I/O Management

I/O Management

- I/O management is a **major component of operating system design and operation**
  - Important aspect of computer operation
  - **I/O devices vary greatly**
  - **Various methods to control them**
  - Performance management
  - New types of devices

I/O Management

- Ports, buses, device controllers **connect to various devices**
- **Device drivers** encapsulate device details
  - Present **uniform device-access interface** to I/O subsystem

I/O Hardware

- Incredible variety of I/O devices
  - **Storage/Machine readable**
    - **Suitable for communicating with electronic equipment**
    - **Disks,**
    - **Tapes**
    - Sensors
    - Controllers
    - **USB Keys**

I/O Hardware

- Incredible variety of I/O devices
  - **Transmission devices/Communication**
    - Suitable for communicating with remote devices
    - network cards
    - modems
  - **Human-interface/Human Readable**
    - Suitable for communicating with the user
    - Terminals i.e. Screen, video display
    - Keyboard,
    - Mouse,
    - Printers

I/O Hardware

- Incredible variety of I/O devices
  - More specialized devices –
    - Eg- Those involved in the steering of a military fighter jet or a space shuttle.
    - In these aircraft,
      - a human gives input to the flight computer via a joystick and foot pedals, and
      - the computer sends output commands that cause motors to move rudders, flaps, and thrusters.

I/O Hardware

Unit of transfer:

- Data may be transferred as a stream of bytes or characters (e.g., terminal I/O) or in larger blocks (e.g., disk I/O).

Data representation:

- Different data encoding schemes are used by different devices, including differences in character code and parity conventions.

Error conditions:

- The nature of errors, the way in which they are reported, their consequences, and the available range of responses differ widely from one device to another.

I/O Hardware

Types of I/O devices-

- Block oriented
- Stream oriented

I/O Hardware

Block-oriented device

- stores information in blocks that are usually of fixed size, and transfers are made one block at a time.
- Generally, it is possible to reference data by its block number.
- Disks and USB keys devices.

I/O Hardware

Stream-oriented device

- transfers data in and out as a stream of bytes, with no block structure.
- Terminals, printers, communications ports, mouse and other pointing devices, and most other devices that are not secondary storage

- How the devices are attached and how the software can control the hardware??

I/O Hardware

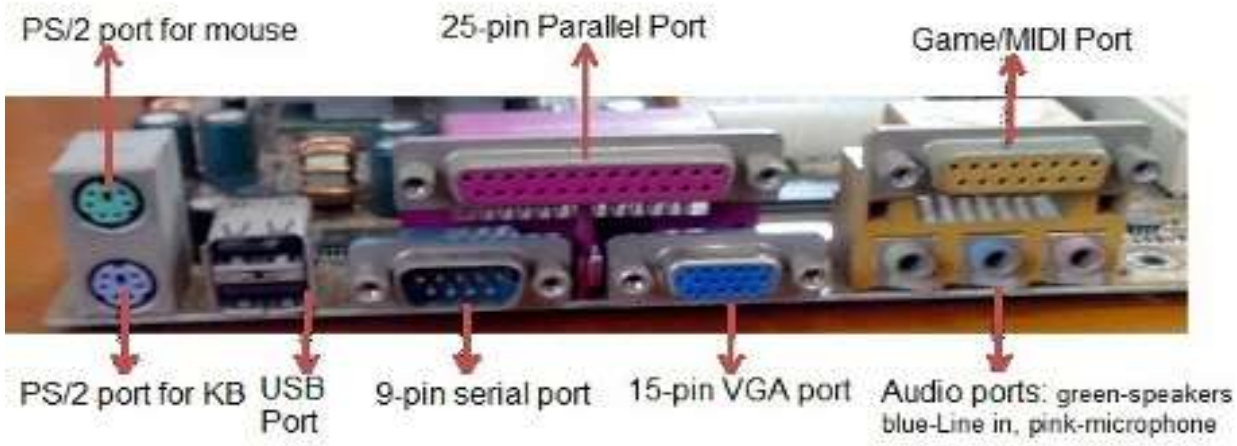
- Common concepts – signals from I/O devices interface with computer
  - **Port** – connection point for device
  - **Bus** - **daisy chain** or shared direct access
    - **PCI** bus common in PCs and servers, PCI Express (**PCIe**)
    - **expansion bus** connects relatively slow devices
  - **Controller** (**host adapter**) – electronics that operate port, bus, device
    - Sometimes **integrated**
    - Sometimes **separate** circuit board (host adapter)
    - Contains processor, microcode, private memory, bus controller, etc
      - Some talk to per-device controller with bus controller, microcode, memory, etc

I/O Hardware

- A device communicates with a computer system by sending signals over a cable or even through the air.

Port-

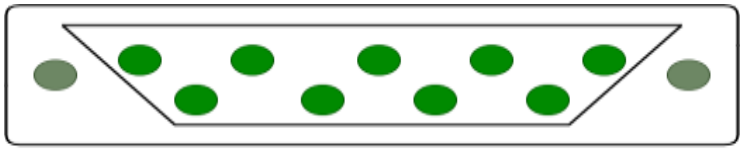
- A connection point via which the device communicates with the machine. Eg- a serial port.



I/O Hardware

Serial Port:

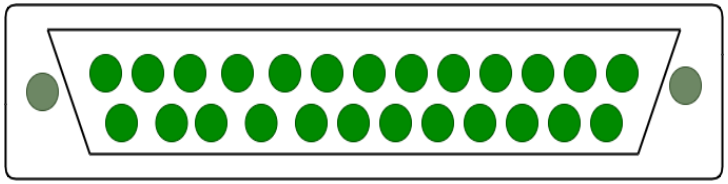
- An interface that is used for connecting the serial lines to attain the serial communication.
- These ports can dock a 9-pin D-shaped connector that connects to the transmission line, is called DB-9 connectors.



DB-9

Parallel Port:

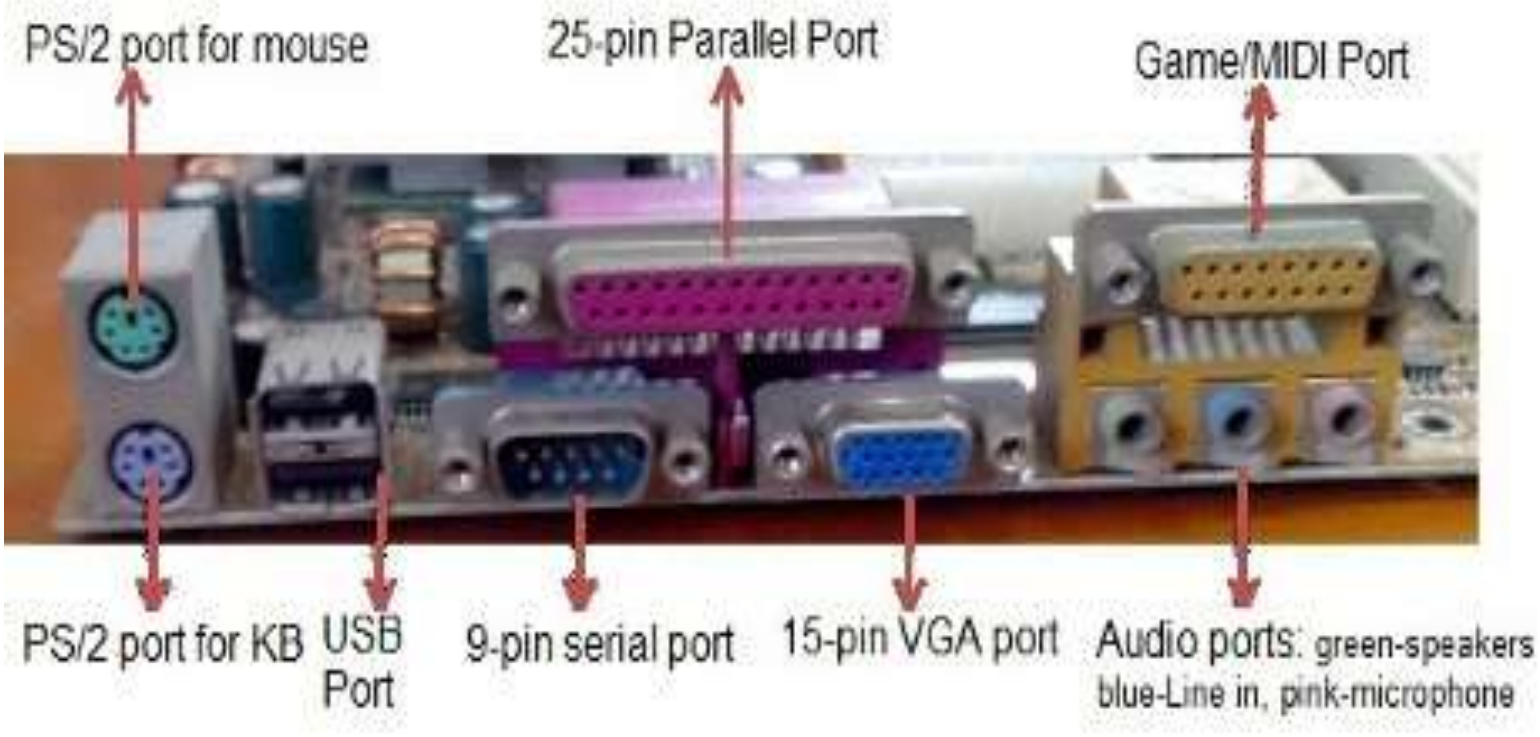
- Unlike serial port, a parallel port can move a set of 8 bits at a time on eight different wires. That's why it is faster in comparison to the serial communication.
- Unlike serial port, It uses a 25 pin connector that is called DB-25 connector.



