MILITARY STANDARD

SIXTEEN-BIT COMPUTER INSTRUCTION SET ARCHITECTURE

TO ALL HOLDERS OF MIL-STD-1750A (USAF):

1. THE FOLLOWING PAGES OF MIL-STD-1750A (USAF) HAVE BEEN REVISED AND SUPERSEDE THE PAGES LISTED:

NEW PAGE	DATE	SUPERSEDED PAGE	DATE
7	2 July 1980	(REPRINTED WITHOUT CHANGE)	
8	21 May 1982	8	2 July 1980
8a	21 May 1982	-	-
9	21 May 1982	9	2 July 1980
10	2 July 1980	(REPRINTED WITHOUT CHANGE)	
13	21 May 1982	13	2 July 1980
13a	21 May 1982	-	-
14	2 July 1980	(REPRINTED WITHOUT CHANGE)	
15	21 May 1982	15	2 July 1980
15a	21 May 1982	-	-
16	2 July 1980	(REPRINTED WITHOUT CHANGE)	
17	2 July 1980	(REPRINTED WITHOUT CHANGE)	
18	21 May 1982	18	2 July 1980
18a	21 May 1982	-	-
21	2 July 1980	(REPRINTED WITHOUT CHANGE)	
22	21 May 1982	22	2 July 1980
22a	21 May 1982	-	-
25	21 May 1982	25	2 July 1980
26	21 May 1982	26	2 July 1980
27	21 May 1982	27	2 July 1980
28	21 May 1982	28	2 July 1980
29	21 May 1982	29	2 July 1980
30	21 May 1982	30	2 July 1980
31	21 May 1982	31	2 July 1980
32	21 May 1982	32	2 July 1980
32a	21 May 1982	-	-
33	21 May 1982	33	2 July 1980
33a	21 May 1982	-	-
34	2 July 1980	(REPRINTED WITHOUT CHANGE)	
61	2 July 1980	(REPRINTED WITHOUT CHANGE)	
62	21 May 1982	62	2 July 1980
91	2 July 1980	(REPRINTED WITHOUT CHANGE)	
92	21 May 1982	92	2 July 1980

NO DELIVERABLE DATA REQUIRED BY THIS DOCUMENT.

FSC IPSC

NEW PAGE	DATE	SUPERSEDED PAGE	DATE
93	21 May 1982	93	2 July 1980
94	2 July 1980	(REPRINTED WITHOUT CHANGE)	
95	2 July 1980	(REPRINTED WITHOUT CHANGE)	
96	21 May 1982	96	2 July 1980
97	21 May 1982	97	2 July 1980
97a	21 May 1982	-	-
98	2 July 1980	(REPRINTED WITHOUT CHANGE)	
101	2 July 1980	(REPRINTED WITHOUT CHANGE)	
102	21 May 1982	102	-
103	21 May 1982	103	-
104	2 July 1980	(REPRINTED WITHOUT CHANGE)	
105	2 July 1980	(REPRINTED WITHOUT CHANGE)	
106	21 May 1982	106	2 July 1980
107	21 May 1982	107	2 July 1980
107a	21 May 1982	-	-
108	2 July 1980	(REPRINTED WITHOUT CHANGE)	
117	21 May 1982	117	2 July 1980
117a	21 May 1982	-	-
118	21 May 1982	118	2 July 1980
119	2 July 1980	(REPRINTED WITHOUT CHANGE)	
120	21 May 1982	120	2 July 1980
121	21 May 1982	121	2 July 1980
121a	21 May 1982	-	-
122	2 July 1980	(REPRINTED WITHOUT CHANGE)	
127	21 May 1982	127	2 July 1980
128	2 July 1980	(REPRINTED WITHOUT CHANGE)	
129	21 May 1982	129	2 July 1980
130	2 July 1980	(REPRINTED WITHOUT CHANGE)	
135	21 May 1982	135	2 July 1980
136	2 July 1980	(REPRINTED WITHOUT CHANGE)	
138a	21 May 1982	-	-

2. RETAIN THIS NOTICE AND INSERT BEFORE TABLE OF CONTENTS.

3. Holders of MIL-STD-1750A (USAF) will verify that page changes and additions indicated above have been entered. This notice page will be retained as a check sheet. This issuance, together with appended pages, is a separate publication. Each notice is to be retained by stocking points until the Military Standard is completely revised or canceled.

4. Changes from previous issue. The margins of these page changes are marked with vertical lines to indicate where changes (additions, modifications, corrections) from the previous issue were made. This was done as a convenience only and the Government assumes no liability whatsoever for any inaccuracies in these notations. Bidders and contractors are cautioned to evaluate the requirements of this document based on the entire content irrespective of the marginal notations and relationship to the last previous issue.

Custodians:
Air Force - 11

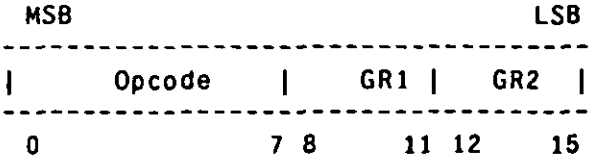
Reviewing activities:
Air Force - 02,90,99

Preparing activity:
Air Force - 11

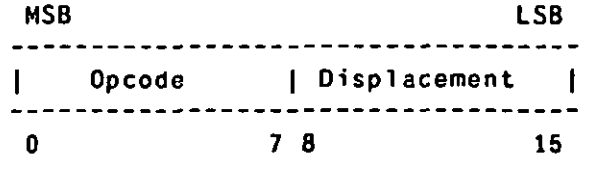
(Project IPSC-F142-01)

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.

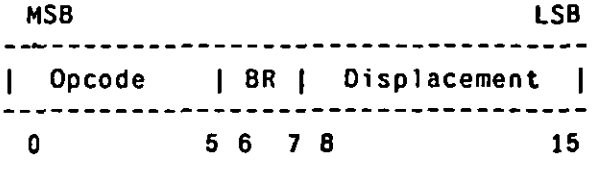
4.2.1 Register-to-register format. The register-to-register format is a 16-bit instruction consisting of an 8-bit opcode and two 4-bit general register (GR) fields that typically specify any of 16 general registers. In addition, these fields may contain a shift count, condition code, opcode extension,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



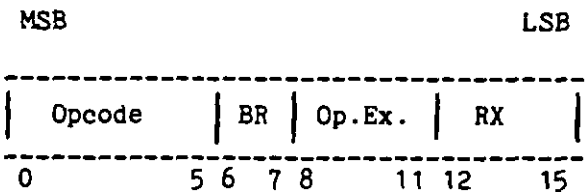
BR = 0 implies general register 12

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.



