UNIT I INTRODUCTION 7

Computer System – Elements and organization; Operating System Overview – Objectives and Functions – Evolution of Operating System; Operating System Structures–Operating System Services–User Operating System Interface – System Calls – System Programs – Design and Implementation–Structuring method.

I. Computer System

INTRODUCTION

Operating System is a program that acts as an interface between a user of a computer and the computer hardware. Eg. Windows (Microsoft), iOS (Apple), Android (Google), Linux/Ubuntu (open source)

Goals of Operating System:

- Execute user programs and make solving the user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner

1. COMPUTER SYSTEM

A computer system can be divided roughly into four components: the **hardware,** the operating system, **the** application programs, **and a** user

Hardware: The Central Processing Unit (CPU), the Memory, and the Input/output (I/O) devices provides the basic computing resources for the system.

Application Programs: Such as word processors, spreadsheets, compilers, and web browsers define the ways in which these resources are used to solve users' computing problems.

Operating System: Controls the hardware and coordinates its use among the various application programs for the various users.

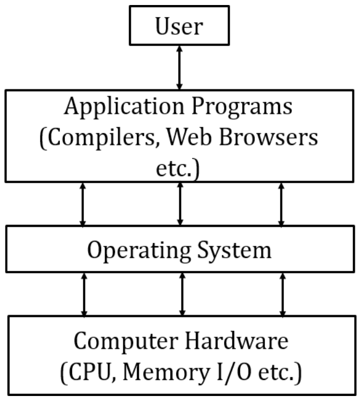

User
Application Programs (Compilers, Web Browsers etc.)
Operating System
Computer Hardware (CPU, Memory I/O etc.)

Figure: Computer System

The operating system provides the means for proper use of these resources in the operation of the computer system.

Operating Systems from two viewpoints:

- User View
- System View

- **User View**

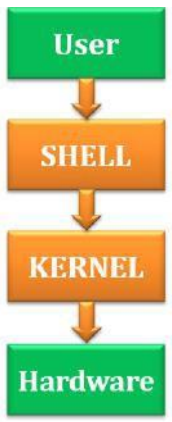
The user's view of the computer varies according to the interface being used. Many computer users sit with a laptop or in front of a PC consisting of a monitor, keyboard, and mouse. Such a system is designed for one user to control its resources. For this, the operating system is designed mostly for **ease of use**, performance and security and not to **resource utilization.**

Many users interact with mobile devices such as smartphones and tablets. These devices are connected to networks through cellular or other wireless technologies. The user interface for mobile computers generally features a touch screen, where the user interacts with the system by pressing and swiping fingers across the screen rather than using a physical keyboard and mouse. Many mobile devices also allow users to interact through a voice recognition interface, such as Apple's Siri.

- **System View**

From the computer's point of view, the operating system is the program most intimately involved with the hardware. It is a **resource allocator**. A computer system has many resources that may be required to solve a problem: CPU time, memory space, storage space, I/O devices, and so on. The operating system acts as the manager of these resources. The operating system must decide how to allocate the resources to specific programs and users so that it can operate the computer system efficiently and fairly.

Components of Operating System



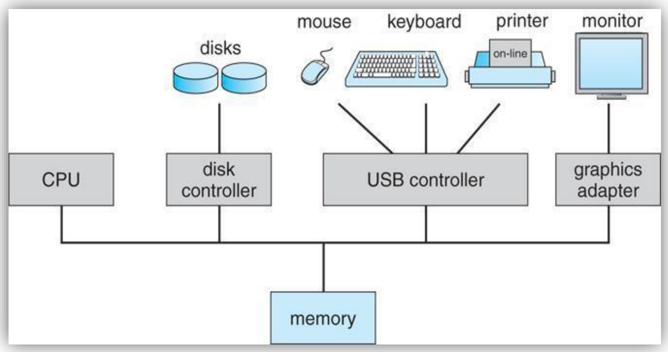
Shell :

- ❖ Environment that gives a user an interface to access the operating system's services.
- ❖ Launch Applications
- ❖ User Mode

Kernel :

- ❖ Keep track of the hardware and computer's operations.
- ❖ Kernel Mode

2. COMPUTER SYSTEM – ELEMENTS AND ORGANIZATION



A computer system consists of CPU and a number of device controllers connected through a common **bus** that provides access between components and shared memory. Each device controller is in charge of a specific type of device, more than one device may be attached. For example, one system USB port can connect to a USB hub, to which several devices can be connected. 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.

