# MARK 5/9 *Reference Manual*

POINT 4
DATA CORPORATION

POINT 4 DATA CORPORATION

# POINT 4™ MARK 5/9 COMPUTER REFERENCE MANUAL

Revision 01

# NOTICE

Every attempt has been made to make this reference manual complete, accurate and up-to-date. However, all information herein is subject to change due to updates. All inquiries concerning this manual should be directed to POINT 4 Data Corporation.

POINT 4 Data Corporation
15442 Del Amo Avenue
Tustin, CA 92680
(714) 259-0777

# REVISION RECORD

**PUBLICATION NUMBER: HM-085-0023**

| Revision | Description | Date |
|---|---|---|
| 01 | Complete revision of original POINT 4 User Reference Manual incorporating information concerning the MARK 5, MARK 5 with EIS, and MARK 9 (61000 Assembly Series CPU Boards) | 07/22/86 |

# LIST OF EFFECTIVE PAGES

Changes, additions, and deletions to information in this manual are indicated by vertical bars in the margins or by a dot near the page number if the entire page is affected. A vertical bar by the page number indicates pagination rather than content has changed. The effective revision for each page is shown below.

| Page | Rev | Page | Rev | Page | Rev |
|---|---|---|---|---|---|
| Cover | - | | | | |
| Title | 01 | | | | |
| ii thru xv | 01 | | | | |
| 1-1 thru 1-20 | 01 | | | | |
| 2-1 thru 2-22 | 01 | | | | |
| 3-1 thru 3-29 | 01 | | | | |
| 4-1 thru 4-37 | 01 | | | | |
| 5-1 thru 5-41 | 01 | | | | |
| 6-1 thru 6-23 | 01 | | | | |
| 7-1 thru 7-30 | 01 | | | | |
| 8-1 thru 8-9 | 01 | | | | |
| Appendix Title | - | | | | |
| A-1 | 01 | | | | |
| B-1 | 01 | | | | |
| C-1 | 01 | | | | |
| D-1, D-2 | 01 | | | | |
| E-1 thru E-9 | 01 | | | | |
| F-1 | 01 | | | | |
| G-1 thru G-4 | 01 | | | | |
| H-1, H2 | 01 | | | | |
| Index-1 thru Index-9 | 01 | | | | |
| Comment Sheet | 01 | | | | |
| Mailer | - | | | | |
| Back Cover | - | | | | |

# PREFACE

This is a reference manual for the POINT 4 MARK 5 and MARK 9 Computers. These computers are 16-bit, high-speed, general-purpose minicomputers.

The introduction describes POINT 4 MARK 5 and MARK 9 standard features and system architecture. It also provides detailed information on equipment and performance characteristics.

Step-by-step directions for installation and detailed operating procedures are provided. A section describing input/output interfaces is included.

Separate sections describe the POINT 4 MARK 5 and MARK 9 standard and extended instruction sets. The next section describes stack, hardware multiply/divide, and mapped memory instructions for the POINT 4 MARK 9 Computer. Optional features are discussed fully in the final section.

The appendices include pertinent information on POINT 4 MARK 5/9 instructions, and offer programming examples.

Related manuals include:

| Title | Pub. Number |
|---|---|
| POINT 4 (MARK 5) Diagnostics Manual | HM-080-0014 |
| MIGHTY MUX User Manual | HM-042-0015 |
| MIGHTY MUX Diagnostics Manual | HM-042-0007 |
| Cassette Tape Unit Hardware Manual | HM-130-0017 |
| LOTUS Controllers User Manual | HM-120-0033 |
| LOTUS Cache Memory Hardware Manual | HM-160-0026 |
| MARK 9 CPU Diagnostics Document | HTP0041 |
| MARK 8 Verifier/Diagnostics Document | HTP0047 |
| MARK 5/9 Lowboy Computer System Unpacking and Installation Tech Memo | |
| MARK 5/9 Highboy Computer System Unpacking and Installation Tech Memo | |

# CONTENTS

## APPENDICES

## INDEX

## FIGURES

## TABLES

# Section 1
# INTRODUCTION

## 1.1 GENERAL DESCRIPTION

The POINT 4 MARK 5 and MARK 9 computers are 16-bit, high-speed, general-purpose minicomputers with versatile instruction sets. These POINT 4 computers employ a novel design architecture (Microprogrammed Sequencer*) to achieve the simplicity and flexibility of microprogrammed design with the speed of hard-wired logic design. In addition, the design allows direct addressing of 64K words of MOS random access memory (RAM). With mapped memory, the MARK 9 Computer can access 256K words of memory. An Extended Instruction Set on the MARK 9 (optional on MARK 5) computer increases the system throughput while decreasing software requirements. These features make POINT 4 MARK 5 and MARK 9 computers well-suited to OEM applications in business data systems, communications, and control systems. Figures 1-1 and 1-2 are photographs of POINT 4 MARK 5 and MARK 9 computer systems.

*U.S. Patent No. 4,370,729

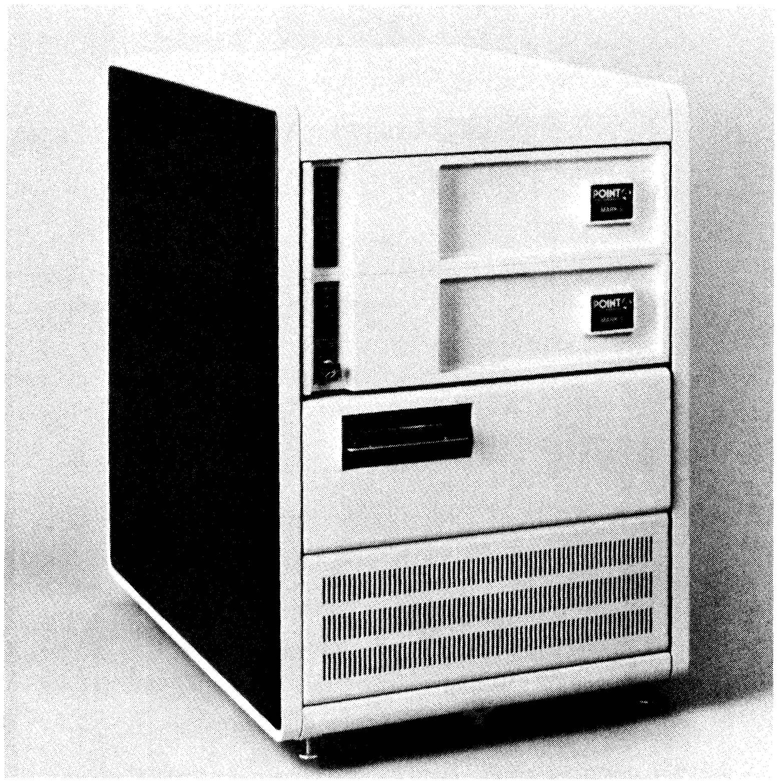


Figure 1-1. POINT 4 MARK 5 Computer

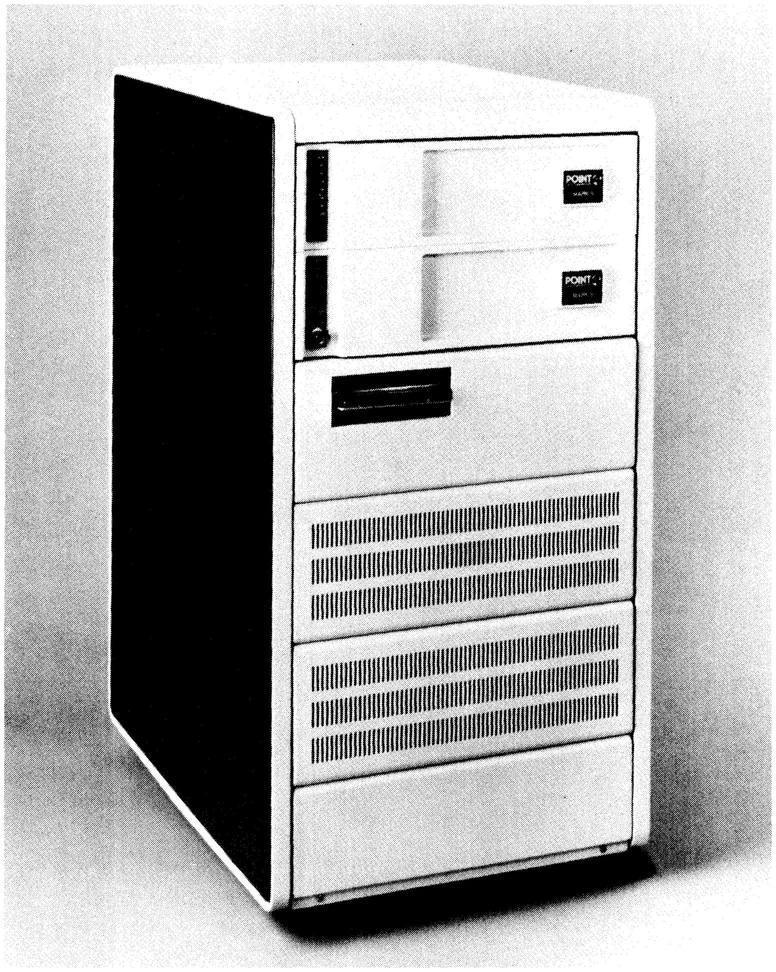


Figure 1-2. POINT 4 MARK 9 Computer

### 1.1.1 FEATURES

**MARK 5 Features**

The POINT 4 MARK 5 Computer includes the following features:

- CPU and 64K words of RAM on the same board
- Nova*-compatible instruction set
- High-speed, 400-nanosecond cycle time (fetch and execution)
- Standard or high-speed data channel (jumper option)
- Main memory Battery Backup option
- Main memory parity error halt option
- Virtual Control Panel
- Built-in Self-Test for extensive verification of memory and CPU operation
- Jumper-free backplane
- Detachable Operator Control Unit option
- Seven-slot front-load chassis and separate power supply
- Full IRIS Operating System compatibility
- 12V regulator option
- POINT 4 Extended Instruction Set (EIS) option

*Nova is a trademark of Data General Corporation

## MARK 9 Features

The POINT 4 MARK 9 Computer includes the following features:

- CPU with 128K words of onboard RAM
- 256K words of onboard RAM (option)
- Nova-compatible instruction set
- POINT 4 Extended Instruction Set (EIS)
- MARK 9-exclusive hardware multiply/divide and stack instructions
- High-speed, 300-nanosecond cycle time (fetch and execution)
- Mapped memory
- Standard or high-speed data channel (jumper option)
- Modified hamming error code circuitry
- Main memory Battery Backup option
- Virtual Control Panel
- Built-in Self-Test for 64K words of memory and CPU printed circuit board operation
- Jumper-free backplane
- Detachable Operator Control Unit option
- Seven-slot, front-load chassis and separate power supply
- Full IRIS Operating System compatibility

### 1.1.2 MODELS

The POINT 4 MARK 5 CPU board is available in two models: standard and universal. Both models use the same CPU and memory. The standard POINT 4 MARK 5 CPU board is used with a POINT 4 chassis. The MARK 5 universal CPU board is designed to replace CPU and memory boards in a Nova-compatible chassis. The MARK 5 universal CPU board includes a series of switches and light-emitting diode (LED) monitors. These switches and LED monitors perform most of the functions that are performed by the Mini-Panel on the POINT 4 CPU chassis.

The POINT 4 MARK 9 CPU board is also available in both standard and universal models as described above.

**NOTE**

> Unless otherwise noted, information in this manual pertains to both standard and universal models of POINT 4 MARK 5 and 9 CPUs.

### 1.1.3 PERIPHERALS SUPPORTED

The following peripherals, available from various manufacturers, are supported: Teletypes, video display terminals, printing terminals, diskettes, fixed-head disks, cartridge disks, disk pack drives, magnetic tape units, cassettes, paper tape readers, paper tape punches, line printers, character printers, plotters, card readers, and card punches.

### 1.1.4 OPTIONS

**MARK 5 Options**

The following options are available to enhance the POINT 4 MARK 5 Computer:

1. The main memory parity error halt option allows detection of data errors in memory transfers.

2. Main memory battery backup maintains the contents of memory for at least 2 hours in the event of a power failure.

3. An Operator Control Unit may be attached to the CPU front panel or extended by cable to a convenient work surface.

4. The voltage regulator option provides a 12V regulator on the PC board for use in a chassis that does not provide a regulated +12V supply.

5. The POINT 4 Extended Instruction Set (EIS) provides increased versatility and efficiency.

6. The Micro-Programmed Floating Point Unit (MFPU) reduces floating-point computation time. This option is available only if the MARK 5 CPU also has the EIS option.

**MARK 9 Options**

The following options are available to enhance the POINT 4 MARK 9 Computer:

1. Main memory battery backup maintains the contents of memory for at least 2 hours in the event of a power failure.

2. An Operator Control Unit may be attached to the CPU front panel or extended by cable to a convenient work surface.

3. The MARK 9 is equipped with 128K words of memory. Optional memory may be purchased to increase main memory to 256K words.

## 1.2 EQUIPMENT SPECIFICATIONS

**MARK 5 CPU Specifications**

| | |
|---|---|
| Word length | 16 bits |
| General purpose accumulators | 4 |
| Memory cycle time | 400 nanoseconds |
| RAM access time | 200 nanoseconds |
| Crystal-controlled clock frequency | 30 MHz |
| Memory | 64K words |
| Standard data channel | Input - 1100 nanoseconds<br>Output - 1433 nanoseconds |
| High-speed data channel | Input - 800 nanoseconds<br>Output - 933 nanoseconds |
| Interrupt response | 1200 nanoseconds |
| CPU board size | 15 in. x 15 in.<br>(38.10 cm x 38.10 cm) |
| CPU board power consumption (max) | -5V(1mA)+5V(4A)+12V(1.5A) |

**MARK 9 CPU Specifications**

| | |
|---|---|
| Word length | 16 bits |
| General purpose accumulators | 4 |
| Cycle time (fetch and execution) | 300 nanoseconds |
| RAM access time | 120 nanoseconds |
| Crystal-controlled clock frequency | 40 MHz |
| Memory size | 128K words<br>256K words (optional) |
| Standard data channel | Input - 1600 nanoseconds<br>Output - 2000 nanoseconds |
| High-speed data channel | Input - 900 nanoseconds<br>Output - 1200 nanoseconds |
| Interrupt response | 1175 nanoseconds |
| CPU board size | 15 in. x 15 in.<br>(38.10 cm x 38.10 cm) |
| CPU board power consumption (max) | +5V (14.0A) without battery backup<br>+5V (11.0A) with battery backup<br>+5V battery backup (3.0A) |

## Electrical Specifications

AC Input - 99 to 129V AC, 6A max, 47 to 64 Hz
Optional - 199 to 257V AC, 3A max, 47 to 64 Hz
AC Power Consumption - 450 watts max

## Mechanical Specifications

Processor Chassis:
Height - 5.25 in. (13.34 cm)
Width - 19.0 in. (48.26 cm)
Depth - 17.5 in. (44.45 cm)
Weight - 20 lbs (9.07 kg)

Operator Control Unit:
Height - 3.62 in. (9.19 cm)
Width - 6.12 in. (15.54 cm)
Depth - 1.1 in. (2.79 cm)
Weight - .75 lbs (.34 kg)

Power Supply Chassis:
Height - 5.25 in. (13.34 cm)
Width - 19.0 in. (48.26 cm)
Depth - 10.0 in. (25.40 cm)
Weight - 25 lbs (11.34 kg)

## Environmental Specifications

Temperature Range - 0 to 50°C (32 to 122°F)
Relative Humidity - 10 to 90% noncondensing

# 1.3 SYSTEM ARCHITECTURE

The POINT 4 MARK 5 and MARK 9 use an architecture that reduces signal interfacing between boards to a minimum. This allows maximum speed of instruction execution. A combination of central processor logic and 64K words (128K or 256K words on MARK 9) of RAM on a single circuit board eliminates time delays due to long, asynchronous, memory access bus paths. The CPU board contains all basic functions of the computer. This leaves only I/O controller boards to be added to the system. Figure 1-3 illustrates a typical configuration of a POINT 4 MARK 5 or MARK 9 system.

## 1.3.1 SYSTEM FUNCTIONAL UNITS

POINT 4 MARK 5 and MARK 9 computers utilize five functional units; the CPU and memory board, the processor chassis and front panel, the Mini-Panel, an optional Operator Control Unit, and the power supply. Figure 1-4 is a diagram of the computer system showing the functions each unit performs.

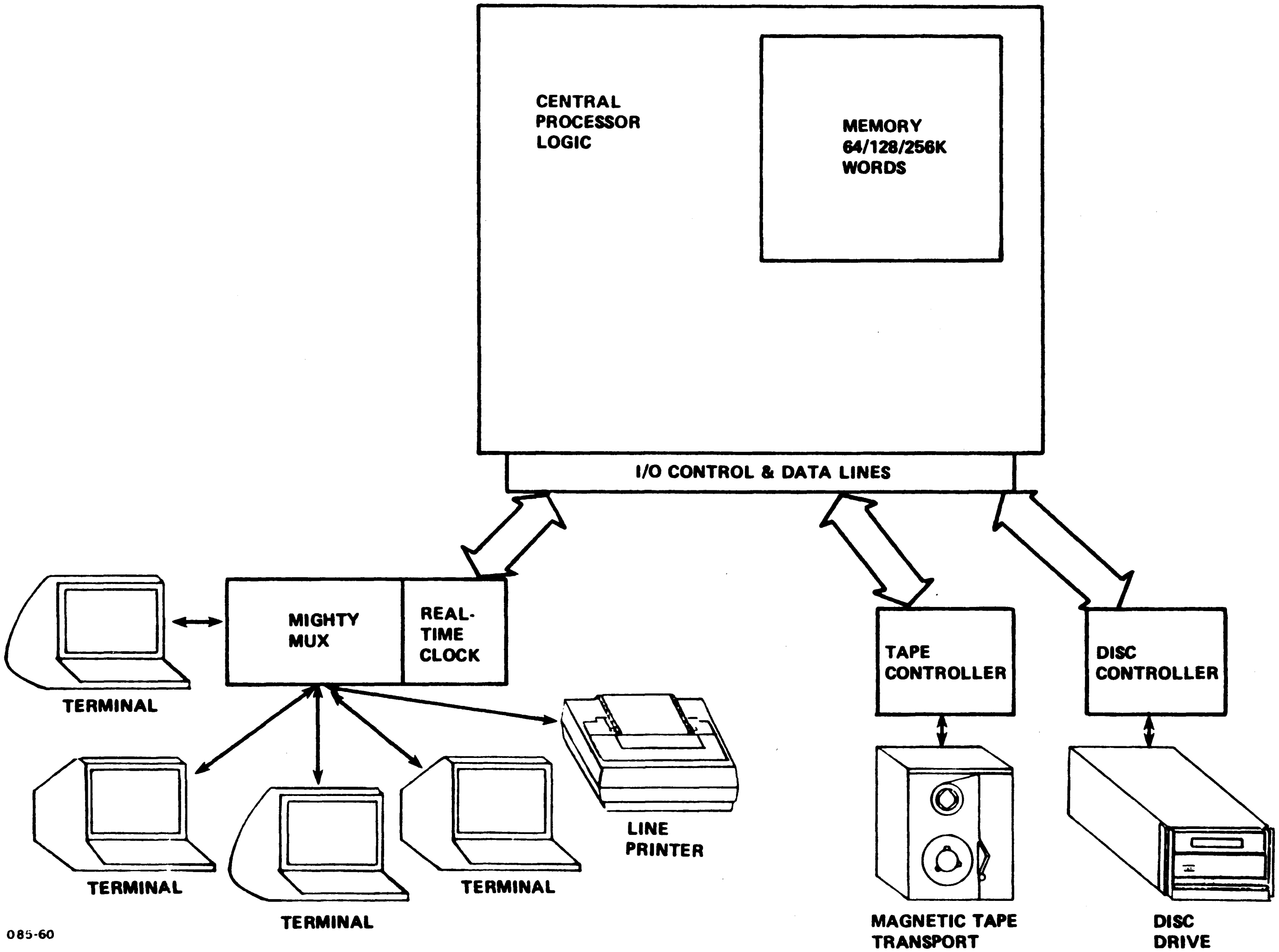


**Figure 1-3. Typical POINT 4 MARK 5/9 Computer System Configuration**

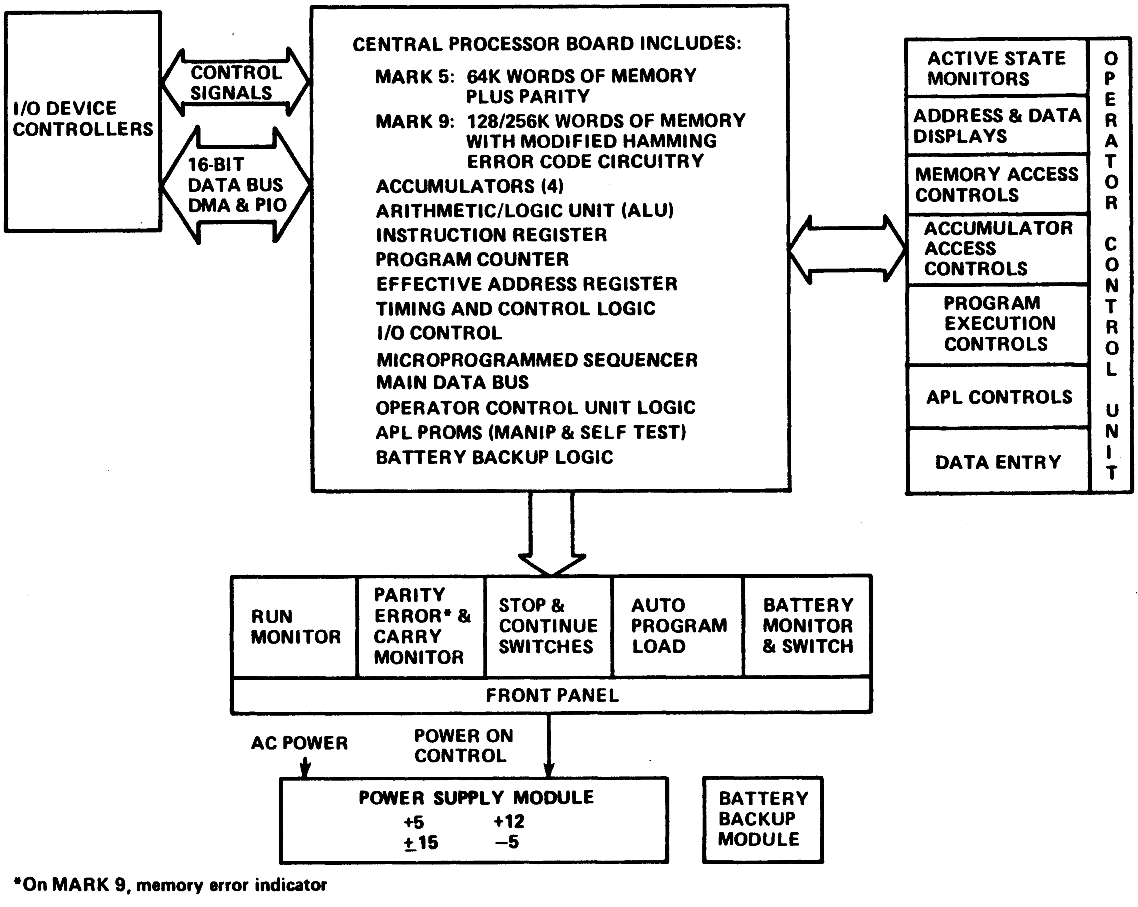


**Figure 1-4. POINT 4 MARK 5/9 Computer System Block Diagram**

#### 1.3.1.1 Central Processor and Memory Board

The central processor board contains the basic elements of the CPU:

- Four general-purpose accumulators
- Arithmetic/logic unit (ALU)
- Instruction register
- Main data bus
- Program counter
- Effective address register
- Timing control
- I/O control

In addition to these basic CPU elements, the CPU board contains a variety of features that are often omitted or housed on separate boards. The compact CPU board contains the following features:

- 64K words of RAM (MARK 5)
- 128K or 256K words of RAM (MARK 9)
- Parity generation and error detection logic (option)
- Error detection and correction (EDAC) circuitry (MARK 9)
- Operator Control Unit interface logic
- Battery Backup interface logic
- Automatic program load (APL) PROMs containing a program to implement the Virtual Control Panel features
- Self-Test PROMs containing a hardware checkout routine
- Switches and indicators allowing onboard Mini-Panel functions (universal CPU model)
- +12V regulators for use in a chassis that does not provide a regulated +12V supply (universal CPU model)

Figure 1-5 is a block diagram of the MARK 5 CPU/memory board. Figure 1-6 is a block diagram of the MARK 9 CPU/memory board.

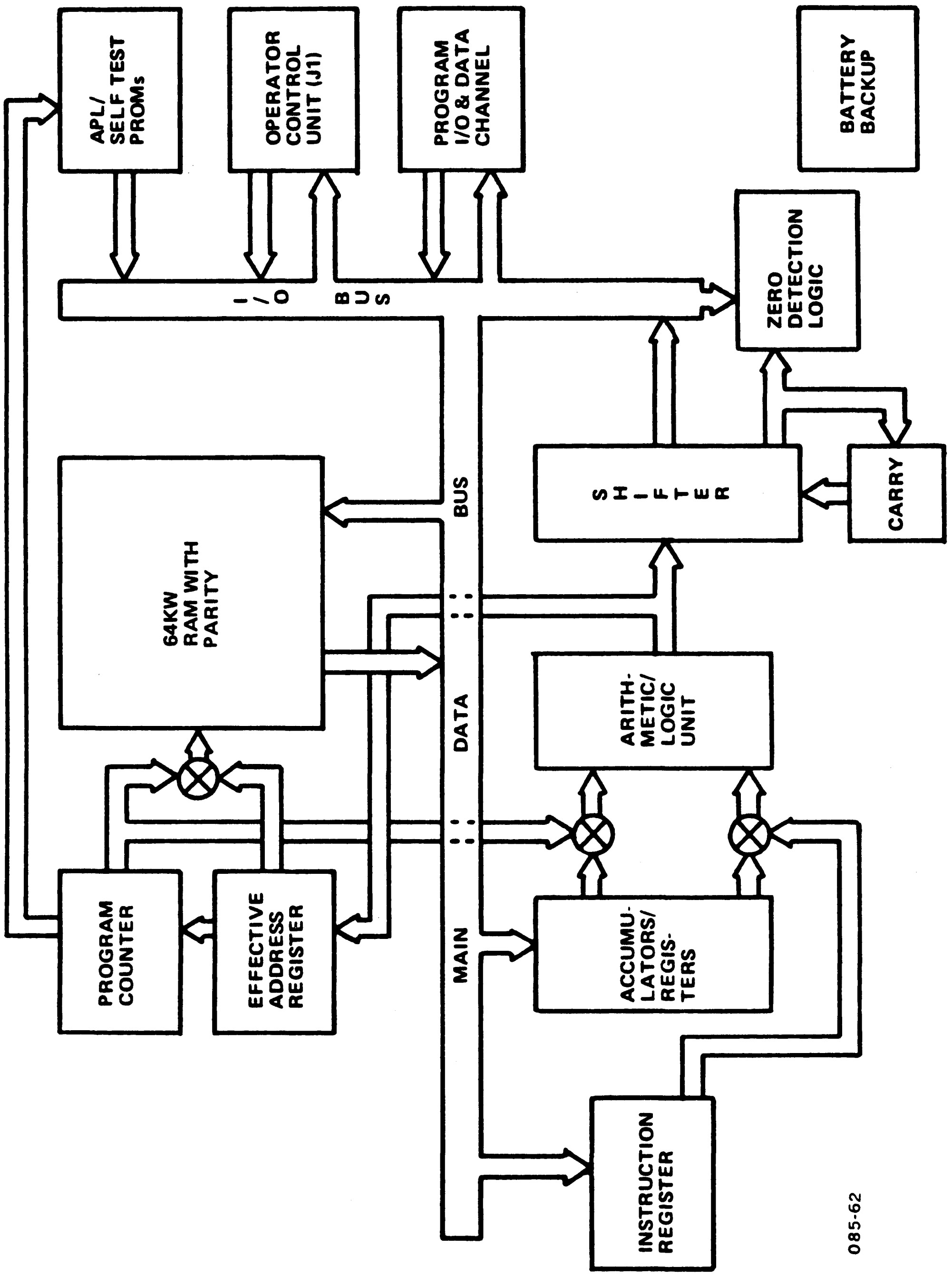
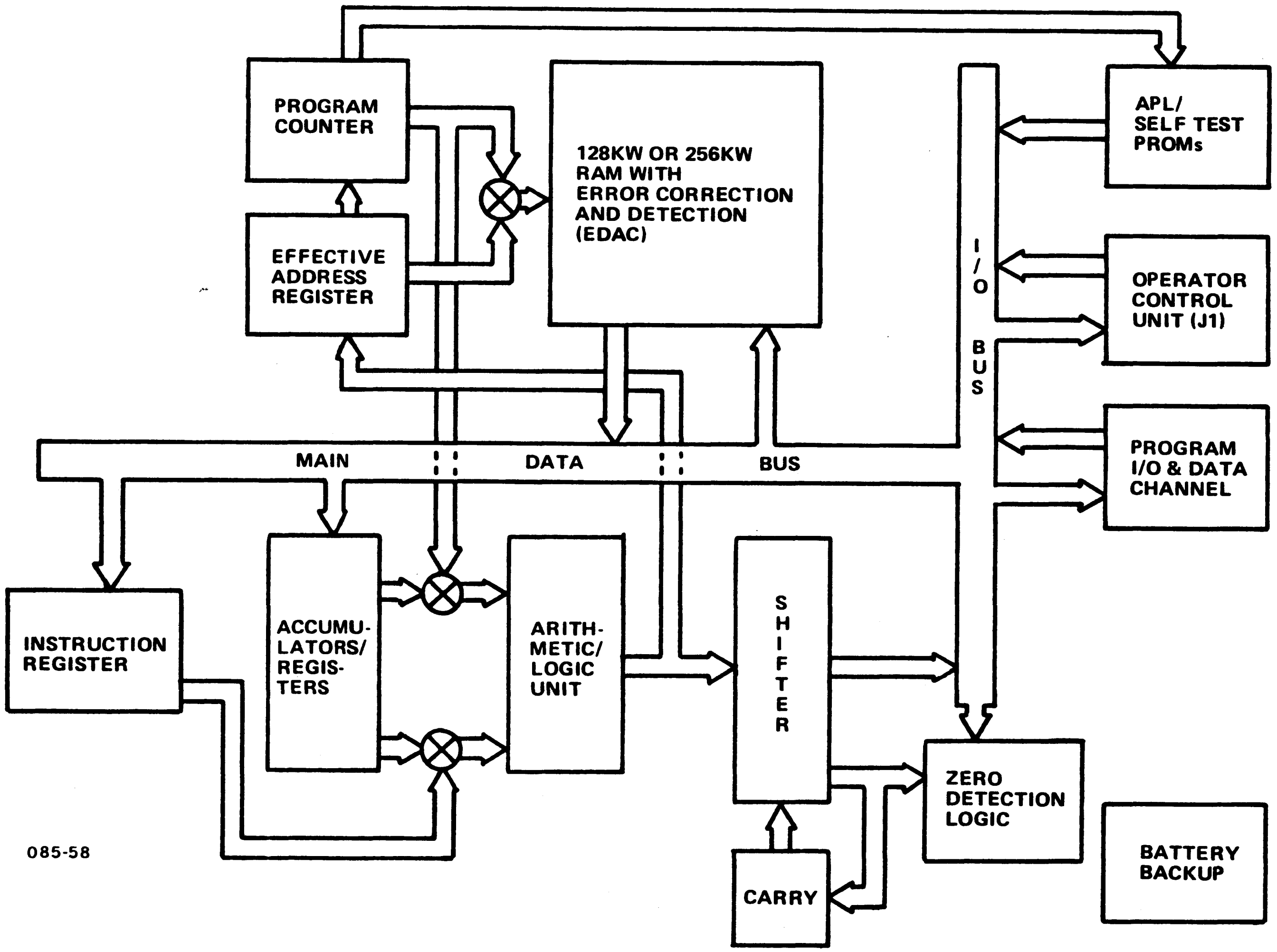


**Figure 1-5. POINT 4 MARK 9 CPU/Memory Block Diagram**



PROGRAM COUNTER
EFFECTIVE ADDRESS REGISTER
128KW OR 256KW RAM WITH ERROR CORRECTION AND DETECTION (EDAC)
I/O BUS
APL/ SELF TEST PROMs
OPERATOR CONTROL UNIT (J1)
PROGRAM I/O & DATA CHANNEL
MAIN
DATA
BUS
INSTRUCTION REGISTER
ACCUMU-LATORS/ REGIS-TERS
ARITH-METIC/ LOGIC UNIT
SHIFTER
ZERO DETECTION LOGIC
CARRY
BATTERY BACKUP
085-58

Figure 1-6. POINT 4 MARK 9 CPU/Memory Block Diagram

### 1.3.1.2 Processor Chassis and Front Panel

The POINT 4 MARK 5/9 processor chassis is designed to be mounted in a standard 19-inch equipment rack. The chassis contains seven slots spaced on .6-inch centers. The processor chassis is 5.25 inches high, 19 inches wide, and 17.5 inches deep.

Cooling is provided by two whisper-quiet fans. These are mounted on the left side of the chassis behind the Mini-Panel.

The front panel snaps on and off. No screws or hinges hold it in place. No cabling exists between the front panel and the chassis (except optional Operator Control Unit cabling) since the Mini-Panel is mounted directly onto the chassis and its controls are accessible through a slot on the left side of the front panel.

### 1.3.1.3 Processor Mini-Panel

The Mini-Panel on the POINT 4 MARK 5/9 chassis houses an efficient set of controls and indicators for basic processor operation. Mini-Panel indicators include light-emitting diode indicators for monitoring of parity errors (MARK 5), memory errors (MARK 9), the Carry flag, CPU operation, power OK, and battery OK. Processor control switches include a four-position switch controlling ON, AUTO operation, STANDBY, and OFF. Program control switches allow stopping and restarting of program execution and enabling of the Virtual Control Panel. For further details, see Section 3.3 on Mini-Panel operations.

The Virtual Control Panel allows monitoring and control of the processor from a master terminal through use of the manipulator program MANIP. For further details, see Section 3.5 on use of the Virtual Control Panel.

### 1.3.1.4 Operator Control Unit (Optional)

In addition to the basic controls and indicators on the POINT 4 MARK 5 or 9 processor chassis, an Operator Control Unit is available that enhances operator access to the processor. This detachable control unit can be attached to the front panel of the chassis, or can be extended via ribbon cable to any convenient work surface. The compact control unit contains all switches and indicators necessary to monitor and control the processor. See Figure 1-7 for a photograph of the Operator Control Unit.

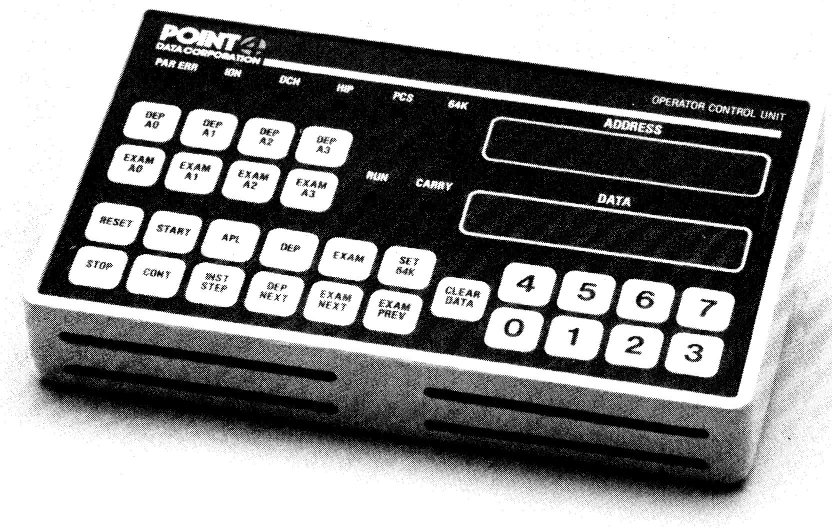


Figure 1-7. Operator Control Unit

Displays include two octal displays for address and data, and eight light-emitting diodes indicating activity in the following areas: data channel, programmable control store, high-speed interprocessor bus, 64K-word addressing, program execution, interrupts enabled, carry condition, and parity error detected. Sealed membrane switches provide an octal data entry panel, a clear data button, and the following controls:

- Memory is accessible through control switches to examine and deposit in memory, as well as to enable 64K-word addressing
- Accumulators are accessible through examine and deposit controls
- Program execution controls include reset, start, stop
- Automatic Program Load (APL) switch

See Section 3.4 for further description of the Operator Control Unit.

### 1.3.1.5 Power Supply Module

The power supply module is housed in a separate chassis (19 inches wide, 5.25 inches high, and 10 inches deep) for flexibility of installation. The power supply consists of two units: a regular power supply for normal use, and a battery backup unit (optional), which is used in case of power failure.

The regular power supply module delivers the power required for the CPU board and front panel logic, the Operator Control Unit, plus the power required for the I/O device controllers housed in the POINT 4 MARK 5 or MARK 9 chassis.

The power supply requires an input voltage of 117V AC or 234V AC, 47 to 64 Hz. Power supply output voltages are:

| Voltage | Supplied to |
|---|---|
| +5V | CPU board, Operator Control Unit, I/O controllers |
| -5V | CPU, I/O controllers* |
| +15V | For use by I/O controllers |
| -15V | For use by I/O controllers |
| +17V Pk<br>60 Hz | CPU slot** |

*Not used by MARK 9 CPU. MARK 5 CPUs use this voltage.

**Not used in POINT 4 chassis. Used in non-POINT 4 chassis as a timing signal.

These voltages are available for I/O controller applications. Output current available for each voltage and supply type are:

| Voltage | Output | Power Supply Type |
|---|---|---|
| +5V DC | 35.0 amps | Switching supply |
| -5V DC | 1.0 amps | Switching supply |
| +15V DC | 2.5 amps | Switching supply |
| -15V DC | 2.5 amps | Switching supply |

The +5V power supply needs a minimum consumption level of 3.5 amps for the other voltages to be in tolerance. The CPU board, when plugged in, will provide this load. The battery backup option protects memory contents for at least 2 hours in the event of a power failure. The battery backup unit is maintained in a charged state by the power supply as long as the unit is plugged in and AC power is available (even if the key switch is in the OFF position). Battery backup voltages are:

| Voltage | Supplied to |
|---|---|
| +5V BU | CPU board, Mini-Panel, memory refresh logic |
| -5V BU | Memory on CPU board (MARK 5 only) |
| +12V BU | Memory on CPU board (MARK 5 only) |

These outputs are supplied to the CPU board only. They are low current output.

The power supply is divided into two PCB subassemblies with switching regulators on each board. The Primary Inverter has an "off-line" switcher that works directly off the AC power line. The Battery Backup (BBU) has a "flyback" switcher that operates off a low-voltage output of an AC line transformer. The AC transformer also powers the supply chassis and computer chassis fans. Figure 1-8 is a block diagram of the power supply unit.

Battery backup interface logic includes a sensor to determine whether the power control cable is connected to the CPU chassis. The power supply unit will not supply power to the processor if this cable is disconnected. The processor may also be nonfunctional if the cable is connected but the CPU Mini-Panel keyswitch is set to the OFF position, disabling all power to the system.

If the Mini-Panel keyswitch is set to any position except OFF, power will be supplied to the system and operation of the unit depends on the presence or lack of the Battery Backup Unit.

When the BBU is installed, the system switches to battery power upon loss of AC. A control signal informs the CPU of the Power Fail condition before the power is out of tolerance. The +5V BU, -5V BU and +12V BU take over supply of power to CPU Memory only.

Logic on the comparator board tests whether output voltages are in tolerance. If a voltage is in tolerance, the appropriate LED on the power supply Mini-Panel is illuminated. If all voltage outputs are in tolerance, a signal is sent to the CPU Mini-Panel which turns on the POWER OK LED. When batteries are charging, the BTRY OK LED on the CPU Mini-Panel is off; it illuminates when installed batteries are fully charged.

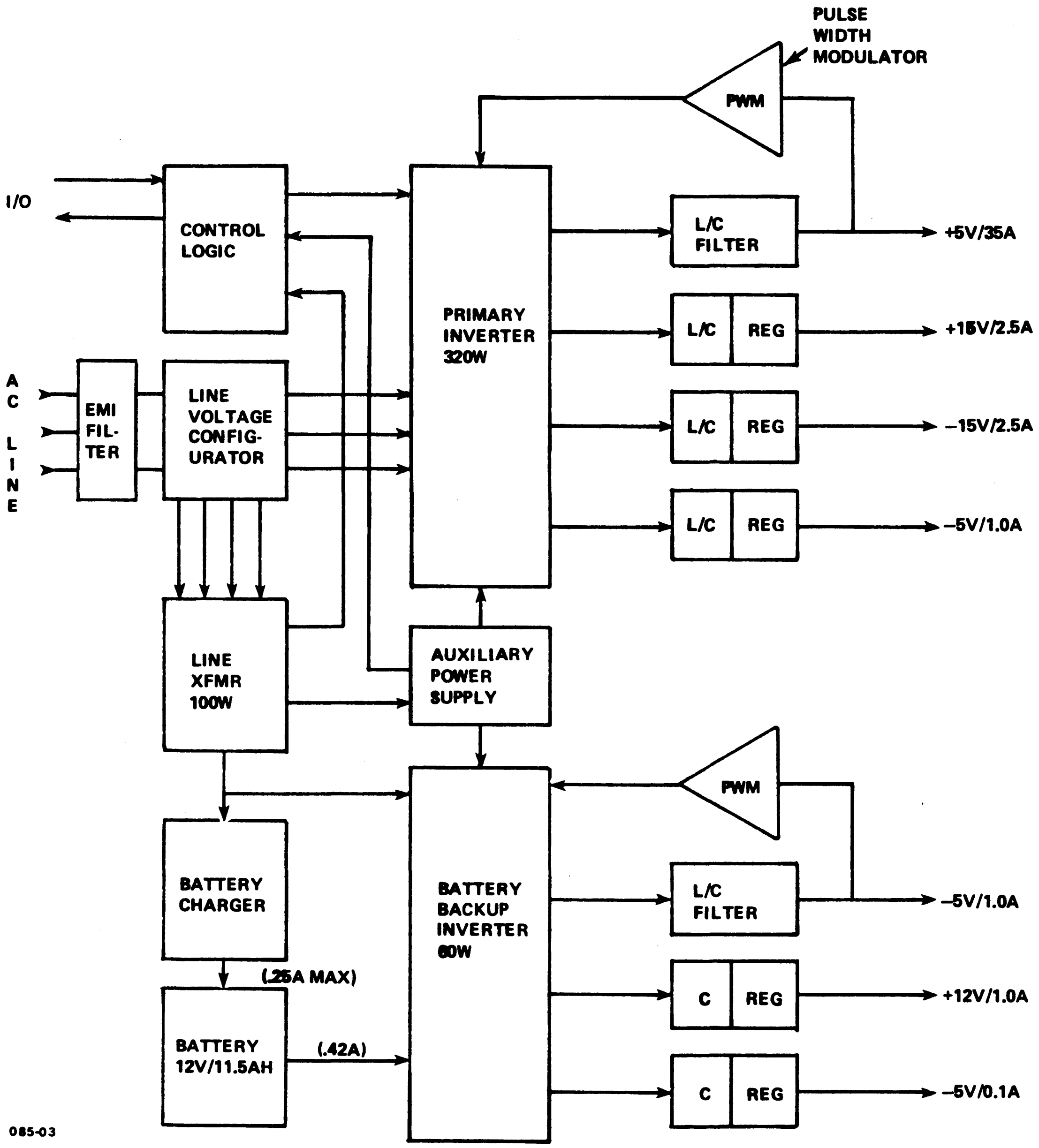


Figure 1-8. POINT 4 MARK 5/9 Power Supply
Block Diagram

### 1.3.2 DATA CHANNEL

An interrupt and the execution of several instructions are required for each word transferred between input/output (I/O) devices and memory via program control. To allow greater transfer rates between memory and external devices, the processor is equipped with a data channel through which a device can, at its own request, access memory directly using a minimum of processor time. A high-speed device, such as a disk or tape drive, can access memory without assistance from the program. Program execution simply pauses momentarily while a data channel transfer takes place on the I/O bus.

The data channel releases processor time by allowing execution of a program concurrently with data transfers for a device. Many devices may share the data channel.

The POINT 4 MARK 5 and MARK 9 computers have an option that allows jumper selection of data channel speed:

| | MARK 5 | MARK 9 |
|---|---|---|
| Standard Data Channel | | |
| Input - (nanoseconds) | 1100 | 1600 |
| Output - (nanoseconds) | 1483 | 2000 |
| High-Speed Data Channel | | |
| Input - (nanoseconds) | 800 | 900 |
| Output - (nanoseconds) | 933 | 1200 |

The choice of standard or high-speed data channel depends on the ability of all peripherals to respond within the maximum time allowed. See Section 4.4 for data channel jumpering instructions.

# Section 2
# INSTALLATION

## 2.1 ENVIRONMENTAL REQUIREMENTS

Careful consideration must be given to the placement of any computer system prior to installation to ensure that all power and environmental requirements are met. Necessary pre-installation considerations are discussed in the following subsections.

### 2.1.1 POWER REQUIREMENTS

The POINT 4 MARK 5/9 requires a power source of 117V AC, 47 to 64 Hz with 6 amperes maximum current draw; or a 237V AC, 47 to 64 Hz power source with 3 amperes current draw. In addition to the power requirements for the POINT 4 MARK 5/9, the power resources and electrical outlets needed to handle all peripheral devices must be considered.

### 2.1.2 TEMPERATURE REQUIREMENTS

Adequate environmental controls are needed to maintain the POINT 4 MARK 5/9 in the desired temperature range of 20 to 30$^{O}$C (68 to 86$^{O}$F). The maximum operating range is 0 to 50$^{O}$C (32 to 122$^{O}$F).

### 2.1.3 ENCLOSURE REQUIREMENTS

The POINT 4 MARK 5/9 is packaged in a seven-slot chassis. The power supply module is mounted in a separate chassis. This separation provides for flexibility of installation and also helps isolate heat and noise from the processor. Each chassis measures 5.25 inches high. The two chassis are designed to be mounted in a standard 19-inch equipment rack. The units require sufficient free space to allow for cooling air flow.

## 2.2 UNPACKING INSTRUCTIONS

POINT 4 MARK 5/9 Computer products receive a complete test and inspection prior to shipment. It is recommended that each unit be inspected for completeness and shipping damage prior to installation. Each carton should be inspected for any evidence of damage due to dropping, puncturing, or crushing. If damage is evident, contact the carrier and the POINT 4 Data Corporation Sales Representative for further instructions.

The following subsections describe unpacking for separate chassis units. Separate unpacking/installation documents are available for MARK 5 and MARK 9 lowboy and highboy systems.

### 2.2.1 UNPACKING THE CARTONS

The POINT 4 MARK 5 or MARK 9 and the power supply chassis are packaged in double-walled, corrugated cartons. Styrofoam packing inserts surround the chassis. The CPU board is packaged in a cardboard carton and placed on top of the chassis. Figure 2-1 illustrates the processor chassis packaging. Figure 2-2 illustrates the power supply chassis packaging.

### 2.2.2 CONTAINER CONTENTS

Each item removed from the carton should be checked against the packing slip. All items, including cable connectors, should be inspected for damage. If items are damaged or broken, contact the POINT 4 Data Corporation Sales Representative.

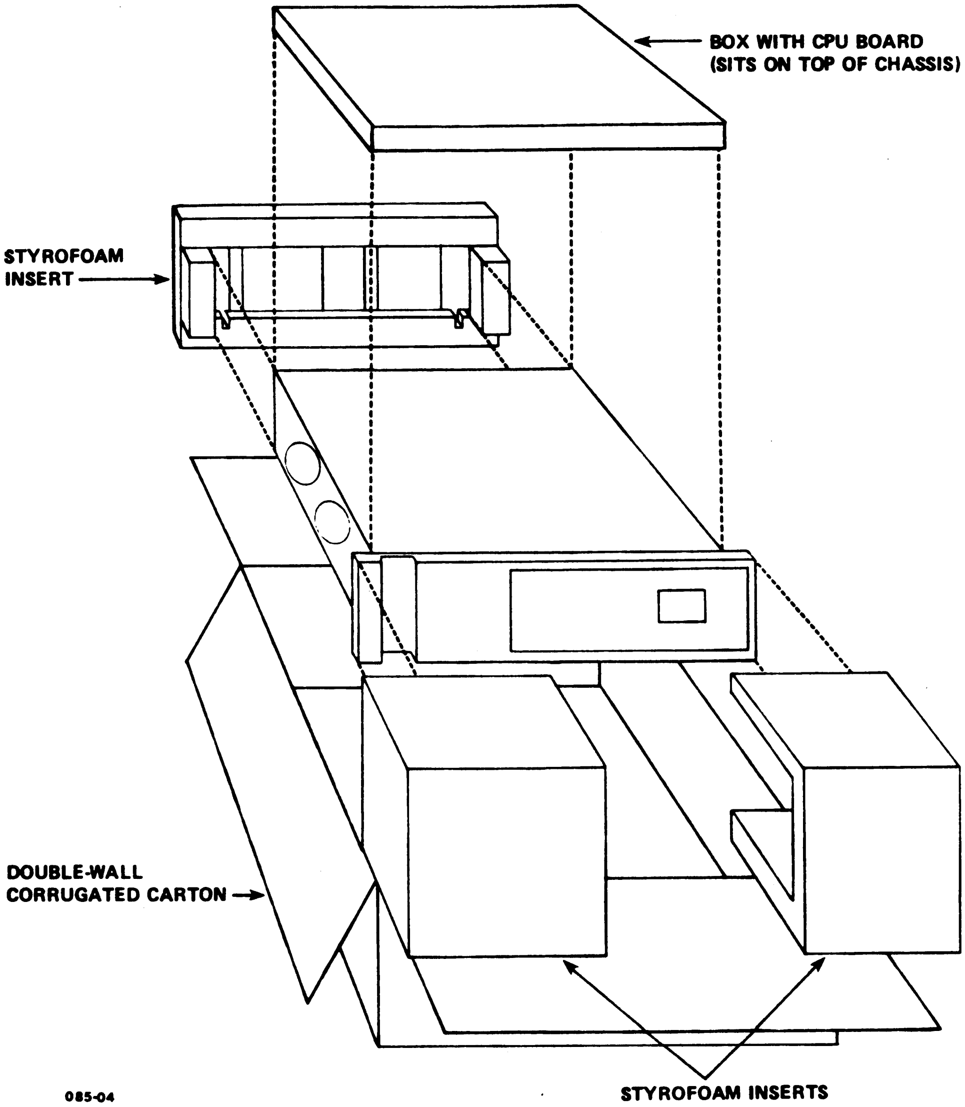


Figure 2-1. POINT 4 MARK 5/9 Processor Chassis Packaging

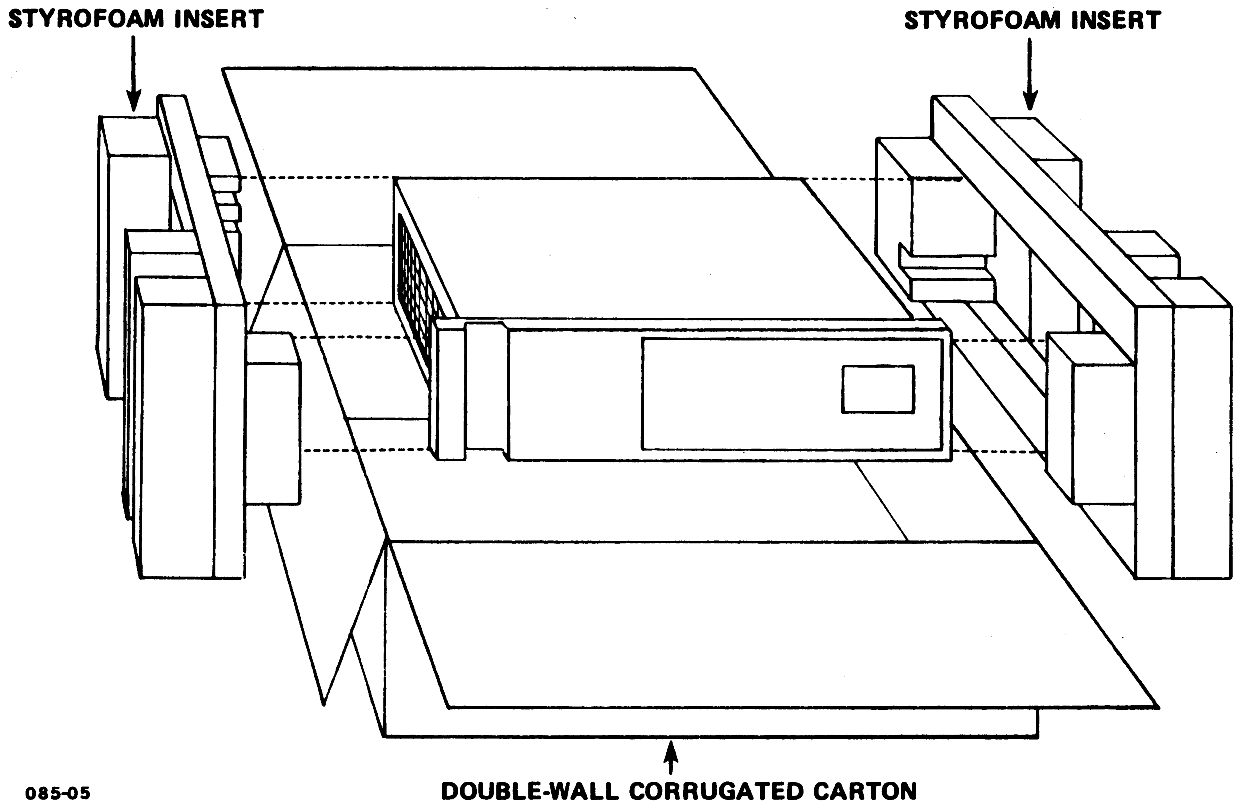


Figure 2-2. POINT 4 MARK 5/9 Power Supply Packaging

## 2.3 PROCESSOR AND POWER SUPPLY MOUNTING

The processor and the power supply chassis are designed to be mounted in a standard 19-inch equipment rack. Although the exact procedure may vary, each chassis is mounted according to the following general procedure.

