

# Chapter 1: Introduction





# Chapter 1: Introduction

---

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection
- Virtualization
- Distributed Systems
- Kernel Data Structures
- Computing Environments
- Free/Libre and Open-Source Operating Systems





# Objectives

---

- Describe the general organization of a computer system and the role of interrupts
- Describe the components in a modern, multiprocessor computer system
- Illustrate the transition from user mode to kernel mode
- Discuss how operating systems are used in various computing environments
- Provide examples of free and open-source operating systems





# What Does the Term Operating System Mean?

---

- An operating system is “fill in the blanks”
- What about:
  - Car
  - Airplane
  - Printer
  - Washing Machine
  - Toaster
  - Compiler
  - Etc.





# 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

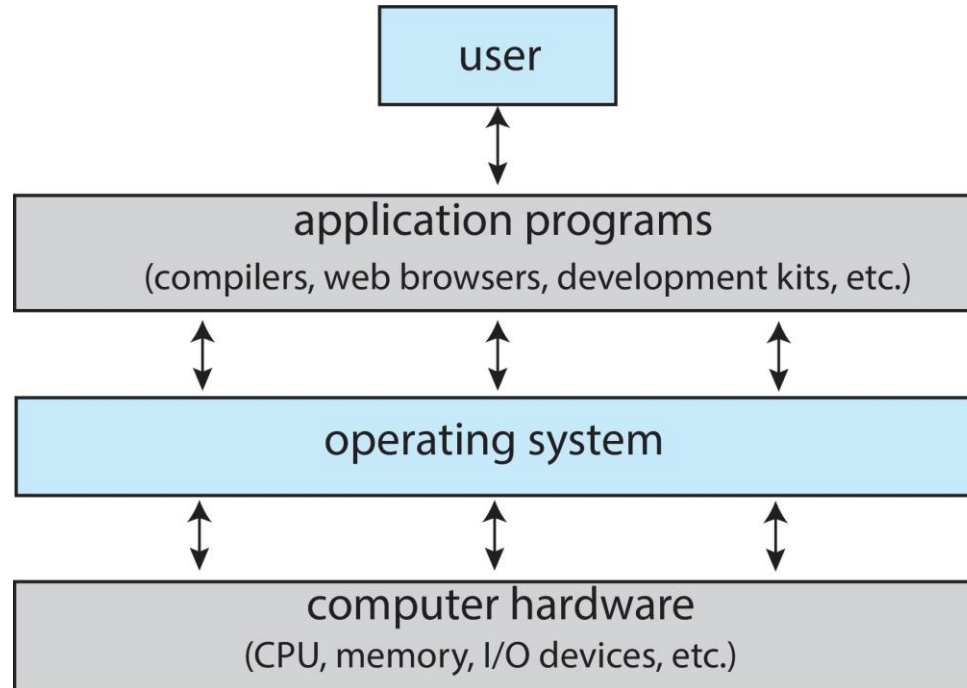
---

- 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





# Abstract View of Components of Computer





# What Operating Systems Do

---

- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
  - Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tables are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
  - Run primarily without user intervention







# Defining Operating Systems

---

- Term OS covers many roles
  - Because of myriad designs and uses of OSES
  - Present in toasters through ships, spacecraft, game machines, TVs and industrial control systems
  - Born when fixed use computers for military became more general purpose and needed resource management and program control





# Operating System Definition

---

- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is a good approximation
  - But varies wildly
- “The one program running at all times on the computer” is the **kernel**, part of the operating system
- Everything else is either
  - A **system program** (ships with the operating system, but not part of the kernel) , or
  - An **application program**, all programs not associated with the operating system
- Today’s OSES for general purpose and mobile computing also include **middleware** – a set of software frameworks that provide additional services to application developers such as databases, multimedia, graphics





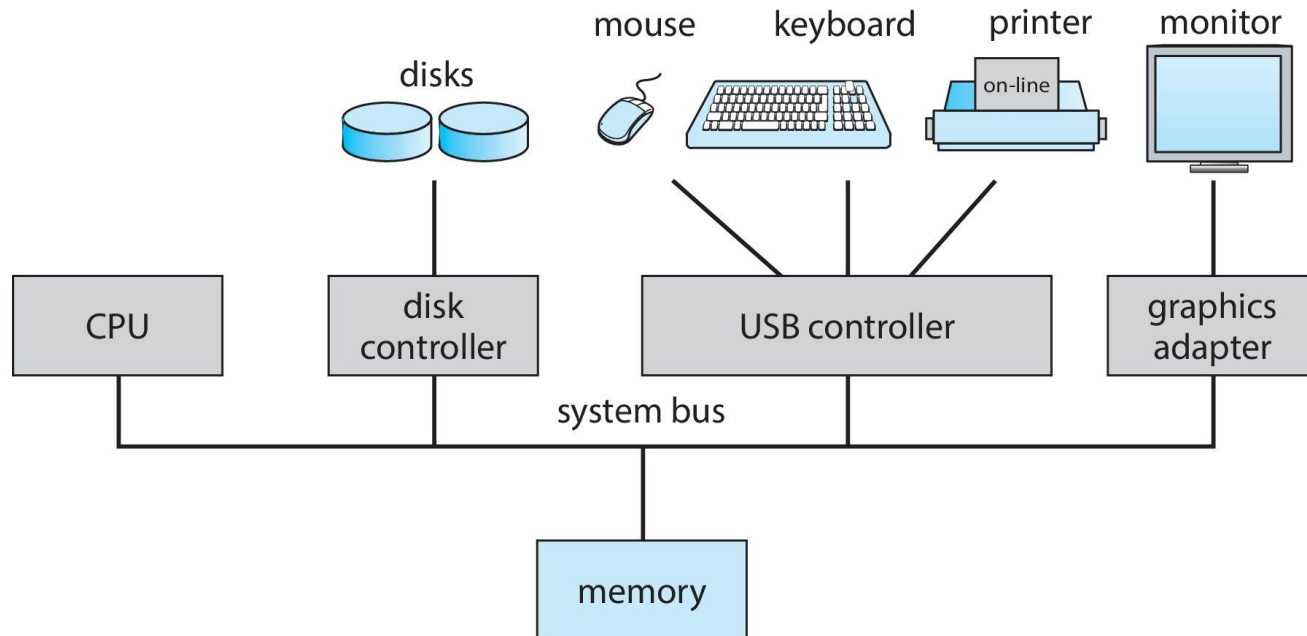
# Overview of Computer System Structure





# Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common **bus** providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles





# Computer-System Operation

---

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- Each device controller type has an operating system **device driver** to manage it
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an **interrupt**





# What are interrupts

---

- **Interrupt handling** is a crucial feature of computer systems that allows the CPU to respond to **asynchronous events**.
- Interrupts are signals generated by hardware devices or software programs to indicate that immediate attention is needed.
- Interrupt handling ensures that the system can manage these signals without losing track of the ongoing tasks, effectively balancing system performance and responsiveness.





# Interrupt examples

---

I/O devices generate interrupts to signal that they need service from the CPU. For example:

- A keyboard interrupt signals when a key is pressed.
- A network card sends an interrupt when a data packet has been received.
- A disk drive interrupts when a read/write operation is complete.





# Common Functions of Interrupts

---

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**







# I/O interrupt example

---

## I/O Interrupt in Practice: Example

Let's take the example of a **keyboard interrupt**:

The user presses a key on the keyboard.

The keyboard controller sends an interrupt request via its Interrupt Request(IRQ) line to the CPU.

The CPU stops what it's doing and saves the current program's state.

The CPU checks the interrupt vector table to find the Interrupt service routine (ISR) for the keyboard.

The CPU executes the ISR, which reads the key code from the keyboard buffer and places it in memory for the running program to use.

After the ISR finishes processing the key press, the CPU clears the interrupt and returns to the previously interrupted task.





# Interrupt Handling

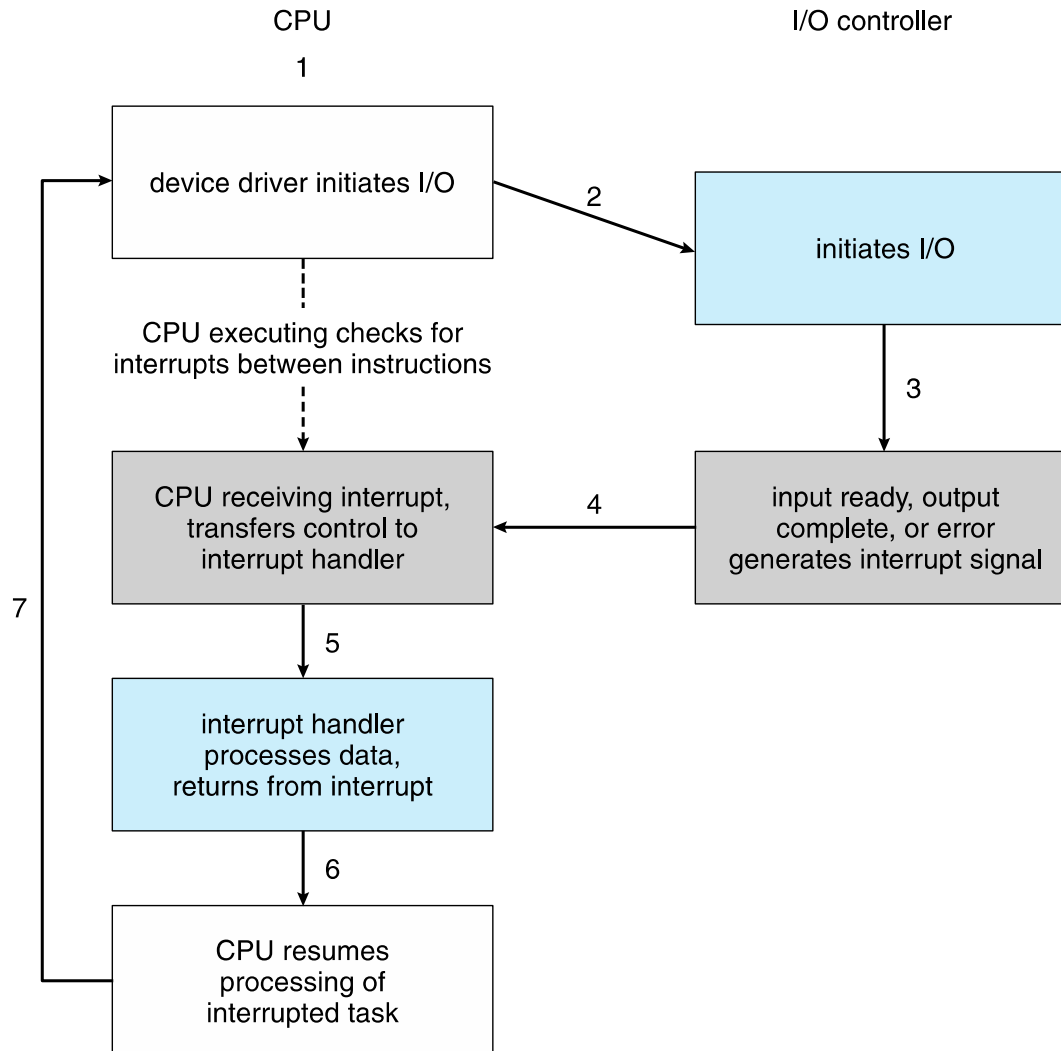
---

- The operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred:
- Separate segments of code determine what action should be taken for each type of interrupt





# Interrupt-drive I/O Cycle





# I/O Structure

---

- Two methods for handling I/O
  - After I/O starts, control returns to user program only upon I/O completion
  - After I/O starts, control returns to user program without waiting for I/O completion





# I/O Structure (Cont.)

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the OS to allow user to wait for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt





# I/O Subsystem

---

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
  - General device-driver interface
  - Drivers for specific hardware devices



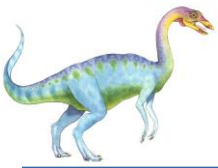


# Computer Startup

---

- **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





# Storage Structure







# Storage Structure

---

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
  - Typically **random-access memory** in the form of **Dynamic Random-access Memory (DRAM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity





# Storage Structure (Cont.)

---

- **Hard Disk Drives (HDD)** – 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
- **Non-volatile memory (NVM)** devices– faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular as **capacity and performance increases, price drops**





# Storage Definitions and Notation Review

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. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few. A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage. For example, most computers don't have an instruction to move a bit but do have one to move a byte. 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. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes. A **kilobyte**, or KB, is 1,024 bytes; a **megabyte**, or MB, is  $1,024^2$  bytes; a **gigabyte**, or GB, is  $1,024^3$  bytes; a **terabyte**, or TB, is  $1,024^4$  bytes; and a **petabyte**, or PB, is  $1,024^5$  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

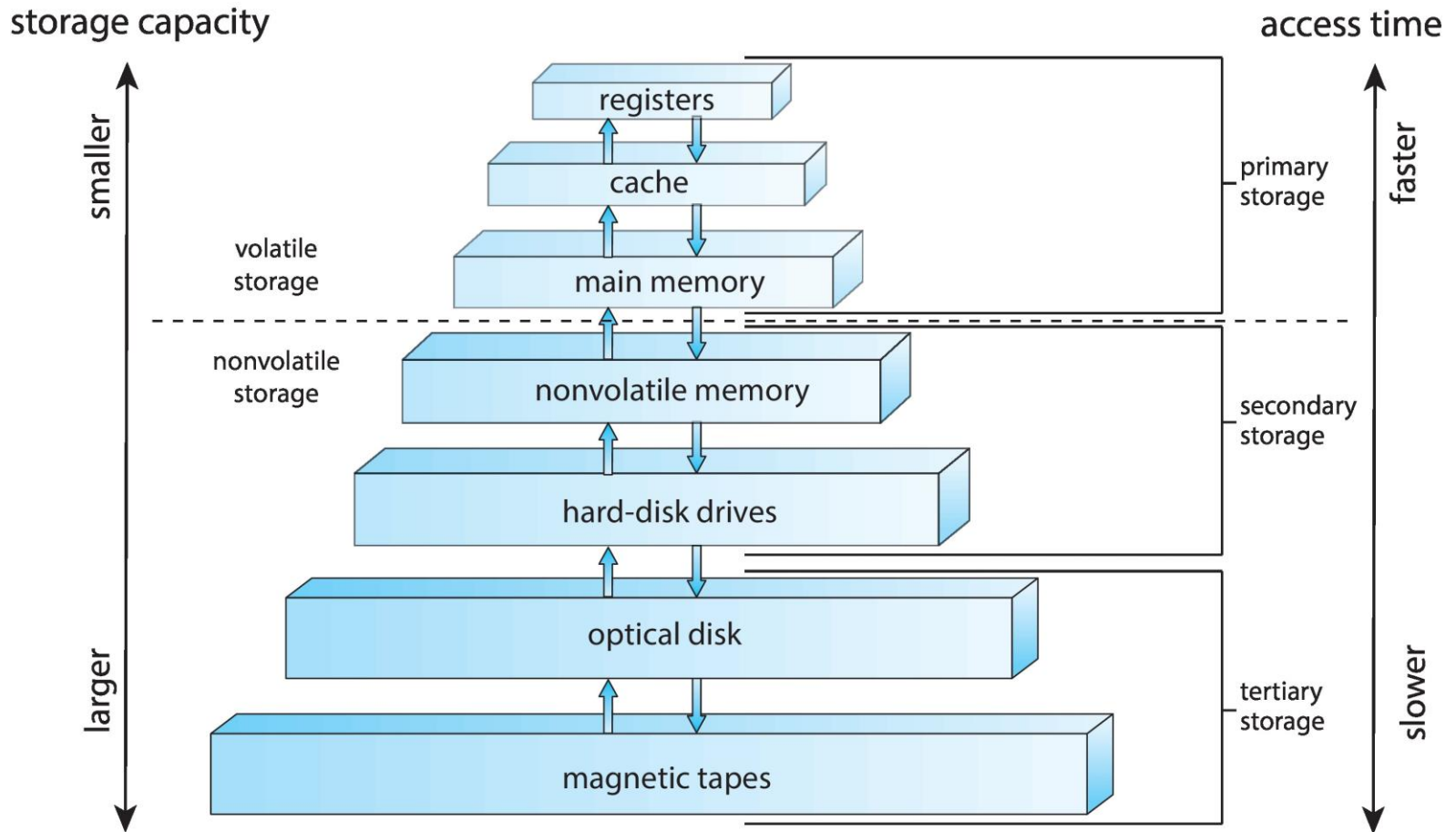
---

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information 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



