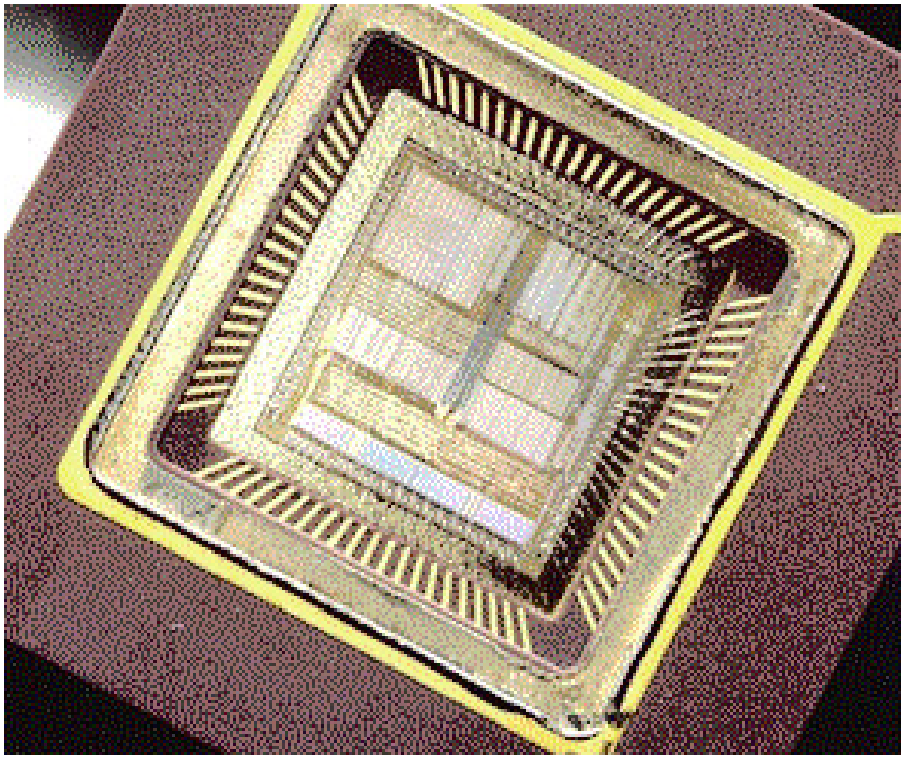
MIL-STD-1750A

Military Standard Sixteen-Bit Computer Instruction Set Architecture



www.xgc.com

MIL-STD-1750A

Military Standard Sixteen-Bit Computer Instruction Set Architecture

Order Number: XGC-MIL-STD-1750A-030122

XGC Technology

London
UK
Web: <www.xgc.com>

MIL-STD-1750A: Military Standard Sixteen-Bit Computer Instruction Set Architecture

Publication date January 2003

Abstract

This document is provided for use with XGC compilation systems targeted to the MIL-STD-1750A and specifies the 1750A instruction set and architecture.

The text of this document is based on MIL-STD-1750A, 2 Jul 1980, with updated Notice 1, 21 May 1982.

Contents

Figures

Tables

About This Document

This document contains the text of the military standard MIL-STD-1750A. This second edition is nearly complete, lacking only table V, which is too large to reproduce here. Tables X and XI are split into left and right halves.

This document is in no way intended to supersede the *MIL-STD-1750A Specification*, which is the definitive document describing the architecture of 1750 computers.

1. Reader's Comments

We welcome any comments and suggestions you have on this and other XGC manuals.

You can send your comments in the following ways:

- Internet electronic mail: `readers_comments@xgc.com`

Please include the following information along with your comments:

- The full title of the book.
- The section numbers and page numbers of the information on which you are commenting.
- The software version you are using.

Chapter 1

Scope and Purpose

1.1. Scope

This standard defines the instruction set architecture (ISA) for airborne computers. It does not define specific implementation details of a computer.

1.2. Purpose

The purpose of this document is to establish a uniform instruction set architecture for airborne computers which shall be used in Air Force avionic weapon systems.

1.3. Applicability

This standard is intended to be used to define only the ISA of airborne computers. System-unique requirements such as speed, weight, power, additional input/output commands, and

environmental operating characteristics are defined in the computer specification for each computer. Application is not restricted to any particular avionic function or specific hardware implementation by this standard. Generally, the ISA is applicable to, and shall be used for, computers that perform such functions as moderate accuracy navigation, computed air release points, weapon delivery, air rendezvous, stores management, aircraft guidance, and aircraft management. This standard is not restricted to implementations of "stand-alone" computers such as a mission computer or a fire control computer. Application to the entire range of avionics functions is encouraged such as stability and control, display processing and control, thrust management, and electrical power control.

1.4. Benefits

The expected benefits of this standard ISA are the use and re-use of available support software such as compilers and instruction level simulators. Other benefits may also be achieved such as: (a) reduction in total support software gained by the use of the standard ISA for two or more computers in a weapon system, and (b) software development independent of hardware development.

Chapter 2

Referenced Documents

Not applicable.

Chapter 3

Definitions

Accumulator	A register in the arithmetic logic unit used for intermediate storage, algebraic sums and other arithmetic and logical results.
Address	A number which identifies a location in memory where information is stored.
Arithmetic Logic Unit (ALU)	That portion of hardware in the central processing unit in which arithmetic and logical operations are performed.
Avionics	All the electronic and electro-mechanical systems and subsystems (hardware and software) installed in an aircraft or attached to it. Avionics systems interact with the crew or

	other aircraft systems in these functional areas: communications, navigation, weapons delivery, identification, instrumentation, electronic warfare, reconnaissance, flight control, engine control, power distribution, and support equipment.
Base Register	Any general register used to provide the base address portion of the derived address for instructions using the base relative or base relative-indexed addressing modes.
Bit	Contraction of binary digit; may be either zero or one. In information theory, a binary digit is equal to one binary decision or the designation of one of two possible values or states of anything used to store or convey information.
Byte	A group of eight binary digits.
Central Processing Unit (CPU)	That portion of a computer that controls and performs the execution of instructions.
Control Unit	That portion of hardware in the CPU that directs sequence of operations, interprets coded instructions, and initiates proper commands to other parts of the computer.
General Purpose Register	A register that may be used for arithmetic and logical operations, indexing, shifting, input, output,

	and general storage of temporary data.
Index Register	A register that contains a quantity for modification of an address without permanently modifying the address.
Input/Output (I/O)	That portion of a computer which interfaces to the external world.
Instruction	A program code which tells the computer what to do.
Instruction Counter (IC)	A register in the CPU that holds the address of the next instruction to be executed.
Instruction Set Architecture (ISA)	The attributes of a digital computer as seen by a machine (assembly) language programmer. ISA includes the processor and input/output instruction sets, their formats, operation codes, and addressing modes; memory management and partitioning if accessible to the machine language programmer; the speed of accessible clocks; interrupt structure; and the manner of use and format of all registers and memory locations that may be directly manipulated or tested by a machine language program. This definition excludes the time or speed of any operation, internal computer partitioning, electrical and physical organization, circuits and components of the computer, manufacturing technology, memory organization, memory

	cycle time, and memory bus widths.
Interrupt	A special control signal that suspends the normal flow of the processor operations and allows the processor to respond to a logically unrelated or unpredictable event.
Memory	That portion of a computer that holds data and instructions and from which they can be accessed at a later time.
Operation Code (OPCODE)	That part of an instruction that defines the machine operation to be performed.
Operand	That part of an instruction that specifies the address of the source, the address of the destination, or the data itself on which the processor is to operate.
Page Register	A register which is used to supply additional address bits in paged memory addressing schemes.
Programmed Input/Output (PIO)	A type of I/O channel that allows program control of information transfer between the computer and an external device.
Register	A device in the CPU for the temporary storage of one or more words to facilitate arithmetical, logical, or transfer operations.
Register Transfer Language (RTL)	A language used to describe operations (upon registers) which

	are caused by the execution of each instruction.
Reserved	Must not be used.
Spare	A framework for usage is defined by the standard with particulars to be defined by the application requirements.
Stack	A sequence of memory locations in which data may be stored and retrieved on a last-in-first-out (LIFO) basis.
Stack Pointer	A register that points to the last item on the stack.
Status Word Register	A register whose state is defined by some prior event occurrence in the computer.
Word	Sixteen bits.

Chapter 4

General Requirements

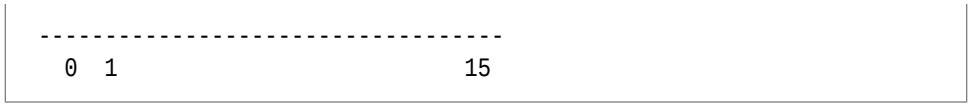
4.1. Data Formats

The instruction set shall support 16-bit fixed point single precision, 32-bit fixed point double precision, 32-bit floating point and 48-bit floating point extended precision data in 2's complement representation.

4.1.1. Single Precision Fixed Point Data

Single precision 16-bit fixed point data shall be represented as a 16-bit 2's complement integer number with the most significant bit (MSB) as the sign bit:

```
 MSB                             LSB
---------------------------------
| S|                              |
```



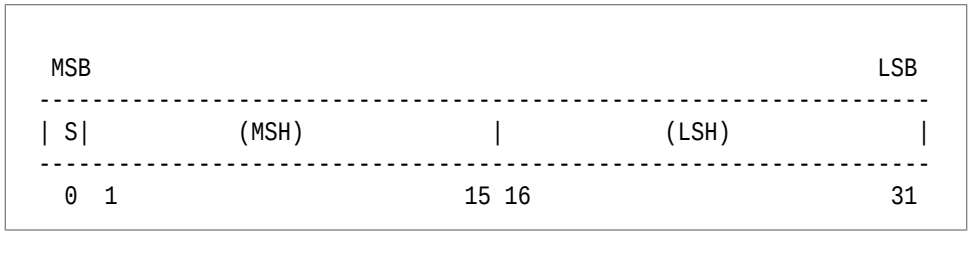
Examples of single precision fixed point numbers are shown in Table I, “Single Precision Fixed Point Numbers” [12].

Table I. Single Precision Fixed Point Numbers

Integer	16-Bit Hexadecimal Word
32767	7 F F F
16384	4 0 0 0
4096	1 0 0 0
2	0 0 0 2
1	0 0 0 1
-1	F F F F
-2	F F F E
-4096	F 0 0 0
-16384	C 0 0 0
-32767	8 0 0 1
-32768	8 0 0 0

4.1.2. Double Precision Fixed Point Data

Double precision 32-bit fixed point data shall be represented as a 32-bit 2's complement integer number with the most significant bit (MSB) of the first word as the sign bit.



Examples of machine representation for double precision fixed point numbers are shown in Table II, “Double Precision Fixed Point Numbers” [13].

Table II. Double Precision Fixed Point Numbers

Integer	32-Bit Hexadecimal Word
2,147,483,647	7 F F F F F F F
1,073,741,824	4 0 0 0 0 0 0 0
2	0 0 0 0 0 0 0 2
1	0 0 0 0 0 0 0 1
0	0 0 0 0 0 0 0 0
-1	F F F F F F F F
-2	F F F F F F F E
-1,073,741,825	C 0 0 0 0 0 0 0
-2,147,483,647	8 0 0 0 0 0 0 1
-2,147,483,648	8 0 0 0 0 0 0 0

4.1.3. Fixed Point Operands

All operands for fixed point adds, subtracts, multiplies and divides are integer. A fixed point overflow shall be defined as arithmetic overflow if the result is greater than $7FFF_{16}$ or less than 8000_{16} for single precision and greater than 7FFF $FFFF_{16}$ or less than 8000 0000_{16} for double precision.

4.1.4. Results on Fixed Point Overflow

On fixed point operations which cause overflow, the operation shall be performed to completion as if the MSBs are present and the 16 LSBs for single precision or the 32 LSBs for double precision shall be retained in the proper register(s). Division by zero shall produce a fixed point overflow and return results of all zeros.

4.1.5. Floating Point Data

Floating point data shall be represented as a 32-bit quantity consisting of a 24-bit 2's complement mantissa and an 8-bit 2's complement exponent.

Floating point numbers are represented as a fractional mantissa times 2 raised to the power of the exponent. All floating point numbers are assumed normalized or floating point zero at the beginning of a floating point operation and the results of all floating point operations are normalized (a normalized floating point number has the sign of the mantissa and the next bit of opposite value) or floating point zero. A floating point zero is defined as $0000\ 0000_{16}$, that is, a zero mantissa and a zero exponent (00_{16}). An extended floating point zero is defined as $0000\ 0000\ 0000_{16}$, that is, a zero mantissa and a zero exponent. Some examples of the machine representation for 32-bit floating point numbers are shown in Table III, “32-Bit Floating Point Numbers” [14].

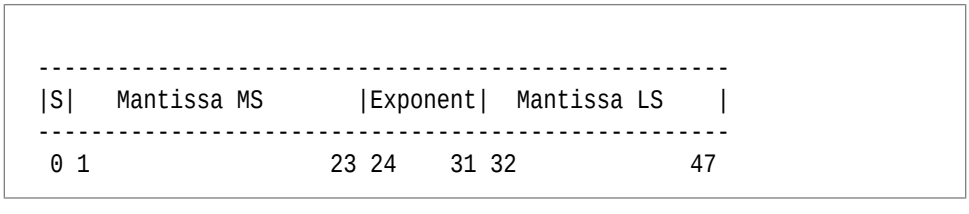
Table III. 32-Bit Floating Point Numbers

Decimal Number	Hexadecimal Notation
	Mantissa Exp
0.9999998×2^{127}	7FFF FF 7F
0.5×2^{127}	4000 00 7F
0.625×2^{4}	5000 00 04
0.5×2^{1}	4000 00 01
0.5×2^{0}	4000 00 00
0.5×2^{-1}	4000 00 FF
0.5×2^{-128}	4000 00 80
0.0×2^{0}	0000 00 00

Decimal Number	Hexadecimal Notation Mantissa Exp
-1.0×2^{0}	8000 00 00
$-0.5000001 \times 2^{-128}$	BFFF FF 80
-0.7500001×2^{4}	9FFF FF 04

4.1.6. Extended Precision Floating Point Data

Extended floating point data shall be represented as a 48-bit quantity consisting of a 40-bit 2's complement mantissa and an 8-bit 2's complement exponent. The exponent bits 24 to 31 lay between the split mantissa bits 0 to 23 and bits 32 to 47. The most significant bit of the mantissa is the sign bit 0, and the least significant bit of the mantissa is bit 47.



Some examples of the machine representation of 48-bit extended floating point numbers are shown in Table IV, “48-Bit Extended Floating Point Numbers” [15].

Table IV. 48-Bit Extended Floating Point Numbers

Decimal Number	Mantissa (MS)	Exp	Mantissa (LS)
0.5×2^{127}	400000	7F	0000
0.5×2^{0}	400000	00	0000
0.5×2^{-1}	400000	FF	0000
0.5×2^{-128}	400000	80	0000
-1.0×2^{127}	800000	7F	0000
-1.0×2^{0}	800000	00	0000
-1.0×2^{-1}	800000	FF	0000

Decimal Number	Mantissa (MS)	Exp	Mantissa (LS)
-1.0×2^{-128}	800000	80	0000
0.0×2^{0}	000000	00	0000
-0.75×2^{-1}	A00000	FF	0000

For both floating point and extended floating point numbers, an overflow is defined as an exponent overflow and an underflow is defined as an exponent underflow.

4.1.7. Floating Point Operands

All operands for floating point instructions must be normalized or a floating point zero. A floating point overflow shall be defined as exponent overflow if the exponent is greater than $7F_{16}$. The results of an operation which causes a floating point overflow shall be the largest positive number if the sign of the resulting mantissa was positive, or shall be the smallest negative number if the sign of the resulting mantissa was negative. Underflow shall be defined as exponent underflow if the exponent is less than 80_{16}. The results of an operation which causes a floating point underflow shall be floating point zero. Separate interrupts are set for overflow and underflow. Only the floating point instructions shall set the underflow interrupt.

4.1.8. Truncation of Floating Point Results

All floating point results shall be truncated toward negative infinity.

4.1.9. Results of Division

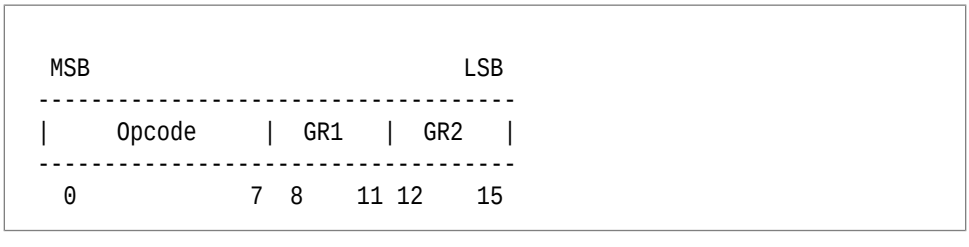
The sign of any non-zero remainder is the same as the dividend for all division instructions; the remainder is only accessible for single precision integer divides with 16 bit dividends and for single precision integer divides with 32 bit dividends.

4.2. Instruction Formats

Six basic instruction formats shall support 16 and 32-bit instructions. The operation code (opcode) shall normally consist of the 8 most significant bits of the instruction.

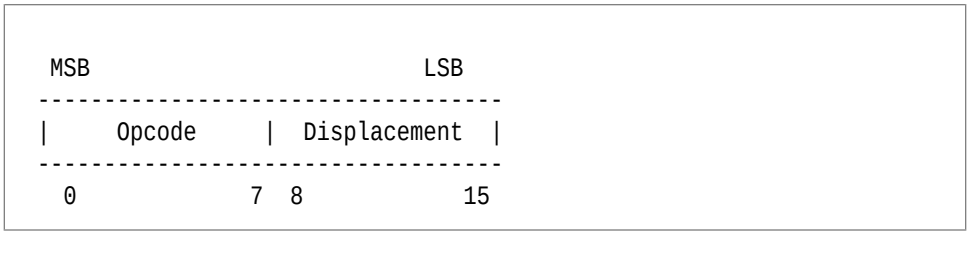
4.2.1. Register-to-Register Format

The register-to-register format is a 16-bit instruction consisting of an 8-bit opcode and two 4-bit general register (GR) fields that typically specify any of 16 general registers. In addition, these fields may contain a shift count, condition code, opcode extension, bit number, or the operand for immediate short instructions.



4.2.2. Instruction Counter Relative Format

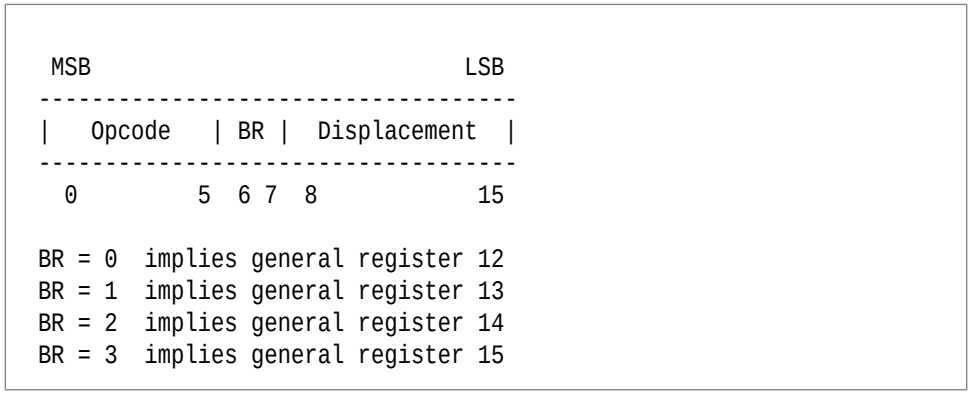
The Instruction Counter (IC) Relative Format is a 16-bit instruction consisting of an 8-bit opcode and an 8-bit displacement field.



4.2.3. Base Relative Format

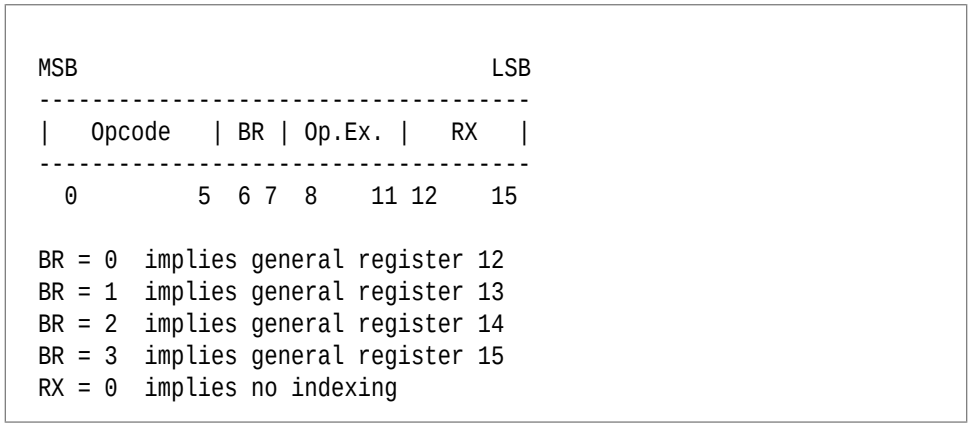
The base relative instruction format is a 16-bit instruction consisting of a 6-bit opcode, a 2-bit base register field and an 8-bit

displacement field. The base register (BR) field allows the designation of one of four different registers.



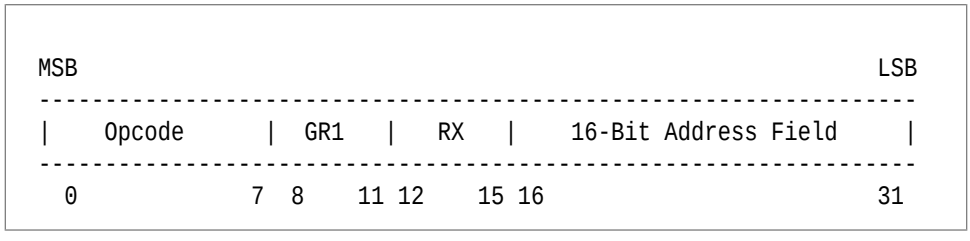
4.2.4. Base Relative Indexed Format

The base relative indexed instruction format is a 16-bit instruction consisting of a 6-bit opcode, a 2-bit base register field, a 4-bit opcode extension and a 4-bit index register field. The base register (BR) field allows the designation of one of four different base registers and the index register (RX) field allows the designation of one of fifteen different index registers.



4.2.5. Long Instruction Format

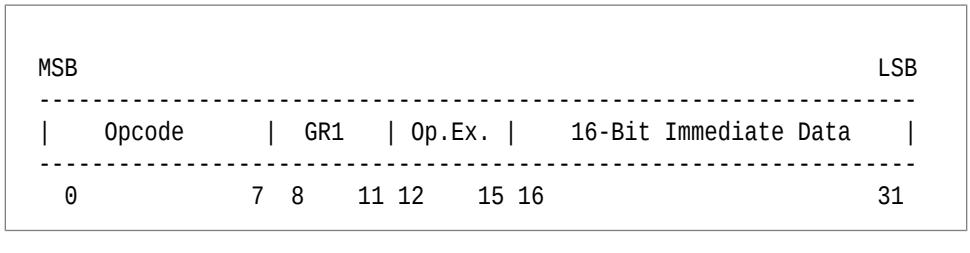
The Long Instruction Format is a 32-bit instruction consisting of an 8-bit opcode, a 4-bit general register field, a 4-bit index register field and a 16-bit address field.



Typically, GR1 is one of the 16 general registers on which the instruction is performing the operation. RX is one of the 15 general registers being used as an index register. The 16-bit address field is either a full 16-bit memory address or a 16-bit operand if the instruction specifies immediate addressing.

