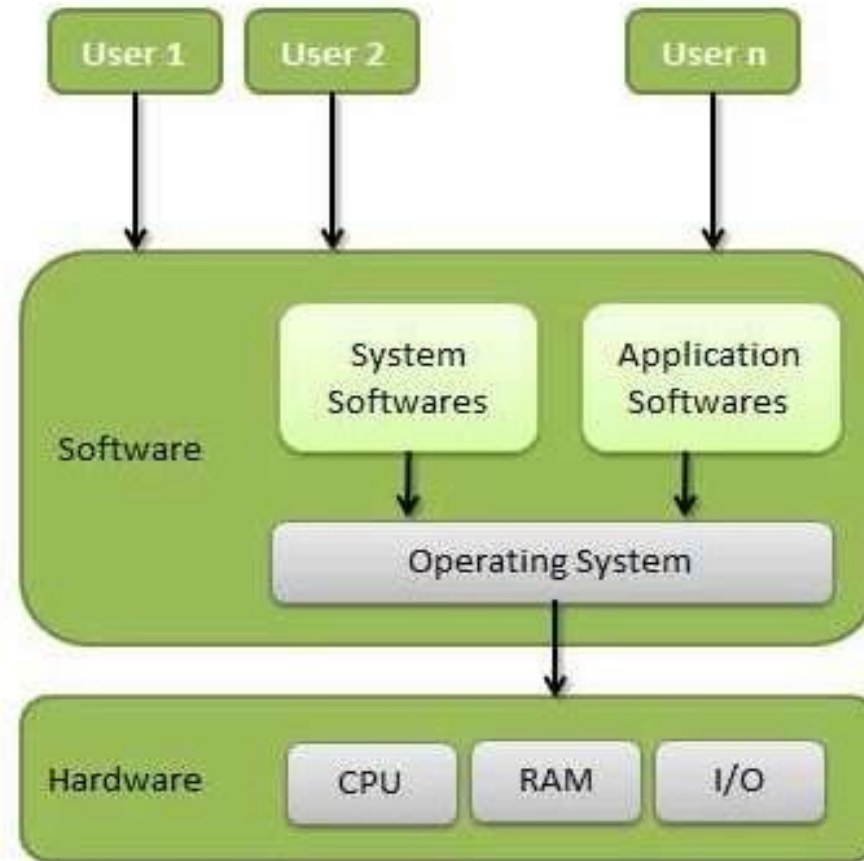


Module-1

Introduction to operating systems, Operating System Services

What is an Operating System?

An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- It is also called as resource allocator
- **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



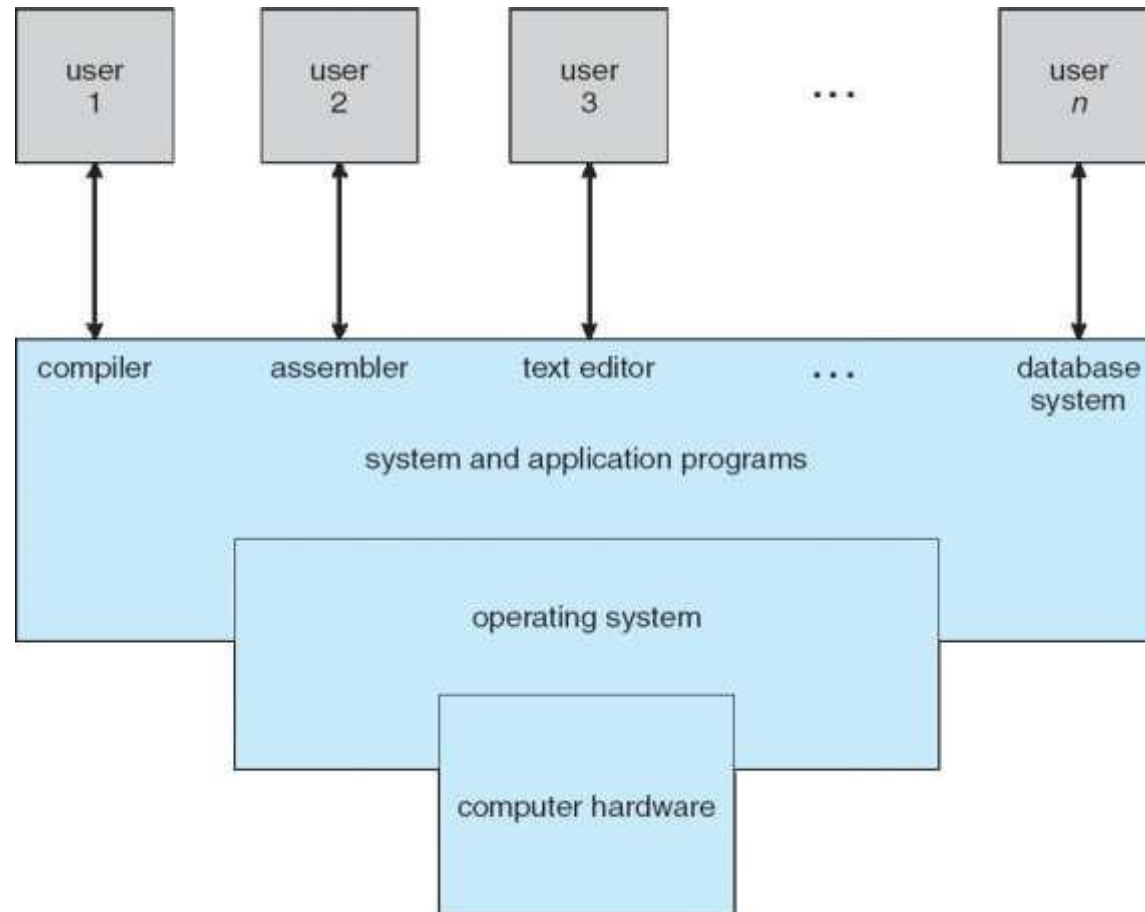
Functions of Operating System

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

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

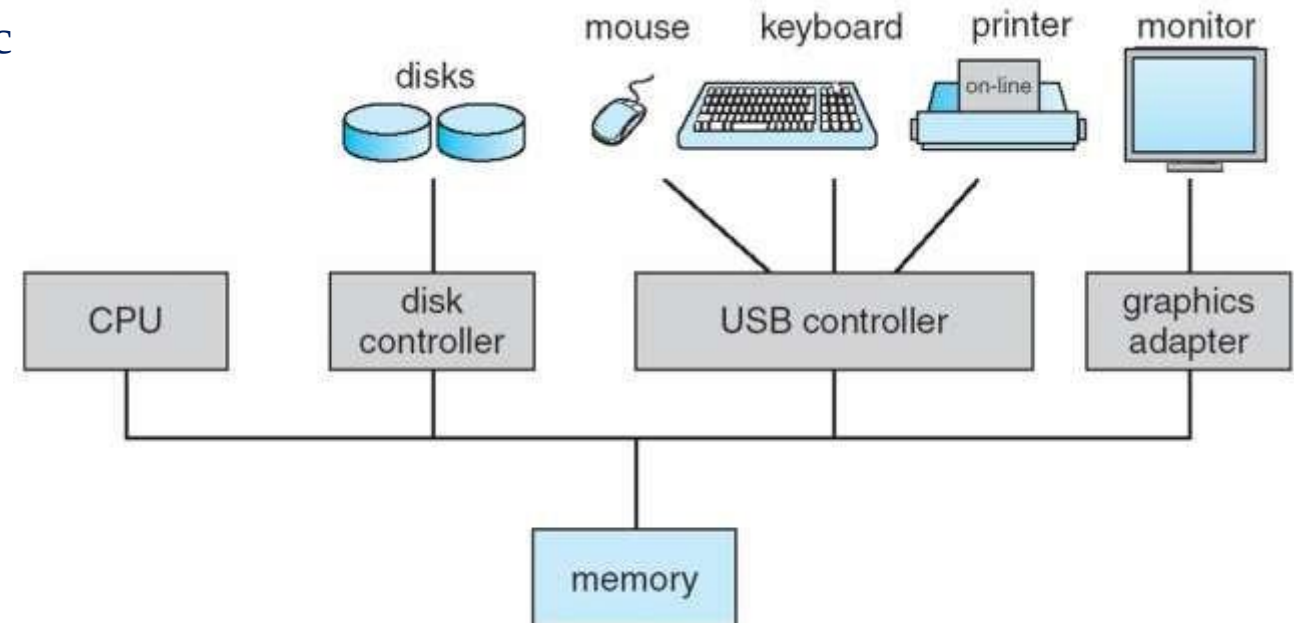


Operating System Definition

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

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 c



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.
- 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*.

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* is a software-generated interrupt caused either by an error or a user request.
- An operating system is **interrupt driven**.