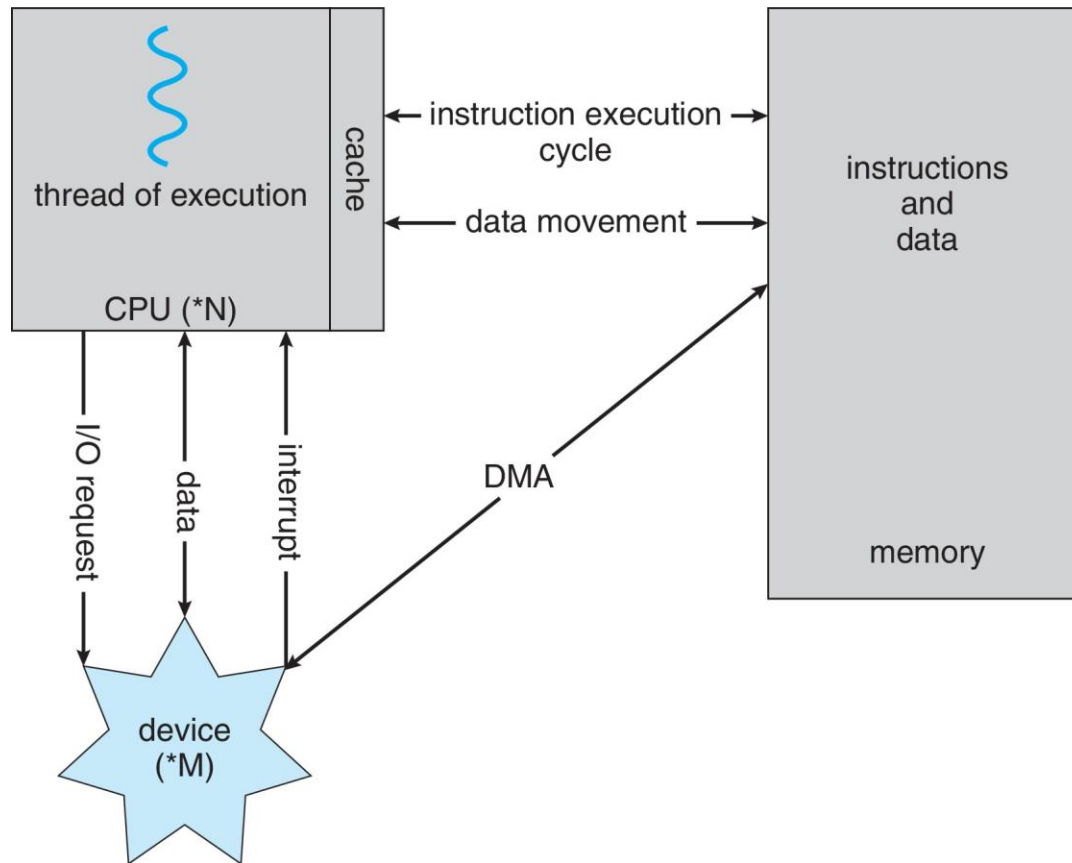


# Storage-Device Hierarchy





# How a Modern Computer Works



*A von Neumann architecture*





# Direct Memory Access Structure

---

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte





# Operating-System Operations

---

- Bootstrap program – simple code to initialize the system, load the kernel
- Kernel loads
- Starts **system daemons** (services provided outside of the kernel)
- Kernel **interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap**):
    - ▶ Software error (e.g., division by zero)
    - ▶ Request for operating system service – **system call**
    - ▶ Other process problems include infinite loop, processes modifying each other or the operating system







# Operating-System Operations

---

## **BIOS** (Basic Input/Output System)

- It is firmware embedded on a computer's motherboard, which is the first software that runs when a computer is powered on.
- Its primary function is to initialize and test the system's hardware components and load the operating system (OS) into the computer's memory from the storage device.
- Essentially, BIOS serves as the intermediary between the hardware and the operating system.





# The booting process

---

We can describe the boot process in six steps:

## 1. The Startup

It is the first step that involves switching the power ON. It supplies electricity to the main components like BIOS and processor.

## 2. BIOS: Power On Self Test

It is an initial test performed by the BIOS. Further, this test performs an initial check on the input/output devices, computer's main memory, disk drives, etc. Moreover, if any error occurs, the system produces a beep sound.

## 3. Loading of OS

In this step, the operating system is loaded into the main memory. The operating system starts working and executes all the initial files and instructions.





# The booting process

---

## 4. System Configuration

In this step, the drivers are loaded into the main memory. Drivers are programs that help in the functioning of the peripheral devices.

## 5. Loading System Utilities

System utilities are basic functioning programs, for example, volume control, antivirus, etc. In this step, system utilities are loaded into the memory.

## 6. User Authentication

If any password has been set up in the computer system, the system checks for user authentication. Once the user enters the login Id and password correctly the system finally starts.





# Multiprogramming (Batch system)

---

- Single user cannot always keep CPU and I/O devices busy
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- When job has to wait (for I/O for example), OS switches to another job

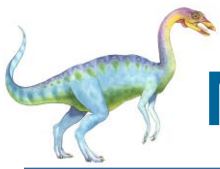




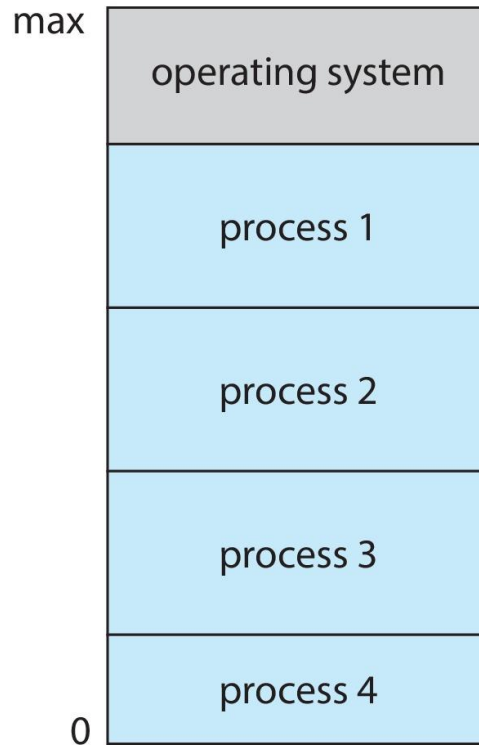
# Multitasking (Timesharing)

- A logical extension of Batch systems– the CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be  $< 1$  second
  - Each user has at least one program executing in memory  
⇒ **process**
  - If several jobs ready to run at the same time ⇒ **CPU scheduling**
  - If processes don't 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





