Operating System

Unit – 7

I/O Management

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

Computer System Organization

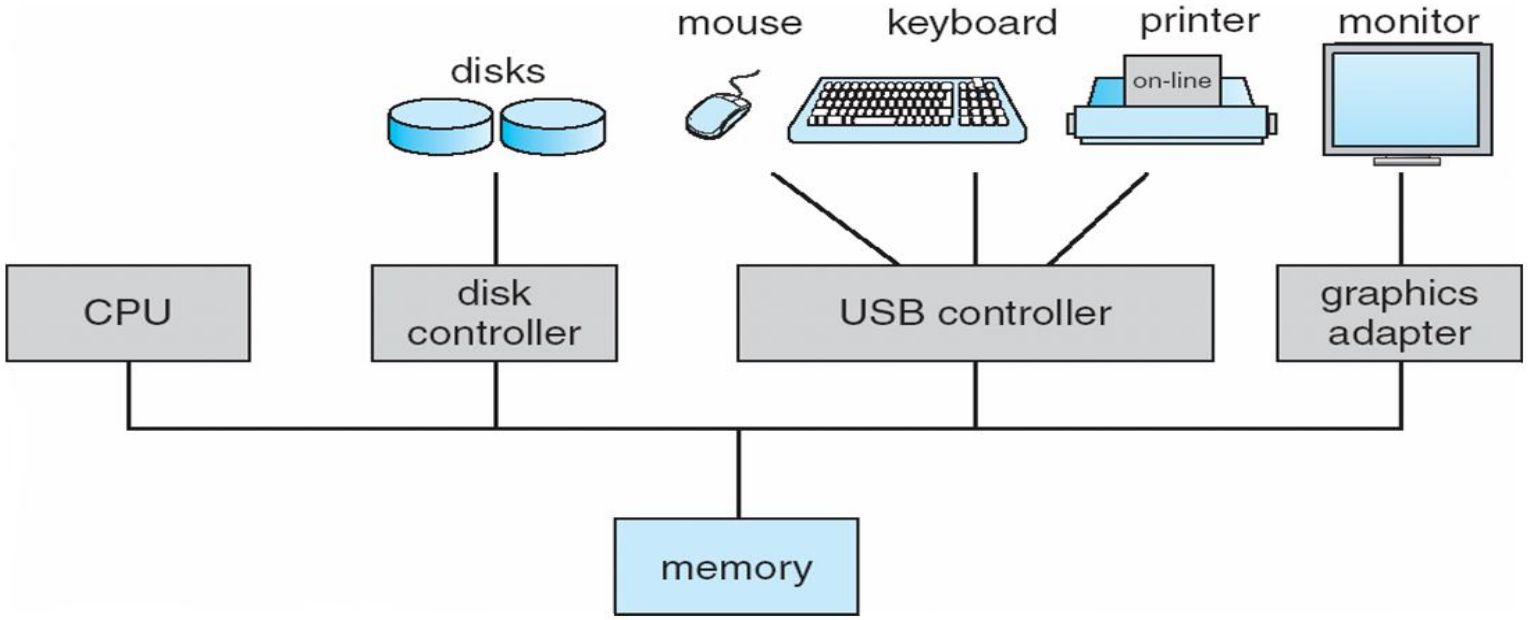


Fig. A typical PC computer system

- A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a **common bus** that provides access between components and shared memory (as shown in Figure). Every driver contains local buffer.

- Each device controller is in charge of a specific type of device. Depending on the controller, more than one device may be attached. For instance, one system USB port can connect to a USB hub, to which several devices can connect.

- Typically, operating systems have a device driver for each device controller. This device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device.

- The CPU and the device controllers can execute in parallel, competing for memory cycles. To ensure orderly access to the shared memory, a memory controller synchronizes access to the memory.

- A device controller maintains some local buffer storage and a set of special-purpose registers. The device controller is responsible for moving the data between the peripheral devices that it controls and its local buffer storage.

