Microprocessor & Interfaces

(4CS3-04)

UNIT 2

Presented by: Shruti Sharma

Assistant Professor

Department of Computer Science Engineering

AIET, Jaipur

TOPIC:

INSTRUCTION SET OF 8085

Instruction Set of 8085

- An instruction is a binary pattern designed inside a microprocessor to perform a specific function.
- The entire group of instructions that a microprocessor supports is called Instruction Set.
- 8085 has 246 instructions.
- Each instruction is represented by an 8-bit binary value. These 8-bits of binary value is called Op-Code or Instruction Byte.

Classification of Instruction Set

- Data Transfer Instruction
- Arithmetic Instructions
- Logical Instructions
- Branching Instructions
- Control Instructions

Data Transfer Instruction

- These instructions move data between registers, or between memory and registers.
- These instructions copy data from source to destination(without changing the original data).

MOV-Copy from source to destination

Opcode	Operand
MOV	**Rd, Rs** **M, Rs** **Rd, M**

- This instruction copies the contents of the source register into the destination register(contents of the source register are not altered).
- If one of the operands is a memory location, its location is specified by the contents of the HL registers.

Example: MOV B,C or MOV B,M

BEFORE EXECUTION

A	20	B	

MOV B,A

AFTER EXECUTION

A	20	B	20

A		F	
B	30	C	
D		E	
H	20	L	50

MOV M,B

A		F	
B	30	C	
D		E	
H	20	L	50

30

A		F	
B		C	
D		E	
H	20	L	50

40

MOV C,M

A		F	
B		C	40
D		E	
H	20	L	50

40

MVI-Move immediate 8-bit

Opcode	Operand
MVI	Rd, Data M, Data

- 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 H-L registers.

Example: MVI B, 60H or MVI M, 40H

BEFORE EXECUTION

A		F
B		C
D		E
H		L

MVI B, 60_H

AFTER EXECUTION

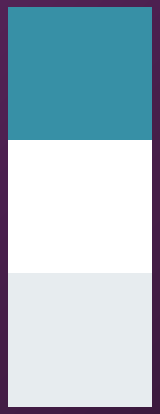
A		F
B	60	C
D		E
H		L

BEFORE EXECUTION

204FH

HL=2050

2051H



MVI M, 40_H

AFTER EXECUTION

204F

HL=2050

2051H

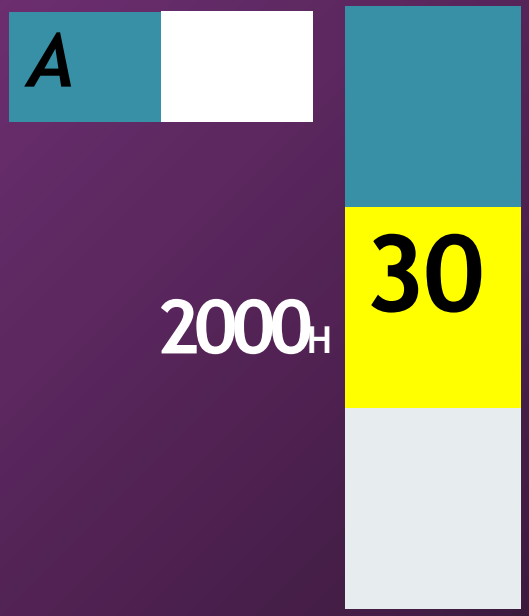
LDA-Load accumulator

Opcode	Operand
LDA	16-bit address

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

Example: LDA 2000_H

BEFORE EXECUTION



LDA 2000H

AFTER EXECUTION

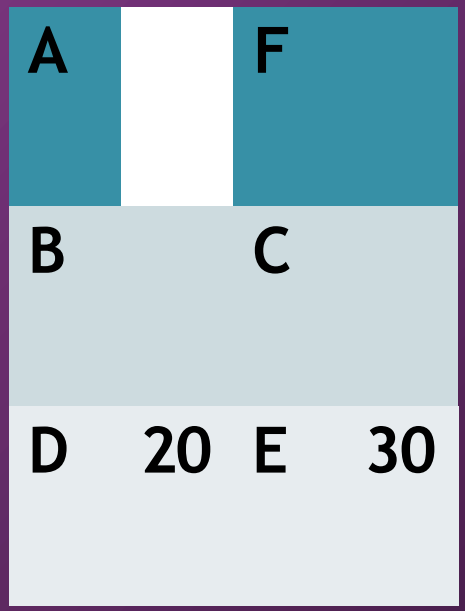
LDAX-Load accumulator indirect

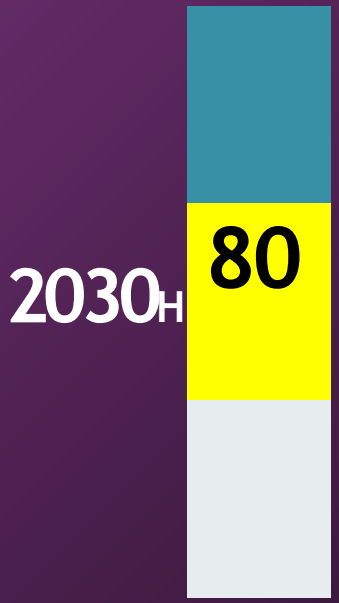
Opcode	Operand
LDAX	B/D Register Pair

- The contents of the designated register pair point to a memory location.
- This instruction copies the contents of that memory location into the accumulator.
- The contents of either the register pair or the memory location are not altered.

Example: LDAX D

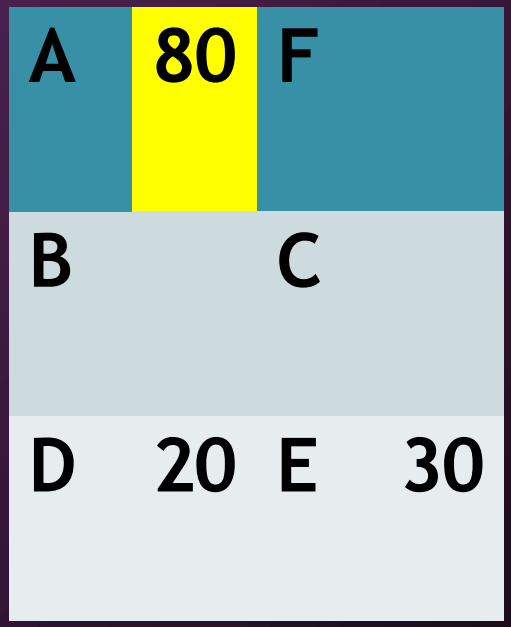
BEFORE EXECUTION





LDAX D

AFTER EXECUTION



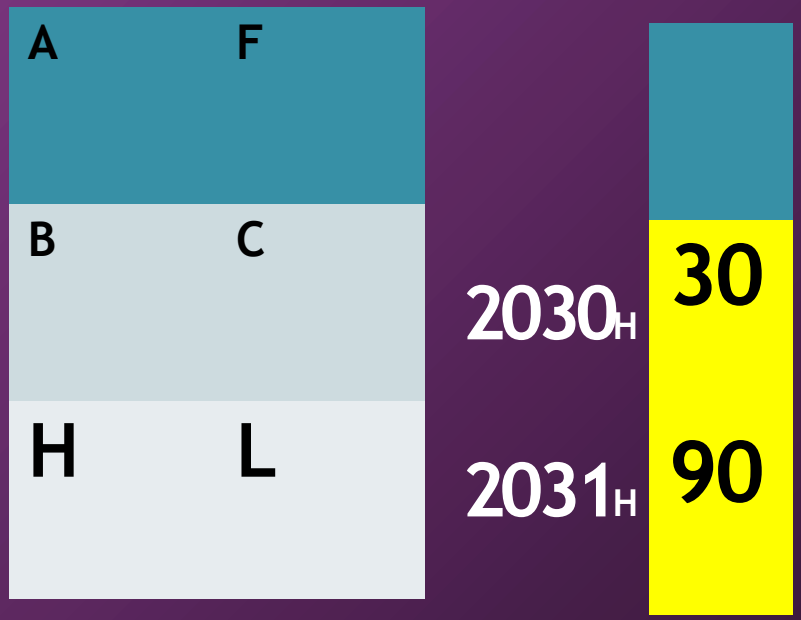
LXI-Load register pair immediate

Opcode	Operand
LXI	Reg. pair, 16-bit data

- This instruction loads 16-bit data in the register pair.

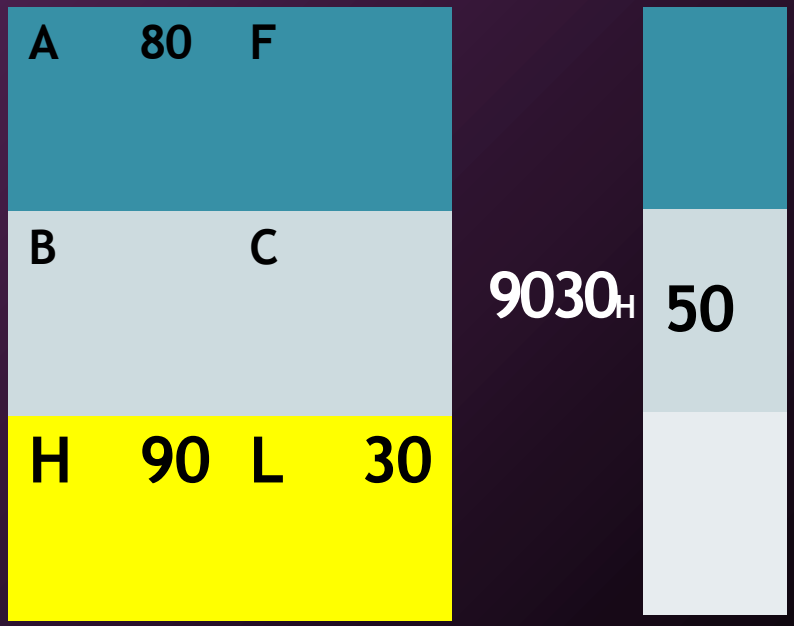
Example: LXI H, 2030 H

BEFORE EXECUTION



LXI H, 2030

AFTER EXECUTION



M=50

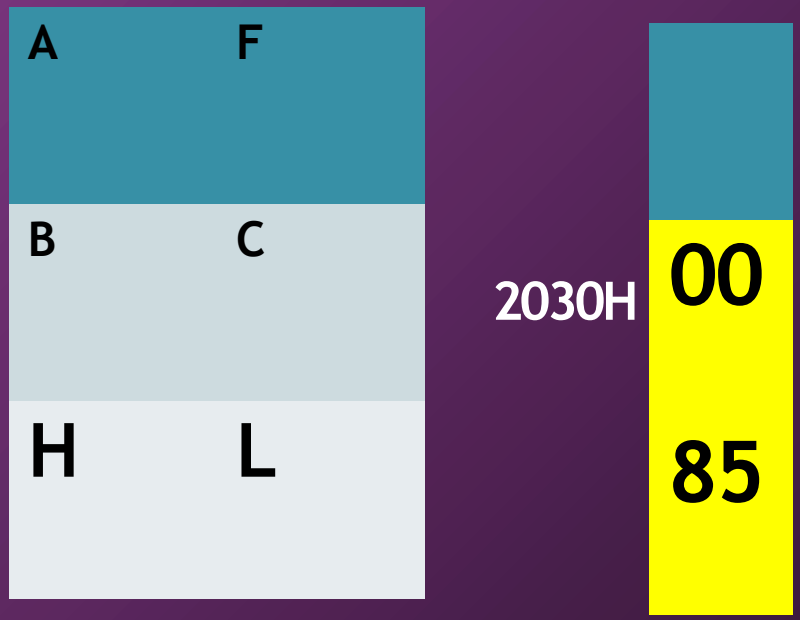
LHLD-Load H and L registers direct

Opcode	Operand
LHLD	16-bit address

- This instruction copies the contents of memory location pointed out by 16-bit address into register L.
- It copies the contents of next memory location into register H.