BR = 0 implies general register 12

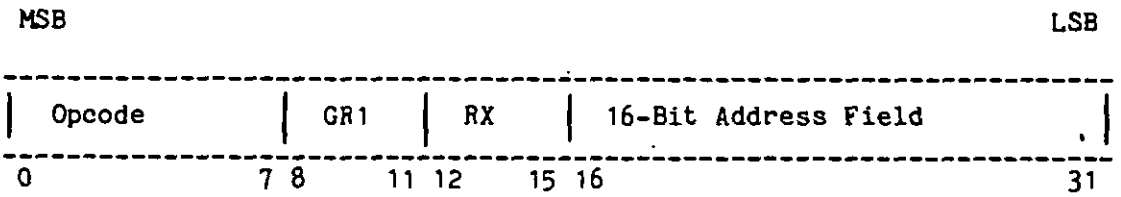
BR = 1 implies general register 13

BR = 2 implies general register 14

BR = 3 implies general register 15

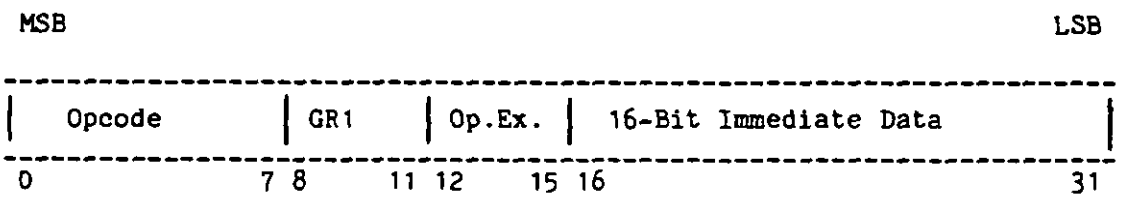
RX = 0 implies no indexing

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

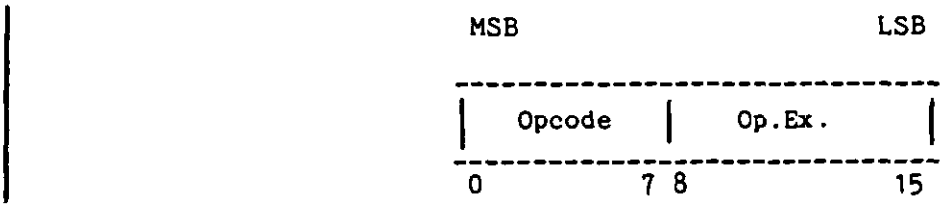


Typically, GR1 is one of the 16 general registers on which the instruction is performing the operation. RX is one of the 15 general registers being used as an index register. The 16-bit address field is either a full 16-bit memory address or a 16-bit operand if the instruction specifies immediate addressing.

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



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



4.3 Addressing modes. Table V 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 allows 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 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.

(THIS PAGE LEFT INTENTIONALLY BLANK)

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
	0 7 8 11 12 15					Extended Precision Floating Point Instructions require addressing of three operands located at DA, DA+1, and DA+2.
1. Register Direct	| O.C. | RA | RB |					
"R"		(RB)	(RB,RB+1)	RB	RB,RB+1	
	0 7 8 11 12 15 16 31					
2. Memory Direct	| O.C. | RA | RX | | A |					
"D"	RX=0 (Non-Indexed)	[A]	[A,A+1]	A	A,A+1	
"DX"	RX≠0 (Indexed)	[A+(RX)]	[A+(RX),A+1+(RX)]	A+(RX)	A+(RX),A+1+(RX)	
	0 7 8 11 12 15 16 31					
3. Memory Indirect	| O.C. | RA | RX | | A |					
"I"	RX=0 (Non-Indexed)	[[A]]	[[A],[A]+1]	[A]	[A],[A]+1	
"IX"	RX≠0 (Indexed)	[[A+(RX)]]	[[A+(RX)],[A+(RX)]+1]	[A+(RX)]	[A+(RX)],[A+(RX)]+1	
4. Immediate Long						
	0 7 8 11 12 15 16 31					
a. Not Indexible	| O.C. | RA |OCX | | I |					
"IM"		I	-----	---	-----	
	0 7 8 11 12 15 16 31					
b. Indexible	| O.C. | RA | RX | | I |					
"IM"	RX=0 (Non-Indexed)	I	-----	---	-----	
"IMX"	RX≠0 (Indexed)	I+(RX)	-----	---	-----	
5. Immediate Short						
	0 7 8 11 12 15					
a. Positive	| O.C. | RA | I |					
"ISP"		+(I+1)	-----	---	-----	
	0 7 8 11 12 15					
b. Negative	| O.C. | RA | I |					
"ISN"		-(I+1)	-----	---	-----	
	0 7 8 15					
6. IC Relative	| O.C. | D |					
"ICR"		---	-----	D+(IC-1)	-----	-128≤D≤127
7. Base Relative						
	0 5 6 7 8 15					Base registers, BR=R12, R13, R14, and R15.
a. Not Indexible	| O.C. |BR'| DU |					
"B"	BR'=BR-12	[DU+(BR)]	[DU+(BR),DU+1+(BR)]	DU+(BR)	DU+(BR),DU+1+(BR)	0≤DU≤255
	0 5 6 7 8 11 12 15					
b. Indexible	| O.C. |BR'|OCX | RX |					
"B"	RX=0 (Non-Indexed)	[(BR)]	[(BR),(BR)+1]	(BR)	(BR),(BR)+1	
"BX"	RX≠0 (Indexed)	[(BR)+(RX)]	[(BR)+(RX),(BR)+1+(RX)]	(BR)+(RX)	(BR)+(RX),(BR)+1+(RX)	

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.