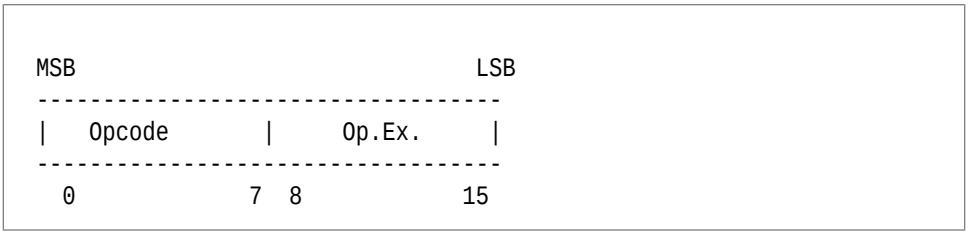
4.2.6. Immediate Opcode Extension Format

The immediate opcode extension format is a 32-bit instruction consisting of an 8-bit opcode, a 4-bit general register field, a 4-bit opcode extension and a 16-bit data field. Typically, GR1 is one of the 16 general registers on which the instruction is performing the operation. Op.Ex. is an opcode extension.



4.2.7. Special Format

The special instruction format is a 16-bit instruction consisting of an 8-bit opcode followed by an 8-bit opcode extension (Op.Ex.).



4.3. Addressing Modes

Table V, “Addressing Modes and Instruction Formats” [20] specifies the instruction word format, the Derived Address (DA), and the Derived Operand (DO) for each addressing mode that shall be implemented. The smallest addressable memory word is 16 bits: hence, the 16-bit address fields allow direct addressing of 64K (65,536) words. There is no restriction on the location of double word operands in memory.

Table V. Addressing Modes and Instruction Formats

TBS	See original MIL-STD-1750A

4.3.1. Register Direct (R)

An addressing mode in which the instruction specified register contains the required operand. (With the exception of this address mode, DA denotes a memory address.)

4.3.2. Memory Direct (D)

An addressing mode in which the instruction contains the memory address of the operand.

4.3.3. Memory Direct-Indexed (DX)

An addressing mode in which the memory address of the required operand is specified by the sum of the content of an index register and the instruction address field. Registers R1, R2, ..., R15 may be specified for indexing.

4.3.4. Memory Indirect (I)

An addressing mode in which the instruction specified memory address contains the address of the required operand.

4.3.5. Memory Indirect with Pre-Indexing (IX)

An addressing mode in which the sum of the content of a specified index register and the instruction address field is the address of the address of the required operand. Registers R1, R2, ..., R15 may be specified for pre-indexing.

4.3.6. Immediate Long (IM)

There shall be two methods of Immediate Long addressing: one which allows indexing and one which does not. The indexable form of immediate addressing is defined in Table V, “Addressing Modes and Instruction Formats” [20]. If the specified index register, RX, is not equal to zero, the content of RX is added to the immediate field to form the required operand; otherwise the immediate field contains the required operand.

4.3.7. Immediate Short (IS)

An addressing mode in which the required (4-bit) operand is contained within the (16-bit) instruction. There shall be two methods of Immediate Short addressing: one which interprets the content of the immediate field as positive data, and a second which interprets the content of immediate field as negative data.

4.3.7.1. Immediate Short Positive (ISP)

The immediate operand is treated as a positive integer between 1 and 16.

4.3.7.2. Immediate Short Negative (ISN)

The immediate operand is treated as a negative integer between 1 and 16. Its internal form shall be a 2's complement, sign-extended 16-bit number.

4.3.8. Instruction Counter Relative (ICR)

This addressing mode is used for 16-bit branch instructions. The contents of the instruction counter minus one (i.e., the address of the current instruction) is added to the sign extended 8-bit displacement field of the instruction. The sum points to the memory address to which control may be transferred if a branch is executed. This mode allows addressing within a memory region of 80_{16} to $7F_{16}$ words relative to the address of the current instruction.

4.3.9. Base Relative (B)

An addressing mode in which the content of an instruction specified base register is added to the 8-bit displacement field of the (16-bit) instruction. The displacement field is taken to be a positive number between 0 and 255. The sum points to the memory address of the required operand. This mode allows addressing within a memory region of 256 words beginning at the address pointed to by the base register.

4.3.10. Base Relative-Indexed (BX)

The sum of the contents of a specified index register and a specified base register is the address of the required operand. Registers R1, R2, ..., R15 may be specified for indexing.

4.3.11. Special (S)

The special addressing mode is used where none of the other addressing modes are applicable.

4.4. Registers and Support Features

4.4.1. General Registers

The instruction set shall support a minimum of 16 registers (R0 through R15). The registers may be used as accumulators, index registers, base registers, temporary operand memory, and stack pointers with the following restrictions:

- Only registers R1, R2, ..., R15 may be used as index registers (RX).
- Only four registers, R12, R13, R14, and R15 may be used as base registers for instructions having the Base Relative address mode.
- R15 is the implicit stack pointer for the Push and Pop Multiple instructions (Opcode $8F_{16}$ and $9F_{16}$).
- The general registers are not in the logical memory address space.
- Instructions having the Base Relative addressing mode have a single accumulator. The register pair (R0,R1) is the accumulator for double precision and floating point operations. Register R2 is the accumulator for single precision operations, except multiply and divide base relative also use R3.

The general registers shall functionally appear to be 16 bits in length. For instructions requiring a 32-bit operation, adjacent registers shall be concatenated to form effective 32-bit registers. Instructions requiring 48-bit operation shall concatenate three adjacent registers to form an effective 48-bit register.

When registers are concatenated, the register specified by the instruction shall represent the most significant word. The register set wraps around, that is, R15 concatenates with R0 for 32-bit operations and R15 concatenates with R0 and R1 for 48-bit operations.

4.4.2. Special Registers

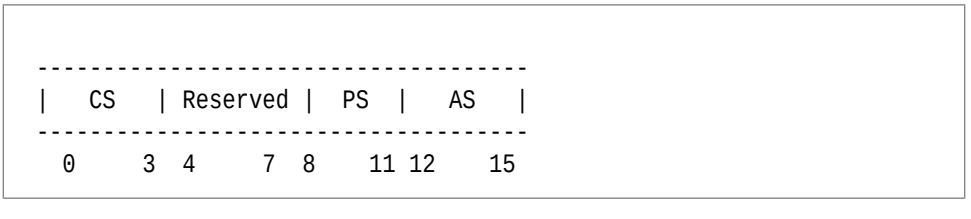
The instructions shall make use of the following special registers: instruction counter, status word, fault register, interrupt mask, pending interrupt register, and input/output interrupt code registers.

4.4.2.1. Instruction Counter (IC)

A 16-bit register used for program sequencing. It allows instructions within a range of 65,536 words to be executed. It is external to the general registers. It is saved in memory when an interrupt is serviced.

4.4.2.2. Status Word (SW)

The instruction set shall reference a 16-bit status word register whose state is defined by some prior event occurrence in the computer. The figure below indicates the format for the SW with the following paragraphs describing the meaning of the Condition Status (CS) field, reserved bits, the Processor State (PS) field, and the Address State (AS) field.



CS Bits:

A four-bit field (bits 0 through 3) of the status word shall be dedicated to instruction result (i.e., instruction status bits) and is defined as condition status (CS). Bits 0, 1, 2, and 3 shall be identified as C, P, Z, and N, respectively, and their meanings are given by the following register transfer description:

$C = (CS)_0 = 1$ if result generates a carry from an addition or no borrow from a subtraction

$P = (CS)_1 = 1$ if result is greater than (zero)

$Z = (CS)_2 = 1$ if result is equal to (zero)

$N = (CS)_3 = 1$ if result is less than (zero)

Reserved Bits:

Bits 4 through 7 of the status word shall be reserved.

PS Bits:

A four-bit field (bits 8 through 11) of the status word shall be dedicated to the processor state (PS) code. The code value defined by the PS shall be used for the following two functions:

For implementations which include the memory access lock feature of the expanded memory addressing option (see Section 4.5.2.2, “Page Register Word Format” [33]), PS shall define the memory access key code for all instructions and operand references to memory. References to memory during the interrupt recognition sequence for vector table pointer fetches and linkage/service parameter store/read references shall not use PS to define the memory access key code, but shall use an implied PS = 0 value.

PS shall determine the legal/illegal criteria for privileged instructions. When PS = 0 and a privileged instruction execution is attempted, the instruction shall be legal and shall be executed properly as defined. When PS /= 0 and a privileged instruction execution is attempted, the instruction shall be illegal, shall be aborted, and the privileged instruction fault bit in the fault register (FT_{10}) shall be set to one.

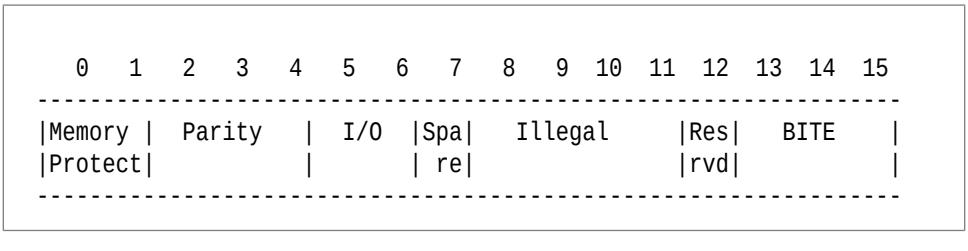
AS bits:

A four-bit field (bits 12 through 15) of the status word shall be dedicated to the address state (AS) code. For implementations which do not include the expanded memory addressing option, an address state fault shall be generated for any operation which attempts to modify AS to a non-zero value. For implementations which include the expanded memory

addressing option, AS shall define the group (pair) of page register sets to be used for all instruction and operand references to memory. References to memory during the interrupt recognition sequence for vector table pointer fetches and service parameter load references shall not use AS to define the operand page register set, but shall use an implied AS = 0 value. The linkage parameter store references shall use the AS field of the new status word. For partial implementations which include less than 16 groups of page register sets for the expanded memory addressing option (see Section 4.5.2.3, “Partial Implementations of Expanded Memory Addressing” [37]), the address state fault bit in the fault register (FT_{11}) shall be set to one if any operation attempts to establish an AS value that is not implemented.

4.4.2.3. Fault Register (FT)

The fault register is a 16-bit register used for indicating machine error conditions. The logical OR of the fault register bits is used to generate the machine error interrupt. The fault register shall be read and cleared by an XIO instruction. If a particular fault bit is not implemented, then the bit shall be set to zero. The fault bits shall be assigned as specified in the following:



The bits shall have the following meaning when set to one (1) :

Bit 0:
: CPU Memory Protection Fault. The CPU has encountered an access fault, write protect fault, or execute protect fault.

Bit 1:
: DMA Memory Protection Fault. A DMA device has encountered an access fault or a write protect fault.

Bit 2:
Memory Parity Fault.

Bit 3:
PIO Channel Parity Fault.

Bit 4:
DMA Channel Parity Fault.

Bit 5:
Illegal I/O Command Fault. An attempt has been made to execute an unimplemented or reserved I/O command.

Bit 6:
PIO Transmission Fault. Other I/O error checking devices, if used, may be ORed into this bit to indicate an error.

Bit 7:
Spare.

Bit 8:
Illegal Address Fault. A memory location has been addressed which is not physically present.

Bit 9:
Illegal Instruction Fault. An attempt has been made to execute a reserved code.

Bit 10:
Privileged Instruction Fault. An attempt has been made to execute a privileged instruction with PS /= 0.

Bit 11:
Address State Fault. An attempt has been made to establish an AS value for an unimplemented page register set.

Bit 12:
Reserved.

Bit 13:
Built-in Test Fault. Hardware built-in test equipment (BITE) error has been detected.

Bit 14-15:

Spare BITE. These bits are for use by the designer for future defining (coding, etc.) the BITE error which is detected. This can be used with Bit 13 to give a more complete error description.

4.4.2.4. Interrupt Mask (MK)

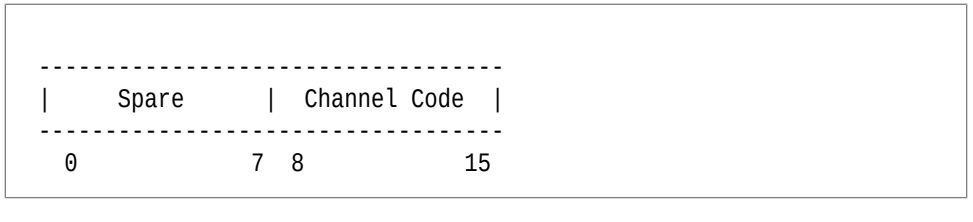
The interrupt mask register is software controlled and contains a mask bit for each of the system interrupts. The interrupt system is defined in Section 4.6, “Interrupt Control” [38].

4.4.2.5. Pending Interrupt Register (PI)

The pending interrupt request register is software and hardware controlled and contains the pending interrupts that are attempting to vector the instruction counter. A pending interrupt is set by a system interrupt signal. The pending interrupt bit that generates the interrupt request is cleared by hardware action during the interrupt processing prior to initiating software at the address defined by the new IC value. The register may be set, cleared, and read by the I/O instructions.

4.4.2.6. Input/Output Interrupt Code Registers (IOIC) (optional)

The input/output interrupt code registers, if implemented, are used to indicate which channel generated the input/output interrupt. One register is assigned for each of the two input/output interrupts. Each register is set by hardware to reflect the address of the highest priority channel requesting that level of interrupt. The address shall be 00_{16} for channel number 0, $0F_{16}$ for channel number 15, $7F_{16}$ for channel number 127, etc. The IOICs shall not be altered once the interrupt sequence has commenced until they are read by an I/O instruction.

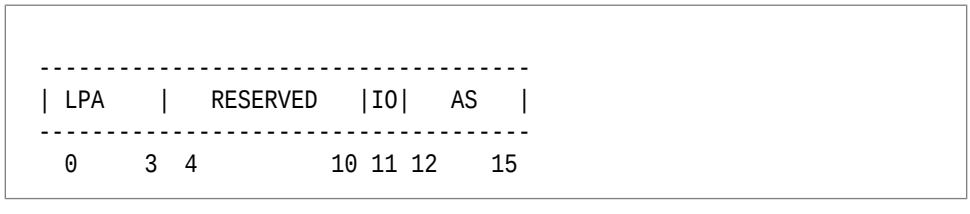


4.4.2.7. Page Registers (optional)

Up to 512 sixteen bit registers for optional expanded memory addressing.

4.4.2.8. Memory Fault Status Register (MFSR) (optional)

The memory fault status register provides the page register selection designators associated with memory faults. The page register designators (below) captured by the MFSR are valid for the memory reference causing the fault. The faults setting bits 0, 1, 2, or 8 of the Fault Register (FT) shall cause MFSR to be set.



LPA:
Address of page register within the set.

RESERVED:
Must not be used.

IO:
Instruction/operand page set selector (1 = instruction).

AS:
Address of selected group.

4.4.3. Stack

The instruction set shall support a stack mechanism. The operation of the stacking mechanism shall be such that the “last-in, first-out” concept is used for adding items to the stack and the Stack Pointer (SP) register always contains the memory address where the last item is stored on the stack. The stack provides for nested subroutine linkage using register 15. The stack shall also reside in a user defined memory area. Two instructions shall use register number

15 (R15) as the implied system stack pointer: Push Multiple registers, PSHM (see Section 5.54, “Push Multiple Registers onto the Stack” [106]), and Pop Multiple registers, POPM (see Section 5.44, “Pop Multiple Registers off the Stack” [98]). The stack expands linearly toward zero as items are added to it.

Two instructions, Stack IC and Jump to Subroutine, SJS (see Section 5.36, “Stack IC and Jump to Subroutine” [92]), and Unstack IC and Return from Subroutine, URS (see Section 5.37, “Unstack IC and Return from Subroutine” [92]), allow the programmer to specify any of the 16 general registers as the stack pointer. The memory block immediately preceding the stack area may be protected (by user using memory protect RAM), thus providing a means of knowing (memory protect interrupt) when the stack limit is exceeded. The stack shall be addressed by the Stack IC and Jump to Subroutine, Unstack IC and Return from Subroutine, Push Multiple, and Pop Multiple instructions.

4.4.4. Processor Initialization

4.4.4.1. Processor Reset State

Table VI, “Processor Reset State” [30] defines the processor reset state:

Table VI. Processor Reset State

Register/Device/Function	Condition After Reset
Instruction Counter	All zeros
Status Word	All zeros
Fault Register	All zeros
Pending Interrupt Register	All zeros
Interrupt Mask Register	All zeros
General Registers	Indeterminate
Interrupts	Disabled
Timers A & B	Started and all zeros [a]
Page Registers	Group 0 enabled [a]

Register/Device/Function	Condition After Reset
Page Registers AL Field	All zeros [a]
Page Registers W Field	Zero [a]
Page Registers E Field	Zero [a]
Page Registers PPA Field	Exact logical to physical [a]
Memory Protect RAM	Disabled and all zeros [ab]
Start Up ROM	Enabled [a]
DMA Enable	Disabled [a]
Input Discretes	Indeterminate [a]
Trigger Go Indicator	Started [a]
Discrete Outputs	All zeros [a]

[a] If implemented (optional)
[b] Main Memory Globally Protected

4.4.4.2. Power Up

Upon application of power, the processor shall enter the reset state, the normal power up discrete shall be set (if implemented), and execution shall begin.

4.4.5. Interval Timers (optional)

If implemented, then two interval timers shall be provided in the computer and shall be referred to as *Timer A* and *Timer B*. Both timers can be loaded, stopped, started, and read with the commands described in the XIO paragraph (see Section 5.1, “Execute Input/Output” [53]). The two timers shall be 16-bit counters which operate as follows. Effectively, a one is automatically added to the least significant bit of the timer. Bit fifteen is the least significant bit and shall represent the specified increment value of that timer, i.e., either 10 or 100 microseconds. An interrupt request is generated when a timer increments from FFFF to 0000_{16}. After power up, if the timers are not loaded by software, then an interrupt request is generated after 65,536 counts. A sample of the 16-bit counting sequence (shown in hex) is 0000, 0001, ..., 7FFF, 8000, ..., FFFF, 0000, ...,. At system reset or power up, the timers are initialized in accordance with Section 4.4.4.1, “Processor Reset State” [30]. The

timers are halted when a breakpoint, BPT (see Section 5.97, “Break Point” [151]), instruction is executed and the console is connected.

4.5. Memory

4.5.1. Memory Addressing

The instruction set shall use 16-bit logical addresses to provide for referencing of 65,536 words. When the expanded memory option (see Section 4.5.2, “Expanded Memory Addressing (optional)” [32]) is not implemented, physical addresses shall equal logical addresses.

4.5.1.1. Memory Addressing Arithmetic

Arithmetic performed on memory logical addresses shall be modulo 65,536 such that references to the maximum logical address of $FFFF_{16}$ plus 1 shall be to logical address 0000_{16}.

4.5.1.2. Memory Addressing Boundary Constraints

There shall be no odd or even memory address boundary constraints.

4.5.2. Expanded Memory Addressing (optional)

If used, then expanded memory addressing shall be performed via a memory paging scheme as depicted in Figure 1, “Expanded Memory Mapping Diagram” [36]. There shall be a maximum of 512 page registers in the page file (not in logical memory space). These shall functionally be partitioned into 16 groups with 2 sets per group and 16 page registers per set. Within a group, one set shall be designated for instruction references and the other set for operand references. The page size shall be 4096 words such that one set of 16 page registers shall be capable of mapping 65,536 words defined by a 16-bit logical address. The page group shall be selected by the 4-bit Address State (AS) field of the Status Word (SW). The instruction/operand set within the group shall be selected by the hardware that differentiates between instruction and operand

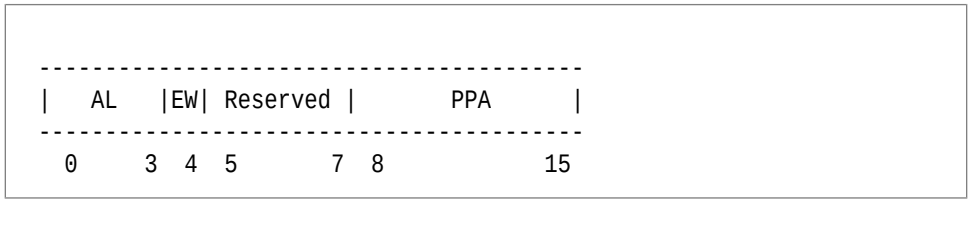
memory references. The 4 most significant bits of any 16-bit logical address shall select the page register within that set. The 8-bit Physical Page Address (PPA) within the page register shall be concatenated with the 12 least significant bits of the logical address to form a 20-bit physical address, allowing addressing of 1,048,576 words of physical memory. If expanded memory addressing is implemented, then devices other than the CPU which access memory may utilize either an unmapped 20-bit physical address or a mapped 16-bit logical address. If the devices other than the CPU which access memory utilize 16-bit addressing, a separate address state value must be provided.

4.5.2.1. Group Selection

During instruction and operand references to memory, the address state (AS) field of the status word shall be used to designate the page file group. During an interrupt recognition sequence, the operand set of group zero shall be used for vector table and service pointer references to memory; while the linkage pointer references to memory shall use the operand set specified by the AS of the new status word. During memory accesses by devices other than the CPU which utilize 16-bit logical addressing, the address state value provided by the device shall be used to designate the page register group. Device accesses shall utilize the operand set of the selected group.

4.5.2.2. Page Register Word Format

Each page register shall be 16 bits. The figure below indicates the format for the page register words with the following paragraphs describing the meaning of the access lock (AL) field, the execute protect (E) bit, the write protect (W) bit, reserved bits, and the Physical Page Address (PPA) field.



AL Field:

The access lock and key feature is optional if expanded memory addressing is implemented. If the access lock and key feature is not implemented, then the AL field shall always be zero. If it is implemented, then a 4-bit field (bits 0 through 3) of each page register shall contain the access lock (AL) code for the associated page register, which shall be used with the access key codes to determine access permission. If the access lock and key feature is implemented, the access key code is normally supplied by the PS field of the status word. However, during memory accesses by devices other than a CPU which utilize 16-bit logical addressing, the access code must be supplied by the device.

For each of the possible 16 values of the AL code, access shall be permitted for the reference according to Table VII, “AL Code to Access Key Mapping” [35]. References supplying an unacceptable access key code shall not modify any memory location or general registers and an access fault shall be generated. An access fault resulting from a CPU reference attempt shall set fault register bit 0 to cause a machine error interrupt. An access fault resulting from a DMA attempt shall set fault register bit 1 to cause a machine error interrupt. Note that the access lock and key codes defined in the above table have the following characteristics:

- An access lock code of F_{16} is an "unlocked" lock code and allows any and all access key codes to be acceptable.
- An access key code of 0 is a "master" key code and is acceptable to any and all access lock codes.
- Access key codes 1 through E_{16} are acceptable to only their own "matched" lock code or the "unlocked" lock code of F_{16}.
- An access key code of F_{16} is acceptable to only the “unlocked” lock code of F_{16}.

E Bit:

For instruction page register sets only, bit 4 shall be defined as the E bit and shall determine the acceptable/unacceptable

criteria for read references for instruction fetches. When E=1, any attempted instruction read reference designating that associated page register shall be terminated and an execute protect fault shall be generated. An execute protect fault shall set fault register bit 0 to cause a machine error interrupt.

W Bit:

For operand page registers only, bit 4 shall be defined as the W bit and shall determine the acceptable/unacceptable criteria for write references. When W=1, any attempted write reference designating that associated page register shall not modify any memory location and a write protect fault shall be generated. A write protect fault resulting from a CPU reference attempt shall set fault register bit 0 to cause a machine error interrupt. A write protect fault resulting from a DMA reference attempt shall set fault register bit 1 to cause a machine error interrupt.

Reserved Bits:

Bits 5 through 7 of all the page registers shall be reserved and shall always be 0.

