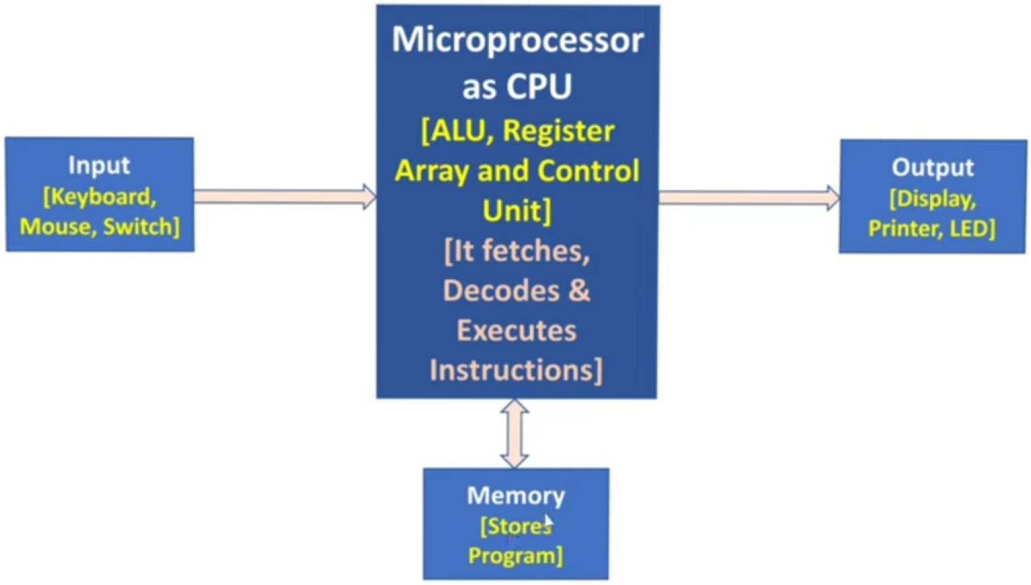
How to Count Memory?

- Byte : It is 8 bits Binary number. It is also 2 digit Hex number.
- Word : It is 16 bits Binary number. It is also 4 digit Hex number.
- Why we prefer HEX number system over decimal number system?
  - With decimal digits (0-9), it needs 4 bits to represents each digits (0 = 0000 to 9 = 1001).
  - With hexadecimal digits (0-F), it needs 4 bits to represents each digits (0 = 0000 to F = 1111).
  - So by decimal number system, we don't use 6 code of binary (1010 to 1111)
  - Also decimal to binary conversion is required in computer system for different arithmetic operation, which is not required in Hexadecimal system.
- For 10 Address lines total address = 2^{10}= 1024 = 1K
- For 20 Address lines total address = 2^{20}= $2^{10} \times 2^{10}$ = 1M
- For 30 Address lines total address = 2^{30}= $2^{20} \times 2^{10}$ = 1G
- For 40 Address lines total address = 2^{40}= $2^{30} \times 2^{10}$ = 1T

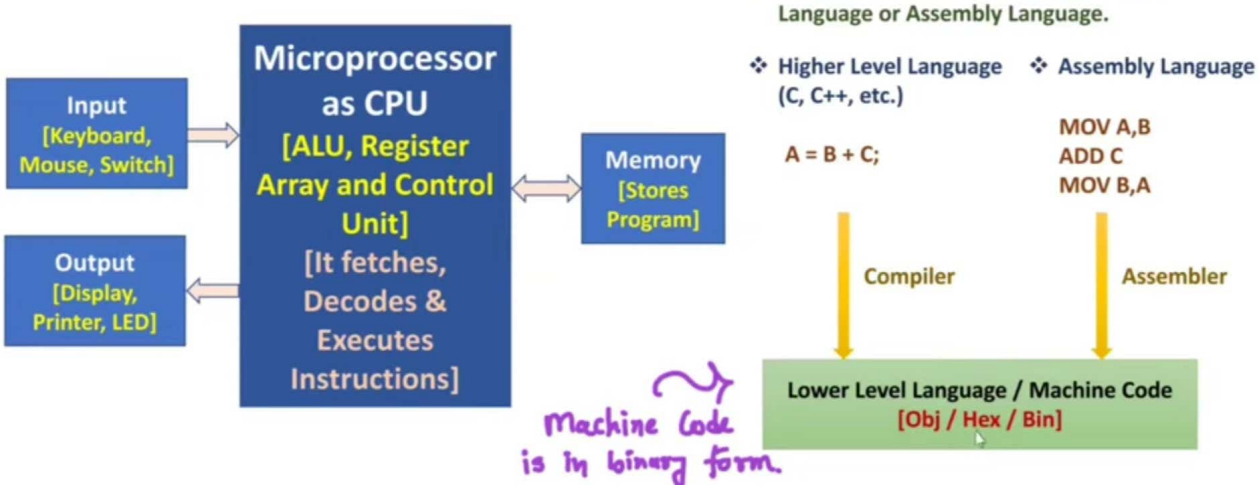
$2^1 = 2$	$2^9 = 512$
$2^2 = 4$	$2^{10} = 1024$
$2^3 = 8$	$2^{11} = 2048$
$2^4 = 16$	$2^{12} = 4096$
$2^5 = 32$	$2^{13} = 8192$
$2^6 = 64$	$2^{14} = 16384$
$2^7 = 128$	$2^{15} = 32768$
$2^8 = 256$	$2^{16} = 65536$

- Examples
  - $8K \times 8$ means 13 Address lines and 8 Data lines.
  - $32K \times 8$ means 15 Address lines and 8 Data lines.
  - $32M \times 8$ means 25 Address lines and 8 Data lines.
  - $4K \times 4$ means 12 Address lines and 4 Data lines.

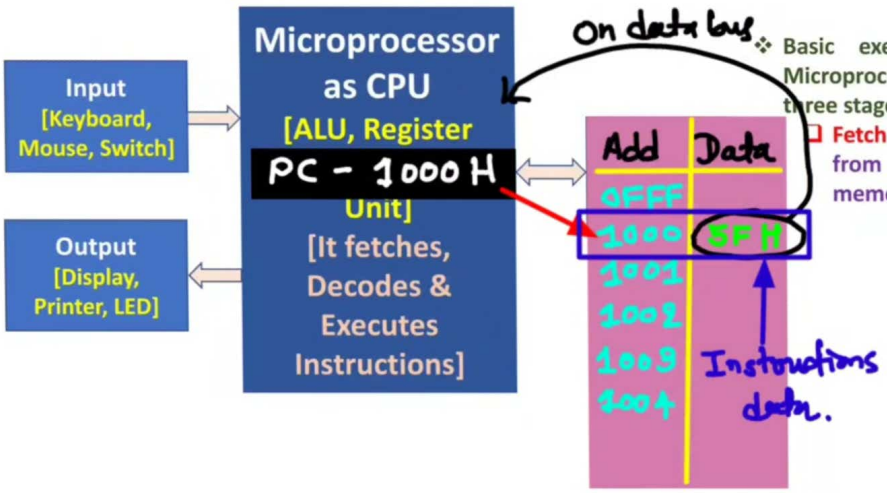
What is Computer?



How Program is loaded in Memory of Processor?

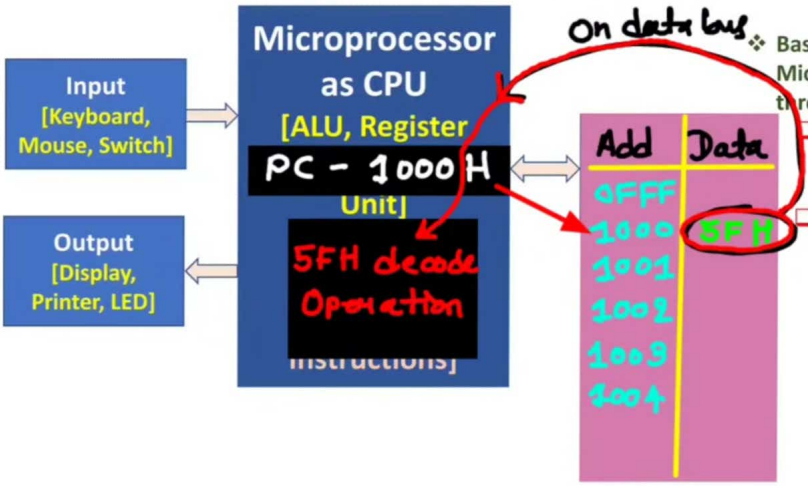


How Program is Executed in Processor?



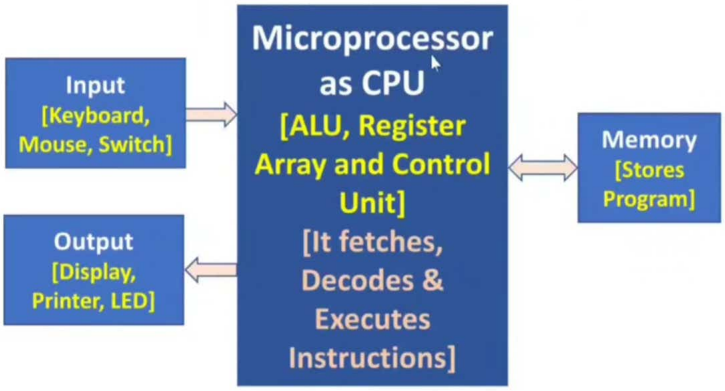
- Basic execution of Instruction with any Microprocessor/ Microcontroller is there at three stages.
  - Fetch [Here, processor load instruction from Memory. It Fetches opcode from memory to processor.]

How Program is Executed in Processor?



