Lecture-20

INSTRUCTION SET

As discussed in previous lecture, Intel 8085A μp has 74 basic instructions and 246 variations. These instructions are used for data transformation and data manipulations in the takes place in the μp based system. The instructions of 8085A μp are 1 to 3 bytes in length. The first byte of the instruction is always the opcode of the instruction. The 2nd and 3rd bytes, if available, are the data bytes or the address bytes. The instructions normally involve either one operand or two operands. The position of the operand(s) is(are) identified by the addressing mode. The addressing modes are Register addressing mode, Direct addressing mode, Register indirect addressing mode, Immediate addressing mode and implied addressing mode. Depending on the type of actions taken by the processor, these instructions can be put in different categories.

Classification of Instruction Set:

The 74 instructions available in the 8085A can be divided into five groups depending on their function:

1. **Data Transfer group**: This group comprises instructions that move data between internal registers of μp, between internal register and external memory location and I/O transfer.
2. **Arithmetic group**: Instructions meant for arithmetic operations that add, subtract, increment or decrement data in a register are put in this group.
3. **Logic group**: Instructions that carry out logic operation, such as AND, OR, EX-OR, compare data in the accumulator with another

internal or external register, complement and rotate data in the accumulator are considered in this group.

4. **Branch group**: This group consists of instructions that change the execution sequence of a program, such as conditional and unconditional jumps, subroutine call and return instructions.
5. **Stack Machine Control group:** The instructions used for maintaining the stack and internal control flags are put in this group.

DATA TRANSFER GROUP:

It consists of 15 basic instructions and 86 variations. The basic operation involved is DATA transfer between two register of a microprocessor system. One of the register is always located in the μp itself; the other may be located in one of the following

1) An I/o device
2) Memory
3) The microprocessor

Registers located within the microprocessor are referred to as internal registers. The accessible internal registers are A, B, C, D, E, H, L, SP, PC. The registers located in memory ROM, RWM or I/O devices are referred to as external registers. Therefore, this group includes transfer of data from internal register to another internal register, internal register to memory, memory to internal register, accumulator (A) to output device, or from input device to accumulator.

The register from which data is transferred is the source register and the register to which data is transferred in the destination register. A transfer involves copying the contents of the source register into the destination register; the contents of the source

register are not altered. Each data transfer instruction identifies the source register and the destination register. Identification of one or both of these register may be implied by the instruction in memories or may be explicit. Internal registers are frequently implied; whereas the external registers are usually identified by an explicit address that is part of the instruction. The operation code format for the instructions of this group is shown in fig.5.1.

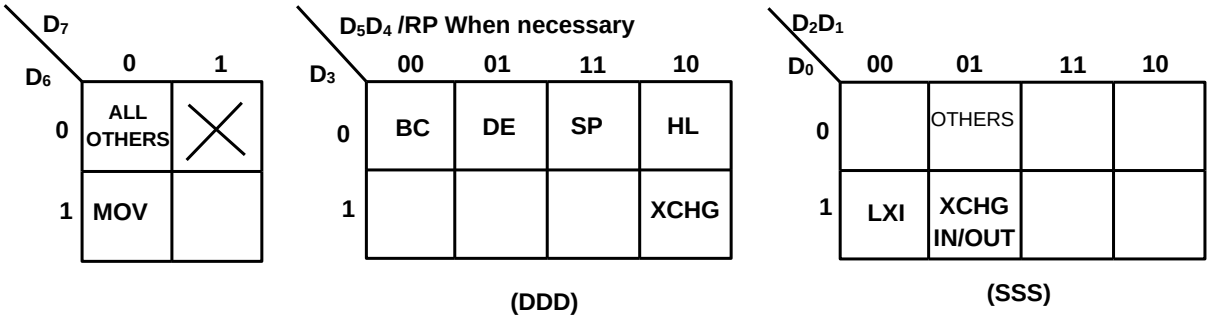


Fig.5.1 Operation Code Format of Data Transfer Group Instructions

1. MOV r_1, r_2: This is an ALP statement. MOV is the mnemonic for move operation. r_2 is the source register and r_1 is the destination register in the operand fields. The meaning of the instruction is "Move the contents of the register r_2 into r_1". The content of register r_2 is not destroyed. Content of r_1 is destroyed and new value form r_2 takes it place. The macro RTL implemented is

$$(r_1) \longleftarrow (r_2)$$

This is a single byte instruction at memory location N. The opcode of the instruction will be

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
0	1	D	D	D	S	S	S	N

The instruction has 49 variations (7x7), seven combinations for source registers (SSS) and seven combinations for destination registers (DDD) other than 110. It has register addressing mode.

