AACS3064
Computer Systems Architecture

Chapter 11: Input and Output Facilities

Chapter Overview

1) Characteristics of typical I/O devices
2) I/O Techniques
3) Programmed I/O
4) Interrupts
   - Servicing Interrupts
   - The Uses of Interrupts
   - Multiple Interrupts and Prioritization
4) Direct Memory access (DMA)
5) I/O System Architecture
   - I/O Bus Architecture
   - Channel Architecture
6) I/O Modules

1. Characteristics of typical I/O devices

1. Characteristics of typical I/O devices

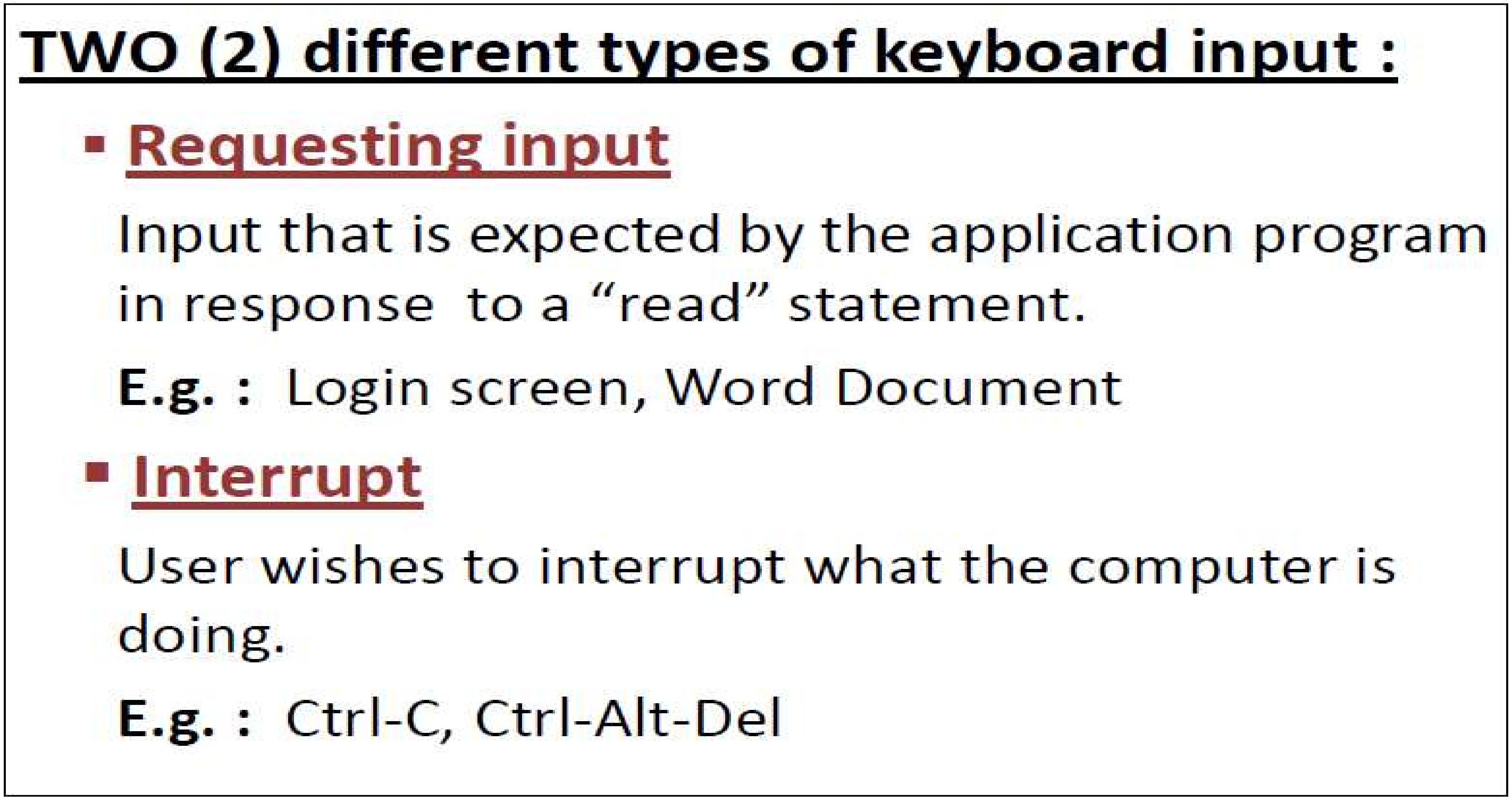
Keyboard

- A **character-based** device, one character at a time.

  →the data rate for keyboard is very slow
- Why slow?
  - dependent on the **speed of typing** and on the **thought process** of the user.
- Computer spend most of its time waiting for input from the keyboard.

1. Characteristics of typical I/O devices (Continued)

Keyboard



TWO (2) different types of keyboard input :

- **Requesting input**

  Input that is expected by the application program in response to a "read" statement.

  E.g. : Login screen, Word Document

- **Interrupt**

  User wishes to interrupt what the computer is doing.

  E.g. : Ctrl-C, Ctrl-Alt-Del

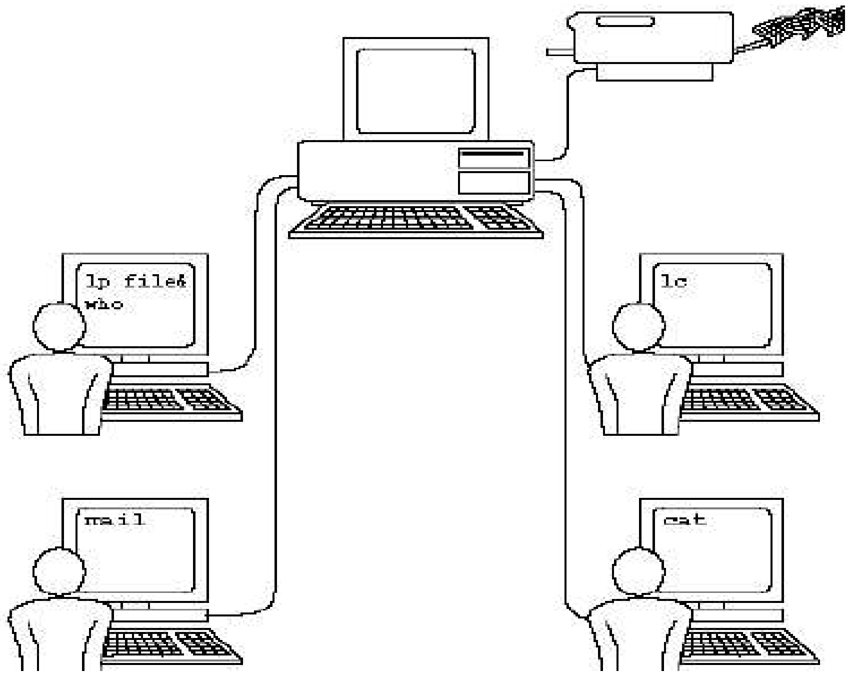
1. Characteristics of typical I/O devices (Continued)

Keyboard

Multi users system :

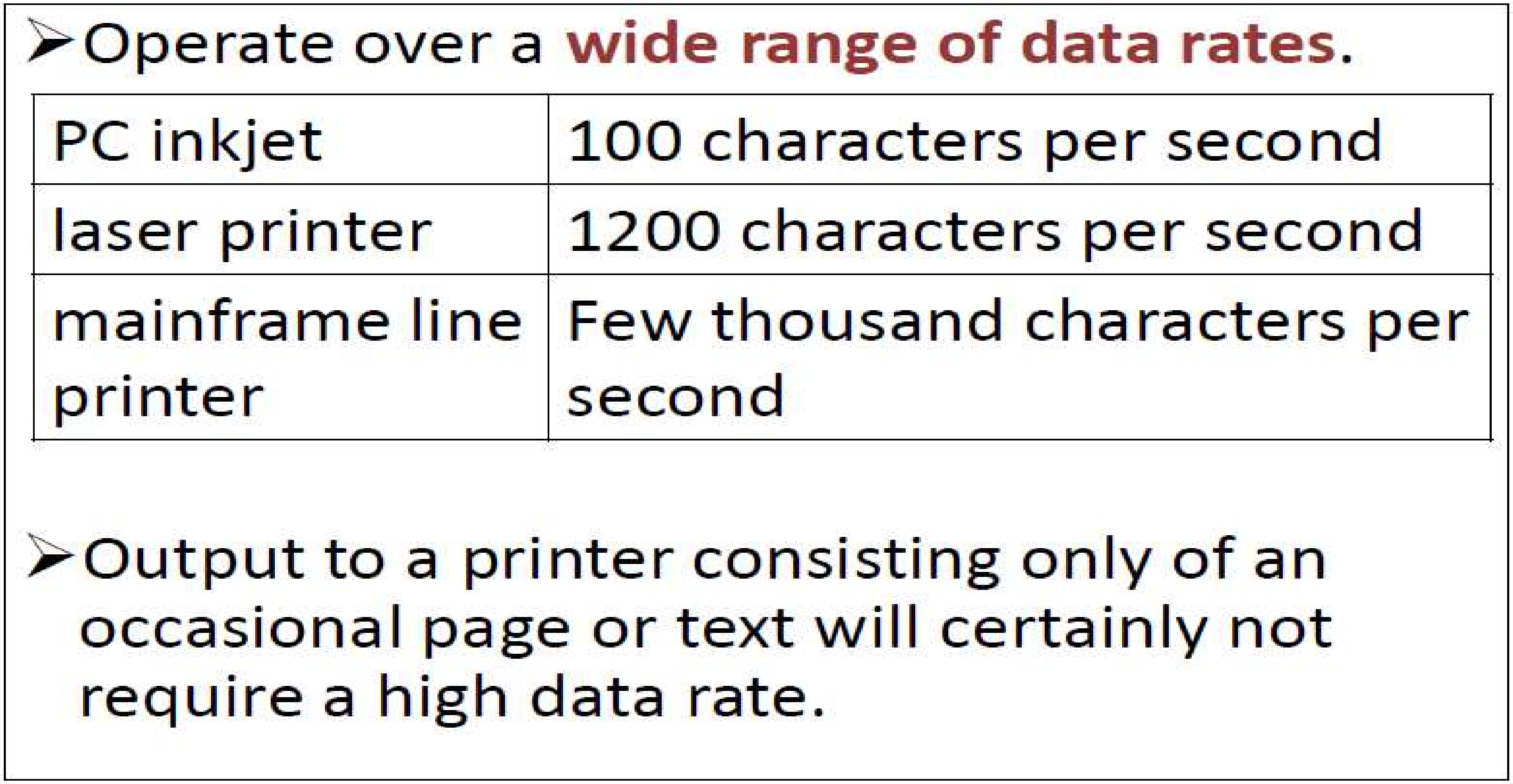
- ➢Many keyboards connected to a computer.
- ➢Computer must be able to distinguish them.

- **No data lose** even if several keyboards send a character simultaneously.
- Able to **respond quickly** to each keyboard.



1. Characteristics of typical I/O devices (Continued)

Printer



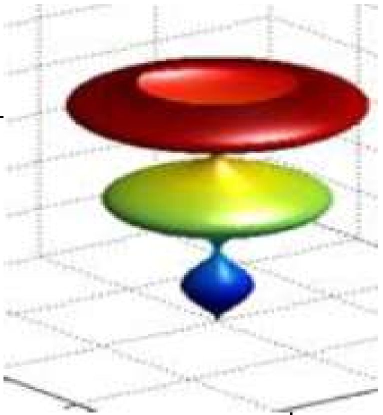
➢Operate over a **wide range of data rates**.

PC inkjet	100 characters per second
laser printer	1200 characters per second
mainframe line printer	Few thousand characters per second

➢Output to a printer consisting only of an occasional page or text will certainly not require a high data rate.

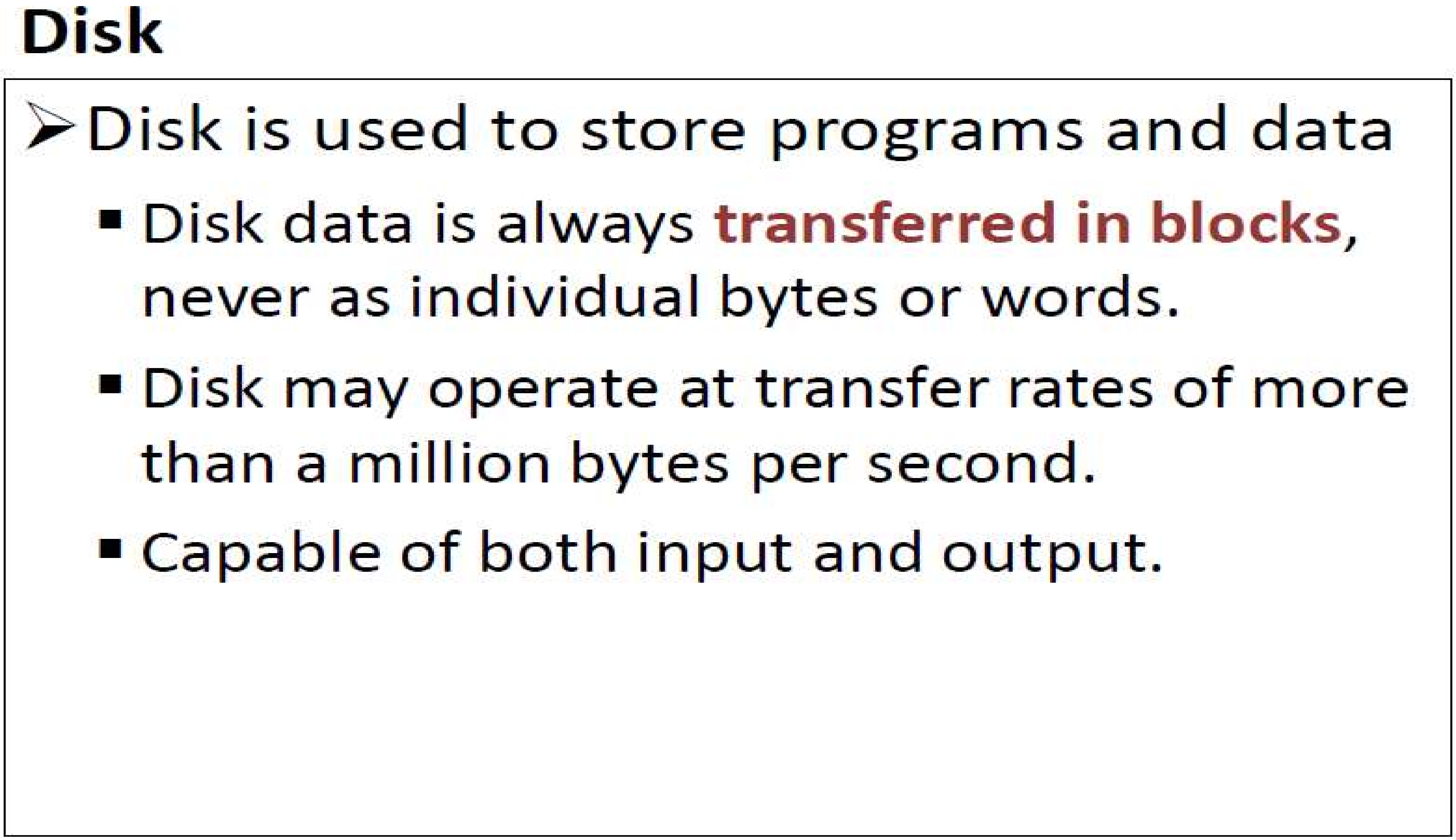
1. Characteristics of typical I/O devices (Continued)

Graphical output



- **Object images:**
  - Described by geometrical information
  - **E.g. :** Text instructions, lines & curves
  - The amount of data : **small**.
- **Bitmap images:**
  - Data for each pixel is to be produced
  - **E.g. :** Photograph picture
  - The amount of data : **huge**.
  - High speed data transfer is essential.

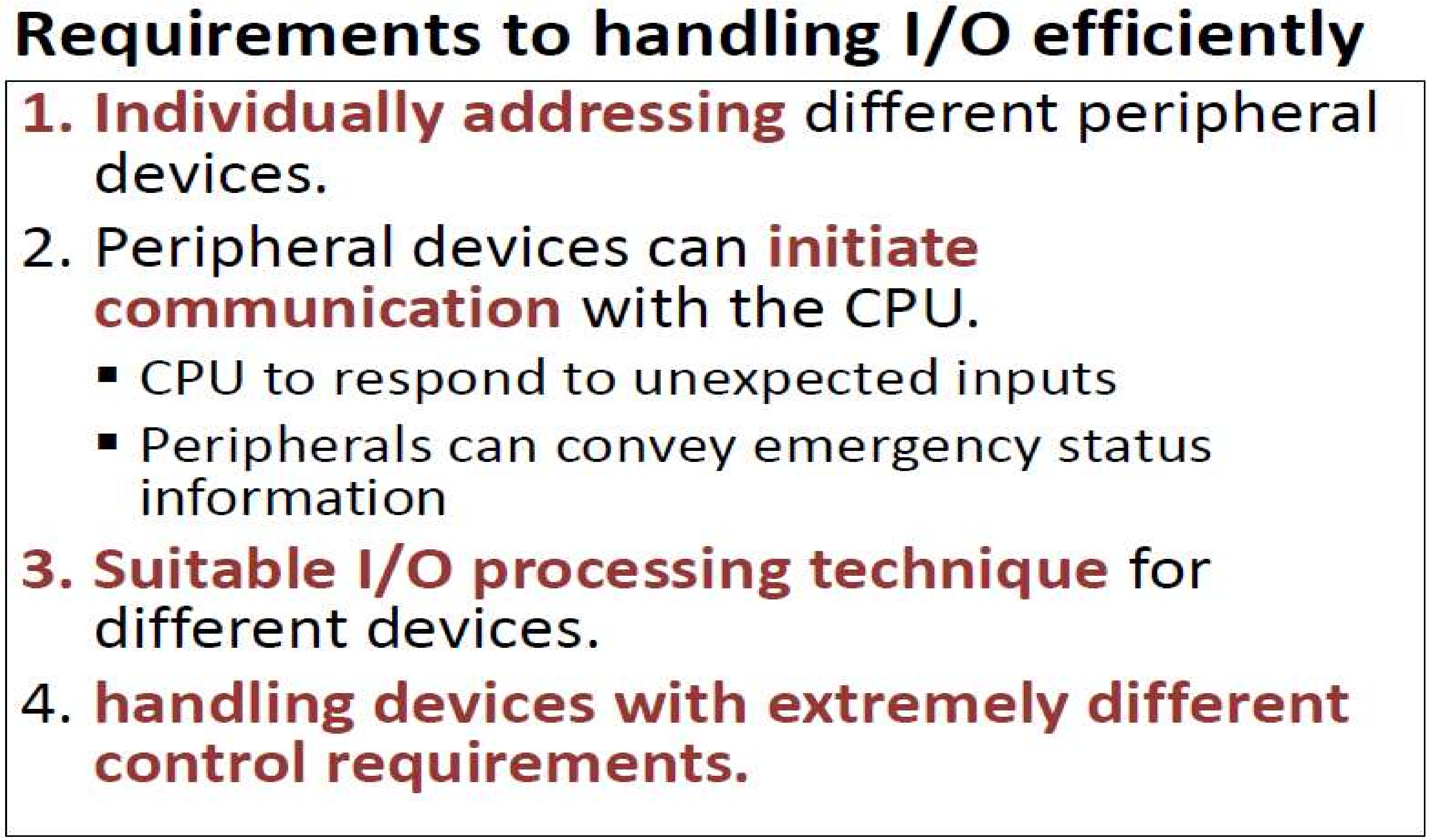
1. Characteristics of typical I/O devices (Continued)



Disk

- Disk is used to store programs and data
  - Disk data is always **transferred in blocks**, never as individual bytes or words.
  - Disk may operate at transfer rates of more than a million bytes per second.
  - Capable of both input and output.

