HEWLETT PACKARD

21MX Computer Series

Reference Manual

2000 series

21MX COMPUTER SERIES

reference manual

HEWLETT PACKARD

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

MANUAL PART NO. 02108-90002
MICROFICHE PART NO. 02108-90003

Printed: JUN 1976
Printed in U.S.A.

CONTENTS

Section I Page
SYSTEM FEATURES
- Microprogramming Capabilities . . . 1-2
  - System Speed . . . 1-2
  - Memory Space . . .
  - Special Functions and Security . . .
  - Additional Hardware Facilities . . .
- Hewlett-Packard Software . . . 1-3
- Input/Output . . . 1-3
- Specifications . . . 1-4
- System Expansion and Enhancement . . . 1-4

Section II Page
OPERATING FEATURES
- Hardware Registers . . . 2-1
  - A-Register . . . 2-1
  - B-Register . . . 2-1
  - M-Register . . . 2-1
  - T-Register . . . 2-1
  - P-Register . . . 2-1
  - S-Register . . . 2-1
  - Extend Register . . . 2-1
  - Overflow Register . . . 2-1
  - Display Register . . . 2-2
  - X- and Y-Registers . . . 2-2
- Control and Indicators . . . 2-2
  - Operator Panel . . . 2-2
  - Rear Panel . . . 2-2
  - Internal Switches . . . 2-2
- Basic Operating Examples . . . 2-5
  - Cold Start Procedure . . . 2-5
  - Manual Loading . . . 2-6
  - Running Programs . . . 2-6
- Special Register Display Mode . . . 2-6

Section III Page
PROGRAMMING INFORMATION
- Data Formats . . . 3-1
- Memory Addressing . . . 3-1
  - Paging . . . 3-1
  - Direct and Indirect Addressing . . . 3-3
  - Reserved Memory Locations . . . 3-3
  - Nonexistent Memory . . . 3-4
- Base Set Instruction Formats . . . 3-4
  - Memory Reference Instructions . . . 3-4
  - Register Reference Instructions . . . 3-4
  - Input/Output Instructions . . . 3-4
  - Extended Arithmetic Memory Reference Instructions . . . 3-5
  - Extended Arithmetic Register Reference Instructions . . . 3-5
- Base Set Instruction Coding . . . 3-5
  - Memory Reference Instructions . . . 3-5
  - Register Reference Instructions . . . 3-7
    - Shift/Rotate Group . . . 3-7
    - Alter/Skip Group . . . 3-10
  - Input/Output Instructions . . . 3-11
  - Extended Arithmetic Memory Reference Instructions . . . 3-13
  - Extended Arithmetic Register Reference Instructions . . . 3-14
- Extended Instruction Group Coding . . . 3-16
  - Index Register Instructions . . . 3-16
  - Jump Instructions . . . 3-19
  - Byte Manipulation Instructions . . . 3-20
  - Bit Manipulation Instructions . . . 3-21
  - Word Manipulation Instructions . . . 3-22
- Floating Point Instruction Coding . . . 3-23
- Dynamic Mapping System . . . 3-24
  - Memory Addressing . . . 3-24
  - Map Register Loading . . . 3-24
  - Status and Violation Registers . . . 3-24
  - Map Segmentation . . . 3-25
  - Power Fail Characteristics . . . 3-26
  - DMS Instruction Coding . . . 3-26
  - Sample Map Load/Enable Routine . . . 3-33
  - Additional DMS Definitions . . . 3-33
    - Alternate Map . . . 3-33
    - Protected Mode . . . 3-33
    - MEM Violations . . . 3-33
    - DCPC Operation In a DMS Environment . . . 3-34
- Instruction Execution Times . . . 3-34
- Interrupt System . . . 3-34
  - Power Fail Interrupt . . . 3-38
  - Parity Error Interrupt . . . 3-40
  - Memory Protect/DMS Interrupt . . . 3-40
  - Dual-Channel Port Controller Interrupt . . . 3-41
  - Input/Output Interrupt . . . 3-41
  - Central Interrupt Register . . . 3-43
  - Interrupt System Control . . . 3-43

Section IV Page
INPUT/OUTPUT SYSTEM
- Input/Output Addressing . . . 4-1
- Input/Output Priority . . . 4-1
- Interface Elements . . . 4-3
  - Control Bit . . . 4-3
  - Flag Bit . . . 4-3
  - Buffer . . . 4-4
- Input/Output Data Transfer . . . 4-4
  - Input Data Transfer (Interrupt Method) . . . 4-4
  - Output Data Transfer (Interrupt Method) . . . 4-5
  - Noninterrupt Data Transfer . . . 4-5
    - Input . . . 4-5
    - Output . . . 4-6
  - Dual-Channel Port Controller . . . 4-6
    - DCPC Operation . . . 4-6
    - DCPC Initialization . . . 4-7

Appendix Page
- Computer Physical Layout . . . A-2
- Character Codes . . . A-3
- Octal Arithmetic . . . A-4

CONTENTS (continued)

Octal/Decimal Conversions A-5
Mathematical Equivalents A-6
Octal Combining Tables A-8
Instruction Codes in Octal A-9
Base Set Instruction Codes in Binary A-10
Extended Instruction Group Codes in Binary . . . A-11
Dynamic Mapping System Instruction
  Codes in Binary A-12
Extend and Overflow Examples A-13
Interrupt and I/O Control Summary A-14

ILLUSTRATIONS

Title Page

Microprogrammable Processors 1-0
Operator Panel Controls and Indicators 2-2
Special Register Display Mode Pointers 2-7
Data Formats and Octal Notation 3-2
Base Set Instruction Formats 3-4
Shift and Rotate Functions 3-7
Examples of Double-Word Shifts and Rotates . . . 3-15
Basic Memory Addressing Scheme 3-24
Expanded Memory Addressing Scheme 3-24
Basic Word Format Vs Map Register Format . . . 3-24

Title Page

Map Segmentation 3-26
Input/Output System 4-1
I/O Address Assignments 4-2
Priority Linkage 4-2
Interrupt Sequences 4-3
Input Data Transfer (Interrupt Method) 4-4
Output Data Transfer (Interrupt Method) 4-5
DCPC Input Data Transfer 4-7
DCPC Control Word Formats 4-7

TABLES

Title Page

Specifications 1-5
Options and Accessories 1-9
Operator Panel Control and Indicator Functions . . 2-3
Special Register Display Mode
  Switch Operation 2-8
Effects of Storing/Displaying Special
  Registers 2-9
Memory Paging 3-3
Reserved Memory Locations 3-3
Shift/Rotate Group Combining Guide 3-7
Alter/Skip Group Combining Guide 3-10

Title Page

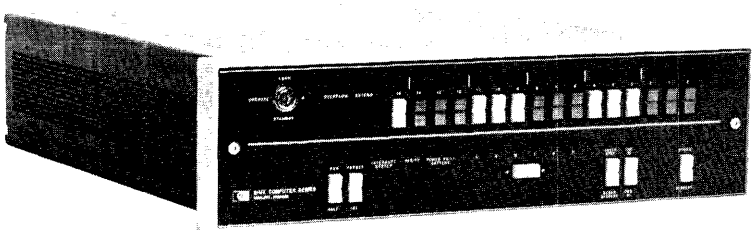
MEM Status Register Format 3-25
MEM Violation Register Format 3-25
Sample DMS Load/Enable Routine 3-35
Instruction Execution Times 3-36
HP 2105A Interrupt Assignments 3-38
HP 2108A Interrupt Assignments 3-38
HP 2112A Interrupt Assignments 3-38
Sample Power Fail Subroutine 3-39
Sample Memory Protect/Parity Error Subroutine . . 3-42
Noninterrupt Transfer Routines 4-6
DCPC Initialization Program 4-8

ALPHABETICAL INDEX OF STANDARD INSTRUCTIONS

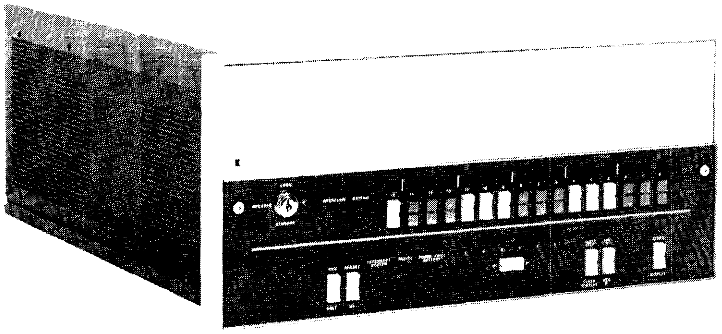
Instruction		Page
ADA	Add to A	3-5
ADB	Add to B	3-5
ADX	Add Memory to X	3-16
ADY	Add Memory to Y	3-16
ALF	Rotate A Left Four	3-8
ALR	A Left Shift, Clear Sign	3-8
ALS	A Left Shift	3-8
AND	"And" to A	3-5
ARS	A Right Shift	3-8
ASL	Arithmetic Shift Left (32)	3-14
ASR	Arithmetic Shift Right (32)	3-14
BLF	Rotate B Left Four	3-8
BLR	B Left Shift, Clear Sign	3-8
BLS	B Left Shift	3-8
BRS	B Right Shift	3-8
CAX	Copy A to X	3-17
CAY	Copy A to Y	3-17
CBS	Clear Bits	3-21
CBT	Compare Bytes	3-20
CBX	Copy B to X	3-17
CBY	Copy B to Y	3-17
CCA	Clear and Complement A	3-10
CCB	Clear and Complement B	3-10
CCE	Clear and Complement E	3-10
CLA	Clear A	3-10
CLB	Clear B	3-10
CLC	Clear Control	3-12
CLE	Clear E	3-9,3-10
CLF	Clear Flag	3-12
CLO	Clear Overflow	3-12
CMA	Complement A	3-10
CMB	Complement B	3-10
CME	Complement E	3-11
CMW	Compare Words	3-22
CPA	Compare to A	3-6
CPB	Compare to B	3-6
CXA	Copy X to A	3-17
CXB	Copy X to B	3-17
CYA	Copy Y to A	3-17
CYB	Copy Y to B	3-17
DIV	Divide	3-14
DLD	Double Load	3-14
DST	Double Store	3-14
DSX	Decrement X and Skip if Zero	3-17
DSY	Decrement Y and Skip if Zero	3-17
ELA	Rotate E Left with A	3-9
ELB	Rotate E Left with B	3-9
ERA	Rotate E Right with A	3-9
ERB	Rotate E Right with B	3-9
FAD	Floating Point Add	3-23
FDV	Floating Point Divide	3-23
FIX	Floating Point to Integer	3-23
FLT	Integer to Floating Point	3-24
FMP	Floating Point Multiply	3-24
FSB	Floating Point Subtract	3-24
HLT	Halt	3-12
INA	Increment A	3-11
INB	Increment B	3-11
IOR	"Inclusive Or" to A	3-6
ISX	Increment X and Skip if Zero	3-17
ISY	Increment Y and Skip if Zero	3-17
ISZ	Increment and Skip if Zero	3-6
JLY	Jump and Load Y	3-19
JMP	Jump	3-6
JPY	Jump Indexed by Y	3-20
JSB	Jump to Subroutine	3-6
LAX	Load A Indexed by X	3-18
LAY	Load A Indexed by Y	3-18
LBT	Load Byte	3-20
LBX	Load B Indexed by X	3-18
LBY	Load B Indexed by Y	3-18
LDA	Load A	3-6
LDB	Load B	3-6
LDX	Load X from Memory	3-18
LDY	Load Y from Memory	3-18
LIA	Load Input to A	3-12
LIB	Load Input to B	3-12
LSL	Logical Shift Left (32)	3-16
LSR	Logical Shift Right (32)	3-16
MBT	Move Bytes	3-21
MIA	Merge Into A	3-12
MIB	Merge Into B	3-12
MPY	Multiply	3-14
MVW	Move Words	3-23
NOP	No Operation	3-9
OTA	Output A	3-13
OTB	Output B	3-13
RAL	Rotate A Left	3-9
RAR	Rotate A Right	3-9
RBL	Rotate B Left	3-9
RBR	Rotate B Right	3-9
RRL	Rotate Left (32)	3-16
RRR	Rotate Right (32)	3-16
RSS	Reverse Skip Sense	3-11
SAX	Store A Indexed by X	3-18
SAY	Store A Indexed by Y	3-19
SBS	Set Bits	3-22
SBT	Store Byte	3-21
SBX	Store B Indexed by X	3-19
SBY	Store B Indexed by Y	3-19
SEZ	Skip if E is Zero	3-11
SFB	Scan For Byte	3-21
SFC	Skip if Flag Clear	3-13
SFS	Skip if Flag Set	3-13
SLA	Skip if LSB of A is Zero	3-10,3-11
SLB	Skip if LSB of B is Zero	3-10,3-11
SOC	Skip if Overflow Clear	3-13
SOS	Skip if Overflow Set	3-13
SSA	Skip if Sign of A is Zero	3-11
SSB	Skip if Sign of B is Zero	3-11
STA	Store A	3-6
STB	Store B	3-7
STC	Set Control	3-13
STF	Set Flag	3-13
STO	Set Overflow	3-13
STX	Store X to Memory	3-19
STY	Store Y to Memory	3-19
SZA	Skip if A is Zero	3-11
SZB	Skip if B is Zero	3-11
TBS	Test Bits	3-22
XAX	Exchange A and X	3-19
XAY	Exchange A and Y	3-19
XBX	Exchange B and X	3-19
XBY	Exchange B and Y	3-19
XOR	"Exclusive Or" to A	3-7

ALPHABETICAL INDEX OF DYNAMIC MAPPING SYSTEM INSTRUCTIONS

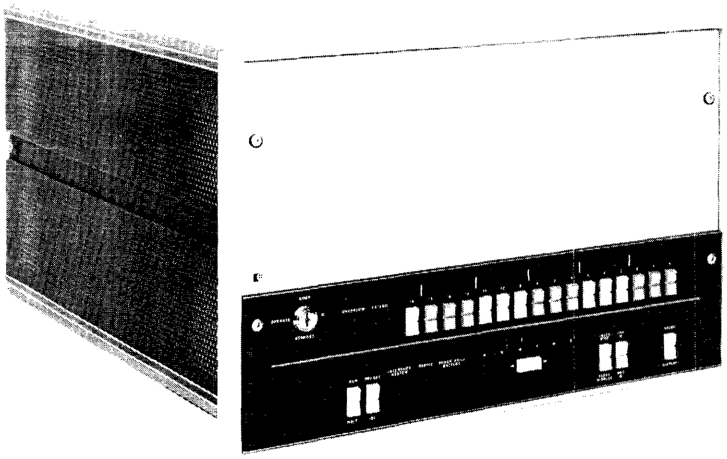
Instruction		Page
DJP	Disable MEM and JMP	3-26
DJS	Disable MEM and JSB	3-26
JRS	Jump and Restore Status	3-26
LFA	Load Fence From A	3-27
LFB	Load Fence From B	3-27
MBF	Move Bytes From Alternate Map	3-27
MBI	Move Bytes Into Alternate Map	3-27
MBW	Move Bytes Within Alternate Map	3-28
MWF	Move Words From Alternate Map	3-28
MWI	Move Words Into Alternate Map	3-28
MWW	Move Words Within Alternate Map	3-28
PAA	Load/Store Port A Map per A	3-28
PAB	Load/Store Port A Map per B	3-29
PBA	Load/Store Port B Map per A	3-29
PBB	Load/Store Port B Map per B	3-29
RSA	Read Status Register Into A	3-29
RSB	Read Status Register Into B	3-29
RVA	Read Violation Register Into A	3-29
RVB	Read Violation Register Into B	3-29
SJP	Enable System Map and JMP	3-29
SJS	Enable System Map and JSB	3-30
SSM	Store Status Register Into Memory	3-30
SYA	Load/Store System Map per A	3-30
SYB	Load/Store System Map per B	3-30
UJP	Enable User Map and JMP	3-30
UJS	Enable User Map and JSB	3-31
USA	Load/Store User Map per A	3-31
USB	Load/Store User Map per B	3-31
XCA	Cross Compare A	3-31
XCB	Cross Compare B	3-31
XLA	Cross Load A	3-32
XLB	Cross Load B	3-32
XMA	Transfer Maps Internally per A	3-32
XMB	Transfer Maps Internally per B	3-32
XMM	Transfer Maps or Memory	3-32
XMS	Transfer Maps Sequentially	3-33
XSA	Cross Store A	3-33
XSB	Cross Store B	3-33



HP 2105A



HP 2108A



HP 2112A

Figure 1-1. Microprogrammable Processors

SYSTEM FEATURES

The microprogrammable processors shown in figure 1-1 accommodate a variety of memory configurations and processor options to form the new and powerful HP 21MX Computer Series. Salient features of these computers, which utilize the latest developments in semiconductor technology, are as follows.

PROCESSOR

- Powerful user-microprogrammable processor with 178 micro-orders and 4K of control store space.
- 128 standard instructions including 80 instructions which emulate the HP 2100 Series Computer; 42 new instructions for indexing, byte and bit manipulation, byte and word moves, and byte string scanning; and 6 single-precision floating point instructions.
- 4 general-purpose registers, two of which may be used as index registers.
- Fully microprogrammed processor, including all arithmetic functions, input/output, and operator panel control.
- Initial binary loader is ROM resident and callable by a pushbutton switch on the operator panel. A paper tape loader ROM is standard; provision is made for up to three additional loader ROM's, which are available as options or may be user-generated.
- Operator panel is standard.
- Writable Control Store (WCS) is optional.
- Programmable ROM (pROM) Writer available as a supporting product.
- Dynamic Mapping System (DMS) is optional on HP 2108A and HP 2112A Processors.

POWER SUPPLY

- Power module supplies power for the processor and all processor options, the memory system and all memory options, all standard and optional microcode packages, and a wide range of I/O controller configurations.
- Power module operates over a wide range of line voltage and line frequency variations; operates through a line loss of 2.5 cycles at 60 hertz.
- Over-current and over-voltage protection on all power supply outputs.
- Input voltage protection.
- Line standby mode for memory integrity.

MEMORY

- N-channel MOS semiconductor memory with 650-nanosecond cycle time; expandable as follows:

  HP 2101A Memory — expandable in mainframe to 32K in HP 2105A; 64K in HP 2108A, and 128K in HP 2112A.

  HP 2102A Memory — expandable in mainframe to 32K in HP 2105A; 64K in HP 2108A, and 128K in HP 2112A.
- Memory parity generation and checking are standard.
- Up to 2 hours of memory sustaining power with optional power fail recovery system; this option provides an automatic restart capability.
- Optional Dual-Channel Port Controller (DCPC) with bidirectional input/output transfer; provides for direct memory access, program assignable, to any I/O channel.
- Memory, I/O, and infinite indirect addressing protection by the memory protect or DMS option (not available for HP 2105A Processor).
- Memory integrity is maintained through a line loss of 160 milliseconds.
- Optional HP 12990A Memory Extender adds space and power for eight semiconductor memory modules.

INPUT/OUTPUT

- Fully microprogrammed I/O instructions.
- HP 2105A has four I/O channels in mainframe.
- HP 2108A has nine I/O channels in mainframe.
- HP 2112A has 14 I/O channels in mainframe.
- I/O channels can be increased with one or two optional HP 12979A I/O Extenders (16 additional channels each).
- Compatibility maintained with existing HP 2100 Series Computer I/O interfaces.

These compact data processors are supplied with a comprehensive set of software, including assemblers, compilers, and operating systems. A full line of Hewlett-Packard peripherals and I/O interface kits are available to provide a flexible and efficient systems package. The programming languages are those commonly encountered in data processing and control.

All three processors are microprogrammed and microprogrammable, which allows program-structured commands to be coded into a more concise language than is possible using machine language. The ROM modules in which microprograms are stored are referred to collectively as *control store*. Standard control store consists of 1,024 directly addressable locations configured into four modules of 256 locations each. Each control store location accommodates one microinstruction, which in turn consists of a 24-bit word encompassing six micro-orders. The control store address space of each processor is 4,096 words.

Microprograms in standard control store for executing the various machine functions are as follows:

- Base instruction set (modules 0 and 1)
- Floating point instructions (module 14)
- Extended instruction group (module 15)

Unused modules of control store are available for user-supplied microprograms. Microprogramming capabilities from a hardware standpoint are provided in *Microprogramming HP 21MX Computers Operating and Reference Manual*, part no. 02108-90008. This manual covers in detail the following:

a. HP Microassembler

b. HP Micro Debug Editor

c. HP Programmable ROM Writer

d. HP Writable Control Store (WCS) I/O Utility Routine

e. Basic Control System (BCS) and Disc Operating System (DOS) versions of a through d above.

Some of the more important benefits of microprogramming are presented in following paragraphs.

1-2. SYSTEM SPEED

Microprogramming can increase the system speed in many ways. Since microinstructions are executed from 5 to 10 times faster than machine language instructions, a frequently used software subroutine will execute much faster when in the form of a microprogram. With 14 additional registers available to a microprogram, the number of main memory accesses can be greatly reduced. This is particularly significant in real-time systems which are compute-bound (i.e., systems in which the I/O is performed faster than the computation).

1-3. MEMORY SPACE

By converting software routines into microprograms, main memory space is freed for other purposes. The routines remain instantly callable, as opposed to routines which are relegated to disc or drum storage.

1-4. SPECIAL FUNCTIONS AND SECURITY

The computer instruction set can be expanded to perform functions that are oriented to specific applications. Thus, the general-purpose computer can become a special-purpose machine uniquely adapted to a particular environment. Because of the relative inaccessibility of microprograms as compared to conventional software in main memory, proprietary packages which are coded as microprograms inherently have a high degree of security.

1-5. ADDITIONAL HARDWARE FACILITIES

Microinstructions in control store are 24 bits long whereas machine language instructions residing in main memory are 16 bits long. In addition, microinstructions have access to many internal registers and logic functions that machine language instructions cannot use. Software can be invented and implemented in control store to reference these registers as required by the microprogram.

1-6. WRITABLE CONTROL STORE. The Writable Control Store (WCS) option provides a read-write control store module which can be used for the development and execution of user-supplied microprograms. Microprograms in WCS are executed at the same speed as those in the read-only control store. Four WCS cards may be used with HP 2112A, two WCS cards may be used in an HP 2108A, and one WCS card may be used in an HP 2105A. The module containing the basic instruction set must always be in the form of Read-Only-Memory (ROM) integrated circuit chips. Each WCS module consists of a single card which plugs into the I/O PCA cage, thus eliminating the need for extensive cabling or an additional power supply. A WCS card contains 256 24-bit locations of Random-Access-Memory (RAM), including all necessary address and read/write circuits. WCS can be written into or read under computer control using standard input/output instructions. An I/O utility routine makes it possible for FORTRAN and ALGOL programs to write into or read from a WCS module using a conventional subroutine call. A WCS module is read at full speed by way of a flat cable connecting it to the control section of the processor. Software supplied with the WCS card includes a micro-assembler, a micro debug editor, a WCS I/O driver, a WCS I/O utility routine, and a WCS diagnostic.

1-7. PROGRAMMABLE ROM WRITER. The Programmable ROM Writer option makes it possible for the user to permanently transfer microprograms to programmable Read-Only-Memory chips which can then be physically added to the control section of the computer. The Programmable ROM Writer consists of a single card which plugs into a computer I/O slot, thus eliminating the need for extensive cabling or an additional power supply. A small box is connected to the Programmable ROM Writer card by way of a cable; the programmable ROM chip to be burned is mounted on the box by the computer operator. A stand-alone computer program, supplied with the Programmable ROM Writer, burns and verifies the chip using punched tape input.

1-8. DYNAMIC MAPPING SYSTEM. The Dyanmic Mapping System (DMS) option gives the user the capability to address physical memory configurations larger than the standard 32,768 word limitation. The DMS provides a 20-bit-wide memory address bus which allows an addressing space of 1,048,576 words of main memory and allows the user to specify each 1,024-word page within physical memory to be read and/or write protected for program security. Separate memory translation maps provide isolation of user from system and user from user. All systems using the DMS execute with the same memory cycle time as those systems having 32K words or less of memory.

The DMS consists of a Memory Expansion Module (MEM) and a Memory Protect PCA which plug into the memory PCA cage; microcode for implementing the additional 38 machine language instructions associated with the DMS is mounted in the control store section of the Central Processor Unit (CPU).

1-9. FAST FORTRAN PROCESSOR. The Fast FORTRAN Processor (FFP) option provides the system with 13 subroutines implemented in three control store ROM modules (modules 3, 4, and 5). Included are five fast FORTRAN subroutines (.GOTO, ..MAP, .ENTR, .ENTP, and .SETP) and eight extended precision subroutines (DBLE, SNGL, .XMPY, .XDIV, .DFER, .XFER, .XADD, and .XSUB). These subroutines are executed up to 28 times faster than the same routines executed under software control.

Software for the HP 21MX Computer Series includes four high-level programming languages: HP FORTRAN, HP FORTRAN IV, HP ALGOL, and HP BASIC, plus an efficient, extended assembler which is callable by FORTRAN and ALGOL. Utility software includes a debugging routine, a symbolic editor, and a library of commonly used computational procedures such as Boolean, trigonometric, and plotting functions, real/integer conversions, natural log, square root, etc.

Hewlett-Packard provides several systems built around BASIC interpreters. The single-terminal BASIC system allows the user to prepare and run BASIC language programs conversationally through a teleprinter. Programs can also be entered through a tape reader and punched out on tape punches. A similar system, Educational BASIC, allows BASIC programs to be translated from marked cards. The time-shared BASIC systems provide an extended version of the BASIC language to 16 or 32 users simultaneously. The extensions to BASIC allow the user to store and access large amounts of data in an external mass memory, to manipulate strings of characters, and to store and retrieve programs in mass memory.

Several operating systems are available, covering a wide range of applications. The Basic Control System, which simplifies the control of input/output operations, also provides relocatable loading and linking of user programs. The time-shared systems, using conversational BASIC language, permit up to 32 terminals to be connected to the system, either directly or by telephone lines via Dataphones. The Hewlett-Packard Real-Time Executive (RTE) system permits several programs to run in real-time concurrently with general-purpose background programs. This allows multiple data-processing capabilities where separate computers are not economically feasible. The user can write programs in HP Assembly, FORTRAN, or ALGOL languages. A Magnetic Tape System and a Disc Operating System are also available. These systems greatly increase the speed and simplicity of assembling, compiling, loading, and executing user programs.

The Hewlett-Packard User Library includes over 800 tested and documented BASIC programs contributed for users. The library is segmented into the following five categories:

- Data Handling and Programming Utilities
- Scientific and Numerical Analyses
- Operations Research and Business Applications
- Education
- Demonstration Routines

Interfacing of peripheral devices is accomplished by plug-in interface printed-circuit assemblies (PCA's). The HP 2105A, HP 2108A, and HP 2112A mainframes can accommodate four, nine, and fourteen interface PCA's, respectively. The I/O capability can be increased with one or two optional HP 12979A Input/Output Extenders (16 additional channels each). Interface PCA's are available for a wide variety of peripheral devices, and virtually all interfaces developed for use with HP 2100 Computers may be used with the HP 21MX Computer Series.

All I/O channels are buffered and bidirectional, and are serviced through a multilevel vectored priority interrupt structure. The two Dual-Channel Port Controller (DCPC) channels are program-assignable to any two of the I/O channels in the mainframe, expandable to all I/O channels if a DCPC is installed in the I/O extender. DCPC transfers occur on an I/O cycle-stealing basis, not subject to the I/O priority structure. The total bandwidth through both DCPC channels is 616,666 words per second. It is possible for the CPU to interleave memory cycles while the DCPC is operating at full bandwidth.

Table 1-1 lists the specifications of the HP 2105A, HP 2108A, and HP 2112A Microprogrammable Processors and the HP 2101A and HP 2102A Memory Systems. All three processors have been approved by the Underwriters' Laboratories (UL) and the Canadian Standards Association (CSA).

Table 1-2 lists the options and accessories available to expand or enhance the computer system. On an original order, specify the desired system configuration by option number. For a field upgrade of an existing system, specify the system addition by accessory number.

Table 1-1. Specifications

PROCESSOR	
CONTROL STORE	
Type:	Bipolar LSI ROM semiconductor.
Size:	Up to sixteen 256-word modules.
CONTROL PROCESSOR	
Address Space:	4,096 words.
Word Size:	24 bits.
Word Formats:	Four.
Word Fields:	Five.
ROM Cycle:	325 nanoseconds.
REGISTERS	
Accumulators:	Two (A and B), 16 bits each. Explicitly addressable; also implicitly addressable as memory.
Index:	Two (X and Y), 16 bits each.
Memory Control:	Two (T and P), 16 bits each; one (M) 15 bits.
Supplementary:	Two (overflow and extend), one bit each.
Manual Data:	One (display), 16 bits.
Scratch Pads:	Twelve, 16 bits each.
DYNAMIC MAPPING SYSTEM (OPTION)	
Installation:	Plugs into slots 111 and 112 of HP 2108A and HP 2112A memory PCA cage; control store module mounts on bottom of CPU.
Address Space:	1,048,576 words.
Program Security:	Read and write protection; programmable base page fence protection.
Memory Cycle Time:	650 nanoseconds.
System Control:	38 machine instructions.
Memory Maps:	Four dynamically alterable maps; two for program execution and two for Dual-Channel Port Controller. Allows DCPC to communicate to an area separate from program space; loading and unloading may be done from noncontiguous memory segments.
Map Registers:	32 registers per map; 12 bits per register.
Parity Error Interrupt:	Yes.
Program Compatibility:	Compatible with existing software for HP 21MX and HP 2100 Computers.
I/O Control:	Operating system may be given complete control of I/O logic.
MEMORY PARITY CHECK	
HP 2105A Processor:	Monitors all words read from memory. Switch selectable to either halt or ignore parity when detected. A parity indication is displayed on operator panel.
HP 2108A and HP 2112A Processors:	Same as for HP 2105A. With memory protect or DMS option, interrupt on parity error occurs.
POWER FAIL INTERRUPT	
Priority:	Highest priority interrupt.
Power Failure:	Detects power failure and generates an interrupt to trap cell for user-written power-failure routine. A minimum of 500 μs is available for the routine. Automatic restart is provided as a memory system option.

Table 1-1. Specifications (Continued)

PROCESSOR (Continued)

PROTECTION

Loaders:	All loaders reside in special ROM's separate from control ROM and are loaded into last 64 words of logical main memory by activating operator panel switches. Paper tape loader is standard; three additional switch-selectable loader spaces are provided to accommodate other modes of operation as a user option. User-generated loaders may be written in Assembly Language.
Volatility:	Mains ac standby mode and sustaining power for line loss of 2.5 cycles at 60 Hz before entering power fail routine. Power fail recovery is a memory system option.

INPUT/OUTPUT

Priority Interrupt:	Multilevel vectored priority interrupt determined by interface channel assignment.
I/O Channels:	HP 2105A: four internal I/O channels; expandable to 36 channels with two I/O extenders. HP 2108A: nine internal I/O channels; expandable to 41 channels with two I/O extenders. HP 2112A: 14 internal I/O channels; expandable to 46 channels with two I/O extenders.
Current Available to I/O:	

SUPPLY	HP 2105A	HP 2108A	HP 2112A
+5V	6.0V	13.0A	18.0A
-2V	2.0A	4.0A	9.1A
+12V	1.0A	1.5A	3.0A
-12V	1.0A	1.5A	3.0A

Note: Current availability to I/O assumes maximum memory in mainframe, Dual-Channel Port Controller, and maximum available control store mounted to CPU.

PHYSICAL CHARACTERISTICS

Width:	16-3/4 inches (42.55 cm) behind rack mount; 19 inches (48.26 cm) operator panel width on sides.
Depth:	23-1/2 inches (59.69 cm); 23 inches (58.42 cm) behind operator panel.
Height:	HP 2105A: 5-1/4 inches (13.31 cm) in rack mount. HP 2108A: 8-3/4 inches (22.23 cm) in rack mount. HP 2112A: 12-1/4 inches (30.87 cm) in rack mount.
Weight:	HP 2105A: 39 pounds (17.69 kg). HP 2108A: 45 pounds (20.41 kg). HP 2112A: 75 pounds (34.02 kg).

ELECTRICAL CHARACTERISTICS

Input Line Voltage:	110V or 220V ac (±20%), single phase.
Line Frequency:	47 to 66 Hz.
Power:	HP 2105A: 400W maximum. HP 2108A: 525W maximum. HP 2112A: 800W maximum.
Line Overvoltage Protect:	Input crowbar in series with line breaker.
Output Protect:	All voltages protected against overvoltage and overcurrent.
Output Voltage Regulation:	±5%.
Thermal Sensing:	Monitors internal temperature and automatically shuts down if temperature exceeds specified level.

Table 1-1. Specifications (Continued)

PROCESSOR (Continued)	
ENVIRONMENTAL LIMITATIONS	
Ambient Temperature:	Operating: 32° to 131°F (0° to 55°C). Nonoperating: -40° to 167°F (-40° to 75°C).
Altitude:	Operating: 15,000 feet (4,573 meters). Nonoperating: 25,000 feet (7,622 meters).
Relative Humidity:	20 to 95% at 77° to 104°F (25° to 40°C).
Shock:	Tested for 30g shock for 11 milliseconds over a 1/2 sine wave shape.
Vibration:	Can withstand vibration of 1g at 44 cycles per second.
VENTILATION	
Air Flow:	Intake on left-hand side; exhaust on right-hand side.
Heat Dissipation:	HP 2105A: 1365 BTU's (344 kilocalories)/hour, max. HP 2108A: 1795 BTU's (452 kilocalories)/hour, max. HP 2112A: 2732 BTU's (688 kilocalories)/hour, max.
MEMORY SYSTEMS	
HP 2101A MEMORY	
Density:	High density; 16K words per module.
Configuration:	Available in 8K or 16K configuration; only one 8K module allowed per system.
HP 2102A MEMORY	
Density:	Medium or high density:
Configuration:	Available in 4K, 8K, and 16K configurations; only one 4K module allowed per system.
MEMORY ORGANIZATION	
Type:	4K chip N-channel MOS/RAM semiconductor.
Word Size:	16 bits plus parity bit.
Configuration:	Controller and multiple plug-in memory modules.
Page Size:	1,024 words.
Address Space:	1,048,576 words.
System Cycle Time:	650 nanoseconds.
Volatility Protection:	Mains ac standby mode and sustaining power for line loss of 160 milliseconds is standard. Power fail recovery system is optional.
MEMORY PROTECT	
Installation:	Plugs into slot 111 of HP 2108A or HP 2112A memory PCA cage.
Priority:	Second highest priority interrupt (shared with memory parity).
Operation:	Initiated under programmed control; protects any amount of memory, I/O, or privileged instruction whem implemented in the HP 2108A or HP 2112A Processor.
Fence Register:	Set under program control; memory below fence is protected.
Interrupt:	Interrupts to trap cell for subroutine when user program (1) attempts to alter a protected location, (2) attempts to jump into the protected area, or (3) attempts to execute an I/O instruction.
Violation Register:	Contains memory address of violating instruction.
Parity Error Interrupt:	Provides interrupt signal when parity error is detected; saves address of error in violation register.
Infinite Indirect Protect:	Interrupts are enabled after three levels of indirect addressing.

Table 1-1. Specifications (Continued)

MEMORY SYSTEMS (Continued)	
DUAL-CHANNEL PORT CONTROLLER	
Installation:	Plugs into slot 110 of memory PCA cage.
Number of Channels:	Two.
Number of Memory Ports:	One.
Registers/Channel:	Two (word count and address).
Word Size:	16 bits.
Maximum Block Size:	32,768 words.
I/O Assignable:	Assignable to any two I/O channels; all logic necessary to facilitate bidirectional direct memory to and from I/O is contained on this controller.
Transfer Rate:	616,666 words per second maximum.
Priority:	Highest: DCPC Channel 1. Middle: DCPC Channel 2. Lowest: Processor.
POWER FAIL RECOVERY SYSTEM	
Power Restart:	Detects resumption of power and generates an interrupt to trap cell for user-written restart program which has been protected in memory by the sustaining battery.
Power Control and Charge Unit:	Monitors battery charge status and provides trickle charge.
Sustaining Battery:	Type: 12V nickel cadmium. Charging rate: 350 milliamperes. Capacity: 4 ampere-hours; will sustain 128K of HP 2101A or 64K of HP 2102A main memory for 2 hours.

