SEM – IV - COA - Phase – T4

UNIT - 10 - Input-Output Organization (9%)

1. Peripheral Devices:

- Input or output devices attached to the computer are also called peripherals.
- Among the most common peripherals are keyboards, display units, and printers.
- Peripherals that provide auxiliary storage for the system are magnetic disks and tapes.
- Peripherals are electromechanical and electromagnetic devices.
- **Magnetic disks** have high-speed rotational surfaces coated with magnetic material. Access is achieved by moving a read-write mechanism to a track in the magnetized surface.
- **Magnetic tapes** are used mostly for storing files of data. It is one of the cheapest and slowest methods for storage and has the advantage that tapes can be removed when not in use.

2. Input and Output Interface:

- Input-output interface provides a method for transferring information between internal storage and external I/O devices.
- The purpose of the interface is to resolve the differences that exist between the central computer and each peripheral.

The major differences are :-

- Peripherals are electromechanical and electromagnetic devices & CPU and memory are electronic devices so a conversion of signal values may be required.
- The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, so a synchronization mechanism is required.

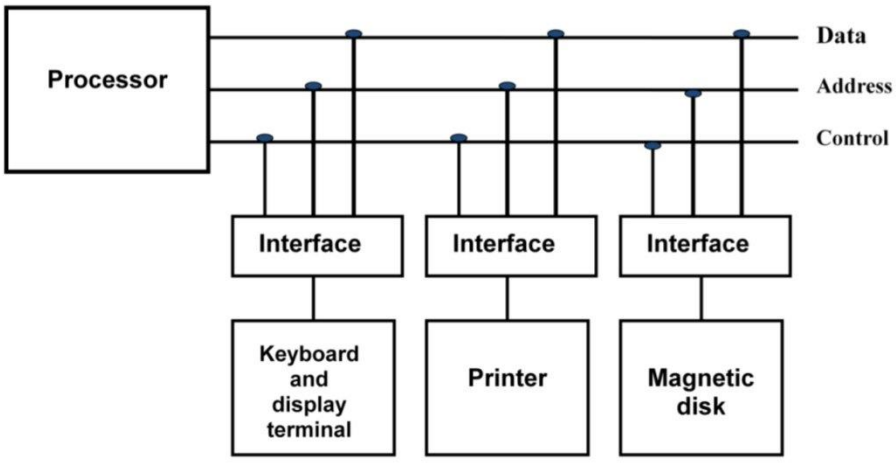
- Data codes and formats in peripherals differ from the word format in the CPU and memory.
- The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

What is interface?

- To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers.
- These components are called **interface** units because they interface between the processor bus and the peripheral device.

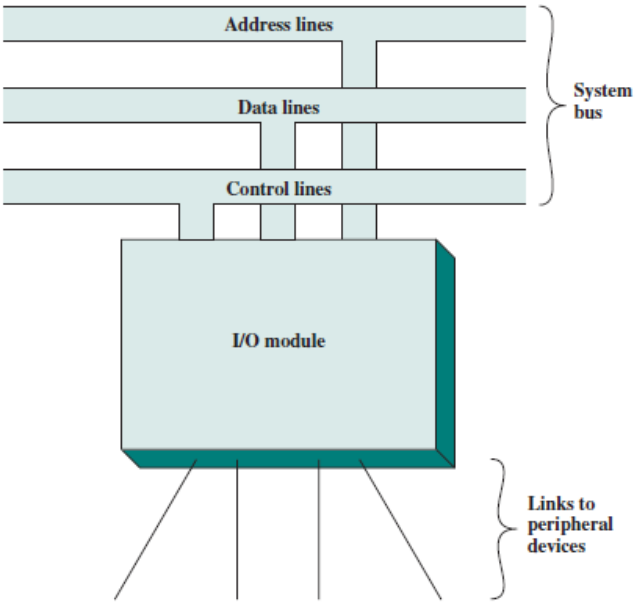
3. Connection of I/O bus to input-output devices:

- A typical communication link between the processor and several peripherals is shown in Fig.

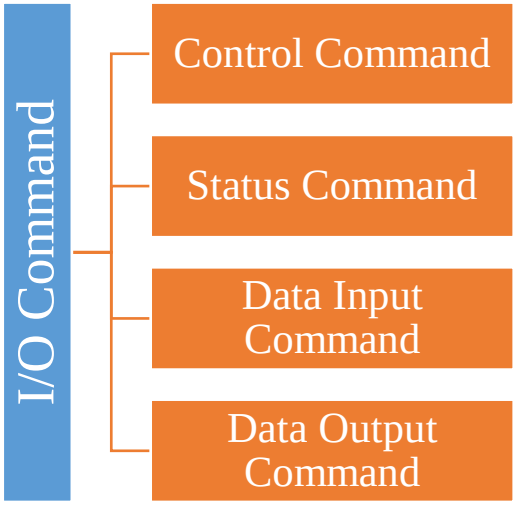


Connection of I/O bus to input-output devices

- The I/O bus consists of data lines, address lines, and control lines.



- The magnetic tape is used in some computers for backup storage.
- Each peripheral device has associated with it an interface unit.
- Each interface decodes the address and control received from the I/O bus.
- The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address lines.
- Each interface attached to the I/O bus contains an address decoder that monitors the address lines.
- When the interface detects its own address, it activates the path between the bus lines and the device that it controls.
- There are four types of commands that an interface may receive. They are classified as control, status, data output, and data input.