1. Characteristics of typical I/O devices (Continued)

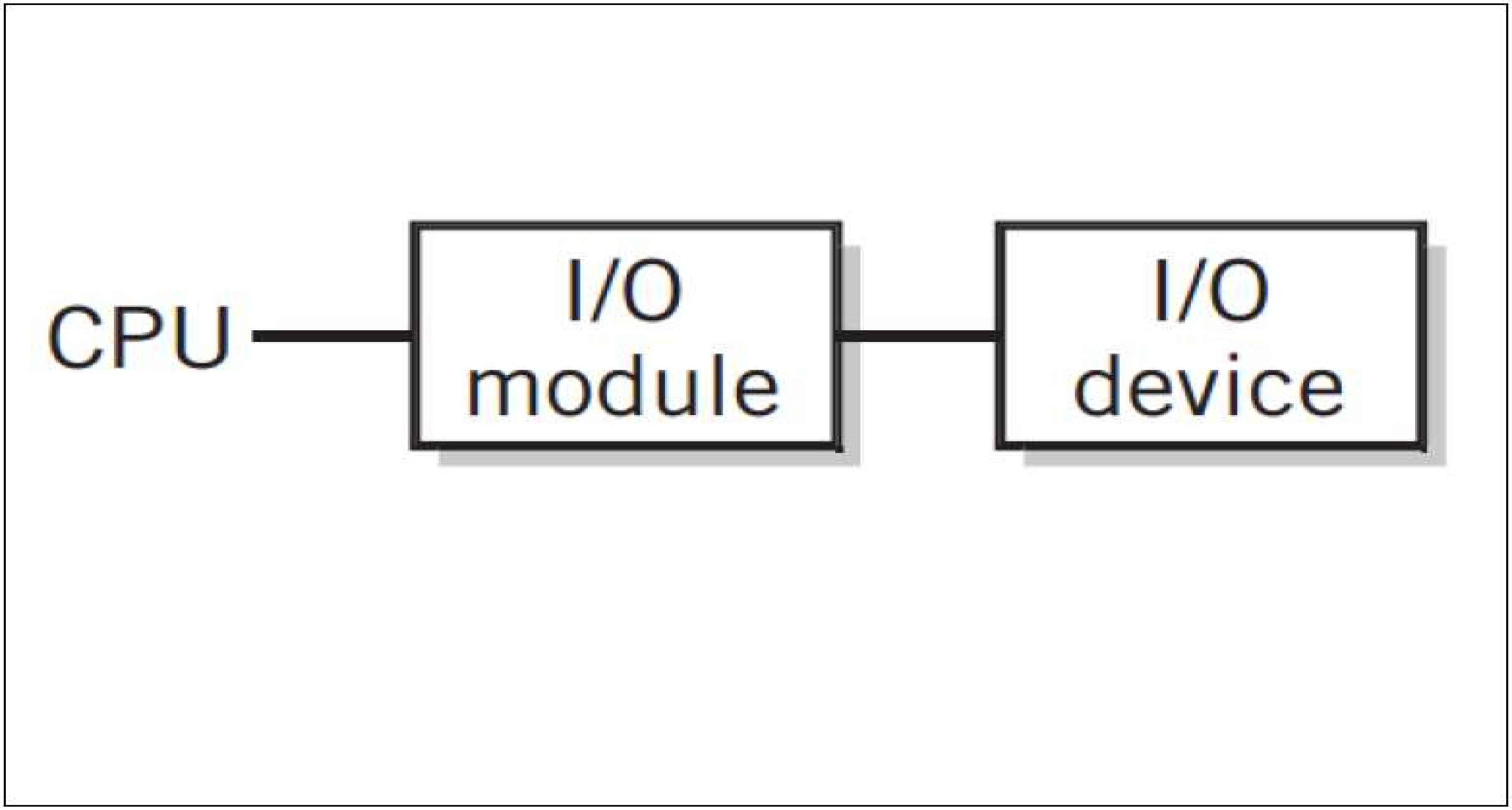


Requirements to handling I/O efficiently

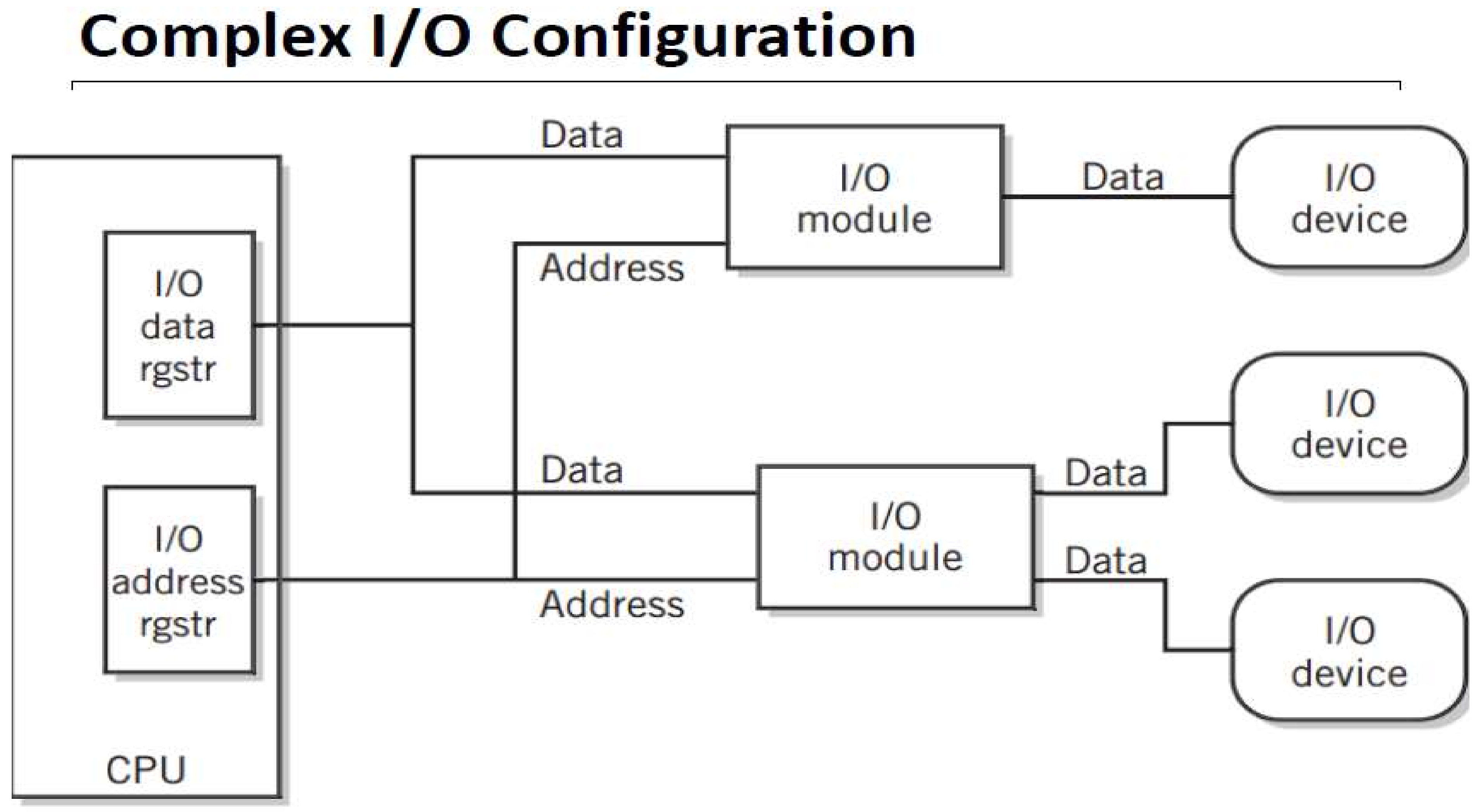
1. **Individually addressing** different peripheral devices.
2. Peripheral devices can **initiate communication** with the CPU.
   - CPU to respond to unexpected inputs
   - Peripherals can convey emergency status information
3. **Suitable I/O processing technique** for different devices.
4. **handling devices with extremely different control requirements.**

1. Characteristics of typical I/O devices (Continued)

Simple I/O Configuration

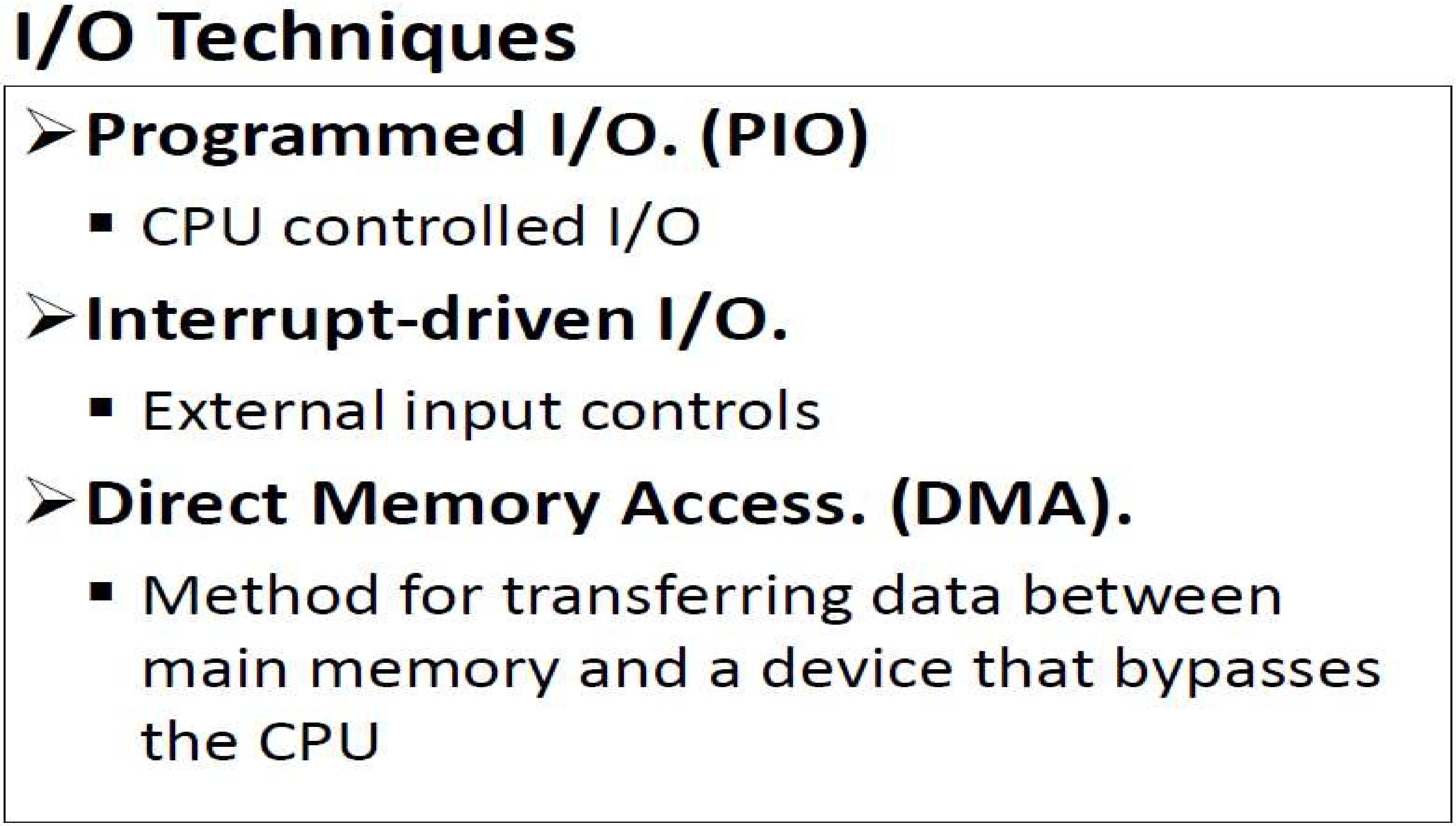


1. Characteristics of typical I/O devices (Continued)



2. I/O Techniques

2. I/O Techniques



I/O Techniques

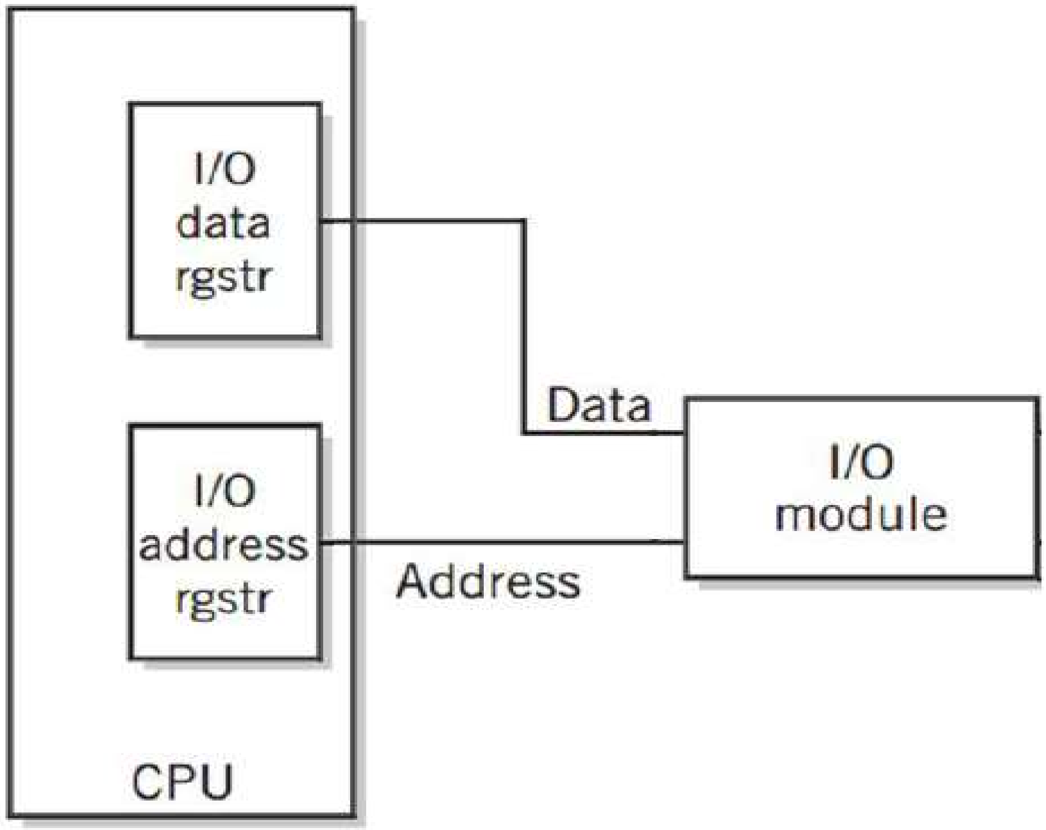
- ➢ **Programmed I/O. (PIO)**
  - CPU controlled I/O
- ➢ **Interrupt-driven I/O.**
  - External input controls
- ➢ **Direct Memory Access. (DMA).**
  - Method for transferring data between main memory and a device that bypasses the CPU

3. Programmed I/O

3. Programmed I/O

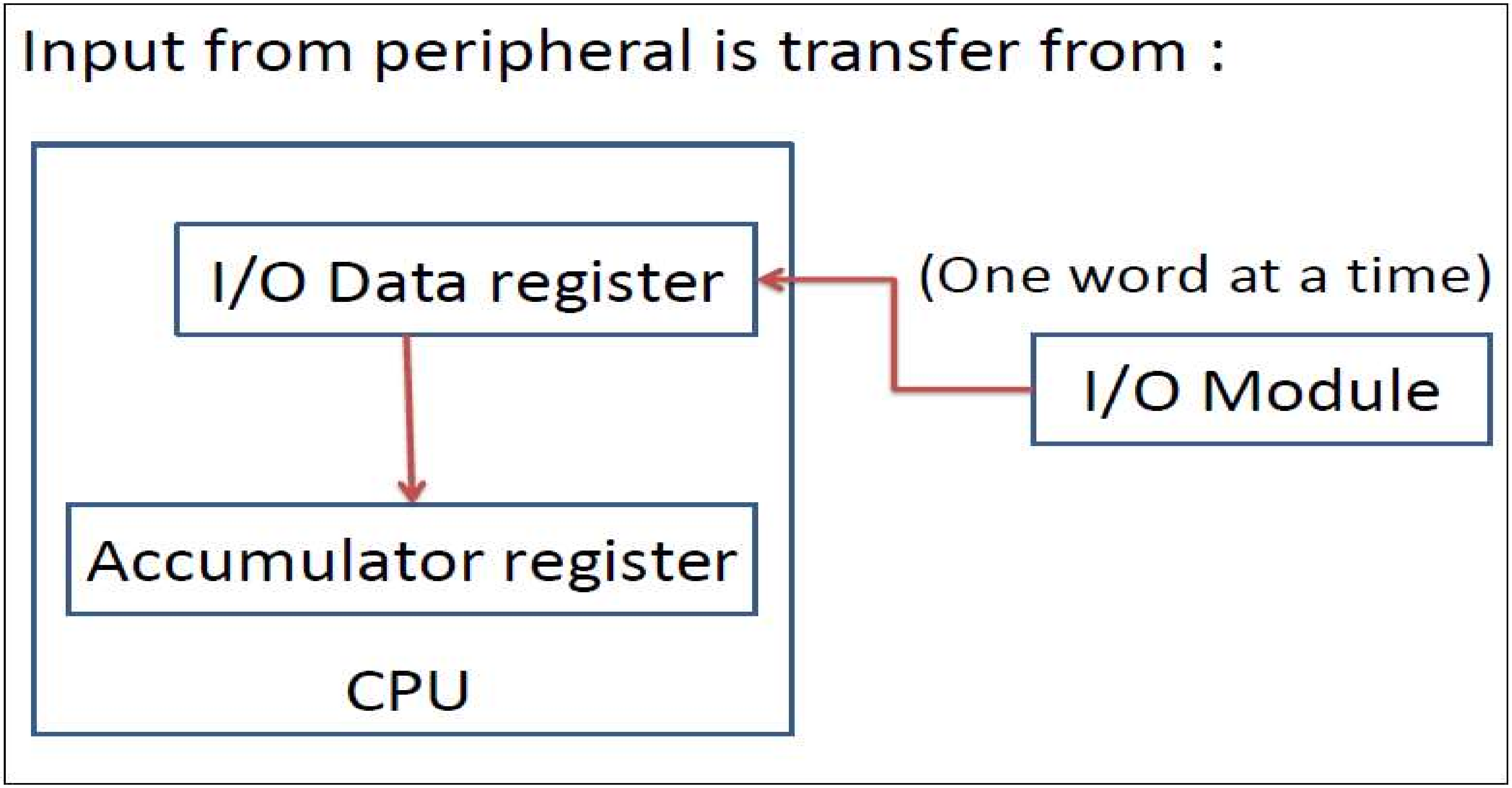
Programmed I/O

- The simplest method for performing I/O.
- An I/O module is connected to a pair of I/O registers in the CPU via a bus.



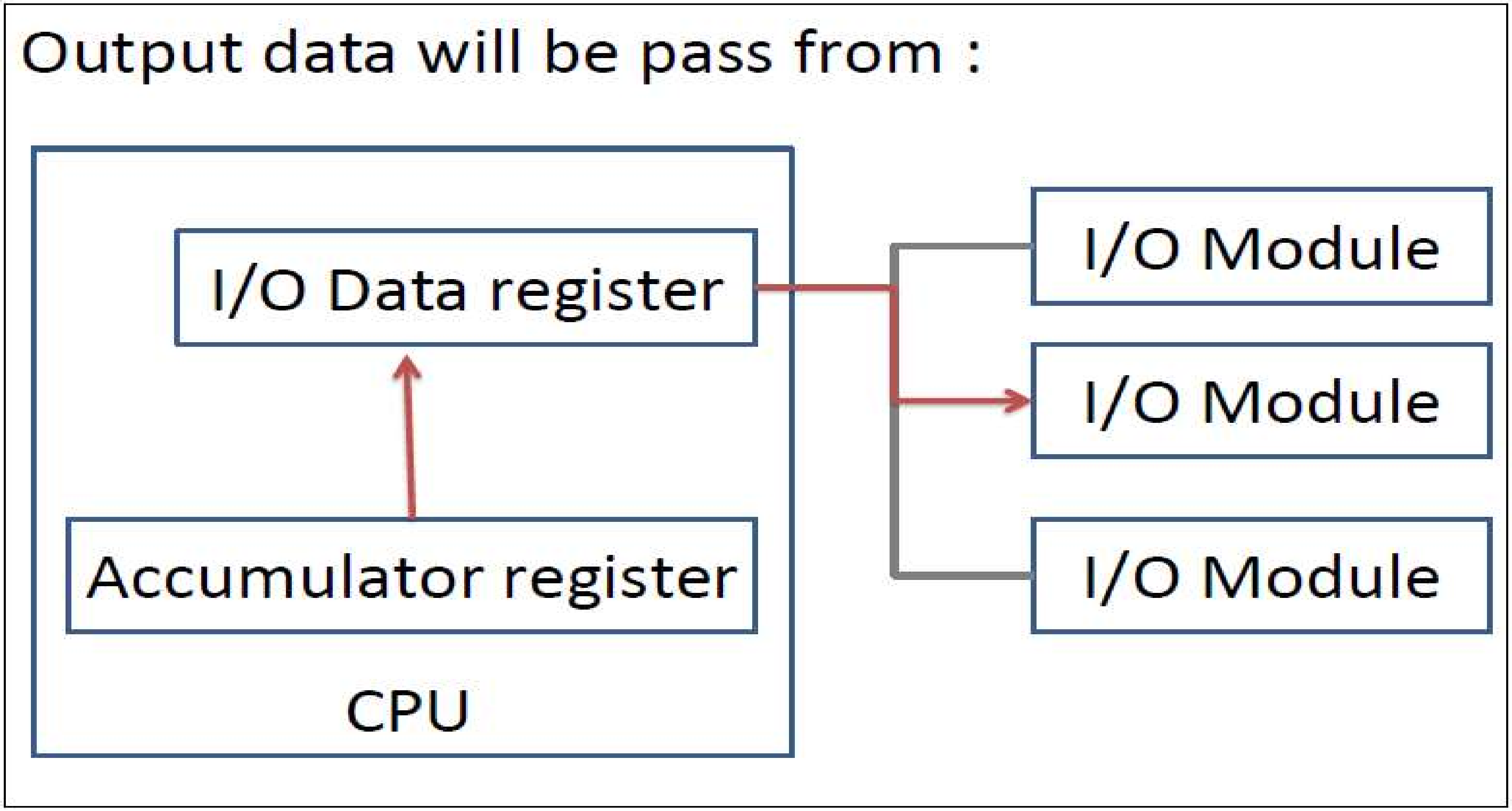
3. Programmed I/O (Continued)

Programmed I/O

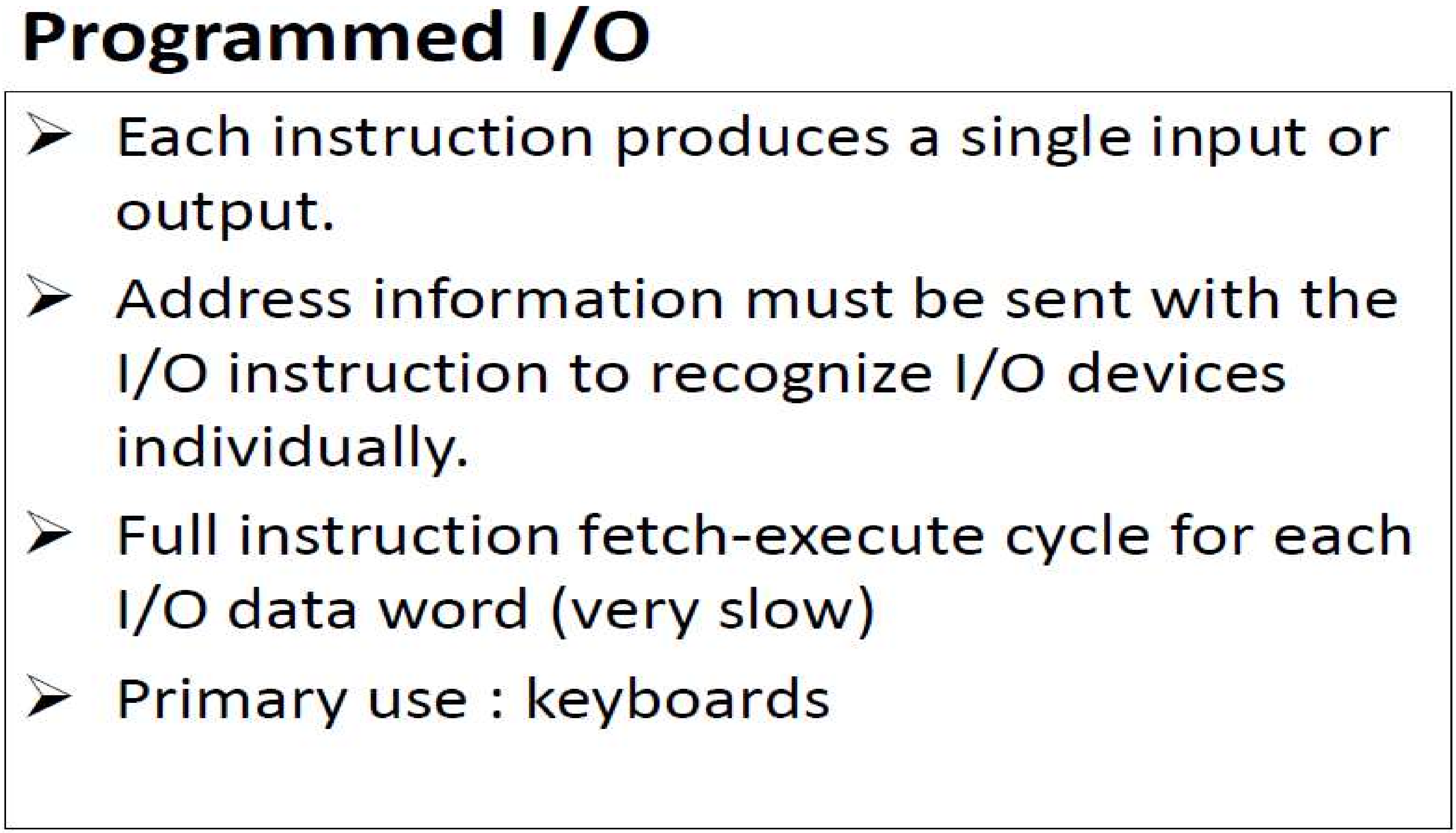

Input from peripheral is transfer from :

