Section 1

INTRODUCTION

AP-101S WITH SHUTTLE INSTRUCTION SET

The AP-101S is a high-speed general-purpose computer intended primarily for real-time applications such as guidance, navigation, control, and data processing. The AP-101S is a member of the advanced System/4 Pi family of digital computers, and is software compatible with AP-101C/M, described in IBM No. 6246156B, 30 Jan. 1979. This family shares and is unified by extensive design experience, proven technology base, and common manufacturing processes.

This Principles of Operation manual provides a direct comprehensive description of the system structure; the arithmetic, logical, branching, and status switching; and the interruption system. This publication defines and describes features common to all AP-101 computers. These features are the basis for IBM-developed support software and are compatible with compiler development efforts now in process.

Execution times and nonstandard features and functions are described in separate documents. This is because aerospace computers characteristically include user defined features such as unique input/output channels, and special discretes. These will be incorporated into the AP-101S as pluggable options. Furthermore, the AP-101S is microprogrammed and is designed to permit incorporation of additional instructions and operations without redesign and requalification. Such extensions are also described separately.

Note: This document is also applicable to the AP101S/G, the ground version of the AP101S computer.

Section 2

AP-101S STRUCTURE

SHUTTLE INSTRUCTION SET

The AP-101S system structure encompasses the functional operation of main storage, the central processing unit (CPU), and program-controlled I/O facilities. The overall definition is open ended and includes all the basic facilities necessary to accommodate additional specialized and/or application-dependent I/O channels and features.

The modular AP-101S system structure can support configuration alternatives ranging from a self-contained single processor to a full symmetrical shared-storage multiprocessing system.

MAIN STORAGE

The functional operation of main storage is unrelated to the physical width of the information paths or cycle time.

INFORMATION FORMATS

The system transmits information between main storage and the CPU in units of 16 bits, or in integer multiple of 16 bits. Each 16-bit unit of information is called a halfword. ~~A parity bit and a~~ storage protection bits are also associated with each halfword, but later references in this manual to the size of data fields exclude these bits.

Halfwords may be handled separately or in pairs. A fullword is a group of two consecutive halfwords. Both halfword and fullword instructions and operands are used. Their location is always specified by the address of the leftmost halfword. The instruction length is designated implicitly in every instruction; the operand length is also implicit.

Within any instruction and operand format, the bits making up the format are consecutively numbered from left to right, starting with the number 0, as shown in Figure 2-1.

ADDRESSING

Halfword locations in storage are consecutively numbered starting with 0. Each number is considered the address of the corresponding halfword. The addressing technique uses a 19-bit binary address to accommodate a maximum of 2^{19} halfword addresses. This set of main storage addresses includes some locations reserved for special purposes, such as program status words; consequently, these special locations should not be used for any purpose not implicitly defined.

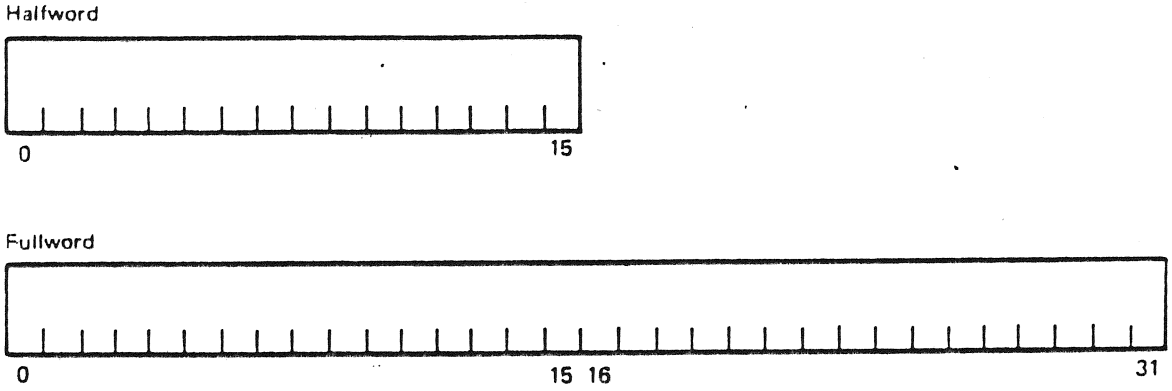


Figure 2-1. Instruction and Operand Bit Numbering

INFORMATION POSITIONING

Fullword operands must be located in main storage on even halfword boundaries. That is, the least significant bit of the operand address, when expressed in binary, must always be zero. Fullword instructions may begin at any address.

CENTRAL PROCESSING UNIT

The central processing unit (CPU) contains facilities for addressing main storage, for fetching or storing information, for arithmetic and logical processing of data, for sequencing instructions in the desired order, and for initiating the communication between storage and external devices.

The control section guides the CPU through the functions necessary to execute the program.

PROGRAM ADDRESSABLE REGISTERS

Two sets of eight fixed-point general registers and one set of eight floating-point registers are under explicit program control. The contents of one or more of these registers (32 bits) participate in most CPU operations.

Conceptually, an additional doubleword status register, called the program status word (PSW), is the focal point for machine status. The contents of the PSW are updated during each instruction. Consequently, the PSW reflects current machine status following every instruction. The PSW participates implicitly in status switching, branching operations, and address calculations.

In addition to the PSW and the general and floating-point registers, the CPU also contains working registers used for storage addressing, storage buffering, shift and iteration counting, and operand storage. These registers are of no direct concern to the programmer and are not described herein.

The contents of the PSW specify which of the two sets of general registers is in current use. Only the contents of the selected general register set can participate in arithmetic operations and the contents of unselected sets of general registers can not be altered by a program. An alternate set of general registers can be selected by changing the PSW. Only one set of the fixed-point, general-purpose registers and the floating-point registers are available to the program at any one time.

General register contents can be used interchangeably as operands for arithmetic, logical, and shifting operations, or as base and index registers for relative addressing. Each set of general registers is numbered from 0 through 7 and is addressed as shown in Figure 2-2.

General Register Number	Register Function		
	Operand	Base	Index
0	000	00	None
1	001	01	001
2	010	10	010
3	011	11 or None	011
4	100		100
5	101		101
6	110		110
7	111		111

Figure 2-2. General Register Addresses

Note that general registers 4 through 7 cannot contain base addresses and that general register 0 cannot contain an index.

AP

For some operations, an even/odd pair of general registers are linked to form a 64-bit doubleword register. The most significant half of a doubleword operand is contained in the even-numbered register; the least significant half of the doubleword in the next higher odd-numbered register. Doubleword operands are addressed by specifying the even numbered address of the register containing the most significant portion of the operand.

For addressing data, general registers 0-3 can be augmented by 4 bit Data Sector Extension (DSE) registers or by the DSR in the PSW to address beyond 16 bit capabilities. There are 16 DSE's, one for each of the 8 general purpose registers in each of the two sets of general registers. This feature shall not be used by a program of less than 128K full words.

FIXED-POINT DATA REPRESENTATION

Data representation is fractional, with negative numbers represented in two's complement form. A halfword operand is 15 bits plus sign; a fullword operand is 31 bits plus sign, as shown in Figure 2-3.

In fractional data representation, the binary point is immediately to the right of the sign.

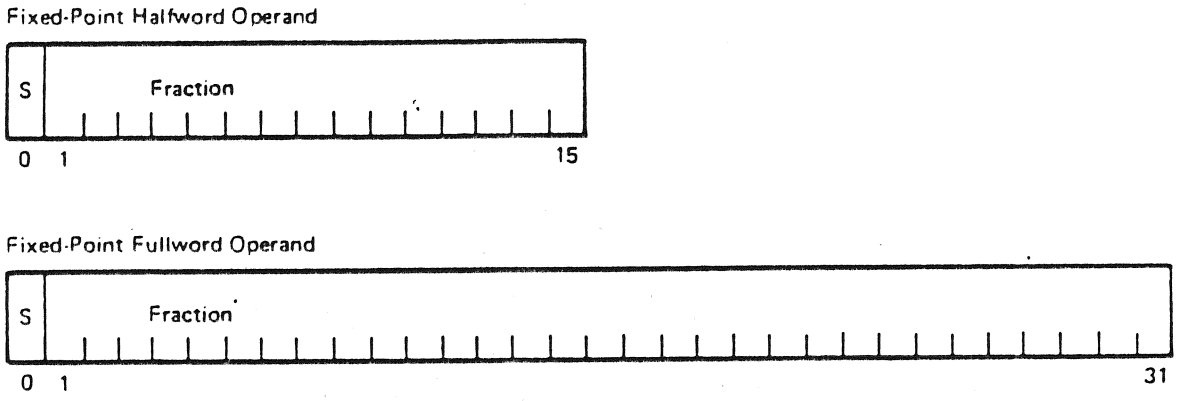


Figure 2-3. Fixed-Point Operand Formats

INSTRUCTION FORMATS

The length of an instruction format can be either one or two halfwords. Long format instructions provide maximum range and extended flexibility for addressing storage operands. Short instructions are used to (1) specify register-to-register operations, and (2) specify storage operands in cases where a small displacement is sufficient and complete address modification capability is not required.

Instruction formats overlap. Programs are written so that in many instances any given operation can be coded using either a halfword or a fullword instruction. In such cases, maximum use of halfword instructions results in increased storage efficiency and performance.

The three basic instruction formats are as shown in Figure 2-4. Halfword instructions are automatically selected by the assembler unless otherwise specified by the programmer.

4.1 GENERAL SYSTEM OPERATION

The AP-101S was formed by the integration of a redesigned B1-B AP-101F processor and a repackaged Input/Output Processor (IOP) from the existing Shuttle computer. Redesign and repackaging permits both of these elements to be housed in a single structure. Figure 2 on page 4 shows the AP-101S Block Diagram.

The elements utilized from the AP-101F are the CPU, MMU (Memory Management Unit), and Interrupt sections. The microcode has been modified so existing shuttle software can be used on the AP-101S. The Timing page, SDI (Software Development Interface) page and the SIB bus have been eliminated. The unused circuitry in the MMU has been removed to permit integration of the timing and SDI functions into the MMU.

The IOP has been repackaged using medium scale integration to reduce the number of pages from fourteen to seven. The IOP has maintained the same timing as the original processor.

All of the pages use Modular Computer System (MCS) page technology. The repackaging allows the AP-101S to be housed in a single box.

The CPU performs the functions of computation, storage and communication of data for the Shuttle Orbiter. The CPU executes instructions from main store. Main store is controlled by the MMU, which handles all memory access requests from the CPU and IOP.

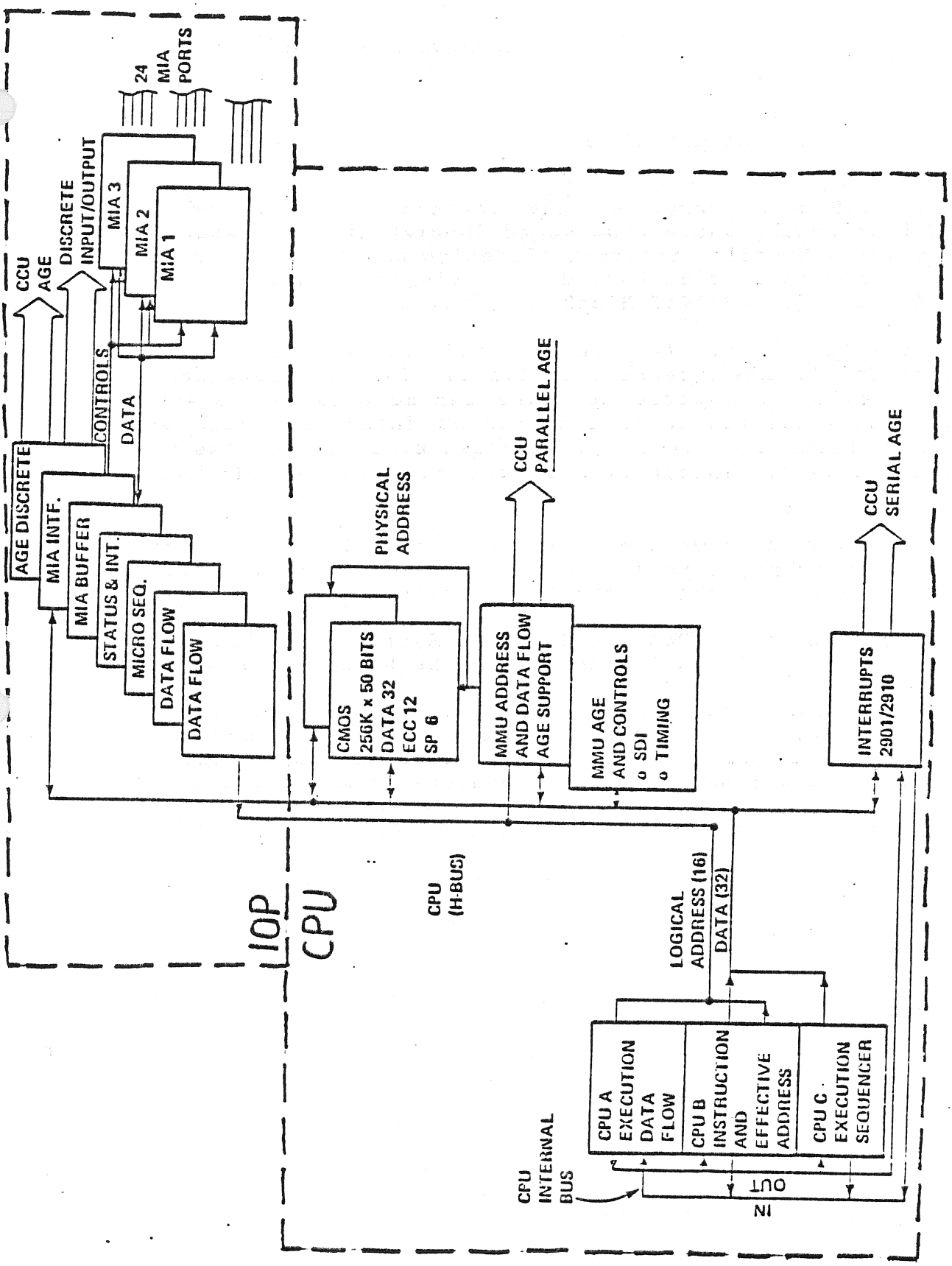


Figure 2. AP-101S Block Diagram

The IOP functions as a programmable, time-shared processor that transmits and receives Shuttle Orbiter subsystems data under control of the CPU.

The CPU communicates with the IOP by means of Program Controlled Input/Output (PCIO) instructions. PCIO transmissions involve a command word and data from either the CPU or IOP.

The Shuttle Orbiter subsystems are connected to the IOP by 24 serial, 1-MHz data buses. Data bus-to-IOP interface is accomplished by 24 Multiplexer Interface Adapters (MIAs) located in the IOP. The MIAs perform such functions as parallel/serial conversion, Manchester encode and decode, parity generation and detection, and bit count detection. The IOP handles the processing required to service the 24 data buses.

The 24 data buses each have a Bus Control Element (BCE). The BCEs are given instructions by the Master Sequence Controller (MSC) on how to handle data. The MSC executes instructions from main store as directed by PCIO instructions from the CPU. The MSC/BCE instructions and data are fetched from main store through Direct Memory Access requests. The MSC has a set of programmable registers in Local Store. These registers include a PCIO register, index register and program counter.

The BCEs execute programs from main store specified by MSC instructions. Each BCE also has a set of programmable registers in Local Store and can read or write I/O data into main memory via Direct Memory Access (DMA). Included in the registers is an indicator register which contains one bit for each BCE. This bit is set and reset by a BCE to communicate with the MSC.

Each BCE is sequenced by a timing 'wheel' which allows one microinstruction from each BCE to be executed at a time. The MSC is also in this timing sequence, but it gets eight slots in a complete turn of the 'wheel' while each BCE gets only one. One MSC microinstruction is executed after three BCE microinstructions. Some MSC and BCE instructions may take more than one rotation of the 'wheel' to be executed.

The Interrupt page contains a processor to handle interrupts. The interrupt processor prioritizes, masks, categorizes and performs any other processing that is necessary before giving information to the CPU. A one byte word is generated to inform the CPU into which category the interrupt falls. Additional information allows the CPU to formulate a six bit address for a PSW swap and begin processing the interrupt.

Each of the three major components of a GPC (CPU, IOP, and Interrupt Page) is controlled by independent microcode. The CPU microarchitecture is described in detail in "Microcontrol Implementation For CPU" on page 113, the Interrupt microarchitecture is described in "Microcontrol Implementation For INT" on page 191, and the IOP

microarchitecture is described in "Microcontrol Implementation For IOP" on page 211.

4.2 AP-101S CENTRAL PROCESSING UNIT

The AP-101S central processor unit is optimized for both MMP and MIL-STD-1750A Notice 2 architectures and is comprised of these functional units:

- Instruction Unit (I-unit)
- Effective Address Unit (EA-unit)
- Execution Unit (EX-unit)
- Fractional Data Flow
- Exponential Data Flow
- Sequencer

These units are organized to execute instructions in a pipeline fashion designed to provide results at a rate of one per machine cycle (250 ns) when operating on simple instructions (with the pipe full). The pipeline is shown in Figure 3 on page 7.

The Instruction unit is responsible for prefetching instructions. It provides a logical instruction address to the Memory Management Unit, which then translates this to a physical address before fetching the instruction. The EA-unit decodes the instruction to determine what type of addressing the instruction specifies, and uses its data flow to calculate (if necessary) the effective (logical) address of the operand. This logical address is translated to a physical address in the MMU, and the operand is fetched. The EA-unit provides the operand and decoded instruction to the Execution unit and selects the general registers specified by the instruction.

The EX-unit performs the actual execution of the instruction via microprogramming (i.e., microcode provides the signals that control the data flow through the hardware). Each macroinstruction corresponds to one or more microinstructions. At the end of a microcode routine which implements a macroinstruction, a 1:256-way branch in the microcode is executed in order to access the section of microcode required for execution of the following macroinstruction.

The CPU machine cycle is 250 ns and is the time required to read, compute, and write the result of a simple register to register operation such as add (RA = RA + RB). Each pipeline operation is completed in one machine cycle and data can be passed from one stage of the pipe to the next at this rate when the EX-unit is operating at its maximum rate. Three additional cycle times for the EX-unit are

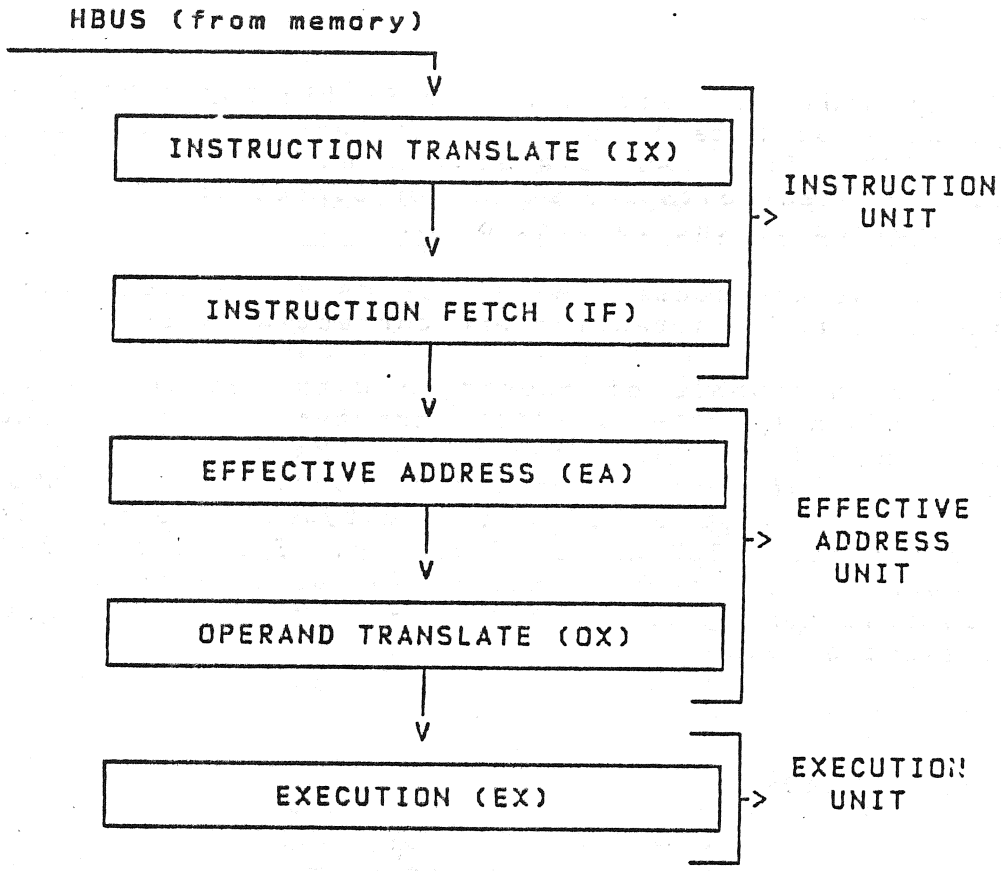


Figure 3. CPU Pipeline

provided to speed up the execution of multicycle instructions: 125 ns and 150 ns are for microcoded operations which do not require a full machine cycle to execute, and 100 ns is used for high-speed iterations typical in operations such as multiplication, division, and shifting.

Synchronization of the pipeline is accomplished by means of the ENDOP command which is issued at the end of each macroinstruction by the microprogram. The ENDOP command signals each stage of the pipeline to output its results (pass them on to the next stage) and to begin working on its new input (the output from the previous stage) at the beginning of the next machine cycle. When the EX-unit is operating on simple instructions, the ENDOP command may be issued every 250 ns, one machine cycle. When the EX-unit requires more than 250 ns for the execution of an instruction, the operation of all other stages in the pipeline is suspended (no ENDOP is issued) except for the prefetching of instructions by the I-unit (which continues independently until the 16 x 16-bit instruction file is full). When the EX unit has completed its operation the microprogram issues an ENDOP and all stages of the pipeline restart at the beginning of the following machine cycle. The ENDOP signal also signifies the end of a microcode routine, causing the EX-unit to branch (1:256-way branch) to the start of a new routine based on the next macroinstruction.

4.3 MAIN STORAGE

The AP-101S contains two battery-backed Static RAM CMOS pages, each containing 128K X 32 bits plus store protect bits and Error Correction Code (ECC) bits. Associated with each main memory halfword are three store protect bits and six Error Correction Code (ECC) bits which are determined by the 16 data bits.

The CMOS memory has an access time of 250 ns and a cycle time of 250 ns. This includes error detection and correction (EDC).

The AP-101S is also capable of operation with dynamic memory pages of the type found in the B1-B AP-101F computer. A signal indicating the type of memory in use is generated on the memory page, and this signal is used to configure the interface portion of the MMU. Both memory pages in use must be of the same type. The dynamic memory configuration provides 128K words of memory. Except for the difference in memory size, the type of memory in use is transparent to the software. Dynamic memory is not battery-backed and will not retain data in the event of power loss.

4.4 INFORMATION FORMATS

The system transmits information between main storage and the CPU in units of 16 bits, or in integer multiples of 16 bits. Each 16-bit unit of information is called a halfword. Six error correction bits and three voted storage protection bits are also associated with each halfword for the AP-101S, but later references in this workbook to the size of the data fields exclude these bits.

Halfwords may be handled separately or in pairs. A fullword is a group of two consecutive halfwords. Both halfword and fullword instructions are used. Their location is always specified by the address of the most significant halfword. The instruction length is designated implicitly in every instruction. The operand length is also implicit.

Within any instruction and operand format, the bits making up the format are consecutively numbered from left to right, starting with the number zero, as shown in Figure 4 on page 9.

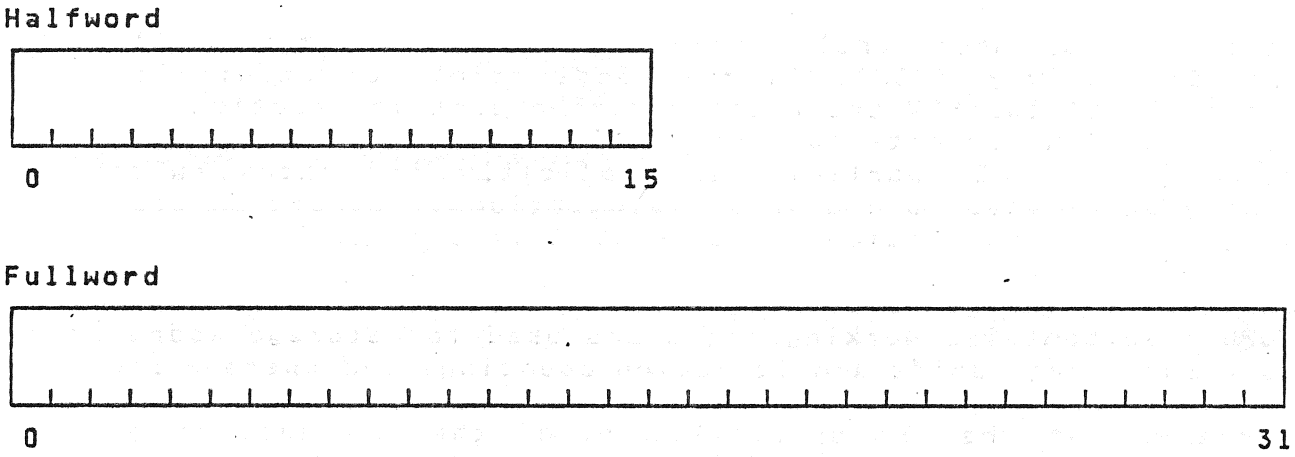


Figure 4. Instruction and Operand Bit Numbering

4.5 ADDRESSING

Halfword locations in storage are consecutively numbered starting with zero. Each number is considered the address of the corresponding halfword. The addressing technique uses a 19-bit binary address to accommodate a maximum of 512K halfword addresses. This set of main storage addresses includes some locations reserved for special purposes, such as program status words. Consequently, these special locations should not be used for any purpose not explicitly defined.

4.6 INFORMATION POSITIONING

Unlike previous versions of the AP-101 computer, the AP-101S does not require either fullword instructions or fullword/doubleword operands to be located in main storage on even boundaries.

4.7 PROGRAM ADDRESSABLE REGISTERS

Two sets of eight fixed-point general registers and one set of eight floating-point registers are under explicit program control. The contents of one or more of these registers (32 bits each) participate in most CPU operations. Associated with each of the general purpose registers is a 4-bit addressing extension register (Data Sector Ex-

tension or DSE), the use of which is described below in Extended Addressing.

Conceptually, an additional doubleword status register, called the Program Status Word (PSW), is the focal point for machine status. The contents of the PSW are updated during each instruction. Consequently, the PSW reflects current machine status following every instruction. The PSW participates implicitly in status switching, branching operations, and address calculations. Condition codes resulting from an instruction are also part of the PSW.

In addition to the PSW and the general and floating-point registers, the CPU also contains working registers used for storage addressing, storage buffering, shift and iteration counting, and operand storage.

The contents of the PSW specify which of the two sets of general registers is in current use. Only the contents of the selected general register set can participate in arithmetic operations and the contents of unselected sets of general registers cannot be altered by a program. An alternate set of general registers can be selected by changing the PSW. Only one set of the fixed point, general purpose registers and the floating-point registers are available to the program at any one time.

General register contents can be used interchangeably as operands for arithmetic, logical and shifting operations, or as base and index registers for relative addressing. Each of the general registers is numbered from 0 through 7 and is addressed as shown in Figure 5.

General Register Number	Register Function		
	Operand	Base	Index
0	000	00	Not Used
1	001	01	001
2	010	10	010
3	011	11 or none*	011
4	100		100
5	101		101
6	110		110
7	111		111

*11 = Register 3 for SRS; none for RS

Figure 5. General Register Addresses

Note that general registers 4 through 7 cannot contain base addresses and that general register 0 cannot contain an index.

For addressing data, general registers 0-3 can be augmented by 4-bit Data Sector Extension (DSE) registers or by the DSR in the PSW to address beyond 16-bit capabilities. There are 16 DSEs, one for each of the eight general-purpose registers in each of the two sets of general registers.

For some operations, a pair of general registers is linked to form a 64-bit doubleword register. The most significant half of a doubleword operand is contained in the specified register; the least significant half of the doubleword is in the next higher-numbered register (determined by modulo 8 addition of one (1) to the specified register). Note: If Reg 7 is specified, the least significant half of the double word operand is contained in Reg. 0.

One set of eight 32-bit floating-point registers is provided and these registers are separate and distinct from the general-purpose registers.

4.8 DATA REPRESENTATION

Fixed-point data representation is both integer and fractional, with negative numbers represented in twos complement form. A halfword operand is 15 bits plus sign, a fullword operand is 31 bits plus sign, and a doubleword operand is 63 bits plus sign, as shown in Figure 6 on page 12. In fractional data representation, the binary point is immediately to the right of the sign. In integer arithmetic, the binary point is to the right of bit 15.

