CHAPTER 4
PDP-8/E PROGRAMMING SYSTEMS

GENERAL

This chapter deals with the concepts required to program the PDP-8/E and identifies the system programs available to the user. Two handbooks, INTRODUCTION TO PROGRAMMING and PROGRAMMING LANGUAGES, provide a more detailed treatment and description of the commonly used programming languages and programming systems.

The chapter is divided into 2 sections. Section 1 provides basic programming guidelines and section 2 identifies the various programming systems and commonly used languages available to the user.

SECTION 1

PDP-8/E PROGRAMMING FUNDAMENTALS

Organization of the standard core memory or any 4096-word field of extended memory is summarized as follows:

Total locations (decimal)	0-4095 or 4096
Total addresses (octal)	0-7777 or 10,000
Number of pages (decimal)	0-31 or 32
Page designations (octal)	0-37 or 40
Number of locations per page (decimal)	0-127 or 128
Addresses within a page (octal)	0-177 or 200

Routines using 128 instructions or less can be written in one page using direct addresses for looping and indirect addresses for data stored in other pages. When planning the location of instructions and data in core memory, the following locations are reserved for special purposes:

Address	Purpose
0 (octal)	Stores the contents of the program counter following a program interrupt.
1 (octal)	Stores the first instruction to be executed following a program interrupt.

10 (octal)—17 (octal) Auto-indexing.

MEMORY ADDRESSING

The programmer has 4096 (decimal) locations which he may address. However, as illustrated in Figure 3-2 of Chapter 3, when an instruction is fetched from memory, only bits 5 through 11 contain the address of the data. Addressing is accomplished using octal notation. Therefore the 4096 possible locations require addresses in octal from 0000 to 7777. This means that a total of 12 bits is required to specify an absolute address. So that all locations may be addressed as efficiently as possible, memory is addressed in terms of pages with a coding scheme that allows easy access to any one of the 10,000 octal locations. The page addressing scheme is illustrated in Figure 4-1 which shows the relationship of the 40 octal pages with the 10,000 octal locations. The programmer is interested in only three pages in memory at any one time:

a. The current page
b. Page 0
c. A location on other than the current page or page 0.

Page 0 is used to store commonly used operands and off-page pointers. For instance, the location of an indirect address used by instructions is usually on Page 0.

ABSOLUTE ADDR. (OCTAL)	CORE MEMORY PAGE (OCTAL)	PAGE ADDR. (OCTAL)
7777 7600	37	177 000
7577 7400	36	177 000
7377 7200	35	177 000
7177 7000	34	177 000
6777 6600	33	177 000
6577 6400	32	177 000
6377 6200	31	177 000
6177 6000	30	177 000
5777 5600	27	177 000
5577 5400	26	177 000
5377 5200	25	177 000
5177 5000	24	177 000
4777 4600	23	177 000
4577 4400	22	177 000
4377 4200	21	177 000
4177 4000	20	177 000
3777 3600	17	177 000
3577 3400	16	177 000
3377 3200	15	177 000
3177 3000	14	177 000
2777 2600	13	177 000
2577 2400	12	177 000
2377 2200	11	177 000
2177 2000	10	177 000
1777 1600	7	177 000
1577 1400	6	177 000
1377 1200	5	177 000
1177 1000	4	177 000
0777 0600	3	177 000
0577 0400	2	177 000
0377 0200	1	177 000
0177 0000	0	177 000

Figure 4-1 Memory Addressing Scheme

The format of the Central Processor (CPMA) Memory Address Register establishing the memory page must first be considered. The MA register is an unequally divided 12-bit register in which the least significant bits of the MA (bits 5-11) are called the page address bits and the most significant bits (bits 0-4) are called the page bits. The 12 bits of the Memory Address are established by the program counter (PC) and are re-established each time the PC loads an absolute address for the next instruction. Bits 0-4 are used to establish the memory page as shown in Figure 4-2. Because the pages to be addressed include pages 0-37 (octal), only five bits are required. The first two bits represent numbers from 0 to 3 and the next three represent numbers from 0 to 7. Because the locations to be addressed on any given page include locations 0-177 (octal), only seven bits are required to specify any one location. Bit 5 represents an octal 1 or 0; bits 6, 7, and 8 represent the second octal digit from 0 to 7; bits 9, 10, and 11 represent the last octal digit from 0 to 7. Thus, on the example shown in Figure 4-2, the MA register is addressing page 5, location 73 (absolute address 1273).

When the user first receives his PDP-8/E, he should assume that it has no information content in its memory. Before he can load instructions into memory, he must first perform the initializing and loading procedures described in Chapter 2. The following discussion assumes that the preliminary procedures have been completed and that the programmer now wants to load into core those instructions which will be called upon after a program has been written. His main concern is to decide in which memory locations he desires to place his instructions and in which locations he wishes to place the corresponding data.

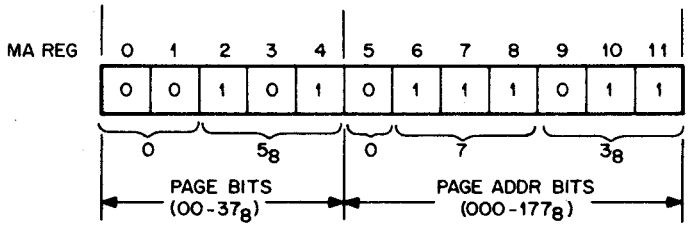

Figure 4-2 Format Establishing Memory Page

Initially, the programmer must load the Central Processor Memory Address Register with an address and then deposit a 12-bit instruction word in the format shown in Figure 4-3.

