1
INTRODUCTION

This manual describes the functionality and provides the theory of operation for the Level 6 Model 43/53 Central Processor Unit (CPU).

1.1 GENERAL DESCRIPTION

The CPU is a high-speed general purpose digital computer designed to process data from main memory and associated system devices. Its logic elements are fabricated on a 15- by 16-inch control board and a 12- by 14-inch four-layer CPU board. The CPU board is physically supported by the control board, hereafter referred to as the controller, in a piggyback manner. The controller houses the CPU control logic including some of the CPU registers, clock network, and Megabus* interface logic. The CPU board contains associated functional elements including the microprocessor, Random Access Memory (RAM), and bootstrap Programmable Read Only Memory (PROM). A Memory Management Unit (MMU) and cache memory are available as CPU options and, when included as part of the Model 43/53 system configuration, they are also housed on the CPU board. For details concerning the MMU and cache memory options, refer to the applicable documentation listed in subsection 1.2

CPU communication with main memory and other units of the Level 6 system is over a common bus, called the Megabus (see Figure 1-1). The interface signals between the CPU and the Mega-

*Trademark of Honeywell Information Systems, Inc.

bus are shown in Figure 1-2. The CPU can also operate in a multi-processor environment consisting of up to four central processing units, and each CPU can communicate directly with the following optional processor type controllers over the Megabus.

- Commercial Instruction Processor (CIP)
- Scientific Instruction Processor (SIP)

For details concerning the CIP and SIP options, refer to the applicable documentation listed in subsection 1.2.

The CPU uses hardware, with the aid of firmware, to communicate directly with the MMU or cache memory, to communicate with main memory over the system Megabus, and to decode all CPU instructions, performing the necessary arithmetic, logical, or shift operations (see Figure 1-3). The following list provides additional information relative to the general characteristics of the CPU:

1. 8-, 16-, or 32-bit data
2. Up to two megabytes of directly addressable main memory
3. Bit, byte, word, and double-word registers
4. Bit test, set, and mask capability
5. 26 program visible general registers, including multiple accumulators, multiple address, index, and control registers
6. Immediate, register-to-register, and register-to-memory operations
7. 64 vectored interrupt levels
8. Stack/Queue handling
9. Multiple vectored trap structure
10. Hardware supported context save and restore
11. Multiple addressing modes, including indexing, indirect, base plus displacement, program counter relative, auto increment/decrement, etc.
12. Real-time clock and watchdog timer
13. Power failure detection
14. Automatic restart
15. Bootstrap
16. Writable Control Store (WCS).

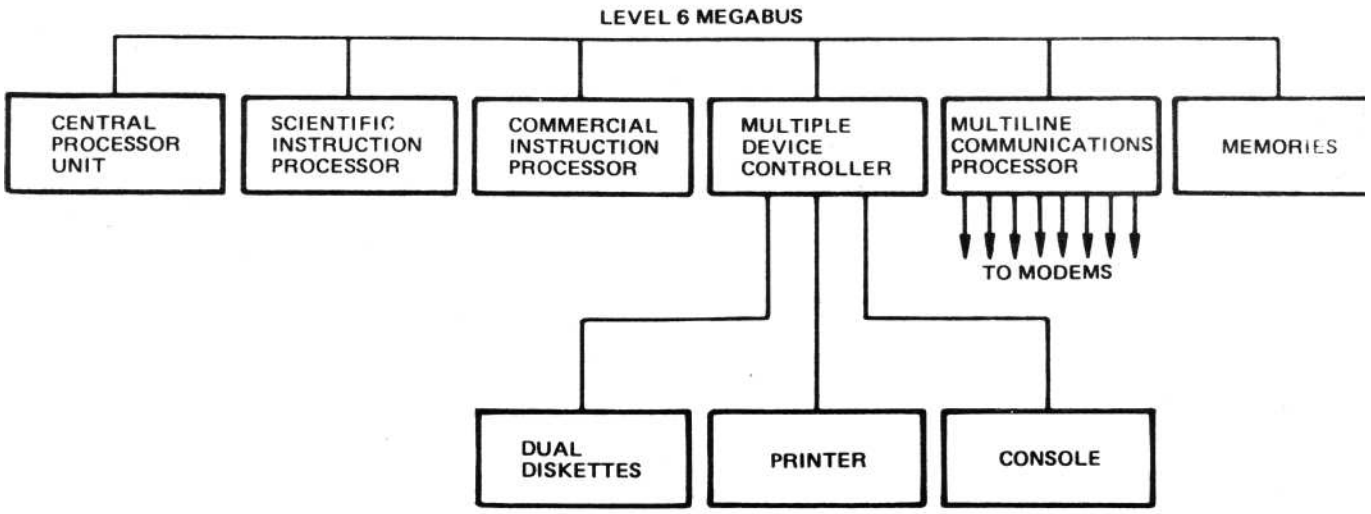

LEVEL 6 MEGABUS
CENTRAL PROCESSOR UNIT
SCIENTIFIC INSTRUCTION PROCESSOR
COMMERCIAL INSTRUCTION PROCESSOR
MULTIPLE DEVICE CONTROLLER
MULTILINE COMMUNICATIONS PROCESSOR
MEMORIES
TO MODEMS
DUAL DISKETTES
PRINTER
CONSOLE

