# 8085 µP

## Buses in 8085 µP

→ In this, 16 address lines ($A_0 - A_{15}$)
8 Data lines ($D_0 - D_7$)

* Buses plays an important role to make a connection between µP and Memory or I/O unit.

### Address lines

→ Address lines are unidirectional lines.
→ Range is from 0000H to FFFFH. It can interface 64K addresses.

### Data line

→ Data bus is bidirectional
→ 1 byte data is exchanges with single operation.

### Control line

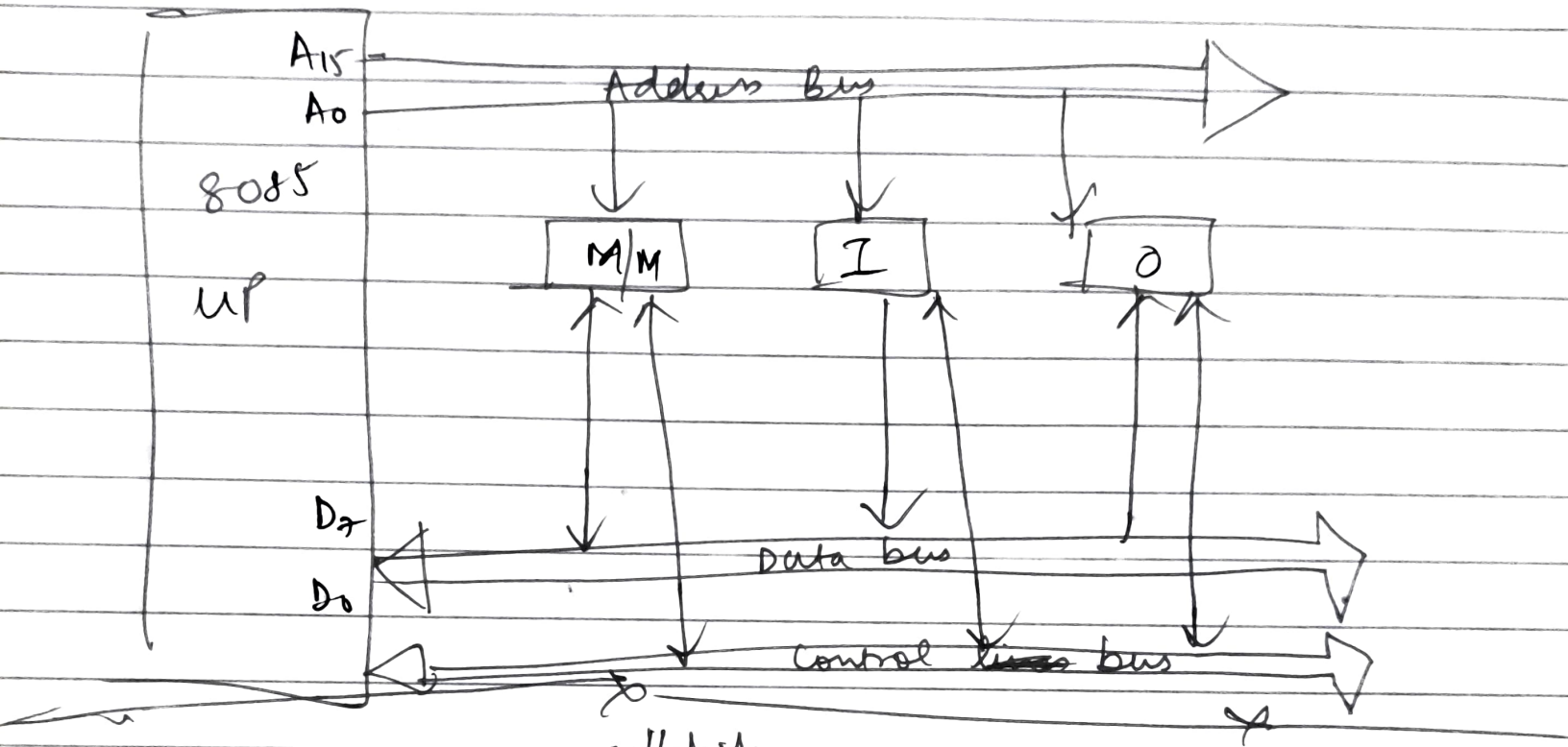
→ Control lines are bidirectional
→ Example MERD → memory Read
MEWR → memory write
IORD → I/O Read
IO WR → I/O Write.

