Computer Organization (CO)

UNIT-4

I/O Organization

Syllabus:

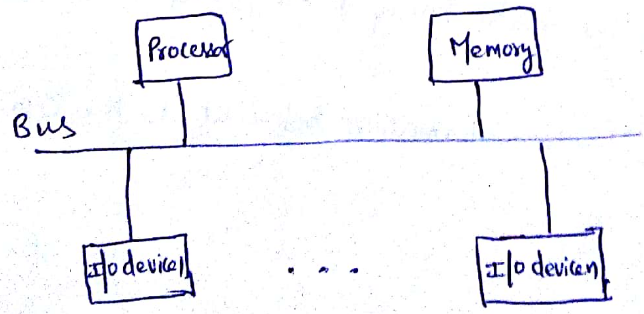
- Accessing I/O devices
- Interrupts
  - Interrupt Hardware
  - Enabling and Disabling Interrupts
  - Handling Multiple devices. ✓
- Direct Memory Access ✓
- Buses
  - Synchronous Bus
  - Asynchronous Bus ✓
- Interface Circuits ✓
- Standard I/O Interface ✓
  - Peripheral Component Interconnect (PCI) Bus
  - Universal Serial Bus (USB) ✓

① Accessing I/O devices:

→ The basic feature of a Computer is its ability to exchange data with other devices.

→ The bus enables all the devices connected to it to exchange information.

→ A Single bus structure is depicted as



→ ~~Bus~~ The bus consists of 3 sets of lines

- Address lines
- Data lines
- Control lines

- Each I/O device is assigned a unique set of addresses
- When I/O device and the memory share the same address space, the arrangement is called Memory-mapped I/O.

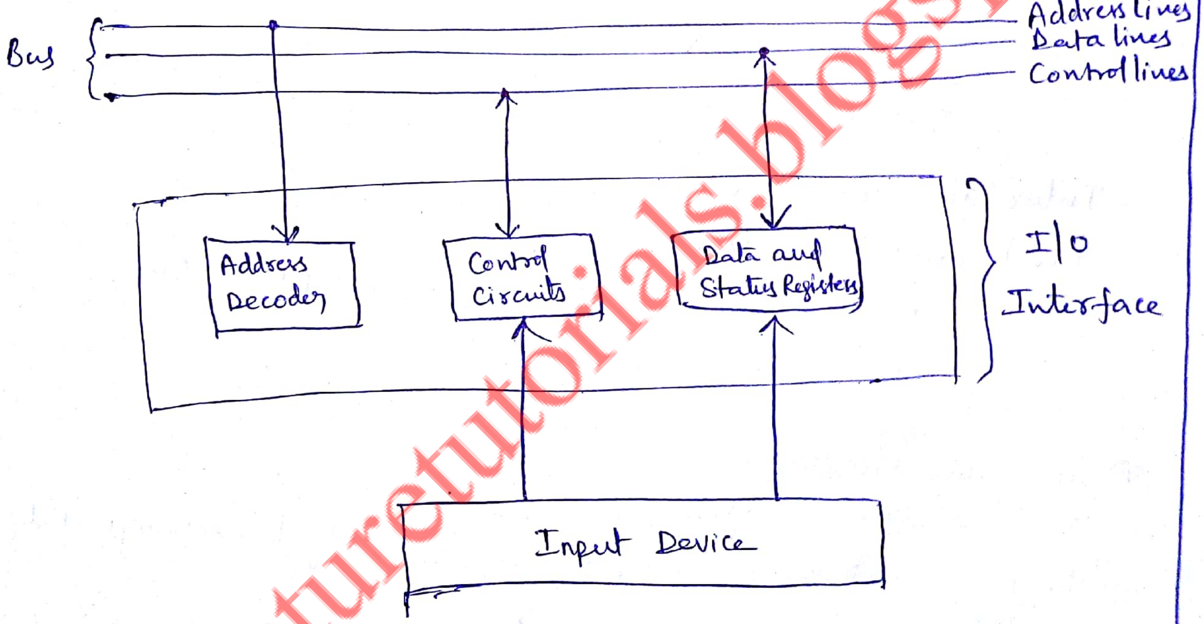
Memory-mapped I/O:

- With Memory-Mapped I/O, any machine instruction that can access memory can be used to transfer data to (or) from an I/O device.

Example:

MOVE DATAIN, R0 (Input Device to Processor Register)
MOVE R0, DATAOUT (Processor Register to Output Device)

- The I/O Interface for an Input device is depicted as



- Here, the address decoder enables the device to recognize its address when this address appears on the address lines.
- The data register holds the data being transferred to (or) from the processor
- The status register contains information relevant to the operation of the I/O device.

② Interrupts :

→ An Interrupt will stop the continuous progress of an activity (or) process.

→ An Interrupt is a signal to the processor emitted by hardware (or) software Indicating an event that needs immediate attention.

→ ~~An Interrupt~~ It alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.

→ The bus control line, called interrupt-request line is used for interrupts.

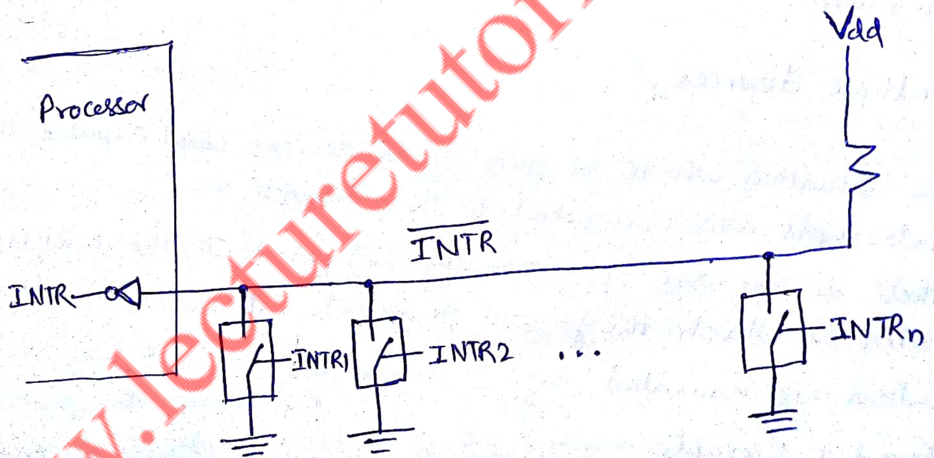
i) Interrupt Hardware :

→ An I/o device requests an interrupt by activating a bus line called interrupt-request.

→ Most computers are likely to have several I/o devices that can request an interrupt.

→ A Single interrupt request line may be used to serve n-devices

→ An equivalent circuit for an open-drain bus used to implement a common interrupt-request line is depicted as



→ All devices are connected to the line via switches to ground.

→ To request an interrupt, a device closes its associated switch.

→ Thus, if all interrupt_request signals $INTR_1$ to $INTR_n$ are inactive, i.e, if all switches are open, the voltage on the interrupt-request line will be equal to V_{dd}.

→ This is the inactive state of the line.

→ Since the closing of one (or) more switches will cause the line voltage to drop to 0, the value of INTR is the logical OR of the requests from individual devices, i.e,

$$INTR = INTR_1 + INTR_2 + \dots + INTR_n$$

→ The $\overline{INTR}$ signal is active when in the low voltage state.

ii) Enabling and Disabling Interrupts:

→ The sequence of events involved in handling interrupt request from a single device are,

- The device raises an interrupt request.
- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the PS (Processor Status register)
- The device is informed that its request has been recognized, and in response, it deactivates the interrupt request signal
- The action requested by the interrupt is performed by the interrupt-service routine
- Interrupts are enabled and execution of the interrupt program is resumed.