Figure 1-1 CPU Communications Block Diagram

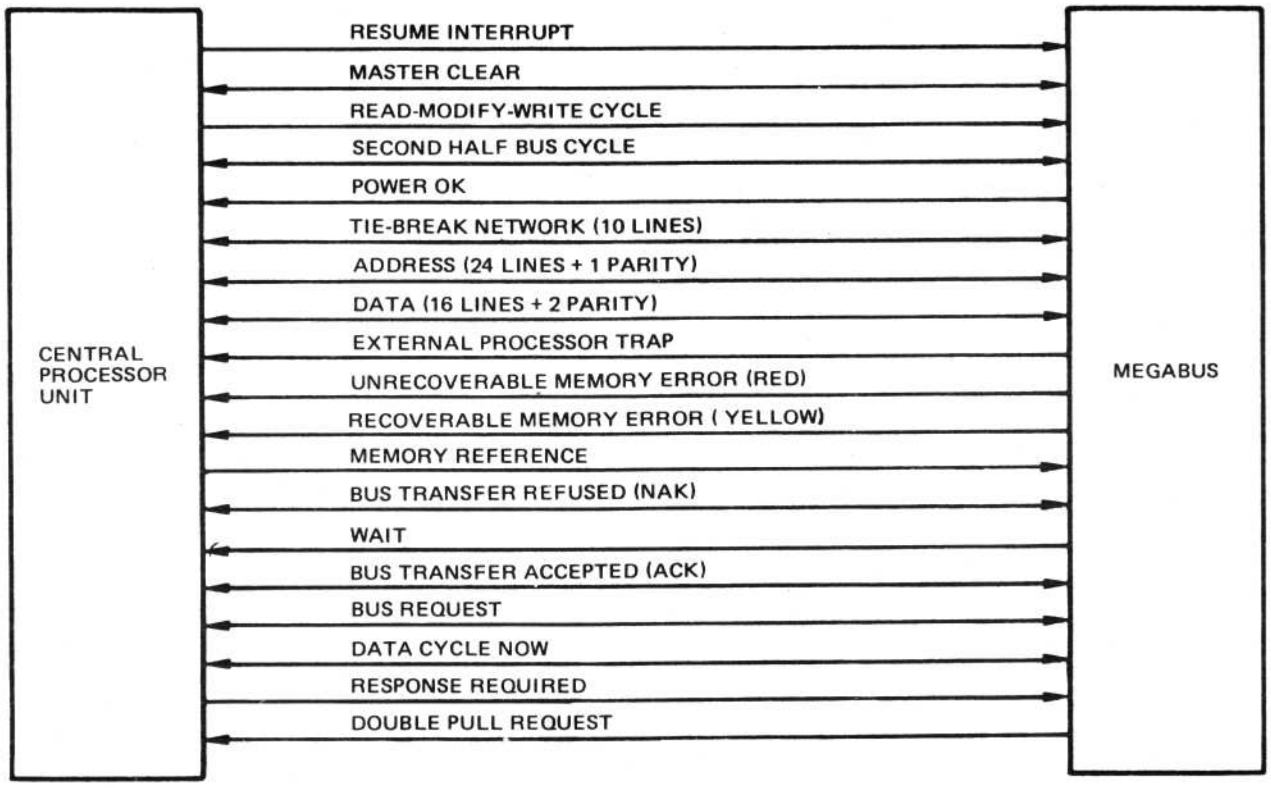


Figure 1-2 CPU/Megabus Interface Signals

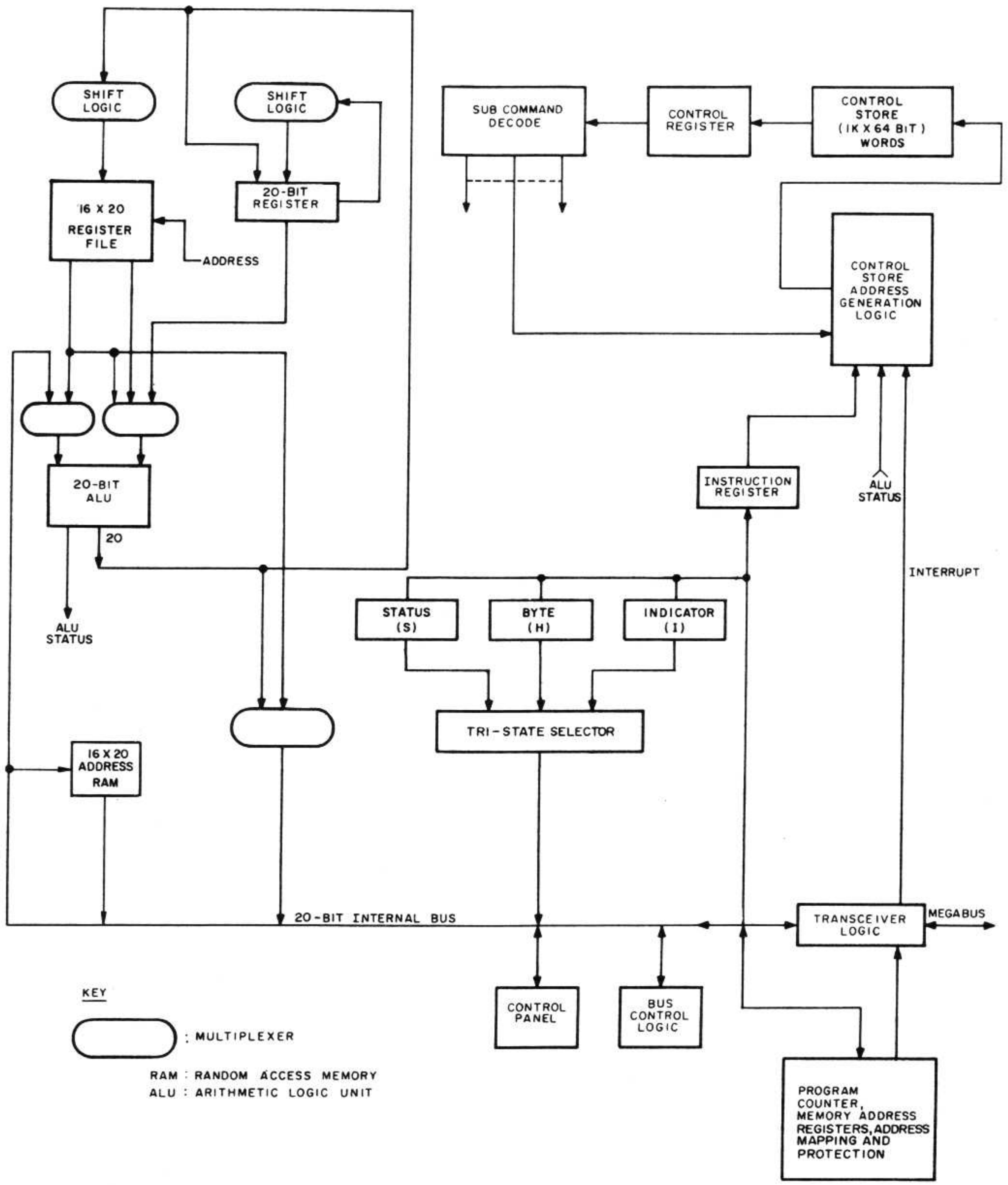


Figure 1-3 CPU Functional Block Diagram

1.2 REFERENCE DOCUMENTATION

The following support documents for the Model 43/53 CPU contain supplementary information for the data presented in this manual.