3. Programmed I/O (Continued)

Programmed I/O


Output data will be pass from :

3. Programmed I/O (Continued)



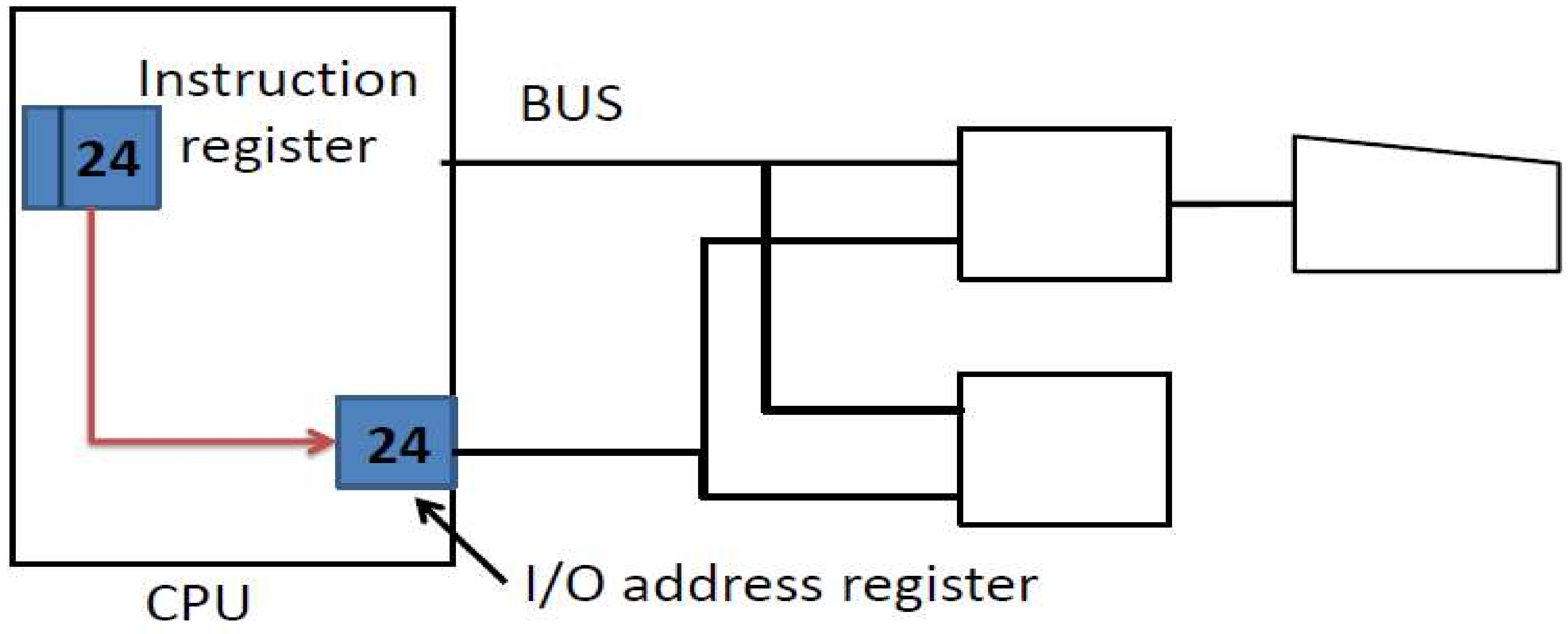
Programmed I/O

- Each instruction produces a single input or output.
- Address information must be sent with the I/O instruction to recognize I/O devices individually.
- Full instruction fetch-execute cycle for each I/O data word (very slow)
- Primary use : keyboards

3. Programmed I/O (Continued)

Example : Programmed I/O

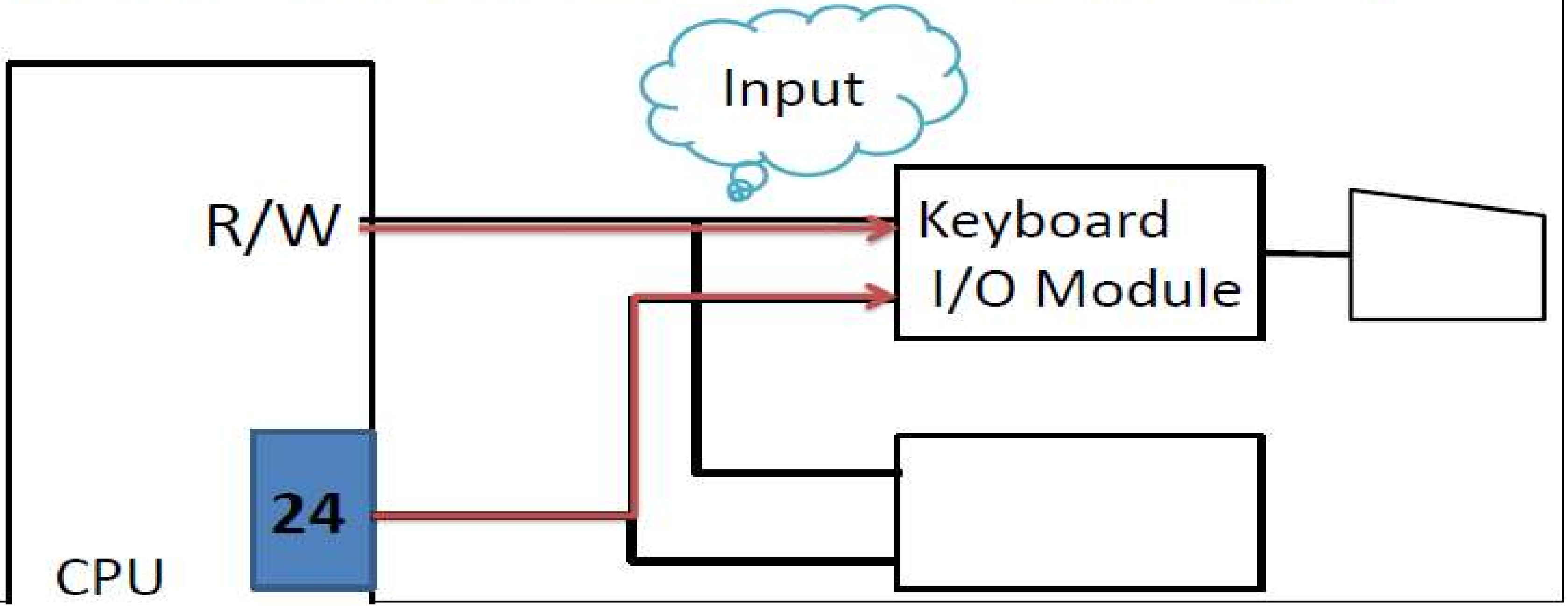
1. CPU executes INPUT 24 instruction. Address 24 is copied to the I/O address register.



3. Programmed I/O (Continued)

Example : Programmed I/O

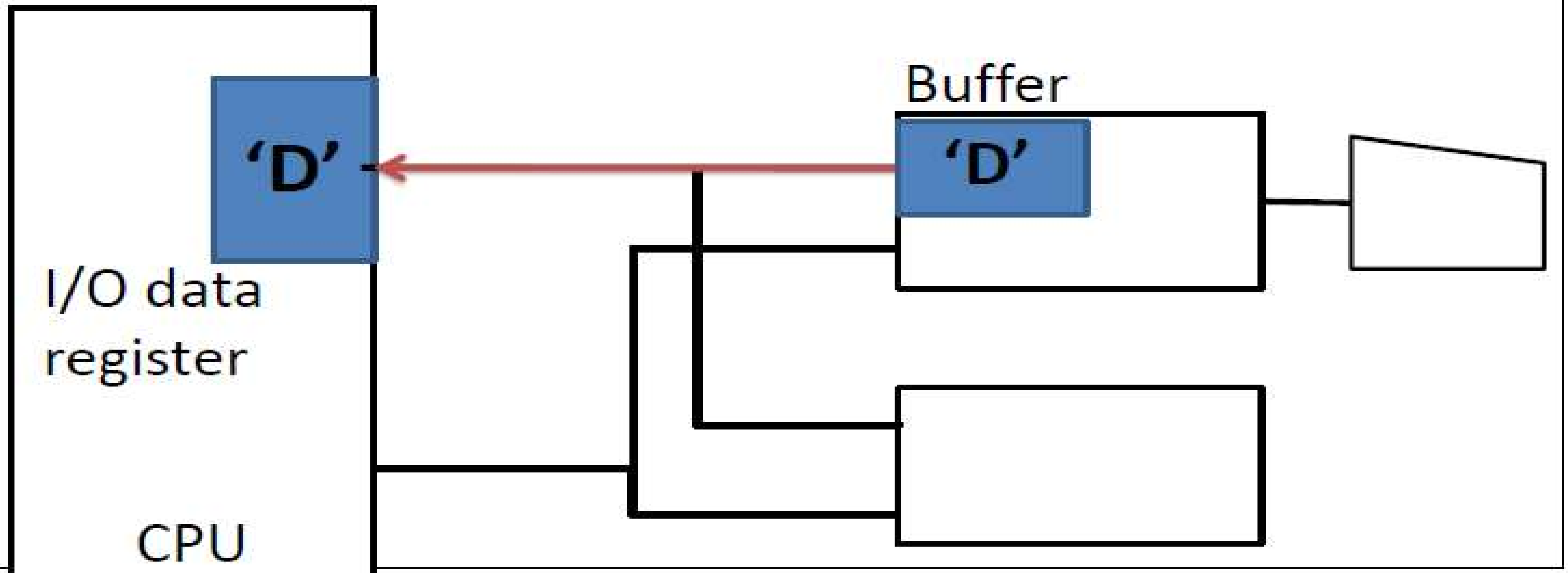
2. Address 24 is recognized by the keyboard I/O module. A read/write Control line indicates that the instruction is an INPUT.



3. Programmed I/O (Continued)

Example : Programmed I/O

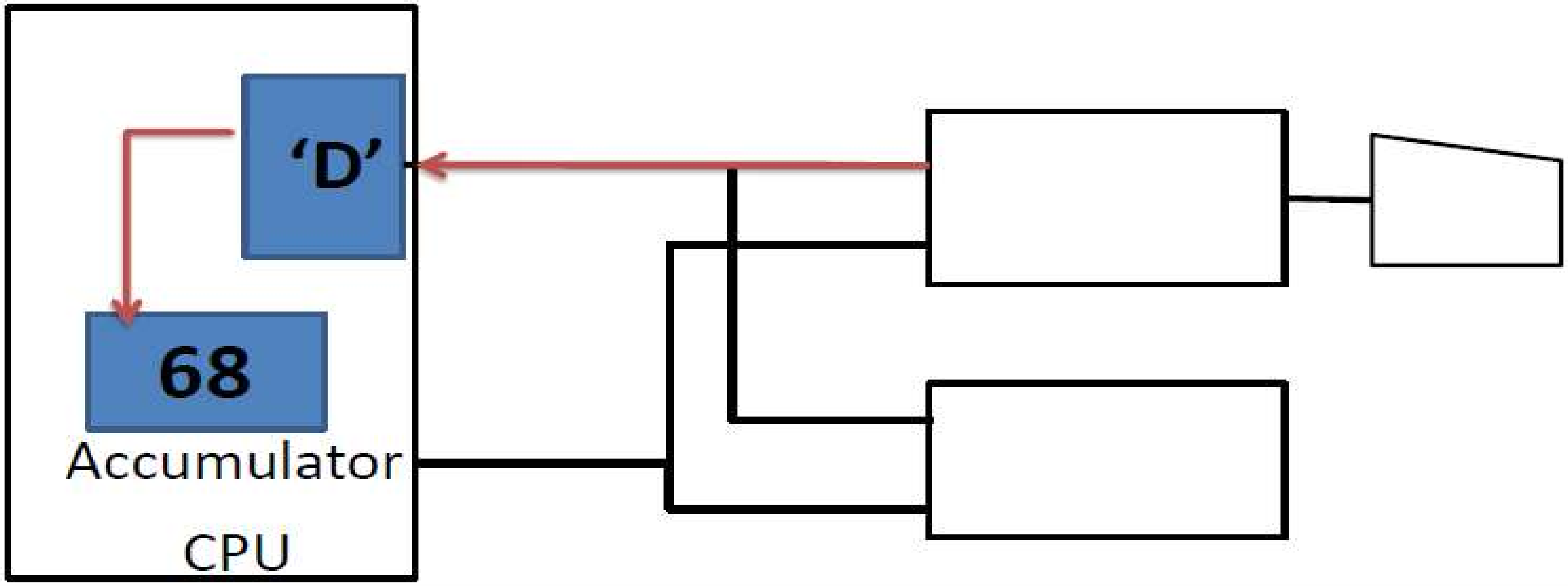
3. A buffer in the I/O module holds a keystroke, in this case ASCII 68, the letter "D". The data is transferred to the I/O data register.



3. Programmed I/O (Continued)

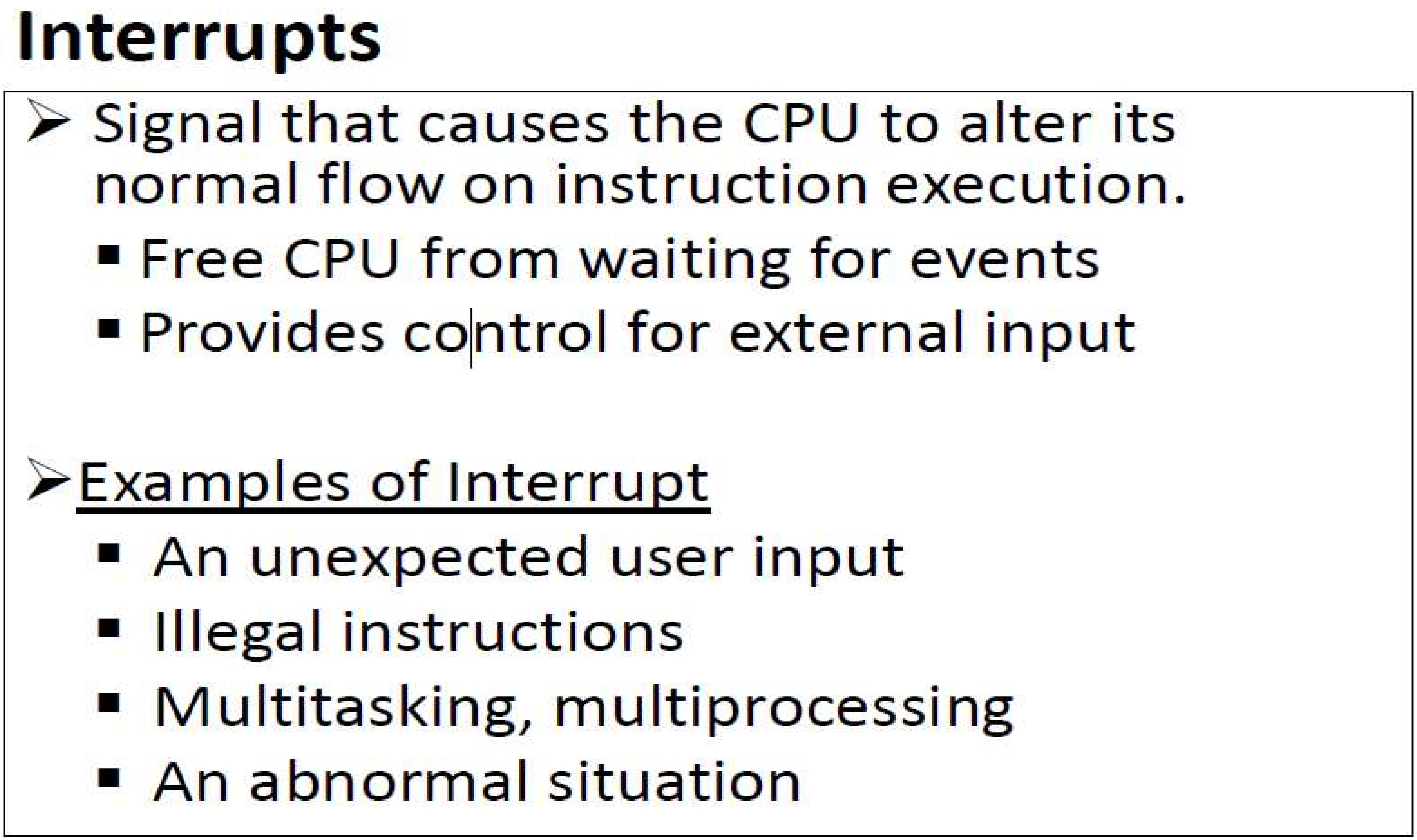
Example : Programmed I/O

4. From there, it is copied to the appropriate accumulator or general purpose register, completing the operation.



4. Interrupts

4. Interrupts

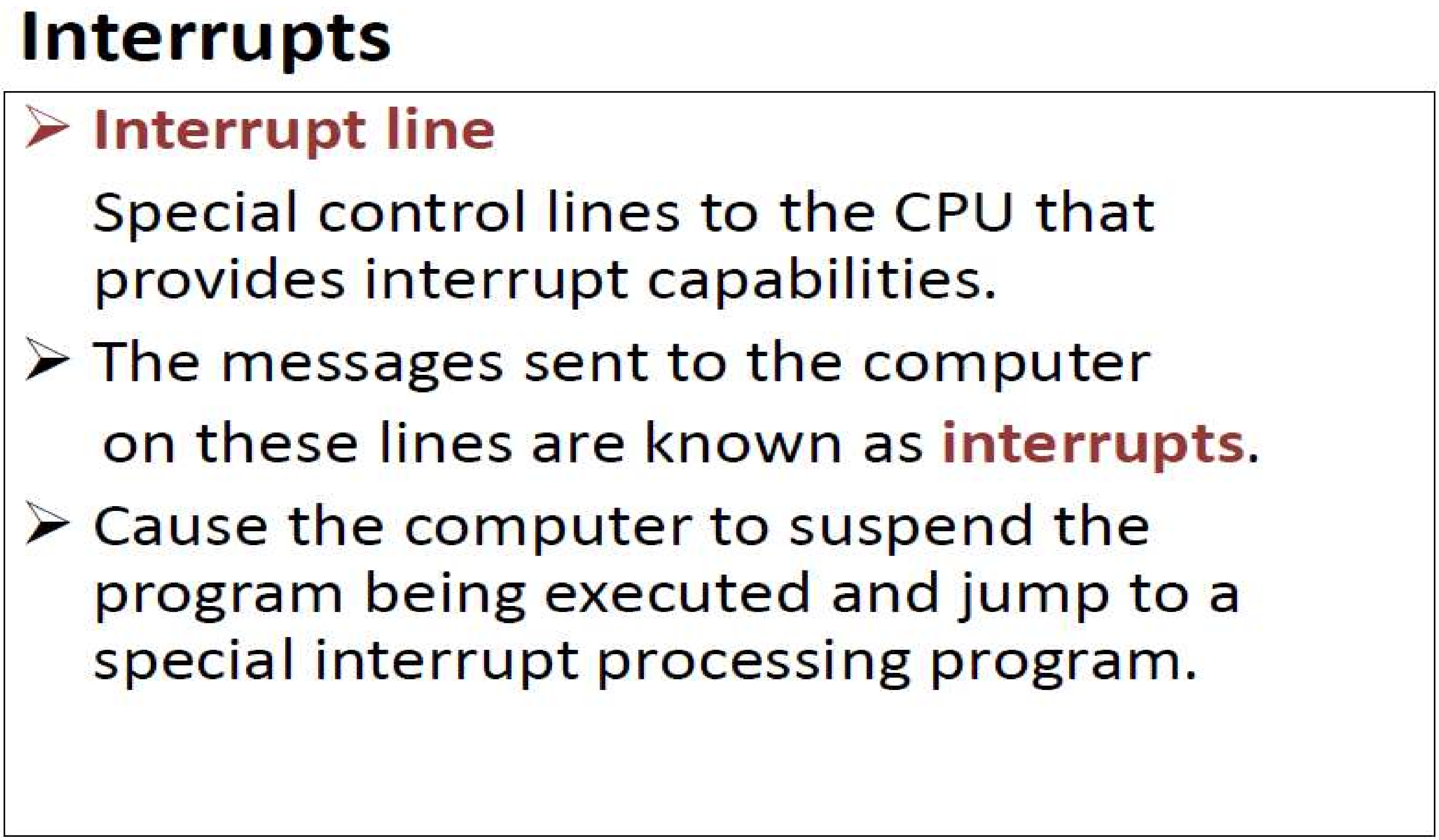


Interrupts

- Signal that causes the CPU to alter its normal flow on instruction execution.
  - Free CPU from waiting for events
  - Provides control for external input

- <u>Examples of Interrupt</u>
  - An unexpected user input
  - Illegal instructions
  - Multitasking, multiprocessing
  - An abnormal situation

4. Interrupts (Continued)

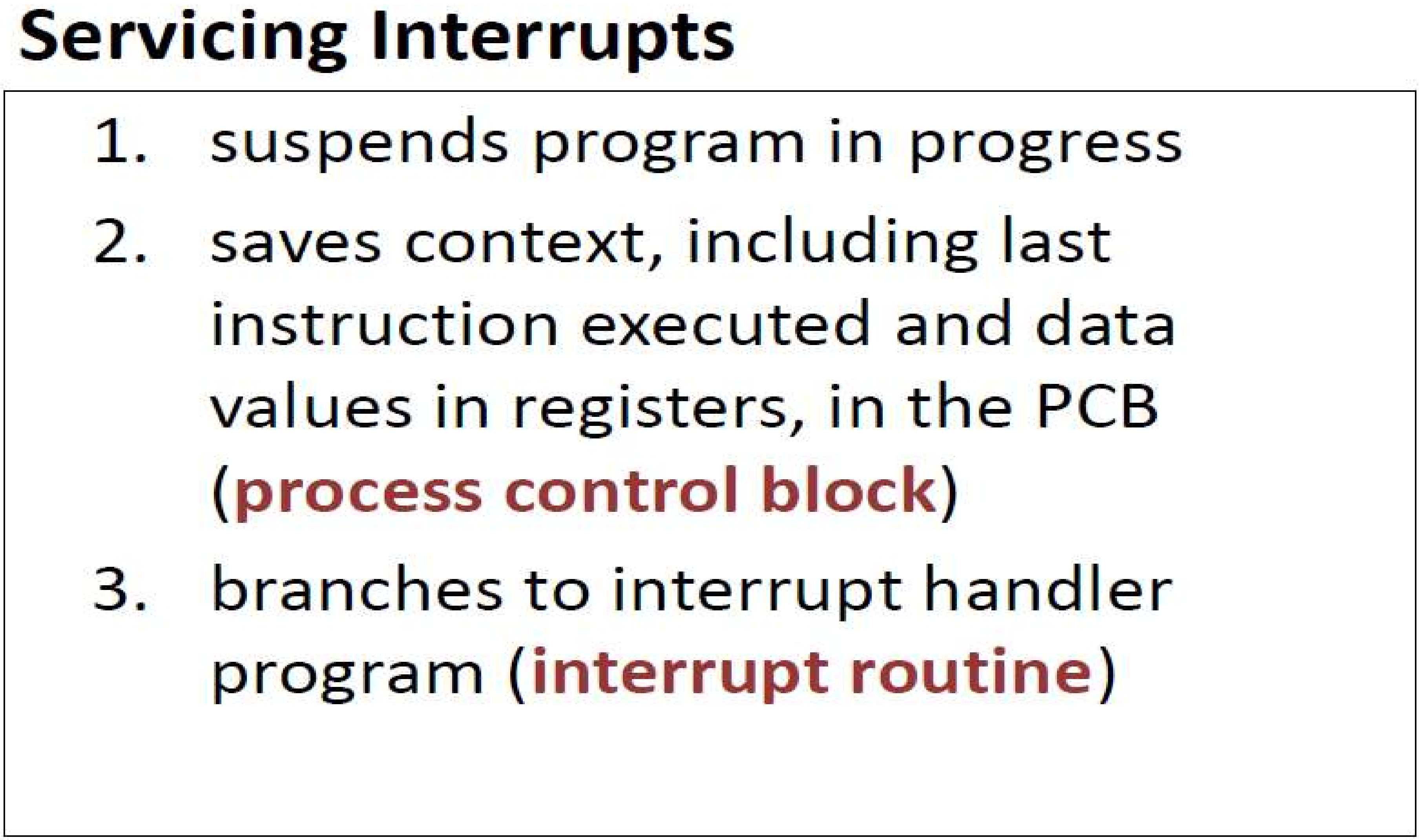


Interrupts

- ➢ **Interrupt line**

  Special control lines to the CPU that provides interrupt capabilities.
- ➢ The messages sent to the computer on these lines are known as **interrupts**.
- ➢ Cause the computer to suspend the program being executed and jump to a special interrupt processing program.

