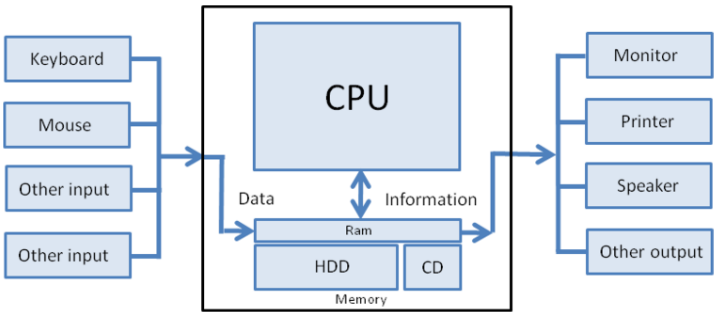
UNIT V

Basic I/O, Memory R/W & Interrupt Operations

#Explain I/O Operations method Memory Mapped I/O and I/O Mapped I/O.

CPU uses two methods to perform input/output operations between the CPU and peripheral devices in the computer. These two methods are called **memory mapped IO** and **IO mapped IO**.

- **Memory-mapped IO** uses the ***same address space*** to address both memory and I/O devices.
- On the other hand, **IO mapped IO** uses ***separate address spaces*** to address memory and IO devices.



Memory Mapped I/O

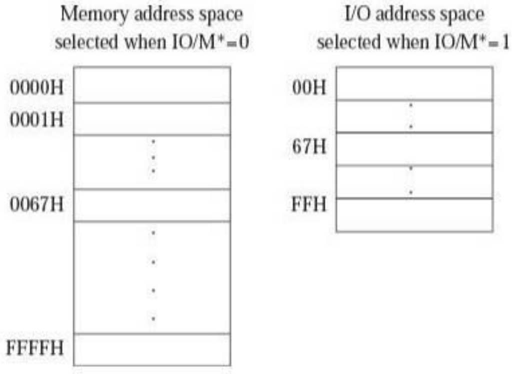
Memory mapped IO uses one address space for memory and input and output devices. In other words, some addresses are assigned to memory while others are assigned to store the addresses of IO devices.

There is one set of read and write instruction lines. The same set of instructions work for both memory and IO operations. Therefore, the instructions used to manipulate memory can be used for IO devices too. Hence, it can lessen the addressing capability of memory because some are occupied by the IO.

I/O Mapped I/O

IO mapped IO uses two separate address spaces for memory locations and for IO devices. There are two separate control lines for both memory and IO transfer. In other words, there are different read-write instruction for both IO and memory.

IO read and IO write are for IO transfer whereas memory read and memory write are for memory transfer. IO mapped IO is also called port-mapped IO or isolated IO.



The **main difference** between memory mapped IO and IO mapped IO is that the **memory mapped IO uses the same address space for both memory and IO device while the IO mapped IO uses two separate address spaces for memory and IO device.**

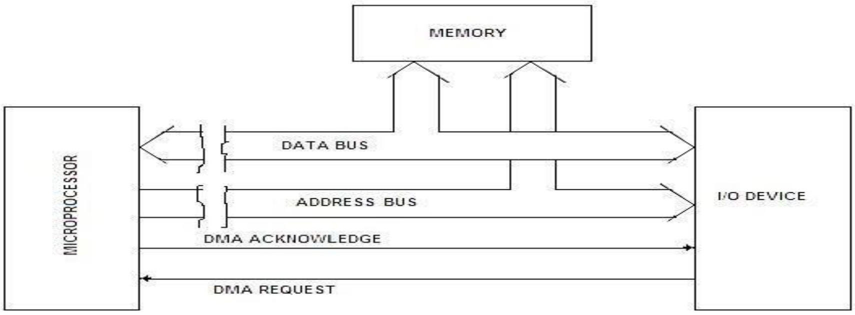
#Differences between Memory-mapped I/O and I/O mapped I/O