(iii) Handling Multiple devices:

→ Consider the situation where a number of devices ~~capd~~ capable of initiating interrupts are connected to the processor.

→ Because these devices are operationally independent, there is no definite order in which they will generate interrupts.

→ This situation is handled by 3 methods.

(a) Vectored Interrupts

(b) Interrupt Nesting

(c) Simultaneous Requests.

a) Vectored Interrupts:

→ A device requesting an interrupt may identify directly to the processor

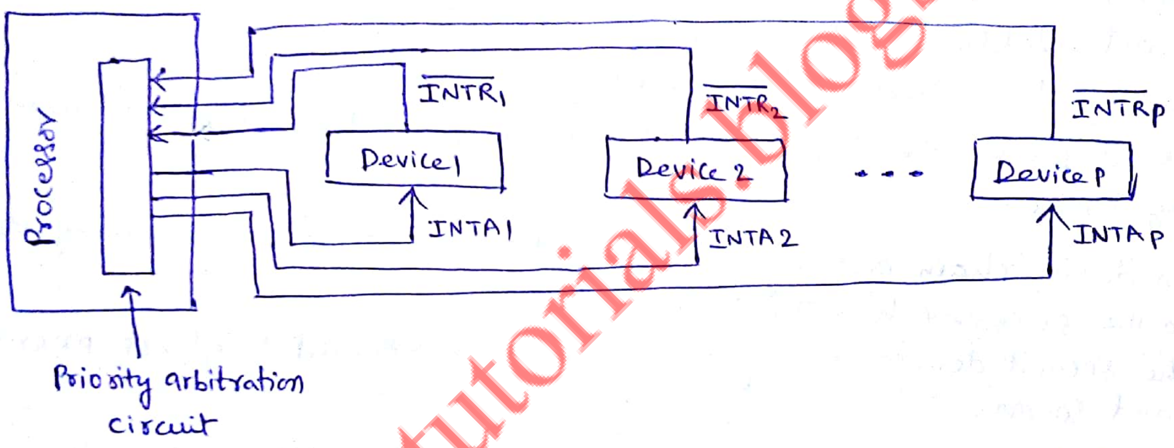
→ Then the processor can immediately start executing the corresponding interrupt-service routine.

→ This interrupt handling Scheme is known as Vectored Interrupts.

→ A commonly used scheme is to allocate permanently an area in the memory to hold the addresses of interrupt-service routines.
→ These addresses are referred as Interrupt vectors, and they are said to constitute the interrupt-vector table.
→ For example, 128 bytes may be allocated to hold a table of 32 interrupt vectors.

b) Interrupt Nesting:

→ Implementation of Interrupt priority using individual interrupt-request and Acknowledge lines is depicted as,

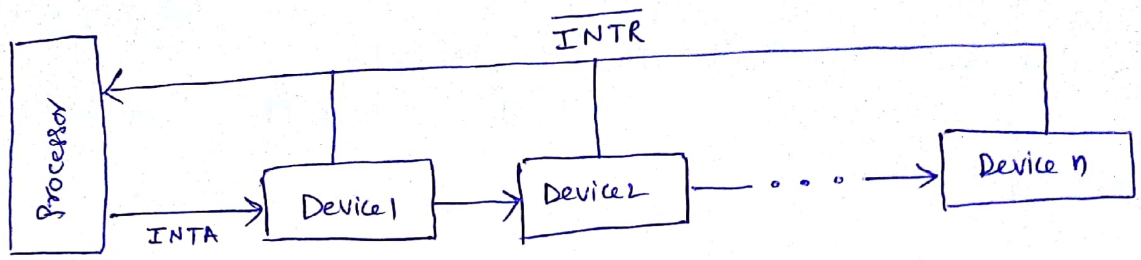


→ Here, Each of the interrupt-request lines is assigned a different Priority level.
→ Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.
→ A request is accepted only if it has a higher priority level than that currently assigned to the processor.

c) Simultaneous Requests:

→ Consider the problem of simultaneous arrivals of interrupts requests from two (or) more devices.
→ The processor must have some means of deciding which request to service first.
→ The processor simply accepts the request having the highest priority.
→ Priority is determined by the order in which the devices are polled, i.e, the polling the status registers of the I/O devices.

→ A Daisy chain priority interrupt Scheme is depicted as,



→ The Interrupt-request line INTR is common to all devices.

→ The Interrupt-Acknowledge line INTA, is connected in a daisy-chain fashion.

→ The INTA signal propagates serially through the devices.

→ When several devices raise an interrupt request and the INTR line is activated, the processor responds by setting the INTA line to 1.

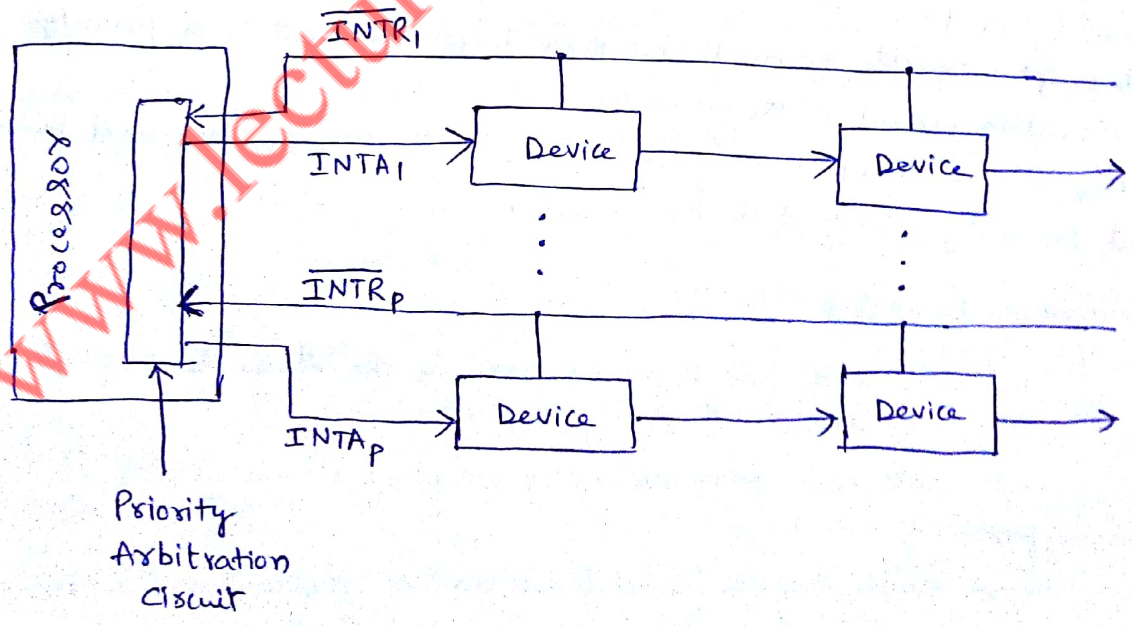
→ This signal is received by device 1.

→ Device 1 passes the signal on to device 2 only if it doesnot require any service.

→ In daisy-chain arrangement, the device that is electronically closest to the processor has the highest priority.

→ The second device along the chain has second highest priority, and so on.

→ The arrangement of priority groups is depicted as,



→ Here, Devices are organized in groups, and each group is connected at a ~~at~~ different priority level.

→ Within a group, devices are connected in a daisy-chain.

→ This organization is used in many computer systems.