PPA Field:

An eight-bit field (bits 8 through 15) of each page register shall be dedicated to the physical page address which is used to define the physical address as depicted in Figure 1, “Expanded Memory Mapping Diagram” [36].

Table VII. AL Code to Access Key Mapping

AL Code	Acceptable Access Key Codes
0	0
1	0,1
2	0,2
3	0,3
4	0,4
5	0,5
6	0,6
7	0,7
8	0,8

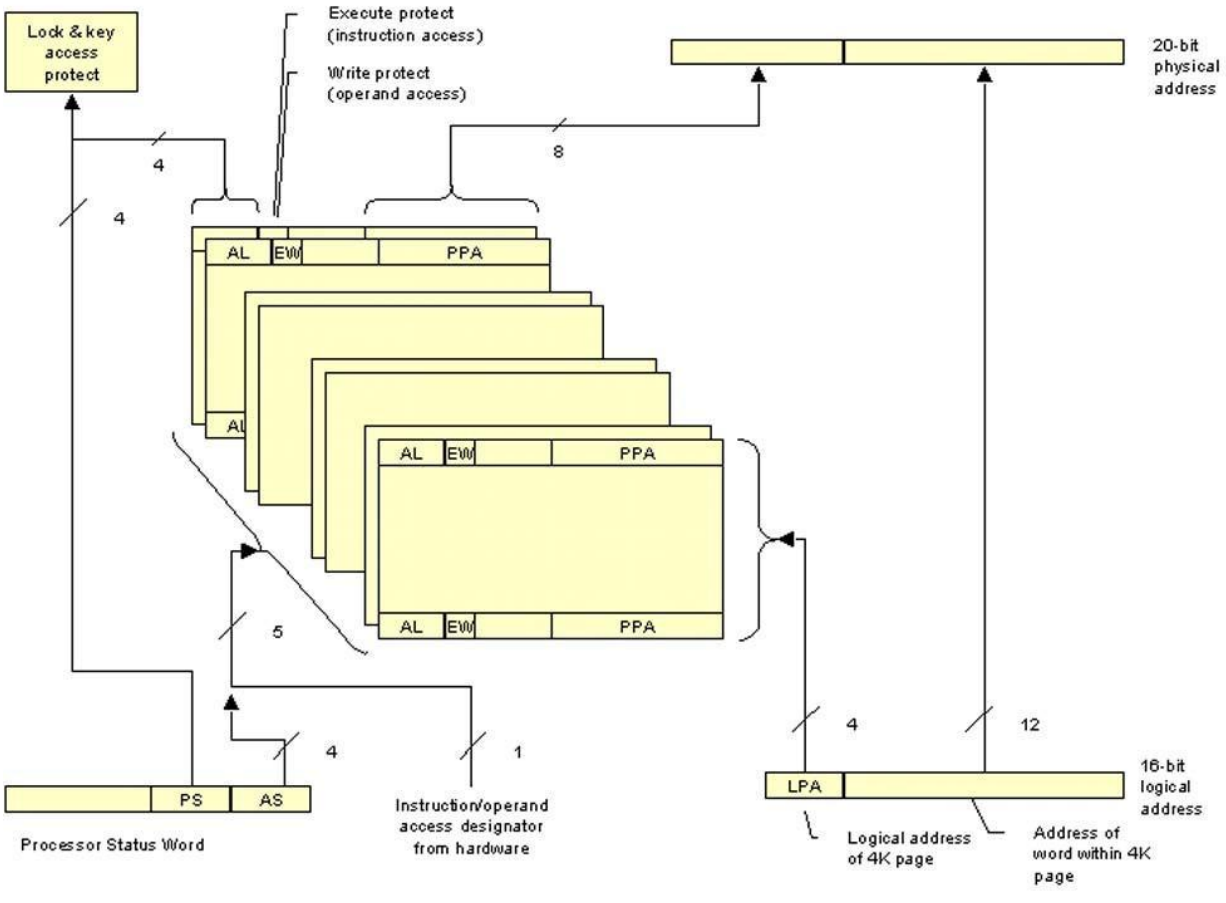


Figure 1. Expanded Memory Mapping Diagram

AL Code	Acceptable Access Key Codes
9	0,9
A	0,A
B	0,B
C	0,C
D	0,D
E	0,E
F	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

4.5.2.3. Partial Implementations of Expanded Memory Addressing

A given implementation of this standard may include a partial implementation of the expanded addressing option. That partial implementation may use 2, 4, or 8 groups of page registers as follows:

Number of Groups	AS Group Codes
2	0 and 1
4	0 through 3
8	0 through 7

Within any full or partial implementation, the lock feature may or may not be included.

4.5.3. Memory Parity (optional)

If used, then bit 2 in the fault register shall be set to indicate a memory parity error.

4.5.4. Memory Block Protect (optional)

If used, shall be as described by the input/output instructions. For operations which contain multiple memory references, each store operation shall be as defined by the memory protection for that specific memory address.

4.5.5. References to Unimplemented Memory

Attempted access to physical addresses which are not implemented shall generate an illegal address fault and shall cause the referencing action to terminate. An illegal address fault shall set fault register bit 8 to cause a machine error interrupt.

4.5.6. Start up ROM (optional)

If used, the start up read only memory (ROM) address range shall be contiguous starting from physical address 0 up to a maximum of 65,536, as required by the system application. When the start up ROM is enabled, if an I/O or CPU store function is executed whose address is within the start up ROM, then the store is attempted into the main memory. When the start up ROM is enabled, if a read function (instruction or operand) is executed from either I/O or the CPU whose address is to the start up ROM, then the read shall be from the start up ROM. When disabled, the start up ROM cannot be accessed.

4.5.7. Reserved Memory Locations

Locations 2 through $1F_{16}$ are reserved. Locations 20_{16} through $3F_{16}$ are used by the hardware and the stored program as defined by Table VIII, “Interrupt Definitions” [39].

4.6. Interrupt Control

4.6.1. Interrupts

The instruction set shall support a minimum of sixteen (16) interrupts as shown in Table VIII, “Interrupt Definitions” [39]. An interrupt request may occur at any time; however, the interrupt processing must wait until the current instruction is completed. An exception to this is the Move Multiple Word which may be interrupted after each single word transfer. The overall procedure for acceptance of, responding to, and processing of an interrupt shall be as illustrated by the flow chart of Figure 2, “Interrupt System Flowchart” [51].

4.6.1.1. Interrupt Acceptance

The interrupt system shall have the capability to accept external and internal interrupts. Figure 2, “Interrupt System Flowchart” [51] indicates the relationship between the interrupt signals, the pending

interrupt register, the interrupt signals and the fundamental communications between the interrupt system and the CPU.

4.6.1.2. Interrupt Software Control

Software shall be able to input from or output to the interrupt mask register as well as the pending interrupt register. Also, software shall be able to disallow recognition of interrupts via the “disable interrupts” signal (without inhibiting interrupt acceptance into the pending interrupt register) and to allow recognition of interrupts via the "enable interrupts" signal. The disabling shall not allow any interrupt after the beginning of the disable instruction. The CPU's interrupt service hardware shall continue to “disable interrupts” for one instruction after the Enable Interrupts instruction has completed. Full descriptions of the interrupt instructions are given in the input/output instruction repertoire.

Table VIII. Interrupt Definitions

Interrupt Number	Interrupt Mask Bit Number	Interrupt Linkage Pointer Address (Hex)	Interrupt Service Pointer Address (Hex)	
0	0	20	21	Power Down (cannot be masked or disabled)
1	1	22	23	Machine Error (cannot be disabled)
2	2	24	25	Spare
3	3	26	27	Floating Point Overflow
4	4	28	29	Fixed Point Overflow
5	5	2A	2B	Executive Call (cannot be masked or disabled)
6	6	2C	2D	Floating Point Underflow
7	7	2E	2F	Timer A (if implemented)
8	8	30	31	Spare
9	9	32	33	Timer B (if implemented)
10	10	34	35	Spare

Interrupt Number	Interrupt Mask Bit Number	Interrupt Linkage Pointer Address (Hex)	Interrupt Service Pointer Address (Hex)	
11	11	36	37	Spare
12	12	38	39	Input/Output Level 1 (if implemented)
13	13	3A	3B	Spare
14	14	3C	3D	Input/Output Level 2 (if implemented)
15	15	3E	3F	Spare

Note Interrupt number 0 has the highest priority. Priority decreases with increasing interrupt number.

4.6.1.3. Interrupt Priority Definitions

The priority definitions of the interrupts and their required relationship to the interrupt mask and interrupt pointer addresses are illustrated in Table VIII, “Interrupt Definitions” [39], Interrupt Definitions. The power down interrupt shall initiate the power down sequence and cannot be masked or disabled during normal operation of the computer. The executive call interrupt, used with the Branch to Executive instruction, BEX, (see Section 5.30, “Branch to Executive” [87]) also cannot be masked or disabled. The machine error interrupt cannot be disabled but can be masked during normal operation of the computer. All other interrupts can be disabled and masked. If a floating point overflow/underflow or fixed point overflow condition occurs, then the instruction generating that condition shall be interrupted at its completion if the interrupt is unmasked and enabled.

4.6.1.4. Interrupt Vectoring Mechanism

The vectoring mechanism shall be as illustrated on Figure 3, “Interrupt Vectoring System” [51]. For each interrupt there shall be two fixed memory locations in the “vector table”: (1) the first memory location (Linkage Pointer) shall be defined as the address

of where to store the current (old) state of the computer (i.e., “old interrupt mask”,“old status word”, and “old instruction counter”); and (2) the second memory location (Service Pointer) shall be defined as the address of the next (new) state of the computer (i.e., “new interrupt mask”, “new status word”, and “new instruction counter”). Returning from interrupts may be accomplished by executing the Load Status (LST/LSTI) instruction with the value/address of the Linkage Pointer for an address field.

4.7. Input/Output

In conjunction with the spare command codes, the I/O interrupts, and the I/O interrupt code registers, the I/O instructions provide a framework within which the user can implement his system interfaces. The particulars of the system interfaces outside of this framework (such as dedicated memory locations, channel register definitions, command code assignments/definitions, multiple channel priorities, page register access, etc.) are not included in this standard.

4.7.1. Input

The input instructions transfer data from an external I/O device or an internal special register to a CPU general register. This command is used to read data from peripheral devices, timers, status word, fault register, discretes, interrupt mask, etc. A full description of the input instructions is given in the instruction repertoire.

4.7.2. Output

The output instructions transfer data from a CPU general register to an external I/O device or special register. This command is used to write data to peripheral devices, discretes, start and stop timers, enable and disable interrupts and DMA, set and clear interrupt requests, masks and pending interrupt bits, etc. A full description of the output instructions is given in the instruction repertoire.

4.7.3. Input/Output Commands

Input/output commands are classified as mandatory, optional, reserved, or spare. Mandatory I/O commands must be implemented as defined. Optional I/O commands must be implemented as defined, if implemented. Reserved I/O commands must not be implemented. Spare I/O commands may be implemented as required by the application. Attempted execution of an unimplemented optional or spare I/O command or a reserved I/O command shall cause the illegal I/O command fault to be set in the fault register (FT_5) causing a machine error interrupt.

Input/output command words shall be fully decoded. "TBDs" in input/output instruction descriptions refer to parameters to be determined by the application system requirements. Within these classifications, the use of the command is defined in the instruction description.

4.7.4. Input/Output Command Partitioning

The I/O command space shall be divided into 128 channels. Up to 512 commands within each channel group (256 input and 256 output) may be used with each I/O interface. Table IX, “Input/Output Channel Groups” [44] lists the 128 I/O channel groups. The attempted execution of an unimplemented I/O command shall cause bit 5 of the fault register to be set, generate a machine error interrupt, and abort to completion.

4.7.5. Input/Output Interrupts (optional)

Input/output level 1 and level 2 interrupts are available to the user. Either interrupt level or both may be implemented for an interface as defined by the particular application specification. The interrupts shall be used in conjunction with the input/output interrupt code registers to provide I/O channel to process communications. Two levels of interrupts allow easy differentiation of normal reporting from error reporting.

4.7.6. Dedicated I/O Memory Locations

If dedicated memory locations are used to communicate information to and/or from an I/O channel, these locations shall be consecutive memory locations starting at an implementation defined location. Locations 40_{16} through $4F_{16}$ are optional for I/O usage.

4.8. Instructions

4.8.1. Invalid Instructions

Attempted execution of an instruction whose first 16 bits are not defined by this standard shall cause the invalid instruction bit in the fault register (FT_9) to be set, generating a machine error interrupt. The Built-In-Function is an exception; implemented Built-In-Functions do not cause FT_9 to be set or the machine error interrupt to be generated. All undefined bit patterns in the first 16 bits of an instruction are reserved.

4.8.2. Mnemonic Conventions

Each instruction has an associated mnemonic convention. In general, the operation is one or two letters, e.g., L for *l*oad, A for *a*dd, ST for *st*ore.

Floating point operations have a prefix of F, e.g., FL for *f*loating *l*oad, FA for floating add.

Double precision operations have a prefix of D, e.g., DL for *d*ouble *l*oad, DA for *d*ouble *a*dd.

Extended precision floating point operations have a prefix of EF, e.g., EFA for extended precision floating point add.

Register-to-register operations have a suffix of R, e.g., AR for single precision add register-to-register, FAR for floating add register-to-register.

Indirect memory reference is indicated by a suffix I, e.g., LI for *L*oad *I*ndirect.

Immediate addressing, using the address field as an operand, is indicated by a suffix of IM, e.g., AIM for single precision *a*dd *im*mediate. Use of indexing is specified in assembly language by the occurrence of the operational field after the address field, e.g., FA A2,ALPHA,A5: floating add to register A2 from memory location ALPHA indexed by register A5.

Table IX. Input/Output Channel Groups

Output	Input	Usage
00XX	80XX	PIO
03XX	83XX	PIO
04XX	84XX	Spare
1FXX	9FXX	Spare
20XX	A0XX	Processor & Auxiliary Register Control
21XX	A1XX	Reserved
2FXX	AFXX	Reserved
30XX	B0XX	Spare
3FXX	BFXX	Spare
40XX	C0XX	Processor & Auxiliary Register Control
41XX	C1XX	Reserved
4FXX	CFXX	Reserved
50XX	D0XX	Memory Protect RAM
51XX	D1XX	Memory Address Extension (page register commands)
52XX	D2XX	Memory Address Extension (page register commands)
53XX	D3XX	Spare
7FXX	FFXX	Spare

4.8.3. Instruction Matrix

Table X, “Operation Code Matrix (Left)” [48] contains the order type matrix which relates each instruction operation code to an assigned symbol. The numbers shown across the top of the matrix are hexadecimal numbers which represent the higher order four bits of the operation code, and the hexadecimal numbers along the left side represent the lower order four bits of the operation code. Table XI, “Extended Operation Codes (Left)” [50] contains the order types and assigned mnemonics for the extended Operation Code instructions.

4.8.4. Instruction Set Notation

The text and register transfer descriptions are intended to complement each other. Ambiguities or omissions in one are resolved by the other. The following definitions and special symbols are associated with the instruction descriptions.

CPU Registers	
R0, R1, ..., R15	The 16, 16-bit general registers
IC	Instruction Counter
SW	Status Word
CS	Condition Status. A 4-bit quantity that is set according to the result of instruction executions.
LP	Linkage Pointer
SP	Stack Pointer; R15 for the Push and Pop Multiple instructions
SVP	Service Pointer
MK	Interrupt Mask Register
PI	Pending Interrupt Register
RA, RB	An unspecified general register
Addressing Modes	
R	Register Direct

D, DX	Memory Direct, Memory Direct-Indexed
I, IX	Memory Indirect, Memory Indirect with Pre-Indexing
IM, IMX	Immediate Long, Immediate Long with Indexing
ISP, ISN	Immediate Short with Positive Operand, Immediate Short with Negative Operand
ICR	IC-Relative
B, BX	Base Relative, Base Relative with Indexing
S	Special
Data Quantities	
MSH,LSH	Most Significant Half, Least Significant Half
MSB,LSB	Most Significant Bit, Least Significant Bit
S.P., D.P., Ft.P., E.F.P	Abbreviation for “Single Precision,” “Double Precision,” “Floating Point,” and “Extended Floating Point” operations respectively.
MO	Floating Point Derived Operand mantissa (fractional part): $DO_{0\text{-}23}$ (Ft.P), $DO_{0\text{-}23}$ $DO_{32\text{-}47}$ (E.F.P.)
EO	Floating point 8-bit 2's complement Derived Operand characteristic (exponent): $DO_{24\text{-}31}$
MA	Floating point register accumulator mantissa (fractional part): $(RA,RA+1)_{0\text{-}23}$ (Ft.P.), $(RA,RA+1)_{0\text{-}23}$ $(RA+2)_{32\text{-}47}$ (E.F.P.)
EA	Floating point 8-bit 2's complement register accumulator characteristic (exponent): $(RA,RA+1)_{24\text{-}31}$
RQ, MP, MQ	An entity used for register level transfer description clarification. These registers are not part of the general register file.
Miscellaneous	
(X)	Contents of Register X

(X,X+1)	Contents of concatenated Registers X and X+1
[X]	Contents of memory address X
[X,X+1]	Contents of sequential memory locations X and X+1
OVM	Mantissa (fractional part) overflow
Exit	Indicates termination of present register transfer operation (except the setting of the CS bits)
DA	Derived Address
DO	Derived Operand
N,M,n	An integer number
DSPL	Displacement
X_n	If X is a CPU register or a data quantity (see above), then n specifies a bit position in X. If X is not a CPU register or a data quantity, then the number X is to the base n. If X is a number and n=16, then X is a 2's complement hexadecimal number.
X^i	If X is a CPU register or a memory address, then i specifies the state of X. This notation is used in the register transfer descriptions to refer to the contents of a CPU register or a memory address at different times (states) of the execution of the instruction. If X is not a CPU register or a memory address, then the number X is raised to the ith power.

Symbols	
<--	Unilateral transfer designator
<-->	Bilateral transfer designator
:	Comparison Designator
x	Indicates a “don't care” bit when used in a binary number
>	Greater than
<	Less than

=	Equals
>=	Greater than or equal
<=	Less than or equal
^	Logical AND
v	Logical OR
xor	Exclusive OR
~	Logical NOT
||	Absolute value

Table X. Operation Code Matrix (Left)

	Load Store	Integer Arithmetic	Floating Point	Logic Compare	Opcode Extensions	Bit	Shift	Jump
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	LB BR12	AB BR12	FAB BR12	ORB BR12	BRX BR12[a]	SB	SLL	JC
1	LB BR13	AB BR13	FAB BR13	ORB BR13	BRX BR13[a]	SBR	SRL	CR
2	LB BR14	AB BR14	FAB BR14	ORB BR14	BRX BR14[a]	SBI	SRA	CISP
3	LB BR15	AB BR15	FAB BR15	ORB BR15	BRX BR15[a]	RB	SLC	CISM
4	DLB BR12	SBB BR12	FSB BR12	ANDB BR12		RBR		CBL
5	DLB BR13	SBB BR13	FSB BR13	ANDB BR13		RBI	DSLL	
6	DLB BR14	SBB BR14	FSB BR14	ANDB BR14		TB	DSRL	DC
7	DLB BR15	SBB BR15	FSB BR15	ANDB BR15		TBR	DSRA	DCR
8	STB BR12	MB BR12	FMB BR12	CB BR12	XIO[ab]	TBI	DSLC	FC
9	STB BR13	MB BR13	FMB BR13	CB BR13	VIO[ab]	TSB		FCR

	Load Store	Integer Arithmetic	Floating Point	Logic Compare	Opcode Extensions	Bit	Shift	Jump
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
A	STB BR14	MB BR14	FMB BR14	CB BR14	IMML	SVBR	SLR	EFC
B	STB BR15	MB BR15	FMB BR15	CB BR15			SAR	EFCR
C	DSTB BR12	DB BR12	FDB BR12	FCB BR12		RVBR	SCR	LSTI[b]
D	DSTB BR13	DB BR13	FDB BR13	FCB BR13			DSLR	LST[b]
E	DSTB BR14	DB BR14	FDB BR14	FCB BR14		TVBR	DSAR	SJS
F	DSTB BR15	DB BR15	FDB BR15	FCB BR15	BIF[c]		DSCR	URS

[a]These order types represent instructions which have “extended” operation codes and are fully described in the instruction specifications and in Table V, “Addressing Modes and Instruction Formats” [20].

[b]Privileged instructions

[c]User Defined Built-In Function Opcode.

Table Xr. Operation Code Matrix (Right)

	Load	Store	Add	Sub	Mult	Divide	Logical	Compare
	8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	L	ST	A	S	MS	DV	OR	C
1	LR	STC	AR	SR	MSR	DVR	ORR	CR
2	LISP	STCI	AISP	SISP	MISP	DISP	AND	CISP
3	LISN	MOV	INCM	DECM	MISN	DISN	ANDR	CISM
4	LI	STI	ABS	NEG	M	D	XOR	CBL
5	LIM		DABS	DNEG	MR	DR	XORR	
6	DL	DST	DA	DS	DM	DD	N	DC
7	DLR	SRM	DAR	DSR	DMR	DDR	NR	DCR
8	DLI	DSTI	FA	FS	FM	FD	FLX	FC
9	LM	STM	FAR	FSR	FMR	FDR	FLT	FCR
A	EFL	EFST	EFA	EFS	EFM	EFD	EFLX	EFC

Load	Store	Add	Sub	Mult	Divide	Logical	Compare
8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
B LUB	STUB	EFAR	EFSR	EFMR	EFDR	EFLT	EFCR
C LLB	SLTB	FABS	FNEG			XBR	
D LUBI	SUBI					XWR	
E LLBI	SLBI						
F POPM	PSHM						NOP

Table XI. Extended Operation Codes (Left)

MSH[a]	Format[b]	0	1	2	3	4	5	6	7
40	BRX BR12	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX
41	BRX BR13	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX
42	BRX BR14	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX
43	BRX BR15	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX
4A	IMM		AIM	SIM	MIM	MSIM	DIM	DVIM	ANDM

[a]Most Significant Half

[b]Base Relative Indexed Format

Table XIr. Extended Operation Codes (Right)

MSH[a]	Format[b]	8	9	A	B	C	D	E	F
40	BRX BR12	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
41	BRX BR13	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
42	BRX BR14	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
43	BRX BR15	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
4A	IMM	ORIM	XORM	CIM	NIM				

[a]Most Significant Half

[b]Base Relative Indexed Format

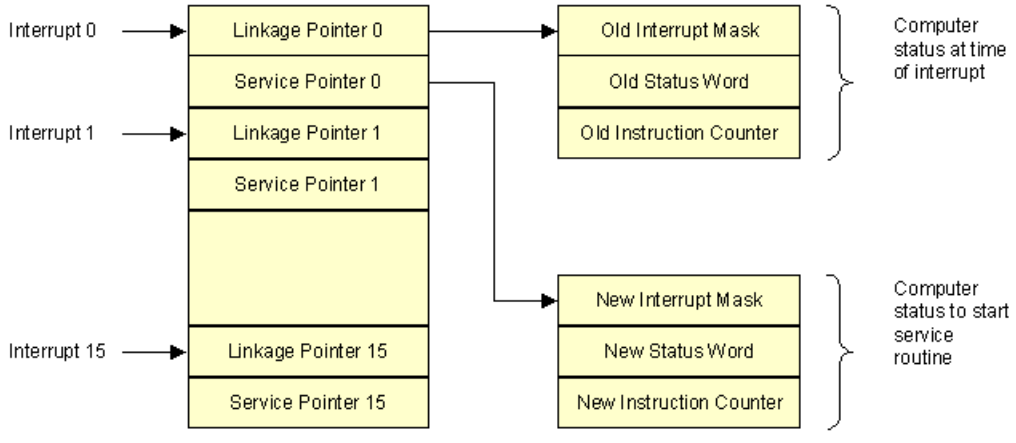


Figure 3. Interrupt Vectoring System

Chapter 5

Detailed Requirements

5.1. Execute Input/Output

```
Addr
Mode  Mnemonic          Format/Opcode
                            8      4      4           16
IM    XIO  RA,CMD       ---------------------------------------
IMX   XIO  RA,CMD,RX    |  48  |  RA  |  RX  |  |    CMD     |
                        ---------------------------------------
```

Description. The input/output instruction transfers data between an external/internal device and the register RA. The Derived Operand, DO, specifies the operation to be performed or the device to be addressed. The immediate operand field may be viewed as an operation code extension field. Note that if indexing is specified, then the input/output operation or device address is formed by summing the contents of the register RX and the immediate field. This is a privileged instruction.

