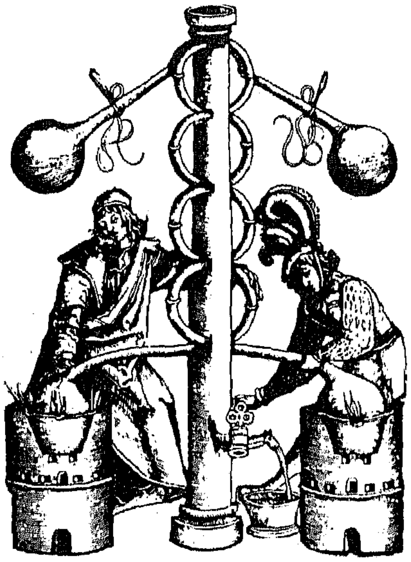
Reprinted with permission from American Laboratory, Vol. 8, No. 11, pp. 114, 116.



Microcomputer interfacing

The MOV and MVI8080 instructions

By Jonathan A. Titus, David G. Larsen, and Peter R. Rony

Mr, Titus is President, Tychon, II/c,. Dr. ROilY. Department oj Chemical Engineering, "nd Mr. Lursell, Department ofChemistry, are with the Virginia Polytechnic Institute IIIId State Vl/il'(l1'-.l'ity.

IN THE October 1976 column, we indicated that seven general-purpose registers (13, C, D, E, H, L, and the accumulator) exist within an 8080 microprocessor chip and that cach register operates on eight bits at a time. These registers are used for many varied purposes, e.g., the storage of an 8-bit constant, an 8-bit timing byte, a 16-bit pointer address, 01' an intermediate result from an arithmetic or logical operation.

Let us cxamine thc data transfer instructions MOV D,S in the 8080 instruction set to sec how the general-purposc registers can be used. We will assume that some data arc initially present within each register. There arc sixty-three different MOV instructions, each of which speci ries both the source register S of thc data and the destination register D to which thc data are moved. For example, to move data from register E to registcr B, you would use the instruction MClV B, E, which is the Intel mnclllonic notation for the operation of moving data to rcgister B from registcr E. Unfortunately, an 8080-bascd microcomputer has no way to understand or interpret a mnemonic instl'llction such as MOV II, E. What is required is the binary representation for MOV n, E: 01000011_2. These eight bits, which comprise the instruction code for MOV n, E, can be manipulated digitally; i.e., they can bc storcd in a semiconductor mcmory device, tmnsmitted over a data bus, storcd in an instruction register within the 8080 chip, and dccoded by an illstruction decoder into a scries of actions that the 8080 performs internally.

How do you convert from the general mnemonic for a move instruction MOV D, S to the specific binary instruction code? Note two si mple ru les: I) the general form of any MOV instruction is,

01	d d d	s s s
MOV class of instructions	J·b!t billary code for des-tination rcgistcr	3·hil binary code J'or source register

and 2) each general-purpose register has associated with it a unique 3-bit codc,

Register	3-bit register code
B	000
C	001
D	010
E	011
H	100
L	101
II	110
accumula-tor (A)	III

ato be discussed later

Once you select the source and destination registers, you insert their respective 3-bit codes into the appropriate places in the general MOV instruction format given above. Some examples are given in *Table 1.* Since binary code is difficult to remember, it is convenient to represent the above 8-bit instruction codes in octal code as shown in *Table*

Table 1

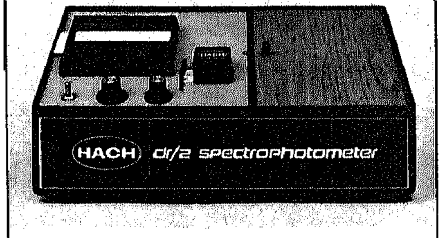
Data transfer operation	Mnemonic	Instruction code
E B	MOVB,E	01000011
H A	MOVA,H	01111100
B C	MOVe,B	01001 000
D L	MOV L,D	01101.010
L → D	MOV D,L	01010 101
E → E	MOV E,E	01011011