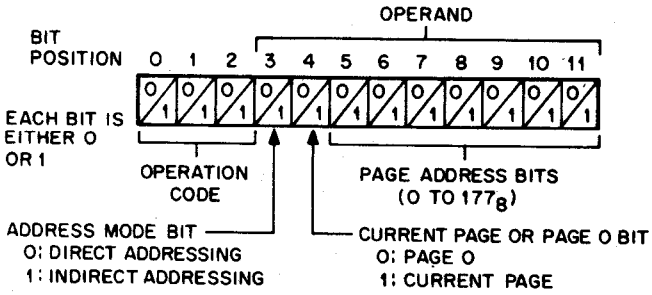

Figure 4-3 Format of a Memory Reference Instruction

The first three bits contain the instruction operation code and have nothing to do with addressing. The last seven bits address the location (from 0 to 177 octal), which will contain either data or a 12-bit address. Those address bits are located in the Memory Buffer Register, and are ineffective until bits 3 and 4 are decoded, at which time the address bits are transferred from the Memory Buffer to the Central Processor Memory Address Register. The page address (the first five bits of the MA register) is determined by whether bit 4 is a 0 or 1 (see Figure 4-4).

Where to place the instruction word or Data word is a very important consideration. At this point, the programmer has three choices in the location of the data:

a. the current page (that page containing the Instruction)
b. page 0
c. a page other than page 0 or the current page.

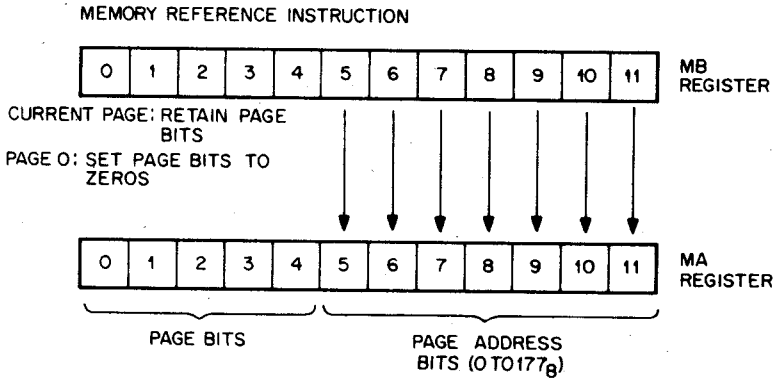

Figure 4-4 Transfer of Address Portion of MRI into MA Register

Current Page—If the programmer desires the data to be located in the current page, he must make bit 4 a 1 in the original instruction word. The logic within the processor causes the first five bits of the MA to remain, and transfers the last seven bits of the MB (the new address within a page) to the last seven bits of the MA Register. This method of updating the MA Register is illustrated in Figure 4-4.

Page 0—Page 0 is commonly used to store operands or address of operands or routines. The programmer must set bit 4 word to 0. The logic within the processor then places all zeros in MA bits 0 through 4 and transfers the content of the last 7 bits of the MB register to the last 7 bits of the MA Register. Thus, the page address is now page 0 and the address within page 0 is some address between 0 and 177 octal. This is illustrated in Figure 4-4.

Addressing A Page Other Than the Current Page or Page 0—The programmer may address a page other than the current page or page 0 by placing a 1 in bit 3 of the original instruction word. As before, the computer then goes to an address on the current page or on page 0, depending on the state of bit 4. The logic within the processor responds to bit 3 being a 1 by going into a defer state for a new address. This procedure is called "Indirect Addressing."

INDIRECT ADDRESSING

In the preceding section, the method of directly addressing 400(octal) memory locations by an MRI was described—namely those on page 0 and those on the current page. This section describes the method for addressing the other 7400(octal) memory locations. Bit 3 of an MRI designates the address mode. When bit 3 is a 0, the operand is a direct address. When bit 3 is a 1, the operand is an indirect address. An indirect address (pointer address) identifies the location that contains the desired address (effective address). To address a location that is not directly addressable, the absolute address of the desired location is stored in one of the 400(octal) directly addressable locations (pointer address); the pointer address is written as the operand of the MRI; and the letter I is written between the mnemonic and the operand. (During assembly, the presence of the I results in bit 3 of the MRI being set to 1.) Upon execution, the MRI will operate on the contents of the location identified by the address contained in the pointer location.

The two examples in Figure 4-5 illustrate the difference between direct addressing and indirect addressing. The first example shows a TAD instruction that uses direct addressing to get data stored on page 0 in location 50; the second is a TAD instruction that uses indirect addressing, with a pointer on page 0 in location 50, to obtain data stored in location 1275. (When references are made to them from various pages, constants and pointer addresses can be stored on page 0 to avoid the necessity of storing them on each applicable page.) The octal value 1050, in the first example, represents direct addressing (bit 3 = 0); the octal value 1450, in the second example, represents indirect addressing (bit 3 = 1). Both examples assume that the accumulator has previously been cleared.