Logical Organization of Computer

I/O Architecture

Developed By:
Vivek Vyas
Department of MCA
DDU

❑ Computer I/O

- The input and output subsystem of a computer provides an efficient mode of communication between the CPU and the outside environment.
- Programs and data must be entered into the memory for processing, and results obtained from computations must be recorded or displayed.
- To make a computer work properly, data paths must be provided that let information flow between CPU(s), RAM, and the score of I/O devices that can be connected to a personal computer.
- These data paths, which are denoted as the *buses* , act as the primary communication channels inside the computer.

❑ Computer I/O

- Any computer has a *system bus* that connects most of the internal hardware devices.
- A typical system bus is the PCI (*Peripheral Component Interconnect*) bus.
- Among the input and output devices commonly found in computer systems are keyboards, displays, printers, magnetic drives, and compact disk read-only memory (CD-ROM) drives.
- Other input and output devices frequently encountered are modems or other communication interfaces, scanners, and sound cards with speakers and microphones.

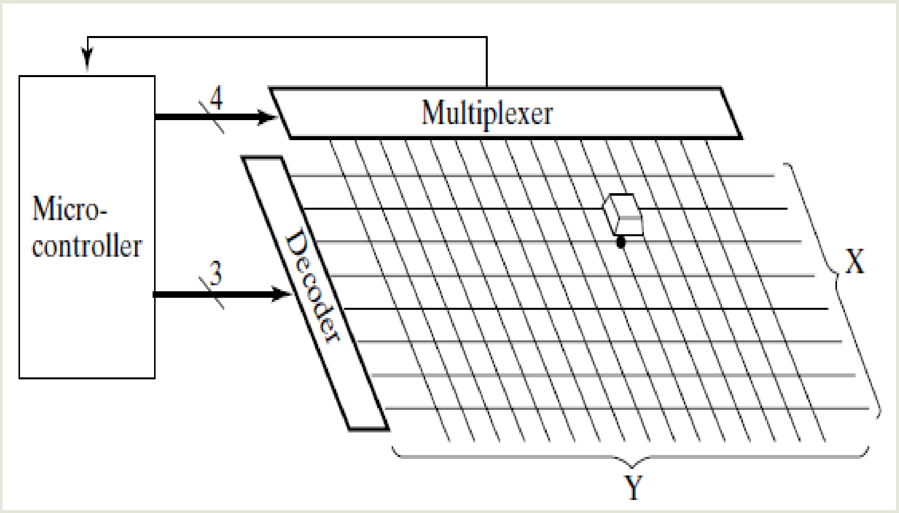
❑ SAMPLE PERIPHERALS

- Devices that the CPU controls directly are said to be connected *online.*
- These devices communicate directly with the CPU or transfer binary information into or out of the memory upon command from the CPU.
- Input or output devices attached to the computer online are called *peripherals.*
- For example: *Keyboards*, *display units* and *printers* are common peripheral devices.

❑ Keyboard

- The keyboard is among the simplest of the electromechanical devices attached to the typical computer.
- Since it is manually controlled, it has one of the slowest data rates of any peripheral.
- The keyboard consists of a collection of keys that can be pressed by the user.
- It is necessary to detect which of the keys have been pressed.
- To do this, a *scan matrix* that lies beneath the keys is used, as shown in diagram.

❑ Keyboard



❑ Keyboard

- The matrix shown in the figure is 8 × 16, giving 128 intersections, so it can handle up to 128 keys.
- A decoder drives the *X* lines of the matrix.
- A multiplexer is attached to the *Y* lines of the matrix.
- The decoder and the multiplexer are controlled by a microcontroller, a tiny computer that contains RAM, ROM, a timer, and simple I/O interfaces.

❑ Keyboard

- The microcontroller is programmed to periodically scan all intersections in the matrix by manipulating the control inputs of the decoder and multiplexer.
- If the key is pressed at an intersection, a signal path is closed from an output of the *X* decoder to an input of the *Y* multiplexer. The existence of this path is sensed at an input to the microcontroller.
- Whether a key is pressed or released, the control code at the time of the event is sensed and is translated by the microcontroller into a *K-scan code*.
- When a key is pressed, a *make code* is produced; when a key is released, a *break code* is produced.

