and the TCx stat otional uns llowing table	re com us bit is keywo e show	pared with s set to 1; and and o vs, the un	n TAx in the otherwise, n M40 for s keyword	
		for accu	mulator	compar	isons.					
		uns	src	dst	Comparison					
		no	TAx	TAy	16-bit signed					
		no	TAx	ACy	16-bit signed	•				
		no	ACx							
		no	ACx	ACy	if $M40 = 0, 32$ if $M40 = 1, 40$					
		yes	TAx	TAy	16-bit unsigne	-	-			
		yes	TAx	ACy	16-bit unsigne	ed compariso	n in A-u	init ALU		
		yes	ACx	TAy	16-bit unsigne	ed compariso	n in A-u	init ALU		
		yes	ACx	ACy	if M40 = 0, 32 if M40 = 1, 40	•	•			
		Compa	tibility v	vith C54	4x devices (C	54CM = 1)				

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

# Compare Accumulator, Auxiliary, or Temporary Register Content

Status Bits	Affected by	C54CM, M40		
	Affects	TCx		
Repeat	This instruction	a can be repeated.		
See Also	See the following other related instructions:			
	Compare A	accumulator, Auxiliary, or Temporary Register Content with AND		
	Compare A	Accumulator, Auxiliary, or Temporary Register Content with OR		
	Compare A	ccumulator, Auxiliary, or Temporary Register Content Maximum		
	Compare A	ccumulator, Auxiliary, or Temporary Register Content Minimum		
	Compare N	Memory with Immediate Value		

# Example 1

Syntax		Descr	Description					
TC1= AC1 = = T1			The signed content of AC1(15–0) is compared to the content of T1 and because they are equal, TC1 is set to 1.					
Before			After					
AC1	00 0028	0400	AC1	00 0028 0400				
T1		0400	Τ1	0400				
TC1		0	TC1	1				

Syntax	Description
TC1= T1 > = AC1	The content of T1 is compared to the signed content of AC1(15–0). The content of
	T1 is greater than the content of AC1, TC1 is set to 1.

Before			After			
т1		0500	Т1			0500
AC1	80 0000	0400	AC1	80	0000	0400
TC1		0	TC1			1

Compare Accumulator, Auxiliary, or Temporary Register Content with AND

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	TCx = TCy & ur	ns(src RELOP dst	)	Yes	3	1	Х	
[2]	TCx = !TCy & u	u <mark>ns(</mark> src RELOP ds	t)	Yes	3	1	Х	
Description		ALU. Two accu are compared.	ions perform a compa umulator, auxiliary regi When an accumulato ster TAx, the 16 lowest	sters, and ter or ACx is con	mporar npared	y register with an a	s contents auxiliary or	
Status Bits		Affected by C54CM, M40, TCy						
		Affects TCx						
See Als	50	See the following other related instructions:						
		Compare Accumulator, Auxiliary, or Temporary Register Content						
		Compare Accumulator, Auxiliary, or Temporary Register Content with OR						
		Compare Accumulator, Auxiliary, or Temporary Register Content Maxim					t Maximum	
		Compare Accumulator, Auxiliary, or Temporary Register Conte			er Conter	nt Minimum		
		Compare Memory with Immediate Value						

Compare Accumulator, Auxiliary, or Temporary Register Content with AND

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = TCy & uns(src RELOP dst)				
[1a]	TC1 = TC2 & uns(src RELOP dst)	Yes	3	1	Х
[1b]	TC2 = TC1 & uns(src RELOP dst)	Yes	3	1	х

Opcode

0001 001E FSSS cc01 FDDD Outt

**Operands** dst, RELOP, src, TC1, TC2

**Description** This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ANDed with TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	If M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	If M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

## Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

 Status Bits
 Affected by
 C54CM, M40, TCy

 Affects
 TCx

**Repeat** This instruction can be repeated.

Syntax		Descri	Description				
TC2 = TC	C1 & AC1 == AC2		The content of AC1(31–0) is compared to the content of AC2(31–0). The contents are equal (true), TC2 = TC1 & 1.				
Before		After					
AC1	80 0028 0400	AC1	80 0028 0400				
AC2	00 0028 0400	AC2	00 0028 0400				
м40	0	M40	0				
TC1	1	TC1	1				
TC2	0	TC2	1				

Compare Accumulator, Auxiliary, or Temporary Register Content with AND

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = !TCy & uns(src RELOP dst)				
[2a]	TC1 = !TC2 & uns(src RELOP dst)	Yes	3	1	х
[2b]	TC2 = !TC1 & uns(src RELOP dst)	Yes	3	1	х

Opcode

0001 001E FSSS cc01 FDDD lutt

**Operands** dst, RELOP, src, TC1, TC2

**Description** This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ANDed with the complement of TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

## Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

 Status Bits
 Affected by
 C54CM, M40, TCy

 Affects
 TCx

**Repeat** This instruction can be repeated.

Syntax		Descrip	Description				
TC2 = !T(	C1 & AC1 == AC2		The content of AC1(31–0) is compared to the content of AC2(31–0). The contents are equal (true), TC2 = $!TC1 \& 1$ .				
Before		After					
AC1	80 0028 0400	AC1	80 0028 0400				
AC2	00 0028 0400	AC2	00 0028 0400				
M40	0	M40	0				
TC1	1	TC1	1				
TC2	0	TC2	0				

Compare Accumulator, Auxiliary, or Temporary Register Content with OR

### **Syntax Characteristics**

No. Syntax		Enable Bit	Size	Cycles	Pipeline
[1] TCx = TCy   <b>L</b>	ns(src RELOP dst)	Yes	3	1	Х
[2] TCx = !TCy	<mark>uns(</mark> src RELOP dst <mark>)</mark>	Yes	3	1	х

**Description** These instructions perform a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU.

## Status Bits Affected by C54CM, M40, TCy

Affects TCx

See Also See the following other related instructions:

- Compare Accumulator, Auxiliary, or Temporary Register Content
- Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- Compare Memory with Immediate Value

### Compare Accumulator, Auxiliary, or Temporary Register Content with OR

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = TCy   uns(src RELOP dst)				
[1a]	TC1 = TC2   uns(src RELOP dst)	Yes	3	1	х
[1b]	TC2 = TC1   uns(src RELOP dst)	Yes	3	1	х

Opcode

0001 001E FSSS cc10 FDDD Outt

Operands dst, RELOP, src, TC1, TC2

**Description** This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ORed with TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

## Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits	Affected by	C54CM, M40, TCy
	Affects	TCx

**Repeat** This instruction can be repeated.

Syntax	Description
TC2 = TC1   uns(AC1 != AR1)	The unsigned content of AC1(15–0) is compared to the unsigned content
	of AR1. The contents are equal (false), TC2 = TC1   0.

Before		After	
AC1	00 8028 0400	AC1	00 8028 0400
AR1	0400	AR1	0400
TC1	1	TC1	1
TC2	0	TC2	1

### Compare Accumulator, Auxiliary, or Temporary Register Content with OR

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	TCx = !TCy   uns(src RELOP dst)				
[2a]	TC1 = !TC2   uns(src RELOP dst)	Yes	3	1	Х
[2b]	TC2 = !TC1   uns(src RELOP dst)	Yes	3	1	х

Opcode

0001 001E FSSS cc10 FDDD lutt

Operands dst, RELOP, src, TC1, TC2

**Description** This instruction performs a comparison in the D-unit ALU or in the A-unit ALU. Two accumulator, auxiliary registers, and temporary registers contents are compared. When an accumulator ACx is compared with an auxiliary or temporary register TAx, the 16 lowest bits of ACx are compared with TAx in the A-unit ALU. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0. The result of the comparison is ORed with the complement of TCy; TCx is updated with this operation.

The comparison depends on the optional uns keyword and on M40 for accumulator comparisons. As the following table shows, the uns keyword specifies an unsigned comparison and M40 defines the comparison bit width for accumulator comparisons.

uns	src	dst	Comparison Type
no	TAx	TAy	16-bit signed comparison in A-unit ALU
no	TAx	ACy	16-bit signed comparison in A-unit ALU
no	ACx	TAy	16-bit signed comparison in A-unit ALU
no	ACx	ACy	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU
yes	TAx	TAy	16-bit unsigned comparison in A-unit ALU
yes	TAx	ACy	16-bit unsigned comparison in A-unit ALU
yes	ACx	TAy	16-bit unsigned comparison in A-unit ALU
yes	ACx	ACy	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU

## Compatibility with C54x devices (C54CM = 1)

Contrary to the corresponding C54x instruction, the C55x register comparison instruction is performed in execute phase of the pipeline.

When C54CM = 1, the conditions testing the accumulators content are all performed as if M40 was set to 1.

Status Bits	Affected by	C54CM, M40, TCy		
	Affects	TCx		

**Repeat** This instruction can be repeated.

Syntax	Description
TC2 = !TC1   uns(AC1 != AR1)	The unsigned content of AC1(15–0) is compared to the unsigned content
	of AR1. The contents are equal (false), TC2 = !TC1   0.

Before		After	
AC1	00 8028 0400	AC1 00 8028 0400	
AR1	0400	AR1 0400	
TC1	1	TC1 1	
TC2	1	тс2 0	

Compare Accumulator, Auxiliary, or Temporary Register Content

	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = <b>max(</b> sro	:, dst <b>)</b>		Yes	2	1	Х
Opcod	e			00	10 11	1E FSS	S FDDD
Operar	nds	dst, sro	;				
Descri	ption	A-unit conten auxilia with TA	struction performs a maximu ALU. Two accumulator, au ts are compared. When an y or temporary register TAx x in the A-unit ALU. If the co herwise, it is cleared to 0.	xiliary registers a accumulator <i>a</i> , the 16 lowest	, and t ACx is bits of	temporary compare ACx are	y registers ed with an compared
		🗋 Wł	nen the destination operand	(dst) is an accu	umulato	or:	
		•	If an auxiliary or temporary instruction, the 16 LSBs of extended to 40 bits accord	the auxiliary or t		-	
			The operation is performed	d on 40 bits in tl	ne D-u	nit ALU:	
			If M40 = 0, src(31–0) content extremum value is stored content, the CARRY status	in dst. If the e	extremu	im value	is the src
			step1:if (src(31-	0) > dst(31-	-0))		
			<pre>step2: { CARRY = 0</pre>	; dst(39-0)	= src	:(39-0)	}
			else				
			<pre>step3:CARRY = 1</pre>				
			If M40 = 1, src(39–0) conte extremum value is stored content, the CARRY status	in dst. If the e	extremu	im value	is the src
			step1:if (src(39-	0) > dst(39-	-0))		
			<pre>step2: { CARRY = 0</pre>	; dst(39-0)	= src	:(39-0)	}
			else				
			<pre>step3:CARRY = 1</pre>				
			There is no overflow detec	tion, overflow re	eport, a	and satura	ation.

- When the destination operand (dst) is an auxiliary or temporary register:
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The operation is performed on 16 bits in the A-unit ALU:

The src(15–0) content is compared to the dst(15–0) content. The extremum value is stored in dst.

```
step1:if (src(15-0) > dst(15-0))
step2:dst = src
```

There is no overflow detection and saturation.

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. When the destination operand (dst) is an auxiliary or temporary register, the instruction execution is not impacted by the C54CM status bit. When the destination operand (dst) is an accumulator, this instruction always compares the source operand (src) with AC1 as follows:

- If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD
- The operation is performed on 40 bits in the D-unit ALU:

The src(39–0) content is compared to AC1(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

step1: if (src(39-0) > AC1(39-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
else
step3: { CARRY = 1; dst(39-0) = AC1(39-0) }

There is no overflow detection, overflow report, and saturation.

Status Bits Affected by C54CM, M40, SXMD

- Affects CARRY
- **Repeat** This instruction can be repeated.

## See Also See the following other related instructions:

- Compare Accumulator, Auxiliary, or Temporary Register Content
- Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
- Compare and Select Accumulator Content Maximum
- Compare Memory with Immediate Value

### Example 1

Syntax			Desc	Description					
				The content of AC2 is less than the content of AC1, the content of AC1 remains the same and the CARRY status bit is set to 1.					
Before				After					
AC2	00	0000	0000	AC2	00	0000	0000		
AC1	00	8500	0000	AC1	00	8500	0000		
SXMD			1	SXMD			1		
M40			0	M40			0		
CARRY			0	CARRY			1		

## Example 2

Syntax			Desc	Description					
AC1 = max	x(AR1,	AC1)			ent of AR1 is less than the content of AC1, the content of AC1 remains and the CARRY status bit is set to 1.				
Before				After					
AR1			8020	AR1			8020		
AC1	00	0000	0040	AC1	00 0	0000	0040		
CARRY			0	CARRY			1		

Syntax		Desc	Description				
			e content of AC1(15–0) is greater than the content of T1, the content of C1(15–0) is stored in T1 and the CARRY status bit is cleared to 0.				
Before			After				
AC1	00 0000	8020	AC1	00 0000 8020			
Т1		8010	T1	8020			
CARRY		0	CARRY	0			

Compare Accumulator, Auxiliary, or Temporary Register Content Minimum

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = <b>min(</b> src	, dst <b>)</b>			Yes	2	1	Х
Opcod	e				00	11 00	OE FSS	S FDDD
Opera	nds	dst, sro	;					
Descri	ption	A-unit conten auxiliar with TA	struction performs a ALU. Two accumula ts are compared. V y or temporary regis x in the A-unit ALU. herwise, it is cleared	ator, auxilia Vhen an ac ster TAx, the If the comp	y registers cumulator e 16 lowest	, and t ACx is bits of	temporar compare ACx are	y registers ed with an compared
		🗋 Wł	en the destination o	operand (dst	) is an accu	umulato	or:	
		•	If an auxiliary or ten instruction, the 16 I extended to 40 bits	LSBs of the a	auxiliary or		•	. ,
			The operation is pe	erformed on	40 bits in t	he D-u	nit ALU:	
			If M40 = 0, src(31- extremum value is content, the CARR	s stored in a	dst. If the e	extremu	um value	is the src
			step1:if (s	src(31-0)	< dst(31-	-0))		
			step2:{ CAR	RY = 0; d	st(39-0)	= src	:(39-0)	}
			else					
			step3:CARRY	1 = 1				
			If M40 = 1, src(39- extremum value is content, the CARR	s stored in a	dst. If the e	extremu	um value	is the src
			step1:if (s	src(39-0)	< dst(39-	-0))		
			step2:{ CAR	RY = 0; d	st(39-0)	= src	:(39-0)	}
			else					
			step3:CARRY					
			There is no overflo	w detection	overflow r	eport, a	and satur	ation.

- When the destination operand (dst) is an auxiliary or temporary register:
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The operation is performed on 16 bits in the A-unit ALU:

The src(15–0) content is compared to the dst(15–0) content. The extremum value is stored in dst.

step1:if (src(15-0) < dst(15-0))
step2:dst = src</pre>

There is no overflow detection and saturation.

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. When the destination operand (dst) is an auxiliary or temporary register, the instruction execution is not impacted by the C54CM status bit. When the destination operand (dst) is an accumulator, this instruction always compares the source operand (src) with AC1 as follows:

- ☐ If an auxiliary or temporary register is the source operand (src) of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD
- The operation is performed on 40 bits in the D-unit ALU:

The src(39–0) content is compared to AC1(39–0) content. The extremum value is stored in dst. If the extremum value is the src content, the CARRY status bit is cleared to 0; otherwise, it is set to 1.

```
step1:if (src(39-0) < AC1(39-0))
step2: { CARRY = 0; dst(39-0) = src(39-0) }
else
step3: { CARRY = 1; dst(39-0) = AC1(39-0) }</pre>
```

There is no overflow detection, overflow report, and saturation.

Status Bits	Affected by	C54CM, M40, SXMD			
	Affects	CARRY			
Repeat	This instruction can be repeated.				

See the following other related instructions:

- Compare Accumulator, Auxiliary, or Temporary Register Content
- Compare Accumulator, Auxiliary, or Temporary Register Content with AND
- Compare Accumulator, Auxiliary, or Temporary Register Content with OR
- Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
- Compare and Select Accumulator Content Minimum
- Compare Memory with Immediate Value

### Example

See Also

Syntax		Desc	Description				
			,	15–0) is greater than the content of T1, the content of T1 nd the CARRY status bit is set to 1.			
Before			After				
AC1	00 8000	0000	AC1	00 8000 0000			
T1		8020	T1	8020			
CARRY		0	CARRY	1			

Compare and Branch

**Syntax Characteristics** 

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	compare (uns(src RELOP K8)) goto L8	No	4	7/6	Х

† x/y cycles: x cycles = condition true, y cycles = condition false

Opcode	0110	1111 FSSS ccxu KKKK KKKK LLLL LLLL					
Operands	K8, L8, RELOP, src						
Description	This instruction performs a comparison operation between a source (src) register content and an 8-bit signed value, K8. The instruction performs a comparison in the D-unit ALU or in the A-unit ALU. The comparison is performed in the execute phase of the pipeline. If the result of the comparison is true, a branch occurs.						
	The program branch address is specified as an 8-bit signed offset, L8, relative to the program counter (PC). Use this instruction to branch within a 256-byte window centered on the current PC value. The comparison depends on the optional uns keyword and, for accumulator comparisons, on M40.						
	In the case of an unsigned comparison, the 8-bit constant, K8, is zer extended to:						
	16 bits, if the so	urce (src) operand is an auxiliary or temporary register.					
	■ 40 bits, if the s	ource (src) operand is an accumulator.					
	In the case of a extended to:	signed comparison, the 8-bit constant, K8, is sign					
	16 bits, if the so	urce (src) operand is an auxiliary or temporary register.					
	■ 40 bits, if the source (src) operand is an accumulator.						
	•	e shows, the uns keyword specifies an unsigned es the comparison bit width of the accumulator.					
	uns src	Comparison Type					
	no TAx	16-bit signed comparison in A-unit ALU					
	no ACx	if M40 = 0, 32-bit signed comparison in D-unit ALU if M40 = 1, 40-bit signed comparison in D-unit ALU					
	yes TAx	16-bit unsigned comparison in A-unit ALU					
	yes ACx	if M40 = 0, 32-bit unsigned comparison in D-unit ALU if M40 = 1, 40-bit unsigned comparison in D-unit ALU					

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the conditions testing the accumulator contents are all performed as if M40 was set to 1.

Status Bits Affected by C54CM, M40

Affects none

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

Branch Conditionally

Branch Unconditionally

Branch on Auxiliary Register Not Zero

#### Example 1

PC

Syntax	Description
compare (AC0 >= #12) goto branch	The signed content of AC0 is compared to the sign-extended 8-bit value (12). Because the content of AC0 is greater than or equal to 12, program control is passed to the program address label defined by branch (004078h).

004078

	compare (AC0 >= #12)			
			address:	00 4075
branch				00 4078
•				
Before		After		
AC0	00 0000 3000	AC0	00 0000	3000

PC

004071

Syntax	Description
compare (T1 != #1) goto branch	The content of T1 is not equal to 1, program control is passed to the next instruction (the branch is not taken).

branch	compare (T1 != #1  	address:	00407D 004080	
: Before T1	0000	After T1	0000	
PC	4079	PC	407D	

# Compare and Select Accumulator Content Maximum

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	max_diff(ACx,	ACy, ACz, ACw <b>)</b>		Yes	3	1	Х
[2]	max_diff_dbl(/	ACx, ACy, ACz, AG	Cw, TRNx <b>)</b>	Yes	3	1	Х
Descrij	otion		performs two paralleled on [2] performs a single				
Status	Bits	Affected by C54CM, M40, SATD					
		Affects	ACOVw, CARRY				
See Als	50	See the followi	ing other related instru	ictions:			
		Compare Accumulator, Auxiliary, or Temporary Register Content					
		Accumulator, Auxiliary, o	or Temporary	Registe	er Conten	t Maximum	
		Compare a	and Select Accumulato	r Content Mir	nimum		

### Compare and Select Accumulator Content Maximum

### Syntax Characteristics Parallel **Enable Bit** No. Syntax Size **Pipeline** Cycles Yes 3 [1] max\_diff(ACx, ACy, ACz, ACw) 1 Х Opcode 0001 000E DDSS 1100 SSDD nnnn Operands ACw, ACx, ACy, ACz Description This instruction performs two paralleled 16-bit extremum selections in the D-unit ALU in one cycle. This instruction performs a dual maximum search. The two operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulators are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path). For each datapath (high and low): ACx and ACy are the source accumulators. The differences are stored in accumulator ACw. The subtraction computation is equivalent to the dual 16-bit subtractions instruction. For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVw) is set. For the operations performed in the ALU low part, overflow is detected at bit position 15. ■ For the operations performed in the ALU high part, overflow is detected at bit position 31. For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31. $\Box$ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed: For the operations performed in the ALU low part, saturation values are 7FFFh (positive) and 8000h (negative). For the operations performed in the ALU high part, saturation values are 00 7FFFh (positive) and FF 8000h (negative).

- The extremum is stored in accumulator ACz.
- □ The extremum is searched considering the selected bit width of the accumulators:
  - for the lower 16-bit data path, the sign bit is extracted at bit position 15
  - for the higher 24-bit data path, the sign bit is extracted at bit position 31
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs:
  - TRN0 tracks the decision for the high part data path
  - TRN1 tracks the decision for the low part data path

If the extremum value is the ACx high or low part, the decision bit is cleared to 0; otherwise, it is set to 1:

```
TRN0 = TRN0 >> #1
TRN1 = TRN1 >> #1
ACw(39-16) = ACy(39-16) - ACx(39-16)
ACw(15-0) = ACy(15-0) - ACx(15-0)
If (ACx(31-16) > ACy(31-16))
    { bit(TRN0, 15) = #0 ; ACz(39-16) = ACx(39-16) }
else
    { bit(TRN0, 15) = #1 ; ACz(39-16) = ACy(39-16) }
if (ACx(15-0) > ACy(15-0))
    { bit(TRN1, 15) = #0 ; ACz(15-0) = ACx(15-0) }
else
    { bit(TRN1, 15) = #1 ; ACz(15-0) = ACy(15-0) }
```

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit data path (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD			
	Affects	ACOVw, CARRY			
Repeat	This instruction can be repeated.				

Syntax				Descriptio	Description					
max_diff(AC0, AC1, AC2, AC1)				from the cc Since SATI (saturation) AC1(15–0) in AC2. The is greater th TRN0(15) i	The difference is stored in AC1. The content of AC0(39–16) is subtracted from the content of AC1(39–16) and the result is stored in AC1(39–16). Since SATD = 1 and an overflow is detected, AC1(39–16) = FF 8000h (saturation). The content of AC0(15–0) is subtracted from the content of AC1(15–0) and the result is stored in AC1(15–0). The maximum is stored in AC2. The content of TRN0 and TRN1 is shifted right 1 bit. AC0(31–16) is greater than AC1(31–16), AC0(39–16) is stored in AC2(39–16) and TRN0(15) is cleared to 0. AC0(15–0) is greater than AC1(15–0), AC0(15–0) is stored in AC2(15–0) and TRN1(15) is cleared to 0.					
Before				After						
AC0	10	2400	2222	AC0	10	2400	2222			
AC1	90	0000	0000	AC1	FF	8000	DDDE			
AC2	00	0000	0000	AC2	10	2400	2222			
SATD			1	SATD			1			
TRN0			1000	TRN0			0800			
TRN1			0100	TRN1			0080			
ACOV1			0	ACOV1			1			
CARRY			1	CARRY			0			

# Compare and Select Accumulator Content Maximum

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[2a]		ACx.	ACy, ACz, ACw <b>, TRN0)</b>		Yes	3120	1	Х	
[2b]	-		ACy, ACz, ACw <b>, TRN1)</b>		Yes	3	1	х	
			••• •• •						
Opcode	9	T	RN0	0001	000E DD	SS 11	LO1 SSE	D xxx0	
		T	RN1	0001	000E DD	SS 11	LO1 SSE	D xxxl	
Operan	nds	AC	w, ACx, ACy, ACz, TRNx						
Descrip	otion		is instruction performs a sin is instruction performs a m	-		selection	on in the D	D-unit ALU.	
			ACx and ACy are the two	source	accumulato	rs.			
			The difference between the source accumulators is stored in accumulator ACw.						
			The subtraction computation is equivalent to the subtraction instruction.						
			Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.						
			When an overflow is detected, the accumulator is saturated according to SATD.						
			The extremum between the source accumulators is stored in accumulator ACz.						
			The extremum computat maximum instruction. Ho the extremum search but	wever, t	he CARRY :	status t	bit is not u		
			According to the extremute the MSBs to the LSBs. If cleared to 0; otherwise, it	the extr	emum value				

```
If M40 = 0:
   TRNx = TRNx >> #1
   ACw(39-0) = ACy(39-0) - ACx(39-0)
```

```
If (ACx(31-0) > ACy(31-0))
   \{ bit(TRNx, 15) = \#0 ; ACz(39-0) = ACx(39-0) \}
else
   \{ bit(TRNx, 15) = #1; ACz(39-0) = ACy(39-0) \}
```

### If M40 = 1:

```
TRNx = TRNx >> #1
ACw(39-0) = ACy(39-0) - ACx(39-0)
If (ACx(39-0) > ACy(39-0))
   \{ bit(TRNx, 15) = \#0; ACz(39-0) = ACx(39-0) \}
else
   \{ bit(TRNx, 15) = #1; ACz(39-0) = ACy(39-0) \}
```

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. However to ensure compatibility versus overflow detection and saturation of the destination accumulator, this instruction must be executed with M40 = 0.

Status Bits	Affected by	C54CM, M40, SATD		
	Affects	ACOVw, CARRY		
Repeat	This instruction can be repeated			

Repeat

#### Example

Syntax					Description	on		
max_diff_	, AC2,	AC3, TRN1)	The difference is stored in AC3. The content of AC0 is subtracted from the content of AC1 and the result is stored in AC3. The maximum is stored in AC2. The content of TRN1 is shifted right 1 bit. AC0 is greater than AC1, AC0 is stored in AC2 and TRN1(15) is cleared to 0.					
Before				After				
AC0	10	2400	2222	AC0	10	2400	2222	
AC1	00	8000	DDDE	AC1	00	8000	DDDE	
AC2	00	0000	0000	AC2	10	2400	2222	
AC3	00	0000	0000	AC3	FO	5C00	BBBC	
M40			1	M40			1	
SATD			1	SATD			1	
TRN1			0080	TRN1			0040	
ACOV3			0	ACOV3			0	
CARRY			0	CARRY			0	

SPRU375G

# Compare and Select Accumulator Content Minimum

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	•	min_diff(ACx, ACy, ACz, ACw)				1	X
	•			Yes	3	·	
[2]	min_diff_dbl(A	Cx, ACy, ACz, AC	Cw, TRNx <b>)</b>	Yes	3	1	Х
Descrip	otion		performs two paralleled on [2] performs a single				
Status	Bits	Affected by	C54CM, M40, SATE	)			
		Affects	ACOVw, CARRY				
See Als	50	See the follow	ing other related instru	ictions:			
		Compare /	Accumulator, Auxiliary	, or Tempora	y Regi	ster Cont	ent
		Accumulator, Auxiliary,	or Temporary	Regist	er Conter	nt Minimum	
		Compare a	and Select Accumulato	or Content Ma	ximum		

### Compare and Select Accumulator Content Minimum

### Syntax Characteristics Parallel **Enable Bit** No. Size **Pipeline** Syntax Cycles Yes 3 [1] min\_diff(ACx, ACy, ACz, ACw) 1 Х Opcode 0001 000E DDSS 1110 SSDD xxxx Operands ACw, ACx, ACy, ACz Description This instruction performs two paralleled 16-bit extremum selections in the D-unit ALU in one cycle. This instruction performs a dual minimum search. The two operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulators are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit data path). For each datapath (high and low): ACx and ACy are the source accumulators. The differences are stored in accumulator ACw. The subtraction computation is equivalent to the dual 16-bit subtractions instruction. For each of the two computations performed in the ALU, an overflow detection is made. If an overflow is detected on any of the data paths, the destination accumulator overflow status bit (ACOVw) is set. For the operations performed in the ALU low part, overflow is detected at bit position 15. ■ For the operations performed in the ALU high part, overflow is detected at bit position 31. For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31. $\Box$ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed: For the operations performed in the ALU low part, saturation values are 7FFFh (positive) and 8000h (negative). For the operations performed in the ALU high part, saturation values are 00 7FFFh (positive) and FF 8000h (negative).

- The extremum is stored in accumulator ACz.
- □ The extremum is searched considering the selected bit width of the accumulators:
  - for the lower 16-bit data path, the sign bit is extracted at bit position 15
  - for the higher 24-bit data path, the sign bit is extracted at bit position 31
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs:
  - TRN0 tracks the decision for the high part data path
  - TRN1 tracks the decision for the low part data path

If the extremum value is the ACx high or low part, the decision bit is cleared to 0; otherwise, it is set to 1:

```
TRN0 = TRN0 >> \#1
TRN1 = TRN1 >> \#1
ACw(39-16) = ACy(39-16) - ACx(39-16)
ACw(15-0) = ACy(15-0) - ACx(15-0)
If (ACx(31-16) < ACy(31-16))
{ bit(TRN0, 15) = #0 ; ACz(39-16) = ACx(39-16) }
else
{ bit(TRN0, 15) = #1 ; ACz(39-16) = ACy(39-16) }
if (ACx(15-0) < ACy(15-0))
{ bit(TRN1, 15) = #0 ; ACz(15-0) = ACx(15-0) }
else
{ bit(TRN1, 15) = #1 ; ACz(15-0) = ACy(15-0) }
```

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit data path (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD		
	Affects	ACOVw, CARRY		
Repeat	This instruction can be repeated.			

Example	Э
---------	---

Syntax		Descriptio	on		
min_diff(A	C0, AC1, AC2, AC1)	from the cc Since SATI (saturation AC1(15–0) in AC2 (sig TRN1 is sh AC1(31–16 to 1. AC0(1	The difference is stored in AC1. The content of AC0(39–16) is subtracted from the content of AC1(39–16) and the result is stored in AC1(39–16). Since SATD = 1 and an overflow is detected, AC1(39–16) = FF 8000h (saturation). The content of AC0(15–0) is subtracted from the content of AC1(15–0) and the result is stored in AC1(15–0). The minimum is store in AC2 (sign bit extracted at bits 31 and 15). The content of TRN0 and TRN1 is shifted right 1 bit. AC0(31–16) is greater than or equal to AC1(31–16), AC1(39–16) is stored in AC2(39–16) and TRN0(15) is set to 1. AC0(15–0) is greater than or equal to AC1(15–0), AC1(15–0) is stored in AC2(15–0) and TRN1(15) is set to 1.		
Before		After			
AC0	10 2400 2222	AC0	10 2400 2222		
AC1	00 8000 DDDE	AC1	FF 8000 BBBC		
AC2	10 2400 2222	AC2	00 8000 DDDE		
SATD	1	SATD	1		
TRN0	0800	TRN0	8400		
TRN1	0040	TRN1	8020		
ACOV1	0	ACOV1	1		
CARRY	0	CARRY	1		

# Compare and Select Accumulator Content Minimum

	<b>a</b> .				Parallel	<u>.</u>	<b>.</b> .	
No.	Syntax	0	A Que A Q- A Que TRNO		Enable Bit	Size	Cycles	Pipeline
[2a]			Yes	3	1	Х		
[2b]	min_diff_dbl(A	ACx, /	ACy, ACz, ACw <b>, TRN1)</b>		Yes	3	1	Х
Opcode	e	T	RN0	0001	000E DD	SS 11	L11 SSD	D xxx0
		TI	RN1	0001	000E DD	SS 11	L11 SSE	D xxxl
Operar	lds	ACw, ACx, ACy, ACz, TRNx						
Descrij	otion		This instruction performs a single 40-bit extremum selection in the D-unit ALU. This instruction performs a minimum search.					
			ACx and ACy are the two	o source	accumulato	rs.		
			The difference between the ACw.	ne source	e accumulato	ors is st	tored in ac	cumulator
🗋 The		The subtraction computation is equivalent to the subtraction instruction.						
	Overflow detection and CARRY status bit depends on M subtraction borrow bit is reported in the CARRY status bit; the b is the logical complement of the CARRY status bit.							
			When an overflow is detected, the accumulator is saturated accordin SATD.			ccording to		
			The extremum between the source accumulators is stored in accur ACz.				cumulator	
			The extremum computer maximum instruction. Ho the extremum search but	wever, t	he CARRY s	status k	bit is not u	
According to the extremum found, a decision bit is shifted in the MSBs to the LSBs. If the extremum value is ACx, the dec cleared to 0; otherwise, it is set to 1.								

```
If M40 = 0:
```

```
TRNx = TRNx >> #1
ACw(39-0) = ACy(39-0) - ACx(39-0)
If (ACx(31-0) < ACy(31-0))
    { bit(TRNx, 15) = #0 ; ACz(39-0) = ACx(39-0) }
else
    { bit(TRNx, 15) = #1 ; ACz(39-0) = ACy(39-0) }</pre>
```

### If M40 = 1:

```
TRNx = TRNx >> #1
ACw(39-0) = ACy(39-0) - ACx(39-0)
If (ACx(39-0) < ACy(39-0))
{ bit(TRNx, 15) = #0 ; ACz(39-0) = ACx(39-0) }
else
{ bit(TRNx, 15) = #1 ; ACz(39-0) = ACy(39-0) }</pre>
```

### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if M40 status bit was locally set to 1. However to ensure compatibility versus overflow detection and saturation of the destination accumulator, this instruction must be executed with M40 = 0.

Status Bits	Affected by	C54CM, M40, SATD		
	Affects	ACOVw, CARRY		

**Repeat** This instruction can be repeated.

Syntax	Description
min_diff_dbl(AC0, AC1, AC2, AC3, TRN0)	The difference is stored in AC3. The content of AC0 is subtracted from the content of AC1 and the result is stored in AC3. The minimum is stored in AC2. The content of TRN0 is shifted right 1 bit. If AC0 is less than AC1, AC0 is stored in AC2 and TRN0(15) is cleared to 0; otherwise, AC1 is stored in AC2 and TRN0(15) is set to 1.

# Compare Memory with Immediate Value

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TC1 = (Smem :	== K16 <b>)</b>			No	4	1	Х
[2]	<b>TC2 = (</b> Smem :	<b>TC2</b> = (Smem == K16)					1	Х
Opcod	e	TC1	1111	0000 AAAA	AAAI KK	KK KI	ккк   ккк	к кккк
		TC2	1111	0001 AAAA	AAAI KK	кк кі	KKK KKK	к кккк
Operar	nds	K16, Smem, T	Cx					
Descri	ption	operand Sme	m is con	ms a comparison npared to the 1 bit is set to 1; o	6-bit signed	consta	ant, K16.	•
		if((Smem) =	= K16)					
		TCx = 1						
		else						
		TCx = 0						
Status	Bits	Affected by	none					
		Affects	TCx					
Repeat	t	ing mode to ac	cess the	be repeated wh memory opera tion can be rep	ind (Smem);	`	,	
See Al	SO	See the follow	ving othe	r related instruc	ctions:			
		Compare	Accumu	lator, Auxiliary,	or Temporar	y Regi	ster Cont	ent

# Example 1

Syntax		Descriptio	Description						
TC1 = (*AR	1+ == #400h)		nt addressed by AR1 is compared to the signed 16-bit value cause they are equal, TC1 is set to 1. AR1 is incremented b						
Before		After							
AR1	0285	AR1	0286						
0285	0400	0285	0400						
TC1	0	TC1	1						

Syntax		Descriptio	Description						
TC2 = (*AR1	l == #400h)		t addressed by AR1 is com ause they are not equal, T	pared to the signed 16-bit value C2 is cleared to 0.					
Before		After							
AR1	0285	AR1	0285						
0285	0000	0285	0000						
TC2	0	TC2	0						

## Complement Accumulator, Auxiliary, or Temporary Register Bit

## **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	cbit(src, Baddr)					No	3	1	X	
Opcod	e				1110	1100 AA	AA A	AAI FSS	SS 011x	
Operar	nds	Ва	ddr, src		Ī	I		I		
Descri	ption	Th	is instruction	performs a bit	manipu	ulation:				
			In the D-un	it ALU, if the s	ource (s	rc) register o	operan	d is an ac	cumulator.	
			In the A-un temporary	it ALU, if the s register.	source (	src) register	opera	nd is an a	auxiliary or	
			The instruction complements a single bit, as defined by the bit addressing mode, Baddr, of the source register.							
		The generated bit address must be within:								
			bit address	accessing acc are used to c not within 0–39	letermin	e the bit po	sition).	If the ge	nerated bit	
				accessing aux rated address	•		-			
Status	Bits	Aff	ected by	none						
		Aff	ects	none						
Repeat	t	Th	is instruction	can be repeat	ed.					
See Al	so	Se	e the followir	ng other relate	d instru	ctions:				
			Clear Accu	mulator, Auxilia	ary, or T	emporary R	egister	Bit		
			Compleme	nt Accumulato	r, Auxilia	ary, or Temp	orary F	Register C	ontent	
			Compleme	nt Memory Bit						
			Set Accum	ulator, Auxiliar	y, or Ter	nporary Reg	jister B	it		

Syntax		Description	
cbit(T0, AR1)		The bit at the posi	ition defined by the content of AR1(3–0) in T0 is complemented.
Before		After	
Т0	E000	Т0	F000
AR1	000C	AR1	000C

Complement Accumulator, Auxiliary, or Temporary Register Content

## **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit Size Cycles	Pipeline
[1]	dst = ~src	Yes 2 1	Х
Opcode	e	0011 011E FSSS	FDDD
Operar	nds	dst, src	
Descrip	otion	This instruction computes the 1s complement (bitwise complement content of the source register (src).	nt) of the
		When the destination (dst) operand is an accumulator:	
		The bit inversion is performed on 40 bits in the D-unit ALL result is stored in the destination accumulator.	J and the
		If an auxiliary or temporary register is the source (src) operatinstruction, the 16 LSBs of the auxiliary or temporary register extended.	
		When the destination (dst) operand is an auxiliary or temporary	register:
		The bit inversion is performed on 16 bits in the A-unit ALL result is stored in the destination auxiliary or temporary reg	
		<ul> <li>If an accumulator is the source (src) operand of the instru- 16 LSBs of the accumulator are used to perform the operation</li> </ul>	
Status	Bits	Affected by none	
		Affects none	
Repeat	:	This instruction can be repeated.	
See Als	50	See the following other related instructions:	
		Complement Accumulator, Auxiliary, or Temporary Register Bit	
		Negate Accumulator, Auxiliary, or Temporary Register Content	

## Example

Syntax			Descri	Description					
AC1 = ~AC0			The co	The content of AC0 is complemented and the result is stored in AC1.					
Before				After					
AC0	7E	2355	4FC0	AC0	7E 2355 4FC0				
AC1	00	2300	5678	AC1	81 DCAA B03F				

SPRU375G

# Complement Memory Bit

# Syntax Characteristics

Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
cbit(Smem, src)		No	3	1	Х
9	1110	0011 AA	AA A	AAI FSS	S 111x
nds	Smem, src				
otion	complements a single bit, as defin	ed by the c			
	5	•	ly the 4	LSBs of t	he register
Bits	Affected by none				
	Affects none				
	This instruction can be repeated.				
50	See the following other related instru	uctions:			
	Clear Memory Bit				
	Complement Accumulator, Auxil	iary, or Temp	orary F	Register B	it
	Complement Accumulator, Auxil	iary, or Temp	orary F	Register C	ontent
	Set Memory Bit				
	cbit(Smem, src)	cbit(Smem, src)         e       1110         ads       Smem, src         ption       This instruction performs a bit manip complements a single bit, as defin operand, of a memory (Smem) locat         The generated bit address must be w are used to determine the bit positio         Bits       Affected by none         Affects       none         This instruction can be repeated.         so       See the following other related instru-         Clear Memory Bit       Complement Accumulator, Auxil	Syntax       Enable Bit         cbit(Smem, src)       No         e       1110 0011   AA         nds       Smem, src         obtion       This instruction performs a bit manipulation in the complements a single bit, as defined by the coordination operand, of a memory (Smem) location.         The generated bit address must be within 0–15 (on are used to determine the bit position).         Bits       Affected by none         Affects       none         This instruction can be repeated.         so       See the following other related instructions:         Clear Memory Bit         Complement Accumulator, Auxiliary, or Temp	Syntax       Enable Bit       Size         cbit(Smem, src)       No       3         e        1110       0011       AAAA       A         nds       Smem, src	Syntax       Enable Bit       Size       Cycles         cbit(Smem, src)       No       3       1         e        1110       0011       AAAA       AAAI       FSS         nds       Smem, src       Indiana       Indiana       AAAI       FSS         obtion       This instruction performs a bit manipulation in the A-unit ALU. The complements a single bit, as defined by the content of the scooperand, of a memory (Smem) location.       The generated bit address must be within 0–15 (only the 4 LSBs of the are used to determine the bit position).         Bits       Affected by none       Affects none       This instruction can be repeated.         so       See the following other related instructions:       Clear Memory Bit         Complement Accumulator, Auxiliary, or Temporary Register B       Complement Accumulator, Auxiliary, or Temporary Register C

Syntax	Description
cbit(*AR3, AC0)	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is complemented.

# Compute Exponent of Accumulator Content

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	Tx = <b>exp(</b> ACx <b>)</b>			Yes	3	1	Х		
Opcod	e		0001	000E xxS	SS 10	00 xxd	ld xxxx		
Operar	nds	ACx, Tx							
Descri	ption	D-unit shifter. T	n computes the exponer The result of the operatic J is used to make the m	on is stored in	the ter				
		exponent is co subtracting 8 fr	is a signed 2s-complet omputed by calculating om this value. The numb needed to align the ac	the number per of leading	of lead j bits is	ling bits i the numb	n ACx and per of shifts		
		ACx is not modified after the execution of this instruction. If ACx is equal to 0, Tx is loaded with 0.							
			n produces in Tx the o ssa and Exponent of Accu	••		•	•		
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	This instructior	n can be repeated.						
See Als	S0	See the followi	ing other related instruc	ctions:					
		Compute N	Mantissa and Exponent	of Accumula	ator Co	ontent			

## Example

Syntax			Descr	Description						
T1 = exp(AC0)			conter	The exponent is computed by subtracting 8 from the number of leading bits in the content of AC0. The exponent value is a signed 2s-complement value in the –8 to 31 range and is stored in T1.						
Before				After						
ACO FF FFFF F			FFCB	CB ACO FF FFFF FFCB		FFCB				
Т1 0			0000	00 T1 0019						

SPRU375G

# Compute Mantissa and Exponent of Accumulator Content

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	ACy = mant(A	ACx), Tx = -exp(ACx)	Yes	3	1	X2			
Opcod	le	0001	000E DDS	SS 10	01 xxd	d xxxx			
Opera	nds	ACx, ACy, Tx							
Descri	ption	This instruction computes the exaccumulator ACx. The computation executed in the D-unit shifter. The e temporary register Tx. The A-unit is mantissa is stored in the accumulato	of the expo xponent is co used to make	nent a mpute	nd the m d and sto	nantissa is pred in the			
		The exponent is a signed 2s-complement value in the –31 to 8 range. The exponent is computed by calculating the number of leading bits in ACx and subtracting this value from 8. The number of leading bits is the number of shifts to the MSBs needed to align the accumulator content on a signed 40-bit representation.							
		The mantissa is obtained by aligning the ACx content on a signed 32-bit representation. The mantissa is computed and stored in ACy.							
		The shift operation is performed	on 40 bits.						
		When shifting to the LSBs, b	it 39 of ACx is	s exter	ided to bi	t 31.			
		■ When shifting to the MSBs, (	) is inserted a	it bit po	osition 0.				
		If ACx is equal to 0, Tx is loaded	with 8000h.						
		This instruction produces in Tx the Compute Exponent of Accumulator C	••		•	•			
Status	Bits	Affected by none							
		Affects none							
Repea	t	This instruction can be repeated.							
See Al	so	See the following other related instru	ctions:						
		Compute Exponent of Accumula	tor Content						

# Example 1

Syntax				Description				
AC1 = mant(AC0), T1 = -exp(AC0)			the content ment value computed b	of AC n the y alig	0 from -31 to ning th	tted by subtracting the number of leading bits in n 8. The exponent value is a signed 2s-comple- o 8 range and is stored in T1. The mantissa is he content of AC0 on a signed 32-bit representa- ie is stored in AC1.		
Before				After				
AC0	21	0A0A	0A0A	AC0	21	0A0A	0A0A	
AC1	FF	FFFF	F001	AC1	00	4214	1414	
Т1			0000	Т1			0007	

Syntax			Description					
AC1 = mant(AC0), T1 = -exp(AC0)			The exponent is computed by subtracting the number of leading bits in the content of AC0 from 8. The exponent value is a signed 2s-complement value in the $-31$ to 8 range and is stored in T1. The mantissa is computed by aligning the content of AC0 on a signed 32-bit representation. The mantissa value is stored in AC1.					
Before			After					
AC0	00 E804	0000	AC0	00 E804 0000				
AC1	FF FFFF	F001	AC1	00 7402 0000				
Т1		0000	Т1	0001				

## Count Accumulator Bits

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Tx = count(AC	x, ACy <b>, TC1)</b>			Yes	3	1	Х
[2]	Tx = count(AC	x, ACy <b>, TC2)</b>			Yes	3	1	х
Opcod	e	TC1		0001	000E xxS	S 10	10 SSd	d xxx0
		TC2		0001	000E XXS	S 10	10 SSd	d xxxl
Operar	nds	ACx, ACy, Tx,	ТСх					
Descri	ption		n performs bit fie e selected tempo e operation.					
		Accumulator ACx is ANDed with accumulator ACy. The number of bits set t 1 in the intermediary result is evaluated and stored in the selected temporar register (Tx). If the number of bits is even, the selected TCx status bit is cleare to 0. If the number of bits is odd, the selected TCx status bit is set to 1.						temporary t is cleared
Status	Bits	Affected by	none					
		Affects	ТСх					
Repeat	t	This instructior	n can be repeate	ed.				

Syntax	Description
T1 = count(AC1, AC2, TC1)	The content of AC1 is ANDed with the content of AC2, the number of bits set to 1 in the result is evaluated and stored in T1. The number of bits set to 1 is odd, TC1 is set to 1.
Before	After

Belore				ALCEL				
AC1	7E	2355	4FC0	AC1	7E	2355	4FC0	
AC2	OF	E340	5678	AC2	0F	E340	5678	
т1			0000	Т1			000B	
TC1			0	TC1			1	

## Dual 16-Bit Additions

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACy) = HI(Lr LO(ACy) = LO(			)	No	3	1	x
[2]	HI(ACx) = HI(Lr LO(ACx) = LO(				No	3	1	х
Descrij	ption	Th	ese instructi	ons perform two para	lleled additior	n opera	itions in o	ne cycle.
		loc ace	ally in dual cumulator a	s are executed on 40 16-bit mode. The 1 re separated from th higher 16-bit datapa	6 lower bits eir higher 24	of bot	h the AL	U and the
Status	Bits	Aff	Affected by C54CM, SATD, SXMD					
		Aff	Affects ACOVx, ACOVy, CARRY					
See Als	50	See the following other related instructions:						
			Addition					
			Addition or	Subtraction Conditio	nally			
			Addition or	Subtraction Conditio	nally with Shi	ft		
			Addition w	ith Parallel Store Accu	umulator Con	tent to	Memory	
			Addition, S	Subtraction, or Move A	Accumulator C	Content	Conditio	nally
			Dual 16-Bi	t Addition and Subtra	ction			
			Dual 16-Bi	t Subtraction and Add	lition			

## Dual 16-Bit Additions

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACy) = HI(L LO(ACy) = LO(				No	3	1	X
Opcod	e			1110	1110   AA	AA AZ	AAI SSI	D 000x
Operar	nds	AC	cx, ACy, Lmem					
Descri	ption	ope in c are	is instruction performs treations are executed o dual 16-bit mode. The 1 separated from their h her 16-bit datapath).	n 40 bits in t 6 lower bits	he D-unit AL s of both the	U that ALU a	is configu and the ad	ured locally
			The data memory ope	rand dbl(Ln	nem) is divic	led into	o two 16-b	oit parts:
			■ the lower part is us	ed as one of	f the 16-bit o	perand	s of the Al	LU low part
			the higher part is s used in the ALU h	-	ed to 24 bits	accord	ding to SX	(MD and is
			The data memory ope	rand dbl(Ln	nem) addres	ses ar	e aligned	:
			if Lmem address significant word =		most signifi	cant w	ord = Ln	nem, least
			if Lmem address significant word =		nost signific	ant w	ord = Ln	nem, least
			For each of the two of detection is made. If a destination accumulat	n overflow is	s detected o	n any c	of the data	
			■ For the operations at bit position 15.	performedi	n the ALU lo	w part,	overflow	is detected
			For the operation detected at bit post	•	ed in the A	LU hig	jh part, c	overflow is
			For all instructions, the part is reported in the extracted at bit positio	CARRY stat	-	-		-

- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) + HI(AC1), LO(AC0) = LO(*AR3) + LO(AC1)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of AC1(39–16) is added to the content addressed by AR3 and the result is stored in AC0(39–16). The content of AC1(15–0) is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

## Dual 16-Bit Additions

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[2]	HI(ACx) = HI(Li LO(ACx) = LO(					No	3	1	Х
Opcod	e				1110	1110 AA	AA AZ	AAI ssI	D 100x
Operai	nds	AC	x, Ln	nem, Tx					
Description		ope in c are	eratio dual e sep	truction performs ons are executed 16-bit mode. The arated from their 6-bit datapath).	on 40 bits in t 16 lower bits	the D-unit Al s of both the	U that ALU a	is configu and the ad	ured locally
			The	temporary regis	ter Tx:				
				is used as one o	f the 16-bit o	perands of t	he ALl	J low part	
				is duplicated and used in the ALU	-	o SXMD, sig	in exte	nded to 24	4 bits to be
			The	data memory op	erand dbl(Ln	nem) is divic	led into	o two 16-b	oit parts:
				the lower part is u	ised as one of	f the 16-bit o	perand	ls of the Al	LU low part
				the higher part is used in the ALU	-	ed to 24 bits	accord	ding to SX	(MD and is
			The	data memory op	erand dbl(Ln	nem) addres	sses ar	e aligned	:
				if Lmem addres		most signifi	cant w	rord = Ln	nem, least
				if Lmem addre significant word		nost signific	cant w	ord = Ln	nem, least
			dete	each of the two ection is made. If tination accumula	an overflow is	s detected o	n any c	of the data	
				For the operation at bit position 15	•	in the ALU lo	w part,	overflow	is detected
				For the operation detected at bit per		ed in the A	LU hig	gh part, c	overflow is

	part is rep	tructions, the carry of the operation performed in the ALU high orted in the CARRY status bit. The CARRY status bit is always at bit position 31.
		ently on each data path, if SATD = 1 when an overflow is on the data path, a saturation is performed:
		e operations performed in the ALU low part, saturation values FFh and 8000h.
		e operations performed in the ALU high part, saturation values ) 7FFFh and FF 8000h.
	Compatibility	with C54x devices (C54CM = 1)
	0. Overflow is a	= 1, this instruction is executed as if SATD is locally cleared to only detected and reported for the computation performed in the latapath (overflow is detected at bit position 31).
Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVx, CARRY
Repeat	This instructio	n can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) + T0, LO(AC0) = LO(*AR3) + T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is added to the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

## Dual 16-Bit Addition and Subtraction

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	No	3	1	Х
[2]	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx	No	3	1	Х

# **Description** These instructions perform two paralleled addition and subtraction operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Affects ACOVx, ACOVy, CARRY

- **See Also** See the following other related instructions:
  - Addition
  - Dual 16-Bit Additions
  - Dual 16-Bit Subtractions
  - Dual 16-Bit Subtraction and Addition
  - □ Subtraction

## Dual 16-Bit Addition and Subtraction

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Sm LO(ACx) = Sm				No	3	1	Х
Opcod	е			1101	1110 AA	AA Ai	AI ssI	D 1000
Opera	nds	ACx,	Smem, Tx					
Descri	ption	an ac D-uni both t	instruction performs ddition and subtract it ALU that is configu the ALU and the acc ard bits are attached	ion. The oper ured locally in umulator are	rations are dual 16-bit separated f	execut mode. rom the	ed on 40 The 16 lo	bits in the ower bits of
		ПΤ	he data memory op	erand Smem	:			
			is used as one o	f the 16-bit op	perands of t	he ALl	J low part	
			is duplicated and used in the ALU	-	o SXMD, sig	jn exte	nded to 24	4 bits to be
		ПТ	he temporary regist	ter Tx:				
			is used as one o	f the 16-bit op	perands of t	he ALl	J low part	
			is duplicated and used in the ALU	-	o SXMD, sig	in exte	nded to 24	4 bits to be
		d	for each of the two etection is made. If estination accumula	an overflow is	detected o	n any c	of the data	
			For the operation at bit position 15	•	n the ALU lo	w part,	overflow	is detected
			For the operation detected at bit per		d in the A	LU hig	jh part, c	overflow is
		p	or all instructions, to art is reported in the xtracted at bit positi	e CARRY stat	•	•		-

- □ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVx, CARRY
Repeat	This instruction	can be repeated.

Syntax	Description
HI(AC1) = *AR1 + T1, LO(AC1) = *AR1 – T1	Both instructions are performed in parallel. The content addressed by AR1 is added to the content of T1 and the result is stored in AC1(39–16). The duplicated content of T1 is subtracted from the duplicated content addressed by AR1 and the result is stored in AC1(15–0).

Before		After	
AC1	00 2300 0000	AC1	00 2300 A300
Т1	4000	Т1	4000
AR1	0201	AR1	0201
201	E300	201	E300
SXMD	1	SXMD	1
M40	1	M40	1
ACOV0	0	ACOV0	0
CARRY	0	CARRY	1

## Dual 16-Bit Addition and Subtraction

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[2]	HI(ACx) = HI(Li LO(ACx) = LO(					No	3	1	Х
Opcod	e				1110	1110 AA	AA AA	AAI ssI	DD 110x
Operands ACx, Lmem, Tx									
Descri	ption	an D-u bot	addi unit A th the	truction perform tion and subtra- LU that is config ALU and the ac bits are attache	ction. The ope gured locally ir ccumulator are	rations are dual 16-bit separated f	execut mode. rom the	ed on 40 The 16 lo	bits in the ower bits of
			The	temporary regi	ster Tx:				
				is used as one	of the 16-bit o	perands of t	he ALL	J low part	
				is duplicated ar used in the AL	-	o SXMD, sig	in exte	nded to 24	4 bits to be
			The	data memory c	operand dbl(Ln	nem) is divid	led into	o two 16-b	oit parts:
				the lower part is	used as one of	f the 16-bit o	perand	s of the Al	LU low part
				the higher part used in the AL	-	ed to 24 bits	accord	ding to SX	(MD and is
			The	data memory c	perand dbl(Ln	nem) addres	sses ar	e aligned	:
				if Lmem addre significant word		most signifi	cant w	ord = Ln	nem, least
				if Lmem addr significant word		nost signific	cant w	ord = Ln	nem, least
			dete	each of the tw ection is made. I tination accumu	f an overflow is	s detected o	n any c	of the data	
				For the operation at bit position 1	•	in the ALU lo	w part,	overflow	is detected
				For the operated detected at bit	•	ed in the A	LU hig	jh part, c	overflow is

- □ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- □ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

 Status Bits
 Affected by
 C54CM, SATD, SXMD

 Affects
 ACOVx, CARRY

**Repeat** This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) + T0, LO(AC0) = LO(*AR3) – T0	Both instructions are performed in parallel. When the Lmem address is even $(AR3 = even)$ : The content of T0 is added to the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

## Dual 16-Bit Subtractions

No.	Syntax			Parallel Enable bit	Size	Cycles	Pipeline	
[1]		.Cx) – <b>HI(</b> Lmem), (ACx) – <b>LO(</b> Lmem	))	No	3	1	Х	
[2]		mem) – <b>HI(</b> ACx), (Lmem) – <b>LO(</b> AC>	<)	No	3	1	Х	
[3]	HI(ACx) = Tx - LO(ACx) = Tx -	• •		No	3	1	Х	
[4]	HI(ACx) = HI(L LO(ACx) = LO		No	3	1	х		
Descri	ption	These instruct	ions perform two paral	leled subtract	ion ope	erations in	one cycle.	
locally in dual 16-b accumulator are se		s are executed on 40 I 16-bit mode. The 1 are separated from the e higher 16-bit datapa	6 lower bits eir higher 24	of bot	h the AL	U and the		
Status	Bits	Affected by	C54CM, SATD, SXM	SXMD				
	Affects ACOVx, A		ACOVx, ACOVy, CA	/y, CARRY				
See Al	so	See the follow	ing other related instru	uctions:				
		Addition o	r Subtraction Conditio	nally				
		Addition o	r Subtraction Conditio	nally with Shi	ft			
		Addition, S	Subtraction, or Move A	Accumulator C	Content	t Conditio	nally	
	Dual 16-Bit Addition		it Addition and Subtra	ction				
		Dual 16-B	it Subtraction and Add	dition				
		Subtract (	Conditionally					

- Subtract Conditionally
- □ Subtraction
- Subtraction with Parallel Store Accumulator Content to Memory

## Dual 16-Bit Subtractions

N -	Cumtere				Parallel	0:	Oucles	Dineline
<u>No.</u>	Syntax				Enable Bit	Size	Cycles	Pipeline
[1]	HI(ACy) = H LO(ACy) = I		HI(Lmem), – <b>LO(</b> Lmem)		No	3	1	X
Opcode				1110	1110 AA	AA A	AAI SSI	D 001x
Operar	Dperands ACx, ACy, Lmem							
Descri	ption	The loca acci	s instruction perform operations are exe Illy in dual 16-bit r umulator are separ ched to the higher ?	ecuted on 40 bi node. The 16 ated from thei	its in the D- lower bits r higher 24	unit AL of bot	U that is h the AL	configured U and the
			The data memory of	operand dbl(Lm	nem) is divid	ded into	o two 16-b	oit parts:
			the lower part is used as one of the 16-bit operands of the				ls of the Al	_U low part
			the higher part is sign extended to 24 bits according to SXI used in the ALU high part				MD and is	
			The data memory of	operand dbl(Lm	nem) addres	sses ar	e aligned	:
			■ if Lmem addressignificant word		most signifi	cant w	rord = Ln	nem, least
			■ if Lmem addr significant word		nost signific	cant w	ord = Ln	nem, least
			For each of the tw detection is made. I destination accumu	f an overflow is	detected o	n any c	of the data	
			■ For the operation at bit position 1		n the ALU lo	w part,	overflow	is detected
			For the operation detected at bit	•	ed in the A	LU hiç	gh part, c	overflow is
			For all instructions, part is reported in the extracted at bit pos	ne CARRY stat	-	-		-

- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	can be repeated.

Syntax	Description
HI(AC0) = HI(AC1) – HI(*AR3), LO(AC0) = LO(AC1) – LO(*AR3)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content addressed by AR3 (sign extended to 24 bits) is subtracted from the content of AC1(39–16) and the result is stored in AC0(39–16). The content addressed by AR3 + 1 is subtracted from the content of AC1(15–0) and the result is stored in AC0(15–0).

## Dual 16-Bit Subtractions

N	Cumter-				Parallel	0:	Qualar	Dineline		
No.	Syntax				Enable bit	Size	Cycles	Pipeline		
[2]	HI(ACy) = HI( LO(ACy) = LC	-			No	3	1	X		
Opcod	e			1110	1110 AA	AA A	AAI SSI	DD 010x		
Operar	nds	AC	x, ACy, Lmem							
Descri	ption	The loc ace	This instruction performs two paralleled subtraction operations in on The operations are executed on 40 bits in the D-unit ALU that is cor locally in dual 16-bit mode. The 16 lower bits of both the ALU a accumulator are separated from their higher 24 bits (the 8 guard attached to the higher 16-bit datapath).							
			The data memory	operand dbl(Lm	nem) is divid	ded into	o two 16-b	oit parts:		
			■ the lower part	is used as one of	the 16-bit o	perand	ls of the Al	LU low part		
			the higher par used in the Al		ed to 24 bits	to 24 bits according to SXMD and is				
			The data memory	operand dbl(Lm	nem) addres	sses ar	e aligned	:		
			■ if Lmem add significant wo	lress is even: r rd = Lmem + 1	most signifi	cant w	vord = Ln	nem, least		
				dress is odd: n rd = Lmem – 1	nost signific	cant w	ord = Ln	nem, least		
			For each of the t detection is made destination accum	. If an overflow is	detected o	n any c	of the data			
			■ For the operat at bit position	ions performed i 15.	n the ALU lo	w part,	overflow	is detected		
			For the operative opera	ations performe t position 31.	d in the A	LU hiç	gh part, c	overflow is		
			For all instructions part is reported in extracted at bit po	the CARRY stat	-	-		-		

- Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) – HI(AC1), LO(AC0) = LO(*AR3) – LO(AC1)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of AC1(39–16) is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The content of AC1(15–0) is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

## Dual 16-Bit Subtractions

No.	Syntax					Parallel Enable bit	Size	Cycles	Pipeline
[3]	HI(ACx) = Tx - LO(ACx) = Tx					No	3	1	Х
Opcod	e				1110	1110 AA	AA Ai	AAI ssI	DD 011x
Opera	nds	AC	x, Ln	nem, Tx					
Description		This instruction performs two paralleled subtraction operations in one c. The operations are executed on 40 bits in the D-unit ALU that is config locally in dual 16-bit mode. The 16 lower bits of both the ALU and accumulator are separated from their higher 24 bits (the 8 guard bits attached to the higher 16-bit datapath).							configured U and the
			The	temporary reg	ister Tx:				
				is used as one	of the 16-bit o	perands of t	he ALl	J low part	
			•	is duplicated an used in the AL	-	o SXMD, sig	jn exte	nded to 24	4 bits to be
			The	data memory	operand dbl(Ln	nem) is divid	ded into	o two 16-b	oit parts:
				the lower part is	s used as one o	f the 16-bit o	perand	ls of the Al	LU low part
			•	the higher part used in the AL	-	ed to 24 bits	accord	ding to SX	(MD and is
			The	data memory o	operand dbl(Ln	nem) addres	sses ar	e aligned	:
				if Lmem addr significant word		most signifi	cant w	rord = Ln	nem, least
			•	if Lmem addr significant word		nost signifi	cant w	ord = Ln	nem, least
			dete	each of the tw ection is made. tination accum	If an overflow is	s detected o	n any c	of the data	
				For the operation at bit position 1	•	in the ALU Ic	w part,	overflow	is detected
			•	For the opera detected at bit	•	ed in the A	LU hig	gh part, c	overflow is

	part is repo	ructions, the carry of the operation performed in the ALU high orted in the CARRY status bit. The CARRY status bit is always at bit position 31.
		ently on each data path, if SATD = 1 when an overflow is on the data path, a saturation is performed:
		e operations performed in the ALU low part, saturation values FFh and 8000h.
		e operations performed in the ALU high part, saturation values 7FFFh and FF 8000h.
	Compatibility	with C54x devices (C54CM = 1)
	0. Overflow is c	= 1, this instruction is executed as if SATD is locally cleared to only detected and reported for the computation performed in the atapath (overflow is detected at bit position 31).
Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVx, CARRY
Repeat	This instruction	n can be repeated.

Syntax	Description
HI(AC0) = T0 – HI(*AR3), LO(AC0) = T0 – LO(*AR3)	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content addressed by AR3 is subtracted from the content of T0 and the result is stored in AC0(39–16). The content addressed by AR3 + 1 is subtracted from the duplicated content of T0 and the result is stored in AC0(15–0).

## Dual 16-Bit Subtractions

No.	Syntax					Parallel Enable bit	Size	Cycles	Pipeline
[4]	HI(ACx) = HI(L LO(ACx) = LO					No	3	1	Х
Opcod	le				1110	1110 AA	AA AA	AAI ssI	D 101x
Opera	nds	AC	x, Tx	, Lmem					
Description		This instruction performs two paralleled subtraction operations in The operations are executed on 40 bits in the D-unit ALU that is locally in dual 16-bit mode. The 16 lower bits of both the AL accumulator are separated from their higher 24 bits (the 8 gua attached to the higher 16-bit datapath).						U that is h the AL	configured U and the
			The	e temporary reg	ister Tx:				
				is used as one	e of the 16-bit op	perands of t	he ALL	J low part	
				is duplicated a used in the AL	nd, according to U high part.	o SXMD, sig	gn exte	nded to 24	4 bits to be
			The	e data memory	operand dbl(Lm	nem) is divid	ded into	o two 16-b	oit parts:
				the lower part i	s used as one of	f the 16-bit o	perand	ls of the Al	_U low part
				the higher part used in the AL	t is sign extende .U high part	ed to 24 bits	accore	ding to SX	(MD and is
			The	e data memory	operand dbl(Lm	nem) addres	sses ar	e aligned	:
				if Lmem add significant wor	ress is even: ı d = Lmem + 1	most signifi	cant w	vord = Ln	nem, least
				if Lmem add significant wor	ress is odd: n d = Lmem – 1	nost signifi	cant w	ord = Ln	nem, least
			dete	ection is made.	wo computation If an overflow is ulator overflow	s detected o	n any c	of the data	
				For the operati at bit position	ons performed i 15.	in the ALU Ic	w part,	overflow	is detected
				For the opera detected at bit	ations performe position 31.	ed in the A	LU hiç	gh part, c	overflow is

	part is re	structions, the carry of the operation performed in the ALU high ported in the CARRY status bit. The CARRY status bit is always d at bit position 31.
		dently on each data path, if SATD = 1 when an overflow is on the data path, a saturation is performed:
		he operations performed in the ALU low part, saturation values 7FFFh and 8000h.
		he operations performed in the ALU high part, saturation values 00 7FFFh and FF 8000h.
	Compatibili	ty with C54x devices (C54CM = 1)
	0. Overflow is	M = 1, this instruction is executed as if SATD is locally cleared to s only detected and reported for the computation performed in the datapath (overflow is detected at bit position 31).
Status Bits	Affected by	C54CM, SATD, SXMD
	Affects	ACOVx, CARRY
Repeat	This instructi	on can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) – T0, LO(AC0) = LO(*AR3) – T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is subtracted from the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

## Dual 16-Bit Subtraction and Addition

## **Syntax Characteristics**

No.	Syntax	Parallel Enable bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	No	3	1	Х
[2]	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx	No	3	1	х

# Description These instructions perform two paralleled subtraction and addition operations in one cycle.

The operations are executed on 40 bits in the D-unit ALU that is configured locally in dual 16-bit mode. The 16 lower bits of both the ALU and the accumulator are separated from their higher 24 bits (the 8 guard bits are attached to the higher 16-bit datapath).

Status Bits	Affected by	C54CM, SATD, SXMD
-------------	-------------	-------------------

Affects ACOVx, ACOVy, CARRY

- **See Also** See the following other related instructions:
  - Addition
  - Dual 16-Bit Additions
  - Dual 16-Bit Addition and Subtraction
  - Dual 16-Bit Subtractions
  - Subtraction

## Dual 16-Bit Subtraction and Addition

No.	Syntax				Parallel Enable bit	Size	Cycles	Pipeline
[1]	HI(ACx) = Sn LO(ACx) = S				No	3	1	Х
Opcod	e			1101	1110 AA	AA A	AAI ssI	D 1001
Operai	nds	ACx,	Smem, Tx					
Descri	ption	a sub D-uni both t	otraction and t ALU that is the ALU and t	rforms two parallele addition. The oper configured locally in he accumulator are tached to the higher	ations are o dual 16-bit separated f	execute mode. rom the	ed on 40 The 16 lo	bits in the ower bits of
		ПТ	he data mem	nory operand Smem	:			
			is used as	one of the 16-bit op	perands of t	he ALl	J low part	
			•	ed and, according to e ALU high part	o SXMD, sig	gn exte	nded to 24	4 bits to be
		ПТ	he temporary	/ register Tx:				
			is used as	one of the 16-bit op	perands of t	he ALl	J low part	
			•	ed and, according to e ALU high part	o SXMD, sig	jn exte	nded to 24	4 bits to be
		d	etection is ma	ne two computation ade. If an overflow is cumulator overflow	detected o	n any c	of the data	
			For the ope at bit posit	erations performed i tion 15.	n the ALU lo	w part,	overflow	is detected
		•		perations performe at bit position 31.	d in the A	LU hiç	gh part, c	overflow is
		p	art is reported	ions, the carry of the d in the CARRY stat it position 31.	•	-		-

- □ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

Status Bits	Affected by	C54CM, SATD, SXMD				
	Affects	ACOVx, CARRY				
Repeat	This instruction can be repeated.					

Syntax	Description
HI(AC0) = *AR3 – T0, LO(AC0) = *AR3 + T0	Both instructions are performed in parallel. The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the duplicated content addressed by AR3 and the result is stored in AC0(15–0).

## Dual 16-Bit Subtraction and Addition

No.	Syntax					Parallel Enable bit	Size	Cycles	Pipeline
[2]	HI(ACx) = HI(L LO(ACx) = LO(					No	3	1	Х
Opcod	e				1110	1110 AA	AA AZ	AAI ssI	DD 111x
Operar	nds	AC	x, Ln	nem, Tx					
Descri	ption	a s D-u bot	subtra unit A th the	action and ad ALU that is con ALU and the a	rms two parallele dition. The oper ifigured locally ir accumulator are hed to the highe	ations are o dual 16-bit separated f	execute mode. rom the	ed on 40 The 16 lo	bits in the ower bits of
			The	e temporary re	gister Tx:				
				is used as on	e of the 16-bit o	perands of t	he ALl	J low part	
			•	is duplicated a used in the A	and, according to LU high part	o SXMD, sig	jn exte	nded to 24	4 bits to be
			The	e data memory	v operand dbl(Ln	nem) is divid	ded into	o two 16-b	oit parts:
				the lower part	is used as one of	f the 16-bit o	perand	ls of the Al	LU low part
				the higher pa used in the A	rt is sign extende LU high part	ed to 24 bits	accord	ding to SX	(MD and is
			The	e data memory	v operand dbl(Ln	nem) addres	sses ar	e aligned	:
					dress is even: ord = Lmem + 1	most signifi	cant w	rord = Ln	nem, least
					dress is odd: r ord = Lmem – 1	nost signifi	cant w	ord = Ln	nem, least
			dete	ection is made	two computation a. If an overflow is nulator overflow	s detected o	n any c	of the data	
			•	For the operated at bit position	tions performed i 15.	in the ALU lo	w part,	overflow	is detected
				For the oper detected at bi	rations performe it position 31.	ed in the A	LU hig	gh part, c	overflow is

- □ For all instructions, the carry of the operation performed in the ALU high part is reported in the CARRY status bit. The CARRY status bit is always extracted at bit position 31.
- □ Independently on each data path, if SATD = 1 when an overflow is detected on the data path, a saturation is performed:
  - For the operations performed in the ALU low part, saturation values are 7FFFh and 8000h.
  - For the operations performed in the ALU high part, saturation values are 00 7FFFh and FF 8000h.

## Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, this instruction is executed as if SATD is locally cleared to 0. Overflow is only detected and reported for the computation performed in the higher 24-bit datapath (overflow is detected at bit position 31).

 Status Bits
 Affected by
 C54CM, SATD, SXMD

 Affects
 ACOVx, CARRY

**Repeat** This instruction can be repeated.

Syntax	Description
HI(AC0) = HI(*AR3) – T0, LO(AC0) = LO(*AR3) + T0	Both instructions are performed in parallel. When the Lmem address is even (AR3 = even): The content of T0 is subtracted from the content addressed by AR3 and the result is stored in AC0(39–16). The duplicated content of T0 is added to the content addressed by AR3 + 1 and the result is stored in AC0(15–0).

# Execute Conditionally

## **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) execu		No	2	1	AD			
[2]	if (cond) execu	if (cond) execute(D_Unit)					2	1	Х
Descri	ption	These instructions evaluate a single condition defined by the cond field and allow you to control execution of all operations implied by the instruction or part of the instruction. See Table 1–3 for a list of conditions.							
			ase to the other	allows you to o execute phase ecution flow fro control of the e	of the pip om the exe	eline. Instrue ecute phase	ction [2 of the	] allows y pipeline.	/ou to only The use of
			These ins	structions may	be execut	ted alone.			
			These ins	structions may	be execut	ted with two	paralle	led instru	ctions.
Į			These instructions may be executed with the instruction with which it is paralleled.						
			These ins	structions may	be execut	ted with the p	oreviou	s instruc	tion.
			These instructions may be executed with the previous instruction and two paralleled instructions.						
			These instructions cannot be repeated.						
			These ins structure.	structions cann	ot be used	d as the last i	nstruc	tion in a r	epeat loop
				structions cann structions:	ot control	the executio	n of the	e followin	g program
			goto	(cond) goto	intr	blockrepeat			return_int
			call	(cond) call	idle	(cond) exect	ute(AD_	_unit)	
			return	(cond) return	reset	(cond) exect	ute(D_u	nit)	
			trap	localrepeat	repeat	while (cond)	repeat		
Status	Bits	Aff	ected by	ACOVx, CA	RRY, C54	4CM, M40, T	Сх		
		Aff	ects	ACOVx					

SPRU375G

## Execute Conditionally

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	if (cond) execu	ıte(Al	D_Unit)	No	2	1	AD
Opcod	e			10	01 01	.10 0CC	C CCCC
				10	01 11	10 0CC	C CCCC
				10	01 11	.11 0CC	c cccc
			e assembler selects the opcode de alleled pair.	epending on	the inst	ruction po	osition in a
Operar	nds	cor	nd				
Descri	ption	allo the	s instruction evaluates a single on the execution flows you to control the execution flow address phase to the execute ph of conditions.	ow of an instr	uction,	or instruct	ions, from
		cor the fals to	ten this instruction moves into the ndition specified in the cond field is conditional instruction(s) is read se, the conditional instruction(s) is the instruction following the cond dress defined by label. There is a	evaluated. If and execute not read and litional instru	the test d; if the progra ction(s)	ted condit tested co m control or to the	ion is true, ondition is is passed e program
			This instruction may be executed	d alone:			
			if(cond) execute(AL instruction_execute label:		onally		
			This instruction may be executed	l with two pa	ralleled	instructio	ins:
			<pre>if(cond) execute(AD instruction_1_execu    instruction_2_ex label:</pre>	tes_condit			
			This instruction may be execute paralleled:	ed with the	nstruct	ion with v	which it is
			if(cond) execute(AD    instruction_exec label:		ltiona	lly	

This instruction may be executed with a previous instruction:

```
previous_instruction
    || if(cond) execute(AD_unit)
    instruction_executes_conditionally
label:
```

☐ This instruction may be executed with a previous instruction and two paralleled instructions:

```
previous_instruction
    || if(cond) execute(AD_unit)
    instruction_1_executes_conditionally
    || instruction_2_executes_conditionally
label:
```

This instruction cannot be used as the last instruction in a repeat loop structure.