On MARK 5 and MARK 9 lowboy and highboy systems, the chassis are mounted prior to shipment.

### 2.3.1 FRONT PANEL

The POINT 4 MARK 5/9 processor and power supply front panels snap off for easy removal. No screws or hinges hold them in place. Removing the front panel reveals mounting slots on each side of the chassis.

CAUTION

> The POINT 4 MARK 9 airflow system requires the front panels to be in place when the computer is being run.

### 2.3.2 CHASSIS

The chassis has two mounting slots on each side. Figure 2-3 shows the mounting slot locations on both processor and power supply chassis.

CAUTION

> Care must be taken not to crimp, or in any way damage the cables connected to the processor chassis while mounting the chassis.

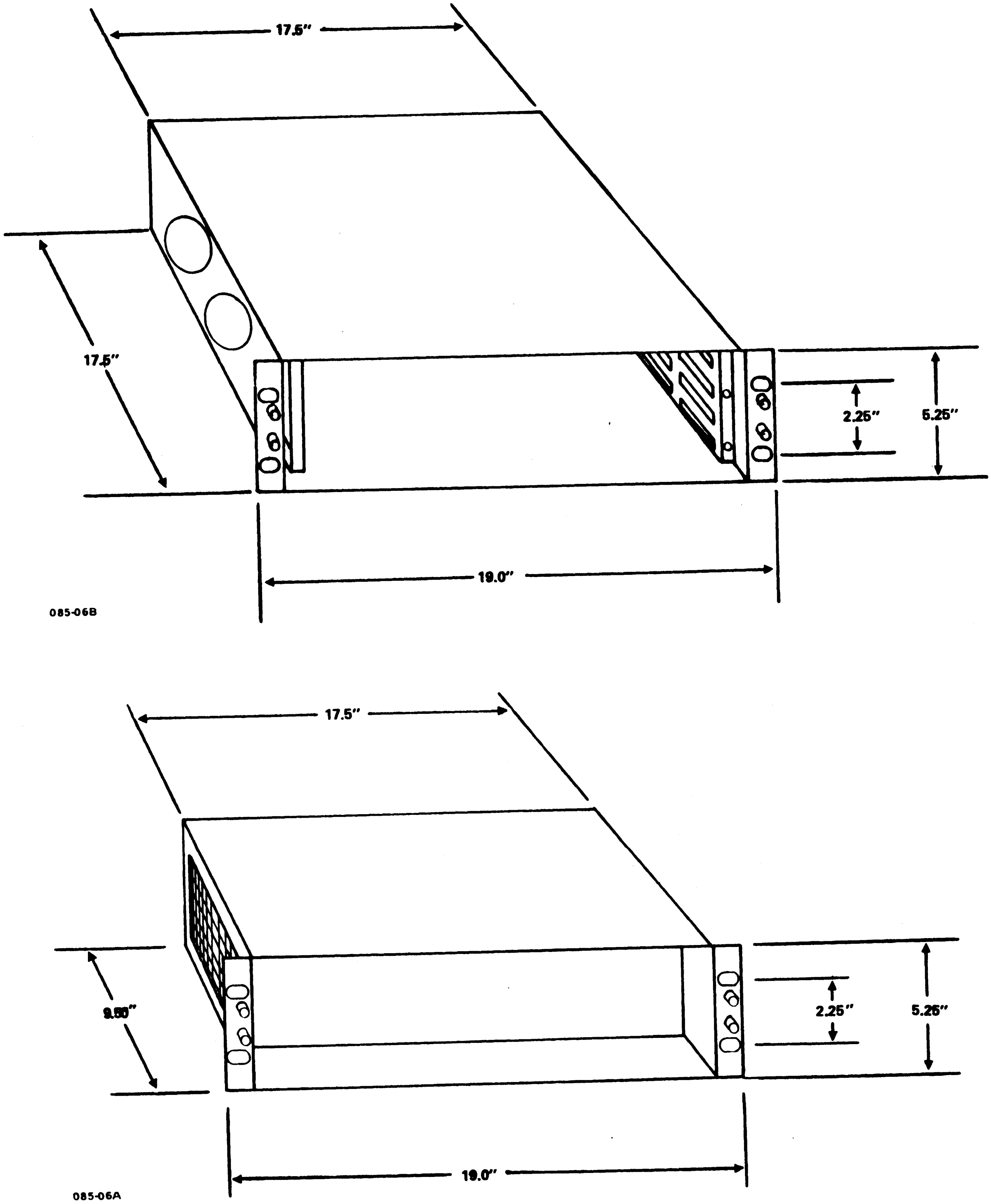


Figure 2-3. Processor Chassis and Power Supply Chassis Mounting Slots

### 2.3.3 PRINTED CIRCUIT BOARDS

The seven slots on the processor chassis are numbered 1-7 from top to bottom. The processor memory board always occupies the top slot. Printed circuit boards are mounted component-side up. A typical board configuration would be:

- POINT 4 MARK 5/9 CPU/memory board
- POINT 4 LOTUS 730 Multifunction Controller Board (or POINT 4 LOTUS 710 Disk Controller, POINT 4 LOTUS 720 Tape Controller)
- POINT 4 MIGHTY MUX (8-port)
- POINT 4 MIGHTY MUX 16-port Asynchronous Expansion Board
- POINT 4 LOTUS Cache Memory

The jumper-free backplane allows for flexibility in board organization. There is no need to jumper data-channel interrupt priority around unused slots. Figure 2-4 illustrates a typical board configuration.

Except for the CPU board (see Section 2.2.1), POINT 4 Data Corporation-manufactured printed circuit boards are shipped in separate cartons. They should be installed in the chassis with care. The card edge connectors must slide smoothly into the backplane sockets.

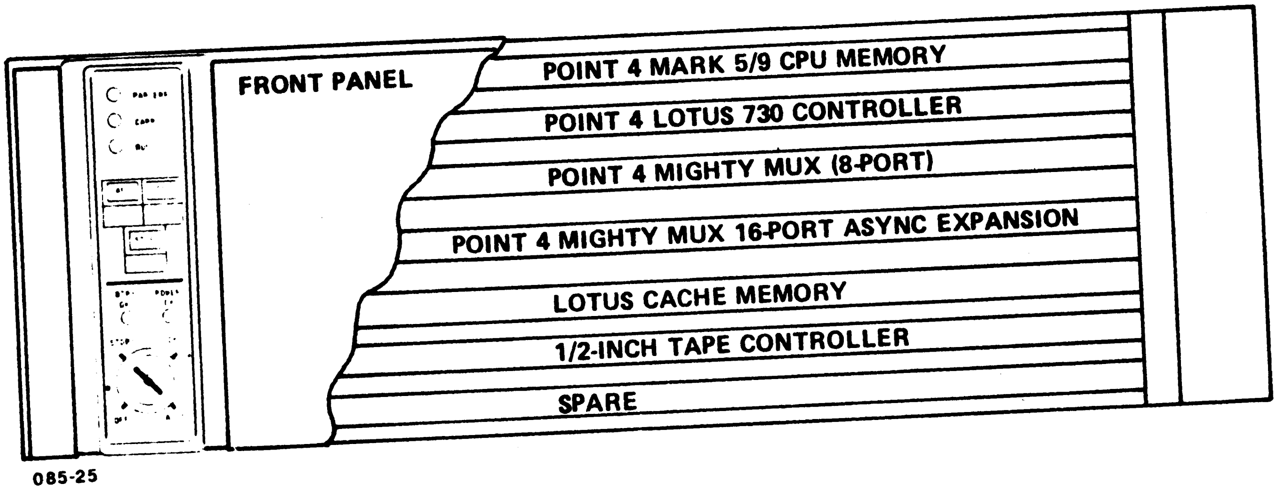

FRONT PANEL
POINT 4 MARK 5/9 CPU MEMORY
POINT 4 LOTUS 730 CONTROLLER
POINT 4 MIGHTY MUX (8-PORT)
POINT 4 MIGHTY MUX 16-PORT ASYNC EXPANSION
LOTUS CACHE MEMORY
1/2-INCH TAPE CONTROLLER
SPARE
085-25

Figure 2-4. Typical Computer Chassis Configuration

### 2.3.4 POWER SUPPLY CABLING

The POINT 4 MARK 5/9 processor chassis comes with a 6-foot cable assembly attached to the rear of the chassis. This cable assembly consists of a ribbon cable and a wire bundle strapped together. Processor chassis connections are made at the factory. Check to ensure that these cables are properly connected to the chassis. Figure 2-5 illustrates mounting positions on the processor chassis. Connectors and their mounting positions are:

| Connector | Mounting Position |
|---|---|
| 12-pin wire bundle (main DC power to chassis) | Top center of the POINT 4 MARK 5/9 backplane |
| 26-pin ribbon cable (Mini-Panel and control cable) | Upper right side of the POINT 4 MARK 5/9 backplane |
| 3 sheathed power wires (AC fan power) | Connected inside of the fan module |

The free ends of the cable assembly must be connected to the rear of the power supply. Figure 2-6 illustrates mounting positions on the power supply chassis. The connectors and their mounting positions are as follows:

| Connector | Mounting Position |
|---|---|
| 25-pin ribbon cable (power supply control cable) | Rear upper left-hand corner of power supply chassis |
| 15-pin wire bundle (main DC power and (AC fan power) | Rear lower left-hand corner of power supply chassis |

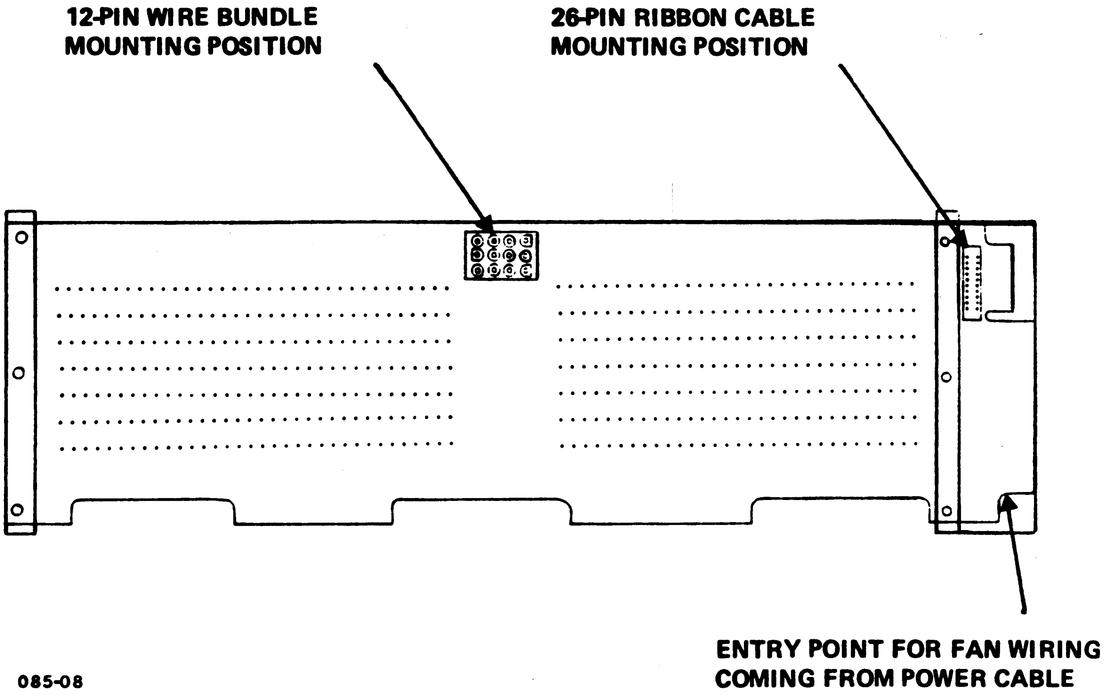
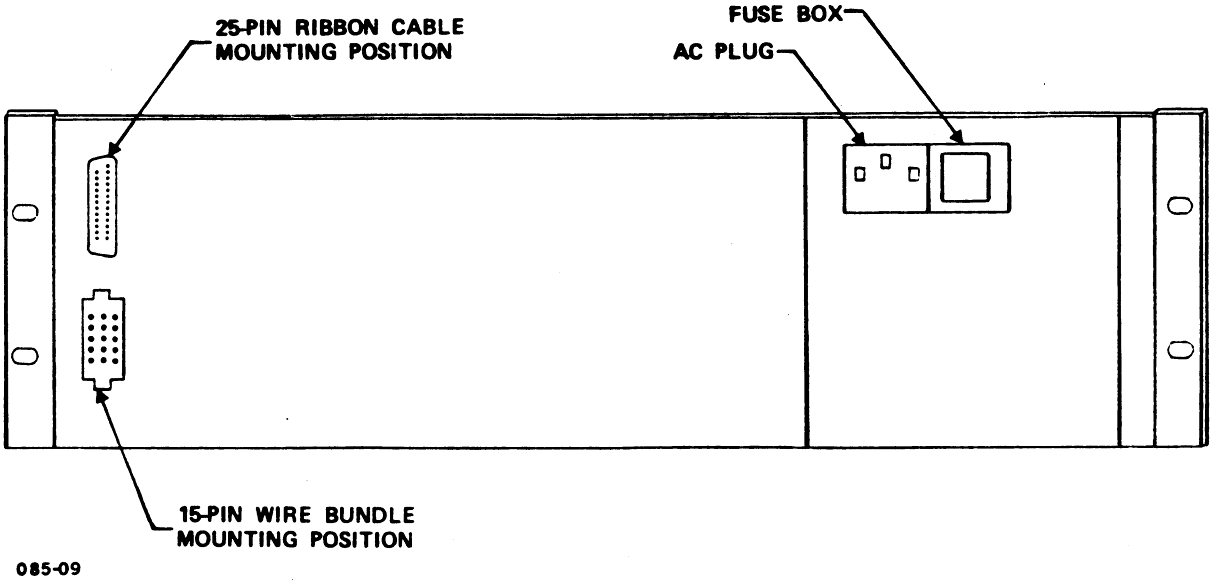

12-PIN WIRE BUNDLE MOUNTING POSITION
26-PIN RIBBON CABLE MOUNTING POSITION
ENTRY POINT FOR FAN WIRING COMING FROM POWER CABLE
085-08

**Figure 2-5. Processor Chassis Backplane Connector Mounting Positions**



25-PIN RIBBON CABLE MOUNTING POSITION
FUSE BOX
AC PLUG
15-PIN WIRE BUNDLE MOUNTING POSITION
085-09

**Figure 2-6. Power Supply Chassis Rear Connector Mounting Positions**

## 2.4 POWERING UP THE SYSTEM

The following steps should be followed when first supplying power to the POINT 4 MARK 5/9 power supply and processor:

1. Connect the red lead to the plus terminal on the battery (removed for shipment).

2. Turn the processor Mini-Panel keyswitch to the OFF position before connecting the 6-foot AC power cable to the AC power source. See Section 3.3 for processor Mini-Panel switch positions.

3. Plug the AC power cable into the AC socket. The AC IN LED on the power supply indicator panel should be illuminated. See Section 3.2 for power supply indicator panel LED positions.

4. If the AC IN LED does not illuminate, check the following:

   a. The AC power is properly connected.

   b. The power control cable is properly connected to the processor chassis (see Section 2.3.4 for cable connection instructions).

   c. The fuse in the power supply fuse box is installed and good.

   If the AC IN LED still does not illuminate, do not proceed. Return the power supply for repair.

5. Turn the Mini-Panel keyswitch to the standby (STDBY) position. If the power supply contains battery backup, the +5 BU, the -5 BU, and the +12 BU LEDs should be illuminated. The BTRY OK light may blink; this indicates that the battery is charging. The BTRY OK light is continuously illuminated when the battery is fully charged. See Section 3.2 for positions of LEDs on the power supply indicator panel.

6. Turn the Mini-Panel keyswitch to the ON position. The POWER OK LED on the processor Mini-Panel should illuminate. This indicates that power is available to the processor chassis. The power supply panel LEDs should all be illuminated. This indicates that all voltages are in tolerance. If any LED on the power supply panel does not illuminate, one of the following conditions exists:

   - The power supply for that voltage is out of tolerance
   - The LED is bad
   - Loose cable plug

### 2.4.1 GROUNDING

When the logic ground of a device is connected to the logic ground of a POINT 4 MARK 5/9 chassis, the device frame chassis ground should be isolated from the device logic ground. The device frame ground and the chassis should be tied together by at least a 0.5-inch braided copper strap. The chassis grounds should be isolated from the logic grounds to prevent a pseudo power failure. The POINT 4 MARK 5/9 logic ground is isolated from the frame ground by a 10-ohm, 2-watt resistor at the backplane motherboard.

### 2.4.2 BATTERY SHELF-LIFE

The red lead to the battery is removed from the battery pack prior to shipment. Check the power supply to confirm that the red lead is connected to the positive terminal on the battery. When powering up the system, the BTRY OK light may blink; this indicates that the battery is charging. The blinking stops when the battery is fully charged. When storing the power supply for more than a week, remove the red lead on the battery to increase its shelf-life.

## 2.5 DIAGNOSTIC CHECKS

The POINT 4 MARK 5/9 CPU has a comprehensive, built-in Self-Test diagnostic program contained in PROMs (Programmable Read-Only Memory).

The Self-Test diagnostic may be used either as a one-pass hardware verifier, or as a continuous reliability test. It contains the following tests:

1. Halt instruction test
2. Compare instruction test
3. ALU and data bus test
4. ALU source operand test
5. Exhaustive ALU instruction test
6. Page 0 and base 3 addressing modes test
7. Relative, base 2, and indirect addressing modes test
8. Auto-indexing test
9. Limited I/O instruction test
10. Multilevel indirect addressing test
11. 64K-word addressing capability test
12. Worst-case test of all memory locations

The following subsections provide general procedures for starting the POINT 4 MARK 5/9 Self-Test. See the POINT 4 (MARK 5) Diagnostic Manual for details on tests executed, expected results, and halt interpretations.

Software diagnostic programs are available for the MARK 5 with EIS option and for the MARK 9. Refer to the MARK 8 (MARK 5 with EIS) Verifier/Diagnostic Document and the MARK 9 Diagnostics Document.

### 2.5.1 NORMAL SELF-TEST OPERATION

1. Snap off the front panel.

2. Locate the Self-Test switch on the front edge of the CPU board. See Figure 2-7 for Self-Test switch location. Press STOP. Press the Self-Test switch while holding the APL switch on the processor Mini-Panel. This loads the Self-Test program into main memory.

3. The first test verifies operation of the halt instruction. Press the CONTinue switch on the processor Mini-Panel to continue execution of Self-Test.

4. After a few preliminary tests, the message POINT 4 CPU SELF-TEST is displayed on the master terminal. If a master terminal is not used, the CARRY indicator will blink with every Self-Test pass.

5. The message 64K CPU OK is displayed on the master terminal after testing all instructions, addressing modes, and basic CPU operations.

6. The message MEMORY OK is displayed on the master terminal after completion of the worst-case memory test.

7. A V is displayed on the master terminal each time Self-Test repeats.

8. Replace front panel.

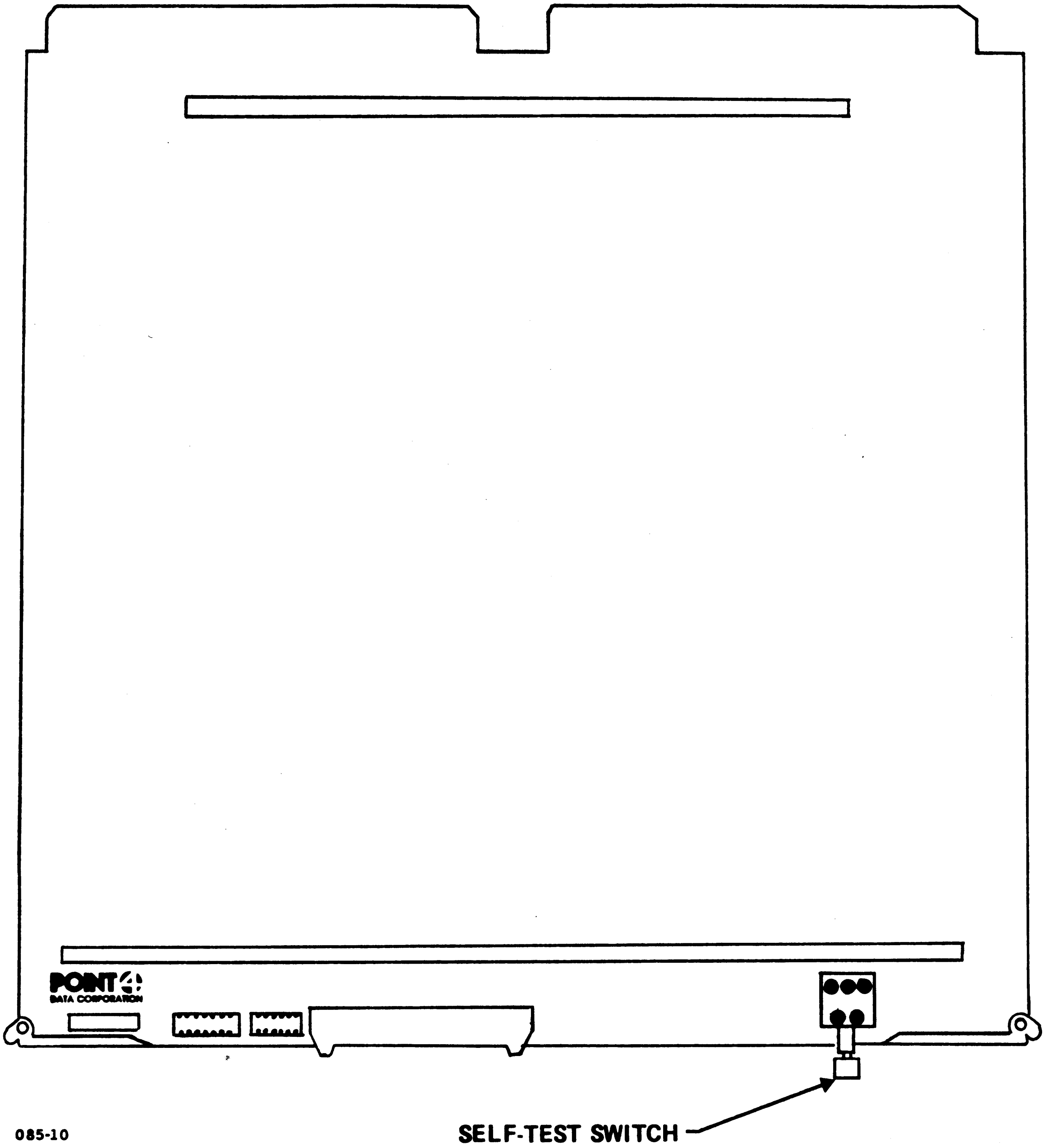


**Figure 2-7. POINT 4 MARK 5/9 Self-Test Switch Location**

### 2.5.2 HOW TO INTERPRET A HALT

An error is indicated if Self-Test halts (run light goes out) after the initial halt. Refer to the POINT 4 (MARK 5) Diagnostic Manual for detailed information on each step of the Self-Test diagnostic programs, program listings, and interpretations of halts at various locations.

### 2.5.3 SELF-TEST AS A CONTINUOUS RELIABILITY TEST

As previously noted, Self-Test may be used either as a one-pass hardware verifier or as a continuous reliability test.

- If a one-pass execution is desired, wait until the first V appears, then press the STOP switch.
- If a continuous test is desired, do not press the STOP switch. The Self-Test program will continue running until the STOP switch is pressed, or an error is detected.

## 2.6 UNIVERSAL CPU BOARD INSTALLATION

The POINT 4 MARK 5 or MARK 9 universal CPU board is designed to replace CPU and memory boards in a Nova-compatible chassis.

The POINT 4 MARK 5 or MARK 9 universal CPU board can be installed in a single I/O slot of any Nova-type chassis. However, in order to accomplish this, certain requirements and restrictions must be met.

### 2.6.1 UNIVERSAL CPU BOARD VOLTAGE REQUIREMENTS

A POINT 4 MARK 5 universal CPU board requires three voltages for operation: +5V, -5V, and +12V. A POINT 4 MARK 9 universal CPU board requires only +5V for operation. Voltages and backplane pin assignments are listed below:

| Voltage | Current | Backplane Pin |
|---|---|---|
| +5V (regulated) | 4A | A3,A4,B3,B4 |
| -5V (regulated)* | 10mA* | A6* |
| +12V (regulated)* | 1.25A* | A10* |

The +5V and -5V requirements above are standard on a Nova-type chassis. However, a typical Nova-type chassis has +15V unregulated supply at pin A10. Therefore, the POINT 4 voltage regulator option (see Section 6.4) is required for MARK 5 Universal CPU boards. This option converts the unregulated supply to +12V of regulated voltage.

The POINT 4 voltage regulator option is not required for MARK 9 universal CPU boards.

*Not used by MARK 9 CPUs.

### 2.6.2 RESERVED BACKPLANE PINS

The POINT 4 MARK 5/9 CPU board communicates with its chassis Mini-Panel through a set of reserved backplane pin connections. These backplane pins must remain unconnected when using a POINT 4 MARK 5/9 universal CPU board in a Nova-type chassis. These pins are not normally used in the Nova-type chassis. Any system that does not contain a specially bused backplane will experience no problems in this area. Verify that a chassis does not use the reserved backplane connections listed below.

| Signal Name | Backplane Pin |
|---|---|
| PEL- | A84 |
| RUNL- | A86 |
| CL- | A88 |
| CONT- | A90 |
| SELF- | A91 |
| STOP- | A92 |

### 2.6.3 UNIVERSAL CPU BOARD DATA CHANNEL AND INTERRUPT PRIORITY CHAINS

Data channel and interrupt priority chains must be preserved after installation of the POINT 4 MARK 5/9 universal CPU board in the chassis.

Data channel and interrupt priority input (DCHPIN- and INTPIN-; backplane pins A96 and A94, respectively) of the highest priority device(s) are grounded (usually at slot 3) and the data channel/interrupt priority chain is then passed to the input of the next highest priority device. If no data channel/interrupt request exists in the highest priority device, the priority is passed to the next device in the chain.

The POINT 4 MARK 5/9 universal CPU does not pass on data channel or interrupt priorities. If it is necessary to insert the CPU board into a slot that physically interrupts the priority chain, the data channel and interrupt signals must be jumpered past the CPU board.

Standard-speed and high-speed data channel operations are described in Section 4.4. The POINT 4 MARK 5/9 universal CPU is normally delivered with the high-speed data channel enabled. To switch to the standard-speed data channel, ground pin A93 on the CPU backplane slot.

## 2.7 OPERATOR CONTROL UNIT INSTALLATION

The optional Operator Control Unit (OCU) enhances operator access to the CPU. There are two methods of installing the OCU:

1. Attachment to the processor front panel
2. Connection via ribbon cable

### 2.7.1 ATTACHMENT TO THE PROCESSOR FRONT PANEL

The Operator Control Unit can be attached to the center of the processor front panel for location with the processor. The unit has an edge connector protruding from the upper rear of the unit. This connector inserts directly into the slot provided on the center of the front panel, mating with a socket behind the front panel. This socket connects via ribbon cable to a socket on the front edge of the CPU board. See Figure 2-8 for an illustration of this attachment.

### 2.7.2 CONNECTION VIA RIBBON CABLE

The Operator Control Unit can be extended via ribbon cable to any convenient work surface. Extension is via a six-foot ribbon cable. The connector on the rear of the Operator Control Unit attaches to a socket on one end of the ribbon cable. The ribbon cable threads through a slot on the lower edge of the center of the front panel and the other connector connects to the socket on the front edge of the CPU board. See Figure 2-9 for an illustration of the remote connection.

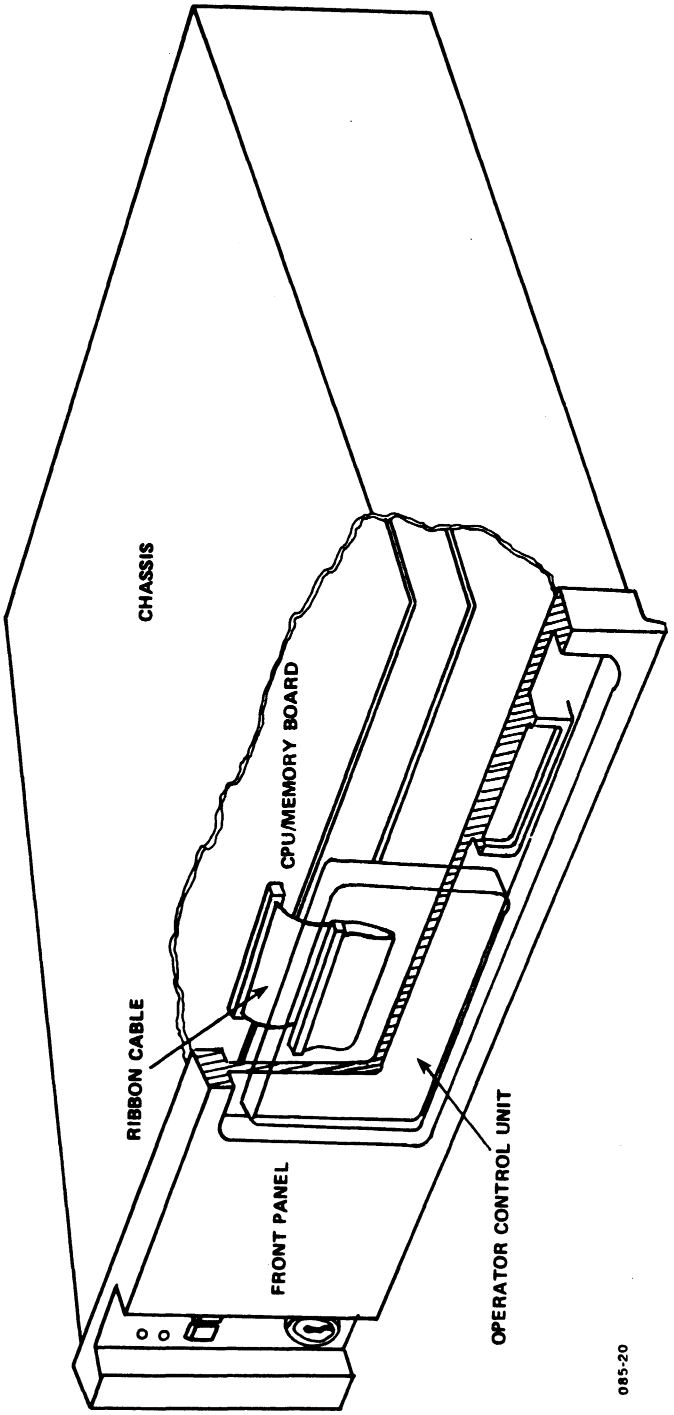


Figure 2-8. Operator Control Unit Attachment to Front Panel

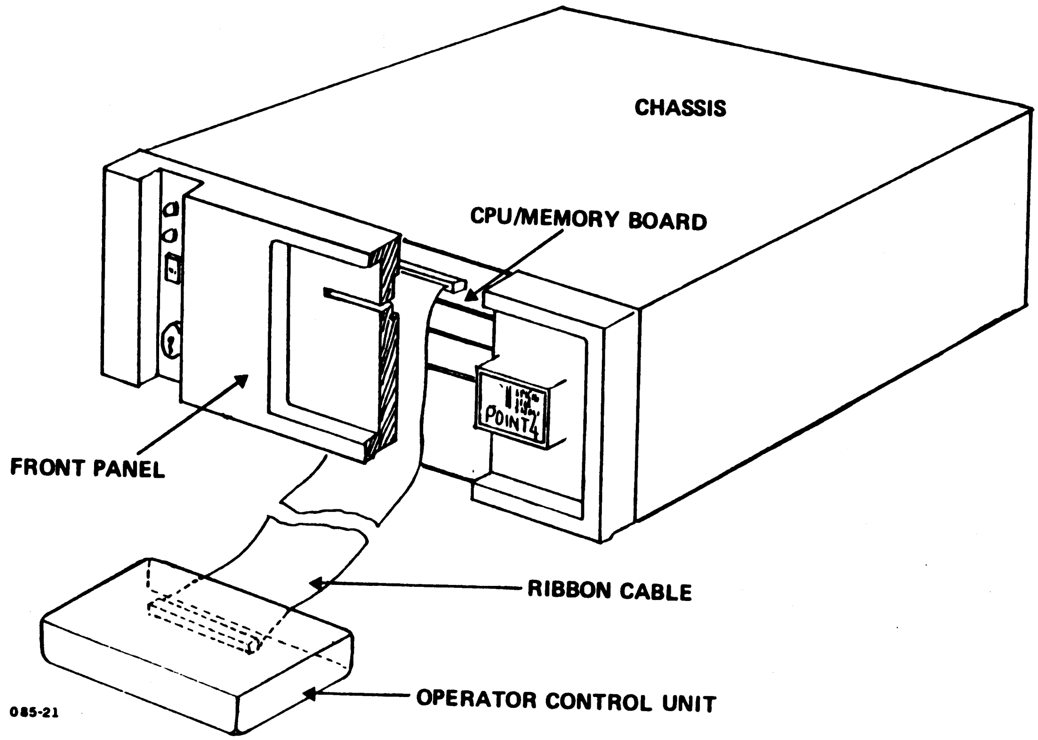


**Figure 2-9. Operator Control Unit Remote Attachment**

# Section 3
# OPERATING PROCEDURES

## 3.1 INTRODUCTION

This section describes the capabilities and operating procedures for the power supply indicator panel and operation control units. Three types of control units exist for the POINT 4 MARK 5/9:

- Processor Mini-Panel
- Detachable Operator Control Unit (optional)
- Virtual Control Panel

Specific procedures for performing common types of operations are provided. The controls and indicators are also described.

## 3.2 POWER SUPPLY INDICATOR PANEL

The POINT 4 MARK 5 or MARK 9 power supply chassis houses a set of light-emitting diode indicators. These indicators monitor power supply voltages and battery backup voltages. The power supply indicator panel is located on the left-hand side of the power supply chassis. Figure 3-1 is an illustration of the power supply indicator panel. Table 3-1 explains the meaning of illuminated power supply panel indicators.

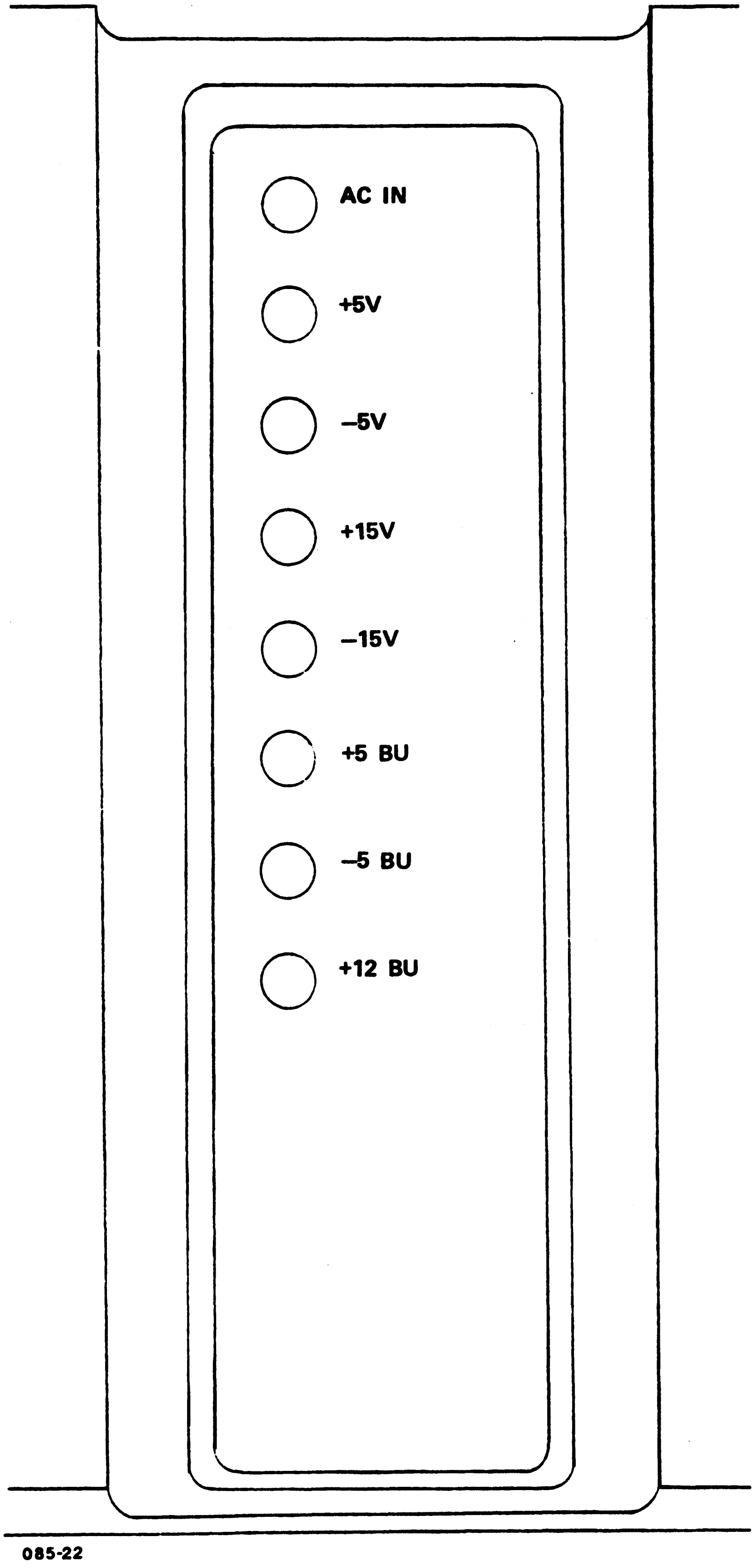


**Figure 3-1. Power Supply Indicator Panel**

TABLE 3-1. POWER SUPPLY PANEL INDICATORS

| Indicator | Meaning |
|---|---|
| AC IN | Indicates that AC power has been applied to the power supply unit. |
| +5V | The +5V output voltage is in tolerance. This output voltage is available for user applications. |
| -5V | The -5V output voltage is in tolerance. This output voltage is available for user applications. |
| +15V | The +15V output voltage is in tolerance. This output voltage is available for user applications. |
| -15V | The -15V output voltage is in tolerance. This output voltage is available for user applications. |
| +5 BU | The +5V battery backup supply is in tolerance. If the battery backup unit is not installed, this light has the same meaning as the +5V light. |
| -5 BU | The -5V battery backup supply is in tolerance. This output voltage is available only to the POINT 4 MARK 5 CPU/memory board. If the battery backup unit is not installed, this light has the same meaning as the -5V light. |
| +12 BU | The +12V battery backup supply is in tolerance. This output voltage is available only to the POINT 4 MARK 5 CPU memory board. The +12V supply is operational with or without the battery backup unit. |

## 3.3 PROCESSOR MINI-PANEL

The POINT 4 MARK 5/9 processor chassis houses essential controls and indicators for basic processor control functions. The processor Mini-Panel is located on the left-hand side of the chassis.

### 3.3.1 MINI-PANEL OPERATING FUNCTIONS

The three types of operating functions on the Mini-Panel are: power controls and indicators, processor monitoring indicators, and program execution controls. Figure 3-2 illustrates the processor Mini-Panel controls and indicators.

The POINT 4 MARK 5/9 universal CPU board provides the user with a series of switches and light-emitting diodes on the printed circuit board itself, which are normally found on the POINT 4 Mini-Panel. These include: an APL switch, a CONTinue switch, a STOP switch, a parity error indicator (MARK 5), a memory error indicator (MARK 9), a carry indicator, a run indicator, and a +5V power indicator.

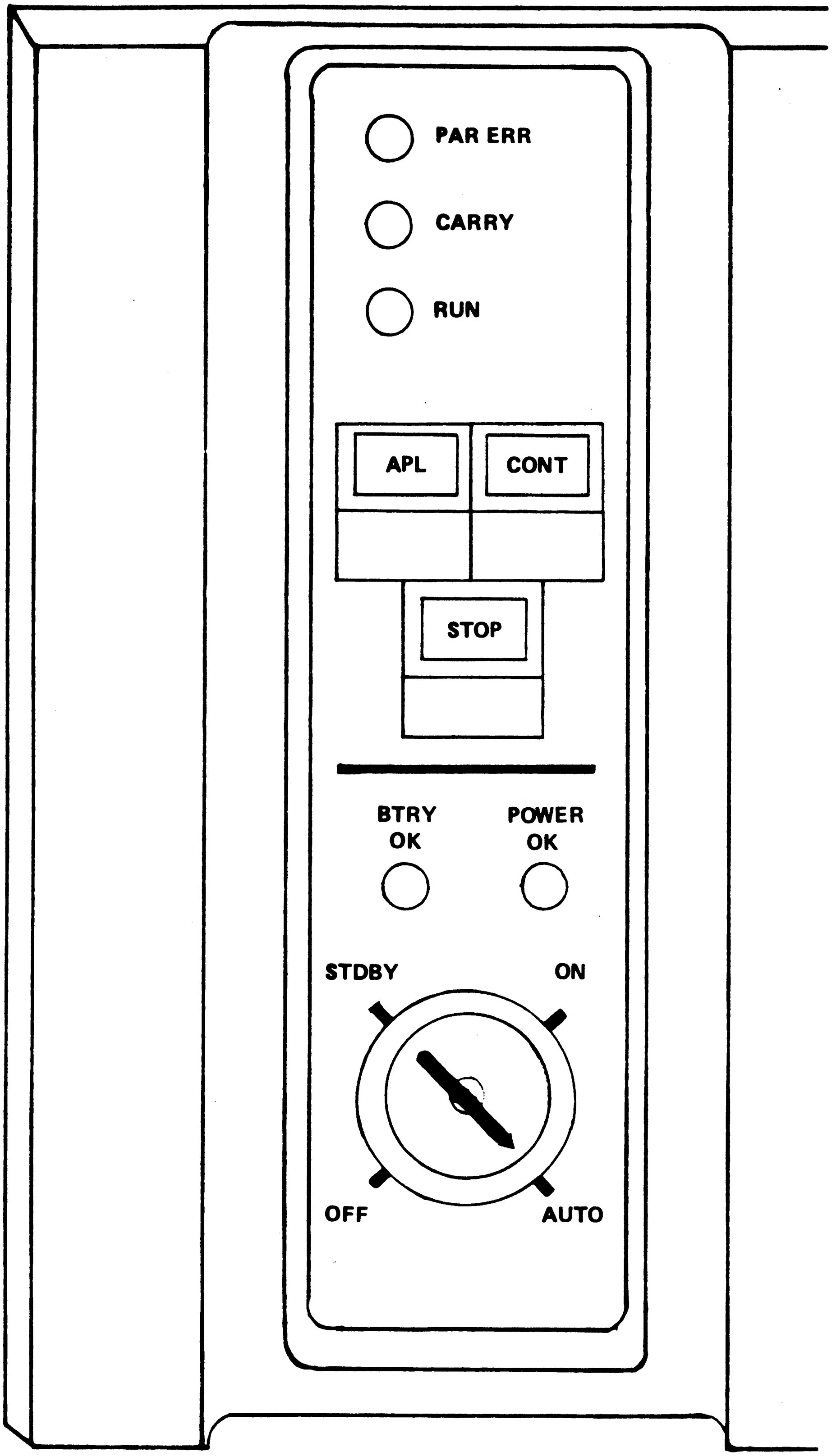


**Figure 3-2. POINT 4 MARK 5/9 Processor Mini-Panel**

### 3.3.1.1 Power Controls and Indicators

The lower half of the Mini-Panel is devoted to power control and monitoring.

#### 3.3.1.1.1 POWER CONTROLS

The power switch is controlled by a four-position, key-operated rotary switch. This switch controls the four functions described in Table 3-2.

**TABLE 3-2. PROCESSOR MINI-PANEL POWER CONTROL SWITCH SETTINGS**

| Switch Setting | Function |
|---|---|
| OFF | Disables the battery backup unit. This switch position is used for storage or extended power-off conditions. The battery charger is on as long as the power supply is plugged in. |
| STDBY | Turns the processor off, leaving the battery backup unit operational (if installed). Voltages for +5V and +15V are removed from the chassis but +5 BU, -5 BU, and +12 BU are maintained for CPU slot use only. |
| ON | Turns on power to the processor and places the Mini-Panel in the panel-on mode. In this mode, all controls and indicators on the Mini-Panel are enabled. The power-fail auto-restart feature (available with the battery backup option) is disabled, and the processor must be started manually whenever AC power is turned ON. |
| AUTO | Maintains processor power on and places the Mini-Panel in the panel-off mode. In this mode, all controls on the Mini-Panel are disabled, and all indicators remain active. Disabling the controls prevents interference with the operation of the CPU. The power-fail auto-restart feature is enabled if the battery backup option has been installed. This feature causes the processor to resume operation automatically upon AC power restoration following power failure. |

### 3.3.1.1.2 POWER INDICATOR INTERPRETATIONS WITHOUT BATTERY BACKUP

Light-emitting diode (LED) indicators illuminate to indicate an active state for AC power and battery backup power. Table 3-3 gives interpretations for the POWER OK LED for systems without the Battery Backup option.

**TABLE 3-3. POWER INDICATOR INTERPRETATIONS WITHOUT BATTERY BACKUP**

| Power Supply AC IN | Processor Chassis POWER OK | Interpretation |
|---|---|---|
| OFF | OFF | Power supply is not connected to AC. |
| ON | OFF | Power supply is operational but power is not available to processor chassis. If the keyswitch is in ON or AUTO position, it indicates that one of the power supply voltages is out of tolerance (see Section 3.2, Power Supply Indicator Panel) or that there is a problem in the cabling between the processor and the power supply. |
| ON | ON | All power supply voltages are in tolerance and available to the processor chassis. |

### 3.3.1.1.3 POWER INDICATOR INTERPRETATIONS WITH BATTERY BACKUP

The BTRY OK LED is operational if the Battery Backup option has been installed in the power supply chassis. The meaning of the BTRY OK LED depends on the state of the AC IN LED. These interpretations are shown in Table 3-4.

Figure 3-2 illustrates the positions on the key-operated rotary switch, the POWER OK indicator, and the BTRY OK indicator.

**TABLE 3-4. POWER INDICATOR INTERPRETATIONS WITH BATTERY BACKUP**

| Power Supply AC IN | Processor Chassis BTRY OK | Interpretation |
|---|---|---|
| OFF | OFF | AC power source is off and the battery backup unit has been fully discharged. |
| OFF | ON | The battery backup unit is supplying +5V BU, -5V BU (MARK 5) and +12V BU (MARK 5) to the CPU/memory board to maintain the contents of memory. |
| ON | OFF | The battery backup unit is being recharged but is not yet fully charged. The LED indicator blinks as the batteries are being charged. CPU is not functional. |
| ON | ON | The battery backup unit batteries are fully charged. The CPU is functional. |

### 3.3.2 PROCESSOR OPERATION MONITORING INDICATORS

The Mini-Panel has three LED indicators in addition to the power monitoring indicators. These indicators perform the functions shown in Table 3-5. (See Figure 3-2 for the locations of the operation monitoring LED indicators.)

TABLE 3-5. PROCESSOR OPERATION MONITORING INDICATORS

| Indicator | Function |
|---|---|
| PAR ERR (MARK 5) | Indicates that a parity error has occurred during a memory-read operation. The LED indicates that the processor has come to a halt pending operator action. Pressing the CONTinue or APL switch causes the processor to resume operation and turns off the parity error light. This LED is operational only if the parity error option is installed. |
| PAR ERR (MEM ERR MARK 9) | Indicates type of memory error. This LED blinks when the EDAC circuitry detects and corrects single-bit errors. Double-bit errors cause continuous illumination of this LED. This LED is off when no errors occur. |
| CARRY | Indicates the current state of the processor Carry flag. The LED illuminates when the Carry flag is set to a 1. |
| RUN | Indicates that the processor is in normal operation (i.e., executing one instruction after another). When the processor is stopped, the LED goes off. |

### 3.3.3 PROGRAM EXECUTION CONTROLS

Three momentary contact switches are available to control program execution in the processor. These switches are enabled in the panel-on mode (keyswitch set to the ON position). They are disabled in the panel-off mode (keyswitch set to the AUTO position). Table 3-6 describes the switches and their functions. (See Figure 3-2 for the locations of the program execution controls.)

**TABLE 3-6. PROGRAM EXECUTION CONTROLS**

| Switch | Function |
|---|---|
| STOP | Stops processor operation before executing the next instruction. The processor finishes current instruction, fetches the next instruction and then stops. The program counter (PC) then points to the next instruction to be executed.<br>**NOTE**<br>This note applies to 32K-word addressing mode only. If the processor is caught in an infinite indirect addressing loop that prevents completion of the instruction, the STOP control will not work. Press APL to stop processor and load MANIP for debugging (see Section 3.5 for use of MANIP). |
| CONT | Causes program execution to resume, starting at address contained in the PC. |
| APL | The automatic program load (APL) switch performs these functions:<br>1. Loads contents of an octal debugger/manipulator PROM (MANIP) into the top 1000 (octal) words of memory. The debugger/manipulator is used for access to accumulators and memory. Allows examination and deposit of data for operation monitoring and control. Optionally allows loading of system software from disk or other DMA devices. See Section 3.5 for debugger/manipulator program commands. Note that if CPU board switches are set to 2xx (octal), the CPU performs an automatic APL from device xx when the APL switch is pressed.<br>2. May also be used in combination with the Self-Test switch (located on front edge of CPU circuit board) to load contents of Self-Test PROM into memory. The Self-Test program performs a complete check of hardware functions and executes a worst-case memory test. Self-Test does not examine MARK 9 map. It can be used either as a hardware verifier or as a continuous reliability test. See Section 2.5 for description of self-test diagnostics.<br>If the CPU is running, press STOP before pressing APL. However, if the processor is performing a multilevel indirect addressing instruction, pressing APL causes the processor to stop without use of STOP switch. |

## 3.4 OPERATOR CONTROL UNIT (OPTIONAL)

In addition to the basic controls and indicators on the POINT 4 MARK 5/9 processor chassis, an Operator Control Unit (OCU) is available that enhances operator access to the processor. This detachable control unit can be extended via ribbon cable to any convenient work surface. The compact control unit contains all switches and indicators necessary to monitor and control the processor. See Figure 3-3 for an illustration of the Operator Control Unit.