	DESCRIPTION	ORDER NUMBER
1.	Model 53 CPU Manual - Volume 2 (cache memory)	FQ30
2.	Model 53 CPU Manual - Volume 3 (MMU)	FW85
3.	Model 47 CPU Manual - Volume 2 (CIP)	FW82
4.	Type CPF9503 Scientific Instruction Processor Manual	FN30

1.3 SOFTWARE VISIBLE REGISTERS

Twenty six program visible registers, counters, and discrete signal lines are available for use by software to maintain software related data or instructions, and to display selected CPU status conditions for software interrogation. These elements include:

- Seven base registers
- Seven data registers
- Indicator register
- Seven mode registers
- Program counter
- Remote data descriptor base register
- Status/security register
- Stack pointer.

The software application for each of the above is described in the following subsections.

1.3.1 Base Registers (Bl Through B7)

Seven base registers (Bl through B7) are located in the register file portion of the microprocessor (refer to subsection 4.4.2), and maintain a 20-bit address of any procedure, data, array, or arbitrary location in memory.

1.3.2 Data Registers (Dl Through D7)

Seven data registers (Dl through D7) are located in the register file portion of the microprocessor (refer to subsection 4.4.2), and serve as 16-bit (10 through 1F) general purpose registers or accumulators. Bit 10 of each register is considered the most significant bit. Each data register can also be used for post-indexing of addresses (i.e., as index registers).

NOTE

In software notation these registers are designated R1 through R7.

1.3.3 Indicator Register (I)

The indicator register (I) contains several single bit indicators that provide temporary storage for overflow and program status information.

1.3.4 Mode Registers (M1 Through M7)

The mode registers (M1 through M7) reside in locations 1 through 7 of the CPU Random Access Memory (RAM), refer to subsection 4.9.1. These registers retain mode information pertaining to the CPU and other processors (i.e., CIP and/or SIP), and can be modified by the MTM instruction. Currently, registers M2, M6, and M7 are reserved for future use, while the remaining four registers (M1 and M3 through M5) define the following system features:

- M1 - CPU trace and overflow trap masks
- M3 - CIP truncation and overflow trap masks
- M4 - SIP memory and accumulator length control and round/ truncate mode control
- M5 - SIP exponent underflow, significance, and precision error trap masks.

1.3.5 Program Counter (P Register)

The program counter (or P register) normally contains the storage address of the next instruction to be executed by the CPU.

1.3.6 Remote Descriptor Base Register (RDBR)

The Remote Descriptor Base Register (RDBR) resides in location B of the CPU Random Access Memory (RAM), refer to subsection 4.9.1. The contents of this register point to a table of up to 4,096 data descriptors.

1.3.7 Status/Security Register (S)

The status/security register (S) contains the system status and security keys.

1.3.8 Stack Pointer (T Register)

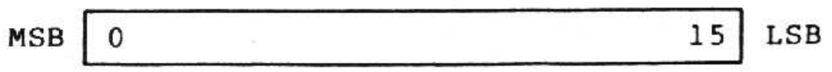
The stack pointer (or T register) resides in location A of the RAM. The contents of this register point to the base of the current stack structure in memory.

1.4 DATA WORD FORMATS

This subsection defines the various data word formats that are used by the CPU.

1.4.1 Memory Data

All data elements, such as a bit or byte, are based on 16-bit memory words. The format of each word is defined from left to right with the first bit numbered 0:

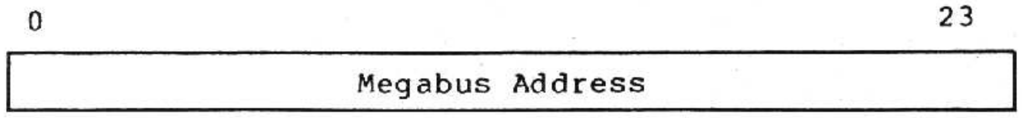


Memory data may be accessed by instructions to the bit, byte, digit, word, or multiword data item level. In all cases, the leftmost element is the most significant element of the word; e.g., bit 0 (above) is the first bit, bit 1 is the second bit, bits 0 through 7 are the first byte, bits 8 through 15 are the second byte, etc. Multiword items require successive word locations; the lowest address is defined as the leftmost or most significant part of the data item.

1.4.2 Addresses

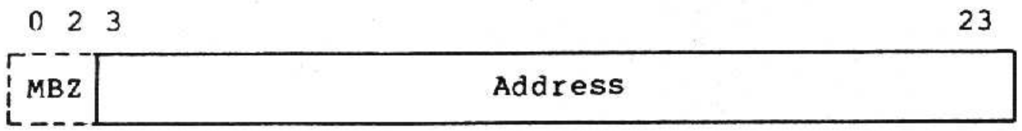
An address pointer is used to point to bit, byte, word, or multiword data items. This address indicates the leftmost and most significant element of the data item. Within an array, data items are numbered from left to right.

The CPU may operate in either Long or Short Address Form (LAF or SAF). LAF provides virtual addressability to 1M words, whereas SAF provides addressability to 64K words. Addresses are unsigned. Physical byte addresses must be presented to the Megabus and must contain 24 bits.



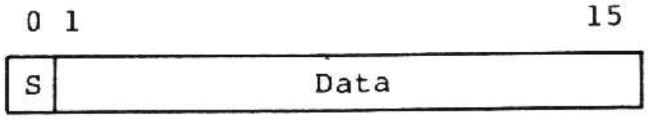
Bit positions of processor address registers are numbered to correspond to their positions on the Megabus with an appropriate number of leading zeros.

The CPU generates addresses which may contain 21 significant bits:



1.4.3 Signed Integer Data Word

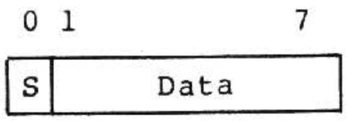
The data field is a 16-bit integer (in two's complement form) with the radix point to the right of bit 15, the least significant bit. Bit 0 indicates the Sign (S) of the data field. The format of the signed integer data word is:



S = Zero, sign is positive.
S = One; sign is negative.

