NPTEL ONLINE CERTIFICATION COURSES

Operating System Fundamentals
Santanu Chattopadhyay
Electronics and Electrical Communication Engg.

Introduction

CONCEPTS COVERED

Concepts Covered:

What is an Operating System?
Computer-System Organization
Operating-System Structure
Operating-System Operations
Process Management
Memory Management
Storage Management
Protection and Security
Kernel Data Structures
Computing Environments

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware

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

Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System
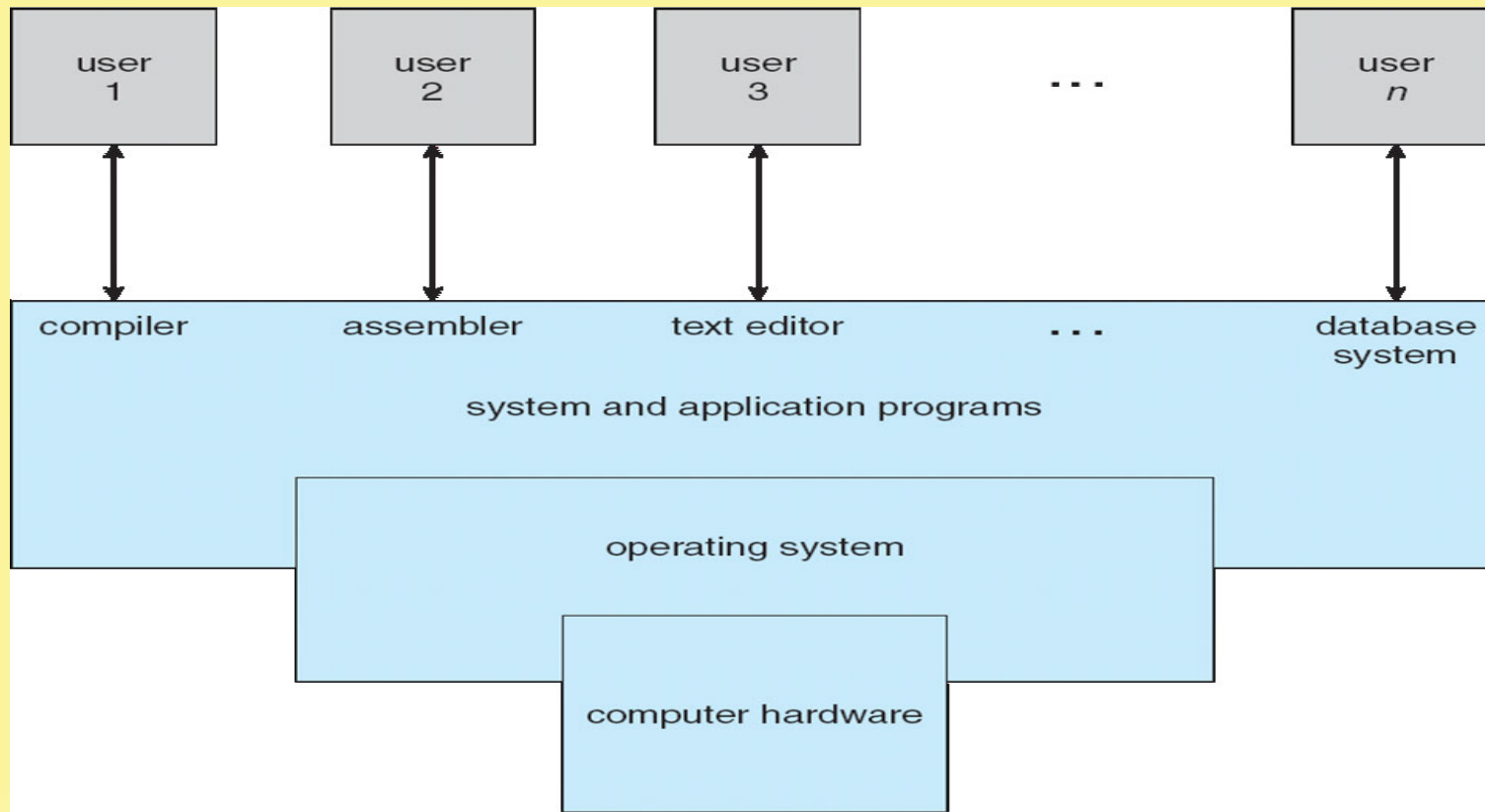
What Operating Systems Do

- The operating system controls the hardware and coordinates its use among the various application programs for the various users.

- We can also view a computer system as consisting of hardware, software, and data.

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

- An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an **environment** within which other programs can do useful work.

- To understand more fully the operating system's role, we need to explore operating systems from two viewpoints:
  - The user
  - The system.

User View

The user's view of the computer varies according to the interface being used

- **Single user computers** (e.g., PC, workstations). Such systems are designed for one user to monopolize its resources. The goal is to maximize the work (or play) that the user is performing. The operating system is designed mostly for **ease of use** and **good performance.**

- **Multi user computers** (e.g., mainframes, computing servers). These users share resources and may exchange information. The operating system in such cases is designed to maximize resource utilization -- to assure that all available CPU time, memory, and I/O are used efficiently and that no individual users takes more than their air share.

User View (Cont.)

- **Handheld computers** (e.g., smartphones and tablets). The user interface for mobile computers generally features a **touch screen.** The systems are resource poor, optimized for usability and battery life.