- A control command is issued to activate the peripheral and to inform it what to do. For example, a magnetic tape unit may be instructed to backspace the tape by one record, to rewind the tape, or to start the tape moving in the forward direction.
- A status command is used to test various status conditions in the interface and the peripheral. For example, the computer may wish to check the status of the peripheral before a transfer is initiated. During the transfer, one or more errors may occur which are detected by the interface. These errors are designated by setting bits in a status register that the processor can read at certain intervals.
- A data output command causes the interface to respond by transferring data from the bus into one of its registers.

Consider an example with a tape unit.

- The computer starts the tape moving by issuing a control command. The processor then monitors the status of the tape by means of a status command. When the tape is in the correct position, the processor issues a data output command. The interface responds to the address and command and transfers the information from the data lines in the bus to its buffer

register. The interface then communicates with the tape controller and sends the data to be stored on tape.

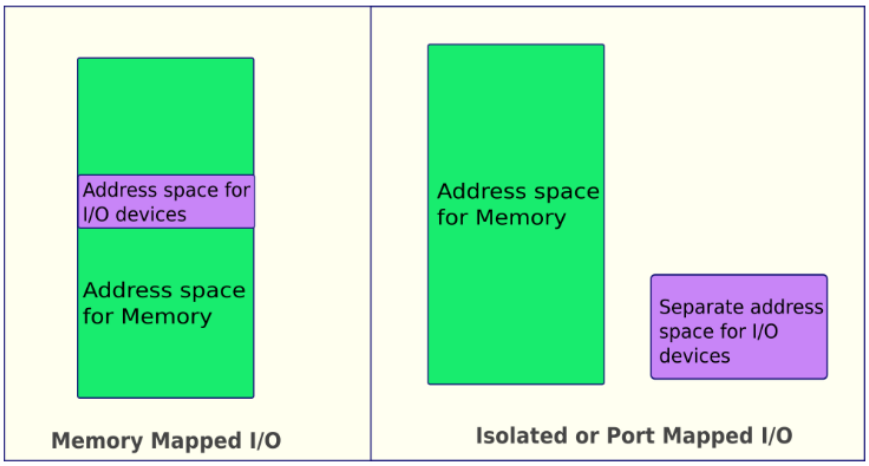
- The data input command is the opposite of the data output. In this case the interface receives an item of data from the peripheral and places it in its buffer register. The processor checks if data are available by means of a status command and then issues a data input command. The interface places the data on the data lines, where they are accepted by the processor.
- There are three ways that computer buses can be used to communicate with memory and I/O:

  1. Use two separate buses, one for memory and the other for I/O.

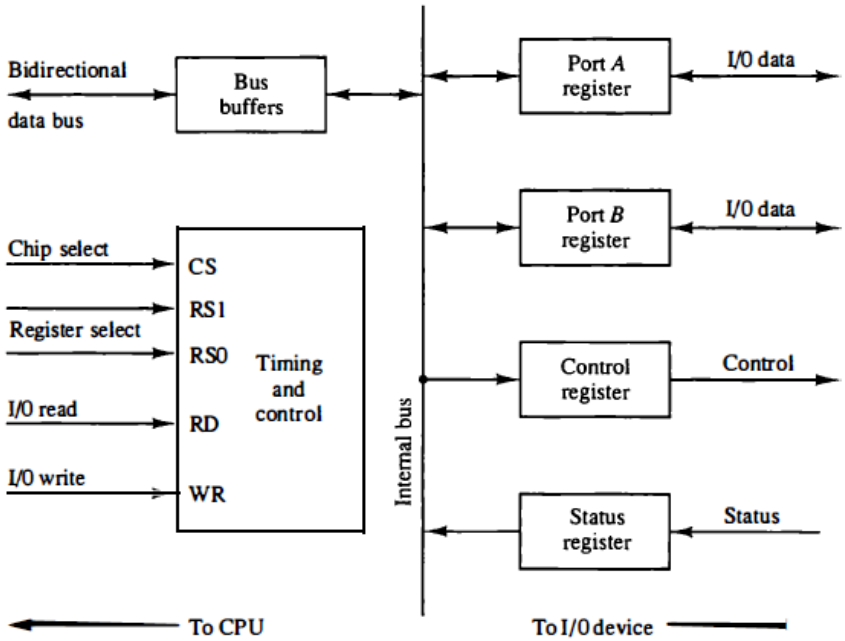
  2. Use one common bus for both memory and I/O but have separate control lines for each.

  3. Use one common bus for memory and I/O with common control lines

Isolated I/O	Memory-mapped I/O
Different address spaces are used for computer memory and I/O devices. I/O devices have dedicated address space.	Same address space is used for memory and I/O devices.
Separate control unit and control instructions are used in case of I/O devices.	Control units and instructions are same for memory and I/O devices.
More complex and costlier than memory -mapped I/O as more bus are used.	Easier to build and cheap as it's less complex.
Entire address space can be used by memory as I/O devices have separate address space.	Some part of the address space of computer memory is consumed by I/O devices as address space is shared.
Computer memory and I/O devices use different control instructions for read write.	Computer memory and I/O devices can both use same set of read and write instructions.
Separate control bus is used for computer memory and I/O devices. Though same address and data bus are used.	Address, data and control bus are same for memory and I/O devices.