- To know how such system operates, three key aspects of the system which are, *interrupts* (which alert the CPU to events that require attention), *storage structure* and *I/O structure can be useful*.

Bootstrap Program:

- A bootstrap program is the first code to run when a computer starts, and it's essential for the operating system to work.

- The bootstrap program, also known as the **bootstrap loader**, loads the operating system into the computer's memory and initializes system components (for example, every device will have driver file, so all those driver file will be loaded).

- The bootstrap program is typically stored in the computer's Read Only Memory (ROM). The bootstrap program runs automatically when the computer starts or restarts. Without the bootstrap program, the computer wouldn't know how to load the operating system or start applications.

Interrupts :

- An interrupt is a signal emitted by a device attached to a computer or from a program within the computer. It requires the operating system (OS) to stop and figure out what to do next.

- Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the **system bus**. (There may be many buses within a computer system, but the system bus is the main communications path between the major components.)

- Interrupts are used for many other purposes as well and are a key part of how operating systems and hardware interact.

- An interrupt signal might be **planned** (i.e., specifically requested by a program) or it may be **unplanned** (i.e., caused by an event that may not be related to a program that's currently running on the system).

- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to a fixed location. The fixed location usually contains the starting address where the service routine for the interrupt is located.

- The **Interrupt Service Routine (ISR)** executes; on completion, the CPU resumes the interrupted computation.

- Interrupts are an important part of a computer architecture. Each computer design has its own interrupt mechanism, but several functions are common.

- The interrupt must transfer control to the appropriate interrupt service routine.

- The straightforward method for managing this transfer would be to invoke a generic routine to examine the interrupt information.

