

Chapter 5

Input/output

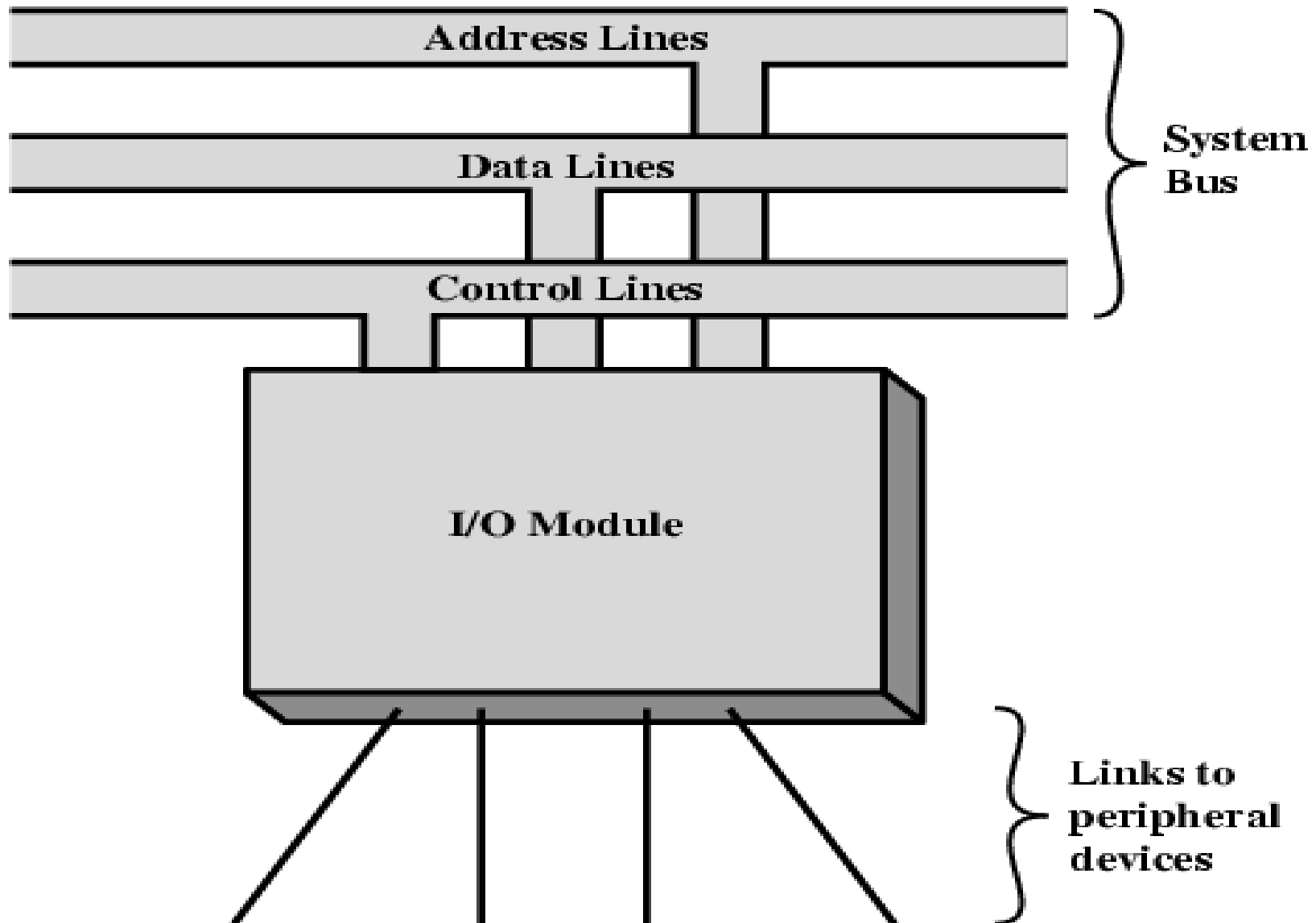
Input/output

- In addition to the CPU and a set of memory modules, the third key element of a computer system is a set of **I/O modules**
- Each module interfaces to the system bus or central switch and controls one or more peripheral devices
- I/O module contains logic for performing a communication function between the peripheral and the bus.

Peripherals do not directly connected to system bus. Due to the following reasons

- There are wide variety of peripherals (external devices) with various methods of operation. It would be impractical to incorporate the necessary logic within the processor to control a range of devices.
- The data transfer rate of peripherals is often slower than CPU and RAM. Thus, it is impractical to use the high-speed system bus to communicate directly with a peripheral.
- On the other hand, the data transfer rate of some peripherals is faster than that of the memory or processor. Again, the mismatch would lead to inefficiencies if not managed properly.
- Peripherals often use different data formats and word lengths than the computer to which they are attached.
- Thus , an **I/O module** is required.
- I/O module has two major functions
 - Interface to the CPU and memory via the system bus₃
 - Interface to one or more peripheral devices by tailored

