# NATIONAL ECONOMICS UNIVERSITY

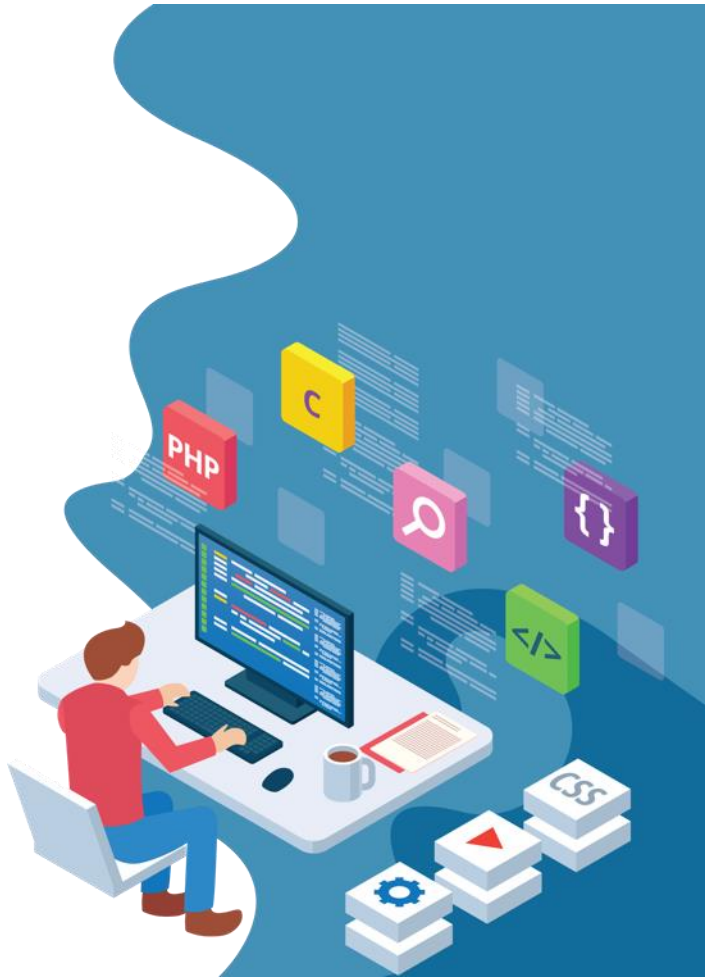
SCHOOL OF INFORMATION TECHNOLOGY AND DIGITAL ECONOMICS


## CHAPTER I

# INTRODUCTION

# OUTLINE

- What is an Operating System?
- Computer System Organization
- Operating-System Operations
- Process Management
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



# OBJECTIVES

- Describe the general organization of a computer system
- Describe the components in a modern, multiprocessor computer system
- Illustrate the process for booting an operating system
- Apply tools for monitoring operating system performance
- Discuss about factors affecting the development history of operating systems

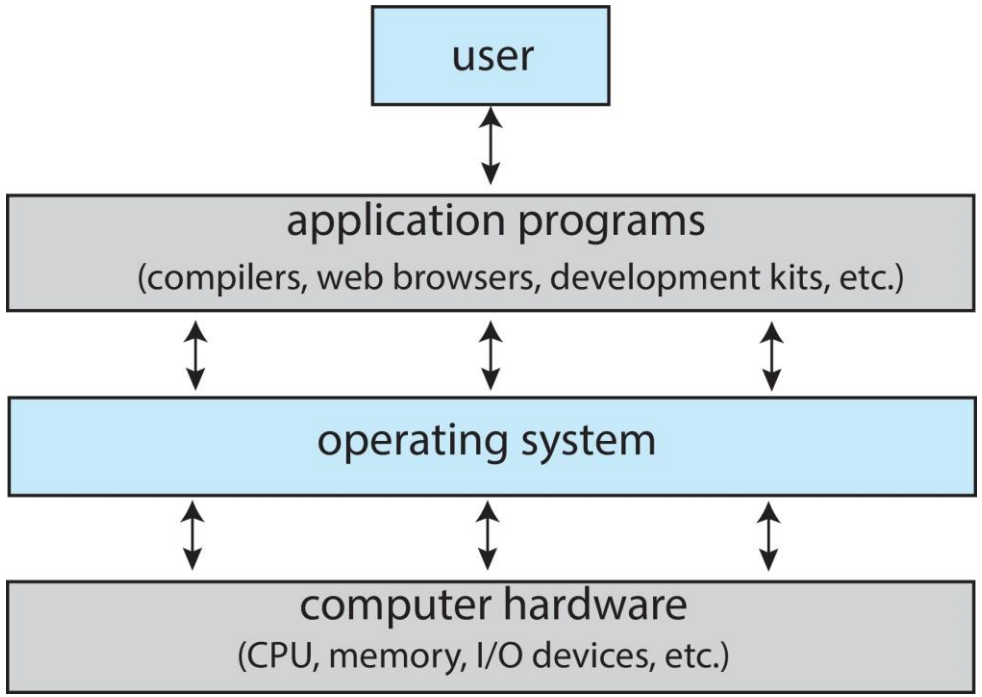
# WHAT IS AN OPERATING SYSTEM?

- 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

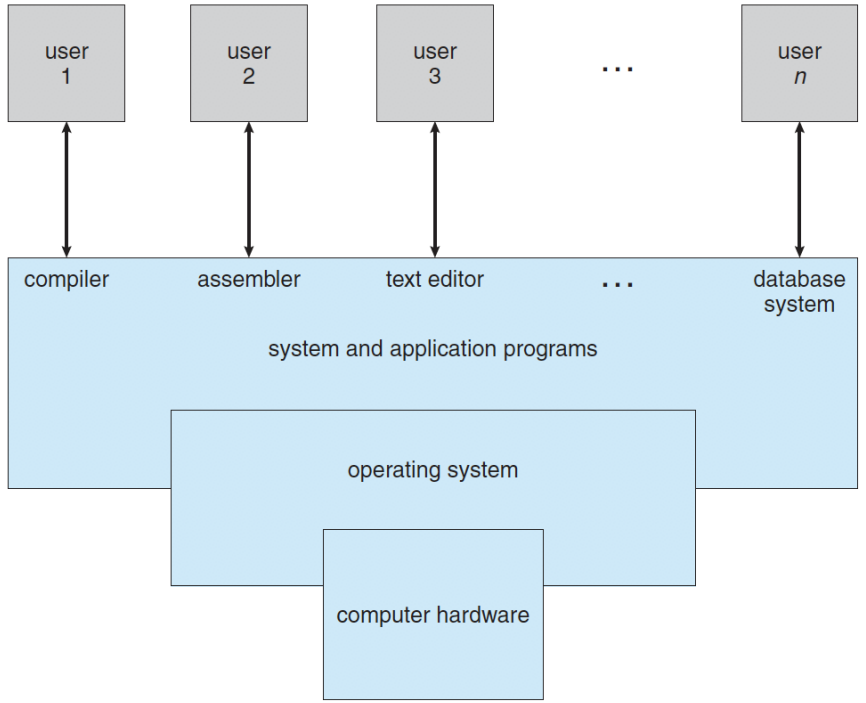
- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define the ways in which the system 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 IS AN OPERATING SYSTEM?

- An **operating system** is a program that manages a computer's hardware.
- It also provides a basis for application programs and acts as an **intermediary** between the computer user and the computer hardware.



# 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

# OPERATING SYSTEM DEFINITIONS

- 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 DEFINITIONS (CONT.)

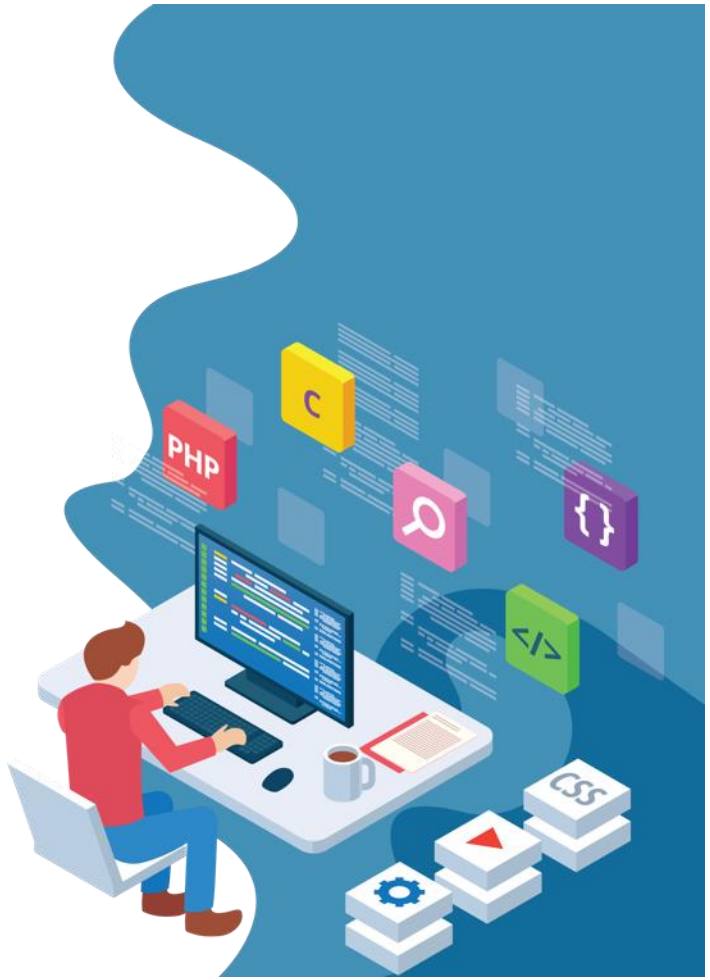
- No universally accepted definition
- **Resource allocator** – manages and allocates resources.
- **Control program** – controls the execution of user programs and operations of I/O devices .
- **Kernel** – the one program running at all times, Along with the kernel, there are two other types of programs:
  - 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.