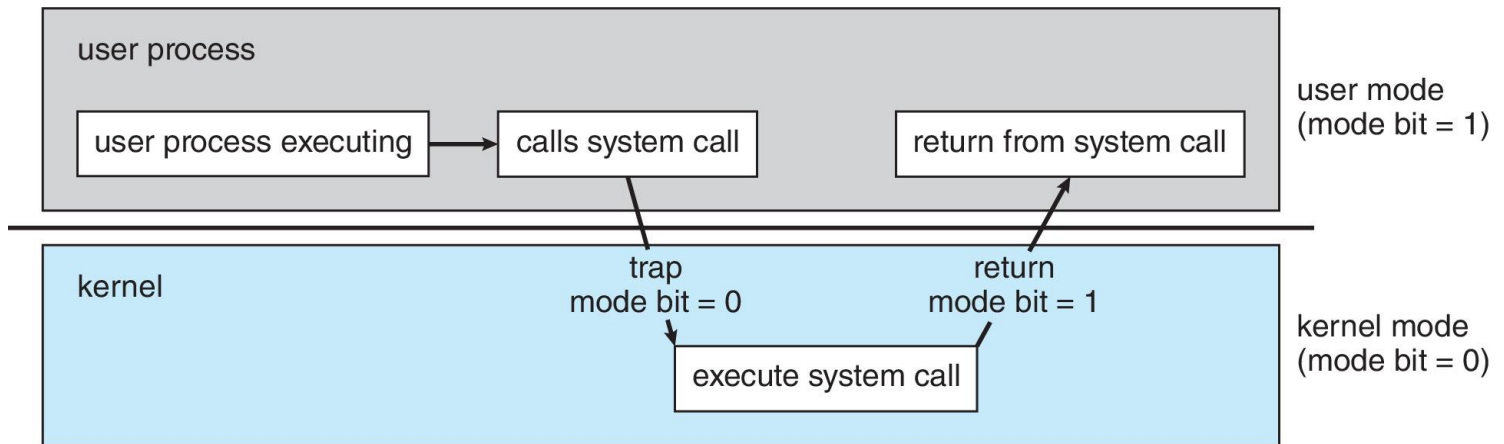
# Dual-mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
- **Mode bit** provided by hardware
  - Provides ability to distinguish when system is running user code or kernel code.
  - When a user is running  $\Rightarrow$  mode bit is “user”
  - When kernel code is executing  $\Rightarrow$  mode bit is “kernel”
- How do we guarantee that user does not explicitly set the mode bit to “kernel”?
  - System call changes mode to kernel, return from call resets it to user
- Some instructions designated as **privileged**, only executable in kernel mode





# Transition from User to Kernel Mode







# Timer

- Timer to prevent infinite loop (or process hogging resources)
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time





# Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a **passive entity**; process is an **active entity**.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads





## Multi-threaded

**Most software applications that run on modern computers and mobile devices are multithreaded.**

For example:

- A web browser might have **one thread** display images or text while **another thread** retrieves data from the network.
- A word processor may have a **thread** for displaying graphics, **another thread** for responding to keystrokes from the user, and a **third** thread for performing spelling and grammar checking in the background.





## II Process Management Activities

---

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling





# III Memory Management

---

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed





# IV File-system Management

---

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - ▶ Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - **OS activities include**
    - ▶ Creating and deleting files and directories
    - ▶ Primitives to manipulate files and directories
    - ▶ Mapping files onto secondary storage
    - ▶ Backup files onto stable (non-volatile) storage media





# V Mass-Storage Management

---

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- **OS activities**
  - Mounting and unmounting
  - Free-space management
  - Storage allocation
  - Disk scheduling
  - Partitioning
  - Protection





# Caching

---

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

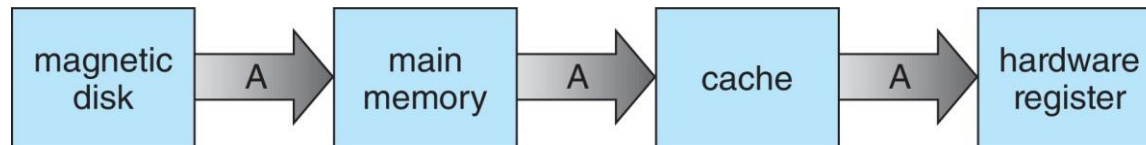






# Migration of data “A” from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 19

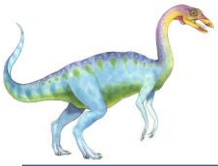




# VI Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- O. S determine who can access what parts of programs and data
- **Security** – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service,
- Systems generally first distinguish among users, to determine who can do what
  - User identities (**user IDs**, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
  - **Privilege escalation** allows user to change to effective ID with more rights





In security, external threats are involved ( for example antivirus programs)  
while in protection, internal threats are involved  
**A protection-oriented** system provides a means to distinguish between authorized and unauthorized usage,





# Distributed Systems

---

- Collection of separate, possibly heterogeneous, systems networked together
  - **Network** is a communications path, **TCP/IP** most common
    - ▶ **Local Area Network (LAN)**
    - ▶ **Wide Area Network (WAN)**
    - ▶ **Metropolitan Area Network (MAN)**
    - ▶ **Personal Area Network (PAN)**
- **Network Operating System** provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system