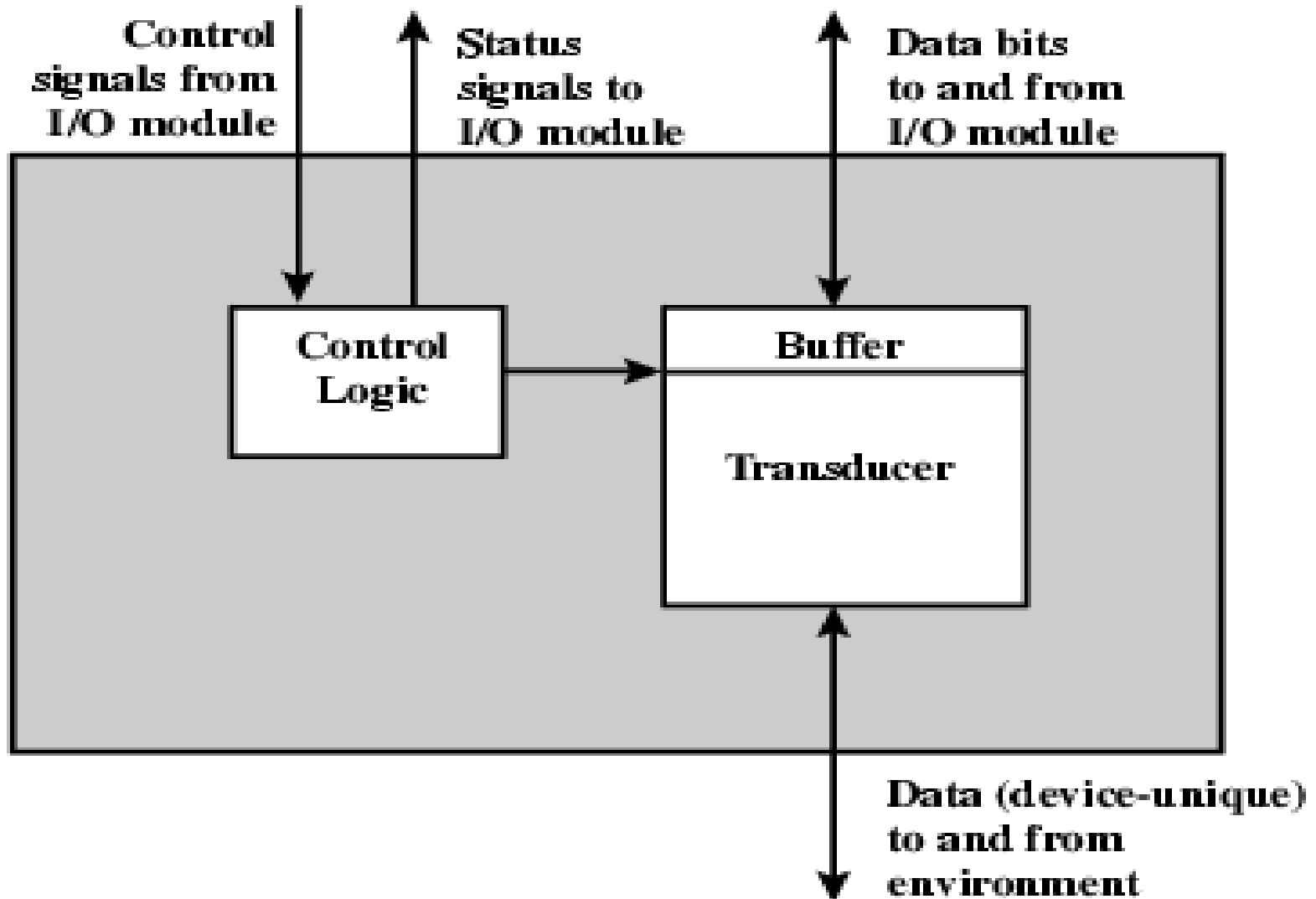
Generic Model of I/O Module



External Devices classification

- An external device connected to an I/O module is often referred to as a *peripheral Device* or , a peripheral.
- External devices are classified into three categories:
- Human readable
 - Used to communicate with the user
 - Examples: Printers, Video Display Terminals
- Machine readable
 - Used to communicate with electronic equipment
 - Examples: Disk and tap drives, Sensors, Controllers
- Communication
 - allow a computer to exchange data with a remote device
 - Examples: Modems

External Device Block Diagram



Control, Data, and Status signals

- **Control signals** *determine the function that the device will perform, such as send data to the I/O module (INPUT or READ), accept data from the I/O module (OUTPUT or WRITE), report status, or perform some control function particular to the device (e.g., position a disk head).*
- **Data** *are in the form of a set of bits to be sent to or received from the I/O module.*
- **Status signals** *indicate the state of the device.*
- **Control logic** *associated with the device controls the device's operation in response to direction from the I/O module.*
- The **transducer** *converts data from electrical to other forms of energy during output and from other forms to electrical during input.*
- **buffer** *is associated with the transducer to temporarily hold data being transferred between the I/O module and the external environment*

Functions of I/O Module

- Control & Timing
 - Coordinates the flow of traffic between internal resources and external devices
- CPU Communication
 - Involves command decoding, data, status reporting, address recognition
- Device Communication
 - Involves commands, status information, and data
- Data Buffering
 - Performs the needed buffering operation to balance device and memory speeds
- Error Detection
 - Detects and reports transmission errors