DB-25

I/O Hardware

Check Serial Port and Parallel Port



Difference between Serial port and Parallel Ports:

S.NO	SERIAL PORT	PARALLEL PORT
1.	Serial port is used to achieve serial transmission.	While parallel port is used to achieve parallel transmission.
2.	The transmission speed of serial port is comparatively low as compared to parallel port transmission speed.	While transmission speed of parallel port is higher than serial port transmission speed.
3.	In serial port communication less number of wires are used.	While in parallel port communication more number of wires are used as compared to serial port.
4.	A serial port is capable of delivering the single stream of data.	While a parallel port is capable of delivering multiple streams of data.
5.	Serial port send a bit after another bite at a time.	While parallel ports send multiple bits at once.
6.	In a serial port, male ports are involved.	While in a parallel port, female ports are involved.
7.	Serial ports are typically implemented in modems, connecting devices, security cameras and controllers.	Parallel ports are typically implemented in zip-drives, printers, hard drives, CD-ROM drives, etc.

I/O Hardware

Bus-

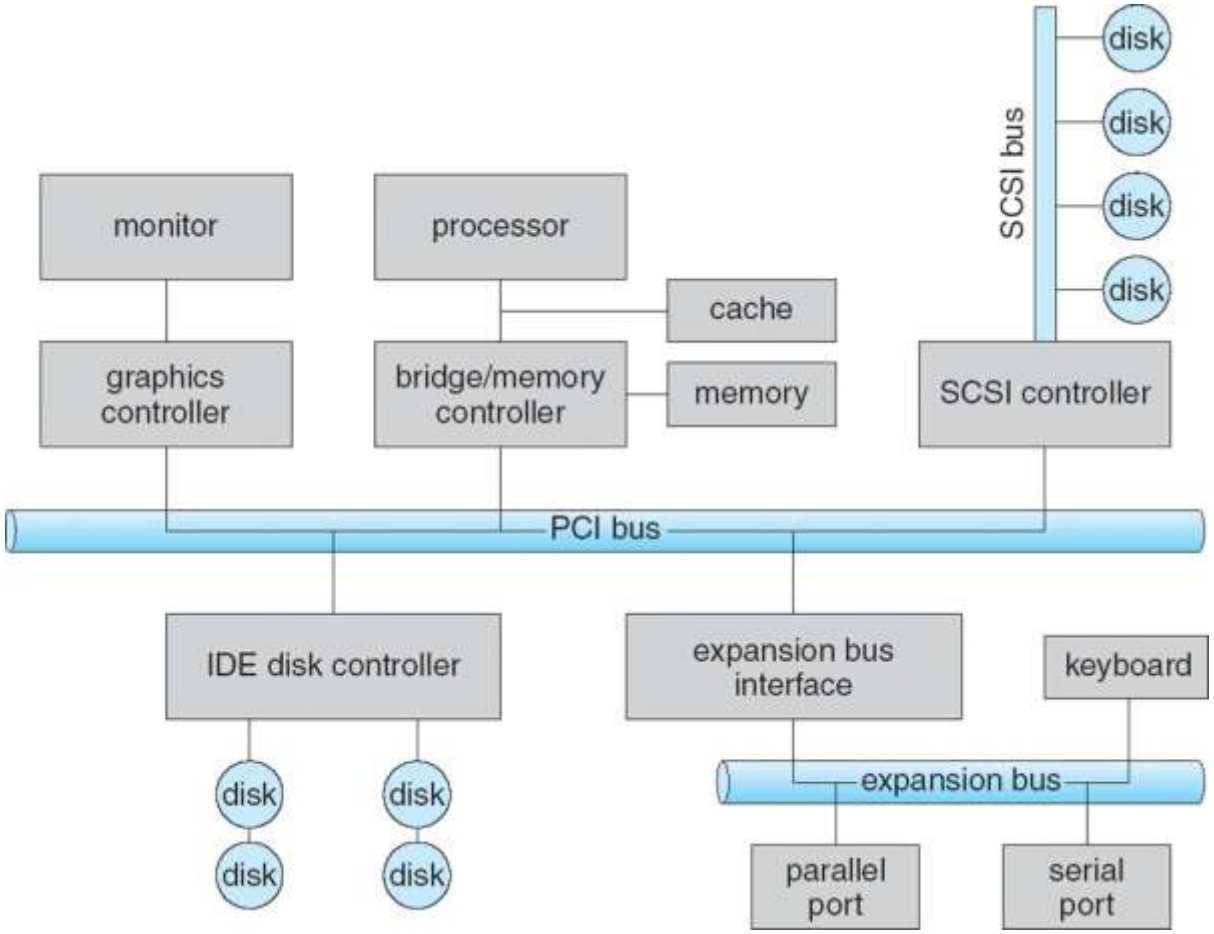
- If devices use a common set of wires, the connection is called a *bus.*
- **Set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.**
- The messages are conveyed by patterns of electrical voltages applied to the wires with defined timings.

I/O Hardware

- **Daisy Chain-**
- When device *A* has a cable that plugs into device *B*, and
- Device *B* has a cable that plugs into device C,
- Device C plugs into a port on the computer,

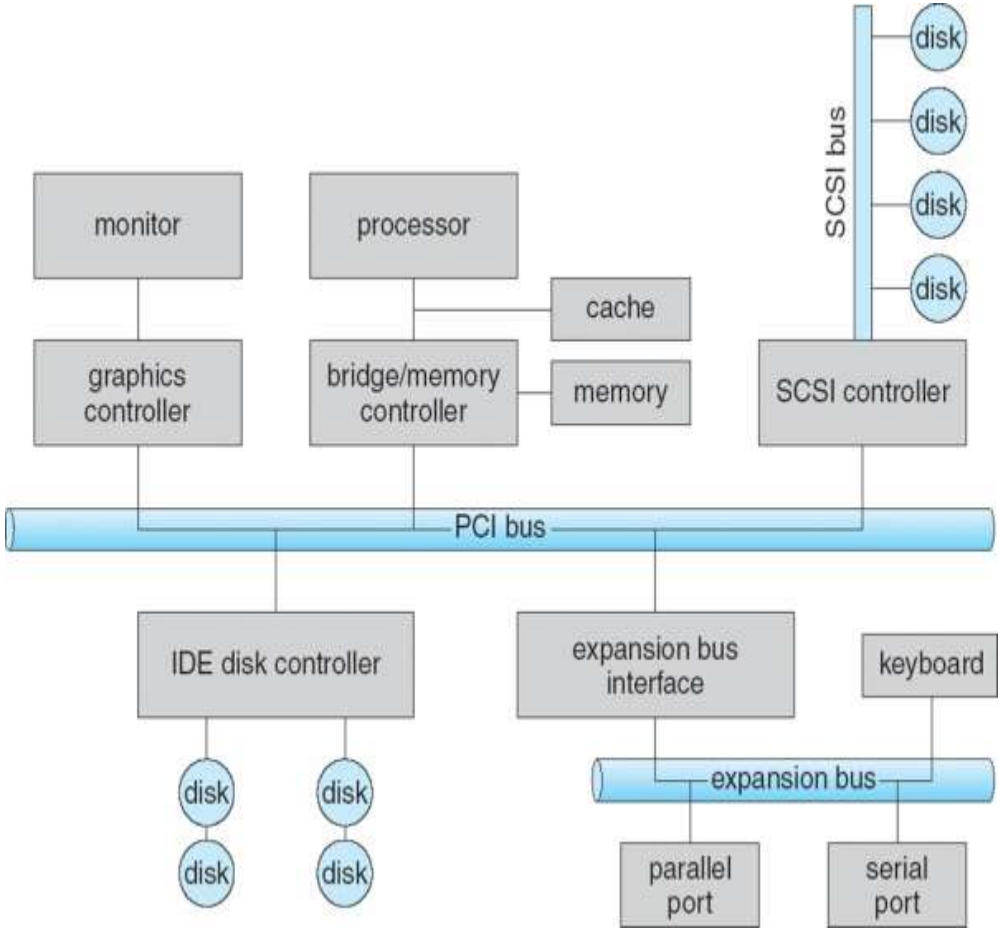
- Daisy Chain usually operates as a bus.

A Typical PC Bus Structure



A Typical PC Bus Structure

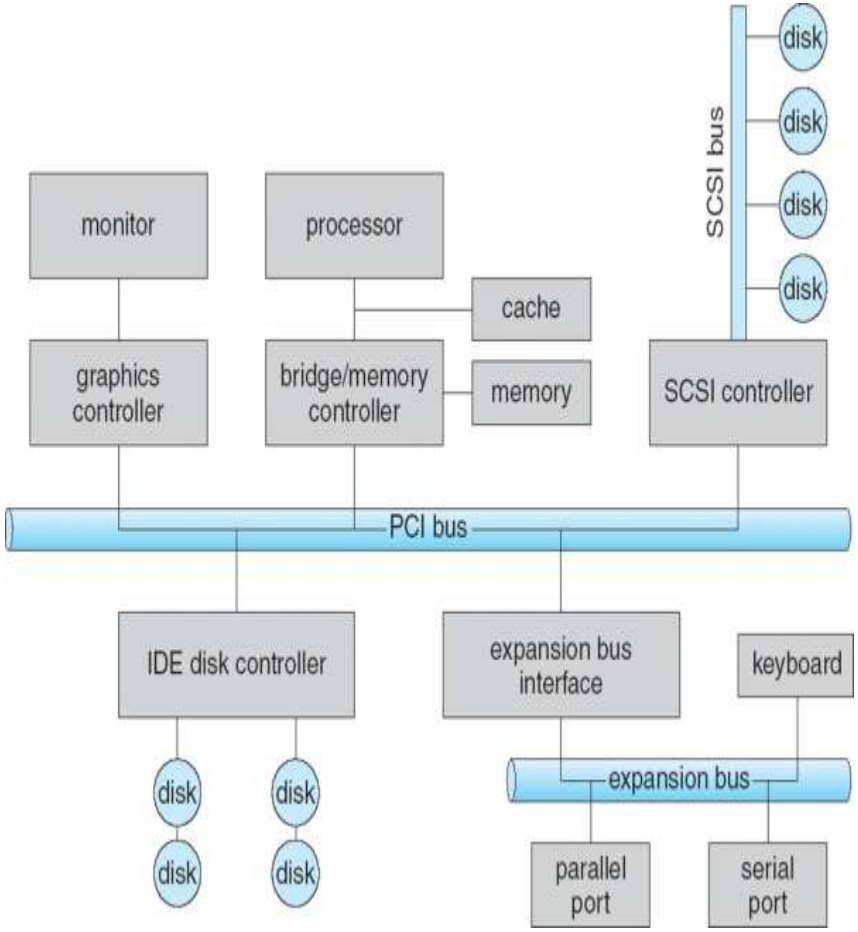
- "Peripheral Component Interconnect." hardware bus
- **Common PC system bus**
  - connects the processor-memory subsystem to the fast devices
- **Expansion Bus**
  - connects relatively slow devices, such as the keyboard and serial and USB ports.