Unless otherwise stated, fixed-point arithmetic operations assume a fractional data type.

Floating point data occupies either a fullword format or a doubleword format. These formats differ between the MMP and 1750A architectures, as depicted in Figure 7 on page 13 and Figure 8 on page 14.

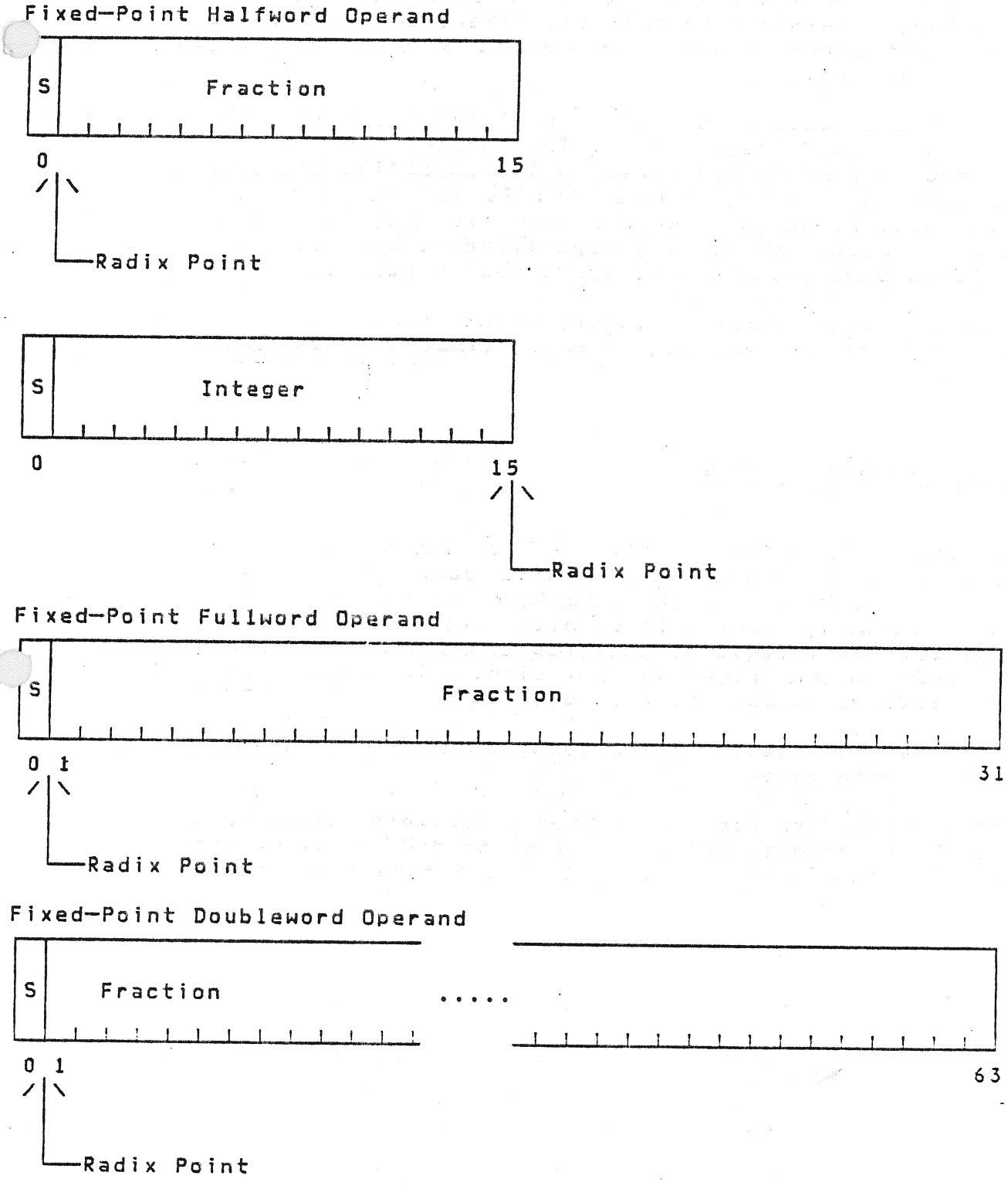
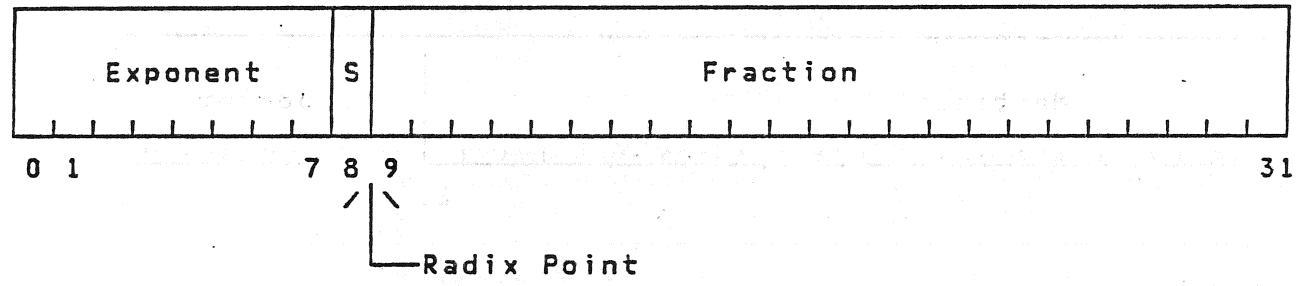
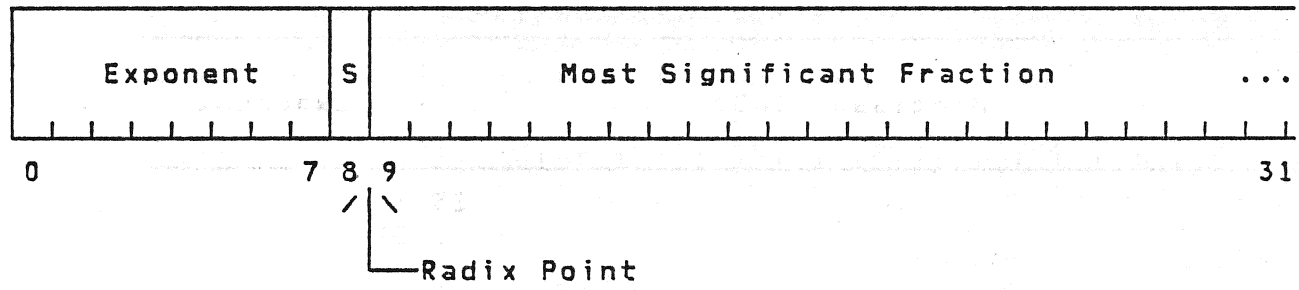


Figure 6. Fixed-Point Operand Formats

Short Floating-Point Number (MMP Architecture)



Long Floating-Point Number (MMP Architecture)



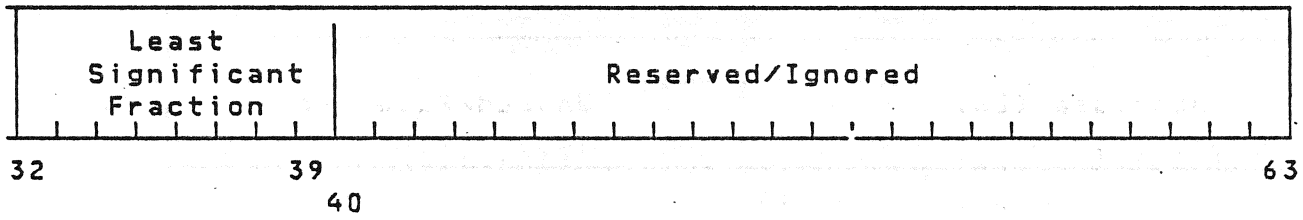


Figure 7. Floating-Point Operand Formats (MMP Architecture)

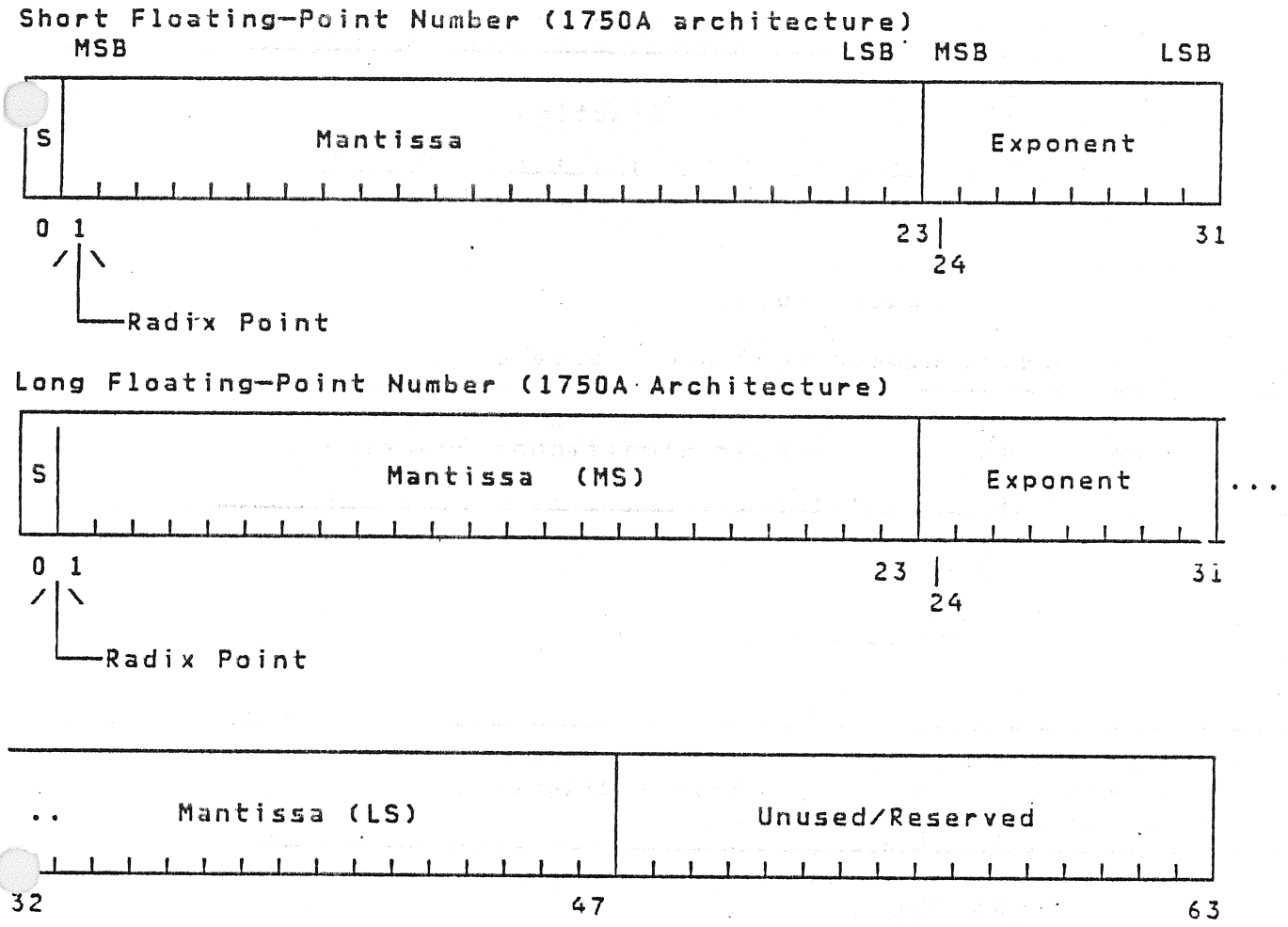


Figure 8. Floating-Point Operand Formats (1750 Architecture)

4.9 INSTRUCTION FORMATS

The length of an instruction format can be either one or two 16-bit words. In contrast with 16-bit (halfword) instructions, 32-bit (fullword) instructions provide increased addressing, permit the specification of additional address modification, and make possible the designation of special conditions to test for in Jump instructions. Halfword instructions are used to (1) specify register-to-register operations, and (2) specify storage operands in cases where a small displacement is sufficient and complete address modification is not required.

Instruction formats overlap. Programs are written so that, in many instances, any given operation can be coded using either a halfword or a fullword instruction. In such cases, maximum use of halfword

11.1 BACKPANEL FUNCTIONAL DESCRIPTION

The Backpanel provides the means of connecting all the pages in the AP101S Computer to each other and the outside world. It is a Multi-layer Interconnection Board (MIB) with connectors for each page, the Power Converters, and the Input/Output (I/O) Wiring Harness. Figure 130 on page 256 gives a side view of the AP101S Computer showing the Backpanel and which page is in each connector.

11.1.1.1 Backpanel Layout

There are 23 slots or places for connectors in the backpanel as de-fined below. The input voltages available to each slot are also listed.

SLOT	DESCRIPTION	INPUT VOLTAGES
A01	I/O Harness	+5 MEMORY
A02	AD Page (Age and Discretes)	+5V,+12V,+5 MEMORY
A03	MIA Page (Manchester Interface Adapter)	+5V,+12V,-12V
A04	MIA Page (Manchester Interface Adapter)	+5V,+12V,-12V
A05	MIA Page (Manchester Interface Adapter)	+5V,+12V,-12V
A06	MC Page (Master Sequencer Controller)	+5V
A07	IB Page (I/O Buffer)	+5V
A08	SI Page (Status and Interrupt	+5V
A09	FT Page (Flow Top)	+5V
A10	Spare Slot	+5V
A11	Spare Slot	+5V
A12	FB Page (Flow Bottom)	+5V
A13	IM Page (Interface and MIA Control)	+5V
A14	CC Page (CPU 3)	+5V
A15	CB Page (CPU 2)	+5V
A16	CA Page (CPU 1)	+5V
A17	IN Page (Interrupt)	+5V
A18	MB Page (MMU 2)	+5V
A19	MA Page (MMU 1)	+5V
A20	CMOS Memory Page	+5 MEMORY
A21	CMOS Memory Page	+5 MEMORY
A22	+5 Volt Converter (Power Supply)	28 VDC
A23	12 Volt Converter (Power Supply)	28 VDC

Figure 129. Backpanel Slot Input Voltages

All the connectors have 296 pins except the I/O Harness connector (A01) which has 300 pins and the +5 Volt and 12 Volt Power Converters (A22 and A23) which have 125 pins.

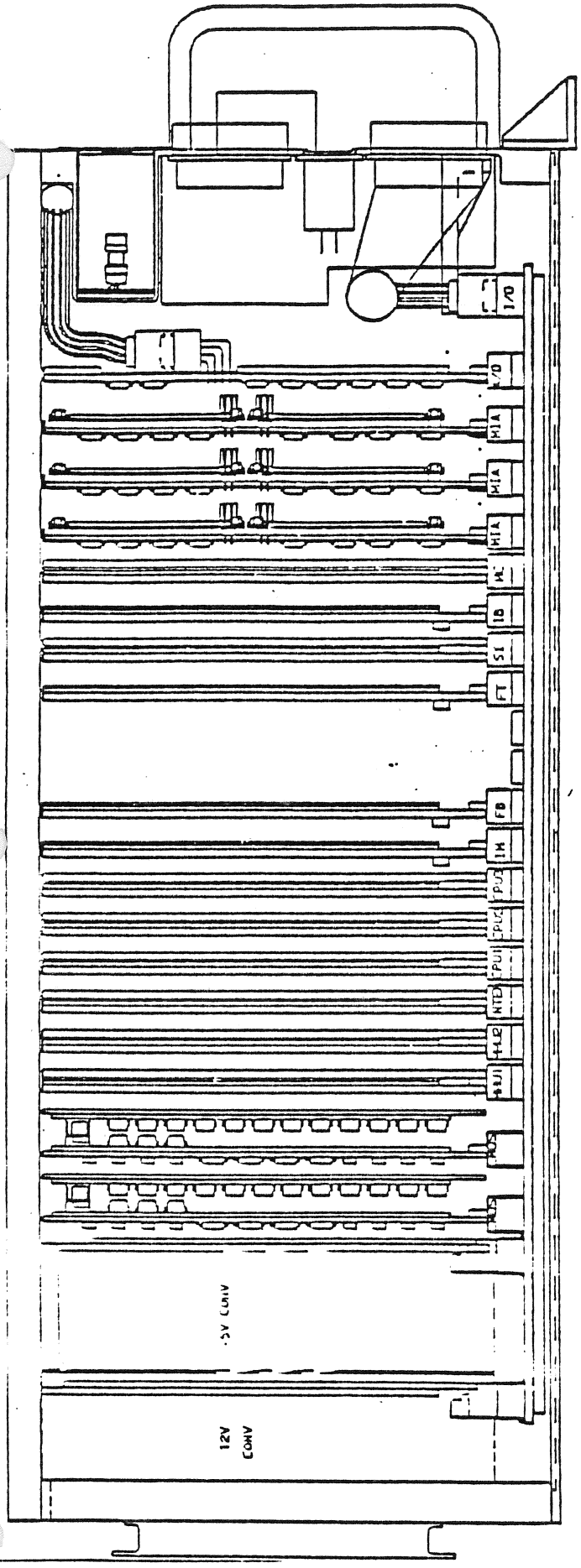


Figure 130. AP101S Side View

11.1.1.2 Backpanel Stackup

The Backpanel consists of 23 layers as shown in Figure 131 on page 257. These include the "0" top and "0" bottom, eleven signal layers, and various voltage and ground layers. One signal layer is divided to provide straight runs for the MIA channels without any interference from other signals. This divider separates A01 through A05 from the other backpanel slots. Some of the voltage layers are also divided.

LAYER NUMBER	COPPER THICKNESS	DESCRIPTION
1	1 OZ	"0" TOP, FOIL
2	2 OZ	28V, +5V
3	2 OZ	SIG 1
4	1 OZ	SIG 2
5	1 OZ	+12V, 28V RETN
6	1 OZ	-12V, 28V
7	1 OZ	SIG 3
8	1 OZ	SIG 4
9	1 OZ	+5V, 28V RETN
10	1 OZ	BATTERY, +10V CHARGE 1 & 2
11	1 OZ	SIG 5
12	1 OZ	SIG 6
13	1 OZ	CMOS +5V, CHAS GND
14	1 OZ	GND, 28V
15	1 OZ	SIG 7
16	1 OZ	SIG 8
17	1 OZ	GND, 28V RETN
18	1 OZ	+5V, 28V RETN
19	1 OZ	SIG 9
20	2 OZ	SIG 10
21	2 OZ	GND
22	2 OZ	SIG 11, MIA
23	1 OZ	"0" BOTTOM

Figure 131. Backpanel Stackup

11.2.1 CPU Functional Description

The AP-101S Central Processor Unit is optimized for both MMP and MIL-STD-1750A Notice 2 architectures, although the 1750A architecture is not implemented in the standard AP-101S configuration. The AP-101SG/1750, a special groundbase development configuration of the AP-101S, implements the 1750A architecture and shares with the AP-101S a common Central Processor Unit. The CPU flow diagrams are shown in Figure 132 on page 259, Figure 133 on page 260, and Figure 134 on page 261.

11.2.1.1 Instruction Unit

The Instruction unit uses its own instruction counter (IU-PC) to prefetch instructions from memory during unused memory cycles. Instructions are fetched two words (16 bits each) at a time and are put into a 16 x 16-bit FIFO instruction file, shown in Figure 135 on page 262.

The 16 word instruction file is organized as two 8 x 16-bit buffers. The most significant 16-bit instruction word is placed in the even address portion, and the least significant is placed in the odd address portion. The file is further divided between the higher order addresses (A) and the lower order addresses (B) so that it is accessed as shown in Figure 135 on page 262.

In addition to the A and B sets of buffers, the instruction file also has a C set of buffers to minimize delays when a branch is taken. When a branch instruction is encountered, the EA-unit generates the branch address, prefetches two words from that location, and places them in the C set of buffers. If the branch is not taken, the instruction file continues to fill up the A and B sets of buffers as before. However, when the Execution unit determines that a branch is taken, it directs the instruction file to switch from the A and B buffers to the A and C buffers and to start fetching instructions from the location following the branch address (branch address plus two, since the EA-unit has already fetched the two words located at the branch address and placed them in the C buffer). The A and C buffers are now the sources of instructions for the EA-unit, and no time has been lost by the switch. The next branch taken will cause the instruction file to switch from buffers A and C back to buffers A and B again, and so forth.

Instructions can be either one or two 16-bit words long, so two alignment multiplexers (muxes) at the output of the file ensure that a 16-bit instruction or the most significant word of a 32-bit instruction is always output from the left mux. To correctly output a 16-bit instruction at an even or odd address, the left mux chooses its even or odd input, respectively. For 32-bit instructions start-