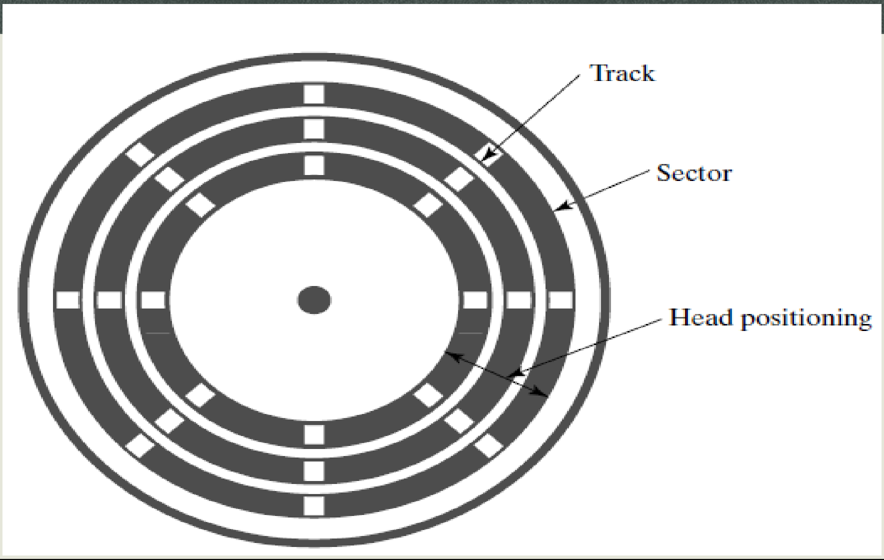
❑ Keyboard

- Thus, there are two codes for each key, one for when the key is pressed and one for when it is released.
- Note that the scanning of the entire keyboard occurs hundreds of times per second, so there is no danger of missing any press or release of a key.

❑ Hard Drive

- The hard drive is the primary intermediate-speed, nonvolatile, writable storage medium for most computers.
- The typical hard drive stores information serially on a nonremovable disk.
- There are one or more read/writer heads per recording surface.
- Each disk is divided into tracks. The set of tracks that are at the same distance from the center of all disk surfaces is referred to as a *cylinder.*

❑ Hard Drive



❑ Hard Drive

- Each track is divided into *sectors* containing a fixed number of bytes.
- The number of bytes per sector typically ranges from 256 to 4K.
- The typical byte address includes the cylinder number, head number, sector number, and word offset within the sector.
- The time required to move the heads from the current cylinder to the desired cylinder is called the *seek time.*
- The time required to rotate the disk from its current position to that having the desired sector under the heads is called the *rotational delay*.

❑ Hard Drive

- In addition, a certain amount of time is required by the drive controller to access and output information. This is the *controller time*.
- The time required to locate a word on the disk is the *disk access time*, which is the sum of the controller time, the seek time, and the rotational delay.
- The transfer rate for a block of words, once the block has been located, is the *disk transfer rate,* typically specified in megabytes/second (MB/s).
- The sum of the disk access time and the disk transfer rate times the number of bytes per sector gives an estimate of the time required to transfer the information in a sector to or from the hard drive.

❑ I/O INTERFACES

- Peripherals connected to a computer need special communication links to interface them with the CPU.
- The purpose of these links is to resolve the differences in the properties of the CPU and memory and the properties of each peripheral.
- The major differences are as follows:
- 1. Peripherals are often electromechanical devices whose manner of operation is different from that of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.

❑ I/O INTERFACES

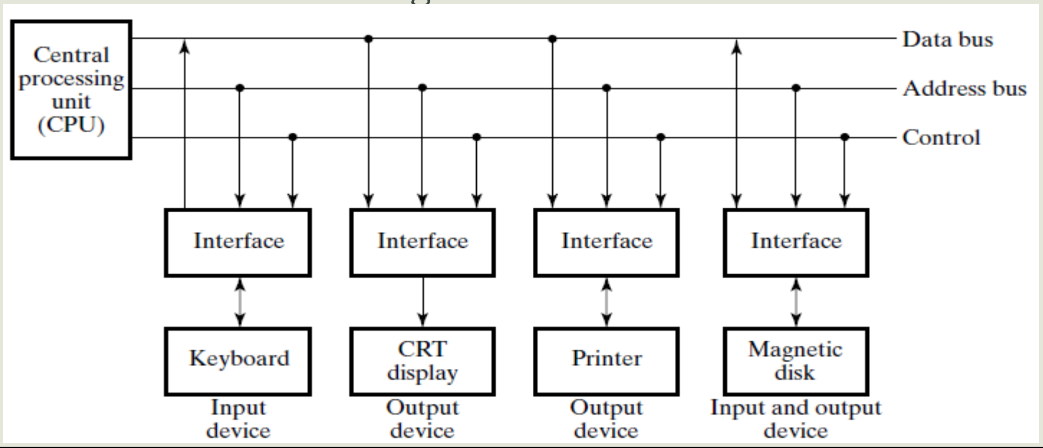
- **2.** The data-transfer rate of peripherals is usually different from the clock rate of the CPU. Consequently, a synchronization mechanism may be needed.
- **3.** Data codes and formats in peripherals differ from the word format in the CPU and memory.
- **4.** The operating modes of peripherals differ from each other, and each must be controlled in a way that does not disturb the operation of other peripherals connected to the CPU.