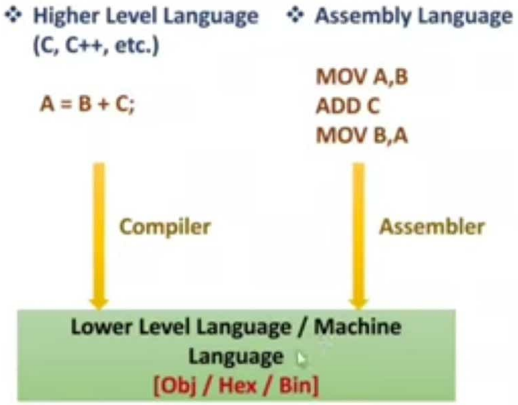
- Basic execution of Instruction with any Microprocessor/ Microcontroller is there at three stages.
  - **Fetch** [Here, processor load instruction from Memory. It Fetches opcode from memory to processor.]
  - **Decode** [Here, processor identifies which operation to be executed.]

How Program is Executed in Processor?



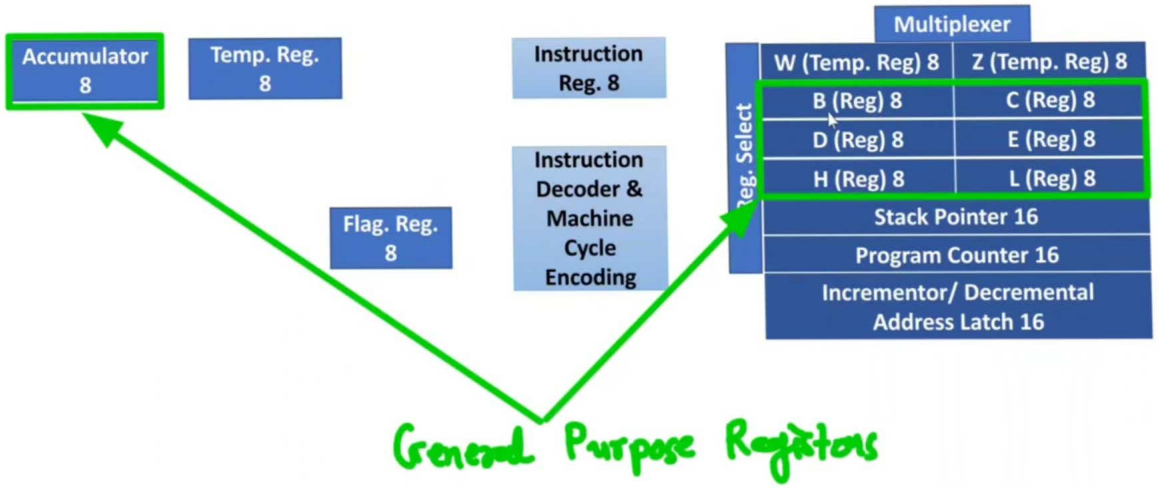
- Basic execution of Instruction with any Microprocessor/ Microcontroller is there at three stages.
  - **Fetch** [Here, processor load instruction from Memory. It Fetches opcode from memory to processor.]
  - **Decode** [Here, processor identifies which operation to be executed.]
  - **Execute** [Here, processor perform the task and write results into registers.]
- This three stages together is called instruction cycle.
- Sequence of Instruction cycle makes program.

Machine Language, Assembly Language & Higher Level Language

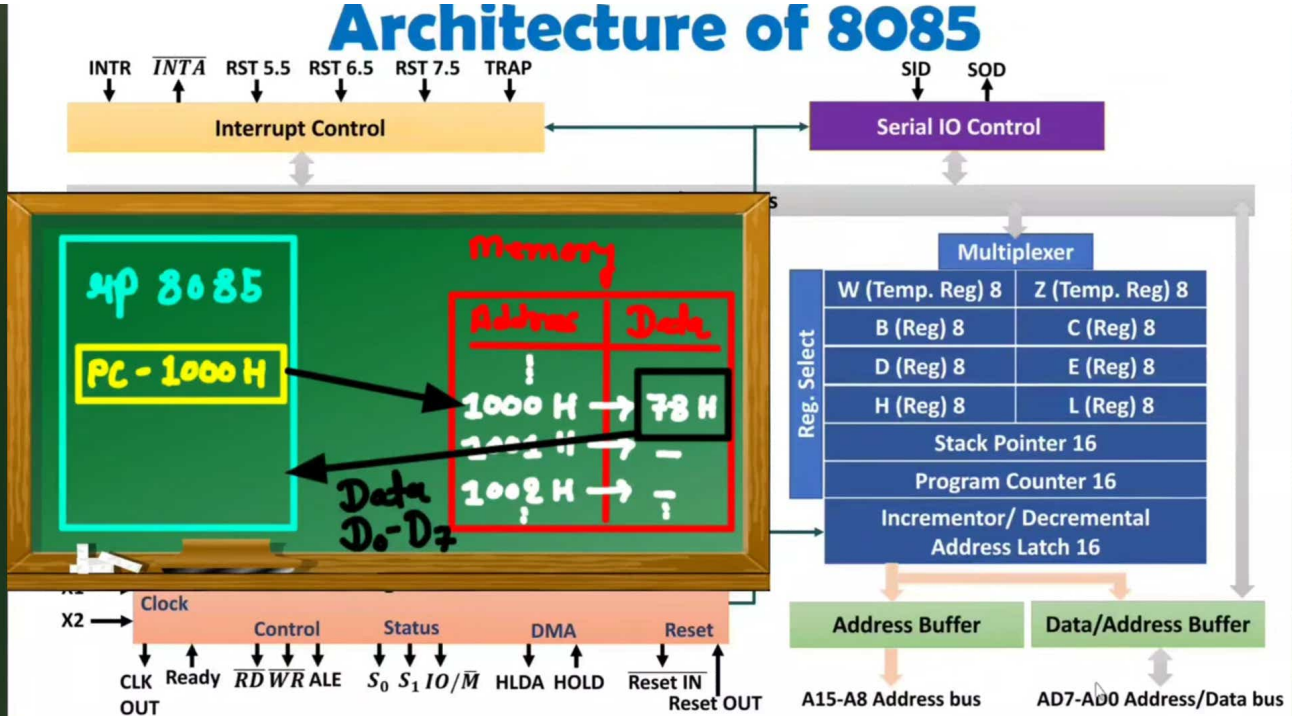


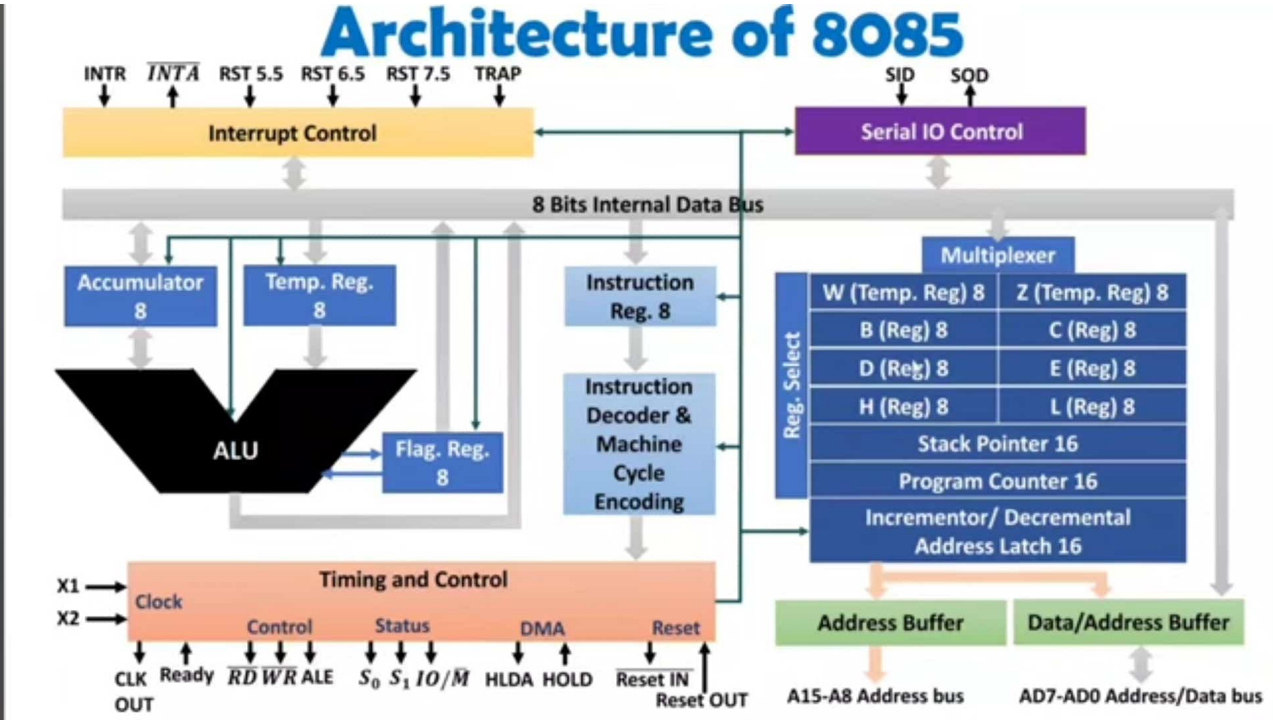
- **Higher Level Language**
  - Commonly used higher level language for microprocessor and microcontroller is C, C++.
- **Assembly Language**
  - In Assembly Language, programing is done with use of Mnemonics of given processor or controller.
- **Mnemonics**
  - In Assembly Language programing instructions are written in terms of Mnemonics.
  - Example of 8085: ADD B and MOV B,C
- **Complier**
  - Main task of complier is to convert HLL into Machine Language.
  - If syntax error is there then it also indicate error in given lines.
- **Assembler**
  - Main task of Assembler is to convert AL into Machine Language.
  - If syntax error is there then it also indicate error in given lines.
- **Machine Language**
  - Machine Language is there in terms of binary which can be easily understands by processor or controller.