This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
trap	localrepeat	repeat	while (cond) repeat	

Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

**Repeat** This instruction cannot be repeated.

#### **Example 1**

Syntax	Description
if (TC1) execute(AD_unit)	TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1).
mar(*AR1+)	The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.
AC1 = AC1 + *AR1	

Before			After		
AC1	00 0000	4300	AC1	00 0000	6321
TC1		1	TC1		1
CARRY		1	CARRY		0
AR1		0200	AR1		0201
200		2020	200		2020
201		2021	201		2021

#### SPRU375G

Syntax		Description	Description				
if (TC1) exe	if (TC1) execute(AD_unit)		TC1 is not equal to 1, the next instruction is not executed (AR1 is not				
mar(*AR1+)		,	incremented). The content of AC1 is added to the content addressed by AR1 (2020h) and the result is stored in AC1.				
AC1 = AC1	+ *AR1	(20201) and th					
Before		After					
AC1	00 0000 4300	AC1	00 0000 6320				
TC1	0	TC1	0				
CARRY	1	CARRY	0				
AR1	0200	AR1	0200				
200	2020	200	2020				
201	2021	201	2021				

Execute Conditionally

### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bi	t Size	Cycles	Pipeline		
[2]	if (cond) execut	te(D_U	Init)	No	2	1	Х		
Opcod	e			1	01 0	110   1CC	c cccc		
				1	001 1	110 100	c cccc		
				1	001 1	111 100	c cccc		
			assembler selects the opcode lleled pair.	depending or	the ins	truction po	osition in a		
Operar	nds	cond	l						
Descrij	ption	allow the e beca	This instruction evaluates a single condition defined by the cond field and allows you to control the execution flow of an instruction, or instructions, from the execute phase of the pipeline. This instruction differs from instruction [1] because in this instruction operations performed in the address phase are always executed. See Table 1–3 for a list of conditions.						
When this instruction move condition specified in the conditional instruction (seconditional instruction) false, the conditional instruction to the instruction following address defined by label. T				is evaluated. I ad and execute is not read an nditional instru	f the tes ed; if the d progra uction(s	e tested condit e tested c am control ) or to the	ion is true, ondition is is passed e program		
			This instruction may be execut	ed alone:					
		1	if(cond) execute( instruction_execu label:		onally				
			This instruction may be execut	ed with two pa	aralleleo	d instructio	ons:		
		1	<pre>if(cond) execute() instruction_1_execute()    instruction_2_label:</pre>	cutes_condi					
		– F	This instruction may be exec paralleled. When this instruction executed conditionally is a stor atency for the condition setting	on syntax is us e-to-memory i	ed and	the instru	ction to be		
		1	if(cond) execute()    instruction_ex Label:		litiona	ally			

SPRU375G

This instruction may be executed with a previous instruction:

```
previous_instruction
    || if(cond) execute(D_unit)
    instruction_executes_conditionally
label:
```

This instruction may be executed with a previous instruction and two paralleled instructions:

```
previous_instruction
    || if(cond) execute(D_unit)
    instruction_1_executes_conditionally
    || instruction_2_executes_conditionally
label:
```

This instruction cannot be used as the last instruction in a repeat loop structure.

This instruction cannot control the execution of the following program control instructions:

goto	(cond) goto	intr	blockrepeat	return_int
call	(cond) call	idle	(cond) execute(AD_unit)	
return	(cond) return	reset	(cond) execute(D_unit)	
trap	localrepeat	repeat	while (cond) repeat	

```
Compatibility with C54x devices (C54CM = 1)
```

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits Affected by ACOVx, CARRY, C54CM, M40, TCx

200

201

Affects ACOVx

**Repeat** This instruction cannot be repeated.

#### Example 1

200

201

Syntax		Description	Description				
if (TC1) execute(D_unit) mar(*AR1+)			TC1 is equal to 1, the next instruction is executed (AR1 is incremented by 1).				
		The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.					
AC1 = AC <sup>2</sup>	1 + *AR1						
Before		After					
AC1	00 0000 4300	AC1	00 0000 6321				
TC1	1	TC1	1				
CARRY	1	CARRY	0				
AR1	0200	AR1	0201				

2020

2021

2020

2021

Syntax		Description	Description				
if (TC1) exe	cute(D_unit)		TC1 is not equal to 1, the next instruction would not be executed; however, since the next instruction is a pointer modification, AR1 is incremented by 1 in the address phase. The content of AC1 is added to the content addressed by AR1 + 1 (2021h) and the result is stored in AC1.				
mar(*AR1+)							
AC1 = AC1	+ *AR1						
Before		After					
AC1	00 0000 4300	AC1	00 0000 6321				
TC1	0	TC1	0				
CARRY	1	CARRY	0				
AR1	0200	AR1	0201				
200	2020	200	2020				
201	2021	201	2021				

## Expand Accumulator Bit Field

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	dst = <b>field_ex</b>	pand(ACx, k16)			No	4	1	Х	
Opcode	e		0111	0110 kkkk	kkkk kkk	ck kk	kk FDD	D 01SS	
Operar	nds	ACx, dst, k16							
Descriț	ption	This instruction performs a bit field manipulation in the D-unit shifter. When the destination register (dst) is an A-unit register (ARx or Tx), a dedicated bus carries the output of the D-unit shifter directly into dst.							
		The 16-bit field mask, k16, is scanned from the least significant bits (LSBs) the most significant bits (MSBs). According to the bit set to 1 in the bit fie mask, the 16 LSBs of the source accumulator (ACx) bits are extracted an separated with 0 toward the MSBs. The result is stored in the dst.					he bit field racted and		
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	This instruction	n can be	repeated.					
See Als	so	See the follow	ing othei	r related instruc	tions:				
		Extract Ac	cumulate	or Bit Field					

### Example

•	
Syntax	Description
T2 = field_expand(AC0,#8024h)	Each bit of the unsigned 16-bit value (8024h) is scanned from the LSB to the MSB to test for a 1. If the bit is set to 1, the bit in AC0 is extracted and separated with 0 toward the MSB in T2; otherwise, the corresponding bit in AC0 is not extracted. The result is stored in T2.

#### Execution

#k16 (8024h)	1000 0000 0010 0100
AC0(15-0)	0010 1011 0110 0 <b>101</b>
Т2	<b>1</b> 000 0000 00 <b>0</b> 0 0 <b>1</b> 00

Before		After	
AC0	00 2300 2B65	AC0	00 2300 2B65
Т2	0000	Т2	8004

#### Extract Accumulator Bit Field

### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	dst = field_ext	ract(ACx, k16)			No	4	1	Х	
Opcode	e		0111	0110 kkkk	kkkk kkk	k kk	kk FDD	D 00SS	
Operar	nds	ACx, dst, k16							
Descriț	ption	This instruction performs a bit field manipulation in the D-unit shifter. When the destination register (dst) is an A-unit register (ARx or Tx), a dedicated bus carries the output of the D-unit shifter directly into dst.							
		The 16-bit field mask, k16, is scanned from the least significant bits (LSBs the most significant bits (MSBs). According to the bit set to 1 in the bit fi mask, the corresponding 16 LSBs of the source accumulator (ACx) bits extracted and packed toward the LSBs. The result is stored in the dst.						he bit field (x) bits are	
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	This instruction	n can be	repeated.					
See Als	S0	See the following other related instructions:							
		Expand A	ccumulat	tor Bit Field					

### Example

Syntax	Description
T2 = field_extract(AC0,#8024h)	Each bit of the unsigned 16-bit value (8024h) is scanned from the LSB to the MSB to test for a 1. If the bit is set to 1, the corresponding bit in AC0 is extracted and packed toward the LSB in T2; otherwise, the corresponding bit in AC0 is not extracted. The result is stored in T2.

#### Execution

#k16 (8024h	)		<b>1</b> 000	0000	0010	0100			
AC0(15-0)			<b>0</b> 101	0101	10 <b>1</b> 0	1 <b>0</b> 10			
Т2			0000	0000	0000	0010			
Before				i	After				
AC0	00	2300	55AA	i	AC0		00	2300	55AA
Т2			0000		Г2				0002

#### SPRU375G

### Finite Impulse Response Filter, Antisymmetrical

No.	Syntax		Parallel Enable Bit Siz	ze Cycles Pipeline							
[1]	<b>firsn(</b> Xmem,	, Ymem, <b>coef(</b> Cmem <b>)</b> , ACx, ACy <b>)</b>	No 4	4 1 X							
Opcod	е	1000 0101 XX	ХМ ММҮҮ ҮМММ	11mm DDx1 DDU%							
Operar	nds	ACx, ACy, Cmem, Xmem, Ymem									
Descrij	ption	This instruction performs two pare (MAC), and subtraction. The firsn	-								
			ACy = ACy + (ACx * Cmem), ACx = (Xmem << #16) - (Ymem << #16)								
		MAC. The input operands of the m the content of a data memory oper	The first operation performs a multiplication and an accumulation in the D MAC. The input operands of the multiplier are the content of ACx(32–16) the content of a data memory operand Cmem, addressed using the coeffic addressing mode, sign extended to 17 bits.								
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.									
		Multiplication overflow detection depends on SMUL.									
			The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.								
			Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.								
		When an addition overflow i according to SATD.	s detected, the acc	umulator is saturated							
		For this instruction, the Cmem op some C55x-based devices, the BE and not to external memory. To p Cmem operand must not be mapp	bus is only connectorevent the generation	ted to internal memory ion of a bus error, the							
		The second operation subtracts the content of data memory operand Ymen shifted left 16 bits, from the content of data memory operand Xmem, shifte left 16 bits.									
		The operation is performed or	a 40 bits in the D-uni	it ALU.							
		Input operands are sign extended to 40 bits according to SXMD.									

	The shift of	operation is equivalent to the signed shift instruction.
	subtractio	detection and CARRY status bit depends on M40. The n borrow bit is reported in the CARRY status bit; the borrow bit cal complement of the CARRY status bit.
	When an o SATD.	overflow is detected, the accumulator is saturated according to
	Compatibility	v with C54x devices (C54CM = 1)
		ruction is executed with $M40 = 0$ , compatibility is ensured. When o overflow detection, report, and saturation is done after the ion.
Status Bits	Affected by	C54CM, FRCT, M40, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy, CARRY
Repeat	This instructio	n can be repeated.
See Also	See the follow	ing other related instructions:
	Finite Imp	ulse Response Filter, Symmetrical

Syntax					Description			
firsn(*AR0, *AR1, coef(*CDP), AC0, AC1)				by the co content of addressed	efficie f AC1 d by A ddress	nt data and th R1 shif	-16) multiplied by the content addressed pointer register (CDP) is added to the ne result is stored in AC1. The content ted left by 16 bits is subtracted from the AR0 shifted left by 16 bits and the result is	
Before				After				
AC0	00	6900	0000	AC0	00	4500	0000	
AC1	00	0023	0000	AC1	FF	D8ED	3F00	
*AR0			3400	*AR0			3400	
*AR1			EF00	*AR1			EF00	
*CDP			A067	*CDP			A067	
ACOV0			0	ACOV0			0	
ACOV1			0	ACOV1			0	
CARRY			0	CARRY			0	
FRCT			0	FRCT			0	
SXMD			0	SXMD			0	

## Finite Impulse Response Filter, Symmetrical

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline				
[1]	firs(Xmem,	Ymem, <b>coef(</b> Cmem <b>)</b> , ACx, ACy <b>)</b>	No	4	1	х				
Opcode	6	1000 0101   XXX	M MMYY YMM	IM 11	Lmm DDx	:0 DDU%				
Operar	nds	ACx, ACy, Cmem, Xmem, Ymem								
Descrip	otion	This instruction performs two para (MAC), and addition. The firs() oper	-	-	ply and a	ccumulate				
		ACy = ACy + (ACx * Cmem) ACx = (Xmem << #16) + (Yn								
		MAC. The input operands of the mu the content of a data memory operar	The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of $ACx(32-16)$ and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.							
		□ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.								
		Multiplication overflow detection	Multiplication overflow detection depends on SMUL.							
			The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.							
			Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.							
		When an addition overflow is detected, the accumulator is sa according to SATD.								
		For this instruction, the Cmem oper some C55x-based devices, the BB I and not to external memory. To pro Cmem operand must not be mappe	ous is only contended to the second tensor is a se	necteo eration	d to intern of a bus	al memory				
		data memory operand Xmem, shif	The second operation performs an addition operation between the content of data memory operand Xmem, shifted left 16 bits, and the content of data memory operand Ymem, shifted left 16 bits.							
		☐ The operation is performed on 4	10 bits in the D	-unit A	ALU.					
		Input operands are sign extended to 40 bits according to SXMD.								

	The shift o	peration is equivalent to the signed shift instruction.
	Overflow o	letection and CARRY status bit depends on M40.
	When an o SATD.	overflow is detected, the accumulator is saturated according to
	Compatibility	with C54x devices (C54CM = 1)
		uction is executed with $M40 = 0$ , compatibility is ensured. When o overflow detection, report, and saturation is done after the ion.
Status Bits	Affected by	C54CM, FRCT, M40, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy, CARRY
Repeat	This instruction	n can be repeated.
See Also	See the follow	ing other related instructions:
	Finite Impute	ulse Response Filter, Antisymmetrical

Syntax					Description			
firs(*AR0, *AR1, coef(*CDP), AC0, AC1)					The content of AC0(32–16) multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC1 and the result is stored in AC1. The content addressed by AR0 shifted left by 16 bits is added to the content addressed by AR1 shifted left by 16 bits and the result is stored in AC0.			
Before				Afte	r			
AC0	00	6900	0000	AC0	00 2300 0000			
AC1	00	0023	0000	AC1	FF D8ED 3F00			
*AR0			3400	*AR0	3400			
*AR1			EF00	*AR1	EF00			
*CDP			A067	*CDP	A067			
ACOV0			0	ACOV	0 0			
ACOV1			0	ACOV	1 0			
CARRY			0	CARR	У 1			
FRCT			0	FRCT	0			
SXMD			0	SXMD	0			

#### Idle

### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	idle				No	4	?	D
Opcode	e		0111	1010 xxxx	XXXX XXX	x xx	xx xx	x 110x
Operan	nds	none						
Descrip	otion	a reset occurs	. The pov	the program be ver-down mode egister access	that the proc	essoro	operates i	n depends
Status	Bits	Affected by	INTM					
		Affects	none					
Repeat	:	This instructio	on cannot	be repeated.				

idle

Least Mean Square

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	Ims(Xmem	, Ymem, ACx, A	Су)		No	4	1	Х	
Opcod	e		1000	0110 XXXM	MMYY YMN	im de	DD 110	x xxx%	
Operar	nds	ACx, ACy	v, Xmem, Yme	em					
Description			-	ns two parallele d addition. The	-		-	ultiply and	
				mem * Ymem), + (Xmem << #	16))				
		MAC. The operand	e input opera Xmem, sign e	orms a multiplic nds of the mult extended to 17 xtended to 17 b	iplier are the bits, and the	e conte	ent of dat	a memory	
		L If FR	☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.						
		🗋 Multip	Multiplication overflow detection depends on SMUL.						
		—	The 32-bit result of the multiplication is sign extended to 40 bits and adde to the source accumulator ACy.						
				detection deper cumulator overfl					
			n an addition ding to SATD	overflow is de	etected, the	accum	nulator is	saturated	
			•	performs an ado a memory opera					
		🗋 The c	peration is p	erformed on 40	bits in the D	-unit A	LU.		
		🗋 Input	operands are	e sign extended	to 40 bits a	ccordir	ng to SXM	1D.	
		The s	hift operation	is equivalent to	o the signed	shift ir	struction		
				n and CARRY s		•			
		🗋 Roun	dina is perfor	med according	to RDM.				

## Compatibility with C54x devices (C54CM = 1)

	C54CM = 1, th	uction is executed with $M40 = 0$ , compatibility is ensured. When e rounding is performed without clearing the 16 lowest bits of ion operation has no overflow detection, report, and saturation g operation.
Status Bits	Affected by	C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy, CARRY
Repeat	This instruction	a can be repeated.

### Repeat

## Example

CARRY

FRCT

Syntax		Description	Description					
lms(*AR0,	*AR1, AC0, AC1)	added to the of addressed by a	dressed by AR0 multiplied by the content addressed by AR1 is content of AC1 and the result is stored in AC1. The content AR0 shifted left by 16 bits is added to the content of AC0. The ed and stored in AC0.					
Before		After						
AC0	00 1111 2222	AC0	00 2111 0000					
AC1	00 1000 0000	AC1	00 1200 0000					
*AR0	1000	*AR0	1000					
*AR1	2000	*AR1	2000					
ACOV0	0	ACOV0	0					
ACOV1	0	ACOV1	0					

0

0

0

0

CARRY

FRCT

# Linear Addressing Qualifier

				Devellel					
No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	linear()			No	1	1	AD		
Opcod	е					100	01 1100		
Operar	nds	none							
Descri	ption	instruction mal addressing. T types of instru	This instruction is an instruction qualifier that can be paralleled only with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. This instruction cannot be executed in parallel with any other types of instructions and it cannot be executed as a stand-alone instruction (assembler generates an error message).						
		pointer registe	truction is used in pa rs used in the indirect er bits 0 to 8 were cle	t addressing m					
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	This instruction	n can be repeated.						

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(Smem << Tx)	No	3	1	Х
[2]	ACx = <b>low_byte(</b> Smem <b>) &lt;&lt; #</b> SHIFTW	No	3	1	х
[3]	ACx = high_byte(Smem) << #SHIFTW	No	3	1	х
[4]	ACx = Smem << #16	No	2	1	х
[5]	ACx = uns(Smem)	No	3	1	х
[6]	ACx = <mark>uns(</mark> Smem) << #SHIFTW	No	4	1	х
[7]	ACx = M40(dbl(Lmem))	No	3	1	х
[8]	LO(ACx) = Xmem, HI(ACx) = Ymem	No	3	1	х

**Description** This instruction loads a 16-bit signed constant, K16, the content of a memory (Smem) location, the content of a data memory operand (Lmem), or the content of dual data memory operands (Xmem and Ymem) to a selected accumulator (ACx).

- Status Bits Affected by C54CM, M40, RDM, SATD, SXMD
  - Affects ACOVx

#### **See Also** See the following other related instructions:

- Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
- Load Accumulator Pair from Memory
- Load Accumulator with Immediate Value
- Load Accumulator, Auxiliary, or Temporary Register from Memory
- Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
- Load Auxiliary or Temporary Register Pair from Memory
- Multiply and Accumulate with Parallel Load Accumulator from Memory
- Multiply and Subtract with Parallel Load Accumulator from Memory

No. Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1] ACx = rnd(Sn	nem << Tx)		No	3	1	х
Opcode		1101	1101 AA	AA AA	AI X%I	DD ssll
Operands	ACx, Smem, T	x				
Description		This instruction loads the content of a memory (Smem) location shifted by t content of Tx to the accumulator (ACx):				
	The input of	operand is sign extende	ed to 40 bits	accord	ling to SX	MD.
		pperand is shifted by the is equivalent to the sign				er. The shift
	_ v	is performed in the D- d keyword is applied to			•	DM, if the
	Compatibility with C54x devices (C54CM = 1)					
	C54CM = 1, n shifting operati The 6 LSBs of	uction is executed with N o overflow detection, re ion. The 6 LSBs of Tx a Tx define a shift quanti o –17, a modulo 16 op 1.	eport, and sa re used to d ty within –32	aturatio etermin to +31	on is don ne the shi I. When t	e after the ift quantity. he value is
Status Bits	Affected by	C54CM, M40, RDM, S	SATD, SXMI	C		
	Affects	ACOVx				
Repeat	This instruction	n can be repeated.				
Example						

Syntax	Description
AC0 = *AR3 << T0	AC0 is loaded with the content addressed by AR3 shifted by the content of T0.

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = low_b	<b>yte(</b> Smem) << #SH	IFTW	No	3	1	Х
Opcod	e		1110	0001 AAA	AA AA	AI DDS	H IFTW
Operar	nds	ACx, SHIFTW, Smem					
Descri	ption		This instruction loads the low-byte content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):				
		The content to SXMD.	nt of the memory locatic	on is sign ext	ended	to 40 bits	according
The input operand is shifted by the 6-bit value in the D-to operation is equivalent to the signed shift instruction.							
		In this instruction, Smem cannot reference to a memory-mapped (MMR). This instruction cannot access a byte within an MMR. If s an MMR, the DSP sends a hardware bus-error interrupt (BE request to the CPU.				If Smem is	
		Compatibility	with C54x devices (C	54CM = 1)			
			ruction is executed with I o overflow detection, re ion.		•	•	
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD			
		Affects	ACOVx				
Repeat	t	This instruction	n can be repeated.				

Syntax	Description
AC0 = low_byte(*AR3) << #31	The low-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACx = high_	<b>byte(</b> Smem <b>) &lt;&lt; #</b> S	HIFTW	No	3	1	Х
Opcod	e		1110	0010 AAA	AA AA	AI DDS	SH IFTW
Opera	nds	ACx, SHIFTW	/, Smem				
Descri	ption	This instruction loads the high-byte content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, to the accumulator (ACx):					
		The content to SXMD.	nt of the memory locatic	on is sign ext	ended	to 40 bits	according
	The input operand is shifted by the 6-bit value in the D-unit shifter. Th operation is equivalent to the signed shift instruction.				er. The shift		
	In this instruction, Smem cannot reference to a memory-mapped (MMR). This instruction cannot access a byte within an MMR. If S an MMR, the DSP sends a hardware bus-error interrupt (BEI request to the CPU.					If Smem is	
		Compatibility	v with C54x devices (C	54CM = 1)			
			ruction is executed with I to overflow detection, re tion.		•	•	
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD			
		Affects	ACOVx				
Repea	t	This instructio	n can be repeated.				
<b>E</b> vere							

Syntax	Description
AC0 = high_byte(*AR3) << #31	The high-byte content addressed by AR3 is shifted left by 31 bits and loaded into AC0.

### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ACx = Smem	<< #16		No	2	1	Х
Opcod	e			101	11 00	DD AAA	IAAA A
Opera	nds	ACx, Smem					
Descri	ption		n loads the content of a accumulator (ACx):	a memory (Sm	iem) lo	cation sh	ifted left by
		The input	operand is sign extend	ded to 40 bits	accord	ling to SX	(MD.
		The shift of	operation is equivalent	to the signed	shift ir	struction	
		The input	operand is shifted left	by 16 bits acc	cording	to M40.	
		Compatibility	y with C54x devices (	(C54CM = 1)			
			ruction is executed with verflow detection, repo		•	•	
Status	Bits	Affected by	C54CM, M40, SATE	D, SXMD			
		Affects	ACOVx				
Repea	t	This instructio	n can be repeated.				
-							

Example				
Syntax		Dese	cription	
AC1 = *A	R3+ << #16		content addre mented by 1.	essed by AR3 shifted left by 16 bits is loaded into AC1. AR3 is
Before			After	
AC1	00 0200	FC00	AC1	00 3400 0000
AR3		0200	AR3	0201
200		3400	200	3400

### Syntax Characteristics

No. Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[5] ACx = uns(Sm	em)		No	3	1	Х	
Opcode		1101	1111 AAA	AA AA	AI	DD 010u	
Operands	ACx, Smem						
Description	This instructio accumulator (A	n loads the content o ACx):	of a memory	v (Sme	em) locat	ion to the	
	The memory	ory operand is extended	to 40 bits a	ccordir	ng to uns		
		ptional uns keyword is a memory location is zero	• •	•	•	the content	
	If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.						
		peration in the accumulanit ALU, the D-unit shift			•	dependent	
	Compatibility	with C54x devices (C	54CM = 1)				
	When this inst	ruction is executed with	M40 = 0, cc	mpatik	oility is er	nsured.	
Status Bits	Affected by	SXMD					
	Affects	none					
Repeat	This instruction	n can be repeated.					
Example							

Syntax	Description
AC0 = uns(*AR3)	The content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

# Syntax Characteristics

No. Synta	x	Parallel Enable Bit	Size	Cycles	Pipeline			
[6] ACx =	uns(Smem) << #SHIFTW	No	4	1	Х			
Opcode	1111 1001   AAAA	A AAAI uxS	SH IF	TW XXI	DD 10xx			
Operands	ACx, SHIFTW, Smem							
Description	This instruction loads the content of a 6-bit value, SHIFTW, to the accumu	• •	em) loo	cation, sh	ifted by the			
	The memory operand is extended	ed to 40 bits a	ccordir	ng to uns				
	If the optional uns keyword is of the memory location is ze	••	•	•	the content			
	If the optional uns keyword content of the memory location SXMD.	• •		• •				
	,	The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.						
	Compatibility with C54x devices (	Compatibility with C54x devices (C54CM = 1)						
	When this instruction is executed with C54CM = 1, no overflow detection, shifting operation.		•	•				
Status Bits	Affected by C54CM, M40, SATE	, SXMD						
	Affects ACOVx							
Repeat	ing mode to access the memory oper	This instruction cannot be repeated when using the *(#k23) absolute address ing mode to access the memory operand (Smem); when using other address ing modes, this instruction can be repeated.						

Syntax	Description
. ,	The content addressed by AR3 is zero extended to 40 bits, shifted left by 31 bits, and loaded into AC0.

	0					Parallel	0	0	D:
No.	Syntax					Enable Bit	Size	Cycles	Pipeline
[7]	ACx = M40(dbl	(Lmer	n <b>))</b>			No	3	1	Х
Opcode	e				1110	1101 AAA	A AA	AI	D 100g
Operar	nds	AC>	k, Lmem						
Description		This instruction loads the content of data memory operand (Lmem) to the accumulator (ACx):							
			The input	operand is sigr	n extende	ed to 40 bits	accord	ling to SX	MD.
				peration in the nit ALU, the D-				•	dependent
		_	Status bit I the input c	M40 is locally s operand.	et to 1, if	the optional	M40 k	eyword is	applied to
		Cor	npatibility	with C54x de	vices (C	54CM = 1)			
		Wh	en this inst	ruction is exec	uted with	M40 = 0, co	mpatik	oility is er	sured.
Status	Bits	Affe	cted by	M40, SATD,	SXMD				
		Affe	ects	ACOVx					
Repeat	:	This	s instructio	n can be repea	ited.				
Examp	le								
Syntax		De	scription						
AC0 = 0	dbl(*AR3–)	Bee		ong word) addres Instruction is a lor Ition.					

### Syntax Characteristics

						Parallel	0.		
No.	Syntax					Enable Bit	Size	Cycles	Pipeline
[8]	LO(ACx) = Xme HI(ACx) = Ymer					No	3	1	Х
Opcod	e				1000	0001 XXX	KM MM	IYY YMM	IM 10DD
Operar	nds	AC	x, Xmem, Y	mem					
Description		This instruction performs a dual 16-bit load of accumulator high and low parts. The operation is executed in dual 16-bit mode; however, it is independent of the 40-bit D-unit ALU. The 16 lower bits of the accumulator are separated from the higher 24 bits and the 8 guard bits are attached to the higher 16-bit datapath.							
		☐ The data memory operand Xmem is loaded as a 16-bit operand to the destination accumulator (ACx) low part. And, according to SXMD the data memory operand Ymem is sign extended to 24 bits and is loaded to the destination accumulator (ACx) high part.							
		For the load operations in higher accumulator bits, overflow detection is performed at bit position 31. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.							
		If SATD is 1 when an overflow is detected on the higher data path saturation is performed with saturation value of 00 7FFFh.					ata path, a		
		Compatibility with C54x devices (C54CM = 1)							
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1, this instruction is executed as if SATD was locally cleared to 0.							
Status	Bits	Affe	ected by	C54CM, M40,	SATD,	SXMD			
		Affe	ects	ACOVx					
Repeat	t	Thi	s instruction	can be repeate	ed.				
Evom									

Syntax	Description
LO(AC0) = *AR3, HI(AC0) = *AR4	The content at the location addressed by AR4, sign extended to 24 bits, is loaded into AC0(39–16) and the content at the location addressed by AR3 is loaded into AC0(15–0).

Load Accumulator from Memory with Parallel Store Accumulator Content to Memory

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	ACy = Xmem Ymem = <b>HI(</b> A0			No	4	1	Х		
Opcod	e		1000 0111 XXXM	MMYY YMI	MM SS	SDD   110	x xxxx		
Operar	nds	ACx, ACy, T2,	Xmem, Ymem						
Descri	ption	This instruction	n performs two operatio	ons in paralle	el: load	and store			
		The first operation loads the content of data memory operand Xmem shifted left by 16 bits to the accumulator ACy.							
		The input operand is sign extended to 40 bits according to SXMD.							
		The shift operation is equivalent to the signed shift instruction.							
		The input operand is shifted left by 16 bits according to M40.							
		stores ACx(31	peration shifts the accu –16) to data memory o 32 to +31, the shift is s in this value.	perand Yme	m. If th	ie 16-bit v	alue in T2		
		The input operand is shifted in the D-unit shifter according to SXMD.							
		After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.							
		Compatibility with C54x devices (C54CM = 1)							
		When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within $-32$ to $+31$ . When the 16-bit value in T2 is between $-32$ to $-17$ , a modulo 16 operation transforms the shift quantity to within $-16$ to $-1$ .							
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD					
		Affects	ACOVy						
Repeat	t	This instruction	n can be repeated.						
SPRU3	75G			Instruction	Set Des	scriptions	5-185		

#### **See Also** See the following other related instructions:

- Load Accumulator from Memory
- Load Accumulator Pair from Memory
- Load Accumulator with Immediate Value
- Load Accumulator, Auxiliary, or Temporary Register from Memory
- Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

Syntax	Description
AC0 = *AR3 << #16, *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 shifted left by 16 bits is stored in AC0. The content of AC1 is shifted by the content of T2, and
	AC1(31–16) is stored at the address of AR4.

				Parallel				
No.	Syntax			Enable Bit	Size	Cycles	Pipeline	
[1]	pair(HI(ACx)) =	= Lmem		No	3	1	Х	
[2]	pair(LO(ACx))	= Lmem		No	3	1	Х	
Description			n loads the content of a nulator pair, ACx and A		ry ope	rand (Lm	em) to the	
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD				
		Affects	ACOVx, ACOV(x + 1	)				
See Als	50	See the following other related instructions:						
		Load Accumulator from Memory						
		Load Accumulator from Memory with Parallel Store Accumulator Content to Memory						
		Load Accumulator with Immediate Value						
		Load Accumulator, Auxiliary, or Temporary Register from Memory						
		Load Accumulator, Auxiliary, or Temporary Register with Immediate Value						
		Load Auxiliary or Temporary Register Pair from Memory						
		Multiply and Accumulate with Parallel Load Accumulator from Memor				Memory		
		Multiply an	d Subtract with Paralle	I Load Accur	nulator	from Me	emory	

## Syntax Characteristics

				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[1]	pair(HI(ACx)) =	- Lmem		No	3	1	Х
Opcode	9		1110	1101 AAA	AA AA	AI XXD	D 101x
Operar	nds	ACx, Lmem					
Description		<ul> <li>the 16 highest data memory of</li> <li>The load op of the D-un</li> <li>Valid accur</li> </ul>	loads the 16 highest b bits of the accumulator perand (Lmem) to the 1 peration in the accumula it ALU, the D-unit shift nulators are AC0 and A with C54x devices (C	r (ACx) and I 6 highest bits ator uses a de er, and the D AC2.	oads t s of acc edicate	he 16 low cumulator ed path inc	vest bits of $AC(x + 1)$ :
<b>Compatibility with C54x devices (C54CM = 1)</b> When this instruction is executed with $M40 = 0$ , compatibility is ensu C54CM = 1, overflow detection, report, and saturation is done operation.					•		
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD			
		Affects	ACOVx, ACOV(x + 1)	)			
<b>Repeat</b> This instruction can be repeated.							

Syntax	Description
pair(HI(AC2)) = *AR3+	The 16 highest bits of the content at the location addressed by AR3 are loaded into AC2(31–16) and the 16 lowest bits of the content at the location addressed by AR3 + 1 are loaded into AC3(31–16). AR3 is incremented by 1.

Before		After
AC2	00 0200 FC00	AC2 00 3400 0000
AC3	00 0000 0000	AC3 00 0FD3 0000
AR3	0200	AR3 0201
200	3400	200 3400
201	0FD3	201 0FD3

				Parallel					
No.	Syntax			Enable Bit	Size	Cycles	Pipeline		
[2]	pair(LO(ACx))	= Lmem		No	3	1	Х		
Opcod	e		1110	1101 AAA	AA AA	AI XXD	D 110x		
Operar	nds	ACx, Lmem							
Description		This instruction loads the 16 highest bits of data memory operand (Lmem) to the 16 lowest bits of the accumulator (ACx) and loads the 16 lowest bits of data memory operand (Lmem) to the 16 lowest bits of accumulator $AC(x + 1)$ :							
		_	The load operation in the accumulator uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.						
		Valid accu	mulators are AC0 and A	AC2.					
		Compatibility	with C54x devices (C	54CM = 1)					
		When this inst	ruction is executed with	M40 = 0, co	mpatik	oility is en	sured		
Status	Bits	Affected by	M40, SXMD						
		Affects	none						
Repeat	t	This instructior	n can be repeated.						
Examp	le								

Syntax	Description
	The 16 highest bits of the content at the location addressed by AR3 are loaded into $AC0(15-0)$ and the 16 lowest bits of the content at the location addressed by AR3 + 1 are loaded into $AC1(15-0)$ .

### Load Accumulator with Immediate Value

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACx = K16 << 4	#16		No	4	1	Х	
[2]	ACx = K16 << #	#SHFT		No	4	1	х	
Description		This instructio (ACx).	n loads a 16-bit signed	constant, K16	, to a se	elected ad	ccumulator	
Status	Bits	Affected by	C54CM, M40, SATE	D, SXMD				
		Affects	ACOVx					
See Al	so	See the following other related instructions:						
		Load Accumulator from Memory						
		Load Accumulator from Memory with Parallel Store Accumulator Content to Memory						
		Load Accumulator Pair from Memory						
		Load Accumulator, Auxiliary, or Temporary Register from Memory						
		Load Accumulator, Auxiliary, or Temporary Register with Immediate Value						
		Load Auxiliary or Temporary Register Pair from Memory						
		Multiply and Accumulate with Parallel Load Accumulator from Memory						
		Multiply a	nd Subtract with Parall	lel Load Accur	nulato	r from Me	emory	

#### Load Accumulator with Immediate Value

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACx = K16 <	< #16			No	4	1	Х	
Opcod	e		0111	1010 кккк	KKKK KKF	к кк	KK XXI	D 101x	
Operar	nds	ACx, K16							
Descrij	ption	This instructio the accumulat		e 16-bit signed	l constant, K	16, shi	fted left b	y 16 bits to	
		The 16-bit	t constant	, K16, is sign e	xtended to 4	0 bits a	according	to SXMD.	
		The shift of	operation	is equivalent to	o the signed	shift ir	struction		
		The input	operand i	s shifted left b	y 16 bits acc	ording	to M40.		
		Compatibility with C54x devices (C54CM = 1)							
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1, overflow detection, report, and saturation is done after the shifting operation.							
Status	Bits	Affected by	C54CN	I, M40, SATD,	SXMD				
		Affects	ACOVx						
Repeat	t	This instructio	on can be	repeated.					
Examp	le								
Syntax	(	Description							
AC0 = 5	#–2 << #16	AC0 is loaded	l with the sig	gned 16-bit valu	e (–2) shifted	left by	16 bits.		

### Load Accumulator with Immediate Value

No. Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[2] ACx = K16 << #	#SHFT			No	4	1	Х	
Opcode		0111	0101 кккк	KKKK KKK	кк кк	KK XXD	D SHFT	
Operands	ACx, K16, SHI	FT						
Description			he 16-bit signed cumulator (ACx		16, sh	ifted left k	by the 4-bit	
	The 16-bit	constan	t, K16, is sign e	extended to 4	0 bits a	according	to SXMD.	
	The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is equivalent to the signed shift instruction.							
	Compatibility with C54x devices (C54CM = 1)							
		o overflo	executed with I ow detection, re	-	•			
Status Bits	Affected by	C54C	M, M40, SXMD					
	Affects	none						
Repeat	This instruction	n can be	repeated.					
Example								
Syntax	Description							

Syntax	Description
AC0 = #-2 << #15	AC0 is loaded with the signed 16-bit value (-2) shifted left by 15 bits.

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	dst = Smem		No	2	1	X			
[2]	dst = uns(high	_ <b>byte(</b> Smem))	No	3	1	х			
[3]	dst = uns(low_		No	3	1	х			
Descri	ption	This instruction loads the content of destination (dst) register.	f a memory (Sm	iem) lo	cation to	a selected			
Status	Bits	Affected by M40, SXMD							
		Affects none							
See Al	SO	See the following other related inst	ructions:						
		Load Accumulator from Memor	у						
		Load Accumulator from Memor to Memory	umulator from Memory with Parallel Store Accumulator Content						
		Load Accumulator Pair from M	emory						
		Load Accumulator with Immedia	ate Value						
		Load Accumulator, Auxiliary, or	Temporary Reg	ister w	ith Imme	diate Value			
		Load Auxiliary or Temporary Research	egister Pair fror	n Mem	ory				
		Multiply and Accumulate with F	Parallel Load Ad	cumul	ator from	Memory			
		Multiply and Subtract with Para	Illel Load Accur	nulatoi	r from Me	emory			
		Store Accumulator, Auxiliary, o	r Temporary Re	gister	Content	to Memory			

### **Syntax Characteristics**

No. Syntax	Parallel Enable Bit Size Cycles Pip	eline
[1] dst = Smem	No 2 1	Х
Opcode	1010 FDDD AAAA A	AAI
Operands	dst, Smem	
Description	This instruction loads the content of a memory (Smem) location to destination (dst) register.	to the
	When the destination register is an accumulator:	
	The content of the memory location is sign extended to 4 according to SXMD.	0 bits
	The load operation in the destination register uses a dedicated independent of the D-unit ALU, the D-unit shifter, and the MACs.	
	When the destination register is an auxiliary or temporary register:	
	The content of the memory location is sign extended to 16 bits	6.
	The load operation in the destination register uses a dedicated independent of the A-unit ALU.	d path
	Compatibility with C54x devices (C54CM = 1)	
	When this instruction is executed with M40 = 0, compatibility is ensure	d.
Status Bits	Affected by M40, SXMD	
	Affects none	
Repeat	This instruction can be repeated.	
Evenue		

Syntax		Description							
AR1 = *AR3+	÷	AR1 is loaded wi	AR1 is loaded with the content addressed by AR3. AR3 is incremented by 1.						
Before		After							
AR1	FC00	AR1	3400						
AR3	0200	AR3	0201						
200	3400	200	3400						

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline		
[2]	dst = <mark>uns(high</mark> _	_byte	(Smerr	ו <mark>))</mark>		No	3	1	Х		
Opcod	e				1101	1111 AAA	AA AA	AI FDI	D 000u		
Operar	nds	dst,	, Smer	m	·			•			
Descri	ption	This instruction loads the high-byte content of a memory (Smem) location to the destination (dst) register.									
			Wher	n the destination	n register is a	n accumulat	or:				
			<b>T</b>	he memory ope	emory operand is extended to 40 bits according to uns.						
			•	If the optional uns keyword is applied to the input operand, content of the memory location is zero extended to 40 bits.							
		<ul> <li>If the optional uns keyword is not applied to the input o content of the memory location is sign extended according to SXMD.</li> </ul>									
			ir	The load operatindependent of IACs.		-			•		
			Wher	n the destination	n register is a	n auxiliary o	r temp	orary reg	ister:		
			<b>T</b>	he memory ope	erand is exter	nded to 16 bi	ts acco	ording to	uns.		
			•		al uns keywo e memory loo						
			•		II uns keyword he memory f SXMD.						
				The load operatindependent of t		-	ster us	es a dedi	cated pat		
			(MMF an M	s instruction, Sn R). This instruct IMR, the DSP est to the CPU.	ion cannot ac	cess a byte v	within a	an MMR.	If Smem i		
		Со	mpati	bility with C54	x devices (C	54CM = 1)					
		\//h	en thia	s instruction is e	executed with	M40 - 0 cc	mnatik	nility is pr	sured		

#### **Syntax Characteristics**

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	M40, SXMD		
	Affects	none		

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = uns(high_byte(*AR3))	The high-byte content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline		
[3]	dst = uns(low_	_byte(	Smem <b>)</b>	)		No	3	1	Х		
Opcod	e				1101	1111 AAA	AA AA	AI FDI	DD 001u		
Operai	nds	dst,	, Smer	n							
Descri	ption			uction loads the low n (dst) register.	-byte cont	tent of a mem	iory (Si	mem) loc	ation to th		
			Wher	the destination re	gister is a	n accumulat	or:				
			<b>T</b>	he memory operar	id is exter	nded to 40 bi	ts acco	ording to	uns.		
			•	•	If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 40 bits.						
			•	If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 40 bits according to SXMD.							
			ir	he load operation i dependent of the IACs.		-					
			Wher	the destination re	gister is a	n auxiliary o	r temp	orary reg	ister:		
			<b>T</b>	he memory operar	id is exter	nded to 16 bi	ts acco	ording to	uns.		
			•	If the optional u content of the m	-						
			•	If the optional un content of the regardless of S	memory						
				he load operation independent of the A		-	ster us	es a dedi	cated pat		
			(MMF an M	s instruction, Smem R). This instruction MR, the DSP ser est to the CPU.	cannot ac	cess a byte v	vithin a	an MMR.	If Smem		
		Со	mpati	bility with C54x de	evices (C	54CM = 1)					
		\ <b>\</b> /h	on thic	s instruction is exer	ruted with	M40 = 0.00	mnatik	nility is pr	sured		

### When this instruction is executed with M40 = 0, compatibility is ensured.

#### Load Accumulator, Auxiliary, or Temporary Register from Memory

Status Bits	Affected by	M40, SXMD		
	Affects	none		

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = uns(low_byte(*AR3))	The low-byte content addressed by AR3 is zero extended to 40 bits and loaded into AC0.

Load Accumulator, Auxiliary, or Temporary Register with Immediate

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	dst = k4	Yes	2	1	Х
[2]	dst = -k4	Yes	2	1	х
[3]	dst = K16	No	4	1	Х

Description	This instruction loads a 4-bit unsigned constant, k4; the 2s complement epresentation of the 4-bit unsigned constant; or a 16-bit signed constant, K16, o a selected destination (dst) register.					
Status Bits	Affected by M40, SXMD					
	Affects none					
See Also	See the following other related instructions:					
	Load Accumulator from Memory					
	Load Accumulator from Memory with Parallel Store Accumulator Content to Memory					
	Load Accumulator Pair from Memory					
	Load Accumulator with Immediate Value					
	Load Accumulator, Auxiliary, or Temporary Register from Memory					
	Load Auxiliary or Temporary Register Pair from Memory					
	Multiply and Accumulate with Parallel Load Accumulator from Memory					
	Multiply and Subtract with Parallel Load Accumulator from Memory					

#### Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

No. Syntax	Parallel Enable Bit Size Cycles Pipeline
[1] dst = k4	Yes 2 1 X
Opcode	0011 110E kkkk FDDD
Operands	dst, k4
Description	This instruction loads the 4-bit unsigned constant, k4, to the destination (dst) register.
	When the destination register is an accumulator:
	The 4-bit constant, k4, is zero extended to 40 bits.
	The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.
	When the destination register is an auxiliary or temporary register:
	The 4-bit constant, k4, is zero extended to 16 bits.
	The load operation in the destination register uses a dedicated path independent of the A-unit ALU.
	Compatibility with C54x devices (C54CM = 1)
	When this instruction is executed with $M40 = 0$ , compatibility is ensured.
Status Bits	Affected by M40
	Affects none
Repeat	This instruction can be repeated.
Example	
Syntax	Description
AC0 = #2	AC0 is loaded with the unsigned 4-bit value (2).

# Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

#### **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = -k4					Yes	2	1	Х
Opcod	le					00	11 11	lle kkł	k FDDD
Opera	nds	dst, k4							
Descri	iption				2s complei nation (dst)	ment represer register.	itation	of the 4-b	it unsigned
		🗋 When	the c	lestinatior	n register is	an accumula	tor:		
		ad	cum	ulator, and		negated in t ded to 40 bits tant.			
		The load operation in the destination register uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.							
		When the destination register is an auxiliary or temporary register:							
		The 4-bit constant, k4, is zero extended to 16 bits and negated in the I-unit before being processed by the A-unit as a signed K16 constant.							
				-	on in the de he A-unit A	estination regi LU.	ster us	es a ded	icated path
		Compatik	ility	with C54	x devices (	(C54CM = 1)			
		When this	instr	uction is e	executed wi	th M40 = 0, c	ompatil	bility is er	nsured.
Status	Bits	Affected b	/	M40					
		Affects		none					
Repea	t	This instru	ction	can be re	epeated.				
Exam	ble								

Syntax	Description
AC0 = #-2	AC0 is loaded with a 2s complement representation of the unsigned 4-bit value (2).

#### Load Accumulator, Auxiliary, or Temporary Register with Immediate Value

No. Syntax	Parallel Enable Bit Size Cycles Pipeline							
[3] dst = K16	No 4 1 X							
Opcode	0111 0110   KKKK KKKK   KKKK KKKK   FDDD 10xx							
Operands	dst, K16							
Description	This instruction loads the 16-bit signed constant, K16, to the destination (dst) register.							
	When the destination register is an accumulator, the 16-bit constant, K16, is sign extended to 40 bits according to SXMD.							
	When the destination register is an auxiliary or temporary register, the load operation in the destination register uses a dedicated path independent of the A-unit ALU.							
	Compatibility with C54x devices (C54CM = 1)							
	When this instruction is executed with M40 = 0, compatibility is ensured.							
Status Bits	Affected by M40, SXMD							
	Affects none							
Repeat	This instruction can be repeated.							
Example								
Syntax	Description							
AC1 = #248	AC1 is loaded with the signed 16-bit value (248).							

AC1 = #248		AC1 IS IO2	6-bit value (248).			
Before			After			
AC1	00 0200 FC	200	AC1	00 000	00 00F8	

# Load Auxiliary or Temporary Register Pair from Memory

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	<b>pair(</b> TAx <b>)</b> = Lm	em	No	3	1	Х			
Opcod	e	1110	1101 AAA	AA AA	AI   FDD	D 111x			
Opera	nds	Lmem, TAx							
Descri	ption	This instruction loads the 16 highest bits of data memory operand (Lmem) to the temporary or auxiliary register (TAx) and loads the 16 lowest bits of data memory operand (Lmem) to temporary or auxiliary register TA( $x + 1$ ):							
		The load operation in the temporary or auxiliary register uses a dedicated path independent of the A-unit ALU.							
		Valid auxiliary registers are AR0, AR2, AR4, and AR6.							
		Valid temporary registers are T0 a	egisters are T0 and T2.						
		Compatibility with C54x devices (C	54CM = 1)						
		When this instruction is executed with	M40 = 0, cc	mpatik	oility is en	sured.			
Status	Bits	Affected by M40							
		Affects none							
Repea	t	This instruction can be repeated.							
See Al	so	See the following other related instruct	tions:						
		Load Accumulator, Auxiliary, or Te	emporary Re	gister	from Men	nory			
		Load Accumulator, Auxiliary, or Ter	mporary Reg	jister w	ith Immed	diate Value			
		Modify Auxiliary or Temporary Reg	gister Conte	nt					

Syntax	Description
pair(T0) = *AR2	The 16 highest bits of the content at the location addressed by AR2 are loaded into T0 and the 16 lowest bits of the content at the location addressed by AR2 + 1 are loaded into T1.

# Load CPU Register from Memory

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BK03 = Smem	No	3	1	X
[2]	BK47 = Smem	No	3	1	x
	BKC = Smem				x
[3]		No	3	1	
[4]	BSA01 = Smem	No	3	1	Х
[5]	BSA23 = Smem	No	3	1	Х
[6]	BSA45 = Smem	No	3	1	Х
[7]	BSA67 = Smem	No	3	1	Х
[8]	BSAC = Smem	No	3	1	х
[9]	BRC0 = Smem	No	3	1	х
[10]	BRC1 = Smem	No	3	1	х
[11]	CDP = Smem	No	3	1	х
[12]	CSR = Smem	No	3	1	х
[13]	DP = Smem	No	3	1	х
[14]	<b>DPH</b> = Smem	No	3	1	х
[15]	PDP = Smem	No	3	1	х
[16]	SP = Smem	No	3	1	х
[17]	SSP = Smem	No	3	1	х
[18]	TRN0 = Smem	No	3	1	х
[19]	TRN1 = Smem	No	3	1	х
[20]	RETA = dbl(Lmem)	No	3	5	х

#### Opcode

See Table 5–1 (page 5-206).

Operands Lmem, Smem

Description	Instructions [1] through [19] load the content of a memory (Smem) location to the destination CPU register. This instruction uses a dedicated datapath independent of the A-unit ALU and the D-unit operators to perform the operation. The content of the memory location is zero extended to the bitwidth of the destination CPU register.
	The operation is performed in the execute phase of the pipeline. There is a 3-cycle latency between PDP, DP, SP, SSP, CDP, BSAx, BKx, BRCx, and CSR loads and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.
	For instruction [10], when BRC1 is loaded, the block repeat save register (BRS1) is also loaded with the same value.
	Instruction [20] loads the content of data memory operand (Lmem) to the 24-bit RETA register (the return address of the calling subroutine) and to the 8-bit CFCT register (active control flow execution context flags of the calling subroutine):
	The 16 highest bits of Lmem are loaded into the CFCT register and into the 8 highest bits of the RETA register.
	The 16 lowest bits of Lmem are loaded into the 16 lowest bits of the RETA register.
	When instruction [20] is decoded, the CPU pipeline is flushed and the instruction is executed in 5 cycles, regardless of the instruction context.
Status Bits	Affected by none
	Affects none
Repeat	Instructions [13] and [20] cannot be repeated; all other instructions can be repeated.
See Also	See the following other related instructions:
	Load CPU Register with Immediate Value

No.	Syntax	Opcode
[1]	<b>BK03</b> = Smem	1101 1100 AAAA AAAI 1001 xx10
[2]	<b>BK47</b> = Smem	1101 1100 AAAA AAAI 1010 xx10
[3]	BKC = Smem	1101 1100 AAAA AAAI 1011 xx10
[4]	BSA01 = Smem	1101 1100 AAAA AAAI 0010 xx10
[5]	BSA23 = Smem	1101 1100 AAAA AAAI 0011 xx10
[6]	BSA45 = Smem	1101 1100 AAAA AAAI 0100 xx10
[7]	BSA67 = Smem	1101 1100 AAAA AAAI 0101 xx10
[8]	BSAC = Smem	1101 1100 AAAA AAAI 0110 xx10
[9]	BRC0 = Smem	1101 1100 AAAA AAAI x001 xx11
[10]	BRC1 = Smem	1101 1100 AAAA AAAI x010 xx11
[11]	CDP = Smem	1101 1100 AAAA AAAI 0001 xx10
[12]	CSR = Smem	1101 1100 AAAA AAAI x000 xx11
[13]	DP = Smem	1101 1100 AAAA AAAI 0000 xx10
[14]	<b>DPH</b> = Smem	1101 1100 AAAA AAAI 1100 xx10
[15]	PDP = Smem	1101 1100 AAAA AAAI 1111 xx10
[16]	SP = Smem	1101 1100 AAAA AAAI 0111 xx10
[17]	SSP = Smem	1101 1100 AAAA AAAI 1000 xx10
[18]	TRN0 = Smem	1101 1100 AAAA AAAI x011 xx11
[19]	TRN1 = Smem	1101 1100 AAAA AAAI x100 xx11
[20]	RETA = dbl(Lmem)	1110 1101 AAAA AAAI xxxx 011x

Table 5–1. Opcodes for Load CPU Register from Memory Instruction

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<b>BK03</b> = k12	Yes	3	1	AD
[2]	<b>BK47</b> = k12	Yes	3	1	AD
[3]	<b>BKC =</b> k12	Yes	3	1	AD
[4]	<b>BRC0</b> = k12	Yes	3	1	AD
[5]	<b>BRC1</b> = k12	Yes	3	1	AD
[6]	<b>CSR</b> = k12	Yes	3	1	AD
[7]	<b>DPH</b> = k7	Yes	3	1	AD
[8]	<b>PDP</b> = k9	Yes	3	1	AD
[9]	<b>BSA01</b> = k16	No	4	1	AD
[10]	<b>BSA23</b> = k16	No	4	1	AD
[11]	<b>BSA45</b> = k16	No	4	1	AD
[12]	<b>BSA67</b> = k16	No	4	1	AD
[13]	<b>BSAC</b> = k16	No	4	1	AD
[14]	<b>CDP</b> = k16	No	4	1	AD
[15]	<b>DP</b> = k16	No	4	1	AD
[16]	<b>SP</b> = k16	No	4	1	AD
[17]	<b>SSP</b> = k16	No	4	1	AD

Load CPU Register with Immediate Value

**Syntax Characteristics** 

Opcode See Table 5–2 (page 5-208).

kх

Operands

Description

This instruction loads the unsigned constant, kx, to the destination CPU register. This instruction uses a dedicated datapath independent of the A-unit ALU and the D-unit operators to perform the operation. The constant is zero extended to the bitwidth of the destination CPU register.

For instruction [5], when BRC1 is loaded, the block repeat save register (BRS1) is also loaded with the same value.

The operation is performed in the address phase of the pipeline.

SPRU375G

Instruction Set Descriptions 5-207

Status Bits	Affected by	none
	Affects	none
Repeat	Instruction [15]	cannot be repeated; all other instructions can be repeated.
See Also	See the following	ng other related instructions:
	Load CPU	Register from Memory

Table 5–2. Opcodes for Load CPU Register with Immediate Value Instruction

No.	Syntax				Орс	ode			
[1]	<b>BK03</b> = k12		0001	011E	kkkk	kkkk	kkkk	0100	
[2]	<b>BK47</b> = k12		0001	011E	kkkk	kkkk	kkkk	0101	
[3]	<b>BKC</b> = k12		0001	011E	kkkk	kkkk	kkkk	0110	
[4]	<b>BRC0 =</b> k12		0001	011E	kkkk	kkkk	kkkk	1001	
[5]	<b>BRC1</b> = k12		0001	011E	kkkk	kkkk	kkkk	1010	
[6]	<b>CSR</b> = k12		0001	011E	kkkk	kkkk	kkkk	1000	
[7]	<b>DPH</b> = k7		0001	011E	xxxx	xkkk	kkkk	0000	
[8]	<b>PDP</b> = k9		0001	011E	xxxk	kkkk	kkkk	0011	
[9]	<b>BSA01</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	011x
[10]	<b>BSA23</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	100x
[11]	<b>BSA45</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	101x
[12]	<b>BSA67</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	110x
[13]	<b>BSAC</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	111x
[14]	<b>CDP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	010x
[15]	<b>DP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	000x
[16]	<b>SP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx1	000x
[17]	<b>SSP</b> = k16	0111	1000	kkkk	kkkk	kkkk	kkkk	xxx0	001x

# Load Extended Auxiliary Register from Memory

#### **Syntax Characteristics**

-								
No.	Suptox			Parallel Enable Bit	Size	Cueles	Dinalina	
NO.	Syntax				Size	Cycles	Pipeline	
[1]	XAdst = dbl(L	.mem)		No	3	1	Х	
Opcod	e		1110	1101 AAA	AA AA	AI XDD	DD 1111	
Operar	nds	Lmem , XAdst						
Descri	ption		n loads the lower 23 bits n) to the 23-bit destinati			-	•	
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructior	n can be repeated.					
See Al	so	See the following other related instructions:						
		Load Extended Auxiliary Register with Immediate Value						
		Modify Extended Auxiliary Register Content						
		Move Exte	nded Auxiliary Register	Content				
		Store Exte	nded Auxiliary Register	Content to I	Memor	у		

Syntax		Descr	Description						
XAR1 = dbl(*	(*AR3) The 7 lowest bits of the content at the location addressed by AR3 and the 16 bit the content at the location addressed by AR3 + 1 are loaded into XAR1.								
Before			After						
XAR1	00	0000	XAR1	12 OFD3					
AR3		0200	AR3	0200					
200		3492	200	3492					
201		0FD3	201	0FD3					

# Load Extended Auxiliary Register with Immediate Value

#### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	XAdst = k23		No	6	1	AD			
Opcod	e	1110	1100 AAA	AA AA	AI ODD	DD 1110			
Operar	nds	k23, XAdst							
Descri	ption	This instruction loads a 23-bit unsigned constant (k23) into the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP). This operation is completed in the address phase of the pipeline by the A-unit address generator. Data memory is not accessed.							
The premodification or postmodification of the auxiliary regis of *port(#K), and the use of the readport() or writeport supported for this instruction. The use of auxiliary register o supported. If the corresponding bit (ARnLC) in status regis to 1, the circular buffer management also controls the resu						ifier is not perations is 2_55 is set			
Status	Bits	Affected by ST2_55							
		Affects none							
Repeat	t	This instruction can be repeated.							
See Al	SO	See the following other related instruc	ctions:						
		Load Extended Auxiliary Register from Memory							
		Modify Extended Auxiliary Register Content							
		Move Extended Auxiliary Register Content							
		Store Extended Auxiliary Register Content to Memory							

Syntax	Description
XAR0 = #7FFFFFh	The 23-bit value (7FFFFFh) is loaded into XAR0.

# Load Memory with Immediate Value

# Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = K8					No	3	1	Х
[2]	Smem = K16					No	4	1	Х
Opcod	e	K8			1110	0110 AA	AA AA	ААІ ККК	к кккк
		К16	1111	1011	AAAA	AAAI KK	кк кі	ккк   ккк	к кккк
Operar	nds	Kx, Smem							
Descrij	ption	These instructions initialize a data memory location. These instructions store an 8-bit signed constant, K8, or a 16-bit signed constant, K16, to a memory (Smem) location. They use a dedicated datapath to perform the operation. For instruction [1], the immediate value is always signed extended to 16 bits before being stored in memory.							
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	Instruction [1] can be repeated. Instruction [2] cannot be repeated when using the *(#k23) absolute addressing mode to access the memory operand (Smem); when using other addressing modes, this instruction can be repeated.							y operand
See Al	SO	See the follow	ing othe	r related	l instruc	ctions:			
		Move Men	nory to N	/lemory					

-							
Syntax		Description					
*(#0501h) =	#248	The signed 16-bi	The signed 16-bit value (248) is loaded to address 501h.				
Before		After					
0501	FC00	0501	F800				

# Memory Delay

# Syntax Characteristics

_									
No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	delay(Smem)			No	2	1	Х		
Opcod	е			10	11 03	110 AAA	IAAA AA		
Operar	nds	Smem							
Descri	ption	This instruction copies the content of the memory (Smem) location into the next higher address (Smem + 1). When the data is copied, the content of the addressed location remains the same. A dedicated datapath is used to make this memory move.							
		When this instruction is executed, the two address register arithmetic units ARAU X and Y, of the A-unit data address generator unit, are used to compute the two addresses Smem and Smem + 1. The address generation is not affected by circular addressing; if Smem points to the end of a circular buffer, Smem + 1 will point to an address outside the circular buffer.							
		The soft dual memory addressing mode mechanism cannot be applied to this instruction. This instruction cannot use the *port(#k16) addressing mode or be paralleled with the readport() or writeport() operand qualifier.							
		This instruction cannot be used for accesses to I/O space. Any illegal ac to I/O space generates a hardware bus-error interrupt (BERRINT) t handled by the CPU.							
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	This instruction	can be repeated.						

Syntax		Description	
delay(*AR1+)		The content addr is incremented by	essed by AR1 is copied to the next higher address, AR1 + 1. AR1 y 1.
Before		After	
AR1	0200	AR1	0201
200	3400	200	3400
201	0D80	201	3400
202	2030	202	2030

No.	Syntax								rallel ble Bit	Size	Cycles	Pi	ipeline
[1]	mmap()							Ν	No	1	1		D
Opcod	e										100	)1	1000
Operai	nds	none									·		
Descri	ption	a Smem you to lo pointer (	or Lm ocally p SP) or ve to t	nem o prev r the	direct ent the local	memo e dma data p	ry acce acces age re	ess (di ss fron egister	ma). Tl n being (DP). I	nis ope g relati It force	y instruct rand qua ve to the s the dma page sta	lifie dat a ac	r allow a stac cess t
		This operand qualifier cannot be executed:											
	as a stand-alone instruction (assembler generates an error message)												
		in parallel with instructions not embedding an Smem or Lmem data memory operand											
		Accu [2] a Stor	umulat nd [3]	tor, A  ; Loa umul	Auxilia ad Aco lator, <i>I</i>	ry, or cumula Auxilia	Tempo ator fro	orary R om Me	egiste mory i	r from I nstruct	a registe Memory i ions [2] a Content	nstr nd	uction [3]; an
		The MMRs are mapped as 16-bit data entities between addresses 0h and 5Fh The scratch-pad memory that is mapped between addresses 60h and 7Fh o each main data pages of 64K words cannot be accessed through this mechanism.											
		Any insti user-def			-			difier c	annot l	be com	bined wit	h ar	iy othe
Status	Bits	Affected	by	no	ne								
		Affects		no	ne								
Repea	t	This inst	ructior	n cai	n be re	epeate	ed.						
Examp	le												
Syntax	-	Descrip	tion										

 $T2 = @(AC0_L))$ AC0\_L is a keyword representing AC0(15-0). The content of AC0(15-0) is copied|| mmap()into T2.

# Modify Auxiliary Register Content

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	mar(Smem)			No	2	1	AD			
Opcod	e			101	11 01	.00 AAA	A AAAI			
Opera	nds	Smem								
Descri	ption	This instruction performs, in the A-unit address generation units, the auxiliary register modification specified by Smem as if a word single data memory operand access was made. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.								
		If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management controls the result stored in the destination register.								
		Compatibility with C54x devices (C54CM = 1)								
		In the translat C54CM set to	ed code section, the m 1.	ar() instructi	on mu	st be exe	cuted with			
		When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.								
Status	Bits	Affected by	ST2_55							
		Affects	none							
Repea	t	This instructio	n can be repeated.							

#### See Also See the following other related instructions:

- Modify Auxiliary or Temporary Register Content
- Modify Auxiliary or Temporary Register Content by Addition
- Modify Auxiliary or Temporary Register Content by Subtraction
- Modify Auxiliary Register Content with Parallel Multiply
- D Modify Auxiliary Register Content with Parallel Multiply and Accumulate
- Modify Auxiliary Register Content with Parallel Multiply and Subtract
- Modify Extended Auxiliary Register Content
- Parallel Modify Auxiliary Register Contents

Syntax	Description
mar(*AR3+)	The content of AR3 is incremented by 1.

# Modify Auxiliary Register Content with Parallel Multiply

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<b>mar(</b> Xmem <b>),</b> ACx = <mark>M40(rnc</mark>	<mark>l(uns</mark> (Ymem) * <mark>uns(coef(</mark> Cmem <b>)</b> )))	No	4	1	Х
Opcode	e	1000 0010 xxx	M MMYY YM	MM 11	Lmm uux	x DDg%
Operar	nds	ACx, Cmem, Xmem, Ymem				
Description		This instruction performs two paralle register (MAR) and multiply. The op MACs.	-	-		
		The first operation performs an aux register modification is specified b Xmem.				-
		The second operation performs a moperands of the multiplier are the cextended to 17 bits, and the contraddressed using the coefficient addressed using the coe	content of data	i memo memo	ory opera ory operar	nd Ymem, nd Cmem,
		Input operands are extended to	17 bits accord	ding to	uns.	
		If the optional uns keyword is of the memory location is zero.		•	•	he content
		<ul> <li>If the optional uns keyword content of the memory location</li> <li>SXMD.</li> </ul>				
		If FRCT = 1, the output of the m	nultiplier is shif	ted left	by 1 bit.	
		Multiplication overflow detection	n depends on a	SMUL.		
		The 32-bit result of the multiplic	ation is sign e	xtende	d to 40 bi	ts.
		Rounding is performed accordination applied to the instruction.	ng to RDM, if t	he opti	ional rnd l	keyword is
		<ul> <li>Overflow detection depends o destination accumulator overflo</li> </ul>				ected, the
		When an overflow is detected, t SATD.	he accumulato	or is sa	turated ac	ccording to

	This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
	□ For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.
	Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:
	■ mar(Xmem)
	■ mar(Ymem)
	■ mar(Cmem)
Status Bits	Affected by FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects ACOVx
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Modify Auxiliary Register Content
	Modify Auxiliary Register Content with Parallel Multiply and Accumulate
	Modify Auxiliary Register Content with Parallel Multiply and Subtract
	Multiply

Syntax	Description
mar(*AR3+), AC0 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0.

Modify Auxiliary Register Content with Parallel Multiply and Accumulate

	0			Parallel	0.	0	D' I'	
No.	Syntax			Enable Bit	Size	Cycles	Pipeline	
[1]	mar(Xmem), ACx = M40(rnd	(ACx + <b>(</b> uns(Ymer	m) * uns( <b>coef(</b> Cmem <b>))</b> )))	No	4	1	Х	
[2]	mar(Xmem), ACx = M40(rnde uns(coef(Cmen	( <b>(</b> ACx >> <b>#16)</b> + (t n <b>)))</b> )	uns(Ymem) *	No	4	1	х	
<b>Description</b> These instructions perform two parallel operations in one cy auxiliary register (MAR), and multiply and accumulate (MAC). The are executed in the two D-unit MACs.						•	•	
Status	Bits	Affected by	FRCT, M40, RDM, SA	ATD, SMUL, SXMD				
		Affects	ACOVx, ACOVy					
See Al	S0	See the following other related instructions:						
		Modify Auxiliary Register Content						
		Modify Aux	kiliary Register Content	with Paralle	l Multip	bly		
		Modify Auxiliary Register Content with Parallel Multiply and Subtract					ubtract	
		Multiply and Accumulate						

#### Modify Auxiliary Register Content with Parallel Multiply and Accumulate

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<b>mar(</b> Xmem <b>),</b> ACx = M40(rn	<mark>d(</mark> ACx	+ <b>(</b> uns(Ymem) * uns(		No	4	1	Х
Opcod	e		1000	0011 XXXM	MMYY YMI	MM 11	.mm uux	x DDg%
Operar	nds	AC	x, Cmem, Xmem, Y	/mem				
Description		reg	s instruction perforr ister (MAR), and r ecuted in the two D-	multiply and ac	-	-		
		reg	e first operation per ister modification i em.					•
		D-u mei mei	e second operation init MAC. The input mory operand Yme mory operand Cme ended to 17 bits.	ut operands of em, extended t	the multiplie o 17 bits, a	er are t nd the	the conte content	ent of data of a data
			Input operands are	e extended to 1	7 bits accord	ling to	uns.	
			•	uns keyword is a location is zero		•	•	he content
			•	uns keyword is memory locatior			• •	
			If FRCT = 1, the o	utput of the mul	tiplier is shif	ted left	by 1 bit.	
			Multiplication over	flow detection d	lepends on S	SMUL.		
			The 32-bit result of to the source accu	-	on is sign exte	ended	to 40 bits	and added
			Rounding is perfor applied to the instr	-	to RDM, if t	he opti	onal rnd	keyword is
			Overflow detectio destination accum	•				ected, the

	When an overflow is detected, the accumulator is saturated according to SATD.
	This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
	For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.
whi	ch data flow can also disable the usage of the corresponding MAC unit, le allowing the modification of auxiliary registers in the three address heration units through the following instructions:
	■ mar(Xmem)
	■ mar(Ymem)

■ mar(Cmem)

 Status Bits
 Affected by
 FRCT, M40, RDM, SATD, SMUL, SXMD

 Affects
 ACOVx

 Repeat
 This instruction can be repeated.

Syntax	Description
mar(*AR3+), AC0 = AC0 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0.

#### Modify Auxiliary Register Content with Parallel Multiply and Accumulate

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(Xmem), ACx = M40(mo uns(coef(Cme	•••	x >> <b>#16)</b> + <b>(</b> uns(Ymem) *		No	4	1	Х
Opcod	e		1000 0100	XXXM	MMYY YMN	MM 01	.mm uux	x DDg%
Operar	nds	AC	x, Cmem, Xmem, Ymem					
Descrij	ption	reg	s instruction performs two p ister (MAR), and multiply ecuted in the two D-unit MA	and acc	•			• •
		reg	e first operation performs a ister modification is specif em.					
		D-u me me	e second operation perform init MAC. The input opera mory operand Ymem, exte mory operand Cmem, addr ended to 17 bits.	nds of t ended to	the multiplie o 17 bits, a	r are t nd the	the conte	ent of data of a data
			Input operands are extend	led to 17	bits accord	ling to	uns.	
			■ If the optional uns keyv of the memory location			•	•	he content
			<ul> <li>If the optional uns key content of the memory SXMD.</li> </ul>				• •	
			If FRCT = 1, the output of	the mult	tiplier is shift	ed left	by 1 bit.	
			Multiplication overflow det	ection de	epends on S	SMUL.		
			The 32-bit result of the mult to the source accumulate operation is performed w ACx(39).	or ACx	shifted right	t by 16	6 bits. Tl	he shifting
			Rounding is performed ac applied to the instruction.	cording	to RDM, if th	ne opti	onal rnd l	keyword is

Instruction Set Descriptions 5-221

		detection depends on M40. If an overflow is detected, the accumulator overflow status bit (ACOVx) is set.				
	When an o SATD.	verflow is detected, the accumulator is saturated according to				
		provides the option to locally set M40 to 1 for the execution of if the optional M40 keyword is applied to the instruction.				
	For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.					
	while allowing	v can also disable the usage of the corresponding MAC unit, the modification of auxiliary registers in the three address ts through the following instructions:				
	■ mar(Xmem)					
	■ mar(Yr	mem)				
	■ mar(Ci	mem)				
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD				
	Affects	ACOVx				
Repeat	This instruction	n can be repeated.				

Syntax						D	Description				
mar(*AR2+), AC0 = ((AC0 >> #16) + (uns(*AR1) * uns(coef(*CDP))))					) ir b a ((	Both instructions are performed in parallel. AR2 is ncremented by 1. The unsigned content addressed by AR1 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 shifted right by 16 bits and the result is stored in AC0. An overflow s detected in AC0.					
Before				After							
AC0	00	6900	0000	AC0	00	95C0	0 9200				
AC1	00	0023	0000	AC1	00	0023	3 0000				
*AR1			EF00	*AR1			EF00				
AR2			0201	AR2			0202				
*CDP			A067	*CDP			A067				
ACOV0			0	ACOV0			1				
ACOV1			0	ACOV1			0				
CARRY			0	CARRY			0				
M40			0	M40			0				
FRCT			0	FRCT			0				
SATD			0	SATD			0				

Modify Auxiliary Register Content with Parallel Multiply and Subtract

#### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	mar(Xmem), ACx = M40(rno	<mark>l(</mark> ACx	- (uns(Ymem) * uns(coef	(Cmem)))))	No	4	1	Х		
Opcod	e		1000 01	01 XXXM	MMYY YMI	MM 00	)mm uux	x DDg%		
Opera	nds	AC	x, Cmem, Xmem, Ymer	n						
Descri	ption	reg	is instruction performs tw jister (MAR), and multiply he two D-unit MACs.	•		•				
		reg	e first operation perform jister modification is sp nem.					-		
		MA ope ope	The second operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.							
			Input operands are ext	ended to 1	7 bits accord	ding to	uns.			
			If the optional uns k of the memory loca	•		•	•	he content		
			If the optional uns content of the mem SXMD.	•						
			If FRCT = 1, the output	t of the mul	tiplier is shif	ted left	by 1 bit.			
			Multiplication overflow	detection d	epends on \$	SMUL.				
		The 32-bit result of the multiplication is sign extended to 40 bits ar subtracted from the source accumulator ACx.						0 bits and		
		Rounding is performed according to RDM, if the optional rnd key applied to the instruction.								
			Overflow detection de destination accumulato	•				ected, the		

5-223

	When an SATD.	overflow is detected, the accumulator is saturated according to
		uction provides the option to locally set M40 to 1 for the execution ruction, if the optional M40 keyword is applied to the instruction.
	on some	nstruction, the Cmem operand is accessed through the BB bus; C55x-based devices, the BB bus is only connected to internal and not to external memory. To prevent the generation of a bus Cmem operand must not be mapped on external memory.
	while allowing	w can also disable the usage of the corresponding MAC unit, g the modification of auxiliary registers in the three address its through the following instructions:
	■ mar()	Kmem)
	■ mar()	(mem)
	■ mar(0	Cmem)
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx
Repeat	This instruction	on can be repeated.
See Also	See the follow	ving other related instructions:
	🗋 Modify Au	uxiliary Register Content
	🔲 Modify Au	uxiliary Register Content with Parallel Multiply
	🔲 Modify Au	uxiliary Register Content with Parallel Multiply and Accumulate
	Multiply a	nd Subtract
	/	

Syntax	Description
mar(*AR3+), AC0 = AC0 - (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. AR3 is incremented by 1. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0 and the result is stored in AC0.

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mar(TAy = TAx)			No	3	1	AD
[2]	mar(TAx = P8)			No	3	1	AD
[3]	<b>mar(</b> TAx = D16)	)		No	4	1	AD
Descrip	otion	The	ese instructions perform, in the A-u	unit address (	genera	tion units	:
			a move from auxiliary or tempora register TAy	ry register TA	x to au	xiliary or	temporary
			a load in the auxiliary or tempora defined by a program address lab	, ,			m address
			a load in the auxiliary or tempor address signed constant D16	ary registers	TAx o	f the abs	olute data
			e operation is performed in the addr mory is not accessed.	ess phase of	the pip	eline, hov	wever data
Status	Bits	Aff	ected by none				
		Aff	ects none				
See Als	50	Se	e the following other related instru-	ctions:			
			Load Auxiliary or Temporary Reg	ister from Me	mory		
			Modify Auxiliary Register Conten	t			
			Modify Auxiliary or Temporary Re	egister Conter	nt by A	ddition	
			Modify Auxiliary or Temporary Re	egister Conter	nt by S	ubtractio	n
			Modify Extended Auxiliary Regist	er Content			

#### **Syntax Characteristics**

	0				Parallel	0.	0	
No.	Syntax				Enable Bit	Size	Cycles	•
[1]	mar(TAy = TAx)				No	3	1	AD
Opcode	e		I	0001	010E FSS	S xx	xx FDD	D 0001
				0001	010E FSS	S xx	xx FDD	D 1001
		The assembler paralleled pair.	selects the opco	ode de	pending on tl	ne inst	ruction po	osition in a
Operan	nds	TAx, TAy						
Descrip	ption	the auxiliary or	performs, in the temporary regist sperformed in the accessed.	er TAx	to auxiliary	or tem	porary re	gister TAy.
Status	Bits	Affected by	none					
		Affects	none					
Repeat	:	This instruction	can be repeated	d.				

#### Example 1

Syntax	Description
mar(AR0 = AR1)	The content of AR1 is copied to AR0.

Syntax	Description
mar(T0 = T1)	The content of T1 is copied to T0.

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2]	<b>mar(</b> TAx = P8)			No	3	1	AD
Opcod	e		·	010E   PPF 010E   PPF			
		The assemble paralleled pair.	r selects the opcode de				
Operar	nds	TAx, P8					
Descri	ption	auxiliary or te program addre	n performs, in the A-unit mporary registers TAx ess label assembled into a of the pipeline; howeve	of a progra P8. The ope	am ado eration	dress del is perfor	fined by a med in the
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	n can be repeated.				
Examp	ole 1						
Syntax	(	Description					

# Example 2

mar(AR0 = #255)

Syntax	Description
mar(T0 = #255)	The unsigned 8-bit value (255) is copied to T0.