4. Example of I/O Interface Unit:



- It consists of two data registers called ports, a control register, a status register, bus buffers, and timing and control circuits.
- The interface communicates with the CPU through the data bus. The chip select and register select inputs determine the address assigned to the interface.
- The I/O read and write are two control lines that specify an input or output, respectively. The four registers communicate directly with the I/O device attached to the interface.

- The I/O data to and from the device can be transferred into either port A or port B.

CS	RS1	RS0	Register selected
0	×	×	None: data bus in high-impedance
1	0	0	Port *A* register
1	0	1	Port *B* register
1	1	0	Control register
1	1	1	Status register

- This circuit enables the chip select (CS) input when the interface is selected by the address bus.
- The two register select inputs RSl and RS0 are usually connected to the two least significant lines of the address bus.
- These two inputs select one of the four registers in the interface as specified in the table.
- The content of the selected register is transfer into the CPU via the data bus when the I/O read signal is enabled. The CPU transfers binary information into the selected register via the data bus when the I/O write input is enabled.

5. Asynchronous Data Transfer:

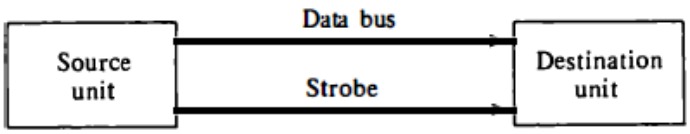
- The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator.
- Two units, such as a CPU and an I/O interface, are designed independently of each other. If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be synchronous.

- In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be asynchronous to each other.
- Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.
- Asynchronous data transfer methods: 1. Strobe Pulse method
  2. Handshaking
- One way of achieving this is by means of a strobe pulse supplied by one of the units to indicate to the other unit when the transfer has to occur.
- Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus. The unit receiving the data item responds with another control signal to acknowledge receipt of the data.
- This type of agreement between two independent units is referred to as handshaking.

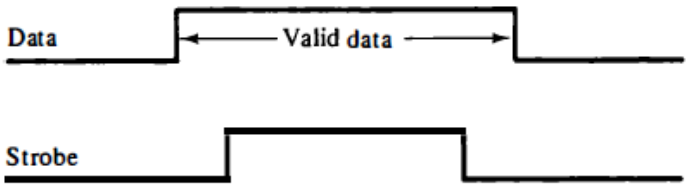
6. Strobe Pulse Method / Strobe Control Method:

- The strobe control method of asynchronous data transfer employs a single control line to time each transfer.
- The strobe may be activated by either the source or the destination unit.
- Strobe Pulse method can be applied in two ways:
  1. Source Initiated Strobe for data transfer
  2. Destination Initiated Strobe for data transfer

Source Initiated Strobe for data transfer:



(a) Block diagram



(b) Timing diagram

- The data bus carries the binary information from source unit to the destination unit.
- The strobe is a single line that informs the destination unit when a valid data word is available in the bus.
- As shown in timing diagram:

Source unit first places the data on the data bus

↓

After a brief delay to ensure that the data settle to a steady value, the source activates the strobe pulse

↓

The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data

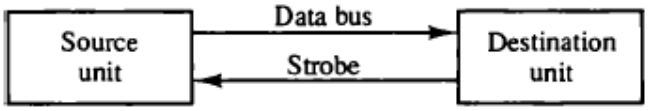
↓

The source removes the data from the bus a brief period after it disables its strobe pulse

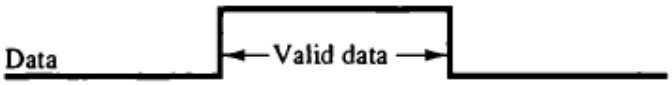
↓

The fact that the strobe signal is disabled indicates that the data bus does not contain valid data. New valid data will be available only after the strobe is enabled again

Destination Initiated Strobe for data transfer:



(a) Block diagram



(b) Timing diagram

Destination unit activates the strobe pulse, informing the source to provide the data.

The source unit responds by placing the requested binary information on the data bus

The destination unit accepts the data and then disables the strobe.

The source removes the data from the bus after a predetermined time interval

Disadvantage of Strobe Pulse method for data transfer:

- The disadvantage of the strobe method is that the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus.

- The handshake method solves this problem by introducing a second control signal that provides a reply to the unit that initiates the transfer.

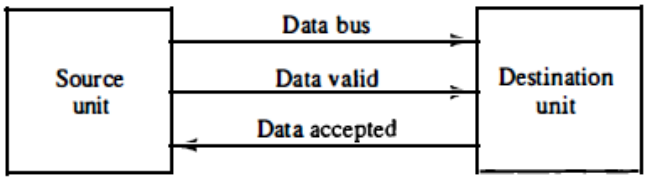
7. Handshaking Method:

- The basic principle of the handshaking method of data transfer is as follows: >>> One control line is in the same direction as the data flow in the bus from the source to the destination. It is used by the source unit to inform the destination unit whether there are valid data in the bus.

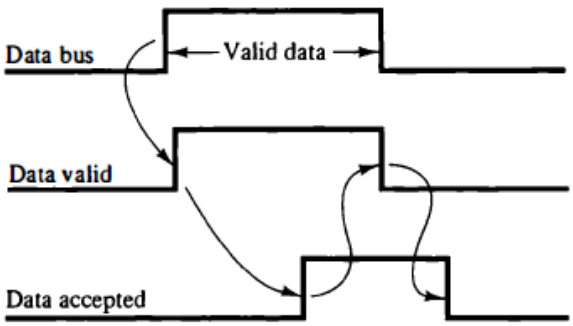
  >>> The other control line is in the other direction from the destination to the source. It is used by the destination unit to inform the source whether it can accept data.