A Typical PC Bus Structure

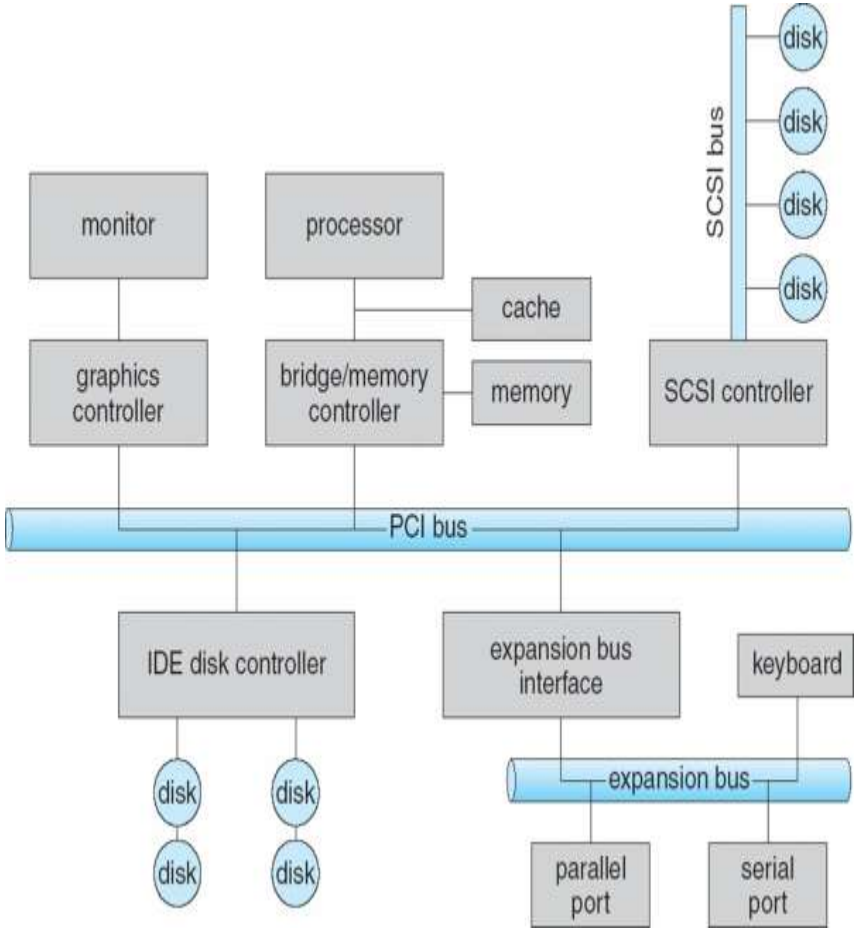
- **Top-right portion** ,
  - **four disks** are connected together
  - **on a SCSI bus** plugged into a SCSI controller.

- ***Small Computer System Interface (SCSI)***
  - set of **parallel interface standards** that allows PCs to communicate with peripheral hardware faster



A Typical PC Bus Structure

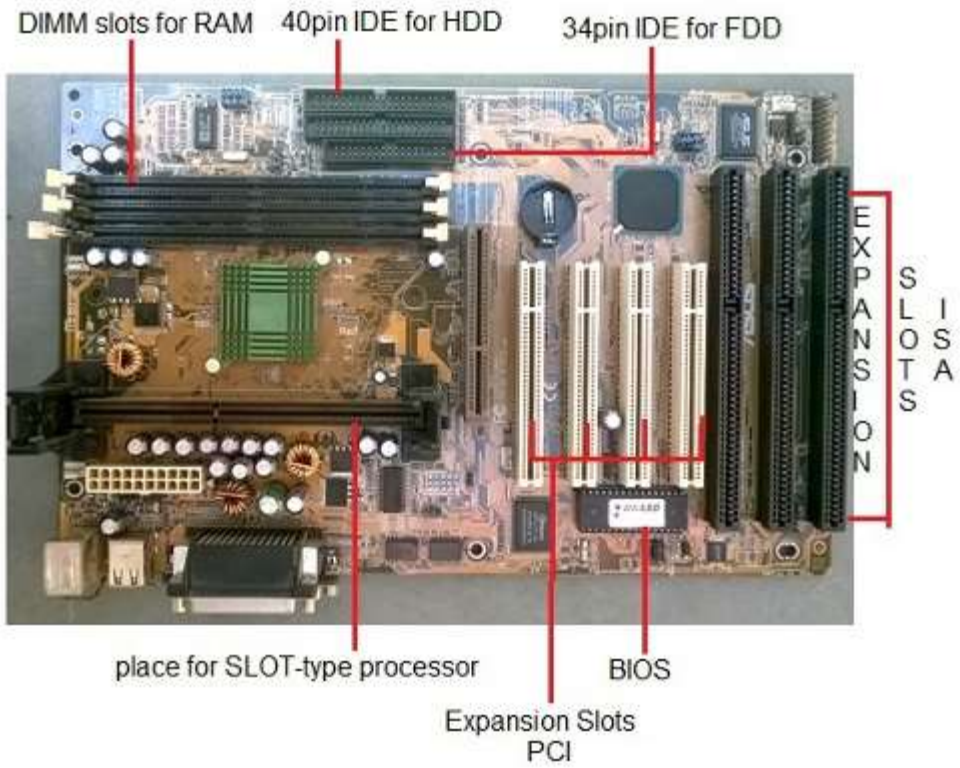
- Other common buses used to interconnect main parts of a computer
  - PCI-x with throughput up to 4.3 GB;
  - PCI Express (PCie), with throughput up to 16 GB
  - Hypertransport with throughput up to 20 GB.



Expansion Slots

ISA slots.

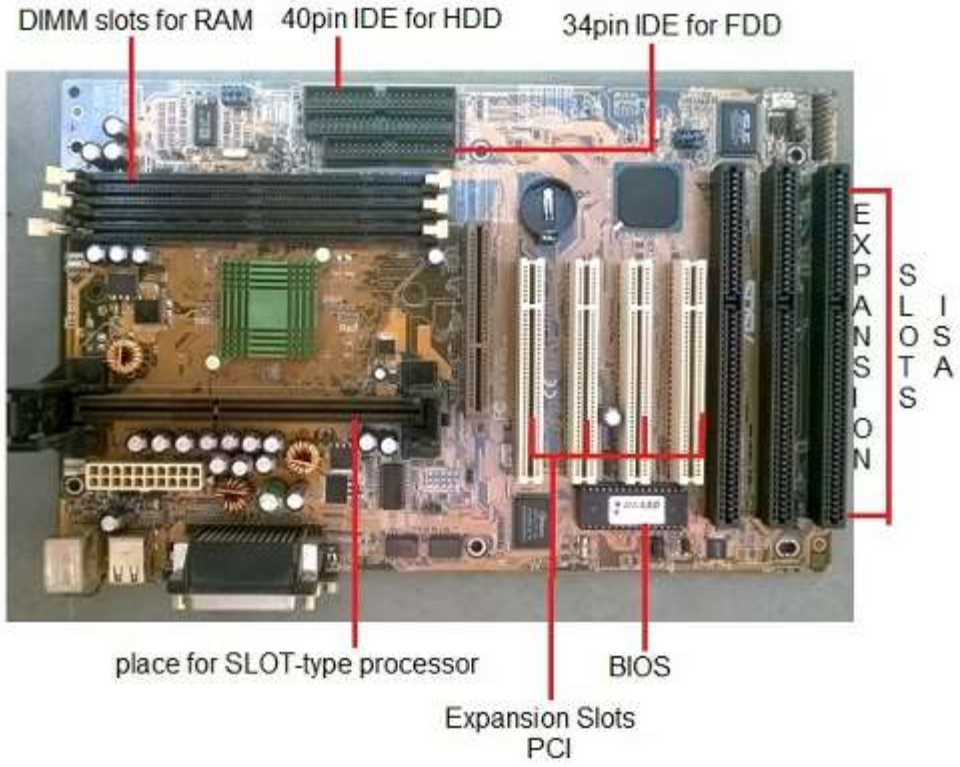
- Oldest expansion slots in the history of motherboards.
- Used in AT boards and are identified by black color.
- Conventional display cards or sound cards were installed in these slots.
- The full form of ISA is **Industry Standard Architecture** and is a 16-bit bus.



Expansion Slots

PCI Slots.