The Operator Control Unit measures 6.12 inches wide, 3.62 inches high and 1.10 inches deep. It may be mounted on the front panel at the slot provided in the center of the front panel, or extended to a convenient work surface by mounting a 6-foot ribbon cable to the processor PC board. Instructions for attaching the Operator Control Unit to the processor chassis are provided in Section 6.3.

### 3.4.1 OPERATOR CONTROL UNIT CAPABILITIES

The Operator Control Unit is designed to aid in detecting possible problems in the system, debugging these problems, and for entering program and data changes into the system. The operator can monitor activity on the system via eight light-emitting diode indicators. These indicators monitor the following processor functions: data channel, high-speed interprocessor bus (not used by POINT 4 equipment), 64K-word addressing, program execution, interrupt enabling, carry condition, and parity error (memory error - MARK 9) detection. Octal displays allow the operator to observe the contents of memory and accumulators as well as the entries made via the data entry switches.

Switches provide the operator with the ability to enter data, access memory to examine and deposit into it, access accumulators to examine and deposit into them, enable 64K-word addressing, and control program execution. In addition, an APL switch is provided.

### 3.4.2 CPU STATE INDICATORS

Eight light-emitting diode (LED) indicators are used to monitor processor operation, illuminating when the function is active. Table 3-7 describes the function of each LED indicator.

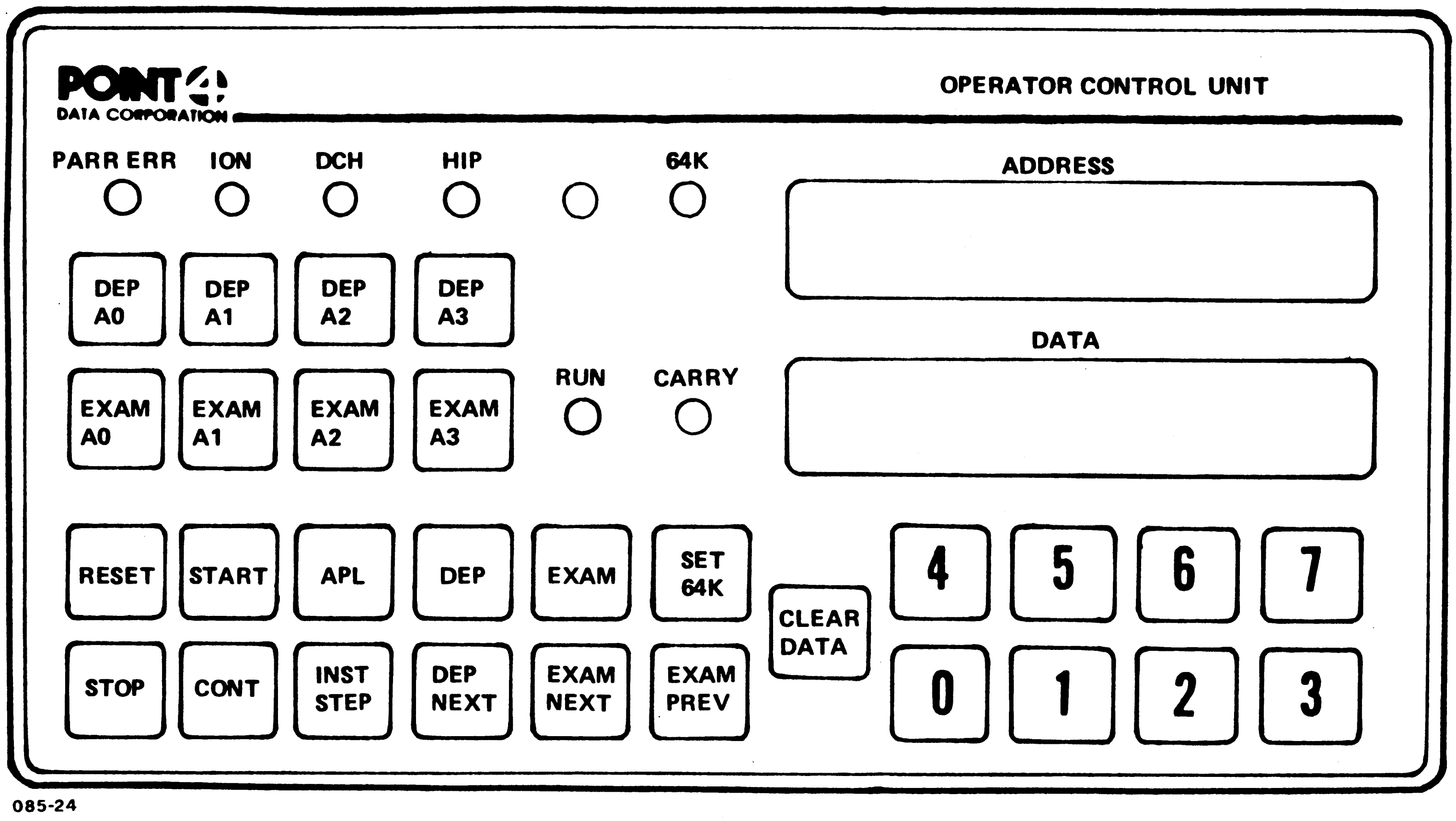


Figure 3-3. Operator Control Unit Controls and Indicators

TABLE 3-7. CPU OPERATION INDICATORS

| LED Indicator | Function |
|---|---|
| PAR ERR | Illuminates when a parity error has occurred during a memory-read operation. The indicator is enabled only when the parity error detection option has been installed. Light indicates that the processor has come to a halt pending operator action. RESET must be pressed to enable use of the other control unit switches when a parity error has occurred. Pressing CONTinue will cause the processor to resume operation, regardless of previous parity error. Error may be investigated and/or corrected through use of examine and deposit capabilities of the Operator Control Unit.<br>**NOTE**<br>The MARK 9 CPU uses the OCU PAR ERR LED to indicate memory errors. The OCU PAR ERR LED blinks when the EDAC circuitry detects and corrects single-bit errors. Double-bit errors cause continuous illumination of this LED. This LED is off when no errors occur. |
| ION | Interrupts are enabled. Indicates that the processor will respond to interrupt requests from peripheral devices. |
| DCH | Data channel is active. Indicates that a data channel direct memory access transfer is currently taking place. |
| HIP | Not used. |
| 64K | Indicates that 64K-word addressing mode is enabled. |
| RUN | Indicates that the processor is in normal operation (executing one instruction after another). When the processor is halted, this LED goes off. |
| CARRY | Indicates current state of processor Carry flag. Illuminated when Carry flag is set to 1. |

### 3.4.3 OCTAL DISPLAYS

Two separate six-digit octal displays are provided: one for address (ADDRESS) and one for data (DATA). These LED displays are located at the upper right-hand corner of the Operator Control Unit (see Figure 3-3). The contents of these displays are controlled by the examine switches and the data entry switches on the control unit.

The ADDRESS display is always equal to the program counter (PC). While in RUN, ADDRESS monitors the program by continuously displaying the program counter. When the processor is halted, ADDRESS displays the PC where execution was stopped. Any control unit operation in which an address is entered, incremented, or decremented (EXAM, EXAM NEXT, etc.) changes the PC and ADDRESS simultaneously.

### 3.4.4 DATA ENTRY AND PROCESSOR MANIPULATION CONTROLS

All data entry and processor control buttons on the Operator Control Unit are sealed, laminated membrane switches. These buttons are used to monitor and enter data into memory and accumulators and to control program execution. They serve as useful system monitoring and problem debugging tools for system programmers.

Eight membrane switches are provided for data entry in octal form. These buttons are numbered 0, 1, 2, 3, 4, 5, 6, and 7, enabling the operator to make six-digit octal entries for deposit in accumulators or memory and entry of memory addresses. The octal digits entered are shifted into the DATA display. The destination of the octal entry is controlled by the accumulator deposit buttons for the accumulators, by EXAM for memory addresses and by DEPosit and DEPosit NEXT for memory data. The number in the DATA display may be read by the CPU via use of a READS ac,CPU instruction.

The 16-bit word is divided into six octal digits. The most significant bit of the 16 bits forms the most significant octal digit. Since this digit consists of only one bit, it can have the values 0 and 1 only. Therefore, if a value greater than 1 is shifted into the most significant bit, that value will be truncated to a 1-bit number as follows:

```
any even number --> 0

any odd number  --> 1
```

A CLEAR DATA button is provided to clear the DATA display to 000000. Corrections may also be made by entering zeros followed by the correct octal digits. The most significant digit will be shifted off the left end of the display as new digits are entered. (See Figure 3-3 for positions of the data entry and CLEAR DATA buttons.)

Processor access and program execution controls consist of 20 sealed membrane switches. These controls can be grouped by basic functions into three types: memory access, accumulator access and program execution. (See Figure 3-3 for positions of these controls on the Operator Control Unit.)

Memory access controls (disabled while the processor is in RUN mode) are described in Table 3-8.

## TABLE 3-8. OCU MEMORY ACCESS CONTROLS

| Control | Function |
|---|---|
| EXAM | Examine - the octal value in the DATA display is moved into the ADDRESS display. The contents of memory at the memory location in the ADDRESS display is then displayed in the DATA display. The PC is set equal to the ADDRESS display. Note that pressing EXAM twice corresponds to indirect addressing. |
| EXAM<br>NEXT | Examine next - the address in the ADDRESS display is incremented by 1 and the content of the new memory address is displayed in the DATA display. The PC is set equal to the incremented memory address, which is displayed in the ADDRESS display. |
| EXAM<br>PREV | Examine previous - the address in the ADDRESS display is decremented by 1 and the content of the new memory address is displayed in the octal DATA display. The PC is set equal to the decremented memory address, which is displayed in the ADDRESS display. |
| DEP | Deposit - deposits the value in the DATA display into the memory address displayed in the ADDRESS display. The value in DATA may be a value read from memory, or an accumulator, or a value entered via the data entry buttons. The PC is left equal to the ADDRESS display. |
| DEP<br>NEXT | Deposit next - the address in the ADDRESS display is incremented by 1 and the value in the DATA display is deposited into the incremented memory address. The value in DATA may be a value read from memory, or an accumulator, or a value entered via the data entry buttons. The PC is set equal to the incremented address displayed in ADDRESS. |
| SET<br>64K | Enable 64K-word addressing - this control enables addresses 100000 through 177777. Memory addresses are 16 bits long for 64K-word addressing (instead of 15 bits long as in 32K-word addressing). In 64K-word addressing mode, multilevel indirect addressing is not permitted. |

Access to accumulators 0-3 is available through the examine and deposit controls (disabled when the processor is in the RUN mode) described in Table 3-9.

### TABLE 3-9. OCU ACCUMULATOR ACCESS CONTROLS

| Control | Function |
|---|---|
| EXAM<br>A0 | Examine accumulator 0 - displays, in the DATA display, the contents of accumulator 0 (A0). |
| EXAM<br>A1 | Examine accumulator 1 - displays, in the DATA display, the contents of accumulator 1 (A1). |
| EXAM<br>A2 | Examine accumulator 2 - displays, in the DATA display, the contents of accumulator 2 (A2). |
| EXAM<br>A3 | Examine accumulator 3 - displays, in the DATA display, the contents of accumulator 3 (A3). |
| DEP<br>A0 | Deposit in accumulator 0 - deposits in accumulator 0 (A0) the value in the DATA display. The value in DATA may be a value read from memory or an accumulator, or a value entered via the data entry buttons. |
| DEP<br>A1 | Deposit in accumulator 1 - deposits in accumulator 1 (A1) the value in the DATA display. The value in DATA may be a value read from memory or an accumulator, or a value entered via the data entry buttons. |
| DEP<br>A2 | Deposit in accumulator 2 - deposits in accumulator 2 (A2) the value in the DATA display. The value in DATA may be a value read from memory or an accumulator, or a value entered via the data entry buttons. |
| DEP<br>A3 | Deposit in accumulator 3 - deposits in accumulator 3 (A3) the value in the DATA display. The value in DATA may be a value read from memory or an accumulator, or a value entered via the data entry buttons. |

The remaining six controls are used for control of program execution. They are described in Table 3-10.

## TABLE 3-10. OCU PROGRAM EXECUTION CONTROLS

<table>
<tr><th>Control</th><th>Function</th></tr>
<tr><td>RESET</td><td>Resets all I/O devices on system to idle state and clears processor ION flag and 64K-addressing mode.</td></tr>
<tr><td>STOP</td><td>Stops processor operation before executing the next instruction. The processor finishes current instruction, fetches next instruction, and then stops. The program counter and ADDRESS display point to next instruction to be executed. Contents of the DATA display are not changed.<br><br><b>NOTE</b><br><br>If the processor is caught in an infinite indirect addressing loop that prevents completion of the instruction, STOP control will not work. Press RESET to stop processor.</td></tr>
<tr><td>START</td><td>Moves contents of DATA display to the PC. Processor then begins normal operation by fetching and executing instruction at address just loaded into the PC. RUN light will illuminate.</td></tr>
<tr><td>CONT</td><td>Causes program execution to resume, starting at address contained in the PC (and in ADDRESS display).</td></tr>
<tr><td>INST<br>STEP</td><td>Causes processor to execute one instruction cycle beginning at address in ADDRESS display. When instruction cycle is completed, processor stops. ADDRESS display contains new value of program counter. Information displayed in DATA display depends on type of instruction:<br><br>
<table>
<tr><th><u>Instruction</u></th><th><u>Data Displayed</u></th></tr>
<tr><td>LDA</td><td>Data fetched from memory</td></tr>
<tr><td>STA</td><td>Data written into memory</td></tr>
<tr><td>ISZ, DSZ<br>JMP or JSR</td><td>Octal value of next instruction</td></tr>
<tr><td>Arithmetic/<br>Logic</td><td>Operand in destination accumulator after instruction execution is complete</td></tr>
<tr><td>Input/<br>Output</td><td>Data transferred (except in skip instructions, which displays the data in A0)</td></tr>
</table>
</td></tr>
</table>

TABLE 3-10. OCU PROGRAM EXECUTION CONTROLS (Cont)

| Control | Function |
|---|---|
| APL | The automatic progam load (APL) switch performs two separate functions:<br>1. Loads contents of an octal debugger/manipulator PROM into the top 1000 words of memory. The debugger/manipulator serves as a virtual control panel and operates through the master terminal. It offers several higher level functions than the Operator Control Unit, such as memory searches, moves, byte addressing, virtual addressing, etc. It also allows loading of system software from disk or other DMA devices. See Section 3.5 for a description of debugger/manipulator commands.<br>2. May also be used in combination with the Self-Test switch (located on front edge of CPU circuit board) to load contents of the Self-Test PROM into memory. The Self-Test program performs a complete check of hardware functions and executes a worst-case memory test. It can be used either as a hardware verifier or as a continuous reliability test. See Section 2.5 for description of Self-Test operation.<br>If the CPU is running, press STOP before pressing APL. |

## 3.5 VIRTUAL CONTROL PANEL

The POINT 4 MARK 5/9 has the ability to perform many front panel operations plus many extra system monitoring functions from a master terminal. This feature is designed for use by programmers to debug system problems and to manipulate the contents of registers and memory. The feature is implemented in a stand-alone program called MANIP, which is loaded into RAM from a PROM when the APL switch is pressed. Note that if CPU board switches are set to 2xx (octal), MANIP cannot be accessed from memory. Pressing APL causes MANIP to program load from device xx.

MANIP is a simple, but powerful, position-independent memory manipulator and debug package. MANIP occupies only 1000 (octal) words of memory.* All operations are executed by typing one letter followed by octal parameters as required (except a colon (:), which is also preceded by an octal parameter) and ending with a <RETURN>.

Table 3-11 shows the functions provided by MANIP (the number in parentheses indicates the number of parameters required for that particular function). The MANIP functions are described in detail in the following subsections.

MANIP is initially loaded into locations 177000 through 177777.

Location 177000 is reserved for saving the initial value of the program counter (PC); that is, the value of PC where the CPU had halted before MANIP was started. MANIP may be moved at any time by use of its MOVE (M) instruction.

The carry light flashes while MANIP is waiting for an input character to be entered (except in I mode). This is a signal that MANIP is active and will respond to input.

If an error is made while entering control information, it may be corrected in two ways.

1. Press <ESC> (or any other control character except <RETURN>) to delete the type-in and enable a new type-in.

2. If the error was made in entering an octal value, type a number of zeros followed by the correct octal number, as MANIP uses only the last six octal digits typed in for the octal word.

*For those who are familiar with POINT 4 Data Corporation's IRIS Operating System, MANIP is comparable to DBUG. The main differences are that MANIP does not have (1) symbolic capability, (2) breakpoints or trace, (3) disk read or write, and (4) CTRL-H/CTRL-A (backspace) capability. MANIP occupies only 1000 (octal) words of memory, while DBUG occupies 3400 (octal) words of memory.

### TABLE 3-11. MANIP COMMAND FUNCTIONS AND PARAMETERS

| Code | Function | Parameters Required |
|---|---|---|
| A | Type initial PC, accumulators, and carry flip-flop | (0) |
| C | Change accumulator or carry flip-flop | (2) |
| D | Dump (octal, word or byte) | (1 or 2) |
| E | Enter octal into sequential locations | (1 or 2) |
| F | Set up an address offset* | (0, 1, or 2) |
| J | Jump with accumulators and carry restored | (1 or 2) |
| K | Store a constant in a block of memory | (3) |
| M | Move a block in memory | (3) |
| N | Search memory for not-equal (with mask) | (3 or 4) |
| O | Output ASCII string on master terminal | (1 or 2) |
| P | Program load from disk | (0 or 1) |
| S | Search memory for constant using a mask | (3 or 4) |
| X | Calculate a checksum for a block of memory | (2) |
| Y | Set up an output delay after each RETURN (for proper scrolling) | (1) |
| : | Examine or deposit into a specified location | (2) |
| CTRL | Control characters are used to access CTU | ** |

*F command is not supported by MARK 9 MANIP.
**Refer to the Cassette Tape Unit Hardware Manual for parameters.

### 3.5.1 ADDRESSING MODES

For many commands, MANIP allows either word or byte addressing, using either real memory addresses or offset (virtual) memory addresses based on an offset that has been previously entered (via F command). MANIP is designed to allow addressing up to 64K words of memory. This is accomplished by having two word-addressing modes (real and virtual), and three byte-addressing modes (one virtual plus two real modes: lower 32K and upper 32K).

These modes are invoked by the optional second parameter "a" shown for commands D, E, J, and O (see Section 3.5.2).

**NOTE**

> The J command does not permit byte addresses.
>
> When no "a" parameter is given, the addressing mode is "word address, including offset, if any." (If there is no "a" parameter, the preceding comma is optional.)

The "a" parameter definitions are as follows:

| a Parameter | Definition |
|---|---|
| omitted | word address, including offset, if any |
| 0 | word address, absolute |
| 1 | byte address, using offset, if any |
| 2 | byte address, lower 64KB |
| 3 | byte address, upper 64KB |

### 3.5.2 COMMAND DESCRIPTIONS

A MANIP command consists of a single letter, which is the command identifier, and parameters that specify addressing modes, memory addresses, and data input. All parameters must be entered in octal form. The letters a, m, n, x, y, and z are used on the following pages to represent octal parameters. Press the RETURN key after entering any command.

Table 3-12 shows MANIP commands and descriptions.

**TABLE 3-12. MANIP COMMAND DESCRIPTIONS**

| Command & Parameters | Definition |
|---|---|
| A | Types out initial value of program counter (PC) saved in first location of MANIP, contents of accumulators A0, A1, A2, A3, and carry flip-flop as they were at the time MANIP was entered. |
| Cx,y | Changes accumulator or carry flip-flop:<br>● If x is 0, 1, 2, or 3, y is stored as saved value for accumulator x (A0, A1, A2, A3, respectively).<br>● If x is 4, saved value of the carry flip-flop is set to 0 if y=0; saved value is set to 1 if y=1.<br>● If x is greater than 4 and an address offset has been established (see F command), x is interpreted as a real address using the offset previously established, and typed out on the master terminal. The y parameter is not used in this case.<br>● Parameter Description<br>x - 1 octal digit 0-7<br>y - 1 word octal |

## TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

| Command & Parameters | Definition |
|---|---|
| Dx,a | Dumps memory in octal, beginning at location x, using addressing mode a. Eight words (or bytes if a byte address mode is used) are typed per line, with the address of the first word (byte) at the beginning of each line.<br>● Parameter Description<br>x - an octal number representing a 16-bit memory address<br>a - 1 digit (0-3 or omitted) representing an addressing mode |
| Ex,a | Enables entry at address x, using addressing mode a. The address (changed to a word address if it was a byte address) is printed, followed by a colon; an octal value may then be entered into the memory location, followed by <RETURN>. The next address (x+1) is then printed and opened for entry. Entry can thus be continued into sequential address locations until ESC is pressed (after <RETURN>) to terminate entry.<br>● If no entry is typed in before <RETURN>, the present contents of the opened location is typed out in octal form, allowing examination of value before entering a new one. If another <RETURN> is entered without an entry, the current value is preserved, and the next address is printed and opened for entry.<br>● If a space character is typed instead of <RETURN>, the contents of the previous address is typed and opened. This feature is convenient for confirming an entry just typed in.<br>● Parameter Description<br>x - octal number representing 16-bit memory address<br>a - 1 digit (0-3 or omitted) representing a memory addressing mode |

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

| Command & Parameters | Definition |
|---|---|
| Fx,y | Establishes an address offset (i.e., a fixed difference between a real memory address on the one hand and an offset address as entered and listed in MANIP on the other). The difference x-y is added to the offset address entered, and subtracted from the memory address before displaying. If y is not entered, it is assumed to be zero. Whenever a nonzero offset is established, an F is typed at the beginning of each line. To revert to real memory addressing, type F0.<br>● Parameter Description<br>x - from 1 to 6 digits (octal) representing a real memory address<br>y - from 1 to 6 digits (octal) representing the offset address that is to be made equivalent to the real address specified in x |
| F | Saves the current offset value and reinstates the previous offset in effect before current one was established. Types offset being reinstated. Makes it convenient to toggle back and forth between two different offsets (or one offset and real memory addressing). |
| Jx,a | Jumps to location x (using addressing mode a) with accumulators and carry stored.<br>● Parameter Description<br>x - an octal number representing 16-bit memory address<br>a - 1 digit (0-3 or omitted) representing a memory addressing mode |
| Kx,y,z | Stores the octal constant z in locations x through y, inclusive.<br>● Parameter Description<br>x - octal number representing 16-bit beginning memory address<br>y - octal number representing 16-bit ending memory address<br>z - octal number representing constant |

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

| Command & Parameters | Definition |
|---|---|
| Mx,y,z | Moves block in memory. Locations x through y are moved to area starting at location z.<br>● Source and destination areas may overlap in either direction without bad effects.<br>● May be used to move MANIP itself as long as destination area does not overlap source area.<br>● Parameter Description<br>x - octal number representing 16-bit beginning memory address<br>y - octal number representing 16-bit ending memory address<br>z - octal number representing 16-bit beginning memory address of new location |
| Nx,y,z,m | Searches for not-equal (see S command). Searches locations x through y for values not equal to constant z. Each word is first ANDed with mask m before comparison with z.<br>● If m is not entered, it is assumed to be 177777.<br>● Use of the mask is best explained by an example: The command Nx,y,0,170000 will search locations x through y for any word whose four MSBs are set to a nonzero value. When such a value is found, its address and contents are displayed in octal form on the CRT.<br>● Parameter Description<br>x - octal number representing 16-bit beginning memory address<br>y - octal number representing 16-bit ending memory address<br>z - octal number representing constant<br>m - octal number representing mask; the default is 177777 |

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

| Command & Parameters | Definition |
|---|---|
| Ox,a | Outputs ASCII. Contents of memory starting at location x (using addressing mode a) are typed out as text. Output is terminated if a zero byte is encountered.<br>● Parameter Description<br>x - octal number representing 16-bit memory address<br>a - 1 digit (0-3 or omitted) representing a memory addressing mode |
| Px | Program load from disk or other DMA devices. Performs standard bootstrap APL function (i.e., gives an NIOS instruction with device code x and then idles at location 377 waiting for the disk to overwrite that location). If x is omitted, reads miniswitches at front edge of CPU board and uses their contents as the device code.<br>**NOTE**<br>If CPU board switches are set to 2xx (octal), MANIP cannot be accessed from memory. Pressing APL causes MANIP to try to boot from device xx.<br>● Parameter Description<br>x - 2-digit octal number (01 through 76) representing the disk device code from which the program is to be loaded |

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

| Command & Parameters | Definition |
|---|---|
| Sx,y,z,m | Searches locations x through y for constant z. Each word is first ANDed with mask m before comparison with z.<br>● If m is omitted, it is assumed to be 177777.<br>● Use of mask is best explained by an example: The command Sx,y,60025,160077 will search locations x through y for any I/O instruction for device 25. When a comparison is found, its address and contents are typed in octal form on CRT.<br>● Parameter Description<br>x - octal number representing 16-bit beginning memory address<br>y - octal number representing 16-bit ending memory address<br>z - 1 to 6 digits representing octal constant<br>m - 1 to 6 digits representing octal mask; the default is 177777 |
| Xx,y | Calculates and types checksum over memory locations x through y. Utilizes a revolving checksum (using a SUBL 0,1 instruction, with A0 = each word from x through y, and A1 = accumulating checksum; initially 0). This ensures that if two words in memory are swapped, the swap will be detected by the checksum. Useful for determining if any word in memory has changed.<br>● Parameter Description<br>x - an octal number representing 16-bit beginning memory address<br>y - an octal number representing a 16-bit ending memory address |

TABLE 3-12. MANIP COMMAND DESCRIPTIONS (Cont)

| Command & Parameters | Definition |
|---|---|
| Yx | Sets up a RETURN delay, required on some CRTs for proper scrolling. After each subsequent carriage return/line feed, MANIP counts up an accumulator from x to 0 before proceeding. For maximum delay set x=0, for no delay set x=177777.<br>● Parameter Description<br>x - 1 to 6 octal digits representing RETURN delay value |
| x:y | Octal value y is stored at location x, and next cell is opened. See E command for more information.<br>● Parameter Description<br>x - octal number representing 16-bit memory address<br>y - 1 to 6 digits representing an octal value |
| CTRL ( ) | Any control character (except <CTRL-W>) is recognized as a cassette tape unit (CTU) command. MANIP will pass the command to the CTU, with the exception of <CTRL-R>, which it uses to read the CPU. Refer to the Cassette Tape Unit Hardware Manual for CTU commands and command functions. |

# Section 4
# INPUT/OUTPUT INTERFACES

## 4.1 INTRODUCTION

This section provides information regarding the basic operating principles and programming methods for the input/output devices that are compatible with the POINT 4 MARK 5/9 computers. Two types of I/O devices are available:

- Those transferring data via I/O programmed instructions only
- Those using the data channel for input/output transfers

The following subsections outline the interrupt handling and priority scheme, conventions for handling the master terminal, programmed transfer handling, and data channel transfer handling. Also provided are I/O bus signal descriptions and I/O transfer timing diagrams.

## 4.2 PROGRAM INTERRUPT AND PRIORITY SCHEME

Many input/output devices require service within a short time after they request it; however, they need service infrequently, relative to the processor speed, and require only a small amount of time for servicing. Failure to service within the specified time (which varies among devices) causes operation of the device to fall below its maximum speed and can result in loss of information.

The use of interrupts in the current program sequence facilitates concurrent operation of the main program and a number of peripheral devices. The program interrupt scheme allows an I/O device to gain control of the processor. When an interrupt occurs, the processor suspends normal program execution and starts a device service routine. When the routine is completed, processing of the interrupted program may resume.

### 4.2.1 INTERRUPT SEQUENCE

When a device needs service, it sets its Interrupt Request flag. The processor begins servicing interrupts if all four of the following conditions exist:

- The processor has just completed an instruction fetch or a data channel transfer
- At least one device has a pending Interrupt Request
- Interrupts are enabled (i.e., ION is set)
- No device is waiting for a data channel transfer

The processor responds to the interrupt request by storing the value of the program counter into memory location 0 and jumping to the instruction addressed by memory location 1. Location 1 must contain the address of the interrupt handling routine. Interrupts are disabled at the start of the interrupt service cycle and must be re-enabled by the software at the end of the interrupt service.

### 4.2.2 DEVICE PRIORITY

The processor features a special interrupt acknowledge instruction that eliminates the need for lengthy device polling. This instruction inputs the device code of the interrupting device into an accumulator register, permitting the interrupt service routine to identify the device requesting service. The computer uses a three-way priority system to determine which device (if any) may interrupt the processor at a given moment. First, the processor contains a programmable Interrupt ON (ION) flag. The processor recognizes interrupt requests only when this flag is set. Second, the processor can selectively disable the interrupt capability of each device with the device Interrupt Mask flags (see Section 5.6.2, MSKO instruction). Third, if two or more devices request interrupts simultaneously, the priority resolution is made by the jumper-saver logic on the POINT 4 MARK 5/9 chassis backplane. A device whose controller board is physically closer to the processor is given priority over a device that is further away. See Appendix E for an example of interrupt programming.

## 4.3 PROGRAMMED TRANSFERS

For programmed input/output, the program directly controls the data transfer between the CPU and the I/O device. As discussed in Section 5.6, each data word is transferred between an accumulator specified in the instruction and an I/O device buffer (A, B, or C) specified in the instruction, as shown in Figure 4-1.

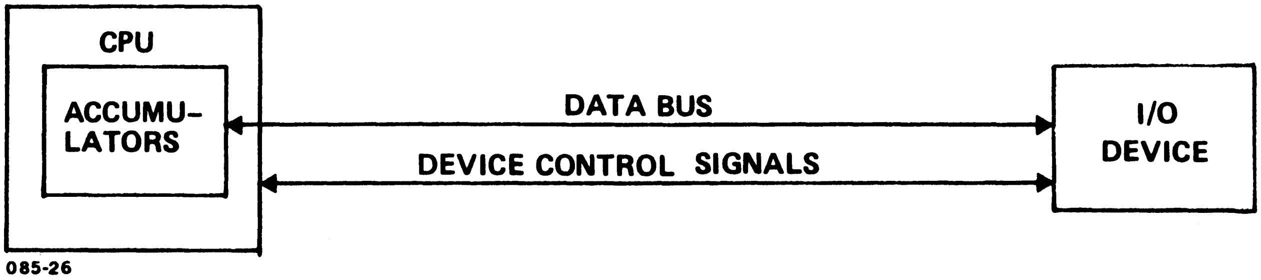


**Figure 4-1. Data Word Transfer Between Specified Accumulator and I/O Device Buffer**

### 4.3.1 MASTER TERMINAL INTERFACE

The standard Teletype or CRT I/O controller has separate interface logic for input and output. Input and output device codes are also separate. The input logic interfaces to the keyboard and, in some Teletypes, a paper tape reader. The output logic interfaces to the printer or video display and, in some Teletypes, the paper tape punch. When the CRT or Teletype is connected to the computer, a character entered on the keyboard for input to the computer must be "echoed" back to the output interface logic on the terminal in order to appear on the screen or paper.

All alphanumeric and control characters are composed of eight bits. The most significant bit (MSB) is an even parity bit. The remaining seven bits represent the ASCII code (see Appendix C) for an alphanumeric or control character.

### 4.3.2 PROGRAMMING CONVENTIONS

Programming conventions for handling CRT and Teletype terminals are described in the following subsections.

#### 4.3.2.1 Instruction Formats

Terminal output and input use separate device codes as specified in bits 10-15 of the instruction. The S or C pulse may be sent to clear Busy and/or Done flags and control the starting of the transfer between the interface and the device. The data transfer output instructions transmit bits 8-15 from the specified accumulator to the output interface register. The input instruction loads the input interface register into bits 8-15 and resets bits 0-7 of the specified accumulator.

#### 4.3.2.2 Terminal Output

Transmission to the terminal takes place when an S pulse is sent, which sets the output Busy flag. When the character has been printed, the interface clears the Busy flag, sets the Done flag, and requests an interrupt, if interrupts are enabled. Terminal output uses the device code 11, the mnemonic TTO, and uses bit 15 to control the Interrupt Mask flag. To transfer a character from an accumulator to the terminal output buffer, use the instruction

```
    DOA ac,TTO
```

where ac may be any of the four accumulators.

If the NIOS instruction is used to send the S pulse, the instructions must appear in the following sequence:

```
    DOA ac,TTO
    NIOS TTO
```

Normally, this operation is done in one instruction:

```
    DOAS ac,TTO
```

#### 4.3.2.3 Terminal Input Via CRT Keyboard

Terminal input uses octal device code 10, the mnemonic TTI, and bit 14 to control the Interrupt Mask flag. When a key is pressed on the keyboard, the character is placed in the input buffer, and Done is set. If interrupts are enabled for that device, an interrupt is requested. The program then reads the character with the instruction

```
    DIA ac,TTI
```

The program then uses an S pulse to clear Done. The S pulse may either be part of the DIA instruction

```
    DIAS ac,TTI
```

or be generated with an NIO instruction

```
    DIA ac,TTI
    NIOS TTI
```

#### 4.3.2.4 Terminal Input Via Paper Tape Reader

When paper tape is used for input, the paper tape reader must be started by the program using the instruction

```
    NIOS TTI
```

This instruction sets Busy and clears Done in the TTY controller. When the character has been read from paper tape into the controller, the device controller clears Busy and sets Done, producing an interrupt if interrupts are enabled. The program then reads the character with the instruction

```
    DIA ac,TTI
```

Sequential characters can be read by using the instruction

```
    DIAS ac,TTI
```

This results in the reading of one character and the restarting of the paper tape reader for reading of the next character.

## 4.4 DATA CHANNEL TRANSFERS

By using the direct memory access (DMA) data channel, mass storage devices, such as tape and disk drives, can transfer blocks of data at high speeds directly into memory without requiring programmed I/O instructions for each word transferred. Data channel device interface logic contains both conventional device registers and flags, and special data channel logic.

The program initiates a data channel transfer by supplying certain parameters to the device registers and starting the device. The device automatically transfers one or more data words to or from memory. When finished with the DMA transfer, the device generates an interrupt if so enabled. At the start of each instruction cycle, the processor checks to see if a device is requesting data channel service. If so, the data channel transfer is performed before going on with the instruction. Several data channel devices can be active at the same time, with devices closest to the processor having channel priority.

The POINT 4 MARK 5 and MARK 9 each have two jumper-selectable data channel speed options:

- Standard data channel (jumper CPU board pin A93 to ground)

  - MARK 5
    Input - 1100 nanoseconds
    Output - 1483 nanoseconds

  - MARK 9
    Input - 1600 nanoseconds
    Output - 2000 nanoseconds

- High-speed data channel (do not jumper CPU board pin A93 to ground; also, ensure that pin A93 is not connected to ground by an etch on the backplane)

  - MARK 5
    Input - 800 nanoseconds
    Output - 933 nanoseconds

  - MARK 9
    Input - 900 nanoseconds
    Output - 1200 nanoseconds

### 4.4.1 SELECTION OF DATA CHANNEL SPEED

CPU logic tests input to pin A93 (top slot of POINT 4 chassis) to determine which speed has been enabled (see Section 4.5.2 for backplane pin assignments). If pin A93 is jumpered to ground, the data channel will operate at standard data channel speeds. If pin A93 is left open, the data channel will operate at high-speed data channel speeds.

### 4.4.2 CRITERION FOR DATA CHANNEL SPEED SELECTION

The high-speed data channel on the POINT 4 MARK 5/9 does not have stringent requirements for controller timing. The controller is given about 200 nanoseconds from the start of DCHA to put its address on the I/O bus. The POINT 4 MARK 5/9 also allows 200 nanoseconds from the start of DCHI before it requires the input data on the I/O bus. (See Section 4.6.3 on data channel transfer approximate timing.)

The result of this relatively long access time to the I/O bus is that many DMA controllers that cannot operate on a high-speed data channel with other computers can operate on the POINT 4 MARK 5/9 high-speed data channel. Therefore, the high-speed data channel should be used unless the system includes a controller with specifications that indicate otherwise. If operation is inconsistent at high speeds, the user can switch to the standard data channel by jumpering pin A93 of slot occupied by POINT 4 MARK 5 or 9 CPU to ground. If operation is then consistent at the standard data channel speed, this speed should be maintained. Section 2.6.3 contains information on data channel speed selection for the MARK 5/9 universal CPU.

### 4.4.3 DATA CHANNEL ACCESS PRIORITIES

The amount of time a device must wait for data channel access depends on when its request is made within an instruction and how many devices of higher priority are also requesting access. Once the processor reaches a point at which it can pause to handle transfers, a given device must wait until all devices closer to the processor on the bus have been serviced. Under normal conditions, a device can preempt all processor time if it requests access at the maximum rate. An exception is made if power-fail has been sensed, in which case the data channel is allowed only every other cycle. The alternate cycle is used for power-fail interrupt processing. At less than the maximum rate, the closest device never waits longer than the time required for the processor to finish the instruction being performed when the request is synchronized. However, indirect addressing can extend this beyond the normal instruction execution time.

# 4.5 INPUT/OUTPUT BUS INTERFACE SIGNALS

Input/output bus signals connect the processor logic to peripheral device logic. The logic for programmed I/O transfers and data channel transfers forms the interface between the processor main data bus and the peripheral device controller logic. Logic to implement both I/O transfer and I/O skip instructions is present in all device controllers. Data channel transfer logic is present only in those controllers that control devices using the data channel. Device-end control logic for these functions may vary widely, depending on the requirements of the particular device. This subsection describes the POINT 4 MARK 5/9 I/O bus and control signals.

## 4.5.1 INPUT/OUTPUT INTERFACE SIGNALS

Signals on the input/output bus can be grouped into the signal classifications shown in Table 4-1.

Figure 4-2 is a diagram of I/O signals across the I/O bus. Table 4-2 lists these signals by classification group and signal name, indicating direction and describing each signal function.

## TABLE 4-1. INPUT/OUTPUT BUS SIGNAL CLASSIFICATIONS

| Signal Classification | Number of Lines | Definition |
|---|---|---|
| Bidirectional data bus | 16 | Used for transfer of all data and address words between the CPU and a peripheral device, for both programmed I/O and data channel transfers. |
| Device code | 6 | Code generated by CPU to specify the address of a peripheral device or a controller used for an input/output instruction. |
| Programmed data transfer signals | 6 | These signals, generated by the CPU in response to input/output instructions for data transfers, are used to control data transfers for programmed input/output devices. |
| Device control signals | 4 | These signals are generated by the CPU in response to input/output instructions, and are used to initialize and control I/O devices. The signals affect only the device whose device code is in the instruction, except in the case of the IORST instruction which resets all devices. |
| Skip testing flags | 2 | Flags supplied to the CPU when skip-testing is required. |
| Interrupt control signals | 5 | Signals used to initialize and control the interrupt sequence. |
| Data Channel control signals | 6 | Signals used to control data channel transfers between CPU memory and a peripheral device. |

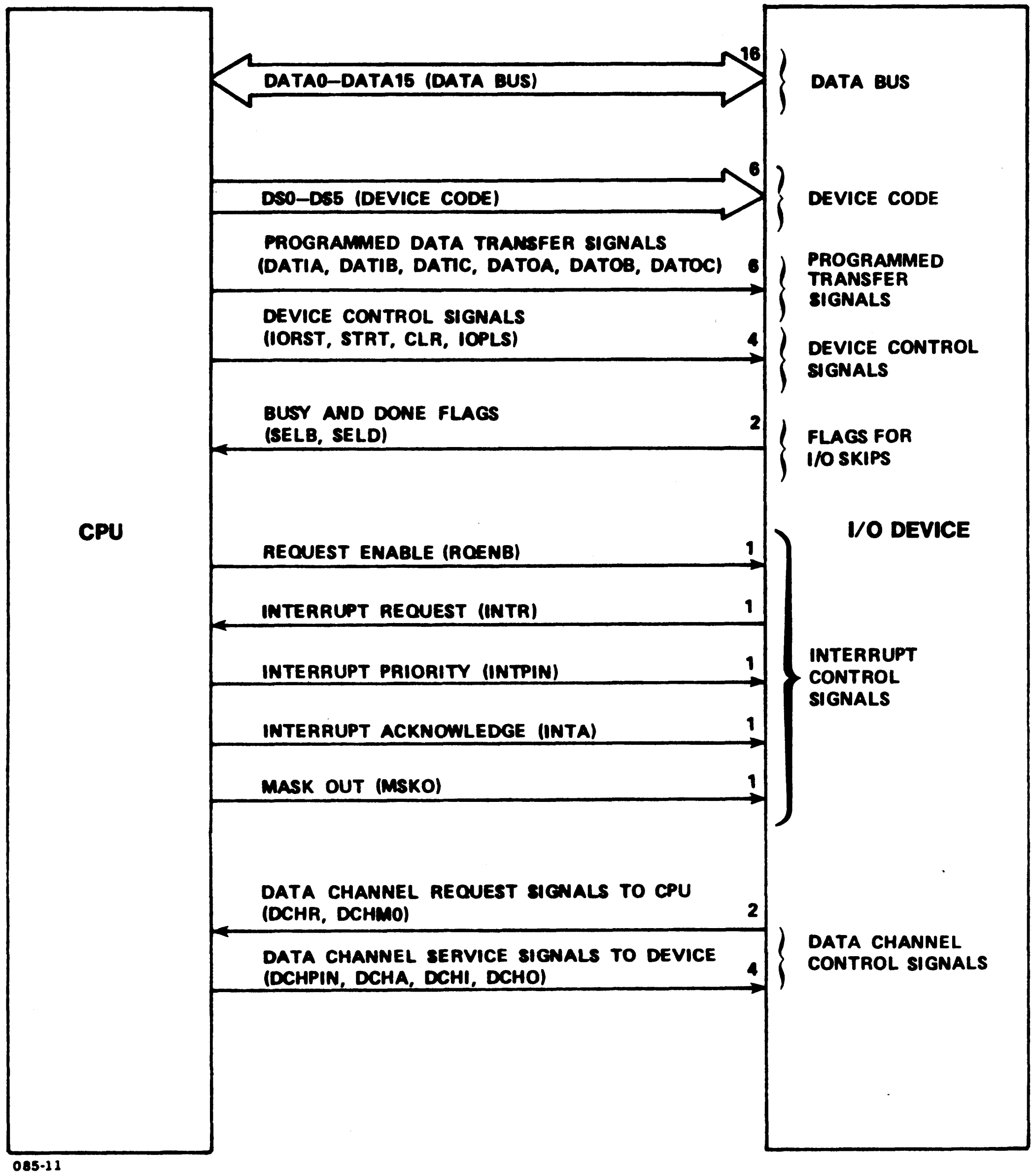

CPU
I/O DEVICE
DATA0—DATA15 (DATA BUS)
16
DATA BUS
DS0—DS5 (DEVICE CODE)
6
DEVICE CODE
PROGRAMMED DATA TRANSFER SIGNALS
(DATIA, DATIB, DATIC, DATOA, DATOB, DATOC)
6
PROGRAMMED TRANSFER SIGNALS
DEVICE CONTROL SIGNALS
(IORST, STRT, CLR, IOPLS)
4
DEVICE CONTROL SIGNALS
BUSY AND DONE FLAGS
(SELB, SELD)
2
FLAGS FOR I/O SKIPS
REQUEST ENABLE (RQENB)
1
INTERRUPT REQUEST (INTR)
1
INTERRUPT PRIORITY (INTPIN)
1
INTERRUPT ACKNOWLEDGE (INTA)
1
MASK OUT (MSKO)
1
INTERRUPT CONTROL SIGNALS
DATA CHANNEL REQUEST SIGNALS TO CPU
(DCHR, DCHM0)
2
DATA CHANNEL SERVICE SIGNALS TO DEVICE
(DCHPIN, DCHA, DCHI, DCHO)
4
DATA CHANNEL CONTROL SIGNALS
085-11

Figure 4-2. Input/Output Signal Diagram

## TABLE 4-2. INPUT/OUTPUT SIGNALS

| Signal Group | Signal Name* | Direction | Description |
|---|---|---|---|
| Data bus | DATA0- to DATA15- | Bidirectional | All data and addresses are supplied to and from the device via these lines. DATA0- is the MSB. |
| Device code | DS0- to DS5- | From CPU | The CPU places the device code (bits 10-15 of the instruction word) on these lines during the execution of an input/output instruction. DS0- is the MSB. |
| Programmed data transfer signals | DATIA+ | From CPU | Data in A - generated by a DIA instruction. Causes buffer A of the device whose device code is on the lines to be placed on the data bus for entry into the accumulator specified by the instruction. |
| | DATOA+ | From CPU | Data out A - generated by a DOA instruction. Causes the accumulator specified in the DOA instruction to be placed on the data bus for entry into the A buffer of the device whose device code is on the lines. |
| | DATIB+ | From CPU | Data in B - generated by a DIB instruction. Functions like data in A, but uses buffer B. |
| | DATOB+ | From CPU | Data out B - generated by a DOB instruction. Functions like data out A, but uses buffer B. |
| | DATIC+ | From CPU | Data in C - generated by a DIC instruction. Functions like data in A, but uses buffer C. |
| | DATOC+ | From CPU | Data out C - generated by a DOC instruction. Functions like data out A, but uses buffer C. |

*Signal names ending with + are active high; those ending with - are active low.

**TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)**

| Signal Group | Signal Name* | Direction | Description |
|---|---|---|---|
| Device control signals | IORST+ | From CPU | Input/output reset - generated when APL is pressed on the Mini-Panel, when RESET is pressed on the Operator Control Unit, when an IORST instruction is being executed, and during power turn-on. |
| | STRT+ | From CPU | Start - generated when the CTRL field of an input/output transfer instruction contains code 01. It usually clears the Done flag and Interrupt request, and sets the Busy flag in the device whose device code is on the lines. |
| | CLR+ | From CPU | Clear - generated when the CTRL field of an input/output transfer instruction contains code 10. It usually clears the Busy and Done flags and the Interrupt request in the device whose device code is on the lines. |
| | IOPLS+ | From CPU | I/O pulse - generated when the CTRL field of an input/output transfer instruction contains code 11. The effect, if any, depends on the device. |
| Skip testing flags | SELB- | From device | Selected device busy - supplied to CPU by the device whose device code is on the lines when the Busy flag is set. Indicates that the device is busy. |
| | SELD- | From device | Selected device done - supplied to the CPU by the device whose device code is on the lines when the Done flag is set. Indicates that the device is done. |

**TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)**

| Signal Group | Signal Name* | Direction | Description |
|---|---|---|---|
| Interrupt control signal | RQENB- | From CPU | Request enable - generated during each memory read/write cycle to synchronize INTR- and DCHR-. In any device, changes in INTR- or DCHR- may only occur following the leading edge (high-to-low transition) of RQENB-. |
| | INTR- | From device | Interrupt request - this signal goes low (following the leading edge of RQENB-) if the device wants to request an interrupt. |
| | INTPIN- | From jumper saver | Interrupt priority input - on POINT 4 chassis, this signal is produced by jumper-saver logic on the backplane for highest priority device requesting an interrupt. |
| | | Jumpered to ground or from higher priority device | On non-POINT 4 chassis, this signal is produced by a jumper to ground on lowest (highest in priority) I/O board. If the highest priority I/O board in the non-POINT 4 chassis has not requested an interrupt, this signal is output as INTPOUT- to the device next in priority. |
| | INTA+ | From CPU | Interrupt acknowledge - generated by an INTA instruction. Causes the device whose INTPIN- line is low to place its device code in bits 10-15 of the data bus for entry into accumulator specified in the instruction. |
| | MSKO- | From CPU | Mask out - generated by a MSKO instruction. Commands all I/O devices to set their interrupt disable flags according to the state of the associated mask bit in the word on the data bus. |

TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)

| Signal Group | Signal Name* | Direction | Description |
|---|---|---|---|
| Data channel control signals | DCHR- | From device | Data channel request - this signal goes low (following the leading edge of RQENB-) if the device wants to request a data channel transfer. |
| | DCHMO- | From device | Data channel mode - generated by a device connected to the data channel while DCHA- is low. Indicates the type of data channel cycle being requested as follows:<br>DCHMO    Type of Cycle<br>0 (high)  Data out (from CPU)<br>1 (low)  Data in (to CPU) |
| | DCHPIN- | From jumper saver | Data channel priority input - on POINT 4 chassis, this signal is produced by jumper-saver logic on the backplane for highest priority device requesting a data channel transfer. |
| | | Jumpered to ground or from higher priority device | On non-POINT 4 chassis, this signal is produced by a jumper to ground on lowest (highest in priority) I/O board. If the highest priority I/O board in the non-POINT 4 chassis has not requested a data channel transfer, this signal is output as DCHPOUT- to the device next in priority. |
| | DCHA- | From CPU | Data channel acknowledge - generated by CPU in response to a data channel request. Initiates a data channel cycle in the device whose DCHPIN- is low. The device places the memory address for data channel access on the data bus. |

TABLE 4-2. INPUT/OUTPUT SIGNALS (Cont)

| Signal Group | Signal Name* | Direction | Description |
|---|---|---|---|
| | DCHI+ | From CPU | Data channel in - when the mode is data in, DCHI+ is generated during the time the device is placing a data word on the data bus. |
| | DCHO+ | From CPU | Data channel out - when the mode is data out, DCHO+ is generated during the time that the word accessed from memory is on the data bus. |

### 4.5.2 BACKPLANE PIN SIGNAL CONNECTORS

All signal connections between the processor and each controller take place via two 100-pin backplane connectors. Figure 4-3 shows the connector pin layout for all I/O signals. The labeled pins refer to the I/O control signals, data transfer signals, and the power lines used by peripheral controllers.

The POINT 4 MARK 5 and MARK 9 have one special signal, STSEL-, which goes through backplane connector A pin 91. It is a remote Self-Test select signal that allows the CPU Self-Test program to be bootstrapped to memory when the signal is at an active low during the APL sequence.