The mandatory and optional input/output immediate command fields are listed below.

Table XII. Mandatory XIO Command Fields and Mnemonics

Code	Mnemonic	Description
0YXX	PO	Programmed Output: This command outputs 16 bits of data from RA to a programmed I/O port. Y may be from 0 through 3.
2000	SMK	Set Interrupt Mask: This command outputs the 16-bit contents of the register RA to the interrupt mask register. A "1" in the corresponding bit position allows the interrupt to occur and a "0" prevents the interrupt from occurring except for those interrupts that are defined such that they cannot be masked.
2001	CLIR	Clear Interrupt Request: All interrupts are cleared (i.e., the pending interrupt register is cleared to all zeros) and the contents of the fault register are reset to zero.
2002	ENBL	Enable Interrupts: This command enables all interrupts which are not masked out. The enable operation takes place after execution of the next instruction.
2003	DSBL	Disable Interrupts: This command disables all interrupts (except those that are defined such that they cannot be disabled) at the beginning of the execution of the DSBL instruction.
2004	RPI	Reset Pending Interrupt: The individual interrupt bit to be reset shall be designated in register RA as a right justified four bit code. (0_{16} represents interrupt number 0, F_{16} represents interrupt number 15). If interrupt 1_{16} is to be cleared, then the contents of the fault register shall also be set to zero.
2005	SPI	Set Pending Interrupt Register: This command ORs the 16-bit contents of RA with the pending interrupt register. If there is a one in the corresponding bit position of the interrupt

Code	Mnemonic	Description
		mask (same bit set in both the PI and the MK), and the interrupts are enabled, then an interrupt shall occur after execution of the next instruction. If PI_5 is set to 1, then N is assumed to be 0 (see Section 5.30, “Branch to Executive” [87]).
200E	WSW	Write Status Word: This command transfers the contents of RA to the status word.
8YXX	PI	Programmed Input: This command inputs 16 bits of data into RA from the programmed I/O port. Y may be from 0 through 3.
A000	RMK	Read Interrupt Mask: The current interrupt mask is transfered into register RA. The interrupt mask is not altered.
A004	RPIR	Read Pending Interrupt Register: This command transfers the contents of the pending interrupt register into RA. The pending interrupt register is not altered.
A00E	RSW	Read Status Word: This command transfers the 16-bit status word into register RA. The status word remains unchanged.
A00F	RCFR	Read and Clear Fault Register: This command inputs the 16-bit fault register to register RA. The contents of the fault register are reset to zero. Bit 1 in the pending interrupt register is reset to zero.

Table XIII. Optional XIO Command Fields and Mnemonics

Code	Mnemonic	Description
0YXX	PO	Programmed Output: This command outputs 16 bits of data from RA to a programmed I/O port. Y may be from 0 through 3.
2008	OD	Output Discretes: This command outputs the 16-bit contents of the register RA to the discrete output buffer. A "1" indicates an "on"

Code	Mnemonic	Description
		condition and a "0" indicates an "off" condition.
200A	RNS	Reset Normal Power Up Discrete: This command resets the normal power up discrete bit.
4000	CO	Console Output: The 16-bit contents (2 bytes) of register RA are output to the console. The eight most significant bits (byte) are sent first. If no console is present, then this command is treated as a NOP (see Section 5.96, “No Operation” [151]).
4001	CLC	Clear Console: This command clears the console interface.
4003	MPEN	Memory Protect Enable: This command allows the memory protect RAM to control memory protection.
4004	ESUR	Enable Start Up ROM: This command enables the start up ROM (i.e., the ROM overlays main memory).
4005	DSUR	Disable Start Up ROM: This command disables the start up ROM.
4006	DMAE	Direct Memory Access Enable: This command enables direct memory access (DMA).
4007	DMAD	Direct Memory Access Disable: This command disables DMA.
4008	TAS	Timer A, Start: This command starts timer A from its current state. The timer is incremented every 10 microseconds.
4009	TAH	Timer A, Halt: This command halts timer A at its current state.
400A	OTA	Output Timer A: The contents of register RA are loaded (i.e., jam transfered) into timer A and the timer automatically starts operation by incrementing from the loaded timer in steps of ten microseconds. Bit fifteen is the least

Code	Mnemonic	Description
		significant bit and shall represent ten microseconds.
400B	GO	Trigger Go Indicator: This command restarts a counter which is connected to a discrete output. The period of time from restart to time-out shall be determined by the system requirements. When the Go timer is started, the discrete output shall go high and remain high for TBD milliseconds, at which time the output shall go low unless another GO is executed. The Go discrete output signal may be used as a software fault indicator.
400C	TBS	Timer B, Start: This command starts timer B from its current state. The timer is incremented every 100 microseconds.
400D	TBH	Timer B, Halt: This command halts timer B at its current state.
400E	OTB	Output Timer B: The contents of register RA are loaded (i.e., jam transfered) into timer B and the timer automatically starts operation by incrementing from the loaded timer in steps of one hundred microseconds. Bit fifteen is the least significant bit and shall represent one hundred microseconds.
50XX	LMP	Load Memory Protect RAM (5000 + RAM address): This command outputs the 16-bit contents of register RA to the memory protect RAM. A "1" in a bit provides write protection and a "0" in a bit permits writing to the corresponding 1024 word physical memory block. The RAM word MSB (bit 0) represents the lowest number block and the RAM word LSB (bit 15) represents the highest block (i.e., bit 0 represents locations 0 through 1023 and bit 15 represents locations 15360 through 16383 for word zero). Each word represents consecutive 16K blocks of physical memory.

Code	Mnemonic	Description
		The RAM words of 0 through 63 apply to processor write protect and words 64 through 127 apply to DMA write protect.
51XY	WIPR	Write Instruction Page Register: This command transfers the contents of register RA to page register Y of the instruction set group X.
52XY	WOPR	Write Operand Page Register: This command transfers the contents of register RA to page register Y of the operand set of group X.
8YXX	PI	Programmed Input: This command inputs 16 bits of data into RA from the programmed I/O port. Y may be from 0 through 3.
A001	RIC1	Read Input/Output Interrupt Code, Level 1: This command inputs the contents of the level 1 IOIC register into register RA. The channel number is right justified.
A002	RIC2	Read Input/Output Interrupt Code, Level 2: This command inputs the contents of the level 2 IOIC register into register RA. The channel number is right justified.
A008	RDOR	Read Discrete Output Register: This command inputs the 16-bit discrete output buffer into register RA.
A009	RDI	Read Discrete Input: This command inputs the 16-bit discrete input word into register RA. A "1" indicates an "on" condition and a "0" indicates an "off" condition.
A00B	TPIO	Test Programmed Output: This command inputs the 16-bit contents of the programmed output buffer into register RA. This command may be used to test the PIO channel by means of a wrap around test.
A00D	RMFS	Read Memory Fault Status: This command transfers the 16-bit contents of the memory fault status register to RA. The fields within

Code	Mnemonic	Description
		the memory fault status register shall delineate memory related fault types and shall provide the page register designators associated with the designated fault.
C000	CI	Console Input: This command inputs the 16-bits (2 bytes) from the console into register RA. The eight most significant bits of RA shall represent the first byte.
C001	RCS	Read Console Status: This command inputs the console interface status into register RA. The status is right justified.
C00A	ITA	Input Timer A: This command inputs the 16-bit contents of timer A into register RA. Bit fifteen is the least significant bit and represents a time increment of ten microseconds.
C00E	ITB	Input Timer B: This command inputs the 16-bit contents of timer B into register RA. Bit fifteen is the least significant bit and represents a time increment of one hundred microseconds.
D0XX	RMP	Read Memory Protect RAM (D000 + RAM address): This command inputs the appropriate memory protect word into register RA. A "1" in a bit provides write protection and a "0" in a bit permits writing to the corresponding 1024 word physical memory block. The RAM words MSB (bit 0) represents the lowest number block and the RAM word LSB (bit 15) represents the highest block (i.e., bit 0 represents locations 0 through 1023 and bit 15 represents locations 15360 through 16383 for word zero). Each word represents consecutive 16K blocks of physical memory. The RAM words of 0 through 63 apply to processor write protect and words 64 through 127 apply to DMA write protect.

Code	Mnemonic	Description
D1XY	RIPR	Read Instruction Page Register: This command transfers the 16-bit contents of the page register Y of the instruction set of group X to register RA.
D2XY	ROPR	Read Operand Page Register: This command transfers the 16-bit contents of page register Y of the operand set of group X to register RA.

Note **** **** User defined XIO functions (see Table IX, “Input/Output Channel Groups” [44]).

Register Transfer Description. Varies depending on the command field.

Registers Affected. Varies depending on the command field.

5.2. Vectored Input/Output

```
Addr
Mode  Mnemonic          Format/Opcode
                            8      4     4             16
D     VIO  RA,ADDR      ------------------------------------
DX    VIO  RA,ADDR,RX   |  49  |  RA  |  RX  |  |    ADDR    |
                        ------------------------------------
```

Description. The vectored input/output instruction performs the I/O operation as specified by the input/output vector table starting at the derived address, DA, as shown below:

```
              --------------------
DA            |       CMD         |
              --------------------
DA + 1        |   Vector Select   |
              --------------------
DA + 2        |      Data         |  }  one data word for each bit
```

```
                ---------------------     set in the vector select
...             |       ...         |
```

The input/output operation or device address is specified by the sum of the CMD and the product of the bit number of the bit set in the vector select times the contents of RA. This device address is then interpreted as specified by the XIO instruction (see Section 5.1, “Execute Input/Output” [53]) with the exception that I/O data is transfered to or from DA + 2 + i rather than RA (where i starts at zero and is incremented after each transfer). This is a privileged instruction. If an illegal XIO command is encountered as part of a VIO chain, the following actions occur:

- The illegal I/O command bit of the fault register (FT_5) is set to a one.
- The VIO chain is terminated, and the illegal XIO is treated as a NOP. This termination shall not affect execution of preceding XIO commands which are part of the VIO chain being executed.

Register Transfer Description.

Step 1. n <-- 0 and i <-- 0;
Step 2. if $[DA + 1]_n = 1$, then I/O command = [DA] + n x (RA);
Step 3. FT_5 <-- 1, exit, if XIO = illegal command;
Step 4. if $[DA + 1]_n = 1$, then I/O data = [DA + 2 + i];
Step 5. if $[DA + 1]_n = 1$, then i <-- i + 1;
Step 6. n <-- n + 1, exit, if n = 16;
Step 7. go to step 2;

Registers Affected. None

5.3. Set Bit

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4
```

```
                          --------------------
R     SBR   N,RB          |  51  |  N  |  RB  |
                          --------------------
                            8      4     4           16
D     SB    N,ADDR        -------------------------------------
DX    SB    N,ADDR,RX     |  50  |  N  |  RX  |  |    ADDR     |
                          -------------------------------------
                            8      4     4           16
I     SBI   N,ADDR        -------------------------------------
IX    SBI   N,ADDR,RX     |  52  |  N  |  RX  |  |    ADDR     |
                          -------------------------------------
```

Description. Bit number N of the Derived Operand, DO, is set to one. The MSB is designated bit number zero and the LSB is designated bit number fifteen.

Register Transfer Description.

DO_N <-- 1;

Registers Affected. RB

5.4. Reset Bit

```
Addr
Mode  Mnemonic          Format/Opcode
                          8      4     4
                          --------------------
R     RBR   N,RB          |  54  |  N  |  RB  |
                          --------------------
                            8      4     4           16
D     RB    N,ADDR        -------------------------------------
DX    RB    N,ADDR,RX     |  53  |  N  |  RX  |  |    ADDR     |
                          -------------------------------------
                            8      4     4           16
I     RBI   N,ADDR        -------------------------------------
IX    RBI   N,ADDR,RX     |  55  |  N  |  RX  |  |    ADDR     |
                          -------------------------------------
```

Description. Bit number N of the Derived Operand, DO, is set to zero. The MSB is designated bit number zero and the LSB is designated bit number fifteen.

Register Transfer Description.

```
DO_N <-- 0;
```

Registers Affected. RB

5.5. Test Bit

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4
                        ---------------------
R     TBR   N,RB        |  57  |  N  |  RB  |
                        ---------------------
                           8      4     4             16
D     TB    N,ADDR      ---------------------------------------
DX    TB    N,ADDR,RX   |  56  |  N  |  RX  |  |    ADDR     |
                        ---------------------------------------
                           8      4     4             16
I     TBI   N,ADDR      ---------------------------------------
IX    TBI   N,ADDR,RX   |  58  |  N  |  RX  |  |    ADDR     |
                        ---------------------------------------
```

Description. Bit number N (0 <= N <= 15) of the Derived Operand, DO, is tested. Then the Condition Status, CS, is set to indicate non-zero if bit number N of the DO contains a one. Otherwise CS is set to indicate zero. The MSB of the DO is designated bit number zero and the LSB of the DO is designated bit number fifteen.

Register Transfer Description.

(CS) <-- 0010 if DO_N = 0 and 0 <= N <= 15;
(CS) <-- 0001 if DO_N = 1 and N = 0;
(CS) <-- 0100 if DO_N = 1 and 1 <= N <= 15;

Registers Affected. CS

5.6. Test and Set Bit

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4            16
D     TSB   N,ADDR      ---------------------------------------
DX    TSB   N,ADDR,RX   |  59  |  N  |  RX  |  |     ADDR    |
                        ---------------------------------------
```

Description. Bit number N (0 <= N <= 15) of the Derived Operand, DO, is tested and set to one. The CS is set according to the test.

Note External memory accesses shall be inhibited until this instruction is complete.

Register Transfer Description.

(CS) <-- 0010 and (DO_N) <-- 1 if DO_N = 0 and 0 <= N <= 15;
(CS) <-- 0001 if (DO_N) = 1 and N = 0;
(CS) <-- 0100 if (DO_N) = 1 and 1 <= N <= 15;

Registers Affected. CS

5.7. Set Variable Bit in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
```

```
                                 ----------------------
R      SVBR   RA,RB              |  5A  |  RA  |  RB  |
                                 ----------------------
```

Description. Bit number N (0 <= N <= 15) of the register RB is set to one where the least significant four bits of the contents of register RA is N. Bits $(RA)_{0\text{-}11}$ have no effect on the operation. If RA = RB, then the count is determined first and then the appropriate bit is changed.

Register Transfer Description.

$(RB)_N$ <-- 1 where N = $(RA)_{12\text{-}15}$;

Registers Affected. RB

5.8. Reset Variable Bit in Register

```
Addr
Mode   Mnemonic           Format/Opcode
                             8      4      4
                           ----------------------
R      RVBR    RA,RB       |  5C  |  RA  |  RB  |
                           ----------------------
```

Description. Bit number N (0 <= N <= 15) of register RB is set to zero where the least significant four bits of the contents of register RA is N. Bits $(RA)_{0\text{-}11}$ have no effect on the operation. If RA = RB, then the count is determined first and then the appropriate bit is changed.

Register Transfer Description.

$(RB)_N$ <-- $0_{12\text{-}15}$ where N = (RA) ;

Registers Affected. RB

5.9. Test Variable Bit in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     TVBR  RA,RB       |  5E  |  RA  |  RB  |
                        ----------------------
```

Description. Bit number N (0 <= N <= 15) of register RB is tested where the least significant four bits of the contents of register RA is N. The Condition Status, CS, is then set to indicate non-zero if bit number N of register RB is a one. Otherwise, CS is set to indicate zero.

Register Transfer Description.

$N = (RA)_{12\text{-}15}$;
(CS) <-- 0010 if $(RB_N) = 0$ and 0 <= N <= 15;
(CS) <-- 0001 if $(RB_N) = 1$ and N = 0;
(CS) <-- 0100 if $(RB_N) = 1$ and 1 <= N <= 15;

Registers Affected. CS

5.10. Shift Left Logical

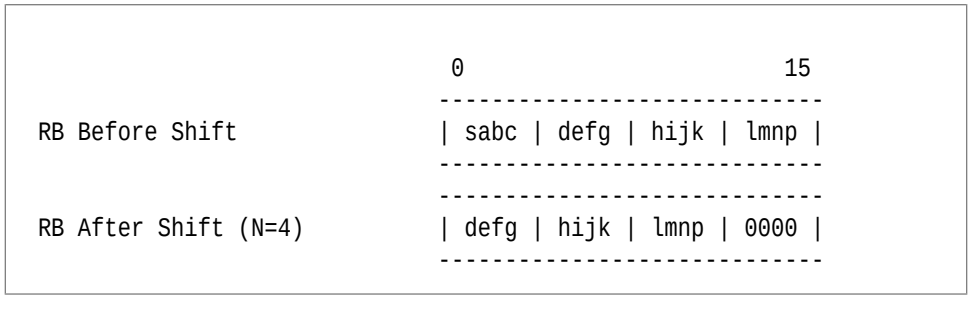
```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                        -----------------------
R     SLL  RB,N         |  60  |  N-1  |  RB  |   1 <= N <= 16
                        -----------------------
```

Description. The contents of the Derived Address, DA (i.e., the contents of register RB) are shifted left logically N positions. The shifted result is stored in RB. The logical shift left operation is as

follows: zeros enter the least significant bit position (bit 15) and bits shifted out of the sign bit position (bit 0) are lost. The condition status, CS, is set based on the result in register RB.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.

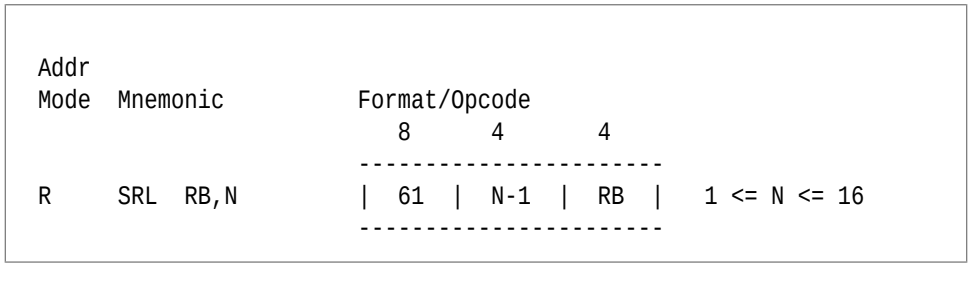


Register Transfer Description.

```
(RB) <-- (RB) Shifted left logically by N positions;
(CS) <-- 0010  if (RB) = 0;
(CS) <-- 0001  if (RB) < 0;
(CS) <-- 0100  if (RB) > 0;
```

Registers Affected. RB,CS

5.11. Shift Right Logical



Description. The contents of the Derived Address, DA (i.e., the contents of register RB), are shifted right logically N positions. The shifted result is stored in RB. The logical shift right operation

is as follows: zeros enter the sign bit position (bit 0) and bits shifted out of the least significant bit position (bit 15) are lost. The condition status, CS, is set based on the result in register RB.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.

```
                        0                       15
                       ----------------------------
RB Before Shift        | sabc | defg | hijk | lmnp |
                       ----------------------------
                       ----------------------------
RB After Shift (N=4)   | 0000 | sabc | defg | hijk |
                       ----------------------------
```

Register Transfer Description.

```
(RB) <-- (RB)  Shifted right logically by N positions;
(CS) <-- 0010  if (RB) = 0;
(CS) <-- 0001  if (RB) < 0;
(CS) <-- 0100  if (RB) > 0;
```

Registers Affected. RB,CS

5.12. Shift Right Arithmetic

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                       ----------------------
R     SRA  RB,N        |  62  |  N-1  |  RB  |  1 <= N <= 16
                       ----------------------
```

Description. The contents of the Derived Address, DA (i.e., the contents of register RB), are shifted right arithmetically N positions. The shifted result is stored in RB. The arithmetic right shift

operation is as follows: the sign bit, which is not changed, is copied into the next position for each position shifted and bits shifted out of the least significant bit position (bit 15) are lost. The condition status, CS, is set based on the result in register RB.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.

```
                        0                       15
                       ----------------------------
RB Before Shift        | sabc | defg | hijk | lmnp |
                       ----------------------------
                       ----------------------------
RB After Shift (N=4)   | ssss | sabc | defg | hijk |
                       ----------------------------
```

Register Transfer Description.

```
(RB) <-- (RB)  Shifted right arithmetically by N positions;
(CS) <-- 0010  if (RB) = 0;
(CS) <-- 0001  if (RB) < 0;
(CS) <-- 0100  if (RB) > 0;
```

Registers Affected. RB,CS

5.13. Shift Left Cyclic

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                       ----------------------
R     SLC   RB,N       |  63  |  N-1  |  RB  |   1 <= N <= 16
                       ----------------------
```

Description. The contents of the Derived Address, DA (i.e., the contents of register RB), are shifted left cyclically N positions. The

shifted result is stored in RB. The cyclic left shift operation is as follows: bits shifted out of the sign bit position (bit 0) enter the least significant bit position (bit 15) and, consequently, no bits are lost. The condition status, CS, is set based on the result in RB.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.

```
                         0                        15
                        ----------------------------
RB Before Shift         | sabc | defg | hijk | lmnp |
                        ----------------------------
                        ----------------------------
RB After Shift (N=4)    | defg | hijk | lmnp | sabc |
                        ----------------------------
```

Register Transfer Description.

