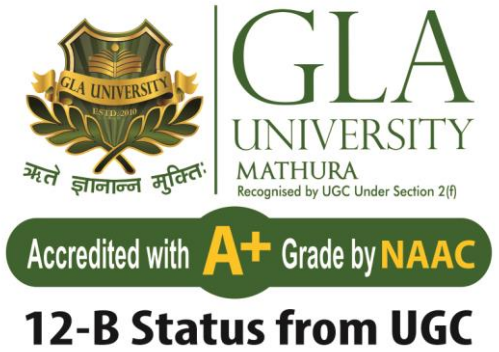
4th PPT

Instruction Set
&
Addressing Modes



Dr Dhirendra Kumar
Asst Prof, ECE department, GLA University, Mathura

Outline

- Instruction Set of 8085
- Data Transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Machine Control Instructions

Data Transfer Instructions

1. MOV R_d, R_s

$R_s \rightarrow$ A, B, C, D, E, H&L

$R_d \rightarrow$ A, B, C, D, E, H & L

i. Copy the data of source register R_s to the destination register R_d

ii. 1-byte instruction

iii. Register addressing mode

iv. One machine cycle (OP) & 4T

v. No flag is affected

Note: *No flag is affected by data transfer instructions*

Note: *OP : Op Code Fetch, MW : Memory write , MR : Memory read, IOR : I/O read, IOW: I/O write.*

2. MOV M, R

R → A, B, C, D, E, H & L

M → Memory address stored in HL pair

i. Copy the data of register R to the memory location whose address is stored in HL pair

ii. 1-byte instruction

iii. Register indirect addressing mode

---

iv. Two machine cycles (OP+MW) & 7T-states.

v. No flag is affected

3. MOV R, M

R→ A, B, C, D, E, H & L

M → Memory address stored in HL pair

i. Copy the data of memory location whose address is stored in HL pair to the register R

ii. 1-byte instruction

iii. Register indirect addressing mode

iv. Two machine cycles (OP+MR) & 7T-states

v. No flag is affected

4. **MVI R, 8 bit data**
   R → A, B, C, D, E, H & L

   i. Copy the 8 bit data immediately to the register R
   ii. 2-byte instruction
   iii. Immediate addressing mode
   iv. Two machine cycles (OP+MR) & 7T-states
   v. No flag is affected

5. **MVI M, 8 bit data**
   M → Memory address stored in HL pair

   i. Copy the 8 bit data immediately to the memory location whose address is stored in HL pair
   ii. 2-byte instruction.
   iii. Immediate-indirect addressing mode.
   iv. Three machine cycles (OP+MR+MW)&10 T-states
   v. No flag is affected.

6. **LXI R_p, 16 bit data**
   R_p→BC, DE, HL & SP

   i. Load the 16 bit data immediately in register pair,R_p
   ii. 3-byte instruction
   iii. Immediate addressing mode
   iv. Three machine cycles (OP+MR+MR)& 10 T-states
   v. No flag is affected .

Note : *While storing instruction in memory, the lower byte of 16 bit is stored at lower address and higher order byte is stored at higher address and at the time of execution the lower order byte stored at lower address is first read into lower order register of register pair R_p and higher order byte stored at higher address is read into higher order register.*

7. LDA 16 bit address

i. Load accumulator directly with the contents of given 16 bit address.
ii. 3-byte instruction
iii. Direct addressing mode
iv. Four machine cycles (OP+MR+MR+MR) & 13 T-states
v. No flag is affected

8. STA 16 bit address

i. Store the contents of 'A' at given 16 bit address
ii. 3-byte instruction
iii. Direct addressing mode
iv. Four machine cycles (OP+MR+MR+MW) & 13 T-states
v. No flag is affected

9. LDAX R_P
$R_P \rightarrow$ BC & DE

i. Load accumulator with contents of memory location whose address is stored in register R_P
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7 T-states.
v. No flag is affected.

10. STAX R_P
$R_P \rightarrow$ BC & DE

i. Store the contents of accumulator at memory location whose address is stored in register pair R_P
ii. 1 byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MW) & 7 T-states.
v. No flag is affected.

11. LHLD 16 bit address

i. Load HL pair with the contents of given address and next address. The contents of given address are moved to L and contents of next address are moved to H.
ii. 3 byte instruction
iii. Direct addressing mode
iv. Five machine cycles (OP+MR+MR+MR+MR) & 16 T-states
v. No flag is affected

12. SHLD 16 bit address

i. Store HL pair at given 16 bit address and next address. The contents of L are moved to given and contents of H are moved to next address.
ii. 3 byte instruction
iii. Direct addressing mode
iv. Five machine cycles (OP+MR+MR+MW+MW) & 16 T-states
v. No flag is affected

13. SPHL

i. The contents of HL pair are copied in stack pointer
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 6 T-states
v. No flag is affected

14. PCHL

i. The contents of HL Pair are copied in program counter.
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 6 T-states
v. No flag is affected

15. XCHG

i. Exchange the contents HL pair with DE pair.
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. No flag is affected

16. IN 8 bit port address

i. Copy the contents of I/O mapped I/O device connected at given 8 bit port address directly to accumulator .
ii. 2-byte instruction
iii. Direct addressing mode
iv. Three machine cycles (OP+MR+IOR) & 10 T-states
v. No flag is affected

17. OUT 8 bit port address

i. Copy the contents of accumulator directly to I/O mapped I/O device connected at given 8 bit port address
ii. 2-byte instruction
iii. Direct addressing mode
iv. Three machine cycles (OP+MR+IOW) & 10 T-states
v. No flag is affected

Ex Consider the execution of the following instructions by a 8085 microprocessor :

LXI H, $01FF_H$

SHLD 2050_H

What are the contents of memory locations 2050_H and 2051_H and the registers H and L after execution of above instructions.

