MIL-STD-1750A (USAF)
2 July 1980
SUPERSEDING
MIL-STD-1750 (USAF)
21 February 1979

MILITARY STANDARD

SIXTEEN-BIT COMPUTER INSTRUCTION SET ARCHITECTURE

··U.S. GOVERNMENT PRINTING OFFICE: 1980—603-121,3202

FSC IPSC

DEPARTMENT OF DEFENSE
Washington D.C. 20360

Sixteen-bit Computer Instruction Set Architecture MIL-STD-1750A (USAF)

1. This Military Standard is approved for use by the Department of the Air Force, and is available for use by all Departments and Agencies of the Department of Defense.

2. Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to: Aeronautical Systems Division, Attn: ASD/ENESS, Wright Patterson Air Force Base, Ohio 45433, by using the self-addressed Standardization Document Improvement Proposal (DD Form 1426) appearing at the end of this document or by letter.

CONTENTS

FIGURES

TABLES

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.

2 REFERENCED DOCUMENTS

Not applicable.

3 DEFINITIONS

3.1 Accumulator. A register in the arithmetic logic unit used for intermediate storage, algebraic sums and other arithmetic and logical results.

3.2 Address. A number which identifies a location in memory where information is stored.

3.3 Arithmetic logic unit (ALU). That portion of hardware in the central processing unit in which arithmetic and logical operations are performed.

3.4 Avionics. All the electronic and electromechanical 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.

3.5 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.

3.6 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.

3.7 Byte. A group of eight binary digits.

3.8 Central processing unit (CPU). That portion of a computer that controls and performs the execution of instructions.

3.9 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.

3.10 General purpose register. A register that may be used for arithmetic and logical operations, indexing, shifting, input, output, and general storage of temporary data.

3.11 Index register. A register that contains a quantity for modification of an address without permanently modifying the address.

3.12 Input/output (I/O). That portion of a computer which interfaces to the external world.

3.13 Instruction. A program code which tells the computer what to do.

3.14 Instruction counter (IC). A register in the CPU that holds the address of the next instruction to be executed.

3.15 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.

3.16 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.

3.17 Memory. That portion of a computer that holds data and instructions and from which they can be accessed at a later time.

3.18 Operation code (OPCODE). That part of an instruction that defines the machine operation to be performed.

3.19 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.

3.20 Page register. A register which is used to supply additional address bits in paged memory addressing schemes.

3.21 Programmed input/output (PIO). A type of I/O channel that allows program control of information transfer between the computer and an external device.

3.22 Register. A device in the CPU for the temporary storage of one or more words to facilitate arithmetical, logical, or transfer operations.

3.23 Register transfer language (RTL). A language used to describe operations (upon registers) which are caused by the execution of each instruction.

3.24 Reserved. Must not be used.

3.25 Spare. A framework for usage is defined by the standard with particulars to be defined by the application requirements.

3.26 Stack. A sequence of memory locations in which data may be stored and retrieved on a last-in-first-out (LIFO) basis.

3.27 Stack pointer. A register that points to the last item on the stack.

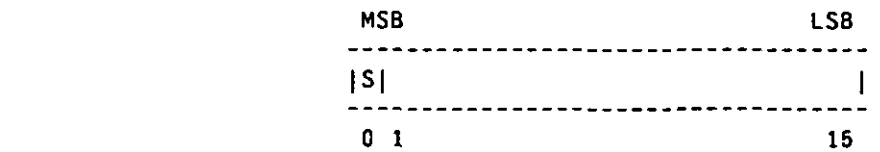
3.28 Status word register. A register whose state is defined by some prior event occurrence in the computer.

3.29 Word. Sixteen bits.

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:

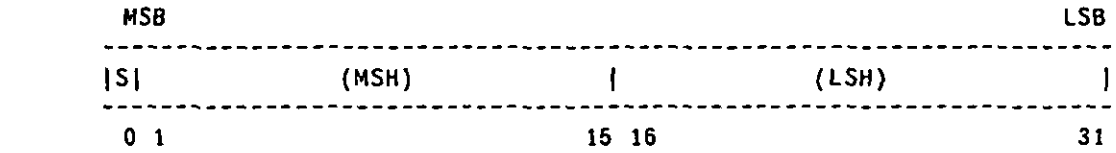


Examples of single precision fixed point numbers are shown in table I.

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
0	0 0 0 0
-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.

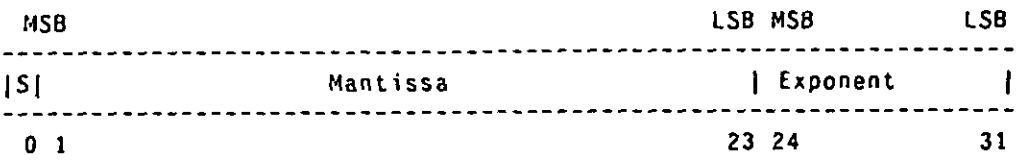
TABLE II. Double precision fixed point numbers

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.

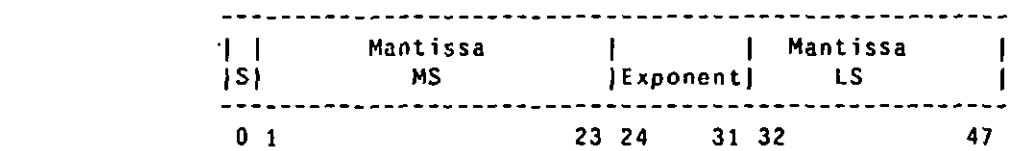


Floating point numbers are represented as a fractional mantissa times 2 raised to the power of the exponent. All floating point numbers are assumed normalized or floating point zero at the beginning of a floating point operation and the results of all floating point operations are normalized (a normalized 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.

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
-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.

TABLE IV. 48-bit extended floating point numbers

Decimal Number	Hexadecimal Notation		
	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
-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 as exponent underflow.

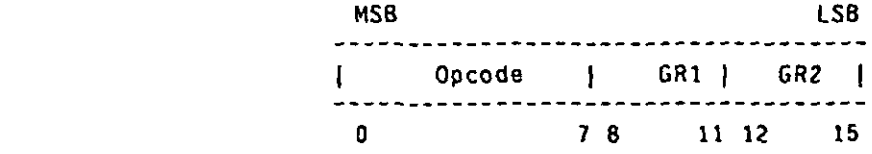
4.1.7 Floating point operands. All operands for floating point instructions must be normalized or a floating point zero. A floating point overflow shall be defined as exponent overflow if the exponent is greater than $7F_{16}$. The results of an operation which causes a floating point overflow shall 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.

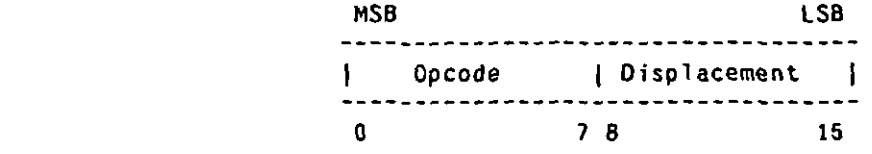
4.1.9 Results of division. The sign of any non-zero remainder is the same as the dividend for all division instructions; the remainder is only accessible for single precision integer divides with 16 bit dividends and for single precision integer divides with 32 bit dividends.

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.

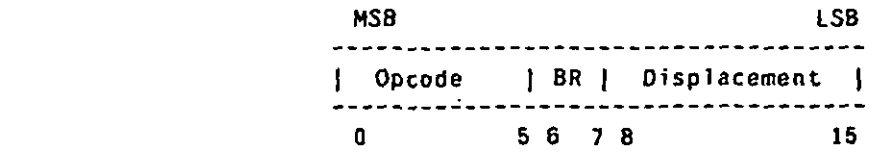
4.2.1 Register-to-register format. The register-to-register format is a 16-bit instruction consisting of an 8-bit opcode and two 4-bit general register (GR) fields that typically specify any of 16 general registers. In addition, these fields may contain a shift count, condition code, opcode extension,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



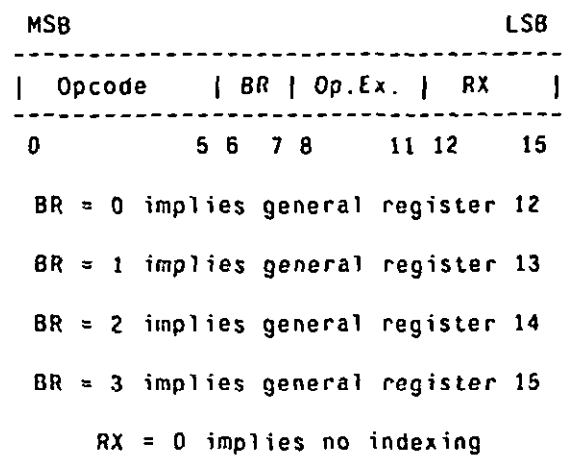
BR = 0 implies general register 12

BR = 1 implies general register 13

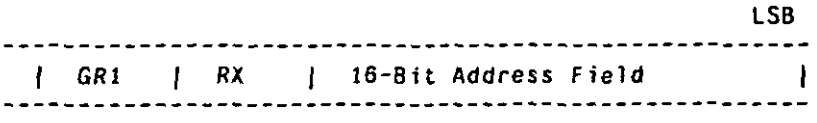
BR = 2 implies general register 14

BR = 3 implies general register 15

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.