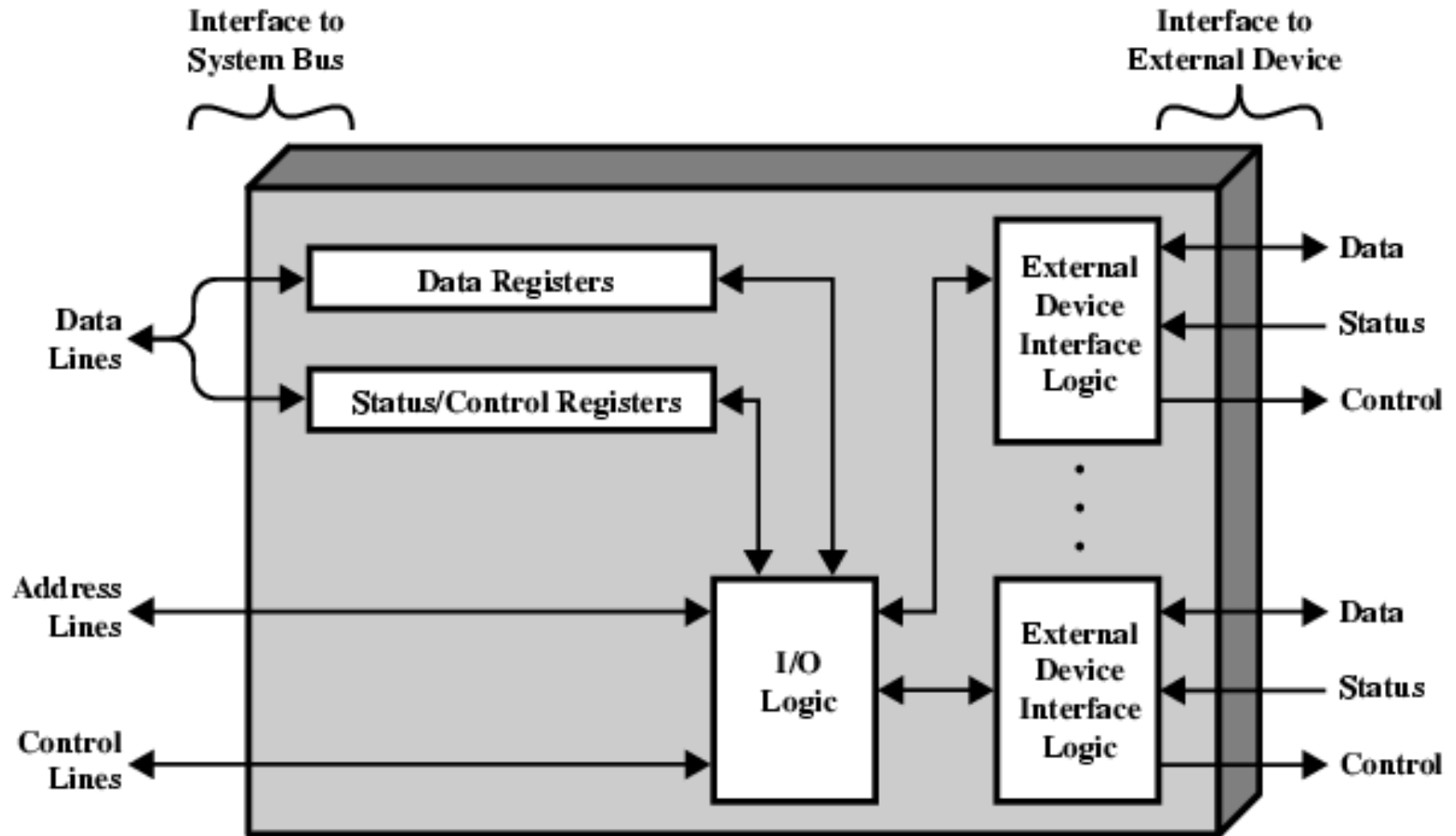
I/O Steps for transfer of data from an external device to the processor

- CPU checks I/O module device status
- I/O module returns status
- If ready, CPU requests data transfer
- I/O module gets data from device
- I/O module transfers data to CPU
- Variations for output, DMA, etc.

Processor communication involves the following:

- **Command decoding:** The I/O module accepts commands from the processor, typically sent as signals on the control bus.
- **Data:** Data are exchanged between the processor and the I/O module over the data bus.
- **Status reporting:** Because peripherals are so slow, it is important to know the status of the I/O module
- **Address recognition:** Just as each word of memory has an address, so does each I/O device. Thus, an I/O module must recognize one unique address for each peripheral it controls.
- **Device communication** involves commands, status information, and data

I/O Module Diagram



Keyboard /Monitor

- The most common means of computer/user interaction is a keyboard/monitor arrangement.
- The user provides input through the keyboard. This input is then transmitted to the computer and may also be displayed on the monitor
- The basic unit of exchange is the **character**. Associated with each character is a code, typically 7 or 8 bits in length. The most commonly used text code is the **International Reference Alphabet (IRA)**. The US national version of IRA is referred to as the **ASCII**.
- Each character in this code is represented by a 7-bit binary code
- Characters are of two types: ***printable and control***.
 - Printable characters are the alphabetic, numeric, and special characters that can be printed on paper or displayed on a screen.
 - Control characters have to do with controlling the printing or displaying of characters; an example is carriage return.