S.N.	Memory-mapped I/O	I/O mapped I/O
1.	The I/O devices and memory, both are treated as memory.	The I/O devices are treated as I/O devices and the memory is treated as memory.
2.	The I/O devices are provided with 16-bit address values (in 8085)	The I/O devices are provided with 8-bit address values. (In 8085)
3.	The interfaced devices are accessed by the memory read or memory write cycles.	The interfaced devices are accessed by the I/O read or I/O write cycles.
4.	The peripherals or the I/O ports are treated as memory locations. Thus, all the instructions related to the memory can be utilized for the data exchange between the processor and the I/O device.	Only the IN and the OUT instructions can be put to use for transferring information between the I/O device and the processor.
5.	In the memory-mapped ports, the information data can be moved to the I/O devices from any register or vice versa.	In the I/O mapped ports, the information bytes can be moved around between the ports and the accumulator register only.
6.	The full memory address space cannot be used solely for addressing memory for interfacing.	The full memory address space can be used solely for addressing memory for interfacing.
7.	Data transfer is possible between any register and I/O device.	Data transfer is possible between the accumulator and I/O device only.
8.	Here, a large number of I/O ports (2^{16} ports) are possible to be used for interfacing.	Only 256 I/O ports i.e., 2^{8} ports, are made available for interfacing.
9.	While executing the memory, write or read cycles, the IO/M` is set to low (I$O/M$` = 0).	While executing the I/O write or read cycles, the IO/M` is set to high (I$O/M$` = 1).
10.	There is more decoder hardware involved.	There is less decoder hardware involved.
11.	Separate control signals are not required since we have a unified memory space.	Special control signals are used here since we have separate memory spaces.
12.	We can perform arithmetic and logical operations on the data.	We cannot perform arithmetic and logical operations on the data.

#Discuss DMA with the help its advantages and application

Direct Memory Access (DMA)

- **Direct memory access (DMA)** is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.
- The process is managed by a chip known as a **DMA controller (DMAC).**
- It moves the data between memory and peripheral without bothering the CPU. This means it frees up both CPU power and bandwidth between CPU and main memory.
- For a CPU based copy, the data has to go into and out of the CPU, which is often a slow process and blocks both the core doing the copy and the other cores. And if the data flow has gaps, as it will in the case of HDD and network, the CPU can sit there wasting time until more data becomes available.
- If the data rate is very low, DMA is overkill. But if the data is either fast or large, DMA offloads the CPU and lets it get on with useful work.
- The problem of slow data transfer between input-output port and memory or between two memories is avoided by implementing Direct Memory Access (DMA) technique.
- This is faster as the microprocessor/computer is bypassed and the control of address bus and data bus is given to the DMA controller.



Advantages & Disadvantages of DMA

Advantages

- Transferring the data without the involvement of the processor will **speed up** the read-write task.
- DMA **reduces the clock cycle** requires to read or write a block of data.
- DMA allows faster processing since the processor can be working on something else while the peripheral can be populating memory.
- DMA enables more efficient use of interrupts.
- High transfer rates.
- DMA capable device can communicate directly with memory.
- Implementing DMA also **reduces the overhead** of the processor.

Disadvantages

- Cost of DMA hardware.
- Data has to be stored in continuous memory locations.
- DMA controller is slow in comparison to CPU.

Application of DMA

- DMA has been a built-in feature of PC architecture since the introduction of the original IBM PC.
- PC-based DMA was used for floppy disk I/O in the original PC and for hard disk I/O in later versions.
- PC-based DMA technology, along with high speed bus technology, is driven by data storage, communications, and graphics needs-all of which require the highest rates of data transfer between system memory and I/O devices.
- *Applications areas are: **cinemas, theatres, hotels, railway stations, shopping centres, trade shows, museums & many more.***

Explain the different DMA transfer Modes

DMA Transfer Modes

There are 3 different modes of DMA data transfer. They vary by how DMA controller determines when to transfer data (but actual data transfer process remains the same in all three cases).