IES(EE,02)

Solution:

LXI H, $01FF_H$[Load H = 01_H, L = FF_H]

SHLD, 2050_H

After the execution of SHLD instruction, the contents of L are stored in 2050 and the contents of H are stored in next memory location i.e. 2051. The content of H and L are not altered

2050_H = FF

2051_H = 01

H = 01

L = FF

Note: *The contents of lower order register are always stored at lower address and higher order register at higher address.*

Arithmetic Instructions

1. ADD R

R→A, B, C, D, E, H & L

i. Add the contents of R to A and store the result in A
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected

2. ADD M

M → Memory address stored in HL pair

i. Add the contents of memory location, whose address is stored in HL pair to A & store result in A
ii. 1-byte instruction
iii. Register indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

3. ADI 8bit data

i. Add the 8 bit data immediately to A and store the result in A
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

4. ADC R

R→A, B, C, D, E, H & L

i. Add the contents of R along with carry flag to contents of A and store the result in A
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected

5. ADC M

M → Memory address stored in HL pair

i. Add the contents of M along with carry flag to contents of A and store the result in A
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

6. ACI 8 bit data

i. Add the 8 bit data along with carry flag to contents of A & store result in A
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

7. DAD R_P

R_p→ BC, DE, SP & HL

i. Add the contents of register pair R_p to contents of HL pair and store the result in HL pair.
ii. 1-byte instruction
iii. Register addressing mode
iv. Three machine cycles & 10 T-states
v. Only CY flag is set if result is more than 16 bits.

No other flag is affected.

8. DAA

i. Decimal Adjust Accumulator. Binary contents of accumulator are changed to its two digit BCD equivalent
ii. 1-byte instruction
iii. Implicit or Implied addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected

Note: *DAA instruction changes the binary values of contents of accumulator to its two digit equivalent BCD number. This instruction makes use of status of CY and AC flags. It performs the operation as follows,*

I. If value of lower order four bits (D_3-D_0) in the accumulator is greater than 9 or if AC flag is set , the instruction adds 0110 to low order four bits.

II. If value of higher order four bits (D_7-D_4) in the accumulator is greater than 9 or if CY flag is set , the instruction adds 0110 to higher order four bits.

Note: *DAA is used for BCD addition only not for BCD subtraction.*

9. SUB R

R→A, B, C, D, E, H & L

i. Subtract contents of R from contents of A and store result in A
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected

10. SUB M

M → Memory address stored in HL pair

i. Subtract contents of M from A and store result in A
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

11. SUI 8bit data

i. Subtract the 8 bit data immediately from contents of A and store the result in A
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

12. SBB R

R→A, B, C, D, E, H & L

i. Subtract the contents of R along with carry flag from contents of A and store the result in A
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected

13. SBB M

M → Memory address stored in HL pair

i. Subtract the contents of M along with carry flag from A and store the result in A
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

14. SBI 8 bit data

i. Subtract the 8 bit data along with carry flag from contents of A & store result in A
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected

15. INR R

R→A, B, C, D, E, H & L

i. Increment contents of R by 1
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected except CY flag

16. INR M

M → Memory address stored in HL pair

i. Increment contents of M by 1
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Three machine cycles (OP+MR+MW) & 10 T-states
v. All flags are affected except CY flag

17. DCR R

R→A, B, C, D, E, H & L

i. Decrement contents of R by 1
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected except CY flag

18. DCR M

M → Memory address stored in HL pair

i. Decrement contents of M by 1
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Three machine cycles (OP+MR+MW) & 10 T-states
v. All flags are affected except CY flag

19. INX R_P

$R_p \rightarrow$BC, DE, HL & SP

i. Increment contents of register pair R_p by 1
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 6 T-states
v. No flag is affected

20. DCX R_P

$R_p \rightarrow$BC, DE, HL & SP

i. Decrement contents of register pair R_p by 1
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 6 T-states
v. No flag is affected

Ex.6.5 An 8085 executes the following instructions.

```
2710  LXI    H, 30A0H
2713  DAD  H
2714  PCHL
```

All addresses and constants are in Hex. What are contents of PC and HL just after executing PCHL?

GATE(EC,08)

Solution:

2710 LXI H, 30A0H ; Loads HL pair with 30A0 H. So, H = 30 H and L = A0 H

2713 DAD H ; Adds the contents of HL pair(i.e. 30A0H) to HL pair (i.e. 30A0H) and store result in HL pair.

HL : 0011 0000 1010 0000

+HL : 0011 0000 1010 0000

HL : 0110 0001 0100 0000

6 1 4 0

After addition, HL = 6140 H

2714 PCHL ; Moves the contents of HL (i.e 6140H) pair to Program Counter. So, PC = 6140 H.

So, after execution of PCHL , HL = 6140 H and PC = 6140 H.

Ex6.6 The following instructions have been executed by an 8085 microprocessor,

ADDRESS (HEX)	INSTRUCTION
6010	LXI H, $8A79_H$
6013	MOV A, L
6014	ADD H
6015	DAA
6016	MOV H, A
6017	PCHL

What will be address of the next instruction be fetched?

