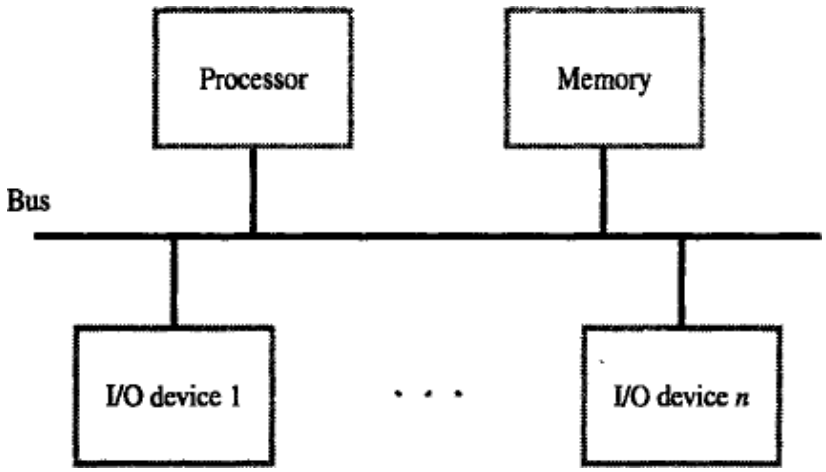
MODULE 1-B INPUT/OUTPUT ORGANIZATION

There are a number of input/output (**I/O**) devices, which can be connected to a computer. The input may be from a keyboard, a sensor, switch, mouse etc. Similarly, output may be a speaker, monitor, printer, a digital display etc.

This variety of I/O devices exchange information in varied format, having different word length, transfer speed is different, but is connected to the same system and exchange information with the same computer. Computer must be capable of handling this wide variety of devices.

ACCESSING I/O-DEVICES

A **single bus-structure** can be used for connecting I/O-devices to a computer. The simple arrangement of connecting set of I/O devices to memory and processor by means of system bus is as shown in the figure. Such an arrangement is called as Single Bus Organization.



- The single bus organization consists of
    - Memory
    - Processor
    - System bus
    - I/O device

- The system bus consists of 3 types of buses:

- Address bus (Unidirectional)
- Data bus (Bidirectional)
- Control bus (Bidirectional)

- The system bus enables all the devices connected to it to involve in the data transfer operation.
- The system bus establishes data communication between I/O device and processor.
- Each I/O device is assigned a unique set of address.
- When processor places an address on address-lines, the intended-device responds to the command.
- The processor requests either a read or write-operation.
- The requested data are transferred over the data-lines

Steps for input operation:

- The address bus of system bus holds the address of the input device.
- The control unit of CPU generates $\overline{\text{IORD}}$ Control signal.
- When this control signal is activated the processor reads the data from the input device (DATAIN) into the CPU register.

Steps for output operation:

- The address bus of system bus holds the address of the output device.
- The control unit of CPU generates $\overline{\text{IOWR}}$ control signal.
- When this control signal is enabled CPU transfers the data from processor register to output device(DATAOUT)

There are 2 schemes available to connect I/O devices to CPU

1. **Memory mapped I/O:**

- In this technique, both memory and I/O devices can share the common memory to store the data, the I/O instructions are mapped to any memory location.
- All memory related instructions are used for data transfer between I/O and processor.
- In case of memory mapped I/O input operation can be implemented as,

MOVE DATAIN , R0

Source destination

This instruction sends the contents of location DATAIN to register R0.

- Similarly output can be implemented as,

MOVE R0, DATAOUT

Source destination

The data is written from R0 to DATAOUT location (address of output buffer)

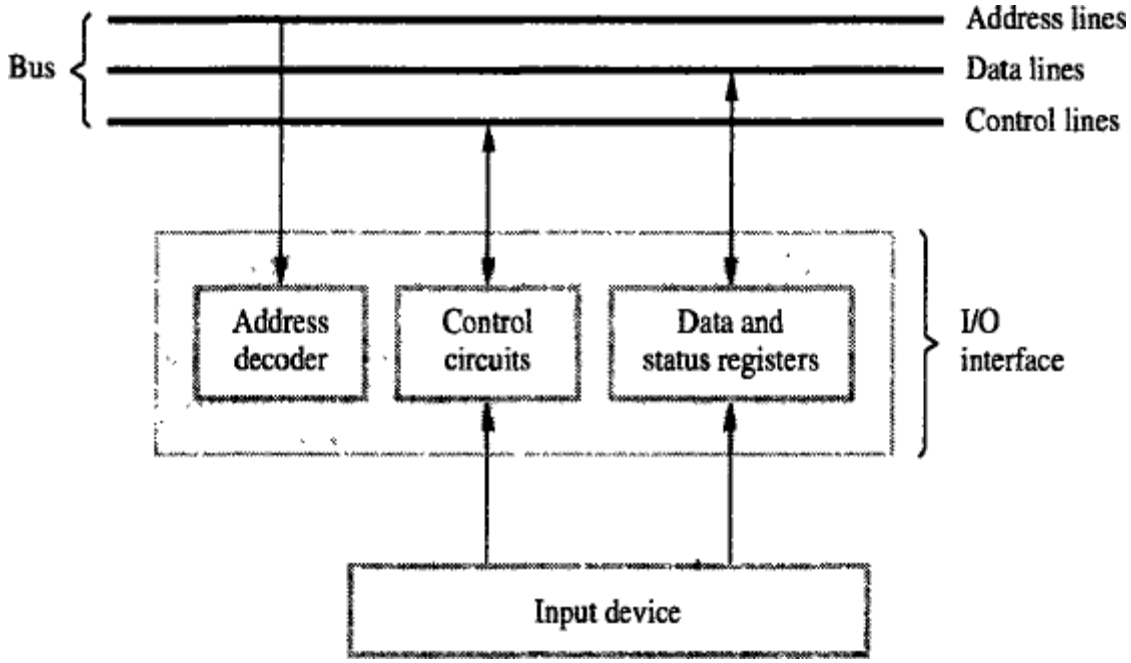
2. I/O Mapped I/O:

- In this technique, a separate address space is allocated for I/O devices. Address space for program and I/O devices are different.
- Hence two sets of instruction are used for data transfer.
- One set for memory operations and another set for I/O operations.
- Whole address space is available for the program.
- Eg – IN AL, DX

Memory Mapped I/O	I/O Mapped I/O
Memory & I/O share the entire address range of processor	Processor provides separate address range for memory & I/O
Processor provides more address lines for accessing memory	Less address lines for accessing I/O
More Decoding is required	Less decoding is required
Memory control signals used to control Read & Write I/O operations	I/O control signals are used to control Read & Write I/O operations

I/O INTERFACE

The hardware arrangement of connecting i/p device to the system bus is as shown in the fig.



This hardware arrangement is called as I/O interface. The I/O interface consists of 3 functional devices namely:

1) Address Decoder:

- Its function is to decode the address, in-order to recognize the input device whose address is available on the unidirectional address bus.
- The recognition of input device is done first, and then the control and data registers becomes active.
- The unidirectional address bus of system bus is connected to input of the address decoder as shown in figure

2) Control Circuit:

- The control bus of system bus is connected to control circuit as shown in the fig.
- The processor sends commands to the I/O system through the control bus.
- It controls the read write operations with respect to I/O device.

3) Status & Data register:

- It specifies type of operation (either read or write operation) to be performed on I/O device. It specifies the position of operation.

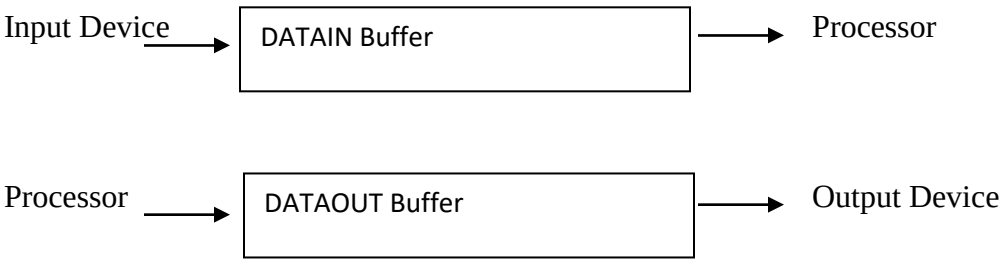
4) Data Register:

- The data bus carries the data from the I/O devices to or from the processor. The data bus is connected to the data/ status register.
- The data register stores the data, read from input device or the data, to be written into output device. There are 2 types:

  DATAIN - Input-buffer associated with keyboard.

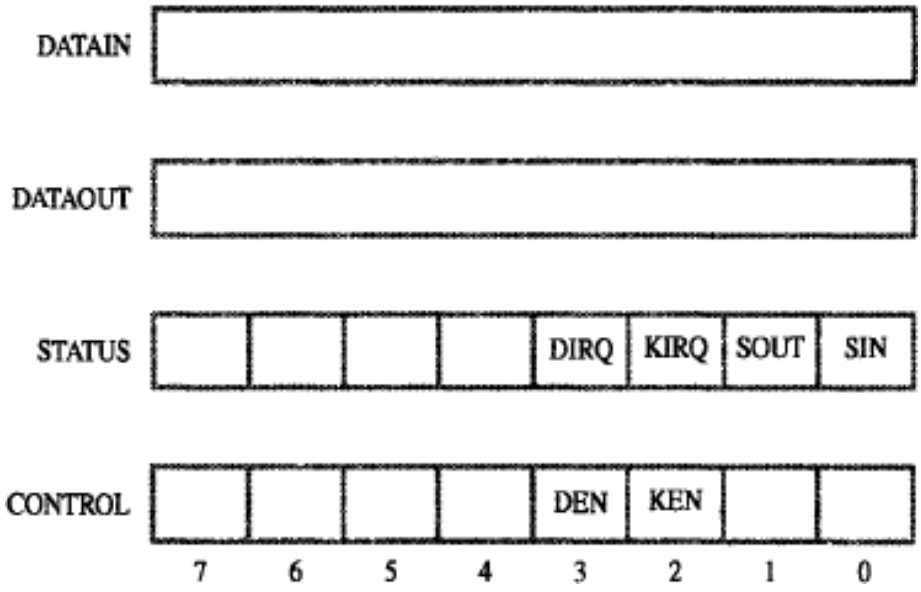
  DATAOUT -Output data buffer of adisplay/printer.