1.4.4 Signed Integer Data Byte

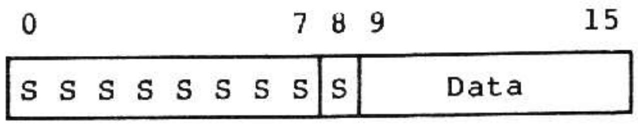
The data field is an 8-bit integer (in two's complement form) with the radix point to the right of bit 7, the least significant bit. Bit 0 indicates the Sign (S) of the data field. The format of the signed integer data byte is:



S = Zero; sign is positive.
S = One; sign is negative.

1.4.5 Sign Extended Integer Byte in Word

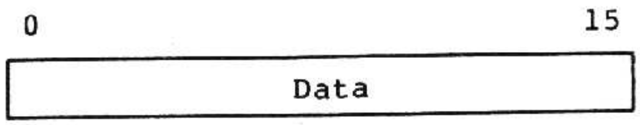
The data field is a 16-bit integer (in two's complement form) with the radix point to the right of bit 15, the least significant bit. The Sign bit (S) is extended from bit 8 through bit 0 of the data word. The format of the sign extended byte in a word is:



S = Zero; sign is positive.
S = One; sign is negative.

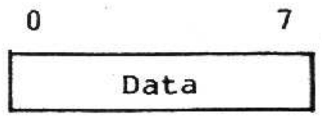
1.4.6 Unsigned Integer Word

The data field is a 16-bit integer. The format of the unsigned integer word is as follows:



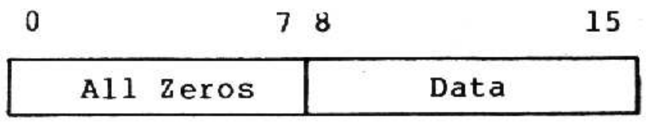
1.4.7 Unsigned Integer Byte

The data field is an 8-bit integer. The format of the unsigned integer byte is as follows:



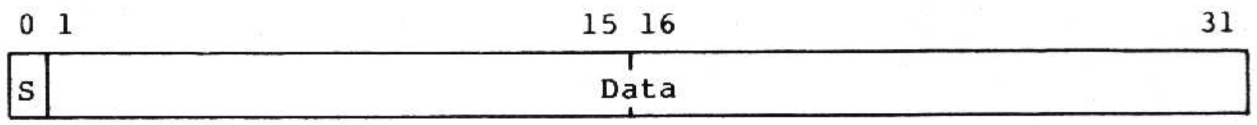
1.4.8 Unsigned Integer Byte in Word

The data field is an 8-bit integer and bits 0 through 7 are zero. The format of the unsigned integer byte in a word is:



1.4.9 Signed Integer Double-Word

The data field is a 32-bit integer (in two's complement form) with the radix point to the right of bit 31, the least significant bit. Bit 0 indicates the Sign (S) of the data field. The format of the signed integer double-word is as follows:



S = Zero; sign is positive.
S = One; sign is negative.

1.5 INSTRUCTION WORD FORMATS

The CPU instruction set is divided into eight categories:

- Double-operand instructions
- Single-operand instructions
- Input/Output instructions
- Short value immediate instructions
- Branch on register instructions
- Branch on indicator instructions
- Shift instructions
- Generic instructions.

For a complete list of Level 6 instructions and their definitions, refer to the Honeywell Level 6 Minicomputer Handbook (order number AS22).

The double-operand and single-operand type instructions use an Address Syllable (AS) field to generate address references. A decode of the AS field usually results in the formulation of an Effective Address (EA), which points to an operand. The address syllable can take one of the following three formats:

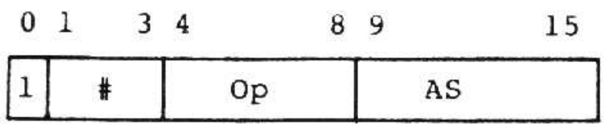
1. Register AS (RAS): The source or destination of the operand is a register (D or B).

2. Immediate Operand (IMO): The operand follows the instruction (special case of MAS, see item 3).

3. Memory AS (MAS): This form specifies a memory location that contains the operand.

1.5.1 Double-Operand Instructions

Double-operand instructions have the following format:



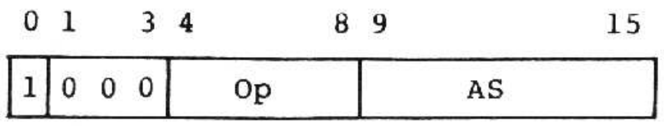
Op = op-code field
# = register number
AS = Address Syllable; refer to subsection 1.6 for the AS format.

Within this group, three types of instruction are available: (1) address register instructions, (2) data register instructions, and (3) mode register instructions. The type of register (D, B, or M) selected by the register number field is a function of the op-code. Depending on whether the address syllable specifies RAS, MAS, or IMO format, these instructions are defined as having the following formats, respectively:

- RR: Register to Register
- RM: Register to Memory
- RI: Register Immediate

1.5.2 Single-Operand Instructions

Single-operand instructions have the following format:



Op = op-code field
AS = Address Syllable; refer to subsection 1.6 for the AS format.

Within the group, three types of instructions are available: (1) control instructions, (2) bit instructions, and (3) modify operand instructions. Depending on whether the address syllable specified RAS, MAS, or IMO format, these instructions are defined as having the following formats, respectively:

- R: Register only
- M: Memory only
- I: Immediate only.

NOTE

Some instructions that modify operands in memory operate in the Read Modify Write (RMW) mode. In this mode, the selected memory cannot be accessed by any other processor in RMW mode until the location is modified by the current RMW. This feature is useful for synchronization in a multiprocessor environment.