Keyboard/Monitor

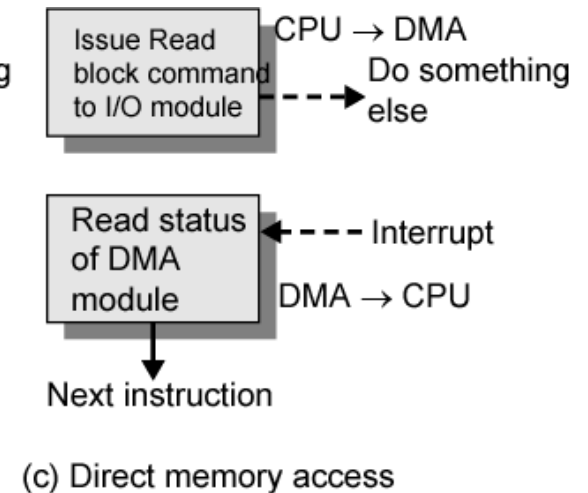
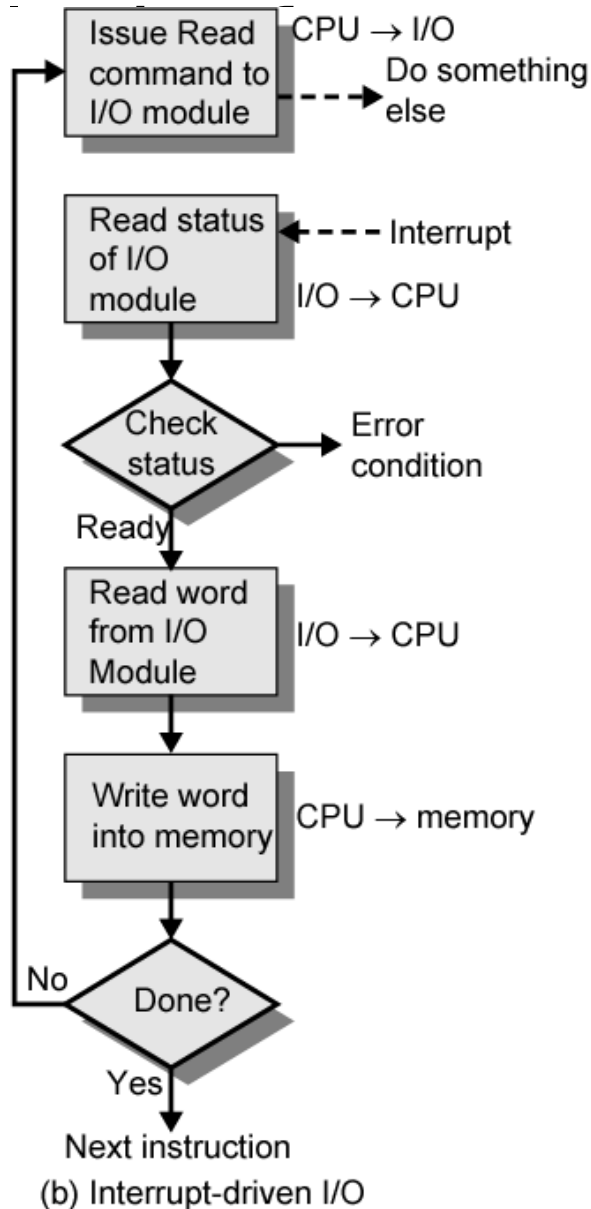
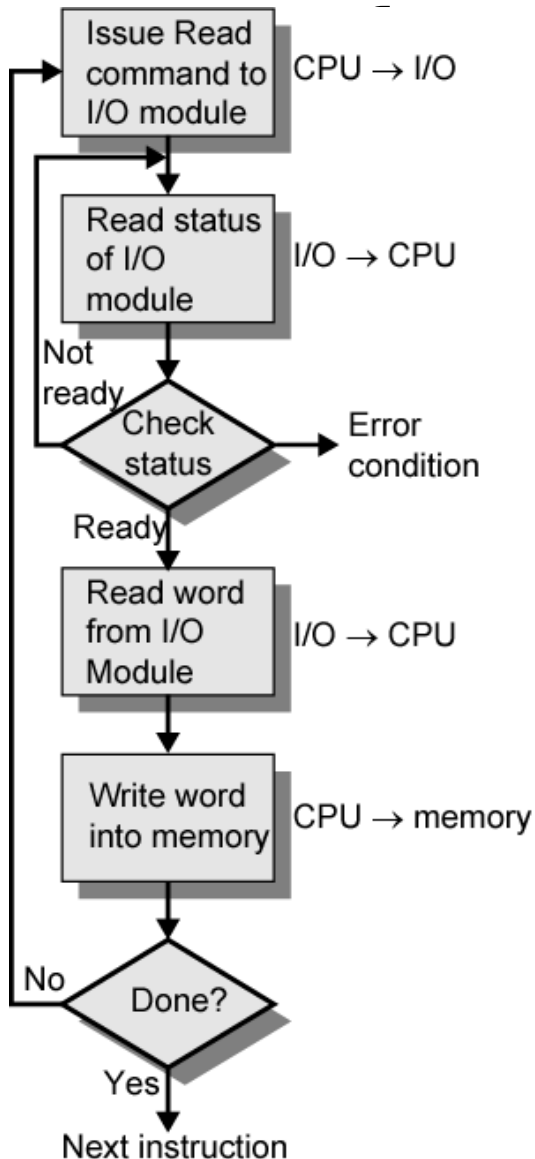
- **For keyboard input**, when the user depresses a key, this generates an electronic signal that is interpreted by the **transducer** in the keyboard and translated into the bit pattern of the corresponding IRA code.
- This bit pattern is then transmitted to the I/O module in the computer. At the computer, the text can be stored in the same IRA code.
- **On output**, IRA code characters are transmitted to an external device from the I/O module.
- The transducer at the device interprets this code and sends the required electronic signals to the output device either to display the indicated character or perform the requested control function.