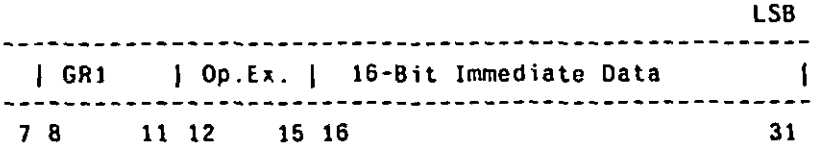


4.2.5 Long instruction format. The Long Instruction Format is a 32-bit instruction consisting of an 8-bit opcode, a 4-bit general register field, a 4-bit index register field and a 16-bit address field.



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.



4.3 Addressing modes. Table V 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.

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.

TABLE V. Addressing modes and instruction word format

MODE	FORMAT	DERIVED OPERAND (DO) S.P.	DERIVED OPERAND (DO) Flt. Pt. and D.P.	DERIVED ADDRESS (DA) S.P.	DERIVED ADDRESS (DA) Flt. Pt. and D.P.	NOTES
1. Register Direct	0 7 8 11 12 15 | O.C. | RA | RX |	(RB)	(RB,RB+1)	4B	RB,RB+1	Extended Precision Floating Point Instructions required address-ing of these operands located at OA+1, and OA+2.
"R"	0 7 8 11 12 15 16 31 | O.C. | RA | RX | A |	[A] [A+(RX)]	[A,A+1] [A+(RX),A+1+(RX)]	A A+(RX)	A,A+1 A+(RX),A+1+(RX)	
2. Memory Direct	RX=0 (Non-Indexed) RX≠0 (Indexed)					
"D"	0 7 8 11 12 15 16 31 | O.C. | RA | RX | A |					
"DX"						
3. Memory Indirect	RX=0 (Non-Indexed) RX≠0 (Indexed)	[[A]] [[A+(RX)]]	[[A],[A]+1] [[A+(RX)],[A+(RX)+1]]	[A] [[A+(RX)]]	[A],[A]+1 [A+(RX)],[A+(RX)+1]	
"I"						
"IX"						
4. Immediate Long	0 7 8 11 12 15 16 31 | O.C. | RA |OCX | 0 |	I	-----	---	-----	
"IM"						
5. Immediate Short	0 7 8 11 12 15 16 31 | O.C. | RA | RX | I |	I	-----	---	-----	
"IM"	RX=0 (Non-Indexed) RX≠0 (Indexed)	I+(RX)	-----	---	-----	
a. Positive	0 7 8 11 12 15 | O.C. | RA | I |	+(I+1)	-----	---	-----	
"ISP"						
b. Negative	0 7 8 11 12 15 | O.C. | RA | I |	-(I+1)	-----	---	-----	
"ISM"						
6. IC Relative	0 7 8 15 | O.C. | 0 |	---	-----	D+(IC)	-----	-128≤D≤127
"ICR"						
7. Base Relative	0 5 6 7 8 15 | O.C. | BR | DU |	[DU+(BR)]	[DU+(BR),DU+1+(BR)]	DU+(BR)	DU+(BR),DU+1+(BR)	Base registers. GR=R12, R13, R14, and R15.
a. Not Indexible						
"B"	RR'=BR-12					0≤DU≤255
	0 5 6 7 8 11 12 15 | O.C. |BR'|OCX | RX |					
b. Indexible	RX=0 (Non-Indexed) RX≠0 (Indexed)	[[BR]] [[BR]+(RX)]	[[BR],[BR]+1] [[BR]+(RX),[BR]+1+(RX)]	[BR] [BR]+(RX)	[BR],[BR]+1 [BR]+(RX),[BR]+1+(RX)	
"BI"						
"BIX"						

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. 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:

a. Only registers R1, R2, ..., R15 may be used as index registers (RX).

b. Only four registers, R12, R13, R14, and R15 may be used as base registers for instructions having the Base Relative address mode.

c. R15 is the implicit stack pointer for the Push and Pop Multiple instructions (Opcode $8F_{16}$ and $9F_{16}$).

d. The general registers are not in the logical memory address space.

e. 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.

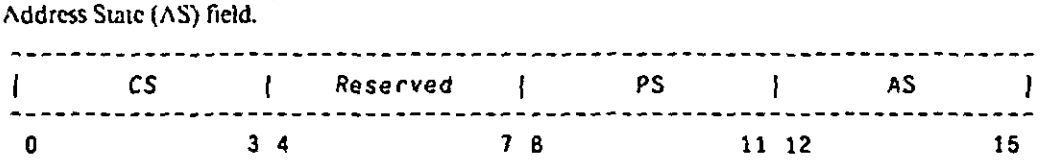
The general registers shall functionally appear to be 16 bits in length. For instructions requiring a 32-bit operation, adjacent registers shall be concatenated to form effective 32-bit registers. Instructions requiring 48-bit operation shall concatenate three adjacent registers to form an effective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

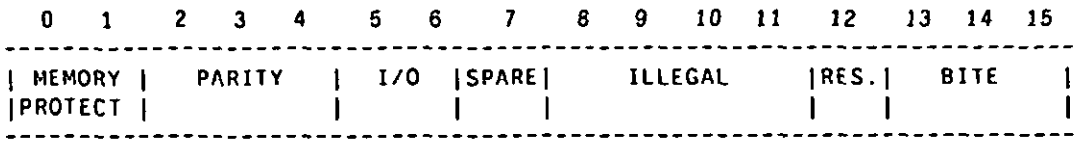
For implementations which include the memory access lock feature of the expanded memory addressing option (see paragraph 4.5.2.2), PS shall define the memory access key code for all instructions and operand references to memory. References to memory during the interrupt recognition sequence for vector 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 paragraph 4.5.2.3), 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.

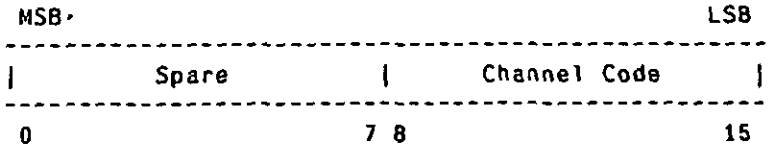
Bit 13: Built-in Test Fault. Hardware built-in test equipment (BITE) error has been detected.

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 paragraph 4.6.

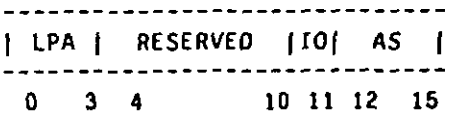
4.4.2.5 Pending interrupt register (PI). The pending interrupt request register is software and hardware controlled and contains the pending interrupts that are attempting to vector the instruction counter. A pending interrupt is set by a system interrupt signal. The pending interrupt bit that generates 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 256 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.



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 page 87), and Pop Multiple registers, POPM (see page 77). The stack expands linearly toward zero as items are added to it.

Two instructions, Stack IC and Jump to Subroutine, SJS (see page 68), and Unstack IC and Return from Subroutine, URS (see page 69), 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 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 [1]
Page Registers	Group 0 enabled [1]
Page Registers AL Field	All zeros [1]
Page Registers W Field	Zero [1]
Page Registers E Field	Zero [1]
Page Registers PPA field	Exact logical to physical [1]
Memory Protect RAM	Disabled and all zeros [1 2]
Start Up ROM	Enabled [1]
DMA Enable	Disabled [1]
Input Discretes	Indeterminate [1]
Trigger Go Indicator	Started [1]
Discrete Outputs	All zeros [1]

[1] If implemented (optional)
[2] 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 page 29). 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 $FFFF_{16}$ 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 paragraph 4.4.4.1. The timers are halted when a breakpoint, BPT (see page 138), 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 paragraph 4.5.2) is not implemented, physical addresses shall equal logical addresses.