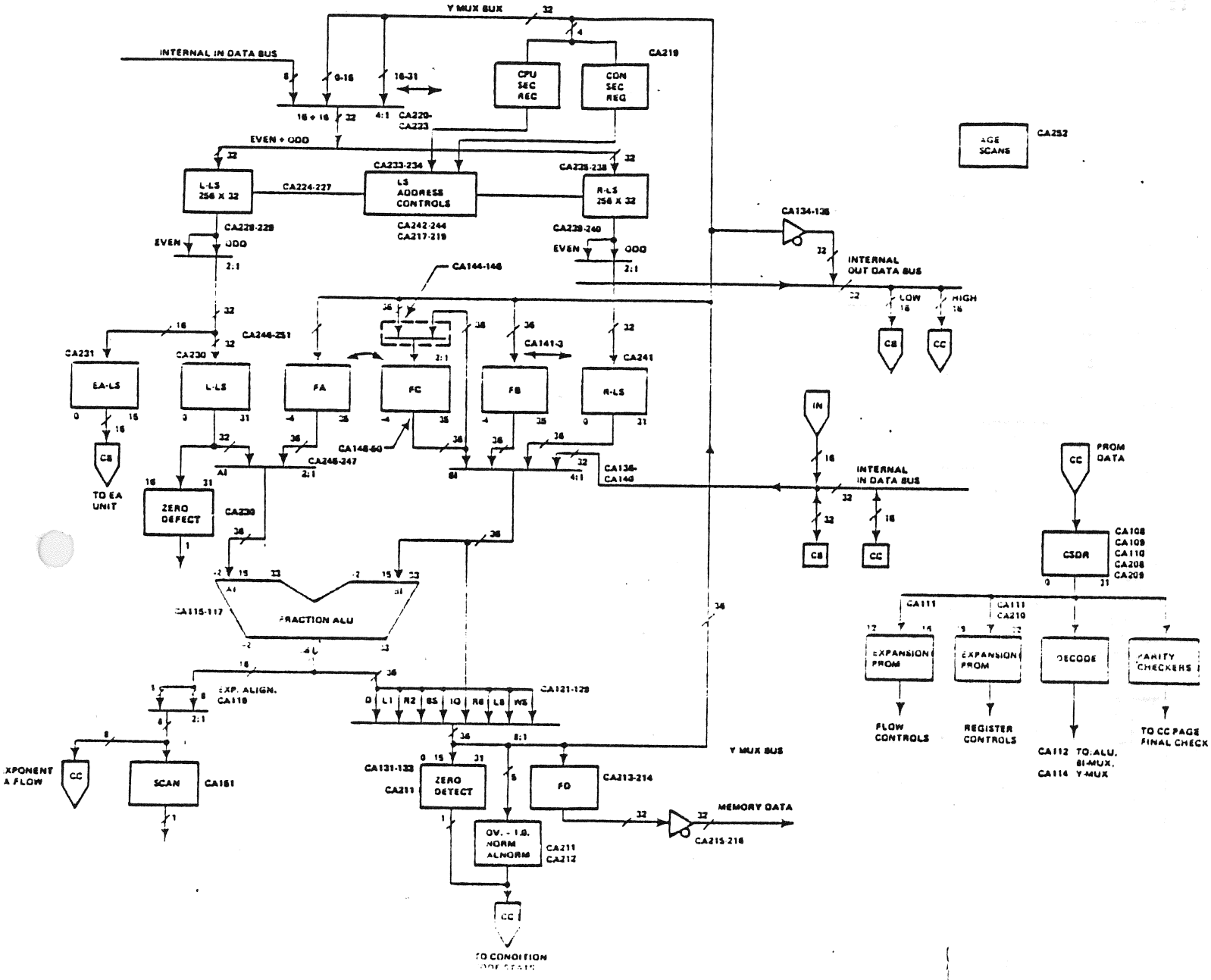


Figure 132. CPU1 Flow Diagram

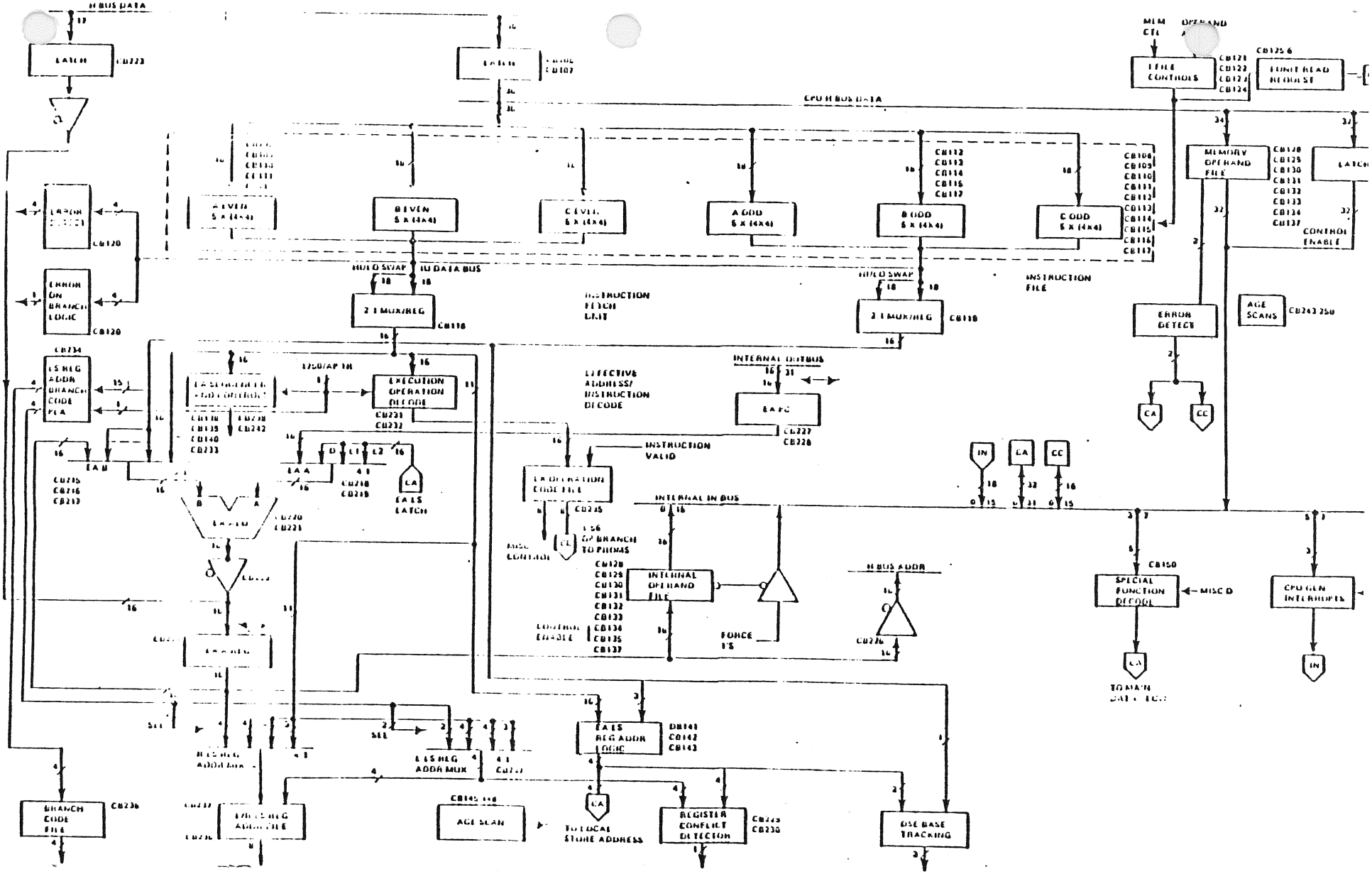


Figure 135. CPU2 Flow Diagram

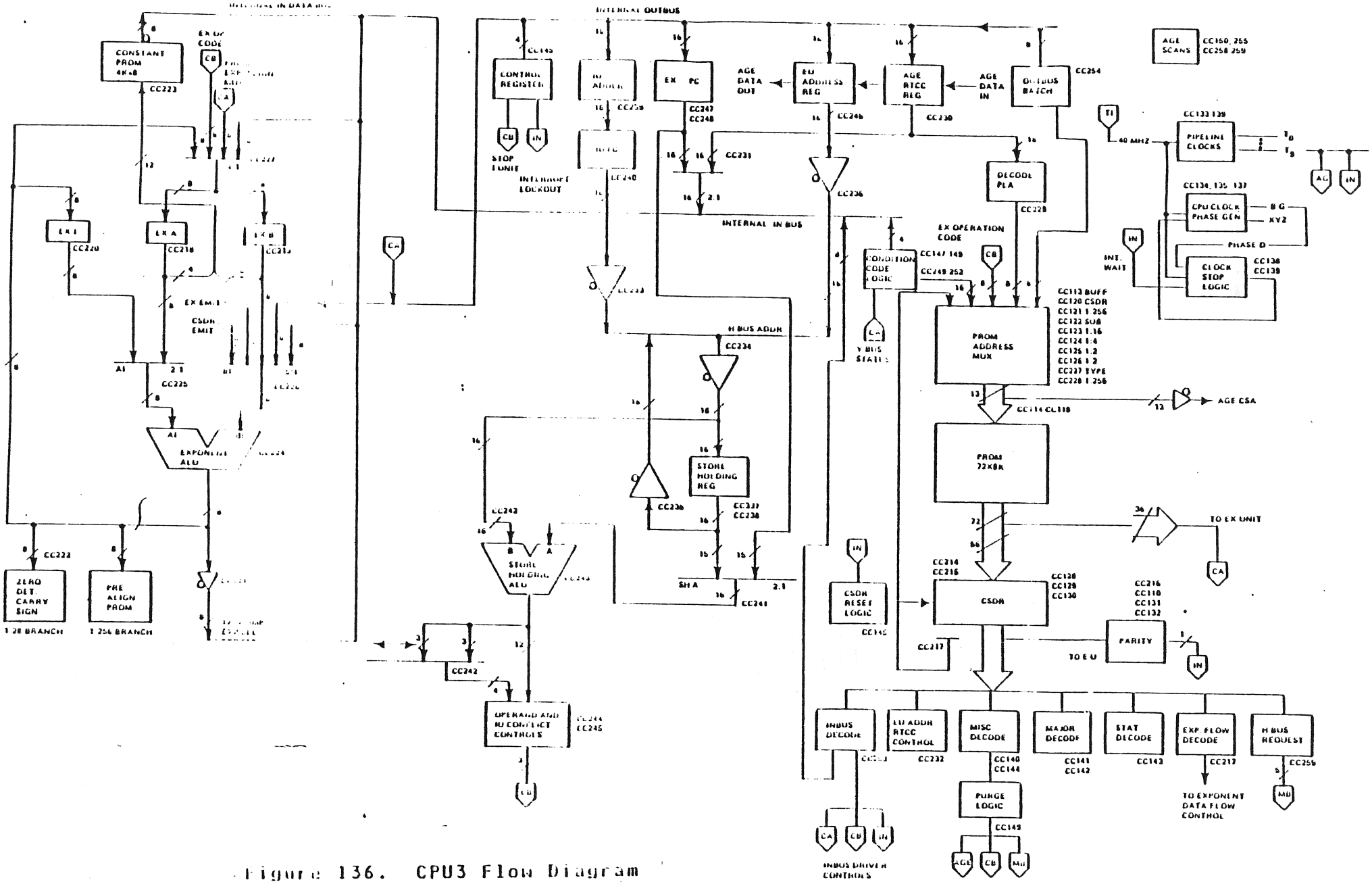


Figure 136. CPU3 Flow Diagram

6458

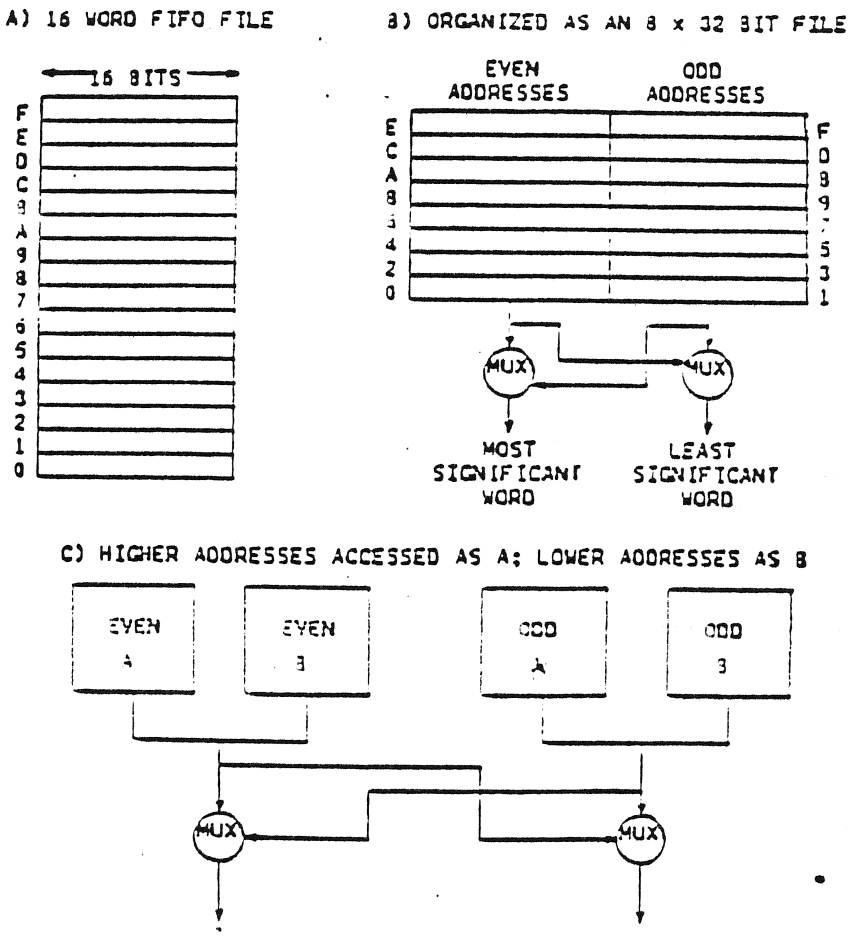


Figure 135. Instruction File

ing at an even or odd address, the left mux again chooses its even or odd input, respectively, and the right mux the complementary input, odd or even, respectively. Figure 136 on page 264 shows how the 16-bit instruction AAFF is output from even and odd locations, and Figure 137 on page 264 shows the 32-bit instruction AAAA FFFF being output from even and odd locations.

11.2.1.2 Effective Address Unit

The EA-unit decodes the instruction and provides this decoded version to the Execution unit. The EA-unit also handles the generation of the operand addresses and prefetches the operands for the EX-unit. Operands or addresses can be provided by the instruction as immediate data, or may need to be calculated by adding any combination of the following:

1. Immediate data
2. Contents of a base register or memory location
3. Contents of an index register
4. Displacement.

The EA-unit and I-unit data flows are shown in Figure 138 on page 265. Instructions sent from the I-unit enter two logic sections in the EA-unit. In the Execution Operation Decode section, the instruction is decoded, converted into an 8-bit code, and sent to the EX Operation Code File for the EX-unit to access when executing an ENDOP 1:256-way branch issued by the microcode. The EA Sequencer and Controls section generates the control signals needed for the EA-ALU and its associated logic to compute the logical addresses of the operands and to prefetch those operands when necessary.

To compute operand addresses, the EA Sequencer and Controls section first determines what type of addressing is used in the instruction. The EA-unit then fetches the contents of any base or index register or memory location (indirect addressing) specified and selects from the instruction any displacement or immediate data for input to the EA-ALU. The EA-unit calculates the address of the operands by summing register or memory contents, immediate data and displacement as indicated by the type of addressing.

The EA-unit places the results of its calculations into the EA-A register, then sends them to the Internal Operand File. General register addresses are set up by the EA-unit for use by the EX-unit as required for the instruction. If an instruction requires an operand from memory, that operand is fetched and placed in the Memory Operand File by the EA-unit. The operands for the instruction have thus been prefetched into one of two files (internal or memory), and the EA-unit controls which of these files will be provided to the EX-unit.

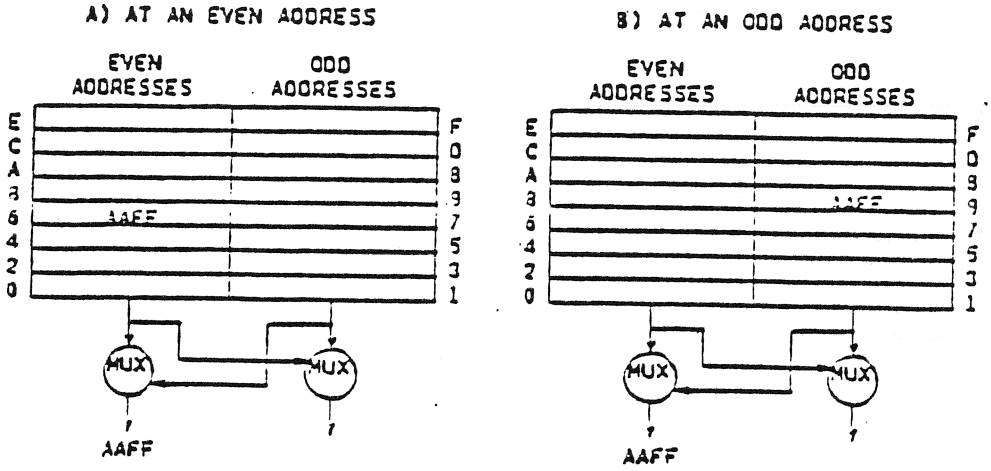


Figure 136. Accessing a 16-Bit Instruction

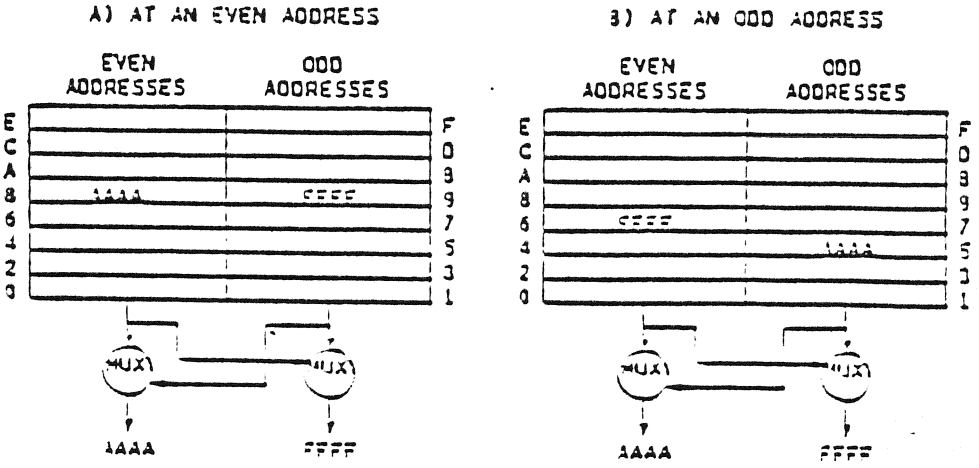


Figure 137. Accessing a 32-Bit Instruction

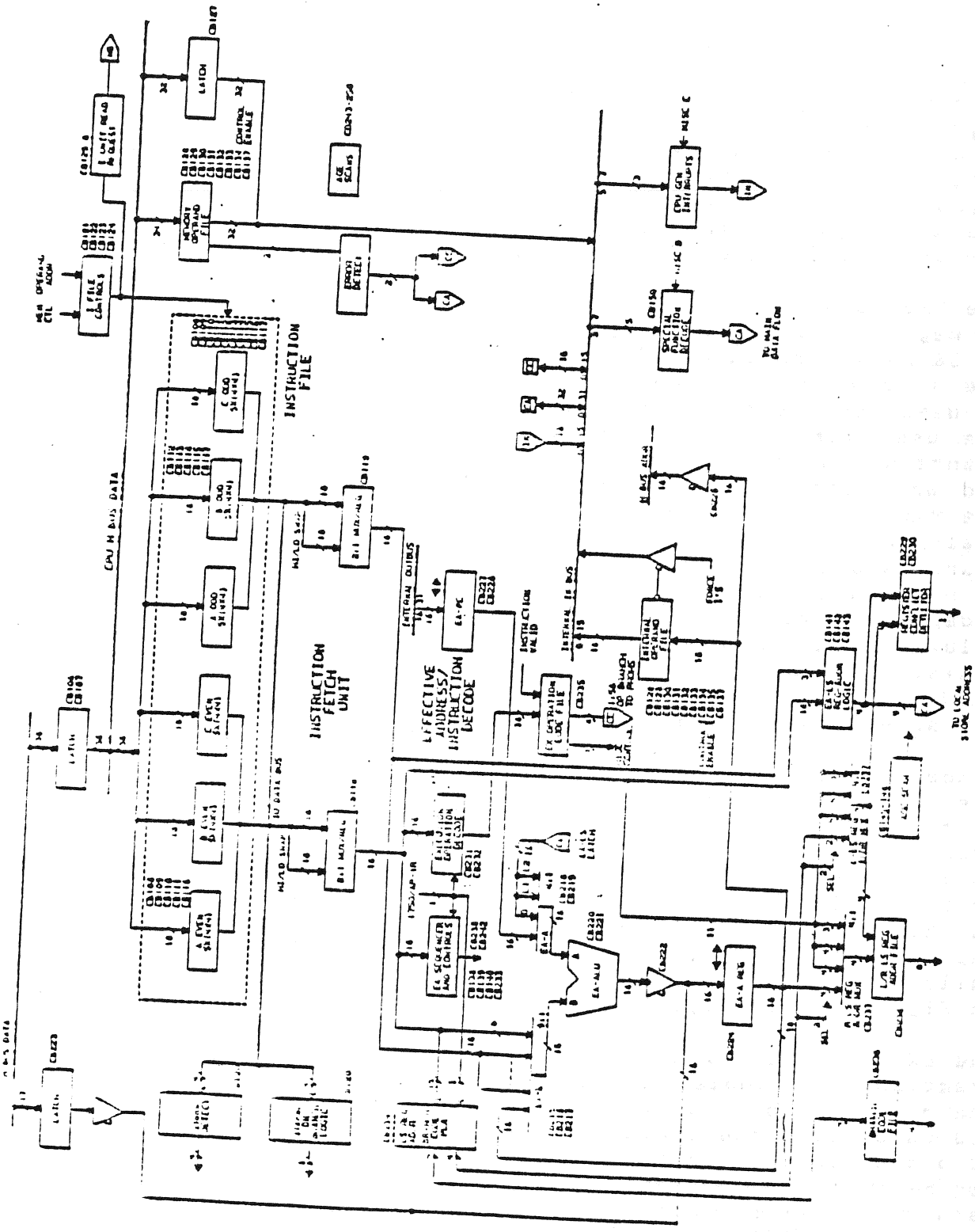


Figure 138. EA-Unit Data Flow Diagram

11.2.1.3 Execution Unit

The Execution unit contains all the logic needed to perform 16- or 32-bit fixed point operations and 32-, 40-, or 48-bit floating point operations. Microcode provides the control signals for the hardware in this unit and is contained in an 8K x 72 Programmable Read Only Memory (PROM). Thirty-two x 8-bit Expansion PROMs are used to minimize the width of the microword while still allowing multiple control signals to the hardware. Five bits in the microword address one of the 32 locations in one or more of the Expansion PROMs to provide a 16- or 24-bit field of control signals.

The CPU local store (LS) consists of two duplicate 256 x 16-bit banks of registers which are organized as 32 sectors (16 CPU, 16 constant) of 16 registers each. The general purpose registers are located in one (1750) or two (MMP) of these sectors, and in MMP another sector is used for the floating-point register set. The remaining sectors are used for temporary storage of partial results or contain constants which are loaded from the constant prom during machine reset and are accessed by the microcode for certain computations. There are two identical LS banks, a left LS and a right LS arranged as a dual-port local store. To the macroprogrammer, the local store appears as one set of general purpose registers; but the two halves may be read independently by the microcoder so that the contents of two independent registers may be used in the same machine cycle. This allows simple operations, such as add or subtract, involving two registers to be completed in one 250 ns machine cycle. When writing to local store, both the left and right halves are written into at the same locations to keep the contents of the two sides identical.

Since both the EX and EA-units may need to access local store during the same machine cycle, provision has been made for local store to be time- shared. In a 250 ns machine cycle, the EX-unit reads local store during the first 75 ns, the EA-unit reads LS during the second 75 ns, and the EX-unit writes to local store during the last 100 ns of the cycle. This requires the EX-unit to perform its computations in the second 75 ns period while the EA is accessing local store. The EA-unit performs its computations in the last 100 ns of the cycle while the EX-unit is writing to local store. This timing is shown in Figure 139 on page 267.

The EX-unit data flow is shown in Figure 140 on page 268. A 36-bit Fraction ALU handles computations involving fixed point numbers and the mantissa portion of floating point numbers. The 8-bit Exponent ALU calculates the exponent in floating point operations and is used as a counter in iterative operations. In addition, the Exponent ALU can be used as an extension of the Fraction ALU to provide an expanded data flow (40 bits) for some Extended Floating Point operations in the MIL-STD-1750A architecture.

Input to the Fraction ALU can come from local store, the FA, FB, or FC registers, and the internal data bus where data from the Internal and Memory Operand Files and from EX-unit memory reads is placed. Provision is made for ALU results to be shifted. At the output of

the ALU, the Y Mux is capable of passing data directly or shifting left 1, right 2, left 8, right 8, 16-bit word swaps, 8-bit byte swaps, or setting up for I/O. Data from the output bus may be sent to local store, the FA, FB, and FC input registers, and the FD register. The FD register is dedicated to holding data which will be stored in memory.

11.2.1.4 Typical Instruction Execution

The following example will illustrate the roles of the EA and EX-units in the execution of a typical instruction. The instruction A (add) of MMP is a 32-bit integer add using the base-relative indexed addressing mode (contents of base register + the displacement (bits 21 - 31 of the instruction) + contents of index register (shifted left 1 for a fullword alignment) = address of the second operand). R1 is the register containing the first operand, D2 is the displacement, X2 is the index register, and B2 is the base register. The result of the add is stored in R1.

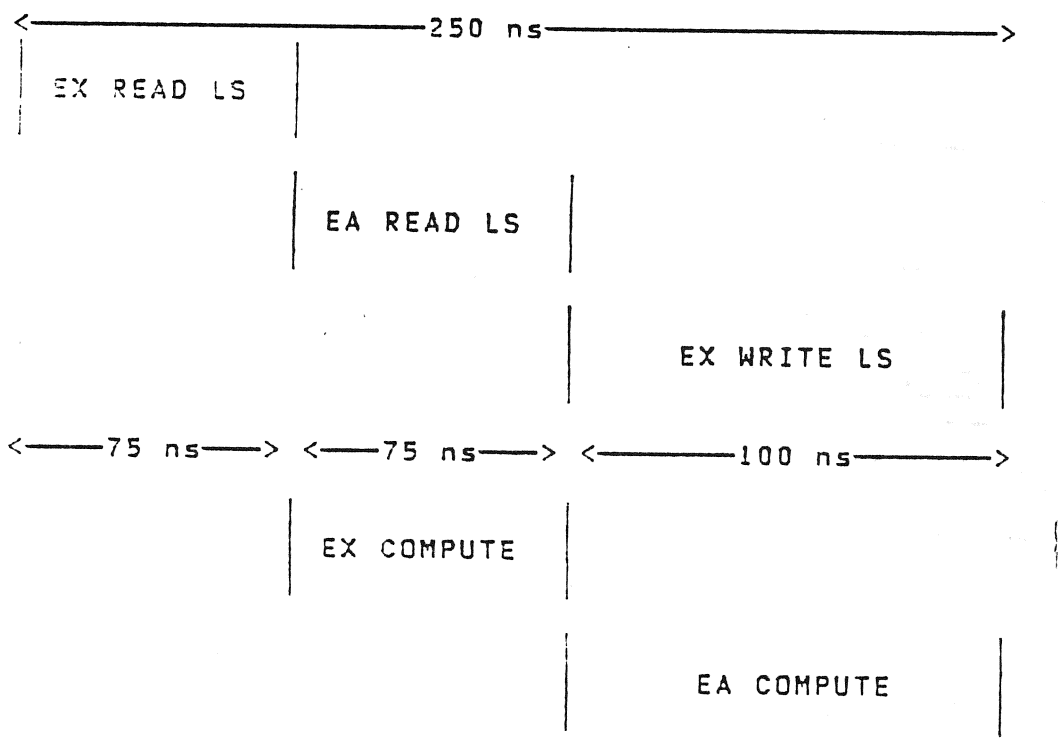


Figure 139. Time-Sharing of Local Store

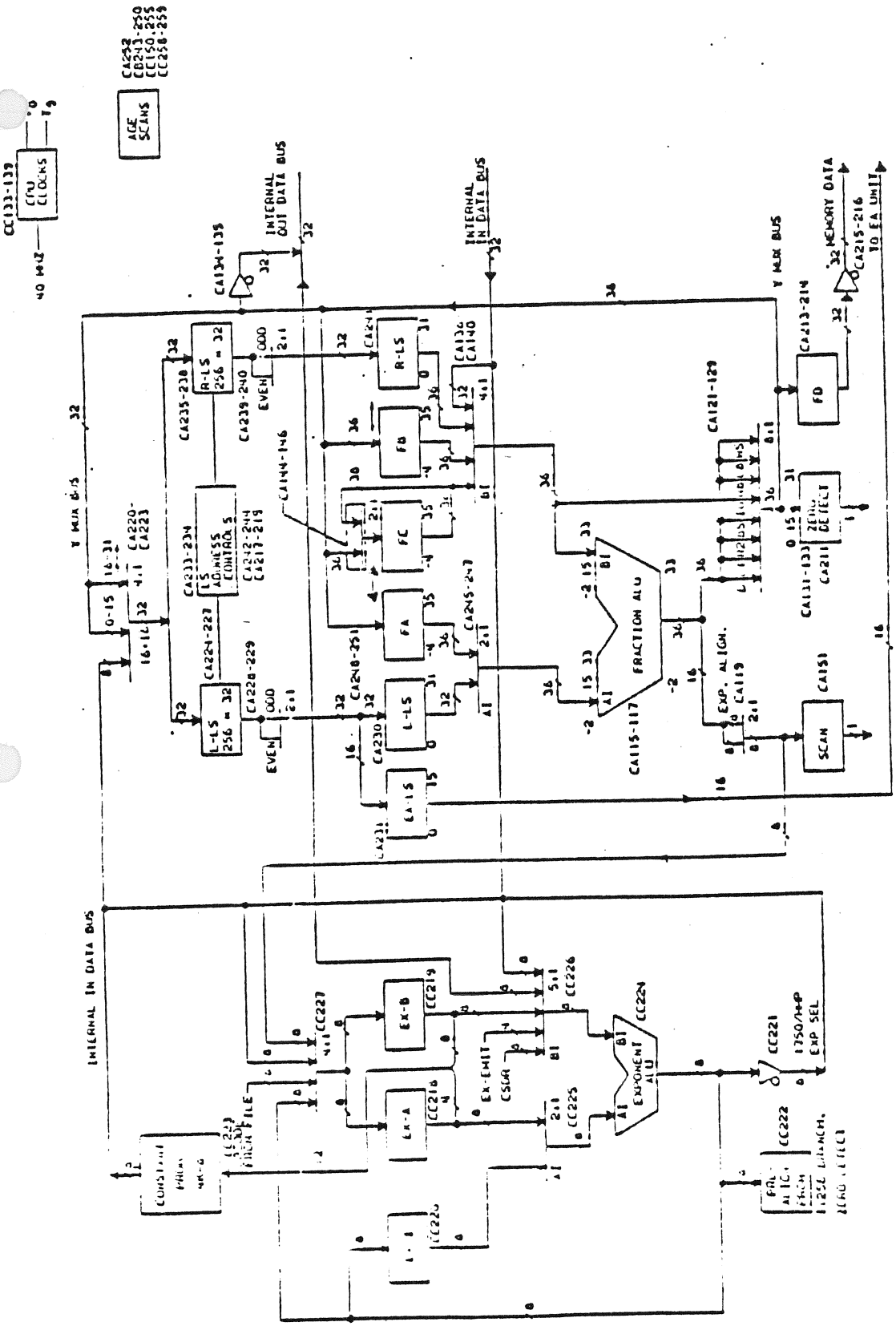


Figure 140. Execution Unit Data Flow Diagram.

Instruction: A R1,D2(X2,B2)

EA-Unit

- Decodes instruction and places an 8-bit value of x'A0' in the EX Operation Code File to be used as a vector for the 1:256-way branch by the EX at ENDOP. This is because the RS format add instruction begins at microcode location x'02A0'.
- Fetches the contents of B2 and adds this to D2, storing the result (the Preliminary Effective Address) in the EA-A register.
- Fetches the contents of X2, shifts these contents left 1 and adds them to the contents of the EA-A register, storing the result (the Effective Address) in the EA-A register.
- Fetches the second operand from the memory address given by the contents of the EA-A reg and places it in the Memory Operand File
- Selects R1 (instruction bits 5-7) as the left local store register address and selects the contents of the Memory Operand file rather than the Internal Operand file to be placed on the Internal In Data Bus, or INBUS, when the ENABLE OPERAND signal from the EX-unit goes high.

EX-Unit

- Selects the left local store input for the fraction AI mux and the INBUS input for the fraction ALU BI mux. The INBUS contains contents of Memory Operand file by default (statb3 must be zero).
- Adds operands
- Outputs operands directly (no shifting) at Y Mux to Y-Bus
- Writes data from Y-Bus to both left and right local store using R1 as the address of the register to be written into.

11.2.1.5 Conflicts and Hazards

Several faults are associated with the operation of a pipelined machine:

1. Register Conflicts
2. Operand Conflicts
3. I-Unit Hazard (Store Within Range)

The CPU contains the logic necessary to detect these conflicts and take appropriate action while minimizing any performance impacts. These conflicts are explained below.

Register Conflicts: A register conflict occurs when the EX-unit modifies the contents of a register which will be used in the EA calculation of any of the next three instructions. When a register conflict is detected, the EA must wait until the EX-unit has completed its register store, then start again using the new contents of the register.

Operand Conflicts: An operand conflict occurs when the EX-unit will modify the contents of a memory location whose contents will be prefetched for any of the next three instructions. When an operand conflict is detected, the EA-unit must wait until the EX-unit has completed storing into the memory location before it can access that location.

I-Unit Hazard (Store Within Range): An I-unit hazard occurs when the EX-unit modifies a memory location which may have been prefetched by the Instruction unit. When this occurs, the entire pipeline must be purged and restarted with the instruction following the store.

11.2.2 Memory Management Unit Functional Description

The AP-101S contains a two page Memory Management Unit (MMU) which incorporates numerous functions in addition to management of main memory. The MMU flow diagrams are shown in Figure 141 on page 272 and Figure 142 on page 273. Included among the diverse tasks performed by the MMU are the following functions:

1. The MMU arbitrates and controls the timing and sequencing of the HBUS.

2. The MMU controls all timing and sequencing to the mainstore in the AP-101S computer.

3. The MMU contains the address expansion logic for the system. The address expansion mechanisms are architecturally defined and are different for each architecture. The MMU accommodates either under external control.

4. The MMU is responsible for detecting, capturing and posting memory related faults. These faults vary according to architecture, system configuration and memory requestor.

5. The MMU directs I/O commands to the proper system element via designate generation.

6. The MMU supports testability by:

   a. Providing various diagnostic modes of operation under control of the MMU mode register.

   b. Providing several serial scan paths.

   c. Providing several IIO (Internal I/O) commands which make various MMU registers accessible to the diagnostic programmer.

   d. Further identifying faults detected by the MMU Memory Fault Extension Register (MFER).

   e. Providing an HBUS arbiter port for the tester.

11.2.2.1 MMU Clock Generation

The MMU generates a 40 MHz clock common to all the pages that are attached to the HBUS and receives a time 9 sync pulse from the CPU-3 page. From these two signals, a series of 10 pulses, each 50 ns in width is created. The 10 clocks are labeled T0 through T9 with the newly created T9 corresponding to the sync time 9. The MMU clocks are illustrated in Figure 143 on page 274.

Figure 141. MMUA Flow figure

Figure 142. MMUB Flow Diagram

MEMORY MANAGEMENT UNIT PAGES (MMU1, MMU2)

The MMU also contains a 24 MHZ oscillator for the 1553 page in the 1750 mode. The oscillator is used to generate 12 MHZ and 24 MHZ clock for the 1553 page.

11.2.2.2 MMU Reset Logic

When an Inhibit Main Store (IMS) is generated, a system reset is issued on the MMU. The Interrupt page generates an IMS pending signal shortly before issuing an IMS. When this signal is active, the MMU stops all HBUS activity. When IMS becomes active, the MMU resets itself and 250 ns later starts all activity again. At this point, any request to the arbiters will be acknowledged.