Table 1-2. Options and Accessories

DESCRIPTION	OPTION NO.	ACCESSORY NO.
HP 2105A PROCESSOR		
Fast FORTRAN Processor (FFP)	-003	12977A
Writable Control Store (WCS)	-005	12978A
Disc Loader ROM	-014	12992A
230V, 50-Hz Operation	-015	– –
User Control Store Board	– –	12945A
Programmable ROM (pROM) Writer	– –	12909B
Slide Mounting Kit	– –	12903A
HP 2108A and HP 2112A PROCESSORS		
Fast FORTRAN Processor (FFP)	-003	12977A
Dynamic Mapping System (DMS)	-004	12976A
Writable Control Store (WCS)	-005	12978A
DMS with FFP (combines -003 and -004)	-006	– –
Disc Loader ROM	-014	12992A
230V, 50-Hz Operation	-015	– –
User Control Store Board	– –	12945A
Programmable ROM (pROM) Writer	– –	12909B
Slide Mounting Kit (HP 2108A)	– –	12903B
Slide Mounting Kit (HP 2112A)	– –	12903C
HP 2101A MOS MEMORY SYSTEM		
Dual-Channel Port Controller (DCPC)	-001	12897A
Memory Protect*	-003	12892A
8K Memory Module	-008	12999A
16K Memory Module	-116	12997A
Memory System Cable	– –	12993A
Power Fail Recovery System (HP 2105A and HP 2108A)	– –	12944A
Power Fail Recovery System (HP 2112A)	– –	12991A
HP 2102A MOS MEMORY SYSTEM		
Dual-Channel Port Controller (DCPC)	-001	12897A
Memory Protect*	-003	12892A
4K Memory Module	-004	12894A
8K Memory Module	-008	12998A
16K Memory Module	-016	13187A
Memory System Cable	– –	12993A
Power Fail Recovery System (HP 2105A and HP 2108A)	– –	12944A
Power Fail Recovery System (HP 2112A)	– –	12991A
HP 12979A INPUT/OUTPUT EXTENDER		
Dual-Channel Port Controller (DCPC)	-001	12898A
2nd I/O Extender	-010	12979A
230V, 50-Hz Operation	-015	– –
Slide Mounting Kit	– –	12903B
HP 12990A MEMORY EXTENDER		– –
230V, 50-Hz Operation	-015	– –
Slide Mounting Kit	– –	12903B

*Memory Protect is included with processor options –004 and –006. Memory Protect cannot be used with HP 2105A Processor.

OPERATING FEATURES

SECTION

II

This section describes the hardware registers accessible to the programmer and the functions of the various operating controls and indicators. Also included are basic operating examples such as a cold start procedure to load a program via a punched-tape reader, manually loading a short program via the operator panel, and running a program after it has been loaded into memory. These examples are included only to illustrate the simplicity of operation. For detailed operating procedures, refer to the *HP 21MX Computer Series Operator's Manual,* part no. 02108-90004.

The computer has eight 16-bit working registers which can be selected for display and modification by operator panel controls; two 1-bit registers; and one 16-bit display register. The functions of these registers are described in following paragraphs.

2-2. A-REGISTER

The A-register is a 16-bit accumulator that holds the results of arithmetic and logical operations performed by programmed instructions. This register can be addressed directly by any memory reference instruction as location 000000 (octal), thus permitting interrelated operations with the B-register (e.g., "add B to A," "compare B with A," etc.) using a single-word instruction.

2-3. B-REGISTER

The B-register is a second 16-bit accumulator, which can hold the results of arithmetic and logic operations completely independent of the A-register. The B-register can be addressed directly by any memory reference instruction as location 000001 (octal) for interrelated operations with the A-register.

2-4. M-REGISTER

The M-register holds the address of the memory cell currently being read from or written into by the CPU.

2-5. T-REGISTER

All data transferred into or out of memory is routed through the T-register. When displayed, the T-register indicates the contents of the memory location currently pointed to by the M-register. The A- or B-register contents are displayed if the M-register contents are 000000 or 000001, respectively.

2-6. P-REGISTER

The P-register holds the address of the next instruction to be fetched from memory. Since this is a "lookahead" register, the P-register contents will frequently differ from the M-register contents.

2-7. S-REGISTER

The S-register is a 16-bit utility register. In the halt or run mode, the S-register can be loaded via the display register. In the run mode, the S-register can be addressed as an input/output device (select code 01) and can input and output data to and from the A- and B-registers.

2-8. EXTEND REGISTER

The one-bit extend register is used to link the A- and B-registers by rotate instructions or to indicate a carry from the most-significant bit (bit 15) of the A- or B-register by an add instruction (ADA, ADB) or an increment instruction (INA, INB, but not ISZ). This is of significance primarily for multiple-precision arithmetic operations. If already set (logic 1), the extend bit cannot be cleared by a carry. However, the extend bit can be selectively set, cleared, complemented, or tested by programmed instructions. When the operator panel EXTEND indicator is lighted, the extend bit is set.

2-9. OVERFLOW REGISTER

The one-bit overflow register is used to indicate that an add instruction (ADA, ADB), divide instruction (DIV), or an increment instruction (INA, INB, but not ISZ) referencing the A- or B-register has caused (or will cause) the accumulators to exceed the maximum positive or negative number that can be contained in these registers. The overflow bit can be selectively set, cleared, or tested by programmed instructions. The operator panel OVERFLOW indicator will remain lighted until the overflow is cleared. The overflow bit cannot be set by any shift or rotate instruction except Arithmetic Shift Left (ASL).

2-10. DISPLAY REGISTER

The display register, which is included on the operator panel, provides a means of displaying and modifying the contents of the six 16-bit working registers when the computer is in the halt mode. An illuminating indicator is located directly above each of the 16 bit switches; a lighted indicator denotes a logic 1 and an unlighted indicator denotes a logic 0. When the computer is in the run mode, the contents of the S-register are displayed automatically.

2-11. X- AND Y-REGISTERS

These two 16-bit registers, designated X and Y, are accessed through the use of 30 index register instructions and 2 jump instructions described under paragraphs 3-22 and 3-23, respectively. These registers may also be accessed from the operator panel by entering the special register display mode described under paragraph 2-20.

2-13. OPERATOR PANEL

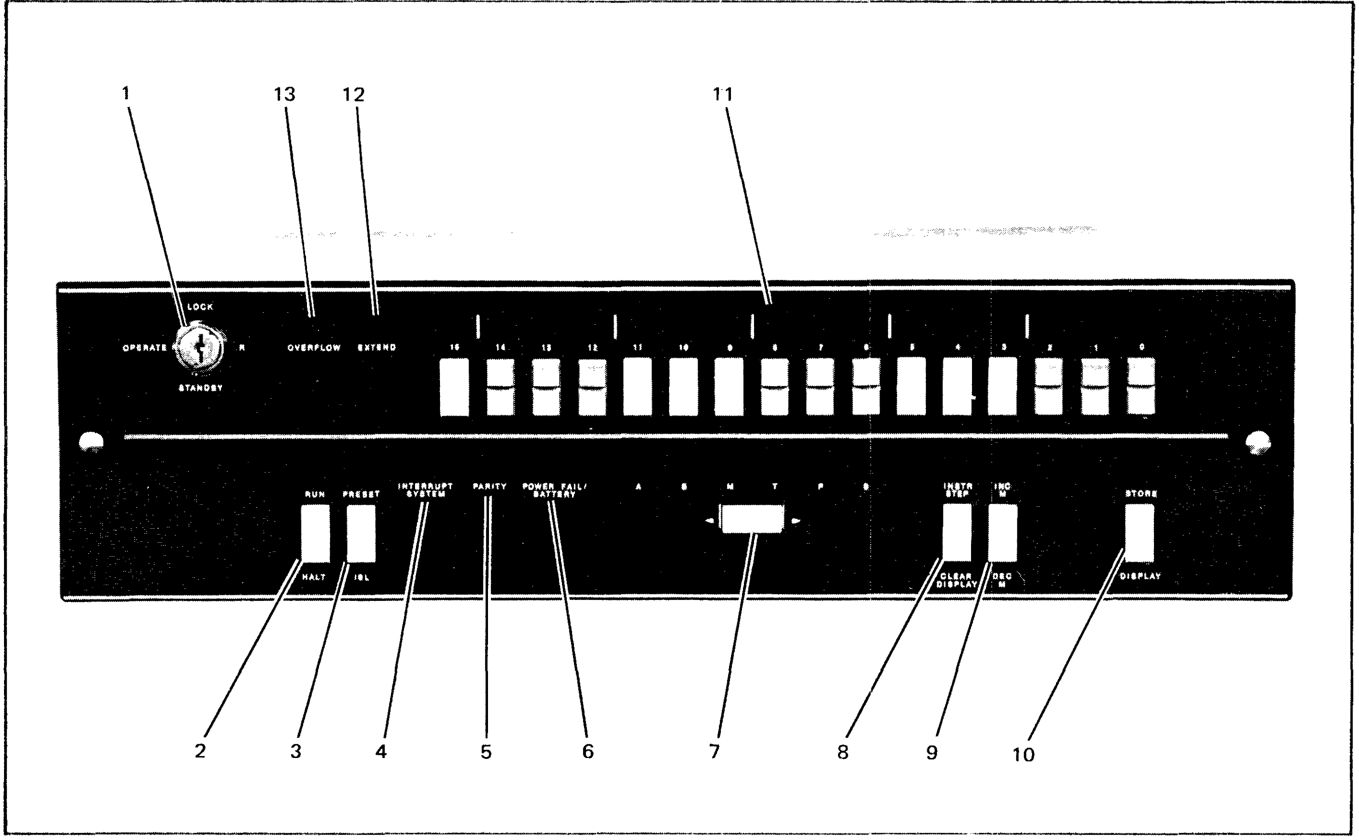
The location and function of the various controls and indicators mounted on the operator panel are illustrated in figure 2-1 and described in table 2-1. All operator panel controls (except the key-operated switch) are two-position, momentary-contact rocker switches; the status of the computer is displayed by light-emitting diodes.

2-14. REAR PANEL

Two switches are located on the rear panel. The ~LINE switch controls the application of ac power to the computer power supply and the ventilating fans. The BATTERY switch allows the optional battery to be disconnected to prevent discharge during extended periods of nonoperation.

2-15. INTERNAL SWITCHES

Two toggle switches are mounted on the rear of the central processor unit printed-circuit assembly (PCA). The setting of the ARS/$\overline{\text{ARS}}$ switch determines the action that the computer will take in the event of a primary power failure



2270-1

Figure 2-1. Operator Panel Controls and Indicators

Table 2-1. Operator Panel Control and Indicator Functions

<table>
<tr><th>FIG. 2-1, INDEX NO.</th><th>NAME</th><th>FUNCTION</th></tr>
<tr><td>1</td><td>STANDBY/OPERATE/ LOCK/R</td><td>Four-position, key-operated switch; does <i>not</i> control the application of ac power to the mainframe. (The ~LINE switch on the rear panel controls the ac power input to the processor power supply and ventilating fans.)

STANDBY. Memory contents are sustained and the battery is on charge. CPU and I/O power are off; I/O interfaces may be installed or removed without damage. Key is removable.

OPERATE. Power is supplied to the entire mainframe. Key is not removable.

LOCK. The RUN and HALT switches are disabled; all other functions are enabled (within the constraints of the run/halt modes). Key is removable.

R (reset). If memory data becomes invalid due to a prolonged power failure, the automatic restoration of power will preclude the CPU from running without operator intervention because the Power On (PON) signal is held off. (This condition will cause the POWER FAIL/BATTERY indicator to be lighted steadily; i.e., not blinking.) Rotating this switch momentarily to R allows the PON signal to be generated and reset this condition. Key is not removable.</td></tr>
<tr><td>2</td><td>RUN/HALT</td><td>RUN. Starts CPU and lights the RUN indicator. All operator panel functions are disabled except Display Register, CLEAR DISPLAY, and HALT. Pressing RUN automatically causes the S-register contents to be displayed, and no other register can be selected during the run mode; thus, the Display Register effectively becomes the S-register, which may be addressed as select code 01 by the program.

HALT. Halts the computer at the end of the current instruction and turns off the RUN indicator. All other operator panel controls become enabled. The T-register is selected automatically for display.</td></tr>
<tr><td>3</td><td>PRESET/IBL</td><td>PRESET. Disables the interrupt system and clears the parity indicator and overflow bit (if set). From I/O channel 06 up, clears control flip-flops and sets flags. Pressing and holding PRESET upon the restoration of power will force a $\overline{\text{ARS}}$ condition (see paragraph 3-43).

IBL (initial binary loader). Causes the contents of the standard paper tape loader ROM or the optional loader ROM's to be written into the uppermost 64 memory locations. Bits 15 and 14 of the S-register select the desired loader ROM as follows:
<table>
<tr><th colspan="2">BITS</th><th rowspan="2">LOADER SELECTED</th></tr>
<tr><th>15</th><th>14</th></tr>
<tr><td>0</td><td>0</td><td>Standard paper tape loader ROM</td></tr>
<tr><td>0</td><td>1</td><td>Option loader 1 ROM</td></tr>
<tr><td>1</td><td>0</td><td>Option loader 2 ROM</td></tr>
<tr><td>1</td><td>1</td><td>Option loader 3 ROM</td></tr>
</table>
Bits 6 through 11 of the S-register must be set to the octal select code of the loading device.</td></tr>
</table>

Table 2-1. Operator Panel Control and Indicator Functions (Continued)

FIG. 2-1, INDEX NO.	NAME	FUNCTION
4	INTERRUPT SYSTEM	Indicates the status of the interrupt system. When lighted, the interrupt system is enabled (Flag set); when turned off, the interrupt system is disabled (Flag clear).
5	PARITY	Lights when a parity error occurs as a result of reading from memory. In the halt mode, the light can be turned off by pressing the PRESET switch. With the memory protect or DMS option* installed and the parity error interrupt enabled, the indicator is turned off automatically by a parity error interrupt and is therefore not ordinarily lighted long enough to be visible.
6	POWER FAIL/BATTERY	If the power fail/automatic restart feature is enabled (i.e., internal ARS/$\overline{\text{ARS}}$ switch is set to ARS position as described in section III), the indicator will light when power is restored. This light can be turned off by pressing the PRESET switch in the halt mode. This light will flash on and off upon the application of mains power until the battery is known to contain enough power to sustain memory.
7	◀ Register Select ▶	In the halt mode, this switch allows any one of the six working registers (A, B, M, T, P, or S) to be selected for display and modification. Pressing the left half (◀) of the switch moves the "dot" indicator left; pressing the right half (▶) of the switch moves the "dot" indicator right. The register currently selected is indicated by the appropriate indicator light. After a programmed or manual halt, the T-register is selected automatically for display. In this case, the T-register holds the contents of the last accessed memory cell. In the case of a programmed halt, the halt instruction will be displayed.
8	INSTR STEP/CLEAR DISPLAY	INSTR STEP. Pressing and releasing this switch while in the halt mode advances the program to the next instruction. If the T-register indicator lights when the switch is released, infinite indirect addressing is indicated. Actuating this switch does not actually place the computer in the run mode. CLEAR DISPLAY. In the run or halt mode, clears the Display Register; i.e., contents become 000000.
9	INC M/DEC M	INC M. In the halt mode, increments the M-register contents. DEC M. In the halt mode, decrements the M-register contents. Note: Incrementing and decrementing occur even when the M-register is not displayed.

*These options not available for HP 2105A Processor.

Table 2-1. Operator Panel Control and Indicator Functions (Continued)

FIG. 2-1, INDEX NO.	NAME	FUNCTION
10	STORE/DISPLAY	STORE. In the halt mode, stores the contents of the Display Register into the selected working register (A, B, M, T, P, or S). If the Register Select "dot" is pointing to T and STORE is pressed, the contents of the Display Register will be loaded into memory cell m, the M-register will be incremented automatically to m + 1, and the Display Register will *not* be updated. This latter feature allows the same data to be stored in consecutive memory locations (e.g., halts in the trap cells, same word into a buffer, etc.). If the Register Select "dot" is pointing to any register other than T, only that one register will be updated when STORE is pressed. DISPLAY. Places the present contents of the selected register into the Display Register. Used to recall a register after the Display Register contents have been changed or to display the next contents of the T-register after STORE is pressed.
11	Display Register	In the halt mode, displays the contents of the register currently pointed to by the Register Select "dot;" only the S-register is displayed during the run mode. A logic 1 is signified when the displayed bit indicator is lighted; a logic 0 is signified when the displayed bit indicator is not lighted. Pressing the upper half of the switch sets that bit to a logic 1; pressing the lower half of the switch sets that bit to a logic 0. The Display Register is cleared to all zeros when the CLEAR DISPLAY switch is pressed.
12	EXTEND	In both the run and halt modes, continuously displays the contents of the extend register. When lighted, the extend bit is set (logic 1).
13	OVERFLOW	In both the run and halt modes, continuously displays the content of the overflow register. When lighted, the overflow bit is set (logic 1).

and the setting of the HLT PE/INT-IGNORE switch determines the action to be taken in the event of a parity error or memory protect violation. Programming considerations concerning these switches are given in section III.

2-17. COLD START PROCEDURE

This procedure describes a cold start using the standard paper tape loader ROM, which will allow a program to be loaded via a punched-tape reader. At the rear of the computer, set the ~LINE and BATTERY switches to ON and proceed as follows:

a. On the operator panel, set key-operated switch to OPERATE.

b. Press left half (◀) or right half (▶) of Register Select switch to select S-register.

c. Press CLEAR DISPLAY and set bits 6 through 11 to display octal select code of tape reader.

d. Set bits 15 and 14 to zeros to select standard paper tape loader ROM.

e. Press STORE and then press IBL. The paper tape loader is now loaded into the uppermost 64 locations of memory and the select code of the tape reader is patched according to the contents of the S-register. The P-register is now pointing to the first instruction of the loader.

f. Turn on tape reader and prepare it for reading. Press PRESET and then press RUN. The program will now be read into memory and the computer will halt with the T-register selected automatically. A successful load is indicated if the Display Register contents are 102077 (octal).

2-18. MANUAL LOADING

Short programs can be loaded manually from the operator panel as follows:

a. Press left half (◀) or right half (▶) of Register Select switch to select M-register.

b. Press CLEAR DISPLAY and set Display Register to starting address of program.

c. Press STORE. Select T-register and change contents of Display Register to binary code of first instruction to be loaded; press STORE.

d. Enter next instruction in Display Register and press STORE. (Pressing STORE with T-register selected automatically increments M-register.)

e. Repeat step d until entire program has been loaded.

2-19. RUNNING PROGRAMS

To run a program after it has been loaded, proceed as follows:

a. Press left half (◀) or right half (▶) of Register Select switch to select P-register.

b. Press CLEAR DISPLAY and set Display Register to starting address of program.

c. Press STORE, PRESET, and RUN.

The RUN indicator will remain lighted as long as the program is running. If the key-operated switch is set to OPERATE, all operator panel controls except the Display Register, CLEAR DISPLAY, and HALT switches are disabled. The S-register is displayed automatically and can be changed manually via the Display Register bit switches. If the key-operated switch is set to LOCK, the RUN and HALT switches are disabled and all other switches are enabled within the constraints of the run and halt modes.

The special register display mode provides the capability of displaying and/or modifying the contents of the following: X and Y registers, scratch pads S3 through S12, CPU counter, overflow and extend registers, and all the optional Dynamic Mapping System (DMS) map registers. To enter the special register display mode, proceed as follows:

a. Press left half (◀) or right half (▶) of Register Select switch to select M-register.

b. Press CLEAR DISPLAY and set bit 15 to a logic 1 (also set bit 14 to a logic 1 if a DMS map register is to be displayed). *Do not press* STORE. (If STORE is pressed, bit 15 will be automatically cleared because the M-register is only 15 bits long. This prevents accidental entry into the special register display mode during normal operation.)

c. Set low-order bits as shown in figure 2-2 to select desired register. *Do not press* STORE.

d. Press right half (▶) of Register Select switch to select T-register. The special register display mode is now entered and the contents of the desired register are displayed.

Once the special register display mode is entered, the register pointer will be displayed when "M" is selected. (The M-register is not affected in this mode.) Pressing INC M will increment the pointer by one. Pressing DEC M will decrement the low-order bits of the pointer modulo 256_{10}; e.g., if the low-order bits are all zeros, pressing DEC M will set the eight low-order bits to all ones.

When "T" is selected, pressing INC M or DEC M will increment or decrement the low-order bits of the pointer. If bits 15-14 are 11_2, bit 6-0 are counted modulo 128_{10} (the number of DMS map registers); if bits 15-14 are 10_2, bits 3-0 are counted modulo 16_{10} (the number of displayable registers). In either case, the unused bits are masked to zeros. These count features maintain the pointer within the range of the number of registers accessible and prevent INC M and DEC M from affecting bits 15-14 of the pointer.

Table 2-2 lists the effects that the operator panel switches have while in the special register display mode and the various ways of reentering the normal register (A, B, M, T, P, S) display mode; table 2-3 lists the various ways of selecting, displaying, and modifying the registers.

REGISTER DESIRED	**POINTER**						
	15	14		3	2	1	0
X	1	0		0	0	0	0
Y	1	0		0	0	0	1
COUNTER	1	0		0	0	1	0
S3	1	0		0	0	1	1
S4	1	0		0	1	0	0
S5	1	0		0	1	0	1
S6	1	0		0	1	1	0
S7	1	0		0	1	1	1
S8	1	0		1	0	0	0
S9	1	0		1	0	0	1
S10	1	0		1	0	1	0
S11	1	0		1	0	1	1
S12	1	0		1	1	0	0
OVERFLOW	1	0		1	1	1	0
EXTEND	1	0		1	1	1	1

DMS MAPS	**POINTER**					
	15	14		6	5	4 3 2 1 0
SYSTEM	1	1		0	0	MAP REG NO.
USER	1	1		0	1	MAP REG NO.
PORT A	1	1		1	0	MAP REG NO.
PORT B	1	1		1	1	MAP REG NO.

Figure 2-2. Special Register Display Mode Pointers

Table 2-2. Special Register Display Mode Switch Operation

SELECTED FOR DISPLAY	SWITCH PRESSED	EFFECT
T	*▶	Register Select "dot" shifts to "P". P-register contents are displayed and special register display mode is terminated.
T	◀	Register Select "dot" shifts to "M". Pointer is displayed per figure 2-2.
T	DISPLAY	Contents of register selected by pointer are displayed per table 2-3. Pointer is unchanged.
T	STORE	Register selected by pointer is loaded with data per table 2-3.
T	INC M	Pointer is incremented by one. Contents of register selected by new pointer value are displayed per table 2-3.
T	DEC M	Pointer is decremented by one. Contents of register selected by new pointer value are displayed per table 2-3.
T	*PRESET	Same as for normal register display mode except display is left unaltered; special register display mode is terminated. (The M-register is displayed if "M" is selected by pressing ◀.)
T	*IBL	Same as normal register display mode; special register display mode is terminated. Contents of last referenced memory address are displayed.
T	*INSTR STEP	Executes the next machine instruction; special register display mode is terminated. Contents of last referenced memory address are displayed.
M	▶	Register Select "dot" shifts to "T". Special register select mode is entered (only if bit 15 = 1) and contents of register selected by pointer are displayed.
M	*◀	Register Select "dot" shifts to "B" and contents of B-register are displayed. Special register display mode is terminated.
M	DISPLAY	Contents of the pointer are restored to the display. This is useful for checking the pointer after the display has been changed by the operator.
M	*STORE	Contents of the display are stored into the M-register. Bit 15 is cleared and the special register display mode is terminated.
M	INC M	Pointer is incremented and displayed.
M	DEC M	Low-order bits of pointer are decremented modulo 256_{10} and displayed.
M	*PRESET	Preset is performed. Special register display mode is terminated but display is unchanged. (Special register display mode may be reentered by pressing ▶.)
M	*IBL	Same as normal register display mode except M-register contents are displayed and special register display mode is terminated.
M or T	*RUN	Same as normal register display mode; special register display mode is terminated.
M	*INSTR STEP	Executes the next machine instruction; special register display mode is terminated. Latest value of M-register (last referenced memory address) is displayed.

*Indicates conditions that terminate special register display mode.

Table 2-3. Effects of Storing/Displaying Special Registers

REGISTER	SELECTED BY DISPLAY, INC M, DEC M, ▶	IF STORE PRESSED WHILE SELECTED
X, Y, S3-S12	Contents of selected register (16 bits) displayed.	Contents of display are loaded into selected register. The display is not altered.
Counter	Counter state is displayed modulo 256_{10} in bits 7-0. Bits 15-8 are all ones.	Bits 7-0 of display are loaded into counter. The display is not altered.
Overflow and Extend	Display will be 177777_8.	Set bit 0 to the desired state and press STORE. The overflow or extend register will be set equal to bit 0 of the display. The display is not altered.
DMS Map Register	The contents of the map register indicated by bits 6-0 of the pointer are displayed. Bits 9-0 of the display indicate the memory page number. If bit 15 = 1, that page is read-protected; if bit 14 = 1, that page is write-protected. If DMS is not installed, the display will be 177777_8.	The contents of the display are stored into the map register indicated by bits 6-0 of the pointer in the same format as described at left. The display is not altered. Read and write protection may be set with bits 15 and 14, respectively.

PROGRAMMING INFORMATION

SECTION III

This section describes the software data formats and machine-language instruction coding required to operate the computer and its associated input/output system. A description of the vectored priority interrupt system is also included.

3-1. DATA FORMATS

As shown in figure 3-1, the basic data format is a 16-bit word in which bit positions are numbered from 0 through 15 in order of increasing significance. Bit position 15 of the data format is used for the sign bit; a logic 0 in this position indicates a positive number and a logic 1 in this position indicates a negative number. The data is assumed to be a whole number and the binary point is therefore assumed to be to the right of the number.

The basic word can also be divided into two 8-bit bytes or combined to form a 32-bit double word. The byte format is used for character-oriented input/output devices; packing two bytes of data into one 16-bit word is accomplished by software drivers. In I/O operations, the higher-order byte (byte 1) is the first to be transferred.

The integer double-word format is used for extended arithmetic in conjunction with the extended arithmetic instructions described under paragraphs 3-19 and 3-20. Bit position 15 of the most-significant word is the sign bit and the binary point is assumed to be to the right of the least-significant word. The integer value is expressed by the remaining 31 bits. When addressing a double word in memory, the address refers to the least-significant word location; the next higher memory address contains the most-significant word. When loaded into the accumulators, the B-register contains the most-significant word and the A-register contains the least-significant word.

The floating-point double-word format is used with floating-point software. Bit position 15 of the most-significant word is the mantissa sign and bit position 0 of the least-significant word is the exponent sign. Bits 1 through 7 of the least-significant word express the exponent and the remaining bits (bits 8 through 15 of the least-significant word and bits 0 through 14 of the most-significant word) express the mantissa. Since the mantissa is assumed to be a fractional value, the binary point appears to the left of the mantissa. Software drivers convert decimal numbers to this binary form and normalize the quantity expressed (sign and leading mantissa differ). If either the mantissa or the exponent is negative, that part is stored in two's complement form. The number must be in the apprriximate range of 10^{-38} to 10^{+38}. When loaded into the accumulators, the A-register contains the most-significant word and the B-register contains the least-significant word.

Figure 3-1 also illustrates the octal notation for both single-length (16-bit) and double-length (32-bit) words. Each group of three bits, beginning at the right, is combined to form an octal digit. A single-length (16-bit) word can therefore be fully expressed by six octal digits and a double-length (32-bit) word can be fully expressed by 11 octal digits. Octal notation is not shown for byte or floating-point formats, since bytes normally represent characters and floating-point numbers are given in decimal form.

The range of representable numbers for single-word data is +32,767 to -32,768 (decimal) or +77,777 to -100,000 (octal). The range of representable numbers for double-word integer data is +2,147,483,647 to -2,147,483,648 (decimal) or +17,777,777,777 to -20,000,000,000 (octal).

3-2. MEMORY ADDRESSING

3-3. PAGING

The computer memory is logically divided into pages of 1,024 words each. A page is defined as the largest block of memory that can be directly addressed by the address bits of a single-length memory reference instruction. (Refer to paragraph 3-8.) These memory reference instructions use 10 bits (bits 0 through 9) to specify a memory address; thus, the page size is 1,024 locations (2000 octal). Octal addresses for each page, up to a maximum memory size of 32K, are listed in table 3-1.

Provision is made to directly address one of two pages: page zero (the base page consisting of locations 00000 through 01777) and the current page (the page in which the instruction itself is located). Memory reference instructions reserve bit 10 to specify one or the other of these two pages. To address locations on any other page, indirect addressing is used as described in following paragraphs. Page references are specified by bit 10 as follows:

a. Logic 0 = Page Zero (Z).

b. Logic 1 = Current Page (C).

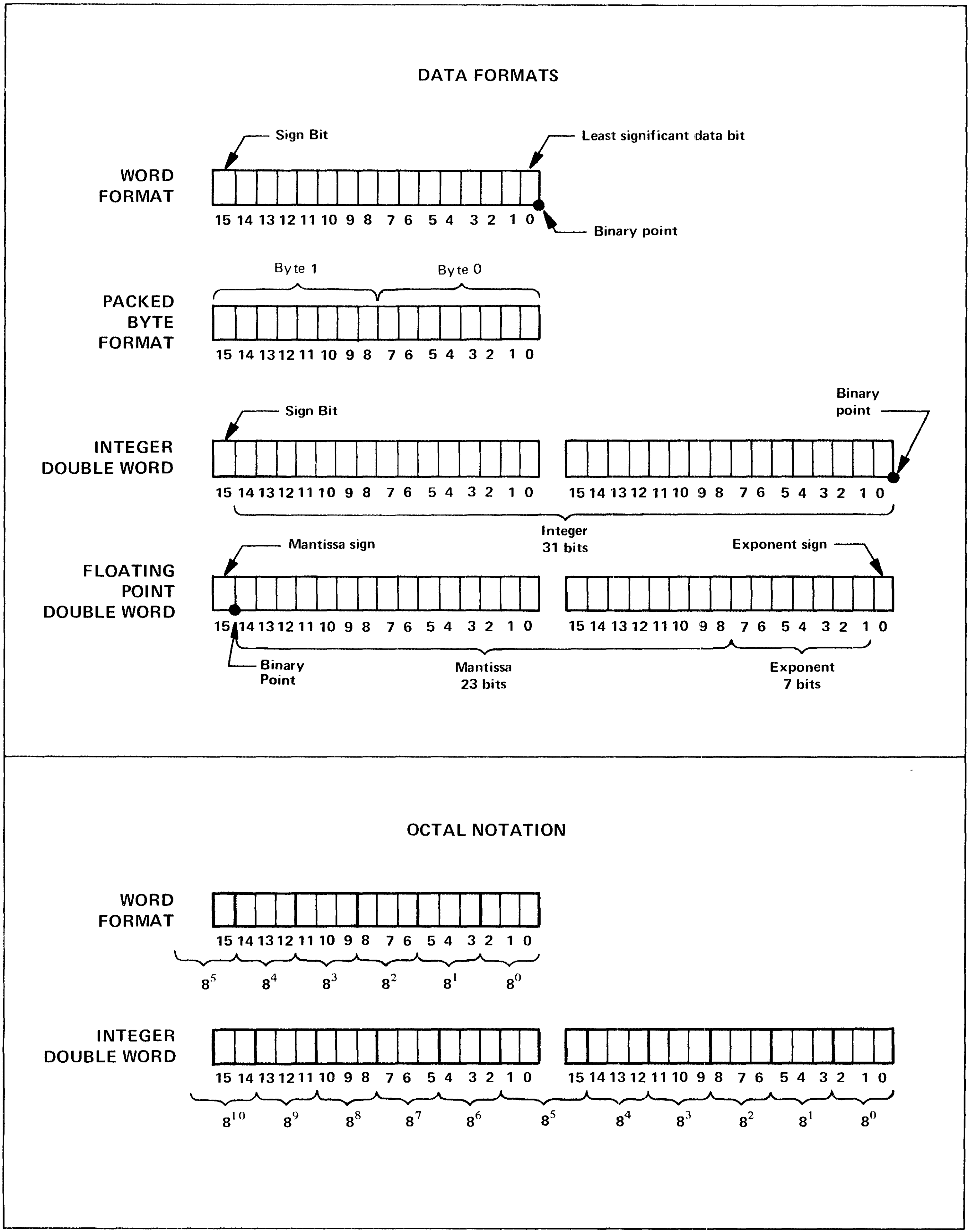


Figure 3-1. Data Formats and Octal Notation

Table 3-1. Memory Paging

MEMORY SIZE	PAGE	OCTAL ADDRESSES
	0	00000 to 01777
	1	02000 to 03777
	2	04000 to 05777
4K ↓	3	06000 to 07777
	4	10000 to 11777
	5	12000 to 13777
	6	14000 to 15777
8K ↓	7	16000 to 17777
	8	20000 to 21777
	9	22000 to 23777
	10	24000 to 25777
12K ↓	11	26000 to 27777
	12	30000 to 31777
	13	32000 to 33777
	14	34000 to 35777
16K ↓	15	36000 to 37777
	16	40000 to 41777
	17	42000 to 43777
	18	44000 to 45777
	19	46000 to 47777
	20	50000 to 51777
	21	52000 to 53777
	22	54000 to 55777
24K ↓	23	56000 to 57777
	24	60000 to 61777
	25	62000 to 63777
	26	64000 to 65777
	27	66000 to 67777
	28	70000 to 71777
	29	72000 to 73777
	30	74000 to 75777
32K ↓	31	76000 to 77777

3-4. DIRECT AND INDIRECT ADDRESSING

All memory reference instructions reserve bit 15 to specify either direct or indirect addressing. For single-length memory reference instructions, bit 15 of the instruction word is used; for extended arithmetic memory reference instructions, bit 15 of the address word is used. Indirect addressing uses the address part of the instruction to access another word in memory, which is taken as the new memory reference for the same instruction. This new address word is a full 16 bits long: 15 address bits plus another direct/indirect bit. The 15-bit length of the address permits access to any location in memory. If bit 15 again specifies indirect addressing, still another address is obtained; thus, multistep indirect addressing may be done to any number of levels. The first address obtained that does not specify another indirect level becomes the effective address for the instruction. Direct or indirect addressing is specified by bit 15 as follows:

a. Logic 0 = Direct (D).

b. Logic 1 = Indirect (I).

3-5. RESERVED MEMORY LOCATIONS

The first 64 memory locations of the base page (octal addresses 00000 through 00077) are reserved as listed in table 3-2. The first two locations are reserved as addresses for the two 16-bit accumulators (the A- and B-registers). Locations 00004 through 00077 are reserved for priority interrupts; as long as locations 00006 through 00077 do not have actual priority interrupt assignments, as determined by the options and input/output devices included in the system configuration, these locations can be used for programming purposes.

The uppermost 64 locations of memory for any given configuration are reserved for the initial binary loader. The initial binary loader is permanently resident in a read-only memory (ROM) and loaded into the uppermost 64 memory locations by a pushbutton switch on the operator panel. These 64 locations are not protected and can therefore be used for temporary storage of data, trap cells, buffers, etc.

Table 3-2. Reserved Memory Locations

MEMORY LOCATION	PURPOSE
00000	A-register address.
00001	B-register address.
00002-00003	Exit sequence if contents of A-register and B-register are used as executable words.
00004	Power-fail interrupt (highest priority).
00005	Memory parity and memory protect interrupt (HP 2108A and HP 2112A).
00006	Reserved for dual-channel port controller (DCPC) channel 1.
00007	Reserved for dual-channel port controller (DCPC) channel 2.
00010-00077	Interrupt locations in decreasing order of priority; e.g., location 00010 has priority over 00011.

