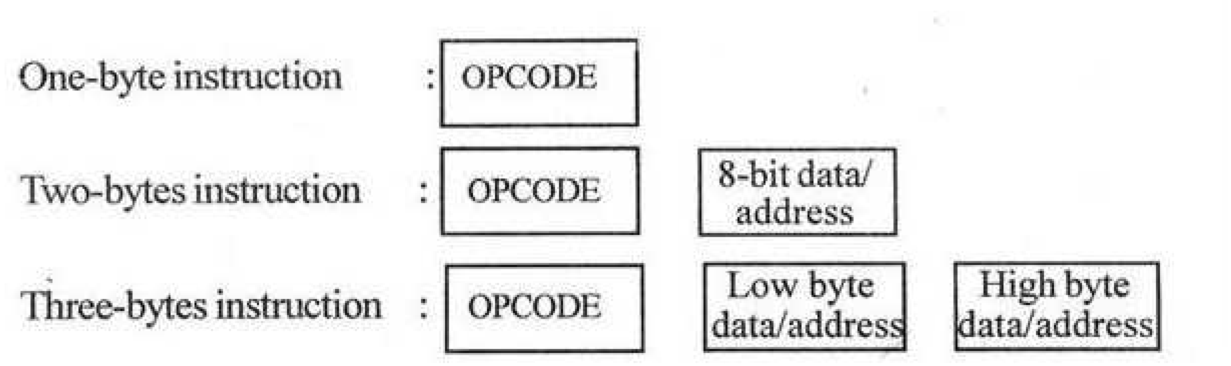
Unit – III 8085 Assembly Language Programming

(1) Explain instruction format and Opcode format of 8085 µP with example. OR With help of examples, explain the formation of opcodes of 8085 OR What is an instruction? List type of instruction based on size.

Each instruction of 8085 has 1 byte opcode. With 8 bit binary code, we can generate 256 different binary codes. In this, 246 codes have been used for opcodes.


One-byte instruction : OPCODE
Two-bytes instruction : OPCODE
8-bit data/ address
Three-bytes instruction : OPCODE
Low byte data/address
High byte data/address

The size of 8085 instructions can be 1 byte, 2 bytes or 3

bytes. The 1-byte instruction has an opcode alone.

The 2 bytes instruction has an opcode followed by an eight-bit address or data.

The 3 bytes instruction has an opcode followed by 16 bit address or data. While storing the 3 bytes instruction in memory, the sequence of storage is, opcode first followed by low byte of address or data and then high byte of address or data.

2. Explain the addressing mode of 8085. OR What do you mean by addressing mode? Explain diff. addressing mode for 8085 with examples.

Every instruction of a program has to operate on a data. Data may be direct in instruction, in Register or in Memory. The method of specifying the data into the instruction is called Addressing mode.

The 8085 has the following 5 different types of addressing.
1. Immediate Addressing mode
2. Register addressing mode
3. Direct Addressing mode

4. Indirect Addressing mode
5. Implied Addressing mode

Immediate Addressing

In immediate addressing mode, the data is specified in the instruction itself. The data will be a part of the program instruction. All instructions that have ‘ I ’ in their mnemonics are of immediate addressing type. For Example, MVI B, 3EH - Move the data 3EH given in the instruction to B register.

Register Addressing

In register addressing mode, the instruction specifies the name of the register in which the data is available. This type of addressing can be identified by register names (such as ‘A’, ‘B’ etc.) in the instruction. For Example, MOV A, B -Move the content of B register to A register.

Direct Addressing

In direct addressing mode, the data will be in memory. The address of the data is specified in the instruction directly.
For Example, LDA 1050H - Load the data available in memory location 1050H in accumulator.

Indirect Addressing

In indirect addressing mode, the data will be in memory. The address of the data is specified in the instruction indirectly i.e. address is store in Registers. This type of addressing can be identified by letter ‘M’ present in the instruction.

For Example: MOV A, M - The memory data addressed by HL pair is moved to A register.

Implied Addressing

In implied addressing mode, there is no operand. i.e. This type of instruction does not have any address, register name, immediate data specified along with it.

For Example, CMA - Complement the content of accumulator.

(3) Explain the classification of instructions of 8085 on the basis of their operation OR Give classification of 8085 instruction set with an example of each

The 8085 instruction set can be classified into the following six functional group.

1) Data Transfer Instructions
2) Arithmetic Instructions
3) Logical Instructions
4) Branching Instructions
5) Stack related instructions

6) Input/output instructions
7) Machine Control Instructions

Group I - DATA TRANSFER INSTRUCTIONS:

These instructions move data between registers, or between memory and registers. These instructions copy data from source to destination. While copying, the contents of source are not modified.

Ex: i) MOV A, B ii) LDA 4600 iii) LHLD 4200

Group II - ARITHMETIC INSTRUCTIONS:

These instructions perform the operations like: Addition, Subtract, Increment, and 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. For example, ADD B

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. For example, SUB C

Increment and 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. Ex: i) INR D ii) INX H

Group III - 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, and Complement

For example, i) ORA B ii) XRA A iii) RAR

Group IV - BRANCHING INSTRUCTIONS:

The branching instruction changes the normal sequential flow of the Program. These instructions alter either unconditionally or conditionally.

For example, i) JZ 4200 ii) RST 7 iii) CALL 4300

Group V - STACK RELATED INSTRUCTIONS:

Stack Related instructions are used for accessing the stack.

For example- PUSH B, POP C

Group VI - I/O INSTRUCTION

I/O instructions are used for reading or writing the input output port. For example IN 80H, OUT 90H

Group VI - MACHINE CONTROL INSTRUCTIONS:

The control instructions control the operation of microprocessor.