How the execution of this instruction takes place? During the T_1 state of the first machine cycle (OFMC), the contents of the program counter are placed on the address bus (A_{15} ...A_8) and address/data bus (AD_7- AD_0). Since address may be low or high, it is customary to use the double sided waveforms, the high byte of the program counter (PCH) goes to the address bus and the low byte (PCL) to the address/data bus.

The ALE signal initially goes high, then midway through the T_1 state, ALE goes low. It is this falling edge that latches the address in to the external latch. Also IO/$\overline{M}$ goes low near the beginning of the T_1 state. This enables the peripheral chips for a memory operation rather than an I/O operation.

During the T_2 state, the program counter is incremented. The address disappears from the address/data bus at the beginning of the T_2 state. This is necessary because an instruction fetch is in progress and the address/data bus is required. The dashed line an AD_7-AD_0 wave from means that the data on the bus is invalid. Towards the end of the T_2 state, the opcode of the instruction appears on the address/data bus. The precise time when opcode appears depends on the memory access time; the length of the buses and other factors.

At the beginning of the T_2 state $\overline{RD}$ goes low and stays low until the middle of the T_3 state. During the T_3 state, the opcode available on the AD bus is copied into the instruction register (IR). During the T_4

state, the instruction is decoded. The μp now knows that the instruction is MOV r_1, r_2. It is represented as MOV r_1, r_2=1 and is executed. After the fetching operation of this instruction, instruction pointer goes back to T_1 state & PC points to next address.

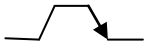
In fact the execution does not take place instantaneously. When the instruction is decoded, first the contents of register r_2 are copied into the temporary register and then contents of the temporary register are copied into register r_1. This operation takes place in the next two states T_1 and T_2. This completes the execution of the MOV instruction. Since in these two extra states during MC-2 only the internal bus is used, address bus and address/data bus are not used, therefore, there buses can be used for some other purpose to save execution time.

One way to save processing time is by starting the next instruction OFMC during the second machine cycle MC-2 of the move instruction. This is called Fetch Execute Overlap (FEO). During T_1 of this machine cycle, the contents of the program counter are outputted to address bus & address/data bus. During T_2 state the next instruction opcode is transferred to address/data bus form memory externally and simultaneously MOV r_1, r_2 instruction is completed internally. Hence, during the MC-2 of the move instruction cycle, MC–1 of the next cycle is operative.

Thus the instruction cycle for MOV r_1,r_2 takes only 4 clock periods (& not six). After the execution PC points to next address. The micro RTL flow for the instruction execution is shown below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

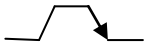
T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MOV r_1, r_2= 1.

Machine Cycle- 2 or (Machine Cycle-1 of next instruction):

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = 

FEO [(Z) ⟵ (r_2)]

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

FEO [(r_1) ⟵ (Z)]

The data flow and the timing waveforms during the execution of this instruction are shown in fig.5.2a and fig.5.2b respectively:

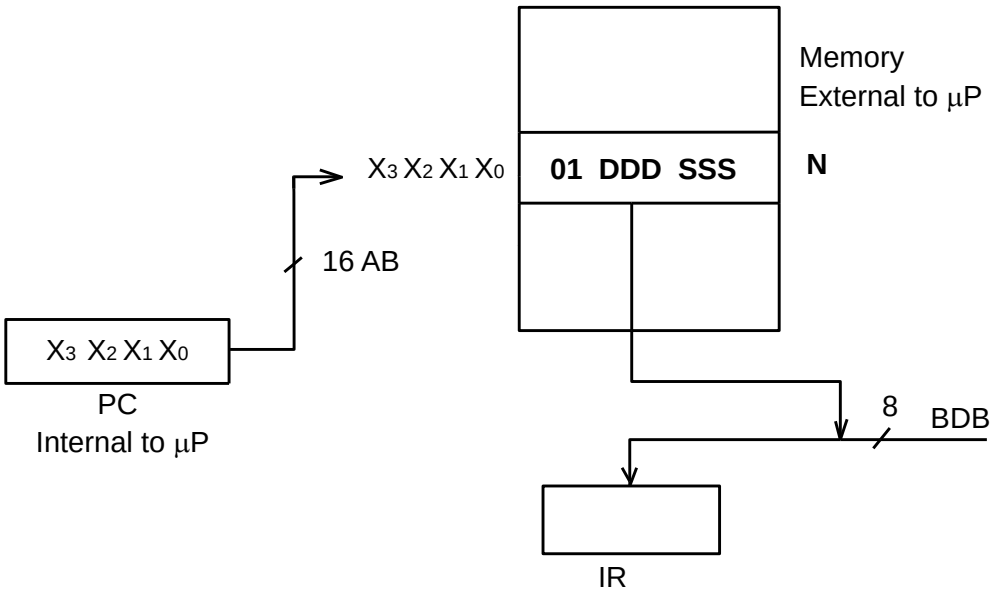


Fig.5.2a Data Flow During the Execution of MOV r_1,r_2 Instruction

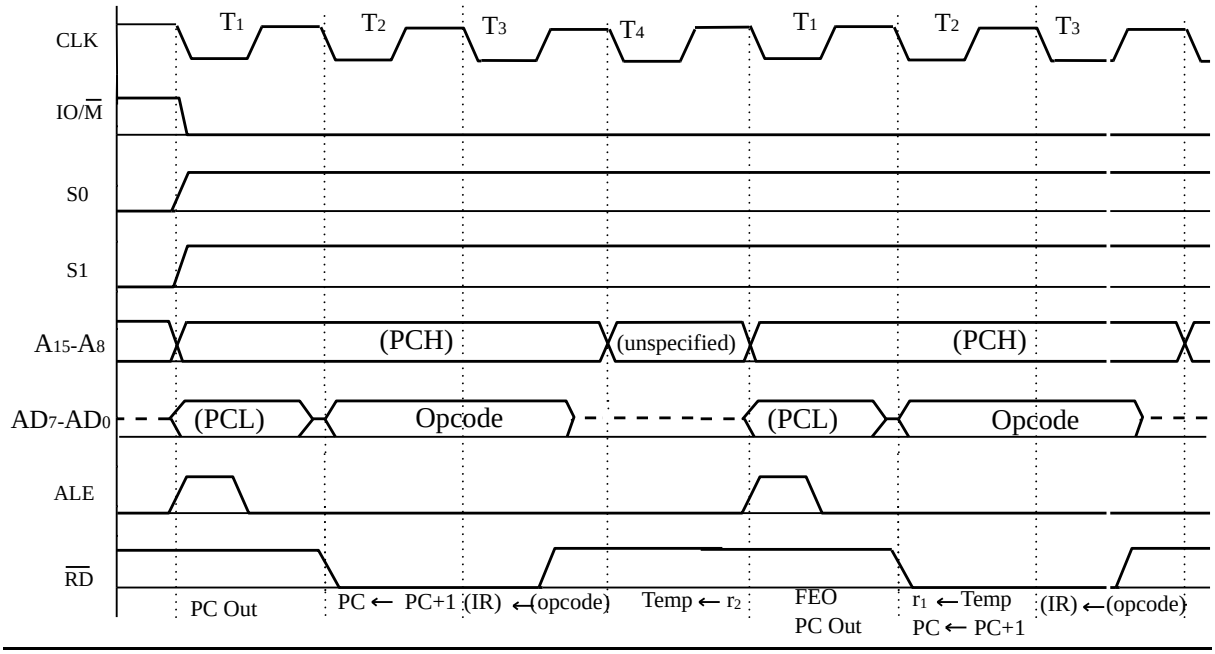


Fig.5.2b Timing Waveforms During the Execution of MOV r_1,r_2 Instruction

2. MOV r, M: This is an ALP statement. The meaning of this instruction is "Move the content of the memory location whose address is available into (H,L) pair into the internal general purpose register r". The macro RTL implemented is

(r) ⟵ M(H,L)

M(H,L) is the source register, (r) is the destination register. It is a single byte instruction at memory location N. The operation code is,

0	1	D	D	D	1	1	0	N

It has seven variations. DDD cannot be 110 because direct memory to memory data transfer is not allowed in this processor. If memory to memory data transfer is required, it has to be routed through one of the internal register. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MOV r, M= 1.