The unsigned 8-bit value (255) is copied to AR0.

#### **Syntax Characteristics**

					Parallel			
No.	Syntax				Enable Bit	Size	Cycles	Pipeline
[3]	<b>mar(</b> TAx = D16	)			No	4	1	AD
Opcode	e		0111	0111 DDDD		DD DD	DD FDD	D xxxx
Operar	nds	TAx, D16						
Descrip	ption	auxiliary or te constant D16.	emporary The ope	ns, in the A-unit registers TAx ration is perform ris not accesse	of the abso red in the add	lute da	ata addre	ess signed
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructio	n can be	repeated.				
Examp	le							

# Syntax Description mar(T1 = #FFFFh) The address FFFFh is copied to T1.

# Modify Auxiliary or Temporary Register Content by Addition

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	mar(TAy + TAx)			No	3	1	AD		
[2]	mar(TAx + P8)			No	3	1	AD		
Description		These instructions perform, in the A-unit address generation units:							
			an addition between two aux and stores the result in TAy	iliary or tempora	ry regi	sters, TA	x and TAy,		
		an addition between the auxiliary or temporary registers TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx							
		The operation is performed in the address phase of the pipeline, however data memory is not accessed.							
		If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management controls the result stored in the destination register.							
Status Bits		Affected by ST2_55							
		Aff	ects none						
See Also		Se	e the following other related in	structions:					
			Modify Auxiliary Register Cor	ntent					
			Modify Auxiliary or Temporary	y Register Conte	nt				
			Modify Auxiliary or Temporary	y Register Conte	nt by S	ubtractio	n		
			Modify Extended Auxiliary Re	egister Content					

#### Modify Auxiliary or Temporary Register Content by Addition

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
[1] <b>mar(</b> TAy + TAx)		No	3	1	AD		
Opcode	0001	010E FSS	S xx	xx FDD	D 0000		
	0001	010E FSS	S xx	xx FDD	D 1000		
	The assembler selects the opcode de paralleled pair.	pending on th	ne insti	ruction po	sition in a		
Operands	TAx, TAy						
Description	This instruction performs, in the A-unit address generation units, an addition between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy. The content of TAx is considered signed. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.						
	If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management controls the result stored in the destination register.						
	Compatibility with C54x devices (C54CM = 1)						
	In the translated code section, the mar() instruction must be executed with C54CM set to 1.						
	When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.						
Status Bits	Affected by ST2_55						
	Affects none						
Repeat	This instruction can be repeated.						

#### Example 1

Syntax	Description		
mar(AR0 + T0)	The content	of AR0 is added	to the signed content of T0 and the result is stored in AR0.
Before		After	
XAR0	01 0000	XAR0	01 8000
тО	8000	TO	8000

Syntax	Description
mar(T0 + T1)	The content of T0 is added to the content of T1 and the result is stored in T0.

#### Modify Auxiliary or Temporary Register Content by Addition

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[2]	mar(TAx + P8)			No	3	1	AD		
Opcod	e		0001	010E PPI	PP PP	PP FDD	D 0100		
			0001	010E PPP	PP PP	PP FDD	D 1100		
		The assembler paralleled pair.	selects the opcode de	epending on t	he inst	ruction p	osition in a		
Operar	nds	TAx, P8							
Description		This instruction performs, in the A-unit address generation units, an addition between the auxiliary or temporary register TAx and a program address defined by a program address label assembled into unsigned P8, and stores the result in TAx. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.							
		If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management controls the result stored in the destination register.							
		Compatibility with C54x devices (C54CM = 1)							
		In the translated code section, the mar() instruction must be executed with C54CM set to 1.							
		When circular modification is selected for the destination auxiliary register, this instruction modifies the selected destination auxiliary register by using BK03 as the circular buffer size register; BK47 is not used.							
Status	Bits	Affected by	ST2_55						
		Affects	none						
Repeat	:	This instruction	can be repeated.						
Examp	le								

Syntax	Description
mar(T0 + #255)	The unsigned 8-bit value (255) is added to the content of T0 and the result is stored in T0.

# Modify Auxiliary or Temporary Register Content by Subtraction

				Parallel				
No.	Syntax			Enable Bit	Size	Cycles	Pipeline	
[1]	mar(TAy – TAx)	)		No	3	1	AD	
[2]	mar(TAx – P8)			No	3	1	AD	
Description		These instructions perform, in the A-unit address generation units:						
			a subtraction between two auxilia and stores the result in TAy	ary or tempora	ary regi	sters, TA	y and TAx,	
			a subtraction between the auxil program address defined by a unsigned P8, and stores the rest	program add	•	-		
		The operation is performed in the address phase of the pipeline, however data memory is not accessed.						
		If the destination register is an auxiliary register and the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management controls the result stored in the destination register.						
Status Bits		Aff	Affected by ST2_55					
		Aff	ects none					
See Als	50	See the following other related instructions:						
			Modify Auxiliary Register Conter	ıt				
			Modify Auxiliary or Temporary Re	egister Conte	nt			
			Modify Auxiliary or Temporary Re	egister Conte	nt by A	ddition		
			Modify Extended Auxiliary Regis	ter Content				

# Modify Auxiliary or Temporary Register Content by Subtraction

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1] <b>mar(</b> TAy – TAx <b>)</b>		No	3	1	AD
Opcode	0001	010E FSS	SS xx	xx FDD	D 0010
	0001	010E FSS	SS xx	xx FDD	D 1010
	The assembler selects the opcode de paralleled pair.	pending on t	he inst	ruction po	osition in a
Operands	TAx, TAy				
Description	This instruction performs, in the A-unit between two auxiliary or temporary re- result in TAy. The content of TAx is performed in the address phase of the accessed.	egisters, TAy considered	/ and T signed	TAx, and d. The op	stores the peration is
	If the destination register is an auxilia (ARnLC) in status register ST2_55 is s controls the result stored in the destin	set to 1, the ci	ircular		-
	Compatibility with C54x devices (C	54CM = 1)			
	In the translated code section, the machine C54CM set to 1.	ar() instructio	on mus	st be exe	cuted with
	When circular modification is selected instruction modifies the selected destias the circular buffer size register; BK	nation auxilia	ary reg	•	-
Status Bits	Affected by ST2_55				
	Affects none				
Repeat	This instruction can be repeated.				

# Example 1

Syntax	Description				
mar(AR0 – T0)	The signed of AR0.	The signed content of T0 is subtracted from the content of AR0 and the result is stored in AR0.			
Before		After			
XAR0	01 8000	XAR0	01 0000		
Т0	8000	Т0	8000		

Syntax	Description
mar(T0 – T1)	The content of T1 is subtracted from the content of T0 and the result is stored in T0.

# Modify Auxiliary or Temporary Register Content by Subtraction

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2]	mar(TAx – P8)			No	3	1	AD
Opcod	e		0001	010E PPI	PP PP	PP FDD	D 0110
			0001	010E PPE	PP PP	PP FDD	D 1110
		The assembler paralleled pair.	selects the opcode de	pending on t	he inst	ruction po	osition in a
Operar	nds	TAx, P8					
Descri	ption	between the a defined by a pr the result in TA	performs, in the A-unit uxiliary or temporary ogram address label as Ax. The operation is p ver, data memory is no	register TAx ssembled interformed in	and a a a	a prograr gned P8, a	n address and stores
		(ARnLC) in stat	on register is an auxili tus register ST2_55 is s sult stored in the destin	set to 1, the c	ircular	•	-
		Compatibility	with C54x devices (C	54CM = 1)			
		In the translate C54CM set to	ed code section, the m	ar() instructio	on mus	st be exe	cuted with
		instruction mod	nodification is selected lifies the selected desti buffer size register; BK	nation auxilia	ary reg		-
Status	Bits	Affected by	ST2_55				
		Affects	none				
Repeat	:	This instruction	can be repeated.				
Examp	le						

Syntax	Description
mar(AR0 – #255)	The unsigned 8-bit value (255) is subtracted from the signed content of AR0 and
	the result is stored in AR0.

# Modify Data Stack Pointer

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SP = SP + K8				Yes	2	1	AD
Opcode	e				010	)0 11	1E   KKK	к кккк
Operar	nds	K8						
Descriț	<b>Description</b> This instruction performs an addition in the A-unit data-address ge unit (DAGEN) in the address phase of the pipeline. The 8-bit signed of K8, is sign extended to 16 bits and added to the data stack pointer (SF in 32-bit stack configuration, the system stack pointer (SSP) is also r Updates of the SP and SSP (depending on the stack configuration) sh be executed in parallel with this instruction.						d constant, SP). When o modified.	
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructio	n can be rep	peated.				

Syntax	Description
SP = SP + #127	The 8-bit value (127) is sign extended to 16 bits and added to the stack pointer (SP).

# Modify Extended Auxiliary Register Content

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline				
[1]	XAdst = mar(S	Smem)	No	3	1	AD				
Opcod	e	1110	1100 AAA	AA AA	AI   XDI	D 1110				
Operar	nds	Smem, XAdst								
Descri	ption	operand field and modifies the 23-bit d XDP, or XCDP). This operation is co	This instruction computes the effective address specified by the Smem operand field and modifies the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP). This operation is completed in the address phase of the pipeline by the A-unit address generator. Data memory is not accessed.							
		The premodification or postmodificatio of *port(#K), and the use of the rea supported for this instruction. The use supported. If the corresponding bit (Al to 1, the circular buffer management a	adport() or v of auxiliary r RnLC) in sta	writepc egister tus reg	ort() qual offset op gister ST2	ifier is not perations is 2_55 is set				
Status	Bits	Affected by ST2_55								
		Affects none								
Repeat	t	This instruction can be repeated.								
See Al	so	See the following other related instruc	tions:							
		Load Extended Auxiliary Register	from Memor	у						
		Load Extended Auxiliary Register	with Immedi	ate Va	lue					
		Modify Auxiliary Register Content								
		Move Extended Auxiliary Register	Content							
		Store Extended Auxiliary Register	Content to I	Memor	у					
Fxamn	he									

Syntax	Description
XAR0 = mar(*AR1)	The content of AR1 is loaded into XAR0.

# Move Accumulator Content to Auxiliary or Temporary Register

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TAx = <b>HI(</b> ACx <b>)</b>			Yes	2	1	Х
Opcod	e			010	00 01	.0E 00S	S FDDD
Operar	nds	ACx, TAx					
Descri	ption	This instruction moves the high part of the accumulator, ACx(31–16), to the destination auxiliary or temporary register (TAx). The 16-bit move operation is performed in the A-unit ALU.					
		Compatibility	with C54x devices	(C54CM = 1)			
		When this inst	ruction is executed w	vith M40 = 0, co	ompatik	oility is er	sured.
Status	Bits	Affected by	M40				
		Affects	none				
Repeat	t	This instruction	n can be repeated.				
See Al	so	See the follow	ing other related inst	ructions:			
		Move Acc	umulator, Auxiliary, o	r Temporary Re	egister	Content	
		Move Aux	iliary or Temporary R	egister Conten	t to Ac	cumulato	r
Examp	ole						
Syntax	K	Description					

eymax			2000	Decemption				
AR2 = HI(AC0)		The	content of AC0(3	31–16) is copied to AR2.				
Before				After				
AC0	01	E500	0030	AC0	01 E500 0030			
AR2			0200	AR2	E500			

# Move Accumulator, Auxiliary, or Temporary Register Content

	0	Parallel
No.	Syntax	Enable Bit Size Cycles Pipeline
[1]	dst = src	Yes 2 1 X
Opcod	le	0010 001E FSSS FDDD
Opera	nds	dst, src
Descri	ption	This instruction moves the content of the source (src) register to the destination (dst) register:
		When the destination (dst) register is an accumulator:
		The 40-bit move operation is performed in the D-unit ALU.
		During the 40-bit move operation, an overflow is detected according to M40:
		<ul> <li>the destination accumulator overflow status bit (ACOVx) is set.</li> </ul>
		<ul> <li>the destination register (ACx) is saturated according to SATD.</li> </ul>
		If the source (src) register is an auxiliary or temporary register, the 16 LSBs of the source register are sign extended to 40 bits according to SXMD.
		When the destination (dst) register is an auxiliary or temporary register:
		The 16-bit move operation is performed in the A-unit ALU.
		If the source (src) register is an accumulator, the 16 LSBs of the accumulator are used to perform the operation.
		Compatibility with C54x devices (C54CM = 1)
		When this instruction is executed with M40 = 0, compatibility is ensured.
Status	Bits	Affected by M40, SATD, SXMD
		Affects ACOVx
Repea	t	This instruction can be repeated.
See Al	so	See the following other related instructions:
		Move Accumulator Content to Auxiliary or Temporary Register
		Move Auxiliary or Temporary Register Content to Accumulator
		Move Auxiliary or Temporary Register Content to CPU Register
		Move Extended Auxiliary Register Content

Syntax		Desc	ription			
AC1 = AC0		The o	The content of AC0 is copied to AC1. Because an overflow occurred, ACOV1 is set to 1.			
Before				After		
AC0	01	E500	0030	AC0	01 E500 0030	
AC1	00	2800	0200	AC1	01 E500 0030	
M40			0	M40	0	
SATD			0	SATD	0	
ACOV1			0	ACOV1	1	

# Move Auxiliary or Temporary Register Content to Accumulator

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	HI(ACx) = TAx			Yes	2	1	Х	
Opcod	le			010	01 00	1E FSS	S 00DD	
Opera	nds	ACx, TAx						
Descri	ption		n moves the content of t t of the accumulator, A0	•	or temp	orary reg	jister (TAx)	
		The 16-bit	move operation is perfe	ormed in the	D-unit	ALU.		
		During the M40:	e 16-bit move operation	, an overflov	w is de	tected ac	cording to	
		the de	stination accumulator o	verflow statu	us bit (/	ACOVx) i	s set.	
		the de	estination register (ACx)	is saturated	accore	ding to S	ATD.	
			rce (src) register is and for the source register are			•	•	
		Compatibility with C54x devices (C54CM = 1)						
		When this inst	ruction is executed with	M40 = 0, co	ompatik	oility is er	sured.	
Status	Bits	Affected by	M40, SATD, SXMD					
		Affects	ACOVx					
Repeat	t	This instructio	n can be repeated.					
See Al	so	See the follow	ing other related instruc	tions:				
		Move Acc	umulator Content to Au	xiliary or Ten	nporary	/ Registe	r	
		Move Acc	umulator, Auxiliary, or Te	emporary Re	egister	Content		
		Move Aux	iliary or Temporary Reg	ister Conten	t to CP	U Regist	er	
		Move Exte	ended Auxiliary Register	r Content				
Fxamn	ble							

Syntax	Description
HI(AC0) = T0	The content of T0 is copied to AC0(31–16).

Move Auxiliary or Temporary Register Content to CPU Register

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	BRC0 = TAx		Yes	2	1	Х	
[2]	BRC1 = TAx		Yes	2	1	Х	
[3]	<b>CDP</b> = TAx		Yes	2	1	х	
[4]	<b>CSR</b> = TAx		Yes	2	1	Х	
[5]	<b>SP</b> = TAx		Yes	2	1	Х	
[6]	<b>SSP</b> = TAx		Yes	2	1	Х	
Opcod	e	See Table 5–3 (page 5-244)					
Operar	nds	TAx					
Descri	ption	This instruction moves the co to the selected CPU registe execute phase of the pipeline of the registers.	r. All the move opera	tions a	re perfor	med in the	
		There is a 3-cycle latency l update and their use in the units or by the P-unit loop co	address phase by the				
		For instruction [2] when BR repeat save register (BRS1)				, the block	
Status	Bits	Affected by none					
		Affects none					
Repeat	t	This instruction can be repeated.					
See Al	so	See the following other related instructions:					
		Move Accumulator Cont	ent to Auxiliary or Ter	nporar	y Registe	r	
		Move Accumulator, Auxi	iliary, or Temporary Ro	egister	Content		
		Move Auxiliary or Tempo	orary Register Conten	t to Ac	cumulato	r	
		Move CPU Register Cor	ntent to Auxiliary or Te	empora	ry Regist	er	
		Move Extended Auxiliary	y Register Content				

# Example

Syntax		Description		
BRC1 = T1		The content of T <sup>2</sup> repeat save regis	is copied to the block repeat register (BRC1) a er (BRS1).	nd to the block
Before		After		
т1	0034	Т1	0034	
BRC1	00EA	BRC1	0034	
BRS1	00EA	BRS1	0034	

# Table 5–3. Opcodes for Move Auxiliary or Temporary Register Content to CPU Register Instruction

No.	Syntax	Opcode
[1]	BRC0 = TAx	0101 001E FSSS 1110
[2]	BRC1 = TAx	0101 001E FSSS 1101
[3]	<b>CDP</b> = TAx	0101 001E FSSS 1010
[4]	<b>CSR</b> = TAx	0101 001E FSSS 1100
[5]	<b>SP</b> = TAx	0101 001E FSSS 1000
[6]	<b>SSP</b> = TAx	0101 001E FSSS 1001

# Move CPU Register Content to Auxiliary or Temporary Register

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	TAx = <b>BRC0</b>		Yes	2	1	Х	
[2]	TAx = <b>BRC1</b>		Yes	2	1	Х	
[3]	TAx = <b>CDP</b>		Yes	2	1	Х	
[4]	TAx = <b>SP</b>		Yes	2	1	Х	
[5]	TAx = <b>SSP</b>		Yes	2	1	Х	
[6]	TAx = <b>RPTC</b>		Yes	2	1	Х	
Opcod	e	See Table 5–4 (page 5-246).					
Operar	nds	TAx					
Description		This instruction moves the content of the selected CPU register to the auxiliary or temporary register (TAx). All the move operations are performed in the execute phase of the pipeline and the A-unit ALU is used to transfer the content of the registers.					
		For instructions [1] and [2], BRCx is a last instruction of a loop. These requirement versus the last instruction	instructions				
		For instructions [3], [4], and [5], ther CDP, and TAx update and their use in generator units or by the P-unit loop	the address p	hase b	y the A-u		
Status	Bits	Affected by none					
		Affects none					
Repeat	t	Instruction [6] cannot be repeated; all other instructions can be repeated.					
See Also		See the following other related instructions:					
		Move Accumulator Content to A	uxiliary or Ter	nporar	v Registe	r	
		Move Auxiliary or Temporary Re	-	-			
		Store CPU Register Content to	-		2.109.00		
			wennory				

#### Example

Syntax		Description		
T1 = BRC1		The content of block repeat register (BRC1) is copied to T1.		
Before		After		
т1	0034	Т1	00EA	
BRC1	00EA	BRC1	00EA	

# Table 5–4. Opcodes for Move CPU Register Content to Auxiliary or Temporary Register Instruction

No.	Syntax	Opcode
[1]	TAx = <b>BRC0</b>	0100 010E 1100 FDDD
[2]	TAx = <b>BRC1</b>	0100 010E 1101 FDDD
[3]	TAx = <b>CDP</b>	0100 010E 1010 FDDD
[4]	TAx = <b>SP</b>	0100 010E 1000 FDDD
[5]	TAx = <b>SSP</b>	0100 010E 1001 FDDD
[6]	TAx = <b>RPTC</b>	0100 010E 1110 FDDD

# Move Extended Auxiliary Register Content

Syntax Characteristics

No.	Syntax	Parallel Enable Bit Size Cycles Pipelin
[1]	xdst = xsrc	No 2 1 X
Opcod	le	1001 0000 XSSS XDD
Opera	nds	xdst, xsrc
Descri	ption	This instruction moves the content of the source register (xsrc) to the destination register (xdst):
		When the destination register (xdst) is an accumulator (ACx) and the source register (xsrc) is a 23-bit register (XARx, XSP, XSSP, XDP, XCDP):
		The 23-bit move operation is performed in the D-unit ALU.
		■ The upper bits of ACx are filled with 0.
		When the source register (xsrc) is an accumulator (ACx) and the destination register (xdst) is a 23-bit register (XARx, XSP, XSSP, XDP, XCDP):
		The 23-bit move operation is performed in the A-unit ALU.
		The lower 23 bits of ACx are loaded into xdst.
		When both the source register (xsrc) and the destination register (xdst) a accumulators, the Move Accumulator Content instruction (dst = src) assembled.
Status	Bits	Affected by none
		Affects none
Repea	t	This instruction can be repeated.
See Al	SO	See the following other related instructions:
		Load Extended Auxiliary Register from Memory
		Load Extended Auxiliary Register with Immediate Value
		Modify Extended Auxiliary Register Content
		<ul> <li>Store Extended Auxiliary Register Content to Memory</li> </ul>
Examp	h	
Cuntor		Description

Syntax	Description
XAR1 = AC0	The lower 23 bits of AC0 are loaded into XAR1.

SPRU375G

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = <b>coef(</b> Cmem <b>)</b>	No	3	1	х
[2]	coef(Cmem) = Smem	No	3	1	Х
[3]	Lmem = <b>dbl(coef(</b> Cmem <b>))</b>	No	3	1	х
[4]	dbl(coef(Cmem)) = Lmem	No	3	1	х
[5]	dbl(Ymem) = dbl(Xmem)	No	3	1	х
[6]	Ymem = Xmem	No	3	1	х

**Description** These instructions store the content of a memory location to a memory location. They use a dedicated datapath to perform the operation.

Status Bits Affected by none

Affects none

**See Also** See the following other related instructions:

- Store Accumulator Content to Memory
- Store Accumulator, Auxiliary, or Temporary Register Content to Memory
- Store Auxiliary or Temporary Register Pair Content to Memory
- □ Store CPU Register Content to Memory
- Store Extended Auxiliary Register Content to Memory

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = <b>coef</b>	(Cmem)		No	3	1	X
Opcode	e		1110	1111 AA	AA A	AAI XXX	x 00mm
Operar	nds	Cmem, Smem	ı				
<b>Description</b> This instruction stores the content addressed using the coefficient act location.							
		For this instruction, the Cmem operand is not accessed through the E On all C55x-based devices, the Cmem operand may be mapped in e or internal memory space.					
Status	Bits	Affected by	none				
		Affects	none				
Repeat	:	This instructio	n can be repeated.				

Syntax		Description		
*(#0500h) =	coef(*CDP)	The content address 0500h.	ressed by the coefficient data pointer register (CDP) is copie	ed to
Before		After		
*CDP	3400	*CDP	3400	
500	0000	500	3400	

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2]	coef(Cmem) :	= Smem			No	3	1	Х
Opcod	e			1110	1111 AA	AA Ai	AAI XXX	x 01mm
Operar	nds	Cmem, Smem	ı					
Description			on stores the cor em) location add			`	,	
			ction, the Cmem ased devices, th mory space.	•			•	
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructio	n can be repeate	ed.				
<b>-</b>	1.							

# Syntax Description coef(\*CDP) = \*AR3 The content addressed by AR3 is copied in the location addressed by the coefficient data pointer register (CDP).

# Syntax Characteristics

No.	Syntax			F	Parallel Inable Bit	Size	Cycles	Pipeline
	•			-				-
[3]	Lmem = dbl(c	:oef(Cmem))			No	3	1	Х
Opcod	e		12	110 2	1111 AA	AA AA	AAI XXX	xx 10mm
Operar	nds	Cmem, Lmem						
DescriptionThis instruction stores the content of two cons locations, addressed using the coefficien consecutive data memory (Lmem) locations.				fficient ad			,	
		For this instruction, the Cmem operand is not accessed through the BB On all C55x-based devices, the Cmem operand may be mapped in external memory space.						
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructior	n can be repeated.					

Syntax	Description
*AR1 = dbl(coef(*(CDP + T0)))	The content (long word) addressed by the coefficient data pointer register (CDP) and CDP + 1 is copied in the location addressed by AR1 and AR1 + 1, respectively. After the memory store, CDP is incremented by the content of T0 (5).

Before		After	
Т0	0005	Т0	0005
CDP	0200	CDP	0205
AR1	0300	AR1	0300
200	3400	200	3400
201	0FD3	201	0FD3
300	0000	300	3400
301	0000	301	0FD3

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dbl(coef(Cm	nem <b>))</b> = Lmem		No	3	1	Х
Opcod	e		1110	1111 AA	AA A	AAI XXX	cx 11mm
Operar	nds	Cmem, Lmem	1				
Descri	ption	locations to tw	n stores the content of o consecutive data mer addressing mode.				,
			ction, the Cmem opera ased devices, the Cme mory space.			0	
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instructio	n can be repeated.				
<b>-</b>							

Syntax	Description
dbl(coef(*CDP)) = *AR3+	The content (long word) addressed by AR3 and AR3 + 1 is copied in the location addressed by the coefficient data pointer register (CDP) and CDP + 1, respectively. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[5]	dbl(Ymem) = d	l <b>bl(</b> Xmem)			No	3	1	Х
Opcode	9			1000	0000 xx	XM MI	nyy   ymm	IM 00xx
Operan	ds	Xmem, Ymem						
Description		This instruction locations, addre memory (Ymer	essed using the					,
Status	Bits	Affected by	none					
		Affects	none					
Repeat		This instruction	n can be repeat	ed.				

Syntax	Description
dbl(*AR1) = dbl(*AR0)	The content addressed by AR0 is copied in the location addressed by AR1 and the content addressed by AR0 + 1 is copied in the location addressed by AR1 + 1.

Before		After	
AR0	0300	AR0	0300
AR1	0400	AR1	0400
300	3400	300	3400
301	0FD3	301	0FD3
400	0000	400	3400
401	0000	401	0FD3

					Parallel			
No.	Syntax				Enable Bit	Size	Cycles	Pipeline
[6]	Ymem = Xmem				No	3	1	Х
Opcode	e			1000	0000 XX	XM MI	MYY YMM	MM 01xx
Operan	nds	Xmem, Ymem						
Descrip	otion	This instruction addressed usi					•	,
Status	Bits	Affected by	none					
		Affects	none					
Repeat	:	This instructio	n can be repe	eated.				
Examp	le							
Syntax		Description						
*AR3 =	*AR5	The content ac	ldressed by AF	R5 is copied	l in the locatio	n addre	essed by A	R3.

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy * ACx)	Yes	2	1	Х
[2]	ACy = rnd(ACx * Tx)	Yes	2	1	х
[3]	ACy = rnd(ACx * K8)	Yes	3	1	х
[4]	ACy = rnd(ACx * K16)	No	4	1	х
[5]	ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	No	3	1	х
[6]	ACy = rnd(Smem * ACx)[, T3 = Smem]	No	3	1	х
[7]	ACx = rnd(Smem * K8)[, T3 = Smem]	No	4	1	х
[8]	ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	No	4	1	х
[9]	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	No	3	1	х

Description	This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are:
	□ ACx(32–16)
	the content of Tx, sign extended to 17 bits
	the 8-bit signed constant, K8, sign extended to 17 bits
	the 16-bit signed constant, K16, sign extended to 17 bits
	the content of a memory (Smem) location, sign extended to 17 bits
	the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits
	the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits
Status Bits	Affected by FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects ACOVx, ACOVy

See Also	See the following other related instructions:
	Modify Auxiliary Register Content with Parallel Multiply
	Multiply and Accumulate
	Multiply and Accumulate with Parallel Multiply
	Multiply and Subtract
	Multiply and Subtract with Parallel Multiply
	Multiply with Parallel Multiply and Accumulate
	Multiply with Parallel Store Accumulator Content to Memory
	Parallel Multiplies
	Square

# Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy	* AC	Sx)			Yes	2	1	Х
Opcode	)					010	)1 01	.0E DDS	S 011%
Operan	ds	AC	x, ACy						
Descrip	otion			n performs a m e multiplier are A	•				The input
			If FRCT =	1, the output of th	ne mul	tiplier is shift	ed left	by 1 bit.	
			Multiplicati	on overflow dete	ction d	epends on S	SMUL.		
		☐ The 32-bit result of the multiplication is sign extended to 40 bits.							
			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.						
			Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.						ected, the
			When an c SATD.	overflow is detected	ed, the	accumulato	r is sat	urated ac	ccording to
		Co	ompatibility	with C54x devid	ces (C	54CM = 1)			
		Wh	nen this insti	ruction is execute	ed with	M40 = 0, cc	mpatik	oility is en	sured.
Status	Bits	Aff	ected by	FRCT, M40, RE	DM, SA	ATD, SMUL			
		Aff	ects	ACOVy					
Repeat		Thi	is instructior	n can be repeated	d.				

#### Example

Syntax		D	escriptio	ı							
AC1 = AC1 * AC0 The content of				t of AC1 is mul	tiplied b	y the	content of	AC0 and the	result is	stored in A	C1.
Before				After							
AC0	02 6	000	3400	AC0	02	6000	3400				
AC1	00 C	000	0000	AC1	00	4800	0000				
M40			1	M40			1				
FRCT			0	FRCT			0				
ACOV1			0	ACOV1			0				
110011			0	110011			Ū				

#### SPRU375G

Instruction Set Descriptions 5-257

Multiply

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[2]	ACy = rnd(ACx	* Tx)			Yes	2	1	Х	
Opcod	e				01	01 10	0E DDS	S ss0%	
Operar	nds	AC	x, ACy, Tx						
Descri	ption	ope		n performs a multipl he multiplier are AC / bits.				•	
			If FRCT =	1, the output of the m	ultiplier is shif	ted left	by 1 bit.		
			Multiplication overflow detection depends on SMUL.						
			The 32-bit result of the multiplication is sign extended to 40 bits.						
			Rounding is performed according to RDM, if the optional rnd keyword applied to the instruction.					keyword is	
			<ul> <li>Overflow detection depends on M40. If an overflow is detected, destination accumulator overflow status bit (ACOVy) is set.</li> <li>When an overflow is detected, the accumulator is saturated accordin SATD.</li> </ul>					ected, the	
								ccording to	
		Compatibility with C54x devices (C54CM = 1)							
		Wł	nen this inst	ruction is executed wit	th M40 = 0, co	ompatik	oility is en	sured.	
Status	Bits	Aff	ected by	FRCT, M40, RDM, S	SATD, SMUL				
		Aff	ects	ACOVy					
Repeat	t	Thi	is instruction	n can be repeated.					
Evom									

Syntax	Description
AC0 = AC1 * T0	The content of AC1 is multiplied by the content of T0 and the result is stored in AC0.

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[3]	ACy = rnd(AC	Cx * K8 <mark>)</mark>		Yes	3	1	Х			
Opcod	е		0	001 111E KKI	KK KF	KKK SSI	DD xx0%			
Operar	nds	ACx, ACy, k	(8							
Descrij	ption	operands of	This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the 8-bit signed constant, K8, sign extended to 17 bits.							
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.								
		The 32-bit result of the multiplication is sign extended to 40 bits.								
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.								
		Compatibility with C54x devices (C54CM = 1)								
		When this ir	nstruction is executed	d with M40 = 0, co	ompatil	bility is er	nsured.			
Status	Bits	Affected by	FRCT, M40, RD	M						
		Affects	none							
Repeat	t	This instruct	tion can be repeated							
Examp	le									
Syntax	( C	Description								
AC0 = /	AC1 * #–2 T	he content of AC	I is multiplied by a signe	ed 8-bit value (–2) ar	nd the re	esult is sto	red in AC0.			

Multiply

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[4]	ACy = rnd(AC	x * K16 <mark>)</mark>		No	4	1	Х		
Opcod	e	0	11 1001 кккк	KKKK KKI	KK KK	KK SSE	D xx0%		
Operar	nds	ACx, ACy, K16							
Descri	ption	-	erforms a multiplic nultiplier are ACx(32 d to 17 bits.						
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.							
		Multiplication overflow detection depends on SMUL.							
		☐ The 32-bit result of the multiplication is sign extended to 40 bits.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.						
		When an over SATD.	flow is detected, the	accumulato	or is sat	urated ac	cording to		
		Compatibility with C54x devices (C54CM = 1)							
		When this instruct	ion is executed with	M40 = 0, co	ompatik	oility is en	sured.		
Status	Bits	Affected by F	RCT, M40, RDM, SA	ATD, SMUL					
		Affects A	COVy						
Repeat	t	This instruction ca	n be repeated.						

Syntax	Description
AC0 = AC1 * #-64	The content of AC1 is multiplied by a signed 16-bit value $(-64)$ and the result is stored in AC0.

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[5]	ACx = <mark>rnd(</mark> Sr	mem * <b>coef(</b> Cmem)	)[, T3 = Smem]	No	3	1	Х			
Opcod	e		1101	0001 AAA	AA AA	AI U%I	DD 00mm			
Operar	nds	ACx, Cmem,	Smem							
Descri	ption	operands of the extended to	on performs a multiplic ne multiplier are the cont 17 bits, and the conten ing the coefficient addre	tent of a men it of a data	nory (S memo	mem) loc ry operai	ation, sign nd Cmem,			
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.								
		Multiplica	Multiplication overflow detection depends on SMUL.							
		The 32-bit result of the multiplication is sign extended to 40 bits.								
		Rounding is performed according to RDM, if the optional rnd keywor applied to the instruction.								
		Overflow detection depends on M40. If an overflow is detected, th destination accumulator overflow status bit (ACOVx) is set.					ected, the			
		When an overflow is detected, the accumulator is saturated according SATD.								
		This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.								
		For this instruction, the Cmem operand is accessed through the BB bus some C55x-based devices, the BB bus is only connected to internal men and not to external memory. To prevent the generation of a bus error, Cmem operand must not be mapped on external memory.					al memory			
		Compatibility with C54x devices (C54CM = 1)								
		When this instruction is executed with M40 = 0, compatibility is ensured.								
Status	Bits	Affected by	FRCT, M40, RDM, S/	ATD, SMUL						
		Affects	ACOVx							
Repeat	:	This instruction	n can be repeated.							
Examp	le									

Syntax	Description
AC0 = *AR3 * coef(*CDP)	The content addressed by AR3 is multiplied by the content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0.

# Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
[6] ACy = rnd(Sm	nem * ACx) [,T3 = Smem]	No	3	1	Х		
Opcode	1101	0011 AAA	AA AA	AI U%I	D 00SS		
Operands	ACx, ACy, Smem						
Description	This instruction performs a multipli operands of the multiplier are ACx( (Smem) location, sign extended to 17	32–16) and					
	☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.						
	Multiplication overflow detection depends on SMUL.						
	The 32-bit result of the multiplication is sign extended to 40 bits.						
	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.						
	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.						
	When an overflow is detected, the accumulator is saturated according to SATD.						
	This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.						
	Compatibility with C54x devices (C54CM = 1)						
	When this instruction is executed wit	h M40 = 0, co	mpatil	oility is er	sured.		
Status Bits	Affected by FRCT, M40, RDM, S	ATD, SMUL					
	Affects ACOVy						
Repeat	This instruction can be repeated.						

Syntax	Description
AC0 = *AR3 * AC1	The content addressed by AR3 is multiplied by the content of AC1 and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[7]	ACx = <mark>rnd(</mark> Sm	nem * K8 <mark>) [,T3 = Sn</mark>	nem]	No	4	1	Х		
Opcod	e		1111 1000 AAAA	AAAI KKI	K KK	KK XXI	DD x0U%		
Operar	nds	ACx, K8, Sme	m						
Description		operands of th	This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.						
		□ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.							
		The 32-bit result of the multiplication is sign extended to 40 bits.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
		This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.							
		Compatibility with C54x devices (C54CM = 1)							
		When this inst	ruction is executed with	n M40 = 0, co	mpatik	oility is er	nsured.		
Status	Bits	Affected by	FRCT, M40, RDM						
		Affects	none						
Repeat	t	ing mode to ac	n cannot be repeated wi cess the memory opera s instruction can be rep	and (Smem);	•				

Syntax	Description
AC0 = *AR3 * #–2	The content addressed by AR3 is multiplied a signed 8-bit value (–2) and the result is stored in AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ACx = M40(I	rnd(uns()	Xmem <mark>)</mark> * uns(`	Ymem <mark>))</mark>	)[, T3 = Xmem]	No	4	1	Х
Opcod	e			1000	0110 XXXM	MMYY YMN	AM XX	DD 000	g uuU%
Operar	nds	AC	x, Xmem, Yn	nem					
Description		ope exte	rands of the	multipl bits,	ms a multipli lier are the co and the conte	ntent of data	memo	ory opera	nd Xmem,
			Input operar	nds are	extended to 1	7 bits accord	ling to	uns.	
					ns keyword is a location is zer	• •	•	•	he content
					uns keyword is nemory location				
			If FRCT = 1	, the ou	Itput of the mu	ltiplier is shift	ted left	by 1 bit.	
			Multiplicatio	n overfl	low detection of	lepends on S	SMUL.		
	The 32-bit result of the multiplication is sign extended to 40 bits.						ts.		
			Rounding is applied to the	•	med according uction.	to RDM, if tl	he opti	onal rnd l	keyword is
					depends on lator overflow				ected, the
			When an ov SATD.	erflow i	s detected, the	e accumulato	r is sat	turated ad	ccording to
					s the option to otional M40 key	•			
			s instruction em in tempo	-	es the option to gister T3.	store the 16	-bit da	ta memoi	ry operand
		Co	mpatibility v	vith C5	4x devices (C	54CM = 1)			
		Wh	en this instru	iction is	executed with	n M40 = 0, co	mpatil	oility is en	sured.

# Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = uns(*AR3) * uns(*AR4)	The unsigned content addressed by AR3 is multiplied by the unsigned
	content addressed by AR4 and the result is stored in AC0.

#### **Syntax Characteristics**

Nia	Suntor				Parallel	<b>C</b> i=-	Ovelas	Dineline
No.	Syntax	(T., * O.,))	[ T0 _ 0 1		Enable Bit	Size	Cycles	Pipeline
[9]	ACx = rnd(uns	(TX * Smem))	[, I 3 = Smem]		No	3	1	Х
Opcod	е			1101	0011 AAA	AA AA	AI U%D	D ulss
Opera	nds	ACx, Smem, Tx						
Description		operands	uction perform of the multiplie It of a memory	r are the cont	ent of Tx, sig	ın exte	nded to 1	7 bits, and
		🗋 If FRC	T = 1, the out	out of the mul	tiplier is shif	ed left	by 1 bit.	
		🗋 Multip	lication overflo	w detection d	lepends on S	SMUL.		
		The 32-bit result of the multiplication is extended to 40 bits according to uns						ling to uns.
			the optional u sult is zero ext	•		the in:	struction,	the 32-bit
			the optional un sult is sign ext	•		o the in	struction	, the 32-bit
		_	ling is perform d to the instruc	-	to RDM, if t	ne opti	onal rnd	keyword is
			ow detection ation accumula	•				ected, the
		U When SATD	an overflow is	detected, the	e accumulato	r is sat	urated ac	cording to
			This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.					y operand
		Compatik	oility with C54	x devices (C	54CM = 1)			
		When this	instruction is e	executed with	M40 = 0, co	mpatik	oility is en	sured.
Status	Bits	Affected b	y FRCT, N	140, RDM, SA	ATD, SMUL			
		Affects	ACOVx					
Repea	t	This instru	ction can be re	epeated.				
Examp	ole							

# Syntax Description AC0 = uns(T0 \* \*AR3) The content addressed by AR3 is multiplied by the content of T0 and the unsigned result is stored in AC0.

# Multiply with Parallel Multiply and Accumulate

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]		nd((AC)	Xmem) * uns(coef(Cmem <b>)</b> ))), / >> <b>#16) + (</b> uns(Ymem) *	No	4	1	Х
Opcod	e		1000 0100 XXXM	MMYY YM	MM 10	mm uuI	D DDg%
Operar	nds	AC	x, ACy, Cmem, Xmem, Ymem				
Description	ption	mu	s instruction performs two paralle Itiply and accumulate (MAC). The c Cs.	•		•	
		ope ext	e first operation performs a multi erands of the multiplier are the co ended to 17 bits, and the conte dressed using the coefficient addre	ontent of data nt of a data	n memo memo	ory opera ry opera	nd Xmem nd Cmem
		D-u me me	e second operation performs a m init MAC. The input operands of mory operand Ymem, extended mory operand Cmem, addressed ended to 17 bits.	the multiplie to 17 bits, a	er are and the	the conte content	ent of data of a data
			Input operands are extended to	17 bits accord	ding to	uns.	
			If the optional uns keyword is of the memory location is zero		•	•	he conten
			If the optional uns keyword is content of the memory location SXMD.				
			If FRCT = 1, the output of the mu	ultiplier is shif	ted left	by 1 bit.	
			Multiplication overflow detection	depends on \$	SMUL.		
			For the first operation, the 32- extended to 40 bits.	bit result of	the m	ultiplicati	on is sigr
			For the second operation, the 3 extended to 40 bits and added to to by 16 bits. The shifting operation source accumulator ACy(39).	he source ac	cumula	tor ACy s	hifted righ

	-	is performed according to RDM, if the optional rnd keyword is the instruction.
		detection depends on M40. If an overflow is detected, the n accumulator overflow status bit is set.
	When an or SATD.	overflow is detected, the accumulator is saturated according to
		n provides the option to locally set M40 to 1 for the execution of , if the optional M40 keyword is applied to the instruction.
	some C55x-ba and not to ex	ction, the Cmem operand is accessed through the BB bus; on ased devices, the BB bus is only connected to internal memory ternal memory. To prevent the generation of a bus error, the ad must not be mapped on external memory.
	while allowing	w can also disable the usage of the corresponding MAC unit, the modification of auxiliary registers in the three address its through the following instructions:
	■ mar(X	(mem)
	mar(Y	/mem)
	■ mar(C	Cmem)
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
Repeat	This instructio	n can be repeated.
See Also	See the follow	ing other related instructions:
	Multiply	
	Multiply a	nd Accumulate

Parallel Multiply and Accumulates

Syntax	Description
AC0 = uns(*AR3) * uns(coef(*CDP)), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The unsigned content addressed by AR3 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 shifted right by 16 bits and the result is stored in AC1.

# Multiply with Parallel Store Accumulator Content to Memory

No.	Syntax		Parallel Enable Bit	Size Cycles	Pipeline		
[1]	ACy = <mark>rnd(</mark> Tx Ymem = <b>HI(</b> A0	* Xmem), Cx <b>&lt;&lt; T2) [,T3 = Xmem]</b>	No	4 1	Х		
Opcode	e	1000 0111 XXXM	I MMYY YMMM	I SSDD 000	)x ssU%		
Operar	lds	ACx, ACy, Tx, Xmem, Ymem					
Descrip	otion	This instruction performs two operati	ons in parallel:	multiply and s	store.		
		The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of data memory operand Xmem, sign extended to 17 bits.					
		☐ If FRCT = 1, the output of the me	ultiplier is shifte	d left by 1 bit.			
		Multiplication overflow detection	depends on SM	IUL.			
		☐ The 32-bit result of the multiplication is sign extended to 40 bits.					
		Rounding is performed accordination applied to the instruction.	g to RDM, if the	e optional rnd	keyword is		
		Overflow detection depends or destination accumulator overflow			tected, the		
		When an overflow is detected, th SATD.	e accumulator i	is saturated a	ccording to		
		This instruction provides the operand Xmem in temporary reg		the 16-bit da	ta memory		
		The second operation shifts the accessores ACx(31–16) to data memory of is not within –32 to +31, the shift is performed with this value.	operand Ymem	. If the 16-bit	value in T2		
		The input operand is shifted in the input operand is shifte	ne D-unit shifter	according to	SXMD.		
		After the shift, the high part of the the memory location.	e accumulator, .	ACx(31–16),	is stored to		

### Compatibility with C54x devices (C54CM = 1)

	When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within $-32$ to $+31$ . When the 16-bit value in T2 is between $-32$ to $-17$ , a modulo 16 operation transforms the shift quantity to within $-16$ to $-1$ .			
Status Bits	Affected by	C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD		
	Affects	ACOVy		
Repeat	This instructior	n can be repeated.		
See Also	See the followi	ng other related instructions:		
	Addition w	th Parallel Store Accumulator Content to Memory		
	Multiply			
	Multiply ar Memory	nd Accumulate with Parallel Store Accumulator Content to		
	Multiply an	d Subtract with Parallel Store Accumulator Content to Memory		

- Store Accumulator Content to Memory
- Subtraction with Parallel Store Accumulator Content to Memory

Syntax	Description
AC1 = rnd(T0 * *AR0+), *AR1+ = HI(AC0 << T2)	Both instructions are performed in parallel. The content addressed by AR0 is multiplied by the content of T0. Since FRCT = 1, the result is multiplied by 2, rounded, and stored in AC1. The content of AC0 is shifted by the content of T2, and AC0(31–16) is stored at the address of AR1. AR0 and AR1 are both incremented by 1.

Before			After			
AC0	FF 8421	1234	AC0	FF	8421	1234
AC1	00 000	0000	AC1	00	2000	0000
AR0		0200	AR0			0201
AR1		0300	AR1			0301
т0		4000	Т0			4000
Т2		0004	Т2			0004
200		4000	200			4000
300		1111	300			4211
FRCT		1	FRCT			1
ACOV1		0	ACOV1			0
CARRY		0	CARRY			0

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy + (ACx * Tx))	Yes	2	1	Х
[2]	ACy = rnd((ACy * Tx) + ACx)	Yes	2	1	Х
[3]	ACy = rnd(ACx + (Tx * K8))	Yes	3	1	Х
[4]	ACy = rnd(ACx + (Tx * K16))	No	4	1	х
[5]	ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	No	3	1	Х
[6]	ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	No	3	1	х
[7]	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]	No	3	1	Х
[8]	ACy = rnd(ACx + (Smem * K8))[, T3 = Smem ]	No	4	1	х
[9]	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х
[10]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	х

#### Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are:

- □ ACx(32–16)
- L the content of Tx, sign extended to 17 bits
- the 8-bit signed constant, K8, sign extended to 17 bits
- the 16-bit signed constant, K16, sign extended to 17 bits
- the content of a memory (Smem) location, sign extended to 17 bits
- ☐ the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits
- ☐ the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits

Status Bits Affected by FRCT, M40, RE	M, SATD, SMUL, SXMD
---------------------------------------	---------------------

Affects ACOVx, ACOVy

SPRU375G

See Also	Se	e the following other related instructions:
		Modify Auxiliary Register Content with Parallel Multiply and Accumulate
		Multiply and Accumulate with Parallel Delay
		Multiply and Accumulate with Parallel Load Accumulator from Memory
		Multiply and Accumulate with Parallel Multiply
		Multiply and Accumulate with Parallel Store Accumulator Content to Memory
		Multiply and Subtract
		Multiply and Subtract with Parallel Multiply and Accumulate
		Multiply with Parallel Multiply and Accumulate
		Parallel Multiply and Accumulates

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy	+ <b>(</b> A	Cx * Tx <b>)</b> )		Yes	2	1	Х
Opcod	e				01	01 01	1E DDS	S 550%
Opera	nds	AC	x, ACy, Tx					
Descri	ption	MA	C. The inpu	n performs a multiplica ut operands of the multi ded to 17 bits.				
			If FRCT =	1, the output of the mu	Iltiplier is shif	ted left	by 1 bit.	
			Multiplicat	ion overflow detection of	depends on \$	SMUL.		
				result of the multiplication rce accumulator ACy.	on is sign ext	ended	to 40 bits	and added
			-	is performed according the instruction.	g to RDM, if t	he opti	onal rnd I	keyword is
				verflow detection depe ation accumulator over				
			When an according	addition overflow is d to SATD.	etected, the	accum	nulator is	saturated
		Co	ompatibility	with C54x devices (C	C54CM = 1)			
		Wł	nen this inst	ruction is executed with	n M40 = 0, co	ompatik	oility is en	sured.
Status	Bits	Aff	ected by	FRCT, M40, RDM, S	ATD, SMUL			
		Aff	ects	ACOVy				
Repea	t	Th	is instructio	n can be repeated.				
Evom								

Syntax	Description
	The content of AC1 multiplied by the content of T0 is added to the content of AC0 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = rnd((ACy	* Tx	() + ACx)		Yes	2	1	Х
Opcod	e				010	01 10	0E DDS	S ss1%
Operar	nds	AC	x, ACy, Tx					
Descrij	otion	MA	C. The inpu	n performs a multiplica t operands of the multi led to 17 bits.				
			If FRCT =	1, the output of the mu	ıltiplier is shif	ted left	by bit.	
			Multiplication	on overflow detection	depends on §	SMUL.		
				esult of the multiplicati ce accumulator ACx.	on is sign ext	ended	to 40 bits	and added
			-	s performed according	g to RDM, if t	ne opti	onal rnd I	keyword is
				verflow detection depe tion accumulator over				
			When an a a	addition overflow is d to SATD.	letected, the	accum	nulator is	saturated
		Со	mpatibility	with C54x devices (C	C54CM = 1)			
		Wł	nen this instr	uction is executed with	h M40 = 0, co	mpatik	oility is en	sured.
Status	Bits	Aff	ected by	FRCT, M40, RDM, S	ATD, SMUL			
		Aff	ects	ACOVy				
Repeat	:	Thi	is instruction	can be repeated.				

cription
content of AC1 multiplied by the content of T1 is added to the content of . The result is rounded and stored in AC1.

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[3] ACy = rnd(ACx	: + <b>(</b> Tx * K8 <b>)</b> )	Yes	3	1	X
Opcode	0001	111E   KKI	KK KK	KK SSE	D ss1%
Operands	АСх, АСу, К8, Тх				
Description	This instruction performs a multiplicate MAC. The input operands of the multip to 17 bits, and the 8-bit signed consta	lier are the co	ontent	of Tx, sigr	nextended
	☐ If FRCT = 1, the output of the mu	ltiplier is shif	ted left	by 1 bit.	
	The 32-bit result of the multiplication to the source accumulator ACx.	on is sign exte	ended	to 40 bits	and added
	Rounding is performed according applied to the instruction.	to RDM, if t	he opti	onal rnd I	keyword is
	Addition overflow detection deper the destination accumulator overf				
	When an addition overflow is de according to SATD.	etected, the	accum	nulator is	saturated
	Compatibility with C54x devices (C	54CM = 1)			
	When this instruction is executed with	n M40 = 0, co	ompatik	oility is en	sured.
Status Bits	Affected by FRCT, M40, RDM, S/	ATD			
	Affects ACOVy				
Repeat	This instruction can be repeated.				
Example					

Syntax	Description
AC0 = AC1 + (T0 * K8)	The content of T0 multiplied by a signed 8-bit value is added to the content of AC1 and the result is stored in AC0.

# Syntax Characteristics

No. Sy	ntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[4] AC	Sy = rnd(ACx + (	Tx * K16 <b>)</b> )		No	4	1	Х	
Opcode			0111 1001 КККК	KKKK KKF	кк кк	KK SSD	D ssl%	
Operands	A	Cx, ACy, K16	, Tx					
DescriptionThis instruction performs aMAC. The input operands ofto 17 bits, and the 16-bit signal			operands of the multipl	lier are the co	ontent	of Tx, sigr	n extended	
		) If FRCT = 1	I, the output of the mul	tiplier is shift	ed left	by 1 bit.		
		Multiplication overflow detection depends on SMUL.						
		The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.						
			s performed according he instruction.	to RDM, if th	ne opti	onal rnd I	keyword is	
			erflow detection depen tion accumulator overfl					
		When an a a a a a a a a a a a a a a a a a	addition overflow is de o SATD.	etected, the	accum	ulator is	saturated	
	С	compatibility	with C54x devices (C	54CM = 1)				
	W	/hen this instr	uction is executed with	M40 = 0, cc	mpatik	oility is en	sured.	
Status Bits	s At	ffected by	FRCT, M40, RDM, SA	TD, SMUL				
	At	ffects	ACOVy					
Repeat	ті	his instruction	can be repeated.					

Syntax	Description				
AC0 = AC1 + (T0 * #FFFFh)	The content of T0 multiplied by a signed 16-bit value (FFFFh) is added to the content of AC1 and the result is stored in AC0.				

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[5]	•	+ (9	mem * <b>coef</b> (	Cmem <b>)))</b> [, T3 = S	Smeml	No	3120	1	Х
[0]		- <b>(</b> 0		omeni <b>)))</b> [, 10 – 0	Smennj	NO	5	I	~
Opcod	е				1101	0001 AAA	AA AA	AI U%D	DD 01mm
Operai	nds	AC	x, Cmem, S	Smem					
Descri	ption	MA loc Crr	C. The inpu ation, sign e	n performs a m toperands of th extended to 17 ssed using the	e multipl bits, and	ier are the co the content	ntent o of a da	f a memo ta memo	ry (Smem) ry operanc
			☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.						
			Multiplicati	ion overflow de	tection d	lepends on S	SMUL.		
		The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVx) is set.						
		When an addition overflow is detected, the accumulator is saturate according to SATD.					saturated		
		This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.							
		sor and	me C55x-ba d not to ext	ction, the Cmer used devices, th ernal memory. d must not be r	ne BB bu To prev	s is only con ent the gene	nected eration	to intern of a bus	al memory
		Co	mpatibility	with C54x de	vices (C	54CM = 1)			
		When this instruction is executed with $M40 = 0$ , compatibility is ensured.							
Status	Bits	Aff	ected by	FRCT, M40,	RDM, SA	ATD, SMUL			
		Aff	ects	ACOVx					
Repea	t	Thi	s instruction	n can be repea	ted.				
SPRU3	75G					Instruction	Set Des	criptions	5-277

Syntax				Description		
AC2 = rnd(AC2 + (*AR1 * coef(*CDP)))				The content addressed by AR1 multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC2. The result is rounded and stored in AC2. The result generated an overflow.		
Before			Aft	er		
AC2	00 EC00	0000	AC2	00 EC00 0000		
AR1		0302	AR2	0302		
CDP		0202	CDP	0202		
302		FE00	302	FEOO		
202		0040	202	0040		
ACOV2		0	ACO	V2 1		

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[6]		( + <b>(</b> Smem * A(	x <b>)</b> )[, T3 = Smem]		No	3120	1	X	
[0]			xj)[, 10 – Offenij		110	5	1	~	
Opcod	e			1101	0010 AAA	AA AA	AI U%D	D 00SS	
Operar	nds	ACx, ACy, S	Smem						
Descri	ption	MAC. The i	tion performs a nput operands o Smem) location	f the multip	olier are ACx	(32–16			
			☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.						
		Multiplication overflow detection depends on SMUL.							
		The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
		Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
		When an addition overflow is detected, the accumulator is saturated according to SATD.							
			tion provides the mporary registe	-	store the 16	-bit da	ta memoi	y operand	
		Compatibi	ity with C54x a	levices (C	54CM = 1)				
		When this i	nstruction is exe	cuted with	M40 = 0, co	ompatik	oility is en	sured.	
Status	Bits	Affected by	FRCT, M40	), RDM, SA	ATD, SMUL				
		Affects	ACOVy						
Repeat	<b>Repeat</b> This instruction can be repeated.								
Examp	le								

Syntax	Description
AC1 = AC1 + (*AR3 * AC0)	The content addressed by AR3 multiplied by the content of AC0 is added to the content of AC1 and the result is stored in AC1.

# Syntax Characteristics

Operands       ACx, ACy, Smem, Tx         Description       This instruction performs a multiplication and an accumulation in the D-un MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory (Smem) location, sign extended to 17 bits.         Image: If FRCT = 1, the output of the multiplier is shifted left by 1 bit.       Multiplication overflow detection depends on SMUL.         Image: The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.       Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.         Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.       When an addition overflow is detected, the accumulator is saturate according to SATD.	No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
Operands       ACx, ACy, Smem, Tx         Description       This instruction performs a multiplication and an accumulation in the D-un MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory (Smem) location, sign extended to 17 bits.         If FRCT = 1, the output of the multiplier is shifted left by 1 bit.       Multiplication overflow detection depends on SMUL.         The 32-bit result of the multiplication is sign extended to 40 bits and adde to the source accumulator ACx.       Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.         Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.       When an addition overflow is detected, the accumulator is saturate according to SATD.         This instruction provides the option to store the 16-bit data memory operant Smem in temporary register T3.       Compatibility with C54x devices (C54CM = 1)         When this instruction is executed with M40 = 0, compatibility is ensured.       Affected by       FRCT, M40, RDM, SATD, SMUL         Affects       ACOVy	[7]	ACy = rnd(ACx	< + <b>(</b> Tx * Smem <b>))[</b> ,	T3 = Smem]	No	3	1	х
Description       This instruction performs a multiplication and an accumulation in the D-un MAC. The input operands of the multiplier are the content of Tx, sign extende to 17 bits, and the content of a memory (Smem) location, sign extended to 17 bits.         If FRCT = 1, the output of the multiplier is shifted left by 1 bit.         Multiplication overflow detection depends on SMUL.         The 32-bit result of the multiplication is sign extended to 40 bits and adde to the source accumulator ACx.         Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.         Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.         When an addition overflow is detected, the accumulator is saturate according to SATD.         This instruction provides the option to store the 16-bit data memory operan Smem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)         When this instruction is executed with M40 = 0, compatibility is ensured.         Affected by       FRCT, M40, RDM, SATD, SMUL         Affects       ACOVy	Opcode			1101	0100 AAA	AA AA	AI U%D	D ssSS
<ul> <li>MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory (Smem) location, sign extended to 17 bits.</li> <li>If FRCT = 1, the output of the multiplier is shifted left by 1 bit.</li> <li>Multiplication overflow detection depends on SMUL.</li> <li>The 32-bit result of the multiplication is sign extended to 40 bits and adde to the source accumulator ACx.</li> <li>Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.</li> <li>Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.</li> <li>When an addition overflow is detected, the accumulator is saturate according to SATD.</li> <li>This instruction provides the option to store the 16-bit data memory operan Smem in temporary register T3.</li> <li>Compatibility with C54x devices (C54CM = 1)</li> <li>When this instruction is executed with M40 = 0, compatibility is ensured.</li> <li>Affected by FRCT, M40, RDM, SATD, SMUL</li> <li>Affects ACOVy</li> </ul>	Opera	nds	ACx, ACy, Sm	iem, Tx				
<ul> <li>Multiplication overflow detection depends on SMUL.</li> <li>The 32-bit result of the multiplication is sign extended to 40 bits and adde to the source accumulator ACx.</li> <li>Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.</li> <li>Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.</li> <li>When an addition overflow is detected, the accumulator is saturate according to SATD.</li> <li>This instruction provides the option to store the 16-bit data memory operan Smem in temporary register T3.</li> <li>Compatibility with C54x devices (C54CM = 1)</li> <li>When this instruction is executed with M40 = 0, compatibility is ensured.</li> <li>Status Bits</li> <li>Affected by FRCT, M40, RDM, SATD, SMUL</li> <li>Affects ACOVy</li> </ul>	Description		MAC. The inpute to 17 bits, and	ut operands of the multip	lier are the co	ontent	of Tx, sigr	n extended
<ul> <li>The 32-bit result of the multiplication is sign extended to 40 bits and adde to the source accumulator ACx.</li> <li>Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.</li> <li>Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.</li> <li>When an addition overflow is detected, the accumulator is saturate according to SATD.</li> <li>This instruction provides the option to store the 16-bit data memory operant Smem in temporary register T3.</li> <li>Compatibility with C54x devices (C54CM = 1)</li> <li>When this instruction is executed with M40 = 0, compatibility is ensured.</li> <li>Status Bits</li> <li>Affected by FRCT, M40, RDM, SATD, SMUL</li> <li>Affects ACOVy</li> </ul>			☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.					
to the source accumulator ACx.         Rounding is performed according to RDM, if the optional rnd keyword i applied to the instruction.         Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.         When an addition overflow is detected, the accumulator is saturate according to SATD.         This instruction provides the option to store the 16-bit data memory operant Smem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)         When this instruction is executed with M40 = 0, compatibility is ensured.         Affected by       FRCT, M40, RDM, SATD, SMUL         Affects       ACOVy			Multiplication overflow detection depends on SMUL.					
<ul> <li>applied to the instruction.</li> <li>Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.</li> <li>When an addition overflow is detected, the accumulator is saturate according to SATD.</li> <li>This instruction provides the option to store the 16-bit data memory operant Smem in temporary register T3.</li> <li><i>Compatibility with C54x devices (C54CM = 1)</i></li> <li>When this instruction is executed with M40 = 0, compatibility is ensured.</li> <li>Status Bits</li> <li>Affected by FRCT, M40, RDM, SATD, SMUL</li> <li>Affects ACOVy</li> </ul>								
the destination accumulator overflow status bit (ACOVy) is set.         When an addition overflow is detected, the accumulator is saturate according to SATD.         This instruction provides the option to store the 16-bit data memory operan Smem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)         When this instruction is executed with M40 = 0, compatibility is ensured.         Status Bits       Affected by         FRCT, M40, RDM, SATD, SMUL         Affects       ACOVy			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.					
according to SATD.         This instruction provides the option to store the 16-bit data memory operan Smem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)         When this instruction is executed with M40 = 0, compatibility is ensured.         Status Bits       Affected by         FRCT, M40, RDM, SATD, SMUL         Affects       ACOVy			Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.					
Smem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)         When this instruction is executed with M40 = 0, compatibility is ensured.         Status Bits       Affected by         FRCT, M40, RDM, SATD, SMUL         Affects       ACOVy								saturated
Status BitsAffected byFRCT, M40, RDM, SATD, SMULAffectsACOVy			This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.					
Status Bits     Affected by     FRCT, M40, RDM, SATD, SMUL       Affects     ACOVy			Compatibility with C54x devices (C54CM = 1)					
Affects ACOVy			When this inst	ruction is executed with	M40 = 0, co	ompatik	oility is en	sured.
	Status	Bits	Affected by	FRCT, M40, RDM, SA	ATD, SMUL			
<b>Repeat</b> This instruction can be repeated.			Affects	ACOVy				
	Repea	t	This instructio	n can be repeated.				

Syntax	Description
AC0 = AC1 + (T0 * *AR3)	The content addressed by AR3 multiplied by the content of T0 is added to the
	content of AC1 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax		rallel ble Bit	Size	Cycles	Pipeline			
[8]		x + (Smem * K8))[, T3 = Smem]	No	4	1	X			
Opcod	e	1111 1000 AAAA AAA	I KKH	K KF	KKK SSI	DD x1U%			
Operar	nds	ACx, ACy, K8, Smem	I		I				
Description		MAC. The input operands of the multiplier ar	This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits, and the 8-bit signed constant, K8, sign extended to 17 bits.						
		☐ If FRCT = 1, the output of the multiplie	□ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.						
		The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
		Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
		When an addition overflow is detected, the accumulator is saturated according to SATD.							
		This instruction provides the option to store the 16-bit data memory operar Smem in temporary register T3.							
		Compatibility with C54x devices (C54CM = 1)							
		When this instruction is executed with M40	= 0, cc	mpatil	oility is er	nsured.			
Status	Bits	Affected by FRCT, M40, RDM, SATD							
		Affects ACOVy							
Repeat	t	This instruction cannot be repeated when us ing mode to access the memory operand (S ing modes, this instruction can be repeated	mem);						

Syntax	Description				
	The content addressed by AR3 multiplied by a signed 8-bit value (FFh) is added to the content of AC1 and the result is stored in AC0.				

No.	Syntax	Parallel Enable Bit Size Cycles Pipeline					
[9]	ACy = M40(rr [, T3 = Xmem	(ACx + (uns(Xmem) * uns(Ymem)))) No 4 1 X					
Opcod	e	1000 0110 XXXM MMYY YMMM SSDD 001g uuU%					
Opera	nds	ACx, ACy, Xmem, Ymem					
Description		This instruction performs a multiplication and an accumulation in the D-ur MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.					
		Input operands are extended to 17 bits according to uns.					
		If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits.					
		If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD.					
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.					
		Multiplication overflow detection depends on SMUL.					
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.					
		Rounding is performed according to RDM, if the optional rnd keyword applied to the instruction.					
		Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVy) is set.					
		When an addition overflow is detected, the accumulator is saturate according to SATD.					
		This instruction provides the option to locally set M40 to 1 for the execution the instruction, if the optional M40 keyword is applied to the instruction.					
		This instruction provides the option to store the 16-bit data memory operan Xmem in temporary register T3.					
		Compatibility with C54x devices (C54CM = 1)					
		When this instruction is executed with M40 = 0, compatibility is ensured.					

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD		
	Affects	ACOVy		
Repeat	This instruction can be repeated.			

Syntax			Description				
AC3 = rnd(	(AC3 + (uns(*AR2+) * เ	uns(*AR3+)))	The unsigned content addressed by AR2 multiplied unsigned content addressed by AR3 is added to the of of AC3. The result is rounded and stored in AC3. The generated an overflow. AR2 and AR3 are both increr by 1.	content e result			
Before		After					
AC3	00 2300 EC00	AC3	00 9221 0000				
AR2	302	AR2	303				
AR3	202	AR3	203				
ACOV3	0	ACOV3	1				
302	FEOO	302	FEOO				
202	7000	202	7000				
M40	0	M40	0				
SATD	0	SATD	0				
FRCT	0	FRCT	0				

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[10]	ACy = M40( [, T3 = Xmer		> <b>#16)</b> + <b>(</b> uns(Xmem)	* uns(Ymem))) <b>)</b>	No	4	1	Х	
Орсо	ode		1000	0110 XXXM	MMYY YMN	MM SS	SDD 010	g uuU%	
Oper	ands	AC	x, ACy, Xmem, Yme	em					
Desc	ription	MA ope	s instruction perform C. The input operater erand Xmem, extender em, extended to 17	nds of the mul ded to 17 bits, a	tiplier are the	e conte	ent of dat	a memory	
			Input operands are	extended to 1	7 bits accord	ling to	uns.		
				ins keyword is a location is zero		•	•	he content	
				uns keyword is nemory locatior					
			If FRCT = 1, the or	utput of the mul	tiplier is shift	ted left	by 1 bit.		
			Multiplication overflow detection depends on SMUL.						
			The 32-bit result of to the source acc operation is perfo ACx(39).	umulator ACx	shifted right	t by 1	6 bits. Tl	he shifting	
			Rounding is perfor applied to the instr		to RDM, if tl	ne opti	onal rnd	keyword is	
			Addition overflow of the destination acc						
			When an addition according to SATD		etected, the	accun	nulator is	saturated	
			s instruction provide instruction, if the o	•	•				
			s instruction provide em in temporary re		store the 16	-bit da	ta memoi	ry operand	

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
-------------	-------------	----------------------------------

Affects ACOVy

**Repeat** This instruction can be repeated.

Syntax	Description
AC0 = (AC1 >> #16) + (uns(*AR3) * uns(*AR4))	The unsigned content addressed by AR3 multiplied by the unsigned content addressed by AR4 is added to the content of AC1 shifted right by 16 bits and the result is stored in AC0.

# Multiply and Accumulate with Parallel Delay

No.	Syntax									Para Enab		Size	e C	ycles	Pi	peline
[1]	ACx = rnd(ACx delay(Smem)	+ <b>(</b> Sn	nem	* coef	(Cmen	n <b>)))[</b> , <sup>·</sup>	T3 = 5	Smem],	I	N	0	3		1		Х
Opcod	e							110	1	0000	AA	AA A	AAI	U%I	DD	xxmm
Operar	nds	AC	k, Cr	nem,	Smen	n										
Descri	MA mul 17 t	C in tiplie oits,	parall er are and th	el with the c he cor	h the conte ntent	delay nt of of a c	memo a men	ory nor <u>j</u> iem	instru y (Sm ory oj	iction nem) peran	. The locati id Cm	inpu on, ៖ em,	t opera sign e	and xter	e D-unit s of the nded to d using	
			lf Fl	RCT =	= 1, th	e out	put o	f the m	nulti	plier i	s shil	fted le	ft by	1 bit.		
			Mul	tiplica	tion o	verflo	ow de	tectior	n de	epend	s on	SMUL				
								Itiplica or ACx.		n is się	gn ext	tendeo	d to 4	0 bits	and	added
			Rounding is performed according to RDM, if the optional rnd keyword applied to the instruction.					word is								
			Addition overflow detection depends on M40. If an overflow is detected the destination accumulator overflow status bit (ACOVx) is set.						etected,							
		When an addition overflow is detected, the accumulator is s according to SATD.						s sa	turated							
					on pro porary			-	to s	store	:he 16	6-bit d	ata r	nemo	ry c	perand
	For this instruction, the Cmem operand is accessed the some C55x-based devices, the BB bus is only connected and not to external memory. To prevent the generation Cmem operand must not be mapped on external memory					ed to n of	intern	al r	nemory							
		inst	ructi	on. Tł	nis ins	tructi	on ca	•	se t	he *p	ort(#k	(16) a	ddre	ssing		d to this de or be
		to I	/O s	pace		erates						-		-	-	access ) to be

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 set to 0, compatibility is ensured.

Status Bits	Affect	ted by	FRCT, M40, RDM, SATD, SMUL
	Affect	S	ACOVx
Repeat	This i	nstruction	can be repeated.
See Also	See t	he followir	ng other related instructions:
	DM	lodify Auxi	iliary Register Content with Parallel Multiply and Accumulate
	DM	lultiply and	d Accumulate
	DM	lultiply and	d Accumulate with Parallel Load Accumulator from Memory
	DM	lultiply and	d Accumulate with Parallel Multiply
	_	lultiply an lemory	d Accumulate with Parallel Store Accumulator Content to
	DM	lultiply and	d Subtract with Parallel Multiply and Accumulate
	DM	lultiply with	h Parallel Multiply and Accumulate
	🗋 P	arallel Mu	Itiply and Accumulates

Syntax	Description
delay(*AR3)	The content addressed by AR3 multiplied by the content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0. The content addressed by AR3 is copied into the next higher address.

### Multiply and Accumulate with Parallel Load Accumulator from Memory

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	ACx = rnd(ACx ACy = Ymem <				No	4	1	Х		
Opcod	e		1000 0110 XX	MXX	MMYY YMN	/IM DE	DD   101	.x ssU%		
Operar	lds	AC	x, ACy, Tx, Xmem, Ymem							
Descri	otion		s instruction performs two ope AC), and load.	ratior	ns in paralle	l: multi	ply and a	ccumulate		
		MA to	e first operation performs a mult C. The input operands of the m 17 bits, and the content of data bits.	ultipl	ier are the co	ontent	of Tx, sigi	n extended		
			If FRCT = 1, the output of the	mult	iplier is shift	ed left	by 1 bit.			
			Multiplication overflow detection depends on SMUL.							
			The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.							
			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			Addition overflow detection detection detection detection accumulator of	•						
			When an addition overflow i according to SATD.	is de	tected, the	accum	nulator is	saturated		
			This instruction provides the operand Xmem in temporary			the 1	6-bit dat	a memory		
			e second operation loads the fted left by 16 bits to the accun			memo	ory opera	and Ymem		
			The input operand is sign exte	ende	d to 40 bits	accord	ling to SX	MD.		
			The shift operation is equivale	ent to	the signed	shift ir	struction			
			The input operand is shifted le	eft by	/ 16 bits acc	ording	to M40.			

# Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
Repeat	This instructior	n can be repeated.
See Also	See the followi	ing other related instructions:
	Modify Aux	kiliary Register Content with Parallel Multiply and Accumulate
	Multiply an	d Accumulate
	Multiply ar	d Accumulate with Parallel Delay
	Multiply ar	d Accumulate with Parallel Multiply
	Multiply an Memory	nd Accumulate with Parallel Store Accumulator Content to
	Multiply ar	d Subtract with Parallel Load Accumulator from Memory
	Multiply wi	th Parallel Multiply and Accumulate
	Parallel Mu	ultiply and Accumulates

Syntax	Description
AC0 = AC0 + (T0 * *AR3), AC1 = *AR4 << #16	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is added to the content of AC0 and the result is stored in AC0. The content addressed by AR4 shifted left by 16 bits is stored in AC1.

# Multiply and Accumulate with Parallel Multiply

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]			ns(Xmem) * uns(c m) * uns(coef(Cm		No	4	1	Х		
Орсо	de		1000	0010 XXXM	MMYY YMM	4M 01	Lmm uuI	DD DDg%		
Opera	ands	ACx, A	ACy, Cmem, Xm	em, Ymem						
Desci	iption	accum	nstruction perfor nulate (MAC), a MACs.	•	•		•			
		MAC. opera	st operation perf The input opera nd Xmem, sign e nd Cmem, addre bits.	ands of the mulextended to 17	tiplier are the bits, and the	e conte conter	ent of dat nt of a da	ta memory ta memory		
		operai extend	econd operation nds of the multip ded to 17 bits, ssed using the c	olier are the con and the conten	ntent of data it of a data	memo memo	ory opera ry opera	nd Ymem, nd Cmem,		
		🗋 In	put operands ar	7 bits accord	7 bits according to uns.					
		-	•	uns keyword is a location is zero		•	•	he content		
		•	-	uns keyword is memory locatior						
		🗋 lf	FRCT = 1, the o	utput of the mu	ltiplier is shif	ted left	by 1 bit.			
		ШM	ultiplication over	flow detection o	lepends on S	SMUL.				
			or the first oper tended to 40 bit				•	-		
			For the second operation, the 32 extended to 40 bits.			2-bit result of the multiplication is sign				
			ounding is perform plied to the inst	-	to RDM, if t	he opti	onal rnd	keyword is		

	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
	For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.
	Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:
	■ mar(Xmem)
	■ mar(Ymem)
	■ mar(Cmem)
Status Bits	Affected by FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects ACOVx, ACOVy
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Modify Auxiliary Register Content with Parallel Multiply and Accumulate
	Multiply and Accumulate
	Multiply and Accumulate with Parallel Delay
	Multiply and Accumulate with Parallel Load Accumulator from Memory
	Multiply and Accumulate with Parallel Store Accumulator Content to Memory
	Multiply and Subtract with Parallel Multiply
	<ul> <li>Multiply with Parallel Multiply and Accumulate</li> </ul>
	Parallel Multiply and Accumulates

Syntax	Description
AC0 = AC0 + (uns(*AR3) * uns(coef(*CDP))), AC1 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by CDP and the result is stored in AC1.

Multiply and Accumulate with Parallel Store Accumulator Content to Memory

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = <mark>rnd(</mark> ACy Ymem = <b>HI(</b> AC		x * Xmem)), <b>T2) [,</b> T3 = Xmem]	No	4	1	Х
Opcod	e		1000 0111 XXXM	MMYY YMN	/M SS	DD 001	x ssU%
Opera	nds	AC	x, ACy, Tx, Xmem, Ymem				
Descri	ption		is instruction performs two operatic AC), and store.	ons in paralle	l: multi	ply and a	ccumulate
		MA to <sup>-</sup>	e first operation performs a multiplic AC. The input operands of the multip 17 bits, and the content of data me bits.	lier are the co	ontent	of Tx, sigi	n extended
			If FRCT = 1, the output of the mu	ltiplier is shift	ed left	by 1 bit.	
			Multiplication overflow detection of	lepends on S	SMUL.		
			The 32-bit result of the multiplication to the source accumulator ACy.	on is sign exte	ended	to 40 bits	and added
			Rounding is performed according applied to the instruction.	to RDM, if tl	ne opti	onal rnd	keyword is
			Addition overflow detection deper the destination accumulator overf				
			When an addition overflow is de according to SATD.	etected, the	accum	nulator is	saturated
			This instruction provides the op operand Xmem in temporary regimentation of the second structure of th		the 1	6-bit dat	a memory
		sto is r	e second operation shifts the accu ores ACx(31–16) to data memory o not within –32 to +31, the shift is s rformed with this value.	perand Yme	m. If th	e 16-bit v	alue in T2/
			The input operand is shifted in the	e D-unit shift	er acco	ording to	SXMD.
			After the shift, the high part of the the memory location.	accumulato	r, ACx(	31–16), i	s stored to

# Compatibility with C54x devices (C54CM = 1)

	When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within $-32$ to $+31$ . When the 16-bit value in T2 is between $-32$ to $-17$ , a modulo 16 operation transforms the shift quantity to within $-16$ to $-1$ .
Status Bits	Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects ACOVy
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Modify Auxiliary Register Content with Parallel Multiply and Accumulate
	Multiply and Accumulate
	Multiply and Accumulate with Parallel Delay
	Multiply and Accumulate with Parallel Load Accumulator from Memory
	Multiply and Accumulate with Parallel Multiply
	Multiply and Subtract with Parallel Store Accumulator Content to Memory
	Multiply with Parallel Multiply and Accumulate
	Parallel Multiply and Accumulates
Example	

Syntax	Description
*AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is added to the content of AC0 and the result is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31–16) is stored at the address of AR4.

