

### 1. Explain one-byte, two-byte and three-byte instructions with appropriate example.

#### Ans. One-byte Instruction

It includes Opcode and Operand in the same byte.

##### Example

MOV B,A  
LDAX D  
STAX B  
PUSH D  
POP B  
RLC

#### Two-byte Instruction

Here,

1<sup>st</sup> Byte : Specifies Opcode

2<sup>nd</sup> Byte: Specifies Operand

##### Example

MVI A,32  
ADI 01  
IN 02  
OUT 05

#### Three-byte Instruction

Here,

1<sup>st</sup> Byte: Specifies Opcode

2<sup>nd</sup> Byte: Specifies lower order 8-bit address

3<sup>rd</sup> Byte: Specifies higher order 8-bit address

##### Example

LXI B,2010  
JMP 3002  
CALL 4060  
LDA 1212  
STA 1213

### 2. Explain Data Transfer Instructions with example.

**Ans.** Instructions copy data from source to destination. While copying, the contents of source is not modified. Data Transfer Instructions do not affect the flags.

#### 1. MOV: Move data from source to destination

It copies the contents of the source register into the destination register.

Contents of the source register is not altered. If one of the operands is a memory, its location is specified by the contents of the HL registers. It is one-byte instruction.

**Syntax:** MOV Destination, Source

**Example:** MOV B, C ; B←C

MOV B, M; B←M[HL]

MOV M, B; M[HL]←B

#### 2. MVI: Load 8-bit to Register/Memory

The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. It is two-byte instruction.

**Syntax:** MVI R/M, 8-bit Data

**Example:** MVI B, 57H; B←12

MVI M, 12H ; M[HL]←12

#### 3. LDA: Load Accumulator

The contents of a memory location, specified by a 16-bit address in the operand, is copied to the accumulator. The contents of the source is not altered.

**Syntax:** LDA 16-bit address

**Example:** LDA 2050H; A←M[2050]

#### 4. STA: Store Accumulator

The contents of accumulator is copied into the memory location specified by the operand.

**Syntax:** STA 16-bit address

**Example:** STA 0002H; M[0002]←A

#### 5. LDAX: Load the accumulator indirect

The contents of a memory location, specified by a 16-bit address in an operand, is copied to the accumulator.

**Syntax:** LDAX R<sub>p</sub>(B/D)

**Example:** LDAX B ; A←M[BC]

LDAX D ; A←M[DE]

#### 6. STAX: Store Accumulator Indirect

The contents of accumulator is copied into memory location specified by the contents of the operand (register pair). The contents of the accumulator is not altered.

**Syntax:** STAX R<sub>p</sub>

**Example:** STAX B; M[BC]←A

### 7. LXI: Load the immediate register pair

The instruction loads immediate 16-bit data to register pair.

**Syntax:** LXI Rp, 16-bit data

**Example:** LXI H, 2034H; HL←2034

### 8. LHLD: Load H and L registers direct

The instruction copies contents of the memory location pointed out by the address into register L and copies the contents of the next memory location into register H. The contents of source memory locations is not altered.

**Syntax:** LHLD 16-bit address

**Example:** LHLD 2050H; L←M[2050], H←M[2051]

### 9. SHLD: Store H and L registers direct

The contents of register L is stored in memory location specified by the 16-bit address in the operand and the contents of H register is stored into the next memory location by incrementing the operand.

**Syntax:** SHLD 16-bit address

**Example:** SHLD 2050H; M[2050] ←L, M[2051] ←H

### 10. XCHG: Exchange H and L with D and E

The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.

**Syntax:** XCHG

**Example:** XCHG ;HL↔DE

### 11. SPHL: Copy H and L registers to stack pointer

The instruction loads the contents of the H and L registers into the stack pointer register, the contents of H register provide the high-order address and the contents of L register provide the low-order address. The contents of the H and L registers are not altered.

**Syntax:** SPHL

**Example:** SPHL; SP←HL

### 12. XTHL: Exchange H and L with top of stack

The contents of L register is exchanged with stack location pointed out by contents SP. The contents of the H register are exchanged with the next stack location (SP+1).

**Syntax:** XTHL

**Example:** XTHL; HL↔M[SP]

### 13. PUSH: Push the register pair onto the stack

The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The SP register is decremented and the contents of the high order register (B, D, H) are copied into that location. The SP register is decremented again and the contents of the low-order register (C, E, L) are copied to that location.

**Syntax:** PUSH R<sub>p</sub>

**Example:** PUSH B; SP ← SP-1, SP ← B, SP ← SP-1, SP ← C

### 14. POP : Pop off stack to the register pair

The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H) of the operand. The stack pointer register is again incremented by 1.

**Syntax:** POP R<sub>p</sub>

**Example:** POP B; C ← SP, SP ← SP+1, B ← SP, SP ← SP+1

### 15. OUT: Output from Accumulator to 8-bit port

The contents of the accumulator are copied into the I/O port specified by the operand.

**Syntax:** OUT 8-bit port address

**Example:** OUT 02

### 16. IN: Input data to accumulator from with 8-bit port

The contents of the input port designated in the operand are read and loaded into the accumulator.

**Syntax:** IN 8-bit port address

**Example:** IN 02

## 3. Explain Arithmetic Instructions with example.

**Ans.**

### 1. ADD: Add register/memory to accumulator

The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, it is specified by the contents of the HL registers. Flags can be modified to reflect the result of the addition.

**Syntax:** ADD R/M

**Example:** ADD B; A ← A + B

ADD M; A ← A + M[HL]

### 2. ADC: Add register to accumulator with carry

The contents of the operand (register or memory) and the Carry flag is added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers.

**Syntax:** ADC R/M

**Example:** ADC B; A ← A + B + CY

ADC M; A ← A + M[HL] + CY

### 3. ADI: Add immediate 8-bit with accumulator

The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator.

**Syntax:** ADI 8-bit data

**Example:** ADI 03; A ← A + 03h

### 4. ACI: Add immediate 8-bit to accumulator with carry

The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator.

**Syntax:** ACI 8-bit data

**Example:** ACI 03; A ← A + 03h + CY

### 5. DAD : Add register pair to H and L registers

The 16-bit contents of the specified register pair are added to the contents of the HL register and the sum is stored in the HL register. The contents of the source register pair are not altered. If the result is larger than 16 bits, the CY flag is set. No other flags are affected.

**Syntax:** DAD R<sub>p</sub>

**Example:** DAD B; HL ← HL + BC

### 6. SUB : Subtract register/memory from accumulator

The contents of the operand (register or memory) is subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers.

**Syntax:** SUB R/M

**Example:** SUB B ; A ← A - B

SUB M ; A ← A - M[HL]

### 7. SBB: Subtract source & borrow from accumulator

The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers.

**Syntax:** SBB R/M

**Example:** SBB B ; A ← A - B - CY

SBB M ; A ← A - M[HL] - CY

### 8. SUI : Subtract immediate 8-bit from accumulator

The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator.

**Syntax:** SUI 8-bit data

**Example:** SUI 08h; A ← A - 08h

### 9. SBI : Subtract immediate from accumulator with borrow

The 8-bit data (operand) and the borrow (CY) are subtracted from the contents of the accumulator and the result is stored in the accumulator.

