INPUT-OUTPUT ORGANIZATION

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.

- Video monitors are the most commonly used peripherals. They consist of a keyboard as the input device and a display unit as the output device. There are different types of video monitors, but the most popular use a cathode ray tube (CRT).

- Printers provide a permanent record on paper of computer output data or text. There are three basic types of character printers: daisywheel, dot matrix, and laser printers.

    - **The daisywheel printer** contains a wheel with the characters placed along the circumference. To print a character, the wheel rotates to the proper position and an energized magnet then presses the letter against the ribbon.

    - **The dot matrix printer** contains a set of dots along the printing mechanism.

    - **The laser printer** uses a rotating photographic drum that is used to imprint the character images. The pattern is

then transferred onto paper in the same manner as a copying machine.

ASCII Alphanumeric Characters

- Input and output devices that communicate with people and the computer are usually involved in the transfer of alphanumeric information to and from the device and the computer is ASCII (American Standard Code for Information Interchange) .It uses seven bits to code 128 characters.

Input-Output Interface

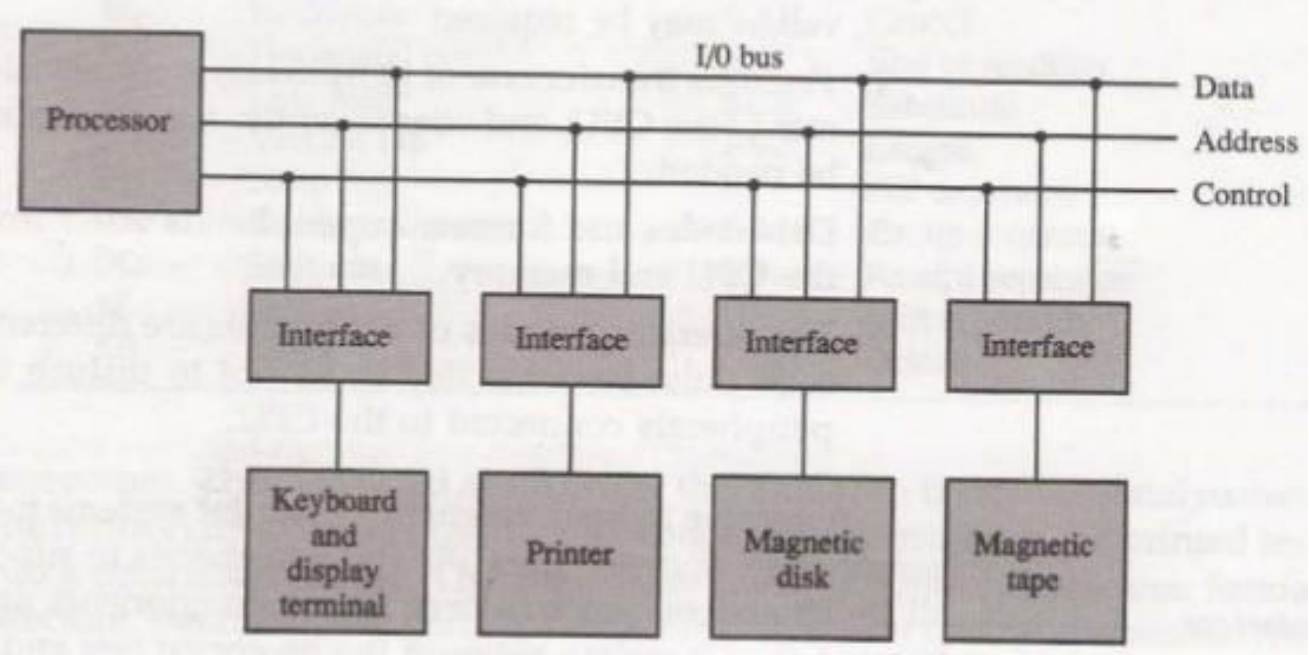
- Input-output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit.

- 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.

- In addition, each device may have its own controller that supervises the operations of the particular mechanism in the peripheral.

I/O Bus and Interface Modules

- Communication link between the processor and several peripherals is shown in following diagram.



- The I/O bus consists of data lines, address lines, and control lines. The magnetic disk, printer, and terminal are employed in practically any general-purpose computer. 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, interprets them for the peripheral, and provides signals for the peripheral controller. It also synchronizes the data flow and supervises the transfer between peripheral and

processor. Each peripheral has its own controller that operates the particular electromechanical device.

- At the same time that the address is made available in the address lines, the processor provides a function code in the control lines. The interface selected responds to the function code and proceeds to execute it. The function code is referred to as an I/O command

- There are four types of commands that an interface may receive**. They are classified as control, status, status, data output, and data input.**

1. **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.