4. Interrupts (Continued)

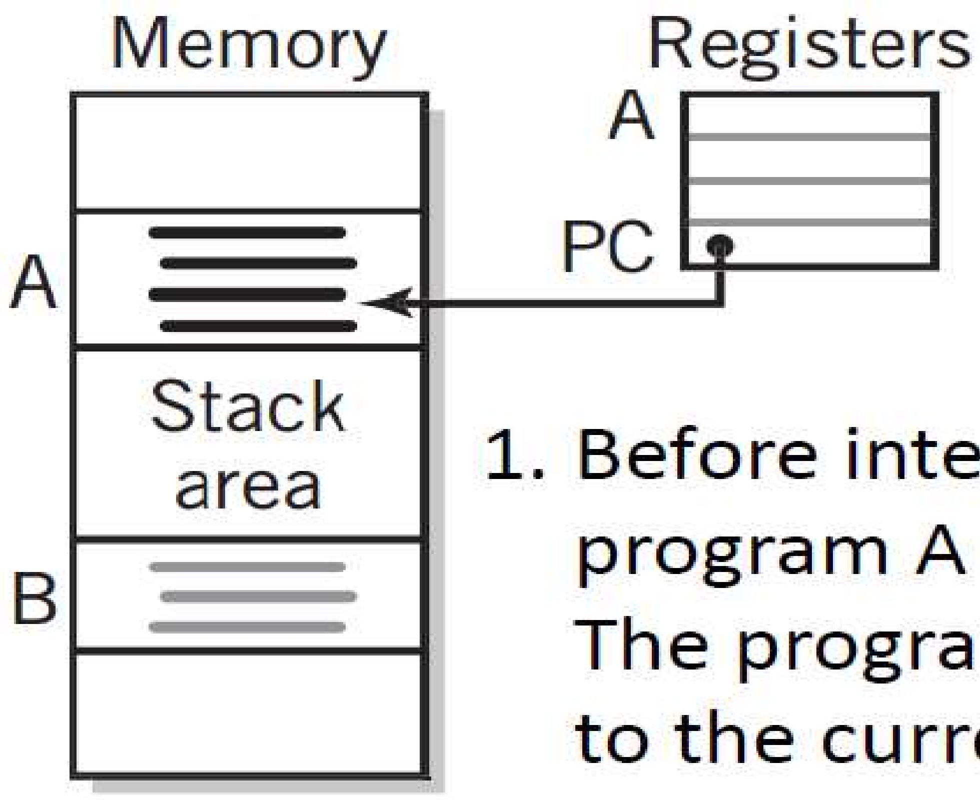


Servicing Interrupts

1. suspends program in progress
2. saves context, including last instruction executed and data values in registers, in the PCB (**process control block**)
3. branches to interrupt handler program (**interrupt routine**)

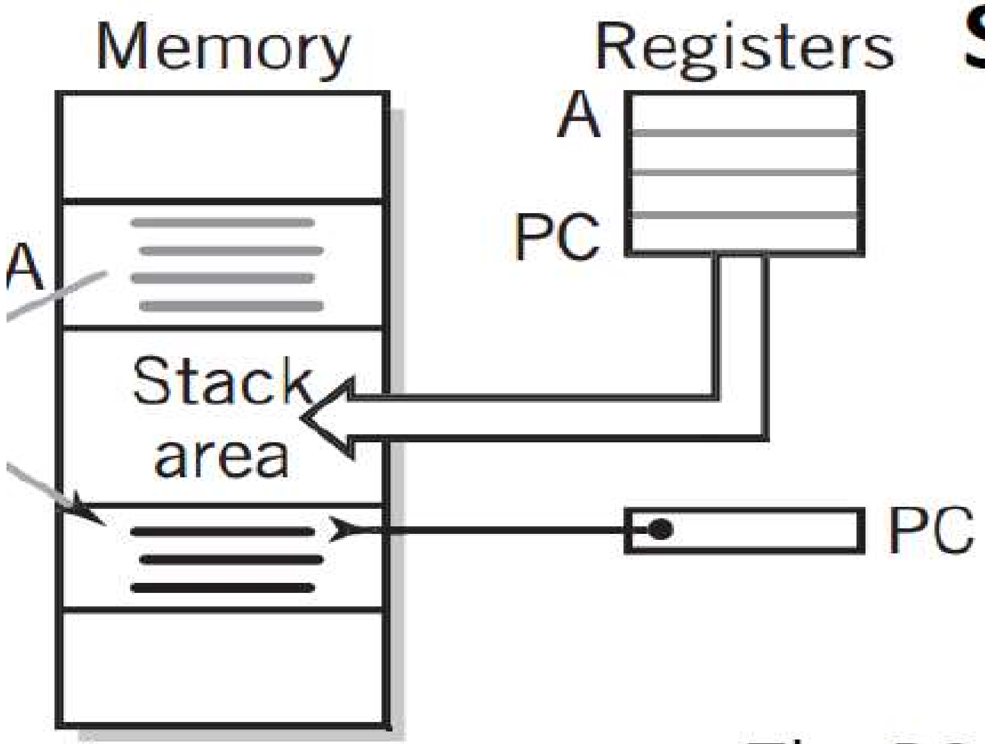
4. Interrupts (Continued)

Servicing Interrupts



1. Before interrupt arrives, program A is executing. The program counter points to the current instruction.

4. Interrupts (Continued)



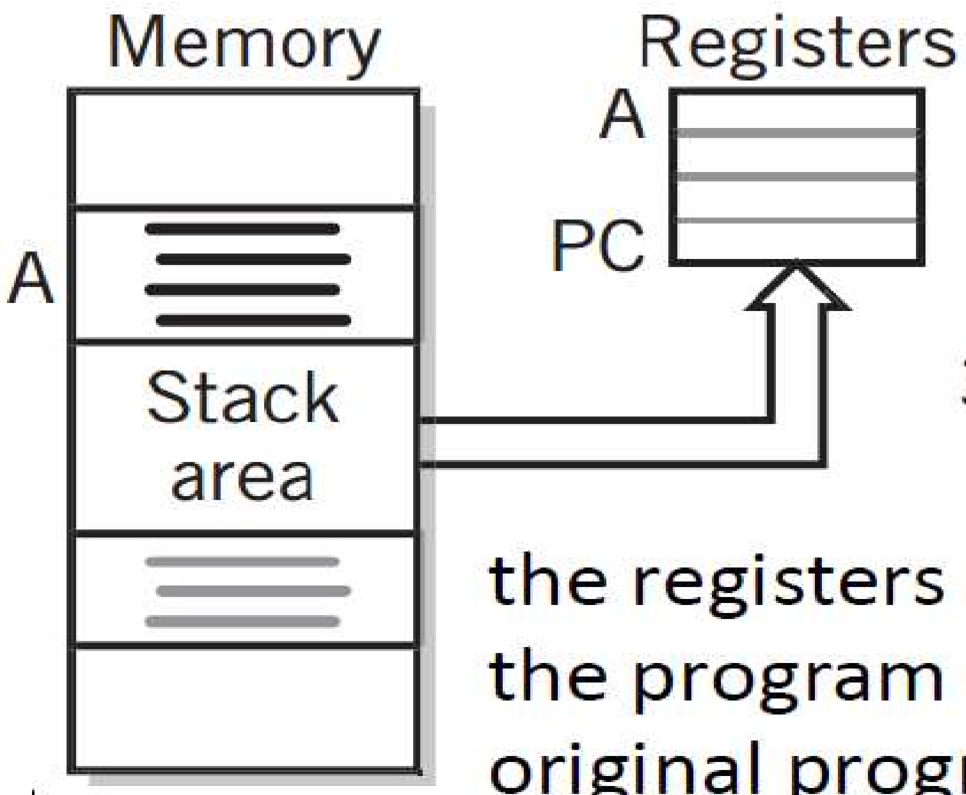
Servicing Interrupts

2. When the interrupt is received by the CPU, the current instruction is completed, all the registers are saved in the stack area (or PCB).

The PC is loaded with the starting location of program B, the interrupt handler program. This causes a jump to program B, which becomes the executing program.

4. Interrupts (Continued)

Servicing Interrupts



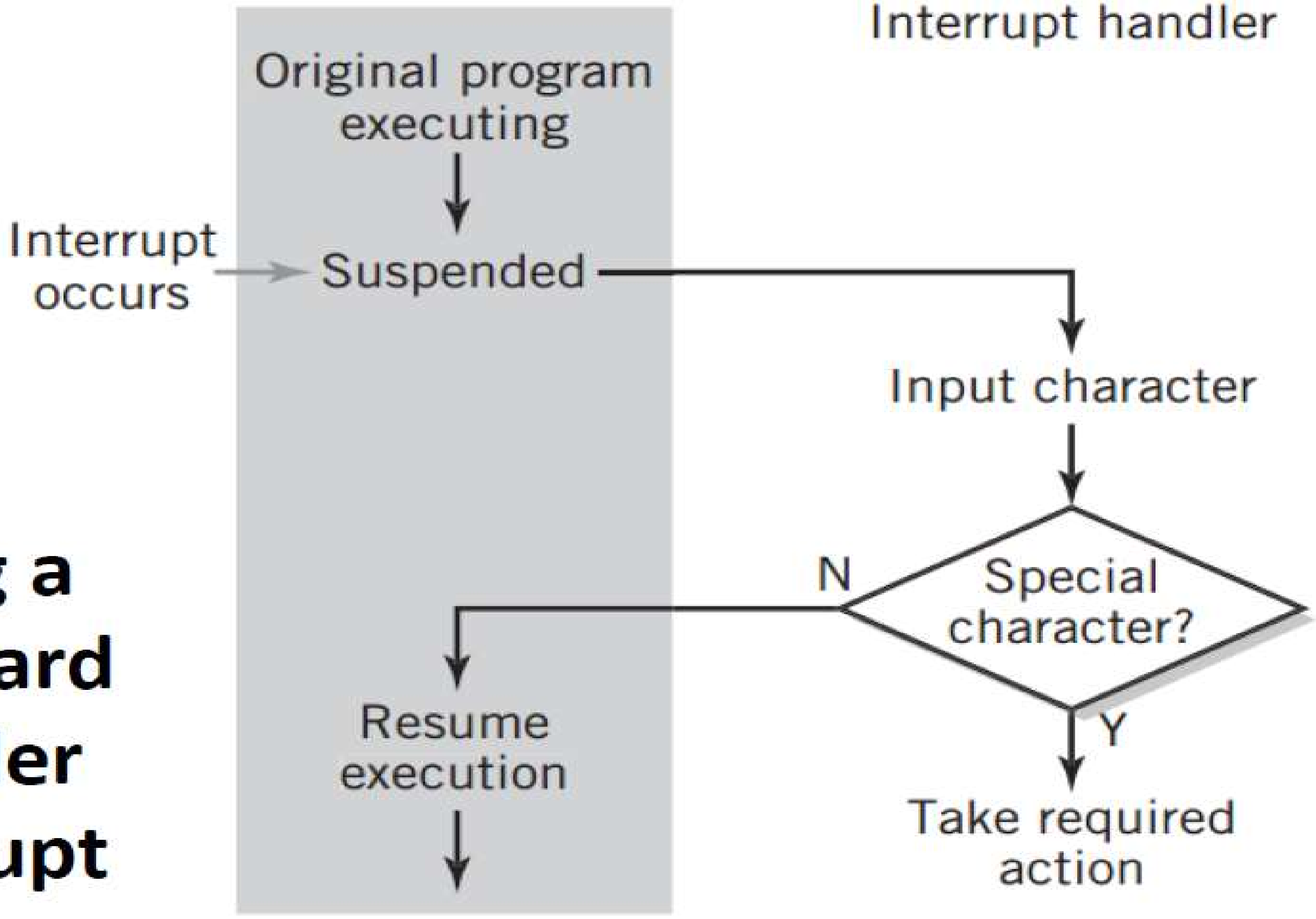
3. When the interrupt routine is complete, the registers are restored, including the program counter, and the original program resumes exactly where it left off.

4. Interrupts (Continued)

The uses of Interrupts

1. **THE INTERRUPT AS AN EXTERNAL EVENT NOTIFIER**
   - Notifying the CPU of external events that require action.
   - Free CPU from polling.
   - User to control the computer from input device.
   - **E.g.** Keyboard Input, Real-time or time-sensitive system

4. Interrupts (Continued)



Using a keyboard handler Interrupt

4. Interrupts (Continued)

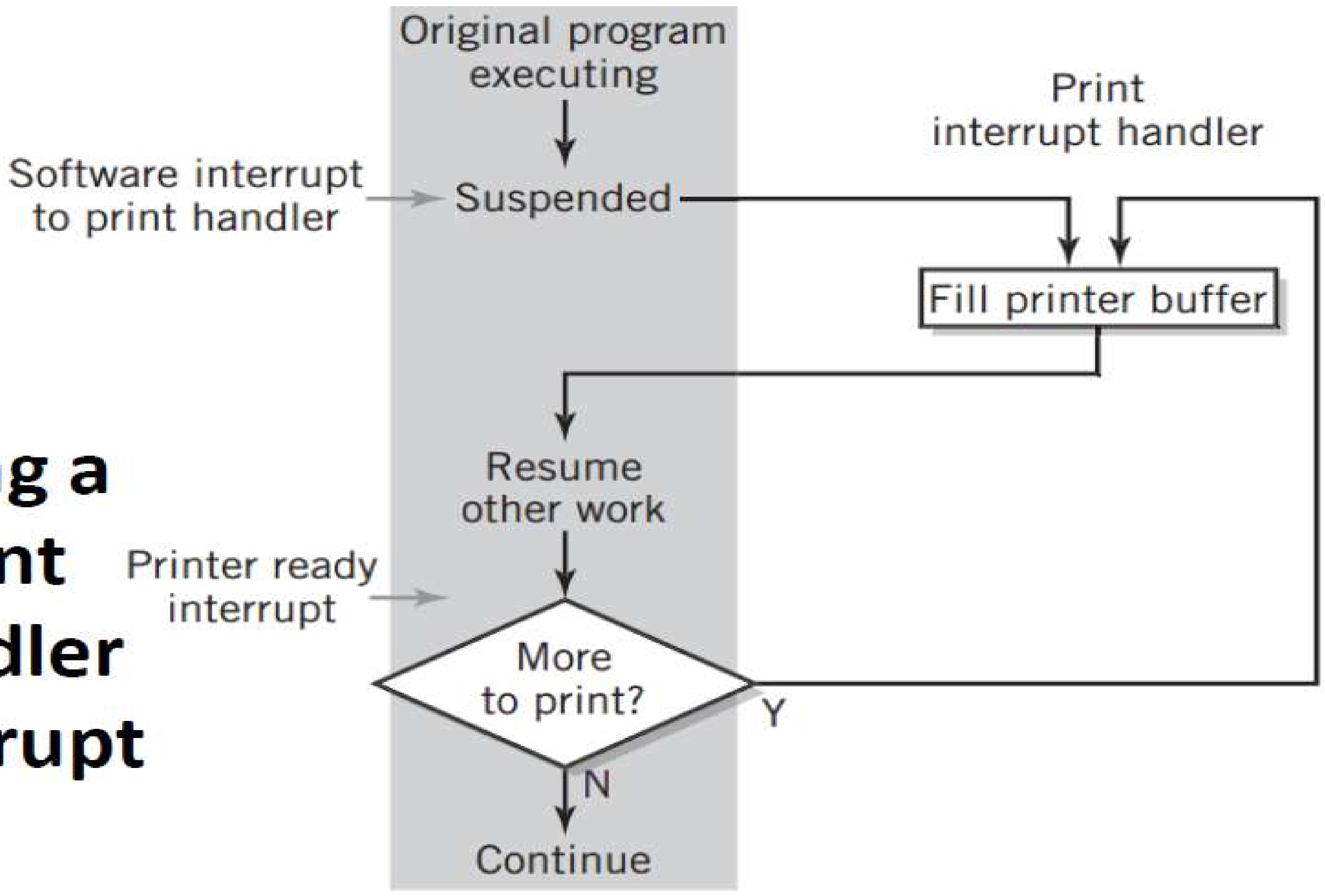
The uses of Interrupts

2. THE INTERRUPT AS A COMPLETION SIGNAL

- Controls the flow of data to the output devices.
- Notify the computer of the completion of a particular course of action.
- **Example :**
  - Printer ready - printer to control the flow of characters to the printer in an efficient way.

4. Interrupts (Continued)

Using a Print Handler Interrupt



4. Interrupts (Continued)

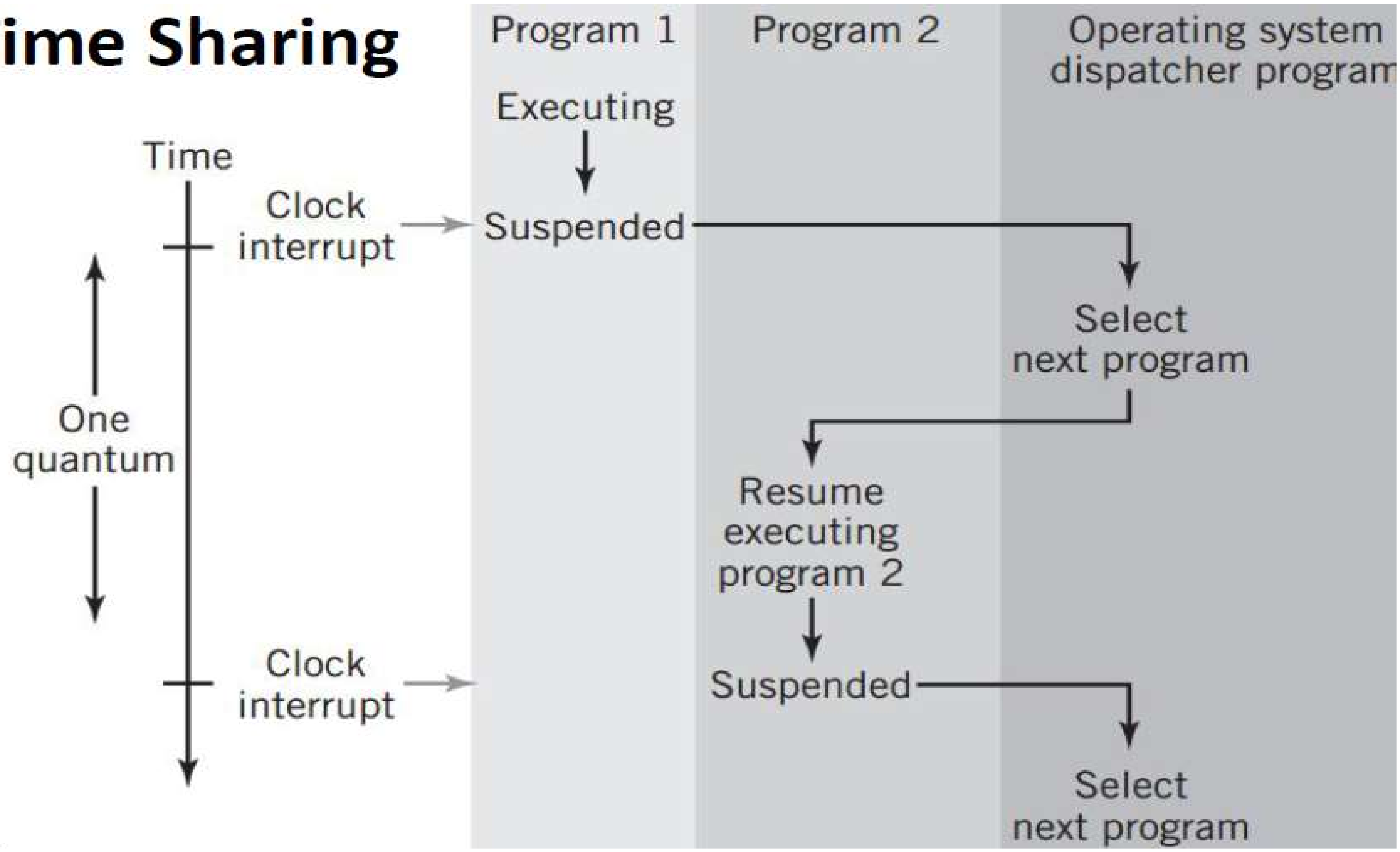
The uses of Interrupts

3. THE INTERRUPT AS A MEANS OF ALLOCATION CPU TIME

- Allocating CPU time to different programs that are sharing the CPU.
- **Example :**
  - Time sharing
  - Since the CPU can only execute one program at a time. Computer system share the CPU by allocating small segment of time to each program, in rapid rotation among them.