3-6. NONEXISTENT MEMORY

Nonexistent memory is defined as those locations not physically implemented in the machine. Any attempt to write into a nonexistent memory location will be ignored (no operation). Any attempt to read from a nonexistent memory location will return an all-zeros word (000000 octal); no parity error occurs.

The base set of instructions are classified according to format. The five formats used are illustrated in figure 3-2 and described in following paragraphs. In all cases where a single bit is used to select one of two cases (e.g., D/I), the choice is made by coding a logic 0 or logic 1, respectively.

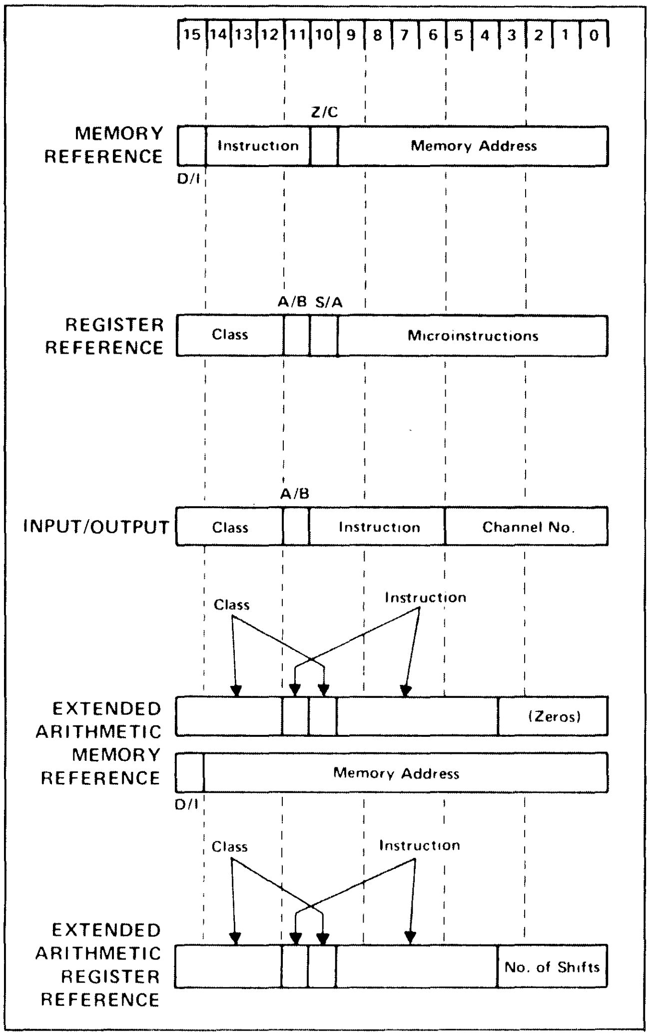


Figure 3-2. Base Set Instruction Formats

3-8. MEMORY REFERENCE INSTRUCTIONS

This class of instructions, which combines an instruction code and a memory address into one 16-bit word, is used to execute some function involving data in a specific memory location. Examples are storing, retrieving, and combining memory data to and from the accumulators (A- and B-registers) or causing the program to jump to a specified location in memory.

The memory cell referenced (i.e., the absolute address) is determined by a combination of 10 memory address bits (0 through 9) in the instruction word and 5 bits (10 through 14) assumed from the current contents of the M-register. This means that memory reference instructions can directly address any word in the current page; additionally, if the instruction is given in some location other than the base page (page zero), bit 10 (Z/C) of the instruction doubles the addressing range to 2,048 locations by allowing the selection of either page zero or the current page. (This causes bits 10 through 14 of the address contained in the M-register to be set to zero instead of assuming the current contents of the M-register.) This feature provides a convenient linkage between all pages of memory, since page zero can be reached directly from any other page.

As discussed under paragraph 3-4, bit 15 is used to specify direct or indirect memory addressing. Note also that since the A- and B-registers are addressable, any single-word memory reference instruction can apply to either of these registers as well as to memory cells. For example, an ADA 0001 instruction adds the contents of the B-register (address 0001) to the contents currently held in the A-register; specify page zero for these operations since the addresses of the A- and B-registers are on page zero.

3-9. REGISTER REFERENCE INSTRUCTIONS

In general, the register reference instructions manipulate bits in the A-register, B-register, and E-register; there is no reference to memory. This group includes 39 basic microinstructions which may be combined to form a one-word multiple instruction that can operate in various ways on the contents of the A-, B-, and E-registers. These 39 instructions are divided into two subgroups: the shift/rotate group (SRG) and the alter/skip group (ASG). The appropriate subgroup is specified by bit 10 (S/A). Typical operations are clear and/or complement a register, conditional skips, and register increment.

3-10. INPUT/OUTPUT INSTRUCTIONS

The input/output instructions use bits 6 through 11 for a variety of I/O instructions and bits 0 through 5 to apply the instructions to a specific I/O channel. This provides the means of controlling all peripherals connected to the I/O channels and for transferring data to and from these peripherals. Included also in this group are instructions to control the interrupt system, overflow bit, and computer halt.

3-11. EXTENDED ARITHMETIC MEMORY REFERENCE INSTRUCTIONS

As the single-word memory reference instruction described previously, the extended arithmetic memory reference instructions include an instruction code and a memory address. In this case, however, two words are required. The first word specifies the extended arithmetic class (bits 12 through 15 and 10) and the instruction code (bits 4 through 9 and 11); bits 0 through 3 are not needed and are coded with zeros. The second word specifies the memory address of the operand. Since the full 15 bits are used for the address, this type of instruction may directly address any location in memory. As with all memory reference instructions, bit 15 is used to specify direct or indirect addressing. Operations performed by this class of instructions are integer multiply and divide (using double-length product and dividend) and double load and double store.

3-12. EXTENDED ARITHMETIC REGISTER REFERENCE INSTRUCTIONS

This class of instructions provides long shifts and rotates on the combined contents of the A- and B-registers. Bits 12 through 15 and 10 identify the instruction class; bits 4 through 9 and 11 specify the direction and type of shift; and bits 0 through 3 control the number of shifts, which can range from 1 to 16 places.

3-13. BASE SET INSTRUCTION CODING

Machine language coding for the base set of instructions are provided in following paragraphs. Definitions for these instructions are grouped according to the instruction type: memory reference, register reference, input/output, extended arithmetic memory reference, and extended arithmetic register reference.

Directly above each definition is a diagram showing the machine language coding for that instruction. The gray shaded bits code the instruction type and the blue shaded bits code the specific instruction. Unshaded bits are further defined in the introduction to each instruction type. The mnemonic code and instruction name are included above each diagram.

In all cases where an additional bit is used to specify a secondary function (D/I, Z/C, or H/C), the choice is made by coding a logic 0 or logic 1, respectively. In other words, a logic 0 codes D (direct addressing), Z (zero page), or H (hold flag); a logic 1 codes I (indirect addressing), C (current page), or C (clear flag).

3-14. MEMORY REFERENCE INSTRUCTIONS

The following 14 memory reference instructions execute a function involving data in memory. Bits 0 through 9 specify the affected memory location on a given memory page or, if indirect addressing is specified, the next address to be referenced. Indirect addressing may be continued to any number of levels; when bit 15 (D/I) is a logic 0 (specifying direct addressing), that location will be taken as the effective address. The A- and B-registers may be addressed as locations 00000 and 00001 (octal), respectively.

If bit 10 (Z/C) is a logic 0, the memory address is on page zero; if bit 10 is a logic 1, the memory address is on the current page. If the A- or B-register is addressed, bit 10 must be a logic 0 to specify page zero, unless the current page is page zero.

ADA ADD TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	0	0	Z/C										

Memory Address

Adds the contents of the addressed memory location to the contents of the A-register. The sum remains in the A-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit.

ADB ADD TO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	0	1	Z/C										

Memory Address

Adds the contents of the addressed memory location to the contents of the B-register. The sum remains in the B-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit.

AND "AND" TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	0	1	0	Z/C										

Memory Address

Combines the contents of the addressed memory location and the contents of the A-register by performing a logical "and" operation. The contents of the memory cell are unaltered.

CPA

COMPARE TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	1	0	Z/C										

Memory Address

Compares the contents of the addressed memory location with the contents of the A-register. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances two counts instead of one count. If the two words are identical, the next sequential instruction is executed. Neither the A-register contents nor memory cell contents are altered.

CPB

COMPARE TO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	0	1	1	Z/C										

Memory Address

Compares the contents of the addressed memory location with the contents of the B-register. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances two counts instead of one count. If the two words are identical, the next sequential instruction is executed. Neither the B-register contents nor memory cell contents are altered.

IOR

"INCLUSIVE OR" TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	1	0	Z/C										

Memory Address

Combines the contents of the addressed memory location and the contents of the A-register by performing a logical "inclusive or" operation. The contents of the memory cell are unaltered.

ISZ

INCREMENT AND SKIP IF ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	1	1	Z/C										

Memory Address

Adds one to the contents of the addressed memory location. If the result of this operation is zero (memory contents incremented from 177777 to 000000), the next instruction is skipped; i.e., the P-register is advanced two counts instead of one count. If the result of this operation is not zero, the next sequential instruction is executed. In either case, the incremented value is written back into the memory cell. An ISZ instruction referencing locations 0000 or 0001 (A- or B-register) cannot set the extend bit or the overflow bit.

JMP

JUMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	0	1	Z/C										

Memory Address

Transfers control to the addressed memory location. That is, a JMP causes the P-register count to set according to the memory address portion of the JMP instruction so that the next instruction will be read from that location.

JSB

JUMP TO SUBROUTINE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	0	1	1	Z/C										

Memory Address

This instruction, executed in location P (P-register count), causes the computer control to jump unconditionally to the memory location (m) specified by the memory address portion of the JSB instruction. The contents of the P-register plus one (return address) is stored in memory location m, and the next instruction to be executed will be that contained in the next sequential memory location (m + 1). A return to the main program sequence at P + 1 will be effected by a JMP indirect through location m.

LDA

LOAD A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	0	0	Z/C										

Memory Address

Loads the contents of the addressed memory location into the A-register. The contents of the memory cell are unaltered.

LDB

LOAD B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	0	1	Z/C										

Memory Address

Loads the contents of the addressed memory location into the B-register. The contents of the memory cell are unaltered.

STA

STORE A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	1	0	Z/C										

Memory Address

Stores the contents of the A-register in the addressed memory location. The previous contents of the memory cell are lost; the A-register contents are unaltered.

STB STORE B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	1	1	1	1	Z/C										

Memory Address

Stores the contents of the B-register in the addressed memory location. The previous contents of the memory cell are lost; the B-register contents are unaltered.

XOR "EXCLUSIVE OR" TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/I	0	1	0	0	Z/C										

Memory Address

Combines the contents of the addressed memory location and the contents of the A-register by performing a logical "exclusive or" operation. The contents of the memory cell are unaltered.

3-15. REGISTER REFERENCE INSTRUCTIONS

The 39 register reference instructions execute functions on data contained in the A-register, B-register, and E-register. These instructions are divided into two groups: the shift/rotate group (SRG) and the alter/skip group (ASG). In each group, several instructions may be combined into one word and are thus individually termed "microinstructions." Since the two groups perform separate and distinct functions, microinstructions from the two groups cannot be mixed. Unshaded bits in the coding diagrams are available for combining other microinstructions.

3-16. SHIFT/ROTATE GROUP. The 20 instructions in the shift/rotate group (SRG) are defined first; this group is specified by setting bit 10 to a logic 0. A comparison of the various shift/rotate functions are illustrated in figure 3-3. Rules for combining microinstructions in this group are as follows (refer to table 3-3):

a. Only one microinstruction can be chosen from each of the two multiple-choice columns.

b. References can be made to either the A-register or B-register, but not both.

c. Sequence of execution is from left to right.

d. In machine code, use zeros to exclude unwanted microinstructions.

e. Code a logic 1 in bit position 9 to enable shifts or rotates in the first position; code a logic 1 in bit position 4 to enable shifts or rotates in the second position.

f. The extend bit is not affected unless specifically stated. However, if a "rotate-with-E" instruction (ELA, ELB, ERA, or ERB) is coded but disabled by a logic 0 in bit position 9 and/or position 4, the E-register will be updated even though the A- or B-register contents are not affected; to avoid this situation, code a "no operation" (three zeros) in the first and/or second positions.

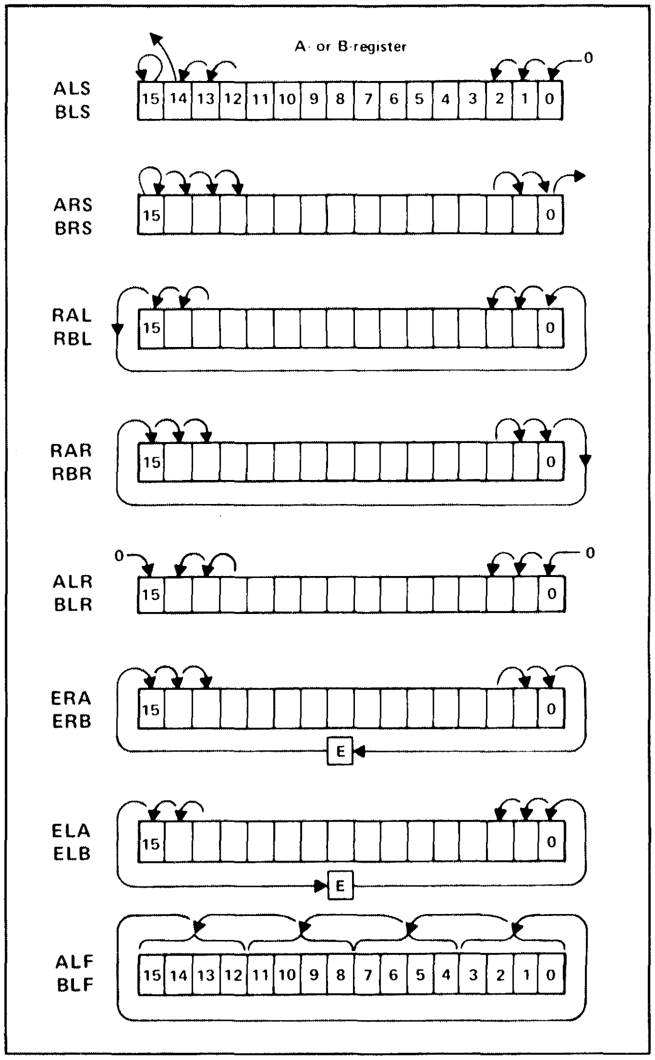
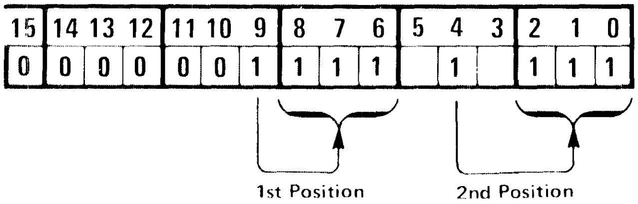


Figure 3-3. Shift and Rotate Functions

Table 3-3. Shift/Rotate Group Combining Guide

ALS ARS RAL RAR ALR ALF ERA ELA	[,CLE]	[,SLA] ,	ALS ARS RAL RAR ALR ALF ERA ELA
BLS BRS RBL RBR BLR BLF ERB ELB	[,CLE]	[,SLB] ,	BLS BRS RBL RBR BLR BLF ERB ELB

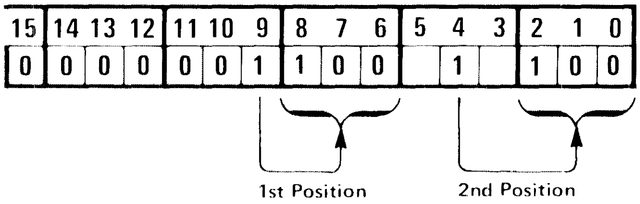
ALF ROTATE A LEFT FOUR



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	1	1		1		1	1	1

Rotates the A-register contents (all 16 bits) left four places. Bits 15, 14, 13, and 12 rotate around to bit positions 3, 2, 1, and 0, respectively. Equivalent to four successive RAL instructions.

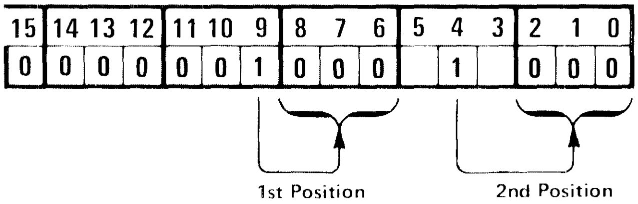
ALR A LEFT SHIFT, CLEAR SIGN



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	0	0		1		1	0	0

Shifts the A-register contents left one place and clears sign bit 15.

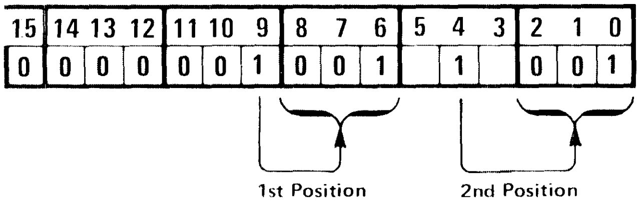
ALS A LEFT SHIFT



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	0		1		0	0	0

Arithmetically shifts the A-register contents left one place, 15 magnitude bits only; bit 15 (sign) is not affected. The bit shifted out of bit position 14 is lost; a logic 0 replaces vacated bit position 0.

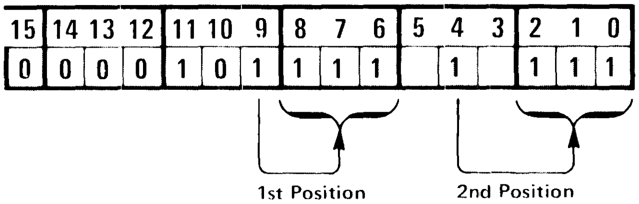
ARS A RIGHT SHIFT



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	0	1		1		0	0	1

Arithmetically shifts the A-register contents right one place, 15 magnitude bits only; bit 15 (sign) is not affected. A copy of the sign bit is shifted into bit position 14; the bit shifted out of bit position 0 is lost.

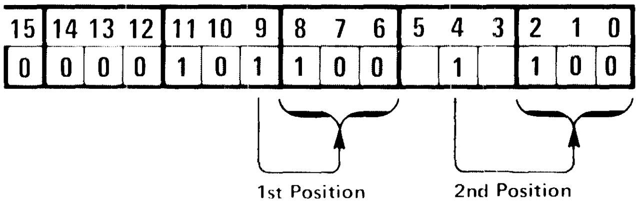
BLF ROTATE B LEFT FOUR



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	1	1		1		1	1	1

Rotates the B-register contents (all 16 bits) left four places. Bits 15, 14, 13, and 12 rotate around to bit positions 3, 2, 1, and 0, respectively. Equivalent to four successive RBL instructions.

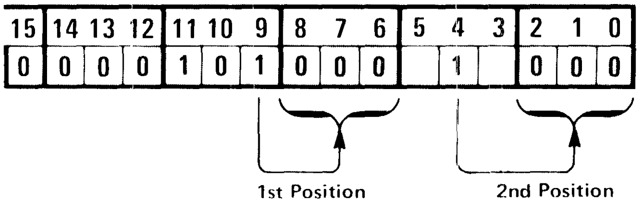
BLR B LEFT SHIFT, CLEAR SIGN



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	0	0		1		1	0	0

Shifts the B-register contents left one place and clears sign bit 15.

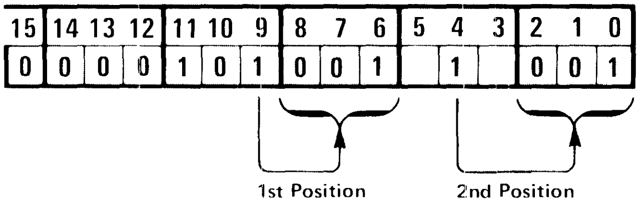
BLS B LEFT SHIFT



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	0		1		0	0	0

Arithmetically shifts the B-register contents left one place, 15 magnitude bits only; bit 15 (sign) is not affected. The bit shifted out of bit position 14 is lost; a logic 0 replaces vacated bit position 0.

BRS B RIGHT SHIFT



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	1		1		0	0	1

Arithmetically shifts the B-register contents right one place, 15 magnitude bits only; bit 15 (sign) is not affected. A copy of the sign bit is shifted into bit position 14; the bit shifted out of bit position 0 is lost.

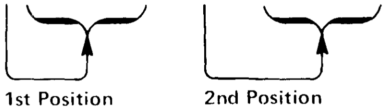
CLE CLEAR E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		0					1					

Clears the E-register; i.e., the extend bit becomes a logic 0.

ELA ROTATE E LEFT WITH A

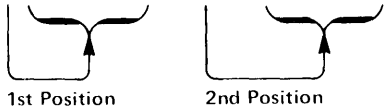
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	1	0		1		1	1	0



Rotates the E-register content left with the A-register contents (one place). The E-register content rotates into bit position 0; bit 15 rotates into the E-register.

ELB ROTATE E LEFT WITH B

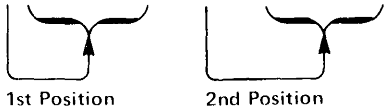
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	1	0		1		1	1	0



Rotates the E-register content left with the B-register contents (one place). The E-register content rotates into bit position 0; bit 15 rotates into the E-register.

ERA ROTATE E RIGHT WITH A

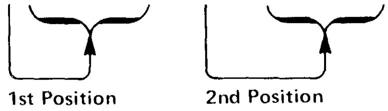
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1	0	1		1		1	0	1



Rotates the E-register content right with the A-register contents (one place). The E-register content rotates into bit position 15; bit 0 rotates into the E-register.

ERB ROTATE E RIGHT WITH B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	0	1		1		1	0	1



Rotates the E-register content right with the B-register contents (one place). The E-register content rotates into bit position 15; bit 0 rotates into the E-register.

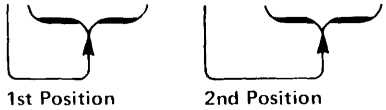
NOP NO OPERATION

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This all-zeros instruction causes a no-operation cycle.

RAL ROTATE A LEFT

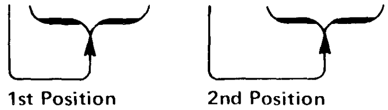
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	0		1		0	1	0



Rotates the A-register contents left one place (all 16 bits). Bit 15 rotates into bit position 0.

RAR ROTATE A RIGHT

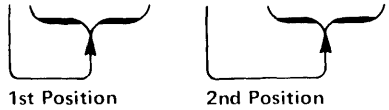
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	1		1		0	1	1



Rotates the A-register contents right one place (all 16 bits). Bit 0 rotates into bit position 15.

RBL ROTATE B LEFT

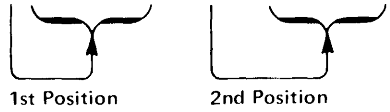
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	1	0		1		0	1	0



Rotates the B-register contents left one place (all 16 bits). Bit 15 rotates into bit position 0.

RBR ROTATE B RIGHT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	1	1		1		0	1	1



Rotates the B-register contents right one place (all 16 bits). Bit 0 rotates into bit position 15.

Programming Information 21MX

SLA SKIP IF LSB OF A IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0							1			

Skips the next instruction if the least-significant bit (bit 0) of the A-register is a logic 0.

SLB SKIP IF LSB OF B IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0							1			

Skips the next instruction if the least-significant bit (bit 0) of the B-register is a logic 0.

3-17. **ALTER/SKIP GROUP.** The 19 instructions comprising the alter/skip group (ASG) are defined next. This group is specified by setting bit 10 to a logic 1. Rules for combining microinstructions are as follows (refer to table 3-4):

a. Only one microinstruction can be chosen from each of the two multiple-choice columns.

b. References can be made to either the A-register or B-register, but not both.

c. Sequence of execution is from left to right.

d. If two or more skip functions are combined, the skip function will occur if either or both conditions are met. One exception exists: refer to the RSS instruction.

e. In machine code, use zeros to exclude unwanted microinstructions.

Table 3-4. Alter/Skip Group Combining Guide

$$\left[\begin{Bmatrix}\text{CLA}\\ \text{CMA}\\ \text{CCA}\end{Bmatrix}\right]\ \text{[,SEZ]}\ \left[,\begin{Bmatrix}\text{CLE}\\ \text{CME}\\ \text{CCE}\end{Bmatrix}\right]\ \text{[,SSA] [,SLA] [,INA] [,SZA] [,RSS]}$$

$$\left[\begin{Bmatrix}\text{CLB}\\ \text{CMB}\\ \text{CCB}\end{Bmatrix}\right]\ \text{[,SEZ]}\ \left[,\begin{Bmatrix}\text{CLE}\\ \text{CME}\\ \text{CCE}\end{Bmatrix}\right]\ \text{[,SSB] [,SLB] [,INB] [,SZB] [,RSS]}$$

CCA CLEAR AND COMPLEMENT A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1								

Clears and complements the A-register contents; i.e., the contents of the A-register become 177777 (octal). This is the two's complement form of -1.

CCB CLEAR AND COMPLEMENT B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1								

Clears and complements the B-register contents; i.e., the contents of the B-register become 177777 (octal). This is the two's complement form of -1.

CCE CLEAR AND COMPLEMENT E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		1			1	1						

Clears and complements the E-register content (extend bit); i.e., the extend bit becomes a logic 1.

CLA CLEAR A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1								

Clears the A-register; i.e., the contents of the A-register become 000000 (octal).

CLB CLEAR B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1								

Clears the B-register; i.e., the contents of the B-register become 000000 (octal).

CLE CLEAR E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		1			0	1						

Clears the E-register; i.e., the extend bit becomes a logic 0.

CMA COMPLEMENT A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	0								

Complements the A-register contents (one's complement).

CMB COMPLEMENT B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0								

Complements the B-register contents (one's complement).

CME COMPLEMENT E

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		1			1	0						

Complements the E-register content (extend bit).

INA INCREMENT A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1								1		

Increments the A-register by one. Can result in setting the extend bit or the overflow bit.

INB INCREMENT B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1								1		

Increments the B-register by one. Can result in setting the extend bit or the overflow bit.

RSS REVERSE SKIP SENSE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		1										1

Skip occurs for any of the following skip instructions, if present, when the non-zero condition is met. An RSS without a skip instruction in the word causes an unconditional skip. If a word with RSS also includes both SSA and SLA (or SSB and SLB), bits 15 and 0 must both be logic 1's for a skip to occur; in all other cases, a skip occurs if one or more skip conditions are met.

SEZ SKIP IF E IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0		1					1					

Skips the next instruction if the E-register content (extend bit) is a logic 0.

SLA SKIP IF LSB OF A IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1							1			

Skips the next instruction if the least-significant bit (bit 0) of the A-register is a logic 0; i.e., skips if an even number is in the A-register.

SLB SKIP IF LSB OF B IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1							1			

Skips the next instruction if the least-significant bit (bit 0) of the B-register is a logic 0; i.e., skips if an even number is in the B-register.

SSA SKIP IF SIGN OF A IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1						1				

Skips the next instruction if the sign bit (bit 15) of the A-register is a logic 0; i.e., skips if a positive number is in the A-register.

SSB SKIP IF SIGN OF B IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1						1				

Skips the next instruction if the sign bit (bit 15) of the B-register is a logic 0; i.e., skips if a positive number is in the B-register.

SZA SKIP IF A IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1									1	

Skips the next instruction if the A-register contents are zero (16 zeros).

SZB SKIP IF B IS ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1									1	

Skips the next instruction if the B-register contents are zero (16 zeros).

3-18. INPUT/OUTPUT INSTRUCTIONS

The following input/output instructions provide the capability of setting or clearing the I/O flag and control bits, testing the state of the overflow and the I/O flag bits, and transferring data between specific I/O devices and the A- and B-registers. In addition, specific instructions in this group control the vectored priority interrupt system and can cause a programmed halt.

Bit 11, where relevant, specifies the A- or B-register or distinguishes between set control and clear control; otherwise, bit 11 may be a logic 0 or a logic 1 without affecting the instruction (although the assembler will assign zeros in this case). In those instructions where bit position 9 includes the letters H/C, the programmer has the choice of holding (logic 0) or clearing (logic 1) the device flag after executing the instruction. (Exception: the H/C bit associated with instructions SOC and SOS holds or clears the overflow bit instead of the device flag.) Bits 8, 7, and 6 specify the appropriate I/O instruction and bits 5 through 0 form a two-digit octal select code (address) to apply the instruction to one of up to 64 input/output devices or functions.

CLC CLEAR CONTROL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	H/C	1	1	1						

Select Code

Clears the control bit of the selected I/O channel or function. This furns off the specific device channel and prevents it from interrupting. A CLC 00 instruction clears all control bits from select code 06 upward, effectively turning off all I/O devices.

CLF CLEAR FLAG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0		1	1	0	0	1						

Select Code

Clears the flag of the selected I/O channel or function. A CLF 00 instruction disables the interrupt system for all select codes except power fail (select code 04) and parity error (select code 05), which are always enabled; this does not affect the status of the individual channel flags.

CLO CLEAR OVERFLOW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	1

Clears the overflow bit.

HLT HALT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0		1	H/C	0	0	0						

Select Code

Halts the computer and holds or clears the flag of the selected I/O channel. The HLT instruction has the same effect as pressing the operator panel HALT pushbutton; i.e., the RUN indicator turns off and all operator panel control switches are enabled. The HLT instruction will be contained in the T-register, which is selected and displayed automatically when the computer halts. The P-register contents will normally contain the HLT location plus one (P + 1).

LIA LOAD INTO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	H/C	1	0	1						

Select Code

Loads the contents of the I/O buffer associated with the selected device into the A-register.

LIB LOAD INTO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	H/C	1	0	1						

Select Code

Loads the contents of the I/O buffer associated with the selected device into the B-register.

MIA MERGE INTO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	H/C	1	0	0						

Select Code

By executing a logical "inclusive or" function, merges the contents of the I/O buffer associated with the selected device into the A-register.

MIB MERGE INTO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	H/C	1	0	0						

Select Code

By executing a logical "inclusive or" function, merges the contents of the I/O buffer associated with the selected device into the B-register.

OTA OUTPUT A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	H/C	1	1	0						

Select Code

Outputs the contents of the A-register to the I/O buffer associated with the selected device. If the I/O buffer is less than 16 bits in length, the least-significant bits of the A-register are normally loaded. (Some exceptions to this exist, depending on the type of output device.) The contents of the A-register are not altered.

OTB OUTPUT B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	H/C	1	1	0						

Select Code

Outputs the contents of the B-register to the I/O buffer associated with the selected device. If the I/O buffer is less than 16 bits in length, the least-significant bits of the B-register are normally loaded. (Some exceptions to this exist, depending on the type of output device.) The contents of the B-register are not altered.

SFC SKIP IF FLAG CLEAR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0		1	0	0	1	0						

Select Code

Skips the next programmed instruction if the flag of the selected channel is clear (device busy).

SFS SKIP IF FLAG SET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0		1	0	0	1	1						

Select Code

Skips the next programmed instruction if the flag of the selected channel is set (device ready).

SOC SKIP IF OVERFLOW CLEAR

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	H/C	0	1	0	0	0	0	0	0	1

Skips the next programmed instruction if the overflow bit is clear. Use the H/C bit (bit 9) to either hold or clear the overflow bit following the completion of this instruction (whether the skip is taken or not).

SOS SKIP IF OVERFLOW SET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	H/C	0	1	1	0	0	0	0	0	1

Skips the next programmed instruction if the overflow bit is set. Use the H/C bit (bit 9) to either hold or clear the overflow bit following the completion of this instruction (whether the skip is taken or not).

STC SET CONTROL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	H/C	1	1	1						

Select Code

Sets the control bit of the selected I/O channel or function.

STF SET FLAG

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0		1	0	0	0	1						

Select Code

Sets the flag of the selected I/O channel or function. An STF 00 instruction enables the interrupt system for all select codes except power fail (select code 04) and parity error (select code 05), which are always enabled.

STO SET OVERFLOW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1

Sets the overflow bit.

3-19. EXTENDED ARITHMETIC MEMORY REFERENCE INSTRUCTIONS

The four extended arithmetic memory reference instructions provide for integer multiply and divide and for loading and storing double-length words to and from the A- and B-registers. The complete instruction requires two words: one for the instruction code and one for the address. When stored in memory, the instruction word is the first to be fetched; the address word is in the next sequential location.

Since 15 bits are available for the address, these instructions can directly address any location in memory. As for all memory reference instructions, indirect addressing to any number of levels may also be used. A logic 0 in bit position 15 specifies direct addressing; a logic 1 specifies indirect addressing.

DIV DIVIDE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
D/I															

Memory Address

Divides a double-word integer in the combined B- and A-registers by a 16-bit integer in the addressed memory location. The result is a 16-bit integer quotient in the A-register and a 16-bit integer remainder in the B-register. Overflow can result from an attempt to divide by zero, or from an attempt to divide by a number too small for the dividend. In the former case (divide by zero), the division will not be attempted and the B- and A-register contents will be unchanged except that a negative quantity will be made positive. In the latter case (divisor too small), the execution will be attempted with unpredictable results left in the B- and A-registers. If there is no divide error, the overflow bit is cleared.

DLD DOUBLE LOAD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
D/I															

Memory Address

Loads the contents of addressed memory location m (and m + 1) into the A- and B-registers, respectively.

DST DOUBLE STORE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
D/I															

Memory Address

Stores the double-word quantity in the A- and B-registers into addressed memory locations m (and m + 1), respectively.

MPY MULTIPLY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
D/I															

Memory Address

Multiplies a 16-bit integer in the A-register by a 16-bit integer in the addressed memory location. The resulting double-length integer product resides in the B- and A-registers, with the B-register containing the sign bit and the most-significant 15 bits of the quantity. The A-register may be used as an operand (i.e., memory address 0), resulting in an arithmetic square. Overflow cannot occur because the instruction clears the overflow bit.

3-20. EXTENDED ARITHMETIC REGISTER REFERENCE INSTRUCTIONS

The six extended arithmetic register reference instructions provide various types of shifting operations on the combined contents of the B- and A-registers. The B-register is considered to be to the left (most-significant word) and the A-register is considered to be to the right (least-significant word). An example of each type of shift operation is illustrated in figure 3-4.

The complete instruction is given in one word and includes four bits (unshaded) to specify the number of shifts (1 to 16). By viewing these four bits as a binary-coded number, the number of shifts is easily expressed; i.e., binary-coded 1 = 1 shift, binary-coded 2 = 2 shifts . . . binary-coded 15 = 15 shifts. The maximum number of 16 shifts is coded with four zeros, which essentially exchanges the contents of the B- and A-registers.

The extend bit is not affected by any of the following instructions. Except for the arithmetic shifts, overflow also is not affected.

ASL ARITHMETIC SHIFT LEFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	1				

Number of Shifts

Arithmetically shifts the combined contents of the B- and A-registers left n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated low-order positions of the A-register. The sign bit is not affected, and data bits are lost out of bit position 14 of the B-register. If any one of the lost bits is a significant data bit ("1" for positive numbers, "0" for negative numbers), the overflow bit will be set; otherwise, overflow will be cleared during execution. See ASL example in figure 3-4. Note that two additional shifts in this example would cause an error by losing a significant '1'.