I/O Structure

- 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 operating system 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.
 - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

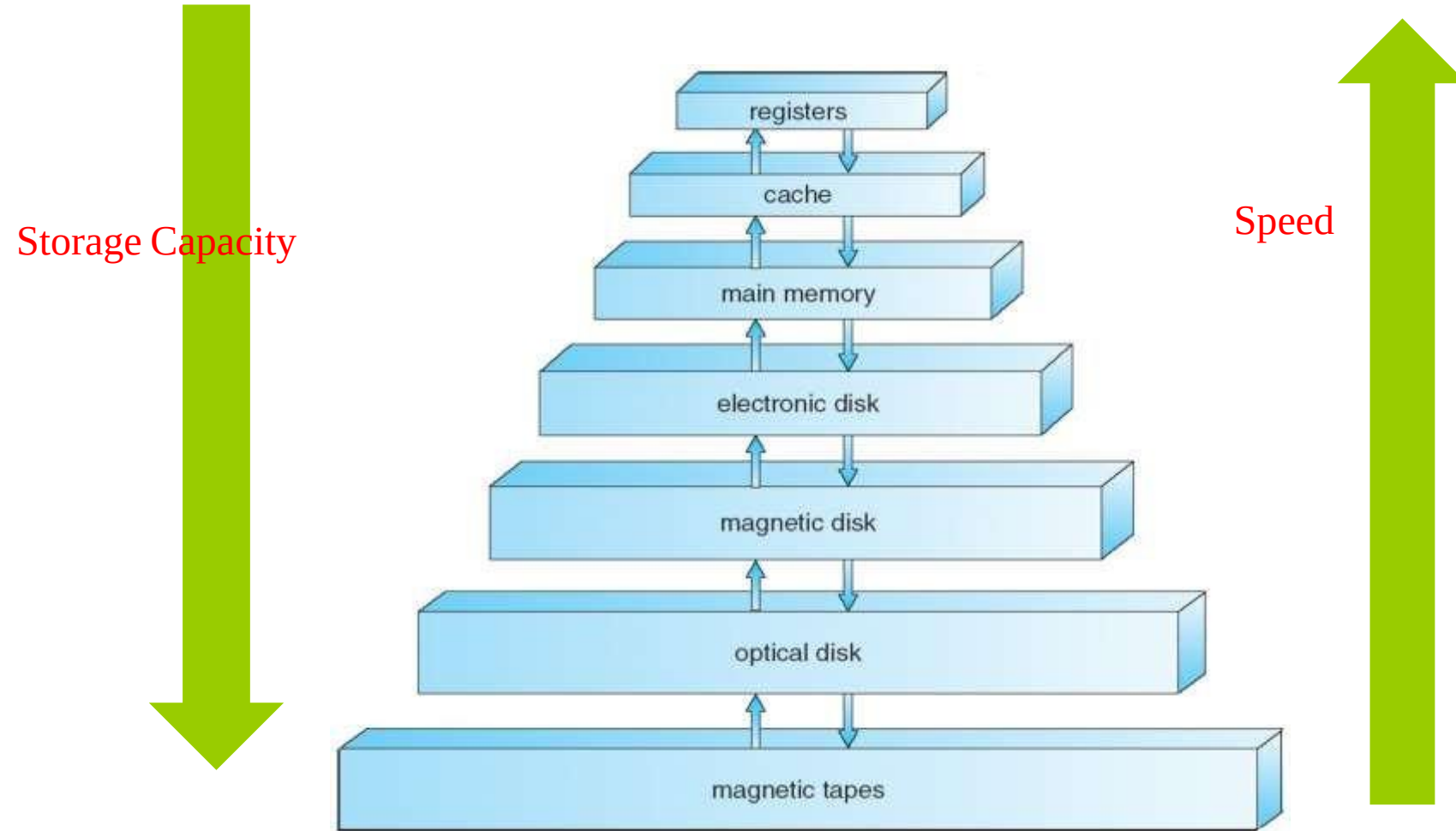
Direct Memory Access Structure

- Direct memory access is a feature of computer systems that allows certain hardware subsystems to access main system memory, independent of the central processing unit
- 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.

Storage Structure

- **Main memory** – only large storage media that the CPU can access directly.
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity.
- **Magnetic 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.

Storage-Device Hierarchy



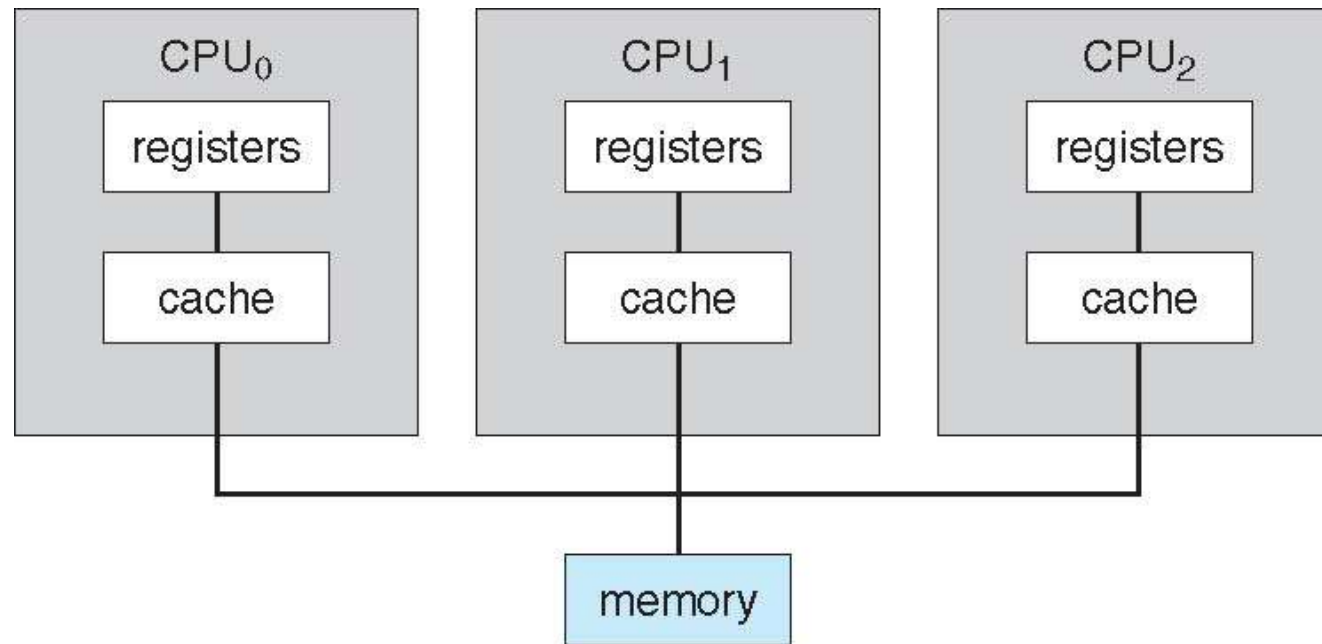
Computer-System Architecture

- Most systems use a single general-purpose processor (PDAs through mainframes).
 - 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
 - **Increased throughput**
 - **Economy of scale**
 - **Increased reliability – graceful degradation or fault tolerance**
 - Two types
 - **Asymmetric Multiprocessing**
 - **Symmetric Multiprocessing**

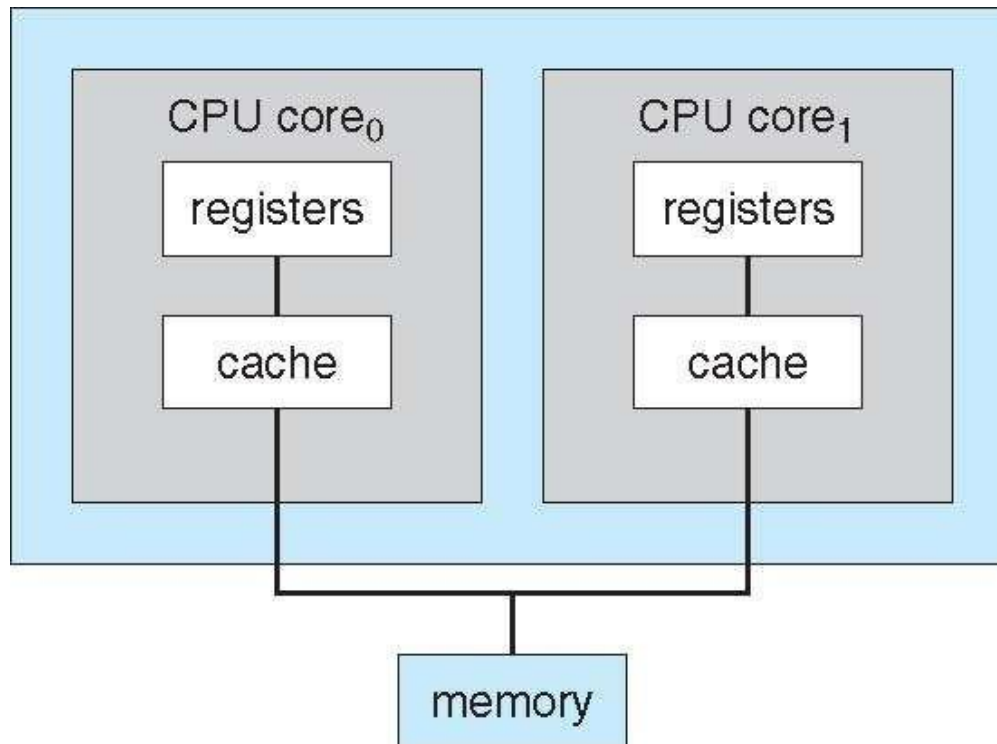
What is Multiprocessing

- The running of two or more programs or sequences of instructions simultaneously by a computer with more than one central processor.