Architecture of 8085

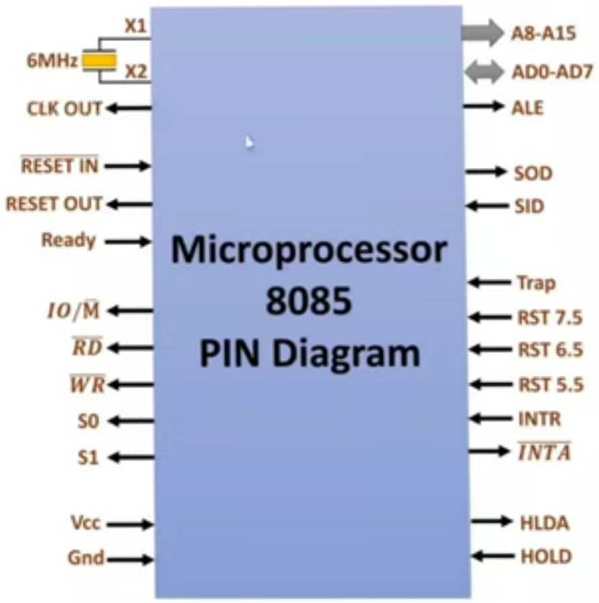


Architecture of 8085




Architecture of 8085
INTR
$\overline{INTA}$
RST 5.5
RST 6.5
RST 7.5
TRAP
SID
SOD
Interrupt Control
Serial IO Control
8 Bits Internal Data Bus
Accumulator 8
Temp. Reg. 8
Instruction Reg. 8
Multiplexer
Reg. Select
W (Temp. Reg) 8
Z (Temp. Reg) 8
B (Reg) 8
C (Reg) 8
D (Reg) 8
E (Reg) 8
H (Reg) 8
L (Reg) 8
Stack Pointer 16
Program Counter 16
Incrementor/ Decremental Address Latch 16
ALU
Flag. Reg. 8
Instruction Decoder & Machine Cycle Encoding
X1
X2
Timing and Control
Clock
Control
Status
DMA
Reset
CLK OUT
Ready
$\overline{RD}$ $\overline{WR}$ ALE
S_0 S_1 $IO/\overline{M}$
HLDA
HOLD
Reset IN
Reset OUT
Address Buffer
Data/Address Buffer
A15-A8 Address bus
AD7-AD0 Address/Data bus

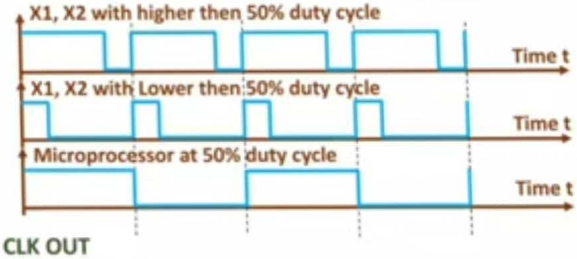
PIN Diagram of 8085



✓ **Power Supply and Clock Frequency**

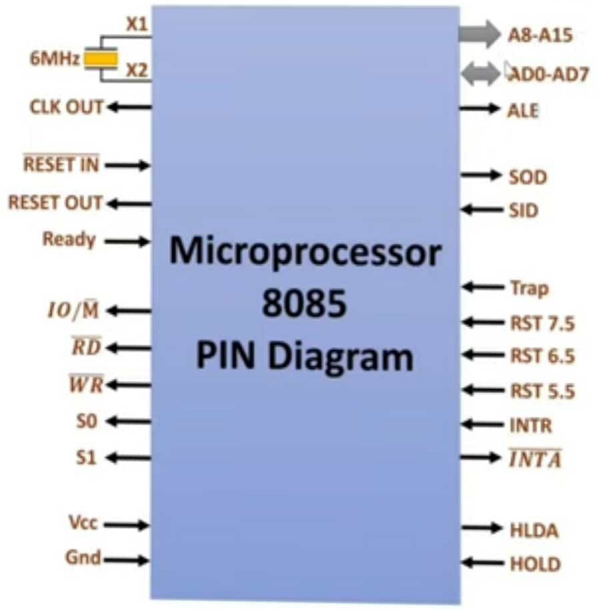
❖ Power Supply
- ❑ 8085 has Vcc = +5V and Gnd = Reference

❖ X1 and X2
- ❑ Crystal is connected in between these two pins.
- ❑ Frequency is internally divided by two.
- ❑ So to operate at 3MHz we should give 6MHz clock.



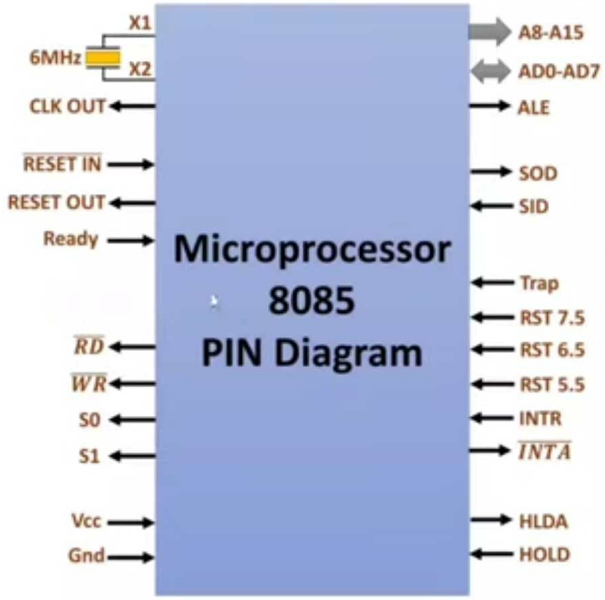
❖ CLK OUT
- ❑ Clock output is used to provide clock for other devices.

PIN Diagram of 8085



- Address Bus
  - 8085 has 16 lines for Address bus.
  - These lines are split into two segments: A15-A8 and AD7-AD0.
  - Here A15-A8 holds higher Byte of Address and AD7-AD0 has address and data lines.
- Data Bus
  - 8085 has 8 lines for Data bus AD7-AD0.
  - In executing an instruction, during earlier part of the cycle, these lines are Address lines and later part of cycle it holds data bus.
- ALE – Address Latch Enable
  - It is a positive going pulse generated every time in machine cycle.
  - It is used to latch Lower address from AD7-AD0.
  - When it is zero, it indicates Data on AD7-AD0.

PIN Diagram of 8085



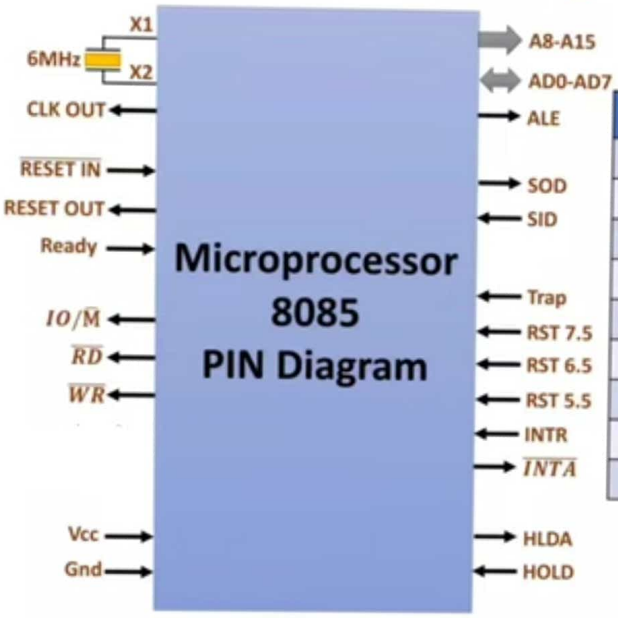
✓ Control Signals [ALE, $\overline{RD}$, $\overline{WR}$, IO/$\overline{M}$, S0 and S1]

❖ $\overline{RD}$ = Read
- ❑ It is Active Low signal which indicates read operation to be performed by 8085 over data lines.
- ❑ This read operation may be there with memory or IO devices.

❖ $\overline{WR}$ = Write
- ❑ It is Active Low signal which indicates write operation to be performed by 8085 over data lines.
- ❑ This write operation may be there with memory or IO devices.

❖ $IO/\overline{M}$ = Input-Output or Memory
- ❑ If it is logic '1', IO operations should be done by 8085.
- ❑ If it is logic '0', Memory operations should be done by 8085.

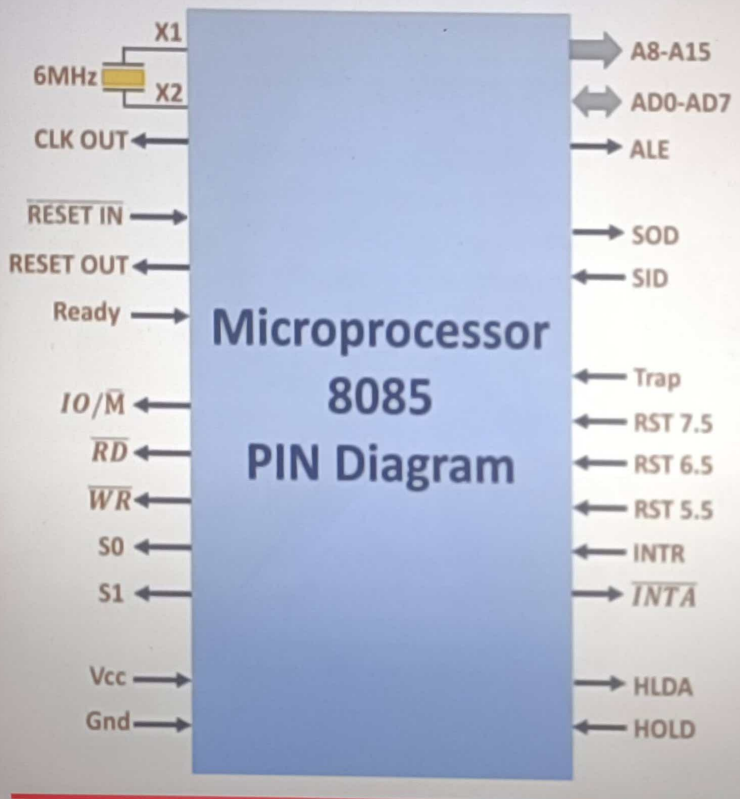
PIN Diagram of 8085



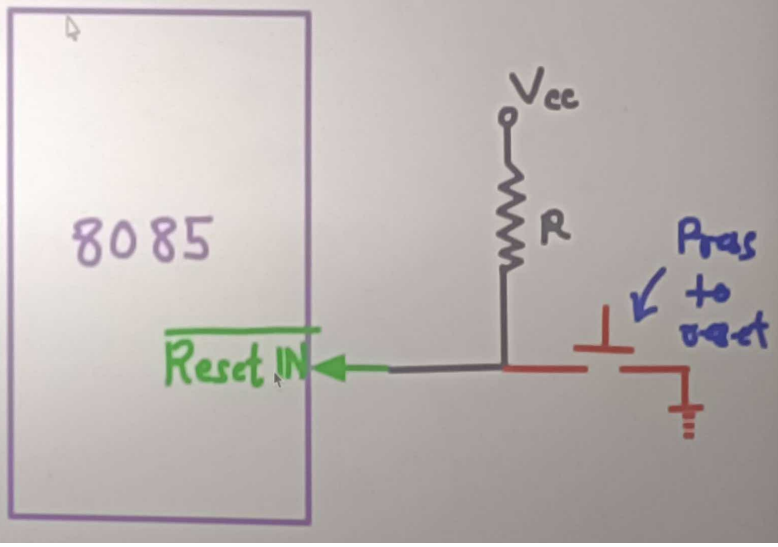
- S0 and S1
  - These are status signals.