11.2.2.3 Mode Control

The MMU supports several different modes of operation. To change these modes, the user must issue the internal I/O (IIO) command of

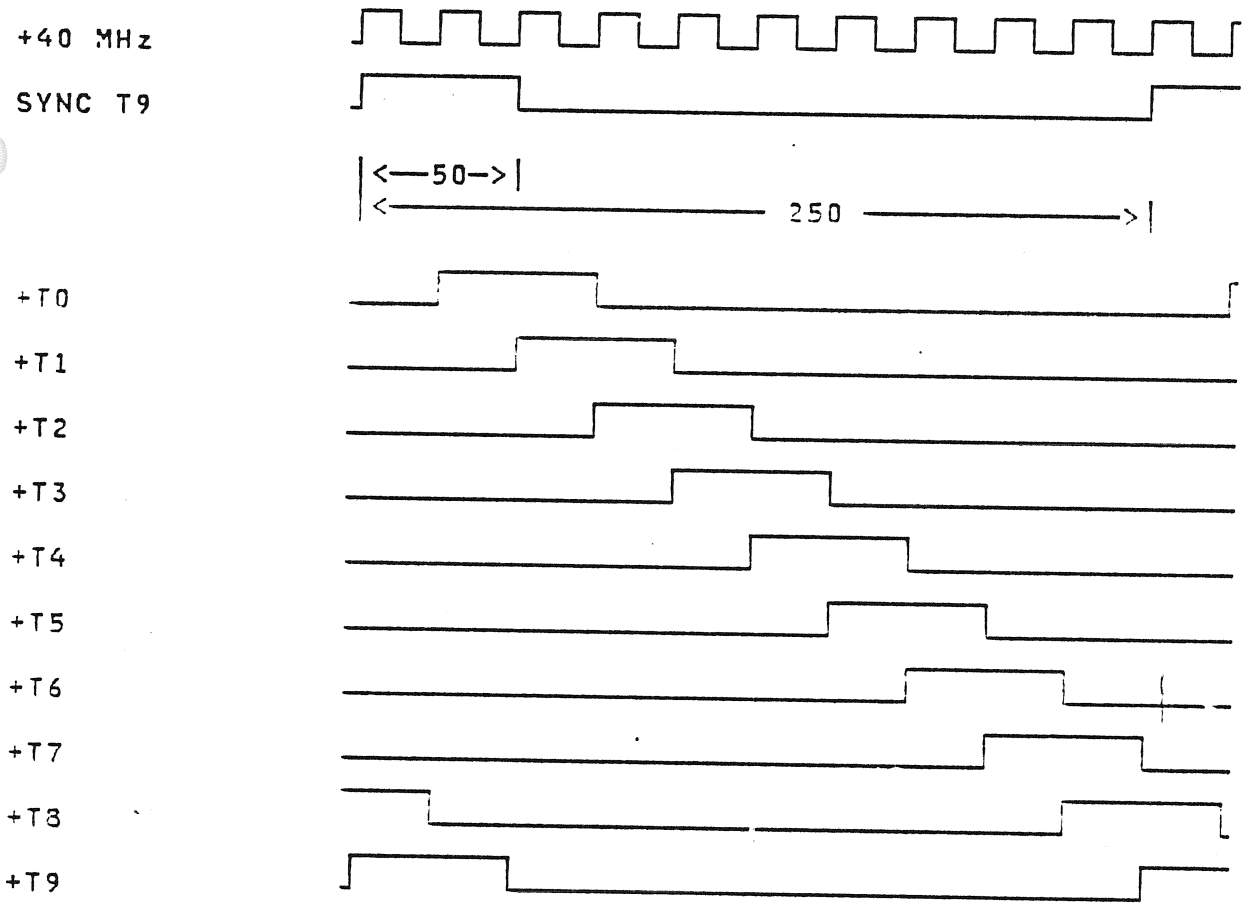


Figure 143. MMU Clocks

X'9407'. To read the current mode of operation, The IIO code of X'140B' can be used. There are 10 functions defined by the mode register (Figure 144 on page 275).

Figure 144. MMU Mode Register

FUNCTION	WRITE BIT	READ BIT
INHIBIT DMA'S	06	22
DISABLE STORAGE ERRORS	07	23
BCE DISABLE	08	24
SPECIAL STORE PROTECT	09	25
TRANSMIT DISABLE	10	26
SYSTEM IPL	11	27
PASSTHRU CMOS	12	28
SYNDROME/CHECK BIT MODE	13	29
CODE IDO	14	30
SCRUB DISABLE	15	31

11.2.2.4 Bus Protocol

The Memory Management Unit (MMU) transfers data between the central processor and the IOP through the HBUS. This is a high-speed synchronous bus developed to transfer memory data at high rates of speed.

11.2.2.5 Memory Address Expansion

The MMU handles all memory address expansion requirements for the AP-101S computer. The general functions performed by the MMU address expansion logic are:

1. A 20-bit Advanced Programmable Tester (APT) address is accommodated on the 16-bit HBUS address bus

2. Memory addressing for 512K halfwords (20-bit) is provided.

3. Either halfword (16-bit) or fullword (32-bit) accesses are permitted during a single memory cycle.

4. No boundary constraints are imposed on fullword accesses.

5. The ability to bypass the address expansion logic is provided.

6. Separate address expansion mechanisms are provided for each architecture.

11.2.2.6 Address Interfaces

11.2.2.6.1 IOP Interface

The IOP in the AP-101S computer always provides a physical 18-bit address on the HBUS.

11.2.2.6.2 CPU

The CPU passes a 16-bit logical address to the MMU via the HBUS. This address and all requestor generated HBUS tag bits are passed on the HBUS during any CPU acknowledge cycle. The CPU unit sourcing this address is determined by the particular acknowledge that was granted. During the address cycle, the 16-bit logical address is selected into the MMU address flow, and must be expanded into a 20-bit physical address (unless the operation type bits specify "no map") in the 1750 architecture or a 19-bit physical address in the MMP architecture.

11.2.2.6.3 Avionics Programmable Tester (APT)

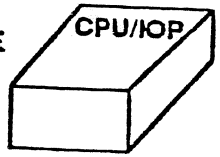
When an SDI (APT) acknowledge is granted, a 20-bit physical address is passed to the MMU. The low order 16 bits are passed over the HBUS via normal HBUS protocol. The high order four bits are serially scanned into a holding register on MMU1 at the same time that the testers serial interface logic (refer to the SDI description) scans in the HBUS Simulator register on the interrupt page. During the SDI ACK, the high four bits and the low 16 bits are concatenated and selected into the MMU address flow. This 20-bit physical address always bypasses the address expansion logic.

11.2.2.7 Address Expansion Logic

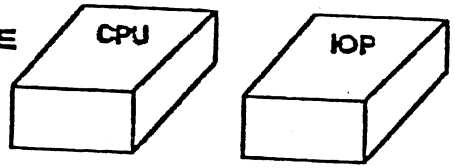
The requirement to support single cycle memory accesses for doublewords on any boundary based on a single address that is passed to the MMU dictates the following MMU hardware support:

1. Address adder

SPACE SHUTTLE AP-101S



SPACE SHUTTLE AP-101B



POWER	560 WATTS	780 WATTS	SAVE 1100 WATTS SLEEP MODE: 56 WATTS
WEIGHT	64 LBS	117 LBS	SAVE 318 LBS
MEMORY	CMOS: 256K FW'S ~~25 BITS~~ 250NS Cycle/Access ~~16 Data~~	CORE: 104K FW'S	0120 G3 ARCHIVE
HALF WORD	16 DATA BITS 6 EDAC BITS 3 STORE PROTECT Memory Error Code Check Bits	16 DATA BITS 1 PARITY BIT 1 STORE PROTECT	ERROR DETECTION AND CORRECTION MEMORY SCRUB
SPEED	> 1000 KOPS	420 KOPS	018F 1.7 TO 1
BATTERY BACKUP	6 RECHARGEABLE NICADS	.....	REMOVABLE SRU
BITE	TEMPERATURE CHARGER BATTERY SOFT ERROR COUNTER	.....	
MTBF	DESIGN: 6,000 HRS OUTLOOK: 10,000 HRS	5250 HRS	CURRENT AP101S: 24,000 HRS

2.0 AP-101S STRUCTURE

2.1 SHUTTLE INSTRUCTION SET

The AP-101S system structure encompasses the functional operation of main storage, the central processing unit (CPU), and program-controlled I/O facilities.

2.1.1 Information Formats

The system transmits information between main storage and the CPU in units of 16 bits, or in integer multiple of 16 bits. Each 16-bit unit of information is called a halfword. Six error correction bits and three voted storage protection bits are also associated with each halfword for the AP-101S but later references in this manual to the size of data fields exclude these bits. The AP-101S/G has two storage protect bits per halfword.

Halfwords may be handled separately or in pairs. A fullword is a group of two consecutive halfwords. Both halfword and fullword instructions and operands are used. Their location is always specified by the address of the leftmost halfword (leftmost halfword is the numerically smallest address). The instruction length is designated implicitly in every instruction; the operand length is also implicit.

Within any instruction and operand format, the bits making up the format are consecutively numbered from left to right, starting with the number 0, as shown in Figure 2-1.

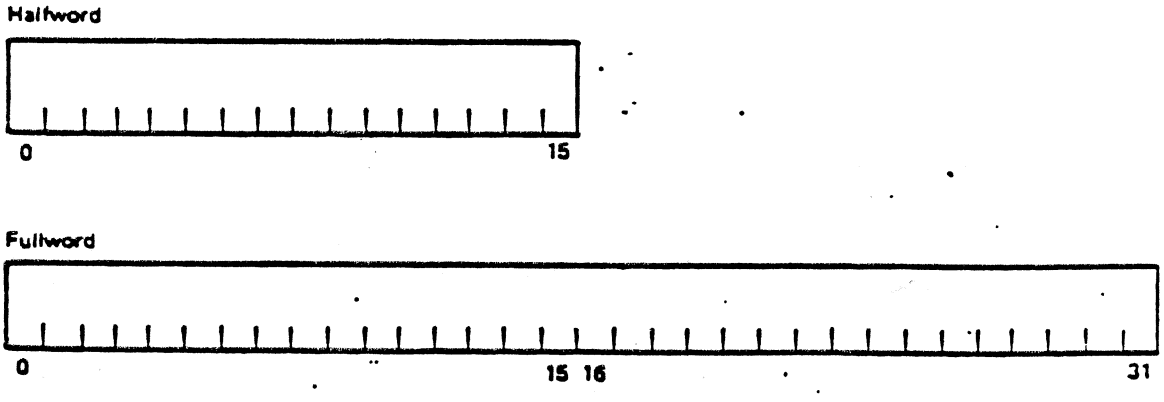


Figure 2-1. Instruction and Operand Bit Numbering

2.1.2 Addressing

Halfword locations in storage are consecutively numbered starting with 0. Each number is considered the address of the corresponding halfword. The addressing technique uses a 19-bit binary address to a maximum of 2^{19} halfword addresses. This set of main storage addresses includes some locations reserved for special purposes, such as program status words; consequently, these special locations should not be used for any purpose not implicitly defined.

2.1.3 Information Positioning

Unlike previous versions of the AP-101 computer, the AP-101S does not require either fullword instructions or fullword/doubleword operands to be located in main storage on even boundaries.

2.2 CENTRAL PROCESSING UNIT

The central processing unit (CPU) contains facilities for addressing main storage, fetching or storing information, for arithmetic and logical processing of data, sequencing instructions in the desired order, and for initiating the communication between storage and external devices.

The control section guides the CPU through the functions necessary to execute the program.

2.2.1 Program Addressable Registers

Two sets of eight fixed point general registers and one set of eight floating point registers are under explicit program control. The contents of one or more of these registers (32 bits) participate in most CPU operations. Associated with each of the fixed point registers is a 4-bit addressing extension register (Data Sector Extension or DSE), the use of which is described below in Extended Addressing.

Conceptually, an additional doubleword status register, called the program status word (PSW), is the focal point for machine status. The contents of the PSW are updated during each instruction. Consequently, the PSW reflects current machine status following every instruction. The PSW participates implicitly in status switching, branching operations, and address calculations. Condition codes resulting from an instruction are also part of the PSW.

In addition to the PSW and the general and floating point registers, the CPU also contains working registers used for storage addressing, storage buffering, shift and operation counting, and operand storage. These registers are of no direct concern to the programmer and are not described herein.

The contents of the PSW specify which of the two sets of general registers is in current use. Only the contents of the selected general register set can participate in arithmetic operations and the contents of unselected sets of general registers cannot be altered by a program. An alternate set of general registers can be selected by changing the PSW. Only one set of the fixed point, general-purpose registers and the floating point registers are available to the program at any one time.

General register contents can be used interchangeably as operands for arithmetic, logical, and shifting operations, or as base and index registers for relative addressing. Each set of general registers is numbered from 0 through 7 and is addressed as shown in Figure 2-2.

General Register Number	Register Function		
	Operand	Base	Index
0	000	00	None
1	001	01	001
2	010	10	010
3	011	11 or None*	011
4	100		100
5	101		101
6	110		110
7	111		111

*11 = Register 3 for SRS; none for RS

Figure 2-2. General Register Addresses

Note that general registers 4 through 7 cannot contain base addresses and that general registar 0 cannot contain an index.

For addressing data, general registers 0-3 can be augmented by 4-bit Data Sector Extension (DSE) registers or by the DSR in the PSW to address beyond 16-bit capabilities. There are 16 DSEs, one for each of the eight general-purpose registers in each of the two sets of general registers.

For some operations, a pair of general registers is linked to form a 64-bit doubleword register. The most significant half of a doubleword operand is contained in the specified register; the least significant half of the doubleword is in the next higher-numbered register (determined by Modulo 8 addition of one (1) to the specified register). Note: If Reg 7 is specified, the least significant half of the double word operand is contained in Reg. 0.

2.2 Fixed Point Data Representation

Data representation is fractional, with negative numbers represented in twos complement form. A halfword operand is 15 bits plus sign, a fullword operand is 31 bits plus sign, and a doubleword operand is 63 bits plus sign, as shown in Figure 2-3.

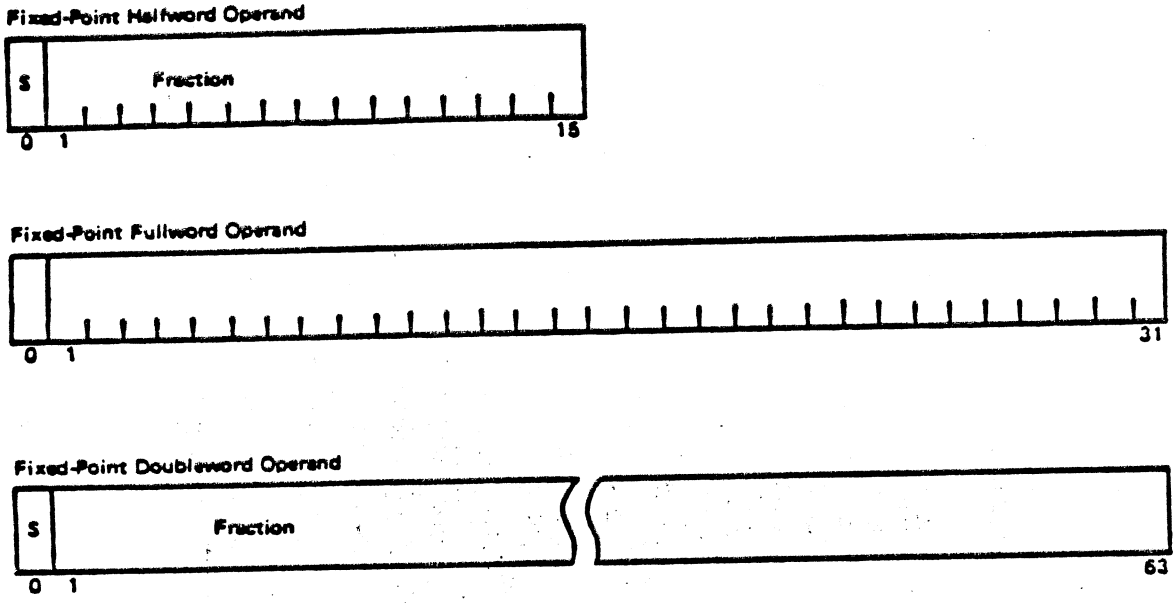


Figure 2-3. Fixed Point Operand Formats

In fractional data representation, the binary point is immediately to the right of the sign.

2.2.3 Instruction Formats

The length of an instruction format can be either one or two halfwords. Long format instructions provide maximum range and extended flexibility for addressing storage operands. Short instructions are used to (1) specify register-to-register operations, and (2) specify storage operands in cases where a small displacement is sufficient and complete address modification capability is not required.

Instruction formats overlap. Programs are written so that, in many instances, any given operation can be coded using either a halfword or a fullword instruction. In such cases, maximum use of halfword instructions results in increased storage efficiency and performance.

The three basic instruction formats are as shown in Figure 2-4. Halfword instructions are automatically selected by the assembler unless otherwise specified by the programmer.

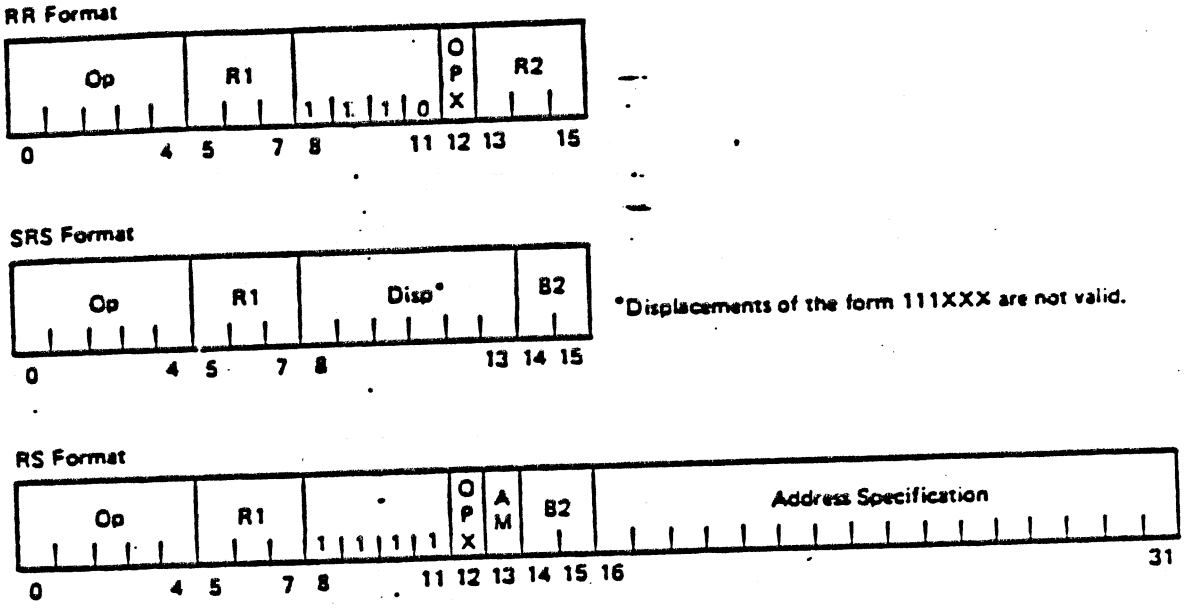


Figure 2-4. Basic Instruction Formats

The fields within the instruction formats usually are used as described below. The exceptions are described in conjunction with the individual formats and instructions.

Op This 5-bit field defines an operation, or the class of operation, to be performed by the CPU.

R1 This 3-bit field designates the register containing the first operand. Except for operations which alter main storage, the result usually replaces the first operand.

R2 This 3-bit field appears only in the RR format. It is used to specify a general register containing either the second operand or the address of the second operand.

B2 This 2-bit field specifies the register containing the base address.

Disp In halfword SRS format instructions, this 6-bit field is called the displacement. For the SRS format, the displacement is added to the base address specified by the B field to obtain a storage address.

OPX This bit is an extension of the OP field.

AM This field designates one of two fullword format addressing options.

Address specification	The second halfword of a fullword instruction is specified as either extended or indexed addressing.

See the Effective Address Generation Summary Chart, page 11-1.

2.2.4 RR Format Instructions

The RR format instructions (Figure 2-5) permit the specification of operations that use two general registers.

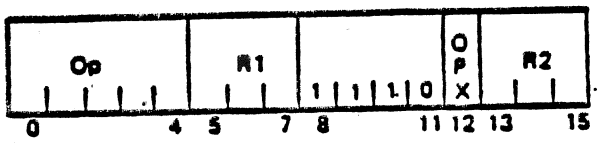
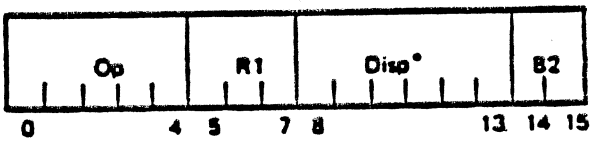


Figure 2-5. The RR Instruction Formats

The operation normally uses as operands the contents of two general registers. The R2 field specifies the second operand while the R1 specifies the first operand. The result of the operation usually replaces the first operand.

2.2.5 SRS Format Instructions

The SRS instruction format (Figure 2-6) is a compression of the RS format. It provides base plus displacement storage addressing.



* Displacements of the form 111XXX are not valid.

B2	Register Containing Base
00	General Register 0
01	General Register 1
10	General Register 2
11	General Register 3

Figure 2-6. SRS Instruction Format

The R1 field specifies the first operand register address. The 19-bit effective address (EA) of the second operand is developed as follows:

Step 1 First the positive integer contained in the displacement field is added to the contents of the base contained in the general register specified by B2.

When addressing halfword operands, the least significant bit of the displacement field (instruction bit 13) is aligned with base register bit 15. The 16-bit result is the sum of the base and the displacement, aligned as shown in Figure 2-7.

When addressing fullword operands using the SRS format, the least significant bit of the displacement field is aligned with base register bit 14 as shown in Figure 2-8.

Unlike previous versions of this architecture, bit 15 of a base register is significant when addressing fullword data. Fullword storage operands may now be located on odd address boundaries. Programs which utilize this feature will not be downward compatible.

Step 2 The 16-bit result of the addition of the base and displacement is expanded (see Expanded Addressing) to a 19-bit effective address (EA), and this is the address of the second operand.

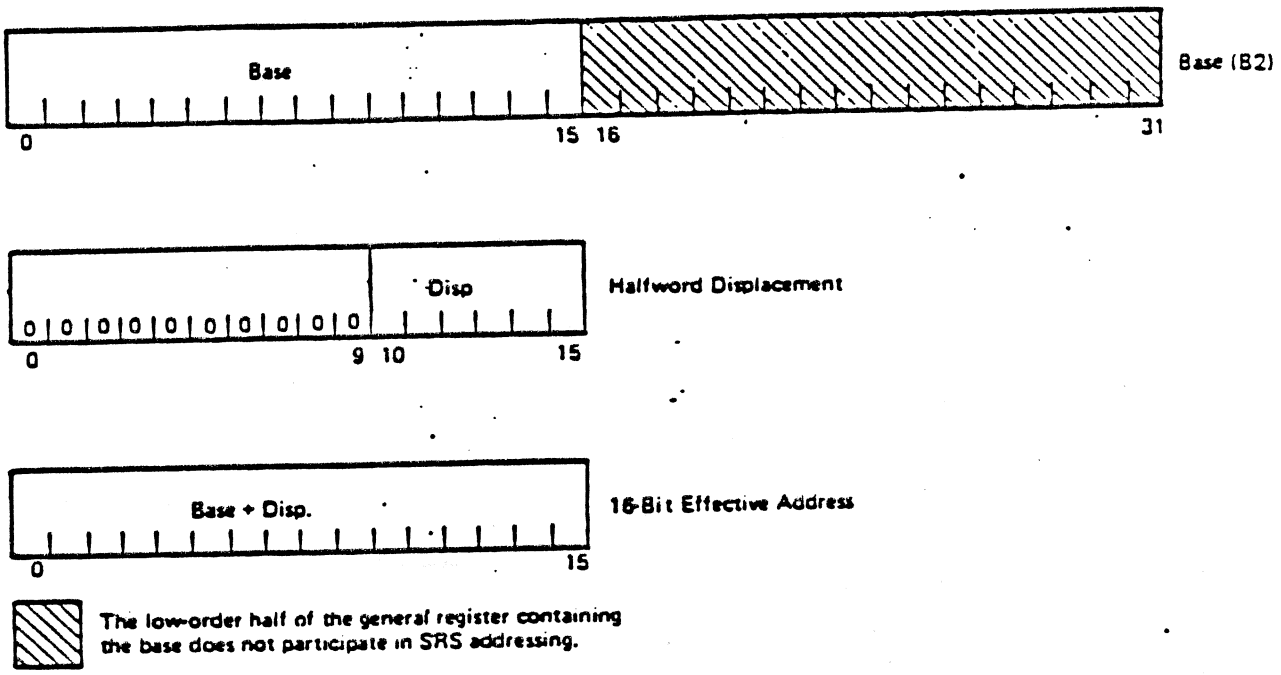


Figure 2-7. SRS Halfword Addressing

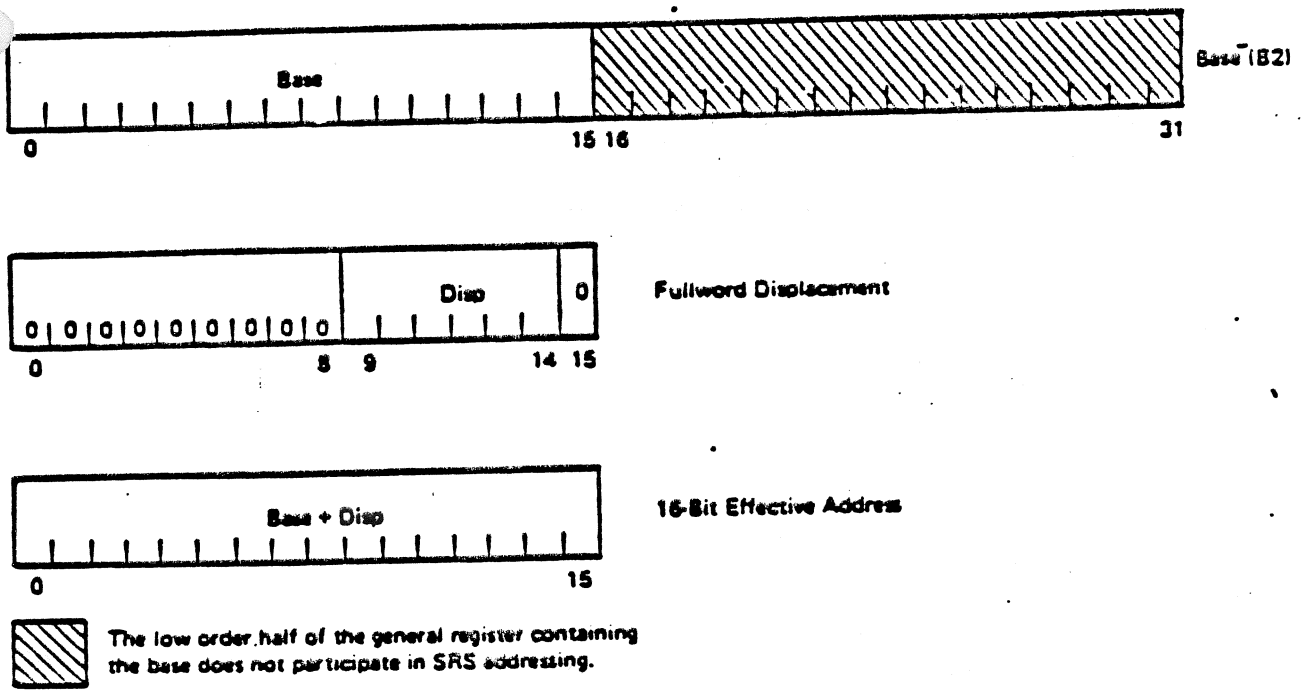


Figure 2-8. SRS Fullword Addressing

Except for store instructions, the result of operation between the first operand (the contents of general register R1) and the second operand (the contents of the EA) replaces the first operand for SRS format operations. The first operand replaces the second operand for store instructions.

2.2.6 SI Instructions

Direct initialization, modification, and testing of main storage is possible through the use of an immediate data halfword appended to an SRS instruction. See Figure 2-9.