Symmetric Multiprocessing Architecture



A Dual-Core Design

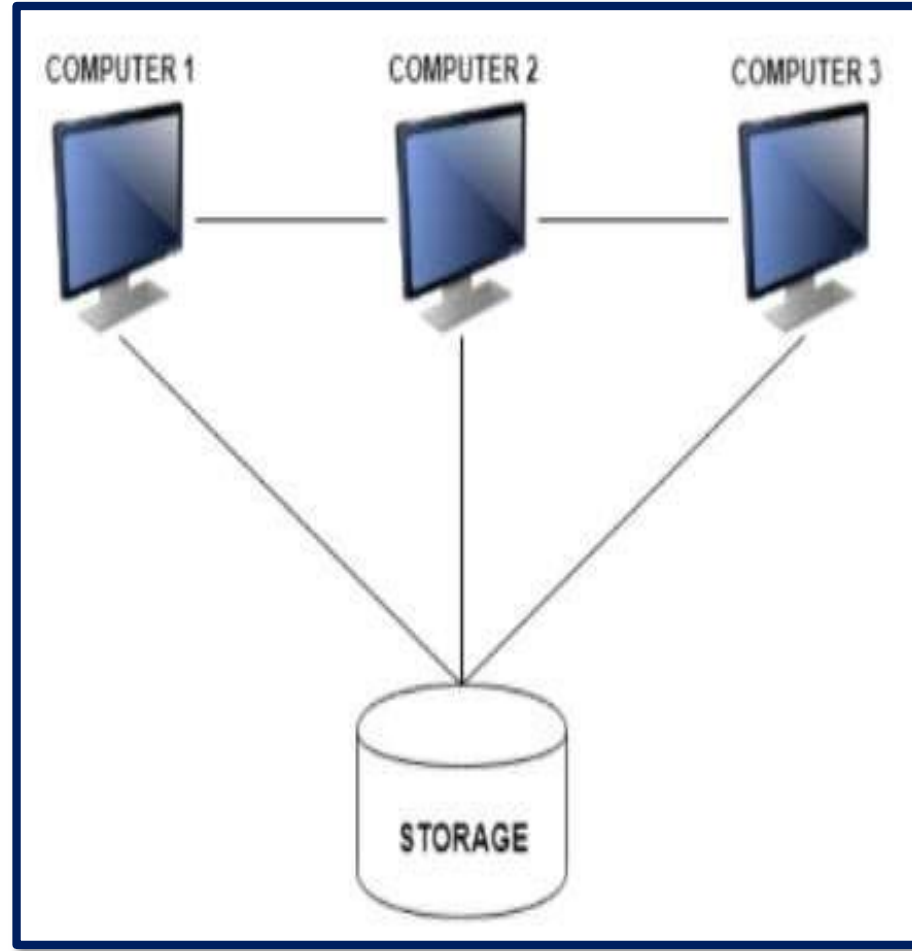


Types of Computing

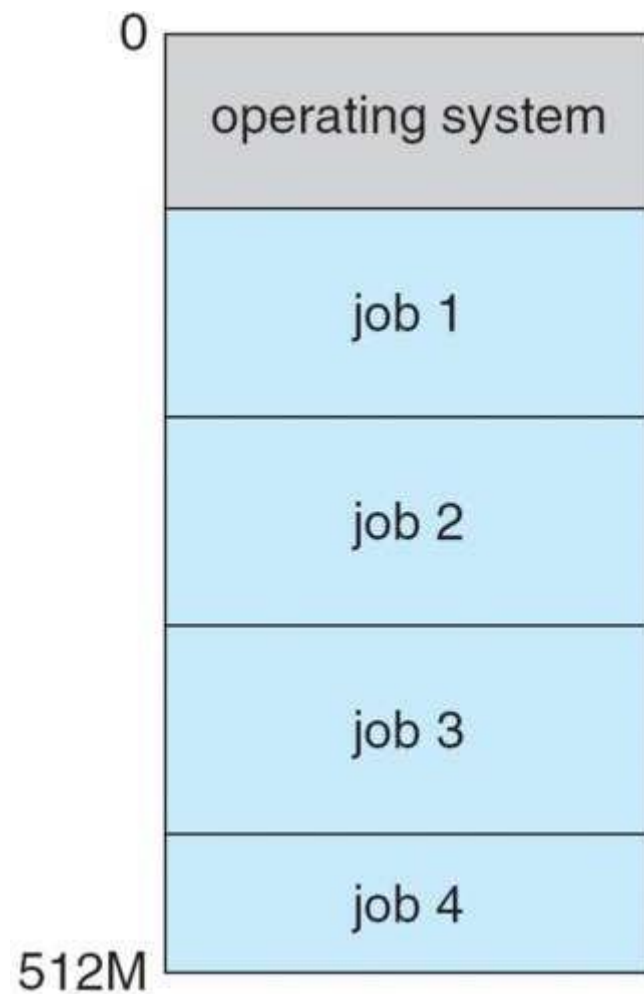
- **Multiprogramming:** The concurrent residency of more than one program in the main memory is referred as multiprogramming.
- **Multiprocessing:** When one system is connected to more than one processor which collectively work for the completion of the task, it is called as multiprocessing systems.
- **Multitasking:** availability of high-speed processor, one can submit more than one task.
- **Multithreading:** A process is divided into several different sub-processes called as threads, which has its own path of execution. This concept is called as multithreading.

What are clustered Systems ?

- Clustered systems are similar to parallel systems as they both have multiple CPUs. However a major difference is that clustered systems are created by two or more individual computer systems merged together. Basically, they have independent computer systems with a common storage and the systems work together.



Memory Layout for Multiprogrammed



Structure of Operating Systems

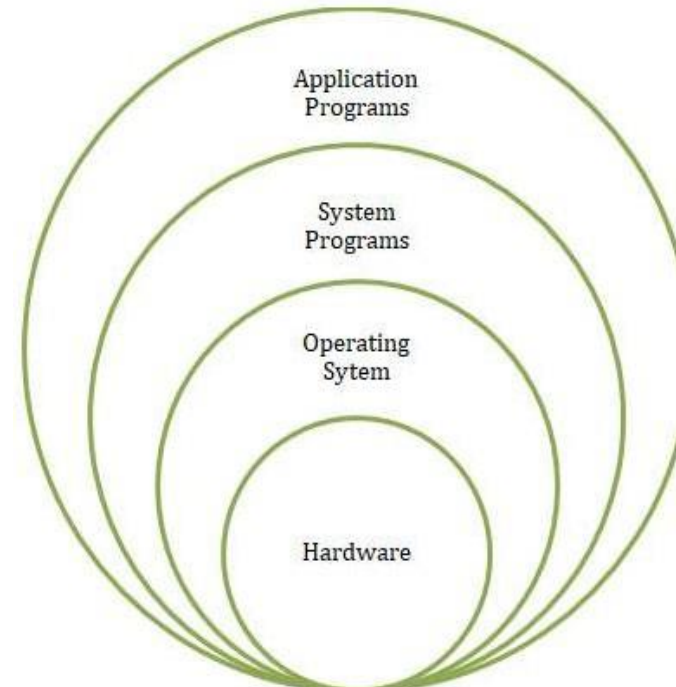
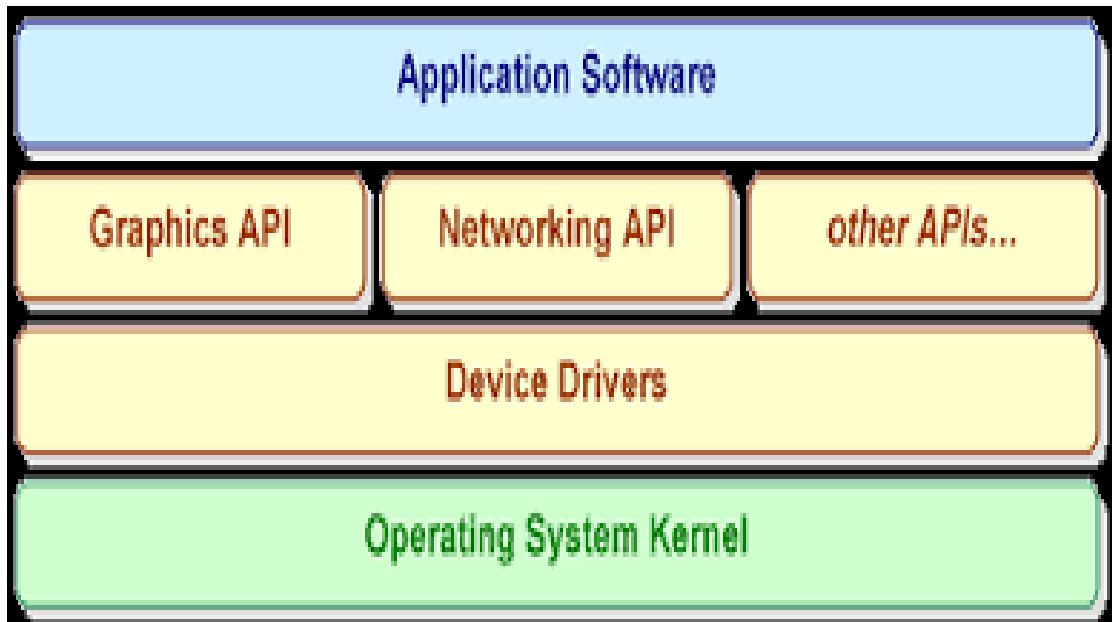