**Syntax:** SBI 8-bit data

**Example:** SBI 08h; A ← A - 08 - CY

### 10. INR: Increment register/memory by 1

The contents of the designated register or memory is incremented by 1 and the result is stored at the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

**Syntax:** INR R/M

**Example:** INR B; B ← B + 01

INR M; M[HL] ← M[HL] + 01

### 11. INX : Increment register pair by 1

The contents of the designated register pair is incremented by 1 and the result is stored at the same place.

**Syntax:** INX R<sub>p</sub>

**Example:** INX D; DE←DE+0001

### 12. DCR: Decrement register/memory by 1

The contents of the designated register or memory is decremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.

**Syntax:** DCR R/M

**Example:** DCR B; B=B-01

DCR M; M[HL]=M[HL]-01

### 13. DCX: Decrement register pair by 1

The contents of the designated register pair is decremented by 1 and their result is stored at the same place.

**Syntax:** DCX R<sub>p</sub>

**Example:** DCX B; BC=BC- 0001

DCX D; DE=DE- 0001

### 14. DAA: Decimal Adjust Accumulator

The contents of the accumulator is changed from a binary value to two 4-bit BCD digits. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.

**Syntax:** DAA

**Example:** DAA

## 4. Explain Logical Instructions with example.

### Ans. 1. CMP: Compare register/memory with accumulator

The contents of the operand (register or memory) is compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags:

if (A) < (reg/mem): carry flag is set(1).

if (A) = (reg/mem): zero flag is set(1).

if (A) > (reg/mem): carry and zero flags are reset(0).

**Syntax:** CMP R/M

**Example:** CMP B; A<>B

CMP M; A<>M[HL]

### 2. CPI: Compare immediate with accumulator

The second byte data is compared with the contents of the accumulator.

The values being compared remain unchanged. The result of the comparison is shown by setting the flags:

if (A) < data: carry flag is set(1).

if (A) = data: zero flag is set(1).

if (A) > data: carry and zero flags are reset(0).

**Syntax:** CPI 8-bit data

**Example:** CPI 03; A<>03H

### 3. ANA: AND register/memory with accumulator

The contents of the accumulator are logically ANDed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers.

S, Z, P are modified to reflect the result of the operation.

CY is reset. AC is set.

**Syntax:** ANA R/M

**Example:** ANA B; A  $\wedge$  B

ANA M; A  $\wedge$  M[HL]

### 4. ANI: AND immediate with accumulator

The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset & AC is set.

**Syntax:** ANI 8-bit data

**Example:** ANI 02H

### 5. ORA: OR register/memory with accumulator

The contents of the accumulator is logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

**Syntax:** ORA R/M

**Example:** ORA B; A  $\vee$  B

ORA M; A  $\vee$  M[HL]

### 6. ORI: OR immediate with accumulator

The contents of the accumulator is logically ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset and AC is set.

**Syntax:** ORI 8-bit data

**Example:** ORI 02H

### 7. XRA: Exclusive OR register/memory with accumulator

The contents of the accumulator is Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator.

If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.

**Syntax:** XRA R/M

**Example:** XRA B  
XRA M

### 8. XRI: Exclusive OR immediate with accumulator

The contents of the accumulator are Exclusive Ored with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset and AC is set.

**Syntax:** XRI 8-bit data

**Example:** XRI 07

### 9. RLC: Rotate accumulator left

Each binary bit of the accumulator is rotated left by one position. Bit D<sub>7</sub> is placed in the position of D<sub>0</sub> as well as in the Carry flag (CY). CY is modified according to bit D<sub>7</sub>. S, Z, P, AC are not affected.

**Syntax:** RLC

**Example:** RLC

### 10. RRC: Rotate accumulator right

Each binary bit of the accumulator is rotated right by one position. Bit D<sub>0</sub> is placed in the position of D<sub>7</sub> as well as in the Carry flag (CY). CY is modified according to bit D<sub>0</sub>. S, Z, P, AC are not affected.

**Syntax:** RRC

**Example:** RRC

### 11. RAL: Rotate accumulator left through carry

Each binary bit of the accumulator is rotated left by one position through the Carry flag. Bit D<sub>7</sub> is placed in the Carry flag, and the Carry flag is placed in the least significant position D<sub>0</sub>. CY is modified according to bit D<sub>7</sub>. S, Z, P, AC are not affected.

**Syntax:** RAL

**Example:** RAL

### 12. RAR: Rotate accumulator right through carry

Each binary bit of the accumulator is rotated right by one position through the Carry flag. Bit D<sub>0</sub> is placed in the Carry flag, and the Carry flag is placed in the most significant position D<sub>7</sub>. CY is modified according to bit D<sub>0</sub>. S, Z, P, AC are not affected.

**Syntax:** RAR

**Example:** RAR



### 13. CMA: Complement accumulator

The contents of the accumulator are complemented. No flags are affected.

**Syntax:** CMA

**Example:** CMA

### 14. CMC: Complement Carry

The Carry flag is complemented. No other flags are affected.

**Syntax:** CMC

**Example:** CMC

### 15. STC: Set Carry

The Carry flag is set (1). No other flags are affected.

**Syntax:** STC

**Example:** STC

## 5. Explain Branch Instructions with example.

**Ans.**

### 1. JMP: Jump unconditionally

The program sequence is transferred to the memory address given in the operand.

**Syntax:** JMP 16-bit address

**Example:** JMP 2050

Memory Address	Instructions	Memory Label	Instructions
0000H	MVI A,05		MVI A,05
0002H	MOV B,A		MOV B,A
0003H	MOV C,B		MOV C,B
0004H	JMP 0009		JMP L1
0007H	ADI 02		ADI 02
0009H	SUB B	L1:	SUB B
000AH	HLT		HLT

### 2. Jump Conditionally

Instruction	Description	Example
JC 16-bit address	Jump on Carry, Flag Status: CY=1	JC 2030H
JNC 16-bit address	Jump on No Carry, Flag Status: CY=0	JNC 2030H
JZ 16-bit address	Jump on Zero, Flag Status: Z=1	JZ 2030H
JNZ 16-bit address	Jump on No Zero, Flag Status: Z=0	JNZ 2030H
JP 16-bit address	Jump on Positive, Flag Status: S=0	JP 2030H
JM 16-bit address	Jump on Minus, Flag Status: S=1	JM 2030H
JPE 16-bit address	Jump on Parity Even, Flag Status: P=1	JPE 2030H
JPO 16-bit address	Jump on Parity Odd, Flag Status: P=0	JPO 2030H

### 3. CALL: Call Unconditionally

Instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction (PC) is pushed onto the stack.