③ DMA : (Direct Memory Access)

→ To transfer large blocks of data at high speed, a special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor.

→ This approach is called Direct Memory Access (DMA)

→ DMA transfers are performed by a control unit that is part of the I/o device interface, called DMA controller.

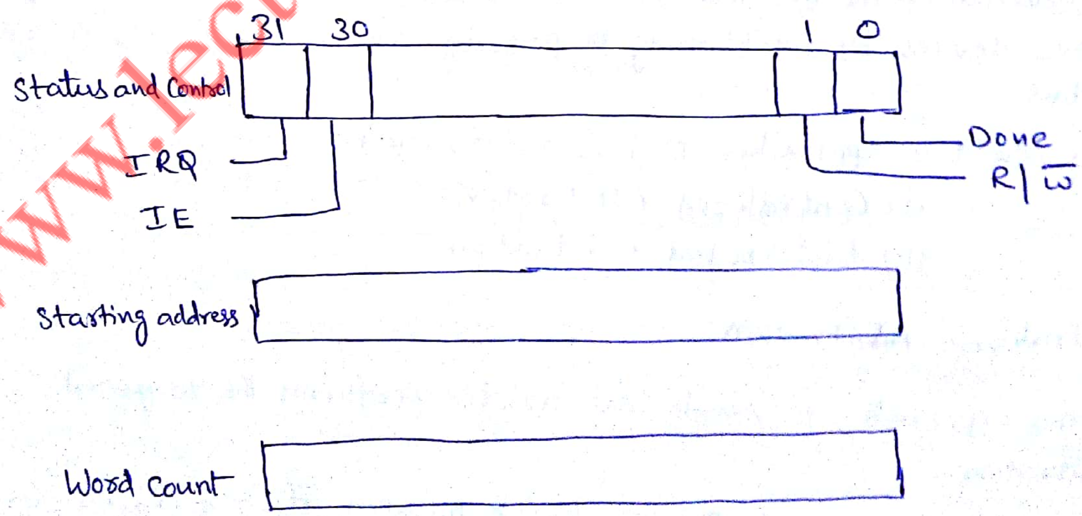
(*) DMA controller:

→ The DMA controller performs the functions that would normally be carried out by the processor when accessing the main memory.

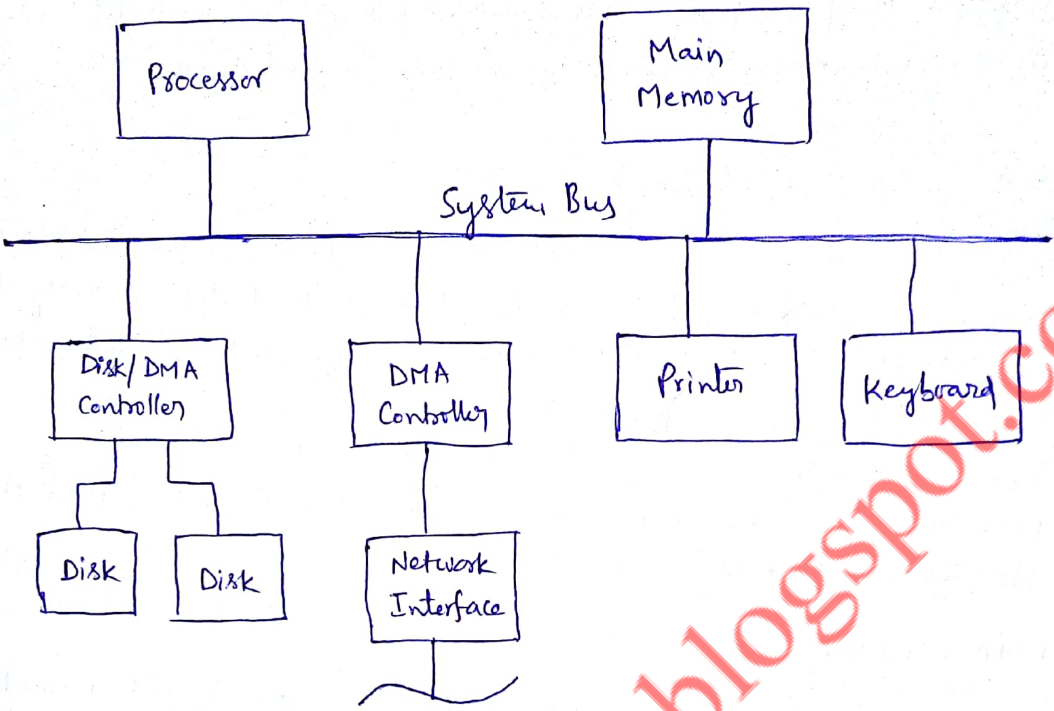
→ For each word transferred, it provides the memory address and all the bus signals that ~~are~~ control data transfer.

→ Although the DMA controller can transfer data without intervention by the processor, its operation must be under the control of a program executed by the processor.

→ The Registers used in DMA Controller are depicted as,



→ The use of DMA Controllers in a Computer System is depicted as,



(*) Bus Arbitration :

→ The device that is allowed to initiate data transfers on the bus at any given time is called the Bus Master .

→ Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.

→ The selection of the bus master must take into account the needs of various devices by establishing a priority system for gaining access to the bus.

→ There are two approaches to bus arbitration

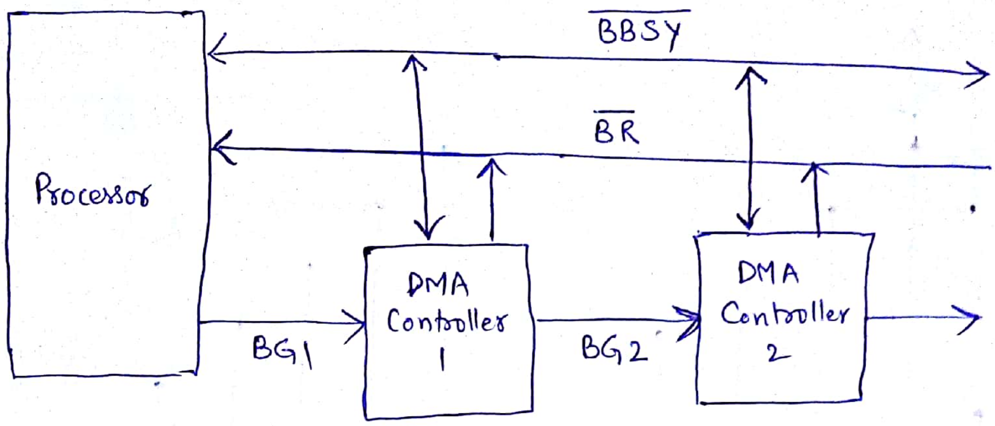
(i) Centralized Arbitration

(ii) Distributed Arbitration

(i) Centralized Arbitration :

→ In this approach, a single bus arbiter performs the required arbitration.

→ A simple arrangement for bus arbitration using a daisy chain is depicted as



→ In this case, the processor is normally the bus master unless it grants bus mastership to one of the DMA controllers.

→ A DMA controller indicates that it needs to become the bus master by activating the Bus-Request line, $\overline{BR}$

→ When Bus-Request is activated, the processor activates the Bus-Grant signal BG1, and this signal is connected to all DMA controllers using a daisy-chain arrangement.

→ The current bus master indicates to all devices that it is using the bus by activating another open-collector line called Bus Busy, $\overline{BBSY}$

→ Hence, after receiving the Bus-Grant signal, a DMA controller waits for Bus-Busy to become inactive, then assumes mastership of the bus.