# OPERATING SYSTEM GOALS

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

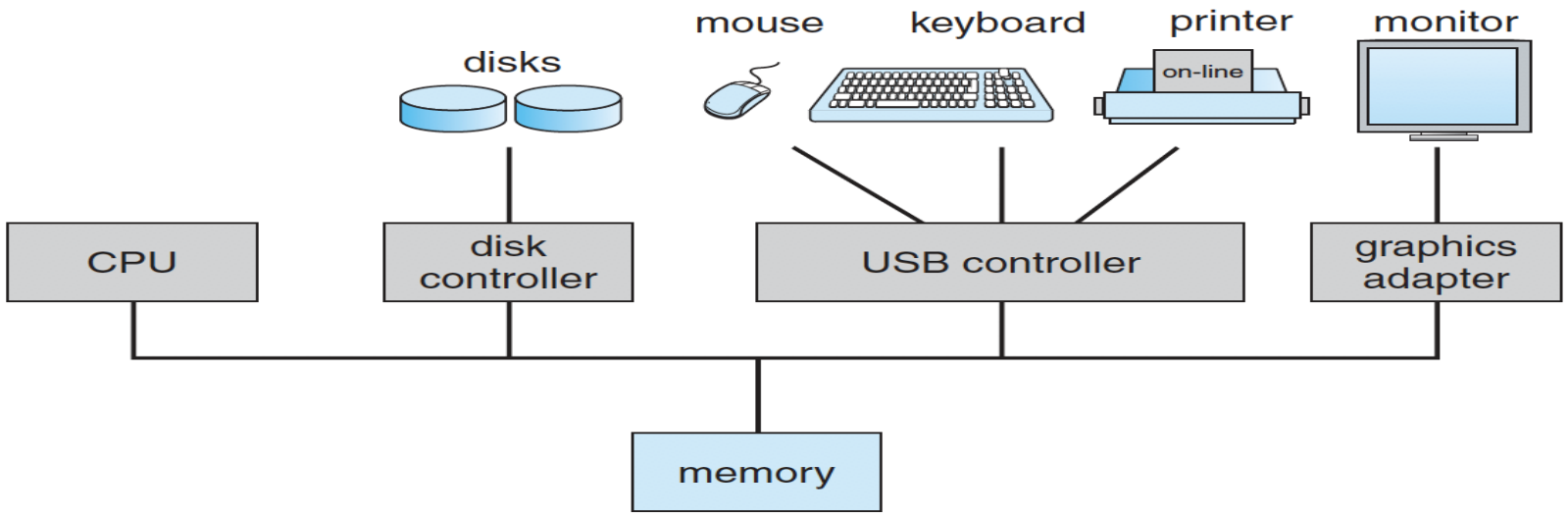
# OUTLINE

- What is an Operating System?
- Computer System Organization
- Operating-System Operations
- Process Management
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



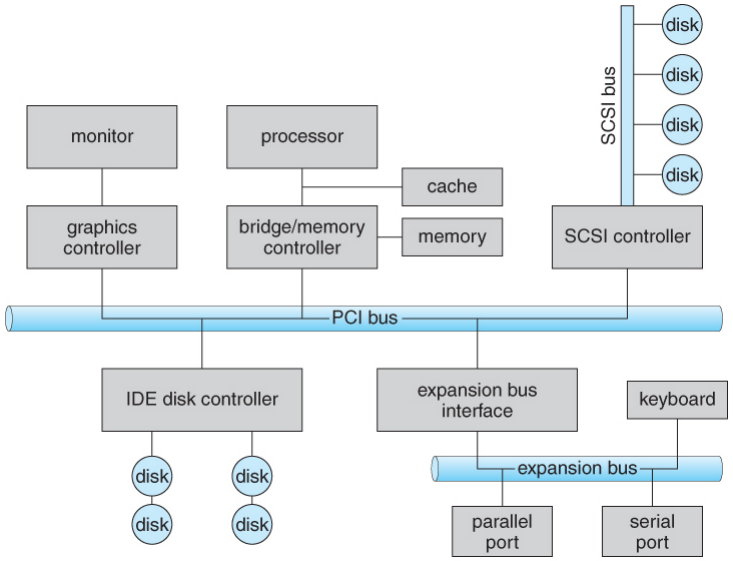
# COMPUTER-SYSTEM ORGANIZATION

- 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



# I/O STRUCTURE

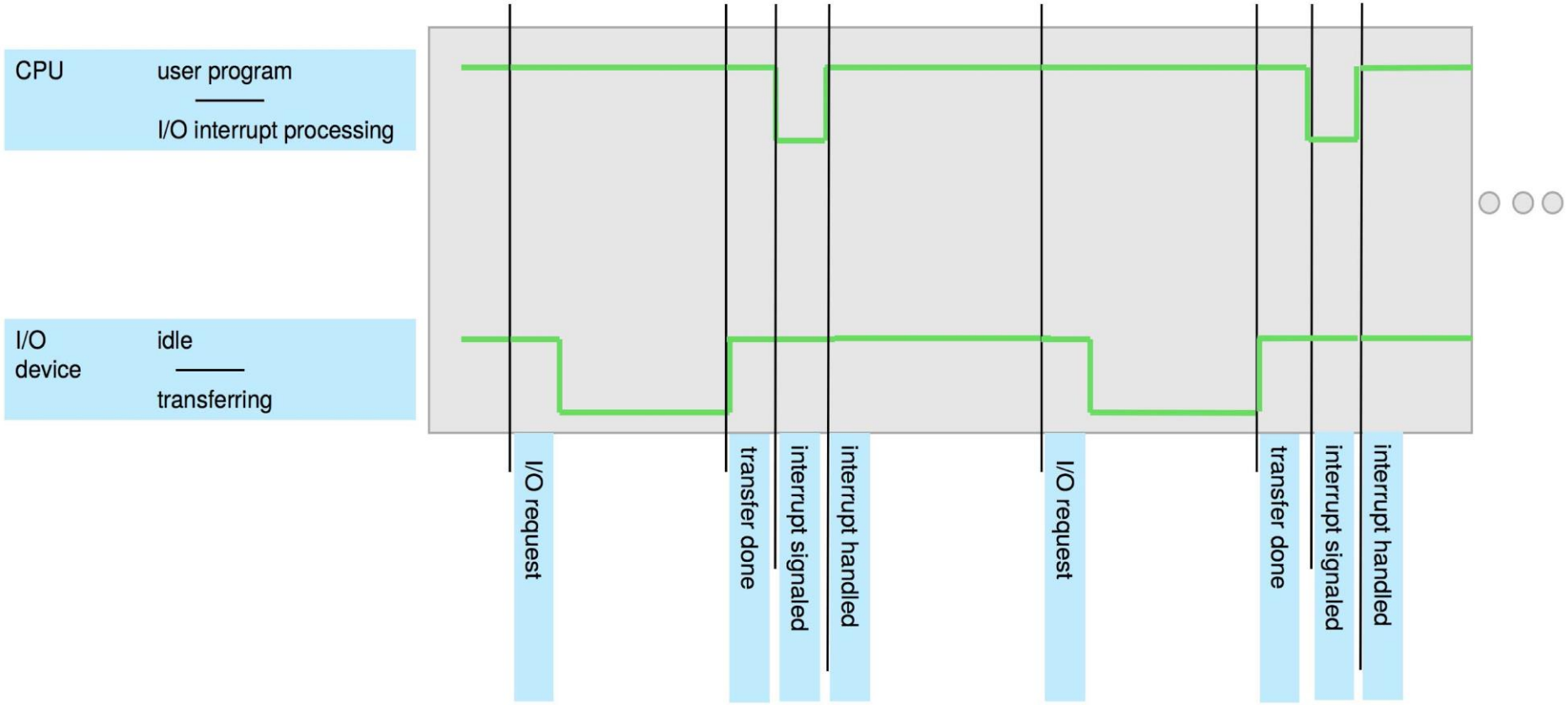
- I/O devices and the CPU can execute concurrently
- A computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- Each device controller is in charge of a specific type of device.
- A device controller maintains some local buffer storage and a set of special-purpose 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
- Device controller informs CPU that it has finished its operation by causing an **interrupt**



