TMS320C55x DSP Mnemonic Instruction Set Reference Guide

Literature Number: SPRU374E
April 2001

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

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

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

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. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Resale of TI's products or services with *statements different from or beyond the parameters* stated by TI for that products or service voids all express and any implied warranties for the associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products.
www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

Preface

Read This First

About This Manual

The TMS320C55x™ DSP is a fixed-point digital signal processor (DSP) in the TMS320™ 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 mnemonic form of the instruction set. It contains information about the instructions used for all types of operations (arithmetical, bit manipulation, logical, move, and program control).

Related Documentation From Texas Instruments

The following books describe the C55x™ 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 DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU. This book also describes how to make individual portions of the DSP inactive to save power.

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 DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the algebraic instructions individually. It also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x Optimizing C Compiler User's Guide (literature number SPRU281) describes the 'C55x 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 devices.

TMS320C55x DSP Programmer's Reference Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and includes application program examples.

Code Composer Studio™ contains the following online guides:

TMS320C55x DSP Instruction Sets Online Guide describes the algebraic and mnemonic instructions individually. It also includes the parallelism features and rules of the instruction sets , a summary of the instruction sets, a list of the instruction opcodes, and a cross-reference of the mnemonic instruction sets.

TMS320C55x DSP Registers Online Guide describes the registers inside the TMS320C55x DSPs and shows the addresses for DSP registers that are mapped to memory.

TMS320C55x DSP CPU Online Guide describes the architecture and operation of the CPU inside the TMS320C55x DSPs. This guide also describes how to make individual portions of the DSP inactive to save power.

Trademarks

Contents

Tables

Chapter 1

Terms, Symbols, and Abbreviations

This chapter lists and defines the terms, symbols, and abbreviations used in the TMS320C55x™ DSP mnemonic instruction set summary and in the individual instruction descriptions. Also provided are instruction set notes and rules.

Topic **Page**

1.1 Instruction Set Terms, Symbols, and Abbreviations

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

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

Symbol	Meaning
[ ]	Optional operands
40	If the optional 40 keyword is applied to the instruction, the instruction provides the option to locally set M40 to 1 for the execution of the instruction
ACOVx	Accumulator overflow status bit: ACOV0, ACOV1, ACOV2, ACOV3
ACw, ACx, ACy, ACz	Accumulator: AC0, AC1, AC2, AC3
ARn_mod	Content of selected auxiliary register (ARn) is premodified (see Table 1–3) or postmodified (see Table 1–4) in the address generation unit.
ARx, ARy	Auxiliary register: AR0, AR1, AR2, AR3, AR4, AR5, AR6, AR7
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
CARRY	Value of CARRY status bit
Cmem	Coefficient indirect operand referencing a 16-bit or 32-bit value in data space
cond	Condition based on accumulator (ACx) value, auxiliary register (ARx) value, temporary register (Tx) value, test control (TCx) flag, or CARRY status bit. See section 1.3.
CSR	Computed single-repeat register
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.
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
Dx	Data address label coded on x bits (absolute address)

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

Symbol	Meaning
kx	Unsigned constant coded on x bits
Kx	Signed constant coded on x bits
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)
Lmem	Long-word single data memory access (32-bit data access). Same legal inputs as Smem.
Parallel Enable Bit	Indicates if the instruction contains a parallel enable bit.
Pipeline	Limiting execution pipeline phase: D Decode AD Address R Read X Execute
pmad	Program memory address
Px	Program or data address label coded on x bits (absolute address)
RELOP	Relational operators: == equal to < less than >= greater than or equal to != not equal to
R or rnd	If the optional R or rnd keyword is applied to the instruction, rounding is performed in the instruction
RPTC	Single-repeat counter register
saturate	If the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated
SHFT	Immediate shift value, 0 to 15
SHIFTW	Immediate shift value, –32 to +31
Size	Instruction size in bytes.
Smem	Word single data memory access (16-bit data access)
SP	Data stack pointer

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

Symbol	Meaning
src	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
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
TCx, TCy	Test control flag: TC1, TC2
TRNx	Transition register: TRN0, TRN1
Tx, Ty	Temporary register: T0, T1, T2, T3
U or uns	If the optional U or uns keyword is applied to the input operand, the operand is zero extended
XAdst	Destination 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): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XARx	All 23 bits of auxiliary register: XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
XAsrc	Source 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): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
xdst	Accumulator: AC0, AC1, AC2, AC3
	Destination 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): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7

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

Symbol	Meaning
xsrc	Accumulator: AC0, AC1, AC2, AC3
	Source 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): XAR0, XAR1, XAR2, XAR3, XAR4, XAR5, XAR6, XAR7
Xmem, Ymem	Indirect dual data memory access (two data accesses)

Table 1–2. Operators Used in Instruction Set

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
|			Bitwise OR	Left to right
^			Bitwise exclusive OR (XOR)	Left to right

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

1.2 Auxiliary Register Modifications

Table 1–3 lists the available premodifications and Table 1–4 lists the available postmodifications to the auxiliary registers.

Table 1–3. Auxiliary Register Premodifications

Operand	Modification
*+ARn	ARn is incremented by 1 before the address is generated: ARn = ARn + 1
*–ARn	ARn is decremented by 1 before the address is generated: ARn = ARn – 1
*ARn(AR0)	ARn is not modified. ARn is used as a base pointer. AR0 is used as an offset from that base pointer.
	See Compatible mode caution, section 1.2.1.
*ARn(T0)	ARn is not modified. ARn is used as a base pointer. T0 is used as an offset from that base pointer.
	See Enhanced mode caution, section 1.2.2.
*ARn(T1)	ARn is not modified. ARn is used as a base pointer. T1 is used as an offset from that base pointer.
*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 base pointer.
	See instruction size note, section 1.2.3.
*+ARn(#K16)	The 16-bit signed constant (K16) is added to ARn before the address is generated: ARn = ARn + K16
	See instruction size note, section 1.2.3.
*ARn(short(#k3))	ARn is not modified. ARn is used as a base pointer. The 3-bit constant (k3) is used as an offset from that base pointer. k3 is in the range 1 to 7.
*CDP(#K16)	CDP is not modified. CDP is used as a base pointer. The 16-bit signed constant (K16) is used as an offset from that base pointer.
	See instruction size note, section 1.2.3.
*+CDP(#K16)	The 16-bit signed constant (K16) is added to CDP before the address is generated: CDP = CDP + K16
	See instruction size note, section 1.2.3.

Table 1–4. Auxiliary Register Postmodifications

Operand	Modification
*ARn+	ARn is incremented by 1 after the address is generated: ARn = ARn + 1
*ARn–	ARn is decremented by 1 after the address is generated: ARn = ARn – 1
*(ARn + AR0)	AR0 is added to ARn after the address is generated: ARn = ARn + AR0
	See Compatible mode caution, section 1.2.1.
*(ARn + T0)	T0 is added to ARn after the address is generated: ARn = ARn + T0
	See Enhanced mode caution, section 1.2.2.
*(ARn – AR0)	AR0 is subtracted from ARn after the address is generated: ARn = ARn – AR0
	See Compatible mode caution, section 1.2.1.
*(ARn – T0)	T0 is subtracted from ARn after the address is generated: ARn = ARn – T0
	See Enhanced mode caution, section 1.2.2.
*(ARn + T0B)	T0 is added to ARn after the address is generated: ARn = ARn + T0 (The addition is done with reverse carry propagation.)
	This operand cannot be used for circular addressing.
*(ARn – T0B)	T0 is subtracted from ARn after the address is generated: ARn = ARn – T0 (The subtraction is done with reverse carry propagation.)
	This operand cannot be used for circular addressing.
*(ARn + T1)	T1 is added to ARn after the address is generated: ARn = ARn + T1
*(ARn – T1)	T1 is subtracted from ARn after the address is generated: ARn = ARn – T1
*CDP+	CDP is incremented by 1 after the address is generated: CDP = CDP + 1
*CDP–	CDP is decremented by 1 after the address is generated: CDP = CDP – 1

1.2.1 Compatible Mode Caution

- ❑ This modifier is available when C54CM = 1
- ❑ This modifier is usable when .c54cm_on is active at assembly time

1.2.2 Enhanced Mode Caution

- ❑ This modifier is available when C54CM = 0
- ❑ This modifier is usable when .c54cm_off is active at assembly time

1.2.3 Auxiliary Register Premodification Instruction Size 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.

1.3 Instruction Set Conditional (cond) Field

The following paragraphs provide the testing conditions available in the cond field of the conditional instructions.

1.3.1 Accumulator Content

The available conditions testing the accumulator content against 0:

ACx == #0 (the content is equal to 0)	ACx != #0 (the content is not equal to 0)
ACx < #0 (the content is less than 0)	ACx <= #0 (the content is less than or equal to 0)
ACx > #0 (the content is greater than 0)	ACx >= #0 (the content is greater than or equal to 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.

1.3.2 Accumulator Overflow Status Bit (ACOVx)

The available conditions testing 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.

overflow(ACx)	!overflow(ACx)

When these conditions are used, the corresponding accumulator overflow status bit is cleared to 0.

1.3.3 Auxiliary Register (ARx) Content

The available conditions testing the auxiliary register (ARx) content against 0:

*ARx == #0 (the content is equal to 0)	*ARx != #0 (the content is not equal to 0)
*ARx < #0 (the content is less than 0)	*ARx <= #0 (the content is less than or equal to 0)
*ARx > #0 (the content is greater than 0)	*ARx >= #0 (the content is greater than or equal to 0)

1.3.4 CARRY Status Bit

The available conditions testing the CARRY status bit against 1; when the optional ! symbol is used before the bit designation, the bit can be tested against 0.

CARRY	!CARRY

1.3.5 Temporary Register (Tx) Content

The available conditions testing the temporary register (Tx) content against 0:

Tx == #0 (the content is equal to 0)	Tx != #0 (the content is not equal to 0)
Tx < #0 (the content is less than 0)	Tx <= #0 (the content is less than or equal to 0)
Tx > #0 (the content is greater than 0)	Tx >= #0 (the content is greater than or equal to 0)

1.3.6 Test Control Flags, TC1 and TC2

The available conditions testing the test control flags (TC1 and TC2). Each of the bits can be tested independently against 1; when the optional ! symbol is used before the bit designation, the bits can be tested independently against 0.

TCx	!TCx

TC1 and TC2 can be combined with an AND (&), OR (|), and XOR (^) logical bit combinations:

TC1 & TC2	TC1 & !TC2
!TC1 & TC2	!TC1 & !TC2
TC1 | TC2	TC1 | !TC2
!TC1 | TC2	!TC1 | !TC2
TC1 ^ TC2	TC1 ^ !TC2
!TC1 ^ TC2	!TC1 ^ !TC2

1.4 Affect of Status Bits

1.4.1 Accumulator Overflow Status Bit (ACOVx)

The ACOV[0–3] depends on M40:

- When M40 = 0, overflow is detected at bit position 31
- 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.

1.4.2 C54CM Status Bit

- When C54CM = 0, the enhanced mode, the CPU supports code originally developed for a TMS320C55x™ 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™ DSP.

1.4.3 CARRY Status Bit

- 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.

1.4.4 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.

1.4.5 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).

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

1.4.6 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™ 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)

1.4.6.1 M40 Status Bit When Sign Shifting

In D-unit shifter:

- 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
- 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
- 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™ DSP compatibility mode

❑ 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)

1.4.6.2 M40 Status Bit When Logically Shifting

In D-unit shifter:

❑ 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

❑ 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:

❑ 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

- 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

1.4.7 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^{15}) 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.
```

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

1.4.8 SATA Status Bit

This status bit controls operations performed in the A unit.

- 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).

1.4.9 SATD Status Bit

This status bit controls operations performed in the D unit.

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

1.4.10 SMUL Status Bit

- 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.

1.4.11 SXMD Status Bit

This status bit controls operations performed in the D unit.

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

1.4.12 Test Control Status Bit (TCx)

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

1.5 Instruction Set Notes and Rules

1.5.1 Notes

- Mnemonic 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.

1.5.2 Rules

- Simple instructions are not allowed to span multiple lines. One exception, single instructions that use the double colons, ::, 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:

```
MPYR40 uns(Xmem), uns(Cmem), ACx
:: MPYR40 uns(Ymem), uns(Cmem), ACy
```

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

```
MOV AC0, AC1 || MOV AC2, AC3

MOV AC0, AC1 ||
MOV AC2, AC3

MOV AC0, AC1
|| MOV AC2, AC3

MOV AC0, AC1
||
MOV AC2, AC3
```

1.5.2.1 Reserved Words

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

1.5.2.2 Mnemonic Syntax Roots

Root	Meaning
ABS	Absolute value
ADD	Addition
AND	Bitwise AND
B	Branch
CALL	Function call
CLR	Assign the value to 0
CMP	Compare
CNT	Count
EXP	Exponent
MAC	Multiply and accumulate
MAR	Modify auxiliary register content
MAS	Multiply and subtract
MAX	Maximum
MIN	Minimum
MOV	Move data
MPY	Multiply
NEG	Negate (2s complement)
NOT	Bitwise complement (1s complement)
OR	Bitwise OR
POP	Pop from top of the stack
PSH	Push to top of the stack
RET	Return
ROL	Rotate left
ROR	Rotate right
RPT	Repeat
SAT	Saturate
SET	Assign the value to 1
SFT	Shift (left or right depending on sign of shift count)
SQA	Square and add
SQR	Square

SQS	Square and subtract
SUB	Subtraction
SWAP	Swap register contents
TST	Test bit
XOR	Bitwise XOR
XPA	Expand
XTR	Extract

1.5.2.3 Mnemonic Syntax Prefixes

Prefix	Meaning
A	Instruction happens in address phase and is subject to circular addressing effects. Also, it occurs in the DAGEN functional unit and cannot be placed in parallel with any instruction that uses dual addressing mode.
B	Bit instruction. Note that B is also a root (branch), suffix (borrow), and prefix (bit). The differences in context should prevent any confusion.

1.5.2.4 Mnemonic Syntax Suffixes

Suffixes can be combined. For the multiply variant instructions, the combination order is: M K R {40, A, Z, or U}. This list does not imply that all of the suffixes will ever be combined at once; but, when they are combined, they will be in this order.

Suffix	Meaning
40	Enables the M40 mode (all 40 bits of the accumulator count)
B	Borrow
C	Carry
CC	Conditional
I	Enable interrupts
K	Multiply has a constant operand
L	Logical shift (left or right depending on sign of shift count)
M	This instruction has the option of assigning a memory operand to T3; regardless of whether that assignment actually occurs.
R	Round
S	Signed shift (left or right depending on sign of shift count)
U	Unsigned
V	Absolute value
Z	Delay on the memory operand

1.5.2.5 Literal and Address Operands

Literals in the mnemonic 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 mnemonic strings denoted by ‘P’, ‘L’, and ‘l’. 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 this syntax rule:

Rule 1

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

- a number (`#123`)
- an identifier (`#F00`)
- a parenthesized expression (`#(F00 + 2)`)

  Note that ‘#’ is not used inside the expression

For addresses only, rules 2 and 3 also apply.

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 Smem-absolute address), addresses generally need a leading '#'. As a convenience, the '#' may be omitted in front of an identifier. These are all legal:

Branch	Absolute Address
`goto #123`	`*(#123)`
`goto #foo`	`*(#foo)`
`goto foo`	`*(foo)`
`goto #(foo+2)`	`*(#(foo+2))`

These are illegal:

`goto 123`	`*(123)`
`goto (foo+2)`	`*((foo+2))`

1.5.2.6 Memory Operands

- Syntax of Smem is the same as that of Lmem or Baddr.
- 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:

  ```
  MAC *AR2+, *CDP+, AC0
  :: MAC *AR3+, *CDP+, AC1
  ```

- 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() 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.

1.5.2.7 Operand Modifiers

Operand modifiers look like function calls on operands. Note that uns is an operand modifier meaning unsigned and that the instruction suffix U also means unsigned. The operand modifier uns is used when the operand is modified on the way to the rest of the operation (MAC). The instruction suffix U is used when the whole operation is affected (MPYMU, CMPU, BCCU).

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 (not used in MOV instructions)

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).

Chapter 2

Parallelism Features and Rules

This chapter describes the parallelism features and rules of the TMS320C55x™ DSP mnemonic instruction set.

Topic **Page**

2.1 Parallelism Features

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

- Built-in parallelism within a single instruction.

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

  ```
  MPY *AR0, *CDP, AC0
  :: MPY *AR1, *CDP, AC1
  ```

  This is a single instruction. The data referenced by AR0 is multiplied by the coefficient referenced by CDP. At the same time, the data referenced by AR1 is multiplied by the same coefficient (CDP).

- 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:

  ```
  MPYM *AR1–, *CDP, AC1
  || XOR AR2, T1
  ```

  The first instruction performs a multiplication in the D-unit. The second instruction performs a logical operation in the A-unit ALU.

- Built-in parallelism can be combined with user-defined parallelism. For example:

  ```
  MPYM T3=*AR3+, AC1, AC2
  || MOV #5, AR1
  ```

  The first instruction includes implied parallelism. The second instruction is paralleled by you.

2.2 Parallelism Rules

Parallelism between two instructions and only two instructions is allowed if all the rules are respected. The execution of a forbidden paralleled pair is not assured although the device is designed to execute a NOP (no operation) instruction.

2.2.1 Rule 1: Instruction length less than 6 bytes

Two instructions can be assembled in parallel if the added length of the instructions does not exceed 48 bits (6 bytes).

2.2.2 Rule 2: Instruction set support for parallelism

Two instructions can be assembled in parallel:

- ❑ If one of the two instructions is provided with a parallel enable bit. The hardware support for such a type of parallelism is called parallel enable mechanism.
- ❑ If both of the instructions make single data memory accesses (Smem only, or dbl(Lmem) only) in indirect addressing mode as specified in Rule 8 (Instructions with Smem Operands and Instructions with dbl(Lmem) Operands). The hardware support for such a type of parallelism is called soft dual mechanism.

2.2.3 Rule 3: Hardware resource conflicts

Two instructions can be paralleled if the memory bus, cross unit bus, and constant bus do not compete for access.

2.2.4 Rule 4: Parallelism between the A-unit, the D-unit, and the P-unit

Parallelism between the three main computation units of the device is allowed without restriction. An operation executed within a single unit can be paralleled with a second operation executed in one of the two other computation units.

2.2.5 Rule 5: Parallelism within the P-unit

The device allows for any parallelism between the following subunits:

- ❑ the P-unit load path
- ❑ the P-unit store path
- ❑ the P-unit control operators

The following summarizes parallelism between the subunits:

Example	Instruction Types
`MOV #3, BRC1` `|| MOV BRC0, T1`	P-unit load and P-unit store
`MOV @variable, BRC1` `|| BCC #label, AC0 >= #0`	P-unit load and P-unit control operator
`MOV BRC1, T1` `|| RPT #5`	P-unit store and P-unit control operator

In addition to the previous parallelism combinations, the device allows for parallelism within the P-unit:

- ❑ two load operations in parallel with the P-unit
- ❑ two store operations in parallel with the P-unit

The following summarizes parallelism within the P-unit:

Example	Instruction Types
`MOV #4, BRC1` `|| MOV T1, BRC0`	Two P-unit loads
`MOV BRC0, *AR3` `|| MOV BRC1, *AR5`	Two P-unit stores

2.2.6 Rule 6: Parallelism within the D-Unit

The device allows for any parallelism between the following subunits:

- ❑ the D-unit load path
- ❑ the D-unit store path
- ❑ the D-unit swap operator
- ❑ the D-unit ALU, shifter, DMAC operators (all considered a single operator)
- ❑ the D-unit shift and store path

Parallelism between the ALU, shifter, and DMAC is not allowed. The following table summarizes parallelism between the subunits:

Example	Instruction Types
`MOV #3, AC1` `|| MOV AC2, dbl(*AR4)`	D-unit load and D-unit store
`MOV @variable, AC1` `|| SWAP AC0, AC2`	D-unit load and D-unit swap
`MOV @variable << #16, AC1` `|| MOV AC1, AC3`	D-unit load and D-unit ALU
`MOV #3 << #16, AC1` `|| MPY T1, AC3`	D-unit load and D-unit MAC
`MOV @variable, AC1` `|| SFTS AC1, #2, AC3`	D-unit load and D-unit shifter
`MOV *AR1, AC1` `|| MOV HI(AC1 << #3), *AR1`	D-unit load and D-unit shift and store
`MOV HI(AC1), @variable` `|| MOV AC1, AC3`	D-unit store and D-unit ALU
`MOV pair(HI(AC0)), dbl(@variable)` `|| MPY T1, AC3`	D-unit store and D-unit MAC
`MOV AC1, @variable` `|| SFTS AC1, T2, AC3`	D-unit store and D-unit shifter
`MOV AC1, *AR2` `|| MOV HI(AC1 << #3), *AR1`	D-unit store and D-unit shift and store
`SWAP AC0, AC2` `|| MOV AC1, AC3`	D-unit swap and D-unit ALU
`SWAP AC0, AC2` `|| MPY T1, AC3`	D-unit swap and D-unit MAC
`SWAP AC1, AC3` `|| SFTS AC1, #2, AC2`	D-unit swap and D-unit shifter
`SWAPP AC0, AC2` `|| MOV HI(AC1 << T2), *AR1`	D-unit swap and D-unit shift and store
`AND *AR2, AC1, AC3` `|| MOV HI(AC1 << T2), *AR1`	D-unit ALU and D-unit shift and store
`AND T1, AC3` `|| MOV rnd(HI(AC1 << #3)), *AR1`	D-unit MAC and D-unit shift and store

Considerations for the D-unit shift and store path:

- ❑ D-unit shift and store operations are not allowed in parallel with other instructions using the D-unit shifter.
- ❑ A maximum of two accumulators can be selected as source operands of the instructions to be executed in parallel within the D-unit.

In addition to the previous parallelism combinations, the device allows for parallelism within the D-unit:

- ❑ two load operations in parallel with the D-unit
- ❑ two store operations in parallel with the D-unit

The following summarizes parallelism within the D-unit:

Example	Instruction Types
`MOV *AR3, AC1` `|| MOV *AR4 << #16, AC2`	Two D-unit loads
`MOV AC1, *AR2` `|| MOV AC2, *AR4`	Two D-unit stores

2.2.7 Rule 7: Parallelism within the A-unit (excluding the data address generation units)

Excluding X, Y, C, and SP data address generation unit operators, the device allows for any parallelism between the following subunits:

- ❑ the A-unit load path
- ❑ the A-unit store path
- ❑ the A-unit swap operator
- ❑ the A-unit ALU operator

Two A-unit ALU operations or two A-unit swap operations cannot be performed in parallel. The following table summarizes parallelism between the subunits:

Example	Instruction Types
`MOV #3, AR1` `|| MOV AR2, *AR4`	A-unit load and A-unit store
`MOV @variable, AR1` `|| MOV AC1, AR3`	A-unit load and A-unit ALU
`MOV #3, AR1` `|| ADD AR1, AR3`	A-unit load and A-unit ALU
`MOV @variable, AR1` `|| SWAPP T0, T2`	A-unit load and A-unit swap
`MOV AR1, @variable` `|| ADD AC1, AR3`	A-unit store and A-unit ALU
`MOV AR1, @variable` `|| SWAPP T0, T2`	A-unit store and A-unit swap
`AND *AR2, AR2, AR3` `|| SWAP4 AR4, T0`	A-unit ALU and A-unit swap

In addition to the previous parallelism combinations, the device allows for parallelism within the A-unit:

- ❑ two load operations in parallel with the A-unit.
- ❑ two store operations in parallel with the A-unit.

The following summarizes parallelism within the A-unit:

Example	Instruction Types
`MOV *AR3, AR1` `|| MOV *AR4, AR2`	Two A-unit loads
`MOV AR1, *AR3` `|| MOV AR2, *AR4`	Two A-unit stores

2.2.8 Rule 8: Parallelism within the A-unit data address generation unit

The data address generation unit (DAGEN) contains four operators:

- ❑ DAGEN X and DAGEN Y are the most generic of the operators, they allow you to generate any of the following addressing modes:
  - ■ Single data memory addressing, Smem or dbl(Lmem)
  - ■ Indirect dual data memory addressing (Xmem, Ymem)
  - ■ Coefficient data memory addressing, Cmem
  - ■ Register bit addressing, Baddr or pair(Baddr)

  DAGEN X and Y operators are also used to perform pointer modification with the MAR instructions.
- ❑ DAGEN C is a dedicated operator that is used for coefficient data memory addressing (Cmem).
- ❑ DAGEN SP is a dedicated operator that is used to address the data and system stacks.

The device enables you to parallel two instructions that each use the address generation unit to generate data memory or register bit addresses. This feature provides:

- ❑ The full bandwidth of the memory buses
- ❑ Memory-based instruction set flexibility

2.2.9 Rule 9: Modifier limitations

When the following addressing modifiers are used within one instruction, this instruction cannot be put in parallel with another instruction:

- *ARn(K16)
- *+ARn(K16)
- *CDP(K16)
- *+CDP(K16)
- *abs16(#k16)
- *(#k23)
- port(#k16)

2.2.10 Rule 10: Illegal program control instruction in paralleled instruction pair

The following control instructions cannot be executed in parallel with any instructions:

- `IDLE`
- `INTR k5`
- `RESET`
- `TRAP k5`

2.2.11 Rule 11: Illegal multicycle instruction in paralleled instruction pair

Multicycle instructions are allowed in a paralleled instruction pair combination as long as the other instruction is a single cycle instruction.

2.2.12 Rule 12: Parallelism with memory-mapped register and peripheral port register access qualifiers

An instruction that uses the mmap() qualifier to indicate an access to a memory-mapped register or registers cannot be placed in parallel with another instruction. The use of the mmap() qualifier is already a form of parallelism.

An instruction that uses the port() qualifier to indicate an access to an I/O space cannot be placed in parallel with another instruction. The use of the port() qualifier is already a form of parallelism.

2.2.13 Hardware Overwrite Rules

These hardware overwrite rules are applied within the device when the parallelism rules are not respected by the execution of an instruction. The assembler tool may generate warning messages instead of error messages for detectable errors.

2.2.13.1 Instruction constant priority

If two paralleled instructions have conflicting use of constants on the KAB or KDB buses, then the constant of the instruction encoded at the higher address (second instruction) is used for both instructions. An example of a conflict where both instructions of a paralleled pair use the KDB bus:

```
ADD #1, AC1
|| MOV #0xFFFF, T2
```

2.2.13.2 Register update priority in the Address pipeline phase

The auxiliary (ARx), temporary (Tx), and coefficient pointer (CDP) registers can be updated in the address pipeline phase:

- With operations performed in DAGEN X, Y, C (Smem, dbl(Lmem), Xmem, Ymem, Cmem, Baddr, and pair(Baddr) addressing modes, and mar() instructions).
- With SWAP instructions.

If two paralleled instructions have conflicting updates of a register, then the destination register is updated with the following priority:

- DAGEN X (highest priority)
- DAGEN Y
- DAGEN C
- SWAP instruction (lowest priority)

2.2.13.3 Register update priority in the Execute pipeline phase

If two paralleled instructions have two identical destination registers (or status bits), then the second instruction encoded at the higher address has priority over the first instruction. A similar mechanism exists for single instructions that have two destination registers that are identical.

2.2.13.4 Status bit update priority

If two paralleled instructions have conflicting updates of a status bit located in status registers ST0, ST1, ST2, or ST3, then the status bit is updated with the following priority:

- Operations performed in the D-unit. (highest priority)
- Operations performed in the A-unit.
- Memory-mapped register (MMR) access to ST0, ST1, ST2, and ST3.
- Operations performed in the P-unit (Reset due to condition evaluation by program control instructions). (lowest priority)

2.3 Instructions Using Single Data Memory Operands, Smem and dbl(Lmem)

The hardware support for this type of parallelism is called soft dual mechanism. Instructions having single data memory operands Smem can be paralleled with each other:

- ❑ if both instructions make indirect addressing to these memory operands
- ❑ if the modifiers used to modify the pointers are those allowed for indirect dual data memory addressing (Xmem, Ymem).

Some restrictions apply:

- ❑ Instructions embedding a Cmem (and Smem) operand can only be paralleled with instructions having an Smem read operand.
- ❑ The following instructions cannot be paralleled using the soft dual mechanism:
  - Instructions embedding high_byte(Smem) and low_byte(Smem).
    - `MOV [uns(]high_byte(Smem)[)], dst`
    - `MOV [uns(]low_byte(Smem)[)], dst`
    - `MOV low_byte(Smem) << #SHIFTW, ACx`
    - `MOV high_byte(Smem) << #SHIFTW, ACx`
    - `MOV src, high_byte(Smem)`
    - `MOV src, low_byte(Smem)`
  - Instructions embedding a delay, which moves the memory operands Smem to the next higher address.
    - `DELAY Smem`
    - `MACM[R]Z [T3 = ]Smem, Cmem, ACx`
  - Instructions reading a memory operand Smem, modifying it in the A or D-unit ALU, and storing it back to the same memory location.
    - `AND k16, Smem`
    - `OR k16, Smem`
    - `XOR k16, Smem`
    - `ADD K16, Smem`
    - `BNOT src, Smem`
    - `BCLR src, Smem`
    - `BSET src, Smem`
    - `BTSTSET k4, Smem, TCx`
    - `BTSTCLR k4, Smem, TCx`
    - `BTSTNOT k4, Smem, TCx`
  - Instructions performing memory-to-memory moves.
    - `MOV Cmem, Smem`
    - `MOV Smem, Cmem`

Instructions having single data memory operands dbl(Lmem) can be paralleled with each other:

- ❑ if both instructions make indirect addressing to these memory operands
- ❑ if the modifiers used to modify the pointers are those allowed for indirect dual data memory addressing (Xmem, Ymem).

2.4 Instructions with Xmem, Ymem, and Cmem Operands

Instructions having the following data memory operands cannot be paralleled with instructions using any of the four DAGEN operators:

- ❑ Indirect dual data memory addressing (Xmem, Ymem)
- ❑ Coefficient data memory addressing (Cmem) in some cases

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.
- ❑ and the Cmem operand is defined as unsigned, 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, Ymem operand must also be defined as unsigned (and reciprocally).

2.5 MAR Instruction

The following MAR (modify auxiliary register) instructions can be paralleled with each other:

- ❑ `AADD TAx, TAy`
- ❑ `ASUB TAx, TAy`
- ❑ `AMOV TAx, TAy`
- ❑ `AADD k8, TAx`
- ❑ `ASUB k8, TAx`
- ❑ `AMOV k8, TAx`

The MAR instruction can also be executed in parallel with instructions using the following addressing modes:

- ❑ Single data memory addressing, Smem or dbl(Lmem)
- ❑ Register bit addressing, Baddr or pair(Baddr)
- ❑ Data and system stack addressing instructions

2.6 Instructions Addressing the Data or System Stack

Instructions addressing the data or system stack cannot be paralleled with each other. These set of instructions include:

- ❏ all push to the top of stack (PSH) instructions
- ❏ all pop to the top of stack (POP) instructions
- ❏ all conditional call (CALLCC) and unconditional call (CALL) instructions
- ❏ all conditional return (RETCC) and unconditional return (RET) from subroutine instructions
- ❏ TRAP, INTR, and return from interrupt (RETI) instructions

Some instructions addressing the data or system stack can be paralleled with instructions using other DAGEN operators.

Any of the following instructions addressing the data or system stack (DAGEN SP operator) …	**can use any of the following instructions using only the DAGEN X, Y, or C operators**
`POP dst1, dst2`	an Smem or dbl(Lmem) write operand
`POP dst`	Xmem and Ymem write operands
`POP dbl(ACx)`	a Baddr or pair(Baddr) operand
`RETCC cond`	ARn_mod operand
`RET`	
`RETI`	

Any of the following instructions addressing the data or system stack (DAGEN SP operator) …	**can use any of the following instructions using only the DAGEN X, Y, or C operators**
`PSH src1, src2`	an Smem or dbl(Lmem) read operand
`PSH src`	Xmem and Ymem read operands
`PSH dbl(ACx)`	
`CALLCC #label, cond`	
`CALL #label`	
`CALL ACx`	

Chapter 3

Instruction Set Summary

The TMS320C55x™ DSP mnemonic instruction set can be divided into six basic types of operations:

- Arithmetical operations
- Bit manipulation operations
- Extended auxiliary register (XAR) operations
- Logical operations
- Move operations
- Program-control operations

In this chapter, each of the types of operations is divided into smaller groups of instructions with similar functions. With each instruction listing, you will find the availability of a parallel enable bit, word count (size), cycle time, what pipeline stage the instruction is executed, and in what unit the instruction is executed.

Topic **Page**

3.1 Arithmetical Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Absolute Distance (page 4-2)					
ABDST Xmem, Ymem, ACx, ACy	No	4	1	X	D unit MAC and ALU
Absolute Value (page 4-4)					
ABS [src,] dst	Yes	2	1	X	A or D unit ALU
Addition (page 4-70)					
ADD [src,] dst	Yes	2	1	X	A or D unit ALU
ADD k4, dst	Yes	2	1	X	A or D unit ALU
ADD K16, [src,] dst	No	4	1	X	A or D unit ALU
ADD Smem, [src,] dst	No	3	1	X	A or D unit ALU
ADD ACx << Tx, ACy	Yes	2	1	X	D unit ALU and shifter
ADD ACx << #SHIFTW, ACy	Yes	3	1	X	D unit ALU and shifter
ADD K16 << #16, [ACx,] ACy	No	4	1	X	D unit ALU
ADD K16 << #SHFT, [ACx,] ACy	No	4	1	X	D unit ALU and shifter
ADD Smem << Tx, [ACx,] ACy	No	3	1	X	D unit ALU and shifter
ADD Smem << #16, [ACx,] ACy	No	3	1	X	D unit ALU
ADD [uns(]Smem[)], CARRY, [ACx,] ACy	No	3	1	X	D unit ALU
ADD [uns(]Smem[)], [ACx,] ACy	No	3	1	X	D unit ALU
ADD [uns(]Smem[)] << #SHIFTW, [ACx,] ACy	No	4	1	X	D unit ALU and shifter
ADD dbl(Lmem), [ACx,] ACy	No	3	1	X	D unit ALU
ADD Xmem, Ymem, ACx	No	3	1	X	D unit ALU
ADD K16, Smem	No	4	1	X	D unit ALU
ADD[R]V [ACx,] ACy	Yes	2	1	X	D unit MAC

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Compare and Select Extremum (page 4-148)					D unit ALU
MAXDIFF ACx, ACy, ACz, ACw	Yes	3	1	X	
DMAXDIFF ACx, ACy, ACz, ACw, TRNx	Yes	3	1	X	
MINDIFF ACx, ACy, ACz, ACw	Yes	3	1	X	
DMINDIFF ACx, ACy, ACz, ACw, TRNx	Yes	3	1	X	
Conditional Addition/Subtraction (page 4-159)					
ADDSUBCC Smem, ACx, TC1, ACy	No	3	1	X	D unit ALU
ADDSUBCC Smem, ACx, TC2, ACy	No	3	1	X	D unit ALU
ADDSUBCC Smem, ACx, TC1, TC2, ACy	No	3	1	X	D unit ALU
ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy	No	3	1	X	D unit shifter
Conditional Shift (page 4-166)					
SFTCC ACx, TCx	Yes	2	1	X	D unit shifter
Conditional Subtract (page 4-167)					
SUBC Smem, [ACx,] ACy	No	3	1	X	D unit ALU
Dual 16-Bit Arithmetic (page 4-169)					D unit ALU
ADDSUB Tx, Smem, ACx	No	3	1	X	
SUBADD Tx, Smem, ACx	No	3	1	X	
ADD dual(Lmem), [ACx,] ACy	No	3	1	X	
SUB dual(Lmem), [ACx,] ACy	No	3	1	X	
SUB ACx, dual(Lmem), ACy	No	3	1	X	
SUB dual(Lmem), Tx, ACx	No	3	1	X	
ADD dual(Lmem), Tx, ACx	No	3	1	X	
SUB Tx, dual(Lmem), ACx	No	3	1	X	
ADDSUB Tx, dual(Lmem), ACx	No	3	1	X	
SUBADD Tx, dual(Lmem), ACx	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Dual Multiply (Accumulate/Subtract) (page 4-190)					D unit MACs
MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
AMAR Xmem :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	No	4	1	X	
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	No	4	1	X	
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
AMAR Xmem :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	No	4	1	X	
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X	
MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	No	4	1	X	
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	No	4	1	X	
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	No	4	1	X	
AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx >> #16	No	4	1	X	
AMAR Xmem, Ymem, Cmem	No	4	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Finite Impulse Response Filter, Symmetrical/Antisymmetrical (page 4-228)					D unit MAC and ALU
FIRSADD Xmem, Ymem, Cmem, ACx, ACy	No	4	1	X	
FIRSSUB Xmem, Ymem, Cmem, ACx, ACy	No	4	1	X	
Implied Paralleled Instructions (page 4-234)					
MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X	D unit MAC and shifter
MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X	D unit MAC and shifter
MASM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X	D unit MAC and shifter
ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	No	4	1	X	D unit ALU and shifter
SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	No	4	1	X	D unit ALU and shifter
MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X	D unit ALU and shifter
MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	No	4	1	X	D unit MAC
MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	No	4	1	X	D unit MAC
Least Mean Square (LMS) (page 4-251)					
LMS Xmem, Ymem, ACx, ACy	No	4	1	X	D unit MAC and ALU
Maximum Comparison (page 4-263)					
MAX [src,] dst	Yes	2	1	X	A or D unit ALU
Minimum Comparison (page 4-266)					
MIN [src,] dst	Yes	2	1	X	A or D unit ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Memory Comparison (page 4-279)					A unit ALU
CMP Smem == K16, TC1	No	4	1	X	
CMP Smem == K16, TC2	No	4	1	X	
Modify Auxiliary Register (MAR) (page 4-289)					A unit DAGENs
AADD TAx, TAy	Yes	3	1	AD	
ASUB TAx, TAy	Yes	3	1	AD	
AMOV TAx, TAy	Yes	3	1	AD	
AADD k8, TAx	Yes	3	1	AD	
ASUB k8, TAx	Yes	3	1	AD	
AMOV k8, TAx	Yes	3	1	AD	
AMOV D16, TAx	No	4	1	AD	
AMAR Smem	No	2	1	AD	
Modify Data Stack Pointer (SP) (page 4-299)					
AADD K8, SP	Yes	2	1	AD	A unit ALU
Multiply (page 4-300)					D unit MAC
SQR[R] [ACx,] ACy	Yes	2	1	X	
MPY[R] [ACx,] ACy	Yes	2	1	X	
MPY[R] Tx, [ACx,] ACy	Yes	2	1	X	
MPYK[R] K8, [ACx,] ACy	Yes	3	1	X	
MPYK[R] K16, [ACx,] ACy	No	4	1	X	
MPYM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X	
SQRM[R] [T3 =]Smem, ACx	No	3	1	X	
MPYM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X	
MPYMK[R] [T3 =]Smem, K8, ACx	No	4	1	X	
MPYM[R][40] [T3 =][uns()]Xmem[)], [uns()]Ymem[)], ACx	No	4	1	X	
MPYM[R][U] [T3 =]Smem, Tx, ACx	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Multiply and Accumulate (MAC) (page 4-315)					D unit MAC
SQA[R] [ACx,] ACy	Yes	2	1	X	
MAC[R] ACx, Tx, ACy[, ACy]	Yes	2	1	X	
MAC[R] ACy, Tx, ACx, ACy	Yes	2	1	X	
MACK[R] Tx, K8, [ACx,] ACy	Yes	3	1	X	
MACK[R] Tx, K16, [ACx,] ACy	No	4	1	X	
MACM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X	
MACM[R]Z [T3 =]Smem, Cmem, ACx	No	3	1	X	
SQAM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X	
MACM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X	
MACM[R] [T3 =]Smem, Tx, [ACx,] ACy	No	3	1	X	
MACMK[R] [T3 =]Smem, K8, [ACx,] ACy	No	4	1	X	
MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	No	4	1	X	
MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16[, ACy]	No	4	1	X	
Multiply and Subtract (MAS) (page 4-333)					D unit MAC
SQS[R] [ACx,] ACy	Yes	2	1	X	
MAS[R] Tx, [ACx,] ACy	Yes	2	1	X	
MASM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X	
SQSM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X	
MASM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X	
MASM[R] [T3 =]Smem, Tx, [ACx,] ACy	No	3	1	X	
MASM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	No	4	1	X	
Negation (page 4-344)					
NEG [src,] dst	Yes	2	1	X	A or D unit ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Normalization (page 4-347)					A unit ALU and D unit shifter
MANT ACx, ACy :: NEXP ACx, Tx	Yes	3	1	X	
EXP ACx, Tx	Yes	3	1	X	
Register Comparison (page 4-377)					A or D unit ALU
CMP[U] src RELOP dst, TCx	Yes	3	1	X	
CMPAND[U] src RELOP dst, TCy, TCx	Yes	3	1	X	
CMPAND[U] src RELOP dst, !TCy, TCx	Yes	3	1	X	
CMPOR[U] src RELOP dst, TCy, TCx	Yes	3	1	X	
CMPOR[U] src RELOP dst, !TCy, TCx	Yes	3	1	X	
Round (page 4-413)					
ROUND [ACx,] ACy	Yes	2	1	X	D unit ALU
Saturate (page 4-415)					
SAT[R] [ACx,] ACy	Yes	2	1	X	D unit ALU
Signed Shift (page 4-417)					
SFTS dst, #–1	Yes	2	1	X	A unit ALU or D unit shifter
SFTS dst, #1	Yes	2	1	X	A unit ALU or D unit shifter
SFTS ACx, Tx[, ACy]	Yes	2	1	X	D unit shifter
SFTSC ACx, Tx[, ACy]	Yes	2	1	X	D unit shifter
SFTS ACx, #SHIFTW[, ACy]	Yes	3	1	X	D unit shifter
SFTSC ACx, #SHIFTW[, ACy]	Yes	3	1	X	D unit shifter

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Square Distance (page 4-449)					
SQDST Xmem, Ymem, ACx, ACy	No	4	1	X	D unit MAC and ALU
Subtraction (page 4-457)					
SUB [src,] dst	Yes	2	1	X	A or D unit ALU
SUB k4, dst	Yes	2	1	X	A or D unit ALU
SUB K16, [src,] dst	No	4	1	X	A or D unit ALU
SUB Smem, [src,] dst	No	3	1	X	A or D unit ALU
SUB src, Smem, dst	No	3	1	X	A or D unit ALU
SUB ACx << Tx, ACy	Yes	2	1	X	D unit ALU and shifter
SUB ACx << #SHIFTW, ACy	Yes	3	1	X	D unit ALU and shifter
SUB K16 << #16, [ACx,] ACy	No	4	1	X	D unit ALU
SUB K16 << #SHFT, [ACx,] ACy	No	4	1	X	D unit ALU and shifter
SUB Smem << Tx, [ACx,] ACy	No	3	1	X	D unit ALU and shifter
SUB Smem << #16, [ACx,] ACy	No	3	1	X	D unit ALU
SUB ACx, Smem << #16, ACy	No	3	1	X	D unit ALU
SUB [uns(]Smem[)], BORROW, [ACx,] ACy	No	3	1	X	D unit ALU
SUB [uns(]Smem[)], [ACx,] ACy	No	3	1	X	D unit ALU
SUB [uns(]Smem[)] << #SHIFTW, [ACx,] ACy	No	4	1	X	D unit ALU and shifter
SUB dbl(Lmem), [ACx,] ACy	No	3	1	X	D unit ALU
SUB ACx, dbl(Lmem), ACy	No	3	1	X	D unit ALU
SUB Xmem, Ymem, ACx	No	3	1	X	D unit ALU

3.2 Bit Manipulation Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Bit Field Comparison (page 4-91)					A unit ALU
BAND Smem, k16, TC1	No	4	1	X	
BAND Smem, k16, TC2	No	4	1	X	
Bit Field Expand (page 4-93)					
BFXPA k16, ACx, dst	No	4	1	X	D unit shifter
Bit Field Extract (page 4-94)					
BFXTR k16, ACx, dst	No	4	1	X	D unit shifter
Memory Bit Test/Set/Clear/Complement (page 4-270)					A unit ALU
BTST src, Smem, TCx	No	3	1	X	
BNOT src, Smem	No	3	1	X	
BCLR src, Smem	No	3	1	X	
BSET src, Smem	No	3	1	X	
BTSTSET k4, Smem, TC1	No	3	1	X	
BTSTSET k4, Smem, TC2	No	3	1	X	
BTSTCLR k4, Smem, TC1	No	3	1	X	
BTSTCLR k4, Smem, TC2	No	3	1	X	
BTSTNOT k4, Smem, TC1	No	3	1	X	
BTSTNOT k4, Smem, TC2	No	3	1	X	
BTST k4, Smem, TC1	No	3	1	X	
BTST k4, Smem, TC2	No	3	1	X	
Register Bit Test/Set/Clear/Complement (page 4-370)					A or D unit ALU
BTST Baddr, src, TCx	No	3	1	X	
BNOT Baddr, src	No	3	1	X	
BCLR Baddr, src	No	3	1	X	
BSET Baddr, src	No	3	1	X	
BTSTP Baddr, src	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Status Bit Set/Clear (page 4-451)					A unit ALU
BCLR k4, ST0_55	Yes	2	1	X	
BSET k4, ST0_55	Yes	2	1	X	
BCLR k4, ST1_55	Yes	2	1	X	
BSET k4, ST1_55	Yes	2	1	X	
BCLR k4, ST2_55	Yes	2	1	X	
BSET k4, ST2_55	Yes	2	1	X	
BCLR k4, ST3_55	Yes	2	1†	X	
BSET k4, ST3_55	Yes	2	1†	X	
BCLR f–name	Yes	2	1†	X	
BSET f–name	Yes	2	1†	X	

† When these instructions are 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.

3.3 Extended Auxiliary Register (XAR) Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Extended Auxiliary Register Move (page 4-227)				
MOV xsrc, xdst	No	2	1	X
Load Effective Address to Extended Auxiliary Register (page 4-256)				
AMAR Smem, XAdst	No	3	1	AD
AMOV k23, XAdst	No	6	1	AD
Load Extended Auxiliary Register from Memory (page 4-257)				
MOV dbl(Lmem), XAdst	No	3	1	X
Pop Extended Auxiliary Register from Stack Pointers (page 4-354)				
POPBOTH xdst	Yes	2	1	X
Push Extended Auxiliary Register to Stack Pointers (page 4-362)				
PSHBOTH xsrc	Yes	2	1	X
Store Extended Auxiliary Register to Memory (page 4-456)				
MOV XAsrc, dbl(Lmem)	No	3	1	X

3.4 Logical Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Bit Field Counting (page 4-92)					
BCNT ACx, ACy,TCx, Tx	Yes	3	1	X	D unit shifter and A unit ALU
Bitwise Complement (page 4-95)					
NOT [src,] dst	Yes	2	1	X	A or D unit ALU
Bitwise AND (page 4-96)					
AND src, dst	Yes	2	1	X	A or D unit ALU
AND k8, src, dst	Yes	3	1	X	A or D unit ALU
AND k16, src, dst	No	4	1	X	A or D unit ALU
AND Smem, src, dst	No	3	1	X	A or D unit ALU
AND ACx << #SHIFTW[, ACy]	Yes	3	1	X	D unit shifter
AND k16 << #16, [ACx,] ACy	No	4	1	X	D unit ALU
AND k16 << #SHFT, [ACx,] ACy	No	4	1	X	D unit shifter
AND k16, Smem	No	4	1	X	A unit ALU
Bitwise OR (page 4-105)					
OR src, dst	Yes	2	1	X	A or D unit ALU
OR k8, src, dst	Yes	3	1	X	A or D unit ALU
OR k16, src, dst	No	4	1	X	A or D unit ALU
OR Smem, src, dst	No	3	1	X	A or D unit ALU
OR ACx << #SHIFTW[, ACy]	Yes	3	1	X	D unit shifter
OR k16 << #16, [ACx,] ACy	No	4	1	X	D unit ALU
OR k16 << #SHFT, [ACx,] ACy	No	4	1	X	D unit shifter
OR k16, Smem	No	4	1	X	A unit ALU

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Bitwise XOR (page 4-114)					
XOR src, dst	Yes	2	1	X	A or D unit ALU
XOR k8, src, dst	Yes	3	1	X	A or D unit ALU
XOR k16, src, dst	No	4	1	X	A or D unit ALU
XOR Smem, src, dst	No	3	1	X	A or D unit ALU
XOR ACx << #SHIFTW[, ACy]	Yes	3	1	X	D unit shifter
XOR k16 << #16, [ACx,] ACy	No	4	1	X	D unit ALU
XOR k16 << #SHFT, [ACx,] ACy	No	4	1	X	D unit shifter
XOR k16, Smem	No	4	1	X	A unit ALU
Logical Shift (page 4-258)					
SFTL dst, #1	Yes	2	1	X	A unit ALU or D unit shifter
SFTL dst, #–1	Yes	2	1	X	A unit ALU or D unit shifter
SFTL ACx, Tx[, ACy]	Yes	2	1	X	D unit shifter
SFTL ACx, #SHIFTW[, ACy]	Yes	3	1	X	D unit shifter
Negation (page 4-344)					
NEG [src,] dst	Yes	2	1	X	A or D unit ALU
Rotate Left (page 4-409)					
ROL BitOut, src, BitIn, dst	Yes	3	1	X	A unit ALU or D unit shifter
Rotate Right (page 4-411)					
ROR BitIn, src, BitOut, dst	Yes	3	1	X	A unit ALU or D unit shifter

3.5 Miscellaneous Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Linear/Circular Addressing Qualifiers (page 4-253)				
<instruction>.LR	No	1	1	AD
<instruction>.CR	No	1	1	AD
Memory-Mapped Register Access Qualifier (page 4-268)				
mmap	No	1	1	D
Peripheral Port Register Access Qualifier (page 4-352)				
port(Smem)	No	1	1	D

3.6 Move Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Accumulator, Auxiliary, or Temporary Register Content Swap (page 4-7)					
SWAP ARx, Tx	Yes	2	1	AD	A unit register file
SWAP Tx, Ty	Yes	2	1	AD	A unit register file
SWAP ARx, ARy	Yes	2	1	AD	A unit register file
SWAP ACx, ACy	Yes	2	1	X	D unit register file
SWAPP ARx, Tx	Yes	2	1	AD	A unit register file
SWAPP T0, T2	Yes	2	1	AD	A unit register file
SWAPP AR0, AR2	Yes	2	1	AD	A unit register file
SWAPP AC0, AC2	Yes	2	1	X	D unit register file
SWAP4 AR4, T0	Yes	2	1	AD	A unit register file
Accumulator, Auxiliary, or Temporary Register Load (page 4-19)					
MOV k4, dst	Yes	2	1	X	A or D unit register file
MOV –k4, dst	Yes	2	1	X	A or D unit register file
MOV K16, dst	No	4	1	X	A or D unit register file
MOV Smem, dst	No	2	1	X	A or D unit register file
MOV [uns()]high_byte(Smem)[)], dst	No	3	1	X	A or D unit register file
MOV [uns()]low_byte(Smem)[)], dst	No	3	1	X	A or D unit register file
MOV K16 << #16, ACx	No	4	1	X	D unit ALU
MOV K16 << #SHFT, ACx	No	4	1	X	D unit shifter
MOV [rnd()]Smem << Tx[)], ACx	No	3	1	X	D unit shifter

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
MOV low_byte(Smem) << #SHIFTW, ACx	No	3	1	X	D unit shifter
MOV high_byte(Smem) << #SHIFTW, ACx	No	3	1	X	D unit shifter
MOV Smem << #16, ACx	No	2	1	X	D unit ALU
MOV [uns(]Smem[)], ACx	No	3	1	X	D unit register file
MOV [uns(]Smem[)] << #SHIFTW, ACx	No	4	1	X	D unit shifter
MOV[40] dbl(Lmem), ACx	No	3	1	X	D unit register file
MOV Xmem, Ymem, ACx	No	3	1	1	D unit register file
MOV dbl(Lmem), pair(HI(ACx))	No	3	1	X	D unit register file
MOV dbl(Lmem), pair(LO(ACx))	No	3	1	X	D unit register file
MOV dbl(Lmem), pair(TAx)	No	3	1	X	A unit register file
Accumulator, Auxiliary, or Temporary Register Move (page 4-41)					
MOV src, dst	Yes	2	1	X	A or D unit ALU
MOV HI(ACx), TAx	Yes	2	1	X	A unit ALU
MOV TAx, HI(ACx)	Yes	2	1	X	D unit ALU
Accumulator, Auxiliary, or Temporary Register Store (page 4-46)					
MOV src, Smem	No	2	1	X	A or D unit register file
MOV src, high_byte(Smem)	No	3	1	X	A or D unit register file
MOV src, low_byte(Smem)	No	3	1	X	A or D unit register file
MOV HI(ACx), Smem	No	2	1	X	D unit register file
MOV [rnd(]HI(ACx)[)], Smem	No	3	1	X	D unit shifter
MOV ACx << Tx, Smem	No	3	1	X	D unit shifter
MOV [rnd(]HI(ACx << Tx)[)], Smem	No	3	1	X	D unit shifter
MOV ACx << #SHIFTW, Smem	No	3	1	X	D unit shifter
MOV HI(ACx << #SHIFTW), Smem	No	3	1	X	D unit shifter

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
MOV [rnd(]HI(ACx << #SHIFTW)[)], Smem	No	4	1	X	D unit shifter
MOV [uns(] [rnd(]HI(saturate(ACx))[))], Smem	No	3	1	X	D unit shifter
MOV [uns(] [rnd(]HI(saturate(ACx << Tx))[))], Smem	No	3	1	X	D unit shifter
MOV [uns(](rnd(]HI(saturate(ACx << #SHIFTW))[))], Smem	No	4	1	X	D unit shifter
MOV ACx, dbl(Lmem)	No	3	1	X	D unit register file
MOV [uns(]saturate(ACx)[)], dbl(Lmem)	No	3	1	X	D unit shifter
MOV ACx >> #1, dual(Lmem)	No	3	1	X	D unit register file
MOV pair(HI(ACx)), dbl(Lmem)	No	3	1	X	D unit register file
MOV pair(LO(ACx)), dbl(Lmem)	No	3	1	X	A unit register file
MOV pair(TAx), dbl(Lmem)	No	3	1	X	D unit shifter
MOV ACx, Xmem, Ymem	No	3	1	X	D unit register file
Memory Delay (page 4-280)					
DELAY Smem	No	2	1	X	A or D unit register file
Memory-to-Memory Move/Memory Initialization (page 4-281)					A or D unit register file
MOV Cmem, Smem	No	3	1	X	
MOV Smem, Cmem	No	3	1	X	
MOV K8, Smem	No	3	1	X	
MOV K16, Smem	No	4	1	X	
MOV Cmem,dbl(Lmem)	No	3	1	X	
MOV dbl(Lmem), Cmem	No	3	1	X	
MOV dbl(Xmem), dbl(Ymem)	No	3	1	X	
MOV Xmem, Ymem	No	3	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
Pop Top of Stack (TOS) (page 4-355)					A or D unit register file
POP dst1,dst2	Yes	2	1	X	
POP dst	Yes	2	1	X	
POP dst, Smem	No	3	1	X	
POP dbl(ACx)	Yes	2	1	X	
POP Smem	No	2	1	X	
POP dbl(Lmem)	No	2	1	X	
Push to Top of Stack (TOS) (page 4-363)					A or D unit register file
PSH src1,src2	Yes	2	1	X	
PSH src	Yes	2	1	X	
PSH src,Smem	No	3	1	X	
PSH dbl(ACx)	Yes	2	1	X	
PSH Smem	No	2	1	X	
PSH dbl(Lmem)	No	2	1	X	
Specific CPU Register Load (page 4-435)					A or D unit register file
MOV k12, BK03	Yes	3	1	AD	
MOV k12, BK47	Yes	3	1	AD	
MOV k12, BKC	Yes	3	1	AD	
MOV k12, BRC0	Yes	3	1	AD	
MOV k12, BRC1	Yes	3	1	AD	
MOV k12, CSR	Yes	3	1	AD	
MOV k7, DPH	Yes	3	1	AD	
MOV k9, PDP	Yes	3	1	AD	
MOV k16, BSA01	No	4	1	AD	
MOV k16, BSA23	No	4	1	AD	
MOV k16, BSA45	No	4	1	AD	
MOV k16, BSA67	No	4	1	AD	
MOV k16, BSAC	No	4	1	AD	
MOV k16, CDP	No	4	1	AD	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
MOV k16, DP	No	4	1	AD	A or D unit register file
MOV k16, SP	No	4	1	AD	
MOV k16, SSP	No	4	1	AD	
MOV Smem, BK03	No	3	1	X	
MOV Smem, BK47	No	3	1	X	
MOV Smem, BKC	No	3	1	X	
MOV Smem, BSA01	No	3	1	X	
MOV Smem, BSA23	No	3	1	X	
MOV Smem, BSA45	No	3	1	X	
MOV Smem, BSA67	No	3	1	X	
MOV Smem, BSAC	No	3	1	X	
MOV Smem, BRC0	No	3	1	X	
MOV Smem, BRC1	No	3	1	X	
MOV Smem, CDP	No	3	1	X	
MOV Smem, CSR	No	3	1	X	
MOV Smem, DP	No	3	1	X	
MOV Smem, DPH	No	3	1	X	
MOV Smem, PDP	No	3	1	X	
MOV Smem, SP	No	3	1	X	
MOV Smem, SSP	No	3	1	X	
MOV Smem, TRN0	No	3	1	X	
MOV Smem, TRN1	No	3	1	X	
MOV dbl(Lmem), RETA	No	3	5	X	
Specific CPU Register Move (page 4-441)					A unit ALU
MOV TAx, BRC0	Yes	2	1	X	
MOV TAx, BRC1	Yes	2	1	X	
MOV TAx, CDP	Yes	2	1	X	
MOV TAx, CSR	Yes	2	1	X	
MOV TAx, SP	Yes	2	1	X	
MOV TAx, SSP	Yes	2	1	X	
MOV BRC0, TAx	Yes	2	1	X	

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline	Executed
MOV BRC1, TAx	Yes	2	1	X	A unit ALU
MOV CDP, TAx	Yes	2	1	X	
MOV RPTC, TAx	Yes	2	1	X	
MOV SP, TAx	Yes	2	1	X	
MOV SSP, TAx	Yes	2	1	X	
Specific CPU Register Store (page 4-445)					A or D unit register file
MOV BK03, Smem	No	3	1	X	
MOV BK47, Smem	No	3	1	X	
MOV BKC, Smem	No	3	1	X	
MOV BSA01, Smem	No	3	1	X	
MOV BSA23, Smem	No	3	1	X	
MOV BSA45, Smem	No	3	1	X	
MOV BSA67, Smem	No	3	1	X	
MOV BSAC, Smem	No	3	1	X	
MOV BRC0, Smem	No	3	1	X	
MOV BRC1, Smem	No	3	1	X	
MOV CDP, Smem	No	3	1	X	
MOV CSR, Smem	No	3	1	X	
MOV DP, Smem	No	3	1	X	
MOV DPH, Smem	No	3	1	X	
MOV PDP, Smem	No	3	1	X	
MOV SP, Smem	No	3	1	X	
MOV SSP, Smem	No	3	1	X	
MOV TRN0, Smem	No	3	1	X	
MOV TRN1, Smem	No	3	1	X	
MOV RETA, dbl(Lmem)	No	3	5	X	

3.7 Program Control Operations

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Branch Conditionally (page 4-123)				
BCC l4, cond	No	2	6/5	R
BCC L8, cond	Yes	3	6/5	R
BCC L16, cond	No	4	6/5	R
BCC P24, cond	No	5	5/5	R
x/y cycles: x cycles = condition true, y cycles = condition false				
Branch Unconditionally (page 4-130)				
B ACx	No	2	10	X
B L7	Yes	2	6†	AD
B L16	Yes	3	6†	AD
B P24	No	4	5	D
† These instructions execute in 3 cycles if the addressed instruction is in the instruction buffer unit.				
Branch on Auxiliary Register Not Zero (page 4-134)				
BCC L16, ARn_mod ! = #0	No	4	6/5	AD
x/y cycles: x cycles = condition true, y cycles = condition false				
Call Conditionally (page 4-137)				
CALLCC L16, cond	No	4	6/5	R
CALLCC P24, cond	No	5	5/5	R
x/y cycles: x cycles = condition true, y cycles = condition false				
Call Unconditionally (page 4-142)				
CALL ACx	No	2	10	X
CALL L16	Yes	3	6	AD
CALL P24	No	4	5	D
Compare and Branch (page 4-146)				
BCC[U] L8, src RELOP K8	No	4	7/6	X
x/y cycles: x cycles = condition true, y cycles = condition false				

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Execute Conditionally (page 4-221)				
XCC [label,]cond	No	2	1	AD
XCCPART [label,]cond	No	2	1	X
Idle (page 4-233)				
IDLE	No	4	?	D
No Operation (NOP) (page 4-346)				
NOP	Yes	1	1	D
NOP_16	Yes	2	1	D
Repeat Block of Instructions Unconditionally (page 4-388)				
RPTBLOCAL pmad	Yes	2	1	AD
RPTB pmad	Yes	3	1	AD
Repeat Single Instruction Conditionally (page 4-394)				
RPTCC k8, cond	Yes	3	1	AD
Repeat Single Instruction Unconditionally (page 4-397)				
RPT CSR	Yes	2	1	AD
RPTADD CSR, TAx	Yes	2	1	X
RPT k8	Yes	2	1	AD
RPTADD CSR, k4	Yes	2	1	X
RPTSUB CSR, k4	Yes	2	1	X
RPT k16	Yes	3	1	AD
Return Conditionally (page 4-405)				
RETCC cond	Yes	3	5/5	R
x/y cycles: x cycles = condition true, y cycles = condition false				
Return Unconditionally (page 4-407)				
RET	Yes	2	5	D

Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
Return from Interrupt (page 4-408)				
RETI	Yes	2	5	D
Software Interrupt (page 4-430)				
INTR k5	No	2	3	D
Software Reset (page 4-432)				
RESET	No	2	?	D
Software Trap (page 4-433)				
TRAP k5	No	2	?	D

Chapter 4

Instruction Set Descriptions

This chapter provides detailed information on the TMS320C55x™ DSP mnemonic 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 3 for a summary of the instruction set. See Chapter 5 for the opcode of each instruction syntax.

4.1 Absolute Distance (ABDST)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ABDST Xmem,Ymem,ACx,ACy	No	4	1	X

Operands: ACx, ACy, Xmem, Ymem

Description

This instruction executes two operations in parallel: one in the D-unit MAC and one in the D-unit ALU:

```
ACy = ACy + |HI(ACx)|
ACx = (Xmem << #16) – (Ymem << #16)
```

The absolute value of accumulator ACx content is computed and added to accumulator ACy content through the D-unit MAC. When an overflow is detected according to M40:

- ❏ the destination accumulator overflow status bit (ACOVy) is set
- ❏ the destination register (ACy) is saturated according to SATD

The Ymem content shifted left 16 bits is subtracted from the Xmem content shifted left 16 bits in the D-unit ALU.

- ❏ Input operands (Xmem and Ymem) are sign extended to 40 bits according to SXMD.
- ❏ CARRY status bit depends on M40. Subtraction borrow bit is reported in CARRY status bit. It is the logical complement of CARRY status bit.
- ❏ When an overflow is detected according to M40:
  - ■ the destination accumulator overflow status bit (ACOVx) is set
  - ■ the destination register (ACx) is saturated according to SATD

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 set to 0, compatibility is ensured.

When C54CM is 1, the subtract operation does not have any overflow detection, report, and saturation after the shifting operation.

Status Bits

Affected by FRCT, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ABDST *AR0+,*AR1,AC0,AC1	The absolute value of the content of AC0 is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 is subtracted from the content addressed by AR0 and the result is stored in AC0. The content of AR0 is incremented by 1.

Before		After	
AC0	00 0000 0000	AC0	00 4500 0000
AC1	00 E800 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

4.2 Absolute Value (ABS)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ABS [src,] dst	Yes	2	1	X

Operands dst, src

Description

This instruction computes the absolute value of the source register (src) content.

- When the destination register (dst) is an accumulator (ACx):
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or temporary register is the source operand of the instruction, the 16 LSBs of the auxiliary or temporary register are sign extended to 40 bits according to SXMD.
  - If M40 = 0, the sign of the source register is extracted at bit position 31. If src(31) = 1, the source register content is negated. If src(31) = 0, the source register content is moved to the destination accumulator.
  - If M40 = 1, the sign of the source register is extracted at bit position 39. If src(39) = 1, the source register content is negated. If src(39) = 0, the source register content is moved to the destination accumulator.
  - During the 40-bit move operation, an overflow and CARRY bit status are detected according to M40:
    - The destination accumulator overflow status bit (ACOVx) is set.
    - The destination register (ACx) is saturated according to SATD.
    - The CARRY status bit is updated as follows: If the result of the operation stored in the destination register is 0, CARRY is set; otherwise, CARRY is cleared.
- When the destination register (dst) is an auxiliary or temporary register (TAx):
  - The operation is performed on 16 bits in the A-unit ALU.
  - If an accumulator is the source operand of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - The sign of the source register is extracted at bit position 15. If src(15) = 1, the source register content is negated. If src(15) = 0, the source register content is moved to the destination register. Overflow is detected at bit position 15.
  - The destination register (TAx) is saturated according to SATA.

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 cleared to 0.

Status Bits

Affected by C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Examples

Syntax	Description
ABS AC0, AC1	The absolute value of the content 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

Syntax	Description
ABS AR1, AC1	The absolute value of the content of AR1 is stored in AC1.

Before		After	
AC1	00 0000 2000	AC1	00 0000 0000
AR1	0000	AR1	0000
CARRY	0	CARRY	1

Syntax	Description
ABS AR1, AC1	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		After	
AC1	00 0000 2000	AC1	00 0000 7900
AR1	8700	AR1	8700
M40	0	M40	0
SXMD	1	SXMD	1

Syntax	Description
ABS AC0, T1	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	2000	T1	1234
AC0	80 0002 1234	AC0	80 0002 1234

Syntax	Description
ABS AC0, T1	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	
T1	2000	T1	6DCC
AC0	80 0002 9234	AC0	80 0002 9234

4.3 Accumulator, Auxiliary, or Temporary Register Content Swap

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[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
[5]	SWAP T0, T2	Yes	2	1	AD
[6]	SWAP T1, T3	Yes	2	1	AD
[7]	SWAP AR0, AR2	Yes	2	1	AD
[8]	SWAP AR1, AR3	Yes	2	1	AD
[9]	SWAP AR0, AR1	Yes	2	1	AD
[10]	SWAP AC0, AC2	Yes	2	1	X
[11]	SWAP AC1, AC3	Yes	2	1	X
[12]	SWAPP AR4, T0	Yes	2	1	AD
[13]	SWAPP AR6, T2	Yes	2	1	AD
[14]	SWAPP T0, T2	Yes	2	1	AD
[15]	SWAPP AR0, AR2	Yes	2	1	AD
[16]	SWAPP AC0, AC2	Yes	2	1	X
[17]	SWAP4 AR4, T0	Yes	2	1	AD

Brief Description

These instructions perform parallel moves between accumulators, auxiliary registers, or temporary registers. These operations are performed in a dedicated datapath independent of the A-unit operators and D-unit operators.

These instructions allow you to move the content of the first accumulator, auxiliary or temporary register to the second accumulator, auxiliary or temporary register, and reciprocally to move the content of the second register to the first register.

Auxiliary and temporary register swapping is performed in the address phase of the pipeline. Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by none

Affects none

4.3.1 Auxiliary and Temporary Register Content Swap: SWAP ARx, Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[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

Operands

ARx, Tx

Description

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 swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAP AR4, T0	The content of AR4 is moved to T0 and the content of T0 is moved to AR4.

Before		After	
T0	6500	T0	0300
AR4	0300	AR4	6500

4.3.2 Temporary Register Content Swap: SWAP Tx, Ty

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	SWAP T0, T2	Yes	2	1	AD
[6]	SWAP T1, T3	Yes	2	1	AD

Operands

Tx

Description

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 performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAP T0, T2	The content of T0 is moved to T2 and the content of T2 is moved to T0.

Before		After	
T0	6500	T0	0300
T2	0300	T2	6500

4.3.3 Auxiliary Register Content Swap: SWAP ARx, ARy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	SWAP AR0, AR2	Yes	2	1	AD
[8]	SWAP AR1, AR3	Yes	2	1	AD
[9]	SWAP AR0, AR1	Yes	2	1	AD

Operands

ARx

Description

This instruction performs parallel moves between two auxiliary registers. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction moves the content of the first auxiliary register (ARx) to the second auxiliary register (ARy), and reciprocally moves the content of the second auxiliary register to the first auxiliary register.

Auxiliary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAP AR0, AR2	The content of AR0 is moved to AR2 and the content of AR2 is moved to AR0.

Before		After	
AR0	6500	AR0	0300
AR2	0300	AR2	6500

4.3.4 Accumulator Content Swap: SWAP ACx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	SWAP AC0, AC2	Yes	2	1	X
[11]	SWAP AC1, AC3	Yes	2	1	X

Operands

ACx

Description

This instruction performs parallel moves between two accumulators. These operations are performed in a dedicated datapath independent of the D-unit operators.

This instruction moves the content of the first accumulator (ACx) to the second accumulator (ACy), and reciprocally moves the content of the second accumulator to the first accumulator.

Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAP AC0, AC2	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 0200        AC2         01 E500 0030
```

4.3.5 Auxiliary and Temporary Register Pair Content Swap: SWAPP AR4, T0

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	SWAPP AR4, T0	Yes	2	1	AD

Operands

AR4, T0

Description

This instruction performs two parallel moves between two auxiliary registers (AR4 and AR5) and two temporary registers (T0 and T1) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction 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

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAPP AR4, T0	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		After	
AR4	0200	AR4	6788
AR5	0300	AR5	0200
T0	6788	T0	0200
T1	0200	T1	0300

4.3.6 Auxiliary and Temporary Register Pair Content Swap: SWAPP AR6, T2

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	SWAPP AR6, T2	Yes	2	1	AD

Operands

AR6, T2

Description

This instruction performs two parallel moves between two auxiliary registers (AR6 and AR7) and two temporary registers (T2 and T3) in one cycle. These operations are performed in a dedicated datapath independent of the A-unit operators.

This instruction performs two parallel moves:

- the content of AR6 to T2, and reciprocally the content of T2 to AR6
- the content of AR7 to T3, and reciprocally the content of T3 to AR7

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAPP AR6, T2	The following two swap instructions are performed in parallel: 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	
AR6	0200	AR6	6788
AR7	0300	AR7	0200
T2	6788	T2	0200
T3	0200	T3	0300

4.3.7 Temporary Register Pair Content Swap: SWAPP T0, T2

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	SWAPP T0, T2	Yes	2	1	AD

Operands Tx

Description

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.

This instruction performs two parallel moves:

- the content of T0 to T2, and reciprocally the content of T2 to T0
- the content of T1 to T3, and reciprocally the content of T3 to T1

Temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAPP T0, T2	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	
T0	0200	T0	6788
T1	0300	T1	0200
T2	6788	T2	0200
T3	0200	T3	0300

4.3.8 Auxiliary Register Pair Content Swap: SWAPP AR0, AR2

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	SWAPP AR0, AR2	Yes	2	1	AD

Operands

ARx

Description

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.

This instruction performs two parallel moves:

- the content of AR0 to AR2, and reciprocally the content of AR2 to AR0
- the content of AR1 to AR3, and reciprocally the content of AR3 to AR1

Auxiliary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAPP AR0, AR2	The following two swap instructions are performed in parallel: the content of 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

4.3.9 Accumulator Pair Content Swap: SWAPP AC0, AC2

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	SWAPP AC0, AC2	Yes	2	1	X

Operands

ACx

Description

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.

This instruction performs two parallel moves:

- the content of AC0 to AC2, and reciprocally the content of AC2 to AC0
- the content of AC1 to AC3, and reciprocally the content of AC3 to AC1

Accumulator swapping is performed in the execute phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAPP AC0, AC2	The following two swap instructions are performed in parallel: the content of 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

4.3.10 Auxiliary and Temporary Register Content Swap: SWAP4 AR4, T0

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	SWAP4 AR4, T0	Yes	2	1	AD

Operands

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.

This instruction performs four 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
- the content of AR6 to T2, and reciprocally the content of T2 to AR6
- the content of AR7 to T3, and reciprocally the content of T3 to AR7

Auxiliary and temporary register swapping is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
SWAP4 AR4, 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
T0	0030	T0	0200
T1	0200	T1	0300
T2	3400	T2	0240
T3	0FD3	T3	0400

4.4 Accumulator, Auxiliary, or Temporary Register Load

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV k4,dst	Yes	2	1	X
[2]	MOV –k4,dst	Yes	2	1	X
[3]	MOV K16,dst	No	4	1	X
[4]	MOV Smem,dst	No	2	1	X
[5]	MOV [uns(]high_byte(Smem)[)], dst	No	3	1	X
[6]	MOV [uns(]low_byte(Smem)[)], dst	No	3	1	X
[7]	MOV K16 << #16, ACx	No	4	1	X
[8]	MOV K16 << #SHFT, ACx	No	4	1	X
[9]	MOV [rnd(]Smem << Tx[)], ACx	No	3	1	X
[10]	MOV low_byte(Smem) << #SHIFTW, ACx	No	3	1	X
[11]	MOV high_byte(Smem) << #SHIFTW, ACx	No	3	1	X
[12]	MOV Smem << #16, ACx	No	2	1	X
[13]	MOV [uns(]Smem[)], ACx	No	3	1	X
[14]	MOV [uns(]Smem[)] << #SHIFTW, ACx	No	4	1	X
[15]	MOV[40] (dbl(Lmem), ACx	No	3	1	X
[16]	MOV Xmem, Ymem, ACx	No	3	1	X
[17]	MOV dbl(Lmem), pair(HI(ACx))	No	3	1	X
[18]	MOV dbl(Lmem), pair(LO(ACx))	No	3	1	X
[19]	MOV dbl(Lmem), pair(TAx)	No	3	1	X

Brief Description

This instruction loads a 4-bit unsigned constant, k4, 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 destination (dst) register.

Status Bits

Affected by C54CM, M40, RDM, SATD, SXMD

Affects ACOVx

4.4.1 Accumulator, Auxiliary, or Temporary Register Load: MOV k4, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV k4, dst	Yes	2	1	X

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 none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV #2, AC0	AC0 is loaded with the unsigned 4-bit value (2).

4.4.2 Accumulator, Auxiliary, or Temporary Register Load: MOV –k4, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MOV –k4, dst	Yes	2	1	X

Operands dst, k4

Description

This instruction loads the 2s complement representation of the 4-bit unsigned constant, k4, to the destination (dst) register.

- When the destination register is an accumulator:
  - The 4-bit constant, k4, is negated in the I-unit, loaded into the accumulator, and sign extended to 40 bits before being processed by the D-unit as a signed constant.
  - 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.
  - 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 none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV #–2, AC0	AC0 is loaded with a 2s complement representation of the unsigned 4-bit value (2).

4.4.3 Accumulator, Auxiliary, or Temporary Register Load: MOV K16, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MOV K16, dst	No	4	1	X

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 register, 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV #248, AC1	AC1 is loaded with the signed 16-bit value (248).

```
Before                          After
AC1         00 0200 FC00        AC1         00 0000 00F8
```

4.4.4 Accumulator, Auxiliary, or Temporary Register Load: MOV Smem, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	MOV Smem, dst	No	2	1	X

Operands dst, Smem

Description

This instruction loads the content of a memory (Smem) location to the destination (dst) register.

- When the destination register is an accumulator:
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - 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 content of the memory location is sign 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *AR3+, AR1	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

4.4.5 Accumulator, Auxiliary, or Temporary Register Load: MOV [uns(]high_byte(Smem)[)], dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MOV [uns(]high_byte(Smem)[)], dst	No	3	1	X

Operands dst, Smem

Description

This instruction loads the high-byte content of a memory (Smem) location to the destination (dst) register.

- When the destination register is an accumulator register:
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 content of the memory location is zero extended to 16 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 16 bits regardless of SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV uns(high_byte(*AR3)), AC0	The high-byte content addressed by AR3 is sign extended to 40 bits and loaded into AC0.

4.4.6 Accumulator, Auxiliary, or Temporary Register Load: MOV [uns(]low_byte(Smem)[)], dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MOV [uns(]low_byte(Smem)[)], dst	No	3	1	X

Operands dst, Smem

Description

This instruction loads the low-byte content of a memory (Smem) location to the destination (dst) register.

- When the destination register is an accumulator:
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 content of the memory location is zero extended to 16 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 16 bits regardless of SXMD, if the optional uns keyword is not applied to the input operand.
  - 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV uns(low_byte(*AR3)), AC0	The low-byte content addressed by AR3 is sign extended to 40 bits and loaded into AC0.

4.4.7 Accumulator Load: MOV K16 << #16, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MOV K16 << #16, ACx	No	4	1	X

Operands ACx, K16

Description

This instruction loads the 16-bit signed constant, K16, shifted left by 16 bits to the accumulator (ACx):

- The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- The shift operation is identical to the signed shift instruction.
- The input operand is shifted left by 16 bits according 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 M40, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV #–2 << #16, AC0	AC0 is loaded with the signed 16-bit value (–2) shifted left by 16 bits.

4.4.8 Accumulator Load: MOV K16 << #SHFT, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	MOV K16 << #SHFT, ACx	No	4	1	X

Operands ACx, K16, SHFT

Description

This instruction loads the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT, to the accumulator (ACx):

- The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is identical to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV #–2 << #15, AC0	AC0 is loaded with the signed 16-bit value (–2) shifted left by 15 bits.

4.4.9 Accumulator Load: MOV [rnd(]Smem << Tx[)], ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	MOV [rnd(]Smem << Tx[)], ACx	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction loads the content of a memory (Smem) location shifted by the content of Tx to the accumulator (ACx):

- The input operand is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 4-bit value in the D-unit shifter. The shift operation is identical to the signed shift instruction.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

- 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, RDM, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

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

4.4.10 Accumulator Load: MOV low_byte(Smem) << #SHIFTW, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	MOV low_byte(Smem) << #SHIFTW, ACx	No	3	1	X

Operands

ACx, SHIFTW, Smem

Description

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 of the memory location is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is identical to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

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

4.4.11 Accumulator Load: MOV high_byte(Smem) << #SHIFTW, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	MOV high_byte(Smem) << #SHIFTW, ACx	No	3	1	X

Operands ACx, SHIFTW, Smem

Description

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 of the memory location is sign extended to 40 bits according to SXMD.
- The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is identical to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

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

4.4.12 Accumulator Load: MOV Smem << #16, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	MOV Smem << #16, ACx	No	2	1	X

Operands

ACx, Smem

Description

This instruction loads the content of a memory (Smem) location shifted left by 16 bits to the accumulator (ACx):

- The input operand is sign extended to 40 bits according to SXMD.
- The shift operation is identical to the signed shift instruction.
- The input operand is shifted left by 16 bits according 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 M40, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *AR3+ << #16, AC1	The content addressed by AR3 shifted left by 16 bits is loaded into AC1. AR3 is incremented by 1.

```
Before                          After
AC1          00 0200 FC00       AC1          00 3400 0000
AR3                  0200       AR3                  0201
200                  3400       200                  3400
```

4.4.13 Accumulator Load: MOV [uns()Smem[])], ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	MOV [uns()Smem[])], ACx	No	3	1	X

Operands

ACx, Smem

Description

This instruction loads the content of a memory (Smem) location to the accumulator (ACx):

- The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
- The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

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

4.4.14 Accumulator Load: MOV [uns()Smem[])] << #SHIFTW, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	MOV [uns()]Smem[])] << #SHIFTW, ACx	No	4	1	X

Operands

ACx, SHIFTW, Smem

Description

This instruction loads the content of a memory (Smem) location, shifted by the 6-bit value, SHIFTW, to the to the accumulator (ACx):

- The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
- The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- The input operand is shifted by the 6-bit value in the D-unit shifter. The shift operation is identical to the signed shift instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by C54CM, M40, SATD, SXMD

Affects ACOVx

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
MOV uns(*AR3) << #31, AC0	The content addressed by AR3 is zero extended to 40 bits, shifted left by 31 bits, and loaded into AC0.

4.4.15 Accumulator Load: MOV[40] dbl(Lmem), ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	MOV[40] dbl(Lmem), ACx	No	3	1	X

Operands

ACx, Lmem

Description

This instruction loads the content of data memory operand (Lmem) to the accumulator (ACx):

- The input operand is sign extended to 40 bits according to SXMD.
- 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.
- Status bit M40 is locally set to 1, if the optional 40 keyword is applied to the input operand.

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

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV40 dbl(*AR3–), AC0	The content (long word) addressed by AR3 and AR3 + 1 is loaded into AC0. Because this instruction is a long-operand instruction, AR3 is decremented by 2 after the execution.

4.4.16 Accumulator Load: MOV Xmem, Ymem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	MOV Xmem, Ymem, ACx	No	3	1	X

Operands ACx, Xmem, Ymem

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 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 is set.
- If SATD is 1 when an overflow is detected on the higher data path, a saturation is performed with saturation value of 00 7FFFh.

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

Affected by C54CM, SATD, SXMD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *AR3, *AR4, AC0	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).

4.4.17 Accumulator Load: MOV dbl(Lmem), pair(HI(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	MOV dbl(Lmem), pair(HI(ACx))	No	3	1	X

Operands

ACx, Lmem

Description

This instruction loads the 16 highest bits of data memory operand (Lmem) to the 16 highest bits of the accumulator (ACx) and loads the 16 lowest bits of data memory operand (Lmem) to the 16 highest 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 accumulator destinations are AC0 and AC2.

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 operation.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVx, ACOV(x + 1)

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV dbl(*AR3+), pair(HI(AC2))	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
```

4.4.18 Accumulator Load: MOV dbl(Lmem), pair(LO(ACx))

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	MOV dbl(Lmem), pair(LO(ACx))	No	3	1	X

Operands 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 accumulator destinations are AC0 and AC2.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV dbl(*AR3), pair(LO(AC0))	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).

4.4.19 Auxiliary or Temporary Register Load: MOV dbl(Lmem), pair(TAx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[19]	MOV dbl(Lmem), pair(TAx)	No	3	1	X

Operands

TAx, Lmem

Description

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 register destinations are AR0, AR2, AR4, and AR6.
- Valid temporary register destinations are T0 and T2.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV dbl(*AR3), pair(T0)	The 16 highest bits of the content at the location addressed by AR3 are loaded into T0 and the 16 lowest bits of the content at the location addressed by AR3 + 1 are loaded into T1.

4.5 Accumulator, Auxiliary, or Temporary Register Move

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV src, dst	Yes	2	1	X
[2]	MOV HI(ACx), TAx	Yes	2	1	X
[3]	MOV TAx, HI(ACx)	Yes	2	1	X

Brief Description

This instruction moves:

- ❑ the content of the source (src) register to the destination (dst) register.
- ❑ the 16 MSBs of the selected accumulator (ACx) to the destination auxiliary or temporary register (TAx).
- ❑ the content of the auxiliary or temporary register (TAx) to the high part of the selected accumulator (ACx).

When the destination register is an accumulator, the 40-bit move operation is performed in the D-unit ALU. When the destination register is an auxiliary or temporary register, the 16-bit move operation is performed in the A-unit ALU.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVx

4.5.1 Accumulator, Auxiliary, or Temporary Register Move: MOV src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV src, dst	Yes	2	1	X

Operands dst, src

Description

This instruction moves the content of the source (src) register to the destination (dst) register:

- When the destination (dst) register is an accumulator (ACx):
  - 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:
    - the destination accumulator overflow status bit (ACOVx) is set.
    - the destination register (ACx) is saturated according to SATD.
  - If the source (src) register is an auxiliary or temporary register, the 16 low bits of the source register are sign extended to 40 bits according to SXMD.
- When the destination (dst) register is an auxiliary or temporary register (TAx):
  - 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 register 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

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0, AC1	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

4.5.2 Accumulator Move: MOV HI(ACx), TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MOV HI(ACx), TAx	Yes	2	1	X

Operands

ACx, TAx

Description

This instruction moves the 16 MSBs of the accumulator (ACx) 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 instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV HI(AC0), AR2	The content of AC0(31–16) is copied to AR2.

```
Before                          After
AC0          01 E500 0030       AC0          01 E500 0030
AR2                  0200       AR2                  E500
```

4.5.3 Auxiliary or Temporary Register Move: MOV TAx, HI(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MOV TAx, HI(ACx)	Yes	2	1	X

Operands ACx, TAx

Description

This instruction moves the content of the auxiliary or temporary register (TAx) to the high part of the accumulator (ACx):

- The 16-bit move operation is performed in the D-unit ALU.
- During the 16-bit move operation, an overflow is detected according to M40:
  - the destination accumulator overflow status bit (ACOVx) is set.
  - the destination register (ACx) is saturated according to SATD.
- If the source (src) register is an auxiliary or temporary register, the 16 low bits of the source register are sign extended to 40 bits according to SXMD.

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

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV T0, HI(AC0)	The content of T0 is copied to AC0(31–16).

4.6 Accumulator, Auxiliary, or Temporary Register Store

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV src, Smem	No	2	1	X
[2]	MOV src, high_byte(Smem)	No	3	1	X
[3]	MOV src, low_byte(Smem)	No	3	1	X
[4]	MOV HI(ACx), Smem	No	2	1	X
[5]	MOV [rnd(]HI(ACx)[)], Smem	No	3	1	X
[6]	MOV ACx << Tx, Smem	No	3	1	X
[7]	MOV [rnd(]HI(ACx << Tx)[)], Smem	No	3	1	X
[8]	MOV ACx << #SHIFTW, Smem	No	3	1	X
[9]	MOV HI(ACx << #SHIFTW), Smem	No	3	1	X
[10]	MOV [rnd(]HI(ACx << #SHIFTW)[)], Smem	No	4	1	X
[11]	MOV [uns)] [rnd(]HI(saturate(ACx))[))], Smem	No	3	1	X
[12]	MOV [uns(] [rnd(]HI(saturate(ACx << Tx))[))], Smem	No	3	1	X
[13]	MOV [uns(] [rnd(]HI(saturate(ACx << #SHIFTW))[))], Smem	No	4	1	X
[14]	MOV ACx, dbl(Lmem)	No	3	1	X
[15]	MOV [uns(]saturate(ACx)[)], dbl(Lmem)	No	3	1	X
[16]	MOV ACx, >> #1, dual(Lmem)	No	3	1	X
[17]	MOV pair(HI(ACx)), dbl(Lmem)	No	3	1	X
[18]	MOV pair(LO(ACx)), dbl(Lmem)	No	3	1	X
[19]	MOV pair(TAx), dbl(Lmem)	No	3	1	X
[20]	MOV ACx, Xmem, Ymem	No	3	1	X

Brief Description

This instruction stores the content of the selected source (src) register 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

4.6.1 Accumulator, Auxiliary, or Temporary Register Store: MOV src, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV src, Smem	No	2	1	X

Operands Smem, src

Description

This instruction stores the content of the source (src) register to a memory (Smem) location.

- When the source register is an accumulator:
  - The low part of the accumulator, ACx(15–0), is stored to the memory 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.
- When the source register is an auxiliary or temporary register:
  - The content of the auxiliary or temporary register is stored to the memory location.
  - The store operation to the memory location uses a dedicated path independent of the A-unit ALU.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0, *(#0E10h)	The content of AC0(15–0) is stored at location E10h.

