CDAC MUMBAI

COS : Day 1

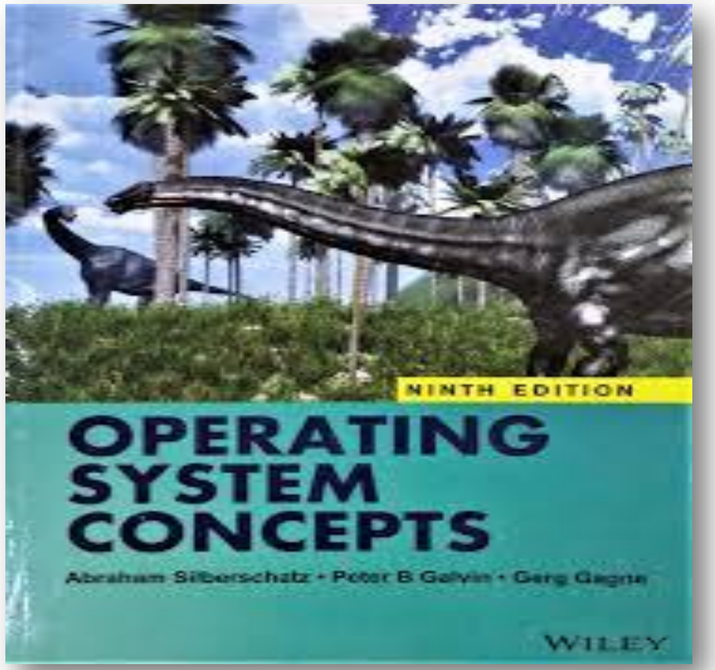
Introduction to OS

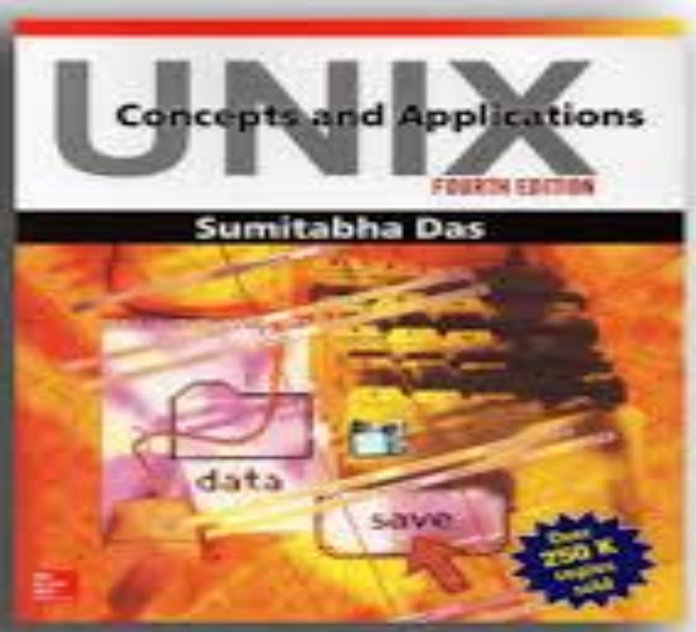
Dr Kiran Waghmare
CDAC Mumbai

Start Slide

Module : COSSDM

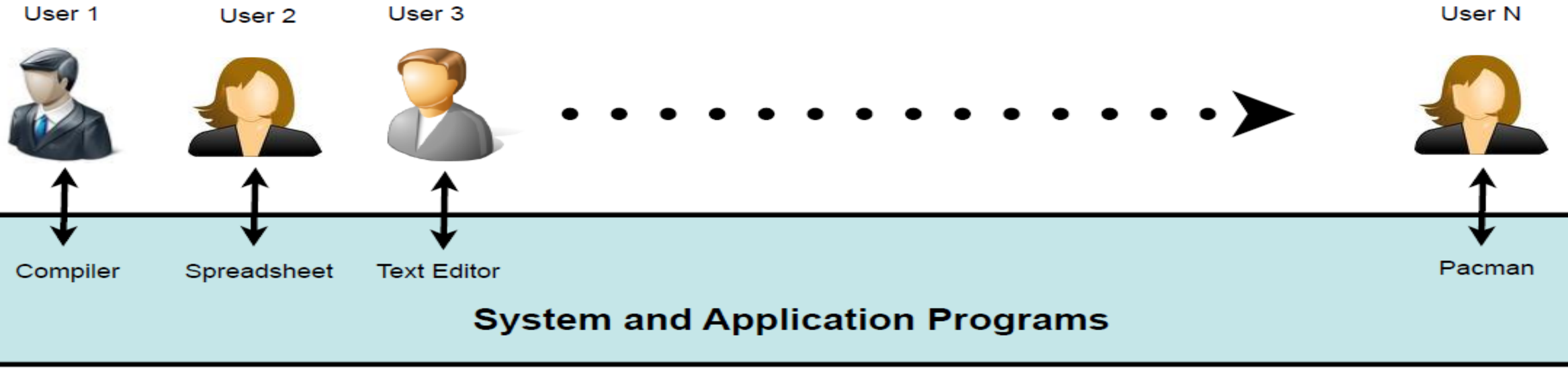
- COS + SDM
- Marks = 30 + 30
- MCQ Exam
- Lab Exam





Agenda

- **Introduction to OS**
  - OS
  - Application Software
  - Hardware dependent
  - Components of OS
  - Difference between :
    - Mobile OS, Embedded system OS,
    - Real Time OS,
    - desktop OS server machine os
  - Functions of OS
  - User and Kernel space & model
  - Interrupts & system calls

