1. What is interrupt? Explain in detail classification of interrupt?

- Interrupt is one kind of process for data transfer between processor and an I/O device. It is also defined as an event that occurs because of either execution of an instruction or due to a signal from external hardware device.
- In general, whenever an interrupt occurs, the processor performs following steps :
  1. It completes the current instruction and suspends the current program.
  2. Saves the address of next instruction on the stack.
  3. Jumps to a special routine known as Interrupt Service Routine (ISR) which performs the necessary data transfer i.e. provides service to device.
  4. Resumes suspended program by getting its address from stack.

Classification of interrupts :

- We can divide interrupts into following two types based on how it occurs.
  1. Software interrupt
  2. Hardware interrupt
- The software interrupt occurs due to execution of special instruction supported by the processor. For example, RST instruction in 8085 processor.
- The hardware interrupt is caused by sending a signal on one of the interrupt pins of the processor.
- The 8085 supports 5 interrupt pins: TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.
- The hardware interrupts are synchronous means they can occur at any time.

The hardware interrupts are also classified in two types:

1. Maskable
2. Non-Maskable.

- Maskable interrupts are disabled by processor but non-maskable interrupts cannot be disabled. That means once the non-maskable interrupt occurs, it must be served.
- The TRAP in 8085 is non-maskable interrupt and other four hardware RST 7.5, RST 6.5, RST 5.5 and INTR are maskable.
- The 8085 uses EI(Enable Interrupt) and DI(Disable Interrupt) instructions to enable and disable the maskable interrupts.

2. How to Enabling and Disabling Interrupt?

8085 Interrupt

- The 8085 provides EI and DI instructions to enable and disable interrupts respectively.
- These instructions can enable and disable the four maskable interrupts, RST 7.5, RST 6.5, RST 5.5 and INTR only.
- They do not affect the non-maskable interrupt TRAP.
- The EI and DI are implied instructions and do not affect the conditional flags.

EI (Enable Interrupts)

- These instructions enable the interrupt system of the 8085 microprocessor following the execution of the next instruction.
- The interrupt will not be recognized during the execution of the EI instruction.

DI (Disable Interrupt)

- This instruction disables the interrupt system of the 8085 microprocessor following the execution of the next instruction.
- The interrupts are not recognized during the execution of DI instructions.
- Whenever interrupt occurs, the processor saves the current status and jumps to an interrupt service routine. After completing the interrupt service routine, the control returns to the interrupted program.
- This increases the time delay in execution of current program that us being executed by the processor. In certain time sensitive applications like real time systems, this type of delay causes the problem of loosing some information or missing of events.
- To avoid such problems, it is advisable to disable the interrupt system before entering into the critical portion of the application program which performs time sensitive job.
- The interrupt system must be enabled again after completing the execution of the critical portion of the program in order to allow the interrupts during the execution of the rest of the program. It can be shown as :

```
DI : disable interrupts
...
...          critical portion
...          of the program
...
...
EI : enable interrupts
```

3. Write down the steps which perform by microprocessor when Interrupt Occurs.

Steps for hardware interrupts :

Let us understand the steps taken by the 8085 when a valid hardware signal is received on one of the hardware interrupt pin.

1. The 8085 checks for hardware interrupt during each machine cycle at the end of the instruction.
2. If a valid signal is present
   a. If it is TRAP interrupt, processor saves the address of next instruction, jumps to specific location, executes ISR and resumes the original program.
   b. If it is one of the other four (RST 7.5, RST 6.5, RST 5.5, INTR) then
      i. If interrupt is enabled processor saves the address of next instruction, jumps to specific location, executes ISR and resumes the original program.
      ii. Otherwise it is ignored.

- For vector interrupts TRAP, RST 7.5, RST 6.5 and RST 5.5, the jump locations are fixed within the 8085 and hence no interrupt acknowledgment is needed. The 8085 provides $\overline{INTA}$ output pin to send the interrupt, acknowledgement.
- In case of the interrupt on INTR pin, the vector address (jump address) is to be received from external hardware. For that, the 8085 sends $\overline{INTA}$ signal to the external hardware to receive the vector address.

4. Explain RST Instruction in 8085.

INTERRUPT USING RST INSTRUCTIONS :

- The 8085 provides eight RST (restart) instructions. The restart instruction takes n (0 to 7) as operand and when it executes it saves the address of next instruction on stack and jumps to location that is multiplied by n. The binary code of the RST n instruction is given below :

D7	D6	D5	D4	D3	D2	D1	D0
1	1	n	n	n	1	1	1

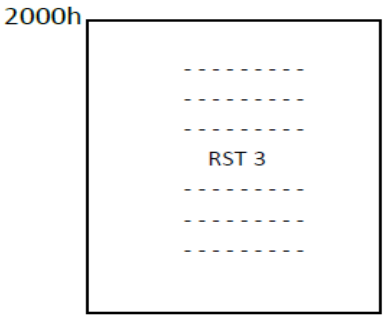
- The bits D_5, D_4 and D_3 are replaced by the 3-bit binary value of the n (0 to 7). The value of n is multiplied by the 8 to get the memory location where control is transferred. Table below lists the opcode and jump locations for all the eight restart instructions.

