Interrupts in 8085 microprocessor

Introduction:

In the 8085 microprocessor, an interrupt is a signal that temporarily suspends the normal execution of a program and redirects the control to a specific interrupt service routine (ISR). Interrupts allow the microprocessor to respond to external events, such as user input, system events, or hardware signals, without the need for constant polling.

There are five interrupt signals in the 8085 microprocessor:

1. **TRAP:** The TRAP interrupt is a non-maskable interrupt that is generated by an external device, such as a power failure or a hardware malfunction. The TRAP interrupt has the highest priority and cannot be disabled.
2. **RST 7.5:** The RST 7.5 interrupt is a maskable interrupt that is generated by a software instruction. It has the second highest priority.
3. **RST 6.5:** The RST 6.5 interrupt is a maskable interrupt that is generated by a software instruction. It has the third highest priority.
4. **RST 5.5:** The RST 5.5 interrupt is a maskable interrupt that is generated by a software instruction. It has the fourth highest priority.
5. **INTR:** The INTR interrupt is a maskable interrupt that is generated by an external device, such as a keyboard or a mouse. It has the lowest priority and can be disabled.

When microprocessor receives any interrupt signal from peripheral(s) which are requesting its services, it stops its current execution and program control is transferred to a sub-routine by generating **CALL** signal and after executing sub-routine by generating **RET** signal again program control is transferred to main program from where it had stopped. When microprocessor receives interrupt signals, it sends an acknowledgement (INTA) to the peripheral which is requesting for its service. Interrupts can be classified into various categories based on different parameters:

1. **Hardware and Software Interrupts –** When microprocessors receive interrupt signals through pins (hardware) of microprocessor, they are known as *Hardware Interrupts*. There are 5 Hardware Interrupts in 8085 microprocessor. They are – *INTR, RST 7.5, RST 6.5, RST 5.5, TRAP Software Interrupts* are those which are inserted in between the program which means these are mnemonics of microprocessor. There are 8 software interrupts in 8085 microprocessor. They are – *RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7*.
2. **Vectored and Non-Vectored Interrupts –** *Vectored Interrupts* are those which have fixed vector address (starting address of sub-routine) and after executing these, program control is transferred to that address. Vector Addresses are calculated by the formula 8 * TYPE

INTERRUPT	VECTOR ADDRESS
TRAP (RST 4.5)	24 H
RST 5.5	2C H
RST 6.5	34 H
RST 7.5	3C H

1. **For Software interrupts vector addresses are given by:**

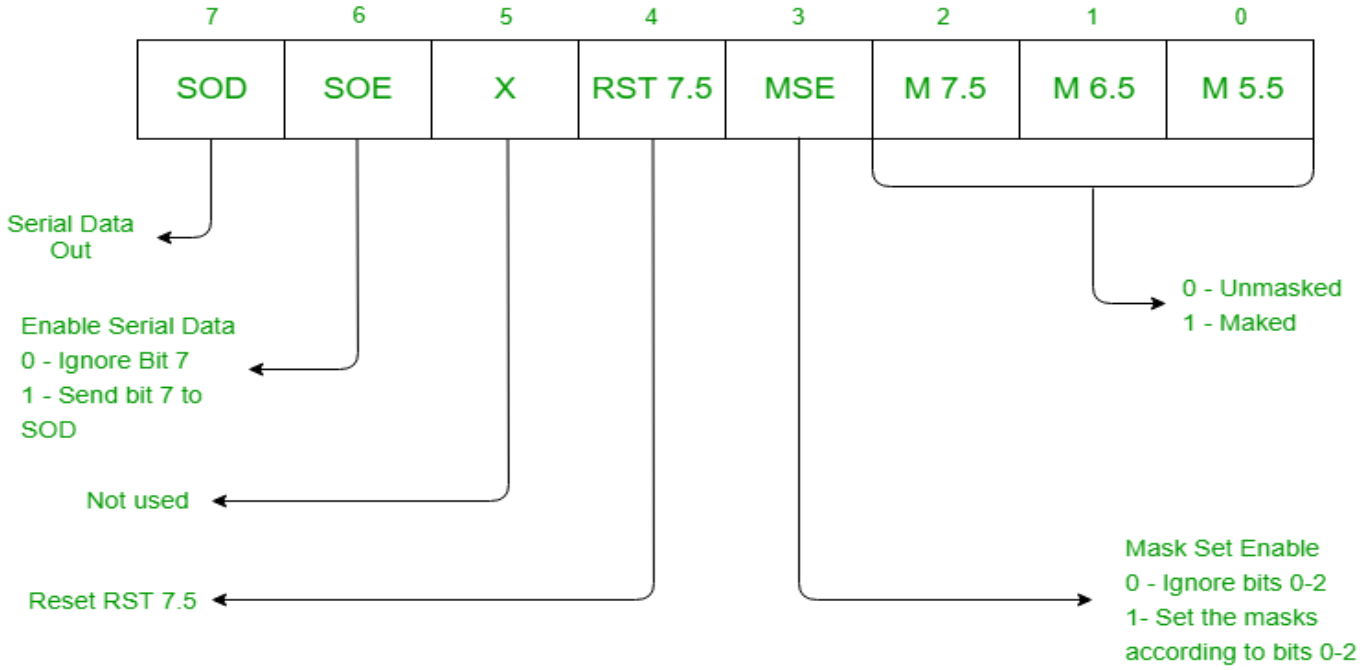
INTERRUPT	VECTOR ADDRESS
RST 0	00 H
RST 1	08 H
RST 2	10 H
RST 3	18 H
RST 4	20 H
RST 5	28 H
RST 6	30 H
RST 7	38 H