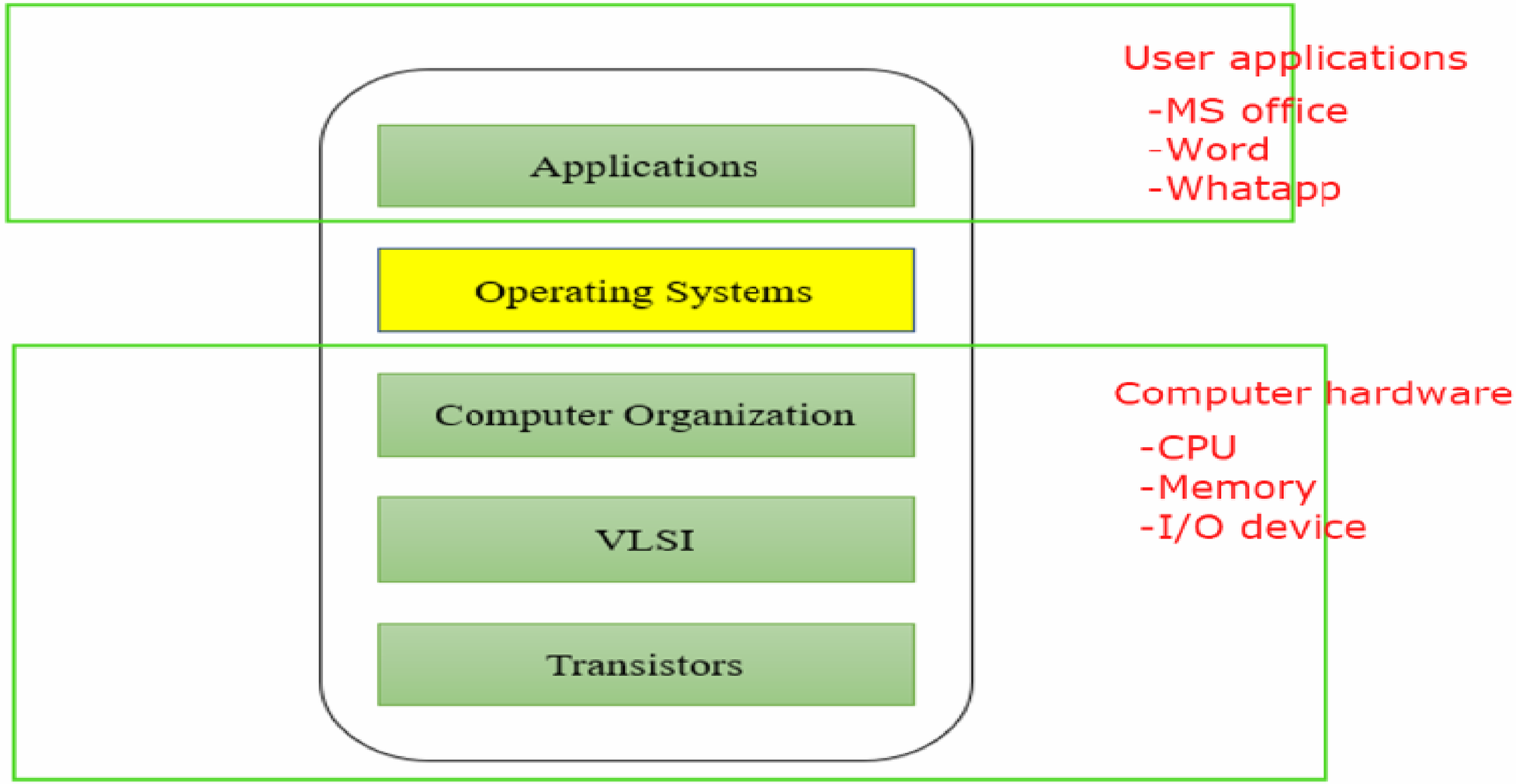


Operating System

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

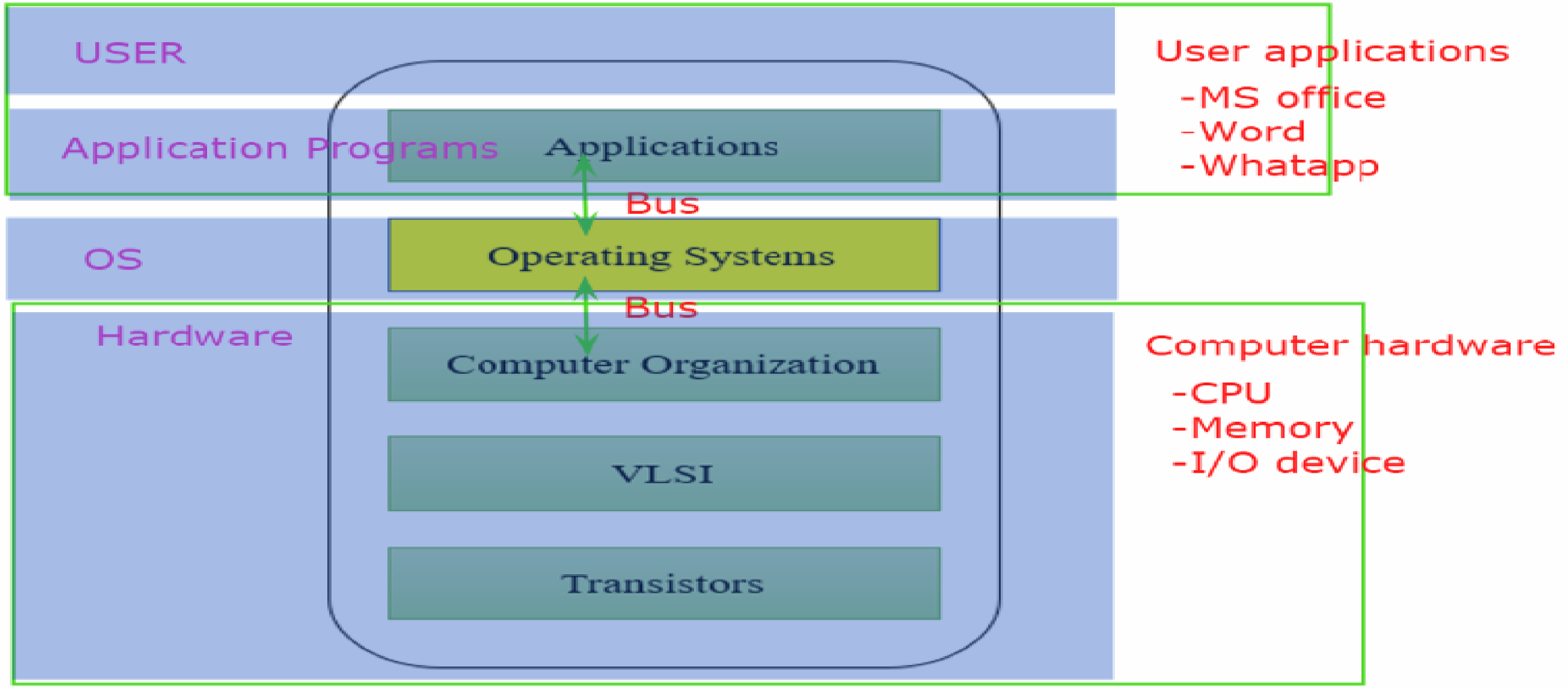
Computer Hardware

The Layers in Systems



1. Computer abstraction
2. Resource mangement

The Layers in Systems

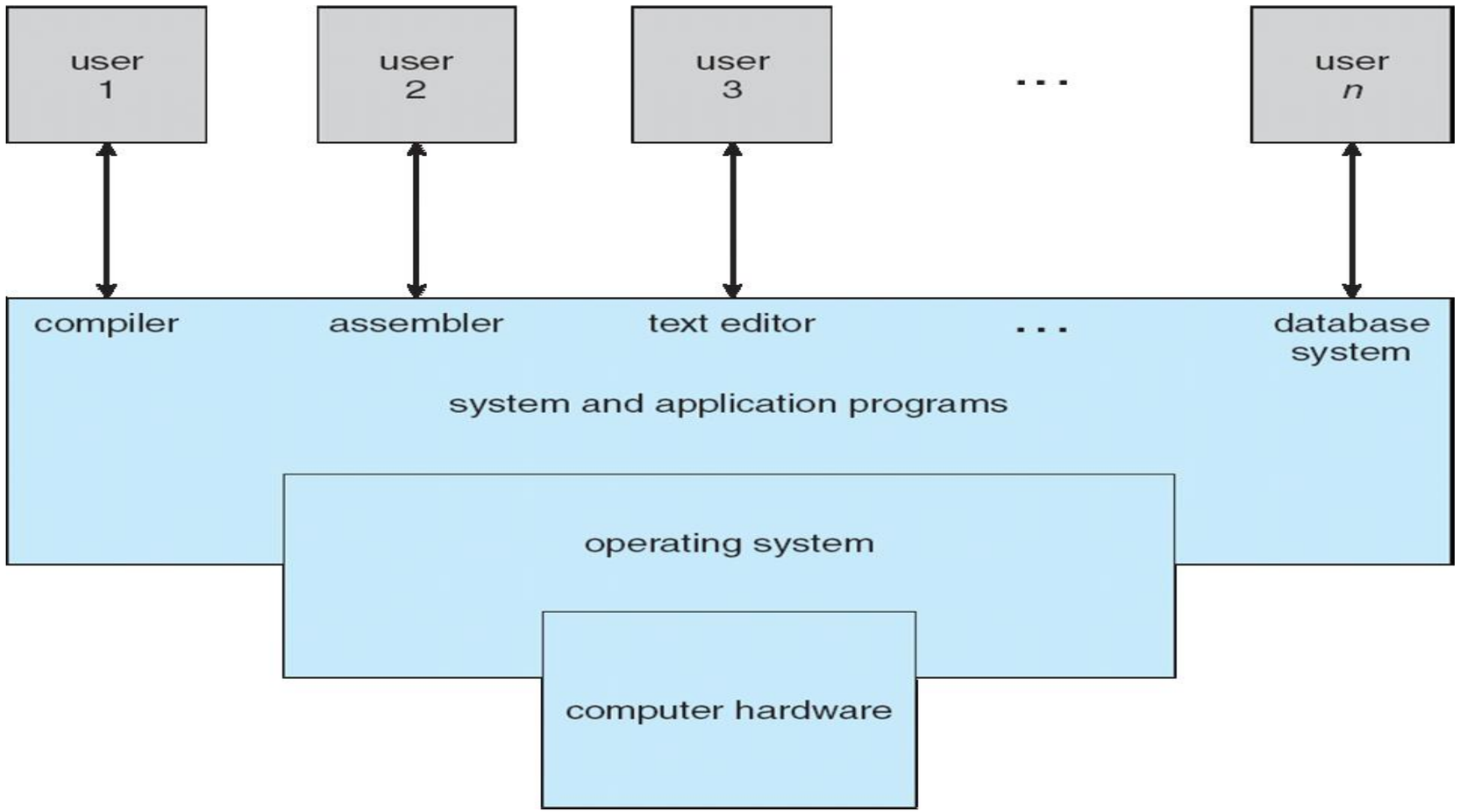


1. Computer abstraction
2. Resource mangement