- **Embedded computers** (e.g., computers in home devices and automobiles) The user interface may have numeric keypads and may turn indicator lights on or off to show status. The operating systems are designed primarily to run without user intervention.

System View

From the computer's point of view, the operating system is the program most intimately involved with the hardware. There are two different views:

- The operating system is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- The operating system is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Defining Operating System

No universally accepted definition of what an OS:

- Operating systems exist to offer a reasonable way to solve the problem of creating a usable computing system.

- The fundamental goal of computer systems is to execute user programs and to make solving user problems easier.

- Since bare hardware alone is not particularly easy to use, application programs are developed.
    - These programs require certain common operations, such as those controlling I/O devices.
    - The common functions of controlling and allocating resources brought together into one piece of software: the **operating system**.

Defining Operating System (Cont.)

No universally accepted definition of what is part of the OS:

- A simple viewpoint is that it includes everything a vendor ships when you order the operating system. The features that are included vary greatly across systems:
  - Some systems take up less than a megabyte of space and lack even a full-screen editor,
  - Some systems require gigabytes of space and are based entirely on graphical windowing systems.

Defining Operating System (Cont.)

No universally accepted definition of what is part of the OS:

- A more common definition, and the one that we usually follow, is that the operating system is the one program running at all times on the computer -- usually called the **kernel**.

- Along with the kernel, there are two other types of programs:
  - System programs, which are associated with the operating system but are not necessarily part of the kernel.
  - Application programs, which include all programs not associated with the operation of the system.

Defining Operating System (Cont.)

- The emergence of mobile devices, have resulted in an increase in the number of features that constituting the operating system.

- Mobile operating systems often include not only a core kernel but also **middleware** -- a set of software frameworks that provide additional services to application developers.

- For example, each of the two most prominent mobile operating systems -- Apple's iOS and Google's Android -- feature a core kernel along with middleware that supports databases, multimedia, and graphics (to name only a few).

Evolution of Computer Systems
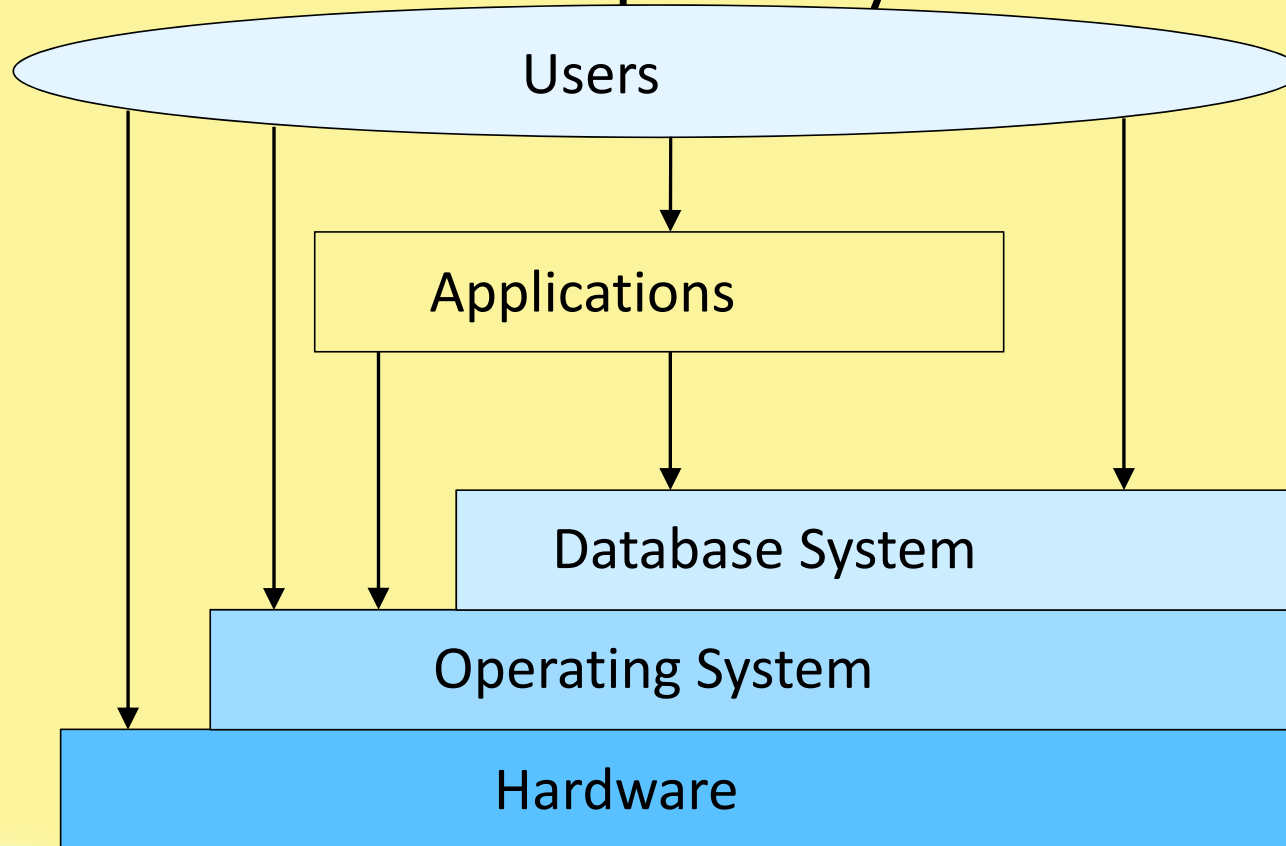
Computer-System Organization

- 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 to shared memory.