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy - (ACx * Tx))	Yes	2	1	Х
[2]	ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]	No	3	1	х
[3]	ACy = rnd(ACy - (Smem * ACx))[, T3 = Smem]	No	3	1	х
[4]	ACy = rnd(ACx - (Tx * Smem))[, T3 = Smem]	No	3	1	х
[5]	ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	No	4	1	Х

#### Description

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are:

- ACx(32–16)
- the content of Tx, sign extended to 17 bits
- Let the content of a memory (Smem) location, sign extended to 17 bits
- ☐ the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits
- the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits
- Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

- See Also See the following other related instructions:
  - Modify Auxiliary Register Content with Parallel Multiply and Subtract
  - Multiply and Accumulate
  - Multiply and Subtract with Parallel Load Accumulator from Memory
  - Multiply and Subtract with Parallel Multiply
  - Multiply and Subtract with Parallel Multiply and Accumulate
  - Multiply and Subtract with Parallel Store Accumulator Content to Memory
  - Parallel Multiply and Subtracts

### **Syntax Characteristics**

						Parallel				
No.	Syntax					Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = rnd(ACy	- <b>(</b> A	Cx * Tx <b>)</b> )			Yes	2	1	Х	
Opcod	e					010	01 01	1E DDS	S ssl%	
Operar	nds	AC	x, ACy, Tx							
Descri	ption	Th	This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are $ACx(32-16)$ and the content of Tx, sign extended to 17 bits.							
			If FRCT =	1, the output o	of the mul	tiplier is shift	ted left	by 1 bit.		
			Multiplicati	on overflow de	etection d	lepends on S	SMUL.			
			The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.							
			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			Overflow detection depends on M40. If an overflow is detected, destination accumulator overflow status bit (ACOVy) is set.						ected, the	
			When an c SATD.	overflow is dete	ected, the	accumulato	r is sat	urated ad	ccording to	
		Co	ompatibility	with C54x de	evices (C	54CM = 1)				
		Wł	nen this inst	ruction is exec	cuted with	M40 = 0, cc	mpatik	oility is en	sured.	
Status	Bits	Aff	ected by	FRCT, M40,	, RDM, SA	ATD, SMUL				
		Aff	ects	ACOVy						
Repeat	t	Th	is instructior	n can be repea	ated.					
	-									

# Example

Syntax				Description	Description						
AC1 = rn	d(AC1 –	(AC0	* T1))		The content of AC0 multiplied by the content of T1 is subtracted from the content of AC1. The result is rounded and stored in AC1.						
Before				After							
AC0	00	EC00	0000	AC0	00	EC00	0000				
AC1	00	3400	0000	AC1	00	1680	0000				
Т1			2000	T1			2000				
M40			0	M40			0				
ACOV1			0	ACOV1			0				
FRCT			0	FRCT			0				

#### SPRU375G

Instruction Set Descriptions 5-295

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACx = rnd(ACx	– <b>(</b> Srr	nem '	* <b>coef(</b> Cmer	m <b>)))[</b> , T3 =	= Smem]	No	3	1	Х
Opcod	e					1101	0001 AAA	AA AZ	AAI U%I	D 10mm
Operar	nds	AC×	k, Cn	nem, Smer	n					
Descri	ption	The loca	inpu ition, em,	ut operand , sign exter	s of the ided to 1	multiplier 7 bits, and	on and a sub are the cont the content nt addressin	ent of of a da	a memo ta memo	ry (Smem) ry operand
			lf FF	RCT = 1, th	e output	of the mul	ltiplier is shif	ted left	t by 1 bit.	
			Mult	tiplication o	verflow	detection c	lepends on S	SMUL.		
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.								
				nding is pe lied to the i		-	to RDM, if t	he opti	ional rnd	keyword is
							M40. If an status bit (A			ected, the
			Wh∉ SAT		low is de	tected, the	e accumulato	r is sa	turated ad	ccording to
		This instruction provides the option to store the 16-bit data memory ope Smem in temporary register T3.						ry operand		
		som and	ne C not	55x-based to externa	devices, I memor	the BB bu y. To prev	nd is access s is only con ent the gene on external r	necteo eration	d to intern of a bus	al memory
		Compatibility with C54x devices (C54CM = 1)								
		Whe	en th	nis instruction	on is exe	ecuted with	M40 = 0, co	mpatil	bility is er	sured.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL
	Affects	ACOVx
Repeat	This instruction	can be repeated.

Syntax				Description				
AC2 = rnd	(AC2 – (*AF	R1 * coef(*C		The content addressed by AR1 multiplied by the content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC2. The result is rounded and stored in AC2.				
Before			Afte	er				
AC2	00 EC0	0 0000	AC2	00 EC01 0000				
AR1		0302	AR2	0302				
CDP		0202	CDP	0202				
302		FE00	302	FEOO				
202		0040	202	0040				
ACOV2		0	ACO	V2 1				
SATD		0	SATI	0				
RDM		0	RDM	0				
FRCT		0	FRC	Г 0				

# Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline		
[3]	ACy = rnd(ACy	/ – <b>(</b> Si	mem * ACx <b>)</b>	[, T3 = Smem]		No	3	1	Х		
Opcod	e				1101	0010 AA	AA AA	AI U%D	DD 01SS		
Opera	nds	AC	x, ACy, Sm	em							
Descri	ption	The	This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the content of a memory (Smem) location, sign extended to 17 bits.								
			If FRCT = 1, the output of the multiplier is shifted left by 1 bit.								
			Multiplicat	ion overflow det	ection c	lepends on S	SMUL.				
The 32-bit result of the multiplication is sig subtracted from the source accumulator ACy						-	exten	ded to 4	0 bits and		
		Rounding is performed according to RDM, if the c applied to the instruction.						optional rnd keyword is			
				detection deper					ected, the		
			When an o SATD.	overflow is detec	ted, the	accumulato	or is sat	urated ac	cording to		
				n provides the o porary register T	•	store the 16	-bit da	ta memoi	ry operand		
		Compatibility with C54x devices (C54CM = 1)									
		Wh	en this inst	ruction is execu	ted with	M40 = 0, co	ompatik	oility is en	sured.		
Status	Bits	Aff	ected by	FRCT, M40, F	RDM, SA	ATD, SMUL					
		Aff	ects	ACOVy							
Repea	t	Thi	s instructio	n can be repeat	ed.						

Syntax	Description
AC0 = AC0 - (*AR3 * AC1)	The content addressed by AR3 multiplied by the content of AC1 is subtracted
	from the content of AC0 and the result is stored in AC0.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[4]	ACy = rnd(ACx	– <b>(</b> T)	x * Smem <b>))[</b> , <sup>-</sup>	Γ3 = Smem]		No	3	1	Х	
Opcod	e			:	1101	0101 AAA	AA AA	AI U%D	D ssSS	
Operar	nds	ACx, ACy, Smem, Tx								
Description		This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the content of a memory (Smem) location, sign extended to 17 bits.								
		□ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.								
		Multiplication overflow detection depends on SMUL.								
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.								
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.								
		Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.								
			When an o SATD.	verflow is detecte	ed, the	e accumulato	r is sat	urated ac	ccording to	
			This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.							
		Со	ompatibility	with C54x devic	es (C	54CM = 1)				
		Wł	nen this instr	uction is execute	d with	n M40 = 0, co	mpatik	oility is en	sured.	
Status	Bits	Aff	ected by	FRCT, M40, RD	M, SA	ATD, SMUL				
		Aff	ects	ACOVy						
Repeat	t	Th	is instructior	a can be repeated	I.					
Examp	le									

Syntax	Description
AC0 = AC1 - (T0 * *AR3)	The content addressed by AR3 multiplied by the content of T0 is subtracted from the content of AC1 and the result is stored in AC0.

Syntax					Parallel Enable Bit	Size	Cycles	Pipeline		
ACy = M40(rnd [, T3 = Xmem]	(ACx	– <b>(</b> uns(Xmem	ı) * uns( <b>`</b>	Ymem <b>))))</b>	No	4	1	Х		
9			1000	0110   XXXM	I MMYY YMI	MM SS	SDD 011	.g uuU%		
ds	AC	x, ACy, Xme	x, ACy, Xmem, Ymem							
otion	This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of data memory operand Ymem, extended to 17 bits.									
		Input opera	nds are	e extended to	17 bits accord	ding to	uns.			
				•		•	•	he content		
		-		-						
		If FRCT = 1	, the ou	utput of the mu	ultiplier is shif	ted left	t by 1 bit.			
	Multiplication overflow detection depends on SMUL.									
				•	-	exter	nded to 4	0 bits and		
		-	-		g to RDM, if t	he opti	ional rnd I	keyword is		
				•				ected, the		
		When an ov SATD.	verflow	is detected, th	e accumulato	or is sa	turated ac	ccording to		
			•		•					
			•	•	o store the 16	i-bit da	ta memo	ry operand		
	Со	mpatibility v	with C5	54x devices (	C54CM = 1)					
	Wh	ien this instru	uction is	s executed wit	h M40 = 0, co	ompatil	bility is en	sured.		
	ACy = M40(rnd	ACy = M40(rnd(ACx [, T3 = Xmem] e ads AC ption Thi The Xm ext	ACy = M40(rnd(ACx - (uns(Xmem [, T3 = Xmem]) e ads ACx, ACy, Xme ption This instruction The input opera Xmem, extended extended to 17 Input opera If the op of the n If the op of the n If FRCT = 1 Multiplication The 32-bit subtracted f Rounding is applied to th Overflow d destination When an ov SATD. This instruction the instruction, This instruction Xmem in tempor Compatibility f	ACy = M40(rnd(ACx - (uns(Xmem) * uns()         [, T3 = Xmem]         e       1000         ads       ACx, ACy, Xmem, Yme         ption       This instruction perform         The input operands of       Xmem, extended to 17         extended to 17 bits.       Input operands are         If the optional u       of the memory         If the optional u       of the memory         If the optional u       of the remory         The 32-bit result o       water the instruction accumu         Overflow detection destinatio	ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem))))         [, T3 = Xmem]         e         1000 0110   XXXM         nds       ACx, ACy, Xmem, Ymem         This instruction performs a multiplication       This instruction performs a multiplication         The input operands of the multiplier at Xmem, extended to 17 bits, and the origonal uns keyword is of the memory location is ze         Input operands are extended to         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is of the memory location is ze         If the optional uns keyword is on the source accure         Multiplication overflow detection         The 32-bit result of the multiplic subtracted from the source accure         Rounding is performed according applied to the instruction.         Overflow detection depends on destination accumulator overflow         When an overflow is detected, the SATD.         This instruct	ACy = M40(md(ACx - (uns(Xmem) * uns(Ymem))))       No         I, T3 = Xmem]       I 1000 0110   XXXM MMYY   YMI         adds       ACx, ACy, Xmem, Ymem         btion       This instruction performs a multiplication and a sub The input operands of the multiplier are the contern Xmem, extended to 17 bits, and the content of data extended to 17 bits.         Input operands are extended to 17 bits accord         If the optional uns keyword is applied to the of the memory location is zero extended t         If the optional uns keyword is not applied content of the memory location is sign exten SXMD.         If FRCT = 1, the output of the multiplier is shift         Multiplication overflow detection depends on S         The 32-bit result of the multiplication is sign subtracted from the source accumulator ACx.         Rounding is performed according to RDM, if the applied to the instruction.         Overflow detection depends on M40. If an destination accumulator overflow status bit (A         When an overflow is detected, the accumulator SATD.         This instruction provides the option to locally set M the instruction, if the optional M40 keyword is app This instruction provides the option to store the 16 Xmem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)	ACy = M40(md(ACx - (uns(Xmem) * uns(Ymem))))       No       4         [, T3 = Xmem]       1000       0110       XXXM       MMYY       YMMM       SS         adds       ACx, ACy, Xmem, Ymem       This instruction performs a multiplication and a subtraction       The input operands of the multiplier are the content of data memerextended to 17 bits, and the content of data memerxtended to 17 bits.       Input operands are extended to 17 bits according to         If the optional uns keyword is applied to the input of the memory location is zero extended to 17 bits.       If the optional uns keyword is not applied to the content of the memory location is sign extended to SXMD.         If FRCT = 1, the output of the multiplier is shifted left       Multiplication overflow detection depends on SMULL.         The 32-bit result of the multiplication is sign externabler to the instruction.       Overflow detection depends on M40. If an overflow destination accumulator overflow status bit (ACOVy)         When an overflow is detected, the accumulator is sa SATD.       This instruction provides the option to locally set M40 to 1 the instruction, if the optional M40 keyword is applied to This instruction provides the option to store the 16-bit da Xmem in temporary register T3.         Compatibility with C54x devices (C54CM = 1)       Actor Act	ACy = M40(md(ACx - (uns(Xmem) * uns(Ymem))))       No       4       1         I, T3 = Xmem]       I 1000       0110       XXXM       MMYY       YMMM       SSDD       0111         Ids       ACx, ACy, Xmem, Ymem       This instruction performs a multiplication and a subtraction in the D       The input operands of the multiplier are the content of data memoo Xmem, extended to 17 bits, and the content of data memory operate extended to 17 bits.       Input operands are extended to 17 bits according to uns.         If the optional uns keyword is applied to the input operand, 1 of the memory location is zero extended to 17 bits.       If the optional uns keyword is not applied to the input operands to the memory location is sign extended to 17 bits are SXMD.         If FRCT = 1, the output of the multiplier is shifted left by 1 bit.       Multiplication overflow detection depends on SMUL.         The 32-bit result of the multiplication is sign extended to 4 subtracted from the source accumulator ACx.       Rounding is performed according to RDM, if the optional rnd applied to the instruction.         Overflow detection depends on M40. If an overflow is det destination accumulator overflow status bit (ACOVy) is set.       When an overflow is detected, the accumulator is saturated are SATD.         This instruction provides the option to locally set M40 to 1 for the exit the instruction, if the optional M40 keyword is applied to the instruction, if the optional M40 keyword is applied to the instruction, in temporary register T3.		

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD	
	Affects	ACOVy	
Repeat	This instruction can be repeated.		

Syntax			Description		
AC3 = AC3	3 – (uns(*AR2+) * uns(		The unsigned content addressed by AR2 multiplied by the unsigned content addressed by AR3 is subtracted from the content of AC3 and the result is stored in AC3. AR2 and AR3 are both incremented by 1.		
Before		After	-		
AC3	00 2300 EC00	AC3	FF B3E0 EC00		
AR2	302	AR2	303		
AR3	202	AR3	203		
ACOV3	0	ACOV3	3 0		
302	FEOO	302	FE00		
202	7000	202	7000		
FRCT	0	FRCT	0		

### Multiply and Subtract with Parallel Load Accumulator from Memory

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACx = rnd(ACx - ACy = Ymem <<	-		No	4	1	Х
Opcode	)		1000 0110 XXXM	MMYY YMN	MM DI	DD 100	x ssU%
Operan	ds	AC	x, ACy, Tx, Xmem, Ymem				
Descrip	otion		s instruction performs two operat AS), and load.	ions in para	llel: m	ultiply an	d subtract
		MA to ´	e first operation performs a multipl C. The input operands of the multip I7 bits, and the content of data men bits.	lier are the co	ontent	of Tx, sigr	n extended
			If FRCT = 1, the output of the mul	ltiplier is shift	ted left	by 1 bit.	
			Multiplication overflow detection of	lepends on S	SMUL.		
			The 32-bit result of the multiplic subtracted from the source accum	-	exten	ded to 4	0 bits and
			Rounding is performed according applied to the instruction.	to RDM, if tl	he opti	onal rnd l	keyword is
			Overflow detection depends on destination accumulator overflow				ected, the
			When an overflow is detected, the SATD.	e accumulato	or is sat	turated ac	ccording to
			This instruction provides the op operand Xmem in temporary regis		the 1	6-bit dat	a memory
			e second operation loads the cor fted left by 16 bits to the accumula		memo	ory opera	ind Ymem
			The input operand is sign extended	ed to 40 bits	accord	ling to SX	MD.
			The shift operation is equivalent to	o the signed	shift ir	struction	
			The input operand is shifted left b	y 16 bits acc	ording	to M40.	
		Со	mpatibility with C54x devices (C	54CM = 1)			
		Wh	en this instruction is executed with	n M40 = 0, co	ompatil	oility is en	sured.

Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
Repeat	This instruction	n can be repeated.
See Also	See the follow	ing other related instructions:
	Modify Aux	xiliary Register Content with Parallel Multiply and Subtract
	Multiply ar	nd Accumulate with Parallel Load Accumulator from Memory
	Multiply ar	nd Subtract
	Multiply ar	nd Subtract with Parallel Multiply
	Multiply ar	nd Subtract with Parallel Multiply and Accumulate
	Multiply an	d Subtract with Parallel Store Accumulator Content to Memory
	Parallel M	ultiply and Subtracts
Example		

Syntax	Description
AC0 = AC0 - (T0 * *AR3), AC1 = *AR4 << #16	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is subtracted from the content of AC0 and the result is stored in AC0. The content addressed by AR4 shifted left by 16 bits is stored in AC1.

# Multiply and Subtract with Parallel Multiply

No.	Syntax	Parallel Enable Bit Size Cycles Pipeli	ine					
[1]		d(ACx - (uns(Xmem) * uns(coef(Cmem)))), No 4 1 X d(uns(Ymem) * uns(coef(Cmem))))						
Орсо	de	1000 0010 XXXM MMYY YMMM 10mm uuDD DD	g%					
Opera	ands	ACx, ACy, Cmem, Xmem, Ymem						
Description		This instruction performs two parallel operations in one cycle: multiply subtract (MAS), and multiply. The operations are executed in the two D-MACs.						
		The first operation performs a multiplication and a subtraction in the D-MAC. The input operands of the multiplier are the content of data men operand Xmem, extended to 17 bits, and the content of a data men operand Cmem, addressed using the coefficient addressing mode, exten to 17 bits.	nory nory					
		The second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.						
		Input operands are extended to 17 bits according to uns.						
		If the optional uns keyword is applied to the input operand, the con of the memory location is zero extended to 17 bits.	tent					
		If the optional uns keyword is not applied to the input operand, content of the memory location is sign extended to 17 bits accordin SXMD.						
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.						
		Multiplication overflow detection depends on SMUL.						
		For the first operation, the 32-bit result of the multiplication is extended to 40 bits and subtracted from the source accumulator AC	-					
		For the second operation, the 32-bit result of the multiplication is extended to 40 bits.	sign					
		Rounding is performed according to RDM, if the optional rnd keywor applied to the instruction.	rd is					

	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
	When an overflow is detected, the accumulator is saturated according to SATD.
	This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.
	For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.
	Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:
	■ mar(Xmem)
	■ mar(Ymem)
	■ mar(Cmem)
Status Bits	Affected by FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects ACOVx, ACOVy
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Modify Auxiliary Register Content with Parallel Multiply and Subtract
	Multiply and Accumulate with Parallel Multiply
	Multiply and Subtract
	Multiply and Subtract with Parallel Load Accumulator from Memory
	Multiply and Subtract with Parallel Multiply and Accumulate
	Multiply and Subtract with Parallel Store Accumulator Content to Memory
	Parallel Multiply and Subtracts

Syntax	Description
AC0 = AC0 - (uns(*AR3) * uns(coef(*CDP))), AC1 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0 and the result is stored in AC0. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by CDP and the result is stored in AC1.

## Multiply and Subtract with Parallel Multiply and Accumulate

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]			· · · · · · · · · · · · · · · · · · ·	f uns(coef(Cmem))))), f uns(coef(Cmem)))))	No	4	1	Х		
[2]	ACx = M40(rnd(A ACy = M40(rnd(( uns(coef(Cmem)	ACy		f <mark>uns(<b>coef(</b>Cmem<b>))</b>)), (Ymem) *</mark>	No	4	1	x		
Descr	iption	sul	btract (MAS),	ns perform two paralle and multiply and ac two D-unit MACs.	•		•			
Status	s Bits	Aff	Affected by FRCT, M40, RDM, SATD, SMUL, SXMD							
		Affects ACOVx, ACOVy								
See A	lso	See the following other related instructions:								
			Modify Auxil	iary Register Content	with Parallel	Multip	ly and S	ubtract		
			Multiply and	Subtract						
			Multiply and	Subtract with Paralle	Load Accur	nulator	from Me	emory		
		Multiply and Subtract with Parallel Multiply								
			Multiply and	Subtract with Parallel	Store Accum	nulator	Content	to Memory		
		Parallel Multiply and Subtracts								

#### Multiply and Subtract with Parallel Multiply and Accumulate

#### Svntax Characteristics Parallel Enable Bit Size No. Syntax Cycles Pipeline ACx = M40(rnd(ACx - (uns(Xmem) \* uns(coef(Cmem))))), No 1 Х [1] 4 ACy = M40(rnd(ACy + (uns(Ymem) \* uns(coef(Cmem))))) 1000 0011 XXXM MMYY YMMM 01mm uuDD DDg% Opcode Operands ACx, ACy, Cmem, Xmem, Ymem Description This instruction performs two parallel operations in one cycle: multiply and subtract (MAS), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs. The first operation performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits. The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits. Input operands are extended to 17 bits according to uns. ■ If the optional uns keyword is applied to the input operand, the content of the memory location is zero extended to 17 bits. If the optional uns keyword is not applied to the input operand, the content of the memory location is sign extended to 17 bits according to SXMD. □ If FRCT = 1, the output of the multiplier is shifted left by 1 bit. Multiplication overflow detection depends on SMUL. For the first operation, the 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx. For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy. Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.

		detection depends on M40. If an overflow is detected, the n accumulator overflow status bit is set.
	When an o SATD.	overflow is detected, the accumulator is saturated according to
		n provides the option to locally set M40 to 1 for the execution of , if the optional M40 keyword is applied to the instruction.
	some C55x-ba and not to ext	ction, the Cmem operand is accessed through the BB bus; on ased devices, the BB bus is only connected to internal memory ternal memory. To prevent the generation of a bus error, the d must not be mapped on external memory.
	while allowing	w can also disable the usage of the corresponding MAC unit, the modification of auxiliary registers in the three address its through the following instructions:
	■ mar(X	mem)
	■ mar(Y	mem)
	■ mar(C	
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
Repeat	This instruction	n can be repeated.
		•

## Example

AC0 = M40(rnd(AC0 - (uns(*AR0) * uns(coef(*CDP))))),       Both instructions are performed in parallel. The unsigned content addressed by AR0 multiplied by the unsigned content addressed by the coefficien data pointer register (CDP) is subtracted from the content of AC0. The result is rounded and stored in AC0. The unsigned content addressed by AR1 multiplied by the unsigned content addressed by CDP is added to the content of AC1. The result is rounded and stored in AC1.         Before       After         AC0       00 6900 0000       AC0       00 486B 0000         AC1       00 0023 0000       AC1       00 95E3 0000         *AR0       3400       *AR0       3400         *AR1       EF00       *AR1       EF00         *CDP       A067       ACOV1       0         ACOV1       0       ACOV1       0         CARRY       0       CARRY       0	Syntax						D	Description	
AC0       00       6900       0000       AC0       00       486B       0000         AC1       00       0023       0000       AC1       00       95E3       0000         *AR0       3400       *AR0       3400         *AR1       EF00       *AR1       EF00         *CDP       A067       *CDP       A067         ACOV0       0       ACOV1       0         ACOV1       0       CARRY       0					, , ,		tt d c A n C	unsigned content addressed by AR0 multi the unsigned content addressed by the co data pointer register (CDP) is subtracted to content of AC0. The result is rounded and s AC0. The unsigned content addressed multiplied by the unsigned content addre CDP is added to the content of AC1. The	iplied by befficient from the stored in by AR1 ssed by
AC1       00       0023       0000       AC1       00       95E3       0000         *AR0       3400       *AR0       3400         *AR1       EF00       *AR1       EF00         *CDP       A067       *CDP       A067         ACOV0       0       ACOV1       0         ACOV1       0       ACOV1       0         CARRY       0       CARRY       0	Before				After				
*AR0       3400       *AR0       3400         *AR1       EF00       *AR1       EF00         *CDP       A067       *CDP       A067         ACOV0       0       ACOV1       0         ACOV1       0       ACOV1       0         CARRY       0       CARRY       0	AC0	00	6900	0000	AC0	00	4861	B 0000	
*AR1EF00*AR1EF00*CDPA067*CDPA067ACOV00ACOV00ACOV10ACOV10CARRY0CARRY0	AC1	00	0023	0000	AC1	00	95E3	3 0000	
*CDPA067*CDPA067ACOV00ACOV00ACOV10ACOV10CARRY0CARRY0	*ARO			3400	*AR0			3400	
ACOV0         0         ACOV0         0           ACOV1         0         ACOV1         0           CARRY         0         CARRY         0	*AR1			EF00	*AR1			EF00	
ACOV10ACOV10CARRY0CARRY0	*CDP			A067	*CDP			A067	
CARRY 0 CARRY 0	ACOV0			0	ACOV0			0	
	ACOV1			0	ACOV1			0	
FRCT 0 FRCT 0	CARRY			0	CARRY			0	
	FRCT			0	FRCT			0	

5-308 Instruction Set Descriptions

# Multiply and Subtract with Parallel Multiply and Accumulate

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[2]		ACy :	•	No	4	1	Х	
Орсо	de		1000 0100 XXXM	MMYY YMN	4M 00	mm uuD	D DDg%	
Opera	Inds	AC	x, ACy, Cmem, Xmem, Ymem					
Descr	iption	sub	s instruction performs two parallel ptract (MAS), and multiply and ac ecuted in the two D-unit MACs.	•		•		
		MA ope ope	e first operation performs a multipl C. The input operands of the mult erand Xmem, extended to 17 bits erand Cmem, addressed using the o 17 bits.	tiplier are the , and the co	e conte ontent	ent of dat of a dat	ta memory a memory	
		D-u me me	unit MAC. The input operands of mory operand Ymem, extended t	cond operation performs a multiplication and an accumulation in the MAC. The input operands of the multiplier are the content of data y operand Ymem, extended to 17 bits, and the content of a data y operand Cmem, addressed using the coefficient addressing mode, ed to 17 bits.				
			Input operands are extended to 1	7 bits accord	ling to	uns.		
			If the optional uns keyword is a of the memory location is zero		•	•	he content	
			If the optional uns keyword is content of the memory location SXMD.			• •		
			If FRCT = 1, the output of the mul	tiplier is shift	ed left	by 1 bit.		
			Multiplication overflow detection d	lepends on S	SMUL.			
			For the first operation, the 32-b extended to 40 bits and subtracted					
			For the second operation, the 32 extended to 40 bits and added to th by 16 bits. The shifting operation source accumulator ACy(39).	ne source acc	cumula	tor ACy s	hifted right	

		•	s performed a ne instruction	•	o RDM,	if the op	otional	rnd key	/word is
			etection depe accumulator				flow is	s detec	ted, the
		When an o <sup>v</sup> SATD.	verflow is dete	ected, the a	accumu	ator is s	aturat	ed acco	ording to
			provides the o if the optional	•	-				
	som and	ne C55x-bas not to exte	ion, the Cme ed devices, th rnal memory. must not be	ne BB bus To preve	is only only only of the g	connect eneratio	ed to ii on of a	nternal	memory
	whi	le allowing	can also disa the modificat s through the	ion of aux	ciliary re	gisters	•	•	
		■ mar(Xn	nem)						
		■ mar(Yn	nem)						
		■ mar(Cr	nem)						
Status Bits	Affe	cted by	FRCT, M40,	RDM, SAT	ΓD, SMI	JL, SXM	1D		
	Affe	ects	ACOVx, ACO	⊃Vy					
Repeat	This	s instruction	can be repea	ited.					

Syntax	Description
AC0 = AC0 - (uns(*AR3) * uns(coef(*CDP))), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0 and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 shifted right by 16 bits and the result is stored in AC1.

Multiply and Subtract with Parallel Store Accumulator Content to

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	ACy = rnd(ACy Ymem = <b>HI(</b> AC)		κ * Xmem <b>)</b> ), <b>T2) [,</b> T3 = Xmem]	No	4	1	Х		
Opcode	e		1000 0111 XXXM	MMYY YMI	MM SS	SDD 010	x ssU%		
Operan	nds	AC	x, ACy, Tx, Xmem, Ymem						
Descrip	otion		is instruction performs two operat AS), and store.	ions in para	illel: m	ultiply an	d subtract		
		MA to <sup>2</sup>	e first operation performs a multipl C. The input operands of the multip 17 bits, and the content of data men bits.	lier are the c	ontent	of Tx, sigr	n extended		
			If FRCT = 1, the output of the mul	ltiplier is shif	ted left	by 1 bit.			
			Multiplication overflow detection of	lepends on S	pends on SMUL.				
		☐ The 32-bit result of the multiplication is sign extended to 40 bits subtracted from the source accumulator ACy.					0 bits and		
		Rounding is performed according to RDM, if the optional rnd keyw applied to the instruction.							
		Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
			When an overflow is detected, the SATD.	e accumulato	or is sat	turated ad	ccording to		
			This instruction provides the op operand Xmem in temporary regis		e the 1	6-bit dat	a memory		
		sto is r	e second operation shifts the accurres ACx(31–16) to data memory operation within –32 to +31, the shift is sufformed with this value.	perand Yme	m. If th	e 16-bit v	alue in T2/		
			The input operand is shifted in the	e D-unit shift	er acco	ording to	SXMD.		
			After the shift, the high part of the the memory location.	accumulato	r, ACx(	(31–16), i	s stored to		

## Compatibility with C54x devices (C54CM = 1)

	When this instruction is executed with M40 = 0, compatibility is ensured. When this instruction is executed with C54CM = 1, the 6 LSBs of T2 are used to determine the shift quantity. The 6 LSBs of T2 define a shift quantity within $-32$ to $+31$ . When the 16-bit value in T2 is between $-32$ to $-17$ , a modulo 16 operation transforms the shift quantity to within $-16$ to $-1$ .
Status Bits	Affected by C54CM, FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects ACOVy
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Modify Auxiliary Register Content with Parallel Multiply and Subtract
	Multiply and Accumulate with Parallel Store Accumulator Content to Memory
	Multiply and Subtract
	Multiply and Subtract with Parallel Load Accumulator from Memory
	Multiply and Subtract with Parallel Multiply
	Multiply and Subtract with Parallel Multiply and Accumulate
	Parallel Multiply and Subtracts

Syntax	Description
AC0 = AC0 - (T0 * *AR3), *AR4 = HI(AC1 << T2)	Both instructions are performed in parallel. The content addressed by AR3 multiplied by the content of T0 is subtracted from the content of AC0 and the result is stored in AC0. The content of AC1 is shifted by the content of T2, and AC1(31–16) is stored at the address of AR4.

## Negate Accumulator, Auxiliary, or Temporary Register Content

#### Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = - src					Yes	2	1	Х
Opcod	e					00	11 01	LOE FSS	SS FDDD
Opera	nds	dst,	src						
Descri	ption	regi	ister (	src). This ins	outes the 2s contraction clears quals 0, the CA	the CARRY st	tatus bi	it to 0 for a	
			Whe	n the destina	ation operand (	dst) is an accu	umulate	or:	
			T	he operation	n is performed	on 40 bits in t	he D-u	nit ALU.	
			∎ Ir	nput operand	ds are sign ext	ended to 40 bi	its acco	ording to	SXMD.
			ir	nstruction, th	or temporary in the second sec	ne auxiliary or t		•	. ,
				Overflow dete	ection and CAI	RRY status bit	depen	ds on M4	40.
				Vhen an ove o SATD.	rflow is detecte	d, the accumu	lator is	saturated	laccording
			Whe	n the destina	ation operand (	dst) is an auxi	liary or	tempora	ry register:
			T	he operation	n is performed	on 16 bits in t	he A-u	nit ALU.	
					lator is the some accumulator	•	. ,		
				Overflow dete	ection is done	at bit position	15.		
				Vhen an ove	erflow is detec SATA.	ted, the destin	nation r	egister is	s saturated
		Cor	mpati	bility with C	C54x devices (	(C54CM = 1)			
		14/1	4b '						

When this instruction is executed with M40 = 0, compatibility is ensured.

#### Negate Accumulator, Auxiliary, or Temporary Register Content

Status Bits	Affected by M40, SATA, SATD, SXMD
	Affects ACOVx, CARRY
Repeat	This instruction can be repeated.
See Also	See the following other related instructions:
	Complement Accumulator, Auxiliary, or Temporary Register Bit
	Complement Accumulator, Auxiliary, or Temporary Register Content
Example	
Syntax	Description
AC0 = -AC1	The 2s complement of the content of AC1 is stored in AC0.

# No Operation (nop)

					Parallel			
No.	Syntax				Enable Bit	Size	Cycles	Pipeline
[1]	nop				Yes	1	1	D
[2]	nop_16				Yes	2	1	D
Opcod	e						001	.0 000E
Operai	nds	none						
Descri	ption	Instruction [1] Instruction [2]	-			registe	er (PC) I	oy 1 byte.
Status	Bits	Affected by	none					
		Affects	none					
Repea	t	This instruction	n can be r	epeated.				
Examp	ole							
Syntax	K	Description						

Cymax	
nop	The program counter (PC) is incremented by 1 byte.

# Parallel Modify Auxiliary Register Contents

## **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	mar(Xmem),	nar(Ymem), mar(coef(Cmem))	No	4	1	Х			
Opcode	9	1000 01	01 XXXM MMYY YM	MM 10	mm xxx	x xxxx			
Operan	ds	Cmem, Xmem, Ymem							
Descrip	otion	This instruction performs three parallel modify auxiliary register (MAR) operations in one cycle. The auxiliary register modification is specified by:							
		the content of data memory operand Xmem							
		the content of data memory operand Ymem							
		—	□ the content of a data memory operand Cmem, addressed using the coefficient addressing mode						
Status	Bits	Affected by none							
		Affects none							
Repeat		This instruction can be rep	eated.						
See Als	60	See the following other related instructions:							
		Modify Auxiliary Regis	ter Content						
		Modify Extended Auxiliary Register Content							

Syntax	Description
	AR3 is incremented by 1. AR4 is decremented by 1. CDP is not modified.

Parallel Multiplies

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]			Xmem) * <mark>uns(coef(</mark> Cmem <b>)</b> ))), Ymem) * <mark>uns(coef(</mark> Cmem <b>)</b> )))	No	4	1	Х		
Opcod	e		1000 0010 XX	XM MMYY YMM	4M 00	mm uuD	DD DDg%		
Operar	nds	AC	x, ACy, Cmem, Xmem, Ymem						
Descri	ption		s instruction performs two para erations are executed in the two		eration	s in one	cycle. The		
		ope exte	The first operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.						
		ope exte	s second operation performs a perands of the multiplier are the ended to 17 bits, and the condressed using the coefficient ad	content of data itent of a data	memo memo	ory opera ry opera	nd Ymem nd Cmem		
			Input operands are extended to 17 bits according to uns.						
			If the optional uns keyword of the memory location is :		•	•	he conten		
			If the optional uns keywor content of the memory loca SXMD.			• •			
			If FRCT = 1, the output of the	multiplier is shift	ed left	by 1 bit.			
			Multiplication overflow detection	on depends on S	SMUL.				
			The 32-bit result of the multipli	ication is sign ex	ktende	d to 40 bi	ts.		
			Rounding is performed accord applied to the instruction.	ling to RDM, if t	he opti	onal rnd I	keyword is		
			Overflow detection depends destination accumulator overfl			w is det	ected, the		
			When an overflow is detected, SATD.	the accumulato	r is sat	urated ac	ccording to		

	This instruction provides the option to locally set M40 to 1 for the execution the instruction, if the optional M40 keyword is applied to the instruction.						
	For this instruction, the Cmem operand is accessed through the BB bus; or some C55x-based devices, the BB bus is only connected to internal memori and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.						
	Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:						
	■ mar(Xmem)						
	■ mar(Ymem)						
	■ mar(Cmem)						
Status Bits	Affected by FRCT, M40, RDM, SATD, SMUL, SXMD						
	Affects ACOVx, ACOVy						
Repeat	This instruction can be repeated.						
See Also	See the following other related instructions:						
	Modify Auxiliary Register Content with Parallel Multiply						
	Multiply						
	Multiply and Accumulate with Parallel Multiply						
	Multiply and Subtract with Parallel Multiply						
	Parallel Multiply and Accumulates						
	Parallel Multiply and Subtracts						

Syntax	Description
AC0 = uns(*AR3) * uns(coef(*CDP)), AC1 = uns(*AR4) * uns(coef(*CDP))	Both instructions are performed in parallel. The unsigned content addressed by AR3 is multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) and the result is stored in AC0. The unsigned content addressed by AR4 is multiplied by the unsigned content addressed by CDP and the result is stored in AC1.

# Parallel Multiply and Accumulates

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1]				* uns(coef(Cmem)))), * uns(coef(Cmem)))))	No	4	1	Х
[2] ACx = M40(rnd((ACx >> #16) + (uns uns(coef(Cmem)))), ACy = M4(rnd(ACy + (uns(Ymem) * 1)))					No	4	1	х
[3]	ACx = M40(rnd(( uns(coef(Cmem) ACy = M40(rnd(( uns(coef(Cmem)	) <mark>)))</mark> , ACy :			No	4	1	Х
Descr	iption			ions perform two par ne cycle. The operatior				. ,
Status	s Bits	Aff	ected by	FRCT, M40, RDM, S	ATD, SMUL,	SXMD		
		Aff	ects	ACOVx, ACOVy				
See A	lso	Se	See the following other related instructions:					
			Modify Auxiliary Register Content with Parallel Multiply and Accumulate					
			Multiply and Accumulate					
			Multiply and Accumulate with Parallel Delay					
			Multiply and Accumulate with Parallel Load Accumulator from Memory					Memory
			Multiply and Accumulate with Parallel Multiply					
			Multiply and Accumulate with Parallel Store Accumulator Content to Memor					
			Multiply and Subtract with Parallel Multiply and Accumulate					
			Multiply with Parallel Multiply and Accumulate					
			Parallel Multiplies					
			Parallel Mu	Iltiply and Subtracts				

## Parallel Multiply and Accumulates

No.	Syntax	Parallel Enable Bit Size Cycles Pipelin
[1]	ACx = M40(rnc	ACx + (uns(Xmem) * uns(coef(Cmem))))), No 4 1 X ACy + (uns(Ymem) * uns(coef(Cmem)))))
Орсо	de	1000 0011 XXXM MMYY YMMM 00mm uuDD DDg
Opera	ands	ACx, ACy, Cmem, Xmem, Ymem
Descr	iption	This instruction performs two parallel multiply and accumulate (MA operations in one cycle. The operations are executed in the two D-unit MAC
		The first operation performs a multiplication and an accumulation in the D-u MAC. The input operands of the multiplier are the content of data memory operand Xmem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, extended to 17 bits.
		The second operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode extended to 17 bits.
		Input operands are extended to 17 bits according to uns.
		If the optional uns keyword is applied to the input operand, the conte of the memory location is zero extended to 17 bits.
		If the optional uns keyword is not applied to the input operand, to content of the memory location is sign extended to 17 bits according SXMD.
		☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.
		Multiplication overflow detection depends on SMUL.
		The 32-bit result of the multiplication is sign extended to 40 bits and add to the source accumulator.
		Rounding is performed according to RDM, if the optional rnd keyword applied to the instruction.
		Overflow detection depends on M40. If an overflow is detected, t destination accumulator overflow status bit is set.
		When an overflow is detected, the accumulator is saturated according SATD.

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

	■ mar(Xr	mem)				
	■ mar(Ymem)					
	■ mar(Ci	mem)				
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD				
	Affects	ACOVx, ACOVy				
Repeat	This instructior	a can be repeated.				

Syntax	Description
AC0 = AC0 + (uns(*AR3) * uns(coef(*CDP))), AC1 = AC1 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 and the result is stored in AC1.

## Parallel Multiply and Accumulates

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2]	uns(coef(Cme	< >> #16) + (uns(Xmer , - (uns(Ymem) * uns(co		No 4 1 X				
Opcod	e		1000	0011 XXXM	MMYY YMM	4M 10	mm uuD	D DDg%
Opera	nds	AC	x, ACy, Cmem, Xme	em, Ymem				
Descri	ption		s instruction perfo erations in one cycle	•				. ,
		MA ope ope	e first operation perfo C. The input opera erand Xmem, exter erand Cmem, addre 17 bits.	nds of the mult ided to 17 bits	tiplier are the , and the c	e conte ontent	ent of dat of a dat	a memory a memory
		D-u me me	e second operation unit MAC. The inpu mory operand Yme mory operand Cme ended to 17 bits.	t operands of em, extended t	the multiplie o 17 bits, a	r are t nd the	the conte	ent of data of a data
			Input operands are	e extended to 1	7 bits accord	ling to	uns.	
			•	ns keyword is a location is zero		•	•	he content
			If the optional content of the n SXMD.	uns keyword is nemory locatior				
			If FRCT = 1, the ou	utput of the mul	tiplier is shif	ed left	by 1 bit.	
			Multiplication overf	low detection d	lepends on S	SMUL.		
			For the first operative extended to 40 bits by 16 bits. The sh source accumulate	and added to th	ne source aco	cumula	tor ACx s	hifted right
			For the second op extended to 40 bits				-	-

		•	s performed according to RDM, if the optional rnd keyword is the instruction.		
			letection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.		
		When an o SATD.	verflow is detected, the accumulator is saturated according to		
	This instruction provides the option to locally set M40 to 1 for the ex the instruction, if the optional M40 keyword is applied to the instruc				
	For this instruction, the Cmem operand is accessed through the BB bus; or some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.				
	Each data flow can also disable the usage of the corresponding MAC un while allowing the modification of auxiliary registers in the three addres generation units through the following instructions:				
		■ mar(Xr	nem)		
		■ mar(Yr	nem)		
		■ mar(Cr	nem)		
Status Bits	Affe	ected by	FRCT, M40, RDM, SATD, SMUL, SXMD		
	Affe	ects	ACOVx, ACOVy		
Repeat	This instruction can be repeated.				

Syntax	Description
AC0 = (AC0 >> #16) + (uns(*AR3) * uns(coef(*CDP))), AC1 = AC1 + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 shifted right by 16 bits and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 and the result is stored in AC1.

## Parallel Multiply and Accumulates

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[3]	uns(coef(Cmem	1 <b>)))))</b> (ACy	,	<b>#16) + (uns</b> (Xmem) * <b>#16) + (uns</b> (Ymem) *	No	Х				
Opcod	e			1000 0100 XXXM	I MMYY YMI	MM 11	Lmm uuI	DD DDg%		
Operar	nds	AC	x, A	Cy, Cmem, Xmem, Ymem						
Descri	ption			nstruction performs two para ons in one cycle. The operation				. ,		
		MA ope ope	∖C. <sup>–</sup> eran	et operation performs a multipli The input operands of the mu d Xmem, extended to 17 bit d Cmem, addressed using the its.	Itiplier are the states and the c	e content	ent of dat of a dat	ta memory a memory		
		The second operation performs a multiplication and an accumulat D-unit MAC. The input operands of the multiplier are the conter memory operand Ymem, extended to 17 bits, and the content memory operand Cmem, addressed using the coefficient address extended to 17 bits.						ent of data of a data		
			Inp	ut operands are extended to	17 bits accord	ding to	uns.			
				If the optional uns keyword is of the memory location is zero	••	•	•	he content		
			•	If the optional uns keyword i content of the memory location SXMD.						
			lf F	RCT = 1, the output of the mu	ultiplier is shif	ted left	by 1 bit.			
		Multiplication overflow detection depends on SMUI						JL.		
			to t	e 32-bit result of the multiplication the source accumulator shifted performed with a sign extension	d right by 16 l	bits. Th	ne shifting	operation		
				unding is performed according blied to the instruction.	g to RDM, if t	he opti	ional rnd	keyword is		

	_	detection depends on M40. If an overflow is detected, the accumulator overflow status bit is set.
	When an o SATD.	overflow is detected, the accumulator is saturated according to
		n provides the option to locally set M40 to 1 for the execution of , if the optional M40 keyword is applied to the instruction.
	some C55x-ba and not to ext	ction, the Cmem operand is accessed through the BB bus; on used devices, the BB bus is only connected to internal memory ternal memory. To prevent the generation of a bus error, the d must not be mapped on external memory.
	while allowing	v can also disable the usage of the corresponding MAC unit, the modification of auxiliary registers in the three address ts through the following instructions:
	■ mar(X	mem)
	■ mar(Y	mem)
	■ mar(C	mem)
Status Bits	Affected by	FRCT, M40, RDM, SATD, SMUL, SXMD
	Affects	ACOVx, ACOVy
Repeat	This instruction	n can be repeated.

Syntax	Description
AC0 = (AC0 >> #16) + (uns(*AR3) * uns(coef(*CDP))), AC1 = (AC1 >> #16) + (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is added to the content of AC0 shifted right by 16 bits and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is added to the content of AC1 shifted right by 16 bits and the result is stored in AC1.

# Parallel Multiply and Subtracts

	0			Parallel	0.	<u> </u>	<b>D</b> ' ''
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[1]			(uns(Xmem) * uns(coef(Cmem))) (uns(Ymem) * uns(coef(Cmem)))		4	1	Х
Орсо	de		1000 0101 XX	ХМ ММҮҮ ҮМ	MM 01	.mm uuI	DD DDg%
Opera	ands	AC	, ACy, Cmem, Xmem, Ymem				
Descr	ription		s instruction performs two para ne cycle. The operations are e			· ,	•
		MA ope ope	first operation performs a mu C. The input operands of the rand Xmem, extended to 17 rand Cmem, addressed using 7 bits.	multiplier are th bits, and the c	e conte	ent of dat of a dat	ta memory a memory
		MA ope ope	second operation performs an C. The input operands of the rand Ymem, extended to 17 rand Cmem, addressed using 7 bits.	multiplier are th bits, and the c	e conte	ent of dat of a dat	ta memory a memory
			Input operands are extended	o 17 bits accore	ding to	uns.	
			If the optional uns keyword of the memory location is		•	•	the conten
			If the optional uns keywork content of the memory local SXMD.				
			If FRCT = 1, the output of the	multiplier is shif	ted left	by 1 bit.	
			Multiplication overflow detection	on depends on	SMUL.		
			The 32-bit result of the multi subtracted from the source ac	-	n exter	ded to 4	0 bits and
			Rounding is performed accord applied to the instruction.	ling to RDM, if t	he opti	onal rnd	keyword i
			Overflow detection depends destination accumulator overfl			ow is det	tected, the
			When an overflow is detected, SATD.	the accumulate	or is sa	turated a	ccording to

This instruction provides the option to locally set M40 to 1 for the execution of the instruction, if the optional M40 keyword is applied to the instruction.

For this instruction, the Cmem operand is accessed through the BB bus; on some C55x-based devices, the BB bus is only connected to internal memory and not to external memory. To prevent the generation of a bus error, the Cmem operand must not be mapped on external memory.

Each data flow can also disable the usage of the corresponding MAC unit, while allowing the modification of auxiliary registers in the three address generation units through the following instructions:

- mar(Xmem)
- mar(Ymem)
- mar(Cmem)

Status Bits Affected by FRCT, M40, RDM, SATD, SMUL, SXMD

Affects ACOVx, ACOVy

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

Modify Auxiliary Register Content with Parallel Multiply and Subtract

- Multiply and Subtract
- Multiply and Subtract with Parallel Load Accumulator from Memory
- Multiply and Subtract with Parallel Multiply
- Multiply and Subtract with Parallel Multiply and Accumulate
- Multiply and Subtract with Parallel Store Accumulator Content to Memory
- Parallel Multiplies
- Parallel Multiply and Accumulates

Syntax	Description
AC0 = AC0 - (uns(*AR3) * uns(coef(*CDP))), AC1 = AC1 - (uns(*AR4) * uns(coef(*CDP)))	Both instructions are performed in parallel. The unsigned content addressed by AR3 multiplied by the unsigned content addressed by the coefficient data pointer register (CDP) is subtracted from the content of AC0 and the result is stored in AC0. The unsigned content addressed by AR4 multiplied by the unsigned content addressed by CDP is subtracted from the content of AC1 and the result is stored in AC1.

## Peripheral Port Register Access Qualifiers

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	readport()			No	1	1	D	
[2]	writeport()			No	1	1	D	
Opcod	e	ı	readport			100	1 1001	
		W	riteport			100	1 1010	
Operar	nds	no	ne					
Descrij	ption	me	ese operand qualifiers allow you emory and enable access to the 6 specified by the Smem, Xmem, o	64K-word I/O sp	ace. T			
			A readport() operand qualifier r a word single data memory acc operation, except instructions u	ess Smem or X	-		-	
		A writeport() operand qualifier may be included in any instruction making a word single data memory access Smem or Ymem that is used in a write operation, except instructions using the delay().						
		A readport() or writeport() operand qualifier cannot be used as a stand-alone instruction (the assembler generates an error message).						
		thc 64 *pc ext	y instruction making a word sin use listed above) can use the *p K-word I/O space with an imme prt(#k16), the 16-bit unsigned tension to the instruction. Becau prt(#k16) cannot be executed in	ort(#k16) addre diate address. constant, k16, use of the exter	ssing r When is en nsion, a	mode to a an instruc coded in an instruc	access the ction uses a 2-byte ction using	
		ins ins	e following indirect operands car truction using one of these ope truction. Because of the extensic lirect operands cannot be execu	erands requires	a 2-by nusing	te extens	sion to the e following	
			*ARn(#K16)					
			*+ARn(#K16)					
			*CDP(#K16)					
			*+CDP(#K16)					

Peripheral Port Register Access Qualifiers (readport/writeport)

Syntax	Description	
Example 1		
Repeat	An instruction	using this operand qualifier can be repeated.
	Affects	none
Status Bits	Affected by	none

Syntax	Description
T2 = *AR3	The content addressed by AR3 (I/O address) is loaded into T2.
readport()	

Syntax	Description
*AR3 = T2    writeport()	The content of T2 is written to the location addressed by AR3 (I/O address).

Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	xdst = <b>popbot</b>	h()	Yes	2	1	Х			
Opcod	e		010	01 00	0E XDD	D 0100			
Operar	nds	xdst							
Descri	ption	This instruction moves the content of two 16-bit data memory locations addressed by the data stack pointer (SP) and system stack pointer (SSP) to accumulator ACx or to the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).							
		The content of xdst(15 $-0$ ) is loaded from the location addressed by SP and the content of xdst(31 $-16$ ) is loaded from the location addressed by SSP.							
		When xdst is a 23-bit register, the upp by SSP are discarded and only the 7 I into the high part of xdst(22–16).							
		When xdst is an accumulator, the (unchanged) with the current value a	-	•	,				
Status	Bits	Affected by none							
		Affects none							
Repeat	t	This instruction can be repeated.							
See Al	SO	See the following other related instru	ctions:						
		Pop Top of Stack							
		Push to Top of Stack							
		Push Accumulator or Extended Au	uxiliary Registe	er Conte	ent to Sta	ck Pointers			

#### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst1, dst2 = <b>po</b>	p()	Yes	2	1	Х
[2]	dst = <b>pop()</b>		Yes	2	1	х
[3]	dst, Smem = <b>p</b> e	op()	No	3	1	Х
[4]	ACx = dbl(pop	())	Yes	2	1	х
[5]	Smem = <b>pop()</b>		No	2	1	х
[6]	dbl(Lmem) = p	ор()	No	2	1	Х
by the data stack pointer (SP		These instructions move the content by the data stack pointer (SP) to: an accumulator, auxiliary, or temp a data memory location			location	addressed
		When the destination register is an		-	•	

16 higher bits of the accumulator, ACx(39-16), are reloaded (unchanged) with the current value and are not modified by these instructions.

The increment operation performed on SP is done by the A-unit address generator dedicated to the stack addressing management.

## Status Bits Affected by none

Affects none

**See Also** See the following other related instructions:

- Dep Accumulator or Extended Auxiliary Register Content from Stack Pointers
- Push to Top of Stack
- Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

## Syntax Characteristics

Na	Symbol			Parallel	Sina	Cycles	Dinalina	
<u>No.</u> [1]	Syntax dst1, dst2 = p	op()		Enable Bit Yes	<b>Size</b> 2	Cycles	Pipeline X	
Opcod			dst1, FDDD = dst2	003	L1 10	)1E  FSS	S FDDD	
Operar	nds	dst1, dst2						
by SP to destination register de				ent of the 16-bit data memory location pointed st1 and moves the content of the 16-bit data ? + 1 to destination register dst2.				
When the destination register, dst1 or dst2, is an accumu the 16-bit data memory operand is moved to the destinat part, ACx(15–0). The guard bits and the 16 higher bits ACx(39–16), are reloaded (unchanged) with the curren modified by this instruction. SP is incremented by 2.					estinati er bits curren	ion accun of the ac	nulator low cumulator,	
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructio	n can be repeated.					

Syntax	Description
AC0, AC1 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to $AC0(15-0)$ and the content of the memory location pointed by SP + 1 is copied to $AC1(15-0)$ . Bits 39–16 of the accumulators are unchanged. The SP is incremented by 2.

Before			After			
AC0	00 4500	0000	AC0	00	4500	4890
AC1	F7 5678	3 9432	AC1	F7	5678	2300
SP		0300	SP			0302
300		4890	300			4890
301		2300	301			2300

#### **Syntax Characteristics**

No. Syntax			Paralle Enable I	-	Cycles	Pipeline	
[2] dst = <b>pop()</b>			Yes	2	1	Х	
Opcode				0101 0	00e   FDI	DD x010	
Operands dst							
Description	This instruction moves the content of the 16-bit data memory location pointed by SP to destination register dst.						
When the destination re data memory operand ACx(15–0). The guard ACx(39–16), are reload modified by this instruct			red to the destin and the 16 high changed) with th	nation ac er bits ne currer	cumulato of the ac	r low part, cumulator,	
Status Bits	Affected by	none					
	Affects	none					
Repeat	This instructior	n can be repeate	d.				
Example							
Syntax	Description						

The content of the memory location pointed by the data stack pointer (SP) is copied to AC0(15–0). Bits 39–16 of AC0 are unchanged. The SP is incremented by 1.

AC0 = pop()

## Syntax Characteristics

				D			
No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[3]	dst, Smem = <b>p</b>	op()		No	3	1	X
Opcod	e		111	0 0100 AAA	AA AZ	AI FDI	DD x1xx
Operar	nds	dst, Smem					
Description		by SP to desti	n moves the content o nation register dst a on pointed by SP + 1	nd moves the	conter	nt of the	16-bit data
When the destination register, dst, is an accumulator, the data memory operand is moved to the destination a ACx(15–0). The guard bits and the 16 higher bits ACx(39–16), are reloaded (unchanged) with the curre modified by this instruction. SP is incremented by 2.				ion ac bits c curren	cumulato of the ac	r low part, cumulator,	
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	n can be repeated.				

Syntax	Description
AC0, *AR3 = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to $ACO(15-0)$ and the content of the memory location pointed by SP + 1 is copied to the location addressed by AR3. Bits 39–16 of AC0 are unchanged. The SP is incremented by 2.

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[4]	ACx = dbl(po	op())		Yes	2	1	X	
Opcod	e			010	01 00	0E XXD	DD x011	
Operar	nds	ACx						
Descrij	ption	by SP to the ac	This instruction moves the content of the 16-bit data memory location pointed by SP to the accumulator high part $ACx(31-16)$ and moves the content of the 16-bit data memory location pointed by SP + 1 to the accumulator low part ACx(15-0).					
		The guard bits of the accumulator, ACx(39–32), are reloaded (un with the current value and are not modified by this instructi incremented by 2.						
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instruction	n can be repeated.					

Syntax	Description
AC1 = dbl(pop())	The content of the memory location pointed by the data stack pointer (SP) is copied to AC1(31–16) and the content of the memory location pointed by SP + 1 is copied to AC1(15–0). Bits 39–32 of AC1 are unchanged. The SP is incremented by 2.

Before		After	
AC1	03 3800 FC00	AC1	03 5644 F800
SP	0304	SP	0306
304	5644	304	5644
305	F800	305	F800

## Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = <b>pop()</b>			No	2	1	Х
Opcode				101	.1 10	11 AAA	A AAAI
Operands		Smem					
Descrip	ption		n moves the content of t memory (Smem) locatio			•	on pointed
Status	Bits	Affected by	none				
		Affects	none				
Repeat		This instruction	n can be repeated.				

Syntax		Description	Description					
*AR1 = pop()			The content of the memory location pointed by the data stack pointer (SP) is copied to the location addressed by AR1. The SP is incremented by 1.					
Before		After						
AR1	0200	AR1	0200					
SP	0300	SP	0301					
200	3400	200	6903					
300	6903	300	6903					

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[6]	dbl(Lmem) = p	oop()		No	2	1	X	
Opcode				101	11 10	000 AAA	A AAAI	
Operai	nds	Lmem						
Description		This instruction moves the content of the 16-bit data memory location pointed by SP to the 16 highest bits of data memory location Lmem and moves the content of the 16-bit data memory location pointed by SP + 1 to the 16 lowest bits of data memory location Lmem.						
		When Lmem is at an even address, the two 16-bit values popped from the stack are stored at memory location Lmem in the same order. When Lmem is at an odd address, the two 16-bit values popped from the stack are stored at memory location Lmem in the reverse order.						
		SP is incremented by 2.						
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructio	n can be repeated.					
Examp	ble							
Syntax	ĸ	Description						

Syntax	Description
dbl(*AR3–) = pop()	The content of the memory location pointed by the data stack pointer (SP) is copied to the 16 highest bits of the location addressed by AR3 and the content of the memory location pointed by SP + 1 is copied to the 16 lowest bits of the location addressed by AR3. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution. The SP is incremented by 2.

# Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	pshboth(xsrc)		Yes	2	1	Х
Opcod	e		010	01 00	OE XSS	S 0101
Operands		XSIC				
<b>Description</b> This instruction moves the lower 32 bits of ACx or the content of source register (XARx, XSP, XSSP, XDP, or XCDP) to the two 16-bit locations addressed by the data stack pointer (SP) and system state (SSP).				oit memory		
		The content of xsrc(15–0) is moved content of xsrc(31–16) is moved to				
		When xsrc is a 23-bit register, the SSP are filled with 0.	upper 9 bits of	the loc	ation add	dressed by
Status	Bits	Affected by none				
		Affects none				
Repeat	t	This instruction can be repeated.				
See Als	SO	See the following other related instructions:				
		Pop Accumulator or Extended Au	uxiliary Register	Conten	t from Sta	ck Pointers
		Pop Top of Stack				
		Push to Top of Stack				

#### Push to Top of Stack

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	push(src1, src2)	Yes	2	1	Х
[2]	push(src)	Yes	2	1	Х
[3]	push(src, Smem)	No	3	1	х
[4]	dbl(push(ACx))	Yes	2	1	Х
[5]	push(Smem)	No	2	1	Х
[6]	push(dbl(Lmem))	No	2	1	х

# **Description** These instructions move one or two operands to the data memory location addressed by the data stack pointer (SP). The operands may be:

- an accumulator, auxiliary, or temporary register
- a data memory location

The decrement operation performed on SP is done by the A-unit address generator dedicated to the stack addressing management.

- Status Bits Affected by none
  - Affects none
- **See Also** See the following other related instructions:
  - Pop Top of Stack
    - Dep Accumulator or Extended Auxiliary Register Content from Stack Pointers
    - D Push Accumulator or Extended Auxiliary Register Content to Stack Pointers

## Push to Top of Stack

## Syntax Characteristics

				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[1]	push(src1, sr	c2)		Yes	2	1	Х
Opcod	e			001	.1 10	OE FSS	S FDDD
		Note: FSSS = s	rc1, FDDD = src2				
Operar	nds	src1, src2					
Descrij	ption	register src1 to	a decrements SP b the 16-bit data me source register src2	mory location poi	nted by	y SP and	moves the
			rce register, src1 w part, ACx(15–(				
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	can be repeated.				

Syntax	Description
push(AR0, AC1)	The data stack pointer (SP) is decremented by 2. The content of AR0 is copied to the memory location pointed by SP and the content of AC1(15–0) is copied to the memory location pointed by SP + 1.
Before	After

AR0		0300	AR0		0300
AC1	03 5644	F800	AC1	03 5644	F800
SP		0300	SP		02FE
2FE		0000	2FE		0300
2FF		0000	2FF		F800
300		5890	300		5890

# Push to Top of Stack

## Syntax Characteristics

No. S	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2] <b>p</b>	oush(src)			Yes	2	1	Х
Opcode				01	01 00	0E FSS	S x110
Operands	6	SIC					
Descriptio	on	This instruction decrements SP by 1, then moves the content of the source register (src) to the 16-bit data memory location pointed by SP. When the source register is an accumulator, the source accumulator low part, ACx(15–0), is moved to the 16-bit data memory operand.					
Status Bit	ts	Affected by	none				
		Affects	none				
Repeat		This instruction	on can be repeate	d.			

Syntax	Description
push(AC0)	The data stack pointer (SP) is decremented by 1. The content of AC0(15–0) is copied to the memory location pointed by SP.

# Push to Top of Stack

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[3]	<b>push(</b> src, Sm	em)			No	3	1	Х
Opcode	e		:	L110	0100 AAA	A AA	AI FSS	S x0xx
Operan	nds	Smem, src						
Descrip	otion	This instruction decrements SP by 2, then moves the content of the sour register (src) to the 16-bit data memory location pointed by SP and moves t content of the data memory (Smem) location to the 16-bit data memory location pointed by SP + 1.						moves the
			rce register is an a moved to the 16-b		-			or low part,
Status	Bits	Affected by	none					
		Affects	none					
Repeat	:	This instructio	n can be repeated	l.				
Examp	le							

Syntax	Description
	The data stack pointer (SP) is decremented by 2. The content of AC0(15–0) is copied to the memory location pointed by SP and the content addressed by AR3 is copied to the memory location pointed by SP + 1.

# Push to Top of Stack

### Syntax Characteristics

_							
No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dbl(push(ACx))	)		Yes	2	1	Х
Opcod	e			010	01 00	0E xxS	S x111
Operar	nds	ACx					
Descrij	ption	This instruction decrements SP by 2, then moves the content of accumulator high part ACx(31–16) to the 16-bit data memory location poir by SP and moves the content of the accumulator low part ACx(15–0) to 16-bit data memory location pointed by SP + 1.					on pointed
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	on can be repeated.				
_	_						

Syntax	Description
dbl(push(AC0))	The data stack pointer (SP) is decremented by 2. The content of AC0(31–16) is copied to the memory location pointed by SP and the content of AC0(15–0) is copied to the memory location pointed by SP + 1.

# Push to Top of Stack

### Syntax Characteristics

				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[5]	push(Smem)			No	2	1	Х
Opcode				101	L1 01	.01 AAA	A AAAI
Operands		Smem					
Description			on decrements SP by 1 em) location to the 16-bi				
Status	Bits	Affected by	none				
		Affects	none				
Repeat		This instructio	n can be repeated.				
	la.						

Syntax Description					
push(*AR1)         The data stack pointer (SP) decremented by 1. The content copied to the memory location pointed by SP.				ntent addressed by AR1 is	
Before		After			
*AR1	6903	*AR1	6903		
SP	0305	SP	0304		
304	0000	304	6903		
305	0300	305	0300		

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[6]	push(dbl(Lmen	n <b>))</b>		No	2	1	X
Opcod	e			101	11 01	.11 AAA	IAAA A
Operar	nds	Lmem					
Descri	<b>scription</b> This instruction decrements SP by 2, then moves the 16 highest bits of memory location Lmem to the 16-bit data memory location pointed by S moves the 16 lowest bits of data memory location Lmem to the 16-b memory location pointed by SP + 1.						by SP and
		When Lmem is at an even address, the two 16-bit values pushed onto stack are stored at memory location Lmem in the same order. When Lme at an odd address, the two 16-bit values pushed onto the stack are store memory location Lmem in the reverse order.					
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instruction	n can be repeated.				
Examp	le						
Syntax	(	Description					

Syntax	Description
push(dbl(*AR3–))	The data stack pointer (SP) is decremented by 2. The 16 highest bits of the content at the location addressed by AR3 are copied to the memory location pointed by SP and the 16 lowest bits of the content at the location addressed by AR3 are copied to the memory location pointed by SP + 1. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution.

# Repeat Block of Instructions Unconditionally

No.	Syntax				Paral Enable		Size	Cycles	Pipeline
[1]	localrepeat{}				Yes	6	2	1	AD
[2]	blockrepeat{}				Yes	6	3	1	AD
Descri	ption	The	ese instructior	ns repeat a blo	ock of instructions	the nu	umber	of times s	pecified by:
					, if no loop has a if one level of the	-			
			op structures aracteristics:	s defined by	y these instruct	ions i	must	have the	e following
			The minimu	m number of	instructions exect	uted w	rithin o	ne loop ite	eration is 2.
			The minimu	ım number o	f cycles executed	d withi	in one	loop itera	ation is 2.
			The maxim	um loop size	is 64K bytes.				
		The block-repeat counter registers (BRCx) must be read 3 full cycle before the end of the loops in order to extract the correct loop iteration number from these registers without any pipeline stall.						•	
		The block-repeat operation can only be cleared by branching to destination address outside the active block-repeat loop.					ching to a		
			C54CM bit i	in ST1_55 ca	annot be modified	d withi	in a bl	ock-repea	at loop.
		Th	ese instructio	ons cannot be	e repeated.				
			e section 1.5 echanism.	for a list of ir	nstructions that c	annot	be us	ed in a re	peat block
Status	Bits	Aff	ected by	none					
		Aff	ects	none					
See Al	so	See the following other related instructions:							
			Repeat Sing	gle Instructio	n Conditionally				
			Repeat Sing	gle Instructio	n Unconditionally	/			
			Repeat Sing	gle Instruction	n Unconditionally	and D	Decrer	nent CSR	
			Repeat Sing	gle Instructio	n Unconditionally	/ and	Increr	nent CSR	2

### Repeat Block of Instructions Unconditionally

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	localrepeat{}				Yes	2	1	AD	
Opcod	e				01	00 1	01E   111	.1 1111	
Operar	nds	noi	ne						
Description			This instruction repeats a block of instructions the number of times specified by:						
			the	e content of BRC0 + 1, if no lo	oop has already	been d	etected. Ir	n this case:	
				In the address phase of the address of the first instruct			ded with th	e program	
				The program address of t two parallel instructions) pipeline and stored in RE	is computed in			-	
				BRC0 is decremented at the loop when its content	ented at the address phase of the last instruction of content is not equal to 0.				
				BRC0 contains 0 after the	e block-repeat o	peratio	n has end	led.	
				e content of BRS1 + 1, if one this case:	level of the loop	has alr	eady beer	n detected.	
			•	BRC1 is loaded with the c repeat block instruction.	ontent of BRS1	in the a	address pł	nase of the	
				In the address phase of the address of the first instruct			ded with th	ie program	
				The program address of t two parallel instructions) pipeline and stored in RE	is computed in		• •	•	
			•	BRC1 is decremented at the loop when its content			he last ins	struction of	
				BRC1 contains 0 after the	e block-repeat o	peratio	n has end	led.	
				BRS1 content is not impa	icted by the bloc	k-repe	at operati	on.	
	_								

Loop structures defined by this instruction must have the following characteristics:

- The minimum number of instructions executed within one loop iteration is 2.
- The minimum number of cycles executed within one loop iteration is 2.
- The maximum loop size is 64K bytes.
- The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.
- The block-repeat counter registers (BRCx) must be read 3 full cycles before the end of the loops in order to extract the correct loop iteration number from these registers without any pipeline stall.
- □ C54CM bit in ST1\_55 cannot be modified within a block-repeat loop.
- The following instructions cannot be used as the last instruction in the loop structure:

while (cond && (RPTC < k8)) repeat	repeat(k8)	repeat(CSR), CSR += k4
if (cond) execute(AD_Unit)	repeat(k16)	repeat(CSR), CSR += TAx
if (cond) execute(D_Unit)	repeat(CSR)	repeat(CSR), CSR -= k4

A local loop is defined as when all the code of the loop is repeatedly executed from within the instruction buffer queue (IBQ):

- ☐ All the code of the local loop must fit within the 64-byte, 4-byte-aligned IBQ; therefore, local repeat blocks are limited to 64 bytes minus the 0 to 3 bytes of first-instruction misalignment. The 64th byte of the IBQ can only occur in a paralleled instruction. See Figure 5–2 for legal uses of the local repeat instruction.
- The following instructions cannot be used as the last instruction in the local loop:

while (cond && (RPTC < k8)) repeat	repeat(k8)	repeat(CSR), CSR += k4
if (cond) execute(AD_Unit)	repeat(k16)	repeat(CSR), CSR += TAx
if (cond) execute(D_Unit)	repeat(CSR)	repeat(CSR), CSR -= k4

- Nested local repeat block instructions are allowed.
- See section 1.5 for a list of instructions that cannot be used in the local loop code.

		The only branch instructions allowed in a localrepeat structure are the branch instructions with a target branch address pointing to an instruction included within the loop code and being at a higher address than the branching instruction. In this case, the branch conditionally instruction is executed in 3 cycles and the condition is evaluated in the address phase of the pipeline (there is a 3-cycle latency on the condition setting).
	Co	mpatibility with C54x devices (C54CM = 1)
	Wł	en C54CM =1:
		This instruction only uses block-repeat level 0; block-repeat level 1 is disabled.
		The block-repeat active flag (BRAF) is set to 1. BRAF is cleared to 0 at the end of the block-repeat operation when BRC0 contains 0.
		You can stop an active block-repeat operation by clearing BRAF to 0.
		Block-repeat control registers for level 1 are not used. Nested block-repeat operations are supported using the C54x convention with context save/restore and BRAF. The control-flow context register (CFCT) values are not used.
		BRAF is automatically cleared to 0 when a far branch (FB) or far call (FCALL) instruction is executed.
Status Bits	Aff	ected by none
	Aff	ects none
Repeat	Th	s instruction cannot be repeated.

Syntax	Description				
localrepeat	A block of instructions	s is repeated as	defined by the c	content of BRC0	+ 1.
	Address	BRC0	RSA0	REA0	BRS1
BRC0 = #3		0003	0000	0000	0000
localrepeat {	004003	?*	4005	400D	?
	004005	?	?	?	?
	00400D	DTZ**	?	?	?
}		0000	4005	400D	0000
*?: Unchanged **DTZ: Decrease till			-	-	

#### Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction

(a) 60-Byte Unaligned Loop—Legal Use

		; no alignment directive
localrepeat {		
	1st instruction	
		} 60-byte loop body
	Last instruction	
}		
next instruction		

The entire local repeat block and the *next instruction* reside in the IBQ, this code is accepted by the assembler.

(b) 61-Byte Unaligned Loop with Single Instruction at End of Loop—Illegal Use

		; no alignment directive
localrepeat {		
	1st instruction	
		} 61-byte loop body
	Last instruction (nonparalleled = single)	
}		
next instruction		

The localrepeat instruction is not aligned; the *next instruction* may not be fetched in the IBQ. Because the last instruction of the localrepeat block is a nonparalleled (single) instruction, the CPU must confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is rejected by the assembler.

### Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(c) 61-Byte Unaligned Loop with Paralleled Instruction at End of Loop-Legal Use

		; no alignment directive
localrepeat {		
	1st instruction	
		} 61-byte loop body
	Last instruction (paralleled)	
}		
next instruction		

The localrepeat instruction is not aligned; the *next instruction* may not be fetched in the IBQ. Because the last instruction of the localrepeat block is a paralleled instruction, the CPU does not need to confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

(d) 61-Byte Aligned Loop with Single Instruction at End of Loop-Legal Use

align 4		; alignment directive
localrepeat {		
	1st instruction	
		} 61-byte loop body
	Last instruction (nonparalleled = single)	
}		
next instruction		

The localrepeat instruction is aligned, so the entire localrepeat block and the *next instruction* reside in the IBQ. Because the *next instruction* is in the IBQ, the CPU can confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

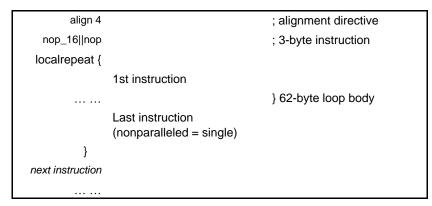
#### Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(e) 62-Byte Unaligned Loop—Illegal Use

		; no alignment directive
localrepeat {		
	1st instruction	
		} 62-byte loop body
	Last instruction	
}		
next instruction		

The localrepeat instruction is not aligned; the entire localrepeat block may not reside in the IBQ. Because the last instruction of the localrepeat block may not reside in the IBQ, this code is rejected by the assembler.

(f) 62-Byte Aligned Loop with Single Instruction at End of Loop—Legal Use



The nop instructions are aligned so the localrepeat instruction, the entire localrepeat block, and the *next instruction* reside in the IBQ. Because the *next instruction* is in the IBQ, the CPU can confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

### Figure 5–2. Legal Uses of Repeat Block of Instructions Unconditionally (localrepeat) Instruction (Continued)

(g) 64-Byte Aligned Loop with Paralleled Instruction at End of Loop-Legal Use

align 4		; alignment directive
nop_16		; 2-byte instruction
localrepeat {		
	1st instruction	
		} 64-byte loop body
	Last instruction (paralleled)	
}		
next instruction		

The nop instruction is aligned, so the localrepeat instruction and the entire localrepeat block reside in the IBQ; the *next instruction* is not fetched in the IBQ. Because the last instruction of the localrepeat block is a paralleled instruction, the CPU does not need to confirm that the *next instruction* does not have a parallel enable bit; therefore, this code is accepted by the assembler.

### Repeat Block of Instructions Unconditionally

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline
[2]	blockrepeat{}						Yes	3	1	AD
Opcod	е					0000	111E   11	11 1	111   113	11 1111
Operar	nds	noi	ne							
Descri	ption	Thi	s ins	struction repe	ats a block o	ofinstru	uctions the n	umber	of times s	pecified by:
			the	e content of B	RC0 + 1, if n	o loop	has already	been d	etected. I	n this case
					•		peline, RSA0 n of the loop		ded with th	ne program
			•	two paralle		is) is c	ast instruction		• •	•
							address pha ot equal to (		he last in	struction of
				BRC0 cont	ains 0 after	the blo	ock-repeat o	peratio	n has en	ded.
				e content of B this case:	RS1 + 1, if o	ne leve	el of the loop	has alr	eady bee	n detected
					aded with the k instructior		ent of BRS1	in the a	address p	hase of the
					•		beline, RSA n of the loop		ded with th	ne program
				two paralle		is) is c	ast instruction		• •	•
							address pha ot equal to (		he last in	struction of
				BRC1 cont	ains 0 after	the blo	ock-repeat o	peratio	n has en	ded.
				BRS1 cont	ent is not im	pacted	d by the bloc	ck-repe	at operat	ion.