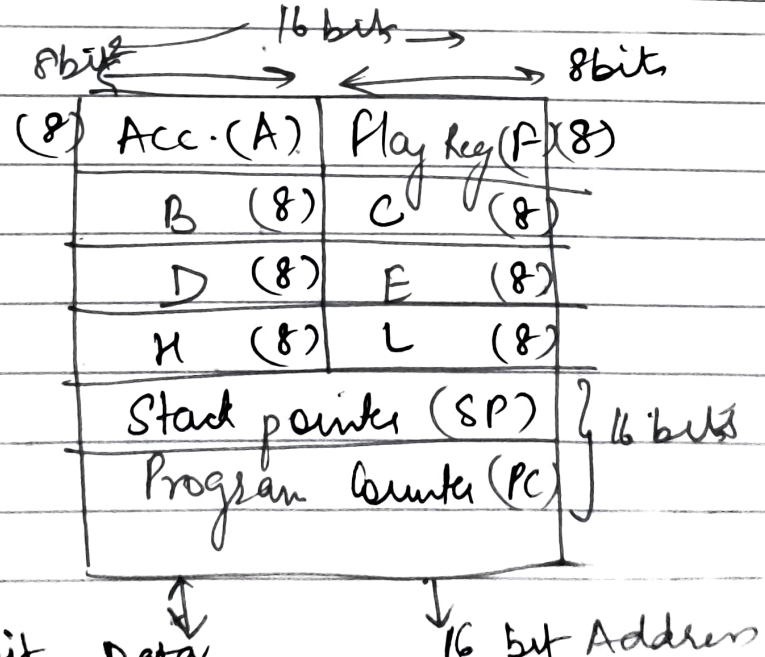
## Programming Model

→ Two types of registers
- 8 bit: A, Flag Reg., B, C, D, E, H, L
- 16 bit: (BC, DE, HL), Stack pointer, Program Counter.

ALU → It is 8 bit register.
→ ALU is used to perform various operations. (Addition, Sub, multiplication, shifting, etc.)

# µP Buses


A15
A0
8085
µP
Address bus
M/M
I
O
D7
D0
Data bus
Control bus


16 bits
8bit
8bit
(8) Acc.(A)
Flag Reg (F) (8)
B (8)
C (8)
D (8)
E (8)
H (8)
L (8)
Stack pointer (SP)
Program Counter (PC)
16 bits
8bit Data
16 bit Address

Flag Reg → It shows status of program.

B, C, D, E, H, L → Individually - 8 bits
→ In pair i-e BC, DE, HL as 16 bits register.

Stack pointer → It points the stack memory.

Program counter → It points the address at which next instruction will get executed.
→ It loads the next instruction address

## ✳ Flag Register

↳ shows the status of program
↳ It is of 8 bits.
↳ There are 5 flag bits in 8085.

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| S | Z | _ | AC | _ | P | _ | CY |

↳ $D_5$, $D_3$, $D_1$ are unused bits in flag reg.

- S - Sign Flag

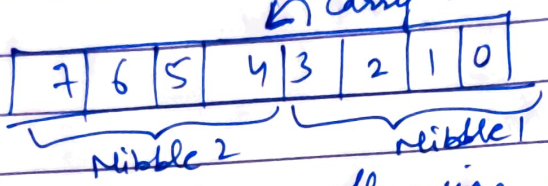
→ It is associated with MSB of accumulator
MSB (Most significant bit)
→ if MSB = 0, then data is positive
if MSB = 1, then data is negative

- **Z flag** (Zero flag)

During execution if accumulator is set to 00H then ~~accumulator~~ Zero flag set to 1, otherwise Z = 0.

- **AC** (Auxiliary Carry)

→ Nibble to Nibble carry in operation.
(Nibble means 4 bits)




then AC = 1, otherwise 0.