```
Before                          After
AC0         23 0400 6500        AC0         23 0400 6500
0E10                0000        0E10                6500
```

4.6.2 Accumulator, Auxiliary, or Temporary Register Store: MOV src, high_byte(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MOV src, high_byte(Smem)	No	3	1	X

Operands Smem, src

Description

This instruction stores the low part (bits 7–0) content of the source (src) register to the high byte of the memory (Smem) location.

- When the source register is an accumulator:
  - The low part of the accumulator, ACx(7–0), is stored to the high byte of the memory 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.
- When the source register is an auxiliary or temporary register:
  - The low part (bits 7–0) content of the auxiliary or temporary register is stored to the high byte of the memory location.
  - The store operation to the memory location uses a dedicated path independent of the A-unit ALU.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC1, high_byte(*AR1)	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
```

4.6.3 Accumulator, Auxiliary, or Temporary Register Store: MOV src, low_byte(Smem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MOV src, low_byte(Smem)	No	3	1	X

Operands Smem, src

Description

This instruction stores the low part (bits 7–0) content of the source (src) register to the low byte of the memory (Smem) location.

- When the source register is an accumulator:
  - The low part of the accumulator, ACx(7–0), is stored to the low byte of the memory 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.
- When the source register is an auxiliary or temporary register:
  - The low part (bits 7–0) content of the auxiliary or temporary register is stored to the low byte of the memory location.
  - The store operation to the memory location uses a dedicated path independent of the A-unit ALU.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0, low_byte(*AR3)	The content of AC0(7–0) is stored in the low byte at the location addressed by AR3.

4.6.4 Accumulator Store: MOV HI(ACx), Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	MOV HI(ACx), Smem	No	2	1	X

Operands ACx, Smem

Description

This instruction stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location.

- The high part of the accumulator, ACx(31–16), is stored to the memory 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV HI(AC0), *AR3	The content of AC0(31–16) is stored at the location addressed by AR3.

4.6.5 Accumulator Store: MOV [rnd(]HI(ACx)[)], Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MOV [rnd(]HI(ACx)[)], Smem	No	3	1	X

Operands

ACx, Smem

Description

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.
- The high part of the accumulator, ACx(31–16), is stored to the memory location.

Status Bits

Affected by RDM

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV rnd(HI(AC0)), *AR3	The content of AC0(31–16) is rounded and stored at the location addressed by AR3.

4.6.6 Accumulator Store: MOV ACx << Tx, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MOV ACx << Tx, Smem	No	3	1	X

Operands ACx, Tx, Smem

Description

- This instruction shifts the accumulator, ACx, by the content of Tx and stores 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.
- After the shift, the low part of the accumulator, ACx(15–0), is stored to the memory location.

Compatibility with C54x devices (C54CM = 1)

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.

Status Bits

Affected by C54CM, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0 << T0, *AR3	The content of AC0 is shifted by the content of T0 and AC0(15–0) is stored at the location addressed by AR3.

4.6.7 Accumulator Store: MOV [rnd()HI(ACx << Tx)[]], Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MOV [rnd()]HI(ACx << Tx)[]], Smem	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction shifts the accumulator, ACx, by the content of Tx and stores ACx(31–16) 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.
- Rounding is performed in the D-unit shifter according to RDM, if the optional rnd keyword is applied to the input operand.
- 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 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.

Status Bits

Affected by C54CM, RDM, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV rnd(HI(AC0 << T0)), *AR3	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.

4.6.8 Accumulator Store: MOV ACx << #SHIFTW, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	MOV ACx << #SHIFTW, Smem	No	3	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores 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.
- After the shift, the low part of the accumulator, ACx(15–0), is stored to the memory location.

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0 << #31, *AR3	The content of AC0 is shifted left by 31 bits and AC0(15–0) is stored at the location addressed by AR3.

4.6.9 Accumulator Store: MOV HI(ACx << #SHIFTW), Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	MOV HI(ACx << #SHIFTW), Smem	No	3	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores ACx(31–16) to the memory (Smem) location.

- The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.
- After the shift, the low part of the accumulator, ACx(31–16), is stored to the memory location.

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV HI(AC0 << #31), *AR3	The content of AC0 is shifted left by 31 bits and AC0(31–16) is stored at the location addressed by AR3.

4.6.10 Accumulator Store: MOV [rnd()HI(ACx << #SHIFTW)[)], Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	MOV [rnd()HI(ACx << #SHIFTW)[)], Smem	No	4	1	X

Operands

ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores ACx(31–16) to the memory (Smem) 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.
- After the shift, the high part of the accumulator, ACx(31–16), is stored to the memory location.

Status Bits

Affected by RDM, SXMD

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
MOV rnd(HI(AC0 << #31)), *AR3	The content of AC0 is shifted left by 31 bits, is rounded, and AC0(31–16) is stored at the location addressed by AR3.

4.6.11 Accumulator Store: MOV [uns()][rnd()HI(saturate(ACx))[])], Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	MOV [uns()][rnd()HI(saturate(ACx))[])], Smem	No	3	1	X

Operands ACx, Smem

Description

This instruction stores the high part of the accumulator, ACx(31–16), to the memory (Smem) location.

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- When a rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- 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 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 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

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV uns(rnd(HI(saturate(AC0)))), *AR3	The unsigned content of AC0 is rounded, is saturated, and AC0(31–16) is stored at the location addressed by AR3.

4.6.12 Accumulator Store: MOV [uns(] [rnd(]HI(saturate(ACx << Tx))[])], Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	MOV [uns(] [rnd(]HI(saturate(ACx << Tx))[])], Smem	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction shifts the accumulator, ACx, by the content of Tx and stores ACx(31–16) 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.

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- The input operand is shifted in the D-unit shifter according to SXMD.
- When shifting, the sign position of the input operand is compared to the shift quantity.
  - If the optional uns keyword is applied to the input operand, this comparison is performed against bit 32 of the shifted operand.
  - If the optional uns keyword is not applied, this comparison is performed against bit 31 of the shifted operand that is considered signed (the sign is defined by bit 39 of the input operand and SXMD).
  - An overflow is generated accordingly.
- If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- When a shift or rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- 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 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.
- 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.

Status Bits

Affected by C54CM, RDM, SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV uns(rnd(HI(saturate(AC0 << T0)))), *AR3	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.

4.6.13 Accumulator Store: MOV [uns(] [rnd(]HI(saturate(ACx << #SHIFTW))[]))], Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	MOV [uns(](rnd(]HI(saturate(ACx << #SHIFTW))[]))], Smem	No	4	1	X

Operands ACx, SHIFTW, Smem

Description

This instruction shifts the accumulator, ACx, by the 6-bit value, SHIFTW, and stores ACx(31–16) to the memory (Smem) location.

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- The input operand is shifted by the 6-bit value in the D-unit shifter according to SXMD.
- When shifting, the sign position of the input operand is compared to the shift quantity.
  - If the optional uns keyword is applied to the input operand, this comparison is performed against bit 32 of the shifted operand.
  - If the optional uns keyword is not applied, this comparison is performed against bit 31 of the shifted operand that is considered signed (the sign is defined by bit 39 of the input operand and SXMD).
  - An overflow is generated accordingly.
- If the optional rnd keyword is applied to the input operand, rounding is performed in the D-unit shifter according to RDM.
- When a shift or rounding overflow is detected and if the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- 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 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

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
MOV uns(rnd(HI(saturate(AC0 << #31)))), *AR3	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.

4.6.14 Accumulator Store: MOV ACx, dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	MOV ACx, dbl(Lmem)	No	3	1	X

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 Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
MOV AC0, dbl(*AR3)	The content of AC0 is stored at the locations addressed by AR3 and AR3 + 1.

4.6.15 Accumulator Store: MOV [uns(]saturate(ACx)[)], dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	MOV [uns(]saturate(ACx)[)], dbl(Lmem)	No	3	1	X

Operands

ACx, Lmem

Description

This instruction stores the content of the accumulator, ACx(31–0), to the data memory operand (Lmem).

- If the optional uns keyword is applied to the input operand, the input operand is considered unsigned.
- If the optional saturate keyword is applied to the input operand, the 40-bit output of the operation is saturated:
  - If the optional uns keyword is applied to the input operand, saturation value is 00 FFFF FFFFh.
  - If the optional uns keyword is not applied, saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).
- The store operation to the memory location uses the D-unit shifter.

Compatibility with 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 SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV uns(saturate(AC0)), dbl(*AR3)	The unsigned content of AC0 is saturated and stored at the locations addressed by AR3 and AR3 + 1.

4.6.16 Accumulator Store: MOV ACx >> #1, dual(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	MOV ACx >> #1, dual(Lmem)	No	3	1	X

Operands

ACx, Lmem

Description

This instruction is executed in the D-unit shifter:

- The 16 highest bits of the accumulator (ACx), 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).
- The 16 lowest bits of ACx, shifted right by 1 bit (bit 15 is sign extended according to SXMD), are stored to the 16 lowest bits of the data memory operand (Lmem).

Status Bits

Affected by SXMD

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0 >> #1,dual(*AR3)	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.

4.6.17 Accumulator Store: MOV pair(HI(ACx)), dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	MOV pair(HI(ACx)), dbl(Lmem)	No	3	1	X

Operands

ACx, Lmem

Description

This instruction stores the 16 highest bits of the accumulator (ACx) 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 accumulators are AC0 and AC2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV pair(HI(AC0)), dbl(*AR1+)	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
```

4.6.18 Accumulator Store: MOV pair(LO(ACx)), dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	MOV pair(LO(ACx)), dbl(Lmem)	No	3	1	X

Operands

ACx, Lmem

Description

This instruction stores the 16 lowest bits of the accumulator (ACx) 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):

- 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 accumulators are AC0 and AC2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV pair(LO(AC0)), dbl(*AR3)	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.

4.6.19 Auxiliary or Temporary Register Store: MOV pair(TAx), dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[19]	MOV pair(TAx), dbl(Lmem)	No	3	1	X

Operands

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):

- The store operation to the memory location uses a dedicated path independent of the A-unit ALU.
- Valid auxiliary register destinations are AR0, AR2, AR4, and AR6.
- Valid temporary register destinations are T0 and T2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV pair(T0), dbl(*AR3)	The content of T0 is stored at the location addressed by AR3 and the content of T1 is stored at the location addressed by AR3 + 1.

4.6.20 Accumulator Store: MOV ACx, Xmem, Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[20]	MOV ACx, Xmem, Ymem	No	3	1	X

Operands

ACx, Xmem, Ymem

Description

This instruction stores the 16 lowest bits of the accumulator (ACx) to data memory operand Xmem and stores the 16 highest bits of ACx to data memory operand Ymem.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0, *AR1, *AR2	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.

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

4.7 Addition (ADD)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ADD [src,] dst	Yes	2	1	X
[2]	ADD k4, dst	Yes	2	1	X
[3]	ADD K16, [src,] dst	No	4	1	X
[4]	ADD Smem, [src,] dst	No	3	1	X
[5]	ADD ACx << Tx	Yes	2	1	X
[6]	ADD ACx << #SHIFTW, ACy	Yes	3	1	X
[7]	ADD K16 << #16, [ACx,] ACy	No	4	1	X
[8]	ADD K16 << #SHFT, [ACx,] ACy	No	4	1	X
[9]	ADD Smem << Tx, [ACx,] ACy	No	3	1	X
[10]	ADD Smem << #16, [ACx,] ACy	No	3	1	X
[11]	ADD [uns(]Smem[)], CARRY, [ACx,] ACy	No	3	1	X
[12]	ADD [uns(]Smem[)], [ACx,] ACy	No	3	1	X
[13]	ADD [uns(]Smem[)] << #SHIFTW, [ACx,] ACy	No	4	1	X
[14]	ADD dbl(Lmem), [ACx,] ACy	No	3	1	X
[15]	ADD Xmem, Ymem, ACx	No	3	1	X
[16]	ADD K16, Smem	No	4	1	X
[17]	ADDV [ACx,] ACy	Yes	2	1	X

Brief Description

These instructions perform an addition operation:

- ❏ In the D-unit ALU, if the destination operand is an accumulator (ACx).
- ❏ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).
- ❏ In the D-unit ALU, if the destination operand is the memory (Smem).
- ❏ In the D-unit shifter, if the instruction has a shift quantity other than the immediate 16 bit shift.

Status Bits

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

Affects ACOVx, ACOVy, CARRY

4.7.1 Addition: ADD [src,] dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ADD [src,] dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs an addition operation between two registers contents.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - If an auxiliary or data register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or data register are sign extended according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- When the destination (dst) operand is an auxiliary or data 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.
  - Addition 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD AC1, AC0	The content of AC1 is added to the content of AC0 and the result is stored in AC0.

4.7.2 Addition: ADD k4, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ADD k4, dst	Yes	2	1	X

Operands dst, k4

Description

This instruction performs an addition operation between a register content and a 4-bit unsigned constant, k4.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- When the destination (dst) operand is an auxiliary or data register:
  - The operation is performed on 16 bits in the A-unit ALU.
  - Addition 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.

Example

Syntax	Description
ADD #15, AC0	The content of AC0 is added to an unsigned 4-bit value (15) and the result is stored in AC0.

4.7.3 Addition: ADD K16, [src,] dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ADD K16, [src,] dst	No	4	1	X

Operands dst, K16, src

Description

This instruction performs an addition operation between a register content and a 16-bit signed constant, K16.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or data register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or data register are sign extended according to SXMD.
  - The 16-bit constant, lk, is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- When the destination (dst) operand is an auxiliary or data 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.
  - Addition 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

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

4.7.4 Addition: ADD Smem, [src,] dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ADD Smem, [src,] dst	No	3	1	X

Operands dst, Smem, src

Description

This instruction performs an addition operation between a register content and the content of a memory (Smem) location.

- When the destination (dst) operand is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - If an auxiliary or data register is the source (src) operand of the instruction, the 16 LSBs of the auxiliary or data register are sign extended according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - When an overflow is detected, the accumulator is saturated according to SATD.
- When the destination (dst) operand is an auxiliary or data 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.
  - Addition 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD *AR3+, T0, T1	The content of T0 is added to the content addressed by AR3 and the result is stored in T1. AR3 is incremented by 1.

Before		After	
AR3	0302	AR3	0303
302	EF00	302	EF00
T0	3300	T0	3300
T1	0	T1	2200
CARRY	0	CARRY	0

4.7.5 Addition: ADD ACx << Tx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ADD ACx << Tx, ACy	Yes	2	1	X

Operands ACx, ACy, Tx

Description

This instruction performs an addition operation between an accumulator content ACy and an accumulator content ACx shifted by the content of Tx.

- 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 identical 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD AC1 << T0, AC0	The content of AC1 shifted by the content of T0 is added to the content of AC0 and the result is stored in AC0.

4.7.6 Addition: ADD ACx << #SHIFTW, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	ADD ACx << #SHIFTW, ACy	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

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 SXMD.
- The shift operation is identical 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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

4.7.7 Addition: ADD K16 << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ADD K16 << #16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, K16

Description

This instruction performs an addition operation between an accumulator content ACx and the 16-bit signed constant, K16, shifted left by 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 identical 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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

4.7.8 Addition: ADD K16 << #SHFT, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	ADD K16 << #SHFT, [ACx,] ACy	No	4	1	X

Operands ACx, ACy, K16, SHFT

Description

This instruction performs an addition operation between an accumulator content ACx and the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT.

- 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 identical 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 M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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

4.7.9 Addition: ADD Smem << Tx, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ADD Smem << Tx, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Tx, Smem

Description

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 on 40 bits in the D-unit shifter.
- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is identical 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD *AR1 << T0, AC1, AC0	The content addressed by AR1 shifted left by the content of T0 is added to the content of AC1 and the result is stored in AC0.

Before		After	
AC0	00 0000 0000	AC0	00 2330 0000
AC1	00 2300 0000	AC1	00 2300 0000
T0	000C	T0	000C
AR1	0200	AR1	0200
200	0300	200	0300
SXMD	0	SXMD	0
M40	0	M40	0
ACOV0	0	ACOV0	0
CARRY	0	CARRY	1

4.7.10 Addition: ADD Smem << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	ADD Smem << #16, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

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 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 identical to the signed shift instruction.
- Overflow detection and CARRY status bit depends on M40. If the result of the addition generates a carry, the CARRY status bit is set; 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.

Example

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

4.7.11 Addition: ADD [uns()]Smem[])], CARRY, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	ADD [uns()]Smem[])], CARRY, [ACx,] ACy	No	3	1	X

Operands

ACx, ACy, Smem

Description

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 operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Status Bits

Affected by CARRY, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD uns(*AR3), CARRY, AC1, AC0	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.

4.7.12 Addition: ADD [uns()Smem[])], [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	ADD [uns()Smem[])], [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction performs an addition operation between the accumulator content ACx and the content of a memory (Smem) location.

- The operation is performed on 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

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

4.7.13 Addition: ADD [uns()Smem[])] << #SHIFTW, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	ADD [uns()Smem[])] << #SHIFTW, [ACx,] ACy	No	4	1	X

Operands ACx, ACy, SHIFTW, Smem

Description

This instruction performs an addition operation between an accumulator content ACx and the content of a memory (Smem) location 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 SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- The shift operation is identical 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

Affects ACOVy, CARRY

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
ADD uns(*AR3) << #31, AC1, AC0	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.

4.7.14 Addition: ADD dbl(Lmem), [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	ADD dbl(Lmem), [ACx,] ACy	No	3	1	X

Operands

ACx, ACy, Lmem

Description

This instruction performs an addition operation between the accumulator content ACx and the content of data memory operand dbl(Lmem).

- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least 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 SXMD.
- 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.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD dbl(*AR3+), AC1, AC0	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.

4.7.15 Addition: ADD Xmem, Ymem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	ADD Xmem, Ymem, ACx	No	3	1	X

Operands ACx, Xmem, Ymem

Description

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 identical 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

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD *AR3, *AR4, AC0	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.

4.7.16 Addition: ADD K16, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	ADD K16, Smem	No	4	1	X

Operands

K16, Smem

Description

This instruction performs an addition operation between a 16-bit signed constant, K16, and the content of a memory (Smem) location.

- 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 instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by SATD, SXMD

Affects ACOV0, CARRY

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
ADD #FFFFh, *AR3	The content addressed by AR3 is added to a signed 16-bit value and the result is stored back into the location addressed by AR3.

4.7.17 Addition: ADDV [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	ADDV [ACx,] ACy	Yes	2	1	X

Operands

ACx, ACy

Description

This instruction is performed in the D-unit MAC:

- The absolute value of accumulator ACx is computed by multiplying ACx(32–16) by 00001h or 1FFFFh depending on bit 32 of the source accumulator.
- If FRCT = 1, the absolute value is multiplied by 2.
- 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 is set.
- When an addition overflow is detected, the accumulator is saturated according to SATD.
- The result of the absolute value of the higher part of the source accumulator is in the lower part of the destination accumulator.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

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

4.8 Bit Field Comparison (BAND)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BAND Smem, k16, TCx	No	4	1	X

Operands k16, Smem, Tx

Description

This instruction performs a bit field manipulation in the A-unit ALU. The 16-bit field mask, k16, is ANDed with the memory (Smem) operand and the result is compared to 0:

```
if( ((Smem) AND k16 ) == 0)
   TCx = 0
else
   TCx = 1
```

Status Bits

Affected by none

Affects TCx

Repeat

This instruction cannot be repeated.

Examples

Syntax	Description
BAND *AR0, #0060h, TC1	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	Description
BAND *AR3, #00A0h, TC2	The unsigned 16-bit value (00A0h) is ANDed with the content addressed by AR3. The result is 0, TC2 is cleared to 0.

Before		After	
*AR3	0040	*AR3	0040
TC2	0	TC2	0

4.9 Bit Field Counting (BCNT)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BCNT ACx, ACy,TCx, Tx	Yes	3	1	X

Operands ACx, ACy, Tx, TCx

Description

This instruction performs bit field manipulation in the D-unit shifter. The result is stored in the selected temporary register (Tx). The A-unit ALU is used to make the move operation.

Accumulator ACx is ANDed with accumulator ACy. The number of bits set to 1 in the intermediary result is evaluated and stored in the selected temporary register (Tx). If the number of bits is even, the selected TCx status bit is cleared to 0. If the number of bits is odd, the selected TCx status bit is set to 1.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Example

Syntax	Description
BCNT AC1, AC2, TC1, T1	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
AC1         7E 2355 4FC0      AC1       7E 2355 4FC0
AC2         0F E340 5678      AC2       0F E340 5678
T1                  0000      T1                000B
TC1                    0      TC1                  1
```

4.10 Bit Field Expand (BFXPA)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BFXPA k16, ACx, dst	No	4	1	X

Operands ACx, dst, k16

Description

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 (EFC) 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) to the most significant bits (MSBs). According to the bit set to 1 in the bit field mask, the 16 LSBs of the source accumulator (ACx) bits are extracted and separated with 0 toward the MSBs. The result is stored in dst.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
BFXPA #8024h, AC0, T2	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 0101
T2                    1000 0000 0000 0100

Before                          After
AC0         00 2300 2B65        AC0         00 2300 2B65
T2                  0000        T2                  8004
```

4.11 Bit Field Extract (BFXTR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BFXTR k16, ACx, dst	No	4	1	X

Operands ACx, dst, k16

Description

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 (EFC) 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) to the most significant bits (MSBs). According to the bit set to 1 in the bit field mask, the corresponding 16 LSBs of the source accumulator (ACx) bits are extracted and packed toward the LSBs. The result is stored in dst.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
BFXTR #8024h, AC0, T2	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)          1000 0000 0010 0100
AC0(15–0)             0101 0101 1010 1010
T2                    0000 0000 0000 0010

Before                          After
AC0         00 2300 55AA        AC0         00 2300 55AA
T2                  0000        T2                  0002
```

4.12 Bitwise Complement (NOT)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	NOT [src,] dst	Yes	2	1	X

Operands

dst, src

Description

This instruction computes the 1s complement (bitwise complement) of the content of the source register (src).

- When the destination (dst) operand is an accumulator:
  - The bit inversion is performed on 40 bits in the D-unit ALU and the result is stored in the destination accumulator.
  - 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 bit inversion is performed on 16 bits in the A-unit ALU and the result is stored in the destination 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
NOT AC0, AC1	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
```

4.13 Bitwise AND

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	AND src, dst	Yes	2	1	X
[2]	AND k8, src, dst	Yes	3	1	X
[3]	AND k16, src, dst	No	4	1	X
[4]	AND Smem, src, dst	No	3	1	X
[5]	AND ACx << #SHIFTW[, ACy]	Yes	3	1	X
[6]	AND k16 << #16, [ACx,] ACy	No	4	1	X
[7]	AND k16 << #SHFT, [ACx,] ACy	No	4	1	X
[8]	AND k16, Smem	No	4	1	X

Brief 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.

Status Bits

Affected by C54CM, M40

Affects none

4.13.1 Bitwise AND: AND src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	AND src, dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs a bitwise AND operation between two registers content.

- 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.

Example

Syntax	Description
AND AC0, AC1	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         0F E340 5678    AC1         0E 2340 4640
```

4.13.2 Bitwise AND: AND k8, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	AND k8, src, dst	Yes	3	1	X

Operands dst, k8, src

Description

This instruction performs a bitwise AND 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.

Example

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

4.13.3 Bitwise AND: AND k16, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	AND k16, src, dst	No	4	1	X

Operands dst, k16, src

Description

This instruction performs a bitwise AND operation between a source (src) register content and a 16-bit value, k16.

- 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.

Example

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

4.13.4 Bitwise AND: AND Smem, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	AND Smem, src, dst	No	3	1	X

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) 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.

Example

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

4.13.5 Bitwise AND: AND ACx << #SHIFTW[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	AND ACx << #SHIFTW[, ACy]	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

This instruction performs a bitwise AND operation between an accumulator (ACy) content and a shifted accumulator (ACx) content.

- 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 operand (ACx) is shifted by an 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, M40

Affects none

Repeat

This instruction can be repeated.

Example

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

4.13.6 Bitwise AND: AND k16 << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	AND k16 << #16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, k16

Description

This instruction performs a bitwise AND operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The operation 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 the MSBs.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AND #FFFFh << #16, AC1, AC0	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.

4.13.7 Bitwise AND: AND k16 << #SHFT, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	AND k16 << #SHFT, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, k16, SHFT

Description

This instruction performs a bitwise AND operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- 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 operand (k16) is shifted by an immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AND #FFFFh << #15, AC1, AC0	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.

4.13.8 Bitwise AND: AND k16, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	AND k16, Smem	No	4	1	X

Operands

k16, Smem

Description

This instruction performs a bitwise AND operation between a memory (Smem) location and a 16-bit value, k16.

- The operation is performed on 16 bits in the A-unit ALU.
- The result is stored in memory.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
AND #0FC0, *AR1	The content addressed by AR1 is ANDed with the unsigned 16-bit value (FC0h) and the result is stored in the location addressed by AR1.

```
Before                  After
*AR1          5678      *AR1          0640
```

4.14 Bitwise OR

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	OR src, dst	Yes	2	1	X
[2]	OR k8, src, dst	Yes	3	1	X
[3]	OR k16, src, dst	No	4	1	X
[4]	OR Smem, src, dst	No	3	1	X
[5]	OR ACx << #SHIFTW[, ACy]	Yes	3	1	X
[6]	OR k16 << #16, [ACx,] ACy	No	4	1	X
[7]	OR k16 << #SHFT, [ACx,] ACy	No	4	1	X
[8]	OR k16, Smem	No	4	1	X

Brief 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, M40

Affects none

4.14.1 Bitwise OR: OR src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	OR src, dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs a bitwise OR operation between two registers content.

- 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.

Example

Syntax	Description
OR AC1, AC0	The content of AC0 is ORed with the content of AC1 and the result is stored in AC0.

4.14.2 Bitwise OR: OR k8, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	OR k8, src, dst	Yes	3	1	X

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 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.

Example

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

4.14.3 Bitwise OR: OR k16, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	OR k16, src, dst	No	4	1	X

Operands dst, k16, src

Description

This instruction performs a bitwise OR operation between a source (src) register content and a 16-bit value, k16.

- 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.

Example

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

4.14.4 Bitwise OR: OR Smem, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	OR Smem, src, dst	No	3	1	X

Operands dst, Smem, src

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 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.

Example

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

4.14.5 Bitwise OR: OR ACx << #SHIFTW[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	OR ACx << #SHIFTW[, ACy]	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

This instruction performs a bitwise OR operation between an accumulator (ACy) content and a shifted accumulator (ACx) content.

- The shift and OR 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 an 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, M40

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
OR AC0 << #4, AC1	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          0F E340 5678       AC1          0F F754 FE78
```

4.14.6 Bitwise OR: OR k16 << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	OR k16 << #16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, k16

Description

This instruction performs a bitwise OR operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The operation 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 the MSBs.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
OR #FFFFh << #16, AC1, AC0	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.

4.14.7 Bitwise OR: OR k16 << #SHFT, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	OR k16 << #SHFT, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, k16, SHFT

Description

This instruction performs a bitwise OR operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The shift and OR operations are performed in one cycle in the D-unit shifter.
- Input operands are zero extended to 40 bits.
- The input operand (k16) is shifted by an immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
OR #FFFFh << #15, AC1, AC0	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.

4.14.8 Bitwise OR: OR k16, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	OR k16, Smem	No	4	1	X

Operands

k16, Smem

Description

This instruction performs a bitwise OR operation between a memory (Smem) location and a 16-bit value, k16.

- The operation is performed on 16 bits in the A-unit ALU.
- The result is stored in memory.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
OR #0FC0h, *AR1	The content addressed by AR1 is ORed with the unsigned 16-bit value (FC0h) and the result is stored in the location addressed by AR1.

```
Before                  After
*AR1        5678        *AR1        5FF8
```

4.15 Bitwise XOR

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XOR src,]dst	Yes	2	1	X
[2]	XOR k8, src, dst	Yes	3	1	X
[3]	XOR k16, src, dst	No	4	1	X
[4]	XOR Smem, src, dst	No	3	1	X
[5]	XOR ACx << #SHIFTW[, ACy]	Yes	3	1	X
[6]	XOR k16 << #16, [ACx,] ACy	No	4	1	X
[7]	XOR k16 << #SHFT, [ACx,] ACy	No	4	1	X
[8]	XOR k16, Smem	No	4	1	X

Brief 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, M40

Affects none

4.15.1 Bitwise XOR: XOR src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XOR src, dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between two registers content.

- 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.

Example

Syntax	Description
XOR AC0, AC1	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          0F E340 5678   AC1          71 C015 19B8
```

4.15.2 Bitwise XOR: XOR k8, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	XOR k8, src, dst	Yes	3	1	X

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.

Example

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

4.15.3 Bitwise XOR: XOR k16, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	XOR k16, src, dst	No	4	1	X

Operands dst, k16, src

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and a 16-bit value, k16.

- 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.

Example

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

4.15.4 Bitwise XOR: XOR Smem, src, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	XOR Smem, src, dst	No	3	1	X

Operands dst, Smem, src

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between a source (src) register content and a memory (Smem) location.

- 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.

Example

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

4.15.5 Bitwise XOR: XOR ACx << #SHIFTW[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	XOR ACx << #SHIFTW[, ACy]	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACy) content and a shifted accumulator (ACx) content.

- 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 an 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, M40

Affects none

Repeat

This instruction can be repeated.

Example

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

4.15.6 Bitwise XOR: XOR k16 << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	XOR k16 << #16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, k16

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- The operation 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 the MSBs.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
XOR #FFFFh << #16, AC1, AC0	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.

4.15.7 Bitwise XOR: XOR k16 << #SHFT, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	XOR k16 << #SHFT, [ACx,] ACy	No	4	1	X

Operands ACx, ACy, k16, SHFT

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between an accumulator (ACx) content and a shifted 16-bit value, k16.

- 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) is shifted by an immediate value in the D-unit shifter.
- The CARRY status bit is not affected by the logical shift operation.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.15.8 Bitwise XOR: XOR k16, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	XOR k16, Smem	No	4	1	X

Operands

k16, Smem

Description

This instruction performs a bitwise exclusive-OR (XOR) operation between a memory (Smem) location and a 16-bit value, k16.

- The operation is performed on 16 bits in the A-unit ALU.
- The result is stored in memory.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
XOR #FFFFh, *AR3	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.

4.16 Branch Conditionally (BCC)

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	BCC l4, cond	No	2	6/5	R
[2]	BCC L8, cond	Yes	3	6/5	R
[3]	BCC L16, cond	No	4	6/5	R
[4]	BCC P24, cond	No	5	5/5	R

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

Brief 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 l4, 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.

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

4.16.1 Branch Conditionally: BCC l4, cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	BCC l4, cond	No	2	6/5	R

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

Operands

cond, l4

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 l4. 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.

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.

Examples

Syntax	Description
BCC branch, *AR0 != #0	The content of AR0 is not equal to 0, control is passed to the program address label defined by branch.

```
        BCC branch, *AR0 != #0
        … …                                  address:  004057
        … …
branch  ……                                           00405A
:
```

Before		After	
AR0	3000	AR0	3000
PC	004055	PC	00405A

Syntax	Description
BCC branch, TC1	TC1 is set to 1, control is passed to the program address label defined by branch.

```
        BCC branch, TC1
        … …                                  address:  00406C
        … …
branch  … …                                           00406E
:
```

Before		After	
TC1	1	TC1	1
PC	00406A	PC	00406E

4.16.2 Branch Conditionally: BCC Lx, cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[2]	BCC L8, cond	Yes	3	6/5	R
[3]	BCC L16, cond	No	4	6/5	R

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

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.

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.

Examples

Syntax	Description
BCC branch, *AR0 != #0	The content of AR0 is not equal to 0, control is passed to the program address label defined by branch.

```
branch  ……                                  00305A
:
        BCC branch, *AR0 != #0
        … …                          address: 004057
        … …
```

Before		After	
AR0	3000	AR0	3000
PC	004055	PC	00305A

Syntax	Description
BCC branch, TC1	TC1 is set to 1, control is passed to the program address label defined by branch.

```
branch  … …                                  00306E
:
        BCC branch, TC1
        … …                          address: 00406C
        … …
```

Before		After	
TC1	1	TC1	1
PC	00406A	PC	00306E

4.16.3 Branch Conditionally: BCC P24, cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[4]	BCC P24, cond	No	5	5/5	R

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

Operands

cond, P24

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 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.

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.

Examples

Syntax	Description
BCC branch, *AR0 != #0	The content of AR0 is not equal to 0, control is passed to the program address label defined by branch.

```
        .sect “code1”
        … …
        BCC branch, *AR0 != #0
        … …                                  address:  004057
        .sect “code2”
branch  ……                                           00F05A
:
```

Before		After	
AR0	3000	AR0	3000
PC	004055	PC	00F05A

Syntax	Description
BCC branch, TC1	TC1 is set to 1, control is passed to the program address label defined by branch.

```
        .sect “code1”
        BCC branch, TC1
        … …                                  address:  00406C
        .sect “code2”
branch  … …                                           00F06E
:
```

Before		After	
TC1	1	TC1	1
PC	00406A	PC	00F06E

4.17 Branch Unconditionally (B)

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

† Instructions [2] and [3] execute in 3 cycles if the addressed instruction is in the instruction buffer unit.

Brief 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

4.17.1 Branch Unconditionally: B ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	B ACx	No	2	10	X

Operands

ACx

Description

This instruction branches to a 24-bit program address defined by the content of the 24 lowest bits of an accumulator (ACx).

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
B AC0	Program control 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
```

4.17.2 Branch Unconditionally: B Lx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[2]	B L7	Yes	2	6	AD
[3]	B L16	Yes	3	6	AD

† Executes in 3 cycles if the addressed instruction is in the instruction buffer unit.

Operands

Lx

Description

This instruction branches to a program address defined by a program address label assembled into Lx.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

```
        B branch
        MOV #1, AC0                    address:  004044
        … …
branch  … …                                      006047
:
        MOV #0, AC0
```

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

4.17.3 Branch Unconditionally: B P24

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	B P24	No	4	5	D

Operands

P24

Description

This instruction branches to a program address defined by a program address label assembled into P24.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

```
        B branch
        MOV #1, AC0                    address:  004044
        … …
branch  … …                                      006047
:
        MOV #0, AC0
```

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

4.18 Branch on Auxiliary Register Not Zero (BCC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	BCC L16, ARn_mod != #0	No	4	6/5	AD

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

Operands ARn_mod, L16

Description

This instruction performs a conditional branch (selected auxiliary register content not equal to 0) of the program counter (PC). The program branch address is specified as a 16-bit signed offset, L16, relative to PC. Use this instruction to branch within a 64K-byte window centered on the current PC value.

The possible addressing operands can be grouped into three categories:

- ARx not modified (ARx as base pointer), some examples:
  *AR1; No modification or offset
  *AR1(#15); Use 16-bit immediate value (15) as offset
  *AR1(T0); Use content of T0 as offset
  *AR1(short(#4)); Use 3-bit immediate value (4) as offset
- ARx modified before being compared to 0, some examples:
  *–AR1; Decrement by 1 before comparison
  *+AR1(#20); Add 16-bit immediate value (20) before comparison
- ARx modified after being compared to 0, some examples:
  *AR1+; Increment by 1 after comparison
  *(AR1 – T1); Subtract content of T1 after comparison

1) The content of the selected auxiliary register (ARn) is premodified in the address generation unit.
2) The (premodified) content of ARn is compared to 0 and sets the condition in the address phase of the pipeline.
3) If the condition is not true, a branch occurs. If the condition is true, the instructions are executed in sequence.
4) The content of ARn is postmodified in the address generation 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 none

Affects none

Repeat

This instruction cannot be repeated.

Examples

Syntax	Description
BCC branch, *AR1(#6) != #0	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.

```
        BCC branch, *AR1(#6) != #0      address:  004004
        … …                             ;         00400A


        … …
branch  … …                             ;         00400C
:
```

Before		After	
AR1	0005	AR1	0005
PC	004004	PC	00400C

Syntax	Description
BCC branch, *AR3– != #0	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.

```
        BCC branch, *AR3– != #0          address:  00400F
        … …                              ;         004013
        … …
branch  … …                              ;         004015
:
```

Before		After	
AR3	0000	AR3	FFFF
PC	00400F	PC	004013

4.19 Call Conditionally (CALLCC)

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	CALLCC L16, cond	No	4	6/5	R
[2]	CALLCC P24, cond	No	5	5/5	R

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

Brief Description

These 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 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.

The instruction selection depends on the branch offset between the current PC value and program subroutine address specified by the label.

These instructions cannot be repeated.

Status Bits

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

4.19.1 Call Conditionally: CALLCC L16, cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	CALLCC L16, cond	No	4	6/5	R

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

Operands

cond, L16

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 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.

If a subroutine call occurs:

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The program counter (PC) is loaded with the subroutine program address. The active control flow execution context flags are cleared.

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

Syntax	Description
CALLCC (subroutine), AC1 >= #2000h	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.

4.19.2 Call Conditionally: CALLCC P24, cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[2]	CALLCC P24, cond	No	5	5/5	R

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

Operands

cond, P24

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.

If a subroutine call occurs:

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The program counter (PC) is loaded with the subroutine program address. The active control flow execution context flags are cleared.

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

Syntax	Description
CALLCC FOO, TC1	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.

4.20 Call Unconditionally (CALL)

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

Brief 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.

These instructions cannot be repeated.

Status Bits

Affected by none

Affects none

4.20.1 Call Unconditionally: CALL ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	CALL ACx	No	2	10	X

Operands

ACx

Description

This instruction passes control to a specified subroutine program address defined by the content of the 24 lowest bits of the accumulator, ACx.

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the called subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

4.20.2 Call Unconditionally: CALL L16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	CALL L16	Yes	3	6	AD

Operands

L16

Description

This instruction passes control to a specified subroutine program address defined by a program address label assembled into L16.

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the called subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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.

4.20.3 Call Unconditionally: CALL P24

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	CALL P24	No	4	5	D

Operands P24

Description

This instruction passes control to a specified subroutine program address defined by a program address label assembled into P24.

- The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 LSBs of RETA are pushed to the top of SP.
- The system stack pointer (SSP) is decremented by 1 word in the address phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- The return address of the called subroutine is saved in RETA. The active control flow execution context flags are saved in CFCT.
- The PC is loaded with the subroutine program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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

4.21 Compare and Branch (BCC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	BCC[U] L8, src RELOP K8	No	4	7/6	X

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

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 unsigned (U) keyword and, for accumulator comparisons, on M40.

- In the case of an unsigned comparison, the 8-bit constant, K8, is zero extended to:
  - 16 bits, if the source (src) operand is an auxiliary or temporary register.
  - 40 bits, if the source (src) operand is an accumulator.
- In the case of a signed comparison, the 8-bit constant, K8, is sign extended to:
  - 16 bits, if the source (src) operand is an auxiliary or temporary register.
  - 40 bits, if the source (src) operand is an accumulator.

As the following table shows, the U keyword specifies an unsigned comparison; M40 defines the comparison bit width of the accumulator.

U	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.

Examples

Syntax	Description
BCC branch, AC0 >= #12	The content of AC0 is compared to the signed 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).

```
        BCC branch, AC0 >= #12
        … …                                 address:  00 4075
        … …
branch  … …                                           00 4078
:
```

Before		After	
AC0	00 0000 3000	AC0	00 0000 3000
PC	004071	PC	004078

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

```
        BCC branch, T1 != #1
        … …                                 address:  00407D
        … …
branch  … …                                           004080
:
```

Before		After	
T1	0000	T1	0000
PC	4079	PC	407D

4.22 Compare and Select Extremum

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MAXDIFF ACx, ACy, ACz, ACw	Yes	3	1	X
[2]	DMAXDIFF ACx, ACy, ACz, ACw, TRNx	Yes	3	1	X
[3]	MINDIFF ACx, ACy, ACz, ACw	Yes	3	1	X
[4]	DMINDIFF ACx, ACy, ACz, ACw, TRNx	Yes	3	1	X

Brief Description

Instructions [1] and [3] perform two paralleled 16-bit extremum selections in the D-unit ALU. Instructions [2] and [4] perform a single 40-bit extremum selection in the D-unit ALU.

Status Bits

Affected by C54CM, M40, SATD

Affects ACOVw, CARRY

4.22.1 Compare and Select Extremum: MAXDIFF ACx, ACy, ACz, ACw

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MAXDIFF ACx, ACy, ACz, ACw	Yes	3	1	X

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 dual 16-bit arithmetic operation 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.
- 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

Syntax	Description
MAXDIFF AC0, AC1, AC2, AC1	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

4.22.2 Compare and Select Extremum: DMAXDIFF ACx, ACy, ACz, ACw, TRNx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	DMAXDIFF ACx, ACy, ACz, ACw, TRNx	Yes	3	1	X

Operands ACw, ACx, ACy, ACz, TRNx

Description

This instruction performs a single 40-bit extremum selection in the D-unit ALU. This instruction performs a maximum search.

- ACx and ACy are the two source accumulators.
- The difference between the source accumulators is stored in accumulator ACw.
- The subtraction computation is identical 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 computation is similar to the maximum instruction. However, the CARRY status bit is not updated by the extremum search but by the subtraction instruction.
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs. If the extremum value is ACx, the decision bit is 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) }
```

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.