2. **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.

3. **A data output command** causes the interface to respond by transferring data from the bus into one of its registers.

4. **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.

I/O Versus Memory Bus

- In addition to communicating with I/O, the processor must communicate with the memory unit.

- Like the I/O bus, the memory bus contains data, address, and read/write control lines. 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 Versus Memory-Mapped I/O

Isolated I/O configuration

- **In the isolated I/O configuration**, the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register.

- When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines.

- At the same time, it enables the I/O read (for input) or I/O write (for output) control line. This informs the external components that are attached to the common bus that the

address in the address lines is for an interface register and not for a memory word.

- On the other hand, when the CPU is fetching an instruction or an operand from memory, it places the memory address on the address lines and enables the memory read or memory write control line. This informs the external components that the address is for a memory word and not for an I/O interface.

Memory-Mapped Configuration

- Only one set of read and write signals and do not distinguish between memory and I/O addresses. This configuration is referred to as memory mapped I/O.

- In a memory-mapped I/O organization there are no specific input or output instructions. The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words.

Asynchronous Data Transfer

- The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator.

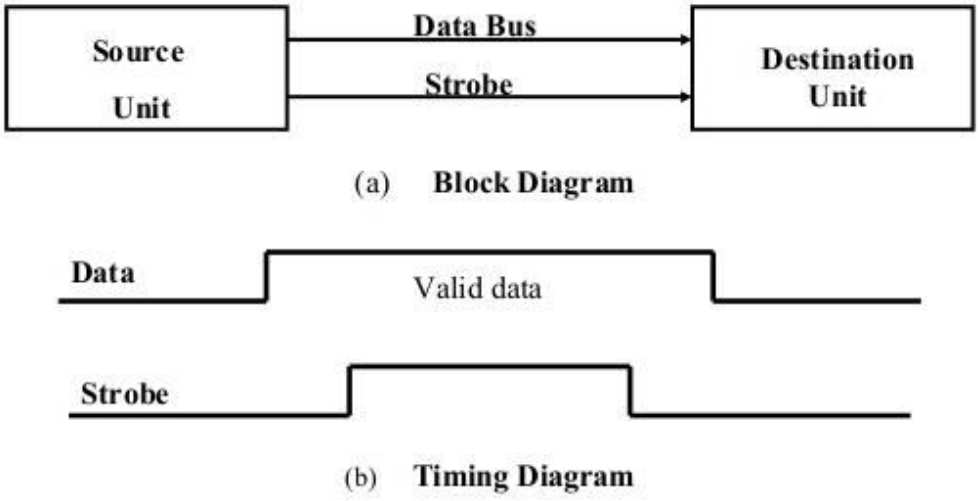
- In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers. In that case, the two units are said to be asynchronous to each other.

- 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.

- Two ways to achieve are as follows
    1. **strobe pulse**
    2. **handshaking**
- It is customary to specify the asynchronous transfer between two independent units by means of **a timing diagram** that shows the timing relationship that must exist between the control signals and the data in buses.

Strobe Control

- 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. diagram shows a source-initiated transfer.



Source-Initiated strobe for Data Transfer

- The data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word.

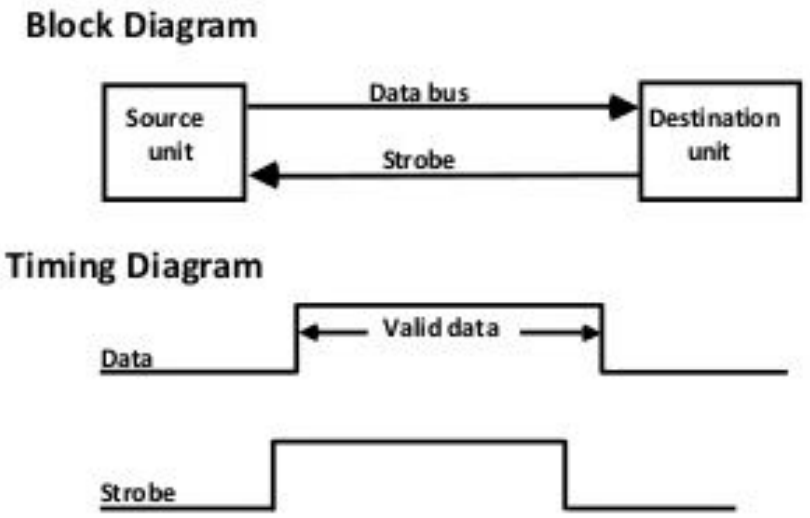
- The strobe is a single line that informs the destination unit when a valid data word is available in the bus. As shown in the timing diagram of the source unit first places the data on the data bus.