For example i) SIM ii) RIM iii) HLT

4. Explain the Data transfer instructions of 8085 with example.

Copy from source to destination		
MOV	Rd, Rs	This instruction copies the contents of the source register into the destination register, the contents of Rd, M 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
	M, Rs	
	Rd, M	

Move immediate 8-bit		
MVI	Rd, 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 HL registers. **Example:** MVI B, 57H or MVI M, 57H
	M, data	

Load accumulator		
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 2034H

Load accumulator indirect		
LDAX	B/D Reg. 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 B

Load register pair immediate		
LXI	Reg. pair, 16-bit data	The instruction loads 16-bit data in the register pair designated in the operand. **Example:** LXI H, 2034H or LXI H, XYZ

Load H and L registers direct		
LHLD	16-bit address	The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H. The contents of source memory locations are not altered. **Example**: LHLD 2040H

Store accumulator direct		
STA	16-bit address	The contents of the accumulator are copied into the memory location specified by the operand. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example:** STA 4350H

Store accumulator Indirect		
STAX	Reg. pair	The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered. **Example:** STAX B

Store H and L registers direct		
SHLD	16-bit address	The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address. **Example:** SHLD 2470H

Exchange H and L with D and E		
XCHG	none	The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. **Example:** XCHG

Copy H and L registers to the stack pointer		
SPHL	none	The instruction loads the contents of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address. The contents of the H and L registers are not altered. **Example:** SPHL

Exchange H and L with top of stack		
XTHL	none	The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered. **Example:** XTHL

Push register pair onto stack		
PUSH	Reg. pair	The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high- order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location. **Example:** PUSH B or PUSH A

Pop off stack to register pair		
POP	Reg. pair	The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. **Example:** POP H or POP A

Output data from accumulator to a port with 8-bit address		
OUT	8-bit port address	The contents of the accumulator are copied into the I/O port specified by the operand. **Example:** OUT F8H

Input data to accumulator from a port with 8-bit address		
IN	8-bit port address	The contents of the input port designated in the operand are read and loaded into the accumulator. **Example:** IN 8CH

5. Explain the Arithmetic instructions of 8085 with example

ADDR/M –Add register or memory to accumulator:

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

Example: ADD B or ADD M

ADC R/ M- Add register to accumulator with carry:

The contents of the operand (register or memory) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition.
Example: ADC B or ADC M

ADI 8-bit data - Add immediate to accumulator:

The 8-bit data (operand) is added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.
Example: ADI 45H

ACI 8-bit data- Add immediate to accumulator with carry:

The 8-bit data (operand) and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the addition.
Example: ACI 45H

DAD Reg. pair - Add register pair to H and L registers:

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

SUB R / M- Subtract register or memory from accumulator:

The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.
Example: SUB B or SUB M

SBB R / M- Subtract source and borrow from accumulator:

The contents of the operand (register or memory) and the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction.
Example: SBB B or SBB M

SUI 8-bit data- Subtract immediate from accumulator:

The 8-bit data (operand) is subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.
Example: SUI 45H

SBI 8-bit data - Subtract immediate from accumulator with borrow:

The 8-bit data (operand) and the Borrow flag are subtracted from the contents of the accumulator and the result is stored in the accumulator. All flags are modified to reflect the result of the subtraction.
Example: SBI 45H

INR R/ M - Increment register or memory by 1:

The contents of the designated register or memory) are incremented by 1 and the result is stored in the same place. If the operand is a memory location, its location is specified by the contents of the HL registers.
Example: INR B or INR M

INX R - Increment register pair by 1:

The contents of the designated register pair are incremented by 1 and the result is stored in the same place.
Example: INX H

DCR R/ M- Decrement register or memory by 1:

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

DCX R - Decrement register pair by 1:

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

DAA none - Decimal adjust accumulator:

The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion, and the conversion procedure is described below. S, Z, AC, P, CY flags are altered to reflect the results of the operation.

If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag is set, the instruction adds 6 to the high-order four bits.
Example: DAA

6.Explain the Logical instructions of 8085 with example.

ANA R/ M- Logical AND register or memory with accumulator:

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

ANI 8-bit data - Logical AND immediate with accumulator:

The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.
Example: ANI 86H

XRA R/ M - Exclusive OR register or memory with accumulator:

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

XRI 8-bit data - Exclusive OR immediate with accumulator:

The contents of the accumulator are Exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.
Example: XRI 86H

ORA R/ M- Logical OR register or memory with accumulator:

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

ORI 8-bit data - Logical OR immediate with accumulator:

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

Example: ORI 86H

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:
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

CPI 8-bit data - Compare immediate with accumulator:

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

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

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

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 D0. CY is modified according to bit D7. S, Z, P, AC are not affected.
Example: RAL

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 D7. CY is modified according to bit D0. S, Z, P, AC are not affected.
Example: RAR

CMA none - Complement accumulator:

The contents of the accumulator are complemented. No flags are affected. This instruction is use to find 1's compliment of data.
Example: CMA

CMC none - Complement carry:

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

STC none- Set Carry:

The Carry flag is set to 1. No other flags are affected.
Example: STC

7.Explain the Branching instructions of 8085 with example. Jump unconditionally

JMP 16-bit address:

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

Jump conditionally

The program sequence is transferred to the memory location specified by the 16-bit address given in the instruction based on the specified flag of the PSW as described below.
Example: JZ 2034H or JZ XYZ

Opcode	Description	Flag Status
JC	Jump on Carry	CY=1
JNC	Jump on no Carry	CY=0
JZ	Jump on zero	Z=1
JNZ	Jump on no zero	Z=0
JP	Jump on positive	S=0
JM	Jump on minus	S=1
JPE	Jump on parity even	P=1
JPO	Jump on parity odd	P=0

Unconditional subroutine call

CALL 16-bit address:

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 2034H or CALL XYZ

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 as described below. 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 2034H or CZ XYZ

Opcode	Description	Flag Status
CC	Call on Carry	CY=1
CNC	Call on no Carry	CY=0
CZ	Call on zero	Z=1
CNZ	Call on no zero	Z=0
CP	Call on positive	S=0
CM	Call on minus	S=1
CPE	Call on parity even	P=1
CPO	Call on parity odd	P=0

Return from subroutine unconditionally

RET none:

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

Return from subroutine conditionally

The program sequence is transferred from the subroutine to the calling program based on the specified flag of the PSW as described below. 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

Opcode	Description	Flag Status
RC	Return on Carry	CY=1
RNC	Return on no Carry	CY=0
RZ	Return on zero	Z=1
RNZ	Return on no zero	Z=0
RP	Return on positive	S=0
RM	Return on minus	S=1
RPE	Return on parity even	P=1
RPO	Return on parity odd	P=0

Restart

RST 0-7:

The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are:

Instruction	Restart Address
RST0	0000H
RST1	0008H
RST2	0010H
RST3	0018H
RST4	0020H
RST5	0028H
RST6	0030H
RST7	0038H

The 8085 has four additional interrupts and these interrupts generate RST instructions internally and thus do not require any external hardware. These instructions and their Restart addresses are:

Interrupt	Restart Address
TRAP	0024H
RST 5.5	002CH
RST 6.5	0034H
RST 7.5	003CH

8.Explain the Machine Control instructions of 8085 with example.