**Syntax:** CALL 16-bit address

**Example:** CALL 000CH

Line	Instruction	Address	PC
1	LXI H,1002	[0000]	[0003]
2	LXI D,3002	[0003]	[0006]
3	<b>CALL ADD1</b>	[0006]	<b>[000C]</b>
4	LXI B,4002	<b>[0009]</b>	[000C]
5	<b>ADD1:MOV A,D</b>	<b>[000C]</b>	[000D]
6	ADD H	[000D]	[000E]
7	RET	[000E]	<b>[0009]</b>

### 4. CALL: Call Conditionally

Instruction	Description	Example
CC 16-bit address	Call on Carry, Flag Status: CY=1	CC 2030H
CNC 16-bit address	Call on No Carry, Flag Status: CY=0	CNC 2030H
CZ 16-bit address	Call on Zero, Flag Status: Z=1	CZ 2030H
CNZ 16-bit address	Call on No Zero, Flag Status: Z=0	CNZ 2030H
CP 16-bit address	Call on Positive, Flag Status: S=0	CP 2030H
CM 16-bit address	Call on Minus, Flag Status: S=1	CM 2030H
CPE 16-bit address	Call on Parity Even, Flag Status: P=1	CPE 2030H
CPO 16-bit address	Call on Parity Odd, Flag Status: P=0	CPO 2030H

### 5. RET: Return from subroutine Unconditionally

The program sequence is transferred from the subroutine to the calling program.

**Syntax:** RET

**Example:** RET

### 6. RET: Return from subroutine Conditionally

Instruction	Description	Example
RC 16-bit address	Return on Carry, CY=1	RC
RNC 16-bit address	Return on No Carry, CY=0	RNC
RZ 16-bit address	Return on Zero, Z=1	RZ
RNZ 16-bit address	Return on No Zero, Z=0	RNZ
RP 16-bit address	Return on Positive, S=0	RP
RM 16-bit address	Return on Minus, S=1	RP
RPE 16-bit address	Return on Parity Even, Flag Status: P=1	RPE
RPO 16-bit address	Return on Parity Odd, Flag Status: P=0	RPO

### 7. PCHL: Load program counter with HL contents

The contents of registers H & L are copied into the program counter. The contents of H are placed as the high-order byte and the contents of L as the low-order byte.

**Syntax:** PCHL

**Example:** PCHL

### 8. RST : Restart

The RST instruction is used as software instructions in a program to transfer the program execution to one of the following eight locations.

Instruction	Restart Address
RST 0	0000H
RST 1	0008H
RST 2	0010H
RST 3	0018H
RST 4	0020H
RST 5	0028H
RST 6	0030H
RST 7	0038H

**Syntax:** RST 0-7(N)

**Example:** RST 5

The 8085 has additionally 4 interrupts, which can generate RST instructions internally and doesn't require any external hardware.

Instruction	Description	Example
TRAP	It restart from address 0024H	TRAP
RST 5.5	It restart from address 002CH	RST 5.5
RST 6.5	It restart from address 0034H	RST 6.5
RST 7.5	It restart from address 003CH	RST 7.5

## 6. Explain Control Instructions with example.

**Ans.**

### 1. NOP: No Operation

No operation is performed. The instruction is fetched and decoded. However no operation is executed. It is used to increase processing time of execution. One CPU cycle is "wasted" to execute a NOP instruction.

**Syntax:** NOP

**Example:** NOP

### 2. HLT: Halt

The CPU finishes executing the current instruction and stops further execution. An interrupt or reset is necessary to exit from the halt state.

**Syntax:** HLT

**Example:** HLT

### 3. DI: Disable Interrupt

The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.

**Syntax:** DI

**Example:** DI

### 4. EI: Enable Interrupt

The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. This instruction is necessary to re enable the interrupts (except TRAP).

**Syntax:** EI

**Example:** EI

### 5. SIM: Set Interrupt Mask

This is a multipurpose instruction used to :

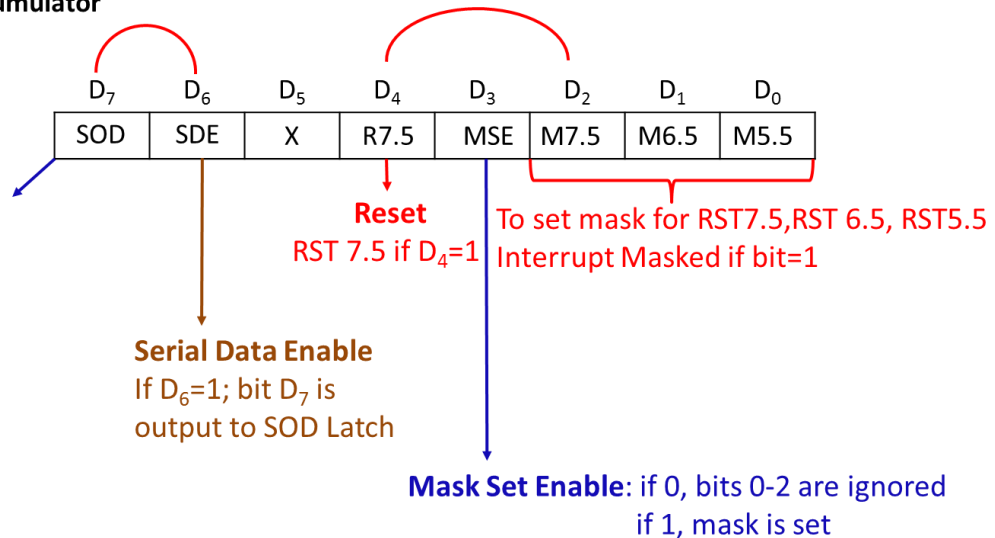
- Set the status of interrupts 7.5, 6.5, 5.5
- Set serial data input bit.

The instruction loads eight bits into accumulator with the ABOVE interpretations.

**Syntax:** SIM

**Example:** SIM

**A:Accumulator**



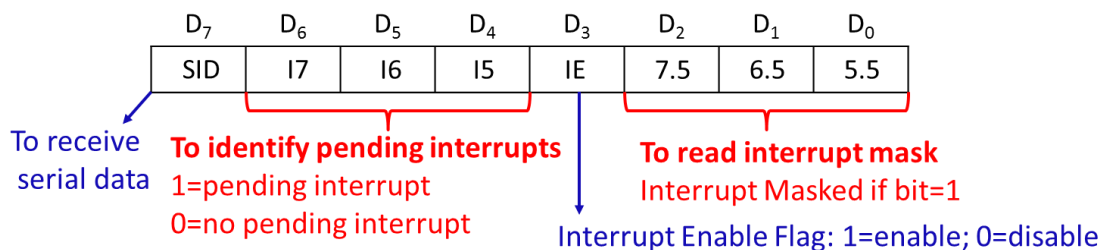
### 6. RIM: Read Interrupt Mask

This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5. Read serial data input bit. It reads eight bits from accumulator with following interpretations.

**Syntax:** RIM

**Example:** RIM

**A:Accumulator**



### 7. Explain the working of rotate instructions of 8085 with proper example in each case.

**Ans. RLC: Rotate accumulator left**

- Each binary bit of the accumulator is rotated left by one position.
- Bit D<sub>7</sub> is placed in the position of D<sub>0</sub> as well as in the Carry flag (CY).
- CY is modified according to bit D<sub>7</sub>.
- S, Z, P, AC are not affected.