Fig: Structure of an Operating System

Modes in Operating Systems

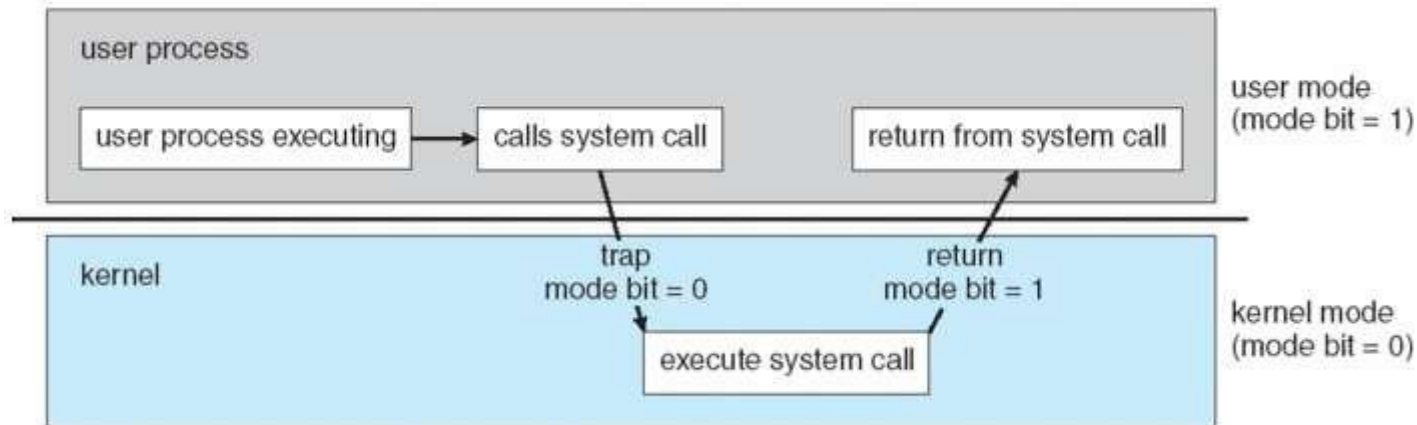
- **User Mode:** In this type user has limited access to system resources , it functions on predefined set of system calls
- **Examples:** File access
- **Kernel mode :** It is also known as supervisor mode or privileged mode, directly accessing system hardware and settings and critical tasks. It is un restricted access.
- **Examples:** Memory Management, Device drivers management, system configuration, installing new software, process scheduling

Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
 - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **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
 - Some instructions designated as **privileged**, only executable in kernel mode
 - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
 - Set interrupt after specific period
 - Operating system decrements counter
 - When counter zero generate an interrupt
 - Set up before scheduling process to regain control or terminate program that exceeds allotted time

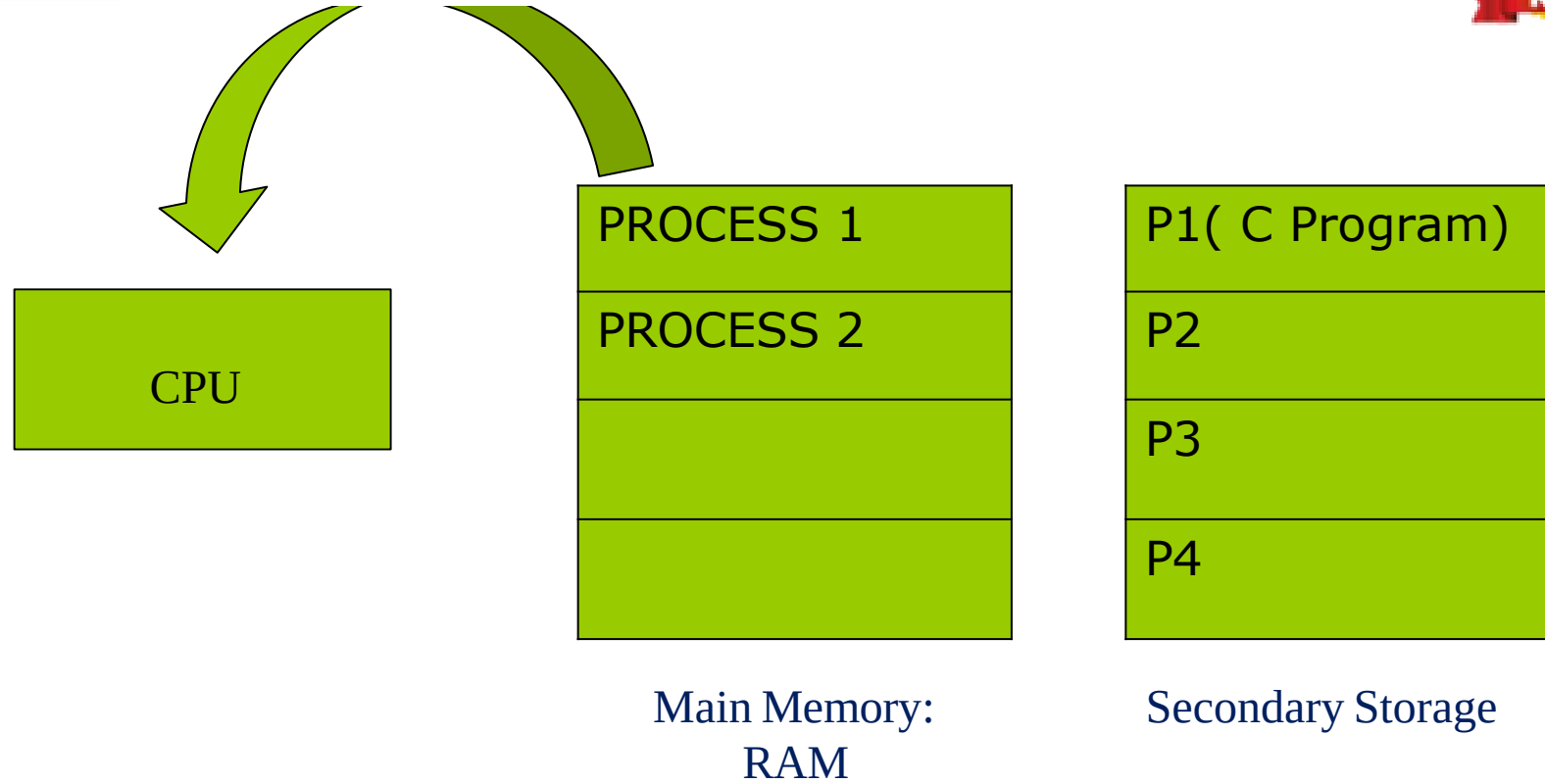


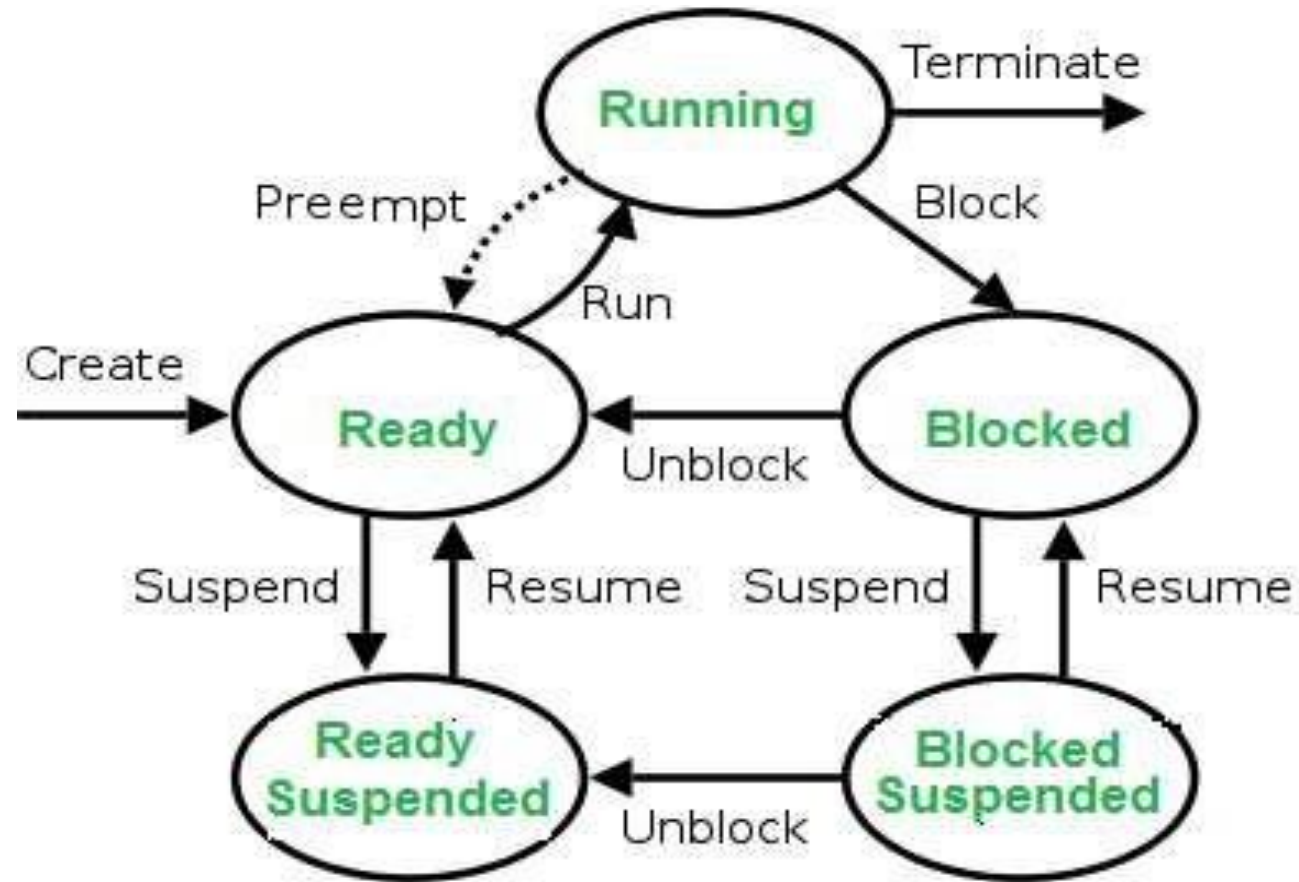
Process Management

- 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.