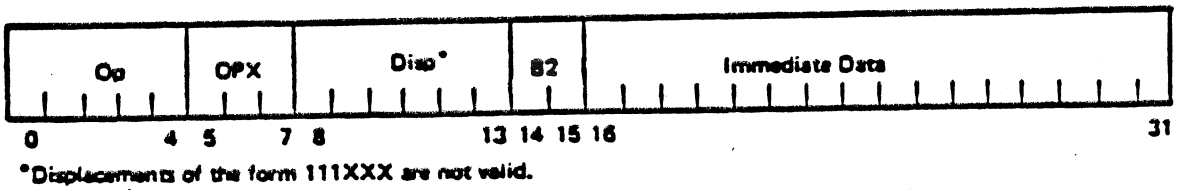


Figure 2-9. SI Instructions

a address of the halfword second operand is developed in the normal manner for SRS instructions using halfword addressing. Except for test instructions, the result of a operation between the halfword second operand and the immediate data replaces

the second operand. The second operand is not altered for test instructions. The first operand is never altered for SI instructions.

2.2.7 RI Instructions

Using an immediate data halfword appended to an RR instruction (Figure 2-10) permits direct initialization, modification, and testing of the most significant 16 bits contained in a general register.

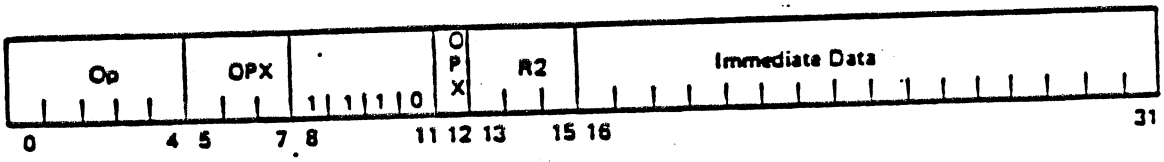


Figure 2-10. RI Instructions

Except for test instructions, the result of the operation between the second operand and the immediate data replaces the second operand. The second operand is not altered for test instructions. The immediate data first operand is never altered for RI instructions.

2.2.8 RS Format Instructions

There are two major classes of RS instructions, extended and indexed addressing modes, differing in the techniques used to specify the second operand. See Figure 2-11.

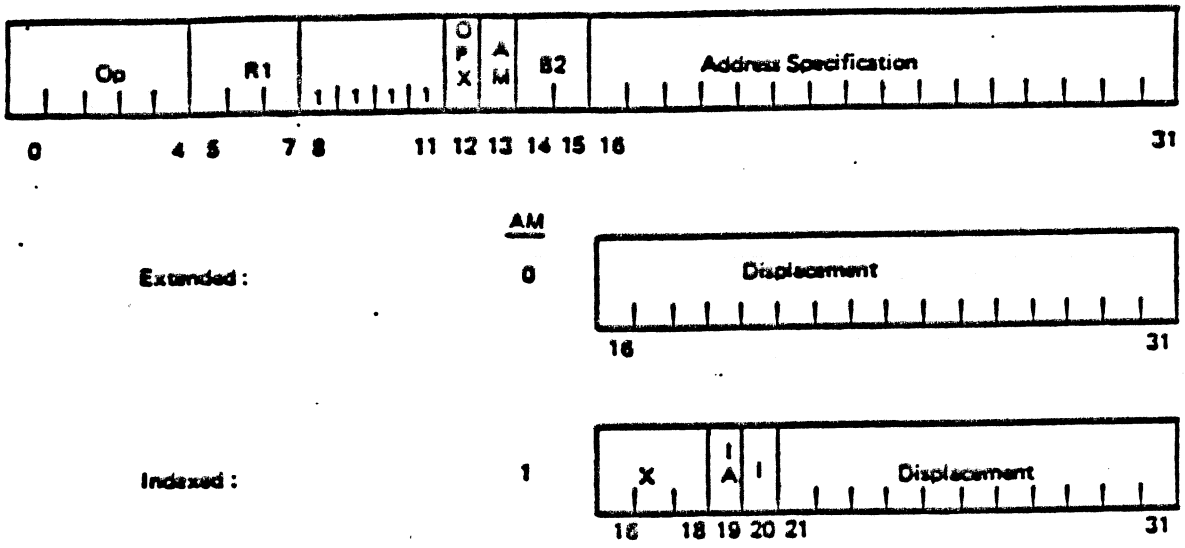


Figure 2-11. RS Instruction Formats

Extended addressing is specified when RS format bit 13 (AM) equals 0. This addressing mode provides a full 16-bit halfword displacement. The base and displacement are aligned as shown in Figure 2-12 when base addressing is performed.

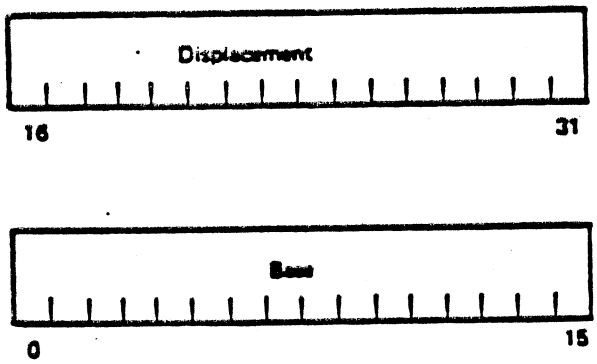


Figure 2-12. Displacement Alignment for Extended Addressing

Aside from the size and alignment of the displacement, RS extended addressing differs from SRS addressing in two other respects:

1. The alignment of the displacement is the same whether addressing doubleword, fullword or halfword operands.

2. When B2 equals 11, base addressing is not performed. In this case, the displacement is instead used directly as the effective address.

Indexed addressing is specified by RS format bit 13 (AM) equal to 1. This addressing mode contains three additional fields. Normally, they contribute to the effective address generation as follows:

X — This 3-bit field specifies one of seven general registers containing the index. Indexing is not performed when X is equal to 000. An index is contained in the upper halfword of a general register. The index is automatically aligned as illustrated in Figure 2-13. For additional information on index alignment, see Section 14. Consistent with the restrictions that apply to register usage and indirect addressing, general register contents can be used interchangeably as either a base or an index or both. When indirect addressing is specified, indexing follows indirect addressing (postindexing).

IA — This format bit, when a one, specifies indirect addressing. Indirect addressing is not performed when this bit is zero. In the instruction descriptions, the symbol @ denotes IA for the assembler.

I — This format bit, in conjunction with X and IA, specifies various address modes which are explained below. In the instruction descriptions, the symbol * denotes I for the assembler.

The development of the EA for the indexed mode (including IC relative) of operand addressing is explained in detail in the subsequent steps:

1. Indexed addressing is specified by RS format bit 13 (AM) equal to 1. This addressing mode provides an 11-bit displacement. The base and displacement are aligned as shown in Figure 2-14 when indexed addressing is performed.

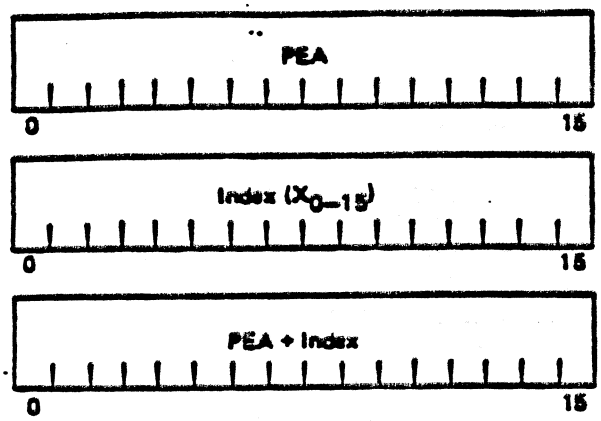
   The displacement is aligned so that bit 31 corresponds to base or index bit 15 and displacement bit 21 corresponds to base or index bit 5. The displacement is expanded to 16 bits by appending five leading zeros.

2. If B2 is not equal to 11, the 16-bit base, contained in the higher order half of the specified register, is added to the aligned displacement. This results in a preliminary effective address (PEA) whereby the PEA = (B) + Displacement.

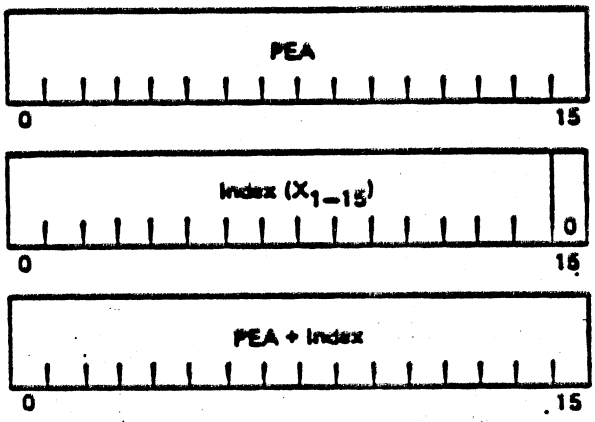
   If B2 is equal to 11, the aligned displacement is added to zero. This result is the preliminary effective address (PEA) whereby the PEA=Displacement.

3. If the X field is all zeros, IA (bit 19) is a zero and I (bit 20) is a zero, then the 16-bit result of Step 2 is added to the contents of the updated instruction counter (IC) to form the 16-bit EA whereby EA=(updated) IC + PEA. (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section, with the exception that the Branch Sector Register (BSR) bits are used instead of the Data Sector Register (DSR bits).

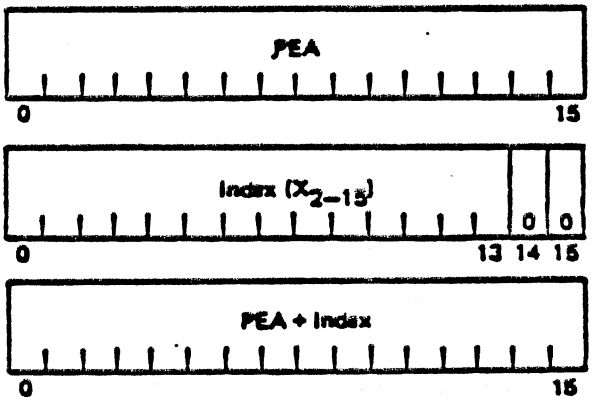
4. If the X field is all zeros, IA (bit 19) is a zero and I (bit 20) is a one, the 16-bit result of Step 2 is subtracted from the contents of the updated IC to form the 16-bit EA whereby EA = (updated) IC - PEA. (This



a. Halfword Index Alignment



b. Fullword Index Alignment



c. Double Word Index Alignment

Figure 2-13. Automatic Index Alignment

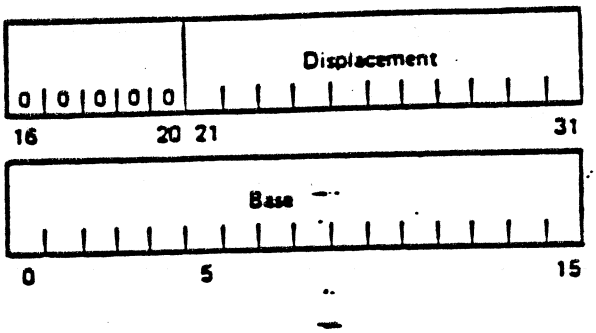
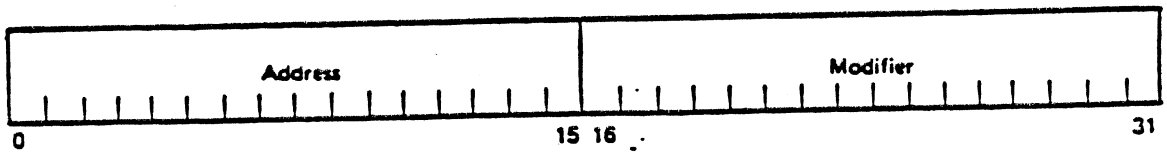


Figure 2-14. Displacement Alignment for Indexed Addressing

EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section with the exception that the Branch Sector Register (BSR) bits are used instead of the Data Sector Register (DSR) bits.)

5. If the X field is all zeros, IA (bit 19) is a one and I (bit 20) is a zero, then Indirect Addressing is performed. The 16-bit result of Step 2 is expanded to a 19-bit address and is used as the address of a main storage halfword. This halfword is then fetched and expanded to 19 bits by using expanded addressing to form the EA. EA=MS (PEA). Functional equivalency to preindexing capability can be obtained through modification of the base.

6. If the X field is all zeros, IA (bit 19) is a one and I (bit 20) is a one, Indirect Addressing is performed as described in Step 5 with a fullword main storage pointer. Then, after the EA has been formed, storage modification is automatically performed. The indirect address is contained in a fullword. A modifier is contained in bits 16 through 31. An address is contained in bits 0 through 15. The modifier is added to the address and the resulting modified address replaces bits 0 through 15 of the indirect address word (see Figure 2-15).



Modified Address = MS (PEA) ← MS (PEA) + MS (PEA + 1)

Figure 2-15. The Contents of Indirect Address Storage Modification Word

7. If the X field is not zeros, IA (bit 19) is a zero and I (bit 20) is a zero, the most significant 16 bits of the general register specified by the X field are aligned, and then added to the 16-bit result of Step 2 (PEA) to form the 16-bit EA (See Figure 2-13). (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section.)

If the X field is not all zeros, IA (bit 19) is a zero and I (bit 20) is a one, the most significant 16 bits of the general register specified by the X field are aligned, and then added to the 16-bit result of Step 2 (PEA) to form the 16-bit EA (see Figure 2-13). (This EA is then expanded to a 19-bit EA, as explained in the Expanded Addressing section.) (The modifier is added to the address and the resulting modified address replaces bits 0 through 15 of the index register after the EA is determined.) Figure 2-16 illustrates the address and modifier format in the index register.

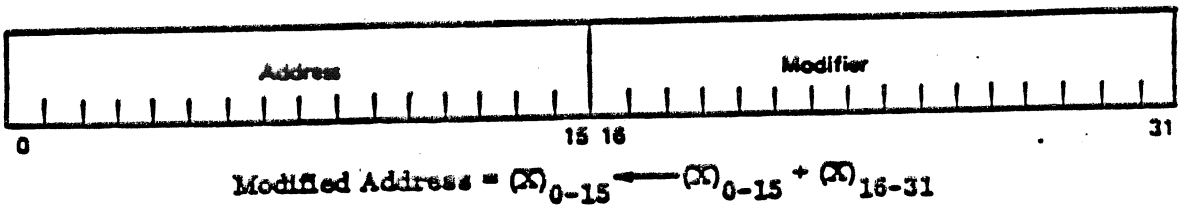
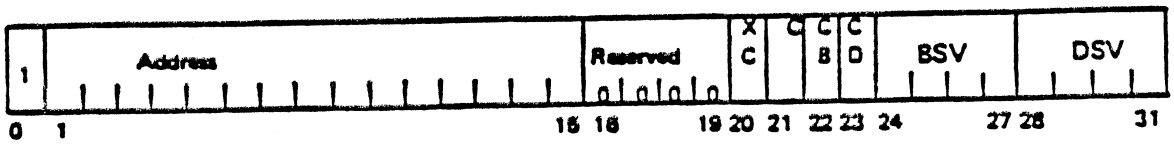


Figure 2-16. The Contents of Index Register X

9. If the X field is not all zeros, IA (bit 19) is a one and I (bit 20) is a zero, Indirect Addressing (IA) with postindexing is performed. The 16-bit result of Step 2 is expanded to a 19-bit address and is used to fetch a main storage halfword. The index contained in the general register specified by X is aligned and then added to the fetched halfword to form the 16-bit EA (see Figure 2-13). This EA is then expanded to a 19-bit EA by using expanded addressing. Functional equivalency to preindexing capability can be obtained through modification of the base.

10. If the X field is not all zeros, IA (bit 19) is a one and I (bit 20) is a one, an indirect addressing mode is defined using a 32-bit fullword indirect address pointer as follows:

a. First, the PEA from Step 2 must locate a fullword indirect address pointer, with the format as illustrated in Figure 2-17.



Field	Function
X_C	Index Control
C	Control to allow PSW modification
C_B	Control BSV Usage
C_D	Control DSV Usage
BSV (Branch Sector Vector)	Selectively replaces BSR in PSW
DSV (Data Sector Vector)	Selectively replaces DSR in PSW

Figure 2-17. Fullword Indirect Address Pointer

b. If C (bit 21) equals 0, XC (bit 20) equals 1, and the instruction is not a branch type instruction, the 19-bit EA equals the 4-bit DSV with the 15-bit address field appended. When C (bit 21) equals 0, XC (bit 20) equals 0, and the instruction is not a branch type instruction, the 19-bit EA equals the 15-bit address field added to the index value in indexing register X with the result appended to the DSV. The current PSW's DSR is not changed.

   If C (bit 21) equals 0 and the instruction is a branch type instruction, the current PSW's BSR in conjunction with bits 0 through 15 of the fullword indirect address pointer will be used to form the branch address (BA). If XC=0, postindexing will occur. When C (bit 21) equals zero, CB and CD are reserved and should be set to zero.

c. If C (bit 21) equals 1 and the instruction is a branch type instruction and the branch is taken, the BSV and DSV fields selectively replace the BSR and DSR fields in the current PSW, based on the CB and CD bit values as follows:

CB	CD	Result
0	0	Use current PSW's BSR to form the BA.
0	1	Replace the current PSW's DSR with the DSV. Form the BA normally.
1	0	Replace the current PSW's BSR with the BSV before forming the BA.
1	1	First, replace the current PSW's DSR with the DSV. Then, replace the current PSW's BSR with the BSV before forming the BA.

d. When C (bit 21) equals 1 and XC (bit 20) equals 1, postindexing is not performed. When C (bit 21) equals 1 and XC (bit 20) equals 0, the BA calculation includes a final addition of the index value in index registers X.

   If C (bit 21) equals 1, XC equals 1, and the instruction is not a branch, the 19-bit EA equals the current PSW's DSR and the 15-bit field appended. If XC=0, postindexing will occur.

The results of indexed mode RS operations normally replace the first operand except for store operation where the first operand replaces the second operand. The second operand is unaltered for nonstore operations, and the first operand is unaltered for store operation.

2.2.9 Expanded Addressing

The addressing philosophy accommodates 64K halfword addresses since a full 16-bit address is provided. Extending the addressing range beyond 64K halfword locations

... to 512K halfword locations is provided by utilizing PSW bits and Data Sector Extension (DSE) registers.

Expanding to 19 bits is achieved by replacing the high-order bit of a 16-bit address with 4 bits, as shown in Figure 2-18. Data operand addresses are extended to 19 bits ~~by specifying~~ with a 4-bit Data Sector Register (DSR), a DSE, or an implied DSR ~~(DSR=0)~~ of zero. When the high-order bit of a 16-bit data address is 1, a 4-bit DSR (PSW bits 28 through 31) is selected to replace the high-order bit. When the high-order bit of a 16-bit data register is 0 and a base register is used to determine the address, the 4-bit DSE for that base register is selected to replace the higher order bit. When the high order bit of a 16-bit data address is a 0, and no base register is used, an implied DSR containing 0000 is selected. Note that indirect addressing locates the indirect address pointer as if the pointer were a data operand. Second stage expansion of the indirect address pointer uses an implied DSR of zero if the high order bit of the 16-bit address is 0. Branch addresses are also extended to 19 bits. When the high-order bit of a 16-bit branch address is a 1, a 4-bit Branch Sector Register (BSR-PSW bits 24 through 27) is selected to replace the high-order bit. When the high-order bit is a 0, an implied BSR containing 0000 is selected.

~~*Second stage expansion of the indirect address pointer uses an implied DSR of zero if the high order bit of the 16 bit address is 0.~~ AND NO BASE REGISTER IS USED. IF THE HIGH ORDER BIT OF THE 16 BIT ADDRESS IS Ø AND A BASE REGISTER IS USED, THEN THE 4-BIT DSE FOR THAT BASE REGISTER IS SELECTED TO REPLACE THE HIGH ORDER BIT

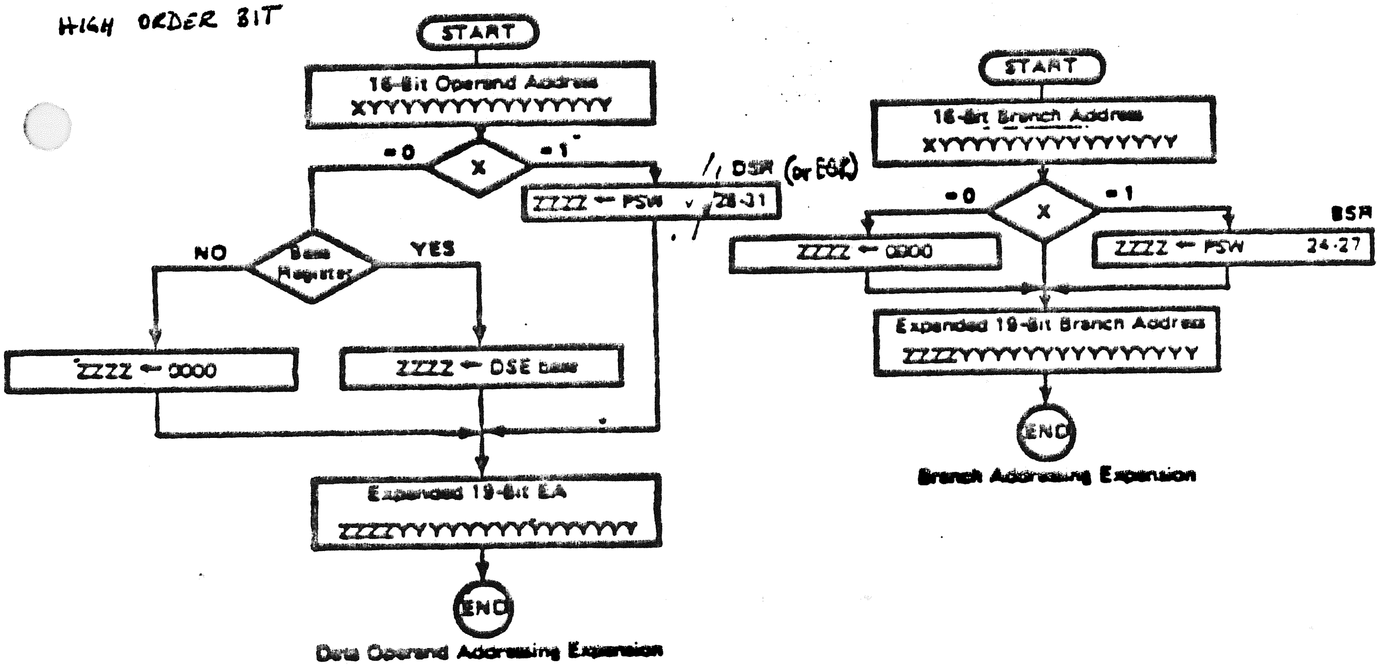
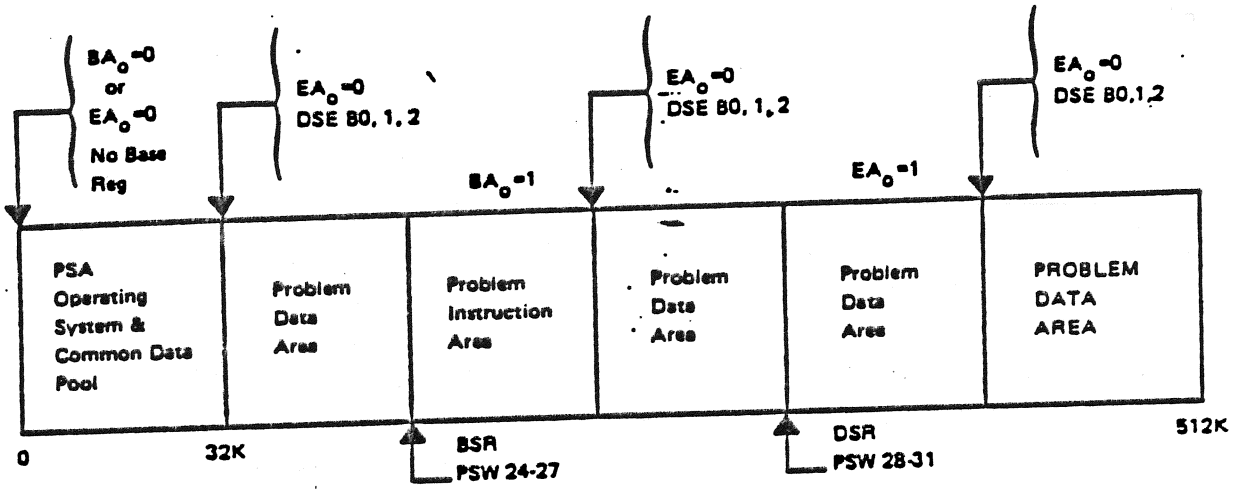


Figure 2-18. Expanded Addressing

Pictorially, main storage can be visualized as follows:



This permits efficient communication from the problem program to the operating system, the preferred storage area, (PSA) or a common data area.

It should be cautioned that instruction address incrementing or address calculations used to form the EA are performed on the low 16 bits only, and will not alter the BSR, DSR, or DSE. The BSR or DSR may be altered only via a PSW swap, special instruction operations (SVC, LPS) or by use of the indirect address pointer described in this section. The DSE registers are loaded by the LXA and LDM instructions.

2.3 PROGRAM EXECUTION

The CPU program consists of instruction and control words specifying the operations to be performed. This information resides in main storage and addressable registers and may be operated on as data. Instruction execution control is as defined under the section on Machine Status and General System Operation. Insert Storage Protect Bits, Load Program Status, Internal Control and Set System Mask instructions are privileged instructions and can only be executed in the Supervisor State. The Program Status Word determines the current state of the CPU and the Supervisor Call instruction can be used by the problem program to enter Supervisor State.

2.4 STORAGE PROTECTION FEATURES

The storage protection feature prevents modification of specific main storage locations. Any location which could, for example, contain constant data or program instructions can be selectively protected from Store operations without restricting the use of other areas. Traps on store operations to specific data words can be inserted during program checkout. A privileged instruction, Insert Storage Protect Bits, is provided to set/reset the protection bits associated with each halfword of

main storage. Attempting to store data in a protected location 90035 C43
will result in a

program interrupt ~~unless it is previously masked by setting the machine check mask (bit 45) to zero.~~ In this case, the store operation does not occur.

2.4.1 Instruction Monitor Feature

The storage protection bits described can also be used to flag an inadvertent attempt to execute, as instructions, data stored in unprotected areas. The feature will ensure that no program will continue to execute data as program instructions. An attempt to execute an instruction word which is unprotected will result in an interrupt if PSW bit 34 is a one. The feature can be masked by a System Mask Bit (bit 34 of the PSW). During program checkout, this feature permits use of special software to aid debugging.

An Instruction Monitor difference is the state the effective address is left in following the interrupt handling. In the AP-101B, the Instruction Counter is incremented to point to the next instruction to be executed. The AP-101S Instruction Counter is not incremented and is left pointing to the offending instruction.

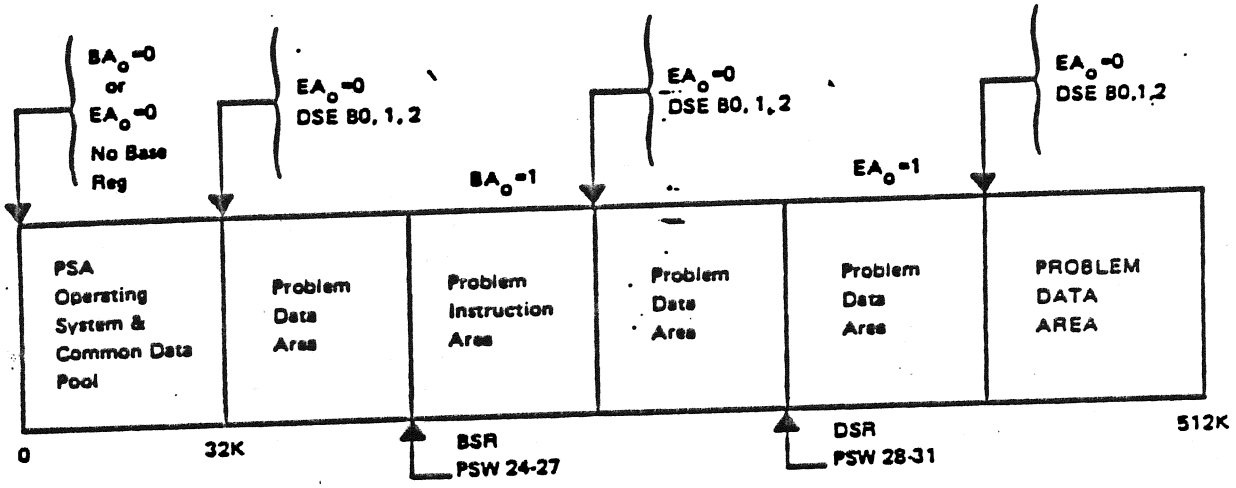
2.5 MACHINE STATUS

System status can be altered by the occurrence of interrupts and by the program. A doubleword register within the CPU contains a program status word (PSW) and is the focal point for CPU and system status conditions.

