Appendix A INSTRUCTION SET

A-1 INTRODUCTION

The programming model indicates that the DSP56300 Core central processor architecture can be viewed as three functional units operating in parallel: data arithmetic logic unit (ALU), address generation unit (AGU), and program control unit. The goal of the instruction set is to provide the capability to keep each of these units busy each instruction cycle, achieving maximum speed and minimum program size.

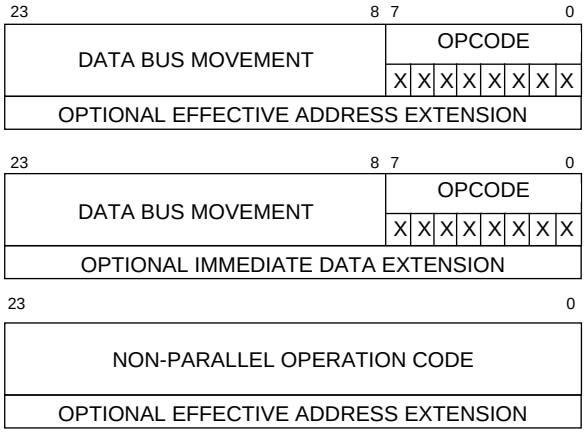
This section introduces the DSP56300 Core instruction set and instruction format. The complete range of instruction capabilities combined with the flexible addressing modes used in this processor provide a very powerful assembly language for implementing digital signal processing (DSP) algorithms. The instruction set has been designed to allow efficient coding for DSP high-level language compilers such as the C compiler. Execution time is minimized by the hardware looping capabilities, use of an instruction pipeline, and parallel moves.

A-2 INSTRUCTION FORMATS AND SYNTAX

The DSP56300 Core instructions consist of one or two 24-bit words – an operation word and an optional extension word. This extension word can be either effective address extension word or an immediate data extension word. General formats of the instruction word are shown in Figure A-1 Most instructions specify data movement on the XDB, YDB, and data ALU operations in the same operation word. The DSP56300 Core is designed to perform each of these operations in parallel.

The data bus movement field provides the operand reference type, which selects the type of memory or register reference to be made, the direction of transfer, and the effective address(es) for data movement on the XDB and/or YDB. This field may require additional information to fully specify the operand for certain addressing modes. An extension word following the operation word is used to provide immediate data, absolute address or address displacement, if required. Examples of operations that may include the extension word include move operation such as: MOVE X:$100,X0

Figure A-1. General Formats of an Instruction Word



The opcode field of the operation word specifies the data ALU operation or the program control unit operation to be performed.

The operation codes form a very versatile microcontroller unit (MCU) style instruction set, providing highly parallel operations in most programming situations.

The instruction syntax has two formats - Parallel and NonParallel, as shown in Table A-1 and Table A-2. Parallel instruction is organized into five columns: opcode, operands, and two parallel-move fields, each of them is optional, and an optional condition field. The condition field is used to disable the execution of the opcode if the condition is not true, and cannot be used in conjunction with the parallel move fields. Assembly-language source codes for some typical one-word instructions are shown in Table A-1. Because of the multiple bus structure and the parallelism of the DSP56300 Core, up to three data transfers can be specified in the instruction word – one on the X data bus (XDB), one on the Y data bus (YDB), and one within the data ALU. These transfers are explicitly specified. A fourth data transfer is implied and occurs in the program control unit (instruction word prefetch, program looping control, etc.). The opcode column indicates the data ALU operation to be performed but may be excluded if only a MOVE operation is needed. The operands column specifies the operands to be used by the opcode. The XDB and YDB columns specify optional data transfers over the XDB and/or YDB and the associated addressing modes. The address space qualifiers (X:, Y:, and L:) indicate which address space is being referenced.

Table A-1. Parallel Instructions Format

	Opcode	Operands	XDB	YDB	Condition
Example 1:	MAC	X0,Y0,A	X:(R0)+,X0	Y:(R4)+,Y0	
Example 2:	MOVE		X:-(R1),X1		
Example 3:	MAC	X1,Y1,B			
Example 4:	MPY	X0,Y0,A			IF eq

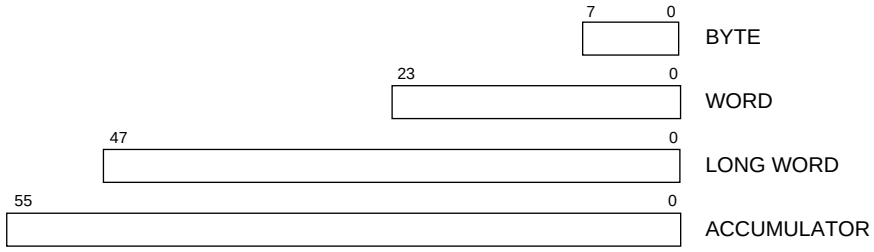
NonParallel instruction is basically organized into two columns: opcode and operands. Assembly-language source codes for some typical one-word instructions are shown in Table A-2. Nonparallel instructions include all the program control, looping and peripherals read/write instructions. They also include some Data ALU instructions that are impossible to be encoded in the opcode field of the Parallel format.

Table A-2. NonParallel Instructions Format

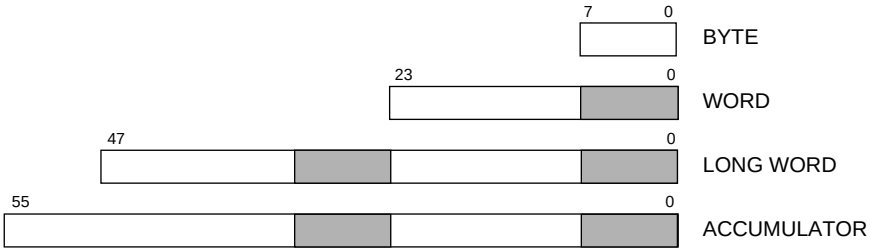
	Opcode	Operands
Example 1:	JEQ	(R5)
Example 2:	MOVEP	#data,X:ipr
Example 3:	RTS	

A-2.1 Operand Sizes

Operand sizes are defined as follows: a byte is 8 bits long, a word is 24 bits long, a long word is 48 bits long, and an accumulator is 56 bits long (see following diagram). The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation.



When in Sixteen Bit Arithmetic mode the operand sizes are as follows: a byte is 8 bits long, a word is 16 bits long, a long word is 32 bits long, and an accumulator is 40 bits long.



A-2.2 Data Organization in Registers

The ten data ALU registers support 8- or 24-bit data operands, and 16-bit data in Sixteen Bit mode. Instructions also support 48- or 56-bit data operands (32- or 40-bit in Sixteen Bit mode) by concatenating groups of specific data ALU registers. The eight address registers in the AGU support 24-bit address or data operands. The eight AGU offset registers support 24-bit offsets or may support 24-bit address or data operands. The eight AGU modifier registers support 24-bit modifiers or may support 24-bit address or data operands. The program counter (PC) supports 24-bit address operands. The status register (SR) and operating mode register (OMR) support 8,16 or 24-bit data operands. Both the loop counter (LC) and loop address (LA) registers support 24-bit address operands.

A-2.3 Data ALU Registers

The eight main data registers are 24 bits wide. Word operands occupy one register; long-word operands occupy two concatenated registers. The least significant bit (LSB) is the right-most bit (bit 0); whereas, the most significant bit (MSB) is the left-most bit (bit 23 for word operands and bit 47 for long-word operands). When in Sixteen Bit mode, the least significant bit (LSB) is bit 8; bits 24 to 31 are ignored for long-word operands; whereas, the most significant bit (MSB) is the left-most bit.

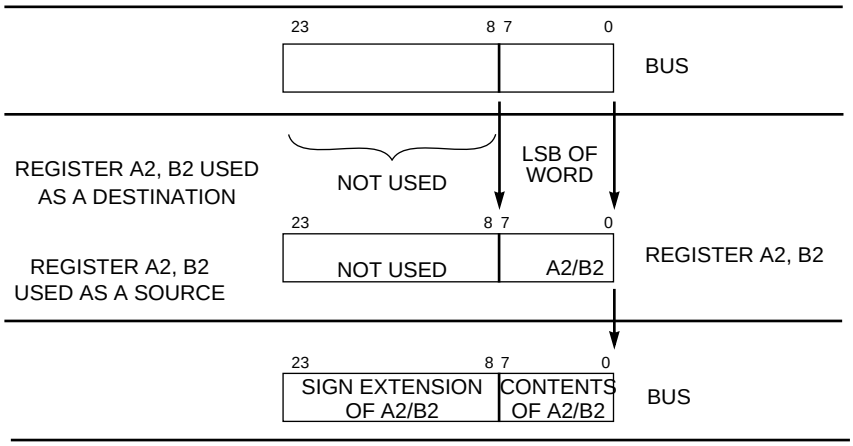
The two accumulator extension registers are eight bits wide. When an accumulator extension register is used as a source operand, it occupies the low-order portion (bits 0–7) of the word; the high-order portion (bits 8–23) is sign extended (see Table A-2). When used as a destination operand, this register receives the low-order portion of the word, and the high-order portion is not used. Accumulator operands occupy an entire group of three registers (i.e., A2:A1:A0 or B2:B1:B0). The LSB is the right-most bit (bit 0 for 24 bit mode and bit 8 for Sixteen Bit mode), and the MSB is the left-most bit (bit 55).