GATE(EC,97)

Solution :

6010 LXI H, 8A79 H ; Loads HL pair with 8A79H . With H = 8AH and L = 79H

6013 MOV A, L ; Moves the contents of L (i.e. 79H) to A. So, A = 79H

6014 ADD H ; Adds the content of H(i.e. 8AH) to contents of A and store the result in A.

79H=0111 1001

8AH=1000 1010

1111

1 0000 0011

CY 0 3H

6015 DAA ; As both AC and CY flags are set during execution of ADDH, So, DAA, adds 0110H to lower order bits and 0110H to higher order bits to A to adjust binary result to Binary coded decimal.

03H=0000 0011

0110 0110

0110 1001=69H

So, A = 69H

6016 MOV H, A ; Moves the contents of A (i.e. 69H) to H.So, H = 69H

6017 PCHL ; Copies the contents of HL pair (i.e. H = 69H and L = 79H) to program counter. So, PC = 6979H

After the execution of PCHL the program counter has 6979H . So, the address of next instruction to be fetched will be 6979H.

Logical Instructions

1. **ANA R**
   R→A, B, C, D, E, H & L

   i. Contents of R are ANDed with contents of A and result is stored in A
   ii. 1-byte instruction
   iii. Register addressing mode
   iv. One machine cycle (OP) & 4 T-states
   v. All flags are affected. CY flag is reset and AC flag is set.

Note: *ANA A does not affect the contents of accumulator but resets the carry flag and sets auxiliary carry flag.*

2. **ANA M**
   M → Memory address stored in HL pair

   i. Contents of M are ANDed with contents of A and result is stored in A
   ii. 1-byte instruction
   iii. Indirect addressing mode
   iv. Two machine cycles (OP+MR) & 7T-states
   v. All flags are affected. CY flag is reset and AC flag is set.

3. **ANI 8bit data**

   i. 8 bit data is ANDed with contents of A and result is stored in A
   ii. 2-byte instruction
   iii. Immediate addressing mode
   iv. Two machine cycles (OP+MR) & 7T-states
   v. All flags are affected. CY flag is reset and AC flag is set.

4. **ORA R**
   R→A, B, C, D, E, H & L

   i. Contents of R are ORed with contents of A and result is stored in A
   ii. 1-byte instruction
   iii. Register addressing mode
   iv. One machine cycle (OP) & 4 T-states
   v. All flags are affected. Both CY and AC flag are reset.

Note: *ORA A does not affect the contents of accumulator but resets the carry and auxillary carry flags.*

5. ORA M

M → Memory address stored in HL pair

i. Contents of M are ORed with contents of A and result is stored in A
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected. Both CY and AC flag are reset.

6. ORI 8bit data

i. 8 bit data is ORed with contents of A and result is stored in A
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected. Both CY and AC flag are reset.

7. XRA R

R→A, B, C, D, E, H & L

i. Contents of R are Ex-ORed with contents of A and result is stored in A
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected. Both CY and AC flag are reset.

Note: *XRA A clears the contents of accumulator but resets the carry and auxiliary carry flags.*

8. XRA M

i. Contents of M are Ex-ORed with contents of A and result is stored in A
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected. Both CY and AC flag are reset.

9. XRI 8bit data

i. 8 bit data is Ex-ORed with contents of A and result is stored in A
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected. Both CY and AC flag are reset.

10. CMA

i. Compliment the contents of Accumulator
ii. 1-byte instruction
iii. Implicit or Implied addressing mode
iv. One machine cycle (OP) & 4 T-states
v. No flag is affected

Note: *CMA performs the logical NOT operation.*

11. CMC

i. Compliment the Carry flag
ii. 1-byte instruction
iii. No addressing mode
iv. One machine cycle (OP) & 4 T-states
v. CY flag is complemented. No other flag is affected.

12. STC

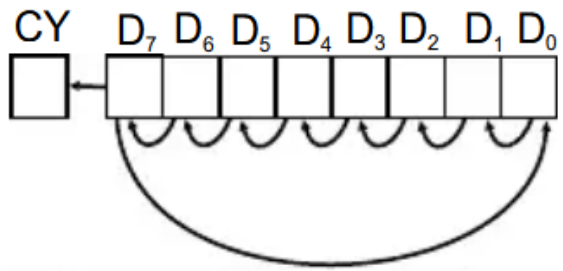
i. Set the Carry flag
ii. 1-byte instruction
iii. No addressing mode
iv. One machine cycle (OP) & 4 T-states
v. CY flag is set. No other flag is affected.

13. RLC

i. Rotate the contents of Accumulator to left without Carry(CY) flag. The contents of bit D_7 are shifted to CY flag as well as D_0 bit.

13. RLC

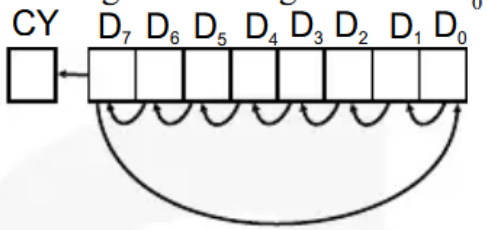
i. Rotate the contents of Accumulator to left without Carry(CY) flag. The contents of bit D_7 are shifted to CY flag as well as D_0 bit.



ii. 1-byte instruction
iii. Implicit or Implied addressing mode
iv. One machine cycle (OP) & 4 T-states
v. Only CY flag is affected.