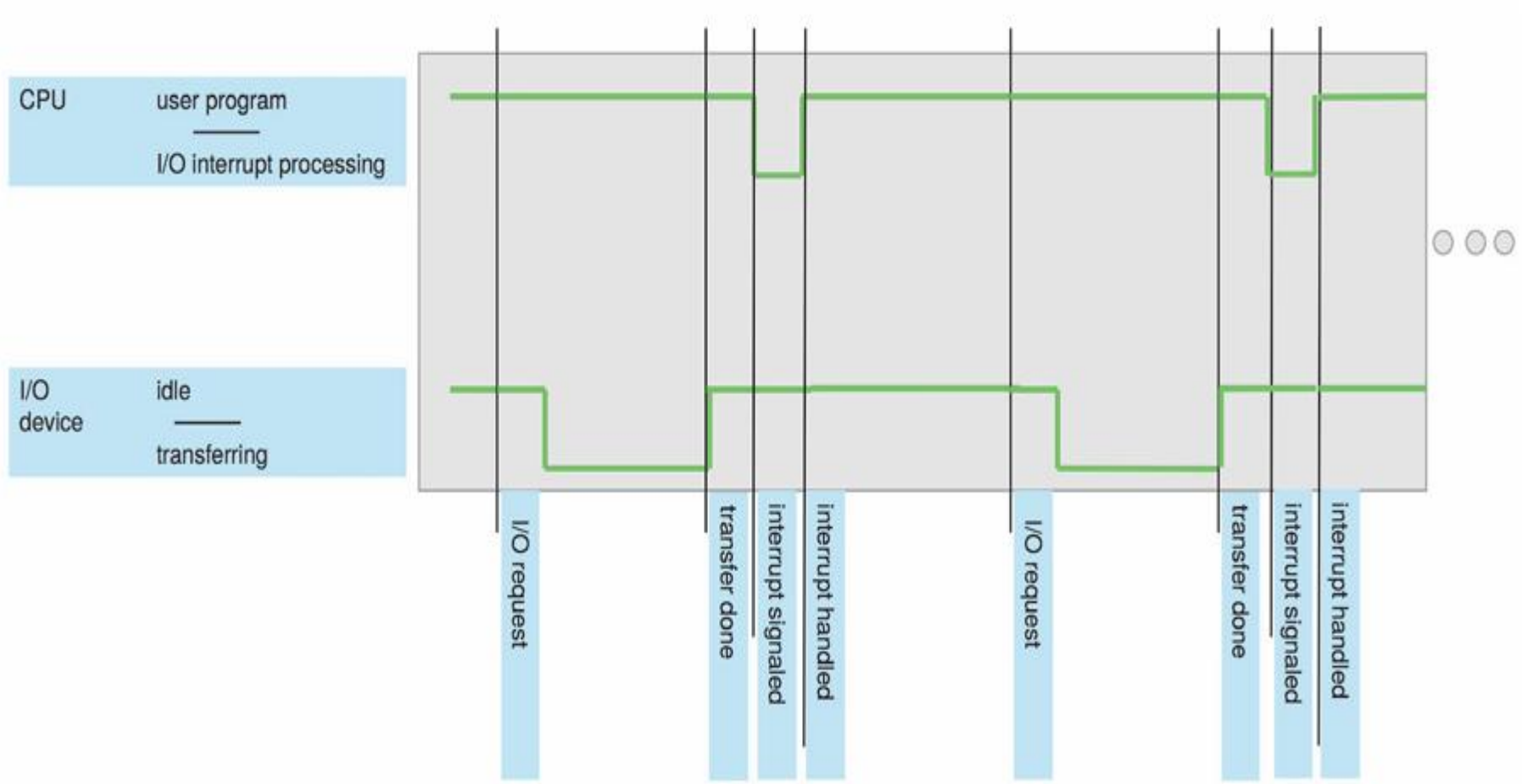


Fig. Interrupt timeline for a single program doing output.

- The routine, in turn, would call the **interrupt-specific handler**. However, interrupts must be handled quickly, as they occur very frequently.

- A table of pointers to interrupt routines can be used instead to provide the necessary speed. The interrupt routine is called indirectly through the table, with no intermediate routine needed.

- Generally, the table of pointers is stored in low memory (the first hundred or so locations). These locations hold the addresses of the interrupt service routines for the various devices.

- This array, or **interrupt vector**, of addresses is then indexed by a unique number, given with the interrupt request, to provide the address of the interrupt service routine for the interrupting device.

- The interrupt architecture must also save the state information of whatever was interrupted, so that it can restore this information after servicing the interrupt.

- *The basic interrupt mechanism works as follows.* The CPU hardware has a wire called the **interrupt-request line** that the CPU senses after executing every instruction.

- When the CPU detects that a controller has asserted a signal on the interrupt-request line, it reads the interrupt number and jumps to the interrupt-handler routine by using that interrupt number as an index into the interrupt vector. It then starts execution at the address associated with that index.

- The interrupt handler saves any state it will be changing during its operation, determines the cause of the interrupt, performs the necessary processing, performs a state restore, and executes a return_from_interrupt instruction to return the CPU to the execution state prior to the interrupt.

- The device controller ***raises*** an interrupt by asserting a signal on the interrupt request line, the CPU ***catches*** the interrupt and ***dispatches*** it to the interrupt handler, and the handler ***clears*** the interrupt by servicing the device.