(a) Burst Mode

- Entire block of data is transferred in one continuous sequence.
- Once the DMA controller is granted access to the system bus by CPU, it transfer all bytes of data in the data block before relinquishing control of system buses back to the CPU.
- This mode is useful for loading programs or data files into memory, but it keeps CPU idle for relatively long period of time.

(b) Cycle Stealing Mode

- DMA controller obtains access to system bus as in burst mode; transfers one byte of data and returns the control of the system bus to CPU. It continually issues requests using Bus Request (BR) signals, transferring one byte of data per request, until it has transferred its entire block of data. (steals one CPU cycle).
- The data block is not transferred as quickly as in burst mode, but the CPU is not idled for long period of time as in burst mode.

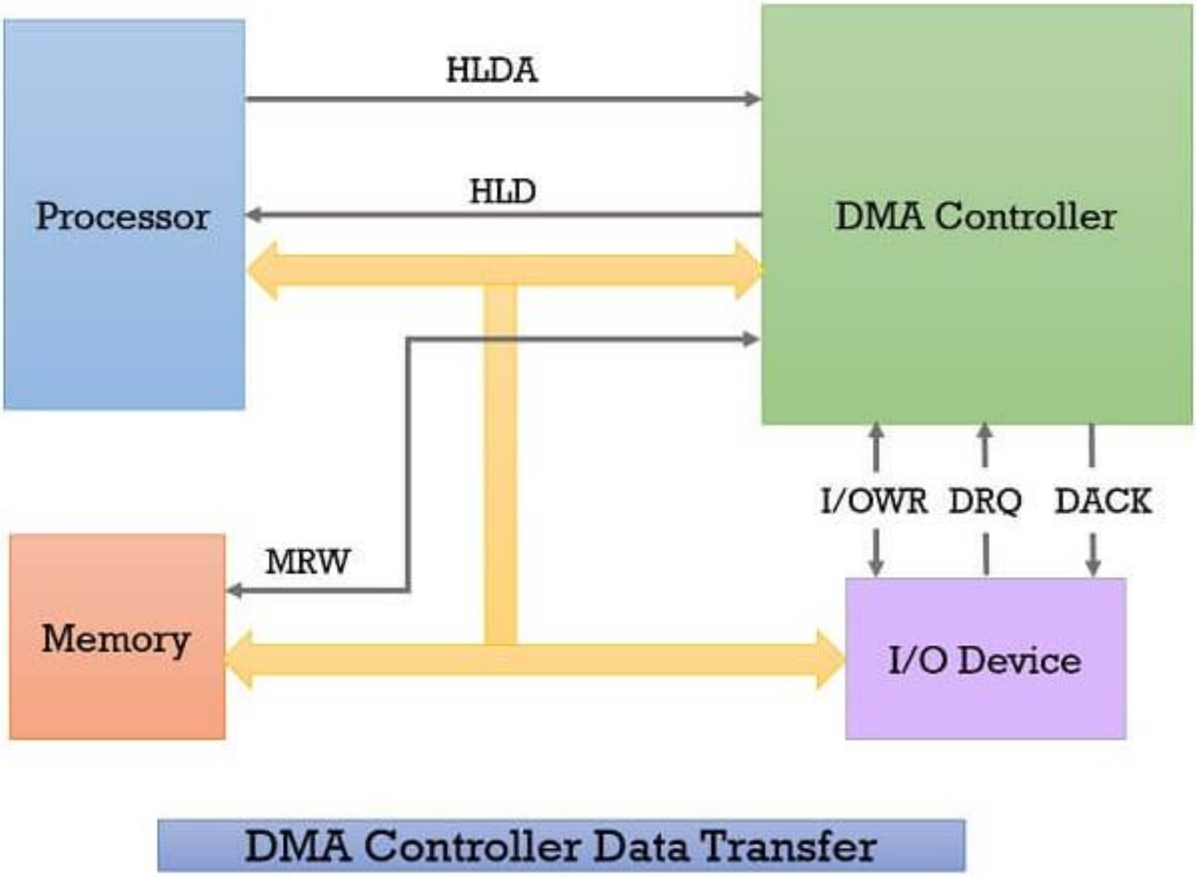
(c) Transparent Mode

- DMA controller only transfers data when CPU is performing operations that do not use system buses.
- The main advantage of this mode is that CPU never stops executing its program.
- The main disadvantages of this mode are
- Hardware needed to determine when the CPU is not using the system buses can be quite complex and relatively expensive.
- Requires highest time to transfer a block of data as compared to above two modes.