Example

Syntax	Description
DMAXDIFF AC0, AC1, 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	F0 5C00 BBBC
M40	1	M40	1
SATD	1	SATD	1
TRN1	0080	TRN1	0040
ACOV3	0	ACOV3	0
CARRY	0	CARRY	0

4.22.3 Compare and Select Extremum: MINDIFF ACx, ACy, ACz, ACw

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MINDIFF ACx, ACy, ACz, ACw	Yes	3	1	X

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 dual 16-bit arithmetic operation 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.
- 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

Syntax	Description
MINDIFF AC0, AC1, AC2, AC1	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 stored 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

4.22.4 Compare and Select Extremum: DMINDIFF ACx, ACy, ACz, ACw, TRNx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	DMINDIFF ACx, ACy, ACz, ACw, TRNx	Yes	3	1	X

Operands ACw, ACx, ACy, ACz, TRNx

Description

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 source accumulators.
- The difference between the source accumulators is stored in accumulator ACw.
- The subtraction computation is identical 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 computation is similar to the maximum instruction. However, the CARRY status bit is not updated by the extremum search but by the subtraction instruction.
- According to the extremum found, a decision bit is shifted in TRNx from the MSBs to the LSBs. If the extremum value is ACx, the decision bit is 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) }
```

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.

Example

Syntax	Description
DMINDIFF 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.

4.23 Conditional Addition or Subtraction (ADDSUBCC)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy	No	3	1	X
[2]	ADDSUBCC Smem, ACx, TCx, ACy	No	3	1	X
[3]	ADDSUBCC Smem, ACx, TC1, TC2, ACy	No	3	1	X

Brief Description

This instruction evaluates the selected TCx status bit and based on the result of the test, an addition, a move, or a subtraction is performed. Evaluation of the condition on the TCx status bit is performed during the Execute phase of the instruction.

Status Bits

Affected by C54CM, M40, SATD, SXMD, TC1, TC2

Affects ACOVy, CARRY

4.23.1 Conditional Addition or Subtraction: ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ADDSUB2CC Smem, ACx, Tx, TC1, T2, ACy	No	3	1	X2

Operands ACx, ACy, Tx, Smem, TC1, TC2

Description

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	TC2	Operation
0	0	ACy = ACx – (Smem << Tx)
0	1	ACy = ACx – (Smem << #16)
1	0	ACy = ACx + (Smem << Tx)
1	1	ACy = ACx + (Smem << #16)

If TC2 = 1 and TC1 = 1, 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.

If TC2 = 0 and TC1 = 1, 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.

- ❏ 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 identical 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.

If TC2 = 1 and TC1 = 0, 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.

If TC2 = 0 and TC1 = 0, then ACy = ACx – (Smem << Tx): 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.

- ❑ 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 identical 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:

- ❑ 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.

Example

Syntax

ADDSUB2CC *AR2, AC0, T1, TC1, TC2, AC2

Description

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 EC00 0000	AC0	00 EC00 0000
AC2	00 0000 0000	AC2	00 EC00 CC00
AR2	0201	AR2	0201
201	3300	201	3300
T1	0002	T1	0002
TC1	1	TC1	1
TC2	0	TC2	0
M40	0	M40	0
ACOV2	0	ACOV2	1
CARRY	0	CARRY	0

4.23.2 Conditional Addition or Subtraction: ADDSUBCC Smem, ACx, TCx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ADDSUBCC Smem, ACx, TCx, ACy	No	3	1	X

Operands ACx, ACy, Smem, TCx

Description

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.

If TCx = 1, 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 ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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.

If TCx = 0, 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 ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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.

Examples

Syntax	Description
ADDSUBCC *AR3, AC1, TC1, AC0	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	Description
ADDSUBCC *AR1, AC0, TC2, AC1	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
M40	0	M40	0
ACOV1	0	ACOV1	1
CARRY	0	CARRY	1

4.23.3 Conditional Addition or Subtraction: ADDSUBCC Smem, ACx, TC1, TC2, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ADDSUBCC Smem, ACx, TC1, T2, ACy	No	3	1	X

Operands ACx, ACy, Smem, TC1, TC2

Description

This instruction evaluates the selected TCx status bit and based on the result of the test, an addition, a move, or a subtraction is performed. Evaluation of the condition on the TCx status bit is performed during the Execute phase of the instruction.

TC1	TC2	Operation
0	0	ACy = ACx – (Smem << #16)
0	1	ACy = ACx
1	0	ACy = ACx + (Smem << #16)
1	1	ACy = ACx

If TC2 = 1, then ACy = ACx: this instruction moves the content of ACx to ACy.

- ❏ 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:
  - ■ the destination accumulator overflow status bit (ACOVx) is set.
  - ■ the destination register (ACx) is saturated according to SATD.

If TC2 = 0 and TC1 = 1, 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 ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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.

If TC2 = 0 and TC1 = 0, 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 ALU.

- Input operands are sign extended to 40 bits according to SXMD.
- The shift operation is identical 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.

Example

Syntax	Description
ADDSUBCC *AR3, AC1, TC1, TC2, AC0	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.

4.24 Conditional Shift (SFTCC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SFTCC ACx, TCx	Yes	2	1	X

Operands ACx, TCx

Description

If the source accumulator ACx(39–0) is equal to 0, this instruction sets the TCx status bit to 1.

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

If the source accumulator ACx(31–0) does not have two sign bits, this instruction sets the TCx status bit to 1.

The sign bits are extracted at bit positions 31 and 30.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTCC AC0, TC1	Because AC0(31) XORed with AC0(30) equals 1, the content of AC0 is not shifted left and TC1 is set to 1.

```
Before                    After
AC0        FF 8765 0055   AC0        FF 8765 0055
TC1                   0   TC1                   1
```

Syntax	Description
SFTCC 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
```

4.25 Conditional Subtract (SUBC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SUBC Smem, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction performs a conditional subtraction in the D-unit ALU. The D-unit shifter is not used to perform the memory operand shift.

- The 16-bit data memory operand Smem is sign extended to 40 bits according to SXMD, shifted left by 15 bits, and subtracted from the content of the source accumulator ACx.
  - The shift operation is identical to the signed shift instruction.
  - Overflow and CARRY bit is always detected at bit position 31. The subtraction borrow bit is reported in the CARRY status bit; the borrow bit is the logical complement of the CARRY status bit.
  - If an overflow is detected and reported in accumulator overflow bit ACOVy, no saturation is performed on the result of the operation.
- If the result of the subtraction is greater than 0 (bit 39 = 0), the result is shifted left by 1 bit, added to 1, and stored in the destination accumulator ACy.
- If the result of the subtraction is less than 0 (bit 39 = 1), the source accumulator ACx is shifted left by 1 bit and stored in the destination accumulator ACy.

```
if ((ACx – (Smem << #15)) >= 0)
    ACy = (ACx – (Smem << #15)) << #1 + 1
else
    ACy = ACx << #1
```

This instruction is used to make a 16 step 16-bit by 16-bit division. The divisor and the dividend are both assumed to be positive in this instruction. SXMD affects this operation:

- If SXMD = 1, the divisor must have a 0 value in the most significant bit
- If SXMD = 0, any 16-bit divisor value produces the expected result

The dividend, which is in the source accumulator ACx, must be positive (bit 31 = 0) during the computation.

Status Bits

Affected by SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Examples

Syntax	Description
SUBC *AR1, AC0, AC1	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

Syntax	Description
repeat (CSR) SUBC *AR1, 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 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 result generated a carry.

Before		After	
AC1	00 0746 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

4.26 Dual 16-Bit Arithmetic

	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ADDSUB Tx, Smem, ACx	No	3	1	X
[2]	SUBADD Tx, Smem, ACx	No	3	1	X
[3]	ADD dual(Lmem), [ACx,] ACy	No	3	1	X
[4]	SUB dual(Lmem), [ACy,] ACy	No	3	1	X
[5]	SUB ACx, dual(Lmem), ACy	No	3	1	X
[6]	SUB dual(Lmem), Tx, ACx	No	3	1	X
[7]	ADD dual(Lmem), Tx, ACx	No	3	1	X
[8]	SUB Tx, dual(Lmem), ACx	No	3	1	X
[9]	ADDSUB Tx, dual(Lmem), ACx	No	3	1	X
[10]	SUBADD Tx, dual(Lmem), ACx	No	3	1	X

Brief Description

These instructions perform two paralleled arithmetical operations in one cycle:

- ❑ addition and subtraction
- ❑ subtraction and addition
- ❑ two additions
- ❑ two subtractions

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

4.26.1 Parallel 16-Bit Addition and Subtraction: ADDSUB Tx, Smem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ADDSUB Tx, Smem, ACx	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction performs two paralleled arithmetical operations in one cycle: an addition and subtraction. 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).

- The data memory operand Smem:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- 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 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.
- 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.

Example

Syntax	Description
ADDSUB T1, *AR1, AC1	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
T1	4000	T1	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

4.26.2 Parallel 16-Bit Subtraction and Addition: SUBADD Tx, Smem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	SUBADD Tx, Smem, ACx	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction performs two paralleled arithmetical operations in one cycle: a subtraction and addition. 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).

- The data memory operand Smem:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- 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 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.
- 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.

Example

Syntax	Description
SUBADD T0, *AR3, AC0	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).

4.26.3 Parallel 16-Bit Additions: ADD dual(Lmem), [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	ADD dual(Lmem), [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction performs two paralleled 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).

- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
ADD dual(*AR3), AC1, AC0	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).

4.26.4 Parallel 16-Bit Subtractions: SUB dual(Lmem), [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	SUB dual(Lmem), [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction performs two paralleled 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 data path).

- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
SUB dual(*AR3), AC1, AC0	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).

4.26.5 Parallel 16-Bit Subtractions: SUB ACx, dual(Lmem), ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	SUB ACx, dual(Lmem), ACy	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction performs two paralleled 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).

- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
SUB AC1, dual(*AR3), AC0	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).

4.26.6 Parallel 16-Bit Subtractions: SUB dual(Lmem), Tx, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	SUB dual(Lmem), Tx, ACx	No	3	1	X

Operands ACx, Tx, Lmem

Description

This instruction performs two paralleled 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).

- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
SUB dual(*AR3), T0, AC0	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).

4.26.7 Parallel 16-Bit Additions: ADD dual(Lmem), Tx, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	ADD dual(Lmem), Tx, ACx	No	3	1	X

Operands ACx, Tx, Lmem

Description

This instruction performs two paralleled 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).

- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
ADD dual(*AR3), T0, AC0	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).

4.26.8 Parallel 16-Bit Subtractions: SUB Tx, dual(Lmem), ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	SUB Tx, dual(Lmem), ACx	No	3	1	X

Operands ACx, Tx, Lmem

Description

This instruction performs two paralleled 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).

- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
SUB T0, dual(*AR3), AC0	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).

4.26.9 Parallel 16-Bit Addition and Subtraction: ADDSUB Tx, dual(Lmem), ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	ADDSUB Tx, dual(Lmem), ACx	No	3	1	X

Operands ACx, Tx, Lmem

Description

This instruction performs two paralleled arithmetical operations in one cycle: an addition and subtraction. 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).

- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
ADDSUB T0, dual(*AR3), AC0	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).

4.26.10 Parallel 16-Bit Subtraction and Addition: SUBADD Tx, dual(Lmem), ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	SUBADD Tx, dual(Lmem), ACx	No	3	1	X

Operands ACx, Tx, Lmem

Description

This instruction performs two paralleled arithmetical operations in one cycle: a subtraction and addition. 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).

- The temporary register Tx:
  - is used as one of the 16-bit operands of the ALU low part
  - is duplicated and, according to SXMD, sign extended to 24 bits to be used in the ALU high part
- The data memory operand dbl(Lmem) is divided into two 16-bit parts:
  - the lower part is used as one of the 16-bit operands of the ALU low part
  - the higher part is sign extended to 24 bits according to SXMD and is used in the ALU high part
- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least significant word = Lmem – 1
- 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 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.
- 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.

Example

Syntax	Description
SUBADD T0, dual(*AR3), AC0	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).

4.27 Dual Multiply (Accumulate/Subtract)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[2]	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[3]	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[4]	AMAR Xmem :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx	No	4	1	X
[5]	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[6]	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[7]	AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx	No	4	1	X
[8]	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[9]	AMAR Xmem :: MAS[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx	No	4	1	X
[10]	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16 :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X
[11]	MPY[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16	No	4	1	X
[12]	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx >> #16 :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16	No	4	1	X
[13]	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy >> #16	No	4	1	X
[14]	AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx >> #16	No	4	1	X
[15]	AMAR Xmem, Ymem, Cmem	No	4	1	X

Brief Description

These instructions perform the following parallel operations in one cycle:

- ❑ two multiplies
- ❑ multiply and accumulate (MAC), and multiply
- ❑ multiply and subtract (MAS), and multiply
- ❑ modify auxiliary register (MAR) and multiply
- ❑ two multiply and accumulates (MACs)
- ❑ multiply and subtract (MAS), and multiply and accumulate (MAC)
- ❑ modify auxiliary register (MAR), and multiply and accumulate (MAC)
- ❑ two multiply and subtracts (MASs)
- ❑ modify auxiliary register (MAR), and multiply and subtract (MAS)
- ❑ multiply, and multiply and accumulate (MAC)
- ❑ three modify auxiliary registers (MARs)

Status Bits

Affected by	FRCT, SMUL, M40, RDM, SATD
Affects	ACOVx, ACOVy

4.27.1 Parallel Multiplies: MPY[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MPY[R][40] [uns()Ymem()], [uns()Cmem()], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MPY[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MPY[R][40] [uns()Ymem()], [uns()Cmem()], ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel multiply operations in one cycle. The operations are executed in the two D-unit MACs.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

This second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPY uns(*AR3), uns(*CDP), AC0 :: MPY uns(*AR4), uns(*CDP), AC1	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.

4.27.2 Parallel MAC and Multiply: MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: multiply and accumulate (MAC), and multiply. The operations are executed in the two D-unit MACs.

The first 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 Xmem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

This second operation performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Ymem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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 added to the source accumulator ACx.
- ❏ For the second operation, 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MAC uns(*AR3), uns(*CDP), AC0 :: MPY uns(*AR4), uns(*CDP), AC1	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.

4.27.3 Parallel MAS and Multiply: MAS[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MPY[R][40] [uns()Ymem()], [uns()Cmem()], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MAS[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MPY[R][40] [uns()Ymem()], [uns()Cmem()], ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: multiply and subtract (MAS), and multiply. 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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
- ❏ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MAS uns(*AR3), uns(*CDP), AC0 :: MPY uns(*AR4), uns(*CDP), AC1	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.

4.27.4 Parallel MAR and Multiply: AMAR Xmem :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	AMAR Xmem :: MPY[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx	No	4	1	X

Operands ACx, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR) and multiply. The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❏ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AMAR *AR3+ :: MPY uns(*AR4), uns(*CDP), AC0	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.

4.27.5 Parallel MACs: MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MAC[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first 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 Xmem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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.
- ❏ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MAC uns(*AR3), uns(*CDP), AC0 :: MAC uns(*AR4), uns(*CDP), AC1	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.

4.27.6 Parallel MAS and MAC: MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MAS[R][40] [uns(]Xmem[)], [uns(]Cmem[)], ACx :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACy	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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.
- ❏ 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASR40 uns(*AR0), uns(*CDP), AC0 :: MACR40 uns(*AR1), uns(*CDP), AC1	Both instructions are performed in parallel. The unsigned content addressed by AR0 multiplied by the unsigned content addressed by the coefficient 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	*CDP	A067
ACOV0	0	ACOV0	0
ACOV1	0	ACOV1	0
CARRY	0	CARRY	0
FRCT	0	FRCT	0

4.27.7 Parallel MAR and MAC: AMAR Xmem :: MAC[R][40] [uns()Ymem[]], [uns()Cmem[]], ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	AMAR Xmem :: MAC[R][40] [uns()Ymem[]], [uns()Cmem[]], ACx	No	4	1	X

Operands ACx, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❏ 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.
- ❏ 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AMAR *AR3+ :: MAC uns(*AR4), uns(*CDP), AC0	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.

4.27.8 Parallel MASs: MAS[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MAS[R][40] [uns()Ymem()], [uns()Cmem()], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	MAS[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MAS[R][40] [uns()Ymem()], [uns()Cmem()], ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel multiply and subtract (MAS) operations in one cycle. 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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.
- ❏ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MAS uns(*AR3), uns(*CDP), AC0 :: MAS uns(*AR4), uns(*CDP), AC1	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.

4.27.9 Parallel MAR and MAS: AMAR Xmem :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	AMAR Xmem :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	No	4	1	X

Operands ACx, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and subtract (MAS). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AMAR *AR3+ :: MAS uns(*AR4), uns(*CDP), AC0	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.

4.27.10 Parallel MACs: MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first 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 Xmem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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 added to the source accumulator ACx shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- ❏ 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.
- ❏ 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MAC uns(*AR3), uns(*CDP), AC0 >> #16 :: MAC uns(*AR4), uns(*CDP), AC1	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.

4.27.11 Parallel Multiply and MAC: MPY[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MAC[R][40] [uns()Ymem()], [uns()Cmem()], ACy >> #16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	MPY[R][40] [uns()Xmem()], [uns()Cmem()], ACx :: MAC[R][40] [uns()Ymem()], [uns()Cmem()], ACy >> #16	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: multiply, and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- 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.
- For the second operation, the 32-bit result of the multiplication is sign extended to 40 bits and added to the source accumulator ACy shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
- 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	**Description**
MPY uns(*AR3), uns(*CDP), AC0 :: MAC uns(*AR4), uns(*CDP), AC1 >> #16	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.

4.27.12 Parallel MACs: MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel multiply and accumulate (MAC) operations in one cycle. The operations are executed in the two D-unit MACs.

The first 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 Xmem, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❏ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❏ 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator bit 39.
- ❏ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	**Description**
MAC uns(*AR3), uns(*CDP), AC0 >> #16 :: MAC uns(*AR4), uns(*CDP), AC1 >> #16	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.

4.27.13 Parallel MAS and MAC: MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	No	4	1	X

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign 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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACy(39).
- 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	**Description**
MAS uns(*AR3), uns(*CDP), AC0 :: MAC uns(*AR4), uns(*CDP), AC1 >> #16	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.

4.27.14 Parallel MAR and MAC: AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx >> #16

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	AMAR Xmem :: MAC[R][40] [uns(]Ymem[)], [uns(]Cmem[)], ACx >> #16	No	4	1	X

Operands ACx, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations in one cycle: modify auxiliary register (MAR), and multiply and accumulate (MAC). The operations are executed in the two D-unit MACs.

The first operation performs an auxiliary register modification. The auxiliary register modification is specified by the content of data memory operand Xmem.

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, sign extended to 17 bits, and the content of a data memory operand Cmem, addressed using the coefficient addressing mode, sign extended to 17 bits.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand. When Xmem memory operand is defined as unsigned, Ymem should also be defined as unsigned (and reciprocally).
- ❑ 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- ❑ 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 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:

- AMAR Xmem
- AMAR Ymem
- AMAR Cmem

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
AMAR *AR2+ :: MAC uns(*AR1), uns(*CDP), AC0 >> #16	Both instructions are performed in parallel. AR2 is incremented 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 is detected in AC0.

Before		After	
AC0	00 6900 0000	AC0	00 95C0 9200
AC1	00 0023 0000	AC1	00 0023 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

4.27.15 Parallel MARs: AMAR Xmem, Ymem, Cmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	AMAR Xmem, Ymem, Cmem	No	4	1	X

Operands

Cmem, Xmem, Ymem

Description

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 repeated.

Example

Syntax	Description
AMAR *AR3+, *AR4–, *CDP	AR3 is incremented by 1. AR4 is decremented by 1. CDP is not modified.

4.28 Execute Conditionally (XCC)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XCC [label,]cond	No	2	1	AD
[2]	XCCPART [label,]cond	No	2	1	X

Brief Description

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. Instruction [1] allows you to control the entire execution flow from the address phase to the execute phase of the pipeline. Instruction [2] allows you to only control the execution flow from the execute phase of the pipeline. The conditions that can be tested are defined by the cond field. The use of a label, where control of the execute conditionally instruction ends, is optional.

- ❑ These instructions may be executed alone.
- ❑ These instructions may be executed with two paralleled instructions.
- ❑ These instructions may be executed with the instruction with which it is paralleled.
- ❑ These instructions may be executed with the previous instruction.
- ❑ These instructions may be executed with the previous instruction and two paralleled instructions.
- ❑ These instructions cannot be repeated.
- ❑ These instructions cannot control the execution of the following program control instructions:

B (branch)	BCC	IDLE	INTR	XCC	RESET
CALL	CALLCC	RPT	RPTCC	XCCPART	TRAP
RET	RETCC	RETI	RPTB	RPTBLOCAL	

Status BIts

Affected by ACOVx, CARRY, C54CM, M40, TCx

Affects ACOVx

4.28.1 Execute Conditionally (Address Phase): XCC [label,]cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	XCC [label,]cond	No	2	1	AD

Operands

cond

Description

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 address phase to the execute phase of the pipeline.

When this instruction moves into the address phase of the pipeline, the condition specified in the cond field is evaluated. If the tested condition is true, the conditional instruction(s) is read and executed; if the tested condition is false, the conditional instruction(s) is not read and program control is passed to the instruction following the conditional instruction(s) or to the program address defined by label. There is a 3-cycle latency for the condition testing.

- This instruction may be executed alone:

```
      XCC [label, ]cond
      instruction_executes_conditionally
[label:]
```

- This instruction may be executed with two paralleled instructions:

```
      XCC [label, ]cond
      instruction_1_executes_conditionally
      || instruction_2_executes_conditionally
[label:]
```

- This instruction may be executed with the instruction with which it is paralleled:

```
      XCC [label, ]cond
      || instruction_executes_conditionally
[label:]
```

- This instruction may be executed with a previous instruction:

```
      previous_instruction
      || XCC [label, ]cond
      instruction_executes_conditionally
[label:]
```

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

```
      previous_instruction
      || XCC [label, ]cond
      instruction_1_executes_conditionally
      || instruction_2_executes_conditionally
[label:]
```

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

B (branch)	BCC	IDLE	INTR	XCC	RESET
CALL	CALLCC	RPT	RPTCC	XCCPART	TRAP
RET	RETCC	RETI	RPTB	RPTBLOCAL	

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.

Examples

Syntax	Description
XCC branch, *AR0 != #0 ADD *AR2+, AC0	The content of AR0 is not equal to 0, the next (ADD) instruction is executed. The content of AC0 is added to the content addressed by AR2 and the result is stored in AC0. AR2 is incremented by 1.

Before		After	
AR0	3000	AR0	3000
AR2	0405	AR2	0406
405	EF00	405	EF00
AC0	00 0000 000C	AC0	00 0000 EF0C

Syntax	Description
XCC *AR0 != #0 ADD *AR2+, AC0	The content of AR0 is equal to 0, the next (ADD) instruction is not executed and control is passed to the instruction following the conditionally executed (ADD) instruction.

Before		After	
AR0	0000	AR0	0000
AR2	0405	AR2	0405
405	EF00	405	EF00
AC0	00 0000 000C	AC0	00 0000 000C

4.28.2 Execute Conditionally (Execute Phase): XCCPART [label,]cond

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	XCCPART [label,]cond	No	2	1	X

Operands

cond

Description

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.

When this instruction moves into the execute phase of the pipeline, the condition specified in the cond field is evaluated. If the tested condition is true, the conditional instruction(s) is read and executed; if the tested condition is false, the conditional instruction(s) is not read and program control is passed to the instruction following the conditional instruction(s) or to the program address defined by label. There is a 0-cycle latency for the condition testing.

- This instruction may be executed alone:

```
      XCCPART [label, ]cond
      instruction_executes_conditionally
[label:]
```

- This instruction may be executed with two paralleled instructions:

```
      XCCPART [label, ]cond
      instruction_1_executes_conditionally
      || instruction_2_executes_conditionally
[label:]
```

- This instruction may be executed with the instruction with which it is paralleled. When this instruction syntax is used and the instruction to be executed conditionally is a store-to-memory instruction, there is a 1-cycle latency for the condition setting.

```
      XCCPART [label, ]cond
      || instruction_executes_conditionally
[label:]
```

- This instruction may be executed with a previous instruction:

```
      previous_instruction
      || XCCPART [label, ]cond
      instruction_executes_conditionally
[label:]
```

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

```
       previous_instruction
       || XCCPART [label, ]cond
       instruction_1_executes_conditionally
       || instruction_2_executes_conditionally
[label:]
```

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

B (branch)	BCC	IDLE	INTR	XCC	RESET
CALL	CALLCC	RPT	RPTCC	XCCPART	TRAP
RET	RETCC	RETI	RPTB	RPTBLOCAL	

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.

Examples

Syntax	Description
XCCPART branch, *AR0 != #0 ADD *AR2+, AC0	The content of AR0 is not equal to 0, the next (ADD) instruction is executed. The content of AC0 is added to the content addressed by AR2 and the result is stored in AC0. AR2 is incremented by 1.

Before		After	
AR0	3000	AR0	3000
AR2	0405	AR2	0406
405	EF00	405	EF00
AC0	00 0000 000C	AC0	00 0000 EF0C

Syntax	Description
XCCPART *AR0 != #0 ADD *AR2+, AC0	The content of AR0 is equal to 0, the next (ADD) instruction is not executed and control is passed to the instruction following the conditionally executed (ADD) instruction; however, since the next (ADD) instruction includes a pointer modification, AR2 is incremented by 1 in the address phase.

Before		**After**	
AR0	0000	AR0	0000
AR2	0405	AR2	0406
405	EF00	405	EF00
AC0	00 0000 000C	AC0	00 0000 000C

4.29 Extended Auxiliary Register Move

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV xsrc, xdst	No	2	1	X

Operands xdst, xsrc

Description

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, or 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, or 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) are accumulators, the temporary register move instruction (MOV src, dst) is assembled.

Status BIts

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0, XAR1	The lower 23 bits of AC0 are loaded into XAR1.

4.30 Finite Impulse Response Filter, Symmetrical/Antisymmetrical,

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	FIRSADD Xmem, Ymem, Cmem, ACx, ACy	No	4	1	X
[2]	FIRSSUB Xmem, Ymem, Cmem, ACx, ACy	No	4	1	X

Brief Description

These instructions perform two parallel operations in one cycle.

❑ The FIRSADD operation is executed:

```
ACy = ACy + (ACx * Cmem)
:: ACx = (Xmem << #16) + (Ymem << #16)
```

❑ The FIRSSUB operation is executed:

```
ACy = ACy + (ACx * Cmem)
:: ACx = (Xmem << #16) – (Ymem << #16)
```

Status BIts

Affected by FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.30.1 Symmetrical FIR Filter: FIRSADD Xmem, Ymem, Cmem, ACx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	FIRSADD Xmem, Ymem, Cmem, ACx, ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations: multiply and accumulate (MAC), and addition. The operation is executed:

```
ACy = ACy + (ACx * Cmem)
:: ACx = (Xmem << #16) + (Ymem << #16)
```

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 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 saturated according to SATD.

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.

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 40 bits in the D-unit ALU.
- ❏ Input operands are sign extended to 40 bits according to SXMD.
- ❏ The shift operation is identical 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, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
FIRSADD *AR0, *AR1, *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		After	
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	ACOV0	0
ACOV1	0	ACOV1	0
CARRY	0	CARRY	1
FRCT	0	FRCT	0
SXMD	0	SXMD	0

4.30.2 Antisymmetrical FIR Filter: FIRSSUB Xmem, Ymem, Cmem, ACx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	FIRSSUB Xmem, Ymem, Cmem, ACx, ACy	No	4	1	X

Operands ACx, ACy, Cmem, Xmem, Ymem

Description

This instruction performs two parallel operations: multiply and accumulate (MAC), and subtraction. The operation is executed:

```
ACy = ACy + (ACx * Cmem)
:: ACx = (Xmem << #16) – (Ymem << #16)
```

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 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 saturated according to SATD.

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.

The second operation subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, 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 identical 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, no overflow detection, report, and saturation is done after the shifting operation.

Status Bits

Affected by FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
FIRSSUB *AR0, *AR1, *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 AR1 shifted left by 16 bits is subtracted from the content addressed by AR0 shifted left by 16 bits and the result is stored in AC0.

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

4.31 Idle

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	IDLE	No	4	?	D

Operands none

Description

This instruction forces the program being executed to wait until an interrupt or a reset occurs. The power-down mode that the processor operates in depends on a configuration register accessible through the peripheral access mechanism.

Status BIts

Affected by INTM

Affects none

Repeat

This instruction cannot be repeated.

4.32 Implied Paralleled Instructions

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X
[2]	MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X
[3]	MASM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X
[4]	ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	No	4	1	X
[5]	SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	No	4	1	X
[6]	MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X
[7]	MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	No	4	1	X
[8]	MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	No	4	1	X

Brief Description

This instruction performs the following parallel operations:

- ❑ multiply and store
- ❑ multiply and accumulate (MAC), and store
- ❑ multiply and subtract (MAS), and store
- ❑ addition and store
- ❑ subtraction and store
- ❑ load and store
- ❑ multiply and accumulate (MAC), and load
- ❑ multiply and subtract (MAS), and load

Status BIts

Affected by FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.32.1 Parallel Multiply and Store: MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X

Operands ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and 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 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 R keyword is applied to the instruction.
- ❑ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an overflow is detected, ACOVy is saturated according to SATD.
- ❑ This instruction provides the option to store the 16-bit data-memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31–16) to data memory operand Ymem. If the 16-bit value in T2 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.
- ❑ 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 FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYMR *AR0+, T0, AC1 :: MOV HI(AC0 << T2), *AR1+	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 0000 0000	AC1	00 2000 0000
AR0	0200	AR0	0201
AR1	0300	AR1	0301
T0	4000	T0	4000
T2	0004	T2	0004
200	4000	200	4000
300	1111	300	4211
FRCT	1	FRCT	1
ACOV1	0	ACOV1	0
CARRY	0	CARRY	0

4.32.2 Parallel MAC and Store: MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X

Operands ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and accumulate (MAC), and store.

The first operation performs a multiplication and an accumulation 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 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 R keyword is applied to the instruction.
- ❏ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❏ When an addition overflow is detected, ACOVy is saturated according to SATD.
- ❏ This instruction provides the option to store the 16-bit data-memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31–16) to data memory operand Ymem. If the 16-bit value in T2 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.
- ❏ 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 FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACM *AR3, T0, AC0 :: MOV HI(AC1 << T2), *AR4	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.

4.32.3 Parallel MAS and Store: MASM[R] [T3 =]Xmem, Tx, ACy, :: MOV HI(ACx << T2), Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MASM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X

Operands ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and subtract (MAS), and store.

The first operation 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 data memory operand Xmem, 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 ACy.
- ❏ Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- ❏ Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❏ When an overflow is detected, ACOVy is saturated according to SATD.
- ❏ This instruction provides the option to store the 16-bit data-memory operand Xmem in temporary register T3.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31–16) to data memory operand Ymem. If the 16-bit value in T2 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.
- ❏ 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 FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASM *AR3, T0, AC0, :: MOV HI(AC1 << T2), *AR4	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.

4.32.4 Parallel Addition and Store: ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	No	4	1	X

Operands ACx, ACy, T2, Xmem, Ymem

Description

This instruction performs two operations in parallel: addition and store.

The first operation performs an addition between an accumulator content and the content of data memory operand Xmem shifted left by 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 identical to the signed shift instruction.
- ❏ Overflow detection and CARRY status bit depends on M40. 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.
- ❏ When an overflow is detected, the accumulator is saturated according to SATD.

The second operation shifts the accumulator ACy by the content of T2 and stores ACy(31–16) to data memory operand Ymem. If the 16-bit value in T2 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.
- ❏ After the shift, the high part of the accumulator, ACy(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, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
ADD *AR3 << #16, AC1, AC0 :: MOV HI(AC0 << T2), *AR4	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.

4.32.5 Parallel Subtraction and Store: SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	No	4	1	X

Operands ACx, ACy, T2, Xmem, Ymem

Description

This instruction performs two operations in parallel: subtraction and store.

The first operation subtracts an accumulator content from the content of data memory operand Xmem shifted left by 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 identical 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 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.
- ❏ When an overflow is detected, the accumulator is saturated according to SATD.

The second operation shifts the accumulator ACy by the content of T2 and stores ACy(31–16) to data memory operand Ymem. If the 16-bit value in T2 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.
- ❏ After the shift, the high part of the accumulator, ACy(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, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB *AR3 << #16, AC1, AC0 :: MOV HI(AC0 << T2), *AR4	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.

4.32.6 Parallel Load and Store: MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem	No	4	1	X

Operands ACx, ACy, T2, Xmem, Ymem

Description

This instruction performs two operations in parallel: 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 identical to the signed shift instruction.
- ❏ The input operand is shifted left by 16 bits according to M40.

The second operation shifts the accumulator ACx by the content of T2 and stores ACx(31–16) to data memory operand Ymem. If the 16-bit value in T2 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.
- ❏ 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

This instruction can be repeated.

Example

Syntax	Description
MOV *AR3 << #16, AC0 :: MOV HI(AC1 << T2), *AR4	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.

4.32.7 Parallel MAC and Load: MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	No	4	1	X

Operands ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and accumulate (MAC), and load.

The first operation performs a multiplication and an accumulation 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 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition overflow is detected, ACOVx is saturated according to SATD.
- This instruction provides the option to store the 16-bit data-memory operand Xmem in temporary register T3.

The second operation loads the content of data memory operand Ymem 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 identical to the signed shift instruction.
- The input operand is shifted left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACM *AR3, T0, AC0 :: MOV *AR4 << #16, AC1	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.

4.32.8 Parallel MAS and Load: MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	No	4	1	X

Operands ACx, ACy, Tx, Xmem, Ymem

Description

This instruction performs two operations in parallel: multiply and subtract (MAS), and load.

The first operation 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 data memory operand Xmem, 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 R keyword is applied to the instruction.
- Overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an overflow is detected, ACOVx is saturated according to SATD.
- This instruction provides the option to store the 16-bit data-memory operand Xmem in temporary register T3.

The second operation loads the content of data memory operand Ymem 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 identical to the signed shift instruction.
- The input operand is shifted left by 16 bits according to M40.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASM *AR3, T0, AC0 :: MOV *AR4 << #16, AC1	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.

4.33 Least Mean Square (LMS)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	LMS Xmem, Ymem, ACx, ACy	No	4	1	X

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs two paralleled operations in one cycle: multiply and accumulate (MAC), and addition. The instruction is executed:

```
ACy = ACy + (Xmem * Ymem)
:: ACx = rnd(ACx + (Xmem << #16))
```

The first 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 Xmem, sign extended to 17 bits, and the content of data memory operand Ymem, 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 is detected, the accumulator is saturated according to SATD.

The second operation performs an addition between an accumulator content and the content of data memory operand Xmem shifted left by 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 identical 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.
- ❏ Rounding is performed according to RDM.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1:

- ❏ the rounding is performed without clearing the 16 lowest bits of ACx
- ❏ the addition operation has no overflow detection, report, and saturation after the shifting operation

Status Bits

Affected by FRCT, SMUL, C54CM, M40, RDM, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
LMS *AR0, *AR1, AC0, AC1	The content addressed by AR0 multiplied by the content addressed by AR1 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 of AC0. The result is rounded 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
CARRY	0	CARRY	0
FRCT	0	FRCT	0

4.34 Linear/Circular Addressing Qualifiers

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<instruction>.LR	No	1	1	AD
[2]	<instruction>.CR	No	1	1	AD

Brief Description

These instructions are instruction qualifiers that can be paralleled with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. These instructions cannot be executed in parallel with other types of instructions and they cannot be executed alone.

Status BIts

Affected by none

Affects none

4.34.1 Linear Addressing: <instruction>.LR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	<instruction>.LR	No	1	1	AD

Operands none

Description

This instruction is an instruction qualifier that can be paralleled with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. This instruction cannot be executed in parallel with other types of instructions and it cannot be executed alone.

When this instruction is used in parallel, all modifications of ARx and CDP pointer registers used in the indirect addressing mode are done linearly (as if ST2_55 register bits 0 to 8 were cleared to 0).

Status BIts

Affected by none

Affects none

Repeat

This instruction can be repeated.

4.34.2 Circular Addressing: <instruction>.CR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	<instruction>.CR	No	1	1	AD

Operands none

Description

This instruction is an instruction qualifier that can be paralleled with any instruction making an indirect Smem, Xmem, Ymem, Lmem, Baddr, or Cmem addressing. This instruction cannot be executed in parallel with other types of instructions and it cannot be executed alone.

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.

4.35 Load Effective Address to Extended Auxiliary Register

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	AMAR Smem, XAdst	No	3	1	AD
[2]	AMOV k23, XAdst	No	6	1	AD

Operands k23, Smem, XAdst

Description

If a memory (Smem) location is specified, 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.

If a 23-bit unsigned constant (k23) is specified, this instruction loads the constant into the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).

The premodification or postmodification of the auxiliary register (ARx), the use of port(#K), or the use of the port(Smem) qualifier is not supported for this instruction. The use of auxiliary register offset operations is supported. If the corresponding bit (ARnLC) in status register ST2_55 is set to 1, the circular buffer management also controls the result stored in XAdst.

Status BIts

Affected by ST2_55

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
AMAR *AR1+, XAR0	The content of AR1 is incremented by 1 and loaded into XAR0.
AMOV #7FFFFFh, XAR0	Move the 23-bit value (7FFFFFh) into XAR0.

4.36 Load Extended Auxiliary Register from Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV dbl(Lmem), XAdst	No	3	1	X

Operands Lmem, XAdst

Description

This instruction loads the lower 23 bits of the data addressed by data memory operand (Lmem) to the 23-bit destination register (XARx, XSP, XSSP, XDP, or XCDP).

Status BIts

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV dbl(*AR3), XAR1	The 7 lowest bits of the content at the location addressed by AR3 and the 16 bits of the content at the location addressed by AR3 + 1 are loaded into XAR1.

Before		After	
XAR1	00 0000	XAR1	12 0FD3
AR3	0200	AR3	0200
200	3492	200	3492
201	0FD3	201	0FD3

4.37 Logical Shift (SFTL)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SFTL dst, #1	Yes	2	1	X
[2]	SFTL dst, #–1	Yes	2	1	X
[3]	SFTL ACx, Tx[, ACy]	Yes	2	1	X
[4]	SFTL ACx, #SHIFTW[, ACy]	Yes	3	1	X

Brief Description

These instructions perform an unsigned shift by an immediate value, 1 or SHIFTW, or the content of a temporary register (Tx):

- ❑ In the D-unit shifter, if the destination operand is an accumulator (ACx).
- ❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).

Status BIts

Affected by C54CM, M40

Affects CARRY

4.37.1 Logical Shift Left: SFTL dst, #1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SFTL dst, #1	Yes	2	1	X

Operands

dst

Description

This instruction shifts left by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit shifter.
  - 0 is inserted at bit position 0.
  - The shifted-out bit is extracted at a bit position according to M40.
- When 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.
  - The shifted-out bit is extracted at bit position 15 and stored in the CARRY status bit.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by M40

Affects CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTL AC2, #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	
AC1	8F E340 5678	AC1	00 C680 ACF0
CARRY	0	CARRY	1
M40	0	M40	0

4.37.2 Logical Shift Right: SFTL dst, #–1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	SFTL dst, #–1	Yes	2	1	X

Operands dst

Description

This instruction shifts right by 1 bit the input operand (dst). The CARRY status bit contains the shifted-out bit.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit shifter.
  - 0 is inserted at a bit position according to M40.
  - The shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit.
- When 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 15.
  - The shifted-out bit is extracted at bit position 0 and stored in the CARRY status bit.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by M40

Affects CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTL AC0, #–1	The content of AC0 is logically shifted right by 1 bit and the result is stored in AC0.

4.37.3 Logical Shift: SFTL ACx, Tx[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	SFTL ACx, Tx[, ACy]	Yes	2	1	X

Operands ACx, ACy, Tx

Description

This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. If the 16-bit value contained in Tx is out of the –32 to +31 range, the shift is saturated to –32 or +31 and the shift operation is performed with this value. However, no overflow is reported when such saturation occurs.

- ❑ 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 repeated.