# 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**

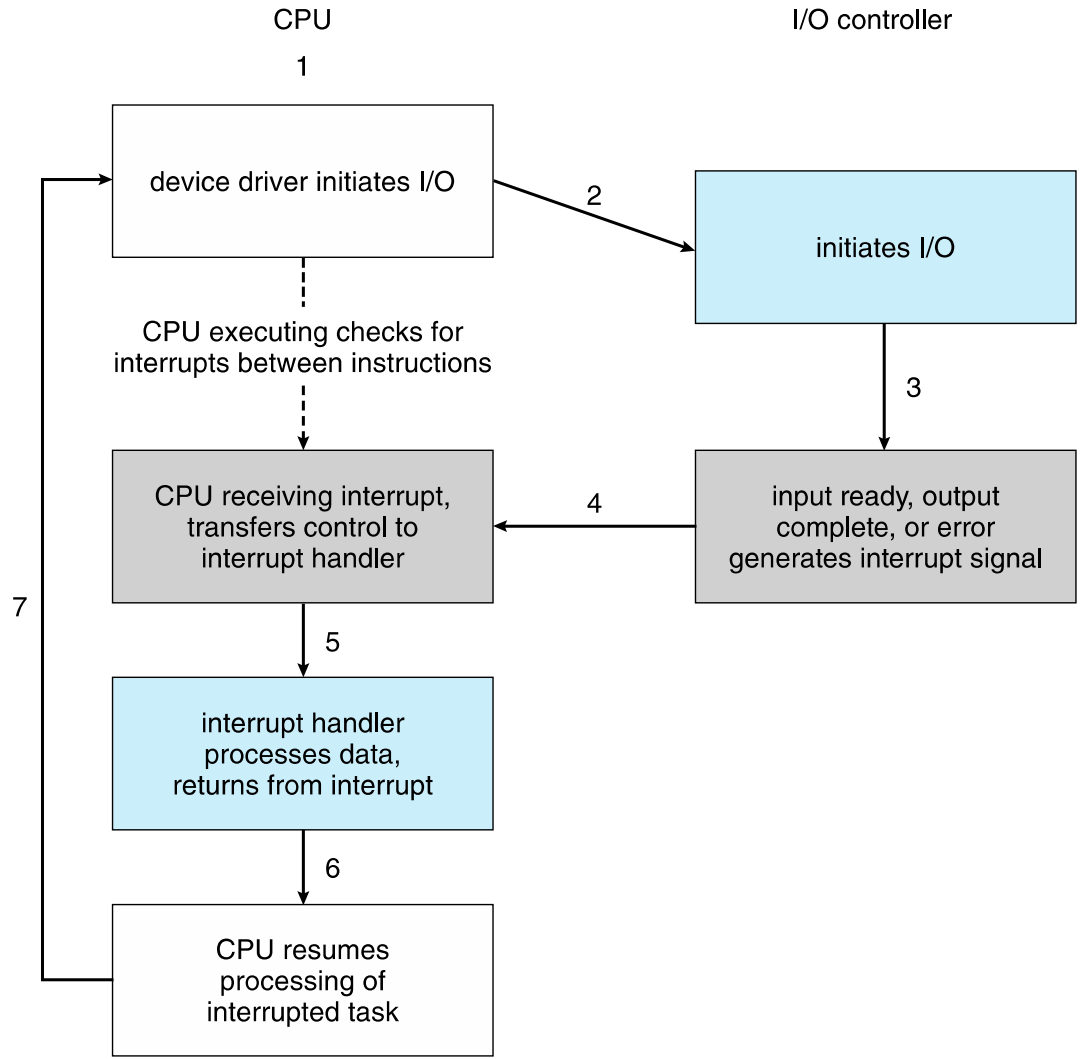
# INTERRUPT TIMELINE



# 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



# 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.)

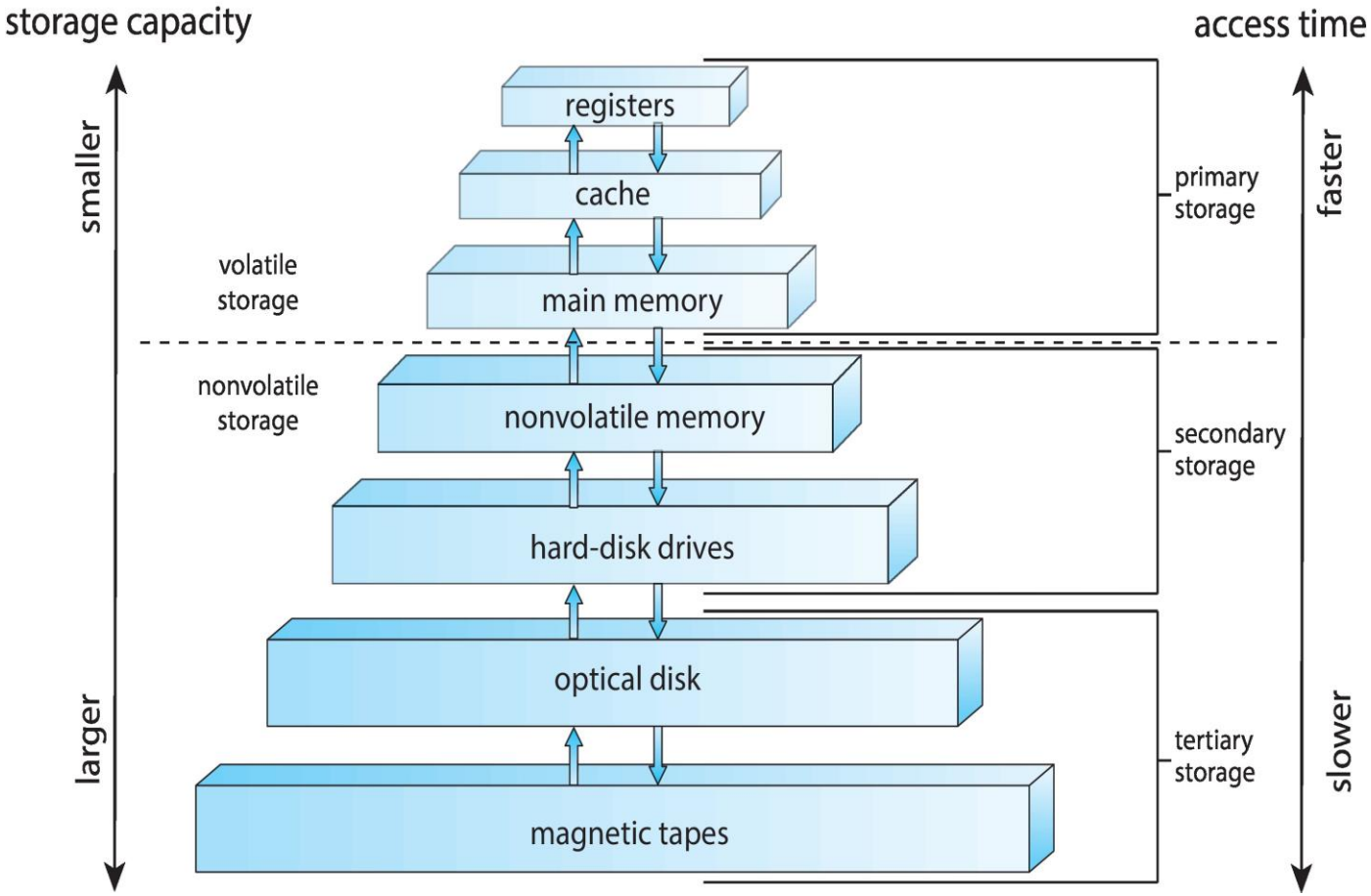
- **Hard Disk Drives** (**HDD**) – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Non-volatile memory** (**NVM**) devices– faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular as capacity and performance increases, price drops

# STORAGE HIERARCHY

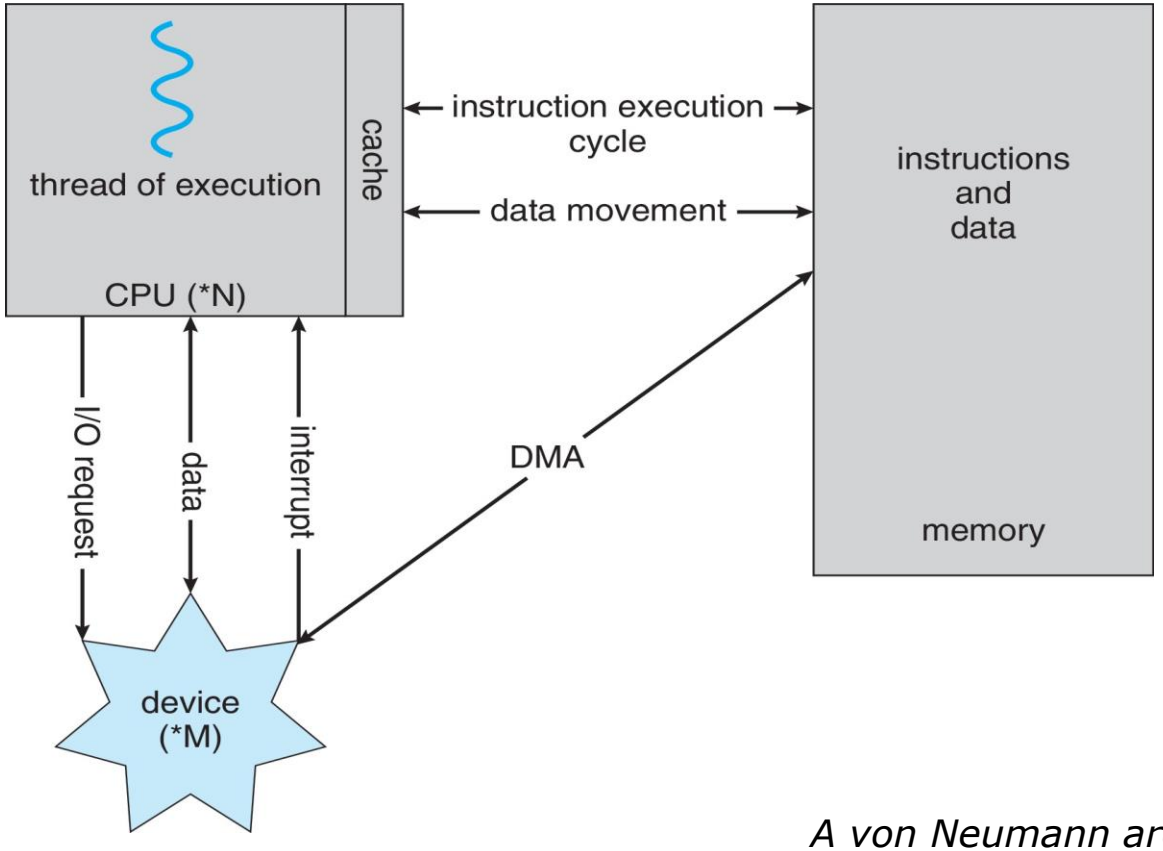
- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