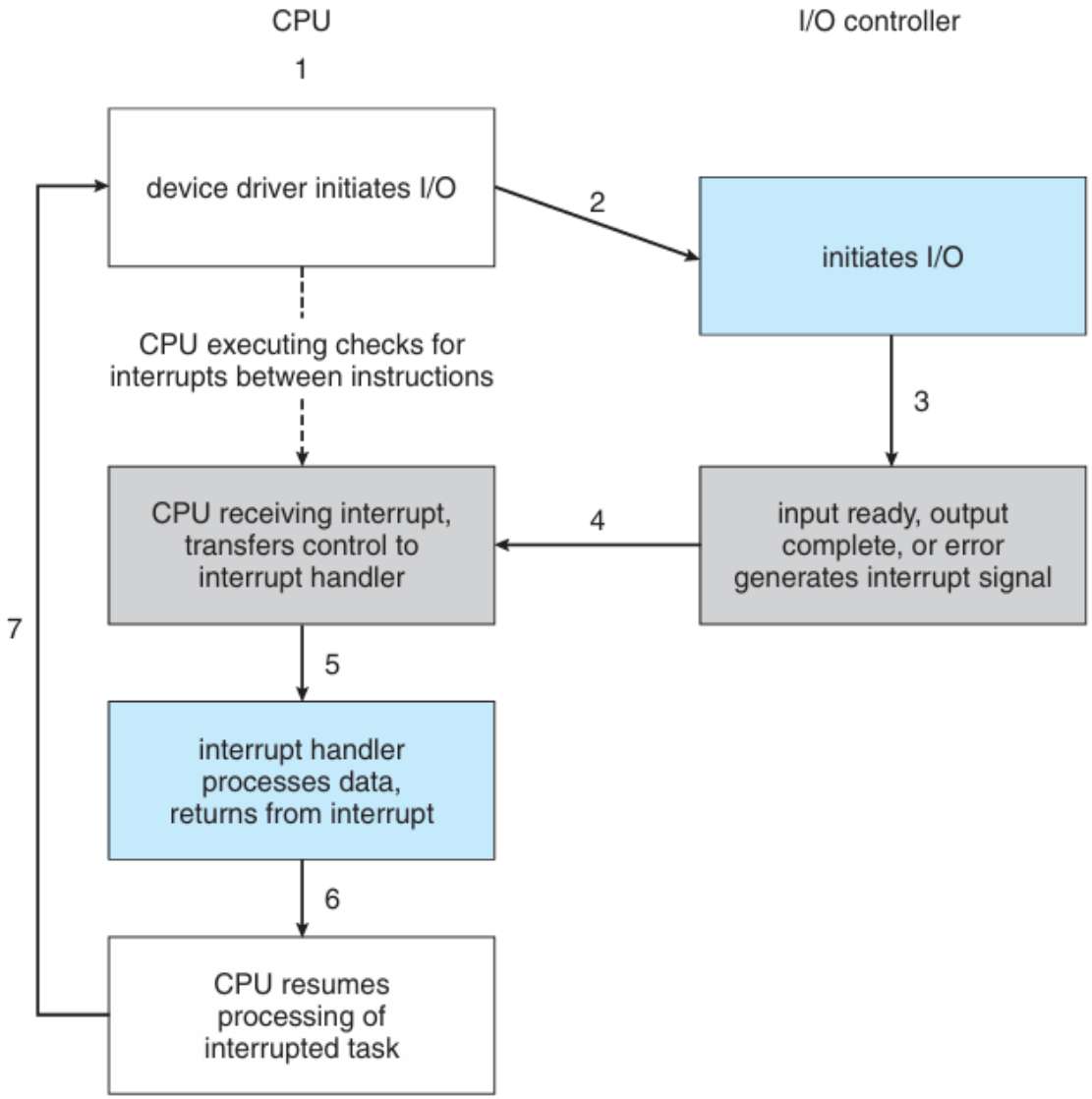


Fig. Interrupt-driven I/O cycle.

Types of Interrupts :

➢ There are two main types of interrupts:

1. **Hardware Interrupts:** The interrupt signal generated from external devices and i/o devices are made interrupt to CPU when the instructions are ready. Hardware interrupts are classified into two types which are as follows −
   - Maskable Interrupt − The hardware interrupts that can be delayed when a highest priority interrupt has occurred to the processor.
   - Non Maskable Interrupt − The hardware interrupts that cannot be delayed and immediately be serviced by the processor.