Instruction	Binary Code	Opcode (Hex)	Jump location
RST 0	1 1 0 0 0 1 1 1	C7h	0000h
RST 1	1 1 0 0 1 1 1 1	CFh	0008h
RST 2	1 1 0 1 0 1 1 1	D7h	0010h
RST 3	1 1 0 1 1 1 1 1	DFh	0018h
RST 4	1 1 1 0 0 1 1 1	E7h	0020h
RST 5	1 1 1 0 1 1 1 1	EFh	0028h
RST 6	1 1 1 1 0 1 1 1	F7h	0030h
RST 7	1 1 1 1 1 1 1 1	FFh	0038h

Software interrupt using RST :

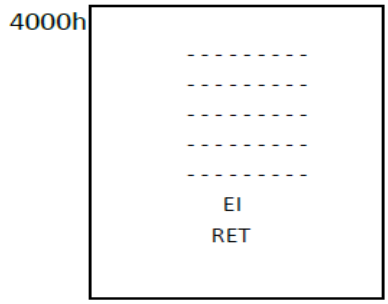
- Observe that the distance between two jump locations for subsequent RST instructions is only eight bytes. The eight bytes are not sufficient for writing an ISR for any purpose.
- To solve this problem, generally the ISR is written at some other location in memory and the jump instruction to that location (where actual service routine is written) is provided at the RST jump location. Figure below illustrates the concept :
- As shown in the figure, the mainline program is written starting from location 2000h and contains RST 3 instruction.
- The jump location for RST 3 instruction 0018h which stores the jump instruction JMP 4000h to transfer control to actual service routine.
- During the execution of mainline program, when RST 3 instruction executes, it transfers control to location 0018h. The execution JMP 4000h instruction at that location further transfers the control at the location 4000h.
- The service routine at location 4000h starts executing and provides the service. At the end of service routine, execution of RET instruction transfers control to mainline program at the instruction after RST.
- Remember that RST stores the address of next instruction on stack and then jumps to its jump location. This is how the software interrupts using restart instructions works in the 8085.

; mainline program



0018h jmp 4000h; RST 3 location

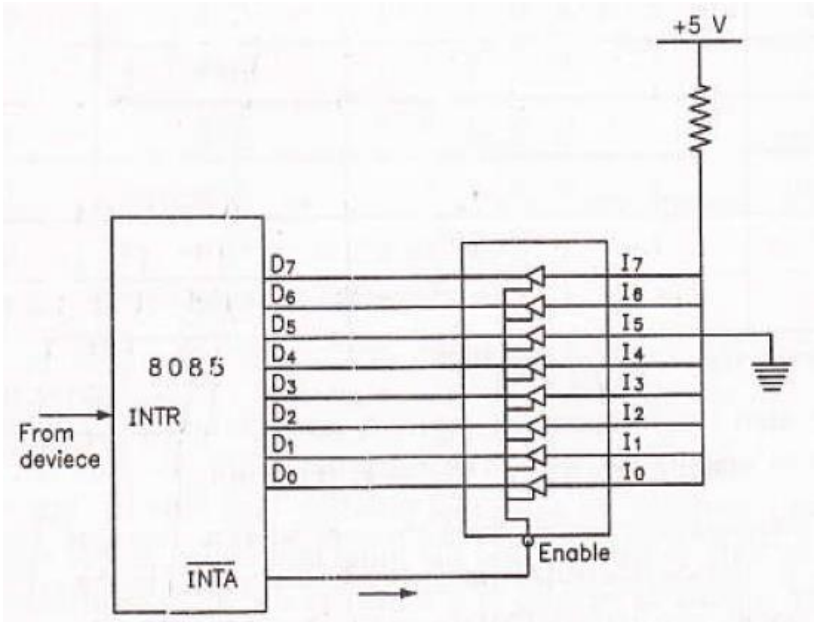
;ISR



(INTERRUPT SERVICE ROOTINE)

Hardware interrupt using RST :

- The RST instruction is single byte instruction and when executes it saves the address of next instruction on to the stack and jumps to its jump location.
- Figure below shows how the hardware interrupt is implemented using external hardware and the RST instruction :