# STORAGE STRUCTURE

- The CPU can load instructions only from **main memory**, so any programs to run must be stored there.
- A **register** may hold an instruction, a storage address, or any kind of data.
- Most computer systems provide **secondary storage** as an extension of main memory



# 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

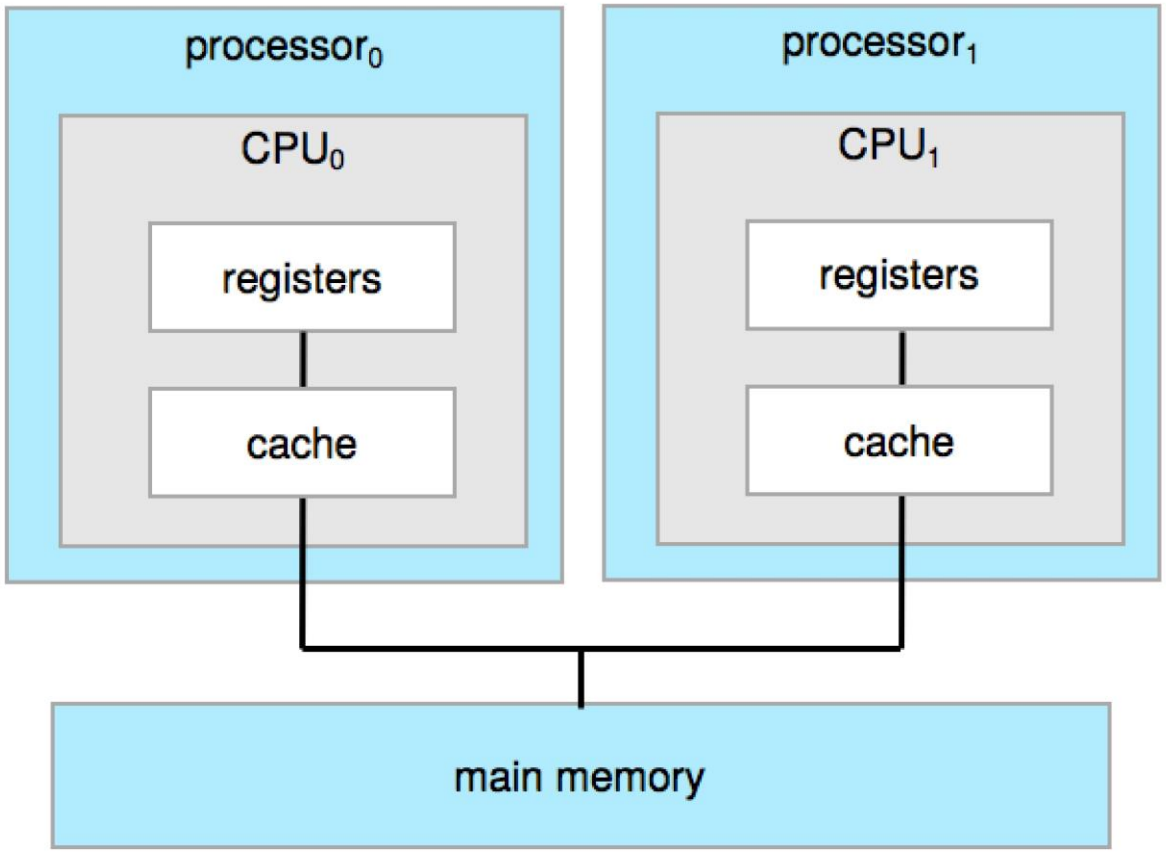
# COMPUTER STARTUP

- **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 ARCHITECTURE

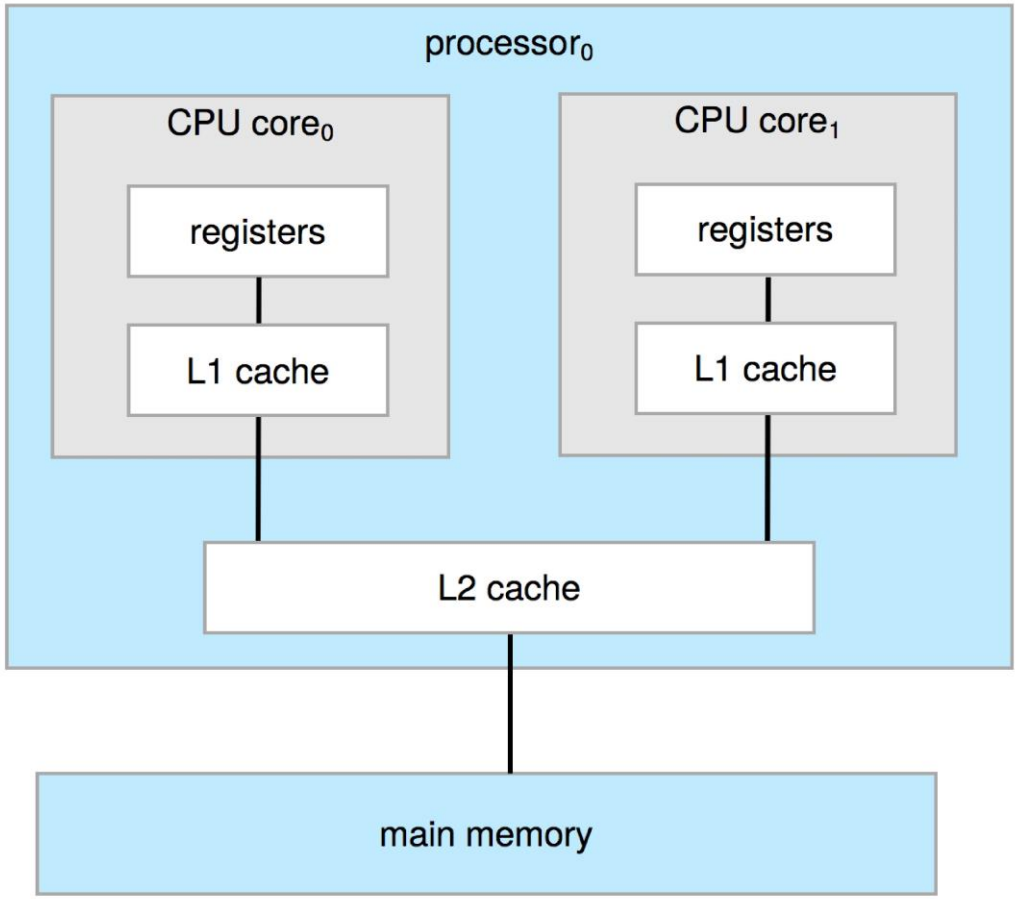
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    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

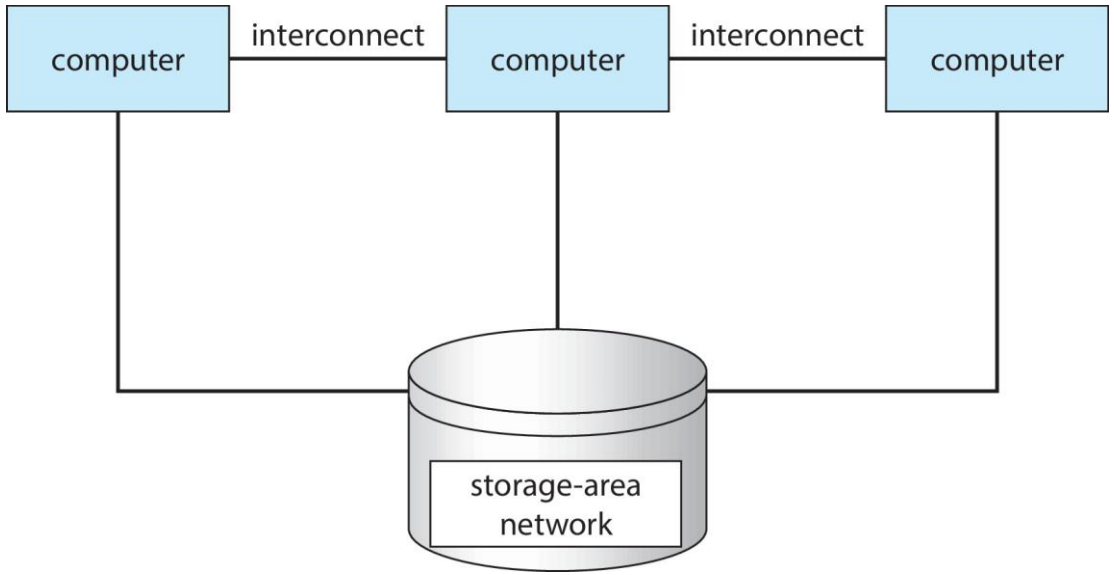
- Multi-chip and **multicore**
- Systems containing all  chips
  - Chassis containing multiple separate systems



# CLUSTERED SYSTEMS

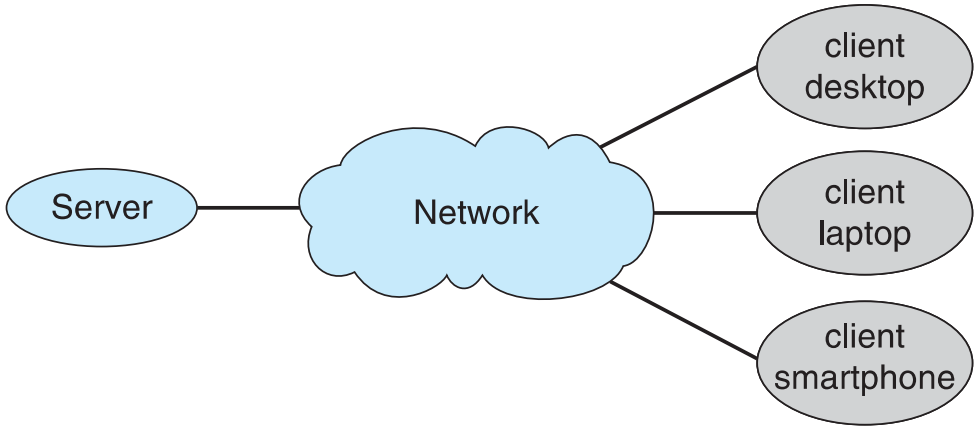
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network** (**SAN**)
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** 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