Input Output Techniques

- Programmed I/O
- Interrupt driven I/O
- Direct Memory Access (DMA)

	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)

Three Techniques for



Programmed I/O

- The processor executes a program that gives it direct control of the I/O operation, including
 - Sensing status
 - Read/write commands
 - Transferring data
- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.
- If the processor is faster than the I/O module, this is wasteful of processor time.

Programmed I/O - detail

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- I/O module does not inform CPU directly
- I/O module does not interrupt CPU
- CPU may wait or come back later

I/O Commands

- To execute an I/O-related instruction, the processor issues an address, specifying the particular I/O module and external device, and an I/O command CPU issues command.
- There are four types of I/O commands that an I/O module may receive when it is addressed by a processor:
 - **Control:** Used to activate a peripheral and tell it what to do.
 - **Test:** Used to test various status conditions associated with an I/O module and its peripherals.
 - e.g. power? Error?
 - **Read:** Causes the I/O module to obtain an item of data from the peripheral and place it in an internal buffer
 - **Write:** Causes the I/O module to take an item of data (byte or word) from the data bus and subsequently transmit that data item to the peripheral.

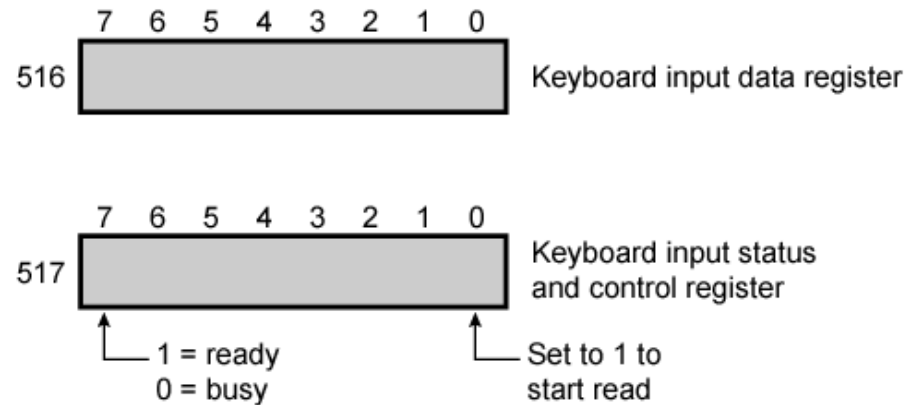
I/O Instructions

- Under programmed I/O data transfer is very like memory access (CPU viewpoint)
- Each device given unique identifier
- CPU commands contain identifier (address)
- When the processor issues an I/O command, the command contains the address of the desired device.
- When the processor, main memory, and I/O share a common bus, two modes of addressing are possible
 - memory mapped
 - isolated

I/O Mapping

- Memory mapped I/O
 - there is a single address space for memory locations and I/O devices.
 - a single read line and a single write line are needed on the bus.
 - No special commands for I/O
 - Large selection of memory access commands available
- Isolated I/O
 - Separate address spaces
 - Need I/O or memory select lines
 - Special commands for I/O
 - Limited set