NOP none - No operation is performed:

The instruction is fetched and decoded. However no operation is executed.
Example: NOP

HLT none -Halt and enter wait state:

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

DI none - Disable interrupts:

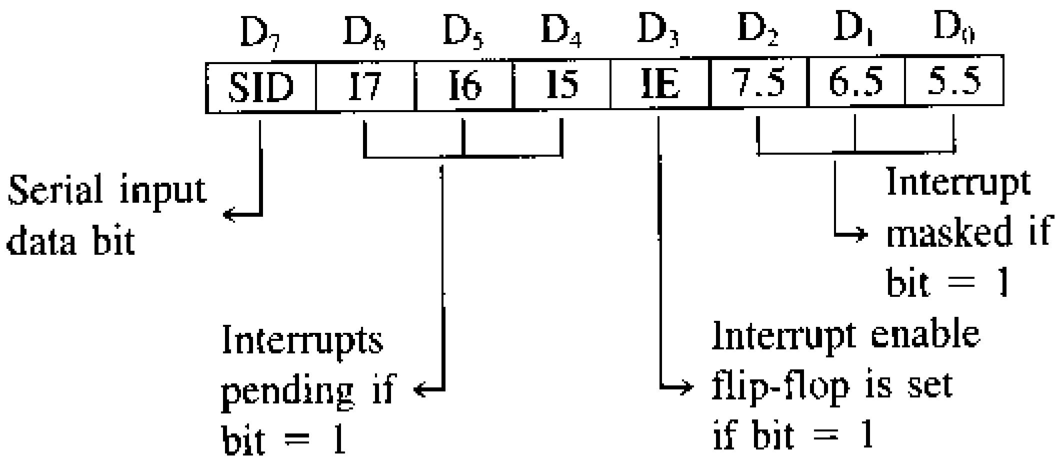
The interrupt enable flip-flop is reset and all the interrupts except the TRAP are disabled. No flags are affected.
Example: DI

EI none - Enable interrupts:

The interrupt enable flip-flop is set and all interrupts are enabled. No flags are affected. After a system reset or the acknowledgement of an interrupt, the interrupt enable flip flop is reset, thus disabling the interrupts. This instruction is necessary to re enable the interrupts (except TRAP).
Example: EI

RIM none Read interrupt mask:

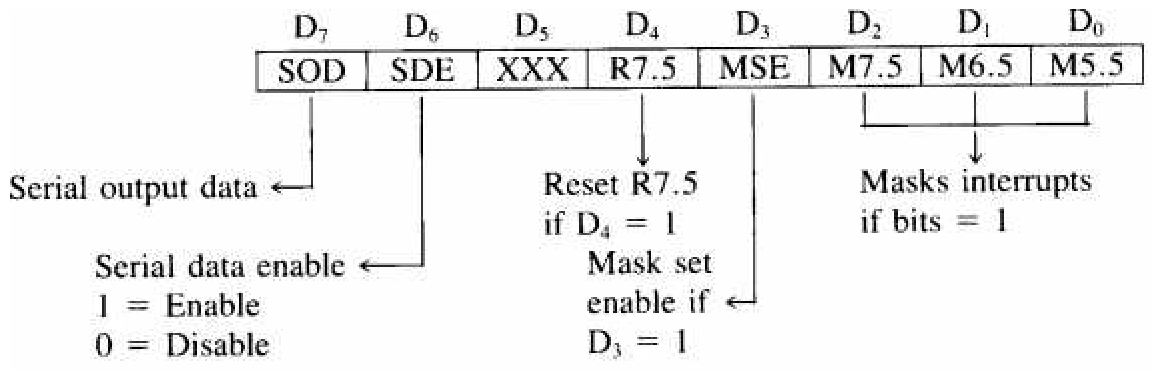
This is a multipurpose instruction used to read the status of interrupts 7.5, 6.5, 5.5 and read serial data input bit. The instruction loads eight bits in the accumulator with the following interpretations.



D7 D6 D5 D4 D3 D2 D1 D0
SID I7 I6 I5 IE 7.5 6.5 5.5
Serial input data bit
Interrupts pending if bit = 1
Interrupt enable flip-flop is set if bit = 1
Interrupt masked if bit = 1

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.



D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
SOD SDE XXX R7.5 MSE M7.5 M6.5 M5.5
Serial output data
Serial data enable 1 = Enable 0 = Disable
Reset R7.5 if $D_4 = 1$
Mask set enable if $D_3 = 1$
Masks interrupts if bits = 1

- SOD — Serial Output Data: Bit D_7 of the accumulator is latched into the SOD output line and made available to a serial peripheral if bit $D_6 = 1$.
- SDE — Serial Data Enable: If this bit = 1, it enables the serial output. To implement serial output, this bit needs to be enabled.
- XXX — Don't Care
- R7.5 — Reset RST 7.5: If this bit = 1, RST 7.5 flip-flop is reset. This is an additional control to reset RST 7.5.
- MSE — Mask Set Enable: If this bit is high, it enables the functions of bits D_2, D_1, D_0. This is a master control over all the interrupt masking bits. If this bit is low, bits D_2, D_1, and D_0 do not have any effect on the masks.
- M7.5 — D_2 = 0, RST 7.5 is enabled.
  = 1, RST 7.5 is masked or disabled.
- M6.5 — D_1 = 0, RST 6.5 is enabled.
  = 1, RST 6.5 is masked or disabled.
- M5.5 — D_0 = 0, RST 5.5 is enabled.
  = 1, RST 5.5 is masked or disabled.

Example: SIM

9.What is stack? Explain stack related instruction with example OR Give function of stack. OR What is stack? Explain the stack operations using examples.

- The stack is a group of memory location in the R/W memory (RAM) that is used for temporary storage of data during the execution of a program.
- Address of the stack is stored into the stack pointer register.
- The 8085 provide two instructions PUSH & POP for storing information on the stack and reading it back.
  a. Data in the register pairs stored on the stack by using the instruction PUSH.
  b. Data is read from the stack by using the instruction POP.
  c. PUSH & POP both instruction works with register pairs only.
  d. The storage and retrieval of the content of registers on the stack fallows the LIFO(Last-In-First-Out) sequence.