For a MARK 5 or MARK 9 universal CPU, see Section 2.6.2 for information about backplane connection requirements.

| BOTTOM | | A | | TOP | |
|---|---|---|---|---|---|
| GND | | 2 | 1 | GND | |
| +5V | | 4 | 3 | +5V | |
| –5V | | 6 | 5 | +5BU | • |
| | | 8 | 7 | PWRGON | • |
| +15V | | 10 | 9 | –5BU | • |
| | | 12 | 11 | PWRF | • |
| | | 14 | 13 | | |
| | | 16 | 15 | | |
| | | 18 | 17 | | |
| | | 20 | 19 | | |
| | | 22 | 21 | | |
| | | 24 | 23 | | |
| | | 26 | 25 | | |
| | | 28 | 27 | | |
| | | 30 | 29 | | |
| | | 32 | 31 | | |
| GND | | 34 | 33 | GND | |
| | | 36 | 35 | | |
| MSKO– | | 38 | 37 | | |
| INTA+ | | 40 | 39 | | |
| DATIB+ | | 42 | 41 | | |
| DATIA+ | | 44 | 43 | | |
| DS3– | | 46 | 45 | | |
| DATOC+ | | 48 | 47 | | |
| CLR+ | | 50 | 49 | | |
| STRT+ | | 52 | 51 | | |
| DATIC+ | | 54 | 53 | | |
| DATOB+ | | 56 | 55 | | |
| DATOA+ | | 58 | 57 | | |
| DCHA– | | 60 | 59 | | |
| DS4– | | 62 | 61 | | |
| DS5– | | 64 | 63 | | |
| DS2– | | 66 | 65 | | |
| DS1– | | 68 | 67 | | |
| IORST+ | | 70 | 69 | | |
| DS0– | | 72 | 71 | | |
| IOPLS+ | | 74 | 73 | | |
| | | 76 | 75 | | |
| | | 78 | 77 | | |
| SELD– | | 80 | 79 | | |
| SELB– | | 82 | 81 | | |
| PEL– | | 84 | 83 | | |
| RUNL– | | 86 | 85 | | |
| CL– | | 88 | 87 | | |
| CONT– | | 90 | 89 | | |
| STOP– | | 92 | 91 | STSEL – | • |
| | | 94 | 93 | HISPDC + | • |
| | | 96 | 95 | INTPOUT– | |
| +5V | | 98 | 97 | +5V | |
| GND | | 100 | 99 | GND | |

CPU MINI-PANEL •

| BOTTOM | | B | | TOP | |
|---|---|---|---|---|---|
| GND | | 2 | 1 | GND | |
| +5 | | 4 | 3 | +5V | |
| 60Hz | •• | 6 | 5 | | |
| | | 8 | 7 | | |
| | | 10 | 9 | | |
| | | 12 | 11 | | |
| | | 14 | 13 | | |
| | | 16 | 15 | | |
| | | 18 | 17 | DCHM0– | |
| | | 20 | 19 | | |
| | | 22 | 21 | | |
| | | 24 | 23 | | |
| | | 26 | 25 | | |
| | | 28 | 27 | | |
| | | 30 | 29 | INTR– | |
| | | 32 | 31 | OPNCBL.T– | • |
| | | 34 | 33 | DCHO+ | |
| | | 36 | 35 | DCHR– | |
| | | 38 | 37 | DCHI+ | |
| | | 40 | 39 | | |
| | | 42 | 41 | RQENB– | |
| | | 44 | 43 | | |
| +15V | | 46 | 45 | | |
| | | 48 | 47 | | |
| GND | | 50 | 49 | | |
| | | 52 | 51 | | |
| | | 54 | 53 | | |
| DATA14+ | | 56 | 55 | DATA7– | |
| DATA11+ | | 58 | 57 | DATA5– | |
| DATA8– | | 60 | 59 | DATA12– | |
| DATA0– | | 62 | 61 | DATA4– | |
| DATA13– | | 64 | 63 | DATA9– | |
| DATA15– | | 66 | 65 | DATA1– | |
| | | 68 | 67 | | |
| | | 70 | 69 | | |
| | | 72 | 71 | | |
| | | 74 | 73 | DATA3– | |
| | | 76 | 75 | DATA10– | |
| | | 78 | 77 | | |
| | | 80 | 79 | | |
| DATA2– | | 82 | 81 | –5V | |
| +15V | | 84 | 83 | | |
| | | 86 | 85 | | |
| | | 88 | 87 | | |
| | | 90 | 89 | | |
| GND | | 92 | 91 | | |
| +12BU | • | 94 | 93 | | |
| | | 96 | 95 | DATA6– | |
| +5V | | 98 | 97 | +5V | |
| GND | | 100 | 99 | GND | |

•CPU ONLY

BACKPLANE SLOT 1

••CPU SLOT ONLY,
NOT USED BY POINT 4 CPU

085-27A

Figure 4-3. Backplane I/O Signals (1 of 2)

| BOTTOM | A | | TOP |
|---|---|---|---|
| GND | 2 | 1 | GND |
| +5V | 4 | 3 | +5V |
| –5V • | 6 | 5 | |
| | 8 | 7 | |
| +15V • | 10 | 9 | |
| | 12 | 11 | |
| | 14 | 13 | |
| | 16 | 15 | |
| | 18 | 17 | |
| | 20 | 19 | |
| | 22 | 21 | |
| | 24 | 23 | |
| | 26 | 25 | |
| | 28 | 27 | |
| | 30 | 29 | |
| | 32 | 31 | |
| GND | 34 | 33 | GND |
| | 36 | 35 | |
| MSKO– | 38 | 37 | |
| INTA+ | 40 | 39 | |
| DATIB+ | 42 | 41 | |
| DATIA+ | 44 | 43 | |
| DS3 | 46 | 45 | |
| DATOC+ | 48 | 47 | |
| CLR+ | 50 | 49 | |
| STRT+ | 52 | 51 | |
| DATIC+ | 54 | 53 | |
| DATOB+ | 56 | 55 | |
| DATOA+ | 58 | 57 | |
| DCHA– | 60 | 59 | |
| DS4– | 62 | 61 | |
| DS5– | 64 | 63 | |
| DS2– | 66 | 65 | |
| DS1– | 68 | 67 | |
| IORST+ | 70 | 69 | |
| DS0– | 72 | 71 | |
| IOPLS+ | 74 | 73 | |
| | 76 | 75 | |
| | 78 | 77 | |
| SELD– | 80 | 79 | |
| SELB– | 82 | 81 | |
| | 84 | 83 | |
| | 86 | 85 | |
| | 88 | 87 | |
| | 90 | 89 | |
| | 92 | 91 | |
| DCHPIN– • | 94 | 93 | DCHPOUT–• |
| INTPIN– • | 96 | 95 | INTPOUT– |
| +5V | 98 | 97 | +5V |
| GND | 100 | 99 | GND |

| BOTTOM | B | | TOP |
|---|---|---|---|
| GND | 2 | 1 | GND |
| +5 | 4 | 3 | +5V |
| | 6 | 5 | |
| | 8 | 7 | |
| | 10 | 9 | |
| | 12 | 11 | |
| | 14 | 13 | |
| | 16 | 15 | |
| | 18 | 17 | DCHM0– |
| | 20 | 19 | |
| | 22 | 21 | |
| | 24 | 23 | |
| | 26 | 25 | |
| | 28 | 27 | |
| | 30 | 29 | INTR– |
| | 32 | 31 | |
| | 34 | 33 | DCHO+ |
| | 36 | 35 | DCHR– |
| | 38 | 37 | DCHI+ |
| | 40 | 39 | |
| | 42 | 41 | RQENB– |
| | 44 | 43 | |
| +15V • | 46 | 45 | |
| | 48 | 47 | |
| GND | 50 | 49 | |
| | 52 | 51 | |
| | 54 | 53 | |
| DATA14– | 56 | 55 | DATA7– |
| DATA11– | 58 | 57 | DATA5– |
| DATA8– | 60 | 59 | DATA12– |
| DATA0– | 62 | 61 | DATA4– |
| DATA13– | 64 | 63 | DATA9– |
| DATA15– | 66 | 65 | DATA1– |
| | 68 | 67 | |
| | 70 | 69 | |
| | 72 | 71 | |
| | 74 | 73 | DATA3– |
| | 76 | 75 | DATA10– |
| | 78 | 77 | |
| | 80 | 79 | |
| DATA2– | 82 | 81 | –5V • |
| +15V • | 84 | 83 | |
| | 86 | 85 | |
| | 88 | 87 | |
| | 90 | 89 | |
| GND | 92 | 91 | –15V • |
| | 94 | 93 | –15V • |
| | 96 | 95 | DATA6– |
| +5V | 98 | 97 | +5V |
| GND | 100 | 99 | GND |

*NOT ON CPU SLOT

BACKPLANE SLOTS 2–7

085-27B

**Figure 4-3. Backplane I/O Signals (2 of 2)**

## 4.6 INPUT/OUTPUT TIMING

Three classes of operations take place on the I/O bus: those associated with programmed I/O instructions, those associated with interrupt handling, and those associated with data channel transfers. Timing diagrams in this section represent each signal or group of signals by a horizontal line with a raised section representing the active state. Control signals generated at a specific time to control a particular function show the raised line for the time that the signal is active. For signals carrying binary information, the raised line indicates the amount of time that information remains on the bus. Raised lines may represent either high or low voltage levels, depending on whether the signal is active when low or high. (See Table 4-1 for active voltages on all signals.) Times on timing diagrams are given in nanoseconds.

### 4.6.1 PROGRAMMED I/O INSTRUCTION TIMING

Figures 4-4 through 4-6 are programmed I/O instruction timing diagrams. These figures show the timing for data in (DIx) and data out (DOx) instructions both with and without a control pulse (S, C or P) specified. In all cases, the processor first places the appropriate device code on the device select lines DS0-DS5. The selected device then responds to the signals that follow.

#### 4.6.1.1 Data In

During data-in transfers, the processor generates a DATIA+, DATIB+, or DATIC+ signal. The device selected then places the contents of its appropriate buffer onto the data transfer lines. At the end of the DATIx active signal, the processor strobes the value on the data lines into the appropriate processor accumulator. Following the transfer, the processor generates the pulse for a START (S), CLEAR (C), or PULSE (P), if called for by the instruction.

#### 4.6.1.2 Data Out

During data-out transfers, the processor places the contents of the accumulator selected by the instruction onto the data transfer lines. The processor then generates a DATOA+, DATOB+, or DATOC+ signal, which causes the device to strobe in the data to the buffer specified. When the data has been loaded into the buffer, the processor generates the pulse for START (S), CLEAR (C), or PULSE (P), if called for by the instruction.

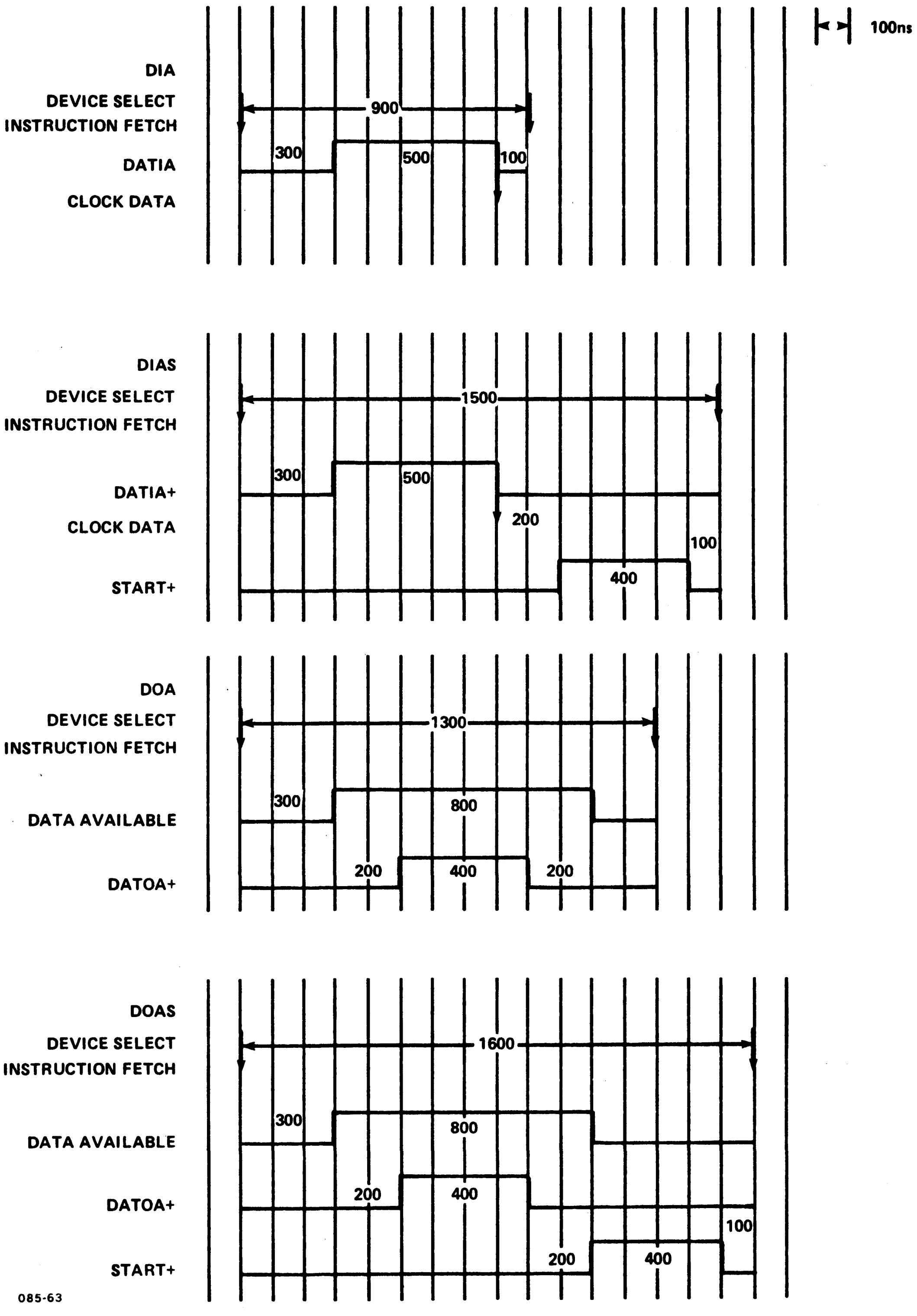


**Figure 4-4. Programmed I/O Instruction Timing for MARK 5**

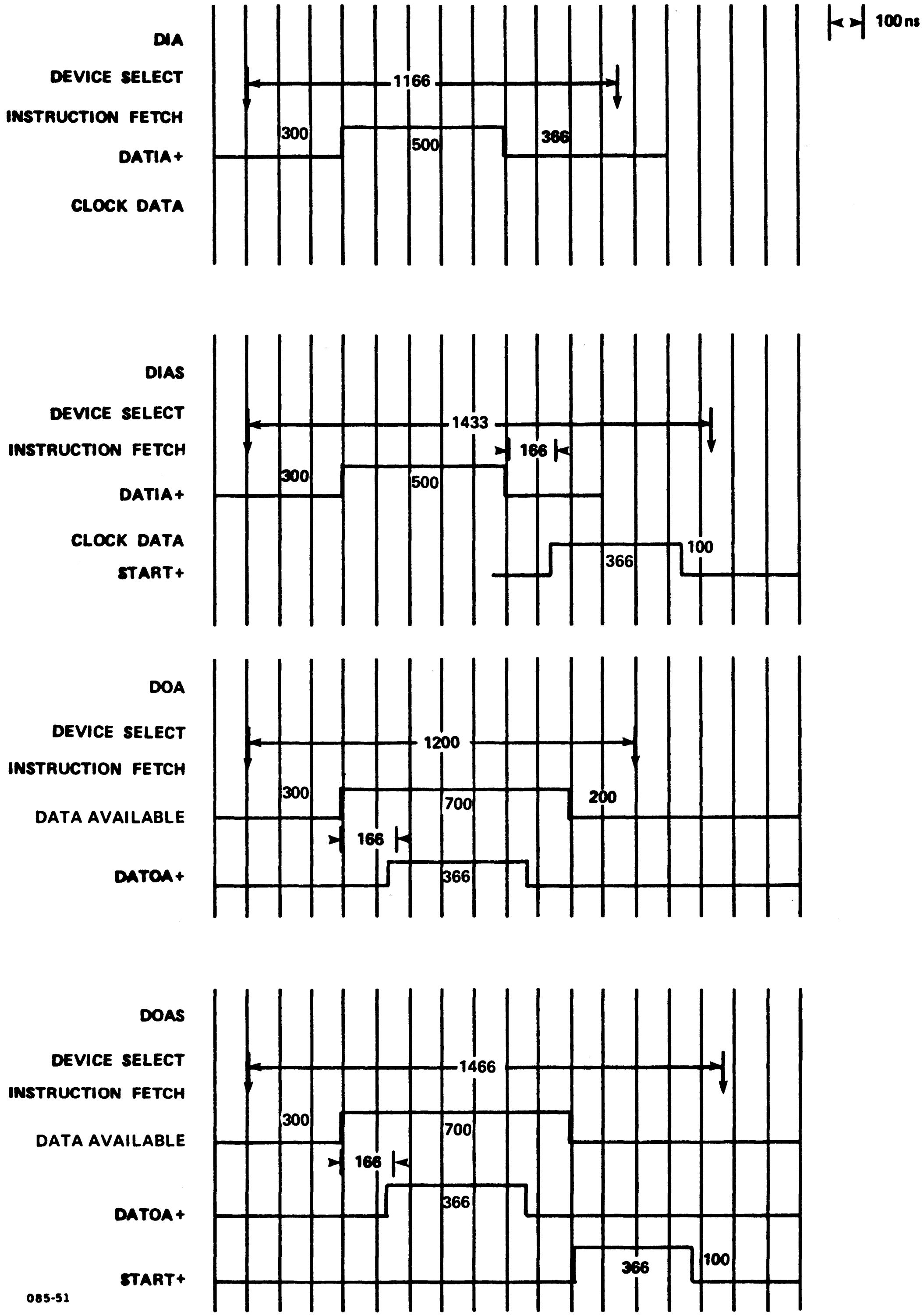


**Figure 4-5. Programmed I/O Instruction Timing for MARK 5 with EIS**

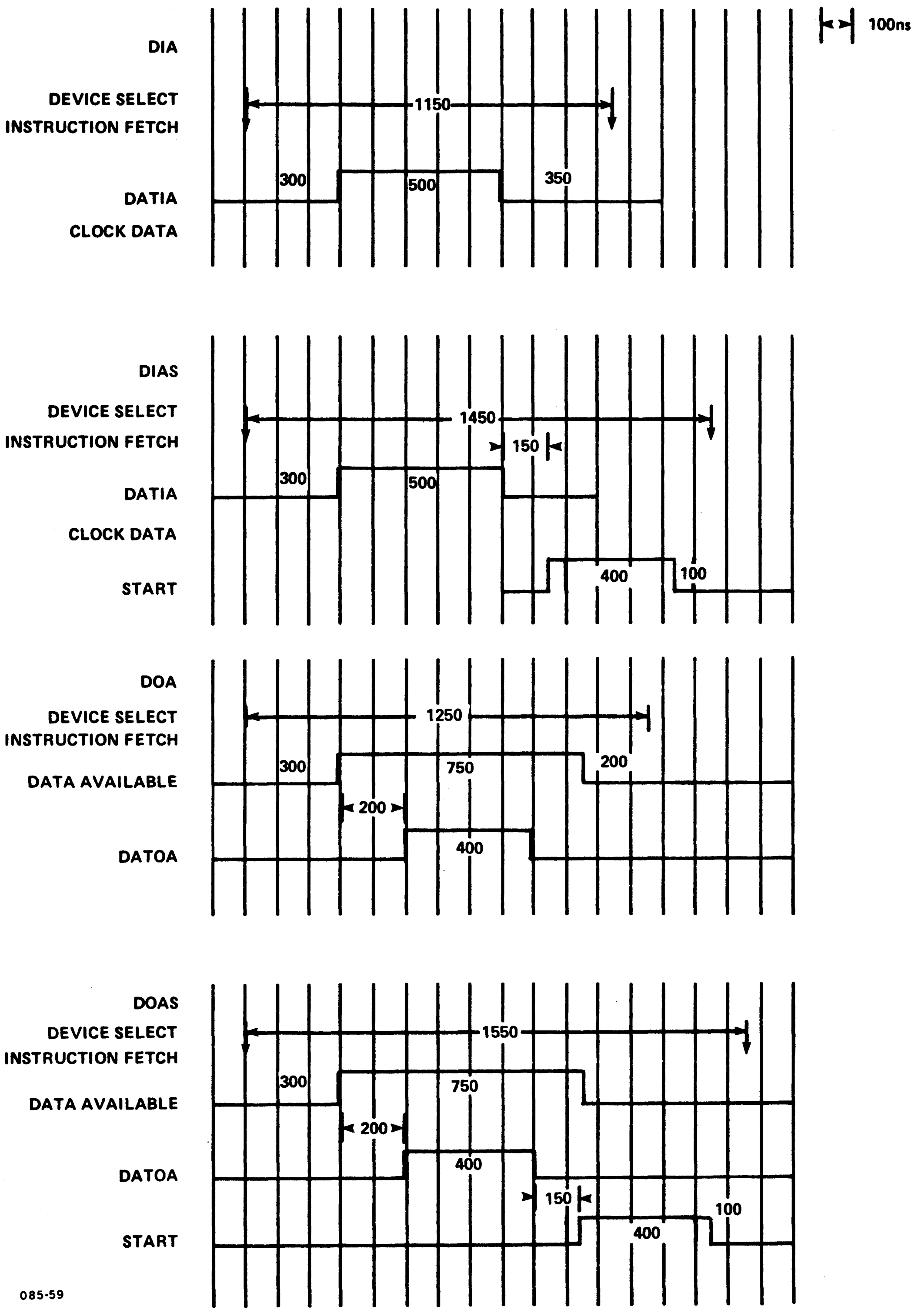
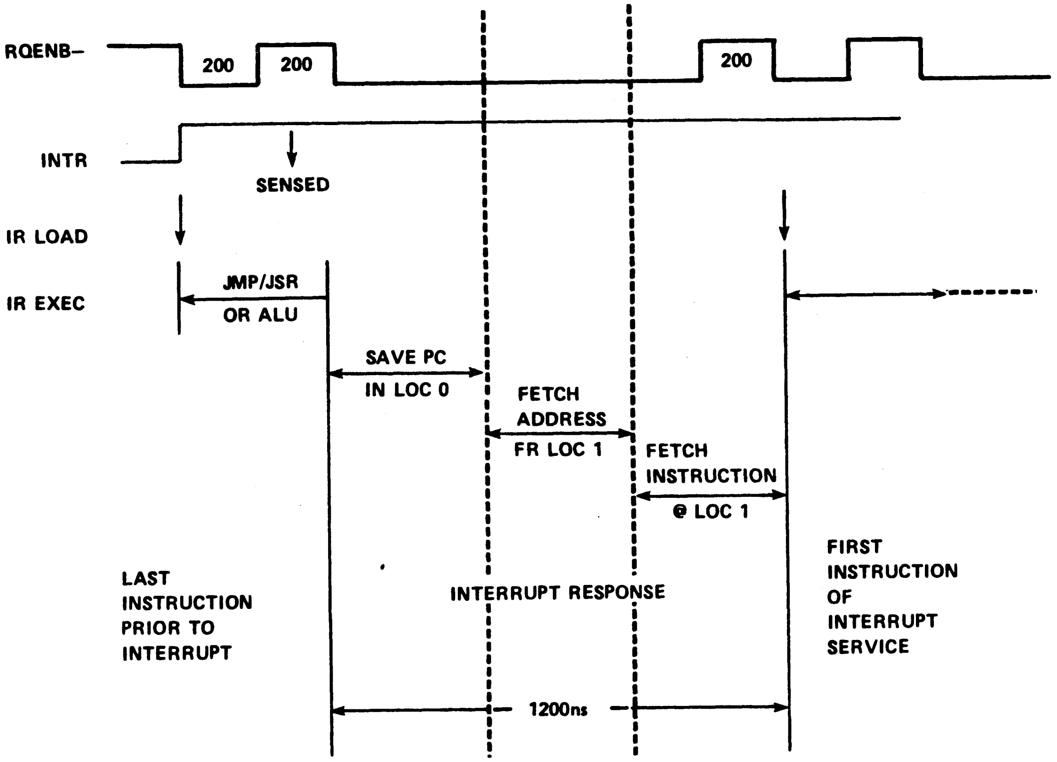


**Figure 4-6. Programmed I/O Instruction Timing for MARK 9**

### 4.6.2 PROGRAM INTERRUPT TIMING

At the end of every memory cycle, the processor generates the signal RQENB- and places it on the I/O bus. All devices receive the RQENB- signal and each responds according to its need for service. Any device requiring interrupt servicing pulls the signal INTR- low.

Figures 4-7 through 4-9 are timing diagrams of interrupt handling.

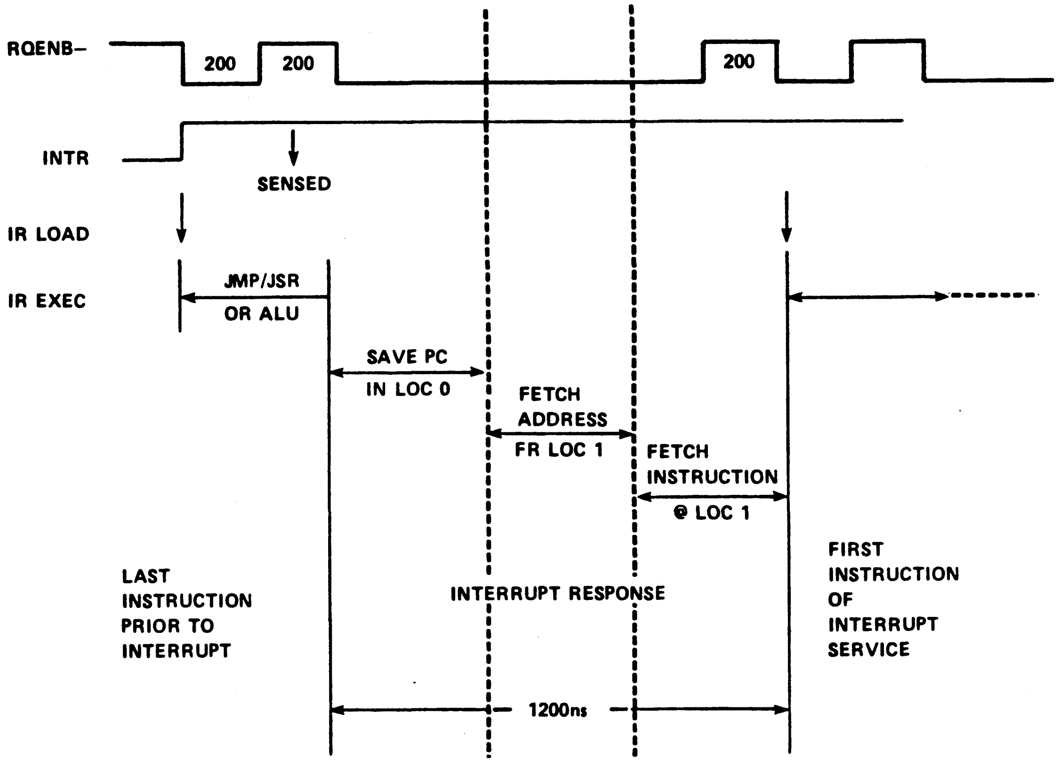


RQENB– = REQUEST ENABLE: PROVIDES A CLOCK SIGNAL TO SYNCHRONIZE INTERRUPT REQUESTS (AS WELL AS DATA CHANNEL REQUESTS) FROM PERIPHERAL CONTROLLERS. ITS NOMINAL PERIOD IS 400ns (200ns OFF, 200ns ON). CONTROLLERS MAY CHANGE INTR (INTERRUPT REQUEST) OR DCHR– (DATA CHANNEL REQUEST) ONLY AT THE LEADING EDGE OF RQENB–.

INTR = INTERRUPT REQUEST: SENSED BY CPU ABOUT 100ns BEFORE THE END OF EACH INSTRUCTION. IF INTR IS PRESENT, THE NEXT THREE MEMORY CYCLES (1200ns) ARE TAKEN TO SAVE PC IN LOCATION 0 AND TO JUMP TO THE INTERRUPT SERVICE ROUTINE WHOSE ADDRESS IS IN LOCATION 1.

085-64

Figure 4-7. Interrupt Timing for MARK 5

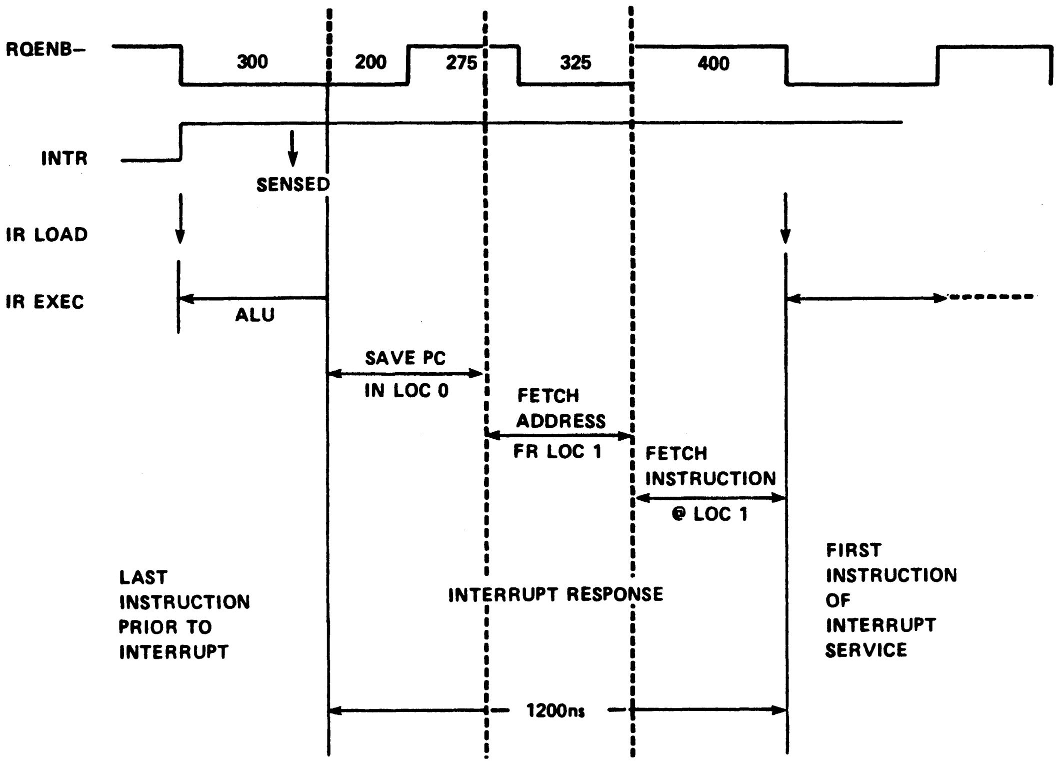


RQENB– = REQUEST ENABLE: PROVIDES A CLOCK SIGNAL TO SYNCHRONIZE INTERRUPT REQUESTS (AS WELL AS DATA CHANNEL REQUESTS) FROM PERIPHERAL CONTROLLERS. ITS NOMINAL PERIOD IS 400ns (200ns OFF, 200ns ON). CONTROLLERS MAY CHANGE INTR (INTERRUPT REQUEST) OR DCHR– (DATA CHANNEL REQUEST) ONLY AT THE LEADING EDGE OF RQENB–.

INTR = INTERRUPT REQUEST: SENSED BY CPU APPROXIMATELY 100ns BEFORE THE END OF EACH INSTRUCTION. IF INTR IS PRESENT, THE NEXT THREE MEMORY CYCLES (1200ns) ARE TAKEN TO SAVE PC IN LOCATION 0 AND TO JUMP TO THE INTERRUPT SERVICE ROUTINE WHOSE ADDRESS IS IN LOCATION 1.

085-54

Figure 4-8. Interrupt Timing for MARK 5 with EIS



RQENB– = REQUEST ENABLE: PROVIDES A CLOCK SIGNAL TO SYNCHRONIZE INTERRUPT REQUESTS (AS WELL AS DATA CHANNEL REQUESTS) FROM PERIPHERAL CONTROLLERS. ITS NOMINAL PERIOD IS 600ns (300ns OFF, 300ns ON). CONTROLLERS MAY CHANGE INTR (INTERRUPT REQUEST) OR DCHR– (DATA CHANNEL REQUEST) ONLY AT THE LEADING EDGE OF RQENB–.

INTR = INTERRUPT REQUEST: SENSED BY CPU ABOUT 100ns BEFORE THE END OF EACH INSTRUCTION. IF INTR IS PRESENT, THE NEXT THREE MEMORY CYCLES (1200ns) ARE TAKEN TO SAVE PC IN LOCATION 0 AND TO JUMP TO THE INTERRUPT SERVICE ROUTINE WHOSE ADDRESS IS IN LOCATION 1.

085-55

**Figure 4-9. Interrupt Timing for MARK 9**

### 4.6.3 DATA CHANNEL TRANSFER TIMING

Data channel transfers are in either the input or output direction (data channel input is a write into memory and data channel output is a read from memory). In either case, the device first requests use of the I/O bus. When the processor acknowledges the request, it stops program execution long enough to conduct the transfer between the device and memory.

Operations in data channel requests are similar to those of an Interrupt request. At the end of every memory cycle, the processor generates the signal RQENB- and places it on the I/O bus. All devices receive the RQENB- signal and each responds according to its need for service. Any device requiring data channel service pulls the DCHR- line low. The jumper saver logic on the POINT 4 chassis backplane then determines which is the highest priority device requesting data channel service, and sends DCHP- (data channel priority) to that device. Devices whose DCHP- lines are inactive (high) ignore subsequent data channel control signals.

When the processor is ready to process the data channel request, it activates the signal DCHA- (data channel address). The device whose DCHP- line is active (low) places the address for the DMA transfer on the I/O bus during DCHA-. At the same time, the device also activates or negates DCHM0- to specify whether an input or output transfer is to take place.

When DCHA- terminates, the processor strobes the address into its memory address register. From this point on, the operation depends on the direction of the data channel transfer.

- Data channel input - when a data input transfer is required, the processor transmits DCHI+ immediately following the trailing edge of DCHA-. The device then places the data word onto lines DATA0 through DATA15. Near the trailing edge of DCHI+, the processor stores the data word into memory, and the device removes the data word from lines DATA0- through DATA15-.

- Data channel output - in an output transfer, the processor starts a read memory cycle at the trailing edge of DCHA-. When the data has been fetched from memory, the processor places the word on lines DATA0- through DATA15- and activates a DCHO+ signal. The device then fetches the data from the data lines.

When the transfer required is a single-word transfer, the device clears DCHR- the next time it receives RQENB-. If the transfer required is several words in consecutive data channel cycles, the DCHR- flag should remain active until the leading edge of RQENB- following the DCHA- of the last transfer desired.

Figures 4-10 through 4-12 are timing diagrams of standard data channel operations and Figures 4-13 through 4-15 are timing diagrams of high-speed data channel operations.

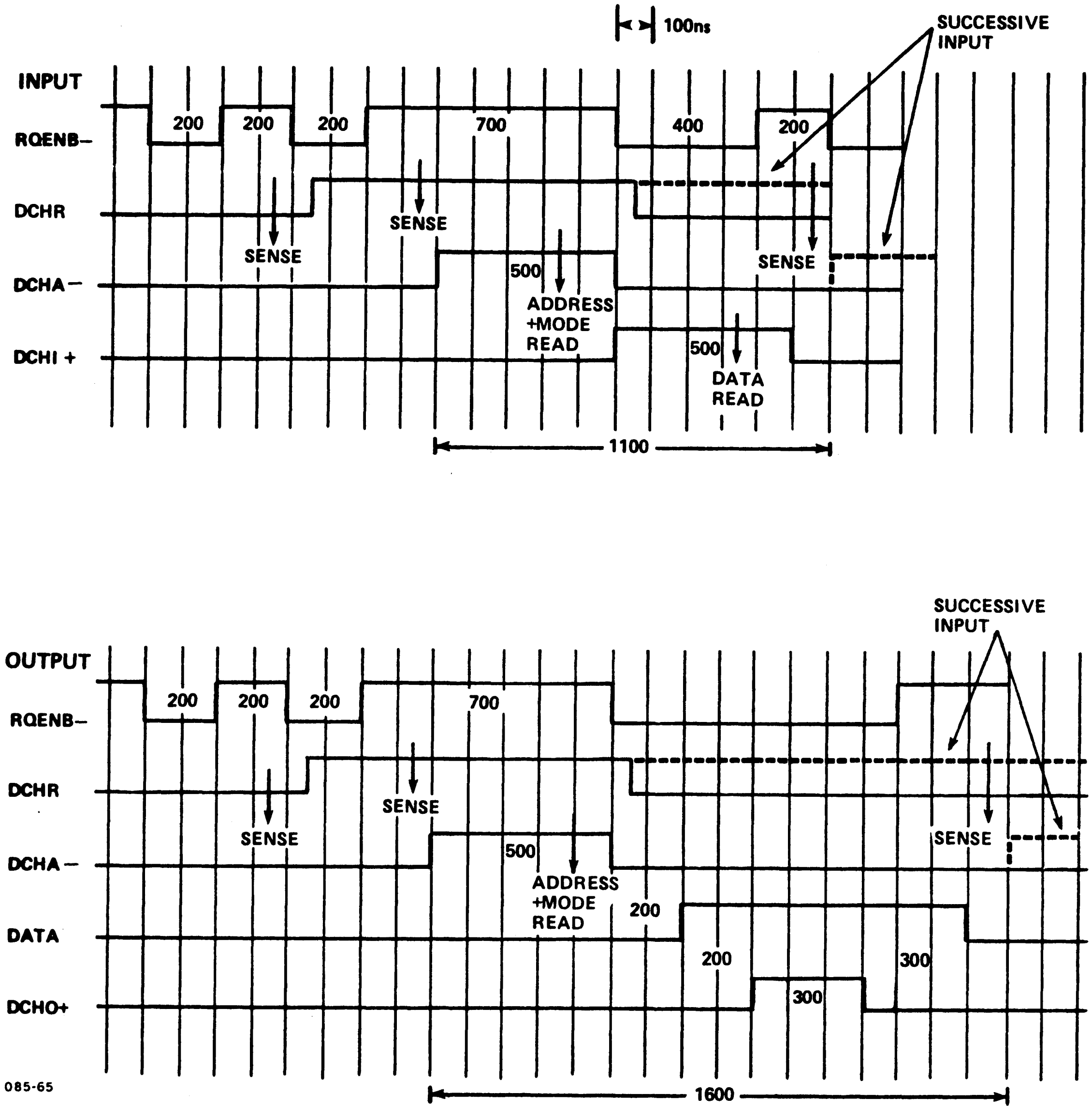


**Figure 4-10. Standard Data Channel Timing for MARK 5**

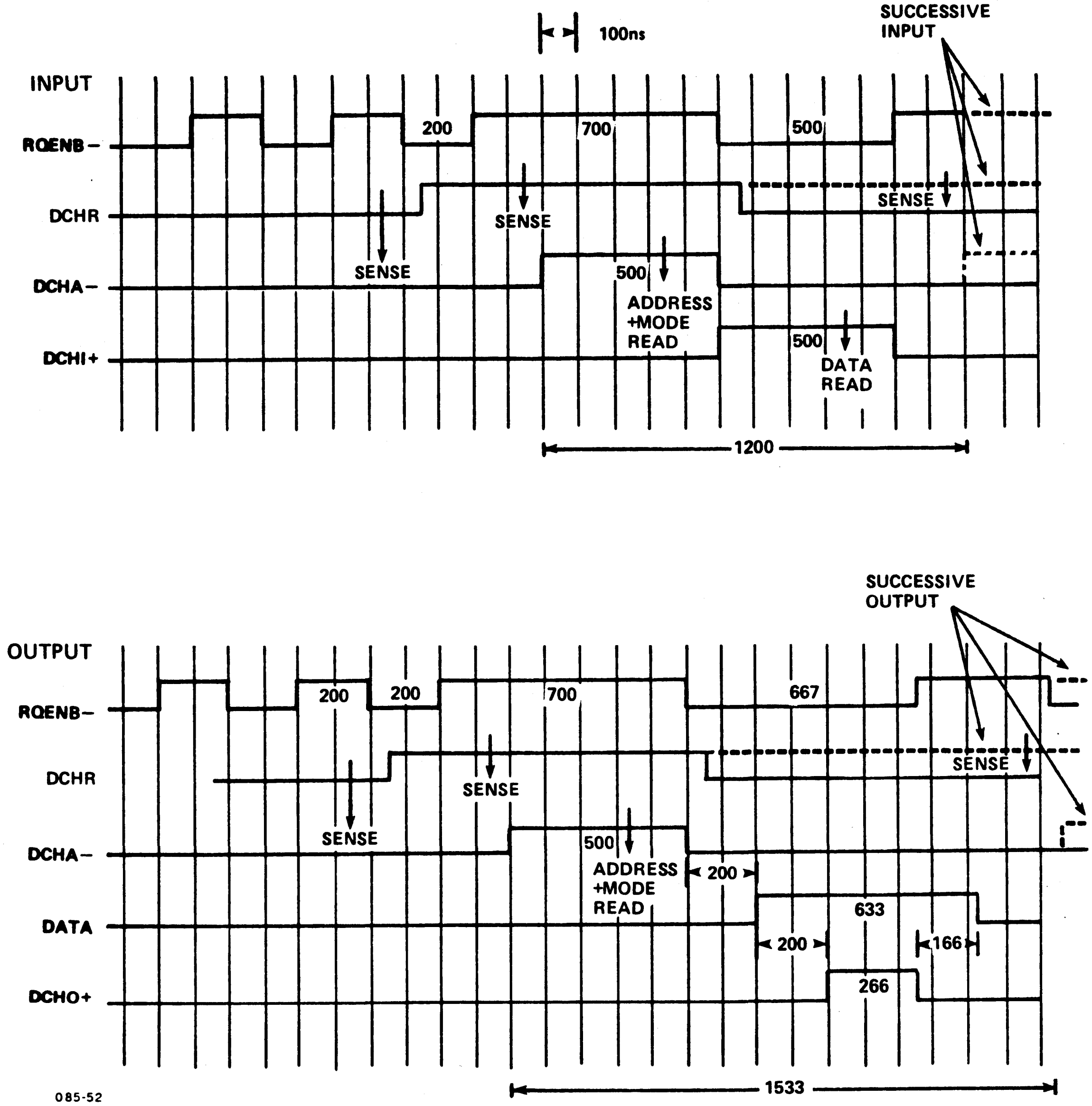


**Figure 4-11. Standard Data Channel Timing for MARK 5 with EIS**

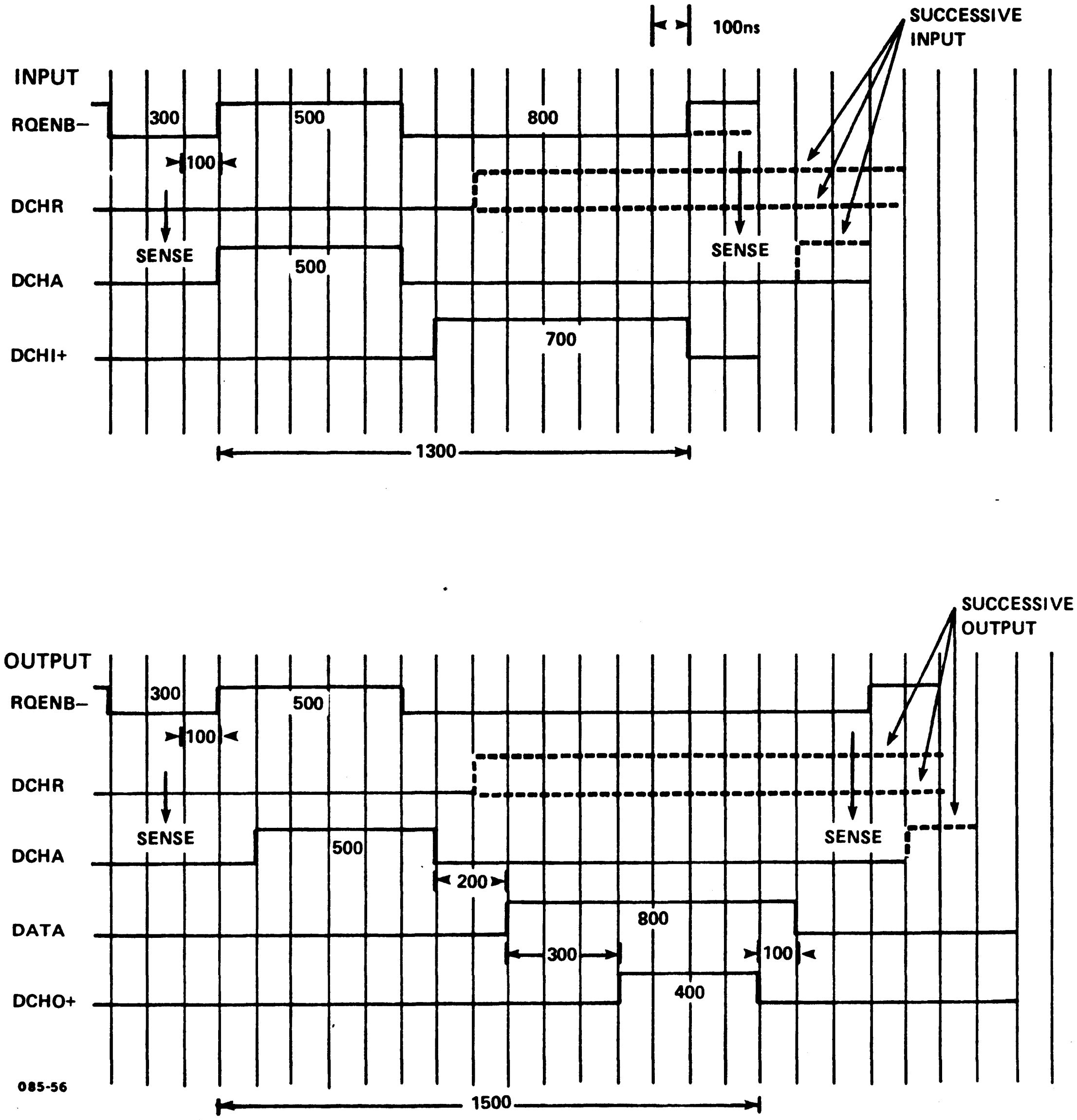


**Figure 4-12. Standard Data Channel Timing for MARK 9**

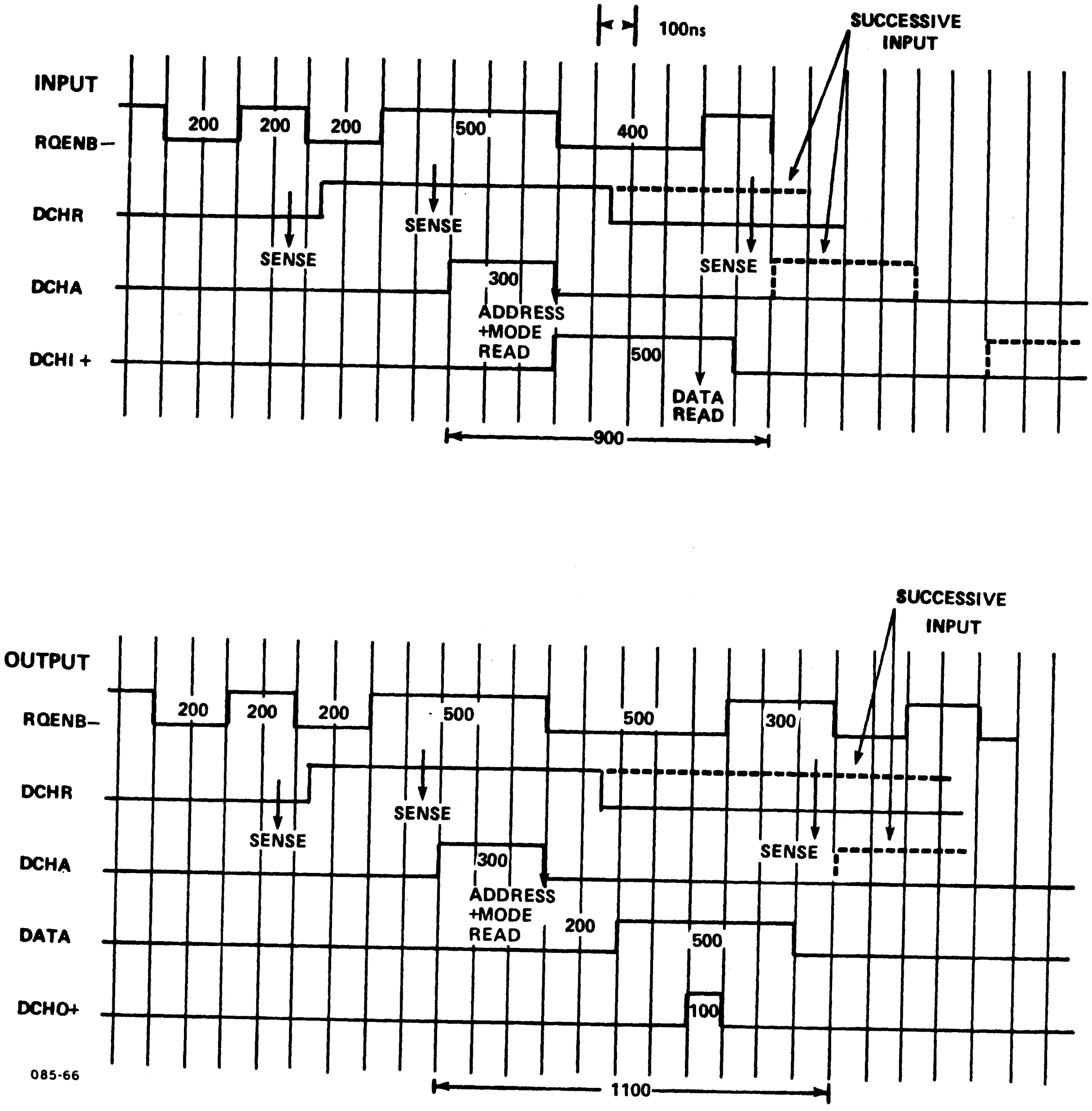


**Figure 4-13. High-Speed Data Channel Timing for MARK 5**

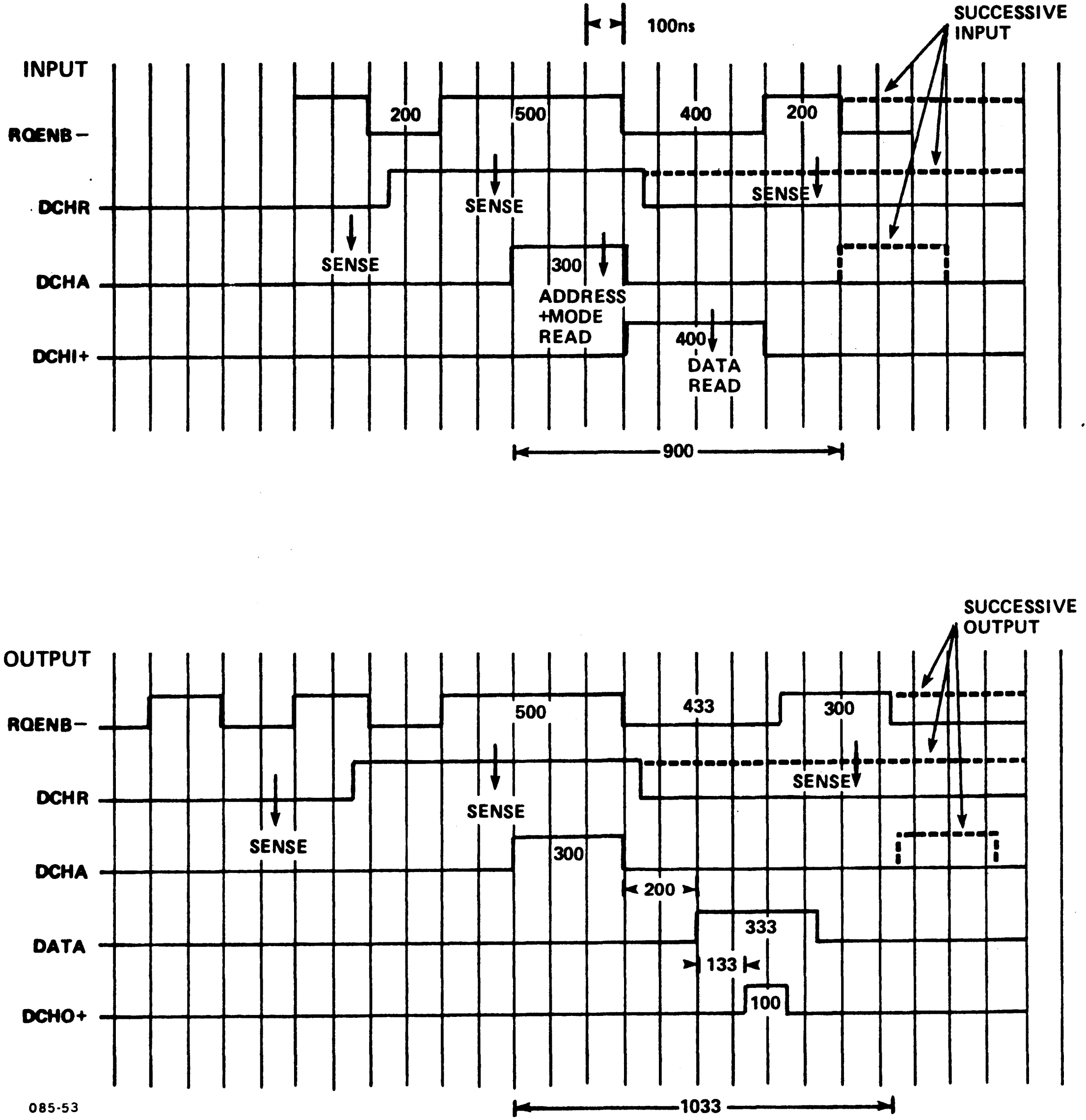


**Figure 4-14. High-Speed Data Channel Timing for MARK 5 with EIS**

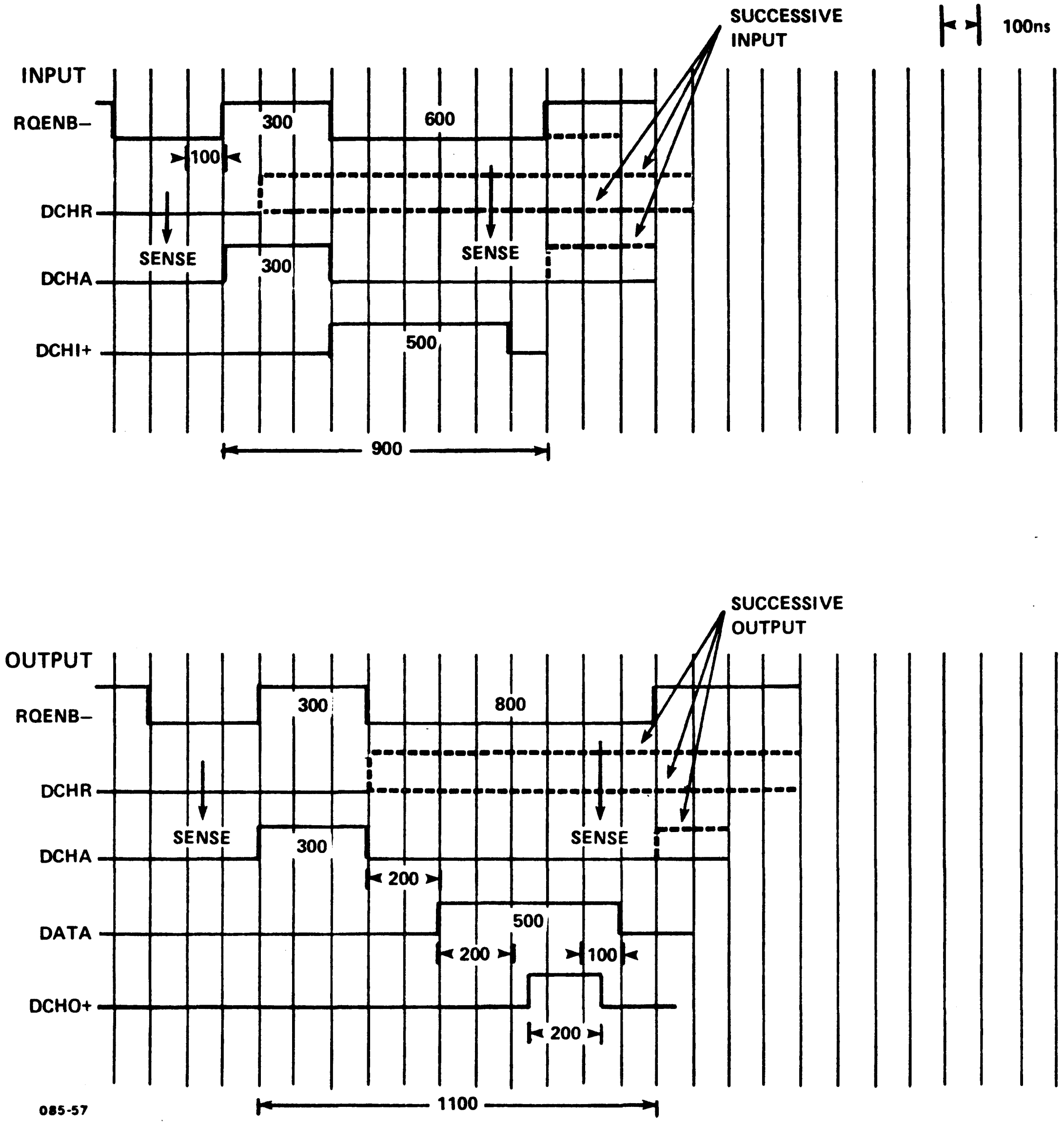


**Figure 4-15. High-Speed Data Channel Timing for MARK 9**

## 4.7 INPUT/OUTPUT CONNECTORS

Four connector sockets may be found on the rear of the POINT 4 MARK 5/9 processor chassis. Figure 4-16 illustrates the positions of these connectors on the processor backplane.