- **P flag** (Parity flag)

→ Even parity = 1
→ Odd parity = 0

- **CY** (Carry flag)

→ It shows carry in operation of ALU.

## Example

Reg H = BC, Reg A = DD

execute → ADD H

AC = 1

| | | | |
|---|---|---|---|
| H | = 25 / 16 / 9 | BC = 12 / 13 | = 25 / 16 / 9 |
| A | = | DD = 13 | |
| A | = | 1 9 9 | |

| | | |
|---|---|---|
| | 1 1 1 1 | 1 |
| | 1 0 1 1 | 1 1 0 0 |
| + | 1 1 0 1 | 1 1 0 1 |
| | 1 1 0 0 1 | 1 0 0 1 |
| | 9 | 9 |

Answer → 1 1 0 0 1 1 0 0 1

MSB = 1, Z = 0, AC = 1, P = 1 (Even parity), CY = 1

# * Control Signals

**ALE → Address latch Enable**

- It is active high signal
- It is used to demultiplex $AD_0$ to $AD_7$ lines.
- If ALE = 1, $AD_0 - AD_7$ represents address
  ALE = 0, $AD_0 - AD_7$ represents data.

**$\overline{RD}$ – Read data**

- Active low signal.
- If $\overline{RD} = 0$ → MPU performs read operation.

**$\overline{WR}$ = Write data**

- Active low signal
- If $\overline{WR} = 0$ → MPU performs write operation

**IO / $\overline{M}$ – Input output / Memory**

- It is used to differentiate IO and M/M devices.
- $IO/\overline{M} = 1$, IO operation will happen
- $IO/\overline{M} = 0$, M/M operation will happen