Sequence of events occurred in Source initiated handshaking:

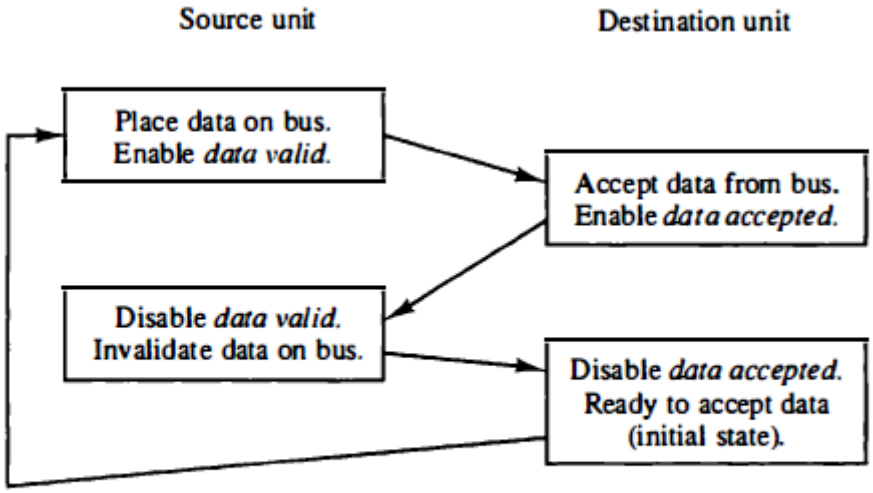
- The two handshaking lines are *data valid*, which is generated by the source unit, and *data accepted*, generated by the destination unit. The timing diagram shows the exchange of signals between the two units.



(a) Block diagram

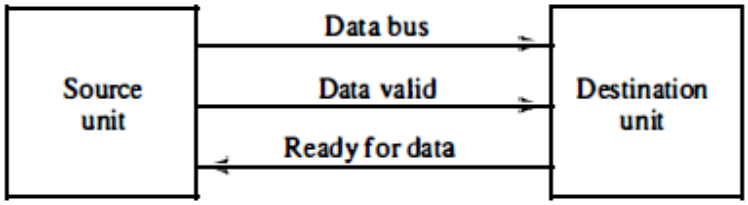


(b) Timing diagram

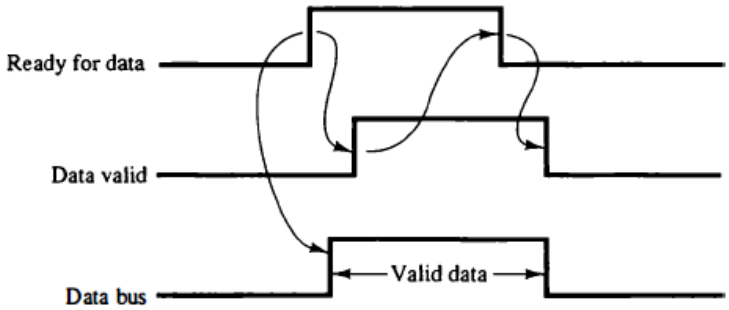


(c) Sequence of events

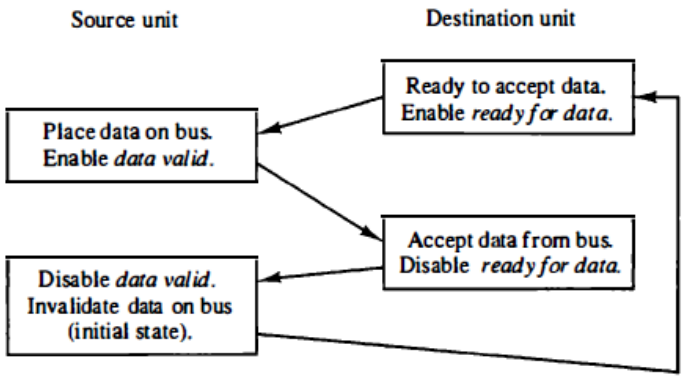
Sequence of events occurred in Destination initiated handshaking:



(a) Block diagram



(b) Timing diagram



(c) Sequence of events

- The handshaking scheme provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units. If one unit is faulty, the data transfer will not be completed.

8. Modes of Transfer:

- Data transfer to and from peripherals may be handled in one of three possible modes:

1. Programmed I/O

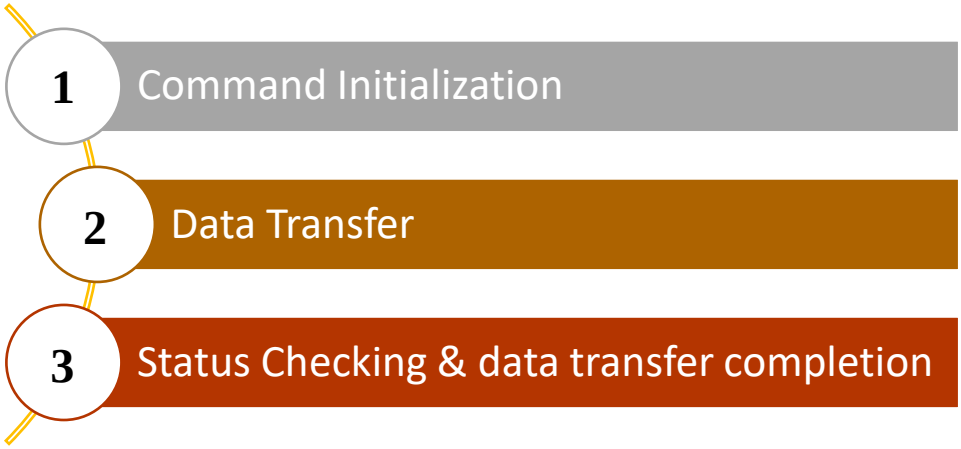
2. Interrupt-initiated I/O

3. Direct memory access (DMA)

Programmed I/O

- Programmed I/O (Input/Output) refers to a method of data transfer between a computer's central processing unit (CPU) and its peripherals.
- In programmed I/O, the CPU directly controls the data transfer process by issuing commands and monitoring the status of the I/O operations.