Operating System

- An operating system is a program which **manages all the computer hardwares.**
- It provides the base for application program and acts as an intermediary between a user and the computer hardware.
- The operating system has **two objectives** such as:
  - Firstly, an operating system **controls the computer's hardware**.
  - The second objective is to **provide an interactive interface** to the user and interpret commands so that it can communicate with the hardware.
- The operating system is very important part of almost every computer system.

Four Components of a Computer System



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

- **OS is mainly designed in order to serve two basic purposes:**

1. The operating system mainly controls the allocation and use of the computing System's resources among the various user and tasks.

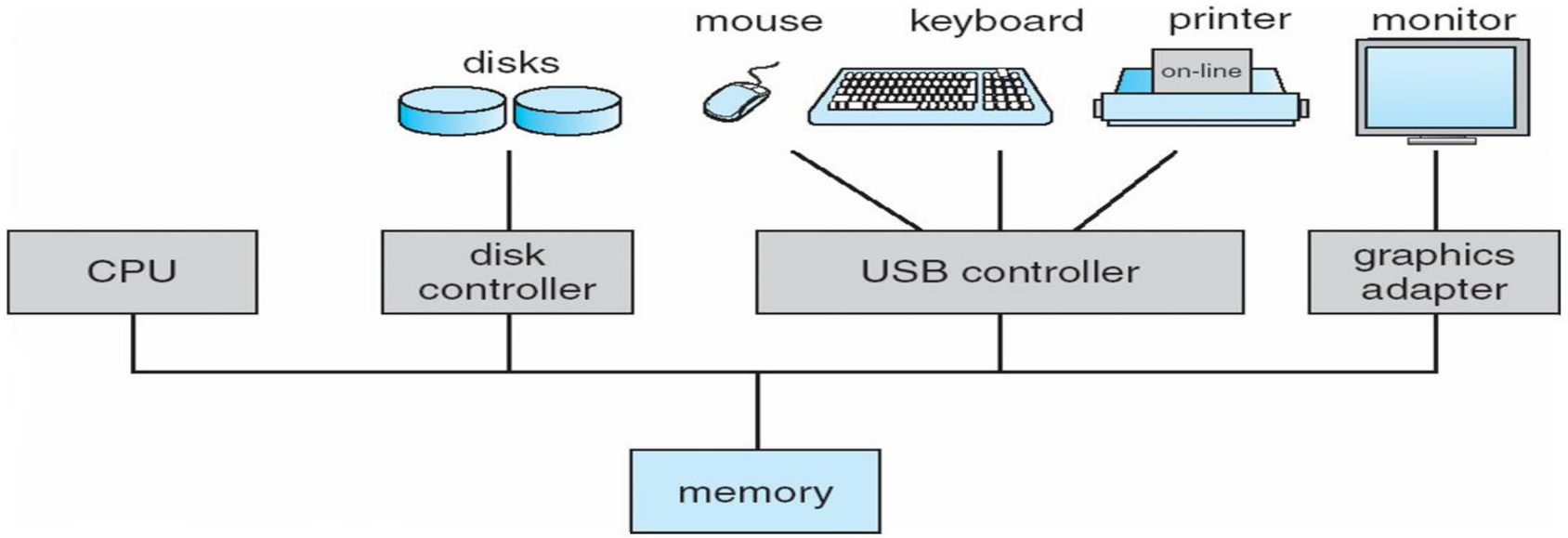
2. It mainly provides an interface between the computer hardware and the programmer that simplifies and makes feasible for coding, creation of application programs and debugging