❑ I/O INTERFACES

- To resolve these differences, computer systems include special hardware components between the CPU and the peripherals to supervise and synchronize all input and output transfers. These components are called *interface units.*
- In addition, each device has its own controller to supervise the operations of the particular mechanism of that peripheral.
- For example, the controller in a printer attached to a computer controls the motion of the paper, the timing of the printing, and the selection of the characters to be printed.

❑ I/O Bus and Interface Unit

- A typical communication structure between the CPU and several peripherals is shown in diagram.



❑ I/O Bus and Interface Unit

- Each peripheral has an interface unit associated with it. The common bus from the CPU is attached to all peripheral interfaces.
- To communicate with a particular device, the CPU places a device address on the address bus.
- Each interface attached to the common 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.

❑ I/O Bus and Interface Unit

- All peripherals with addresses that do not correspond to the address on the bus ignore the bus activity.
- At the same time that the address is made available on the address bus, the CPU provides a function code on the control lines.
- The selected interface responds to the function code and proceeds to execute it.
- If data must be transferred, the interface communicates with both the device and the CPU data bus to synchronize the transfer.

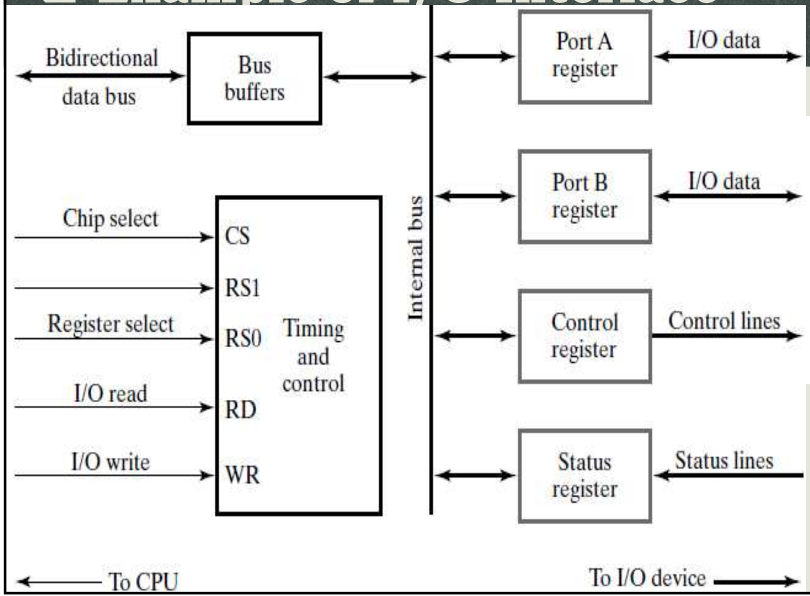
❑ I/O Bus and Interface Unit

- In addition to communicating with the I/O devices, the CPU of a computer must communicate with the memory unit through an address and data bus.
- There are two ways that external computer buses communicate with memory and I/O.
- One method uses common data, address, and control buses for both memory and I/O. This is referred to this configuration as *memory-mapped I/O.*

❑ I/O Bus and Interface Unit

- The second alternative is to share a common address bus and data bus, but use different control lines for memory and I/O.
- This method is referred to as the *isolated I/O configuration.*

❑ Example of I/O Interface



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

❑ Example of I/O Interface

- A typical I/O interface unit consists of two data registers called *ports*, a control register, a status register, a bidirectional data bus, and timing and control circuits.
- The function of the interface is to translate the signals between the CPU buses.
- The I/O data from the device can be transferred into either port A or port B.
- The interface may operate with an output device, with an input device, or with a device that requires both input and output.

❑ Example of I/O Interface

- If the interface is connected to a printer, it will only output data; if it services a scanner, it will only input data. A
- hard drive transfers data in both directions, but not at the same time; so the interface needs only one set of I/O bidirectional data lines.
- The *control register* receives control information from the CPU.
- The bits in the status register are used for status conditions and for recording errors that may occur during data transfer.

❑ Example of I/O Interface

- The interface registers communicate with the CPU through the bidirectional data bus.
- The circuit sets the chip select (*CS*) input when the interface is selected by the address bus.
- The two *register select inputs RS*1 and *RS*0 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.