14. RAL

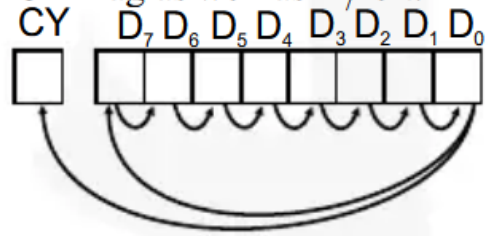
i. Rotate the contents of Accumulator to left through Carry flag. The contents of bit D_7 are shifted to CY flag and CY flag shifted to D_0.



ii. 1-byte instruction
iii. Implicit or Implied addressing mode
iv. One machine cycle (OP) & 4 T-states
v. Only CY flag is affected.

15. RRC

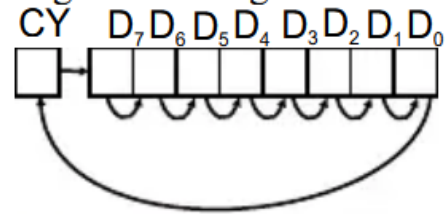
i. Rotate the contents of Accumulator to right without Carry flag. The contents of bit D_0 are shifted to CY flag as well as D_7 bit.



ii. 1-byte instruction
iii. Implicit or Implied addressing mode
iv. One machine cycle (OP) & 4 T-states
v. Only CY flag is affected.

16. RAR

i. Rotate the contents of Accumulator to right through Carry flag. The contents of bit D_0 are shifted to CY flag and CY flag shifted to D_7.



ii. 1-byte instruction
iii. Implicit or Implied addressing mode
iv. One machine cycle (OP) & 4 T-states
v. Only CY flag is affected.

17. CMP R

i. Compare the contents of R with contents of A.
ii. 1-byte instruction
iii. Register addressing mode
iv. One machine cycle (OP) & 4 T-states
v. All flags are affected.

Note: *Comparison is made by subtracting the contents of register R from contents of Accumulator. The contents of A and R remain unaltered after comparison. The result of operation is reflected by status of flags as under,*

If $A - R > 0$ *then* $S = 0 ; Z = 0 ; CY = 0.$
If $A - R = 0$ *then* $S = 0 ; Z = 1 ; CY = 0.$
If $A - R < 0$ *then* $S = 1 ; Z = 0 ; CY = 1.$

18. CMP M

i. Compare the contents of M with contents of A.
ii. 1-byte instruction
iii. Indirect addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected.

Note: *Comparison is made by subtracting the contents of memory location M from contents of Accumulator. The contents of A and M remain unaltered after comparison. The result of operation is reflected by status of flags as under,*

If $A - M > 0$ *then* $S = 0 ; Z = 0 ; CY = 0.$
If $A - M = 0$ *then* $S = 0 ; Z = 1 ; CY = 0.$
If $A - M < 0$ *then* $S = 1 ; Z = 0 ; CY = 1.$

19. CPI 8-bit data

i. Compare the 8-bit data with contents of A.
ii. 2-byte instruction
iii. Immediate addressing mode
iv. Two machine cycles (OP+MR) & 7T-states
v. All flags are affected.

Note: *Comparison is made by subtracting the 8-bit data from contents of Accumulator. The contents of A remain unaltered after comparison. The result of operation is reflected by status of flags as under,*

If A – (8-bit data) > 0 then S = 0 ; Z = 0 ; CY = 0.
If A – (8-bit data) = 0 then S = 0 ; Z = 1 ; CY = 0.
If A – (8-bit data) < 0 then S = 1 ; Z = 0 ; CY = 1.

Ex.6.9 An 8085 assembly language program is given below.

```
Line  1 : MVI A, B5H
      2 : MVI B, 0EH
      3 : XRI 69H
      4 : ADD B
      5 : NI 9BH
      6 : CPI 9FH
      7 : STA 3010H
      8 : HLT
```

What is the status of the CY and Z flags after execution of line 7 of the program?

GATE(EC,07)

Solution:

Line 1 : MVI A, B5H ; Moves B5 H to accumulator, A. So , A = B5H

2 : MVI B, 0EH ; Moves 0E H to B. So, B = 0EH

3 : XRI 69H ; Contents of A are ExORed with 69H and result is stored in A.

A : 1011 0101 (B5H)

69H : 0110 1001

1101 1100 = DCH

So, A = DCH

4: ADD B ; Adds contents of B to contents of A and result is stored in A.

A : 1101 1100

B : 0000 1110

1110 1010 = EAH

So, A = EAH

5: ANI 9BH ; Contents of A are ANDed with 9BH and result is stored in A.

A : 1110 1010

9BH : 1001 1011

1000 1010 = 8AH

So, A = 8AH

6: CPI 9FH ; Compares 9FH with contents of A. The contents of A remains unaffected but status of result is reflected by flags.

A :1000 1010

9F H :1001 1111

1111 1011

So, CY = 1 & Z = 0

7: STA 3010H ; Stores the contents of A at memory location 3010H.

8: HLT ; Halts the execution of program.

Thus, after execution of line 7 of the program, the CY flag is'1' and Z flag is '0'.

Machine Control Instructions

1. HLT

i. Halt the execution of program
ii. 1-byte instruction
iii. No addressing mode
iv. Two or more machine cycles & 5 or more T-states
v. No flag is affected.