Destination

- Diagram shows a data transfer initiated by the destination unit. In this case the 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 data must be valid and remain in the bus long enough for the destination unit to accept it.

- The falling edge of the strobe pulse can be used again to trigger a destination register. The destination unit then disables the strobe. The source removes the data from the bus after a predeter-mined time interval.



- **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. Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

Handshaking

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

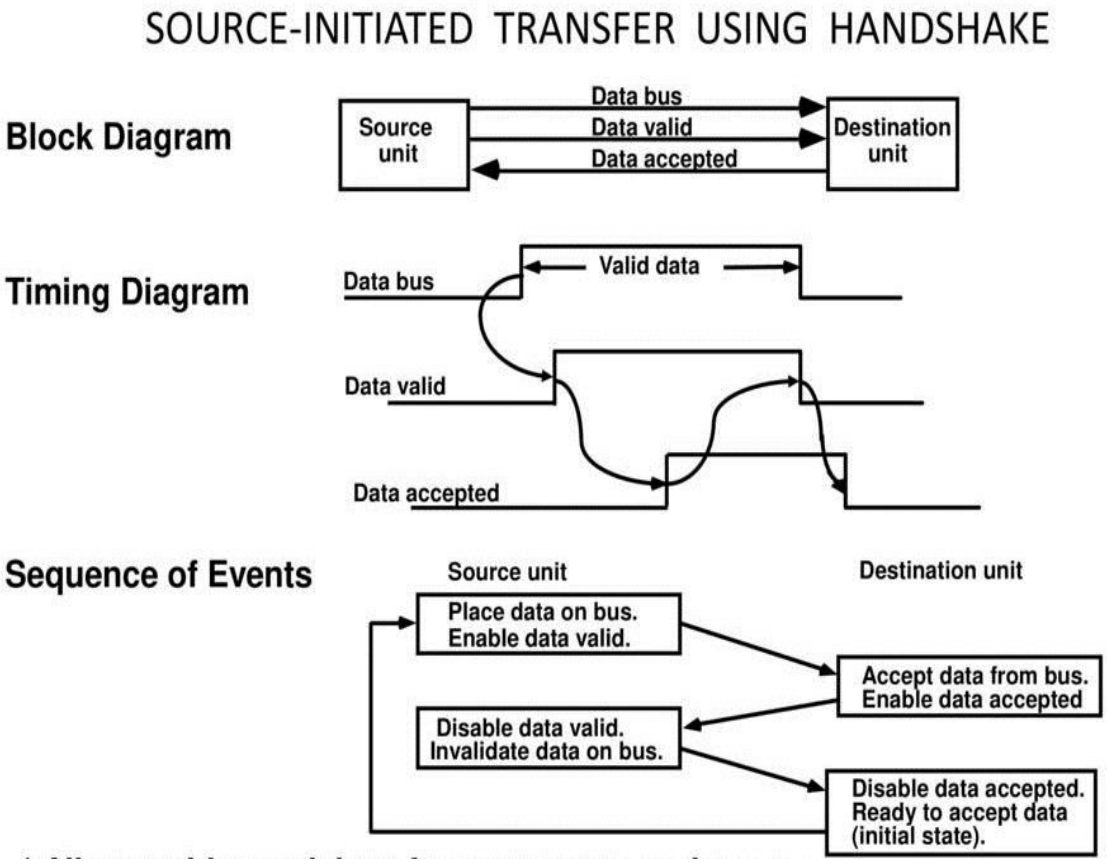
- Handshaking method of data transfer is as follows. One control line is in the same direction as the data flow in the bus from the source to the destination.

- It is used by the source unit to inform the destination unit whether there are valued 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. The sequence of control during the transfer depends on the unit that initiates the transfer.