When a 56-bit accumulator (A or B) is specified as a **source** operand S, the accumulator value is optionally shifted according to the scaling mode bits S0 and S1 in the system status register (SR). If the data out of the shifter indicates that the accumulator extension register is in use and the data is to be moved into a 24-bit destination, the value stored in

the destination is limited to a maximum positive or negative saturation constant to minimize truncation error. Limiting does not occur if an individual 24-bit accumulator register (A1, A0, B1, or B0) is specified as a source operand instead of the full 56-bit accumulator (A or B). This limiting feature allows block floating-point operations to be performed with error detection since the L bit in the condition code register is latched.

When a 56-bit accumulator (A or B) is specified as a **destination** operand D, any 24-bit source data to be moved into that accumulator is automatically extended to 56 bits by sign extending the MS bit of the source operand (bit 23) and appending the source operand with 24 LS zeros. Note that for 24-bit source operands both the automatic sign-extension and zeroing features may be disabled by specifying the destination register to be one of the individual 24-bit accumulator registers (A1 or B1).

Figure A-2. Reading and Writing the ALU Extension Registers



When in Sixteen Bit mode, the move operations associated with Data ALU registers are altered. For further details refer to Section 3.4.1.

A-2.4 AGU Registers

The 24 AGU registers, which are 24 bits wide, may be accessed as word operands for address, address offset, address modifier, and data storage. The notation Rn is used to designate one of the eight address registers, R0–R7; the notation Nn is used to designate one of the eight address offset registers, N0–N7; and the notation Mn is used to designate one of the eight address modifier registers, M0–M7.

A-2.5 Program Control Registers

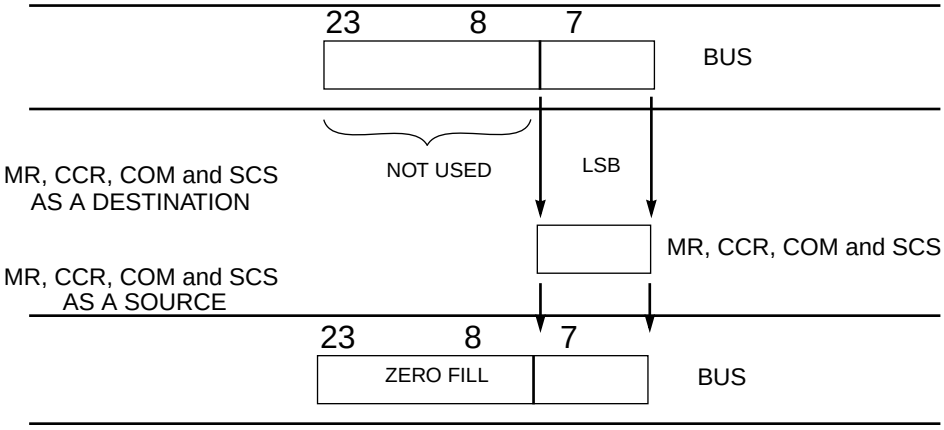
The 24-bit OMR has the chip operating mode register (COM) occupying the low-order eight bits and the extended chip operating mode register (EOM) occupying the middle-order eight bits and the system stack control status register (SCS) occupying the high-order eight bits. The Operating Mode Register (OMR) and the Vector Base Address (VBA) are accessed as word operands; however, not all of their bits are defined. These bits read as zero and should be written with zero for future compatibility. The 24-bit SR has the user condition code register (CCR) occupying the low-order eight bits and the system mode

register (MR) occupying the middle-order eight bits and the extended mode register (EMR) occupying the high-order eight bits. The SR may be accessed as a word operand. The MR and CCR may be accessed individually as word operands (see Figure A-3). The Loop Counter (LC), Loop Last Address (LA), stack size (SZ), system stack high (SSH), and system stack low (SSL) registers are 24 bits wide and are accessed as word operands. The system stack pointer (SP) is a 24-bit register that is accessed as a word operand. The PC, a special 24-bit-wide program counter register, is generally referenced implicitly as a word operand, but may also be referenced explicitly (by all PC-relative operation codes) also as a word operand.

A-2.6 Data Organization in Memory

The 24-bit program memory can store both 24-bit instruction words and instruction extension words. The 48-bit system stack (SS) can store the concatenated PC and SR registers (PC:SR) for subroutine calls, interrupts, and program looping. The SS also supports the concatenated LA and LC registers (LA:LC) for program looping. The 24-bit-wide X and Y memories can store word and byte operands. When in Sixteen Bit Arithmetic mode the X and Y memories can store 16-bit words, that occupy the low-portion of the memory word. Byte operands, which usually occupy the low-order portion of the X or Y memory word, are either zero extended or sign extended on the XDB or YDB.

Figure A-3. Reading and Writing Control Registers



A-3 INSTRUCTION GROUPS

The instruction set is divided into the following groups:

- Arithmetic
- Logical
- Bit Manipulation

- Loop
- Move
- Program Control

Each instruction group is described in the following paragraphs.

A-3.1 Arithmetic Instructions

The arithmetic instructions perform all of the arithmetic operations within the data ALU. These instructions may affect all of the CCR bits. Arithmetic instructions are register based (register direct addressing modes used for operands) so that the data ALU operation indicated by the instruction does not use the XDB, the YDB, or the global data bus (GDB). Optional data transfers may be specified with most arithmetic instructions, which allows for parallel data movement over the XDB and YDB or over the GDB during a data ALU operation. This parallel movement allows new data to be prefetched for use in subsequent instructions and allows results calculated in previous instructions to be stored.A ✔ sign in a table cell in the “Parallel Instruction” column indicates that the corresponding instruction is a parallel instruction, while a blank table cell indicates that the instruction is not a parallel instruction. The move operation that can be specified in parallel to the instruction marked is one of the parallel instructions listed in Table A-7. Arithmetic instructions can be executed conditionally, based on the condition codes generated by the previous instructions. Conditional arithmetic instructions don’t allow parallel data movement over the various data busses. Table A-3 lists the arithmetic instructions.

Table A-3. Arithmetic Instructions

Mnemonic	Description	Parallel Instruction
ABS	Absolute Value	✔
ADC	Add Long with Carry	✔
ADD	Add	✔
ADD (imm.)	Add (immediate operand)	
ADDL	Shift Left and Add	✔
ADDR	Shift Right and Add	✔
ASL	Arithmetic Shift Left	✔
ASL (mb.)	Arithmetic Shift Left (multi-bit)	
ASL (mb., imm.)	Arithmetic Shift Left (multi-bit, immediate operand)	
ASR	Arithmetic Shift Right	✔
ASR (mb.)	Arithmetic Shift Right (multi-bit)	

Mnemonic	Description	Parallel Instruction
ASR (mb., imm.)	Arithmetic Shift Right (multi-bit, immediate operand)	
CLR	Clear an Operand	✔
CMP	Compare	✔
CMP (imm.)	Compare (immediate operand)	
CMPM	Compare Magnitude	✔
CMPU	Compare Unsigned	
DEC	Decrement Accumulator	
DIV	Divide Iteration	
DMAC	Double Precision Multiply-Accumulate	
INC	Increment Accumulator	
MAC	Signed Multiply-Accumulate	✔
MAC (su,uu)	Mixed Multiply-Accumulate	
MACI	Signed Multiply-Accumulate (immediate operand)	
MACR	Signed Multiply-Accumulate and Round	✔
MACRI	Signed Multiply-Accumulate and Round (immediate operand)	
MAX	Transfer By Signed Value	✔
MAXM	Transfer By Magnitude	✔
MPY	Signed Multiply	✔
MPY (su,uu)	Mixed Multiply	
MPYI	Signed Multiply (immediate operand)	
MPYR	Signed Multiply and Round	✔
MPYRI	Signed Multiply and Round (immediate operand)	
NEG	Negate Accumulator	✔
NORM	Normalize	
NORMF	Fast Accumulator Normalize	

Mnemonic	Description	Parallel Instruction
RND	Round	✔
SBC	Subtract Long with Carry	✔
SUB	Subtract	✔
SUB (imm.)	Subtract (immediate operand)	
SUBL	Shift Left and Subtract	✔
SUBR	Shift Right and Subtract	✔
Tcc	Transfer Conditionally	
TFR	Transfer Data ALU Register	✔
TST	Test an Operand	✔

A-3.2 Logical Instructions

The logical instructions, which execute in one instruction cycle, perform all of the logical operations within the data ALU (except ANDI and ORI). They may affect all of the CCR bits and, like the arithmetic instructions, are register based. Optional data transfers may be specified with most logical instructions, allowing parallel data movement over the XDB and YDB or over the GDB during a data ALU operation. This parallel movement allows new data to be prefetched for use in subsequent instructions and allows results calculated in previous instructions to be stored. A ✔ sign in a table cell in the “Parallel Instruction” column indicates that the corresponding instruction is a parallel instruction, while a blank table cell indicates that the instruction is not a parallel instruction. The move operation that can be specified in parallel to the instruction marked is one of the parallel instructions listed in Table A-7. Table A-4 lists the logical instructions.