Data buffering is an essential task of an I/O interface. Data transfer rates of processor and memory are high, when compared with the I/O devices, hence the data are buffered at the I/O interface circuit and then forwarded to output device, or forwarded to processor in case of input devices.



Input & Output registers –

Various registers in keyboard and display devices -



DATAIN register is a part of input device. It is used to store the ASCII characters read from keyboard.

DATAOUT register is a part of output device. It is used to store the ASCII characters to be displayed on the output device.

STATUS register stores the status of working of I/O devices –

- SIN flag – This flag is set to 1, when DATAIN buffer contains the data from keyboard. The flag is set to 0, after the data is passed from DATAIN buffer to the processor.
- SOUT flag – This flag is set to 1, when DATAOUT buffer is empty and the data can be added to it by processor. The flag is set to 0, when DATAOUT buffer has the data to be displayed.
- KIRQ (Keyboard Interrupt Request) – By setting this flag to 1, keyboard requests the processor to obtain its service and an interrupt is sent to the processor. It is used along with the SIN flag.
- DIRQ(Display Interrupt Request) – The output device request the processor to obtain its service for output operation, by activating this flag to 1.

Control registers

KEN (keyboard Enable) – Enables the keyboard for input operations.
DEN (Display Enable) – Enables the output device for input operations.

Program Controlled I/O

- It is the process of controlling the input and output operations by executing 2 sets of instruction, one set for input operation and the next set for output operation.
- The program checks the status of I/O register and reads or displays data. Here the I/O operation is controlled by program.

```
WAITK     TestBit #0, STATUS      (Checks SIN flag)
          Branch = 0 WAITK
          Move DATAIN, R0         (Read character)
[*Code to read a character from DATAIN to R0]
```

This code checks the SIN flag, and if it is set to 0 (ie. If no character in DATAIN Buffer), then move back to WAITK label. This loop continues until SIN flag is set to 1. When SIN is 1, data is moved from DATAIN to R0 register. Thus the program, continuously checks for input operation.

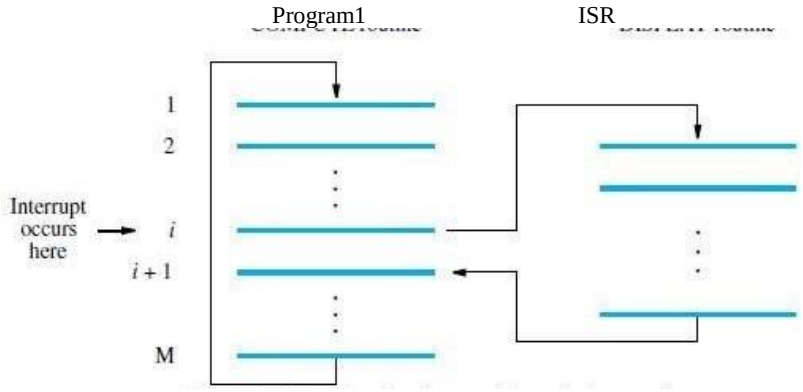
Similarly code for Output operation,

```
WAITD     TestBit #0, STATUS      (Checks SOUT flag)
          Branch = 0 WAITD
          Move R0, DATAOUT        (Send character for display)
```

The code checks the SOUT flag, and if it is set to 1 (ie. If no character in DATAOUT Buffer), then move back to WAITK label. This loop continues until SOUT flag is set to 0. When SOUT is 0, data is moved from R0 register to DATAOUT (ie. Sent by processor).

Interrupt

- It is an event which suspends the execution of one program and begins the execution of another program.
- In program controlled I/O, a program should continuously check whether the I/O device is free. By this continuous checking the processor execution time is wasted. It can be avoided by I/O device **sending an 'interrupt' to the processor, when I/O device is free**.
- The interrupt invokes a subroutine called **Interrupt Service Routine (ISR)**, which resolves the cause of interrupt.
- The occurrence of interrupt causes the processor to transfer the execution control from user program to ISR.



The following steps takes place when the interrupt related instruction is executed:

- After the execution of current instruction i.
- Transfer the execution control to sub program from main program.
- Increments the content of PC by 4 memory location.
- It decrements SP by 4 memory locations.
- Pushes the contents of PC into the stack segment memory whose address is stored in SP.
- It loads PC with the address of the first instruction of the sub program.

The following steps takes place when '*return*' instruction is executed in ISR -

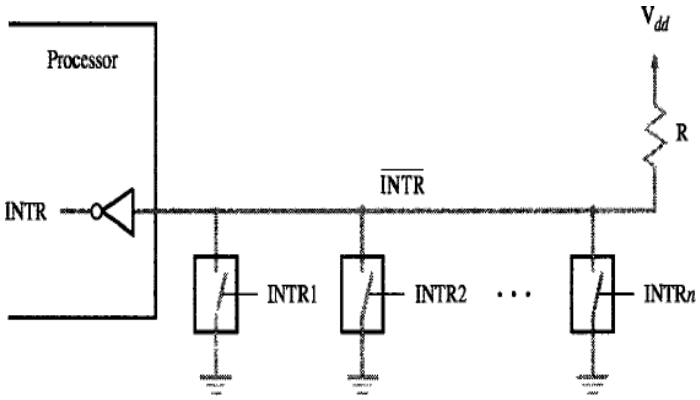
- It transfers the execution control from ISR to user program.
- It retrieves the content of stack memory location whose address is stored in SP into the PC.
- After retrieving the return address from stack memory location into the PC it increments the Content of SP by 4 memory location.

Interrupt Latency / interrupt response time is the delay between the time taken for receiving an interrupt request and start of the execution of the ISR.

Generally, the long interrupt latency in unacceptable.

INTERRUPT HARDWARE

- The external device (I/O device) sends interrupt request to the processor by activating a bus line and called as interrupt request line.
- All I/O device uses the same single interrupt-request line.
- One end of this interrupt request line is connected to input power supply by means of a register.
- The another end of interrupt request line is connected to INTR (Interrupt request) signal of processor as shown in the fig.



- The I/O device is connected to interrupt request line by means of switch, which is grounded as shown in the fig.
- When all the switches are open the voltage drop on interrupt request line is equal to the V_{DD} and INTR value at process is 0.
- This state is called as in-active state of the interrupt request line.

- The I/O device interrupts the processor by closing its switch.
- When switch is closed the voltage drop on the interrupt request line is found to be zero, as the switch is grounded, hence $\overline{INTR}$=0 and INTR=1.
- The signal on the interrupt request line is logical OR of requests from the several I/O devices. Therefore, $\overline{INTR}=\overline{INTR1} + \overline{INTR2} + ... + \overline{INTRn}$

ENABLING AND DISABLING THE INTERRUPTS

The arrival of interrupt request from external devices or from within a process, causes the suspension of on-going execution and start the execution of another program.

- Interrupt arrives at any time and it alters the sequence of execution. Hence the interrupt to be executed must be selected carefully.
- All computers can enable and disable interruptions as desired.
- When an interrupt is under execution, other interrupts should not be invoked. This is performed in a system in different ways.
- The problem of infinite loop occurs due to successive interruptions of active INTR signals.
- There are 3 mechanisms to solve problem of infinite loop:
  1) Processor should ignore the interrupts until execution of first instruction of the ISR.
  2) Processor should automatically disable interrupts before starting the execution of the ISR.
  3) Processor has a special INTR line for which the interrupt-handling circuit.

     Interrupt-circuit responds only to leading edge of signal. Such line is called edge-triggered.
- Sequence of events involved in handling an interrupt-request:
  1) The device raises an interrupt-request.
  2) The processor interrupts the program currently being executed.
  3) Interrupts are disabled by changing the control bits in the processor status register (PS).
  4) The device is informed that its request has been recognized.

     In response, the device deactivates the interrupt-request signal.
  5) The action requested by the interrupt is performed by the interrupt-service routine.
  6) Interrupts are enabled and execution of the interrupted program is resumed.

HANDLING MULTIPLE DEVICES

While handling multiple devices, the issues concerned are:

- How can the processor recognize the device requesting an interrupt?
- How can the processor obtain the starting address of the appropriate ISR?
- Should a device be allowed to interrupt the processor while another interrupt is being serviced?
- How should 2 or more simultaneous interrupt-requests be handled?

VECTORED INTERRUPT

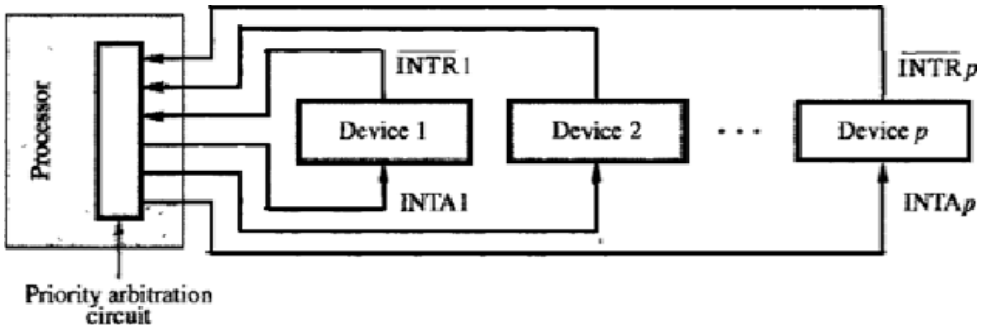
- A device requesting an interrupt identifies itself by sending a special-code to processor over bus.
- Then, the processor starts executing the ISR.
- The special-code indicates starting-address of ISR.
- The special-code length ranges from 4 to 8 bits.
- The location pointed to by the interrupting-device is used to store the staring address to ISR.
- The staring address to ISR is called the **interrupt vector**.
- Processor

  → loads interrupt-vector into PC &

  → executes appropriate ISR.
- When processor is ready to receive interrupt-vector code, it activates INTA line.
- Then, I/O-device responds by sending its interrupt-vector code & turning off the INTR signal.
- The interrupt vector also includes a new value for the Processor Status Register

INTERRUPT NESTING

- A multiple-priority scheme is implemented by using separate INTR & INTA lines for each device
- Each INTR line is assigned a different priority-level as shown in Figure.