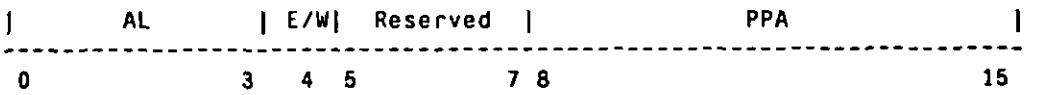
4.5.1.1 Memory addressing arithmetic. *Arithmetic performed on memory logical addresses shall be modulo* 65,536 such that references to the maximum logical address of $FFFF_{16}$ plus 1 shall be to logical address 0000_{16}.

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. 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.

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 pointer references to memory.

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. The access key codes may be supplied by either the status word or the DMA channel. For each of the possible 16 values of the AL code, access shall be permitted *for the reference according to table VII.*
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

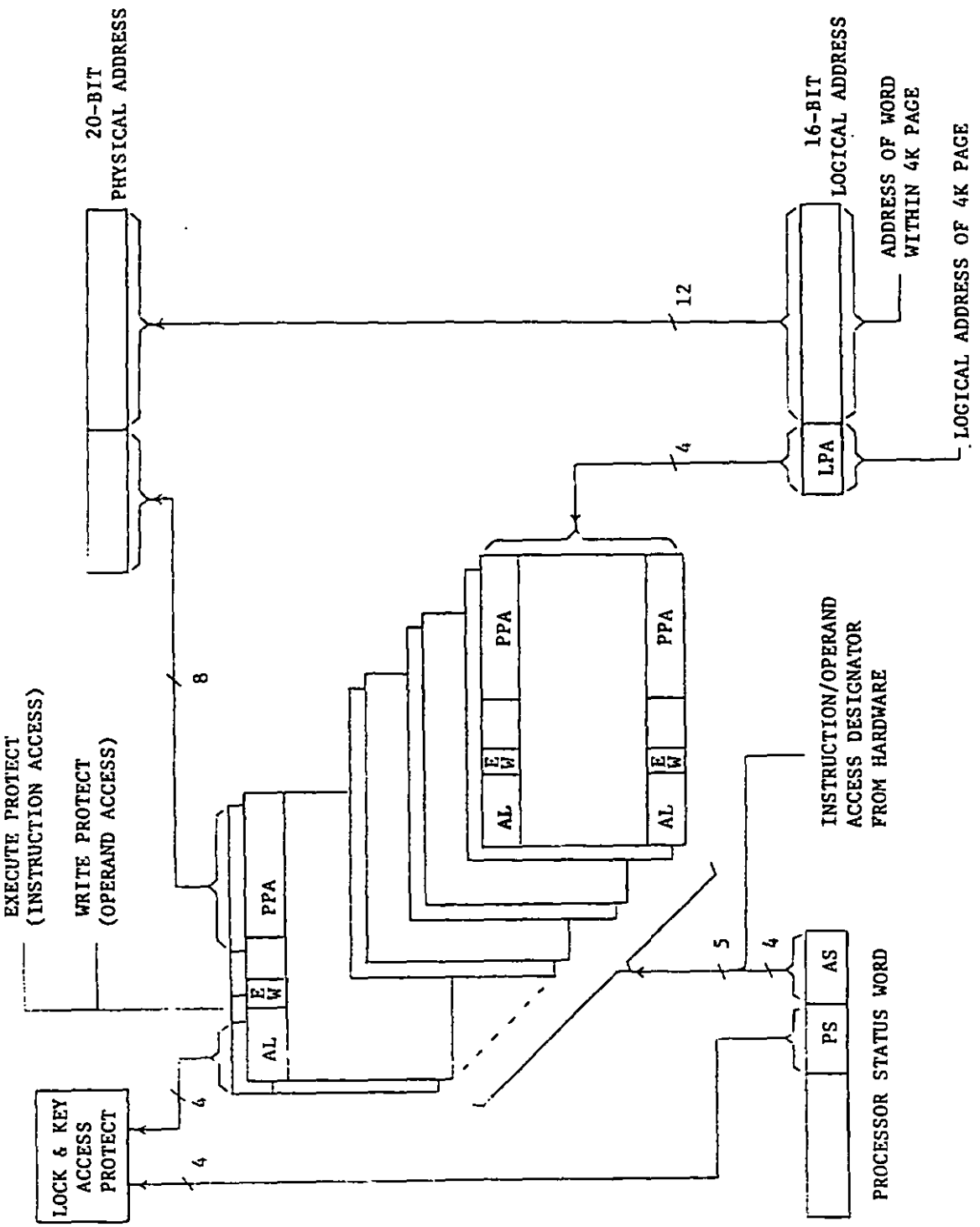


FIGURE 1. Expanded memory mapping diagram

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
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

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:

a. An access lock code of F_{16} is an "unlocked" lock code and allows any and all access key codes to be acceptable.

b. An access key code of 0 is a "master" key code and is acceptable to any and all access lock codes.

c. Access key codes 1 through E_{16} are acceptable to only their own "matched" lock code or the "unlocked" lock code of F_{16}.

d. 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 of 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.

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 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.

4.6 Interrupt control.

4.6.1 Interrupts. The instruction set shall support a minimum of sixteen (16) interrupts as shown in table VIII. 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.

4.6.1.1 Interrupt acceptance. The interrupt system shall have the capability to accept external and internal interrupts. Figure 2 indicates the relationship between the interrupt signals, the pending interrupt register, the interrupt mask register, the priority control logic, the software controllable/accessible 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 interrupts 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

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
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

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

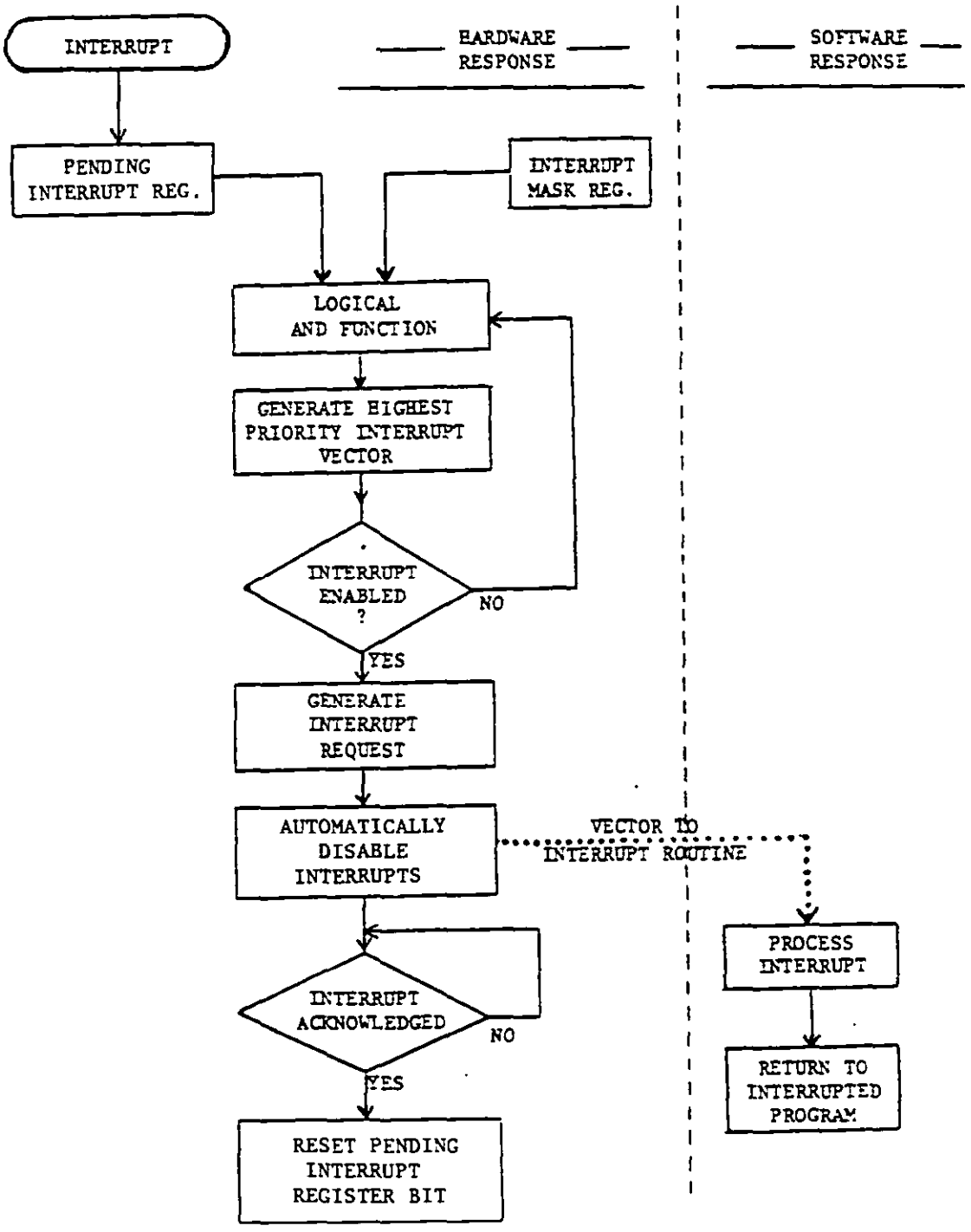


FIGURE 2. Interrupt system flowchart

are given in the input/output instruction repertoire.