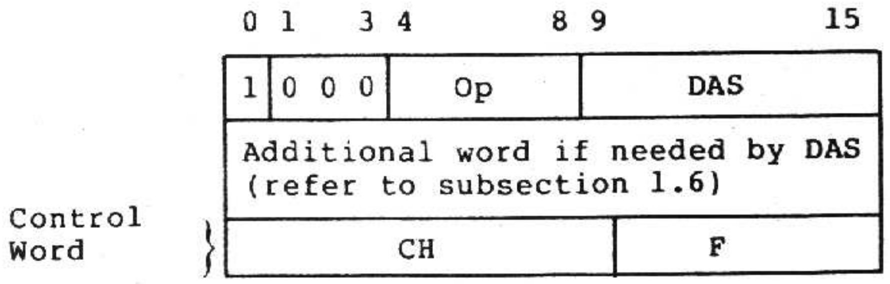
1.5.3 Input/Output Instructions

The I/O instructions are defined by their format as follows:

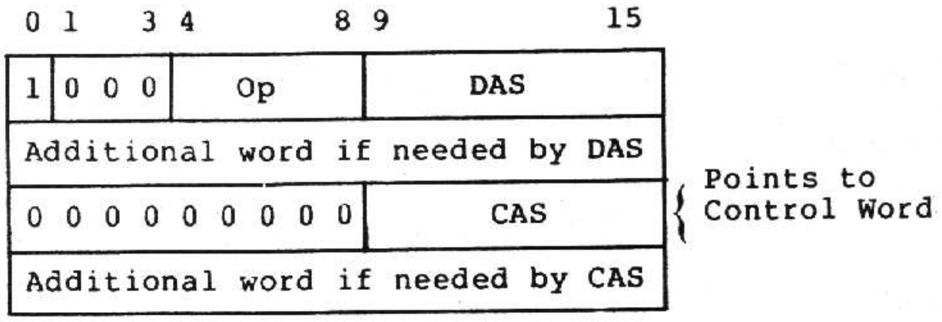
- Data and command I/O instructions
- Address and range output instruction.

1.5.3.1 Data and Command I/O Instructions

These instructions specify two quantities: (1) the data word that is identified by an address syllable identical to the one used for single-operand instructions, and (2) the control word that identifies the external channel (or device) and the function it has to perform. The control word may be imbedded in the procedure as follows:



or it may be nonprocedural, in which case the format is as follows:



Op = op-code field.

DAS = Data Address Syllable; specifies the location from/to which the data are transferred to/from the Megabus (refer to subsection 1.6 for DAS format).

CH = Channel number or the device address.

F = Function code, where:

- If F is even, data are transferred from the controller to the CPU.
- If F is odd, data are transferred from the CPU to the controller.

CAS = Controll Address Syllable; points to control word that contains CH and F (refer to subsection 1.6 for CAS format).

1.5.3.2 Address and Range Output Instructions

This instruction specifies three quantities: (1) the address, which is identified by an address syllable that is identical to the one used for single operand instructions, (2) the control word, which identifies the external channel (or device) and the function it has to perform, and (3) the range which is identified by an address syllable. The control word may be imbedded in the procedure as follows:

0	1 3	4 8	9 15
1	0 0 0	Op	AAS
Additional word if needed by AAS			
CH			F(09)
0 0 0 0 0 0 0 0 0			RAS
Additional word if needed by RAS			

or it may be nonprocedural, in which case the format is as follows:

0	1 3	4 8	9 15
1	0 0 0	Op	AAS
Additional word if needed by AAS			
0 0 0 0 0 0 0 0 0			CAS
Additional Word if needed by CAS			
0 0 0 0 0 0 0 0 0			RAS
Additional word if needed by RAS			

The definitions of the above words are the same as those specified in subsection 1.5.3.1 with the following additions:

AAS = Address Address Syllable; the byte effective address formulation from the AAS is transferred to the Megabus (refer to subsection 1.6 for AAS format).

RAS = Range Address Syllable; specifies the location from which the range is transferred to the Megabus (refer to subsection 1.6 for RAS format).

F = Function code; must specify the function code that is used to load the channel address register; otherwise, the operation is unspecified.

NOTE

All I/O instructions are privileged. If the privilege bit is zero, a trap will result (using trap vector 13) in lieu of execution (refer to Table 2-5).

1.5.4 Short Value Immediate Instructions

Short value immediate instructions have the following format:

0	1 3	4 7	8 15
0	#	Op	V

Op = op-code filed.

= register number; selects one of seven word operand registers.

V = immediate operand value; the value is between -128 and +127, inclusive.

These instructions operate on the word operand (D) registers and perform the following operations:

- Load
- Compare
- Add
- Multiply.

1.5.5 Branch On Register Instructions

Branch on register instructions have the following format:

0	1 3	4 8	9 15
0	#	Op	d

Op = op-code field.

= register number; selects one of seven word operand registers.

d = displacement; the relative address of the branch destination (refer to subsection 1.6.1).

These instructions enable branching on specified conditions of a selected word operand register, such as:

- Equal to zero
- Less than zero
- Increment and test
- Decrement and test.

1.5.6 Branch on Indicator Instructions

Branch on indicator instructions have the following format:

0	1 3	4 8	9 15
0	0 0 0	Op	d

Op = op-code field.

d = displacement; the relative address of the branch destination (refer to subsection 1.6.1).

These instructions enable branching on various indicators, such as:

- Carry
- Equal
- Less than
- Greater than
- I/O bit.

1.5.7 Shift Instructions

The shift instructions have two formats: shift short and shift long.

1.5.7.1 Shift Short Format

The shift short instruction format is as follows:

0	1 3	4 7	8	9 11	12 15
0	#	0000	0	T	d

# = register number; selects one of seven word operand registers.

T = type and direction of the shift.

d = distance (value between 1 and 15, inclusive); if d = 0, substitute the contents of D1 (bits 12 through 15) for distance.

1.5.7.2 Shift Long Format

The shift long instruction format is as follows:

0	1 3	4 7	8	9 10	11 15
0	#	0000	1	T	d

# = register number; selects one of three word operand register pairs.

T = type and direction of the shift.

d = distance (value between 1 and 31, inclusive); if d = 0, substitute the contents of D1 (bits 11 through 15) for distance.

Various types of shifts on single or double operands (two registers linked together) are possible; e.g., closed, open, arithmetic, left, right, etc. For double-operand shifts, the register number must equal 3, 5, or 7; otherwise, the operation is unspecified. The pairing of the registers is as follows:

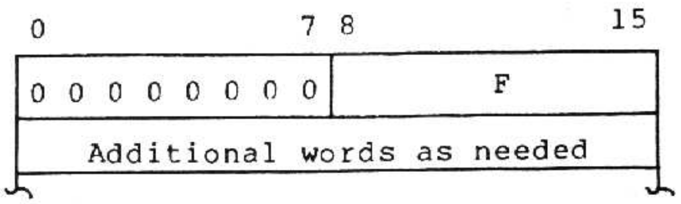
#'	#
2	3
4	5
6	7

#' = the number of the implied register that is linked together with the selected register (#).

Some shift operations modify the Carry (C) or Overflow (OV) indicators. The Carry indicator reflects the state of the last bit shifted out. If the Carry or Overflow indicator is to be modified and the actual shift distance is zero, the Carry or Overflow indicator is clear.