ASR ARITHMETIC SHIFT RIGHT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	0	0	0	1				

Number of Shifts

Arithmetically shifts the combined contents of the B- and A-registers right n places. The value of n may be any number from 1 through 16. The sign bit is unchanged and

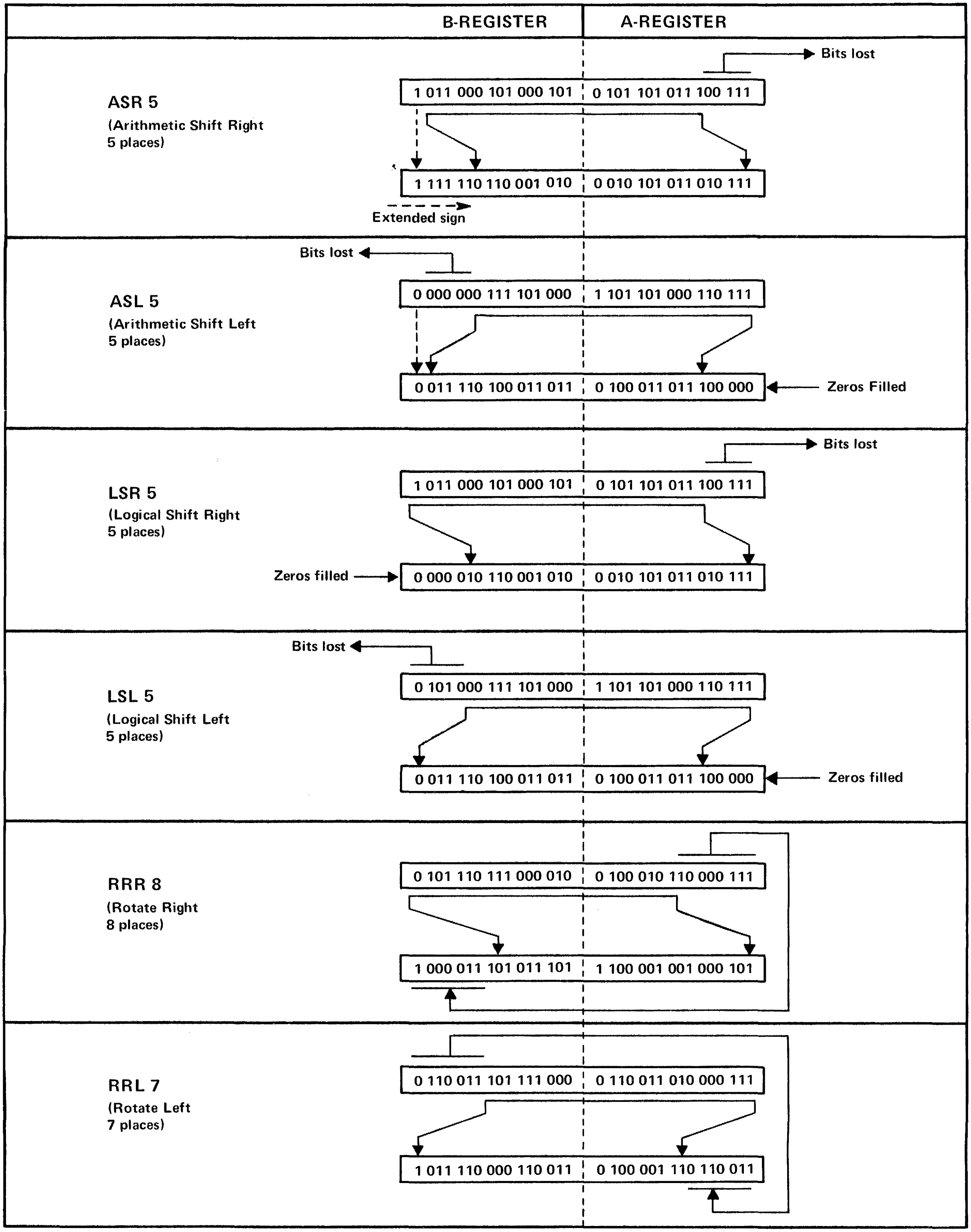


Figure 3-4. Examples of Double-Word Shifts and Rotates

is extended into bit positions vacated by the right shift. Data bits shifted out of the least-significant end of the A-register are lost. Overflow cannot occur because the instruction clears the overflow bit.

LSL LOGICAL SHIFT LEFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	1	0				

Number of Shifts

Logically shifts the combined contents of the B- and A-registers left n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated low-order bit positions of the A-register; data bits are lost out of the high-order bit positions of the B-register.

LSR LOGICAL SHIFT RIGHT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	0	0	1	0				

Number of Shifts

Logically shifts the combined contents of the B- and A-registers right n places. The value of n may be any number from 1 through 16. Zeros are filled into vacated high-order bit positions of the B-register; data bits are lost out of the low-order bit positions of the A-register.

RRL ROTATE LEFT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	1	0	0				

Number of Shifts

Rotates the combined contents of the B- and A-registers left n places. The value of n may be any number from 1 through 16. No bits are lost or filled in. Data bits shifted out of the high-order end of the B-register are rotated around to enter the low-order end of the A-register.

RRR ROTATE RIGHT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	0	1	0	0				

Number of Shifts

Rotates the combined contents of the B- and A-registers right n places. The value of n may be any number from 1 through 16. No bits are lost or filled in. Data bits shifted out of the low-order end of the A-register are rotated around to enter the high-order end of the B-register.

3-21. EXTENDED INSTRUCTION GROUP CODING

The extended instruction group includes index register instructions, bit and byte manipulation instructions, and move and compare instructions. The index registers are two hardware registers which are not accessible by the base set instructions described previously.

Instructions comprising the extended instruction group are one, two, or three words in length. The first word is always the instruction code; operand addresses are given in the words following the instruction code or in the A- and B-registers. The operand addresses are 15 bits long, with bit 15 (most-significant bit) indicating direct or indirect addressing. Bit 15 must be a zero for the JPY address since indirect addressing is not allowed.

Operand addresses given in the A- and B-registers are 15 bits long for word addresses and 16 bits long for byte addresses. No indirect addressing is allowed. Bit 15 is ignored for word addresses.

3-22. INDEX REGISTER INSTRUCTIONS

The index registers are two of the 14 scratch pad registers normally accessible only by microprograms. The following instructions make these two 16-bit registers (X and Y) directly accessible by the software.

ADX ADD MEMORY TO X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	1	1	0
D/I															

Memory Address

Adds the contents of the addressed memory location to the contents of the X-register. The sum remains in the X-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit.

ADY ADD MEMORY TO Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	1	1	0
D/I															

Memory Address

Adds the contents of the addressed memory location to the contents of the Y-register. The sum remains in the Y-register and the contents of the memory cell are unaltered. The result of this addition may set the extend bit or the overflow bit.

CAX COPY A TO X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	1

Copies the contents of the A-register into the X-register. The contents of the A-register are unaltered.

CAY COPY A TO Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	1	0	0	1

Copies the contents of the A-register into the Y-register. The contents of the A-register are unaltered.

CBX COPY B TO X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	0	0	1

Copies the contents of the B-register into the X-register. The contents of the B-register are unaltered.

CBY COPY B TO Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	0	0	1

Copies the contents of the B-register into the Y-register. The contents of the B-register are unaltered.

CXA COPY X TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	0	1	0	0

Copies the contents of the X-register into the A-register. The contents of the X-register are unaltered.

CXB COPY X TO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	1	0	0

Copies the contents of the X-register into the B-register. The contents of the X-register are unaltered.

CYA COPY Y TO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0

Copies the contents of the Y-register into the A-register. The contents of the Y-register are unaltered.

CYB COPY Y TO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	1	0	0

Copies the contents of the Y-register into the B-register. The contents of the Y-register are unaltered.

DSX DECREMENT X AND SKIP IF ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	0	0	1

Subtracts one from the contents of the X-register. If the result of this operation is zero (X-register decremented from 000001 to 000000), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

DSY DECREMENT Y AND SKIP IF ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	0	0	1

Subtracts one from the contents of the Y-register. If the result of this operation is zero (Y-register decremented from 000001 to 000000), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

ISX INCREMENT X AND SKIP IF ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0

Adds one to the contents of the X-register. If the result of this operation is zero (X-register rolls over to 000000 from 177777), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

ISY INCREMENT Y AND SKIP IF ZERO

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	0	0	0

Adds one to the contents of the Y-register. If the result of this operation is zero (Y-register rolls over to 000000 from 177777), the next instruction is skipped; i.e., the P-register count is advanced two counts instead of one count. If the result is not zero, the next sequential instruction is executed.

LAX LOAD A INDEXED BY X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	0	0	1	0
D/I															

Operand Address

Loads the A-register with the contents indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the X-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LAY LOAD A INDEXED BY Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	1	0	1	0
D/I															

Operand Address

Loads the A-register with the contents indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the Y-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LBX LOAD B INDEXED BY X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	0	1	0
D/I															

Operand Address

Loads the B-register with the contents indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the X-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LBY LOAD B INDEXED BY Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	0	1	0
D/I															

Operand Address

Loads the B-register with the contents indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the X-register and memory contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

LDX LOAD X FROM MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	1	0	1
D/I															

Memory Address

Loads the contents of the addressed memory location into the X-register. The A- and B-registers may be addressed as locations 00000 and 00001, respectively; however, if it is desired to load from the A- or B-register, copy instructions CAX or CBX should be used since they are more efficient.

LDY LOAD Y FROM MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	1	0	1
D/I															

Memory Address

Loads the contents of the addressed memory location into the Y-register. The A- and B-registers may be addressed as locations 00000 and 00001, respectively; however, if it is desired to load from the A- or B-register, copy instructions CAY or CBY should be used since they are more efficient.

SAX STORE A INDEXED BY X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
D/I															

Operand Address

Stores the contents of the A-register into the location indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the A- and X-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

SAY STORE A INDEXED BY Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0
D/I															

Operand Address

Stores the contents of the A-register into the location indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the A- and Y-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

SBX STORE B INDEXED BY X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0
D/I															

Operand Address

Stores the contents of the B-register into the location indicated by the effective address, which is computed by adding the contents of the X-register to the operand address. The effective address is loaded into the M-register; the B- and X-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

SBY STORE B INDEXED BY Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	0	0	0
D/I															

Operand Address

Stores the contents of the B-register into the location indicated by the effective address, which is computed by adding the contents of the Y-register to the operand address. The effective address is loaded into the M-register; the B- and Y-register contents are not altered. Indirect addressing is resolved before indexing; bit 15 of the effective address is ignored.

STX STORE X TO MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	0	1	1
D/I															

Memory Address

Stores the contents of the X-register into the addressed memory location. The A- and B-registers may be addressed as locations 00000 and 00001, respectively. The X-register contents are not altered.

STY STORE Y TO MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1
D/I															

Memory Address

Stores the contents of the Y-register into the addressed memory location. The A- and B-registers may be addressed as locations 00000 and 00001, respectively. The Y-register contents are not altered.

XAX EXCHANGE A AND X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	0	1	1	1

Exchanges the contents of the A- and X-registers.

XAY EXCHANGE A AND Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1

Exchanges the contents of the A- and Y-registers.

XBX EXCHANGE B AND X

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	0	1	1	1

Exchanges the contents of the B- and X-registers.

XBY EXCHANGE B AND Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	0	1	1	1	1

Exchanges the contents of the B- and Y-registers.

3-23. JUMP INSTRUCTIONS

The following two jump instructions involving the Y-register allow a program to either jump to or exit from a subroutine.

JLY JUMP AND LOAD Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	0	1	0
D/I															

Memory Address

This instruction is designed for entering a subroutine. The instruction, executed in location P, causes computer control to jump unconditionally to the memory location specified in the memory address. Indirect addressing may be specified. The contents of the P-register plus two (return address) is loaded into the Y-register. A return to the main program sequence at P + 2 may be effected by a JPY instruction (described next). A memory protect check is performed by this instruction. The effective address may not be below the fence, including the addressable A- and B-registers.

JPY JUMP INDEXED BY Y

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	0	1	0
0															

Operand Address

Transfers control to the effective address, which is computed by adding the contents of the Y-register to the operand address. Indirect addressing is not allowed. The effective address is loaded into the P-register; the Y-register contents are not altered. A memory protect check is performed by this instruction. The effective address may not be below the fence, including the addressable A- and B-registers.

3-24. BYTE MANIPULATION INSTRUCTIONS

A byte address is defined as two times the word address plus zero or one, depending on whether the byte is in the high-order position (bits 8 through 15) or low-order position (bits 0 through 7) of the word containing it. If the byte of interest is in bit positions 8 through 15 of memory location 100, for example, then the address of that byte is 2 * 100 + 0, or 200; the address of the low-order byte in the same location is 201 (2 * 100 + 1). Because of the way byte addresses are defined, 16 bits are required to cover all possible byte addresses in a 32K-word memory configuration. Hence, for byte addressing, bit 15 does not indicate indirect addressing.

Byte addresses 000 through 003 reference bytes in the A- and B-registers. These addresses will not cause memory violations. The user should, however, be careful in referencing these byte addresses; for example, storing into byte address 002 or 003 would destroy the byte address originally contained in the B-register.

CBT COMPARE BYTES

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	0
D/I															

Memory Address

Compares the bytes in string 1 with those in string 2. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the string count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first byte address of string 1 and the B-register contains the first byte address of string 2.

The number of bytes to be compared is given by a 16-bit positive non-zero integer addressed by Word 2 of the instruction. The strings are compared one byte at a time; the ith byte in string 1 is compared with the ith byte in string 2. The comparison is performed arithmetically; i.e., each byte is treated as a positive number. If all bytes in string 1 are identical with all bytes in string 2, the "equal" exit is taken. As soon as two bytes are compared and found to be different, the "less than" or "greater than" exit is taken, depending on whether the byte in string 1 is less than or greater than the byte in string 2. The three ways this instruction exits are as follows:

a. No skip if string 1 is equal to string 2; the P-register advances one count from Word 3 of the instruction. The A-register contains its original value incremented by the count stored in the address specified in Word 2.

b. Skips one word if string 1 is less than string 2; the P-register advances two counts from Word 3 of the instruction. The A-register contains the address of the byte in string 1 where the comparison stopped.

c. Skips two words if string 1 is greater than string 2; the P-register advances three counts from Word 3 of the instruction. The A-register contains the address of the byte in string 1 where the comparison stopped.

For all three exits, the B-register will contain its original value incremented by the count stored in the address specified in Word 2. This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

LBT LOAD BYTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	0	1	1

This one word instruction loads into the A-register the byte whose address is contained in the B-register. The byte is right-justified with leading zeros in the left byte. The B-register is incremented by one.

MBT MOVE BYTES

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	1	0	1
D/I															

Memory Address

Moves bytes in a left-to-right manner; i.e., the byte having the lowest address from the source is moved first. This is a three word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the byte count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first byte address source and the B-register contains the first byte address destination.

The number of bytes to be moved is given by a 16-bit positive non-zero integer addressed by Word 2 of the instruction. The byte address in the A- and B-registers are incremented as each byte is being moved. Thus, at the end of the operation, the A- and B-registers are incremented by the number of bytes moved. Wraparound of the byte address would result from a carry out of bit position 15; therefore, if the destination became 000, 001, 002, or 003, the next byte would be moved into the A- or B-register and destroy the proper byte addresses for the move operation. For each byte move, a memory protect check is performed.

This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

SBT STORE BYTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	1	0	0

Stores the A-register low-order (right) byte in the byte address contained in the B-register. The B-register is incremented by one. A memory protect check is performed before the byte is stored. The left byte in the A-register does not have to be zeros. The other byte in the same word of the stored byte is not altered.

SFB SCAN FOR BYTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	1

This is a one word instruction with the operands in the A- and B-registers. The A-register contains a termination byte (high-order byte) and a test byte (low-order byte). The B-register contains the first byte address of the string to be scanned.

A string of bytes is scanned starting at the byte address given in the B-register. Scanning terminates when a byte in the string matches either the test byte or the termination byte in the A-register. The manner in which the instruction exits depends on which byte is matched first. If a byte in the string matches the test byte, the instruction will not skip upon exit; the B-register will contain the address of the byte matching the test byte. If a byte in the string matches the termination byte, the instruction will skip one word upon exit; the B-register will contain the address of the byte matching the termination byte *plus one*.

The scanning operation will not continue indefinitely even if neither the termination byte nor test byte exists in memory. These bytes are in the A-register with byte addresses 000 and 001, respectively. Thus, if no match is made by the time the B-register points to the last byte in memory, the B-register will roll over to zero and the next test will match the termination byte in the A-register with itself.

This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers.

3-25. BIT MANIPULATION INSTRUCTIONS

The following three instructions allow any number of bits in a specified memory location to be cleared, set, or tested.

CBS CLEAR BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0
D/I															
D/I															

Memory Address

Clears bits in the addressed location. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of a 16-bit mask, and

Word 3 = Address of word where bits are to be cleared.

The bits to be cleared correspond to logic 1's in the mask. The bits corresponding to logic 0's in the mask are not affected. A memory protect check is performed prior to modifying the word in memory.

SBS SET BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	0	1	1
D/I															
D/I															

Memory Address

Sets bits in the addressed location. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of a 16-bit mask, and

Word 3 = Address of word where bits are to be set.

The bits to be set correspond to logic 1's in the mask. The bits corresponding to logic 0's in the mask are not affected. A memory protect check is performed prior to modifying the word in memory.

TBS TEST BITS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	1	0	1
D/I															
D/I															

Memory Address

Tests (compares) bits in the addressed location. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of a 16-bit mask, and

Word 3 = Address of word in which bits are to be tested.

The bits in the addressed memory word corresponding to logic 1's in the mask are tested. If all the bits tested are 1's, the instruction will not skip; otherwise the instruction will skip one word (i.e., the P-register will advance two counts from Word 3 of the instruction).

3-26. WORD MANIPULATION INSTRUCTIONS

The following instructions facilitate the comparing and moving of word arrays.

CMW COMPARE WORDS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	0
D/I															

Memory Address

Compares the words in array 1 with those in array 2. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the word count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first word address of array 1 and the B-register contains the first word address of array 2. Bit 15 of the addresses in the A- and B-registers are ignored; i.e., no indirect addressing allowed.

The number of words to be compared is given by a 16-bit positive non-zero integer addressed by Word 2 of the instruction. The arrays are compared one word at a time; the ith word in array 1 is compared with the ith word in array 2. This comparison is performed arithmetically; i.e., each word is considered a two's complement number. If all words in array 1 are equal to all words in array 2, the "equal" exit is taken. As soon as two words are compared and found to be different, the "less than" or "greater than" exit is taken, depending on whether the word in array 1 is less than or greater than the word in array 2. The three ways this instruction exits are as follows:

a. No skip if array 1 is equal to array 2; the P-register advances one count from Word 3 of the instruction. The A-register contains its original value incremented by the word count stored in the address specified in Word 2.

b. Skips one word if array 1 is less than array 2; the P-register advances two counts from Word 3 of the instruction. The A-register contains the address of the word in array 1 where the comparison stopped.

c. Skips two words if array 1 is greater than array 2; the P-register advances three counts from Word 3 of the instruction. The A-register contains the address of the word in array 1 where the comparison stopped.

For all three exits, the B-register will contain its original value incremented by the word count stored in the address specified in Word 2. This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

MVW MOVE WORDS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1
D/I															

Memory Address

Moves words in a left-to-right manner; i.e., the word having the lowest address in the source is moved first. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Address of word containing the count, and

Word 3 = All-zeros word reserved for use by microcode.

The operand addresses are in the A- and B-registers. The A-register contains the first word address source and the B-register contains the first word address destination. The number of words to be moved is a 16-bit positive non-zero integer addressed by Word 2 of the instruction. The word addresses in the A- and B-registers are incremented as each word is being moved. Thus, at the end of the operation, the A- and B-registers are incremented by the number of words moved.

Wraparound of the word address would result from a carry into bit position 15 (i.e., at 32767). If the destination address became 000 or 001, the next word would be moved into the A- or B-register and destroy the proper word addresses for the move operation. For each word move, a memory protect check is performed.

This instruction is interruptible. The interrupt routine is expected to save and restore the contents of the A- and B-registers. During the interrupt, the remaining count is stored in Word 3 of the instruction.

FLOATING POINT INSTRUCTION [illegible]

The following six floating point instructions make it possible to add, subtract, multiply, and divide floating point numbers and to convert quantities from floating point format to integer format or vice versa.

Each of the four arithmetic instructions requires two words of memory: one for the instruction code and one for the operand address. Since a full 15 bits are available for the operand address, these instructions can directly address any location in memory. As with all memory reference instructions, indirect addressing to any number of levels is permitted. A logic 0 in bit position 15 specifies direct addressing; a logic 1 specifies indirect addressing.

The execution times of the floating point instructions are specified under paragraph 3-28. These instructions are noninterruptible; any attempted interrupt is held off for the full execution time of the currently active floating point instruction. However, data transfer via the dual-channel port controller (DCPC) is not held off.

FAD FLOATING POINT ADD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
D/I															

Memory Address

Adds the floating point quantity in the A- and B-registers to the floating point quantity in the specified memory locations. The floating point result is returned in the A- and B-registers. Overflow occurs if the result lies outside the range -2^{-127} through $(1-2^{-23}) * 2^{127}$. In such a case, the overflow flag is set and the result $(1-2^{-23}) * 2^{-129}$ is returned to the A- and B-registers. Underflow occurs if the result lies within the range $(1+2^{-22}) * -2^{129}$ through 2^{-129}. In such a case, the overflow flag is set and the result 0 is returned to the A- and B-registers.

FDV FLOATING POINT DIVIDE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	0	1	1	0	0	0	0
D/I															

Memory Address

Divides the floating point quantity in the A- and B-registers by the floating point quantity in the specified memory locations. The floating point quantity is returned to the A- and B-registers. Overflow and underflow are as described for the FAD instruction.

FIX FLOATING POINT TO INTEGER

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0

Converts the floatint point quantity in the A- and B-registers to integer format. The integer result is returned to the A-register. If the magnitude of the floating point number is <1, regardless of sign, the integer 0 is returned. If the magnitude of the exponent of the floating point number is $\geqslant 2^{16}$, regardless of sign, the integer 32767 (077777 octal) is returned and the overflow flag is set.

FLT INTEGER TO FLOATING POINT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0

Converts the integer quantity in the A-register to floating point format. The floating point result is returned to the A- and B-registers.

FMP FLOATING POINT MULTIPLY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0
D/I															

Memory Address

Multiplies the floating point quantity in the A- and B-registers by the floating point quantity in the specified memory locations. The floating point result is returned to the A- and B-registers. Overflow and underflow are as described for the FAD instruction.

FSB FLOATING POINT SUBTRACT

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0
D/I															

Memory Address

Subtracts the floating point quantity in the specified memory locations from the floating point quantity in the A- and B-registers. The floating point result is returned to the A- and B-registers. Overflow and underflow are as described for the FAD instruction.

3-28. DYNAMIC MAPPING SYSTEM

The basic addressing space of the HP 21MX Computer Series is 32,768 words, which is referred to as *logical* memory. The amount of MOS memory actually installed in the computer system is referred to as *physical* memory. An HP 21MX Computer with the optional Dynamic Mapping System (DMS) has an addressing capability for one million words of physical memory. The DMS allows logical memory to be mapped into physical memory through the use of four dynamically alterable memory maps.

3-29. MEMORY ADDRESSING

The basic memory addressing scheme provides for addressing 32 pages of logical memory, each of which consists of 1,024 words. This memory is addressed through a 15-bit memory address bus shown in figure 3-5. The upper 5 bits of this bus provide the page address and the lower 10 bits provide the relative word address within the page.

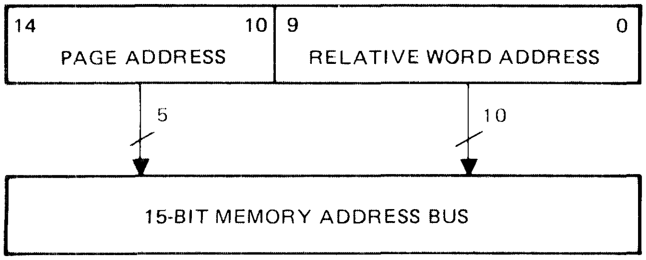


Figure 3-5. Basic Memory Addressing Scheme

The Memory Expansion Module (MEM), which is part of the DMS option, converts the 5-bit page address into a 10-bit page address and thereby allows 1,024 (2^{10}) pages to be addressed. This conversion is accomplished by allowing the original 5-bit address to identify one of the 32 registers within a "memory map." Each of these map registers contains the new user-specified 10-bit page address. This new page address is combined with the original 10-bit relative address to form a 20-bit memory address bus as shown in figure 3-6.

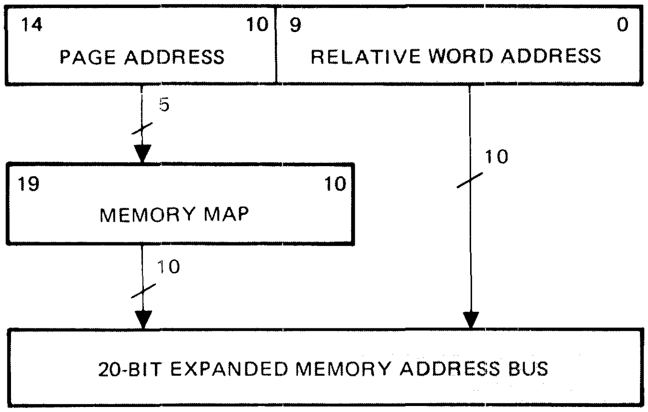


Figure 3-6. Expanded Memory Addressing Scheme

3-30. MAP REGISTER LOADING

Conversion of the basic 16-bit word data format to and from the map register 12-bit word data format is shown in figure 3-7. Bits 13 through 10 of the basic data format are not used by the memory map registers. Read and write memory protect violations are discussed in paragraph 3-31.

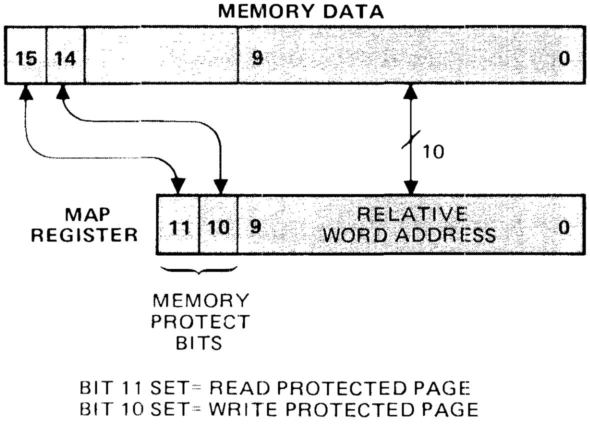


Figure 3-7. Basic Word Format Vs Map Register Format

3-31. STATUS AND VIOLATION REGISTERS

The MEM also includes a status register and a violation register. As shown in table 3-5, the MEM status register contents enable the programmer to determine whether the MEM was enabled or disabled at the time of the last interrupt and the address of the base page fence. The MEM violation register contents enable the programmer to determine whether a fault occurred in the hardware or the software so that the proper corrective steps may be taken. Refer to table 3-6.

Table 3-5. **MEM Status Register Format**

BIT	SIGNIFICANCE
15	0 = MEM disabled at last interrupt 1 = MEM enabled at last interrupt
14	0 = System map selected at last interrupt 1 = User map selected at last interrupt
13	0 = MEM disabled currently 1 = MEM enabled currently
12	0 = System map selected currently 1 = User map selected currently
11	0 = Protected mode disabled currently 1 = Protected mode enabled currently
10	Portion mapped*
9	Base page fence bit 9
8	Base page fence bit 8
7	Base page fence bit 7
6	Base page fence bit 6
5	Base page fence bit 5
4	Base page fence bit 4
3	Base page fence bit 3
2	Base page fence bit 2
1	Base page fence bit 1
0	Base page fence bit 0

*Bit 10	Mapped Address (M)
0	Fence $\leqslant$ M $<$ 2000_8
1	1 $<$ M $<$ Fence

Note: The base page fence separates the reserved (mapped) memory from the shared (unmapped) memory. Bit 10 specifies which area is reserved (mapped). (Refer to LFA and LFB instructions contained in paragraph 3-34.)

Table 3-6. **MEM Violation Register Format**

BIT	SIGNIFICANCE
15	Read violation*
14	Write violation*
13	Base page violation*
12	Privileged instruction violation*
11	Reserved
10	Reserved
9	Reserved
8	Reserved
7	0 = ME bus disabled at violation 1 = ME bus enabled at violation
6	0 = MEM disabled at violation 1 = MEM enabled at violation
5	0 = System map enabled at violation 1 = User map enabled at violation
4	Map address bit 4
3	Map address bit 3
2	Map address bit 2
1	Map address bit 1
0	Map address bit 0

*Significant when associated bit is set.

Any attempt to read from a read-protected page will result in a read violation and the memory read will not occur. Any attempt to write into a write-protected page will result in a write violation and the memory will not be altered. In addition, if a page is write protected, a jump or jump indirect instruction to that page will cause a write violation and the jump will not occur. It should be noted that all violation rules are ignored for DCPC signals.

If a read or write violation occurs, the MEM signals the memory protect logic that a violation has occurred which causes the memory protect logic to generate an interrupt. As discussed in paragraph 3-45, memory violations are interrupted to select code 05 and a DMS violation can be distinguished from a memory protect violation by executing an SFS 05 instruction. If the skip occurs, DMS is in violation; if no skip occurs, memory protect is in violation.

3-32. MAP SEGMENTATION

All registers within the memory map are dynamically alterable. To maximize the system performance capability, the MEM includes four separate memory maps: the

User Map, System Map, and two Dual-Channel Port Controller (DCPC) Maps. (See figure 3-8.) These maps, which are manipulated through the use of 38 machine-language instructions, are addressed as a contiguous register block. It should be noted that the base page fence applies to both the System Map and the User Map.

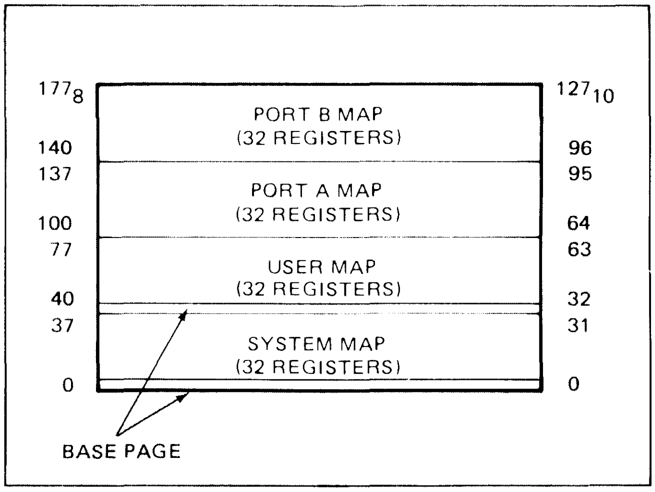


Figure 3-8. Map Segmentation

3-33. POWER FAIL CHARACTERISTICS

A power failure automatically enables the System Map, and a minimum of 500 microseconds is assured the programmer for executing a power fail routine. Since all maps are disabled and none are considered valid upon the restoration of power, the power fail routine should include instructions to save as many maps as desired.

3-34. DMS INSTRUCTION CODING

Machine language coding and definitions of the 38 Dynamic Mapping System instructions are provided on this and following pages. A sample map load and enable routine is given in paragraph 3-35.

DJP DISABLE MEM AND JUMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	0	1	0
D/I															

Memory Address

Disables the translation and protection features of the MEM hardware. Prior to disabling, the P-register is set to the effective memory address. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

DJS DISABLE MEM AND JUMP TO SUBROUTINE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	0	1	1
D/I															

Memory Address

Disables the translation and protection features of the MEM hardware. Prior to disabling, the P-register is set one count past the effective memory address (m + 1) and the return address is stored in location m. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

JRS JUMP AND RESTORE STATUS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	1
D/I															
D/I															

Memory Address

Causes the status of the MEM to be restored. This is a three-word instruction where

Word 1 = Instruction code,

Word 2 = Status word address, and

Word 3 = Jump address.

Only bits 15 and 14 of the status word are used; the remaining bits (13-0) of the status word are ignored. Bits 15 and 14 restore the MEM status as follows:

Bit 15 = 0 = MEM will be disabled
= 1 = MEM will be enabled

Bit 14 = 0 = System map will be selected
= 1 = User map will be selected

As a result of executing this instruction, normal I/O interrupt are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the system map is enabled, the instruction is allowed.

LFA LOAD FENCE FROM A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	0	1	1	1

Loads the contents of the A-register into the base page fence register. Bits 9-0 of the A-register specify the address in page zero where shared (unmapped) memory is separated from reserved (mapped) memory. Bit 10 is used as follows to specify which portion is mapped:

Bit 10	Mapped Address (M)
0	Fence $\leqslant M < 2000_8$
1	$1 < M <$ Fence

This instruction will normally generate an MEM violation when executed in the protected mode; however, it is allowed if the system map is enabled. When an MEM violation does occur, the fence is not altered.

LFB LOAD FENCE FROM B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	1	1	1

Loads the contents of the B-register into the base page fence register. Bits 9-0 of the B-register specify the address in page zero where shared (unmapped) memory is separated from reserved (mapped) memory. Bit 10 is used as follows to specify which portion is mapped:

Bit 10	Mapped Address (M)
0	Fence $\leqslant M < 2000_8$
1	$1 < M <$ Fence

This instruction will normally generate an MEM violation when executed in the protected mode; however, it is allowed if the system map is enabled. When an MEM violation does occur, the fence is not altered.

MBF MOVE BYTES FROM ALTERNATE MAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	1

Moves a string of bytes using the alternate program map for source reads and the currently enabled map for destination writes. The A-register contains the source byte address and the B-register contains the destination byte address. The initial byte addresses in the A- and B-registers must be even byte addresses. The byte in bits 15 through 8 of a word is the even byte. The X-register contains the octal number of bytes to be moved. The number of bytes to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred. Both the source and destination must begin on word boundaries.

The instruction is interruptible on an even number of byte transfers, thus maintaining the even word boundaries in the A- and B-registers. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the byte string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of bytes moved.

This instruction can cause an MEM violation only if read or write protection rules are violated.

MBI MOVE BYTES INTO ALTERNATE MAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	0

Moves a string of bytes using the currently enabled map for source reads and the alternate program map for destination writes. The A-register contains the source byte address and the B-register contains the destination byte address. The initial byte addresses in the A- and B-registers must be even byte addresses. The byte in bits 15 through 8 of a word is the even byte. The X-register contains the octal number of bytes to be moved. The number of bytes to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred. Both the source and destination must begin on word boundaries.

The instruction is interruptible on an even number of byte transfers, thus maintaining the even word boundaries in the A- and B-registers. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the byte string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of bytes moved.

This instruction will always cause an MEM violation when executed in the protected mode and no bytes will be transferred.

MBW MOVE BYTES WITHIN ALTERNATE MAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	0	1	0	0

Moves a string of bytes with both the source and destination addresses established through the alternate program map. The A-register contains the source byte address and the B-register contains the destination byte address. The initial byte addresses in the A- and B-registers must be even byte addresses. The byte in bits 15 through 8 of a word is the even byte. The X-register contains the octal number of bytes to be moved. The number of bytes to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred. Both the source and destination must begin on word boundaries.

The instruction is interruptible on an even number of byte transfers, thus maintaining the even word boundaries in the A- and B-registers. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the byte string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of bytes moved.

This instruction will always cause an MEM violation when executed in the protected mode and no bytes will be transferred.

MWF MOVE WORDS FROM ALTERNATE MAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	0	1	1	0

Moves a string of words using the alternate program map for source reads and the currently enabled map for destination writes. The A-register contains the source address and the B-register contains the destination address. The X-register contains the octal number of words to be moved. The number of words to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred.

The instruction is interruptible. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the word string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of words moved.

This instruction can cause an MEM violation only if read and write protection rules are violated.

MWI MOVE WORDS INTO ALTERNATE MAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	0	1	0	1

Moves a string of words with both the source and destination addresses established through the alternate program map. The A-register contains the source address and the B-register contains the destination address. The X-register contains the octal number of words to be moved. The number of words to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred.

The instruction is interruptible. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the word string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of words moved.

This instruction will always cause an MEM violation when executed in the protected mode and no words will be transferred.