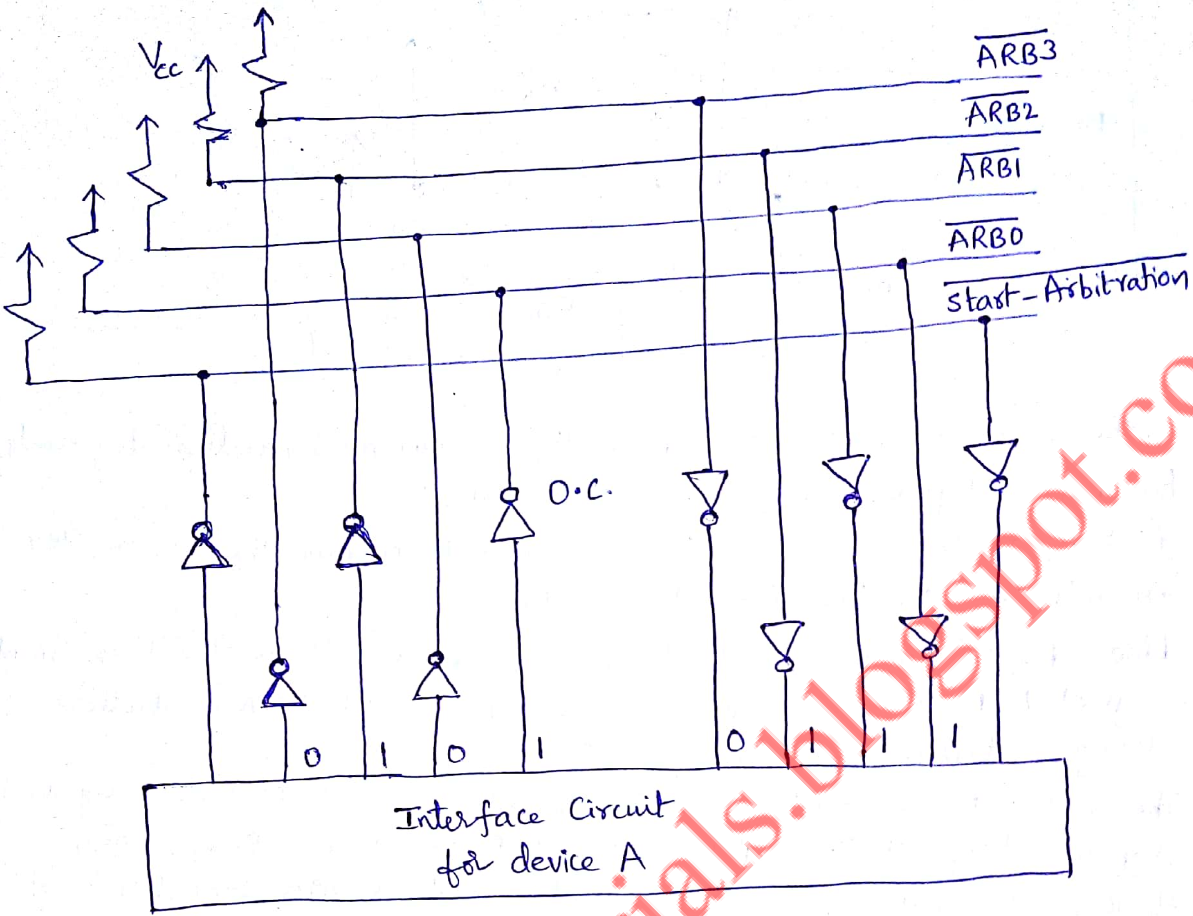
→ At this time, it activates Bus-Busy to prevent other devices from using the bus at the same time.

(ii) Distributed Arbitration:

→ In this approach, all devices participate in the selection of the next bus master.

→ Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.

→ A distributed arbitration scheme is depicted as,



→ Each device on the bus is assigned a 4-bit identification number.

→ When one (or) more devices request the bus, they assert the $\overline{\text{Start-Arbitration}}$ signal and place their 4-bit ID numbers on four open-collector lines, $\overline{ARB0}$ through $\overline{ARB3}$

→ A winner is selected as a result of the interaction among the signals transmitted over these lines by all contenders.

→ The net outcome is that the code on the four lines represents the request that has the highest ID number.

(4) BUSES :

→ The processor, main memory, and I/O devices can be interconnected by means of a Common bus.

→ The Bus primary function is to provide a communication path for the transfer of data.

→ The bus includes the lines needed to support interrupts and arbitration.

→ The bus lines used for transferring data may be grouped into three types.

- data lines
- address lines
- Control lines

— In any data transfer operation, one device plays the role of a master.

— This is the device that initiates data transfers by issuing read (or) write Commands on the bus, hence, it may be called an initiator.

— Normally, the processor (or) devices with DMA capability become bus masters.

— The device addressed by the master is referred to as a slave (or) target.

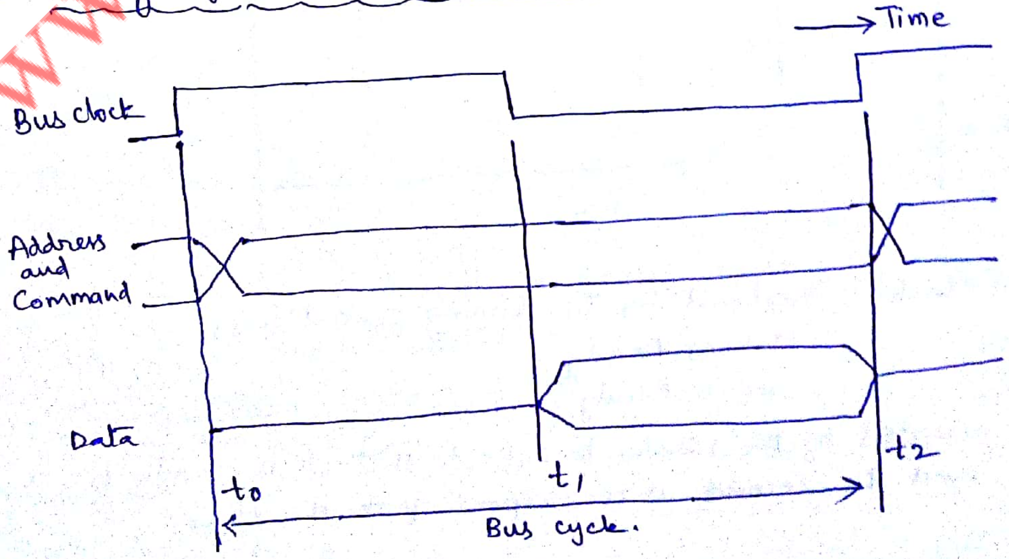
(i) Synchronous Bus :

— In a synchronous bus, all devices derive timing information from a common clock line.

— Equally spaced pulses on this line define equal time intervals.

— In the simplest form of a synchronous bus, each of these intervals constitutes a bus cycle during which one data transfer can take place.

— Timing of an input transfer on a Synchronous bus is depicted as



→ The address and datalines are high and low at the same time.

→ This is a common convention indicating that some lines are high and ~~some~~ some low, depending on the particular address (or) data pattern being transmitted.