1. *Non-Vectored Interrupts* are those in which vector address is not predefined. The interrupting device gives the address of sub-routine for these interrupts. *INTR* is the only non-vectored interrupt in 8085 microprocessor.
2. **Maskable and Non-Maskable Interrupts –** *Maskable Interrupts* are those which can be disabled or ignored by the microprocessor. These interrupts are either edge-triggered or level-triggered, so they can be disabled. *INTR, RST 7.5, RST 6.5, RST 5.5* are maskable interrupts in 8085 microprocessor. Non-Maskable Interrupts are those which cannot be disabled or ignored by microprocessor. *TRAP* is a non-maskable interrupt. It consists of both level as well as edge triggering and is used in critical power failure conditions.

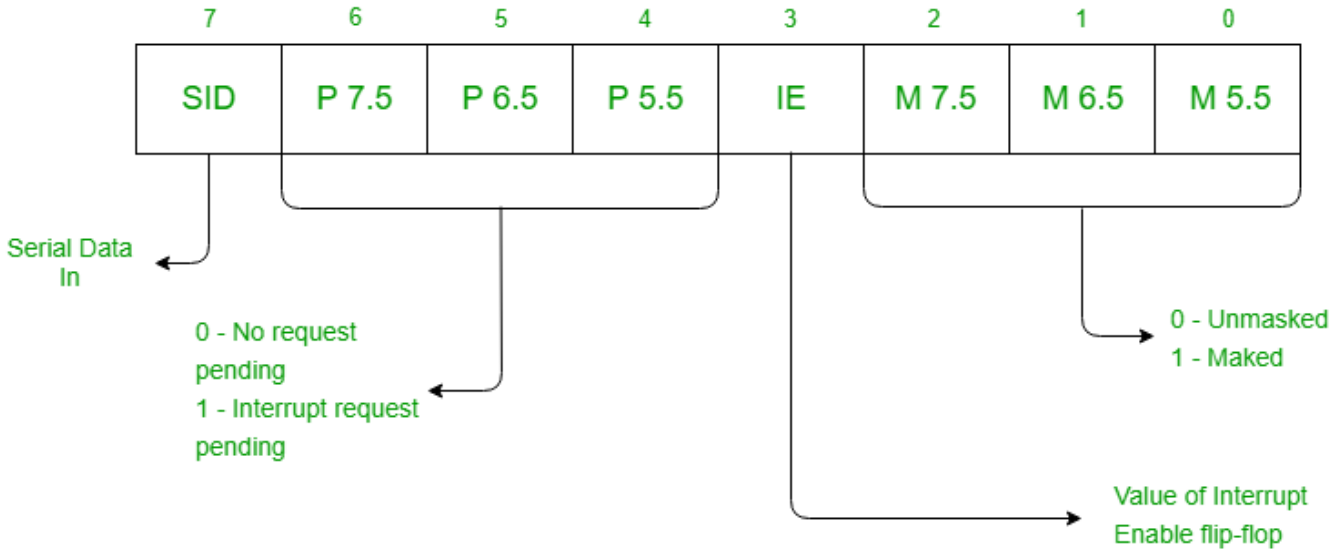
Priority of Interrupts – When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.