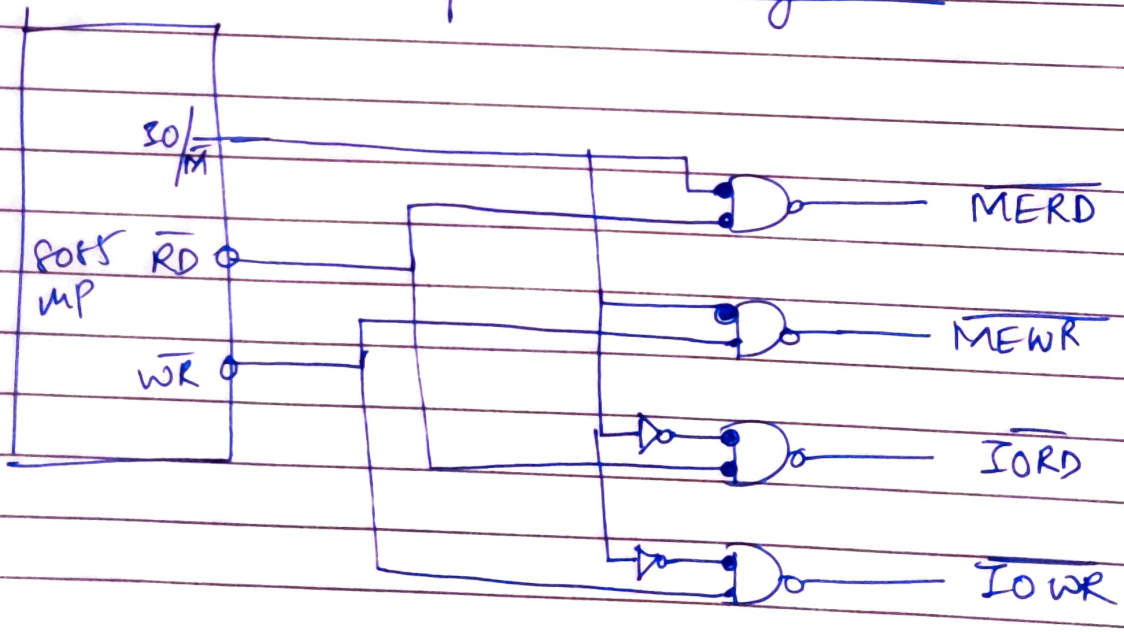
**$S_0$ & $S_1$ – Status signals**

-

| Operation | IO/$\overline{M}$ | $S_1$ | $S_0$ | Status |
|---|---|---|---|---|
| Opcode fetch | 0 | 1 | 1 | $\overline{RD}$ = 0 |
| M/M Read | 0 | 1 | 0 | $\overline{RD}$ = 0 |
| M/M Write | 0 | 0 | 1 | $\overline{WR}$ = 0 |
| IO Read | 1 | 1 | 0 | $\overline{RD}$ = 0 |
| IO write | 1 | 0 | 1 | $\overline{WR}$ = 0 |
| Interrupt Ack | 1 | 1 | 1 | $\overline{INTA}$ = 0 |
| ↳ HALT | Z | 0 | 0 | $\overline{RD}$, $\overline{WR}$ = Z |
| ↳ HOLD | Z | X | X | |
| RESET | Z | X | X | $\overline{INTA}$ = 1 |

Z = high impedence mode
X = Don't care condition

## ✱ Generation of Control Signals.




Nand why → All are active low signals

# ✱ Addressing Modes in 8085

## 1) Immediate Addressing Mode

→ In this mode, the 8 or 16 bit data is specified in the instruction itself as one of its operand.

eg:- MVI A, 2F h

A = 2F h ← output

→ In this mode data is directly transferred into register.

## 2) Direct Addressing Mode

→ In this mode, data is directly copied from given address to the register

eg:- LDA 5000H

A = 65h ← output

| |
|---|
| 4FFF h |
| 5000 h → 65H |
| 5001 h |

## 3) Indirect Addressing Mode

→ In this mode, data is transferred from memory location pointed by register.

— eg: LDAX B

| B | C |
|---|---|
| 10 | 50 |

| |
|---|
| 1050 h → 22H |

A = 22 H ← output

4) Register Addressing Mode

→ In this we copy data from one register to other register.

eg - MOV A, B

Before execution A = 10H B = 20H

After execution A = 20H B = 20H

5) Implied Addressing Mode

→ In this mode, we don't need any operand.

→ Data execution will happen with that register only

eg - CMP A

If A = CCH = 1100 1100

after execution

A = 33H = 0011 0011

# Interrupt Service Routine (ISR)

Step 1: The interrupt process should be enabled by writing the EI instruction and disabled by DI instruction.

- EI (Enable Interrupt)
  - 1 byte instruction
  - It is used to enable interrupt
- DI (Disable Interrupt)
  - 1 byte instruction.
  - It is used to disable interrupt.

Step 2: During the execution of program in 8085 microprocessor checks the INTR line during execution of all instructions

Step 3: If the INTR is high then interrupt is enabled, then ~~up~~ uP completes current instruction and then disable interrupt flip flop and then sends $\overline{INTA}$ (Interrupt Acknowledge~~ment~~ ). The processor cannot accept any interrupt request until the interrupt flip flop is enabled again.

Step 4: The $\overline{INTA}$ is used to insert RST instruction through ~~hard drive~~ external hardware