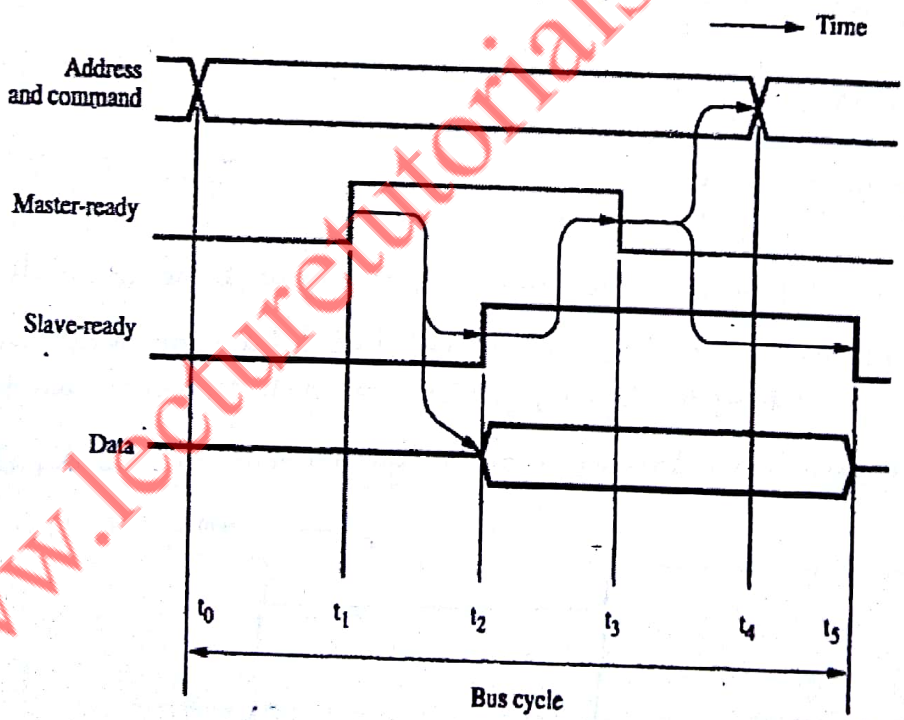
→ The crossing points indicate the times at which these patterns change.

→ A signal line is an indeterminate (or) high-impedance state is represented by an intermediate level half-way between the low and high signal levels.

(ii) Asynchronous Bus:

→ An alternative scheme for controlling data transfer on the bus is based on the use of a handshake between the master and the slave.

→ Handshake control of data transfer during an input operation is depicted as,



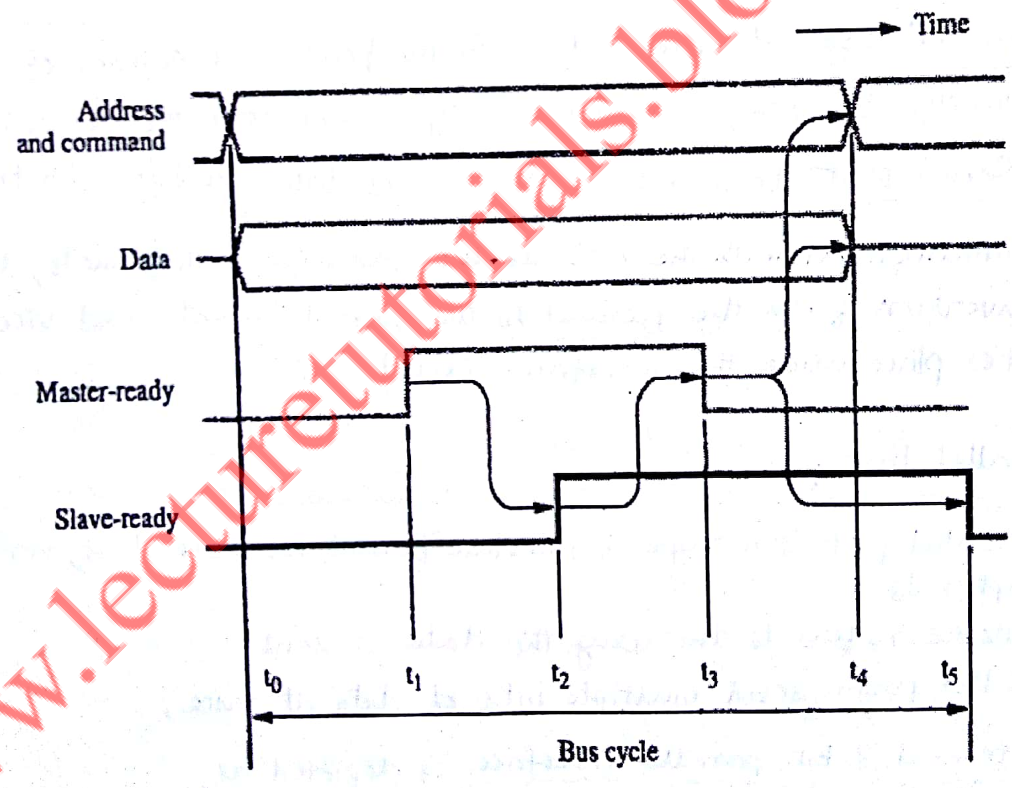
→ The common clock is replaced by two timing control lines
- Master-Ready
- Slave-Ready

→ The first is asserted by the master to indicate that it is ready for a transaction, and the second is a response from the slave.

→ In principle, a datatransfer controlled by a handshake proceeds as,

- The master places the address and command information on the bus.
- Then it indicates to all devices that it has done so by activating the Master-ready line.
- This causes all devices on the bus to decode the address.
- The selected slave performs the required operation and informs the processor it has done so by activating the Slave-ready line.
- The master waits for Slave-ready to become asserted before it removes its signals from the bus.
- In the case of a read operation, it also strobes the data into its input buffer.

→ Handshake control of data-transfer during an output operation is depicted as



→ In this case, the master places the output data on the data lines at the same time that it transmits the address and command information.

→ The selected slave strobes the data into its output buffer when it receives the Master-ready signal and indicates that it has done so by setting the Slave-ready signal to 1.

→ The remainder of the cycle is identical to the input operation.

5 Interface Circuits:

→ An I/O interface consists of the circuitry required to connect an I/O device to a computer bus.

→ On one side of the interface, we have the bus signals for address, data and control.

→ On the otherside we have a data path with its associated controls to transfer data between the interface and the I/O device.

→ This is called a port.

→ It can be classified as

(i) Parallel Port
(ii) Serial Port

→ A parallel port transfers data in the form of a number of bits, typically 8 (or) 16, simultaneously to (or) from the device.

→ A Serial port transmits and receives data one bit at a time

→ Communication with the bus is the same for both formats, the conversion from the parallel to the serial format, and vice versa, takes place inside the interface circuit.