2.5.1 Program Status Word

The program status word (PSW), contains the basic information required for proper program execution. The 64-bit PSW includes the next instruction address, the current condition code, the carry and overflow indicators, the system mask for interrupts, and other fields significant to CPU operations. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. The active or controlling PSW is called the "current PSW". By storing the current PSW during an interruption, the status of the CPU can be preserved for subsequent use. By loading a new PSW or part of a PSW, the state of the CPU can be initialized or changed. Figure 2-19 shows the PSW format.

Pictorially, main storage can be visualized as follows:



This permits efficient communication from the problem program to the operating system, the preferred storage area, (PSA) or a common data area.

It should be cautioned that instruction address incrementing or address calculations used to form the EA are performed on the low 16 bits only, and will not alter the BSR, DSR, or DSE. The BSR or DSR may be altered only via a PSW swap, special instruction operations (SVC, LPS) or by use of the indirect address pointer described in this section. The DSE registers are loaded by the LXA and LDM instructions.

2.3 PROGRAM EXECUTION

The CPU program consists of instruction and control words specifying the operations to be performed. This information resides in main storage and addressable registers and may be operated on as data. Instruction execution control is as defined under the section on Machine Status and General System Operation. Insert Storage Protect Bits, Load Program Status, Internal Control and Set System Mask instructions are privileged instructions and can only be executed in the Supervisor State. The Program Status Word determines the current state of the CPU and the Supervisor Call instruction can be used by the problem program to enter Supervisor State.

2.4 STORAGE PROTECTION FEATURES

The storage protection feature prevents modification of specific main storage locations. Any location which could, for example, contain constant data or program instructions can be selectively protected from Store operations without restricting the use of other areas. Traps on store operations to specific data words can be inserted during program checkout. A privileged instruction, Insert Storage Protect Bits, is provided to set/reset the protection bits associated with each halfword of

main storage. Attempting to store data in a protected location will result in a 90035 C43

program interrupt. ~~unless it is previously masked by setting the machine check mask (... bit 45) to zero.~~ In this case, the store operation does not occur.

2.4.1 Instruction Monitor Feature

The storage protection bits described can also be used to flag an inadvertent attempt to execute, as instructions, data stored in unprotected areas. The feature will ensure that no program will continue to execute data as program instructions. An attempt to execute an instruction word which is unprotected will result in an interrupt if PSW bit 34 is a one. The feature can be masked by a System Mask Bit (bit 34 of the PSW). During program checkout, this feature permits use of special software to aid debugging.

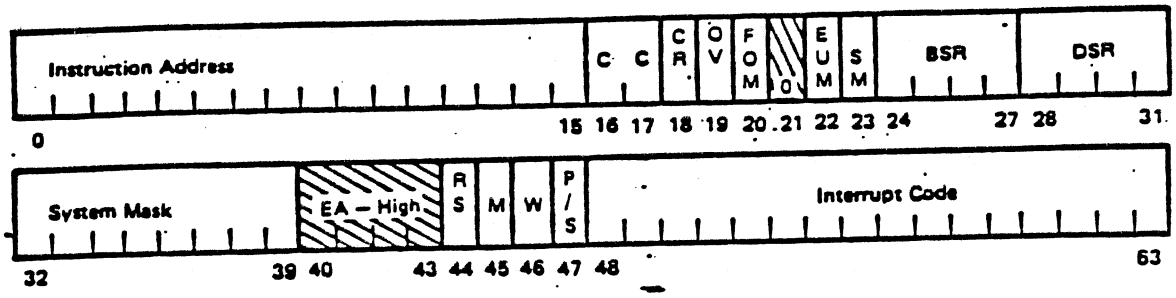
An Instruction Monitor difference is the state the effective address is left in following the interrupt handling. In the AP-101B, the Instruction Counter is incremented to point to the next instruction to be executed. The AP-101S Instruction Counter is not incremented and is left pointing to the offending instruction.

2.5 MACHINE STATUS

System status can be altered by the occurrence of interrupts and by the program. A doubleword register within the CPU contains a program status word (PSW) and is the focal point for CPU and system status conditions.

2.5.1 Program Status Word

The program status word (PSW), contains the basic information required for proper program execution. The 64-bit PSW includes the next instruction address, the current condition code, the carry and overflow indicators, the system mask for interrupts, and other fields significant to CPU operations. In general, the PSW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. The active or controlling PSW is called the "current PSW". By storing the current PSW during an interruption, the status of the CPU can be preserved for subsequent use. By loading a new PSW or part of a PSW, the state of the CPU can be initialized or changed. Figure 2-19 shows the PSW format.



0-15	Next Instruction Address	
16-17	Condition Code	
18	Carry Indicator	
19	Overflow Indicator	
20	Fixed-Point Arithmetic Overflow Mask*	
21	Reserved	
22	Floating Point Exponent Underflow Mask*	
23	Significance Mask*	
24-27	Branch Sector Register	
28-31	Data Sector Register	
32	Counter 1 Mask	System* Mask
33	Counter 2 Mask	
34	Instruction Monitor Mask	
35	External Interrupt 0 Mask	
36	External Interrupt 1 Mask	System* Mask
37	External Interrupt 2 Mask	
38	External Interrupt 3 Mask	
39	External Interrupt 4 Mask	
40-43	Reserved for SVC High Order EA Bits	
44	Register Set (GR set 0 or 1)	
45	Machine Check Mask*	
46	Wait State Bit (Wait/Process)***	
47	Problem/Supervisor State Control Bit**	
48-63	Interrupt Code for Program Check, Machine Check, and Special External Interrupts, or 16 Bit Operand PEA for SVC Instruction	

*Mask bit = 0, interrupt inhibited
= 1, interrupt allowed
**0 = supervisor state
1 = problem state
***0 = process state
1 = wait state

Figure 2-19. PSW Fields

The overall status of the CPU is preserved in the current PSW and the contents of the general registers. The PSW is automatically retained upon taking an interrupt. It is the programmer's responsibility to preserve the contents of the general registers when necessary.

Certain other conditions that contribute to an overall system status situation are not automatically preserved when a CPU is interrupted. There conditions involve additional units and include the dynamic state of all other interrupts, the state of real time counters, and I/O system status.

Masking is accomplished by setting the appropriate PSW bit to zero.

2.5.1.1 PSW Fields

The PSW fields (Figure 2-19) are defined as follows:

1. Instruction Address - Bits 0 through 15 and 24 through 27 of the PSW contain the information to determine the address of the next instruction to be executed. The machine architecture makes provision to address 262,144 fullwords, and the AP-101S space shuttle hardware implementation provides full addressing capability.

2. CPU Status

Bit	Use
16, 17	Condition code for certain arithmetic, logical and I/O instructions
18	Carry status bit indicator
19	Overflow status bit indicator (overflow can be reset by testing or by loading the PSW)
20	Fixed Point Arithmetic Overflow Mask
21	Reserved
22	Floating Point Exponent Underflow Mask
23	Significance Mask

3. Branch Sector Register - Bits 24 through 27 replace the high-order bit of a branch address when that bit is a 1. Otherwise, an implied sector register of 0000 replaces the high-order bit.

4. Data Sector Register - Bits 28 through 31 replace the high-order bit of a data address when that bit is a 1. See "Expanded Addressing" for details when bit 0 is a zero.

5. System Mask - Bits 32 through 39 are mask bits. The first two bits of the System Mask are normally assigned to the two counters and the third to the instruction Monitor Feature. The remaining five masks include I/O end conditions, other application dependent items such as a manual interrupt key, and timer overflow conditions. The instruction SET SYSTEM MASK is provided for modifying this field.

6. EA-High - For an SVC instruction, the 4-bit extension to make the 19-bit effective address is saved in the old PSW bits 40-43.

7. Register Select Field - The register select field, bit 44, controls either of two sets of general registers in current use. When this bit is a zero, then register set 0 is used; when this bit is one, then register set 1 is used. The set of general registers in current use can be selected when a new PSW is loaded. This can result from the execution of the PSW load instruction or from an interrupt.

8. Machine Check Mask - Bit 45 is the mask bit which is used to inhibit machine check interrupts (see Figure 2-20). When this bit is a zero, then machine check interrupts detected by the CPU are inhibited.

Interrupt Priority	Class	Old PSW	New PSW	Not Maskable	PSW Mask Bit	Pending	Int. Code	Interrupt Accept Time	CPU/IOP/AGE Generates	Interrupt
00	POWER	0010	—	X	—	—	N/A	ENDOP	CPU	Power Off****** (Microcode Put Away)
01	POWER	—	0004	X*	—	—	N/A	MCYCLE	CPU	Power On
02	POWER	—	0014	X**	—	—	N/A	MCYCLE	CPU	System Reset
03	POWER	—	—	—	—	—	N/A	—	—	N/A to Shuttle ISA
C0	MC	0040***	0044	—	45	No	0008	MCYCLE	CPU	EA Fault
04	MC	0040***	0044	—	45	No	0005	MCYCLE	CPU	CPU Microstore Parity
05	MC	0040	0044	—	45	No	0006	ENDOP	CPU	Interrupt Page Fault
35	MC	0040	0044	—	45	No	0002	Forced ENDOP	IOP	DMA Memory Multi-bit Error
06	MC	0040 †	0044	—	45	No	0003	Forced ENDOP	CPU	CPU Memory Multi-bit Error
10	MC	—	—	—	—	—	—	—	—	Spare
11	MC	—	—	—	—	—	—	—	—	Spare
12	MC	0040***	0044	X	—	—	0007	MCYCLE	CPU	ENDOP Timeout
13	MC	—	—	—	—	—	—	—	—	Spare
14	MC	0040***	0044	X	—	No	0009	MCYCLE	CPU	CPU Cannot Continue
15	MC	—	—	—	—	—	—	—	—	Reserved
16	MC	—	—	—	—	—	—	ENDOP	AGE	AGE Breakpoint (Tester Service)
30	MC	—	—	X	—	—	—	—	—	N/A to Shuttle ISA
36	MC	—	—	—	—	—	—	—	—	IU Memory Error*****
37	MC	—	—	—	—	—	—	—	—	EU Memory Error*****
17	PE	0070	0074	—	34	No	N/A	ENDOP	CPU	CPU Breakpoint (Instruction Monitor)
20	PE	0048	004C	—	20	Note 1	0004	ENDOP	CPU	Fixed Point Overflow
21	PE	0048	004C	X	—	No	0008	Forced ENDOP	CPU	Floating Point Overflow (Exponent)
22	PE	0048	004C	—	22	No	0009	Forced ENDOP	CPU	Floating Point Underflow
23	PE	—	—	—	—	—	—	—	—	Spare
C1, 34	PE	0048	004C	X	—	No	0000	MCYCLE	CPU	Illegal Inst. or I/O command
C2	PE	0048	004C	X****	—	No	0001	ENDOP	CPU	Privileged Instruction ~~(Maskable Only in Problem State, PSW 47=1)~~
C3	PE	0048	004C	X	—	No	000C	Forced ENDOP	CPU	Divided by Zero (Flt. Pt.)
C4	PE	0048	004C	—	23	No	0005	Forced ENDOP	CPU	Significance
C5	PE	0048	004C	X	—	No	000A	ENDOP	CPU	Convert Overflow
31	PE	0048	004C	X	—	No	0002	Forced ENDOP	CPU	CPU Addr Spec 128K, CB Only
P0-	SC	0058	005C	X	—	No	(Inst)	ENDOP	CPU	Supervisor Call
31	PE	—	—	—	—	—	—	—	—	Spare
32	PE	—	—	—	—	—	—	—	—	N/A to Shuttle ISA
33	PE	0048	004C	—	[illegible]	—	0007	Forced ENDOP	CPU	Store Protect Violation
07	PE	—	—	—	—	—	—	—	—	N/A to Shuttle ISA
40-43	SYS	—	—	—	—	—	—	—	—	N/A to Shuttle ISA
44	SYS	—	—	—	—	—	—	—	—	Spare
45	SYS	0060	0064	—	32	Yes	—	ENDOP	CPU	Interval Timer No. 1
46	SYS	0068	006C	—	33	Yes	—	ENDOP	CPU	Interval Timer No. 2
47	SYS	—	—	—	—	—	—	—	—	N/A to Shuttle ISA
50	SYS	0078	007C	—	35	Yes	0000	ENDOP	IOP	External 0 (IOP Voter, IOP Req. A)
50	SYS	0078	007C	—	35	Yes	0000	ENDOP	IOP	External 0 (C/M Idle, IOP Req. A)
50	SYS	0078	007C	—	35	Yes	0000	ENDOP	IOP	External 0 (IOP ROS Parity, IOP Req. A)
50	SYS	0078	007C	—	35	Yes	0000	ENDOP	IOP	External 0 (IOP Fault, IOP Req. A)
50	SYS	0078	007C	—	35	Yes	0000	ENDOP	IOP	External 0 (Watchdog Timer, IOP Req. A)
51	SYS	0080	0084	—	36	Yes	0000	ENDOP	IOP	EXT 1 IOP Data Flow Error Encode (see Read Interrupt key 8 in Section I)
51	SYS	0080	0084	—	36	Yes	0000	ENDOP	IOP	Ext 1 Q Overflow (IOP Req. B)
51	SYS	0080	0084	—	36	Yes	0000	ENDOP	IOP	Ext 1 DMA Timeout (IOP Req. B)
51	SYS	0080	0084	—	36	Yes	0004	ENDOP	CPU	Ext 1 DMA Store Protect Violation
53	SYS	0088	008C	—	37	Yes	—	ENDOP	IOP	Ext 2 IOP Programmed Interrupts (1-12)
54	SYS	0090	0094	—	38	Yes	—	ENDOP	IOP	Spare External 3
55	SYS	0098	009C	—	39	Yes	—	ENDOP	IOP	Spare External 4
~~55~~	~~SYS~~	~~00A0~~	~~00A4~~						~~IOP~~	~~Spare~~
56	SYS	00A8	00AC	—	—	—	—	—	~~IOP~~	Spare
52	SYS	0080	0084	—	36	Yes	0006	ENDOP	AGE	Shuttle AGE Interrupt

* CPU must not be in the halt mode
** CPU must be in halt mode
*** PSW can vary, maybe updated PC or unupdated PC
**** Only occurs when in problem state
***** Valid only during execution in Diagnose Instruction
****** If power off during long instruction, IC may be backed up
(INST) 16 Bit Operand PEA of SVC instruction
Note 1 Status held active in PSW 19

† see note in paragraph 2.5.2.1

Figure 2-20. Interrupt Structure and Priority

9. Wait State - Bit 46 determines the wait or processing (run) states. When this bit is a zero, the CPU is in the processing state. When this bit is a one, the CPU is in the Wait State.

10. Problem/Supervisor - Bit 47 determines the problem or supervisor states. When this bit is a zero, the CPU is in the supervisor state and privileged instructions can be executed. When this bit is a one, the CPU is in the problem state and attempts to execute privileged instructions are inhibited resulting in an interrupt.

11. Bits 48 through 63 are reserved for the interrupt code. Program and machine check interrupt conditions and associated interrupt codes are given in Figure 2-20.

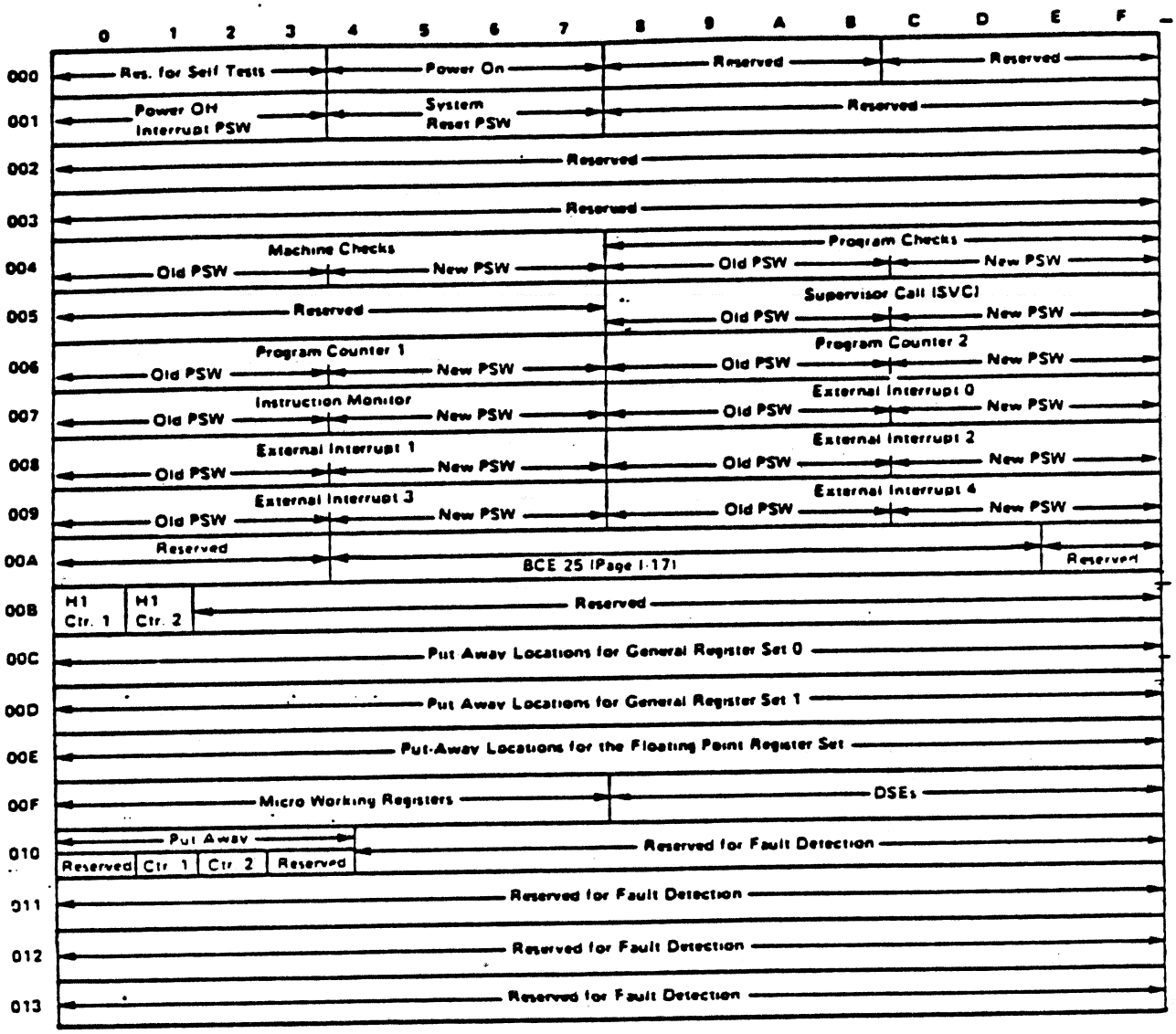
2.5.2 Interrupts

1. Power - This interrupt occurs when primary power is removed from the system for any reason. The current PSW, the general register set 1 and 2, the floating point registers, counters 1 and 2, and the current DSEs are put away (stored) in main storage for future reference. Figure 2-21 shows the PSA assignments including putaway. When primary power is restored, operation is initiated with the "power on PSW" (if the power-up mode is defined as Run). This power-up condition is explained in General System Operation.

2. Machine Check - When not masked, this interrupt class occurs following the detection of a malfunction. The current instruction is then terminated and the interrupt taken. A diagnostic procedure may then be initiated. When masked the interrupt does not remain pending.

3. Program - This class of interrupt arises from improper specification or use of instructions or data. Bits 20, 22, and 23 (1=interrupt enabled, 0=interrupt disabled) in the PSW are provided to permit masking program interrupts due to arithmetic exceptions such as fixed point overflow. Bit 34 in the PSW is provided to permit masking the instruction monitor interrupt. ~~Bit 45 of the PSW (Machine Check Mask) masks a store protection violation.~~ When masked, program interrupts do not remain pending. When invalid instruction or address detection is provided, the resulting program interrupts cannot be masked.

4. Supervisor Call (SVC) - This interrupt results from the execution of the SVC instruction. The four MSBs of the 19-bit extended EA are placed into the EA-high field (bits 40-43) of the old PSW, and the nonextended 16-bit EA is placed into the interrupt code (bits 48-63) of the old PSW. This instruction can be used to switch from the problem to the supervisor state.



DSE PUTAWAY FORMAT

ADDR	REGISTER SET 0				REGISTER SET 1			
00F8	RESV	DSE0	RESV	DSE1	RESV	DSE0	RESV	DSE1
00FA	RESV	DSE2	RESV	DSE3	RESV	DSE2	RESV	DSE3
00FC	RESV	DSE4	RESV	DSE5	RESV	DSE4	RESV	DSE5
00FE	RESV	DSE6	RESV	DSE7	RESV	DSE6	RSEV	DSE7
BITS	0 3	4 7	8 11	12 15	16 19	20 23	24 27	28 31

Figure 2-21. Preferred Storage Area Assignments

5. System - This class of interrupt results from program counter timeouts and conditions outside the CPU. Provision is made for seven interrupt levels within this class, and each is provided with a unique set of PSWs and a mask bit. Two are program counters and five are external interrupts.

   Any number of the five external interrupt conditions may be grouped into a single level by the external equipment. In the event of simultaneous external interrupt conditions, the lowest numbered (bit within the system mask field in the PSW) interrupt is taken first. These interrupts remain pending when masked.

The two program interval timers are each 32 bits wide and decrement. The lower 16 bits (least significant halfword) of each counter resides in 16-bit binary hardware counters that count down by one every microsecond. The high 16 bits (most significant halfword) of each counter resides in main store. The high halfword lies in main store location 0080 for counter 1 and main store location 0081 for counter 2. When the low halfword (in the hardware counter) passes from 0000 (hex) to FFFF (hex) an interrupt occurs which can cause the high halfword in main store (via microcode) to be decremented by one. This interrupt is transparent to the programmer until the high halfword in main store equals 0000 (hex). When such an interrupt occurs, the high halfword is decremented to FFFF (hex) and a PSW swap occurs, telling the programmer that the counter has timed out. Note that if the interrupt is masked the high halfword will not be decremented by the microcode. The low halfword continues to count down. The interrupt although, remains pending and if unmasked within 65 ms, the upper halfword will be decremented without a loss of a count.

The counters can be loaded and read by the Internal Control instruction, described in Section 10.

2.5.2.1 Interrupt Handling

The machine check, program, SVC, and each system interrupt have two related PSWs called "old" and "new" in unique main store locations. This zone of main store is referred to as a preferred storage area (PSA), which is illustrated in Figure 2-21.

In all cases, an interruption involves merely storing the current PSW in its old position and making the PSW at the new position the current PSW. The old PSW holds all the necessary status information in the system existing at time of interruption. If, at the conclusion of the interruption routine, there is an instruction to make the old PSW the current PSW, the system is restored to the state prior to the interruption, and the interrupted routine continues. This means the programmer must clear the fixed point overflow indicator before being reloaded. Note that it is possible to switch to the alternate set of general registers when the PSW swap takes place. This set of registers is defined by bit 44 in the new PSW.

Interruptions can only be taken when the CPU is interruptible for a given source. The system mask, machine check mask bit, floating point exponent underflow mask, the significance mask, and the fixed point overflow mask bits in the PSW define the interruptible state of the CPU with respect to those sources. When masked, system interrupts remain pending while machine check and program interrupts are ignored.

[redacted]

The power transient, certain program interrupts, and the SVC interrupt cannot be masked.

CPU multibit error PSW note (A)

2.5.2.2 Interrupt Priority

Figure 2-20 presents the repertoire of interrupts with approximate priority levels. Individual interrupts are listed in order by classification, rather than by priority. The priority of each interrupt is represented by a two-digit code, which is interpreted as follows:

First Digit - represents the capture latch number (lower-numbered capture latches are examined first) or, if alphabetic, the fact that the interrupt is generated by the CPU - either a Command Interrupt (C), or a Supervisor Call PSW swap (P).

Second Digit - represents the priority of the interrupt within a grouping (hardware or "other").

Conceptually, the order of processing (in the case of interrupts received simultaneously) is as follows:

1. Group 0 Interrupts - ~~If any of the interrupts in this group are received, the Interrupt Page processor is reset and all other pending or queued interrupts are lost.~~ (B)

2. Command Interrupts - These are usually interrupts which demand direct communication from the CPU to the Interrupt Page Processor. Often, they are included within a CPU microcode procedure. Action taken by the CPU is usually to request the interrupt and then loop at one microword, waiting for the Interrupt Page to reset the Control Store Data Register, thereby forcing a branch to zero.

3. Group 1, 2, or 3 Interrupts - These interrupts differ from the following two groups in that the hardware freezes the CPU microcode at the next ENDOP when one of them is detected.

4. Group 4 or 5 Interrupts - These interrupts are the only types that are held pending until they are unmasked with no additional higher-priority interrupts present. They are only accepted at ENDOP time and generally cause only slight CPU processing delays if they are masked OFF.

When more than one unmasked interrupt requests service, the current (old) PSW is stored into and the new PSW is fetched from two PSA locations assigned to the first interrupt to be processed. Then, the same procedure is followed using the PSA locations of the second interrupt, with the exception that the "old" PSW is the former new PSW as fetched for the first interrupt. This procedure of "passing" the PSW is continued until the last interrupt request is acknowledged. Then, instruction execution is commenced using the PSW last fetched. The order of execution of the interrupt service routines is, consequently, the reverse of the order in which the string of "new" PSWs were fetched. Machine Check and Power Transient interruptions supersede all other interrupts when they are encountered.

A

NOTE ON CPU MULTI BIT ERROR OLD PSW

NOTE: THE PIPELINE IS THE DRIVER FOR CPU MULTI BIT ERRORS (IU & EA) THEREFORE, THE MACHINE CHECK OLD PSW FOR CPU MULTI BIT ERROR WILL REFLECT THE UPDATED PC - NOT THE ADDRESS OF THE MULTI BIT ERROR. THE FOLLOWING ARE THE WAYS IN WHICH A CPU MULTI BIT ERROR MAY BE ENCOUNTERED:

1) THE INSTRUCTION UNIT (IU) PREFETCHING INSTRUCTIONS (UP TO 23 HALFWORDS AHEAD OF THE PC)

2) THE EFFECTIVE ADDRESS UNIT (EA) PREFETCHING DATA (ANYWHERE IN MEMORY)

3) THE EA PREFETCHING A BRANCH TARGET ADDRESS (ANYWHERE IN MEMORY)

IN THE EVENT OF THIS TYPE ERROR, THE ERROR DETECTION AND CORRECTION (EDAC) REGISTER MAY BE READ FOR DETERMINATION OF THE ACTUAL MULTI BIT ERROR ADDRESS.

B

REWRITE GROUP 0 INTERRUPTS SECTION AS FOLLOWS:
"GROUP 0 INTERRUPTS - THESE ARE THE HIGHEST PRIORITY - THE POWER/MACHINE CHECK TYPE INTERRUPTS. THE POWER, SYSTEM RESET, AND IPL INTERRUPTS CLEAR ALL PENDING INTERRUPTS - THE REMAINING GROUP 0 INTERRUPTS DO NOT. SEE PAGE 2-21 FOR INTERRUPT STRUCTURE AND PRIORITY.

The priority scheme as outlined above is used to resolve race conditions due to multiple interrupt conditions. However, since in the case of most normal interrupts (those expected to be encountered during the execution of typical application software) separate mask bits and PSW locations are provided for each external source, the priority of handling these interrupts is further affected by the contents of the PSWs actually fetched during the interrupt service overhead. That is, as each PSW swap occurs, further action with regard to System (and Machine Check) interrupts is determined by the mask fields encountered within the new PSW.

Two major exceptions to the above process involve the Instruction Monitor Interrupt and Supervisor Call. Instruction Monitor conditions are monitored by hardware and cause no processing delays if masked OFF, since the Interrupt Page will not even be notified of the condition in that event. It could be argued that Supervisor Call might not be considered an interrupt at all, since it is not an unexpected condition and is appropriately handled by the CPU microcode, but it is included in the list because its execution necessitates a PSW SWAP and, therefore, cooperation by the Interrupt Page processor in that portion of the instruction implementation.

2.5.2.3 Interrupt Masking

Individual masking of several of the interrupt types is possible. When masked off, the interruption is either ignored or remains pending for later execution. The masking capability for each of the interrupt types is as follows:

1. Power Transient - Cannot be masked off.

2. Machine Check - Can be masked off by setting the machine check mask bit 45 in the PSW equal to zero. When masked off, normal instruction sequencing occurs, and the interrupts do not remain pending.

3. Program - Three of the 11 program interrupts are capable of being masked off; fixed point arithmetic overflow, exponent underflow, and significance, by setting the appropriate mask bits in the PSW equal to zero. When masked off, these interruptions do not remain pending. ~~Also, the storage protect interrupt can be masked via the machine check mask (PSW bit 45).~~ Note that if a PSW with both Fixed Point Overflow Indicator and mask (bits 19 and 20) set is used, the interrupt will occur.

4. Supervisor Call - Cannot be masked off.

5. System - Each level of external interrupts can individually be masked off by setting the corresponding system mask bit in the PSW equal to zero. Interrupts that are masked remain pending.