4. Interrupts (Continued)

Time Sharing



4. Interrupts (Continued)

The uses of Interrupts

4. THE INTERRUPT AS AN ABNORMAL EVENT INDICATOR

- Handle abnormal events that affect operation of the computer system.
- Usage is similar to external input events, but in this case, the events are directed at the problems within the computer system itself.
- **Example :**
  - Power failure, Illegal instruction, Hardware error.

4. Interrupts (Continued)

Software Interrupts

- Generated by Operating system (Windows Kernel).
- The interrupt instruction works in the same way as a hardware interrupt, saving appropriate registers and transferring control to an interrupt handling procedure.
- **Example :**
  - Handling timer expiration
  - Exception Handler (Try-Catch Exceptions)

4. Interrupts (Continued)

Multiple Interrupts and prioritization

- ➢ When an interrupt occurs
  - ▪ Are there other interrupts already waiting service?
  - ▪ How does the computer identify the interrupting device?
- ➢ Can be handled by assigning priorities to each interrupt.

4. Interrupts (Continued)

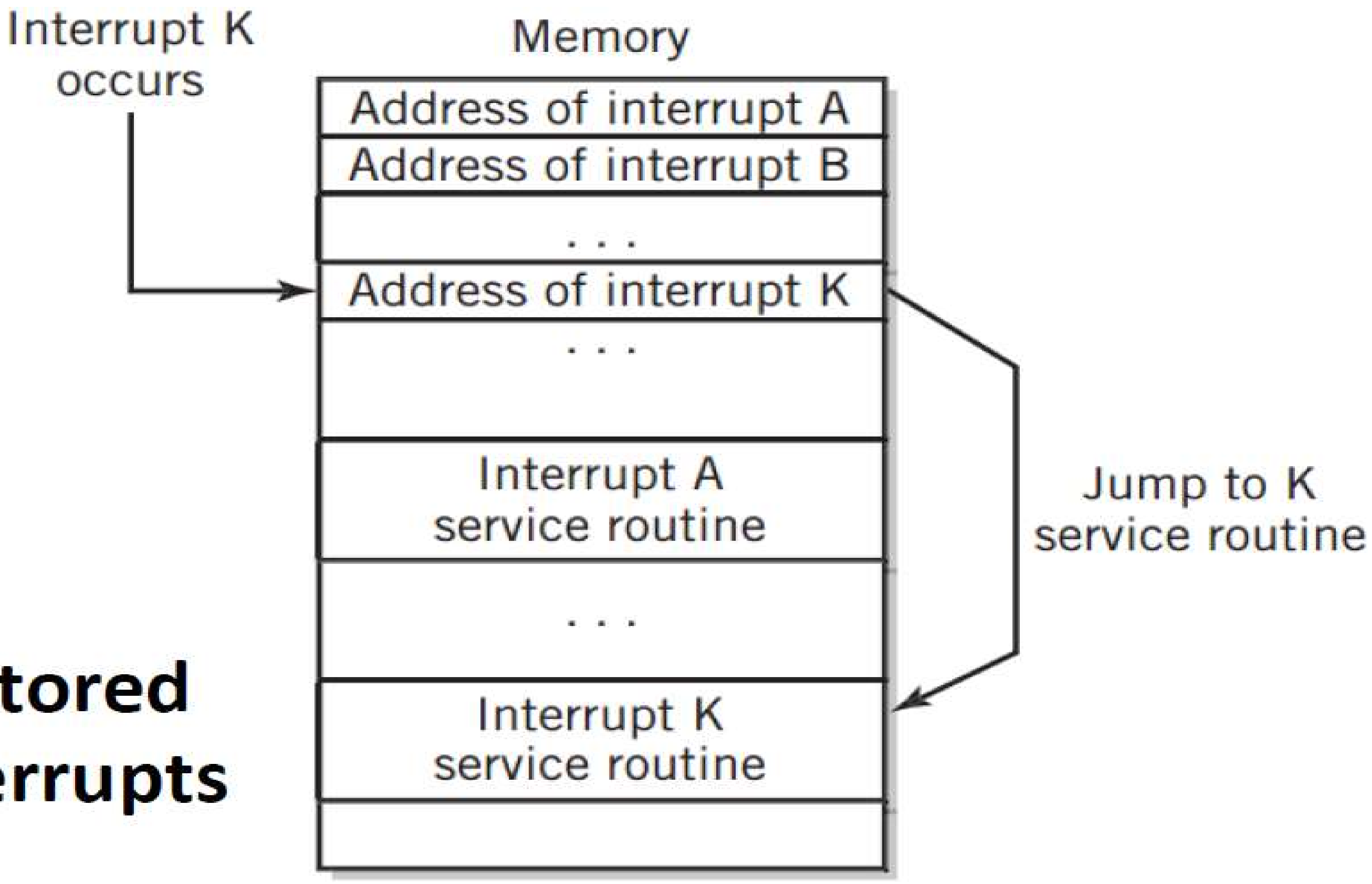
Multiple Interrupts and prioritization

➢ Processing methods to identify devices that initiate the interrupt :

- ***Vectored interrupt :*** the address of the interrupting device is included as part of the interrupt.
- ***Polled interrupt :*** the general interrupt that shared by all devices.

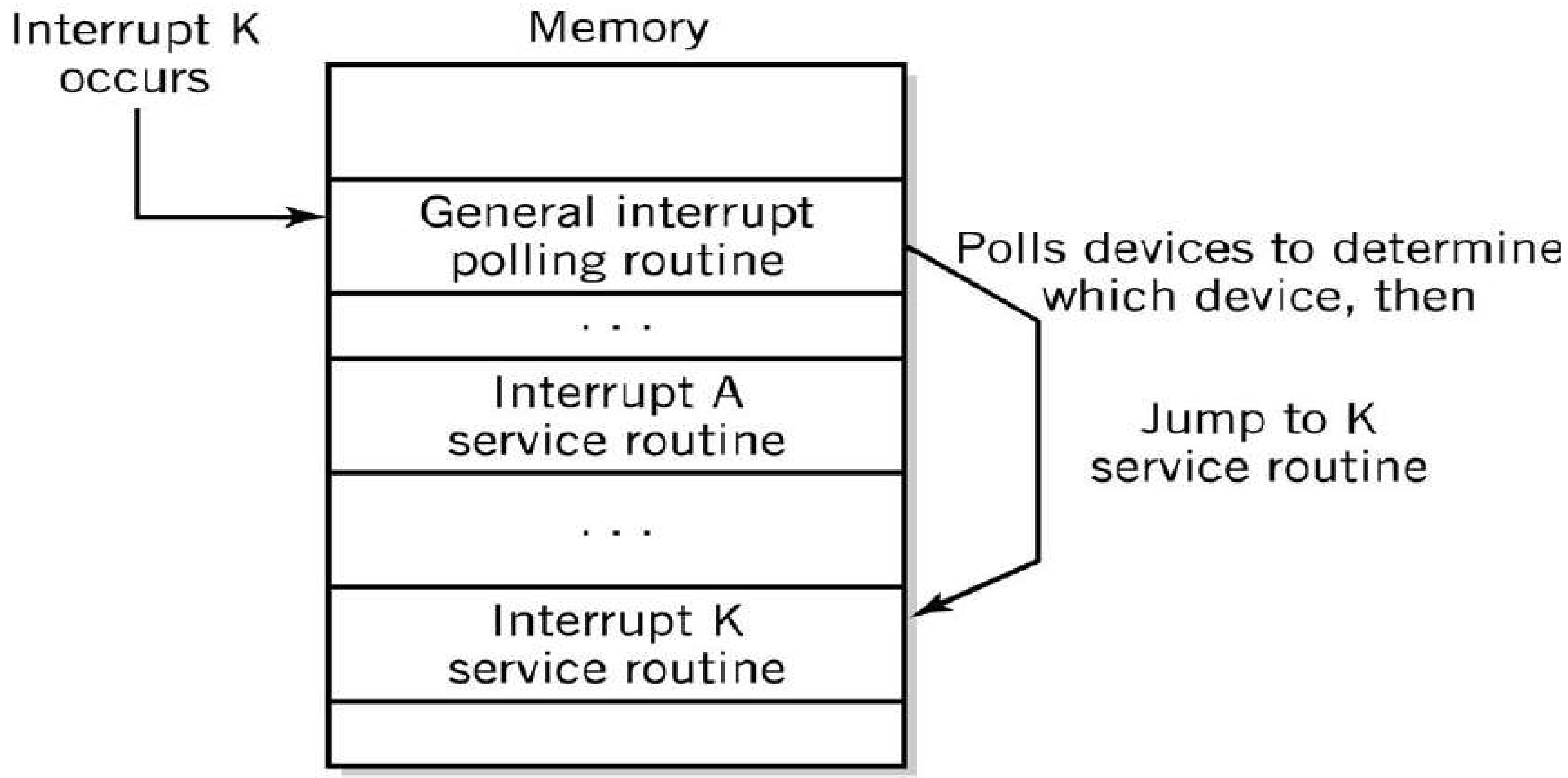
  Identifies the Interrupting device by polling each device.

4. Interrupts (Continued)



Vectored interrupts

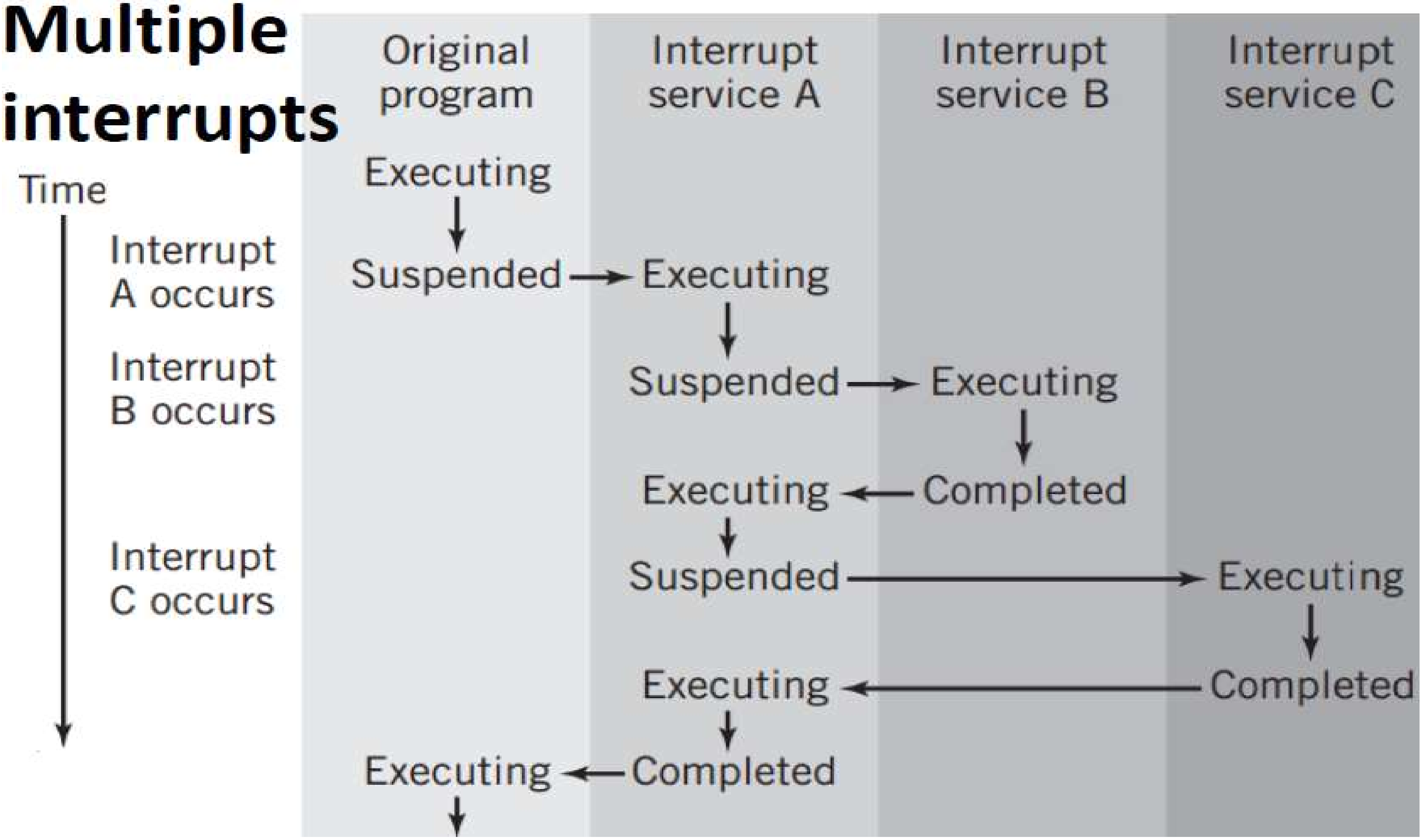
4. Interrupts (Continued)


Interrupt K occurs
Memory
General interrupt polling routine
. . .
Interrupt A service routine
. . .
Interrupt K service routine
Polls devices to determine which device, then
Jump to K service routine

Polled interrupts

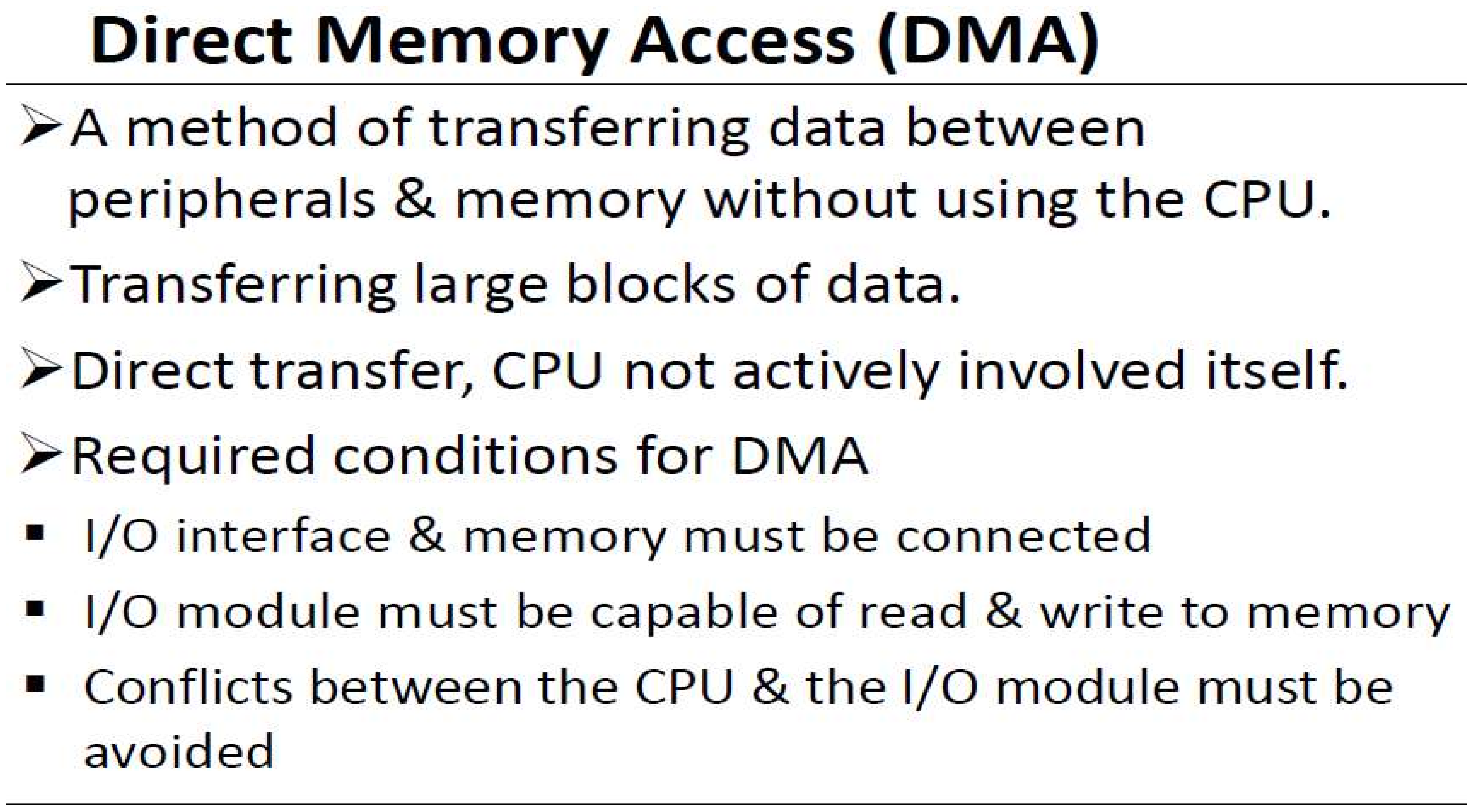
4. Interrupts (Continued)

Multiple interrupts



5. Direct Memory access (DMA)

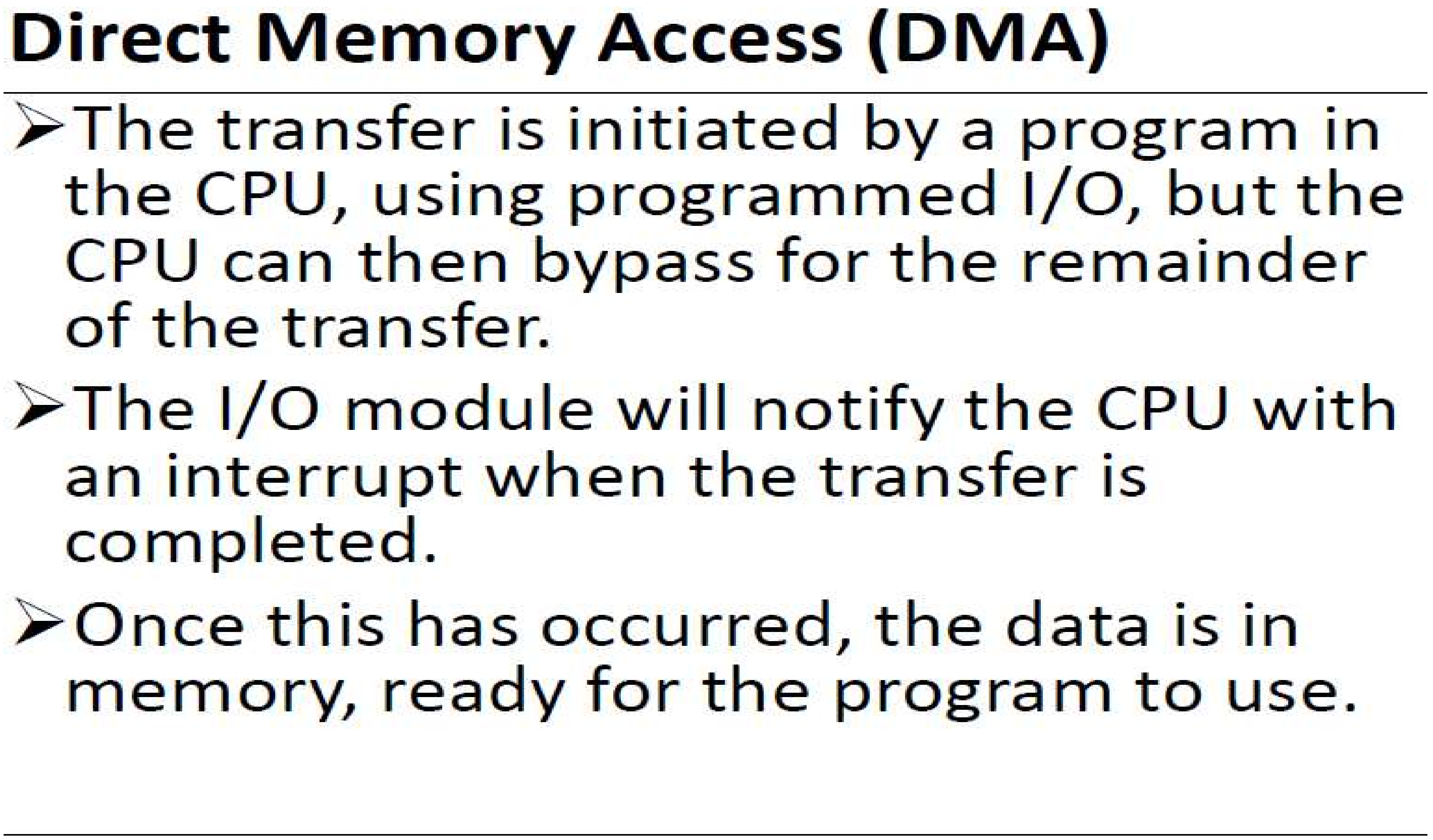
5. Direct Memory access (DMA)



Direct Memory Access (DMA)

- A method of transferring data between peripherals & memory without using the CPU.
- Transferring large blocks of data.
- Direct transfer, CPU not actively involved itself.
- Required conditions for DMA
  - I/O interface & memory must be connected
  - I/O module must be capable of read & write to memory
  - Conflicts between the CPU & the I/O module must be avoided

5. Direct Memory access (DMA) (Continued)



Direct Memory Access (DMA)

- The transfer is initiated by a program in the CPU, using programmed I/O, but the CPU can then bypass for the remainder of the transfer.
- The I/O module will notify the CPU with an interrupt when the transfer is completed.
- Once this has occurred, the data is in memory, ready for the program to use.