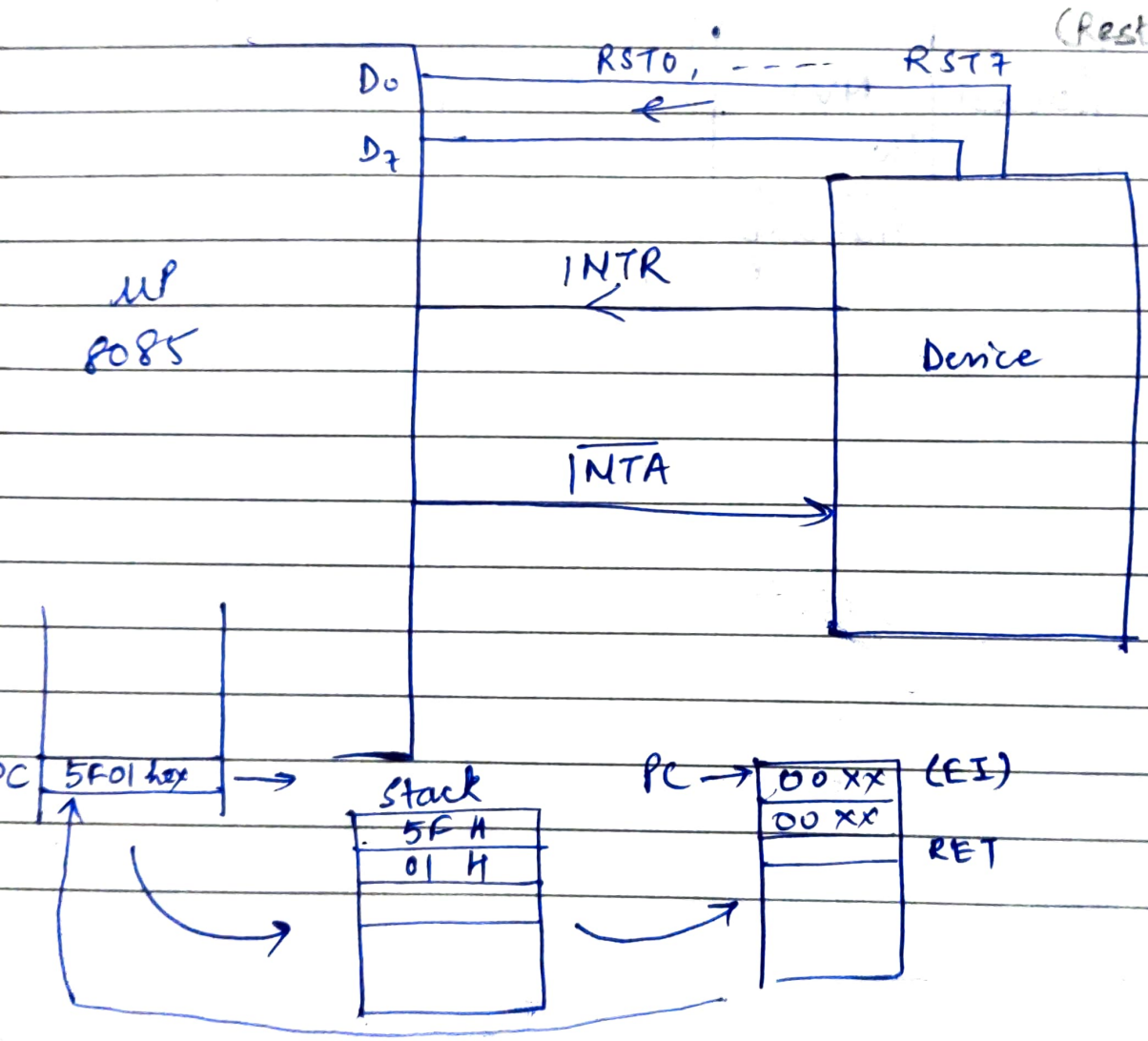
Step 5: → When uP receives RST instruction, ~~it~~ it saves the memory address to next instruction on the stack.

→ Then program control get transfers to new location on page 00H.

Step 6: After performing interrupt task, µP again jumps to original program. That subroutine is known as ISR (Interrupt Service Routine)

Step 7: The ISR should include EI at the beginning.

Step 8: At the end of ISR, RET instruction retrives the memory address at original address.



# * Hardware Interrupts in 8085 . uP

Interrupt is a signal to the processor, generated by hardware / software indicating an immediate attention needed by an event.

5 hardware interrupts are there in 8085

| No. | Name | Priority | Vector Add. | Masking | Types of trigger |
|---|---|---|---|---|---|
| 1 | TRAP | Highest | 0024H | NMI | edge & level |
| 2 | RST 7.5 | ↓ | 003C H | Maskable | edge trigger |
| 3 | RST 6.5 | ↓ | 0034H | | level trigger |
| 4 | RST 5.5 | ↓ | 002CH | | |
| 5 | INTR | Lowest | NVI (Non vectored interrupt) | | |

# INSTRUCTION SET FOR 8085 µP

There are 5 groups of instructions.

(I) Data Transfer Instructions

→ These instructions are used to transfer data in between register to register and register to memory.