- Priority-level of processor is the priority of program that is currently being executed.
- Processor accepts interrupts only from devices that have higher-priority than its own.
- At the time of execution of ISR for some device, priority of processor is raised to that of the device.
- Thus, interrupts from devices at the same level of priority or lower are disabled.

Privileged Instruction

- Processor's priority is encoded in a few bits of PS word. (PS = Processor-Status).
- Encoded-bits can be changed by **Privileged Instructions** that write into PS.
- Privileged-instructions can be executed only while processor is running in **Supervisor Mode**.
- Processor is in supervisor-mode only when executing operating-system routines.

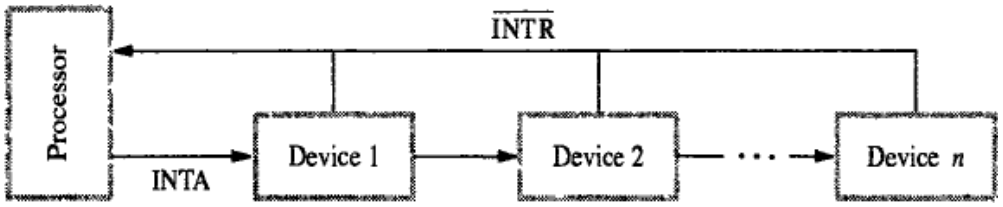
Privileged Exception

- User program cannot
  → accidently or intentionally change the priority of the processor &
  → disrupt the system-operation.
- An attempt to execute a privileged-instruction while in user-mode leads to a **Privileged Exception**.

SIMULTANEOUS REQUESTS

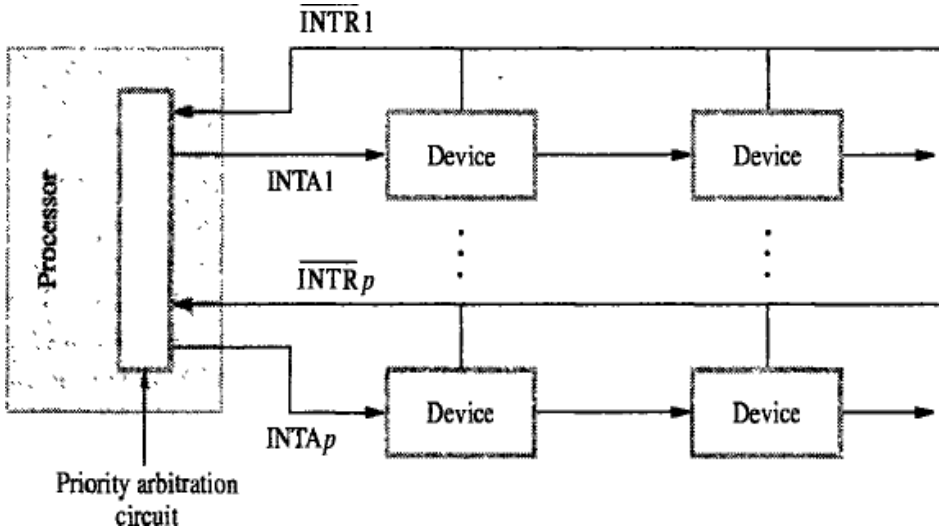
DAISY CHAIN

- The daisy chain with multiple priority levels is as shown in the figure.



- The interrupt request line INTR is common to all devices as shown in the fig.
- The interrupt acknowledge line is connected in a daisy fashion as shown in the figure.
- This signal propagates serially from one device to another device.
- The several devices raise an interrupt by activating INTR signal. In response to the signal, processor transfers its device by activating INTA signal.
- This signal is received by device 1. The device-1 blocks the propagation of INTA signal to device-2, when it needs processor service.
- The device-1 transfers the INTA signal to next device when it does not require the processor service.
- In daisy chain arrangement device-1 has the highest priority.
- **Advantage:** It requires fewer wires than the individual connections.

ARRANGEMENT OF PRIORITY GROUPS



- In this technique, devices are organizes in a group and each group is connected to the processor at a different priority level.

- Within a group device are connected in a daisy chain fashion as shown in the figure.

EXCEPTIONS

- **Exception** is an event that causes an interruption. These are the interrupts caused by the running program. Such interrupts are called exceptions.

- **Types of Exception**

1. Recovery from Errors

- These are techniques to ensure that all hardware components are operating properly.
  For ex: Many computers include an ECC in memory which allows detection of errors in stored-data. (ECC = Error Checking Code, ESR= Exception Service Routine).
- If an error occurs, control-hardware
  → detects the errors &
  → informs processor by raising an interrupt.
- When exception processing is initiated (as a result of errors), processor -
  → suspends program being executed &
  → starts an ESR. This routine takes appropriate action to recover from the error.
  Does the same steps as interrupt handling.

2. Debugging

Debugger is used to find errors in a program and uses exceptions to provide 2 important facilities:

i) Trace & ii) Breakpoints

i) Trace

- When a processor is operating in trace-mode, an exception occurs after execution of every instruction (using debugging-program as ESR).
- Debugging-program enables user to examine contents of registers, memory-locations and so on.
- On return from debugging-program, next instruction in program being debugged is executed, then debugging-program is activated again.
- The trace exception is disabled during the execution of the debugging-program.

ii) Breakpoints

- Here, the program being debugged is interrupted only at specific points selected by user.
- An instruction called Trap (or Software interrupt) is usually provided for this purpose.
- When program is executed & reaches breakpoint, the user can examine memory & register contents.

3. Privilege Exception

- To protect OS from being corrupted by user-programs, **Privileged Instructions** are executed only while processor is in supervisor-mode.
- For example, When processor runs in user-mode, it will not execute instruction that

change priority of processor.

- An attempt to execute privileged-instruction will produce a **Privilege Exception**.
- As a result, processor switches to supervisor-mode & begins to execute an appropriate routine in OS.

Direct Memory Access (DMA)

- Direct Memory Access is the process of transferring the block of data at high speed in between main memory and external device (I/O devices) without continuous intervention of CPU.
- This operation is performed by the control circuit, called as DMA controller.
- DMA controller is a part of the I/O interface.
- The data transfer operation in DMA is processed by the help of DMA controller.
- To initiate Directed data transfer between main memory and external devices DMA controller needs parameters from the CPU.
- These 3 Parameters are:

1) Starting address of the memory block.
2) No of words to be transferred.
3) Type of operation (Read or Write).

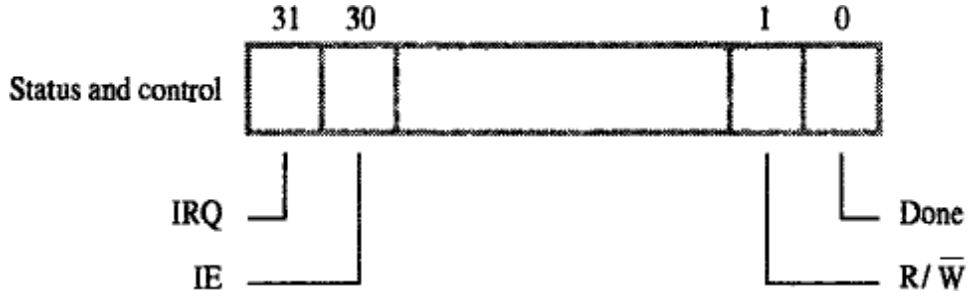
After receiving these 3 parameters from CPU, DMA controller establishes directed data transfer operation between main memory and external devices without the involvement of CPU. Hence the processor is free to execute other programs.

Register of DMA Controller:
It consists of 3 type of register:

Starting address register:
The format of starting address register is as shown in the fig. It is used to store the starting address of the memory block.

Word-Count register:
The format of word count register is as shown in fig. It is used to store the no of words to be transferred from main memory to external devices and vice versa.

Status and Controller register:
The format of status and controller register is as shown in fig.



a)DONE bit:

- The DMA controller sets this bit to 1 when it completes the direct data transfer between main memory and external devices.
- This information is informed to CPU by means of DONE bit.

b)R/$\overline{W}$ (Read or Write):

- This bit is used to differentiate between memory read and memory write operation. It is set by a program instruction.
- The R/$\overline{W}$ = 1 for read operation and
  = 0 for write operation.
- When this bit is set to 1, DMA controller performs read operation and transfers one block of data from main memory to external device.
- When this bit is set to 0, DMA controller performs write operation and transfers one block of data from external device to main memory.

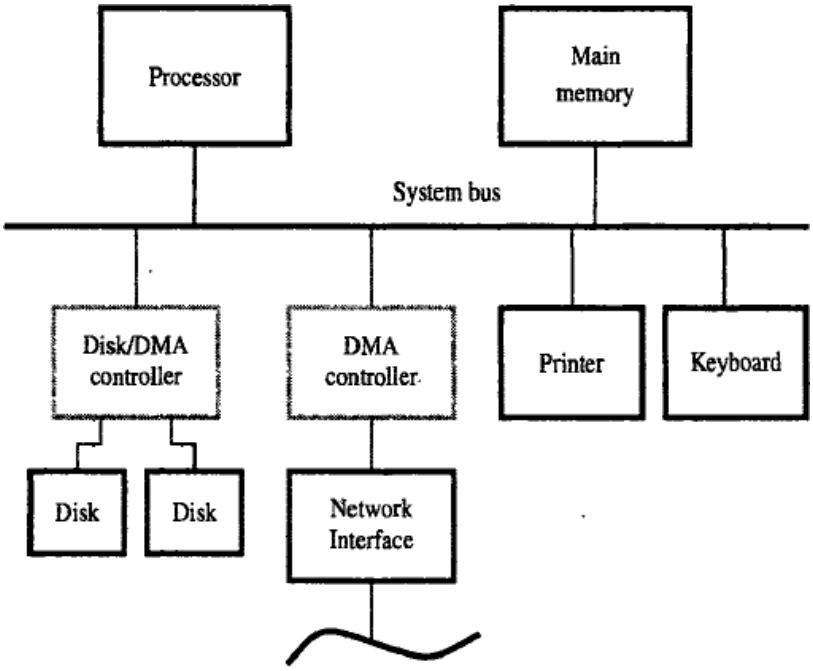
c) IE (Interrupt enable) bit:

- The DMA controller enables the interrupt enable bit after the completion of DMA operation

d)**Interrupt request (IRQ):**

- The DMA controller requests the CPU for permission and data, to transfer new block of data from source to destination by activating this bit.