Table A-4. Logical Instructions

Mnemonic	Description	Parallel Instruction
AND	Logical AND	✔
AND (imm.)	Logical AND (immediate operand)	
ANDI	AND Immediate to Control Register	
CLB	Count Leading Bits	
EOR	Logical Exclusive OR	✔
EOR (imm.)	Logical Exclusive OR (immediate operand)	

Mnemonic	Description	Parallel Instruction
EXTRACT	Extract Bit Field	
EXTRACT (imm.)	Extract Bit Field (immediate operand)	
EXTRACTU	Extract Unsigned Bit Field	
EXTRACTU (imm.)	Extract Unsigned Bit Field (immediate operand)	
INSERT	INSERT Bit Field	
INSERT (imm.)	INSERT Bit Field (immediate operand)	
LSL	Logical Shift Left	✔
LSL (mb.)	Logical Shift Left (multi-bit)	
LSL (mb., imm.)	Logical Shift Left (multi-bit, immediate operand)	
LSR	Logical Shift Right	✔
LSR (mb.)	Logical Shift Right (multi-bit)	
LSR (mb.,imm.)	Logical Shift Right (multi-bit, immediate operand)	
MERGE	Merge Two Half Words	
NOT	Logical Complement	✔
OR	Logical Inclusive OR	✔
OR (imm.)	Logical Inclusive OR (immediate operand)	
ORI	OR Immediate to Control Register	
ROL	Rotate Left	✔
ROR	Rotate Right	✔

A-3.3 Bit Manipulation Instructions

The bit manipulation instructions test the state of any single bit in a memory location and then optionally set, clear, or invert the bit. The carry bit of the CCR will contain the result of the bit test. Table A-5 lists the bit manipulation instructions.

Table A-5. Bit Manipulation Instructions

Mnemonic	Description	Parallel Instruction
BCHG	Bit Test and Change	
BCLR	Bit Test and Clear	
BSET	Bit Test and Set	
BTST	Bit Test	

A-3.4 Loop Instructions

The hardware DO loop executes with no overhead cycles – i.e., it runs as fast as straight-line code. Replacing straight-line code with DO loops can significantly reduce program memory. The loop instructions control hardware looping by 1) initiating a program loop and establishing looping parameters or by 2) restoring the registers by pulling the SS when terminating a loop. Initialization includes saving registers used by a program loop (LA and LC) on the SS so that program loops can be nested. The address of the first instruction in a program loop is also saved to allow no-overhead looping. Table A-6 lists the loop instructions.

Table A-6. Loop Instructions

Mnemonic	Description	Parallel Instruction
BRKcc	Conditionally Break the current Hardware Loop	
DO	Start Hardware Loop	
DOR	Start Hardware Loop to PC-Related End-Of-Loop Location	
DO FOREVER	Start Forever Hardware Loop	
DOR FOREVER	Start Forever Hardware Loop to PC-Related End-Of-Loop Location	
ENDDO	Abort and Exit from Hardware Loop	

The ENDDO instruction is not used for normal termination of a DO loop; it is only used to terminate a DO loop before the LC has been decremented to one.

A-3.5 Move Instructions

The move instructions perform data movement over the XDB and YDB or over the GDB.

Move instructions, most of which allow Data ALU opcode in parallel, do not affect the CCR except the limit bit L if limiting is performed when reading a data ALU accumulator register. Table A-7 lists the move instructions.

Table A-7. Move Instructions

Mnemonic	Description	Parallel Instruction
LUA	Load Updated Address	
LRA	Load PC-Relative Address	
MOVE	Move Data Register	✔
MOVEC	Move Control Register	
MOVEM	Move Program Memory	
MOVEP	Move Peripheral Data	
U MOVE	Update Move	✔

A-3.6 Program Control Instructions

The program control instructions include jumps, conditional jumps, and other instructions affecting the PC, SS and the program Cache. Program control instructions may affect the CCR bits as specified in the instruction. Optional data transfers over the XDB and YDB may be specified in some of the program control instructions. Table A-8 lists the program control instructions.

Table A-8. Program Control Instructions

Mnemonic	Description	Parallel Instruction
IFcc.U	Execute Conditionally and Update CCR	
IFcc	Execute Conditionally	
Bcc	Branch Conditionally	
BRA	Branch Always	
BRCLR	Branch if Bit Clear	
BRSET	Branch if Bit Set	
BScc	Branch to Subroutine Conditionally	
BSR	Branch to Subroutine Always	
BSCLR	Branch to Subroutine if Bit Clear	
BSSET	Branch to Subroutine if Bit Set	

Mnemonic	Description	Parallel Instruction
DEBUGcc	Enter into the Debug Mode Conditionally	
DEBUG	Enter into the Debug Mode Always	
Jcc	Jump Conditionally	
JMP	Jump Always	
JCLR	Jump if Bit Clear	
JSET	Jump if Bit Set	
JScc	Jump to Subroutine Conditionally	
JSR	Jump to Subroutine Always	
JSCLR	Jump to Subroutine if Bit Clear	
JSSET	Jump to Subroutine if Bit Set	
NOP	No Operation	
PLOCK	Lock Program Cache Sector	
PUNLOCK	Unlock Program Cache Sector	
PLOCKR	Lock PC-Related Program Cache Sector	
PUNLOCKR	Unlock PC-Related Program Cache Sector	
PFREE	Unlock all Program Cache Locked Sectors	
PFLUSH	Reset Program Cache State	
PFLUSHUN	Reset Program Cache State to all Unlocked Sectors	
REP	Repeat Next Instruction	
RESET	Reset On-Chip Peripheral Devices	
RTI	Return from Interrupt	
RTS	Return from Subroutine	
STOP	Stop Processing (Low-Power Standby)	
TRAPcc	Trap Conditionally	
TRAP	Trap Always	
WAIT	Wait for Interrupt (Low-Power Standby)	

A-4 INSTRUCTION GUIDE

The following information is included in each instruction description:

1. **Name and Mnemonic:** The mnemonic is highlighted in **bold** type for easy reference.
2. **Assembler Syntax and Operation:** For each instruction syntax, the corresponding operation is symbolically described. If there are several operations indicated on a single line in the operation field, those operations do not necessarily occur in the order shown but are generally assumed to occur in parallel. If a parallel data move is allowed, it will be indicated in parenthesis in both the assembler syntax and operation fields. If a letter in the mnemonic is optional, it will be shown in parenthesis in the assembler syntax field.
3. **Description:** A complete text description of the instruction is given together with any special cases and/or condition code anomalies of which the user should be aware when using that instruction.
4. **Condition Codes:** The status register is depicted with the condition code bits which can be affected by the instruction. Not all bits in the status register are used. Those which are reserved are indicated with a gray box covering its area.
5. **Instruction Format:** The instruction fields, the instruction opcode, and the instruction extension word are specified for each instruction syntax. When the extension word is optional, it is so indicated. The values which can be assumed by each of the variables in the various instruction fields are shown under the instruction field's heading.

A-4.1 NOTATION

Each instruction description contains symbols used to abbreviate certain operands and operations. Table A-9 lists the symbols used and their respective meanings. Depending on the context, registers refer to either the register itself or the contents of the register.