Machine Cycle	$IO/\overline{M}$	S1	S0	Control Signals
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
IO Read	1	1	0	$\overline{RD} = 0$
IO Write	1	0	1	$\overline{WR} = 0$
Interrupt Acknowledge	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD} = \overline{WR} = Z$ $\overline{INTA} = 1$
Hold	Z	X	X	
Reset	Z	X	X	

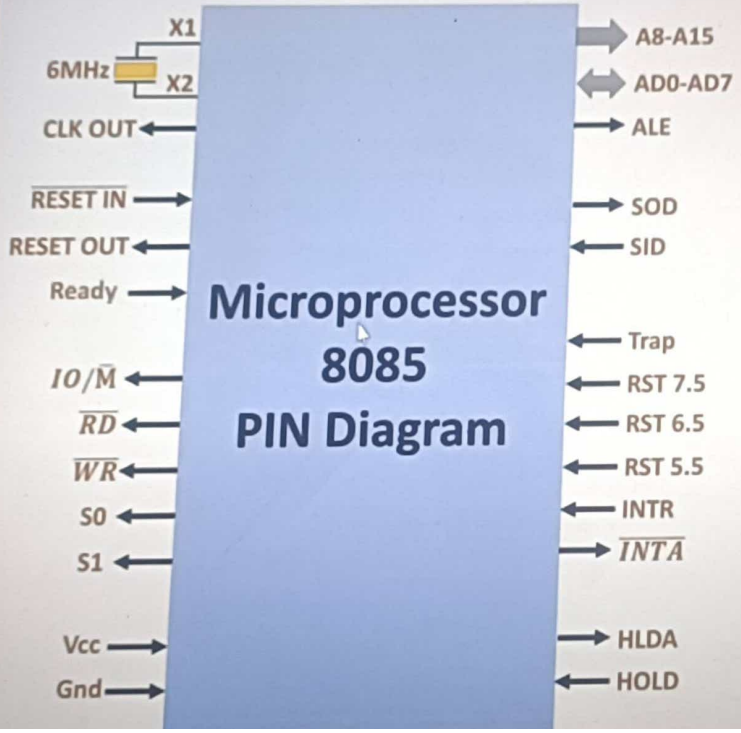
PIN Diagram of 8085



- ✓ Reset and Ready
- ❖ $\overline{RESET\ IN}$
  - ❑ When it goes low, 8085 is getting reset.
  - ❑ PC = 0000H
  - ❑ The Buses are tristate during that.



PIN Diagram of 8085



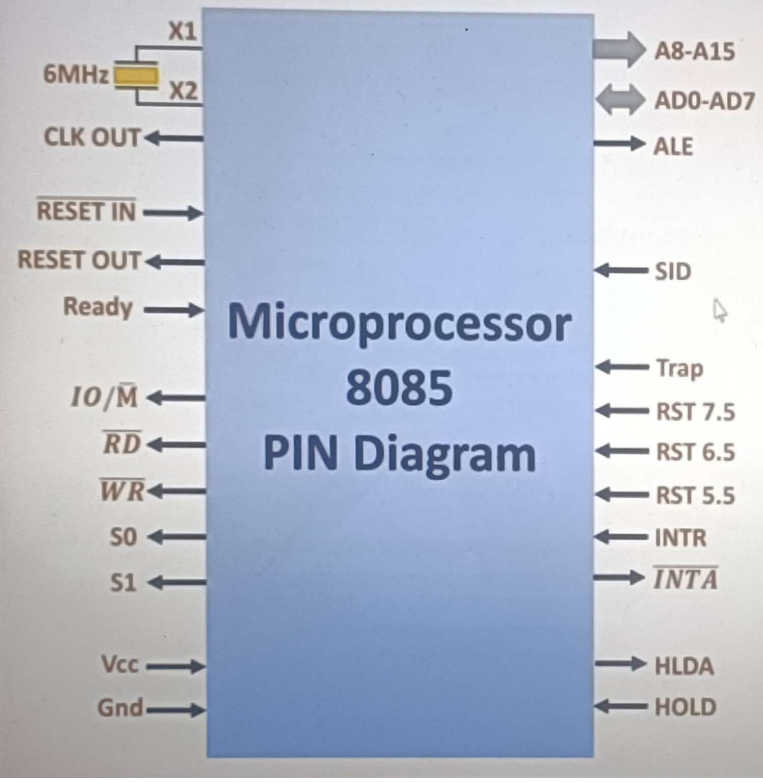
✓ Reset and Ready

❖ $\overline{RESET\ IN}$
- ❑ When it goes low, 8085 is getting reset.
- ❑ PC = 0000H
- ❑ The Buses are tristate during that.

❖ RESET OUT
- ❑ It indicates that the MPU is being reset. It can be used to reset other devices.

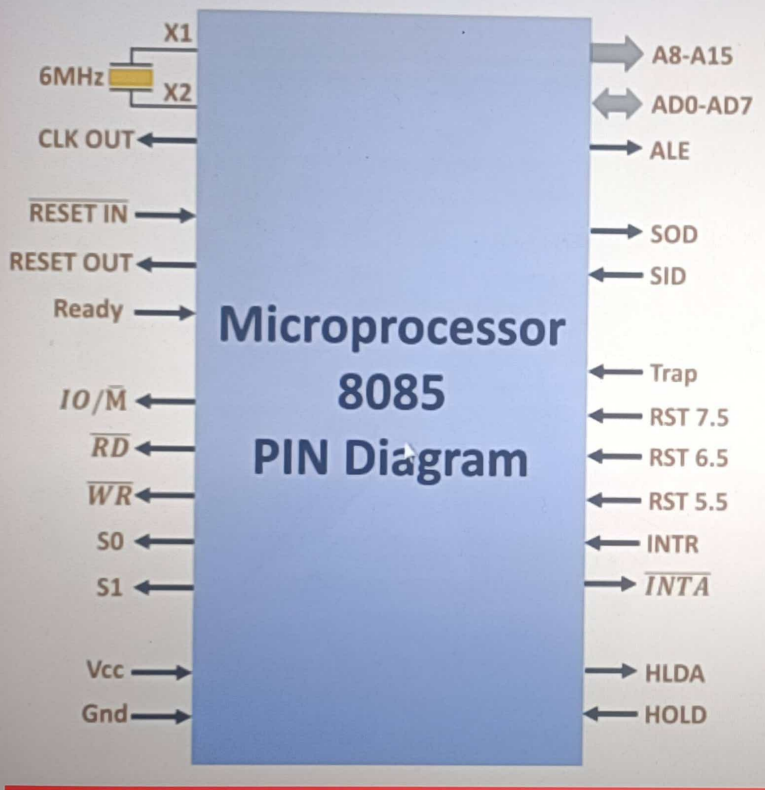
❖ Ready
- ❑ This signal is used to delay the microprocessor read or write cycles until a slow responding peripheral is ready to send or receive data.
- ❑ When it goes low 8085 will wait for integer numbers of cycles until it goes high.

PIN Diagram of 8085



- ✓ Serial Data Communication
- ❖ SID – Serial Input Data
  - ❑ It is used for serial data input.
- ❖ SOD – Serial Output Data
  - ❑ It is used for serial data output.