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.

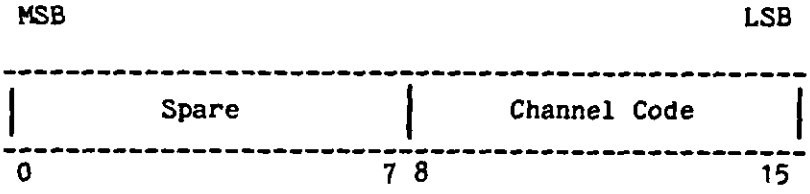
Bit 13: Built-in Test Fault. Hardware built-in test equipment (BITE) error has been detected.

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.

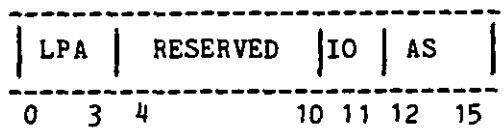
4.4.2.5 Pending interrupt register (PI). The pending interrupt request register is software and hardware controlled and contains the pending interrupts that are attempting to vector the instruction counter. A pending interrupt is set by a system interrupt signal. The pending interrupt bit that generates 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/ouput interrupt code registers (IOIC)(optional). The input/output interrupt code registers, if implemented, are used to indicate which channel generated the input/output interrupt. One register is assigned for each of the two input/output interrupts. Each register is set by hardware to reflect the address of the highest priority channel requesting that level of interrupt. The address shall be 00_{16} for channel number 0, $0F_{16}$ for channel number 15, $7F_{16}$ for channel number 127, etc. The IOICs shall not be altered once the interrupt sequence has commenced until they are read by an I/O instruction.



4.4.2.7 Page registers (optional). Up to 512 sixteen bit registers for optional expanded memory addressing.

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



LPA: Address of page register within the set.

RESERVED: Must not be used.

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

AS: Address of selected group.

(THIS PAGE LEFT INTENTIONALLY BLANK)

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}.

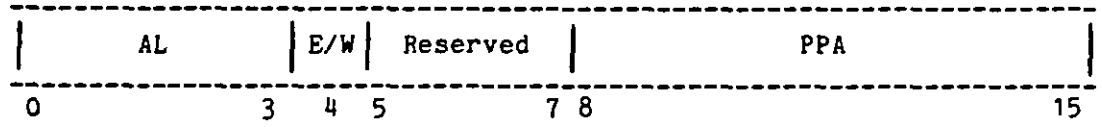
4.5.1.2 Memory addressing boundary constraints. There shall be no odd or even memory address boundary constraints.

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. If expanded memory addressing is implemented, then devices other than the CPU which access memory may utilize either an unmapped 20-bit physical address or a mapped 16-bit logical address. If the devices other than the CPU which access memory utilize 16-bit addressing, a separate address state value must be provided.

4.5.2.1 Group selection. During instruction and operand references to memory, the address state (AS) field of the status word shall be used to designate the page register group. During an interrupt recognition sequence, the operand set of group zero shall be used for vector table and service pointer references to memory; while the linkage pointer references to memory shall use the operand set specified by the AS of the new status word. During memory accesses by

devices other than the CPU which utilize 16-bit logical addressing, the address state value provided by the device shall be used to designate the page register group. Device accesses shall utilize the operand set of the selected group.

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



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

For each of the possible 16 values of the AL code, access shall be permitted for the reference according to table VII. 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

(THIS PAGE LEFT INTENTIONALLY BLANK)