2.5.2.4 Preferred Storage Area (PSA) Assignments

The contents of the PSA are shown in Figure 2-21 with the main store address expressed in hexadecimal notation. The following PSA locations must not be store protected:

1. Power off interrupt PSW
2. All old PSW locations
3. ~~3CE~~ 2C Processor Storage (00A4-00A5)
4. ~~3.~~ Counter 1 and 2, high halfword locations 00B0 and 00B1
5. ~~4.~~ Putaway locations (00C0 through 0102)
6. ~~5.~~ Diagnostics (104-13F).

2.5.3 General System Operation

The various states entered by the computer and their relationship to the basic operator controls are shown in Figure 2-22. The basic controls provided for the operator are power-on, initial program load (IPL) and the system reset key. Among the many controls available, these functions have special significance because of their relationship to an unconditional system reset sequence. These functions each produce a system reset sequence which applies to the computer, I/O channels, and peripherals. Further operation within the system differs, however, as explained in the following sections.

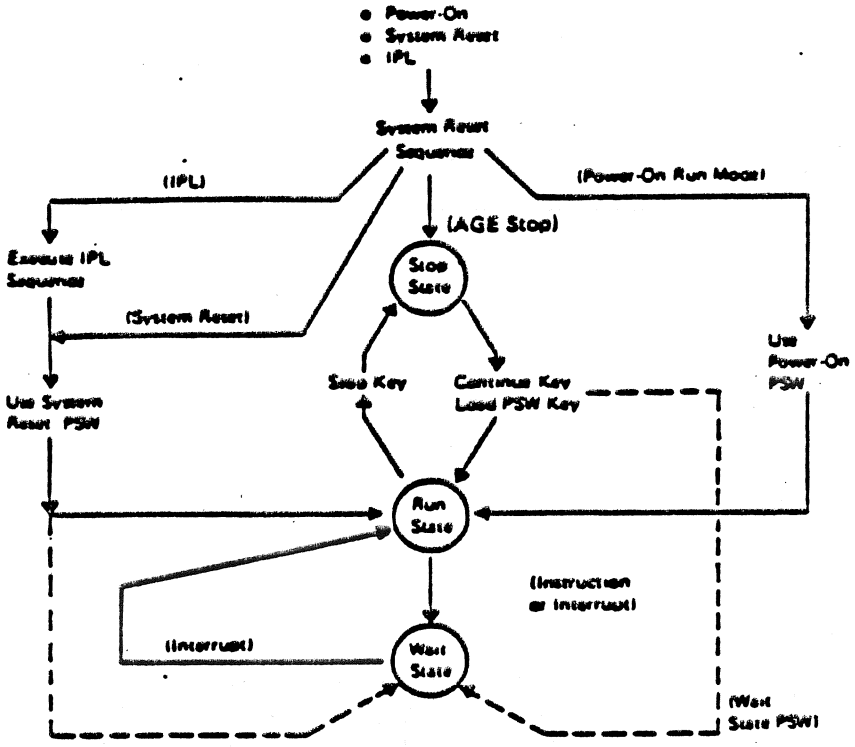


Figure 2-22. CPU Mode Switching

2.5.3.1 Power-On

One of two modes of operation must be specified for the system at power-on. The first results in a system reset followed by the computer entering the stop state. In this state, instructions are not processed, interrupts are not accepted, and system timers are not updated. This system is termed "manual" because further operation must be determined by the operator.

The second mode at power-on enters the run state after the system reset is complete. The instruction stream is initiated and interrupts are processed. The computer can be removed from the run state by certain instructions, interruptions, and by manual intervention.

2.5.3.2 System Reset

The system reset function resets the computer system to a known state such that processing can be initiated without the presence of machine checks, except for those used by subsequent machine malfunctions. The system reset function causes the following:

- CPU pending interrupts are reset
- Internal timers are reset to all ones (1's)
- Status registers are reset
- DSE registers are set to zero.

2.5.3.3 IPL

The use of the IPL function is independent of the prior state of the system. IPL first causes a system reset function and the writing of C6C6 (hex) by the CPU to all memory locations above and including address 20000 Hex with memory store protected. IOP microcode at IPL writes C9FB (hex) to all locations from 0 to 1FFFF Hex, with memory store protected.

2.5.4 Operating State

The run state and wait state shown in Figure 2-22 are collectively termed the operating state for the system. When the computer is in the run state, instructions are executed in the normal manner. An instruction may be encountered or an interrupt processed that forces the computer into the wait state. The computer does not execute instructions in the wait state, but it is interruptible when not masked. System timers are updated and input/output operations continue in the wait state.

The wait state may also be entered after completing IPL or by special operating intervention via the stop state (dotted lines on Figure 2-22). This action is the result of the wait bit being set in the controlling PSW.

2.5.4.1 Program State Alternatives

Certain other states exist within the CPU that contribute to its overall status. These states are directly related to program operation and are:

1. Masked or Interruptible State - The computer may be masked for certain interrupt conditions at any given time. These conditions generally remain pending within the system until the masked condition is changed by the program. Certain error conditions cannot be masked off, while other error conditions, such as program checks, are ignored when specifically masked.

2. Supervisor or Problem State - In the supervisor state, all instructions are valid. In the problem state, I/O and certain other instructions are invalid, and their use produces an error interrupt. This state is controlled by bit 47 in the PSW. The SVC instruction is provided to switch from problem to supervisor state. The LOAD PSW instruction is used

to switch from supervisor to problem state.

3. General Register Selection - Bit 44 is the current PSW and selects the set of general registers in current use.

2.5.5 Architectural Growth

Throughout this Principles of Operation manual, architecture conventions are defined or facilities are marked "reserved" to retain flexibility for future implementations and extensions. The computer operates in conformance to this manual when architecture definitions are followed consistently. Hardware operation, when these rules are violated, is not defined and is properly outside the scope of this manual to retain flexibility of implementation. "Programmer discovered" operations that violate or go beyond the definitions described herein, but produce "useful" functions, should not be used and should be considered "reserved", because the results obtained may vary from computer to computer, or even release levels for one computer, depending upon options selected or the design release level to which the hardware is manufactured.

11.0 AP-101S SHUTTLE INSTRUCTION SET

11.1 EFFECTIVE ADDRESS GENERATION SUMMARY CHART

	SRS, SI Formats	RS Format				
		Extended Addressing (AM=0)	Indexed Addressing (AM=1)			
			LA	I	X=000	X=$\overline{000}$
					PEA=(B)+DISP	
B2≠11	EA=(B)+DISP	EA=(B)+DISP	00		EA=IC+PEA	EA=$(X)_{0-15}$+PEA
			01		EA=IC-PEA	EA=$(X)^{*}_{0-15}$+PEA
			10		EA=MS(PEA)	EA=MS(PEA)+$(X)_{0-15}$
			11		EA=MS(PEA)**	EA=MS(PEA)***+$(X)_{0-15}$
					PEA=DISP	
B2=11	EA=(B)+DISP	EA=DISP	00		EA=IC+PEA	EA=$(X)_{0-15}$+PEA
			01		EA=IC-PEA	EA=$(X)^{*}_{0-15}$+PEA
			10		EA=MS(PEA)	EA=MS(PEA)+$(X)_{0-15}$
			11		EA=MS(PEA)**	EA=MS(PEA)***+$(X)_{0-15}$

Definitions

EA	Effective address, main storage address of second operand
PEA	Preliminary effective address
(RN)	Contents of bits 0-15 of general register N specified by B2 or X
RN	General register "N", where N = 0 to 7
(B)	Contents of bits 0-15 of general register specified by the B2 field
B2	B field of SRS, SI, or RS format instruction
MS( )	Contents of the main storage location specified by the contents of the parenthesis
DISP	Displacement field of instruction
X	X field of RS format instruction with indexed mode of addressing
$(X)_{0-15}$	Most significant halfword (bits 0-15) of the content of index register X automatically aligned.
AM	AM (addressing mode) field of RS format instruction
LA	LA (indirect address) field of RS format instruction with the indexed mode of addressing
I	I field of RS format instruction with indexed mode of addressing
IC	Updated Instruction Counter
*	Automatic Index Modification
**	Automatic Storage Modification
***	Direct Storage Addressing with/without Post Indexing

X	INDEX VALUE	X	INDEX VALUE
000	Zero	100	(R4)
001	(R1)	101	(R5)
010	(R2)	110	(R6)
011	(R3)	111	(R7)

12.0 AP-101S INSTRUCTION REPERTOIRE

12.1 SHUTTLE INSTRUCTION SET

Name	Mnemonics	Format
Fixed Point Operations		
Add	AR,A	RR,SRS,RS
Add Halfword	AH	SRS,RS
Add Halfword Immediate	AHI	RI
Add to Storage	AST	RS
Compare	CR,C	RR,SRS,RS
Compare Between Limits	CBL	RR
Compare Halfword	CH	SRS,RS
Compare Halfword Immediate	CHI	RI
Compare Immediate with Storage	CIST	SI
Divide	DR,D	RR,SRS,RS
Exchange Upper and Lower Halfwords	XUL	RR
Insert Address Low	IAL	SRS,RS
Insert Halfword Low	IHL	RS
Load	LR,L	RR,SRS,RS
Load Address	LA	SRS,RS
Load Arithmetic Complement	LCR	RR
Load Fixed Immediate	LFXI	RR
Load Halfword	LH	SRS,RS
Load Multiple	LM	RS
Modify Storage Halfword	MSTH	SI
Multiply	MR,M	RR,SRS,RS
Multiply Halfword	MH	SRS,RS
Multiply Halfword Immediate	MHI	RI
Multiply Integer Halfword	MIH	RS
Store	ST	SRS,RS
Store Halfword	STH	SRS,RS
Store Multiple	STM	RS
Subtract	SR,S	RR,SRS,RS
Subtract from Storage	SST	RS
Subtract Halfword	SH	SRS,RS
Tally Down	TD	SRS,RS

Name	Mnemonics	Format
Floating Point Operations		
Add (Long Operand)	AEDR,AED	RR,RS
Add (Short Operands)	AER,AE	RR,SRS,RS
Compare (Short Operand)	CER,CE	RR,RS
Compare (Long Operand)	CEDR,CED	RR,RS
Convert to Fixed Point	CVFX	RR
Convert to Floating Point	CVFL	RR
Divide (Extended Operand)	DEDR,DED	RR,RS
Divide (Short Operand)	DER,DE	RR,SRS,RS
Load (Long Operand)	LED	RS
Load (Short Operand)	LER,LE	RR,SRS,RS
Load Complement (Short Operand)	LECR	RR
Load Fixed Register	LFXR	RR
Load Floating Immediate	LFLI	RR
Load Floating Register	LFLR	RR
Midvalue Select (Short Operands)	MVS	RS
Multiply (Extended Operand)	MEDR,MED	RR,RS
Multiply (Short Operand)	MER,ME	RR,SRS,RS
Subtract (Long Operand)	SEDR,SED	RR,RS
Subtract (Short Operand)	SER,SE	RR,SRS,RS
Store (Long Operand)	STED	RS
Store (Short Operand)	STE	SRS,RS
Special Operations		
Diagnose*	-	RS
Store Extended Address	STXA	RR,RS
Store DSE Multiple	STDM	RS
Insert Storage Protect Bits*	ISPB	RS
Load Program Status*	LPS	RS
Move Halfword Operands	MVH	RR
Set Program Mask	SPM	RR
Set System Mask*	SSM	RS
Stack Call	SCAL	RS
Stack Return	SRET	RR
Load DSE Multiple	LDM	RS
Load Extended Address	LXA	RR,RS
Supervisor Call	SVC	RS
Test and Set	TS	RS
Test and Set Bits	TSB	SI
Internal Control Operations		
Internal Control*	ICR	RR
I/O Operations		
Program Controlled Input/Output*	PC	RR

*Privileged Instruction

Name	Mnemonics	Format
Branch Operations		
Branch and Link	BALR,BAL	RR,RS
Branch and Index	BIX	RS
Branch on Condition	BCR,BC	RR,RS
Branch on Condition Backward	BCB	SRS
Branch on Condition (Extended)	BCRE	RR
Branch on Condition Forward	BCF	SRS
Branch on Count	BCTR,BCT	RR,RS
Branch on Count Backward	BCTB	SRS
Branch on Overflow and Carry	BVCR,BVC	RR,RS
Branch on Overflow and Carry Forward	BVCF	SRS
Shift Operations		
Normalize and Count	NCT	RR
Shift Left Logical	SLL	SRS
Shift Left Double Logical	SLDL	SRS
Shift Right Arithmetic	SRA	SRS
Shift Right Double Arithmetic	SRDA	SRS
Shift Right Logical	SRL	SRS
Shift Right Double Logical	SRDL	SRS
Shift Right and Rotate	SRR	SRS
Shift Right Double and Rotate	SRDR	SRS
Logical Operations		
AND	NR,N	RR,SRS,RS
AND Halfword Immediate	NHI	RI
AND Immediate with Storage	NIST	SI
AND to Storage	NST	RS
Exclusive-OR	XR,X	RR,SRS,RS
Exclusive-OR Halfword Immediate	XHI	RI
Exclusive-OR Immediate with Storage	XIST	SI
Exclusive-OR to Storage	XST	RS
OR	OR,O	RR,SRS,RS
OR Halfword Immediate	OHI	RI
OR to Storage	OST	RS
Search Under Mask	SUM	RR
Set Bits	SB	SI
Set Halfword	SHW	SRS,RS
Test Bits	TB	SI
Test Register Bits	TRB	RI
Test Halfword	TH	SRS,RS
Zero Bits	ZB	SI
Zero Register Bits	ZRB	RI
Zero Halfword	ZH	SRS,RS

16.0 PIPELINE TIMING CONSIDERATIONS

The AP-101S computer is a pipelined machine which exhibits significant throughput improvement over nonpipelined sequential machines. The pipeline which is involved is based on prefetching both instructions and operands from memory. Instructions and operands are prefetched assuming sequential instruction execution. This means that as long as the sequence of instruction execution is not altered, all prefetched information will be used.

Some branch instructions alter the sequence of execution, and therefore nullify any prefetched information. The time required to restart the pipeline in this case may be directly attributed to the branch instruction. Instruction execution times for branch instructions include all overhead required to restart the pipeline, if the order of execution is altered.

Other factors also exist which have an impact on the throughput of the pipeline. These factors may not be attributed directly to any one instruction in general, rather they are a function of the order and relationship of instruction execution. Three factors may be classified as follows:

Register conflict	Modification of base or index register needed to prefetch an operand
Store conflict	Modification of prefetched operand
I unit hazard	Modification of prefetched instruction

Instruction execution times do not include any overhead due to these factors. Any penalty in execution time must be considered independant of instruction execution time. The total time required to execute a given sequence of instructions must include any applicable penalty due to these factors.

It is for this reason that a separate description of conflicts and hazards is presented. Not only will this description explain the various conflicts and hazards as previously mentioned, it will also discuss how the conflicts and hazards are resolved and what the execution time impact is associated with these events. Furthermore, numerous conditions, such as branching and store instructions, will be discussed with an emphasis on pipeline operation. Instructions of this type change the nature of pipeline processing near that instruction, but are not a conflict or hazard. In order to aid understanding of the AP-101S computer and the pipeline, these instructions have been included in this discussion. Any execution time impacts due to the pipeline have already been included in the stated instruction execution times.

16.1 INSTRUCTION EXECUTION - PIPELINE BASICS

Every instruction requires at least four stages in order to execute. First, the instruction must be read from memory during the instruction fetch stage. Second, the instruction must be decoded both in terms of what type of operation is specified (add, multiply, shift, etc.) and the effective address of the second operand must be

mputed. Next, the second operand is read from memory using the effective address (EA) during the operand fetch stage. Finally, the instruction may be executed, generally resulting in modification of the general purpose registers. In the case of the AP-101S computer, two additional stages are required in support of the memory references. Since the AP-101S utilizes expanded addressing, an additional stage of address translation is required for every memory operation. Therefore, an instruction address translation stage and operand address translation stage are required. Figure 16-1 shows the relationship between all six stages of the AP-101S computer.

Each stage represents a specific function which is relatively independent of the other functions, except for the given time relationship. It is this independence and the timing sequence which permits the construction of a six stage pipeline. Within the pipeline, each function, or stage, is contained and controlled completely by an independent hardware element. The timing relationship between an instruction and each hardware element is shown in Figure 16-2.

The advantage of using a pipelined organization is obvious when considering the execution of three simple instructions. Figure 16-3 indicates that a total of 18 machine cycles would be required for a sequential machine to execute just three instructions, assuming that each stage of the instruction could be completed in a single machine cycle. Each hardware element is capable of independent operation, which permits pipeline operation as shown in the figure. Notice that a total of 8 machine cycles are required to execute three instructions. Considering pipeline operation for a sequence of a single type of instruction yields the mean time required to execute that instruction. The example shown is for an RS format instruction. If the example were extended indefinitely, the execution time would average to 2 cycles per instruction. Completing a similar pipeline chart for SRS instructions would indicate 1.5 cycles per instruction, and 1 cycle for RR format instructions. For the AP-101S computer, the pipeline cycle time is 0.250 microseconds.

16.2 LONG INSTRUCTIONS - NON-SINGLE-CYCLE EXECUTION

Not all instructions may be executed by the execution unit within a single pipeline cycle. These instructions, referred to as long instructions, force the pipeline to stop while execution proceeds, as indicated in Figure 16-4. This is actually accomplished by postponing further EA calculations until the last machine cycle of the long instruction. Instruction execution times as indicated include any effects of long instructions, as necessary. Notice that even though the pipeline waits for a number of cycles, there are no unused cycles in the execution unit.

16.3 BRANCH INSTRUCTIONS - RESTART THE PIPELINE

Branch instructions, as previously discussed, cause any prefetched information to be discarded and the pipeline must be restarted. The branch instruction shown in Figure 16-5 indicates that 3 machine cycles within the execution unit are unused during the pipeline restart. Also, notice that the target instruction has

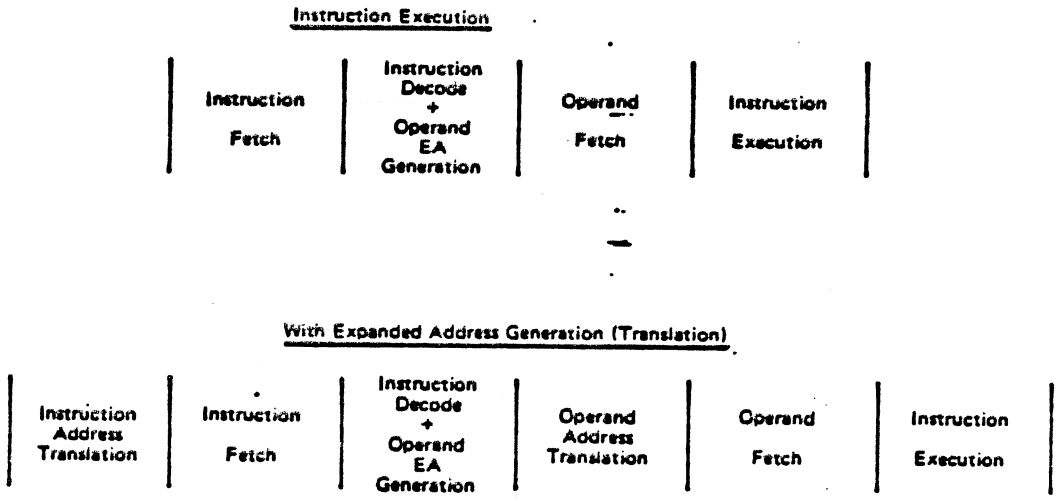


Figure 16-1. Dissection of Instruction

- Sequence of 6 functions → 6 stage pipeline
- Independent hardware per stage/function

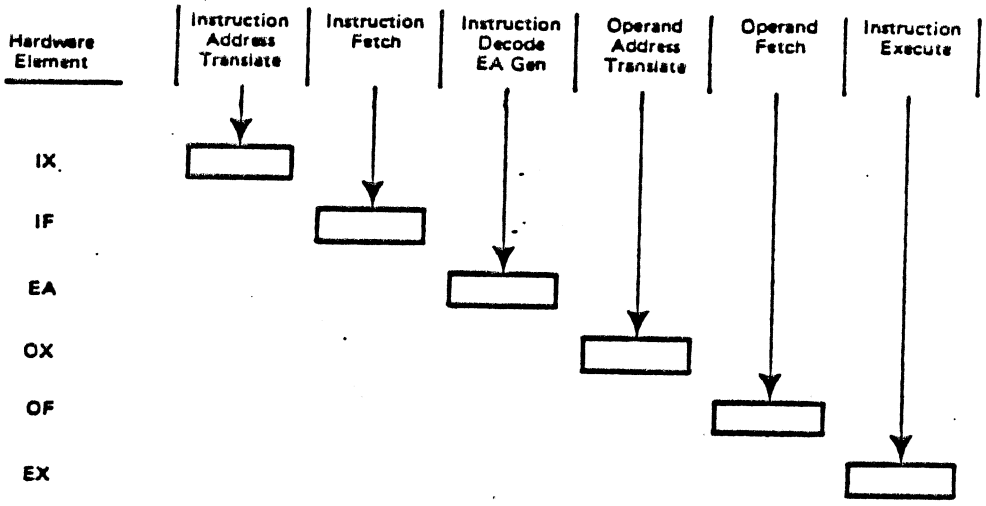


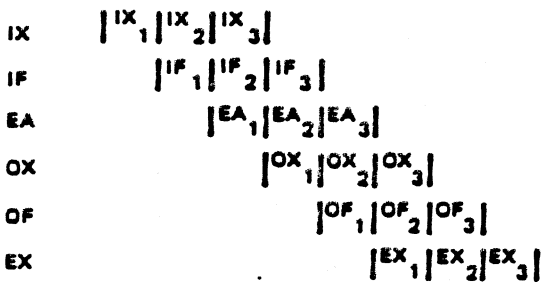
Figure 16-2. Pipeline Hardware Elements

- Consider the instruction sequence 1,2,3
- Sequential machine operation is:

$$|IX_1|IF_1|EA_1|OX_1|OF_1|EX_1|IX_2|IF_2|EA_2|--|--|--|EA_3|OX_3|OF_3|EX_3|$$

6 cycles x 3 instructions = 18 cycles to complete 3 instructions

- Pipeline machine execution is:



8 cycles to complete 3 instructions

Therefore, over a period of time, pipelined instructions would average:

2 cycles / RS instruction
1.5 cycles / SRS instruction
1 cycle / RR instruction

Figure 16-3. Pipeline Advantage

- Not a hazard or conflict
- Instructions which require more than 1 pipeline cycle to execute
- Postpones EA calculations until end of instruction

LOC	INSTR	
L	AE	
L+2	—	(SHORT FLT PU ADD)
L+4	—	
L+6	—	

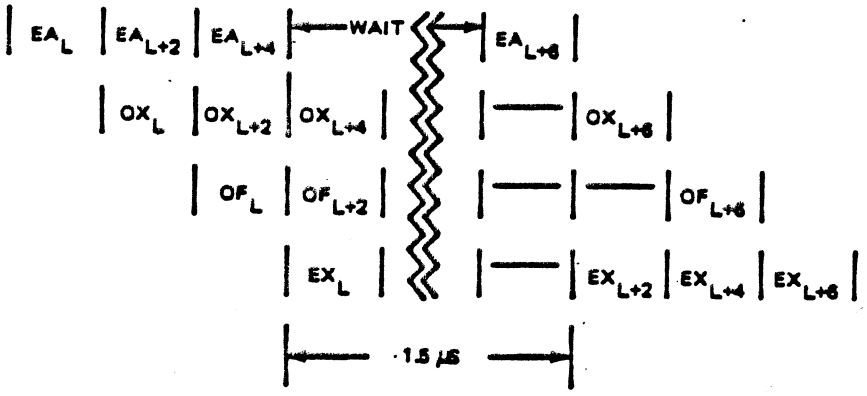


Figure 16-4. Long Instruction

- Not a hazard or conflict
- Harmful to pipeline throughput — 3 cycles to restart
- Example:

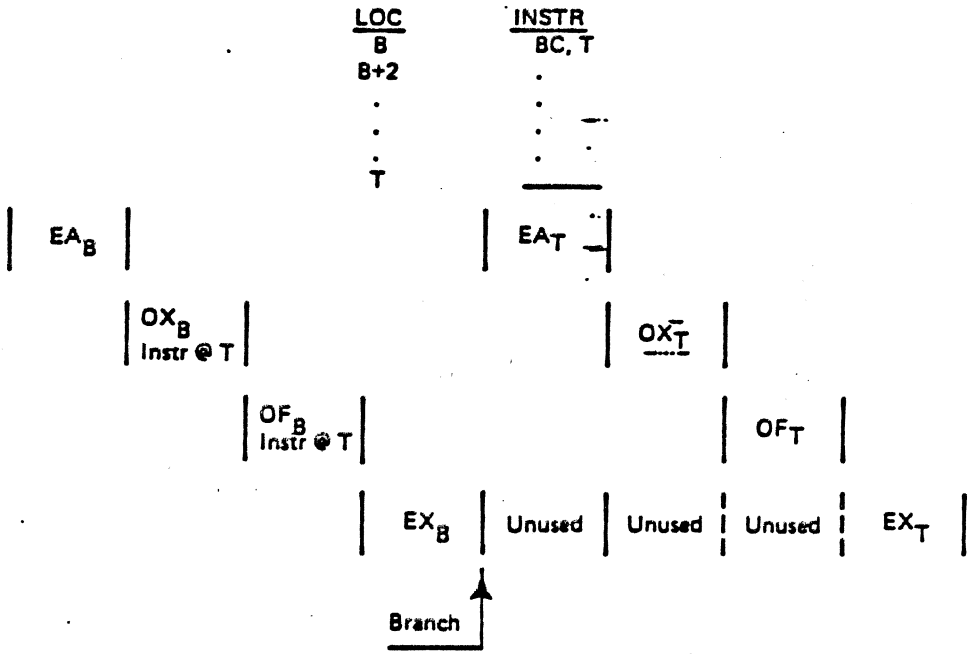


Figure 16-5. Branch Taken

previously been prefetched by the EA unit in order to minimize the restart time. If a conditional branch is not taken, then the pipeline is not restarted. Indicated instruction execution times include all effects of restarting the pipeline.

16.4 REGISTER CONFLICT - MODIFY BASE OR INDEX REGISTER

Register conflicts can only occur for instructions which use either a base or an index register to compute the effective address of a memory operand. A conflict arises if a preceding instruction (within three instructions) modifies the contents of the register which is used for the base or index value. In order to minimize the penalty involved, register conflicts are detected and totally controlled by hardware resources. EA unit operation is postponed, as shown in Figure 16-6, until the register involved has been loaded with the correct value. At most, three machine cycles will be unused by the EA unit while waiting for valid register data. This results in three unused machine cycles in the execution unit hardware. This penalty will decrease, depending upon the number of instructions between the register-modifying instruction and the register-using instruction. Any penalty involved with register conflicts has not been included with the stated instruction execution times, and must be evaluated separately if necessary.

- Caused by loading & using a base/index register within 3 instructions
- Detected and handled by hardware
- Forces sequential instruction execution within pipeline
- Postpones fetch of base/index register by 1, 2, or 3 cycles
- Example:

```
LOC     INSTR
 R      AHR R3, R5
R+1      M  R1, ADDR(R3)
R+3     ---
```

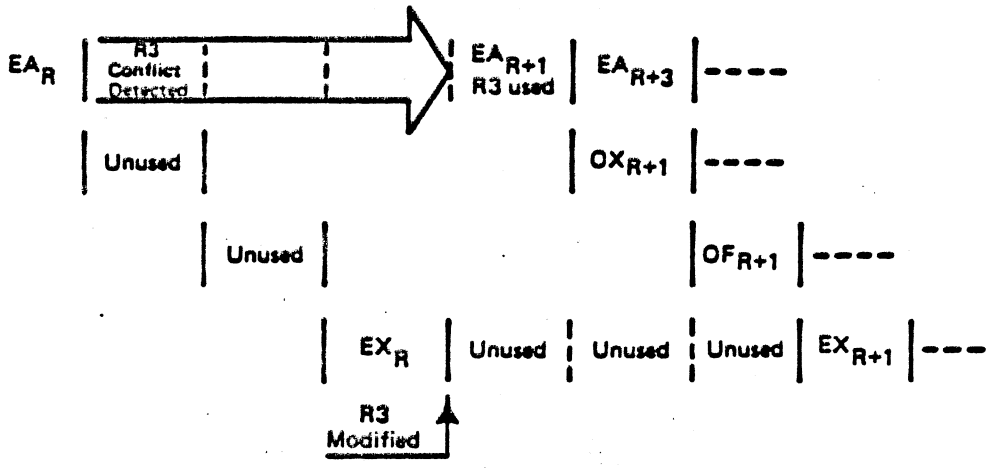


Figure 16-6. Register Conflict

16.5 STORE INSTRUCTIONS - MULTIPLE MEMORY CYCLES

The pipeline structure has been implemented to maximize performance for memory read operations. Memory write operations do not fit into the same pipeline structure as read operations and, as a result, the pipeline is disturbed in the area of a store instruction. Figure 16-7 indicates that two additional memory cycles are needed to perform the actual memory write operation. Also notice that the EA unit performs a pre-read of the memory location in order to assist the memory management unit in storage protection error detection. At most, two cycles will be unavailable for instruction execution due to this pipeline disturbance. The actual number of cycles lost is dependent upon the nature of the instruction following the store instruction. Therefore, the instruction execution time presented for store instructions is a typical value. The corresponding note for applicable store instructions indicates some criteria for determining the exact time required to execute a specific store instruction. Only simple store instructions operate in this fashion. These are; ST, STH, STE and STB.