(1) MOV $R_1, R_2$ (transfer data from $R_2$ to $R_1$)

example: MOV A, B

if B = 20 H

then, after execution

A = 20H

(2) MOV M, R (transfer register data to memory).

example: H = 30H

L = 40H

M = 3040H , B = 30H

After execution: MOV M, B

M → 3040H = 30H

(3) MOV R, M (transfer memory data to Register)

H = 40H

L = 60H

M = 4060H = 11H data

After execution MOV C, M

C = 11H

(4) MVI R, data (8 bit) { 8 bit It moves data into register R }

Example: MVI B, 77H
After execution B = 77H

(5) MVI M, data (8 bit) { It transfer 8 bit data into memory }

example:
MVI M, 77H
HL = 3040H = 77H

(6) LXI SP, 8 bit data { data is transferred to stack pointer }

example: LXI SP, 77H
SP → 2010 H = 77H

## (II) Arithmetic Instructions

(1) Addition

(a) ADD R { Addition with Acc. and result is stored in Accumulator }

A = A + R

(b) ADD M { Memory data is indicated by HL pair }
A = A + M

(c) ADI
A = A + data (8 bit).

(d) ADC R

A = A + R + CY

(e) ADC M

A = A + M + CY

(f) ADC data (8 bit)

A = A + data (8 bit) + CY

## (2) Subtraction

(a) SUB R

A = A - R

(b) SUB M

A = A - M

(c) SUI (8 bit data)

A = A - 8 bit data

(d) SBB R

A = A - R - CY (CY = Borrow)

(e) SBB M

A = A - M - CY

(f) SBI (8 bit data)

A = A - 8 bit data - CY

## (3) Increment

(a) INR R
$R = R+1$

(b) INR M
$M = M+1$

(c) INX $R_p$ ($R_p$ = Register pair data)
$R_p = R_p+1$

## (4) Decrement

(a) DCR R
$R = R-1$

(b) DCR M
$M = M-1$

(c) DCX $R_p$
$R_p = R_p - 1$

## (5) Special Arithmetic Instructions

DAD $R_p$
$HL = HL + R_p$

## (III) Logical Instructions

### (1) AND Operation

(a) ANA R
A = A AND R

(b) ANA M
A = A AND M

(c) ANI (8bit data)
A = A AND 8bit data

### (2) OR operation

(a) ORA R
A = A OR R

(b) ORA M
A = A OR M

(c) ORI (8bit data)
A = A OR 8 bit data.

### (3) XOR operation

(a) XRA R
A = A ⊕ R

(b) XRA M
A = A ⊕ M

(c) XRI 8bit data
A = A ⊕ 8 bit data

## (4) Comparision

(a) CMP R — {Compare Acc. and Reg.}
A - R is performed if
A > R, sub is +ive, CY = 0, Z = 0
A < R, sub is -ive, CY = 1, Z = 0
A = R, CY = 0, Z = 1

(b) CMP M — {compare memory data with Acc.}

(c) CPI (8 bit)

## (5) Rotate

(a) RLC — {Rotate data left without carry}
(b) RRC — {Rotating data right without carry}
(c) RAL — {Rotating data left with carry}
(d) RAR — {Rotating data right with carry.

## (6) Special operation

(a) CMA — {complementing Acc.}
$A = \bar{A}$

(b) CMC — {complementing carry}
$CY = \bar{C}$

(c) STC — {used to set carry}
CY = 1

## (IV) Branching Instructions

- **Jump**
  - Unconditional
    - JMP (Address)
  - Conditional
- **Call**
  - Unconditional
    - CALL (Address)
  - Conditional
- **Return**
  - Unconditional
    - RET (Address)
  - Conditional

# Conditional Jump, call, return.

| Condition | Jump | Call | Return. |
|---|---|---|---|
| Z=0 | JNZ Address | CNZ Add. | RNZ Add. |
| Z=1 | JZ Add. | CZ Add. | RZ Add. |
| C=0 | JNC Add. | CNC Add. | RNC Add. |
| C=1 | JC Add. | CC Add. | RC Add. |
| P=1 | JPE Add. | CPE Add. | RPE Add. |
| P=0 | JPO Add. | CPO Add. | RPO Add. |
| S=0 | JP Add. | CP Add. | RP Add. |
| S=1 | JM Add. | CM Add. | RM Add. |