# Computer System Architecture





# Computer-System Architecture

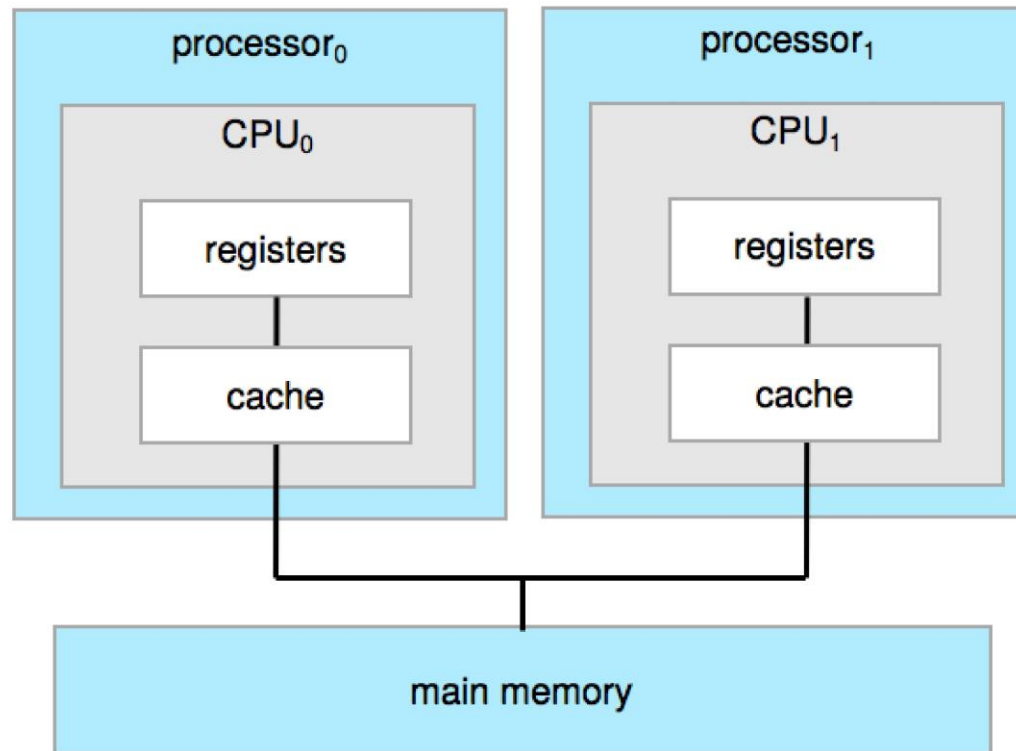
---

- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput ( Total amount of data/time)**
    2. **Economy of scale ( cost gained when increasing production)**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks





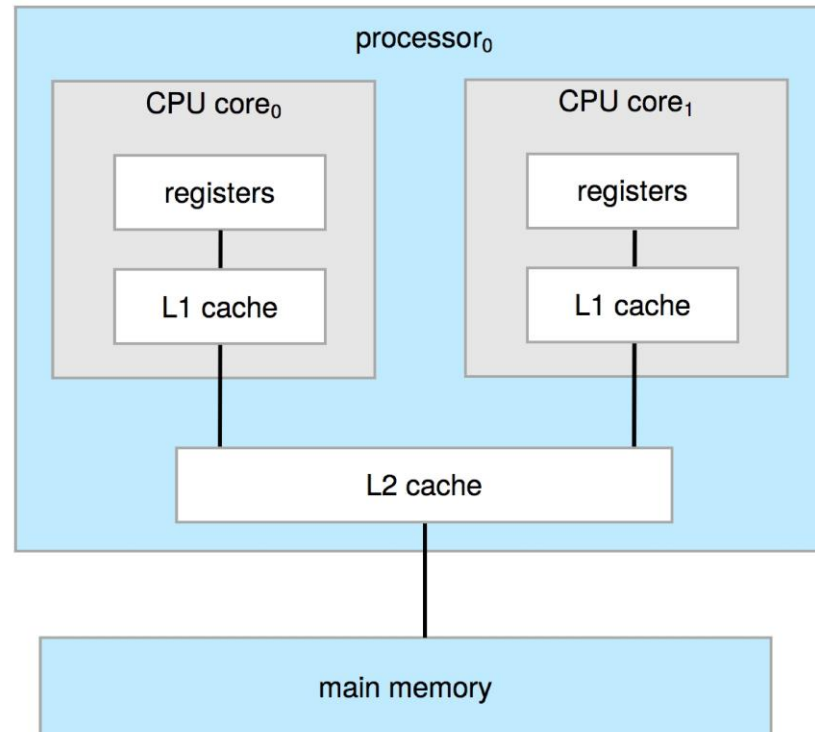
# Symmetric Multiprocessing Architecture





# Dual-Core Design

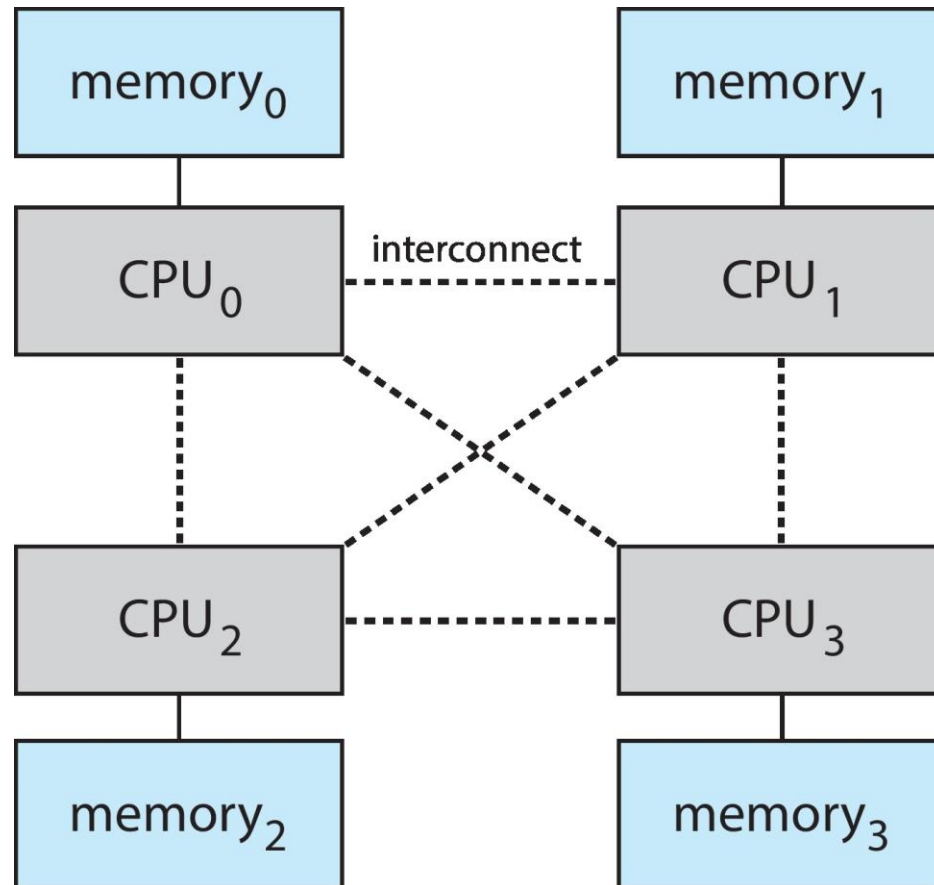
- Multi-chip and **multicore**
- Systems containing all chips
  - Chassis containing multiple separate systems