Table A-9. Instruction Description Notation

Data ALU Registers Operands	
Xn	Input Register X1 or X0 (24 Bits)
Yn	Input Register Y1 or Y0 (24 Bits)
An	Accumulator Registers A2, A1, A0 (A2 — 8 Bits, A1 and A0 — 24 Bits)
Bn	Accumulator Registers B2, B1, B0 (B2 — 8 Bits, B1 and B0 — 24 Bits)
X	Input Register X = X1: X0 (48 Bits)

Data ALU Registers Operands

Y	Input Register Y = Y1: Y0 (48 Bits)
A	Accumulator A = A2: A1: A0 (56 Bits)
B	Accumulator B = B2: B1: B0 (56 BIts)
AB	Accumulators A and B = A1: B1 (48 Bits)
BA	Accumulators B and A = B1: A1 (48 Bits)
A10	Accumulator A = A1: A0 (48 Bits)
B10	Accumulator B= B1:B0 (48 bits)

Program Control Unit Registers Operands

PC	Program Counter Register (24 Bits)
EMR	Extended Mode Register (8 Bits)
MR	Mode Register (8 Bits)
CCR	Condition Code Register (8 Bits)
SR	Status Register = EMR:MR:CCR (24 Bits)
SCS	System Stack Control Status Register (8 Bits)
EOM	Extended Chip Operating Mode Register (8 Bits)
COM	Chip Operating Mode Register (8 Bits)
OMR	Operating Mode Register = SCS:EOM:COM (24 Bits)
SZ	System Stack Size Register (24 Bits)
SC	System Stack Counter Register (5 Bits)
VBA	Vector Base Address (24 Bits, 8 of them are always zero)
LA	Hardware Loop Address Register (24 Bits)
LC	Hardware Loop Counter Register (24 Bits)
SP	System Stack Pointer Register (24 Bits)
SSH	Upper Portion of the Current Top of the Stack (24 Bits)
SSL	Lower Portion of the Current Top of the Stack (24 Bits)
SS	System Stack RAM = SSH: SSL (16 Locations by 32 Bits)

Address Operands

ea	Effective Address
eax	Effective Address for X Bus
eay	Effective Address for Y Bus
xxxx	Absolute or Long Displacement Address (24 Bits)
xxx	Short or Short Displacement Jump Address (12 Bits)
xxx	Short Displacement Jump Address (9 Bits)
aaa	Short Displacement Address (7 Bits Sign Extended)
aa	Absolute Short Address (6 Bits, Zero Extended)
pp	High I/O Short Address (6 Bits, Ones Extended)
qq	Low I/O Short Address (6 Bits)
<. . .>	Specifies the Contents of the Specified Address
X:	X Memory Reference
Y:	Y Memory Reference
L:	Long Memory Reference = X Concatenated with Y
P:	Program Memory Reference

Miscellaneous Operands

S, Sn	Source Operand Register
D, Dn	Destination Operand Register
D [n]	Bit n of D Destination Operand Register
#n	Immediate Short Data (5 Bits)
#xx	Immediate Short Data (8 Bits)
#xxx	Immediate Short Data (12 Bits)
#xxxxxx	Immediate Data (24 Bits)
r	Rounding Constant
#bbbbb	Operand Bit Select (5 Bits)

Unary Operands

-	Negation Operator
—	Logical NOT Operator (Overbar)
PUSH	Push Specified Value onto the System Stack (SS) Operator

Unary Operands

PULL	Pull Specified Value from the System Stack (SS) Operator
READ	Read the Top of the System Stack (SS) Operator
PURGE	Delete the Top Value on the System Stack (SS) Operator
| |	Absolute Value Operator

Binary Operands

+	Addition Operator
-	Subtraction Operator
*	Multiplication Operator
÷, /	Division Operator
+	Logical Inclusive OR Operator
•	Logical AND Operator
⊕	Logical Exclusive OR Operator
→	"Is Transferred To" Operator
:	Concatenation Operator

Addressing Mode Operators

<<	I/O Short Addressing Mode Force Operator
<	Short Addressing Mode Force Operator
>	Long Addressing Mode Force Operator
#	Immediate Addressing Mode Operator
#>	Immediate Long Addressing Mode Force Operator
#<	Immediate Short Addressing Mode Force Operator

Mode Register Symbols

LF	Loop Flag Bit Indicating When a DO Loop is in Progress
DM	Double Precision Multiply Bit Indicating if the Chip is in Double Precision Multiply Mode
SB	Sixteen Bit Arithmetic Mode
RM	Rounding Mode

Mode Register Symbols

S1, S0	Scaling Mode Bits Indicating the Current Scaling Mode
I1, I0	Interrupt Mask Bits Indicating the Current Interrupt Priority Level

Condition Code Register (CCR) Symbols

S	Block Floating Point Scaling Bit Indicating Data Growth Detection
L	Limit Bit Indicating Arithmetic Overflow and/or Data Shifting/Limiting
E	Extension Bit Indicating if the Integer Portion of Data ALU result is in Use
U	Unnormalized Bit Indicating if the Data ALU Result is Unnormalized
N	Negative Bit Indicating if Bit 55 of the Data ALU Result is Set
Z	Zero Bit Indicating if the Data ALU Result Equals Zero
V	Overflow Bit Indicating if Arithmetic Overflow has Occurred in Data ALU
C	Carry Bit Indicating if a Carry or Borrow Occurred in Data ALU Result

( )	Optional Letter, Operand, or Operation
(...)	Any Arithmetic or Logical Instruction Which Allows Parallel Moves
EXT	Extension Register Portion of an Accumulator (A2 or B2)
LS	Least Significant
LSP	Least Significant Portion of an Accumulator (A0 or B0)
MS	Most Significant
MSP	Most Significant Portion of a n Accumulator (A1 or B1)
S/L	Shifting and/or Limiting on a Data ALU Register
Sign Ext	Sign Extension of a Data ALU Register
Zero	Zeroing of a Data ALU Register

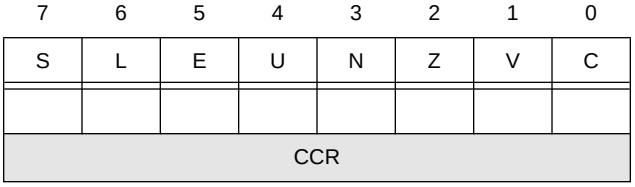
Address ALU Registers Operands

Rn	Address Registers R0 - R7 (24 Bits)
Nn	Address Offset Registers N0 - N7 (24 Bits)
Mn	Address Modifier Registers M0 - M7 (24 Bits)

A-5 CONDITION CODE COMPUTATION

The condition code register (CCR) portion of the status register (SR) consists of eight defined bits.

Condition Codes:



S — Scaling Bit
L — Limit Bit
E — Extension Bit
U — Unnormalized Bit
N — Negative Bit
Z — Zero Bit
V — Overflow Bit
C — Carry Bit

The E, U, N, Z, V, and C bits are **true** condition code bits that reflect the condition of the **result of a data ALU operation**. These condition code bits are **not "sticky"** and are **not affected by address ALU calculations or by data transfers** over the X, Y, or global data buses. The L bit is a **"sticky" overflow bit** which indicates that an overflow has occurred in the data ALU or that data limiting has occurred when moving the contents of the A and/or B accumulators. The S bit is a "sticky" bit used in block floating point operations to indicate the need to scale the number in A or B.

The full description of every instruction contains an illustration showing how the instruction affects the various condition codes.

An instruction can affect a condition code according to three different rules:

standard mark	The affect on the condition code
×	Unchanged by the instruction
✔	Changed by the instruction, according to the standard definition of the condition code
●	Changed by the instruction, according to a special definition of the condition code, depicted as part of the instruction full description

The standard definition of the condition code bits follows.

S (Scaling Bit) This bit is computed, according to the logical equations in the table below, when an instruction or a parallel move reads the contents of accumulator A or B to XDB or YDB. It is a “sticky” bit, cleared only by an instruction that specifically clears it, or hardware reset.

S0	S1	Scaling Mode	S bit equation
0	0	No scaling	S = (A46 XOR A45) OR (B46 XOR B45) OR S (previous)
0	1	Scale down	S = (A47 XOR A46) OR (B47 XOR B46) OR S (previous)
1	0	Scale up	S = (A45 XOR A44) OR (B45 XOR B44) OR S (previous)
1	1	Reserved	S undefined

The scaling bit (S) is used to detect data growth, which is required in Block Floating Point FFT operation. The scaling bit will be set if the absolute value in the accumulator, before scaling, was greater or equal to 0.25 and smaller than 0.75. Typically, the bit is tested after each pass of a radix 2 decimation-in-time FFT and, if it is set, the appropriate scaling mode should be activated in the next pass. The Block Floating Point FFT algorithm is described in the Motorola application note APR4/D, “Implementation of Fast Fourier Transforms on Motorola’s DSP56000/DSP56001 and DSP96002 Digital Signal Processors.”