Example

Syntax	Description
SFTL AC0, T0, AC1	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 B000 1234       AC0          5F B000 1234
AC1          00 C680 ACF0       AC1          00 02C0 0048
T0                   FFFA       T0                   FFFA
M40                     0       M40                     0
```

4.37.4 Logical Shift: SFTL ACx, #SHIFTW[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	SFTL ACx, #SHIFTW[, ACy]	Yes	3	1	X

Operands

ACx, ACy, SHIFTW

Description

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 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, SHIFTW = 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.

Status Bits

Affected by M40

Affects CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTL AC1, #31, AC0	The content of AC1 is logically shifted left by 31 bits and the result is stored in AC0.

4.38 Maximum Comparison (MAX)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MAX [src,] dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs a maximum 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.

- When the destination operand (dst) is an accumulator:
  - 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:

    If M40 = 0, src(31–0) content is compared to dst(31–0) 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(31–0) > dst(31–0))
step2: { CARRY = 0; dst(39–0) = src(39–0) }
   else
step3: CARRY = 1
```

    If M40 = 1, src(39–0) content is compared to dst(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) > dst(39–0))
step2: { CARRY = 0; dst(39–0) = src(39–0) }
   else
step3: CARRY = 1
```

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

❑ 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.

Examples

Syntax	Description
MAX AC2, AC1	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

Syntax	Description
MAX AR1, AC1	The content of AR1 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	
AR1	8020	AR1	8020
AC1	00 0000 0040	AC1	00 0000 0040
CARRY	0	CARRY	1

Syntax	Description
MAX AC1, T1	The content of AC1(15–0) is greater than the content of T1, the content of AC1(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
T1	8010	T1	8020
CARRY	0	CARRY	0

4.39 Minimum Comparison (MIN)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MIN [src,] dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs a minimum 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.

- When the destination operand (dst) is an accumulator:
  - 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:

    If M40 = 0, src(31–0) content is compared to dst(31–0) 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(31–0) < dst(31–0))
step2: { CARRY = 0; dst(39–0) = src(39–0) }
   else
step3: CARRY = 1
```

    If M40 = 1, src(39–0) content is compared to dst(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) < dst(39–0))
step2: { CARRY = 0; dst(39–0) = src(39–0) }
   else
step3: CARRY = 1
```

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

Example

Syntax	Description
MIN AC1, T1	The content of AC1(15–0) is greater than the content of T1, the content of T1 remains the same and 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

4.40 Memory-Mapped Register Access Qualifier (mmap)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	mmap	No	1	1	D

Operands none

Description

This is an operand qualifier that can be paralleled with any instruction making a Smem or Lmem direct memory access (dma). This operand qualifier allows you to locally prevent the dma access from being relative to the data stack pointer (SP) or the local data page register (DP). It forces the dma access to be relative to the memory-mapped register (MMR) data page start address, 00 0000h.

This operand qualifier cannot be executed:

- ❏ alone
- ❏ in parallel with instructions not embedding an Smem or Lmem data memory operand
- ❏ in parallel with instructions loading or storing a byte to a register (see Accumulator, Auxiliary, or Temporary Register Load instructions [5], [6], [10], and [11]; Accumulator, Auxiliary, or Temporary Register Store instructions [2] and [3])

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 of each main data pages of 64K words cannot be accessed through this mechanism.

Any instruction using the mmap modifier cannot be combined with any other user-defined parallelism instruction. The following instruction is not valid:

```
MOV AR1, mmap(@BSAC)
|| BSET CDPLC
```

The following instruction is valid:

```
MOV AR1, @BSAC
|| mmap
```

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV AC0, T2 || mmap	The content of AC0(15–0) is copied into T2.

4.41 Memory Bit Test/Set/Clear/Complement

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BTST src, Smem, TCx	No	3	1	X
[2]	BNOT src, Smem	No	3	1	X
[3]	BCLR src, Smem	No	3	1	X
[4]	BSET src, Smem	No	3	1	X
[5]	BTSTSET k4, Smem, TCx	No	3	1	X
[6]	BTSTCLR k4, Smem, TCx	No	3	1	X
[7]	BTSTNOT k4, Smem, TCx	No	3	1	X
[8]	BTST k4, Smem, TCx	No	3	1	X

Brief Description

These instructions perform a bit manipulation in the A-unit ALU. These instructions manipulate a single bit of a memory (Smem) location. The bit manipulated is defined by either the content of the source (src) operand or a 4-bit immediate value, k4. The following operations can be performed on the selected bit:

- ❏ copy the bit into TCx
- ❏ complement the bit in Smem
- ❏ clear the bit in Smem
- ❏ set the bit in Smem
- ❏ copy the bit into TCx and set the bit in Smem
- ❏ copy the bit into TCx and clear the bit in Smem
- ❏ copy the bit into TCx and complement the bit in Smem

Status Bits

Affected by none

Affects TCx

4.41.1 Memory Bit Test: BTST src, Smem, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BTST src, Smem, TCx	No	3	1	X

Operands Smem, src, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location. The tested bit is copied into the selected TCx status bit.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax	Description
BTST AC0, *AR0, TC1	The bit at the position defined by AC0(3–0) in the content addressed by AR0 is tested and the tested bit is copied into TC1.

Before		After	
AC0	00 0000 0008	AC0	00 0000 0008
*AR0	00C0	*AR0	00C0
TC1	0	TC1	0

Syntax	Description
BTST AC0, *AR0, TC2	The bit at the position defined by AC0(3–0) in the content addressed by AR0 is tested and the tested bit is copied into TC2.

Before		After	
AC0	00 0000 0007	AC0	00 0000 0007
*AR0	00C0	*AR0	00C0
TC2	0	TC2	1

4.41.2 Memory Bit Complement: BNOT src, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	BNOT src, Smem	No	3	1	X

Operands

Smem, src

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction complements a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.41.3 Memory Bit Clear: BCLR src, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	BCLR src, Smem	No	3	1	X

Operands Smem, src

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction clears to 0 a single bit, as defined by the content of the source (src) operand, of a memory (Smem) location.

The generated bit address must be within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.41.4 Memory Bit Set: BSET src, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	BSET src, Smem	No	3	1	X

Operands 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 within 0–15 (only the 4 LSBs of the register are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
BSET AC0, *AR3	The bit at the position defined by AC0(3–0) in the content addressed by AR3 is set to 1.

4.41.5 Memory Bit Test/Set: BTSTSET k4, Smem, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	BTSTSET k4, Smem, TCx	No	3	1	X

Operands k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx and is set to 1 in Smem.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax	Description
BTSTSET #12, *AR3, TC1	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.
BTSTSET #12, *AR3, TC2	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 TC2. The selected bit (12) in the content addressed by AR3 is set to 1.

4.41.6 Memory Bit Test/Clear: BTSTCLR k4, Smem, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	BTSTCLR k4, Smem, TCx	No	3	1	X

Operands k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx and is cleared to 0 in Smem.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax	Description
BTSTCLR #12, *AR3, TC1	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.
BTSTCLR #12, *AR3, TC2	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 TC2. The selected bit (12) in the content addressed by AR3 is cleared to 0.

4.41.7 Memory Bit Test/Complement: BTSTNOT k4, Smem, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	BTSTNOT k4, Smem, TCx	No	3	1	X

Operands k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location and the tested bit is copied into status bit TCx and is complemented in Smem.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax	Description
BTSTNOT #12, *AR0, TC1	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	
*AR0	0040	*AR0	1040
TC1	0	TC1	0

Syntax	Description
BTSTNOT #12, *AR3, TC2	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 TC2. The selected bit (12) in the content addressed by AR3 is complemented.

Before		After	
*AR3	0040	*AR3	1040
TC2	0	TC2	0

4.41.8 Memory Bit Test: BTST k4, Smem, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	BTST k4, Smem, TCx	No	3	1	X

Operands

k4, Smem, TCx

Description

This instruction performs a bit manipulation in the A-unit ALU. The instruction tests a single bit, as defined by a 4-bit immediate value, k4, of a memory (Smem) location. The tested bit is copied into status bit TCx.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax	Description
BTST #12, *AR3, TC1	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.
BTST #12, *AR3, TC2	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 TC2.

4.42 Memory Comparison (CMP)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	CMP Smem == K16, TC1x	No	4	1	X

Operands K16, Smem, TCx

Description

This instruction performs a comparison in the A-unit ALU. The data memory operand Smem is compared to the 16-bit signed constant, K16. If they are equal, the TCx status bit is set to 1; otherwise, it is cleared to 0.

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Examples

Syntax
CMP *AR1+ == #400h, TC1

Description
The content addressed by AR1 is compared to the signed 16-bit value (400h). Because they are equal, TC1 is set to 1. AR1 is incremented by 1.

Before		After	
AR1	0285	AR1	0286
0285	0400	0285	0400
TC1	0	TC1	1

Syntax
CMP *AR1 == #400h, TC2

Description
The content addressed by AR1 is compared to the signed 16-bit value (400h). Because they are not equal, TC2 is cleared to 0.

Before		After	
AR1	0285	AR1	0285
0285	0000	0285	0000
TC2	0	TC2	0

4.43 Memory Delay (DELAY)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	DELAY Smem	No	2	1	X

Operands

Smem

Description

This instruction copies the content of the memory (Smem) location into the next higher address. 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 soft dual memory addressing mode mechanism can not be applied to this instruction.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
DELAY *AR1+	The content addressed by AR1 is copied to the next higher address, AR1 + 1. AR1 is incremented by 1.

Before		After	
AR1	0200	AR1	0201
200	3400	200	3400
201	0D80	201	3400
202	2030	202	2030

4.44 Memory-to-Memory Move/Memory Initialization (MOV)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV Cmem, Smem	No	3	1	X
[2]	MOV Smem, Cmem	No	3	1	X
[3]	MOV K8, Smem	No	3	1	X
[4]	MOV K16, Smem	No	4	1	X
[5]	MOV Cmem, dbl(Lmem)	No	3	1	X
[6]	MOV dbl(Lmem), Cmem	No	3	1	X
[7]	MOV dbl(Xmem), dbl(Ymem)	No	3	1	X
[8]	MOV Xmem, Ymem	No	3	1	X

Brief Description

These instructions initialize data memory locations. They use a dedicated datapath to perform the operation.

Status Bits

Affected by none

Affects none

4.44.1 Memory-to-Memory Move: MOV Cmem, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV Cmem, Smem	No	3	1	X

Operands

Cmem, Smem

Description

This instruction stores the content of a data memory operand Cmem, addressed using the coefficient addressing mode, to a memory (Smem) location.

For this instruction, the Cmem operand is not accessed through the BB bus (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *CDP, *(#0500h)	The content addressed by the coefficient data pointer register (CDP) is copied to address 0500h.

Before		After	
*CDP	3400	*CDP	3400
500	0000	500	3400

4.44.2 Memory-to-Memory Move: MOV Smem, Cmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MOV Smem, Cmem	No	3	1	X

Operands Cmem, Smem

Description

This instruction stores the content of a memory (Smem) location to a data memory location (Cmem) addressed using the coefficient addressing mode.

For this instruction, the Cmem operand is not accessed through the BB bus (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *AR3, *CDP	The content addressed by AR3 is copied in the location addressed by the coefficient data pointer register (CDP).

4.44.3 Memory Initialization: MOV Kx, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MOV K8, Smem	No	3	1	X
[4]	MOV K16, Smem	No	4	1	X

Operands

Kx, Smem

Description

This instruction stores an 8-bit signed constant, K8, or a 16-bit signed constant, K16, to a memory (Smem) location. For instruction [3], the immediate value is always signed extended to 16 bits before being stored in memory.

Status Bits

Affected by none

Affects none

Repeat

Instruction [3] can be repeated. Instruction [4] cannot be repeated.

Example

Syntax	Description
MOV #248, *(#0501h)	The signed 16-bit value (248) is loaded to address 501h.

```
Before                  After
0501          FC00      0501          F800
```

4.44.4 Memory-to-Memory Move: MOV Cmem, dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MOV Cmem,dbl(Lmem)	No	3	1	X

Operands Cmem, Lmem

Description

This instruction stores the content of two consecutive data memory (Cmem) locations, addressed using the coefficient addressing mode, to two consecutive data memory (Lmem) locations.

For this instruction, the Cmem operand is not accessed through the BB bus (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *(CDP + T0), dbl(*AR1)	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	
T0	0005	T0	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

4.44.5 Memory-to-Memory Move: MOV dbl(Lmem) Cmem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MOV dbl(Lmem), Cmem	No	3	1	X

Operands Cmem, Lmem

Description

This instruction stores the content of two consecutive data memory (Lmem) locations to two consecutive data memory (Cmem) locations addressed using the coefficient addressing mode.

For this instruction, the Cmem operand is not accessed through the BB bus (contrary to instructions like Dual Multiply and Accumulate using the Cmem operand). On all C55x-based devices, the Cmem operand may be mapped in external or internal memory space.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV dbl(*AR3+), *CDP	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.

4.44.6 Memory-to-Memory Move: MOV dbl(Xmem), dbl(Ymem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MOV dbl(Xmem), dbl(Ymem)	No	3	1	X

Operands

Xmem, Ymem

Description

This instruction stores the content of two consecutive data memory (Xmem) locations, addressed using the dual addressing mode, to two consecutive data memory (Ymem) locations.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV dbl(*AR0), dbl(*AR1)	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

4.44.7 Memory-to-Memory Move: MOV Xmem, Ymem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	MOV Xmem, Ymem	No	3	1	X

Operands Xmem, Ymem

Description

This instruction stores the content of data memory (Xmem) location, addressed using the dual addressing mode, to data memory (Ymem) location.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV *AR5, *AR3	The content addressed by AR5 is copied in the location addressed by AR3.

4.45 Modify Auxiliary Register (MAR)

No.	Syntax†	Parallel Enable Bit	Size	Cycles	Pipeline
[1a]	AADD TAx, TAy	Yes	3	1	AD
[1b]	AADD TAx, TAy	Yes	3	1	AD
[2a]	ASUB TAx, TAy	Yes	3	1	AD
[2b]	ASUB TAx, TAy	Yes	3	1	AD
[3a]	AMOV TAx, TAy	Yes	3	1	AD
[3b]	AMOV TAx, TAy	Yes	3	1	AD
[4a]	AADD k8, TAx	Yes	3	1	AD
[4b]	AADD k8, TAx	Yes	3	1	AD
[5a]	ASUB k8, TAX	Yes	3	1	AD
[5b]	ASUB k8, TAx	Yes	3	1	AD
[6a]	AMOV k8, TAx	Yes	3	1	AD
[6b]	AMOV k8, TAx	Yes	3	1	AD
[7]	AMOV D16, TAx	No	4	1	AD
[8]	AMAR Smem	No	2	1	AD

† For instructions [1]–[6], the assembler selects the correct instruction opcode depending on the instruction position in a paralleled pair.

Brief Description

These instructions perform, in the A-unit address generation units:

- ❑ an addition between two auxiliary or temporary registers, TAx and TAy, and stores the result in TAy
- ❑ a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy
- ❑ a move from auxiliary or temporary register TAx to auxiliary or temporary register TAy
- ❑ an addition between the auxiliary or temporary registers TAx and the unsigned constant k8, and stores the result in TAx
- ❑ a subtraction between the auxiliary or temporary registers TAx and the unsigned constant k8, and stores the result in TAx
- ❑ a load in the auxiliary or temporary registers TAx of the unsigned constant k8
- ❑ a load in the auxiliary or temporary registers TAx of the absolute data address signed constant D16
- ❑ an 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.

For instructions [1], [2], [4], [5], and [8], 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

Affects none

4.45.1 Modify Auxiliary Register: AADD TAx, TAy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	AADD TAx, TAy	Yes	3	1	AD

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 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.

Examples

Syntax	Description
AADD AR1, AR0	The content of AR0 is added to the content of AR1 and the result is stored in AR0.
AADD T1, T0	The content of T0 is added to the content of T1 and the result is stored in T0.

4.45.2 Modify Auxiliary Register: ASUB TAx, TAy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	ASUB TAx, TAy	Yes	3	1	AD

Operands

TAx, TAy

Description

This instruction performs, in the A-unit address generation units, a subtraction between two auxiliary or temporary registers, TAy and TAx, and stores the result in TAy. 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.

Examples

Syntax	Description
ASUB AR1, AR0	The content of AR1 is subtracted from the content of AR0 and the result is stored in AR0.
ASUB T1, T0	The content of T1 is subtracted from the content of T0 and the result is stored in T0.

4.45.3 Modify Auxiliary Register: AMOV TAx, TAy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	AMOV TAx, TAy	Yes	3	1	AD

Operands

TAx, TAy

Description

This instruction performs, in the A-unit address generation units, a move from the auxiliary or temporary registers TAx to the auxiliary or temporary registers TAy. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

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 none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
AMOV AR1, AR0	The content of AR1 is copied to AR0.
AMOV T1, T0	The content of T1 is copied to T0.

4.45.4 Modify Auxiliary Register: AADD k8, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	AADD k8, TAx	Yes	3	1	AD

Operands

TAx, k8

Description

This instruction performs, in the A-unit address generation units, an addition between the auxiliary or temporary registers TAx and the unsigned constant k8, 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.

Examples

Syntax	Description
AADD #255, T0	The unsigned 8-bit value (255) is added to the content of T0 and the result is stored in T0.
AADD #255, AR0	The unsigned 8-bit value (255) is added to the content of AR0 and the result is stored in AR0.

4.45.5 Modify Auxiliary Register: ASUB k8, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	ASUB k8, TAx	Yes	3	1	AD

Operands

TAx, k8

Description

This instruction performs, in the A-unit address generation units, a subtraction between the auxiliary or temporary registers TAx and the unsigned constant k8, 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.

Examples

Syntax	Description
ASUB #255, AR0	The unsigned 8-bit value (255) is subtracted from the content of AR0 and the result is stored in AR0.
ASUB #255, T0	The unsigned 8-bit value (255) is subtracted from the content of T0 and the result is stored in T0.

4.45.6 Modify Auxiliary Register: AMOV k8, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	AMOV k8, TAx	Yes	3	1	AD

Operands

TAx, k8

Description

This instruction performs, in the A-unit address generation units, a load in the auxiliary or temporary registers TAx of the unsigned constant k8. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

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 none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
AMOV #255, AR0	The unsigned 8-bit value (255) is copied to AR0.
AMOV #255, T0	The unsigned 8-bit value (255) is copied to T0.

4.45.7 Modify Auxiliary Register: AMOV D16, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	AMOV D16, TAx	No	4	1	AD

Operands TAx, D16

Description

This instruction performs, in the A-unit address generation units, a load in the auxiliary or temporary registers TAx of the absolute data address signed constant D16. The operation is performed in the address phase of the pipeline; however, data memory is not accessed.

Compatibility with C54x devices (C54CM = 1)

In the translated code section, the MAR() instruction must be executed with C54CM set to 1.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AMOV #FFFFh, T1	The address FFFFh is copied to T1.

4.45.8 Modify Auxiliary Register: AMAR Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	AMAR Smem	No	2	1	AD

Operands Smem

Description

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 translated code section, the MAR() instruction must be executed with C54CM set to 1.

Status Bits

Affected by ST2_55

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AMAR *AR3+	The content of AR3 is incremented by 1.

4.46 Modify Data Stack Pointer (AADD)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	AADD K8, SP	Yes	2	1	AD

Operands K8

Description

This instruction performs an addition in the A-unit ALU in the address phase of the pipeline. The 8-bit constant K8 is sign extended to 16 bits and added to the data stack pointer (SP). When in 32-bit stack configuration, the system stack pointer (SSP) is also modified. Updates of the SP and SSP (depending on the stack configuration) should not be executed in parallel with this instruction.

The latencies versus any address generation through SP is 3 cycles.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
AADD #127, SP	The 8-bit value (127) is sign extended to 16 bits and added to the stack pointer (SP).

4.47 Multiply (MPY)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQR[R] [ACx,] ACy	Yes	2	1	X
[2]	MPY[R] [ACx,] ACy	Yes	2	1	X
[3]	MPY[R] Tx, [ACx,] ACy	Yes	2	1	X
[4]	MPYK[R] K8, [ACx,] ACy	Yes	3	1	X
[5]	MPYK[R] K16, [ACx,] ACy	No	4	1	X
[6]	MPYM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X
[7]	SQRM[R] [T3 =]Smem, ACx	No	3	1	X
[8]	MPYM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X
[9]	MPYMK[R] [T3 =]Smem, K8, ACx	No	4	1	X
[10]	MPYM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx	No	4	1	X
[11]	MPYM[R][U] [T3 =]Smem, Tx, ACx	No	3	1	X

Brief 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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.47.1 Multiply: SQR[R] [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQR[R] [ACx,] ACy	Yes	2	1	X

Operands

ACx, ACy

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier 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.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQR AC1, AC0	The content of AC1 is squared and the result is stored in AC0.

4.47.2 Multiply: MPY[R] [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MPY[R] [ACx,] ACy	Yes	2	1	X

Operands

ACx, ACy

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and ACy(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.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPY AC1, AC0	The content of AC1 is multiplied by the content of AC0 and the result is stored in AC1.

```
Before                          After
AC0         02 6000 3400        AC0         02 6000 3400
AC1         00 C000 0000        AC1         00 4800 0000
M40                    1        M40                    1
FRCT                   0        FRCT                   0
ACOV1                  0        ACOV1                  0
```

4.47.3 Multiply: MPY[R] Tx, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MPY[R] Tx, [ACx,] ACy	Yes	2	1	X

Operands ACx, ACy, Tx

Description

This instruction performs a multiplication 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 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 R keyword is applied to the instruction.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPY T0, AC1, AC0	The content of AC1 is multiplied by the content of T0 and the result is stored in AC0.

4.47.4 Multiply: MPYK[R] K8, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	MPYK[R] K8, [ACx,] ACy	Yes	3	1	X

Operands

ACx, ACy, K8

Description

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 R keyword is applied to the instruction.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, RDM

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYK #–2, AC1, AC0	The content of AC1 is multiplied by a signed 8-bit value (–2) and the result is stored in AC0.

4.47.5 Multiply: MPYK[R] K16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MPYK[R] K16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, K16

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are ACx(32–16) and the 16-bit signed constant, K16, 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.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYK #–64, AC1, AC0	The content of AC1 is multiplied by a signed 16-bit value (–64) and the result is stored in AC0.

4.47.6 Multiply: MPYM[R] [T3 =]Smem, Cmem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MPYM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X

Operands ACx, Cmem, Smem

Description

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 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 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 R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand Smem in temporary register T3.

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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYM *AR3, *CDP, AC0	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.

4.47.7 Multiply: SQRM[R] [T3 =]Smem, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	SQRM[R] [T3 =]Smem, ACx	No	3	1	X

Operands

ACx, Smem

Description

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.
- 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 R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQRM *AR3, AC0	The content addressed by AR3 is squared and the result is stored in AC0.

4.47.8 Multiply: MPYM[R] [T3 =]Smem, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	MPYM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction performs a multiplication 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.
- 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 R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYM *AR3, AC1, AC0	The content addressed by AR3 is multiplied by the content of AC1 and the result is stored in AC0.

4.47.9 Multiply: MPYMK[R] [T3 =]Smem, K8, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	MPYMK[R] [T3 =]Smem, K8, ACx	No	4	1	X

Operands

ACx, K8, Smem

Description

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 R 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 instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, RDM

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
MPYMK *AR3, #–2, AC0	The content addressed by AR3 is multiplied a signed 8-bit value (–2) and the result is stored in AC0.

4.47.10 Multiply: MPYM[R][40] [T3 =][uns()Xmem()], [uns()Ymem()], ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	MPYM[R][40] [T3 =][uns()Xmem()], [uns()Ymem()], ACx	No	4	1	X

Operands ACx, Xmem, Ymem

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- 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 R keyword is applied to the instruction.
- 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 40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYM uns(*AR3), uns(*AR4), AC0	The unsigned content addressed by AR3 is multiplied by the unsigned content addressed by AR4 and the result is stored in AC0.

4.47.11 Multiply: MPYM[R][U] [T3 =]Smem, Tx, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	MPYM[R][U] [T3 =]Smem, Tx, ACx	No	3	1	X

Operands ACx, Tx, Smem

Description

This instruction performs a multiplication in the D-unit MAC. The input operands of the multiplier are the content of Tx and the content of a memory (Smem) location.

- The input operands are zero extended to 17 bits, if the optional U keyword is applied to the instruction.
- 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.
- The 32-bit result of the multiplication is zero extended to 40 bits, if the optional U keyword is applied to the instruction.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MPYMU *AR3, T0, AC0	The unsigned content addressed by AR3 is multiplied by the unsigned content of T0 and the result is stored in AC0.

4.48 Multiply and Accumulate (MAC)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQA[R] [ACx,] ACy	Yes	2	1	X
[2]	MAC[R] ACx, Tx, ACy[, ACy]	Yes	2	1	X
[3]	MAC[R] ACy, Tx, ACx, ACy	Yes	2	1	X
[4]	MACK[R] Tx, K8, [ACx,] ACy	Yes	3	1	X
[5]	MACK[R] Tx, K16, [ACx,] ACy	No	4	1	X
[6]	MACM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X
[7]	MACM[R]Z [T3 =]Smem, Cmem, ACx	No	3	1	X
[8]	SQAM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X
[9]	MACM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X
[10]	MACM[R] [T3 =]Smem, Tx, [ACx,] ACy	No	3	1	X
[11]	MACMK[R] [T3 =]Smem, K8, [ACx,] ACy	No	4	1	X
[12]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	No	4	1	X
[13]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16 [, ACy]	No	4	1	X

Brief Description

This instruction performs a multiplication and an accumulation 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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.48.1 Multiply and Accumulate (MAC): SQA[R] [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQA[R] [ACx,] ACy	Yes	2	1	X

Operands

ACx, ACy

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition 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.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQA AC1, AC0	The content of AC1 squared is added to the content of AC0 and the result is stored in AC0.

4.48.2 Multiply and Accumulate (MAC): MAC[R] ACx, Tx, ACy[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MAC[R] ACx, Tx, ACy[, ACy]	Yes	2	1	X

Operands

ACx, ACy, Tx

Description

This instruction performs a multiplication and an accumulation 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 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition 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.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MAC AC1, T0, AC0	The content of AC1 multiplied by the content of T0 is added to the content of AC0 and the result is stored in AC0.

4.48.3 Multiply and Accumulate (MAC): MAC[R] ACy, Tx, ACx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MAC[R] ACy, Tx, ACx, ACy	Yes	2	1	X

Operands

ACx, ACy, Tx

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are ACy(32–16) and the content of Tx, 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 ACx.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition 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.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACR AC1, T1, AC0, AC1	The content of AC1 multiplied by the content of T1 is added to the content of AC0. The result is rounded and stored in AC1.

4.48.4 Multiply and Accumulate (MAC): MACK[R] Tx, K8, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	MACK[R] Tx, K8, [ACx,] ACy	Yes	3	1	X

Operands

ACx, ACy, Tx, K8

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, 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 and added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition 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.

Status Bits

Affected by FRCT, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACK T0, #FFh, AC1, AC0	The content of T0 multiplied by a signed 8-bit value (FFh) is added to the content of AC1 and the result is stored in AC0.

4.48.5 Multiply and Accumulate (MAC): MACK[R] Tx, K16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MACK[R] Tx, K16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, Tx, K16

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of Tx, sign extended to 17 bits, and the 16-bit signed constant, K16, 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 ACx.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition 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.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACK T0, #FFFFh, AC1, AC0	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.

4.48.6 Multiply and Accumulate (MAC): MACM[R] [T3 =]Smem, Cmem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MACM[R] [T3 =]Smem, Cmem, ACx	No	3	1	X

Operands ACx, Cmem, Smem

Description

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 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 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 operand Smem in temporary register T3.

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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACMR *AR1, *CDP, AC2	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                          After
AC2          00 EC00 0000       AC2          00 EC00 0000
AR1                  0302       AR2                  0302
CDP                  0202       CDP                  0202
302                  FE00       302                  FE00
202                  0040       202                  0040
ACOV2                   0       ACOV2                   1
```

4.48.7 Parallel Multiply and Accumulate (MAC) and Delay: MACM[R]Z [T3 =]Smem, Cmem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MACM[R]Z [T3 =]Smem, Cmem, ACx	No	3	1	X

Operands ACx, Cmem, Smem

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC in parallel with the delay memory instruction. The input operands of the multiplier are the content of a memory (Smem) location, sign extended to 17 bits, 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 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 operand Smem in temporary register T3.

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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 set to 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACMZ *AR3, *CDP, AC0	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.

4.48.8 Multiply and Accumulate (MAC): SQAM[R] [T3 =]Smem, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	SQAM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

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 = 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQAM *AR3, AC1, AC0	The content addressed by AR3 squared is added to the content of AC1 and the result is stored in AC0.

4.48.9 Multiply and Accumulate (MAC): MACM[R] [T3 =]Smem, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	MACM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction performs a multiplication and an accumulation 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.
- 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACM *AR3, AC0, AC1	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.

4.48.10 Multiply and Accumulate (MAC): MACM[R] [T3 =]Smem, Tx, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	MACM[R] [T3 =]Smem, Tx, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Tx, Smem

Description

This instruction performs a multiplication and an accumulation 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 added to the source accumulator ACx.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACM *AR3, T0, AC1, AC0	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.

4.48.11 Multiply and Accumulate (MAC): MACMK[R] [T3 =]Smem, K8, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	MACMK[R] [T3 =]Smem, K8, [ACx,] ACy	No	4	1	X

Operands ACx, ACy, K8, Smem

Description

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 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit 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 operand 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

Affects ACOVy

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
MACMK *AR3, #FFh, AC1, AC0	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.

4.48.12 Multiply and Accumulate (MAC): MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	No	4	1	X

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- 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 R keyword is applied to the instruction.
- Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- When an addition 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 40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACMR uns(*AR2+), uns(*AR3+), AC3	The unsigned content addressed by AR2 multiplied by the unsigned content addressed by AR3 is added to the content of AC3. The result is rounded and stored in AC3. The result generated an overflow. AR2 and AR3 are both incremented by 1.

Before		After	
AC3	00 2300 EC00	AC3	00 9221 0000
AR2	302	AR2	303
AR3	202	AR3	203
ACOV3	0	ACOV3	1
302	FE00	302	FE00
202	7000	202	7000
M40	0	M40	0
SATD	0	SATD	0
FRCT	0	FRCT	0

4.48.13 Multiply and Accumulate (MAC): MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16[, ACy]	No	4	1	X

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier are the content of data memory operand Xmem, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❑ 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 shifted right by 16 bits. The shifting operation is performed with a sign extension of source accumulator ACx(39).
- ❑ Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- ❑ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❑ When an addition 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 40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MACM uns(*AR3), uns(*AR4), AC1 >> #16, AC0	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.

4.49 Multiply and Subtract (MAS)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQS[R] [ACx,] ACy	Yes	2	1	X
[2]	MAS[R] Tx, [ACx,] ACy	Yes	2	1	X
[3]	MASM[R] [T3 =]Smem, Cmemf, ACx	No	3	1	X
[4]	SQSM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X
[5]	MASM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X
[6]	MASM[R] [T3 =]Smem, Tx, [ACx,] ACy	No	3	1	X
[7]	MASM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	No	4	1	X

Brief 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
- 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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx, ACOVy

4.49.1 Multiply and Subtract (MAS): SQS[R] [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQS[R] [ACx,] ACy	Yes	2	1	X

Operands

ACx, ACy

Description

This instruction performs a multiplication and a subtraction in the D-unit MAC. The input operands of the multiplier 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 subtracted from the source accumulator ACy.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQS AC0, AC1	The content of AC0 squared is subtracted from the content of AC1 and the result is stored in AC1.

4.49.2 Multiply and Subtract (MAS): MAS[R] Tx, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	MAS[R] Tx, [ACx,] ACy	Yes	2	1	X

Operands ACx, ACy, Tx

Description

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 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.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax

MASR T1, AC0, AC1

Description

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

AC0	00 EC00 0000
AC1	00 3400 0000
T1	2000
M40	0
ACOV1	0
FRCT	0

After

AC0	00 EC00 0000
AC1	00 1680 0000
T1	2000
M40	0
ACOV1	0
FRCT	0

4.49.3 Multiply and Subtract (MAS): MASM[R] [T3 =]Smem, Cmemf, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	MASM[R] [T3 =]Smem, Cmemf, ACx	No	3	1	X

Operands ACx, Cmem, Smem

Description

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, 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 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 R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand Smem in temporary register T3.

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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured.

Status Bits

Affected by FRCT, SMUL, M40, RDM, SATD

Affects ACOVx

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASMR *AR1, *CDP, AC2	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		After	
AC2	00 EC00 0000	AC2	00 EC01 0000
AR1	0302	AR2	0302
CDP	0202	CDP	0202
302	FE00	302	FE00
202	0040	202	0040
ACOV2	0	ACOV2	1
SATD	0	SATD	0
RDM	0	RDM	0
FRCT	0	FRCT	0

4.49.4 Multiply and Subtract (MAS): SQSM[R] [T3 =]Smem, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	SQSM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

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 R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQSM *AR3, AC1, AC0	The content addressed by AR3 squared is subtracted from the content of AC1 and the result is stored in AC0.

4.49.5 Multiply and Subtract (MAS): MASM[R] [T3 =]Smem, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MASM[R] [T3 =]Smem, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

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.
- 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.
- Rounding is performed according to RDM, if the optional R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASM *AR3, AC1, AC0	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.

4.49.6 Multiply and Subtract (MAS): MASM[R] [T3 =]Smem, Tx, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	MASM[R] [T3 =]Smem, Tx, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Tx, Smem

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 R keyword is applied to the instruction.
- 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 store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASM *AR3, T0, AC1, AC0	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.

4.49.7 Multiply and Subtract (MAS): MASM[R][40] [T3 =][uns()]Xmem[]), [uns()]Ymem[])], [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MASM[R][40] [T3 =][uns()]Xmem[]), [uns()]Ymem[])], [ACx,] ACy	No	4	1	X

Operands ACx, ACy, Xmem, Ymem

Description

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, sign extended to 17 bits, and the content of data memory operand Ymem, sign extended to 17 bits.

- ❑ The content of the memory location is zero extended to 17 bits, if the optional uns keyword is applied to the input operand.
- ❑ 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 R keyword is applied to the instruction.
- ❑ 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 40 keyword is applied to the instruction.

This instruction provides the option to store the 16-bit data-memory operand 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, SMUL, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
MASM uns(*AR2+), uns(*AR3+), AC3	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	0
302	FE00	302	FE00
202	7000	202	7000
FRCT	0	FRCT	0

4.50 Negation (NEG)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	NEG [src,] dst	Yes	2	1	X

Operands dst, src

Description

This instruction computes the 2s complement of the content of the source register (src). This instruction clears the CARRY status bit to 0 for all nonzero values of src. If src equals 0, the CARRY status bit is set to 1.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - 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 according to SXMD.
  - Overflow detection and CARRY status bit depends on M40.
  - 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.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
NEG AC1, AC0	The 2s complement of the content of AC1 is stored in AC0.

4.51 No Operation (NOP)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	NOP	Yes	1	1	D
[2]	NOP_16	Yes	2	1	D

Operands none

Description

Instruction [1] increments the program counter register (PC) by 1 byte. Instruction [2] increments the PC by 2 bytes.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
NOP	The program counter (PC) is incremented by 1 byte.

4.52 Normalization

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MANT ACx, ACy :: NEXP ACx, Tx	Yes	3	1	X2
[2]	EXP ACx, Tx	Yes	3	1	X

Brief Description

Instruction [1] computes the exponent and mantissa of the source accumulator ACx. The computation of the exponent and the mantissa is executed in the D-unit shifter. The exponent is computed and stored in the temporary register Tx. The A-unit is used to make the move operation. The mantissa is stored in the accumulator ACy.

Instruction [2] computes the exponent of the source accumulator ACx in the D-unit shifter. The result of the operation is stored in the temporary register Tx. The A-unit ALU is used to make the move operation.

Instruction [2] produces in Tx the opposite result than computed by instruction [1].

Status Bits

Affected by none

Affects none

4.52.1 Normalization: MANT ACx, ACy :: NEXP ACx, Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MANT ACx, ACy :: NEXP ACx, Tx	Yes	3	1	X2

Operands

ACx, ACy, Tx

Description

This instruction computes the exponent and mantissa of the source accumulator ACx. The computation of the exponent and the mantissa is executed in the D-unit shifter. The exponent is computed and stored in the temporary register Tx. The A-unit is used to make the move operation. The mantissa is stored in the accumulator ACy.

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, bit 39 of ACx is extended to bit 31.
  - When shifting to the MSBs, 0 is inserted at bit position 0.
- If ACx is equal to 0, Tx is loaded with 8000h.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax

MANT AC0, AC1
:: NEXP AC0, T1

Description

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         21 0A0A 0A0A  AC0         21 0A0A 0A0A
AC1         FF FFFF F001  AC1         00 4214 1414
T1                 0000   T1                 0007
```

Syntax

MANT AC0, AC1
:: NEXP AC0, T1

Description

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
T1                 0000   T1                 0001
```

Syntax

MANT AC2
:: NEXP AC2, T2

Description

The exponent is computed by subtracting the number of leading bits in the content of AC2 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T2. The mantissa is computed by aligning the content of AC2 on a signed 32-bit representation. The mantissa value is stored in AC2.

```
Before                    After
AC2         00 0723 2400  AC2         00 7232 4000
T2                 0000   T2                 FFFC
```

Syntax

MANT AC1
:: NEXP AC1, T3

Description

The exponent is computed by subtracting the number of leading bits in the content of AC1 from 8. The exponent value is a signed 2s-complement value in the –31 to 8 range and is stored in T3. The mantissa is computed by aligning the content of AC1 on a signed 32-bit representation. The mantissa value is stored in AC1.

```
Before                    After
AC1         F9 3400 8600  AC0         FF 9340 0860
T3                 0000   T3                 0004
```

4.52.2 Exponent: EXP ACx, Tx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	EXP ACx, Tx	Yes	3	1	X

Operands

ACx, Tx

Description

This instruction computes the exponent of the source accumulator ACx in the D-unit shifter. The result of the operation is stored in the temporary register Tx. The A-unit ALU is used to make the move operation.

This exponent is a signed 2s-complement value in the –8 to 31 range. The exponent is computed by calculating the number of leading bits in ACx and subtracting 8 from this value. 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.

ACx is not modified after the execution of this instruction. If ACx is equal to 0, Tx is loaded with 0.

This instruction produces in Tx the opposite result than computed by instruction [1].

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Examples

Syntax	Description
EXP AC0, T1	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
AC0        FF FFFF FFCB     AC0        FF FFFF FFCB
T1                 0000     T1                 0019
```

Syntax

EXP AC0, T1

Description

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**	
AC0	07 8543 2105	AC0	07 8543 2105
T1	0000	T1	FFFC

4.53 Peripheral Port Register Access Qualifier (port)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	port(Smem)	No	1	1	D
[2]	port(k16)	No	3	1	D

Operands k16, Smem

Description

These operand qualifiers allow you to locally disable access toward the data memory and enable access to the 64K-word I/O space. The I/O data location is specified by the Smem, Xmem, or Ymem fields.

- ❑ An operand qualifier may be included in any instruction making a word single data memory access Smem or Xmem that is used to read a memory operand.
- ❑ An operand qualifier may be included in any instruction making a word single data memory access Smem or Ymem that is used to write a memory operand, except instructions using the keyword delay.
- ❑ An operand qualifier cannot be executed alone.

Any instruction making a word single data memory access Smem (except those listed above) can use the port(k16) addressing mode to access the 64K-word I/O space with an immediate address. When an instruction uses port(k16), the unsigned 16-bit constant, k16, is encoded in a 2-byte extension to the instruction. Because of the extension, an instruction using port(k16) cannot be executed in parallel with another instruction.

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 to the instruction. Because of the extension, an instruction using one of the following indirect operands cannot be executed with these operand qualifiers.

- ❑ *ARn(#K16)
- ❑ *+ARn(#K16)
- ❑ *CDP(#K16)
- ❑ *+CDP(#K16)

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV port(*CDP+), T2	The content addressed by CDP (I/O address) is loaded into T2. After being used for the address, CDP is incremented by 1.
MOV *CDP, port(#456h)	The content addressed by CDP is written to I/O address 456h.

4.54 Pop Extended Auxiliary Register from Stack Pointers (POPBOTH)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	POPBOTH xdst	Yes	2	1	X

Operands xdst

Description

This instruction moves the content of two 16-bit data memory locations addressed by the 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 upper 9 bits of the data memory addressed by SSP are discarded and only the 7 lower bits of the data memory are loaded into the high part of xdst(22–16).

When xdst is an accumulator, the guard bits, ACx(39–32), are reloaded (unchanged) with the current value and are not modified by this instruction.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

4.55 Pop Top of Stack (POP)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	POP dst1, dst2	Yes	2	1	X
[2]	POP dst	Yes	2	1	X
[3]	POP dst, Smem	No	3	1	X
[4]	POP dbl(ACx)	Yes	2	1	X
[5]	POP Smem	No	2	1	X
[6]	POP dbl(Lmem)	No	2	1	X

Brief Description

These instructions move the content of the data memory location addressed by the data stack pointer (SP) to:

- ❑ an accumulator, auxiliary, or temporary register
- ❑ a data memory location

When the destination register is an accumulator, the guard bits and the 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

4.55.1 Pop Top of Stack: POP dst1, dst2

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	POP dst1, dst2	Yes	2	1	X

Operands

dst

Description

This instruction moves the content of the 16-bit data memory location pointed by SP to destination register dst1 and moves the content of the 16-bit data memory location pointed by SP + 1 to destination register dst2.

When the destination register, dst1 or dst2, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15–0). The guard bits and the 16 higher bits of the accumulator, ACx(39–16), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
POP AC0, AC1	The content of the memory location pointed by the 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 9432	AC1	F7 5678 2300
SP	0300	SP	0302
300	4890	300	4890
301	2300	301	2300

4.55.2 Pop Top of Stack: POP dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	POP dst	Yes	2	1	X

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 register, dst, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15–0). The guard bits and the 16 higher bits of the accumulator, ACx(39–16), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 1.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
POP AC0	The content of the memory location pointed by the stack pointer (SP) is copied to AC0(15–0). Bits 39–16 of AC0 are unchanged. The SP is incremented by 1.

4.55.3 Pop Top of Stack: POP dst, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	POP dst, Smem	No	3	1	X

Operands dst, Smem

Description

This instruction moves the content of the 16-bit data memory location pointed by SP to destination register dst and moves the content of the 16-bit data memory location pointed by SP + 1 to data memory (Smem) location.

When the destination register, dst, is an accumulator, the content of the 16-bit data memory operand is moved to the destination accumulator low part, ACx(15–0). The guard bits and the 16 higher bits of the accumulator, ACx(39–16), are reloaded (unchanged) with the current value and are not modified by this instruction. SP is incremented by 2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
POP AC0, *AR3	The content of the memory location pointed by the stack pointer (SP) is copied to AC0(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.

4.55.4 Pop Top of Stack: POP dbl(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	POP dbl(ACx)	Yes	2	1	X

Operands

ACx

Description

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 (unchanged) with the current value and are not modified by this instruction. SP is incremented by 2.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
POP dbl(AC1)	The content of the memory location pointed by the 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
```

4.55.5 Pop Top of Stack: POP Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	POP Smem	No	2	1	X

Operands

Smem

Description

This instruction moves the content of the 16-bit data memory location pointed by SP to data memory (Smem) location. SP is incremented by 1.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
POP *AR1	The content of the memory location pointed by the 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

4.55.6 Pop Top of Stack: POP dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	POP dbl(Lmem)	No	2	1	X

Operands

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 16 lowest bits of the 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

This instruction can be repeated.

Example

Syntax	Description
POP dbl(*AR3–)	The content of the memory location pointed by the 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.

4.56 Push Extended Auxiliary Register to Stack Pointers (PSHBOTH)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	PSHBOTH xsrc	Yes	2	1	X

Operands xsrc

Description

This instruction moves the lower 32 bits of ACx or the content of the 23-bit source register (XARx, XSP, XSSP, XDP, or XCDP) to the two 16-bit memory locations addressed by the stack pointer (SP) and system stack pointer (SSP).

The content of xsrc(15–0) is moved to the location addressed by SP and the content of xsrc(31–16) is moved to the location addressed by SSP.

When xsrc is a 23-bit register, the upper 9 bits of the location addressed by SSP are filled with 0.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

4.57 Push to Top of Stack (PSH)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	PSH src1, src2	Yes	2	1	X
[2]	PSH src	Yes	2	1	X
[3]	PSH src,Smem	No	3	1	X
[4]	PSH dbl(ACx)	Yes	2	1	X
[5]	PSH Smem	No	2	1	X
[6]	PSH dbl(Lmem)	No	2	1	X

Brief 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

4.57.1 Push to Top of Stack: PSH src1, src2

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	PSH src1,src2	Yes	2	1	X

Operands

src

Description

This instruction decrements SP by 2, then moves the content of the source register src1 to the 16-bit data memory location pointed by SP and moves the content of the source register src2 to the 16-bit data memory location pointed by SP + 1.

When the source register, src1 or src2, is an accumulator, the source accumulator low part, ACx(15–0), is moved to the 16-bit data memory operand.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
PSH AR0, AC1	The 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

4.57.2 Push to Top of Stack: PSH src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	PSH src	Yes	2	1	X

Operands

src

Description

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 Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
PSH AC0	The SP is decremented by 1. The content of AC0(15–0) is copied to the memory location pointed by the stack pointer (SP).

4.57.3 Push to Top of Stack: PSH src, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	PSH src,Smem	No	3	1	X

Operands

Smem, src

Description

This instruction decrements SP by 2, then moves the content of the source register (src) to the 16-bit data memory location pointed by SP and moves the content of the data memory (Smem) location to the 16-bit data memory location pointed by SP + 1.

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 Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
PSH AC0, *AR3	The SP is decremented by 2. The content of AC0(15–0) is copied to the memory location pointed by the stack pointer (SP) and the content addressed by AR3 is copied to the memory location pointed by SP + 1.

4.57.4 Push to Top of Stack: PSH dbl(ACx)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	PSH dbl(ACx)	Yes	2	1	X

Operands

ACx

Description

This instruction decrements SP by 2, then moves the content of the accumulator high part ACx(31–16) to the 16-bit data memory location pointed by SP and moves the content of the accumulator low part ACx(15–0) to the 16-bit data memory location pointed by SP + 1.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
PSH dbl(AC0)	The SP is decremented by 2. The content of AC0(31–16) is copied to the memory location pointed by the stack pointer (SP) and the content of AC0(15–0) is copied to the memory location pointed by SP + 1.

4.57.5 Push to Top of Stack: PSH Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	PSH Smem	No	2	1	X

Operands

Smem

Description

This instruction decrements SP by 1, then moves the content of the data memory (Smem) location to the 16-bit data memory location pointed by SP.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
PSH *AR1	The SP is decremented by 1. The content addressed by AR1 is copied to the memory location pointed by the stack pointer (SP).

Before		After	
*AR1	6903	*AR1	6903
SP	0305	SP	0304
304	0000	304	6903
305	0300	305	0300

4.57.6 Push to Top of Stack: PSH dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	PSH dbl(Lmem)	No	2	1	X

Operands

Lmem

Description

This instruction decrements SP by 2, then moves the 16 highest bits of data memory location Lmem to the 16-bit data memory location pointed by SP and moves the 16 lowest bits of data memory location Lmem to the 16-bit data memory location pointed by SP + 1.

When Lmem is at an even address, the two 16-bit values pushed onto the stack are stored at memory location Lmem in the same order. When Lmem is at an odd address, the two 16-bit values pushed onto the stack are stored at memory location Lmem in the reverse order.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
PSH dbl(*AR3–)	The 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 the stack pointer (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.