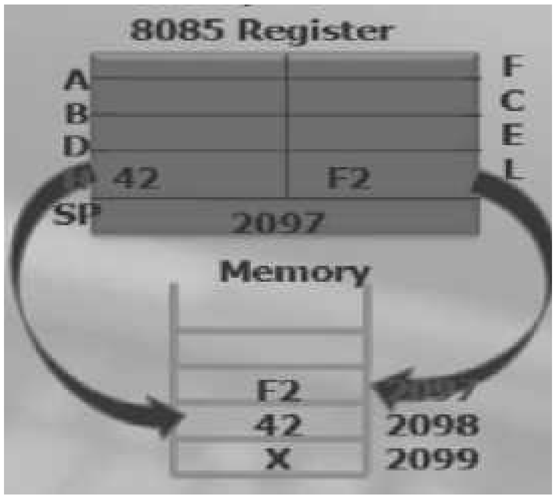
Operation of the stack by PUSH and POP Instruction

```
2000    LXI SP, 2099H        ; this instruction define stack
2003    LXI H, 42F2H         ; this instruction store 42F2 in to the HL pair

2006    PUSH H               ; store HL pair on to the stack
2010    POP H                ; store data from top of the stack to HL pair
```

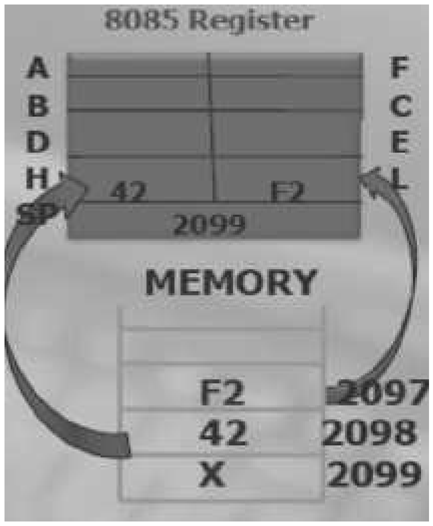
For PUSH H

- The stack pointer is decremented by one to 2098H, and the contents of the h register are copied to memory location 2098H.The stack pointer register is again decremented by one to 2097H,and the contents of the L register are copied to memory location 2097H.The contents of the register pair HL are not destroyed.



For POP H

→ The contents of the top of the stack location shown by the stack pointer are copied in the L register and the stack pointer register is incremented by one to 2098 H. The contents of the top of the stack (now it is 2098H) are copied in the H register, and the stack pointer is incremented by one. The contents of memory location 2097H and 2098 are not destroyed until some other data bytes are stored in these location.



10.Explain Subroutine with CALL and RET Instruction.

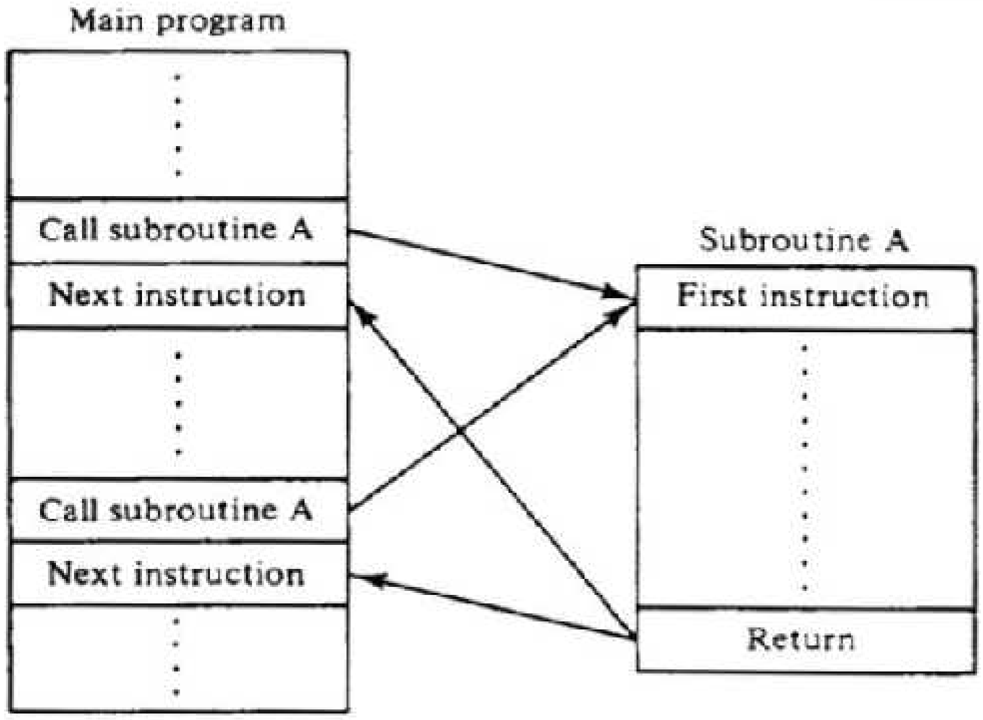
→ A subroutine is a group of instructions that will be used repeatedly in different locations of the program. Rather than repeat the same instructions several times, they can be grouped into a one program which is called subroutine.

- When main program calls a subroutine the program execution is transferred to the subroutine. After the completion of the subroutine, the program execution returns to the main program.
- The microprocessor uses the stack to store the return address of the subroutine.
- The 8085 has two instructions for dealing with subroutines.

– The CALL instruction is used to CALL the subroutine.

– The RET instruction is used to return to the main program at the end of the subroutine.

– Subroutine process is shown in figure below.



The CALL Instruction

CALL 16-bit address

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 ofthe program counter) is pushed onto the stack.