❑ Example of I/O Interface

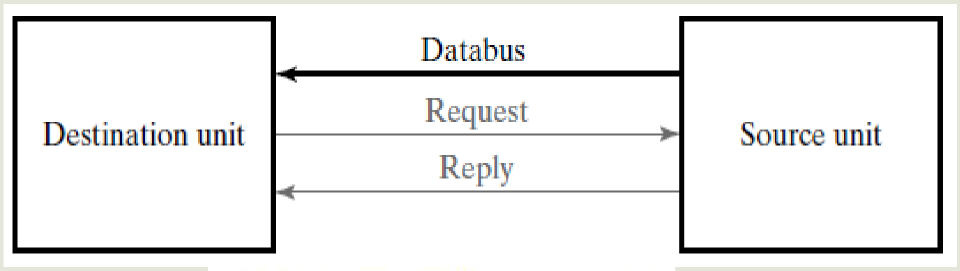
- The CPU, interface, and I/O device are likely to have different clocks that are not synchronized with each other. Thus, these units are said to be *asynchronous* with respect to each other.

❑ Handshaking

- The *handshaking* method uses two control signals to deal with the timing of transfers.
- In addition to the signal from the unit initiating the transfer, there is a second control signal from the other unit involved in the transfer.
- The basic principle of a two-signal handshaking procedure for data transfer is as follows.
- One control line from the initiating unit is used to request a response from the other unit.

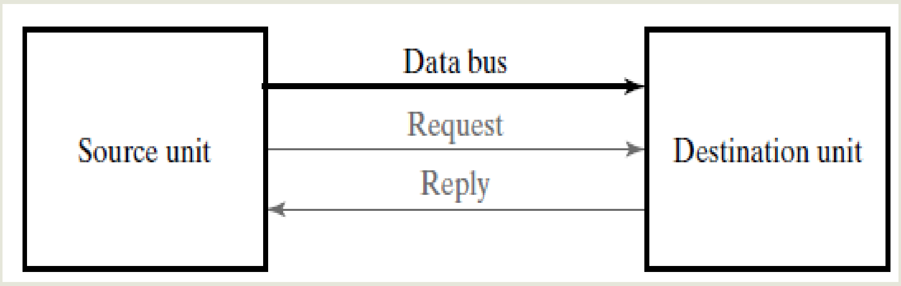
❑ Handshaking

- The second control line from the other unit is used to reply to the initiating unit that the response is occurring.
- In this way, each unit informs the other of its status, and the result is an orderly transfer through the bus.



(a) Destination-initiated transfer

❑ Handshaking



(b) Source-initiated transfer

❑ SERIAL COMMUNICATION

- The transfer of data between two units may be parallel or serial.
- In parallel data transfer, each bit of the message has its own path, and the entire message is transmitted at one time.
- This means that an *n*-bit message is transmitted in parallel through *n* separate paths.
- In serial data transmission, each bit in the message is sent in sequence, one at a time.

❑ SERIAL COMMUNICATION

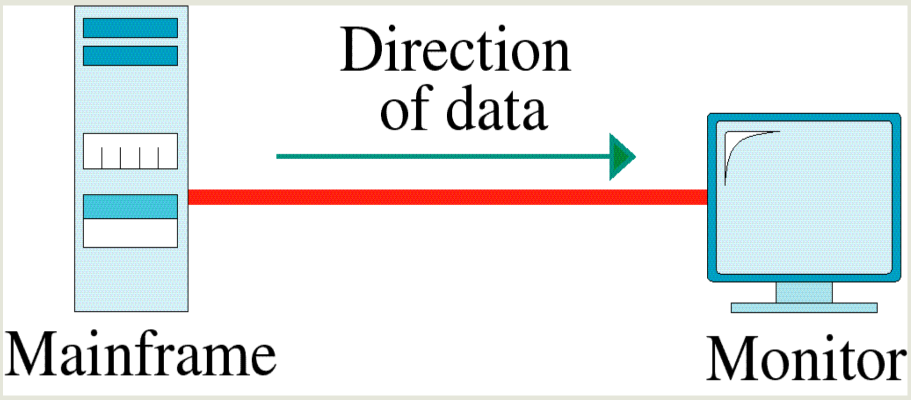
- Parallel transmission is faster, because multiple signal lines operate in parallel.
- It is used for short distances and when speed is important.
- Serial transmission is slower, but less expensive.
- One way that computers and terminals that are remote from each other are connected is via telephone lines.
- Since telephone lines were originally designed for voice communication, but computers communicate in terms of digital signals, some form of conversion is needed.