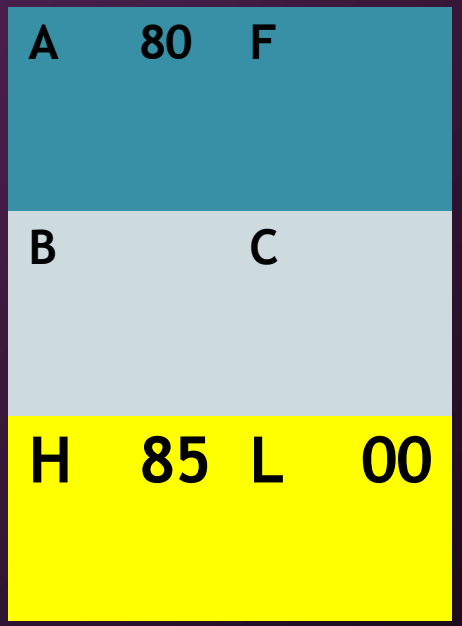
Example: LHLD 2030 H

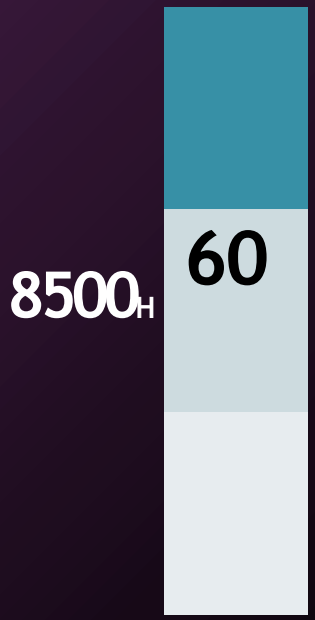
BEFORE EXECUTION



LHLD 2030

AFTER EXECUTION





M=60

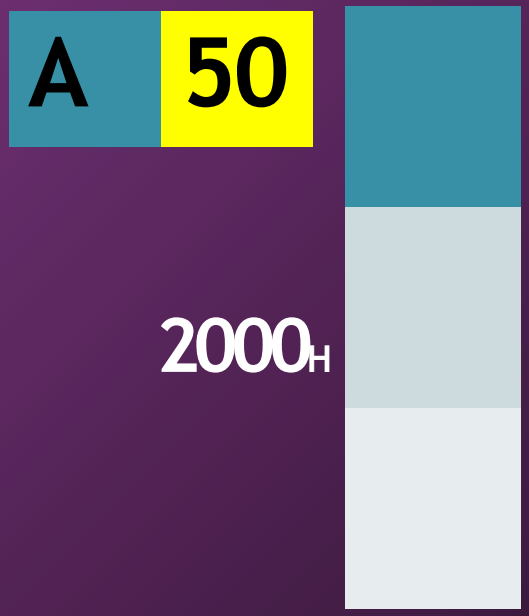
STA-Store accumulator direct

STA	16-bit address

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

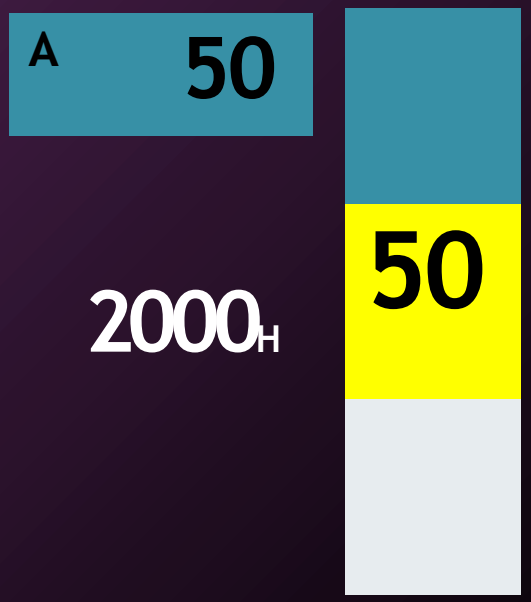
Example: STA 2000_H

BEFORE EXECUTION



STA
2000H

AFTER EXECUTION



STAX-Store accumulator indirect

Opcode	Operand
STAX	Reg. pair

- The contents of accumulator are copied into the memory location specified by the contents of the register pair.

Example: STAX B

BEFORE EXECUTION

AFTER EXECUTION

B	85	C	00
A = 1A$_H$			

STAX B

8500$_H$

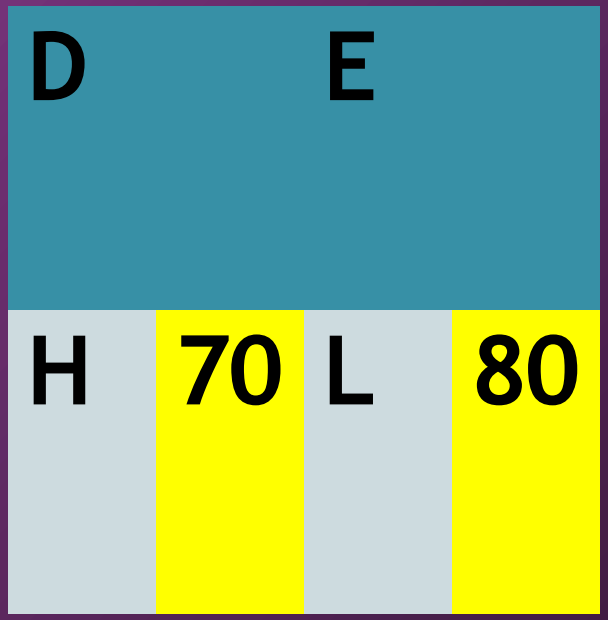
1A

SHLD-Store H and L registers direct

Opcode	Operand
SHLD	**16-bit address**

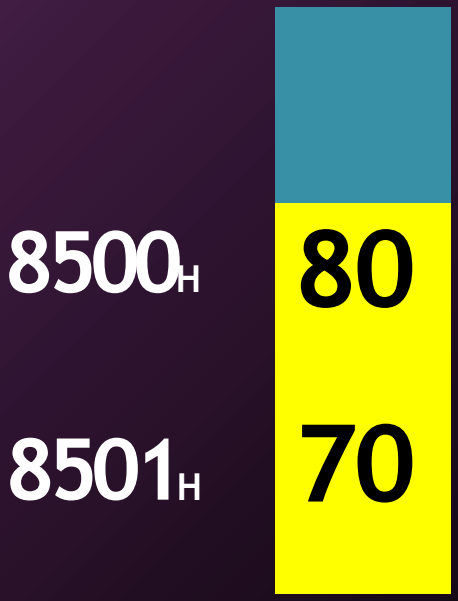
- The contents of register L are stored into memory location specified by the 16-bitaddress.
- The contents of register H are stored into the next memory location.
- **Example:** SHLD 2550_H

BEFORE EXECUTION



SHL
D
8500

AFTER EXECUTION



XCHG-Exchange H and L with D and E

Opcode	Operand
XCHG	None

- The contents of register H are exchanged with the contents of register D.
- The contents of register L are exchanged withthe contents of register E.

Example: XCHG

BEFORE EXECUTION

D	20	E	40
H	70	L	80

XCHG

AFTER EXECUTION

D	70	E	80
H	20	L	40

SPHL-Copy H and L registers to the stack Pointer

Opcode	Operand
SPHL	None

- This instruction loads the contents of H-L pair into SP.

Example: SPHL

BEFORE EXECUTION

SP			
H	25	L	00

SPHL

AFTER EXECUTION

SP	2500	
H	25	L 00

XTHL- Exchange H and L with top of stack

Opcode	Operand
XTHL	None

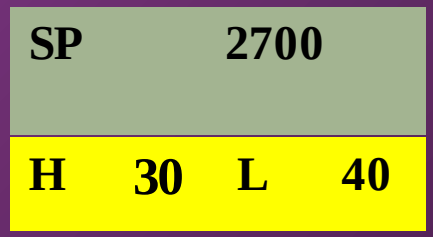
- The contents of L register are exchanged withthe location pointed out by the contents of the SP.
- The contents of H register are exchanged with the next location (SP + 1).

Example: XTHL

L=SP

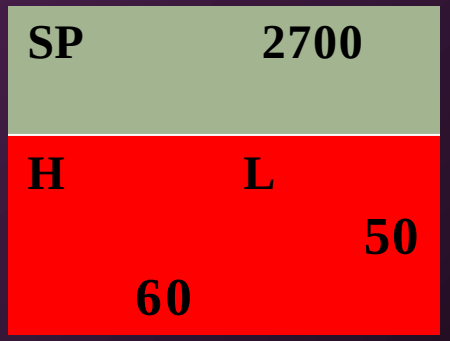
H=(SP+1)

BEFORE EXECUTION



XTHL

AFTER EXECUTION



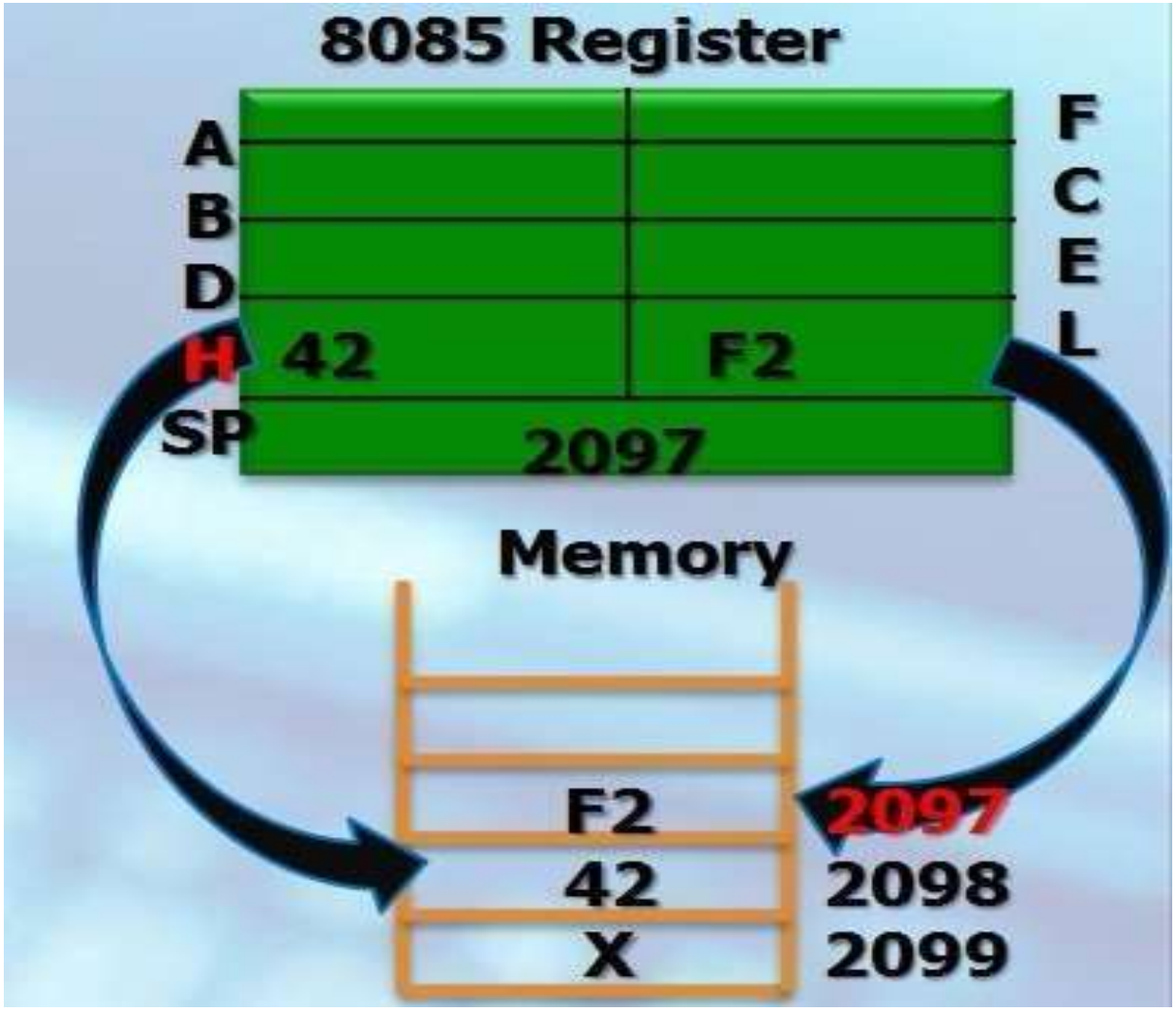
PUSH-Push register pair onto stack

Opcode	Operand
PUSH	Reg. pair

- The contents of register pair are copied onto stack.
- SP is decremented and the contents of high-order registers (B, D, H, A) are copied into stack.
- SP is again decremented and the contents of low-order registers (C, E, L, Flags) are copied into stack.