```
(RB) <-- (RB)  Shifted left cyclically by N positions;
(CS) <-- 0010  if (RB) = 0;
(CS) <-- 0001  if (RB) < 0;
(CS) <-- 0100  if (RB) > 0;
```

Registers Affected. RB,CS

5.14. Double Shift Left Logical

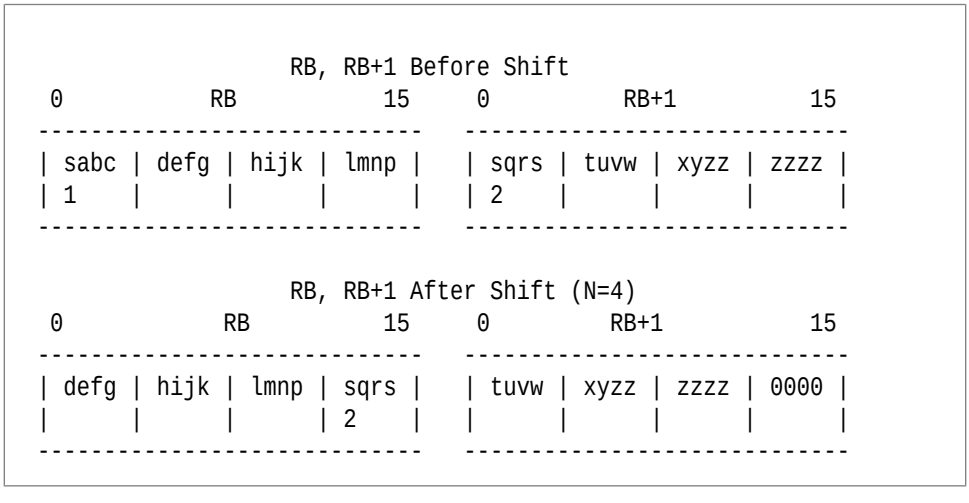
```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                        ----------------------
R     DSLL   RB,N       |  65  |  N-1  |  RB  |   1 <= N <= 16
                        ----------------------
```

Description. The concatenated contents of the Derived Address, DA, and DA+1 (i.e., the concatenated contents of RB and RB+1),

are shifted left logically N positions. The shifted results are stored in RB and RB+1. The double left shift logical operation is as follows: zeros enter the least significant bit position of RB+1, bits shifted out of the sign bit position of RB+1 enter the least significant bit of RB and bits shifted out of the sign bit of RB are lost. The condition status, CS, is set based on the result in registers RB and RB+1.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.



Register Transfer Description.

(RB,RB+1) <-- (RB,RB+1) Shifted left logically by N positions;
(CS) <-- 0010 if (RB,RB+1) = 0;
(CS) <-- 0001 if (RB,RB+1) < 0;
(CS) <-- 0100 if (RB,RB+1) > 0;

Registers Affected. RB, RB+1, CS

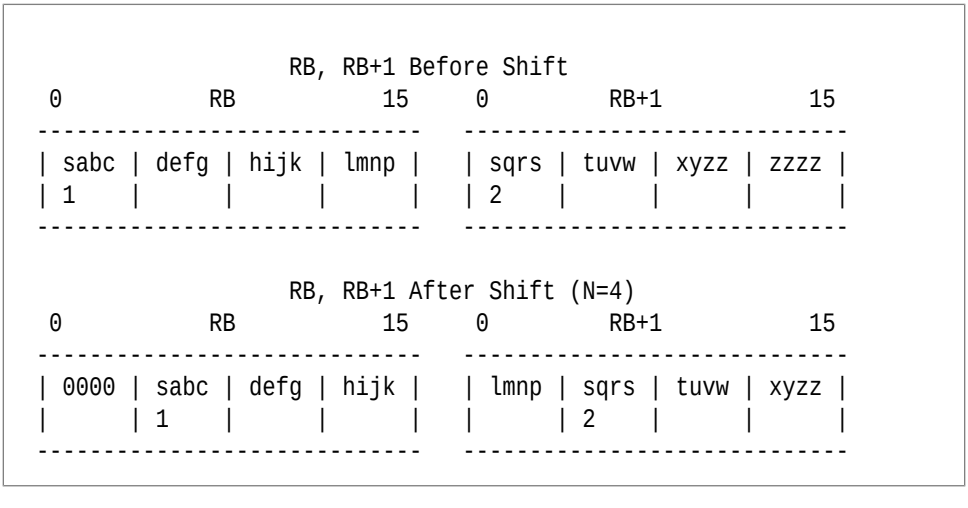
5.15. Double Shift Right Logical

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                        -----------------------
R     DSRL   RB,N       |  66  |  N-1  |  RB  |     1 <= N <= 16
                        -----------------------
```

Description. The concatenated contents of the Derived Address, DA, and DA+1 (i.e., the concatenated contents of RB and RB+1), are shifted right logically N positions. The shifted results are stored in RB and RB+1. The double logical right shift operation is as follows: zeros enter the sign bit position of RB, bits shifted out of the least significant bit position of RB enter the sign bit position of RB+1 and bits shifted out of the least significant bit position of RB+1 are lost. The condition status, CS, is set based on the result in register RB and RB+1.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.



Register Transfer Description.

```
(RB,RB+1) <-- (RB,RB+1)  Shifted right logically by N positions;
(CS) <-- 0010  if (RB,RB+1) = 0;
(CS) <-- 0001  if (RB,RB+1) < 0;
(CS) <-- 0100  if (RB,RB+1) > 0;
```

Registers Affected. RB, RB+1, CS

5.16. Double Shift Right Arithmetic

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                        ----------------------
R     DSRA   RB,N       |  67  |  N-1  |  RB  |     1 <= N <= 16
                        ----------------------
```

Description. The concatenated contents of the Derived Address, DA, and DA+1 (i.e., the concatenated contents of RB and RB+1), are shifted right arithmetically N positions. The shifted results are stored in RB and RB+1. The double right shift arithmetic operation is as follows: the sign bit of RB, which is not changed, is copied into the next position for each position shifted, bits shifted out of the least significant position of RB enter the sign bit position of RB+1, and bits shifted out of the least significant bit position of RB+1 are lost. The condition status, CS, is set based on the result in register RB and RB+1.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.

```
                  RB, RB+1 Before Shift
 0            RB          15     0          RB+1          15
-----------------------------   -----------------------------
| sabc | defg | hijk | lmnp |   | sqrs | tuvw | xyzz | zzzz |
| 1    |      |      |      |   | 2    |      |      |      |
-----------------------------   -----------------------------
```

```
                  RB, RB+1 After Shift (N=4)
 0            RB          15     0          RB+1          15
-----------------------------   -----------------------------
| ssss | sabc | defg | hijk |   | lmnp | sqrs | tuvw | xyzz |
| 1111 | 1    |      |      |   |      | 2    |      |      |
-----------------------------   -----------------------------
```

Register Transfer Description.

```
(RB,RB+1) <-- (RB,RB+1)  Shifted right arithmetically by N positions;
(CS) <-- 0010  if (RB,RB+1) = 0;
(CS) <-- 0001  if (RB,RB+1) < 0;
(CS) <-- 0100  if (RB,RB+1) > 0;
```

Registers Affected. RB, RB+1, CS

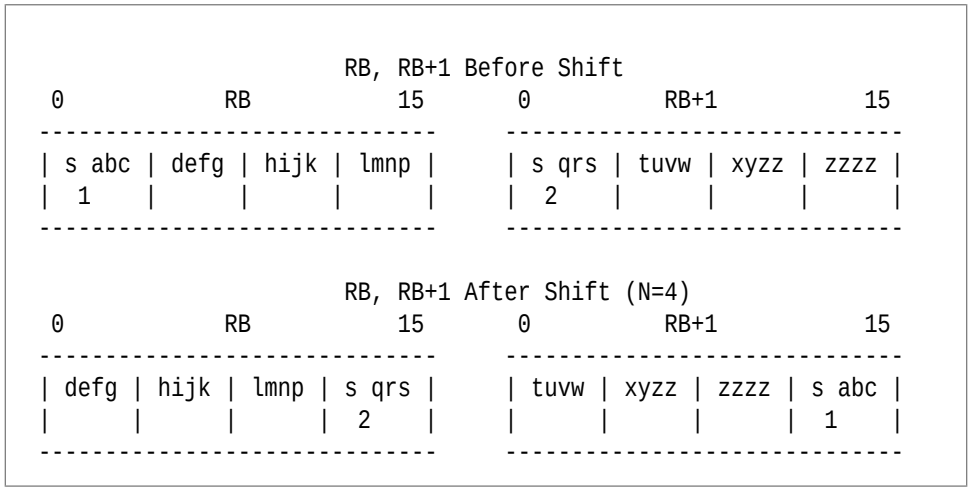
5.17. Double Shift Left Cyclic

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4
                        -----------------------
R     DSLC   RB,N       |  68  |  N-1  |  RB  |     1 <= N <= 16
                        -----------------------
```

Description. The concatenated contents of the Derived Address, DA, and DA+1 (i.e., the concatenated contents of RB and RB+1), are shifted left cyclically N positions. The shifted results are stored in RB and RB+1. The double left shift cyclic operation is as follows: bits shifted out of the sign bit position of RB enter the least significant bit position of RB+1, bits shifted out of the sign bit position of RB+1 enter the least significant bit position of RB, and, consequently, no bits are lost. The condition status, CS, is set based on the result in RB and RB+1.

Note N-1 = 0 represents a shift of one position.

Note N-1 = 15 represents a shift of sixteen positions.



Register Transfer Description.

(RB,RB+1) <-- (RB,RB+1) Shifted left cyclically by N positions;
(CS) <-- 0010 if (RB,RB+1) = 0;
(CS) <-- 0001 if (RB,RB+1) < 0;
(CS) <-- 0100 if (RB,RB+1) > 0;

Registers Affected. RB, RB+1, CS

5.18. Shift Logical, Count in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                            8      4      4
                        ----------------------
R     SLR   RA,RB       |  6A  |  RA  |  RB  |     |(RB)| <= 16
                        ----------------------
```

Description. The contents of register RA are shifted logically N positions, where N is the contents of register RB. If N is positive

($((RB_0)=0$), then the shift direction is left; if N is negative (2's complement notation, $(RB_0)=1$), then the shift direction is right. The condition status, CS, is set based on the result in RA.

Note N = 0 represents a shift of zero positions.

Note If |N| >= 16, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]).

Note The contents of RB remain unchanged, unless RA = RB; in this event the contents are shifted N positions.

Note (See "Description" of the logical shift instructions, SLL and SRL (see Section 5.10, “Shift Left Logical” [66]) and Section 5.11, “Shift Right Logical” [67]), for the definition of shift operations.)

Register Transfer Description.

```
PI4 <-- 1, exit,
   if |N| >= 16
(RA) <-- (RA) Shifted left logically by (RB) positions,
 if 0 < (RB) <= 16;
(RA) <-- (RA) Shifted right logically by -(RB) positions,
 if 0 >= (RB) >== -16;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, RB, CS, PI

5.19. Shift Arithmetic, Count in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
```

```
R     SAR   RA,RB       |  6B  |  RA  |  RB  |     |(RB)| <= 16
                       ----------------------
```

Description. The contents of register RA are shifted arithmetically N positions, where N is the contents of register RB. If N is positive ($(RB_0) = 0$), then the shift direction is left; if N is negative (2's complement notation, $(RB_0) = 1$), then the shift direction is right. The condition status, CS, is set based on the result in RA.

Note N = 0 represents a shift of zero positions.

Note If |N| >= 16, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]).

Note The contents of RB remain unchanged, unless RA = RB; in this event, the contents are shifted N positions.

Note (See “Description” of the arithmetic shift instruction SRA (see Section 5.12, “Shift Right Arithmetic” [68]) for definition of the right shift operation. Left shift causes “zeros” to be shifted into low order position of result.)

Note Fixed point overflow occurs if the sign bit changes during a left shift.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if |N| >= 16;
(RA) <-- (RA)  Shifted left arithmetically (RB) positions,
 if 16 >== (RB) > 0;
(RA) <-- (RA) Shifted right arithmetically -(RB) positions,
 if 0 >= (RB) >== -16;
PI4 <-- 1,
   if (RA0) changes during the shift;
(CS) <-- 0010  if (RA) = 0;
```

```
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, RB, CS, PI

5.20. Shift Cyclic, Count in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     SCR   RA,RB       |  6C  |  RA  |  RB  |    |(RB)| <= 16
                        ----------------------
```

Description. The contents of register RA are shifted cyclically N positions, where N is the contents of register RB. If N is positive ($(RB_0) = 0$), then the shift direction is left; if N is negative (2's complement notation, $(RB_0) = 1$), then the shift direction is right. The condition status, CS, is set based on the result in RA.

Note N = 0 represents a shift of zero positions.

Note If |N| >= 16, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]).

Note (See "Description" of the cyclic shift instruction, SLC (see Section 5.13, “Shift Left Cyclic” [69]), for definition of shift operations.)

Note The contents of RB remain unchanged, unless RA = RB in this event, the contents are shifted N positions.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if |N| >= 16;
(RA) <-- (RA) Shifted left cyclically by (RB) positions,
```

```
 if 0 < (RB) <= 16;
(RA) <-- (RA) Shifted right cyclically by -(RB) positions,
 if 0 >= (RB) >== -16;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, RB, CS, PI

5.21. Double Shift Logical, Count in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DSLR   RA,RB      |  6D  |  RA  |  RB  |     |(RB)| <= 32
                        ----------------------
```

Description. The concatenated contents of register RA and RA+1 are shifted logically N positions where register RB contains the count, N. If the count is positive ((RB_0) = 0), then the shift direction is left. If the count is negative (2's complement notation, (RB_0) = 1), then the shift direction is right. The condition status, CS, is set based on the result in RA and RA+1.

Note N = 0 represents a shift of zero positions.

Note If |N| >= 32, the fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]).

Note (See "Description" of the double shift logical instructions, DSRL and DSLL (see Section 5.15, “Double Shift Right Logical” [72] and Section 5.14, “Double Shift Left Logical” [70]), for definition of shift operations.)

Note The contents of RB remain unchanged, unless RA = RB; in this event, the contents are shifted N positions.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if |N| >= 32;
(RA,RA+1) <-- (RA,RA+1) Shifted left logically by (RB) positions

if 32 >== (RB) > 0;
(RA,RA+1) <-- (RA,RA+1) Shifted right logically by -(RB) positions

if 0 >= (RB) >== -32;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, RB, CS, PI

5.22. Double Shift Arithmetic, Count in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
```

```
R     DSAR   RA,RB      |  6E  |  RA  |  RB  |     |(RB)| <= 32
                       ----------------------
```

Description. The concatenated contents of register RA and RA+1 are shifted arithmetically N positions where register RB contains the count, N. If the count is positive ((RB_0)=0), then the shift direction is left. If the count is negative (2's complement notation, (RB_0)=1), then the shift direction is right. The condition status, CS, is set based on the result in RA and RA+1.

Note N = 0 represents a shift of zero positions.

Note If |N| >= 32, the fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]).

Note The contents of RB remain unchanged, unless RA = RB; in this event, the contents are shifted N positions.

Note (See "Description" of the double shift arithmetic instruction, DSRA (see Section 5.16, “Double Shift Right Arithmetic” [73]), for the definition of the right shift operation. Left shift causes "zeros" to be shifted into low order position of result.)

Note Fixed point overflow occurs if the sign bit is changed during a left shift.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if |N| >= 32;
(RA,RA+1) <-- (RA,RA+1) Shifted left arithmetically (RB) positions,
if 32 >= (RB) >== 0;
(RA,RA+1) <-- (RA,RA+1) Shifted right arithmetically -(RB) positions,
if 0 >= (RB) >== -32;
PI4 <-- 1,
   if (RA0) changes during the shift;
(CS) <-- 0010  if (RA,RA+1) = 0;
```

```
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, RB, CS, PI

5.23. Double Shift Cyclic, Count in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DSCR   RA,RB      |  6F  |  RA  |  RB  |    |(RB)| <= 32
                        ----------------------
```

Description. The concatenated contents of registers RA and RA+1 are shifted cyclically N positions, where register RB contains the count, N. If the count is positive ($(RB_0)=0$), the shift direction is left.

If the count is negative (2's complement notation, $(RB_0)=1$), the shift direction is right. The condition status, CS, is set based on the result in RA and RA+1.

Note N = 0 represents a shift of zero positions.

Note If |N| >= 32, the fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]).

Note (See "Description" of the double shift cyclic instruction, DSLC (see Section 5.17, “Double Shift Left Cyclic” [74]), for the definition of shift operations.)

Note The contents of RB remain unchanged, unless RA = RB; in this event, the contents are shifted N positions.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if |N| >= 32;
(RA,RA+1) <-- (RA,RA+1) Shifted left cyclically by (RB) positions
      if 32 >= (RB) >== 0;
(RA,RA+1) <-- (RA,RA+1) Shifted right cyclically by -(RB) positions
      if 0 >= (RB) >== -32;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, RB, CS, PI

5.24. Jump on Condition

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4             16
D     JC   C,LABEL      ---------------------------------------
DX    JC   C,LABEL,RX   |  70  |  C  |  RX  |  |    LABEL    |
                        ---------------------------------------
                           8      4     4             16
I     JCI  C,ADDR       ---------------------------------------
IX    JCI  C,ADDR,RX    |  71  |  C  |  RX  |  |     ADDR    |
                        ---------------------------------------
```

Description. This is a conditional jump instruction wherein the instruction sequence jumps to the Derived Address, DA, if a logical one results from the following operation:

1. The 4-bit C field is bit-by-bit ANDed with the 4-bit condition status, CS

2. The resulting 4-bits are ORed together

3. or if C = 7 or C = F.

Otherwise, the next sequential instruction is executed.

Condition Codes.

C_2	C_{16}	Jump Condition	Mnemonic		
0000	0	NOP	--	--	--
0001	1	less than (zero)	LT	LZ	M
0010	2	equal to (zero)	EQ	EZ	--
0011	3	less than or equal to (zero)	LE	LEZ	NP
0100	4	greater than (zero)	GT	GZ	P
0101	5	not equal to (zero)	NE	NZ	--
0110	6	greater than or equal to (zero)	GE	GEZ	NM
0111	7	unconditional	--	--	--
1000	8	carry	CY	--	--
1001	9	carry or LT	--	--	--
1010	A	carry or EQ	--	--	--
1011	B	carry or LE	--	--	--
1100	C	carry or GT	--	--	--
1101	D	carry or NE	--	--	--
1110	E	carry or GE	--	--	--
1111	F	unconditional	--	--	--

Register Transfer Description.

(IC) <-- DA if C = 7, or
if C = F, or
if $(C_0 \wedge CS_0) \vee (C_1 \wedge CS_1) \vee (C_2 \wedge CS_2) \vee (C3 \wedge CS_3) = 1$;

Registers Affected. IC (if jump is executed)

5.25. Jump to Subroutine

Addr Mode	Mnemonic		Format/Opcode				
			8	4	4	16	
D	JS	RA,LABEL	72	RA	RX	LABEL	
DX	JS	RA,LABEL,RX	72	RA	RX	LABEL	

Description. The value of the instruction counter (the address of the next sequential instruction) is stored into register RA. Then, the IC is set to the derived address, DA, thus effecting the jump.

This sets up the return from subroutine to the address stored in the register RA, i.e., an indexed unconditional jump from location zero using RA as the index register shall transfer control to the instruction following the JS instruction.

Note If RA = RX, then the derived address, DA, is calculated before the IC is stored in RA.

Register Transfer Description.

```
(RA) <-- (IC);
(IC) <-- DA;
```

Registers Affected. RA, IC

5.26. Subtract One and Jump

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4            16
D     SOJ   RA,LABEL    -------------------------------------
DX    SOJ   RA,LABEL,RX |  73  |  RA  |  RX  |  |   LABEL    |
                        -------------------------------------
```

Description. The 16 bit contents of register RA are decremented by one. Then if the content of register RA is zero, the next sequential instruction is executed. If the content of register RA is non-zero, then a jump to the Derived Address, DA, occurs.

Note If RA = RX, then the derived address, DA, is calculated before RA is decremented.

Register Transfer Description.

```
(RA) <-- (RA) - 1;
(IC) <-- DA if (RA) /= 0;
(CS) <-- 0010  if (RA) = 0;
```

```
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS, IC (if the jump is executed)

5.27. Branch Unconditionally

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      8
                        --------------
ICR   BR   LABEL        |  74  |  D  |     -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA.

Register Transfer Description.

```
(IC) <-- DA;
```

Registers Affected. IC

5.28. Branch if Equal to (Zero)

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      8
                        --------------
ICR   BEZ   LABEL       |  75  |  D  |   -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that

the previous result which set the CS is equal to (zero). Otherwise, the next sequential instruction is executed.

Register Transfer Description.

```
(IC) <-- DA if (CS) = X010;
```

Registers Affected. IC (if the jump is executed)

5.29. Branch if Less Than (Zero)

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      8
                        --------------
ICR   BLT   LABEL       |  76  |  D  |   -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is less than (zero). Otherwise, the next sequential instruction is executed.

Register Transfer Description.

```
(IC) <-- DA if (CS) = X001;
```

Registers Affected. IC (if the jump is executed)

5.30. Branch to Executive

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4       4
                        ----------------------
```

```
S     BEX   N          |  77  |  0000  |  N  |
                       ----------------------
```

Description. This instruction provides a means to jump to a routine in another address state, AS. It is typically used to make controlled, protected calls to an executive. The 4-bit literal N selects one of 16 executive entry points to be used. Execution of this instruction causes an interrupt to occur using the EXEC call interrupt vector (interrupt 5). The new IC is loaded from the Nth location following the SW in the new processor state. The linkage pointer (LP), service pointer (SVP), and the new processor state (new MK, new SW, and new IC) are fetched from address state zero. The current processor state (old MK, old SW, and old IC) are stored in the address state specified by the new SW AS field. Interrupts are disabled when BEX is executed. The EXEC call interrupt cannot be masked or disabled. Arguments associated with the BEX instruction are passed by software convention. The processor lock and key function is ignored when this instruction is executed. An attempt to branch into an execute protected area of memory shall result in FT being set to 1.

Register Transfer Description.

(RQ,RQ+1,RQ+2) <-- (MK,SW,IC);
(SVP) <-- [$2B_{16}$], where AS = 0;
(MK,SW,IC) <-- [(SVP),(SVP)+1,(SVP)+2+N)], where AS = 0;
(LP) <-- [$2A_{16}$], where AS = 0;
[(LP),(LP)+1,(LP)+2] <-- (RQ,RQ+1,RQ+2), where AS = SW_{12-15};

Registers Affected. MK, SW, IC, PI

5.31. Branch if Less Than or Equal to (Zero)

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      8
                        --------------
```

```
ICR   BLE   LABEL       |  78  |  D  |    -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is less than or equal to (zero). Otherwise, the next sequential instruction in executed.

Register Transfer Description.

```
(IC) <-- DA if (CS) = X010 or (CS) = X001;
```

Registers Affected. IC (if the jump is executed)

5.32. Branch if Greater Than (Zero)

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      8
                        --------------
ICR   BGT   LABEL       |  79  |  D  |    -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is greater than (zero). Otherwise, the next sequential instruction is executed.

Register Transfer Description.

```
(IC) <-- DA if (CS) = X100;
```

Registers Affected. IC (if the jump is executed)

5.33. Branch if Not Equal to (Zero)

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      8
                        --------------
ICR   BNZ   LABEL       |  7A  |  D  |    -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is not equal to (zero). Otherwise, the next sequential instruction is executed.

Register Transfer Description.

```
(IC) <-- DA if (CS) = X100 or (CS) = X001;
```

Registers Affected. IC (if the jump is executed)

5.34. Branch if Greater Than or Equal to (Zero)

```
Addr
Mode  Mnemonic          Format/Opcode
                           8     8
                        --------------
ICR   BGE   LABEL       |  7B  |  D  |    -128 <= D <= 127
                        --------------
```

Description. A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is greater than or equal to (zero). Otherwise, the next sequential instruction is executed.

Register Transfer Description.

```
(IC) <-- DA if (CS) = X100 or (CS) = X010;
```

Registers Affected. IC (if the jump is executed)

5.35. Load Status

```
Addr
Mode  Mnemonic          Format/Opcode
                           8       4      4            16
D     LST   ADDR        ---------------------------------------
DX    LST   ADDR,RX     |  7D  | 0000 |  RX  |  |    ADDR    |
                        ---------------------------------------
                           8       4      4            16
I     LSTI  ADDR        ---------------------------------------
IX    LSTI  ADDR,RX     |  7C  | 0000 |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of the Derived Address, DA, and DA+1, and DA+2 are loaded into the Interrupt Mask register, Status Word register and Instruction Counter, respectively. This is a privileged instruction.

Note This instruction is an unconditional jump and is typically used to exit from an interrupt routine. DA, DA+1, and DA+2, in this typical case, contain the Interrupt Mask, Status Word, and Instruction Counter values for the interrupted program and the execution of LST causes the program to return to its status prior to being interrupted.

Register Transfer Description.

```
(MK, SW, IC) <-- [DA, DA+1, DA+2];
```

Registers Affected. MK, SW, IC

5.36. Stack IC and Jump to Subroutine

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4            16
D     SJS   RA,LABEL    ---------------------------------------
DX    SJS   RA,LABEL,RX |  7E  |  RA  |  RX  |  |   LABEL    |
                        ---------------------------------------
```

Description. The contents of register RA are decremented by one. The address of the instruction following the SJS instruction is stored into the memory location pointed to by RA. Program control is then transferred to the instruction at the Derived Address, DA. RA is the stack pointer and can be selected by the programmer as any one of the 16 general registers.

Note If RA = RX, then the derived address, DA, is calculated before RA is decremented.

Register Transfer Description.