L (Limit Bit) Set if the overflow bit V is set or if an instruction or a parallel move causes the data shifter/limiters to perform a limiting operation while reading the contents of accumulator A or B to XDB or YDB. In Arithmetic Saturation Mode, the limit bit is also set when an arithmetic saturation occurs in the Data ALU result. Not affected otherwise. This bit is “sticky” and must be cleared only by an instruction that specifically clears it, or hardware reset.

E (Extension Bit) Cleared if all the bits of the **signed integer portion** of the Data ALU

result are the **same** – i.e., the bit patterns are either 00. . . 00 or 11. . . 11. Set otherwise.
The signed integer portion is defined by the scaling mode as shown in the following table:

S1	S0	Scaling Mode	Integer Portion
0	0	No Scaling	Bits 55,54.....48,47
0	1	Scale Down	Bits 55,54.....49,48
1	0	Scale Up	Bits 55,54.....47,46

Note that the **signed integer portion** of an accumulator **IS NOT** necessarily the same as the **extension register portion** of that accumulator. The signed integer portion of an accumulator consists of the MS 8, 9, or 10 bits of that accumulator, depending on the scaling mode being used. The extension register portion of an accumulator (A2 or B2) is always the MS 8 bits of that accumulator. **The E bit refers to the signed integer portion of an accumulator and NOT the extension register portion of that accumulator.** For example, if the current scaling mode is set for no scaling (i.e., S1=S0=0), the signed integer portion of the A or B accumulator consists of **bits 47 through 55**. If the A accumulator contained the signed 56-bit value $00:800000:000000 as a **result of a data ALU operation**, the E bit **would** be set (E=1) since the **9** MS bits of that accumulator were not all the same (i.e., neither 00.. 00 nor 11.. 11). This means that data limiting **will** occur if that 56-bit value is specified as a **source** operand in a move-type operation. This limiting operation will result in either a positive or negative, 24-bit or 48-bit saturation constant being stored in the specified destination. The **only** situation in which the signed integer portion of an accumulator and the extension register portion of an accumulator are the same is in the “Scale Down” scaling mode (i.e., S1=0 and S0=1).

U (Unnormalized Bit) Set if the two MS bits of the MSP portion of the Data ALU result are the same. Cleared otherwise. The MSP portion is defined by the scaling mode. The U bit is computed as follows:

S1	S0	Scaling Mode	U Bit Computation
0	0	No Scaling	U = $\overline{\text{(Bit 47 xor Bit 46)}}$
0	1	Scale Down	U = $\overline{\text{(Bit 48 xor Bit 47)}}$
1	0	Scale Up	U = $\overline{\text{(Bit 46 xor Bit 45)}}$

The result of calculating the U bit in this fashion is that the definition of positive normalized number, p, is $0.5 \leq p < 1.0$ and the definition of negative normalized number, n, is $-1.0 \leq n < -0.5$.

N (Negative Bit) Set if the MS bit (bit 55 in arithmetic instructions or bit 47 in logical instructions) of the Data ALU result is set. Cleared otherwise.

Z (Zero Bit) Set if the Data ALU result equals zero. Cleared otherwise.

V (Overflow Bit) Set if an arithmetic overflow occurs in the 56-bit Data ALU result (40-bit result in Sixteen Bit mode). Cleared otherwise. This indicates that the result cannot be represented in the 56-bit (40-bit) accumulator; thus, the accumulator has overflowed.
In Arithmetic Saturation Mode, an arithmetic overflow occurs if the Data ALU result is not representable in the accumulator without the extension part, i.e. 48-bit accumulator (32-bit in Sixteen Bit Mode).

C (Carry Bit) Set if a carry is generated out of the MS bit of the Data ALU result of an addition or if a borrow is generated out of the MS bit of the Data ALU result of a subtraction. Cleared otherwise. The carry or borrow is generated out of bit 55 of the Data ALU result. The carry bit is also affected by bit manipulation, rotate, shift and compare instructions. The carry bit is not affected by the Arithmetic Saturation Mode.

A-6 INSTRUCTIONS DESCRIPTIONS

The following section describes each instruction in the DSP56300 Core instruction set in complete detail. Instructions which allow parallel moves include the notation “(parallel move)” in both the **Assembler Syntax** and the **Operation** fields. The MOVE instruction is equivalent to a NOP with parallel moves. Therefore, a detailed description of each parallel move is given with the MOVE instruction details.

Whenever an instruction uses an accumulator as both a destination operand for data ALU operation and as a source for a parallel move operation, the parallel move operation will use the value in the accumulator prior to execution of any data ALU operation.

ABS ABS

Absolute Value

Operation:

| D | → D (parallel move)

Assembler Syntax:

ABS D (parallel move)

Description: Take the absolute value of the destination operand D and store the result in the destination accumulator.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	×
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

ABS D

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 1 0	d 1 1 0
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

ADC ADC

Add Long with Carry

Operation:

S+C+D → D (parallel move)

Assembler Syntax:

ADC S,D (parallel move)

Description: Add the source operand S and the carry bit C of the condition code register to the destination operand D and store the result in the destination accumulator. Long words (48 bits) may be added to the (56-bit) destination accumulator.

Note: The carry bit is set correctly for multiple precision arithmetic using long-word operands if the extension register of the destination accumulator (A2 or B2) is the sign extension of bit 47 of the destination accumulator (A or B).

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	✔
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

ADC S,D

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 1 J	d 0 0 1
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

{S} **J** Source register [X,Y] (see Table A-11 on page A-239)

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

ADD ADD

Add

Operation:	Assembler Syntax:
S+D→D (parallel move)	ADD S,D (parallel move)
#xx+D→D	ADD #xx,D
#xxxxxx+D→D	ADD #xxxxxx,D

Description: Add the source operand S to the destination operand D and store the result in the destination accumulator. The source can be a register (word - 24 bits, long word - 48 bits or accumulator - 56 bits), short immediate (6 bits) or long immediate (24 bits).

When using 6-bit immediate data, the data is interpreted as an unsigned integer. That is, the 6 bits will be right aligned and the remaining bits will be zeroed to form a 24-bit source operand.

Note: The carry bit is set correctly using word or long-word source operands if the extension register of the destination accumulator (A2 or B2) is the sign extension of bit 47 of the destination accumulator (A or B). Thus, the carry bit is always set correctly using accumulator source operands, but can be set incorrectly if A1, B1, A10, B10 or immediate operand are used as source operands and A2 and B2 are not replicas of bit 47.

Condition Codes:

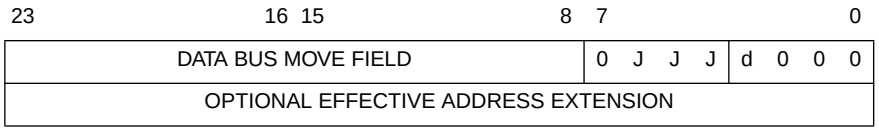
7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	✔
CCR							

✔ This bit is changed according to the standard definition

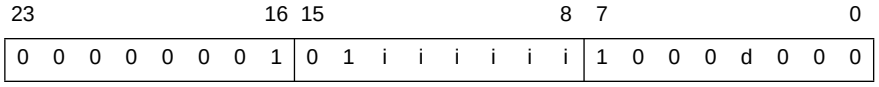
× This bit is unchanged by the instruction

Instruction Formats and opcodes:

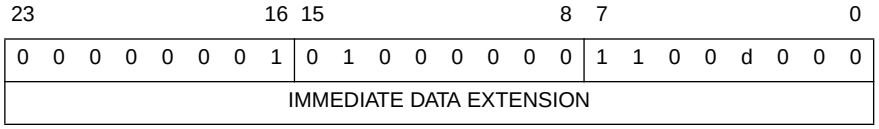
ADD S,D



ADD #xx,D



ADD #xxxxxx,D



Instruction Fields:

{S}	**JJJ**	Source register [B/A,X,Y,X0,Y0,X1,Y1] (see Table A-14 on page A-240)
{D}	**d**	Destination accumulator [A/B] (see Table A-10 on page A-239)
{#xx}	**iiiiii**	6-bit Immediate Short Data
{#xxxxxx}		24-bit Immediate Long Data extension word

ADDL ADDL

Shift Left and Add Accumulators

Operation:

S+2*D→D (parallel move)

Assembler Syntax:

ADDL S,D (parallel move)

Description: Add the source operand S to two times the destination operand D and store the result in the destination accumulator. The destination operand D is arithmetically shifted one bit to the left, and a zero is shifted into the LS bit of D prior to the addition operation. The carry bit is set correctly if the source operand does not overflow as a result of the left shift operation. The overflow bit may be set as a result of either the shifting or addition operation (or both). This instruction is useful for efficient divide and decimation in time (DIT) FFT algorithms.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	●	✔
CCR							