		CY	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
Before RLC	<b>A8</b>		1	0	1	0	1	0	0	0
After RLC	<b>51</b>	1	0	1	0	1	0	0	0	D <sub>7</sub> (1)

**RRC: Rotate accumulator right**

- Each binary bit of the accumulator is rotated right by one position.
- Bit D<sub>0</sub> is placed in the position of D<sub>7</sub> as well as in the Carry flag (CY).
- CY is modified according to bit D<sub>0</sub>.
- S, Z, P, AC are not affected.

		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	CY
Before RRC	<b>A8</b>	1	0	1	0	1	0	0	0	
After RRC	<b>54</b>	D <sub>0</sub> (0)	1	0	1	0	1	0	0	0

**RAL: Rotate accumulator left through carry**

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D<sub>7</sub> is placed in the Carry flag, and the Carry flag is placed in the least significant position D<sub>0</sub>.
- CY is modified according to bit D<sub>7</sub>.
- S, Z, P, AC are not affected.

		CY	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
Before RAL	<b>A8</b>	0	1	0	1	0	1	0	0	0
After RAL	<b>50</b>	1	0	1	0	1	0	0	0	CY(0)

**RAR: Rotate accumulator right through carry**

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit D<sub>0</sub> is placed in the Carry flag, and the Carry flag is placed in the most significant position D<sub>7</sub>.
- CY is modified according to bit D<sub>0</sub>.
- S, Z, P, AC are not affected.

		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	CY
Before RAR	<b>A8</b>	1	0	1	0	1	0	0	0	1
After RAR	<b>D4</b>	CY(1)	1	0	1	0	1	0	0	0

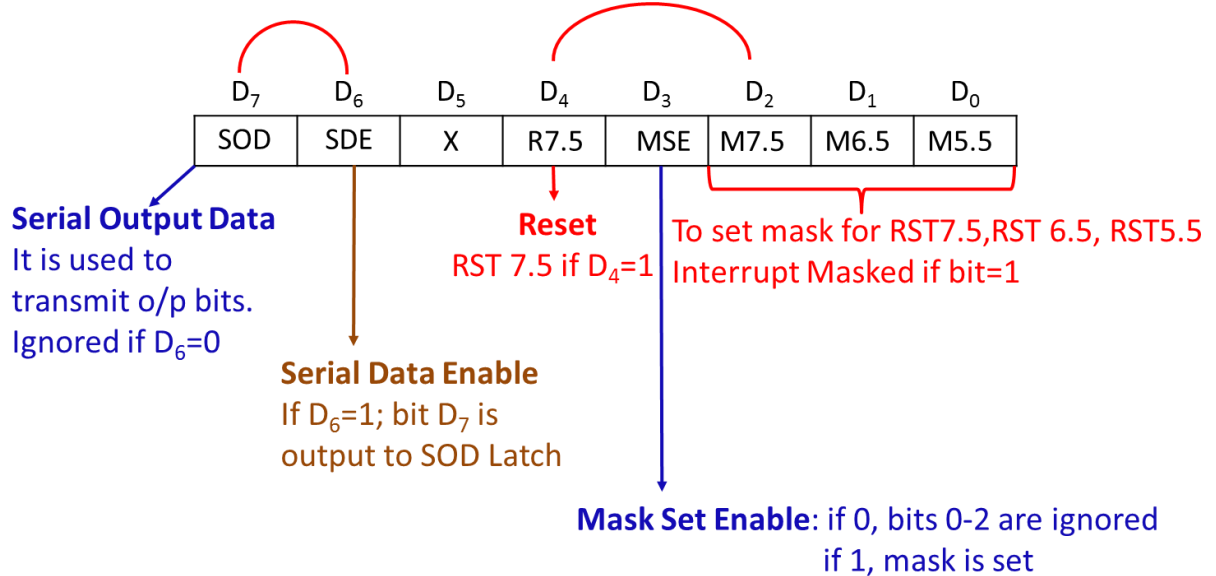
## 8. Explain RIM and SIM instructions with pseudo code example.

### Ans. SIM Instruction

This is a multipurpose instruction used to :

1. Set the status of interrupts 7.5, 6.5, 5.5.
2. Set serial data input bit.

The instruction loads eight bits in the accumulator with the following interpretations.



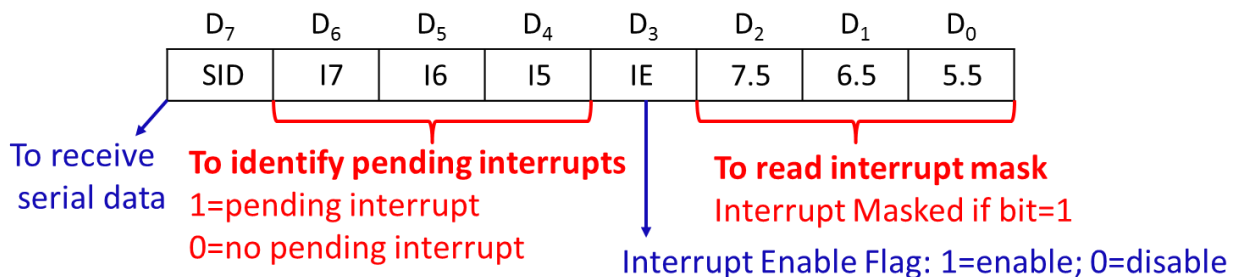
**Example 1:** MVI A,08H  
SIM

### RIM Instruction

This is a multipurpose instruction used to

1. Read the status of interrupts 7.5, 6.5, 5.5.
2. Read serial data input bit.

It reads eight bits from accumulator with following interpretations.



## 9. Explain the PUSH and POP instructions of the 8085 microprocessor with example.

**Ans. PUSH**

The contents of the register pair designated in the operand are copied onto the stack in the following sequence.

1. The SP register is decremented and the contents of the high order register (B, D, H) are copied into that location.
2. The SP register is decremented again and the contents of the low-order register (C, E, L) are copied to that location.

**Syntax:** PUSH Register\_pair

**Example:** PUSH B

```
SP <- SP-1
SP <- B    ;transfer high order bit to TOS
SP <- SP-1
SP <- C    ;transfer low order bit to TOS
```



### POP

The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L) of the operand.

1. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H) of the operand.
2. The stack pointer register is again incremented by 1.

**Syntax:** POP Register\_pair

**Example:** POP B

```
C <- SP    ; transfer to low order bit from TOS
SP <- SP+1
B <- SP    ; transfer to high order bit from TOS
SP <- SP+1
```





### 10. Explain various addressing modes of 8085 microprocessor with examples.

**Ans.**

#### 1) Immediate Addressing Mode

8/16 bit immediate data is specified in an instruction as one of its operand.

##### Example

MVI B 20;            20H is copied into register B  
LXI D 1000 ;        DE←1000H

#### 2) Direct Addressing Mode

8/16 bit address is directly specified in an instruction as one of its operand.

##### Example

LDA 2000;            2000H is memory address  
LHDL 3001;          L←M[3001], H←M[3002]  
IN 08;                08H is port address  
OUT 10;               10H is port address

#### 3) Register Addressing Mode

It specifies register or register pair that contains data.