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. 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 page 62) 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. 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.

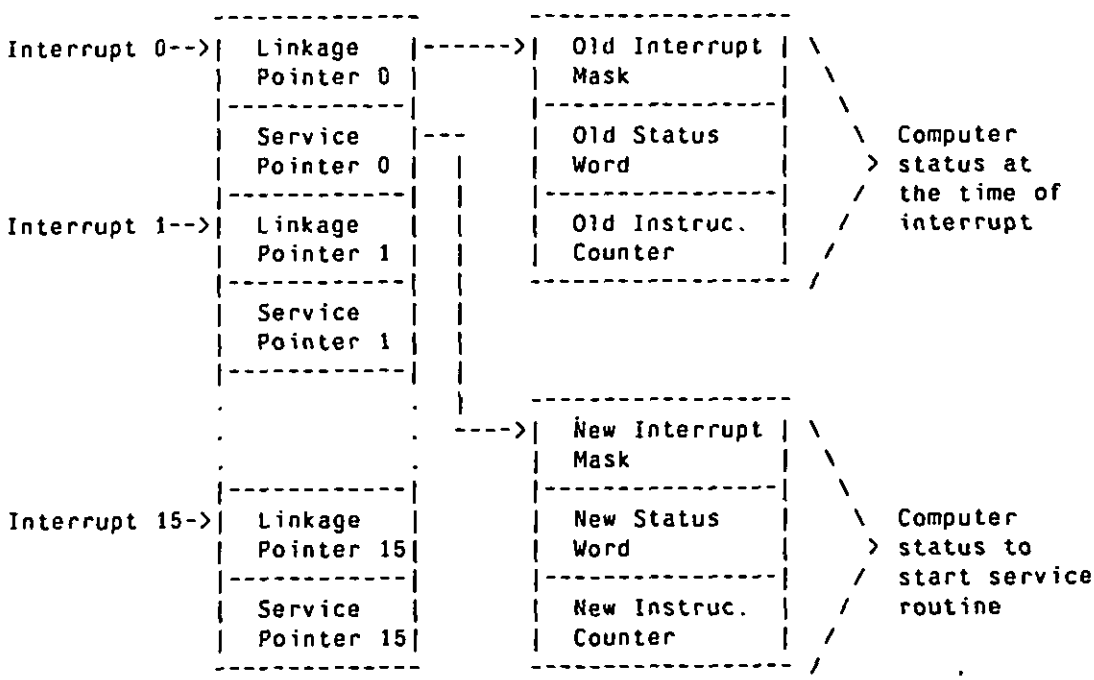


FIGURE 3. Interrupt vectoring system

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 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. 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 load, A for add, ST for store.

Floating point operations have a prefix of F, e.g., FL for floating load, FA for floating add.

Double precision operations have a prefix of D, e.g., DL for double load, DA for double add.

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 load indirect.

Immediate addressing, using the address field as an operand, is indicated by a suffix of IM, e.g., AIM for single

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 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. 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 load, A for add, ST for store.

Floating point operations have a prefix of F, e.g., FL for floating load, FA for floating add.

Double precision operations have a prefix of D, e.g., DL for double load, DA for double add.

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 load indirect.

Immediate addressing, using the address field as an operand, is indicated by a suffix of IM, e.g., AIM for single

TABLE IX. Input/output channel groups

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

precision add immediate.
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.

4.8.3 Instruction matrix. Table X 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 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.)

FO	Floating point 8-bit 2's complement Derived Operand characteristic (exponent): DO_{24-31} MA
Floating point register accumulator mantissa (fractional part): $(RA,RA+1)_{0-23}$ (F.L.P.), $(RA,RA+1)_{0-23}$ $(RA+2)_{32-47}$ (E.F.P.)	
FA	Floating point 8-bit 2's complement register accumulator characteristic (exponent): $(RA,RA+1)_{24-31}$
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
⊕	Exclusive OR
¬	Logical NOT
||	Absolute value

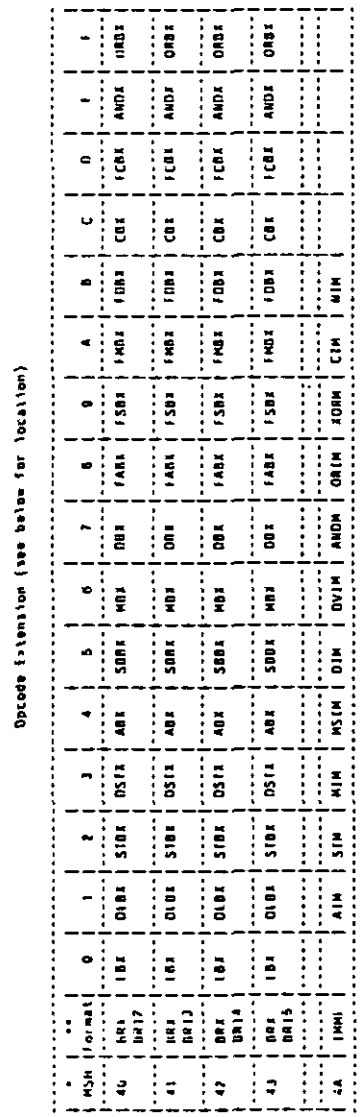
TABLE X. Operation code matrix

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	LOAD STORE	INTEGER ARITHMETIC	FLOATING POINT*	LOGICAL COMPARE	OPCODE EXTENSIONS	BIT	SHIFT	JUMP	LOAD	STORE	ADD	SUB	MULT	DIVIDE	LOGICAL	COMPARE
0	LB R0,R12	AB R0,R12	FAB R0,R12	OPB R0,R12	0Rx R0,R12*	SB	SLL	JC	L	ST	A	S	MS	DV	OR	C
1	LB R0,R13	AB R0,R13	FAB R0,R13	ORB R0,R13	0Rx R0,R13*	SBR ?	SRL	JCT	LR	STC	AR	SR	MSR	DVR	ORR	CR
2	LB R0,R14	AB R0,R14	FAB R0,R14	ORB R0,R14	0Rx R0,R14*	SBI	SRA	JS	LISP	STCI	AISP	SISP	MISP	DISP	AND	CISP
3	LB R0,R15	AB R0,R15	FAB R0,R15	ORB R0,R15	0Rx R0,R15*	RB	SLC	SOJ	LISW	MOV	INCM	DECM	MISW	DISW	ANDM	CISW
4	LBR R0,R12	SUB R0,R12	FSB R0,R12	ANDB R0,R12		RBR		BR	LI	STI	ABS	NEG	M	D	XOR	CBS
5	LBR R0,R13	SUB R0,R13	FSB R0,R13	ANDB R0,R13		RBI	DSLL	BLT	LLM		DABS	DNEG	MR	DR	XORM	
6	LBR R0,R14	SBR R0,R14	FSB R0,R14	ANDB R0,R14		IB	DSAL	BLE	DI	DST	DA	DS	DM	DD	M	DC
7	LBR R0,R15	SBR R0,R15	FSB R0,R15	ANDR R0,R15		IBM	DSRA	BNZ	DLM	SBM	DAR	DSR	DMR	DDM	MR	DCM
8	LHI R0,R12	MB R0,R12	FMB R0,R12	CB R0,R12	EIO* #	IBI	DSLC	BLT	DLI	DSTI	AA	AS	AM	ID	XM	EC
9	LHI R0,R13	MB R0,R13	FMB R0,R13	CB R0,R13	VIO* #	ISB		BGT	LM	STM	FAR	FSR	FMR	FDR	FIR	FCR
A	LHI R0,R14	MB R0,R14	FMB R0,R14	CB R0,R14	IMMI *	SVBM	SLW	BNZ	FFI	LCSI	FIA	FIS	FIM	FID	FIXR	FFC
B	LHI R0,R15	MB R0,R15	FMB R0,R15	CB R0,R15			SAR	BGT	LUB	STUB	FIAR	FISM	FIMR	FIDR	FIFR	FICR
C	LHLI R0,R12	LR R0,R12	FDB R0,R12	FCB R0,R12		RVBR	SCR	LSIIM	LIB	STIB	FABS	FNEG			XBR	
D	LHLI R0,R13	AR R0,R13	FMB R0,R13	FCB R0,R13			DSLB	LSIB	SUBL	SUBI					AWR	
E	DSIB R0,R14	SH R0,R14	FDB R0,R14	FCB R0,R14		IVBR	DSAR*	SJS	LIBI	SIBI						
F	DSTR R0,R15	IO R0,R15	FDB R0,R15	FCB R0,R15			DSCR	URS	POPM	PSHM						INDP/RPT

* These order types represent instructions which have "extended" operation codes and are fully described in the instruction specifications and in Table 4-5.