1.5.8 Generic Instructions

The generic instructions have the following format:



F = function code:

Within this group, the following types of instructions are available:

1. Acquire Stack Space
2. Activate Segment Descriptor
3. Breakpoint Trap
4. Control Real-Time Clock
5. Control Watchdog Timer
6. Dequeue On Address
7. Dequeue From Head
8. Halt
9. Load Remote Descriptor Base
10. Load Stack Pointer
11. Memory-To-Memory Move
12. Monitor Call
13. Queue On Head
14. Queue On Tail
15. Reconfigure External Processor
16. Relinquish Stack Space
17. Return From Trap
18. Store Stack Pointer
19. Store Remote Descriptor Base
20. Validate Address, Range, and Access Rights.

1.6 CPU/MEMORY EFFECTIVE ADDRESSING

Address generation depends on the following address types contained in the instruction.

- Displacement
- Address Syllable

In some cases, the contents of the Program Counter (P) are used to generate the effective address; P is assumed to be pointing to the word that contains the displacement in question.

1.6.1 Displacement Address Type

The first type of address definition field is the displacement type used with branch on register and branch on indicator instructions (refer to subsection 1.5.5 and 1.5.6, respectively).

The formulation of the effective address is:

1. If d = 0, the effective address is obtained from the second (and third) word(s) of the instruction.

2. If d = 1, the effective address is obtained by adding to P the Displacement (DSP) contained in the second word of the instruction.

3. If d ≠ 0 or 1, the effective address is obtained by adding the displacement (a two's complement number with a value between -64 and +63, inclusive) to the address of the instruction (P).

1.6.2 Address Syllable

The single-operand and double-operand instructions generate address references through a field called the Address Syllable (AS). The format of the address syllable is:

9 11	12	13 15
m	@	n

m = address modifier.
@ = indirect addressing bit.
n = register number; values between 0 and 7, inclusive.

Figure 1-4 is a representation of the complete address syllable. Table 1-1 lists a set of definitions to facilitate the use of these AS descriptions.

1.6.2.1 Register Address Syllable

The Register Address Syllable (RAS) addresses a register that is the source or destination for the operand. The following subset of Figure 1-4 is defined as RAS:

	N > 0
m	@ = 0
5	B register or D register

	n=0		n>0			
m	@=0	@=1	@=0	@=1		
0	IMA	*IMA	Bn	*Bn		
1	IMA+D1	*IMA+D1	Bn+D1	*Bn+D1		
2	IMA+D2	*IMA+D2	Bn+D2	*Bn+D2		
3	IMA+D3	*IMA+D3	Bn+D3	*Bn+D3		
4	P+DSP	*(P+DSP)	Bn+DSP	*(Bn+DSP)		
5	RFU	RFU	Bn register or Dn register	n=1,2, or 3 Bn + ↓D1	n=4 RFU	n=5,6, or 7 B(n-4) + D1↑
6	RFU	RFU	↓Bn	n=1,2, or 3 Bn + ↓D2	n=4 RFU	n=5,6, or 7 B(n-4) + D2↑
7	IMO	IV+DSP	Bn↑	n=1,2, or 3 Bn + ↓D3	n=4 RFU	n=5,6, or 7 B(n-4) + D3↑

NOTE

Entries in this figure are mnemonics for the various address forms available. These forms are described in subsections 1.6.1 through 1.6.3.

Figure 1-4 Address Syllable

Table 1-1 Address Syllable Notation

NOTATION	DESCRIPTION
DSP	DSP indicates a 16-bit signed displacement that follows the address syllable
*	Indirect operator (≠ @)
+D	Specifies indexing
↑	Auto-increment (B↑ or D↑ indicates postincrementation)
↓	Auto-decrement (↓B or ↓D indicates predecrementation)
IMA	Immediate Address
B	Base register
D	Operand register
P	Program counter; for the purpose of P relative addressing, P points to the word containing the displacement
()	Logical binding
[]	Contents of
+	Addition operator
−	Subtraction operator
x	Multiplication operator
←	Is replaced by
EA	Effective Address
IEA	Intermediate Address
IMO	Immediate Operand
IV	Interrupt Vector (points to Interrupt Save Area of Current Level)

The interpretation of RAS is determined by the op-code. For op-codes on Base (B) registers, RAS is defined as follows:

	n > 0
m	@ = 0
5	B register

Select word operand register n (n = value between 1 and 7, inclusive).

For all other single-operand and double-operand instructions, excluding LAB, LNJ, JMP, ENT, SAVE, and RSTR, RAS is defined as follows:

	n > 0
m	@ = 0
5	D register

Selects word operand register n (n = value between 1 and 7, inclusive).

1.6.2.2 Immediate Operand Address Syllable

The Immediate Operand Address Syllable (IMO) specifies an operand of appropriate size, which immediately follows the instruction. The following subset of Figure 1-4 is defined as IMO:

m	n = 0
	@ = 0
7	IMO

1.6.2.3 Memory Address Syllable

The Memory Address Syllable (MAS) specifies the effective address of a memory location. The MAS can have one of three formats: (1) P relative, (2) Immediate Address, or (3) B relative.

P Relative Format

The P Relative format is as follows:

	n = 0	
m	@ = 0	@ = 1
4	P + DSP	*(P + DSP)

P + DSP: The effective address is formed by adding DSP to the contents of P

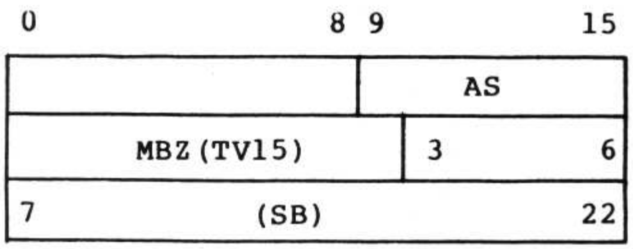
*(P + DSP): The effective address is pointed to by P + DSP.