MWW MOVE WORDS WITHIN ALTERNATE MAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	0	1	1	1

Moves a string of words using the currently enabled map for source reads and the alternate program map for destination writes. The A-register contains the source address and the B-register contains the destination address. The X-register contains the octal number of words to be moved. The number of words to be moved is restricted to a positive integer greater than zero. If the contents of the X-register is zero, the instruction will be a NOP. If the contents of the X-register is a negative integer, a large indeterminate block of memory will be transferred.

The instruction is interruptible. The interrupt routine is expected to save and restore the current contents of the A-, B-, and X-registers to allow continuation of the instruction at the next entry. When the word string move is completed, the X-register will always be zero and the A- and B-registers will contain their original value incremented by the number of words moved.

This instruction will always cause an MEM violation when executed in the protected mode and no words will be transferred.

PAA LOAD/STORE PORT A MAP PER A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	0	1	0	1	0

Transfers the 32 Port A map registers to or from memory. If bit 15 of the A-register is clear, the Port A map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the Port A map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port A map is allowed within the constraints of write protected memory.

PAB LOAD/STORE PORT A MAP PER B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	1	0	1	0

Transfers the 32 Port A map registers to or from memory. If bit 15 of the B-register is clear, the Port A map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the Port A map is *stored* into memory starting at the address specified in bits 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port A map is allowed within the constraints of write protected memory.

PBA LOAD/STORE PORT B MAP PER A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	0	1	0	1	1

Transfers the 32 Port B map registers to or from memory. If bit 15 of the A-register is clear, the Port B map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the Port B map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port B map is allowed within the constraints of write protected memory.

PBB LOAD/STORE PORT B MAP PER B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	1	0	1	1

Transfers the 32 Port B map registers to or from memory. If bit 15 of the B-register is clear, the Port B map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the Port B map is *stored* into memory starting at the address specified in bit 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

An attempt to load any map register when in the protected mode will cause an MEM violation. An attempt to store the Port B map is allowed within the constraints of the write protected memory.

RSA READ STATUS REGISTER INTO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0

Reads the contents of the MEM status register into the A-register. This instruction can be executed at any time. The format of the MEM status register is given in table 3-5.

RSB READ STATUS REGISTER INTO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	0	0	0

Reads the contents of the MEM status register into the B-register. This instruction can be executed at any time. The format of the MEM status register is given in table 3-5.

RVA READ VIOLATION REGISTER INTO A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1

Reads the contents of the MEM violation register into the A-register. This instruction can be executed at any time. The format of the MEM violation register is given in table 3-6.

RVB READ VIOLATION REGISTER INTO B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	0	0	1

Reads the contents of the MEM violation register into the B-register. This instruction can be executed at any time. The format of the MEM violation register is given in table 3-6.

SJP ENABLE SYSTEM MAP AND JUMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	0
D/I															

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the System map, for translating all programmed memory references. Prior to enabling the System map, the P-register is set to the effective memory address. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed and effectively executes a JMP *+1,I.

SJS ENABLE SYSTEM MAP AND JUMP TO SUBROUTINE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	1
D/I															

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the System map, for translating all programmed memory references. Prior to enabling the System map, the P-register is set one count past the effective memory address (m + 1). After enabling the System map, the return address is stored in m. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the system map is enabled, the instruction is allowed and effectively executes a JSB *+1,I.

SSM STORE STATUS REGISTER INTO MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
D/I															

Memory Address

Stores the 16-bit contents of the MEM status register into the address memory location. The status register contents are not altered. This instruction is used in conjunction with the JRS instruction to allow easy processing of interrupts, which always select the System map (if the MEM is enabled). The format of the MEM status register is listed in table 3-5.

This instruction can cause an MEM violation only if write protection rules are violated.

SYA LOAD/STORE SYSTEM MAP PER A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0

Transfers the 32 System map registers to or from memory. If bit 15 of the A-register is clear, the System map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the System map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

An attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the System map is allowed within the constraints of write protected memory.

SYB LOAD/STORE SYSTEM MAP PER B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	1	0	0	0

Transfers the 32 System map registers to or from memory. If bit 15 of the B-register is clear, the System map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the System map is *stored* into memory starting at the address specified in bits 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

An attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the System map is allowed within the constraints of write protected memory.

UJP ENABLE USER MAP AND JUMP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	1	1	0
D/I															

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the User map, for translating all programmed memory references. Prior to enabling the

User map, the P-register is set to the effective memory address. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

UJS ENABLE USER MAP AND JUMP TO SUBROUTINE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1
D/I															

Memory Address

Causes the MEM hardware to use the set of 32 map registers, referred to as the User map, for translating all programmed memory references. Prior to enabling the User map, the P-register is set one count past the effective memory address (m + 1). After enabling the System map, the return address is stored in m. As a result of executing this instruction, normal I/O interrupts are held off until the first opportunity following the fetch of the next instruction, unless three or more levels of indirect addressing are used.

This instruction will normally generate an MEM violation when executed in the protected mode. In this case, the status of the MEM is not affected and the jump will not occur; however, if the System map is enabled, the instruction is allowed.

USA LOAD/STORE USER MAP PER A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	1

Transfers the 32 User map registers to or from memory. If bit 15 of the A-register is clear, the User map is *loaded* from memory starting from the address specified in bits 14-0 of the A-register. If bit 15 of the A-register is set, the User map is *stored* into memory starting at the address specified in bits 14-0 of the A-register. When the load/store operation is complete, the A-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

An attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the User map is allowed within the constraints of write protected memory.

USB LOAD/STORE USER MAP PER B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	0	1	0	0	1

Transfer the 32 User map registers to or from memory. If bit 15 of the B-register is clear, the User map is *loaded* from memory starting from the address specified in bits 14-0 of the B-register. If bit 15 of the B-register is set, the User map is *stored* into memory starting at the address specified in bits 14-0 of the B-register. When the load/store operation is complete, the B-register will be incremented by 32 to allow multiple map instructions.

Note: If not in the protected mode, the MEM provides no protection against altering the contents of maps while they are currently enabled.

Any attempt to load any map in the protected mode will cause an MEM violation. An attempt to store the User map is allowed within the constraints of write protected memory.

XCA CROSS COMPARE A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	0	1	1	0
D/I															

Memory Address

Compares the contents of the A-register with the contents of the addressed memory location. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances three counts instead of two counts. If the two words are identical, the next instruction is executed. Neither the A-register contents nor memory cell contents are altered.

This instruction uses the alternate program map to determine the addressed memory location. If the MEM is currently disabled, then a compare directly with physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XCB CROSS COMPARE B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	1	1	0
D/I															

Memory Address

Compares the contents of the B-register with the contents of the addressed memory location. If the two 16-bit words are not identical, the next instruction is skipped; i.e., the P-register advances three counts instead of two counts. If the two words are identical, the next instruction is executed. Neither the B-register contents nor memory cell contents are altered.

This instruction uses the alternate program map to determine the addressed memory location. If the MEM is currently disabled, then a compare directly with physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XLA CROSS LOAD A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0
D/I															

Memory Address

Loads the contents of the specified memory address into the A-register. The contents of the memory cell are not altered.

This instruction uses the alternate program map to fetch the operand. If the MEM is currently disabled, then a load directly from physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XLB CROSS LOAD B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	1	0	0
D/I															

Memory Address

Loads the contents of the specified memory address into the B-register. The contents of the memory cell are not altered.

This instruction uses the alternate program map to fetch the operand. If the MEM is currently disabled, then a load directly from physical memory occurs.

This instruction will cause an MEM violation only if read protection rules are violated.

XMA TRANSFER MAPS INTERNALLY PER A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	0	0	1	0

Transfers a copy of the entire contents (32 map registers) of the System map or the User map to the Port A map or the Port B map as determined by the control word in the A-register:

Bit*	Significance
15	0 = System Map 1 = User Map
0	0 = Port A Map 1 = Port B Map

*Bits 14-1 are ignored.

This instruction will always generate an MEM violation when executed in the protected mode.

XMB TRANSFER MAPS INTERNALLY PER B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	0	1	0

Transfers a copy of the entire contents (32 map registers) of the System map or the User map to the Port A map or the Port B map as determined by the control word in the B-register:

Bit*	Significance
15	0 = System Map 1 = User Map
0	0 = Port A Map 1 = Port B Map

*Bits 14-1 are ignored.

This instruction will always generate an MEM violation when executed in the protected mode.

XMM TRANSFER MAPS OR MEMORY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	0	0	0

Transfers a number of words either from sequential memory locations to sequential map registers or vice versa. The A-register points to the first map register to be accessed and the B-register points to the first word of a group of words (table) in sequential memory locations. The X-register indicates the number of maps (0 to 127_{10}) to be transferred. If the content of the X-register is a positive integer, words are moved from memory to map registers; if the content is a negative integer, words are moved from map registers to memory.

Map registers are addressed as a contiguous space and a wraparound count from 127 to 0 can and will occur. It is the programmer's responsibility to avoid this error.

The contents of the maps are transferred in blocks of 16 registers or less. This instruction is interruptible only after each block has been completely transferred.

An attempt to load any map register in the protected mode will generate an MEM violation. An attempt to store map registers is allowed within the constraints of write protected memory.

XMS TRANSFER MAPS SEQUENTIALLY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	0	0	1

Transfers a number of words to sequential map registers. The A-register points to the first register to be accessed, the B-register contains the base quantity, and the X-register indicates the number of maps (0 to 127$_{10}$) to be loaded. If the contents of the X-register is a positive integer, the contents of the B-register will be used as the base quantity to be loaded into the first map register. The second register will be loaded with the base quantity plus one, the third register will be loaded with the base quantity plus two, and so forth up to the number of map registers specified in the X-register. If the contents of the X-register is less than or equal to zero, an effective NOP will occur, leaving the contents of the A-, B-, and X-registers unaltered.

This instruction is interruptible after each group of 16 registers has been transferred. The A-, B-, and X-registers are then reset to allow reentry at a later time. The X-register will always be zero at the completion of the instruction and the A- and B-registers will be advanced by the number of registers moved.

An attempt to load any map register in the protected mode will generate an MEM violation.

XSA CROSS STORE A

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	1	1	1	0	1	0	1	0	1
D/I															

Memory Address

Stores the contents of the A-register into the addressed memory location. The previous contents of the memory cell are lost; the A-register contents are not altered.

This instruction uses the alternate program map for the write operation. If the MEM is currently disabled, then a store directly into physical memory occurs.

This instruction will always cause an MEM violation when executed in the protected mode.

XSB CROSS STORE B

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	1	0	1	0	1	0	1
D/I															

Memory Address

Stores the contents of the B-register into the addressed memory location. The previous contents of the memory cell are lost; the B-register contents are not altered.

This instruction uses the alternate program map for the write operation. If the MEM is currently disabled, then a store directly into physical memory occurs.

This instruction will always cause an MEM violation when executed in the protected mode.

3-35. SAMPLE MAP LOAD/ENABLE ROUTINE

Table 3-7 provides a sample DMS map load and enable routine. This routine begins by loading 32 registers for the System Map and 32 registers for the User Map and continues by setting the Port A Map to the area for User number one. The Port B Map is then set to point into a new area where a third User's program would be loaded. Next, the Base Page Fence is set so that the System Fence value is used. Finally, the mapping functions of the DMS are enabled and program control is transferred to the System area beginning at address 1000$_8$.

3-36. ADDITIONAL DMS DEFINITIONS

The following paragraphs further define the terms "alternate map" and "protected mode" and contain definitive discussions for MEM violations and DCPC operation in a DMS environment.

3-37. ALTERNATE MAP. If the system map is currently enabled, the user map is the alternate map. If the user map is currently enabled, the system map is the alternate map. The DCPC maps are never the alternate maps.

3-38. PROTECTED MODE. If the DMS and memory protect are enabled, the computer is in the protected mode. DMS will operate in the unprotected mode (DMS enabled, memory protect disabled), but none of the DMS safeguards will be operative.

3-39. MEM VIOLATIONS. The MEM violations are designed to safeguard DMS. The four types of violations are read protect, write protect, base page, and privileged instruction. Throughout the following paragraphs, references to logical memory refers to the memory address before mapping and references to physical memory refers to the memory address after mapping.

If the computer is in the protected mode and bit 11, the read protect bit, of a system or user map register equals 1, any attempt by the system or user to read from the associated memory page causes a read protect violation and the read does not occur. If the computer is in the unprotected mode, the read occurs. In either case, bit 15 of the MEM violation register will be set to 1. For example, suppose the computer is in the protected mode and the system or user map register 3 contains 4043$_8$. Any attempt by the system or user to read from page 43$_8$ using map register 3 (i.e., read from physical addresses in the 106000$_8$ to 107777$_8$ range), causes a read protect violation.

If the computer is in the protected mode and bit 10, the write protect bit, of a system or user map register equals 1, any attempt by the system or user to write onto the associated memory page causes a write protect violation and the write does not occur. If the computer is in the unprotected mode, the write occurs. In either case, bit 14 of the MEM violation register will be set to 1. For example, suppose the computer is in the protected mode and the system or user map register 3 contains 2043_8. Any attempt by the system or user to write onto page 43_8 using map register 3 (i.e., write onto physical addresses in the 106000_8 to 107777_8 range), causes a write protect violation.

If the computer is in the protected mode, any attempt by the system or user to write onto the physical base page causes a base page violation and the write does not occur. If the computer is in the unprotected mode, the write occurs. In either case, bit 13 of the MEM violation register will be set to 1. For example, suppose the computer is in protected mode, the system or user map register 0 contains 0040_8, the base page fence is set at 1000_8, and bit 10 of the MEM status register equals 1 (i.e., logical addresses below the base page fence are mapped). If the system or user attempts to write to a logical memory address of 1500_8, MEM detects that the base page addresses above the base page fence are not mapped and begins to access physical memory address 1500_8. However, MEM then detects that a write to physical base page is being attempted which causes a base page violation and the write does not occur. If the computer was in the unprotected mode, the write would have occurred. In either case, bit 13 of the MEM violation register will be set to 1. If the system or user attempts to write to a logical memory address of 500_8, MEM detects that addresses below the base page fence are mapped and begins to access physical memory address $100000_8 + 500_8 = 100500_8$ where the write will occur providing standard memory protect is not violated. Note that standard memory protect checks the logical address (i.e., 500_8), not the physical address (i.e., 100500_8). Reading from logical or physical base page will not generate a base page violation. From the previous discussion, it can be seen that a DMS memory space has its base page in two pieces which may or may not be contiguous. Regardless, the total base page available for any DMS memory space is 1024 locations. The part of the physical base page accessible by all memory spaces is also referred to as the unmapped or shared part of the base page. Note that the logical addresses of 0 and 1 access the A and B registers, respectively.

If the computer is in the protected mode, any attempt by the user to load into any MEM register, except the MEM address register, will cause a privileged instruction violation and the load will not occur. Any attempt by the system to load into any of the MEM map registers will cause a privileged instruction violation and the load will not occur. If the computer is in the unprotected mode, the load occurs. In either case, bit 12 of the MEM violation register will be set to 1. The system can always load into the MEM state register or MEM fence register. Under microprogrammed control the user can always load into the MEM state register or MEM fence register. The system or user can always load into the MEM address register, and can always read the MEM map registers. All MEM violations cause an interrupt to select code 5. Instruction SFS 5 will skip only for an MEM violation, allowing DMS interrupts to be differentiated from memory protect or parity error interrupts.

3-40. DCPC OPERATION IN A DMS ENVIRONMENT. DCPC activity disables the MEM violation logic. Therefore, the DCPC's can read or write physical memory without generating MEM violations. Note that mapping remains enables during DCPC activity and that the base page partitioning is the same. For example, if a DCPC input transfer were aimed at logical memory addresses 0 to 77777_8, which happened to map to physical addresses 100000_8 to 177777_8, and the conditions cited in the base page examples prevailed, then the input data would be written into physical addresses 100000_8 to 100777_8, 1000_8 to 1777_8, and 102000_8 to 177777_8.

[illegible]

Table 3-8 lists the execution times required for the various instructions. Since the timing requirements of the input/output group instructions depend on the time period in which the instruction begins, programs should not rely on the execution times for accurate, real-time measurements.

There are no phases in the processors in the true sense of the word. Phases such as execute, indirect, interrupt, and fetch are inherent in the state of the processor, which can be defined as the contents of the read-only memory (ROM) address register.

3-42. INTERRUPT SYSTEM

The vectored priority interrupt system has up to 60 distinct interrupt levels, each of which has a unique priority assignment. Each interrupt level is associated with a numerically corresponding interrupt location in memory.

Of the 60 interrupt levels, the two highest priority levels are reserved for hardware faults (power fail and parity error), the next two are reserved for Dual-Channel Port Controller completion interrupts, and the remaining levels are available for I/O device channels. Tables 3-9, 3-10, and 3-11 list the interrupt levels in priority order for the HP 2105A, HP 2108A, and HP 2112A Processors, respectively.

As an example of the simplicity of the interrupt system, an interrupt request from I/O channel 12 will cause an interrupt to memory location 00012. This request for service will be granted on a priority basis higher than afforded to channel 13 but lower than that afforded to channel 11. Thus, a transfer in progress via channel 13 would be suspended to allow channel 12 to proceed. On the other hand, a transfer in progress via channel 11 cannot be interrupted by channel 12.

Any device can be selectively enabled or disabled under program control, thus switching the device into or out of the interrupt structure. In addition, the entire interrupt system, except power fail and parity error interrupts, can be enabled or disabled under program control using a single instruction.

Interrupt requests received while the computer is in the halt mode will be processed, in order of priority, when the computer is placed in the run mode. Input/output priority is covered in more detail in section IV.

Table 3-7. Sample DMS Load/Enable Routine

LABEL	OPCODE	OPERAND	COMMENTS
DMS	NOP		DMS Load/Enable Routine
	LDA	S.TABL	Load address of System Map Table
	LDB	U.TABL	Load address of User Map Table
	SYA		Load System Map from memory
	USB		Load User Map from memory
	LDA	A.TABL	Load address of Port A Map Table
	LDB	B.TABL	Load address of Port B Map Table
	PAA		Load Port A Map from memory
	PBB		Load Port B Map from memory
	LDA	FNC,I	Load fence value
	LFA		Load Base Page Fence register
	SJP	SYSTRT,I	Enable System Map and jump to operating system entry point
SYSTRT	OCT	1000	Operating system begins at 1000_8
FNC	DEF	SYSF	Points to System Fence
S.TABL	DEF	SYSTEM	Points to System Table
U.TABL	DEF	US01	Points to first User Table
A.TABL	DEF	US01	Points to first User Table
B.TABL	DEF	US03	Points to third User Table
SYSF	OCT	100	Fence for operating system
SYSTEM	OCT	0	System Map Table
	OCT	1	System Map Table
	OCT	2	System Map Table
⋮			
US01F	OCT	1000	Fence for first user
US01	OCT	40	First User Map Table
	OCT	41	First User Map Table
	OCT	42	First User Map Table
⋮			
US02F	OCT	2044	Fence for second user
US02	OCT	100	Second User Map Table
	OCT	101	Second User Map Table
	OCT	102	Second User Map Table
⋮			
US03F	OCT	0	Fence for third user
US03	OCT	200	Third User Map Table
	OCT	201	Third User Map Table
	OCT	202	Third User Map Table
⋮			

Table 3-8. Instruction Execution Times

INSTRUCTION	EXECUTION TIME (μS)
Memory Reference Group[1,2]	
ADA/B, AND, IOR, LDA/B, XOR	1.94
STA/B	2.27
CPA/B (no skip)	2.27
(skip)	2.59
ISZ (no skip)	2.59
(skip)	2.92
JMP	1.94
JSB	2.27
Shift/Rotate Group[3]	2.59 - 2.92
Alter/Skip Group[3]	
No skip, no increment	2.59
No skip, increment A/B	2.92
Skip, no increment	2.59
Skip, increment A/B	2.92
Input/Output Group[4]	2.59 - 3.89
Extended Arithmetic Group[5]	
ASL, ASR, LSL, LSR, RRL, RRR	3.57 - 8.43
DLD	4.54
DST	4.86
MPY	12.32 - 13.30
DIV	15.92 - 18.20
Floating Point Group	
FAD	21.78 - 53.95
FDV	41.2 - 75.72
FIX	6.50 - 12.02
FLT	10.72 - 34.42
FMP	48.10 - 56.88
FSB	22.75 - 57.20

INSTRUCTION	EXECUTION TIME (μS)
Extended Instruction Group	
CAX, CBX, CAY, CBY CXA, CXB, CYA, CYB	2.275
XAX, XBX, XAY, XBY ISX, ISY, DSX, DSY	3.250
LDX, LDY	
(direct address)	4.875
(indirect address)	4.875[a]
STX, STY	
(direct address)	5.20
(indirect address)	5.20[a]
LAX, LBX, LAY, LBY	
(direct address)	4.875
(indirect address)	5.525[a]
SAX, SBX, SAY, SBY	
(direct address)	5.20
(indirect address)	5.85[a]
ADX, ADY	
(direct address)	4.875
(indirect address)	4.875[a]
JLY (direct address)	5.525
(indirect address)	5.525[a]
JPY	4.55
LBT	4.875 avg
SBT	6.01 avg
MBT	8.775[a,b,g]
MVW	7.8[a,c,g]
CBT	8.775[a,d,g]
CMW	7.8[a,e,g]
SFB (for test byte match)	3.575[f,g]
(for term. byte match)	2.275[f,g]
CBS, SBS	7.8[a]
TBS	8.125[a]

---

[1] Memory refresh consumes 0.65 μS maximum no more often than every 30 μS.

[2] Add 1.3 μS for each indirect address level.

[3] NOP or RSS requires 2.92 μS whereas a JMP *+1 or JMP *+2 requires only 1.94 μS.

[4] Depends on which I/O time period (T2, 3, 4, 5, 6) the instruction begins.

[5] Depends on number of shifts specified (1 to 16).

a. Add 1.3 μS for each indirect address level.
b. Add 7.31 μS for each byte moved or compared.
c. Add 3.25 μS for each word moved or compared.
d. Add 8.125 μS for each byte moved or compared.
e. Add 3.575 μS for each word moved or compared.
f. Add 4.875 μS for each byte moved or compared.
g. Add 7.15 μS for each interrupt of the instruction.

Table 3-8. Instruction Execution Times (Continued)

INSTRUCTION	EXECUTION TIME (μS)	NOTES
Dynamic Mapping System Group		
DJP, SJP	5.85^a	
DJS, SJS	6.50^a	
JRS	$9.10 - 10.40^a$	
LFA/B	3.57	
MBF, MBI, MBW	$6.50^{b,c}$	
MWF, MWI, MWW	3.25^b	
PAA/B, SYA/B	47.125 – 47.80	
PBA/B, USA/B	47.125 – 47.80	
RSA/B	2.60	
RVA/B	2.275	
SSM	5.85^a	
UJP	5.525^a	
UJS	6.175^a	
XCA/B	6.175^a	
XLA/B, XSA/B	5.525^a	
XMA/B	15.275 – 16.575	
XMM	9.75^e	
XMS	8.45^d	

a. Add 1.3 μS for each indirect address level.
b. Add 2.925 μS for each word moved.
c. Add 3.575 μS for last odd byte.
d. Add 0.975 μS for each word loaded into map register.
e. Add 1.3 μS for each word exchanged between maps and memory.

Table 3-9. HP 2105A Interrupt Assignments

CHANNEL (Octal)	INTERRUPT LOCATION	ASSIGNMENT
04	00004	Power Fail Interrupt
06	00006	DCPC Channel 1 Completion Interrupt
07	00007	DCPC Channel 2 Completion Interrupt
10	00010	I/O Device (highest priority)
11 - 13	00011-00013	I/O Device (Mainframe)
14 - 35	00014-00035	I/O Device (Extender No. 1)
36 - 57	00036-00057	I/O Device (Extender No. 2)

Table 3-10. HP 2108A Interrupt Assignments

CHANNEL (Octal)	INTERRUPT LOCATION	ASSIGNMENT
04	00004	Power Fail Interrupt
05	00005	Memory Parity/Protect Interrupt
06	00006	DCPC Channel 1 Completion Interrupt
07	00007	DCPC Channel 2 Completion Interrupt
10	00010	I/O Device (highest priority)
11 - 20	00011-00020	I/O Device (Mainframe)
21 - 42	00021-00042	I/O Device (Extender No. 1)
43 - 64	00043-00064	I/O Device (Extender No. 2)

Table 3-11. HP 2112A Interrupt Assignments

CHANNEL (Octal)	INTERRUPT LOCATION	ASSIGNMENT
04	00004	Power Fail Interrupt
05	00005	Memory Parity/Protect Interrupt
06	00006	DCPC Channel 1 Completion Interrupt
07	00007	DCPC Channel 2 Completion Interrupt
10	00010	I/O Device (highest priority)
11 - 25	00011-00025	I/O Device (Mainframe)
26 - 47	00026-00047	I/O Device (Extender No. 1)
50 - 71	00050-00071	I/O Device (Extender No. 2)

3-43. POWER FAIL INTERRUPT

The computer is equipped with power-sensing circuits. When primary (mains) power fails or drops below a predetermined operating level while the computer is running, an interrupt to memory location 00004 is automatically generated. This interrupt is given the highest priority in the system and cannot be turned off or otherwise disabled. Memory location 00004 is intended to contain a jump-to-subroutine (JSB) instruction referencing the entry point of a power fail subroutine; however, location 00004 may alternatively contain a halt (HLT) instruction. The interrupt cabability of lower-priority operations is automatically inhibited while a power fail subroutine is in process.

A minimum of 500 microseconds is available between the detection of a power failure and the loss of usable power supply power to execute a power fail subroutine; the purpose of such a subroutine is to transfer the current state of the computer system into memory and then halt the computer. A sample power fail subroutine is given in table 3-12. The optional battery will supply enough power to preserve the contents of memory for a sustained power mains outage of up to 2 hours.

If the optional Dynamic Mapping System (DMS) is installed and a power failure occurs, the System Map is automatically enabled just prior to fetching the instruction in location 00004. Since all maps are disabled and none are considered valid upon the restoration of power, the power fail subroutine should include the necessary instructions to save as many maps as desired and restore them prior to enabling the DMS.

Since the computer might be unattended by an operator, the user has a switch-selectable option of what action the computer will take upon the restoration of primary power. When the switch (A1S2) is set to the $\overline{\text{ARS}}$ position, the computer will halt when power is restored regardless of whether the computer was running or halted when the failure occurred. (No operator panel indication is given.)

Note: Switch A1S2 is mounted on the CPU and is not considered an operator control. The setting of this switch is normally determined prior to or during system installation.

When A1S2 is in the ARS position, the automatic restart feature is enabled. After a built-in delay of about half a second following the return to normal power levels, another interrupt to location 00004 occurs. This time the power-down portion of the subroutine is skipped and the power-up portion begins. (Refer to table 3-12). If the computer was not running when the power failure occurred, the computer is halted immediately. If the computer was running, those conditions existing at the time of the power fail interrupt are restored and the computer continues the program from the point of the interruption. Alternatively, if location 00004 contains a HLT instruction instead of a JSB instruction, the

Table 3-12. Sample Power Fail Subroutine

LABEL	OPCODE	OPERAND	COMMENTS
PFAR	NOP		Power Fail/Auto Restart Subroutine
	STF	6B	Terminates DCPC Channel 1
	STF	7B	Terminates DCPC Channel 2
	SFC	4B	Skip if interrupt was caused by a power failure
	JMP	UP	Power is being restored, reset state of computer system
DOWN	STA	SAVA	Save A-register contents
	CCA		Set switch indicating that the computer was running
	STA	SAVR	when power failed
	STB	SAVB	Save B-register contents
	ERA,ALS		Transfer E-register content to A-register bit 15
	SOC		Increment A-register if Overflow
	INA		is set
	STA	SAVEO	Save E- and O-register contents
	LDA	PFAR	Save contents of P-register at time of
	STA	SAVP	power failure
	LIA	1B	Save contents of
	STA	SAVS	S-register
	STX	SAVX	Save contents of X-register
	STY	SAVY	Save contents of Y-register
	⋮		Insert user-written routine to save I/O
			device states
	CLC	4B	Turn on restart logic so computer will restart when power is restored
			after momentary power failure
	HLT		Shutdown
UP	LDA	SAVR	Was computer running
	SZA,RSS		when power failed?
	JMP	HALT	No
	CLA		Yes, reset computer Run switch to
	STA	SAVR	initial state
	LDA	FENCE	Restore the memory protect
	OTA	5B	fence register contents
	⋮		Insert user-written routine to restore
			I/O device states
	LDA	SAVEO	Restore the contents
	CLO		of the
	SLA,ELA		E-register and
	STF	1B	O-register
	LDA	SAVS	Restore the contents of the
	OTA	1B	S-register
	LDA	SAVA	Restore A-register contents
	LDB	SAVB	Restore B-register contents
	LDX	SAVX	Restore X-register contents
	LDY	SAVY	Restore Y-register contents
	STC	4B	Reset power fail logic for next power failure
	STC	5B	Turn on memory protect
	JMP	SAVP,I	Transfer control to program in execution at time of power failure
HALT	HLT		Return computer to halt mode
FENCE	OCT	2000	Fence address storage (must be updated each time fence is changed)
SAVEO	OCT	0	Storage for E and O
SAVA	OCT	0	Storage for A
SAVB	OCT	0	Storage for B
SAVS	OCT	0	Storage for S
SAVX	OCT	0	Storage for X
SAVY	OCT	0	Storage for Y
SAVP	OCT	0	Storage for P
SAVR	OCT	0	Storage for Run switch

computer will halt and light the POWER FAIL/BATTERY indicator.

To allow for the possibility of a second power failure occurring while the power-up portion of the subroutine is in process, the user should limit the combined power-down and power-up instructions to less than 100. If the computer memory does not contain a subroutine to service the interrupt, location 00004 should contain a HLT 04 instruction (102004 octal).

A Set Control instruction (STC 04) must be given at the end of any restart routine. This instruction re-initializes the power-fail logic and restores the interrupt capability to the lower priority functions. Pressing the PRESET switch on the operator panel performs the same function as the STC 04 instruction. Pressing and holding the PRESET switch will force a halt when the key-operated switch is set to OPERATE.

The optional battery sustains the contents of memory when mains power is off. If the battery becomes discharged when mains power is off, the operator must turn the operator panel key switch to the reset (R) position before the computer will operate with mains power restored. This will also clear the entire memory to zeros in order to restore correct parity (only if memory is lost).

3-44. PARITY ERROR INTERRUPT

Parity checking of memory is a stantdard feature in the computer. The parity logic continuously generates correct parity for all words written into memory and monitors the parity of all words read out of memory. Correct parity is defined as having the total number of "1" bits in a 17-bit memory word (16 data bits plus the parity bit) equal to an odd value. If a "1" bit (or any odd number of "1" bits) is either dropped or added in the transfer process, a Parity Error signal is generated when that word is read out of memory.

The Parity Error signal may either halt the computer or cause the computer to take some other action as determined by an internal switch (A1S1) mounted on the CPU. When the switch is in the HALT PE position and a parity error occurs, the computer will halt and light the PARITY indicator. The PARITY indicator will remain lighted until the PRESET switch is pressed.

Note: Switch A1S1 is mounted on the CPU and is not considered an operator control. The setting of this switch is normally determined prior to or during installation or when the memory protect PCA is installed at the user's site.

If switch A1S1 is in the INT/IGNORE position, the action that the computer will take when a parity error occurs is as follows:

a. If the memory protect PCA is installed and the parity error logic has not been disabled by a CLF 05 instruction, an interrupt to memory location 00005 is generated. This location may alternatively contain a JSB instruction referencing the entry point of a user-written memory protect subroutine, or contain a HLT instruction.

b. If the memory protect PCA is not installed, or if the memory protect option is installed but the parity error logic has been disabled by a CLF 05 instruction, the parity error will be ignored and the PARITY indicator will light.

Note: Memory protect is an option available only with the HP 2108A and HP 2112A Processors. The memory protect PCA is dedicated to memory slot 111.

In conjunction with memory protect, it is possible to determine the memory address containing the parity error. The error address will be loaded automatically into the violation register of the memory protect logic and from there it is accessible to the user by programming an LIA 05 or LIB 05 instruction.

When a parity error occurs, it is recommended that the entire program or set of data containing the error location be reloaded. However, by knowing the address and the contents of the error location, the user may be able to determine what operations have taken place as a result of reading the erroneous word. For example, if the erroneous word was an instruction, several other locations may be affected. By individually checking and correcting the contents of all affected memory locations, the user may resume running the program without the necessity of a complete reload. If software is being generated, this may also need correcting.

3-45. MEMORY PROTECT/DMS INTERRUPT

The memory protect option provides the capability of protecting a selected block of memory of any size, from a settable fence address downward, against alteration by programmed instructions except those directly involving the A- and B-registers. Any programmed instruction except JMP may freely address the A- and B-registers as locations 00000 and 00001 (octal), respectively.

The memory protect logic, when enabled by an STC 05 instruction, also prohibits the execution of all I/O instructions (including HLT 01) except those referencing I/O select code 01, the S-register and the overflow register. This feature limits the control of I/O operations to interrupt control only. Thus, by programming the system to direct all I/O interrupts to an executive program residing in protected memory, the executive program can have exclusive control of the I/O system.

The memory protect logic is disabled automatically by any interrupt (except when the interrupt location contains an I/O instruction) and must be re-enabled by an STC 05 instruction at the end of each interrupt subroutine.

The optional DMS hardware includes additional memory protect features, which are enabled or disabled simultaneous with the memory protect hardware. When enabled by an STC 05 instruction, the DMS hardware provides the capability of read/write protecting memory on a 1024-word page basis. Included in the DMS are several privileged instructions which are not allowed when the memory protect logic is enabled. Upon detection of a violation, an interrupt to location 00005 is generated. A DMS interrupt is signified by executing an SFS 05 instruction since the DMS will set the flag on channel 05; a memory protect interrupt is signified by executing an SFC 05 instruction.

Programming rules pertaining to the use of memory protect are as follows (assuming that an STC 05 instruction has been given):

a. Location 00002 is the lower boundary of protected memory. (Locations 00000 and 00001 are the A- and B-register addresses.)

b. JMP instructions may not reference the A- or B-register; however, a JSB instruction may do so.

c. The upper boundary (memory address) is loaded into the fence register from the A- or B-register by an OTA 05 or OTB 05 instruction, respectively. Memory locations below but not including this address are protected.

d. Execution will be inhibited and an interrupt to location 00005 will occur if a JMP, JSB, ISZ, STA, STB, or DST instruction* either directly or indirectly addresses a location in protected memory, or if any I/O instruction is attempted (including HLT but excluding those addressing select code 01, the S-register and the overflow register). After three successive levels of indirect addressing, the memory protect logic will allow a pending I/O interrupt.

e. Any instruction not mentioned in step d of this paragraph is legal even if the instruction directly references a protected memory address. In addition, indirect addressing through protected memory by those instructions listed in step d is legal provided that the ultimate effective address is outside the protected memory area.

Following a memory protect interrupt, the address of the illegal instruction will be present in the violation register. This address is made accessible to the programmer by an LIA 05 or LIB 05 instruction, which loads the address into the A- or B-register.

Since parity error and memory protect share the same interrupt location, it is necessary to distinguish which type of error is responsible for the interrupt. A parity error is indicated if, after the LIA (or LIB) 05 instruction is executed, bit 15 of the selected register is a logic 1; a memory protect violation is indicated if bit 15 is a logic 0. In either case, the remaining 15 bits of the selected register contains the octal address of the error location.

Table 3-13 illustrates a sample memory protect and parity error subroutine. An assumption made for this example is that the location immediately following the error location is an appropriate return point. This may not always be the case, however, because it may be deemed advisable to abort the program in process and return to a supervisory program.