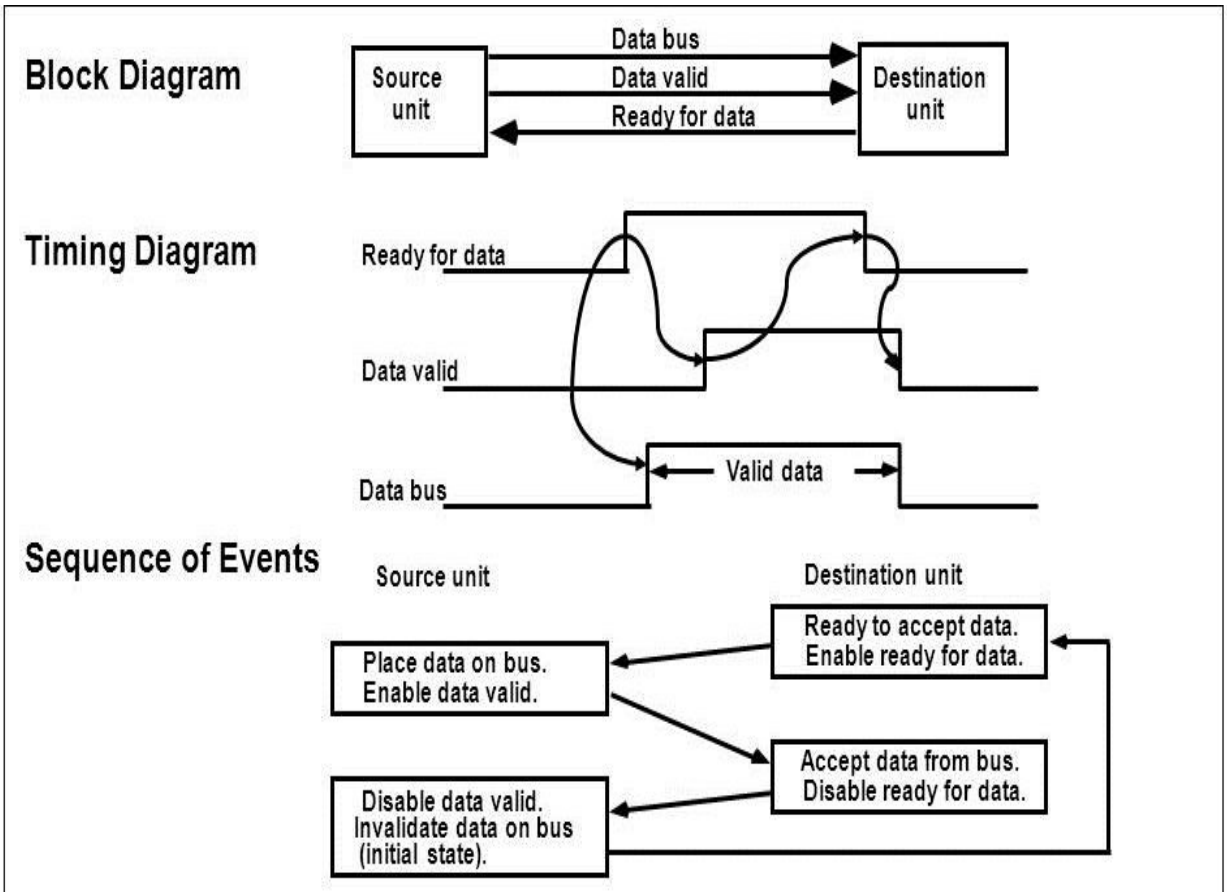
- Diagram shows the data transfer procedure when initiated by the source. The two handshaking lines are data valid, which is generated by the source unit, and data accepted, generated by the destination unit. The timing diagram shows the exchange of signals between the two units.

- The sequence of events listed in part (c) shows the four possible states that the system can be at any given time.



1. The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.
2. The data accepted signal is activated by the destination unit after it. Accepts the data from the bus.
3. The source unit then disables its data valid signal, which invalidates the data on the bus.
4. The destination unit then disables its data accepted signal and the system goes into its initial state.

- The destination-initiated transfer using handshaking lines is shown in diagram the name of the signal generated by the destination unit is ready for data. The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit.



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 operations are the result of I/O instructions written in the computer program. Each data item transfer is initiated by an instruction in the program.

- Usually, the transfer is to and from a CPU register and peripheral. Other instructions are needed to transfer the data to and from CPU and memory. Transferring data under program control requires constant monitoring of the peripheral by the CPU.

- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made. In the programmed I/O method, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly

Interrupt-initiated I/O

- It can be avoided by using an interrupt facility and special commands to inform the interface to issue an interrupt request signal when the data are available from the device.

- In the meantime the CU can proceed to execute another program. The interface meanwhile keeps monitoring the device. When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer. Upon detecting the external interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process the I/O transfer, and then returns to the task it was originally performing

Direct memory access (DMA)

- In direct memory access (DMA), the interface transfers data into and out of the memory unit through the memory bus. The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks.