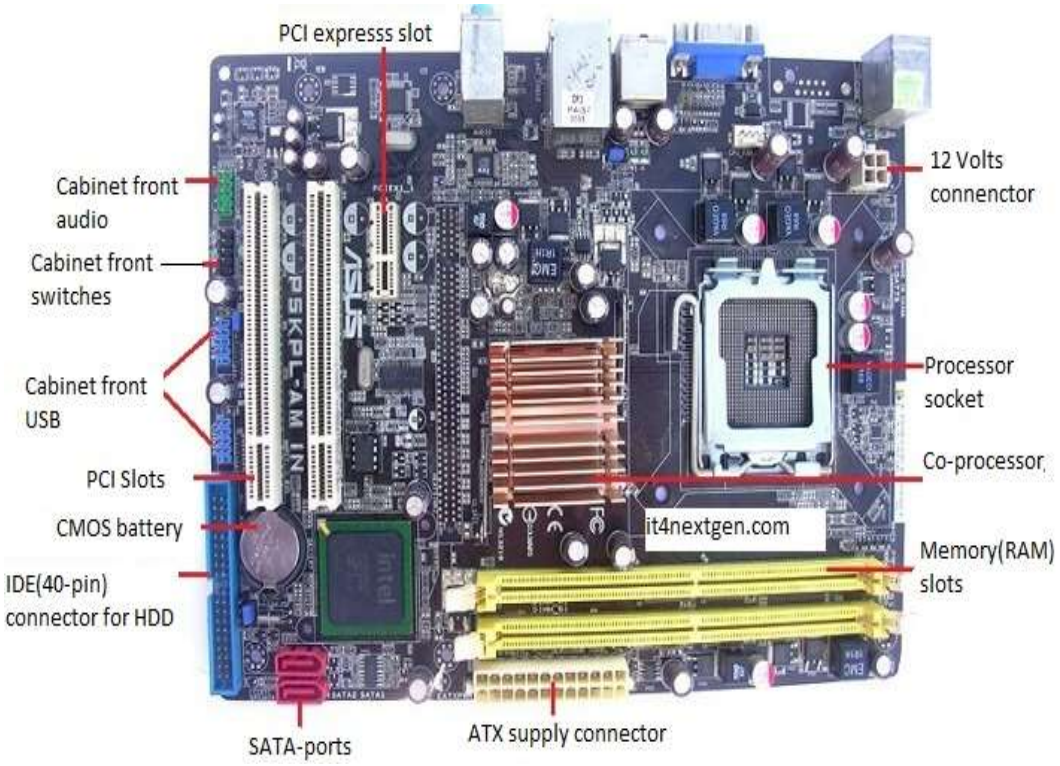
- **PCI (Peripheral Component Interconnect) Slot:** Supports peripherals like sound cards, DVD decoders, and graphic accelerators
- There are usually anywhere from 1 to 6 PCI slots available on the motherboard.
- The PCI supports 64-bit high-speed bus.



Expansion Slots

PCI express.

- Also known as PCIe,
- Latest and the fastest component of the motherboard to support add-on cards.
- It supports full duplex serial bus.



11.2 ORGANIZATION OF THE I/O FUNCTION

Three techniques for performing I/O:

- **Programmed I/O**
- **Interrupt-driven I/O**
- **Direct memory access (DMA)**

11.2 ORGANIZATION OF THE I/O FUNCTION

Programmed I/O:

- The processor issues an I/O command, on behalf of a process, to an I/O module;
- that process then busy waits for the operation to be completed before proceeding.

11.2 ORGANIZATION OF THE I/O FUNCTION

Interrupt-driven I/O:

- The processor issues an I/O command on behalf of a process.
- There are then two possibilities.
- If the I/O instruction from the process is non blocking,
  - the processor continues to execute instructions from the process that issued the I/O command.
- If the I/O instruction is blocking,
  - then the next instruction that the processor executes is from the OS,
  - which will put the current process in a blocked state Interrupt the current process and schedule another process.

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The processor sends a request for the transfer of a block of data to the DMA module and is interrupted only after the entire block has been transferred.
- A DMA module controls the exchange of data between main memory and an I/O module.

- DMA is the dominant form of transfer that must be supported by the operating system.

11.2 ORGANIZATION OF THE I/O FUNCTION

	No Interrupts	Use of Interrupts
I/O-to-Memory Transfer through Processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-Memory Transfer		Direct memory access (DMA)

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The DMA unit is capable of mimicking the processor
- Taking control of the system bus just like a processor
- It needs to do this to transfer data to and from memory over the system bus.

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The DMA technique works as follows.
- When the processor wishes to read or write a block of data, it issues a command to the DMA module by sending to the DMA module some information:

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The information is:
  - Whether a read or write is requested,
  - Using the read or write control line between the processor and the DMA module
  - The address of the I/O device involved, communicated on the data lines

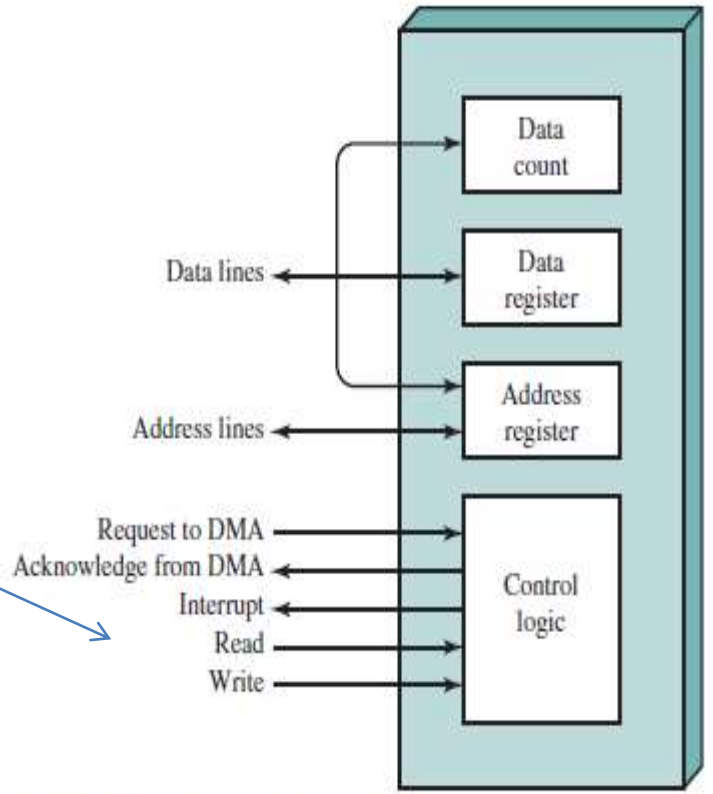


Figure 11.2 Typical DMA Block Diagram

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The starting location in memory to read from or write to, communicated on the data lines and stored by the DMA module in its address register
- The number of words to be read or written, again communicated via the data lines and stored in the data count register

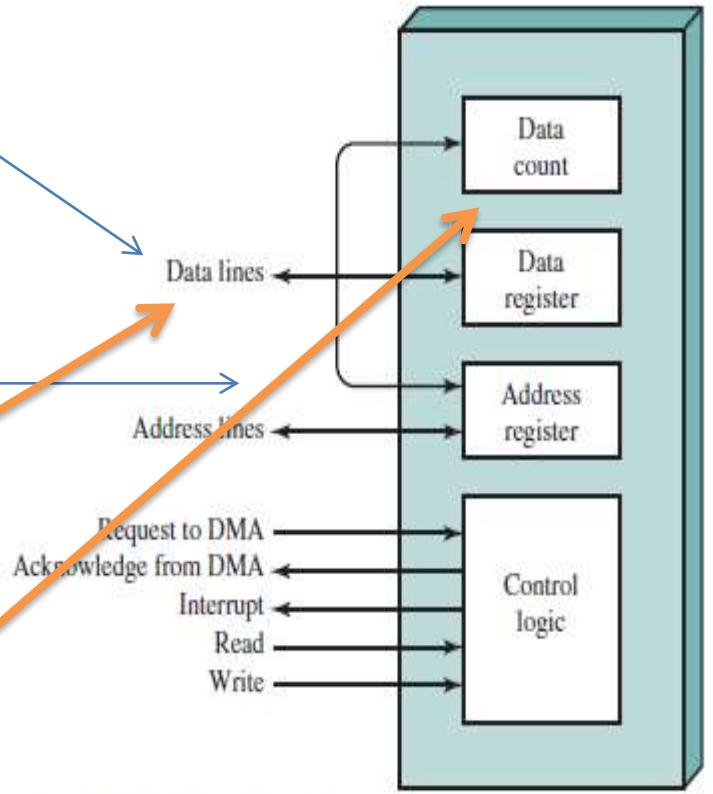


Figure 11.2 Typical DMA Block Diagram

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The processor then continues with other work.
  - It has delegated this I/O operation to the DMA module

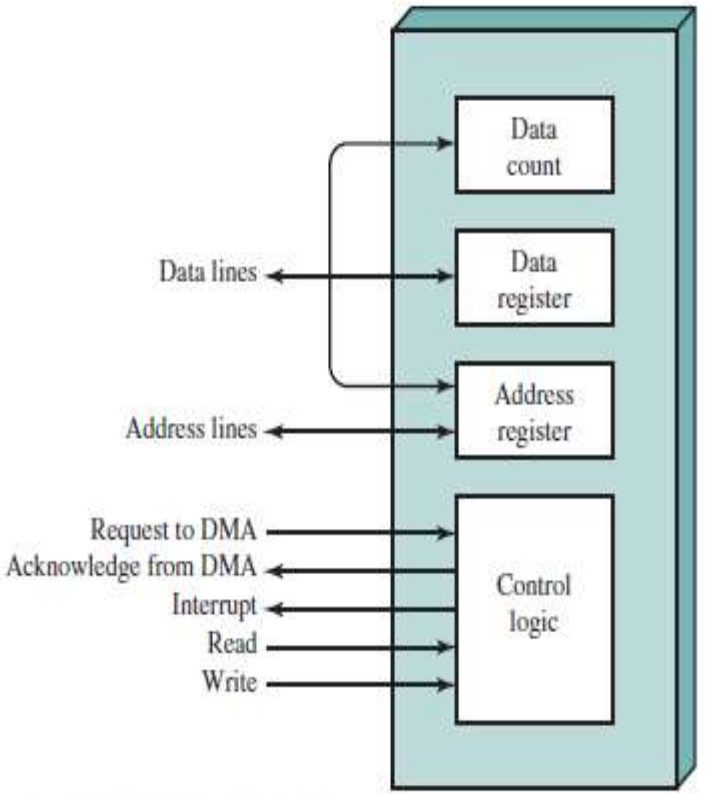


Figure 11.2 Typical DMA Block Diagram

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- The DMA module transfers the entire block of data,
  - one word at a time,
  - directly to or from memory,
  - **without going through the processor**