Example: PUSH B

PUSH H


8085 Register
A
B
D
H
SP
42
F2
2097
F
C
E
L
Memory
F2
42
X
2097
2098
2099

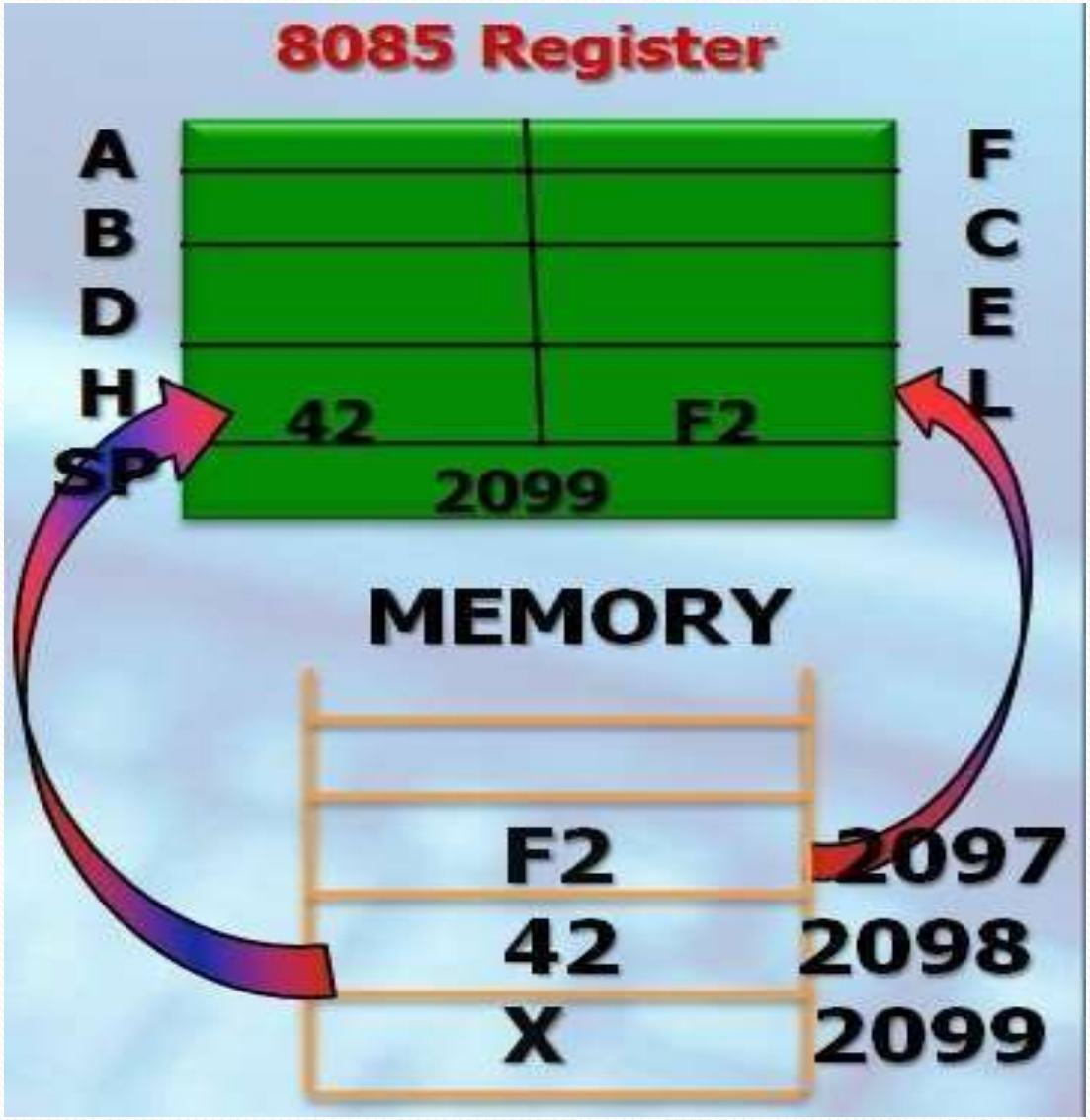
POP

Opcode	Operand
POP	Reg. pair

- The contents of **top of stack** are **copied into register pair**.
- The contents of location pointed out by SP are copied to the low-order register (C, E, L, Flags).
- **SP is incremented and the contents of locationare copied** to the high-order register (B, D, H, A).

Example: POP H

POP H


8085 Register
A
B
D
H
SP
F
C
E
L
42
F2
2099
MEMORY
F2
42
X
2097
2098
2099

IN- Copy data to accumulator from a port with 8-bit address

Opcode	Operand
IN	8-bit port address

- The contents of I/O port are copied into accumulator.

Example: IN $8C_H$

BEFORE EXECUTION

IN 80_H

AFTEREXECUTION

OUT- Copy data from accumulator to a port

Opcode	Operand
OUT	8-bit port address

- The contents of accumulator are copied into the I/O port.

Example: OUT 78_H

BEFORE EXECUTION

OUT 50H

AFTEREXECUTION

2.Arithmetic Instructions

These instructions perform the operations like:

- Addition
- Subtract
- Increment
- Decrement

Addition

- Any 8-bit number, or the contents of register, or the contents of memory location can be added to the contents of accumulator.
- The result (sum) is stored in the accumulator.
- No two other 8-bit registers can be added directly.

Example: The contents of register B cannot be added directly to the contents of register C.

ADD

Opcode	Operand	Description
ADD	R M	Add register or memory to accumulator

- The contents of register or memory are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.

Example: ADD B or ADD M

BEFOREEXECUTION

A	04		
B		C	05
D		E	
H		L	

ADD C

A=A+C

AFTEREXECUTION

A.	09		
B.		C	05
D		E	
H		L	

04+05=09

BEFORE EXECUTION

A	04		
B		C	
D		E	
H	20	L	50

ADD M

A=A+M

10

2050

AFTEREXECUTION

A	14		
B		C	
D		E	
H	20	L	50

10

04+10=14

2050

ADC

Opcode	Operand	Description
ADC	R M	Add register or memory to accumulator with carry

- The contents of register or memory and Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of the addition.

Example: ADC B or ADC M

BEFOREEXECUTION

CY	01
A	50

B		C	05
D		E	
H		L	

ADCC
A=A+C+CY

AFTEREXECUTION

A	56

B		C	20
D		E	
H		L	

50+05+01=56

BEFOREEXECUTION

CY	1
A	06

2050H

30

H	20	L	50

ADCM
A=A+M+CY

AFTEREXECUTION

A	37

2050H

30

H	20	L	50

06+1+30=37

ADI

Opcode	Operand	Description
ADI	8-bit data	Add immediate to accumulator

- The 8-bit data is added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.

Example: ADI 45 H

BEFOREEXECUTION

AFTEREXECUTION

ADI 05_H

A=A+DATA(8)

03+05=08

ACI

Opcode	Operand	Description
ACI	8-bit data	Add immediate to accumulator with carry

- The 8-bit data and the Carry Flag (CY) are added to the contents of accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of the addition.

Example: ACI 45H

BEFOREEXECUTION

CY	1
A	05

ACI 20_H

A=A+DATA (8)+CY

AFTEREXECUTION

A	26

05+20+1=26

DAD

Opcode	Operand	Description
DAD	Reg. pair	Add register pair to H-L pair

- The 16-bit contents of the register pair are added to the contents of H-L pair.
- The result is stored in H-L pair.
- If the result is larger than 16 bits, then CY is set. No other flags are changed.

Example: DAD B or DAD D

BEFOREEXECUTION

D	12	E	34
H	23	L	45

DAD D

AFTEREXECUTION

D	12	E	34
H	35	L	79

1234
2345 +

3579

DAD D HL=HL+DE
DAD B HL=HL+BC

Subtraction

- Any 8-bit number, or the contents of register, or the contents of memory location can be subtracted from the contents of accumulator.
- The result is stored in the accumulator.
- Subtraction is performed in 2's complement form.
- If the result is negative, it is stored in 2's complement form.
- No two other 8-bit registers can be subtracted directly.

SUB

Opcode	Operand	Description
SUB	R M	Subtract register or memory from accumulator

- The contents of the register or memory location are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.

Example: SUB B or SUB M

BEFOREEXECUTION

A	09		
B		C	04
D		E	
H		L	

SUBC
A=A-C

AFTEREXECUTION

A.	05		
B.		C	04
D		E	
H		L	

09-04=05

BEFORE EXECUTION

A	14		
B		C	
D		E	
H	20	L	50

SUBM
A=A-M

10

2050

AFTEREXECUTION

A	04		
B		C	
D		E	
H	20	L	50

10

14-10=04

2050

SBB

Opcode	Operand	Description
SBB	R M	Subtract register or memory from accumulator with borrow

- The contents of the register or memory location and Borrow Flag (i.e. CY) are subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- If the operand is memory location, its address is specified by H-L pair.
- All flags are modified to reflect the result of subtraction.

Example: SBB B or SBB M

BEFOREEXECUTION

CY	01
A	08

B		C	05
D		E	
H		L	

SBBC
A=A-C-CY

AFTEREXECUTION

A	02

B		C	05
D		E	
H		L	

08-05-01=02

BEFOREEXECUTION

CY	1
A	06

2050H

02

H	20	L	50

SBBM
A=A-M-CY

AFTEREXECUTION

A	03

2050H

02

H	20	L	50

06-02-1=03

SUI

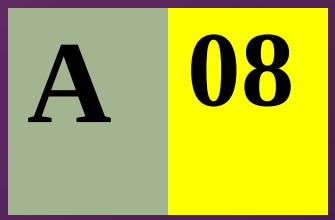
Opcode	Operand	Description
SUI	8-bit data	Subtract immediate from accumulator

- The 8-bit data is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.

Example: SUI 05_H

BEFOREEXECUTION

AFTEREXECUTION



SUI 05H

A=A-DATA(8)

08-05=03

SBI

Opcode	Operand	Description
SBI	8-bit data	Subtract immediate from accumulator with borrow

- The 8-bit data and the Borrow Flag (i.e. CY) is subtracted from the contents of the accumulator.
- The result is stored in accumulator.
- All flags are modified to reflect the result of subtraction.

Example: SBI 45 H

BEFOREEXECUTION

CY	1
A	25

SBI 20_H

A=A-DATA

(8)-CY

AFTEREXECUTION

A	04

25-20-01=04

Increment / Decrement

- The 8-bit contents of a register or a memory location can be incremented or decremented by 1.
- The 16-bit contents of a register pair can be incremented or decremented by 1.
- Increment or decrement can be performed on any register or a memory location.

Increment

Opcode	Operand	Description
INR	R M	Increment register or memory by 1

- The contents of register or memory location are incremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.

Example: INR B or INR M

Increment

Opcode	Operand	Description
INX	R	Increment register pair by 1

- The contents of register pair are incremented by 1.
- The result is stored in the same place.

Example: INX H

Decrement

Opcode	Operand	Description
DCR	R M	Decrement register or memory by 1

- The contents of register or memory location are decremented by 1.
- The result is stored in the same place.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- Example: DCR B or DCR M

Decrement

Opcode	Operand	Description
DCX	R	Decrement register pair by 1

- The contents of register pair are decremented by 1.
- The result is stored in the same place.
- Example: DCX H

Logical Instructions

- These instructions perform logical operations on data stored in registers, memory and status flags.

- The logical operations are:
  - AND
  - OR
  - XOR
  - Rotate
  - Compare
  - Complement

AND, OR, XOR

- Any 8-bit data, or the contents of register, or memory location can logically have
  - AND operation
  - OR operation
  - XOR operation

  with the contents of accumulator.
- The result is stored in accumulator.

Rotate

- Each bit in the accumulator can be shifted either left or right to the next position.

Compare

- Any 8-bit data, or the contents of register, or memory location can be compares for:
  - Equality
  - Greater Than
  - Less Than

  with the contents of accumulator.
- The result is reflected in status flags.

Complement

- The contents of accumulator can be complemented.
- Each 0 is replaced by 1 and each 1 is replaced by 0.

Logical Instructions

Opcode	Operand	Description
ANA	R M	Logical AND register or memory with accumulator

- The contents of the accumulator are logically ANDed with the contents of register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- **Example:** ANA B or ANA M.

Logical Instructions

Opcode	Operand	Description
ANI	8-bit data	Logical AND immediate with accumulator

- The contents of the accumulator are logically ANDed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- **Example:** ANI 86H.

Logical Instructions

Opcode	Operand	Description
ORA	R M	Logical OR register or memory with accumulator

- The contents of the accumulator are logically ORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result.
- **Example:** ORA B or ORA M.

Logical Instructions

Opcode	Operand	Description
ORI	8-bit data	Logical OR immediate with accumulator

- The contents of the accumulator are logically ORed with the 8- bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- **Example:** ORI 86H.

Logical Instructions

Opcode	Operand	Description
XRA	R M	Logical XOR register or memory with accumulator

- The contents of the accumulator are XORed with the contents of the register or memory.
- The result is placed in the accumulator.
- If the operand is a memory location, its address is specified by the contents of H-L pair.
- S, Z, P are modified to reflect the result of the operation.
- **Example:** XRA B or XRA M.

Logical Instructions

Opcode	Operand	Description
XRI	8-bit data	XOR immediate with accumulator

- The contents of the accumulator are XORed with the 8-bit data.
- The result is placed in the accumulator.
- S, Z, P are modified to reflect the result.
- **Example:** XRI 86H.