```
(RA) <-- (RA) - 1;
[(RA)] <-- (IC);
(IC) <-- DA;
```

Registers Affected. IC, RA

5.37. Unstack IC and Return from Subroutine

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        --------------------
S     URS   RA          |  7F  |  RA  |  0  |
                        --------------------
```

Description. The contents of the memory location pointed to by register RA is loaded into the instruction counter, IC. RA is then incremented by one. Any one of the 16 general registers may be designated as the stack pointer. This instruction is the subroutine return for SJS, Stack and Jump to Subroutine.

Register Transfer Description.

```
(IC) <-- [(RA)];
(RA) <-- (RA) + 1;
```

Registers Affected. RA, IC

5.38. Single Precision Load

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     LR    RA,RB       |  81  |  RA  |  RB  |
                        ----------------------
                           4     2     2        8       12<=BR<=15
                        ------------------------------
B     LB    BR,DSPL     |  0  |  0  |  BR'  |  DSPL  |  BR'=BR-12
                        ------------------------------  RA=R2
                           4     2    2      4    4     12<=BR<=15
                        ------------------------------
BX    LBX   BR,RX       |  4  |  0 |  BR' |  0 |  RX |  BR'=BR-12
                        ------------------------------  RA=R2
                           8      4       4
                        ----------------------
ISP   LISP  RA,N        |  82  |  RA  |  N-1  |         1<=N<=16
                        ----------------------
                           8      4       4
                        ----------------------
ISN   LISN  RA,N        |  83  |  RA  |  N-1  |         1<=N<=16
                        ----------------------
                           8      4      4            16
D     L     RA,ADDR     ------------------------------------
```

```
DX    L     RA,ADDR,RX  |  80  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4           16
IM    LIM   RA,DATA     -------------------------------------
IMX   LIM   RA,DATA,RX  |  85  |  RA  |  RX  |  |    DATA    |
                        -------------------------------------
                           8      4      4           16
I     LI    RA,ADDR     -------------------------------------
IX    LI    RA,ADDR,RX  |  84  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The single precision Derived Operand, DO, is loaded into the register RA. The Condition Status, CS, is set based on the result in register RA.

Register Transfer Description.

```
(RA) <-- DO;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS

5.39. Double Precision Load

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DLR   RA,RB       |  87  |  RA  |  RB  |
                        ----------------------
                           4     2     2        8       12<=BR<=15
                        ------------------------------
B     DLB   BR,DSPL     |  0  |  1  |  BR'  |  DSPL  |  BR'=BR-12
                        ------------------------------  RA=R0
                           4     2     2      4   4     12<=BR<=15
                        ------------------------------
```

```
BX    DLBX  BR,RX       |  4  |  0  |  BR'  | 1 | RX |  BR'=BR-12
                       ------------------------------  RA=R0
                          8      4      4           16
D     DL    RA,ADDR    ------------------------------------
DX    DL    RA,ADDR,RX |  86  |  RA  |  RX  |  |    ADDR    |
                       ------------------------------------
                          8      4      4           16
I     DLI   RA,ADDR    ------------------------------------
IX    DLI   RA,ADDR,RX |  88  |  RA  |  RX  |  |    ADDR    |
                       ------------------------------------
```

Description. The double precision Derived Operand, DO, is loaded into the register RA and RA+1 such that the MSH of DO is in RA. The Condition Status, CS, is set based on the result in RA and RA+1.

Register Transfer Description.

```
(RA,RA+1) <-- DO;
(CS) <-- 0010  if (RA,RA+1) = 0 (Double fixed point zero);
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS

5.40. Load Multiple Registers

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4           16
D     LM   N,ADDR       ------------------------------------
DX    LM   N,ADDR,RX    |  89  |  N  |  RX  |  |    ADDR    |
                        ------------------------------------
                                  0 <= N <= 15
```

Description. The contents of the Derived Address, DA, are loaded into register R0, then the contents of the DA+1 are loaded into register R1, ..., finally, the contents of DA+N are loaded into

RN. Effectively, this instruction allows the transfer of (N+1) words from memory to the register file.

Register Transfer Description.

```
(R0) <-- [DA];
(R1) <-- [DA + 1];
(R2) <-- [DA + 2];
(RN) <-- [DA + N];
```

Registers Affected. R0 through RN

5.41. Extended Precision Floating Point Load

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4            16
D     EFL   RA,ADDR     ----------------------------------------
DX    EFL   RA,ADDR,RX  |  8A  |  RA  |  RX  |  |     ADDR     |
                        ----------------------------------------
```

Description. The extended precision floating point Derived Operand, DO, is loaded into registers RA, RA+1, and RA+2 such that the most significant 16-bits of the word are loaded into register RA. The condition status, CS, is set based on the results in registers RA, RA+1, and RA+2.

Register Transfer Description.

```
(RA, RA+1, RA+2) <-- DO;
(CS) <-- 0010  if (RA, RA+1, RA+2) = 0;
(CS) <-- 0001  if (RA, RA+1, RA+2) < 0;
(CS) <-- 0100  if (RA, RA+1, RA+2) > 0;
```

Registers Affected. RA, RA+1, RA+2, CS

5.42. Load from Upper Byte

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4           16
D     LUB   RA,ADDR     ---------------------------------------
DX    LUB   RA,ADDR,RX  |  8B  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
                           8      4      4           16
I     LUBI  RA,ADDR     ---------------------------------------
IX    LUBI  RA,ADDR,RX  |  8D  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The MSH (upper byte) of the Derived Operand, DO, is loaded into the LSH (lower byte) of register RA. The MSH (upper byte) of RA is unaffected. The condition status, CS, is set based on the result in RA.

Register Transfer Description.

$(RA)_{8\text{-}15}$ <-- $DO_{0\text{-}7}$;
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0001 if (RA) < 0;
(CS) <-- 0100 if (RA) > 0;

Registers Affected. RA, CS

5.43. Load from Lower Byte

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4           16
D     LLB   RA,ADDR     ---------------------------------------
DX    LLB   RA,ADDR,RX  |  8C  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
                           8      4      4           16
```

```
I     LLBI RA,ADDR      ------------------------------------
IX    LLBI RA,ADDR,RX   |  8E  |  RA  |  RX  |  |    ADDR    |
                        ------------------------------------
```

Description. The LSH (lower byte) of the Derived Operand, DO, is loaded into the LSH (lower byte) of register RA. The MSH (upper byte) of RA is unaffected. The condition status, CS, is set based on the result in RA.

Register Transfer Description.

$(RA)_{8\text{-}15}$ <-- $DO_{8\text{-}15}$;
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0001 if (RA) < 0;
(CS) <-- 0100 if (RA) > 0;

Registers Affected. RA, CS

5.44. Pop Multiple Registers off the Stack

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
S     POPM  RA,RB       |  8F  |  RA  |  RB  |
                        ----------------------
```

Description. For RA <= RB, registers RA through RB are loaded sequentially from a stack in memory using R15 as the stack pointer. For RA > RB, registers RA through R14 and then R0 through RB are loaded sequentially from the stack.

In both cases,

- as each word is popped from the stack, R15 is incremented by one;
- if R15 is included in the transfer, then it is effectively ignored;

- on completion, R15 points to the top word of the stack remaining.

Register Transfer Description.

```
if RA <= RB then
   for i = 0 thru RB - RA do
      begin
      if RA + i /= 15 then (RA + i) <-- [(R15)];
      (R15) <-- (R15) + 1;
      end;
else
      begin
      for i = 0 thru 15 - RA do
         begin
         if RA + i /= 15 then (RA + i) <-- [(R15)];
         (R15) <-- (R15) + 1;
         end;
      for i = 0 thru RB do
         begin
         (i) <-- [(R15)];
         (R15) <-- (R15) + 1;
         end;
      end;
```

Registers Affected. RA through R14, R0 through RB, R15

5.45. Single Precision Store

```
Addr
Mode  Mnemonic          Format/Opcode
                           4     2     2        8       12<=BR<=15
                        -----------------------------
B     STB   BR,DSPL     |  0  |  2  |  BR'  |  DSPL  |  BR'=BR-12
                        -----------------------------  RA=R2
                           4     2     2      4   4     12<=BR<=15
                        -----------------------------
BX    STBX  BR,RX       |  4  |  0  |  BR'  | 2 | RX |  BR'=BR-12
                        -----------------------------  RA=R2
```

```
                              8      4      4           16
D     ST    RA,ADDR      -------------------------------------
DX    ST    RA,ADDR,RX   |  90  |  RA  |  RX  |  |    ADDR    |
                         -------------------------------------
                              8      4      4           16
I     STI   RA,ADDR      -------------------------------------
IX    STI   RA,ADDR,RX   |  94  |  RA  |  RX  |  |    ADDR    |
                         -------------------------------------
```

Description. The contents of the register RA are stored into the Derived Address, DA.

Register Transfer Description.

```
[DA] <-- (RA);
```

Registers Affected. None

5.46. Store a Non-Negative Constant

```
Addr
Mode  Mnemonic           Format/Opcode
                              8      4     4           16
D     STC    N,ADDR      -------------------------------------
DX    STC    N,ADDR,RX   |  91  |  N  |  RX  |  |    ADDR    |
                         -------------------------------------
                              8      4     4           16
I     STCI   N,ADDR      -------------------------------------
IX    STCI   N,ADDR,RX   |  92  |  N  |  RX  |  |    ADDR    |
                         -------------------------------------
```

Description. The constant N, where N is an integer (0 <= N <= 15) is stored at the Derived Address, DA. For the special case of storing zero into memory the mnemonics STZ ADDR,RX for direct addressing and STZI ADDR,RX for indirect addressing may be used. In this special case, the N field equals 0.

Register Transfer Description.

```
[DA] <-- N, where 0 <= N <= 15;
```

Registers Affected. None

5.47. Move Multiple Words, Memory-to-Memory

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
S     MOV  RA,RB        |  93  |  RA  |  RB  |
                        ---------------------
```

Description. This instruction allows the memory-to-memory transfer of N words where N is an integer between zero and 2^{16} - 1 and is represented by the contents of RA+1. The contents of RB are the address of the first word to be transferred and the contents of RA are the address of where the first word is to be transferred. After each word transfer, RA and RB are incremented, and RA+1 is decremented.

Note Any pending interrupts are honored after each single word transfer is completed. The IC points to the current instruction location until the last transfer is completed.

Note RA has a final value of the last stored address plus one; RA+1 has a final value of zero.

Note RB has a final value equal to the address of the last word transferred plus one.

Register Transfer Description.

```
Step 1: [(RA)] <-- [(RB)] if (RA+1) ≥= 0;  Go to Step 4 otherwise;
Step 2: (RA) <-- (RA)+1, (RB) <-- (RB)+1, (RA+1) <-- (RA+1)-1;
Step 3: REPEAT STEPS 1 and 2;
Step 4: Set IC to next instruction address;
```

Registers Affected. RA, RA+1, RB

5.48. Double Precision Store

```
Addr
Mode  Mnemonic          Format/Opcode
                          4     2     2        8       12<=BR<=15
                        ------------------------------
B     DSTB   BR,DSPL    |  0  |  3  |  BR'  |  DSPL  |  BR'=BR-12
                        ------------------------------  RA=R0
                          4     2     2      4   4     12<=BR<=15
                        ------------------------------
BX    DSTX   BR,RX      |  4  |  0  |  BR'  | 3 | RX |  BR'=BR-12
                        ------------------------------  RA=R0
                          8      4      4           16
D     DST    RA,ADDR    ---------------------------------------
DX    DST    RA,ADDR,RX |  96  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
                          8      4      4           16
I     DSTI   RA,ADDR    ---------------------------------------
IX    DSTI   RA,ADDR,RX |  98  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of registers RA and RA+1 are stored at the Derived Address, DA, and DA+1, respectively.

Register Transfer Description.

```
[DA, DA+1] <-- (RA, RA+1);
```

Registers Affected. None

5.49. Store Register Through Mask

```
Addr
Mode  Mnemonic          Format/Opcode
```

```
                                 8      4      4           16
D      SRM    RA,ADDR       -------------------------------------
DX     SRM    RA,ADDR,RX    |  97  |  RA  |  RX  |  |    ADDR    |
                            -------------------------------------
```

Description. The contents of register RA are stored into the Derived Address, DA, through the mask in register RA+1. For each position in the mask that is a one, the corresponding bit of register RA is stored into the corresponding bit of the DA. For each position in the mask that is a zero no change is made to the corresponding bit stored in the DA.

Register Transfer Description.

```
[DA] <-- {[DA] ^ ~(RA+1)} v {[RA] ^ [RA+1]};
(RA+1) = MASK, (RA) = DATA;
or, equivalently,
(RQ) <-- [DA];
(RQ)i <-- (RA)i if (RA+1)i = 1 for i = 0, 1, ..., 15;
[DA] <-- (RQ);
```

Registers Affected. None

5.50. Store Multiple Registers

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4           16
D     STM  N,ADDR       -------------------------------------
DX    STM  N,ADDR,RX    |  99  |   N  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The contents of register R0 are stored into the Derived Address, DA; then the contents of R1 are stored into DA+1; ...; finally, the contents of RN are stored into DA+N where N is an integer, 0 <= N <= 15. Effectively, this instruction allows the transfer of (N+1) words from the register file to memory.

Register Transfer Description.

```
[DA] <-- (R0);
[DA+1] <-- (R1);
[DA+2] <-- (R2);
[DA+N] <-- (RN)  0 <= N <= 15;
```

Registers Affected. None

5.51. Extended Precision Floating Point Store

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4           16
D     EFST   RA,ADDR    ---------------------------------------
DX    EFST   RA,ADDR,RX |  9A  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of registers RA, RA+1, RA+2 are stored at the Derived Address, DA, DA+1, and DA+2.

Register Transfer Description.

```
[DA, DA+1, DA+2] <-- (RA, RA+1, RA+2);
```

Registers Affected. None

5.52. Store into Upper Byte

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4           16
D     STUB   RA,ADDR    ---------------------------------------
DX    STUB   RA,ADDR,RX |  9B  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

```
                              8      4      4          16
I     SUBI   RA,ADDR     -------------------------------------
IX    SUBI   RA,ADDR,RX | 9D  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The LSH (lower byte) of register RA is stored into the MSH (upper byte) of the Derived Address, DA. The LSH (lower byte) of the DA is unchanged.

Register Transfer Description.

$[DA]_{0\text{-}7}$ <-- $(RA)_{8\text{-}15}$;

Registers Affected. None

5.53. Store into Lower Byte

```
Addr
Mode  Mnemonic          Format/Opcode
                              8      4      4          16
D     STLB   RA,ADDR     -------------------------------------
DX    STLB   RA,ADDR,RX | 9C  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                              8      4      4          16
I     SLBI   RA,ADDR     -------------------------------------
IX    SLBI   RA,ADDR,RX | 9E  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The LSH (lower byte) of register RA is stored into the LSH (lower byte) of the Derived Address, DA. The MSH (upper byte) of the DA is unchanged.

Register Transfer Description.

$[DA]_{8\text{-}15}$ <-- $(RA)_{8\text{-}15}$;

Registers Affected. None

5.54. Push Multiple Registers onto the Stack

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
S     PSHM   RA,RB      |  9F  |  RA  |  RB  |
                        ----------------------
```

Description. For RA <= RB, the contents of RB through RA are pushed onto a stack in memory using R15 as the stack pointer. As each register contents are pushed onto the memory stack, R15 is decremented by one word for each word pushed. On completion, R15 points to the last item on the stack, the contents of RA.

For RA > RB, the contents of RB through R0, and then the contents of R15 through RA, are pushed onto the stack. On completion, R15 points to the last item on the stack, the contents of RA.

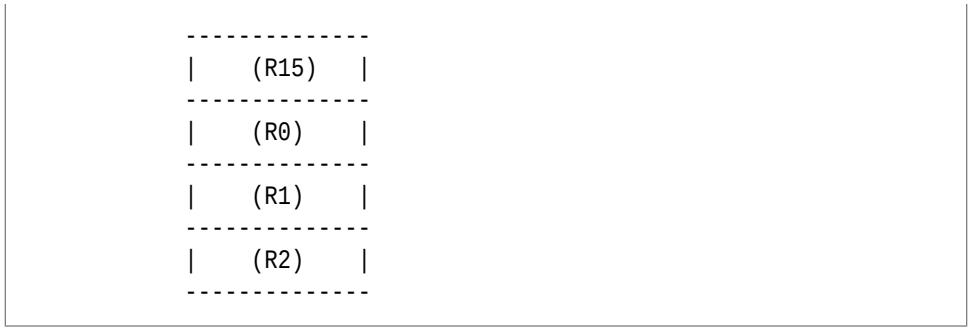
In both cases, successive increasing addresses on the stack correspond to successive increasing register addresses, with a point discontinuity between R15 and R0 in the latter case.

PSHM R3,R5 results in:

```
             --------------
(R15) -->= |    (R3)    |
           --------------
           |    (R4)    |
           --------------
           |    (R5)    |
           --------------
```

PSHM R14,R2 results in:

```
             --------------
(R15) -->= |    (R14)   |
```



Register Transfer Description.

```
if RA <= RB then
   for i = 0 thru RB - RA do
      begin
      (R15) <-- (R15) - 1;
      [(R15)] <-- (RB - i);
      end;
else
      begin
      for i = 0 thru RB do
         begin
         (R15) <-- (R15) - 1;
         [(R15)] <-- (RB - i);
         end;
      for i = 0 thru 15 - RA do
         begin
         (R15) <-- (R15) - 1;
         [(R15)] <-- (R15 - i);
         end;
      end;
```

Registers Affected. R15

5.55. Single Precision Integer Add

```
Addr
```

```
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     AR    RA,RB       |  A1  |  RA  |  RB  |
                        ---------------------
                          4   2   2      8        12<=BR<=15
                        ---------------------
B     AB    BR,DSPL     | 1 | 0 | BR' | DSPL |    BR'=BR-12
                        ---------------------    RA=R2
                          4   2   2     4   4     12<=BR<=15
                        ------------------------
BX    ABX   BR,RX       | 4 | 0 | BR' | 4 | RX |  BR'=BR-12
                        ------------------------  RA=R2
                           8      4      4
                        ----------------------
ISP   AISP  RA,N        |  A2  |  RA  |  N-1  |   1<N<16
                        ----------------------
                           8      4      4           16
D     A     RA,ADDR     -------------------------------------
DX    A     RA,ADDR,RX  |  A0  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4           16
                        -------------------------------------
IM    AIM   RA,DATA     |  4A  |  RA  |  1   |  |    DATA    |
                        -------------------------------------
```

Description. The Derived Operand (DO) is added to the contents of the RA register. The result (a 2's complement sum) is stored in register RA. The condition status (CS) is set based on the result in register RA and carry. A fixed point overflow occurs if both operands are of the same sign and the sum is of opposite sign.

Register Transfer Description.

$(RA)^2$ <-- $(RA)^1$ + DO;
PI_4 <-- 1,
  if $(RA_0)^1 = DO_0$ and $(RA_0)^1$ /= $(RA_0)^2$
(CS) <-- 0010 if carry = 0 and (RA) = 0;
(CS) <-- 0001 if carry = 0 and (RA) < 0;
(CS) <-- 0100 if carry = 0 and (RA) > 0;
(CS) <-- 1010 if carry = 1 and (RA) = 0;

(CS) <-- 1001 if carry = 1 and (RA) < 0;
(CS) <-- 1100 if carry = 1 and (RA) > 0;

Registers Affected. RA, CS, PI

5.56. Increment Memory by a Positive Integer

```
Addr
Mode  Mnemonic          Format/Opcode
                            8      4       4          16
D     INCM   N,ADDR     ---------------------------------------
DX    INCM   N,ADDR,RX  |  A3  |  N-1  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of the memory location specified by the Derived Address, DA, is incremented by N, where N is an integer, 1 <= N <= 16. This instruction adds a positive constant to memory. The condition status, CS, is set based on the results of the addition and carry. A fixed point overflow occurs if the operand in memory is positive and the result is negative. The memory location specified is updated to contain the result of the addition process even if a fixed point overflow occurs.

Register Transfer Description.

$[DA]^2$ <-- $[DA]^1$ + N, where 1 <= N <= 16;
PI_4 <-- 1,
  if $[DA]^2 < 0 < [DA]^1$;
(CS) <-- 0010 if carry = 0 and [DA] = 0;
(CS) <-- 0001 if carry = 0 and [DA] < 0;
(CS) <-- 0100 if carry = 0 and [DA] > 0;
(CS) <-- 1010 if carry = 1 and [DA] = 0;
(CS) <-- 1001 if carry = 1 and [DA] < 0;
(CS) <-- 1100 if carry = 1 and [DA] > 0;

Registers Affected. CS, PI

5.57. Single Precision Absolute Value of Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     ABS   RA,RB       |  A4  |  RA  |  RB  |
                        ----------------------
```

Description. If the sign bit of the Derived Operand, DO (i.e., the sign bit of register RB), is a one, its negative or 2's complement is stored into register RA. However, if the sign bit of DO is a zero, it is stored, unchanged, into RA. The condition status, CS, is set based on the result in register RA.

Note RA may equal RB.

Note The absolute value of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Register Transfer Description.

(RA) <-- |DO|;
PI_4 <-- 1, exit, if DO = 8000_{16};
(CS) <-- 0001 if (RA) = 8000_{16};
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0100 if (RA) > 0;

Registers Affected. RA, CS, PI

5.58. Double Precision Absolute Value of Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
```

```
                          ----------------------
R     DABS   RA,RB        |  A5  |  RA  |  RB  |
                          ----------------------
```

Description. If the sign bit of the double precision Derived Operand, DO (i.e., the sign bit of register (RB,RB+1)), is a one, its negative or 2's complement is stored into register RA and RA+1, such that register RA contains the MSH of the result. However, if the sign bit of DO is a zero, it is stored, unchanged, into RA and RA+1. The condition status, CS, is set based on the result in register RA and RA+1.

Note RA may equal RB.

Note The absolute value of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Register Transfer Description.

(RA, RA+1) <-- |DO|;
PI_4 <-- 1, exit, if DO = 8000 0000_{16};
(CS) <-- 0001 if (RA,RA+1) = 8000 0000_{16};
(CS) <-- 0010 if (RA,RA+1) = 0;
(CS) <-- 0100 if (RA,RA+1) > 0;

Registers Affected. RA, RA+1, CS, PI

5.59. Double Precision Integer Add

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DAR    RA,RB      |  A7  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     DA     RA,ADDR    -------------------------------------
```

```
DX    DA    RA,ADDR,RX  |  A6  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The double precision Derived Operand (DO) is added to the contents of registers RA and RA+1. The result (a 2's complement 32-bit sum) is stored in registers RA and RA+1. The MSH is in RA. The condition status (CS) is set based on the double precision results in RA and RA+1, and carry. A fixed point overflow occurs if both operands are of the same sign and the sum is of opposite sign.

Register Transfer Description.

$(RA,RA+1)^2$ <-- $(RA,RA+1)^1$ + DO;
PI_4 <-- 1 if $(RA_0)^1 = DO_0$ and $(RA_0)^1$ /= $(RA_0)^2$
(CS) <-- 0010 if carry = 0 and (RA,RA+1) = 0;
(CS) <-- 0001 if carry = 0 and (RA,RA+1) < 0;
(CS) <-- 0100 if carry = 0 and (RA,RA+1) > 0;
(CS) <-- 1010 if carry = 1 and (RA,RA+1) = 0;
(CS) <-- 1001 if carry = 1 and (RA,RA+1) < 0;
(CS) <-- 1100 if carry = 1 and (RA,RA+1) > 0;

Registers Affected. RA, RA+1, CS, PI

5.60. Floating Point Add

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     FAR   RA,RB       |  A9  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8
                        -----------------------     12<=BR<=15
B     FAB   BR,DSPL     | 2 | 0 | BR' | DSPL |     BR'=BR-12
                        -----------------------     RA=R0
                          4   2   2     4   4
                        ------------------------    12<=BR<=15
```

```
BX    FABX  BR,RX       | 4 | 0 | BR' | 8 | RX |   BR'=BR-12
                       -----------------------   RA=R0
                          8      4      4           16
D     FA    RA,ADDR    -------------------------------------
DX    FA    RA,ADDR,RX |  A8  |  RA  |  RX  |  |    ADDR    |
                       -------------------------------------
```

Description. The floating point Derived Operand, DO, is floating point added to the contents of registers RA and RA+1. The result is stored in registers RA and RA+1. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent incremented by one for each bit shifted until the exponents are equal. The mantissas are then added. If the sum overflows the 24-bit mantissa, then the sum is shifted right one position, the sign bit restored, and the exponent incremented by one. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the sum does not result in exponent overflow, the result is normalized. If in the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Register Transfer Description.