3-46. DUAL-CHANNEL PORT CONTROLLER INTERRUPT

The optional Dual-Channel Port Controller (DCPC) allows high-speed block transfer of data between input/output devices and memory. For the most part, the DCPC operates independently of the interrupt system in that the only time that a DCPC interrupt occurs is when the specified block of data has been transferred. Since there are two DCPC channels, two interrupt locations are reserved for this purpose; location 00006 is reserved for channel 1 and location 00007 is reserved for channel 2. Channel 1 interrupt has priority over the channel 2 interrupt. Because DCPC interrupts are primarily completion signals to the programmer, and are therefore application dependent, no interrupt subroutine example is considered necessary.

3-47. INPUT/OUTPUT INTERRUPT

The remaining interrupt locations (00010 through 00077 octal) are reserved for I/O devices; this represents a total of 56 (decimal) locations, one for each I/O channel. In a typical I/O operation, the computer issues a programmed command such as Set Control/Clear Flag (STC,C) to one or more external devices to initiate an input (read) or an output (write) operation. Each device will then either put data into or accept data from an input/output buffer on its associated interface PCA. During this time, the computer may continue running a program or may be programmed into a waiting loop to wait for a specific device to complete a read or write operation. Upon the completion of a read or write operation, each device returns a Flag signal to the computer. These Flag signals are passed through a priority network which allows only one device to be serviced regardless of the number of Flag signals present at that time. The Flag signal with the highest priority generates an Interrupt signal at the end of the current machine cycle except under the following circumstances:

---

*Also CBT, JLY, JPY, MVB, MVW, SAX, SAY, SBX, SBY, STX, and STY of the extended instruction group.

Table 3-13. Sample Memory Protect/Parity Error Subroutine

LABEL	OPERCODE	OPERAND	COMMENTS
MPPE	NOP		Memory Protect/Parity Error Subroutine
	CLF	0B	Turn off interrupt system to inhibit I/O devices
	STA	SVA	Save A-register contents
	STB	SVB	Save B-register contents
	LIA	5B	Get contents of violation register in MP logic
	CLF	5B	Turn off PE interrupt during subroutine; clear violation register bit 15
	SSA		Check bit 15 to determine kind of error
	JMP	PERR	If a 1, go to parity error routine
	JMP	MPTR	If a 0, go to memory protect routine
MPTR	—		User's routine in case of memory protect violation
	—		
	—		
	etc.		
	—		
	—		
	—		
	LDA	SVA	Restore A-register
	LDB	SVB	Restore B-register
	STF	0B	Enable interrupt system
	STF	5B	Turn on parity error interrupt
	STC	5B	Turn on memory protect interrupt
	JMP	MPPE,I	Exit the subroutine
PERR	—		User's routine in case of parity error
	—		
	—		
	etc.		
	—		
	—		
	—		
	JMP	PERR-6B	Restore accumulators, turn on interrupts, exit

a. Interrupt system disabled or interface PCA interrupt disabled.

b. JMP indirect or JSB indirect instruction not sufficiently executed. These instructions inhibit all interrupts except power fail or memory protect until the succeeding instruction is executed.

c. Instruction in an interrupt location not sufficiently executed, even if that interrupt is of lower priority. Any interrupt inhibits the entire interrupt system until the succeeding instruction is executed.

d. Optional dual-channel port controller in the process of transferring data.

e. Current instruction is one that may affect the priorities of I/O devices; e.g., Set Control (STC), Clear Control (CLC), Set Flag (STF), and Clear Flag (CLF). The interrupt in this case must wait until the succeeding instruction is executed.

After an interface PCA has been issued a Set Control command and its Flag flip-flop becomes set, all interrupt requests from lower-priority devices are inhibited until this Flag flip-flop is cleared by a Clear Flag (CLF) instruction. A service subroutine in process for any device can be interrupted only by a higher-priority device; then, after the higher-priority device is serviced, the interrupted service subroutine may continue. In this way it is possible for several service subroutines to be in the interrupt state at one time; each of these service subroutines will be allowed to continue after the higher-priority device is serviced. All such service subroutines normally end with a JMP indirect instruction to return the computer to the point of the interrupt.

3-48. CENTRAL INTERRUPT REGISTER

Each time an interrupt occurs, the address of the interrupt location is stored in the central interrupt register. The contents of this register are accessible at any time by executing an LIA 04 or LIB 04 instruction. This loads the address of the most recent interrupt into the A- or B-register. As described in section II, the central interrupt register contents can be accessed from the operator panel be entering the special register display mode.

3-49. INTERRUPT SYSTEM CONTROL

I/O address 00 is the master control address for the interrupt system. An STF 00 instruction enables the entire interrupt system and a CLF 00 disables the interrupt system. The two exceptions to this are the power fail interrupt, which cannot be disabled, and parity error interrupt, which can only be selective enabled or disabled by an STF 05 or CLF 05, respectively.

Whenever power is initially applied, a clear signal to I/O address 00 automatically disables the interrupt system. Programs dependent on the interrupt mode of operation must include an STF 00 instruction to ensure that the interrupt system is enabled in the run mode.

SECTION

IV

INPUT/OUTPUT SYSTEM

The purpose of the input/output system is to transfer data between the computer and external devices. As shown in figure 4-1, data is normally transferred through the A- or B-register. An input transfer of this type occurs in three distinct steps: (1) between the external device and its interface PCA in the computer, (2) between the interface PCA and the A- or B-register via the I/O bus and CPU, and (3) between the A- or B-register and memory via the S-bus and memory controller. This three-step process also applies to an output transfer except in reverse order. This type of transfer, which is executed under program control, allows the computer logic to manipulate the data during the transfer process.

Also shown in figure 4-1, data may be transferred automatically under control of the Dual-Channel Port Controller (DCPC) option. Once the DCPC has been initialized, no programming is involved and the transfer is reduced to a two-step process: (1) between the external device and its interface PCA in the computer and (2) between the interface PCA and memory via the I/O bus, S-bus, and memory controller. The two DCPC channels are assignable to operate with any two device interface PCA's.

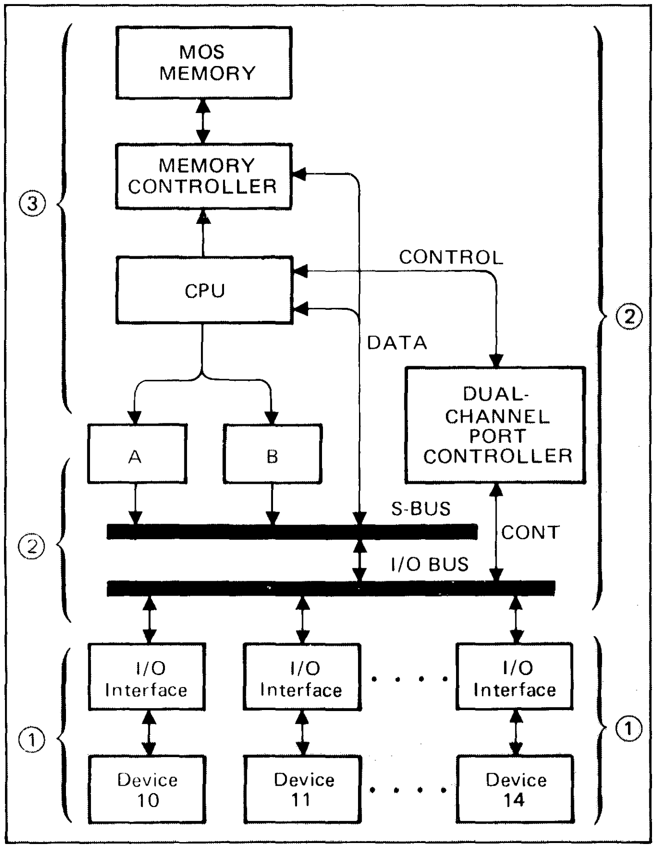


Figure 4-1. Input/Output System

Since a DCPC transfer eliminates programmed loading and storing via the accumulators, the time involved is very short. Thus, the DCPC is used with high-speed devices capable of transferring data at rates up to 616,666 sixteen-bit words per second. Further information on the DCPC option is given under paragraph 4-13.

4-1. INPUT/OUTPUT ADDRESSING

As shown in figure 4-2, an external device is connected by cable directly to an interface PCA located inside the computer mainframe. The interface PCA, in turn, plugs into one of the input/output slots, each of which is assigned a fixed address commonly referred to as the device select code. The computer can then communicate with a specific device on the basis of its select code.

Figure 4-2 shows an interface PCA inserted in the I/O slot having the highest priority; this channel is assigned select code 10 (octal). If it is decided that the associated device should have lower priority, its interface PCA and cable may simply be exchanged with those occupying some other I/O slot. This will change both the priority and the I/O address; however, due to priority chaining (refer to paragraph 4-2), there can be no vacant slots from select code 10 to the highest used select code (if the interrupt mode is to be used).

Only select codes 10 through 77 (octal) are available for input/output devices; the lower select codes (00 through 07) are reserved for other features. Figure 4-2 illustrates the I/O select codes available in the HP 2105A, HP 2108A, and HP 2112A Processor mainframes.

Select codes (channels) higher than those shown in figure 4-2 are available through the use of one or two I/O extenders. Each I/O extender provides an additional 16 I/O channels, which are an extension of the processor's vectored priority interrupt system. Select codes in the extender(s) operate at the same speed and with the same versatility as those in the processor mainframe.

4-2. INPUT/OUTPUT PRIORITY

When a device is ready to be serviced, it causes its interface PCA to request an interrupt so that the computer will interrupt the current program and service the device. Since many device interface PCA's will be requesting service at random times, it is necessary to establish an orderly sequence for granting interrupts. Secondly, it is desirable that high-speed devices should not have to wait for low-speed device transfers. Both of these requirements are met by a series-linked priority structure illustrated by

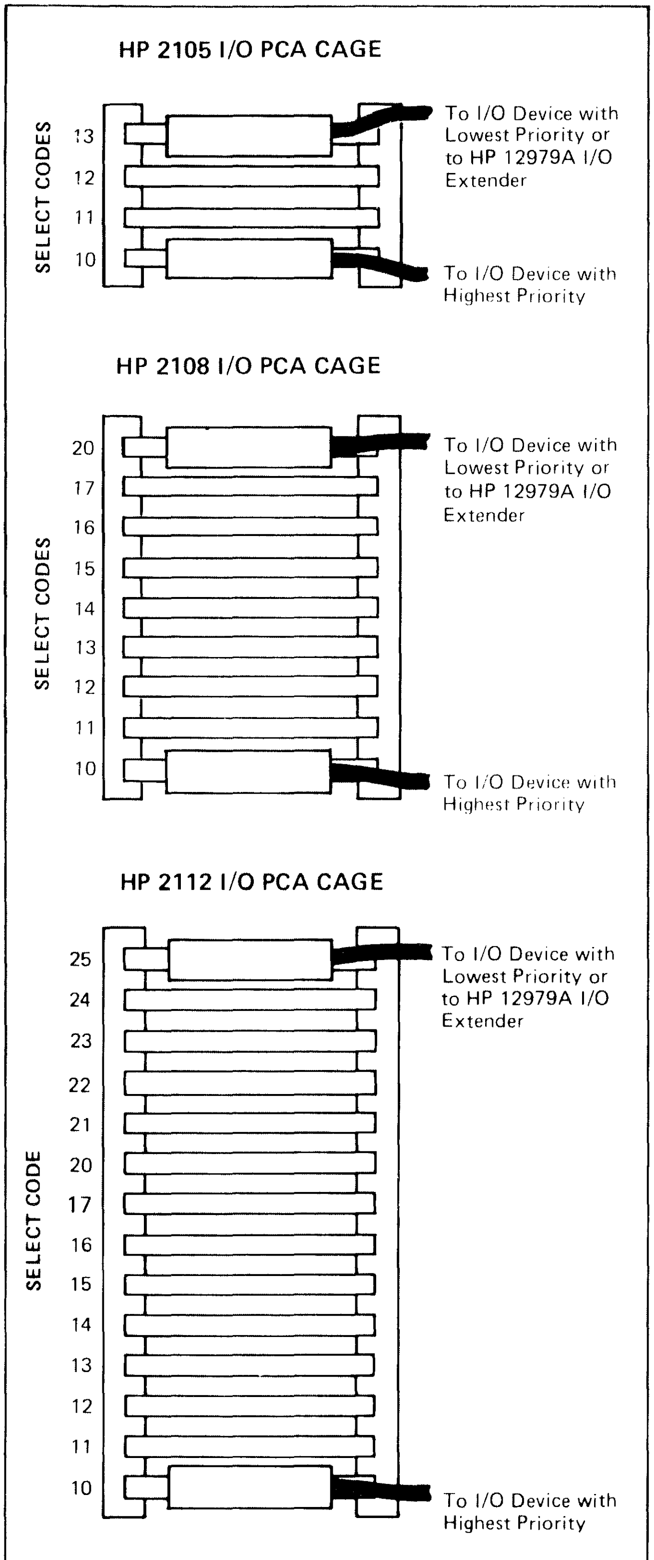


Figure 4-2. I/O Address Assignments

figure 4-3. The bold line, representing a priority enabling signal, is routed in series through each PCA capable of causing an interrupt. The PCA cannot interrupt unless this enabling signal is present at its input.

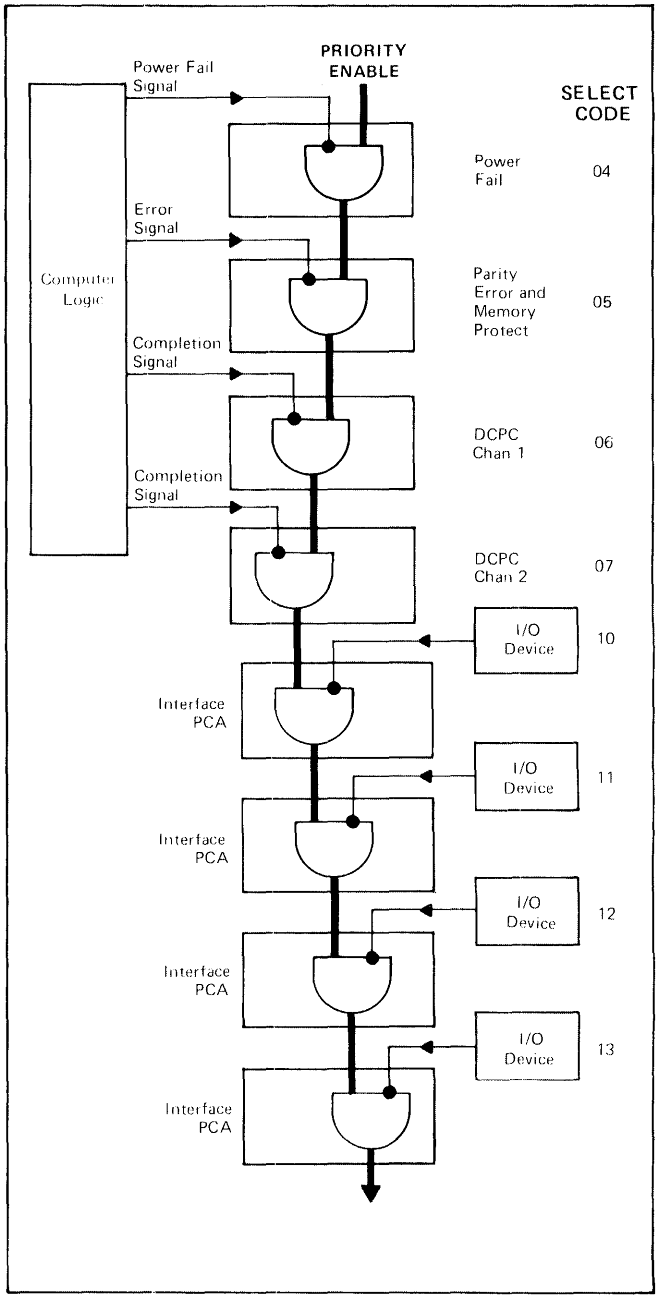


Figure 4-3. Priority Linkage

Each device (or other interrupt function) can break the enabling line when it requests an interrupt. If two devices simultaneously request an interrupt, obviously the device with the lowest select code will be the first one that can interrupt because it has broken the enable line for the higher select code. The other device cannot begin its service routine until the first device is finished; however, a still higher priority device (one with a lower select code) may interrupt the service routine of the first device. Figure 4-4 illustrates a hypothetical case in which several devices require service by interrupting a CPU program. Both simultaneous and time-separated interrupt requests are considered.

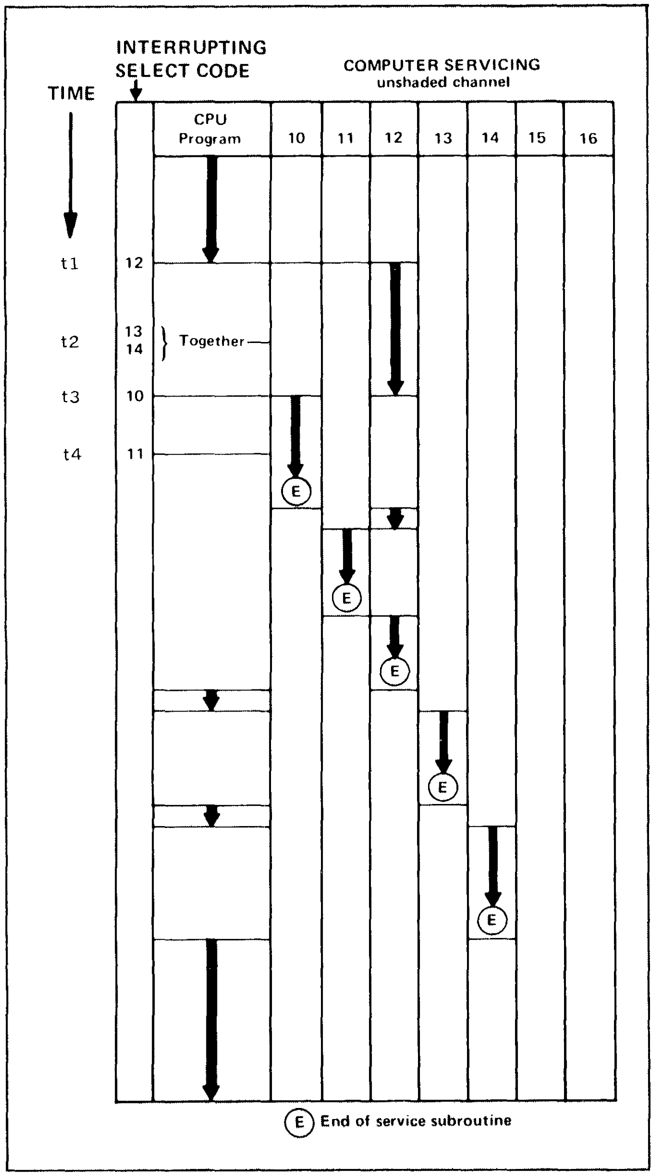


Figure 4-4. Interrupt Sequences

Assume that the computer is running a CPU program when an interrupt from I/O channel 12 occurs (at reference time t1). A JSB instruction in the interrupt location for select code 12 causes a program jump to the service routine for the channel 12 device. The JSB instruction automatically saves the return address (in a location which the programmer must reserve in his routine) for a later return to the CPU program.

The routine for channel 12 is still in progress when several other devices request service (set flag). First, channels 13 and 14 request simultaneously at t2; however, since neither one has priority over channel 12, their flags are ignored and channel 12 continues its transfer. But at t3, a higher priority device on channel 10 requests service. This request interrupts the channel 12 transfer and causes the channel 10 transfer to begin. The JSB instruction saves the return address for return to the channel 12 routine.

During the channel 10 transfer, device 11 sets the channel 11 flag (t4). Since it has lower priority than channel 10, device 11 must wait until the end of the channel 10 routine. And since the channel 10 routine, when it ends, contains a return address to the channel 12 routine, program control temporarily returns to channel 12 (even though the waiting channel 11 has higher priority). The JMP,I instruction used for the return inhibits all interrupts until fully executed. At the end of this short interval, the channel 11 interrupt request is granted.

When channel 11 has finished its routine, control is returned to channel 12, which at last has sufficient priority to complete its routine. Since channel 12 has been saving a return address in the main CPU program, it returns control to this point.

The two waiting interrupt requests from channels 13 and 14 are now enabled. Channel 13 has the higher priority and goes first. At the end of the channel 13 routine, control is temporarily returned to the CPU program. Then, the lowest priority channel (channel 14) interrupts and completes its transfer. Finally, control is returned to the CPU program, which resumes processing.

[illegible]

The interface PCA provides the communication link between the computer and an external device. The interface PCA includes three basic elements which either the computer or the device can control in order to effect the necessary communication. These three elements are the control bit, flag bit, and buffer.

4-4. CONTROL BIT

This is a one-bit register used by the computer to turn on the device channel. When set, the control bit generates a start command to the device, allowing it to perform one operation cycle (e.g., read or write one character or word). The interface PCA cannot interrupt unless the control bit is set. The control bit is set by an STC (set control) instruction and cleared by a CLC (clear control) instruction, both of which must be accompanied by a specific select code (e.g., STC 12 or CLC 12). The device cannot affect the control bit.

4-5. FLAG BIT

This is a one-bit register primarily used by the device to indicate (when set) that a transmission between the device and the interface PCA buffer has been completed. Computer instructions can also set the flag (STF), clear the flag (CLF), test if it is set (SFS), and test if it is clear (SFC). The device cannot clear the flag bit. If the corresponding control bit is set, priority is high, and the interrupt system is enabled, setting the flag bit will cause an interrupt to the location corresponding to the device select code.

4-6. BUFFER

The buffer register is used for intermediate storage of data. Typically, the data capacity is 8 or 16 bits, but this is entirely dependent on the type of device.

The following paragraphs describe how data is transferred between memory and input/output devices. A summary of I/O group instructions pertinent to the computer interrupt and control functions is provided in the appendix. The sequences presented for interrupt and noninterrupt methods of data transfer are highly simplified in order to present an overall view without the involvement of software operating systems and device drivers. For more detailed information, refer to the documentation supplied with the appropriate software system or I/O subsystem.

4-8. INPUT DATA TRANSFER (INTERRUPT METHOD)

Figure 4-5 illustrates the sequence of events required to input data using the interrupt method. Note that some operations are under control of the computer program (programmer's responsibility) and some of the operations are automatic. Note also that the interface PCA (device controller) is installed in the slot assigned to select code 12.

The operations begins (1) with the programmed instruction STC 12,C which sets the Control flip-flop and clears the Flag flip-flop on the interface PCA. Since the next few operations are under control of the hardware, the computer program may continue the execution of other instructions. Setting the Control flip-flop causes the PCA to output a Start signal (2) to the device, which reads out a data character and asserts the Done signal (3).

The device Done signal sets the PCA Flag flip-flop, which in turn generates an interrupt (4) assuming that the interrupt conditions are met; i.e., the interrupt system must be on (STF 00 previously given), no higher priority interrupt is pending, and the Control flip-flop is set (done in step 1).

The interrupt causes the current computer program to be suspended and control is transferred to a service subroutine (5). It is the programmer's responsibility to provide the linkage between the interrupt location (00012 in this case) and the service subroutine. It is also the programmer's responsibility to include in his service subroutine the instructions for processing the data (loading into an accumulator, manipulating if necessary, and storing into memory).

The subroutine may then issue further STC 12,C commands to transfer additional data characters. One of the final instructions in the service subroutine must be CLC 12. This step (6) restores the interrupt capability to lower priority devices and returns the interface PCA to its static "ready" condition (Control clear and Flag set). This condition is initially established by the computer at power turn-on and it is the programmer's responsibility to return the interface PCA to the same condition on the completion of each data transfer operation. At the end of the subroutine, control is returned to the interrupted program via previously established linkages.

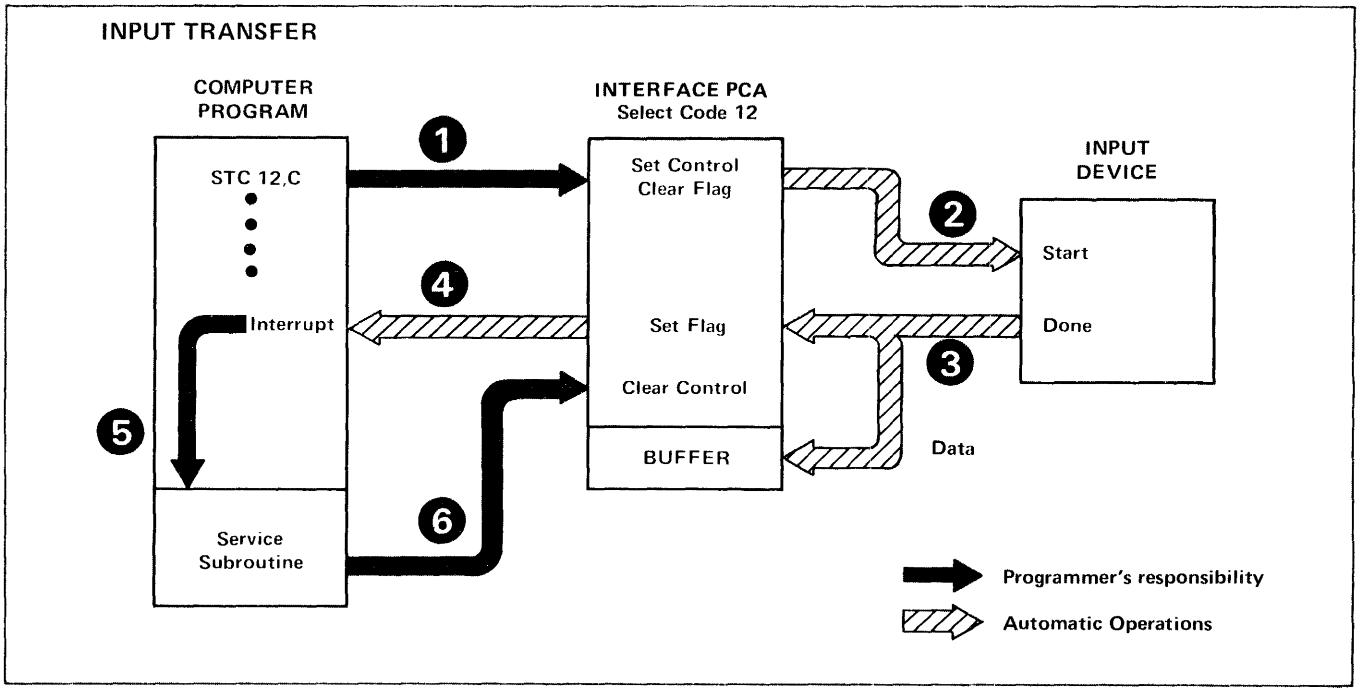


Figure 4-5. Input Data Transfer (Interrupt Method)

4-9. OUTPUT DATA TRANSFER (INTERRUPT METHOD)

Figure 4-6 illustrates the sequence of events required to output data using the interrupt method. Again note the distinction between programmed and automatic instructions. It is assumed that the data to be transferred has been loaded into the A-register and is in a form suitable for output. The interface PCA in this example is assumed to be in the slot assigned to select code 13.

The output operation begins with a programmed instruction (OTA 13) to transfer the contents of the A-register to the interface PCA buffer (1). This is followed (2) by the instruction STC 13,C which sets the Control flip-flop and clears the Flag flip-flop on the interface PCA. Since the next few operations are under control of the hardware, the computer program may continue the execution of other instructions. Setting the Control flip-flop causes the PCA to output the buffered data and a Start signal (3) to the device, which writes (e.g., punches, stores, etc.) the data character and asserts the Done signal (4).

The device Done signal sets the PCA Flag flip-flop, which in turn generates an interrupt (5) provided that the interrupt system is on, priority is high, and the Control flip-flop is set (done in step 2). The interrupt causes the current computer program to be suspended, and control is transferred to a service subroutine (6). It is the programmer's responsibility to provide the linkage between the interrupt location (00013 in this case) and the service subroutine. The detailed contents of the subroutine are also the programmer's responsibility, and the contents will vary with the type of device. The subroutine may then output further data to the interface PCA and reissue the STC 13,C command for additional data character transfers. One of the final instructions in the service subroutine must be a clear control (CLC 13). This step (7) allows lower priority devices to interrupt, and restores the channel to its static "ready" condition (Control clear and Flag set). At the end of the subroutine, control is returned to the interrupted program via previously established linkages.

4-10. NONINTERRUPT DATA TRANSFER

It is also possible to transfer data without using the interrupt system. This involves a "wait-for-flag" method in which the computer commands the device to operate and then waits for the completion response. In using this method to transfer data, it is assumed that the computer time is relatively unimportant. The programming is very simple, consisting of only four words of in-line coding as shown in table 4-1. Each of these routines will transfer one word or character of data. It is also assumed that the interrupt system is turned off (STF 00 not previously given).

4-11. INPUT. As described under paragraph 4-8, an STC 12,C instruction begins the operation by commanding the device to read one word or character. The computer then goes into a waiting loop, repeatedly checking the status of the flag bit. If the Flag flip-flop is not set, the JMP *-1 instruction causes a jump back to the SFS instruction. (The *-1 operand is assembler notation for "this location minus one.") When the Flag flip-flop is set, the skip condition for SFS is met and the JMP instruction is skipped. The computer thus exits from the waiting loop

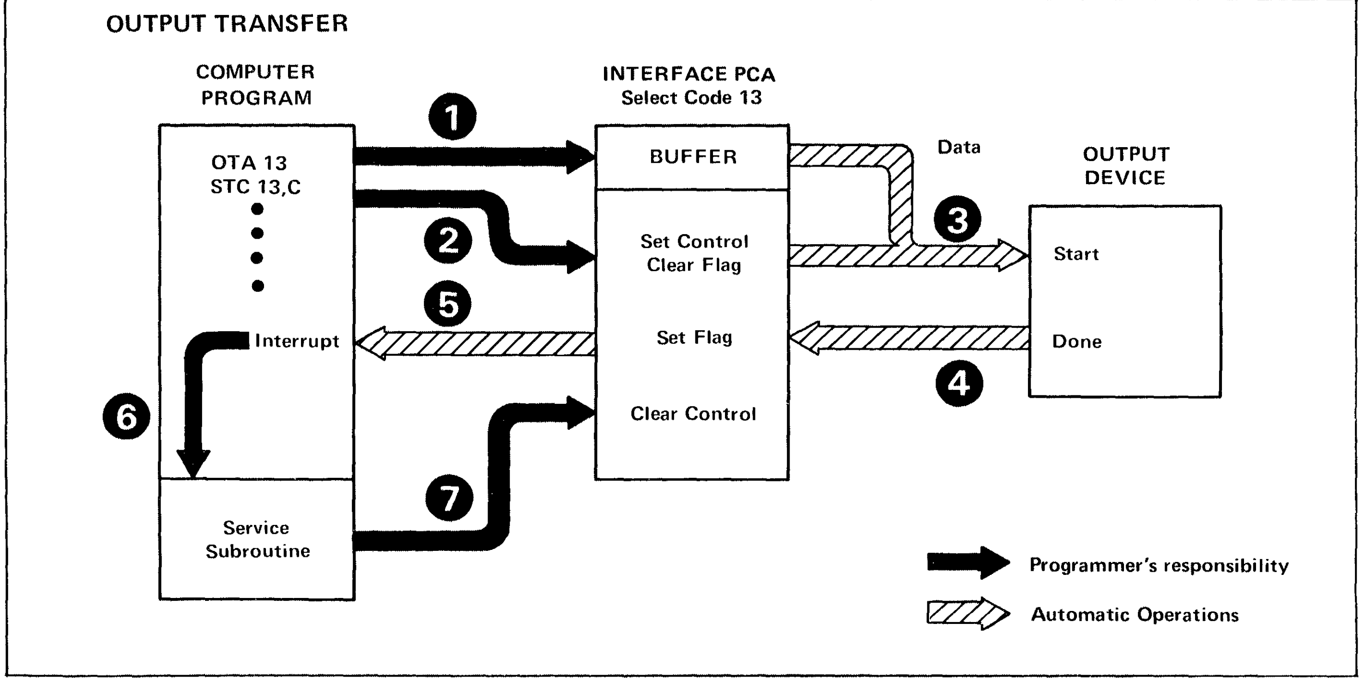


Figure 4-6. Output Data Transfer (Interrupt Method)

and the LIA 12 instruction loads the device input data into the A-register.

Table 4-1. Noninterrupt Transfer Routines

INPUT

INSTRUCTIONS	COMMENTS
STC 12,C	Start device
SFS 12	Is input ready?
JMP *-1	No, repeat previous instruction
LIA 12	Yes, load input into A-register

OUTPUT

INSTRUCTIONS	COMMENTS
OTA 13	Output A-register to buffer
STC 13,C	Start device
SFS 13	Has device accepted the data?
JMP *-1	No, repeat previous instruction
NOP	Yes, proceed

4-12. **OUTPUT.** The first step, which is to transfer the data to the interface PCA buffer, is the OTA 13 instruction. Then STC 13,C commands the device to operate and accept the data. The computer then goes into a waiting loop as described in the preceding paragraph. When the Flag flip-flop becomes set, indicating that the device has accepted the output data, the computer exits from the loop. (The final NOP is for illustration purposes only.)

4-13. DUAL-CHANNEL PORT CONTROLLER

The optional Dual-Channel Port Controller (DCPC) provides a direct data path, software assignable, between memory and a high-speed peripheral device; the DCPC accomplishes this by stealing an I/O cycle instead of interrupting to a service subroutine. The DCPC logic is capable of stealing every consecutive I/O cycle and can therefore transfer data at rates up to 616,666 words per second.

There are two DCPC channels, each of which may be separately assigned to operate with any I/O interface PCA, including those installed in the optional HP 12979A Input/Output Extender (assuming that the I/O extender DCPC option is installed). When both DCPC channels are operating simultaneously, channel 1 has priority over channel 2. The combined maximum transfer rate for both channels operating together is 616,666 words per second; the rate available to channel 2 is therefore the rate difference between 616,666 and the actual operating rate of channel 1. Since the memory cycle rate is somewhat faster than the I/O cycle rate, it is possible for the CPU to interleave memory cycles while the DCPC is operating at full bandwidth.

Transfers via the DCPC are on a full-word basis; hardware packing and unpacking of bytes are not provided. The word count register is a full 16 bits in length, and data transfers are accomplished in blocks. The transfer is initiated by an initialization routine, and from then on the operation is under automatic control of the hardware. The initialization routine specifies the direction of the data transfer (in or out), where in memory to read or write, which I/O channel to use, and how much data to transfer. Completion of the block transfer is signalled by an interrupt to location 00006 (for channel 1) or to location 00007 (for channel 2) if the interrupt system is enabled. It is also possible to check for completion by testing the status of the flag for select code 06 or 07, or by interrogating the word count register with an LIA/B to select code 02 (for channel 1) or to select code 03 (for channel 2). A block transfer in process can be aborted with an STF 06 or 07 instruction.

4-14. **DCPC OPERATION.** Figure 4-7 illustrates the sequence of operations for a DCPC input data transfer. A comparison with the conventional interrupt method (figure 4-5) shows that much more of the DCPC operation is automatic. Remember that the procedure in figure 4-5 must be repeated for each word or character. In figure 4-7, the automatic DCPC operation will transfer a block of data of any size limited only by the available memory space. The sequence of events is as follows. (An input data transfer is illustrated; the minor differences for an output transfer are explained in text.)

The initialization routine sets up the control registers on the DCPC (1) and issues the first start command (STC 12,C) directly to the interface PCA. (If the operation is an output, the interface PCA buffer is also loaded at this time.) The DCPC logic is now turned on and the computer program continues with other instructions.

Setting the Control and clearing the Flag flip-flops (2) causes the interface PCA to send a Start signal (with a data word if it is an output transfer) to the external device (3). The device goes through a read or write cycle and returns a Done signal (with a data word if it is an input transfer). The Done signal (4) sets the PCA Flag flip-flop which, regardless of priority, immediately requests the DCPC logic to steal an I/O cycle (5) and transfer a word into (or out of) memory. The process now repeats back to the beginning of this paragraph to transfer the next word.