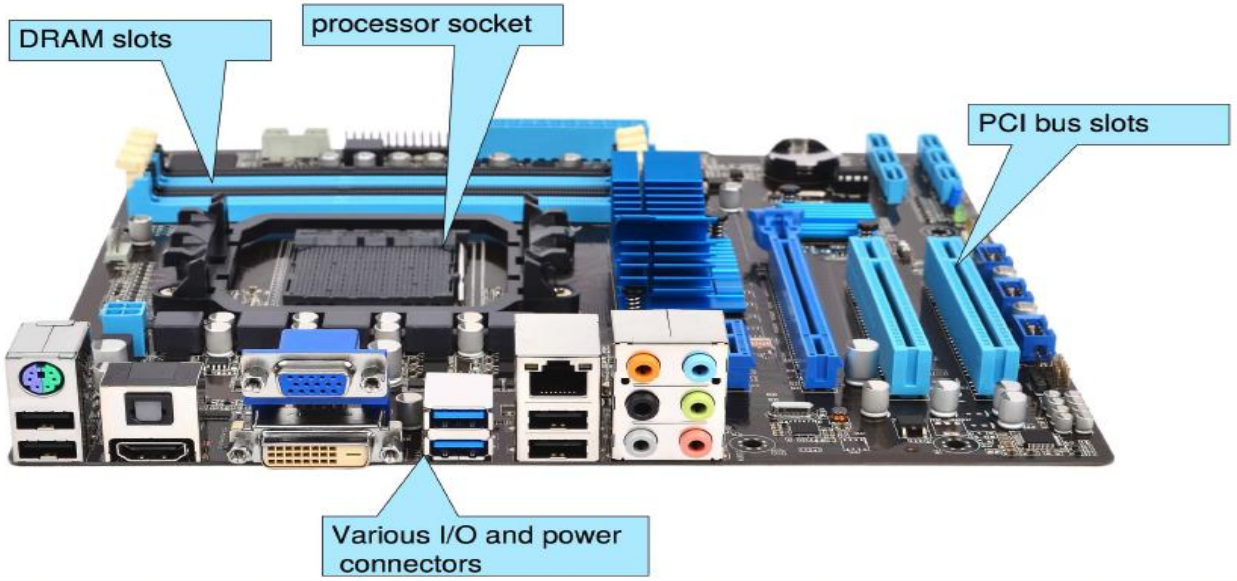
# CLIENT SERVER

- 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



# PC MOTHERBOARD

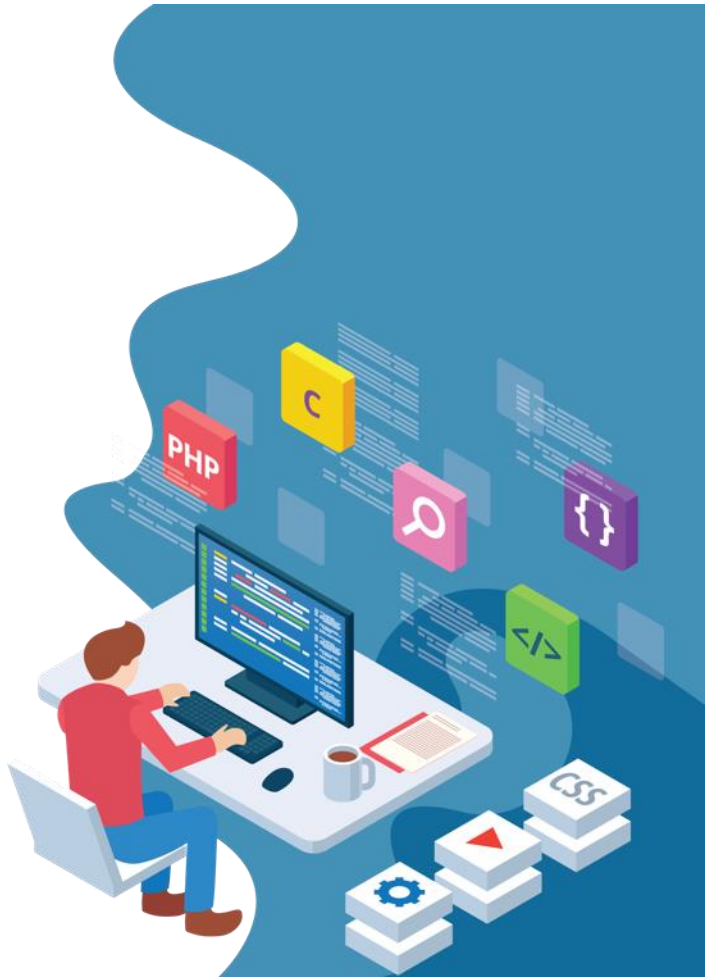
Consider the desktop PC motherboard with a processor socket shown below:




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.

# OUTLINE

- What is an Operating System?
- Computer System Organization
- Operating-System Operations
- Process Management
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



# 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

# 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

# MULTIPROGRAM

- One of the most important aspects of operating systems is the ability to **multiprogram**
- The idea is as follows:
  - The operating system keeps several jobs in memory simultaneously.
  - In general, main memory is too small to accommodate all jobs, the jobs are kept initially on the disk in the **job pool**.
  - This pool consists of all processes residing on disk awaiting allocation of main memory.

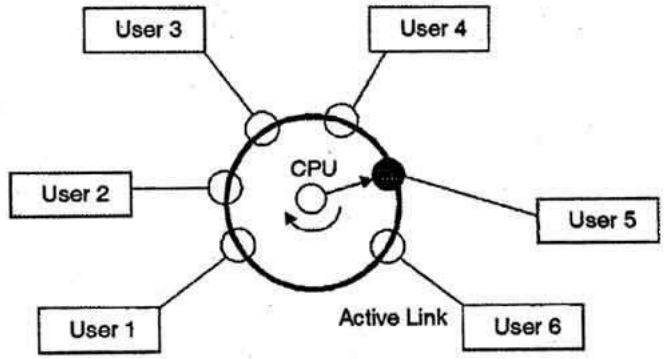
# MULTIPROGRAM

- The set of jobs in memory can be a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in memory.
- Eventually, the job may have to wait for some task, such as an I/O operation, to complete

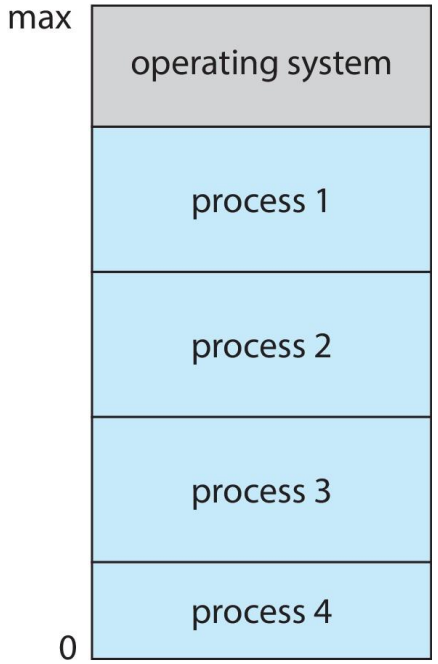
How about **non-multiprogrammed** system?

# TIME SHARING (OR MULTITASKING)

- 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



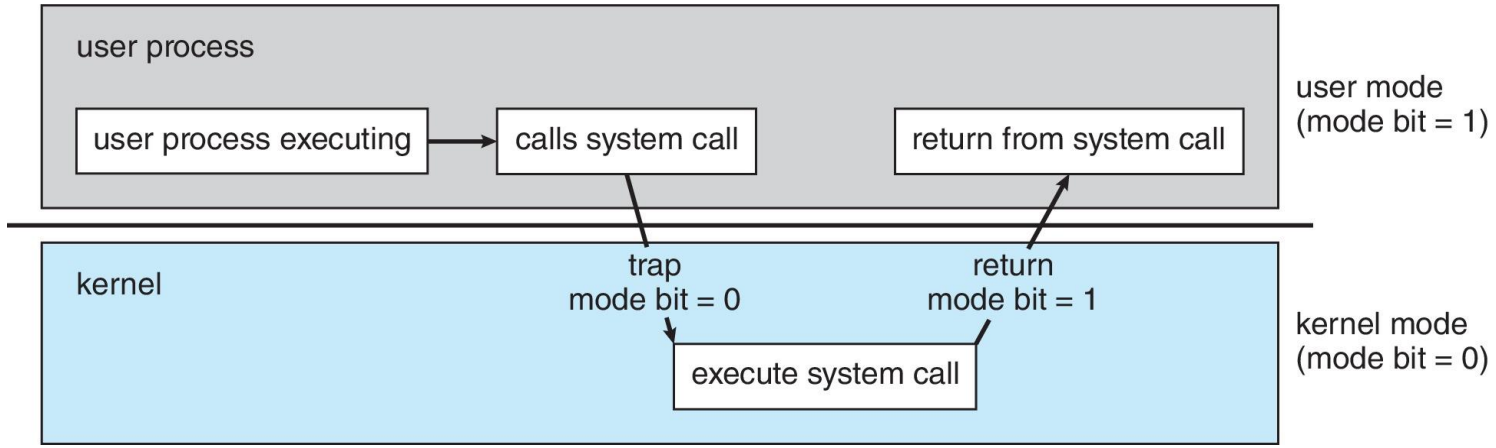
# MEMORY LAYOUT FOR MULTIPROGRAMMED SYSTEM