- Each device controller is in charge of a specific type of device (for example, disk drives, audio devices, or video displays). Each device controller has a local buffer.

- CPU moves data from/to main memory to/from local buffers.

- 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.

Modern Computer System
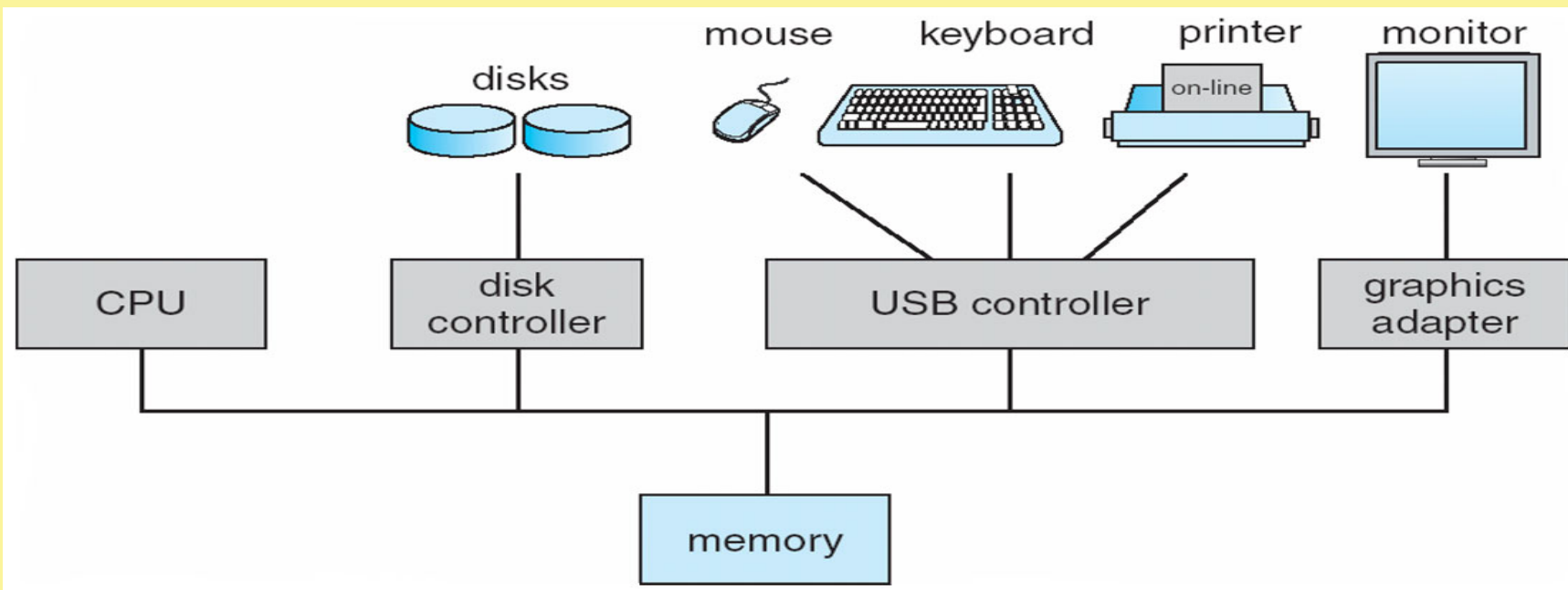
Computer Startup

- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer-System Operation

- Once the kernel is loaded and executing, it can start providing services to the system and its users.

- Some services are provided outside of the kernel, by system programs that are loaded into memory at boot time to become **system processes**, or **system daemons** that run the entire time the kernel is running.

- On UNIX, the first system process is **init** and it starts many other daemons. Once this phase is complete, the system is fully booted, and the system waits for some event to occur.

- The occurrence of an event is usually signaled by an **interrupt.**

Interrupts

- There are two types of interrupts**:**
  - **Hardware** -- a device may trigger an interrupt by sending a signal to the CPU, usually by way of the system bus.
  - **Software** -- a program may trigger an interrupt by executing a special operation called a **system call.**
- A software-generated interrupt (sometimes called **trap** or **exception**) is caused either by an error (e.g., divide by zero) or a user request (e.g., an I/O request).
- An operating system is **interrupt driven.**

Common Functions of Interrupts

- When an interrupt occurs, the operating system preserves the state of the CPU by storing the registers and the program counter

- Determines which type of interrupt has occurred and transfers control to the interrupt-service routine.

- An interrupt-service routine is a collection of routines (modules), each of which is responsible for handling one particular interrupt (e.g., from a printer, from a disk)

- The transfer is generally through the **interrupt vector**, which contains the addresses of all the service routines

- Interrupt architecture must save the address of the interrupted instruction.