### 4.7.1 POWER CONNECTORS

Two of these sockets on the processor chassis receive power cables carrying power supply voltages from the POINT 4 MARK 5 or MARK 9 Power Supply. These connector sockets are:

| Socket | Mounting Position |
|---|---|
| 12-pin | Top center of the POINT 4 MARK 5/9 backplane |
| 26-pin | Upper right side of POINT 4 MARK 5/9 backplane |

Instructions for the connection of these cables between the POINT 4 MARK 5/9 power supply chassis and the POINT 4 MARK 5/9 processor chassis are found in Section 2.3.4.

### 4.7.2 EXTERNAL I/O DEVICE CONNECTOR

I/O bus signals from external peripheral controllers (i.e., controllers not housed inside the POINT 4 MARK 5/9 chassis) are carried to the processor via a 50-connector cable. The connector for this cable is located on the lower right-hand side of the POINT 4 MARK 5/9 backplane. This I/O connector is high priority. Pin assignments for this connector are shown in Table 4-3.

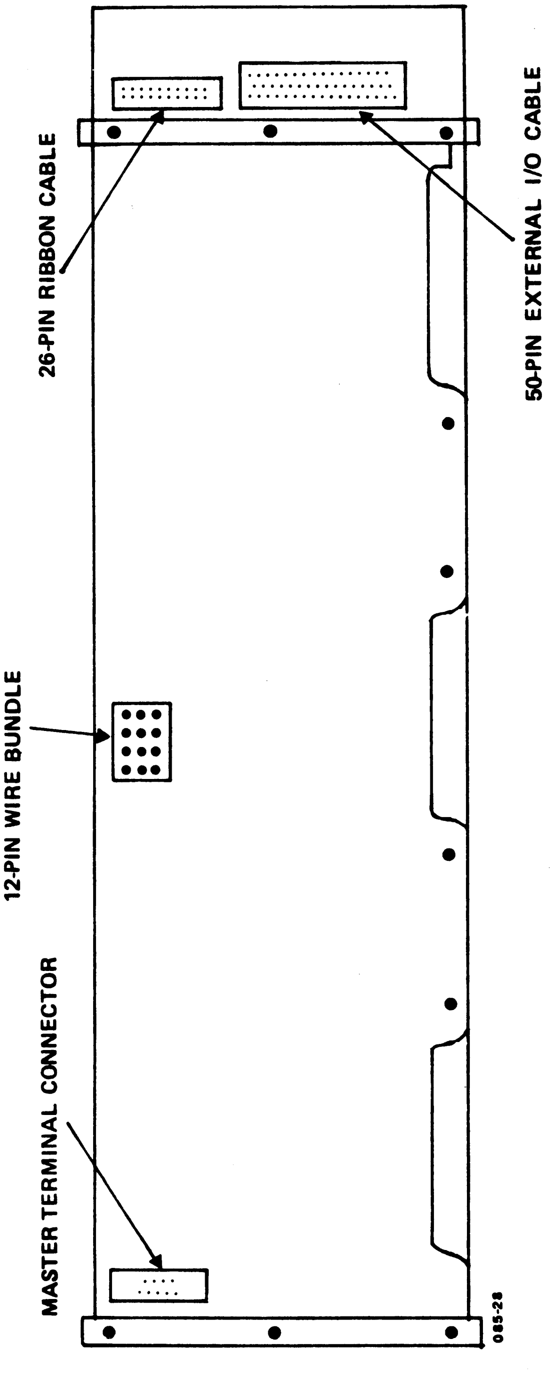


Figure 4-16. Input/Output Connector Locations

## TABLE 4-3. EXTERNAL I/O DEVICE CONNECTOR PIN ASSIGNMENTS

| Pin Number | Signal Name | Backplane Reference |
|---|---|---|
| 1 | GND | A1,2 |
| 2 | CLR+ | A50 |
| 3 | DATA0- | B62 |
| 4 | DATA1- | B65 |
| 5 | DATA2- | B82 |
| 6 | DATA3- | B73 |
| 7 | DATA4- | B61 |
| 8 | DATA5- | B57 |
| 9 | DATA6- | B95 |
| 10 | DATA7- | B55 |
| 11 | DATA8- | B60 |
| 12 | DATA9- | B63 |
| 13 | DATA10- | B75 |
| 14 | DATA11- | B58 |
| 15 | DATA12- | B59 |
| 16 | DATA13- | B64 |
| 17 | DATA14- | B56 |
| 18 | DATA15- | B66 |
| 19 | DATIA+ | A44 |
| 20 | DATIB+ | A42 |
| 21 | DATIC+ | A54 |
| 22 | DATOA+ | A58 |
| 23 | DATOB+ | A56 |
| 24 | DATOC+ | A48 |
| 25 | DCHA- | A60 |
| 26 | DCHI+ | B37 |
| 27 | DCHM0- | B17 |
| 28 | * | B21 |
| 29 | DCHO+ | B33 |
| 30 | DCHPIN- | JUMPER SAVER |
| 31 | DCHR- | JUMPER SAVER |
| 32 | DSO- | A72 |
| 33 | DSI- | A68 |
| 34 | DS2- | A66 |
| 35 | DS4- | A46 |
| 36 | DS4- | A62 |
| 37 | DS5- | A64 |
| 38 | INTA+ | A40 |
| 39 | INTPIN- | JUMPER SAVER |
| 40 | INTR- | JUMPER SAVER |
| 41 | IOPLS+ | A74 |
| 42 | IORST+ | A70 |
| 43 | MSKO- | A38 |
| 44 | * | B38 |
| 45 | RQENB- | B41 |
| 46 | SELB- | A82 |
| 47 | SELD- | A80 |
| 48 | STRT+ | A52 |
| 49 | +5V | A3,4 |
| 50 | GND | A1,2 |

*Not used by POINT 4 MARK 5 or 9 CPU

### 4.7.3 MASTER TERMINAL CONNECTOR

The master terminal can be interfaced to the POINT 4 MARK 5/9 via a cable connected at the upper left-hand corner of the backplane. To use this connector, the controller (standard I/O, device code 10/11) must be in CPU slot 2 or 3 (second or third from top). If the controller board is in slot 3, slot 2 must not be used. Pin assignments for master terminal interface are shown in Table 4-4. Note that POINT 4 MUX boards do not use these pins.

**TABLE 4-4. MASTER TERMINAL INTERFACE PIN ASSIGNMENTS**

| Pin Number | Signal Name | Backplane Reference |
|---|---|---|
| 1 | Not Connected | - |
| 2 | Not Connected | - |
| 3 | TTYIN | 2B69 |
| 4 | -5V | A6 |
| 5 | Not Connected | - |
| 6 | TTYOUT | 2A85 |
| 7 | +15V | A10 |
| 8 | STPBIT- | 2A87 |
| 9 | GND | A1,2 |

# Section 5
# STANDARD INSTRUCTION SET

## 5.1 INTRODUCTION

This section explains the function and use of POINT 4 MARK 5/9 instructions. Included is a discussion of two's complement notation, addressing modes, and the individual instructions in the memory reference, arithmetic/logic, and input/output instruction groups. Input/output instructions and interrupt handling instructions are presented, with details given for special code-77 (CPU) instructions.

## 5.2 OCTAL REPRESENTATION AND TWO'S COMPLEMENT NOTATION

The computer uses 16-bit binary words for program instructions and data. The bits are numbered 0 through 15 with bit 0 the most significant bit (MSB) and bit 15 the least significant bit (LSB). For convenience, binary words are represented in 6-digit octal form. Each octal digit represents three bits and can have values between 0 and 7, except the most significant digit, which represents a single bit and has a maximum value of 1. The POINT 4 MARK 5/9 16-bit binary word format is shown in Figure 5-1.

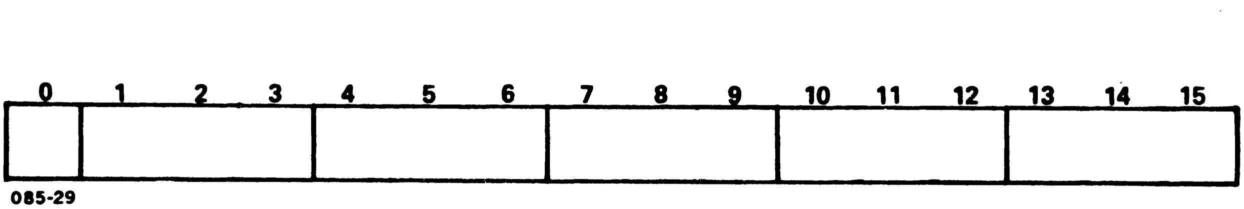


**Figure 5-1. POINT 4 MARK 5/9 16-bit Binary Word Format**

The reader is presumed to be familiar with binary and octal notations. For a simple review, the following example shows the correspondence between decimal, binary, and octal representation:

| Decimal | Binary | Octal |
|---|---|---|
| 0 | 0000000000000000 | 000000 |
| 1 | 0000000000000001 | 000001 |
| 2 | 0000000000000010 | 000002 |
| 8 | 0000000000001000 | 000010 |
| 64 | 0000000001000000 | 000100 |
| 5407 | 0001010100011111 | 012437 |
| 32,767 | 0111111111111111 | 077777 (15 bit max.) |
| 65,535 | 1111111111111111 | 177777 (16 bit max.) |

The computer represents negative numbers in two's complement form. Signed positive and negative numbers are used both as 16-bit operands and as 8-bit address displacements in memory reference instructions. A review of two's complement arithmetic follows.

In two's complement arithmetic, positive and negative values are distinguished by a 0 or 1 in the leftmost bit position (sign bit). Positive numbers have a sign bit of 0, with the numerical value expressed in ordinary binary form by the remaining bits. Negative numbers have a sign bit value of 1 and the numerical value expressed in two's complement form. The two's complement is found by taking the one's complement or logical complement of the number including the sign bit (changing all 0s to 1s and all 1s to 0s) and adding 1.

The number zero is represented by 0s in all bit positions. There is only one representation for zero, since the two's complement of zero is also zero. Zero is a nonnegative value. For this reason also, there is one more negative number than there are nonzero positive numbers.

The range of signed, 8-bit fields is as follows:

|  | Binary Representation | Octal Value |
|---|---|---|
| Largest positive | 01 111 111 | +177 |
|  | 01 111 110 | +176 |
|  | . |  |
|  | . |  |
|  | . |  |
|  | 00 000 001 | +1 |
|  | 00 000 000 | 0 |
|  | 11 111 111 | -1 |
|  | 11 111 110 | -2 |
|  | . |  |
|  | . |  |
|  | . |  |
|  | 10 000 001 | -177 |
| Most negative | 10 000 000 | -200 |

## 5.3 INSTRUCTION TYPES

From the programmer's point of view, the POINT 4 MARK 5/9 Computer contains four accumulators, 64K words memory, and an input/output bus. The instructions control and manipulate the data flowing between these elements.

Instruction words can be classified into one of the following three categories:

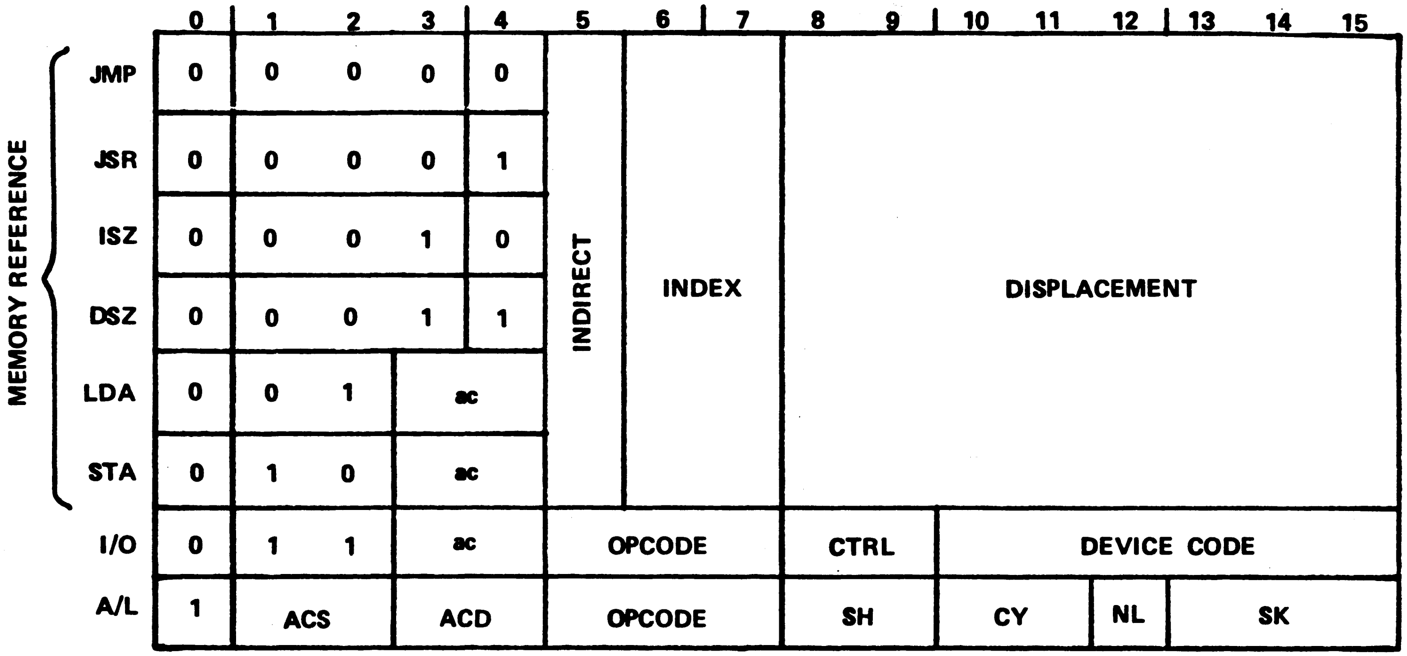
1. Memory reference instructions are instructions that reference a memory location. These include:

   LDA - Load an accumulator from memory
   STA - Store an accumulator into memory
   JMP - Jump to another location in memory
   JSR - Jump to a subroutine in memory
   ISZ - Increment memory and skip if zero
   DSZ - Decrement memory and skip if zero

2. Arithmetic/logic instructions are instructions that specify a particular arithmetic or logical operation to be performed on one or two operands stored in the accumulators, and allow for testing the result for skip conditions.

3. Input/output instructions are instructions for input/output operations with a specific peripheral device.

Figure 5-2 is an overview of the formats for each type of instruction. Each of these three classes is discussed in detail in the following subsections.



| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MEMORY REFERENCE | JMP | 0 | 0 | 0 | 0 | 0 | INDIRECT | INDEX | | DISPLACEMENT | | | | | | | |
| | JSR | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | | |
| | ISZ | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | | |
| | DSZ | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | |
| | LDA | 0 | 0 | 1 | ac | | | | | | | | | | | | |
| | STA | 0 | 1 | 0 | ac | | | | | | | | | | | | |
| | I/O | 0 | 1 | 1 | ac | | OPCODE | | | CTRL | | DEVICE CODE | | | | | |
| | A/L | 1 | ACS | | ACD | | OPCODE | | | SH | | CY | | NL | SK | | |

| | | |
|---|---|---|
| ac | = | ACCUMULATOR |
| CTRL | = | CONTROL PULSE |
| ACS | = | SOURCE ACCUMULATOR |
| ACD | = | DESTINATION ACCUMULATOR |
| SH | = | SHIFT CONTROL |
| CY | = | CARRY PRESELECTION |
| NL | = | NO-LOAD |
| SK | = | SKIP CONDITION |

085-30

Figure 5-2. POINT 4 MARK 5/9 Instruction Format Summary

## 5.4 MEMORY REFERENCE INSTRUCTIONS

Six memory reference instructions are used to move data between memory locations and accumulators, to transfer program control to a new location, and to modify and test memory words. The memory reference instructions fall into three general categories, as follows:

1. Move data instructions: LDA, STA
2. Jump instructions: JMP, JSR
3. Modify memory instructions: ISZ, DSZ

Before describing the function of each instruction in this group, it is necessary to describe the way in which they address memory.

### 5.4.1 MEMORY ADDRESSING

Each memory reference instruction uses one of several addressing modes to determine an effective memory address, E. The processor accesses the location specified by the effective memory address and uses the contents as the operand of the instruction.

The jump instructions (JMP, JSR) and the modify memory instructions (ISZ, DSZ) both use the binary format shown in Figure 5-3.

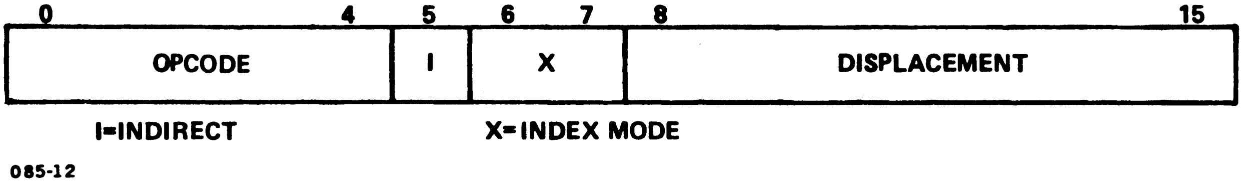


**Figure 5-3. Jump and Modify Memory Instruction Binary Word Format**

Bits 0-4 of the instruction word are the OPCODE field. Bit 5 is the indirect or I field; bits 6 and 7 are the index mode or X field; and bits 8-15 are the displacement or D field.

The move data instructions (LDA, STA) use the binary format shown in Figure 5-4.

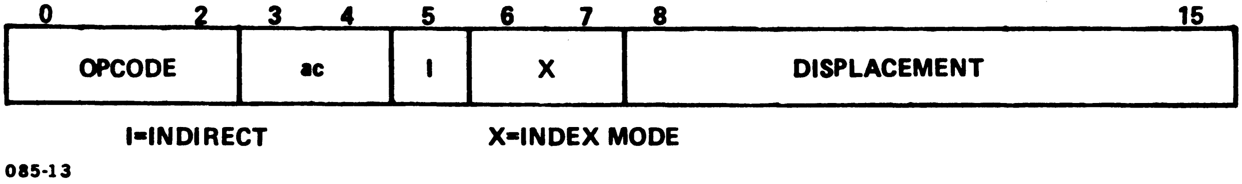


**Figure 5-4. Move Data Instruction Binary Word Format**

All addresses, both direct and indirect, are entered into the Effective Address Register. When this register contains the effective address E, the instruction specified in bits 0 through 4 of the command is executed.

Table 5-1 defines the memory reference instructions, indicating bits, their fields, and field definitions.

**TABLE 5-1. MEMORY REFERENCE INSTRUCTIONS**

| Bits | Field | Definition |
|---|---|---|
| 0-2 | OPCODE | Determines which instruction is performed. For the move data instructions (LDA, STA), bits 0, 1 and 2 make up the OPCODE field. For the jump instructions (JMP, JSR), and modify memory instructions (ISZ, DSZ), bits 0-4 make up the OPCODE field. |
| 3,4 | Accumulator (ac) | The ac field for the Move Data instructions (LDA, STA) specifies one of the four general accumulators. The specified accumulator will either have data stored in it or data transferred from it. |
| 5 | Indirect (I) | Determines whether the X and D fields specify the effective address (E) directly or whether indirect addressing is to be used. |
| 6,7 | Index Mode (X) | Defines one of the four addressing modes. Each addressing mode may be thought of as a page of 256 words that the instruction can address directly. |
| 8-15 | Displacement (D) | Specifies the word addressed on the selected page. |

#### 5.4.1.1 Indexing Mode

The X field selects one of the indexing modes shown in Table 5-2.

**TABLE 5-2. INDEXING MODES**

| Bits 6-7 | Definition |
|---|---|
| 00 | Page zero - defined as the first 256 memory locations (addresses in the range from 000000 to 000377 octal). The effective memory address in page zero addressing is equal to the value of the D field, which is an unsigned binary integer that can have values from 000 octal to 377 octal. |
| 01 | Relative addressing - in the relative addressing mode, the address placed in the Effective Address Register is equal to the address in the program counter (PC) plus the value of the displacement in the D field. In this case, the displacement D is a signed binary integer. Bit 8 is the sign (0 = positive 1 = negative) and the integer may have any value in the range from -200 to +177 octal (decimal -128 to +127). The address in PC can be visualized as the center of a 256-word page and any address between the bottom (128 words below the PC) and top (127 words above PC) of the page can be specified by the displacement D. |
| 10<br>or<br>11 | Base register addressing - in the base register addressing mode, the address placed in the Effective Address Register is equal to the address in accumulator register A2 (code 10) or A3 (code 11) plus the value of the displacement in the D field. In this case the displacement D is a signed binary integer. Bit 8 is the sign (0 = positive, 1 = negative) and the integer may have any value in the range from -200 octal to +177 octal (decimal -128 to +127). The address in A2 or A3 can be visualized as the center of a 256-word page and any address between the bottom (128 words below A2 or A3) and top (127 words above A2 or A3) of the page can be specified by the displacement D. |

### 5.4.1.2 Indirect Addressing Operations

When the I field (bit 5) of the memory reference instruction contains a 1, an indirect addressing sequence is required. In this case, the address in the Effective Address Register (determined by the X and D fields) is the memory address from which a second address word is to be fetched.

This address word can be interpreted as follows:

- If the most significant bit of the address word equals 0, this address word is the effective address, E.
- If the most significant bit of the address word equals 1, the action taken depends on whether the processor is set in 32K or 64K-addressing mode, as described below.

When the processor is in 32K (normal) mode, a second level of indirect addressing is allowed. In this case, if the most significant bit (bit 0) of the address word fetched from memory contains a 1, another level of indirect addressing is required, and the address word in the Effective Address Register specifies the address word to be fetched from memory. The process continues until an address word with bit 0=0 is found. Through programming error, it is possible to become caught in an infinite loop of indirect addressing. Caution should be taken to avoid this problem when using indirect addressing.

When the processor is in 64K mode, a second level of indirect addressing is not permitted. In this case, all 16 bits of the word fetched from memory are used as the effective address. A 1 in bit 0 of the address word simply indicates an address in the upper 32K words (100000-177777) of memory.

#### 5.4.1.3 Automatic Incrementing and Decrementing of Locations

If at any time during the indirect addressing sequence the Effective Address Register contains an address in the range from 000020 octal to 000037 octal, the following auto-indexing action is performed:

1. The contents of the memory location specified by the Effective Address Register are fetched, and the contents are either incremented or decremented by one, as follows:

   - If the address is in the range 000020 octal through 000027 octal, the contents are incremented.

   - If the address is in the range 000030 octal through 000037 octal, the contents are decremented.

2. The incremented or decremented value is written back into the same memory location from which the value was fetched in step 1.

3. The incremented or decremented value produced in step 1 is stored in the Effective Address Register and used for the next level of indirect addressing (if bit 0=1 and the processor is set to 32K-addressing mode), or for the effective address (if bit 0=0 or the processor is in 64K-addressing mode).

**NOTE**

> The value of bit 0 _following_ incrementing or decrementing controls continuation of indirect addressing.

### 5.4.2 TYPES OF MEMORY REFERENCE INSTRUCTIONS

When the Effective Address Register contains the effective address E, one of two groups of memory reference instructions is performed as determined by the operation codes. Refer to Section 5.4.1 for basic memory reference instruction formats and field definitions. Refer to Appendix A (Von Neumann Map of the POINT 4 MARK 5/9 Command Structure) for octal formats of each instruction, and to Appendix B (POINT 4 MARK 5/9 Instruction Reference Chart) for octal-to-symbolic conversion of memory reference instructions.

#### 5.4.2.1 Move Data Instructions

When the code in bits 0-2 of the OPCODE field is not 000, and the effective address (E) is in the Effective Address Register, two operations are performed, as shown in Table 5-3.

TABLE 5-3. MOVE DATA INSTRUCTIONS

| Bits 0-2 | OPCODE | Definition |
|---|---|---|
| 001 | LDA | Load accumulator instruction - the contents of memory location E are stored in the accumulator specified by the ac field (bits 3 and 4). The contents of E are unaffected; the original contents of the accumulator are lost. |
| 010 | STA | Store accumulator instruction - the data in the accumulator specified by the ac field is transferred to memory location E. The contents of the accumulator are unaffected; the original contents of E are lost. |

### 5.4.2.2 Jump and Modify Memory Instructions

When bits 0, 1 and 2 are all zero, and the effective address (E) is in the Effective Address Register, one of four operations is performed. The operation is specified by the code in bits 3 and 4 of the OPCODE extension field. These jump and modify memory instructions are shown in Table 5-4.

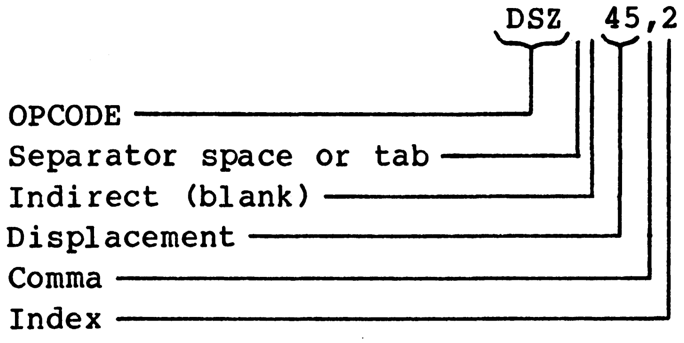
**TABLE 5-4. JUMP AND MODIFY MEMORY INSTRUCTIONS**

| Bits 0-4 | OPCODE | Definition |
|---|---|---|
| 00000 | JMP | Jump instruction - the effective address E is transferred from the Effective Address Register to the program counter (PC). The next instruction is then fetched from jump address E and sequential execution is continued from there. |
| 00001 | JSR | Jump to subroutine instruction - after the effective address E has been calculated, the address in PC is incremented and the incremented value is stored in accumulator A3. Then the effective address E is transferred from the Effective Address Register to the program counter (PC). The next instruction is then fetched from jump address E. Execution of another JMP or JSR instruction that specifies A3 will cause the program to return to the address in A3 plus or minus any desired displacement D. |
| 00010 | ISZ | Increment and skip if zero - the contents of effective address E are fetched, incremented, and written back into address E. If the incremented value is equal to zero, PC is incremented by one to skip the next instruction. |
| 00011 | DSZ | Decrement and skip if zero - the contents of the location specified by effective address E are decremented and written back into address E. If the decremented value is equal to zero, PC is incremented by one to skip the next instruction. |

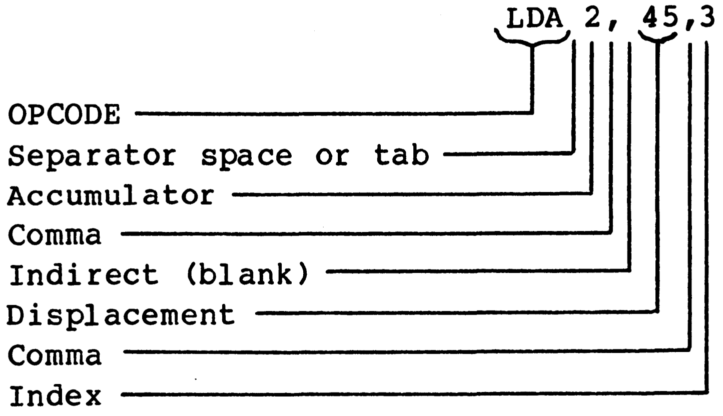
### 5.4.2.3 Assembler Language Conventions and Addressing Examples

The assembler language memory reference instruction consists of the instruction OPCODE mnemonic (STA, LDA, JMP, etc.) followed by symbols that specify the accumulator, the addressing mode, and the memory address. The assembler program translates these statements into binary code which the processor executes. Table 5-5 shows the programming conventions for memory reference instructions.

The format for modify memory and jump instructions requires the instruction mnemonic and a memory address, including displacement, indexing, and indirect addressing indicators. The assembly language instruction will be formatted as follows:



The move data instructions LDA and STA also require that an accumulator (A0-A3) be specified. For example



Fields that are not specified will be assembled containing 0s. An "@" symbol denotes indirect addressing and places a 1 in bit 5 of the instruction. An example of indirect page-zero (X Field = 00) addressing is as follows:

```
LDA 1,@20
```

Relative addressing is formatted as follows:

```
LDA 0,. +15
```

The symbol "." indicates X = 01 (relative addressing) and thus "." represents the current value of the program counter.

## TABLE 5-5. ASSEMBLER LANGUAGE CONVENTIONS FOR MEMORY REFERENCE INSTRUCTIONS

| Instruction Function | OPCODE Mnemonic | Separator Space or Tab | Accumulator Number | , | Memory Address | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | Indirect | Displcmt | , | Index |
| Load accumulator | LDA | | ac* | , | blank<br>or<br>@ | Displacement | , | blank<br>1**<br>2<br>3 |
| Store accumulator | STA | | | | | | | |
| Jump | JMP | | none | | | | | |
| Jump subroutine | JSR | | | | | | | |
| Increment and skip if Zero | ISZ | | | | | | | |
| Decrement and skip if zero | DSZ | | | | | | | |

*ac = 0, 1, 2, 3 representing A0, A1, A2, A3

**Instead of "displacement,1" the following sequence may be used: ".±displacement"

## 5.5 ARITHMETIC/LOGIC INSTRUCTION GROUP

The eight arithmetic/logic instructions perform binary addition, subtraction and logical functions on 16-bit operands. These instructions are:

- Arithmetic: ADD, ADC, INC, SUB, NEG
- Logic: MOV, COM, AND

All arithmetic and logic instructions contain a 1 in bit 0 and have their basic arithmetic/logic unit (ALU) function specified by bits 5-7, as shown in Figure 5-5. The fields of the arithmetic/logic instruction format are as follows:

- Source accumulator (ACS)
- Destination accumulator (ACD)
- OPCODE
- Shifter/swapper (SH)
- Carry preselect (CY)
- No-load (NL)
- Skip condition tester (SK)

| 0 | 1 2 | 3 4 | 5 7 | 8 9 | 10 11 | 12 | 13 15 |
|---|---|---|---|---|---|---|---|
| 1 | ACS | ACD | OPCODE | SH | CY | NL | SK |

085-14

**Figure 5-5. Arithmetic/Logic Instruction Format**

### 5.5.1 ARITHMETIC/LOGIC PROCESSING

The organization of the arithmetic/logic processing unit must be described before discussion of the eight arithmetic and logic instructions and their auxiliary control fields. Subsystem organization is shown in Figure 5-6 and described in the following subsections.

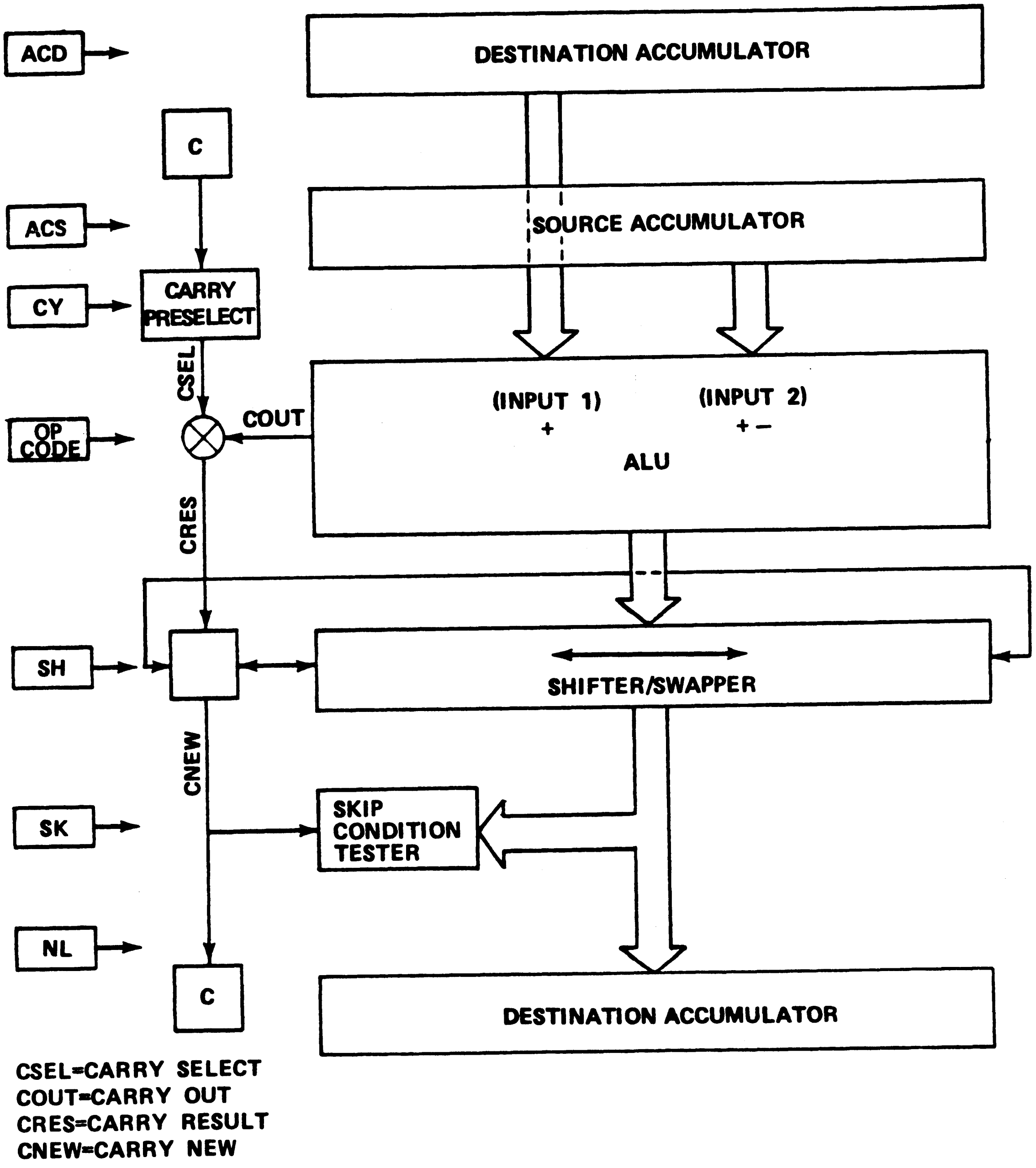


Figure 5-6. Arithmetic/Logic Operations

### 5.5.1.1 Arithmetic/Logic Operations

The heart of the subsystem is the arithmetic/logic unit (ALU), which performs the actual addition, subtraction, or logical operation. It has provision for two inputs:

- Input 1 - comes from the accumulator selected by the ACD field and is used only in the operations that require two operands (ADD, SUB, ADC, and AND).
- Input 2 - comes from the accumulator selected by the ACS field and is used in all operations.

The ALU performs the arithmetic or logical operation specified by the OPCODE field (bits 5-7). The result of this operation may cause a carry-out (COUT) to occur from the most significant bit of the ALU. In the case of an operation that adds unsigned integers, a carry-out is equivalent to overflow; however, this is not always true. See Section 5.5.1.2 for a more complete discussion of carry and overflow operation.

If the result of the arithmetic or logical operation involves a carry-out, the carry preselected by the CY field of the instruction (CSEL) is complemented. The resulting carry (CRES) along with the 16-bit operation result generated by the ALU is applied as a 17-bit operand to the shifter, where a shift-left, shift-right, or swap may occur as determined by the SH field of the instruction. After shifting, the carry (CNEW) and the 16-bit operation result are loaded into the Carry flag (C) and the destination accumulator (ACD) unless this is prevented by a 1 in the No-Load (NL) field. In either case, they are tested for a skip condition (i.e., to determine if the next instruction should be skipped) as specified in the SK field of the instruction.

### 5.5.1.2 Overflow and Carry-Out Operations

The 16-bit numbers processed by the ALU may be thought of as unsigned integers between 0 and 65535 or as signed integers between -32768 and +32767.

| Binary Number | Unsigned Interpretation | | Signed Interpretation | |
|---|---|---|---|---|
| (in ALU) | Octal | Decimal | Octal | Decimal |
| 1111111111111111 | 177777 | 65535 | -00001 | -1 |
| 1111111111111110 | 177776 | 65534 | -00002 | -2 |
| . | | | | |
| . | | | | |
| . | | | | |
| 1000000000000001 | 100001 | 32769 | -77777 | -32767 |
| 1000000000000000 | 100000 | 32768 | -100000 | -32768 |
| 0111111111111111 | 077777 | 32767 | +77777 | 32767 |
| 0111111111111110 | 077776 | 32766 | +77776 | 32766 |
| . | | | | |
| . | | | | |
| . | | | | |
| 0000000000000001 | 000001 | 1 | +00001 | 1 |
| 0000000000000000 | 000000 | 0 | 00000 | 0 |

When working with either interpretation, there is the possibility of an overflow (answer greater than the maximum number that can be represented) or underflow (less than the minimum). In general, the ALU will produce the correct result if no overflow or underflow occurs, and will produce 65536 more than or less than the correct result if there is underflow or overflow, respectively.

The relationship between underflow/overflow and the carry-out from the ALU MSB is shown in the following paragraphs.

1. Unsigned integers:

   Decimal: 0 <= x <= 65535
   Octal: 0 <= x <= 177777

   When ADDing two numbers, if the true result is less than or equal to 65535, the ALU will produce the correct result and no carry-out will result. If the true result is greater than 65535, the ALU will produce 65536 less than the true result (i.e., the true result truncated to 16 bits) and a carry-out will result. Note that in these cases, a carry-out is synonymous with overflow and indicates that the ALU output is not the true result.

   SUBtraction is accomplished in the ALU by complementing the subtrahend and adding it to the minuend with a carry-in. Therefore, when SUBtracting one unsigned integer from another, if the true result is positive or zero, the ALU will produce the true result and will also produce a carry-out.

If the true result is negative, the ALU will produce the true result plus 65536 (since all numbers are interpreted as positive) and no carry-out will result. Note that, in these cases, a carry-out is the opposite of underflow and indicates that the ALU output is the true result.

2. Signed integers:

   Decimal: -32768 <= x <= 32767
   Octal: -100000 <= x <= 77777

   When ADDing two positive integers (or SUBtracting a negative integer from a positive one), if the true result is less than or equal to 32767, the ALU will produce the true result and no carry-out. If the true result is greater than 32767, the ALU output will appear negative (since the MSB=1), will be 65536 less than the true result, and no carry-out will occur. Note that, in this case, an overflow is not signaled by a carry-out.

   When ADDing two integers with opposite signs (or SUBtracting two numbers having the same sign) the ALU will always produce the true result since the true result must be between -32768 and +32767. A carry-out will occur if the result is positive and not if it is negative.

   When ADDing two negative numbers (or SUBtracting a positive number from a negative one), if the true result is greater than or equal to -32768, the ALU will produce the true result. If the true result is less than -32768, the ALU output will appear positive (MSB=0) and will be 65536 greater than the true result. In either case, a carry-out will always occur.

   These relationships are illustrated in Figure 5-7.

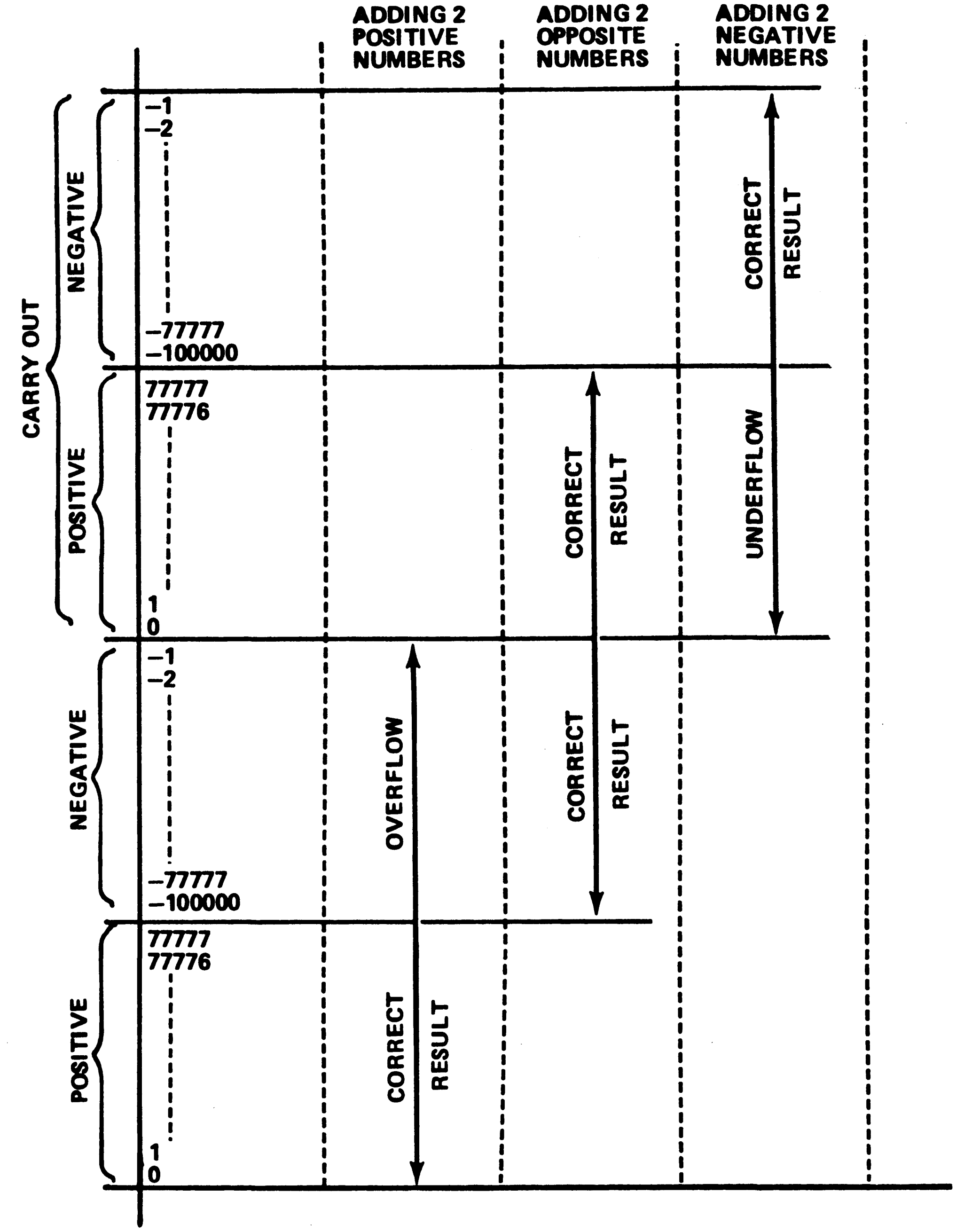


Figure 5-7. Overflow and Carry Operations Analysis for Signed Integers

### 5.5.2 ARITHMETIC/LOGIC FUNCTIONS

The OPCODE field (bits 5 through 7) defines one of eight arithmetic/logic operations to be performed by the 16-bit ALU, as shown in Table 5-6.

TABLE 5-6. ARITHMETIC/LOGIC FUNCTIONS

| Bits 5-7 | OPCODE | Definition |
|---|---|---|
| 000 | COM | Complement - complement the contents of ACS. Do not modify the preselected carry bit. |
| 001 | NEG | Negate - produce the two's complement of the contents of ACS. If ACS=0, complement the preselected carry bit. |
| 010 | MOV | Move - supply the unmodified contents of ACS. Do not modify the preselected carry bit. |
| 011 | INC | Increment - add 1 to the contents of ACS. If the result is 0, complement the preselected carry bit. |
| 100 | ADC | Add complement - add the complement of ACS to ACD. Complement the preselected carry bit if ACS is less than ACD.* |
| 101 | SUB | Subtract - subtract ACS from ACD. Complement the preselected carry bit if ACS is less than or equal to ACD.* |
| 110 | ADD | Add - add the contents of ACS to the contents of ACD. If the unsigned sum is greater than or equal to 2 to the 16th power, complement the preselected carry bit. |
| 111 | AND | And - logically AND the contents of ACS with the contents of ACD. Do not modify the preselected carry bit. |
| *Using a 16-bit unsigned integer interpretation. | | |

### 5.5.3 SECONDARY FUNCTIONS

The shift (SH), carry (CY), no-load (NL), and skip (SK) fields specify secondary operations performed on the ALU result produced by the OPCODE field. These fields are discussed in the following sections.

#### 5.5.3.1 Shift Field (SH)

The SH field (bits 8 and 9) determines the shifting action (if any) produced by the shifter on the result of the calculation produced by the ALU, as shown in Table 5-7.

TABLE 5-7. SHIFT FIELD DEFINITIONS

| Bits 8-9 | Mnemonic | Definition |
|---|---|---|
| 00 | - | No shift - do not modify the ALU result. The carry resulting from the ALU operation is unaffected. |
| 01 | L | Left rotate - shift the result one place to the left and insert the state of the carry resulting from the ALU (CRES) in the LSB (bit 15) position. Insert the out-shifted MSB (bit 0) into the carry bit (CNEW). |
| 10 | R | Right rotate - shift the result one place to the right and insert the state of the carry resulting from the ALU (CRES) into the MSB (bit 0) position. Insert the out-shifted LSB (bit 15) into the carry bit (CNEW). |
| 11 | S | Swap - swap the eight MSBs of the result with the eight LSBs. The carry resulting from the ALU is unaffected. |

### 5.5.3.2 Carry Control Field (CY)

The CY field (bits 10 and 11) specifies the base to be supplied to the ALU for carry calculation, as shown in Table 5-8.

TABLE 5-8. CARRY CONTROL FIELD

| Bits 10-11 | Mnemonic | Definition |
|---|---|---|
| 00 | - | No change - the current state of the Carry flag is supplied to the ALU as a base for carry calculation. |
| 01 | Z | Zero - the value 0 is supplied to the ALU as a base for carry calculation. |
| 10 | O | One - the value 1 is supplied to the ALU as a base for carry calculation. |
| 11 | C | Complement - the complement of the current state of the Carry flag is supplied to the ALU as a base for carry calculation. |

The three logical functions (MOV, COM, AND) supply the values listed above as the carry bit to the shifter. The five arithmetic functions (ADD, ADC, INC, SUB, NEG) supply the complement of the base value if the ALU operation produces a carry-out of bit 0; otherwise, they supply the value listed above.

### 5.5.3.3 No-Load Field (NL)

The NL field (bit 12) determines whether the output of the shifter is stored in ACD and in carry. If bit 12=0, the shifter output is stored in ACD and in carry. If bit 12=1, no storage action occurs.

### 5.5.3.4 Skip Control Field (SK)

The SK field determines the type of skip test to be performed on the shifter output. If the selected skip test is affirmative, the next instruction is skipped. The skip tests that can be selected by the SK field (bits 13-15) are shown in Table 5-9.