- Assume that the RST 3 instruction in mainline program. Assume that during the execution of mainline program the device sends a high signal on the INTR pin. We know that the 8085 checks the hardware interrupts at the end of each instruction.
- The 8085 finds the valid signal on INTR pin. The next cycle it generates is the interrupt acknowledgement cycle which consists of three machine cycles as follows.
  - During M1, the $\overline{INTA}$ signal goes low which enables the buffer so that code for the RST 3 is placed on the data bus which is decoded as RST 3 instruction.
  - During M2, the lower byte of program counter is saved on to the stack.
  - During M3, the higher byte of program counter is saved on to the stack.
- Whenever a hardware interrupt occurs, the processor stops current program, disables the interrupts and then jumps to the ISR. This is done to ensure that the execution of ISR is not interrupted by the other interrupts.
- The 8085 automatically does not enable the interrupts again. Hence, it is responsibility of the user to put the EI instruction to enable the hardware interrupts so that once interrupted program is resumed, the system can accept the valid interrupts.

5. Explain in detail 8085 vector interrupt

8085 Vectored Interrupts :

- The 8085 provides five interrupt input pins which are used by I/O devices to send interrupts to the 8085 in form of hardware signal. They include TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

- The TRAP is non-maskable and cannot be disabled. The other four are maskable and can be disabled. The location at which the control is transferred when any interrupt occurs is called its vector addresses or jump location. It is fixed four TRAP, RST 7.5, RST 6.5 and RST 5.5 as shown in the table below :

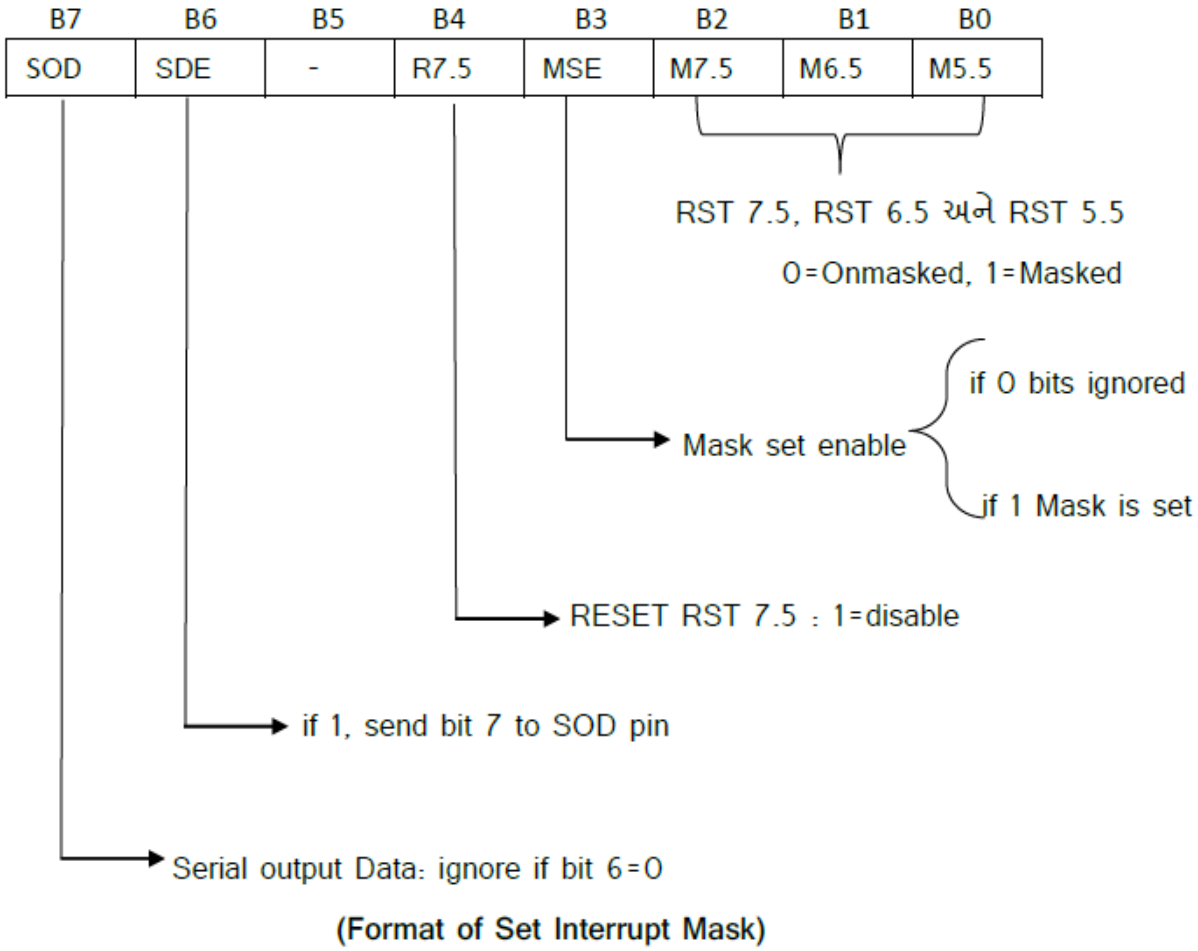
Interrupt	Vector address	Priority
TRAP	0024h	1
RST 7.5	003Ch	2
RST 6.5	0034h	3
RST 5.5	002Ch	4
INTR	Provided by external hardware	5