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







Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory 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

Storage 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 dirs
 - 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:
 - Free-space management
 - Storage allocation
 - Disk scheduling
- Some storage need not be fast
 - Tertiary storage includes optical storage, magnetic tape
 - Still must be managed
 - Varies between WORM (write-once, read-many-times) and RW (read-write)

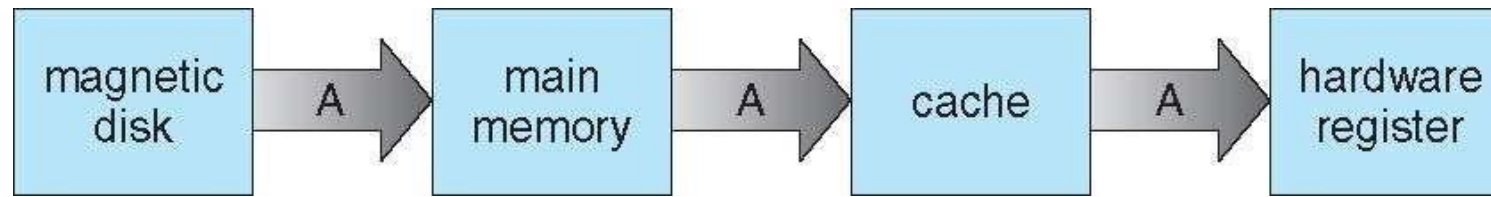
Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Migration of Integer 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 17

I/O Subsystem

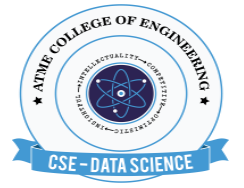
- One purpose of OS is to hide peculiarities of hardware devices from the user.
- I/O subsystem responsible for:
 - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance) (the overlapping 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



ISO 9001:2015



Computing Environments

- Traditional computer
 - Blurring over time
 - Office environment
 - PCs connected to a network, terminals attached to mainframe or minicomputers providing batch and timesharing
 - Now portals allowing networked and remote systems access to same resources
 - Home networks
 - Used to be single system, then modems
 - Now firewalled, networked



Peer-to-Peer Computing

- 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*

Web-Based Computing

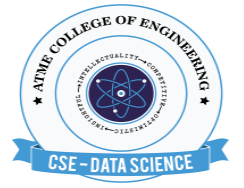
- Web has become ubiquitous
- PCs most prevalent devices
- More devices becoming networked to allow web access
- New category of devices to manage web traffic among similar servers: **load balancers**
- Use of operating systems like Windows 95, client-side, have evolved into Linux and Windows XP, which can be clients and servers



A T M E
College of Engineering



ISO 9001:2015



Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has “copyleft” **GNU Public License (GPL)**
- Examples include **GNU/Linux**, **BSD UNIX** (including core of **Mac OS X**), and **Sun Solaris**

Operating System Services

- One set of operating-system services provides functions that are helpful to the user:
 - **User interface** - Almost all operating systems have a user interface (UI)
 - Varies between **Command-Line (CLI)**, **Graphics User Interface (GUI)**, **Batch**
 - **Program execution** - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
 - **I/O operations** - A running program may require I/O, which may involve a file or an I/O device
 - **File-system manipulation** - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

User Operating System Interface - CLI

- Command Line Interface (CLI) or command interpreter allows direct command entry
- Sometimes implemented in kernel, sometimes by systems program
- Sometimes multiple flavors implemented – shells
- Primarily fetches a command from user and executes it
- Sometimes commands built-in, sometimes just names of programs

What is a System call ?

- In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on.
- This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling.
- System calls provide an essential interface between a process and the operating system.

System Call Implementation

- Typically, a number associated with each system call
 - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
 - Just needs to obey API and understand what OS will do as a result call
 - Most details of OS interface hidden from programmer by API
 - Managed by run-time support library (set of functions built into libraries included with compiler)

System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
 - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
 - **Simplest:** pass the parameters in *registers*
 - In some cases, may be more parameters than registers
 - **Parameters stored in a *block***, or table, in memory, and address of block passed as a parameter in a register
 - This approach taken by Linux and Solaris
 - **Parameters placed, or *pushed*, onto the *stack*** by the program and *popped* off the stack by the operating system
 - Block and stack methods do not limit the number or length of parameters being passed

Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection



System Programs

- System programs provide a convenient environment for program development and execution. They can be divided into:
 - File manipulation
 - Status information
 - File modification
 - Programming language support
 - Program loading and execution
 - Communications
 - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls



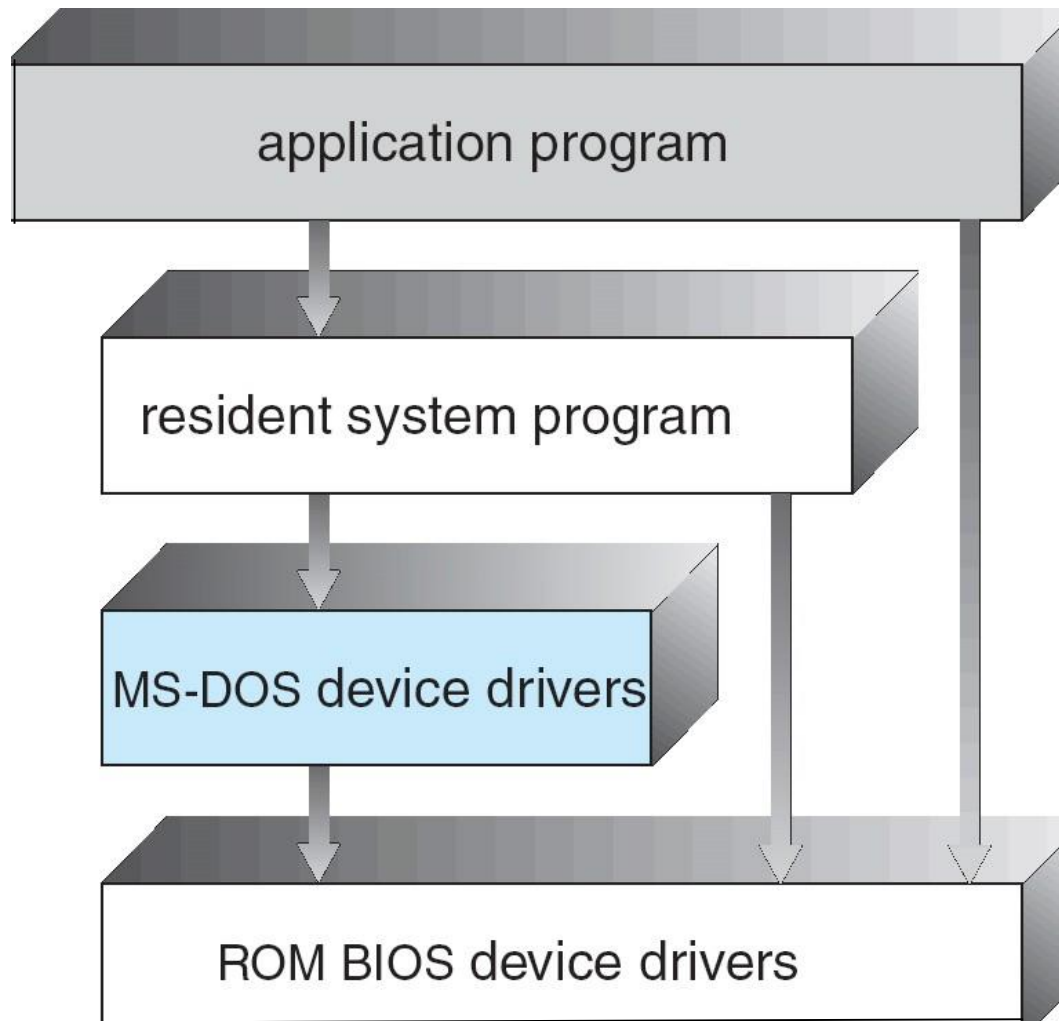
Operating System Design and Implement-----

- Design and Implementation of OS not “solvable”, but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start by defining goals and specifications
- Affected by choice of hardware, type of system
- *User goals and System goals*
 - **User goals** – operating system should be convenient to use, easy to learn, reliable, safe, and fast
 - **System goals** – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient

Simple Structure

- MS-DOS – written to provide the most functionality in the least space
 - Not divided into modules
 - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