Memory Mapped and Isolated I/O



ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

Interrupt Driven I/O

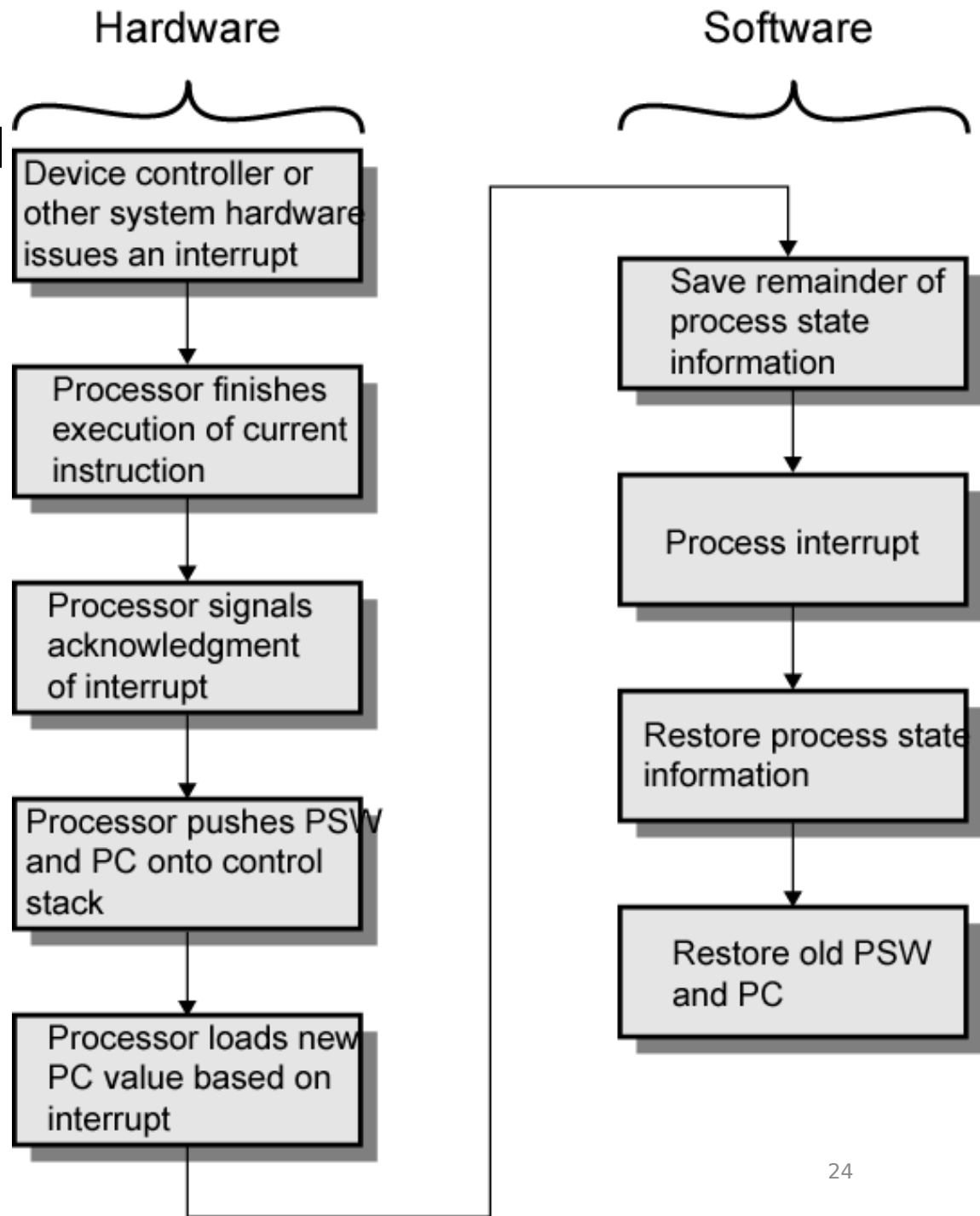
- With interrupt-driven I/O, the processor issues an I/O command, continues to execute other instructions, and is interrupted by the I/O module when the latter has completed its work.
- The problem with programmed I/O is that the processor has to wait a long time for the I/O module of concern to be ready 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.
- The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor. The processor then executes the data transfer, as before, and then resumes its former processing.

Interrupt Driven I/O

Basic Operation

- CPU issues read command
- I/O module gets data from peripheral whilst CPU does other work
- I/O module interrupts CPU
- CPU requests data
- I/O module transfers data