# 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 ⇨ mode bit is “user”
  - When kernel code is executing ⇨ 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



# EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

*EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS*

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

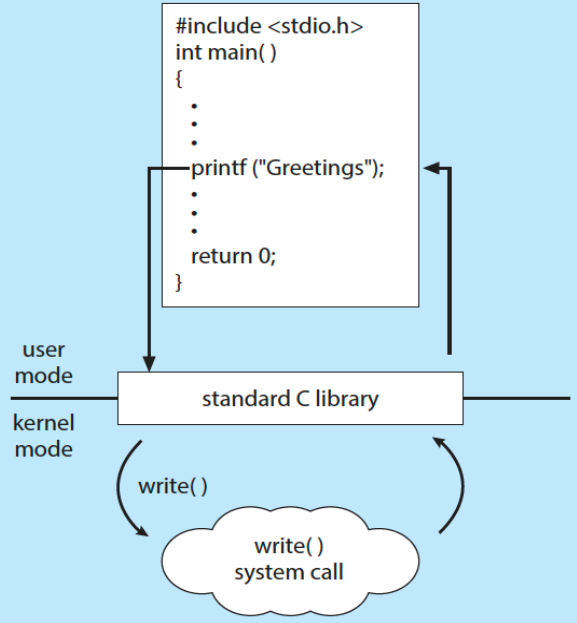
| | Windows | Unix |
|---|---|---|
| **Process control** | `CreateProcess()`<br>`ExitProcess()`<br>`WaitForSingleObject()` | `fork()`<br>`exit()`<br>`wait()` |
| **File management** | `CreateFile()`<br>`ReadFile()`<br>`WriteFile()`<br>`CloseHandle()` | `open()`<br>`read()`<br>`write()`<br>`close()` |
| **Device management** | `SetConsoleMode()`<br>`ReadConsole()`<br>`WriteConsole()` | `ioctl()`<br>`read()`<br>`write()` |
| **Information maintenance** | `GetCurrentProcessID()`<br>`SetTimer()`<br>`Sleep()` | `getpid()`<br>`alarm()`<br>`sleep()` |
| **Communications** | `CreatePipe()`<br>`CreateFileMapping()`<br>`MapViewOfFile()` | `pipe()`<br>`shm_open()`<br>`mmap()` |
| **Protection** | `SetFileSecurity()`<br>`InitlializeSecurityDescriptor()`<br>`SetSecurityDescriptorGroup()` | `chmod()`<br>`umask()`<br>`chown()` |

# STANDARD C LIBRARY EXAMPLE

- C program invoking printf() library call, which calls write() system call

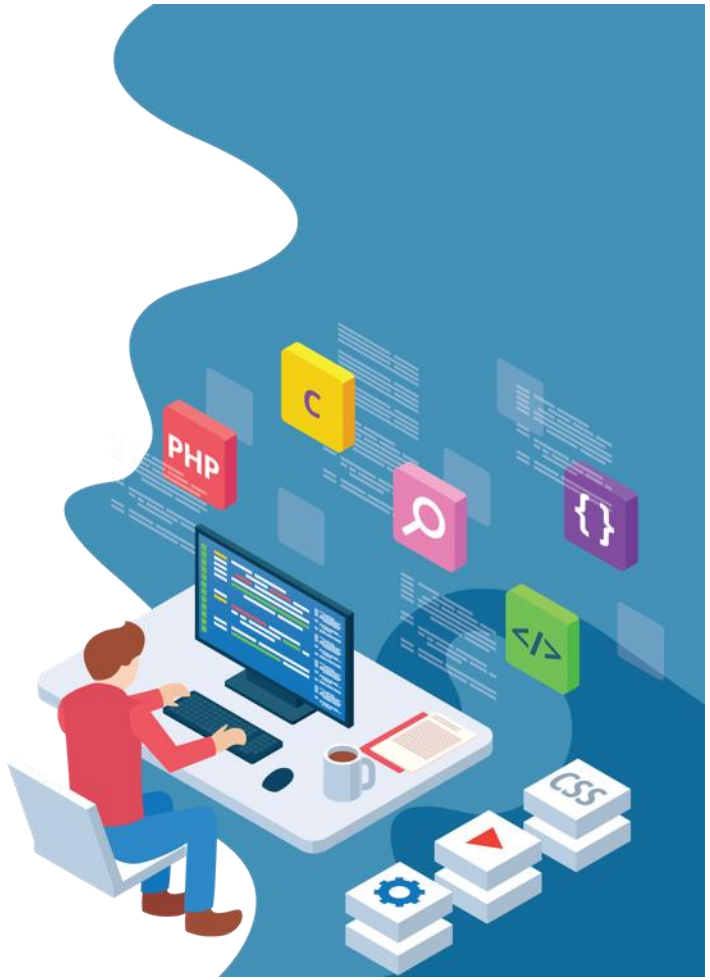
***THE STANDARD C LIBRARY***

The standard C library provides a portion of the system-call interface for many versions of UNIX and Linux. As an example, let's assume a C program invokes the `printf()` statement. The C library intercepts this call and invokes the necessary system call (or calls) in the operating system—in this instance, the `write()` system call. The C library takes the value returned by `write()` and passes it back to the user program:



# OUTLINE

- What is an Operating System?
- Computer System Organization
- Operating-System Operations
- Process Management
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



# 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

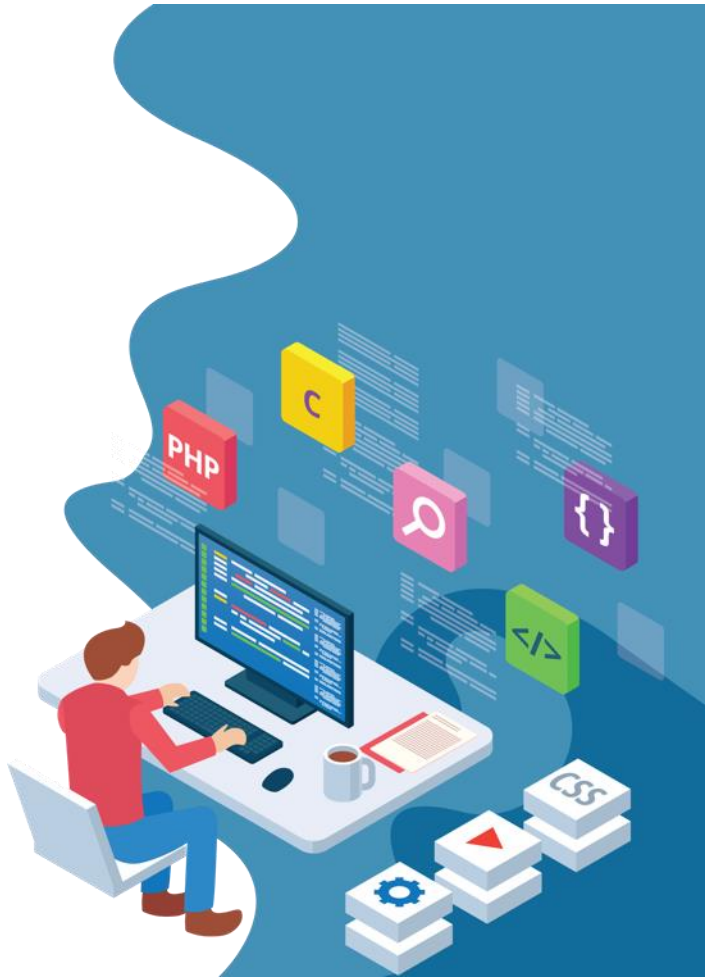
# PROCESS MANAGEMENT ACTIVITIES

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

# OUTLINE

- What is an Operating System?
- Computer System Organization
- Operating-System Operations
- Process Management
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



# 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

# 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

# MASS-STORAGE MANAGEMENT

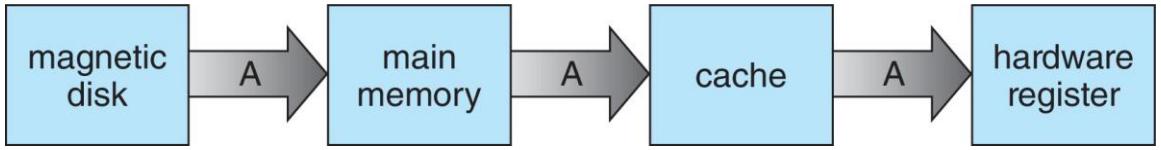
- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting
  - Free-space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