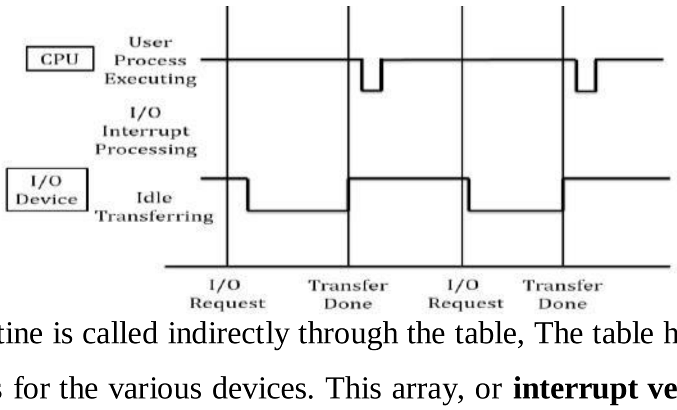
Operating systems have a **device driver** for each device controller. This device driver understands the device controller and provides the rest of the operating system with a uniform interface to the device. The CPU and the device controllers can execute in parallel.

Three key aspects of the system are

- ❖ Interrupts
- ❖ Storage Structure
- ❖ I/O Structure

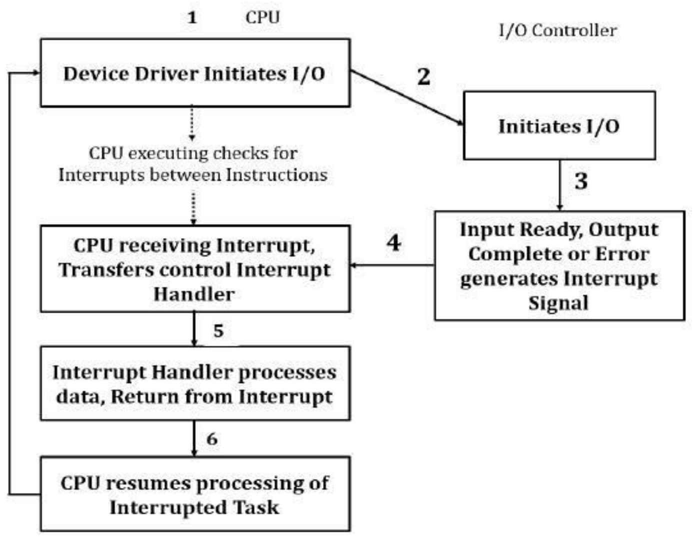
❖ Interrupts

Hardware may trigger an interrupt by sending a signal to the CPU. Software may trigger an interrupt by executing a special operation called a **system call.** 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 executes; on completion, the CPU resumes the interrupted computation. A time line of this operation is shown below



The interrupt routine is called indirectly through the table, The table holds the addresses of the interrupt service routines for the various devices. This array, or **interrupt vector,** of addresses is then indexed by a unique device 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 address of the interrupted instruction. After the interrupt is serviced, the saved return address is loaded into the program counter, and the interrupted computation resumes as though the interrupt had not occurred. 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.


1 CPU
I/O Controller
Device Driver Initiates I/O
2
Initiates I/O
CPU executing checks for Interrupts between Instructions
3
CPU receiving Interrupt, Transfers control Interrupt Handler
4
Input Ready, Output Complete or Error generates Interrupt Signal
5
Interrupt Handler processes data, Return from Interrupt
6
CPU resumes processing of Interrupted Task

Most CPUs have two interrupt request lines. One is the **nonmaskable interrupt**, which is reserved for events such as unrecoverable memory errors. The second interrupt line is **maskable**: it can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted. The maskable interrupt is used by device controllers to request service.