\# Privileged instructions

TABLE XI. Extended operation codes



5 DETAILED REQUIREMENTS

5.1 Execute input/output.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
			8 4 4 16
IM	XIO	RA,CMD	48 RA RX CMD
IMX	XIO	RA,CMD,RX	

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.

Mandatory XIO Command Fields and Mnemonics

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 outputs the 16-bit contents of RA to the pending interrupt register. If there is a one in the corresponding bit position of the interrupt 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 paragraph 5.30).

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 registerus 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.

Optional XIO Command Fields and Mnemonics

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" 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 page 137).

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 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 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 memory. 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.

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 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 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 memory. The RAM words of 0 through 63 apply to processor write protect and words 64 through 127 apply to DMA write protect.

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 transfer the 16-bit contents of page register Y of the operand set of group X to register RA.

**** **** — User defined XIO functions (see table IX).

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	49 RA RX ADDR
DX	VIO RA,ADDR,RX	

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 paragraph 5.1) 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.

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.  if [DA+1]n=1, then I/O data = [DA+2+i];

Step 4.  if [DA+1]n=1, then i <-- i+1;

Step 5.  n <-- n + 1, exit, if n = 16;

Step 6.  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	55 | N | RX | ADDR
IX	RBI	N,ADDR,RX	

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	56 | N | RX | ADDR
DX	TB	N,ADDR,RX	
			8 4 4 16
I	TBI	N,ADDR	58 | N | RX | ADDR
IX	TBI	N,ADDR,RX	

DESCRIPTION: Bit number N ($0 \leq N \leq 15$) 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 $0 \leq N \leq 15$;
(CS) <-- 0001 if DO_N = 1 and N = 0;
(CS) <-- 0100 if DO_N = 1 and $1 \leq N \leq 15$;

REGISTERS AFFECTED: CS

5.6 Test and set bit.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
D	TSB N,ADDR	8 4 4 16
DX	TSB N,ADDR,RX	59 | N | RX | ADDR

DESCRIPTION: Bit number N ($0 \leq N \leq 15$) 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 $0 \leq N \leq 15$;
(CS) <-- 0001 if (DO_N) = 1 and N = 0;
(CS) <-- 0100 if (DO_N) = 1 and $1 \leq N \leq 15$;

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 ($0 \leq N \leq 15$) of the register RB is set to one where the least significant four bits of the contents of register RA is N. Bits $(RA)_{0-11}$ 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 = $(RA)_{12-15}$;

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 ($0 \leq N \leq 15$) of register RB is set to zero where the least significant four bits of the contents of register RA is N. Bits $(RA)_{0-11}$ 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 where N = $(RA)_{12-15}$;

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 ($0 \leq N \leq 15$) 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:

$N = (RA)_{12-15}$

(CS) <-- 0010 if $(RB_N) = 0$ and $0 \leq N \leq 15$;
(CS) <-- 0001 if $(RB_N) = 1$ and $N = 0$;
(CS) <-- 0100 if $(RB_N) = 1$ and $1 \leq N \leq 15$;

REGISTERS AFFECTED: CS

5.10 Shift left logical.

ADDR MODE	MNEMONIC	FORMAT/OPCODE	
		8 4 4	
R	SLL RB,N	| 60 |N-1 | RB |	$1 \leq N \leq 16$

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.

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

```
                                    0                    15
                                    ------------------------
EXAMPLE:  RB Before Shift           | sabc| defg| hijk| lmnp|
                                    ------------------------

          RB After Shift (N=4)      ------------------------
                                    | defg| hijk| lmnp| 0000|
                                    ------------------------
```

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.

ADDR MODE	MNEMONIC	FORMAT/OPCODE	
		8 4 4	
R	SRL RB,N	| 61 |N-1 | RB |	1 ≤ N ≤ 16

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.

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

```
                                  0                    15
                                  -------------------------
EXAMPLE:  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.

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

EXAMPLE: RB Before Shift

0			15
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 conditions status, CS, is set based on the result in RB.

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

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

```
                                    0                        15
                                   --------------------------
        EXAMPLE:  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 position 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.

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

EXAMPLE: RB, RB+1 Before Shift

0 RB 15	0 RB+1 15
$\|s_1$abc| defg| hijk| lmnp|	$\|s_2$qrs| tuvw| xyzz| zzzz|

RB, RB+1 After Shift (N=4)

0 RB 15	0 RB+1 15
| defg| hijk| lmnp|$s_2$qrs|	| tuvw| xyzz| zzzz| 0000|

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.

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

EXAMPLE: RB, RB+1 Before Shift

0 RB 15	0 RB+1 15
$\|s_1$abc| defg| hijk| lmnp|	$\|s_2$qrs| tuvw| xyzz| zzzz|

RB, RB+1 After Shift (N=4)

0 RB 15	0 RB+1 15
0000|$s_1$abc| defg| hijk|	lmnp|$s_2$qrs| tuvw| xyzz|

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.

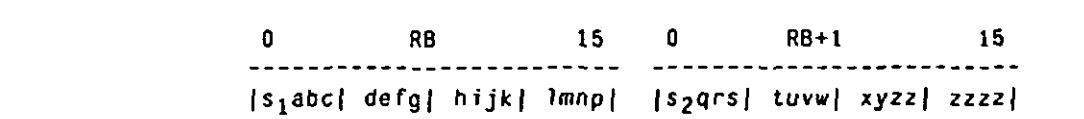
ADDR MODE	MNEMONIC	FORMAT/OPCODE	
		8 4 4	
R	DSRA RB,N	| 67 |N-1 | RB |	1 ≤ N ≤ 16

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.

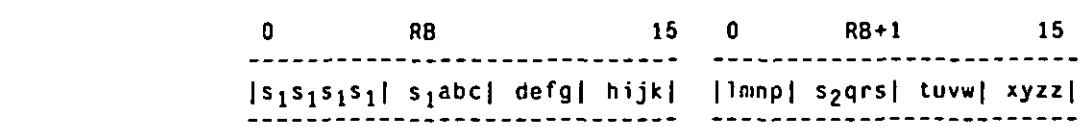
Note: N-1 = 0 represents a shift of one position.

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

EXAMPLE: RB, RB+1 Before Shift



RB, RB+1 After Shift (N=4)



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.

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

EXAMPLE: RB, RB+1 Before Shift

0 RB 15	0 RB+1 15
|$s_1$abc| defg| hijk| lmnp|	|$s_2$qrs| tuvw| xyzz| zzzz|

RB, RB+1 After Shift (N=4)

0 RB 15	0 RB+1 15
|defg| hijk| lmnp| $s_2$qrs|	|tuvw| xyzz| zzzz| $s_1$abc|

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	$\|(RB)\| \leq 16$

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.

If |N| > 16, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP (see page 137).

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

(See "Description" of the logical shift instructions, SLL and SRL, (see pages 41 and 42), 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	$\|(RB)\| \leq 16$

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.

If |N| > 16, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP (see page 137).

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

(See "Description" of the arithmetic shift instruction SRA (see page 43) for definition of the right shift operation. Left shift causes "zeros" to be shifted into low order position of result.)

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.

If |N| > 16, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP (see page 137).

(See "Description" of the cyclic shift instruction, SLC (see page 44), for definition of shift operations.)

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 registers 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.

If |N| > 32, the fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP (see page 137).