Example: CALL 2034H or CALL XYZ

We can also call the subroutine by using conditional CALL instruction. For Example,CC

16-bit address	Call on if CY = 1
CNC16-bit address	Call on no Carry CY = 0
CP16-bit address	Call on positive S = 0
CM16-bit address	Call on minus S = 1
CZ 16-bit address	Call on zero Z = 1
CNZ16-bit address	Call on no zero Z = 0
CPE16-bit address	Call on parity even P = 1
CPO16-bit address	Call on parity odd P = 0

RET Instruction

RET none
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

We can also return from the subroutine by using conditional RET instruction. For Example,RC 16-

bit address	Return if CY = 1
RNC16-bit address	Return if CY = 0
RP16-bit address	Return if S = 0
RM16-bit address	Return if S = 1 RZ
16-bit address	Return if Z = 1
RNZ16-bit address	Return if Z = 0
RPE16-bit address	Return if P = 1
RPO16-bit address	Return if P = 0

11.Describe the looping and counting techniques. OR Explain looping, counting &indexing with an example.

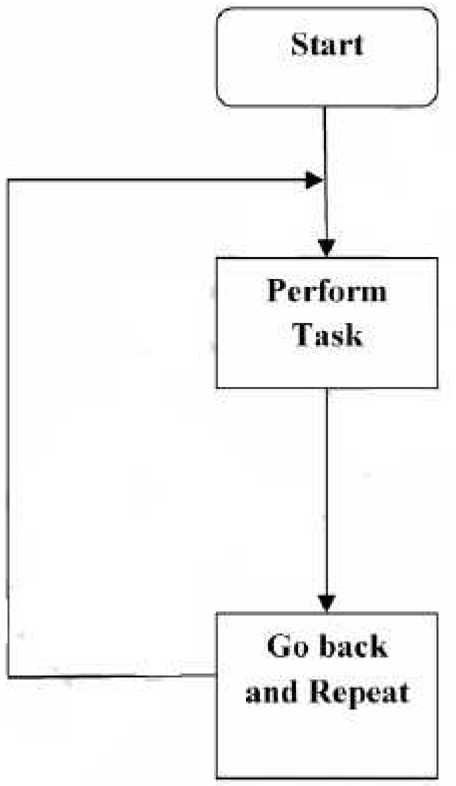
The Programming Technique used to instruct the microprocessor to repeat task is called looping. This process is accomplished by using jump instructions.
A loop can be classified into two groups:
Continuous loop- repeats a task continuously
Conditional loop-repeats a task until certain data condition are met

Continuous loop

A continuous loop is set up by using the unconditional jump Instruction shown in the flowchart. A program with Continuous loop does not stop repeating the tasks until the system is reset.

Conditional Loop

A Conditional loop is setup by the conditional jump instructions. These instructions Check flags (zero, carry, etc.) and repeat the specified task if the conditions are satisfied. These loops usually include counting and indexing. Conditional loop is shown by the Flowchart as follow.



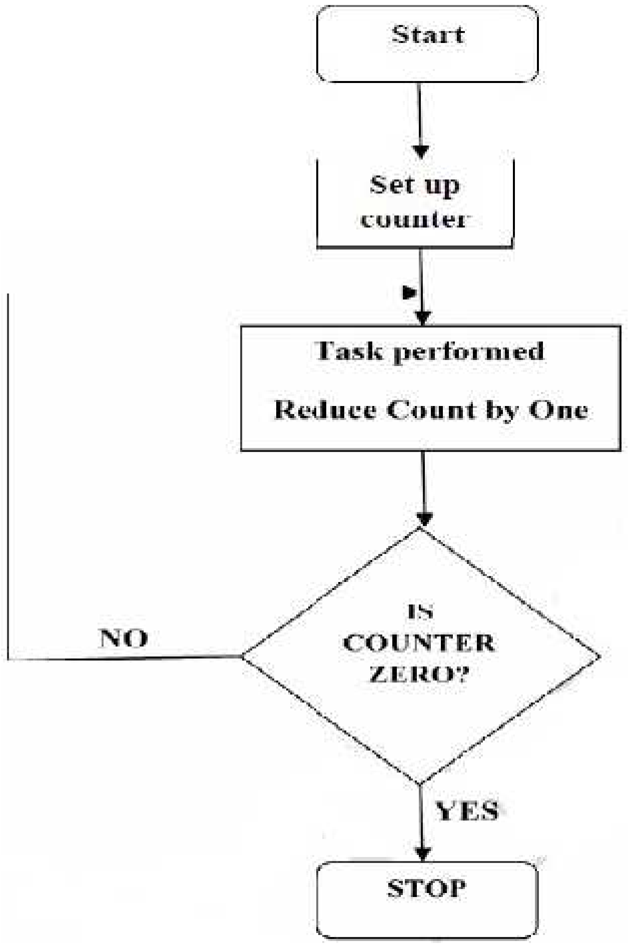


Fig: - Continuous Loop Fig: - Conditional Loop

The Flowchart is translated into the program as follows:

1. Counter is setup by loading an appropriate count in a register.
2. Counting is performed by either incrementing or decrementing the counter.
3. Loop is set up by a conditional jump instruction.
4. End of counting is indicated by a flag

<u>12</u> Write short note on Software and hardware interrupt in 8085 based systemOR List Hardware Interrupts of 8085 with its Address & Priority.

e. **<u>Interrupt</u>**: It means *interrupting* the normal execution of the microprocessor. When microprocessor receives interrupt signal, it discontinues whatever it was executing. It starts executing new program indicated by the interrupt signal.

f. <u>Sequence of Steps Whenever There is an Interrupt</u>

→ It pushes the content of PC (Program Counter) to stack.

→ Then loads the vector address in PC and starts executing the Interrupt Service Routine

(ISR) stored in this vector address.

→ At the end of ISR, a return instruction – IRET will be placed. When the IRET instructionis executed, the processors POP the content of stack to PC.

→ Hence the processor control returns to the main program after servicing the interrupt.

g. Five Hardware Interrupts in 8085

(1) TRAP
(2) RST 7.5
(3) RST 6.5
(4) RST 5.5
(5) INTR

Classification of Interrupts

(1) Maskable and Non-Maskable
(2) Vectored and Non-Vectored
(3) Edge Triggered and Level Triggered
(4) Priority Based Interrupts

❖ Maskable Interrupts

Maskable interrupts are those interrupts which can be *enabled* or *disabled*. Enabling and Disabling is done by software instructions. The interrupts can be masked by moving an appropriate data to accumulator and then executing SIM instruction. (SIM - Set Interrupt Mask).The status of maskable interrupts can be read into accumulator by executing RIM instruction (RIM - Read Interrupt Mask).