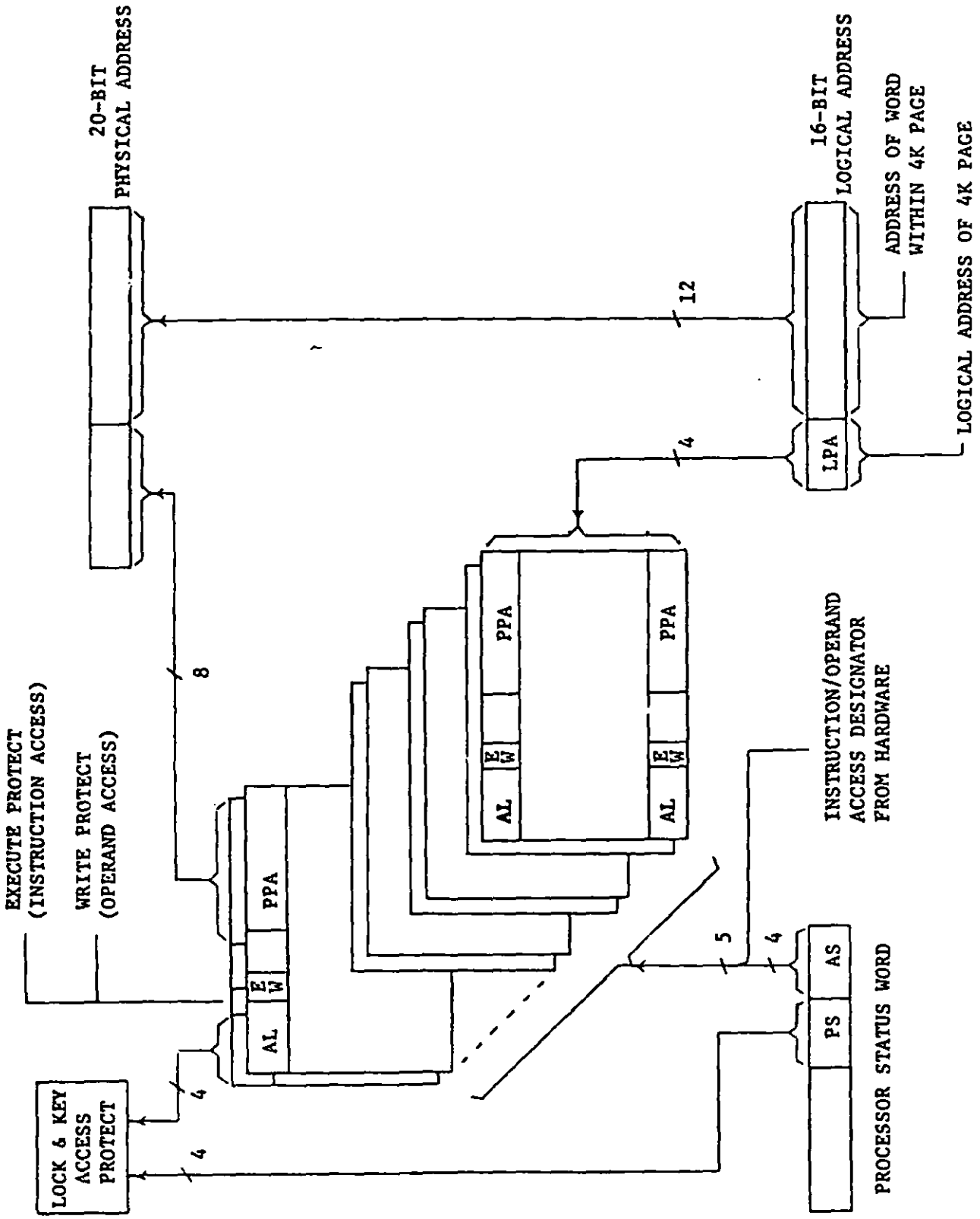


FIGURE 1. Expanded memory mapping diagram

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

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

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

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

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

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

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

4.5.7 Reserved memory locations. Locations 2 through $1F_{16}$ are reserved. Locations 20_{16} through $3F_{16}$ are used by the hardware and the stored program as defined by table VIII.

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, except for those interrupts that cannot be disabled. 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

(THIS PAGE LEFT INTENTIONALLY BLANK)

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.

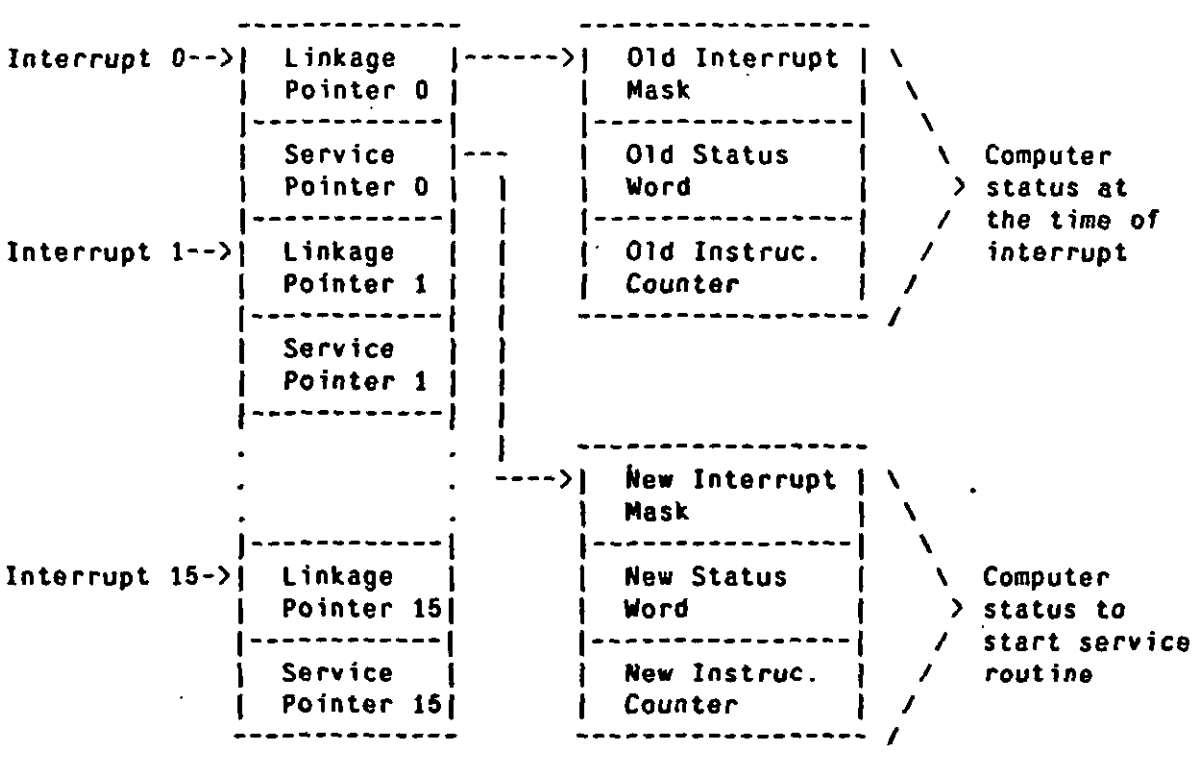


FIGURE 3. Interrupt vectoring system

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. The Built-In-Function is an exception; implemented Built-In-Functions do not cause FT_9 to be set or the machine error interrupt to be generated. All undefined bit patterns in the first 16 bits of an instruction are reserved.

4.8.2 Mnemonic conventions. Each instruction has an associated mnemonic convention. In general, the operation is one or two letters, e.g., L for 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 IM, e.g., AIM for single

(THIS PAGE LEFT INTENTIONALLY BLANK)

EO	Floating point 8-bit 2's complement Derived Operand characteristic (exponent): DO_{24-31}
MA	Floating point register accumulator mantissa (fractional part): $(RA,RA+1)_{0-23}$ (Ft.P.), $(RA,RA+1)_{0-23}$ $(RA+2)_{32-47}$ (E.F.P.)
EA	Floating point 8-bit 2's complement register accumulator characteristic (exponent): $(RA,RA+1)_{24-31}$
RQ, MP, MQ, RR	Entities used for register level transfer description clarification. These registers are not part of the general register file. Q = quotient, P = product, R = remainder.