Interrupt Timeline
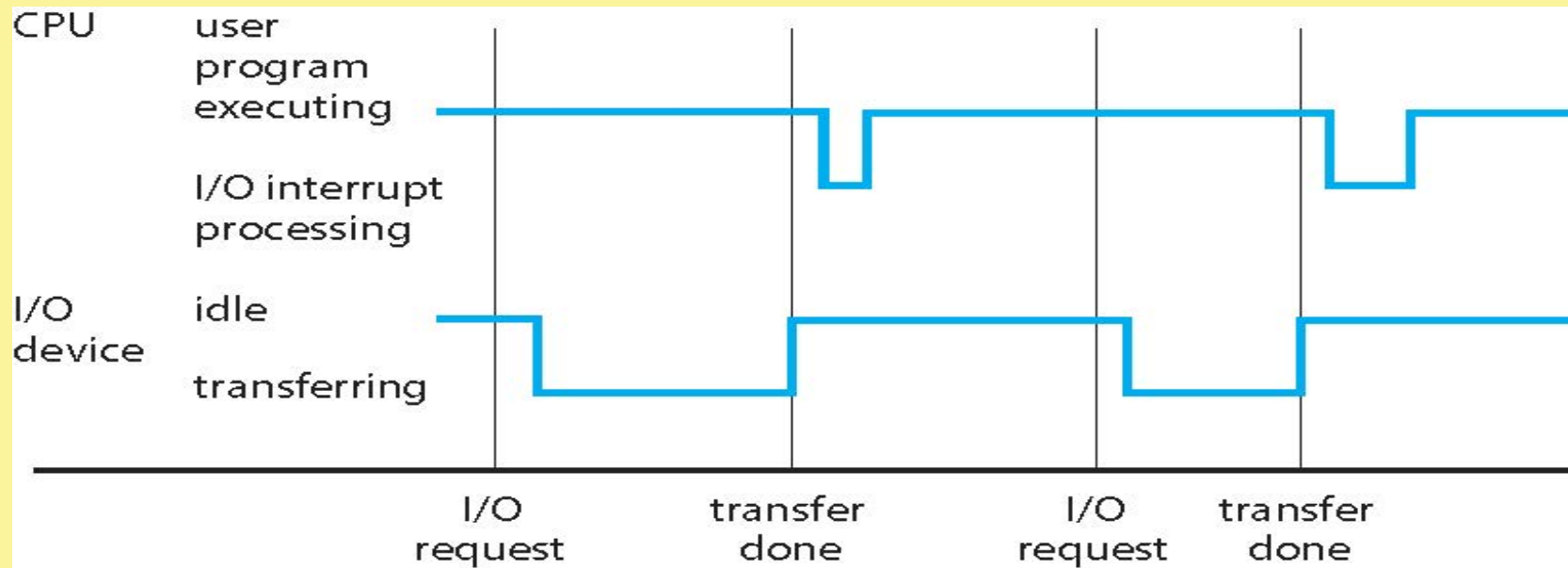
Interrupt-driven I/O cycle
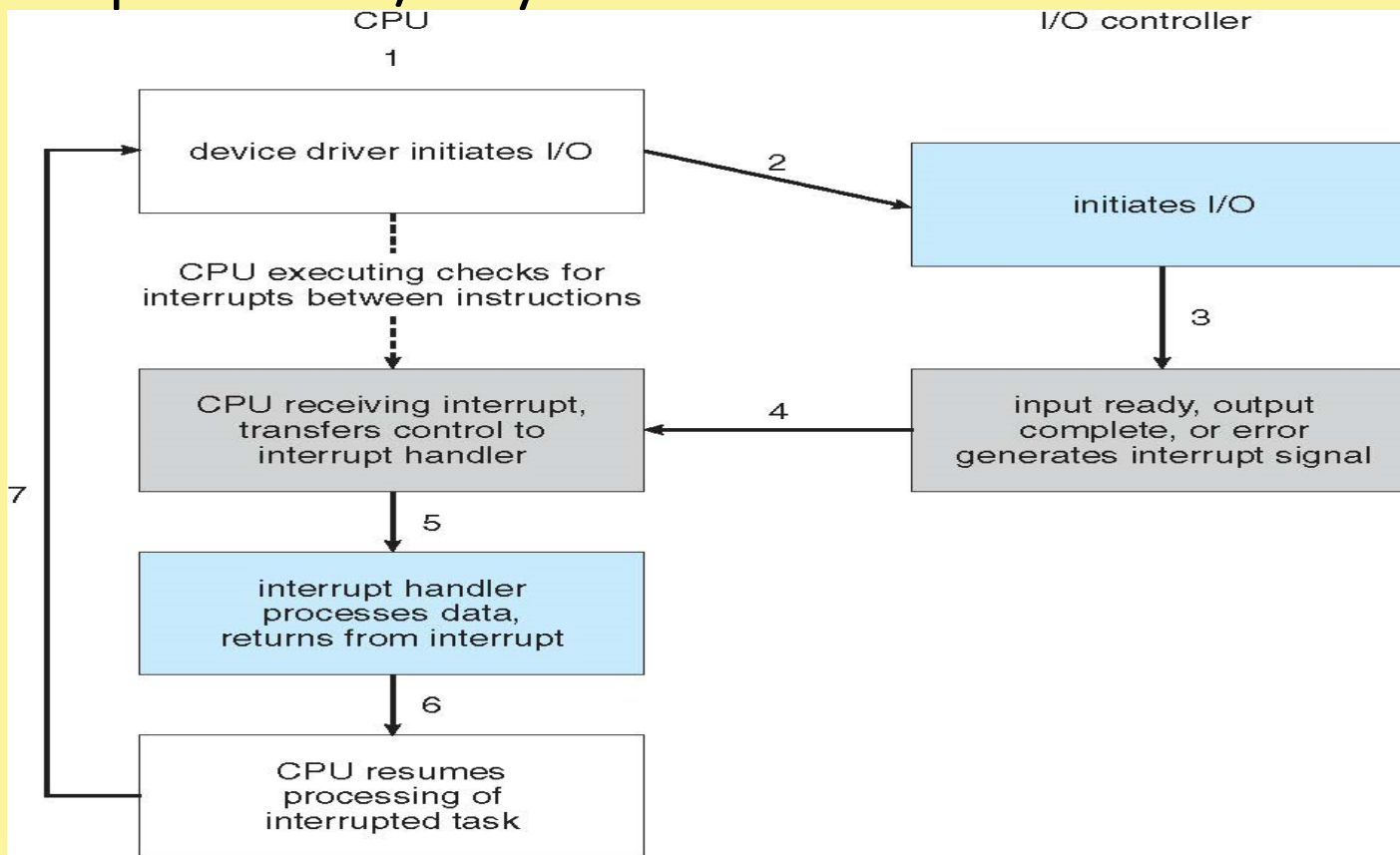
Intel Pentium processor event-vector table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Storage Structure