- ● V Set if overflow has occurred in A or B result or the MS bit of the destination operand is changed as a result of the instruction's left shift
- ✔ This bit is changed according to the standard definition
- × This bit is unchanged by the instruction

Instruction Formats and opcodes:

ADDL S,D

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 0 1	d 0 1 0
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

{S} The source accumulator is B if the destination accumulator (selected by the **d** bit in the opcode) is A, or A if the destination accumulator is B

ADDR ADDR

Shift Right and Add Accumulators

Operation:

S+D / 2→D (parallel move)

Assembler Syntax:

ADDR S,D (parallel move)

Description: Add the source operand S to one-half the destination operand D and store the result in the destination accumulator. The destination operand D is arithmetically shifted one bit to the right while the MS bit of D is held constant prior to the addition operation. In contrast to the ADDL instruction, the carry bit is always set correctly, and the overflow bit can only be set by the addition operation and not by an overflow due to the initial shifting operation. This instruction is useful for efficient divide and decimation in time (DIT) FFT algorithms.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	✔	✔
CCR							

✔ This bit is changed according to the standard definition

× This bit is unchanged by the instruction

Instruction Formats and opcodes:

ADDR S,D

23 16	15 8	7 0
DATA BUS MOVE FIELD	0 0 0 0	d 0 1 0
OPTIONAL EFFECTIVE ADDRESS EXTENSION		

Instruction Fields:

{D} **d** Destination accumulator [A,B] (see Table A-10 on page A-239)

{S} The source accumulator is B if the destination accumulator (selected by the **d** bit in the opcode) is A, or A if the destination accumulator is B

AND AND

Logical AND

Operation:

S • D[47:24]→D[47:24] (parallel move)

#xx • D[47:24]→D[47:24]

#xxxxxx • D[47:24]→D[47:24]

where •denotes the logical AND operator

Assembler Syntax:

AND S,D (parallel move)

AND #xx,D

AND #xxxxxx,D

Description: Logically AND the source operand S with bits 47-24 of the destination operand D and store the result in bits 47-24 of the destination accumulator. The source can be a 24-bit register, 6-bit short immediate or 24-bit long immediate. This instruction is a 24-bit operation. The remaining bits of the destination operand D are not affected.

When using 6-bit immediate data, the data is interpreted as an unsigned integer. That is, the 6 bits will be right aligned and the remaining bits will be zeroed to form a 24-bit source operand.

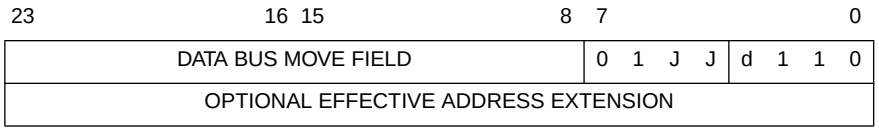
Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	×	×	×	●	●	●	×
CCR							

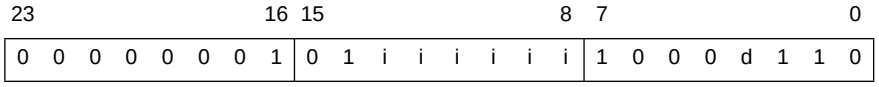
- ● N Set if bit 47 of the result is set
- ● Z Set if bits 47-24 of the result are zero
- ● V Always cleared
- ✔ This bit is changed according to the standard definition
- × This bit is unchanged by the instruction

Instruction Formats and opcodes:

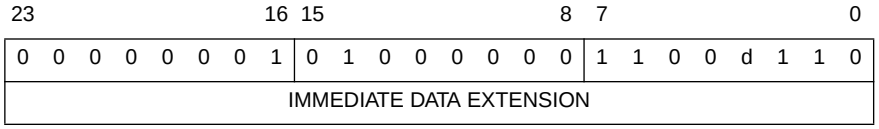
AND S,D



AND #xx,D



AND #xxxxxx,D



Instruction Fields:

{S}	**JJ**	Source input register [X0,X1,Y0,Y1] (see Table A-12 on page A-239)
{D}	**d**	Destination accumulator [A/B] (see Table A-10 on page A-239)
{#xx}	**iiiiii**	6-bit Immediate Short Data
{#xxxxxx}		24-bit Immediate Long Data extension word

ANDI ANDI

AND Immediate with Control Register

Operation:
#xx • D→D
where •denotes the logical AND operator

Assembler Syntax:
AND(I) #xx,D

Description: Logically AND the 8-bit immediate operand (#xx) with the contents of the destination control register D and store the result in the destination control register. The condition codes are affected only when the condition code register (CCR) is specified as the destination operand.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
•	•	•	•	•	•	•	•
CCR							

For CCR Operand:

- S Cleared if bit 7 of the immediate operand is cleared
- L Cleared if bit 6 of the immediate operand is cleared
- E Cleared if bit 5 of the immediate operand is cleared
- U Cleared if bit 4 of the immediate operand is cleared
- N Cleared if bit 3 of the immediate operand is cleared
- Z Cleared if bit 2 of the immediate operand is cleared
- V Cleared if bit 1 of the immediate operand is cleared
- C Cleared if bit 0 of the immediate operand is cleared

For MR and OMR Operands: The condition codes are not affected using these operands.

Instruction Formats and opcodes:

	23 16	15 8	7 0
AND(I) #xx,D	0 0 0 0 0 0 0 0	i i i i i i i i	1 0 1 1 1 0 E E

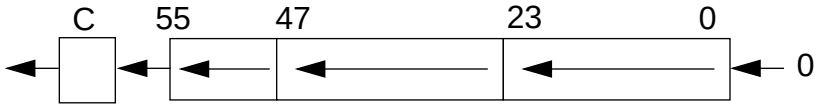
Instruction fields:

{D}	**EE**	Program Controller register [MR,CCR,COM,EOM] (see Table A-13 on page A-239)
{#xx}	**iiiiiiii**	Immediate Short Data

ASL ASL

Arithmetic Shift Accumulator Left

Operation:



Assembler Syntax:

ASL D (parallel move)

ASL #ii,S2,D

ASL S1,S2,D

Description:

Single bit shift:

Arithmetically shift the destination accumulator D one bit to the left and store the result in the destination accumulator. The MS bit of D prior to instruction execution is shifted into the carry bit C and a zero is shifted into the LS bit of the destination accumulator D.

Multi-bit shift:

The contents of the source accumulator S2 are shifted left #ii bits. Bits shifted out of position 55 are lost, but for the last bit which is latched in the carry bit C. Zeros are supplied to the vacated positions on the right. The result is placed into destination accumulator D. The number of bits to shift is determined by the 6-bit immediate field in the instruction, or by the 6-bit unsigned integer located in the 6 LSBs of the control register S1. If a zero shift count is specified, the carry bit is cleared. The difference between ASL and LSL is that ASL operates on the entire 56 bits of the accumulator and therefore sets the V bit if the number overflowed.

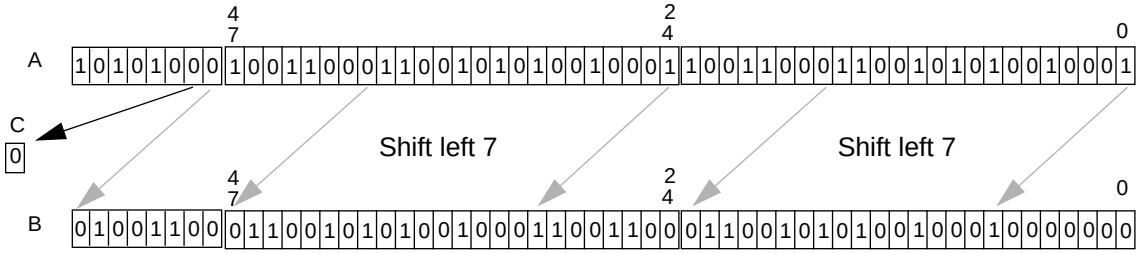
This is a 56 bit operation.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	●	●
CCR							

- ● V Set if bit 55 is changed any time during the shift operation. Cleared otherwise.
- ● C Set if the last bit shifted out of the operand is set. Cleared otherwise. Cleared for a shift count of zero.
- × This bit is unchanged by the instruction
- ✔ This bit is changed according to the standard definition

Example: ASL #7,A, B



Instruction Formats and opcodes:

ASL D

23								8	7						0
DATA BUS MOVE FIELD									0	0	1	1	d	0	1	0
OPTIONAL EFFECTIVE ADDRESS EXTENSION																

ASL #ii,S2,D

23								16	15							8	7							0
0	0	0	0	1	1	0	0		0	0	0	1	1	1	0	1	S	i	i	i	i	i	i	D

ASL S1,S2,D

23								16	15							8	7							0
0	0	0	0	1	1	0	0		0	0	0	1	1	1	1	0	0	1	0	S	s	s	s	D

Instruction Fields:

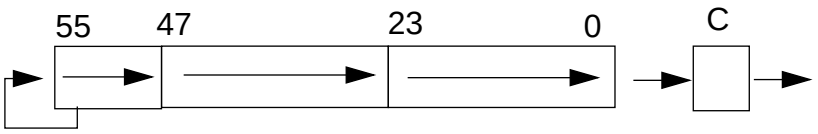
{S2}	**S**	Source accumulator [A,B] (see Table A-10 on page A-239)
{D}	**D**	Destination accumulator [A,B] (see Table A-10 on page A-239)
{S1}	**sss**	Control register [X0,X1,Y0,Y1,A1,B1] (see Table A-15 on page A-240)
{#ii}	**iiiiii**	6 bit unsigned integer [0-55] denoting the shift amount

In the **control register** S1: bits 5-0 (LSB) are used as #ii field, and the rest of the register is ignored.