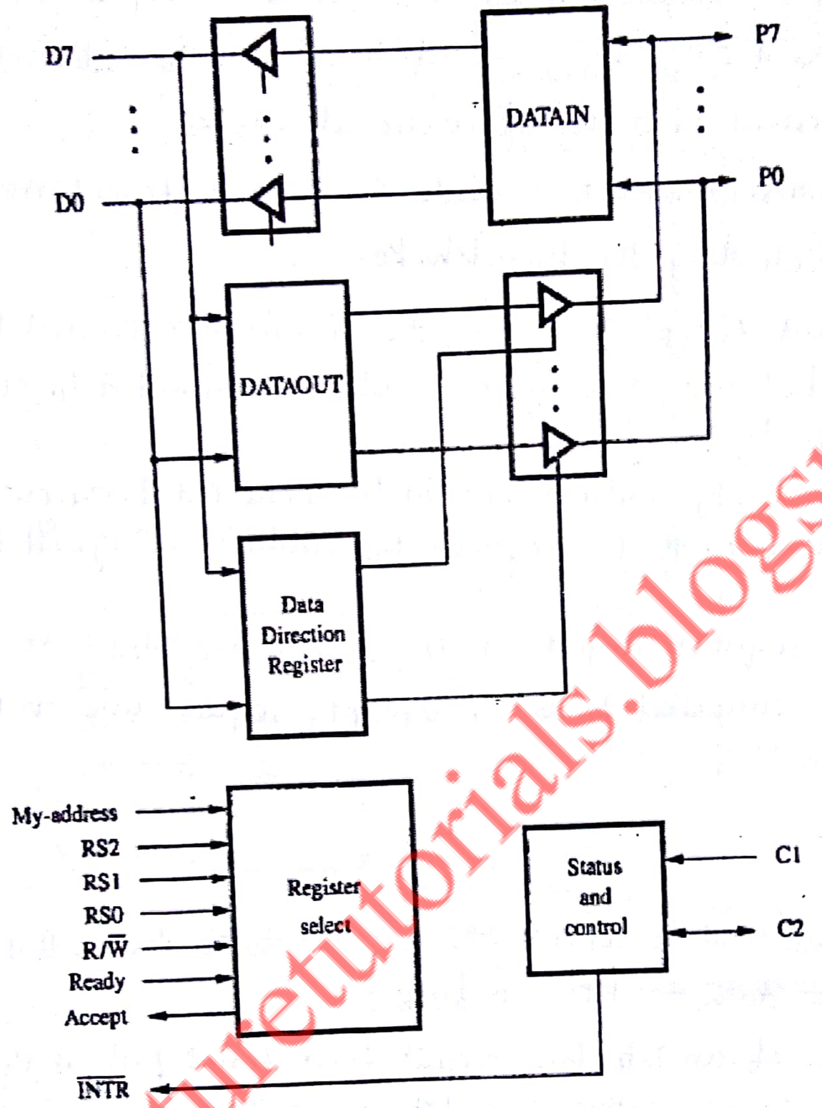
(i) Parallel Port:

→ A parallel port is a type of interface found on computers, for connecting peripherals.

→ The name refers to the way the data is sent

→ parallel ports send multiple bits of data at once.

→ A general 8-bit parallel interface is depicted as,



→ Data lines P7 through P0 can be used for either input (or) output purposes.

→ For increased flexibility, the circuit makes it possible for some lines to ~~have~~ serve as inputs and some lines to serve as outputs, under Program control.

→ The DATAOUT register is connected to these lines via three-state drivers that are controlled by a data direction register, DDR.

→ For a given bit, if the DDR value is 1, the corresponding data lines acts as an output line, otherwise, it acts as an input line.

→ Two lines C1 and C2, are provided to control the interaction between the interface circuit and the I/O device it serves.

→ Line C2 is bidirectional to provide several different modes of signaling, including the handshake

→ The Ready and Accept lines are the handshake control lines on the processor bus side, and hence would be connected to Master-ready and Slave-ready.

→ The input signal My-address should be connected to the output of an address decoder that recognizes the address assigned to the interface.

→ An interrupt request output, $\overline{INTR}$, is also provided.

→ It should be connected to the interrupt-request line on the Computer bus.

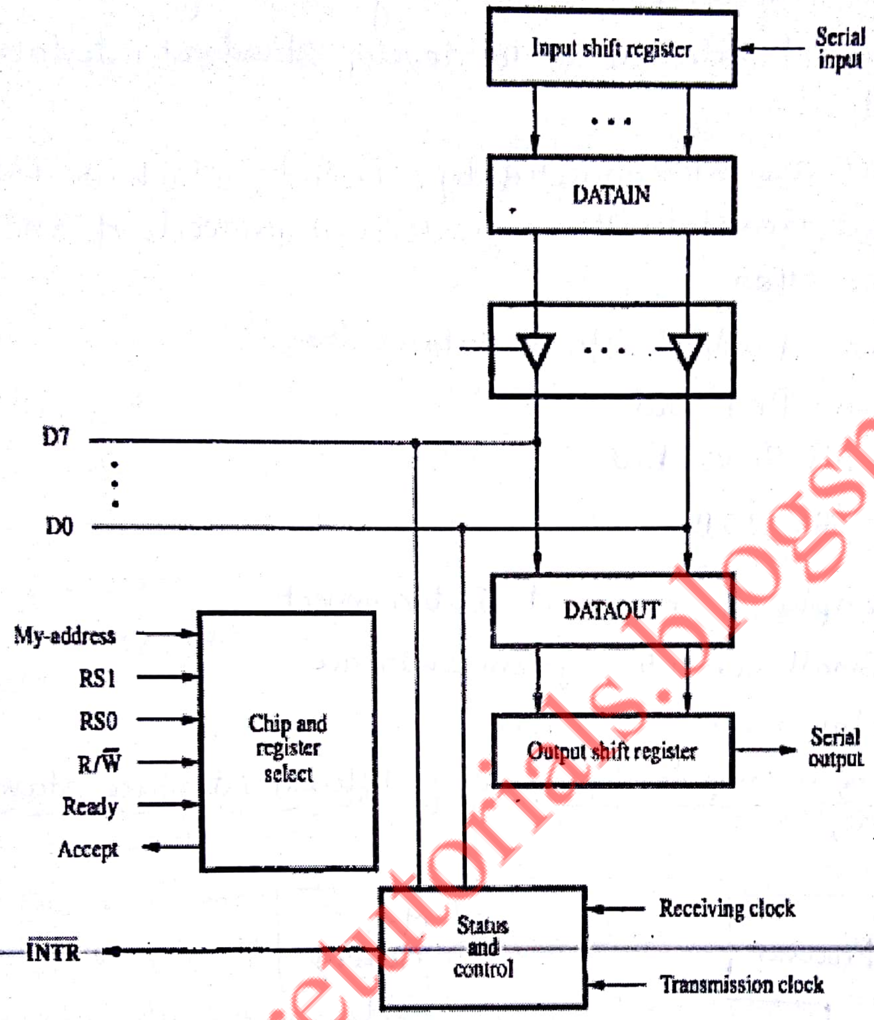
(ii) <u>Serial Port</u>:

→ A serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time.

→ The key feature of an interface circuit for a serial port is that it is capable of communicating in a bit-serial fashion on the device side and in a bit-parallel fashion on the bus side.

→ The transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.

→ A serial interface is depicted as,



→ It includes the familiar DATAIN and DATAOUT registers.

→ The input shift register accepts bit-serial input from the I/O device.

→ When all 8 bits of data have been received, the contents of this shift register are loaded in parallel into the DATAIN register.

→ Similarly, output data in the DATAOUT register are loaded into the output shift register, from which the bits are shifted out and sent to the I/O device.

⑥ Standard I/O Interfaces:

→ A different interface may have to be designed for every combination of I/O device and computer, resulting in many different interfaces.