Immediate Address Format

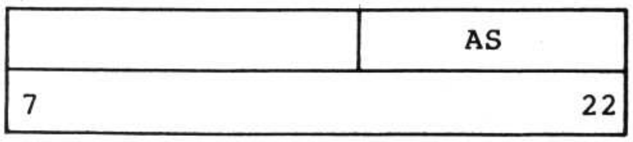
The Immediate Address (IMA) format is as follows:

	n = 0	
m	@ = 0	@ = 1
0	IMA	*IMA
1	IMA + D1	*IMA + D1
2	IMA + D2	*IMA + D2
3	IMA + D3	*IMA + D3

IMA: In LAF, use the two words that follow the address syllable as shown below:



IMA: In SAF, use the 16 bits in the word that follows the address syllable:



IMA: The effective address is contained in the location(s) pointed to by IMA (is the indirect operator).

IMA + D (1, 2, or 3): The effective address is IMA indexed by the appropriate index register (see the following notes).

*IMA + D (1, 2, or 3): The effective address is obtained by adding the contents of the appropriate index register to the contents of the location(s) pointed to by IMA; indirect, post indexing (see the following notes).

NOTES

1. If the operand is larger or smaller than 16 bits, scale the index value before adding.

2. If in the LAF mode, indirection extracts a double-word (IMA and IMA + 1) as follows:

0	11 12	15
MBZ (TV15)	3	6
7		22

B Relative Format

The B Relative format is as follows:

	n > 0		
m	@ = 0	@ = 1	
0	Bn	*Bn	
1	Bn + D1	*Bn + D1	
2	Bn + D2	*Bn + D2	
3	Bn + D3	*Bn + D3	
4	Bn + DSP	*(Bn + DSP)	
5		n = 1,2,3 Bn + ↓D1	n = 5,6,7 B(n-4) + D1↑
6	Bn	n = 1,2,3 Bn + ↓D2	n = 5,6,7 B(n-4) + D2↑
7	Bn	n = 1,2,3 Bn + ↓D3	n = 5,6,7, B(n-4) + D3↑

Bn: The effective address is contained in base register n (n = value between 1 and 7, inclusive).

*Bn: The effective address is contained in the memory location pointed to by base register n (n = value between 1 and 7, inclusive).

Bn + D1, D2, or D3: The effective address is obtained by adding the contents of the appropriate index register to the contents of base register n (n = value between 1 and 7, inclusive). Refer to the following note.

*Bn + D1, D2, or D3: The effective address is obtained by adding the contents of the appropriate index register to the contents of the memory location(s) pointed to by base register n (n = value between 1 and 7, inclusive). Refer to the following note.

NOTE

If the operand is larger or smaller than 16 bits, scale the index value before adding.

Bn + DSP: The effective address is obtained by adding DSP to the contents of base register n (n = value between 1 and 7, inclusive).

*(Bn + DSP): The effective address is contained in the location(s) pointed to by Bn + DSP (n = value between 1 and 7, inclusive).

↓Bn: The effective address is contained in base register n after it is decremented by the operand size in words (n = value between 1 and 7, inclusive).

Bn↑: The effective address is contained in base register n (n = value between 1 and 7, inclusive). The base register is incremented by the operand size in words after effective address formation and prior to instruction execution.

Bn + Dx: The effective address is obtained by adding the contents of the appropriate (m - 4) index register (after it is decremented by 1) to the contents of the selected (n - 4) base register. If the operand is larger or smaller than 16 bits, the index value is scaled to operand size before addition.

B (n-4) + Dx: The effective address is obtained by adding the contents of the appropriate (m - 4) index register to the contents of the selected (n - 4) base register. After effective address formation and prior to execution of the op-code, the selected index register is incremented by 1. If the operand is larger or smaller than 16 bits, the index value is scaled to operand size before addition.

1.6.3 Indexing

Many address syllable forms specify indexing. During effective address generation, one of three index registers (D1, D2, or D3) is algebraically added as the last step, after any indirection, to the base address. The index value is treated as a signed integer data word.

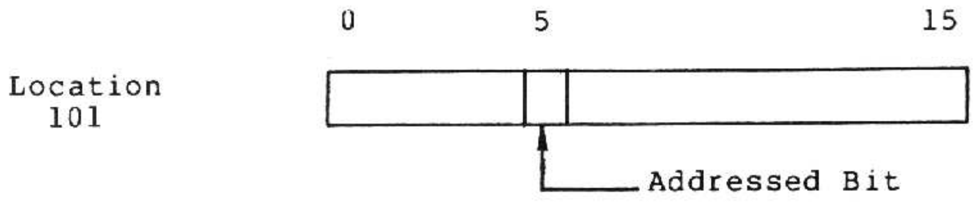
While indexing, the hardware automatically aligns the index value to correspond to the size of the item being referenced. The size of this item is determined by the op-code type (bit, byte, digit, word, double-word, or quadruple-word).

1.6.3.1 Bit Addressing

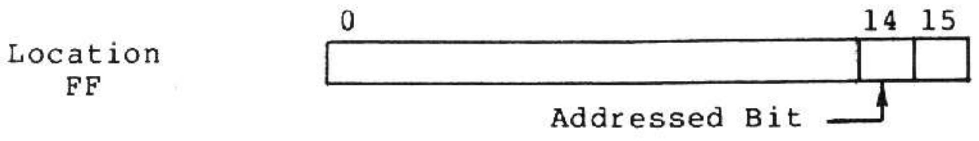
The index value for bit instructions (LB, LBF, LBT, LBC, and LBS), where the address syllable calls for indexing (IMA + Dx, Bn + Dx, *Bn + Dx, etc.), is aligned to denumerate bits in the array, with bit 0 being the leftmost bit in word 0 of the array. As an example, consider the following instruction.

LB, B1 + D1

If B1 = 100 and D1 = +21, then the addressed bit is bit 15 in location 101.



If B1 = 100 and D1 = -2, then the addressed bit is bit 14 in location FF.