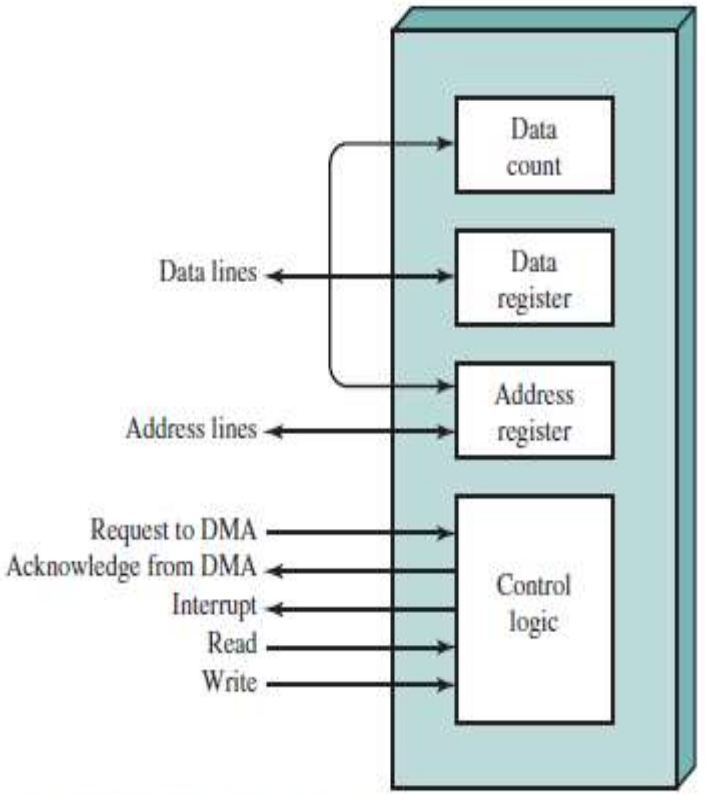


Figure 11.2 Typical DMA Block Diagram

11.2 ORGANIZATION OF THE I/O FUNCTION

Direct memory access (DMA):

- When the transfer is complete,
- the DMA module sends an interrupt signal to the processor.
- The processor is involved only at the beginning and end of the transfer

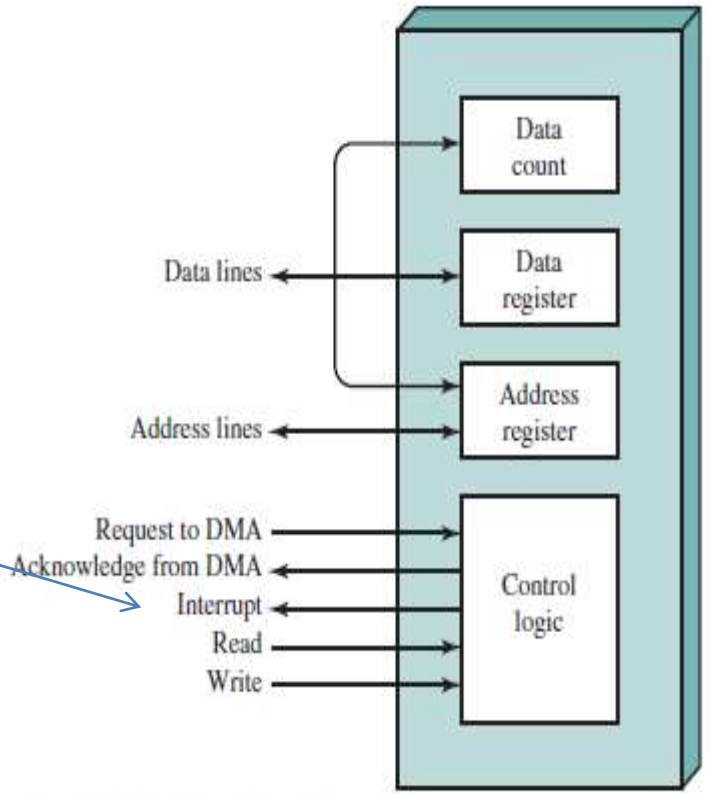


Figure 11.2 Typical DMA Block Diagram

Direct Memory Access (DMA)

- DMA Controller is a hardware device that allows I/O devices to directly access memory with less participation of the processor.

- For The data transfer between a storage media such as magnetic disk and memory unit. The peripherals can directly communicate with each other using the memory buses, removing the intervention of the CPU. This type of data transfer technique is known as DMA or direct memory access.

Direct Memory Access (DMA)

- During DMA, the CPU is idle and it has no control over the memory buses. The DMA controller takes over the buses to manage the transfer directly between the I/O devices and the memory unit.

Direct Memory Access (DMA)

GATE | GATE-CS-2016 (Set 1) | Question 41

The size of the data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29,154 kilobytes from disk to main memory. The memory is byte addressable. The minimum number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to main memory is _____

(A) 3644
(B) 3645
(C) 456
(D) 1823

Direct Memory Access (DMA)

GATE | GATE-CS-2016 (Set 1) | Question 41

The size of the data count register of a DMA controller is 16 bits. The processor needs to transfer a file of 29,154 kilobytes from disk to main memory. The memory is byte addressable. The minimum number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to main memory is _____

(A) 3644
(B) 3645
(C) 456
(D) 1823

Answer: (C)

Explanation: Size of data count register of the DMA controller = 16 bits
Data that can be transferred in one go = 2^{16} bytes = 64 kilobytes
File size to be transferred = 29154 kilobytes
So, number of times the DMA controller needs to get the control of the system bus from the processor to transfer the file from the disk to main memory = ceil(29154/64) = 456

OPERATING SYSTEM DESIGN ISSUES FOR I/O

Design Objectives

Two objectives are paramount in designing the I/O facility:

- Efficiency
- Generality

OPERATING SYSTEM DESIGN ISSUES

Efficiency

- Most I/O devices are **extremely slow compared with main memory and the processor.**

- Solution??

OPERATING SYSTEM DESIGN ISSUES

Efficiency

- Soln=M**ultiprogramming,**
- Allow **some processes to be waiting** on I/O operations while **another process is executing**.
- However, even with the vast size of main memory in today's machines, it will still often be the case that **I/O is not keeping up with the activities of the processor.**

OPERATING SYSTEM DESIGN ISSUES

Generality

- In the interests of **simplicity** and freedom from error, it is desirable to **handle all devices in a uniform manner.**
- Uniformity=
  - The way in which **processes view I/O** devices
  - The way in which the **OS manages I/O** devices and operations.
- Due to **diversity** of device characteristics, it is **difficult** to achieve **true generality**

OPERATING SYSTEM DESIGN ISSUES

Generality

Soln=

- Use a **hierarchical, modular approach** to the design of the I/O function.
- This approach **hides details** of device I/O in lower-level routines so that
- **User processes and Upper levels** of the OS see devices in terms of **general functions**, such as
  - read,
  - write,
  - open,
  - close,
  - lock,
  - unlock

Logical Structure of the I/O Function

Hierarchical Approach-
Leads to an organization of the OS into a series of layers

- **Each layer performs a related subset of the functions required of the OS**
- **It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions.**
- **It provides services to the next higher layer.**

Logical Structure of the I/O Function

Hierarchical Approach-

- Ideally, the layers should be defined so that changes in one layer do not require changes in other layers.
- Decomposed one problem into a number of more manageable sub problems.

Logical Structure of the I/O Function

- The Hierarchy/Logical Structure for I/O
- The details of the organization will depend on type of device and the application
- Most operating systems approach I/O in **approximately this way.**