# Non-Uniform Memory Access System





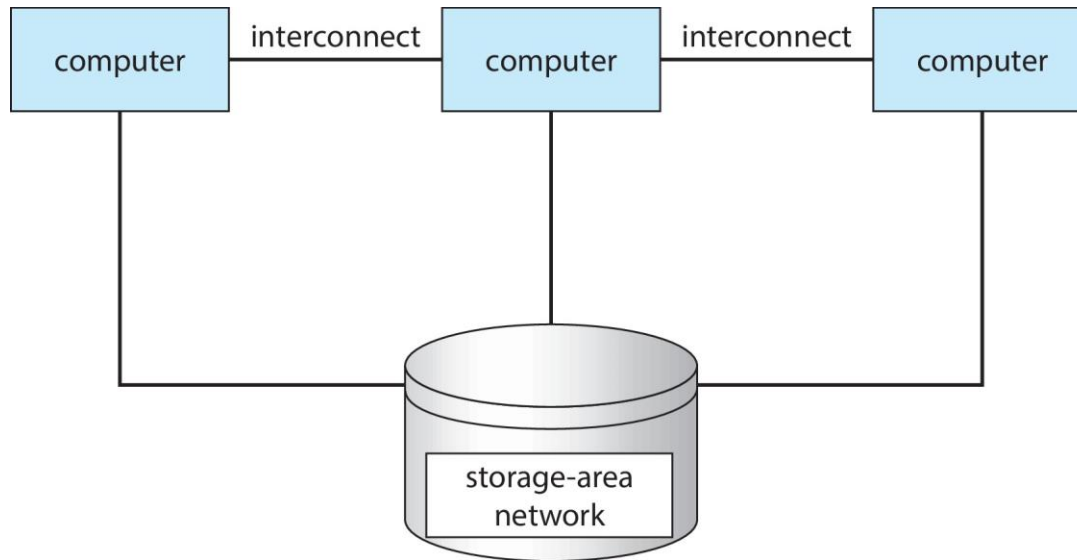
# Clustered Systems

- 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





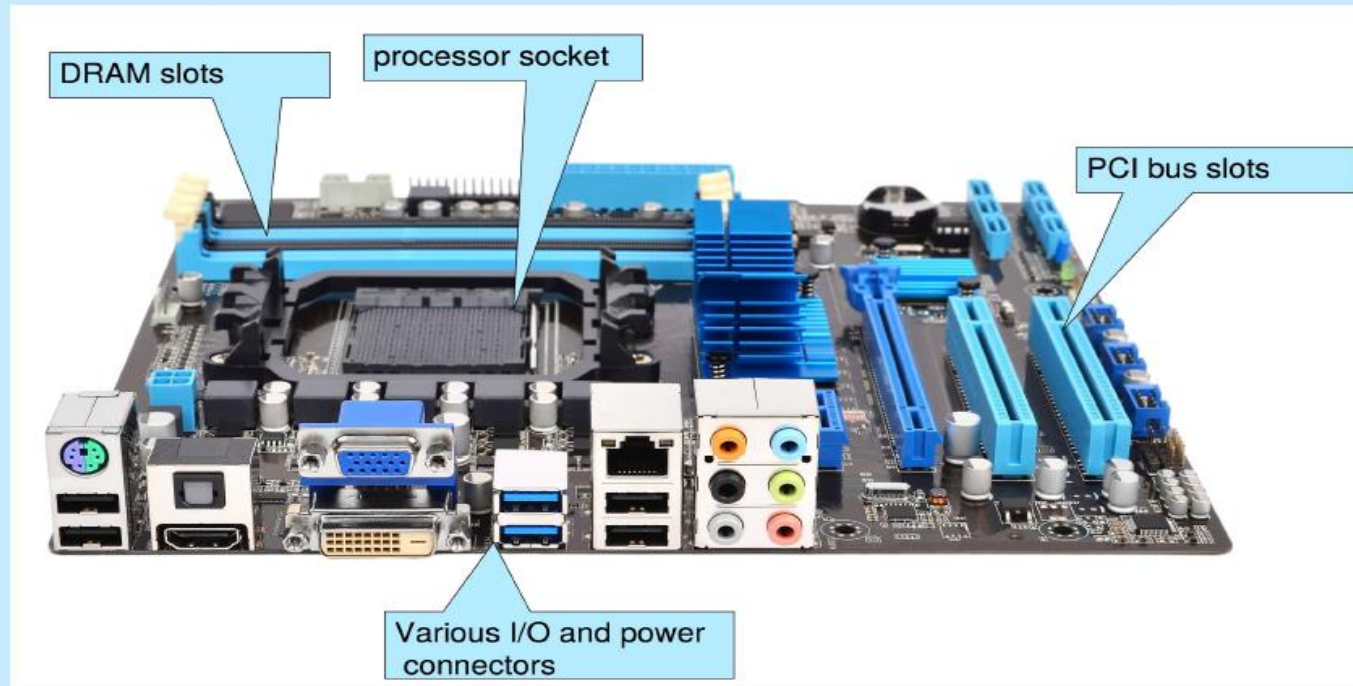
# Clustered Systems





# PC Motherboard

Consider the desktop PC motherboard with a processor socket shown below:



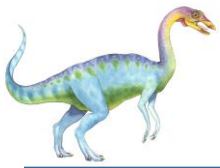
This board is a fully-functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.