The computer with DMA controller is as shown in the fig.:



- In the sample architecture shown above, the DMA controller connects two external devices namely disk 1 and disk 2 to system bus.
- The DMA controller also interconnects high speed network devices to system bus as shown in the above fig.
- Let us consider direct data transfer operation by means of DMA controller without the involvement of CPU in between main memory and disk 1.
- To establish direct data transfer operation between main memory and disk 1. DMA controller request the processor to obtain 3 parameters namely:
  1) Starting address of the memory block.
  2) No of words to be transferred.
  3) Type of operation (Read or Write).

- After receiving these 3 parameters from processor, DMA controller directly transfers block of data between main memory and external devices (disk 1) depending on the operation.
- This information is informed to CPU by setting respective bits in the status and controller register of DMA controller.
  These are 2 types of request with respect to system bus
  1). CPU request.
  2). DMA request.
  Highest priority will be given to DMA request.

- Actually the CPU generates memory cycles to perform read and write operations.
  The DMA controller steals memory cycles from the CPU to perform read and write operations. This approach is called as **"Cycle stealing".**

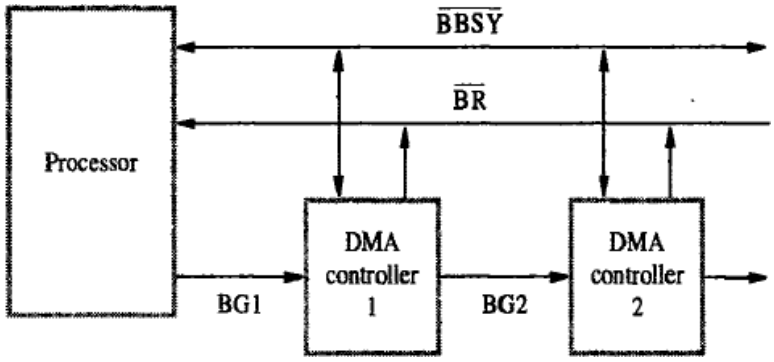
- An exclusive option will be given for DMA controller to transfer block of data between external devices and main memory. Only after the transfer of whole block, signal is sent to the processor. This technique is called as "**Burst mode of operation."**
- Conflict may arise, if CPU and multiple DMA controllers, request for bus, at the same time. This is resolved by bus arbitration.

BUS ARBITRATION

- Any device which initiates data transfer operation on bus at any instant of time is called as Bus-Master.
- When the bus mastership is transferred from one device to another device, the next device is ready to obtain the bus mastership.
- The bus-mastership is transferred from one device to another device based on the principle of priority system. There are two types of bus-arbitration technique:

a)Centralized bus arbitration:

In this technique CPU acts as a bus-master or any control unit connected to bus can be acts as a bus master.



The schematic diagram of centralized bus arbitration is as shown in the fig.:

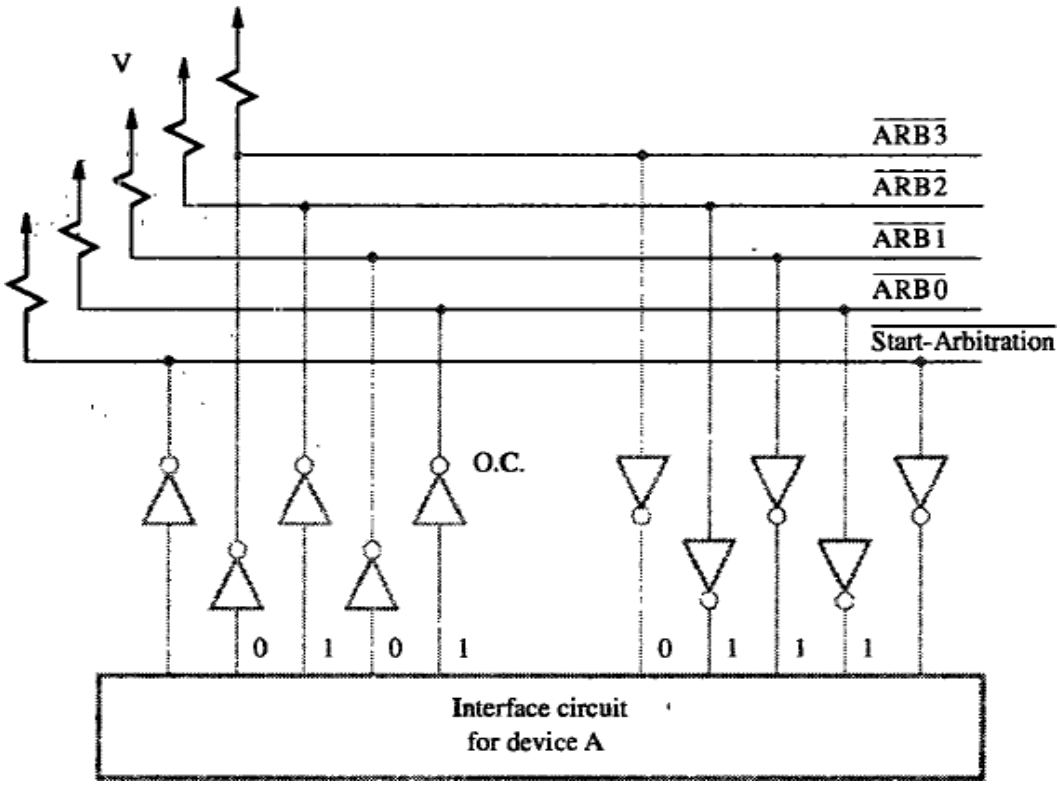
The following steps are necessary to transfer the bus mastership from CPU to one of the DMA controller:

- The DMA controller request the processor to obtain the bus mastership by activating $\overline{BR}$ (Bus request) signal
- In response to this signal the CPU transfers the bus mastership to requested devices DMA controller1 in the form of BG (Bus grant).
- When the bus mastership is obtained from CPU the DMA controller1 blocks the propagation of bus grant signal from one device to another device.
- The $\overline{BG}$ signal is connected to DMA controller2 from DMA controller1, and so on as in daisy fashion style as shown in the figure.
- When the DMA controller1 has not sent BR request, it transfers the bus mastership to DMA controller2 by unblocking bus grant signal.

- When the DMA controller1 receives the bus grant signal, it blocks the signal from passing to DMA controller2 and enables BBSY signal. When BBSY signal is set to 1 the set of devices connected to system bus doesn't have any rights to obtain the bus mastership from the CPU.

b)Distributed bus arbitration:

- In this technique 2 or more devices trying to access system bus at the same time may participate in bus arbitration process.
- The schematic diagram of distributed bus arbitration is as shown in the figure:



- The external device requests the processor to obtain bus mastership by enabling start arbitration signal.
- In this technique 4 bit code is assigned to each device to request the CPU in order to obtain bus mastership.
- Two or more devices request the bus by placing 4 bit code over the system bus.
- The signals on the bus interpret the 4 bit code and produces winner as a result from the CPU.
- When the input to the one driver = 1, and input to the another driver = 0, on the same bus line, this state is called as "Low level voltage state of bus".
- Consider 2 devices namely A & B trying to access bus mastership at the same time.
  Let assigned code for devices A & B are 5 (0101) & 6 (0110) respectively.
- The device A sends the pattern (0101) and device B sends its pattern (0110) to master. The signals on the system bus interpret the 4 bit code for devices A & B produces device B as a winner.
- The device B can obtain the bus mastership to initiate direct data transfer between external devices and main memory.

BUSES

- The primary function of the bus is to inter connect 3 functional device namely CPU, memory and I/O devices.
- It is defined as set of similar wires used to establish data transfer operation between CPU and memory as well as CPU and I/O devices.
- It consists of 3 types:
  a) Uni-directional address line.
  b) Bi-directional data lines.
  c) Control lines.
- The address bus of system bus is used to carry either the address of I/O device or the address of memory.
- The bi-directional data bus is used to carry data to be returned into I/O device or read from I/O device.
- The control bus of system bus is used to carry control signals as well as timing information. It is designed to carry control signals as R/W.

  The R/$\overline{W}$ = 1 for read operation.
  = 0 for write operation.

  The control bus also specifies the timing information to indicate when the processor and I/O devices may place data or receive data from the bus.