Loop structures defined by these instructions must have the following characteristics:

- The minimum number of instructions executed within one loop iteration is 2.
- The minimum number of cycles executed within one loop iteration is 2.
- The maximum loop size is 64K bytes.
- ☐ The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.
- ☐ The block-repeat counter registers (BRCx) must be read 3 full cycles before the end of the loops in order to extract the correct loop iteration number from these registers without any pipeline stall.
- □ C54CM bit in ST1\_55 cannot be modified within a block-repeat loop.
- The following instructions cannot be used as the last instruction in the loop structure:

while (cond && (RPTC < k8)) repeat	repeat(k8)	repeat(CSR), CSR += k4
if (cond) execute(AD_Unit)	repeat(k16)	repeat(CSR), CSR += TAx
if (cond) execute(D_Unit)	repeat(CSR)	repeat(CSR), CSR -= k4

□ See section 1.5 for a list of instructions that cannot be used in the block-repeat loop code.

#### Compatibility with C54x devices (C54CM = 1)

When C54CM =1:

- This instruction only uses block-repeat level 0; block-repeat level 1 is disabled.
- ☐ The block-repeat active flag (BRAF) is set to 1. BRAF is cleared to 0 at the end of the block-repeat operation when BRC0 contains 0.
- You can stop an active block-repeat operation by clearing BRAF to 0.
- Block-repeat control registers for level 1 are not used. Nested block-repeat operations are supported using the C54x convention with context save/restore and BRAF. The control-flow context register (CFCT) values are not used.
- BRAF is automatically cleared to 0 when a far branch (FB) or far call (FCALL) instruction is executed.

Repeat Block of Instructions Unconditionally (blockrepeat)

Status Bits	Affected by	none	

Affects none

**Repeat** This instruction cannot be repeated.

Syntax	Description
blockrepeat	A block of instructions is repeated as defined by the content of BRC0 + 1. A second loop of instructions is repeated as defined by the content of BRS1 + 1 (BRC1 is loaded with the content of BRS1).

	Address	BRC0	RSA0	REA0	BRS1	BRC1	RSA1	REA1
BRC0 = #3		0003	0000	0000	0000	0000	0000	0000
BRC1 = #1		?*	?	?	0001	0001	?	?
blockrepeat {	004006	?	4009	4017	?	?	?	?
	004009	?	?	?	?	?	?	?
localrepeat {	00400B	?	?	?	?	(BRS1)	400D	4015
	00400D	?	?	?	?	?	?	?
	004015	?	?	?	?	DTZ**	?	?
}								
	004017	DTZ**	?	?	?	?	?	?
}		0000	4009	4017	0001	0000	400D	4015
*?: Unchanged								
**DTZ: Decrease	till zero							

# Repeat Single Instruction Conditionally

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	while (cond &	& (RF	PTC < k8)) repeat	Yes	3	1	AD		
Opcod	e		0000	000E xC	CC C	CCC kkk	k kkkk		
Operar	nds	со	nd, k8	·		Ļ			
Descri	ption	lor ins va	is instruction evaluates a single cong as the condition is true, the new structions is repeated the number of lue, k8 + 1. The maximum number ralleled instructions is 2 <sup>8</sup> –1 (255).	t instruction of times spector of executio	or the ified by ns of a	next two y an 8-bit given ins	parallelec immediate struction of		
		Th	e 8 LSBs of the repeat counter reg	ister (RPTC	):				
			Are loaded with the immediate va	lue at the ad	dress p	hase of th	ne pipeline		
			Are decremented by 1 in the dec	ode phase o	f the re	epeated in	struction.		
		Th	e 8 MSBs of RPTC:						
			Are loaded with the cond code at	the address	phase	e of the pip	peline.		
			Are untouched during the while/re	epeat structu	ire exe	cution.			
		At each step of the iteration, the condition defined by the cond field is teste in the execute phase of the pipeline. When the condition becomes false, th instruction repetition stops.							
			If the condition becomes false at a the 8 LSBs of RPTC are corrected were not performed.	•		•			
			Since the condition is evaluated instruction, when the condition is iterations of that repeated instruct access, and read phases of the modified the pointer registers use memory operands addresses in t	s tested false ion may have e pipeline. ed in the DAC	e, som e gone Therefo GEN ur	e of the s through th ore, they nits to ger	ucceeding e address may have		
			When the while/repeat structu single-repeat register (CSR) conta instructions have gone through t may then use the Repeat Single In to rewind the pointer registers. Not a false condition has been met in	ent enables y he address nstruction Ur te that this m	/ou to o phase ncondit ust only	determine of the pip ionally ins y be perfor	how many beline. You truction [3 rmed wher		

☐ The following table provides the 8 LSBs of RPTC and CSR once the while/repeat structure is exited.

If the condition is met	RPTC[7:0] content after exiting loop	CSR content after exiting loop
At 1 <sup>st</sup> iteration	RPTCinit + 1	4
At 2 <sup>nd</sup> iteration	RPTCinit	4
At 3 <sup>rd</sup> iteration	RPTC – 1	4
At RPTCinit – 2 iteration	4	3
At RPTCinit – 1 iteration	3	2
At RPTCinit iteration	2	1
At RPTCinit + 1 iteration	1	0
Never	0	0

RPTCinit is the number of requested iterations minus 1.

The repeat single mechanism triggered by this instruction is interruptible. Saving and restoring the RPTC content in ISRs enables you to preserve the while/repeat structure context.

When the while/repeat structure contains any form of a store-to-memory instruction, the store-to-memory instruction is only disabled one cycle after the condition is evaluated to be false. Therefore, the store-to-memory instruction is executed once more than other processing instructions updating CPU registers. This enables you to store the last values obtained in these registers when the condition was met.

Instead of programming a number of iterations (minus 1) equal to 0, it is recommended that you use the conditional execute() structure.

This instruction cannot be used as the last instruction in a repeat loop structure.

See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.

Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx
	Affects	ACOVx
Repeat	This instruction	a cannot be repeated.

**See Also** See the following other related instructions:

- Repeat Block of Instructions Unconditionally
- Repeat Single Instruction Unconditionally
- Repeat Single Instruction Unconditionally and Decrement CSR
- Repeat Single Instruction Unconditionally and Increment CSR

#### Example

Syntax	Description
while (AC1 > #0 && (RPTC < #7)) repeat	As long as the content of AC1 is greater than 0 and the repeat counter is not equal to 0, the next single instruction is repeated as defined by the unsigned 8-bit value $(7) + 1$ . At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

while (AC1 > #0 && (RPTC < #7)) repeat					ad	dress:	0040	04	
AC1 = AC1 - (T0 * *AR1)							0040	08	
								0040	0B
Before					After				
AC1	00	2359	0340		AC1	00	1FC2	7B40	
Т0			0340		т0			0340	
*AR1			2354		*AR1			2354	
RPTC		4	1106†		RPTC			0000	

<sup>†</sup> At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

### Repeat Single Instruction Unconditionally

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	repeat(k8)	Yes	2	1	AD
[2]	repeat(k16)	Yes	3	1	AD
[3]	repeat(CSR)	Yes	2	1	AD

Description	This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1 or an immediate value, kx + 1. This value is loaded into the repeat counter register (RPTC). The maximum number of executions of a given instruction or paralleled instructions is $2^{16} - 1$ (65535).						
	The repeat single mechanism triggered by these instructions is interruptible.						
	These instructions cannot be repeated.						
	These instructions cannot be used as the last instruction in a repeat loop structure.						
	Two paralleled instructions can be repeated when following the parallelism general rules.						
	See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.						
Status Bits	Affected by none						
	Affects none						
See Also	See the following other related instructions:						
	Repeat Block of Instructions Unconditionally						
	Repeat Single Instruction Conditionally						
	Repeat Single Instruction Unconditionally and Decrement CSR						
	Repeat Single Instruction Unconditionally and Increment CSR						

# Repeat Single Instruction Unconditionally

# Syntax Characteristics

Na	Suntor			Parallel	<b>C</b> !	Cucles	Dineline	
No.	Syntax			Enable Bit Yes	Size	Cycles	Pipeline AD	
[1]	repeat(k8)							
[2]	repeat(k16)			Yes	3	1	AD	
Opcod	e	k8		01	00 1	10E kkk	k kkkk	
		k16	000	) 110E kk	kk k	kkk kkk	k kkkk	
Operar	nds	kx						
Descri	ption	This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by an immediate value, $kx + 1$ . The repeat counter register (RPTC):						
		Is loaded w	vith the immediate va	lue in the add	lress p	hase of th	e pipeline.	
		Is decrement	ented by 1 in the deco	de phase of	the rep	eated inst	ruction.	
		Contains 0	at the end of the rep	of the repeat single mechanism.				
		Must not b mechanisn	e accessed when it is n.	being decrei	menteo	l in the rep	peat single	
		The repeat sing	gle mechanism trigge	red by this in	structio	on is interr	uptible.	
		Two paralleled general rules.	instructions can be	repeated whe	en follo	wing the I	oarallelism	
		This instruction structure.	n cannot be used a	s the last in	structio	on in a re	epeat loop	
		See section 1.5 mechanism.	5 for a list of instructio	ns that canno	t be us	ed in a rej	beat single	
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instructior	a cannot be repeated.					
SPRU3	75G			Instruction	Set Do	scriptions	5-361	

3400

7900

402

403

# Example 1

Syntax		Description					
repeat(#3)		The single ins	The single instruction following the repeat instruction is repeated four times.				
AC1 = AC1	+ *AR3+ * *AR4+						
		-					
Before		After					
AC1	00 0000 0000	AC1	00 3376 AD10				
AR3	0200	AR3	0204				
AR4	0400	AR4	0404				
200	AC03	200	AC03				
201	3468	201	3468				
202	FE00	202	FE00				
203	23DC	203	23DC				
400	D768	400	D768				
401	6987	401	6987				

### Example 2

402

403

Syntax	Description
repeat(#513)	A single instruction is repeated as defined by the unsigned 16-bit value + 1 (513 + 1).

3400

7900

# Repeat Single Instruction Unconditionally

No. Synta	ix		Parallel Enable Bit	Size	Cycles	Pipeline		
[3] <b>repe</b> a	it(CSR)		Yes	2	1	AD		
Opcode			01	00 1	00E xxx	x x000		
Operands	none							
Description	instruct	This instruction repeats the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. The repeat counter register (RPTC):						
	🗋 Is I	paded with CSR content	in the address ph	ase of	the pipeli	ne.		
	🗋 ls c	Is decremented by 1 in the decode phase of the repeated instruction.						
	🗋 Co	Contains 0 at the end of the repeat single mechanism.						
		Must not be accessed when it is being decremented in the repeat single mechanism.						
	The rep	The repeat single mechanism triggered by this instruction is interruptible.						
	•	Two paralleled instructions can be repeated when following the parallelism general rules.						
		This instruction cannot be used as the last instruction in a repeat loop structure.						
	See se mecha	ction 1.5 for a list of instru iism.	ictions that canno	t be us	ed in a re	peat single		
Status Bits	Affecte	by none						
	Affects	none						
Repeat	This ins	truction cannot be repea	ated.					

Syntax		Description	on				
repeat(CSR)		-	The single instruction following the repeat instruction is repeated as defined				
AC1 = AC1	+ *AR3+ * *AR4+	by the cor	by the content of CSR + 1.				
Before		After					
AC1	00 0000 0000	AC1	00 3376 AD10				
CSR	0003	CSR	0003				
AR3	0200	AR3	0204				
AR4	0400	AR4	0404				
200	AC03	200	AC03				
201	3468	201	3468				
202	FEOO	202	FEOO				
203	23DC	203	23DC				
400	D768	400	D768				
401	6987	401	6987				
402	3400	402	3400				
403	7900	403	7900				

Repeat Single Instruction Unconditionally and Decrement CSR

### Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	repeat(CSR),	<b>CSR</b> –= k4		Yes	2	1	Х
Opcode	9			01	00 1	00E   kkk	k x011
Operan	ds	k4					
Descrip	otion	instructions the	n repeats the next e number of times sp egister (CSR) + 1. Th	pecified by the	e conte	ent of the	computed
		Is loaded v	vith CSR content in th	ne address ph	ase of	the pipeli	ne.
		Is decrement	ented by 1 in the deco	ode phase of	the rep	eated inst	truction.
		Contains 0 at the end of the repeat single mechanism.					
		Must not be accessed when it is being decremented in the repeat single mechanism.					
		decremented b of the pipeline;	it ALU, this instruct y k4. The CSR modifi there is a 3-cycle lat address phase.	ication is perfo	ormedi	in the exe	cute phase
		The repeat single mechanism triggered by this instruction is interruptible.					
		Two paralleled instructions can be repeated when following the parallelism general rules.					
		This instruction structure.	n cannot be used a	as the last in	structio	on in a re	epeat loop
		See section 1.5 mechanism.	5 for a list of instructio	ons that canno	t be us	ed in a re	peat single
Status	Bits	Affected by	none				
		Affects	none				
Repeat		This instructior	a cannot be repeated.				

SPRU375G

#### Repeat Single Instruction Unconditionally and Decrement CSR (repeat)

See Also S	ee the following other related instructions:		
	Repeat Block of Instructions Unconditionally		
	Repeat Single Instruction Conditionally		
C	Repeat Single Instruction Unconditionally		
	Repeat Single Instruction Unconditionally and Increment CSR		
Example			
Syntax	Description		
repeat(CSR), CSR -= #2	A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is decremented by the unsigned 4-bit value (2).		

Repeat Single Instruction Unconditionally and Increment CSR

# Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	repeat(CSR), CSR += TAx	Yes	2	1	Х
[2]	repeat(CSR), CSR -= k4	Yes	2	1	Х

Description	These instructions repeat the next instruction or the next two paralleled instructions the number of times specified by the content of the computed single repeat register (CSR) + 1. This value is loaded into the repeat counter register (RPTC). The maximum number of executions of a given instruction or paralleled instructions is $2^{16}$ –1 (65535).
	With the A-unit ALU, these instructions allow the content of CSR to be incremented. The CSR modification is performed in the execute phase of the pipeline; there is a 3-cycle latency between the CSR modification and its usage in the address phase.
	The repeat single mechanism triggered by these instructions is interruptible.
	Two paralleled instructions can be repeated when following the parallelism general rules.
	These instructions cannot be repeated.
	These instructions cannot be used as the last instruction in a repeat loop structure.
	See section 1.5 for a list of instructions that cannot be used in a repeat single mechanism.
Status Bits	Affected by none
	Affects none
See Also	See the following other related instructions:
	Repeat Block of Instructions Unconditionally
	Repeat Single Instruction Conditionally
	Repeat Single Instruction Unconditionally
	Repeat Single Instruction Unconditionally and Decrement CSR

SPRU375G

### Repeat Single Instruction Unconditionally and Increment CSR

### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	repeat(CSR),	CSR += TAx		Yes	2	1	Х	
Opcod	e			01	00 1	00E FSS	S x001	
Opera	nds	TAx						
Descri	ption	instructions th	on repeats the next e number of times s register (CSR) + 1. Th	pecified by the	e conte	ent of the	computed	
		Is loaded with CSR content in the address phase of the pipeline.						
		Is decremented by 1 in the decode phase of the repeated instruction.						
		Contains 0 at the end of the repeat single mechanism.						
		Must not b mechanisi	be accessed when it is m.	s being decrer	nentec	l in the re	peat single	
		incremented b execute phase	hit ALU, this instruc y the content of TAx. e of the pipeline; ther nd its usage in the ad	The CSR mod e is a 3-cycle	ificatio	n is perfor	med in the	
		The repeat single mechanism triggered by this instruction is interruptible						
		Two paralleled general rules.	d instructions can be	repeated whe	en follo	wing the	parallelism	
		This instruction structure.	on cannot be used a	as the last in	structio	on in a re	epeat loop	
		See section 1. mechanism.	5 for a list of instructio	ons that canno	t be us	ed in a re	peat single	
Status	Bits	Affected by	none					
		Affects	none					
Repeat	t	This instruction	n cannot be repeated					

Syntax	Description
repeat(CSR), CSR += T1	A single instruction is repeated as defined by the content of CSR + 1. The
	content of CSR is incremented by the content of temporary register T1.

# Repeat Single Instruction Unconditionally and Increment CSR

No.	Syntax		1	Parallel Enable Bit	Size	Cycles	Pipeline		
[2]	repeat(CSR)	, <b>CSR +</b> = k4		Yes	2	1	Х		
Opcod	e			01	00 1	00E kkk	k x010		
Operar	nds	k4		•		ļ			
Descri	ption	instructions th	on repeats the next ins e number of times spec register (CSR) + 1. The re	ified by the	e conte	ent of the	computed		
		Is loaded	Is loaded with CSR content in the address phase of the pipeline.						
		Is decremented by 1 in the decode phase of the repeated instruction.							
		Contains 0 at the end of the repeat single mechanism.							
		Must not be accessed when it is being decremented in the repeat single mechanism.							
		incremented b of the pipeline	nit ALU, this instruction by k4. The CSR modificati ; there is a 3-cycle latence e address phase.	ion is perfo	rmed i	n the exe	cute phas		
		The repeat single mechanism triggered by this instruction is interruptible.							
		Two parallele general rules.	d instructions can be rep	eated whe	n follo	wing the	parallelisn		
		This instruction structure.	on cannot be used as t	he last ins	structio	on in a re	epeat loo		
		See section 1. mechanism.	5 for a list of instructions	that cannot	t be us	ed in a re	peat singl		
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t	This instructio	n cannot be repeated.						
Examp	le								

Syntax	Description
repeat(CSR), CSR += #2	A single instruction is repeated as defined by the content of CSR + 1. The content of CSR is incremented by the unsigned 4-bit value (2).

SPRU375G

### Return Conditionally

# Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles <sup>†</sup>	Pipeline
[1]	if (cond) return	Yes	3	5/5	R

t x/y cycles: x cycles = condition true, y cycles = condition false

Opcode	0000 001E xCCC CCCC xxxx xxxx
Operands	cond
Description	This instructions evaluates a single condition defined by the cond field in the read phase of the pipeline. If the condition is true, a return occurs to the return address of the calling subroutine. There is a 1-cycle latency on the condition setting. A single condition can be tested as determined by the cond field of the instruction. See Table 1–3 for a list of conditions.
	After returning from a called subroutine, the CPU restores the value of two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.
	In the slow-return process (default), the return address (from the PC) and the loop context bits are restored from the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.
	In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. For fast-return mode operation, see the <i>TMS320C55x DSP CPU Reference Guide</i> (SPRU371).
	When a return from a subroutine occurs:
	The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the read phase of the pipeline.
	☐ The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the read phase of the pipeline.

	System Stack (SSP)		Data Stack (SP)
$\begin{array}{lll} \mbox{Before} & \rightarrow & \mbox{SSP} = x \\ \mbox{Return} \end{array}$	(Loop bits):PC(23–16)	$\begin{array}{lll} \mbox{Before} & \rightarrow & \mbox{SP} = y \\ \mbox{Return} \end{array}$	PC(15–0)
$\begin{array}{rcl} \mbox{After} & \rightarrow & \mbox{SSP} = x + 1 \\ \mbox{Return} \end{array}$	Previously stored data	$\begin{array}{rcl} \mbox{After} & \rightarrow & \mbox{SP} = y + 1 \\ \mbox{Return} \end{array}$	Previously stored data

### Compatibility with C54x devices (C54CM = 1)

	When C54CM = 1, the comparison of accumulators to 0 is performed as if M40 was set to 1.			
Status Bits	Affected by	ACOVx, CARRY, C54CM, M40, TCx		
	Affects	ACOVx		
Repeat	This instructior	This instruction cannot be repeated.		
See Also	See the following other related instructions:			
	Call Conditionally			
	Call Unconditionally			
	Return from Interrupt			
	Return Unconditionally			

Syntax	Descr	Description				
if (ACOV0 = #0) return		The AC0 overflow bit is equal to 0, the program counter (PC) is loaded with the return address of the calling subroutine.				
Before		After				
ACOV0	0	ACOV0	0			
PC		PC	(return address)			
SP		SP				

# Return Unconditionally

No.	Syntax	Parallel Enable Bit Size Cycles Pipeline				
[1]	return	Yes 2 5 D				
Opcod	le	0100 100E xxxx x100				
Opera	nds	none				
Descri	ption	This instruction passes control back to the calling subroutine.				
		After returning from a called subroutine, the CPU restores the value of two internal registers: the program counter (PC) and a loop context register. The CPU uses these values to re-establish the context of the program sequence.				
		In the slow-return process (default), the return address (from the PC) and the loop context bits are restored from the stacks (in memory). When the CPU returns from a subroutine, the speed at which these values are restored is dependent on the speed of the memory accesses.				
		In the fast-return process, the return address (from the PC) and the loop context bits are restored from the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. For fast-return mode operation, see the <i>TMS320C55x DSP CPU Reference Guide</i> (SPRU371).				
		The loop context bits concatenated with the 8 MSBs of the return address are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word in the address phase of the pipeline.				
		The 16 LSBs of the return address are popped from the top of the data stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.				
Befor Return Afte Return	n er → SSP:	System Stack (SSP)Data Stack (SP)SP = x(Loop bits):PC(23-16)Before $\rightarrow$ SP = yPC(15-0)= x + 1Previously stored dataAfter $\rightarrow$ SP = y + 1Previously stored data				

Status Bits	Affected by	none
	Affects	none
Repeat	This instruction	a cannot be repeated.
See Also	See the following	ng other related instructions:
	Call Condit	ionally
	Call Unconditionally	
	Return Conditionally	
	Return from	n Interrupt

Syntax	Description
return	The program counter is loaded with the return address of the calling subroutine.

# Return from Interrupt

No.	Syntax	_		Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	return_int			Yes	2	5	D	
Opcod	e			01	00 1	00E xxx	x x101	
Opera	nds	none						
Descri	ption	Thi	s instruction passes control b	ack to the interru	pted ta	isk.		
		res pro	er returning from an interrupt tores the value of some CP ogram counter (PC) and a loop re-establish the context of the	U registers and context register.	two in The CP	ternal reg	isters: the	
		cor me	he slow-return process (defauntext bits, and some CPU in mory). When the CPU returns a restored is dependent on the	registers are res from an ISR, the s	tored a	from the at which th	stacks (in ese values	
		cor cor CF reg	the fast-return process, the intext bits are restored from the totol-flow context register (CFC CT as a pair with dedicated, 3 gisters are restored from the paration, see the <i>TMS320C55x</i>	he return addres CT). You can read 32-bit load and sto stacks (in mem	s regis d from o ore insti ory). F	ster (RET) or write to ructions. S or fast-re	A) and the RETA and Some CPU turn mode	
			The loop context bits concate are popped from the top of t incremented by 1 word in the	the system stack	pointe	r (SSP). 1		
			The 16 LSBs of the return a stack pointer (SP). The SP is of the pipeline.			•		
			The debug status register (I SSP. The SSP is incremen pipeline.	,		•	•	
			The status register 1 (ST1_5 SP is incremented by 1 word					
			The 7 higher bits of status reg are popped from the top of S read phase of the pipeline.					

☐ The status register 2 (ST2\_55) content is popped from the top of SP. The SP is incremented by 1 word in the read phase of the pipeline.

			System Stack (SSI	P)				Data Stack (SP)
Before	$\rightarrow$	SSP = x	(Loop bits):PC(23-1	16)	Before	$\rightarrow$	SP = y	PC(15–0)
Return		SSP = x + 1	DBSTAT		Return		SP = y + 1	ST1_55
		SSP = x + 2	ST0_55(15–9)				SP = y + 2	ST2_55
After Return	$\rightarrow$	SSP = x + 3	Previously stored da	ata	After Return	$\rightarrow$	SP = y + 3	Previously stored data
Status B	Bits	ts Affected by none Affects none						
Repeat		This instruction cannot be repeated.						
See Also	D	See the following other related instructions:						
		Return Conditionally						
		Return Unconditionally						
		Ę	Software Interrupt					
		Ę	Software Trap					
<b>-</b>	_							

Syntax	Description
return_int	The program counter (PC) is loaded with the return address of the interrupted task.

dst, src

#### Rotate Left Accumulator, Auxiliary, or Temporary Register Content

#### **Syntax Characteristics**

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	dst = BitOut \\ src \\ BitIn				
[1]	dst = TC2 \\ src \\ TC2	Yes	3	1	Х
[2]	dst = TC2 \\ src \\ CARRY	Yes	3	1	х
[3]	dst = CARRY \\ src \\ TC2	Yes	3	1	Х
[4]	dst = CARRY \\ src \\ CARRY	Yes	3	1	Х

#### Opcode

0001 001E FSSS xx11 FDDD 0xvv

#### Operands

**Description** This instruction performs a bitwise rotation to the MSBs. Both TC2 and CARRY can be used to shift in one bit (BitIn) or to store the shifted out bit (BitOut). The one bit in BitIn is shifted into the source (src) operand and the shifted out bit is stored to BitOut.

- When the destination (dst) operand is an accumulator:
  - if an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the register are zero extended to 40 bits
  - the operation is performed on 40 bits in the D-unit shifter
  - BitIn is inserted at bit position 0
  - BitOut is extracted at a bit position according to M40

When the destination (dst) operand is an auxiliary or temporary register:

- if an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation
- the operation is performed on 16 bits in the A-unit ALU
- BitIn is inserted at bit position 0
- BitOut is extracted at bit position 15

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits Affected by CARRY, M40, TC2

Affects CARRY, TC2

**Repeat** This instruction can be repeated.

See Also See the following other related instructions:

Rotate Right Accumulator, Auxiliary, or Temporary Register Content

Syntax		Description			
AC1 = CARR	RY \\ AC1 \\ TC2	The value of TC2 (1) before the execution of the instruction is shifted into the LSB of AC1 and bit 31 shifted out from AC1 is stored in the CARRY status bit. The rotated value is stored in AC1. Because $M40 = 0$ , the guard bits (39–32) are cleared.			
Before		After			
AC1	OF E340 5678	AC1	00 C680 ACF1		
TC2	1	TC2	1		
CARRY	1	CARRY	1		
M40	0	M40	0		

dst, src

#### Rotate Right Accumulator, Auxiliary, or Temporary Register Content

#### Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
	dst = BitIn // src // BitOut				
[1]	dst = TC2 // src // TC2	Yes	3	1	Х
[2]	dst = TC2 // src // CARRY	Yes	3	1	Х
[3]	dst = CARRY // src // TC2	Yes	3	1	Х
[4]	dst = CARRY // src // CARRY	Yes	3	1	х

#### Opcode

0001 001E FSSS xx11 FDDD 1xvv

#### Operands

**Description** This instruction performs a bitwise rotation to the LSBs. Both TC2 and CARRY can be used to shift in one bit (BitIn) or to store the shifted out bit (BitOut). The one bit in BitIn is shifted into the source (src) operand and the shifted out bit is stored to BitOut.

- When the destination (dst) operand is an accumulator:
  - if an auxiliary or temporary register is the source (src) operand of the instruction, the 16 LSBs of the register are zero extended to 40 bits
  - the operation is performed on 40 bits in the D-unit shifter
  - BitIn is inserted at a bit position according to M40
  - BitOut is extracted at bit position 0

U When the destination (dst) operand is an auxiliary or temporary register:

- if an accumulator is the source (src) operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation
- the operation is performed on 16 bits in the A-unit ALU
- BitIn is inserted at bit position 15
- BitOut is extracted at bit position 0

#### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits	Affected by	CARRY, M40, TC2
-------------	-------------	-----------------

Affects CARRY, TC2

**Repeat** This instruction can be repeated.

See Also See the following other related instructions:

Rotate Left Accumulator, Auxiliary, or Temporary Register Content

Syntax		Description	Description					
AC1 = TC2	2 // AC0 // TC2	The value of TC2 (1) before the execution of the instruction is shifted into bit 31 of AC0 and the LSB shifted out from AC0 is stored in TC2. The rotated value is stored in AC1. Because $M40 = 0$ , the guard bits (39–32) are cleared.						
Before		After						
AC0	5F B000 1234	AC0	5F B000 1234					
AC1	00 C680 ACF1	AC1	00 D800 091A					
TC2	1	TC2	0					
M40	0	M40	0					

#### Round Accumulator Content

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACx)			Yes	2	1	Х
Opcod	e			01	01 0	10E DDS	S 101%
Operands			x, ACy				
Description			s instruction performs a roundi	ng of the sour	ce acc	umulator	ACx in the
			The rounding operation dependent	ds on RDM:			
			■ When RDM = 0, the biase 8000h (2 <sup>15</sup> ) is added to the				
			■ When RDM = 1, the unbias According to the value accumulator ACx, 8000h (2	of the 17 LS			
			if( 8000h < bit(15-0)	) < 10000h)			
			add 8000h to the	40-bit sourc	ce aco	cumulato	r ACx
			else if( bit(15-0) ==	= 8000h)			
			if( bit(16) == 1)				
			add 8000h to the	40-bit sourc	ce aco	cumulato	r ACx
			If a rounding has been perf cleared to 0.	formed, the 16	lowest	bits of the	e result are
			Addition overflow detection dep	pends on M40.			
			No addition carry report is store	ed in CARRY s	tatus I	oit.	
			If an overflow is detected, the d (ACOVy) is set.	lestination accu	umulat	or overflov	w status bi
			When an overflow is detected, SATD.	the accumulate	or is sa	aturated a	ccording to
		Со	mpatibility with C54x devices	(C54CM = 1)			
		C5	en this instruction is executed wi 4CM = 1, the rounding is pe cumulator ACx.		•	•	

Status Bits	Affected by	C54CM, M40, RDM, SATD		
	Affects	ACOVy		
Repeat	This instruction cannot be repeated.			

Syntax			Desc	Description					
AC1 = rnd	(AC0)		The c is stor	The content of AC0 is added to 8000h, the 16 LSBs are cleared to 0, and the result is stored in AC1. M40 is cleared to 0, so overflow is detected at bit 31; SATD is cleared to 0, so AC1 is not saturated.					
Before				After					
AC0	EF	OFFO	8023	AC0	EF 0FF0 8023				
AC1	00	0000	0000	AC1	EF 0FF1 0000				
RDM			1	RDM	1				
M40			0	M40	0				
SATD			0	SATD	0				
ACOV1			0	ACOV1	1				

#### Saturate Accumulator Content

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = satura	ite(rnd(ACx))	Yes	2	1	Х
Opcod	e		01	01 0	10E DDS	S 110%
Opera	nds	ACx, ACy				
Descri	ption	This instruction performs a sat 32-bit width frame in the D-unit		ce acc	umulator	ACx to the
		A rounding is performed i instruction. The rounding o	•	-		lied to the
		When RDM = 0, the b 8000h (2 <sup>15</sup> ) is added to	-			
		When RDM = 1, the ur According to the val accumulator ACx, 8000	ue of the 17 LS			
		if( 8000h < bit(15	5-0) < 10000h)			
		add 8000h to t	he 40-bit sourc	ce aco	cumulato	r ACx
		else if( bit(15-0)	) == 8000h)			
		if( bit(16) ==	-			
		add 8000h to tl				-
		If a rounding has been cleared to 0.	performed, the 16	lowest	t bits of the	e result are
		An overflow is detected at	bit position 31.			
		No addition carry report is	stored in CARRY s	tatus I	oit.	
		If an overflow is detected, the (ACOVy) is set.	he destination accu	umulat	or overflov	w status bit
		When an overflow is detended Saturation values are FF 8000 0000h (negative of the second s	00 7FFF FFFFh	ition re (pos	-	saturated. erflow) or
		Compatibility with C54x devi	ces (C54CM = 1)			
		When this instruction is execute C54CM = 1, the rounding is accumulator ACx.		•	•	

Status Bits	Affected by	C54CM, RDM		
	Affects	ACOVy		
Repeat	This instruction can be repeated.			

-

Example 1	I						
Syntax		Description					
AC1 = satu	urate(A	C0)		32-bit width conte pred in AC1.	nt of AC0	is saturat	ed and the saturated value, FF 8000 0000,
Before				After			
AC0	EF	0FF0	8023	AC0	EF OF	F0 8023	
AC1	00	0000	0000	AC1	FF 80	00 0000	
ACOV1			0	ACOV1		1	

Syntax Description							
AC1 = sat rate(rnd(A				The 32-bit width content of AC0 is saturated. The saturated value, 00 7FFF FF is rounded, 16 LSBs are cleared, and stored in AC1.			
Before				After			
AC0	00	7fff	8000	AC0	00 7FFF 8000		
AC1	00	0000	0000	AC1	00 7FFF 0000		
RDM			0	RDM	0		
ACOV1			0	ACOV1	1		

# Set Accumulator, Auxiliary, or Temporary Register Bit

#### **Syntax Characteristics**

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	bit(src, Baddr)	= #1	No	3	1	Х			
Opcod	e	111	0 1100 AA	AA A	AAI FSS	S 000x			
Opera	nds	Baddr, src							
Descri	ption	This instruction performs a bit mani	pulation:						
		In the D-unit ALU, if the source	(src) register	operan	id is an ac	cumulator.			
		In the A-unit ALU, if the source temporary register.	e (src) register	r opera	ind is an a	auxiliary or			
		The instruction sets to 1 a single bi Baddr, of the source register.	t, as defined b	by the b	oit address	sing mode,			
		The generated bit address must be within:							
		<ul> <li>0–39 when accessing accumula bit address are used to determ address is not within 0–39, the s</li> </ul>	ine the bit po	sition).	If the ge	nerated bit			
		0–15 when accessing auxiliary of the generated address are u		-	· •				
Status	Bits	Affected by none							
		Affects none							
Repea	t	This instruction can be repeated.							
See Al	so	See the following other related inst	uctions:						
		Clear Accumulator, Auxiliary, or	Temporary R	egister	Bit				
		Complement Accumulator, Aux	iliary, or Temp	orary F	Register B	it			
		Set Memory Bit							
		Set Status Register Bit							
Evamr									

Syntax	Description	
bit(AC0, AR3) = #1	The bit at the position defined by the content of AR3(4–0) in AC0 is set to 1.	

Set Memory Bit

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	bit(Smem, src)	= #1	No	3	1	X			
Opcod	e	1110	0011 AA	AA AZ	AAI FSS	S 1100			
Operar	nds	Smem, src							
Description			This instruction performs a bit manipulation in the A-unit ALU. The instruction sets to 1 a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.						
		The generated bit address must be wi are used to determine the bit position	•	ly the 4	LSBs of t	he register			
Status	Bits	Affected by none							
		Affects none							
Repeat	t	This instruction can be repeated.							
See Als	SO	See the following other related instructions:							
		Clear Memory Bit							
		Complement Memory Bit							
		Set Accumulator, Auxiliary, or Te	mporary Reg	jister B	it				
		Set Status Register Bit							

Syntax	Description
bit(*AR3, AC0) = #1	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is set to 1.

# Set Status Register Bit

#### **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	bit(ST0, k4) = #1	Yes	2	1	Х
[2]	bit(ST1, k4) = #1	Yes	2	1	х
[3]	bit(ST2, k4) = #1	Yes	2	1	х
[4]	bit(ST3, k4) = #1	Yes	2	1†	Х

<sup>†</sup> When this instruction is decoded to modify status bit CAFRZ (15), CAEN (14), or CACLR (13), the CPU pipeline is flushed and the instruction is executed in 5 cycles regardless of the instruction context.

Opcode	STO	0100 011E kkkk 0001
	ST1	0100 011E kkkk 0011
	ST2	0100 011E kkkk 0101
	ST3	0100 011E kkkk 0111
Operands	k4, STx	
Description	These instructions perform a bit manipulation	n in the A-unit ALU.
	These instructions set to 1 a single bit, as defi k4, in the selected status register (ST0, ST1,	•
	Compatibility with C54x devices (C54CM =	= 1)
	C55x DSP status registers bit mapping (Figure correspond to C54x DSP status register bits.	,
Status Bits	Affected by none	
	Affects Selected status bits	
Repeat	This instruction cannot be repeated.	
See Also	See the following other related instructions:	
	Clear Status Register Bit	
	G Set Accumulator, Auxiliary, or Temporary	Register Bit
	Set Memory Bit	

Syntax			Description			
bit(ST0, ST0_	_CARRY) = #1; \$	ST0_CARRY = bit 11	The ST0 bit position defined by the label (ST0_CARRY, bit 11) is set to 1.			
Before		After				
ST0	0000	ST0	0800			

ST0_55							
15	14	13		12	11	10	9
ACOV2 <sup>†</sup>	ACOV3	TC1	Т	C2 C	ARRY	ACOV0	ACOV1
R/W-0	R/W–0	R/W-	1 R/	W–1 R	/W–1	R/W-0	R/W–0
8							0
			[	OP			
			R/	W–0			
ST1_55							
15	14	13	12	11	10	9	8
BRAF	CPL	XF	HM	INTM	<b>M40</b> <sup>†</sup>	SATD	SXMD
R/W–0	R/W–0	R/W–1	R/W–0	R/W–1	R/W-0	R/W–0	R/W–1
7	6	5	4				0
C16	FRCT	C54CM <sup>†</sup>			ASM		
R/W–0	R/W-0	R/W–1			R/W-0		
ST2_55							
15	14	13	12	11	10	9	8
ARMS	Rese	erved	DBGM	EALLOW	RDM	Reserved	CDPLC
R/W–0			R/W-1	R/W–0	R/W–0		R/W–0
7	6	5	4	3	2	1	0
AR7LC							U U
ARTLO	AR6LC	AR5LC	AR4LC	AR3LC	AR2LC	AR1LC	AR0LC
R/W–0	R/W–0	AR5LC R/W–0	AR4LC R/W–0	AR3LC R/W–0	AR2LC R/W–0	AR1LC R/W–0	·
J							AR0LC
R/W–0							AR0LC
R/W–0 ST3_55	R/W–0	R/W–0	R/W–0	R/W–0 11	R/W–0		AROLC R/W–0 8
R/W–0 <b>ST3_55</b> 15	R/W–0 14	R/W–0 13	R/W–0 12	R/W–0 11	R/W–0	R/W-0	AROLC R/W–0 8
R/W-0 ST3_55 15 CAFRZ <sup>†</sup>	R/W-0 14 <b>CAEN</b> †	R/W-0 13 <b>CACLR</b> †	R/W-0 12 <b>HINT</b> ‡	R/W–0 11	R/W–0	R/W-0	AROLC R/W–0 8
R/W-0 <b>ST3_55</b> 15 <b>CAFRZ</b> † R/W-0	R/W–0 14 <b>CAEN†</b> R/W–0	R/W-0 13 <b>CACLR</b> † R/W-0	R/W-0 12 HINT‡ R/W-1 4	R/W–0 11 R	R/W–0 eserved (alwa	R/W–0	AROLC R/W–0 8

#### Figure 5–3. Status Registers Bit Mapping

**Legend:** R = Read; W = Write; -n = Value after reset

<sup>†</sup> Highlighted bit: If you write to the protected address of the status register, a write to this bit has no effect, and the bit always appears as a 0 during read operations.

<sup>‡</sup> The HINT bit is not used for all C55x host port interfaces (HPIs). Consult the documentation for the specific C55x DSP.

§ The reset value of MPNMC may be dependent on the state of predefined pins at reset. To check this for a particular C55x DSP, see the boot loader section of its data sheet.

5-388 Instruction Set Descriptions

# Shift Accumulator Content Conditionally

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACx = sftc(AC	к, <b>ТС1)</b>		Yes	2	1	Х	
[2]	ACx = sftc(AC	≺, <b>TC2)</b>		Yes	2	1	Х	
Opcod	e	TC1		01	01 10	)1E DDx	x xx10	
		TC2		01	01 10	)1E DDx	x xx11	
Operar	nds	ACx, TCx						
Descri	ption	If the source ac status bit to 1.	cumulator ACx(39–0)	is equal to 0, t	this inst	truction se	ets the TCx	
		If the source accumulator ACx(31–0) has two sign bits:						
		this instruction shifts left the 32-bit accumulator ACx by 1 bit						
		☐ the TCx status bit is cleared to 0						
			accumulator ACx(31- the TCx status bit to	,	t have	two sigr	n bits, this	
		The sign bits a	re extracted at bit pos	itions 31 and	30.			
Status	Bits	Affected by	none					
		Affects	TCx					
Repeat	t	This instructior	n can be repeated.					
See Al	SO	See the followi	ng other related instru	ictions:				
		Shift Accur	mulator Content Logic	ally				
		Shift Accur	nulator, Auxiliary, or Te	mporary Regi	ster Co	ontent Log	ically	
		Signed Sh	ift of Accumulator Cor	ntent				
		Signed Shi	ift of Accumulator, Aux	kiliary, or Terr	porary	Register	Content	

#### Example 1

Syntax			Desci	ription				
AC0 = sftc(AC0, TC1)				Because AC0(31) XORed with AC0(30) equals 1, the content of AC0 is not shifted eft and TC1 is set to 1.				
Before				After				
AC0	FF	8765	0055	AC0	FF 8765 0055			
TC1			0	TC1	1			

# Example 2

Syntax	Description
AC0 = sftc(AC0, TC2)	Because AC0(31) XORed with AC0(30) equals 0, the content of AC0 is shifted left by 1 bit and TC2 is cleared to 0.

Before				After			
AC0	00	1234	0000	AC0	00	2468	0000
TC2			0	TC2			0

# Shift Accumulator Content Logically

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	ACy = ACx <<<	Tx		Yes	2	1	Х		
[2]	ACy = ACx <<<	#SHIFTW		Yes	3	1	Х		
Description		These instructions perform an unsigned shift by an immediate value, SHIFTW, or the content of a temporary register (Tx) in the D-unit shifter.							
Status	Bits	Affected by	C54CM, M40						
		Affects	CARRY						
See Als	50	See the following other related instructions:							
		Shift Accur	mulator Content Cond	itionally					
		Shift Accur	nulator, Auxiliary, or Te	mporary Regis	ster Co	ntent Log	ically		
		Signed Sh	ift of Accumulator Cor	itent					
		Signed Sh	ift of Accumulator, Aux	kiliary, or Tem	oorary	Register	Content		

# Shift Accumulator Content Logically

# Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1] ACy = ACx <<<	Tx	Yes	2	1	Х			
Opcode		010	)1 11	.0E DDS	S ss00			
Operands	ACx, ACy, Tx							
Description	This instruction shifts by the ta (ACx) content and stores the 16-bit value contained in Tx is to $-32$ or $+31$ and the shift oper overflow is reported when suc	e shifted-out bit in the out of the –32 to +31 r eration is performed w	e CARI ange, t	RY status the shift is	s bit. If the saturated			
	The operation is performed on 40 bits in the D-unit shifter.							
	The shift operation is performed according to M40.							
	The CARRY status bit contains the shifted-out bit. When the shift count is zero, $Tx = 0$ , the CARRY status bit is cleared to 0.							
	Compatibility with C54x devices (C54CM = 1)							
	When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the 6 LSBs of Tx define the shift quantity within $-32$ to $+31$ . When the value is between $-32$ to $-17$ , a modulo 16 operation transforms the shift quantity to within $-16$ to $-1$ .							
Status Bits	Affected by C54CM, M40	)						
	Affects CARRY							
Repeat	This instruction can be repea	ted.						

Syntax		Descr	Description				
st			stored	The content of AC0 is logically shifted right by the content of T0 and the result is stored in AC1. There is a right shift because the content of T0 is negative ( $-6$ ). Because M40 = 0, the guard bits (39–32) are cleared.			
Before				After			
AC0	5F	в000	1234	AC0	5F B000 1234		
AC1	00	C680	ACF0	AC1	00 02C0 0048		
т0			FFFA	тО	FFFA		
M40			0	M40	0		

# Shift Accumulator Content Logically

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[2]	ACy = ACx <<<	<b>#SHIFTW</b>		Yes	3	1	Х	
Opcod	e		0001	000E DDS	SS 01	.11 xxS	H IFTW	
Operar	nds	ACx, ACy, SH	IFTW					
Descri	ption	This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit.						
		The opera	tion is performed on 40	bits in the D	-unit s	hifter.		
		The shift of	peration is performed a	according to I	M40.			
			RY status bit contains the $TW = 0$ , the CARRY st				nift count is	
		Compatibility	with C54x devices (C	54CM = 1)				
		When this inst	ruction is executed with	M40 = 0, co	mpatik	oility is en	sured.	
Status	Bits	Affected by	M40					
		Affects	CARRY					
Repeat	t	This instruction	n can be repeated.					
Examp	le							
Syntax	1	Description						
AC0 = .	AC1 <<< #31	The content of A	C1 is logically shifted left I	by 31 bits and	the res	sult is store	ed in AC0.	

Shift Accumulator, Auxiliary, or Temporary Register Content Logically

					Paralle			
No.	Syntax				Enable E	Bit Size	Cycles	Pipeline
[1]	dst = dst <<< #1				Yes	2	1	Х
[2]	dst = dst >>> #1				Yes	2	1	Х
Descri	otion	The	ese instructio	ons perform an un	signed shift by	/ 1 bit:		
			In the D-un	it shifter, if the des	stination opera	ind is an a	accumula	tor (ACx).
			In the A-un register (TA	it ALU, if the desti Ax).	nation operan	d is an au	ixiliary or	temporary
Status	Bits	Aff	ected by	C54CM, M40				
		Aff	ects	CARRY				
See Als	50	Se	e the followi	ng other related in	structions:			
			Shift Accur	nulator Content Co	onditionally			
			Shift Accur	nulator Content Lo	ogically			
			Signed Shi	ft of Accumulator	Content			
			Signed Shi	ft of Accumulator,	Auxiliary, or Te	emporary	Register	Content

# Shift Accumulator, Auxiliary, or Temporary Register Content Logically

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst <<< #1					Yes	2	1	Х
Opcod	e					01	01 00	)0e FDI	DD x000
Operar	nds	dst							
Descri	ption				n shifts left by 1 bit th e shifted-out bit.	e input operan	d (dst).	The CAF	RRY status
			When	the	destination operand	(dst) is an accu	umulate	or:	
			∎ Th	ne op	peration is performed	d on 40 bits in t	he D-u	nit shifter	:
			■ 0 is	s ins	serted at bit position	0.			
			∎ Th	ne sh	ifted-out bit is extrac	cted at a bit pos	sition a	ccording	to M40.
			When	the	destination operand	(dst) is an auxi	liary or	tempora	ry register
			∎ Th	ne op	peration is performed	d on 16 bits in t	he A-u	nit ALU.	
			■ 0 i	s ins	serted at bit position	0.			
					nifted-out bit is extra Y status bit.	acted at bit pos	ition 1	5 and sto	ored in the
		Con	npatib	oility	with C54x devices	(C54CM = 1)			
		Whe	en this	inst	ruction is executed w	vith M40 = 0, co	ompatil	oility is er	nsured.
Status	Bits	Affe	cted by	y	M40				
		Affe	cts		CARRY				
Repeat	t	This	s instru	ctior	n can be repeated.				
<b>-</b>									

#### Syntax Characteristics

#### Example

Syntax	Description
AC1 = AC1 <<< #1	The content of AC1 is logically shifted left by 1 bit and the result is stored in AC1. Because $M40 = 0$ , the CARRY status bit is extracted at bit 31 and the guard bits (39–32) are cleared.
Before	After

Derore		ALCEL	
AC1	8F E340 5678	AC1	00 C680 ACF0
CARRY	0	CARRY	1
M40	0	М40	0

SPRU375G

Instruction Set Descriptions 5-395

#### Shift Accumulator, Auxiliary, or Temporary Register Content Logically

#### **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[2] dst = dst >>> #1		Yes	2	1	Х			
Opcode		010	01 00	0E FDD	D x001			
Operands	dst							
Description	This instruction shifts right by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.							
	U When the destination operand (d	st) is an accu	umulato	or:				
	The operation is performed of	n 40 bits in tl	he D-u	nit shifter				
	0 is inserted at a bit position	according to	M40.					
	The shifted-out bit is extract CARRY status bit.	ed at bit po	sition (	) and sto	ored in the			
	U When the destination operand (d	st) is an auxi	liary or	tempora	ry register:			
	The operation is performed or	n 16 bits in tl	he A-ui	nit ALU.				
	0 is inserted at bit position 15	ö.						
	The shifted-out bit is extract CARRY status bit.	ed at bit po	sition (	) and sto	ored in the			
	Compatibility with C54x devices (C	C54CM = 1)						
	When this instruction is executed with	n M40 = 0, co	ompatik	oility is en	sured.			
Status Bits	Affected by M40							
	Affects CARRY							
Repeat	This instruction can be repeated.							
Example								
Syntax	Description							

The content of AC0 is logically shifted right by 1 bit and the result is stored in AC0.

 $\mathsf{AC0}=\mathsf{AC0}>>>\#1$ 

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx << Tx	Yes	2	1	Х
[2]	ACy = ACx <b>&lt;<c< b=""> Tx</c<></b>	Yes	2	1	Х
[3]	ACy = ACx << #SHIFTW	Yes	3	1	х
[4]	ACy = ACx <b>&lt;<c #<="" b="">SHIFTW</c></b>	Yes	3	1	х

Description		ons perform a signed shift by an immediate value, SHIFTW, or of a temporary register (Tx) in the D-unit shifter.
Status Bits	Affected by	C54CM, M40, SATA, SATD, SXMD
	Affects	ACOVx, ACOVy, CARRY
See Also	See the following	ng other related instructions:
	Shift Accur	nulator Content Conditionally
	Shift Accur	nulator Content Logically
	Shift Accun	nulator, Auxiliary, or Temporary Register Content Logically
	Signed Shi	ift of Accumulator, Auxiliary, or Temporary Register Content

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ACy = ACx <<	тх		Yes	2	1	Х
Opcod	e			01	01 1	10E DDS	S ss01
Operar	nds	ACx,	ACy, Tx				
Description		(AC× the s	instruction shifts by the tem c) content. If the 16-bit value hift is saturated to –32 or +3 c; a destination accumulato rs.	contained in Tx is I and the shift ope	out of t ration i	he –32 to s performe	+31 range ed with this
			The operation is performed	on 40 bits in the [	D-unit s	shifter.	
			When M40 = 0, the input to the the the modified input is sh			ording to	SXMD and
		I	If SXMD = 0, 0 is substitution instead of ACx(39−32),	-	d bits (	39–32) as	s the input
		I	If SXMD = 1, bit 31 of the bits (39–32) as the input	•			-
			The sign position of the sour This comparison depends o	•	nparec	I to the shi	ift quantity
		I	■ if M40 =0, comparison is	s performed versu	us bit 3	1	
		I	if M40 =1, comparison is	s performed versu	us bit 3	9	
			) is inserted at bit position 0				
			The shifted-out bit is extract	ed according to M	140.		
			After shifting, unless otherw	ise noted, when N	/140 = 0	D:	
		I	<ul> <li>overflow is detected at a destination ACOVy bit is</li> </ul>	•	in over	flow is de	tected, the
		I	<ul> <li>if SATD = 1, when accumulator saturation overflow) or FF 8000 00</li> </ul>	values are 00	) 7FF	F FFFFh	

Example	
•	
Repeat	This instruction can be repeated.
	Affects ACOVy
Status Bits	Affected by C54CM, M40, SATD, SXMD
	□ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to +31. When the value is between -32 to -17, a modulo 16 operation transforms the shift quantity to within -16 to -1.
	There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.
	These instructions are executed as if M40 status bit was locally set to 1.
	When $C54CM = 1$ :
	Compatibility with C54x devices (C54CM = 1)
	■ if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)
	<ul> <li>overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)</li> </ul>
	$\Box$ After shifting, unless otherwise noted, when M40 = 1:

Syntax	Description
AC0 = AC1 << T0	The content of AC1 is shifted by the content of T0 and the result is stored in AC0.

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ACy = ACx <b>&lt;<c< b=""></c<></b>	Тx		Yes	2	1	Х
Opcod	e			01	01 1	10e DDS	S ss10
Operar	nds	AC	х, АСу, Тх				
Descri	ption	(AC 16- to -	s instruction shifts by the tempora Cx) content and stores the shifted bit value contained in Tx is out of t -32 or +31 and the shift operation i cumulator overflow is reported wh	d-out bit in th he–32 to +31 s performed v	e CAR range, vith this	RY status the shift is value; a c	s bit. If the ssaturated
			The operation is performed on 4	0 bits in the D	D-unit s	shifter.	
			When M40 = 0, the input to the sl then the modified input is shifted			ording to	SXMD and
			■ if SXMD = 0, 0 is substituted instead of ACx(39–32), to the	•	d bits (	39–32) as	s the input
			■ if SXMD = 1, bit 31 of the so bits (39–32) as the input, ins	•			-
			The sign position of the source of This comparison depends on M4	•	nparec	I to the shi	ift quantity.
			■ if M40 =0, comparison is pe	rformed versu	us bit 3	1	
			■ if M40 =1, comparison is pe	rformed versu	us bit 3	9	
			0 is inserted at bit position 0.				
			The shifted-out bit is extracted ac status bit. When the shift count cleared to 0.	-			
			After shifting, unless otherwise r	noted, when N	/140 = 0	D:	
			<ul> <li>overflow is detected at bit per destination ACOVy bit is set</li> </ul>	•	in over	flow is de	tected, the
			■ if SATD = 1, when an accumulator saturation va overflow) or FF 8000 0000h	alues are 00	) 7FF	F FFFFh	

	After shifti	ng, unless otherwise noted, when $M40 = 1$ :
		ow is detected at bit position 39 (if an overflow is detected, the ation ACOVy bit is set)
	accum	TD = 1, when an overflow is detected, the destination nulator saturation values are 7F FFFF FFFFh (positive ow) or 80 0000 0000h (negative overflow)
	Compatibility	with C54x devices (C54CM = 1)
	When C54CM	= 1:
	These inst	ructions are executed as if M40 status bit was locally set to 1.
	There is no by the D-u	o overflow detection, overflow report, and saturation performed nit shifter.
	Tx define a	is of Tx are used to determine the shift quantity. The 6 LSBs of a shift quantity within $-32$ to $+31$ . When the value is between $-32$ nodulo 16 operation transforms the shift quantity to within $-16$
Status Bits	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	n can be repeated.

Syntax	Description
AC2 = AC2 < <c t1<="" td=""><td>The content of AC2 is shifted left by the content of T1 and the saturated result is stored in AC2. The shifted out bit is stored in the CARRY status bit. Since SATD = 1 and M40 = 0, AC2 = FF 8000 0000 (saturation).</td></c>	The content of AC2 is shifted left by the content of T1 and the saturated result is stored in AC2. The shifted out bit is stored in the CARRY status bit. Since SATD = 1 and M40 = 0, AC2 = FF 8000 0000 (saturation).

Before			After				
AC2	80 AA(	00 1234	AC2	FF	8000	0000	
т1		0005	т1			0005	
CARRY		0	CARRY			1	
M40		0	M40			0	
ACOV2		0	ACOV2			1	
SXMD		1	SXMD			1	
SATD		1	SATD			1	

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ACy = ACx <<	< #SHIFT	W		Yes	3	1	Х
Opcod	e			0001	000E DD	SS 01	101 xxS	H IFTW
Operar	nds	ACx,	ACy, SHIFTW	,			ļ	
Descri	ption	This conte	instruction shifts be	ue, SHIFTV	V, the	accumula	ator (ACx	
		П	he operation is perf	ormed on 40	bits in the D	)-unit s	shifter.	
			Vhen M40 = 0, the in hen the modified inp	•			-	SXMD and
			if SXMD = 0, 0 is instead of ACx(3		-	d bits (	39–32) as	the inpu
			I if SXMD = 1, bit 3 bits (39–32) as t		•			-
			he sign position of t his comparison dep			npared	l to the shi	ft quantity
			if M40 =0, compa	arison is perf	ormed versu	ıs bit 3	1	
			if M40 =1, compa	arison is perf	ormed versu	ıs bit 3	9	
			is inserted at bit po	sition 0.				
		T []	he shifted-out bit is	extracted ac	cording to N	I40.		
		<b>_</b> A	fter shifting, unless	otherwise no	ted, when N	140 = 0	):	
			<ul> <li>overflow is detected destination ACO</li> </ul>		sition 31 (if a	n over	flow is de	tected, th
			if SATD = 1, accumulator sa overflow) or FF 8	turation valu	ues are 00	) 7FF		
		<b>_</b> A	fter shifting, unless	otherwise no	oted, when N	140 = 1	1:	
			overflow is detected destination ACO		sition 39 (if a	n over	flow is de	tected, th
			I if SATD = 1, accumulator sa overflow) or 80 0	turation valu	ues are 7F	FFF		

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

 Status Bits
 Affected by
 C54CM, M40, SATD, SXMD

 Affects
 ACOVy

**Repeat** This instruction can be repeated.

#### Example 1

Syntax	Description
AC0 = AC1 << #31	The content of AC1 is shifted left by 31 bits and the result is stored in AC0.

Syntax	Description
AC0 = AC1 << #-32	The content of AC1 is shifted right by 32 bits and the result is stored in AC0.

						Parallel	0.	<u> </u>	<b>D</b> :		
No.	Syntax					Enable Bit	Size	Cycles	Pipeline		
[4]	ACy = ACx <b>&lt;<c< b=""></c<></b>	; #S⊦	lIFT∖	V		Yes	3	1	Х		
Opcode	e				0001	000E DD	SS 0	110 xx8	GH IFTW		
Operar	nds	AC	x, A	Cy, SHIFTW							
Description		This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) co and stores the shifted-out bit in the CARRY status bit.									
			The	operation is p	performed on 40	) bits in the [	D-unit s	shifter.			
					e input to the sh input is shifted			according to SXMD and le, SHIFTW:			
					0 is substituted x(39–32), to the	-	d bits (	(39–32) as	s the input,		
					bit 31 of the sou as the input, inst	•			-		
					of the source op depends on M4		nparec	l to the sh	ift quantity.		
				if M40 =0, co	mparison is perf	ormed versu	us bit 3	51			
				if M40 =1, co	mparison is perf	ormed versu	us bit 3	9			
			0 is	inserted at bit	t position 0.						
			sta		t is extracted acc he shift count is	-					
			Afte	er shifting, unle	ess otherwise no	oted, when N	/140 = 0	0:			
					etected at bit pos COVy bit is set)	sition 31 (if a	an over	flow is de	tected, the		
			•	accumulator	1, when an o saturation val F 8000 0000h (	ues are 00	) 7FF	F FFFF			

 $\Box$  After shifting, unless otherwise noted, when M40 = 1:

- overflow is detected at bit position 39 (if an overflow is detected, the destination ACOVy bit is set)
- if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 7F FFFF FFFFh (positive overflow) or 80 0000 0000h (negative overflow)

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	can be repeated.

Syntax I			Descr	Description						
					s shifted right by 5 bits and the result is stored in AC1. The shifted e CARRY status bit.					
Before				After						
AC0	FF	8765	0055	AC0	FF 8765 0055					
AC1	00	4321	1234	AC1	FF FC3B 2802					
CARRY			0	CARRY	1					
SXMD			1	SXMD	1					

Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	dst = dst >> #1				Yes	2	1	X	
[2]	dst = dst << #1				Yes	2	1	X	
							· ·		
Descrip	otion	The	ese instructi	ons perform a shift o	f 1 bit:				
			In the D-un	nit shifter, if the destir	nation operand	l is an a	accumula	tor (ACx).	
			In the A-un register (TA	it ALU, if the destina Ax).	tion operand i	s an ai	uxiliary or	temporary	
Status	Bits	Affected by C54CM, M40, SATA, SATD, SXMD							
		Aff	ects	ACOVx, ACOVy, C	ARRY				
See Als	50	See the following other related instructions:							
			Shift Accur	nulator Content Con	ditionally				
			Shift Accur	mulator Content Logi	cally				
			Shift Accun	nulator, Auxiliary, or To	emporary Regi	ster Co	ontent Log	ically	
			Signed Shi	ft of Accumulator Co	ntent				

# Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst >> <b>#1</b>			Yes	2	1	Х
Opcode	9			0	100 0	10E 01×	0 FDDD
Operands							
Description		Thi	s instruction shifts right l	by 1 bit the content of t	he dest	ination reg	gister (dst).
		lf tł	ne destination operand (	(dst) is an accumulato	r:		
			The operation is perfor	rmed on 40 bits in the	D-unit s	shifter.	
			When M40 = 0, the input then the modified input			cording to	SXMD and
			■ if SXMD = 0, 0 is s instead of ACx(39-	substituted for the gua -32), to the shifter	rd bits (	(39–32) as	s the input,
				of the source operand input, instead of ACx			-
			Bit 39 is extended acco	ording to SXMD			
			The shifted-out bit is ex	xtracted at bit position	0.		
			After shifting, unless of	therwise noted, when	M40 = 0	0:	
			overflow is detected	ed at bit position 31			
			accumulator satu	hen an overflow is ration values are 0 00 0000h (negative ov	0 7FF	F FFFFh	
			After shifting, unless of	therwise noted, when	M40 =	1:	
			<ul> <li>overflow is detected</li> </ul>	ed at bit position 39			
			accumulator satu	hen an overflow is ration values are 7 00 0000h (negative ov	F FFF		

If the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- Bit 15 is sign extended.
- After shifting, unless otherwise noted:
  - overflow is detected at bit position 15
  - if SATA = 1, when an overflow is detected, the destination register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits	Affected by	C54CM, M40, SATA, SATD, SXMD		
	Affects	none		
Repeat	This instruction can be repeated.			

Syntax	Description
AC0 = AC0 >> #1	The content of AC0 is shifted right by 1 bit and the result is stored in AC0.

# Signed Shift of Accumulator, Auxiliary, or Temporary Register Content

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[2]	dst = dst << #1			Yes	2	1	Х
Opcod	6			01	00 02	10E 01x	1 FDDD
Operands				·		·	
Descri	otion	This instruction shifts left by 1 bit the content of the destination register (dst).					
		lf th	ne destination operand (dst) is a	n accumulator	:		
			The operation is performed on	40 bits in the E	D-unit s	shifter.	
			When $M40 = 0$ , the input to the then the modified input is shifted		ed acc	ording to	SXMD an
			■ if SXMD = 0, 0 is substitute instead of ACx(39–32), to t	-	d bits (	39–32) as	s the inpu
			■ if SXMD = 1, bit 31 of the s bits (39–32) as the input, in	•			-
			The sign position of the source This comparison depends on M	-	npared	I to the sh	ift quantit
			■ if M40 =0, comparison is p	erformed versu	ıs bit 3	1	
			■ if M40 =1, comparison is p	erformed versu	ıs bit 3	9	
			0 is inserted at bit position 0.				
			The shifted-out bit is extracted	according to N	140.		
			After shifting, unless otherwise	noted, when N	/140 = 0	D:	
			<ul> <li>overflow is detected at bit p destination ACOVx bit is set</li> </ul>	•	in over	flow is de	tected, th
			■ if SATD = 1, when an accumulator saturation workflow) or FF 8000 0000	alues are 00	) 7FF	F FFFFh	
			After shifting, unless otherwise	noted, when N	/140 = ^	1:	
			<ul> <li>overflow is detected at bit p destination ACOVx bit is set</li> </ul>	•	in over	flow is de	tected, th
			■ if SATD = 1, when an accumulator saturation v overflow) or 80 0000 0000	alues are 7F	FFF		

If the destination operand (dst) is an auxiliary or temporary register:

- The operation is performed on 16 bits in the A-unit ALU.
- 0 is inserted at bit position 0.
- After shifting, unless otherwise noted:
  - overflow is detected at bit position 15 (if an overflow is detected, the destination ACOVx bit is set)
  - if SATA = 1, when an overflow is detected, the destination register saturation values are 7FFFh (positive overflow) or 8000h (negative overflow)

#### Compatibility with C54x devices (C54CM = 1)

When C54CM = 1, these instructions are executed as if M40 status bit was locally set to 1. There is no overflow detection, overflow report, and saturation performed by the D-unit shifter.

Status Bits	Affected by	C54CM, M40, SATA, SATD, SXMD		
	Affects	ACOVx		
Repeat	This instruction can be repeated.			

Syntax	Description					
T2 = T2 << #1		The content of T	The content of T2 is shifted left by 1 bit and the result is stored in T2.			
Before		After				
Т2	EF27	Т2	DE4E			
SATA	1	SATA	1			

Software Interrupt

	•	Parallel							
No.	Syntax	Enable Bit Size Cycles Pipeline							
[1]	intr(k5)	No 2 3 D							
Opcod	e	1001 0101 0xxk kkkk							
Operar	nds	k5							
Descrij	otion	This instruction passes control to a specified interrupt service routine (ISR) and interrupts are globally disabled (INTM bit is set to 1 after ST1_55 content is pushed onto the data stack pointer). The ISR address is stored at the interrupt vector address defined by the content of an interrupt vector pointer (IVPD or IVPH) combined with the 5-bit constant, k5. This instruction is executed regardless of the value of INTM bit.							
		Note:							
		DBSTAT (the debug status register) holds debug context information used during emulation. Make sure the ISR does not modify the value that will be returned to DBSTAT.							
		Before beginning an ISR, the CPU automatically saves the value of some CPU registers and two internal registers: the program counter (PC) and a loop context register. The CPU can use these values to re-establish the context of the interrupted program sequence when the ISR is done.							
		In the slow-return process (default), the return address (from the PC), the loop context bits, and some CPU registers are stored to the stacks (in memory). When the CPU returns from an ISR, the speed at which these values are restored is dependent on the speed of the memory accesses.							
	In the fast-return process, the return address (from the PC) and the loop context bits are saved to registers, so that these values can always be restored quickly. These special registers are the return address register (RETA) and the control-flow context register (CFCT). You can read from or write to RETA and CFCT as a pair with dedicated, 32-bit load and store instructions. Some CPU registers are saved to the stacks (in memory). For fast-return mode operation see the <i>TMS320C55x DSP CPU Reference Guide</i> (SPRU371).								
		When control is passed to the ISR:							
		The data stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The status register 2 (ST2_55) content is pushed to the top of SP.							

- □ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 7 higher bits of status register 0 (ST0\_55) concatenated with 9 zeroes are pushed to the top of SSP.
- □ The SP is decremented by 1 word in the access phase of the pipeline. The status register 1 (ST1\_55) content is pushed to the top of SP.
- The SSP is decremented by 1 word in the access phase of the pipeline. The debug status register (DBSTAT) content is pushed to the top of SSP.
- ☐ The SP is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- The SSP is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ☐ The PC is loaded with the ISR program address. The active control flow execution context flags are cleared.

		System Stack (SSP)		Data Stack (SP)		
After $\rightarrow$	SSP = x - 3	(Loop bits):PC(23-16)	After $\rightarrow$ SP = y - 3	PC(15–0)		
Save	SSP = x - 2	DBSTAT	<b>Save</b> SP = y - 2	ST1_55		
	SSP = x - 1	ST0_55(15–9)	SP = y - 1	ST2_55		
Before $\rightarrow$	SSP = x	Previously saved data	Before $\rightarrow$ SP = y	Previously saved data		
Save			Save			
<b>0</b> , 1 <b>D</b> ,		<i></i>				
Status Bits	A	ffected by none				
	A	ffects INTM				
Repeat	This instruction cannot be repeated.					
See Also	See the following other related instructions:					
		Return from Interrup	t			
		Software Trap				

Syntax	Description
intr(#3)	Program control is passed to the specified interrupt service routine. The interrupt vector address is
	defined by the content of an interrupt vector pointer (IVPD) combined with the unsigned 5-bit value (3).

# Software Reset

				Parallel			
No.	Syntax			Enable bit	Size	Cycles	Pipeline
[1]	reset			No	2	?	D
Opcod	e			10	01 03	100 xxx	x xxxx
Operar	nds	none					
Descri	ption	This instruction performs a nonmaskable software reset that can be used ar time to put the device in a known state.					e used any
		The reset instruction affects ST0_55, ST1_55, ST2_55, IFR0, IFR1, and T2 (Table 5–5 and Figure 5–4); status register ST3_55 and interrupt vectors pointer registers (IVPD and IVPH) are not affected. When the reset instruction is acknowledged, the INTM is set to 1 to disable maskable interrupts. All pending interrupts in IFR0 and IFR1 are cleared. The initialization of the system control register, the interrupt vectors pointer, and the peripheral registers is different from the initialization performed by a hardware reset.					instruction errupts. All tion of the peripheral
Status Bits Affected by none							
		Affects	IFR0, IFR1, ST0_	_55, ST1_55, ST	2_55		
Repeat	t	This instruction	on cannot be repeate	ed.			

Register	Bit	Reset Value	Comment
T2	All	0	All bits are cleared. To ensure TMS320C54x DSP compatibility, instructions affected by ASM bit will use a shift count of 0 (no shift).
IFR0	All	0	All pending interrupt flags are cleared.
IFR1	All	0	All pending interrupt flags are cleared.
ST0_55	ACOV2	0	AC2 overflow flag is cleared.
	ACOV3	0	AC3 overflow flag is cleared.
	TC1	1	Test control flag 1 is cleared.
	TC2	1	Test control flag 2 is cleared.
	CARRY	1	CARRY bit is cleared.
	ACOV0	0	AC0 overflow flag is cleared.
	ACOV1	0	AC1 overflow flag is cleared.
	DP	0	All bits are cleared, data page 0 is selected.
ST1_55	BRAF	0	This flag is cleared.
	CPL	0	The DP (rather than SP) direct addressing mode is selected. Direct accesses to data space are made relative to the data page register (DP).
	XF	1	External flag is set.
	HM	0	When an active HOLD signal forces the DSP to place its external interface in the high-impedance state, the DSP continues executing code from internal memory.
	INTM	1	Maskable interrupts are globally disabled.
	M40	0	32-bit (rather than 40-bit) computation mode is selected for the D unit.
	SATD	0	CPU will not saturate overflow results in the D unit.
	SXMD	1	Sign-extension mode is on.
	C16	0	Dual 16-bit mode is off. For an instruction that is affected by C16, the D- unit ALU performs one 32-bit operation rather than two parallel 16-bit op- erations.
	FRCT	0	Results of multiply operations are not shifted.
	C54CM	1	TMS320C54x-compatibility mode is on.
	ASM	0	Instructions affected by ASM will use a shift count of 0 (no shift).

Table 5–5. Effects of a Software Reset on DSP Registers

Register	Bit	Reset Value	Comment
ST2_55	ARMS	0	When you use the AR indirect addressing mode, the DSP mode (rather than control mode) operands are available.
	DBGM	1	Debug events are disabled.
	EALLOW	0	A program cannot write to the non-CPU emulation registers.
	RDM	0	When an instruction specifies that an operand should be rounded, the CPU uses rounding to the infinite (rather than rounding to the nearest).
	CDPLC	0	CDP is used for linear addressing (rather than circular addressing).
	AR7LC	0	AR7 is used for linear addressing.
	AR6LC	0	AR6 is used for linear addressing.
	AR5LC	0	AR5 is used for linear addressing.
	AR4LC	0	AR4 is used for linear addressing.
	AR3LC	0	AR3 is used for linear addressing.
	AR2LC	0	AR2 is used for linear addressing.
	AR1LC	0	AR1 is used for linear addressing.
	AR0LC	0	AR0 is used for linear addressing.

Table 5–5. Effects of a Software Reset on DSP Registers (Continued)

ST0_55										
15	14	13		1	2		11	10	9	
ACOV2	ACOV3	TC1		TC	C2	C	ARRY	ACOV0	ACOV1	
0	0	1		1	1		1	0	0	
8					<u> </u>					0
					P )					
				C	J					
ST1_55									-	
15	14	13		12	11		10	9	8	
BRAF	CPL	XF		IM	INT	M	M40	SATD	SXMD	
0	0	1		0	1		0	0	1	
7	0	-								0
7	6	5	4				4 0 1 4			0
C16	FRCT	C54CM					ASM			
0	0	1					0			
ST2_55										
15	14	13		12	11		10	9	8	
ARMS	Rese	erved	DB	BGM	EALLO	SW	RDM	Reserved	CDPLC	;
0				1	0		0		0	
_	0	_			-					
7	6	5		4	3		2	1	0	
AR7LC	AR6LC	AR5LC		4LC	AR3L	_C	AR2LC	AR1LC	AR0LC	
0	0	0		0	0		0	0	0	

Figure 5–4. Effects of a Software Reset on Status Registers

Software Trap

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	trap(k5)		No	2	?	D			
Opcod	e		10	01 01	101   1xx	k kkkk			
Operar	nds	k5							
Description		This instruction passes control and this instruction does not aff stored at the interrupt vector ad vector pointer (IVPD or IVPH) instruction is executed regardles not maskable.	ect INTM bit in S Idress defined by combined with th	T1_55. the co ne 5-bi	The ISR ntent of a t constan	address is an interrup t, k5. This			
		Note:							
		DBSTAT (the debug status register) holds debug context information used during emulation. Make sure the ISR does not modify the value that will be returned to DBSTAT.							
		Before beginning an ISR, the CP registers and two internal regis context register. The CPU can u the interrupted program sequen	ters: the program se these values to	n count o re-est	ter (PC) a ablish the	and a loop			
		stacks (ir	om the PC), the loop stacks (in memory). h these values are sses.						
		In the fast-return process, the context bits are saved to registers quickly. These special registers a control-flow context register (CF CFCT as a pair with dedicated, 3 registers are saved to the stacks see the TMS320C55x DSP CPU	s, so that these val are the return addr CT). You can read 32-bit load and sto (in memory). For	lues ca ess reg d from c ore insti fast-ret	n always b ister (RE or write to ructions. s urn mode	be restored TA) and the RETA and Some CPU			
		When control is passed to the ISR:							
		The data stack pointer (SP phase of the pipeline. The s to the top of SP.		-					

- □ The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 7 higher bits of status register 0 (ST0\_55) concatenated with 9 zeroes are pushed to the top of SSP.
- □ The SP is decremented by 1 word in the access phase of the pipeline. The status register 1 (ST1\_55) content is pushed to the top of SP.
- The SSP is decremented by 1 word in the access phase of the pipeline. The debug status register (DBSTAT) content is pushed to the top of SSP.
- □ The SP is decremented by 1 word in the read phase of the pipeline. The 16 LSBs of the return address, from the program counter (PC), of the called subroutine are pushed to the top of SP.
- The SSP is decremented by 1 word in the read phase of the pipeline. The loop context bits concatenated with the 8 MSBs of the return address are pushed to the top of SSP.
- ☐ The PC is loaded with the ISR program address. The active control flow execution context flags are cleared.