- As their vector addresses are fixed, they don't use $\overline{INTA}$. The vector addresses for the INTR is provided by the external hardware using RST or CALL instruction in response to the $\overline{INTA}$ signal. Table also shows their priority with TRAP has highest priority followed by the RST 7.5, RST 6.5, RST 5.5 and INTR with lowest priority.

6. Explain in detail RIM & SIM

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 contents of the accumulator is shown in the figure below :



(Format of Set Interrupt Mask)

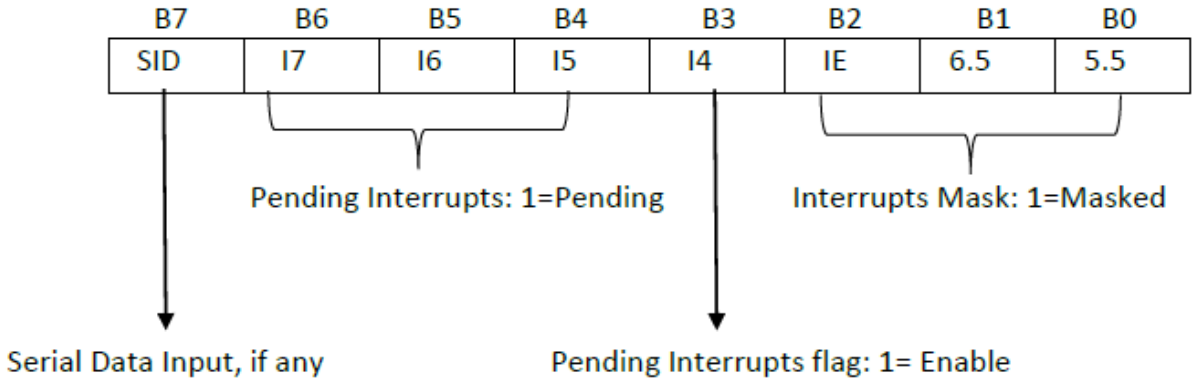
- The higher two bits are used for serial I/O. The 1 in bit-6 enables the serial I/O and bit-7 can be send to SOD pin. The bit-4 is additional control for the RST 7.5. If bit-4=1, the RST 7.5 is disabled.
- The bit-3 is a control bit and should be 1 to make lower three bits effective. The 0 in bit-2, bit-1 and bit-0 will enable the corresponding interrupts and1 will disable the interrupts.
- The sequence of instructions to enable all the interrupt of the 8085is written as follows :
  - EI : enable interrupts
  - MVI A, 08h : load accumulator with suitable bit pattern
  - SIM : enable RST 7.5, RST 6.5 and RST 5.5

- In above example, bit-4=1 in the accumulator makes the lower 3-bits effective. The 0 in lower 3-bits of the accumulator enables the interrupts RST 7.5, RST 6.5 and RST 5.5.

- As 8085 receives the other interrupts on the lines which are pending for execution. The RIM (Read Interrupt Mask) instruction gives the details of pending interrupts.

RIM (Read Interrupt Mask)

- This instruction loads the accumulator with the current status of the interrupt masks, the interrupt enable, pending interrupts and serial input data. It is shown in the figure below :



(Format at read interrupt mask)

- The lower 3-bits show the current status of the RST 7.5, RST 6.5 and RST 5.5, whether they are enabled or disabled. The next bits shows the status of interrupt system of the processor as a group, whether it is enabled or disabled.
- The EI instruction sets this flag to 1 and the DI instruction rests it to 0. The next 3-bits shows the pending status of the RST 7.5, RST 6.5 and RST 5.5, whether some or all of them are pending or not. The last bit is a bit received from the SID (Serial Data Input) pin of the processor, if it is available during the execution of the RIM instruction.
- Assume that the RIM instruction loads the accumulator with the data byte 2Ch. What is the meaning of it?
- The accumulator is loaded with the binary number 0010 1100. If it is interrupted, the following meaning evolves.

  (1) RST 7.5 enabled, RST 6.5 and RST 5.5 disabled.

  (2) Interrupt system enabled.

  (3) RST 6.5 pending.