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 of 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, than 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

	LOAD STORE	INTEGER ARITHMETIC	FLOATING POINT	LOGICAL COMPARE	OPCODE EXTENTIONS	BIT	SHIFT	JUMP	LOAD	STORE	ADD	SUB	MULT	DIVIDE	LOGICAL COMPARE
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LB BR12	AB BR12	FAB BR12	ORB BR12	BRX BR12*	SB	SLL	JC	L	ST	A	S	MS	DV	OR	C
1	LB BR13	AB BR13	FAB BR13	ORB BR13	BRX BR13*	SBR	SRL	JCI	LR	STC	AR	SR	MSR	DVR	ORR	CR
2	LB BR14	AB BR14	FAB BR14	ORB BR14	BRX BR14*	SBI	SRA	JS	LISP	STCI	AISP	SISP	MISP	DISP	AND	CISP
3	LB BR15	AB BR15	FAB BR15	ORB BR15	BRX BR15*	RB	SLC	SOJ	LISN	MOV	INCM	DECM	MISN	DISN	ANDR	CISN
4	DLB BR12	SBB BR12	FSB BR12	ANDB BR12		RBR		BR	LI	STI	ABS	NEG	M	D	XDR	CBL
5	DLB BR13	SBB BR13	FSB BR13	ANDB BR13		RBI	DSLL	BEZ	LIM		DABS	DNEG	MR	DR	XDRR	
6	DLB BR14	SBB BR14	FSB BR14	ANDB BR14		TB	DSRL	BLT	DL	DST	DA	DS	DM	DD	N	DC
7	DLB BR15	SBB BR15	FSB BR15	ANDB BR15		TBR	DSRA	BEX	DLR	SBM	DAR	DSR	DMR	DDR	NR	DCR
8	STB BR12	MB BR12	FMB BR12	CB BR12	XIO*#	TBI	DSLC	BLE	DLI	DSTI	FA	FS	FM	FD	FIX	FC
9	STB BR13	MB BR13	FMB BR13	CB BR13	VIO*#	TSB		BGT	LM	STM	FAR	FSR	FMR	FDR	FLT	FCR
A	STB BR14	MB BR14	FMB BR14	CB BR14	INOL*	SVBR	SLR	BNZ	EFL	EFST	EFA	EFS	EFM	EFD	EFIX	EFC
B	STB BR15	MB BR15	FMB BR15	CB BR15			SAR	BGE	LUB	STUB	EFAR	EFSR	EFMR	EFDR	EFLT	EFCR
C	DSTB BR12	DB BR12	FDB BR12	FCB BR12		RVBR	SCR	LST1#	LLB	STLB	FABS	FNEG			XBR	
D	DSTB BR13	DB BR13	FDB BR13	FCB BR13			DSLR	LST#	LUBI	SUBI					XWR	
E	DSTB BR14	DB BR14	FDB BR14	FCB BR14		TVBR	DSAR	BJS	LLBI	SLBI						
F	DSTB BR15	DB BR15	FDB BR15	FCB BR15	BIF**		DSCR	URS	POPM	PSHM						NOP/BPT

* *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*

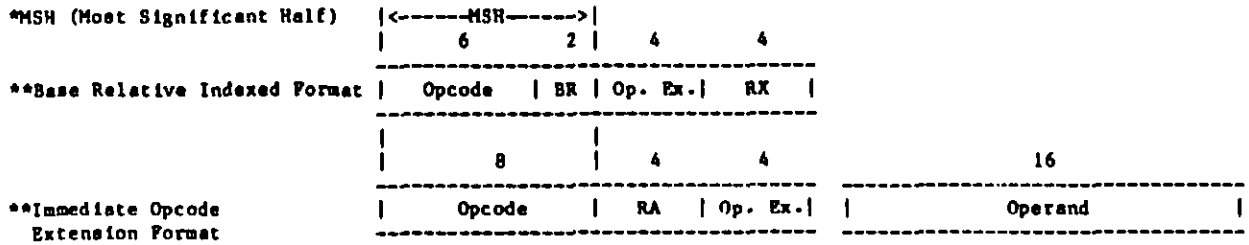
**User Defined Built-In Function Opcode.

TABLE XI. Extended operation codes

Opcode Extension (see below for location)

* MSH	** Format	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	BRX BR12	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
41	BRX BR13	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
42	BRX BR14	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
43	BRX BR15	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
4A	IMML		AIM	SIM	MIM	MSIM	DIM	DVIM	ANDM	ORIM	XORM	CIM	NIM				

*MSH (Most Significant Half)



These Extended Operation Codes do not include those for Built-In-Function (BIF).
The Extended Operation Codes for these instructions are within each instruction's definition.

5 DETAILED REQUIREMENTS

5.1 Execute input/output.

ADDR MODE	MNEMONIC	FORMAT/OPCODE

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

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: The command disables all interrupts (except those that are defined such that they cannot be disabled) at the beginning of the execution of the DSBL instruction.

2004 RPI — Reset Pending Interrupt: The individual interrupt bit to be reset shall be designated in register RA as a right justified four bit code. (0_{16} represents interrupt number 0, F_{16} represents interrupt number 15). If interrupt 1_{16} is to be cleared, then the contents of the fault register shall also be set to zero.

2005 SPI — Set Pending Interrupt Register: This command ORs the 16-bit contents of RA with the pending interrupt register. If there is a one in the corresponding bit position of the interrupt 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.

A000 RMK — Read Interrupt Mask: The current interrupt mask is transfered into register RA. The interrupt mask is not altered.

A004 RPIR — Read Pending Interrupt Register: This command transfers the contents of the pending interrupt register into RA. The pending interrupt register is not altered.

A00E RSW — Read Status Word: This command transfers the 16-bit status word into register RA. The status word remains unchanged.

A00F RCFR — Read and Clear Fault Register: This command inputs the 16-bit fault register to register RA. The contents of the fault register are reset to zero. Bit 1 in the pending interrupt register is reset to zero.

Optional 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.

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 the the corresponding 1024 word physical memory block. The RAM word MSB (bit 0) represents the lowest number block and the RAM word LSB (bit 15) represents the highest block (i.e., bit 0 represents locations 0 through 1023 and bit 15 represents locations 15360 through 16383 for word zero). Each word represents consecutive 16K blocks of physical memory. 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 of group X.

52XY WOPR — Write Operand Page Register: This command transfers the contents of register RA to page register Y of the operand set of group X.

8YXX PI — Programmed Input: This command inputs 16 bits of data into RA from the programmed I/O port. Y may be from 0 through 3.