- Based on this timing of placing the data to the bus, the bus can be categorized into two types –
  - Synchronous bus
  - Asynchronous bus

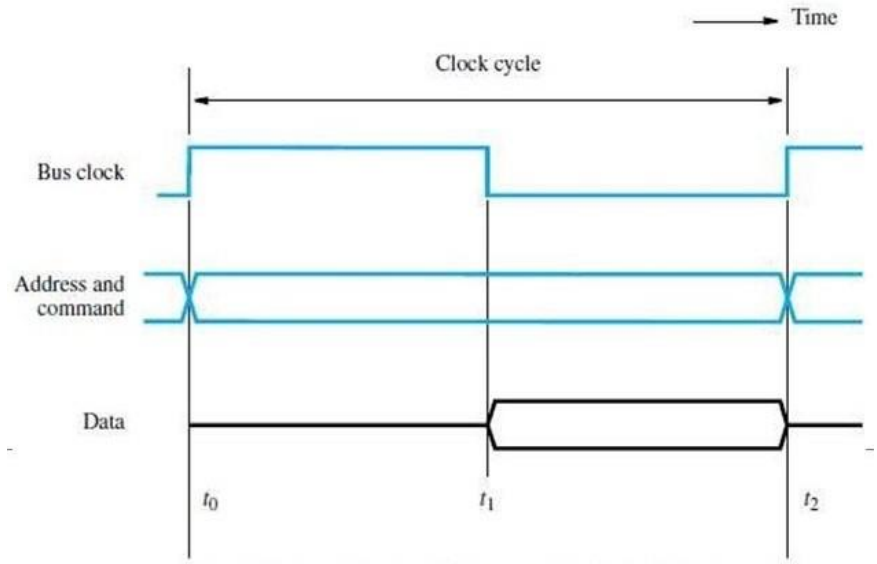
Note : The bus master is the device which has the control over the bus. It initiates the data transfer by issuing read or write signal. It is also called as initiator. The device addressed by the master is called as a slave or target.

Synchronous bus

- In case of Synchronous bus all the devices derive the timing information from common bus line.
- An equally placed pulses on this common bus line are called as timing intervals or timing signals.
- Two or more timing intervals in which single data transfer takes place is called as 'bus cycle'.

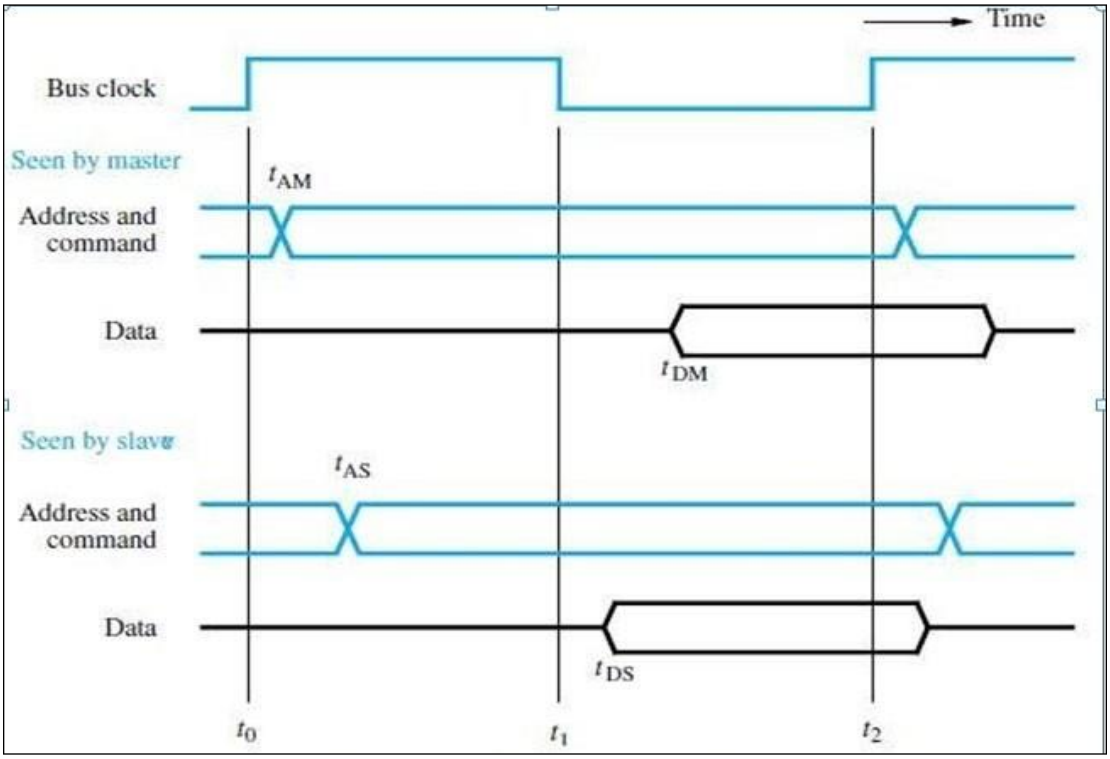
Timing Diagram for the Read-operation that shows a sequence of events during a read-operation

- At time t_0, the master (processor)
  → places the device-address on address-lines &
  → sends an appropriate command on control-lines as shown in Figure.



- The command will
  → indicate an input operation &
  → specify the length of the operand to be read.
- Information travels over bus at a speed determined by physical & electrical characteristics.
- Clock pulse width(t1-t0) must be longer than max. propagation-delay b/w devices connected to bus.
- The clock pulse width should be long to allow the devices to decode the address & control signals.
- The slaves take no action or place any data on the bus before t1.
- Information on bus is unreliable during the period t0 to t1 because signals are changing state.
- Slave places requested input-data on data-lines at time t1.
- At end of clock cycle (at time t2), master strobes (captures) data on data-lines into its input-buffer
- For data to be loaded correctly into a storage device, data must be available at input of that device for a period greater than setup-time of device

A Detailed Timing Diagram for the Read-operation

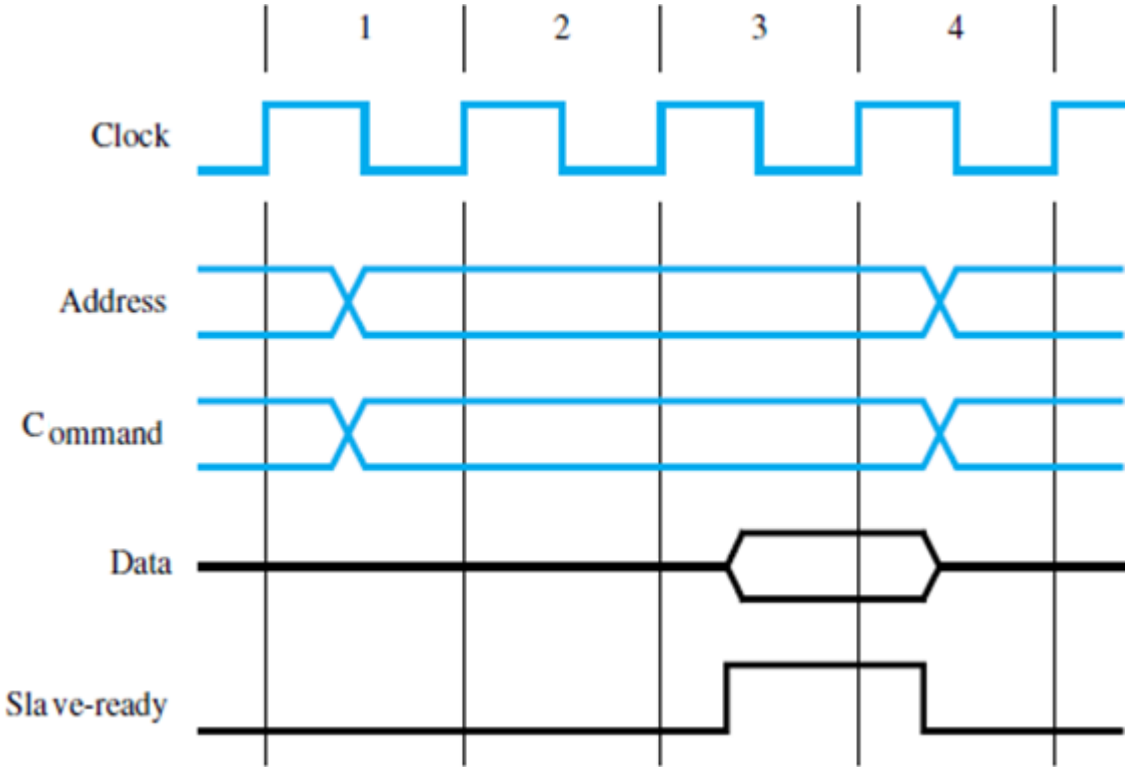


- The Figure shows two views of the signal.
- One view shows the signal seen by the master & the other is seen by the salve.
- Master sends the address & command signals on the rising edge at the beginning of clock period (t0).
- These signals do not actually appear on the bus until tam.
- Sometimes later, at t_{AS} the signals reach the slave.
- The slave decodes the address.
- At t1, the slave sends the requested-data.
- At t2, the master loads the data into its input-buffer.
- Hence the period t2, t_{DM} is the setup time for the masters input-buffer.
- The data must be continued to be valid after t2, for a period equal to the hold time of that buffers.

Disadvantages

- The device does not respond.
- The error will not be detected.

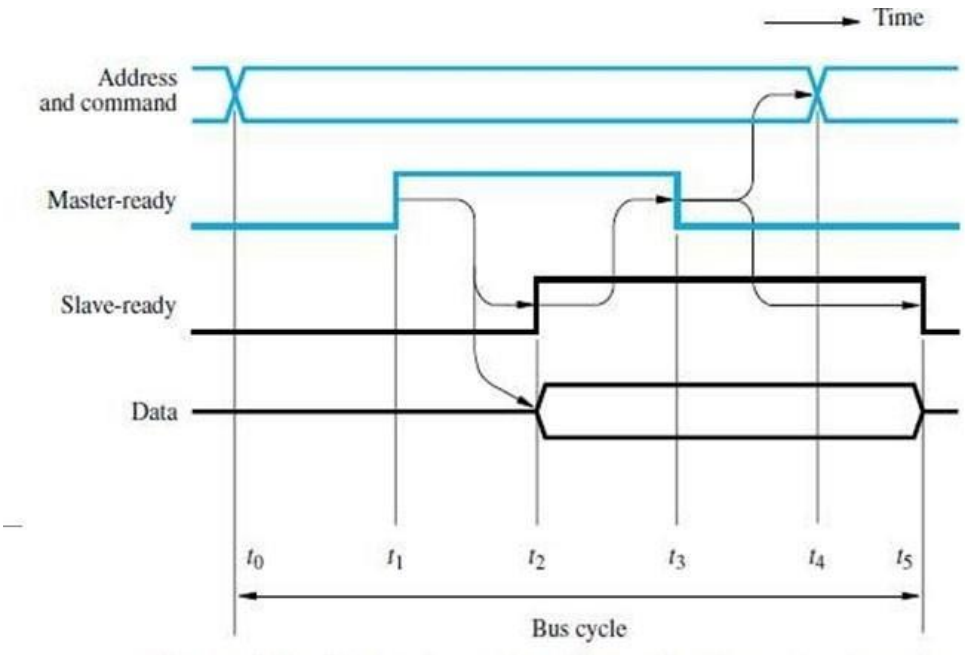
Multiple Cycle Transfer for Read-operation



- During, clock cycle-1, master sends address/command info the bus requesting a "read" operation.
- The slave receives & decodes address/command information.
- At the active edge of the clock i.e. the beginning of clock cycle-2, it makes access to respond immediately.
- The data become ready & are placed in the bus at clock cycle-3.
- At the same times, the slave asserts a control signal called **slave-ready**.
- The master strobes the data to its input-buffer at the end of clock cycle-3.
- The bus transfer operation is now complete.
- And the master sends a new address to start a new transfer in clock cycle4.
- The slave-ready signal is an acknowledgement from the slave to the master.