(See "Description" of the double shift logical instructions, DSRL and DSLI, (see pages 46 and 45), for definition of shift operations.)

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.

If |N| > 32, the fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP (see page 137).

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

(See "Description" of the double shift arithmetic instruction, DSRA (see page 47), for the definition of the right shift operation. Left shift causes "zeros" to be shifted into low order position of result.)

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.

If |N| > 32, the fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP (see page 137).

(See "Description" of the double shift cyclic instruction, DSLC (see page 48), for the definition of shift operations.)

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
0	JC	C,LABEL	70 C RX LABEL
DX	JC	C,LABEL,RX	
			8 4 4 16
I	JCI	C,ADDR	71 C RX ADDR
IX	JCI	C,ADDR,RX	

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 Code

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 $(C_0 \uparrow CS_0) \vee (C_1 \uparrow CS_1) \vee (C_2 \uparrow CS_2) \vee (C_3 \uparrow CS_3) = 1$;

REGISTERS AFFECTED: IC (if jump is executed)

5.25 Jump to subroutine.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
D	JS	RA,LABEL	8 / 4 / 4 / 16 bits: 72 | RA | RX | LABEL
DX	JS	RA,LABEL,RX	

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	73 RA RX LABEL
DX	SOJ	RA,LABEL,RX	

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_0 being set to 1.

REGISTER TRANSFER DESCRIPTION:

(RQ,RQ+1,RQ+2) <-- (MK,SW,IC);

(SVP) <-- [$2B_{16}$], where AS = 0;

PI_5 <-- 1;

(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 is 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	7D 0000 RX ADDR
DX	LST	ADDR,RX	
			8 4 4 16
I	LSTI	ADDR	7C 0000 RX ADDR
IX	LSTI	ADDR,RX	

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	7E | RA | RX | LABEL
DX	SJS	RA,LABEL,RX	

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

5.3S Single precision load.

ADDR MODE	MNEMONIC		FORMAT/OPCODE	
R	LR	RA,RB	8 4 4 | 81 | RA | RB |	
B	LB	BR,DSPL	4 2 2 8 | 0 |0 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
BX	LBX	BR,RX	4 2 2 4 4 | 4 |0 |BR'| 0 | RX |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
ISP	LISP	RA,N	8 4 4 | 82 | RA |N-1 |	1 ≤ N ≤ 16
ISN	LISN	RA,N	8 4 4 | 83 | RA |N-1 |	1 ≤ N ≤ 16
D DX	L L	RA,ADDR RA,ADDR,RX	8 4 4 16 | 80 | RA | RX | | ADDR |	
IM IMX	LIM LIM	RA,DATA RA,DATA,RX	8 4 4 16 | 85 | RA | RX | | DATA |	
I IX	LI LI	RA,ADDR RA,ADDR,RX	8 4 4 16 | 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
B	DLB	BR,DSPL	| 0 |1 |BR'| DSPL | 12 ≤ BR ≤ 15; BR' = BR - 12; RA = R0
			4 2 2 4 4
BX	DLBX	BR,RX	| 4 |0 |BR'| 1 | RX | 12 ≤ BR ≤ 15; 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
D	LM N,ADDR	8 | 4 | 4 | 16
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	8A | RA | RX | ADDR
DX	EFL RA,ADDR,RX	

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	8B RA RX ADDR
DX	LUB	RA,ADDR,RX	
			8 4 4 16
I	LUBI	RA,ADDR	8D RA RX ADDR
IX	LUBI	RA,ADDR,RX	

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:

$(RA)_{8-15}$ <-- DO_{0-7};

(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	8C RA RX ADDR
DX	LLB	RA,ADDR,RX	
			8 4 4 16
I	LLBI	RA,ADDR	8E RA RX ADDR
IX	LLBI	RA,ADDR,RX	

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,

a. as each word is popped from the stack, R15 is incremented by one;

b. if R15 is included in the transfer, then it is effectively ignored;

c. 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
B	STB	BR,DSPL	| 0 |2 |BR'| DSPL |
			4 2 2 4 4
BX	STBX	BR,RX	| 4 |0 |BR'| 2 | RX |
			8 4 4 16
D	ST	RA,ADDR	| 90 | RA | RX | | ADDR |
DX	ST	RA,ADDR,RX	
			8 4 4 16
I	STI	RA,ADDR	| 94 | RA | RX | | ADDR |
IX	STI	RA,ADDR,RX	

$12 \leq BR \leq 15$
$BR' = BR - 12$
$RA = R2$

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
D	STC	N,ADDR	8 / 4 / 4 / 16
DX	STC	N,ADDR,RX	| 91 | N | RX | | ADDR |
I	STCI	N,ADDR	8 / 4 / 4 / 16
IX	STCI	N,ADDR,RX	| 92 | N | RX | | ADDR |

DESCRIPTION: The constant N, where N is an integer ($0 \leq N \leq 15$) 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 $0 \leq N \leq 15$;

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.

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

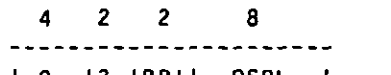
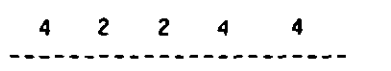
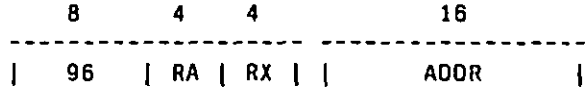
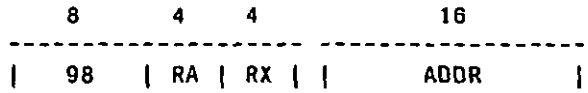
RB has a final value equal to the address of the last word transfered plus one.

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
B	DSTB BR,DSPL	 12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
BX	DSTX BR,RX	 12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
D DX	DST RA,ADDR DST RA,ADDR,RX	
I IX	DSTI RA,ADDR DSTI RA,ADDR,RX	

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:

$$[DA] \leftarrow\!- \{[DA] \uparrow (\overline{RA+1})) \vee \{[RA] \uparrow [RA+1]\};$$

$$(RA+1) = MASK,\ (RA) = DATA;$$

or, equivalently,

$$(RQ) \leftarrow\!- [DA];$$

$$(RQ)_i \leftarrow\!- (RA)_i \text{ if } (RA+1)_i = 1 \text{ for } i = 0, 1, \ldots, 15;$$

$$[DA] \leftarrow\!- (RQ);$$

REGISTERS AFFECTED: None

5.50 Store multiple registers.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
D	STM N,ADDR	8 4 4 16
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, $0 \leq N \leq 15$. 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	9A RA RX ADDR
DX	EFST RA,ADDR,RX	

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	9B RA RX ADDR
DX	STUB	RA,ADDR,RX	
			8 4 16
I	SUBI	RA,ADDR	9D RA RX ADDR
IX	SUBI	RA,ADDR,RX	

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:

$[DA]_{0-7} \leftarrow (RA)_{8-15}$:

REGISTERS AFFECTED: None

5.53 Store into lower byte.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
		8 4 4 16
D	STLB RA,ADDR	9C | RA | RX | ADDR
DX	STLB RA,ADDR,RX	
		8 4 4 16
I	SLBI RA,ADDR	9E | RA | RX | ADDR
IX	SLBI RA,ADDR,RX	

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:

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

REGISTERS AFFECTED: None

5.5.4 Push multiple registers onto the stack.

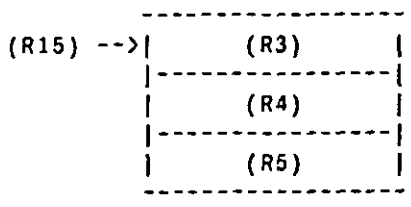
ADDR MODE	MNEMONIC	FORMAT/OPCODE
		8 4 4
S	PSHM RA,RB	9F | RA | RB

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.

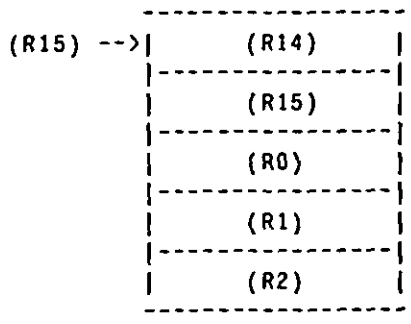
For RA > RB, the contents of RB through R0, and then the contents of R15 through RA, are pushed onto the stack. On completion, R15 points to the last item on the stack, the contents of RA.

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.