##### Example

MOV A B;    A← B  
ADD B;       A←A+B

#### 4) Indirect Addressing Mode

16 bit memory address is indirectly provided with the instruction using a register pair.

##### Example

LDAX D;    A←M[DE]  
STAX D;    M[DE]←A

#### 5) Implicit Addressing Mode

It doesn't require any operand and the data is specified by the Opcode itself.

##### Example

RAL;        rotate left  
XCHG;       exchange DE and HL

**11. Explain all the instructions of 8085 with no. of bytes, machine cycles and T-states required for execution.**

**Ans.**

### DATA TRANSFER INSTRUCTIONS

Sr.	Instruction	Bytes	Machine Cycle	T-States
1	MOV B,C MOV M,A MOV A, M	1	F=1 F+MEMW=2 F+MEMR=2	4T 4+3=7T 4+3=7T
2	MVI B, 57H MVI M, 3CH	2	F+R=2 F+R+MEMW=3	4+3=7T 4+3+3=10T
3	LDA 2034H	3	F+MEMR_L+MEMR_H+MEMR=4	4+3+3+3=13T
4	LDAX B	1	F=1	4T
5	LXI H, 2034H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
6	LHLD 2040H	3	F+MEMR_L+MEMR+ MEMR_H+MEMR=5	4+3+3+3+3=16T
7	STA 4350H	3	F+MEMR_L+MEMR_H+MEMR=4	4+3+3+3=13T
8	STAX B	1	F+MEMW=2	4+3=7T
9	SHLD 2470H	3	F+MEMR_L+MEMR+ MEMR_H+ MEMR=5	4+3+3+3+3=16T
10	XCHG	1	F=1	4T
11	SPHL	1	F=1	6T
12	XTHL	1	F+MEMR+MEMW+MEMR +MEMW=5	4+3+3+3+3=16T
13	PUSH B	1	F+MEMW+MEMW=3	6+3+3=12T
14	POP H	1	F+MEMR+MEMR=3	4+3+3=10T
15	OUT F8H	2	F+R+IOW=3	4+3+3=10T
16	IN 8CH	2	F+R+IOR=3	4+3+3=10T

### ARITHMETIC INSTRUCTIONS

Sr.	Instruction	Bytes	Machine Cycle	T-States
1	ADD B ADD M	1	F=1 F+MEMR=2	4T 4+3=10T
2	ADC R ADC M	1	F=1 F+MEMR=2	4T 4+3=10T
3	ADI 45H	2	F+R=2	4+3=7T
4	ACI 23H	2	F+R=2	4+3=7T
5	DAD H	1	F+B+B=3	4+3+3=10T
6	SUB B SUB M	1	F=1 F+MEMR=2	4T 4+3=7T
7	SBB B SBB M	1	F=1 F+MEMR=2	4T 4+3=7T

8	SUI 45H	2	F+R=2	4+3=7T
9	SBI 45H	2	F+R=2	4+3=7T
10	INR B INR M	1	F=1 F+MEMR+MEMW=3	4T 4+3+3=10T
11	INX H	1	F=1	6T
12	DCR B DCR M	1	F=1 F+MEMR+MEMW=3	4T 4+3+3=10T
13	DCX H	1	F=1	6T
14	DAA	1	F=1	4T

### BRANCHING INSTRUCTIONS

Sr.	Instruction	Bytes	Machine Cycle	T-States
1	JMP 2034H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
2	JC 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
3	JNC 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
4	JP 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
5	JM 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
6	JZ 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
7	JNZ 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
8	JPE 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
9	JPO 2050H	3	F+MEMR_L+MEMR_H=3	4+3+3=10T
10	CALL 2034H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
11	CC 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
12	CNC 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
13	CP 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
14	CM 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
15	CZ 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
16	CNZ 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
17	CPE 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T
18	CPO 2050H	3	F+MEMR+MEMR+MEMW +MEMW=5	6+3+3+3+3=18T

19	RET	1	F+MEMR_L+MEMR_H=3	4+3+3=10T
20	RC	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
21	RNC	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
22	RP	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
23	RM	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
24	RZ	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
25	RNZ	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
26	RPE	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
27	RPO	1	F+MEMR_L+MEMR_H=3	6+3+3=10T
28	PCHL	1	F=1	6T
29	RST 0-7	1	F+MEMW+MEMW=3	4+3+3=10T

### LOGICAL INSTRUCTIONS

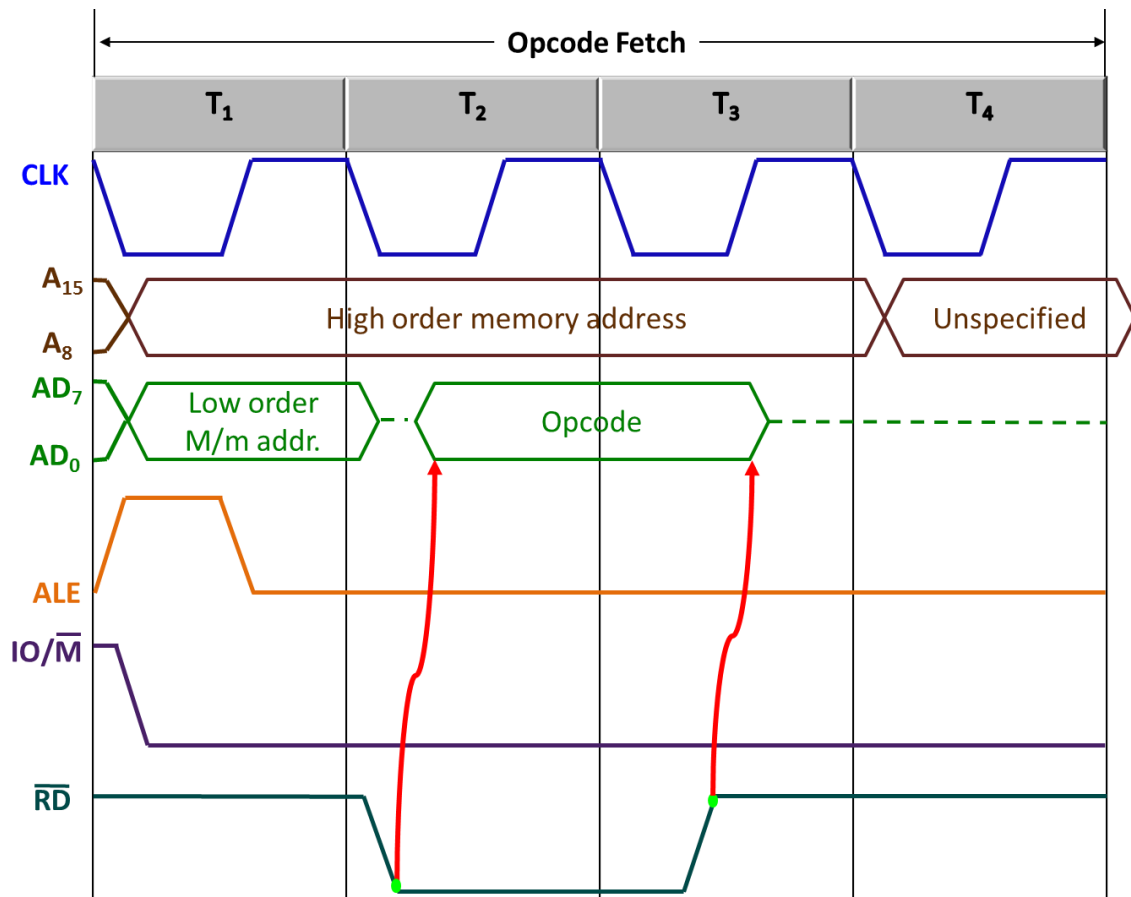
Sr.	Instruction	Bytes	Machine Cycle	T-States
1	CMP B CMP M	1	F=1 F+MEMR=2	4T 4+3=10T
2	CPI 89H	2	F+MEMR=2	4+3=7T
3	ANA B ANA M	1	F=1 F+MEMR=2	4T 4+3=10T
4	ANI 86H	2	F+MEMR=2	4+3=7T
5	XRA B XRA M	1	F=1 F+MEMR=2	4T 4+3=10T
6	XRI 86H	2	F+MEMR=2	4+3=7T
7	ORA B ORA M	1	F=1 F+MEMR=2	4T 4+3=10T
8	ORI 86H	2	F+MEMR=2	4+3=7T
9	RLC	1	F=1	4T
10	RRC	1	F=1	4T
11	RAL	1	F=1	4T
12	RAR	1	F=1	4T
13	CMA	1	F=1	4T
14	CMC	1	F=1	4T
15	STC	1	F=1	4T