❑ SERIAL COMMUNICATION

- The devices that do the conversion are called *data sets* or *modems* (modulator–demodulators).
- Serial data can be transmitted between two points in three different modes: simplex, half duplex, or full duplex.

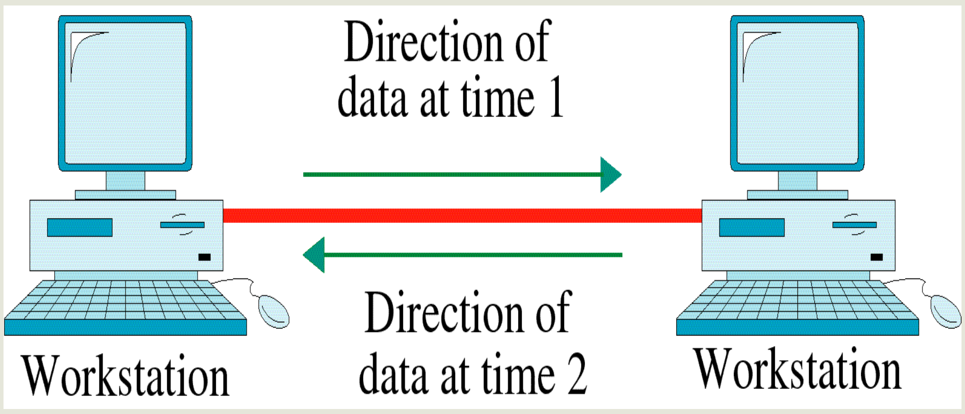
❑ SERIAL COMMUNICATION

- **Simplex**



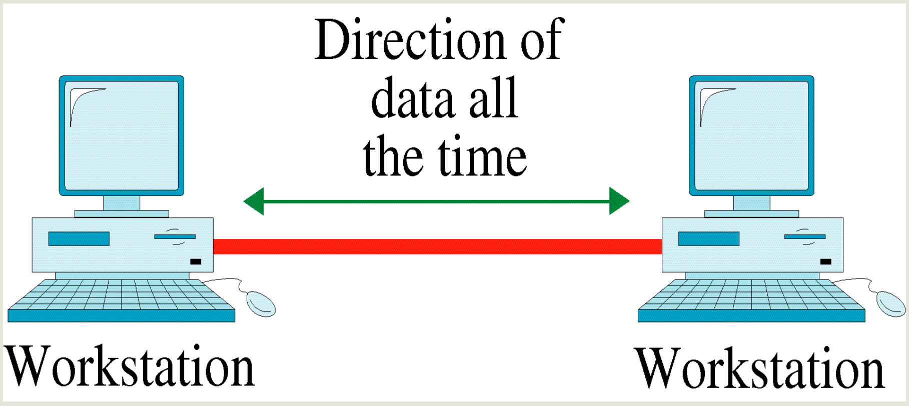
❑ SERIAL COMMUNICATION

- **Half Duplex**



❑ SERIAL COMMUNICATION

- **Full Duplex**



❑ Modes of I/O Data Transfer

- Data transfer between the central computer and I/O devices may be handled in a variety of modes, some of which use the CPU as an intermediate path, while others transfer the data directly to and from the memory.
- Data transfer to and from peripherals may be handled in one of three possible modes:
- **1.** Data transfer under program control.
- **2.** Interrupt-initiated data transfer.
- **3.** Direct memory access transfer.

❑ Modes of I/O Data Transfer

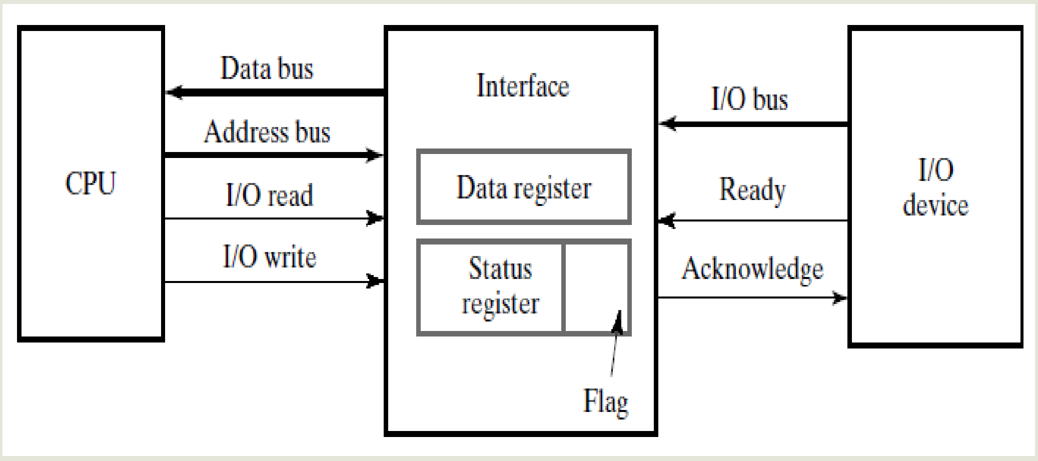
- **1. Data transfer under program control:**
- Program-controlled operations are the result of I/O instructions written in the computer program.
- Each transfer of data is initiated by an instruction in the program. Usually, the transfer is to and from a CPU register and peripheral.
- Other instructions are needed to transfer the data to and from the CPU and memory.
- Transferring data under program control requires constant monitoring of the peripheral by the CPU.