❖ **Storage Structure**

- KB : 1024 Bytes
- MB : 1024 ^{2} Bytes
- GB : 1024 ^{3} Bytes (1 Million Bytes)
- TB : 1024 ^{4} Bytes (1 Billion Bytes)
- PB : 1024 ^{5} Bytes

The CPU load the instructions from memory. General-purpose computers run most of their programs from rewritable memory, called main memory (**RAM**).

The first program to run on computer to power ON is a **bootstrap program**, which then loads the operating system. Since RAM is **volatile,** loses its content when power is turned off or otherwise lost. So the bootstrap program cannot be stored in RAM. So the computer uses electrically erasable programmable read-only memory (EEPROM) and other forms of **firmware,** storage that is infrequently written to and is nonvolatile. EEPROM can be changed but cannot be changed frequently. In addition, it is low speed, and so it contains mostly static programs and data that aren't frequently used. For example, the iPhone uses EEPROM to store serial numbers and hardware information about the device.

All forms of memory provide an array of bytes. Each byte has its own address. Interaction is achieved through a sequence of load or store instructions to specific memory addresses. The load instruction moves a byte or word from main memory to an internal register within the CPU, whereas the store instruction moves the content of a register to main memory.

The CPU automatically loads instructions from main memory for execution from the location stored in the program counter. First fetches an instruction from memory and stores that instruction in the **instruction register**. The instruction is then decoded and may cause operands to be fetched from memory and stored in some internal register. After the instruction on the operands has been executed, the result may be stored back in memory.

The programs and data must be in main memory permanently. This arrangement is not possible on most systems for two reasons:

- ❖ Main memory is usually too small to store all needed programs and data permanently.
- ❖ Main memory, is volatile, it loses its contents when power is turned off or otherwise lost.

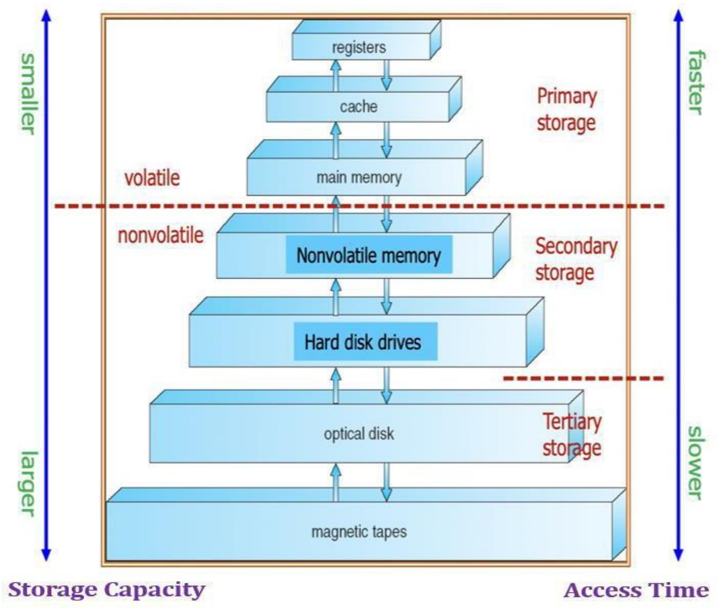
Most computer systems provide **secondary storage** as an extension of main memory. The main requirement for secondary storage is that it be able to hold large quantities of data permanently. The most common secondary-storage devices are **hard-disk drives** (**HDDs**) and **nonvolatile memory** (**NVM**) **devices**, which provide storage for both programs and data. Most programs are stored in secondary storage until they are loaded into memory.

Secondary storage is also much slower than main memory. Other possible components include cache memory, CD-ROM, magnetic tapes, and so on. Those that are slow enough and large enough that they are used only for special purposes, to store backup copies of material stored on other devices, are called **tertiary storage**.