- Main memory – the only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Hard disks – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
    - The **disk controller** determines the logical interaction between the device and the computer
  - **Solid-state disks** – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular
- Tertiary storage

Storage Definition

- The basic unit of computer storage is the **bit**. A bit can contain one of two values, 0 and 1. All other storage in a computer is based on collections of bits.

- A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.

- A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes.

Storage Definition (Cont.)

- Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.

- Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes.

- Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).
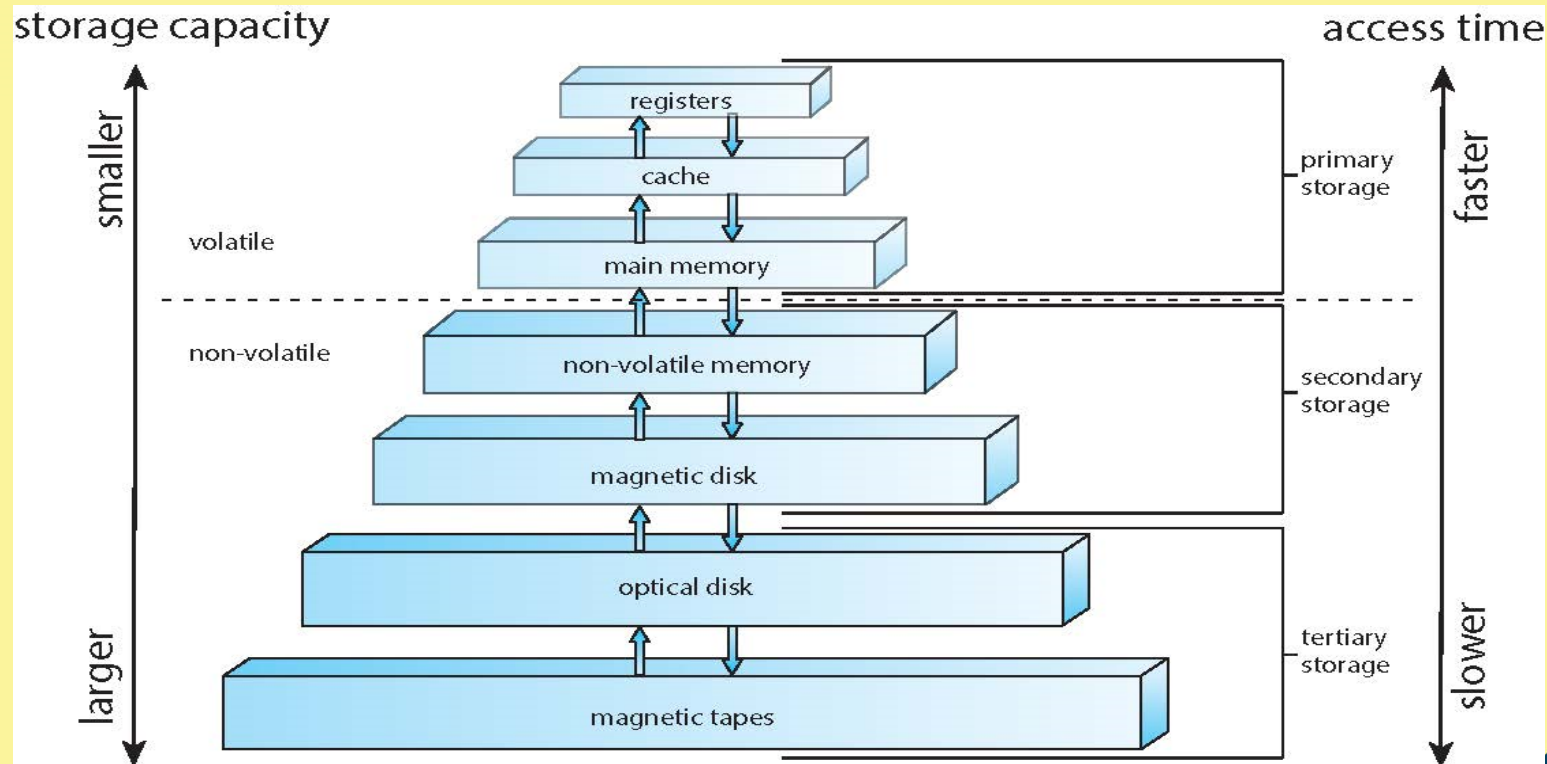
Storage Hierarchy

- Storage systems organized in hierarchy
    - Speed
    - Cost
    - Volatility

- **Caching** – copying information from "slow" storage into faster storage system;
    - Main memory can be viewed as a cache for secondary storage

- **Device Driver** for each device controller to manage I/O
    - Provides uniform interface between controller and kernel

Storage-device hierarchy

I/O Structure

- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.