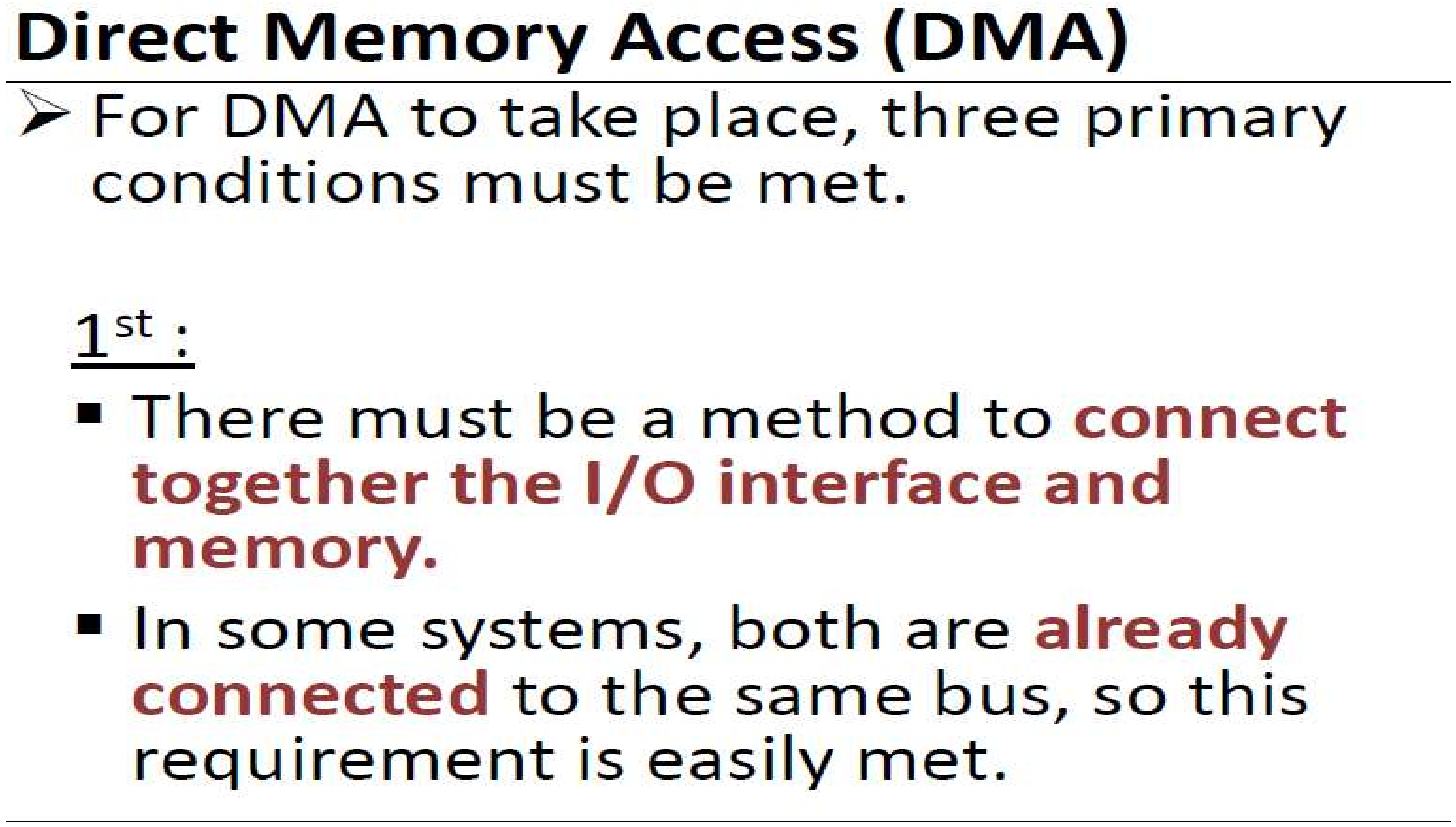
5. Direct Memory access (DMA) (Continued)

Direct Memory Access (DMA)

E.g. : Consider a program that sorts a block of numbers.

- To operate efficiently,
- Transfer the entire block of numbers from HDD to memory
- Perform sorting in the memory.
- Transfer the data back to HDD.

5. Direct Memory access (DMA) (Continued)



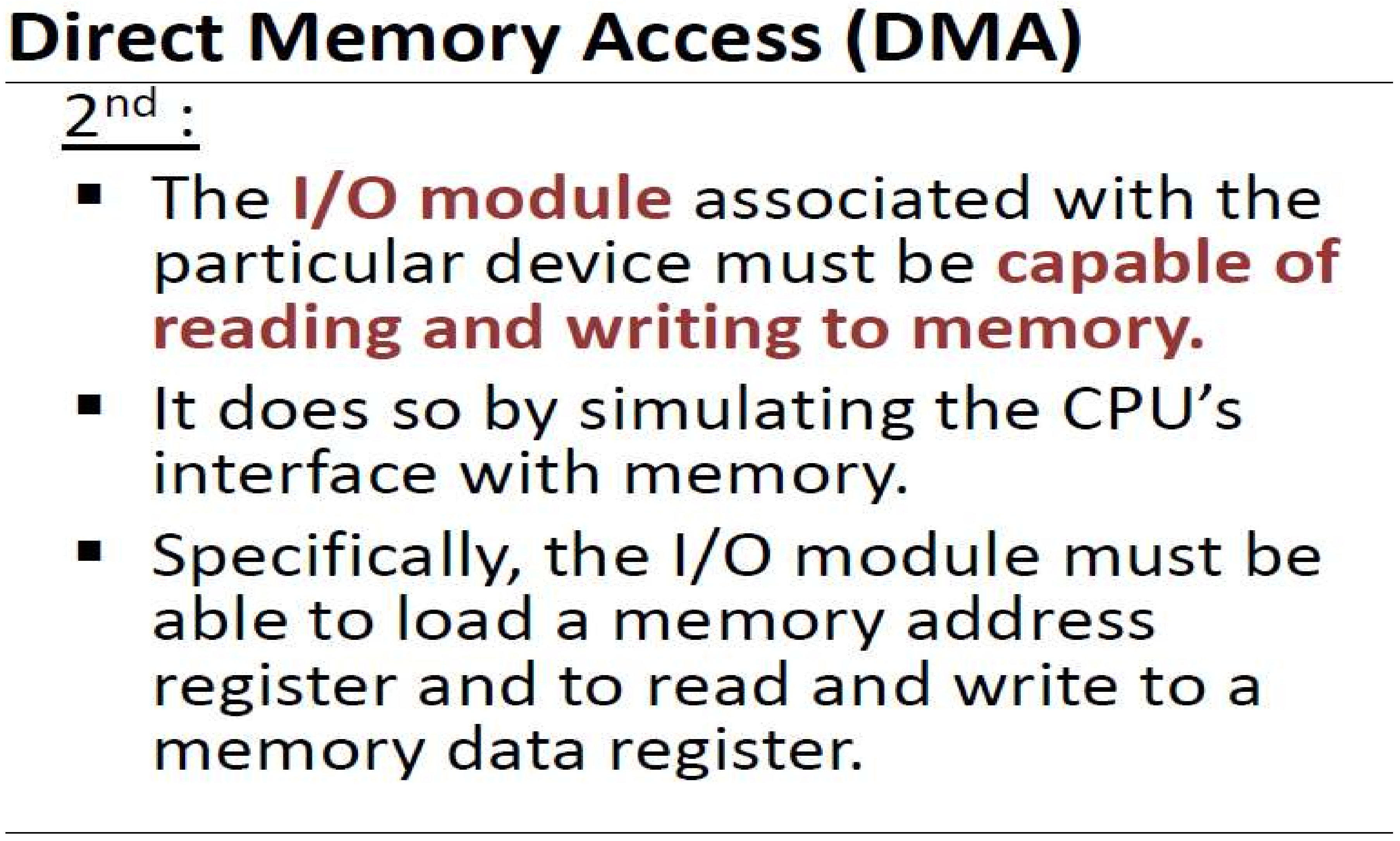
Direct Memory Access (DMA)

- For DMA to take place, three primary conditions must be met.

1st :

- There must be a method to **connect together the I/O interface and memory.**
- In some systems, both are **already connected** to the same bus, so this requirement is easily met.

5. Direct Memory access (DMA) (Continued)

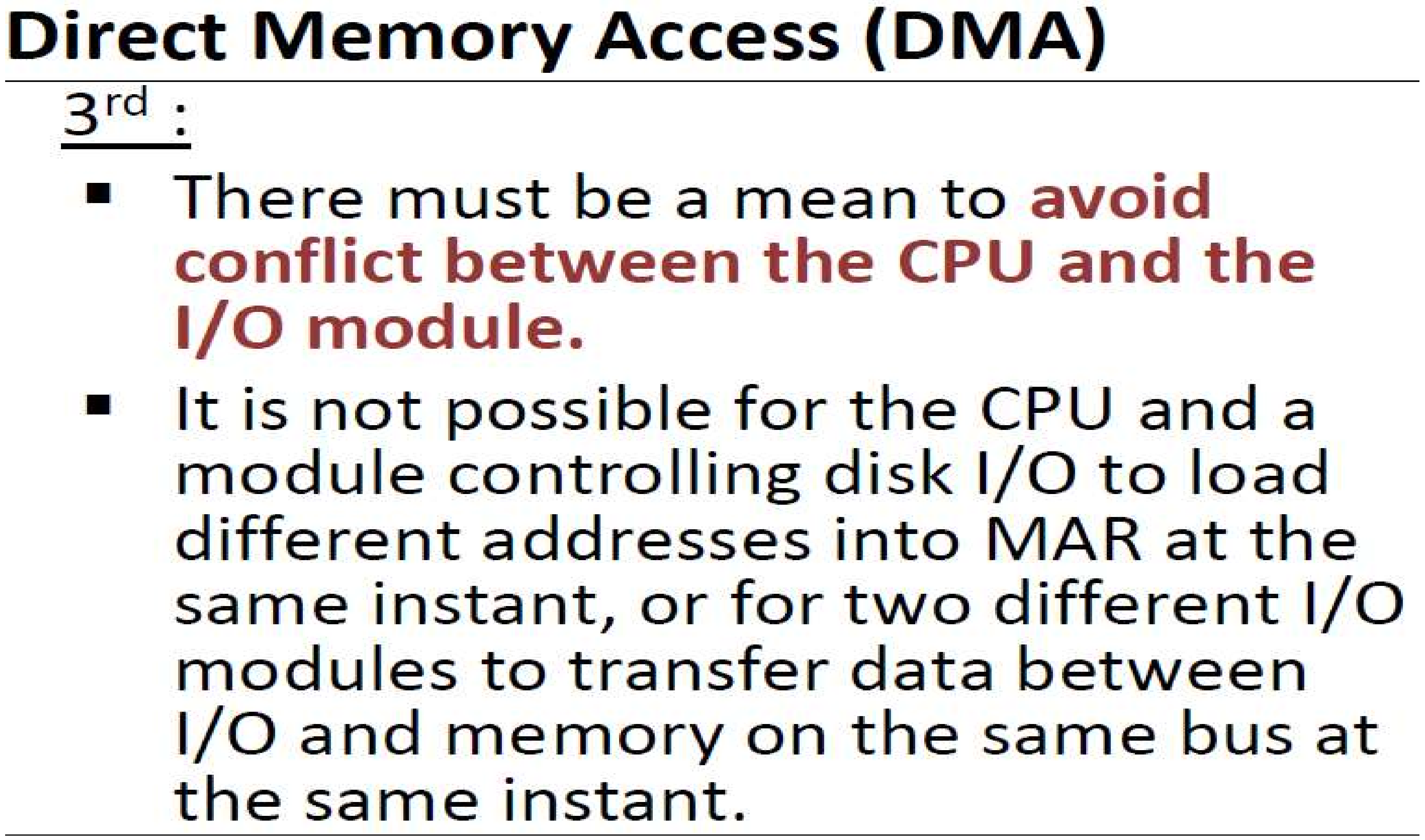


Direct Memory Access (DMA)

<u>2nd :</u>

- The **I/O module** associated with the particular device must be **capable of reading and writing to memory.**
- It does so by simulating the CPU's interface with memory.
- Specifically, the I/O module must be able to load a memory address register and to read and write to a memory data register.

5. Direct Memory access (DMA) (Continued)

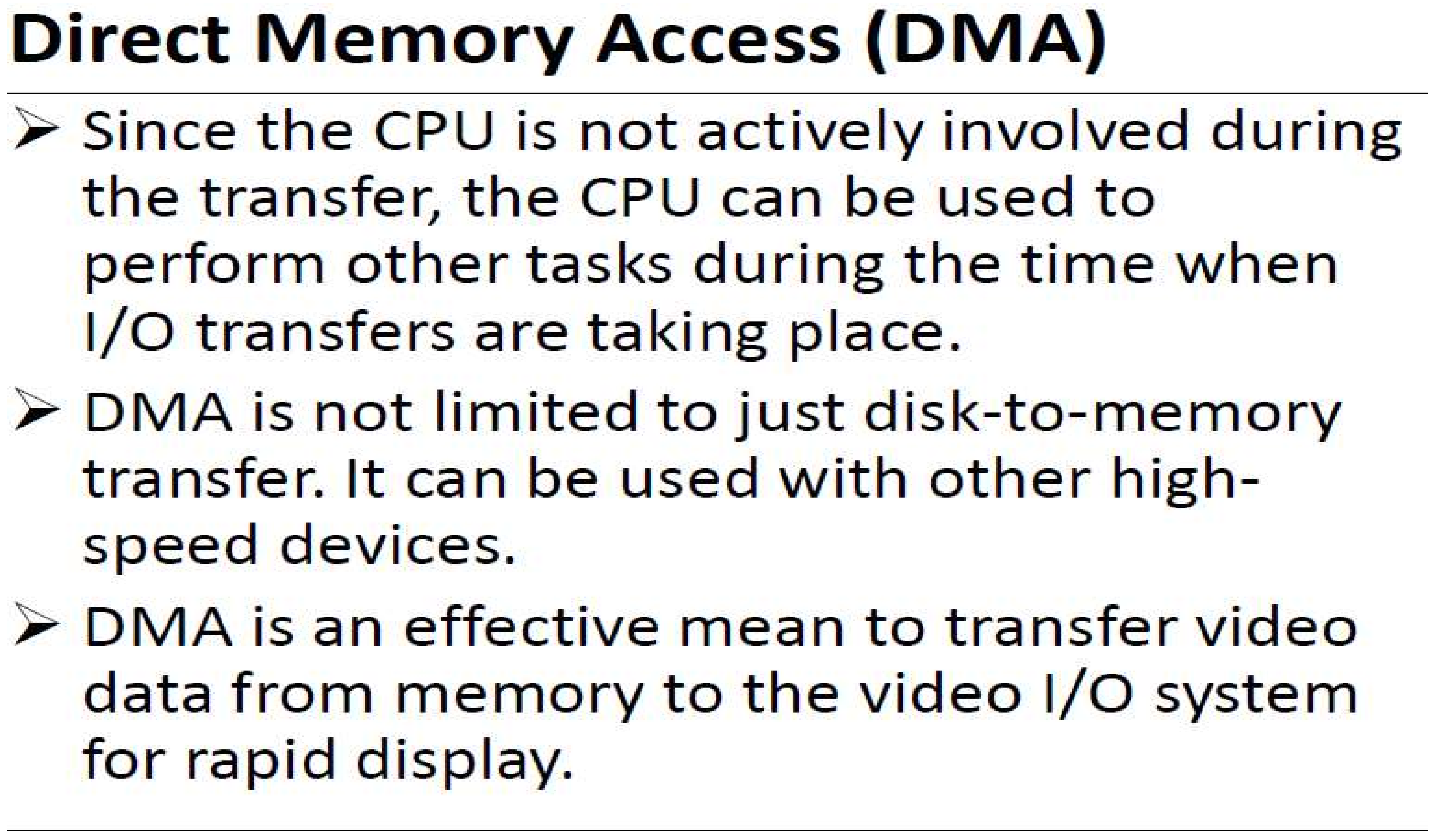


Direct Memory Access (DMA)

3rd :

- There must be a mean to **avoid conflict between the CPU and the I/O module.**
- It is not possible for the CPU and a module controlling disk I/O to load different addresses into MAR at the same instant, or for two different I/O modules to transfer data between I/O and memory on the same bus at the same instant.

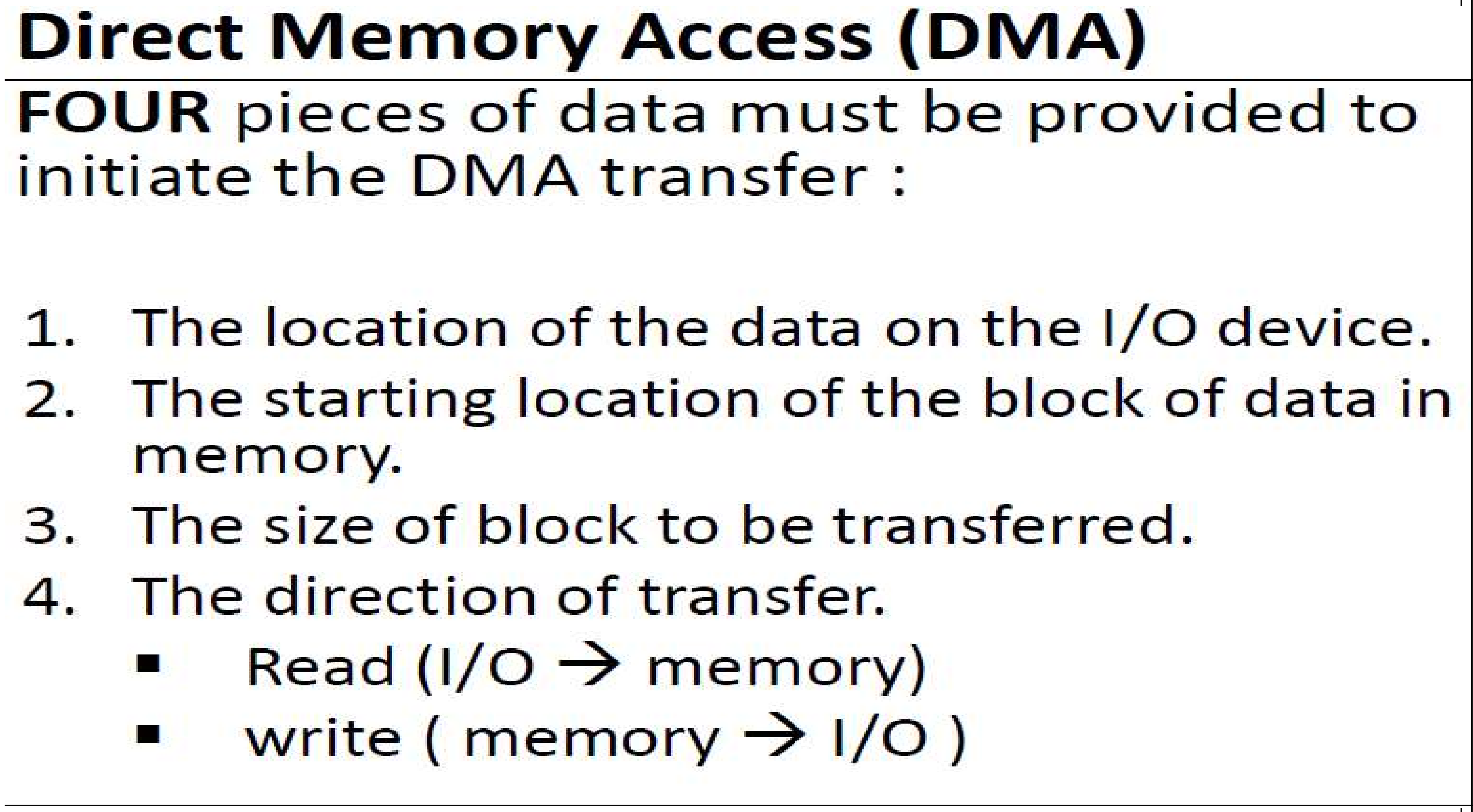
5. Direct Memory access (DMA) (Continued)



Direct Memory Access (DMA)

- ➢ Since the CPU is not actively involved during the transfer, the CPU can be used to perform other tasks during the time when I/O transfers are taking place.
- ➢ DMA is not limited to just disk-to-memory transfer. It can be used with other high-speed devices.
- ➢ DMA is an effective mean to transfer video data from memory to the video I/O system for rapid display.

5. Direct Memory access (DMA) (Continued)

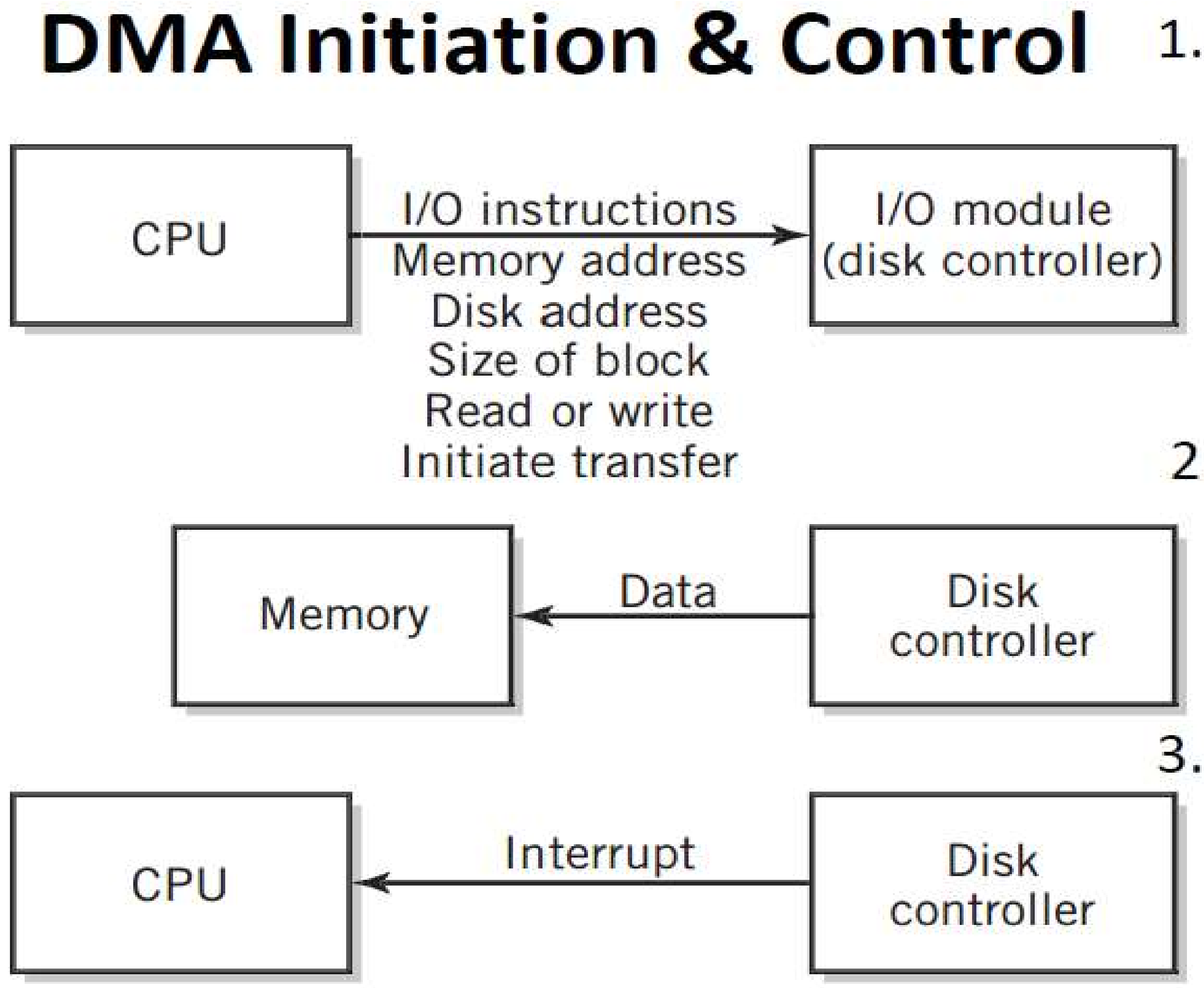


Direct Memory Access (DMA)