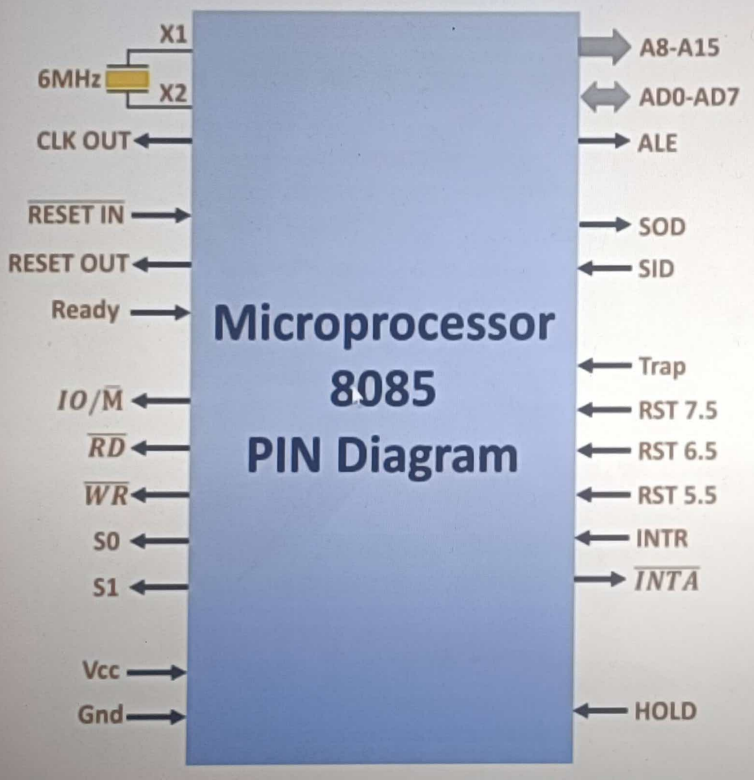
PIN Diagram of 8085



✓ 8085 Interrupt

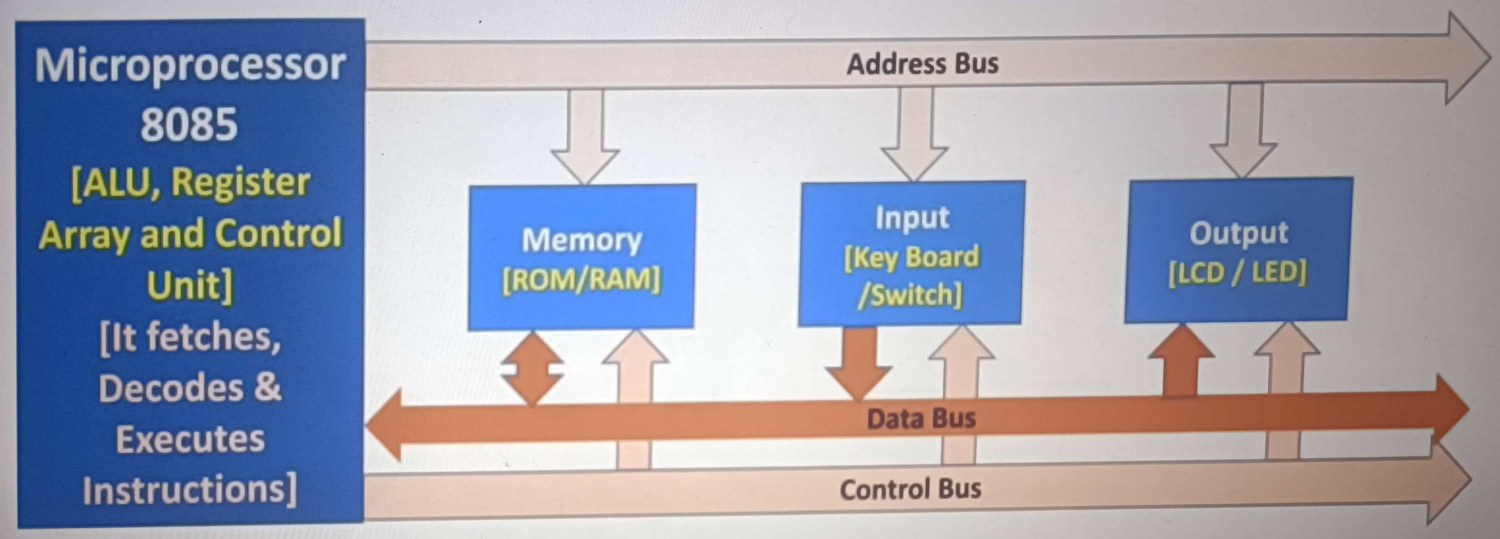
- INTR – Interrupt request
  - It is used for general purpose interrupt.
- $\overline{INTA}$ - Interrupt Acknowledge
  - It is used for Interrupt acknowledge.
- RST 7.5, RST 6.5 and RST 5.5
  - These are vectored interrupts that transfer program control to specific memory locations.
  - They have higher priorities than the INTR interrupt.
- TRAP
  - It is non maskable interrupt.
  - It has highest priority.

PIN Diagram of 8085



- ✓ DMA signals
- ❖ HOLD
  - ❑ This signal indicates that a peripheral such as DMA controller is requesting the use of the address and data bus.
- ❖ HLDA
  - ❑ It is Hold Acknowledge.
  - ❑ It is giving acknowledgment to hold signal of DMA.

Buses in 8085

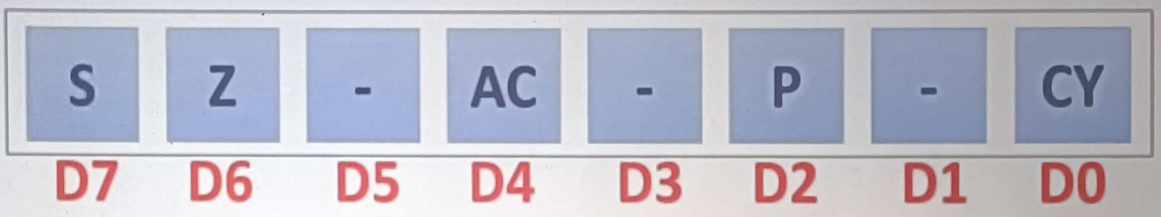


❖ **Address Bus**
- ❑ There are A0-A15 Address buses in 8085.
- ❑ These busses are used to Identify the address of Memory and IO peripherals.

❖ **Control Bus**
- ❑ There four control signals in general from 8085.
- ❑ Memory Read, Memory write, IO Read & IO write.

❖ **Data Bus**
- ❑ There are D0-D7 Data buses with 8085.
- ❑ It is bidirectional buses. Direction of buses is based on read and write operation.

Flag Register in 8085

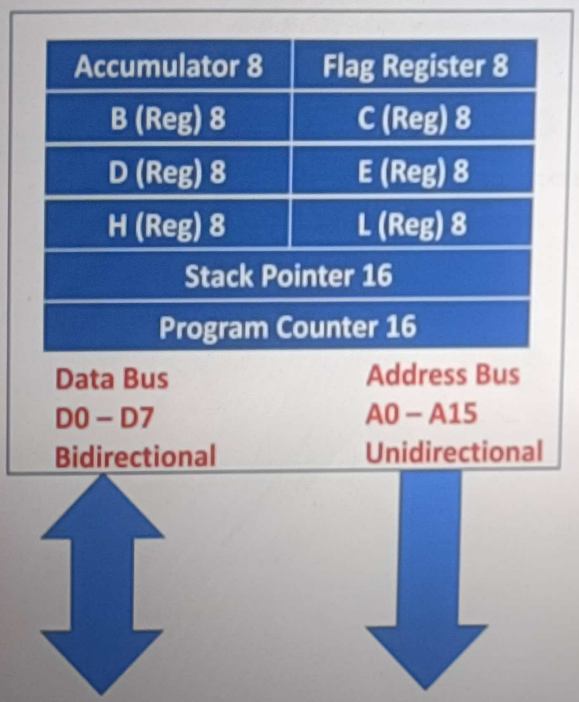


- CY – Carry Flag
  - Cy = 1, Carry is generated in execution of instruction.
  - Cy = 0, Carry is not generated in execution of instruction.
- P – Parity Flag
  - P = 1, Result has even numbers of 1's.
  - P = 0, Result has Odd numbers of 1's.
- AC – Auxiliary Flag
  - AC = 1, Result has carry from D3 to D4 bit, that is referred Nibble to Nibble carry.
  - AC = 0, Result has no carry from D3 to D4 bit, that is referred zero Nibble to Nibble carry.
- Z – Zero Flag
  - Z = 1, Result is Zero after execution of Instruction.
  - Z = 0, Result is Non Zero after execution of Instruction.
- S – Sign Flag
  - S = 1, Result has D7 = 1.
  - S = 0, Result has D7 = 0.

Example:
MVI A, CCH
MVI B, E6H
ADD B

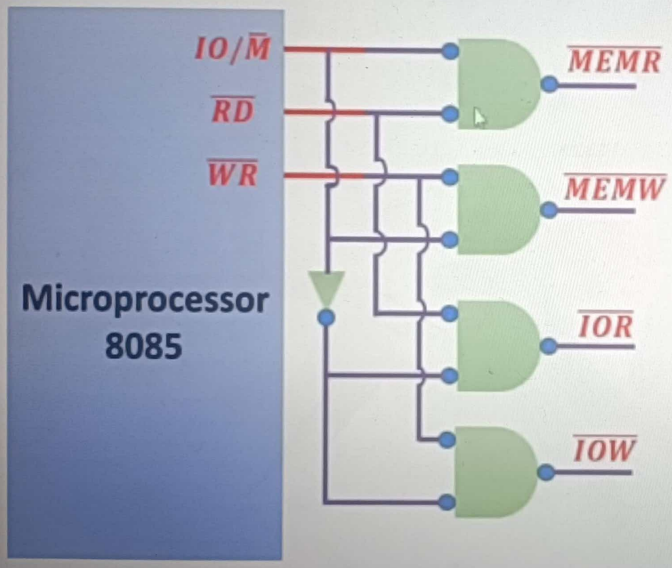
```
          11  1 1
A = CCH   1100 1100
B = E6H   1110 0110
-------------------
A = B2H   1011 0010
```

Programming Model of 8085



- Accumulator (8 bits)
  - This is main register of 8085, as it is default operand of ALU.
  - So 8085 performs all the operations with respect to Accumulator.
- Flag Register (8 bits)
  - 8085 has Five essential flag in flag register (S, Z, AC, P & Cy)
- General Purpose Registers
  - 8085 has seven general purpose registers (A, B, C, D, E, H & L)
  - These registers are used for programming of 8085.
  - We can use BC, DE and HL in pair for programming.
- Program Counter (16 bits)
  - 8085 uses PC to sequence the execution of program.
  - The function of PC is to point the address from which the next byte is to be fetched, when a byte is being fetched, the PC is incremented by one to point next instruction.
- Stack Pointer (16 bits)
  - 8085 uses SP as memory pointer.
  - It points R/W memory, called as Stack.
  - Top of Stack memory address is indicated by SP.