2. **Software Interrupts:** The interrupt signal generated from internal devices and software programs need to access any system call then software interrupts are present. Software interrupt is divided into two types. They are as follows −
   - Normal Interrupts − The interrupts that are caused by the software instructions are called software instructions.
   - Exception − Exception is nothing but an unplanned interruption while executing a program. For example − while executing a program if we got a value that is divided by zero is called an exception.

I/O Devices

- ***Block Devices***
  - **Block devices** are I/O devices that store data in fixed-size blocks or chunks.
  - A block device is one with which the driver communicates by sending entire blocks of data.
  - These devices allow **random access** to the data, meaning the operating system can read or write data at any location, directly accessing a block without needing to process the data in a sequential manner.

Key Characteristics of Block Devices:

- **Random Access**: Block devices support random access to data. Each block of data has a specific address, and the OS can directly access any block without waiting for the previous one to finish. This is ideal for storage devices like hard drives or SSDs, where files and data can be located at different positions on the device.

- **Fixed Block Size**: Data is stored in fixed-size blocks. The size of each block is usually consistent (for example, 512 bytes or 4 KB).

- **File Systems**: Block devices typically require a file system (e.g., NTFS, ext4) to organize and manage data. The file system manages how files are stored, accessed, and retrieved in the blocks on the device.

- **Buffering and Caching**: Block devices often use buffering or caching to improve the performance of I/O operations. Data is read or written in large chunks (blocks), reducing the overhead of frequent read/write operations.

Examples of Block Devices:

- **Hard Disk Drives (HDDs)**: Used for long-term data storage. The data is stored in blocks on the disk.
- **Solid-State Drives (SSDs)**: Similar to HDDs, but they use flash memory, which is faster.
- **USB Flash Drives**: Portable storage devices that store data in blocks.
- **Optical Discs (CDs/DVDs)**: Storage devices that also store data in blocks, typically for media or backup purposes.

I/O Devices (Contd.)

➢ *Character Devices*

- **Character devices** are I/O devices that treat data as a **stream of characters** (or bytes). These devices allow **sequential access**, meaning the data is processed one byte at a time.
- Unlike block devices, character devices do not have a fixed block size and generally do not support random access.
- A character device is one with which the driver communicates by sending and receiving single characters (e.g. Bytes, octets, etc.)

Key Characteristics of Character Devices:

- **Sequential Access**: Character devices provide data in a continuous stream. Data is transferred one byte at a time in sequence. The OS processes data sequentially, which means that you cannot directly access a specific byte or character without processing the previous data first.

- **No Fixed Block Size**: Data is typically transmitted in a continuous stream, without the need for fixed-sized blocks.

- **Minimal Buffering**: Character devices may not use large buffers like block devices. Data is often processed in real-time as it is received or sent, which is crucial for devices that require continuous interaction (such as a keyboard or microphone).

- **Real-Time Operation**: Character devices are often used in scenarios where real-time data processing is necessary (e.g., input from a keyboard or output to a printer).

Examples of Block Devices:

- **Keyboards**: The data is sent as a sequence of keypresses, one at a time.
- **Mice**: Input data is sent as a continuous stream of movements and button clicks.
- **Serial Ports**: Devices connected through serial interfaces (e.g., modems, RS-232 interfaces) transmit data one byte at a time.
- **Printers**: Some printers process data as a continuous stream of characters, printing one character at a time.
- **Terminals/Displays**: Devices that display data as a stream of characters, such as text on a console screen.

Device Controller

- Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

- The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the device controller.

- There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

- Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller.

Synchronous vs asynchronous I/O

- **Synchronous I/O** − In this scheme CPU execution waits while I/O proceeds

- **Asynchronous I/O** − I/O proceeds concurrently with CPU execution

Communication to I/O Devices

The CPU must have a way to pass information to and from an I/O device. There are three approaches available to communicate with the CPU and Devices.