The wide variety of storage systems can be organized in a hierarchy according to storage capacity and access time.

Smaller and faster memory closer to the CPU. Various storage systems are either volatile or nonvolatile. Volatile storage, loses its contents when the power to the device is removed, so data must be written to nonvolatile storage for safekeeping.

The top four levels of memory are constructed using **semiconductor memory**, which consists of semiconductor based electronic circuits. Non Volatile Memory devices, at the fourth level, are faster than hard disks. The most common form of NVM device is flash memory, which is popular in mobile devices such as smartphones and tablets. Increasingly, flash memory is being used for long term storage on laptops, desktops, and servers as well.



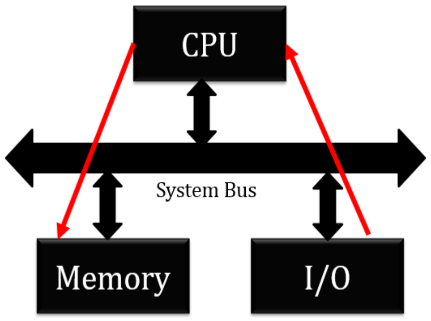
Volatile storage will be referred to simply as **memory**. Nonvolatile storage retains its contents when power is lost. This type of storage can be classified into two distinct types:

- **Mechanical** : Storage systems are HDDs, optical disks, holographic storage, and magnetic tape.
- **Electrical : S**torage systems are flash memory, FRAM, NRAM, and SSD.

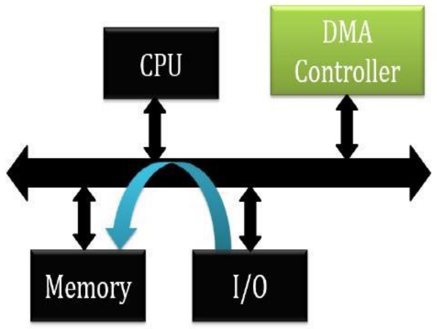
Mechanical storage is generally larger and less expensive per byte than electrical storage. Conversely, electrical storage is typically costly, smaller, and faster than mechanical storage.

❖ I/O Structure

A large portion of operating system code is dedicated to manage I/O. General purpose computer system consists of multiple devices, all of which exchange data via a common bus. The form of interrupt driven I/O is fine for moving small amounts of data, **direct memory access** (**DMA**) is used for bulk data transfer.



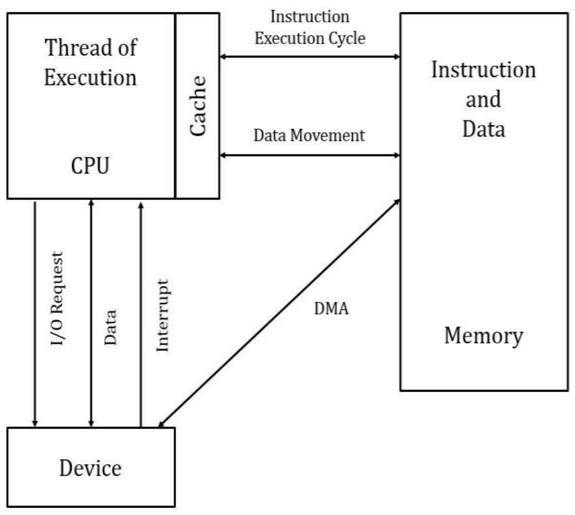
Normal Data Transfer



DMA Data Transfer

After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from the device and main memory, with no intervention by the CPU. Only one interrupt is generated per block, to tell the device driver that the operation has completed, rather than the one interrupt per byte generated for low speed devices. While the device controller is performing these operations, the CPU is available to accomplish other work.

Some high-end systems use switch rather than bus architecture. On these systems, multiple components can talk to other components concurrently, rather than competing for cycles on a shared bus.


Thread of Execution
CPU
Cache
Instruction Execution Cycle
Data Movement
Instruction and Data
Memory
I/O Request
Data
Interrupt
DMA
Device