# I/O SUBSYSTEM

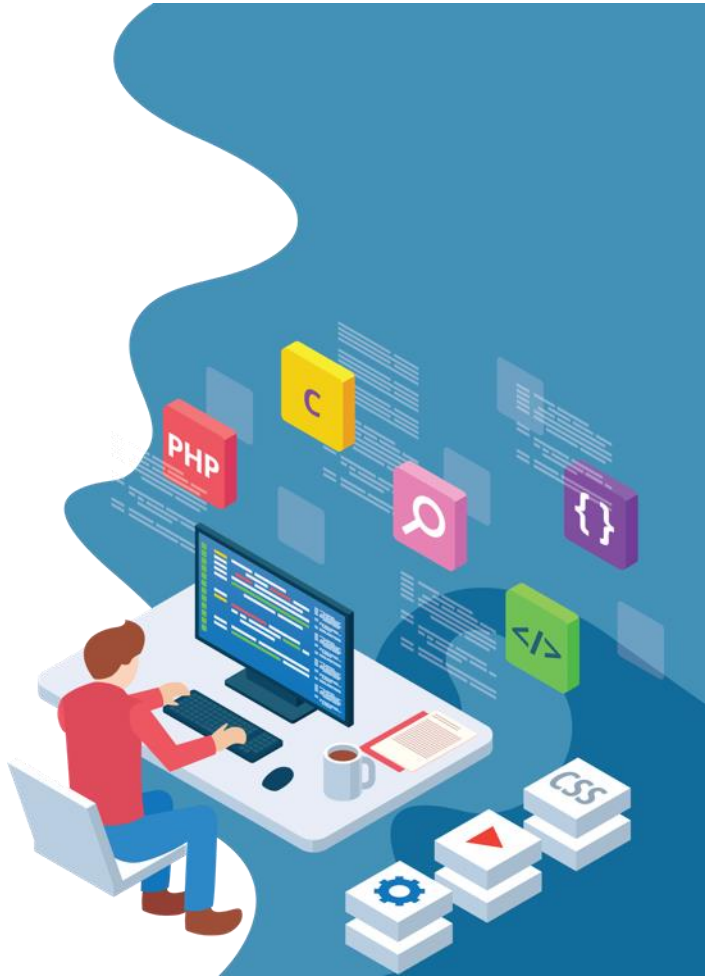
- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping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

# PROTECTION AND SECURITY

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **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

# OUTLINE

- What is an Operating System?
- Computer System Organization
- Operating-System Operations
- Process Management
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



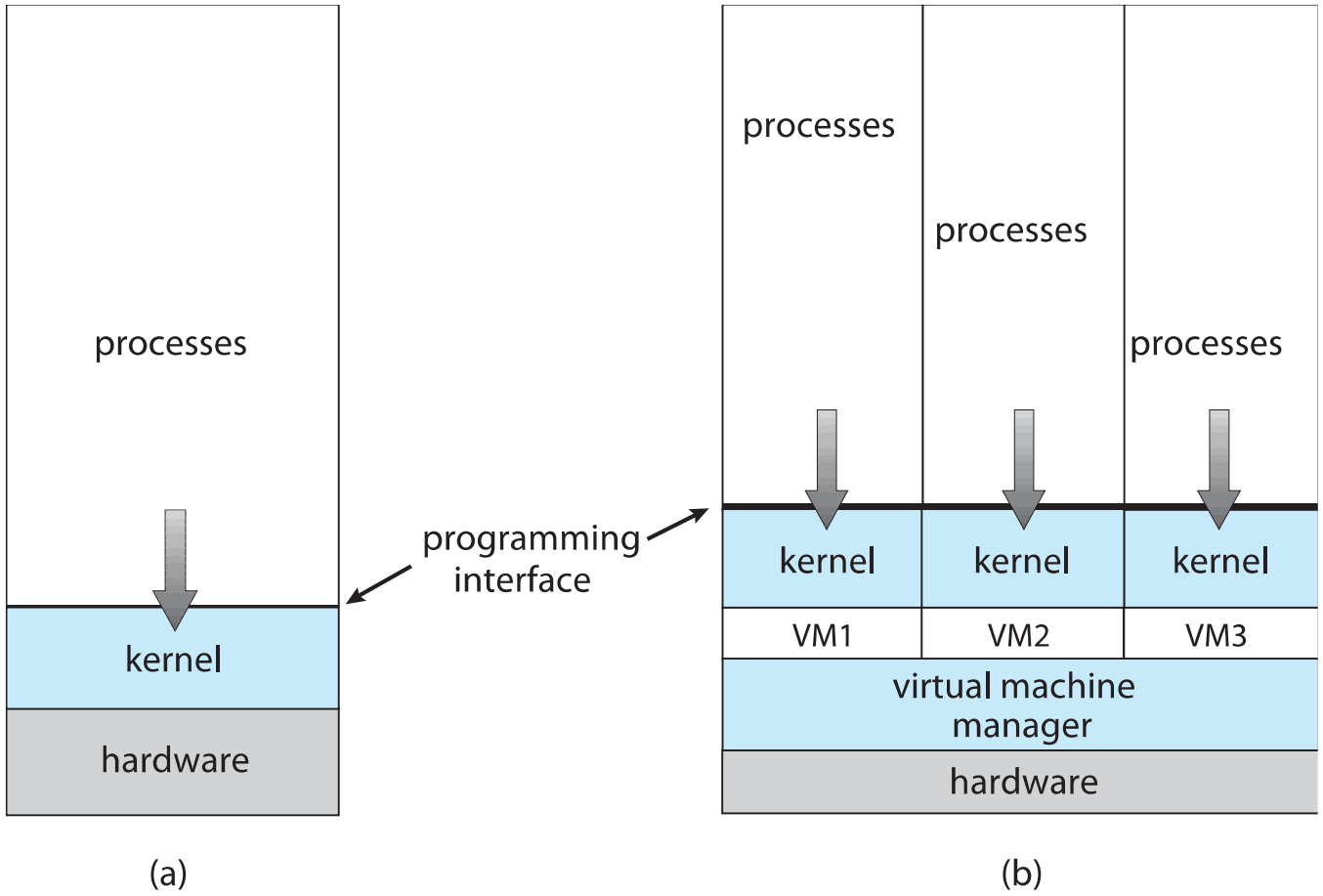
# VIRTUALIZATION

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used when source CPU type different from target type (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

# VIRTUALIZATION (CONT.)

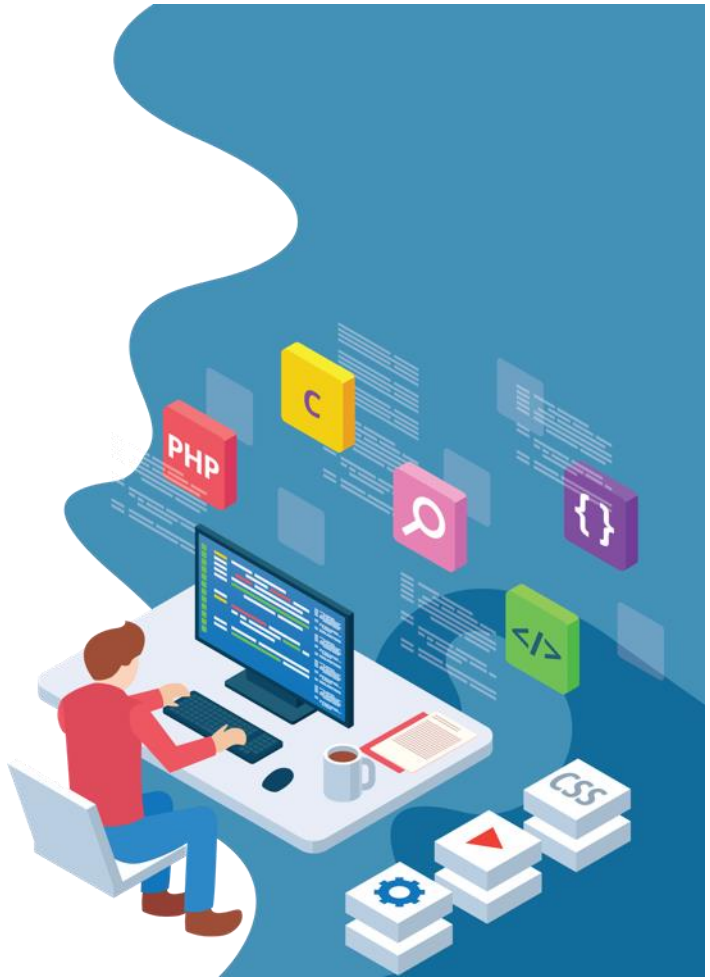
- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - Quality assurance testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general-purpose host then (VMware ESX and Citrix XenServer)

# COMPUTING ENVIRONMENTS - VIRTUALIZATION



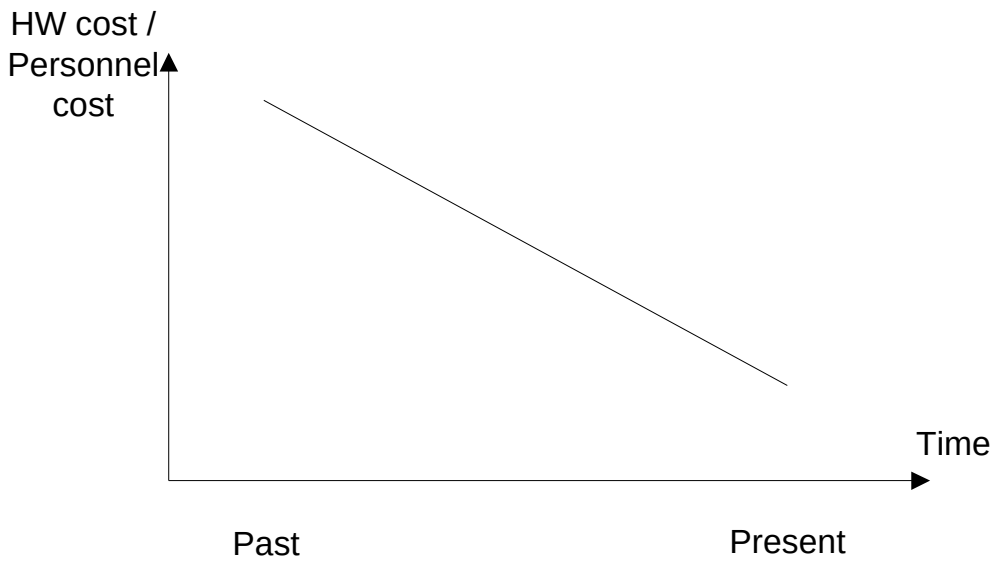
# OUTLINE

- What is an Operating System?
- Operating-System Operations
- Process Management
- Computer System Organization
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



# OPERATING SYSTEM HISTORY

- Much of operating system history (goals and thus characters) is driven by relative cost factors of **hardware** and **people**.