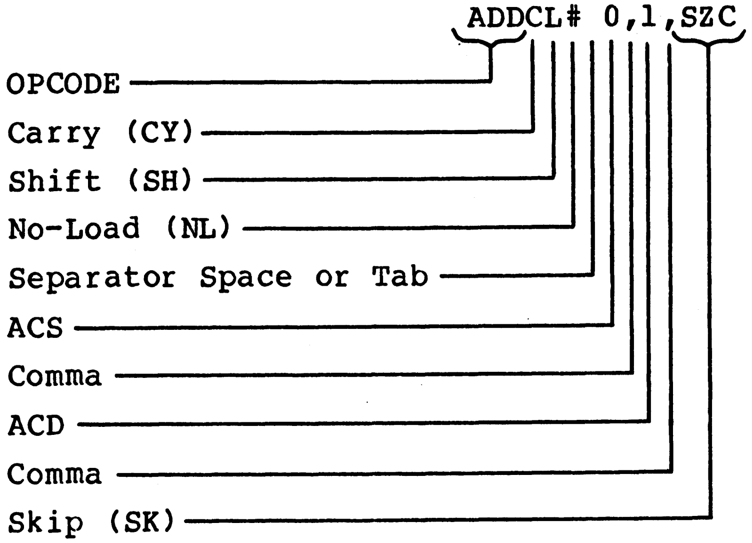
TABLE 5-9. SKIP CONTROL FIELD

| Bits 13-15 | Mnemonic | Definition |
|---|---|---|
| 000 | - | No skip test (never skip) |
| 001 | SKP | Skip unconditionally (no skip test required) |
| 010 | SZC | Skip if carry bit is zero |
| 011 | SNC | Skip if carry bit is nonzero |
| 100 | SZR | Skip if result is zero |
| 101 | SNR | Skip if result is nonzero |
| 110 | SEZ | Skip if either carry bit or result is zero |
| 111 | SBN | Skip if both carry bit and result are nonzero |

### 5.5.4 ASSEMBLER LANGUAGE CONVENTIONS AND EXAMPLES

The assembler language arithmetic or logic instruction consists of the instruction OPCODE mnemonic (ADD, NEG, COM, etc.) followed by symbols that specify the carry indicator, the shift indicator, the load/no-load indicator, a source and a destination accumulator, and the skip conditions. Table 5-10 shows the programming conventions for arithmetic and logical instructions.

The format is as follows:



The CY, SH, NL, and SK fields are specified by adding the appropriate mnemonic symbols. None of these four fields has to be specified, but their symbols must appear in the proper order and place, if they are included. Those fields not specified will be assembled containing 0s. For example,

```
    ADDCL 0,1
```

performs the following operation: Add A0 to A1 and supply the complement of the Carry flag to the ALU. Shift the 17-bit output to the left, and store it into A1 and the Carry flag.

## TABLE 5-10. ASSEMBLER LANGUAGE CONVENTIONS FOR ARITHMETIC AND LOGIC INSTRUCTIONS

| Instruction Function | OPCODE Mnemonic | Optional Secondary Functions | | | Separator Space or Tab | Accumulators | | | , | Optnl Skip SK* |
|---|---|---|---|---|---|---|---|---|---|---|
| | | CY* | SH* | NL* | | ACS | , | ACD | | |
| Add | ADD | | | | | | | | | blank |
| Subtract | SUB | | | | | | | | | SKP |
| Move | MOV | none | none | | | 0 | | 0 | | SZC |
| Increment | INC | Z | L | none | | 1 | , | 1 | , | SNC |
| Negate | NEG | O | R | # | | 2 | | 2 | | SZR |
| Complement | COM | C | S | | | 3 | | 3 | | SNR |
| Add Complement | ADC | | | | | | | | | SEZ |
| Logical And | AND | | | | | | | | | SBN |

*Omission of a mnemonic symbol for these fields will cause the field to be assembled as all zeros.

## 5.6 INPUT/OUTPUT INSTRUCTION GROUP

The input/output instructions enable the processor to communicate with the peripheral devices on the system and also perform various operations within the processor. I/O instructions transfer data between accumulators and devices, start or reset device operation, or check the status of each device. Each I/O instruction contains a 6-bit device code field that specifies the particular device for this data transfer. The system allows up to 63 peripheral devices. Each device is assigned a unique code from 00 through 76 octal. The 77 octal code denotes a special class of instructions that controls certain CPU functions such as interrupt handling. Use of the 00 code is not recommended, since a device with that code would give a default response to an interrupt acknowledge instruction.

All instruction words in this category have the format shown in Figure 5-8.

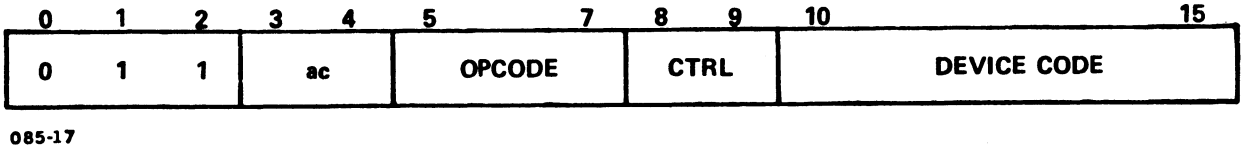


**Figure 5-8. Input/Output Instruction Format**

An instruction in this class is designated by 011 in bits 0-2. The OPCODE and control (CTRL) fields define the I/O operation to be performed. If a data transfer operation is involved, the ac field (bits 3 and 4) specifies the accumulator involved in the data transfer (otherwise it has no effect). Bits 10-15 select the device that is to respond to the instruction.

### 5.6.1 REGULAR I/O INSTRUCTIONS

Regular I/O instructions apply to all device codes except code 77. These instructions fall into two basic categories, depending on whether they transfer data or test the state of the device:

1. I/O transfer: NIO, DIA, DOA, DIB, DOB, DIC, DOC

2. I/O skip: SKPBN, SKPBZ, SKPDN, SKPDZ

The POINT 4 MARK 5/9 input/output system provides for specification of the following functions:

- Full 16-bit data transfer:
  - Three input channels (DIA, DIB, DIC)
  - Three output channels (DOA, DOB, DOC)

- Control functions:

  - Three outgoing control pulses (START, CLEAR, or IOPULSE - designated by S, C, or P, respectively).

  - Two device flags (Busy and Done) that can be sensed by I/O skip instructions

These input/output functions are illustrated in Figure 5-9.

Each device interface contains a 6-bit address decoder (bits 10-15). When the processor executes an I/O instruction, it places the specified device code onto the Device Select lines of the I/O bus. The appropriate device will recognize its own code and thus respond to the I/O instruction. All other devices ignore the instruction.

The control (CTRL) field can have two different functions, depending on the category into which the instruction falls:

- In conjunction with a data transfer instruction, one of the three different control pulses may be sent to the device--START, CLEAR, or IOPULSE

- In conjunction with I/O skip instructions, the control field determines which of the two flags in the I/O device will be tested--Busy or Done

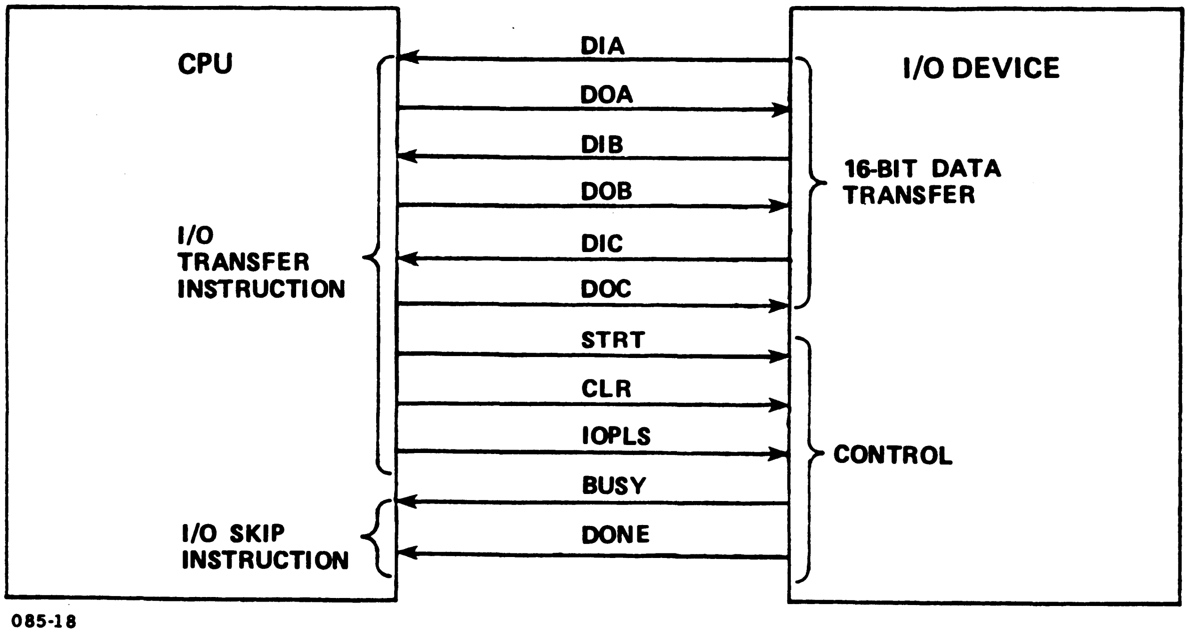


**Figure 5-9. CPU-I/O Device Input/Output Functions**

### 5.6.1.1 I/O Transfer and Device Control Instructions

I/O transfer instructions move data between the processor and the device interface. There are six possible device buffers, labeled A, B and C. Each label may refer to two separate buffers: an input (read) and an output (write) buffer.

The OPCODE field (bits 5-7) of the instruction specifies the type of transfer to take place (data in, data out, no transfer, etc.). Bits 3 and 4 specify the accumulator that supplies or receives the data, and bits 8 and 12 specify a control function (if any). The type of transfer is determined by the code in the OPCODE field as shown in Table 5-11.

TABLE 5-11. I/O TRANSFER-TYPE SPECIFICATION

| Bits 5-7 | OPCODE | Definition |
|---|---|---|
| 000 | NIO | No data transfer involved; device control only. |
| 001 | DIA | Move the contents of the A buffer in device D to the accumulator specified in the ac field, and send the control pulse specified by the control field to the selected device. |
| 010 | DOA | Move the contents of the accumulator specified in the ac field to the A buffer in device D, and send the control pulse specified by the control field to the selected device. The original contents of ac are unaffected. |
| 011 | DIB | Move the contents of the B buffer in device D to the accumulator specified in the ac field, and send the control pulse specified by the control field to the selected device. |
| 100 | DOB | Move the contents of the accumulator specified in the ac field to the B buffer in device D, and send the control pulse specified by the control field to device D. The original contents of ac are unaffected. |
| 101 | DIC | Move the contents of the C buffer in device D to the accumulator ac, and send the control pulse specified by the control field to device D. |
| 110 | DOC | Move the contents of the accumulator specified in the ac field to the C buffer in device D, and send the control pulse specified by the control field to device D. The original contents of ac are unaffected. |

The control field (bits 8 and 9) for I/O transfer instructions determines which control pulse should be transmitted, if any. The processor first performs the data transfer and then outputs the pulse. The control field is defined for regular I/O transfer instructions, as shown in Table 5-12.

**TABLE 5-12. CONTROL FIELD SPECIFICATION**

| OPCODE Bits 5-7 | Control Bits 8-9 | Control Mnemonic | Definition |
|---|---|---|---|
| 000-110 | 00 | None | None |
| 000-110 | 01 | S | Produce the STRT pulse. Typically, this starts the device by clearing its Done flag, setting its Busy flag, and clearing its interrupt request flag. |
| 000-110 | 10 | C | Produce the CLR pulse. Typically this clears both the Busy and Done flags, and the interrupt request flag, idling the device. |
| 000-110 | 11 | P | Pulse the special I/O bus control line (IOPLS). The effect, if any, depends upon the device. |

### 5.6.1.2 I/O Skip Instructions

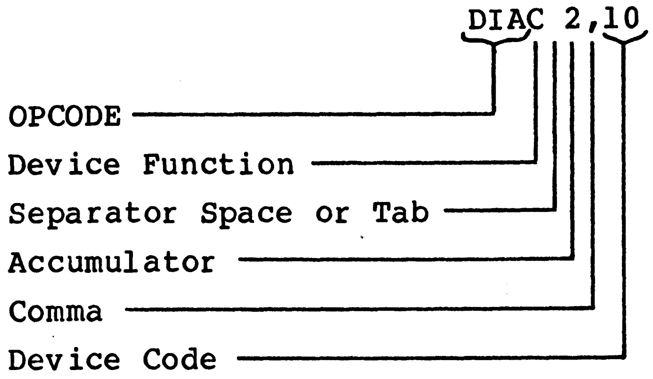
When the OPCODE field (bits 5-7) contains 111, the control field (bits 8 and 9) selects the flag to be tested in the conditional I/O skip. The control codes for skip operations are shown in Table 5-13.

TABLE 5-13. I/O SKIP INSTRUCTIONS

| OPCODE Bits 5-7 | Control Bits 8-9 | Control Mnemonic | Definition |
|---|---|---|---|
| 111 | 00 | SKPBN | Skip the next instruction if the Busy flag in the device is nonzero. |
| 111 | 01 | SKPBZ | Skip the next instruction if the Busy flag in the device is zero. |
| 111 | 10 | SKPDN | Skip the next instruction if the Done flag in the device is nonzero. |
| 111 | 11 | SKPDZ | Skip the next instruction if the Done flag in the device is zero. |

### 5.6.1.3 Assembler Language Conventions and Examples

An assembler language I/O transfer statement consists of the instruction mnemonic, an optional control function, an accumulator, and octal device code. Table 5-14 shows the programming conventions for regular input/output instructions. For example,



This instruction performs the function: Move the data from register A of device 10 into A2. Clear (reset) the device.

The device code may be represented by a device mnemonic. Thus,

```
    DIAC 2,TTI
```

is equivalent to the previous example because TTI represents device 10 (input buffer for a Teletype or CRT terminal).

A no I/O (NIO) or I/O skip instruction will not specify an accumulator since no data transfer occurs:

```
    NIOS TTI

    SKPBZ TTI
```

## TABLE 5-14. ASSEMBLER LANGUAGE CONVENTIONS FOR INPUT/OUTPUT INSTRUCTIONS

| Instruction Function | OPCODE Mnemonic | Optional Device Function | Separator Space or Tab | Ac | , | Device Code |
|---|---|---|---|---|---|---|
| No input/ output | NIO | | | None | None | 00-76 octal |
| Data in buffer A<br>Data out buffer A<br>Data in buffer B<br>Data out buffer B<br>Data in buffer C<br>Data out buffer C | DIA<br>DOA<br>DIB<br>DOB<br>DIC<br>DOC | S Start*<br>C Clear*<br>P General pulse* | | 0<br>1<br>2<br>3 | , | |
| Skip if Busy flag is nonzero<br>Skip if Busy flag is zero<br>Skip if Done flag is nonzero<br>Skip if Done flag is zero | SKPBN<br>SKPBZ<br>SKPDN<br>SKPDZ | | | | | |

*Optional

### 5.6.2 SPECIAL CODE 77 (CPU) INSTRUCTIONS

Certain system functions, setting and testing of processor flags, and interrupt processing control are accomplished via I/O instructions with the octal code 77 in bits 10-15. These instructions do not directly address a particular device and the device code mnemonic is CPU.

CPU instructions have the same general format as regular I/O instructions. The OPCODE field and control field, however, are interpreted differently.

- OPCODE field:
  - Channel A (DIA) is used to read the miniswitches at the front edge of the CPU board, or the DATA display on the optional Operator Control Unit. DOA is not defined.
  - Channel B addresses all I/O devices simultaneously for certain interrupt control functions.
  - Channel C does no data transfer. DIC and DOC are used for resetting all I/O devices and for halting the computer, respectively.
- Control pulses:
  - S and C are used to enable or disable interrupts.
  - P is used to set 32K or 64K-addressing mode, depending on the state of the LSB of A0. See Table 5-15.

The CPU has two flags which can be tested by the I/O skip instructions:

- Busy = ION set (interrupts are enabled)
- Done = power failure has been detected (will cause interrupt if ION set)

The assembler also recognizes several special mnemonics for CPU instructions.

Table 5-15 gives both the regular instruction mnemonic and the special mnemonic. It also provides a definition of the special function of the CPU instructions.

## TABLE 5-15. SPECIAL CPU INSTRUCTIONS

| Instruction | Special Mnemonic | Definition |
|---|---|---|
| NIO CPU | | No action. |
| NIOS CPU | INTEN | Sets the processor's Interrupt On (ION) flag. The processor will now respond to interrupt requests from devices, after execution of one more instruction. |
| NIOC CPU | INTDS | Clears the Interrupt On flag, so that the processor will not respond to interrupt requests. |
| NIOP CPU | None | Sets 32K or 64K-addressing mode, depending on the LSB value of A0. An LSB value of 0 sets 32K mode; a value of 1 sets 64K mode.<br>Enables Extended Instruction Set. Depends on the MSB value of A0. Enables Extended Instruction Set when MSB=1. Disables Extended Instruction Set when MSB=0. This applies to the MARK 5 with EIS and MARK 9 CPUs only. Refer to Sections 6.2 and 6.3 for further information. |
| DIA ac,CPU | READS ac | Reads the setting of the miniswitches at the front edge of the CPU board (or the value in the DATA readout of the optional Operator Control Unit, if installed) into accumulator ac. |
| DIB ac,CPU | INTA ac | Reads the device code of the highest priority device requesting an interrupt into accumulator ac. |
| DOB ac,CPU | MSKO ac | Sets up interrupt disable flags in all devices simultaneously, according to the mask code in accumulator ac. Each device is associated with one of the bits in the accumulator, and its flag is set (mask bit=1) disabling interrupt from that device, or cleared (mask bit=0) enabling interrupts from it. A MSKO with ac=177777 disables interrupts from all devices. |

TABLE 5-15. SPECIAL CPU INSTRUCTIONS (Cont)

| Instruction | Special Mnemonic | Definition |
|---|---|---|
| DICC ac,CPU | IORST | Generate I/O reset to clear the Busy, Done, and Interrupt Disable flags in all devices. This instruction also clears the processor's ION flag and sets it to 32K-addressing mode. (Does not change contents of selected accumulator.) |
| DOC ac,CPU | HALT | Halt the processor. Requires manual action to restart processor. |

Note that the special mnemonic does not allow the programmer to specify the S and C functions. For example,

```
    READS 3
```

when executed, deposits the value entered via the Operator Control Unit or miniswitches into A3. If the programmer wishes to also set ION, the assembler instruction mnemonic would have to be used:

```
    DIAS 3,CPU
```

This instruction sets ION after reading the DATA display on the Operator Control Unit, or the miniswitches.

The instruction IORST, however, assumes the C function. All I/O device flags are reset and the ION flag is cleared. In order to reset the I/O devices without clearing the ION flag, the regular assembler instruction must be used:

```
    DIC 0,CPU
```

As with regular I/O instructions, a value of 111 in bits 5-7 signifies a conditional skip instruction. The function field in this case indicates which processor flag (Interrupt On or Power Fail) will be tested, as shown in Table 5-16.

TABLE 5-16. CONDITIONAL SKIP INSTRUCTIONS

| Bits 8&9 | Instruction | Definition |
|---|---|---|
| 00 | SKPBN CPU | Skip next instruction if ION is nonzero. |
| 01 | SKPBZ CPU | Skip next instruction if ION is zero. |
| 10 | SKPDN CPU | Skip next instruction if the Power Failure flag is nonzero. |
| 11 | SKPDZ CPU | Skip next instruction if the Power Failure flag is zero. |

### 5.6.2.1 Assembler Language Conventions

CPU instructions are usually written using the special mnemonics shown in Section 5.6.2; however, they may also be written in the same manner as regular I/O instructions, specifying the instruction mnemonic, optional control function, optional accumulator, and a device code of 77 octal (mnemonic CPU). For example,

```
    NIOS CPU
```

sets the ION flag in the processor.

## 5.7 INSTRUCTION EXECUTION TIMES

One of the outstanding features of the POINT 4 MARK 5 or MARK 9 Computer is the substantial reduction in instruction time over execution time in comparable minicomputers. Table 5-17 gives instruction execution times for the POINT 4 MARK 5 Computer. Table 5-18 lists instruction execution times for the POINT 4 MARK 5 Computer with EIS option. Table 5-19 lists instruction execution times for the POINT 4 MARK 9 Computer.

These times are exclusive of three types of overhead:

1. A 500-nanosecond (200-nanosecond for the MARK 9) refresh cycle takes place once every 16 microseconds - this adds about 3% overhead.

2. Arithmetic/logic instructions on RAM page boundaries (two least significant digits of address = 76 or 77) take an extra 100 nanoseconds - this results in approximately 0.25% overhead. This does not apply to the MARK 9.

3. If the optional Operator Control Unit is connected to the CPU, a 1.8-microsecond address display cycle occurs once every 600 microseconds, resulting in 0.3% overhead.

## TABLE 5-17. INSTRUCTION EXECUTION TIMES FOR MARK 5

| Instruction Category | Instruction (generic type) | Execution Time (ns) |
|---|---|---|
| MEMORY REFERENCE | Load or store accumulator (LDA,STA) | 800 |
| | Increment or decrement if zero (ISZ,DSZ) | 1100 |
| | Jump (JMP) | 400 |
| | Jump to subroutine (JSR) | 400 |
| | Additional times for various addressing modes: | |
| | - Each level of indirect addressing | 400 |
| | - Auto-indexing | 200 |
| ARITHMETIC/ LOGIC | Arithmetic/logic instructions (COM,NEG,MOV,INC,ADC,SUB,ADD,AND) | 400 |
| | - for skip (SKP) add | 0 |
| INPUT/ OUTPUT | Input | 900 |
| | - for START, CLEAR, or PULSE add | 600 |
| | Output | 1200 |
| | - for START, CLEAR, or PULSE add | 400 |
| | No I/O transfer (NIO) | 1200 |
| | - for START, CLEAR, or PULSE add | 400 |
| | I/O skips (SKPBN,SKPBZ,SKPDN,SKPDZ) | 900 |
| | Interrupt acknowledge (INTA) | 1000 |
| | Interrupt response | 1200 |
| DATA CHANNEL TRANSFERS | Standard data channel transfers: | |
| | - Input | 1100 |
| | - Output | 1700 |
| | High-speed data channel transfers: | |
| | - Input | 900 |
| | - Output | 1300 |

## TABLE 5-18. INSTRUCTION EXECUTION TIMES FOR MARK 5 WITH EIS

| Instruction Category | Instruction (generic type) | Execution Time (ns) |
|---|---|---|
| MEMORY REFERENCE | Load or store accumulator (LDA,STA) | 800 |
| | Increment or decrement if zero (ISZ,DSZ) | 1100 |
| | Jump (JMP) | 400 |
| | Jump to subroutine (JSR) | 400 |
| | Additional times for various addressing modes: | |
| | - Each level of indirect addressing | 400 |
| | - Auto-indexing | 200 |
| ARITHMETIC/ LOGIC | Arithmetic/logic instructions (COM,NEG,MOV,INC,ADC,SUB,ADD,AND) | 400 |
| | - for skip (SKP) add | 0 |
| INPUT/ OUTPUT | Input | 900 |
| | - for START, CLEAR, or PULSE add | 600 |
| | Output | 1200 |
| | - for START, CLEAR, or PULSE add | 400 |
| | No I/O transfer (NIO) | 1200 |
| | - for START, CLEAR, or PULSE add | 400 |
| | I/O skips (SKPBN,SKPBZ,SKPDN,SKPDZ) | 900 |
| | Interrupt acknowledge (INTA) | 1000 |
| | Interrupt response | 1200 |
| DATA CHANNEL TRANSFERS | Standard data channel transfers: | |
| | - Input | 1100 |
| | - Output | 1700 |
| | High-speed data channel transfers: | |
| | - Input | 900 |
| | - Output | 1300 |

### TABLE 5-19. INSTRUCTION EXECUTION TIMES FOR MARK 9

| Instruction Category | Instruction (generic type) | Execution Time (ns) |
|---|---|---|
| MEMORY REFERENCE | Load accumulator (LDA) | 600 |
| | Store accumulator (STA) | 625 |
| | Increment if zero (ISZ) | 825 |
| | Decrement if zero (DSZ) | 900 |
| | Jump (JMP) or Jump to subroutine (JSR): | |
| | - One level | 425 |
| | - Two levels | 925 |
| | - Three levels | 1425 |
| | Additional times for various addressing modes: | |
| | - Indirect addressing | |
| | - Auto-indexing | 275 |
| ARITHMETIC/ LOGIC | Arithmetic/logic instructions (COM,NEG,MOV,INC,ADC,SUB,ADD,AND) | 300 |
| | - for skip (SKP) add | 375 |
| INPUT/ OUTPUT | Input | 1125 |
| | - for START, CLEAR, or PULSE add | 300 |
| | Output | 1500 |
| | - for START, CLEAR, or PULSE add | 300 |
| | No I/O Transfer (NIO) | 1500 |
| | - for START, CLEAR, or PULSE add | 300 |
| | I/O skips (SKPBN,SKPBZ,SKPDN,SKPDZ) | 1000 |
| | Interrupt acknowledge (INTA) | 1675 |
| | Interrupt response | 1175 |
| DATA CHANNEL TRANSFERS | Standard data channel transfers: | |
| | - Input | 1600 |
| | - Output | 2000 |
| | High-speed data channel transfers: | |
| | - Input | 900 |
| | - Output | 1200 |

# Section 6
# EXTENDED INSTRUCTION SET

## 6.1 INTRODUCTION

This section describes the Extended Instruction Set, which is standard on the MARK 9 and is available as an option on the MARK 5 Computer. The POINT 4 Extended Instruction Set is upward compatible with the Standard Instruction Set. Any software that runs on the Standard Instruction Set will run on the Extended Instruction Set with the exception of software that uses an arithmetic no-op skip instruction.

The following subsections describe the operation of each instruction. Recommended mnemonics and syntax for the instructions are also provided.* See Appendices F and G for octal values and execution times of the extended instructions.

*The recommended mnemonics and syntax are not supported by the current assembler supplied by POINT 4. However, the SYMBOLS file may be extended to include definitions that will be helpful in their use.

## 6.2 ENABLING/DISABLING EXTENDED INSTRUCTION SET

### 6.2.1 ENABLING EIS

The MARK 9 requires no hardware changes to enable the use of the Extended Instruction Set. To enable the software, issue a NIOP command to device code 77 (octal) with the MSB of accumulator zero set to 1.

The MARK 5 requires the following hardware and software changes to enable the use of the Extended Instruction Set:

- CPU board jumper W7, located at 17c on the CPU/memory board, should be in position 3. This allows the most significant bit (MSB) to be used as an enabling factor for the Extended Instruction Set.
- CPU board jumper W3, located at 4f on the CPU/memory board, should be in position 2. This enables the NIOP instruction to be used as an enabling factor for the Extended Instruction Set.
- A NIOP command to device code 77 (octal) with the MSB of accumulator zero set to 1 enables the Extended Instruction Set.

### 6.2.2 DISABLING EIS

A NIOP command to device code 77 (octal) with the MSB of accumulator zero set to 0 disables the Extended Instruction Set. An I/O reset (IORST) command also disables the Extended Instruction Set (see Table 4-2).

## 6.3 EXTENDED INSTRUCTION IMPLEMENTATION

The extended instructions are implemented on a MARK 5/9 using the NOP skip command, which is formatted as shown in Figure 6-1.

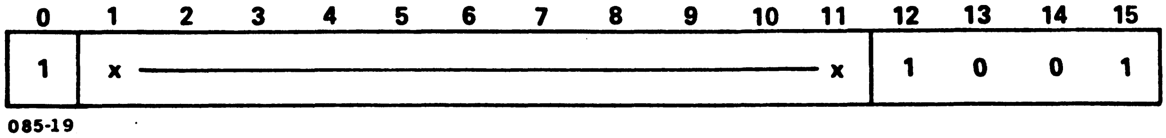


**Figure 6-1. NOP Skip Command Format**

The values in bits 1 through 11 have no effect on the Standard Instruction Set. In the Standard Instruction Set, each instruction having a 1 in the most significant bit and 1001 (binary) in the four least significant bits is a no-op instruction, but causes the next location in sequence to be skipped.

Certain instructions require or generate a 32-bit value. In such cases, the accumulators are specified in the form "AH,AL" where accumulator AH contains the high-order 16 bits and AL contains the low-order 16 bits of the value. For example, "A1,A2" indicates that accumulator A1 contains the high-order half of the value and A2 contains the low-order half of the value.

This Extended Instruction Set includes two-word (32-bit) instructions. In such cases, the first word is formatted as in the diagram above, but the second word is an unrestricted 16-bit operand. These two-word instructions cannot be used following any kind of skip instruction, because a skip instruction only skips over one word and would then attempt to execute the second word (16-bit operand) as an instruction.

### 6.3.1 PAGE ZERO LOCATIONS

Certain page zero locations are used by the POINT 4 MARK 5 with EIS and MARK 9 as follows:

40 Base address for XGETBYTE, XSGETBYTE, XLOAD and XTRANSLATE (see Sections 6.3.7 and 6.3.9)

41 Base address for XPUTBYTE, XSPUTBYTE, and XSTORE (see Sections 6.3.7 and 6.3.9)

45 Stack base pointer (see Section 6.3.5)

46 Stack frame pointer (see Section 6.3.5); Trap instruction (see Section 7.6)

47 Stack limit pointer (see Section 6.3.5); Trap instruction (see Section 7.6)

The contents of these cells are not affected by and do not affect the operation of the computer except as described in the referenced portions of this section.

### 6.3.2 IMMEDIATE COMPARE INSTRUCTIONS

These instructions each contain an 8-bit, unsigned, literal data field within the instruction itself. This field is referred to as the immediate constant in the following descriptions. All these instructions compare the entire contents of accumulator A0 (a 16-bit unsigned integer) with the immediate constant, and cause the next instruction in sequence to be skipped if the conditions are met. Note that the next location must contain a one-word instruction. Table 6-1 lists each instruction and its function. Figure 6-2 shows the binary format for immediate compare instructions. For a complete list of binary formats for all extended instructions and definitions of format terms, see Appendix H.

**TABLE 6-1. IMMEDIATE COMPARE INSTRUCTIONS**

| Instruction | Definition |
|---|---|
| SLEI constant | Skip if less than or equal to immediate constant. Skips if the contents of A0 are less than or equal to the value of the immediate constant. |
| SGRI constant | Skip if greater than immediate constant. Skips if the contents of A0 are greater than the immediate constant. |
| SEQI constant | Skip if equal to immediate constant. Skips if the contents of A0 are equal to the immediate constant. |
| SNEI constant | Skip if not equal to immediate constant. Skips if the contents of A0 are not equal to the immediate constant. |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLEI | 1 | 0 | IMMEDIATE CONSTANT | | | | | | | | 0 | 0 | 1 | 0 | 0 | 1 |
| SGRI | 1 | 0 | IMMEDIATE CONSTANT | | | | | | | | 0 | 1 | 1 | 0 | 0 | 1 |
| SEQI | 1 | 0 | IMMEDIATE CONSTANT | | | | | | | | 1 | 0 | 1 | 0 | 0 | 1 |
| SNEI | 1 | 0 | IMMEDIATE CONSTANT | | | | | | | | 1 | 1 | 1 | 0 | 0 | 1 |

085-31

**Figure 6-2. Binary Format for Immediate Compare Instructions**

### 6.3.2.1 Other Immediate Operations

As with the immediate compare instructions, these instructions have a literal constant within the instruction itself. However, these instructions allow loading, adding, or subtracting any 8-bit unsigned constant to or from accumulator A0. No other accumulator is affected.

Table 6-2 defines other immediate operations. Figure 6-3 shows the binary format for these instructions.

**TABLE 6-2. OTHER IMMEDIATE OPERATIONS**

| Instruction | Definition |
|---|---|
| LDI constant | Load immediate constant. Loads the literal constant into accumulator A0. The eight most significant bits of A0 are zeroed. Carry is not affected. |
| ADI constant | Add immediate constant. The literal constant is added to the contents of A0. Carry is then propagated as in an ADD instruction, and carry is toggled if overflow occurs. |
| SBI constant | Subtract immediate constant. The literal constant is subtracted from the contents of A0. Borrow is propagated as with a SUB instruction, and the carry is toggled if borrow does not occur (if the result is not negative). This is equivalent to adding the 16-bit two's complement of the literal constant to A0 with carry propagated. |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LDI | 1 | 1 | IMMEDIATE CONSTANT | | | | | | | | 0 | 0 | 1 | 0 | 0 | 1 |
| ADI | 1 | 1 | IMMEDIATE CONSTANT | | | | | | | | 0 | 1 | 1 | 0 | 0 | 1 |
| SBI | 1 | 1 | IMMEDIATE CONSTANT | | | | | | | | 1 | 0 | 1 | 0 | 0 | 1 |

085-32

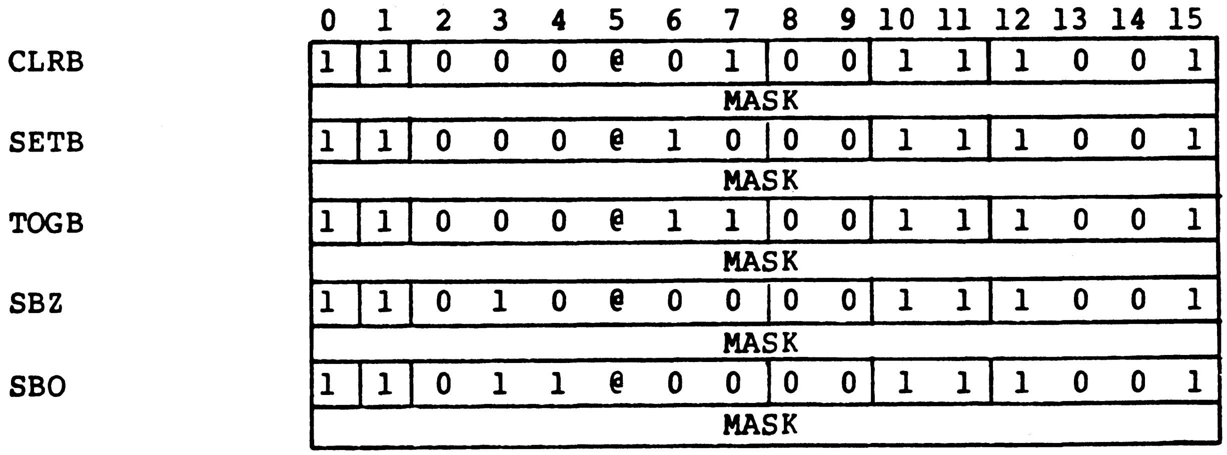
**Figure 6-3. Binary Format for Other Immediate Operations**

### 6.3.3 BIT MANIPULATION AND TEST INSTRUCTIONS

These instructions allow setting, clearing, toggling, and testing one or more selected bits within the operand. The operand is normally the contents of accumulator A0. However, if an @ symbol is appended to the instruction mnemonic, the contents of location (A1)+(A2) becomes the operand and the result is left in A0 as well as in the addressed location. These are two-word instructions. The second word in each instruction is the mask which selects the bits to be acted upon. These instructions must not be preceded by any form of a skip instruction, and must not be followed by any two-word instruction. No accumulators other than A0 are changed in any case. Table 6-3 defines the bit manipulation and test instructions. Figure 6-4 shows the binary format for these instructions.

**TABLE 6-3. BIT MANIPULATION AND TEST INSTRUCTIONS**

| Instruction | Definition |
|---|---|
| CLRB mask | Each "1" bit in the mask causes the corresponding bit in the operand to be cleared to "0". |
| SETB mask | Each "1" bit in the mask causes the corresponding bit in the operand to be set to "1". |
| TOGB mask | Each "1" bit in the mask causes the corresponding bit in the operand to be toggled. An exclusive OR is performed between the operand and the mask. |
| SBZ mask | Skip the next location in sequence if all "1" bits in the mask correspond to zero bits in the operand, or if there are no "1" bits in the mask. The operand is not changed. |
| SBO mask | Skip the next location in sequence if any "1" bit in the mask corresponds to a "1" bit in the operand. The operand is not changed. |



|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLRB | 1 | 1 | 0 | 0 | 0 | @ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SETB | 1 | 1 | 0 | 0 | 0 | @ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TOGB | 1 | 1 | 0 | 0 | 0 | @ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBZ | 1 | 1 | 0 | 1 | 0 | @ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBO | 1 | 1 | 0 | 1 | 1 | @ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

085-33

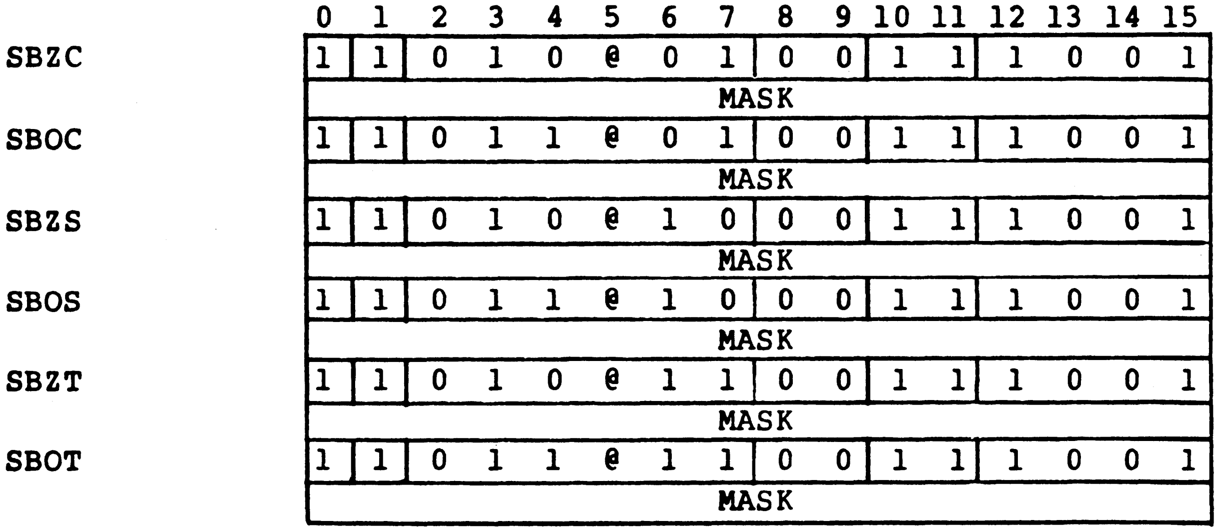
**Figure 6-4. Binary Format for Bit Manipulation and Test Instructions**

### 6.3.4 BIT TEST INSTRUCTION COMBINATIONS

The bit test (skip) instruction may also be combined with the clear, set, or toggle instructions. Table 6-4 shows the possible combinations. Figure 6-5 shows the binary format for these instructions.

**TABLE 6-4. BIT TEST INSTRUCTION COMBINATIONS**

| Instruction | Definition |
|---|---|
| SBZC mask | Performs an SBZ, followed by a CLRB |
| SBOC mask | Performs an SBO, followed by a CLRB |
| SBZS mask | Performs an SBZ, followed by a SETB |
| SBOS mask | Performs an SBO, followed by a SETB |
| SBZT mask | Performs an SBZ, followed by a TOGB |
| SBOT mask | Performs an SBO, followed by a TOGB |



|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SBZC | 1 | 1 | 0 | 1 | 0 | @ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBOC | 1 | 1 | 0 | 1 | 1 | @ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBZS | 1 | 1 | 0 | 1 | 0 | @ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBOS | 1 | 1 | 0 | 1 | 1 | @ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBZT | 1 | 1 | 0 | 1 | 0 | @ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SBOT | 1 | 1 | 0 | 1 | 1 | @ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
|  | MASK |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

085-34

**Figure 6-5. Binary Format for Bit Test Instruction Combinations**

### 6.3.5 POINT 4 EIS STACK INSTRUCTIONS

The POINT 4 EIS stack instructions are supported by the MARK 5 with the EIS option and by the MARK 9. For MARK 9-exclusive stack instructions, see Section 7.3.

The POINT 4 EIS stack instructions utilize the stack pointer (SP) and three reserved page zero cells. Their location, tags, and names are shown below:

| Loc | Tag | Name |
|---|---|---|
| 45 | SB | Stack base pointer |
| 46 | FP | Frame pointer; used by trap instruction in MARK 9 (refer to Section 7.6) |
| 47 | SL | Stack limit pointer; used by trap instruction in MARK 9 (refer to Section 7.6) |

Another special cell, the stack fault vector (SF), must be set by the software at (SB)+1. Due to the sequence in which the stack overflow test is performed, the cell at location (SL)-1 must be reserved. The stack organization for new registers is shown in Figure 6-6.

Table 6-5 defines the instructions used for new registers. Figure 6-7 shows the binary format for these instructions.

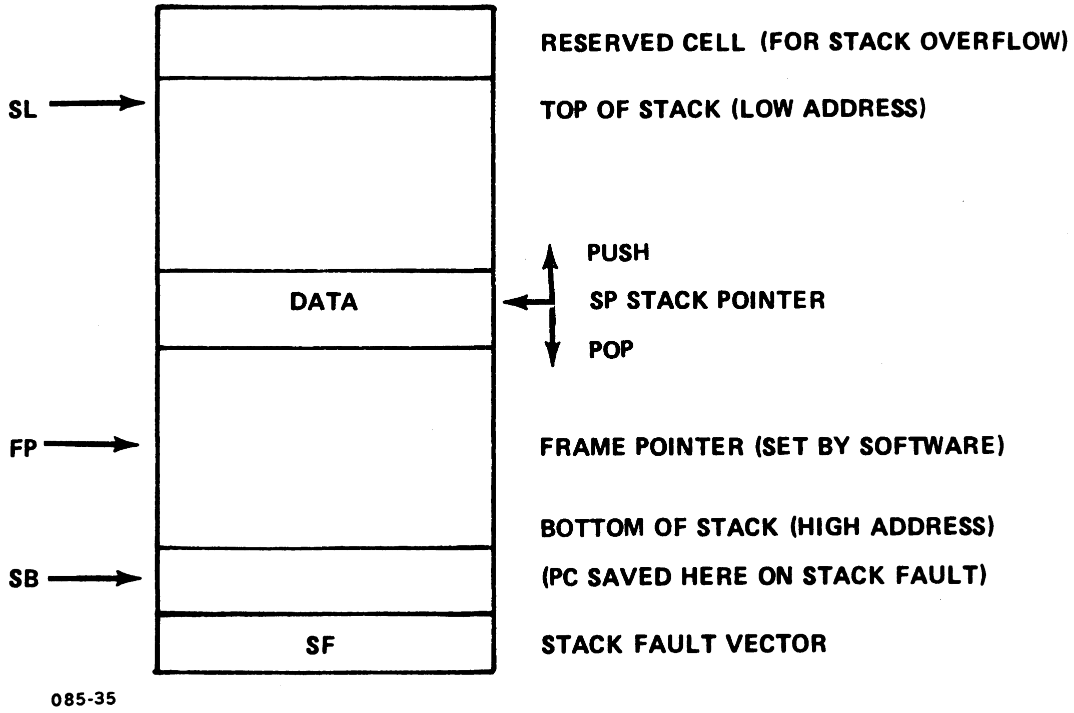

RESERVED CELL (FOR STACK OVERFLOW)
SL
TOP OF STACK (LOW ADDRESS)
PUSH
DATA
SP STACK POINTER
POP
FP
FRAME POINTER (SET BY SOFTWARE)
BOTTOM OF STACK (HIGH ADDRESS)
SB
(PC SAVED HERE ON STACK FAULT)
SF
STACK FAULT VECTOR
085-35

**Figure 6-6. Stack Organization for New Registers**

TABLE 6-5. INSTRUCTIONS USED FOR NEW REGISTERS

| Instruction | Definition |
|---|---|
| SETSP ac | Set stack pointer. The contents of accumulator ac (where ac=0, 1, 2, or 3) are loaded into the SP register. No other register is affected. |
| REDSP ac | Read stack pointer. The current contents of the SP are copied into accumulator ac (where ac=0, 1, 2, or 3). No other register, including the SP, is affected. |
| PUSH ac | Push data onto stack. The SP is decremented, and the contents of accumulator ac are stored at SP. If (SP)>(SL), stack overflow has occurred, in which case (PC)+1 is stored at SB, and PC is loaded with SF at (SB)+1. Specifically:<br>SP <= (SP)-1, (SP) <= (ac)<br>If (SP)<(SL) then (SB)<=(PC)+1, PC<=(SF) |
| POP ac | Pop data from stack. The contents of the location pointed to by the SP are loaded into accumulator ac (where ac=0, 1, 2, or 3), and the SP is incremented. If (SP)>(FP), then stack underflow has occurred, in which case (PC)+1 is stored at SB, and PC is loaded with SF to do a fault branch. Specifically:<br>ac <= ((SP)), SP <= (SP)+1<br>If (SP)>(FP) then (SB)<=(PC)+1, PC<=(SF) |
| PUSHJUMP<br>Address | Push and jump. This two-word instruction pushes a return address onto the stack and jumps to the absolute address given in the second word of the instruction. Specifically:<br>SP <= (SP)-1, (SP) <= (PC)+2,<br>PC <= ((PC)+1)<br>If (SP)<(SL) then (SB)<=(PC)+1, PC<=(SF) |

**TABLE 6-5. INSTRUCTIONS USED FOR NEW REGISTERS (Cont)**

| Instruction | Definition |
|---|---|
| | **NOTE**<br>The relative address, as well as the displacement in the following two instructions, is added to the contents of the specified register and any overflow is ignored. Therefore, the relative address or displacement may be considered by the programmer to be either an unsigned or a signed integer. |
| RPUSHJUMP Displacement | Relative push and jump - this is similar to PUSHJUMP except that it jumps to (PC)+displacement. |
| I2PUSHJUMP Displacement | Indexed (A2) push and jump - this is similar to PUSHJUMP except that it jumps to location (A2)+displacement. |
| I3PUSHJUMP Displacement | Indexed (A3) push and jump - this is similar to PUSHJUMP except that it jumps to location (A3)+displacement. |
| POPJUMP | Pop and jump - this is the normal return from a subroutine called by a PUSHJUMP. The stack is popped into PC. Specifically:<br>SP <= (SP)+1<br>If (SP)>(FP) then (SB)<=(PC)+1, PC<=(SF)<br>else PC <= ((SP)-1) |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SETSP | 1 | 1 | 0 | ACCU | | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| REDSP | 1 | 1 | 0 | ACCU | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| PUSH | 1 | 1 | 1 | ACCU | | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| POP | 1 | 1 | 1 | ACCU | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| PUSHJ | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| RPUSH | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| I2PUS | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| I3PUS | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| POPJ | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

085-36

**Figure 6-7. Binary format for New Register Instructions**

### 6.3.6 NYBBLE MANIPULATION INSTRUCTIONS

A nybble is a four-bit data group (half a byte), and each 16-bit word holds four nybbles. When data is manipulated in binary coded decimal (BCD), each BCD digit occupies one nybble. These instructions facilitate easy handling of BCD data. Table 6-6 defines the nybble manipulation instructions. Figure 6-8 shows the binary format for these instructions.

TABLE 6-6. NYBBLE MANIPULATION INSTRUCTIONS

| Instruction | Definition |
|---|---|
| NYBSL | Nybble shift left. A2 indicates the operand's memory location, and A0 indicates the size of the operand (number of words-1). The word at A2 contains the most significant four nybbles. The (A0)+1 words starting at (A2)+(A0) and working back through A2 are shifted left one nybble (four bits), and the low-order nybble of A1 is shifted in. The old high-order nybble of the word at A2 is returned in the low-order nybble of A1. The new high-order nybble of the word at A2 is returned in the high-order nybble of A0. The rest of A1 and A0 is zeroed. Accumulators A2 and A3 are unchanged. Carry is not preserved. |
| NYBSR | Nybble shift right. The (A0)+1 words starting at A2 are shifted right one nybble (four bits), and the low-order nybble of A1 is shifted into the top of the word at A2. If the nybble shifted out of the last word is five or greater, the carry is set; otherwise the carry is cleared. Accumulator A0 is -1; A1 is destroyed; A2 points to the last word+1 of the operand and A3 is unchanged. |
| DECADD | Decimal add. The BCD numbers in accumulators A0 and A1 are added with the carry, and the result is stored in accumulator A0 with the overflow stored in carry. Accumulators A1, A2, and A3 are unchanged. Specifically:<br>(A0) <= (A0)+(A1)+(C)<br>(C) <= carry out |

TABLE 6-6. NYBBLE MANIPULATION INSTRUCTIONS (Cont)

| Instruction | Definition |
|---|---|
| DECSUB | Decimal subtract. The BCD number in accumulator A1 and the complement of the carry are subtracted from the BCD number in accumulator A0. The result is stored in accumulator A0. This sets the carry if there is no borrow. Accumulators A1, A2, and A3 are unchanged. Specifically:<br>A0 <= (A0)-(A1)-(C-)<br>C <= borrow out- |
| CONVERT | Convert BCD to binary. This instruction is designed for use as a step in a conversion process from a BCD number to a binary number, using A1 and A2 as a two-word binary accumulator for the conversion product. The 32-bit unsigned binary number in A1, A2 is multiplied by ten (decimal), and the value in A0 is added to the result. Carry is set if overflow occurs, otherwise carry is cleared. Accumulators A0 and A3 are unchanged.<br>A1,A2 <= (A1,A2)*10+(A0) |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NYBSL | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| NYBSR | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| DECAD | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| DECSU | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| CONVE | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

085-37

**Figure 6-8. Binary Format for Nybble Manipulation Instructions**

### 6.3.7 EXTENDED LOAD AND STORE INSTRUCTIONS

These instructions load or store accumulator A0 to/from the address in A2 relative to the base address in location 40 or 41 octal. Table 6-7 defines the extended load and store instructions. Figure 6-9 shows the binary format for these instructions.

**TABLE 6-7. EXTENDED LOAD AND STORE INSTRUCTIONS**

| Instruction | Definition |
|---|---|
| XLOAD | Extended load. Loads A0 with the contents of location (40)+(A2). Only accumulator A0 is changed. |
| XSTORE | Extended store. Stores the contents of A0 at location (41)+(A2). No accumulators are changed. |

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

085-38

**Figure 6-9. Binary Format for Extended Load and Store Instructions**

### 6.3.8 VECTORED JUMP TO SUBROUTINE INSTRUCTION

A vectored jump-to-subroutine instruction (VJSR subnum) performs a JSR to a subroutine via a vector table that immediately precedes the interrupt service routine. It is assumed that memory location 1 contains a pointer to the interrupt service routine. Specifically:

(A3) <= (PC)+1, (PC) <= ((1)-subnum-1)

where the subnum is in the range of 0 to 37 octal. Figure 6-10 shows the binary format for this instruction.

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

085-39

**Figure 6-10. Binary Format for Vectored Jump to Subroutine Instruction**

### 6.3.9 BYTE ACCESS INSTRUCTIONS

All these instructions operate only on accumulator A0 using a byte address in accumulator A1, as described for the particular instruction.

The only exceptions to this are SGETBYTE and XSGETBYTE, which use a byte address in accumulator A2. When a byte address is used, it is shifted one bit to the right to form a word address. The bit shifted out selects the byte of the word (indicated by the address). Byte zero is the high-order byte of each word.

All get byte instructions load the addressed byte into the low-order half of A0, and the high-order half of A0 is zeroed. The addressed word is unchanged.

All put byte instructions store the byte from the low-order half of A0 and ignore the high-order half. A0 is unchanged, and the other byte in the addressed word is unchanged. Table 6-8 defines the byte access instructions. Figure 6-11 shows the binary format for these instructions.