ASR ASR

Arithmetic Shift Accumulator Right

Operation:



Assembler Syntax:

ASR D (parallel move)

ASR #ii,S2,D

ASR S1,S2,D

Description:

Single bit shift:

Arithmetically shift the destination operand D one bit to the right and store the result in the destination accumulator. The LS bit of D prior to instruction execution is shifted into the carry bit C, and the MS bit of D is held constant.

Multi-bit shift:

The contents of the source accumulator S2 are shifted right #ii bits. Bits shifted out of position 0 are lost, but for the last bit which is latched in the carry bit. Copies of the MSB are supplied to the vacated positions on the left. The result is placed into destination accumulator D. The number of bits to shift is determined by the 6-bit immediate field in the instruction, or by the 6-bit unsigned integer located in the 6 LSBs of the control register S1. If a zero shift count is specified, the carry bit is cleared.

This is a 56- or 40-bit operation, depending on SA bit value in status register.

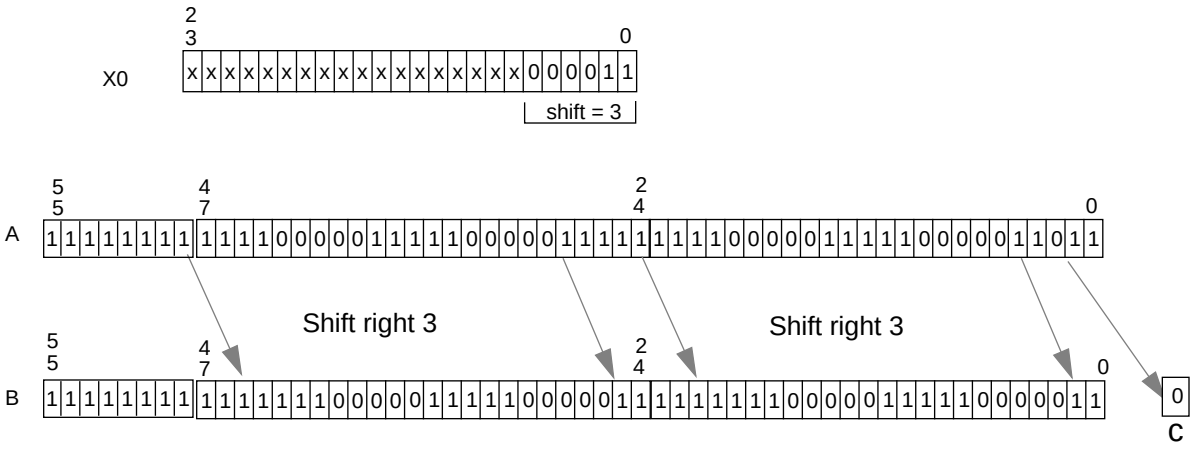
Note: if the number of shifts indicated by the 6 LSBs of the control register or by the immediate field, exceeds the value of 56 (40 in sixteen bit arithmetic mode), then the result would be undefined.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
✔	✔	✔	✔	✔	✔	●	●
CCR							

- ● V Always cleared
- ● C Set if the last bit shifted out of the operand is set. Cleared otherwise. Cleared for a shift count of zero
- x This bit is unchanged by the instruction
- ✔ This bit is changed according to the standard definition

Example: ASR X0,A,B



Instruction Formats and opcodes:

ASR D

23								8	7							0
DATA BUS MOVE FIELD									0	0	1	0	d	0	1	0
OPTIONAL EFFECTIVE ADDRESS EXTENSION																

ASR #ii,S2,D

23								16	15							8	7							0
0	0	0	0	1	1	0	0		0	0	0	1	1	1	0	0	S	i	i	i	i	i	i	D

ASR S1,S2,D

23								16	15							8	7							0
0	0	0	0	1	1	0	0		0	0	0	1	1	1	1	0	0	1	1	S	s	s	s	D

Instruction Fields:

{S2}	**S**	Source accumulator [A,B] (see Table A-10 on page A-239)
{D}	**D**	Destination accumulator [A,B] (see Table A-10 on page A-239)
{S1}	**sss**	Control register [X0,X1,Y0,Y1,A1,B1] (see Table A-15 on page A-240)
{#ii}	**iiiiii**	6 bit unsigned integer [0-55] denoting the shift amount

In the **control register** S1: bits 5-0 (LSB) are used as #ii field, and the rest of the register is ignored.

Bcc Bcc

Branch Conditionally

Operation:

If	cc,	then	PC+xxxx	→	PC
		else	PC+1	→	PC
If	cc,	then	PC+xxx	→	PC
		else	PC+1	→	PC
If	cc,	then	PC+Rn	→	PC
		else	PC+1	→	PC

Assembler Syntax:

Bcc	xxxx
Bcc	xxx
Bcc	Rn

Description: If the specified condition is true, program execution continues at location PC+displacement. If the specified condition is false, the PC is incremented and program execution continues sequentially. The displacement is a 2's complement 24-bit integer that represents the relative distance from the current PC to the destination PC. Short Displacement, Long Displacement and Address Register PC Relative addressing modes may be used. The Short Displacement 9-bit data is sign extended to form the PC relative displacement.

The conditions that the term "**cc**" can specify are listed on Table A-42 on page A-250.

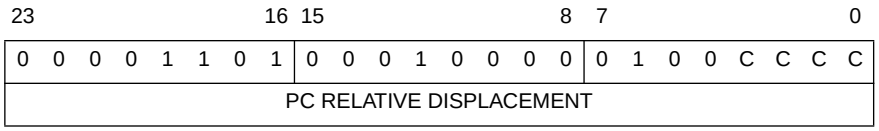
Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
×	×	×	×	×	×	×	×
CCR							

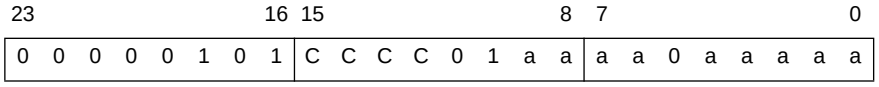
× This bit is unchanged by the instruction

Instruction Formats and opcodes:

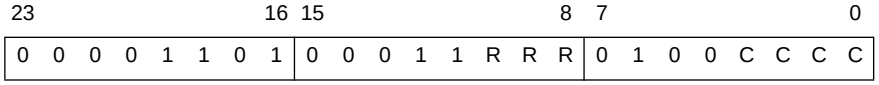
Bcc xxxx



Bcc xxx



Bcc Rn



Instruction Fields:

{cc}	**CCCC**	Condition code (see Table A-43 on page A-251)
{xxxx}		24-bit PC Relative Long Displacement
{xxx}	**aaaaaaaaa**	Signed PC Relative Short Displacement
{Rn}	**RRR**	Address register [R0-R7]

BCHG BCHG

Bit Test and Change

Operation:		Assembler Syntax:
D[n] → C	$\overline{D[n]}$ → D[n]	BCHG #n,[XorY]:ea
D[n] → C	$\overline{D[n]}$ → D[n]	BCHG #n,[XorY]:aa
D[n] → C	$\overline{D[n]}$ → D[n]	BCHG #n,[XorY]:pp
D[n] → C	$\overline{D[n]}$ → D[n]	BCHG #n,[XorY]:qq
D[n] → C	$\overline{D[n]}$ → D[n]	BCHG #n,D

Description: Test the n^{th} bit of the destination operand D, complement it, and store the result in the destination location. The state of the n^{th} bit is stored in the carry bit C of the condition code register. The bit to be tested is selected by an immediate bit number from 0–23. This instruction performs a read-modify-write operation on the destination location using two destination accesses before releasing the bus. This instruction provides a test-and-change capability which is useful for synchronizing multiple processors using a shared memory. This instruction can use all memory alterable addressing modes.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
●	●	●	●	●	●	●	●
CCR							

For destination operand SR:

- C Complemented if bit 0 is specified. Not affected otherwise.
- V Complemented if bit 1 is specified. Not affected otherwise.
- Z Complemented if bit 2 is specified. Not affected otherwise.
- N Complemented if bit 3 is specified. Not affected otherwise.
- U Complemented if bit 4 is specified. Not affected otherwise.
- E Complemented if bit 5 is specified. Not affected otherwise.
- L Complemented if bit 6 is specified. Not affected otherwise.
- S Complemented if bit 7 is specified. Not affected otherwise.