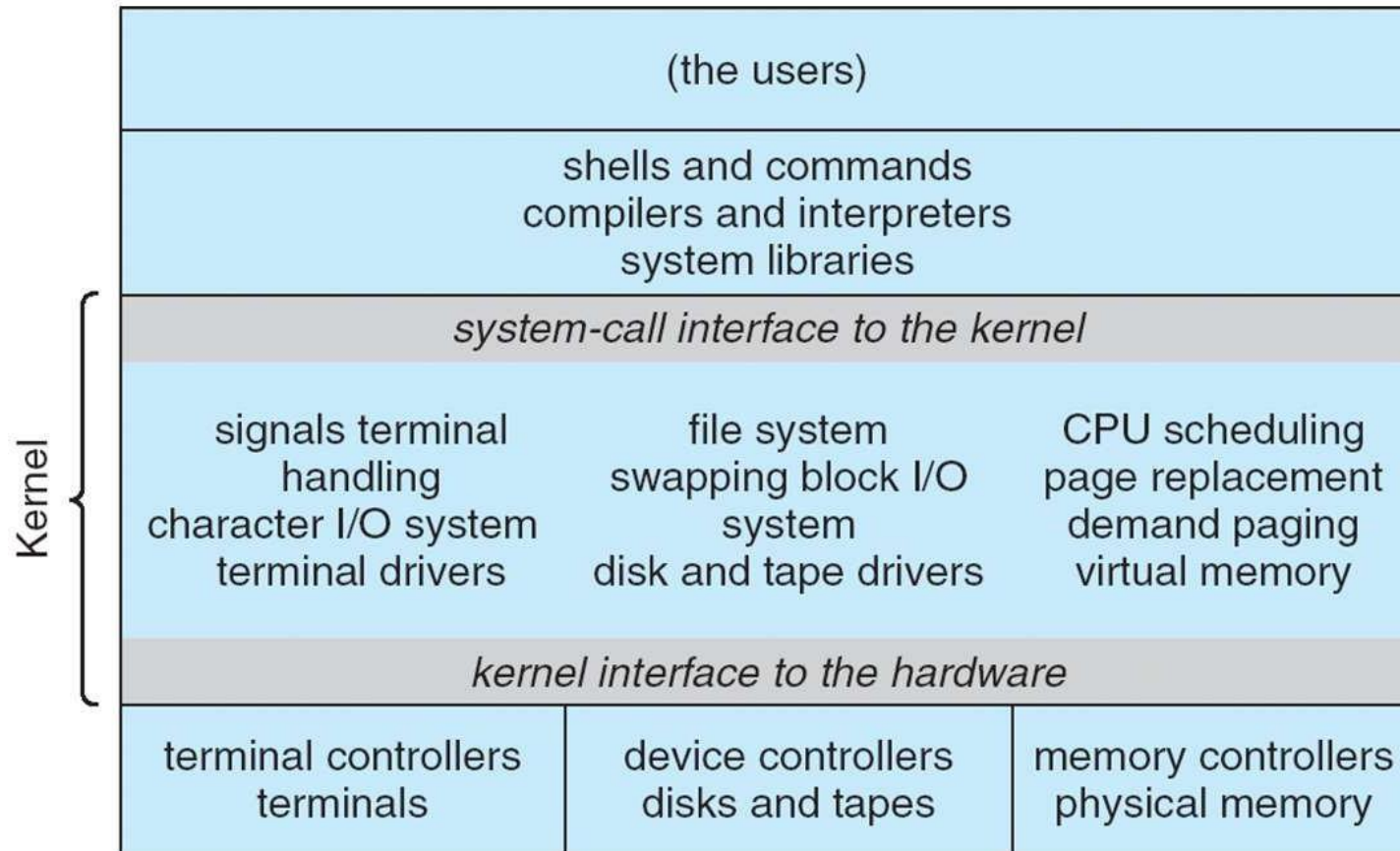
MS-DOS Layer Structure



Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

Traditional UNIX System Structure



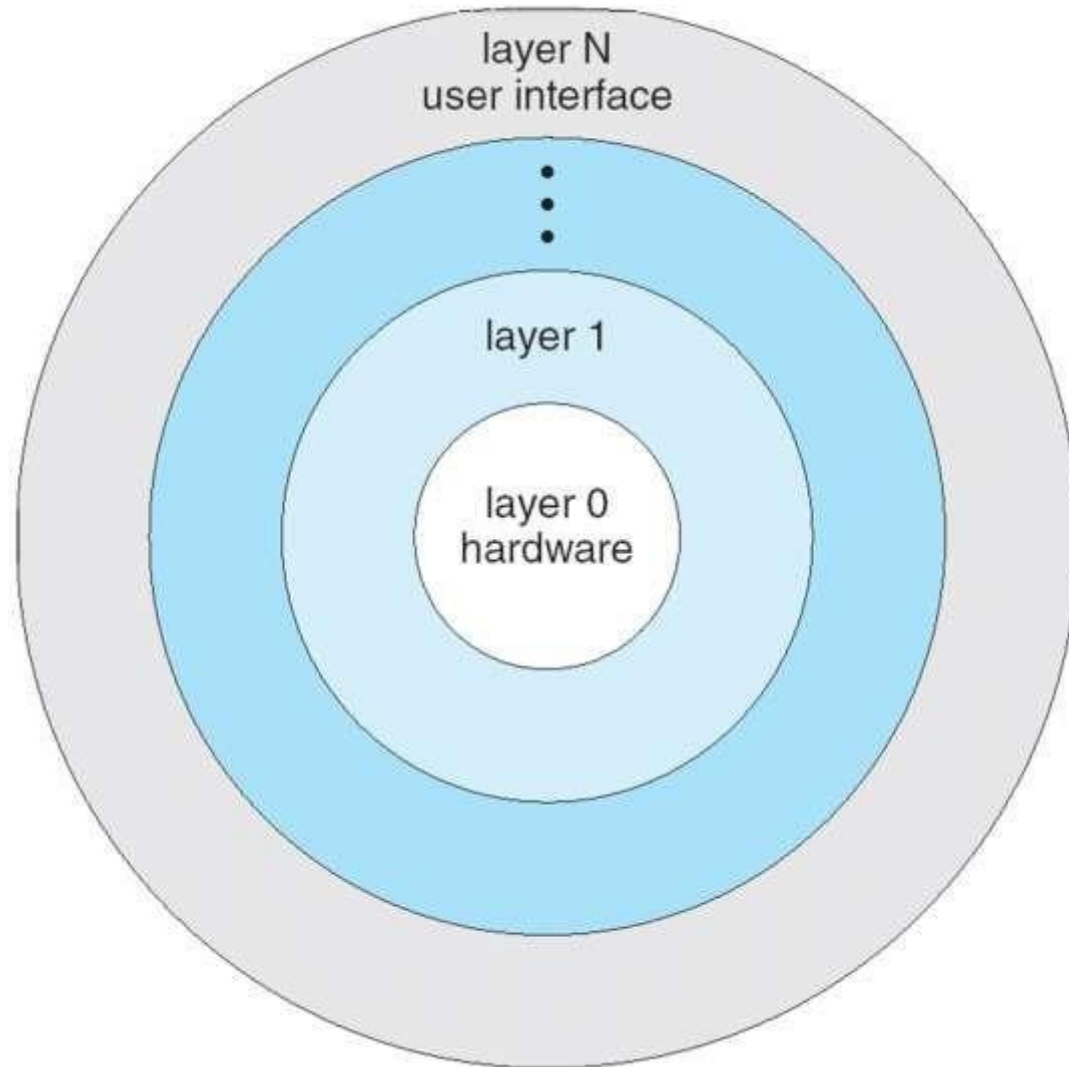
What is a Shell?

- In **Unix**, the **shell** is a program that interprets commands and acts as an intermediary between the user and the inner workings of the **operating system**. Most **shells** double as interpreted programming languages. ... To automate tasks, you may write scripts containing built-in **shell** and **Unix** commands.

UNIX

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring. The UNIX OS consists of two separable parts
 - Systems programs
 - The kernel
 - Consists of everything below the system-call interface and above the physical hardware
 - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

Layered Operating System



What is Microkernel?

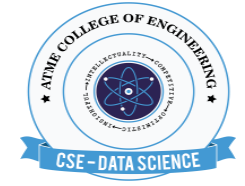
- A **microkernel** is a minimal computer **operating system** kernel which, in its purest form, provides no **operating system** services at all, only the mechanisms needed to implement such services, such as low-level address space management, thread management, and inter-process communication (IPC).

Microkernel System Structure

- Moves as much from the kernel into “user” space
- Communication takes place between user modules using message passing
- Benefits:
 - Easier to extend a microkernel
 - Easier to port the operating system to new architectures
 - More reliable (less code is running in kernel mode)
 - More secure
- Detriments:
 - Performance overhead of user space to kernel space communication

Modules

- Most modern operating systems implement kernel modules
 - Uses object-oriented approach
 - Each core component is separate
 - Each talks to the others over known interfaces
 - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible



Virtual Machines

- A **virtual machine** (or "VM") is an emulated **computer** system created using software. It uses physical system resources, such as the CPU, RAM, and disk storage, but is isolated from other software on the **computer**.
- **Example:** You can create a virtual machine using virtualization software. Examples include Microsoft Hyper-V Manager, VMware Workstation Pro, and Parallels Desktop.

What are virtual Machines ?