		System Stack (SSP)	_	Data Stack (SP)
After $\rightarrow$	SSP = x - 3	(Loop bits):PC(23-16)	After $\rightarrow$ SP = y - 3	PC(15–0)
Save	SSP = x - 2	DBSTAT	<b>Save</b> SP = y - 2	ST1_55
	SSP = x - 1	ST0_55(15–9)	SP = y - 1	ST2_55
Before $\rightarrow$	SSP = x	Previously saved data	Before $\rightarrow$ SP = y	Previously saved data
Save			Save	
Status Bits		ffected by none		
Demost	т	This is started as a second to		
Repeat	I	his instruction cannot b	e repeated.	
See Also	S	See the following other r	elated instructions:	
		Return from Interrup	t	
		Software Interrupt		

Syntax	Description
trap(5)	Program control is passed to the specified interrupt service routine. The interrupt vector address is
	defined by the content of an interrupt vector pointer (IVPD) combined with the unsigned 5-bit value (5).

Square

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = rnd(ACx)	* ACx)		Yes	2	1	X	
[2]	ACx = <mark>rnd(</mark> Sme	em * Smem)[, T3 =	Smem]	No	3	1	х	
Descriț	otion	This instruction operands of the	n performs a multiplic e multiplier are:	cation in the	D-un	it MAC.	The input	
		<ul><li>ACx(32–16</li><li>the content</li></ul>	) of a memory (Smem)	location, sig	n exter	nded to 1	7 bits	
Status Bits		Affected by FRCT, M40, RDM, SATD, SMUL						
		Affects	ACOVx, ACOVy					
See Als	50	See the following other related instructions:						
		Multiply						
		Square and Accumulate						
		Square and	Subtract					
		Square Dis	tance					

Square

Square

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = rnd(ACx	* AC	x)			Yes	2	1	Х	
Opcode	9					010	01 01	0e DDS	S 100%	
Operan	ds	AC	x, ACy							
Descrip	otion			on performs a r ne multiplier are <i>i</i>	•		e D-un	it MAC.	The input	
			If FRCT =	1, the output of	the multip	olier is shif	ed left	by 1 bit.		
		Multiplication overflow detection depends on SMUL.								
		The 32-bit result of the multiplication is sign extended to 40 bits.								
			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
			When an overflow is detected, the accumulator is saturated according SATD.						ccording to	
		Compatibility with C54x devices (C54CM = 1)								
		Wh	nen this ins	truction is execut	ted with N	/I40 = 0, co	mpatik	oility is er	sured.	
Status	Bits	Aff	ected by	FRCT, M40, R	RDM, SAT	ATD, SMUL				
		Aff	ects	ACOVy						
Repeat		Thi	s instructio	n can be repeate	ed.					
Examp	e									

Syntax	Description
AC0 = AC1 * AC1	The content of AC1 is squared and the result is stored in AC0.

## Square

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[2]	ACx = rnd(Sm	iem * Smem <mark>)[</mark> , T3	= Smem]		No	3	1	Х		
Opcod	e			1101	0011 AAA	AA AA	AI U%I	DD 10xx		
Operai	nds	ACx, Smem								
Description		operands of	This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits.							
			☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.							
		Multiplica	Multiplication overflow detection depends on SMUL.							
		The 32-b	The 32-bit result of the multiplication is sign extended to 40 bits.							
		_	Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
		_	Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVx) is set.							
		When an SATD.	When an overflow is detected, the accumulator is saturated according to SATD.							
		This instruction provides the option to store the 16-bit data memory operand Smem in temporary register T3.								
		Compatibility with C54x devices (C54CM = 1)								
		When this in	struction is exe	cuted with	M40 = 0, co	ompatik	oility is er	sured.		
Status	Bits	Affected by	FRCT, M40	), RDM, SA	TD, SMUL					
		Affects	ACOVx							
Repea	t	This instructi	on can be repe	eated.						
Examp	ole									
Syntax	(	Description								

Syntax	Description
AC0 = *AR3 * *AR3	The content addressed by AR3 is squared and the result is stored in AC0.

# Square and Accumulate

				Parallel				
No.	Syntax			Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = rnd(ACy	+ (ACx * ACx))		Yes	2	1	Х	
[2]	ACy = rnd(ACx	+ (Smem * Smem)	) [,T3 = Smem]	No	3	1	Х	
Descrij	otion	This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are:						
<ul><li>ACx(32–16)</li><li>the content of a memory (Smem) location,</li></ul>				location, sig	n exter	nded to 1	7 bits	
Status Bits		Affected by	ATD, SMUL					
		Affects	ACOVx, ACOVy					
See Als	50	See the followir	ng other related instruc	ctions:				
		Multiply and	d Accumulate					
		Square						
	Square Distance		tance					
		Square and	I Subtract					

# Square and Accumulate

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	ACy = rnd(ACy	+ <b>(</b> A	Cx * ACx <b>)</b> )		Yes	2	1	Х		
Opcod	e				010	01 01	0e DDS	S 001%		
Operar	nds	AC	Cx, ACy							
Descri	ption			n performs a multiplica It operands of the mult				the D-unit		
			If FRCT =	1, the output of the mu	Itiplier is shif	ted left	by 1 bit.			
			Multiplicati	on overflow detection	depends on S	SMUL.				
			The 32-bit to the sour	on is sign ext	ended	to 40 bits	and added			
			•	s performed according	g to RDM, if t	he opti	onal rnd I	keyword is		
						s on M40. If an overflow is detected, v status bit (ACOVy) is set.				
			When an according	addition overflow is d to SATD.	etected, the	accum	nulator is	saturated		
		Co	ompatibility	with C54x devices (C	C54CM = 1)					
		Wł	nen this instr	ruction is executed with	n M40 = 0, co	ompatik	oility is en	sured.		
Status	Bits	Aff	ected by	FRCT, M40, RDM, S	ATD, SMUL					
		Aff	ects	ACOVy						
Repeat	t	Th	is instructior	a can be repeated.						

Syntax	Description
AC0 = AC0 + (AC1 * AC1)	The content of AC1 squared is added to the content of AC0 and the result is stored in AC0.

# Square and Accumulate

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[2]	ACy = rnd(AC	x + <b>(</b> Smem * Smen	n <b>)</b> ) [,T3 = Smem]	No	3	1	Х			
Opcod	e		1101	0010 AAA	AA AA	AI U%D	DD 10SS			
Opera	nds	ACx, ACy, Sm	em							
Descri	ption	MAC. The inpu	This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits.							
		□ If FRCT =	☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.							
		Multiplication overflow detection depends on SMUL.								
		The 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACx.								
			Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
			Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
		When an according	addition overflow is de to SATD.	etected, the	accum	nulator is	saturated			
		This instruction provides the option to store the 16-bit data memory opera Smem in temporary register T3.								
		Compatibility	with C54x devices (C	54CM = 1)						
		When this inst	ruction is executed with	M40 = 0, co	mpatik	oility is en	sured.			
Status	Bits	Affected by	Affected by FRCT, M40, RDM, SATD, SMUL							
		Affects	ACOVy							
Repeat	t	This instruction	n can be repeated.							
_										

Syntax	Description
AC0 = AC1 + (*AR3 * *AR3)	The content addressed by AR3 squared is added to the content of AC1 and the result is stored in AC0.

# Square and Subtract

	•			Parallel	<u>.</u>	<u> </u>	
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[1]	ACy = rnd(ACy	– (ACx * ACx))		Yes	2	1	Х
[2]	ACy = rnd(ACx	- (Smem * Smem	<b>)</b> )[, T3 = Smem]	No	3	1	Х
Descrij	ption	The input oper	performs a multiplication ands of the multiplier an 6) t of a memory (Smem)	re:			
				, e.g.			
Status	Bits	Affected by FRCT, M40, RDM, SATD, SMUL					
		Affects ACOVx, ACOVy					
See Als	S0	See the followi	ng other related instruc	tions:			
		Multiply an	d Subtract				
		Square					
		Square and	d Accumulate				
		Square Dis	stance				

# Square and Subtract

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	ACy = rnd(ACy	- <b>(</b> A	Cx * ACx <b>)</b> )		Yes	2	1	Х			
Opcod	e				010	01 01	0e DDS	S 010%			
Operar	nds	AC	Cx, ACy								
Descri	ption			n performs a multiplicati ands of the multiplier a			n in the D	-unit MAC.			
			I If FRCT = 1, the output of the multiplier is shifted left by 1 bit.								
			Multiplication overflow detection depends on SMUL.								
			The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACy.								
			•	is performed according the instruction.	ng to RDM, if the optional rnd keyword is						
				Overflow detection depends on M40. If an overflow is detected, the lestination accumulator overflow status bit (ACOVy) is set.							
			When an overflow is detected, the accumulator is saturated according SATD.								
		Co	ompatibility	with C54x devices (C	54CM = 1)						
		Wł	nen this inst	ruction is executed with	n M40 = 0, co	ompatik	oility is en	sured.			
Status	Bits	Aff	ected by	FRCT, M40, RDM, S/	ATD, SMUL						
		Aff	ects	ACOVy							
Repeat	:	Th	is instructior	n can be repeated.							

Syntax	Description
. , ,	The content of AC0 squared is subtracted from the content of AC1 and the result is stored in AC1.

Square and Subtract

No	Suptox					Parallel Enable Bit	Size	Cycles	Dipolino
No.	Syntax	(0	* • • • • •				Size	Cycles	Pipeline
[2]	ACy = rnd(ACX	-(S	mem " Smem	))[, T3 = Smem]		No	3	1	Х
Opcod	e				1101	0010 AAA	AA AA	AI U%D	D 11SS
Operar	nds	AC	x, ACy, Sme	em					
Descri	ption	This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits.							
		□ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.							
		Multiplication overflow detection depends on SMUL.							
		☐ The 32-bit result of the multiplication is sign extended to 40 bits and subtracted from the source accumulator ACx.							
		Rounding is performed according to RDM, if the optional rnd keyword is applied to the instruction.							
		Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.							
		When an overflow is detected, the accumulator is saturated according to SATD.							
		This instruction provides the option to store the 16-bit data memory opera							ry operand
		Co	mpatibility	with C54x dev	ices (C	54CM = 1)			
		Wł	nen this instr	uction is execut	ted with	n M40 = 0, co	mpatik	oility is en	sured.
Status	Bits	Affected by FRCT, M40, RDM, SATD, SMUL							
		Aff	ects	ACOVy					
Repeat This instruction can be repeated.									
Examp	le								

Syntax	Description
AC0 = AC1 - (*AR3 * *AR3)	The content addressed by AR3 squared is subtracted from the content of AC1 and the result is stored in AC0.

# Square Distance

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline						
[1]	sqdst(Xmerr	n, Ymem, ACx, ACy <b>)</b>	No	4	1	х						
Opcod	e	1000 0110	XXXM MMYY YM	MM D	DDD   111	.0 xxn%						
Operar	nds	ACx, ACy, Xmem, Ymem										
Descrij	ption	This instruction performs two p (MAC), and subtract:	This instruction performs two parallel operations: multiply and accumulate (MAC), and subtract:									
		ACy = ACy + (ACx * ACx), ACx = (Xmem << #16) - (Ymem << #16)										
		The first operation performs a m MAC. The input operands of the	•			n the D-unit						
		If FRCT = 1, the output of	☐ If FRCT = 1, the output of the multiplier is shifted left by 1 bit.									
		Multiplication overflow determined	Multiplication overflow detection depends on SMUL.									
		The 32-bit result of the multip to the source accumulator A	-	ended	to 40 bits	and added						
		<ul> <li>Addition overflow detection the destination accumulator</li> </ul>	-									
		When an addition overflow according to SATD.	is detected, the	accui	mulator is	saturated						
		The second operation subtracts shifted left 16 bits, from the con left 16 bits.			• •							
		The operation is performed	on 40 bits in the [	D-unit /	ALU.							
		Input operands are sign ext	ended to 40 bits a	iccordi	ng to SXN	ID.						
		The shift operation is equivalent	alent to the signed	l shift i	nstruction							
		Overflow detection and C subtraction borrow bit is rep is the logical complement or	orted in the CARF	RY stat								
		When an overflow is detected SATD.	ed, the accumulate	or is sa	aturated a	ccording to						

### Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, during the subtraction an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.

Status Bits Affected by C54CM, FRCT, M40, SATD, SMUL, SXMD

Affects ACOVx, ACOVy, CARRY

**Repeat** This instruction can be repeated.

**See Also** See the following other related instructions:

- Absolute Distance
- Square
- Square and Accumulate
- Square and Subtract

Syntax				Description	Description					
sqdst(*AR	:0, *AR	1, AC(	), AC1)	is stored in	AC1. Trom th	The con	ntent add tent addre	ded to the content of AC1 and the result Iressed by AR1 shifted left by 16 bits is essed by AR0 shifted left by 16 bits and		
Before				After						
AC0	FF	ABCD	0000	AC0	FF	FFAB	0000			
AC1	00	0000	0000	AC1	00	1BB1	8229			
*AR0			0055	*AR0			0055			
*AR1			00AA	*AR1			00AA			
ACOV0			0	ACOV0			0			
ACOV1			0	ACOV1			0			
CARRY			0	CARRY			0			
FRCT			0	FRCT			0			

### Syntax Characteristics

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	Smem = <b>HI(</b> ACx)	No	2	1	Х
[2]	Smem = HI(rnd(ACx))	No	3	1	х
[3]	Smem = <b>LO(</b> ACx << Tx <b>)</b>	No	3	1	х
[4]	Smem = <b>HI(rnd(</b> ACx << Tx <b>)</b> )	No	3	1	х
[5]	Smem = <b>LO(</b> ACx << <b>#</b> SHIFTW <b>)</b>	No	3	1	х
[6]	Smem = <b>HI(</b> ACx << <b>#</b> SHIFTW <b>)</b>	No	3	1	х
[7]	Smem = <b>HI(rnd(</b> ACx << <b>#</b> SHIFTW <b>)</b> )	No	4	1	х
[8]	Smem = <b>HI</b> (saturate(uns(rnd(ACx))))	No	3	1	х
[9]	Smem = <b>HI</b> (saturate(uns(rnd(ACx << Tx))))	No	3	1	х
[10]	Smem = <b>HI</b> (saturate(uns(rnd(ACx << #SHIFTW))))	No	4	1	х
[11]	dbl(Lmem) = ACx	No	3	1	х
[12]	dbl(Lmem) = saturate(uns(ACx))	No	3	1	х
[13]	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	No	3	1	х
[14]	Xmem = <b>LO(</b> ACx <b>)</b> , Ymem = <b>HI(</b> ACx <b>)</b>	No	3	1	Х

**Description** This instruction stores the content of the selected accumulator (ACx) to a memory (Smem) location, to a data memory operand (Lmem), or to dual data memory operands (Xmem and Ymem).

Status Bits Affected by C54CM, RDM, SXMD

Affects none

See Also	Se	e the following other related instructions:
		Addition with Parallel Store Accumulator Content to Memory
		Load Accumulator from Memory with Parallel Store Accumulator Content to Memory
		Load Accumulator, Auxiliary, or Temporary Register from Memory
		Multiply and Accumulate with Parallel Store Accumulator Content to Memory
		Multiply and Subtract with Parallel Store Accumulator Content to Memory
		Multiply with Parallel Store Accumulator Content to Memory
		Store Accumulator Pair Content to Memory
		Store Accumulator, Auxiliary, or Temporary Register Content to Memory
		Store Auxiliary or Temporary Register Pair Content to Memory
		Subtraction with Parallel Store Accumulator Content to Memory

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Smem = <b>HI(</b> AC)	×)		No	2	1	Х
Opcode	e			101	.1 11	SS AAA	IAAA A
Operan	nds	ACx, Smem					
Descrip	ption	This instruction stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location. The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.					tion uses a
Status	Bits	Affected by	none				
		Affects	none				
Repeat	:	This instructio	n can be repeated.				
Examp	le						

Syntax	Description
*AR3 = HI(AC0)	The content of AC0(31–16) is stored at the location addressed by AR3.

-								
No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[2]	Smem = <b>HI</b> (rr	d(ACx))			No	3	1	Х
Opcod	e			1110	1000 AAA	AA AA	AI SSx	x x0x%
Operar	nds	ACx, Smem						
Descrij	ption	This instruction stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand						unit shifter
Status	Bits	Affected by	RDM					
		Affects	none					
Repeat	:	This instruction can be repeated.						
Examp	le							
Syntax	(	Description						
*AR3 =	HI(rnd(AC0))	The content of A	C0(31–16)	is rounded and	stored at the	locatior	addresse	ed by AR3.

## Syntax Characteristics

-				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[3]	Smem = <b>LO(</b> A	Cx << Tx)		No	3	1	Х
Opcod	е		1110	0111 AAA	AA AA	AISS	s 00xx
Operar	nds	ACx, Smem, T	x				
Descri	ption	This instruction shifts the accumulator, ACx, by the content of Tx and stores the low part of the accumulator, ACx(15–0), to the memory (Smem) location. If the 16-bit value in Tx is not within $-32$ to $+31$ , the shift is saturated to $-32$ or +31 and the shift is performed with this value. The input operand is shifted in the D-unit shifter according to SXMD. <b>Compatibility with C54x devices (C54CM = 1)</b> When this instruction is executed with C54CM = 1, the 6 LSBs of Tx are used to determine the shift quantity. The 6 LSBs of Tx define a shift quantity within -32 to $+31$ . When the 16-bit value in Tx is between $-32$ to $-17$ , a modulo 16 operation transforms the shift quantity to within $-16$ to $-1$ .					n) location. d to –32 or s shifted in x are used ntity within
Status	Bits	Affected by	C54CM, SXMD				
		Affects	none				
Repeat	t	This instruction	can be repeated.				

Syntax	Description
*AR3 = LO(AC0 << T0)	The content of AC0 is shifted by the content of T0 and AC0(15–0) is stored at the location addressed by AR3.

Syntax	Characteris	0.005					
No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[4]	Smem = HI(I	rnd <b>(</b> ACx << Tx <b>)</b> )		No	3	1	Х
Opcod	e		1110	0111 AAA	AA AA	AISS	s 10x%
Operands ACx, Smem, Tx							
<b>Description</b> This instruction shifts the accumulator, ACx, by the contr high part of the accumulator, ACx(31–16), to the memory the 16-bit value in Tx is not within –32 to +31, the shift is +31 and the shift is performed with this value. The input the D-unit shifter according to SXMD. Rounding is per shifter according to RDM, if the optional rnd keyword is operand.					emory shift is input c is perfe	(Smem) saturated operand is ormed in	location. If d to –32 or s shifted in the D-unit
		Compatibility w	vith C54x devices (C	54CM = 1)			
		When this instruction is executed with C54CM = 1, the 6 LSBs of Tx are to determine the shift quantity. The 6 LSBs of Tx define a shift quantity w $-32$ to $+31$ . When the 16-bit value in Tx is between $-32$ to $-17$ , a modul operation transforms the shift quantity to within $-16$ to $-1$ .					
Status	Bits	Affected by	C54CM, RDM, SXMD	)			
		Affects	none				
Repea	t	This instruction can be repeated.					
Examp	ole						
Syntax	K	Description					

Syntax	Description
	The content of AC0 is shifted by the content of T0, is rounded, and
	AC0(31–16) is stored at the location addressed by AR3.

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[5]	Smem = LO(A	.Cx << #SHIFTW <b>)</b>			No	3	1	Х
Opcode	e			1110	1001 AAA	AA AA	AI SSS	H IFTW
Operan	nds	ACx, SHIFTW	/, Smem					
Descrip	otion	This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores the low part of the accumulator, ACx(15–0), to the memory (Smem) location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.					ry (Smem)	
Status	Bits	Affected by	SXMD					
		Affects	none					
Repeat	:	This instructio	on can be repeate	ed.				
Examp	le							

Syntax	Description
*AR3 = LO(AC0 << #31)	The content of AC0 is shifted left by 31 bits and AC0(15–0) is stored at the
	location addressed by AR3.

### **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[6]	Smem = <b>HI(</b> AC	x << #SHIFTW <b>)</b>			No	3	1	Х
Opcode	e			L110	1010 AAA	A AA	AI SSS	H IFTW
Operan	nds	ACx, SHIFTW,	Smem					
Descrip	otion	This instruction shifts the accumulator stores the high part of the accumulato location. The input operand is shifted according to SXMD.			r, ACx(31–16	6), to th	ne memoi	ry (Smem)
Status	Bits	Affected by	SXMD					
		Affects	none					
Repeat		This instruction	n can be repeated	l.				

Syntax	Description
*AR3 = HI(AC0 << #31)	The content of AC0 is shifted left by 31 bits and AC0(31–16) is stored at the
	location addressed by AR3.

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[7] Smem = <b>HI</b> (rnd <b>(</b> ACx << <b>#</b> SHIFTW <b>)</b> )					No	4	1	Х		
Opcode	e		1111	1010 AAAA	AAAI XXS	SH IF	TW SSx	x x0x%		
Operands		ACx, SHIFTW	ACx, SHIFTW, Smem							
Description		This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, an stores the high part of the accumulator, ACx(31–16), to the memory (Smen location. The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD. Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.						ry (Smem) unit shifter		
Status	Bits	Affected by RDM, SXMD								
		Affects	none							
Repeat		This instruction cannot be repeated when using the *(#k23) absolute address- ing mode to access the memory operand (Smem); when using other address- ing modes, this instruction can be repeated.								

Syntax	Description
*AR3 = HI(rnd(AC0 << #31))	The content of AC0 is shifted left by 31 bits, is rounded, and AC0(31–16) is stored at the location addressed by AR3.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[8]	Smem = <b>HI(</b> sa	iturate	(uns(rnd <b>(</b>	ACx <b>)))</b>		No	3	1	Х	
Opcod	e				1110	1000 AAA	AA AA	AI SSx	x xlu%	
Operar	nds	AC	x, Smen	า						
Descri	ption	This instruction stores the high part of the accumulator, ACx(31–16 memory (Smem) location.						16), to th		
			Input o	perands are c	onsidered sig	gned or unsig	gned a	ccording	to uns.	
				ne optional uns he memory lo	•		•	perand, t	he conte	
				f the optional uns keyword is not applied to the input operand, the content of the memory location is considered signed.						
			If the optional rnd keyword is applied to the input operand, rou performed in the D-unit shifter according to RDM.						ounding	
			☐ When a rounding overflow is detected and if the optional saturate is applied to the input operand, the 40-bit output of the ope saturated:						•	
				ne optional un ue is 00 FFFF	•	applied to the	e input	operand,	saturatio	
			00	he optional u 7FFF FFFFh erflow).	•					
		Co	ompatibi	lity with C54	x devices (C	54CM = 1)				
		out	When this instruction is executed with $C54CM = 1$ , overflow do output of the shifter consists of checking if the sign of the inpidentical to the most-significant bits of the 40-bit result of the row					he input o	operand	
		If the optional uns keyword is applied to the input operand, then bits 39 of the result are compared to 0.						bits 39–3		
				ptional uns ke of the result ar	•		-	•		
Status	Bits	Aff	ected by	C54CM,	RDM, SXMD	)				
		Aff	ects	none						

# **Repeat** This instruction can be repeated.

Syntax	Description
*AR3 = HI(saturate(uns(rnd(AC0))))	The unsigned content of AC0 is rounded, is saturated, and
	AC0(31–16) is stored at the location addressed by AR3.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[9]	Smem = <b>HI(</b>	saturate	(uns	<mark>(rnd(</mark> ACx << Tx <b>)</b> ))		No	3	1	Х
Opcod	e				1110	0111 AA	AA AA	AI SSs	s 11u%
Operar	nds	AC	x, S	Smem, Tx					
Descri	otion	the If th	hig ne 1	struction shifts the acc h part of the accumulat 6-bit value in Tx is not id the shift is performe	or, ACx(3 within –3	1–16), to the 2 to +31, the	ememo	ory (Smen	n) locatior
			Inp	out operands are cons	dered sig	gned or unsi	gned a	ccording	to uns.
				If the optional uns key of the memory locati			•	perand, t	he conter
				If the optional uns ke content of the memo	•			• •	erand, th
		The input operand is shifted in the D-unit shifter according to SXM							SXMD.
			<ul> <li>When shifting, the sign position of the input operand is compared shift quantity.</li> </ul>						
				If the optional uns comparison is perfor	•			• •	
				If the optional uns performed against b signed (the sign is de	it 31 of t	he shifted c	peranc	I that is o	considere
				An overflow is gener	ated acco	ordingly.			
				he optional rnd keywo rformed in the D-unit s			•	perand, r	ounding i
			ke	nen a shift or rounding yword is applied to the saturated:				•	
				If the optional uns ke value is 00 FFFF FF		applied to th	e input	operand,	saturatio
				If the optional uns 00 7FFF FFFFh (po overflow).	•				

	Compatibility with C54x devices (C54CM = 1)	
	When this instruction is executed with C54CM = 1:	
	Overflow detection at the output of the shifter consists of checkin sign of the input operand is identical to the most-significant bits 40-bit result of the shift and round operation.	-
	If the optional uns keyword is applied to the input operand, the 39–32 of the result are compared to 0.	ien bits
	If the optional uns keyword is not applied to the input op then bits 39–31 of the result are compared to bit 39 of the operand and SXMD.	
	❑ The 6 LSBs of Tx are used to determine the shift quantity. The 6 L Tx define a shift quantity within -32 to +31. When the 16-bit valu is between -32 to -17, a modulo 16 operation transforms the shift of to within -16 to -1.	ie in Tx
Status Bits	Affected by C54CM, RDM, SXMD	
	Affects none	
Repeat	This instruction can be repeated.	

Syntax	Description
	The unsigned content of AC0 is shifted by the content of T0, is rounded, is saturated, and AC0(31–16) is stored at the location addressed by AR3.

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline		
[10]	Smem = I	HI(saturate(	uns	(rnd(ACx << #SHII	-TW <b>))))</b>	No	4	1	Х		
Opcod	е			1111	1010 AAAA	AAAI uxSH	IFI	W SSxx	x x1x%		
Operands		AC	x, 8	SHIFTW, Smem		·		·			
Description		sto	This instruction shifts the accumulator, ACx, by the 6-bit value, SH stores the high part of the accumulator, ACx(31–16), to the memorilocation.								
			In	out operands are	considered sig	ned or unsign	ed ac	cording to	o uns.		
				•	tional uns keyword is applied to the input operand, the content emory location is considered unsigned.						
				If the optional content of the	uns keyword is memory locatio			• •	rand, th		
			The input operand is shifted by the 6-bit value in the according to SXMD.						nit shifte		
				hen shifting, the ift quantity.	sign position of	f the input ope	rand i	s compai	red to th		
				If the optional comparison is	uns keyword performed agai			• •			
					I uns keyword linst bit 31 of tl n is defined by l	he shifted ope	erand	that is co	onsidere		
				An overflow is	generated acco	ordingly.					
				the optional rnd erformed in the D	• • • •			erand, ro	unding i		
			ke	hen a shift or rou yword is applied saturated:	-			•			
				If the optional uvalue is 00 FFI	ins keyword is a FF FFFFh.	applied to the i	nput o	perand, s	aturatio		
				If the optional 00 7FFF FFFF overflow).	uns keyword h (positive ove						

Compatibility with	h C54x devices	(C54CM = 1)
--------------------	----------------	-------------

When this instruction is executed with $C54CM = 1$ , overflow detection at the output of the shifter consists of checking if the sign of the input operand is identical to the most-significant bits of the 40-bit result of the shift and round operation.
☐ If the optional uns keyword is applied to the input operand, then bits 39–32 of the result are compared to 0.

☐ If the optional uns keyword is not applied to the input operand, then bits 39–31 of the result are compared to bit 39 of the input operand and SXMD.

 Status Bits
 Affected by
 C54CM, RDM, SXMD

 Affects
 none

RepeatThis instruction cannot be repeated when using the \*(#k23) absolute address-<br/>ing mode to access the memory operand (Smem); when using other address-<br/>ing modes, this instruction can be repeated.

Syntax	Description
*AR3 = HI(saturate(uns(rnd(AC0 << #31))))	The unsigned content of AC0 is shifted left by 31 bits, is rounded, is saturated, and AC0(31–16) is stored at the location addressed by AR3.

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[11]	dbl(Lmem) =	ACx			No	3	1	Х
Opcode	•			1110	1011 AAA	A AAZ	AI XXS	S 10x0
Operands		ACx, Lmem						
Description		This instruction stores the content of the accumulator, ACx(31–0), to the data memory operand (Lmem). The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.						
Status E	Bits	Affected by	none					
		Affects	none					
Repeat		This instructio	n can be repea	ated.				

Syntax	Description
dbl(*AR3) = AC0	The content of AC0 is stored at the locations addressed by AR3 and AR3 + 1.

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline				
[12]	dbl(Lmem) = s	saturate(uns(ACx))	No	3	1	Х				
Opcod	e	1110	1011 AAA	AA AZ	AI XXS	S 10u1				
Operar	nds	ACx, Lmem								
Description		This instruction stores the content of the accumulator, $ACx(31-0)$ , to the data memory operand (Lmem).								
		Input operands are considered signed or unsigned according to uns.								
		If the optional uns keyword is of the memory location is con	••	•	operand, t	he content				
		If the optional uns keyword is not applied to the input op content of the memory location is considered signed.								
		The 40-bit output of the operation is saturated:								
		If the optional uns keyword is value is 00 FFFF FFFFh.	applied to the	e input	operand,	saturation				
		If the optional uns keyword 00 7FFF FFFFh (positive ov overflow).								
		The store operation to the memory	ry location us	es the	D-unit sł	nifter.				
		Compatibility with C54x devices (0	C54CM = 1)							
		When this instruction is executed wit output of the shifter consists of chec identical to the most-significant bits o operation.	king if the sig	gn of t	he input	operand is				
		If the optional uns keyword is appl of the result are compared to 0.	ied to the inpu	it oper	and, then	bits 39–32				
		If the optional uns keyword is no 39–31 of the result are compared			•					
Status	Bits	Affected by C54CM, SXMD								
		Affects none								

# **Repeat** This instruction can be repeated.

Syntax	Description
dbl(*AR3) = saturate(uns(AC0))	The unsigned content of AC0 is saturated and stored at the locations
	addressed by AR3 and AR3 + 1.

## Syntax Characteristics

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[13]	HI(Lmem) = HI LO(Lmem) = L					No	3	1	Х	
Opcod	e			1	110	1011 AAA	AA AA	AI XXS	S 1101	
Operar	nds	ACx, Lr	nem							
Description		This instruction performs two store operations in parallel and is executed in the D-unit shifter:								
		The 16 highest bits of the accumulator, ACx(31–16), shifted right by 1 bit (bit 31 is sign extended according to SXMD), are stored to the 16 highest bits of the data memory operand (Lmem).								
		acc	ording	est bits, ACx(15–0 to SXMD), are stor .mem).		• •	•	-		
Status	Bits	Affected	d by	SXMD						
		Affects		none						
Repeat	t	This instruction can b		n can be repeated.						

Syntax	Description
LO(*AR1) = LO(AC0) >> #1	The content of AC0(31–16), shifted right by 1 bit, is stored at the location addressed by AR1 and the content of AC0(15–0), shifted right by 1 bit, is stored at the location addressed by AR1 + 1.

### **Syntax Characteristics**

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline
NO.	Syntax								Cycles	•
[14]	Xmem = <b>LO(</b> AC Ymem = <b>HI(</b> AC)	•					No	3	1	Х
Opcode	9				10	000	0000 XXX	KM MM	YMMY YMM	M 10SS
Operan	lds	ACx, >	Kmem, Y	mem						
Description		This instruction performs two store operations in parallel:								
		_		west bits perand X		cum	ulator, ACx(	15–0),	are store	ed to data
		🗋 Th	ne 16 higł	nest bits, A	ACx(31–16	6), ar	e stored to da	ta men	nory opera	and Ymem.
Status Bits		Affecte	ed by	none						
		Affects	6	none						
Repeat		This in	structior	n can be	repeated.					

Syntax	Description
*AR1 = LO(AC0), *AR2 = HI(AC0)	The content of AC0(15–0) is stored at the location addressed by AR1 and the content of AC0(31–16) is stored at the location addressed by AR2.
////2 = /////00/	

Before			After				
AC0	01 4500	0030	AC0	01	4500	0030	
AR1		0200	AR1			0200	
AR2		0201	AR2			0201	
200		3400	200			0030	
201		0FD3	201			4500	

# Store Accumulator Pair Content to Memory

#### Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	Lmem = <b>pair(Hi</b>	(ACx)	)		No	3	1	Х			
[2]	Lmem = <b>pair(LC</b>	<b>)(</b> AC×	.))		No	3	1	Х			
Descr	ription		This instruction stores the content of the selected accumulator pair, ACx and $AC(x + 1)$ , to a data memory operand (Lmem).								
Statu	s Bits	Aff	ected by	none							
		Aff	ects	ects none							
See A	lso	Se	See the following other related instructions:								
			Addition with Parallel Store Accumulator Content to Memory								
				Load Accumulator from Memory with Parallel Store Accumulator Content to Memory							
			Load Accumulator, Auxiliary, or Temporary Register from Memory								
			Multiply an	d Accumulate with Par	rallel Store Accur	nulator	Content	to Memor			
			Multiply ar	nd Subtract with Paral	lel Store Accum	ulator	Content t	o Memor			
			Multiply wi	th Parallel Store Acc	umulator Conte	nt to M	lemory				
			Store Accu	umulator Content to N	Memory						
			Store Accu	umulator, Auxiliary, oi	r Temporary Re	gister (	Content t	o Memor			
			Store Auxiliary or Temporary Register Pair Content to Memory								
			Subtraction with Parallel Store Accumulator Content to Memory					ry			

# Store Accumulator Pair Content to Memory

#### **Syntax Characteristics**

				Parallel					
No.	Syntax			Enable Bit	Size	Cycles	Pipeline		
[1]	Lmem = <b>pair(</b>	HI(ACx))		No	3	1	Х		
Opcod	e		1110	1011 AAA	AA AA	AI XXS	S 1110		
Operar	nds	ACx, Lmem							
Descrij	ption	This instruction stores the 16 highest bits of the accumulator, $ACx(31-16)$ , to the 16 highest bits of the data memory operand (Lmem) and stores the 16 highest bits of $AC(x + 1)$ to the16 lowest bits of data memory operand (Lmem):							
			The store operation to the memory location uses a dedicated path independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.						
		Valid accumul	ators are AC0 and A	AC2.					
Status	Bits	Affected by no	Affected by none						
		Affects no	one						
Repeat	t	This instruction ca	in be repeated.						

Syntax			Descr	Description							
*AR1+ = pair(HI(AC0))			conter	The content of AC0(31–16) is stored at the location addressed by AR1 and the content of AC1(31–16) is stored at the location addressed by AR1 + 1. AR1 is incremented by 2.							
Before				After							
AC0	01	4500	0030	AC0	01	4500	0030				
AC1	03	5644	F800	AC1	03	5644	F800				
AR1			0200	AR1			0202				
200			3400	200			4500				
201			0FD3	201			5644				

# Store Accumulator Pair Content to Memory

#### **Syntax Characteristics**

				Parallel					
No.	Syntax			Enable Bit	Size	Cycles	Pipeline		
[2]	Lmem = <b>pair(I</b>	L <b>O(</b> ACx <b>))</b>	No	3	1	Х			
Opcod	е		1110	1011   AAA	AA AA	AI XXS	S 1111		
Operar	nds	ACx, Lmem							
Descrij	ption	This instruction stores the 16 lowest bits of the accumulator, $ACx(15-0)$ , to the 16 highest bits of the data memory operand (Lmem) and stores the 16 lowest bits of $AC(x + 1)$ to the16 lowest bits of data memory operand (Lmem):							
		independe	independent of the D-unit ALU, the D-unit shifter, and the D-unit MACs.						
		Valid accur	mulators are AC0 and a	AC2.					
Status	Bits	Affected by	none						
		Affects	none						
Repeat	t								

Syntax	Description
*AR3 = pair(LO(AC0))	The content of AC0(15–0) is stored at the location addressed by AR3 and the content of AC1(15–0) is stored at the location addressed by AR3 + 1.

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	Smem = src				No	2	1	х		
[2]	high_byte(Sm	em <b>)</b> =	= SIC		No	3	1	Х		
[3]	low_byte(Sme	m <b>)</b> =	src		No	3	1	Х		
Descri	ption		is instructior mory (Smer	n stores the content of m) location.	f the selected	source	e (src) re	gister to a		
Status	Bits	Aff	ected by	none						
		Aff	ects	none						
See Al	so	See the following other related instructions:								
		Addition with Parallel Store Accumulator Content to Memory								
			Load Accumulator from Memory with Parallel Store Accumulator Content to Memory							
			Load Accumulator, Auxiliary, or Temporary Register from Memory							
		Multiply and Accumulate with Parallel Store Accumulator Content to Memory								
		Multiply and Subtract with Parallel Store Accumulator Content to Memory								
		Multiply with Parallel Store Accumulator Content to Memory								
			Store Accu	imulator Content to Me	emory					
			Store Accumulator Pair Content to Memory							
			Store Auxi	liary or Temporary Reg	gister Pair Co	ntent to	Memory	/		
			Subtraction	n with Parallel Store A	ccumulator Co	ontent	to Memo	ry		

#### **Syntax Characteristics**

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	Smem = src					No	2	1	X	
Opcod	e					110	)0 FS	SS AAA	A AAAI	
Operar	nds	Smerr	n, src							
Description			nstruction m) location		e content of	the source (	src) re	gister to	a memory	
		When the source register is an accumulator:								
		-	■ The low part of the accumulator, ACx(15–0), is stored to the memory location.							
		-		•		emory locati , the D-unit sh			•	
		When the source register is an auxiliary or temporary register:								
		•		ontent of t	•	or temporary	/ regis	ter is sto	red to the	
		-		•	tion to the m	emory locati U.	on use	s a dedio	cated path	
Status	Bits	Affect	ed by	none						
		Affect	S	none						
Repeat	t	This instruction can be repeated.								

Syntax	Description
*(#0E10h) = AC0	The content of AC0(15–0) is stored at location E10h.

Before			After			
AC0	23 0400 6	6500	AC0	23	0400	6500
0E10	(	0000	0E10			6500

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[2]	high_byte(Si	mem) = src	No	3	1	Х			
Opcod	e	1110	0101 AAA	AA AA	AI FSS	S 01x0			
Operar	nds	Smem, src							
Descri	ption	This instruction stores the low byte (bits 7–0) of the source (src) register to the high byte (bits 15–8) of the memory (Smem) location. The low byte (bits 7–0) of Smem is unchanged.							
		When the source register is an ac	cumulator:						
		The low part of the accumulate the memory location.	or, ACx(7–0),	is store	ed to the h	igh byte of			
		The store operation to the m independent of the D-unit ALU	-			-			
		When the source register is an au	ixiliary or tem	nporary	/ register	:			
		The low part (bits 7–0) content stored to the high byte of the		•	emporary	register is			
		The store operation to the m independent of the A-unit ALL	•	on use	es a dedio	cated path			
		(MMR). This instruction cannot ac	(MMR). This instruction cannot access a byte within an MMR. If an MMR, the DSP sends a hardware bus-error interrupt (B						
Status	Bits	Affected by none							
		Affects none							
Repeat	t	This instruction can be repeated.							

#### Svntax Characteristics

Syntax		Desc	Description						
high_byte	(*AR1) = AC1		The content of AC1(7–0) is stored in the high byte (bits 15–8) at the location addressed by AR1.						
Before			After						
AC1	20 FC00	6788	AC1	20 FC00	6788				
AR1		0200	AR1		0200				
200		6903	200		8803				

# Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[3]	low_byte(Sme	em) = src			No	3	1	Х
Opcod	е			1110	0101 AAA	A AA	AI FSS	S 01x1
Operar	nds	Smerr	, src					
Descri	ption	low by	struction stores the low te (bits 7–0) of the mem em is unchanged.	•	•			
		D W	hen the source register	is an ac	cumulator:			
		-	The low part of the acc the memory location.	cumulato	or, ACx(7–0),	is store	ed to the	ow byte of
		-	The store operation to independent of the D-u		•			•
		D W	hen the source register	is an au	xiliary or tem	nporary	register:	
		-	The low part (bits 7–0) stored to the low byte			•	emporary	register is
		-	The store operation to independent of the A-		•	on use	s a dedio	cated path
		In this instruction, Smem cannot reference to a memory-mapped registe (MMR). This instruction cannot access a byte within an MMR. If Smem is an MMR, the DSP sends a hardware bus-error interrupt (BERRINT request to the CPU.						f Smem is
Status	Bits	Affecte	ed by none					
		Affects	s none					
Repeat	t	This ir	struction can be repeat	ed.				

Syntax	Description
low_byte(*AR3) = AC0	The content of AC0(7–0) is stored in the low byte (bits 7–0) at the location addressed by AR3.

Store Auxiliary or Temporary Register Pair Content to Memory

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Lmem = pair(	TAx)		No	3	1	X
Opcod	e		1110	1011 AAA	AA AA	AI FSS	SS 1100
Operar	nds	TAx, Lmem					
Description		This instruction stores the content of the temporary or auxiliary register (TAx) to the 16 highest bits of the data memory operand (Lmem) and stores the content of $TA(x + 1)$ to the 16 lowest bits of data memory operand (Lmem):					
			operation to the men ent of the A-unit ALU.	nory location	n uses	a dedio	cated path
		Valid auxil	iary registers are AR0,	AR2, AR4, a	nd AR	6.	
		Valid temp	oorary registers are T0 a	and T2.			
Status	Bits	Affected by	none				
		Affects	none				
Repeat	:	This instruction	n can be repeated.				
See Al	50	See the follow	ing other related instruc	tions:			
		🗋 Load Accu	imulator, Auxiliary, or Te	emporary Re	gister	from Mer	nory
		Store Accu	umulator, Auxiliary, or Te	emporary Re	gister	Content	to Memory

Syntax	Description
*AR2 = pair(T0)	The content of T0 is stored at the location addressed by AR2 and the content of T1 is stored at the location addressed by AR2 + 1.

# Store CPU Register Content to Memory

# Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	Syntax Smem = <b>BK03</b>	No	3	1	X
[2]	Smem = <b>BK47</b>	No	3	1	Х
[3]	Smem = <b>BKC</b>	No	3	1	Х
[4]	Smem = <b>BSA01</b>	No	3	1	Х
[5]	Smem = <b>BSA23</b>	No	3	1	Х
[6]	Smem = <b>BSA45</b>	No	3	1	Х
[7]	Smem = <b>BSA67</b>	No	3	1	Х
[8]	Smem = <b>BSAC</b>	No	3	1	х
[9]	Smem = BRC0	No	3	1	х
[10]	Smem = BRC1	No	3	1	х
[11]	Smem = CDP	No	3	1	Х
[12]	Smem = CSR	No	3	1	Х
[13]	Smem = <b>DP</b>	No	3	1	Х
[14]	Smem = DPH	No	3	1	Х
[15]	Smem = PDP	No	3	1	х
[16]	Smem = <b>SP</b>	No	3	1	х
[17]	Smem = SSP	No	3	1	х
[18]	Smem = TRN0	No	3	1	х
[19]	Smem = TRN1	No	3	1	х
[20]	dbl(Lmem) = RETA	No	3	5	х

Opcode

See Table 5–6 (page 5-461).

Operands Lmem, Smem

5-458 Instruction Set Descriptions

Description	These instructions store the content of the selected source CPU register to a memory (Smem) location or a data memory operand (Lmem).		
	For instructions [9] and [10], the block repeat register (BRCx) is decremented in the address phase of the last instruction of the loop. These instructions have a 3-cycle latency requirement versus the last instruction of the loop.		
	For instruction [20], the content of the 24-bit RETA register (the return address of the calling subroutine) and the 8-bit CFCT register (active control flow execution context flags of the calling subroutine) are stored to the data nemory operand (Lmem):		
	The content of the CFCT register and the 8 highest bits of the RETA register are stored in the 16 highest bits of Lmem.		
	The 16 lowest bits of the RETA register are stored in the 16 lowest bits of Lmem.		
	When instruction [20] is decoded, the CPU pipeline is flushed and the instruction is executed in 5 cycles, regardless of the instruction context.		
Status Bits	Affected by none		
	Affects none		
Repeat	Instruction [20] cannot be repeated; all other instructions can be repeated.		
See Also	See the following other related instructions:		
	Load CPU Register from Memory		
	Load CPU Register with Immediate Value		
	Move CPU Register Content to Auxiliary or Temporary Register		
	Store Accumulator Content to Memory		
	Store Accumulator Pair Content to Memory		
	Store Accumulator, Auxiliary, or Temporary Register Content to Memory		
	Store Auxiliary or Temporary Register Pair Content to Memory		

Syntax	Description				
*AR1+ = SP The content of the data stack pointer (SP) is stored in the location addressed AR1 is incremented by 1.					
Before		After			
AR1	0200	AR1	0201		
SP	0200	SP	0200		
200	0000	200	0200		

# Example 2

Syntax	Descripti	Description				
*AR1+ = SSP		The content of the system stack pointer (SSP) is stored in the location addressed by AR1. AR1 is incremented by 1.				
Before		After				
AR1	0201	AR1	0202			
SSP	0000	SSP	0000			
201	OOFF	201	0000			

# Example 3

Syntax	Description
*AR1+ = TRN0	The content of the transition register (TRN0) is stored in the location addressed by AR1. AR1 is incremented by 1.

Before		After	
AR1	0202	AR1	0203
TRN0	3490	TRN0	3490
202	0000	202	3490

# Example 4

Syntax	Descripti	on		
*AR1+ = TRN1		The content of the transition register (TRN1) is stored in the location addressed by AR1. AR1 is incremented by 1.		
Before		After		
AR1	0203	AR1	0204	

AR1	0203	AR1	0204
TRN1	0020	TRN1	0020
203	0000	203	0020

Syntax	Description
dbl(*AR3) = RETA	The contents of the RETA and CFCT are stored in the location addressed by AR3 and AR3 + 1.

No.	Syntax	Opcode	_
[1]	Smem = <b>BK03</b>	1110 0101 AAAA AAAI 1001 10xx	
[2]	Smem = <b>BK47</b>	1110 0101 AAAA AAAI 1010 10xx	
[3]	Smem = <b>BKC</b>	1110 0101 AAAA AAAI 1011 10xx	
[4]	Smem = <b>BSA01</b>	1110 0101 AAAA AAAI 0010 10xx	
[5]	Smem = <b>BSA23</b>	1110 0101 AAAA AAAI 0011 10xx	
[6]	Smem = <b>BSA45</b>	1110 0101 AAAA AAAI 0100 10xx	
[7]	Smem = <b>BSA67</b>	1110 0101 AAAA AAAI 0101 10xx	
[8]	Smem = <b>BSAC</b>	1110 0101 AAAA AAAI 0110 10xx	
[9]	Smem = BRC0	1110 0101 AAAA AAAI x001 11xx	
[10]	Smem = BRC1	1110 0101 AAAA AAAI x010 11xx	
[11]	Smem = CDP	1110 0101 AAAA AAAI 0001 10xx	
[12]	Smem = <b>CSR</b>	1110 0101 AAAA AAAI x000 11xx	
[13]	Smem = <b>DP</b>	1110 0101 AAAA AAAI 0000 10xx	
[14]	Smem = <b>DPH</b>	1110 0101 AAAA AAAI 1100 10xx	
[15]	Smem = <b>PDP</b>	1110 0101 AAAA AAAI 1111 10xx	
[16]	Smem = <b>SP</b>	1110 0101 AAAA AAAI 0111 10xx	
[17]	Smem = SSP	1110 0101 AAAA AAAI 1000 10xx	
[18]	Smem = TRN0	1110 0101 AAAA AAAI x011 11xx	
[19]	Smem = TRN1	1110 0101 AAAA AAAI x100 11xx	
[20]	dbl(Lmem) = RETA	1110 1011 AAAA AAAI xxxx 01xx	

Table 5–6. Opcodes for Store CPU Register Content to Memory Instruction

# Store Extended Auxiliary Register Content to Memory

#### **Syntax Characteristics**

				Parallel			
No.	Syntax			Enable Bit	Size	Cycles	Pipeline
[1]	dbl(Lmem) =	XAsrc		No	3	1	Х
Opcod	e		1110	1101 AA	AA AA	AAI XSS	S 0101
Opera	nds	Lmem, XAsrc					
Description		XSSP, XDP, or	n moves the content of <sup>r</sup> XCDP) to the 32-bit da nd (Lmem). The upper 9	ata memory l	ocatior	n address	ed by data
Status	Bits	Affected by	none				
		Affects	none				
Repea	t	This instruction	n can be repeated.				
See Al	so	See the follow	ing other related instrue	ctions:			
		Load Exte	nded Auxiliary Register	r from Memo	ory		
		Load Exte	nded Auxiliary Register	r with Immec	liate Va	alue	
		Modify Ext	tended Auxiliary Regist	er Content			
		Move External	ended Auxiliary Registe	r Content			
Evom							

Syntax	Des	ription					
dbl(*AR3) = XAR1	by A	The 7 highest bits of XAR1 are moved to the 7 lowest bits of the location addressed by AR3, the 9 highest bits are filled with 0, and the 16 lowest bits of XAR1 are moved to the location addressed by AR3 + 1.					
Before		After					
XAR1	7F 3492	XAR1	7F 3492				

AARI	1 5	3492	AARI	1 Г	3492
AR3		0200	AR3		0200
200		3765	200		007F
201		0FD3	201		3492

Subtract Conditionally

**Syntax Characteristics** 

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[1]	subc(Smem	i, ACx, ACy <b>)</b>	No	3	1	х
Opcod	e		1101 1110 AA	AAA A	AAI SSI	DD 0011
Opera	nds	ACx, ACy, Smem				
Descri	ption	This instruction performs a cor D-unit shifter is not used to perf				ALU. The
		The 16-bit data memory of according to SXMD, shifted of the source accumulator A	left by 15 bits, and	-		
		<ul> <li>The shift operation is ed</li> <li>Overflow and carry bit subtraction borrow bit is bit is the logical comple</li> </ul>	is always detection reported in the C	ted at ARRY	bit positic status bit;	on 31. The
		<ul> <li>If an overflow is detect ACOVy, no saturation is</li> </ul>	•			
		If the result of the subtraction shifted left by 1 bit, added to ACy.	-	•		
		If the result of the subtract accumulator ACx is shifted accumulator ACy.		•	,	
		if ((ACx - (Smem << #	15)) >= 0)			
		ACy = (ACx - (Smem)	<< #15)) <<	#1 + 1	L	
		else ACy = ACx << #1				
		This instruction is used to make and the dividend are both assu affects this operation:				
		If SXMD = 1, the divisor mu	st have a 0 value	in the	most sign	ificant bit
		☐ If SXMD = 0, any 16-bit divi	sor value produc	es the e	expected	result
		The dividend, which is in the (bit $21 - 0$ ) during the computed		ator AC	x, must t	pe positive

(bit 31 = 0) during the computation.

Status Bits	Affected by	SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	can be repeated.
See Also	See the following	ng other related instructions:
	Addition or	Subtraction Conditionally
	Addition or	Subtraction Conditionally with Shift
	Addition, S	ubtraction, or Move Accumulator Content Conditionally
	Dual 16-Bit	Subtraction and Addition
	Subtraction	1
	Subtraction	with Parallel Store Accumulator Content to Memory

# Example 1

Syntax			De	Description						
subc(*AR1, AC0, AC1)			cor 1 b	The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC0. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The result generated an overflow and a carry.						
Before				After						
AC0	23	4300	0000	AC0	23	4300	0000			
AC1	00	0000	0000	AC1	46	8400	0001			
AR1			300	AR1			300			
300			200	300			200			
SXMD			0	SXMD			0			
ACOV1			0	ACOV1			1			
CARRY			0	CARRY			1			

# Example 2

Syntax		Description						
repeat (CS subc(*AR1	SR) I, AC1, AC1)	conter 1 bit, a AR1 sl greate	The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The content addressed by AR1 shifted left by 15 bits is subtracted from the content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The content of AC1. The result is greater than 0; therefore, the result is shifted left by 1 bit, added to 1, and the new result stored in AC1. The result active a carry.					
Before			After					
AC1	00 0746 0	0000	AC1	00 1A18 0007				
AR1		200	AR1	200				
200	(	0100	200	0100				
CSR		1	CSR	0				
ACOV1		0	ACOV1	0				
CARRY		0	CARRY	1				

5-464 Instruction Set Descriptions

#### **Syntax Characteristics**

		Parallel			
No.	Syntax	Enable Bit	Size	Cycles	Pipeline
[1]	dst = dst - src	Yes	2	1	Х
[2]	dst = dst - k4	Yes	2	1	Х
[3]	dst = src - K16	No	4	1	Х
[4]	dst = src - Smem	No	3	1	Х
[5]	dst = Smem – src	No	3	1	Х
[6]	ACy = ACy – <b>(</b> ACx << Tx <b>)</b>	Yes	2	1	Х
[7]	ACy = ACy – (ACx << #SHIFTW)	Yes	3	1	Х
[8]	ACy = ACx – <b>(</b> K16 <b>&lt;&lt; #16)</b>	No	4	1	Х
[9]	ACy = ACx – <b>(</b> K16 <b>&lt;&lt; #</b> SHFT <b>)</b>	No	4	1	х
[10]	ACy = ACx - <b>(</b> Smem << Tx <b>)</b>	No	3	1	х
[11]	ACy = ACx - <b>(</b> Smem <b>&lt;&lt; #16)</b>	No	3	1	Х
[12]	ACy = <b>(</b> Smem <b>&lt;&lt; #16)</b> – ACx	No	3	1	х
[13]	ACy = ACx - uns(Smem) - BORROW	No	3	1	Х
[14]	ACy = ACx - uns(Smem)	No	3	1	х
[15]	ACy = ACx - (uns(Smem) << #SHIFTW)	No	4	1	Х
[16]	ACy = ACx - <b>dbl(</b> Lmem <b>)</b>	No	3	1	Х
[17]	ACy = <b>dbl(</b> Lmem <b>)</b> – ACx	No	3	1	Х
[18]	ACx = (Xmem << #16) – (Ymem << #16)	No	3	1	х

# Description

These instructions perform a subtraction operation.

**Status Bits** 

Affected by CARRY, C54CM, M40, SATA, SATD, SXMD

#### ACOVx, ACOVy, CARRY Affects

See Also

See the following other related instructions:	
---	--

- □ Addition
- Addition or Subtraction Conditionally
- Addition or Subtraction Conditionally with Shift
- Addition, Subtraction, or Move Accumulator Content Conditionally
- Dual 16-Bit Addition and Subtraction
- Dual 16-Bit Subtractions
- Dual 16-Bit Subtraction and Addition
- Subtract Conditionally
- Subtraction with Parallel Store Accumulator Content to Memory

**Syntax Characteristics** 

No.	Syntax					Parallel nable Bit	Size	Cycles	Pipeline
[1]	dst = dst – src					Yes	2	1	Х
Opcode	<u> </u>						10 01	11E FSS	S FDDD
Operan		dst, src				100			
Descrip			uction	performs a su	htraction c	peration	betwee	en two rec	nisters
2000.1				estination ope				-	
				eration is perfo	. ,				
				perands are sig					SXMD.
		∎ If a ins	an au struct	xiliary or tempo ion, the 16 LSE ed according to	orary regis Ss of the au	ter is the	source	operand	(src) of the
		su	ubtrac	w detection a tion borrow bit e logical comp	is reported	d in the C	ARRY	status bit;	
			/hen a SATE	n overflow is de D.	etected, th	e accumi	ulator is	saturated	laccordin
		🗋 When	the d	estination ope	rand (dst)	is an aux	iliary o	r tempora	ry registe
		∎ Th	ne ope	eration is perfo	ormed on 1	6 bits in	the A-u	init ALU.	
				cumulator is tl s of the accum			. ,		
				w detection is		•			
				an overflow is ng to SATA.	detected,	the desti	nation	register is	s saturate
		Compatib	oility v	with C54x dev	vices (C54	CM = 1)			
		When this	instru	uction is execu	ted with M	40 = 0, c	ompati	bility is er	nsured.
Status	Bits	Affected by	У	M40, SATA, S	SATD, SXN	/ID			
		Affects		ACOVx, CAR	RY				
Repeat		This instru	uction	can be repeat	ed.				
Examp	le								
Syntax		Description							
AC0 = /	AC0 – AC1	The content of	f AC1	is subtracted from	m the conte	nt of AC0 a	and the r	esult is sto	red in AC0.

# Syntax Characteristics

No. Syntax	Parallel Enable Bit Size Cycles Pipeline
[2] dst = dst - k4	Yes 2 1 X
Opcode	0100 011E kkkk FDDD
Operands	dst, k4
Description	This instruction subtracts a 4-bit unsigned constant, k4, from a register.
	When the destination operand (dst) is an accumulator:
	■ The operation is performed on 40 bits in the D-unit ALU.
	Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
	When an overflow is detected, the accumulator is saturated according to SATD.
	When the destination operand (dst) is an auxiliary or temporary register:
	The operation is performed on 16 bits in the A-unit ALU.
	Overflow detection is done at bit position 15.
	When an overflow is detected, the destination register is saturated according to SATA.
	Compatibility with C54x devices (C54CM = 1)
	When this instruction is executed with M40 = 0, compatibility is ensured.
Status Bits	Affected by M40, SATA, SATD
	Affects ACOVx, CARRY
Repeat	This instruction can be repeated.
Fxample	

Syntax	Description
	An unsigned 4-bit value (15) is subtracted from the content of AC0 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax						Parallel Enable Bit	Size	Cycles	Pipeline		
[3]	dst = src - K16						No	4	1	Х		
Opcode				0111	1100 K	ккк	кккк кк	KK KI	KKK   FDI	DD FSSS		
Operands			t, K16, src									
Description		This instruction subtracts a 16-bit signed constant, K16, from a register.										
			When the	e destinat	ion operan	nd (de	st) is an acc	umulat	or:			
			The operation is performed on 40 bits in the D-unit ALU.									
			instru	iction, the	•	of the	auxiliary or	the source operand (src) of the ry or temporary register are sign				
			■ The 1 SXMI		nstant, K16	6, is	sign extend	led to	40 bits ad	ccording to		
			subtra	action bo	rrow bit is r	epor	RY status ted in the CA	ARRY s	status bit;			
			■ When to SA		low is dete	cted,	the accumu	llator is	saturated	laccording		
			When the	destinat	ion operan	ıd (ds	st) is an aux	iliary o	r tempora	ry register:		
			The c	peration	is perform	ed o	n 16 bits in t	the A-u	init ALU.			
							ce operand are used to p					
			Overf	low dete	ction is dor	ne at	bit position	15.				
				n an over ding to S		tecte	d, the destii	nation	register is	saturated		
		Co	ompatibility	y with C	54x device	es (C	54CM = 1)					
		Wh	nen this ins	truction is	s executed	l with	M40 = 0, c	ompati	bility is er	nsured.		

SPRU375G

Status Bits	Affected by	M40, SATA, SATD, SXMD				
	Affects	ACOVx, CARRY				
Repeat	This instruction can be repeated.					
Example						
Syntax	Description					
AC0 = AC1 – FFFFh	A signed 16-bit is stored in AC0	value (FFFFh) is subtracted from the content of AC1 and the result ).				

# **Syntax Characteristics**

No.	Syntax								Parallel Enable Bit	Size	Cycles	Pipeline
[4]	dst = src - Sm	em							No	3	1	Х
Opcod	e						110	)1	0111 AA	AA A	AAI   FDI	DD FSSS
Operar	nds	dst	, Sr	merr	n, src							
Description			This instruction subtracts the content of a memory (Smem) location from a register content.									
			W	hen	the des	tinatior	operand	(ds	st) is an acc	umula	tor:	
				Th	he opera	ition is	performed	lo	n 40 bits in <sup>-</sup>	the D-	unit ALU.	
				ins	struction	, the 10		he	gister is the auxiliary or		•	. ,
					he conte			ory	location is	sign	extended	to 40 bits
			•	su	ubtractio	n borro	w bit is rep	or	RY status ted in the Ca of the CARF	ARRY	status bit;	
					/hen an c SATD.	overflow	v is detecte	ed,	the accumu	ulator is	saturated	laccording
			W	hen	the des	tinatior	operand	(ds	st) is an aux	iliary o	r tempora	ry register:
				Th	he opera	ition is	performed	lo	n 16 bits in	the A-ι	unit ALU.	
									ce operand are used to p			
				0	verflow	detectio	on is done	at	bit position	15.		
					/hen an ccording			cte	d, the desti	nation	register is	s saturated
		Со	mp	oatib	oility wit	h C54	devices	(C	54CM = 1)			
		Wh	nen	this	instruct	ion is e	executed w	/ith	n M40 = 0, c	ompat	ibility is er	nsured.

Status Bits	Affected by	M40, SATA, SATD, SXMD					
	Affects	ACOVx, CARRY					
Repeat	This instructio	This instruction can be repeated.					
Example							
Syntax	Description						
AC0 = AC1 - *AR3	The content ac	ddressed by AR3 is subtracted from the content of AC1 and the result					

is stored in AC0.

# **Syntax Characteristics**

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline			
[5]	dst = Smem – si	rc			No	3	1	Х			
Opcod	e			1101	1000 AA	AA A	AAI   FDI	D FSSS			
Operar	nds	dst, S	Smem, src								
Description		This instruction subtracts a register content from the content of a memory (Smem) location.									
		When the destination operand (dst) is an accumulator:									
		■ The operation is performed on 40 bits in the D-unit ALU.									
		I	<ul> <li>If an auxiliary or ter instruction, the 16 L extended according</li> </ul>	SBs of the	auxiliary or		•	• •			
			The content of the according to SXME	•	location is	sign e	extended	to 40 bits			
		I	<ul> <li>Overflow detection subtraction borrow bit is the logical con</li> </ul>	bit is repor	ted in the CA	ARRY	status bit;				
			When an overflow is to SATD.	s detected,	the accumu	lator is	saturated	laccording			
			Vhen the destination o	perand (de	st) is an aux	iliary o	r tempora	ry register:			
			The operation is pe	erformed o	n 16 bits in t	he A-u	ınit ALU.				
		I	If an accumulator i 16 LSBs of the acc		•	• •					
			Overflow detection	is done at	bit position	15.					
			When an overflow according to SATA.		d, the destir	nation	register is	saturated			
		Com	patibility with C54x c	levices (C	54CM = 1)						
		Whe	n this instruction is exe	ecuted with	n M40 = 0, c	ompati	bility is er	sured.			

SPRU375G

Status Bits	Affected by	M40, SATA, SATD, SXMD					
	Affects	ACOVx, CARRY					
Repeat	This instruction	This instruction can be repeated.					
Example							
Syntax	Description						
AC0 = *AR3 – AC1	The content of A	AC1 is subtracted from the content addressed by AR3 and the result					

is stored in AC0.

Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline				
[6]	ACy = ACy - c	(ACx << Tx <b>)</b>		Yes	2	1	Х				
Opcod	le			01	01 10	D1E DDS	SS ss01				
Opera	nds	ACx, ACy, Tx									
Descri	ption	This instruction subtracts an accumulator content ACx shifted by the content of Tx from an accumulator content ACy.									
		☐ The operation is performed on 40 bits in the D-unit shifter.									
		Input operands are sign extended to 40 bits according to SXMD.									
		The shift operation is equivalent to the signed shift instruction.									
		Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.									
		When an overflow is detected, the accumulator is saturated according to SATD.									
		Compatibility with C54x devices (C54CM = 1)									
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1:									
			ary shift operation is p detection,report,an			-					
		□ The 6 LSBs of Tx are used to determine the shift quantity. The 6 LS Tx define a shift quantity within -32 to +31. When the value is betwee to -17, a modulo 16 operation transforms the shift quantity to withi to -1.									
Status	Bits	Affected by 0	C54CM, M40, SATD,	SXMD							
		Affects A	ACOVy, CARRY								
Repea	t	This instruction c	an be repeated.								
Evamr											

Syntax	Description
AC0 = AC0 - (AC1 << T0)	The content of AC1 shifted by the content of T0 is subtracted from the content of
	AC0 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline				
[7]	_	ACx << #SHIFTW	/)	Yes	3	1	X				
Opcod	e		0001	000E DD	SS 0:	100 xxS	GH IFTW				
Operar	nds	ACx, ACy, S⊦	IIFTW								
Descri	ption	This instruction subtracts an accumulator content ACx shifted by the 6-bit value, SHIFTW, from an accumulator content ACy.									
		The operation	☐ The operation is performed on 40 bits in the D-unit shifter.								
		Input ope	Input operands are sign extended to 40 bits according to SXMD.								
		The shift operation is equivalent to the signed shift instruction.									
		Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.									
		When an SATD.	When an overflow is detected, the accumulator is saturated according to SATD.								
		Compatibility with C54x devices (C54CM = 1)									
		C54CM = 1, a	truction is executed with an intermediary shift ope no overflow detection, r tion.	eration is per	rforme	d as if M4	0 is locally				
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD							
		Affects	ACOVy, CARRY								
Repeat	t	This instruction	on can be repeated.								

Syntax	Description
AC0 = AC0 - (AC1 << #31)	The content of AC1 shifted left by 31 bits is subtracted from the content of AC0 and the result is stored in AC0.

# **Syntax Characteristics**

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline					
	ACx – <b>(</b> K16 <b>&lt;&lt; #16)</b>	No	4	1	X					
Opcode	0111 1010 KKKI	к кккк   кк	KK KI	KKK SSI	DD 001x					
Operands	ACx, ACy, K16									
Description	This instruction subtracts the 16-bit s from an accumulator content ACx.	This instruction subtracts the 16-bit signed constant, K16, shifted left by 16 bits from an accumulator content ACx.								
	The operation is performed on 4	The operation is performed on 40 bits in the D-unit ALU.								
	Input operands are sign extended	Input operands are sign extended to 40 bits according to SXMD.								
	The shift operation is equivalent	The shift operation is equivalent to the signed shift instruction.								
	subtraction borrow bit is reporte	Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.								
	When an overflow is detected, the SATD.	an overflow is detected, the accumulator is saturated according to								
	Compatibility with C54x devices (	С54CM = 1)								
	When this instruction is executed with C54CM = 1, an intermediary shift or set to 1 and no overflow detection, shifting operation.	peration is pe	rforme	d as if M4	0 is locally					
Status Bits	Affected by C54CM, M40, SATE	D, SXMD								
	Affects ACOVy, CARRY									
Repeat	This instruction can be repeated.									

Syntax	Description
AC0 = AC1 – (FFFFh << #16)	A signed 16-bit value (FFFFh) shifted left by 16 bits is subtracted from the
	content of AC1 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline		
[9]	-	K16 << #SHFT)		No	4	1	X		
Opcod	e		0111 0001 KKKK	κκκκ κκ	KK KI	KKK   SSI	DD SHFT		
Operar	nds	ACx, ACy, K16	, SHFT						
Descri	ption		n subtracts the 16-bit s FT, from an accumulat	•		6, shifted	left by the		
		The operat	tion is performed on 40	) bits in the [	D-unit s	shifter.			
		Input operands are sign extended to 40 bits according to SXMD.							
		The shift operation is equivalent to the signed shift instruction.							
		Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.							
		When an overflow is detected, the accumulator is saturated according to SATD.							
		Compatibility with C54x devices (C54CM = 1)							
		C54CM = 1, ar	uction is executed with n intermediary shift ope o overflow detection, r on.	eration is per	formed	d as if M4	0 is locally		
Status	Bits	Affected by	M40, SATD, SXMD						
		Affects	ACOVy, CARRY						
Repeat	t	This instruction	n can be repeated.						

Syntax	Description
AC1 = AC0 - (#9800h << #5)	A signed 16-bit value (9800h) shifted left by 5 bits is subtracted from the
	content of AC0 and the result is stored in AC1.

**Syntax Characteristics** 

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline			
[10]	ACy = ACx -	(Smem << Tx)		No	3	1	Х			
Opcod	e		1101	1101 AA	AA A	AAI SSI	DD ss01			
Operai	nds	ACx, ACy, Smem, Tx								
Descri	ption		on subtracts the content f Tx from an accumulate	•	•	n) locatior	n shifted by			
		The oper	ation is performed on 40	D bits in the I	D-unit s	shifter.				
		Input ope	Input operands are sign extended to 40 bits according to SXMD.							
		The shift	The shift operation is equivalent to the signed shift instruction.							
		subtractio	Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow b is the logical complement of the CARRY status bit.							
		When an overflow is detected, the accumulator is saturated according to SATD.								
		Compatibility with C54x devices (C54CM = 1)								
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1:								
			ediary shift operation is p ow detection, report, ar			•				
		Tx define	Bs of Tx are used to det a shift quantity within –3 modulo 16 operation tra	2 to +31. Wh	en the	value is be	etween –32			
Status	Bits	Affected by	C54CM, M40, SATD	, SXMD						
		Affects	ACOVy, CARRY							
Repeat	t	This instruction	on can be repeated.							
_	-									

Syntax	Description
AC0 = AC1 - (*AR3 << T0)	The content addressed by AR3 shifted by the content of T0 is subtracted from the content of AC1 and the result is stored in AC0.

# **Syntax Characteristics**

No. Syntax	Parallel Enable Bit Size Cycles Pipeline							
[11] ACy = AC	Cx - (Smem << #16)							
Opcode	1101 1110 AAAA AAAI SSDD 0101							
Operands	ACx, ACy, Smem							
Description	This instruction subtracts the content of a memory (Smem) location shifted left by 16 bits from an accumulator content ACx.							
	The operation is performed on 40 bits in the D-unit ALU.							
	Input operands are sign extended to 40 bits according to SXMD.							
	The shift operation is equivalent to the signed shift instruction.							
	Overflow detection and CARRY status bit depends on M40. If the result of the subtraction generates a borrow, the CARRY status bit is cleared; otherwise, the CARRY status bit is not affected.							
	When an overflow is detected, the accumulator is saturated according to SATD.							
	Compatibility with C54x devices (C54CM = 1)							
	When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is locally set to 1 and no overflow detection, report, and saturation is done after the shifting operation.							
Status Bits	Affected by C54CM, M40, SATD, SXMD							
	Affects ACOVy, CARRY							
Repeat	This instruction can be repeated.							

Syntax	Description
AC0 = AC1 - (*AR3 << #16)	The content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

# **Syntax Characteristics**

No.	Syntax			E	Parallel Enable Bit	Size	Cycles	Pipeline		
[12]	ACy = (Smem	<< #16) – AC	K		No	3	1	Х		
Opcod	е			1101	1110 AA	AA AA	AAI SSI	DD 0110		
Operar	nds	ACx, ACy,	Smem							
Descri	ption	This instruction subtracts an accumulator content ACx from the content of a memory (Smem) location shifted left by 16 bits.								
		🗋 The o	The operation is performed on 40 bits in the D-unit ALU.							
		Input operands are sign extended to 40 bits according to SXMD.								
		The shift operation is equivalent to the signed shift instruction.								
		Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.								
		U When SATD	an overflow is detec	ted, the a	accumulato	or is sa	turated ad	ccording to		
		Compatibility with C54x devices (C54CM = 1)								
		C54CM =	instruction is execute 1, an intermediary s nd no overflow dete eration.	shift opera	ation is per	formed	d as if M4	0 is locally		
Status	Bits	Affected b	y C54CM, M40,	SATD, S	SXMD					
		Affects	ACOVy, CARF	RY						
Repeat	t	This instru	ction can be repeate	ed.						

Syntax	Description
AC0 = (*AR3 << #16) – AC1	The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[13]	ACy = ACx – u	ins(Smem) – BORROW	No	3	1	Х			
Opcod		L. L	l 1111   AA	AA A	AAI SSI	DD 101u			
Operar	nds	ACx, ACy, Smem							
Descri	ption	This instruction subtracts the logical complement of the CARRY status bit (borrow) and the content of a memory (Smem) location from an accumulator content ACx.							
		The operation is performed on 4	0 bits in the D	D-unit A	ALU.				
		Input operands are extended to	40 bits accore	ding to	uns.				
		If the optional uns keyword is of the memory location is ze		•	•	he content			
		<ul> <li>If the optional uns keyword content of the memory location SXMD.</li> </ul>	• •		• •				
		Overflow detection and CARF subtraction borrow bit is reported is the logical complement of the	d in the CARF	RY stat					
		When an overflow is detected, th SATD.	ne accumulato	or is sa	iturated ad	ccording to			
		Compatibility with C54x devices (	C54CM = 1)						
		When this instruction is executed wi	th M40 = 0, co	ompati	bility is er	sured.			
Status	Bits	Affected by CARRY, M40, SATE	), SXMD						
		Affects ACOVy, CARRY							
Repeat	t	This instruction can be repeated.							

5-482 Instruction Set Descriptions

Syntax				Description		
AC1 = AC0 – uns(*AR1) – BORROW				The complement of the CARRY bit (1) and the unsigned content addressed by AR1 (F000h) are subtracted from the content of AC0 and the result is stored in AC1.		
Before			Af	fter		
AC0	00 EC00	0000	AC	CO 00 EC00 0000		
AC1	00 0000	0000	AC	C1 00 EBFF 0FFF		
AR1		0302	AF	R1 0302		
302		F000	30	02 F000		
CARRY		0	CA	ARRY 1		

# Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
$[14] \qquad ACy = ACx - u$	ns(Smem)	No	3	1	Х		
Opcode	110	1 1111 AA	AA A	AAI SSI	DD 111u		
Operands	ACx, ACy, Smem						
Description	This instruction subtracts the conte accumulator content ACx.	nt of a memor	y (Sme	em) locatio	on from an		
	The operation is performed on a	40 bits in the I	D-unit /	ALU.			
	Input operands are extended to 40 bits according to uns.						
	If the optional uns keyword i of the memory location is z	•	he content				
	If the optional uns keyword content of the memory locat SXMD.			• •			
	Overflow detection and CARRY status bit depends on M40. subtraction borrow bit is reported in the CARRY status bit; the borro is the logical complement of the CARRY status bit.						
	When an overflow is detected, t SATD.	<b>3</b> • • • • • • • • • • • • • • • • • • •					
	Compatibility with C54x devices	(C54CM = 1)					
	When this instruction is executed w	ith M40 = 0, c	ompati	bility is er	sured.		
Status Bits	Affected by M40, SATD, SXMD	)					
	Affects ACOVy, CARRY						
Repeat	This instruction can be repeated.						

Syntax	Description
	The unsigned content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

# Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
[15]	-	– <mark>(uns(</mark> Smem) << #SHIFTW <b>)</b>	No	4	1	X		
Opcode	9	1111 1001 AAAA	AAAI ux	SH I	FTW SSI	DD 01xx		
Operands		ACx, ACy, SHIFTW, Smem						
Description		This instruction subtracts the content of a memory (Smem) location shifted by the 6-bit value, SHIFTW, from an accumulator content ACx.						
		The operation is performed on 40 bits in the D-unit shifter.						
		Input operands are extended to 40 bits according to uns.						
		If the optional uns keyword is of the memory location is ze		•	•	he content		
		If the optional uns keyword content of the memory location SXMD.						
		The shift operation is equivalent	to the signed	l shift i	nstruction			
		Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.						
		When an overflow is detected, the SATD.	e accumulato	or is sa	turated ad	ccording to		
		Compatibility with C54x devices (	C54CM = 1)					
		When this instruction is executed with C54CM = 1, an intermediary shift op set to 1 and no overflow detection, shifting operation.	eration is per	forme	d as if M4	0 is locally		
Status	Bits	Affected by C54CM, M40, SATE	, SXMD					
		Affects ACOVy, CARRY						

# RepeatThis instruction cannot be repeated when using the \*(#k23) absolute address-<br/>ing mode to access the memory operand (Smem); when using other address-<br/>ing modes, this instruction can be repeated.