Note: *When HLT statement is given , the microprocessor completes the execution of current instruction and halts further execution. The microprocessor enters into Halt acknowledgment machine cycle and Wait states are inserted in every clock cycle. All the address and data buses are tri-stated (High impedance state) and contents of registers are unaffected during execution of HLT instruction. The processor can be brought out of the wait state either by pressing RESET key or by giving an interrupt.*

2. NOP

i. No operation is performed.
ii. 1-byte instruction
iii. No addressing mode
iv. One machine cycle (OP) & 4 T-states
v. No flag is affected.

Note: *NOP instruction is used to introduce a delay of 4 T states between two successive instructions.*

Ex.6.15. In 8085 with 2 MHz clock frequency, what is the time delay obtained after execution of 4 NOP instructions ?

IAS(2007)

Solution:

The instruction NOP require only Opcode fetch machine cycle.

No. of T-states in opcode fetch machinecycle = 4

No. of T-states required in 4 NOP instructions = 16

One T state is precisely equal to one time period of clock signal.

So, Time of one T-state, $T = \frac{1}{2}\ \mu s$

Time of 16 T-states = 16T = $16 \times \frac{1}{2} = 8\ \mu s$

So, execution of 4 NOP instructions introduces a time delay of 8 μs.

6.4.6 Stack and Its Related Instructions

Stack is the group of memory locations used by the microprocessor or programmer to store the data temporarily during the execution of a program. The stack can be used to store the contents of program counter (PC) by the microprocessor automatically when a subroutine is called. The stack can also be used by the programmer for storing the contents of register pairs of BC, DE, HL & PSW (Program Status word). Program Status Word indicates the contents of Accumulator and Flag register. Here Accumulator is considered as higher order and Flag register is considered as lower order register. Top of the stack is pointed by a 16 bit register called stack pointer(SP). The stack pointer is the programmable register which always stores the address of top most location of the stack. The stack pointer can be initiated by the instruction, LXI SP, 16 bit data. The contents of a register pair can be stored on the top of the stack by using PUSH instruction and contents from the top of the stack can be loaded in to a regster pair using POP instruction. Stack pointer is always decremented by 2 after the execution of each PUSH instruction and it is incremented by 2 after the execution of each POP instructions. It works on the principle of last-in first-out. The information on stack is not destroyed until new information is stored in those locations.

Note: *The data storage on the stack begins from the address next to the address with which stack pointer is initiated. For example when SP is initiated with 2000H the data storage will begin from 1FFF H.*

Note: *Stack pointer must be initiated before using PUSH, POP or Call instructions.*

Jump Instructions

I. Unconditional jump instruction:

1. JMP 16 bit address

- **i.** Jump execution of program immediately at the given 16-bit address
- **ii.** 3-byte instruction
- **iii.** Immediate addressing mode
- **iv.** Three machine cycles (OP+MR+MR) &10 T-states
- **v.** No flag is affected.

II. Conditional jump instruction.

1. JC 16 bit address - Jump if CY = 1

2. JNC 16 bit address - Jump if CY = 0

3. JZ 16 bit address - Jump if Z = 1

4. JNZ 16 bit address - Jump if Z = 0

5. JP 16 bit address - Jump if plus i.e. S = 0

6. JM 16 bit address - Jump if minus i.e S = 1

7. **JPE** **16 bit address** - Jump if P = 1

8. **JPO** **16 bit address** - Jump if P = 0

i. Jump execution of program immediately at the given 16-bit address if condition is true

ii. 3-byte instruction

iii. Immediate addressing mode

iv. **a.** Three machine cycles (OP+MR+MR) & 10 T-states if condition is true

b. Two machine cycles & 7 T-states if condition is false

v. No flag is affected

Note : *When jump instruction is executed by microprocessor it takes total three machine cycles. During first machine cycle it fetches the Opcode of instruction. During second and third machine cycles it reads the address which is operand of the instruction and stores it in temporary register pair WZ. Then microprocessor places the address of from WZ pair on address lines to transfer the execution to new address. The program counter is then loaded with WZ +1 i.e address of next byte to be fetched and thereafter normal execution is resumed through program counter. In conditional return microprocessor checks the conditon during first two machine cycles and does not go for third machine cycle if the condition fails.*

Ex.6.16 Consider the program given below, which transfers a block of data from one place in memory to another :

```
        MVI     C, 0B_H
        LXI     H, 2400_H
        LXI     D, 3400_H
L1 :    MOV     A, M
        STAX    D
        INR     L
        INR     E
        DCR     C
        JNZ     LI
```

What is the total number of memory accesses (including instruction fetches) carried out ?

IES(EE,05)

Solution:

Mnemonic	*Operand*	*No. of machine cycles/memory accesses*
MVI	C, $0B_H$	2
LXI	H,2400_H	3
LXI	D, 3400_H	3
L1 : MOV	A, M	2
STAX	D	2
INR	L	1
INR	E	1
DCR	C	1
JNZ	LI	3 or 2

Instructions before L1 needs 8 memory accesses and L1 to JNZ L1 requires 10 memory accesses during each iteration upto last iteration but 9 machine cycles in last iteration when Z = 0. The loop is executed 11 times till the contents of C becomes zero. So, the total number of memory accesses are 8 + 10*10 + 9 = 117. JNZ requires 2 machine cycles when condition is false. So total machine cycles will be 117.