❑ Modes of I/O Data Transfer

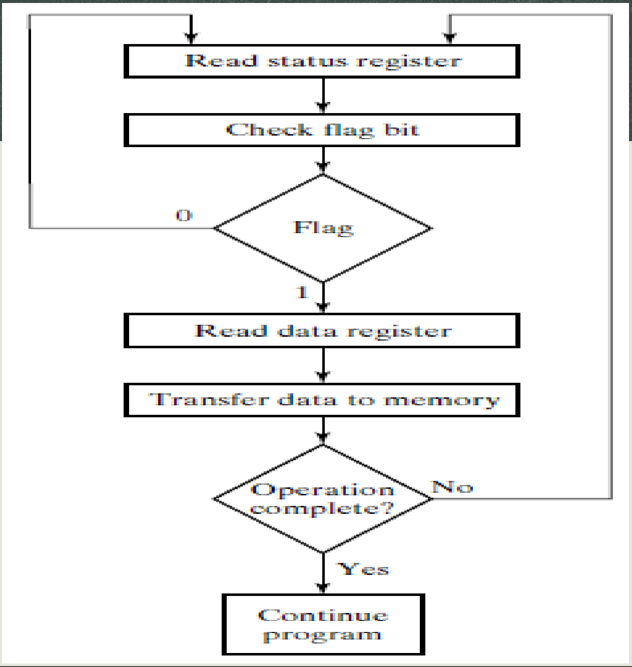
- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.
- In the program-controlled transfer, the CPU stays in a program loop called a busy-wait loop until the I/O unit indicates that it is ready for data transfer.
- This is a time-consuming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

❑ Modes of I/O Data Transfer

▪ **Example of Program-Controlled Transfer**



❑ Modes of I/O Data Transfer



❑ Modes of I/O Data Transfer

- **2. Interrupt-initiated data transfer:**
- An alternative to having the CPU constantly monitor the flag is to let the interface inform the computer when it is ready to transfer data. This mode of transfer uses the interrupt facility.
- 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 that the flag has been set.
- The CPU drops what it is doing to take care of the input or output transfer.

❑ Modes of I/O Data Transfer

- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.
- The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack or register, and then control branches to a service routine that processes the required I/O transfer.
- The way that the processor chooses the branch address of the service routine varies from one unit to another.
- In principle, there are two methods for accomplishing this: *vectored interrupt* and *nonvectored interrupt*.

❑ Modes of I/O Data Transfer

- **PRIORITY INTERRUPT**
- A typical computer has a number of I/O devices attached to it that are able to originate an interrupt request.
- The first task of the interrupt system is to identify the source of the interrupt.
- There is also the possibility that several sources will request service simultaneously.
- In this case, the system must decide which device to service first.