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

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

Logical Instructions

Opcode	Operand	Description
CMP	R M	Compare register or memory with accumulator

- if (A) < (reg/mem): carry flag is set
- if (A) = (reg/mem): zero flag is set
- if (A) > (reg/mem): carry and zero flags are reset.
- **Example:** CMP B or CMP M

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- The 8-bit data is compared with the contents of accumulator.
- The values being compared remain unchanged.
- The result of the comparison is shown by setting the flags of the PSW as follows:

Logical Instructions

Opcode	Operand	Description
CPI	8-bit data	Compare immediate with accumulator

- if (A) < data: carry flag is set
- if (A) = data: zero flag is set
- if (A) > data: carry and zero flags are reset
- **Example:** CPI 89H

Logical Instructions

Opcode	Operand	Description
RLC	None	Rotate accumulator left

- Each binary bit of the accumulator is rotated left by one position.
- Bit D7 is placed in the position of D0 as well as in the Carry flag.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RLC.

Logical Instructions

Opcode	Operand	Description
RRC	None	Rotate accumulator right

- Each binary bit of the accumulator is rotated right by one position.
- Bit D0 is placed in the position of D7 as well as in the Carry flag.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RRC.

Logical Instructions

Opcode	Operand	Description
RAL	None	Rotate accumulator left through carry

- Each binary bit of the accumulator is rotated left by one position through the Carry flag.
- Bit D7 is placed in the Carry flag, and the Carry flag is placed in the least significant position D0.
- CY is modified according to bit D7.
- S, Z, P, AC are not affected.
- **Example:** RAL.

Logical Instructions

Opcode	Operand	Description
RAR	None	Rotate accumulator right through carry

- Each binary bit of the accumulator is rotated right by one position through the Carry flag.
- Bit D0 is placed in the Carry flag, and the Carry flag is placed in the most significant position D7.
- CY is modified according to bit D0.
- S, Z, P, AC are not affected.
- **Example:** RAR.

Logical Instructions

Opcode	Operand	Description
CMA	None	Complement accumulator

- The contents of the accumulator are complemented.
- No flags are affected.
- **Example:** CMA.

Logical Instructions

Opcode	Operand	Description
CMC	None	Complement carry

- The Carry flag is complemented.
- No other flags are affected.
- **Example:** CMC.

Branching Instructions

- The branching instruction alter the normal sequential flow.
- These instructions alter either unconditionally or conditionally.

Branching Instructions

Opcode	Operand	Description
JMP	16-bit address	Jump unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- **Example:** JMP 2034 H.

Branching Instructions

Opcode	Operand	Description
Jx	16-bit address	Jump conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- **Example:** JZ 2034 H.

Jump Conditionally

Opcode	Description	Status Flags
JC	Jump if Carry	CY = 1
JNC	Jump if No Carry	CY = 0
JP	Jump if Positive	S = 0
JM	Jump if Minus	S = 1
JZ	Jump if Zero	Z = 1
JNZ	Jump if No Zero	Z = 0
JPE	Jump if Parity Even	P = 1
JPO	Jump if Parity Odd	P = 0

Branching Instructions

Opcode	Operand	Description
CALL	16-bit address	Call unconditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand.
- Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.
- **Example:** CALL 2034 H.

Branching Instructions

Opcode	Operand	Description
Cx	16-bit address	Call conditionally

- The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW.
- Before the transfer, the address of the next instruction after the call (the contents of the program counter) is pushed onto the stack.
- **Example:** CZ 2034 H.

Call Conditionally

Opcode	Description	Status Flags
CC	Call if Carry	CY = 1
CNC	Call if No Carry	CY = 0
CP	Call if Positive	S = 0
CM	Call if Minus	S = 1
CZ	Call if Zero	Z = 1
CNZ	Call if No Zero	Z = 0
CPE	Call if Parity Even	P = 1
CPO	Call if Parity Odd	P = 0

Branching Instructions

Opcode	Operand	Description
RET	None	Return unconditionally

- The program sequence is transferred from the subroutine to the calling program.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example:** RET.

Branching Instructions

Opcode	Operand	Description
Rx	None	Call conditionally

- The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW.
- The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.
- **Example:** RZ.

Return Conditionally

Opcode	Description	Status Flags
RC	Return if Carry	CY = 1
RNC	Return if No Carry	CY = 0
RP	Return if Positive	S = 0
RM	Return if Minus	S = 1
RZ	Return if Zero	Z = 1
RNZ	Return if No Zero	Z = 0
RPE	Return if Parity Even	P = 1
RPO	Return if Parity Odd	P = 0

Branching Instructions

Opcode	Operand	Description
RST	0 – 7	Restart (Software Interrupts)

- The RST instruction jumps the control to one of eight memory locations depending upon the number.
- These are used as software instructions in a program to transfer program execution to one of the eight locations.
- **Example:** RST 3.

Restart Address Table

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

Control Instructions

- The control instructions control the operation of microprocessor.

Control Instructions

Opcode	Operand	Description
NOP	None	No operation

- No operation is performed.
- The instruction is fetched and decoded but no operation is executed.
- **Example:** NOP

Control Instructions

Opcode	Operand	Description
HLT	None	Halt

- The CPU finishes executing the current instruction and halts any further execution.
- An interrupt or reset is necessary to exit from the halt state.
- **Example:** HLT

Control Instructions

Opcode	Operand	Description
DI	None	Disable interrupt

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

Control Instructions

Opcode	Operand	Description
EI	None	Enable interrupt

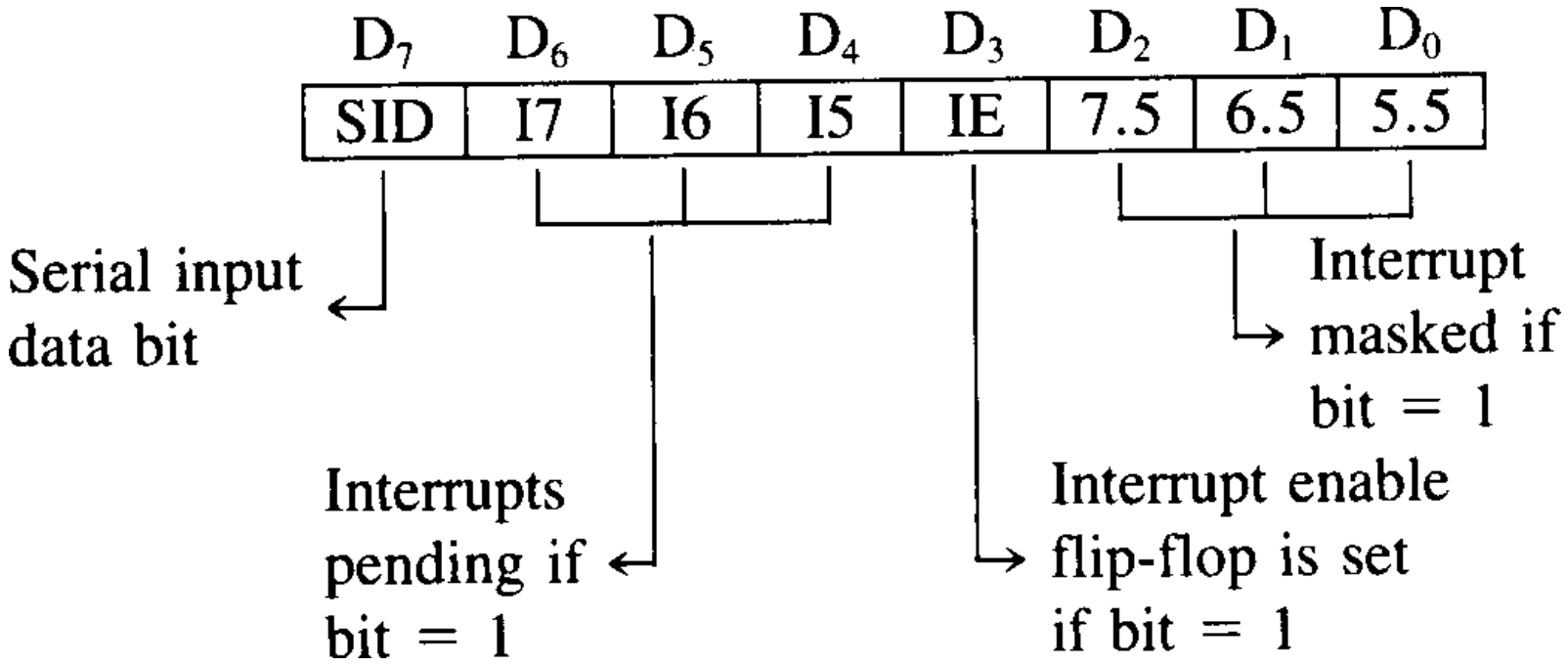
- The interrupt enable flip-flop is set and all interrupts are enabled.
- No flags are affected.
- This instruction is necessary to re-enable the interrupts (except TRAP).
- **Example:** EI

Control Instructions

Opcode	Operand	Description
RIM	None	Read Interrupt Mask

- This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit.
- The instruction loads eight bits in the accumulator with the following interpretations.
- **Example:** RIM

RIM
Instruction

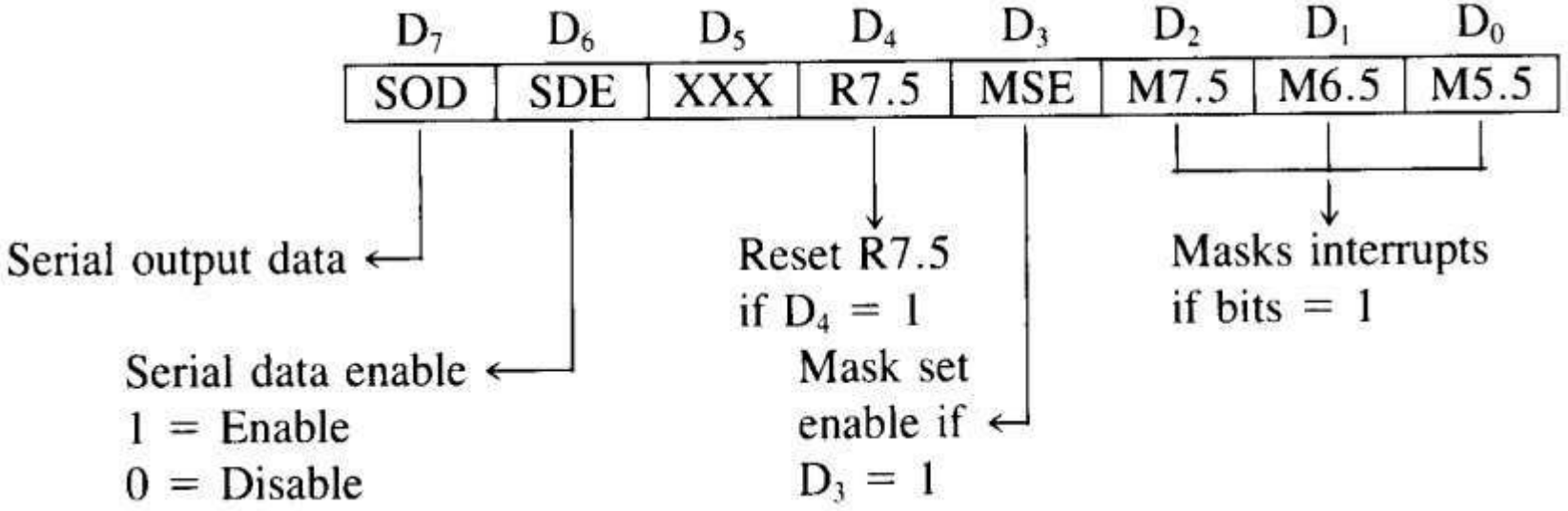


Control Instructions

Opcode	Operand	Description
SIM	None	Set Interrupt Mask

- This is a multipurpose instruction and used to implement the 8085 interrupts 7.5, 6.5, 5.5, and serial data output.
- The instruction interprets the accumulator contents as follows.
- Example: SIM

SIM Instruction



TOPIC :

ADDRESSING MODE

Addressing Modes of 8085

- To perform any operation, we have to give the corresponding instructions to the microprocessor.
- In each instruction, programmer has to specify 3 things:

1. Operation to be performed i.e. Opcode
2. Address of source of data i.e. Operands
3. Address of destination of result.

Definition

- The various ways of specifying operands for an instruction are called the addressing modes.
- The various addressing modes that are defined in a given instruction set architecture each instruction.
- In any microprocessor instruction, the source and destination operands can be :
  - Register
  - Memory Location
  - 8-bit number

The method by which the address of source of data and the address of the destination of the result is given in the instruction are called the Addressing modes.