### CONTROL INSTRUCTIONS

Sr.	Instruction	Bytes	Machine Cycle	T-States
1	NOP	1	F=1	4T
2	HLT	1	F+B=2	5T
3	DI	1	F=1	4T
4	EI	1	F=1	4T
5	RIM	1	F=1	4T
6	SIM	1	F=1	4T

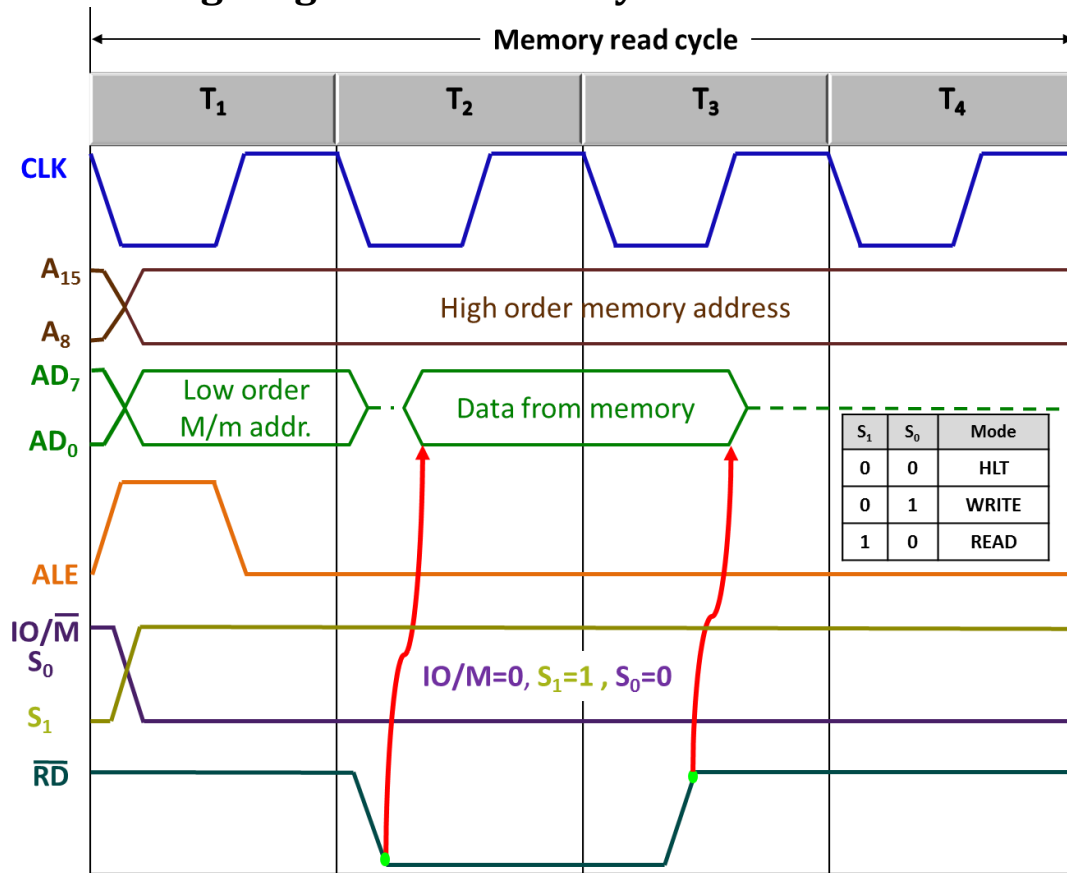
### 12. Draw Timing diagram for Opcode Fetch.

Ans.



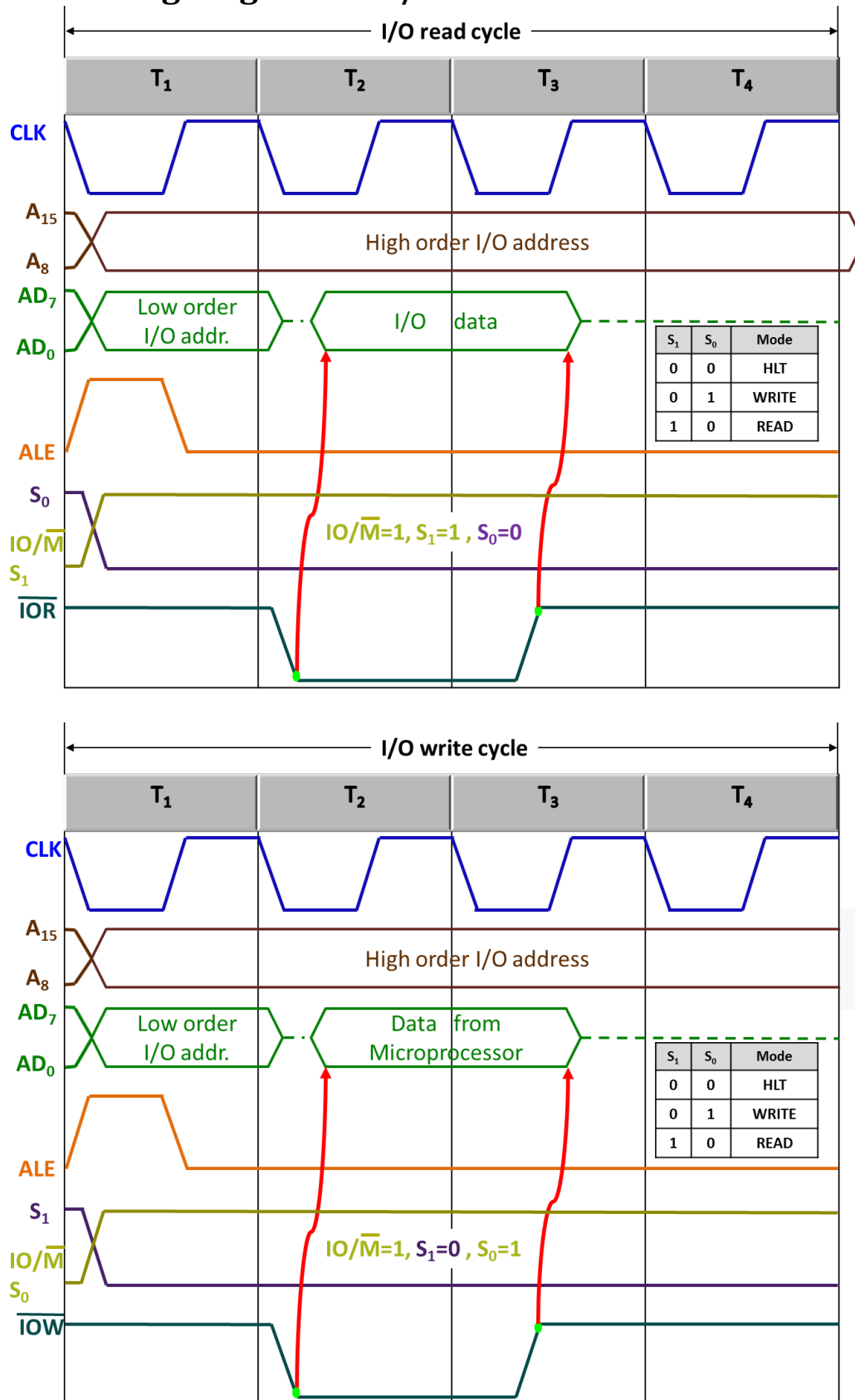
13. Draw Timing diagram for Memory read and write.