- When the transfer is made, the DMA requests memory cycles through the memory bus. When the request is granted by the memory controller, the DMA transfers the data directly into memory

Priority Interrupt

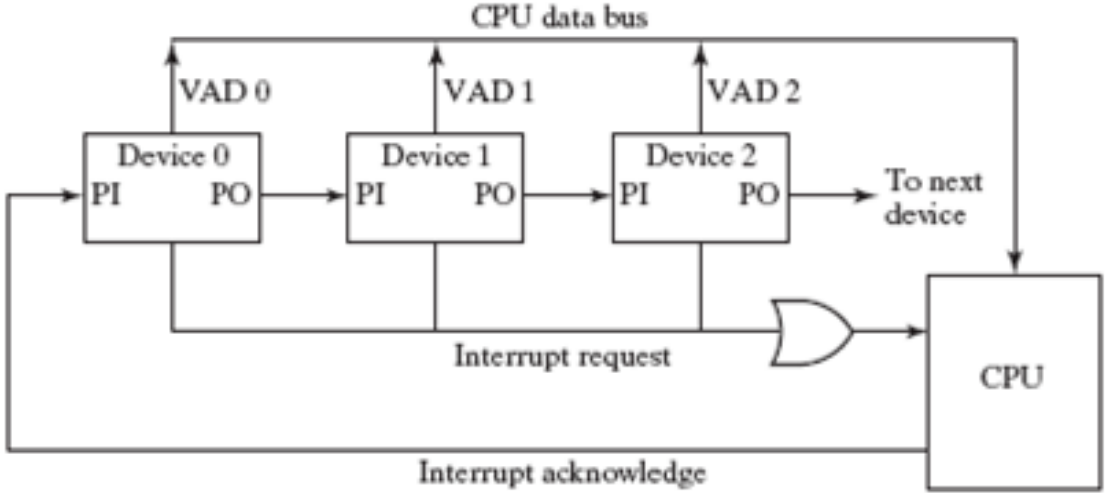
- A priority interrupts is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more request arrive simultaneously.

- The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.

- When two devices interrupt the computer at the same time, the computer services the device, with the higher priority first.

Daisy-Chaining Priority

- 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 diagram



- The interrupt request line is common to all devices and forms a wired logic connection. If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU.

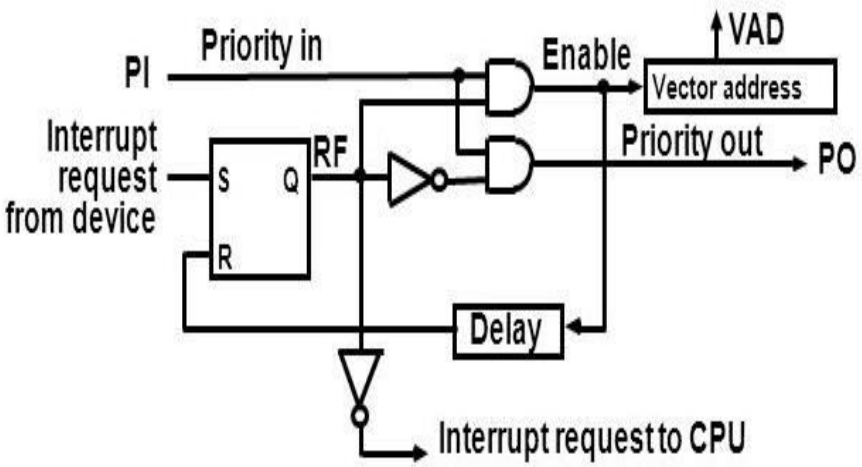
- When no interrupts are pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU. This is equivalent to a negative logic OR operation.

- The CPU responds to an interrupt request by enabling the interrupt acknowledge line. This signal is received by device 1 at its PI (priority in) input. The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.

- If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output. It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle. Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its VAD on the data bus.

- **Diagram shows the internal logic** that must be included with in each device when connected in the daisy-chaining scheme. The device sets its RF flip-flop when it wants to interrupt the CPU. The output of the RF flip-flop goes through an open-collector inverter, a circuit that provides the wired logic for the common interrupt line. If PI = 0, both PO and the enable line to VAD are equal to 0, irrespective of the value of RF.

One stage of the daisy chain priority arrangement



PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

- If PI = 1 and RF = 0, then PO = 1 and the vector address is disabled. This condition passes the acknowledge signal to the next device through PO.