Syntax	Description
AC0 = AC1 - (uns(*AR3) << #31)	The unsigned content addressed by AR3 shifted left by 31 bits is subtracted from the content of AC1 and the result is stored in AC0.

Subtraction

# Syntax Characteristics

		Parallel					
No. Syntax		Enable Bit	Size	Cycles	Pipeline		
[16]  ACy = ACx - c	dbl(Lmem)	No	3	1	Х		
Opcode	1110	1101 AA	AA A	AAI SSI	DD 001n		
Operands	ACx, ACy, Lmem	ACx, ACy, Lmem					
Description	This instruction subtracts the content of data memory operand dbl(Lr from an accumulator content ACx.						
	The data memory operand dbl(L	mem) addres	sses ai	e aligned	:		
	<ul> <li>if Lmem address is even: significant word = Lmem + 1</li> </ul>	most signifi	cant w	rord = Ln	nem, least		
	If Lmem address is odd: most significant word = Lmem, significant word = Lmem − 1						
	The operation is performed on 40 bits in the D-unit ALU.						
	Input operands are sign extended to 40 bits according to SXMI						
	subtraction borrow bit is reported	Overflow detection and CARRY status bit depends on M40. The subtraction borrow bit is reported in the CARRY status bit; the borrow is the logical complement of the CARRY status bit.					
	When an overflow is detected, th SATD.	e accumulate	or is sa	iturated a	ccording to		
	Compatibility with C54x devices (0	C54CM = 1)					
	When this instruction is executed wit	h M40 = 0, c	ompati	bility is er	sured.		
Status Bits	Affected by M40, SATD, SXMD						
	Affects ACOVy, CARRY						
Repeat	This instruction can be repeated.						
Example							

#### Ψ

Syntax	Description
AC0 = AC1 - dbl(*AR3+)	The content (long word) addressed by AR3 and AR3 + 1 is subtracted from the content of AC1 and the result is stored in AC0. Because this instruction is a long-operand instruction, AR3 is incremented by 2 after the execution.

Subtraction

#### Syntax Characteristics

No. Syntax		Parallel Enable Bit	Size	Cycles	Pipeline
[17] ACy = <b>dbl(</b> Lme	m) – ACx	No	3	1	Х
Opcode	1110	1101 AA	AA A	AAI SSI	DD 010x
Operands	ACx, ACy, Lmem				
Description	This instruction subtracts an accumula memory operand dbl(Lmem).	ator content A	ACx fro	m the cont	tent of data
	The data memory operand dbl(Lr	mem) addres	sses ar	e aligned	:
	If Lmem address is even: significant word = Lmem + 1	ord = Ln	nem, least		
	If Lmem address is odd: r significant word = Lmem − 1	nost signific	ant w	ord = Ln	nem, least
	☐ The operation is performed on 40 bits in the D-unit ALU.				
	Input operands are sign extended to 40 bits according to S>				
	Overflow detection and CARRY status bit depends on M40. subtraction borrow bit is reported in the CARRY status bit; the borro is the logical complement of the CARRY status bit.				
	When an overflow is detected, the SATD.	e accumulato	or is sa	turated ad	ccording to
	Compatibility with C54x devices (C	C54CM = 1)			
	When this instruction is executed with	n M40 = 0, c	ompati	bility is er	sured.
Status Bits	Affected by M40, SATD, SXMD				
	Affects ACOVy, CARRY				
Repeat	This instruction can be repeated.				

Syntax	Description	
AC0 = dbl(*AR3) - AC1	The content of AC1 is subtracted from the content (long word) addressed by AR3 and AR3 + 1 and the result is stored in AC0.	

#### Subtraction

#### **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[18]	ACx = (Xmem	No	3	1	х			
Opcod	e		1000	0001 XX	XM MI	MYY YMM	IM 01DD	
Operar	nds	ACx, Xmem, Y	mem					
Descri	ption		This instruction subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, shifted 16 bits.					
		The opera	tion is performed on 40	bits in the [	D-unit A	ALU.		
		Input oper	ands are sign extended	l to 40 bits a	iccordi	ng to SXN	/ID.	
		The shift operation is equivalent to the signed shift instruction						
		Overflow detection and CARRY status bit depends on M40. T subtraction borrow bit is reported in the CARRY status bit; the borrow is the logical complement of the CARRY status bit.						
		When an o SATD.	When an overflow is detected, the accumulator is saturated according SATD.					
		Compatibility	with C54x devices (C	54CM = 1)				
		When this instruction is executed with $M40 = 0$ , compatibility is ensured. When C54CM = 1, an intermediary shift operation is performed as if M40 is loca set to 1 and no overflow detection, report, and saturation is done after t shifting operation.					0 is locally	
Status	Bits	Affected by	C54CM, M40, SATD,	SXMD				
		Affects	ACOVx, CARRY					
Repeat	t	This instruction	n can be repeated.					
Examp	le							

Syntax	Description
	The content addressed by AR4 shifted left by 16 bits is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0.

## Subtraction with Parallel Store Accumulator Content to Memory

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	ACy = (Xmem - Ymem = <b>HI(</b> AC			No	4	1	Х	
Opcod	e		1000 0111 XXXM	MMYY YMI	MM SS	SDD   101	x xxxx	
Opera	nds	AC	x, ACy, T2, Xmem, Ymem					
Descri	ption	Thi	s instruction performs two operation	ons in paralle	el: subt	raction ar	nd store.	
			e first operation subtracts an accur mory operand Xmem shifted left b		nt from	n the cont	ent of data	
			The operation is performed on 40 bits in the D-unit ALU.					
			Input operands are sign extended	d to 40 bits a	ccordir	ng to SXM	ID.	
		The shift operation is equivalent to the signed shift instruction.						
			Overflow detection and CARR subtraction borrow bit is reported is the logical complement of the C intermediary shift operation is per no overflow detection, report, ar operation.	in the CARR CARRY status rformed as if	Y statu s bit. W M40 is	us bit; the /hen C54( s locally s	borrow bit CM = 1, an et to 1 and	
			When an overflow is detected, the SATD.	e accumulato	or is sat	turated ad	ccording to	
		sto is r	e second operation shifts the accures ACy(31–16) to data memory on the shift is strong within –32 to +31, the shift is strong with this value.	perand Yme	m. If th	e 16-bit v	alue in T2/	
			The input operand is shifted in th	e D-unit shift	er acco	ording to	SXMD.	
			After the shift, the high part of the the memory location.	e accumulato	r, ACy(	(31–16), i	s stored to	
		Co	mpatibility with C54x devices (C	C54CM = 1)				
		this det to	en this instruction is executed with instruction is executed with C54 ermine the shift quantity. The 6 LSE +31. When the 16-bit value in T2 eration transforms the shift quantity	CM = 1, the Bs of T2 defin is between	6 LSB e a shit -32 to	s of T2 a it quantity o –17, a i	re used to within –32	

Status Bits	Affected by	C54CM, M40, SATD, SXMD
	Affects	ACOVy, CARRY
Repeat	This instruction	n can be repeated.
See Also	See the follow	ing other related instructions:
	Addition or	r Subtraction Conditionally
	Addition or	r Subtraction Conditionally with Shift
	Addition, S	Subtraction, or Move Accumulator Content Conditionally
	Dual 16-Bi	it Addition and Subtraction
	Dual 16-Bi	it Subtractions
	Dual 16-Bi	it Subtraction and Addition
	Subtraction	n
	Subtract C	Conditionally
<b>F</b> actoria de		

Syntax	Description
AC0 = (*AR3 << #16) – AC1, *AR4 = HI(AC0 << T2)	Both instructions are performed in parallel. The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0. The content of AC0 is shifted by the content of T2, and AC0(31–16) is stored at the address of AR4.

# Swap Accumulator Content

## **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
	swap(ACx, AC)	y)					
[1]	swap(AC0, AC		Yes	2	1	Х	
[2]	swap(AC1, AC	3)		Yes	2	1	Х
Opcod	e	swap(AC0,	AC2)	010	)1 11	1E 000	0 0000
		swap(AC1,	AC3)	010	)1 11	1E 000	0 0001
Operar	nds	ACx, ACy					
Descri	Description This instruction performs operations are performe operators.						
		accumulator (	n moves the content of the ACy), and reciprocally the first accumulator.				
		Accumulator s	wapping is performed in	n the execute	e phase	e of the p	ipeline.
Status	Bits	Affected by	none				
		Affects	none				
Repeat	t	This instructior	n can be repeated.				
See Al	so	See the followi	ing other related instruc	ctions:			
		Swap Accu	umulator Pair Content				
		🗋 Swap Auxi	iliary Register Content				
		🗋 Swap Auxi	iliary and Temporary Re	egister Conte	ent		
		Swap Tem	porary Register Conter	nt			

Syntax		Descr	Description				
swap(AC0	, AC2)	The co	The content of AC0 is moved to AC2 and the content of AC2 is moved to AC0				
Before			After				
AC0	01 E500	0030	AC0	00 2800 0200			
AC2	00 2800	200	AC2	01 E500 0030			

# Swap Accumulator Pair Content

## Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	swap(pair(A0	C0), pair(AC2))	Yes	2	1	Х		
Opcod	e		010	)1 11	1E 000	1 0000		
Operar	nds	AC0, AC2	·		•			
Descri	ption	This instruction performs two parallel moves between four accumulators (AC0 and AC2, AC1 and AC3) in one cycle. These operations are performed in a dedicated datapath independent of the D-unit operators. Accumulator swapping is performed in the execute phase of the pipeline.						
		This instruction performs two parallel moves:						
		the content of AC0 to AC2, and respectively.	eciprocally th	e conte	ent of AC	2 to AC0		
		the content of AC1 to AC3, and read to AC3.	eciprocally th	e conte	ent of AC	3 to AC1		
Status	Bits	Affected by none						
		Affects none						
Repeat	t	This instruction can be repeated.						
See Al	so	See the following other related instruct	ctions:					
		Swap Accumulator Content						
		Swap Auxiliary Register Pair Con	itent					
		Swap Auxiliary and Temporary Research	egister Pair C	Conten	t			
		Swap Temporary Register Pair C	ontent					

## Example

Syntax		Description					
swap(pair	(AC0), pair(AC2))	AC0 is moved	The following two swap instructions are performed in parallel: the content AC0 is moved to AC2 and the content of AC2 is moved to AC0, and the content of AC1 is moved to AC3 and the content of AC3 is moved to AC1.				
Before		After					
AC0	01 E500 0030	AC0	00 2800 0200				
AC1	00 FFFF 0000	AC1	00 8800 0800				
AC2	00 2800 0200	AC2	01 E500 0030				
AC3	00 8800 0800	AC3	00 FFFF 0000				

#### SPRU375G

# Swap Auxiliary Register Content

## **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
	swap(ARx, AR	y)			0.20	.,	
[1]	swap(AR0, AR	1)		Yes	2	1	AD
[2]	swap(AR0, AR	2)		Yes	2	1	AD
[3]	swap(AR1, AR	3)		Yes	2	1	AD
Opcod	е	swap(AR0,	AR1)	010	)1 11	1E 001	1 1000
•		swap(AR0,		010		1E 000	
		- swap(AR1,	AR3)	010	)1 11	.1E 000	0 1001
Opera	nds	ARx, ARy		I		I	
Descri	ption		n performs parallel m ons are performed in a s.				•
This instruction moves the content of the first auxiliary regist second auxiliary register (ARy), and reciprocally moves the second auxiliary register to the first auxiliary register.				•	,		
		Auxiliary regist	er swapping is perform	ed in the add	ress pł	nase of th	e pipeline.
Status	Bits	Affected by	none				
		Affects	none				
Repea	t	This instruction	n can be repeated.				
See Al	so	See the followi	ng other related instrue	ctions:			
		🗋 Swap Accu	umulator Content				
		🗋 Swap Auxi	liary and Temporary R	egister Conte	ent		
		🗋 Swap Auxi	liary Register Pair Cor	itent			
		🗋 Swap Tem	porary Register Conte	nt			
		-					

# Example

Syntax		Description	Description				
swap(AR0, AR2)		The content of A	The content of AR0 is moved to AR2 and the content of AR2 is moved to AF				
Before		After					
AR0	6500	AR0	0300				
AR2	0300	AR2	6500				

5-494 Instruction Set Descriptions

SPRU375G

# Swap Auxiliary Register Pair Content

Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
[1]	swap(pair(AR	0), pair(AR2))	Yes	2	1	AD		
Opcod	e		010	)1 11	1E 000	1 1000		
Operar	nds	AR0, AR2						
Descri	ption	This instruction performs two parallel moves between four auxiliary registers (AR0 and AR2, AR1 and AR3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Auxiliary register swapping is performed in the address phase of the pipeline.						
		This instruction performs two parallel moves:						
		the content of AR0 to AR2, and re	eciprocally th	e conte	ent of AR	2 to AR0		
		the content of AR1 to AR3, and re	eciprocally th	e conte	ent of AR	3 to AR1		
Status	Bits	Affected by none						
		Affects none						
Repeat	t	This instruction can be repeated.						
See Al	so	See the following other related instruct	ctions:					
		Swap Accumulator Pair Content						
		Swap Auxiliary Register Content						
		Swap Auxiliary and Temporary Re	egister Pair C	Conten	t			
		Swap Temporary Register Pair Control	ontent					

Syntax		Description					
swap(pair(AR0), pair(AR2))		AR0 is mo	The following two swap instructions are performed in parallel: the content AR0 is moved to AR2 and the content of AR2 is moved to AR0, and the content of AR1 is moved to AR3 and the content of AR3 is moved to AR1.				
Before		After					
AR0	0200	AR0	6788				
AR1	0300	AR1	0200				
AR2	6788	AR2	0200				
AR3	0200	AR3	0300				

## Swap Auxiliary and Temporary Register Content

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
	swap(ARx, Tx)							
[1]	swap(AR4, T0)		Yes	2	1	AD		
[2]	swap(AR5, T1)		Yes	2	1	AD		
[3]	swap(AR6, T2)		Yes	2	1	AD		
[4]	swap(AR7, T3)		Yes	2	1	AD		
Opcod	e	swap(AR4, T0)	010	)1 11	1E 000	0 1100		
		swap(AR5, T1)	010	01 11	1E 000	0 1101		
		swap(AR6, T2)	010	)1 11	1E 000	0 1110		
		swap(AR7, T3)	010	)1 11	1E 000	0 1111		
Operar	nds	ARx, Tx						
Descrij	ption	This instruction performs parallel moves between auxiliary registers and temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.						
		This instruction moves the content of the auxiliary register (ARx) to the temporary register (Tx), and reciprocally moves the content of the temporary register to the auxiliary register.						
		Auxiliary and temporary register swap of the pipeline.	oping is perfo	rmed ir	the addr	ess phase		
Status	Bits	Affected by none						
		Affects none						
Repeat	t	This instruction can be repeated.						
See Als	SO	See the following other related instructions:						
		Swap Accumulator Content						
		Swap Auxiliary Register Content						
		Swap Auxiliary and Temporary Register Pair Content						
		Swap Auxiliary and Temporary Register Pairs Content						
		Swap Temporary Register Conte	nt					

Syntax		Description	Description				
swap(AR4, T	0)	The content of A	R4 is moved to T0 and the content of T0 is moved to AR4.				
		_					
Before		After					
Т0	6500	тО	0300				
AR4	0300	AR4	6500				

## Swap Auxiliary and Temporary Register Pair Content

No	Suptax		Parallel	Sizo	Cycles	Dincline			
No.	Syntax	nair/Ty))	Enable Bit	Size	Cycles	Pipeline			
[4]	swap(pair(ARx)		Vee	2	1				
[1]	swap(pair(AR4		Yes	_		AD			
[2]	swap(pair(AR6	), pair(12))	Yes	2	1	AD			
Opcode	9	<pre>swap(pair(AR4), pair(T0))</pre>	010	)1 11	1E 000	1 1100			
		<pre>swap(pair(AR6), pair(T2))</pre>	010	)1 11	1E 000	1110			
Operan	ds	ARx, Tx							
Descrip	otion	This instruction performs two parallel moves between two auxiliary registers and two temporary registers in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Auxiliary and temporary register swapping is performed in the address phase of the pipeline.							
		Instruction [1] performs two parallel moves:							
		the content of AR4 to T0, and reciprocally the content of T0 to AR4							
		the content of AR5 to T1, and reciprocally the content of T1 to AR5							
		Instruction [2] performs two parallel n	noves:						
		the content of AR6 to T2, and rec	iprocally the	conter	t of T2 to	AR6			
		the content of AR7 to T3, and rec	iprocally the	conter	nt of T3 to	AR7			
Status	Bits	Affected by none							
		Affects none							
Repeat		This instruction can be repeated.							
See Als	50	See the following other related instrue	ctions:						
		Swap Accumulator Pair Content							
		Swap Auxiliary Register Pair Con	itent						
		Swap Auxiliary and Temporary R	egister Conte	ent					
		Swap Auxiliary and Temporary R	egister Pairs	Conte	nt				
		Swap Temporary Register Pair Content							

Example	
---------	--

Syntax		Description					
swap(pair(AR4), pair(T0))		AR4 is mo	The following two swap instructions are performed in parallel: the content of AR4 is moved to T0 and the content of T0 is moved to AR4, and the content of AR5 is moved to T1 and the content of T1 is moved to AR5.				
Before		364					
Berore		After					
AR4	0200	AR4	6788				
AR5	0300	AR5	0200				
т0	6788	Т0	0200				
Т1	0200	Т1	0300				

## Swap Auxiliary and Temporary Register Pairs Content

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	swap(block(Al	R4), block(T0))		Yes	2	1	AD	
Opcod	e			010	)1 11	1E 001	.0 1100	
Operar	nds	AR4, T0						
Description		This instruction performs four parallel moves between four auxiliary registers (AR4, AR5, AR6, and AR7) and four temporary registers (T0, T1, T2, and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Auxiliary and temporary register swapping is performed in the address phase of the pipeline.						
		This instruction performs four parallel moves:						
		the content o	of AR4 to T0, and reci	iprocally the	conter	t of T0 to	AR4	
		the content o	of AR5 to T1, and reci	iprocally the	conter	t of T1 to	AR5	
		the content o	of AR6 to T2, and reci	iprocally the	conter	t of T2 to	AR6	
		the content o	of AR7 to T3, and reci	iprocally the	conter	t of T3 to	AR7	
Status	Bits	Affected by r	none					
		Affects r	none					
Repeat	:	This instruction c	an be repeated.					
See Als	50	See the following	other related instruc	tions:				
		Swap Auxilia	ry and Temporary Re	egister Conte	ent			
		Swap Auxilia	ry and Temporary Re	egister Pair C	Conten	t		

Example								
Syntax		Descriptio	Description					
swap (block(	AR4), block(T0))	The following four swap instructions are performed in parallel: the content of AR4 is moved to T0 and the content of T0 is moved to AR4, the content of AR5 is moved to T1 and the content of T1 is moved to AR5, the content of AR6 is moved to T2 and the content of T2 is moved to AR6, and the content of AR7 is moved to T3 and the content of T3 is moved to AR7.						
Before		After						
AR4	0200	AR4	0030					
AR5	0300	AR5	0200					
AR6	0240	AR6	3400					
AR7	0400	AR7	0FD3					
т0	0030	Т0	0200					
T1	0200	T1	0300					
Т2	3400	Т2	0240					
Т3	0FD3	Т3	0400					

# Swap Temporary Register Content

## Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline		
	<b>swap(</b> Tx, Ty <b>)</b>							
[1]	swap(T0, T2)		Yes	2	1	AD		
[2]	swap(T1, T3)		Yes	2	1	AD		
Opcod	e	swap(T0, T2)	010	)1 11	1E 000	0 0100		
		<pre>swap(T1, T3)</pre>	010	)1 11	1E 000	0 0101		
Operar	nds	Тх, Ту			·			
Descri	ption	This instruction performs parallel moves between two temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.						
		This instruction moves the content of the first temporary register (Tx) to the second temporary register (Ty), and reciprocally moves the content of the second temporary register to the first temporary register.						
		Temporary register swapping is p pipeline.	performed in th	ne ado	lress pha	ase of the		
Status	Bits	Affected by none						
		Affects none						
Repeat	t	This instruction can be repeated.						
See Al	so	See the following other related inst	ructions:					
		Swap Accumulator Content						
		Swap Auxiliary Register Content	nt					
		Swap Auxiliary and Temporary	Register Conte	ent				
		Swap Temporary Register Pair	Content					

Syntax		Description	Description				
swap(T0, T2)	The content of T0 is moved to T2 and the content of T2 is moved to T0.						
Before		After					
Т0	6500	Т0	0300				
Т2	0300	Т2	6500				

# Swap Temporary Register Pair Content

Syntax Characteristics

No.	Syntax		Parallel Enable Bit	Size	Cycles	Pipeline			
[1]	swap(pair(T(	0), pair(T2))	Yes	2	1	AD			
Opcod	e		010	)1 11	1E 000	1 0100			
Operar	nds	T0, T2							
Descri	ption	(T0 and T2, T1 and T3) in one cycle dedicated datapath independent of the	This instruction performs two parallel moves between four temporary registers (T0 and T2, T1 and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators. Temporary register swapping is performed in the address phase of the pipeline.						
		This instruction performs two parallel	moves:						
		the content of T0 to T2, and recip	procally the co	ontent	of T2 to T	ГО			
		the content of T1 to T3, and recip	procally the co	ontent	of T3 to T	Г1			
Status	Bits	Affected by none							
		Affects none							
Repeat	t	This instruction can be repeated.							
See Al	so	See the following other related instruct	ctions:						
		Swap Accumulator Pair Content							
		Swap Auxiliary Register Pair Con	itent						
		Swap Auxiliary and Temporary R	egister Pair C	Conten	t				
		Swap Temporary Register Content	nt						

Syntax		Descripti	Description					
swap(pair(T	0), pair(T2))	T0 is mov	The following two swap instructions are performed in parallel: the content of T0 is moved to T2 and the content of T2 is moved to T0, and the content of T1 is moved to T3 and the content of T3 is moved to T1.					
Before		After						
Т0	0200	т0	6788					
T1	0300	т1	0200					
Т2	6788	т2	0200					
Т3	0200	т3	0300					

# Test Accumulator, Auxiliary, or Temporary Register Bit

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	<b>TC1 = bit(</b> src, E	Badd	r)			No	3	1	Х	
[2]	<b>TC2 = bit(</b> src, E	Badd	r)			No	3	1	Х	
Opcod	e	Г	C1		1110	1100 AA	AA AA	AAI FSS	S 1000	
		Г	C2		1110	1100 AA		AAI FSS	S 1001	
Operar	nds	Ba	ddr, src, TC	x						
Descri	ption	Th	is instructior	performs a bit	manipu	ulation:				
			In the D-ur	it ALU, if the so	ource (src) register operand is an accumulator.					
The instruction tests a sing the bit addressing mode, B status bit. The generated b					r. The t	ested bit is c	opied iı		•	
			bit address	s are used to de	accumulator bits (only the 6 LSBs of the generated to determine the bit position). If the generated bit 0–39, 0 is stored into the selected TCx status bit.					
				-	•	temporary register bits (only the 4 LSBs ed to determine the bit position).				
Status	Bits	Aff	ected by	none						
		Aff	ects	TCx						
Repeat	t	Th	is instructior	a can be repeate	ed.					
See Al	SO	Se	e the followi	ng other related	instru	ctions:				
			Clear Accu	imulator, Auxilia	ry, or T	emporary R	egister	Bit		
			Compleme	nt Accumulator,	Auxilia	ary, or Temp	orary F	Register B	it	
			Set Accum	ulator, Auxiliary,	, or Ter	mporary Reg	gister B	it		
			Test Accun	nulator, Auxiliary	/, or Te	mporary Re	gister E	Bit Pair		
			Test Memo	ory Bit						

Syntax		Description					
TC1 = bit(T0, @#12)			The bit at the position defined by the register bit address (12) in T0 is tested and t tested bit is copied into TC1.				
Before		After					
тО	FEOO	Т0	FEOO				
TC1	0	TC1	1				

# Test Accumulator, Auxiliary, or Temporary Register Bit Pair

Operands       Baddr, src         Description       This instruction performs a bit manipulation: <ul> <li>In the D-unit ALU, if the source (src) register operand is an accumulation:</li> <li>In the A-unit ALU, if the source (src) register operand is an auxiliary temporary register.</li> </ul> The instruction tests two consecutive bits of the source register location	No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
Operands       Baddr, src         Description       This instruction performs a bit manipulation: <ul> <li>In the D-unit ALU, if the source (src) register operand is an accumular</li> <li>In the A-unit ALU, if the source (src) register operand is an auxiliary temporary register.</li> </ul> The instruction tests two consecutive bits of the source register location defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits a copied into status bits TC1 and TC2:            TC1 tests the bit that is defined by Baddr <ul> <li>TC2 tests the bit defined by Baddr + 1</li> </ul> The generated bit address must be within:             O –38 when accessing accumulator bits (only the 6 LSBs of the generated address is not within 0–38:             If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.             In all other cases, 0 is stored into TC1 and TC2.             If the generated address are used to determine the bit position). If the generated address is not within 0–14:             In all other cases, 0 is stored into TC1 and TC2.             If the generated address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.             In all other cases, 0 is stored into TC1 and TC2.             In the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.	[1]	<b>bit(</b> src, <b>pair(</b> Ba	ddr <b>))</b>				No	3	1	Х
Description       This instruction performs a bit manipulation:         □       In the D-unit ALU, if the source (src) register operand is an accumular         □       In the A-unit ALU, if the source (src) register operand is an auxiliary temporary register.         The instruction tests two consecutive bits of the source register location defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits a copied into status bits TC1 and TC2:         ■       TC1 tests the bit that is defined by Baddr         ■       TC2 tests the bit defined by Baddr + 1         The generated bit address must be within:       O-38 when accessing accumulator bits (only the 6 LSBs of the generated address is not within 0–38:         ■       If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.         ■       In all other cases, 0 is stored into TC1 and TC2.         ■       If the generated address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         ■       If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         ■       If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         ■       In all other cases, 0 is stored into TC1 and TC2.         ■       If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         ■       In all other cases, 0 is stored into TC1 and TC2.	Opcod	e				1110	1100 AA	AA A	AAI FSS	SS 010x
<ul> <li>In the D-unit ALU, if the source (src) register operand is an accumulated in the A-unit ALU, if the source (src) register operand is an auxiliary temporary register.</li> <li>The instruction tests two consecutive bits of the source register location defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits is copied into status bits TC1 and TC2:</li> <li>TC1 tests the bit that is defined by Baddr</li> <li>TC2 tests the bit defined by Baddr + 1</li> <li>The generated bit address must be within:</li> <li>O-38 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated address is not within O-38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> </ul>	Opera	nds	Baddr, s	src		•				
<ul> <li>In the A-unit ALU, if the source (src) register operand is an auxiliary temporary register.</li> <li>The instruction tests two consecutive bits of the source register location defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits a copied into status bits TC1 and TC2:</li> <li>TC1 tests the bit that is defined by Baddr</li> <li>TC2 tests the bit defined by Baddr + 1</li> <li>The generated bit address must be within:</li> <li>O-38 when accessing accumulator bits (only the 6 LSBs of the general bit address are used to determine the bit position). If the generated address is not within O-38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>If the generated address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> </ul>	Descri	ption	This ins	truction	n performs a l	pit manipu	lation:			
temporary register.         The instruction tests two consecutive bits of the source register location defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits a copied into status bits TC1 and TC2:         TC1 tests the bit that is defined by Baddr         TC2 tests the bit defined by Baddr + 1         The generated bit address must be within:         0-38 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated address is not within 0-38:         If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.         In all other cases, 0 is stored into TC1 and TC2.         0-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address is not within 0-14:         If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         In all other cases, 0 is stored into TC1 and TC2.         If the generated bit address is not within 0-14:         If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         In all other cases, 0 is stored into TC1 and TC2.         In all other cases, 0 is stored into TC1 and TC2.         In all other cases, 0 is stored into TC1 and TC2. </td <td></td> <td></td> <td>🗋 In ti</td> <td>ne D-ur</td> <td>nit ALU, if the</td> <td>source (s</td> <td>rc) register (</td> <td>operan</td> <td>d is an ac</td> <td>cumulator.</td>			🗋 In ti	ne D-ur	nit ALU, if the	source (s	rc) register (	operan	d is an ac	cumulator.
<ul> <li>defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits a copied into status bits TC1 and TC2:</li> <li>TC1 tests the bit that is defined by Baddr</li> <li>TC2 tests the bit defined by Baddr + 1</li> <li>The generated bit address must be within:</li> <li>0-38 when accessing accumulator bits (only the 6 LSBs of the generat bit address are used to determine the bit position). If the generated address is not within 0-38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>0-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>If the generated bit address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> </ul>						source (	src) register	· opera	nd is an a	auxiliary or
<ul> <li>TC2 tests the bit defined by Baddr + 1</li> <li>The generated bit address must be within:</li> <li>0-38 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated address is not within 0-38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>0-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>Status Bits</li> </ul>			The instruction tests two consecutive bits of the source register location a defined by the bit addressing mode, Baddr and Baddr + 1. The tested bits are copied into status bits TC1 and TC2:							
<ul> <li>The generated bit address must be within:</li> <li>0-38 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated address is not within 0-38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>0-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> </ul>				TC1 te	ests the bit that	at is define	ed by Baddr			
<ul> <li>O-38 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated address is not within 0–38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>O-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address are used to determine the bit position). If regenerated bit address is not within 0–14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC2.</li> </ul>			TC2 tests the bit defined by Baddr + 1							
<ul> <li>bit address are used to determine the bit position). If the generated address is not within 0–38:</li> <li>If the generated bit address is 39, bit 39 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>0–14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address are used to determine the bit position). If the generated bit address is not within 0–14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1.</li> </ul>			The generated bit address must be within:							
<ul> <li>TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>0-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address are used to determine the bit position). If i generated bit address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> </ul> Status Bits Affected by none			bit a	addres	s are used to	determin	• •			-
<ul> <li>0-14 when accessing auxiliary or temporary register bits (only the 4 LS of the generated address are used to determine the bit position). If it generated bit address is not within 0-14:</li> <li>If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.</li> <li>In all other cases, 0 is stored into TC1 and TC2.</li> <li>Status Bits</li> </ul>			•	-				of the r	egister is	stored into
of the generated address are used to determine the bit position). If generated bit address is not within 0–14:         If the generated bit address is 15, bit 15 of the register is stored in TC1 and 0 is stored into TC2.         In all other cases, 0 is stored into TC1 and TC2.         Status Bits       Affected by none				In all c	other cases, 0	is stored	into TC1 an	d TC2		
TC1 and 0 is stored into TC2.In all other cases, 0 is stored into TC1 and TC2.Status BitsAffected by none			of t	he gen	erated addres	ss are use	ed to detern	-	· •	
Status Bits Affected by none				-				of the r	egister is	stored into
·				In all c	other cases, 0	is stored	into TC1 an	d TC2		
Affects TC1, TC2	Status	Bits	Affected	d by	none					
			Affects		TC1, TC2					

Repeat	This instruction can be repeated.
--------	-----------------------------------

**See Also** See the following other related instructions:

- Clear Accumulator, Auxiliary, or Temporary Register Bit
- Complement Accumulator, Auxiliary, or Temporary Register Bit
- Set Accumulator, Auxiliary, or Temporary Register Bit
- Test Accumulator, Auxiliary, or Temporary Register Bit
- Test Memory Bit

Syntax	Description
bit(AC0, pair(AR1(T0)))	The bit at the position defined by the content of $AR1(T0)$ in AC0 is tested and the tested bit is copied into TC1. The bit at the position defined by the content of $AR1(T0) + 1$ in AC0 is tested and the tested bit is copied into TC2.

Before		After	
AC0	E0 1234 0000	ACO E	0 1234 0000
AR1	0026	AR1	0026
т0	0001	Т0	0001
TC1	0	TC1	1
TC2	0	TC2	0

# Test Memory Bit

No.	Syntax					Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TCx = <b>bit(</b> Smer	m, sro	;)			No	3	1	X
[2]	TCx = <b>bit(</b> Smer	m, k4)	)			No	3	1	х
Descrij	ption	inst def	tructions tes ined by eith	ions perform st a single bit o er the content tested bit is co	of a mer of the so	mory (Smem) ource (src) op	locatio	on. The b or a 4-bit	it tested is immediate
				[1], the generation [1], the generation of the second second second second second second second second second s					5 (only the
Status	Bits	Affe	ected by	none					
		Affe	ects	ТСх					
See Als	SO	See	e the followi	ng other relate	ed instru	ictions:			
			Clear Mem	nory Bit					
		Complement Memory Bit							
			Set Memor	ry Bit					
			Test Accumulator, Auxiliary, or Temporary Register Bit						
			Test Accur	nulator, Auxilia	ary, or Te	emporary Re	gister E	Bit Pair	
			Test and C	lear Memory E	Bit				
			Test and C	omplement Me	emory E	Bit			
			Test and S	et Memory Bit					

# Test Memory Bit

## Syntax Characteristics

No.	Syntax				Parallel Enable Bit	Size	Cycles	Pipeline
[1a]	TC1 = bit(Smer	m, src <b>)</b>			No	3	1	X
[1b]	TC2 = bit(Smer	m, src <b>)</b>			No	3	1	х
Opcode	9	TC1		1110	0000 AA	AA AA	AAI FSS	S xxx0
		TC2		1110	0000 AA	AA AA	AAI FSS	S xxx1
Operands		Smem, src, TC	X					
Description		This instruction performs a bit manipulation in the A-unit ALU. This instruction tests a single bit of a memory (Smem) location. The bit tested is defined by the content of the source (src) operand. The tested bit is copied into the selected TCx status bit.						
		-	bit address mus termine the bit p			lly the 4	LSBs of t	he register
Status	Bits	Affected by	none					
		Affects	ТСх					
Repeat		This instructior	n can be repeate	ed.				

# Example

\*AR0

TC1

00C0

0

\*AR0

TC1

Syntax			Descr	ption	
TC1 = bit(*	AR0, A	C0)			on defined by AC0(3–0) in the content addressed by AR0 is d bit is copied into TC1.
Before				After	
AC0	00	0000	0008	AC0	00 0000 0008

00C0

0

## Test Memory Bit

## Syntax Characteristics

					Parallel			
No.	Syntax				Enable Bit	Size	Cycles	Pipeline
[2a]	TC1 = bit(Sme	m, k4 <b>)</b>			No	3	1	Х
[2b]	TC2 = bit(Sme	<b>TC2 = bit(</b> Smem, k4 <b>)</b>			No	3	1	х
Opcode	9	TC1 TC2		1101	1100   AA 1100   AA		ļ	
Operan	ds	k4, Smem, TC	x	I	I		I	
Descrip	This instruction performs a bit manipulatests a single bit of a memory (Smem)4-bit immediate value, k4. The tested bbit.			location. Tl	ne bit te	ested is de	efined by a	
Status	Bits	Affected by	none					
		Affects	TCx					
Repeat		This instruction can be repeated.						

Syntax	Description
TC1 = bit(*AR3, #12)	The bit at the position defined by an unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1.

# Test and Clear Memory Bit

## **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TC1 = bit(Sme	m, k4 <b>), bit(</b> Smem, k4 <b>) = #0</b>		No	3	1	Х
[2]	[2] <b>TC2</b> = <b>bit(</b> Smem, k4), <b>bit(</b> Smem, k4) = <b>#0</b>			No	3	1	Х
Opcod	е	TC1	1110	0011 AA	AA AA	AAI   kkk	k 010x
		TC2	1110	0011 AA	AA AA	AAI kkk	k 011x
Operar	nds	k4, Smem, TCx					
Descri	ption	on This instruction performs a bit manipulation in the A-unit ALU. The inst tests a single bit, as defined by a 4-bit immediate value, k4, of a n (Smem) location. The tested bit is copied into status bit TCx and is cle 0 in Smem.				a memory	
Status	Bits	Affected by none					
		Affects TCx					
Repeat	t	This instruction can be repea	ated.				
See Als	so	See the following other relate	ed instrue	ctions:			
		Clear Memory Bit					
		Complement Memory Bit	t				
		Set Memory Bit					
		Test and Complement M	emory B	it			
		Test and Set Memory Bit	:				
		Test Memory Bit					

Syntax	Description
	The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR3 is cleared to 0.

# Test and Complement Memory Bit

# Syntax Characteristics

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline	
[1]	TC1 = bit(Sme	m, k4 <b>), cbit(</b> Smem, k4 <b>)</b>		No	3	1	Х	
[2]	[2] <b>TC2 = bit(</b> Smem, k4 <b>), cbit(</b> Smem, k4 <b>)</b>			No	3	1	х	
Opcod	e	TC1	1110	0011 AA	AA AA	AAI   kkk	k 100x	
		TC2	1110	0011 AA	AA AA	AAI kkk	k 101x	
Operar	nds	k4, Smem, TCx						
tests (Sme		This instruction performs a tests a single bit, as define (Smem) location and the complemented in Smem.	ed by a 4-	bit immedia	te valu	ie, k4, of	a memory	
Status	Bits	Affected by none						
		Affects TCx						
Repeat	t	This instruction can be repe	eated.					
See Al	so	See the following other rela	ated instrue	ctions:				
		Clear Memory Bit						
		Complement Memory E	Bit					
		Set Memory Bit						
		Test and Clear Memory	/ Bit					
		Test and Set Memory Bit						
		Test Memory Bit						

## Example

Syntax		Description
TC1 = bit(*AR0, #12), cbi	t(*AR0, #12)	The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR0 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR0 is complemented.
Before	After	

Before		After	
*AR0	0040	*AR0	1040
TC1	0	TC1	0

5-512 Instruction Set Descriptions

# Test and Set Memory Bit

## **Syntax Characteristics**

No.	Syntax			Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TC1 = bit(Sme	m, k4 <b>), bit(</b> Smem, k4 <b>) = #1</b>		No	3	1	Х
[2]	[2] <b>TC2</b> = <b>bit(</b> Smem, k4), <b>bit(</b> Smem, k4) = <b>#1</b>			No	3	1	Х
Opcod	е	TC1	1110	0011   AA	AA AA	AAI   kkk	k 000x
		TC2	1110	0011 AA	AA AA	AAI kkk	k 001x
Operar	nds	k4, Smem, TCx					
Descri	scriptionThis instruction performs a bit manipulation in the A-unit ALU. T tests a single bit, as defined by a 4-bit immediate value, k4, (Smem) location. The tested bit is copied into status bit TCx an Smem.			e, k4, of	a memory		
Status	Bits	Affected by none					
		Affects TCx					
Repeat	t	This instruction can be repea	ated.				
See Al	SO	See the following other relate	ed instrue	ctions:			
		Clear Memory Bit					
		Complement Memory Bit	t				
		Set Memory Bit					
		Test and Clear Memory E	Bit				
		Test and Complement M	emory B	it			
		Test Memory Bit					

Syntax	Description
TC1 = bit(*AR3, #12), bit(*AR3, #12) = #1	The bit at the position defined by the unsigned 4-bit value (12) in the content addressed by AR3 is tested and the tested bit is copied into TC1. The selected bit (12) in the content addressed by AR3 is set to 1.

# Chapter 6

# **Instruction Opcodes in Sequential Order**

This chapter provides the opcode in sequential order for each TMS320C55x<sup>™</sup> DSP instruction syntax.

#### 

# 6.1 Instruction Set Opcodes

Table 6–1 lists the opcodes of the instruction set. See Table 6–2 (page 6-16) for a list of the symbols and abbreviations used in the instruction set opcode. See Table 1–1 (page 1-2) and Table 1–2 (page 1-6) for a list of the terms, symbols, and abbreviations used in the algebraic syntax.

Table 6–1. Instruction Set Opcodes

Opcode	Algebraic syntax
0000000E xCCCCCCC kkkkkkk	while (cond && (RPTC < k8)) repeat
0000001E xCCCCCCC xxxxxxxx	if (cond) return
0000010E xCCCCCCC LLLLLLL	if (cond) goto L8
0000011E LLLLLLL LLLLLLL	goto L16
0000100E LLLLLLL LLLLLLL	call L16
0000110E kkkkkkk kkkkkkk	repeat(k16)
0000111E 1111111 11111111	blockrepeat{}
0001000E DDSS0000 xxSHIFTW	ACy = ACy & (ACx <<< #SHIFTW)
0001000E DDSS0001 xxSHIFTW	ACy = ACy   (ACx <<< #SHIFTW)
0001000E DDSS0010 xxSHIFTW	ACy = ACy ^ (ACx <<< #SHIFTW)
0001000E DDSS0011 xxSHIFTW	ACy = ACy + (ACx << #SHIFTW)
0001000E DDSS0100 xxSHIFTW	ACy = ACy - (ACx << #SHIFTW)
0001000E DDSS0101 xxSHIFTW	ACy = ACx << #SHIFTW
0001000E DDSS0110 xxSHIFTW	ACy = ACx < <c #shiftw<="" td=""></c>
0001000E DDSS0111 xxSHIFTW	ACy = ACx <<< #SHIFTW
0001000E xxSS1000 xxddxxxx	Tx = exp(ACx)
0001000E DDSS1001 xxddxxxx	ACy = mant(ACx), Tx = -exp(ACx)
0001000E xxSS1010 SSddxxxt	Tx = count(ACx, ACy, TCx)
0001000E DDSS1100 SSDDnnnn	max_diff(ACx,ACy,ACz,ACw)
0001000E DDSS1101 SSDDxxxr	max_diff_dbl(ACx,ACy,ACz,ACw,TRNx)
0001000E DDSS1110 SSDDxxxx	min_diff(ACx,ACy,ACz,ACw)
0001000E DDSS1111 SSDDxxxr	min_diff_dbl(ACx,ACy,ACz,ACw,TRNx)
0001001E FSSScc00 FDDDxuxt	TCx = uns(src RELOP dst)
0001001E FSSScc01 FDDD0utt	TCx = TCy & uns(src RELOP dst)
0001001E FSSScc01 FDDDlutt	TCx = !TCy & uns(src RELOP dst)
0001001E FSSScc10 FDDD0utt	TCx = TCy   uns(src RELOP dst)
0001001E FSSScc10 FDDDlutt	TCx = !TCy   uns(src RELOP dst)
0001001E FSSSxx11 FDDD0xvv	dst = BitOut \\ src \\ BitIn

Opcode	Algebraic syntax
0001001E FSSSxx11 FDDD1xvv	dst = BitIn // src // BitOut
0001010E FSSSxxxx FDDD0000	mar(TAy + TAx)
0001010E FSSSxxxx FDDD0001	mar(TAy = TAx)
0001010E FSSSxxxx FDDD0010	mar(TAy – TAx)
0001010E PPPPPPP FDDD0100	mar(TAx + P8)
0001010E PPPPPPP FDDD0101	mar(TAx = P8)
0001010E PPPPPPP FDDD0110	mar(TAx – P8)
0001010E FSSSxxxx FDDD1000	mar(TAy + TAx)
0001010E FSSSxxxx FDDD1001	mar(TAy = TAx)
0001010E FSSSxxxx FDDD1010	mar(TAy – TAx)
0001010E PPPPPPP FDDD1100	mar(TAx + P8)
0001010E PPPPPPP FDDD1101	mar(TAx = P8)
0001010E PPPPPPP FDDD1110	mar(TAx – P8)
0001011E xxxxxkkk kkkk0000	DPH = k7
0001011E xxxkkkkk kkkk0011	PDP = k9
0001011E kkkkkkk kkkk0100	BK03 = k12
0001011E kkkkkkk kkkk0101	BK47 = k12
0001011E kkkkkkk kkkk0110	BKC = k12
0001011E kkkkkkk kkkk1000	CSR = k12
0001011E kkkkkkk kkkk1001	BRC0 = k12
0001011E kkkkkkk kkkk1010	BRC1 = k12
0001100E kkkkkkk FDDDFSSS	dst = src & k8
0001101E kkkkkkk FDDDFSSS	dst = src   k8
0001110E kkkkkkk FDDDFSSS	$dst = src \wedge k8$
0001111E KKKKKKKK SSDDxx0%	ACy = rnd(ACx * K8)
0001111E KKKKKKKK SSDDss1%	ACy = rnd(ACx + (Tx * K8))
0010000E	nop
0010001E FSSSFDDD	dst = src
0010010E FSSSFDDD	dst = dst + src
0010011E FSSSFDDD	dst = dst - src
0010100E FSSSFDDD	dst = dst & src
0010101E FSSSFDDD	dst = dst   src
0010110E FSSSFDDD	dst = dst ^ src

Table 6–1. Instruction Set Opcodes (Continued)

SPRU375G

Instruction Opcodes in Sequential Order

Opcode	Algebraic syntax
0010111E FSSSFDDD	dst = max(src, dst)
0011000E FSSSFDDD	dst = min(src, dst)
0011001E FSSSFDDD	dst =  src
0011010E FSSSFDDD	dst = -src
0011011E FSSSFDDD	dst = ~src
0011100E FSSSFDDD (Note: FSSS = src1, FDDD = src2)	push(src1, src2)
0011101E FSSSFDDD (Note: FSSS = dst1, FDDD = dst2)	dst1, dst2 = pop()
0011110E kkkkFDDD	dst = k4
0011111E kkkkFDDD	dst = -k4
0100000E kkkkFDDD	dst = dst + k4
0100001E kkkkFDDD	dst = dst - k4
0100010E 00SSFDDD	TAx = HI(ACx)
0100010E 01x0FDDD	dst = dst >> #1
0100010E 01x1FDDD	dst = dst << #1
0100010E 1000FDDD	TAx = SP
0100010E 1001FDDD	TAx = SSP
0100010E 1010FDDD	TAx = CDP
0100010E 1100FDDD	TAx = BRC0
0100010E 1101FDDD	TAx = BRC1
0100010E 1110FDDD	TAx = RPTC
0100011E kkkk0000	bit(ST0, k4) = #0
0100011E kkkk0001	bit(ST0, k4) = #1
0100011E kkkk0010	bit(ST1, k4) = #0
0100011E kkkk0011	bit(ST1, k4) = #1
0100011E kkkk0100	bit(ST2, k4) = #0
0100011E kkkk0101	bit(ST2, k4) = #1
0100011E kkkk0110	bit(ST3, k4) = #0
0100011E kkkk0111	bit(ST3, k4) = #1
0100100E xxxxx000	repeat(CSR)
0100100E FSSSx001	repeat(CSR), CSR += TAx
0100100E kkkkx010	repeat(CSR), CSR += k4

Table 6–1. Instruction Set Opcodes (Continued)

6-4 Instruction Opcodes in Sequential Order

Opcode	Algebraic syntax
0100100E kkkkx011	repeat(CSR), CSR -= k4
0100100E xxxxx100	return
0100100E xxxxx101	return_int
0100101E OLLLLLL	goto L7
0100101E 11111111	localrepeat{}
0100110E kkkkkkkk	repeat(k8)
0100111E KKKKKKKK	SP = SP + K8
0101000E FDDDx000	dst = dst <<< #1
0101000E FDDDx001	dst = dst >>> #1
0101000E FDDDx010	dst = pop()
0101000E xxDDx011	ACx = dbl(pop())
0101000E FSSSx110	push(src)
0101000E xxSSx111	dbl(push(ACx))
0101000E XDDD0100	xdst = popboth()
0101000E XSSS0101	pshboth(xsrc)
0101001E FSSS00DD	HI(ACx) = TAx
0101001E FSSS1000	SP = TAx
0101001E FSSS1001	SSP = TAx
0101001E FSSS1010	CDP = TAx
0101001E FSSS1100	CSR = TAx
0101001E FSSS1101	BRC1 = TAx
0101001E FSSS1110	BRC0 = TAx
0101010E DDSS000%	ACy = rnd(ACy +  ACx )
0101010E DDSS001%	ACy = rnd(ACy + (ACx * ACx))
0101010E DDSS010%	ACy = rnd(ACy - (ACx * ACx))
0101010E DDSS011%	ACy = rnd(ACy * ACx)
0101010E DDSS100%	ACy = rnd(ACx * ACx)
0101010E DDSS101%	ACy = rnd(ACx)
0101010E DDSS110%	ACy = saturate(rnd(ACx))
0101011E DDSSss0%	ACy = rnd(ACy + (ACx * Tx))
0101011E DDSSss1%	ACy = rnd(ACy - (ACx * Tx))
0101100E DDSSss0%	ACy = rnd(ACx * Tx)
0101100E DDSSss1%	ACy = rnd((ACy * Tx) + ACx)

Table 6–1. Instruction Set Opcodes (Continued)

SPRU375G

6-5

Opcode	Algebraic syntax
0101101E DDSSss00	ACy = ACy + (ACx << Tx)
0101101E DDSSss01	ACy = ACy - (ACx << Tx)
0101101E DDxxxx1t	ACx = sftc(ACx,TCx)
0101110E DDSSss00	ACy = ACx <<< Tx
0101110E DDSSss01	ACy = ACx << Tx
0101110E DDSSss10	ACy = ACx < <c td="" tx<=""></c>
0101111E 00kkkkkk	swap()
01100111 lccccccc	if (cond) goto I4
01101000 хССССССС РРРРРРРР РРРРРРР РРРРРРРР	if (cond) goto P24
01101001 xCCCCCCC PPPPPPPP PPPPPPP PPPPPPPP	if (cond) call P24
01101010 РРРРРРР РРРРРРР РРРРРРР	goto P24
01101100 РРРРРРР РРРРРРР РРРРРРР	call P24
01101101 xCCCCCCC LLLLLLL LLLLLLL	if (cond) goto L16
01101110 xCCCCCCC LLLLLLL LLLLLLL	if (cond) call L16
01101111 FSSSccxu KKKKKKKK LLLLLLL	compare (uns(src RELOP K8)) goto L8
01110000 KKKKKKKK KKKKKKKK SSDDSHFT	ACy = ACx + (K16 << #SHFT)
01110001 KKKKKKKK KKKKKKKK SSDDSHFT	ACy = ACx – (K16 << #SHFT)
01110010 kkkkkkk kkkkkkk SSDDSHFT	ACy = ACx & (k16 <<< #SHFT)
01110011 kkkkkkk kkkkkkk SSDDSHFT	ACy = ACx   (k16 <<< #SHFT)
01110100 kkkkkkk kkkkkkk SSDDSHFT	ACy = ACx ^ (k16 <<< #SHFT)
01110101 KKKKKKKK KKKKKKKK xxDDSHFT	ACx = K16 << #SHFT
01110110 kkkkkkk kkkkkkk FDDD00SS	dst = field_extract(ACx,k16)
01110110 kkkkkkk kkkkkkk FDDD01SS	dst = field_expand(ACx,k16)
01110110 KKKKKKKK KKKKKKKK FDDD10 $ imes$ x	dst = K16
01110111 DDDDDDDD DDDDDDDD FDDDxxxx	mar(TAx = D16)
01111000 kkkkkkk kkkkkkk xxx0000x	DP = k16
01111000 kkkkkkk kkkkkkk xxx0001x	SSP = k16
01111000 kkkkkkk kkkkkkk xxx0010x	CDP = k16
01111000 kkkkkkk kkkkkkkk xxx0011x	BSA01 = k16
01111000 kkkkkkk kkkkkkkk xxx0100x	BSA23 = k16
01111000 kkkkkkk kkkkkkk xxx0101x	BSA45 = k16

Table 6–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
01111000 kkkkkkk kkkkkkkk xxx011	0x BSA67 = k16
01111000 kkkkkkk kkkkkkkk xxx011	1x BSAC = k16
01111000 kkkkkkk kkkkkkkk xxx100	0x SP = k16
01111001 KKKKKKKK KKKKKKKK SSDDxx	ACy = rnd(ACx * K16)
01111001 KKKKKKKK KKKKKKKK SSDDss	1% ACy = rnd(ACx + (Tx * K16))
01111010 KKKKKKKK KKKKKKKK SSDD00	0x ACy = ACx + (K16 << #16)
01111010 KKKKKKKK KKKKKKKK SSDD00	1x ACy = ACx - (K16 << #16)
01111010 kkkkkkk kkkkkkkk SSDD01	0x ACy = ACx & (k16 <<< #16)
01111010 kkkkkkk kkkkkkkk SSDD01	1x ACy = ACx   (k16 <<< #16)
01111010 kkkkkkk kkkkkkkk SSDD10	0x ACy = ACx ^ (k16 <<< #16)
01111010 KKKKKKKK KKKKKKKK xxDD10	1x ACx = K16 << #16
01111010 xxxxxxx xxxxxxx xxxx11	0x idle
01111011 KKKKKKKK KKKKKKKK FDDDFS	dst = src + K16
01111100 KKKKKKKK KKKKKKKK FDDDFS	dst = src – K16
01111101 kkkkkkk kkkkkkk FDDDFS	dst = src & k16
01111110 kkkkkkk kkkkkkk FDDDFS	ss dst = src   k16
01111111 kkkkkkk kkkkkkk FDDDFS	dst = src $^{1}$ k16
10000000 XXXMMMYY YMMM00xx	dbl(Ymem) = dbl(Xmem)
10000000 XXXMMMYY YMMM01xx	Ymem = Xmem
10000000 XXXMMMYY YMMM10SS	Xmem = LO(ACx), Ymem = HI(ACx)
10000001 XXXMMMYY YMMM00DD	ACx = (Xmem << #16) + (Ymem << #16)
10000001 XXXMMMYY YMMM01DD	ACx = (Xmem << #16) – (Ymem << #16)
10000001 XXXMMMYY YMMM10DD	LO(ACx) = Xmem, HI(ACx) = Ymem
10000010 XXXMMMYY YMMM00mm uuDDDD	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))
10000010 XXXMMMYY YMMM01mm uuDDDD	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))
10000010 XXXMMMYY YMMM10mm uuDDDD	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))
10000010 XXXMMMYY YMMM11mm uuxxDD	ng% mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(coef(Cmem))))

Table 6–1. Instruction Set Opcodes (Continued)

SPRU375G

	Ор	code		Algebraic syntax
10000011	XXXMMMYY	YMMM00mm	uuDDDDg%	ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))
10000011	XXXMMMYY	YMMM01mm	uuDDDDg%	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))
10000011	XXXMMMYY	YMMM10mm	uuDDDDg%	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))
10000011	XXXMMMYY	YMMM11mm	uuxxDDg%	mar(Xmem), ACx = <mark>M40(rnd(</mark> ACx + ( <mark>uns(</mark> Ymem) * <mark>uns(</mark> coef(Cmem)))))
10000100	XXXMMMYY	YMMM00mm	uuDDDDg%	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
10000100	XXXMMMYY	YMMM01mm	uuxxDDg%	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
10000100	XXXMMMYY	YMMM10mm	uuDDDDg%	ACx = M40(rnd(uns(Xmem) * uns(coef(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
10000100	XXXMMMYY	YMMM11mm	uuDDDDg%	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))
10000101	XXXMMMYY	YMMM00mm	uuxxDDg%	mar(Xmem), ACx = <mark>M40(rnd(</mark> ACx – ( <mark>uns(</mark> Ymem) * <mark>uns(</mark> coef(Cmem)))))
10000101	XXXMMMYY	YMMM01mm	uuDDDDg%	ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy - (uns(Ymem) * uns(coef(Cmem)))))
10000101	XXXMMMYY	YMMM10mm	xxxxxxx	mar(Xmem) ,mar(Ymem) ,mar(coef(Cmem))
10000101	XXXMMMYY	YMMM11mm	DDx0DDU%	firs(Xmem, Ymem, coef(Cmem), ACx, ACy)
10000101	XXXMMMYY	YMMM11mm	DDx1DDU%	firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)
10000110	XXXMMMYY	YMMMxxDD	000guuU%	ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [,T3 = Xmem]
10000110	XXXMMMYY	YMMMSSDD	001guuU%	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]
10000110	XXXMMMYY	YMMMSSDD	010guuU%	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]
10000110	XXXMMMYY	YMMMSSDD	011guuU%	ACy = M40(rnd(ACx - (uns(Xmem) * uns(Ymem)))) [,T3 = Xmem]

Table 6–1. Instruction Set Opcodes (Continued)

Instruction Opcodes in Sequential Order

6-8

	Ор	code		Algebraic syntax
10000110	XXXMMMYY	YMMMDDDD	100xssU%	ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]
10000110	XXXMMMYY	YMMMDDDD	101xssU%	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]
10000110	XXXMMMYY	YMMMDDDD	110xxxx%	lms(Xmem, Ymem, ACx, ACy)
10000110	XXXMMMYY	YMMMDDDD	1110xxn%	sqdst(Xmem, Ymem, ACx, ACy)
10000110	XXXMMMYY	YMMMDDDD	1111xxn%	abdst(Xmem, Ymem, ACx, ACy)
10000111	XXXMMMYY	YMMMSSDD	000xssU%	ACy = <mark>rnd(</mark> Tx * Xmem), Ymem = HI(ACx << T2) [, <mark>T3 = Xmem]</mark>
10000111	XXXMMMYY	YMMMSSDD	001xssU%	ACy = <mark>rnd(</mark> ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [, <mark>T3 = Xmem]</mark>
10000111	XXXMMMYY	YMMMSSDD	010xssU%	ACy = <mark>rnd(</mark> ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [, <mark>T3 = Xmem]</mark>
10000111	XXXMMMYY	YMMMSSDD	100xxxxx	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)
10000111	XXXMMMYY	YMMMSSDD	101xxxxx	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)
10000111	XXXMMMYY	YMMMSSDD	110xxxxx	ACy = Xmem << #16, Ymem = HI(ACx << T2)
10010000	XSSSXDDD			xdst = xsrc
10010001	xxxxxSS			goto ACx
10010010	xxxxxSS			call ACx
10010100	xxxxxxx			reset
10010101	0xxkkkkk			intr(k5)
10010101	1xxkkkkk			trap(k5)
10010110	0CCCCCCC			if (cond) execute(AD_unit)
10010110	1CCCCCCC			if (cond) execute(D_unit)
10011000				mmap()
10011001				readport()
10011010				writeport()
10011100				linear()
10011101				circular()
10011110	000000000			if (cond) execute(AD_unit)
10011110	1CCCCCCC			if (cond) execute(D_unit)
10011111	0CCCCCCC			if (cond) execute(AD_unit)

Table 6–1. Instruction Set Opcodes (Continued)

Instruction Opcodes in Sequential Order

Opcode	Algebraic syntax
10011111 1CCCCCCC	if (cond) execute(D_unit)
1010FDDD AAAAAAAI	dst = Smem
101100DD AAAAAAAI	ACx = Smem << #16
10110100 AAAAAAI	mar(Smem)
10110101 AAAAAAAI	push(Smem)
10110110 AAAAAAI	delay(Smem)
10110111 AAAAAAI	push(dbl(Lmem))
10111000 AAAAAAI	dbl(Lmem) = pop()
10111011 AAAAAAI	Smem = pop()
101111SS AAAAAAAI	Smem = HI(ACx)
1100FSSS AAAAAAAI	Smem = src
11010000 AAAAAAAI U%DDxxmm	ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem], delay(Smem)
11010001 AAAAAAAI U%DD00mm	ACx = rnd(Smem * coef(Cmem)) [,T3 = Smem]
11010001 AAAAAAAI U%DD01mm	ACx = rnd(ACx + (Smem * coef(Cmem))) [,T3 = Smem]
11010001 AAAAAAAI U%DD10mm	ACx = rnd(ACx - (Smem * coef(Cmem))) [,T3 = Smem]
11010010 AAAAAAAI U%DD00SS	ACy = rnd(ACy + (Smem * ACx)) [,T3 = Smem]
11010010 AAAAAAAI U%DD01SS	ACy = rnd(ACy – (Smem * ACx)) [,T3 = Smem]
11010010 AAAAAAAI U%DD10SS	ACy = rnd(ACx + (Smem * Smem)) [,T3 = Smem]
11010010 AAAAAAAI U%DD11SS	ACy = rnd(ACx – (Smem * Smem)) [,T3 = Smem]
11010011 AAAAAAAI U%DD00SS	ACy = rnd(Smem * ACx) [,T3 = Smem]
11010011 AAAAAAAI U%DD10xx	ACx = rnd(Smem * Smem) [,T3 = Smem]
11010011 AAAAAAAI U%DDulss	ACx = rnd(uns(Tx * Smem)) [,T3 = Smem]
11010100 AAAAAAAI U%DDssSS	ACy = rnd(ACx + (Tx * Smem)) [,T3 = Smem]
11010101 AAAAAAAI U%DDssSS	ACy = rnd(ACx - (Tx * Smem)) [,T3 = Smem]
11010110 AAAAAAAI FDDDFSSS	dst = src + Smem
11010111 AAAAAAAI FDDDFSSS	dst = src – Smem
11011000 AAAAAAAI FDDDFSSS	dst = Smem – src
11011001 AAAAAAAI FDDDFSSS	dst = src & Smem
11011010 AAAAAAAI FDDDFSSS	dst = src   Smem
11011011 AAAAAAAI FDDDFSSS	dst = src ^ Smem
11011100 AAAAAAAI kkkkxx00	TC1 = bit(Smem, k4)
11011100 AAAAAAAI kkkkxx01	TC2 = bit(Smem, k4)

Table 6–1. Instruction Set Opcodes (Continued)

6-10 Instruction Opcodes in Sequential Order

Opcode	Algebraic syntax
11011100 AAAAAAI 0000xx10	DP = Smem
11011100 AAAAAAI 0001xx10	CDP = Smem
11011100 AAAAAAI 0010xx10	BSA01 = Smem
11011100 AAAAAAI 0011xx10	BSA23 = Smem
11011100 AAAAAAI 0100xx10	BSA45 = Smem
11011100 AAAAAAI 0101xx10	BSA67 = Smem
11011100 AAAAAAAI 0110xx10	BSAC = Smem
11011100 AAAAAAAI 0111xx10	SP = Smem
11011100 AAAAAAAI 1000xx10	SSP = Smem
11011100 AAAAAAAI 1001xx10	BK03 = Smem
11011100 AAAAAAAI 1010xx10	BK47 = Smem
11011100 AAAAAAAI 1011xx10	BKC = Smem
11011100 AAAAAAI 1100xx10	DPH = Smem
11011100 AAAAAAAI 1111xx10	PDP = Smem
11011100 AAAAAAAI x000xx11	CSR = Smem
11011100 AAAAAAAI x001xx11	BRC0 = Smem
11011100 AAAAAAAI x010xx11	BRC1 = Smem
11011100 AAAAAAAI x011xx11	TRN0 = Smem
11011100 AAAAAAAI x100xx11	TRN1 = Smem
11011101 AAAAAAAI SSDDss00	ACy = ACx + (Smem << Tx)
11011101 AAAAAAAI SSDDss01	ACy = ACx - (Smem << Tx)
11011101 AAAAAAAI SSDDss10	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)
11011101 AAAAAAAI x%DDssll	ACx = rnd(Smem << Tx)
11011110 AAAAAAAI SSDD0000	ACy = adsc(Smem, ACx, TC1)
11011110 AAAAAAAI SSDD0001	ACy = adsc(Smem, ACx, TC2)
11011110 AAAAAAAI SSDD0010	ACy = adsc(Smem, ACx, TC1, TC2)
11011110 AAAAAAAI SSDD0011	subc(Smem, ACx, ACy)
11011110 AAAAAAAI SSDD0100	ACy = ACx + (Smem << #16)
11011110 AAAAAAAI SSDD0101	ACy = ACx - (Smem << #16)
11011110 AAAAAAAI SSDD0110	ACy = (Smem << #16) – ACx
11011110 AAAAAAAI ssDD1000	HI(ACx) = Smem + Tx, LO(ACx) = Smem - Tx

Table 6–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11011110 AAAAAAAI ssDD1001	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx
11011111 AAAAAAAI FDDD000u	dst = <mark>uns(</mark> high_byte(Smem))
11011111 AAAAAAAI FDDD001u	dst = <mark>uns(</mark> low_byte(Smem))
11011111 AAAAAAAI xxDD010u	ACx = uns(Smem)
11011111 AAAAAAAI SSDD100u	ACy = ACx + uns(Smem) + CARRY
11011111 AAAAAAAI SSDD101u	ACy = ACx - uns(Smem) - BORROW
11011111 AAAAAAAI SSDD110u	ACy = ACx + uns(Smem)
11011111 AAAAAAAI SSDD111u	ACy = ACx - uns(Smem)
11100000 AAAAAAAI FSSSxxxt	TCx = bit(Smem, src)
11100001 AAAAAAAI DDSHIFTW	ACx = low_byte(Smem) << #SHIFTW
11100010 AAAAAAAI DDSHIFTW	ACx = high_byte(Smem) << #SHIFTW
11100011 AAAAAAAI kkkk000x	TC1 = bit(Smem, k4), bit(Smem, k4) = #1
11100011 AAAAAAAI kkkk001x	TC2 = bit(Smem, k4), bit(Smem, k4) = #1
11100011 AAAAAAAI kkkk010x	TC1 = bit(Smem, k4), bit(Smem, k4) = #0
11100011 AAAAAAAI kkkk011x	TC2 = bit(Smem, k4), bit(Smem, k4) = #0
11100011 AAAAAAAI kkkkl00x	TC1 = bit(Smem, k4), cbit(Smem, k4)
11100011 AAAAAAAI kkkkl01x	TC2 = bit(Smem, k4), cbit(Smem, k4)
11100011 AAAAAAAI FSSS1100	bit(Smem, src) = #1
11100011 AAAAAAAI FSSS1101	bit(Smem, src) = #0
11100011 AAAAAAAI FSSS111x	cbit(Smem, src)
11100100 AAAAAAAI FSSSx0xx	push(src, Smem)
11100100 AAAAAAAI FDDDx1xx	dst, Smem = pop()
11100101 AAAAAAAI FSSS01x0	high_byte(Smem) = src
11100101 AAAAAAAI FSSS01x1	low_byte(Smem) = src
11100101 AAAAAAAI 000010xx	Smem = DP
11100101 AAAAAAAI 000110xx	Smem = CDP
11100101 AAAAAAAI 001010xx	Smem = BSA01
11100101 AAAAAAAI 001110xx	Smem = BSA23
11100101 AAAAAAA 010010xx	Smem = BSA45
11100101 AAAAAAAI 010110xx	Smem = BSA67
11100101 AAAAAAA 011010xx	Smem = BSAC
11100101 AAAAAAAI 011110xx	Smem = SP

Table 6–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11100101 AAAAAAAI 100010xx	Smem = SSP
11100101 AAAAAAI 100110xx	Smem = BK03
11100101 AAAAAAI 101010xx	Smem = BK47
11100101 AAAAAAI 101110xx	Smem = BKC
11100101 AAAAAAI 110010xx	Smem = DPH
11100101 AAAAAAI 111110xx	Smem = PDP
11100101 AAAAAAAI x00011xx	Smem = CSR
11100101 AAAAAAAI x00111xx	Smem = BRC0
11100101 AAAAAAAI x01011xx	Smem = BRC1
11100101 AAAAAAAI x01111xx	Smem = TRN0
11100101 AAAAAAAI x10011xx	Smem = TRN1
11100110 АААААААІ КККККККК	Smem = K8
11100111 AAAAAAAI SSss00xx	Smem = LO(ACx << Tx)
11100111 AAAAAAAI SSssl0x%	Smem = HI(rnd(ACx << Tx))
11100111 AAAAAAAI SSssllu%	Smem = HI(saturate(uns(rnd(ACx << Tx))))
11101000 AAAAAAAI SSxxx0x%	Smem = HI(rnd(ACx))
11101000 AAAAAAAI SSxxxlu%	Smem = HI(saturate(uns(rnd(ACx))))
11101001 AAAAAAAI SSSHIFTW	Smem = LO(ACx << #SHIFTW)
11101010 AAAAAAAI SSSHIFTW	Smem = HI(ACx << #SHIFTW)
11101011 AAAAAAAI xxxx01xx	dbl(Lmem) = RETA
11101011 AAAAAAAI xxSS10x0	dbl(Lmem) = ACx
11101011 AAAAAAAI xxSS10ul	dbl(Lmem) = saturate <mark>(uns</mark> (ACx))
11101011 AAAAAAAI FSSS1100	Lmem = pair(TAx)
11101011 AAAAAAAI xxSS1101	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1
11101011 AAAAAAAI xxSS1110	Lmem = pair(HI(ACx))
11101011 AAAAAAAI xxSS1111	Lmem = pair(LO(ACx))
11101100 AAAAAAAI FSSS000x	bit(src, Baddr) = #1
11101100 AAAAAAAI FSSS001x	bit(src, Baddr) = #0
11101100 AAAAAAAI FSSS010x	bit(src, pair(Baddr))
11101100 AAAAAAAI FSSS011x	cbit(src, Baddr)
11101100 AAAAAAAI FSSS100t	TCx = bit(src, Baddr)
11101100 AAAAAAAI XDDD1110	XAdst = mar(Smem)

Table 6–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11101101 AAAAAAAI SSDD000n	ACy = ACx + dbl(Lmem)
11101101 AAAAAAAI SSDD001n	ACy = ACx - dbl(Lmem)
11101101 AAAAAAAI SSDD010x	ACy = dbl(Lmem) - ACx
11101101 AAAAAAAI xxxx011x	RETA = dbl(Lmem)
11101101 AAAAAAAI xxDD100g	ACx = M40(dbl(Lmem))
11101101 AAAAAAAI xxDD101x	pair(HI(ACx)) = Lmem
11101101 AAAAAAAI xxDD110x	pair(LO(ACx)) = Lmem
11101101 AAAAAAAI FDDD111x	pair(TAx) = Lmem
11101101 AAAAAAAI XDDD1111	XAdst = dbl(Lmem)
11101101 AAAAAAAI XSSS0101	dbl(Lmem) = XAsrc
11101110 AAAAAAAI SSDD000x	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)
11101110 AAAAAAAI SSDD001x	HI(ACy) = HI(ACx) - HI(Lmem), LO(ACy) = LO(ACx) - LO(Lmem)
11101110 AAAAAAAI SSDD010x	HI(ACy) = HI(Lmem) - HI(ACx), LO(ACy) = LO(Lmem) - LO(ACx)
11101110 AAAAAAAI ssdd011x	HI(ACx) = Tx - HI(Lmem), LO(ACx) = Tx - LO(Lmem)
11101110 AAAAAAAI ssDD100x	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx
11101110 AAAAAAAI ssDDl01x	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) - Tx
11101110 AAAAAAAI ssDD110x	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) - Tx
11101110 AAAAAAAI ssDD111x	HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) + Tx
11101111 AAAAAAAI xxxx00mm	Smem = coef(Cmem)
11101111 AAAAAAAI xxxx01mm	coef(Cmem) = Smem
11101111 AAAAAAAI xxxx10mm	Lmem = dbl(coef(Cmem))
11101111 AAAAAAAI xxxx11mm	dbl(coef(Cmem)) = Lmem
11110000 аааааааг кккккккк ккккккк	TC1 = (Smem == K16)
11110001 АААААААІ КККККККК ККККККК	TC2 = (Smem == K16)
11110010 AAAAAAAI kkkkkkk kkkkkkk	TC1 = Smem & k16
11110011 AAAAAAAI kkkkkkkk kkkkkkk	TC2 = Smem & k16
11110100 AAAAAAAI kkkkkkk kkkkkkk	Smem = Smem & k16

Table 6–1. Instruction Set Opcodes (Continued)

Table 6–1. Instruction Set Opcodes (Continued)

Opcode	Algebraic syntax
11110101 AAAAAAAI kkkkkkkk kkkkkkk	Smem = Smem   k16
11110110 AAAAAAAI kkkkkkk kkkkkkk	Smem = Smem ^ k16
11110111 АААААААІ КККККККК ККККККК	Smem = Smem + K16
11111000 AAAAAAAI KKKKKKKK xxDDx0U%	ACx = rnd(Smem * K8) [,T3 = Smem]
11111000 AAAAAAAI KKKKKKKK SSDDx1U%	ACy = rnd(ACx + (Smem * K8)) [,T3 = Smem]
11111001 AAAAAAAI uxSHIFTW SSDD00xx	ACy = ACx + (uns(Smem) << #SHIFTW)
11111001 AAAAAAAI uxSHIFTW SSDD01xx	ACy = ACx - ( <mark>uns(</mark> Smem) << #SHIFTW)
11111001 AAAAAAAI uxSHIFTW xxDD10xx	ACx = <mark>uns(</mark> Smem) << #SHIFTW
11111010 AAAAAAAI xxSHIFTW SSxxx0x%	Smem = HI <mark>(rnd(</mark> ACx << #SHIFTW))
11111010 AAAAAAAI uxSHIFTW SSxxxlx%	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))
11111011 АААААААІ КККККККК ККККККК	Smem = K16
11111100 AAAAAAAI LLLLLLL LLLLLLL	if (ARn_mod != #0) goto L16

### 6.2 Instruction Set Opcode Symbols and Abbreviations

Table 6–2 lists the symbols and abbreviations used in the instruction set opcode.