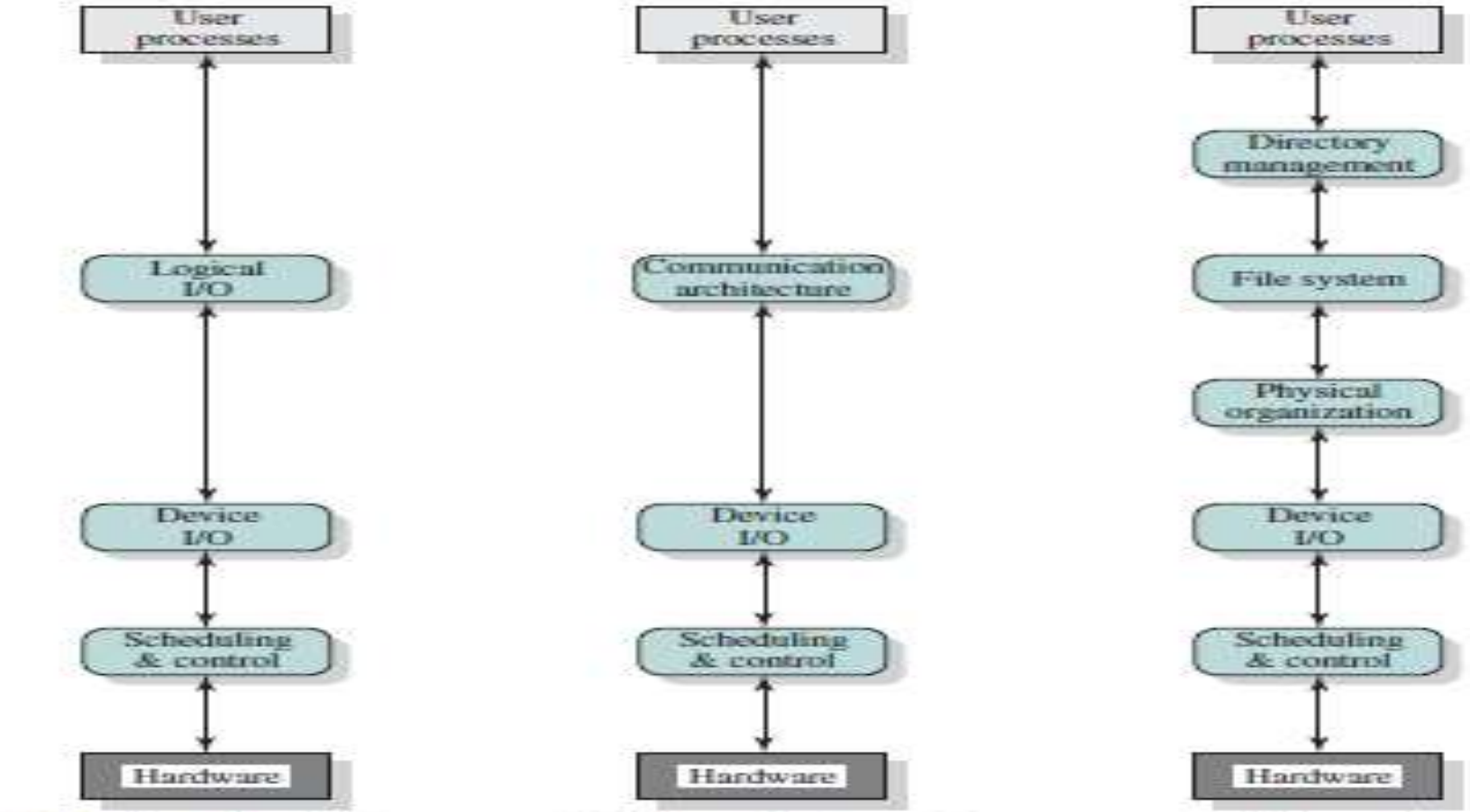


Figure 11.4 A Model of I/O Organization

Logical Structure of the I/O Function

Logical I/O:

- Deals with the **device as a logical resource**
- **Not concerned** with the details of actually **controlling the device.**

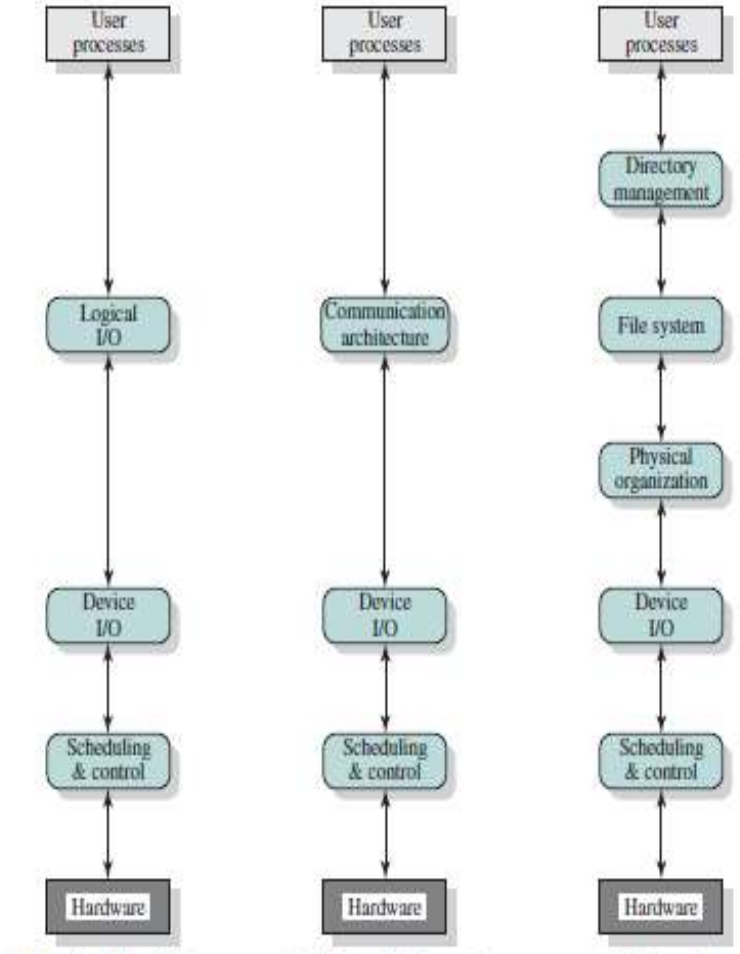


Figure 11.4 A Model of I/O Organization

Logical Structure of the I/O Function

Logical I/O:

- Concerned with managing general I/O functions on behalf of user processes,
  - allowing them to deal with the device in terms of **a device identifier**
  - **simple commands such as open, close, read, and write**

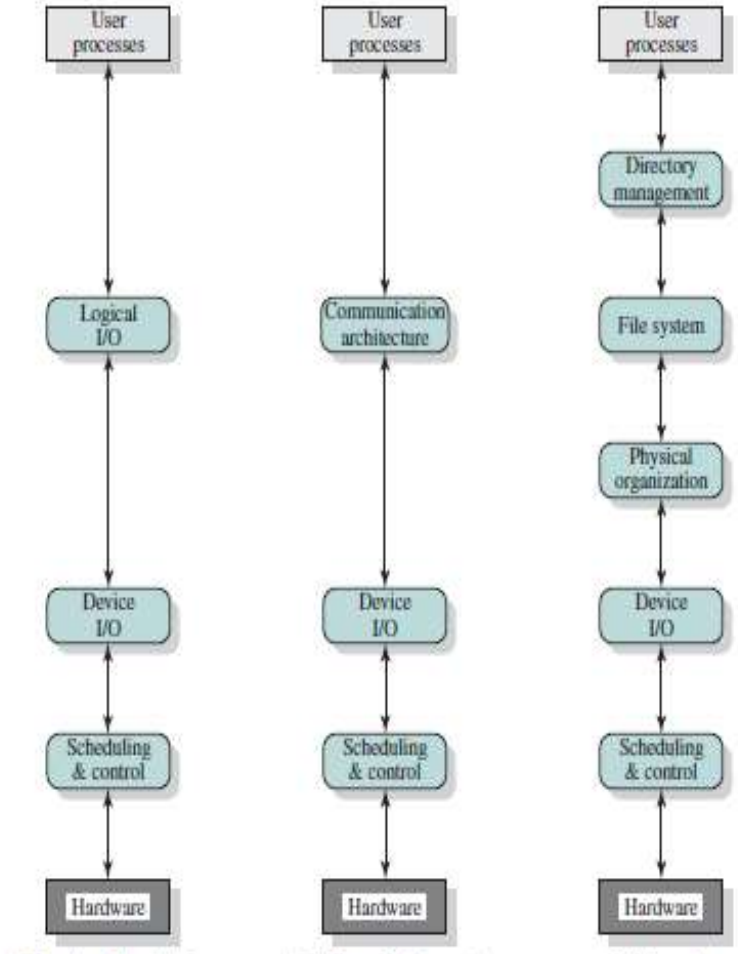


Figure 11.4 A Model of I/O Organization

Logical Structure of the I/O Function

Device I/O:

- The requested **operations** and data (buffered characters, records, etc.) **are converted** into
  - **appropriate sequences of I/O instructions**,
  - channel commands,
  - **controller orders**

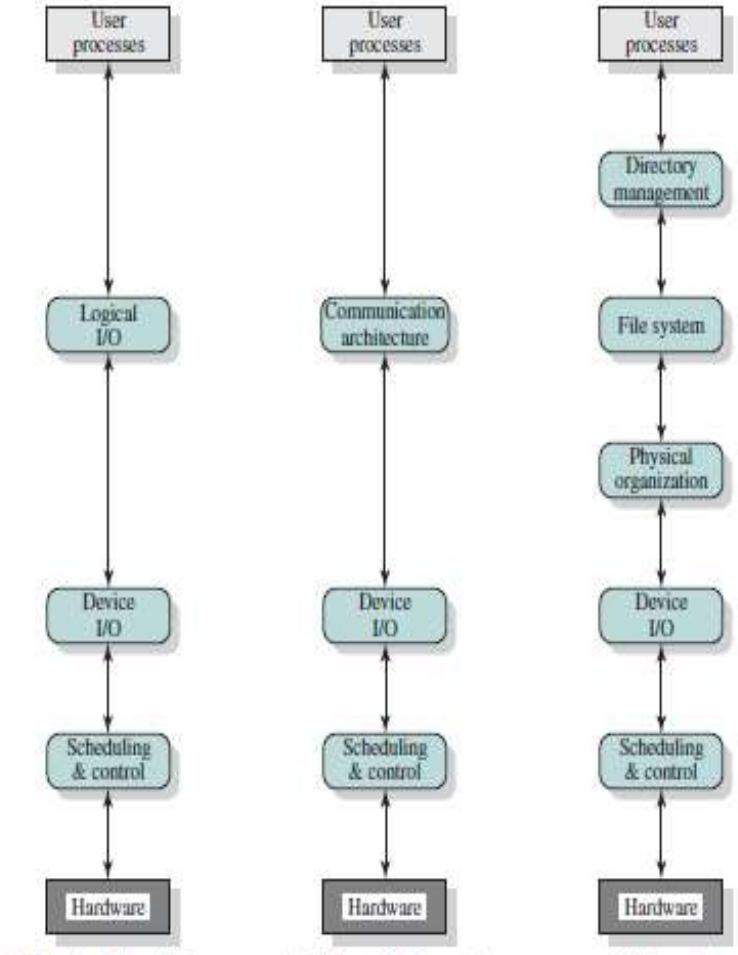
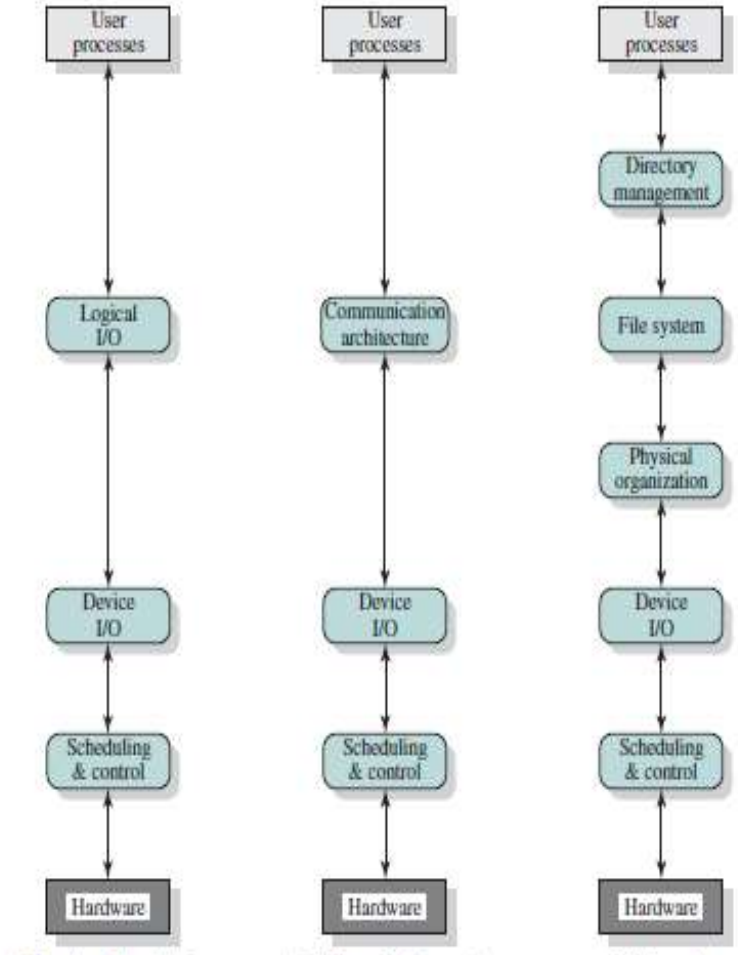