### TABLE 6-8. BYTE ACCESS INSTRUCTIONS

| Instruction | Definition |
|---|---|
| GETBYTE | Gets the byte from byte address (A1). |
| PUTBYTE | Stores the byte at byte address (A1). |
| SGETBYTE | Sequential get byte - gets the byte from byte address (A2), and A2 is then incremented. |
| SPUTBYTE | Sequential put byte - stores the byte at byte address (A1), and A1 is then incremented. |
| XGETBYTE | Extended get byte - gets the byte from byte address (A1) relative to the base word address in location 40 octal. |
| XPUTBYTE | Extended put byte - stores the byte at byte address (A1) relative to the base word address in location 41 octal. |
| XSGETBYTE | Extended sequential get byte - similar to SGETBYTE except relative to the base word address in location 40 octal. |
| XSPUTBYTE | Extended sequential put byte - similar to SPUTBYTE except relative to the base word address in location 41 octal. |
| IGETBYTE d<br>(displacement) | Indirect get byte - the contents of location (A2)+displacement are used as the byte address, where displacement is in the range 0 to 17 octal. If this byte address is greater than or equal to (A1), the instruction is a no-op and does not skip. If the byte address is less than (A1), it is incremented and the resulting byte address is stored at (A2)+displacement. The byte from that byte address is loaded into the low-order half of A0. The high-order half of A0 is zeroed, and a skip occurs. Only accumulator A0 is changed. |

**TABLE 6-8. BYTE ACCESS INSTRUCTIONS (Cont)**

| Instruction | Definition |
|---|---|
| IPUTBYTE d<br>(displacement) | Indirect put byte - the contents of location (A2)+displacement are used as the byte address, where displacement is in the range 0 to 17 octal. If this byte address is greater than or equal to (A1), the instruction is a no-op and does not skip. If the byte address is less than (A1), it is incremented and the resulting byte address is stored at (A2)+displacement. The byte in the low-order half of A0 is stored at the resulting byte address, and a skip occurs. All accumulators are unchanged. |
| XTRANSLATE | Extended sequential translate byte - a translation table, which consists of up to 256 single-word translation values, immediately follows this instruction. The instruction loads a byte into accumulator A0 from the byte address (A1) relative to the base word address at location 40 octal. A1 is incremented. The contents of the word at word address (PC)+(A0) are compared with octal 377. If the value of the loaded word is greater than octal 377, the PC jumps to the word address. If the value is less than or equal to octal 377, it is loaded into accumulator A0 and a jump is performed using (A3) as the word address. |

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GETBY | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| PUTBY | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SGETB | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SPUTB | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XGETB | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XPUTB | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XSGET | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XSPUT | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| IGETB | 1 | 1 | 1 | 0 | D | D | D | D | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| IPUTB | 1 | 1 | 1 | 1 | D | D | D | D | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| XTRAN | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

085-40

**Figure 6-11. Binary Format for Byte Access Instructions**

## 6.3.10 BINARY MULTIPLY AND DIVIDE INSTRUCTIONS

Table 6-9 defines the binary multiply and divide instructions for the Extended Instruction Set. Figure 6-12 shows the binary format for these instructions.

TABLE 6-9. BINARY MULTIPLY AND DIVIDE INSTRUCTIONS

| Instruction | Definition |
|---|---|
| ABINMULTIPLY | Accumulative binary multiply - the unsigned 16-bit binary integers in A0 and A2 are multiplied to form a 32-bit binary product. The 16-bit value in A1 is added to the product, and the result is returned in A1,A2. Accumulators A0 and A3 are unchanged. Carry is not preserved.<br>A0*A2+A1 --> A1,A2 |
| BINDIVIDE | Binary divide - the 32-bit unsigned dividend in A1,A2 is divided by the 16-bit unsigned divisor in A0. The quotient is returned in A2, and the remainder is returned in A1. Accumulators A0 and A3 are unchanged. Carry is set to one if overflow occurs (i.e., if A0 is not greater than A1). If overflow does not occur, carry is zeroed.<br>A1,A2/A0 --> A2 R A1 |

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

085-41

Figure 6-12. Binary Format for Binary Multiply and Divide Instructions

### 6.3.11 MOVE AND SEARCH INSTRUCTIONS

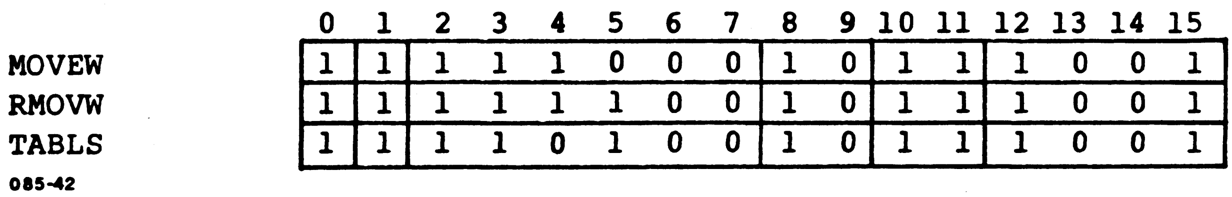
The move and search instructions operate on a block of words in memory. Table 6-10 defines the move and search Instructions. Figure 6-13 shows the binary format for these instructions.

TABLE 6-10. MOVE AND SEARCH INSTRUCTIONS

| Instruction | Definition |
|---|---|
| *MOVEWORDS* | Move block of words. Moves a specified number of words from one contiguous area in memory to another specified area. Before executing the MOVEWORDS instruction, the accumulators must be set up as follows:<br>A0 = one less than the number of words to be moved<br>A1 = address of beginning of source area<br>A2 = address of beginning of destination area<br>The word at (A1)+(A0) is stored at (A2)+(A0), and A0 is decremented. The process continues if (A0) is not = -1. After completion of the cycle, the carry is zero and (A0) = -1. All other registers are unchanged. Note that the last word is moved first. |
| RMOVWORDS | Reverse move words. Should be used instead of MOVEWORDS if the destination area ends within the source area. Because MOVEWORDS moves from the end of the source area to the beginning, the source would be overlayed. Before executing the RMOVWORDS instruction, the accumulators must be set up as follows:<br>A0 = one less than the number of words to be moved<br>A1 = address of end of source area<br>A2 = address of end of destination area<br>The word stored at (A1)-(A0) is stored at (A2)-(A0), and A0 is decremented. The process repeats if (A0) is not = -1. After completion of the instruction, (A0) = -1, and the carry is zero. All other accumulators are unchanged. |

**TABLE 6-10. MOVE AND SEARCH INSTRUCTIONS (Cont)**

| Instruction | Definition |
|---|---|
| TABLSEARCH | Performs a search of a specified area of memory for a given value. Before executing this instruction, the accumulators must be set up as follows:<br>A0 = value to be searched for<br>A1 = increment value<br>A2 = address of beginning of area to search<br>A3 = address one greater than end of area to search<br>The contents of location (A2) are compared with the value in A0. If equal, execution is complete and a skip is executed. If not equal, (A1) is added to (A2) and the instruction is repeated. If (A2) becomes greater than or equal to (A3) without finding a match, execution is complete and no skip occurs. |



|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MOVEW | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| RMOVW | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| TABLS | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

085-42

**Figure 6-13. Binary Format for Move and Search Instructions**

### 6.3.12 TRACE INSTRUCTION

The MARK 9 CPU does not support this instruction.

TRACE is a two-word instruction that allows the remote execution of an instruction while operating in a host program. The second word of the instruction contains the address of the remote instruction to be executed. When an instruction is traced, the following sequence occurs using a new hardware register, XPC.

```
    XPC <= (PC)+1
    PC <= ((XPC))
```

Execute the instruction at (PC).

```
    (XPC) <= (PC)
    PC <= (XPC)+1
```

Resume execution at PC.

Figure 6-14 shows the binary format for the trace instruction.

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

085-43

**Figure 6-14. Binary Format for Trace Instruction**

# Section 7
# MARK 9 EXCLUSIVE INSTRUCTIONS

## 7.1 INTRODUCTION

This section describes the additional instructions and hardware that are contained in the MARK 9 CPU (which are not available on the MARK 5 CPU). The additional instructions are: hardware multiply, divide, and stack instructions. In addition, the MARK 9 contains a Mapped Memory that is capable of providing up to 256 1K-word pages of physical memory; up to 64 of these 1K-word pages may be mapped to logical memory at any given time. The MARK 9 CPU contains additional hardware and instructions which provide for the effective management of this memory map.

## 7.2 HARDWARE MULTIPLY AND HARDWARE DIVIDE

The following instruction multiplies a 16-bit unsigned number in accumulator 1 and a 16-bit unsigned number in accumulator 2:

```
    DOCP 2,1
```

The final product is added to the unsigned number in accumulator 0 and the result is placed in accumulators 0 and 1. Bit 0 of accumulator 0 is the most significant bit of the product. Bit 15 of accumulator 1 is the least significant bit of the product. Figure 7-1 illustrates the bit format for this instruction. The unsigned number in accumulator 2 is not changed.

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

089-9

**Figure 7-1. Hardware Multiply Instruction Format**

The following instruction divides a 32-bit unsigned number contained in accumulators 0 and 1 by a 16-bit unsigned number in accumulator 2 (accumulator 0 contains the most significant bits of the dividend):

```
    DOCS 2,1
```

Before division occurs, accumulators 0 and 2 are compared. If the number in accumulator 0 is less than the number in accumulator 2, division occurs, the quotient is stored in accumulator 1, accumulator 0 stores the remainder, and the carry bit is set to zero. The contents of accumulator 2 do not change. If the number in accumulator 0 is greater than or equal to the number in accumulator 2, the carry bit is set to 1, the operation is terminated, and all operands remain unchanged. Figure 7-2 illustrates the bit format for this instruction.

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

089-10

**Figure 7-2. Hardware Divide Instruction Format**

## 7.3 STACK OPERATION

Before performing the first stack operation, the stack and frame pointers must be initialized as follows:

1. The stack pointer must be initialized to the beginning address of the stack area minus 1.

2. If the main program uses the frame pointer, the frame pointer must be initialized to the same value as the stack pointer. Otherwise, the frame pointer can be initialized in a subroutine by the SAVE instruction.

During every instruction that pushes data onto the stack, a check is made for stack overflow. If the instruction places data in a word whose address is an integral multiple of 256 (decimal), a stack overflow is indicated. When a stack overflow is indicated, the following actions occur:

1. The current instruction is completed.

2. The internal Stack Overflow flag is set to 1.

3. A stack fault routine is performed if the Interrupt On flag is 1. If the Interrupt On flag is 0, the Stack Overflow flag remains set to 1. When interrupts are enabled, the stack fault routine is performed.

The stack fault routine performs the following actions:

1. Inhibits program map if enabled.

2. Sets Interrupt On flag to 0.

3. Sets Stack Overflow flag to 0.

4. Updates program counter.

5. Places updated program counter in physical location 0.

6. Executes jump indirect to physical location 3.

The stack instructions follow standard I/O instruction format. Stack manipulations occur within the CPU and memory. No data transfers take place in response to these instructions.

The MARK 9 CPU supports two sets of stack instructions: the EIS stack instructions described in Section 6.3.5, and the MARK 9-exclusive stack instructions described here.

The MARK 9-exclusive stack instructions are:

| | |
|---|---|
| DIB ac,1 | Push accumulator |
| DIBC ac,1 | Pop accumulator |
| DIC 0,1 | Save |
| DOA ac,1 | Move to stack pointer |
| NIO ac,1 | Move to frame pointer |
| DOAC ac,1 | Move from stack pointer |
| NIOC ac,1 | Move from frame pointer |
| DICC 0,1 | Return |

### 7.3.1 PUSH ACCUMULATOR

The following instruction pushes the contents of the designated accumulator on top of the stack:

```
    DIB ac,1
```

The contents of the accumulator do not change. Figure 7-3 illustrates the format for this instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

089-1

**Figure 7-3. Push Accumulator Instruction Format**

### 7.3.2 POP ACCUMULATOR

The following instruction pops the top word in the stack:

```
    DIBC ac,1
```

The popped word is placed in the designated accumulator. Figure 7-4 illustrates the format for the pop accumulator instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

089-31

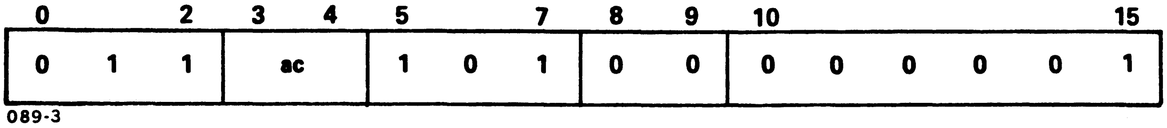
**Figure 7-4. Pop Accumulator Instruction Format**

### 7.3.3 SAVE

The following instruction saves the value of the stack pointer in the frame pointer and accumulator 3:

```
    DIC 0,1
```

Figure 7-5 illustrates the format for this instruction.



| 0 | | 2 | 3 4 | 5 | | 7 | 8 | 9 | 10 | | | | | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 7-5. Save Instruction Format**

When using 32K-word mode, the Save instruction pushes a five-word return block on the stack. Table 7-1 describes the destination of these five words.

**TABLE 7-1. 32K-WORD MODE STACK RETURN BLOCK DESTINATIONS**

| Word | Destination |
|---|---|
| 1 | Contains the contents of accumulator 0 |
| 2 | Contains the contents of accumulator 1 |
| 3 | Contains the contents of accumulator 2 |
| 4 | Bit 0 - always 0<br>Bits 1-15 - contain the value of the frame pointer before execution of this instruction |
| 5 | Bit 0 - represents Carry flag<br>Bits 1-15 - contain the contents of accumulator 3 |

When using 64K-word mode, the Save instruction pushes a six-word return block on the stack. Table 7-2 describes the destinations of these six words.

**TABLE 7-2. 64K-WORD MODE STACK RETURN BLOCK DESTINATIONS**

| Word | Destination |
|---|---|
| 1 | Contains the contents of accumulator 0 |
| 2 | Contains the contents of accumulator 1 |
| 3 | Contains the contents of accumulator 2 |
| 4 | Contains the value of the frame pointer before execution of this instruction |
| 5 | Contains the contents of accumulator 3 |
| 6 | Bit 0 - represents carry flag<br>Bits 1-15 - not defined |

After the last word of the return block is placed on the stack, the value of the stack pointer is placed in the frame pointer and accumulator 3. The contents of accumulators 0, 1, and 2 do not change.

### 7.3.4 MOVE TO STACK POINTER

The following instruction moves bits 0 through 15 of the designated accumulator into the stack pointer:

```
    DOA ac,1
```

The contents of the accumulator do not change. Figure 7-6 illustrates the format for this instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

089-4

Figure 7-6. Move to Stack Pointer Instruction Format

### 7.3.5 MOVE TO FRAME POINTER

The following instruction moves bits 0 through 15 of the designated accumulator into the frame pointer:

```
    NIO ac,1
```

The contents of the accumulator do not change. Figure 7-7 illustrates the format for this instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

089-5

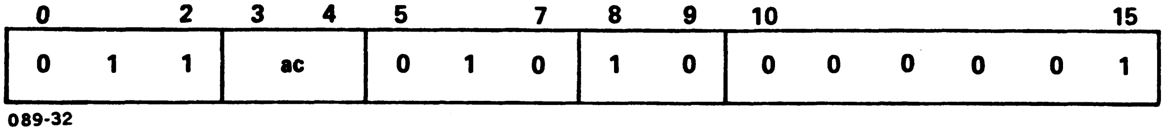
Figure 7-7. Move to Frame Pointer Instruction Format

### 7.3.6 MOVE FROM STACK POINTER

The following instruction moves the contents of the stack pointer into bits 0 through 15 of the designated accumulator:

```
    DOAC ac,1
```

The contents of the stack pointer do not change. Figure 7-8 illustrates the format for this instruction.



| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

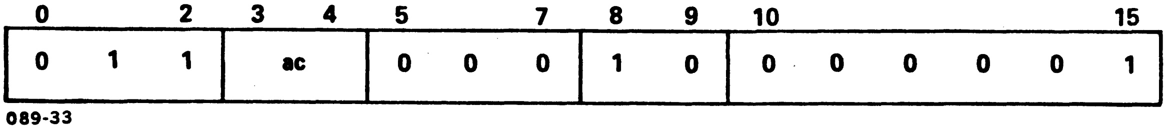
**Figure 7-8. Move from Stack Pointer Instruction Format**

### 7.3.7 MOVE FROM FRAME POINTER

The following instruction moves the contents of the frame pointer into bits 0 through 15 of the designated accumulator:

```
    NIOC ac,1
```

The contents of the frame pointer do not change. Figure 7-9 illustrates the format for this instruction.



| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

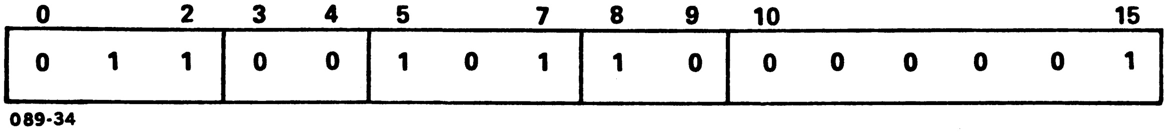
**Figure 7-9. Move from Frame Pointer Instruction Format**

### 7.3.8 RETURN

The following instruction places the contents of the frame pointer in the stack pointer:

```
    DICC 0,1
```

Figure 7-10 illustrates the format for this instruction.



| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Figure 7-10. Return Instruction Format**

When using 32K-word mode instructions, five words are popped from the stack after the frame pointer contents are stored in the stack pointer. Table 7-3 describes the destinations of these five words.

**TABLE 7-3. 32K-WORD MODE RETURN DESTINATIONS**

| Word | Destination |
|---|---|
| 1 | Bit 0 - placed in the carry flag<br>Bits 1-15 - placed in the program counter |
| 2 | Bits 1-15 - placed in the frame pointer<br>Bits 0-15 - placed in accumulator 3 |
| 3 | Placed in accumulator 2 |
| 4 | Placed in accumulator 1 |
| 5 | Placed in accumulator 0 |

When using 64K-word mode instructions, six words are popped from the stack after the frame pointer contents are stored in the stack pointer. Table 7-4 describes the destinations of these six words.

**TABLE 7-4. 64K-WORD MODE RETURN DESTINATIONS**

| Word | Destination |
|---|---|
| 1 | Bit 0 - placed in the carry flag<br>Bits 1-15 - not defined |
| 2 | Placed in accumulator 3 |
| 3 | Placed in frame pointer |
| 4 | Placed in accumulator 2 |
| 5 | Placed in accumulator 1 |
| 6 | Placed in accumulator 0 |

## 7.4 MAPPED MEMORY OPERATION

The map manages the MARK 9 memory. Two user maps with 32 pages of 1K-word blocks and two data channel maps with 32 pages of 1K-word blocks are available when using 32K-word addressing mode. Two user maps with 64 pages of 1K-word blocks, one data channel map with 64 pages of 1K-word blocks, and one utility map with 64 pages of 1K-word blocks are available when using 64K-word addressing mode.

A particular map--user or data channel--may be enabled or not. If the map is enabled, then all memory references for that operation--user or data channel--are mapped. If the map is not enabled, then all memory references for that operation will access the lower 32K or 64K words of memory.

The data channel maps are enabled or disabled if a Write Map Status instruction occurs with bit 1 of the designated accumulator set to 1 or 0, respectively.

The user or program map is enabled in two stages: first, a Write Map Status instruction must occur with bit 0 of the designated accumulator set to 1. Then, the user map is enabled after the first indirect memory reference following the Write Map Status instruction.

### 7.4.1 DEVICE INTERRUPT PROCESSING

The Map has been designed to allow for orderly processing of I/O interrupt requests by a supervisor program. When an I/O device requests an interrupt, the inhibit program map bit in the read-map status word is set to 1. This immediately disables the translating of user addresses so that the remainder of the interrupt process happens in the same manner as in those computers that have no address translation hardware. That is, the Interrupt On flag is set to 0, the updated program counter is placed in physical memory location 0, and the CPU executes a jump indirect to physical memory location 1. A similar process occurs for stack overflow, and normal trap instructions.

## 7.5 MEMORY MAP INSTRUCTIONS

The following instructions are used to support the mapped memory features of the POINT 4 MARK 9 CPU:

| | |
|---|---|
| DOB <f> ac,MAP | (load map) |
| DOA <f> ac,MAP1 | (initiate page check) |
| DIA <f> ac,MAP1 | (page check) |
| DIA <f> ac,MAP | (read map status) |
| DOA <f> ac,MAP | (write map status) |
| NIOP MAP | (map single-cycle) |
| NIOC MAP1 | (clear map) |
| DOC ac,MAP | (load map memory from main memory) |
| DIC ac,MAP | (load main memory from map memory) |

Map instructions follow the standard I/O instruction format. Data transfers take place between the CPU and map. No data transfers take place between the CPU and peripheral devices, although the instructions follow the I/O instruction format. Refer to Section 5.6 for detailed information on the I/O instruction format.

Device codes 2 and 3 are reserved for the memory mapped unit. The mnemonic MAP refers to device code 2. The mnemonic MAP1 refers to device code 3. The following flag control commands are used by device code 2:

f = S - reserved for future use

f = C - reserved for future use

f = P - the second non-data channel memory address is mapped after issuing this command; the proper map is indicated by the single-cycle select bit in the map status word

The following flag control commands are used by device code 3:

f = S - reserved for future use

f = C - the program map and data channel sections of the map are disabled; all internal map logic is initialized

f = P - reserved for future use

### 7.5.1 LOAD MAP

The following instruction loads the map memory by transferring the contents of the designated accumulator to the map memory:

```
    DOB [f] ac,MAP
```

The contents of the accumulator do not change. Figure 7-11 illustrates the format for this instruction.

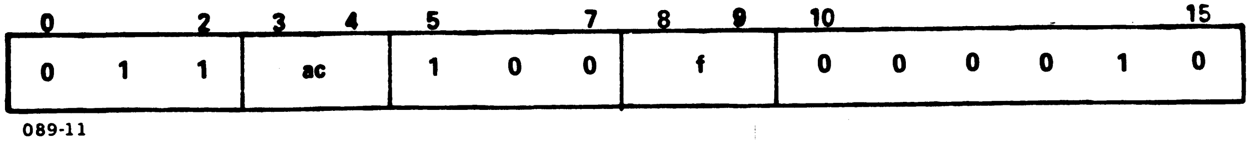


**Figure 7-11. Load Map Instruction Format**

Figure 7-12 illustrates the format for the designated accumulator. Table 7-5 describes the function of the accumulator bits.

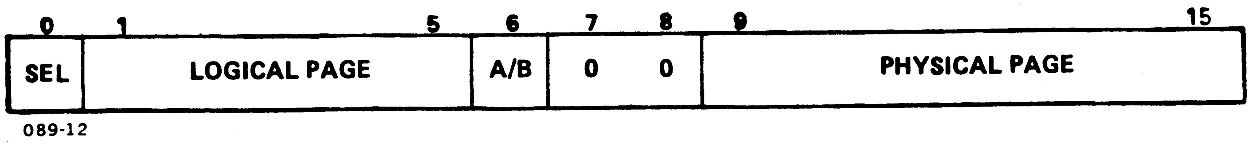


**Figure 7-12. Accumulator Format for Load Map Instruction**

TABLE 7-5. ACCUMULATOR FUNCTIONS FOR LOAD MAP INSTRUCTION

| Bit | Name | Function |
|---|---|---|
| 0 | SEL (map select) | 0 - address translation for the program map<br>1 - address translation for the data channel map |
| 1-5 | Logical page number | Range: 0-37 (octal) |
| 6 | A/B | 0 - address translation for map A of map selected by bit 0<br>1 - address translation for map B of map selected by bit 0 |
| 7-8 | Reserved | Must be set to 0 |
| 9-15 | Physical page number | Range: 0-177 (octal) |

### 7.5.2 INITIATE PAGE CHECK

The following instruction initiates a page check by transferring the contents of the designated accumulator and the map to a temporary register:

```
    DOA [f] ac,MAP1
```

The contents of the accumulator do not change. Figure 7-13 illustrates the instruction format for this instruction.

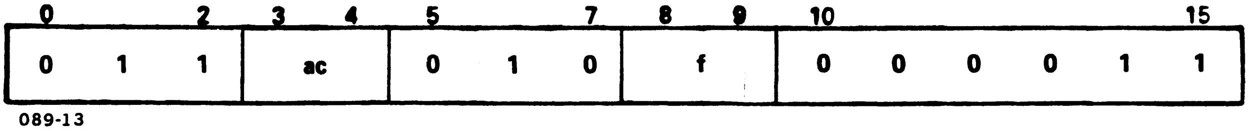


**Figure 7-13. Initiate Page Check Instruction Format**

Figure 7-14 illustrates the format for the designated accumulator. Table 7-6 describes the functions of the accumulator bits.

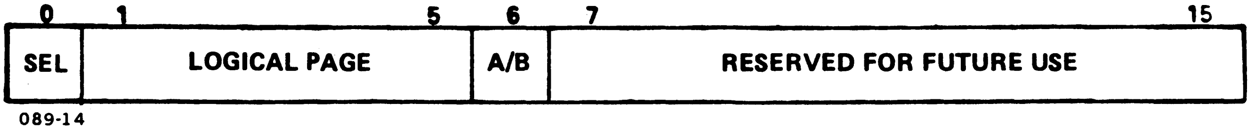


**Figure 7-14. Accumulator Format for Initiate Page Check Instruction**

### TABLE 7-6. ACCUMULATOR FUNCTIONS FOR INITIATE PAGE CHECK INSTRUCTION

| Bit | Name | Function |
|---|---|---|
| 0 | SEL (map select) | 0 - page check is for program map<br>1 - page check is for data channel map |
| 1-5 | Logical page | Range: 0-37 (octal) |
| 6 | A/B | 0 - page check for map A of map selected by bit 0<br>1 - page check for map B of map selected by bit 0 |
| 7-15 | Reserved | Should be set to 0 |

### 7.5.3 PAGE CHECK

The following instruction places the physical page number of the logical page in bits 9 through 15 of the designated accumulator:

```
    DIA [f] ac,MAP1
```

Bits 0 through 8 of the designated accumulator for the last Initiate Page Check instruction are placed in bits 0 through 8 of the designated accumulator. The initiate page check instruction (DOA [f] ac,MAP1) specifies the logical page. Figure 7-15 illustrates the format for this instruction.

| 0 |  | 2 | 3 4 | 5 |  | 7 | 8 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac | 0 | 0 | 1 | f | 0 | 0 | 0 | 0 | 1 | 1 |

089-15

**Figure 7-15. Page Check Instruction Format**

Figure 7-16 illustrates the format for the designated accumulator. Table 7-7 describes the functions of the accumulator bits.

| 0 | 1 5 | 6 | 7 | 8 | 9 15 |
|---|---|---|---|---|---|
| SEL | LOGICAL PAGE | A/B | 0 | 0 | PHYSICAL PAGE |

089-16

**Figure 7-16. Accumulator Format for Page Check Instruction**

TABLE 7-7. ACCUMULATOR FUNCTIONS FOR PAGE CHECK INSTRUCTION

| Bit | Name | Function |
|---|---|---|
| 0 | SEL (map select) | Copied from initiate page check instruction:<br>0 - page check for program map<br>1 - page check for data channel map |
| 1-5 | Logical page | Range: 0-37 (octal); copied from initiate page check instruction |
| 6 | A/B | Copied from initiate page check instruction:<br>0 - page check for map A of map selected by bit 0<br>1 - page check for map B of map selected by bit 0 |
| 7-8 | Reserved | Should be 0 |
| 9-15 | Physical page number | Physical page number - Range: 0-177 (octal); this page number is the physical page number of the logical page specified in the initiate page check instruction |

### 7.5.4 READ MAP STATUS

The following instruction reads map status:

DIA [f] ac,MAP

The 16-bit map status word is placed in the designated accumulator. Figure 7-17 illustrates the format for this instruction.

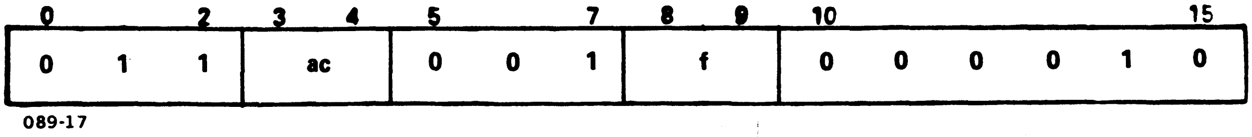


**Figure 7-17. Read Map Status Instruction Format**

Figure 7-18 illustrates the format for the designated accumulator. Table 7-8 describes the functions of the accumulator bits.

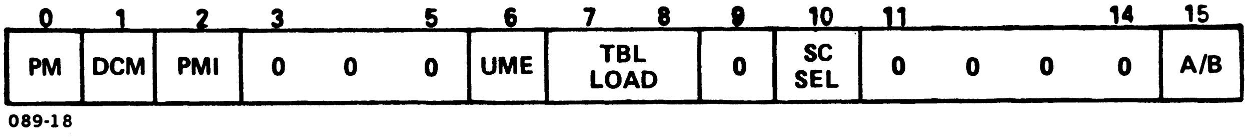


**Figure 7-18. Accumulator Format for Read Map Status Instruction**

## TABLE 7-8. ACCUMULATOR FUNCTIONS FOR READ MAP STATUS INSTRUCTION

| Bit | Name | Function |
|---|---|---|
| 0 | PM | 0 - disables program map<br>1 - enables program map |
| 1 | DCM | 0 - disables data channel map<br>1 - enables data channel map |
| 2 | PMI | 0 - program map not inhibited<br>1 - inhibits program map; overrides bit 0 |
| 3-5 | Reserved | Reserved for future use; set to 0 during normal operation* |
| 6 | UME | 0 - disables utility map<br>1 - enables utility map; overrides program map and map select (PM and A/B) |
| 7-8 | TBL Load | Specifies map for next DOC instruction:<br>00 - map A<br>01 - map B<br>10 - data channel map<br>11 - utility map |
| 9 | Reserved | Reserved for future use; should be 0 |
| 10 | SC SEL | Single-cycle select:<br>0 - A<br>1 - B |
| 11-14 | Reserved | Reserved for future use; should be 0 |
| 15 | A/B | Program map select:<br>0 - A<br>1 - B |
| *bit 5 is used during diagnostic testing | | |

### 7.5.5 WRITE MAP STATUS

The following instruction places the contents of the designated accumulator in the map status word:

```
    DOA [f] ac,MAP
```

The inhibit program map bit (bit 2) in the status word is set to 0. Figure 7-19 illustrates the format for this instruction.

| 0 |  | 2 | 3 4 | 5 |  | 7 | 8 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac | 0 | 1 | 0 | f | 0 | 0 | 0 | 0 | 1 | 0 |

089-19

**Figure 7-19. Write Map Status Instruction Format**

Figure 7-20 illustrates the format for the designated accumulator. Table 7-9 describes the functions of the accumulator bits.

| 0 | 1 | 2 |  |  | 5 | 6 | 7 8 | 9 | 10 | 11 |  |  | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PM | DCM | 0 | 0 | 0 | 0 | UME | TBL LOAD | 0 | SC A/B | 0 | 0 | 0 | 0 | A/B |

089-20

**Figure 7-20. Accumulator Format for Write Map Status Instruction**

### TABLE 7-9. ACCUMULATOR FUNCTIONS FOR WRITE MAP STATUS INSTRUCTION

| Bit | Name | Function |
|---|---|---|
| 0 | PM | Enable program map |
| 1 | DCM | Enable data channel map |
| 2-5 | Reserved | Reserved for future use; set to 0 during normal operation* |
| 6 | UME | Enables utility map; overrides program map and map select (PM and A/B) |
| 7-8 | TBL Load | Specifies map for next DOC instruction:<br>00 - map A<br>01 - map B<br>10 - data channel map<br>11 - utility map |
| 9 | Reserved | Reserved for future use; set to 0 |
| 10 | SC A/B | Single-cycle select:<br>0 - A<br>1 - B |
| 11-14 | Reserved | Reserved for future use; set to 0 |
| 15 | A/B | Program map select:<br>0 - A<br>1 - B |

*bit 5 is used during diagnostic testing

### 7.5.6 MAP SINGLE CYCLE

The following instruction maps the second nondata channel memory reference with the map indicated by the single-cycle select bit in the map status word:

```
    NIOP MAP
```

Figure 7-21 illustrates the format for this instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

089-24

**Figure 7-21. Map Single-Cycle Instruction Format**

### 7.5.7 CLEAR MAP

The following instruction clears the map:

```
    NIOC MAP1
```

This instruction disables the program and data channel maps. All map logic is initialized. Figure 7-22 illustrates the format for this instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

089-26

**Figure 7-22. Clear Map Instruction Format**

### 7.5.8 LOAD MAP MEMORY FROM MAIN MEMORY

The following instruction loads the map memory from main memory:

```
    DOC ac,MAP
```

The accumulator contains the address of the first memory location to use in loading the map. The map will continue to load until either 64 words are loaded or a -1 (177777 octal) is located. This instruction is used with the TBL bits in the Read Map Status instruction. Figure 7-23 illustrates the format for this instruction.

| 0 2 | 3 4 | 5 7 | 8 9 | 10 15 |
|---|---|---|---|---|
| 0 1 1 | ac | 1 1 0 | f | 0 0 0 0 1 0 |

089-27

**Figure 7-23. Load Map Memory From Main Memory Instruction Format**

Figure 7-24 illustrates the format for each entry of the table. Table 7-10 describes the function of each entry of the table.

| 0 5 | 6 | 7 | 8 15 |
|---|---|---|---|
| LOGICAL PAGE | 0 | 0 | PHYSICAL PAGE |

089-28

**Figure 7-24. Table Format for Load Map Memory from Main Memory Instruction**

## TABLE 7-10. TABLE FUNCTIONS FOR LOAD MAP MEMORY FROM MAIN MEMORY INSTRUCTION

| Bit | Name | Function |
|---|---|---|
| 0-5 | Logical page number | Range 0-77 (octal) |
| 6-7 | Reserved | Reserved for future use; set to 0 |
| 8-15 | Physical page number | Range 0-377 |

### 7.5.9 LOAD MAIN MEMORY FROM MAP MEMORY

The following instruction loads memory from the map memory:

```
    DIC ac,MAP
```

64 words of the map will be stored into memory starting at the address in ac. This instruction is used with the TBL bits in the status word. Figure 7-25 illustrates the format for this instruction.

| 0 |  | 2 | 3 | 4 | 5 |  | 7 | 8 | 9 | 10 |  |  |  |  | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | ac |  | 1 | 0 | 1 | f |  | 0 | 0 | 0 | 0 | 1 | 0 |

089-29

**Figure 7-25. Load Main Memory from Map Memory Instruction Format**

Figure 7-26 illustrates the format for each entry of the table. Table 7-11 describes the function of each entry of the table.

| 0 | 5 | 6 | 7 | 8 | 15 |
|---|---|---|---|---|---|
| LOGICAL PAGE |  | 0 | 0 | PHYSICAL PAGE |  |

089-28

**Figure 7-26. Table Format for Load Main Memory from Map Memory Instruction**

### TABLE 7-11. TABLE FUNCTIONS FOR LOAD MAIN MEMORY FROM MAP MEMORY INSTRUCTION

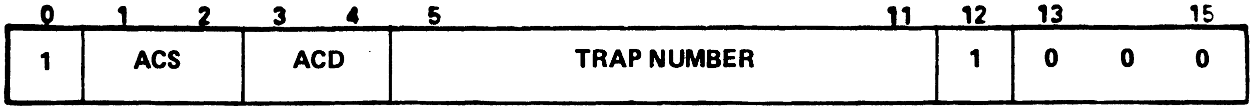
| Bit | Name | Function |
|---|---|---|
| 0-5 | Logical page number | Range 0-77 (octal) |
| 6-7 | Reserved | Reserved for future use; should be 0 |
| 8-15 | Physical page number | Range 0-377 |

## 7.6 TRAP INSTRUCTION

The following instruction causes ALU traps when in 32K mode:

ALU NO LOAD, NO SKIP

The logical address of this instruction is placed in bits 0 through 15 of memory location 46 (octal). The processor places the contents of location 47 (octal) into the program counter. This instruction is disabled when either the 64K mode (MARK 5/9) or the Extended Instruction Set is enabled. Figure 7-27 illustrates the format for this instruction. Table 7-12 describes the functions of the instruction bits.



089-30

**Figure 7-27. Trap Instruction Format**

**TABLE 7-12. TRAP INSTRUCTION FUNCTIONS**

| Bit | Name | Function |
|---|---|---|
| 0 | None | Set to 1; indicates arithmetic/logic instruction group |
| 1-2 | ACS | Designates source accumulator |
| 3-4 | ACD | Designates destination accumulator |
| 5-11 | Trap | Specified by operation code |
| 12 | NL | No-load field; set to 1; does not store shifter output (refer to Section 5.6) |
| 13-15 | None | Skip control field; set to 0; does not allow skip test (refer to Section 5.5) |

**CAUTION**

Destruction of data may result if care is not exercised when using this instruction. The MARK 5 CPU executes an ALU NO LOAD, NO SKIP instruction as a no operation. The 32K mode in the MARK 9 CPU executes the ALU NO LOAD, NO SKIP instruction as a trap. The EIS mode in MARK 5 and MARK 9 CPUs uses locations 46 (octal) as the frame pointer and 47 (octal) as the stack limit pointer. These locations are checked when a stack operation is performed.

# Section 8
# OPTIONAL FEATURES

## 8.1 INTRODUCTION

This section describes the following optional features:

- Battery Backup
- Operator Control Unit
- Voltage Regulator
- Micro-programmed Floating Point Unit (MFPU)
- Mapped Memory

**NOTE**

> Unless otherwise noted, information in this section applies to the MARK 5, MARK 5 with EIS, and MARK 9 computers.

## 8.2 BATTERY BACKUP

The Battery Backup option provides three features:

- Memory contents protection for at least 2 hours in case of power failure
- Power fail auto-restart after resumption of power service
- Standby mode, which removes Main Power Regulator +5V, ±15V and -5V power, but maintains +5V BU, -5V BU, and +12V BU power to the CPU slot to preserve memory contents

**NOTE**

> The MARK 9 CPU only uses +5V and +5V BU.

### 8.2.1 BATTERY BACKUP VOLTAGES

The battery unit is maintained in a charged state by the power supply as long as the unit is plugged in, and AC power is available. The battery backup module includes one 12V sealed lead acid battery which provides the following voltages to the CPU/memory board and Mini-Panel in case of power failure:

| Voltage | Supplied to |
|---|---|
| + 5V BU | CPU board memory refresh logic, Mini-Panel |
| - 5V BU | For memory on CPU board (MARK 5) |
| +12V BU | For memory on CPU board (MARK 5) |

Battery backup voltages can be monitored on the power supply chassis Mini-Panel by the light-emitting diode (LED) labeled as BU voltages. Illumination of these LED indicators signifies that these voltages are in tolerance. If one or more of these LED indicators is not illuminated, one of the following conditions exists:

| AC Power | LED Indicator OFF |
|---|---|
| ON | Backup power supply is out of tolerance* |
| OFF | Batteries are discharged and memory contents are lost |

*Battery recharging after a discharge causes the LED indicators to blink.

### 8.2.2 POWER MONITOR AUTO-RESTART HANDLING

When AC power fails, an unmaskable interrupt is produced. A power-fail interrupt is indicated if the CPU Done flag is set to 1. Two input/output CPU instructions are associated with a power-fail interrupt:

| Instruction | Function |
|---|---|
| SKPDN | Skip if a power-fail interrupt has occurred |
| SKPDZ | Skip if a power-fail interrupt has not occurred |

These instructions may be used to transfer control to a routine for handling power-fail conditions, and performing the following functions:

- Save the accumulators, Carry flag, and program counter (PC) where the interrupt occurred. The PC is in location 0 of memory, and must be moved to another location.
- Perform other cleanup functions that may be necessary.
- Put an appropriate auto-restart instruction in memory location 0.
- Halt.

While the power-fail condition exists, the Backup Battery Unit retains the contents of memory for at least 2 hours, if the batteries were fully charged when the power-fail occurred.

When AC power is restored, the CPU will begin execution at memory location 0 if the processor Mini-Panel keyswitch is set to AUTO. If the keyswitch is not set to AUTO, the processor will halt, with the program counter equal to 0, pending operator action.

## 8.3 OPERATOR CONTROL UNIT (OCU)

The OCU enhances operator access to the CPU. An extensive discussion of Operator Control Unit capabilities and operating procedures is in Section 3.4. Section 2.7 describes installation of the OCU.

## 8.4 VOLTAGE REGULATOR

The Voltage Regulator option provides the user with a 12-volt regulator on the PC board itself. This option is designed for use in a chassis that does not provide a regulated +12V supply, as in a typical Nova-type chassis with a +15V unregulated supply. This regulator converts an unregulated supply of at least +13.2 volts to +12 volts of regulated voltage.

**NOTE**

> The MARK 9 CPU requires only +5V. The +12V is not used by the MARK 9 CPU.

## 8.5 MICRO-PROGRAMMED FLOATING POINT UNIT

The Micro-Programmed Floating Point Unit (MFPU) is a firmware option designed to serve as a floating point binary arithmetic unit, which significantly reduces floating-point computation time. MFPU is available only on MARK 5 CPUs that also have the EIS option.

### 8.5.1 MFPU FORMATS

The MFPU allows two number formats: single precision (two words) and double precision (four words). In both formats, the first word contains the sign, exponent, and most significant eight bits of the mantissa. The exponent is 7 bits long, in excess-64 notation, and represents a power of 16. Data used for a calculation is assumed to be normalized (i.e., the first four bits of the mantissa are not all 0); after the calculation, the result is normalized and stored. If the first word of a number is equal to 0, the MFPU interprets the entire number as equal to 0. The MFPU formats are shown in Figure 8-1.

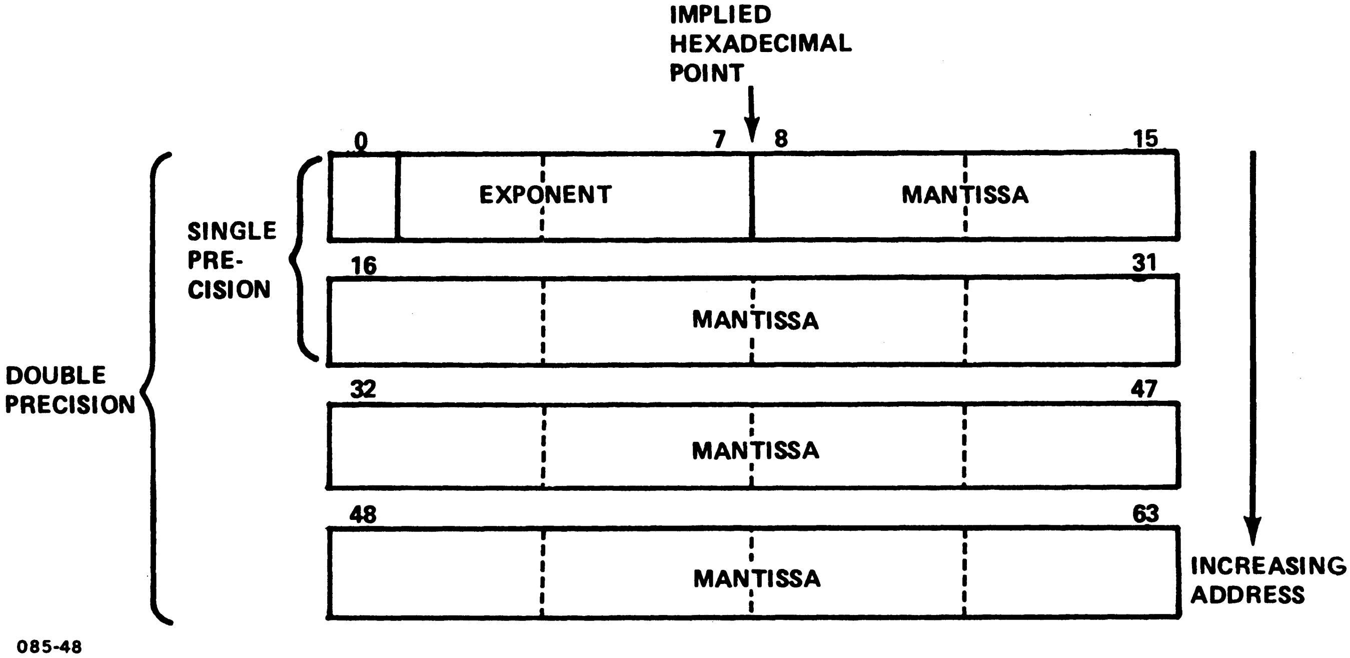


Figure 8-1. MFPU Formats

### 8.5.2 MFPU COMMANDS

Ten MFPU commands for single and double precision may be specified. Table 8-1 defines these commands. Their binary formats are shown in Table 8-2.

**TABLE 8-1. MFPU COMMANDS**

| Command | Definition |
|---|---|
| FPADD | Add single precision |
| FPSUB | Subtract single precision |
| FPMUL | Multiply single precision |
| FPDIV | Divide single precision |
| FDADD | Add double precision |
| FDSUB | Subtract double precision |
| FDMUL | Multiply double precision |
| FDDIV | Divide double precision |
| FIX* | Convert floating single precision to one-word, fixed-point unsigned integer (truncated) |
| FLT* | Convert one-word, fixed-point unsigned integer to floating single precision |

*Double-precision forms of FIX and FLT are not provided because:

- FIX - the last two words have no effect on the answer; the single-precision floating form is already more precise than a one-word integer.
- FLT - for a double-precision answer, set the last two words to 0.

**TABLE 8-2. MFPU COMMAND FORMATS**

| Command* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FPADD | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FPSUB | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FPMUL | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FPDIV | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FDADD | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FDSUB | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FDMUL | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FDDIV | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FIX | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| FLT | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

*These commands would be no-op skips in a MARK 5 (with EIS) without MFPU.

### 8.5.3 OPERAND ADDRESSING

All MFPU commands are two-address instructions:

- A0 holds the address of the source operand
- A1 holds the address of the destination operand

For example, if A0=S and A1=D, MFPU commands would result in the following operations:

| | |
|---|---|
| ADD | D <-- D+S |
| SUB | D <-- D-S |
| MUL | D <-- D*S |
| DIV | D <-- D/S |
| FIX | D <-- S(converted) |
| FLT | D <-- S(converted) |

Note that S remains unchanged in all cases. Accumulators A2 and A3 are not changed by the MFPU, but the carry is left in an undefined state.

### 8.5.4 OVERFLOW AND UNDERFLOW

All commands (except FLT) when executed without error (overflow condition) cause the next instruction in the sequence to be skipped; the next instruction is executed in case of error/overflow, as shown in Table 8-3.

TABLE 8-3. ERROR OVERFLOW

| Command | Overflow |
|---|---|
| FPADD,FDADD | Absolute value of result $\geq 16^{63}$ |
| FPSUB,FDSUB | Absolute value of result $\geq 16^{63}$ |
| FPMUL,FDMUL | Absolute value of result $\geq 16^{63}$ |
| FPDIV,FDDIV | Divisor = 0 or result $\geq 16^{63}$ |
| IX | Value $\geq 2^{16}$ |
| FLT | No error possible |

In case of underflow (result < $16^{-65}$), the result is set = 0 and no error indication is given.

### 8.5.5 ENABLING AND DISABLING

MFPU instructions are enabled and disabled at the same time as the Extended Instruction Set (see Section 6.2).

## 8.6 256K WORD MEMORY

The MARK 9 is equipped with 128K words of memory. Optional memory may be purchased to increase main memory size to 256K words.