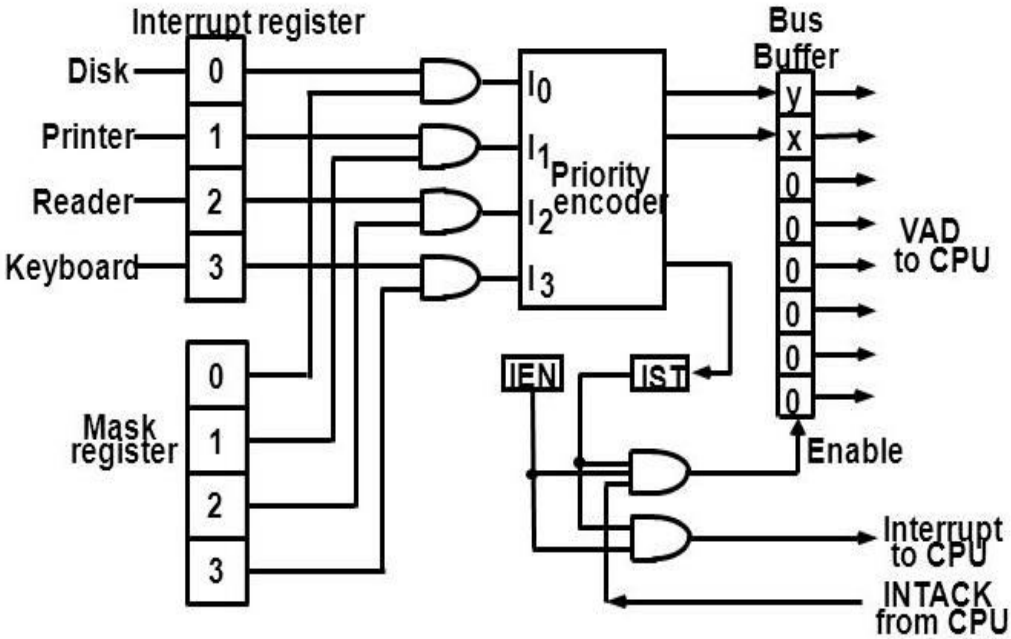
- The device is active when PI = 1 and RF = 1. This condition places a 0 in PO and enables the vector address for the data bus. It is assumed that each device has its own distinct vector address. The RF flip-flop is reset after a sufficient delay

Parallel Priority Interrupt

- The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device. Priority is established according to the position of the bits in the register.

- In addition to the interrupt register the circuit may include a mask register whose purpose is to control the status of each interrupt request. The mask register can be programmed to disable lower-priority interrupts while a higher-priority device is being serviced. It can also provide a facility that allows a high-priority device to interrupt the CPU while a lower-priority device is being serviced.

- The priority logic for a system of four interrupt sources is shown in diagram. It consists of an interrupt register whose individual bits are set by external conditions and cleared by program instructions.



- The magnetic disk, being a high-speed device, is given the highest priority. The printer has the next priority, followed by a character reader and a keyboard.
- The mask register has the same number of bits as the interrupt register.. Each interrupt bit and its corresponding mask bit are applied to an AND gate to produce the four inputs to a priority encoder.
- In this way an interrupt is recognized only if its corresponding mask bit is set to 1 by the program. The priority encoder generates two bits of the vector address, which is transferred to the CPU.
- Another output from the encoder sets an interrupt status flip-flop IST when an interrupt that is not masked occurs. The interrupt enable flip-flop IEN can be set or cleared by the program to provide an overall control over the interrupt system.
- The outputs of IST ANDed with IEN provide a common interrupt signal for the CPU. The interrupt acknowledge INTACK signal from the CPU enables the bus buffers in the output register and a vector address VAD is placed into the data bus. We will now explain the priority encoder circuit and then discuss the interaction between the priority interrupt controller and the CPU.

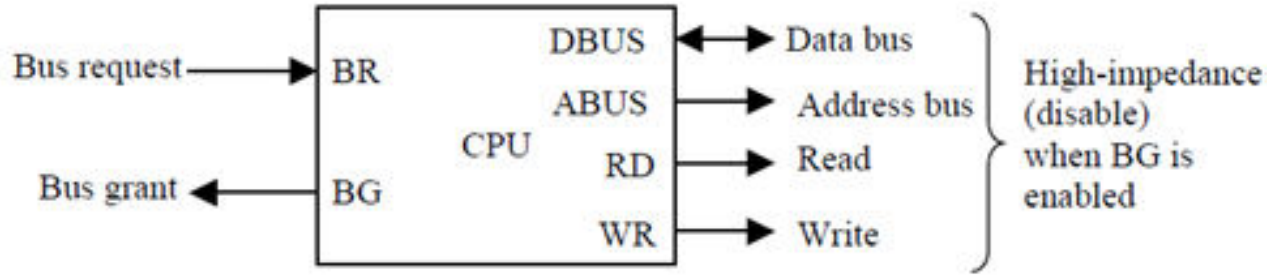
Direct Memory Access (DMA)

- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called direct memory access (DMA).

- During DMA transfer, the CPU is idle and has no control of the memory buses. A DMA controller takes over the buses to manage the transfer directly between the I/O device and memory.

- The CPU may be placed in an idle state in a variety of ways. One common method extensively used in microprocessors is to disable the buses through special control signals.

- Diagram shows two control signals in the CPU that facilitate the DMA transfer.



- The bus request (BR) input is used by the DMA controller to request the CPU to relinquish control of the buses. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state.

behaves like an open circuit, which means that the output is disconnected and does not have a logic significance.