Figure 11.4 A Model of I/O Organization

Logical Structure of the I/O Function

Device I/O:

- **Buffering techniques may be used to improve utilization**



(a) Local peripheral device (b) Communications port (c) File system

Figure 11.4 A Model of I/O Organization

Logical Structure of the I/O Function

Scheduling and control:

- The actual **queueing and scheduling** of I/O operations occurs at this layer, as well as the control of the operations.
- **Interrupts are handled**

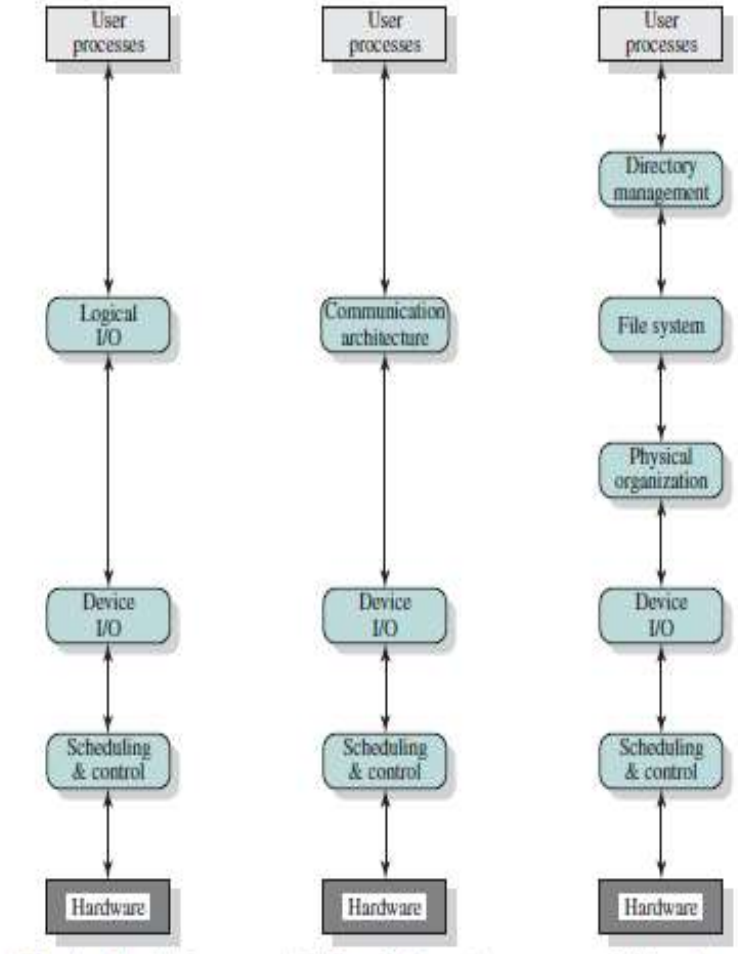


Figure 11.4 A Model of I/O Organization

Logical Structure of the I/O Function

Scheduling and control:

- **I/O status** is collected and reported
- Layer of software that **actually interacts with the I/O module** and hence the device hardware

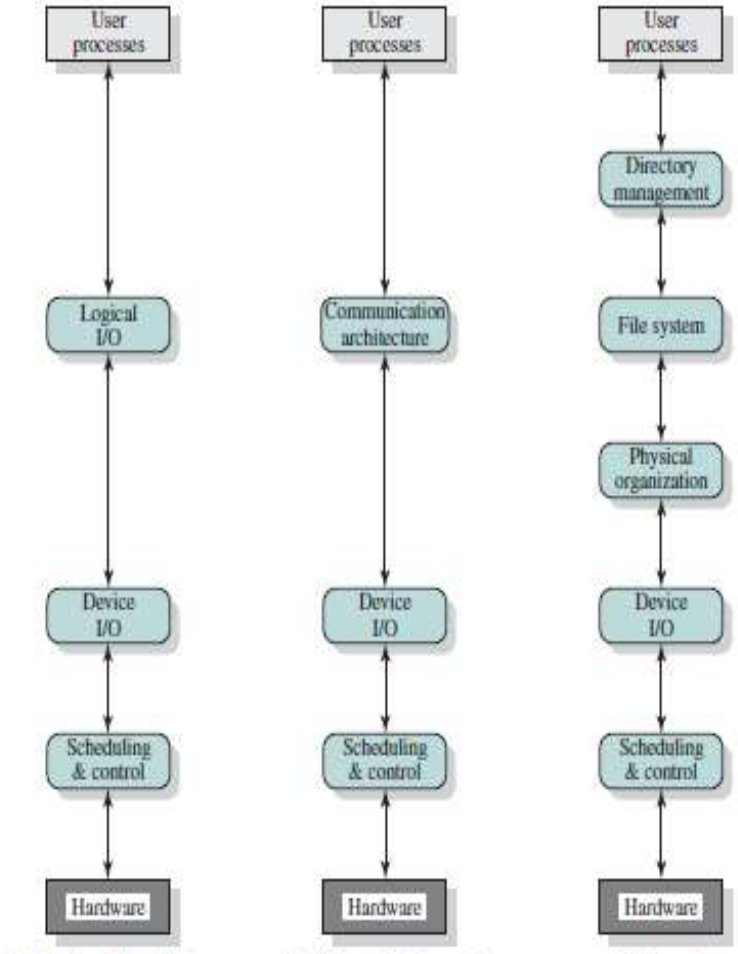


Figure 11.4 A Model of I/O Organization

I/O BUFFERING

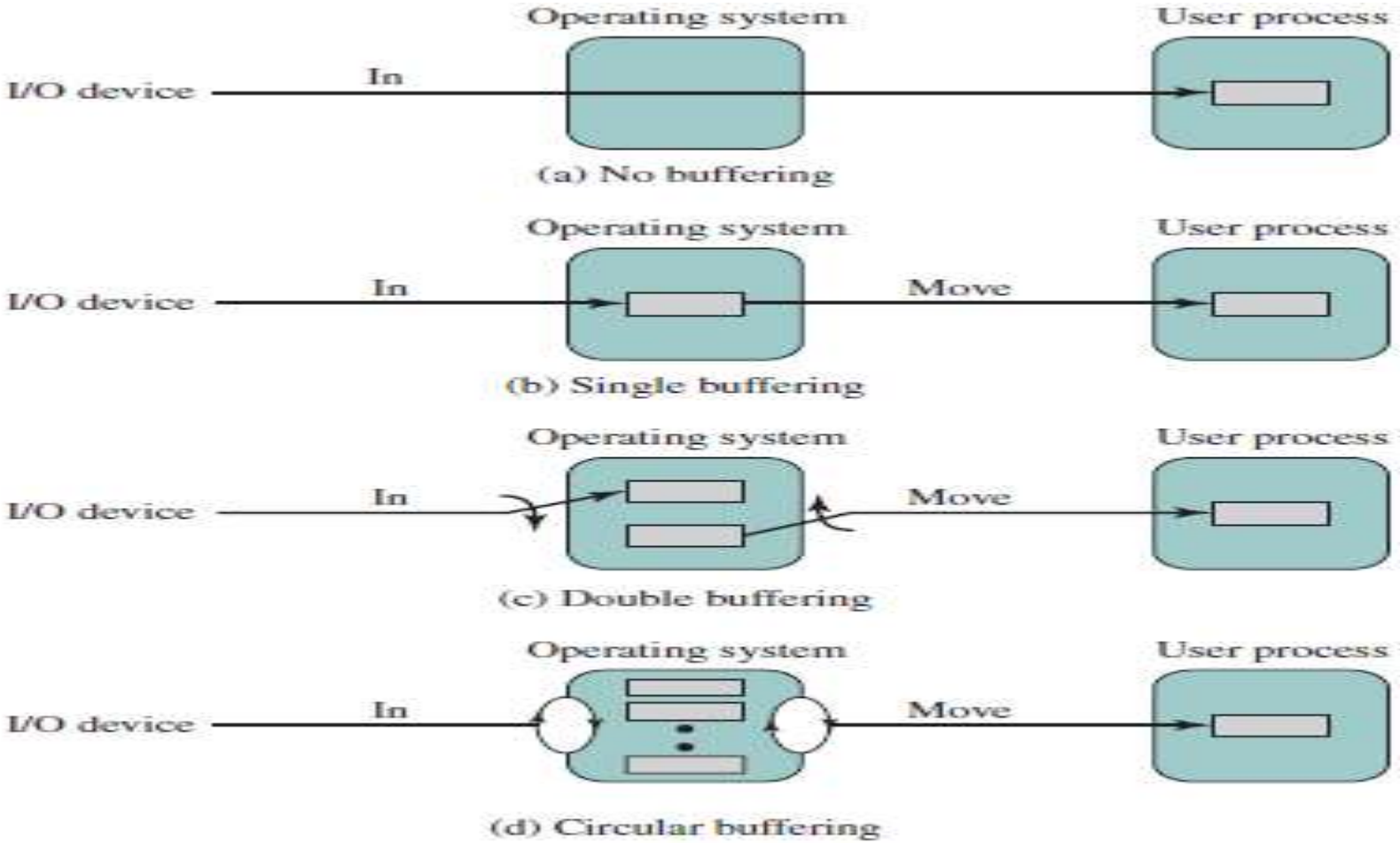
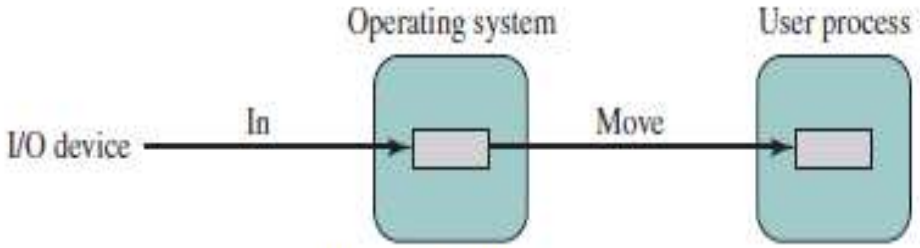