Note :- (1) for call instruction

$SP \rightarrow SP - 2$

(2) for return instruction

$SP \rightarrow SP + 2$

## (V) Special Instructions

### (1) SPHL

Copy HL register into stack pointer.

example: If H = 25H, L = 35H

After execution of SPHL

SP → 2535H

Address

### (2) PCHL

Load program counter with HL pair.

example:

If H = 01H, L = FFH

After execution of PCHL

PC → 01FFH

Address

### (3) SHLD

Store HL pair direct

example: If H = 01H, L = FFH

After execution

if SHLD 2050H

2050H → L = FFH

2051H → H = 01H

### (4) LHLD

Load HL pair direct.

example: If 2001 = 90H

2002 = 01H

~~After execution~~
~~LHLD 2050 H~~

After execution
LHLD 2001 H
→ L = 90H
H = 01 H

(5) XTHL
exchange HL pair with top of stack

Example: If H = A2H, L = 57H
SP → 2001 H = 25H
2002 H = 35H

After execution of XTHL
L = 25 H and SP → 2001H = 57H
H = 35H 2002H = A2H

(6) STC
→ set carry instruction
→ CY = 1

(7) NOP
→ No operation
→ It is utilized to provide delay.

# (V) RIM & SIM Instruction.

## (1) RIM (Read Interrupt Mask)

→ This is multipurpose instruction used to read the status of Interrupts 7.5, 6.5, 5.5 and serial data input bit.
→ This instruction loads eight bit data of accumulator with following interpretations :

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| SID | $I_7$ | $I_6$ | $I_5$ | IE | 7.5 | 6.5 | 5.5 |
| Serial Input data bit | Interrupts pending if =1 | | | Interrupt Enable =1 | Interrupt is Masked if =1 | | |

→ If Accumulator data A = 49H, RIM

0 1 0 0 1 0 0 1

SID = 0 ← 0 1 0 0 → 1 → 0 0 1

- 0 1 0 0 : RST 7.5 is Pending
- 1 : IE flag is set
- 0 0 1 : RST 7.5, 6.5 (enable); RST 5.5 (Masked)

## (2) SIM (Set Interrupt Mask)

→ This instruction is used to implement interrupts (RST 7.5, 6.5 & 5.5) and serial data output.

→ This instruction interrupts following data of accumulator.