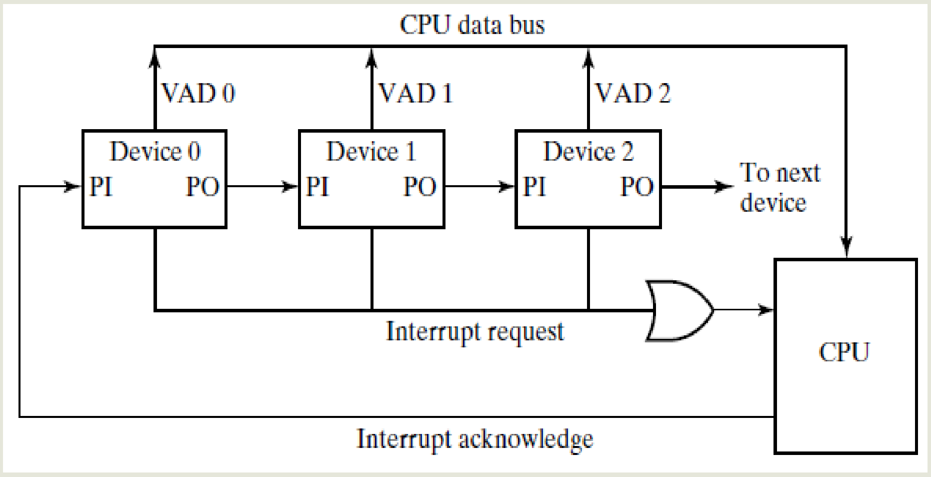
❑ Modes of I/O Data Transfer

- A priority interrupt system establishes a priority over the various interrupt sources to determine which interrupt request to service first when two or more are pending simultaneously.
- Devices with high-speed transfers such as hard drives are given high priority, and slow devices such as keyboards receive the lowest priority.
- When two devices interrupt the computer at the same time, the computer services the device with the higher priority first.

❑ Modes of I/O Data Transfer

- **Daisy Chain Priority**
- The *daisy chain* 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 devices of priority in descending order, down 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.

❑ Modes of I/O Data Transfer

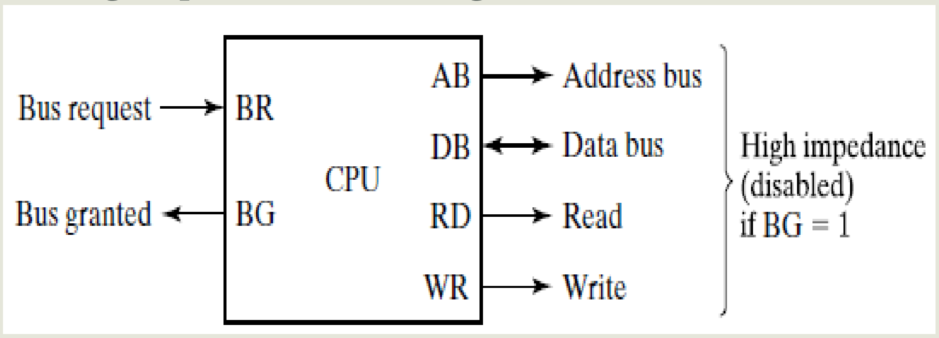


❑ Modes of I/O Data Transfer

- **DIRECT MEMORY ACCESS**
- In *direct memory access* (DMA), the interface unit 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 needing to be transferred and then proceeds to execute other tasks.
- When the transfer is made, the interface requests memory cycles through the memory bus.
- When the request is granted by the memory controller, the interface transfers the data directly into memory.

❑ Modes of I/O Data Transfer

- DMA may capture the buses in a number of ways. One common method extensively used in microprocessors is to disable the buses through special control signals.



❑ Modes of I/O Data Transfer

- Diagram shows two control signals in a 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 *BR* input is active, the CPU places the address bus, the data bus, and the read and write lines into a high-impedance state.
- After this is done, the CPU activates the bus granted (*BG*) output to inform the external DMA that it can take control of the buses.

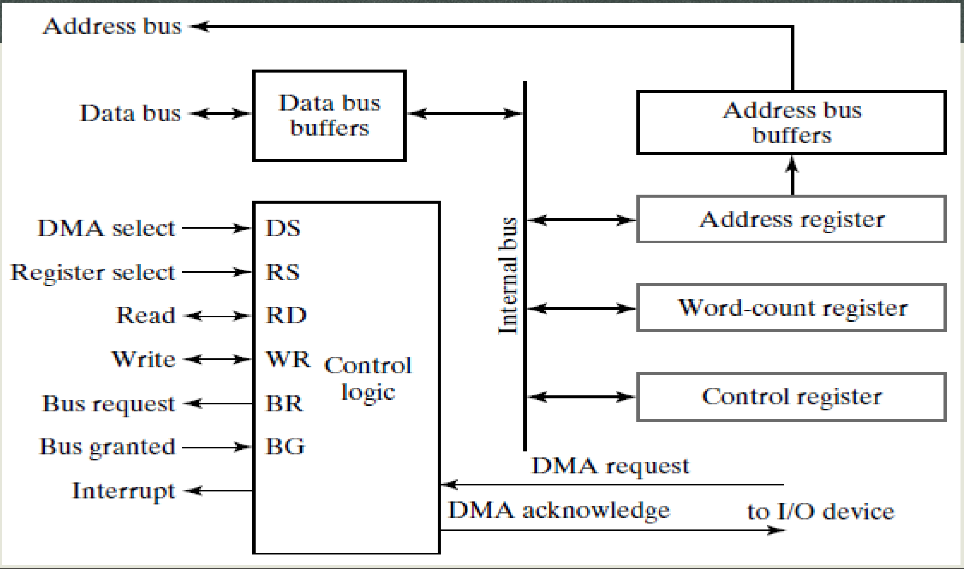
❑ Modes of I/O Data Transfer

- As long as the *BG* line is active, the CPU is unable to proceed with any operations requiring access to the buses.
- When the bus request input is reset by the DMA, the CPU returns to its normal operation, resets the *BG* output, and takes control of the buses.