Simple Interrupt Processing



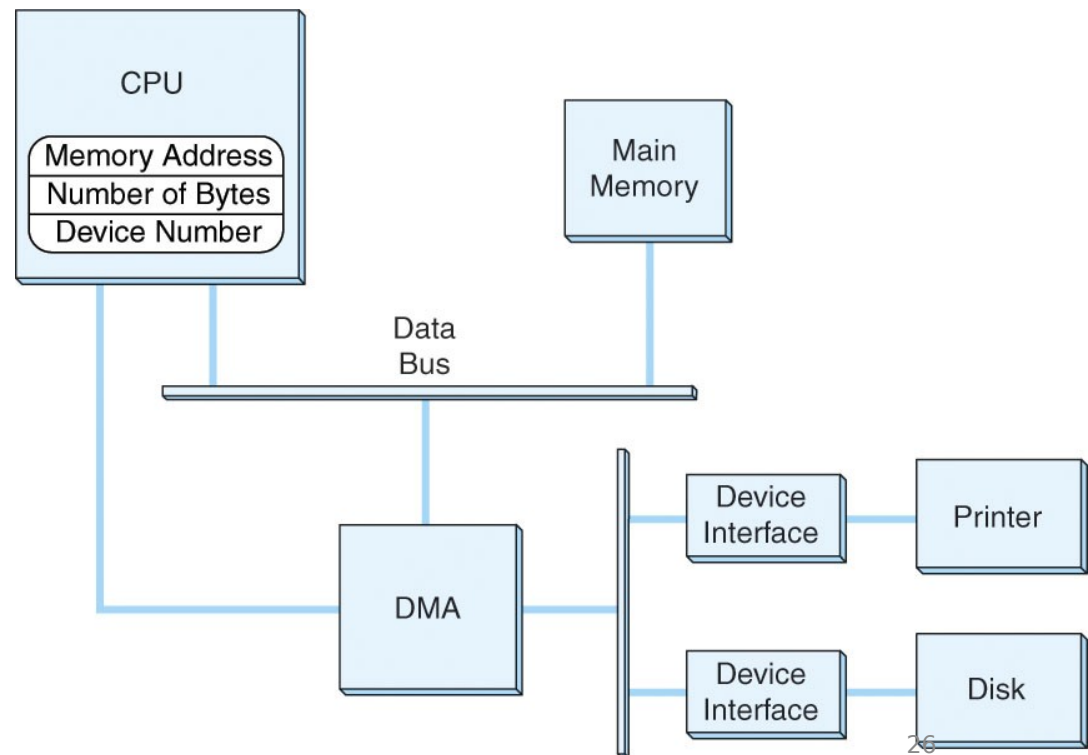
Direct Memory Access

- Interrupt-driven I/O, though more efficient than simple programmed I/O, still requires the active intervention of the processor to transfer data between memory and an I/O module
- Drawbacks of Programmed and Interrupt-Driven I/O
 - The I/O transfer rate is limited by the speed with which the processor can test and service a device.
 - The processor is tied up in managing an I/O transfer; a number of instructions must be executed for each I/O transfer
- When large volumes of data are to be moved, a more efficient technique is required: direct memory access (DMA).
- In DMA, the I/O module and main memory exchange data directly, without processor involvement.

DMA I/O

- Direct memory access allows an I/O module to communicate directly with memory so that the CPU does not have to be interrupted for each data movement
 - In any large transfer, DMA I/O is preferable to Interrupt-driven I/O
 - The DMA controller only interrupts the CPU once, once the entire data transfer is complete
 - Note: memory is a slave device so it cannot initiate bus transfers, the DMA controller then acts as a memory master

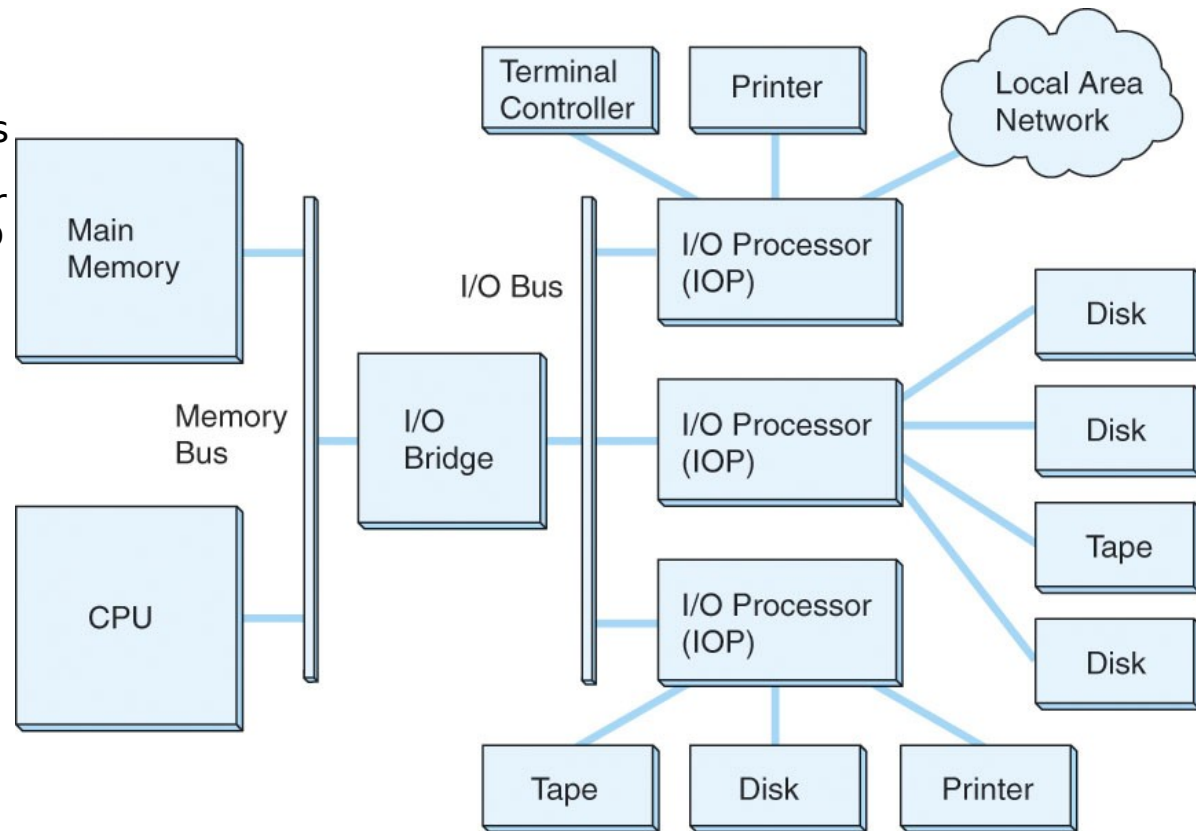
- The DMA controller includes a count register set by the CPU to count the number of bytes still to be transferred
- The DMA controller acts as an interface between I/O device and memory