EXAMPLE: PSHM R3,R5 results in



PSHM R14,R2 results in



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	
B	AB	BR,DSPL	| 1 |0 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	ABX	BR,RX	| 4 |0 |BR'| 4 | RX |	12 ≤ BR ≤ 15 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 $(RA_0)^1 \neq (RA_0)^2$

(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
0	INCM N,ADDR	A3 | N-1 | RX | ADDR
DX	INCM N,ADDR,RX	

DESCRIPTION: The contents of the memory location specified by the Derived Address, DA, is incremented by N, where N is an integer, $1 \leq N \leq 16$. 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 $1 \leq N \leq 16$;

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.

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:

PI_4 <-- 1, exit, if DO = 8000_{16};

(RA) <-- |DO|;

(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.

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:

PI_4 <-- 1, exit, if DO = $8000\ 0000_{16}$;

(RA,RA+1) <-- |DO|;

(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
R	DAR	RA,RB	8 | 4 | 4 A7 | RA | RB
D	DA	RA,ADDR	8 | 4 | 4 | 16 A6 | RA | RX | ADDR
DX	DA	RA,ADDR,RX	

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:

$(RA,RA+1)^2 \leftarrow (RA,RA+1)^1 + DO;$

$PI_4 \leftarrow 1$, if $(RA_0)^1 = DO_0$ and $(RA_0)^1 \neq (RA_0)^2$

```
(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	
B	FAB	BR,DSPL	| 2 |0 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
			4 2 2 4 4	
BX	FABX	BR,RX	| 4 |0 |BR'| 8 | RX |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
			8 4 4 16	
D	FA	RA,ADDR	| A8 | RA | RX | | ADDR |	
DX	FA	RA,ADDR,RX		

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;

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 <-- $\overline{MA}_0$, EA <-- EA+1,
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 - DO;

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 <--M̄n0, EA <-- EA+1,
       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.

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

REGISTER TRANSFER DESCRIPTION:

EA <-- EA+1, MA <-- $4000\ 00_{16}$, if MO = $8000\ 00_{16}$;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- $7FFF\ FF_{16}$, exit, if EA > $7F_{16}$;

EA <-- EO, MA <-- -MO, if MO < 0, MO ≠ $8000\ 00_{16}$;

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	
B	SBB	BR,DSPL	| 1 |1 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	SBBX	BR,RX	| 4 |0 |BR'| 5 | RX |	12 ≤ BR ≤ 15 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 | | DATA |	

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) + $\overline{DO}$} + 1;

PI_4 <-- 1, if $(RA_0)^1 \neq DO_0$ and $(RA_0)^2 = OO_0$

(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
D	DECM N,ADDR	8 4 4 16
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, $1 \leq N \leq 16$. This is the 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 $1 \leq N \leq 16$;

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 Operand, 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.

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:

PI_4 <-- 1, exit, if DO = 8000_{16};

(RA) <-- -DO;

(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.

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:

PI_4 <-- 1, exit, if DO = $8000\ 0000_{16}$;

(RA,RA+1) <-- -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

5.67 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	| B6 | RA | RX | | ADDR |
DX	DS	RA,ADDR,RX	

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 $((RA,RA+1) + \overline{DO}) + 1$;

PI_4 <-- 1, if $(RA_0)^1 \neq DO_0$ 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

5.6.8 Floating point subtract.

ADDR MODE	MNEMONIC		FORMAT/OPCODE	
			8 4 4	
R	FSR	RA,RB	| B9 | RA | RB |	
			4 2 2 8	
B	FSB	BR,DSPL	| 2 |1 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
			4 2 2 4 4	
BX	FSBX	BR,RX	| 4 |0 |BR'| 9 | RX |	12 ≤ BR ≤ 15 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 - E0;

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 <-- $\overline{MA}_0$, EA <-- EA+1,
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

5.69 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	BA | RA | RX | ADDR
DX	EFS RA,ADDR,RX	

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 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 - E0;

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+1,
       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 0000 16, 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 is 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_6 <-- 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 |	$1 \leq N \leq 16$
			8 4 4	
ISN	MISN	RA,N	| C3 | RA |N-1 |	$1 \leq N \leq 16$
			8 4 4 16	
D	MS	RA,ADDR	| C0 | RA | RX | | ADDR |	
DX	MS	RA,ADDR,RX		
			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:

$(RQ, RQ+1) \leftarrow (RA)^1 \times DO;$

$(RA)^2 \leftarrow (RQ+1);$

$PI_4 \leftarrow 1,$ if $\{\{(RA_0)^1 = DO_0$ and $\{(RQ) \neq 0$ or $(RQ+1_0) = 1\}\}$ or
$\{(RA_0)^1 \neq DO_0$ and $\{(RQ) \neq FFFF_{16}$ or $(RQ+1_0) = 0\}$ and
$\{(RA)^1 \neq 0$ and $DO \neq 0\}\};$

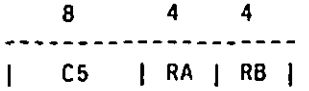
(CS) <-- 0010 if (RA) = 0;
(CS) <-- 0001 if (RA) < 0;
(CS) <-- 0100 if (RA) > 0;

REGISTERS AFFECTED: RA, CS, PI

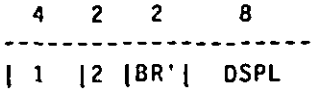
5.7.2 Single precision integer multiply with 32-bit product.

ADDR MODE MNEMONIC FORMAT/OPCODE

R MR RA,RB

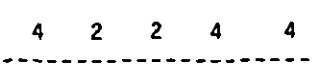


B MB BR,DSPL



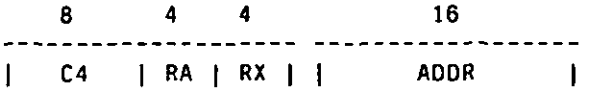
12 ≤ BR ≤ 15
BR' = BR - 12
RA = R2

BX MBX BR,RX

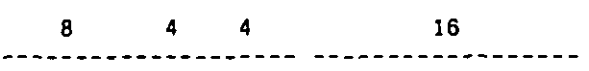


12 ≤ BR ≤ 15
BR' = BR - 12
RA = R2

D M RA,ADDR
DX M RA,ADDR,RX



IM MIM RA,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	C6 | RA | RX | ADDR
DX	DM	RA,ADDR,RX	

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:

$(RQ,RQ+1,RQ+2,RQ+3) \leftarrow (RA,RA+1)^1 \times DO;$

$(RA,RA+1)^2 \leftarrow (RQ+2,RQ+3);$

$PI_4 \leftarrow 1,\ if\ \{\{(RA_0)^1 = DO_0\ and\ \{(RQ,RQ+1) \neq 0\ or\ (RQ+2_0) = 1\}\}\ or$
$\{(RA_0)^1 \neq DO_0\ and\ \{(RQ,RQ+1) \neq FFFF\ FFFF_{16}\ or\ (RQ+2_0) = 0\}\ and$
$\{(RA)^1 \neq 0\ and\ DO \neq 0\}\};$

$(CS) \leftarrow 0010\ \ if\ \ (RA,RA+1) = 0;$
$(CS) \leftarrow 0001\ \ if\ \ (RA,RA+1) < 0;$
$(CS) \leftarrow 0100\ \ if\ \ (RA,RA+1) > 0;$

REGISTERS AFFECTED: RA, RA+1, CS, PI

5.7.4 Floating point multiply.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
			8 4 4
R	FMR	RA,RB	| C9 | RA | RB |
			4 2 2 8
B	FMB	BR,DSPL	| 2 |2 |BR'| DSPL | 12 ≤ BR ≤ 15; BR' = BR - 12; RA = R0
			4 2 2 4 4
BX	FMBX	BR,RX	| 4 |0 |BR'| A | RX | 12 ≤ BR ≤ 15; BR' = BR - 12; RA = R0
			8 4 4 16
D	FM	RA,ADDR	| C8 | RA | RX | | ADDR |
DX	FM	RA,ADDR,RX	

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, it 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 + E0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF_{16}, exit, if n > $7F_{16}$ and MA_0 = MO_0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00_{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_{0-23} <-- 4000 00_{16}, if MP_{0-23} = 8000 00_{16};

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF_{16}, exit, if n > $7F_{16}$ and MP_0 = 0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00_{16}, exit, if n > $7F_{16}$ and MP_0 = 1;

n, MP <-- normalized n, MP;

PI_6 <-- 1, EA <-- 0, MA <-- 0, exit, if n < 80_{16};

EA <-- n;

MA <-- MP_{0-23};

(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 + E0;

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_{0-39} <-- 4000 00 0000_{16}, if MP_{0-39} = 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_{16}, 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_{0-39};

(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,R8	| D1 | RA | RB |	
			8 4 4	
ISP	DISP	RA,N	| D2 | RA |N-1 |	$1 \leq N \leq 16$
			8 4 4	
ISN	DISN	RA,N	| D3 | RA |N-1 |	$1 \leq N \leq 16$
			8 4 4 16	
D	DV	RA,ADDR		
DX	DV	RA,ADDR,RX	| D0 | 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;

PI4 <-- 1, if DO = 0 or (RA = 8000₁₆ and DO = FFFF₁₆);

(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	
B	DB	BR,DSPL	| 1 |3 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	DBX	BR,RX	| 4 |0 |BR'| 7 | RX |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			8 4 4 16	
D	D	RA,ADDR	| D4 | RA | RX | | ADDR |	
DX	D	RA,ADDR,RX		
			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 the magnitude of the MSH of the dividend is equal to or greater than the magnitude of the divisor (i.e., the quotient exceeds 15 bits).

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

REGISTER TRANSFER DESCRIPTION:

```
(RA,RA+1) <-- (RA,RA+1) / DO;

PI4 <-- 1, if DO = 0 or |(RA)| ≥ |DO|;


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

REGISTERS AFFECTED: RA, RA+1, CS, PI

5.7.8 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	| 06 | RA | RX | | ADDR |
DX	DD	RA,ADDR,RX	

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_{16} and the divisor is $FFFF_{16}$.

REGISTER TRANSFER DESCRIPTION:

(RA,RA+1) <-- (RA,RA+1) / DO;

PI_4 <-- 1, if DO = 0 or {RA = 8000_{16} and DO = $FFFF_{16}$};

(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

5.79 Floating point divide.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
			8 4 4
R	FDR	RA,RB	| D9 | RA | RB |
			4 2 2 8
B	FDB	BR,DSPL	| 2 |3 |BR'| DSPL | 12 ≤ BR ≤ 15; BR' = BR - 12; RA = R0
			4 2 2 4 4
BX	FDBX	BR,RX	| 4 |0 |BR'| B | RX | 12 ≤ BR ≤ 15; BR' = BR - 12; RA = R0
			8 4 4 16
D	FD	RA,ADDR	| D8 | RA | RX | | ADDR |
DX	FD	RA,ADDR,RX	

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;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF_{16}, exit,
if MA_0 = MO_0 and {n > $7F_{16}$ or D0 = 0};

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00_{16}, exit,
if MA_0 ≠ MO_0 and {n > $7F_{16}$ or D0 = 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_{16}, exit, if n > $7F_{16}$ and MQ_0 = 0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00_{16}, exit, if n > $7F_{16}$ and MQ_0 = 1;

EA <-- n;

MA <-- MQ_{0-23};

(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 - EO;

PI3 <-- 1, EA <-- 7F16, MA <-- 7FFF FF FFFF16, exit,
        if MA0 = MO0 and (n > 7F16 or DO = 0);

PI3 <-- 1, EA <-- 7F16, MA <-- 8000 00 000016, 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 FF FFFF16, 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	
B	ORB	BR,DSPL	| 3 |0 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	ORBX	BR,RX	| 4 |0 |BR'| F | RX |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			8 4 4 16	
D	OR	RA,ADDR	| E0 | RA | RX | | ADDR |	
DX	OR	RA,ADDR,RX		
			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

5.8.2 Logical AND.

ADDR MODE	MNEMONIC		FORMAT/OPCODE	
			8 4 4	
R	ANDR	RA,RB	| E3 | RA | RB |	
			4 2 2 8	
B	ANDB	BR,DSPL	| 3 |1 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	ANDX	BR,RX	| 4 |0 |BR'| E | RX |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			8 4 4 16	
D	AND	RA,ADDR	| E2 | RA | RX | | ADDR |	
DX	AND	RA,ADDR,RX		
			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	E4 | RA | RX | ADDR
DX	XOR	RA,ADDR,RX	
			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) ⊕ 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	E6 | RA | RX | ADDR
DX	N	RA,ADDR,RX	
			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:

$(RA) \leftarrow \overline{(RA)} \uparrow DO;$

```
(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:

PI_4 <-- 1, exit, if EO > $0F_{16}$;

(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.

An integer zero, 0000_{16}, is converted to a floating point zero, $0000\ 0000_{16}$.

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:

PI_4 <-- 1, exit, if EO > $1F_{16}$;

(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.

An integer zero, $0000\ 0000_{16}$, is converted to an extended floating point zero, $0000\ 0000\ 0000_{16}$.

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
S	XBR RA	8 | 4 | 4 — 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	
B	CB	BR,DSPL	| 3 |2 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	CBX	BR,RX	| 4 |0 |BR'| C | RX |	12 ≤ BR ≤ 15 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	| F0 | RA | RX | | ADDR |	
DX	C	RA,ADDR,RX		
			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	F4 RA RX ADDR
DX	CBL	RA,ADDR,RX	

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	F6 | RA | RX | ADDR
DX	DC	RA,ADDR,RX	

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	
B	FCB	BR,DSPL	| 3 |3 |BR'| DSPL |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
			4 2 2 4 4	
BX	FCBX	BR,RX	| 4 |0 |BR'| D | RX |	12 ≤ BR ≤ 15 BR' = BR - 12 RA = R0
			8 4 4 16	
D	FC	RA,ADDR	| F8 | RA | RX | | ADDR |	
DX	FC	RA,ADDR,RX		

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

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	FA | RA | RX | ADDR
DX	EFC RA,ADDR,RX	

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 page 137). 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

Custodian:
Air Force - 11

Reviewing activity:
Air Force - 02

Preparing activity:
Air Force - 11

Project IPSC-F142

INDEX

L	70
LB	70
LBX	70
LI	70
LIM	70
LISN	70
LISP	70
LLB	76
LLBI	76
LM	73
LMP	31
LR	70
LST	67
LSTI	67
LUB	75
LUBI	75
M	110
MB	110
MBX	110
MIM	110
MISN	109
MISP	109
MOV	80
MPEN	30
MR	110
MS	109
MSIM	109
MSR	109
N	125
NEG	102
NIM	125

RSW	30
RVBR	39
S	99
SAR	50
SB	34
SBB	99
SBBX	99
SBI	34
SBR	34
SCR	51
SIM	99
SISP	99
SJS	68
SLBI	86
SLC	44
SLL	41
SLR	49
SMK	29
SOJ	58
SPI	29
SR	99
SRA	43
SRL	42
SRM	82
ST	78
STB	78
STBX	78
STC	79
STCI	79
STI	78
STLB	86
STM	83
STUB	85

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

INSTRUCTIONS: This form is provided to solicit beneficial comments which may improve this document and enhance its use. DoD contractors, government activities, manufacturers, vendors, or other prospective users of the document are invited to submit comments to the government. Fold on lines on reverse side, staple in corner, and send to preparing activity. Attach any pertinent data which may be of use in improving this document. If there are additional papers, attach to form and place both in an envelope addressed to preparing activity. A response will be provided to the submitter, when name and address is provided, within 30 days indicating that the 1426 was received and when any appropriate action on it will be completed.

NOTE: This form shall not be used to submit requests for waivers, deviations or clarification of specification requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document(s) or to amend contractual requirements.

DOCUMENT IDENTIFIER *(Number)* AND TITLE

MIL-STD-1750A

NAME OF ORGANIZATION AND ADDRESS OF SUBMITTER

☐ VENDOR ☐ USER ☐ MANUFACTURER

1. ☐ HAS ANY PART OF THE DOCUMENT CREATED PROBLEMS OR REQUIRED INTERPRETATION IN PROCUREMENT USE? ☐ IS ANY PART OF IT TOO RIGID, RESTRICTIVE, LOOSE OR AMBIGUOUS? PLEASE EXPLAIN BELOW.

A. GIVE PARAGRAPH NUMBER AND WORDING

B. RECOMMENDED WORDING CHANGE

C. REASON FOR RECOMMENDED CHANGE(S)

2. REMARKS

SUBMITTED BY *(Printed or typed name and address — Optional)*	TELEPHONE NO.
	DATE

DD FORM 1 OCT 76 1426 PREVIOUS EDITION WILL BE USED.

FOLD

ASD/ENESS
Wright-Patterson AFB, O 45433

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

POSTAGE AND FEES PAID
DEPARTMENT OF AIR FORCE
DOD-318

U.S. MAIL

ASD/ENESS
Wright-Patterson AFB, O 45433

FOLD