Explain DMA operation with the help of its block diagram and timing diagram.

Direct Memory Access Controller & it's Working

DMA controller is a **hardware unit** that allows I/O devices to access memory directly without the participation of the processor. Here, we will discuss the working of the DMA controller. Below we have the diagram of DMA controller that explains its working:

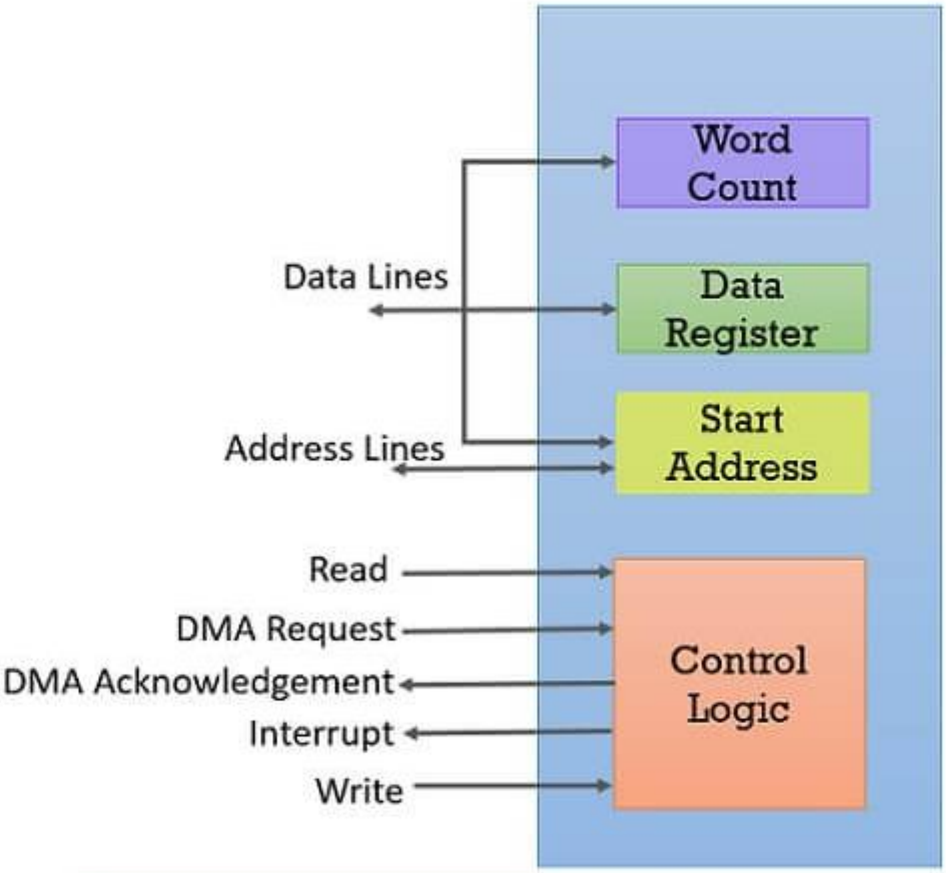


1. Whenever an I/O device wants to transfer the data to or from memory, it sends the DMA request (**DRQ**) to the DMA controller. DMA controller accepts this DRQ and asks the CPU to hold for a few clock cycles by sending it the Hold request (**HLD**).
2. CPU receives the Hold request (HLD) from DMA controller and relinquishes the bus and sends the Hold acknowledgement (**HLDA**) to DMA controller.
3. After receiving the Hold acknowledgement (HLDA), DMA controller acknowledges I/O device **(DACK)** that the data transfer can be performed and DMA controller takes the charge of the system bus and transfers the data to or from memory.
4. When the data transfer is accomplished, the DMA raise an **interrupt** to let know the processor that the task of data transfer is finished and the processor can take control over the bus again and start processing where it has left.