After the specified number of words have been transferred, the interface PCA Control flip-flop is cleared (7) and the DCPC logic generates a completion interrupt (8). The program control is now forced to a completion routine (9), the contents of which is the programmer's responsibility.

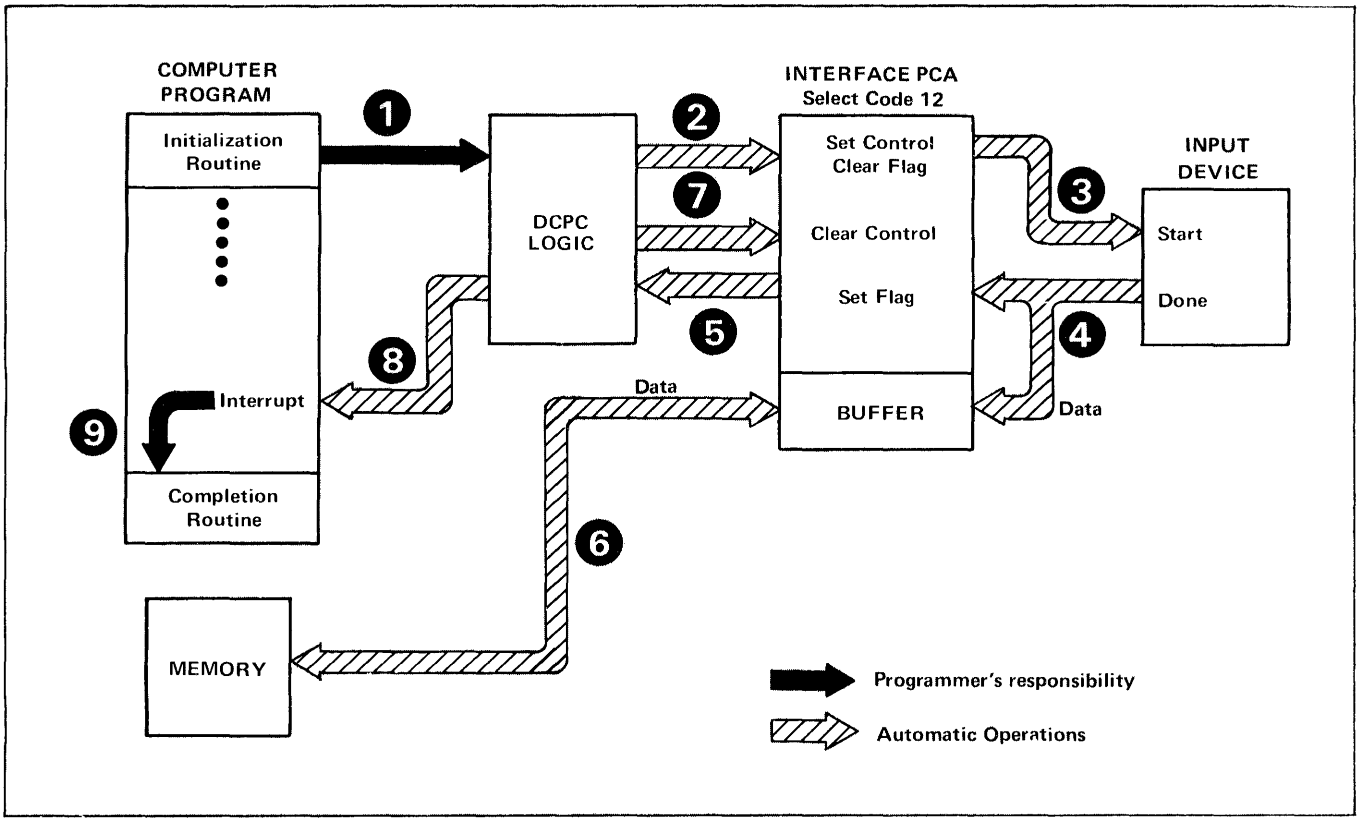


Figure 4-7. DCPC Input Data Transfer

4-15. **DCPC INITIALIZATION.** The information required to initialize the DCPC (direction, memory allocation, I/O channel assignment, and block length) are given by three control words. These three words must be addressed specifically to the DCPC. Figure 4-8 illustrates the format of the three control words. Control Word 1 (CW1) identifies the I/O channel to be used and provides two options selectable by the programmer:

Bit 15
1 = give STC (in addition to CLF) to I/O channel at end of each DCPC cycle (except on last cycle, if input)

0 = no STC

Bit 13
1 = give CLC to I/O channel at end of block transfer

0 = no CLC

Control Word 2 (CW2) gives the starting memory address for the block transfer and bit 15 determines whether data is to go into memory (logic 1) or out of memory (logic 0). Control Word 3 (CW3) is the two's complement of the number of words to be transferred into or out of memory (i.e., the block length). This number can be from 1 to 32,768, although it is limited in the practical case by available memory.

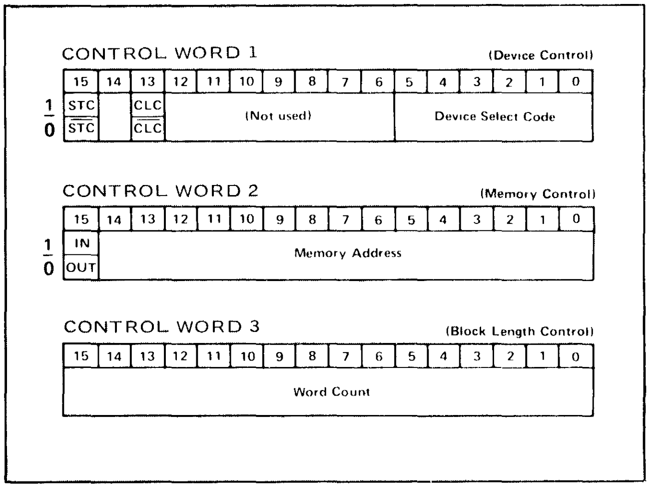


Figure 4-8. DCPC Control Word Formats

Table 4-2 gives the basic program sequence for outputting the control words to the DCPC. As shown in this table, CLC 2 and STC 2 perform switching functions to prepare the logic for either CW2 or CW3. The device is assumed to be in I/O slot channel 10, and it is also assumed that its start command is STC 10B, C. The sample values of CW1, CW2, and CW3 will read a block of 50 words and store these in locations 200 through 261 (octal). The STC 06B,C instruction starts the DCPC operation. A flag-status method of detecting the end-of-transfer is used in this example; an interrupt to location 00006 could be substituted for this test. The program in table 4-2 could easily be changed to operate on channel 2 by changing select codes 2 to 3 and 6 to 7.

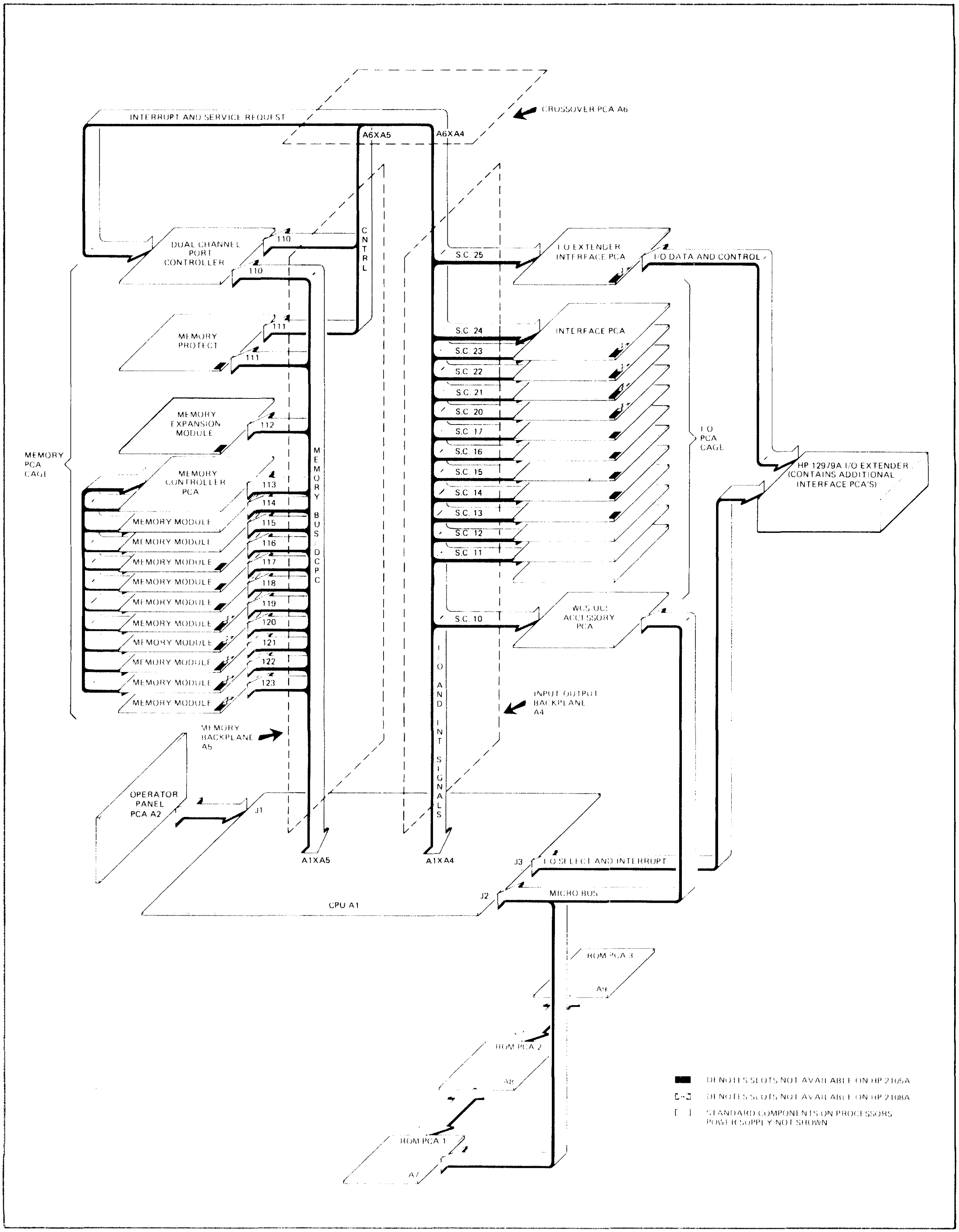
One important difference should be noted when doing a DCPC input operation from a disc or a drum. Due to the synchronous nature of disc or drum memories and the design of the interface PCA, the order of starting must be reversed from the order given; i.e., start the DCPC first and then start the disc (or drum).

Table 4-2. DCPC Initialization Program

LABEL	OPCODE	OPERAND	COMMENTS
ASGN1	LDA	CW1	Fetches control word 1 (CW1) from memory and loads it in A-register.
	OTA	6B	Outputs CW1 to DCPC Channel 1.
MAR1	CLC	2B	Prepares Memory Address Register to receive control word 2 (CW2).
	LDA	CW2	Fetches CW2 from memory and loads it in A-register.
	OTA	2B	Outputs CW2 to DCPC Channel 1.
WCR1	STC	2B	Prepares Word Count Register to receive control word 3 (CW3).
	LDA	CW3	Fetches CW3 from memory and loads it in A-register.
	OTA	2B	Outputs CW3 to DCPC Channel 1.
STRT1	STC	10B,C	Start input device.
	STC	6B,C	Activate DCPC Channel 1.
	SFS JMP	6B *-1	Wait while data transfer takes place or, if interrupt processing is used, continue program.
⋮	⋮	⋮	
	HLT		Halt
CW1	OCT	120010	Assignment for DCPC Channel 1 (ASGN1); specifies I/O channel select code address (10_8), STC after each word is transferred, and CLC after final word is transferred.
CW2	OCT	100200	Memory Address Register control. DCPC Channel 1 (MAR1); specifies memory input operation and starting memory address (200_8).
CW3	DEC	-50	Word Count Register control. DCPC Channel 1 (WCR1); specifies the 2's complement of the number of character words in the block of data to be transferred (50_{10}).

APPENDIX

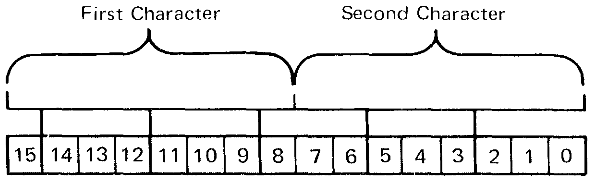
COMPUTER PHYSICAL LAYOUT



7021-24

CHARACTER CODES

ASCII Character	First Character Octal Equivalent	Second Character Octal Equivalent
A	040400	000101
B	041000	000102
C	041400	000103
D	042000	000104
E	042400	000105
F	043000	000106
G	043400	000107
H	044000	000110
I	044400	000111
J	045000	000112
K	045400	000113
L	046000	000114
M	046400	000115
N	047000	000116
O	047400	000117
P	050000	000120
Q	050400	000121
R	051000	000122
S	051400	000123
T	052000	000124
U	052400	000125
V	053000	000126
W	053400	000127
X	054000	000130
Y	054400	000131
Z	055000	000132
a	060400	000141
b	061000	000142
c	061400	000143
d	062000	000144
e	062400	000145
f	063000	000146
g	063400	000147
h	064000	000150
i	064400	000151
j	065000	000152
k	065400	000153
l	066000	000154
m	066400	000155
n	067000	000156
o	067400	000157
p	070000	000160
q	070400	000161
r	071000	000162
s	071400	000163
t	072000	000164
u	072400	000165
v	073000	000166
w	073400	000167
x	074000	000170
y	074400	000171
z	075000	000172
0	030000	000060
1	030400	000061
2	031000	000062
3	031400	000063
4	032000	000064
5	032400	000065
6	033000	000066
7	033400	000067
8	034000	000070
9	034400	000071
NUL	000000	000000
SOH	000400	000001
STX	001000	000002
ETX	001400	000003
EOT	002000	000004
ENQ	002400	000005
ACK	003000	000006
BEL	003400	000007
BS	004000	000010
HT	004400	000011
LF	005000	000012
VT	005400	000013
FF	006000	000014
CR	006400	000015
SO	007000	000016
SI	007400	000017
DLE	010000	000020
DC1	010400	000021
DC2	011000	000022
DC3	011400	000023
DC4	012000	000024
NAK	012400	000025
SYN	013000	000026
ETB	013400	000027
CAN	014000	000030
EM	014400	000031
SUB	015000	000032
ESC	015400	000033
FS	016000	000034
GS	016400	000035
RS	017000	000036
US	017400	000037
SPACE	020000	000040
!	020400	000041
"	021000	000042
#	021400	000043
$	022000	000044
%	022400	000045
&	023000	000046
'	023400	000047
(	024000	000050
)	024400	000051
*	025000	000052
+	025400	000053
,	026000	000054
–	026400	000055
.	027000	000056
/	027400	000057
:	035000	000072
;	035400	000073
<	036000	000074
=	036400	000075
>	037000	000076
?	037400	000077
@	040000	000100
[	055400	000133
\	056000	000134
]	056400	000135
Δ	057000	000136
—	057400	000137
`	060000	000140
{	075400	000173
|	076000	000174
}	076400	000175
~	077000	000176
DEL	077400	000177



OCTAL ARITHMETIC

ADDITION

TABLE

0	01	02	03	04	05	06	07
1	02	03	04	05	06	07	10
2	03	04	05	06	07	10	11
3	04	05	06	07	10	11	12
4	05	06	07	10	11	12	13
5	06	07	10	11	12	13	14
6	07	10	11	12	13	14	15
7	10	11	12	13	14	15	16

EXAMPLE

Add: 3677 octal
+ 1331 octal
(111-) carries
5230 octal

MULTIPLICATION

TABLE

1	02	03	04	05	06	07
2	04	06	10	12	14	16
3	06	11	14	17	22	25
4	10	14	20	24	30	34
5	12	17	24	31	36	43
6	14	22	30	36	44	52
7	16	25	34	43	52	61

EXAMPLE

Multiply: 657 octal
X 54 octal
3274
4153
45024 octal

(Reminder: add in octal)

COMPLEMENT

To find the two's complement form of an octal number. (Same procedure whether converting from positive to negative or negative to positive.)

RULE

1. Subtract from the maximum representable octal value.
2. Add one.

EXAMPLE

Two's complement of 556_8 :

177777
- 000556
177221
+ 1
177222_8

OCTAL/DECIMAL CONVERSIONS

OCTAL TO DECIMAL

TABLE

OCTAL	DECIMAL
0- 7	0- 7
10-17	8-15
20-27	16-23
30-37	24-31
40-47	32-39
50-57	40-47
60-67	48-55
70-77	56-63
100	64
200	128
400	256
1000	512
2000	1024
4000	2048
10000	4096
20000	8192
40000	16384
77777	32767

EXAMPLE

Convert 463_8 to a decimal integer.

$400_8 = 256_{10}$

$60_8 = 48_{10}$

$3_8 = \underline{3_{10}}$

307 decimal

DECIMAL TO OCTAL

TABLE

DECIMAL	OCTAL
1	1
10	12
20	24
40	50
100	144
200	310
500	764
1000	1750
2000	3720
5000	11610
10000	23420
20000	47040
32767	77777

EXAMPLE

Convert 5229_{10} to an octal integer.

$5000_{10} = 11610_8$

$200_{10} = 310_8$

$20_{10} = 24_8$

$9_{10} = \underline{11_8}$

12155_8

↑

(Reminder: add in octal)

NEGATIVE DECIMAL TO TWO'S COMPLEMENT OCTAL

TABLE

DECIMAL	2's COMP
-1	177777
-10	177766
-20	177754
-40	177730
-100	77634
-200	177470
-500	177014
-1000	176030
-2000	174040
-5000	166170
-10000	154360
-20000	130740
-32768	100000

EXAMPLE

Convert -629_{10} to two's complement octal.

$-500_{10} = 177014_8$

$-100_{10} = 177634_8$

$-20_{10} = 177754_8$ (Add in octal)

$-9_{10} = \underline{177767_8}$

176613_8

For reverse conversion (two's complement octal to negative decimal):

1. Complement, using procedure on facing page.
2. Convert to decimal, using OCTAL TO DECIMAL table.

MATHEMATICAL EQUIVALENTS

$2^{\pm n}$ IN DECIMAL

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.01562 5
128	7	0.00781 25
256	8	0.00390 625
512	9	0.00195 3125
1 024	10	0.00097 65625
2 048	11	0.00048 82812 5
4 096	12	0.00024 41406 25
8 192	13	0.00012 20703 125
16 384	14	0.00006 10351 5625
32 768	15	0.00003 05175 78125
65 536	16	0.00001 52587 89062 5
131 072	17	0.00000 76293 94531 25
262 144	18	0.00000 38146 97265 625
524 288	19	0.00000 19073 48632 8125
1 048 576	20	0.00000 09536 74316 40625
2 097 152	21	0.00000 04768 37158 20312 5
4 194 304	22	0.00000 02384 18579 10156 25
8 388 608	23	0.00000 01192 09289 55078 125
16 777 216	24	0.00000 00596 04644 77539 0625
33 554 432	25	0.00000 00298 02322 38769 53125
67 108 864	26	0.00000 00149 01161 19384 76562 5
134 217 728	27	0.00000 00074 50580 59692 38281 25
268 435 456	28	0.00000 00037 25290 29846 19140 625
536 870 912	29	0.00000 00018 62645 14923 09570 3125
1 073 741 824	30	0.00000 00009 31322 57461 54785 15625
2 147 483 648	31	0.00000 00004 65661 28730 77392 57812 5
4 294 967 296	32	0.00000 00002 32830 64365 38696 28906 25

$10^{\pm n}$ IN OCTAL

10^n	n	10^{-n}
1	0	1.000 000 000 000 000 000 00
12	1	0.063 146 314 631 463 146 31
144	2	0.005 075 341 217 270 243 66
1 750	3	0.000 406 111 564 570 651 77
23 420	4	0.000 032 155 613 530 704 15
303 240	5	0.000 002 476 132 610 706 64
3 641 100	6	0.000 000 206 157 364 055 37
46 113 200	7	0.000 000 015 327 745 152 75
575 360 400	8	0.000 000 001 257 143 561 06
7 346 545 000	9	0.000 000 000 104 560 276 41
112 402 762 000	10	0.000 000 000 006 676 337 66
1 351 035 564 000	11	0.000 000 000 000 537 657 77
16 432 451 210 000	12	0.000 000 000 000 043 136 32
221 411 634 520 000	13	0.000 000 000 000 003 411 35
2 657 142 036 440 000	14	0.000 000 000 000 000 264 11
34 327 724 461 500 000	15	0.000 000 000 000 000 022 01
434 157 115 760 200 000	16	0.000 000 000 000 000 001 63
5 432 127 413 542 400 000	17	0.000 000 000 000 000 000 14
67 405 553 164 731 000 000	18	0.000 000 000 000 000 000 01

MATHEMATICAL EQUIVALENTS

2^x IN DECIMAL

x	2^x	x	2^x	x	2^x
0.001	1.00069 33874 62581	0.01	1.00695 55500 56719	0.1	1.07177 34625 36293
0.002	1.00138 72557 11335	0.02	1.01395 94797 90029	0.2	1.14869 83549 97035
0.003	1.00208 16050 79633	0.03	1.02101 21257 07193	0.3	1.23114 44133 44916
0.004	1.00277 64359 01078	0.04	1.02811 38266 56067	0.4	1.31950 79107 72894
0.005	1.00347 17485 09503	0.05	1.03526 49238 41377	0.5	1.41421 35623 73095
0.006	1.00416 75432 38973	0.06	1.04246 57608 41121	0.6	1.51571 65665 10398
0.007	1.00486 38204 23785	0.07	1.04971 66836 23067	0.7	1.62450 47927 12471
0.008	1.00556 05803 98468	0.08	1.05701 80405 61380	0.8	1.74110 11265 92248
0.009	1.00625 78234 97782	0.09	1.06437 01824 53360	0.9	1.86606 59830 73615

$n \log_{10} 2$, $n \log_2 10$ IN DECIMAL

n	$n \log_{10} 2$	$n \log_2 10$	n	$n \log_{10} 2$	$n \log_2 10$
1	0.30102 99957	3.32192 80949	6	1.80617 99740	19.93156 85693
2	0.60205 99913	6.64385 61898	7	2.10720 99696	23.25349 66642
3	0.90308 99870	9.96578 42847	8	2.40823 99653	26.57542 47591
4	1.20411 99827	13.28771 23795	9	2.70926 99610	29.89735 28540
5	1.50514 99783	16.60964 04744	10	3.01029 99566	33.21928 09489

MATHEMATICAL CONSTANTS IN OCTAL SCALE

$\pi = (3.11037\ 552421)_{(8)}$

$\pi^{-1} = (0.24276\ 301556)_{(8)}$

$\sqrt{\pi} = (1.61337\ 611067)_{(8)}$

$\ln \pi = (1.11206\ 404435)_{(8)}$

$\log_2 \pi = (1.51544\ 163223)_{(8)}$

$\sqrt{10} = (3.12305\ 407267)_{(8)}$

$e = (2.55760\ 521305)_{(8)}$

$e^{-1} = (0.27426\ 530661)_{(8)}$

$\sqrt{e} = (1.51411\ 230704)_{(8)}$

$\log_{10} e = (0.33626\ 754251)_{(8)}$

$\log_2 e = (1.34252\ 166245)_{(8)}$

$\log_2 10 = (3.24464\ 741136)_{(8)}$

$\gamma = (0.44742\ 147707)_{(8)}$

$\ln \gamma = -(0.43127\ 233602)_{(8)}$

$\log_2 \gamma = -(0.62573\ 030645)_{(8)}$

$\sqrt{2} = (1.32404\ 746320)_{(8)}$

$\ln 2 = (0.54271\ 027760)_{(8)}$

$\ln 10 = (2.23273\ 067355)_{(8)}$

OCTAL COMBINING TABLES

MEMORY REFERENCE INSTRUCTIONS

Indirect Addressing

Refer to octal instruction codes given on the following page.
To combine code for indirect addressing, merge "100000" with octal instruction code.

REGISTER REFERENCE INSTRUCTIONS

Shift-Rotate Group (SRG)

1. select to operate on A or B
2. select 1 to 4 micros, not more than one from each column.
3. combine octal codes (leading zeros omitted) by inclusive or.
4. order of execution is from column 1 to column 4.

A Operations

1	2	3	4
ALS (1000)	CLE (40)	SLA (10)	ALS (20)
ARS (1100)			ARS (21)
RAL (1200)			RAL (22)
RAR (1300)			RAR (23)
ALR (1400)			ALR (24)
ERA (1500)			ERA (25)
ELA (1600)			ELA (26)
ALF (1700)			ALF (27)

B Operations

1	2	3	4
BLS (5000)	CLE (4040)	SLB (4010)	BLS (4020)
BRS (5100)			BRS (4021)
RBL (5200)			RBL (4022)
RBR (5300)			RBR (4023)
BLR (5400)			BLR (4024)
ERB (5500)			ERB (4025)
ELB (5600)			ELB (4026)
BLF (5700)			BLF (4027)

Alter-Skip Group (ASG)

1. select to operate on A or B.
2. select 1 to 8 micros, not more than one from each column.
3. combine octal codes (leading zeros omitted) by inclusive or.
4. order of execution is from column 1 to column 8.

A Operations

1	2	3	4
CLA (2400)	SEZ (2040)	CLE (2100)	SSA (2020)
CMA (3000)		CME (2200)	
CCA (3400)		CCE (2300)	

5	6	7	8
SLA (2010)	INA (6040)	CLE (6100)	RSS (2001)

B Operations

1	2	3	4
CLB (6400)	SEZ (6040)	CLE (6100)	SSB (6020)
CMB (7000)		CME (6200)	
CCB (7400)		CCE (6300)	

5	6	7	8
SLB (6010)	INB (6004)	SZB (6002)	RSS (6001)

INPUT/OUTPUT INSTRUCTIONS

Clear Flag

Refer to octal instruction codes given on the following page.
To clear flag after execution (instead of holding flag), merge "001000" with octal instruction code.

INSTRUCTION CODES IN OCTAL

Memory Reference

AND	01 (0XX)---
XOR	02 (0XX)---
IOR	03 (0XX)---
JSB	01 (1XX)---
JMP	02 (1XX)---
ISZ	03 (1XX)---
ADA	04 (0XX)---
ADB	04 (1XX)---
CPA	05 (0XX)---
CPB	05 (1XX)---
LDA	06 (0XX)---
LDB	06 (1XX)---
STA	07 (0XX)---
STB	07 (1XX)---

↑ Binary

Shift-Rotate

NOP	000000
CLE	000040
SLA	000010
SLB	004010
ALS	001000
BLS	005000
ARS	001100
BRS	005100
RAL	001200
RBL	005200
RAR	001300
RBR	005300
ALR	001400
BLR	005400
ERA	001500
ERB	005500
ELA	001600
ELB	005600
ALF	001700
BLF	005700

Alter-Skip

CLA	002400
CLB	006400
CMA	003000
CMB	007000
CCA	003400
CCB	007400
SEZ	002040
CLE	002100
CME	002200
CCE	002300
SSA	002020
SSB	006020
SLA	002010
SLB	006010
INA	002004
INB	006004
SZA	002002
SZB	006002
RSS	002001

Input/Output

HLT	1020 --
STF	1021 --
CLF	1031 --
SFC	1022 --
SFS	1023 --
MIA	1024 --
MIB	1064 --
LIA	1025 --
LIB	1065 --
OTA	1026 --
OTB	1066 --
STC	1027 --
CLC	1067 --
STO	102101
CLO	103101
SOC	102201
SOS	102301

Extended Arithmetic

MPY	100200
DIV	100400
DLD	104200
DST	104400
ASR	1010 (01X)-
ASL	1000 (01X)-
LSR	1010 (10X)-
LSL	1000 (10X)-
RRR	1011 (00X)-
RRL	1001 (00X)-

↑ Binary

Ext. Inst. Group

ADX	105746
ADY	105756
CAX	101741
CAY	101751
CBS	105774
CBT	105766
CBX	105741
CBY	105751
CMW	105776
CXA	101744
CXB	105744
CYA	101754
CYB	105754
DSX	105761
DSY	105771
ISX	105760
ISY	105770
JLY	105762
JPY	105772
LAX	101742
LAY	101752
LBT	105763
LBX	105742
LBY	105752
LDX	105745
LDY	105755
MBT	105765
MVW	105777
SAX	101740
SAY	101750
SBS	105773
SBT	105764
SBX	105740
SBY	105750
SFB	105767
STX	105743
STY	105753
TBS	105775
XAX	101747
XAY	101757
XBX	105747
XBY	105757

Floating Point

FAD	105000
FDV	105060
FIX	105100
FLT	105120
FMP	105040
FSB	105020

Memory Expansion

DJP	105732
DJS	105733
JRS	105715
LFA	101727
LFB	105727
MBF	105703
MBI	105702
MBW	105704
MWF	105706
MWI	105705
MWW	105707
PAA	101712
PAB	105712
PBA	101713
PBB	105713
RSA	101730
RSB	105730
RVA	101731
RVB	105731
SJP	105734
SJS	105735
SSM	105714
SYA	101710
SYB	105710
UJP	105736
UJS	105737
USA	101711
USB	105711
XCA	101726
XCB	105726
XLA	101724
XLB	105724
XMA	101722
XMB	105722
XMM	105720
XMS	105721
XSA	101725
XSB	105725

*Assuming: no indirect addressing
no combined microinstructions
shifts taken in first position only
hold flag after I/O execution

Refer to preceding page for octal combining tables

BASE SET INSTRUCTION CODES IN BINARY

15	14 13 12	11 10 9	8 7 6	5 4 3	2 1 0
D/I	AND 001	0 Z/C	Memory Address		
D/I	XOR 010	0 Z/C			
D/I	IOR 011	0 Z/C			
D/I	JSB 001	1 Z/C			
D/I	JMP 010	1 Z/C			
D/I	ISZ 011	1 Z/C			
D/I	AD* 100	A/B Z/C			
D/I	CP* 101	A/B Z/C			
D/I	LD* 110	A/B Z/C			
D/I	ST* 111	A/B Z/C			
15	**14 13 12**	**11 10 9**	**8 7 6**	**5 4 3**	**2 1 0**
0	SRG 000	A/B 0 D/E	*LS 000	†CLE D/E ‡SL*	*LS 000
		A/B 0 D/E	*RS 001	D/E	*RS 001
		A/B 0 D/E	R*L 010	D/E	R*L 010
		A/B 0 D/E	R*R 011	D/E	R*R 011
		A/B 0 D/E	*LR 100	D/E	*LR 100
		A/B 0 D/E	ER* 101	D/E	ER* 101
		A/B 0 D/E	EL* 110	D/E	EL* 110
		A/B 0 D/E	*LF 111	D/E	*LF 111
		NOP 000	000	000	000
15	**14 13 12**	**11 10 9**	**8 7 6**	**5 4 3**	**2 1 0**
0	ASG 000	A/B 1 CL*	01 CLE 01	SEZ SS* SL*	IN* SZ* RSS
		A/B CM*	10 CME 10		
		A/B CC*	11 CCE 11		
15	**14 13 12**	**11 10 9**	**8 7 6**	**5 4 3**	**2 1 0**
1	IOG 000	1 H/C	HLT 000	Select Code	
		1 0	STF 001		
		1 1	CLF 001		
		1 0	SFC 010		
		1 0	SFS 011		
		A/B 1 H/C	MI* 100		
		A/B 1 H/C	LI* 101		
		A/B 1 H/C	OT* 110		
		0 1 H/C	STC 111		
		1 1 H/C	CLC 111		
		1 0	STO 001	000	001
		1 1	CLO 001	000	001
		1 H/C	SOC 010	000	001
		1 H/C	SOS 011	000	001
15	**14 13 12**	**11 10 9**	**8 7 6**	**5 4 3**	**2 1 0**
1	EAG 000	MPY** 000	010	000	000
		DIV** 000	100	000	000
		DLD** 100	010	000	000
		DST** 100	100	000	000
		ASR 001	000	0 1	
		ASL 000	000	0 1	
		LSR 001	000	1 0	number of bits
		LSL 000	000	1 0	
		RRR 001	001	0 0	
		RRL 000	001	0 0	

Notes: * = A or B, according to bit 11.
D/I, A/B, Z/C, D/E, H/C coded: 0/1.
**Second word is Memory Address.

†CLE: Only this bit is required.
‡SL*: Only this bit and bit 11 (A/B as applicable) are required.