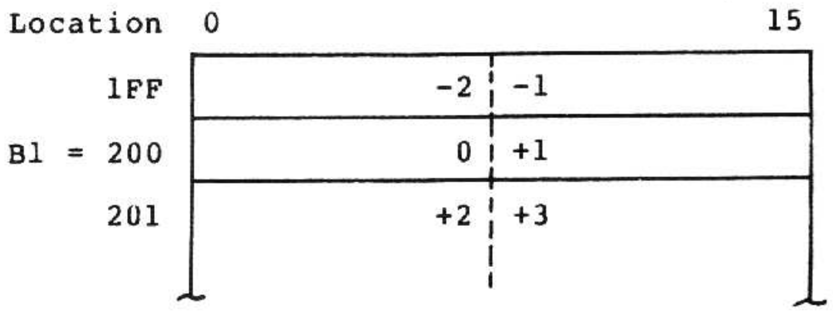


1.6.3.2 Byte Addressing

The index value for byte instructions (LDH, STH, CMH, etc.), where the address syllable calls for indexing, is aligned to denumerate bytes in the array, with byte 0 being the leftmost byte in word 0 of the array. As an example, considering the following instruction:

LDH, D4, B1 + D2

If B1 = 200 and D1 = ±n, then the addressed byte is as shown below:

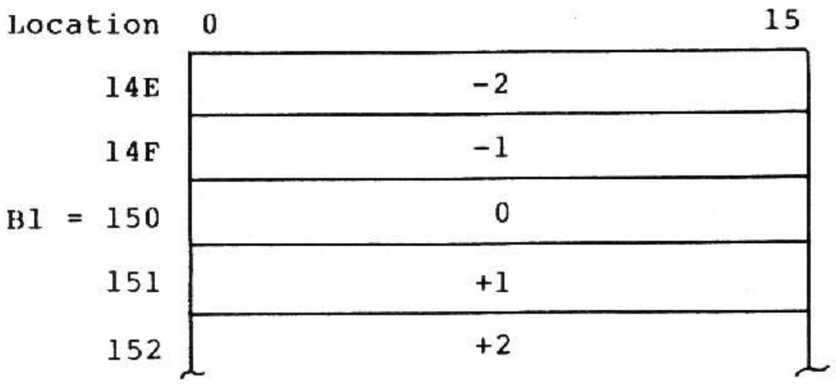


1.6.3.3 Word Addressing

The index value for word instructions (LDR, ADD, OR, etc.), where the address syllable calls for indexing, denumerates words in the array, with word 0 being the leftmost word in the array. As an example, consider the following instruction.

ADD, D3, B1 + D2

If B1 = 150 and D2 = $\pm$n, then the addressed word is as follows:

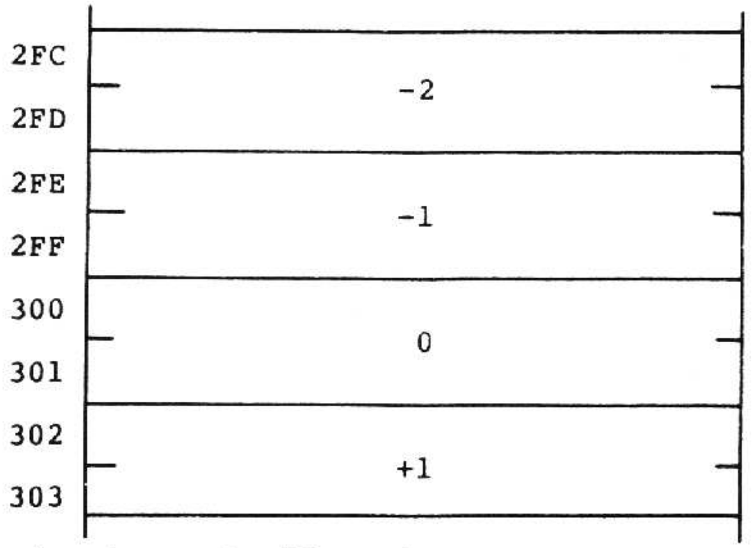


1.6.3.4 Double-Word Addressing

The index value for double-word operands, where the address syllable calls for indexing, denumerates double-words (32 bits) in the array, with the double-word zero (0) being the leftmost double-word. The processor assumes double-word operands for the following: (1) LDI, SDI, and AID instructions, and (2) scientific instructions (except branch instructions) if the memory operand length stored in M4 equals 2, and (3) all base register instrucitons (LDB, CMN, STB, CMB, and SWB) if the processor is in LAF mode. As an illustrative example, consider the following instruction.

LDI, (D6, D7), B4 + D3

If B4 = 300 and D3 = $\pm$n, then the address operand is as follows:



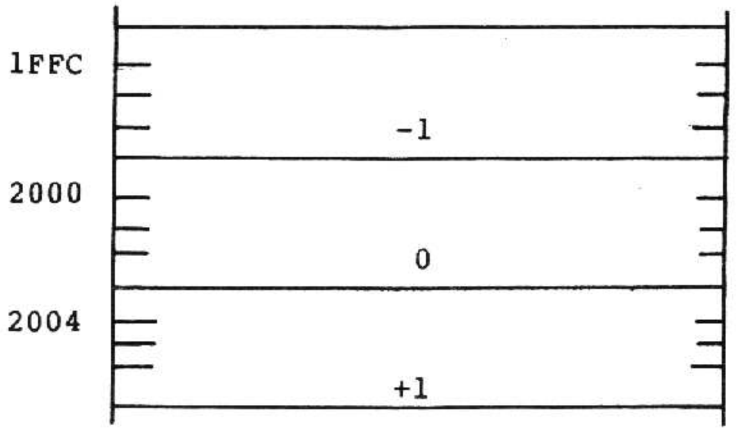
1.6.3.5 Quadruple-Word Addressing

The index value for scientific instructions (except branch instructions), where the memory operand length equals 4 (stored in M4) and the address syllable calls for indexing, assumes an array of quadruple words.

Consider the following example:

FAD, S1, B5 + D1

If B5 = 2000 and D1 = $\pm$n, then the addressed operand is as follows:



1.7 MEMORY ADDRESS BOUNDARIES

When the CPU initiates a memory cycle that addresses non-existent memory. the processor either traps (using trap vector 15, see Table 2-5) or causes undefined results. These boundaries are of interest in defining nonexistent memory:

1. Addresses below zero

2. Addresses beyond the last module on-line at a given installation

3. Addresses greater than 1,024 K-words (i.e., main memory equals 1M-words).

If main memory is less than 1M-words, violation of any of the above three address boundaries causes a trap. If main memory is 1M-words, then trap conditions caused by incrementing the program counter, a hidden register, or a base register may not be detected and cause undefined results.

It should be noted that when the memory management unit (refer to subsection 1.2) is configured into the system, large virtual addresses may roam freely within the CPU and are not subjected to the scrutiny previously described until they are mapped and deposited onto the Megabus. The memory management unit may trap an address for other reasons, for example:

- Violation of protection (trap vector 14)
- Attempt to reference an invalid segment (trap vector 15).