Bit Field Name	Bit Field Value	Bit Field Description
olo	0	Rounding is disabled
	1	Rounding is enabled
AAAA AAAI		Smem addressing mode:
	AAAA AAA0	@dma, direct memory address (dma) direct access
	AAAA AAA1	Smem indirect memory access:
	0001 0001	ABS16(#k16)
	0011 0001	*(#k23)
	0101 0001	*port(#k16)
	0111 0001	*CDP
	1001 0001	*CDP+
	1011 0001	*CDP-
	1101 0001	*CDP(#K16)
	1111 0001	*+CDP(#K16)
	PPP0 0001	*ARn
	PPP0 0011	*ARn+
	PPP0 0101	*ARn-
	PPP0 0111	*(ARn + T0), when C54CM = 0 *(ARn + T0), when C54CM = 1
	PPP0 1001	*(ARn – T0), when C54CM = 0 *(ARn – T0), when C54CM = 1
	PPP0 1011	*ARn(T0), when C54CM = 0 *ARn(T0), when C54CM = 1
	PPP0 1101	*ARn(#K16)
	PPP0 1111	*+ARn(#K16)
	PPP1 0011	*(ARn + T1), when ARMS = 0 *ARn(short(#1)), when ARMS = 1
	PPP1 0101	*(ARn – T1), when ARMS = 0 *ARn(short(#2)), when ARMS = 1

Table 6–2. Instruction Set Opcode Symbols and Abbreviations

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field D	Description
	PPP1 0111		when ARMS = 0 :(#3)), when ARMS = 1
	PPP1 1001		en ARMS = 0 :(#4)), when ARMS = 1
	PPP1 1011	*–ARn, when ARMS = 0 *ARn(short(#5)), when ARMS = 1	
	PPP1 1101	•	0B), when ARMS = 0 ;(#6)), when ARMS = 1
	PPP1 1111		0B), when ARMS = 0 :(#7)), when ARMS = 1
	PPP encode	s an auxiliary	register (ARn) as for XXX and YYY.
CC		Relational	operators (RELOP):
	00	==	(equal to)
	01	<	(less than)
	10	>=	(greater than or equal to)
	11	!=	(not equal to)
CCC CCCC			l field (cond) on source accumulator, auxiliary, or temporary Cx; and CARRY:
	000 FSSS	src == 0	(source is equal to 0)
	001 FSSS	src != 0	(source is not equal to 0)
	010 FSSS	src < 0	(source is less than 0)
	011 FSSS	src <= 0	(source is less than or equal to 0)
	100 FSSS	src > 0	(source is greater than 0)
	101 FSSS	src >= 0	(source is greater than or equal to 0)
	110 00SS	overflow(A	Cx) (source accumulator overflow status bit (ACOVx) is tested against 1)
	110 0100	TC1	(status bit is tested against 1)
	110 0101	TC2	(status bit is tested against 1)
	110 0110	CARRY	(status bit is tested against 1)
	110 0111	Reserved	

Bit Field Name	Bit Field Value	Bit Field De	escription
	110 1000	TC1 & TC2	
	110 1001	TC1 & !TC2	
	110 1010	!TC1 & TC2	
	110 1011	!TC1 & !TC2	2
	110 11xx	Reserved	
	111 00SS	loverflow(A0	Cx) (source accumulator overflow status bit (ACOVx) is tested against 0)
	111 0100	!TC1	(status bit is tested against 0)
	111 0101	!TC2	(status bit is tested against 0)
	111 0110	!CARRY	(status bit is tested against 0)
	111 0111	Reserved	
	111 1000	TC1   TC2	
	111 1001	TC1   !TC2	
	111 1010	!TC1   TC2	
	111 1011	!TC1   !TC2	
	111 1100	TC1 ^ TC2	
	111 1101	TC1 ^ !TC2	
	111 1110	!TC1 ^ TC2	
	111 1111	!TC1 ^ !TC2	2
dd		Destination	temporary register (Tx, Ty):
	00	Temporary r	register 0 (T0)
	01	Temporary r	register 1 (T1)
	10	Temporary r	register 2 (T2)
	11	Temporary r	register 3 (T3)

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
DD		Destination accumulator register (ACw, ACx, ACy, ACz):
	00	Accumulator 0 (AC0)
	01	Accumulator 1 (AC1)
	10	Accumulator 2 (AC2)
	11	Accumulator 3 (AC3)
DDD D		Data address label coded on n bits (absolute address)
E	0	Parallel Enable bit is cleared to 0
	1	Parallel Enable bit is set to 1
FDDD FSSS		Destination or Source accumulator, auxiliary, or temporary register (dst, src, TAx, TAy):
	0000	Accumulator 0 (AC0)
	0001	Accumulator 1 (AC1)
	0010	Accumulator 2 (AC2)
	0011	Accumulator 3 (AC3)
	0100	Temporary register 0 (T0)
	0101	Temporary register 1 (T1)
	0110	Temporary register 2 (T2)
	0111	Temporary register 3 (T3)
	1000	Auxiliary register 0 (AR0)
	1001	Auxiliary register 1 (AR1)
	1010	Auxiliary register 2 (AR2)
	1011	Auxiliary register 3 (AR3)
	1100	Auxiliary register 4 (AR4)
	1101	Auxiliary register 5 (AR5)
	1110	Auxiliary register 6 (AR6)
	1111	Auxiliary register 7 (AR7)

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
g	0	40 keyword is not applied
	1	40 keyword is applied; M40 is locally set to 1
kk kkkk		Swap code for Swap Register Content instruction:
	00 0000	swap(AC0, AC2)
	00 0001	swap(AC1, AC3)
	00 0100	swap(T0, T2)
	00 0101	swap(T1, T3)
	00 1000	swap(AR0, AR2)
	00 1001	swap(AR1, AR3)
	00 1100	swap(AR4, T0)
	00 1101	swap(AR5, T1)
	00 1110	swap(AR6, T2)
	00 1111	swap(AR7, T3)
	01 0000	swap(pair(AC0), pair(AC2))
	01 0001	Reserved
	01 0100	swap(pair(T0), pair(T2))
	01 0101	Reserved
	01 1000	swap(pair(AR0), pair(AR2))
	01 1001	Reserved
	01 1100	swap(pair(AR4), pair(T0))
	01 1101	Reserved
	01 1110	swap(pair(AR6), pair(T2))
	01 1111	Reserved
	10 1000	Reserved
	10 1100	swap(block(AR4), block(T0))
	11 1000	swap(AR0, AR1)
	11 1100	Reserved
	1x 0000	Reserved
	1x 0001	Reserved

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	1x 0100	Reserved
	1x 0101	Reserved
	1x 1001	Reserved
	1x 1101	Reserved
	1x 1110	Reserved
	1x 1111	Reserved
kkk k		Unsigned constant of n bits
ккк к		Signed constant of n bits
111 1		Program address label coded on n bits (unsigned offset relative to program counter register)
LLL L		Program address label coded on n bits (signed offset relative to program counter register)
mm		Coefficient addressing mode (Cmem):
	00	*CDP
	01	*CDP+
	10	*CDP-
	11	*(CDP + T0)
MMM		Modifier option for Xmem or Ymem addressing mode:
	000	*ARn
	001	*ARn+
	010	*ARn-
	011	*(ARn + T0), when C54CM = 0 *(ARn + AR0), when C54CM = 1
	100	*(ARn + T1)
	101	*(ARn – T0), when C54CM = 0 *(ARn – AR0), when C54CM = 1

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	110	*(ARn – T1)
	111	*ARn(T0), when C54CM = 0 *ARn(AR0), when C54CM = 1
n		Reserved bit
PPP	P	Program or data address label coded on n bits (absolute address)
r	0	Select TRN0
	1	Select TRN1
SHFT		4-bit immediate shift value, 0 to 15
SHIFTW		6-bit immediate shift value, -32 to +31
SS		Source temporary register (Tx, Ty):
	00	Temporary register 0 (T0)
	01	Temporary register 1 (T1)
	10	Temporary register 2 (T2)
	11	Temporary register 3 (T3)
SS		Source accumulator register (ACw, ACx, ACy, ACz):
	00	Accumulator 0 (AC0)
	01	Accumulator 1 (AC1)
	10	Accumulator 2 (AC2)
	11	Accumulator 3 (AC3)

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
tt	00	Bit 0: destination TCy bit of Compare Register Content instruction
	01	Bit 1: source TCx bit of Compare Register Content instruction
	10	When value = 0: TC1 is selected
	11	When value = 1: TC2 is selected
u	0	uns keyword is not applied; operand is considered signed
	1	uns keyword is applied; operand is considered unsigned
U	0	No update of T3 with Smem or Xmem content
	1	T3 is updated with Smem or Xmem content
vv	00	Bit 0: shifted-out bit of Rotate instruction
	01	Bit 1: shifted-in bit of Rotate instruction
	10	When value = 0: CARRY is selected
	11	When value = 1: TC2 is selected
x		Reserved bit
XDDD XSSS		Destination or Source accumulator or extended register. All 23 bits of stack pointer (XSP), system stack pointer (XSSP), data page pointer (XDP), coefficient data pointer (XCDP), and extended auxiliary register (XARx).
	0000	Accumulator 0 (AC0)
	0001	Accumulator 1 (AC1)
	0010	Accumulator 2 (AC2)
	0011	Accumulator 3 (AC3)
	0100	Stack pointer (XSP)
	0101	System stack pointer (XSSP)
	0110	Data page pointer (XDP)
	0111	Coefficient data pointer (XCDP)
	1000	Auxiliary register 0 (XAR0)
	1001	Auxiliary register 1 (XAR1)

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	1010	Auxiliary register 2 (XAR2)
	1011	Auxiliary register 3 (XAR3)
	1100	Auxiliary register 4 (XAR4)
	1101	Auxiliary register 5 (XAR5)
	1110	Auxiliary register 6 (XAR6)
	1111	Auxiliary register 7 (XAR7)
XXX YYY		Auxiliary register designation for Xmem or Ymem addressing mode:
	000	Auxiliary register 0 (AR0)
	001	Auxiliary register 1 (AR1)
	010	Auxiliary register 2 (AR2)
	011	Auxiliary register 3 (AR3)
	100	Auxiliary register 4 (AR4)
	101	Auxiliary register 5 (AR5)
	110	Auxiliary register 6 (AR6)
	111	Auxiliary register 7 (AR7)

Table 6–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

## Chapter 7

# **Cross-Reference of Algebraic and Mnemonic Instruction Sets**

This chapter provides a cross-reference between the TMS320C55x<sup>TM</sup> DSP algebraic instruction set and the mnemonic instruction set (Table 7–1). For more information on the mnemonic instruction set, see *TMS320C55x* DSP *Mnemonic Instruction Set Reference Guide*, SPRU374.

Algebraic Syntax	Mnemonic Syntax
Absolute Distance	ABDST: Absolute Distance
abdst(Xmem, Ymem, ACx, ACy)	ABDST Xmem, Ymem, ACx, ACy
Absolute Value	ABS: Absolute Value
dst =  src	ABS [src,] dst
Addition	ADD: Addition
dst = dst + src	ADD [src,] dst
dst = dst + k4	ADD k4, dst
dst = src + K16	ADD K16, [src,] dst
dst = src + Smem	ADD Smem, [src,] dst
ACy = ACy + (ACx << Tx)	ADD ACx << Tx, ACy
ACy = ACy + (ACx << #SHIFTW)	ADD ACx << #SHIFTW, ACy
ACy = ACx + (K16 << #16)	ADD K16 << #16, <mark>[ACx,]</mark> ACy
ACy = ACx + (K16 << #SHFT)	ADD K16 << #SHFT, [ACx,] ACy
ACy = ACx + (Smem << Tx)	ADD Smem << Tx, <mark>[ACx,]</mark> ACy
ACy = ACx + (Smem << #16)	ADD Smem << #16, [ACx,] ACy
ACy = ACx + <mark>uns(</mark> Smem) + CARRY	ADD [uns(]Smem[)], CARRY, [ACx,] ACy
ACy = ACx + uns(Smem)	ADD [uns(]Smem[)], [ACx,] ACy
ACy = ACx + ( <mark>uns(</mark> Smem <mark>)</mark> << #SHIFTW)	ADD [uns(]Smem[)] << #SHIFTW, [ACx,] ACy
ACy = ACx + dbl(Lmem)	ADD dbl(Lmem), [ACx,] ACy
ACx = (Xmem << #16) + (Ymem << #16)	ADD Xmem, Ymem, ACx
Smem = Smem + K16	ADD K16, Smem

Cross-Reference of Algebraic and Mnemonic Instruction Sets

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)
---

Algebraic Syntax	Mnemonic Syntax
Addition with Absolute Value	ADDV: Addition with Absolute Value
ACy = rnd(ACy +  ACx )	ADD[R]V [ACx,] ACy
Addition with Parallel Store Accumulator Content to Memory	ADD::MOV: Addition with Parallel Store Accumulator Content to Memory
ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)	ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem
Addition or Subtraction Conditionally	ADDSUBCC: Addition or Subtraction Conditionally
ACy = adsc(Smem, ACx, TCx)	ADDSUBCC Smem, ACx, TCx, ACy
Addition or Subtraction Conditionally with Shift	ADDSUB2CC: Addition or Subtraction Conditionally with Shift
ACy = ads2c(Smem, ACx, Tx, TC1, TC2)	ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy
Addition, Subtraction, or Move Accumulator Content Conditionally	ADDSUBCC: Addition, Subtraction, or Move Accumulator Content Conditionally
ACy = adsc(Smem, ACx, TC1, TC2)	ADDSUBCC Smem, ACx, TC1, TC2, ACy
Bitwise AND	AND: Bitwise AND
dst = dst & src	AND src, dst
dst = src & k8	AND k8,src, dst
dst = src & k16	AND k16, src, dst
dst = src & Smem	AND Smem, src, dst
ACy = ACy & (ACx <<< #SHIFTW)	AND ACx << #SHIFTW[, ACy]
ACy = ACx & (k16 <<< #16)	AND k16 << #16, [ACx,] ACy
ACy = ACx & (k16 <<< #SHFT)	AND k16 << #SHFT, [ACx,] ACy

Algebraic Syntax	Mnemonic Syntax
Smem = Smem & k16	AND k16, Smem
Bitwise AND Memory with Immediate Value and Compare to Zero	BAND: Bitwise AND Memory with Immediate Value and Compare to Zero
TCx = Smem & k16	BAND Smem, k16, TCx
Bitwise OR	OR: Bitwise OR
dst = dst   src	OR src, dst
dst = src   k8	OR k8, src, dst
dst = src   k16	OR k16, src, dst
dst = src   Smem	OR Smem, src, dst
ACy = ACy   (ACx <<< #SHIFTW)	OR ACx << #SHIFTW[, ACy]
ACy = ACx   (k16 <<< #16)	OR k16 << #16, [ACx,] ACy
ACy = ACx   (k16 <<< #SHFT)	OR k16 << #SHFT, [ACx,] ACy
Smem = Smem   k16	OR k16, Smem
Bitwise Exclusive OR (XOR)	XOR: Bitwise Exclusive OR (XOR)
dst = dst ^ src	XOR src, dst
dst = src ^ k8	XOR k8, src, dst
dst = src ^ k16	XOR k16, src, dst
dst = src ^ Smem	XOR Smem, src, dst
ACy = ACy ^ (ACx <<< #SHIFTW)	XOR ACx << #SHIFTW[, ACy]
ACy = ACx ^ (k16 <<< #16)	XOR k16 << #16, <mark>[ACx,]</mark> ACy
ACy = ACx ^ (k16 <<< #SHFT)	XOR k16 << #SHFT, [ACx,] ACy

Cross-Reference of Algebraic and Mnemonic Instruction Sets

## Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Smem = Smem ^ k16	XOR k16, Smem
Branch Conditionally	BCC: Branch Conditionally
if (cond) goto I4	BCC I4, cond
if (cond) goto L8	BCC L8, cond
if (cond) goto L16	BCC L16, cond
if (cond) goto P24	BCC P24, cond
Branch Unconditionally	B: Branch Unconditionally
goto ACx	B ACx
goto L7	B L7
goto L16	B L16
goto P24	B P24
Branch on Auxiliary Register Not Zero	BCC: Branch on Auxiliary Register Not Zero
if (ARn_mod != #0) goto L16	BCC L16, ARn_mod != #0
Call Conditionally	CALLCC: Call Conditionally
if (cond) call L16	CALLCC L16, cond
if (cond) call P24	CALLCC P24, cond

Algebraic Syntax	Mnemonic Syntax
Call Unconditionally	CALL: Call Unconditionally
call ACx	CALL ACx
call L16	CALL L16
call P24	CALL P24
Circular Addressing Qualifier	.CR: Circular Addressing Qualifier
circular()	<instruction>.CR</instruction>
Clear Accumulator, Auxiliary, or Temporary Register Bit	BCLR: Clear Accumulator, Auxiliary, or Temporary Register Bit
bit(src, Baddr) = #0	BCLR Baddr, src
Clear Memory Bit	BCLR: Clear Memory Bit
bit(Smem, src) = #0	BCLR src, Smem
Clear Status Register Bit	BCLR: Clear Status Register Bit
bit(STx, k4) = #0	BCLR k4, STx_55
	BCLR f-name
Compare Accumulator, Auxiliary, or Temporary Register Content	CMP: Compare Accumulator, Auxiliary, or Temporary Register Content
TCx = uns(src RELOP dst)	CMP[U] src RELOP dst, TCx
Compare Accumulator, Auxiliary, or Temporary Register Content with AND	CMPAND: Compare Accumulator, Auxiliary, or Temporary Register Content with AND
TCx = TCy & uns(src RELOP dst)	CMPAND[U] src RELOP dst, TCy, TCx
TCx = !TCy & uns(src RELOP dst)	CMPAND[U] src RELOP dst, !TCy, TCx

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets	(Continued)
	(0011111000)

Algebraic Syntax	Mnemonic Syntax
Compare Accumulator, Auxiliary, or Temporary Register Content with OR	CMPOR: Compare Accumulator, Auxiliary, or Temporary Register Content with OR
TCx = TCy   uns(src RELOP dst)	CMPOR[U] src RELOP dst, TCy, TCx
TCx = !TCy   uns(src RELOP dst)	CMPOR[U] src RELOP dst, !TCy, TCx
Compare Accumulator, Auxiliary, or Temporary Register Content Maximum	MAX: Compare Accumulator, Auxiliary, or Temporary Register Content Maximum
dst = max(src, dst)	MAX [src,] dst
Compare Accumulator, Auxiliary, or Temporary Register Content Minimum	MIN: Compare Accumulator, Auxiliary, or Temporary Register Content Minimum
dst = min(src, dst)	MIN [src,] dst
Compare and Branch	BCC: Compare and Branch
compare (uns(src RELOP K8)) goto L8	BCC[U] L8, src RELOP K8
Compare and Select Accumulator Content Maximum	MAXDIFF: Compare and Select Accumulator Content Maximum
max_diff(ACx, ACy, ACz, ACw)	MAXDIFF ACx, ACy, ACz, ACw
max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	DMAXDIFF ACx, ACy, ACz, ACw, TRNx
Compare and Select Accumulator Content Minimum	MINDIFF: Compare and Select Accumulator Content Minimum
min_diff(ACx, ACy, ACz, ACw)	MINDIFF ACx, ACy, ACz, ACw
min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)	DMINDIFF ACx, ACy, ACz, ACw, TRNx
Compare Memory with Immediate Value	CMP: Compare Memory with Immediate Value
TCx = (Smem == K16)	CMP Smem == K16, TCx

Cross-Reference of Algebraic and Mnemonic Instruction Sets

Algebraic Syntax	Mnemonic Syntax
Complement Accumulator, Auxiliary, or Temporary Register Bit	BNOT: Complement Accumulator, Auxiliary, or Temporary Register Bit
cbit(src, Baddr)	BNOT Baddr, src
Complement Accumulator, Auxiliary, or Temporary Register Content	NOT: Complement Accumulator, Auxiliary, or Temporary Register Content
dst = ~src	NOT [src,] dst
Complement Memory Bit	BNOT: Complement Memory Bit
cbit(Smem, src)	BNOT src, Smem
Compute Exponent of Accumulator Content	EXP: Compute Exponent of Accumulator Content
Tx = exp(ACx)	EXP ACx, Tx
Compute Mantissa and Exponent of Accumulator Content	MANT::NEXP: Compute Mantissa and Exponent of Accumulator Content
ACy = mant(ACx), Tx = -exp(ACx)	MANT ACx, ACy :: NEXP ACx, Tx
Count Accumulator Bits	BCNT: Count Accumulator Bits
Tx = count(ACx, ACy, TCx)	BCNT ACx, ACy, TCx, Tx

Table 7–1. Cross-Reference of Algebraic ar	nd Mnemonic Instruction Sets	(Continued)
		(Continucu)

Algebraic Syntax	Mnemonic Syntax
Dual 16-Bit Additions	ADD: Dual 16-Bit Additions
HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)	ADD dual(Lmem), [ACx,] ACy
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx	ADD dual(Lmem), Tx, ACx
Dual 16-Bit Addition and Subtraction	ADDSUB: Dual 16-Bit Addition and Subtraction
HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx	ADDSUB Tx, Smem, ACx
HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) - Tx	ADDSUB Tx, dual(Lmem), ACx
Dual 16-Bit Subtractions	SUB: Dual 16-Bit Subtractions
HI(ACy) = HI(ACx) - HI(Lmem), LO(ACy) = LO(ACx) - LO(Lmem)	SUB dual(Lmem), [ACx,] ACy
HI(ACy) = HI(Lmem) - HI(ACx), LO(ACy) = LO(Lmem) - LO(ACx)	SUB ACx, dual(Lmem), ACy
HI(ACx) = Tx - HI(Lmem), LO(ACx) = Tx - LO(Lmem)	SUB dual(Lmem), Tx, ACx
HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) - Tx	SUB Tx, dual(Lmem), ACx
Dual 16-Bit Subtraction and Addition	SUBADD: Dual 16-Bit Subtraction and Addition
HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx	SUBADD Tx, Smem, ACx
HI(ACx) = HI(Lmem) - Tx, LO(ACx) = LO(Lmem) + Tx	SUBADD Tx, dual(Lmem), ACx

Algebraic Syntax	Mnemonic Syntax
Execute Conditionally	XCC: Execute Conditionally
if (cond) execute(AD_Unit)	XCC [label, ]cond
if (cond) execute(D_Unit)	XCCPART [label, ]cond
Expand Accumulator Bit Field	BFXPA: Expand Accumulator Bit Field
dst = field_expand(ACx, k16)	BFXPA k16, ACx, dst
Extract Accumulator Bit Field	BFXTR: Extract Accumulator Bit Field
dst = field_extract(ACx, k16)	BFXTR k16, ACx, dst
Finite Impulse Response Filter, Antisymmetrical	FIRSSUB: Finite Impulse Response Filter, Antisymmetrical
firsn(Xmem, Ymem, coef(Cmem), ACx, ACy)	FIRSSUB Xmem, Ymem, Cmem, ACx, ACy
Finite Impulse Response Filter, Symmetrical	FIRSADD: Finite Impulse Response Filter, Symmetrical
firs(Xmem, Ymem, coef(Cmem), ACx, ACy)	FIRSADD Xmem, Ymem, Cmem, ACx, ACy
Idle	IDLE
idle	IDLE
Least Mean Square (LMS)	LMS: Least Mean Square
lms(Xmem, Ymem, ACx, ACy)	LMS Xmem, Ymem, ACx, ACy
Linear Addressing Qualifier	.LR: Linear Addressing Qualifier
linear()	<instruction>.LR</instruction>

Table 7–1. Cross-Reference of Al	aehraic and Mnemonic	Instruction Sets	(Continued)
	georaie and minemonie	113114011011 0013	(Continucu)

Algebraic Syntax	Mnemonic Syntax
Load Accumulator from Memory	MOV: Load Accumulator from Memory
ACx = <mark>rnd(</mark> Smem << Tx)	MOV [rnd(]Smem << Tx[)], ACx
ACx = low_byte(Smem) << #SHIFTW	MOV low_byte(Smem) << #SHIFTW, ACx
ACx = high_byte(Smem) << #SHIFTW	MOV high_byte(Smem) << #SHIFTW, ACx
ACx = Smem << #16	MOV Smem << #16, ACx
ACx = uns(Smem)	MOV [uns(]Smem[)], ACx
ACx = <mark>uns(</mark> Smem) << #SHIFTW	MOV [uns(]Smem[)] << #SHIFTW, ACx
ACx = <mark>M40(</mark> dbl(Lmem))	MOV <mark>[40]</mark> dbl(Lmem), ACx
LO(ACx) = Xmem, HI(ACx) = Ymem	MOV Xmem, Ymem, ACx
Load Accumulator from Memory with Parallel Store Accumulator Content to Memory	MOV::MOV: Load Accumulator from Memory with Paralle Store Accumulator Content to Memory
ACy = Xmem << #16, Ymem = HI(ACx << T2)	MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem
Load Accumulator Pair from Memory	MOV: Load Accumulator Pair from Memory
pair(HI(ACx)) = Lmem	MOV dbl(Lmem), pair(HI(ACx))
pair(LO(ACx)) = Lmem	MOV dbl(Lmem), pair(LO(ACx))
Load Accumulator with Immediate Value	MOV: Load Accumulator with Immediate Value
ACx = K16 << #16	MOV K16 << #16, ACx
ACx = K16 << #SHFT	MOV K16 << #SHFT, ACx

Algebraic Syntax	Mnemonic Syntax
Load Accumulator, Auxiliary, or Temporary Register from Memory	MOV: Load Accumulator, Auxiliary, or Temporary Register from Memory
dst = Smem	MOV Smem, dst
dst = <mark>uns(</mark> high_byte(Smem))	MOV [uns(]high_byte(Smem)[)], dst
dst = uns(low_byte(Smem))	MOV [uns(]low_byte(Smem)[)], dst
Load Accumulator, Auxiliary, or Temporary Register with Immediate Value	MOV: Load Accumulator, Auxiliary, or Temporary Register with Immediate Value
dst = k4	MOV k4, dst
dst = -k4	MOV –k4, dst
dst = K16	MOV K16, dst
Load Auxiliary or Temporary Register Pair from Memory	MOV: Load Auxiliary or Temporary Register Pair from Memory
pair(TAx) = Lmem	MOV dbl(Lmem), pair(TAx)
Load CPU Register from Memory	MOV: Load CPU Register from Memory
BK03 = Smem	MOV Smem, BK03
BK47 = Smem	MOV Smem, BK47
BKC = Smem	MOV Smem, BKC
BSA01 = Smem	MOV Smem, BSA01
BSA23 = Smem	MOV Smem, BSA23
BSA45 = Smem	MOV Smem, BSA45
BSA67 = Smem	MOV Smem, BSA67
BSAC = Smem	MOV Smem, BSAC
BRC0 = Smem	MOV Smem, BRC0

 Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
BRC1 = Smem	MOV Smem, BRC1
CDP = Smem	MOV Smem, CDP
CSR = Smem	MOV Smem, CSR
DP = Smem	MOV Smem, DP
DPH = Smem	MOV Smem, DPH
PDP = Smem	MOV Smem, PDP
SP = Smem	MOV Smem, SP
SSP = Smem	MOV Smem, SSP
TRN0 = Smem	MOV Smem, TRN0
TRN1 = Smem	MOV Smem, TRN1
RETA = dbl(Lmem)	MOV dbl(Lmem), RETA
Load CPU Register with Immediate Value	MOV: Load CPU Register with Immediate Value
BK03 = k12	MOV k12, BK03
BK47 = k12	MOV k12, BK47
BKC = k12	MOV k12, BKC
BRC0 = k12	MOV k12, BRC0
BRC1 = k12	MOV k12, BRC1
CSR = k12	MOV k12, CSR
DPH = k7	MOV k7, DPH
PDP = k9	MOV k9, PDP
BSA01 = k16	MOV k16, BSA01
BSA23 = k16	MOV k16, BSA23

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
BSA45 = k16	MOV k16, BSA45
BSA67 = k16	MOV k16, BSA67
BSAC = k16	MOV k16, BSAC
CDP = k16	MOV k16, CDP
DP = k16	MOV k16, DP
SP = k16	MOV k16, SP
SSP = k16	MOV k16, SSP
Load Extended Auxiliary Register from Memory	MOV: Load Extended Auxiliary Register from Memory
XAdst = dbl(Lmem)	MOV dbl(Lmem), XAdst
Load Extended Auxiliary Register with Immediate Value	AMOV: Load Extended Auxiliary Register with Immediate Value
XAdst = k23	AMOV k23, XAdst
Load Memory with Immediate Value	MOV: Load Memory with Immediate Value
Smem = K8	MOV K8, Smem
Smem = K16	MOV K16, Smem
Memory Delay	DELAY: Memory Delay
delay(Smem)	DELAY Smem
Memory-Mapped Register Access Qualifier	mmap: Memory-Mapped Register Access Qualifier
mmap()	mmap

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Modify Auxiliary Register Content	AMAR: Modify Auxiliary Register Content
mar(Smem)	AMAR Smem
Modify Auxiliary Register Content with Parallel Multiply	AMAR::MPY: Modify Auxiliary Register Content with Parallel Multiply
mar(Xmem), ACx = <mark>M40(rnd(uns(</mark> Ymem) * <mark>uns</mark> (coef(Cmem))))	AMAR Xmem :: MPY <mark>[R][40] [uns(]</mark> Ymem <mark>[)], [uns(]</mark> Cmem <mark>[)]</mark> , ACx
Modify Auxiliary Register Content with Parallel Multiply and Accumulate	AMAR::MAC: Modify Auxiliary Register Content with Paralle Multiply and Accumulate
mar(Xmem), ACx = <mark>M40(rnd(</mark> ACx + ( <mark>uns(</mark> Ymem) * uns(coef(Cmem)))))	AMAR Xmem :: MAC <mark>[R][40] [uns(]</mark> Ymem <mark>[)], [uns(]</mark> Cmem[)], ACx
mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	AMAR Xmem :: MAC <mark>[R][40] [uns(]</mark> Ymem[)], [uns(]Cmem[)], ACx >> #16
Modify Auxiliary Register Content with Parallel Multiply and Subtract	AMAR::MAS: Modify Auxiliary Register Content with Paralle Multiply and Subtract
mar(Xmem), ACx = <mark>M40(rnd</mark> (ACx – ( <mark>uns(</mark> Ymem) * <mark>uns(</mark> coef(Cmem)))))	AMAR Xmem :: MAS <mark>[R][40] [uns(]</mark> Ymem[)], [ <mark>uns(]</mark> Cmem[)], ACx
Modify Auxiliary or Temporary Register Content	AMOV: Modify Auxiliary or Temporary Register Content
mar(TAy = TAx)	AMOV TAx, TAy
mar(TAx = P8)	AMOV P8, TAx
mar(TAx = D16)	AMOV D16, TAx

Algebraic Syntax	Mnemonic Syntax
Modify Auxiliary or Temporary Register Content by Addition	AADD: Modify Auxiliary or Temporary Register Content by Addition
mar(TAy + TAx)	AADD TAx, TAy
mar(TAx + P8)	AADD P8, TAx
Modify Auxiliary or Temporary Register Content by Subtraction	ASUB: Modify Auxiliary or Temporary Register Content by Subtraction
mar(TAy – TAx)	ASUB TAx, TAy
mar(TAx – P8)	ASUB P8, TAx
Modify Data Stack Pointer	AADD: Modify Data Stack Pointer (SP)
SP = SP + K8	AADD K8, SP
Modify Extended Auxiliary Register Content	AMAR: Modify Extended Auxiliary Register Content
XAdst = mar(Smem)	AMAR Smem, XAdst
Move Accumulator Content to Auxiliary or Temporary Register	MOV: Move Accumulator Content to Auxiliary or Temporary Register
TAx = HI(ACx)	MOV HI(ACx), TAx
Move Accumulator, Auxiliary, or Temporary Register Content	MOV: Move Accumulator, Auxiliary, or Temporary Register Content
dst = src	MOV src, dst
Move Auxiliary or Temporary Register Content to Accumulator	MOV: Move Auxiliary or Temporary Register Content to Accumulator
HI(ACx) = TAx	MOV TAx, HI(ACx)

 Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Move Auxiliary or Temporary Register Content to CPU Register	MOV: Move Auxiliary or Temporary Register Content to CPU Register
BRC0 = TAx	MOV TAx, BRC0
BRC1 = TAx	MOV TAx, BRC1
CDP = TAx	MOV TAx, CDP
CSR = TAx	MOV TAx, CSR
SP = TAx	MOV TAx, SP
SSP = TAx	MOV TAx, SSP
Move CPU Register Content to Auxiliary or Temporary Register	MOV: Move CPU Register Content to Auxiliary or Temporary Register
TAx = BRC0	MOV BRC0, TAx
TAx = BRC1	MOV BRC1, TAx
TAx = CDP	MOV CDP, TAx
TAx = RPTC	MOV RPTC, TAx
TAx = SP	MOV SP, TAx
TAx = SSP	MOV SSP, TAX
Move Extended Auxiliary Register Content	MOV: Move Extended Auxiliary Register Content
xdst = xsrc	MOV xsrc, xdst

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Move Memory to Memory	MOV: Move Memory to Memory
Smem = coef(Cmem)	MOV Cmem, Smem
coef(Cmem) = Smem	MOV Smem, Cmem
Lmem = dbl(coef(Cmem))	MOV Cmem, dbl(Lmem)
dbl(coef(Cmem)) = Lmem	MOV dbl(Lmem), Cmem
dbl(Ymem) = dbl(Xmem)	MOV dbl(Xmem), dbl(Ymem)
Ymem = Xmem	MOV Xmem, Ymem
Multiply	MPY: Multiply
ACy = rnd(ACy * ACx)	MPY[R] [ACx,] ACy
ACy = rnd(ACx * Tx)	MPY[R] Tx, [ACx,] ACy
ACy = rnd(ACx * K8)	MPYK[R] K8, [ACx,] ACy
ACy = rnd(ACx * K16)	MPYK[R] K16, [ACx,] ACy
ACx = rnd(Smem * coef(Cmem))[, T3 = Smem]	MPYM[R] [T3 = ]Smem, Cmem, ACx
ACy = rnd(Smem * ACx)[, T3 = Smem]	MPYM[R] [T3 = ]Smem, [ACx,] ACy
ACx = rnd(Smem * K8)[, T3 = Smem]	MPYMK[R] [T3 = ]Smem, K8, ACx
ACx = M40(rnd(uns(Xmem) * uns(Ymem)))[, T3 = Xmem]	MPYM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], ACx
ACx = rnd(uns(Tx * Smem))[, T3 = Smem]	MPYM[R][U] [T3 = ]Smem, Tx, ACx
Multiply with Parallel Multiply and Accumulate	MPY::MAC: Multiply with Parallel Multiply and Accumulate
ACx = <mark>M40(rnd(uns(</mark> Xmem) * uns(coef(Cmem)))), ACy = <mark>M40(rnd(</mark> (ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	MPY[ <mark>R][40] [uns(</mark> ]Xmem[)], [uns(]Cmem[)], ACx :: MAC[ <mark>R][40] [uns(]</mark> Ymem[)], [uns(]Cmem[)], ACy >> #16

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets	(Continued)
	(Continueu)

Algebraic Syntax	Mnemonic Syntax
Multiply with Parallel Store Accumulator Content to Memory	MPYM::MOV: Multiply with Parallel Store Accumulator Content to Memory
ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [,T3 = Xmem]	MPYM <mark>[R] [T3 = ]</mark> Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
Multiply and Accumulate (MAC)	MAC: Multiply and Accumulate
ACy = rnd(ACy + (ACx * Tx))	MAC[R] ACx, Tx, ACy[, ACy]
ACy = rnd((ACy * Tx) + ACx)	MAC[R] ACy, Tx, ACx, ACy
ACy = rnd(ACx + (Tx * K8))	MACK[ <mark>R]</mark> Tx, K8, [ACx,] ACy
ACy = rnd(ACx + (Tx * K16))	MACK[ <mark>R]</mark> Tx, K16, [ACx,] ACy
ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem]	MACM[R] [T3 = ]Smem, Cmem, ACx
ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]	MACM[R] [T3 = ]Smem, [ACx,] ACy
ACy = <mark>rnd(</mark> ACx + (Tx * Smem))[, T3 = Smem]	MACM[R] [T3 = ]Smem, Tx, [ACx,] ACy
ACy = rnd(ACx + (Smem * K8))[, T3 = Smem ]	MACMK[R] [T3 = ]Smem, K8, [ACx,] ACy
ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]	MACM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy
ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]	MACM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16 [, ACy]
Multiply and Accumulate with Parallel Delay	MACMZ: Multiply and Accumulate with Parallel Delay
ACx = rnd(ACx + (Smem * coef(Cmem)))[, T3 = Smem], delay(Smem)	MACM[R]Z [T3 = ]Smem, Cmem, ACx
Multiply and Accumulate with Parallel Load Accumulator from Memory	MACM::MOV: Multiply and Accumulate with Parallel Load Accumulator from Memory
ACx = <mark>rnd(</mark> ACx + (Tx * Xmem)), ACy = Ymem << #16 <mark>[,T3 = Xmem]</mark>	MACM <mark>[R] [T3 = ]</mark> Xmem, Tx, ACx :: MOV Ymem << #16, ACy

Algebraic Syntax	Mnemonic Syntax
Multiply and Accumulate with Parallel Multiply	MAC::MPY: Multiply and Accumulate with Parallel Multiply
ACx = M40(rnd(ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) * uns(coef(Cmem))))	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy
Multiply and Accumulate with Parallel Store Accumulator Content to Memory	MACM::MOV: Multiply and Accumulate with Parallel Store Accumulator Content to Memory
ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	MACM[R] [T3 = ]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
Multiply and Subtract	MAS: Multiply and Subtract
ACy = rnd(ACy - (ACx * Tx))	MAS[R] Tx, [ACx,] ACy
ACx = rnd(ACx - (Smem * coef(Cmem)))[, T3 = Smem]	MASM[R] [T3 = ]Smem, Cmem, ACx

ACy = rnd(ACy - (Smem \* ACx))[, T3 = Smem]

ACy = rnd(ACx - (Tx \* Smem))[, T3 = Smem]

ACy = M40(rnd(ACx - (uns(Xmem) \* uns(Ymem))))[, T3 = Xmem]

#### Multiply and Subtract with Parallel Load Accumulator from Memory

ACx = rnd(ACx - (Tx \* Xmem)),ACy = Ymem << #16 [,T3 = Xmem]

#### Multiply and Subtract with Parallel Multiply

ACx = M40(rnd(ACx - (uns(Xmem) \* uns(coef(Cmem))))), ACy = M40(rnd(uns(Ymem) \* uns(coef(Cmem))))

MASM[R] [T3 = ]Smem, [ACx,] ACy

MASM[R] [T3 = ]Smem, Tx, [ACx,] ACy

MASM[R][40] [T3 = ][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy

#### MASM::MOV: Multiply and Subtract with Parallel Load Accumulator from Memory

MASM[R] [T3 = ]Xmem, Tx, ACx :: MOV Ymem << #16, ACy

#### MAS::MPY: Multiply and Subtract with Parallel Multiply

MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

#### Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
Multiply and Subtract with Parallel Multiply and Accumulate	MAS::MAC: Multiply and Subtract with Parallel Multiply and Accumulate
ACx = <mark>M40(rnd(</mark> ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy
ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16
Multiply and Subtract with Parallel Store Accumulator Content to Memory	MASM::MOV: Multiply and Subtract with Parallel Store Accumulator Content to Memory
ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]	MASM[R] [T3 = ]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
Negate Accumulator, Auxiliary, or Temporary Register Content	NEG: Negate Accumulator, Auxiliary, or Temporary Register
dst = -src	NEG [src,] dst
No Operation	NOP: No Operation
nop	NOP
nop_16	NOP_16
Parallel Modify Auxiliary Register Contents	AMAR: Parallel Modify Auxiliary Register Contents
mar(Xmem), mar(Ymem), mar(coef(Cmem))	AMAR Xmem, Ymem, Cmem
Parallel Multiplies	MPY::MPY: Parallel Multiplies
ACx = <mark>M40(rnd(uns(</mark> Xmem) * uns(coef(Cmem)))), ACy = <mark>M40(rnd(uns(</mark> Ymem) * uns(coef(Cmem))))	MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets	(Continued)
---	-------------

Algebraic Syntax	Mnemonic Syntax
Parallel Multiply and Accumulates	MAC::MAC: Parallel Multiply and Accumulates
ACx = <mark>M40(rnd(</mark> ACx + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M4(rnd(ACy + (uns(Ymem) * uns(coef(Cmem)))))	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16 :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy
ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(coef(Cmem)))))	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16 :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16
Parallel Multiply and Subtracts	MAS::MAS: Parallel Multiply and Subtracts
ACx = M40(rnd(ACx - (uns(Xmem) * uns(coef(Cmem))))), ACy = M40(rnd(ACy - (uns(Ymem) * uns(coef(Cmem)))))	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy
Peripheral Port Register Access Qualifiers	port: Peripheral Port Register Access Qualifiers
readport()	port(Smem)
writeport()	port(Smem)
Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers	POPBOTH: Pop Accumulator or Extended Auxiliary Register Content from Stack Pointers
xdst = popboth()	POPBOTH xdst
Pop Top of Stack	POP: Pop Top of Stack
dst1, dst2 = pop()	POP dst1, dst2
dst = pop()	POP dst
dst, Smem = pop()	POP dst, Smem
ACx = dbl(pop())	POP ACx
Smem = pop()	POP Smem

#### Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Algebraic Syntax	Mnemonic Syntax
dbl(Lmem) = pop()	POP dbl(Lmem)
Push Accumulator or Extended Auxiliary Register Content to Stack Pointers	PSHBOTH: Push Accumulator or Extended Auxiliary Register Content to Stack Pointers
pshboth(xsrc)	PSHBOTH xsrc
Push to Top of Stack	PSH: Push to Top of Stack
push(src1, src2)	PSH src1, src2
push(src)	PSH src
push(src, Smem)	PSH src, Smem
dbl(push(ACx))	PSH ACx
push(Smem)	PSH Smem
push(dbl(Lmem))	PSH dbl(Lmem)
Repeat Block of Instructions Unconditionally	RPTB: Repeat Block of Instructions Unconditionally
localrepeat{ }	RPTBLOCAL pmad
blockrepeat{ }	RPTB pmad
Repeat Single Instruction Conditionally	RPTCC: Repeat Single Instruction Conditionally
while (cond && (RPTC < k8)) repeat	RPTCC k8, cond
Repeat Single Instruction Unconditionally	RPT: Repeat Single Instruction Unconditionally
repeat(k8)	RPT k8
repeat(k16)	RPT k16
repeat(CSR)	RPT CSR

Algebraic Syntax	Mnemonic Syntax
Repeat Single Instruction Unconditionally and Decrement CSR	RPTSUB: Repeat Single Instruction Unconditionally and Decrement CSR
repeat(CSR), CSR –= k4	RPTSUB CSR, k4
Repeat Single Instruction Unconditionally and Increment CSR	RPTADD: Repeat Single Instruction Unconditionally and Increment CSR
repeat(CSR), CSR += TAx	RPTADD CSR, TAx
repeat(CSR), CSR += k4	RPTADD CSR, k4
Return Conditionally	RETCC: Return Conditionally
if (cond) return	RETCC cond
Return Unconditionally	RET: Return Unconditionally
return	RET
Return from Interrupt	RETI: Return from Interrupt
return_int	RETI
Rotate Left Accumulator, Auxiliary, or Temporary Register Content	ROL: Rotate Left Accumulator, Auxiliary, or Temporary Register Content
dst = BitOut \\ src \\ BitIn	ROL BitOut, src, BitIn, dst
Rotate Right Accumulator, Auxiliary, or Temporary Register Content	ROR: Rotate Right Accumulator, Auxiliary, or Temporary Register Content
dst = BitIn // src // BitOut	ROR BitIn, src, BitOut, dst

 Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

7-24

Table 7–1. Cross-Reference of	of Algebraic and Mnemonic Instruction Sets	(Continued)
-------------------------------	--	-------------

Algebraic Syntax	Mnemonic Syntax
Round Accumulator Content	ROUND: Round Accumulator Content
ACy = rnd(ACx)	ROUND [ACx,] ACy
Saturate Accumulator Content	SAT: Saturate Accumulator Content
ACy = saturate(rnd(ACx))	SAT[R] [ACx,] ACy
Set Accumulator, Auxiliary, or Temporary Register Bit	BSET: Set Accumulator, Auxiliary, or Temporary Register Bit
bit(src, Baddr) = #1	BSET Baddr, src
Set Memory Bit	BSET: Set Memory Bit
bit(Smem, src) = #1	BSET src, Smem
Set Status Register Bit	BSET: Set Status Register Bit
bit(STx, k4) = #1	BSET k4, STx_55
	BSET f-name
Shift Accumulator Content Conditionally	SFTCC: Shift Accumulator Content Conditionally
ACx = sftc(ACx, TCx)	SFTCC ACx, TCx
Shift Accumulator Content Logically	SFTL: Shift Accumulator Content Logically
ACy = ACx <<< Tx	SFTL ACx, Tx[, ACy]
ACy = ACx <<< #SHIFTW	SFTL ACx, #SHIFTW[, ACy]

Algebraic Syntax	Mnemonic Syntax
Shift Accumulator, Auxiliary, or Temporary Register Content Logically	SFTL: Shift Accumulator, Auxiliary, or Temporary Register Content Logically
dst = dst <<< #1	SFTL dst, #1
dst = dst >>> #1	SFTL dst, #–1
Signed Shift of Accumulator Content	SFTS: Signed Shift of Accumulator Content
$ACy = ACx \ll Tx$	SFTS ACx, Tx[, ACy]
ACy = ACx << #SHIFTW	SFTS ACx, #SHIFTW[, ACy]
ACy = ACx < <c td="" tx<=""><td>SFTSC ACx, Tx<mark>[, ACy]</mark></td></c>	SFTSC ACx, Tx <mark>[, ACy]</mark>
ACy = ACx < <c #shiftw<="" td=""><td>SFTSC ACx, #SHIFTW[, ACy]</td></c>	SFTSC ACx, #SHIFTW[, ACy]
Signed Shift of Accumulator, Auxiliary, or Temporary Register Content	SFTS: Signed Shift of Accumulator, Auxiliary, or Temporary Register Content
dst = dst >> #1	SFTS dst, #–1
dst = dst << #1	SFTS dst, #1
Software Interrupt	INTR: Software Interrupt
intr(k5)	INTR k5
Software Reset	RESET: Software Reset
reset	RESET
Software Trap	TRAP: Software Trap
trap(k5)	TRAP k5

 Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets	(Continued)

Algebraic Syntax	Mnemonic Syntax
Square	SQR: Square
ACy = rnd(ACx * ACx)	SQR[R] [ACx,] ACy
ACx = rnd(Smem * Smem)[, T3 = Smem]	SQRM[R] [T3 = ]Smem, ACx
Square and Accumulate	SQA: Square and Accumulate
ACy = rnd(ACy + (ACx * ACx))	SQA[R] [ACx,] ACy
ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]	SQAM[R] [T3 = ]Smem, [ACx,] ACy
Square and Subtract	SQS: Square and Subtract
ACy = rnd(ACy - (ACx * ACx))	SQS[R] [ACx,] ACy
ACy = rnd(ACx - (Smem * Smem))[, T3 = Smem]	SQSM[R] [T3 = ]Smem, [ACx,] ACy
Square Distance	SQDST: Square Distance
sqdst(Xmem, Ymem, ACx, ACy)	SQDST Xmem, Ymem, ACx, ACy
Store Accumulator Content to Memory	MOV: Store Accumulator Content to Memory
Smem = HI(ACx)	MOV HI(ACx), Smem
Smem = HI(rnd(ACx))	MOV [rnd(]HI(ACx)[)], Smem
Smem = LO(ACx << Tx)	MOV ACx << Tx, Smem
Smem = HI(rnd(ACx << Tx))	MOV [rnd(]HI(ACx << Tx)[)], Smem
Smem = LO(ACx << #SHIFTW)	MOV ACx << #SHIFTW, Smem
Smem = HI(ACx << #SHIFTW)	MOV HI(ACx << #SHIFTW), Smem
Smem = HI <mark>(rnd</mark> (ACx << #SHIFTW))	MOV [rnd(]HI(ACx << #SHIFTW)[)], Smem
Smem = HI(saturate(uns(rnd(ACx))))	MOV [uns(] [rnd(]HI[(saturate](ACx)[)))], Smem

Algebraic Syntax	Mnemonic Syntax
Smem = HI(saturate(uns(rnd(ACx << Tx))))	MOV [uns(] [rnd(]HI[(saturate](ACx << Tx)[)))], Smem
Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))	MOV [uns(] [rnd(]HI[(saturate](ACx << #SHIFTW)[)))], Smem
dbl(Lmem) = ACx	MOV ACx, dbl(Lmem)
dbl(Lmem) = saturate(uns(ACx))	MOV [uns(]saturate(ACx)[)], dbl(Lmem)
HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1	MOV ACx >> #1, dual(Lmem)
Xmem = LO(ACx), Ymem = HI(ACx)	MOV ACx, Xmem, Ymem
Store Accumulator Pair Content to Memory	MOV: Store Accumulator Pair Content to Memory
Lmem = pair(HI(ACx))	MOV pair(HI(ACx)), dbl(Lmem)
Lmem = pair(LO(ACx))	MOV pair(LO(ACx)), dbl(Lmem)
Store Accumulator, Auxiliary, or Temporary Register Content to Memory	MOV: Store Accumulator, Auxiliary, or Temporary Register Content to Memory
Smem = src	MOV src, Smem
high_byte(Smem) = src	MOV src, high_byte(Smem)
low_byte(Smem) = src	MOV src, low_byte(Smem)
Store Auxiliary or Temporary Register Pair Content to Memory	MOV: Store Auxiliary or Temporary Register Pair Content to Memory
Lmem = pair(TAx)	MOV pair(TAx), dbl(Lmem)
Store CPU Register Content to Memory	MOV: Store CPU Register Content to Memory
Smem = BK03	MOV BK03, Smem
Smem = BK47	MOV BK47, Smem

Algebraic Syntax	Mnemonic Syntax
Smem = BKC	MOV BKC, Smem
Smem = BSA01	MOV BSA01, Smem
Smem = BSA23	MOV BSA23, Smem
Smem = BSA45	MOV BSA45, Smem
Smem = BSA67	MOV BSA67, Smem
Smem = BSAC	MOV BSAC, Smem
Smem = BRC0	MOV BRC0, Smem
Smem = BRC1	MOV BRC1, Smem
Smem = CDP	MOV CDP, Smem
Smem = CSR	MOV CSR, Smem
Smem = DP	MOV DP, Smem
Smem = DPH	MOV DPH, Smem
Smem = PDP	MOV PDP, Smem
Smem = SP	MOV SP, Smem
Smem = SSP	MOV SSP, Smem
Smem = TRN0	MOV TRN0, Smem
Smem = TRN1	MOV TRN1, Smem
dbl(Lmem) = RETA	MOV RETA, dbl(Lmem)
Store Extended Auxiliary Register Content to Memory	MOV: Store Extended Auxiliary Register Content to Memory
dbl(Lmem) = XAsrc	MOV XAsrc, dbl(Lmem)
Subtract Conditionally	SUBC: Subtract Conditionally
subc(Smem, ACx, ACy)	SUBC Smem, [ACx,] ACy

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

7-29

Cross-Reference of Algebraic and Mnemonic Instruction Sets

Algebraic Syntax	Mnemonic Syntax
Subtraction	SUB: Subtraction
dst = dst - src	SUB [src,] dst
dst = dst - k4	SUB k4, dst
dst = src - K16	SUB K16, [src,] dst
dst = src – Smem	SUB Smem, [src,] dst
dst = Smem – src	SUB src, Smem, dst
ACy = ACy - (ACx << Tx)	SUB ACx << Tx, ACy
ACy = ACy – (ACx << #SHIFTW)	SUB ACx << #SHIFTW, ACy
ACy = ACx – (K16 << #16)	SUB K16 << #16, <mark>[ACx,]</mark> ACy
ACy = ACx – (K16 << #SHFT)	SUB K16 << #SHFT, [ACx,] ACy
ACy = ACx – (Smem << Tx)	SUB Smem << Tx, <mark>[ACx,]</mark> ACy
ACy = ACx – (Smem << #16)	SUB Smem << #16, [ACx,] ACy
ACy = (Smem << #16) – ACx	SUB ACx, Smem << #16, ACy
ACy = ACx - uns(Smem) - BORROW	SUB [uns(]Smem[)], BORROW, [ACx,] ACy
ACy = ACx - uns(Smem)	SUB [uns(]Smem[)], [ACx,] ACy
ACy = ACx - ( <mark>uns(</mark> Smem) << #SHIFTW)	SUB [uns(]Smem[)] << #SHIFTW, [ACx,] ACy
ACy = ACx - dbl(Lmem)	SUB dbl(Lmem), [ACx,] ACy
ACy = dbl(Lmem) – ACx	SUB ACx, dbl(Lmem), ACy
ACx = (Xmem << #16) – (Ymem << #16)	SUB Xmem, Ymem, ACx
Subtraction with Parallel Store Accumulator Content to Memory	SUB::MOV: Subtraction with Parallel Store Accumulator Content to Memory
ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)	SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem

Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

7-30

Table 7–1. Cross-Reference of	of Algebraic and Mnemonic Instruction Sets	(Continued)
-------------------------------	--	-------------

Algebraic Syntax	Mnemonic Syntax	
Swap Accumulator Content	SWAP: Swap Accumulator Content	
swap(ACx, ACy)	SWAP ACx, ACy	
Swap Accumulator Pair Content	SWAPP: Swap Accumulator Pair Content	
swap(pair(AC0), pair(AC2))	SWAPP AC0, AC2	
Swap Auxiliary Register Content	SWAP: Swap Auxiliary Register Content	
swap(ARx, ARy)	SWAP ARx, ARy	
Swap Auxiliary Register Pair Content	SWAPP: Swap Auxiliary Register Pair Content	
swap(pair(AR0), pair(AR2))	SWAPP AR0, AR2	
Swap Auxiliary and Temporary Register Content	SWAP: Swap Auxiliary and Temporary Register Content	
swap(ARx, Tx)	SWAP ARx, Tx	
Swap Auxiliary and Temporary Register Pair Content	SWAPP: Swap Auxiliary and Temporary Register Pair Content	
swap(pair(ARx), pair(Tx))	SWAPP ARx, Tx	
Swap Auxiliary and Temporary Register Pairs Content	SWAP4: Swap Auxiliary and Temporary Register Pairs Content	
swap(block(AR4), block(T0))	SWAP4 AR4, T0	
Swap Temporary Register Content	SWAP: Swap Temporary Register Content	
swap(Tx, Ty)	SWAP Tx, Ty	
Swap Temporary Register Pair Content	SWAPP: Swap Temporary Register Pair Content	
swap(pair(T0), pair(T2))	SWAPP T0, T2	

Algebraic Syntax	Mnemonic Syntax
Test Accumulator, Auxiliary, or Temporary Register Bit	BTST: Test Accumulator, Auxiliary, or Temporary Register Bit
TCx = bit(src, Baddr)	BTST Baddr, src, TCx
Test Accumulator, Auxiliary, or Temporary Register Bit Pair	BTSTP: Test Accumulator, Auxiliary, or Temporary Register Bit Pair
bit(src, pair(Baddr))	BTSTP Baddr, src
Test Memory Bit	BTST: Test Memory Bit
TCx = bit(Smem, src)	BTST src, Smem, TCx
TCx = bit(Smem, k4)	BTST k4, Smem, TCx
Test and Clear Memory Bit	BTSTCLR: Test and Clear Memory Bit
TCx = bit(Smem, k4), bit(Smem, k4) = #0	BTSTCLR k4, Smem, TCx
Test and Complement Memory Bit	BTSTNOT: Test and Complement Memory Bit
TCx = bit(Smem, k4), cbit(Smem, k4)	BTSTNOT k4, Smem, TCx
Test and Set Memory Bit	BTSTSET: Test and Set Memory Bit
TCx = bit(Smem, k4), bit(Smem, k4) = #1	BTSTSET k4, Smem, TCx

#### Table 7–1. Cross-Reference of Algebraic and Mnemonic Instruction Sets (Continued)

7-32

# Index

## A

abdst 5-2 absolute addressing modes 3-3 I/O absolute 3-3 k16 absolute 3-3 k23 absolute 3-3 Absolute Distance (abdst) 5-2 Absolute Value 5-4 Addition 5-7 Addition or Subtraction Conditionally (adsc) 5-31 Addition or Subtraction Conditionally with Shift (ads2c) 5-33 Addition with Absolute Value 5-27 Addition with Parallel Store Accumulator Content to Memory 5-29 Addition, Subtraction, or Move Accumulator Content Conditionally (adsc) 5-36 addressing modes absolute 3-3 direct 3-4 indirect 3-6 introduction 3-2 ads2c 5-33 adsc 5-31, 5-36 affect of status bits 1-9 algebraic instruction set cross-reference to mnemonic instruction set 7-1 AND 5-38 Antisymmetrical Finite Impulse Response Filter (firsn) 5-168 arithmetic absolute distance 5-2 absolute value 5-4 addition 5-7 addition or subtraction conditionally 5-31, 5-36 addition or subtraction conditionally with shift 5-33 addition with absolute value 5-27 compare memory with immediate value 5-126 compute exponent of accumulator content 5-131 compute mantissa and exponent of accumulator content 5-132 dual 16-bit addition and subtraction 5-140 dual 16-bit additions 5-135 dual 16-bit subtraction and addition 5-154 dual 16-bit subtractions 5-145 finite impulse response filter, antisymmetrical 5-168 finite impulse response filter, symmetrical 5-170 least mean square 5-173 multiply 5-255 multiply and accumulate 5-271 multiply and subtract 5-294 negation 5-313 round accumulator content 5-380 saturate accumulator content 5-382 square 5-419 square and accumulate 5-422 square and subtract 5-425 square distance 5-428 subtract conditionally 5-463 subtraction 5-465

#### B

```
bit field comparison 5-47
bit field counting 5-134
bit field expand 5-166
bit field extract 5-167
```

bit manipulation bitwise AND memory with immediate value and compare to zero 5-47 clear accumulator, auxiliary, or temporary register bit 5-88 clear memory bit 5-89 clear status register bit 5-90 complement accumulator, auxiliary, or temporary register bit 5-128 complement accumulator, auxiliary, or temporary register content 5-129 complement memory bit 5-130 expand accumulator bit field 5-166 extract accumulator bit field 5-167 set accumulator, auxiliary, or temporary register bit 5-384 set memory bit 5-385 set status register bit 5-386 test accumulator, auxiliary, or temporary register bit 5-504 test accumulator, auxiliary, or temporary register bit pair 5-506 test and clear memory bit 5-511 test and complement memory bit 5-512 test and set memory bit 5-513 test memory bit 5-508 Bitwise AND 5-38 Bitwise AND Memory with Immediate Value and Compare to Zero 5-47 bitwise complement 5-129 Bitwise Exclusive OR (XOR) 5-57 Bitwise OR 5-48 blockrepeat 5-346 branch conditionally 5-66 on auxiliary register not zero 5-74 unconditionally 5-70 Branch Conditionally (if goto) 5-66 Branch on Auxiliary Register Not Zero (if goto) 5-74 Branch Unconditionally (goto) 5-70



```
call 5-83
conditionally 5-77
unconditionally 5-83
Call Conditionally (if call) 5-77
```

Call Unconditionally (call) 5-83 cbit 5-128, 5-130 circular 5-87 circular addressing 3-20 Circular Addressing Qualifier (circular) 5-87 clear accumulator bit 5-88 auxiliary register bit 5-88 memory bit 5-89 status register bit 5-90 temporary register bit 5-88 Clear Accumulator Bit 5-88 Clear Auxiliary Register Bit 5-88 Clear Memory Bit 5-89 Clear Status Register Bit 5-90 Clear Temporary Register Bit 5-88 compare accumulator, auxiliary, or temporary register content 5-93 accumulator, auxiliary, or temporary register content maximum 5-105 accumulator, auxiliary, or temporary register content minimum 5-108 accumulator, auxiliary, or temporary register content with AND 5-95 accumulator, auxiliary, or temporary register content with OR 5-100 and branch 5-111 and select accumulator content maximum 5-114 and select accumulator content minimum 5-120 memory with immediate value 5-126 Compare Accumulator Content 5-93 **Compare Accumulator Content Maximum** (max) 5-105 **Compare Accumulator Content Minimum** (min) 5-108 Compare Accumulator Content with AND 5-95 Compare Accumulator Content with OR 5-100 Compare and Branch 5-111 compare and goto 5-111 Compare and Select Accumulator Content Maximum (max\_diff) 5-114 Compare and Select Accumulator Content Minimum (min\_diff) 5-120 Compare Auxiliary Register Content 5-93 Compare Auxiliary Register Content Maximum

(max) 5-105

Compare Auxiliary Register Content Minimum (min) 5-108 Compare Auxiliary Register Content with AND 5-95 Compare Auxiliary Register Content with OR 5-100 compare maximum 5-105 Compare Memory with Immediate Value 5-126 compare minimum 5-108 Compare Temporary Register Content 5-93 Compare Temporary Register Content Maximum (max) 5-105 Compare Temporary Register Content Minimum (min) 5-108 Compare Temporary Register Content with AND 5-95 Compare Temporary Register Content with OR 5-100 complement accumulator bit 5-128 accumulator content 5-129 auxiliary register bit 5-128 auxiliary register content 5-129 memory bit 5-130 temporary register bit 5-128 temporary register content 5-129 Complement Accumulator Bit (cbit) 5-128 Complement Accumulator Content 5-129 Complement Auxiliary Register Bit (cbit) 5-128 Complement Auxiliary Register Content 5-129 Complement Memory Bit (cbit) 5-130 Complement Temporary Register Bit (cbit) 5-128 Complement Temporary Register Content 5-129 Compute Exponent of Accumulator Content (exp) 5-131 Compute Mantissa and Exponent of Accumulator Content 5-132 cond field 1-7 conditional addition or subtraction 5-31 addition or subtraction with shift 5-33 addition, subtraction, or move accumulator content 5-36 branch 5-66

call 5-77 execute 5-159 repeat single instruction 5-357 return 5-370 shift 5-389 subtract 5-463 count 5-134 Count Accumulator Bits (count) 5-134 Cross-Reference to Algebraic and Mnemonic Instruction Sets 7-1

## D

delay 5-212 direct addressing modes 3-4 DP direct 3-4 PDP direct 3-5 register-bit direct 3-5 SP direct 3-5 Dual 16-Bit Addition and Subtraction 5-140 Dual 16-Bit Additions 5-135 dual 16-bit arithmetic addition and subtraction 5-140 additions 5-135 subtraction and addition 5-154 subtractions 5-145 Dual 16-Bit Subtraction and Addition 5-154 Dual 16-Bit Subtractions 5-145

#### Ξ

Execute Conditionally (if execute) 5-159 exp 5-131, 5-132 Expand Accumulator Bit Field (field\_expand) 5-166 extended auxiliary register (XAR) load from memory 5-209 load with immediate value 5-210 modify content 5-238 move content 5-247 pop content from stack pointers 5-330 push content to stack pointers 5-338 store to memory 5-462 Extract Accumulator Bit Field (field\_extract) 5-167

# F

field\_expand 5-166 field\_extract 5-167 finite impulse response (FIR) filter antisymmetrical 5-168 symmetrical 5-170 firs 5-170 firsn 5-168



goto 5-70

idle 5-172 if call 5-77 if execute 5-159 if goto 5-66, 5-74 if return 5-370 indirect addressing modes 3-6 AR indirect 3-6 CDP indirect 3-16 coefficient indirect 3-18 dual AR indirect 3-14 initialize memory 5-211 instruction qualifier circular addressing 5-87 linear addressing 5-175 memory-mapped register access 5-213 instruction set abbreviations 1-2 affect of status bits 1-9 conditional fields 1-7 nonrepeatable instructions 1-20 notes 1-14 opcode symbols and abbreviations 6-16 opcodes 6-2 operators 1-6 rules 1-14 symbols 1-2 terms 1-2 instruction set conditional fields 1-7 instruction set notes and rules 1-14

instruction set opcode abbreviations 6-16 symbols 6-16 instruction set opcodes 6-2 instruction set summary 4-1 instruction set terms, symbols, and abbreviations 1-2 interrupt 5-411 intr 5-411



Least Mean Square (lms) 5-173 linear 5-175 Linear Addressing Qualifier (linear) 5-175 List of Algebraic Instruction Opcodes 6-1 lms 5-173 load accumulator from memory 5-176 accumulator from memory with parallel store accumulator content to memory 5-185 accumulator pair from memory 5-187 accumulator with immediate value 5-190 accumulator, auxiliary, or temporary register from memory 5-193 accumulator, auxiliary, or temporary register with immediate value 5-199 auxiliary or temporary register pair from memory 5-203 CPU register from memory 5-204 CPU register with immediate value 5-207 extended auxiliary register (XAR) from memory 5-209 extended auxiliary register (XAR) with immediate value 5-210 memory with immediate value 5-211 Load Accumulator from Memory 5-176, 5-193 Load Accumulator from Memory with Parallel Store Accumulator Content to Memory 5-185 Load Accumulator Pair from Memory 5-187 Load Accumulator with Immediate Value 5-190, 5-199 Load Auxiliary Register from Memory 5-193 Load Auxiliary Register Pair from Memory 5-203 Load Auxiliary Register with Immediate Value 5-199 Load CPU Register from Memory 5-204

Index-4

Load CPU Register with Immediate Value 5-207 Load Extended Auxiliary Register (XAR) from Memory 5-209 Load Extended Auxiliary Register (XAR) with Immediate Value 5-210 Load Memory with Immediate Value 5-211 Load Temporary Register from Memory 5-193 Load Temporary Register Pair from Memory 5-203 Load Temporary Register with Immediate Value 5-199 localrepeat 5-346 logical bitwise AND 5-38 bitwise OR 5-48 bitwise XOR 5-57 count accumulator bits 5-134 shift accumulator content logically 5-391 shift accumulator, auxiliary, or temporary register content logically 5-394

## Μ

mant 5-132 mar 5-214, 5-225, 5-229, 5-233, 5-238, 5-316 max 5-105 max\_diff 5-114 max\_diff\_dbl 5-114 memory bit clear 5-89 complement (not) 5-130 set 5-385 test 5-508 test and clear 5-511 test and complement 5-512 test and set 5-513 Memory Delay (delay) 5-212 Memory-Mapped Register Access Qualifier (mmap) 5-213 min 5-108 min\_diff 5-120 min diff dbl 5-120 mmap 5-213 mnemonic instruction set cross-reference to algebraic instruction set 7-1

modify auxiliary or temporary register content 5-225 auxiliary or temporary register content by addition 5-229 auxiliary or temporary register content by subtraction 5-233 auxiliary register content 5-214 auxiliary register content with parallel multiply 5-216 auxiliary register content with parallel multiply and accumulate 5-218 auxiliary register content with parallel multiply and subtract 5-223 data stack pointer 5-237 extended auxiliary register (XAR) content 5-238 Modify Auxiliary Register Content (mar) 5-214, 5-225 Modify Auxiliary Register Content by Addition (mar) 5-229 Modify Auxiliary Register Content by Subtraction (mar) 5-233 Modify Auxiliary Register Content with Parallel Multiply (mar) 5-216 Modify Auxiliary Register Content with Parallel Multiply and Accumulate (mar) 5-218 Modify Auxiliary Register Content with Parallel Multiply and Subtract (mar) 5-223 Modify Data Stack Pointer 5-237 Modify Extended Auxiliary Register Content (mar) 5-238 Modify Temporary Register Content (mar) 5-225 Modify Temporary Register Content by Addition (mar) 5-229 Modify Temporary Register Content by Subtraction (mar) 5-233 move accumulator content to auxiliary or temporary register 5-239 accumulator, auxiliary, or temporary register content 5-240 auxiliary or temporary register content to accumulator 5-242 auxiliary or temporary register content to CPU register 5-243 CPU register content to auxiliary or temporary register 5-245 extended auxiliary register content 5-247 memory delay 5-212 memory to memory 5-248

move (continued) pop accumulator or extended auxiliary register content from stack pointers 5-330 pop top of stack 5-331 push accumulator or extended auxiliary register content to stack pointers 5-338 push to top of stack 5-339 swap accumulator content 5-492 swap accumulator pair content 5-493 swap auxiliary and temporary register content 5-496 swap auxiliary and temporary register pair content 5-498 swap auxiliary and temporary register pairs content 5-500 swap auxiliary register content 5-494 swap auxiliary register pair content 5-495 swap temporary register content 5-502 swap temporary register pair content 5-503 Move Accumulator Content 5-240 Move Accumulator Content to Auxiliary Register 5-239 Move Accumulator Content to Temporary Register 5-239 Move Auxiliary Register Content 5-240 Move Auxiliary Register Content to Accumulator 5-242 Move Auxiliary Register Content to CPU Register 5-243 Move CPU Register Content to Auxiliary Register 5-245 Move CPU Register Content to Temporary Register 5-245 Move Extended Auxiliary Register (XAR) Content 5-247 Move Memory to Memory 5-248 Move Temporary Register Content 5-240 Move Temporary Register Content to Accumulator 5-242 Move Temporary Register Content to CPU Register 5-243 Multiply 5-255 Multiply and Accumulate (MAC) 5-271 Multiply and Accumulate with Parallel Delay 5-286 Multiply and Accumulate with Parallel Load Accumulator from Memory 5-288

Multiply and Accumulate with Parallel Multiply 5-290 Multiply and Accumulate with Parallel Store Accumulator Content to Memory 5-292 Multiply and Subtract 5-294 Multiply and Subtract with Parallel Load Accumulator from Memory 5-302 Multiply and Subtract with Parallel Multiply 5-304 Multiply and Subtract with Parallel Multiply and Accumulate 5-306 Multiply and Subtract with Parallel Store Accumulator Content to Memory 5-311 Multiply with Parallel Multiply and Accumulate 5-267 Multiply with Parallel Store Accumulator Content to Memory 5-269

# Ν

Negate Accumulator Content 5-313 Negate Auxiliary Register Content 5-313 Negate Temporary Register Content 5-313 negation accumulator content 5-313 auxiliary register content 5-313 temporary register content 5-313 No Operation (nop) 5-315 nonrepeatable instructions 1-20 nop 5-315



operand qualifier 5-328 OR 5-48



Parallel Modify Auxiliary Register Contents (mar) 5-316 Parallel Multiplies 5-317 Parallel Multiply and Accumulates 5-319 Parallel Multiply and Subtracts 5-326

Index-6

parallel operations addition with parallel store accumulator content to memory 5-29 load accumulator from memory with parallel store accumulator content to memory 5-185 modify auxiliary register content with parallel multiply 5-216 modify auxiliary register content with parallel multiply and accumulate 5-218 modify auxiliary register content with parallel multiply and subtract 5-223 modify auxiliary register contents 5-316 multiplies 5-317 multiply and accumulate with parallel delay 5-286 multiply and accumulate with parallel load accumulator from memory 5-288 multiply and accumulate with parallel multiply 5-290 multiply and accumulate with parallel store accumulator content to memory 5-292 multiply and accumulates 5-319 multiply and subtract with parallel load accumulator from memory 5-302 multiply and subtract with parallel multiply 5-304 multiply and subtract with parallel multiply and accumulate 5-306 multiply and subtract with parallel store accumulator content to memory 5-311 multiply and subtracts 5-326 multiply with parallel multiply and accumulate 5-267 multiply with parallel store accumulator content to memory 5-269 subtraction with parallel store accumulator content to memory 5-490 parallelism basics 2-3 parallelism features 2-2 Peripheral Port Register Access Qualifiers 5-328 pop 5-331 Pop Accumulator Content from Stack Pointers (popboth) 5-330

Pop Extended Auxiliary Register (XAR) Content from Stack Pointers (popboth) 5-330

Pop Top of Stack (pop) 5-331

popboth 5-330

program control branch conditionally 5-66 branch on auxiliary register not zero 5-74 branch unconditionally 5-70 call conditionally 5-77 call unconditionally 5-83 compare and branch 5-111 execute conditionally 5-159 idle 5-172 no operation 5-315 repeat block of instructions unconditionally 5-346 repeat single instruction conditionally 5-357 repeat single instruction unconditionally 5-360 repeat single instruction unconditionally and decrement CSR 5-365 repeat single instruction unconditionally and increment CSR 5-367 return conditionally 5-370 return from interrupt 5-374 return unconditionally 5-372 software interrupt 5-411 software reset 5-413 software trap 5-417 pshboth 5-338 push 5-339 Push Accumulator Content to Stack Pointers (pshboth) 5-338 Push Extended Auxiliary Register (XAR) Content to Stack Pointers (pshboth) 5-338 Push to Top of Stack (push) 5-339

## R

readport 5-328 register bit clear 5-88 complement (not) 5-128 set 5-384 test 5-504 test bit pair 5-506 repeat 5-360, 5-365, 5-367 Repeat Block of Instructions Unconditionally 5-346 Repeat Single Instruction Conditionally (while repeat) 5-357 Repeat Single Instruction Unconditionally (repeat) 5-360 Repeat Single Instruction Unconditionally and Decrement CSR (repeat) 5-365 Repeat Single Instruction Unconditionally and Increment CSR (repeat) 5-367 reset 5-413 resource conflicts in a parallel pair 2-4 return 5-372 Return Conditionally (if return) 5-370 Return from Interrupt (return\_int) 5-374 Return Unconditionally (return) 5-372 return int 5-374 rnd 5-380 Rotate Left Accumulator Content 5-376 Rotate Left Auxiliary Register Content 5-376 Rotate Left Temporary Register Content 5-376 Rotate Right Accumulator Content 5-378 Rotate Right Auxiliary Register Content 5-378 Rotate Right Temporary Register Content 5-378 Round Accumulator Content (rnd) 5-380 rounding 5-380

# S

saturate 5-382 Saturate Accumulator Content (saturate) 5-382 set accumulator bit 5-384 auxiliary register bit 5-384 memory bit 5-385 status register bit 5-386 temporary register bit 5-384 Set Accumulator Bit 5-384 Set Auxiliary Register Bit 5-384 Set Memory Bit 5-385 Set Status Register Bit 5-386 Set Temporary Register Bit 5-384 sftc 5-389 Shift Accumulator Content Conditionally (sftc) 5-389 Shift Accumulator Content Logically 5-391, 5-394 Shift Auxiliary Register Content Logically 5-394 shift conditionally 5-389 shift logically 5-391, 5-394 Shift Temporary Register Content Logically 5-394 Signed Shift of Accumulator Content 5-397, 5-406 Signed Shift of Auxiliary Register Content 5-406 Signed Shift of Temporary Register Content 5-406 soft-dual parallelism 2-5 Software Interrupt (intr) 5-411 Software Reset (reset) 5-413 Software Trap (trap) 5-417 sqdst 5-428 Square 5-419 Square and Accumulate 5-422 Square and Subtract 5-425 Square Distance (sqdst) 5-428 status register bit clear 5-90 set 5-386 store accumulator content to memory 5-430 accumulator pair content to memory 5-450 accumulator, auxiliary, or temporary register content to memory 5-453 auxiliary or temporary register pair content to memory 5-457 CPU register content to memory 5-458 extended auxiliary register (XAR) to memory 5-462 Store Accumulator Content to Memory 5-430, 5-453 Store Accumulator Pair Content to Memory 5-450 Store Auxiliary Register Content to Memory 5-453 Store Auxiliary Register Pair Content to Memory 5-457 Store CPU Register Content to Memory 5-458 Store Extended Auxiliary Register (XAR) to Memory 5-462 Store Temporary Register Content to Memory 5-453 Store Temporary Register Pair Content to Memory 5-457 subc 5-463 Subtract Conditionally 5-463 Subtraction 5-465 Subtraction with Parallel Store Accumulator Content to Memory 5-490 swap 5-492, 5-493, 5-494, 5-495, 5-496, 5-498, 5-500, 5-502, 5-503 Swap Accumulator Content (swap) 5-492 Swap Accumulator Pair Content (swap) 5-493 Swap Auxiliary and Temporary Register Content (swap) 5-496 Swap Auxiliary and Temporary Register Pair Content (swap) 5-498

Swap Auxiliary and Temporary Register Pairs Content (swap) 5-500
Swap Auxiliary Register Content (swap) 5-494
Swap Auxiliary Register Pair Content (swap) 5-495
Swap Temporary Register Content (swap) 5-502
Swap Temporary Register Pair Content (swap) 5-503
Symmetrical Finite Impulse Response Filter (firs) 5-170



test

accumulator bit 5-504 accumulator bit pair 5-506 auxiliary register bit 5-504 auxiliary register bit pair 5-506 memory bit 5-508 temporary register bit 5-504 temporary register bit pair 5-506 Test Accumulator Bit 5-504 Test Accumulator Bit Pair 5-506 Test and Clear Memory Bit 5-511 Test and Complement Memory Bit 5-512 Test and Set Memory Bit 5-513 Test Auxiliary Register Bit 5-504 Test Auxiliary Register Bit Pair 5-506 Test Memory Bit 5-508 Test Temporary Register Bit 5-504 Test Temporary Register Bit Pair 5-506 trap 5-417



unconditional branch 5-70 call 5-83 repeat block of instructions 5-346 repeat single instruction 5-360 repeat single instruction and decrement CSR 5-365 repeat single instruction and increment CSR 5-367 return 5-372 return from interrupt 5-374



while repeat 5-357 writeport 5-328



XOR 5-57