A001 RIC1 — Read Input/Output Interrupt Code, Level 1: This command inputs the contents of the level 1 IOIC register into register RA. The Channel number is right justified.

A002 RIC2 — Read Input/Output Interrupt Code, Level 2: This command inputs the contents of the level 2 IOIC register into register RA. The channel number is right justified.

A008 RDOR — Read Discrete Output Register: This command inputs the 16-bit discrete output buffer into register RA.

A009 RDI — Read Discrete Input: This command inputs the 16-bit discrete input word into register RA. A "1" indicates an "on" condition and a "0" indicates an "off" condition.

A00B TPIO — Test Programmed Output: This command inputs the 16-bit contents of the programmed output buffer into register RA. This command may be used to test the PIO channel by means of a wrap around test.

A00D RMFS — Read Memory Fault Status: This command transfers the 16-bit contents of the memory fault status register to RA. The fields within the memory fault status register shall delineate memory related fault types and shall provide the page register designators associated with the designated fault.

C000 CI — Console Input: This command inputs the 16-bits (2 bytes) from the console into register RA. The eight most significant bits of RA shall represent the first byte.

C001 RCS — Read Console Status: This command inputs the console interface status into register RA. The status is right justified.

C00A ITA — Input Timer A: This command inputs the 16-bit contents of timer A into register RA. Bit fifteen is the least significant bit and represents a time increment of ten microseconds.

C00E ITB — Input Timer B: This command inputs the 16-bit contents of timer B into register RA. Bit fifteen is the least significant bit and represents a time increment of one hundred microseconds.

D0XX RMP — Read Memory Protect RAM (D000 + RAM Address): This command inputs the appropriate memory protect word into register RA. A "1" in a bit provides write protection and a "0" in a bit permits writing to the corresponding 1024 word physical memory block. The RAM words MSB (bit 0) represents the lowest number block and the RAM word LSB (bit 15) represents the highest block (i.e., bit 0 represents locations 0 through 1023 and bit 15 represents locations 15360 through 16383 for word zero). Each word represents consecutive 16K blocks of physical memory. The RAM words of 0 through 63 apply to processor write protect and words 64 through 127 apply to DMA write protect.

D1XY RIPR — Read Instruction Page Register: This command transfers the 16-bit contents of the page register Y of the instruction set of group X to register RA.

D2XY ROPR — Read Operand Page Register: This command transfers the 16-bit contents of page register Y of the operand set of group X to register RA.

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

REGISTER TRANSFER DESCRIPTION: Varies depending on the command field.

REGISTERS AFFECTED: Varies depending on the command field.

(THIS PAGE LEFT INTENTIONALLY BLANK)

5.2 Vectored input/output.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
D	VIO RA,ADDR	8 | 4 | 4 | 16
DX	VIO RA,ADDR,RX	49 | RA | RX | ADDR

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

DA	CMD	
DA+1	Vector Select	
DA+2	Data	} one data word for each bit set in the vector select
...	...	

The input/output operation or device address is specified by the sum of the CMD and the product of the bit number of the bit set in the vector select times the contents of RA. This device address is then interpreted as specified by the XIO instruction (see 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. If an illegal XIO command is encountered as part of a VIO chain, the following actions occur:

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

REGISTER TRANSFER DESCRIPTION:

Step 1. n <-- 0 and i <-- 0;

Step 2. if $[DA+1]_n=1$, then I/O command = [DA] + n x (RA) ;

Step 3. FT_5 <-- 1, exit, if XIO = illegal command;

Step 4. if $[DA+1]_n=1$, then I/O data = [DA + 2 + i];

Step 5. if $[DA+1]_n=1$, then i <-- i+1;

Step 6. n <-- n + 1, exit, if n = 16;

Step 7. go to step 2;

REGISTERS AFFECTED: None

(THIS PAGE LEFT INTENTIONALLY BLANK)

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

5.29 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 (N+1)th location following the SW in address state zero. 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;

(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

5.56 Increment memory by a positive integer.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
D	INCM N,ADDR	8 4 4 16
DX	INCM N,ADDR,RX	| A3 |N-1 | RX | | ADDR |

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

(RA) <-- |DO|;

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

(CS) <-- 0001 if (RA) = 8000_{16};

(CS) <-- 0010 if (RA) = 0;

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

REGISTERS AFFECTED: RA, CS, PI

5.58 Double precision absolute value of register.

ADDR MODE	MNEMONIC		FORMAT/OPCODE		
			8	4	4
R	DABS	RA,RB	A5	RA	RB

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

Note: RA may equal RB.

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

REGISTER TRANSFER DESCRIPTION:

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

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

(CS) <-- 0001 if (RA, RA+1) = $8000\ 0000_{16}$;

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

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

5.59 Double precision integer add.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
			8 4 4
R	DAR	RA,RB	| A7 | RA | RB |
			8 4 4 16
0	DA	RA,ADDR	
DX	DA	RA,ADDR,RX	| A6 | RA | RX | | ADDR |

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

REGISTER TRANSFER DESCRIPTION:

$(RA,RA+1)^2$ <-- $(RA,RA+1)^1$ + DO;

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

(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		
DX	FA	RA,ADDR,RX	| A8 | RA | RX | | ADDR |	

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

REGISTER TRANSFER DESCRIPTION:

N = EA - EO;

EA <-- EO, if MA = 0;

MO <-- MO Shifted Right Arithmetic n positions, if n > 0 and MA ≠ 0;

MA <-- MA Shifted Right Arithmetic -n positions, EA <-- EO, if n < 0 and MO ≠ 0;

MA <-- MA + MO;

MA <-- MA Shifted Right Arithmetic 1 position, 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;

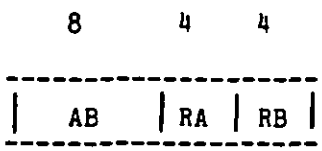
PI_6 <-- 1, EA <-- 0, MA <-- 0, if EA < 80_{16};