List of Maskable Interrupts:

RST 7.5

RST 6.5

RST 5.5INTR

❖ Non-Maskable Interrupts

The interrupts which are always in *enabled* mode are called non maskable interrupts. Theseinterrupts can never be disabled by any software instruction.
TRAP is a non-maskable interrupt.

❖ Vectored Interrupts

The interrupts which have fixed memory location for transfer of control from normalexecution.
List of vectored interrupts:
RST 7.5
RST 6.5
RST 5.5TRAP
The addresses to which program control goes:

Name	Vectored Address
RST 7.5	003C H (7.5 x 0008 H)
RST 6.5	0034 H (6.5 x 0008 H)
RST 5.5	002C H (5.5 x 0008 H)
TRAP	0024 H (4.5 x 0008 H)

❖ Non-Vectored Interrupts

The interrupts which don't have fixed memory location for transfer of control from normal execution is called Non-Vectored Interrupts. The address of the memory location is sent along with the interrupt. INTR is a non-vectored interrupt.

❖ Edge Triggered Interrupts

The interrupts which are triggered at leading or trailing edge are called edge triggeredinterrupts. RST 7.5 is an edge triggered interrupt. It is triggered during the leading (positive) edge.

❖ Level Triggered Interrupts

The interrupts which are triggered at high or low level are called level triggered interrupts.RST 6.5 RST 5.5, INTR are level trigger. TRAP is edge and level triggeredinterrupt

❖ Priority Based Interrupts

Whenever there exists a simultaneous request at two or more pins then the pin with higher priority is selected by the microprocessor. Priority is considered only when there are simultaneous requests.

Priority of interrupts:

Interrupt	Priority
TRAP	1
RST 7.5	2
RST 6.5	3
RST 5.5	4
INTR	5

❖ Software Interrupts

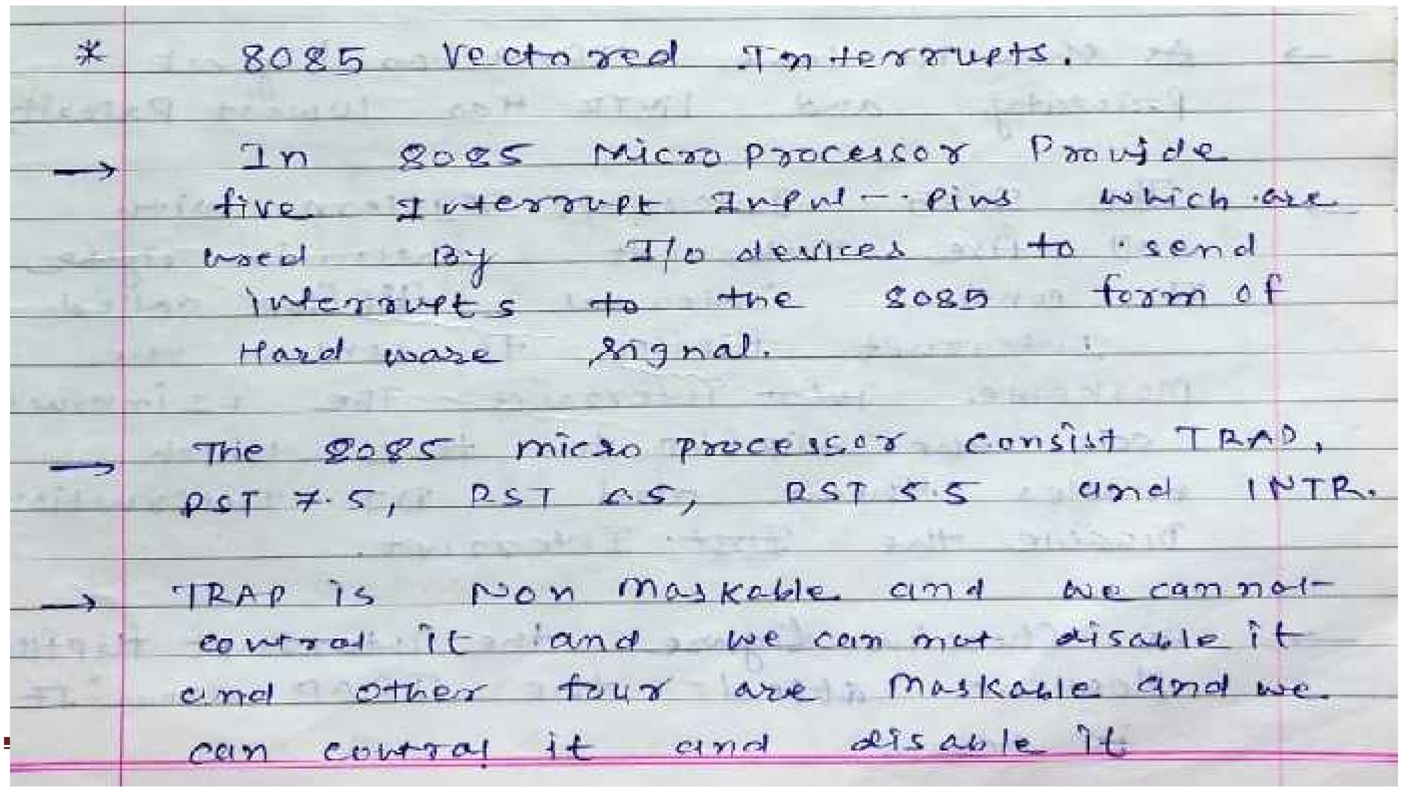
→ The software interrupts are program instructions. These instructions are inserted at desired locations in a program. While running a program, if software interrupt instruction is encountered, then the processor executes an interrupt service routine (ISR).

→ When the instruction is executed, the processor executes an interrupt service routine stored inthe vector address of the software interrupt instruction. The software interrupts of 8085 are RST 0, RST1, RST 2, RST 3, RST 4, RST 5, RST6 and RST 7.