16.6 STORE CONFLICT - MODIFY PREFETCHED MEMORY OPERAND

Store conflicts are a result of prefetching operands from memory. An operand

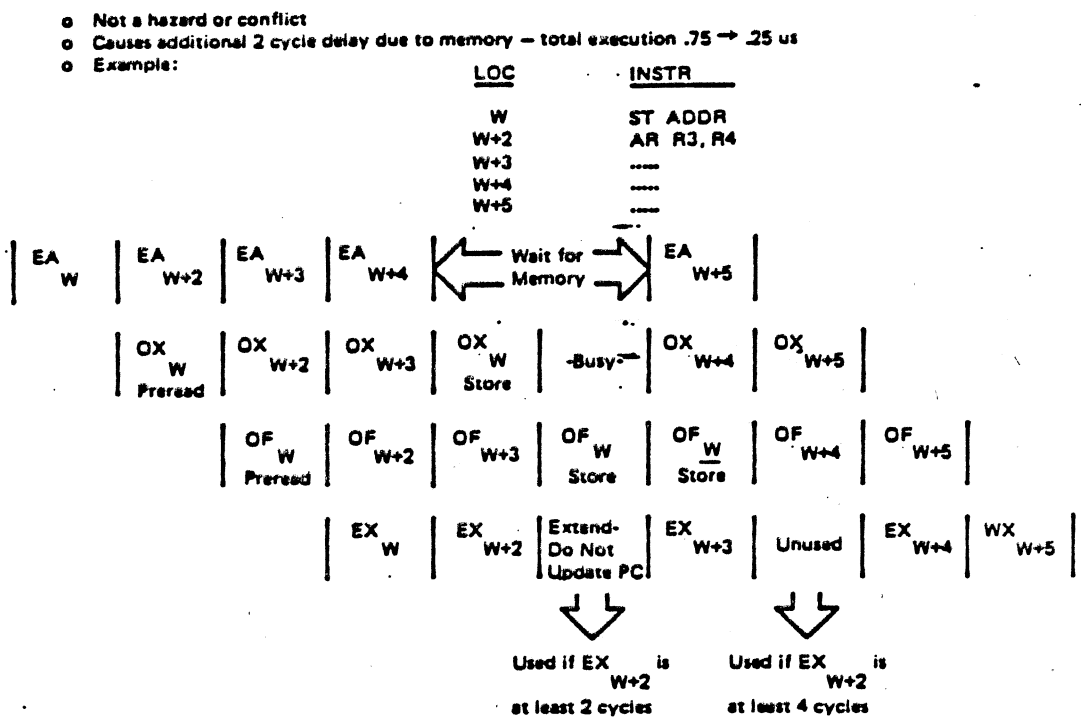


Figure 16-7. Store Instruction

prefetch for a load instruction will actually occur before the memory write is done for a store instruction which precedes the load. If the load and store instructions involve the same memory address, then the operand prefetch for the load instruction must be postponed until the memory write is completed, as shown in Figure 16-8. (The operand fetch actually occurs, however, the data is discarded). In order to minimize the penalty involved, store conflicts are detected and totally controlled by hardware resources. Any penalty involved with store conflicts has not been included with the stated instruction times, and must be evaluated separately if necessary. Store conflicts are applicable for simple store instructions only.

The store conflict hardware has been simplified somewhat by assuming that all memory operations involve two locations, or 32 bits. Therefore, the conflicting instructions only need to deal with memory locations which are within one location of each other in order to cause the detection of a store conflict. Furthermore, store conflicts are detected on the 16 bit logical address, and not the 19 bit physical address. In order to guarantee proper operation with expanded memory addressing, store conflicts are detected on the 15 least significant bits of the logical address. Addresses 7FFF and 0000 are considered to be contiguous, as are addresses FFFF and 8000. At most, two machine cycles will be lost while the operand fetch is postponed. This penalty will decrease to one machine cycle if one other instruction is executed between the conflicting instructions. No conflict will exist if there are two or more intervening instructions.

- Caused by store with successive load from memory within 2 instructions
- Detected and handled by hardware
- Example:

LOC	INSTR
W	ST ADDR
W+2	L ADDR
W+4	

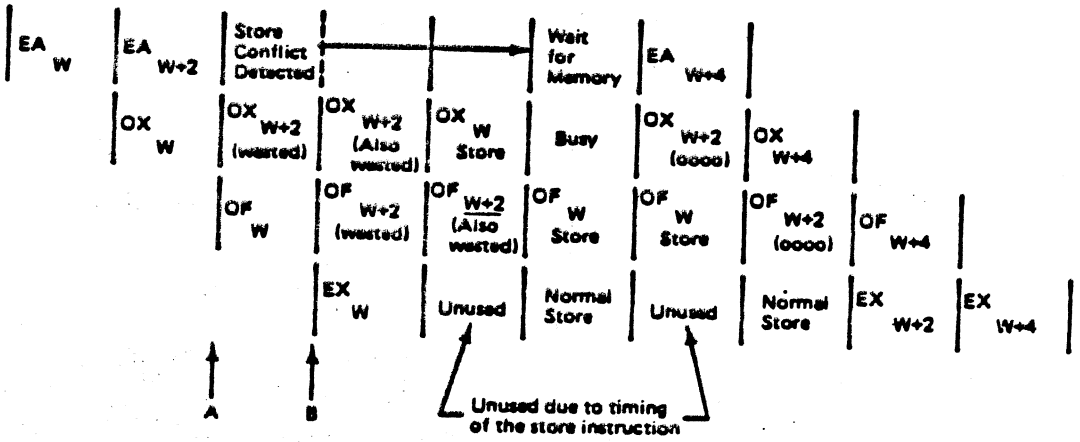


Figure 16-8. Store Conflict

16.7 SUCCESSIVE STORES - BACK-TO-BACK STORES

The execution unit of the CPU contains a store pending register which holds the memory address for simple store instructions. Since only one register exists, only one store instruction can reside in the pipeline at one time. Figure 16-9 indicates that processing by the EA unit for the second store instruction is postponed until the memory write for the first store instruction has been initiated. This situation is not a conflict or hazard, it is only a limitation of the hardware. The guidelines associated with store instruction execution times includes a case for a successive store condition. A penalty of 2 machine cycles has been included with the execution time of the first store instruction. This penalty will decrease to one machine cycle if one other instruction is executed between the store instructions. No penalty exists if there are two or more intervening instructions. The penalty for successive stores is applicable only for simple store instructions.

16.8 I UNIT HAZARD - MODIFICATION OF PREFETCHED INSTRUCTION

An I unit (instruction fetch unit) hazard is the result of a store instruction which modifies memory in the immediate area of the current instruction. The I unit can be at most 22 memory locations ahead of the current instruction. If a store

- o Caused by consecutive store instructions within 2 instructions
- o Detected & handled by hardware
- o Due to existence of one address register for CPU stores

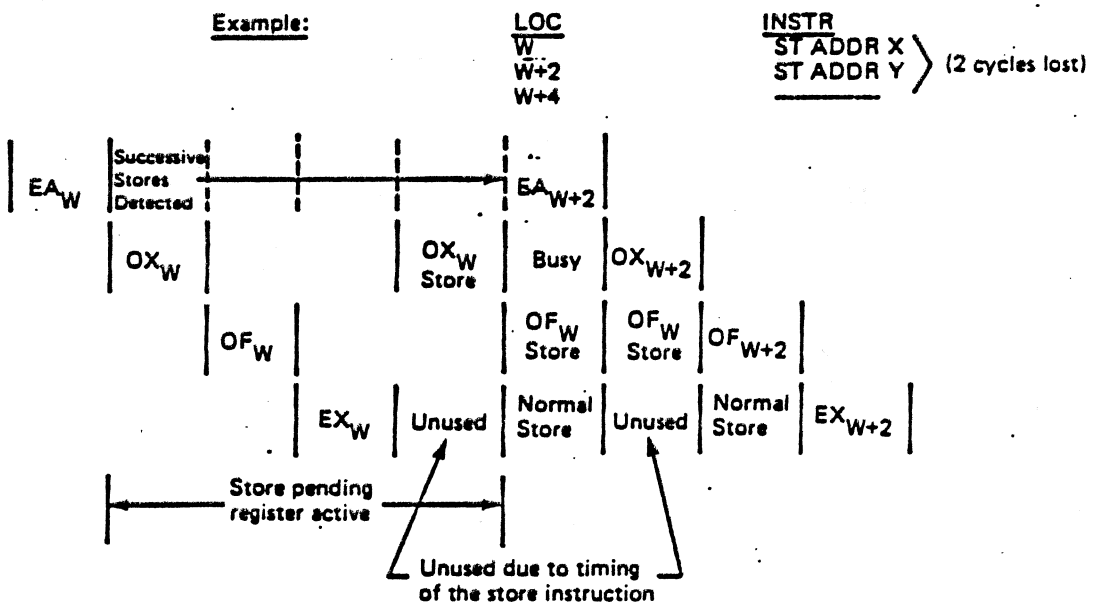


Figure 16-9. Successive Stores

instruction writes to memory in the area from which the I unit may have already prefetched instructions, then an I unit hazard exists. The actual detection circuitry uses the range of IC-1 to IC+23 in order to indicate an I unit hazard. Once a hazard has been detected, the entire pipeline is discarded and restarted from the location following the current instruction, as indicated in Figure 16-10.

I unit hazards are detected on the 16 bit logical address, and not the 19 bit physical address. In order to guarantee proper operation with expanded memory addressing, I unit hazards are detected on the 15 least significant bits of the logical address. Addresses 7FFF and 0000 are considered to be contiguous, as are addresses FFFF and 8000.

The I unit hazard circuitry is provided in order to guard against self-modifying code. This circuitry forces a restart of the pipeline to guarantee that the proper instructions, including modified instructions, are executed. However, it is possible to modify a data location at the end of a program segment and cause an I unit hazard. The I unit hazard circuitry cannot distinguish between memory used for instructions as opposed to data. Therefore, any store within the indicated range will cause an I unit hazard condition whether it is real or not.

16.9 CONFLICT/HAZARD SUMMARY

All effects of the pipeline on instruction execution times have been included with the indicated times except for register conflicts, store conflicts, and I unit

- Caused by a store into memory within the immediate area of the current instruction (-1 ⇔ PC ⇔ +23)
- Forces a restart of the I-unit and pipeline
- Detected by hardware. Handled by microcode

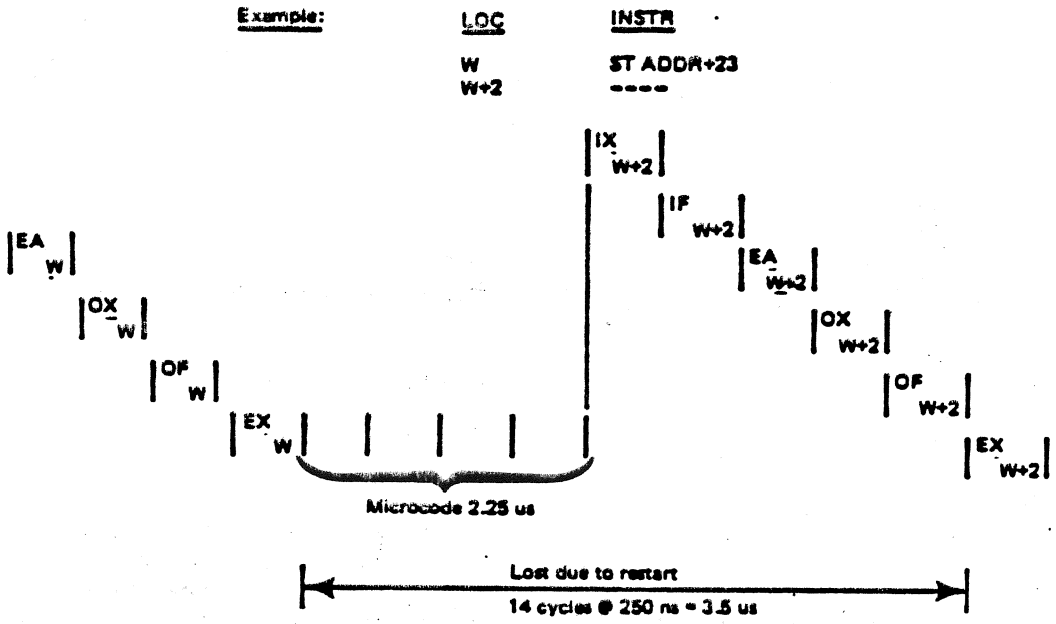


Figure 16-10. I Unit Hazard

hazards. Below is a summary of the penalties involved with each.

	Number of Intervening Instructions		
	0 instr	1 instr	2 instr
Register Conflict	.75 us	.50 us	.25 us
Store Conflict	.50 us	.25 us	----

Independent of Intervening Instructions	
I Unit Hazard	3.50 us

17.0 AP-101S INSTRUCTION EXECUTION TIMES

All floating point execution times have been rounded up to the nearest multiple of 250 nanoseconds and are based on the following assumptions:

- Neither operand is zero, and for the long (64-bit) instructions neither hi or low words of an operand is zero.
- All results will require normalization of 8 bits (2 hex digits).
- All operands are normalized, hence prenormalization of the divisor in the divide instructions is unnecessary.
- For instructions requiring prealignment (Add, Subtract, Compare) the difference in exponents will be 4.
- Operands will not be the same signs (except for the COMPARE instructions in which operands will have identical signs).

MMP INSTRUCTION			INSTRUCTION EXECUTION TIME IN US						
			NORMAL	DOUBLE INDIRECTION				AUTO	AUTO
			ADDRESSING MODES	XC=0 C =0	XC=0 C =1	XC=1 C =0	XC=1 C =1	STORAGE MODIFICATION	INDEXING
A	RS		.250	4.5	4.25	4.25	4.25	5.5	7.25
A	SRS		.250	—	—	—	—	—	—
AE	RS		2.50	6.75	6.5	6.5	6.5	7.5	9.0
AE	SRS		2.50	—	—	—	—	—	—
AED	RS		6.50	10.5	10.25	10.25	10.25	11.5	13.25
AEDR	RR		6.25	—	—	—	—	—	—
AER	RR		2.25	—	—	—	—	—	—
AH	RS		.250	4.50	4.25	4.25	4.25	5.50	7.0
AH	SRS		.250	—	—	—	—	—	—
AHI	RI		.250	—	—	—	—	—	—
AR	RR		.250	—	—	—	—	—	—
AST	RS		.750	6.0	7.0	5.75	7.0	8.25	10.25
BAL	RS		3.75	7.0	10.0	6.75	10.0	8.0	9.5
BALR	RR		BT=3.50; BNT=4.50	—	—	—	—	—	—
BC	RS		BT=1.25; BNT=.250	4.25	7.25	4.0	7.25	5.25	6.25
BCB	SRS		.250	—	—	—	—	—	—
BCF	SRS		.250	—	—	—	—	—	—
BCR	RR		.250	—	—	—	—	—	—
BCRE	RR		BT=5.75; BNT=.50	—	—	—	—	—	—
BCT	RS		BT=1.75; BNT=.750	4.5	7.5	4.25	7.5	5.5	7.0
BCTB	SRS		BT=1.75; BNT=.750	—	—	—	—	—	—
BCTR	RR		BT=1.75; BNT=.750	—	—	—	—	—	—
BIX	RS		BT=2.5; BNT=1.5	5.75	8.7	5.5	8.75	6.75	8.25
BVC	RS		BT=1.25; BNT=.50	4.0	7.0	3.75	7.0	5.0	6.5
BVCF	SRS		BT=1.25; BNT=.50	—	—	—	—	—	—
BVCR	RR		BT=1.25; BNT=.50	—	—	—	—	—	—
C	RS		.250	4.5	4.25	4.25	4.25	5.5	7.25
	SRS		.250	—	—	—	—	—	—
	RR		AVG. = 5.0	—	—	—	—	—	—
	RS		1.75	6.0	5.75	5.75	5.75	6.75	8.5
	RS		5.75	9.75	9.5	9.5	9.5	10.75	12.5
CEDR	RR		5.50	—	—	—	—	—	—
CER	RR		1.50	—	—	—	—	—	—
CH	RS		.250	4.50	4.25	4.25	4.25	5.50	7.0
CH	SRS		.250	—	—	—	—	—	—
CHI	RI		.250	—	—	—	—	—	—
CIST	SI		1.5	—	—	—	—	—	—
CR	RR		.250	—	—	—	—	—	—
CVFL	RR		1.75	—	—	—	—	—	—
CVFX	RR		2.25	—	—	—	—	—	—
D	RS	(R1 EVEN)	AVG. = 4.925	9.05	8.8	8.8	8.8	10.05	11.8
D	RS	(R1 ODD)	AVG. = 4.675	8.8	7.55	7.55	7.55	9.8	10.05
D	SRS	(R1 EVEN)	AVG. = 4.925	—	—	—	—	—	—
D	SRS	(R1 ODD)	AVG. = 4.675	—	—	—	—	—	—
DE	RS		7.50	12	11.5	11.5	11.5	12.75	15.25
DE	SRS		7.50	—	—	—	—	—	—
DED	RS		23.00	27.75	27.75	27.75	27.75	28.75	29.75
DEDR	RR		22.75	—	—	—	—	—	—
DER	RR		7.25	—	—	—	—	—	—
DIAG	RS		SEE POO	—	—	—	—	—	—
DR	RR	(R1 EVEN)	AVG. = 4.925	—	—	—	—	—	—
DR	RR	(R1 ODD)	AVG. = 4.675	—	—	—	—	—	—
IAL	RS		.50	4.0	5.0	3.75	5.0	6.25	8.0
IAL	SRS		.50	—	—	—	—	—	—
ICR	RR		COMMAND DEPENDENT	—	—	—	—	—	—
IHL	RS		.50	4.75	4.50	4.50	4.50	5.75	7.25
ISPB	RS	(R1 = 0)	5.625	8.0	9.0	7.75	9.0	10.25	12.0
ISPB	RS	(R1 = 1)	5.625	8.0	9.0	7.75	9.0	10.25	12.0
ISPB	RS	(R1 = 2)	5.625	8.0	9.0	7.75	9.0	10.25	12.0
ISPB	RS	(R1 = 3)	5.625	8.0	9.0	7.75	9.0	10.25	12.0

MMP INSTRUCTION			INSTRUCTION EXECUTION TIME IN US						
			NORMAL	DOUBLE INDIRECTION				AUTO	AUTO
			ADDRESSING MODES	XC=0 C =0	XC=0 C =1	XC=1 C =0	XC=1 C =1	STORAGE MODIFICATION	INDEXING
ISPB	RS	(R1 = 5)	.125	—	—	—	—	—	—
ISPB	RS	(R1 = 6)	.125	—	—	—	—	—	—
ISPB	RS	(R1 = 7)	.125	—	—	—	—	—	—
L	RS		.250	4.5	4.25	4.25	4.25	5.5	7.25
L	SRS		.250	—	—	—	—	—	—
LA	RS		.250	4.0	5.0	3.75	5.0	6.25	8.0
LA	SRS		.250	—	—	—	—	—	—
LCR	RR		.50	—	—	—	—	—	—
LDM	RS		6.75	10.0	10.0	10.0	10.0	10.25	10.25
LE	RS		1.20	5.0	4.75	4.75	4.75	5.75	8.5
LE	SRS		1.20	—	—	—	—	—	—
LECR	RR		1.00	—	—	—	—	—	—
LED	RS		1.50	5.5	5.0	5.0	5.0	6.25	8.75
LER	RR		1.00	—	—	—	—	—	—
LFLI	RR		.750	—	—	—	—	—	—
LFLR	RR		.750	—	—	—	—	—	—
LFXI	RR		.750	—	—	—	—	—	—
LFXR	RR		.750	—	—	—	—	—	—
LH	RS		.250	4.50	4.25	4.25	4.25	5.50	7.0
LH	SRS		.250	—	—	—	—	—	—
LM	RS		8.5	12.25	13.25	12.0	13.25	14.5	16.25
LPS	RS		10.25	13.25	14.25	13.0	14.25	15.5	17.25
LR	RR		.250	—	—	—	—	—	—
LXA	RR		3.50 (-1.25 for early out)	—	—	—	—	—	—
LXA	RS		3.50 (-1.25 for early out)	6.50	6.25	6.25	6.25	6.50	5.25
M	RS	(R1 EVEN)	2.40	6.53	7.53	6.28	7.53	8.78	10.53
M	RS	(R1 ODD)	2.15	6.28	7.28	6.03	7.28	8.53	10.28
M	SRS	(R1 EVEN)	2.40	—	—	—	—	—	—
M	SRS	(R1 ODD)	2.15	—	—	—	—	—	—
ME	RS	(R1 EVEN)	6.25	10.5	10.25	10.25	10.25	11.5	13.25
ME	RS	(R1 ODD)	5.75	10.0	9.75	9.75	9.75	11.0	12.75
ME	SRS	(R1 EVEN)	5.75	—	—	—	—	—	—
ME	SRS	(R1 ODD)	5.75	—	—	—	—	—	—
MED	RS		19.00	22.5	22.25	22.25	22.25	24.25	25.75
MEDR	RR		18.50	—	—	—	—	—	—
MER	RR	(R1 EVEN)	6.00	—	—	—	—	—	—
MER	RR	(R1 ODD)	5.50	—	—	—	—	—	—
MH	RS		1.35	5.48	5.23	5.23	5.23	6.48	7.98
MH	SRS		1.35	—	—	—	—	—	—
MHI	RI		1.35	—	—	—	—	—	—
MIH	RS		AVG. = 1.7	5.83	5.58	5.58	5.58	6.825	8.025
MR	RR	(R1 EVEN)	2.40	—	—	—	—	—	—
MR	RR	(R1 ODD)	2.15	—	—	—	—	—	—
MSTH	SI		3.0	—	—	—	—	—	—
MVH	RR	(SRC-DEST=1)	9.5+1.75*N (-2.25 FOR DSR)	—	—	—	—	—	—
MVH	RR	(COUNT EVEN)	10.25+.875*N (-2.25 FOR DSR)	—	—	—	—	—	—
MVH	RR	(COUNT ODD)	12.0+.875*(N-1)(-2.25; DSR)	—	—	—	—	—	—
MVH	RR	(COUNT NEG)	7.5 (-2.25 FOR DSR)	—	—	—	—	—	—
MVH	RR	(COUNT ZERO)	7.75 (-2.25 FOR DSR)	—	—	—	—	—	—
MVS	RS		4.75	9.25	9.0	9.0	9.0	10.5	11.75
N	RS		.250	4.75	4.5	4.5	4.5	5.75	6.5
N	SRS		.250	—	—	—	—	—	—
NCT	RR		1.05 + (.075 * N)	—	—	—	—	—	—
NHI	RI		.250	—	—	—	—	—	—
NIST	SI		3.0	—	—	—	—	—	—
NR	RR		.250	—	—	—	—	—	—
NST	RS		.750	6.0	7.0	5.75	7.0	8.25	10.25
O	RS		.250	4.75	4.5	4.5	4.5	5.75	6.5
O	SRS		.250	—	—	—	—	—	—
OHI	RI		.250	—	—	—	—	—	—
OR	RR		.250	—	—	—	—	—	—

JP 7-2-87

MMP INSTRUCTION		INSTRUCTION EXECUTION TIME IN US						
		NORMAL ADDRESSING MODES	DOUBLE INDIRECTION XC=0 C=0	DOUBLE INDIRECTION XC=0 C=1	DOUBLE INDIRECTION XC=1 C=0	DOUBLE INDIRECTION XC=1 C=1	AUTO STORAGE MODIFICATION	AUTO INDEXIN
OST	RS	.750	6.0	7.0	5.75	7.0	8.25	10.25
PC	RR	>4.25 BUT <22.5 (NO CUR DMA)	—	—	—	—	—	—
S	RS	.250	4.5	4.25	4.25	4.25	5.5	7.25
S	SRS	.250	—	—	—	—	—	—
SB	SI	3.0	—	—	—	—	—	—
SCAL	RS	18.125	21.5	24.5	21.25	24.5	22.5	24
SE	RS	2.50	4.75	4.5	4.5	4.5	4.5	9.5
SE	SRS	2.50	—	—	—	—	—	—
SED	RS	6.50	10.75	10.5	10.5	10.5	11.5	13.5
SEDR	RR	6.25	—	—	—	—	—	—
SER	RR	2.25	—	—	—	—	—	—
SH	RS	.250	4.50	4.25	4.25	4.25	5.75	7.25
SH	SRS	.250	—	—	—	—	—	—
SHW	RS	1.50	4.50	5.50	4.25	5.50	6.75	8.50
SHW	SRS	1.50	—	—	—	—	—	—
SLDL	SRS	1.0 + (0.25 * N); N>0	—	—	—	—	—	—
SLL	SRS	.675 + (0.1 * N); N>1	—	—	—	—	—	—
SPM	RR	5.25	—	—	—	—	—	—
SR	RR	.250	—	—	—	—	—	—
SRA	SRS	.650 + (0.1 * N); N>0	—	—	—	—	—	—
SRDA	SRS	1.0 + (0.25 * N); N>0	—	—	—	—	—	—
SRDL	SRS	1.0 + (0.1 * N); N>0	—	—	—	—	—	—
SRDR	SRS	2.0 + (0.5 * N); N<32	—	—	—	—	—	—
SRDR	SRS	2.0 + (0.5 * (N-32)); N>=32	—	—	—	—	—	—
SRET	RR	17.50	—	—	—	—	—	—
SRL	SRS	.650 + (0.1 * N); N>0	—	—	—	—	—	—
SRR	SRS	.650 + (0.1 * N); N>0	—	—	—	—	—	—
SSM	RS	7.75	10.63	11.63	10.38	11.63	12.875	14.625
SST	RS	1.0	—	—	—	—	—	—
ST	RS	0.50	4.75	5.75	4.5	5.75	7.0	9.0
ST	SRS	0.50	—	—	—	—	—	—
STDM	RS	2.25	5.25	6.75	5.0	5.25	7.0	7.5
STE	RS	.500	4.75	4.5	4.5	4.5	4.5	7.5
STE	SRS	.500	—	—	—	—	—	—
STED	RS	1.00	5.25	5.0	5.0	5.0	5.0	7.5
STH	RS	.50	4.50	5.50	4.25	5.50	6.75	8.50
STH	SRS	.50	—	—	—	—	—	—
STM	RS	7.25	10.25	11.25	10.0	11.25	12.5	14.25
STXA	RR	2.50	—	—	—	—	—	—
STXA	RS	2.50	6.50	8.0	6.25	8.0	8.25	8.75
SUM	RR	2.5 * (# ELEMENTS TESTED)	—	—	—	—	—	—
SVC	RS	20.25	22.75	23.75	22.5	23.75	25.0	26.75
TB	SI	2.0	—	—	—	—	—	—
TD	RS	3.0	5.75	5.50	5.50	5.50	6.75	8.25
TD	SRS	3.0	—	—	—	—	—	—
TH	RS	1.75	5.25	5.0	5.0	5.0	6.25	7.75
TH	SRS	1.75	—	—	—	—	—	—
TRB	RI	1.0	—	—	—	—	—	—
TS	RS	3.75	6.50	6.25	6.25	6.25	7.50	9.0
TSB	SI	3.0	—	—	—	—	—	—
X	RS	.250	4.75	4.50	4.50	4.50	5.75	7.50
X	SRS	.250	—	—	—	—	—	—
XHI	RI	.250	—	—	—	—	—	—
XIST	SI	3.0	—	—	—	—	—	—
XR	RR	.250	—	—	—	—	—	—
XST	RS	.750	6.0	7.0	5.75	7.0	8.25	10.25
XUL	RR	1.0	—	—	—	—	—	—
ZB	SI	3.25	—	—	—	—	—	—
ZH	RS	1.50	4.50	5.50	4.25	5.50	6.75	8.50
ZH	SRS	1.50	—	—	—	—	—	—
ZRB	RI	.250	—	—	—	—	—	—