Here's how programmed I/O generally works:



- **Command Initialization:** The CPU sends a command to the I/O device to initiate an operation, such as reading data from a storage device or sending data through a communication port.
- **Data Transfer:** The CPU then transfers data between its own registers and the I/O device's registers. This involves moving data from memory to an I/O register (for output) or from an I/O register to memory (for input).
- **Status Checking:** The CPU periodically checks the status of the I/O device to determine whether the operation has been completed. This involves reading status flags or registers associated with the I/O device.
- **Data Transfer Completion:** Once the I/O device indicates that the operation is complete (through a status flag or register), the CPU can continue its execution or initiate further I/O operations.

Limitations and Drawbacks of Programmed I/O

CPU Involvement: Programmed I/O is a time-consuming process since it keeps the processor busy needlessly

No Parallelism: Since the CPU is directly involved in managing I/O operations, it can't perform other tasks during I/O transfers. This limits the overall system's parallelism and efficiency.

Wasted CPU Cycles: In cases where the I/O operation completes quickly, the CPU spends unnecessary cycles repeatedly checking the status, leading to wasted processing power.

To overcome the drawbacks of Programmed I/O:-

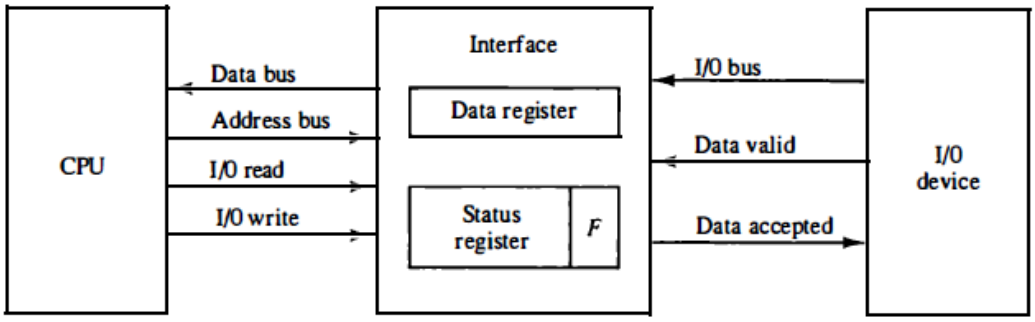
- To overcome the limitations of programmed I/O, modern computer systems often use interrupt-driven I/O or direct memory access (DMA) techniques.
- In interrupt-driven I/O, the CPU is notified by the I/O device when an operation is complete, allowing the CPU to focus on other tasks until it's needed again.

- In DMA, a separate DMA controller takes over the data transfer process between memory and I/O devices, freeing the CPU from direct involvement and allowing for more efficient data transfers

Example of Programmed I/O

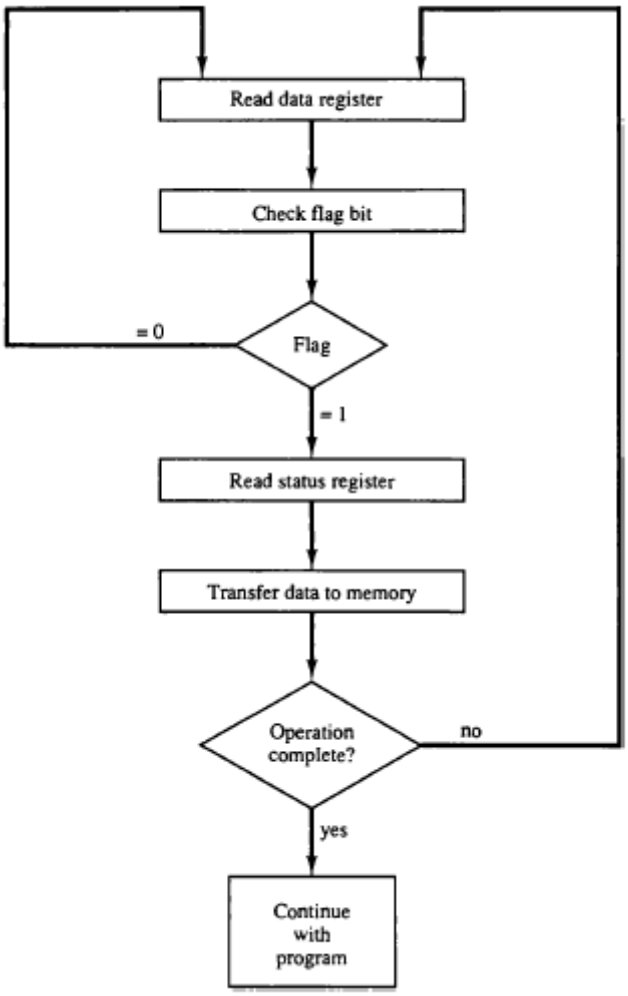
- In the programmed I/O method, the I/O device does not have direct access to memory. A transfer from an I/O device to memory requires the execution of

  several instructions by the CPU.
- An example of data transfer from an I/O device through an interface into the CPU is shown in Fig.