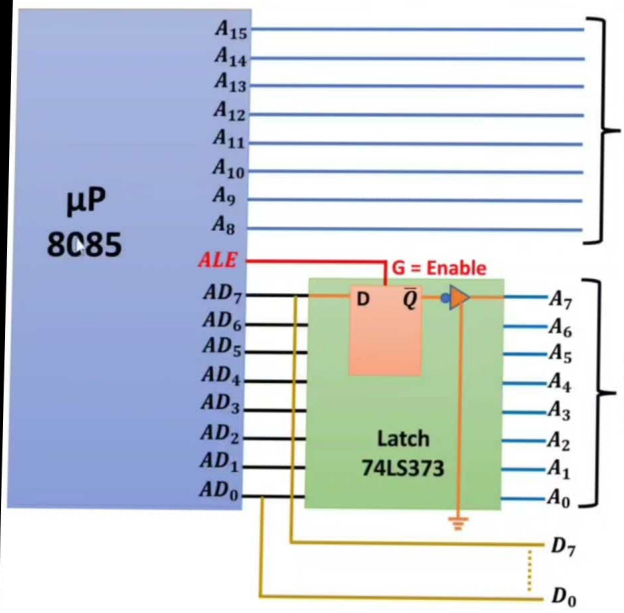
Generation of Control Signals in 8085



- Using $IO/\overline{M}$, $\overline{RD}$ and $\overline{WR}$ we can generate four control signals: Memory Read, Memory Write, IO Read and IO Write.
- $IO/\overline{M}$ = Input-Output or Memory
  - If it is logic '1', IO operations should be done by 8085.
  - If it is logic '0', Memory operations should be done by 8085.
- $\overline{RD}$ = Read
  - It is Active Low signal which indicates read operation to be performed by 8085 over data lines.
  - This read operation may be there with memory or IO devices.
- $\overline{WR}$ = Write
  - It is Active Low signal which indicates write operation to be performed by 8085 over data lines.
  - This write operation may be there with memory or IO devices.

$IO/\overline{M}$	$\overline{RD}$	$\overline{WR}$	Control Signals
0	0	1	Memory Read - $\overline{MEMR}$
0	1	0	Memory Write - $\overline{MEMW}$
1	0	1	Input Output Read - $\overline{IOR}$
1	1	0	Input Output Write - $\overline{IOW}$

Address Data Demultiplexing in 8085



❖ Address / Data Lines of 8085

- ❑ 8085 has 16 lines for Address and Data.
- ❑ Higher Byte of address is there on A15-A8.
- ❑ Lower Byte of address is time multiplexed with data lines which is given by AD7-AD0.
- ❑ To separate address and data we use ALE terminal.
- ❑ When positive going pulse is given by ALE at that time AD7-ADO represents address.
- ❑ When ALE is active low at that time AD7-AD0 works as data lines.

Memory Mapped IO and IO Mapped IO in 8085

Memory Mapped IO	IO Mapped IO
❖ Here IO devices treated as Memory	❖ Here IO devices treated as IO.
❖ 16 bits addressing ($A_0 - A_{15}$)	❖ 8 bits addressing ($A_0 - A_7$)
❖ It can address = 2^{16} = 64K Address	❖ It can address = 2^8 = 256 Address
❖ Number of devices = 64K = 65536	❖ Number of devices = 256
❖ Due to many devices we need more decoder hardware in Memory Mapped IO.	❖ Due to less devices we need less decoder hardware in IO Mapped IO.
❖ Due to limited Address lines (16) with 8085 available memory less with memory mapped IO.	❖ Available memory more with IO mapped IO as number of devices are less in IO mapped IO.
❖ Here we use $\overline{MEMR}$ and $\overline{MEMW}$ control signals.	❖ Here we use $\overline{IOR}$ and $\overline{IOW}$ control signals.
❖ Data transfer happens between any registers and IO.	❖ Data transfer only between Accumulator and IO.
❖ ALU operation is done with all registers.	❖ Not Available.
❖ Instructions Example: LDA XXXXH, STA XXXXH, MOV A,M	❖ Instructions Example: IN XXXXH, OUT XXXXH

Operation Types in a Microprocessor

- All the operations in microprocessor 8085 can be classified into three types:
  - Microprocessor initiated Operations
  - Internal Operations
  - Peripheral initiated operations

- **Microprocessor initiated Operations**
  - Mainly microprocessor 8085 initiates four operations
    - Memory Read
    - Memory Write
    - IO Read [Get data from input]
    - IO Write [Send data to output]

- **Internal Operations of Microprocessor**
  - Following Operations are performed by Microprocessor internally:
    - Arithmetic and Logic Operation
    - Instruction Decode
    - Timing and control
    - Sequence of execution
    - Testing of Condition

- **Peripheral Initiated Operations**
  - Following Operations are initiated by peripheral to microprocessor:
    - RESET
    - Ready
    - Hold
    - Interrupt

Machine Cycles in Microprocessor 8085

❖ For execution of instructions, there are 5 different machine cycles with microprocessor 8085. With each machine cycle microprocessor transfer one Byte data:

- ❑ Opcode Fetch [4T/6T States]
  - ▪ It is fundamental machine cycle associated with all instructions and it is 1st machine cycle of all instructions.
- ❑ Memory Read [3T States]
- ❑ Memory Write [3T States]
- ❑ IO Read [3T States]
- ❑ IO Write [3T States]

Machine Cycle	$IO/\overline{M}$	S1	S0	Control Signals
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
IO Read	1	1	0	$\overline{RD} = 0$
IO Write	1	0	1	$\overline{WR} = 0$
Interrupt Acknowledge	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD} = \overline{WR} = Z$ $\overline{INTA} = 1$
Hold	Z	X	X	
Reset	Z	X	X	

Opcode Fetch Timing Diagram in 8085
Timing Diagram of MOV instruction

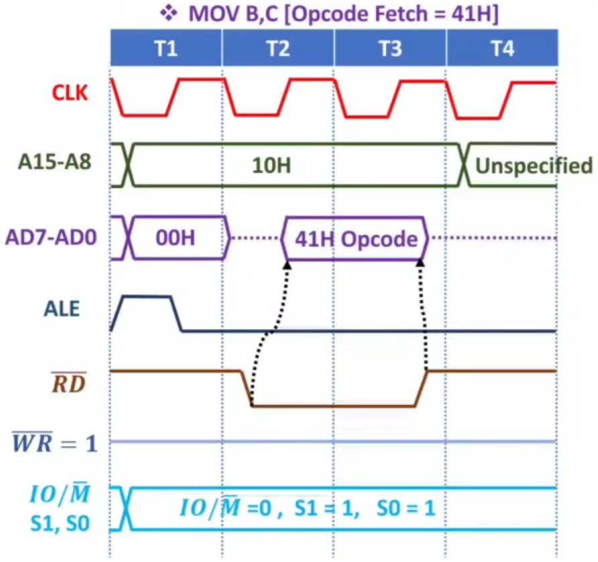
❖ MOV B,C [Opcode =41H]

PC = 1000H
8085
C= 24H

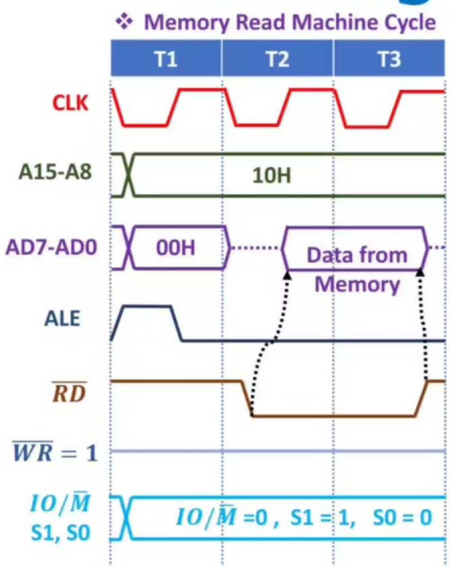
Add	Data
1000H	41H

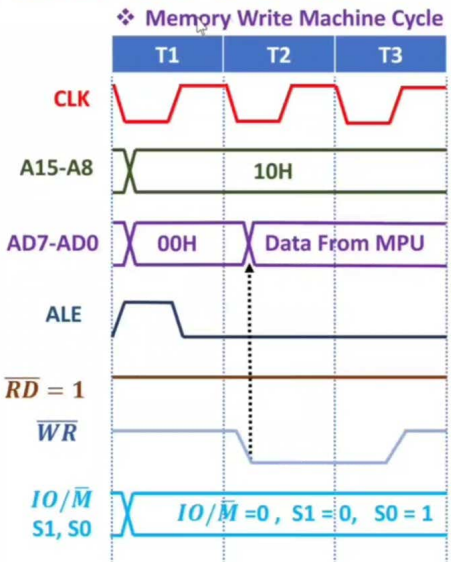
❖ MOV B,C executions needs only opcode fetch machine cycle.
❖ So, After instruction, B = 24H

❖ MOV B,C [Opcode Fetch = 41H]



Memory Read & Memory Write Timing Diagram in 8085





Timing Diagram of MVI instruction in 8085

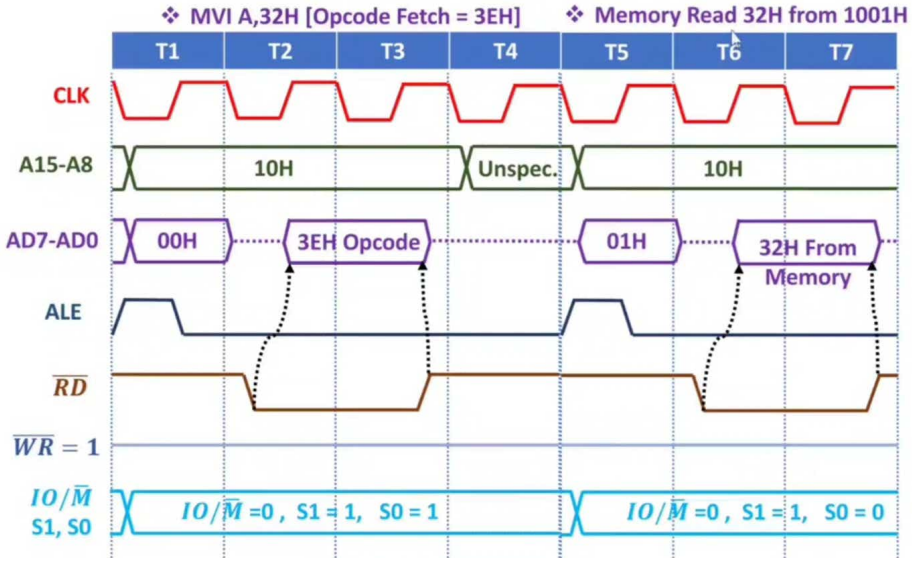
❖ MVI A,32H [Opcode =3EH]

PC = 1000H
8085
A= 24H

Add	Data
1000H	3EH
1001H	32H

❖ MVI A,32H executions needs opcode fetch & Memory Read machine cycle.