Machine Cycle- 2:

MRMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=0

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = ⎍

T_2: $\overline{RD}$ = 0, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (r) ⟵ AD_7-AD_0

This Operation requires only two machine cycles, OFMC & MRMC, and total 7 states. It requires 3.5µsec using 2 MHz clock.

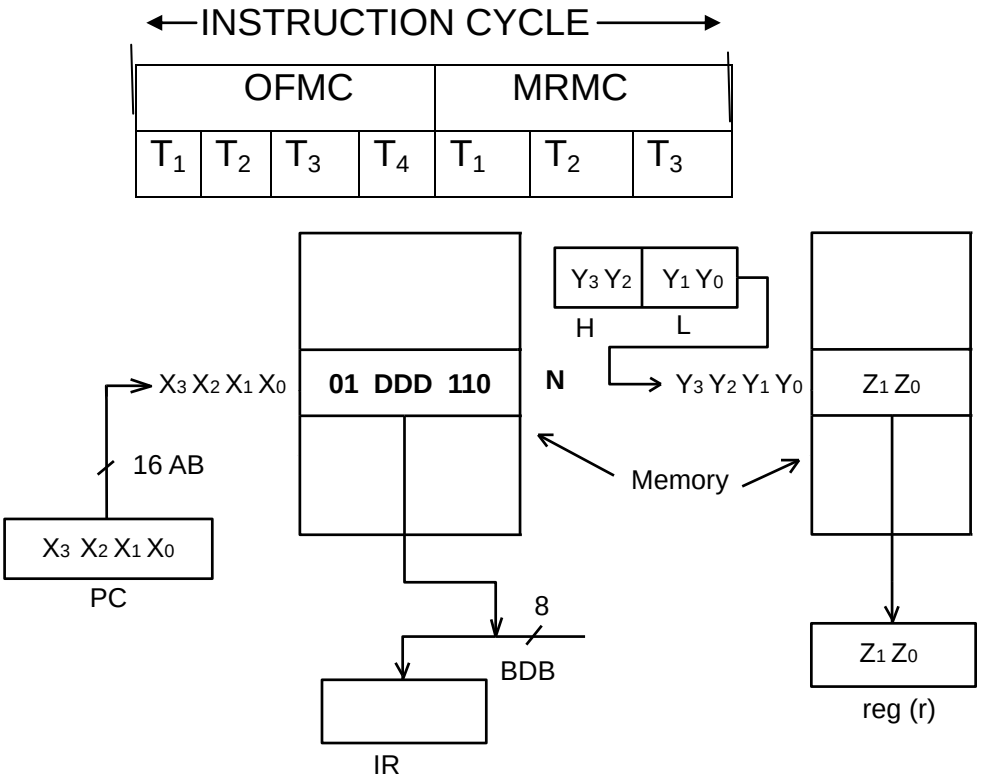


Fig.5.3 Data Flow during the Execution of MOV r, M Instruction

The addressing mode is register indirect addressing mode. Fig.5.3 illustrates the operation cycle.

3. MOV M, r: This is an ALP statement. The meaning of the instruction is "Move the content of the internal general purpose register r into the memory location whose address is available in (H,L) register pair". The macro RTL implemented is

$$M(H,L) \longleftarrow (r)$$

It is a single byte instruction. The operation code is

0	1	1	1	0	S	S	S	N

It has seven variations. SSS cannot be 110 because direct memory to memory data transfer is not allowed. The micro RTL flow is given below:

Machine Cycle- 1:

OFMC: Status signals IO/$\overline{M}$=0, S_1=1, S_0=1

T_1: A_{15}-A_8 ⟵ (PCH), AD_7-AD_0 ⟵ (PCL), ALE = ⎍

T_2: $\overline{RD}$ = 0, (PC) ⟵ (PC) +1, AD_7-AD_0 ⟵ M(AB)

T_3: $\overline{RD}$ = 1, ↑ , (IR) ⟵ AD_7-AD_0

T_4: μp decodes the opcode. MOV M, r = 1.