H/P rate over time

# EARLY DAYS - H/P VERY HIGH

- Computers were huge machines, expensive, single user, use punch card
- Goal: maximize hardware utilization
- **Problem**: Code to manipulate external I/O devices. Is very complex, and is a major source of programming difficulty
- **Solution**: Build a subroutine library (device drivers) to manage the interaction with the I/O devices. The library is loaded into the top of memory and stays there. This is the first example of something that would grow into an operating system

Eniac Computer - 1946

# EARLY DAYS - H/P VERY HIGH

Because the machine is so expensive, it is important to keep it busy.

- **Problem**: computer idles while programmer sets things up. Poor utilization of huge investment.
- Solution: Hire a specialized person to do setup. Faster than programmer, but still a lot slower than the machine. => **batch system**
- **Problem**: At any given time, job is actively using either the CPU or an I/O device, and the rest of the machine is idle and therefore unutilized.
- Solution: Allow the job to overlap computation and I/O. Buffering and interrupt handling added to subroutine library
- **Problem**: one job can't keep both CPU and I/O devices busy. Get poor utilization either of CPU or I/O devices.
- Solution: *multiprogramming* - several jobs share system. Dynamically switch from one job to another when the running job does I/O.

# PHASE SHIFT - H/P DECREASING

- Computers become much cheaper. People costs become significant.
- Goals: make computers easier to use and to improve the productivity of the people. Many people use computer interactively - interactive timesharing.
- Problems: people need reasonable response time from the computer; people need to have their data and programs around; the boss logs in and gets terrible response time.
- Solutions: preemptive scheduling; add file systems for quick access to data; prioritized scheduling.

# H/P EVEN LOWER (1970S)

- Goals: giving one computer to each user.
- Problems: Initial cost is very important in market - Minimal hardware. OS resource consumption becomes a big issue (640K of memory)

# H/P LOW

- Hardware becomes cheaper, stronger and users more sophisticated
- Goals: Allow people to share resource, data and information with other people.
- Problem: Networking, security becomes very important
- Solution: Operating systems become more sophisticated. Start putting back features present in the old time sharing systems

# RISE OF INTERNET

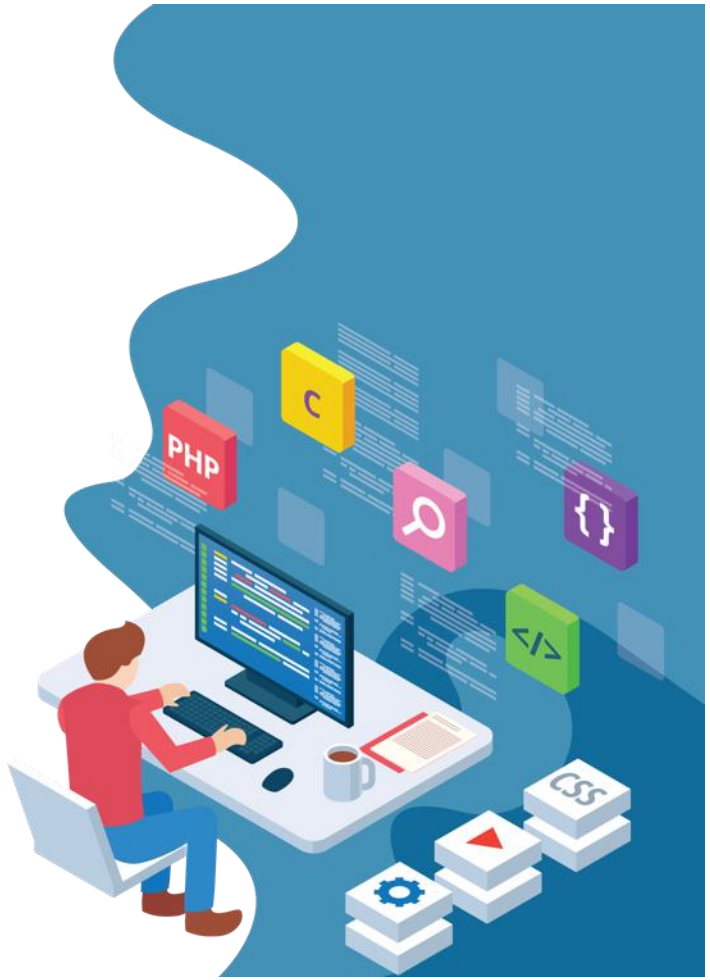
- Internet is a huge popular phenomenon and drives new ways of thinking about computing.
- Goals: Allow people to work with internet easier.
- Problems: interact with internet and security
- Solutions:
  - Java API
  - Network computer

# PORTABLE SYSTEM

- Computers become cheap, physically small and portable.
- Goals: Run usual applications on portable devices
- Problems: weak hardware (CPU, mem), small screen, pseudo-real time information like voice and video
- Solutions: Operating systems will have to adjust to deliver acceptable performance.

# OUTLINE

- What is an Operating System?
- Operating-System Operations
- Process Management
- Computer System Organization
- Resource Management
- Security and protection
- Virtualization
- Operating System History
- Classification of Operating System



# LIST OF POPULAR OPERATING SYSTEMS

- Here are the list of some popular windows based personal computer operating systems:
  - Windows XP
  - Windows 7
  - Windows 8
  - Windows 8.1
  - Windows Vista
  - Windows 10
  - Windows 10
  - and many more
- And the following are the list of some popular linux based operating system, mostly used by advance computer user:
  - Ubuntu
  - Backtrack
  - Kali Linux
  - Fedora
  - Linux Mint
  - and many more

# CLASSIFICATION OF OPERATING SYSTEM

- **Single-User:** just allows one user to use the programs at one time
- **Multi-user:** is the one that concede two or more users to use their programs at the same time. Some of O.S permits hundreds or even thousands of users simultaneously
- **Single-tasking:** Allows different parts of a single program running at any one time
- **Multi-tasking:** Allows multiple programs running at the same time
  - **Multi-processor:** Supports opening the same program more than just in one CPU
- Real-Time Systems

# CLASSIFICATION OF OPERATING SYSTEM

- MainFrame
- Server
- Multi CPU
- PC
- PDA (Embedded OS)
- SmartCard

# MAINFRAME OPERATING SYSTEMS

- These computers distinguish themselves from personal computers in terms of their I/O capacity. A mainframe with 1000 disks and thousands of gigabytes of data.
- The operating systems for mainframes are heavily oriented toward processing many jobs at once.
- They typically offer three kinds of services: batch, transaction processing, and timesharing:
  - A batch system is one that processes routine jobs without any interactive user, for example, sales reporting for a chain of stores is typically done in batch mode.
  - Transaction processing systems handle large numbers of small requests, for example, check processing at a bank or airline reservations. Each unit of work is small, but the system must handle hundreds or thousands per second.
  - Timesharing systems allow multiple remote users to run jobs on the computer at once, such as querying a big database.

# SERVER OPERATING SYSTEMS

- They run on servers, which are either very large personal computers, workstations, or even mainframes.
- They serve multiple users at once over a network and allow the users to share hardware and software resources.
- Servers can provide print service, file service, or Web service. Internet providers run many server machines to support their customers to store the Web sites and handle the incoming requests.
- Typical server operating systems are UNIX and Windows 2000.

# MULTIPROCESSOR OPERATING SYSTEMS

- Multiple CPUs into a single system.
- These systems are called parallel computers, multicomputers, or multiprocessors depending on how they are connected and what is shared.
- They need special operating systems, but often these are variations on the server operating systems, with special features for communication and connectivity.

# PERSONAL COMPUTER OPERATING SYSTEMS

- The next category is the personal computer operating system.
- Their job is to provide a good interface to a single user. They are widely used for word processing, spreadsheets, and Internet access.
- Common examples are Windows 98, Windows 2000, the Macintosh operating system, and Linux.
- Personal computer operating systems are so widely known that probably little introduction is needed. In fact, many people are not even aware that other kinds exist.

# REAL-TIME OPERATING SYSTEMS

- Time is key parameter in this system. For example, in industrial process control systems, real-time computers have to collect data about the production process and use it to control machines in the factory.
- Often there are hard deadlines .If the action absolutely *must* occur at a certain moment (or within a certain range), we have a **hard real-time system. A soft real-time** system, in which missing an occasional deadline is acceptable.

- VxWorks and QNX are well-known real-time operating systems.

# EMBEDDED OPERATING SYSTEMS

- **PDA (Personal Digital Assistant)** is a small computer that fits in a shirt pocket and performs a small number of functions .
- Embedded systems run on the computers that control devices that are not generally thought of as computers, such as TV sets, microwave ovens, and mobile telephones.
- These often have some characteristics of real-time systems but also have size, memory, and power restrictions that make them special.
- Examples of such old general operating systems are PalmOS and Windows CE (Consumer Electronics).

# 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***
- Cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

# SMART CARD OPERATING SYSTEMS

- The smallest operating systems run on smart cards, which are credit card-sized devices containing a CPU chip.
- They have very severe processing power and memory constraints. Some of them can handle only a single function, such as electronic payments, but others can handle multiple functions on the same smart card.
- Some smart cards are Java oriented and holds an interpreter for the Java Virtual Machine (JVM).