Call Instructions

Call instructions are used to call a subroutine during execution of a main program. A subroutine is a set of instructions used to perform a task repeatedly.

I. Unconditional Call Instruction

1. CALL 16 bit address

i. Call immediately the subroutine stored at the given 16- bit address
ii. 3-byte instruction
iii. Immediate/Register Indirect addressing mode
iv. Five machine cycles (OP+MR+MR+MW+MW) & 18 T-states
v. No flag is affected.

II. Conditional Call Instructions

1. CC 16 bit address - Call if CY = 1
2. CNC 16 bit address - Call if CY = 0
3. CZ 16 bit address - Call if Z = 1
4. CNZ 16 bit address - Call if Z = 0
5. CP 16 bit address - Call if plus i.e. S = 0
6. CM 16 bit address - Call if minus i.e S = 1
7. CPE 16 bit address - Call if P = 1
8. CPO 16 bit address - Call if P = 0

i. Call immediately the subroutine stored at the given 16- bit address if condition is true.
ii. 3-byte instruction
iii. Immediate/Register Indirect addressing mode
iv. **a.** Five machine cycles (OP+MR+MR + MW + MW) &18 T-states if condition is true.
b. Two machine cycles & 9T-states if condition is false.
v. No flag is affected

counter is then loaded with WZ +1 i.e address of next byte to be fetched.

Note : *During execution of conditional call instructions if condition is true the number of machines cycles are same as in case of unconditional call but if condition is false only two machines cycles are required.*

Note : *When CALL instruction is executed by microprocessor it takes total five machine cycles. During first machine cycle it fetches the Opcode of instruction. During second and third machine cycles it reads the address of subroutine and stores in temporary register pair WZ. In the 4th and 5th machine cycles the microprocessor saves the contents of Program Counter(PC), which has address of next instruction is to be executed, on the top of stack. So, stack pointer must be initiated in the main program before calling of subroutine.After saving the contents of program counter on the top of stack the microprocessor places the address of subroutine, from WZ pair, on address lines to transfer the execution to subroutine. The program*

Return Instructions

Return instructions are used at the end of the subroutine. When microprocessor encounters a return instruction , the execution of the program is returned to the main program.

I. Unconditional Return Instruction

1. RET

- **i.** Return to main program unconditionally
- **ii.** 1-byte instruction
- **iii.** Register Indirect addressing mode
- **iv.** Three machine cycles (OP+MR+MR) & 10 T-states
- **v.** No flag is affected.

II. Conditional Return Instructions

1. **RC** - Return if CY = 1
2. **RNC** - Return if CY = 0
3. **RZ** - Return if Z = 1
4. **RNZ** - Return if Z = 0
5. **RP** - Return if plus i.e. S = 0
6. **RM** - Return if minus i.e S = 1
7. **RPE** - Return if P = 1
8. **RPO** - Return if P = 0

- **i.** Return to main program if condition is true.
- **ii.** 1-byte instruction
- **iii.** Register Indirect addressing mode
- **iv.** **a.** Three machine cycles (OP+MR+MR) &12 T-states if condition is true.
  b. One machine cycles & 6 T-states if condition is false.
- **v.** No flag is affected.

Note 1: *When RET instruction is executed by microprocessor it takes three machines cycles. During first machine cycle miroprocessor fetches the Opcode and during second and third machines it reads the contents of top of stack and stores in temporary register pair WZ. The contents of WZ pair are ,then, placed on address lines and program counter is loaded with WZ +1 i.e. address of next byte to be fetched.*

Note 2 : *During execution of conditional return instructions if condition is true the number of machines cycles are same as in case of unconditional return but if condition is false only one machines cycle is required.*

Ex.6.28 In an 8085 based system the subroutine TEST given below is called by another program.

```
TEST:   MVI A, 00H
        CALL TEST1
TEST1:  INR A
        RET
```

What are the contents of the accumulator when the processor returns from the subroutine TEST?

GATE(IN,02)

Solution :

```
TEST:   MVI A, 00H      ;  Moves 00H to register A. So,  A = 00H
        CALL TEST1      ;  Calls the subroutine TEST1.
TEST1:  INR A           ;  Increments the contents of A by one.
        RET             ;  Returns to main program.
```

In the above program the instructions INR A is executed twice. Once by calling of TEST1 by Subroutine TEST and once by subroutine TEST itself. So, the contents of accumulator are incremented twice. Therefore, the contents of A after returning from subroutine TEST will be 02_H.

1. **PUSH R_p**
   R_p→BC, DE, HL & PSW

   i. Store the contents of register pair R_p on two top locations of stack.
   ii. 1-byte instruction
   iii. Register indirect addressing mode
   iv. Three machine cycles (OP+MW+MW) & 12 T-states
   v. No flag is affected

Note: *PSW (Program Status Word) represents the contents of the accumulator and the flag regiser; the accumulator is high order register and the flag register is low order register.*

Note: *PUSH instruction first decrements the SP by one and then copies the contents of higher order register of pair R_p on the location shown by stack pointer . Then stack pointer register is again decremented by one and the content of lower order register of pair R_p are copied on the location shown by stack pointer. Thus stack pointer is decremented by two after execution of PUSH instruction.*

2. **POP R_p**
   R_p→BC, DE, HL & PSW

   i. Retrieve the contents of two top locations of stack to register pair R_p .
   ii. 1-byte instruction
   iii. Register indirect addressing mode
   iv. Three machine cycles (OP+MR+MR) & 10 T-states
   v. No flag is affected

Note: *POP instruction first copies the contents of memory location indicated by stack pointer (SP) to lower order register of pair R_p and then increment stack pointer by one . It again copies the contents of memory location indicated by SP to higher order register of pair R_p and then increment stack pointer by one. Thus stack pointer is incremented by two after execution of each POP instruction.*

3. **XTHL**

   i. Exchange the contents of two top locations of stack with contents of HL pair.
   ii. 1-byte instruction
   iii. Register indirect addressing mode
   iv. Five machine cycles (OP+MR+MR + MW + MW) &16 T-states
   v. No flag is affected.