Ans.



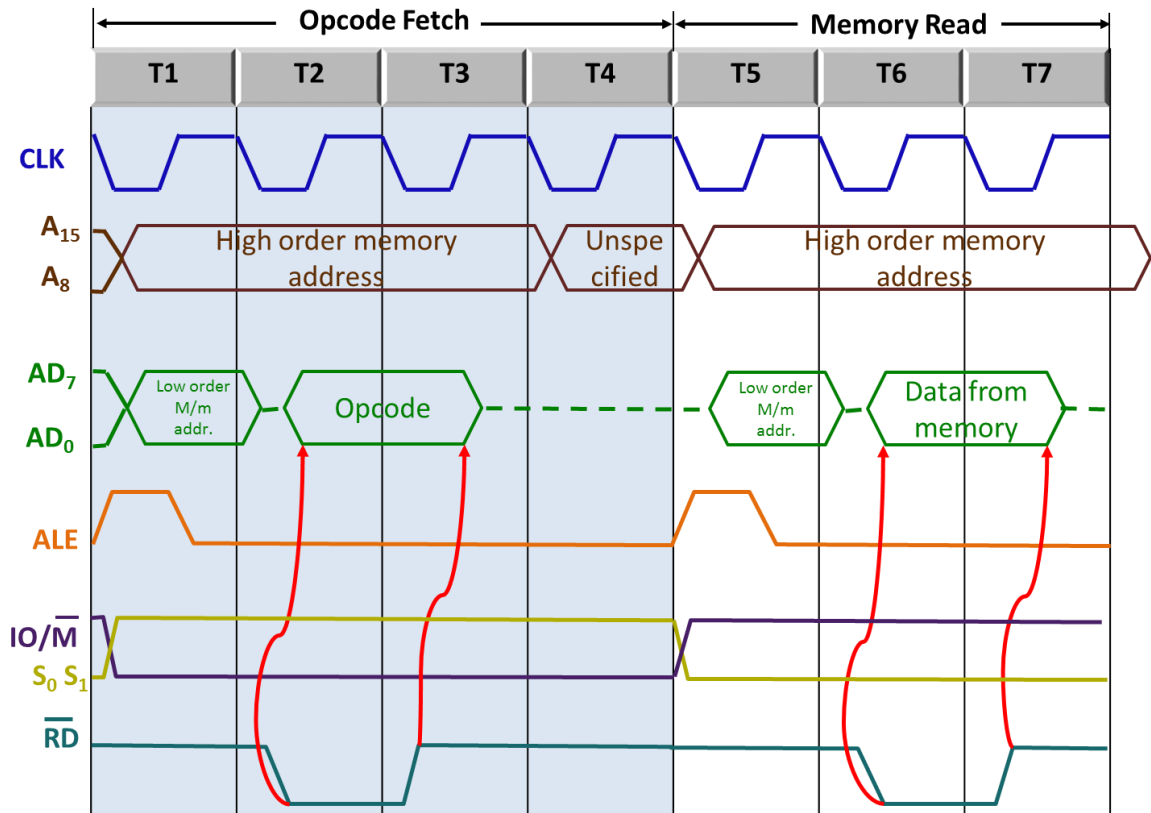
14. Draw Timing diagram for I/O read and write.

Ans.