8085 uses the following addressing modes:

1. Implied/Implicit Addressing Mode
2. Immediate Addressing Mode
3. Register Addressing Mode
4. Direct Addressing Mode
5. Indirect Addressing Mode

Implied/Implicit Addressing Mode

In the implicit addressing mode instructions, the opcode itself specifies the address of the operands.

Since there is no operand in this type of instructions, the length of the instructions is always 1 byte.

Example:

- CMA
- RAR
- XCHG

Immediate Addressing Mode

In the Immediate addressing mode instructions, the Immediate data of 8 –bit or 16-bit specified as a part of instruction.

In 8085 microprocessor the instructions having ‘I’ letter as the last alphabet of the opcode falls under this category.

Example:

- MVI 30H
- SUI 51H
- ANI 7AH

Register Addressing Mode

The Register addressing mode specifies that the source operand, destination operand or both the operand are stored in register or register pair.

The Register addressing mode instructions are faster in execution because there is no need to access memory for operands.

Example:

- MOV B,C
- ADD B
- XRA A

Direct Addressing Mode

In the Direct addressing mode instructions, the address of the data is directly given in to the instruction using it's Hex representation.

Example:

- STA 3000H
- SHLD 2000H
- IN 02

Indirect Addressing Mode

If the address of the data is stored in a register pair and the name of the register pair is shown in the instruction then the instruction is having Indirect addressing mode.

Example:

- STAX B
- LDAX D
- CMP M

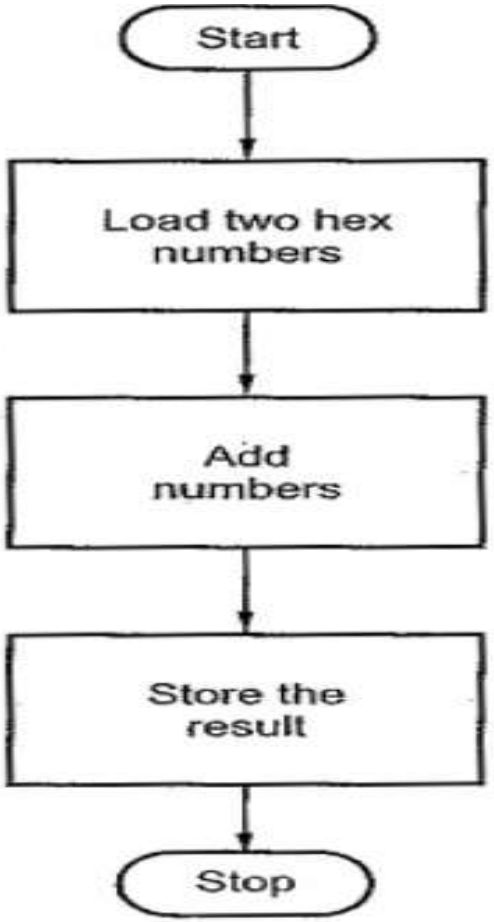
Program Structure

A program is a collection of instructions written in a sequence to perform a specific task.
Program can have two types of structure:

1. Sequential Program
2. Iterative Program

Sequential Program

The instruction of a sequential program are executed one after one without any change in the sequence. Meaning that there is no branch instruction in the program.



Iterative Program

The program in which a segment of program is repeated to complete task or to solve a typical problem, is known as iterative program.

The iterative program structure can be further classified in to following:

1. Conditional iterative program
2. Unconditional iterative program

TOPIC:

PROGRAMMING TECHNIQUES

Programming Techniques

the program is an implementation of certain logic by executing group of instructions. To implement program logic we need to take help of some common Programming Techniques in Microprocessor 8085 such as:

1. Indexing
2. Counting
3. Looping

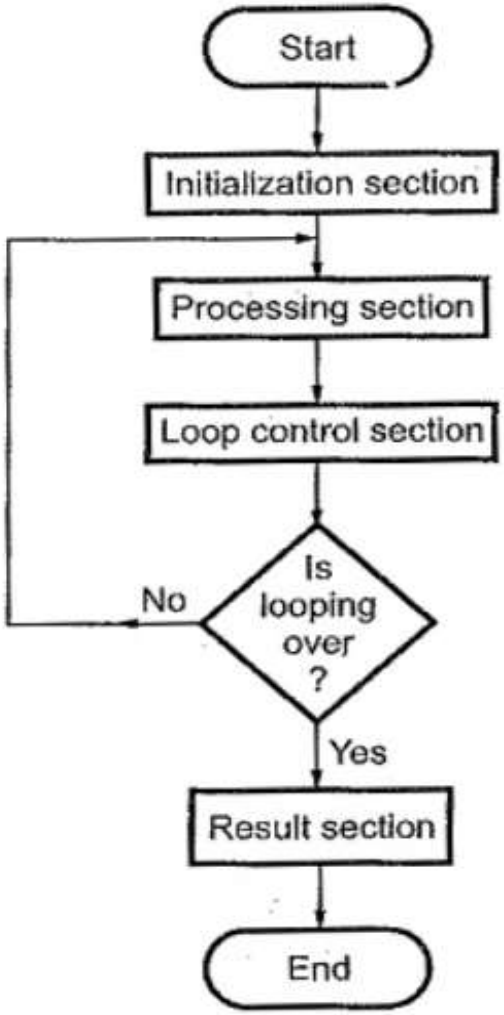
Programming Techniques

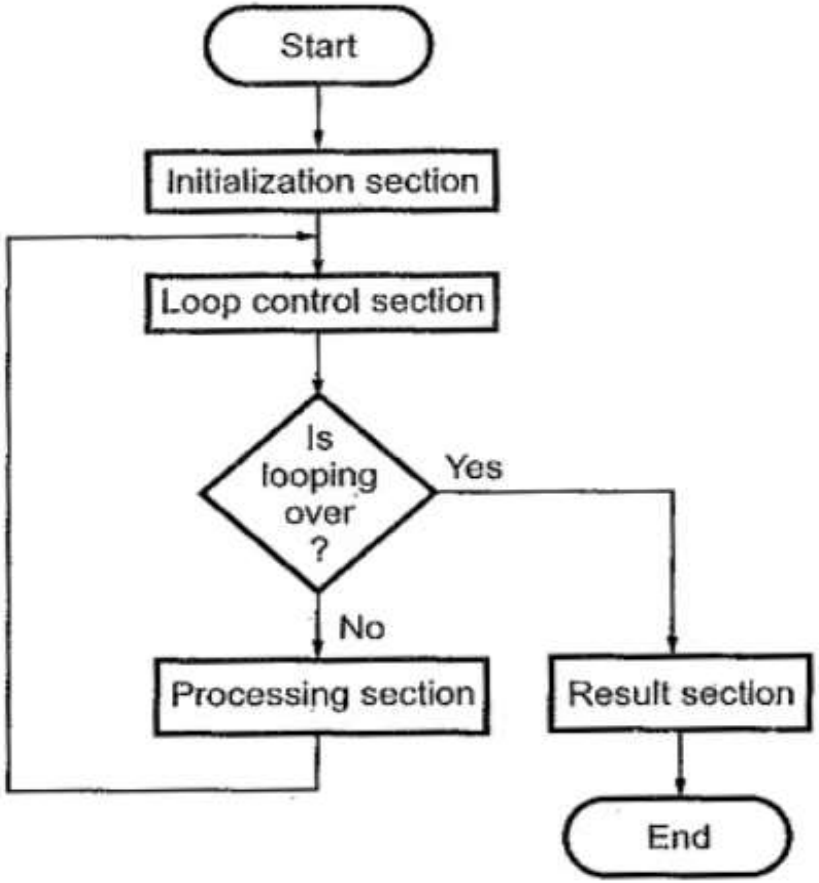
- **Indexing :** This Programming Techniques in Microprocessor 8085 allows programmer to point or refer the data stored in sequential memory locations one by one.
- **Counting :** This technique allows programmer to count how many times the instruction/set of instructions are executed.

Programming Techniques

- **Looping :** In this Programming Techniques in Microprocessor 8085, the Program is instructed to execute certain set of instructions repeatedly to execute a particular task number of times. For example, to add ten numbers stored in the consecutive memory locations we have to perform addition ten times.

Flowchart


Start
Initialization section
Processing section
Loop control section
Is looping over ?
No
Yes
Result section
End


Start
Initialization section
Loop control section
Is looping over ?
Yes
No
Processing section
Result section
End

TOPIC:

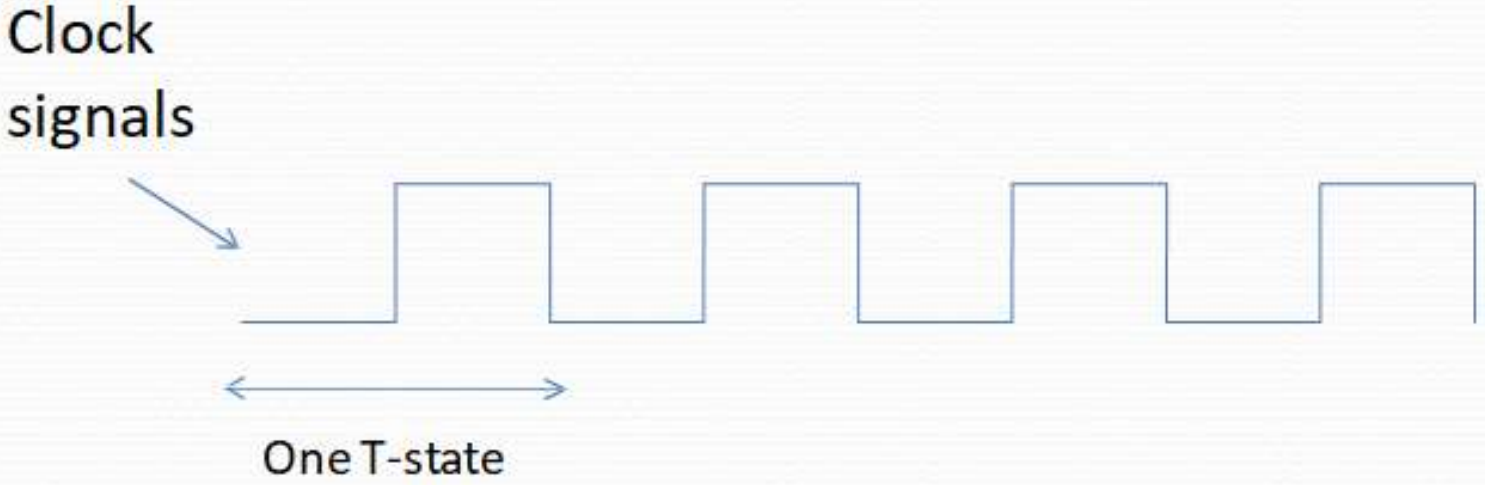
TIMING DIAGRAM

Timing Diagram

- A timing diagram in the field of embedded systems refers to a graphical representation of processes occurring with respect to time.
- In other words, the representation of the changes and variations in the status of signals with respect to time is referred to as a timing diagram.
- The timing diagram represents the clock cycle and duration, delay, content of address bus and data bus, type of operation ie. Read/write/status signals.

T-States

One clock period is called a T-state.



Time period of clock

Instruction cycle

An instruction cycle can be define as total time required by the microprocessor to completely fetch and execute an instruction.

An instruction cycle is comprised of one or more number of machine cycles.