Machine Cycle- 2:

MWRMC: Status signals IO/$\overline{M}$=0, S_1=0, S_0=1

T_1: A_{15}-A_8 ⟵ (H), AD_7-AD_0 ⟵ (L), ALE = ⎍

T_2: $\overline{WR}$ = 0, AD_7-AD_0 ⟵ (r)

T_3: $\overline{WR}$ = 1, ↑, M(AB) ⟵ AD_7-AD_0

The operation requires only two machine cycles- OFMC & MWRMC and total 7 states. It requires 3.5 µsec using 2 MHZ internal clock. The addressing mode is register indirect addressing mode. Fig.5.4 illustrates the operation cycle.

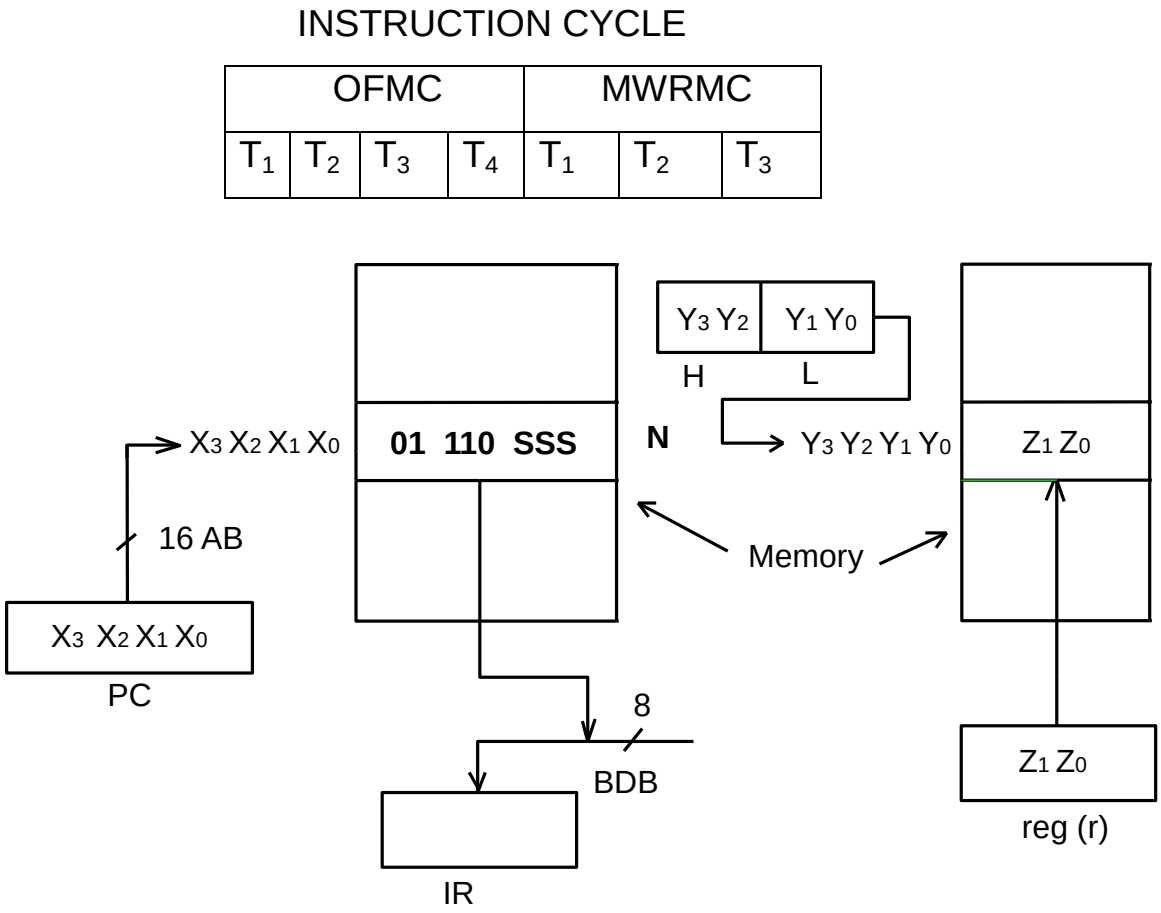


Fig.5.4 Data Flow during the Execution of MOV M, r Instruction