Instruction for Interrupts –

1. **Enable Interrupt (EI) –** The interrupt enable flip-flop is set and all interrupts are enabled following the execution of next instruction followed by EI. No flags are affected. After a system reset, the interrupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to enable the interrupts again (except TRAP).
2. **Disable Interrupt (DI) –** This instruction is used to reset the value of enable flip-flop hence disabling all the interrupts. No flags are affected by this instruction.
3. **Set Interrupt Mask (SIM) –** It is used to implement the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by setting various bits to form masks or generate output data via the Serial Output Data (SOD) line. First the required value is loaded in accumulator then SIM will take the bit pattern from it.



4. **Read Interrupt Mask (RIM) –** This instruction is used to read the status of the hardware interrupts (RST 7.5, RST 6.5, RST 5.5) by loading into the A register a byte which defines the condition of the mask bits for the interrupts. It also reads the condition of SID (Serial Input Data) bit on the microprocessor.



Uses of Interrupts in 8085 microprocessor:

Interrupts in the 8085 microprocessor are used for various purposes, including:

Real-time processing: Interrupts allow the microprocessor to respond quickly to external events in real-time, such as user input or hardware signals. This is particularly useful in applications such as control systems and data acquisition systems where time-critical operations are required.

Multi-tasking: Interrupts enable the microprocessor to perform multiple tasks simultaneously by temporarily suspending the current task and executing the ISR for the interrupting event. This allows the microprocessor to switch between different tasks and maximize the utilization of system resources.

Input/output operations: Interrupts can be used for handling input/output operations, such as data transfer between the microprocessor and external devices. This allows the microprocessor to perform other tasks while waiting for input/output operations to complete.

Error handling: Interrupts can be used for error handling and exception handling, such as detecting and recovering from hardware or software errors.

Power management: Interrupts can be used for power management, such as putting the microprocessor into a low-power mode when it is not needed and waking it up when an interrupt occurs.

Issues of Interrupts in 8085 microprocessor:

There are several issues that need to be considered when using interrupts in the 8085 microprocessor:

1. **Priority conflicts:** The 8085 microprocessor supports multiple interrupt signals with different priorities. If multiple interrupts occur simultaneously, it can lead to priority conflicts and result in incorrect operation or system failure. Therefore, the priority levels of each interrupt signal need to be carefully designed and tested to avoid conflicts.
2. **Race conditions:** Race conditions can occur when multiple processes try to access the same resources, such as registers or memory locations, simultaneously. This can lead to incorrect results or system failure. Therefore, interrupt handlers need to be carefully designed to avoid race conditions, such as by disabling interrupts during critical operations.
3. **Interrupt latency:** Interrupt latency is the time delay between when an interrupt occurs and when the corresponding ISR starts executing. Interrupt latency can affect the system's responsiveness and real-time performance. Therefore, interrupt handlers need to be designed to minimize interrupt latency, such as by using fast interrupt service routines and optimizing the interrupt handling process.
4. **Interrupt nesting:** Interrupt nesting occurs when an interrupt occurs while the microprocessor is executing an ISR for another interrupt. Interrupt nesting can lead to complex interrupt handling and priority conflicts. Therefore, interrupt handlers need to be carefully designed to avoid interrupt nesting, such as by disabling lower-priority interrupts during critical operations.
5. **Interrupt overhead:** Interrupt overhead is the additional processing time and resources required to handle interrupts. Interrupt overhead can affect the system's performance and efficiency, especially if the system experiences a high volume of interrupts. Therefore, interrupt handlers need to be designed to minimize interrupt overhead, such as by optimizing the interrupt handling process and reducing unnecessary operations.