- The CPU activates the Bus grant (BG) output to inform the external DMA that the buses are in the high-impedance state. The DMA that originated the bus request can now take control of the buses to conduct memory transfers without processor intervention. When the DMA terminates the transfer, it disables the bus request line. The CPU disables the bus grant, takes control of the buses, and returns to its normal operation.

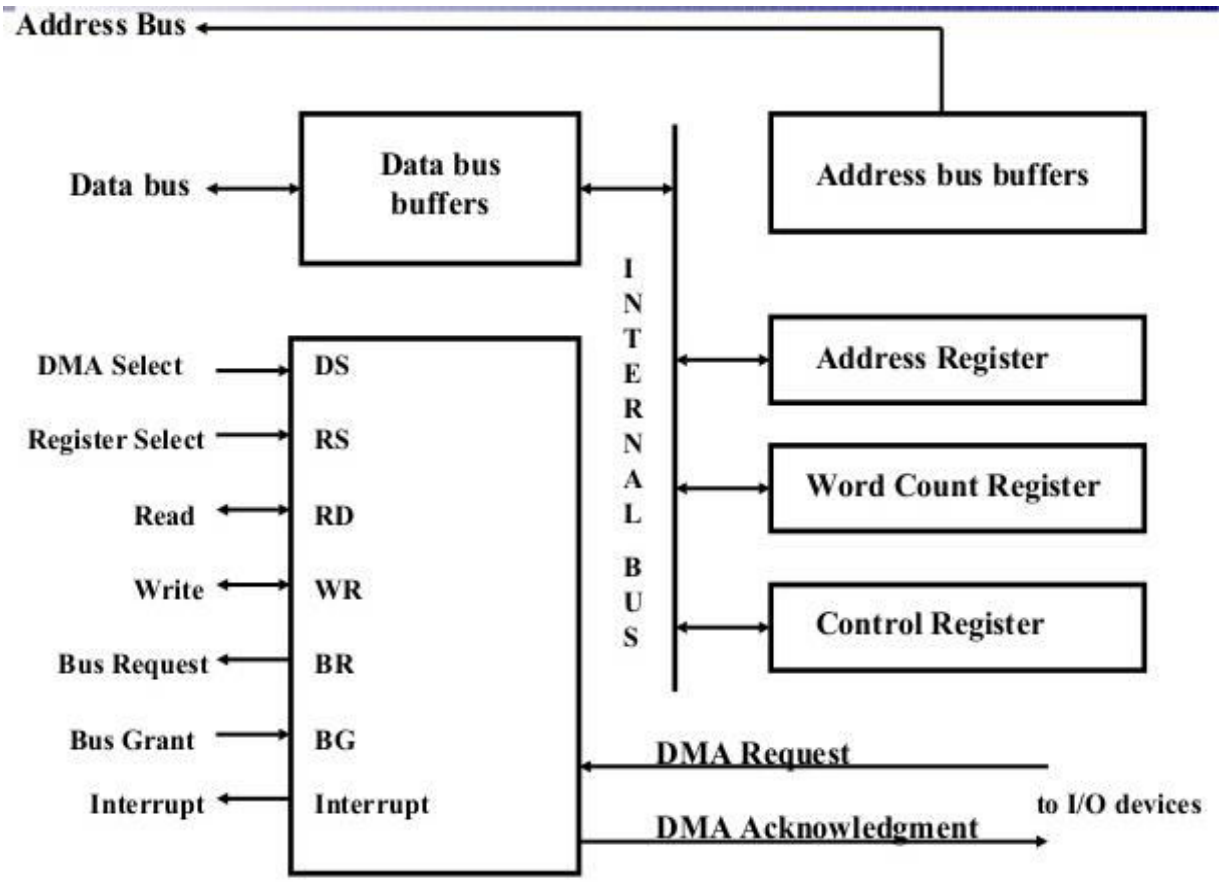
DMA Controller

- The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device. In addition, it needs an address register, a word count register, and a set of address lines.

- The address register and address lines are used for direct communication with the memory. The word count register specifies the number of words that must be transferred. The data transfer may be done directly between the device and memory under control of the DMA.

- Diagram shows the block diagram of a typical DMA controller. The unit communicates with the CPU via the data bus and control lines.

➢



Block Diagram of DMA Controller

- The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (register select) inputs.

- The RD (read) and WR (write) inputs are bidirectional. When the BG (bus grant) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers.

- When BG = 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.

- The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.