EXTENDED INSTRUCTION GROUP CODES IN BINARY

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAX/SAY/SBX/SBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	0	0
CAX/CAY/CBX/CBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	0	1
LAX/LAY/LBX/LBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	0	1	0
STX/STY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	0	1	1
CXA/CYA/CXB/CYB	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	1	0	0
LDX/LDY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	1	0	1
ADX/ADY	1	0	0	0	1	0	1	1	1	1	1	0	X/Y	1	1	0
XAX/XAY/XBX/XBY	1	0	0	0	A/B	0	1	1	1	1	1	0	X/Y	1	1	1
ISX/ISY/DSX/DSY	1	0	0	0	1	0	1	1	1	1	1	1	X/Y	0	0	I/D
JUMP INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	▨	0	1	0
BYTE INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	0	▨	▨	▨
BIT INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	1	▨	▨	▨
WORD INSTRUCTIONS	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	▨

JUMP INSTRUCTIONS:

JLY = 0
JPY = 1

BYTE INSTRUCTIONS:

LBT = 0 1 1
SBT = 1 0 0
MBT = 1 0 1
CBT = 1 1 0
SFB = 1 1 1

BIT INSTRUCTIONS:

SBS = 0 1 1
CBS = 1 0 0
TBS = 1 0 1

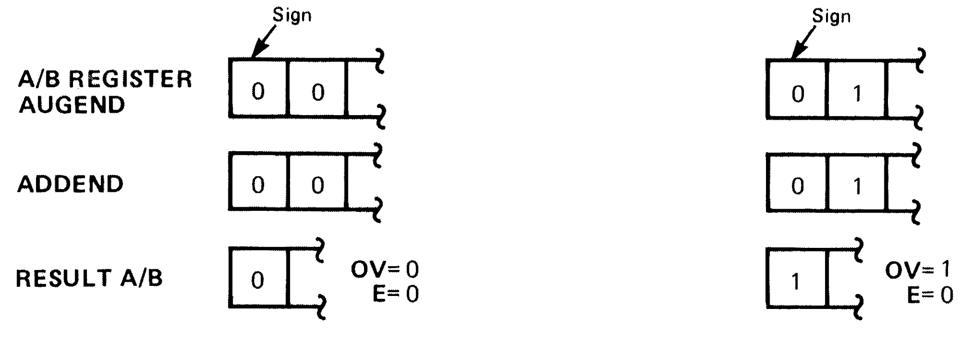
WORD INSTRUCTIONS:

CMW = 0
MVW = 1

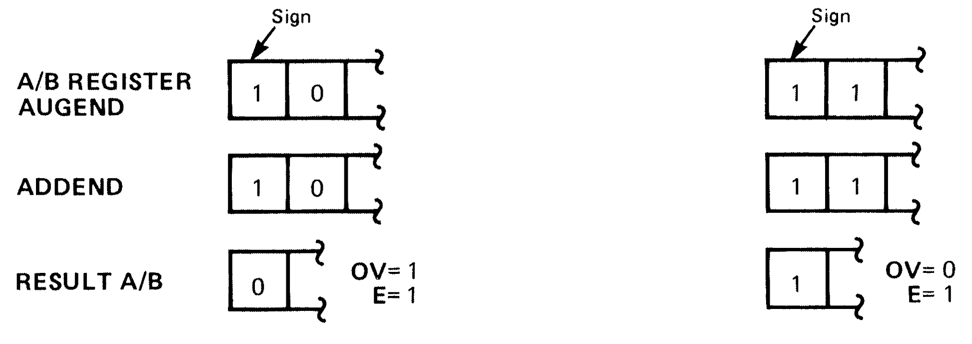
DYNAMIC MAPPING SYSTEM INSTRUCTION CODES IN BINARY

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DJP/DJS/UJP/UJS	1	0	0	0	1	0	1	1	1	1	0	1	1	D/U	1	P/S
JRS	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	1
LFA/LFB	1	0	0	0	A/B	0	1	1	1	1	0	1	0	1	1	1
MBI/MBF	1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	I/F
MBW	1	0	0	0	1	0	1	1	1	1	0	0	0	1	0	0
MWF	1	0	0	0	1	0	1	1	1	1	0	0	0	1	1	0
MWI/MWW	1	0	0	0	1	0	1	1	1	1	0	0	0	1	I/W	1
PAA/PAB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	1	0
PBA/PBB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	1	1
RSA/RSB/RVA/RVB	1	0	0	0	A/B	0	1	1	1	1	0	1	1	0	0	S/V
SJP/SJS	1	0	0	0	1	0	1	1	1	1	0	1	1	1	0	P/S
SSM	1	0	0	0	1	0	1	1	1	1	0	0	1	1	0	0
SYA/SYB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	0	0
USA/USB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	0	0	1
XCA/XCB/XLA/XLB	1	0	0	0	A/B	0	1	1	1	1	0	1	0	1	L/C	0
XMA/XMB	1	0	0	0	A/B	0	1	1	1	1	0	1	0	0	1	0
XMM/XMS	1	0	0	0	1	0	1	1	1	1	0	1	0	0	0	M/S
XSA/XSB	1	0	0	0	A/B	0	1	1	1	1	0	0	1	1	0	1

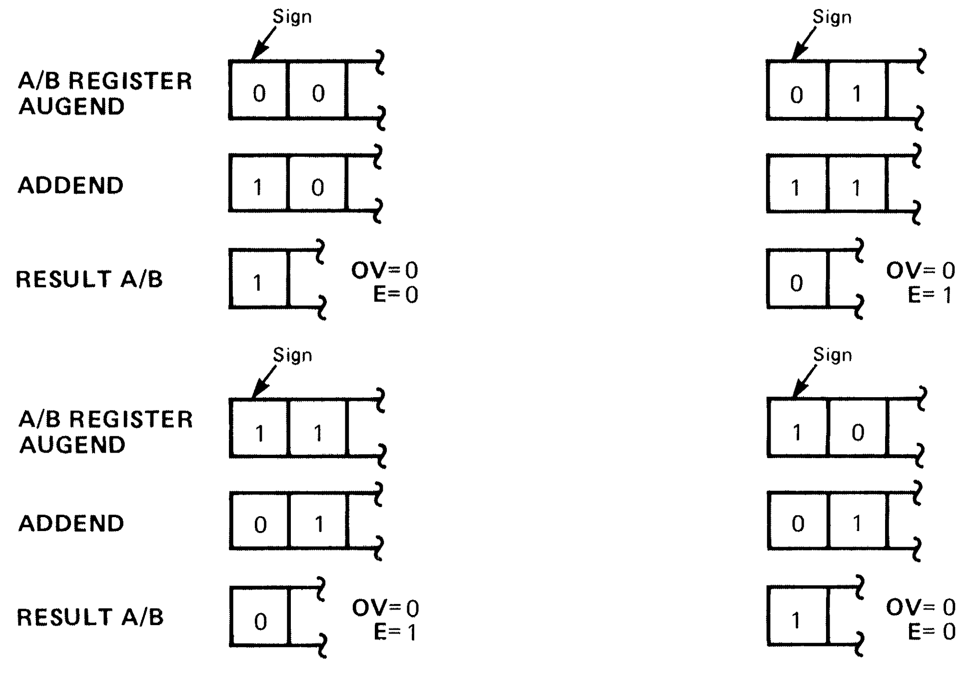
EXTEND AND OVERFLOW EXAMPLES



SAME SIGN (POSITIVE)



SAME SIGN (NEGATIVE)



DIFFERENT SIGNS

INTERRUPT AND I/O CONTROL SUMMARY

INST	S.C. 00	S.C. 01	S.C. 02	S.C. 03
STC	NOP	NOP	Prepares DCPC channel 1 to receive and store the block length in 2's complement form.	Prepares DCPC channel 2 to receive and store the block length in 2's complement form.
CLC	Clears all Control FF's from S.C. 06 and up; effectively turns off all I/O devices.	NOP	Prepares DCPC channel 1 to receive and store the direction of data flow and the starting memory address.	Prepares DCPC channel 2 to receive and store the direction of data flow and the starting memory address.
STF	Turns on interrupt system.	STO sets overflow bit.	NOP	NOP
CLF	Turns off interrupt system except power fail (S.C. 04) and parity error (S.C. 05).	CLO clears overflow bit.	NOP	NOP
SFS	Skip if interrupt system is on.	SOS	NOP	NOP
SFC	Skip if interrupt system is off.	SOC	NOP	NOP
LIA/B	Loads A/B register with all zeros. (Equivalent to CLA/B instruction.)	Loads display register contents into A/B register.	Loads present contents of DCPC channel 1 word count register into A/B register.	Loads present contents of DCPC channel 2 word count register into A/B register.
MIA/B	Equivalent to a NOP.	Merges display register contents into A/B register.	Merges present contents of DCPC channel 1 word count register into A/B register.	Merges present contents of DCPC channel 2 word count register into A/B register.
OTA/B	NOP	Outputs A/B register contents into display register.	1. Outputs to DCPC channel 1 the block length in 2's complement form (previously prepared by an STC 02 instruction). 2. Outputs to DCPC channel 1 the direction of data flow and the starting memory address (previously prepared by a CLC 02 instruction).	1. Outputs to DCPC channel 2 the block length in 2's complement form (previously prepared by an STC 03 instruction). 2. Outputs to DCPC channel 2 the direction of data flow and the starting memory address (previously prepared by a CLC 03 instruction).

S.C. 04	S.C. 05	S.C. 06	S.C. 07	S.C. 10-77
Re-initializes power-fail logic and restores interrupt capability to lower priority functions.	Turns on memory protect.	Sets Control FF on DCPC channel 1 (activates DMA).	Sets Control FF on DCPC channel 2 (activates DMA).	Sets PCA Control FF and turns on device on channel specified by S.C.
Re-initialize power-fail logic and restores interrupt capability to lower priority functions.	NOP	Clears Control FF on DCPC channel 1 (reestablishes priority with STF; does not turn off DCPC).	Clears Control FF on DCPC channel 2 (reestablishes priority with STF; does not turn off DCPC).	Clears PCA Control FF and turns off device.
Flag FF sets automatically when power comes up. (No program control possible.)	Turns on parity error interrupt capability.	1. Hardware Controlled: Sets Flag FF when word count is reached and turns off DCPC channel 1. 2. Program Controlled: Aborts data transfer.	1. Hardware Controlled: Sets Flag FF when word count is reached and turns off DCPC channel 2. 2. Program Controlled: Aborts data transfer.	Sets PCA Flag FF.
Flag FF clears automatically when power fail occurs. (No program control possible.)	Turns off parity error interrupt capability and clears violation register bit 15.	Clears Flag FF on DCPC channel 1.	Clears Flag FF on DCPC channel 2.	Clears PCA Flag FF.
NOP	Skip if Dynamic Mapping System (DMS) interrupt.	Tests if DCPC channel 1 data transfer is complete.	Skip if DCPC channel 2 data transfer is complete.	Skip if I/O channel Flag FF is set.
Skip if power fail has occurred.	Skip if memory protect interrupt.	Tests if DCPC channel 1 data transfer is still in progress.	Skip if DCPC channel 2 data transfer is still in progress.	Skip if I/O channel Flag FF is clear.
Loads contents of central interrupt register (S.C. of last interrupting device) into least-significant bits of A/B register.	Loads contents of violation register into A/B register: Bit 15 = 1 = PE Bit 15 = 0 = MPV	Loads A/B register with all ones. (Equivalent to CCA/CCB instruction.)	Loads A/B register with all ones. (Equivalent to CCA/CCB instruction.)	Loads contents of PCA data buffer into A/B register.
Merges contents of central interrupt register into least-significant bits of A/B register.	Merges contents of violation register into A/B register.	Same as LIA/B 06 above.	Same as LIA/B 07 above.	Merges contents of PCA data buffer into A/B register.
NOP	Outputs first address of unprotected memory to fence register.	Outputs to DCPC channel 1 the S.C. of I/O channel. Specify STC after each word; CLC after block.	Outputs to DCPC channel 2 the S.C. of I/O channel. Specify STC after each word; CLC after block.	Outputs data from A/B register into PCA data buffer.

HEWLETT **hp** *PACKARD*

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

AMERICAN SAMOA
Calculators Only
Oceanic Systems Inc.
P O. Box 777
Pago Pago Bayfront Road
Pago Pago 96799
Tel: 633-5513
Cable: OCEANIC-Pago Pago

ANGOLA
Telectra
Empresa Técnica de
Equipamentos
Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-I°DT.°
Caixa Postal, 6487
Luanda
Tel: 35515/6
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia
Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
P.O. Box 36
Doncaster East, Victoria 3109
Tel: 89-6351
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia
Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia
Pty. Ltd.
153 Greenhill Road
Parkside, 5063, S.A.
Tel: 272-5911
Telex: 82536 ADEL
Cable: HEWPARD ADELAIDE

Hewlett-Packard Australia
Pty. Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex: 93859 PERTH
Cable: HEWPARD PERTH

Hewlett-Packard Australia
Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel: 95-3733
Telex: 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia
Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 29-1544
Telex: 42133 BRISBANE

GUAM
Medical/Pocket Calculators Only
Guam Medical Supply, Inc.
Jay Ease Building, Room 210
P.O. Box 8947
Tamuning 96911
Tel: 646-4513
Cable: EARMED Guam

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connalight Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: H-255291-5
Telex: 74766 SCHMC HX
Cable: SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd.
Bombay 400 020
Tel: 29 50 21
Telex: 2156
Cable: BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 3751
Cable: BLUESTAR

Blue Star Ltd.
14/40 Civil Lines
Kanpur 208 001
Tel: 6 88 82
Telex: 292
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 7655
Cable: BLUESTAR

Blue Star Ltd.
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel: 634770 & 635166
Telex: 2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 430
Cable: BLUESTAR

Blue Star Ltd.
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016 Kerala
Tel: 32069, 32161, 32282
Telex: 046-514
Cable: BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex: 459

Blue Star Ltd.
2/34 Kodambakkan High Road
Madras 600034
Tel: 82056
Telex: 041-379
Cable: BLUESTAR

Blue Star Ltd.
Nathraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel: 7383
Cable: BLUESTAR
Telex: 240

INDONESIA
BERCA Indonesia P.T.
P.O. Box 496
1st Floor JL. Cikini Raya 61
Jakarta
Tel: 56038, 40369, 49886
Telex: 42895
Cable: BERCACON

BERCA Indonesia P.T.
63 JL. Raya Gubeng
Surabaya
Tel: 44309

ISRAEL
Electronics & Engineering Div.
of Motorola Israel Ltd.
16, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 03-389 73
Telex: 33569
Cable: BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd
Ohashi Building
1-59-1 Yoyogi
Shibuya-ku, **Tokyo**
Tel: 03-370-2281/92
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd.
Nissei Ibaraki Building
2-8 Kasuga 2-chrome, Ibaraki-shi
Osaka,567
Tel: (0726) 23-1641
Telex: 5332-385 YHP OSAKA

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya** . 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd
Tanigawa Building
2-24-1 Tsuruya-choo
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsui Building
105, 1-chrome, San-no-maru
Mito, Ibaragi 310
Tel:: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

Yokogawa-Hewlett-Packard Ltd.
Kimura Building
3rd Floor 20
2-chome, Tsukuba
Kumagaya, Saitama 360
Tel: 0485-24-6563

KENYA
Technical Engineering Services
(E.A.)Ltd.,
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable: PROTON

Medical Only
International Aeradio(E.A.)Ltd.,
P.O. Box 19012
Nairobi Airport
Nairobi
Tel: 336055/56
Telex: 22201/22301

KOREA
Samsung Electronics Co., Ltd.
20th Fl. Dongbang Bldg. 250,
2-KA , C.P.O. Box 2775
Taepyung-Ro, Chung-Ku
Seoul
Tel: (24) 2410-9
Telex: 22575

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya,**Selangor**
Tel: Kuala Lumpur-54994
Telex: MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 2400

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1° Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 Negon Mo

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
4-12 Cruickshank Street
Kilbirnie, Wellington 3
Mailing Address: Hewlett-Packard
(N.Z.) Ltd
P.O. Box 9443
Courtney Place
Wellington
Tel: 877-199
Telex: NZ 3839
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel 569-651
Telex: NZ 3839
Cable: HEWPACK.Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Telex: 2958 MEDISUP
Cable: DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St.
Wellington
Tel: 850-799
Telex: 3858
Cable: DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 309
239 Stanmore Road
Christchurch
Tel: 892-019
Cable: DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd
303 Great King Street
P.O. Box 233
Dunedin
Tel: 88-817
Cable: DENTAL, Dunedin

NIGERIA
The Electronics
Instrumentations Ltd
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable: THETEIL Ibadan

The Electronics Instrumentations Ltd.
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable: THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel: 511027, 512927
Telex: KR894
Cable: COOPERATOR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced Systems
Corporation
Filcapital Bldg.
11th Floor, Ayala Ave
Makati, **Rizal**
Tel: 85-34-91, 85-35-81
Telex: 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Salisbury
Tel: 705231 (5 lines)
Telex: RH 4122

SINGAPORE
Hewlett-Packard Singapore
(Pte.) Ltd
Blk. 2, 6th Floor, Jalan
Bukit Merah
Redhill Industrial Estate
Alexandra P.O. Box 58.
Singapore 3
Tel: 633022
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa
(Pty.). Ltd
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard House
Daphne Street, Wendywood.
Sandton, Transvaal 2144
Tel: 802-104016
Telex: SA43-4782JH
Cable: HEWPACK JOHANNESBURG

Hewlett-Packard South Africa
(Pty.). Ltd
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Center, Forest Drive,
Pinelands, Cape Province, 7405
Tel: 53-7955 thu 9
Telex: 57-0006

Hewlett-Packard South Africa
(Pty.). Ltd.
P.O. Box 37099
Overport, Durban 4067
641 Ridge Road, Durban
Durban, 4001
Tel: 88-7478/9
Telex: 6-7954
Cable: HEWPACK

TAIWAN
Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Shiao West Road
Sec. 1, 7th Floor
Taipei
Tel: 3819160-4 (5 Lines)
Telex: 21824 HEWPACK
Cable: HEWPACK TAIPEI

Hewlett-Packard Far East Ltd
Taiwan Branch
68-2, Chung Cheng 3rd Road
Kaohsiung
Tel (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co. Ltd.,
No. 20, yung Sui Road
Taipei, 100
Tel. 3715171-4 (5 lines)
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E.A.), Ltd
P.O. Box 861
Daressalaam
Tel 21251 Ext 265
Telex: 41030

THAILAND
UNIMESA Co. Ltd
Elcom Research Building
Bangjak Sukumvit Ave
Bangkok
Tel 932387, 930338
Cable: UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E.A.), Ltd.,
P.O. Box 2577
Kampala
Tel 54388
Cable INTAERIO Kampala

ZAMBIA
R.J. Tilbury (Zambia) Ltd
P.O. Box 2792
Lusaka
Tel 73793
Cable ARJAYTEE Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1267
Cable: HEWPACK Palo Alto
Telex 034-8300, 034-8493

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11620A - 168 Street
EdmontonT5M 3T9
Tel: (403) 452-3670
TWX: 610-831-2431 EDTH

Hewlett-Packard (Canada) Ltd.
915-42 Ave S.E. Suite 102
Calgary T2G 1Z1
Tel: (403) 287-1672
Twx; 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059 VCR

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H OL8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd
800 Windmill Road
P.O. Box 9331
Dartmouth B2Y 3Z6
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd
1785 Woodward Dr
Ottawa K2C 0P9
Tel: (613) 225-6530
TWX: 610-562-8968

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1M8
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd
275 Hymus Blvd
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada)
Ltd. in Mississauga

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina
S.A.
Av. Leandro N. Alem 822 - 12°
1001**Buenos Aires**
Tel: 31-6063,4,5,6 and 7
Telex: Public Booth N° 9
Cable: HEWPACK ARG

BOLIVIA
Stambuk & Mark (Bolivia) Ltda.
Av. Mariscal, Santa Cruz 1342
La Paz
Tel: 40626, 53163, 52421
Telex: 3560014
Cable: BUKMAR

BRAZIL
Hewlett-Packard do Brasil
I.E.C. Ltda.
Avenida Rio Negro, 980
Alphaville
06400 Barueria **Sao Paulo**
Tel: 429-2148/9;429-2118/9

Hewlett-Packard do Brasil
I.E.C. Ltda.
Rua Padre Chagas, 32
90000-**Pórto Alegre**-RS
Tel: (0512) 22-2998, 22-5621
Cable: HEWPACk pórto Alegre

Hewlett-Packard do Brasil
I.E.C. Ltda.
Rua Siqueira Campos, 53, 4°
andar-Copacabana
20000-**Rio de Janeiro**-GB
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK
Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Alameda 580-Of. 807
Casilla 2118
Santiago, 1
Tel: 398613
Telex: 3520001 CALMET
Cable: CALMET Santiago

Medical Only
General Machinery Co., Ltda.
Paraguay 494
Casilla 13910
Santiago
Tel: 31123, 31124
Cable: GEMCO Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA
Científica Costarricense S.A.
Calle Central, Avenidas 1 y 3
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Medical Only
A.F. Viscaíno Compañia Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 242-150,247-033/034
Cable: Astor Quito

Calculators Only
Computadoras y Equipos
Electrónicos
P.O. Box 2695
990 Toledo (y Cordero)
Quito
Tel: 525-982
Telex: 02-2113 Sagita Ed
Cable: Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento
Electronico de el Salvador
Bulevar de los Heroes 11-48
San Salvador
Tel: 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48,
Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro. Gu

MEXICO
Hewlett-Packard Mexicana,
S.A. de C.V.
Torres Adalid No. 21, 11 Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana,
S.A. de C.V.
Ave. Constitución No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-843

NICARAGUA
Roberto Terán G
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3431103 Curunda,
Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
División Aparatos y Equipos
Médicos
División Aparatos y Equipos
Científicos y de Investigación
P.O. Box 676
Chile-482, Edificio Victoria
Asuncion
Tel: 4-5069, 4-6272
Cable: RAMEL

PERU
Compañía Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 41-4325
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272, Urb. Country Club
Carolina 00639
Tel: (809) 762-7355/7455/7655
Telex: HPIC-PR 3450514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel 40-3102
Cable RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela
C.A.
Apartado 50933, Caracas 105
Edificio Segre
Tercera Transversal
Los Ruices Norte
Caracas 107
Tel 35-01-07 35-00-84,
35-00-65, 35-00-31
Telex 25146 HEWPACK
Cable HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave
Palo Alto, California 94304
Tel (415) 493-1501
TWX 910-373-1260
Cable HEWPACK Palo Alto
Telex 034-8300, 034-8493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges m b H
Handelskai 52
P O box 7
A-1205 **Vienna**
Tel (0222) 35 16 21 to 27
cable HEWPAK Vienna
Telex 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S A N V
Avenue de Col-Vert 1.
(Groenkraaglaan)
B-1170 **Brussels**
Tel (02) 672 22 40
Cable PALOBEN Brussels
Telex 23 494 paloben bru

CYPRUS
Kypronics
19. Gregorios & Xenopoulos Rd
P O Box 1152
CY-**Nicosia**
Tel 45628/29
Cable KYPRONICS PANDEHIS
Telex 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustavu v Bechovicich
CSSR-25097
Bechovice u Prahy
Tel 89 93 41
Telex 121333

DDR
Entwicklungslabor der TU Dresden
Forschungsinstitut Meinsberg
DDR-7305
Waldheim/Meinsberg
Tel 37 667
Telex 518741

Firma Forgber
Schlegelstrasse 15
1040 **Berlin**
Tel 28 27 411
Telex 112889

DENMARK
Hewlett-packard A.S
Datavej 52
DK 3460 **Birkerød**
Tel (02) 81 66 40
Cable HEWPACK AS
Telex 166 40 hpas

Hewlett-Packard A/S
Navervej 1
DK-8600 **Silkeborg**
Tel (06) 82 71 66
Telex 166 40 hpas
Cable HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P O Box 6
SF-00211 **Helsinki** 21
Tel 6923031
Cable HEWPACKOY Helsinki
Telex 12-1563

FRANCE
Hewlett Packard France
Quartier de Courtaboeuf
Boite Postale No 6
F 91401 **Orsay** Cédex
Tel (1) 907 78 25
Cable HEWPACK Orsay
Telex 600048

Hewlett Packard France
"Le Saquin"
Chemin des Mouilles
Boite Postale No 12
F-69130 **Ecully**
Tel (78) 33 81 25.
Cable HEWPACK Eculy
Telex 310617

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière. 20
F-31300 **Toulouse-Le Mirail**
Tel (61) 40 11 12
Cable HEWPACK 51957
Telex 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel (91) 89 12 36
Cable HEWPACK MARGN
Telex 410770

Hewlett-Packard France
Agence Régionale
63. Avenue de Rochester
Boîte Postale
F-35014 **Rennes** Cédex
Tel (99) 36 33 21
Cable HEWPACK 74912
Telex 740912

Hewlett-Packard France
Agence Régionale
74. Allée de la Robertsau
F-67000 **Strasbourg**
Tel (88) 35 23 20/21
Telex 890141
Cable HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel (20) 51 44 14
Telex 820744

GERMAN FEDERAL REPUBLIC

Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Postfach 560 140
D-6000 **Frankfurt** 56
Tel (0611) 50 04-1
Cable HEWPACKSA Frankfurt
Telex 04 13249 hpffmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**. Wurttemberg
Tel (07031) 667-1
Cable HEPAK Böblingen
Telex 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Dusseldorf
Emanuel-Leutze-Str 1 (Seestern)
D-4000 **Dusseldorf**
Tel (0211) 59 71-1
Telex 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel (040) 24 13 93
Cable HEWPACKSA Hamburg
Telex 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Am Grossmarkt 6
D-3000 **Hannover 91**
Tel (0511) 46 60 01
Telex 092 3259

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str 90
D-8500 **Nuremberg**
Tel (0911) 56 30 83/85
Telex 0623 860

Hewlett-Packard GmbH
Technisches Buero München
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel (089) 601 30 61-7
Cable HEWPACKSA München
Telex 0524985

Hewlett Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel (030) 24 90 86
Telex 18 3405 hpbln d

GREECE
Kostas Karayannis
18. Ermou Street
GR-**Athens** 126
Tel 3237731
Cable RAKAR Athens
Telex 21 59 62 rkar gr

Analytical Only
INTECO
G Papathanassiou & Co
Marni 17
GR - **Athens** 103
Tel 522 1915
Cable INTEKNIKA Athens
Telex 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd
52 Skoufa Street
GR - **Athens** 135
Tel 362 6972. 363 3830
Cable etalak athens
Telex 21-4693 ETAL GR

HUNGARY
MTA
Muszerugyi és Méréstechnikai
Szolgalata
Lenin Krt 67
1391 **Budapest** VI
Tel 42 03 38
Telex 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc
Hafnarhvoli - Tryggvatotu
IS-**Reykjavik**
Tel 1 58 20
Cable ELDING Reykjavik

IRAN
Hewlett Packard Iran Ltd
No 13. Fourteenth St
Miremad Avenue
P O Box 41-2419
IR-**Tehran**
Tel 851082-7
Telex 213405 HEWP IR

IRELAND
Hewlett-Packard Ltd
King Street Lane
GB-**Winnersh**, Wokingham
Berks. RG11 5AR
Tel (0734) 78 47 74
Telex 847178

ITALY
Hewlett Packard Italiana S p A
Casella postale 3645
I-20100 **Milano**
Tel (2) 6251 (10 lines)
Cable HEWPACKIT Milano
Telex 32046

Hewlett-Packard Italiana S p A
Via Pietro Maroncelli 40
(ang Via Visentini
I-35100 **Padova**
Tel (49) 66 48 88
Telex 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S p A
Via d Aghiardi, 7
I-56100 **Pisa**
Tel (050) 2 32 04
Telex 32046 via Milano

Hewlett-Packard Italiana S p A
Via G Armellini 10
I-00143 **Roma**
Tel (06) 54 69 61
Telex 61514
Cable HEWPACKIT Roma

Hewlett-Packard Italiana S p A
Via San Quintino. 46
I-10121 **Torino**
Tel (011) 52 82 64/54 84 68
Telex 32046 via Milano

Medical/Calculators Only
Hewlett-Packard Italiana S p A
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel (095) 37 05 04

Hewlett-Packard Italiana S p A
Via Amerigo Vespucci. 9
I-80142 **Napoli**
Tel (081) 33 77 11

Hewlett-Packard Italiana S p A
Via E Masi. 9/B
I-40137 **Bologna**
Tel (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co
P O Box 830
Kuwait
Tel 42 49 10
Cable VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S A N V
Avenue du Col-Vert 1
(Groenkraaglaan)
B-1170 **Brussels**
Tel (02) 672 22 40
Cable PALOBEN Brussels
Telex 23 494

MOROCCO
Gerep
190. Blvd Brahim Roudani
Casablanca
Tel 25-16-76/25-90-99
Cable Gerep-Casa
Telex 23739

NETHERLANDS
Hewlett-Packard Benelux N V
Van Heuven Goedhartlaan 121
P O Box 667
NL **Amstelveen** 1134
Tel (020) 47 20 21
Cable PALOBEN Amsterdam
Telex 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Haslum**
Tel (02) 53 83 60
Telex 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
U1 Stawki 2 6P
00-950**Warsaw**
Tel 39 67 43
Telex 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
U1 Krajowej Rady
Narodowej 51-55
00-800 **Warsaw**
Tel 20 62 21
Telex 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 **Lodz**
Tel 334-41. 337-83

PORTUGAL
Telectra-Empresa Tecnica de
Equipamentos Eléctricos S a r l
Rua Rodrigo da Fonseca 103
P O Box 2531
P-**Lisbon** 1
Tel (19) 68 60 72
Cable TELECTRA Lisbon
Telex 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S a r l
Av A A de Aguiar 138
P O Box 2761
P - **Lisbon**
Tel (19) 53 21 31 7
Cable INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
BD N Balcescu 16
Bucharest
Tel 158023/138885
Telex 10440

I I R U C
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel 12 64 30
Telex 01183716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str (Head office)
P O Box 1228
Jeddah
Tel 31173-332201
Cable ELECTRA

P O Box 2728 (Service center)
Riyadh
Tel 62596-66232
Cable RAOUFCO

SPAIN
Hewlett Packard Española. S A
Jerez No 3
E-**Madrid** 16
Tel (1) 458 26 00 (10 lines)
Telex 23515 hpe

Hewlett-Packard Española. S A.
Milanesado 21-23
E-**Barcelona** 17
Tel (3) 203 6200 (5 lines)
Telex 52603 hpbe e

Hewlett-Packard Española. S A
Av Ramón y Cajal 1-9
(Edificio Sevilla I)
E-**Seville 5**
Tel 64 44 54/58

Hewlett-Packard Española S A
Edificio Albia II 7 B
E-**Bilbao** 1
Tel 23 83 06/23 82 06

Calculators Only
Hewlett Packard Española S A
Gran Via Fernando El Católico, 67
E **Valencia**-8
Tel 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 3
Fack
S-161 20 **Bromma** 20
Tel (08) 730 05 50
Cable MEASUREMENTS
Stockholm
Telex 10721

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-421 32 **Västra Frölunda**
Tel (031) 49 09 50
Telex 10721 Via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zurcherstrasse 20
P O Box 307
CH-8952 **Schlieren-Zurich**
Tel (01) 730 52 40
Cable. HPAG CH
Telex, 53933 hpag ch

Hewlett-Packard (Schweiz) AG
Château Bloc 19
CH-1219 **Le Lignon-Geneva**
Tel (022) 96 03 22
Cable HEWPACKAG Geneva
Telex 27 333 hpag ch

SYRIA
Medical/Calculator only
Sawah & Co.
Place Azmé
B P 2308
SYR-**Damascus**
Tel 16367. 19697. 14268
Cable SAWAH, Damascus

TURKEY
Telekom Engineering Bureau
P O. Box 437
Beyoğlu
TR-**Istanbul**
Tel 49 40 40
Cable TELEMATION Istanbul
Telex 23609

Medical only
E M A
Muhendislik Kollektif Sirketi
Adakale Sokak 41/6
TR-**Ankara**
Tel 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No 16/6
Kizilay
TR-**Ankara**
Tel 25 03 09
Telex 42576 Ozek tr

UNITED KINGDOM
Hewlett-Packard Ltd
King Street Lane
GB-**Winnersh**, Wokingham
Berks. RG11 5AR
Tel. (0734) 78 47 74
Cable. Hewpie London
Telex 847178/9

Hewlett-Packard Ltd
"The Graftons"
Stamford New Road
GB-**Altrincham**
Cheshire WA14 IDQ
Tel (061) 9289021
Cable Hewpie Manchester
Telex 668068

Hewlett-Packard Ltd.
Lygon Court
Dudley Road
GB-**Halesowen**, Worcs
Tel: (021) 550 9911
Telex: 339105

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-**Thornton Heath**
Surrey CR4 6XL
Tel (01) 6640103
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Service Wholesale Centre
Wear Industrial Estate
Washington
GB-**New Town**, County Durham
Tel: Washington 464001 ext. 57/58

Hewlett-Packard Ltd
10, Wesley St.
GB-**Castleford**
West Yorkshire WF10 1AE
Tel. (09775) 50402
Telex: 557355

Hewlett-Packard Ltd
1, Wallace Way
GB-**Hitchin**
Herts
Tel (0462) 52824/56704
Telex: 825981

USSR
Hewlett-Packard
Representative Office USSR
Pokrovsky Boulevard 4/17-KV 12
Moscow 101000
Tel 294-2024
Telex: 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 **Ljubljana**
Tel 31 58 79/32 16 74
Telex: 31300

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m b.H
P.O. Box 7
A-1205 **Vienna**, Austria
Tel: (0222) 35 16 21 to 27
Cable: HEWPAK Vienna
Telex: 75923 hewpak a

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**, Greece
Tel: 8080337/359/429
8081741/742/743/744
Telex: 21-6588
Cable: HEWPACKSA Athens

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S A
7. rue du Bois-du-Lan
P O Box
CH-1217 Meyrin 2 - **Geneva**
Switzerland
Tel (022) 41 54 00

UNITED STATES

ALABAMA
8290 Whitesburg Dr S E
P O Box 4207
Huntsville 35802
Tel (205) 881-4591

Medical Only
228 W Valley Ave
Room 220
Birmingham 35209
Tel (205) 942 2081

ARIZONA
2336 E Magnolia St
Phoenix 85034
Tel (602) 244-1361

2424 East Aragon Rd
Tucson 85706
Tel (602) 294-3148

'ARKANSAS
Medical Service Only
P O Box 5646
Brady Station
Little Rock 72205
Tel (501) 664 8773

CALIFORNIA
1430 East Orangethorpe Ave
Fullerton 92631
Tel (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel (213) 877 1282
TWX 910-499-2170

6305 Arizona Place
Los Angeles 90045
Tel (213) 649-2511
TWX 910-328-6147

'Los Angeles
Tel (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel (408) 249-7000
TWX 910-338-0518

'Ridgecrest
Tel (714) 446-6165

646 W North Market Blvd
Sacramento 95834
Tel (916) 929-7222

9606 Aero Drive
P O Box 23333
San Diego 92123
Tel (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel (203) 389-6551
TWX 710-465-2029

FLORIDA
P O Box 24210
2806 W Oakland Park Blvd
Ft. Lauderdale 33307
Tel (305) 731-2020

'Jacksonville
Medical Service only
Tel (904) 725-6333

P O Box 13910
6177 Lake Ellenor Dr
Orlando 32809
Tel (305) 859-2900

P O Box 12826
Pensacola 32575
Tel (904) 434-3081

GEORGIA
P O Box 105005
Atlanta 30348
Tel (404) 955 1500
TWX 810 766-4890

Medical Service Only
'Augusta 30903
Tel (404) 736 0592

HAWAII
2875 So King Street
Honolulu 96814
Tel (808) 955-4455

ILLINOIS
5201 Tollview Dr
Rolling Meadows 60008
Tel (312) 255-9800
TWX 910-687 2260

INDIANA
7301 North Shadeland Ave
Indianapolis46250
Tel (317)842-1000
TWX 810-260-1797

IOWA
1902 Broadway
Iowa City 52240
Tel (319) 338-9466
Night (319) 338-9467

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr
Suite 207
Louisville 40218
Tel (502) 456-1573

LOUISIANA
P O Box 840
3239 Williams Boulevard
Kenner 70062
Tel (504) 721 6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel (301) 944-5400
TWX 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel (301) 948-6370
TWX 710-828-9684

MASSACHUSETTS
32 Hartwell Ave
Lexington 02173
Tel (617) 861-8960
TWX 710 326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel (313) 476-6400
TWX 810 242 2900

MINNESOTA
2400 N Prior Ave
Roseville 55113
Tel (612) 636-0700
TWX 910-563 3734

MISSISSIPPI
'Jackson
Medical Service only
Tel (601) 982 9363

MISSOURI
11131 Colorado Ave
Kansas City 64137
Tel (816) 763-8000
TWX 910 771 2087

148 Weldon Parkway
Maryland Heights 63043
Tel (314) 567 1455
TWX 910 764 0830

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel (402) 392 0948

NEW JERSEY
W 120 Century Rd
Paramus 07652
Tel (201) 265 5000
TWX 710-990-4951

NEW MEXICO
P O Box 11634
Station E
11300 Lomas Blvd N E
Albuquerque 87123
Tel (505) 292-1330
TWX 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel (505) 526 2485
TWX 910 983 0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel (518) 458-1550

201 South Avenue
Poughkeepsie 12601
Tel (914) 454-7330
TWX 510-248-0012

39 Saginaw Drive
Rochester 14623
Tel (716) 473-9500
TWX 510-253-5981

5858 East Molloy Road
Syracuse 13211
Tel (315) 454-2486

1 Crossways Park West
Woodbury 11797
Tel (516) 921-0300
TWX 710-990-4951

NORTH CAROLINA
P O Box 5188
1923 North Main Street
High Point 27262
Tel (919) 885 8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel (216) 243-7300
TWX 810-423-9431

330 Progress Rd
Dayton 45449
Tel (513) 859-8202
TWX 810-474-2818

1041 Kingsmill Parkway
Columbus 43229
Tel (614) 436-1041

OKLAHOMA
P O Box 32008
Oklahoma City 73132
Tel (405) 721 0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel (412) 782-0400
TWX 710-795-3124

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel (215) 265-7000
TWX 510-660-2670

SOUTH CAROLINA
6941-0 N Trenholm Road
Columbia 29260
Tel (803) 782-6493

TENNESSEE
'Knoxville
Medical Services only
Tel (615) 523-5022

1473 Madison Avenue
Memphis 38104
Tel (901) 274-7472

Nashville
Medical Service only
Tel (615) 244-5448

TEXAS
P O Box 1270
201 E Arapaho Rd
Richardson 75080
tel (214) 231-6101

P O Box 27409
6300 Westpark Drive
Suite 100
Houston 77027
Tel (713) 781-6000

205 Billy Mitchell Road
San Antonio 78226
Tel (512) 434 8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 487-0715

VIRGINIA
Medical Only
P O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel (804) 497-1026/7

P.O Box 9854
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

'WEST VIRGINIA
Medical/Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave.
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia... North Hollywood, California... Rockville, Maryland... Rolling Meadows, Illinois. Their complete addresses are listed above

'Service Only

1/77

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
11000 Wolfe Road, Cupertino, California 95014

PART NO. 02108-90002

Printed in U.S.A. 6/76