- The device transfers bytes of data one at a time as they are available.
- When a byte of data is available, the device places it in the I/O bus and enables its data valid line.
- The interface accepts the byte into its data register and enables the data accepted line.
- The interface sets a bit in the status register that we will refer to as an F or "flag" bit.
- The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by the interface.
- This is according to the handshaking procedure.

- A flowchart of the program that must be written for the CPU is shown in Fig.



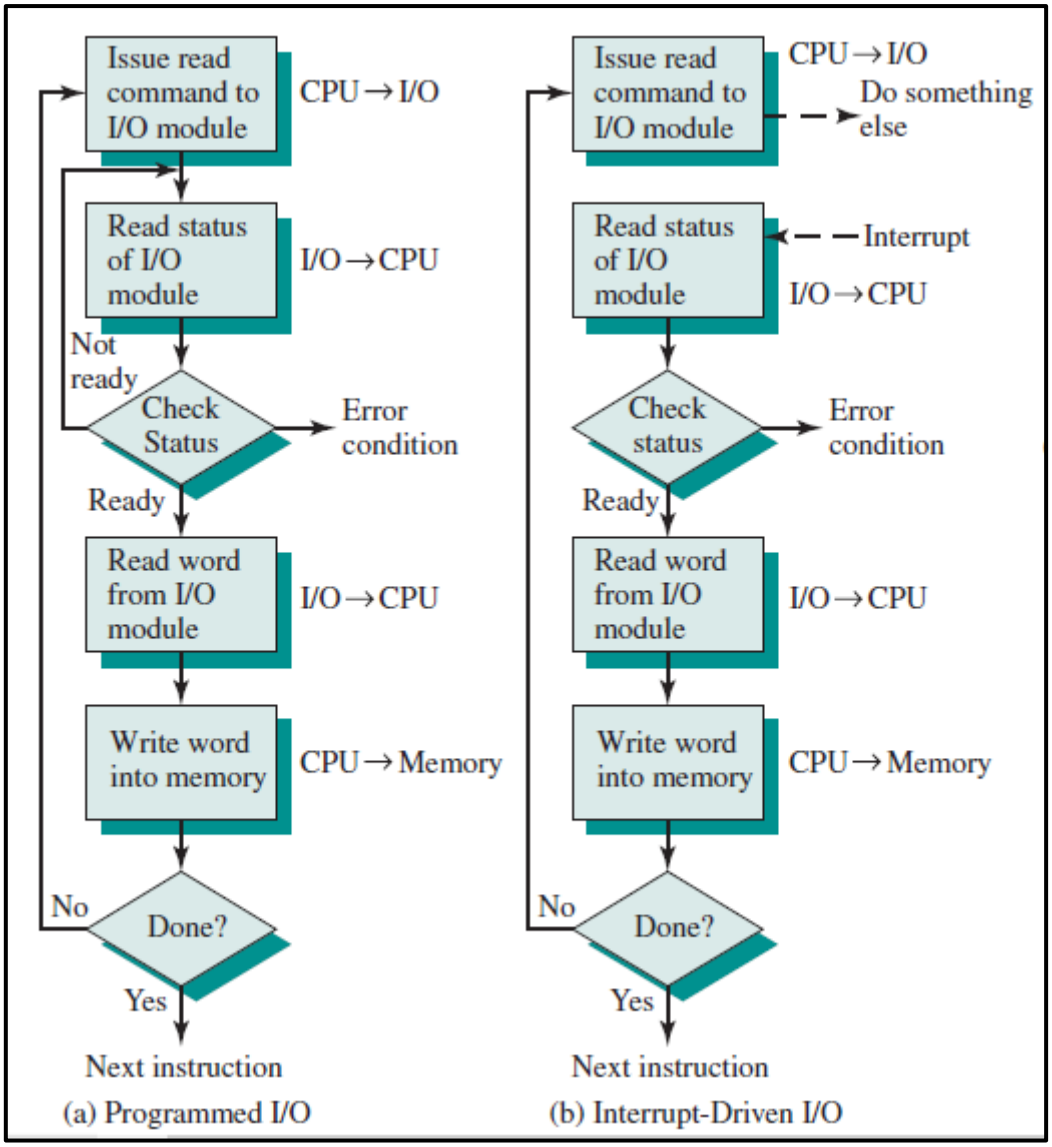
- It is assumed that the device is sending a sequence of bytes that must be stored in memory.
- The transfer of each byte requires three instructions:

1. **Read the status register.**
2. **Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.**
3. **Read the data register.**

- The programmed I/O method is particularly useful in small low-speed computers or in systems that are dedicated to monitor a device continuously.
- The CPU is wasting time while checking the flag instead of doing some other useful processing task

Interrupt Initiated I/O:

- The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready for either reception or transmission of data.
- The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the level of the performance of the entire system is severely degraded.
- An alternative is for the processor to issue an I/O command to a module and then go on to do some other useful work.
- This mode of transfer uses the interrupt facility.
- The CPU responds to the interrupt signal by storing the return address.
- While the CPU is running a program, it does not check the flag. However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- The CPU deviates from what it is doing to take care of the input or output transfer.
- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.