ASYNCHRONOUS BUS

- This method uses handshake-signals between master and slave for coordinating data-transfers.
- There are 2 control-lines:
  1) **Master-Ready (MR)** is used to indicate that master is ready for a transaction.
  2) **Slave-Ready (SR)** is used to indicate that slave is ready for a transaction.



The Read Operation proceeds as follows:

- At t0, master places address/command information on bus.
- At t1, master sets MR-signal to 1 to inform all devices that the address/command-info is ready.
  - MR-signal =1 causes all devices on the bus to decode the address.
  - The delay t1 – t0 is intended to allow for any skew that may occurs on the bus.
  - Skew occurs when 2 signals transmitted from 1 source arrive at destination at different time
  - Therefore, the delay t1 – t0 should be larger than the maximum possible bus skew.
- At t2, slave

  → performs required input-operation &

  → sets SR signal to 1 to inform all devices that it is ready (Figure 7.6).
- At t3, SR signal arrives at master indicating that the input-data are available on bus.
- At t4, master removes address/command information from bus.
- At t5, when the device-interface receives the 1-to-0 transition of MR signal, it removes data and SR signal from the bus. This completes the input transfer.

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 bus signals.
- On the other side, we have a data path with its associated controls to transfer data between the interface and the I/O device known as **port**.

• Two types are:

1. **Parallel Port** transmits and receives data in the form of a number of bits (8 or 16) simultaneously to or from the device.
2. **Serial Port** transmits and receives data one bit at a time.

• Communication with in 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.

• In parallel-port, the connection between the device and the computer uses

→ a multiple-pin connector and

→ a cable with as many wires.

• This arrangement is suitable for devices that are physically close to the computer.

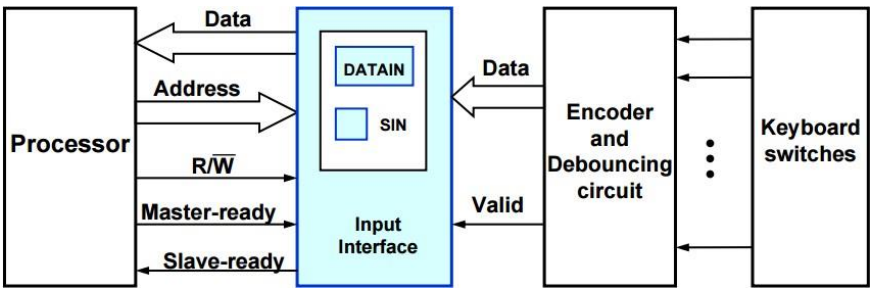
• In serial port, it is much more convenient and cost-effective where longer cables are needed.

Functions of I/O Interface

1) Provides a storage buffer for at least one word of data.
2) Contains status-flags that can be accessed by the processor to determine whether the buffer is full or empty.
3) Contains address-decoding circuitry to determine when it is being addressed by the processor.
4) Generates the appropriate timing signals required by the bus control scheme.
5) Performs any format conversion that may be necessary to transfer data between the bus and the I/O device (such as parallel-serial conversion in the case of a serial port).

Parallel Port

The hardware components needed for connecting a keyboard to a processor.

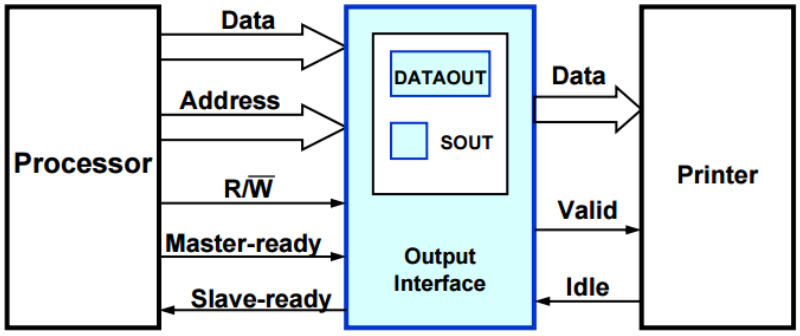


• Keyboard is connected to a processor using a parallel-port.

- Processor uses
  → memory-mapped I/O and
  → asynchronous bus protocol.
- On the processor-side of the interface, we have:
  → Data-lines
  → Address-lines
  → Control or R/W line
  → Master-Ready signal and
  → Slave-Ready signal.
- The output of the encoder consists of
  → bits representing the encoded character and
  → one signal called **valid**, which indicates the key is pressed.

  The information is sent to the interface-circuits
- Interface-circuits contain
  1) Data register DATAIN &
  2) Status-flag SIN.
- When a key is pressed, corresponding signals is sent to the encoder circuit and the circuit generates the ASCII code for the corresponding key. The Valid signal changes from 0 to1.
  Then, SIN=1 , when ASCII code is loaded into DATAIN.
  SIN = 0 , when processor reads the contents of the DATAIN.
- The interface-circuit is connected to the asynchronous bus.
- Data transfers on the bus are controlled using the handshake signals:
  1) Master ready &
  2) Slave ready.

<u>The hardware components needed for connecting a Printer to a processor.</u>



- Processor uses
  → memory-mapped I/O and
  → asynchronous bus protocol.
- On the processor-side of the interface, we have:

→ Data-lines
→ Address-lines
→ Control or R/W line
→ Master-Ready signal and
→ Slave-Ready signal.

The circuit of output interface,

– Slave-ready
– R/W
– Master-ready
– Address decoder
– Handshake control

- Interface-circuits contain
  1) Data register DATAOUT &
  2) Status-flag SOUT.

  SOUT=1 , when device is ready to accept another character, ie. DATAOUT buffer is empty.
  SOUT = 0 , when character is loaded in DATAOUT buffer.

The input and output interfaces can be combined into a single interface.

Serial Port

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

STANDARD I/O INTERFACES

Consider a computer system using different interface standards. The three major standard I/O interfaces discussed here are:

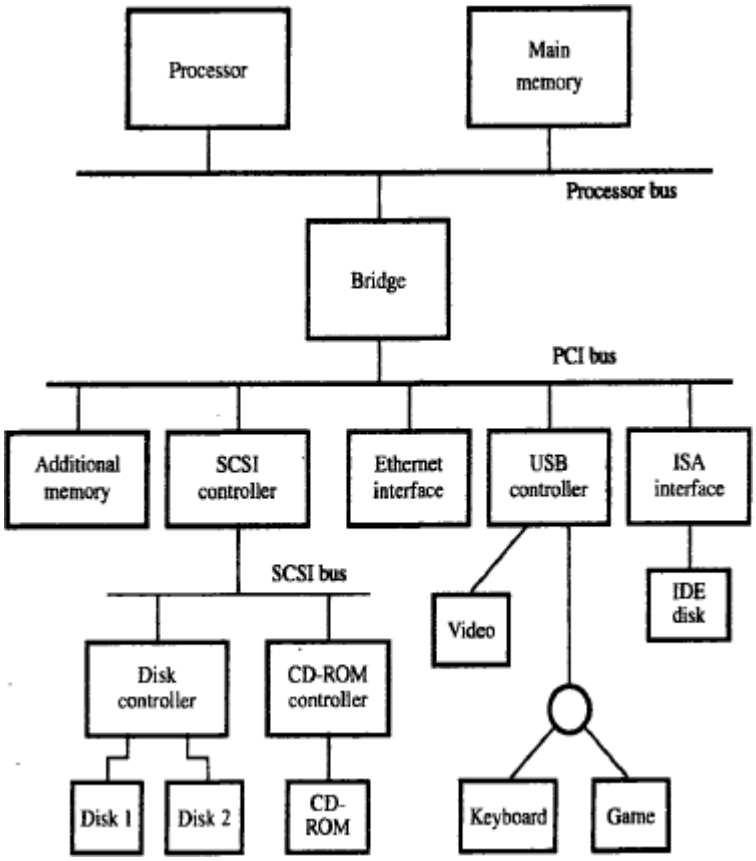
– PCI (Peripheral Component Interconnect)
– SCSI (Small Computer System Interface)
– USB (Universal Serial Bus)

- PCI defines an expansion bus on the motherboard.
- SCSI and USB are used for connecting additional devices both inside and outside the computer-box.

PCI (Peripheral Component Interconnect)

- PCI is developed as a low cost bus that is truly processor independent.

- PCI supports high speed disk, graphics and video devices.
- PCI has plug and play capability for connecting I/O devices.
- To connect new devices, the user simply connects the device interface board to the bus.
- Processor bus and Peripheral Component Interconnect (PCI) bus are interconnected by a circuit called **Bridge**.
- The bridge translates the signals and protocols of one bus into another.
- The bridge-circuit introduces a small delay in data transfer between processor and the devices.