Now the DMA controller can be a separate unit that is shared by various I/O devices, or it can also be a part of the I/O device interface.

Direct Memory Access Diagram

After exploring the working of DMA controller, let us discuss the block diagram of the DMA controller. Below we have a block diagram of DMA controller.



DMA Block Diagram

Whenever a processor is requested to read or write a block of data, i.e. transfer a block of data, it instructs the DMA controller by sending the following information.

1. The first information is whether the data has to be read from memory or the data has to be written to the memory. It passes this information via **read or write control lines** that is between the processor and DMA controllers **control logic unit**.
2. The processor also provides the **starting address** of/ for the data block in the memory, from where the data block in memory has to be read or where the data block has to be written in memory. DMA controller stores this in its **address register**. It is also called the **starting address register**.
3. The processor also sends the **word count**, i.e. how many words are to be read or written. It stores this information in the **data count** or the **word count** register.
4. The most important is the **address of I/O device** that wants to read or write data. This information is stored in the **data register.**

# Interrupt based I/O is efficient compared to polled I/O". 'Justify this statement with general working mechanism in both methods.

Polled interrupt:

Polled interrupt are handled using software and are there fores lower compared to vectored (hardware) interrupts. In this method, there is one common branch address for all interrupts. The program that takes care of interrupts begins at the branch address and polls the interrupts sources in sequence. The order in which they are tested determines the priority of each interrupt. The highest priority source is tested first, and if its interrupt signal is on, control branches to a servicer outline for this source. Otherwise, the next lower priority source is tested, and so on. Thus, the initial service routine for all interrupts consists of a program that tests the interrupt sources in sequence and branches to one of many possible service routines.

Polled interrupts are very simple. But or large number of devices, the time required to poll each device may exceed the service to the device. In such case, the faster mechanism called chained interrupt is used.

Chained interrupt:

This is hardware: concept for f handling the multiple interrupts in this technique, the devices are connected in a chain fashion as shown in figure below for selling up the priority system.

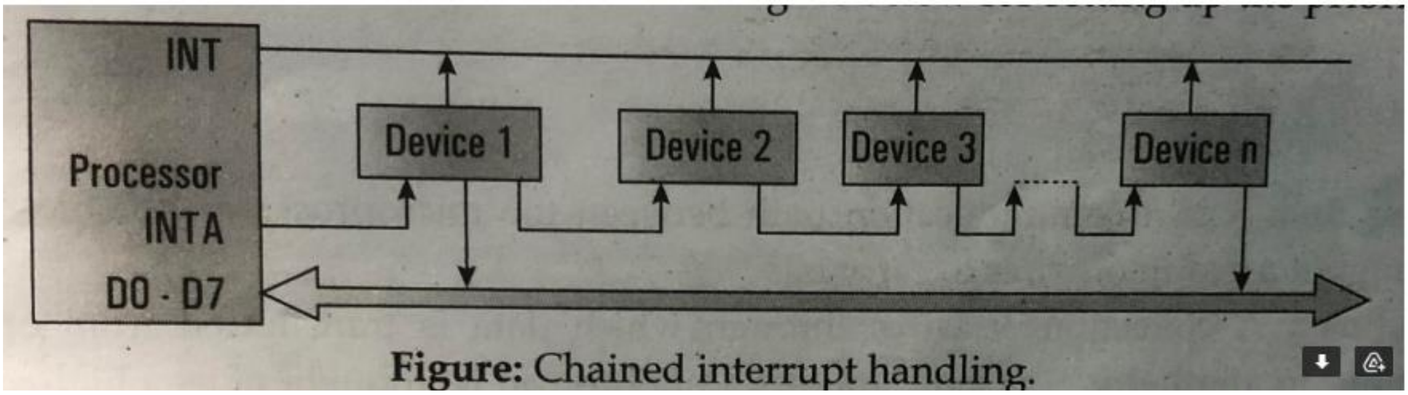


Figure: Chained interrupt handling.