4.58 Register Bit Test/Set/Clear/Complement

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BTST Baddr, src, TCx	No	3	1	X
[2]	BNOT Baddr, src	No	3	1	X
[3]	BCLR Baddr, src	No	3	1	X
[4]	BSET Baddr, src	No	3	1	X
[5]	BTSTP Baddr, src	No	3	1	X

Brief Description

These instructions perform bit manipulations:

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

These instructions manipulate a single bit (instructions [1]–[4]) or two consecutive bits (instruction [5]) of the source (src) register. The bit manipulated is defined by a bit address (Baddr). The following operations can be performed on the selected bit:

- ❑ copy the bit into TCx
- ❑ complement the bit in src
- ❑ clear the bit in src
- ❑ set the bit in src

Status Bits

Affected by none

Affects TCx

4.58.1 Register Bit Test: BTST Baddr, src, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BTST Baddr, src, TCx	No	3	1	X

Operands Baddr, src, TCx

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction tests a single bit of the source register location as defined by the bit addressing mode, Baddr. The tested bit is copied into the selected TCx status bit.

The generated bit address must be within:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, 0 is stored into the selected TCx status bit.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects TCx

Repeat

This instruction can be repeated.

Example

Syntax	Description
BTST @#12, T0, TC1	The bit at the position defined by the register bit address (12) in T0 is tested and the tested bit is copied into TC1.

```
Before                After
T0          FE00      T0          FE00
TC1            0      TC1            1
```

4.58.2 Register Bit Complement: BNOT Baddr, src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	BNOT Baddr, src	No	3	1	X

Operands

Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

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:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
BNOT AR1, T0	The bit at the position defined by the content of AR1(3–0) in T0 is complemented.

Before		After	
T0	E000	T0	F000
AR1	000C	AR1	000C

4.58.3 Register Bit Clear: BCLR Baddr, src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	BCLR Baddr, src	No	3	1	X

Operands

Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction clears to 0 a single bit, as defined by the bit addressing mode, Baddr, of the source register.

The generated bit address must be within:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

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

4.58.4 Register Bit Set: BSET Baddr, src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	BSET Baddr, src	No	3	1	X

Operands Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction sets to 1 a single bit, as defined by the bit addressing mode, Baddr, of the source register.

The generated bit address must be within:

- 0–39 when accessing accumulator bits (only the 6 LSBs of the generated bit address are used to determine the bit position). If the generated bit address is not within 0–39, the selected register bit value does not change.
- 0–15 when accessing auxiliary or temporary register bits (only the 4 LSBs of the generated address are used to determine the bit position).

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
BSET AR3, AC0	The bit at the position defined by the content of AR3(4–0) in AC0 is set to 1.

4.58.5 Register Bit Pair Test: BTSTP Baddr, src

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	BTSTP Baddr, src	No	3	1	X

Operands

Baddr, src

Description

This instruction performs a bit manipulation:

- In the D-unit ALU, if the source (src) register operand is an accumulator.
- In the A-unit ALU, if the source (src) register operand is an auxiliary or temporary register.

The instruction tests two consecutive bits of the source register location as defined by the bit addressing mode, Baddr, and Baddr + 1. The tested bits are 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 bit address is not within 0–38:
  - If the generated bit address is 39, bit 39 of the register is stored into 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 LSBs of the generated address are used to determine the bit position). If the generated bit address is not within 0–14:
  - If the generated bit address is 15, bit 15 of the register is stored into TC1 and 0 is stored into TC2.
  - In all other cases, 0 is stored into TC1 and TC2.

Status Bits

Affected by none

Affects TC1, TC2

Repeat

This instruction can be repeated.

Example

Syntax	Description
BTSTP AR1(T0), AC0	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	AC0	E0 1234 0000
AR1	0026	AR1	0026
T0	0001	T0	0001
TC1	0	TC1	1
TC2	0	TC2	0

4.59 Register Comparison (CMP)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	CMP[U] src RELOP dst, TCx	Yes	3	1	X
[2]	CMPAND[U] src RELOP dst, TCy, TCx	Yes	3	1	X
[3]	CMPAND[U] src RELOP dst, !TCy, TCx	Yes	3	1	X
[4]	CMPOR[U] src RELOP dst, TCy, TCx	Yes	3	1	X
[5]	CMPOR[U] src RELOP dst, !TCy, TCx	Yes	3	1	X

Brief 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. If the comparison is true, the TCx status bit is set to 1; otherwise, it is cleared to 0.

Status Bits

Affected by C54CM, M40, TCy

Affects TCx

4.59.1 Register Comparison:CMP[U] src RELOP dst, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	CMP[U] src RELOP dst, TCx	Yes	3	1	X

Operands dst, RELOP, src, TCx

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

U	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

Affects TCx

Repeat

This instruction can be repeated.

Example

Syntax	Description
CMP AC1 = = T1, TC1	The 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	T1	0400
TC1	0	TC1	1

Syntax	Description
CMP T1 > = AC1, TC1	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	
T1	0500	T1	0500
AC1	80 0000 0400	AC1	80 0000 0400
TC1	0	TC1	1

4.59.2 Register Comparison/AND:CMPAND[U] src RELOP dst, TCy, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	CMPAND[U] src RELOP dst, TCy, TCx	Yes	3	1	X

Operands dst, RELOP, src, TCx, TCy

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

U	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.

Example

Syntax	Description
CMPAND AC1 == AC2, TC1, TC2	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	1

4.59.3 Register Comparison/AND:CMPAND[U] src RELOP dst, !TCy, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	CMPAND[U] src RELOP dst, !TCy, TCx	Yes	3	1	X

Operands dst, RELOP, src, TCx, TCy

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

U	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.

Example

Syntax	Description
CMPAND AC1 == AC2, !TC1, TC2	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

4.59.4 Register Comparison/OR:CMPOR[U] src RELOP dst, TCy, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	CMPOR[U] src RELOP dst, TCy, TCx	Yes	3	1	X

Operands dst, RELOP, src, TCx, TCy

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

U	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.

Example

Syntax	Description
CMPOR AC1 != AR1, TC1, TC2	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

4.59.5 Register Comparison/OR:CMPOR[U] src RELOP dst, !TCy, TCx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	CMPOR[U] src RELOP dst, !TCy, TCx	Yes	3	1	X

Operands dst, RELOP, src, TCx, TCy

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

U	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.

Example

Syntax	Description
CMPOR AC1 != AR1, !TC1, TC2	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	TC2	0

4.60 Repeat Block of Instructions Unconditionally (RPTB)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RPTBLOCAL pmad	Yes	2	1	AD
[2]	RPTB pmad	Yes	3	1	AD

Brief Description

These instructions repeat a block of instructions the number of times specified by:

- ❏ the content of BRC0 + 1, if no loop has already been detected.
- ❏ the content of BRS1 + 1, if one level of the loop has already been detected.

Loop structures defined by these instructions must have the following characteristics:

- ❏ The minimum number of cycles executed within one loop iteration is 2 cycles.
- ❏ The maximum loop size is 64K bytes.
- ❏ 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.
- ❏ The block-repeat operation can only be cleared by branching to a destination address outside the active block-repeat loop.

These instructions cannot be repeated.

Status Bits

Affected by none

Affects none

4.60.1 Repeat Block of Instructions Unconditionally: RPTBLOCAL pmad

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RPTBLOCAL pmad	Yes	2	1	AD

Operands pmad

Description

This instruction repeats a block of instructions the number of times specified by:

- the content of BRC0 + 1, if no loop has already been detected. In this case:
  - In the address phase of the pipeline, RSA0 is loaded with the program address of the first instruction of the loop.
  - The program address (pmad) of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA0.
  - BRC0 is decremented at the address phase of the last instruction of the loop when its content is not equal to 0.
  - BRC0 contains 0 after the block-repeat operation has ended.
- the content of BRS1 + 1, if one level of the loop has already been detected. In this case:
  - BRC1 is loaded with the content of BRS1 in the address phase of the repeat block instruction.
  - In the address phase of the pipeline, RSA1 is loaded with the program address of the first instruction of the loop.
  - The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA1.
  - BRC1 is decremented at the address phase of the last instruction of the loop when its content is not equal to 0.
  - BRC1 contains 0 after the block-repeat operation has ended.
  - BRS1 content is not impacted by the block-repeat operation.

Loop structures defined by this instruction must have the following characteristics:

- The minimum number of cycles executed within one loop iteration is 2 cycles.
- 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.

A loop is defined as local when all the code of the loop is repeatedly executed from within the instruction buffer queue:

- Local loop sizes are limited to 61 bytes. (The assembler checks that the code size of the loop excluding the last (paralleled) instruction is less than or equal to 55 bytes.)
- Nested local repeat block (RPTBLOCAL) instructions are allowed.
- Local loop code must not include the following instructions:

B (branch)	RPTB	IDLE	RET
CALL	INTR	RESET	TRAP

- The only branch instructions allowed in a RPTBLOCAL 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 (BCC) 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).

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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RPTBLOCAL	A block of instructions is repeated as defined by the content of BRC0 + 1.

	Address	BRC0	RSA0	REA0	BRS1
MOV #3, BRC0		0003	0000	0000	0000
RPTBLOCAL {	004003	?*	4005	400D	?
… …	004005	?	?	?	?
… …	00400D	DTZ**	?	?	?
}		0000	4005	400D	0000

*?: Unchanged
**DTZ: Decrease till zero

4.60.2 Repeat Block of Instructions Unconditionally: RPTB pmad

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	RPTB pmad	Yes	3	1	AD

Operands pmad

Description

This instruction repeats a block of instructions the number of times specified by:

- the content of BRC0 + 1, if no loop has already been detected. In this case:
  - In the address phase of the pipeline, RSA0 is loaded with the program address of the first instruction of the loop.
  - The program address (pmad) of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA0.
  - BRC0 is decremented at the address phase of the last instruction of the loop when its content is not equal to 0.
  - BRC0 contains 0 after the block-repeat operation has ended.
- the content of BRS1 + 1, if one level of the loop has already been detected. In this case:
  - BRC1 is loaded with the content of BRS1 in the address phase of the repeat block instruction.
  - In the address phase of the pipeline, RSA1 is loaded with the program address of the first instruction of the loop.
  - The program address of the last instruction of the loop (that may be two parallel instructions) is computed in the address phase of the pipeline and stored in REA1.
  - BRC1 is decremented at the address phase of the last instruction of the loop when its content is not equal to 0.
  - BRC1 contains 0 after the block-repeat operation has ended.
  - BRS1 content is not impacted by the block-repeat operation.

Loop structures defined by these instructions must have the following characteristics:

- The minimum number of cycles executed within one loop iteration is 2 cycles.
- 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.

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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RPTB	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
MOV #3, BRC0		0003	0000	0000	0000	0000	0000	0000
MOV #1, BRC1		?*	?	?	0001	0001	?	?
RPTB {	004006	?	4009	4017	?	?	?	?
… …	004009	?	?	?	?	?	?	?
RPTBLOCAL {	00400B	?	?	?	?	(BRS1)	400D	4015
… …	00400D	?	?	?	?	?	?	?
… …	004015	?	?	?	?	DTZ**	?	?
}								
… …	004017	DTZ**	?	?	?	?	?	?
}		0000	4009	4017	0001	0000	400D	4015

*?: Unchanged
**DTZ: Decrease till zero

4.61 Repeat Single Instruction Conditionally (RPTCC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RPTCC k8, cond	Yes	3	1	AD

Operands cond, k8

Description

This instruction evaluates a single condition defined by the cond field and as long as the condition is true, the next instruction or the next two paralleled instructions is repeated the number of times specified by an 8-bit immediate value, k8 + 1. The maximum number of executions of a given instruction or paralleled instructions is 2^8 –1 (255).

The 8 LSBs of the repeat counter register (RPTC):

- ❏ Are loaded with the immediate value at the address phase of the pipeline.
- ❏ Are decremented by 1 in the decode phase of the repeated instruction.

The 8 MSBs of RPTC:

- ❏ Are loaded with the cond code at the address phase of the pipeline.
- ❏ Are untouched during the while() / repeat() structure execution.

At each step of the iteration, the condition defined by the cond field is tested in the execute phase of the pipeline. When the condition becomes false, the instruction repetition stops.

- ❏ If the condition becomes false at any execution of the repeated instruction, the 8 LSBs of RPTC are corrected to indicate exactly how many iterations were not performed.
- ❏ Since the condition is evaluated in the execute phase of the repeated instruction, when the condition is tested false, some of the succeeding iterations of that repeated instruction may have gone through the address, access, and read phases of the pipeline. Therefore, they may have modified the pointer registers used in the DAGEN units to generate data memory operands addresses in the address phase.

  When the while() / repeat() structure is exited, reading the computed single-repeat register (CSR) content enables you to determine how many instructions have gone through the address phase of the pipeline. You may then use the Repeat Single Instruction Unconditionally instruction [1] to rewind the pointer registers. Note that this must only be performed when a false condition has been met inside the while() / repeat() structure.

- 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 1st iteration	RPTCinit + 1	3
At 2nd iteration	RPTCinit	3
At 3rd iteration	RPTC – 1	3
…	…	…
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.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	RPT	RESET	IDLE	MOV dbl(Lmem), RETA
CALL	RPTB	INTR	XCC	
RET	RPTBLOCAL	TRAP	XCCPART	

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

Syntax	Description
RPTCC #7, AC1 > #0	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.

```
RPTCC #7, AC1 > #0                    address:  004004
                                                004008

… …                                             00400B
```

Before		After	
AC1	00 2359 0340	AC1	00 1FC2 7B40
T0	0340	T0	0340
*AR1	2354	*AR1	2354
RPTC	4106*	RPTC	0000

* At the address phase of the pipeline, RPTC is automatically initialized to 4107h and then is immediately decreased to 4106h.

4.62 Repeat Single Instruction Unconditionally (RPT)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RPT CSR	Yes	2	1	AD
[2]	RPTADD CSR, TAx	Yes	2	1	X
[3]	RPT k8	Yes	2	1	AD
[4]	RPTADD CSR, k4	Yes	2	1	X
[5]	RPTSUB CSR, k4	Yes	2	1	X
[6]	RPT k16	Yes	3	1	AD

Brief 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).

With the A-unit ALU, instructions [2], [4], and [5] allow the content of CSR to be modified. 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.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	IDLE	RESET	TRAP	MOV RPTC, TAx	32-bit instructions using *(#k23) absolute addressing mode
CALL	RPT	INTR	XCC	MOV dbl(Lmem), RETA	
RET	RPTB	RPTBLOCAL	XCCPART		

These instructions cannot be repeated.

Status Bits

Affected by none

Affects none

4.62.1 Repeat Single Instruction Unconditionally: RPT CSR

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RPT CSR	Yes	2	1	AD

Operands none

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. The repeat counter register (RPTC):

- ❏ 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.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	IDLE	RESET	TRAP	MOV RPTC, TAx	32-bit instructions using *(#k23) absolute addressing mode
CALL	RPT	INTR	XCC	MOV dbl(Lmem), RETA	
RET	RPTB	RPTBLOCAL	XCCPART		

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RPT CSR MACM *AR3+, *AR4+, AC1	The single instruction following the repeat instruction is repeated as defined 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	FE00	202	FE00
203	23DC	203	23DC
400	D768	400	D768
401	6987	401	6987
402	3400	402	3400
403	7900	403	7900

4.62.2 Repeat Single Instruction Unconditionally: RPTADD CSR, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	RPTADD CSR, TAx	Yes	2	1	X

Operands TAx

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. The repeat counter register (RPTC):

- ❏ 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.

With the A-unit ALU, this instruction allows the content of CSR to be incremented by the content of TAx. 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 this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	IDLE	RESET	TRAP	MOV RPTC, TAx	32-bit instructions using *(#k23) absolute addressing mode
CALL	RPT	INTR	XCC	MOV dbl(Lmem), RETA	
RET	RPTB	RPTBLOCAL	XCCPART		

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RPTADD 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.

4.62.3 Repeat Single Instruction Unconditionally: RPT kx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	RPT k8	Yes	2	1	AD
[6]	RPT k16	Yes	3	1	AD

Operands

kx

Description

This instruction repeats the next instruction or the next two paralleled instructions the number of times specified an immediate value, kx + 1. The repeat counter register (RPTC):

- Is loaded with the immediate value 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.

The repeat single mechanism triggered by this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	IDLE	RESET	TRAP	MOV RPTC, TAx	32-bit instructions using *(#k23) absolute addressing mode
CALL	RPT	INTR	XCC	MOV dbl(Lmem), RETA	
RET	RPTB	RPTBLOCAL	XCCPART		

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Examples

Syntax	Description
RPT #3 MACM *AR3+, *AR4+, AC1	The single instruction following the repeat instruction is repeated four times. This instruction example provides the same result as instruction [1].

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
402	3400	402	3400
403	7900	403	7900

Syntax	Description
RPT #513	A single instruction is repeated as defined by the unsigned 16-bit value + 1 (513 + 1).

4.62.4 Repeat Single Instruction Unconditionally: RPTADD CSR, k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	RPTADD CSR, k4	Yes	2	1	X

Operands

k4

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. The repeat counter register (RPTC):

- 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.

With the A-unit ALU, this instruction allows the content of CSR to be incremented by k4. 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 this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	IDLE	RESET	TRAP	MOV RPTC, TAx	32-bit instructions using *(#k23) absolute addressing mode
CALL	RPT	INTR	XCC	MOV dbl(Lmem), RETA	
RET	RPTB	RPTBLOCAL	XCCPART		

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RPTADD 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).

4.62.5 Repeat Single Instruction Unconditionally: RPTSUB CSR, k4

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	RPTSUB CSR, k4	Yes	2	1	X

Operands

k4

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. The repeat counter register (RPTC):

- ❏ 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.

With the A-unit ALU, this instruction allows the content of CSR to be decremented by k4. 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 this instruction is interruptible.

Two paralleled instructions can be repeated when following the parallelism general rules.

The following instructions cannot be used in a repeat single mechanism:

B (branch)	IDLE	RESET	TRAP	MOV RPTC, TAx	32-bit instructions using *(#k23) absolute addressing mode
CALL	RPT	INTR	XCC	MOV dbl(Lmem), RETA	
RET	RPTB	RPTBLOCAL	XCCPART		

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RPTSUB 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).

4.63 Return Conditionally (RETCC)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles†	Pipeline
[1]	RETCC cond	Yes	3	5/5	R

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

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.

When the return from subroutine occurs:

- The program counter (PC) is loaded with RETA content (the return address of the calling subroutine). The active control flow execution context flags are updated with the CFCT content.
- The 16 LSBs of RETA are popped from the top of the stack pointer (SP). The SP is incremented by 1 word.
- The 8 MSBs of RETA and CFCT are popped from the top of the system stack pointer (SSP). The SSP is incremented by 1 word.

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

Syntax	Description
RETCC ACOV0 = #0	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	

4.64 Return Unconditionally (RET)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RET	Yes	2	5	D

Operands none

Description

This instruction passes control back to the calling subroutine.

- The program counter (PC) is loaded with RETA content (the return address of the calling subroutine). The active control flow execution context flags are updated with the CFCT content.
- The 16 LSBs of RETA are popped from the top of the stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.
- The 8 MSBs of RETA and CFCT 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.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RET	The program counter is loaded with the return address of the calling subroutine.

4.65 Return from Interrupt (RETI)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RETI	Yes	2	5	D

Operands

none

Description

This instruction passes control back to the interrupted task.

- The program counter (PC) is loaded with RETA content (the return address of the interrupted task). The active control flow execution context flags are updated with the CFCT content.
- The 16 LSBs of RETA are popped from the top of the stack pointer (SP). The SP is incremented by 1 word in the address phase of the pipeline.
- The 8 MSBs of RETA and CFCT 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 status register 1 (ST1_55) content is popped from the top of SP. The SP is incremented by 1 word in the access phase of the pipeline.
- The debug status register (DBGSTAT) content is popped from the top of SSP. The SSP is incremented by 1 word in the access phase of the pipeline.
- The 16 bits of status register 2, ST2_55[15–0], are popped from the top of SP. The SP is incremented by 1 word in the read phase of the pipeline.
- The 7 higher bits of status register 0, ST0_55[15–9], are popped from the top of SSP. The SSP is incremented by 1 word in the read phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
RETI	The program counter (PC) is loaded with the return address of the interrupted task.

4.66 Rotate Left (ROL)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ROL BitOut, src, BitIn, dst				
[1a]	ROL TC2, src, TC2, dst	Yes	3	1	X
[1b]	ROL TC2, src, CARRY, dst	Yes	3	1	X
[1c]	ROL CARRY, src, TC2, dst	Yes	3	1	X
[1d]	ROL CARRY, src, CARRY, dst	Yes	3	1	X

Operands CARRY, dst, src, TC2

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.

Examples

Syntax	Description
ROL CARRY, AC1, TC2, AC1	The value of TC2 before the execution of the instruction (1) 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	0F E340 5678	AC1	00 C680 ACF1
TC2	1	TC2	1
CARRY	1	CARRY	1
M40	0	M40	0

4.67 Rotate Right (ROR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ROR BitIn, src, BitOut, dst				
[1a]	ROR TC2, src, TC2, dst	Yes	3	1	X
[1b]	ROR TC2, src, CARRY, dst	Yes	3	1	X
[1c]	ROR CARRY, src, TC2, dst	Yes	3	1	X
[1d]	ROR CARRY, src, CARRY, dst	Yes	3	1	X

Operands CARRY, dst, src, TC2

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
- 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.

Examples

Syntax	Description
ROR TC2, AC0, TC2, AC1	The value of TC2 before the execution of the instruction (1) 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

4.68 Round

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	ROUND [ACx,] ACy	Yes	2	1	X

Operands ACx, ACy

Description

This instruction performs a rounding of the source accumulator ACx in the D-unit.

- The rounding operation depends on RDM:
  - When RDM = 0, the biased rounding to the infinite is performed. 8000h (2^{15}) is added to the 40-bit source accumulator ACx.
  - When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit source accumulator ACx, 8000h (2^{15}) is added:

```
if( 8000h < bit(15–0) < 10000h)
   add 8000h to the 40-bit source accumulator ACx
else if( bit(15–0) == 8000h)
   if( bit(16) == 1)
   add 8000h to the 40-bit source accumulator ACx
```

    If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.
- Addition overflow detection depends on M40.
- No addition carry report is stored in CARRY status bit.
- 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.

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the LSBs of accumulator ACx.

Status Bits

Affected by C54CM, M40, RDM, SATD

Affects ACOVy

Repeat

This instruction cannot be repeated.

Examples

Syntax	Description
ROUND AC0, AC1	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 0FF0 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

4.69 Saturate (SAT)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SAT[R] [ACx,] ACy	Yes	2	1	X

Operands ACx, ACy

Description

This instruction performs a saturation of the source accumulator ACx to the 32-bit width frame in the D-unit ALU.

- A rounding is performed if the optional R keyword is applied to the instruction. The rounding operation depends on RDM:
  - When RDM = 0, the biased rounding to the infinite is performed. 8000h (2^{15}) is added to the 40-bit source accumulator ACx.
  - When RDM = 1, the unbiased rounding to the nearest is performed. According to the value of the 17 LSBs of the 40-bit source accumulator ACx, 8000h (2^{15}) is added:

```
if( 8000h < bit(15–0) < 10000h)
   add 8000h to the 40-bit source accumulator ACx
else if( bit(15–0) == 8000h)
   if( bit(16) == 1)
   add 8000h to the 40-bit source accumulator ACx
```

  If a rounding has been performed, the 16 lowest bits of the result are cleared to 0.
- An overflow is detected at bit position 31.
- No addition carry report is stored in CARRY status bit.
- If an overflow is detected, the destination accumulator overflow status bit (ACOVy) is set.
- When an overflow is detected, the destination register is saturated. Saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow).

Compatibility with C54x devices (C54CM = 1)

When this instruction is executed with M40 = 0, compatibility is ensured. When C54CM = 1, the rounding is performed without clearing the LSBs of accumulator ACx.

Status Bits

Affected by C54CM, RDM

Affects ACOVy

Repeat

This instruction can be repeated.

Examples

Syntax	Description
SAT AC0, AC1	The 32-bit width content of AC0 is saturated and the saturated value, FF 8000 0000, is stored in AC1.

Before		After	
AC0	EF 0FF0 8023	AC0	EF 0FF0 8023
AC1	00 0000 0000	AC1	FF 8000 0000
ACOV1	0	ACOV1	1

Syntax	Description
SATR AC0, AC1	The 32-bit width content of AC0 is saturated. The saturated value, 00 7FFF FFFFh, 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

4.70 Signed Shift (SFTS)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SFTS dst, #–1	Yes	2	1	X
[2]	SFTS dst, #1	Yes	2	1	X
[3]	SFTS ACx, Tx[, ACy]	Yes	2	1	X
[4]	SFTSC ACx, Tx[, ACy]	Yes	2	1	X
[5]	SFTS ACx, #SHIFTW[, ACy]	Yes	3	1	X
[6]	SFTSC ACx, #SHIFTW[, ACy]	Yes	3	1	X

Brief Description

These instructions perform an unsigned shift by an immediate value, 1 or SHIFTW, or the content of a temporary register (Tx):

- In the D-unit shifter, if the destination operand is an accumulator (ACx).
- In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).

Status Bits

Affected by C54CM, M40, SATA, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

4.70.1 Signed Shift Right: SFTS dst, #–1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SFTS dst, #–1	Yes	2	1	X

Operands dst

Description

This instruction shifts right by 1 bit the content of the destination register (dst).

If the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted right by 1 bit:
  - 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.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- After shifting, unless otherwise noted, when M40 = 1:
  - overflow is detected at bit position 39
  - 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)

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.

Example

Syntax	Description
SFTS AC0, #–1	The content of AC0 is shifted right by 1 bit and the result is stored in AC0.

4.70.2 Signed Shift Left: SFTS dst, #1

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	SFTS dst, #1	Yes	2	1	X

Operands dst

Description

This instruction shifts left by 1 bit the content of the destination register (dst).

If the destination operand (dst) is an accumulator:

- The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted left by 1 bit:
  - 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
- The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 = 0, comparison is performed versus bit 31
  - if M40 = 1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40.
- 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)
  - if SATD = 1, when an overflow is detected, the destination accumulator saturation values are 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- 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)
  - 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)

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.

Example

Syntax	**Description**
SFTS T2, #1	The content of T2 is shifted left by 1 bit and the result is stored in T2.

Before		**After**	
T2	EF27	T2	DE4E
SATA	1	SATA	1

4.70.3 Signed Shift: SFTS ACx, Tx[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	SFTS ACx, Tx[, ACy]	Yes	2	1	X

Operands ACx, ACy, Tx

Description

This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content. If the 16-bit value contained in Tx is out of the –32 to +31 range, the shift is saturated to –32 or +31 and the shift operation is performed with this value; a destination accumulator overflow is reported when such saturation occurs.

- The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the Tx content:
  - 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
- The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 = 0, comparison is performed versus bit 31
  - if M40 = 1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (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 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- 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.
- 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

Affects ACOVy

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTS AC1, T0, AC0	The content of AC1 is shifted by the content of T0 and the result is stored in AC0.

4.70.4 Signed Shift: SFTSC ACx, Tx[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	SFTSC ACx, Tx[, ACy]	Yes	2	1	X

Operands ACx, ACy, Tx

Description

This instruction shifts by the temporary register (Tx) content the accumulator (ACx) content and stores the shifted-out bit in the CARRY status bit. If the 16-bit value contained in Tx is out of the –32 to +31 range, the shift is saturated to –32 or +31 and the shift operation is performed with this value; a destination accumulator overflow is reported when such saturation occurs.

- The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the Tx content:
  - 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
- The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 = 0, comparison is performed versus bit 31
  - if M40 = 1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40 and stored in the CARRY status bit. When the shift count is zero, Tx = 0, the CARRY status bit is cleared to 0.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (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 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- 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.
- 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTSC AC2, T1	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 AA00 1234	AC2	FF 8000 0000
T1	0005	T1	0005
CARRY	0	CARRY	1
M40	0	M40	0
ACOV2	0	ACOV2	1
SXMD	1	SXMD	1
SATD	1	SATD	1

4.70.5 Signed Shift: SFTS ACx, #SHIFTW[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	SFTS ACx, #SHIFTW[, ACy]	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

This instruction shifts by a 6-bit value, SHIFTW, the accumulator (ACx) content.

- The operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the 6-bit value, SHIFTW:
  - 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
- The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 = 0, comparison is performed versus bit 31
  - if M40 = 1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (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 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- 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

Repeat

This instruction can be repeated.

Example

Syntax	Description
SFTS AC1, #31, AC0	The content of AC1 is shifted left by 31 bits and the result is stored in AC0.
SFTS AC1, #–32	The content of AC1 is shifted right by 32 bits and the result is stored in AC1.

4.70.6 Signed Shift: SFTSC ACx, #SHIFTW[, ACy]

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	SFTSC ACx, #SHIFTW[, ACy]	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

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 operation is performed on 40 bits in the D-unit shifter.
- When M40 = 0, the input to the shifter is modified according to SXMD and then the modified input is shifted by the 6-bit value, SHIFTW:
  - 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
- The sign position of the source operand is compared to the shift quantity. This comparison depends on M40:
  - if M40 = 0, comparison is performed versus bit 31
  - if M40 = 1, comparison is performed versus bit 39
- 0 is inserted at bit position 0.
- The shifted-out bit is extracted according to M40 and stored in the CARRY status bit. When the shift count is zero, SHIFTW = 0, the CARRY status bit is cleared to 0.
- After shifting, unless otherwise noted, when M40 = 0:
  - overflow is detected at bit position 31 (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 00 7FFF FFFFh (positive overflow) or FF 8000 0000h (negative overflow)
- 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.

Example

Syntax	Description
SFTSC AC0, #–5, AC1	The content of AC0 is shifted right by 5 bits and the result is stored in AC1. The shifted out bit is stored in the 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

4.71 Software Interrupt (INTR)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	INTR k5	No	2	3	D

Operands k5

Description

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 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.

When the control is passed to the ISR:

- ❏ The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 bits of status register 2, ST2_55[15–0], are 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 to 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 (DBGSTAT) 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 RETA are pushed to the top of SP.
- ❏ The SSP is decremented by 1 word in the read phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- ❏ The return address of the interrupt is saved in RETA. The active control flow execution context flags are saved in CFCT.
- ❏ The program counter (PC) is loaded with the ISR program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects INTM

Repeat

This instruction cannot be repeated.

Example

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).

4.72 Software Reset (RESET)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	RESET	No	2	?	D

Operands none

Description

This instruction performs a nonmaskable software reset that can be used any time to put the device in a known state.

The reset instruction affects ST0_55, ST1_55, ST2_55, IFR0, and IFR1 registers but does not affect status register ST3_55 and interrupt vectors pointer registers (IVPD and IVPH). 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.

Status Bits

Affected by none

Affects IFR0, IFR1, ST0_55, ST1_55, ST2_55

Repeat

This instruction cannot be repeated.

4.73 Software Trap (TRAP)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	TRAP k5	No	2	?	D

Operands k5

Description

This instruction passes control to a specified interrupt service routine (ISR). 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 and this instruction does not affect INTM bit. This instruction is not maskable.

When the control is passed to the ISR:

- ❑ The stack pointer (SP) is decremented by 1 word in the address phase of the pipeline. The 16 bits of status register 2, ST2_55[15–0], are 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 to 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 (DBGSTAT) 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 RETA are pushed to the top of SP.
- ❑ The SSP is decremented by 1 word in the read phase of the pipeline. The 8 MSBs of RETA and the control flow execution context flags register (CFCT) are pushed to the top of SSP.
- ❑ The return address of the interrupt is saved in RETA. The active control flow execution context flags are saved in CFCT.
- ❑ The program counter (PC) is loaded with the ISR program address. The active control flow execution context flags are cleared.

Status Bits

Affected by none

Affects none

Repeat

This instruction cannot be repeated.

Example

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).

4.74 Specific CPU Register Load

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV k12, BK03	Yes	3	1	AD
[2]	MOV k12, BK47	Yes	3	1	AD
[3]	MOV k12, BKC	Yes	3	1	AD
[4]	MOV k12, BRC0	Yes	3	1	AD
[5]	MOV k12, BRC1	Yes	3	1	AD
[6]	MOV k12, CSR	Yes	3	1	AD
[7]	MOV k7, DPH	Yes	3	1	AD
[8]	MOV k9, PDP	Yes	3	1	AD
[9]	MOV k16, BSA01	No	4	1	AD
[10]	MOV k16, BSA23	No	4	1	AD
[11]	MOV k16, BSA45	No	4	1	AD
[12]	MOV k16, BSA67	No	4	1	AD
[13]	MOV k16, BSAC	No	4	1	AD
[14]	MOV k16, CDP	No	4	1	AD
[15]	MOV k16, DP	No	4	1	AD
[16]	MOV k16, SP	No	4	1	AD
[17]	MOV k16, SSP	No	4	1	AD
[18]	MOV Smem, BK03	No	3	1	X
[19]	MOV Smem, BK47	No	3	1	X
[20]	MOV Smem, BKC	No	3	1	X
[21]	MOV Smem, BSA01	No	3	1	X
[22]	MOV Smem, BSA23	No	3	1	X
[23]	MOV Smem, BSA45	No	3	1	X
[24]	MOV Smem, BSA67	No	3	1	X
[25]	MOV Smem, BSAC	No	3	1	X
[26]	MOV Smem, BRC0	No	3	1	X
[27]	MOV Smem, BRC1	No	3	1	X
[28]	MOV Smem, CDP	No	3	1	X
[29]	MOV Smem, CSR	No	3	1	X
[30]	MOV Smem, DP	No	3	1	X
[31]	MOV Smem, DPH	No	3	1	X
[32]	MOV Smem, PDP	No	3	1	X
[33]	MOV Smem, SP	No	3	1	X
[34]	MOV Smem, SSP	No	3	1	X
[35]	MOV Smem, TRN0	No	3	1	X

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[36]	MOV Smem, TRN1	No	3	1	X
[37]	MOV dbl(Lmem), RETA	No	3	5	X

Brief Description

These instructions load an unsigned constant, kx, the content of a memory (Smem) location, or the content of a data memory operand (Lmem) to a selected destination CPU register.

Status Bits

Affected by none

Affects none

4.74.1 Specific CPU Register Load: MOV kx, cpureg

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV k12, BK03	Yes	3	1	AD
[2]	MOV k12, BK47	Yes	3	1	AD
[3]	MOV k12, BKC	Yes	3	1	AD
[4]	MOV k12, BRC0	Yes	3	1	AD
[5]	MOV k12, BRC1	Yes	3	1	AD
[6]	MOV k12, CSR	Yes	3	1	AD
[7]	MOV k7, DPH	Yes	3	1	AD
[8]	MOV k9, PDP	Yes	3	1	AD
[9]	MOV k16, BSA01	No	4	1	AD
[10]	MOV k16, BSA23	No	4	1	AD
[11]	MOV k16, BSA45	No	4	1	AD
[12]	MOV k16, BSA67	No	4	1	AD
[13]	MOV k16, BSAC	No	4	1	AD
[14]	MOV k16, CDP	No	4	1	AD
[15]	MOV k16, DP	No	4	1	AD
[16]	MOV k16, SP	No	4	1	AD
[17]	MOV k16, SSP	No	4	1	AD

Operands

kx

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 loaded with the same value.

The operation is performed in the address phase of the pipeline.

Status Bits

Affected by none

Affects none

Repeat

Instruction [15] cannot be repeated; all other instructions can be repeated.

4.74.2 Specific CPU Register Load: MOV Smem, cpureg

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	MOV Smem, BK03	No	3	1	X
[19]	MOV Smem, BK47	No	3	1	X
[20]	MOV Smem, BKC	No	3	1	X
[21]	MOV Smem, BSA01	No	3	1	X
[22]	MOV Smem, BSA23	No	3	1	X
[23]	MOV Smem, BSA45	No	3	1	X
[24]	MOV Smem, BSA67	No	3	1	X
[25]	MOV Smem, BSAC	No	3	1	X
[26]	MOV Smem, BRC0	No	3	1	X
[27]	MOV Smem, BRC1	No	3	1	X
[28]	MOV Smem, CDP	No	3	1	X
[29]	MOV Smem, CSR	No	3	1	X
[30]	MOV Smem, DP	No	3	1	X
[31]	MOV Smem, DPH	No	3	1	X
[32]	MOV Smem, PDP	No	3	1	X
[33]	MOV Smem, SP	No	3	1	X
[34]	MOV Smem, SSP	No	3	1	X
[35]	MOV Smem, TRN0	No	3	1	X
[36]	MOV Smem, TRN1	No	3	1	X

Operands Smem

Description

This instruction loads 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 [27], when BRC1 is loaded, the block repeat save register (BRS1) is loaded with the same value.

Status Bits

Affected by none

Affects none

Repeat

Instruction [30] cannot be repeated; all other instructions can be repeated.

4.74.3 Return Address Register Load: MOV dbl(Lmem), RETA

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[37]	MOV dbl(Lmem), RETA	No	3	5	X

Operands Lmem

Description

This instruction 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 this instruction 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

This instruction can be repeated.

4.75 Specific CPU Register Move

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV TAx, BRC0	Yes	2	1	X
[2]	MOV TAx, BRC1	Yes	2	1	X
[3]	MOV TAx, CDP	Yes	2	1	X
[4]	MOV TAx, CSR	Yes	2	1	X
[5]	MOV BRC0, TAx	Yes	2	1	X
[6]	MOV BRC1, TAx	Yes	2	1	X
[7]	MOV CDP, TAx	Yes	2	1	X
[8]	MOV RPTC, TAx	Yes	2	1	X
[9]	MOV SP, TAx	Yes	2	1	X
[10]	MOV SSP, TAx	Yes	2	1	X
[11]	MOV TAx, SP	Yes	2	1	X
[12]	MOV TAx, SSP	Yes	2	1	X

Brief Description

These instructions move the content of the auxiliary or temporary register (TAx) to the selected CPU register, or move 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.

Status Bits

Affected by none

Affects none

4.75.1 Auxiliary or Temporary Register Move: MOV TAx, cpureg

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV TAx, BRC0	Yes	2	1	X
[2]	MOV TAx, BRC1	Yes	2	1	X
[3]	MOV TAx, CDP	Yes	2	1	X
[4]	MOV TAx, CSR	Yes	2	1	X
[11]	MOV TAx, SP	Yes	2	1	X
[12]	MOV TAx, SSP	Yes	2	1	X

Operands

TAx

Description

This instruction moves the content of the auxiliary or temporary register (TAx) to the selected CPU register. 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.

There is a 3-cycle latency between SP, SSP, CDP, TAx, CSR, and BRCx update and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

For instruction [2] when BRC1 is loaded with the content of TAx, the block repeat save register (BRS1) is also loaded with the same value.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV T1, BRC1	The content of T1 is copied to the block repeat register (BRC1) and to the block repeat save register (BRS1).

Before		After	
T1	0034	T1	0034
BRC1	00EA	BRC1	0034
BRS1	00EA	BRS1	0034

4.75.2 BRCx Register Move: MOV BRCx, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	MOV BRC0, TAx	Yes	2	1	X
[6]	MOV BRC1, TAx	Yes	2	1	X

Operands

TAx

Description

This instruction moves the content of the selected BRCx to the auxiliary or temporary register (TAx). Since BRCx is decremented in the address phase of the last instruction of a loop, these move instructions have a 3-cycle latency requirement versus the last instruction of a loop.

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.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV BRC1, T1	The content of block repeat register (BRC1) is copied to T1.

```
Before                  After
T1          0034        T1          00EA
BRC1        00EA        BRC1        00EA
```

4.75.3 Specific CPU Register Move: MOV cpureg, TAx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	MOV CDP, TAx	Yes	2	1	X
[8]	MOV RPTC, TAx	Yes	2	1	X
[9]	MOV SP, TAx	Yes	2	1	X
[10]	MOV SSP, TAx	Yes	2	1	X

Operands

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 [7], [9], and [10], there is a 3-cycle latency between SP, SSP, CDP, and TAx update and their use in the address phase by the A-unit address generator units or by the P-unit loop control management.

Status Bits

Affected by none

Affects none

Repeat

Instructions [7], [9], and [10] can be repeated. Instruction [8] cannot be repeated.

Example

Syntax	Description
MOV RPTC, T1	The content of single-repeat counter (RPTC) is copied to T1.

Before		After	
T1	0034	T1	00EA
RPTC	00EA	RPTC	00EA

4.76 Specific CPU Register Store

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV BK03, Smem	No	3	1	X
[2]	MOV BK47, Smem	No	3	1	X
[3]	MOV BKC, Smem	No	3	1	X
[4]	MOV BSA01, Smem	No	3	1	X
[5]	MOV BSA23, Smem	No	3	1	X
[6]	MOV BSA45, Smem	No	3	1	X
[7]	MOV BSA67, Smem	No	3	1	X
[8]	MOV BSAC, Smem	No	3	1	X
[9]	MOV BRC0, Smem	No	3	1	X
[10]	MOV BRC1, Smem	No	3	1	X
[11]	MOV CDP, Smem	No	3	1	X
[12]	MOV CSR, Smem	No	3	1	X
[13]	MOV DP, Smem	No	3	1	X
[14]	MOV DPH, Smem	No	3	1	X
[15]	MOV PDP, Smem	No	3	1	X
[16]	MOV SP, Smem	No	3	1	X
[17]	MOV SSP, Smem	No	3	1	X
[18]	MOV TRN0, Smem	No	3	1	X
[19]	MOV TRN1, Smem	No	3	1	X
[20]	MOV RETA, dbl(Lmem)	No	3	5	X

Brief Description

These instructions store the content of the selected source CPU register to a memory (Smem) location or a data memory operand (Lmem).

Status Bits

Affected by none

Affects none

4.76.1 Specific CPU Register Store: MOV cpureg, Smem

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV BK03, Smem	No	3	1	X
[2]	MOV BK47, Smem	No	3	1	X
[3]	MOV BKC, Smem	No	3	1	X
[4]	MOV BSA01, Smem	No	3	1	X
[5]	MOV BSA23, Smem	No	3	1	X
[6]	MOV BSA45, Smem	No	3	1	X
[7]	MOV BSA67, Smem	No	3	1	X
[8]	MOV BSAC, Smem	No	3	1	X
[9]	MOV BRC0, Smem	No	3	1	X
[10]	MOV BRC1, Smem	No	3	1	X
[11]	MOV CDP, Smem	No	3	1	X
[12]	MOV CSR, Smem	No	3	1	X
[13]	MOV DP, Smem	No	3	1	X
[14]	MOV DPH, Smem	No	3	1	X
[15]	MOV PDP, Smem	No	3	1	X
[16]	MOV SP, Smem	No	3	1	X
[17]	MOV SSP, Smem	No	3	1	X
[18]	MOV TRN0, Smem	No	3	1	X
[19]	MOV TRN1, Smem	No	3	1	X

Operands

Smem

Description

This instruction stores the content of the source CPU register to a memory (Smem) location.

The block repeat register (BRCx) is decremented in the address phase of the last instruction of the loop. Instructions [9] and [10] have a 3-cycle latency requirement versus the last instruction of the loop.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV SP, *AR1+	The content of the stack pointer (SP) is stored in the location addressed by AR1. AR1 is incremented by 1.