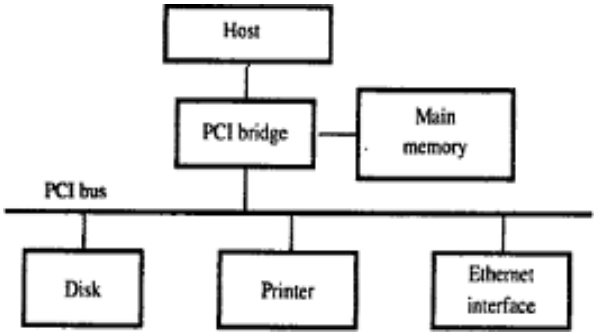


DATA TRANSFER IN PCI

- During **read-operation**,
  - When the processor specifies an address, the memory responds by sending a sequence of data-words from successive memory-locations.
- During **write-operation**,
  - When the processor sends an address, a sequence of data-words is written into successive memory-locations.

- PCI supports read and write-operation.

- A read/write-operation involving a single word is treated as a burst of length one.
- PCI has 3 address-spaces. They are
    1) Memory address-space
    2) I/O address-space &
    3) Configuration address-space.
- I/O Address-space is intended for use with processor.
- Configuration space is intended to give PCI, its plug and play capability.
- **PCI Bridge** provides a separate physical connection to main-memory.

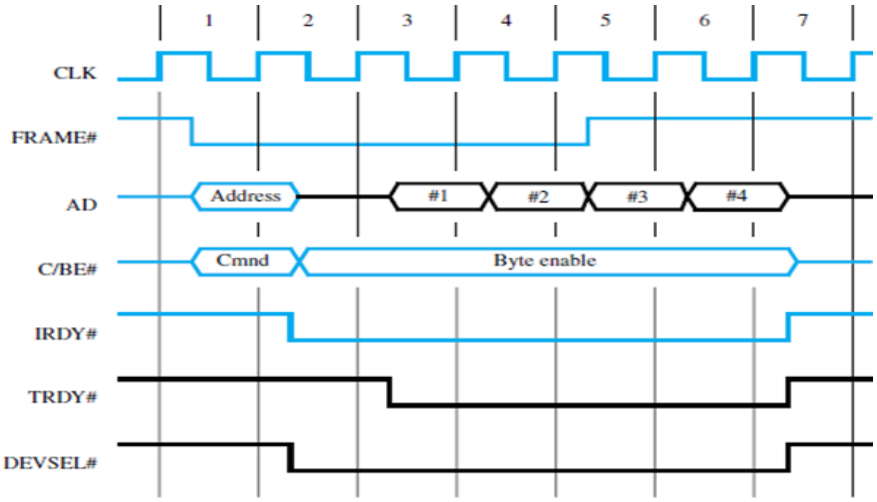


- The master maintains the address information on the bus until data-transfer is completed.
- At any time, only one device acts as **Bus-Master**.
- A master is called "initiator" which is either processor or DMA.
- The addressed-device that responds to read and write commands is called a **Target.**
- A complete transfer operation on the bus, involving an address and burst of data is called a transaction.
- Individual word transfers are called "**phases'**.
- Data transfer signals on PCI bus

Data transfer signals on the PCI bus.

Name	Function
CLK	A 33-MHz or 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transmission
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 command/byte-enable lines (8 for a 64-bit bus)
IRDY#, TRDY#	Initiator-ready and Target-ready signals
DEVSEL#	A response from the device indicating that it has recognized its address and is ready for a data transfer transaction
IDSEL#	Initialization Device Select

Read Operation on PCI Bus



- During Clock cycle-1,
  - ➢ The processor
    - → asserts FRAME# to indicate the beginning of a transaction;
    - → sends the address on AD lines and command on C/BE# Lines.
- During Clock cycle-2,
  - ➢ The processor removes the address and disconnects its drives from AD lines.
  - ➢ Selected target
    - → enables its drivers on AD lines and
    - → fetches the requested-data to be placed on bus.
  - ➢ Selected target
    - → asserts DEVSEL# and
    - → maintains it in asserted state until the end of the transaction.
  - ➢ C/BE# is
    - → used to send a bus command and it is
    - → used for different purpose during the rest of the transaction.
- During Clock cycle-3,
  - ➢ The initiator asserts IRDY# to indicate that it is ready to receive data.
  - ➢ If the target has data ready to send then it asserts TRDY#. In our eg, the target sends 3 more words of data in clock cycle 4 to 6.
- During Clock cycle-5
  - ➢ The indicator uses FRAME# to indicate the duration of the burst, since it read 4 words, the initiator negates FRAME# during clock cycle 5.
- During Clock cycle-7,
  - ➢ After sending 4^{th} word, the target
    - → disconnects its drivers and
    - → negates DEVSEL# during clock cycle 7.

DEVICE CONFIGURATION OF PCI

- The PCI has a configuration ROM that stores information about that device.
- The configuration ROM's of all devices are accessible in the configuration address-space.
- The initialization software read these ROM's whenever the system is powered up or reset.
- In each case, it determines whether the device is a printer, keyboard or disk controller.
- Devices are assigned address during initialization process.
- Each device has an input signal called IDSEL# (Initialization device select) which has 21 address- lines (AD11 to AD31).
- During configuration operation,
  - The address is applied to AD input of the device and
  - The corresponding AD line is set to 1 and all other lines are set to 0. AD11 - AD31 ,**Upper** address-line
    A0 - A10 ,**Lower** address-line: Specify the type of the operation and to access the content of device configuration ROM.
- The configuration software scans all 21 locations. PCI bus has interrupt-request lines.
- Each device may requests an address in the I/O space or memory space

SCSI Bus

- SCSI stands for Small Computer System Interface.
- SCSI refers to the standard bus which is defined by ANSI (American National Standard Institute).
- SCSI bus the several options. It may be,

Narrow bus	It has 8 data-lines & transfers 1 byte at a time.
Wide bus	It has 16 data-lines & transfer 2 byte at a time.
Single-Ended Transmission	Each signal uses separate wire.
HVD (High Voltage Differential)	It was 5v (TTL cells)
LVD (Low Voltage Differential)	It uses 3.3v

- Because of these various options, SCSI connector may have 50, 68 or 80 pins.
- The data transfer rate ranges from 5MB/s to 160MB/s 320Mb/s, 640MB/s. The transfer rate depends on,
  1) Length of the cable
  2) Number of devices connected.
- To achieve high transfer rate, the bus length should be 1.6m for SE signaling and 12m for LVD signaling.
- The SCSI bus us connected to the processor-bus through the SCSI controller.
- The data are stored on a disk in blocks called sectors. Each sector contains several hundreds of

bytes. These data will not be stored in contiguous memory-location.

- SCSI protocol is designed to retrieve the data in the first sector or any other selected sectors.
- Using SCSI protocol, the burst of data are transferred at high speed.
- The controller connected to SCSI bus is of 2 types.

They are1) Initiator

2) Target

1) Initiator

- It has the ability to select a particular target & to send commands specifying the operation to be performed.
- They are the controllers on the processor side.

2) Target

- The disk controller operates as a target.
- It carries out the commands it receive from the initiator.
- The initiator establishes a logical connection with the intended target.

Steps for Read-operation.

The processor sends a command to the SCSI controller, which causes the following sequence of events to take place:

1) The SCSI controller contents for control of the bus (initiator).
2) When the initiator wins the arbitration-process, the initiator
   → selects the target controller and
   → hands over control of the bus to it.
3) The target starts an output operation. The initiator sends a command specifying the required read- operation.
4) The target
   → sends a message to initiator indicating that it will temporarily suspend connection b/w them.
   → then releases the bus.
5) The target controller sends a command to the disk drive to move the read head to the first sector involved in the requested read-operation.
6. The target
   → transfers the contents of the data buffer to the initiator and
   → then suspends the connection again.
7) The target controller sends a command to the disk drive to perform another seek operation.
8) As the initiator controller receives the data, it stores them into the main-memory using the DMA approach.
9) The SCSI controller sends an interrupt to the processor indicating that the data are now available.

BUS SIGNALS OF SCSI

Table 4.4 The SCSI bus signals

Category	Name	Function
Data	–DB(0) to –DB(7)	Data lines: Carry one byte of information during the information transfer phase and identify device during arbitration, selection and reselection phases
	–DB(P)	Parity bit for the data bus
Phase	–BSY	Busy: Asserted when the bus is not free
	–SEL	Selection: Asserted during selection and reselection
Information type	–C/D	Control/Data: Asserted during transfer of control information (command, status or message)
	–MSG	Message: indicates that the information being transferred is a message
Handshake	–REQ	Request: Asserted by a target to request a data transfer cycle
	–ACK	Acknowledge: Asserted by the initiator when it has completed a data transfer operation
Direction of transfer	–I/O	Input/Output: Asserted to indicate an input operation (relative to the initiator)
Other	–ATN	Attention: Asserted by an initiator when it wishes to send a message to a target
	–RST	Reset: Causes all device controls to disconnect from the bus and assume their start-up state

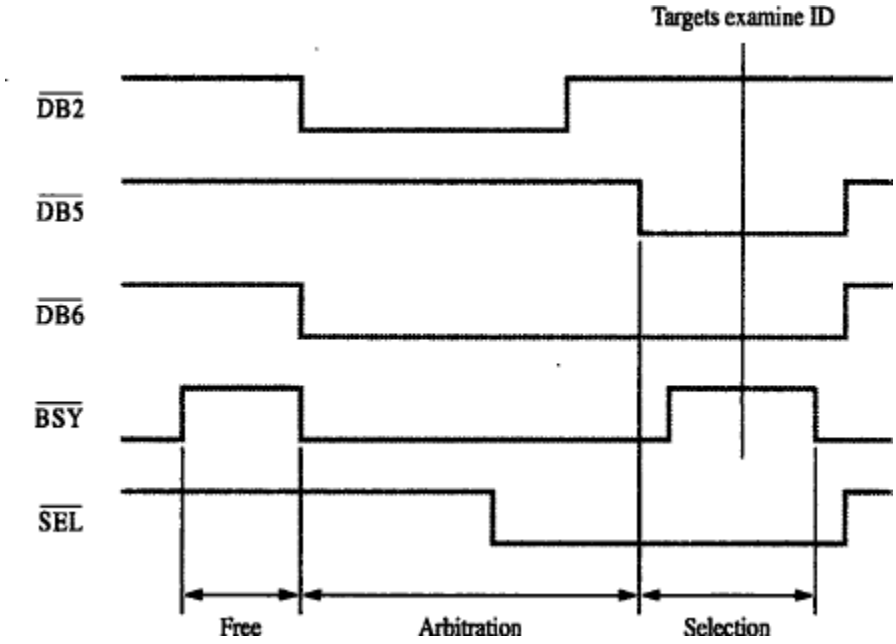
PHASES IN SCSI BUS

- The phases in SCSI bus operation are:
  1) Arbitration
  2) Selection
  3) Information transfer
  4) Reselection

1) Arbitration

- When the –BSY signal is in inactive state,
  → the bus will be free &
  → any controller can request the use of bus.
- SCSI uses distributed arbitration scheme because each controller may generate requests at the same time.
- Each controller on the bus is assigned a fixed priority.
- When BSY becomes active, all controllers that are requesting the bus
  → examines the data-lines &
  → determine whether highest priority device is requesting bus at the same time.