Here the device with the highest priority placed in the first position, followed by lower priority devices. Suppose that on or more devices interrupt the process or at a time. In response, the process or saves its current status and then generates an interrupt acknowledge (INTA) signal to the highest priority device, which is device 1 in our case. If this device has generated the interrupt it will accept the INTA signal from the processor; otherwise, it will pass INTA on to the next device until the INTA is accepted by the interrupting device,

Once accepted, the device provides a means to the process or for finding the interrupt address vector using external hardware. Usually there questing device responds by placing a word on the data lines. With the help of hardware, it generates interrupts vector address. This word is referred to as vector, which the process or used as a pointer to the appropriate device service routine.

This avoids the need to execute a general interrupt service routing first. So this technique is also referred to as vectored interrupts.

#Differentiate between vectored and non- vectored interrupt. Where and how 8259 PIC can be used to handle interrupts.

Vectored internet	Non-Vectored internet
Vectored internet is an interrupt in which the address of the service routine is hardwired.	Non- vectored internet is an internet in which the address of the service routine heads to be supplied externally by the device.
The address of the subroutine is already known to the CPU.	The device will have to supply the address of the subroutine to the microprocessor
In 8085 microprocessor, RIT S.S, RIT 6.5, RST 7.5 and TRAP are vectored interrupts.	In 8085 microprocessor, INTR is non vectored interrupt.
An interrupt for which the internal hardware automatically transfer the program control to a specific memory location is called vectored interment.	An interrupt for which, an external hardware (e.g. I/O device) is required to provide address at which the program control needs to be transfer is called Non- vectored interrupt
Find the address of theses vectored interrupt is very easy.	The address in non- vectored interrupt is not pre-defined.
The TRAP, PST 7.5, RST 6.5 and RST 5.5 are vectored interrupts.	The INTR is a non -vectored interrupt. Hence when a device interrupts through INTR, it has to supply the address of ISR after receiving interrupt acknowledge signal.

The 8259 Programmable Interrupt Controller

The 8259 A programmable interrupt controller designed to work with Intel microprocessors 8085, 8086 and 8088. The 8259A interrupt controller can

- Manage eight interrupts according to the instructions written into its control registers. This is equivalent to proving eight interrupt pins on the processor in place of one INTR (8085) pin.
- Vector can interrupt request anywhere in the memory map. However, all eight interrupts are spaced at the interval of either four or eight locations. This eliminates all the major drawback of the 8085 interrupts in which all interrupts are vectored to memory locations on page **00H**.
- Resolve eight levels of interrupt priorities in a variety of modes, such as fully nested mode, automatic rotation mode, and specific rotation mode.
- Mask each interrupt request individually.
- Read the status of pending interrupts, in-service interrupts, and masked interrupts.
- Be set up to accept either the level- triggered or the edge -triggered interrupt request.
- Be expanded to 64 priority levels by cascading additional 8259As.
- Be set up to work with either the 8085 microprocessor mode or the 886/8088 microprocessor mode.

The 8259A is upward- compatible with its predecessor, the 8259. The main difference between the two is that the 8259 A can be used with Intel's 8086 /88 16-bit microprocessor. It also includes additional features such as the level- triggered mode, buffered mode, and automatic- end-of interrupt mode. To simplify the explanation of the 8259A, illustrative examples will not include the cascade mode or the 8086/88 mode and will be limited to modes continuously used with the 8085.