Note: *The contents of L are exchanged with location indicated by SP and contents of H are exchanged with memory location (SP+1). However , the contents of SP remains unaltered.*

Note: *All instructions related to stack have indirect addressing mode.*

Ex.6.24 Consider the following Assembly Language program :

```
MVI A, 30H
ADI  30H
XRA  A
POP H
```

What are the contents of A at the end of program.

IES(EE.98)

Solutions:

```
MVIA, 30H          ;   Moves data  30H  to A
ADI 30H            ;   Adds 30H   to contents of A and store result in A.
```

$A + A$: 00110000
+ 00110000
= 01100000

$\Rightarrow$ A = 60H

```
XRA  A             ;   Performs  ExOR  operation  the  contents  of  A  and
store                  result in A.
```

$A \oplus A$: 01100000
$\oplus$ 01100000
= 00000000

$\Rightarrow$ A = 00H

```
POP H              ;   Moves the contents of top of  stack to HL pair.
```

Thus, the contents of A after the execution of program will be A = 00H

Ex.6.25 In a 8085 microprocessor the value of the stack pointer (SP) is 2010H and that of DE register pair is 1234H before the following code is executed.

```
LXI H, 0000H
PUSH H
PUSH H
POP B
DAD SP
XCHG
```

What are contents of the DE register pair after the execution of above program?

GATE(IN,05)

Solution:

Given, Contents of DE =1234H
Contents of SP =2010H

LXI H, 0000H	;	Loads HL pair with 0000_H. So, H= 00H and L = 00H
PUSH H	;	Stores the contents of HL pair on the top of the stack. The stack pointer is first decremented by one from 2010H to 200FH and then contents of H(i.e.00H) are stored at 200FH, the stack pointer is again decremented by one from 200FH to 200EH and then contents of L(i.e.00H) are stored at 200EH. So at end of instruction, contents of SP are 200EH, contents of 200EH are 00H and contents of 200FH are 00H.
PUSH H	;	Stores the contents of HL pair on the top of the stack The stack pointer is first decremented by one from 200EH to 200DH and then contents of H(i.e.00H) are stored at 200DH, the stack pointer is again decremented by one from $200D_H$ to 200CH and then contents of L(i.e.00H) are stored at 200CH. So at end of instruction, contents of SP are 200CH, contents of 200CH are 00H and contents of 200DH are 00H.

POP B ; Stores the contents of top of stack in BC pair. The contents of 200CH(i.e.00H) are store in C and then stack pointer is incremented by one from 200CH to 200DH and then contents of 200DH(i.e.00H) are stored in B, the stack pointer is again incremented by one from 200DH to 200EH. So at end of instruction, contents of SP are 200EH, contents of B are 00H and contents of C are 00H.

DAD SP ; Adds the contents of stack pointer(i.e. SP=200EH) to contents of HL pair(i.e.0000H) and store the result in HL pair and contents of SP remains unaltered.

HL : 0000 0000 0000 0000

SP : 0010 0000 0000 1110

0010 0000 0000 1110

2 0 0 E*c*

So, now contents of HL pair becomes 200EH

XCHG ; Exchanges the contents of HL pair(i.e. 200EH) with contents of DE pair(i.e.1234H). At end of instruction HL=1234H and DE = 200EH.

The value of the DE register pair after the execution of program is 200EH.

Addressing Modes of 8085

Every instruction of a microprocessor system has some data on which it performs its operation. There are many techniques of specifying this data. These techniques of specifying the data of instruction are called addressing modes. 8085 microprocessor has five different types of modes which are briefly described as under,

1. **Register Addressing Mode**

   When the data transfer is between two registers the addressing mode is called register addressing mode. e.g. MOV A,B, PCHL, SPHL , ADD B , SUB B etc.

2. **Direct Addressing Mode**

   When the data transfer is with the memory or I/O device and address of memory or I/O device is included in the instruction itself, the addressing mode is called direct addressing mode. e.g. LDA 16 bit address, STA 16 bit address , IN 8bit address, OUT 8bit address etc.

3. **Indirect Addressing Mode**

   When the data transfer is with the memory or I/O device and address of memory or I/O device is not included in the instruction rather it is indicated through a memory pointer, the addressing mode is called indirect addressing mode.
   e.g. MOV M, R ; LDAX D etc.

4. **Implicit or Implied Addressing Mode**

   When the instruction modifies the contents of accumulator without using any operand the mode of addressing is called implicit or implied. In implied addressing mode, the location of the operand is contained within the opcode e.g. CMA, RLC, etc.

5. Immediate Addressing Mode

When 8 bit or 16 bit data is given in the instruction as operand the addressing mode is called immediate addressing mode. e.g. ADI 8 bit data; SUI 8bit data; MVI R, 8 bitdata; LXI R_p, 16 bit data.