N = EA - E0;
EA <-- E0,
  if MA = 0;
MO <-- MO Shifted Right Arithmetic n positions,
  if n > 0 and MA /= 0;
MA <-- MA Shifted Right Arithmetic -n positions, EA <-- E0,
  if n < 0 and MO /= 0;
MA <-- MA + MO;

MA <-- MA Shifted Right Arithmetic 1 position, MA_0 <-- ~MA_0, EA <--
    if OVM = 1;
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- $7FFF\ FF_{16}$, exit,
  if EA >= $7F_{16}$ and MA_0 = 0;
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00_{16}, exit,
  if EA >= $7F_{16}$ and MA_0 = 1;
EA, MA <-- normalized EA, MA;

PI_6 <-- 1, EA <-- 0, MA <-- 0,
  if EA < 80_{16};
(CS) <-- 0010 if (RA,RA+1) = 0;
(CS) <-- 0001 if (RA,RA+1) < 0;
(CS) <-- 0100 if (RA,RA+1) > 0;

Registers Affected. RA, RA+1, CS, PI

5.61. Extended Precision Floating Point Add

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     EFAR  RA,RB       |  AB  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     EFA   RA,ADDR     ---------------------------------------
DX    EFA   RA,ADDR,RX  |  AA  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The extended precision floating point Derived Operand, DO, is extended floating point added to the contents of register RA, RA+1, and RA+2. The result is stored in register RA, RA+1, and RA+2. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent is incremented by one for each bit shifted. When the exponents are equal, the mantissas are added. If the sum overflows the 39-bit mantissa, then the sum is shifted right one position, the sign bit restored, and the exponent is incremented by one. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the sum does not result in exponent overflow, the result is normalized. If in the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Register Transfer Description.

```
n = EA - EO;
EA <-- E0,
   if MA = 0;
MO <-- MO Shifted Right Arithmetic n positions,
   if n > 0 and MA /= 0;
MA <-- MA Shifted Right Arithmetic -n positions, EA <-- E0,
   if n < 0 and MO /= 0;
MA <-- MA + MO;

MA <-- MA Shifted Right Arithmetic 1 position, MA0 <-- ~MA0, EA <--
   if OVM = 1;
PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF FFFF16, exit,
   if EA >= 7F16 and MA0 = 0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 00 000016, exit,
   if EA >= 7F16 and MA0 = 1;
EA, MA <-- normalized EA, MA;
PI6 <-- 1, EA <-- 0, MA <-- 0,
   if EA < 8016;
(CS) <-- 0010  if (RA, RA+1, RA+2) = 0;
(CS) <-- 0001  if (RA, RA+1, RA+2) < 0;
(CS) <-- 0100  if (RA, RA+1, RA+2) > 0;
```

Registers Affected. RA, RA+1, RA+2, CS, PI

5.62. Floating Point Absolute Value of Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     FABS  RA,RB       |  AC  |  RA  |  RB  |
                        ---------------------
```

Description. If the sign bit of the mantissa of the Derived Operand, DO (i.e., the contents of registers RB and RB+1), is a one, its floating point negative is stored in registers RA and RA+1. The negative of DO is computed by taking the 2's complement of the mantissa and leaving the exponent unchanged. Exceptions to

this are negative powers of two: -1.0×2^0, -1.0×2^1, ... The absolute value of these are: 0.5×2^1, 0.5×2^2, ..., in other words, the DO mantissa is shifted logically right one position and the exponent incremented. A floating point overflow shall occur if DO is the smallest negative number, -1.0×2^{127}. If the sign bit of DO is a zero, it is stored unchanged into RA and RA+1. The condition status, CS, is set based on the result in register RA and RA+1.

Note RA may equal RB.

Note DO is assumed to be a normalized number or floating point zero.

Register Transfer Description.

```
EA <-- EA+1, MA <-- 4000 00₁₆,
   if MO = 8000 00₁₆;
PI₃ <-- 1, EA <-- 7F₁₆, MA <-- 7FFF FF₁₆, exit,
   if EA >= 7F₁₆;
EA <-- EO, MA <-- -MO,
   if MO < 0,
   if MO /= 8000 00₁₆;
EA <-- EO, MA <-- MO,
   if MO > 0;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.63. Single Precision Integer Subtract

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     SR    RA,RB       |  B1  |  RA  |  RB  |
                        ----------------------
```

```
                          4   2   2     8          12<=BR<=15
                        ---------------------
B     SBB   BR,DSPL     | 1 | 1 | BR' | DSPL |     BR'=BR-12
                        ---------------------      RA=R2
                          4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    SBBX  BR,RX       | 4 | 0 | BR' | 5 | RX |   BR'=BR-12
                        ------------------------   RA=R2
                           8      4      4
                        ----------------------
ISP   SISP  RA,N        |  B2  |  RA  |  N-1  |    1<=N<=16
                        ----------------------
                           8      4      4           16
D     S     RA,ADDR     ------------------------------------
DX    S     RA,ADDR,RX  |  B0  |  RA  |  RX  |  |    ADDR    |
                        ------------------------------------
                           8      4      4           16
                        ------------------------------------
IM    SIM   RA,DATA     |  4A  |  RA  |  2   |  |    ADDR    |
                        ------------------------------------
```

Description. The Derived Operand (DO) is subtracted from the contents of the RA register. The result, a 2's complement difference, is stored in RA. The condition status (CS) is set based on the result in register RA and carry. A fixed point overflow occurs if both operands are of opposite signs and the derived operand is the same as the sign of the difference.

Register Transfer Description.

$(RA)^2$ <-- $(RA)^1$ - DO,
i.e., (RA) - DO means {(RA) + ~DO} + 1;
PI_4 <-- 1,
  if $(RA_0)^1$ /= DO_0 and $(RA_0)^2$ = DO_0
(CS) <-- 0010 if carry = 0 and (RA) = 0;
(CS) <-- 0001 if carry = 0 and (RA) < 0;
(CS) <-- 0100 if carry = 0 and (RA) > 0;
(CS) <-- 1010 if carry = 1 and (RA) = 0;
(CS) <-- 1001 if carry = 1 and (RA) < 0;
(CS) <-- 1100 if carry = 1 and (RA) > 0;

Registers Affected. RA, CS, PI

5.64. Decrement Memory by a Positive Integer

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4       4          16
D     DECM  N,ADDR      ---------------------------------------
DX    DECM  N,ADDR,RX   |  B3  |  N-1  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of the memory location specified by the Derived Address, DA, are decremented by N, where N is an integer, 1 <= N <= 16. This is equivalent of a "subtract-from-memory instruction". The condition status, CS, is set based on the results of the subtraction and carry. A fixed point overflow occurs if the operand in memory is negative and the result is positive. The memory location specified is updated to contain the result of the subtraction process even if a fixed point overflow occurs.

Register Transfer Description.

$[DA]^2$ <-- $[DA]^1$ - N, where 1 <= N <= 16;
PI_4 <-- 1,
  if $[DA_0]^1 < 0 < [DA_0]^2$;
(CS) <-- 0010 if carry = 0 and [DA] = 0;
(CS) <-- 0001 if carry = 0 and [DA] < 0;
(CS) <-- 0100 if carry = 0 and [DA] > 0;
(CS) <-- 1010 if carry = 1 and [DA] = 0;
(CS) <-- 1001 if carry = 1 and [DA] < 0;
(CS) <-- 1100 if carry = 1 and [DA] > 0;

Registers Affected. CS, PI

5.65. Single Precision Negate Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     NEG  RA,RB        |  B4  |  RA  |  RB  |
                        ----------------------
```

Description. The negative (i.e., the 2's complement) of the Derived Address, DO (i.e., the contents of register RB), is stored into register RA. The condition status, CS, is set based on the result in register RA.

Note The negative of zero is zero.

Note The negative of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Register Transfer Description.

(RA) <-- -DO;
PI_4 <-- 1, exit, if DO = 8000_{16};
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0001 if (RA) < 0;
(CS) <-- 0100 if (RA) > 0;

Registers Affected. RA, CS, PI

5.66. Double Precision Negate Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
```

```
R     DNEG   RA,RB      |  B5  |  RA  |  RB  |
                        ----------------------
```

Description. The negative (i.e., the 2's complement) of the Derived Operand, DO (i.e., the contents of register RB and RB+1), is stored into register RA and RA+1 such that register RA contains the MSH of the result. The condition status, CS, is set based on the result in register RA and RA+1.

Note The negative of zero is zero.

Note The negative of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Register Transfer Description.

```
(RA, RA+1) <-- -DO;
PI4 <-- 1, exit, if DO = 8000 0000₁₆;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.67. Double Precision Integer Subtract

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DSR   RA,RB       |  B7  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     DS    RA,ADDR     ---------------------------------------
DX    DS    RA,ADDR,RX  |  B6  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The double precision Derived Operand, DO, is subtracted from the contents of registers RA and RA+1. The results, a 2's complement 32-bit difference, is stored in registers RA and RA+1. The MSH is RA. The condition status (CS) is set based on the double precision results in RA and RA+1, and carry. A fixed point overflow occurs if both operands are of opposite sign and the derived operand is the same as the sign of the difference.

Register Transfer Description.

$(RA,RA+1)^2$ <-- $(RA,RA+1)^1$ - DO,
i.e., (RA,RA+1) - DO means {(RA,RA+1) + ~DO} + 1;
PI_4 <-- 1,
if $(RA_0)^1$ /= DO_0 and $(RA_0)^2$ = DO_0;
(CS) <-- 0010 if carry = 0 and (RA,RA+1) = 0;
(CS) <-- 0001 if carry = 0 and (RA,RA+1) < 0;
(CS) <-- 0100 if carry = 0 and (RA,RA+1) > 0;
(CS) <-- 1010 if carry = 1 and (RA,RA+1) = 0;
(CS) <-- 1001 if carry = 1 and (RA,RA+1) < 0;
(CS) <-- 1100 if carry = 1 and (RA,RA+1) > 0;

Registers Affected. RA, RA+1, CS, PI

5.68. Floating Point Subtract

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     FSR    RA,RB      |  B9  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8          12<=BR<=15
                        ----------------------
B     FSB    BR,DSPL    | 2 | 1 | BR' | DSPL |     BR'=BR-12
                        ----------------------     RA=R0
                          4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    FSBX   BR,RX      | 4 | 0 | BR' | 9 | RX |   BR'=BR-12
```

```
                                 ----------------------   RA=R0
                                   8       4       4        16
D       FS      RA,ADDR          ------------------------------------
DX      FS      RA,ADDR,RX     |  B8   |  RA   |  RX   |  |    ADDR    |
                                 ------------------------------------
```

Description. The floating point Derived Operand, DO, is floating point subtracted from the contents of registers RA and RA+1. The result is stored in registers RA and RA+1. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent incremented by one for each bit shifted until the exponents are equal. The mantissa of the DO is then subtracted from (RA,RA+1). If the difference overflows the 24-bit mantissa, then it is shifted right one position, the sign bit restored, and the exponent incremented by one. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the sum does not result in exponent overflow, the result is normalized. If during the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Register Transfer Description.

```
n = EA - EO;
EA <-- E0,
   if MA = 0;
MO <-- MO Shifted Right Arithmetic n positions,
   if n > 0 and MA /= 0;
MA <-- MA Shifted Right Arithmetic -n positions, EA <-- EO,
   if n < 0 and MO /= 0;
MA <-- MA - MO;
MA <-- MA Shifted Right Arithmetic 1 position, MA0 <-- ~MA0, EA <-- EA
   if OVM = 1;
PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF16, exit,
   if EA >= 7F16 and MA0 = 0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 0016, exit,
   if EA >= 7F16 and MA0 = 1;
EA, MA <-- normalized EA, MA;
PI6 <-- 1, EA <-- 0, MA <-- 0,
```

```
   if EA < 80₁₆;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.69. Extended Precision Floating Point Subtract

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     EFSR   RA,RB      |  BB  |  RA  |  RB  |
                        ----------------------
                           8      4      4          16
D     EFS    RA,ADDR    ---------------------------------------
DX    EFS    RA,ADDR,RX |  BA  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The extended precision floating point Derived Operand, DO, is extended floating point subtracted from the contents of registers RA, RA+1, and RA+2. The result is stored in registers RA, RA+1, and RA+2. The process of this operation is as follows: The mantissa of the number with the smaller algebraic exponent is shifted right and the exponent is incremented by one for each bit shifted. When the exponents are equal, the mantissas are subtracted. If the difference overflows the 39-bit mantissa, then the difference is shifted right one position, the sign bit restored, and the exponent is incremented. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the difference does not result in exponent overflow, the result is normalized. If during the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Register Transfer Description.

```
n = EA - E0;
EA <-- E0,
   if MA = 0;
MO <-- MO Shifted Right Arithmetic n positions,
   if n > 0 and MA /= 0;
MA <-- MA Shifted Right Arithmetic -n positions, EA <-- E0,
   if n < 0 and MO /= 0;
MA <-- MA - MO;
MA <-- MA Shifted Right Arithmetic 1 position, MA0 <-- ~MA0, EA <-- EA
   if OVM = 1;
PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF FFFF16, exit,
   if EA >= 7F16 and MA0 = 0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 00 000016, exit,
   if EA >= 7F16 and MA0 = 1;
EA, MA <-- normalized EA, MA;
PI6 <-- 1, EA <-- 0, MA <-- 0,
   if EA < 8016;
(CS) <-- 0010  if (RA,RA+1,RA+2) = 0;
(CS) <-- 0001  if (RA,RA+1,RA+2) < 0;
(CS) <-- 0100  if (RA,RA+1,RA+2) > 0;
```

Registers Affected. RA, RA+1, RA+2, CS, PI

5.70. Floating Point Negate Register

Addr Mode	Mnemonic	Format/Opcode	8	4	4
R	FNEG RA,RB		BC	RA	RB

Description. The 24-bit mantissa of the Derived Operand, DO, i.e., the floating point number in registers RB and RB+1, is 2's complemented. The exponent remains unchanged. The result, the negative of the original number, is stored in RA and RA+1. The 2's complement of a floating point zero is a floating point zero. Exceptions to this are all powers of two: -1.0×2^{n} and $0.5 \times 2^{n-1}$,

i.e., when the mantissa either $8000\ 00_{16}$ or $4000\ 00_{16}$. The negation of 0.5×2^n is $-1.0 \times 2^{n-1}$, i.e., the mantissa is shifted left one position and the exponent decremented by one. Conversely, the negation of -1.0×2^n is $0.5 \times 2^{n-1}$; i.e., the mantissa is shifted right one position and the exponent is incremented by one. A floating point overflow occurs for the negation of the smallest negative number, -1.0×2^{127}. A floating point underflow occurs for the negation of the smallest positive number, 0.5×2^{-128}, and causes the result to be zero. The condition status, CS, is set based on the result in registers RA and RA+1.

Note RA may equal RB.

Register Transfer Description.

PI_3 <-- 1, EA <-- $7F_{16}$, MO <-- 7FFF FF_{16}, exit,
  if DO = 8000 $007F_{16}$;
PI_3 <-- 1, EA <-- 0, MA <-- 0, exit,
  if DO = 4000 0080_{16};
EA <-- EO+1, MA <-- 4000 00_{16},
  if MO = 8000 00_{16};
EA <-- EO-1, MA <-- 8000 00_{16},
  if MO = 4000 00_{16};
EA <-- EO, MA <-- -MO,
  if MO /= 8000 00_{16} or 4000 00_{16};
(CS) <-- 0010 if (RA,RA+1) = 0;
(CS) <-- 0001 if (RA,RA+1) < 0;
(CS) <-- 0100 if (RA,RA+1) > 0;

Registers Affected. RA, RA+1, CS, PI

5.71. Single Precision Integer Multiply with 16-Bit Product

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     MSR   RA,RB       |  C1  |  RA  |  RB  |
```

```
                          ---------------------
                            8      4      4
                          ---------------------
ISP   MISP  RA,N         |  C2  |  RA  |  N-1 |    1 <= N <= 16
                          ---------------------
                            8      4      4
                          ---------------------
ISN   MISN  RA,N         |  C3  |  RA  |  N-1 |    1 <= N <= 16
                          ---------------------
                            8      4      4           16
D     MS    RA,ADDR      ------------------------------------
DX    MS    RA,ADDR,RX   |  C0  |  RA  |  RX  |  |    ADDR    |
                          ------------------------------------
                            8      4      4           16
                          ------------------------------------
IM    MSIM  RA,DATA      |  4A  |  RA  |  4   |  |    DATA    |
                          ------------------------------------
```

Description. The Derived Operand, DO, is multiplied by the contents of register RA. The LSH of the result, a 16-bit, 2's complement integer, is stored in register RA. The Condition Status, CS, is set based on the result in register RA. A fixed point overflow occurs if (1) both operands are of the same sign and the MSH of the product is not zero, or the sign bit of the LSH is not zero, or (2) if the operands are of opposite sign and the MSH of the product is not $FFFF_{16}$, or the sign bit of the LSH is not one. A fixed point overflow does not occur if either of the operands is zero.

Register Transfer Description.

$(RQ,RQ+1)^1$ <-- (RA) x DO;
$(RA)^2$ <-- (RQ+1);
PI_4 <-- 1,
  if {$(RA_0)^1$ = DO_0 and {(RQ) /= 0 or $(RQ+1_0)$ = 1}} or
  {$(RA_0)^1$ /= DO_0 and {(RQ) /= $FFFF_{16}$ or $(RQ+1_0)$ = 0} and
  {$(RA)^1$ /= 0 and DO /= 0}};
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0001 if (RA) < 0;
(CS) <-- 0100 if (RA) > 0;

Registers Affected. RA, CS, PI

5.72. Single Precision Integer Multiply with 32-Bit Product

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     MR    RA,RB       |  C5  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8          12<=BR<=15
                        ----------------------
B     MB    BR,DSPL     | 1 | 2 | BR' | DSPL |     BR'=BR-12
                        ----------------------     RA=R2
                          4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    MBX   BR,RX       | 4 | 0 | BR' | 6 | RX |   BR'=BR-12
                        ------------------------   RA=R2
                           8      4      4           16
D     M     RA,ADDR     -------------------------------------
DX    M     RA,ADDR,RX  |  C4  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4           16
                        -------------------------------------
IM    MIM   RA,DATA     |  4A  |  RA  |  3   |  |    DATA    |
                        -------------------------------------
```

Description. The Derived Operand, DO, is multiplied by the contents of register RA. The result, a 32-bit, 2's complement integer, is stored in registers RA and RA+1 with the MSH of the product in register RA. The Condition Status, CS, is set based on the result in registers RA and RA+1.

SPECIAL CASE: DO = (RA) = 8000 (the largest negative number), then DO x (RA) = 4000 0000.

Register Transfer Description.

```
(RA,RA+1) <-- (RA) x DO;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS

5.73. Double Precision Integer Multiply

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DMR  RA,RB        |  C7  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     DM   RA,ADDR      ---------------------------------------
DX    DM   RA,ADDR,RX   |  C6  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The double precision Derived Operand, DO, a 32-bit 2's complement number, is multiplied by the contents of registers RA and RA+1, a 32-bit 2's complement number, with the MSH in RA. The LSH of the product is retained in RA and RA+1 as a 32-bit, 2's complement number. The MSH is lost. The Condition Status, CS, is set based on the double precision result in registers RA and RA+1. A fixed point overflow occurs if (1) both operands are of the same sign and the MSH of the product is not zero, or the sign bit of the LSH is not zero, or (2) if the operands are of opposite sign and the MSH of the product is not FFFF $FFFF_{16}$, or the sign bit of the LSH is not one. A fixed point overflow does not occur if either of the operands is zero.

Register Transfer Description.

(RQ,RQ+1,RQ+2,RQ+3) <-- $(RA,RA+1)i^1$ x DO;
$(RA,RA+1)^2$ <-- (RQ+2,RQ+3);
PI_4 <-- 1,
  if $\{(RA_0)^1$ = DO and {(RQ,RQ+1) /= 0 or $(RQ+2_0) = 1\}\}$ or
  $\{(RA_0)^1 /= DO_0$ and
  {(RQ,RQ+1) /= FFFF $FFFF_{16}$ or $(RQ+2_0) = 0\}$ and
  $\{(RA)^1 /= 0$ and DO /= 0}};
(CS) <-- 0010 if (RA,RA+1) = 0;

```
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.74. Floating Point Multiply

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     FMR   RA,RB       |  C9  |  RA  |  RB  |
                        ---------------------
                          4   2   2     8          12<=BR<=15
                        ----------------------
B     FMB   BR,DSPL     | 2 | 2 | BR' | DSPL |     BR'=BR-12
                        ----------------------     RA=R0
                          4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    FMBX  BR,RX       | 4 | 0 | BR' | A | RX |   BR'=BR-12
                        ------------------------   RA=R0
                           8      4      4           16
D     FM    RA,ADDR     ------------------------------------
DX    FM    RA,ADDR,RX  |  C8  |  RA  |  RX  |  |    ADDR    |
                        ------------------------------------
```

Description. The floating point Derived Operand, DO, is floating point multiplied by the contents of register RA and RA+1. The result is stored in register RA and RA+1. The process of the operation is as follows: the exponents of the operands are added. If the sum exceeds $7F_{16}$, a floating point overflow occurs. If the sum is less than 80_{16}, then underflow occurs and the result set to zero. The operand mantissas are multiplied and the result normalized and stored in RA and RA+1. An exceptional case is when both operands are negative powers of two: $(-1.0 \times 2^{n}) \times (-1.0 \times 2^{m})$; the result is a $0.5 \times 2^{n+m+1}$. If $n+m = 7F_{16}$, this shall yield an exponent overflow, floating point overflow occurs. Also, if is possible that the normalization process may yield an exponent

underflow; if this occurs, then the result is forced to zero. The condition status, CS, is set based on the result in RA and RA+1.

Register Transfer Description.