FOUR pieces of data must be provided to initiate the DMA transfer :

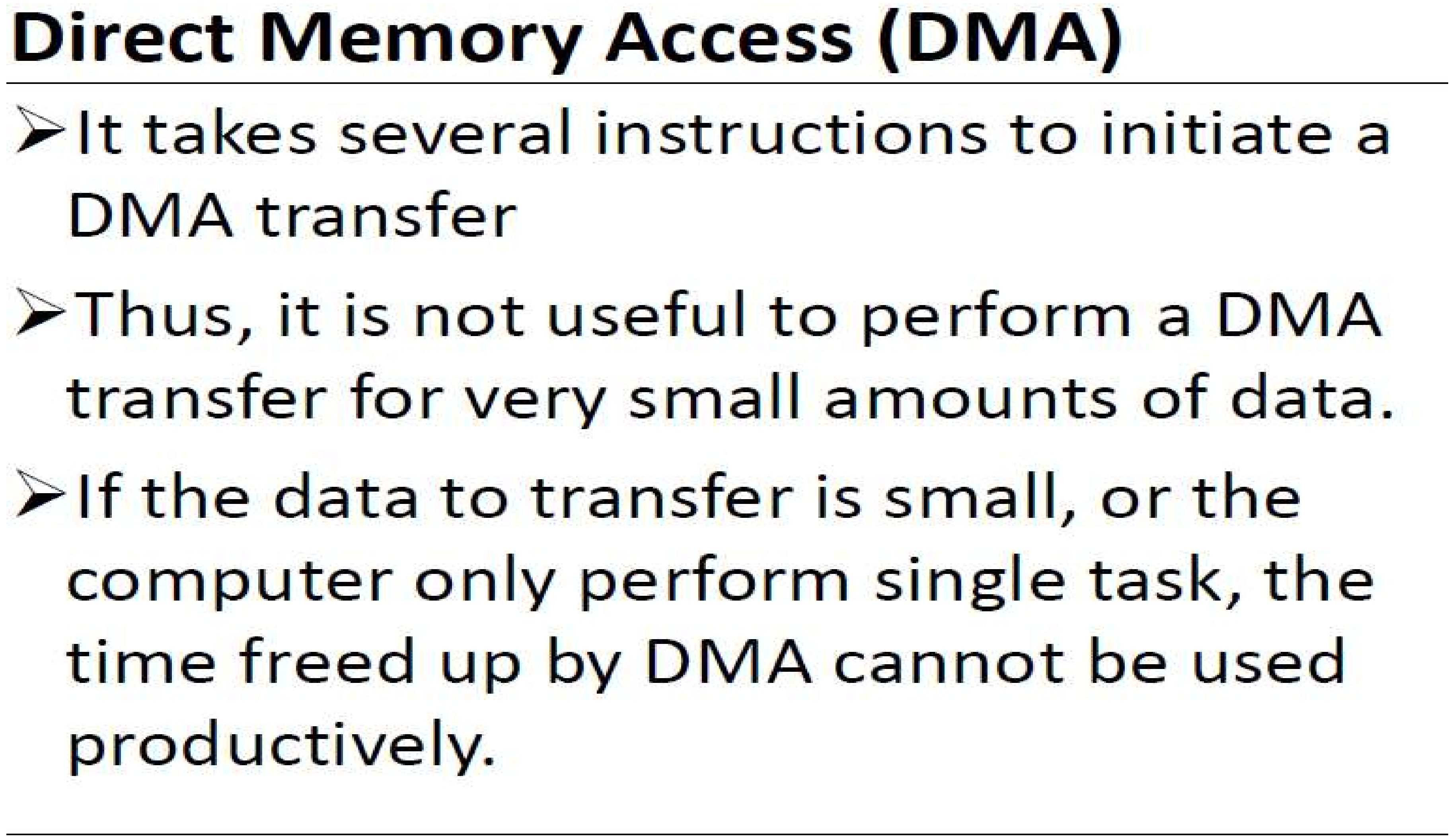
1. The location of the data on the I/O device.
2. The starting location of the block of data in memory.
3. The size of block to be transferred.
4. The direction of transfer.
    - Read (I/O → memory)
    - write (memory → I/O)

5. Direct Memory access (DMA) (Continued)



1. Programmed I/O used to prepare I/O module for transfer by providing required information and initiating transfer.
2. DMA transfer. In this case, data is transferred from disk to memory.
3. Upon completion, disk controller sends completion interrupt to CPU.

5. Direct Memory access (DMA) (Continued)



Direct Memory Access (DMA)

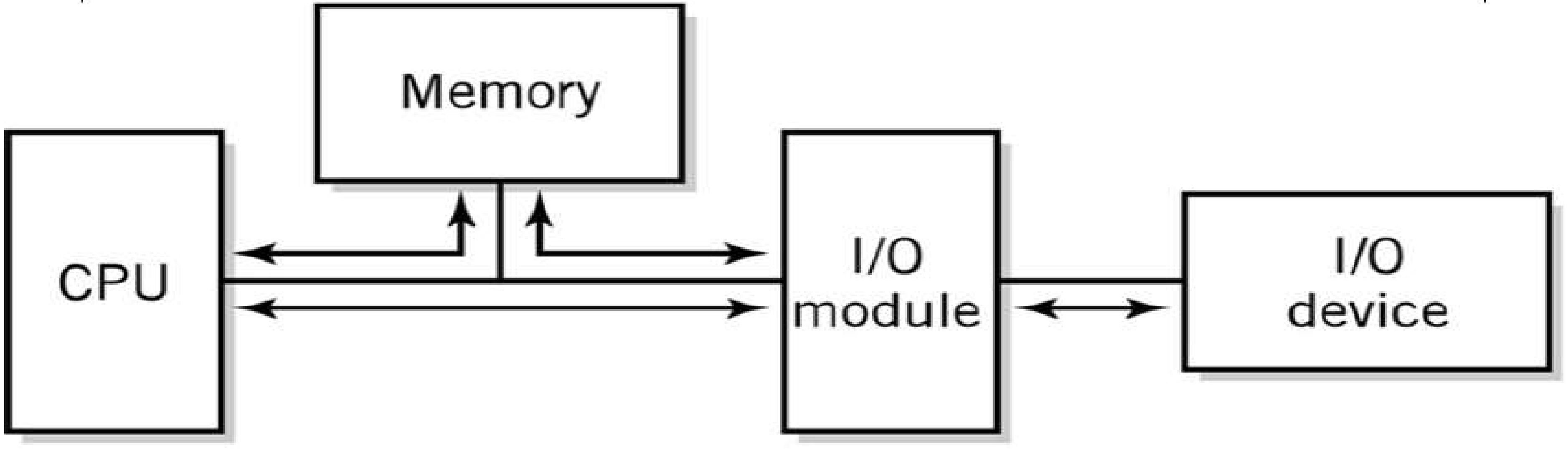
- It takes several instructions to initiate a DMA transfer
- Thus, it is not useful to perform a DMA transfer for very small amounts of data.
- If the data to transfer is small, or the computer only perform single task, the time freed up by DMA cannot be used productively.

6. I/O System Architecture

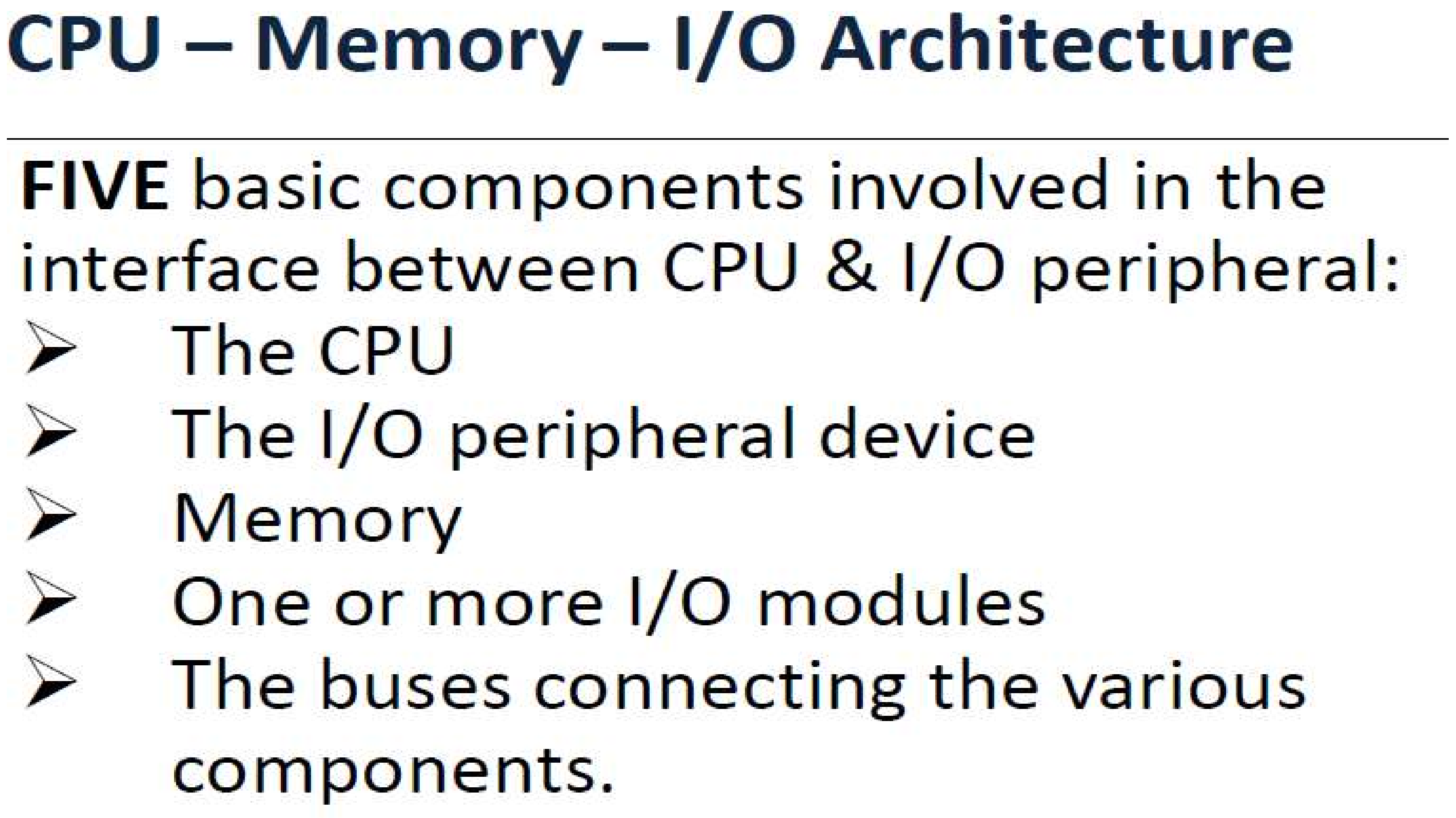
6. I/O System Architecture

CPU – Memory – I/O Architecture

- Basic CPU-Memory-I/O Pathway



6. I/O System Architecture (Continued)

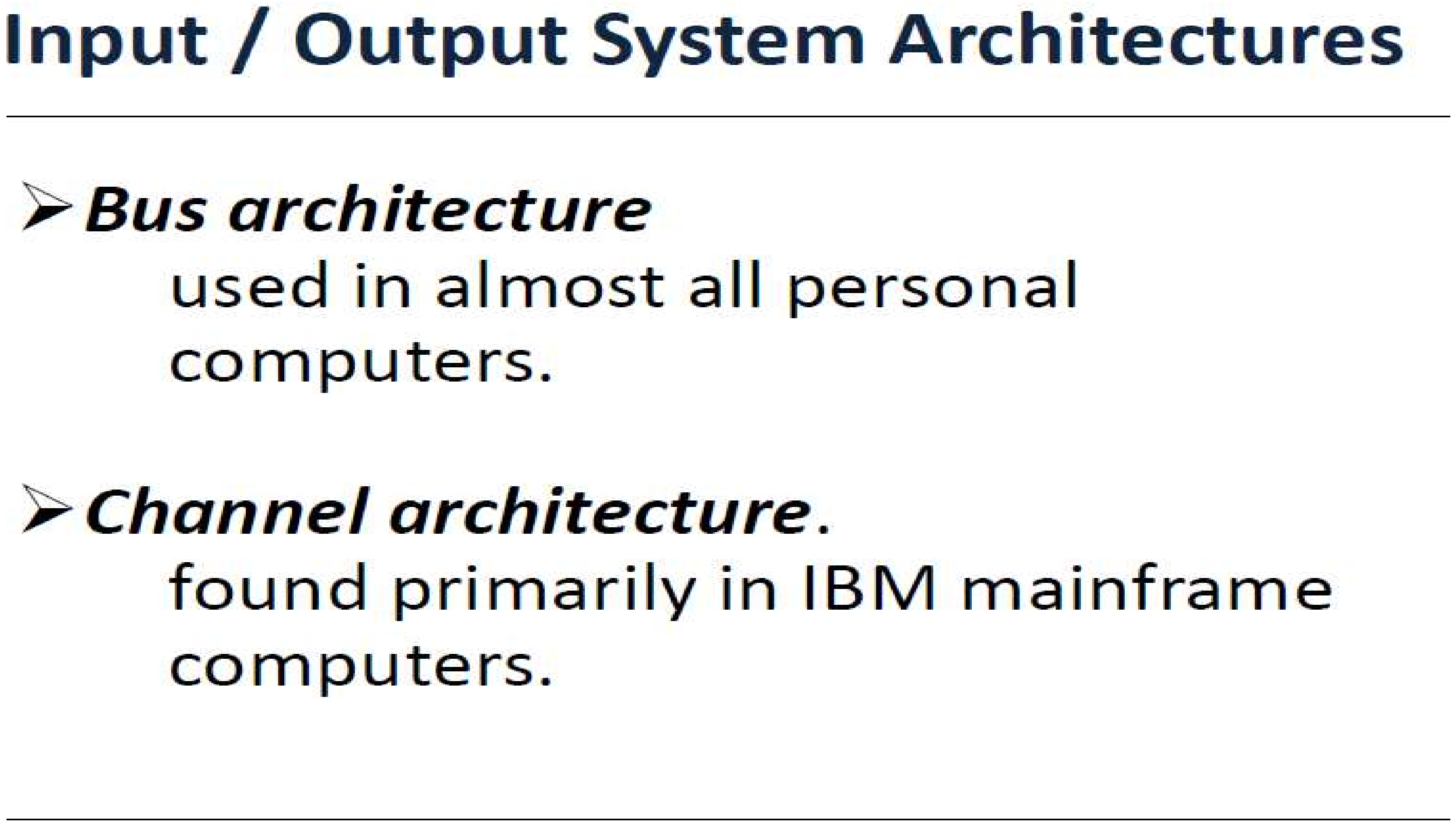


CPU – Memory – I/O Architecture

FIVE basic components involved in the interface between CPU & I/O peripheral:

- ➢ The CPU
- ➢ The I/O peripheral device
- ➢ Memory
- ➢ One or more I/O modules
- ➢ The buses connecting the various components.

6. I/O System Architecture (Continued)



Input / Output System Architectures

- ***Bus architecture***
  used in almost all personal computers.

- ***Channel architecture*.**
  found primarily in IBM mainframe computers.

6. I/O System Architecture (Continued)

I/O Bus Architecture

- Bus Architecture

  Backbone for connections of various components, memory and I/O, to the CPU.

- Simplest form

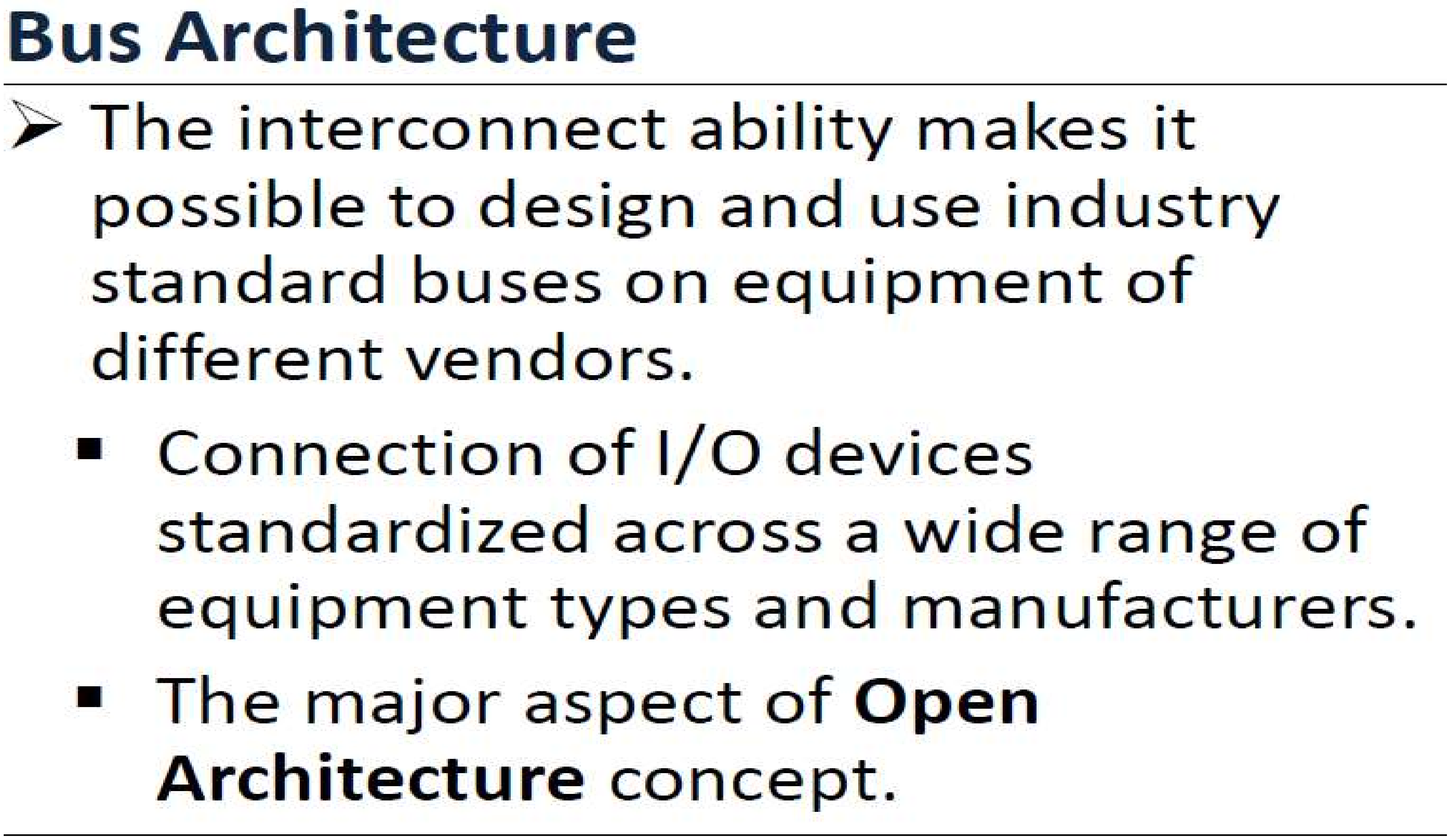
  Single system bus connects the CPU to memory and to all various I/O modules that control I/O devices.

6. I/O System Architecture (Continued)

Bus Architecture

- ➢Consists of a number of interconnected buses.
  - CPU bus, PCI bus, ISA bus
- ➢These buses are interconnected by Bus interfaces (expansion bus interface, bus bridges)
  - Expand the flexibility by converting bus signals from one to another.

6. I/O System Architecture (Continued)

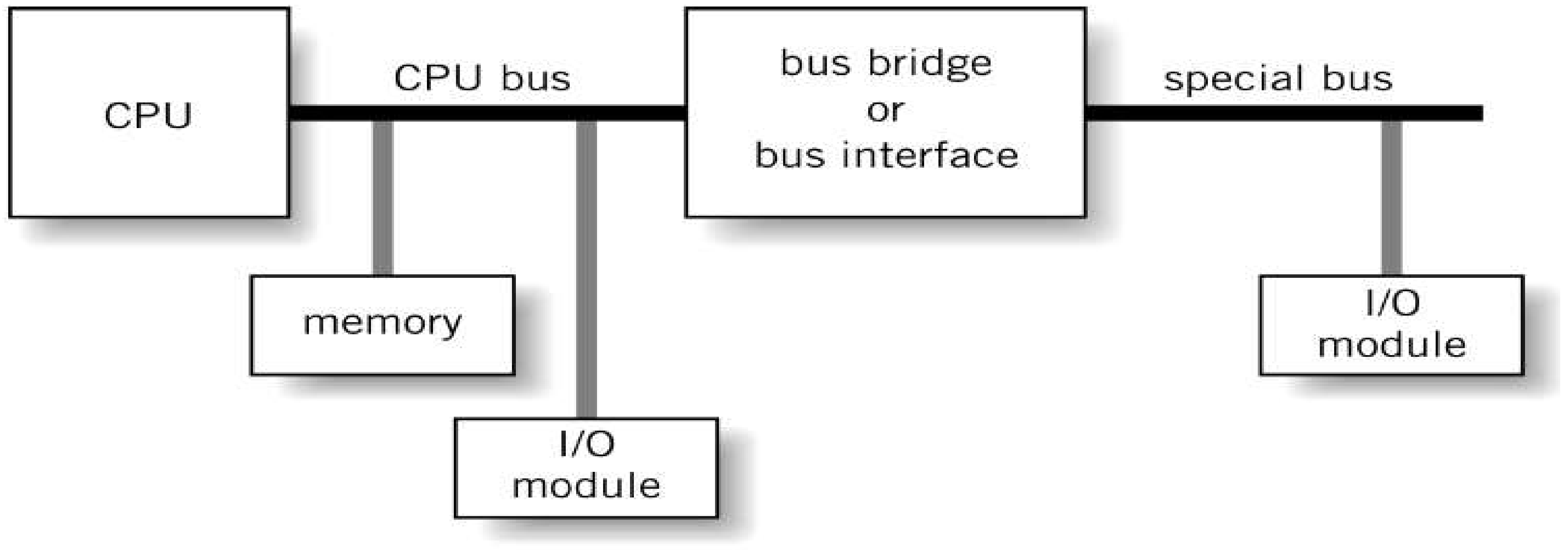


Bus Architecture

- The interconnect ability makes it possible to design and use industry standard buses on equipment of different vendors.
  - Connection of I/O devices standardized across a wide range of equipment types and manufacturers.
  - The major aspect of **Open Architecture** concept.