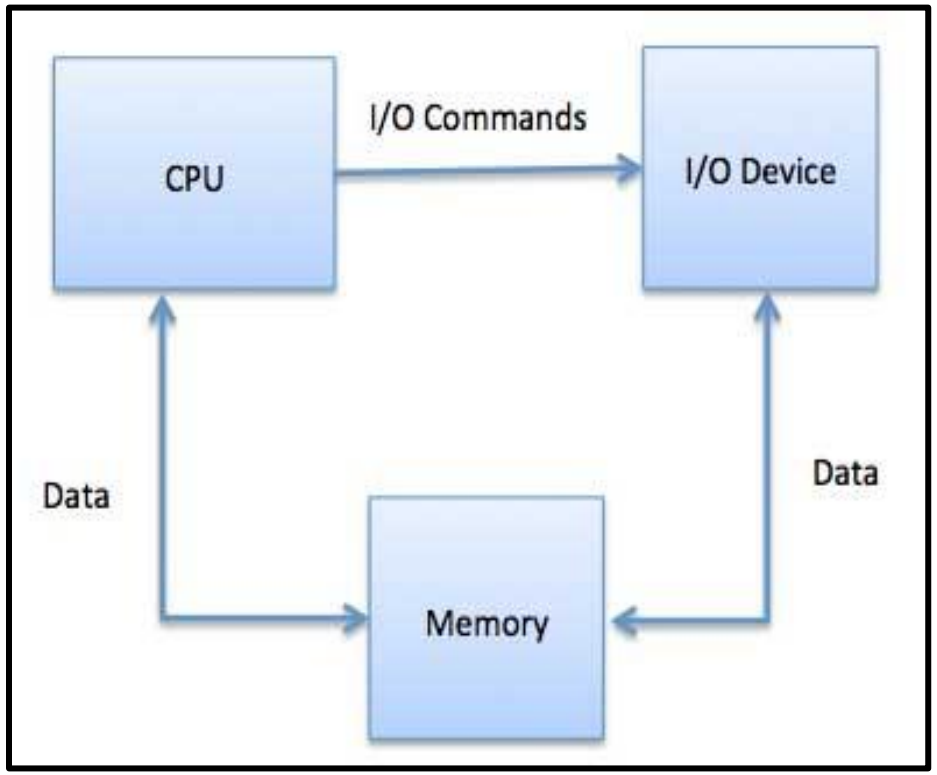
- Special Instruction I/O
- Memory-mapped I/O
- Direct memory access (DMA)

➢ *Special Instruction I/O*

- ❑ This uses CPU instructions that are specifically made for controlling I/O devices.
- ❑ These instructions typically allow data to be sent to an I/O device or read from an I/O device.

➢ *Memory-mapped I/O*

- When using memory-mapped I/O, the same address space is shared by memory and I/O devices.
- The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.
- While using memory mapped IO, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU. I/O device operates asynchronously with CPU, interrupts CPU when finished.
- The advantage to this method is that every instruction which can access memory can be used to manipulate an I/O device. Memory mapped IO is used for most high-speed I/O devices like disks, communication interfaces.