- The controller using the highest numbered line realizes that it has won the arbitration-process.
- At that time, all other controllers disconnect from the bus & wait for –BSY to become inactive again.



2) Information Transfer

- The information transferred between two controllers may consist of
  - → commands from the initiator to the target
  - → status responses from the target to the initiator or
  - → data-transferred to/from the I/0 device.
- Handshake signaling is used to control information transfers, with the target controller taking the role of the bus-master.

3) Selection

- Here, Device
  - → wins arbitration and
  - → asserts –BSY and –DB6 signals.
- The Select Target Controller responds by asserting –BSY.
- This informs that the connection that it requested is established.

4) Reselection

- The connection between the two controllers has been reestablished, with the target in control of the bus as required for data transfer to proceed.

Universal Serial Bus (USB)

- USB stands for Universal Serial Bus.
- USB supports 3 speed of operation. They are,
  1) Low speed (1.5 Mbps)
  2) Full speed (12 mbps) &
  3) High speed (480 mbps).
- The USB has been designed to meet the key objectives. They are,
  1) Provide a simple, low-cost and easy to use interconnection system.
     This overcomes difficulties due to the limited number of I/O ports available on a computer.
  2) Accommodate a wide range of data transfer characteristics for I/O devices. For e.g. telephone and Internet connections
  3) Enhance user convenience through a "plug-and-play" mode of operation.

- **Advantage:** USB helps to add many devices to a computer system at any time without opening the computer-box.

Port Limitation

- Normally, the system has a few limited ports.
- To add new ports, the user must open the computer-box to gain access to the internal expansion bus & install a new interface card.
- The user may also need to know to configure the device & the s/w.

Plug & Play

- The main objective: USB provides a plug & play capability.
- The plug & play feature enhances the connection of new device at any time, while the system is operation.
- The system should
  → Detect the existence of the new device automatically.
  → Identify the appropriate device driver s/w.
  → Establish the appropriate addresses.
  → Establish the logical connection for communication.

DEVICE CHARACTERISTICS OF USB

- The kinds of devices that may be connected to a computer cover a wide range of functionality.
- The speed, volume & timing constrains associated with data transfer to & from devices varies significantly.

Eg: 1 Keyboard

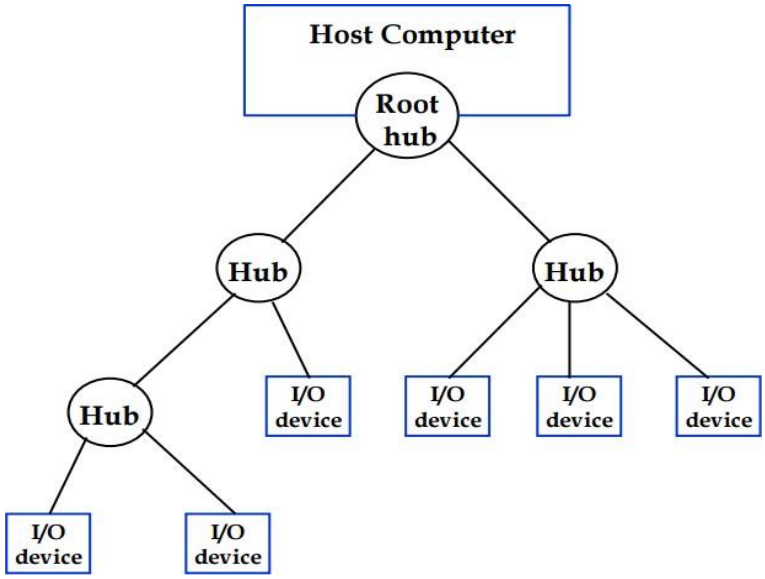
- Since the event of pressing a key is not synchronized to any other event in a computer system, the data generated by keyboard are called asynchronous.
  The data generated from keyboard depends upon the speed of the human operator which is about 100 bytes/sec.

Eg: 2 Microphone attached in a computer system internally/externally

- The sound picked up by the microphone produces an analog electric signal, which must be converted into digital form before it can be handled by the computer.
- This is accomplished by sampling the analog signal periodically.
- The sampling process yields a continuous stream of digitized samples that arrive at regular intervals, synchronized with the sampling clock. Such a stream is called isochronous (i.e.) successive events are separated by equal period of time.
- If the sampling rate in "S" samples/sec then the maximum frequency captured by sampling process is S/2.
- A standard rate for digital sound is 44.1 KHz.

USB architecture

- A serial transmission format has been chosen for the USB because a serial bus satisfies the low-cost and flexibility requirements
- Clock and data information are encoded together and transmitted as a single signal ‹ Hence, there are no limitations on clock frequency or distance, arising from data skew
- 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 device. At the root of the tree, a root hub connects the entire tree to the host computer
- To accommodate a large number of devices that can be added or removed at any time, the USB has the tree structure. Each node has a device called a hub. Root hub, functions, split bus operations – high speed (HS) and Full/Low speed
- Tree structure of a USB is shown fig below



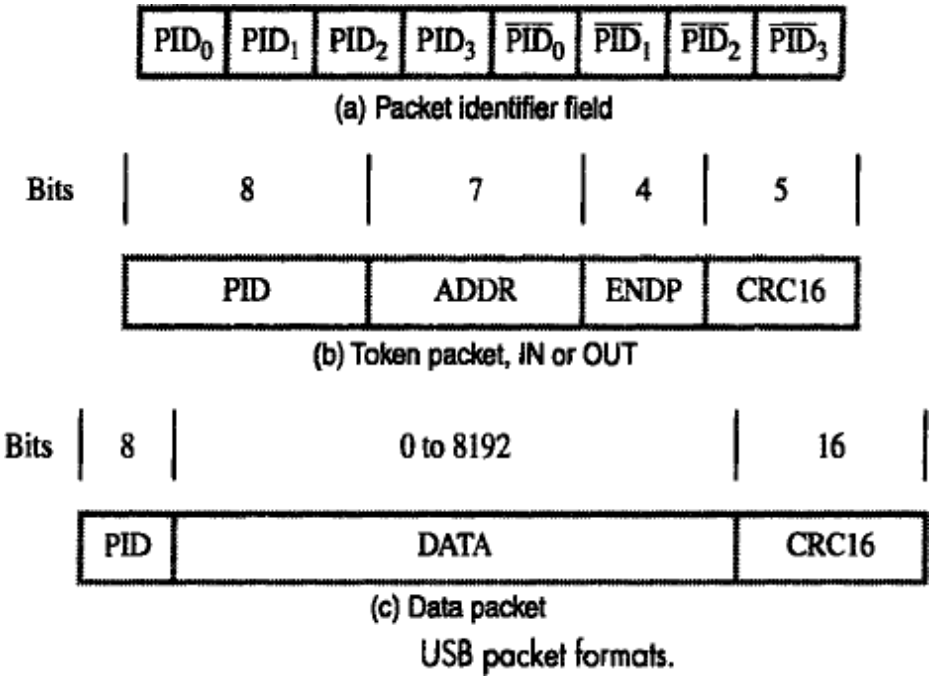
USB ADDRESSING

- Each device may be a hub or an I/O device.
- Each device on the USB is assigned a 7-bit address.
- This address
  - → is local to the USB tree and
  - → is not related in any way to the addresses used on the processor-bus.
- A hub may have any number of devices or other hubs connected to it, and addresses are assigned arbitrarily.
- When a device is first connected to a hub, or when it is powered-on, it has the address 0.
- The hardware of the hub detects the device that has been connected, and it records this fact as part of its own status information.
- Periodically, the host polls each hub to
  - → collect status information and
  - → learn about new devices that may have been added or disconnected.
- When the host is informed that a new device has been connected, it uses sequence of commands to
  - → send a reset signal on the corresponding hub port.
  - → read information from the device about its capabilities.
  - → send configuration information to the device, and
  - → assign the device a unique USB address.
- Once this sequence is completed, the device
  - → begins normal operation and
  - → responds only to the new address.

USB PROTOCOL

- All information transferred over the USB is organized in packets.
- A packet consists of one or more bytes of information.
- There are many types of packets that perform a variety of control functions.
- The information transferred on USB is divided into 2 broad categories: 1) Control and 2) Data.
- Control packets perform tasks such as
  - → addressing a device to initiate data transfer.
  - → acknowledging that data have been received correctly or
  - → indicating an error.

- Data-packets carry information that is delivered to a device.
- A packet consists of one or more fields containing different kinds of information.
- The first field of any packet is called the **Packet Identifier (PID)** which identifies type of that packet.
- They are transmitted twice.
    1) The first time they are sent with their true values and
    2) The second time with each bit complemented.
- The four PID bits identify one of 16 different packet types.
- Some control packets, such as ACK (Acknowledge), consist only of the PID byte.
- Control packets used for controlling data transfer operations are called **Token Packets.**



USB packet formats.