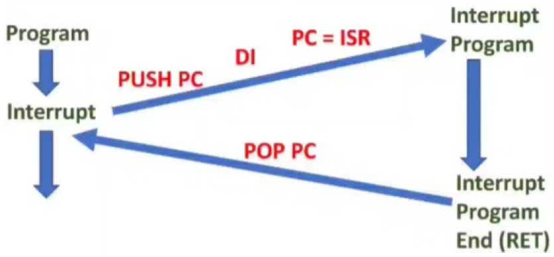
❖ So, After instruction
A = 32H



Interrupt Service Routine in 8085

Interrupt in 8085

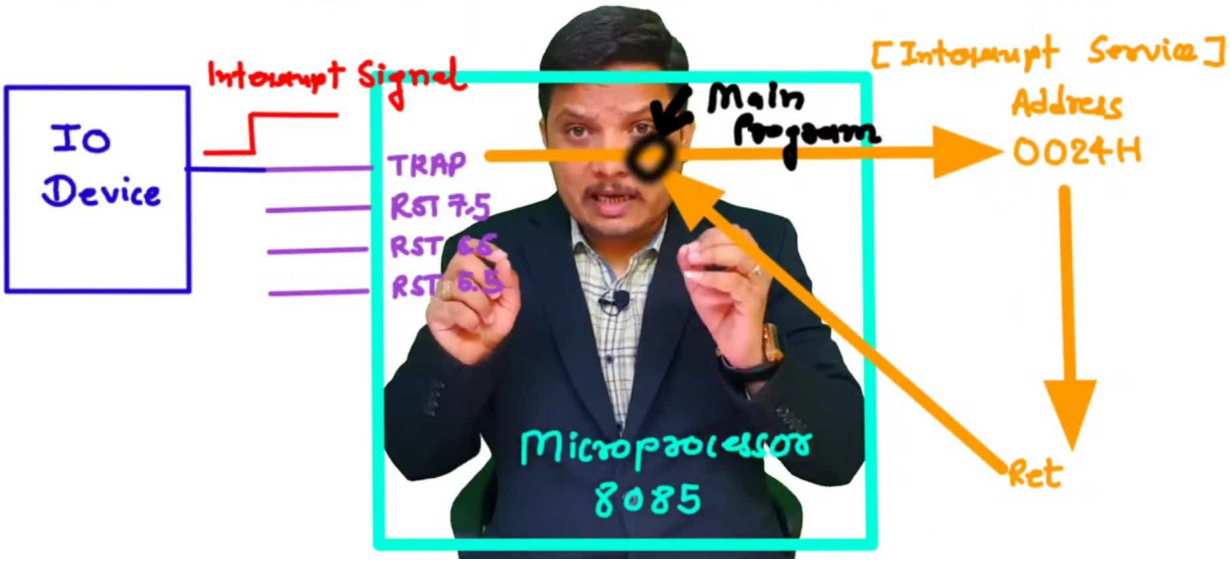
- An Interrupt is special condition that makes the 8085 to execute ISR [Interrupt Service Routine].
- The Microprocessor checks Interrupt during every instructions.
- When Interrupt occurs, The Microprocessor 8085 first completes current instruction.
- After that it pushes PC on the Stack.
- After that it resets the INTE Flip flop so that no more interrupts recognize.
- After that program control transfers to ISR.



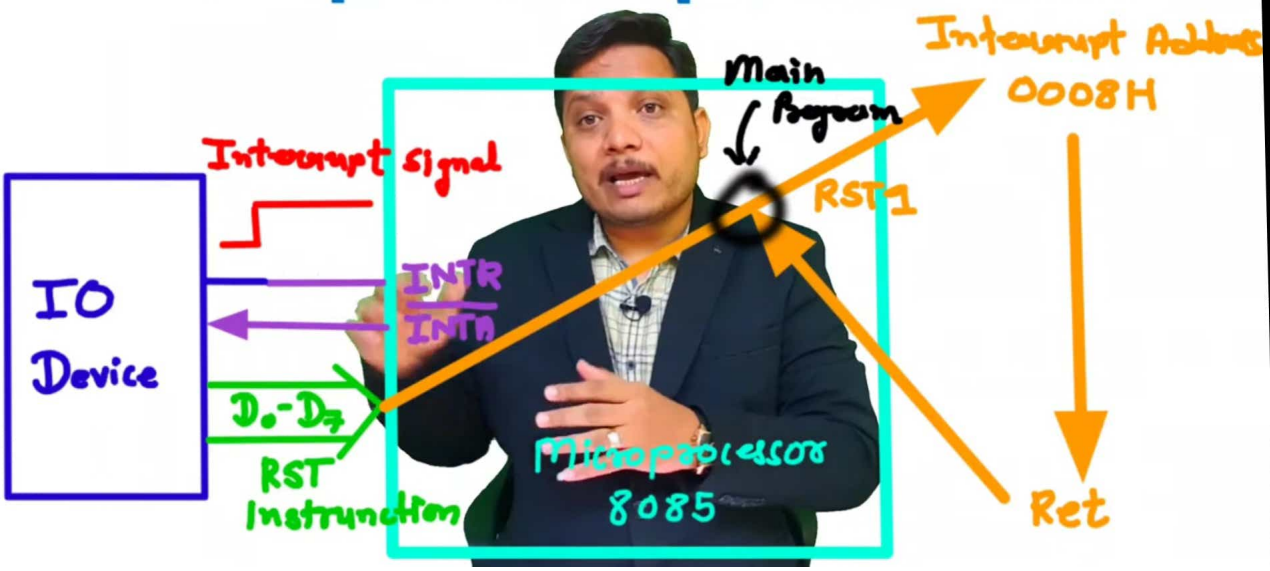
Interrupt Steps in 8085

- Step - 1 - EI instruction enable interrupt process and DI instruction disables the interrupt process.
- Step - 2 - During execution of program 8085 checks INTR line during each instructions.
- Step - 3 - If INTR goes high and the interrupt is enabled then 8085 completes current instruction. 1st 8085 disables Interrupt FF and sends active low signal on $\overline{\text{INTA}}$. Processor can not take interrupt until Interrupt FF is enabled.
- Step - 4 - $\overline{\text{INTA}}$ is used to insert RST or CALL instruction through external hardware.
- Step - 5 - Once 8085 receives RST or CALL instruction, it saves PC on stack. The program is transferred to location defined by RST or CALL.
- Step - 6 - Program on that RST or CALL is referred Interrupt service routine.
- Step - 7 - Interrupt service routine should include EI instruction to enable interrupt again.
- Step - 8 - At the end of subroutine, RET instruction retrieves the PC from stack.

Software Interrupt and Hardware Interrupt in Microprocessor 8085



Software Interrupt and Hardware Interrupt in Microprocessor 8085



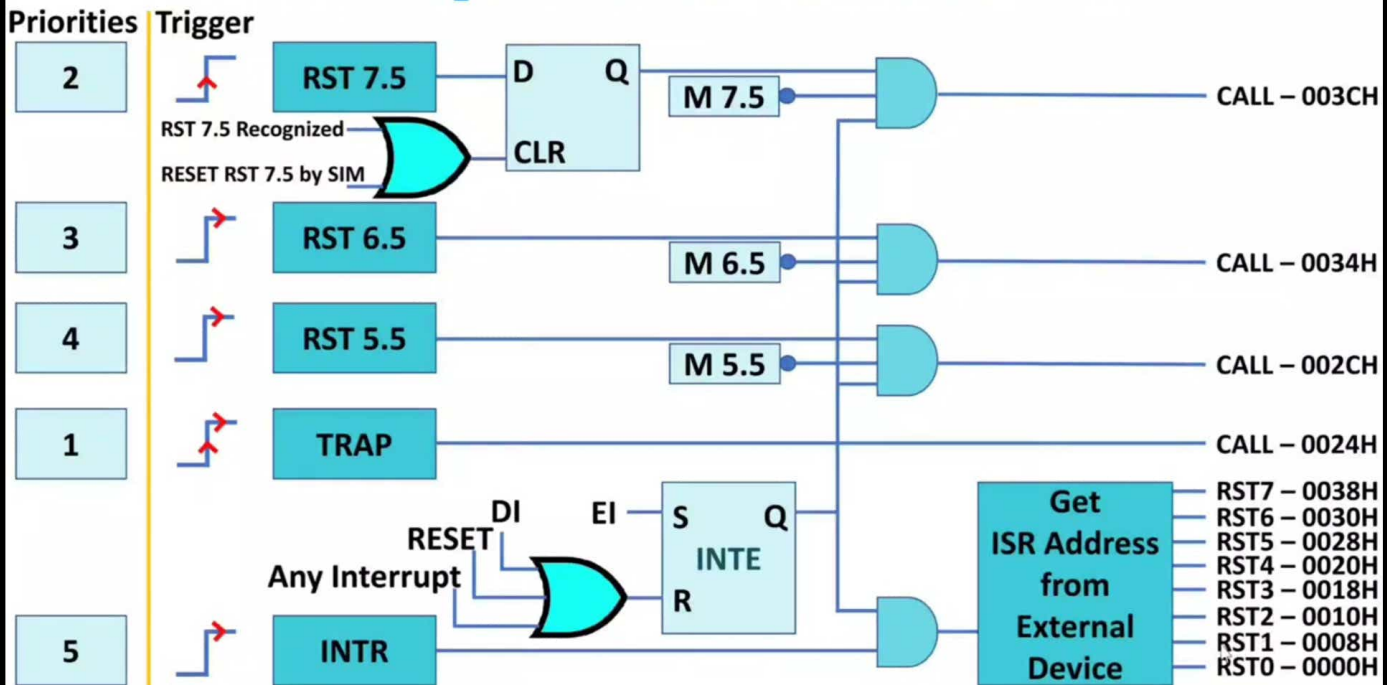
Software Interrupt and Hardware Interrupt in Microprocessor 8085