Table 2

Data transfer operation	Mnemonic	Octal instruc. tlon code
E B	MOV B,E	103
H A	MOVA,H	174
B C	MOVe,S	110
D L	MOV L,D	152
L → D	MOV D,L	125
E → E	MOV E,E	133

Table 3

Data transfer operation	Mnemonic	Octal instruc_ tion code
<B2> → A	MVIA <B2>	076 <B2>
<B2> → B	MVI8 <82>	006 <82>
<B2> → H	MVI H <82>	046 <82>



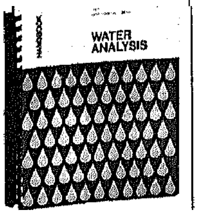


Table 4

Oclal instruction code	Mnemonic	Comments
056	MVI L	Load register L with the following LO address byte
123	123	LO address byte
046	MVIH	Load register H with ·the follOWing HI ad·dress byte
030	030	HI address byte
126	MOV D,M	Move data from the memory location addressed by register pair H, L to register D

2. In each case, you have copied data from one register to another. The destination register contains the copy; *the source register remains unchanged.* Notice that the MOV L, D and MOY D, L instructions have the source and destination registers reversed. In the MOV E, E instruction, you have copied the contents of register E back into register E. The net result is that E remains unchanged. This is a valid 8080 instruction, but it has no visible effect and can be called a "do nothing" instruction. Similar do nothing instructions exist for the other six general-purpose registers.

You may recall that the IN and OUT instructions permit data transfcr to occur between the accumulator (register A) and external I/O devices. The MOV D, S instructions offer the means of temporarily storing input and ou tput data bytes clscwhere within the 8080 chip. You may also inquire how data can be transferred into the general-purpose registers besidcs the IN instruction, There arc generally two ways of doing so:) I'rom the program (for permanent constants) and 2) from memory (for temporary constants, results, data files, etc.).

In order to get data directly from a program into one of the registers, you use *immediate* instructions. These are multibyte instructions that contain thc desircd data within the instruction. The first instruction byte is always the *8080* code; it tells the 8080 chip what to do next. The next one or two instruction bytes contain the actual data. The two-byte *move immediate* instructions,

MV) r
<132>

permit you to move the data con·tained in the sceond byte into the specified register r. The <132> notation means that space must bc left in the program 1'01' the second instruction byte. The general form of the MVI I' instruction is,

00	d d d	I 10
Instruction clas	3-bil code for destination register	

or OD6 in octal code. Some examples are listed in *Tab/e3.*

In order to output the ASCII letter "Q" to output port 6, you would execute the foIlowing program:

Mnemonic	Octal instruction code	
MYIA	076	
321	321	[ASCII "Q"I
OUT	323	
006	006	

As indicated in the program, you firse loael the 8-bit ASCII code into thc accumulator register; having done so, you thcn output the accumulator contents to port 6. Data or constants can be moved into individual rcgisters at any point in an 8080 microcomputer program, and as often as nccded, simply through the use or MVt r instructions. Remember, each MVI r instruction consists or two instruction bytes.

The transfer of data between memory and the general-purpose' registers is more complex since *yOIl 1111/.1'1 clearly and /IIul/nbigliolisly' specify which one of {f possible 65,536 different locations yOIl wish to lise for the transfer opemlion.* This requires a 16-bit address that is stored in the 1:-1, L register pair, register H containing the HI address byte and register L containing the 1.0 address byte. Once you specify these two address bytes, you can readily transfer data back and forth bet ween *the specified lIIem ory location M* and any of the seven gencral-purpose registers. To do so, you use a MOV D. S instruction, wherc the 3·bit code for mcmory location M is 110 (or 6 in octal code). As an cxamplc, if you wish to transfer data from memory location HI = 030 and LO = 123 to registcr D, you executc the program shown in *Table 4.* Remcmber, whencver you perform an 8080 instruction involving mcmory location M as "register M," which has a registcr code of 6_8, you must specify beforehand the absolute memory address of M in register pair H,L.

We shall continue this discussion of the 8080 instruction set in a subscquent column.