Figure 11.5 I/O Buffering Schemes (Input)

I/O BUFFERING

Single Buffer

- The simplest type of support that the OS can provide
- When a user process issues an I/O request,
  - the OS assigns a buffer in the system portion of main memory to the operation.

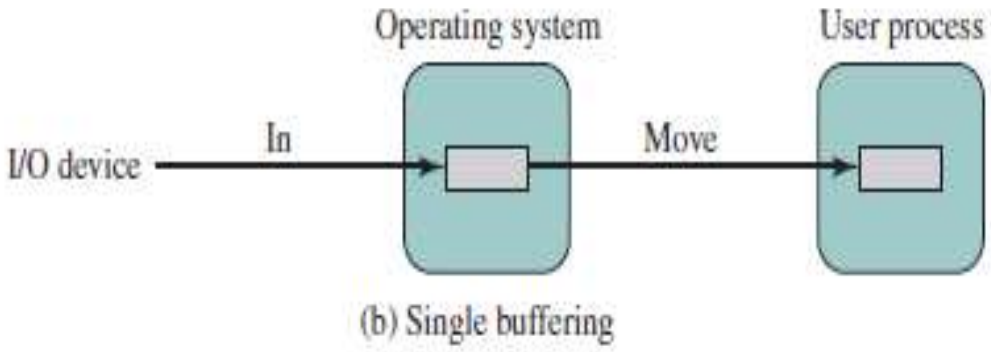


(b) Single buffering

I/O BUFFERING

Single Buffer

- Input transfers are made to the system buffer.
- When the transfer is complete, the process moves the block into user space and immediately requests another block.
- It is called reading ahead, or anticipated input;

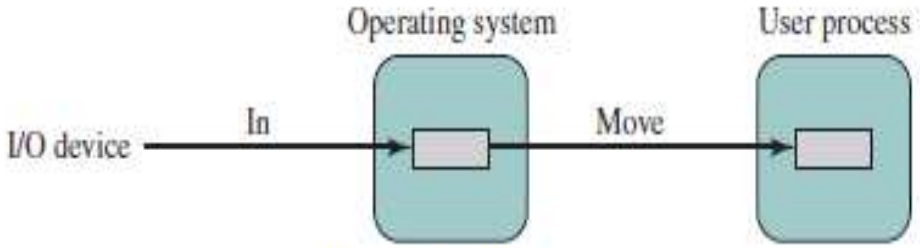


(b) Single buffering

I/O BUFFERING

Single Buffer

- Done in the expectation that the block will eventually be needed.
- This approach provides a speedup compared to no buffering.

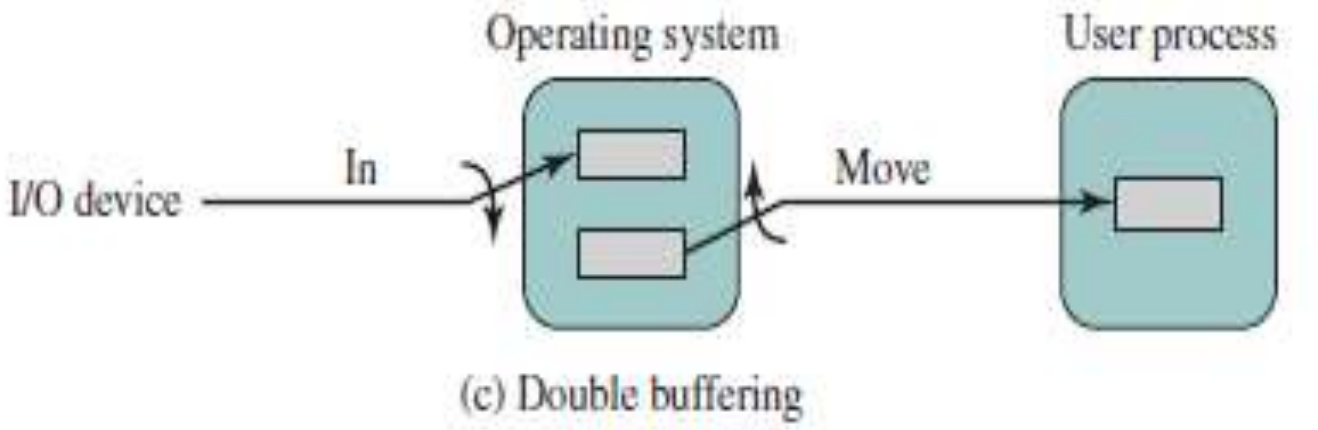


(b) Single buffering

I/O BUFFERING

Double Buffer

- Assigning two system buffers to the operation
  - A process now transfers data to (or from) one buffer
  - while the operating system empties (or fills) the other.
- Known as **buffer swapping** .



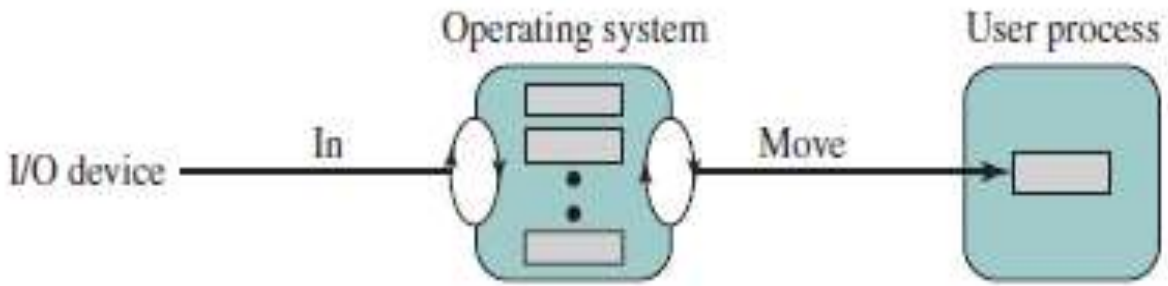
(c) Double buffering

I/O BUFFERING

Circular Buffer

- Double buffering may be **inadequate** if the process performs **rapid bursts of I/O**

- In this case, the problem can often be **alleviated by using more than two buffers.**

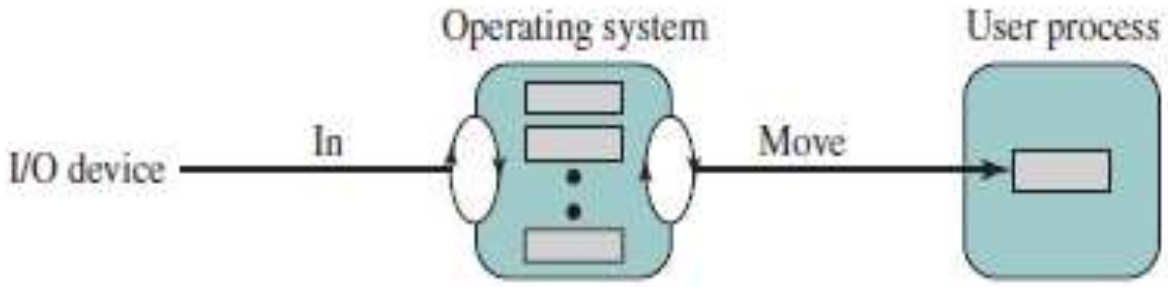


(d) Circular buffering

I/O BUFFERING

Circular Buffer

- When more than two buffers are used,
- the collection of buffers is referred to as circular buffer,
  - with each individual buffer being one unit in the circular buffer.
  - This is simply the bounded-buffer producer/consumer model



(d) Circular buffering

I/O BUFFERING

The Utility of Buffering

- Buffering is a technique that **smoothes out peaks in I/O demand.**

- No amount of buffering will allow an I/O device to keep pace with a process indefinitely
  - when the **average demand of the process is greater than the I/O device can service**
  - **Why??**

I/O BUFFERING

- Even with multiple buffers,
  - all of the buffers will eventually fill up
  - the process will have to wait after processing each chunk of data.
- In a multiprogramming environment,
  - Variety of I/O activity
  - Variety of process activity to service,
  - **buffering can increase the efficiency of the OS and the performance of individual processes.**