- Each device controller is in charge of a specific type of device. More than one device may be attached. For instance, seven or more devices can be attached to the **small computer-systems interface (**SCSI) controller.

- 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.

- 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.

I/O Structure (Cont.)

- To start an I/O operation, the device driver loads the appropriate registers within the device controller.

- The device controller, in turn, examines the contents of these registers to determine what action to take (such as "read" a character from the keyboard).

- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.

- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.

- For other operations, the device driver returns status information.

Direct Memory Access Structure

- Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.

- To solve this problem, **direct memory access** (DMA) is used.
  - 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 its own buffer storage to memory, with no intervention by the CPU.
  - Only one interrupt is generated per block, to tell the device driver that the operation has completed. 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. In this case, DMA is even more effective.

How a Modern Computer Works

A von Neumann architecture and a depiction of the interplay of all components of a computer system.
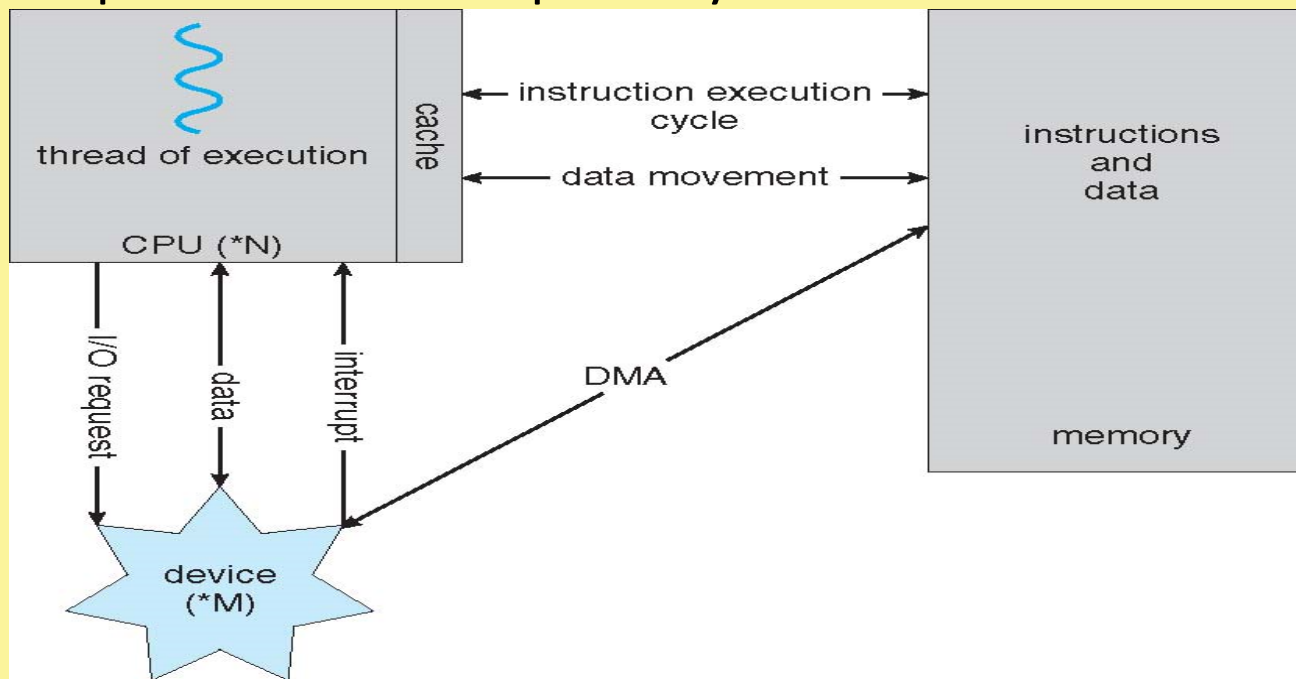
Computer-System Architecture

- **Single general-purpose processor**
    - Most systems have special-purpose processors as well

- **Multiprocessors** systems
    - Also known as **parallel systems**, **tightly-coupled systems**
    - Advantages include:
        - **Increased throughput**
        - **Economy of scale**
        - **Increased reliability** – graceful-degradation/fault-tolerance
    - Two types:
        - **Symmetric Multiprocessing** – each processor performs all tasks
        - **Asymmetric Multiprocessing** – each processor is assigned a specific task.