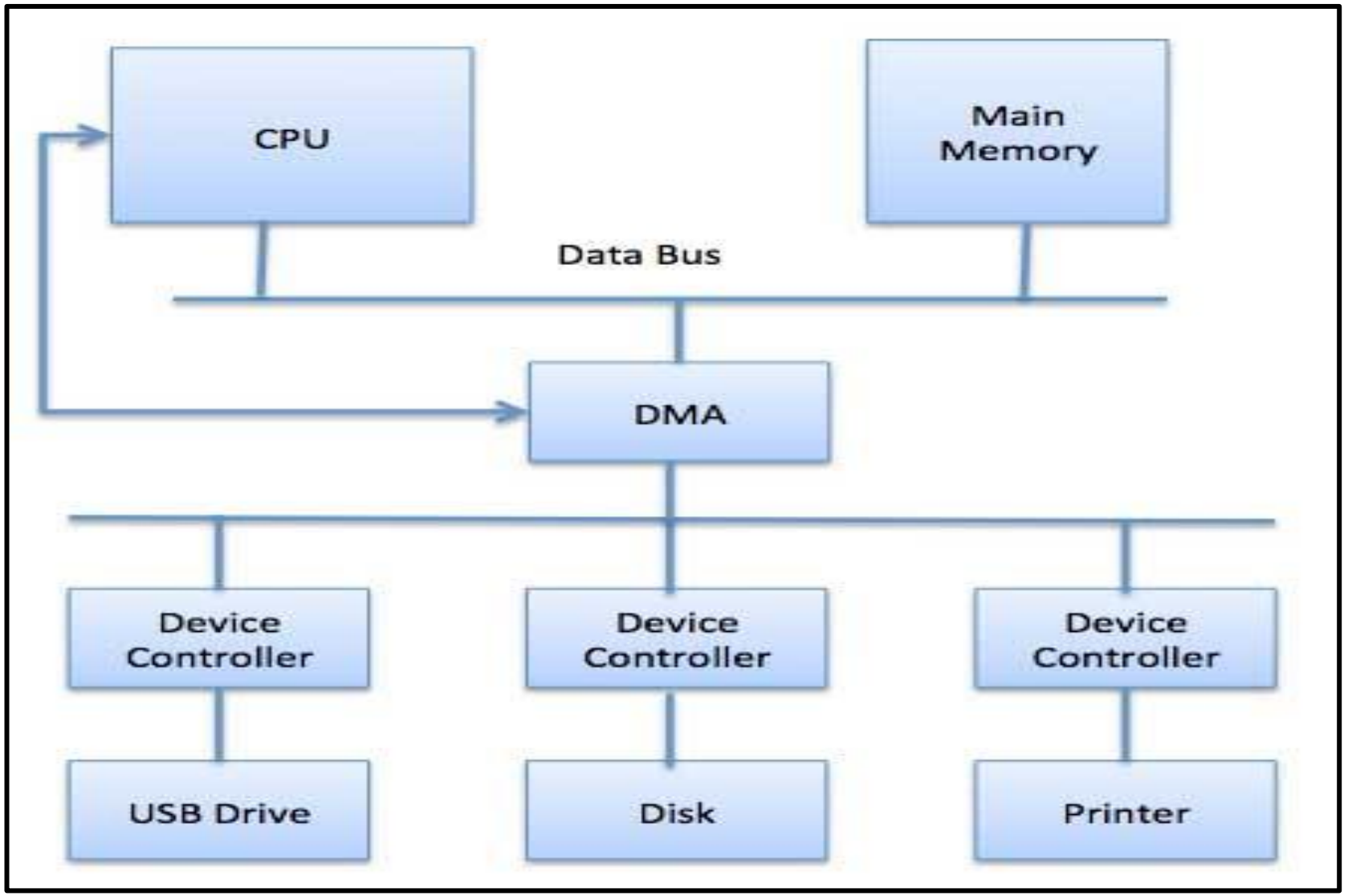


➢ *Direct Memory Access (DMA)*

- Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So, a typical computer uses **Direct Memory Access (DMA)** hardware to reduce this overhead.

- Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.

- Direct Memory Access needs a special hardware called DMA controller (DMAC) that manages the data transfers and arbitrates access to the system bus. The controllers are programmed with source and destination pointers (where to read/write the data), counters to track the number of transferred bytes, and settings, which includes I/O and memory types, interrupts and states for the CPU cycles.


CPU
Main Memory
Data Bus
DMA
Device Controller
Device Controller
Device Controller
USB Drive
Disk
Printer

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition (Pearson Publications), 2014.

3. https://www.geeksforgeeks.org/

4. https://www.javatpoint.com/

5. https://www.tutorialspoint.com/

6. https://www.nesoacademy.org/

7. https://www.tpointtech.com/