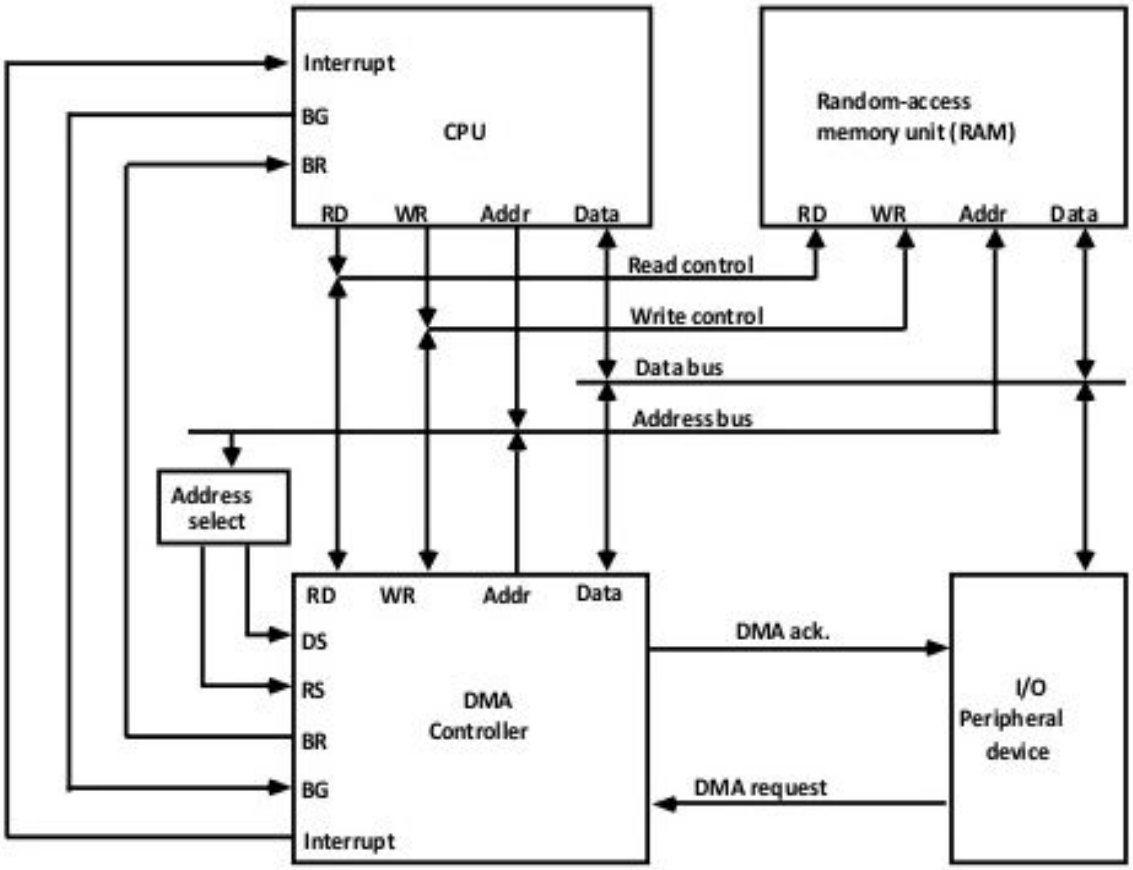
- The DMA controller has three registers: an address register, a word count register, and a control register. The address register contains an address to specify the desired location in memory. The address bits go through bus buffers into the address bus.

- The word count register is incremented after each word that is transferred to memory. The word count register holds the number of words to be transferred. This register is decremented by one after each word transfer and internally tested for zero. The control register specifies the mode of transfer.

DMA Transfer

- The position of the DMA controller among the other components in a computer system is illustrated in diagram. The CPU communicates with the DMA through the address and data buses as with any interface unit. The DMA has its own address, which activates the DS and RS lines.

- The CPU initializes the DMA through the data bus. Once the DMA receives the start control command, it can start the transfer between the peripheral device and the memory.



- When the peripheral device sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses. The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then puts the current value of its address register into the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the peripheral device. Note that the RD and WR lines in the DMA controller are bidirectional.

- The direction of transfer depends on the status of the BG line. When BG line. When BG = 0, the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers. When BG = 1, the RD and WR and output lines from the DMA controller to the random-access memory to specify the read or write operation for the data.

- When the peripheral device receives a DMA acknowledge, it puts a word in the data us (for write) or receives a word from the data bus (for read). Thus the DMA controls the read or write operations and supplies the address for the memory. The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.

- For each word that is transferred, the DMA increments its address register and decrements its word count register. If the word count does not reach zero, the DMA checks the request line coming from the peripheral. For a high-speed device, the line will be active as soon as the previous transfer is completed. A second transfer is then initiated, and the process continues until the entire block is transferred.

- It the word count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of the termination by means of an interrupt. When the CPU responds to the interrupt, it reads the content of the word count register. The zero value of this register indicates that all the words were transferred successfully. The CPU can read this register at any time to check the number of words already transferred.