Machine Cycle

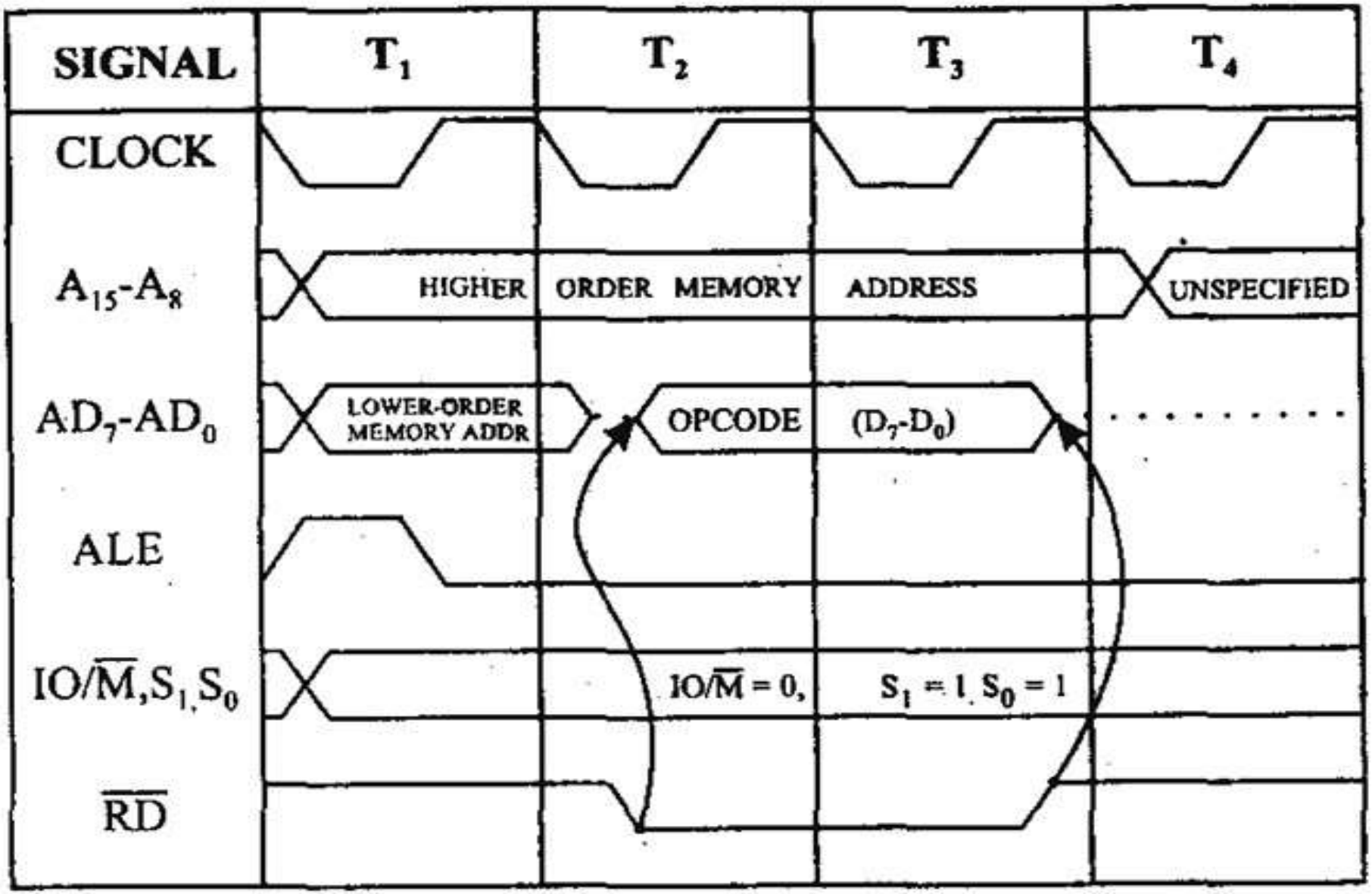
- Machine cycle can be define as time required by the microprocessor to complete an operation of accessing memory or input / output devices.
- A machine cycle is comprise of some member of T-states.

There are following 9 types of machine cycle which the microprocessor can undergo:

1. OP code fetch cycle
2. Memory read cycle
3. Memory write cycle
4. Input / Output read cycle
5. Input / Output write cycle
6. Interrupt Acknowledge
7. Halt
8. Hold
9. Reset

Machine Cycle	Status			No. of Machine cycles	Control
	IO/$\overline{M}$	S1	S0		
Opcode Fetch	0	1	1	4	$\overline{RD}$ =0
Memory Read	0	1	0	3	$\overline{RD}$ =0
Memory Write	0	0	1	3	$\overline{WR}$ =0
I/O Read	1	1	0	3	$\overline{RD}$ =0
I/O Write	1	0	1	3	$\overline{WR}$ =0
INTR Acknowledge	1	1	1	3	$\overline{INTA}$ =0

1.Opcode fetch cycle(4T or 6T)



OPCODE FETCH

- The Opcode fetch cycle, fetches the instructions from memory and delivers it to the instruction register of the microprocessor
- Opcode fetch machine cycle consists of 4 T-states.

T1 State:

During the T1 state, the contents of the program counter are placed on the 16 bit address bus. The higher order 8 bits are transferred to address bus (A8-A15) and lower order 8 bits are transferred to multiplexed A/D (AD0-AD7) bus.

ALE (address latch enable) signal goes high. As soon as ALE goes high, the memory latches the AD0-AD7 bus. At the middle of the T state the ALE goes low

T2 State:

During the beginning of this state, the RD' signal goes low to enable memory. It is during this state, the selected memory location is placed on D0-D7 of the Address/Data multiplexed bus.

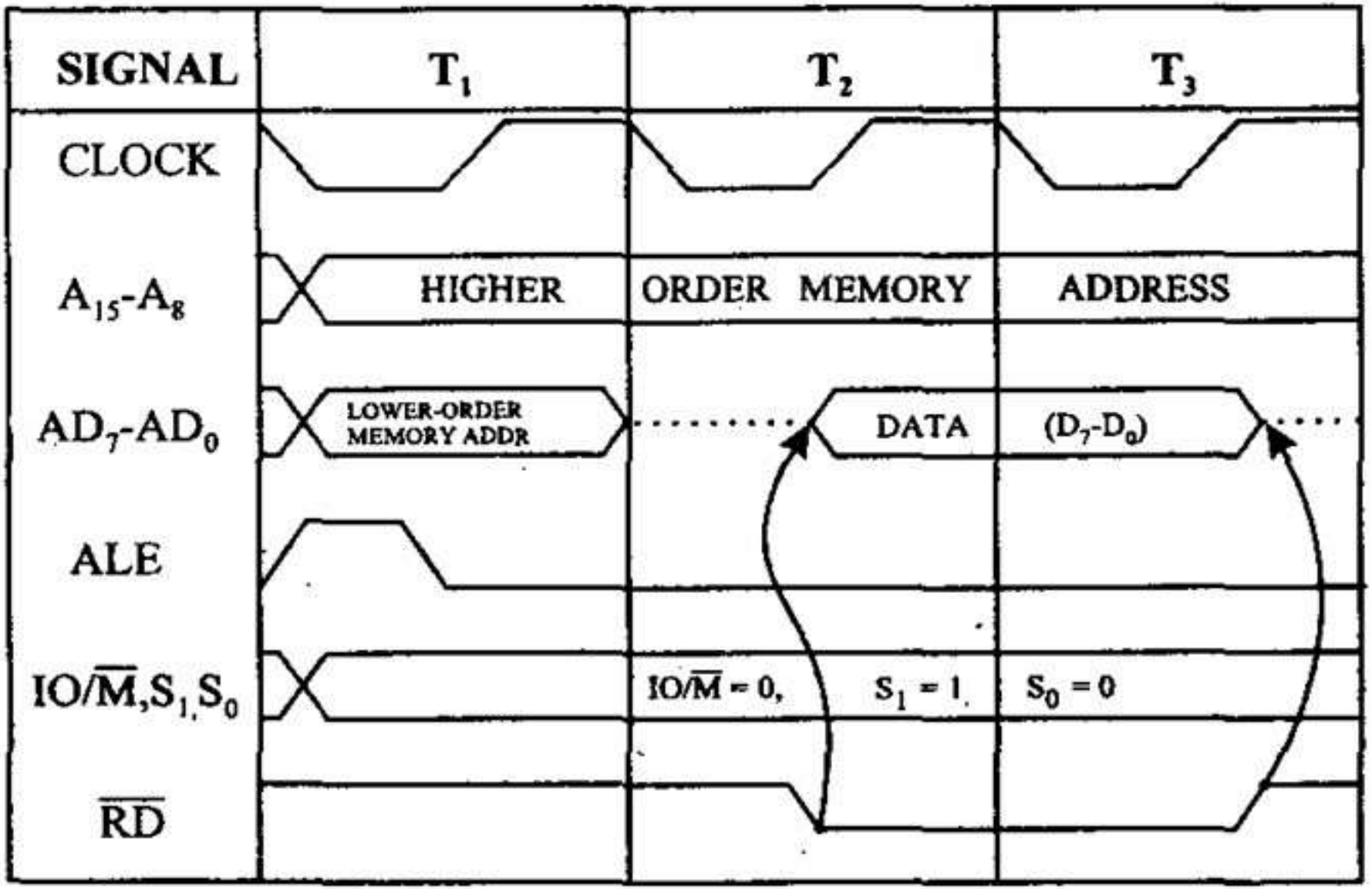
T3 State:

In the previous state the Opcode is placed in D0-D7 of the A/D bus. In this state of the cycle, the Opcode of the A/D bus is transferred to the instruction register of the microprocessor. Now the RD' goes high after this action and thus disables the memory from A/D bus.

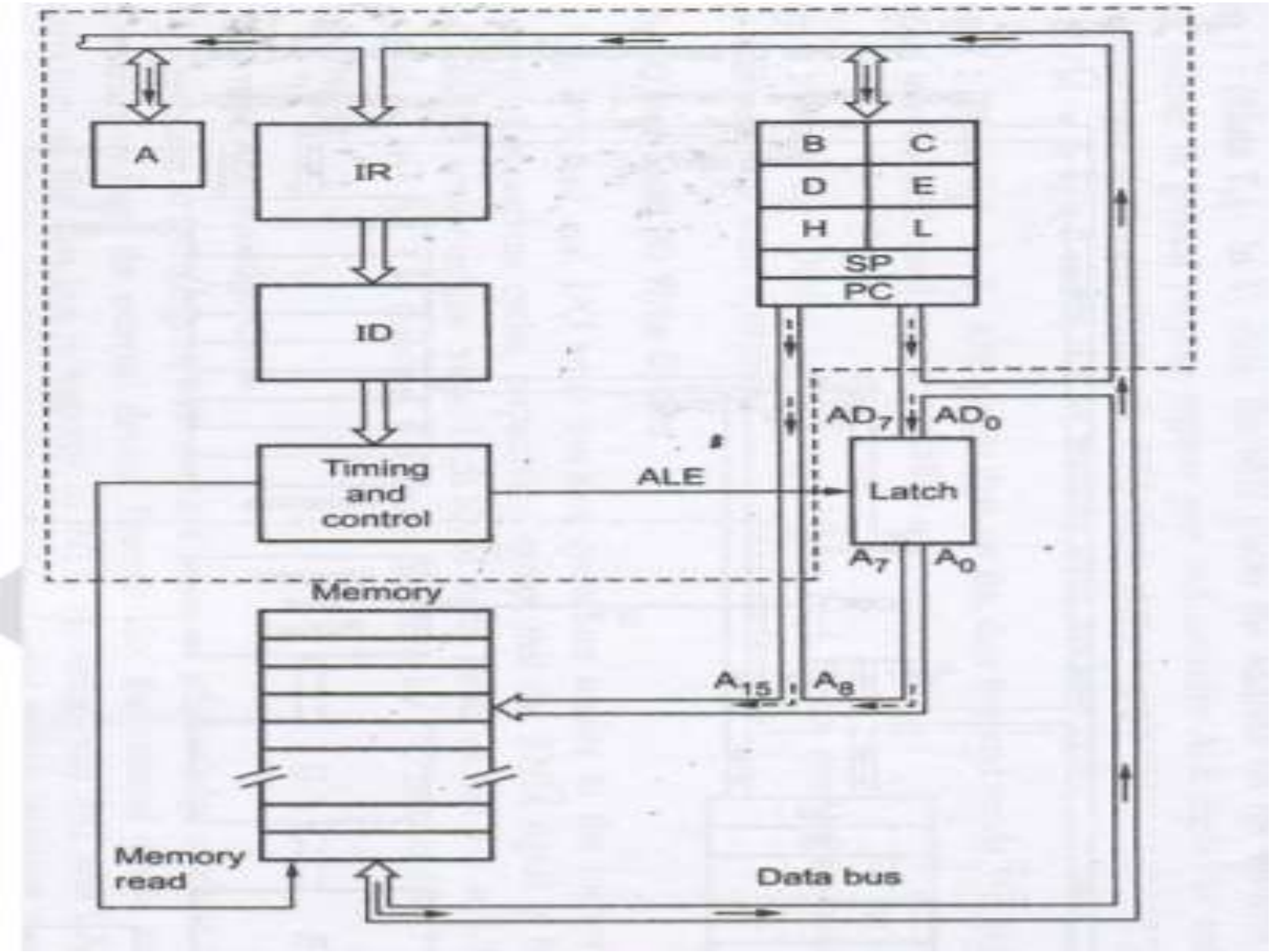
T4 State:

In this state the Opcode which was fetched from the memory is decoded.

2. Memory read cycle (3T)



Memory read cycle



- These machine cycles have 3 T-states.

T1 state:

- The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus. ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.

  The mp identifies the memory read machine cycle from the status signals IO/M'=0, S1=1, S0=0. This condition indicates the memory read cycle.