6. I/O System Architecture (Continued)

Bus Configuration



6. I/O System Architecture (Continued)

Channel Architecture

- Used in all IBM mainframe computers since late 70s
- **Channel subsystem**
  - Separate I/O processor that serves as a separate CPU for I/O operations
  - Channel control words
  - Programs stored in memory, independent of CPU
  - Transfer data between memory and an I/O device using DMA

6. I/O System Architecture (Continued)

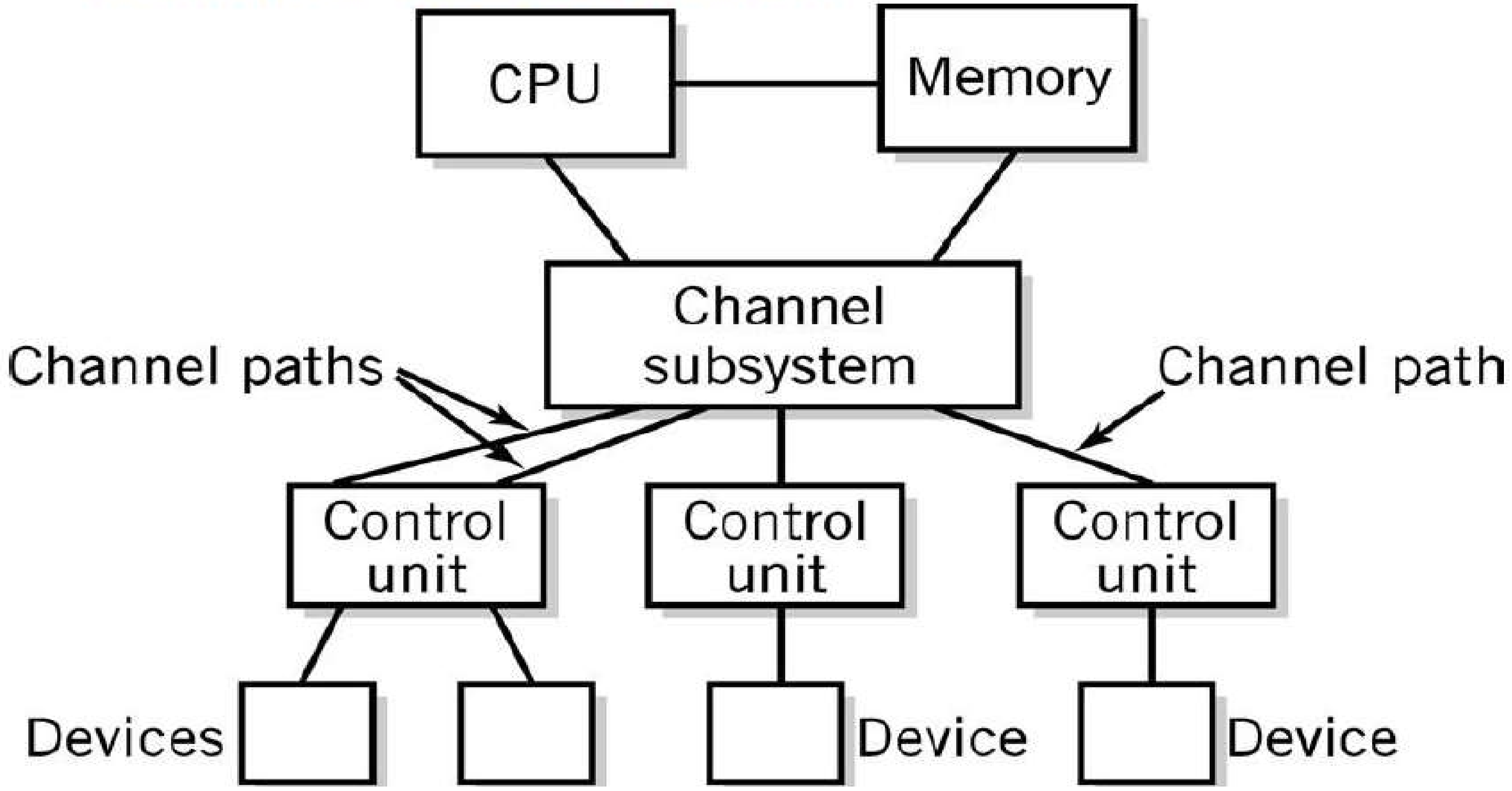
Channel Architecture

- **Sub channels**
  - Each of it is connected to a control unit module through one or more channel paths
  - Similar role to a device controller

- Up to 8 different channel paths between channel subsystem and control unit (used as alternative, if busy)

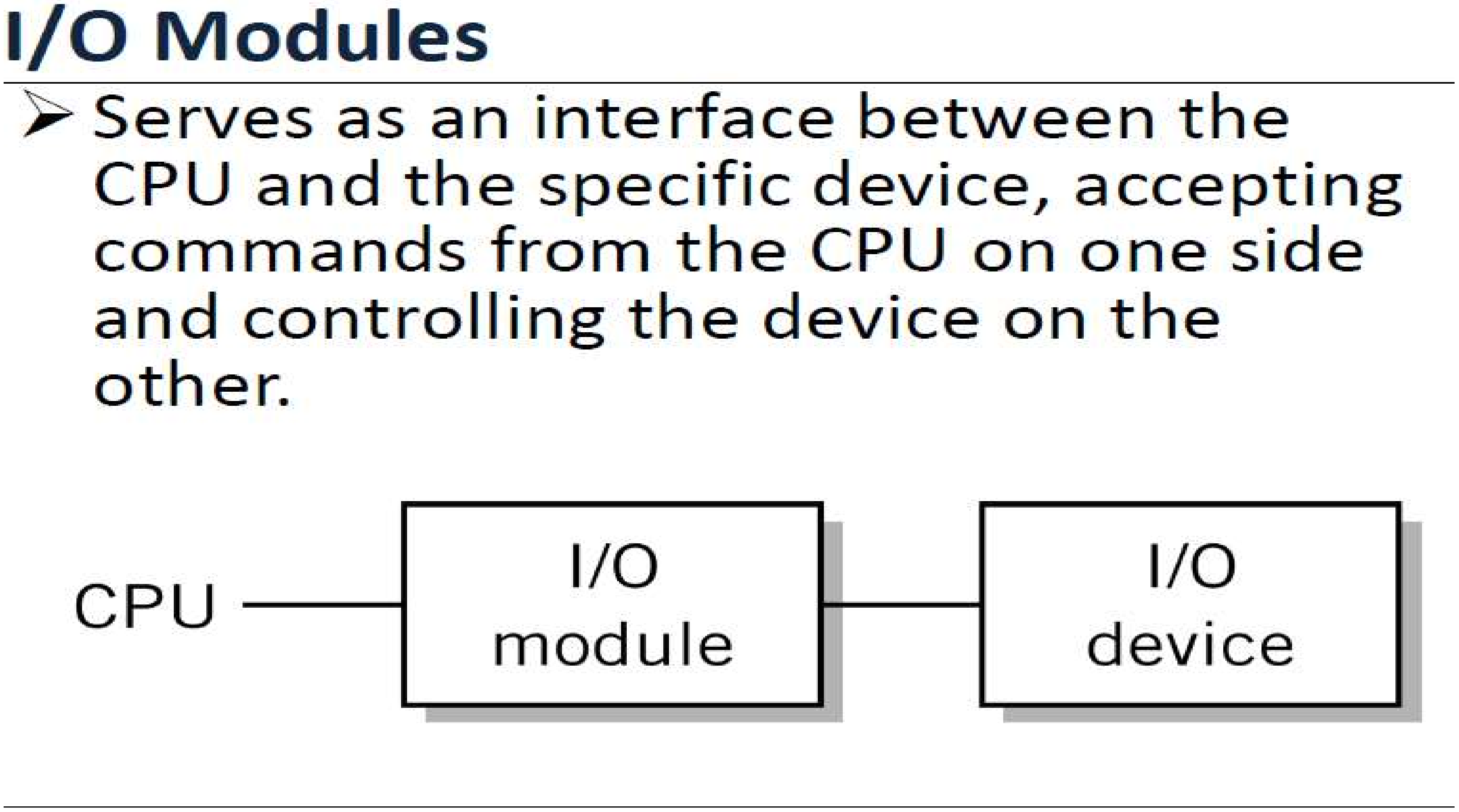
6. I/O System Architecture (Continued)

Channel Architecture



7. I/O Modules

7. I/O Modules (Continued)

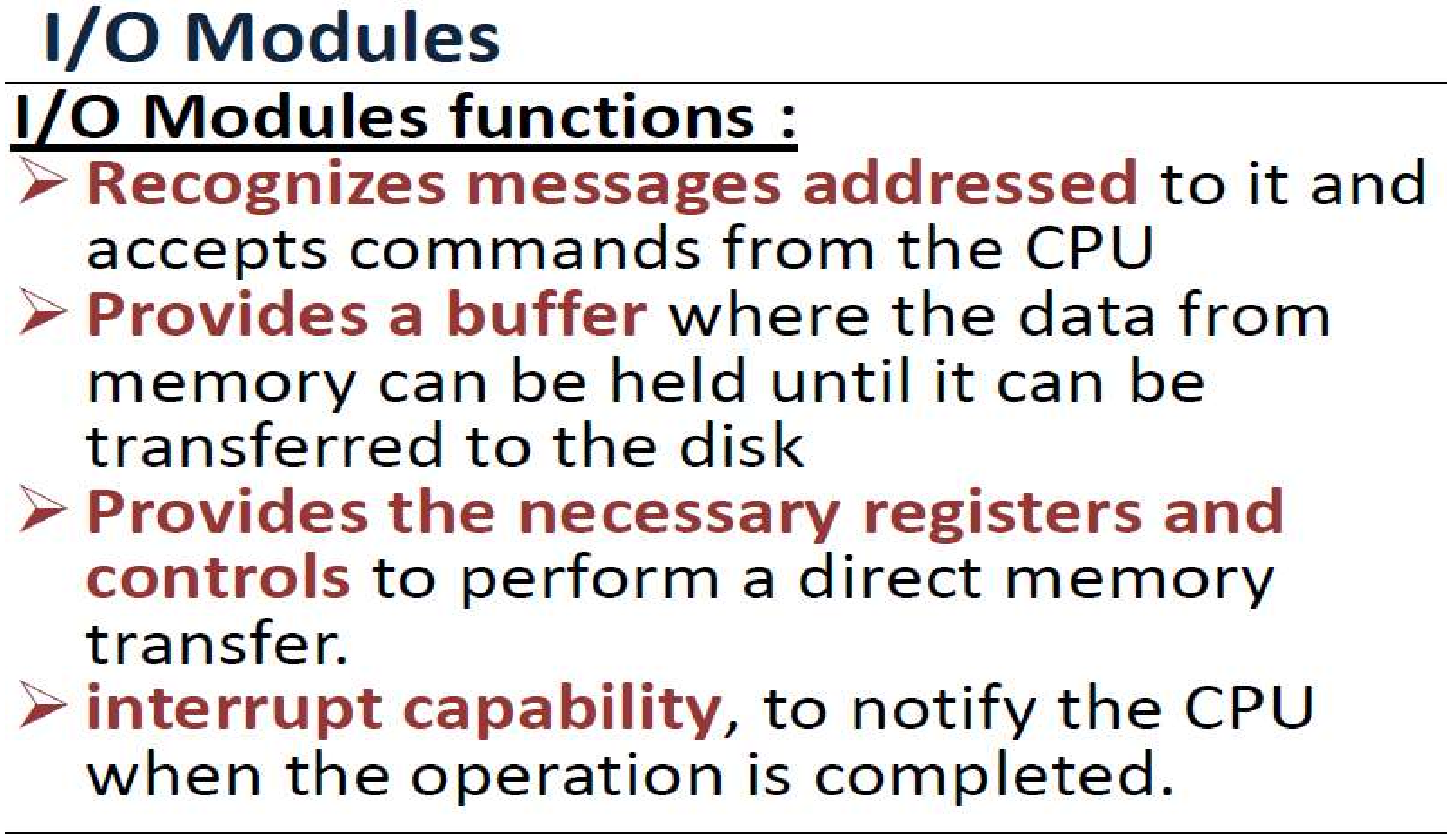


I/O Modules

- Serves as an interface between the CPU and the specific device, accepting commands from the CPU on one side and controlling the device on the other.

CPU
I/O module
I/O device

7. I/O Modules



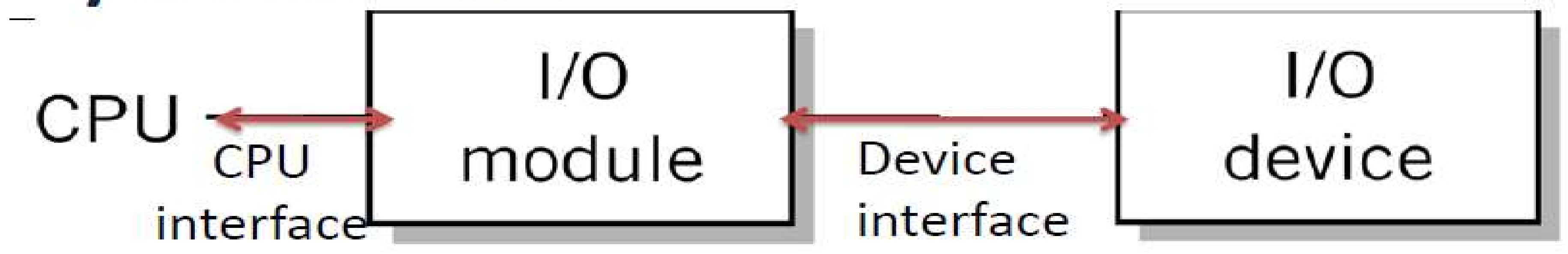
I/O Modules

I/O Modules functions :

- **Recognizes messages addressed** to it and accepts commands from the CPU
- **Provides a buffer** where the data from memory can be held until it can be transferred to the disk
- **Provides the necessary registers and controls** to perform a direct memory transfer.
- **interrupt capability**, to notify the CPU when the operation is completed.

7. I/O Modules (Continued)

I/O Modules



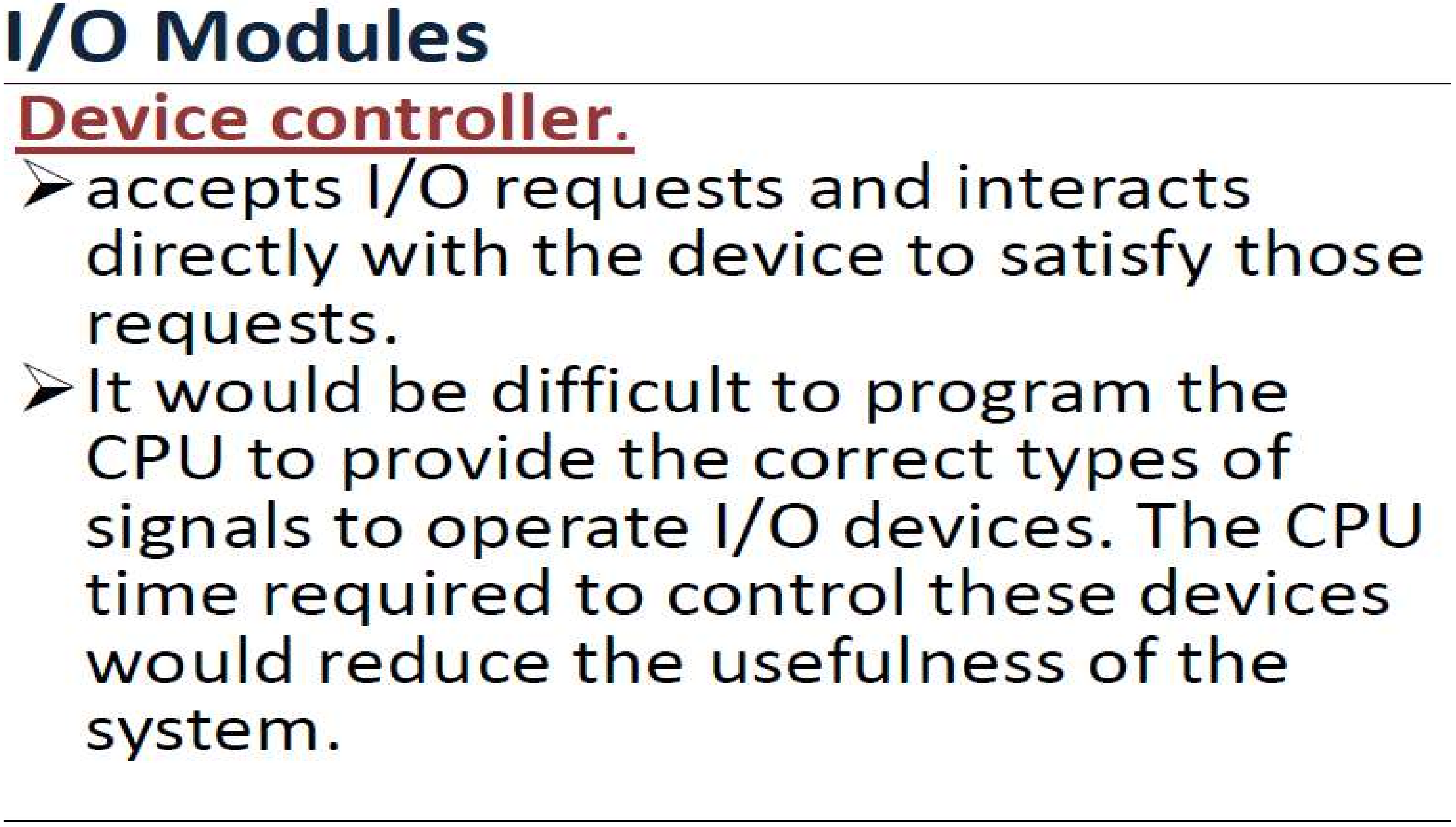
***CPU interface* : performs CPU interfacing tasks:**

- Accepting I/O commands from the CPU
- Sending interrupts and status information to the CPU.

***Device interface* : supplies control of the device:**

- Moving the head to the correct track in a disk drive and rewinding tape.

7. I/O Modules (Continued)

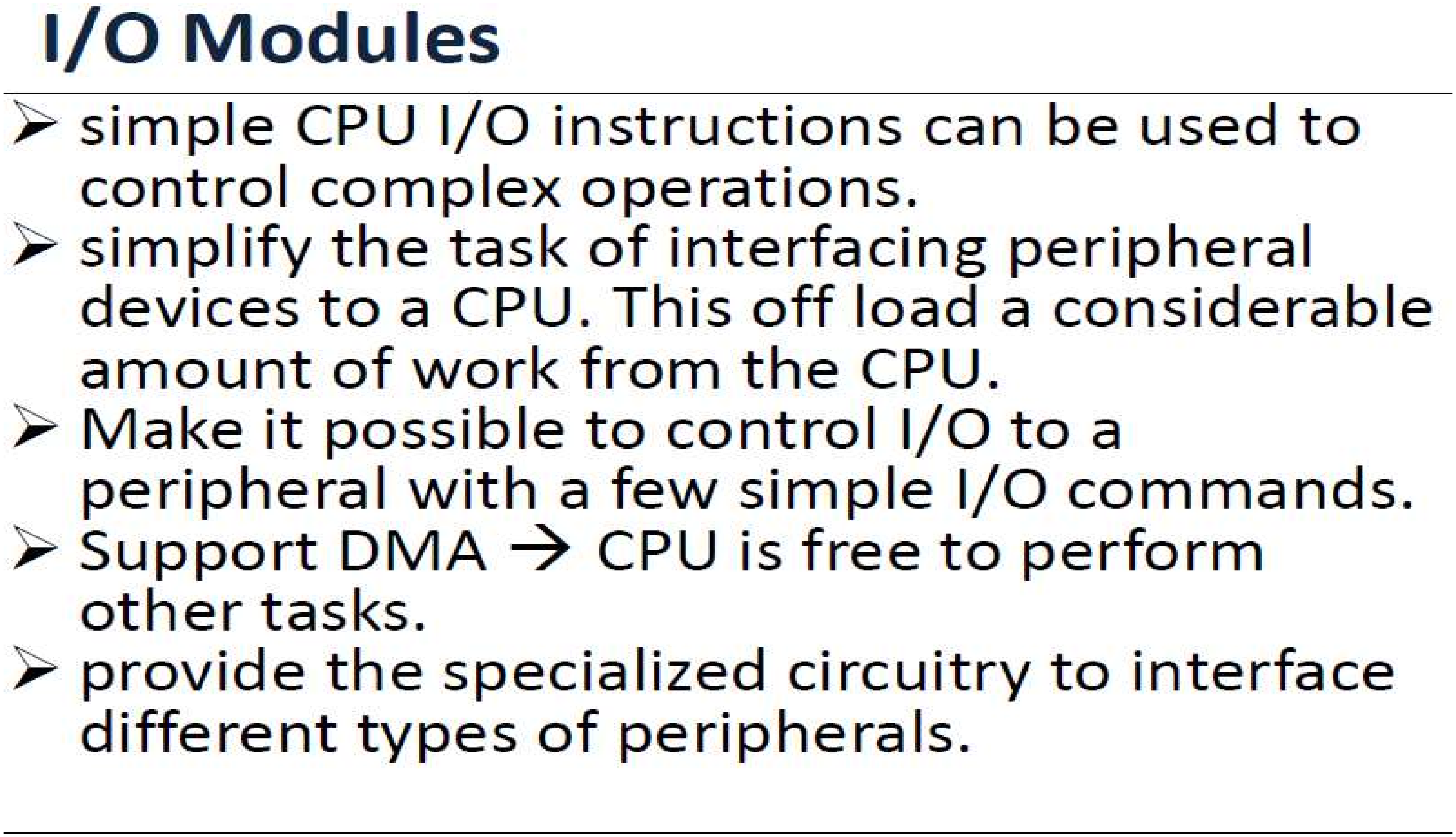


I/O Modules

Device controller.

- accepts I/O requests and interacts directly with the device to satisfy those requests.
- It would be difficult to program the CPU to provide the correct types of signals to operate I/O devices. The CPU time required to control these devices would reduce the usefulness of the system.

7. I/O Modules (Continued)



I/O Modules

- simple CPU I/O instructions can be used to control complex operations.
- simplify the task of interfacing peripheral devices to a CPU. This off load a considerable amount of work from the CPU.
- Make it possible to control I/O to a peripheral with a few simple I/O commands.
- Support DMA → CPU is free to perform other tasks.
- provide the specialized circuitry to interface different types of peripherals.

Chapter Review

1) Characteristics of typical I/O devices
2) I/O Techniques
3) Programmed I/O
4) Interrupts
   - Servicing Interrupts
   - The Uses of Interrupts
   - Multiple Interrupts and Prioritization
4) Direct Memory access (DMA)
5) I/O System Architecture
   - I/O Bus Architecture
   - Channel Architecture
6) I/O Modules