#What is hardware and software interrupt? Difference between Hardware Interrupt and Software Interrupt.

1. **Hardware Interrupt :**

Hardware Interrupt is caused by some hardware device such as request to start an I/O, a hardware failure or something similar. Hardware interrupts were introduced as a way to avoid wasting the processor's valuable time in polling loops, waiting for external events.

For example, when an I/O operation is completed such as reading some data into the computer from a tape drive.

2. Software Interrupt :

Software Interrupt is invoked by the use of INT instruction. This event immediately stops execution of the program and passes execution over to the INT handler. The INT handler is usually a part of the operating system and determines the action to be taken. It occurs when an application program terminates or requests certain services from the operating system.

For example, output to the screen, execute file etc.

Difference between Hardware Interrupt and Software Interrupt.

S.N.	Hardware Interrupt	Software Interrupt
1	Hardware interrupt is an interrupt generated from an external device or hardware.	Software interrupt is the interrupt that is generated by any internal system of the computer.
2	It do not increment the program counter.	It increment the program counter.
3	Hardware interrupt can be invoked with some external device such as request to start an I/O or occurrence of a hardware failure.	Software interrupt can be invoked with the help of INT instruction.
4	It has lowest priority than software interrupts	It has highest priority among all interrupts.
5	Hardware interrupt is triggered by external hardware and is considered one of the ways to communicate with the outside peripherals, hardware.	Software interrupt is triggered by software and considered one of the ways to communicate with kernel or to trigger system calls, especially during error or exception handling.
6	It is an asynchronous event.	It is synchronous event.
7	Hardware interrupts can be classified into two types they are: 1. Maskable Interrupt. 2. Non Maskable Interrupt.	Software interrupts can be classified into two types they are: 1. Normal Interrupts. 2. Exception
8	Keystroke depressions and mouse movements are examples of hardware interrupt.	All system calls are examples of software interrupts

#What is the Difference Between Maskable and Non Maskable Interrupt

The main difference between maskable and non maskable interrupt is that **a CPU can either disable or ignore a maskable interrupt, but it is not possible to disable or ignore a non-maskable interrupt by the instructions of a CPU.**

Generally, an interrupt is an event caused by a component other than the CPU. It indicates the CPU of an external event that requires immediate attention. Furthermore, interrupts occur asynchronously. Maskable and non-maskable interrupts are two types of interrupts.

S.N.	Maskable Interrupt	Non-Maskable Interrupt
1	Maskable interrupt is a hardware Interrupt that can be disabled or ignored by the instructions of CPU.	A non-maskable interrupt is a hardware interrupt that cannot be disabled or ignored by the instructions of CPU.
2	When maskable interrupt occur, it can be handled after executing the current instruction.	When non-maskable interrupts occur, the current instructions and status are stored in stack for the CPU to handle the interrupt.
3	Maskable interrupts help to handle lower priority tasks.	Non-maskable interrupt help to handle higher priority tasks such as watchdog timer.
4	Maskable interrupts used to interface with peripheral device.	Non maskable interrupt used for emergency purpose e.g power failure, smoke detector etc .
5	In maskable interrupts, response time is high.	In non maskable interrupts, response time is low.
6	It may be vectored or non-vectored.	All are vectored interrupts.
7	Operation can be masked or made pending.	Operation Cannot be masked or made pending.
8	RST6.5, RST7.5, and RST5.5 of 8085 are some common examples of maskable Interrupts.	Trap of 8085 microprocessor is an example for non-maskable interrupt.

Some Exam Questions:

a. Describe the working mechanism of DMA. Draw the internal architecture of the 8237 DMAC along with a timing diagram illustrating the process of DMA transfers.
b. Differentiate between vectored and non-vectored interrupts. Where and how 8259 PIC can be used to handle interrupts.
c. What is the importance of interrupt in microprocessor based system? Explain how interrupt controller(8259) can be used to handle interrupts.
d. What is DMA? Explain DMA data transfer with suitable diagram.

****