→ The most practical solution is to develop standard interface signals and protocols

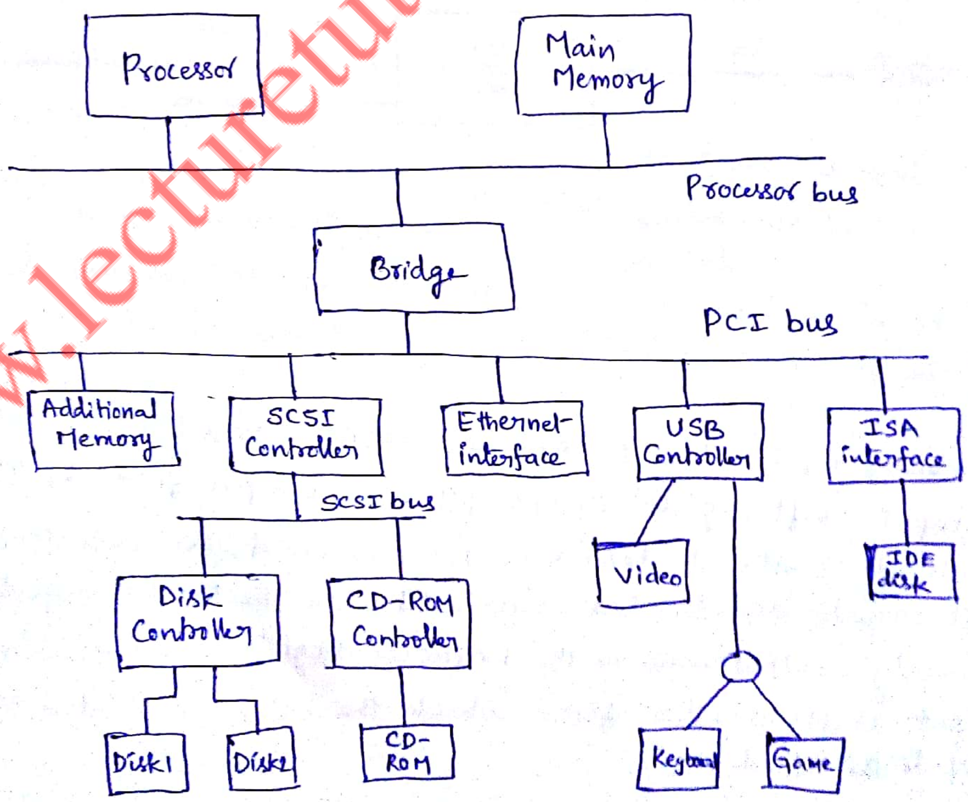
→ The two buses are interconnected by a circuit, which we will call a bridge, that translates the signals and protocols of one bus into those of the other.

→ The different standard I/O Interfaces are,

(i) PCI bus
(ii) SCSI bus
(iii) USB

→ PCI – Peripheral component Interconnect
SCSI – Small Computer System Interface
USB – Universal Serial Bus

→ An example of a computer system using different interface standards is depicted as,



→ ISA – Industry Standard Architecture
IDE – Integrated Device Electronics

→ The PCI standards defines an expansion bus on the motherboard.

→ SCSI and USB are used for connecting additional devices, both inside and outside the computer box.

(i) PCI (Peripheral Component Interconnect) Bus:

→ The PCI bus is a good example of a system bus that grewout of the need for standardization.

→ It supports the functions found on a processor bus but in a standardized format that is independent of any particular processor.

→ Devices connected to the PCI bus appear to the processor as if they were connected directly to the processor bus.

→ They are assigned addresses in the memory address space of the processor.

Data transfer:

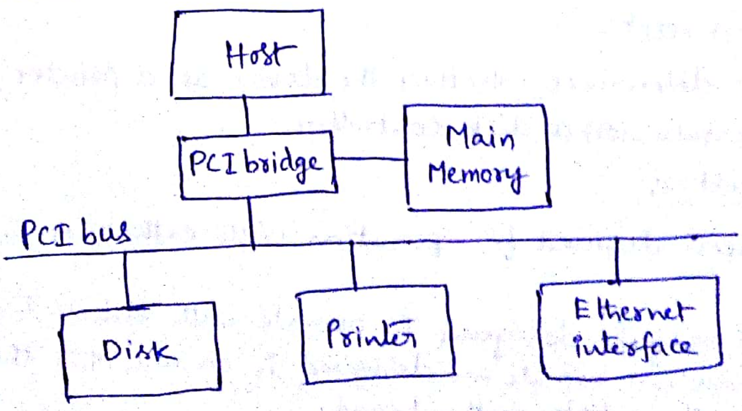
→ When the processor specifies an address and requests a read operation from the main memory, the memory responds by sending a sequence of data words starting at that address.

→ Similarly, during write operation, the processor sends a memory address followed by a sequence of data words, to be written in successive memory locations starting at that address.

→ The PCI is designed primarily to support this mode of operation.

→ A read (or) write operation involving a single word is simply treated as a burst of length one.

→ The use of a PCI bus in a Computer System is depicted as,



→ The PCI bridge provides a seperate physical connection for the main memory.
→ For electrical reasons, the bus may be further divided into segments connected via bridges.
→ However, regardless of which bus segment a device is connected to, it may still be mapped into the processor's memory address space
→ The Data Transfer Signals on the PCI bus is given as

Name	Function
CLK	A 33-MHz (or) 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transaction
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 Command / Byte-Enable lines (8 for 64-bit bus)
IRDY#, TRDY#	Initiator-Ready, Target-Ready signals
IDSEL#	Initialization Device Select

Device Configuration:
→ The PCI simplifies the process by incorporating in each I/o device interface a small Configuration ROM memory that stores information about that device.
→ The Configuration ROMs of all devices are accessible in the Configuration address space.
→ The PCI initialization software reads these ROMs whenever the system is powered up (or) reset.
→ In each case, it determines whether the device is a printer, a keyboard, an Ethernet Interface, (or) a disk controller.

Electrical characteristics:
→ The PCI bus has been defined for operation with either a 5-V (or) 3.3V power supply.
→ The motherboard may be designed to operate with either signaling system.
→ Connectors on expansion boards are designed to ensure that they can be plugged only in a compatible motherboard.

(ii) SCSI Bus :

→ SCSI stands for Small Computer System Interface.

→ It refers to a standard bus defined by the ANSI (American National Standards Institute) under the designation X3.131

→ In the original specifications of the standard, devices such as disks are connected to a computer via a 50-wire cable, which can be upto 25 meters in length and can transfer data at rates up to 5 megabytes/s

→ A SCSI bus may have 8 data lines, in which case it is called a narrow bus and transfers data one byte at a time.

→ A wide SCSI bus has 16 data lines and transfers data 16-bits at a time.

→ The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years.

→ SCSI-2, SCSI-3 have been defined, and each has several options.

→ The different SCSI bus signals are

- DB (Data lines)
- DB(P) (Parity bit for the data bus)
- BSY (Busy)
- SEL (Selection)
- C/D (Control/Data)
- MSG (Message)
- REQ (Request)
- ACK (Acknowledge)
- I/O (Input/Output)
- ATN (Attention)
- RST (Reset)

(iii) USB (Universal Serial Bus):