# Symmetric Multiprocessing Architecture
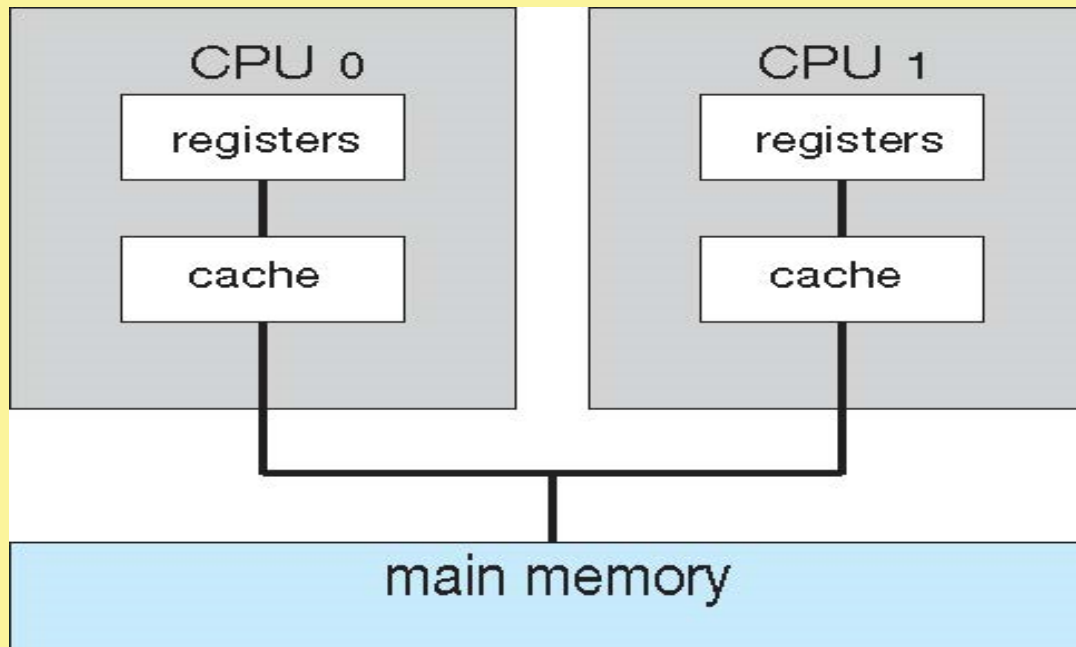
Multicore Systems

- Most CPU design now includes multiple computing cores on a single chip. Such multiprocessor systems are termed **multicore**.

- Multicore systems can be more efficient than multiple chips with single cores because:
  - On-chip communication is faster than between-chip communication.
  - One chip with multiple cores uses significantly less power than multiple single-core chips, an important issue for laptops as well as mobile devices.

- Note -- while multicore systems are multiprocessor systems, not all multiprocessor systems are multicore.

# A dual-core with two cores placed on the same chip
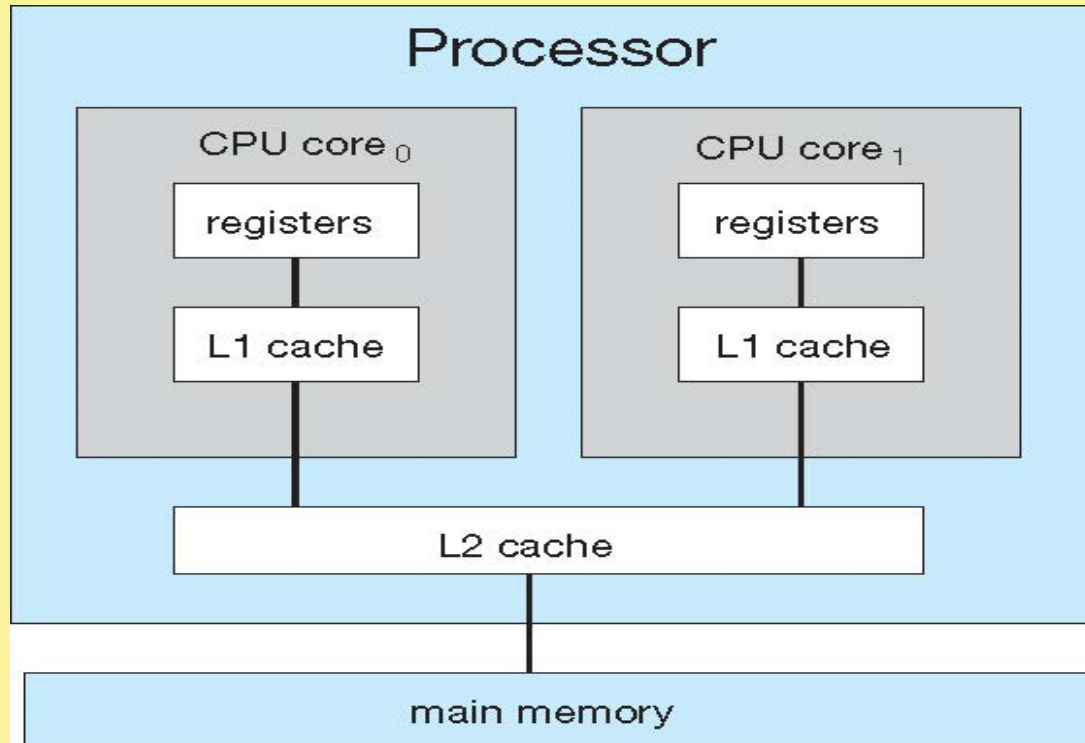
Clustered Systems

Like multiprocessor systems, but multiple systems working together

- Usually sharing storage via a storage-area network (SAN)

- Provides a high-availability service which survives failures
  - Asymmetric clustering has one machine in hot-standby mode
  - Symmetric clustering has multiple nodes running applications, monitoring each other

- Some clusters are for high-performance computing (HPC)
  - Applications must be written to use parallelization