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

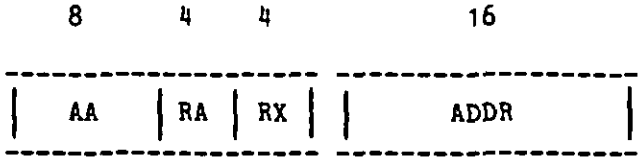
REGISTERS AFFECTED: RA, RA+1, CS, PI

5.61 Extended precision floating point add

ADDR MODE	MNEMONIC		FORMAT/OPCODE



ADDR MODE	MNEMONIC		FORMAT/OPCODE
D	EFA	RA,ADDR	
DX	EFA	RA,ADDR,RX	



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

REGISTER TRANSFER DESCRIPTION:

n = EA - EO;

EA <-- EO, if MA = 0;

MO <-- MO Shifted Right Arithmetic n positions, if n > 0 and MA ≠ 0;

MA <-- MA Shifted Right Arithmetic -n positions, EA <-- EO, if n < 0 and MO ≠ 0;

MA <-- MA + MO;

MA <-- MA Shifted Right Arithmetic 1 position, MA_0, <-- $\overline{MA}_0$, EA <-- EA+1, if OVM = 1;

PI_3 <-- 1, EA <-- $7F_{16}$, MS <-- 7FFF FF $FFFF_{16}$, exit, if EA > $7F_{16}$ and MA_0 = 0;

PI_3 <-- 1, EA <-- $7F_{16}$, MS <-- 8000 00 0000_{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, 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

(THIS PAGE LEFT INTENTIONALLY BLANK)

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

5.64 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:

(RA) <-- -DO;

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

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

REGISTERS AFFECTED: RA, CS, PI

5.66 Double precision negate register.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
R	DNEG RA,RB	8 4 4 | 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

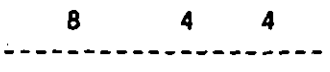
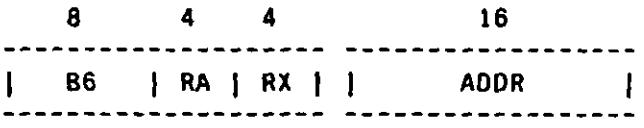
(RA, RA+1) <-- -DO;

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

(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

5.67 Double precision integer subtract.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
R	DSR	RA,RB	
D	DS	RA,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

5.68 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 - EO;

EA <-- EO, if MA = 0;

MO <-- MO Shifted Right Arithmetic n positions, if n > 0 and MA ≠ 0;

MA <-- MA Shifted Right Arithmetic -n positions, EA <-- EO, if n < 0 and MO ≠ 0;

MA <-- MA - MO;

MA <-- MA Shifted Right Arithmetic 1 position, 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;

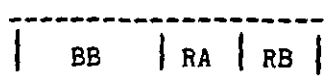
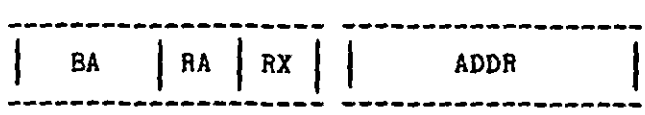
EA, MA <-- normalized EA, MA;

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
R	EFSR RA,RB	
D DX	EFS RA,ADDR EFS RA,ADDR,RX	

DESCRIPTION: The extended precision floating point Derived Operand, DO, is extended floating point subtracted from 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 subtracted. If the difference overflows the 39-bit mantissa, then the difference is shifted right one position, the sign bit restored, and the exponent is incremented. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the difference does not result in exponent overflow, the result is normalized. If during the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

REGISTER TRANSFER DESCRIPTION:

n = EA - EO;

EA <-- EO, if MA = 0;

MO <-- MO Shifted Right Arithmetic n positions, if n > 0 and MA ≠ 0;

MA <-- MA Shifted Right Arithmetic -n positions, EA <-- EO, if n < 0 and MO ≠ 0;

MA <-- MA - MO;