❑ Modes of I/O Data Transfer

- **DMA Controller**
- The DMA controller needs the usual circuits of an interface to communicate with the CPU and the 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 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.

❑ Modes of I/O Data Transfer



❑ Modes of I/O Data Transfer

- The unit communicates with the CPU via the data bus and control lines.
- 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 granted) input is 0, the CPU can communicate with the DMA registers through the data bus to read from or write to those registers.

❑ Modes of I/O Data Transfer

- When *BG* = 1, the CPU has relinquished the buses, and the DMA can communicate directly with memory by specifying an address on the address bus and activating the *RD* or *WR* control.
- The DMA communicates with the external peripheral through the DMA request and DMA acknowledge lines by 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 of a word in memory.

❑ Modes of I/O Data Transfer

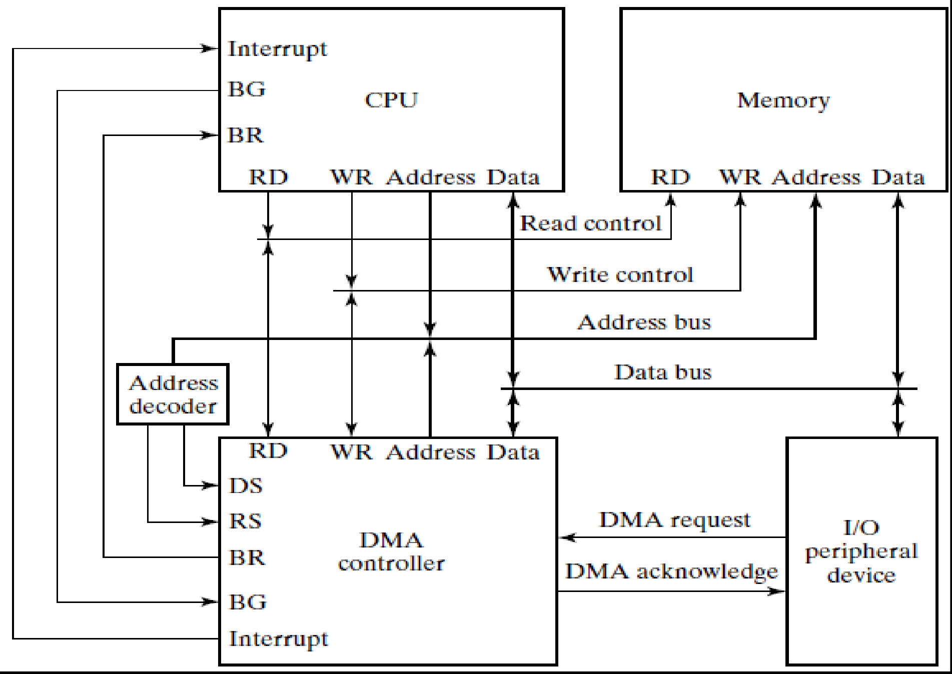
- The address register is incremented after each word is transferred to memory. The word-count register holds the number of words to be transferred.
- This register is decremented by one after each word transfer and internally tested for zero. The control register specifies the mode of transfer.
- All registers in the DMA appear to the CPU as I/O interface registers.
- Thus, the CPU can read from or write to the DMA registers under program control via the data bus.

❑ Modes of I/O Data Transfer

- The CPU initializes the DMA by sending the following information through the data bus:
- **1.** The starting address of the memory block in which data is available (for reading) or data is to be stored (for writing).
- **2.** The word count, which is the number of words in the memory block.
- **3.** A control bit to specify the mode of transfer, such as read or write.
- **4.** A control bit to start the DMA transfer.

❑ Modes of Data Tran

- DMA Transfer



Interrupt
BG
BR
CPU
RD
WR
Address
Data
Memory
RD
WR
Address
Data
Read control
Write control
Address bus
Data bus
Address decoder
RD
WR
Address
Data
DS
RS
BR
BG
Interrupt
DMA controller
DMA request
DMA acknowledge
I/O peripheral device