→ All software interrupts of 8085 are vectored interrupts. The software interrupts cannot be masked and they cannot be disabled. The vector addresses of software interrupts are given in table below

Interrupt	Vector Address
RST0	0000H
RST1	0008H
RST2	0010H
RST3	0018H
RST4	0020H
RST5	0028H
RST6	0030H
RST7	0038H

13. Explain 8085 Vectored interrupts: TRAP, RST 7.5, RST 6.5, RST 5.5 and RST.

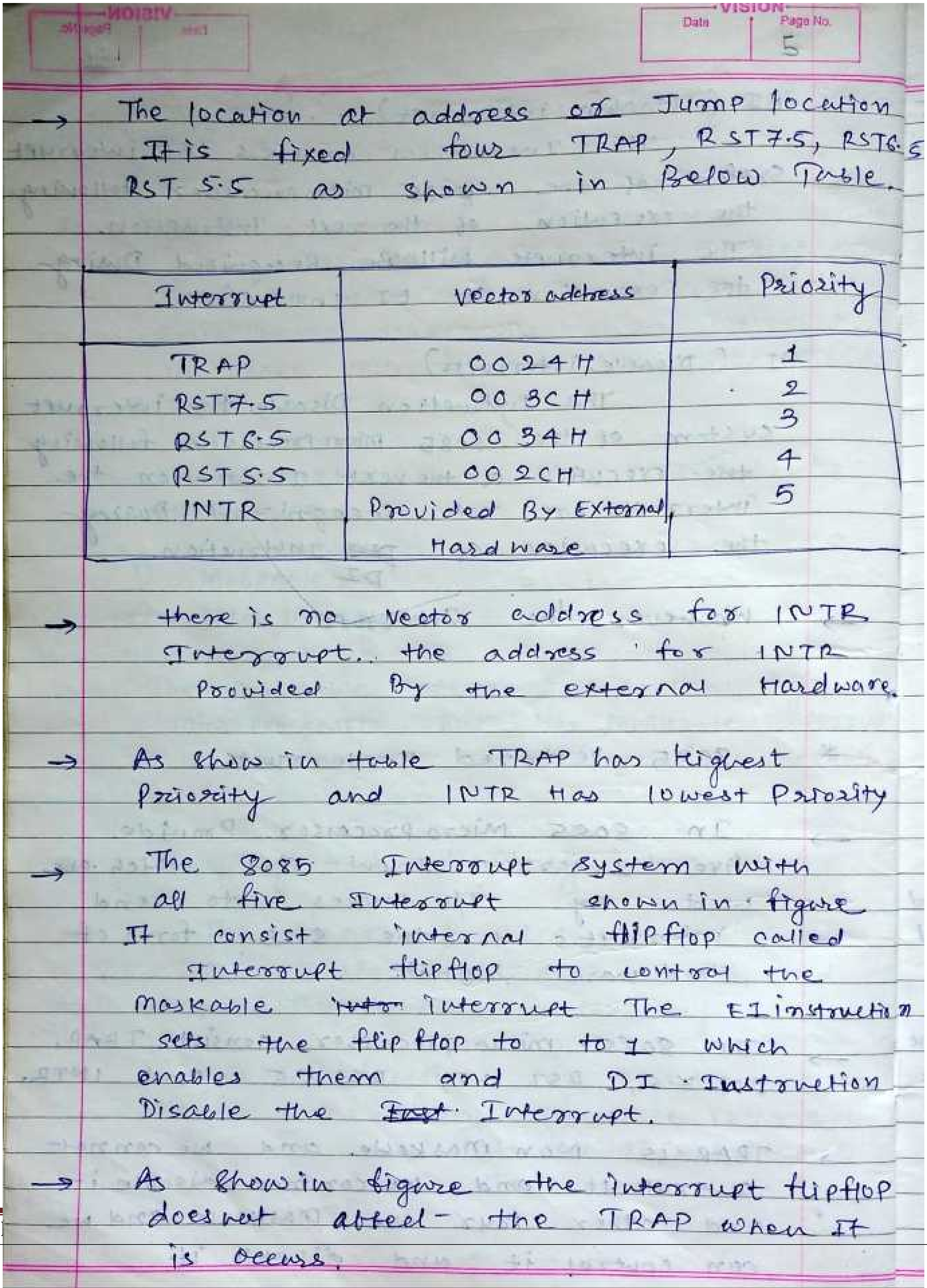


* 8085 Vectored Interrupts.

→ In 8085 Micro processor Provide five Interrupt Input - Pins which are used By I/O devices to send Interrupts to the 8085 form of Hard ware signal.

→ The 8085 micro processor consist TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

→ TRAP is Non maskable and we can not control it and we can not disable it and other four are Maskable and we can control it and disable it



Page No. 5

→ The location at address ~~or~~ Jump location It is fixed four TRAP, RST7.5, RST6.5 RST 5.5 as shown in Below Table.

Interrupt	Vector address	Priority
TRAP	0024H	1
RST7.5	003CH	2
RST6.5	0034H	3
RST5.5	002CH	4
INTR	Provided By External Hardware	5

→ there is no vector address for INTR Interrupt. the address for INTR provided By the external Hardware.