Channel I/O

- In DMA I/O, if there is a failure of the I/O device, the I/O module interrupts the CPU to handle the problem
- A more sophisticated I/O module is known as an I/O channel
 - this is an I/O module with its own mini-processor (known as an I/O processor)
 - these processors contain their own ALUs and control units so that they can perform comparisons and make branches in their own mini-programs

- Channel I/O can handle any problem without having to interrupt the CPU

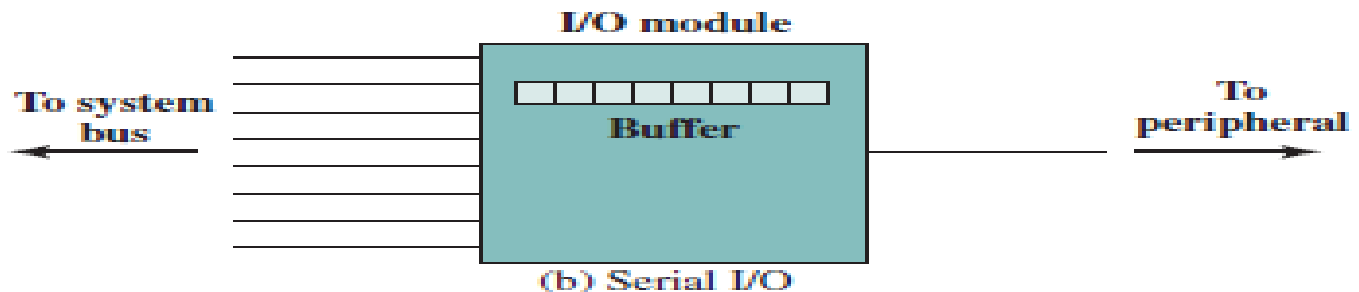
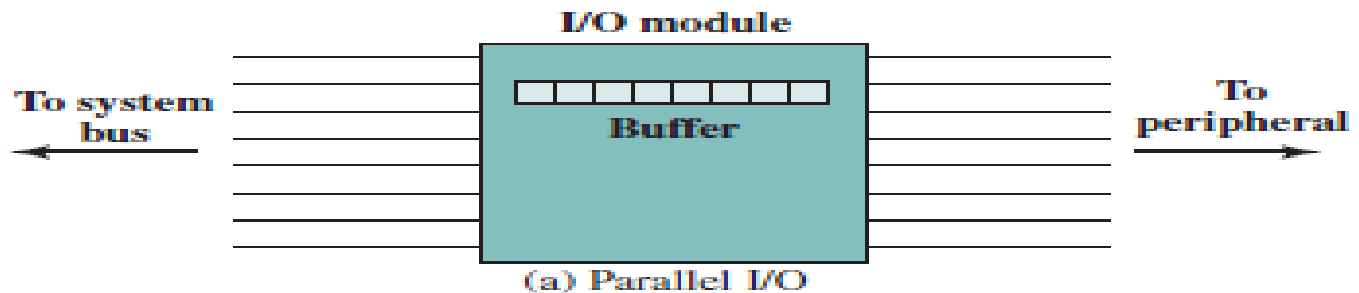


I/O architecture with I/O Channels

THE EXTERNAL INTERFACE

- **Types of Interfaces**

- **parallel interface**, there are multiple lines connecting the I/O module and the peripheral, and multiple bits are transferred simultaneously
- **serial interface**, there is only one line used to transmit data, and bits must be transmitted one at a time



Types of Interfaces

- A **parallel interface** has traditionally been used for **higher-speed peripherals**, such as **tape and disk**, while
- **Serial interface** has traditionally been used for **printers and terminals**.
- With a new generation of high-speed serial interfaces, parallel interfaces are becoming much less common.
- In either case, the I/O module must engage in a dialogue with the peripheral.

Point-to-Point and Multipoint Configurations

- The connection between an I/O module in a computer system and external devices can be either **point-to-point** or **multipoint**.
- A **point-to-point interface** provides a dedicated line between the I/O module and the external device. On small systems (PCs, workstations), typical point-to-point links include those to the **keyboard**, **printer**, and **external modem**.
- **Multipoint external interfaces** are used to support external mass storage devices (disk and tape drives) and multimedia devices (CD-ROMs, video, audio).
 - Two key examples are : **Thunderbolt** and **InfiniBand**

Thunderbolt

- Most recent and fastest peripheral connection technology to become available for general-purpose use
- Developed by Intel with collaboration from Apple
- The technology combines data, video, audio, and power into a single high-speed connection for peripherals such as hard drives, RAID arrays, video-capture boxes, and network interfaces
- Provides up to 10 Gbps throughput in each direction and up to 10 Watts of power to connected peripherals

InfiniBand

- I/O specification aimed at the high-end server market
- First version was released in early 2001
- Heavily relied on by IBM zEnterprise series of mainframes
- Standard describes an architecture and specifications for data flow among processors and intelligent I/O devices
- Has become a popular interface for storage area networking and other large storage configurations
- Enables servers, remote storage, and other network devices to be attached in a central fabric of switches and links
- The switch-based architecture can connect up to 64,000 servers, storage systems, and networking devices