Priority Interrupt:

- Data transfer between the CPU and an VO device is initiated by the CPU. However, the CPU cannot start the transfer unless the device is ready to communicate with the CPU.
- The readiness of the device can be determined from an interrupt signal.
- A priority interrupt is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.
- Higher-priority interrupt levels are assigned to requests which, if delayed or interrupted, could have serious consequences. Devices with high speed

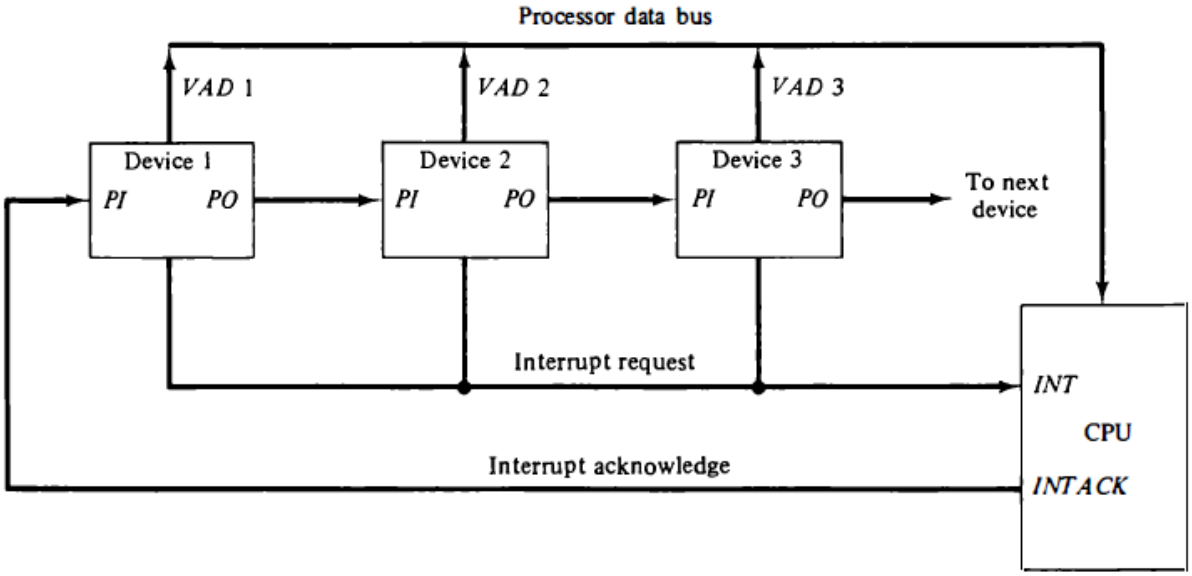
transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority.

- When two devices interrupt the computer at the same time, the computer services the device, with the higher priority first.
- Establishing the priority of simultaneous interrupts can be done by software or hardware.
- A polling procedure is used to identify the highest-priority source by software means.
- In a priority interrupt system, the CPU continuously polls (checks) various interrupt sources to determine which one has the highest priority and should be serviced first.
- This process involves regularly examining each interrupt line to see if it's active or requesting attention from the CPU.
- It's important to note that while polling can be simple to implement, it can also be inefficient in terms of CPU utilization since the CPU spends a lot of time checking for interrupts.

Daisy Chaining Priority:

- Daisy chaining is a technique used in priority interrupt systems to manage and prioritize multiple interrupt sources.
- It involves connecting the interrupt sources in a chain, where each source passes its request along the chain until it's acknowledged and serviced by the CPU.
- The interrupt sources are physically or logically connected in a daisy chain fashion. This means that the output of one interrupt source is connected to the input of the next interrupt source in the chain.
- The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.

- The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain. This method of connection between three devices and the CPU is shown in Fig:



- When no interrupts are pending, the line is in HIGH state. But if any of the devices raises an interrupt, it places the interrupt request line in the LOW state.
- The CPU acknowledges this interrupt request from the line and then enables the interrupt acknowledge line in response to the request.
- This signal is received at the PI(Priority in) input of device 1. If the device has not requested the interrupt, it passes this signal to the next device through its PO(priority out) output. (PI = 1 & PO = 1).

- However, if the device had requested the interrupt, (PI =1 & PO = 0), the device consumes the acknowledge signal and block its further use by placing 0 at its PO(priority out) output.
- The device then proceeds to place its interrupt vector address(VAD) into the data bus of CPU.

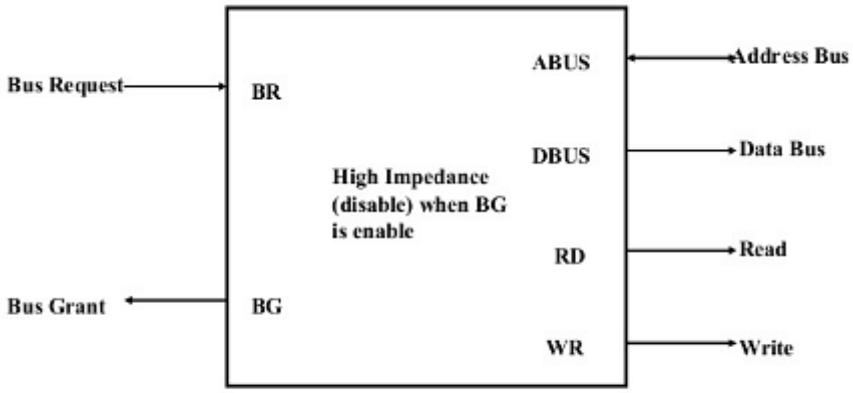
- The device puts its interrupt request signal in HIGH state to indicate its interrupt has been taken care of.
- If a device gets 0 at its PI input, it generates 0 at the PO output to tell other devices that acknowledge signal has been blocked. (PI = 0 & PO = 0)

Hence, the device having PI = 1 and PO = 0 is the highest priority device that is requesting an interrupt. Therefore, by daisy chain arrangement we have ensured that the highest priority interrupt gets serviced first and have established a hierarchy.