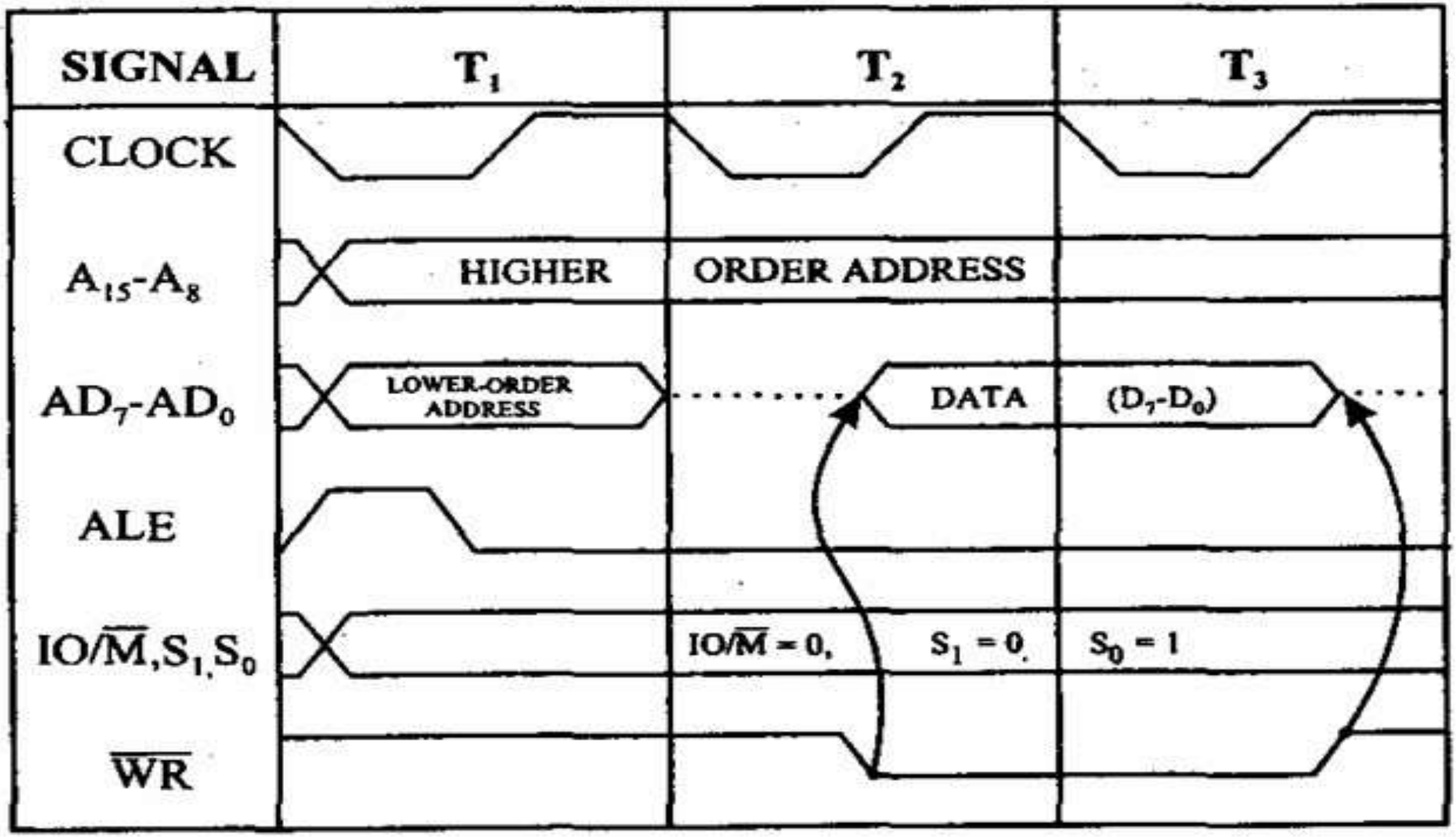
T2 state:

- Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. RD' goes LOW

T3 State:

- The data which was loaded on the previous state is transferred to the microprocessor. In the middle of the T3 state RD' goes high and disables the memory read operation. The data which was obtained from the memory is then decoded.

3. Memory write cycle (3T)



- These machine cycles have 3 T-states.

T1 state:

- The higher order address bus (A8-A15) and lower order address and data multiplexed (AD0-AD7) bus. ALE goes high so that the memory latches the (AD0-AD7) so that complete 16-bit address are available.

  The mp identifies the memory read machine cycle from the status signals IO/M'=0, S1=0, S0=1. This condition indicates the memory read cycle.

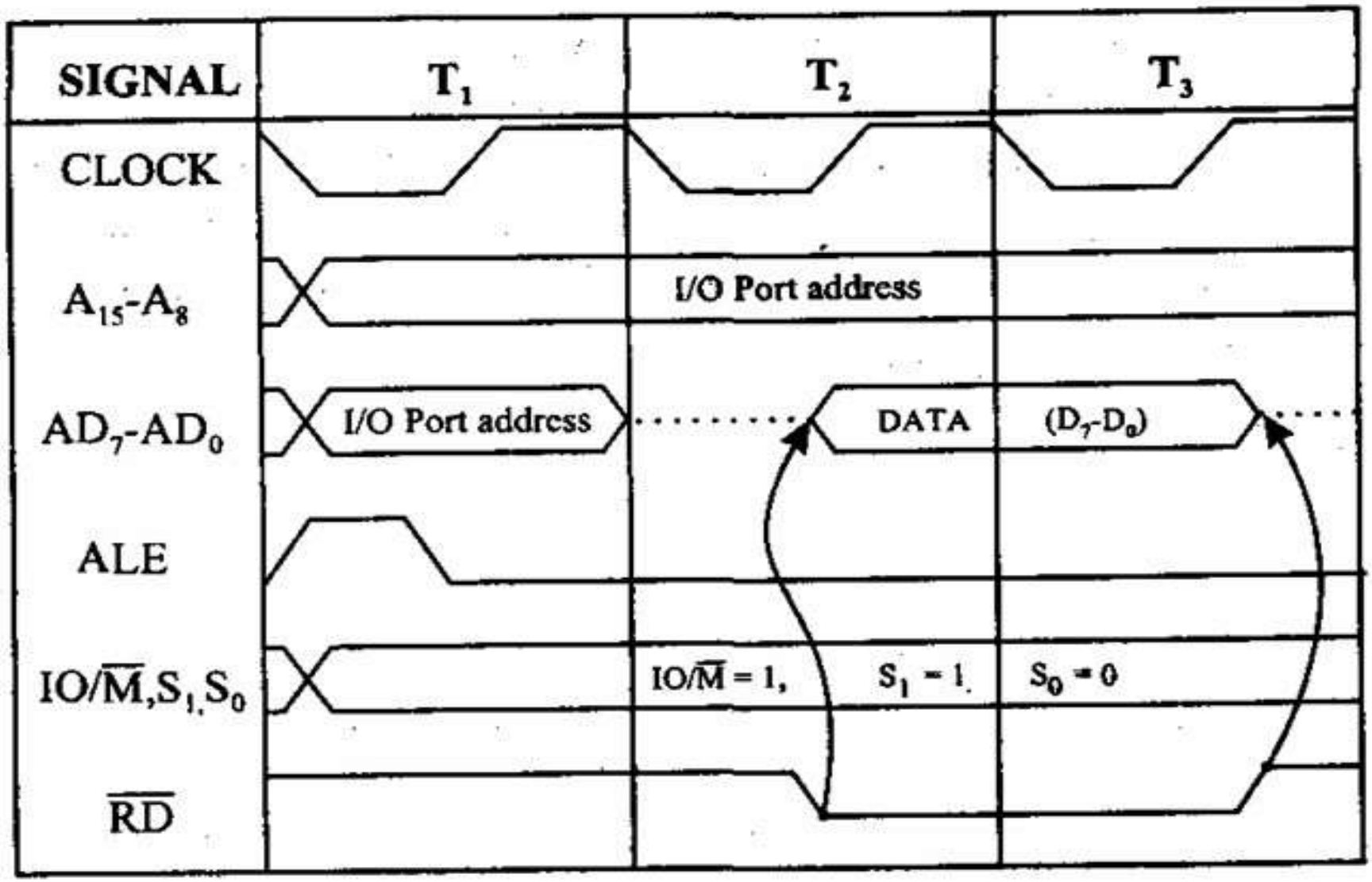
T2 state:

- Selected memory location is placed on the (D0-D7) of the A/D multiplexed bus. WR' goes LOW

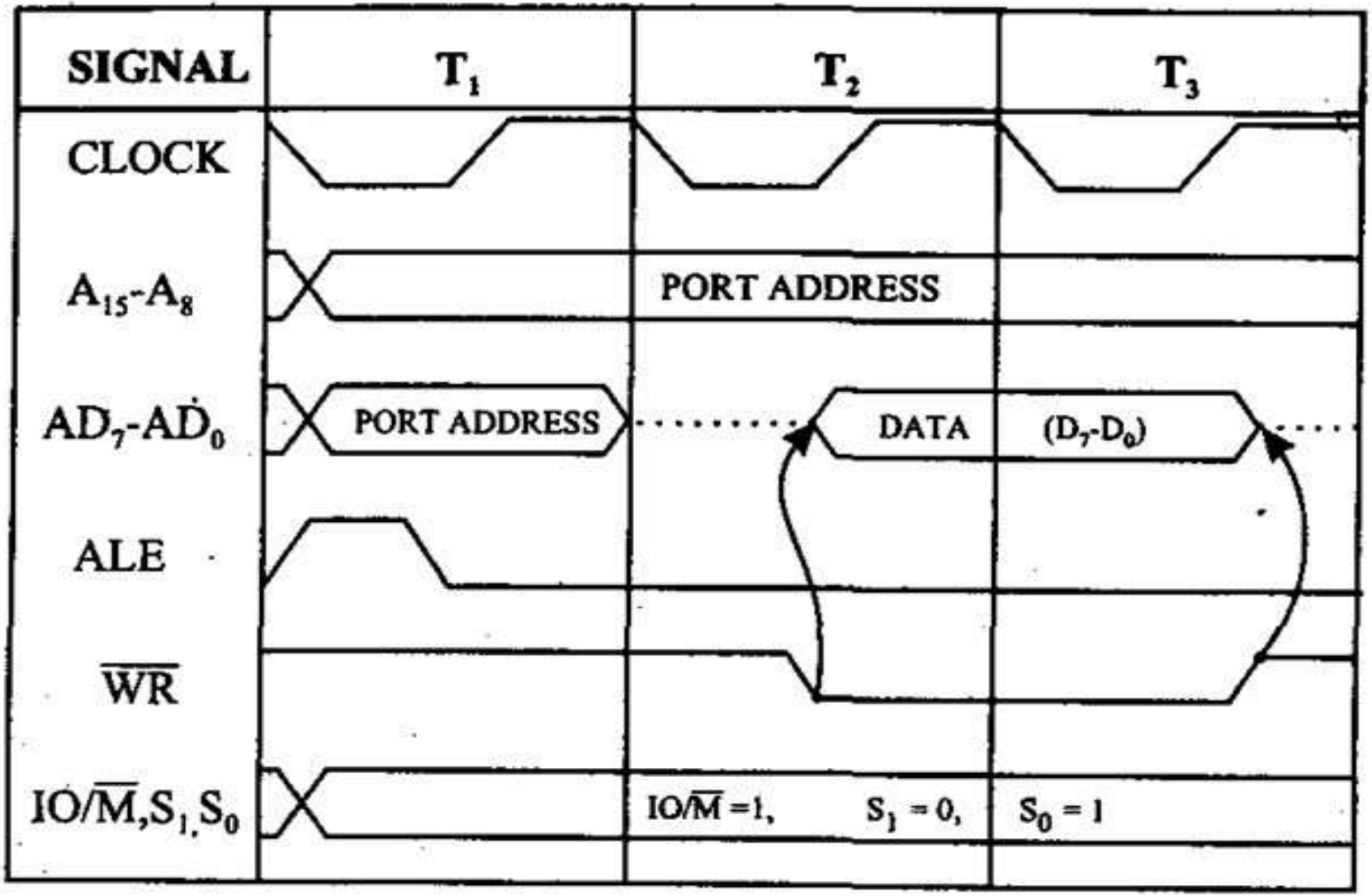
T3 State:

- In the middle of the T3 state WR' goes high and disables the memory write operation. The data which was obtained from the memory is then decoded.