# Computer System Environments



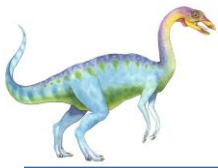


# Computing Environments

---

- Traditional
- Mobile
- Client Server
- Peer-to-Peer
- Cloud computing
- Real-time Embedded





# Traditional

---

- Stand-alone general-purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers (thin clients)** are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks





# Mobile

---

- Handheld smartphones, tablets, etc.
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

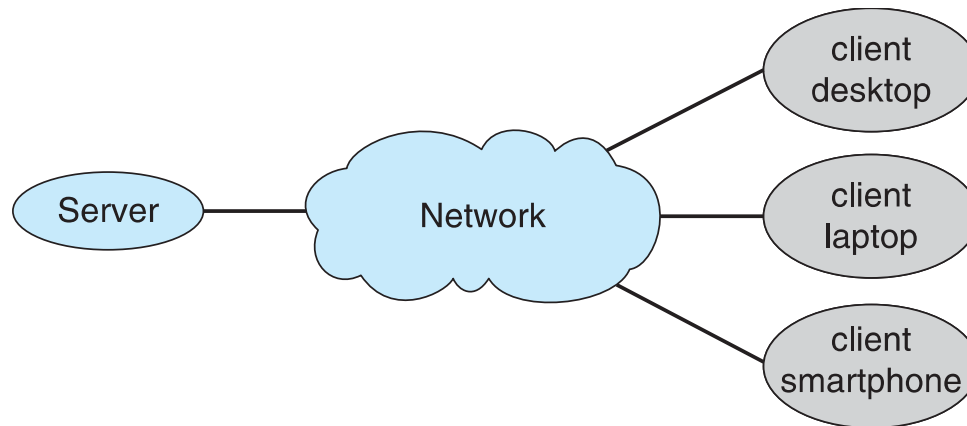






# Client Server

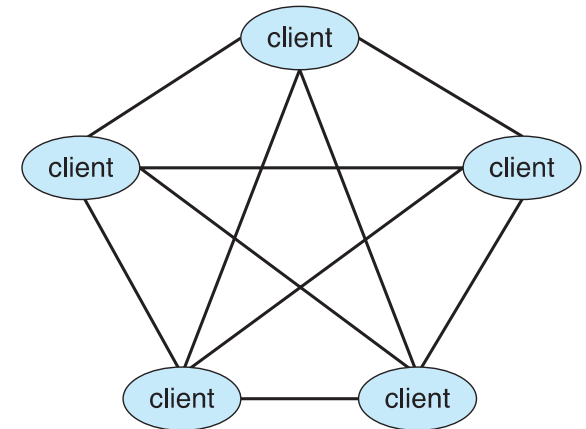
- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - ▶ **Compute-server system** provides an interface to client to request services (i.e., database)
    - ▶ **File-server system** provides interface for clients to store and retrieve files





# Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - ▶ Registers its service with central lookup service on network, or
    - ▶ Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, **Voice over IP (VoIP)** such as Skype





# Cloud Computing

---

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for its functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage





# Cloud Computing (Cont.)

---

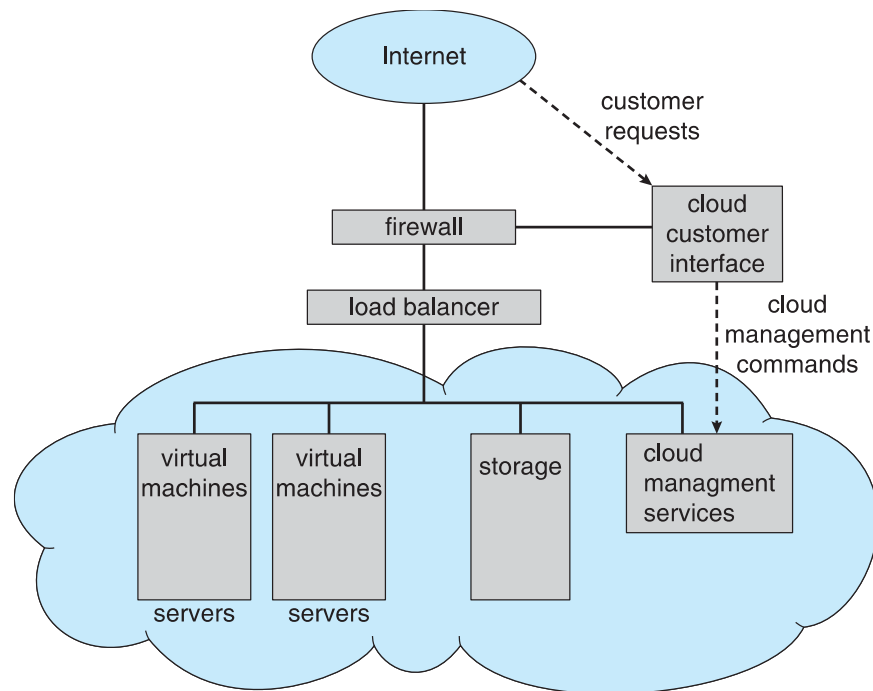
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)





# Cloud Computing (cont.)

- Cloud computing environments composed of traditional OSES, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



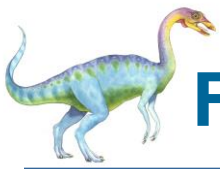


# Real-Time Embedded Systems

---

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met





# Free and Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source** and **proprietary**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has “copyleft” **GNU Public License (GPL)**
  - Free software and open-source software are two different ideas championed by different groups of people
    - ▶ <https://www.gnu.org/philosophy/open-source-misses-the-point.en.html>
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - <http://www.virtualbox.com>)
  - Use to run guest operating systems for exploration



# End of Chapter 1

---