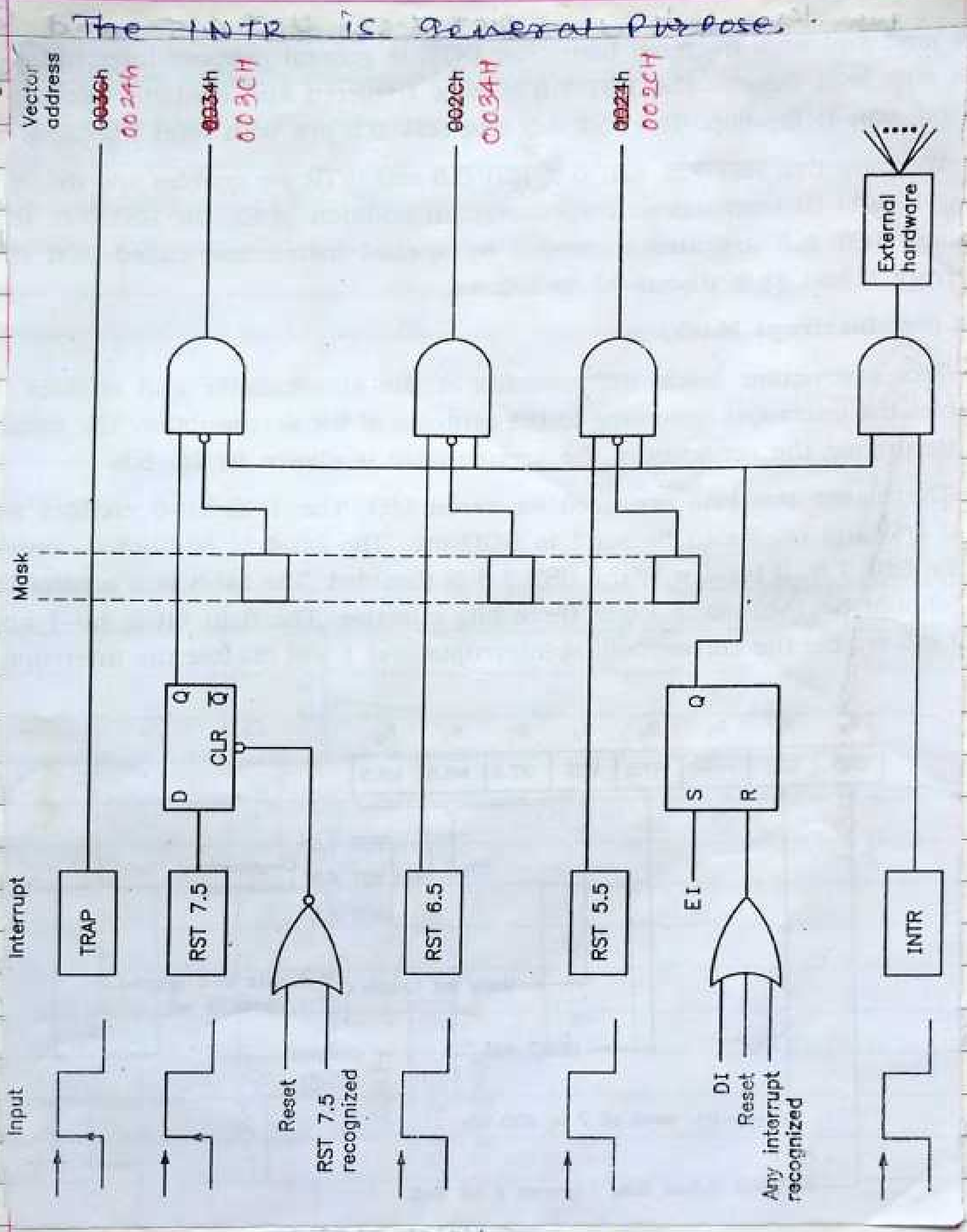
→ As show in table TRAP has highest Priority and INTR Has lowest Priority

→ The 8085 Interrupt system with all five Interrupt shown in figure It consist internal flipflop called Interrupt flipflop to control the Maskable ~~Intr~~ Interrupt The EI instruction sets the flip flop to to 1 which enables them and DI Instruction Disable the ~~Bast~~ Interrupt.

→ As show in figure the interrupt flipflop does not affect the TRAP when It is occurs.

Page No. 6

→ ~~The INTR is general Purpose.~~



→ The INTR is general purpose Interrupt and it is only level trigger.

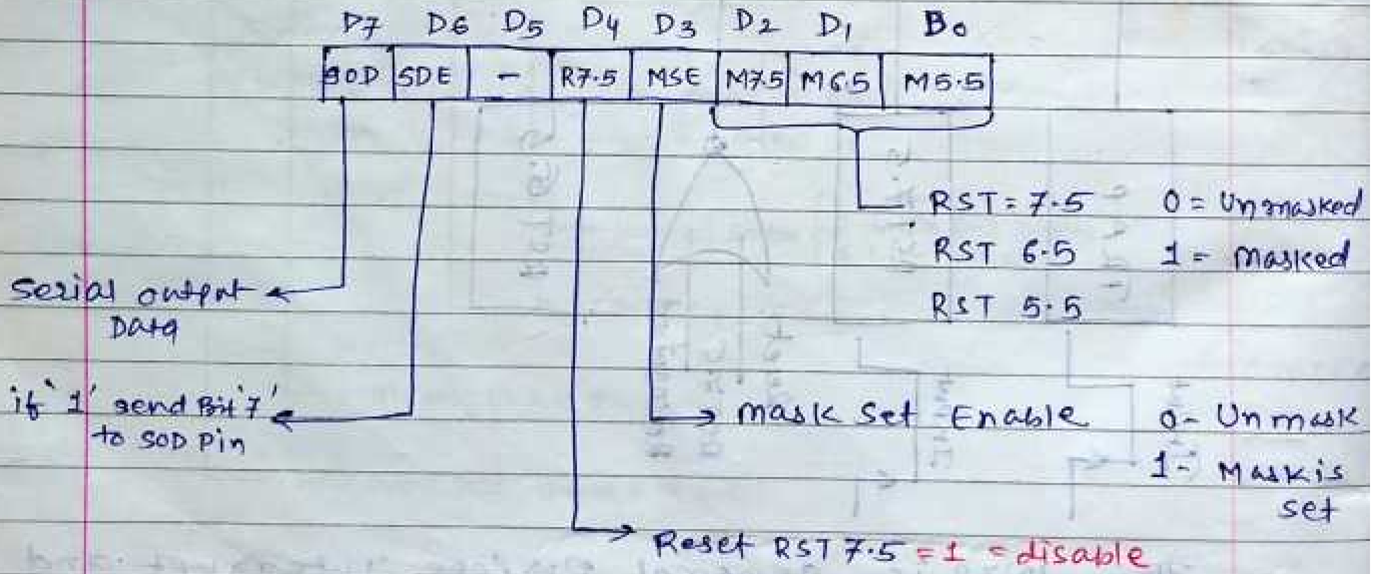
→ TRAP and RST 7.5 are edge trigger and level trigger Both.

→ we know that RST 7·5, RST 6·5 and RST 5·5 and INTR are enable and disable By EI and DI. Instruction.

→ RST 7·5, RST 6·5, RST 5·5 Interrupts are also control By RIM and SIM Instruction.

* SIM (set Interrupt Mask)

This Instruction Reads the contents of the Accumulator and enables or disables the Interrupts according to the contents of the Accumulator The format of Interpreting the content of the Accumulator is shown in fig.



D7 D6 D5 D4 D3 D2 D1 D0
SOD SDE – R7·5 MSE M7·5 M6·5 M5·5
Serial output Data
if '1' send Bit 7' to SOD Pin
Reset RST 7·5 = 1 = disable
mask Set Enable
0 - Unmask
1 - Mask is set
RST 7·5
RST 6·5
RST 5·5
0 = Unmasked
1 = Masked