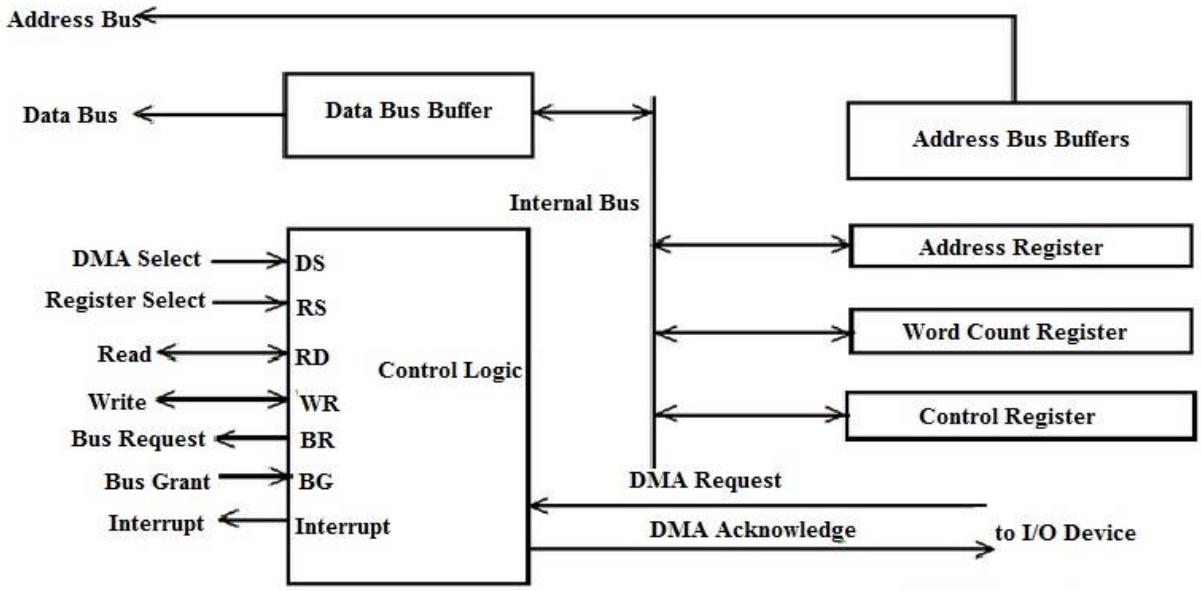
DMA (Direct Memory Access):

- DMA stands for "Direct Memory Access."
- It is a feature of computer systems that allows peripherals (devices like hard drives, network interfaces, graphics cards, etc.) to transfer data to or from the computer's memory without involving the central processing unit (CPU) in every data transfer step.
- This can significantly improve the efficiency of data transfer and free up the CPU to perform other tasks.
- In a typical data transfer scenario without DMA, the CPU would have to actively manage the entire process.
- With DMA, however, the peripheral device itself can directly access the system's memory and perform the data transfer. This process is managed by DMA controllers.
- The CPU sets up the necessary parameters for the transfer, such as source and destination addresses, data size etc and then initiates the transfer.
- Once initiated, the peripheral device and the memory communicate directly with each other, and the CPU is free to perform other tasks.
- **Applications and advantage:** DMA is particularly useful for high-speed data transfers and I/O operations, such as reading or writing large files,

video streaming, and network communication. It reduces CPU involvement and can lead to more efficient use of system resources.



CPU bus Signals for DMA Transfer



DMA Controller:

- A DMA controller, also known as a Direct Memory Access controller, is a hardware component in a computer system that manages direct memory access (DMA) operations.
- Its **primary function** is to process data transfer tasks from the central processing unit (CPU) to peripherals or memory without requiring constant involvement of the CPU.
- The DMA controller registers have three registers as follows:

1. **Address register** – It contains the address to specify the desired location in memory.
2. **Word count register** – It contains the number of words to be transferred.
3. **Control register** – It specifies the mode of transfer.

- **Address Register** is incremented after each word that is transferred to memory.
- **Word count register** is decremented by one after each word transfer.
- **"BG" and "BR"** typically refer to Bus Grant and Bus Request signals, respectively. These signals are important components in a computer system, allowing different devices to request and gain access to the system bus for data transfers
- **BR (Bus Request):** BR input is used by the DMA controller to request the CPU to hand over the control of buses. When this input is active CPU terminates the execution of current instruction and places address bus, data bus and read write lines into a high impedance state (behaves like an open circuit, which means output is disconnected).
- **BG (Bus Grant):** CPU activates BG output (BG = 1) to inform DMA that the buses are in high impedance state. Now DMA takes control over buses for data transfer without processor intervention. When the DMA terminates the transfer, it disables BR line and CPU disables BG line and takes control of buses and returns to its normal operation.