MA <-- MA Shifted Right Arithmetic 1 position, MA_0 <-- $\overline{MA_0}$, EA <-- EA+1, if OVM = 1;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF $FFFF_{16}$, exit, if EA > $7F_{16}$ and MA_0 = 0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00 0000_{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, 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:

```
PI3 <-- 1, EA <-- 7F16, MO <-- 7FFF FF16, exit, if DO = 8000 007F16;

PI6 <-- 1, EA <-- 0, MA <-- 0, exit, if DO = 4000 0080₁₆;

EA <-- EO+1, MA <-- 4000 00₁₆, if MO = 8000 00₁₆;

EA <-- EO-1, MA <-- 8000 00₁₆, if MO = 4000 00₁₆;

EA <-- EO, MA <-- -MO, if MO ≠ 8000 00₁₆ or 4000 00₁₆;

(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

5.77 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 |	$12 \leq BR \leq 15$ BR' = BR - 12 RA = R2
			4 2 2 4 4	
BX	DBX	BR,RX	| 4 | 0 | BR' | 7 | RX |	$12 \leq BR \leq 15$ BR' = BR - 12 RA = R2
			8 4 4 16	
D DX	D D	RA,ADDR RA,ADDR,RX	| D4 | RA | RX | | ADDR |	
			8 4 4 16	
IM	DIM	RA,DATA	| 4A | RA | 5 | | DATA |	

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

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

REGISTER TRANSFER DESCRIPTION:

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

PI_4 <-- 1, if DO = 0 or (RQ, RQ+1) > 0000 $7FFF_{16}$ or (RQ, RQ+1) < $FFFF\ 8000_{16}$

(RA) <-- (RQ+1)

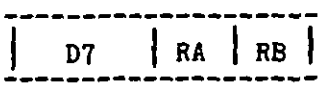
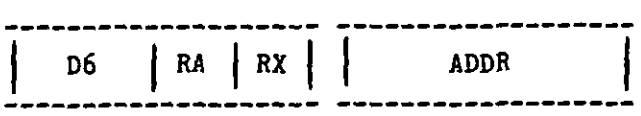
(RA+1) <-- (RR)

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

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

(THIS PAGE LEFT INTENTIONALLY BLANK)

5.78 Double precision integer divide.

ADDR MODE	MNEMONIC		FORMAT/OPCODE
R	DDR	RA,RB	
D	DD	RA,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 $8000\ 0000_{16}$ and the divisor is $FFFF\ FFFF_{16}$.

REGISTER TRANSFER DESCRIPTION:

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

PI_4 <-- 1, if DO = 0 or [RA, RA+1 = $8000\ 0000_{16}$ and DO = $FFFF\ FFFF_{16}$];

(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		
DX	FD	RA,ADDR,RX	| D8 | RA | RX | | ADDR |	

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

REGISTER TRANSFER DESCRIPTION:

n = EA - EO;

n <-- 0, if MA = 0

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

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

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

MQ <-- MA / MO;

MQ <-- MQ Shift Right Arithmetic 1 position, n <-- n + 1, if MQ $\geq$ 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;

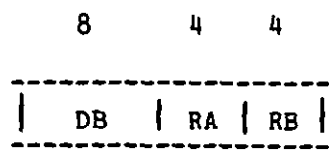
MA <-- MQ_{0-23};

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

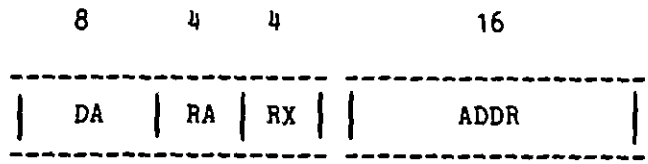
REGISTERS AFFECTED: RA, RA+1, CS, PI

1.80 Extended precision floating point divide.

ADDR MODE	MNEMONIC	FORMAT/OPCODE



ADDR MODE	MNEMONIC	FORMAT/OPCODE



ADDR MODE	MNEMONIC
D	EFD RA,ADDR
DX	EFD RA,ADDR,RX

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, 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;

n <-- 0, if MA = 0;

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

PI_3 <-- 1, EA <-- $7F_{16}$, and MA <-- 8000 00 0000_{16}, exit, if MA_0 ≠ MO_0
and {n > $7F_{16}$ or DO = 0};

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

MQ <-- MA / MO;

MQ <-- MQ Shift Right Arithmetic 1 position, n <-- n + 1, if MQ ≥ 1.0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 7FFF FF $FFFF_{16}$, exit, if n > $7F_{16}$ and MQ_0 = 0;

PI_3 <-- 1, EA <-- $7F_{16}$, MA <-- 8000 00 0000_{16}, exit, if n > $7F_{16}$ and MQ_0 = 1;

EA <-- n;

MA <-- MQ_{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

(THIS PAGE LEFT INTENTIONALLY BLANK)

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

5.86 Convert 16-bit integer to floating point.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
		8 4 4
R	FLT RA,RB	E9 | RA | RB

DESCRIPTION: The integer Derived Operand, DO (i.e., the contents of register RB), is converted to Single Precision floating point format and stored in register RA and RA+1. The condition status, CS, is set based on the results in RA and RA+1. The operation process is as follows: The exponent is initially considered to be $0F_{16}$. The integer value in RB is normalized, i.e., the number is left shifted and the exponent decremented for each shift until the sign bit and the next MSB are unequal, and the exponent and mantissa stored in the proper fields of RA and RA+1.

Note: RA may equal RB.

REGISTER TRANSFER DESCRIPTION:

```
EA <-- 0, MA <-- 0, exit, if (RB) = 0;

EA <-- 0F16;

MA <-- (RB);

EA, MA <-- normalize EA, MA;

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

REGISTERS AFFECTED: RA, RA+1, CS

5.87 Convert extended precision floating point to 32-bit integer.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
		8 4 4
R	EFIX RA,RB	EA | RA | RB

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

Note: The algorithm truncates toward zero.

REGISTER TRANSFER DESCRIPTION:

```
PI4 <-- 1, exit, if EO > 1F16;

(RA,RA+1) <-- Integer portion of DO;

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

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

5.88 Convert 32-bit integer to extended precision floating point.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
		8 4 4
R	EFLT RA,RB	EB | RA | RB

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

Note: RA may equal RB.

REGISTER TRANSFER DESCRIPTION:

EA <-- 0, MA <-- 0, exit, if (RB,RB+1) = 0;

EA <-- $1F_{16}$, MA <-- (RB,RB+1);

EA, MA <-- normalized EA, MA;

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

REGISTERS AFFECTED: RA, RA+1, RA+2, CS

5.89 Exchange bytes in register.

ADDR MODE	MNEMONIC	FORMAT/OPCODE
		8 4 4
S	XBR RA	EC | RA | 0

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

REGISTER TRANSFER DESCRIPTION:

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

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

REGISTERS AFFECTED: RA, CS

5.94 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 DX	FC FC	RA,ADDR RA,ADDR,RX	F8 | RA | RX | ADDR	

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

Note: This instruction does not cause an overflow to occur.

REGISTER TRANSFER DESCRIPTION:

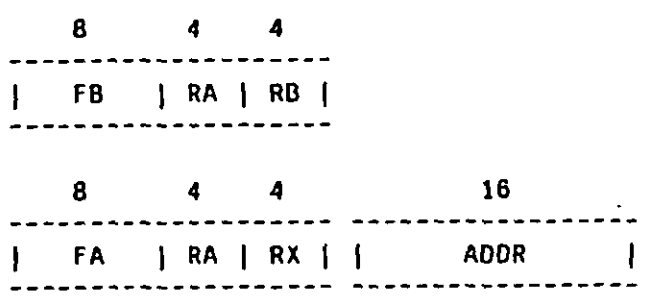
```
(RA, RA+1)  :  DO;

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

REGISTERS AFFECTED: CS

5.95 Extended precision floating point compare.

ADDR MODE	MNEMONIC	FORMAT/OPCODE



R	EFCR RA,RB	
D	EFC RA,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

5.98 Built-In-Function:

ADDR MODE	MNEMONIC	FORMAT/OPCODE	
		8	8
S	BIF Op. Ex.	4F	Op. Ex.

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

REGISTER TRANSFER DESCRIPTION: User defined.

REGISTERS AFFECTED: User defined.

☆U.S. GOVERNMENT PRINTING OFFICE: 1982-505-022/3010

(THIS PAGE LEFT INTENTIONALLY BLANK)