Note: *i. When ever there is letter 'I' at the end in a Mnemonics that indicates immediate addressing mode.*

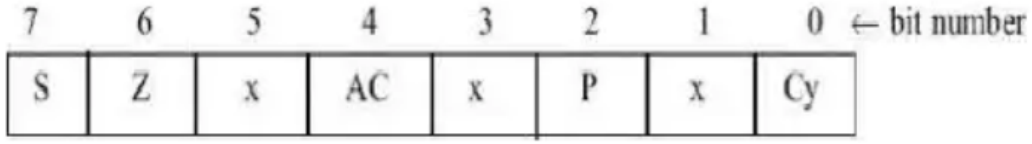
ii. When one of the operand is letter 'M' that indicates the indirect addressing mode.

Flag Register

In 8085 microprocessor, the flags register can have a total of eight flags. Thus a flag can be represented by 1 bit of information. But only five flags are implemented in 8085. And they are:

- **Carry flag (Cy),**
- **Auxiliary carry flag (AC),**
- **Sign flag (S),**
- **Parity flag (P), and**
- **Zero flag (Z).**

The respective position of these flag bits in flag register has been show the below figure. The positions marked by “x” are to be considered as don't care bits in the flags register. The user is not required to memorize the positions of these flags in the flags register.



- **Auxiliary carry flag (Ac):** Now let us consider the addition of any two 8-bit (2-hex digit) numbers, a carry may be generated when we add the LS hex digits of the two numbers. Such a carry is called intermediate carry also known as half carry, or auxiliary carry (AC). Intel prefers to call it AC. In the above Example 1, AC was not generated but in Example 2, AC is generated.

Now consider the programmer's view of 8085 contains the flags register has been depicted in the following figure –

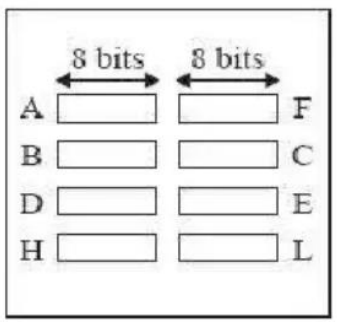


Fig. Programmer's view of 8085 Flags register

These individual flags are either set to 1, or reset to 0 depending on the result of execution of the last executed arithmetic or logical instruction. But in a few arithmetic and logical instructions, some or none of these flags are affected. Also there are some arithmetic and logical instructions, flag bits in the flag register will not get affected as well. As example in the execution of DCX and INX instructions, flag bit in flag register will not get affected at all.

However, in any data transfer instruction, none of the flags bits in flag register are affected.

As this is only an intermediate carry, we may not be interested in storing this bit information. But 8085 microprocessor still stores this AC information in bit position 4 of the flags register. The result of execution of DAA instruction, is affected by the status of this flag. However, in our 8085 instruction set does not provide any instruction, which explicitly uses the AC flag.

- **Carry flag (Cy):** after performing the addition of any two 8-bit numbers, the carry generated can be either 0 or 1. That is only 1-bit. Thus to store the carry information 1-bit storage is enough. The Cy flag is stored in the LS bit position in the flags register. Instructions that use the Cy flag are widely used in the user programs.
- **Example 1:** In the addition of 45H and F3H, the result thus produced will be 38H and with Cy flag = 1, as shown below.

```
Cy AC
 10
  4 5H
 +F 3H
-------
  3 8H = 0011 1000
```

Example 2: In the addition of 85H and 1EH, the result thus produced will be A3H with Cy = 0, as shown below.

```
Cy AC
 01
  8 5H
 +1 EH
-------
  A 3H = 1010 0011
```

- **Zero flag (Z):** The Z flag is set to 1, if after arithmetic and logical operations, the 8-bit result thus produced, is 00H. If the 8-bit result is not equal to 00H, the Z flag is reset to 0. Thus the Z flag is hoisted to indicate that the result is 0.

In previous Example 1, as the 8-bit result is 38H and is non-zero, the Z flag is reset to 0. Also in the other hand in Example 2, as the 8-bit result is A3H, the Z flag is reset to 0 here again. Instructions that use the Z flag are widely used in the user programs.

- **Parity flag (P):** The P flag is set to 1, if the 8-bit result thus produced against any logical and arithmetic operation has an even number of 1's in it. If there are odd number of 1's in the 8-bit result, the P flag is reset to 0.

In our previous Example 1, as the 8-bit result 38H = 0011 1000 has three numbers of 1's, so having odd number of 1's, the parity flag is reset to 0. On the other hand, in Example 2, as the 8-bit result A3H = 1010 0011 has four numbers of 1's (so an even number of 1's), the parity flag is set to 1.

- **Sign flag (S):** The S flag is set to 1, when the result thus produced against any logical or arithmetic operations is negative, indicated by MS bit of 8-bit result being 1. It is reset to 0 otherwise if the result is positive, indicated by MS bit of 8-bit result being 0.

Thus, the value of S flag is essentially the value of the MS bit of the 8-bit result. In the above Example 1, as the 8-bit result is 38H = 0 011 1000, 0 in MSB indicates result is positive and the sign flag is reset to 0. Note that we are not considering here the 9-bit result including the carry, to decide the S flag value. In Example 2, as the 8-bit result is A3H = 1 010 0011, the MSB has become 1 that means negative and the sign flag is set to 1.

Thanks