Before		After	
AR1	0200	AR1	0201
SP	0200	SP	0200
200	0000	200	0200

Syntax	Description
MOV SSP, *AR1+	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	00FF	201	0000

Syntax	Description
MOV TRN0, *AR1+	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

Syntax	Description
MOV TRN1, *AR1+	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
TRN1	0020	TRN1	0020
203	0000	203	0020

4.76.2 Return Address Register Store: MOV RETA, dbl(Lmem)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[20]	MOV RETA, dbl(Lmem)	No	3	5	X

Operands

Lmem

Description

This instruction stores 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) to the data memory 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 this instruction 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

This instruction can be repeated.

Example

Syntax	Description
MOV RETA, dbl(*AR3)	The contents of the RETA and CFCT are stored in the location addressed by AR3 and AR3 + 1.

4.77 Square Distance (SQDST)

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SQDST Xmem, Ymem, ACx, ACy	No	4	1	X

Operands ACx, ACy, Xmem, Ymem

Description

This instruction performs two parallel operations: multiply and accumulate (MAC), and subtract. The operations are executed in the D-unit MAC and D-unit ALU:

```
ACy = ACy + (ACx * ACx)
:: ACx = (Xmem << #16) – (Ymem << #16)
```

The first operation performs a multiplication and an accumulation in the D-unit MAC. The input operands of the multiplier 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.
- ❏ Addition overflow detection depends on M40. If an overflow is detected, the destination accumulator overflow status bit is set.
- ❏ When an addition overflow is detected, the accumulator is saturated according to SATD.

The second operation subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, 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 identical 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, 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 FRCT, SMUL, C54CM, M40, SATD, SXMD

Affects ACOVx, ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SQDST *AR0, *AR1, AC0, AC1	The content of AC0 squared is added to the content of AC1 and the result is stored in AC1. The content addressed by AR1 shifted left by 16 bits is subtracted from the content addressed by AR0 shifted left by 16 bits and the result is stored in AC0.

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

4.78 Status Bit Set/Clear

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BCLR k4, ST0_55	Yes	2	1	X
[2]	BSET k4, ST0_55	Yes	2	1	X
[3]	BCLR k4, ST1_55	Yes	2	1	X
[4]	BSET k4, ST1_55	Yes	2	1	X
[5]	BCLR k4, ST2_55	Yes	2	1	X
[6]	BSET k4, ST2_55	Yes	2	1	X
[7]	BCLR k4, ST3_55	Yes	2	1†	X
[8]	BSET k4, ST3_55	Yes	2	1†	X
[9]	BCLR f–name	Yes	2	1†	X
[10]	BSET f–name	Yes	2	1†	X

† When instruction [7]– [10] 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.

Brief Description

These instructions perform a bit manipulation in the A-unit ALU.

These instructions set to 1 or clear to 0 a single bit, as defined by a 4-bit immediate value, k4, or the one-bit-wide status bit field name, f–name, in the selected status register (ST0_55, ST1_55, ST2_55, or ST3_55). Note that C55x DSP status bit mapping does not correspond to C54x DSP status bits.

These instructions cannot be repeated.

Status Bits

Affected by none

Affects Selected status bits

4.78.1 Status Bit Clear: BCLR k4, STx_55

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	BCLR k4, ST0_55	Yes	2	1	X
[3]	BCLR k4, ST1_55	Yes	2	1	X
[5]	BCLR k4, ST2_55	Yes	2	1	X
[7]	BCLR k4, ST3_55	Yes	2	1†	X

† When instruction [7] 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.

Operands

k4, STx_55

Description

These instructions perform a bit manipulation in the A-unit ALU.

These instructions clear to 0 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0_55, ST1_55, ST2_55, or ST3_55).

Compatibility with C54x devices (C54CM = 1)

C55x DSP status bit mapping does not correspond to C54x DSP status bits.

Status Bits

Affected by none

Affects Selected status bits

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
BCLR AR1LC, ST2_55	The ST2_55 bit position defined by the label (AR1LC, bit 2) is cleared to 0.

```
Before                  After
ST2_55          0006    ST2_55          0002
```

4.78.2 Status Bit Set: BSET k4, STx_55

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	BSET k4, ST0_55	Yes	2	1	X
[4]	BSET k4, ST1_55	Yes	2	1	X
[6]	BSET k4, ST2_55	Yes	2	1	X
[8]	BSET k4, ST3_55	Yes	2	1†	X

† When instruction [8] 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.

Operands

k4, STx_55

Description

These instructions perform a bit manipulation in the A-unit ALU.

These instructions set to 1 a single bit, as defined by a 4-bit immediate value, k4, in the selected status register (ST0_55, ST1_55, ST2_55, or ST3_55).

Compatibility with C54x devices (C54CM = 1)

C55x DSP status bit mapping does not correspond to C54x DSP status bits.

Status Bits

Affected by none

Affects Selected status bits

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
BSET CARRY, ST0_55	The ST0_55 bit position defined by the label (CARRY, bit 11) is set to 1.

```
Before                  After
ST0_55          0000    ST0_55          0800
```

4.78.3 Status Bit Clear: BCLR f–name

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	BCLR f–name	Yes	2	1†	X

† When instruction [9] 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.

Operands

f–name

Description

This instruction performs a bit manipulation in the A-unit ALU.

This instruction clears to 0 a single bit, as defined by the one-bit-wide status bit field name, f–name, in the selected status register (ST0_55, ST1_55, ST2_55, or ST3_55).

Compatibility with C54x devices (C54CM = 1)

C55x DSP status bit mapping does not correspond to C54x DSP status bits.

Status Bits

Affected by none

Affects Selected status bits

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
BCLR AR1LC	The ST2_55 AR1LC (bit 2) is cleared to 0.

```
Before                  After
ST2_55          0006    ST2_55          0002
```

4.78.4 Status Bit Set: BSET f–name

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	BSET f–name	Yes	2	1†	X

† When instruction [10] 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.

Operands

f–name

Description

This instruction performs a bit manipulation in the A-unit ALU.

This instruction sets to 1 a single bit, as defined by the one-bit-wide status bit field name, f–name, in the selected status register (ST0_55, ST1_55, ST2_55, or ST3_55).

Compatibility with C54x devices (C54CM = 1)

C55x DSP status bit mapping does not correspond to C54x DSP status bits.

Status Bits

Affected by none

Affects Selected status bits

Repeat

This instruction cannot be repeated.

Example

Syntax	Description
BSET CARRY	The ST0_55 CARRY (bit 11) is set to 1.

```
Before                  After
ST0_55          0000    ST0_55          0800
```

4.79 Store Extended Auxiliary Register to Memory

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	MOV XAsrc, dbl(Lmem)	No	3	1	X

Operands Lmem, XAsrc

Description

This instruction moves the content of the 23-bit source register (XARx, XSP, XSSP, XDP, or XCDP) to the 32-bit data memory location addressed by data memory operand (Lmem). The upper 9 bits of the data memory are filled with 0.

Status Bits

Affected by none

Affects none

Repeat

This instruction can be repeated.

Example

Syntax	Description
MOV XAR1, dbl(*AR3)	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
AR3	0200	AR3	0200
200	3765	200	007F
201	0FD3	201	3492

4.80 Subtraction (SUB)

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SUB [src,] dst	Yes	2	1	X
[2]	SUB k4, dst	Yes	2	1	X
[3]	SUB K16, [src,] dst	No	4	1	X
[4]	SUB Smem, [src,] dst	No	3	1	X
[5]	SUB src, Smem, dst	No	3	1	X
[6]	SUB ACx << Tx, ACy	Yes	2	1	X
[7]	SUB ACx << #SHIFTW, ACy	Yes	3	1	X
[8]	SUB K16 << #16, [ACx,] ACy	No	4	1	X
[9]	SUB (K16 << #SHFT, [ACx,] ACy	No	4	1	X
[10]	SUB Smem << Tx, [ACx,] ACy	No	3	1	X
[11]	SUB Smem << #16, [ACx], ACy	No	3	1	X
[12]	SUB ACx, Smem << #16, ACy	No	3	1	X
[13]	SUB [uns(]Smem[)], BORROW, [ACx,] ACy	No	3	1	X
[14]	SUB [uns(]Smem[)], [ACx,] ACy	No	3	1	X
[15]	SUB [uns(]Smem[) << #SHIFTW, [ACx,] ACy	No	4	1	X
[16]	SUB dbl(Lmem), [ACx,] ACy	No	3	1	X
[17]	SUB ACx, dbl(Lmem) ACy	No	3	1	X
[18]	SUB Xmem, Ymem, ACx	No	3	1	X

Brief Description

These instructions perform a subtraction operation:

- ❑ In the D-unit ALU, if the destination operand is an accumulator (ACx).
- ❑ In the A-unit ALU, if the destination operand is an auxiliary or temporary register (TAx).
- ❑ In the D-unit shifter, if the instruction has a shift quantity other than the immediate 16 bit shift.

Status Bits

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

Affects ACOVx, ACOVy, CARRY

4.80.1 Subtraction: SUB [src,] dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[1]	SUB [src,] dst	Yes	2	1	X

Operands dst, src

Description

This instruction performs a subtraction operation between two registers contents.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - Input operands are sign extended to 40 bits according to SXMD.
  - 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 according to SXMD.
  - 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.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB AC1, AC0	The content of AC1 is subtracted from the content of AC0 and the result is stored in AC0.

4.80.2 Subtraction: SUB #4, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[2]	SUB k4, dst	Yes	2	1	X

Operands

dst, k4

Description

This instruction subtracts a 4-bit unsigned constant, k4, from a register content.

- 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.

Example

Syntax	Description
SUB #15, AC0	An unsigned 4-bit value is subtracted from the content of AC0 and the result is stored in AC0.

4.80.3 Subtraction: SUB K16, [src,] dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[3]	SUB K16, [src,] dst	No	4	1	X

Operands dst, K16, src

Description

This instruction subtracts a 16-bit signed constant, K16, from a register content.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - 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 according to SXMD.
  - The 16-bit constant, K16, is sign extended to 40 bits according to SXMD.
  - 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.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB #FFFFh, AC1, AC0	A signed 16-bit value is subtracted from the content of AC1 and the result is stored in AC0.

4.80.4 Subtraction: SUB Smem, [src,] dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[4]	SUB Smem, [src,] dst	No	3	1	X

Operands dst, Smem, src

Description

This instruction subtracts the content of a memory (Smem) location from a register content.

- When the destination operand (dst) is an accumulator:
  - The operation is performed on 40 bits in the D-unit ALU.
  - 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 according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - 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.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB *AR3, AC1, AC0	The content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

4.80.5 Subtraction: SUB src, Smem, dst

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[5]	SUB src, Smem, dst	No	3	1	X

Operands dst, 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.
  - 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 according to SXMD.
  - The content of the memory location is sign extended to 40 bits according to SXMD.
  - 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.
  - If an accumulator is the source operand (src) of the instruction, the 16 LSBs of the accumulator are used to perform the operation.
  - 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, SXMD

Affects ACOVx, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB AC1, *AR3, AC0	The content of AC1 is subtracted from the content addressed by AR3 and the result is stored in AC0.

4.80.6 Subtraction: SUB ACx << Tx, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[6]	SUB ACx << Tx, ACy	Yes	2	1	X

Operands ACx, ACy, Tx

Description

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 identical 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:

- 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB AC1 << T0, AC0	The content of AC1 shifted by the content of T0 is subtracted from the content of AC0 and the result is stored in AC0.

4.80.7 Subtraction: SUB ACx << #SHIFTW, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[7]	SUB ACx << #SHIFTW, ACy	Yes	3	1	X

Operands ACx, ACy, SHIFTW

Description

This instruction subtracts an accumulator content ACx shifted by the 6-bit value, SHIFTW, 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 identical 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, 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.

Example

Syntax	Description
SUB AC1 << #31, AC0	The content of AC1 shifted left by 31 bits is subtracted from the content of AC0 and the result is stored in AC0.

4.80.8 Subtraction: SUB K16 << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[8]	SUB K16 << #16, [ACx,] ACy	No	4	1	X

Operands

ACx, ACy, K16

Description

This instruction subtracts the 16-bit signed constant, K16, 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 identical 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, 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.

Example

Syntax	Description
SUB #FFFFh << #16, AC1, AC0	A signed 16-bit value shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

4.80.9 Subtraction: SUB K16 << #SHFT, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[9]	SUB K16 << #SHFT, [ACx,] ACy	No	4	1	X

Operands ACx, ACy, K16, SHFT

Description

This instruction subtracts the 16-bit signed constant, K16, shifted left by the 4-bit value, SHFT, from an accumulator content ACx.

- 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 identical 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, 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 M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB #9800h << #5, AC0, AC1	The signed 16-bit value (9800h) shifted left by 5 bits is subtracted from the content of AC0 and the result is stored in AC1.

4.80.10 Subtraction: SUB Smem << Tx, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[10]	SUB Smem << Tx, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Tx, Smem

Description

This instruction subtracts the content of a memory (Smem) location shifted by the content of Tx from an accumulator content ACx.

- 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 identical 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:

- 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

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB *AR3 << T0, AC1, AC0	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.

4.80.11 Subtraction: SUB Smem << #16, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[11]	SUB Smem << #16, [ACx,] ACy	No	3	1	X

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 identical 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.

Example

Syntax	Description
SUB *AR3 << #16, AC1, AC0	The content addressed by AR3 shifted left by 16 bits is subtracted from the content of AC1 and the result is stored in AC0.

4.80.12 Subtraction: SUB ACx, Smem << #16, ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[12]	SUB ACx, Smem << #16, ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction subtracts an accumulator content ACx from the content of a memory (Smem) location shifted left by 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 identical 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, 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.

Example

Syntax	Description
SUB AC1, *AR3 << #16, AC0	The content of AC1 is subtracted from the content addressed by AR3 shifted left by 16 bits and the result is stored in AC0.

4.80.13 Subtraction: SUB [uns(]Smem[)], BORROW, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[13]	SUB [uns(]Smem[)], BORROW, [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

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 40 bits in the D-unit ALU.
- Input operands are sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Status Bits

Affected by CARRY, M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB uns(*AR1), BORROW, AC0, AC1	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		After	
AC0	00 EC00 0000	AC0	00 EC00 0000
AC1	00 0000 0000	AC1	00 EBFF 0FFF
AR1	0302	AR1	0302
302	F000	302	F000
CARRY	0	CARRY	1

4.80.14 Subtraction: SUB [uns(]Smem[)], [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[14]	SUB [uns(]Smem[)], [ACx,] ACy	No	3	1	X

Operands ACx, ACy, Smem

Description

This instruction subtracts the content of a memory (Smem) location 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 content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- 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.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB uns(*AR3), AC1, AC0	The unsigned content addressed by AR3 is subtracted from the content of AC1 and the result is stored in AC0.

4.80.15 Subtraction: SUB [uns()Smem[])] << #SHIFTW, [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[15]	SUB [uns()Smem[])] << #SHIFTW, [ACx,] ACy	No	4	1	X

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 sign extended to 40 bits according to SXMD.
  - The content of the memory location is zero extended to 40 bits, if the optional uns keyword is applied to the input operand.
  - The content of the memory location is sign extended to 40 bits according to SXMD, if the optional uns keyword is not applied to the input operand.
- The shift operation is identical 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, 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 cannot be repeated.

Example

Syntax	Description
SUB uns(*AR3) << #31, AC1, AC0	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.

4.80.16 Subtraction: SUB dbl(Lmem), [ACx,] ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[16]	SUB dbl(Lmem), [ACx,] ACy	No	3	1	X

Operands

ACx, ACy, Lmem

Description

This instruction subtracts the content of data memory operand dbl(Lmem) from an accumulator content ACx.

- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least 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 SXMD.
- 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.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB dbl(*AR3+), AC1, AC0	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.

4.80.17 Subtraction: SUB ACx, dbl(Lmem), ACy

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[17]	SUB ACx, dbl(Lmem), ACy	No	3	1	X

Operands ACx, ACy, Lmem

Description

This instruction subtracts an accumulator content ACx from the content of data memory operand dbl(Lmem).

- The data memory operand dbl(Lmem) addresses are aligned:
  - if Lmem address is even: most significant word = Lmem, least significant word = Lmem + 1
  - if Lmem address is odd: most significant word = Lmem, least 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 SXMD.
- 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.

Status Bits

Affected by M40, SATD, SXMD

Affects ACOVy, CARRY

Repeat

This instruction can be repeated.

Example

Syntax	Description
SUB AC1, dbl(*AR3), AC0	The content of AC1 is subtracted from the content (long word) addressed by AR3 and AR3 + 1 and the result is stored in AC0.

4.80.18 Subtraction: SUB Xmem, Ymem, ACx

Syntax Characteristics

No.	Syntax	Parallel Enable Bit	Size	Cycles	Pipeline
[18]	SUB Xmem, Ymem, ACx	No	3	1	X

Operands ACx, Xmem, Ymem

Description

This instruction subtracts the content of data memory operand Ymem, shifted left 16 bits, from the content of data memory operand Xmem, 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 identical 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, 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

This instruction can be repeated.

Example

Syntax	Description
SUB *AR3, *AR4, AC0	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.

Chapter 5

Instruction Opcodes in Sequential Order

This chapter provides the opcode in sequential order for each instruction syntax.

Topic **Page**

5.1 Instruction Set Opcodes

Table 5–1 lists the opcodes of the instruction set.

Table 5–1. Instruction Set Opcodes

Opcode	Mnemonic syntax
0000000E XDDDX000	POPBOTH xdst
0000000E XSSSX000	PSHBOTH xsrc
0000000E xCCCCCCC kkkkkkkk	RPTCC k8, cond
0000001E xCCCCCCC xxxxxxxx	RETCC cond
0000010E xCCCCCCC LLLLLLLL	BCC L8, cond
0000011E LLLLLLLL LLLLLLLL	B L16
0000100E LLLLLLLL LLLLLLLL	CALL L16
0000101E GGGGGGGG Glllllll	
0000110E kkkkkkkk kkkkkkkk	RPT k16
0000111E llllllll llllllll	RPTB pmad
0001000E DDSS0000 xxSHIFTW	AND ACx << #SHIFTW[, ACy]
0001000E DDSS0001 xxSHIFTW	OR ACx << #SHIFTW[, ACy]
0001000E DDSS0010 xxSHIFTW	XOR ACx << #SHIFTW[, ACy]
0001000E DDSS0011 xxSHIFTW	ADD ACx << #SHIFTW, ACy
0001000E DDSS0100 xxSHIFTW	SUB ACx << #SHIFTW, ACy
0001000E DDSS0101 xxSHIFTW	SFTS ACx, #SHIFTW[, ACy]
0001000E DDSS0110 xxSHIFTW	SFTSC ACx, #SHIFTW[, ACy]
0001000E DDSS0111 xxSHIFTW	SFTL ACx, #SHIFTW[, ACy]
0001000E xxSS1000 xxddxxxx	EXP ACx, Tx
0001000E DDSS1001 xxddxxxx	MANT ACx, ACy :: NEXP ACx, Tx
0001000E xxSS1010 SSddxxxt	BCNT ACx, ACy,TCx, Tx
0001000E DDSS1100 SSDDnnnn	MAXDIFF ACx, ACy, ACz, ACw
0001000E DDSS1101 SSDDxxxr	DMAXDIFF ACx, ACy, ACz, ACw, TRNx
0001000E DDSS1110 SSDDxxxx	MINDIFF ACx, ACy, ACz, ACw
0001000E DDSS1111 SSDDxxxr	DMINDIFF ACx, ACy, ACz, ACw, TRNx
0001001E FSSScc00 FDDDxuxt	CMP[U] src RELOP dst, TCx
0001001E FSSScc01 FDDD0utt	CMPAND[U] src RELOP dst, TCy, TCx
0001001E FSSScc01 FDDD1utt	CMPAND[U] src RELOP dst, !TCy, TCx
0001001E FSSScc10 FDDD0utt	CMPOR[U] src RELOP dst, TCy, TCx
0001001E FSSScc10 FDDD1utt	CMPOR[U] src RELOP dst, !TCy, TCx

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
0001001E FSSSxx11 FDDD0xvv	ROL BitOut, src, BitIn, dst
0001001E FSSSxx11 FDDD1xvv	ROR BitIn, src, BitOut, dst
0001010E FSSSxxxx FDDD0000	AADD TAx, TAy
0001010E FSSSxxxx FDDD0001	AMOV TAx, TAy
0001010E FSSSxxxx FDDD0010	ASUB TAx, TAy
0001010E PPPPPPPP FDDD0100	AADD k8, TAx
0001010E PPPPPPPP FDDD0101	AMOV k8, TAx
0001010E PPPPPPPP FDDD0110	ASUB k8, TAx
0001010E FSSSxxxx FDDD1000	AADD TAx, TAy
0001010E FSSSxxxx FDDD1001	AMOV TAx, TAy
0001010E FSSSxxxx FDDD1010	ASUB TAx, TAy
0001010E PPPPPPPP FDDD1100	AADD k8, TAx
0001010E PPPPPPPP FDDD1101	AMOV k8, TAx
0001010E PPPPPPPP FDDD1110	ASUB k8, TAx
0001011E xxxxxkkk kkkk0000	MOV k7, DPH
0001011E xxxkkkkk kkkk0011	MOV k9, PDP
0001011E kkkkkkkk kkkk0100	MOV k12, BK03
0001011E kkkkkkkk kkkk0101	MOV k12, BK47
0001011E kkkkkkkk kkkk0110	MOV k12, BKC
0001011E kkkkkkkk kkkk1000	MOV k12, CSR
0001011E kkkkkkkk kkkk1001	MOV k12, BRC0
0001011E kkkkkkkk kkkk1010	MOV k12, BRC1
0001011E xxxxxxxk kkkk11xx	
0001100E kkkkkkkk FDDDFSSS	AND k8, src, dst
0001101E kkkkkkkk FDDDFSSS	OR k8, src, dst
0001110E kkkkkkkk FDDDFSSS	XOR k8, src, dst
0001111E KKKKKKKK SSDDxx0%	MPYK[R] K8, [ACx,] ACy
0001111E KKKKKKKK SSDDss1%	MACK[R] Tx, K8, [ACx,] ACy
0010000E	NOP
0010001E FSSSFDDD	MOV src, dst
0010010E FSSSFDDD	ADD [src,] dst
0010011E FSSSFDDD	SUB [src,] dst
0010100E FSSSFDDD	AND src, dst

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
0010101E FSSSFDDD	OR src, dst
0010110E FSSSFDDD	XOR src, dst
0010111E FSSSFDDD	MAX [src,] dst
0011000E FSSSFDDD	MIN [src,] dst
0011001E FSSSFDDD	ABS [src,] dst
0011010E FSSSFDDD	NEG [src,] dst
0011011E FSSSFDDD	NOT [src,] dst
0011100E FSSSFDDD (Note: FSSS = src1, FDDD = src2)	PSH src1, src2
0011101E FSSSFDDD (Note: FSSS = dst1, FDDD = dst2)	POP dst1, dst2
0011110E kkkkFDDD	MOV k4, dst
0011111E kkkkFDDD	MOV –k4, dst
0100000E kkkkFDDD	ADD k4, dst
0100001E kkkkFDDD	SUB k4, dst
0100010E 00SSFDDD	MOV HI(ACx), TAx
0100010E 01x0FDDD	SFTS dst, #–1
0100010E 01x1FDDD	SFTS dst, #1
0100010E 1000FDDD	MOV SP, TAx
0100010E 1001FDDD	MOV SSP, TAx
0100010E 1010FDDD	MOV CDP, TAx
0100010E 1100FDDD	MOV BRC0, TAx
0100010E 1101FDDD	MOV BRC1, TAx
0100010E 1110FDDD	MOV RPTC, TAx
0100011E kkkk0000	BCLR k4, ST0_55
0100011E kkkk0001	BSET k4, ST0_55
0100011E kkkk0010	BCLR k4, ST1_55
0100011E kkkk0011	BSET k4, ST1_55
0100011E kkkk0100	BCLR k4, ST2_55
0100011E kkkk0101	BSET k4, ST2_55
0100011E kkkk0110	BCLR k4, ST3_55
0100011E kkkk0111	BSET k4, ST3_55
0100011E xxxx1000	
0100011E xxxx1001	

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
0100011E xxxx1010	
0100011E xxxx1100	
0100100E xxxxx000	RPT CSR
0100100E FSSSx001	RPTADD CSR, TAx
0100100E kkkkx010	RPTADD CSR, k4
0100100E kkkkx011	RPTSUB CSR, k4
0100100E xxxxx100	RET
0100100E xxxxx101	RETI
0100101E 0LLLLLLL	B L7
0100101E 1lllllll	RPTBLOCAL pmad
0100110E kkkkkkkk	RPT k8
0100111E KKKKKKKK	AADD K8,SP
0101000E FDDDx000	SFTL dst, #1
0101000E FDDDx001	SFTL dst, #–1
0101000E FDDDx010	POP dst
0101000E xxDDx011	POP dbl(ACx)
0101000E FSSSx110	PSH src
0101000E xxSSx111	PSH dbl(ACx)
0101001E FSSS00DD	MOV TAx, HI(ACx)
0101001E FSSS1000	MOV TAx, SP
0101001E FSSS1001	MOV TAx, SSP
0101001E FSSS1010	MOV TAx, CDP
0101001E FSSS1100	MOV TAx, CSR
0101001E FSSS1101	MOV TAx, BRC1
0101001E FSSS1110	MOV TAx, BRC0
0101010E DDSS000%	ADD[R]V [ACx,] ACy
0101010E DDSS001%	SQA[R] [ACx,] ACy
0101010E DDSS010%	SQS[R] [ACx,] ACy
0101010E DDSS011%	MPY[R] [ACx,] ACy
0101010E DDSS100%	SQR[R] [ACx,] ACy
0101010E DDSS101%	RND [ACx,] ACy
0101010E DDSS110%	SAT[R] [ACx,] ACy
0101011E DDSSss0%	MAC[R] ACx, Tx, ACy[, ACy]

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
0101011E DDSSss1%	MAS[R] Tx, [ACx,] ACy
0101100E DDSSss0%	MPY[R] Tx, [ACx,] ACy
0101100E DDSSss1%	MAC[R] ACy, Tx, ACx, ACy
0101101E DDSSss00	ADD ACx << Tx, ACy
0101101E DDSSss01	SUB ACx << Tx, ACy
0101101E DDxxxx1t	SFTCC ACx, TCx
0101110E DDSSss00	SFTL ACx, Tx[, ACy]
0101110E DDSSss01	SFTS ACx, Tx[, ACy]
0101110E DDSSss10	SFTSC ACx, Tx[, ACy]
0101111E 00kkkkkk	SWAP ( )
01100lll lCCCCCCC	BCC l4, cond
01101000 xCCCCCCC PPPPPPPP PPPPPPPP PPPPPPPP	BCC P24, cond
01101001 xCCCCCCC PPPPPPPP PPPPPPPP PPPPPPPP	CALLCC P24, cond
01101010 PPPPPPPP PPPPPPPP PPPPPPPP	B P24
01101100 PPPPPPPP PPPPPPPP PPPPPPPP	CALL P24
01101101 xCCCCCCC LLLLLLLL LLLLLLLL	BCC L16, cond
01101110 xCCCCCCC LLLLLLLL LLLLLLLL	CALLCC L16, cond
01101111 FSSSccxu KKKKKKKK LLLLLLLL	BCC[U] L8, src RELOP K8
01110000 KKKKKKKK KKKKKKKK SSDDSHFT	ADD K16 << #SHFT, [ACx,] ACy
01110001 KKKKKKKK KKKKKKKK SSDDSHFT	SUB K16 << #SHFT, [ACx,] ACy
01110010 kkkkkkkk kkkkkkkk SSDDSHFT	AND k16 << #SHFT, [ACx,] ACy
01110011 kkkkkkkk kkkkkkkk SSDDSHFT	OR k16 << #SHFT, [ACx,] ACy
01110100 kkkkkkkk kkkkkkkk SSDDSHFT	XOR k16 << #SHFT, [ACx,] ACy
01110101 KKKKKKKK KKKKKKKK xxDDSHFT	MOV K16 << #SHFT, ACx
01110110 kkkkkkkk kkkkkkkk FDDD00SS	BFXTR k16, ACx, dst
01110110 kkkkkkkk kkkkkkkk FDDD01SS	BFXPA k16, ACx, dst
01110110 KKKKKKKK KKKKKKKK FDDD10xx	MOV K16, dst
01110111 DDDDDDDD DDDDDDDD FDDDxxxx	AMOV D16, TAx
01111000 kkkkkkkk kkkkkkkk xxx0000x	MOV k16, DP
01111000 kkkkkkkk kkkkkkkk xxx0001x	MOV k16, SSP
01111000 kkkkkkkk kkkkkkkk xxx0010x	MOV k16, CDP
01111000 kkkkkkkk kkkkkkkk xxx0011x	MOV k16, BSA01

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
01111000 kkkkkkkk kkkkkkkk xxx0100x	MOV k16, BSA23
01111000 kkkkkkkk kkkkkkkk xxx0101x	MOV k16, BSA45
01111000 kkkkkkkk kkkkkkkk xxx0110x	MOV k16, BSA67
01111000 kkkkkkkk kkkkkkkk xxx0111x	MOV k16, BSAC
01111000 kkkkkkkk kkkkkkkk xxx1000x	MOV k16, SP
01111001 KKKKKKKK KKKKKKKK SSDDxx0%	MPYK[R] K16, [ACx,] ACy
01111001 KKKKKKKK KKKKKKKK SSDDss1%	MACK[R] Tx, K16, [ACx,] ACy
01111010 KKKKKKKK KKKKKKKK SSDD000x	ADD K16 << #16, [ACx,] ACy
01111010 KKKKKKKK KKKKKKKK SSDD001x	SUB K16 << #16, [ACx,] ACy
01111010 kkkkkkkk kkkkkkkk SSDD010x	AND k16 << #16, [ACx,] ACy
01111010 kkkkkkkk kkkkkkkk SSDD011x	OR k16 << #16, [ACx,] ACy
01111010 kkkkkkkk kkkkkkkk SSDD100x	XOR k16 << #16, [ACx,] ACy
01111010 KKKKKKKK KKKKKKKK xxDD101x	MOV K16 << #16, ACx
01111010 xxxxxxxx xxxxxxxx xxxx110x	IDLE
01111011 KKKKKKKK KKKKKKKK FDDDFSSS	ADD K16, [src,] dst
01111100 KKKKKKKK KKKKKKKK FDDDFSSS	SUB K16, [src,] dst
01111101 kkkkkkkk kkkkkkkk FDDDFSSS	AND k16, src, dst
01111110 kkkkkkkk kkkkkkkk FDDDFSSS	OR k16, src, dst
01111111 kkkkkkkk kkkkkkkk FDDDFSSS	XOR k16, src, dst
10000000 XXXMMMYY YMMM00xx	MOV dbl(Xmem), dbl(Ymem)
10000000 XXXMMMYY YMMM01xx	MOV Xmem, Ymem
10000000 XXXMMMYY YMMM10SS	MOV ACx, Xmem, Ymem
10000001 XXXMMMYY YMMM00DD	ADD Xmem, Ymem, ACx
10000001 XXXMMMYY YMMM01DD	SUB Xmem, Ymem, ACx
10000001 XXXMMMYY YMMM10DD	MOV Xmem, Ymem, ACx
10000010 XXXMMMYY YMMM00mm uuDDDDg%	MPY[R][40] [uns()Xmem[])], [uns()Cmem[])], ACx :: MPY[R][40] [uns()Ymem[])], [uns()Cmem[])], ACy
10000010 XXXMMMYY YMMM01mm uuDDDDg%	MAC[R][40] [uns()Xmem[])], [uns()Cmem[])], ACx :: MPY[R][40] [uns()Ymem[])], [uns()Cmem[])], ACy
10000010 XXXMMMYY YMMM10mm uuDDDDg%	MAS[R][40] [uns()Xmem[])], [uns()Cmem[])], ACx :: MPY[R][40] [uns()Ymem[])], [uns()Cmem[])], ACy
10000010 XXXMMMYY YMMM11mm uuxxDDg%	AMAR Xmem :: MPY[R][40] [uns()Ymem[])], [uns()Cmem[])], ACx
10000011 XXXMMMYY YMMM00mm uuDDDDg%	MAC[R][40] [uns()Xmem[])], [uns()Cmem[])], ACx :: MAC[R][40] [uns()Ymem[])], [uns()Cmem[])], ACy

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
10000011 XXXMMMYY YMMM01mm uuDDDDg%	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
10000011 XXXMMMYY YMMM10mm uuDDDDg%	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
10000011 XXXMMMYY YMMM11mm uuxxDDg%	AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx
10000100 XXXMMMYY YMMM00mm uuDDDDg%	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16
10000100 XXXMMMYY YMMM01mm uuxxDDg%	AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx >> #16
10000100 XXXMMMYY YMMM10mm uuDDDDg%	MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16
10000100 XXXMMMYY YMMM11mm uuDDDDg%	MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16
10000101 XXXMMMYY YMMM00mm uuxxDDg%	AMAR Xmem :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx
10000101 XXXMMMYY YMMM01mm uuDDDDg%	MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy
10000101 XXXMMMYY YMMM10mm xxxxxxxx	AMAR Xmem, Ymem, Cmem
10000101 XXXMMMYY YMMM11mm DDx0DDU%	FIRSADD Xmem, Ymem, Cmem, ACx, ACy
10000101 XXXMMMYY YMMM11mm DDx1DDU%	FIRSSUB Xmem, Ymem, Cmem, ACx, ACy
10000110 XXXMMMYY YMMMxxDD 000guuU%	MPYM[R][40] [T3 =][uns()]Xmem[)], [uns()]Ymem[)], ACx
10000110 XXXMMMYY YMMMSSDD 001guuU%	MACM[R][40] [T3 =][uns()]Xmem[)], [uns()]Ymem[)], [ACx,] ACy
10000110 XXXMMMYY YMMMSSDD 010guuU%	MACM[R][40] [T3 =][uns()]Xmem[)], [uns()]Ymem[)], ACx >> #16[, ACy]
10000110 XXXMMMYY YMMMSSDD 011guuU%	MASM[R][40] [T3 =][uns()]Xmem[)], [uns()]Ymem[)], [ACx,] ACy
10000110 XXXMMMYY YMMMDDDD 100xssU%	MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy
10000110 XXXMMMYY YMMMDDDD 101xssU%	MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy
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%	MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
10000111 XXXMMMYY YMMMSSDD 001xssU%	MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
10000111 XXXMMMYY YMMMSSDD 010xssU%	MASM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem
10000111 XXXMMMYY YMMMSSDD 100xxxxx	ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem
10000111 XXXMMMYY YMMMSSDD 101xxxxx	SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem
10000111 XXXMMMYY YMMMSSDD 110xxxxx	MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem
10001xxx	
10010001 xxxxxxSS	B ACx
10010010 xxxxxxSS	CALL ACx
10010100 xxxxxxxx	RESET
10010101 0xxkkkkk	INTR k5
10010101 1xxkkkkk	TRAP k5
10010110 0CCCCCCC	XCC [label,]cond
10010110 1CCCCCCC	XCCPART [label,]cond
10010111	
10011000	mmap
10011001	port(Smem)
10011010	port(Smem)
10011100	<instruction>.LR
10011101	<instruction>.CR
10011110 0CCCCCCC	XCC [label,]cond
10011110 1CCCCCCC	XCCPART [label,]cond
10011111 0CCCCCCC	XCC [label,]cond
10011111 1CCCCCCC	XCCPART [label,]cond
1010FDDD AAAAAAAI	MOV Smem, dst
101100DD AAAAAAAI	MOV Smem << #16, ACx
10110100 AAAAAAAI	AMAR Smem
10110101 AAAAAAAI	PSH Smem
10110110 AAAAAAAI	DELAY Smem
10110111 AAAAAAAI	PSH dbl(Lmem)
10111000 AAAAAAAI	POP dbl(Lmem)

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
10111011 AAAAAAAI	POP Smem
101111SS AAAAAAAI	MOV HI(ACx), Smem
1100FSSS AAAAAAAI	MOV src, Smem
11010000 AAAAAAAI U%DDxxmm	MACM[R]Z [T3 =]Smem, Cmem, ACx
11010001 AAAAAAAI U%DD00mm	MPYM[R] [T3 =]Smem, Cmem, ACx
11010001 AAAAAAAI U%DD01mm	MACM[R] [T3 =]Smem, Cmem, ACx
11010001 AAAAAAAI U%DD10mm	MASM[R] [T3 =]Smem, Cmem, ACx
11010010 AAAAAAAI U%DD00SS	MACM[R] [T3 =]Smem, [ACx,] ACy
11010010 AAAAAAAI U%DD01SS	MASM[R] [T3 =]Smem, [ACx,] ACy
11010010 AAAAAAAI U%DD10SS	SQAM[R] [T3 =]Smem, [ACx,] ACy
11010010 AAAAAAAI U%DD11SS	SQSM[R] [T3 =]Smem, [ACx,] ACy
11010011 AAAAAAAI U%DD00SS	MPYM[R] [T3 =]Smem, [ACx,] ACy
11010011 AAAAAAAI U%DD10xx	SQRM[R] [T3 =]Smem, ACx
11010011 AAAAAAAI U%DDu1ss	MPYM[R][U] [T3 =]Smem, Tx, ACx
11010100 AAAAAAAI U%DDssSS	MACM[R] [T3 =]Smem, Tx, [ACx,] ACy
11010101 AAAAAAAI U%DDssSS	MASM[R] [T3 =]Smem, Tx, [ACx,] ACy
11010110 AAAAAAAI FDDDFSSS	ADD Smem, [src,] dst
11010111 AAAAAAAI FDDDFSSS	SUB Smem, [src,] dst
11011000 AAAAAAAI FDDDFSSS	SUB src, Smem, dst
11011001 AAAAAAAI FDDDFSSS	AND Smem, src, dst
11011010 AAAAAAAI FDDDFSSS	OR Smem, src, dst
11011011 AAAAAAAI FDDDFSSS	XOR Smem, src, dst
11011100 AAAAAAAI kkkkxx00	BTST k4, Smem, TC1
11011100 AAAAAAAI kkkkxx01	BTST k4, Smem, TC2
11011100 AAAAAAAI 0000xx10	MOV Smem, DP
11011100 AAAAAAAI 0001xx10	MOV Smem, CDP
11011100 AAAAAAAI 0010xx10	MOV Smem, BSA01
11011100 AAAAAAAI 0011xx10	MOV Smem, BSA23
11011100 AAAAAAAI 0100xx10	MOV Smem, BSA45
11011100 AAAAAAAI 0101xx10	MOV Smem, BSA67
11011100 AAAAAAAI 0110xx10	MOV Smem, BSAC
11011100 AAAAAAAI 0111xx10	MOV Smem, SP
11011100 AAAAAAAI 1000xx10	MOV Smem, SSP

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
11011100 AAAAAAAI 1001xx10	MOV Smem, BK03
11011100 AAAAAAAI 1010xx10	MOV Smem, BK47
11011100 AAAAAAAI 1011xx10	MOV Smem, BKC
11011100 AAAAAAAI 1100xx10	MOV Smem, DPH
11011100 AAAAAAAI 1111xx10	MOV Smem, PDP
11011100 AAAAAAAI x000xx11	MOV Smem, CSR
11011100 AAAAAAAI x001xx11	MOV Smem, BRC0
11011100 AAAAAAAI x010xx11	MOV Smem, BRC1
11011100 AAAAAAAI x011xx11	MOV Smem, TRN0
11011100 AAAAAAAI x100xx11	MOV Smem, TRN1
11011101 AAAAAAAI SSDDss00	ADD Smem << Tx, [ACx,] ACy
11011101 AAAAAAAI SSDDss01	SUB Smem << Tx, [ACx,] ACy
11011101 AAAAAAAI SSDDss10	ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy
11011101 AAAAAAAI x%DDss11	MOV [rnd(]Smem << Tx[)], ACx
11011110 AAAAAAAI SSDD0000	ADDSUBCC Smem, ACx, TC1, ACy
11011110 AAAAAAAI SSDD0001	ADDSUBCC Smem, ACx, TC2, ACy
11011110 AAAAAAAI SSDD0010	ADDSUBCC Smem, ACx, TC1, TC2, ACy
11011110 AAAAAAAI SSDD0011	SUBC Smem, [ACx,] ACy
11011110 AAAAAAAI SSDD0100	ADD Smem << #16, [ACx,] ACy
11011110 AAAAAAAI SSDD0101	SUB Smem << #16, [ACx,] ACy
11011110 AAAAAAAI SSDD0110	SUB ACx, Smem << #16, ACy
11011110 AAAAAAAI ssDD1000	ADDSUB Tx, Smem, ACx
11011110 AAAAAAAI ssDD1001	SUBADD Tx, Smem, ACx
11011111 AAAAAAAI FDDD000u	MOV [uns(]high_byte(Smem)[)], dst
11011111 AAAAAAAI FDDD001u	MOV [uns(]low_byte(Smem)[)], dst
11011111 AAAAAAAI xxDD010u	MOV [uns(]Smem[)], ACx
11011111 AAAAAAAI SSDD100u	ADD [uns(]Smem[)], CARRY, [ACx,] ACy
11011111 AAAAAAAI SSDD101u	SUB [uns(]Smem[)], BORROW, [ACx,] ACy
11011111 AAAAAAAI SSDD110u	ADD [uns(]Smem[)], [ACx,] ACy
11011111 AAAAAAAI SSDD111u	SUB [uns(]Smem[)], [ACx,] ACy
11100000 AAAAAAAI FSSSxxxt	BTST src, Smem, TCx
11100001 AAAAAAAI DDSHIFTW	MOV low_byte(Smem) << #SHIFTW, ACx
11100010 AAAAAAAI DDSHIFTW	MOV high_byte(Smem) << #SHIFTW, ACx

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
11100011 AAAAAAAI kkkk000x	BTSTSET k4, Smem, TC1
11100011 AAAAAAAI kkkk001x	BTSTSET k4, Smem, TC2
11100011 AAAAAAAI kkkk010x	BTSTCLR k4, Smem, TC1
11100011 AAAAAAAI kkkk011x	BTSTCLR k4, Smem, TC2
11100011 AAAAAAAI kkkk100x	BTSTNOT k4, Smem, TC1
11100011 AAAAAAAI kkkk101x	BTSTNOT k4, Smem, TC2
11100011 AAAAAAAI FSSS1100	BSET src, Smem
11100011 AAAAAAAI FSSS1101	BCLR src, Smem
11100011 AAAAAAAI FSSS111x	BNOT src, Smem
11100100 AAAAAAAI FSSSx0xx	PSH src,Smem
11100100 AAAAAAAI FDDDx1xx	POP dst, Smem
11100101 AAAAAAAI FSSS01x0	MOV src, high_byte(Smem)
11100101 AAAAAAAI FSSS01x1	MOV src, low_byte(Smem)
11100101 AAAAAAAI 000010xx	MOV DP, Smem
11100101 AAAAAAAI 000110xx	MOV CDP, Smem
11100101 AAAAAAAI 001010xx	MOV BSA01, Smem
11100101 AAAAAAAI 001110xx	MOV BSA23, Smem
11100101 AAAAAAAI 010010xx	MOV BSA45, Smem
11100101 AAAAAAAI 010110xx	MOV BSA67, Smem
11100101 AAAAAAAI 011010xx	MOV BSAC, Smem
11100101 AAAAAAAI 011110xx	MOV SP, Smem
11100101 AAAAAAAI 100010xx	MOV SSP, Smem
11100101 AAAAAAAI 100110xx	MOV BK03, Smem
11100101 AAAAAAAI 101010xx	MOV BK47, Smem
11100101 AAAAAAAI 101110xx	MOV BKC, Smem
11100101 AAAAAAAI 110010xx	MOV DPH, Smem
11100101 AAAAAAAI 111110xx	MOV PDP, Smem
11100101 AAAAAAAI x00011xx	MOV CSR, Smem
11100101 AAAAAAAI x00111xx	MOV BRC0, Smem
11100101 AAAAAAAI x01011xx	MOV BRC1, Smem
11100101 AAAAAAAI x01111xx	MOV TRN0, Smem
11100101 AAAAAAAI x10011xx	MOV TRN1, Smem
11100110 AAAAAAAI KKKKKKKK	MOV K8, Smem

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
11100111 AAAAAAAI SSss00xx	MOV ACx << Tx, Smem
11100111 AAAAAAAI SSss10x%	MOV [rnd()HI(ACx << Tx)[)], Smem
11100111 AAAAAAAI SSss11u%	MOV [uns() [rnd()HI(saturate(ACx << Tx))[))], Smem
11101000 AAAAAAAI SSxxx0x%	MOV [rnd()HI(ACx)[)], Smem
11101000 AAAAAAAI SSxxx1u%	MOV [uns() [rnd()HI(saturate(ACx))[))], Smem
11101001 AAAAAAAI SSSHIFTW	MOV ACx << #SHIFTW, Smem
11101010 AAAAAAAI SSSHIFTW	MOV HI(ACx << #SHIFTW), Smem
11101011 AAAAAAAI xxxx01xx	MOV RETA, dbl(Lmem)
11101011 AAAAAAAI xxSS10x0	MOV ACx, dbl(Lmem)
11101011 AAAAAAAI xxSS10u1	MOV [uns()saturate(ACx)[)], dbl(Lmem)
11101011 AAAAAAAI FSSS1100	MOV pair(TAx), dbl(Lmem)
11101011 AAAAAAAI xxSS1101	MOV ACx >> #1, dual(Lmem)
11101011 AAAAAAAI xxSS1110	MOV pair(HI(ACx)), dbl(Lmem)
11101011 AAAAAAAI xxSS1111	MOV pair(LO(ACx)), dbl(Lmem)
11101100 AAAAAAAI FSSS000x	BSET Baddr, src
11101100 AAAAAAAI FSSS001x	BCLR Baddr, src
11101100 AAAAAAAI FSSS010x	BTSTP Baddr, src
11101100 AAAAAAAI FSSS011x	BNOT Baddr, src
11101100 AAAAAAAI FSSS100t	BTST Baddr, src, TCx
11101101 AAAAAAAI SSDD000n	ADD dbl(Lmem), [ACx,] ACy
11101101 AAAAAAAI SSDD001n	SUB dbl(Lmem), [ACx,] ACy
11101101 AAAAAAAI SSDD010x	SUB ACx, dbl(Lmem), ACy
11101101 AAAAAAAI xxxx011x	MOV dbl(Lmem), RETA
11101101 AAAAAAAI xxDD100g	MOV[40] dbl(Lmem), ACx
11101101 AAAAAAAI xxDD101x	MOV dbl(Lmem), pair(HI(ACx))
11101101 AAAAAAAI xxDD110x	MOV dbl(Lmem), pair(LO(ACx))
11101101 AAAAAAAI FDDD111x	MOV dbl(Lmem), pair(TAx)
11101110 AAAAAAAI SSDD000x	ADD dual(Lmem), [ACx,] ACy
11101110 AAAAAAAI SSDD001x	SUB dual(Lmem), [ACx,] ACy
11101110 AAAAAAAI SSDD010x	SUB ACx, dual(Lmem), ACy
11101110 AAAAAAAI ssDD011x	SUB dual(Lmem), Tx, ACx
11101110 AAAAAAAI ssDD100x	ADD dual(Lmem), Tx, ACx
11101110 AAAAAAAI ssDD101x	SUB Tx, dual(Lmem), ACx

Table 5–1. Instruction Set Opcodes (Continued)

Opcode	Mnemonic syntax
11101110 AAAAAAAI ssDD110x	ADDSUB Tx, dual(Lmem), ACx
11101110 AAAAAAAI ssDD111x	SUBADD Tx, dual(Lmem), ACx
11101111 AAAAAAAI xxxx00mm	MOV Cmem, Smem
11101111 AAAAAAAI xxxx01mm	MOV Smem, Cmem
11101111 AAAAAAAI xxxx10mm	MOV Cmem,dbl(Lmem)
11101111 AAAAAAAI xxxx11mm	MOV dbl(Lmem), Cmem
11110000 AAAAAAAI KKKKKKKK KKKKKKKK	CMP Smem == K16, TC1
11110001 AAAAAAAI KKKKKKKK KKKKKKKK	CMP Smem == K16, TC2
11110010 AAAAAAAI kkkkkkkk kkkkkkkk	BAND Smem, k16, TC1
11110011 AAAAAAAI kkkkkkkk kkkkkkkk	BAND Smem, k16, TC2
11110100 AAAAAAAI kkkkkkkk kkkkkkkk	AND k16, Smem
11110101 AAAAAAAI kkkkkkkk kkkkkkkk	OR k16, Smem
11110110 AAAAAAAI kkkkkkkk kkkkkkkk	XOR k16, Smem
11110111 AAAAAAAI KKKKKKKK KKKKKKKK	ADD K16, Smem
11111000 AAAAAAAI KKKKKKKK xxDDx0U%	MPYMK[R] [T3 =]Smem, K8, ACx
11111000 AAAAAAAI KKKKKKKK SSDDx1U%	MACMK[R] [T3 =]Smem, K8, [ACx,] ACy
11111001 AAAAAAAI uxSHIFTW SSDD00xx	ADD [uns()Smem[])] << #SHIFTW, [ACx,] ACy
11111001 AAAAAAAI uxSHIFTW SSDD01xx	SUB [uns()Smem[])] << #SHIFTW, [ACx,] ACy
11111001 AAAAAAAI uxSHIFTW xxDD10xx	MOV [uns()Smem[])] << #SHIFTW, ACx
11111010 AAAAAAAI xxSHIFTW SSxxx0x%	MOV [rnd()HI(ACx << #SHIFTW)[])], Smem
11111010 AAAAAAAI uxSHIFTW SSxxx1x%	MOV [uns()(rnd()HI(saturate(ACx << #SHIFTW))[])], Smem
11111011 AAAAAAAI KKKKKKKK KKKKKKKK	MOV K16, Smem
11111100 AAAAAAAI LLLLLLLL LLLLLLLL	BCC L16, ARn_mod ! = #0

5.2 Instruction Set Opcode Symbols and Abbreviations

Table 5–2 lists the symbols and abbreviations used in the instruction set opcode.