Software Interrupt	Hardware Interrupt
❖ It is invoked by Microprocessor 8085.	❖ It is invoked by External Hardware.
❖ There are total 8 Interrupt instruction with 8085, RST0 to RST7.	❖ There are total 5 hardware interrupt available on 8085 Pins : INTR, RST 5.5, RST 6.5, RST 7.5 & TRAP.
❖ There is equal priority.	❖ Highest with TRAP then RST 7.5, RST 6.5, RST 5.5 to INTR with lowest priority.
❖ It is Triggered by Instructions.	❖ It can be Edge Trigger or Level Trigger.
❖ It can be Disabled.	❖ It can be Disabled except TRAP.
❖ All software Interrupt are vectored.	❖ All hardware interrupt are vectored except INTR.
❖ Vector Address are: ❑ RST0 – 0000H ❑ RST1 – 0008H ❑ RST2 – 0010H ❑ RST3 – 0018H ❑ RST4 – 0020H ❑ RST5 – 0028H ❑ RST6 – 0030H ❑ RST7 – 0038H	❖ Vector Address are: ❑ TRAP – 0024H ❑ RST 7.5 – 003CH ❑ RST 6.5 – 0034H ❑ RST 5.5 – 002CH ❑ INTR – Obtain from device

Hardware Interrupt in Microprocessor 8085

Interrupt	Priority	Trigger	Mask by SIM	Disable by DI	Vector Address
TRAP	1	E/L	Nope	Nope	0024H
RST 7.5	2	E	Yes	Yes	003CH
RST 6.5	3	L	Yes	Yes	0034H
RST 5.5	4	L	Yes	Yes	002CH
INTR	5	L	Nope	Yes	Given by External Hardware

Interrupt Structure in 8085



Addressing Modes in Microprocessor 8085

❖ Addressing mode: The Various formats of specifying the operands are called addressing modes.

✓ Immediate Addressing Mode

❑ In this Addressing mode, data (1byte/2bytes) specified in instruction is directly transferred into register.

Example:

```
MVI C,15H        ; 15H will gets transferred to C
LXI B,1000H      ; 1000H will gets transferred to BC pair
```

✓ Register Addressing Mode

❑ In this Addressing mode, data is specified in registers.

Example:

```
MOV A,C          ; C will gets copied into A
INR D            ; D will get incremented by 1
```

✓ Direct Addressing Mode

❑ In this Addressing mode, address of operand is specified in instruction.

Example:

```
LDA 1000H        ; A will get data from 1000H address
STA 1000H        ; A will get stored at address 1000H
```

✓ Indirect Addressing Mode

❑ In this Addressing mode, the address of operand is stored in registers.

Example:

```
STAX B           ; Store the content of A into address
                   pointed by BC Pair
INR M            ; increment the content located by
                   memory HL pair
```

✓ Implied/Implicit Addressing Mode

❑ In this Addressing mode, the operand is implied in instruction.

Example:

```
STC              ; Set the carry flag
CMC              ; Complements the carry flag
```

Data Transfer Instructions in 8085

✓ MOV $R_{Destination}, R_{Source}$ [1 Byte]

- ❑ It will transfer Data of Source register into Destination register.

Example:

MOV A,B ; B will get copied into A

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	None	1	4

✓ MOV $R_{Destination}, M$ [1 Byte]

- ❑ It will transfer Data of Memory (Located by HL pair) to Destination Register.

Example:

MOV A,M ; A = [HL]

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	None	2	7

✓ MVI $R_{Destination}$,8 bits Data [2 Byte]

- ❑ It will transfer Data (Given in instruction) into Destination register.

Example:

MVI A,55H ; A = 55H

Addressing Mode	Flags Effected	Machine Cycles	T States
Immediate	None	2	7

✓ MVI M, 8 bits Data [2 Byte]

- ❑ It will transfer Data (Given in Instruction) into Memory located by HL pair.

Example:

MVI M,55H ; [HL] = 55H

Addressing Mode	Flags Effected	Machine Cycles	T States
Immediate	None	3	10

Data Transfer Instructions in 8085

✓ LXI Rp,16 bits Data [3 Byte]

❑ It will transfer Data (2 Bytes) into register pair (BC, DE or HL).

Example:

LXI B,2350H ; B=23H & C=50H

Addressing Mode	Flags Effected	Machine Cycles	T States
Immediate	None	3	10

✓ LDA 16 bits Address [3 Byte]

❑ It will transfer data of given address into Accumulator register.

Example:

LDA 5000H ; A = [5000H]

Addressing Mode	Flags Effected	Machine Cycles	T States
Direct	None	4	13

✓ STA 16 bits Address [3 Byte]

❑ It will transfer Data of Accumulator into given Address of instruction.

Example:

STA A,5000H ; [5000H]=A

Addressing Mode	Flags Effected	Machine Cycles	T States
Direct	None	4	13

✓ LHLD 16 bits Address [3 Byte]

❑ It will load HL pair from given Memory Address given by instruction.

Example:

LHLD 5000H ; L=[5000H], H=[5001H]

Addressing Mode	Flags Effected	Machine Cycles	T States
Direct	None	5	16

Data Transfer Instructions in 8085

✓ SHLD 16 bits Address [3 Byte]

- ❑ It will store HL pair into memory location specified in the instruction

Example:

SHLD 5000H ; [5000H]=L & [5001H]=H

Addressing Mode	Flags Effected	Machine Cycles	T States
Direct	None	5	16

✓ LDAX Rp [1 Byte]

- ❑ Accumulator is loaded with the content located by memory location pointed by Register Pair.

Example:

LDAX B ; A = [BC]

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	None	2	7

✓ STAX Rp [1 Byte]

- ❑ Accumulator is stored at the location pointed by Register Pair.

Example:

STAX B ; [BC]=A

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	None	2	7

✓ PCHL [1 Byte]

- ❑ Program counter gets the content of HL Pair. It will create branch in the program.

Example:

PCHL ; PC=HL

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	None	1	6

Data Transfer Instructions in 8085

✓ SPHL [1 Byte]

- Stack Pointer gets the content of HL register pair. It will relocate the stack.

Example:

```
SPHL            ; SP=HL
```

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	None	1	6

✓ XCHG [1 Byte]

- It will exchange the content of HL and DE pair.

Example:

```
XCHG            ; HL ⟷ DE
```

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	None	1	4

✓ XTHL [1 Byte]

- It will exchange content of HL pair with the content of SP and SP+1.

Example:

```
XTHL            ; L⟷[SP] and H⟷[SP+1]
```

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	None	5	16

Arithmetic Instructions in 8085

✓ ADD R [1 Byte]

- It will add Register R with A and answer will get store in Accumulator.

Example:

ADD C ; A=Ai+C

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	ALL	1	4

✓ ADD M [1 Byte]

- It will add A with content pointed by memory (HL pair) and stores in Accumulator

Example:

ADD M ; A = Ai+[HL]

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	All	2	7

✓ ADI 8bits Data [2 Byte]

- It will add A with 8 bits data given in instruction and answer will get store in Accumulator.

Example:

ADI 55H ; A=Ai+55H

Addressing Mode	Flags Effected	Machine Cycles	T States
Immediate	All	2	7

✓ ADC R [1 Byte]

- It will add A with given register R along with Carry and answer will store in Accumulator.

Example:

ADC C ; A=Ai+C+Carry

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	All	1	4

Arithmetic Instructions in 8085

✓ **ADC M [1 Byte]**

❑ It will add A with data pointed by HL pair along with carry and answer will get store in Accumulator.

Example:

ADC M ; A=Ai+[HL]+Carry

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	ALL	2	7

✓ **ACI 8bits data [2 Byte]**

❑ It will add A with data given instruction along with carry and stores in Accumulator.

Example:

ACI 55H ; A = Ai+55H+Carry

Addressing Mode	Flags Effected	Machine Cycles	T States
Immediate	All	2	7

✓ **Similarly, Subtraction also done with**

✓ SUB R [1 Byte]
✓ SUB M [1 Byte]
✓ SUI 8bits data [2 Byte]
✓ SBB R [1 Byte]
✓ SBB M [1 Byte]
✓ SBI 8bits data [2 Byte]

Arithmetic Instructions in 8085

✓ INR R [1 Byte]

❑ It will increment the value of given register by 1. Answer will be there in same register.

Example:

INR C ; C=Ci+1

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	All except C	1	4

✓ INR M [1 Byte]

❑ It will increment the value of content pointed by HL pair and load it at same location.

Example:

INR M ; [HL]=[HL]i+1

Addressing Mode	Flags Effected	Machine Cycles	T States
Indirect	All except C	3	10

✓ INX Rp [1 Byte]

❑ It will increment the value of register pair by 1 and stores in same register pair.

Example:

INX H ; HL = HLi + 1

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	None	1	6

✓ Similarly, Decrement also done with

✓ DCR R [1 Byte]

✓ DCR M [1 Byte]

✓ DCX Rp [1 Byte]

Arithmetic Instructions in 8085

✓ DAD Rp [1 Byte]

- It will add HL pair with given register pair and answer will get store with HL pair.

Example:

DAD B ; HL = HLi + BC

Addressing Mode	Flags Effected	Machine Cycles	T States
Register	C	3	10

- It will execute opcode fetch and two machine cycle will be Bus idle.

✓ DAA [1 Byte]

- This instruction is used to get answer in BCD form.
- It will do addition inform of decimal form.
- It is implied addressing mode and work strictly with register A.
- If Lower Nibble > 9 or AC = 1 then add 06H.
- If Higher Nibble > 9 or CY = 1 then add 60H.

Addressing Mode	Flags Effected	Machine Cycles	T States
Implied	All	1	4

- Let say we want to add (55)bcd and (55)bcd.
- To do that in 8085 execute this instructions
- MVI A,55H
- MVI B,55H
- ADD B
- DAA

55H + 55H = AAH AAH + 06H + 60H = 110H

- But that's our answer of Addition in BCD