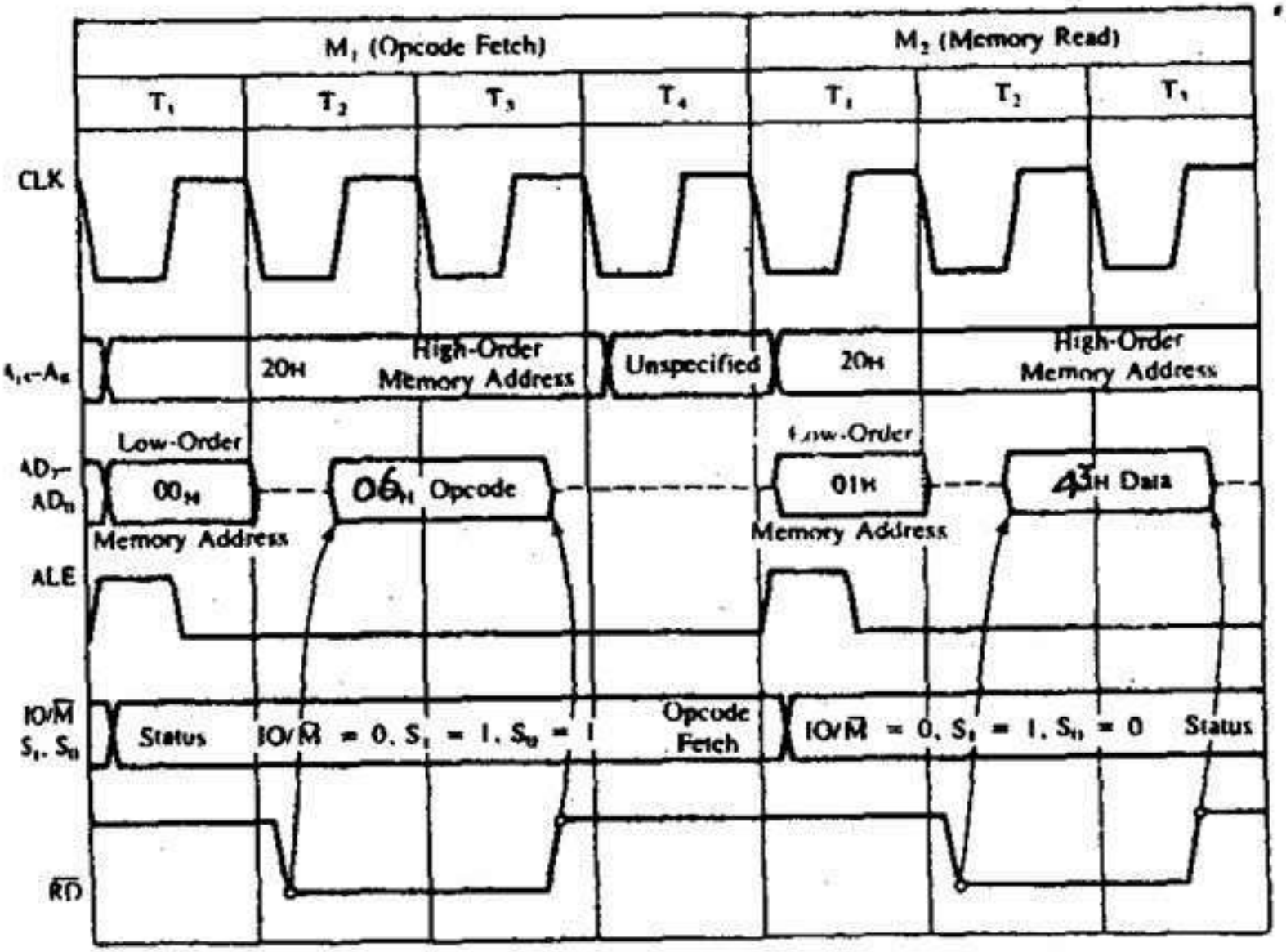
4.I/O read cycle(3T)



5.I/O write cycle(3T)



MVI B, 45


M1 (Opcode Fetch)
M2 (Memory Read)
T1
T2
T3
T4
T1
T2
T3
CLK
A15–A8
20H
High-Order Memory Address
Unspecified
20H
High-Order Memory Address
AD7–AD0
Low-Order
00H
Memory Address
06H Opcode
Low-Order
01H
Memory Address
43H Data
ALE
IO/M
S1, S0
Status
IO/M = 0, S1 = 1, S0 = 1
Opcode Fetch
IO/M = 0, S1 = 1, S0 = 0
Status
RD

TOPIC:

EXAMPLES ON TIMING DIAGRAM

Example 7.1 The 8085 microprocessor is operating on a 2MHz clock signal. Calculate the time required to execute the following instructions :

(i) MVI A, 30H

(ii) MOV A, B

(iii) STA 3050H

Solution : (i) MVI A, 30H

Number of bytes required to store the instruction = 2 bytes. Hence the first part of the instruction cycle requires two machine cycles. Out of these first will be opcode fetch and remaining one will be memory read.

Now the meaning of this instruction MVI A, 30H is to copy the contents 30H to the accumulator. This is internal operation and there is no operation related to memory or input/output. Therefore, no additional machine cycle is needed.

Hence,

Instruction Cycle = Opcode fetch + Memory read (One).

Total T–states = 4T + 3T = 7T.

The microprocessor frequency (F) = 2MHz

$= 2 \times 10^6$Hz

Total time required to execute instruction = 7T

$$= 7\left(\frac{1}{F}\right)$$

$$= \frac{7}{2\times10^6} \text{ sec.}$$

$= 3.5$ μsec.

(ii) **MOV A, B**

Number of bytes required to store the instruction = 1 byte. Hence the first part of the instruction cycle requires only one machine cycle i.e. opcode fetch.

Now the meaning of this instruction MOV A, B is to copy the contents of register B to the accumulator. This is an internal operation and does not need any additional machine cycle.

Hence

Instruction cycle = Opcode fetch

Total T–states = 4T

The microprocessor frequency (F) = 2MHz

$= 2 \times 10^6$Hz

Total time required to execute instruction = 4T

$$= 4\left(\frac{1}{F}\right)$$

$$= \frac{4}{2\times10^6} \text{ sec.}$$

$= 2\mu$ sec.

(iii) STA 3050H

Number of bytes required to store the instruction = 3 bytes. Hence the first part of instruction cycle requires three machine cycles. Out of these first will be opcode fetch and remaining two are memory read.

The meaning of the instruction, STA 3050H is to store the contents of accumulator into the memory location 3050H. This operation requires one additional machine cycle i.e. memory write.

Hence,

Instruction cycle = Opcode fetch + Memory read (two) + Memory write (one)

Total T–states = 4T + 3T + 3T + 3T

= 13T

The microprocessor frequency (F) = 2MHz

$= 2 \times 10^6$Hz

Total time required to execute instruction = 13T

$$= 13\left(\frac{1}{F}\right)$$

$$= \frac{13}{2\times10^6} \text{ sec.}$$

$= 6.5\mu$ sec.

Example 7.2 Draw timing diagram for the instruction MOV A, C and explain how the instruction will be executed. **[Raj. Univ., EE 2001]**

Solution : The given instruction is MOV A, C.

Number of bytes required to store the instruction = 1 byte. Hence the read cycle of instruction requires only one machine cycle i.e. opcode fetch.

Now the meaning of the given instruction is to copy the contents of register C to accumulator. This is an internal operation to the microprocessor. This does not need any memory or input/output related operation. It means there is no execution cycle needed.

Hence

Instruction cycle = Opcode fetch

Therefore, the timing diagram for the given instruction will be exactly similar to timing diagram of opcode fetch machine cycle. This is shown in fig. 7.16.

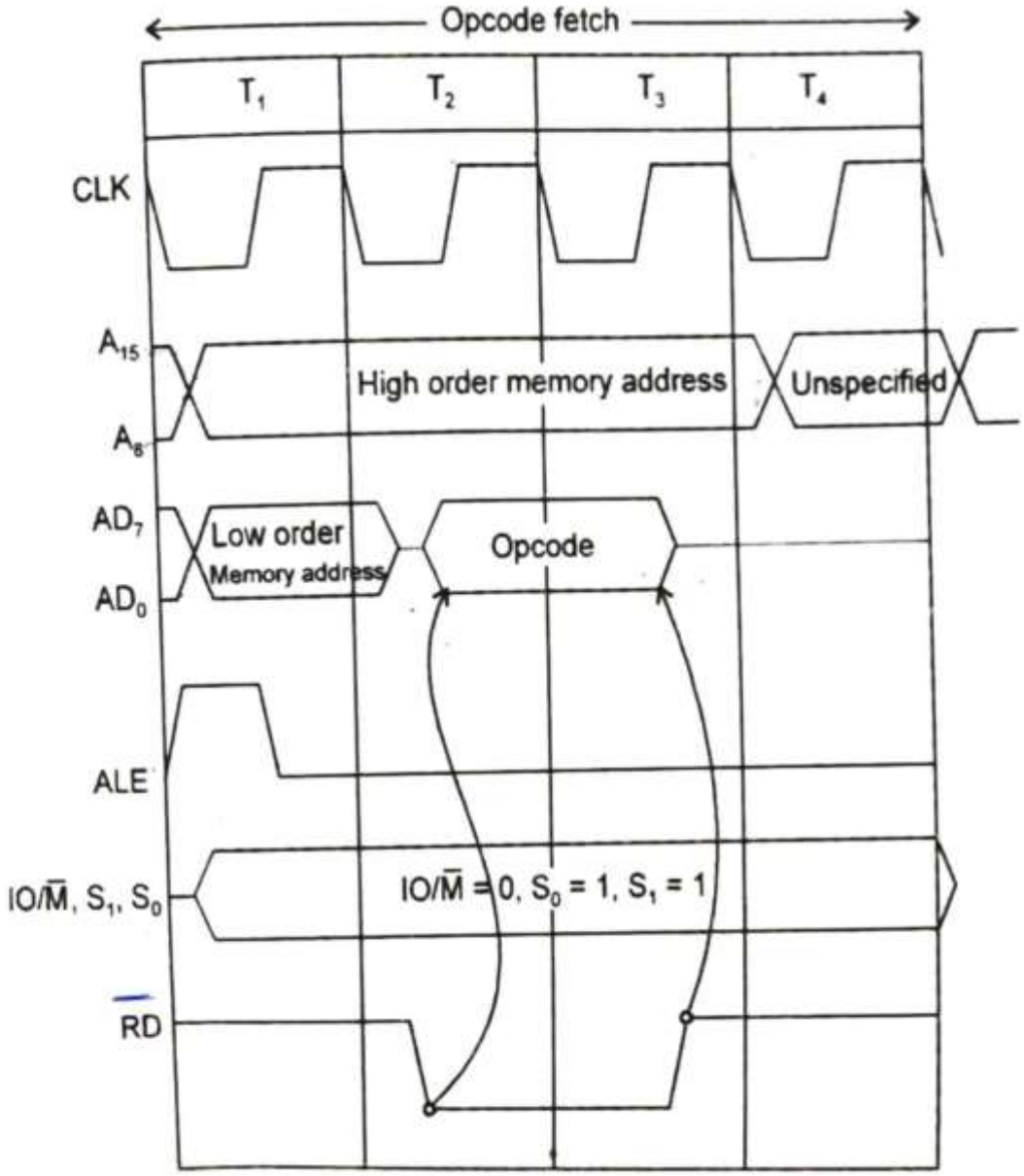


Fig. 7.16 : Timing diagram for instruction MOV A, C

The following steps are performed to execute the instruction –

Step–1 : The microprocessor places the high order address from program counter (PC_H) to the high order address bus and low order address from program counter (PC_L) to low order address/data multiplexed bus. The ALE signal goes high at the starting of T_1 state to latch the low order address in external latch. ALE signal goes low again at the end of T_1 state. The status signals are $IO/\bar{M} = 0$, $S_1 = 1$ and S_0 1. This is shown in T_1 state of timing diagram (Fig. 7.16.)

Step 2 : The control unit sends the control signal $\overline{RD}$ to enable the memory chip. After receiving the $\overline{RD}$ signal, the opcode of the instruction (79H) is loaded to the data bus. This is T_2 state of timing diagram (fig. 7.16).

Step 3 : The control signal $\overline{RD}$ raises to high which disables the memory chip. Before that the microprocessor loads the opcode from data bus to the instruction register.

Step 4 : The opcode byte is placed in the instruction decode and task is carried out according to the instruction. The contents of register C are copied to the accumulator.

Program Debugging

Debugging: the process of finding the clues and interpreting these clues to find the problem

- Choose an input data set
- Predict what the program will do with the input data set at each step of the program, and what the program will do next (the program model)
- Now run the program, and look for data values and program steps that differ from the prediction
- Once we find out where the program deviates from the prediction, we are on track to finding the bugs and fixes for the bugs

Program Debugging

- **Program trace:** stepping through the program one statement at a time. Values of registers are shown during each step. Data elements in memory have to be checked manually.
- **Breakpoints:** much like a high-level language debugger, e.g. Gdb. These cause program flow to be interrupted, and control transferred to the debugger (the user).
- Breakpoints can be set at statements, as well as when certain conditions are met, e.g. data elements become some specific values.

THANK YOU