- Some have distributed lock manager (DLM) to avoid conflicting operations

Clustered Systems
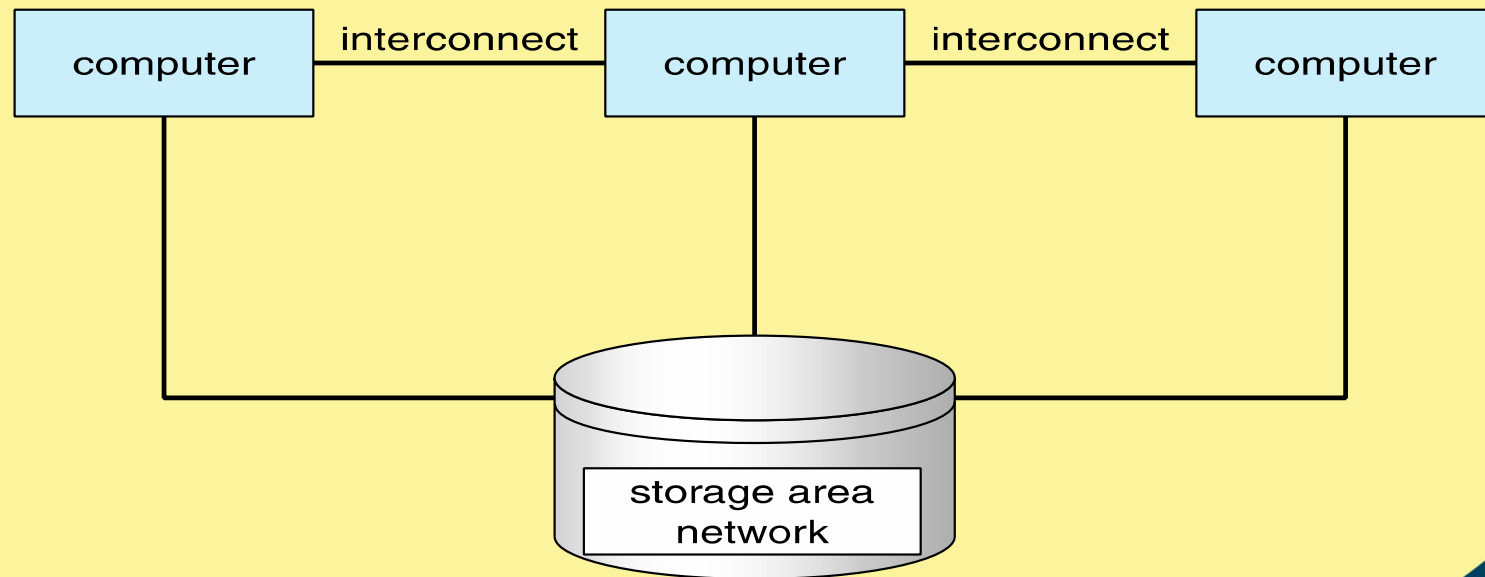
Multiprogrammed System

- Single user cannot keep CPU and I/O devices busy at all times
- **Multiprogramming** organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- Batch systems:
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- Interactive systems:
  - Logical extension of batch systems -- CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing

Interactive Systems

- **Response time** should be < 1 second

- Each user has at least one program executing in memory. Such a program is referred to as a **process**

- If several processes are ready to run at the same time, we need to have **CPU scheduling.**

- If processes do not fit in memory, **swapping** moves them in and out to run

- **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System
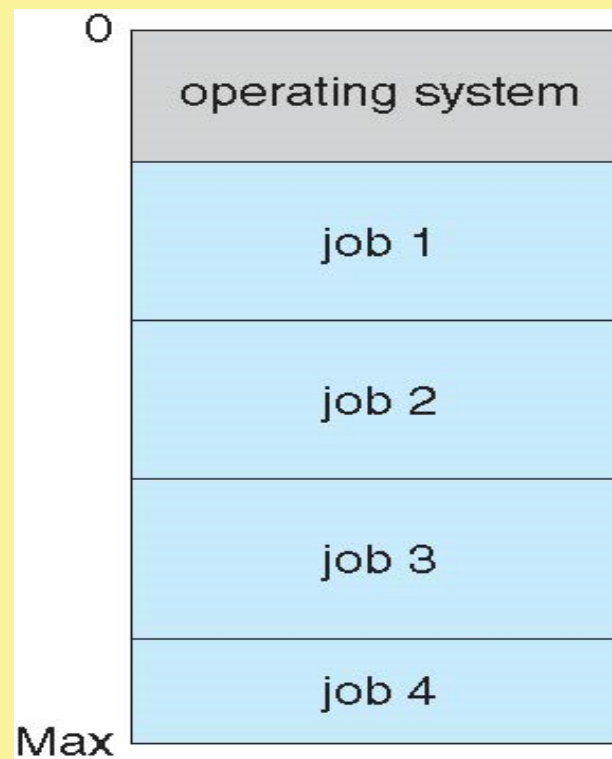
Modes of Operation

- A mechanism that allows the OS to protect itself and other system components

- Two modes:
  - **User mode**
  - **Kernel mode**

- Mode bit (0 or 1) provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code
  - Some instructions designated as **privileged**, only executable in kernel mode
  - Systems call by a user asking the OS to perform some function changes from user mode to kernel mode.
  - Return from a system call resets the mode to user mode.