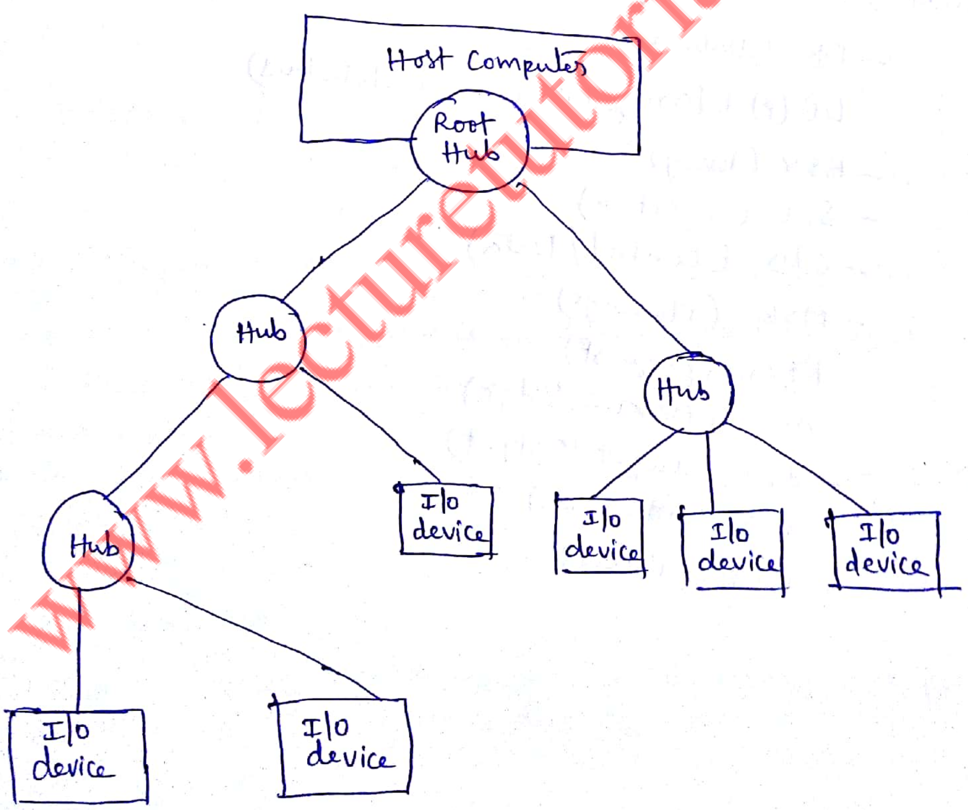
→ A simple, low-cost mechanism to connect the devices to the computer is possible using USB.

→ The USB supports two speeds of operation
- * Low-speed (1.5 Mb/s)
- * Full-speed (12 Mb/s)

→ The recent development is USB 2.0

→ The USB has been designed to meet several key objectives.
- Provide a simple, low-cost, and easy to use interconnection system.
- Accommodate a wide range of data transfer characteristics for I/O devices, including telephone and Internet connections.
- Enhance user convenience through a "plug-and-play" mode of operation.

→ USB Tree structure is depicted as,



→ To accommodate a large number of devices that can be added (or) removed at any time, the USB has the tree structure.

→ Each node of the tree has a device called a hub, which acts as an intermediate control point between the host and the I/O devices.

→ At the root of the tree, a root hub connects the entire tree to the host computer.

→ The leaves of the tree are the I/O devices being served (for example, keyboard, Internet connection, speaker (or) digital TV), which are called functions in USB terminology.

USB protocols:

→ All information transferred over the USB is organized in packets, where a packet consists of one (or) more bytes of information.

→ The information ~~divided~~ transferred on the USB can be divided into two broad categories.

— Control
— Data

→ Control packets perform such tasks as addressing a device to initiate data transfer, acknowledging that data have been received correctly, (or) indicating error.

→ Data packets carry information that is delivered to a device.

→ A packet contains one or more fields with different kinds of information.

→ The first field of any packet is called the packet identifier, PID, which identifies the type of that packet.

→ USB packet formats is depicted as,

PID_0	PID_1	PID_2	PID_3	$\overline{PID_0}$	$\overline{PID_1}$	$\overline{PID_2}$	$\overline{PID_3}$

(a) Packet Identifier fied

Bits	8	7	4	5
	PID	ADDR	ENDP	CRC16

(b) Token Packet, IN (or) Out

Bits	8	0 to 8192	16
	PID	DATA	CRC16

(c) Data packet

→ ADDR – Address
ENDP – Ending Packet
CRC – Cyclic Redundancy ~~Check~~ Check

Isochronous Traffic on USB:

→ One of the key objectives of the USB is to support the transfer of isochronous data, such as sample voice, in a simple manner.

→ Devices that generate (or) receive isochronous data require a time reference to control the sampling process.

→ To provide this reference, transmission over the USB is divided into frames of equal length.

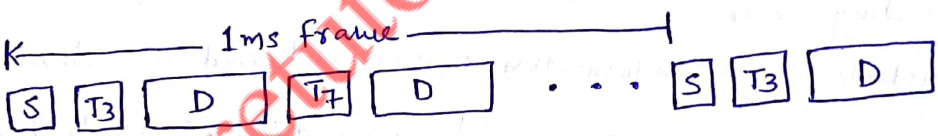
→ A frame is 1ms long for full and low speed data.

→ The root hub generates a Start Of Frame (SOF) control packet precisely once every 1 ms to mark the beginning of a new frame.

→ USB frames depicted as,

Bits	8	11	5
	PID	Frame Number	CRC5

(a) SOF Packet



(b) Frame Example

Electrical characteristics,

→ The cables used for USB connections consist of four wires
- Two are used to carry power, +5V and Ground.
- The other two wires are used to carry data

CO-UNIT4 - Short Answer Questions

① Discuss about Single bus structure.

Page 4.1

② Explain about Memory-mapped I/o.

page 4.2

③ What is programmed I/o?

Programmed I/o:

→ Programmed I/o is a method of transferring data between the CPU and a peripheral.

Advantages:

→ I/o command is issued by the processor to the respective I/o module.
→ Requested I/o instruction is executed at the earliest.
→ I/o module switches to the next task as soon as it completes the task.

Disadvantages:

→ Processor has to wait until the I/o module is ready for performing data transfer.

→ Processor have to interrogate several times regarding the status of I/o module.

④ What is interrupt-initiated data transfer?

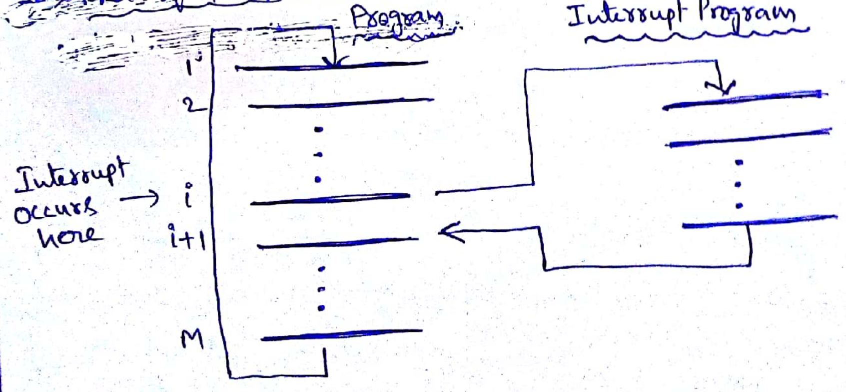
Interrupt-initiated Data transfer:

→ When the I/o device is ready to transfer data, instead CPU keep asking, does not check the flag.

→ When the flag in the interface is set, the interface will initiate an interrupt, and the CPU will drops what it is doing and do the I/o transfer.

→ Until the I/o transfer is done, it returns to what it was doing

→ Transfer of control through the use of Interrupts is depicted as,



⑤ Discuss Daisy-chain priority Interrupt
page 4.6

⑥ Explain a) Interrupt b) Exception

a) page 4.3

b) Exception:

→ An exception is an abnormal (or) unusual termination

→ An exception is a condition that results from software and prevents the processor from executing the current instruction stream.

→ It affects the normal execution of an instruction.

⑦ What is DMA?
page 4.7

⑧ What is the need for Bus Arbitration?
page 4.8

⑨ Discuss handshaking (or) Asynchronous Data transfer (or) Asynchronous Bus
page 4.12

⑩ Explain PCI bus.
page 4.19

⑪ Explain SCSI bus.
page 4.21

⑫ Explain USB.
page 4.22