For other destination operands:

- C Set if bit tested is set. Cleared otherwise.
- V Not affected.
- Z Not affected.
- N Not affected.
- U Not affected.
- E Not affected.
- L According to the standard definition.
- S According to the standard definition.

MR Status Bits:

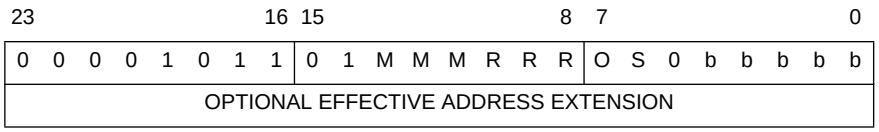
For destination operand SR:

- I0 Changed if bit 8 is specified. Not affected otherwise
- I1 Changed if bit 9 is specified. Not affected otherwise
- So Changed if bit 10 is specified. Not affected otherwise
- S1 Changed if bit 11 is specified. Not affected otherwise
- RM Changed if bit 12 is specified. Not affected otherwise
- SB Changed if bit 13 is specified. Not affected otherwise
- DM Changed if bit 14 is specified. Not affected otherwise
- LF Changed if bit 15 is specified. Not affected otherwise

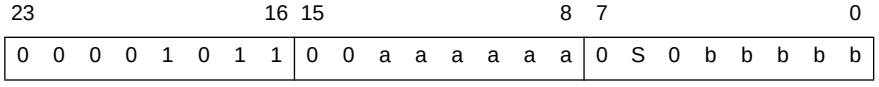
For other destination operands: MR status bits are not affected.

Instruction Formats and opcodes:

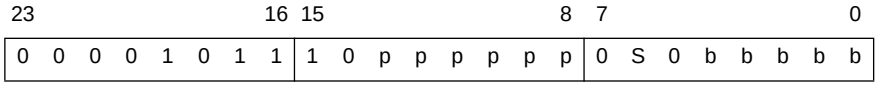
BCHG #n,[X or Y]:ea



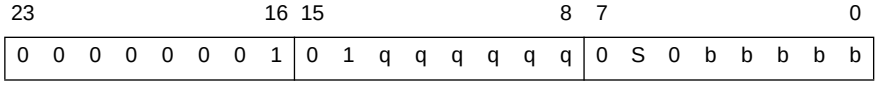
BCHG #n,[X or Y]:aa



BCHG #n,[X or Y]:pp



BCHG #n,[X or Y]:qq



BCHG #n,D

Instruction Fields:

{#n}	**bbbbb**	Bit number [0-23]
{ea}	**MMMRRR**	Effective Address (see Table A-16 on page A-241)
{X /Y}	**S**	Memory Space [X,Y] (see Table A-17 on page A-241)
{aa}	**aaaaaa**	Absolute Address [0-63]
{pp}	**pppppp**	I/O Short Address [64 addresses: $FFFFC0-$FFFFFF]
{qq}	**qqqqqq**	I/O Short Address [64 addresses: $FFFF80-$FFFFBF]
{D}	**DDDDDD**	Destination register [all on-chip registers] (see Table A-22 on page A-243)

BCLR BCLR

Bit Test and Clear

Operation:		Assembler Syntax:	
D[n] → C	0 → D[n]	BCLR	#n,[XorY]:ea
D[n] → C	0 → D[n]	BCLR	#n,[XorY]:aa
D[n] → C	0 → D[n]	BCLR	#n,[XorY]:pp
D[n] → C	0 → D[n]	BCLR	#n,[XorY]:qq
D[n] → C	0 → D[n]	BCLR	#n,D

Description: Test the n^{th} bit of the destination operand D, clear it and store the result in the destination location. The state of the n^{th} bit is stored in the carry bit C of the condition code register. The bit to be tested is selected by an immediate bit number from 0–23. This instruction performs a read-modify-write operation on the destination location using two destination accesses before releasing the bus. This instruction provides a test-and-clear capability which is useful for synchronizing multiple processors using a shared memory. This instruction can use all memory alterable addressing modes.

Condition Codes:

7	6	5	4	3	2	1	0
S	L	E	U	N	Z	V	C
●	●	●	●	●	●	●	●
CCR							

CCR Condition Codes:

For destination operand SR:

- C Cleared if bit 0 is specified. Not affected otherwise.
- V Cleared if bit 1 is specified. Not affected otherwise.
- Z Cleared if bit 2 is specified. Not affected otherwise.
- N Cleared if bit 3 is specified. Not affected otherwise.
- U Cleared if bit 4 is specified. Not affected otherwise.

- E Cleared if bit 5 is specified. Not affected otherwise.
- L Cleared if bit 6 is specified. Not affected otherwise.
- S Cleared if bit 7 is specified. Not affected otherwise.

For other destination operands:

- C Set if bit tested is set. Cleared otherwise.
- V Not affected.
- Z Not affected.
- N Not affected.
- U Not affected.
- E Not affected.
- L According to the standard definition.
- S According to the standard definition.

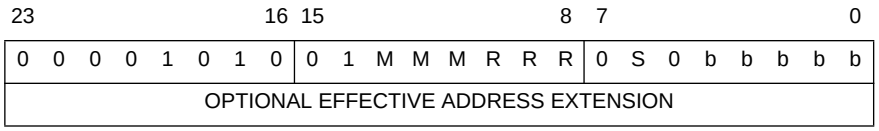
MR Status Bits:

For destination operand SR:

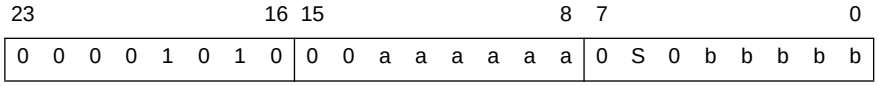
- I0 Changed if bit 8 is specified. Not affected otherwise
- I1 Changed if bit 9 is specified. Not affected otherwise
- So Changed if bit 10 is specified. Not affected otherwise
- S1 Changed if bit 11 is specified. Not affected otherwise
- RM Changed if bit 12 is specified. Not affected otherwise
- SB Changed if bit 13 is specified. Not affected otherwise
- DM Changed if bit 14 is specified. Not affected otherwise
- LF Changed if bit 15 is specified. Not affected otherwise

Instruction Formats and opcodes:

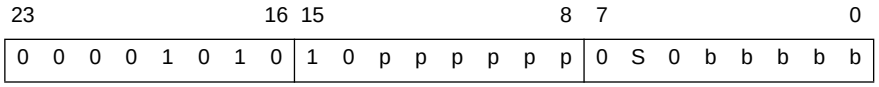
BCLR #n,[X or Y]:ea



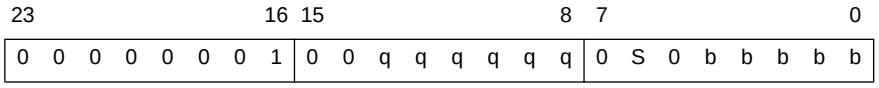
BCLR #n,[X or Y]:aa



BCLR #n,[X or Y]:pp



BCLR #n,[X or Y]:qq



BCLR #n,D

Instruction Fields:

{#n}	**bbbbb**	Bit number [0-23]
{ea}	**MMMRRR**	Effective Address (see Table A-16 on page A-241)
{X/Y}	**S**	Memory Space [X,Y] (see Table A-17 on page A-241)
{aa}	**aaaaaa**	Absolute Address [0-63]
{pp}	**pppppp**	I/O Short Address [64 addresses: $FFFFC0-$FFFFFF]
{qq}	**qqqqqq**	I/O Short Address [64 addresses: $FFFF80-$FFFFBF]
{D}	**DDDDDD**	Destination register [all on-chip registers] (see Table A-22 on page A-243)