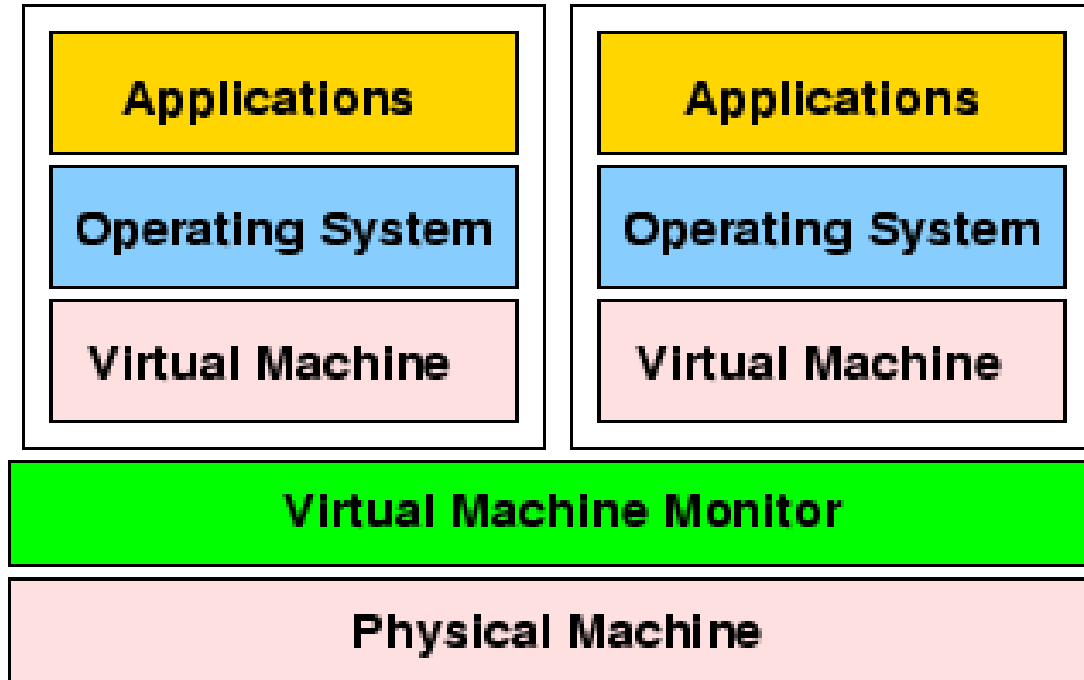
- The concept of the virtual machine was invented by IBM as a method of time-sharing extremely expensive mainframe hardware As defined by IBM, a ``virtual machine" is a fully protected and isolated copy of the underlying physical machine's hardware.
- Thus, each virtual machine user is given the illusion of having a dedicated physical machine. Software developers can also write and test programs without fear of crashing the physical machine and affecting the other users.

Two types of virtualization

- **Hardware :** In this a software layer called hypervisor is installed on the host machine , which creates multiple virtual machines each run its own operating system
- **Operating System level:** Does not require separate OS , it leverages the host os to create isolated containers provide lightweight and efficient virtualization

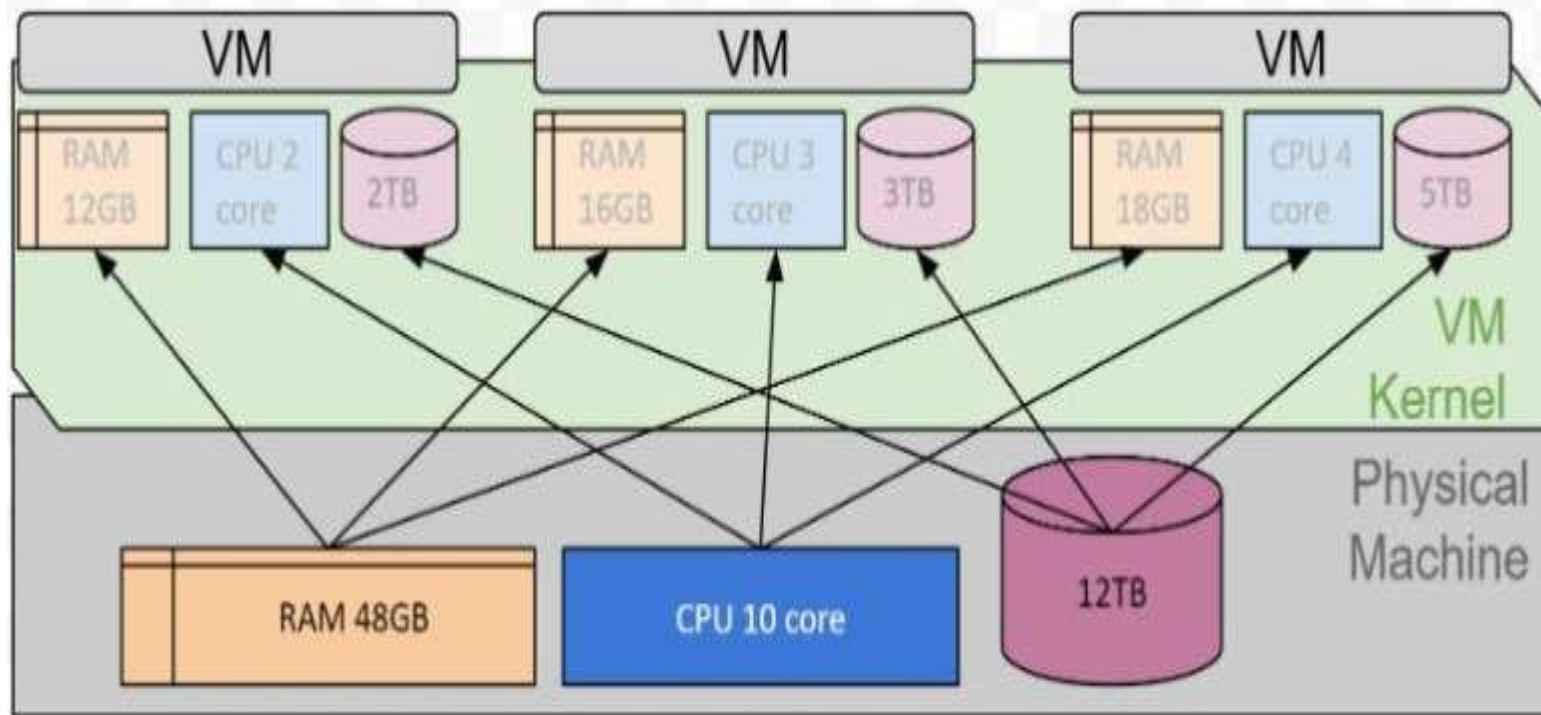


Virtual Machines

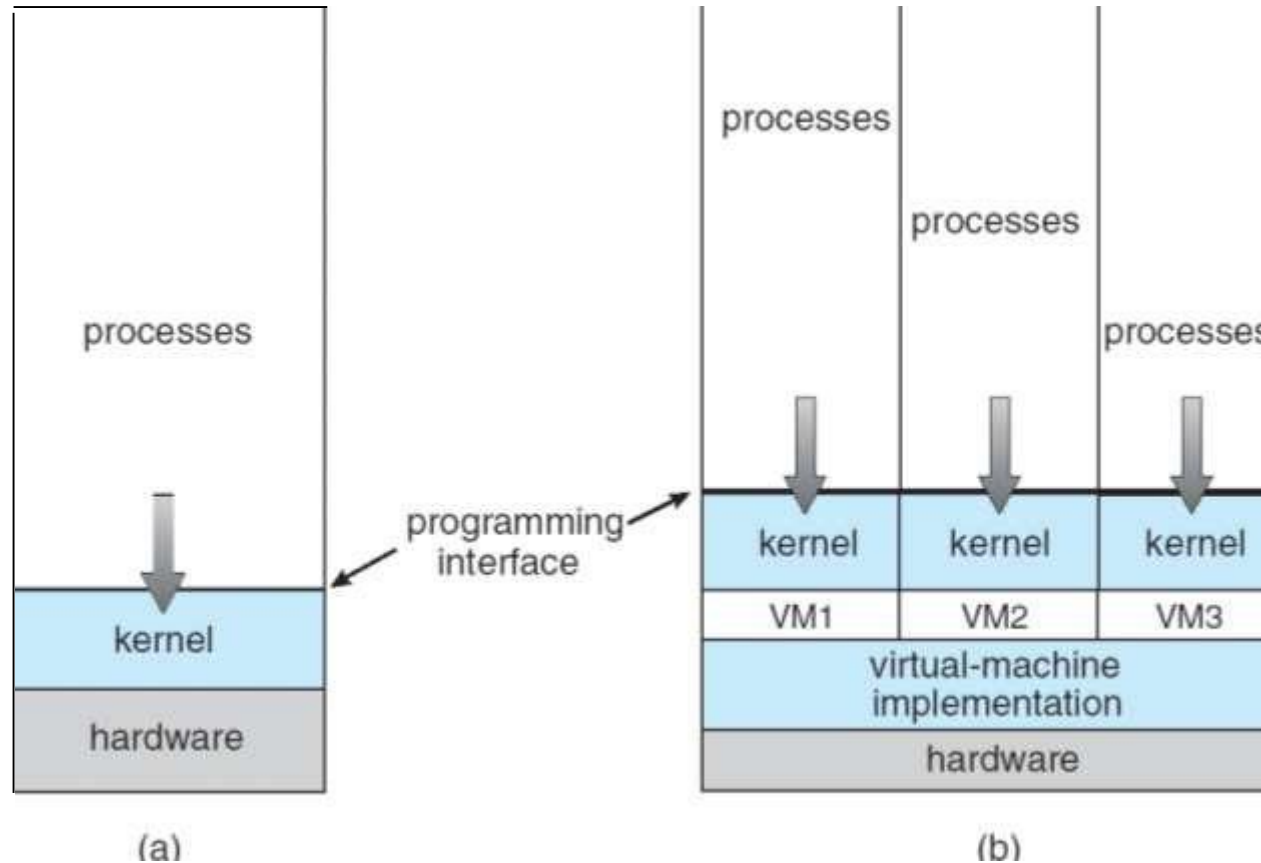


Virtual Machines

A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system. The concept of the virtual machine was invented by IBM.