# APPENDICES

# Appendix A
# VON NEUMAN MAP OF POINT 4 MARK 5/9 STANDARD INSTRUCTIONS

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MEMORY REFERENCE JMP | 0 | 0 | 0 | 0 | 0 | I | X | | DISPLACEMENT | | | | | | | |
| JSR | 0 | 0 | 0 | 0 | 1 | I | X | | DISPLACEMENT | | | | | | | |
| ISZ | 0 | 0 | 0 | 1 | 0 | I | X | | DISPLACEMENT | | | | | | | |
| DSZ | 0 | 0 | 0 | 1 | 1 | I | X | | DISPLACEMENT | | | | | | | |
| LDA | 0 | 0 | 1 | ACC | | I | X | | DISPLACEMENT | | | | | | | |
| STA | 0 | 1 | 0 | ACC | | I | X | | DISPLACEMENT | | | | | | | |
| I/O | 0 | 1 | 1 | ACC | | OPCODE | | | CONTROL | | DEVICE CODE | | | | | |
| ALU INSTRUCTIONS | 1 | ACS | | ACD | | OPCODE | | | SH | | CY | | NL | SK | | |

| I | |
|---|---|
| 0 | — |
| 1 | @ |

| X | |
|---|---|
| 00 | ABS |
| 01 | REL |
| 10 | B2 |
| 11 | B3 |

| ACC | |
|---|---|
| 00 | A0 |
| 01 | A1 |
| 10 | A2 |
| 11 | A3 |

| OPCODE | |
|---|---|
| 000 | NIO |
| 001 | DIA |
| 010 | DOA |
| 011 | DIB |
| 100 | DOB |
| 101 | DIC |
| 110 | DOC |
| 111 | SKP |

| CONTROL | XFR | SKP |
|---|---|---|
| 00 | — | BN |
| 01 | S | BZ |
| 10 | C | DN |
| 11 | P | DZ |

| ACS | |
|---|---|
| 00 | A0 |
| 01 | A1 |
| 10 | A2 |
| 11 | A3 |

| ACD | |
|---|---|
| 00 | A0 |
| 01 | A1 |
| 10 | A2 |
| 11 | A3 |

| OPCODE | |
|---|---|
| 000 | COM |
| 001 | NEG |
| 010 | MOV |
| 011 | INC |
| 100 | ADC |
| 101 | SUB |
| 110 | ADD |
| 111 | AND |

| SH | |
|---|---|
| 00 | — |
| 01 | L |
| 10 | R |
| 11 | S |

| CY | |
|---|---|
| 00 | — |
| 01 | Z |
| 10 | O |
| 11 | C |

| NL | |
|---|---|
| 0 | — |
| 1 | # |

| SK | |
|---|---|
| 000 | — |
| 001 | SKP |
| 010 | SZC |
| 011 | SNC |
| 100 | SZR |
| 101 | SNR |
| 110 | SEZ |
| 111 | SBN |

082–37

# Appendix B
# POINT 4 MARK 5/9 STANDARD INSTRUCTION REFERENCE CHART

## ARITH/LOGIC

| | |
|---|---|
| 100000 | COM |
| 100400 | NEG |
| 101000 | MOV |
| 101400 | INC |
| 102000 | ADC |
| 102400 | SUB |
| 103000 | ADD |
| 103400 | AND |

**SOURCE**

| | |
|---|---|
| 0 | 0 |
| 20000 | 1 |
| 40000 | 2 |
| 60000 | 3 |

**DESTINATION**

| | |
|---|---|
| 0 | 0 |
| 4000 | 1 |
| 10000 | 2 |
| 14000 | 3 |

**SHIFT**

| | |
|---|---|
| 100 | L |
| 200 | R |
| 300 | S |

**CARRY**

| | |
|---|---|
| 20 | Z |
| 40 | O |
| 60 | C |

**NO–LOAD**

| | |
|---|---|
| 10 | # |

**SKIP CONDITION**

| | |
|---|---|
| 1 | SKP |
| 2 | SZC |
| 3 | SNC |
| 4 | SZR |
| 5 | SNR |
| 6 | SEZ |
| 7 | SBN |

## MEMORY REFERENCE

| | |
|---|---|
| 0 | JMP |
| 4000 | JSR |
| 10000 | ISZ |
| 14000 | DSZ |
| 20000 | LDA |
| 40000 | STA |

**ACCUMULATOR**

| | |
|---|---|
| 0 | 0 |
| 4000 | 1 |
| 10000 | 2 |
| 14000 | 3 |

**INDIRECT**

| | |
|---|---|
| 2000 | @ |

**ADDRESS MODE**

| | |
|---|---|
| 0 | ABS |
| 400 | REL |
| 1000 | BASE2 |
| 1400 | BASE3 |

**DISPLACEMENT**

| | |
|---|---|
| 0–177 | POS. |
| 200–377 | NEG. |

**SPECIAL ARITHMETIC TESTS**

| | |
|---|---|
| 101014 | SKZ |
| 101015 | SNZ |
| 101112 | SSP |
| 101113 | SSN |
| 102032 | SGE |
| 102033 | SLS |
| 102414 | SEQ |
| 102415 | SNE |
| 102432 | SGR |
| 102433 | SLE |

## INPUT/OUTPUT

| | |
|---|---|
| 60000 | NIO |
| 60400 | DIA |
| 61000 | DOA |
| 61400 | DIB |
| 62000 | DOB |
| 62400 | DIC |
| 63000 | DOC |

**ACCUMULATOR**

| | |
|---|---|
| 0 | 0 |
| 4000 | 1 |
| 10000 | 2 |
| 14000 | 3 |

**I/O PULSE**

| | |
|---|---|
| 100 | S |
| 200 | C |
| 300 | P |

**I/O SKIP**

| | |
|---|---|
| 63400 | SKPBN |
| 63500 | SKPBZ |
| 63600 | SKPDN |
| 63700 | SKPDZ |

**DEVICE CODE**

| | |
|---|---|
| 10 | TTI |
| 11 | TTO |
| 12 | PTR |
| 13 | PTP |
| 14 | RTC |
| 17 | LPT |

**SPECIAL CPU INSTRUCTIONS**

| | |
|---|---|
| 60177 | INTEN |
| 60277 | INTDS |
| 60477 | READS |
| 61477 | INTA |
| 62077 | MSKO |
| 62677 | IORST |
| 63077 | HALT |

082–38

# Appendix C
# ASCII CODE CHART

**ASCII CODE in OCTAL**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 000 | NUL | <CTRL-@> | 040 | BLANK | 100 | @ | 140 | ` |
| 001 | SOH | <CTRL-A> | 041 | ! | 101 | A | 141 | a |
| 002 | STX | <CTRL-B> | 042 | " | 102 | B | 142 | b |
| 003 | ETX | <CTRL-C> | 043 | # | 103 | C | 143 | c |
| 004 | EOT | <CTRL-D> | 044 | $ | 104 | D | 144 | d |
| 005 | ENO | <CTRL-E> | 045 | % | 105 | E | 145 | e |
| 006 | ACK | <CTRL-F> | 046 | & | 106 | F | 146 | f |
| 007 | BEL | <CTRL-G> | 047 | ' | 107 | G | 147 | g |
| 010 | BKSP | <CTRL-H> | 050 | ( | 110 | H | 150 | h |
| 011 | HTAB | <CTRL-I> | 051 | ) | 111 | I | 151 | i |
| 012 | LF | <CTRL-J> | 052 | * | 112 | J | 152 | j |
| 013 | VTAB | <CTRL-K> | 053 | + | 113 | K | 153 | k |
| 014 | FF | <CTRL-L> | 054 | , | 114 | L | 154 | l |
| 015 | CR | <CTRL-M> | 055 | - | 115 | M | 155 | m |
| 016 | SO | <CTRL-N> | 056 | . | 116 | N | 156 | n |
| 017 | SI | <CTRL-O> | 057 | / | 117 | O | 157 | o |
| 020 | DLE | <CTRL-P> | 060 | 0 | 120 | P | 160 | p |
| 021 | XON | <CTRL-Q> | 061 | 1 | 121 | Q | 161 | q |
| 022 | AUXON | <CTRL-R> | 062 | 2 | 122 | R | 162 | r |
| 023 | XOFF | <CTRL-S> | 063 | 3 | 123 | S | 163 | s |
| 024 | AUXOFF | <CTRL-T> | 064 | 4 | 124 | T | 164 | t |
| 025 | NAK | <CTRL-U> | 065 | 5 | 125 | U | 165 | u |
| 026 | SYN | <CTRL-V> | 066 | 6 | 126 | V | 166 | v |
| 027 | ETB | <CTRL-W> | 067 | 7 | 127 | W | 167 | w |
| 030 | CAN | <CTRL-X> | 070 | 8 | 130 | X | 170 | x |
| 031 | ENDMD | <CTRL-Y> | 071 | 9 | 131 | Y | 171 | y |
| 032 | SUB | <CTRL-Z> | 072 | : | 132 | Z | 172 | z |
| 033 | ESC | <CTRL-[> | 073 | ; | 133 | [ | 173 | { |
| 034 | F SEP | <CTRL-\> | 074 | < | 134 | \ | 174 | \| |
| 035 | G SEP | <CTRL-]> | 075 | = | 135 | ] | 175 | } |
| 036 | R SEP | <CTRL-^> | 076 | > | 136 | ^ | 176 | ~ |
| 037 | U SEP | <CTRL-_> | 077 | ? | 137 | _ | 177 | DEL |

# Appendix D
# VIRTUAL CONTROL PANEL COMMANDS (MANIP)

| | |
|---|---|
| | APL does Auto-Boot if mini-switches are set to 200 + device code |
| A | Display PC, A0, A1, A2, A3, and carry |
| Cx,y (x<5) | Change accumulator x (or carry if x=4) to value y |
| Cx (x>4) | Convert real address x to virtual |
| Dx,a | Dump memory in octal, beginning at address x, using addressing mode a |
| Ex,a | Enable entry at address x, using addressing mode a |
| Fx,y | Establish offset x-y, where x=real memory address, y=virtual (listing) address |
| Jx | Jump to location x, after restoring accumulators and carry |
| Kx,y,z | Store constant z in memory locations x through y |
| Mx,y,z | Move memory block x through y to location z |
| Nx,y,z,m | Search memory x through y for not-equal to z, using mask m (optional) |
| Ox,a | Output memory in ASCII, starting at location x, using addressing mode a, until a zero byte is encountered |

where a is an addressing mode parameter, defined as follows:

| a Parameter | Definition |
|---|---|
| omitted | Word address, including "F" offset, if any |
| 0 | Word address, absolute |
| 1 | Byte address, using offset, if any |
| 2 | Byte address, lower 64KB |

| | |
|---|---|
| Px | Program load from DMA device code x; if x is omitted, reads switches |
| Sx,y,z,m | Search memory x through y for the value z, using mask m (optional) |
| Xx,y | Calculate checksum over x through y |
| Yx | Set up a delay after each CR/LF; x=0 for maximum delay; x=177777 for none |
| x:y | Enter value y at address x, and open next cell for entry |
| ^ | Open previous cell for entry |
| CTRL ( ) | Transmit control character to CTU |

# Appendix E
# PROGRAMMING EXAMPLES

## E.1 NUMBER HANDLING

### E.1.1 GENERATING NUMBERS

Six numbers can be generated with single instructions:

```
        SUB     0,0 --> 0
        SUBZL   0,0 --> 1
        SUBZR   0,0 --> 100000
        ADC     0,0 --> 177777  (= -1)
        ADCZL   0,0 --> 177776  (= -2)
        ADCZR   0,0 --> 77777
```

### E.1.2 NUMBER TESTING

Twenty different sets of numbers can be tested with a single instruction. Figure E-1 shows the conditions under which each of the basic arithmetic test instructions will skip. Figure E-2 shows which instructions to use to test the 20 sets of numbers. The skip condition can be changed from a zero-test to a nonzero-test to obtain the complements of the 20 sets.

| Instruction | SZR skips if: | SZC skips if: |
|---|---|---|
| MOVZ | 0 | all |
| MOVO | 0 | none |
| MOVZL | 0,100000 | >=0 |
| MOVOL | none | >=0 |
| MOVZR | 0,1 | even |
| MOVOR | none | even |
| | | |
| COMZ | -1 | 0 |
| COMO | -1 | none |
| COMZL | -1,-2 | <0 |
| COMOL | none | <0 |
| COMZR | -1,77777 | odd |
| COMOR | none | odd |
| | | |
| INCZ | -1 | not -1 |
| INCO | -1 | -1 |
| INCZL | 77777 | -1<=x<=77777 |
| INCOL | -1 | -1<=x=77777 |
| INCZR | 0 | odd |
| INCOR | -1 | odd |
| | | |
| NEGZ | 0 | not 0 |
| NGEO | 0 | -1 |
| NEGZL | 100000 | -77777<=X<0 |
| NEGOL | 0 | -77777<=x<0 |
| NEGZR | -1 | even |
| NEGOR | 0 | even |
| | | |
| ADDZ | 0,100000 | >=0 |
| ADDO | 0,100000 | <0 |
| ADDZL | 0 | 2d bit = 0 |
| ADDOL | 100000 | 2d bit = 0 |
| ADDZR | 0 | all |
| ADDOR | 100000 | all |

**NOTE**

For ADD instructions, ACS and ACD must be the same.

**Figure E-1. Conditions Under Which Each of the BASIC Arithmetic Test Instructions Will Skip**

|  |  |  |  |  |  |  |  |  |  |  | + ← | → − |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSTRUCTION | TESTS FOR | 0 | 1 | 2 | 3 | 37776 | 37777 | 40000 | 40001 | 77776 | 77777 | 100000 | 100001 | 137776 | 137777 | 140000 | 140001 | 177774=−4 | 177775=−3 | 177776=−2 | 177777=−1 |  |
| MOV 0,0,SZR | 0 | ● |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 VALUE |
| INCZL 0,0,SZR | 77777 |  |  |  |  |  |  |  |  |  | ● |  |  |  |  |  |  |  |  |  |  |  |
| NEGZL 0,0,SZR | 100000 |  |  |  |  |  |  |  |  |  |  | ● |  |  |  |  |  |  |  |  |  |  |
| COM 0,0,SZR | −1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ● |  |
| MOVZR 0,0,SZR | 0,1 | ● | ● |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 2 VALUES |
| COMZR 0,0,SZR | −1,2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ● | ● |  |
| MOVZL 0,0,SZR | 0,100000 | ● |  |  |  |  |  |  |  |  |  | ● |  |  |  |  |  |  |  |  |  |  |
| COMZL 0,0,SZR | 77777,−1 |  |  |  |  |  |  |  |  |  | ● |  |  |  |  |  |  |  |  |  | ● |  |
| MOVL 0,0,SZC | ≥0 | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |  |  |  |  |  |  |  |  |  |  | 32K |
| INCL 0,0,SZC | −1≤X≤77776 | ● | ● | ● | ● | ● | ● | ● | ● | ● |  |  |  |  |  |  |  |  |  |  | ● |  |
| NEGL 0,0,SZC | −77777≤X≤0 | ● |  |  |  |  |  |  |  |  |  |  | ● | ● | ● | ● | ● | ● | ● | ● | ● |  |
| MOVR 0,0,SZC | EVEN | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  |  |
| ADDL 0,0,SZC | 2D BIT = 0 | ● | ● | ● | ● | ● | ● |  |  |  |  | ● | ● | ● | ● |  |  |  |  |  |  |  |
| MOVZR 0,0,SEZ | EVEN OR 1 | ● | ● | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | 32K + 1 |
| NEGZR 0,0,SEZ | EVEN OR −1 | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● | ● |  |
| INCZR 0,0,SEZ | ODD OR 0 | ● | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  |
| COMZR 0,0,SEZ | ODD OR −2 |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● |  | ● | ● | ● |  |
| MOVZL 0,0,SEZ | ≥0 OR 100000 | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |  |  |  |  |  |  |  |  |  | 32K + 1 |
| COMZL 0,0,SEZ | <0 OR 77777 |  |  |  |  |  |  |  |  |  | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |  |
| NEGZL 0,0,SEZ | ≤0 | ● |  |  |  |  |  |  |  |  |  | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |  |

085-49

Figure E-2. The 20 Different Sets of Numbers Which Can Be Tested with a Single Instruction

## E.2 BIT TESTING

Any bit in a word can be tested with a maximum of three instructions, without requiring another accumulator. Three bit positions can be tested with just one instruction (bits 0, 1, and 15). Seven bit positions (bits 2, 3, 7, 8, 9, 10, and 14) require two instructions; the other six bit positions require three. Figure E-3 shows which instructions to use to test for any bit in a word.

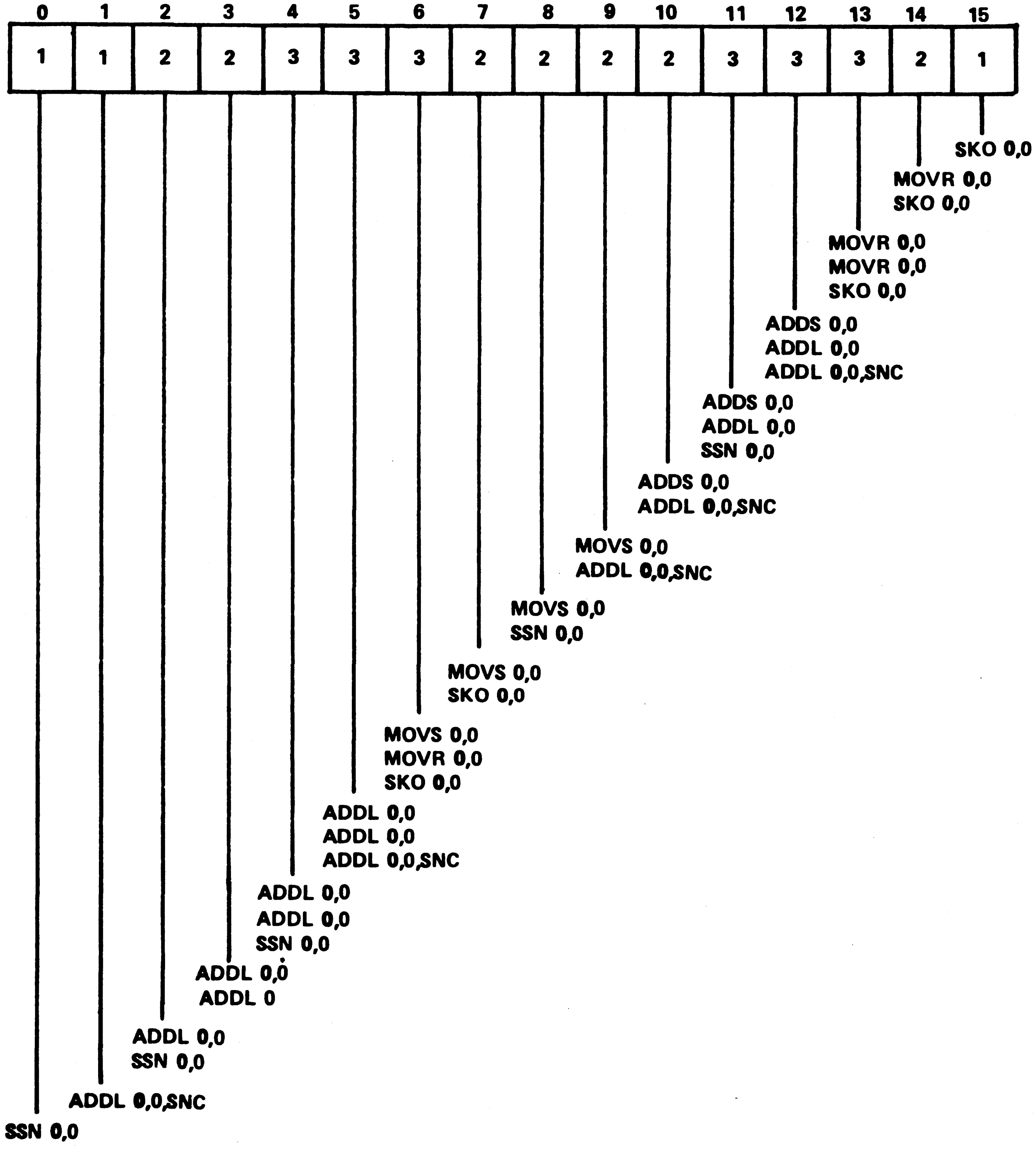


**Figure E-3. How to Test for Any Bit in a Word**

# E.3 ACCUMULATOR HANDLING

### E.3.1 TO "OR" TWO ACCUMULATORS

The following routine forms the inclusive-OR of A0 and A1 in A1. The routine uses the fact that an arithmetic ADD is equivalent to an OR if corresponding bits in the two operands are not both 1.

```
    COM    0,0
    AND    0,1  ;remove those bits where both are 1
    ADC    0,1  ;then add original value
```

### E.3.2 TO "EXCLUSIVE-OR" TWO ACCUMULATORS

This routine utilizes the fact that an arithmetic ADD is the same as an exclusive-OR except for the carry bits.

```
    MOV    1,2
    ANDZL  0,2  ;form the carry bits
    ADD    0,1  ;add the original operands
    SUB    2,1  ;remove the carry bits
```

This routine destroys the Carry flag. To preserve the carry at the expense of an additional instruction, use the following:

```
    COM    1,2              __
    AND    0,2  ;forms A0 * A1
    COM    0,0  __
    AND    0,1  ;A0 * A1        __   __
    ADD    2,1  ;A0 + A1 = A0 * A1 + A0 * A1
```

### E.3.3 TO "DECREMENT" AN ACCUMULATOR

The following routine is used to decrement an accumulator:

```
    NEG    0,0
    COM    0,0
```

### E.3.4 TO "COMPLEMENT" THE MOST SIGNIFICANT BIT

The following instruction complements the MSB and clears the carry bit:

```
    ADDOR  0,0
```

## E.4 PARITY GENERATION OR CHECKING

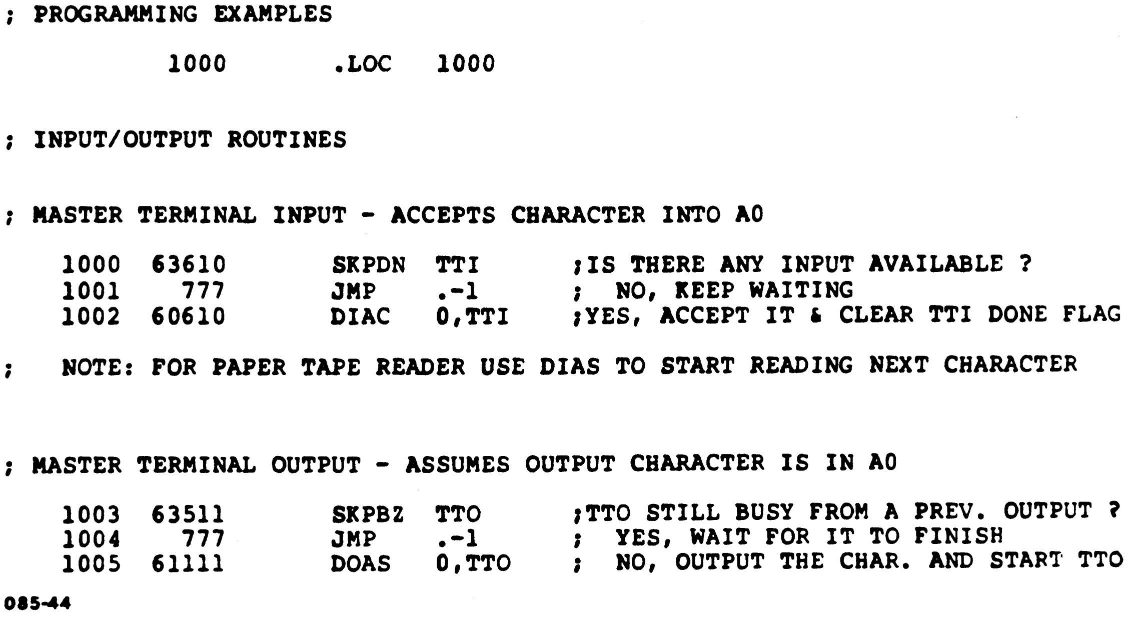
The two-instruction loop

```
    ADD   0,0,SZR
    JMP   .-1
```

will complement the original carry if A0 had odd parity, and leave it unchanged if A0 had even parity.

## E.5 I/O PROGRAMMING FOR THE MASTER TERMINAL

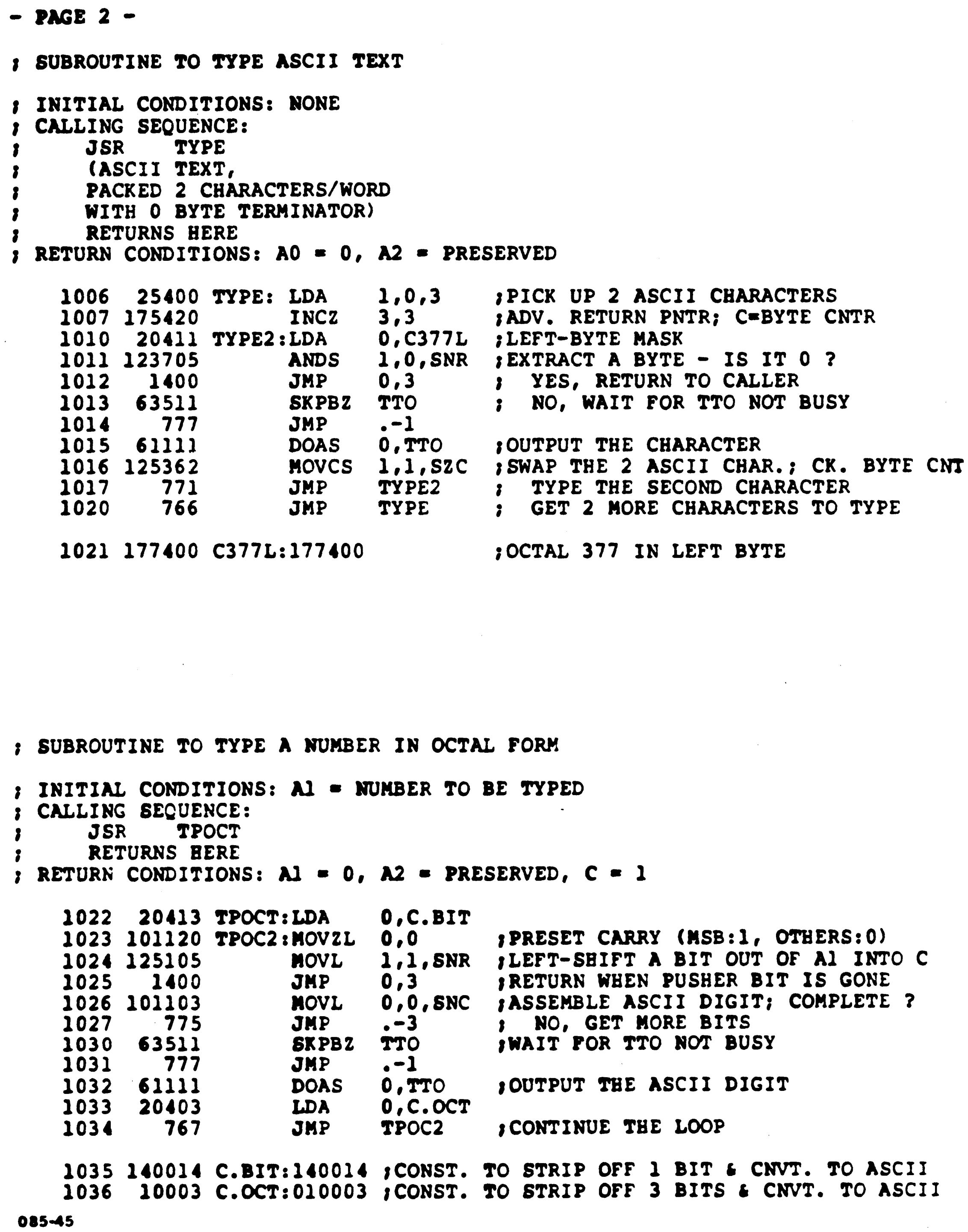
The assembler listings in Figure E-4 provide examples of inputting and outputting to a master terminal (Teletype or CRT), using the standard device code 10/11-type controller. They also illustrate byte handling and interrupt handling conventions. For further information on I/O interfaces, see Section 4.



```
; PROGRAMMING EXAMPLES

          1000        .LOC   1000

; INPUT/OUTPUT ROUTINES

; MASTER TERMINAL INPUT - ACCEPTS CHARACTER INTO A0

    1000  63610       SKPDN  TTI       ;IS THERE ANY INPUT AVAILABLE ?
    1001    777       JMP    .-1       ;  NO, KEEP WAITING
    1002  60610       DIAC   0,TTI     ;YES, ACCEPT IT & CLEAR TTI DONE FLAG

;   NOTE: FOR PAPER TAPE READER USE DIAS TO START READING NEXT CHARACTER


; MASTER TERMINAL OUTPUT - ASSUMES OUTPUT CHARACTER IS IN A0

    1003  63511       SKPBZ  TTO       ;TTO STILL BUSY FROM A PREV. OUTPUT ?
    1004    777       JMP    .-1       ;  YES, WAIT FOR IT TO FINISH
    1005  61111       DOAS   0,TTO     ;  NO, OUTPUT THE CHAR. AND START TTO
```

085-44

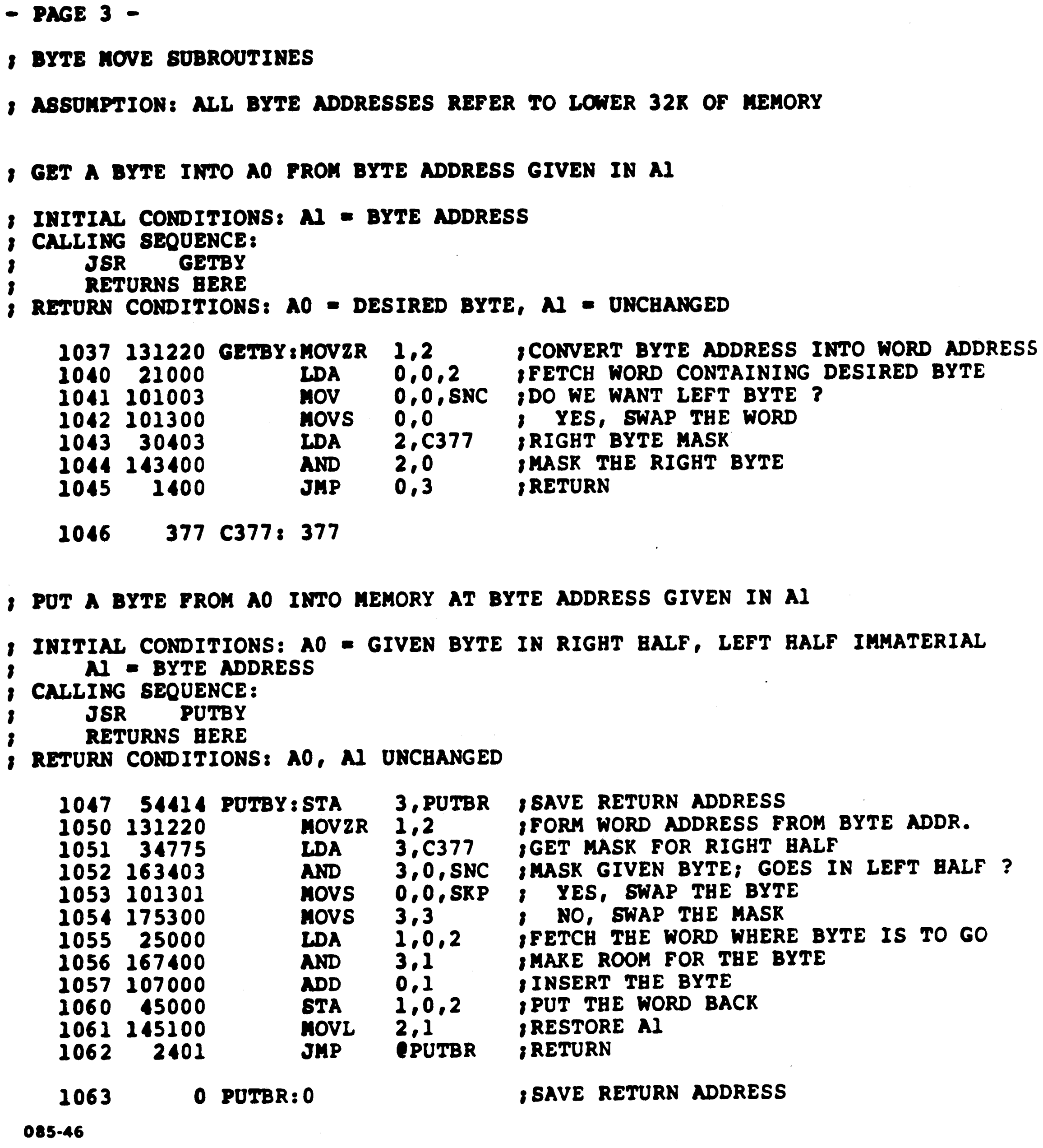
**Figure E-4. Master Terminal I/O Programming Examples (1 of 4)**



```
- PAGE 2 -

; SUBROUTINE TO TYPE ASCII TEXT

; INITIAL CONDITIONS: NONE
; CALLING SEQUENCE:
;     JSR    TYPE
;     (ASCII TEXT,
;     PACKED 2 CHARACTERS/WORD
;     WITH 0 BYTE TERMINATOR)
;     RETURNS HERE
; RETURN CONDITIONS: A0 = 0, A2 = PRESERVED

    1006  25400 TYPE: LDA    1,0,3     ;PICK UP 2 ASCII CHARACTERS
    1007 175420       INCZ   3,3       ;ADV. RETURN PNTR; C=BYTE CNTR
    1010  20411 TYPE2:LDA    0,C377L   ;LEFT-BYTE MASK
    1011 123705       ANDS   1,0,SNR   ;EXTRACT A BYTE - IS IT 0 ?
    1012   1400       JMP    0,3       ;  YES, RETURN TO CALLER
    1013  63511       SKPBZ  TTO       ;  NO, WAIT FOR TTO NOT BUSY
    1014    777       JMP    .-1
    1015  61111       DOAS   0,TTO     ;OUTPUT THE CHARACTER
    1016 125362       MOVCS  1,1,SZC   ;SWAP THE 2 ASCII CHAR.; CK. BYTE CNT
    1017    771       JMP    TYPE2     ;  TYPE THE SECOND CHARACTER
    1020    766       JMP    TYPE      ;  GET 2 MORE CHARACTERS TO TYPE

    1021 177400 C377L:177400           ;OCTAL 377 IN LEFT BYTE




; SUBROUTINE TO TYPE A NUMBER IN OCTAL FORM

; INITIAL CONDITIONS: A1 = NUMBER TO BE TYPED
; CALLING SEQUENCE:
;     JSR    TPOCT
;     RETURNS HERE
; RETURN CONDITIONS: A1 = 0, A2 = PRESERVED, C = 1

    1022  20413 TPOCT:LDA    0,C.BIT
    1023 101120 TPOC2:MOVZL  0,0       ;PRESET CARRY (MSB:1, OTHERS:0)
    1024 125105       MOVL   1,1,SNR   ;LEFT-SHIFT A BIT OUT OF A1 INTO C
    1025   1400       JMP    0,3       ;RETURN WHEN PUSHER BIT IS GONE
    1026 101103       MOVL   0,0,SNC   ;ASSEMBLE ASCII DIGIT; COMPLETE ?
    1027    775       JMP    .-3       ;  NO, GET MORE BITS
    1030  63511       SKPBZ  TTO       ;WAIT FOR TTO NOT BUSY
    1031    777       JMP    .-1
    1032  61111       DOAS   0,TTO     ;OUTPUT THE ASCII DIGIT
    1033  20403       LDA    0,C.OCT
    1034    767       JMP    TPOC2     ;CONTINUE THE LOOP

    1035 140014 C.BIT:140014 ;CONST. TO STRIP OFF 1 BIT & CNVT. TO ASCII
    1036  10003 C.OCT:010003 ;CONST. TO STRIP OFF 3 BITS & CNVT. TO ASCII
```

085-45

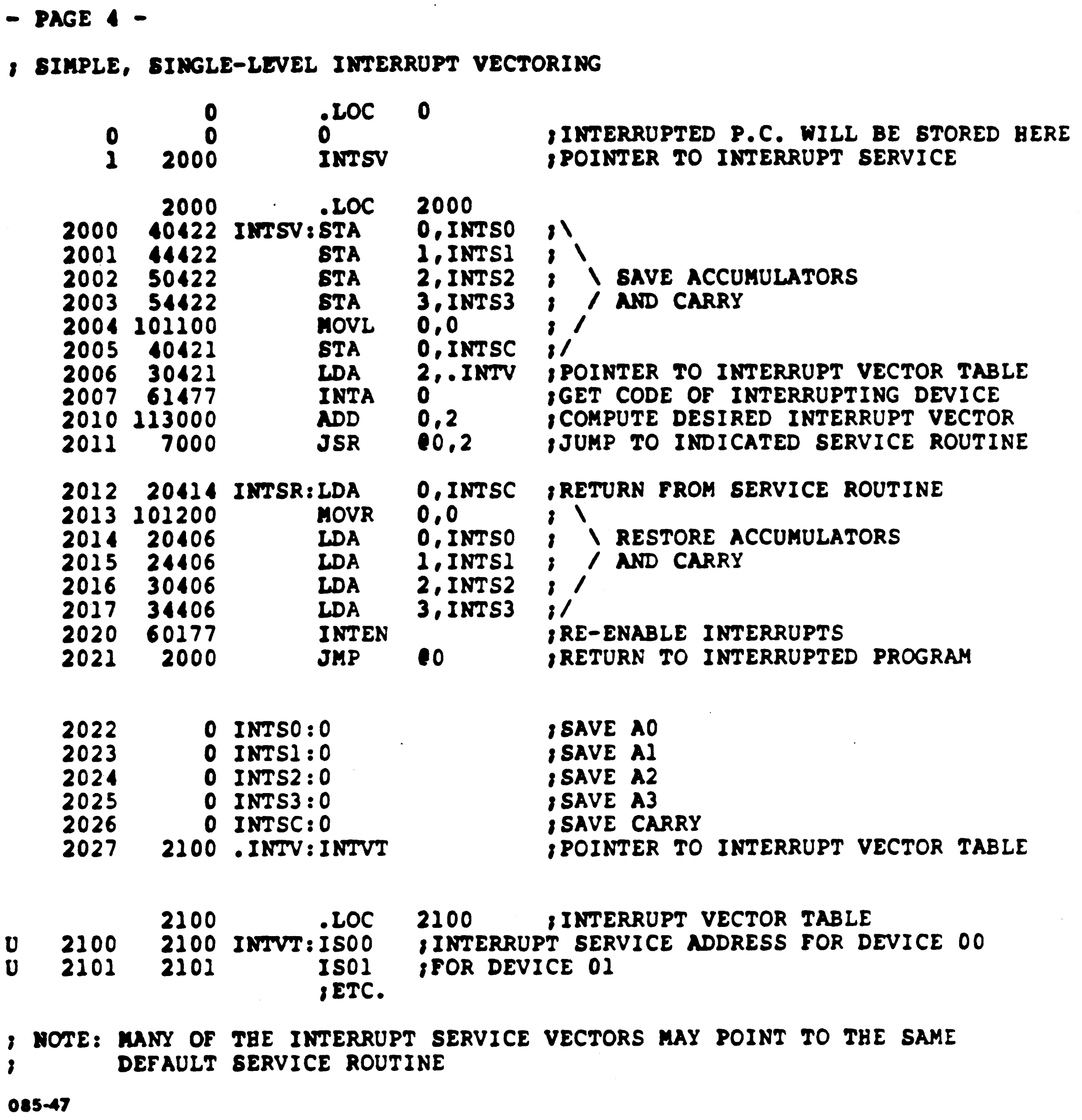
Figure E-4. Master Terminal I/O
Programming Examples (2 of 4)



```
- PAGE 3 -

; BYTE MOVE SUBROUTINES

; ASSUMPTION: ALL BYTE ADDRESSES REFER TO LOWER 32K OF MEMORY


; GET A BYTE INTO A0 FROM BYTE ADDRESS GIVEN IN A1

; INITIAL CONDITIONS: A1 = BYTE ADDRESS
; CALLING SEQUENCE:
;     JSR    GETBY
;     RETURNS HERE
; RETURN CONDITIONS: A0 = DESIRED BYTE, A1 = UNCHANGED

    1037 131220 GETBY:MOVZR  1,2       ;CONVERT BYTE ADDRESS INTO WORD ADDRESS
    1040  21000       LDA    0,0,2     ;FETCH WORD CONTAINING DESIRED BYTE
    1041 101003       MOV    0,0,SNC   ;DO WE WANT LEFT BYTE ?
    1042 101300       MOVS   0,0       ;  YES, SWAP THE WORD
    1043  30403       LDA    2,C377    ;RIGHT BYTE MASK
    1044 143400       AND    2,0       ;MASK THE RIGHT BYTE
    1045   1400       JMP    0,3       ;RETURN

    1046    377 C377: 377


; PUT A BYTE FROM A0 INTO MEMORY AT BYTE ADDRESS GIVEN IN A1

; INITIAL CONDITIONS: A0 = GIVEN BYTE IN RIGHT HALF, LEFT HALF IMMATERIAL
;     A1 = BYTE ADDRESS
; CALLING SEQUENCE:
;     JSR    PUTBY
;     RETURNS HERE
; RETURN CONDITIONS: A0, A1 UNCHANGED

    1047  54414 PUTBY:STA    3,PUTBR   ;SAVE RETURN ADDRESS
    1050 131220       MOVZR  1,2       ;FORM WORD ADDRESS FROM BYTE ADDR.
    1051  34775       LDA    3,C377    ;GET MASK FOR RIGHT HALF
    1052 163403       AND    3,0,SNC   ;MASK GIVEN BYTE; GOES IN LEFT HALF ?
    1053 101301       MOVS   0,0,SKP   ;  YES, SWAP THE BYTE
    1054 175300       MOVS   3,3       ;  NO, SWAP THE MASK
    1055  25000       LDA    1,0,2     ;FETCH THE WORD WHERE BYTE IS TO GO
    1056 167400       AND    3,1       ;MAKE ROOM FOR THE BYTE
    1057 107000       ADD    0,1       ;INSERT THE BYTE
    1060  45000       STA    1,0,2     ;PUT THE WORD BACK
    1061 145100       MOVL   2,1       ;RESTORE A1
    1062   2401       JMP    @PUTBR    ;RETURN

    1063      0 PUTBR:0                ;SAVE RETURN ADDRESS
```

085-46

**Figure E-4. Master Terminal I/O
Programming Examples (3 of 4)**



```
- PAGE 4 -

; SIMPLE, SINGLE-LEVEL INTERRUPT VECTORING

             0        .LOC   0
       0     0        0             ;INTERRUPTED P.C. WILL BE STORED HERE
       1  2000        INTSV         ;POINTER TO INTERRUPT SERVICE

          2000        .LOC   2000
    2000 40422 INTSV:STA    0,INTS0  ;\
    2001 44422       STA    1,INTS1  ; \
    2002 50422       STA    2,INTS2  ;  \ SAVE ACCUMULATORS
    2003 54422       STA    3,INTS3  ;  / AND CARRY
    2004 101100      MOVL   0,0      ; /
    2005 40421       STA    0,INTSC  ;/
    2006 30421       LDA    2,.INTV  ;POINTER TO INTERRUPT VECTOR TABLE
    2007 61477       INTA   0        ;GET CODE OF INTERRUPTING DEVICE
    2010 113000      ADD    0,2      ;COMPUTE DESIRED INTERRUPT VECTOR
    2011  7000       JSR    @0,2     ;JUMP TO INDICATED SERVICE ROUTINE

    2012 20414 INTSR:LDA    0,INTSC  ;RETURN FROM SERVICE ROUTINE
    2013 101200      MOVR   0,0      ; \
    2014 20406       LDA    0,INTS0  ;  \ RESTORE ACCUMULATORS
    2015 24406       LDA    1,INTS1  ;  / AND CARRY
    2016 30406       LDA    2,INTS2  ; /
    2017 34406       LDA    3,INTS3  ;/
    2020 60177       INTEN           ;RE-ENABLE INTERRUPTS
    2021  2000       JMP    @0       ;RETURN TO INTERRUPTED PROGRAM


    2022     0 INTS0:0               ;SAVE A0
    2023     0 INTS1:0               ;SAVE A1
    2024     0 INTS2:0               ;SAVE A2
    2025     0 INTS3:0               ;SAVE A3
    2026     0 INTSC:0               ;SAVE CARRY
    2027  2100 .INTV:INTVT           ;POINTER TO INTERRUPT VECTOR TABLE

          2100       .LOC   2100     ;INTERRUPT VECTOR TABLE
U   2100  2100 INTVT:IS00   ;INTERRUPT SERVICE ADDRESS FOR DEVICE 00
U   2101  2101       IS01   ;FOR DEVICE 01
                     ;ETC.

; NOTE: MANY OF THE INTERRUPT SERVICE VECTORS MAY POINT TO THE SAME
;       DEFAULT SERVICE ROUTINE
```

085-47

**Figure E-4. Master Terminal I/O Programming Examples (4 of 4)**

# Appendix F
# OCTAL VALUES FOR EXTENDED INSTRUCTIONS

This appendix applies to the MARK 5 with EIS and MARK 9 only.

## EXTENDED INSTRUCTIONS

| Instr | Octal Value |
|---|---|
| SLEI | 100011 + byte*100 |
| SGRI | 100031 + byte*100 |
| SEQI | 100051 + byte*100 |
| SNEI | 100071 + byte*100 |
| LDI | 140011 + byte*100 |
| ADI | 140031 + byte*100 |
| SBI | 140051 + byte*100 |
| CLRB | 140471 + @*2000 |
| SETB | 141071 + @*2000 |
| TOGB | 141471 + @*2000 |
| SBZ | 150071 + @*2000 |
| SBO | 154071 + @*2000 |
| SBZC | 150471 + @*2000 |
| SBOC | 154471 + @*2000 |
| SBZS | 151071 + @*2000 |
| SBOS | 155071 + @*2000 |
| SBZT | 151471 + @*2000 |
| SBOT | 155471 + @*2000 |
| SETSP | 142171 + acc*4000 |
| REDSP | 140171 + acc*4000 |
| PUSH | 162171 + acc*4000 |
| POP | 160171 + acc*4000 |
| PUSHJ | 142271 |
| RPUSH | 142671 |
| I2PUS | 143271 |
|  | I3PUS^143671 |
| POPJ | 140271 |
| NYBSL | 164271 |
| NYBSR | 164671 |
| DECAD | 160271 |
| DECSU | 162271 |
| CONVE | 166271 |
| XLOAD | 154271 |
| XSTOR | 156271 |
| VJSR | 140371 + subnum*400 |
| GETBY | 144271 |
| PUTBY | 145271 |
| SGETB | 146671 |
| SPUTB | 145671 |
| XGETB | 150271 |
| XPUTB | 153271 |
| XSGET | 150671 |
| XSPUT | 153671 |
| IGETB | 160371 + disp*400 |
| IPUTB | 170371 + disp*400 |
| XTRAN | 160071 |
| ABINM | 170271 |
| BINDI | 170671 |
| MOVEW | 174271 |
| RMOVW | 176271 |
| TABLS | 172271 |
| TRACE* | 177471 |

*The TRACE instruction is not supported by the MARK 9 CPU.

# Appendix G
# EXECUTION TIMES FOR EXTENDED INSTRUCTIONS

This appendix lists the execution times of instructions on the MARK 5 with EIS option and the MARK 9.

## MARK 5 (WITH EIS) EXTENDED INSTRUCTION EXECUTION TIMES

| Instruction | Execution Time |
|---|---|
| SLEI | 700 ns + 100 ns if Skip occurs |
| SGRI | 700 ns + 100 ns if Skip occurs |
| SEQI | 700 ns |
| SNEI | 700 ns + 100 ns if Skip occurs |
| LDI | 600 ns |
| ADI | 600 ns |
| SBI | 600 ns |
| CLRB | 1500 ns if indirect, + 500 ns |
| SETB | 1400 ns if indirect, + 500 ns |
| TOGB | 1500 ns if indirect, + 500 ns |
| SBZ | 1300 ns if indirect, + 400 ns |
| SBO | 1400 ns if indirect, + 400 ns |
| SBZC | 1600 ns if indirect, + 500 ns |
| SBOC | 1700 ns if indirect, + 500 ns |
| SBZS | 1400 ns if indirect, + 500 ns |
| SBOS | 1500 ns if indirect, + 500 ns |
| SBZT | 1600 ns if indirect, + 500 ns |
| SBOT | 1700 ns if indirect, + 500 ns |
| SETSP | 700 ns |
| REDSP | 700 ns |
| PUSH | 1300 ns |

| Instruction | Execution Time |
|---|---|
| POP | 1400 ns |
| PUSHJ | 1900 ns |
| RPUSH | 1900 ns |
| I2PUS | 1900 ns |
| I3PUS | 1900 ns |
| POPJ | 1600 ns |
| NYBSL | 2600 ns/word + 1100 ns overhead |
| NYBSR | 3100 ns/word + 200 ns overhead |
| DECAD | 900 ns |
| DECSU | 800 ns |
| CONVE | 1600 ns |
| XLOAD | 1400 ns |
| XSTOR | 1400 ns |
| VJSR | 1100 ns |
| GETBY | 1200 ns |
| PUTBY | 1400 ns |
| SGETB | 1200 ns |
| SPUTB | 1400 ns |
| XGETB | 1400 ns |
| XPUTB | 1600 ns |
| XSGET | 1400 ns |
| XSPUT | 1600 ns |
| IGETB | 2400 ns |
| IPUTB | 2600 ns |
| XTRAN | 2100 ns |
| ABINM | 4900 ns |
| BINDI | 15000 ns maximum, 1400 ns minumum |
| MOVEW | 1000 ns/word + 400 ns overhead |
| RMOVW | 1000 ns/word + 400 ns overhead |
| TABLS | 900 ns/word + 400 ns overhead |
| TRACE | 1700 ns |

## MARK 9 EXTENDED INSTRUCTION EXECUTION TIMES

| Instruction | Execution Time |  |
|---|---|---|
| SLEI | 575 ns + 100 ns if Skip occurs |  |
| SGRI | 575 ns + 100 ns if Skip occurs |  |
| SEQI | 575 ns |  |
| SNEI | 575 ns + 100 ns if Skip occurs |  |
| LDI | 525 ns |  |
| ADI | 525 ns |  |
| SBI | 525 ns |  |
| CLRB | 1300 ns | if indirect, + 550 ns |
| SETB | 1200 ns | if indirect, + 550 ns |
| TOGB | 1300 ns | if indirect, + 550 ns |
| SBZ | 1175 ns | if indirect, + 375 ns |
| SBO | 1250 ns | if indirect, + 375 ns |
| SBZC | 1375 ns | if indirect, + 550 ns |
| SBOC | 1475 ns | if indirect, + 550 ns |
| SBZS | 1275 ns | if indirect, + 550 ns |
| SBOS | 1375 ns | if indirect, + 550 ns |
| SBZT | 1375 ns | if indirect, + 550 ns |
| SBOT | 1475 ns | if indirect, + 550 ns |
| SETSP | 525 ns |  |
| REDSP | 525 ns |  |
| PUSH | 1450 ns |  |
| POP | 1675 ns |  |
| PUSHJ | 1850 ns |  |
| RPUSH | 1850 ns |  |
| I2PUS | 1850 ns |  |
| I3PUS | 1850 ns |  |
| POPJ | 1650 ns |  |
| NYBSL | 2400 ns/word + 300 ns overhead |  |
| NYBSR | 2525 ns/word + 100 ns overhead |  |
| DECAD | 725 ns |  |
| DECSU | 525 ns |  |
| CONVE | 678 ns |  |

| Instruction | Execution Time |
|---|---|
| XLOAD | 1275 ns |
| XSTOR | 1175 ns |
| VJSR | 1300 ns |
| GETBY | 1075 ns |
| PUTBY | 1500 ns |
| SGETB | 1075 ns |
| SPUTB | 1500 ns |
| XGETB | 1375 ns |
| XPUTB | 1800 ns |
| XSGET | 1375 ns |
| XSPUT | 1800 ns |
| IGETB | 1700 ns |
| IPUTB | 2325 ns |
| XTRAN | 2150 ns |
| ABINM | 3400 ns |
| BINDI | 8775 ns maximum, 775 ns minumum |
| MOVEW | 1000 ns/word + 275 ns overhead |
| RMOVW | 1000 ns/word + 275 ns overhead |
| TABLS | 875 ns/word + 325 ns overhead |
| * | |

*The TRACE instruction is not supported by the MARK 9 CPU.

# Appendix H
# VON NEUMANN CHART OF EXTENDED INSTRUCTIONS

This appendix applies to the MARK 5 with EIS and MARK 9 only.

KEY:

| | | | |
|---|---|---|---|
| ACCU = | Accumulator | V = | Vector |
| @ = | Indirect | D = | Displacement |
| S = | Sequential | | |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLEI | 1 | 0 | B Y T E | | | | | | | | 0 | 0 | 1 | 0 | 0 | 1 |
| SGRI | 1 | 0 | | | | | | | | | 0 | 1 | 1 | 0 | 0 | 1 |
| SEQI | 1 | 0 | | | | | | | | | 1 | 0 | 1 | 0 | 0 | 1 |
| SNEI | 1 | 0 | | | | | | | | | 1 | 1 | 1 | 0 | 0 | 1 |
| LDI | 1 | 1 | | | | | | | | | 0 | 0 | 1 | 0 | 0 | 1 |
| ADI | 1 | 1 | | | | | | | | | 0 | 1 | 1 | 0 | 0 | 1 |
| SBI | 1 | 1 | | | | | | | | | 1 | 0 | 1 | 0 | 0 | 1 |
| CLRB | 1 | 1 | 0 | 0 | 0 | @ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SETB | 1 | 1 | 0 | 0 | 0 | @ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| TOGB | 1 | 1 | 0 | 0 | 0 | @ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBZ | 1 | 1 | 0 | 1 | 0 | @ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBO | 1 | 1 | 0 | 1 | 1 | @ | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBZC | 1 | 1 | 0 | 1 | 0 | @ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBOC | 1 | 1 | 0 | 1 | 1 | @ | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBZS | 1 | 1 | 0 | 1 | 0 | @ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBOS | 1 | 1 | 0 | 1 | 1 | @ | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBZT | 1 | 1 | 0 | 1 | 0 | @ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SBOT | 1 | 1 | 0 | 1 | 1 | @ | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XTRAN | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| Rsvd for FPU | 1 | 1 | 1 | x | x | x | x | x | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| TRACE* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| REDSP | 1 | 1 | 0 | ACCU | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| SETSP | 1 | 1 | 0 | ACCU | | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| POP | 1 | 1 | 1 | ACCU | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| PUSH | 1 | 1 | 1 | ACCU | | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

*TRACE is not supported by the MARK 9 CPU.

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POPJ | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| PUSHJUMP | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| RPUSHJUMP | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| I2PUSHJUMP | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| I3PUSHJUMP | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| GETBY | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| [S]PUTBY | 1 | 1 | 0 | 0 | 1 | 0 | 1 | S | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| SGETBY | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| X[S]GETBY | 1 | 1 | 0 | 1 | 0 | 0 | 0 | S | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| X[S]PUTBY | 1 | 1 | 0 | 1 | 0 | 1 | 1 | S | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XLOAD | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| XSTORE | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| DECADD | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| DECSUB | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| NYBSL | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| NYBSR | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| CONVERT | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| BINMUL | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| BINDIV | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| MOVEWORDS | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| RMOVWORDS | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| TABLSEARCH | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| VJSR | 1 | 1 | 0 | V | V | V | V | V | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| IGETBY | 1 | 1 | 1 | 0 | D | D | D | D | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| IPUTBY | 1 | 1 | 1 | 1 | D | D | D | D | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

# INDEX

# INDEX

# COMMENT SHEET

MANUAL TITLE POINT 4 MARK 5/9 Computer Reference Manual

PUBLICATION NO. HM-085-0023 REVISION 01

FROM: NAME/COMPANY:

BUSINESS ADDRESS:

CITY/STATE/ZIP:

COMMENTS: Your evaluation of this manual will be appreciated by POINT 4 Data Corporation. Notation of any errors, suggested additions or deletions, or general comments may be made below. Please include page number references where appropriate.

NO POSTAGE NECESSARY IF MAILED IN U.S.A.

POINT 4 Data Corporation
PUBLICATIONS DEPARTMENT
15442 Del Amo Avenue
Tustin, CA 92680

CUT ON THIS LINE

POINT 4 DATA CORPORATION

15442 Del Amo Avenue
Tustin, CA 92680
(714) 259-0777