**15. Draw Timing diagram for MOV B, M.**

**Ans.**

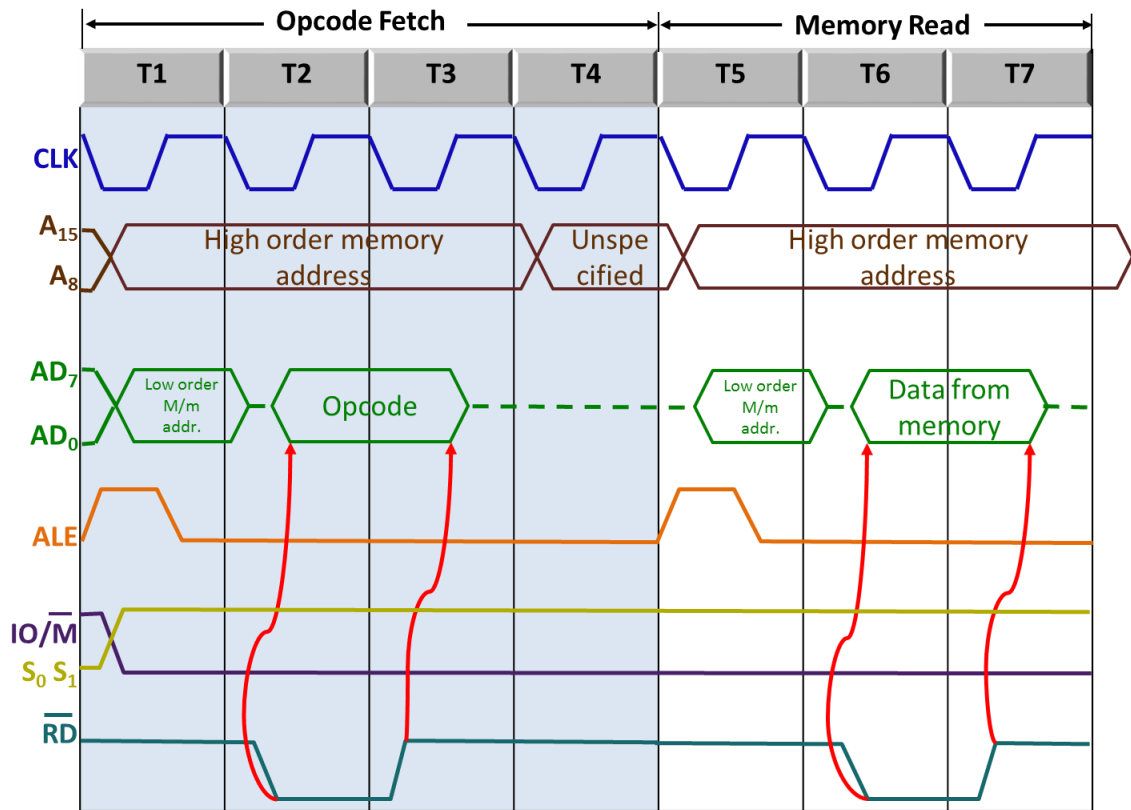


Instruction	N-Byte	Machine Cycle	Instruction Cycle	Addressing Mode
MOV B,M	1	F+MEMR = 2	4T+3T=7T	Indirect Addressing



**18. Draw Timing diagram for MVI A, 32H.**

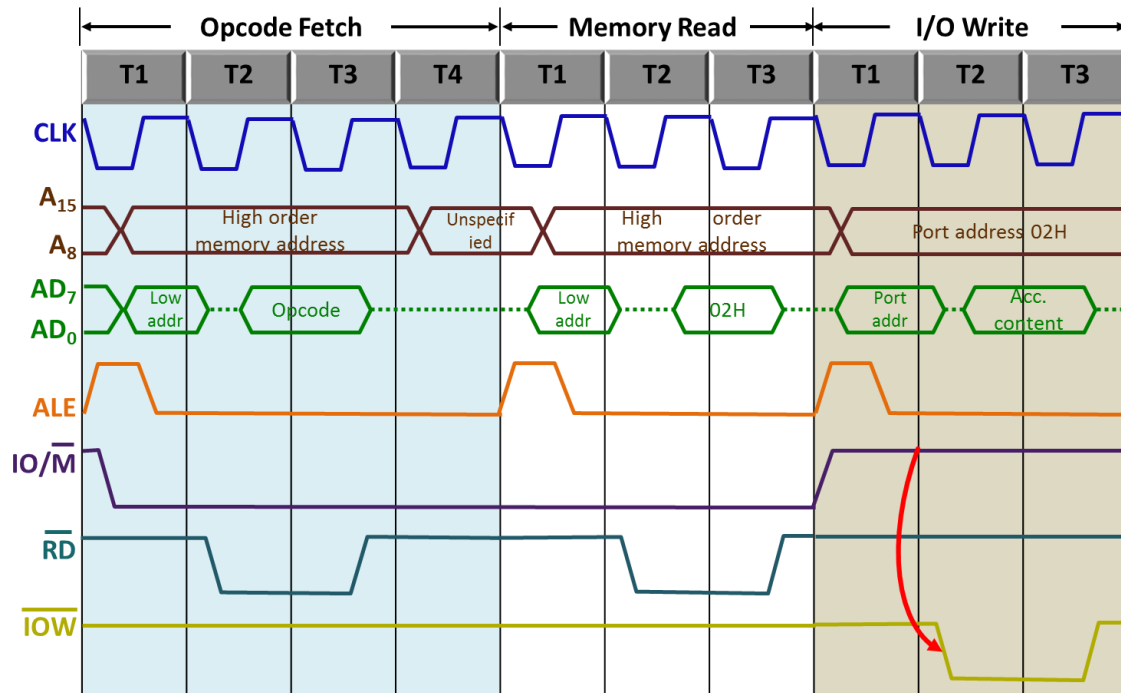
**Ans.**



Instruction	N-Byte	Machine Cycle	Instruction Cycle	Addressing Mode
OUT 02H	2	F+R+IOW=3	4T+3T+3T=10T	Direct Addressing

### 19. Draw Timing diagram for OUT 02H.

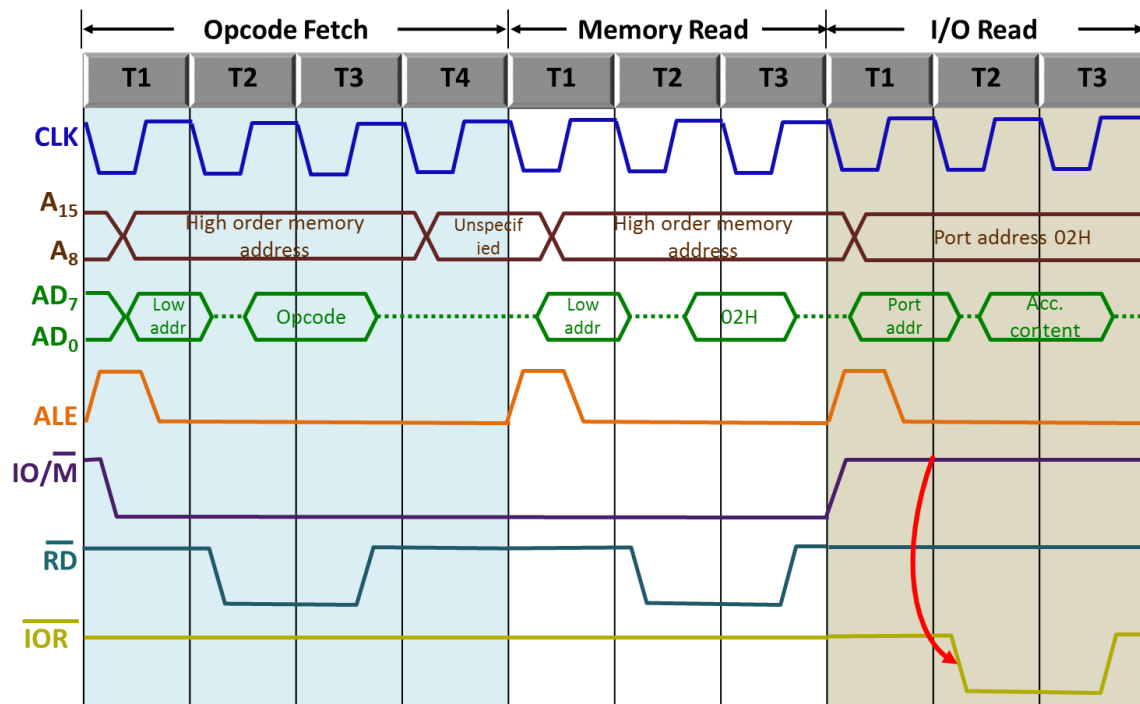
Ans.



Instruction	N-Byte	Machine Cycle	Instruction Cycle	Addressing Mode
OUT 02H	2	F+R+IOW=3	4T+3T+3T=10T	Direct Addressing

### 20. Draw Timing diagram for IN 02H.

Ans.



Instruction	N-Byte	Machine Cycle	Instruction Cycle	Addressing Mode
IN 02H	2	F+R+IOR=3	4T+3T+3T=10T	Direct Addressing