Virtual Machines (Cont)



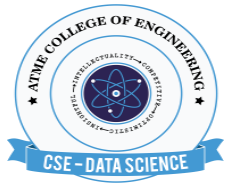
(a) Nonvirtual machine (b) virtual machine



A T M E
College of Engineering



ISO 9001:2015



Paravirtualization

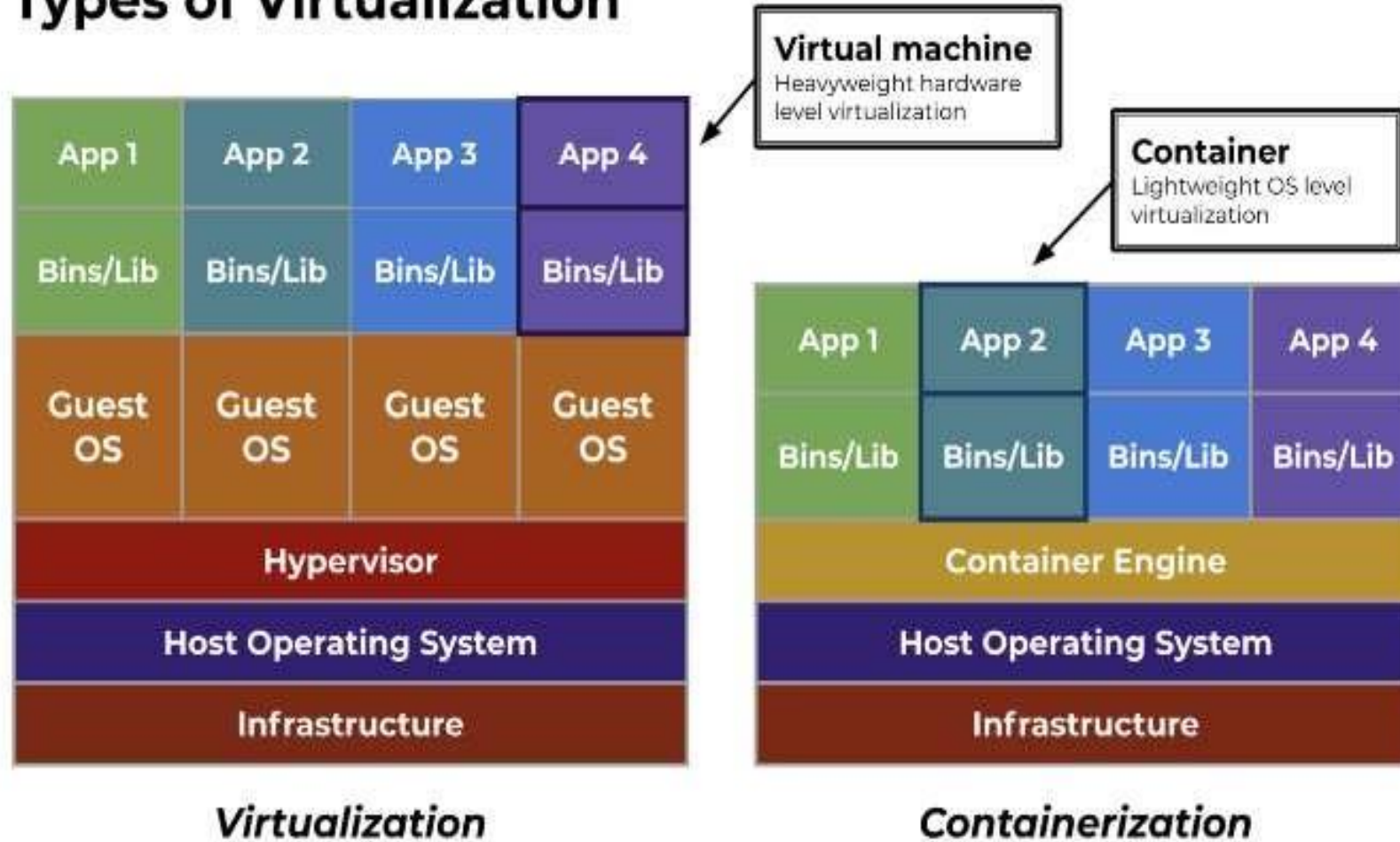
- Paravirtualization is an enhancement of virtualization technology in which a guest OS is modified prior to installation inside a virtual machine (VM) in order to allow all guest OSes within the system to share resources and successfully collaborate, rather than attempt to emulate (ಅನುಕರಿಸಿ) an entire hardware environment.



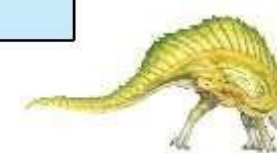
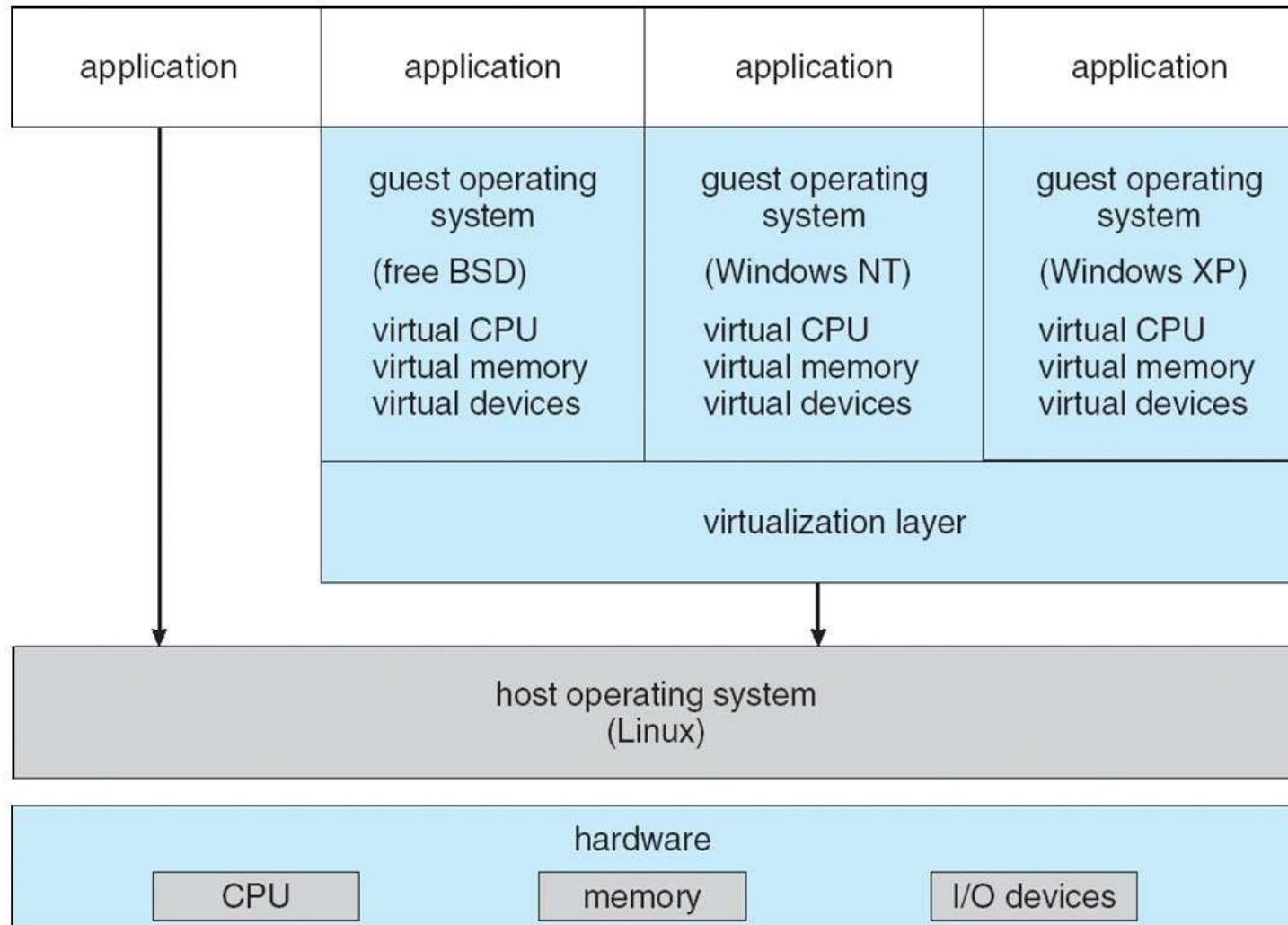
What is a container?

- Containers are based on operating system-level virtualization , where host operating systems kernel is shared among multiple containers
- Containers are **a form of operating system virtualization**. A single container might be used to run anything from a small microservice or software process to a larger application. Inside a container are all the necessary executables, binary code, libraries, and configuration files.

Types of Virtualization



VMware Architecture



What is a bug? & Debugging?

- When an error is found in a set of instructions given to a computer, it is called a bug.
- The process of finding the error in a set of computer instructions is called debugging

Debugging Process





Operating System Generation..

- Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site
- SYSGEN program obtains information concerning the specific configuration of the hardware system
- **Booting** – starting a computer by loading the kernel
- **Bootstrap program** – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution

System Boot

- Operating system must be made available to hardware so hardware can start it
 - **Small piece of code – bootstrap loader**, locates the kernel, loads it into memory, and starts it
 - **Sometimes two-step process** where **boot block** at fixed location loads bootstrap loader
 - **When power initialized on system**, execution starts at a fixed memory location
 - Firmware used to hold initial boot code