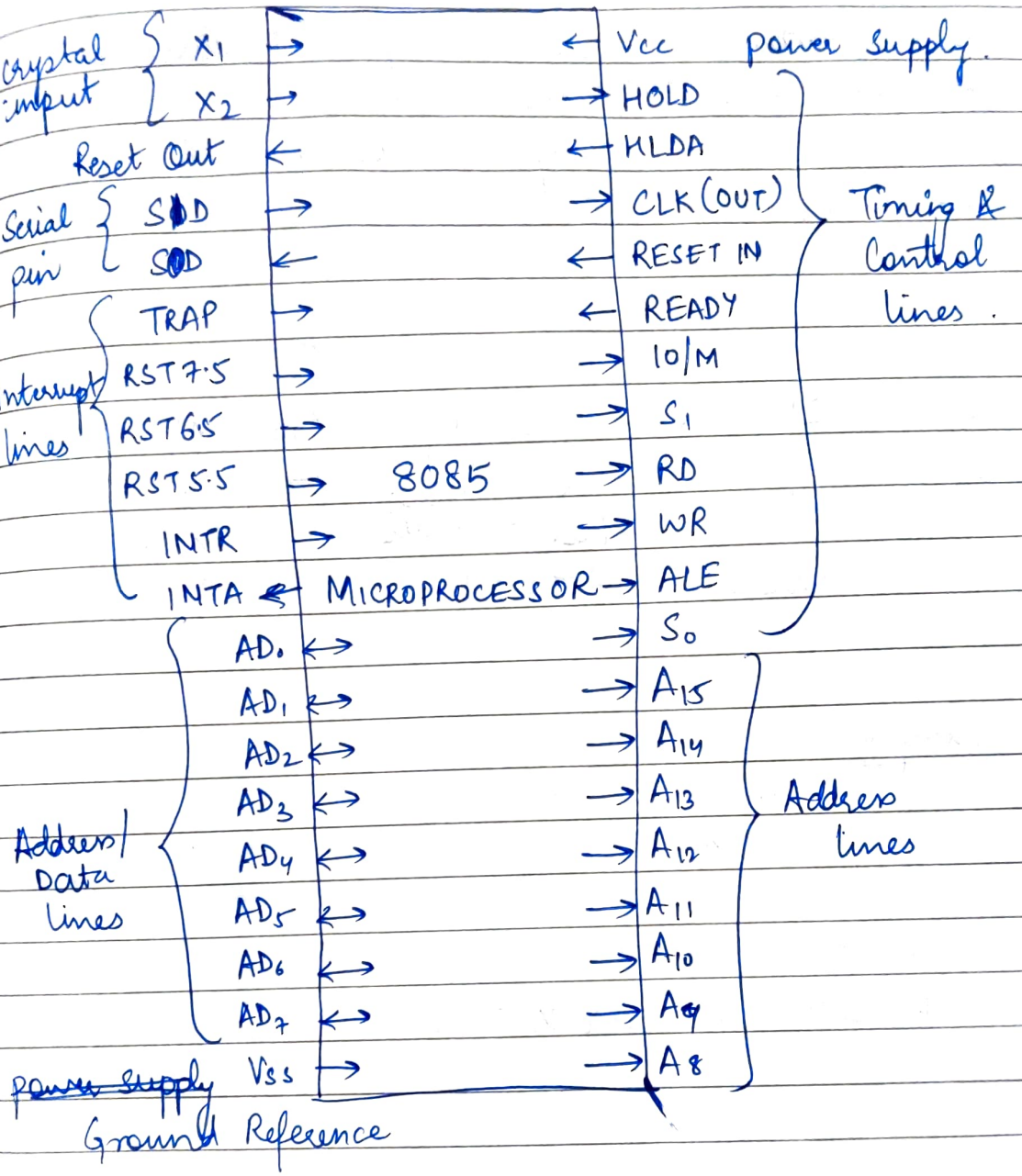
| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| SOD | SDE | X | R7.5 | MSE | M7.5 | M6.5 | M5.5 |
| Serial Out data | Serial data enable | | Reset RST 7.5=1 | Mask enable=1 | Masks if bit = 1 | | |

→ example

```
MVI A, 0EH
SIM
```

| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SOD | SDE | X | R7.5 | MSE | M7.5 | M6.5 | M5.5 |

# 8085 Pin Configuration




→ Crystal input :- It is used to tune the internal frequency of 8085 µP.

→ RESET IN :- • It is active low signal.
• When signal on this pin is low for atleast 3 clocking cycles, it forces the microprocessor to reset itself.

→ RESET OUT

- It is an active high signal.
- The output on this pin goes high whenever RESET IN is given low signal.
- The output remains high as long as RESET IN is kept low.

→ SID (Serial Input data)

- Store the bit at 8th position (MSB) of Accumulator.
- RIM (Read Interrupt Mask) instruction is used to transfer the bit.

→ SOD (Serial Output Data)

- Takes the bit from the 8th position (MSB) of Accumulator.
- ~~Tak~~ SIM (Set Interrupt Mask) instruction is used to transfer the bit.

* Note: Maskable Interrupts are those interrupts which can be enabled or disabled.

→ HOLD

It indicates that another device is requesting to use of the address and data bus. Having received HOLD request the uP relinquishes the use of the buses as soon as the current machine cycle is completed. Internal processing may continue. After the removal of the HOLD signal the processor regains the bus.

→ HLDA

→ uP uses this pin to acknowledge the receipt of HOLD signal.

→ When HLDA signal goes high, address bus, data bus, RD, WR, IO/M pins are cut-off from external environment (tri-stated).