```
n = EA + EO;
PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF16, exit,
   if n >= 7F16 and MA0 = MO0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 0016, exit,
   if n >= 7F16 and MA0 /= MO0;
PI6 <-- 1, EA <-- 0, MA <-- 0, exit,
   if n < 8016;
MP <-- MA x MO;  (integer multiply)
MP <-- MP shift left 1 position;
n <-- n + 1, MP0-23 <-- 4000 0016,
   if MP0-23  = 8000 0016;
PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF16, exit,
   if n >= 7F16 and MP0 = 0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 0016, exit,
   if n >= 7F16 and MP0 = 1;
n,MP <-- normalized n,MP;
PI6 <-- 1, EA <-- 0, MA <-- 0, exit,
   if n < 8016;
EA <-- n;
MA <-- MP0-23;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.75. Extended Precision Floating Point Multiply

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     EFMR   RA,RB      |  CB  |  RA  |  RB  |
```

```
                          ---------------------
                            8      4      4          16
D      EFM     RA,ADDR    -------------------------------------
DX     EFM     RA,ADDR,RX |  CA  |  RA  |  RX  |  |    ADDR    |
                          -------------------------------------
```

Description. The extended precision floating Derived Operand, DO, is extended floating point multiplied by the contents of registers RA, RA+1, and RA+2. The result is stored in registers RA, RA+1, and RA+2. The process of the operation is as follows: the exponent of the operands are added. If the sum exceeds $7F_{16}$, a floating point overflow occurs. If the sum is less than 80_{16}, then underflow occurs and the result set to zero. The operand mantissas are multiplied and the result normalized and stored in RA, RA+1, and RA+2. The condition status, CS, is set based on the result in RA, RA+1, and RA+2.

Register Transfer Description.

n = EA + EO;
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF $FFFF_{16}$, exit,
  if n >= $7F_{16}$ and MA_0 = MO_0;
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00 0000_{16}, exit,
  if n >= $7F_{16}$ and MA_0 /= MO_0;
PI_6 <-- 1, EA <-- 0, MA <-- 0, exit,
  if n < 80_{16};
MP <-- MA x MO; (integer multiply)
MP <-- MP shift left 1 position;
n <-- n + 1, $MP_{0\text{-}39}$ <-- 4000 00 0000_{16},
  if $MP_{0\text{-}39}$ = 8000 00 0000_{16};
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF $FFFF_{16}$, exit,
  if n >= $7F_{16}$ and MP_0 = 0;
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00 0000 , exit,
  if n >= $7F_{16}$ and MP_0 = 1;
n, MP <-- normalized n, MP;
PI_6 <-- 1, EA <-- 0, MA <-- 0,
  if n < 80_{16};
EA <-- n;
MA <-- $MP_{0\text{-}39}$;
(CS) <-- 0010 if (RA,RA+1,RA+2) = 0;

```
(CS) <-- 0001  if (RA,RA+1,RA+2) < 0;
(CS) <-- 0100  if (RA,RA+1,RA+2) > 0;
```

Registers Affected. RA, RA+1, RA+2, CS, PI

5.76. Single Precision Integer Divide with 16-Bit Dividend

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DVR   RA,RB       |  D1  |  RA  |  RB  |
                        ----------------------
                           8      4      4
                        ----------------------
ISP   DISP  RA,N        |  D2  |  RA  |  N-1 |    1 <= N <= 16
                        ----------------------
                           8      4      4
                        ----------------------
ISN   DISN  RA,N        |  D3  |  RA  |  N-1 |    1 <= N <= 16
                        ----------------------
                           8      4      4           16
D     DV    RA,ADDR     -------------------------------------
DX    DV    RA,ADDR,RX  |  DO  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4           16
                        -------------------------------------
IM    DVIM  RA,DATA     |  4A  |  RA  |  6   |  |    DATA    |
                        -------------------------------------
```

Description. The contents of register RA are divided by the Derived Operand, DO, a single precision, 2's complement number. The result is stored in registers RA and RA+1 such that RA stores the single precision integer quotient and RA+1 stores the remainder. The Condition Status, CS, is set based on the result in RA. A fixed point overflow occurs if the divisor, DO, is zero, or if the dividend is 8000_{16} and the divisor is $FFFF_{16}$.

Note The sign of the non-zero remainder is the same as the sign of the dividend.

Register Transfer Description.

(RA,RA+1) <-- (RA) / DO;
PI_4 <-- 1,
  if DO = 0 or {RA = 8000_{16} and DO = $FFFF_{16}$};
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0001 if (RA) < 0;
(CS) <-- 0100 if (RA) > 0;

Registers Affected. RA, RA+1, CS, PI

5.77. Single Precision Integer Divide with 32-Bit Dividend

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DR    RA,RB       |  D5  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8        12<=BR<=15
                        ----------------------
B     DB    BR,DSPL     | 1 | 3 | BR' | DSPL |   BR'=BR-12
                        ----------------------   RA=R2
                          4   2   2     4   4    12<=BR<=15
                        ------------------------
BX    DBX   BR,RX       | 4 | 0 | BR' | 7 | RX |  BR'=BR-12
                        ------------------------  RA=R2
                           8      4      4          16
D     D     RA,ADDR     -------------------------------------
DX    D     RA,ADDR,RX  |  D4  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4          16
                        -------------------------------------
IM    DIM   RA,DATA     |  4A  |  RA  |  5   |  |    DATA    |
                        -------------------------------------
```

Description. The contents of registers RA and RA+1, a double precision 2's complement number, are divided by the Derived Operand, DO, a single precision, 2's complement number. RA contains the MSH of the 32-bit dividend. The result is stored in registers RA and RA+1 such that RA stores the single precision integer quotient and RA+1 stores the remainder. The Condition Status, CS, is set based on the result in RA. A fixed point overflow occurs if the divisor equals zero or if a positive quotient exceeds $7FFF_{16}$ or a negative quotient is less than 8000_{16}.

Note The sign of the non-zero remainder is the same as that of the dividend.

Register Transfer Description.

```
(RQ, RQ+1, RR) <-- (RA,RA+1) / DO;
PI4 <-- 1,
   if DO = 0 or (RQ, RQ+1) > 0000 7FFF16 or (RQ, RQ+1) < FFFF 800016
(RA) <-- (RQ+1)
(RA+1) <-- (RR)
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.78. Double Precision Integer Divide

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DDR   RA,RB       |  D7  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     DD    RA,ADDR     ---------------------------------------
DX    DD    RA,ADDR,RX  |  D6  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of registers RA and RA+1, a double precision 2's complement number, are divided by the Derived Operand, DO, a double precision 2's complement number. RA contains the MSH of the 32-bit dividend. The quotient part of the integer result is stored in registers RA and RA+1 (with the MSH in RA) and the remainder is lost. The Condition Status, CS, is set based on the results in registers RA and RA+1. A fixed point overflow occurs if the divisor, DO, is zero, or if the dividend is $8000\ 0000_{16}$ and the divisor is $FFFF\ FFFF_{16}$.

Register Transfer Description.

```
(RA,RA+1) <-- (RA,RA+1) / DO;
PI4 <-- 1,
   if DO = 0 or {RA, RA+1 = 8000 0000₁₆ and DO = FFFF FFFF₁₆};
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.79. Floating Point Divide

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     FDR   RA,RB       |  D9  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8         12<=BR<=15
                        ----------------------
B     FDB   BR,DSPL     | 2 | 3 | BR' | DSPL |    BR'=BR-12
                        ----------------------    RA=R0
                          4   2   2     4   4     12<=BR<=15
                        ------------------------
BX    FDBX  BR,RX       | 4 | 0 | BR' | B | RX |  BR'=BR-12
                        ------------------------  RA=R0
                           8      4      4           16
D     FD    RA,ADDR     -------------------------------------
```

```
DX    FD    RA,ADDR,RX  |  D8  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The floating point number in registers RA and RA+1 is divided by the floating point Derived Operand, DO. The result is stored in register RA and RA+1. A floating point overflow occurs if the exponent result exceeds $7F_{16}$ at any point in the calculation process. Underflow occurs if the exponent result is less than 80_{16} at any point in the process. If underflow occurs, then the quotient is forced to zero. A divide by zero yields a floating point overflow.

Register Transfer Description.

```
n = EA - E0;
n <-- 0,
   if MA = 0
PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF16, exit,
   if MA0 = MO0 and {n >= 7F16 or DO = 0};
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 0016, exit,
   if MA0 /= MO0 and {n >= 7F16 or DO = 0};
PI6 <-- 1, EA <-- 0, MA <-- 0, exit,
   if n < 8016;
MQ <-- MA / MO;
MQ <-- MQ Shift Right Arithmetic 1 position, n <-- n + 1,
   if MQ >== 1.0;

PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF16, exit,
   if n >= 7F16 and MQ0 = 0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 0016, exit,
   if n >= 7F16 and MQ0 = 1;
EA <-- n;
MA <-- MQ0-23;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.80. Extended Precision Floating Point Divide

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     EFDR   RA,RB      |  DB  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     EFD    RA,ADDR    ---------------------------------------
DX    EFD    RA,ADDR,RX |  DA  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The contents of registers RA, RA+1, and RA+2 are extended precision floating point divided by the extended precision floating point Derived Operand, DO. The result is stored in register RA, RA+1, and RA+2. A floating point overflow occurs if the exponent result exceeds $7F_{16}$ at any point in the calculation process. Underflow occurs if the exponent result is less than 80_{16} at any point in the process. If underflow occurs, then the quotient is forced to zero. A divide by zero yields a floating point overflow.

Register Transfer Description.

n = EA - E0;
n <-- 0,
  if MA = 0;
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF $FFFF_{16}$, exit,
  if MA_0 = MO_0 and {n >= $7F_{16}$or DO = 0};
PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00 0000_{16}, exit,
  if MA_0 /= MO_0 and {n >= $7F_{16}$or DO = 0};
PI_6 <-- 1, EA <-- 0, MA <-- 0, exit,
  if n < 80_{16};
MQ <-- MA / MO;
MQ <-- MQ Shift Right Arithmetic 1 position, n <-- n + 1,
  if MQ >== 1.0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF $FFFF_{16}$, exit,

```
   if n >= 7F16 and MQ0 = 0;
PI3 <-- 1, EA <-- 7F16, MA <-- 8000 00 000016, exit,
   if n >= 7F16 and MQ0 = 1;
EA <-- n;
MA <-- MQ0-39;
(CS) <-- 0010  if (RA,RA+1,RA+2) = 0;
(CS) <-- 0001  if (RA,RA+1,RA+2) < 0;
(CS) <-- 0100  if (RA,RA+1,RA+2) > 0;
```

Registers Affected. RA, RA+1, RA+2, CS, PI

5.81. Inclusive Logical OR

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     ORR    RA,RB      |  E1  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8          12<=BR<=15
                        ----------------------
B     ORB    BR,DSPL    | 3 | 0 | BR' | DSPL |     BR'=BR-12
                        ----------------------     RA=R2
                          4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    ORBX   BR,RX      | 4 | 0 | BR' | F | RX |   BR'=BR-12
                        ------------------------   RA=R2
                           8      4      4           16
D     OR     RA,ADDR    -------------------------------------
DX    OR     RA,ADDR,RX |  EO  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4           16
                        -------------------------------------
IM    ORIM   RA,DATA    |  4A  |  RA  |  8   |  |    DATA    |
                        -------------------------------------
```

Description. The Derived Operand, DO, is bit-by-bit inclusively ORed with the contents of RA. The result is stored in register RA. The condition status, CS, is set based on the result in register RA.

Register Transfer Description.

```
(RA) <-- (RA) v DO;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS

5.82. Logical AND

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     ANDR   RA,RB      |  E3  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8         12<=BR<=15
                        ----------------------
B     ANDB   BR,DSPL    | 3 | 1 | BR' | DSPL |    BR'=BR-12
                        ----------------------    RA=R2
                          4   2   2     4   4     12<=BR<=15
                        ------------------------
BX    ANDX   BR,RX      | 4 | 0 | BR' | E | RX |  BR'=BR-12
                        ------------------------  RA=R2
                           8      4      4           16
D     AND    RA,ADDR    -------------------------------------
DX    AND    RA,ADDR,RX |  E2  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
                           8      4      4           16
                        -------------------------------------
IM    ANDM   RA,DATA    |  4A  |  RA  |  7   |  |    DATA    |
                        -------------------------------------
```

Description. The Derived Operand, DO, is bit-by-bit ANDed with the contents of register RA. The result is stored in register RA. The condition status, CS, is set based on the result in register RA.

Register Transfer Description.

```
(RA) <-- (RA) ^ DO;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS

5.83. Exclusive Logical OR

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     XORR   RA,RB      |  E5  |  RA  |  RB  |
                        ---------------------
                           8      4      4           16
D     XOR    RA,ADDR    ---------------------------------------
DX    XOR    RA,ADDR,RX |  E4  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
                           8      4      4           16
                        ---------------------------------------
IM    XORM   RA,DATA    |  4A  |  RA  |  9   |  |    DATA    |
                        ---------------------------------------
```

Description. The Derived Operand, DO, is bit-by-bit exclusively ORed with the contents of RA. The result is stored in RA. The condition status, CS, is set based on the result in RA.

Register Transfer Description.

```
(RA) <-- (RA) XOR DO;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS

5.84. Logical NAND

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     NR   RA,RB        |  E7  |  RA  |  RB  |
                        ----------------------
                           8      4      4          16
D     N    RA,ADDR     ---------------------------------------
DX    N    RA,ADDR,RX  |  E6  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
                           8      4      4          16
                        ---------------------------------------
IM    NIM  RA,DATA     |  4A  |  RA  |  B   |  |    DATA    |
                        ---------------------------------------
```

Description. The Derived Operand, DO, is bit-by-bit logically NANDed with the contents of register RA. The result is stored in RA.

Note The logical NOT of a register can be attained with a NR instruction with RA = RB.

Register Transfer Description.

```
(RA) <-- ~((RA) ^ DO);
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS

5.85. Convert Floating Point to 16-Bit Integer

```
Addr
```

```
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     FIX   RA,RB       |  E8  |  RA  |  RB  |
                        ---------------------
```

Description. The integer portion of the floating point Derived Operand, DO (i.e., the contents of registers RB and RB+1), is stored into register RA. If the actual value of the DO floating point exponent is greater than $0F_{16}$, then RA remains unchanged and a fixed point overflow occurs. The condition status, CS, is set based on the result in RA.

Note The algorithm truncates toward zero.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if EO > 0F16;
(RA) <-- Integer portion of DO;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS, PI

5.86. Convert 16-Bit Integer to Floating Point

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     FLT   RA,RB       |  E9  |  RA  |  RB  |
                        ---------------------
```

Description. The integer Derived Operand, DO (i.e., the contents of register RB), is converted to Single Precision floating point format and stored in register RA and RA+1. The condition status,

CS, is set based on the results in RA and RA+1. The operation process is as follows: The exponent is initially considered to be $0F_{16}$. The integer value in RB is normalized, i.e., the number is left shifted and the exponent decremented for each shift until the sign bit and the next MSB are unequal, and the exponent and mantissa stored in the proper fields of RA and RA+1.

Note RA may equal RB.

Register Transfer Description.

```
EA <-- 0, MA <-- 0, exit,
   if (RB) = 0;
EA <-- 0F16;
MA <-- (RB);
EA, MA <-- normalize EA, MA;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS

5.87. Convert Extended Precision Floating Point to 32-Bit Integer

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
R     EFIX   RA,RB      |  EA  |  RA  |  RB  |
                        ---------------------
```

Description. The integer portion of the floating point Derived Operand, DO (i.e., the contents of registers RB, RB+1, and RB+2), is stored into register RA and RA+1. If the actual value of the DO floating point exponent is greater than $1F_{16}$, then RA and RA+1 remain unchanged and a fixed point overflow occurs. The condition status, CS, is set based on the result in RA and RA+1.

Note The algorithm truncates toward zero.

Register Transfer Description.

```
PI4 <-- 1, exit,
   if EO >= 1F16;
(RA,RA+1) <-- Integer portion of DO;
(CS) <-- 0010  if (RA,RA+1) = 0;
(CS) <-- 0001  if (RA,RA+1) < 0;
(CS) <-- 0100  if (RA,RA+1) > 0;
```

Registers Affected. RA, RA+1, CS, PI

5.88. Convert 32-bit Integer to Extended Precision Floating Point

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     EFLT   RA,RB      |  EB  |  RA  |  RB  |
                        ----------------------
```

Description. The double precision integer Derived Operand, DO (i.e., the contents of registers RB and RB+1), is converted to Extended Precision floating point format and stored in register RA, RA+1, and RA+2. The condition status, CS, is set based on the result in RA, RA+1, and RA+2. The operation process is as follows: The exponent is initially considered to be $1F_{16}$. The integer value in RB, RB+1 is normalized, i.e., the number is left shifted and the exponent decremented for each shift until the sign bit and the next MSB are unequal, and the exponent and mantissa stored in the proper field of RA, RA+1, and RA+2.

Note RA may equal RB.

Register Transfer Description.

```
EA <-- 0, MA <-- 0, exit,
   if (RB,RB+1) = 0;
EA <-- 1F16, MA <-- (RB,RB+1);
EA, MA <-- normalized EA, MA;
(CS) <-- 0010  if (RA,RA+1,RA+2) = 0;
(CS) <-- 0001  if (RA,RA+1,RA+2) < 0;
(CS) <-- 0100  if (RA,RA+1,RA+2) > 0;
```

Registers Affected. RA, RA+1, RA+2, CS

5.89. Exchange Bytes in Register

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ---------------------
S     XBR  RA           |  EC  |  RA  |  0  |
                        ---------------------
```

Description. The upper byte of register RA is exchanged with the lower byte of register RA. The CS is set based on the result in register RA.

Register Transfer Description.

```
(RA)0-7 <-->= (RA)8-15;
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, CS

5.90. Exchange Words in Registers

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     XWR   RA,RB       |  ED  |  RA  |  RB  |
                        ----------------------
```

Description. The contents of register RA are exchanged with the contents of register RB. The CS is set based on the result in register RA.

Register Transfer Description.

```
(RA) <-->= (RB);
(CS) <-- 0010  if (RA) = 0;
(CS) <-- 0001  if (RA) < 0;
(CS) <-- 0100  if (RA) > 0;
```

Registers Affected. RA, RB, CS

5.91. Single Precision Compare

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     CR    RA,RB       |  F1  |  RA  |  RB  |
                        ----------------------
                          4   2   2     8          12<=BR<=15
                        ----------------------
B     CB    BR,DSPL     | 3 | 2 | BR' | DSPL |     BR'=BR-12
                        ----------------------     RA=R2
                          4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    CBX   BR,RX       | 4 | 0 | BR' | C | RX |   BR'=BR-12
```

```
                          ------------------------   RA=R2
                             8      4      4
                          ----------------------
ISP   CISP  RA,N          |  F2  |  RA  |  N-1  |    1<=N<=16
                          ----------------------
                             8      4      4
                          ----------------------
ISN   CISN  RA,N          |  F3  |  RA  |  N-1  |    1<=N<=16
                          ----------------------
                             8      4      4           16
D     C     RA,ADDR       -------------------------------------
DX    C     RA,ADDR,RX    |  F0  |  RA  |  RX  |  |    ADDR    |
                          -------------------------------------
                             8      4      4           16
                          -------------------------------------
IM    CIM   RA,DATA       |  4A  |  RA  |  A   |  |    DATA    |
                          -------------------------------------
```

Description. The single precision Derived Operand, DO, is compared to the contents of RA. Then, the Condition Status, CS, is set based on whether the contents of RA is less than, equal to, or greater than the DO. The contents of RA are unchanged.

Register Transfer Description.

```
(RA) : DO;
(CS) <-- 0010  if (RA) = DO;
(CS) <-- 0001  if (RA) < DO;
(CS) <-- 0100  if (RA) > DO;
```

Registers Affected. CS

5.92. Compare Between Limits

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4           16
D     CBL   RA,ADDR     -------------------------------------
```

```
DX    CBL   RA,ADDR,RX  |  F4  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The contents of register RA are compared to two different sixteen bit derived operands, DO1 and DO2. The derived operands, DO1 and DO2 are located at DA and DA+1, respectively, and their values are defined such that DO1 <= DO2. The CS is set based on the results. If the values for DO1 and DO2 are defined incorrectly (that is, DO1 >= DO2), then CS is set to 1000.

Register Transfer Description.

```
(CS) <-- 1000  if  DO1 > DO2, exit;
(CS) <-- 0001  if (RA) < DO1;
(CS) <-- 0010  if  DO1 ≤ (RA) ≤ DO2;
(CS) <-- 0100  if (RA) > DO2;
```

Registers Affected. CS

5.93. Double Precision Compare

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     DCR   RA,RB       |  F7  |  RA  |  RB  |
                        ----------------------
                           8      4      4          16
D     DC    RA,ADDR     ---------------------------------------
DX    DC    RA,ADDR,RX  |  F6  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The double precision Derived Operand, DO, is compared to the contents of registers RA and RA+1 where RA contains the MSH of a double precision word. Then, the Condition Status, CS, is set based on whether the contents of RA, RA+1 is less than, equal to, or greater than the DO. The contents of RA and RA+1 are unchanged.

Register Transfer Description.

```
(RA,RA+1) : DO;
(CS) <-- 0010  if (RA,RA+1) = DO;
(CS) <-- 0001  if (RA,RA+1) < DO;
(CS) <-- 0100  if (RA,RA+1) >= DO;
```

Registers Affected. CS

5.94. Floating Point Compare

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     FCR    RA,RB      |  F9  |  RA  |  RB  |
                        ----------------------
                           4   2   2     8          12<=BR<=15
                        ----------------------
B     FCB    BR,DSPL    | 3 | 3 | BR' | DSPL |      BR'=BR-12
                        ----------------------      RA=R0
                           4   2   2     4   4      12<=BR<=15
                        ------------------------
BX    FCBX   BR,RX      | 4 | 0 | BR' | D | RX |    BR'=BR-12
                        ------------------------    RA=R0
                            4      8      8           16
D     FC     RA,ADDR    -------------------------------------
DX    FC     RA,ADDR,RX |  F8  |  RA  |  RX  |  |    ADDR    |
                        -------------------------------------
```

Description. The floating point number in registers RA and RA+1 is compared to the floating point Derived Operand, DO. Then, the Condition Status, CS, is set based on whether the contents of RA, RA+1 is less than, equal to, or greater than the DO. The contents of RA and RA+1 are unchanged.

Note This instruction does not cause an overflow to occur.

Register Transfer Description.

```
(RA, RA+1) : DO;
(CS) <-- 0010  if (RA,RA+1) = DO;
(CS) <-- 0001  if (RA,RA+1) < DO;
(CS) <-- 0100  if (RA,RA+1) >= DO;
```

Registers Affected. CS

5.95. Extended Precision Floating Point Compare

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4      4
                        ----------------------
R     EFCR   RA,RB      |  FB  |  RA  |  RB  |
                        ----------------------
                           8      4      4           16
D     EFC    RA,ADDR    ---------------------------------------
DX    EFC    RA,ADDR,RX |  FA  |  RA  |  RX  |  |    ADDR    |
                        ---------------------------------------
```

Description. The extended precision floating Derived Operand, DO, is compared to the contents of registers RA, RA+1, and RA+2 where RA contains the most significant 16-bits of the extended precision floating point word. The condition status, CS, is set based on whether the contents of RA, RA+1, and RA+2 are less than, equal to or greater than the DO. The contents of RA, RA+1, and RA+2 are unchanged.

Note This instruction does not cause overflow to occur.

Register Transfer Description.

```
(RA, RA+1, RA+2) : DO;
(CS) <-- 0010  if (RA, RA+1, RA+2) = DO;
(CS) <-- 0001  if (RA, RA+1, RA+2) < DO;
(CS) <-- 0100  if (RA, RA+1, RA+2) >= DO;
```

Registers Affected. CS

5.96. No Operation

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4
                        -------------------
S     NOP               |  FF  |  0  |  0  |
                        -------------------
```

Description. No operation is performed.

Register Transfer Description.

None

Registers Affected. None

5.97. Break Point

```
Addr
Mode  Mnemonic          Format/Opcode
                           8      4     4
                        -------------------
S     BPT               |  FF  |  F  |  F  |
                        -------------------
```

Description. This instruction is typically used for halting the processor during maintenance and diagnostic procedures when the maintenance console is connected to the system. If the console is not connected, this instruction is treated as a NOP (see Section 5.96, “No Operation” [151]). Restarting the processor after a BPT can only be done by: the maintenance console or the power on sequence.

Register Transfer Description.

None

Registers Affected. None

5.98. Built-In-Function

```
Addr
Mode  Mnemonic          Format/Opcode
                           8         8
                        --------------------
S     BIF Op Ex.        |  4F  |  Op. Ex.  |
                        --------------------
```

Description. This instruction invokes special operations defined by the user. Note that this instruction may use one or more additional words immediately following it, the number and interpretation of which are determined by the Op. Ex.

Register Transfer Description.

User defined.

Index

E

F

I

J

T

U

V

X