Table 5–2. Instruction Set Opcode Symbols and Abbreviations

Bit Field Name	Bit Field Value	Bit Field Description
%	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 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	PPP1 0111	*ARn(T1), when ARMS = 0 *ARn(short(#3)), when ARMS = 1
	PPP1 1001	*+ARn, when ARMS = 0 *ARn(short(#4)), when ARMS = 1
	PPP1 1011	*–ARn, when ARMS = 0 *ARn(short(#5)), when ARMS = 1
	PPP1 1101	*(ARn + T0B), when ARMS = 0 *ARn(short(#6)), when ARMS = 1
	PPP1 1111	*(ARn – T0B), when ARMS = 0 *ARn(short(#7)), when ARMS = 1
	PPP encodes 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		Conditional field (cond) on source accumulator, auxiliary, or temporary register; TCx; 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(ACx) 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
	110 1000	TC1 & TC2

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
	110 1001	TC1 & !TC2
	110 1010	!TC1 & TC2
	110 1011	!TC1 & !TC2
	110 11xx	Reserved
	111 00SS	!overflow(ACx) 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
dd ss		Destination or 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)

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
DD SS		Destination or Source 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 5–2. 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 Register Content Swap 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	SWAPP AC0, AC2
	01 0001	Reserved
	01 0100	SWAPP T0, T2
	01 0101	Reserved
	01 1000	SWAPP AR0, AR2
	01 1001	Reserved
	01 1100	SWAPP AR4, T0
	01 1101	Reserved
	01 1110	SWAPP AR6, T2
	01 1111	Reserved
	10 1000	Reserved
	10 1100	SWAP4 AR4, T0
	11 1000	SWAP AR0, AR1
	11 1100	Reserved
	1x 0000	Reserved
	1x 0001	Reserved

Table 5–2. 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
KKK . . . K		Signed constant of n bits
lll . . . l		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 5–2. 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
SHIFTW		6-bit immediate shift value, –32 to +31
SHFT		4-bit immediate shift value, 0 to 15
tt	00	Bit 0: destination TCy bit of Register Comparison instruction
	01	Bit 1: source TCx bit of Register Comparison instruction
	10	When value = 0: TC1 is selected
	11	When value = 1: TC2 is selected
u	0	U or uns keyword is not applied; operand is considered signed
	1	U or 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

Table 5–2. Instruction Set Opcode Symbols and Abbreviations (Continued)

Bit Field Name	Bit Field Value	Bit Field Description
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).
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)

Chapter 6

Cross-Reference of Algebraic and Mnemonic Instruction Sets

This chapter provides a cross-reference between the mnemonic instruction set and the algebraic instruction set. For more information on the algebraic instruction set, see *TMS320C55x DSP Algebraic Instruction Set Reference Guide*, SPRU375.

Mnemonic Syntax	Algebraic Syntax
Absolute Distance	**Absolute Distance**
ABDST Xmem, Ymem, ACx, ACy	abdst(Xmem, Ymem, ACx, ACy)
Absolute Value	**Absolute Value**
ABS [src,] dst	dst = |src|
Accumulator, Auxiliary, or Temporary Register Content Swap	**Accumulator, Auxiliary, or Temporary Register Content Swap**
SWAP ARx, Tx	swap(ARx, Tx)
SWAP Tx, Ty	swap(Tx, Ty)
SWAP ARx, ARy	swap(ARx, ARy)
SWAP ACx, ACy	swap(ACx, ACy)
SWAPP ARx, Tx	swap(pair(ARx), pair(Tx))
SWAPP T0, T2	swap(pair(T0), pair(T2))
SWAPP AR0, AR2	swap(pair(AR0), pair(AR2))
SWAPP AC0, AC2	swap(pair(AC0), pair(AC2))
SWAP4 AR4, T0	swap(block(AR4), block(T0))

Mnemonic Syntax	Algebraic Syntax
Accumulator, Auxiliary, or Temporary Register Load	**Accumulator, Auxiliary, or Temporary Register Load**
MOV k4, dst	dst = k4
MOV –k4, dst	dst = –k4
MOV K16, dst	dst = K16
MOV Smem, dst	dst = Smem
MOV [uns()]high_byte(Smem)[)], dst	dst = uns(high_byte(Smem))
MOV [uns()]low_byte(Smem)[)], dst	dst = uns(low_byte(Smem))
MOV K16 << #16, ACx	ACx = K16 << #16
MOV K16 << #SHFT, ACx	ACx = K16 << #SHFT
MOV [rnd()]Smem << Tx[)], ACx	ACx = rnd(Smem << Tx)
MOV low_byte(Smem) << #SHIFTW, ACx	ACx = low_byte(Smem) << #SHIFTW
MOV high_byte(Smem) << #SHIFTW, ACx	ACx = high_byte(Smem) << #SHIFTW
MOV Smem << #16, ACx	ACx = Smem << #16
MOV [uns()]Smem[)], ACx	ACx = uns(Smem)
MOV [uns()]Smem[)] << #SHIFTW, ACx	ACx = uns(Smem) << #SHIFTW
MOV[40] dbl(Lmem), ACx	ACx = M40(dbl(Lmem))
MOV Xmem, Ymem, ACx	LO(ACx) = Xmem, HI(ACx) = Ymem
MOV dbl(Lmem), pair(HI(ACx))	pair(HI(ACx)) = Lmem
MOV dbl(Lmem), pair(LO(ACx))	pair(LO(ACx)) = Lmem
MOV dbl(Lmem), pair(TAx)	pair(TAx) = Lmem
Accumulator, Auxiliary, or Temporary Register Move	**Accumulator, Auxiliary, or Temporary Register Move**
MOV src, dst	dst = src
MOV HI(ACx), TAx	TAx = HI(ACx)
MOV TAx, HI(ACx)	HI(ACx) = TAx
Accumulator, Auxiliary, or Temporary Register Store	**Accumulator, Auxiliary, or Temporary Register Store**
MOV src, Smem	Smem = src
MOV src, high_byte(Smem)	high_byte(Smem) = src
MOV src, low_byte(Smem)	low_byte(Smem) = src
MOV HI(ACx), Smem	Smem = HI(ACx)

Mnemonic Syntax	Algebraic Syntax
MOV [rnd()]HI(ACx)[)], Smem	Smem = HI(rnd(ACx))
MOV ACx << Tx, Smem	Smem = LO(ACx << Tx)
MOV [rnd()]HI(ACx << Tx)[)], Smem	Smem = HI(rnd(ACx << Tx))
MOV ACx << #SHIFTW, Smem	Smem = LO(ACx << #SHIFTW)
MOV HI(ACx << #SHIFTW), Smem	Smem = HI(ACx << #SHIFTW)
MOV [rnd()]HI(ACx << #SHIFTW)[)], Smem	Smem = HI(rnd(ACx << #SHIFTW))
MOV [uns()] [rnd()]HI(saturate(ACx))[))], Smem	Smem = HI(saturate(uns(rnd(ACx))))
MOV [uns()] [rnd()]HI(saturate(ACx << Tx))[))], Smem	Smem = HI(saturate(uns(rnd(ACx << Tx))))
MOV [uns()(rnd()]HI(saturate(ACx << #SHIFTW))[))], Smem	Smem = HI(saturate(uns(rnd(ACx << #SHIFTW))))
MOV ACx, dbl(Lmem)	dbl(Lmem) = ACx
MOV [uns()]saturate(ACx)[)], dbl(Lmem)	dbl(Lmem) = saturate(uns(ACx))
MOV pair(HI(ACx)), dbl(Lmem)	Lmem = pair(HI(ACx))
MOV pair(LO(ACx)), dbl(Lmem)	Lmem = pair(LO(ACx))
MOV pair(TAx), dbl(Lmem)	Lmem = pair(TAx)
MOV ACx >> #1, dual(Lmem)	HI(Lmem) = HI(ACx) >> #1, LO(Lmem) = LO(ACx) >> #1
MOV ACx, Xmem, Ymem	Xmem = LO(ACx), Ymem = HI(ACx)
Addition	**Addition**
ADD [src,] dst	dst = dst + src
ADD k4, dst	dst = dst + k4
ADD K16, [src,] dst	dst = src + K16
ADD Smem, [src,] dst	dst = src + Smem
ADD ACx << Tx, ACy	ACy = ACy + (ACx << Tx)
ADD ACx << #SHIFTW, ACy	ACy = ACy + (ACx << #SHIFTW)
ADD K16 << #16, [ACx,] ACy	ACy = ACx + (K16 << #16)
ADD K16 << #SHFT, [ACx,] ACy	ACy = ACx + (K16 << #SHFT)
ADD Smem << Tx, [ACx,] ACy	ACy = ACx + (Smem << Tx)
ADD Smem << #16, [ACx,] ACy	ACy = ACx + (Smem << #16)
ADD [uns()]Smem[)], CARRY, [ACx,] ACy	ACy = ACx + uns(Smem) + CARRY
ADD [uns()]Smem[)], [ACx,] ACy	ACy = ACx + uns(Smem)
ADD [uns()]Smem[)] << #SHIFTW, [ACx,] ACy	ACy = ACx + (uns(Smem) << #SHIFTW)
ADD dbl(Lmem), [ACx,] ACy	ACy = ACx + dbl(Lmem)
ADD Xmem, Ymem, ACx	ACx = (Xmem << #16) + (Ymem << #16)

Mnemonic Syntax	Algebraic Syntax
ADD K16, Smem	Smem = Smem + K16
ADD[R]V [ACx,] ACy	ACy = rnd(ACy + |ACx|)
Bit Field Comparison	**Bit Field Comparison**
BAND Smem, k16, TCx	TCx = Smem & k16
Bit Field Counting	**Bit Field Counting**
BCNT ACx, ACy, TCx, Tx	Tx = count(ACx, ACy, TCx)
Bit Field Expand	**Bit Field Expand**
BFXPA k16, ACx, dst	dst = field_expand(ACx, k16)
Bit Field Extract	**Bit Field Extract**
BFXTR k16, ACx, dst	dst = field_extract(ACx, k16)
Bitwise Complement	**Bitwise Complement**
NOT [src,] dst	dst = ~src
Bitwise AND	**Bitwise AND**
AND src, dst	dst = dst & src
AND k8,src, dst	dst = src & k8
AND k16, src, dst	dst = src & k16
AND Smem, src, dst	dst = src & Smem
AND ACx << #SHIFTW[, ACy]	ACy = ACy & (ACx <<< #SHIFTW)
AND k16 << #16, [ACx,] ACy	ACy = ACx & (k16 <<< #16)
AND k16 << #SHFT, [ACx,] ACy	ACy = ACx & (k16 <<< #SHFT)
AND k16, Smem	Smem = Smem & k16
Bitwise OR	**Bitwise OR**
OR src, dst	dst = dst | src
OR k8, src, dst	dst = src | k8
OR k16, src, dst	dst = src | k16
OR Smem, src, dst	dst = src | Smem
OR ACx << #SHIFTW[, ACy]	ACy = ACy | (ACx <<< #SHIFTW)

Mnemonic Syntax	Algebraic Syntax
OR k16 << #16, [ACx,] ACy	ACy = ACx | (k16 <<< #16)
OR k16 << #SHFT, [ACx,] ACy	ACy = ACx | (k16 <<< #SHFT)
OR k16, Smem	Smem = Smem | k16
Bitwise XOR	**Bitwise XOR**
XOR src, dst	dst = dst ^ src
XOR k8, src, dst	dst = src ^ k8
XOR k16, src, dst	dst = src ^ k16
XOR Smem, src, dst	dst = src ^ Smem
XOR ACx << #SHIFTW[, ACy]	ACy = ACy ^ (ACx <<< #SHIFTW)
XOR k16 << #16, [ACx,] ACy	ACy = ACx ^ (k16 <<< #16)
XOR k16 << #SHFT, [ACx,] ACy	ACy = ACx ^ (k16 <<< #SHFT)
XOR k16, Smem	Smem = Smem ^ k16
Branch Conditionally	**Branch Conditionally**
BCC l4, cond	if (cond) goto l4
BCC L8, cond	if (cond) goto L8
BCC L16, cond	if (cond) goto L16
BCC P24, cond	if (cond) goto P24
Branch Unconditionally	**Branch Unconditionally**
B ACx	goto ACx
B L7	goto L7
B L16	goto L16
B P24	goto P24
Branch on Auxiliary Register Not Zero	**Branch on Auxiliary Register Not Zero**
BCC L16, ARn_mod != #0	if (ARn_mod != #0) goto L16
Call Conditionally	**Call Conditionally**
CALLCC L16, cond	if (cond) call L16
CALLCC P24, cond	if (cond) call P24

Mnemonic Syntax	Algebraic Syntax
Call Unconditionally	**Call Unconditionally**
CALL ACx	call ACx
CALL L16	call L16
CALL P24	call P24
Compare and Branch	**Compare and Branch**
BCC[U] L8, src RELOP K8	compare (uns(src RELOP K8)) goto L8
Compare and Select Extremum	**Compare and Select Extremum**
MAXDIFF ACx, ACy, ACz, ACw	max_diff(ACx, ACy, ACz, ACw)
DMAXDIFF ACx, ACy, ACz, ACw, TRNx	max_diff_dbl(ACx, ACy, ACz, ACw, TRNx)
MINDIFF ACx, ACy, ACz, ACw	min_diff(ACx, ACy, ACz, ACw)
DMINDIFF ACx, ACy, ACz, ACw, TRNx	min_diff_dbl(ACx, ACy, ACz, ACw, TRNx)
Conditional Addition/Subtraction	**Conditional Addition/Subtraction**
ADDSUBCC Smem, ACx, TCx, ACy	ACy = adsc(Smem, ACx, TCx)
ADDSUBCC Smem, ACx, TC1, TC2, ACy	ACy = adsc(Smem, ACx, TC1, TC2)
ADDSUB2CC Smem, ACx, Tx, TC1, TC2, ACy	ACy = ads2c(Smem, ACx, Tx, TC1, TC2)
Conditional Shift	**Conditional Shift**
SFTCC ACx, TCx	ACx = sftc(ACx, TCx)
Conditional Subtract	**Conditional Subtract**
SUBC Smem, [ACx,] ACy	subc(Smem, ACx, ACy)
Dual 16-Bit Arithmetic	**Dual 16-Bit Arithmetic**
ADDSUB Tx, Smem, ACx	HI(ACx) = Smem + Tx, LO(ACx) = Smem – Tx
SUBADD Tx, Smem, ACx	HI(ACx) = Smem – Tx, LO(ACx) = Smem + Tx
ADD dual(Lmem), [ACx,] ACy	HI(ACy) = HI(Lmem) + HI(ACx), LO(ACy) = LO(Lmem) + LO(ACx)
SUB dual(Lmem), [ACx,] ACy	HI(ACy) = HI(ACx) – HI(Lmem), LO(ACy) = LO(ACx) – LO(Lmem)
SUB ACx, dual(Lmem), ACy	HI(ACy) = HI(Lmem) – HI(ACx), LO(ACy) = LO(Lmem) – LO(ACx)

Mnemonic Syntax	Algebraic Syntax
SUB dual(Lmem), Tx, ACx	HI(ACx) = Tx – HI(Lmem), LO(ACx) = Tx – LO(Lmem)
ADD dual(Lmem), Tx, ACx	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) + Tx
SUB Tx, dual(Lmem), ACx	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) – Tx
ADDSUB Tx, dual(Lmem), ACx	HI(ACx) = HI(Lmem) + Tx, LO(ACx) = LO(Lmem) – Tx
SUBADD Tx, dual(Lmem), ACx	HI(ACx) = HI(Lmem) – Tx, LO(ACx) = LO(Lmem) + Tx
Dual Multiply (Accumulate/Subtract)	**Dual Multiply (Accumulate/Subtract)**
MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(uns(Ymem) * uns(Cmem)))
AMAR Xmem :: MPY[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	mar(Xmem), ACx = M40(rnd(uns(Ymem) * uns(Cmem)))
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	ACx = M40(rnd(ACx + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))
AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	mar(Xmem), ACx = M40(rnd(ACx + (uns(Ymem) * uns(Cmem))))
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd(ACy – (uns(Ymem) * uns(Cmem))))
AMAR Xmem :: MAS[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx	mar(Xmem), ACx = M40(rnd(ACx – (uns(Ymem) * uns(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(Cmem)))), ACy = M40(rnd(ACy + (uns(Ymem) * uns(Cmem))))
MPY[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	ACx = M40(rnd(uns(Xmem) * uns(Cmem))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))
MAC[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx >> #16 :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	ACx = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))

Mnemonic Syntax	Algebraic Syntax
MAS[R][40] [uns()]Xmem[)], [uns()]Cmem[)], ACx :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACy >> #16	ACx = M40(rnd(ACx – (uns(Xmem) * uns(Cmem)))), ACy = M40(rnd((ACy >> #16) + (uns(Ymem) * uns(Cmem))))
AMAR Xmem :: MAC[R][40] [uns()]Ymem[)], [uns()]Cmem[)], ACx >> #16	mar(Xmem), ACx = M40(rnd((ACx >> #16) + (uns(Ymem) * uns(Cmem))))
AMAR Xmem, Ymem, Cmem	mar(Xmem), mar(Ymem), mar(Cmem)
Execute Conditionally	**Execute Conditionally**
XCC [label,]cond	if (cond) execute(AD_Unit)
XCCPART [label,]cond	if (cond) execute(D_Unit)
Extended Auxiliary Register Move	**Extended Auxiliary Register Move**
MOV xsrc, xdst	xdst = xsrc
Finite Impulse Response Filter, Symmetrical/ Antisymmetrical	**Finite Impulse Response Filter, Symmetrical/ Antisymmetrical**
FIRSADD Xmem, Ymem, Cmem, ACx, ACy	firs(Xmem, Ymem, Cmem, ACx, ACy)
FIRSSUB Xmem, Ymem, Cmem, ACx, ACy	firsn(Xmem, Ymem, Cmem, ACx, ACy)
Idle	**Idle**
IDLE	idle
Implied Paralleled Instructions	**Implied Paralleled Instructions**
MPYM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	ACy = rnd(Tx * Xmem), Ymem = HI(ACx << T2) [,T3 = Xmem]
MACM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	ACy = rnd(ACy + (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]
MASM[R] [T3 =]Xmem, Tx, ACy :: MOV HI(ACx << T2), Ymem	ACy = rnd(ACy – (Tx * Xmem)), Ymem = HI(ACx << T2) [,T3 = Xmem]
ADD Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	ACy = ACx + (Xmem << #16), Ymem = HI(ACy << T2)
SUB Xmem << #16, ACx, ACy :: MOV HI(ACy << T2), Ymem	ACy = (Xmem << #16) – ACx, Ymem = HI(ACy << T2)
MOV Xmem << #16, ACy :: MOV HI(ACx << T2), Ymem	ACy = Xmem << #16, Ymem = HI(ACx << T2)
MACM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	ACx = rnd(ACx + (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]

Mnemonic Syntax	Algebraic Syntax
MASM[R] [T3 =]Xmem, Tx, ACx :: MOV Ymem << #16, ACy	ACx = rnd(ACx – (Tx * Xmem)), ACy = Ymem << #16 [,T3 = Xmem]
Least Mean Square (LMS)	**Least Mean Square (LMS)**
LMS Xmem, Ymem, ACx, ACy	lms(Xmem, Ymem, ACx, ACy)
Linear/Circular Addressing Qualifiers	**Linear/Circular Addressing Qualifiers**
<instruction>.LR	linear()
<instruction>.CR	circular()
Load Effective Address to Extended Auxiliary Register	**Load Effective Address to Extended Auxiliary Register**
AMAR Smem, XAdst	XAdst = mar(Smem)
AMOV k23, XAdst	XAdst = k23
Load Extended Auxiliary Register from Memory	**Load Extended Auxiliary Register from Memory**
MOV dbl(Lmem), XAdst	XAdst = dbl(Lmem)
Logical Shift	**Logical Shift**
SFTL dst, #1	dst = dst <<< #1
SFTL dst, #–1	dst = dst >>> #1
SFTL ACx, Tx[, ACy]	ACy = ACx <<< Tx
SFTL ACx, #SHIFTW[, ACy]	ACy = ACx <<< #SHIFTW
Maximum Comparison	**Maximum Comparison**
MAX [src,] dst	dst = max(src, dst)
Minimum Comparison	**Minimum Comparison**
MIN [src,] dst	dst = min(src, dst)
Memory-Mapped Register Access Qualifier	**Memory-Mapped Register Access Qualifier**
mmap	mmap()

Mnemonic Syntax	Algebraic Syntax
Memory Bit Test/Set/Clear/Complement	**Memory Bit Test/Set/Clear/Complement**
BTST src, Smem, TCx	TCx = bit(Smem, src)
BNOT src, Smem	cbit(Smem, src)
BCLR src, Smem	bit(Smem, src) = #0
BSET src, Smem	bit(Smem, src) = #1
BTSTSET k4, Smem, TCx	TCx = bit(Smem, k4), bit(Smem, k4) = #1
BTSTCLR k4, Smem, TCx	TCx = bit(Smem, k4), bit(Smem, k4) = #0
BTSTNOT k4, Smem, TCx	TCx = bit(Smem, k4), cbit(Smem, k4)
BTST k4, Smem, TCx	TCx = bit(Smem, k4)
Memory Comparison	**Memory Comparison**
CMP Smem == K16, TCx	TCx = (Smem == K16)
Memory Delay	**Memory Delay**
DELAY Smem	delay(Smem)
Memory-to-Memory Move/Memory Initialization	**Memory-to-Memory Move/Memory Initialization**
MOV Cmem, Smem	Smem = Cmem
MOV Smem, Cmem	Cmem = Smem
MOV K8, Smem	Smem = K8
MOV K16, Smem	Smem = K16
MOV Cmem, dbl(Lmem)	Lmem = dbl(Cmem)
MOV dbl(Lmem), Cmem	dbl(Cmem) = Lmem
MOV dbl(Xmem), dbl(Ymem)	dbl(Ymem) = dbl(Xmem)
MOV Xmem, Ymem	Ymem = Xmem
Modify Auxiliary Register (MAR)	**Modify Auxiliary Register (MAR)**
AADD TAx, TAy	mar(TAy + TAx)
ASUB TAx, TAy	mar(TAy – TAx)
AMOV TAx, TAy	mar(TAy = TAx)
AADD k8, TAx	mar(TAx + k8)
ASUB k8, TAx	mar(TAx – k8)

Mnemonic Syntax	Algebraic Syntax
AMOV k8, TAx	mar(TAx = k8)
AMOV D16, TAx	mar(TAx = D16)
AMAR Smem	mar(Smem)
Modify Data Stack Pointer (SP)	**Modify Data Stack Pointer (SP)**
AADD K8, SP	SP = SP + K8
Multiply (MPY)	**Multiply (MPY)**
SQR[R] [ACx,] ACy	ACy = rnd(ACx * ACx)
MPY[R] [ACx,] ACy	ACy = rnd(ACy * ACx)
MPY[R] Tx, [ACx,] ACy	ACy = rnd(ACx * Tx)
MPYK[R] K8, [ACx,] ACy	ACy = rnd(ACx * K8)
MPYK[R] K16, [ACx,] ACy	ACy = rnd(ACx * K16)
MPYM[R] [T3 =]Smem, Cmem, ACx	ACx = rnd(Smem * Cmem)[, T3 = Smem]
SQRM[R] [T3 =]Smem, ACx	ACx = rnd(Smem * Smem)[, T3 = Smem]
MPYM[R] [T3 =]Smem, [ACx,] ACy	ACy = rnd(Smem * ACx)[, T3 = Smem]
MPYMK[R] [T3 =]Smem, K8, ACx	ACx = rnd(Smem * K8)[, T3 = Smem]
MPYM[R][40] [T3 =][uns()]Xmem[)], [uns()]Ymem[)], ACx	ACx = M40(rnd(uns(Xmem) * uns(Ymem))) [, T3 = Xmem]
MPYM[R][U] [T3 =]Smem, Tx, ACx	ACx = rnd(uns(Tx * Smem))[, T3 = Smem]
Multiply and Accumulate (MAC)	**Multiply and Accumulate (MAC)**
SQA[R] [ACx,] ACy	ACy = rnd(ACy + (ACx * ACx))
MAC[R] ACx, Tx, ACy[, ACy]	ACy = rnd(ACy + (ACx * Tx))
MAC[R] ACy, Tx, ACx, ACy	ACy = rnd((ACy * Tx) + ACx)
MACK[R] Tx, K8, [ACx,] ACy	ACy = rnd(ACx + (Tx * K8))
MACK[R] Tx, K16, [ACx,] ACy	ACy = rnd(ACx + (Tx * K16))
MACM[R] [T3 =]Smem, Cmem, ACx	ACx = rnd(ACx + (Smem * Cmem))[, T3 = Smem]
MACM[R]Z [T3 =]Smem, Cmem, ACx	ACx = rnd(ACx + (Smem * Cmem))[, T3 = Smem], delay(Smem)
SQAM[R] [T3 =]Smem, [ACx,] ACy	ACy = rnd(ACx + (Smem * Smem))[, T3 = Smem]
MACM[R] [T3 =]Smem, [ACx,] ACy	ACy = rnd(ACy + (Smem * ACx))[, T3 = Smem]
MACM[R] [T3 =]Smem, Tx, [ACx,] ACy	ACy = rnd(ACx + (Tx * Smem))[, T3 = Smem]
MACMK[R] [T3 =]Smem, K8, [ACx,] ACy	ACy = rnd(ACx + (Smem * K8))[, T3 = Smem]

Mnemonic Syntax	Algebraic Syntax
MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	ACy = M40(rnd(ACx + (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]
MACM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], ACx >> #16[, ACy]	ACy = M40(rnd((ACx >> #16) + (uns(Xmem) * uns(Ymem))))[, T3 = Xmem]
Multiply and Subtract (MAS)	**Multiply and Subtract (MAS)**
SQS[R] [ACx,] ACy	ACy = rnd(ACy – (ACx * ACx))
MAS[R] Tx, [ACx,] ACy	ACy = rnd(ACy – (ACx * Tx))
MASM[R] [T3 =]Smem, Cmem, ACx	ACx = rnd(ACx – (Smem * Cmem))[, T3 = Smem]
SQSM[R] [T3 =]Smem, [ACx,] ACy	ACy = rnd(ACx – (Smem * Smem))[, T3 = Smem]
MASM[R] [T3 =]Smem, [ACx,] ACy	ACy = rnd(ACy – (Smem * ACx))[, T3 = Smem]
MASM[R] [T3 =]Smem, Tx, [ACx,] ACy	ACy = rnd(ACx – (Tx * Smem))[, T3 = Smem]
MASM[R][40] [T3 =][uns(]Xmem[)], [uns(]Ymem[)], [ACx,] ACy	ACy = M40(rnd(ACx – (uns(Xmem) * uns(Ymem)))) [, T3 = Xmem]
Negation	**Negation**
NEG [src,] dst	dst = –src
No Operation (NOP)	**No Operation (NOP)**
NOP	nop
NOP_16	nop_16
Normalization	**Normalization**
MANT ACx, ACy :: NEXP ACx, Tx	ACy = mant(ACx), Tx = –exp(ACx)
EXP ACx, Tx	Tx = exp(ACx)
Peripheral Port Register Access Qualifier	**Peripheral Port Register Access Qualifiers**
port(Smem)	readport()
port(Smem)	writeport()
Pop Extended Auxiliary Register from Stack Pointers	**Pop Extended Auxiliary Register from Stack Pointers**
POPBOTH xdst	xdst = popboth()

Mnemonic Syntax	Algebraic Syntax
Pop Top of Stack (TOS)	**Pop Top of Stack (TOS)**
POP dst1, dst2	dst1, dst2 = pop()
POP dst	dst = pop()
POP dst, Smem	dst, Smem = pop()
POP ACx	ACx = dbl(pop())
POP Smem	Smem = pop()
POP dbl(Lmem)	dbl(Lmem) = pop()
Push Extended Auxiliary Register to Stack Pointers	**Push Extended Auxiliary Register to Stack Pointers**
PSHBOTH xsrc	pshboth(xsrc)
Push to Top of Stack (TOS)	**Push to Top of Stack (TOS)**
PSH src1, src2	push(src1, src2)
PSH src	push(src)
PSH src, Smem	push(src, Smem)
PSH ACx	dbl(push(ACx))
PSH Smem	push(Smem)
PSH dbl(Lmem)	push(dbl(Lmem))
Register Bit Test/Set/Clear/Complement	**Register Bit Test/Set/Clear/Complement**
BTST Baddr, src, TCx	TCx = bit(src, Baddr)
BNOT Baddr, src	cbit(src, Baddr)
BCLR Baddr, src	bit(src, Baddr) = #0
BSET Baddr, src	bit(src, Baddr) = #1
BTSTP Baddr, src	bit(src, pair(Baddr))
Register Comparison	**Register Comparison**
CMP[U] src RELOP dst, TCx	TCx = uns(src RELOP dst)
CMPAND[U] src RELOP dst, TCy, TCx	TCx = TCy & uns(src RELOP dst)
CMPAND[U] src RELOP dst, !TCy, TCx	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	TCx = !TCy | uns(src RELOP dst)

Mnemonic Syntax	Algebraic Syntax
Repeat Block of Instructions Unconditionally	**Repeat Block of Instructions Unconditionally**
RPTBLOCAL pmad	localrepeat{}
RPTB pmad	blockrepeat{}
Repeat Single Instruction Conditionally	**Repeat Single Instruction Conditionally**
RPTCC k8, cond	while (cond && (RPTC < k8)) repeat
Repeat Single Instruction Unconditionally	**Repeat Single Instruction Unconditionally**
RPT CSR	repeat(CSR)
RPTADD CSR, TAx	repeat(CSR), CSR += TAx
RPT k8	repeat(k8)
RPTADD CSR, k4	repeat(CSR), CSR += k4
RPTSUB CSR, k4	repeat(CSR), CSR –= k4
RPT k16	repeat(k16)
Return Conditionally	**Return Conditionally**
RETCC cond	if (cond) return
Return Unconditionally	**Return Unconditionally**
RET	return
Return from Interrupt	**Return from Interrupt**
RETI	return_int
Rotate Left	**Rotate Left**
ROL BitOut, src, BitIn, dst	dst = BitOut \\ src \\ BitIn
Rotate Right	**Rotate Right**
ROR BitIn, src, BitOut, dst	dst = BitIn // src // BitOut
Round	**Round**
RND [ACx,] ACy	ACy = rnd(ACx)

Mnemonic Syntax	Algebraic Syntax
Saturate	**Saturate**
SAT[R] [ACx,] ACy	ACy = saturate(rnd(ACx))
Signed Shift	**Signed Shift**
SFTS dst, #–1	dst = dst >> #1
SFTS dst, #1	dst = dst << #1
SFTS ACx, Tx[, ACy]	ACy = ACx << Tx
SFTSC ACx, Tx[, ACy]	ACy = ACx <<C Tx
SFTS ACx, #SHIFTW[, ACy]	ACy = ACx << #SHIFTW
SFTSC ACx, #SHIFTW[, ACy]	ACy = ACx <<C #SHIFTW
Software Interrupt	**Software Interrupt**
INTR k5	intr(k5)
Software Reset	**Software Reset**
RESET	reset
Software Trap	**Software Trap**
TRAP k5	trap(k5)
Specific CPU Register Load	**Specific CPU Register Load**
MOV k12, BK03	BK03 = k12
MOV k12, BK47	BK47 = k12
MOV k12, BKC	BKC = k12
MOV k12, BRC0	BRC0 = k12
MOV k12, BRC1	BRC1 = k12
MOV k12, CSR	CSR = k12
MOV k7, DPH	MDP = k7
MOV k9, PDP	PDP = k9
MOV k16, BSA01	BOF01 = k16
MOV k16, BSA23	BOF23 = k16
MOV k16, BSA45	BOF45 = k16
MOV k16, BSA67	BOF67 = k16
MOV k16, BSAC	BOFC = k16

Mnemonic Syntax	Algebraic Syntax
MOV k16, CDP	CDP = k16
MOV k16, DP	DP = k16
MOV k16, SP	SP = k16
MOV k16, SSP	SSP = k16
MOV Smem, BK03	BK03 = Smem
MOV Smem, BK47	BK47 = Smem
MOV Smem, BKC	BKC = Smem
MOV Smem, BSA01	BOF01 = Smem
MOV Smem, BSA23	BOF23 = Smem
MOV Smem, BSA45	BOF45 = Smem
MOV Smem, BSA67	BOF67 = Smem
MOV Smem, BSAC	BOFC = Smem
MOV Smem, BRC0	BRC0 = Smem
MOV Smem, BRC1	BRC1 = Smem
MOV Smem, CDP	CDP = Smem
MOV Smem, CSR	CSR = Smem
MOV Smem, DP	DP = Smem
MOV Smem, DPH	MDP = Smem
MOV Smem, PDP	PDP = Smem
MOV Smem, SP	SP = Smem
MOV Smem, SSP	SSP = Smem
MOV Smem, TRN0	TRN0 = Smem
MOV Smem, TRN1	TRN1 = Smem
MOV dbl(Lmem), RETA	RETA = dbl(Lmem)
Specific CPU Register Move	**Specific CPU Register Move**
MOV TAx, BRC0	BRC0 = TAx
MOV TAx, BRC1	BRC1 = TAx
MOV TAx, CDP	CDP = TAx
MOV TAx, CSR	CSR = TAx
MOV TAx, SP	SP = TAx
MOV TAx, SSP	SSP = TAx
MOV BRC0, TAx	TAx = BRC0
MOV BRC1, TAx	TAx = BRC1

Mnemonic Syntax	Algebraic Syntax
MOV CDP, TAx	TAx = CDP
MOV RPTC, TAx	TAx = RPTC
MOV SP, TAx	TAx = SP
MOV SSP, TAx	TAx = SSP
Specific CPU Register Store	**Specific CPU Register Store**
MOV BK03, Smem	Smem = BK03
MOV BK47, Smem	Smem = BK47
MOV BKC, Smem	Smem = BKC
MOV BSA01, Smem	Smem = BOF01
MOV BSA23, Smem	Smem = BOF23
MOV BSA45, Smem	Smem = BOF45
MOV BSA67, Smem	Smem = BOF67
MOV BSAC, Smem	Smem = BOFC
MOV BRC0, Smem	Smem = BRC0
MOV BRC1, Smem	Smem = BRC1
MOV CDP, Smem	Smem = CDP
MOV CSR, Smem	Smem = CSR
MOV DP, Smem	Smem = DP
MOV DPH, Smem	Smem = MDP
MOV PDP, Smem	Smem = PDP
MOV SP, Smem	Smem = SP
MOV SSP, Smem	Smem = SSP
MOV TRN0, Smem	Smem = TRN0
MOV TRN1, Smem	Smem = TRN1
MOV RETA, dbl(Lmem)	dbl(Lmem) = RETA
Square Distance	**Square Distance**
SQDST Xmem, Ymem, ACx, ACy	sqdst(Xmem, Ymem, ACx, ACy)
Status Bit Set/Clear	**Status Bit Set/Clear**
BCLR k4, STx_55	bit(STx, k4) = #0
BSET k4, STx_55	bit(STx, k4) = #1

Mnemonic Syntax	Algebraic Syntax
Store Extended Auxiliary Register to Memory	**Store Extended Auxiliary Register to Memory**
MOV XAsrc, dbl(Lmem)	dbl(Lmem) = XAsrc
Subtraction	**Subtraction**
SUB [src,] dst	dst = dst – src
SUB k4, dst	dst = dst – k4
SUB K16, [src,] dst	dst = src – K16
SUB Smem, [src,] dst	dst = src – Smem
SUB src, Smem, dst	dst = Smem – src
SUB ACx << Tx, ACy	ACy = ACy – (ACx << Tx)
SUB ACx << #SHIFTW, ACy	ACy = ACy – (ACx << #SHIFTW)
SUB K16 << #16, [ACx,] ACy	ACy = ACx – (K16 << #16)
SUB K16 << #SHFT, [ACx,] ACy	ACy = ACx – (K16 << #SHFT)
SUB Smem << Tx, [ACx,] ACy	ACy = ACx – (Smem << Tx)
SUB Smem << #16, [ACx,] ACy	ACy = ACx – (Smem << #16)
SUB ACx, Smem << #16, ACy	ACy = (Smem << #16) – ACx
SUB [uns(]Smem[)], BORROW, [ACx,] ACy	ACy = ACx – uns(Smem) – BORROW
SUB [uns(]Smem[)], [ACx,] ACy	ACy = ACx – uns(Smem)
SUB [uns(]Smem[)] << #SHIFTW, [ACx,] ACy	ACy = ACx – (uns(Smem) << #SHIFTW)
SUB dbl(Lmem), [ACx,] ACy	ACy = ACx – dbl(Lmem)
SUB ACx, dbl(Lmem), ACy	ACy = dbl(Lmem) – ACx
SUB Xmem, Ymem, ACx	ACx = (Xmem << #16) – (Ymem << #16)

Index

A

B

C

D

E

F

I

L

M

N

O

P

R

S

T

U

X