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 competing for memory cycles



System and Application Programs

Operating System

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

Computer Hardware

System and Application Programs

Operating System

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

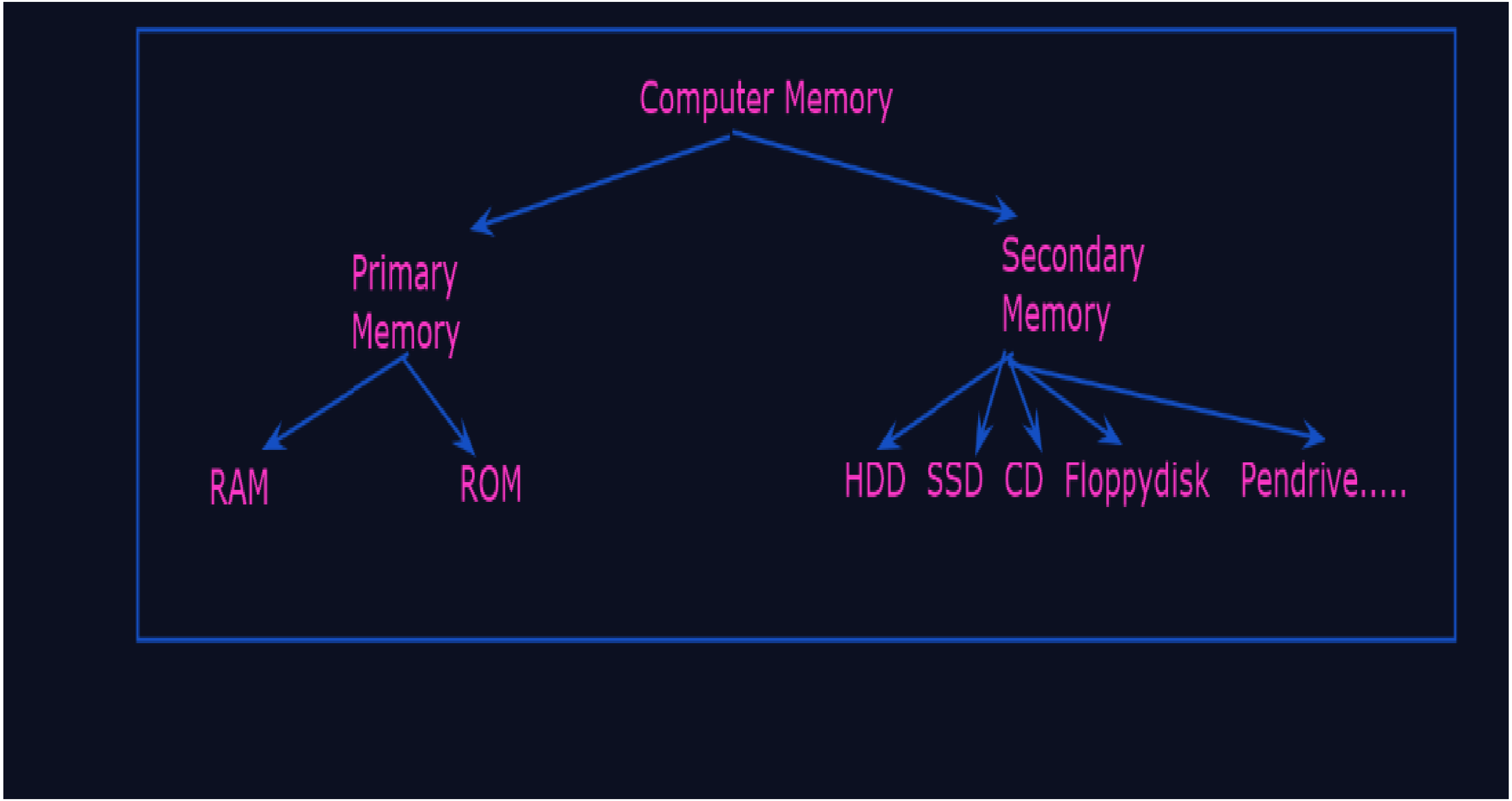
Bootstrap program

Kernel

Computer Hardware

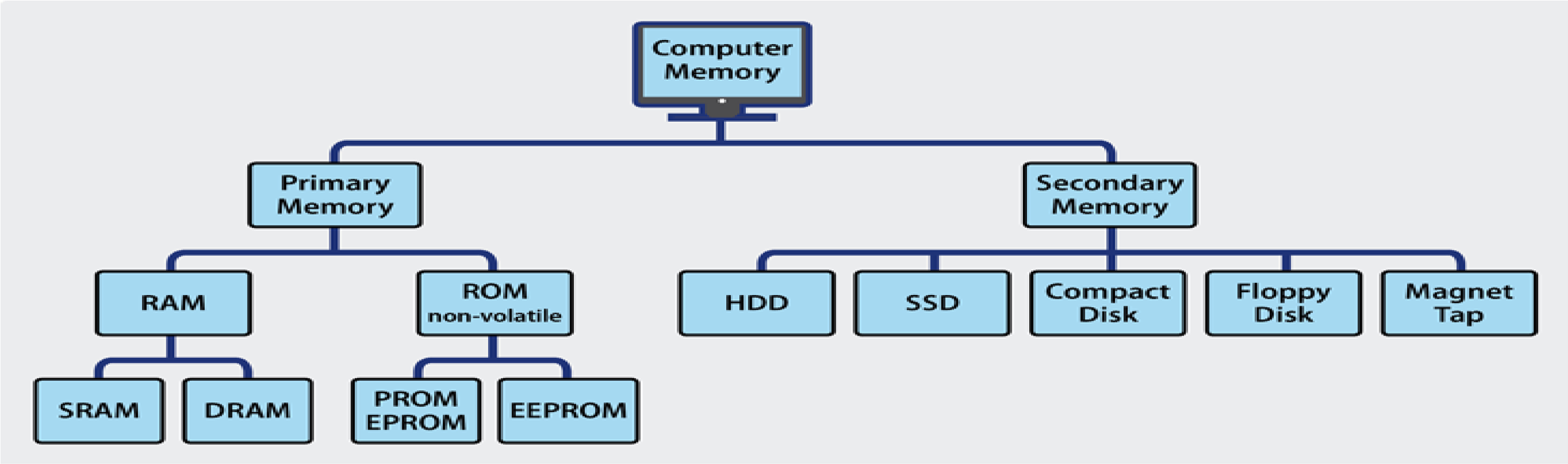
System Boot

- **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

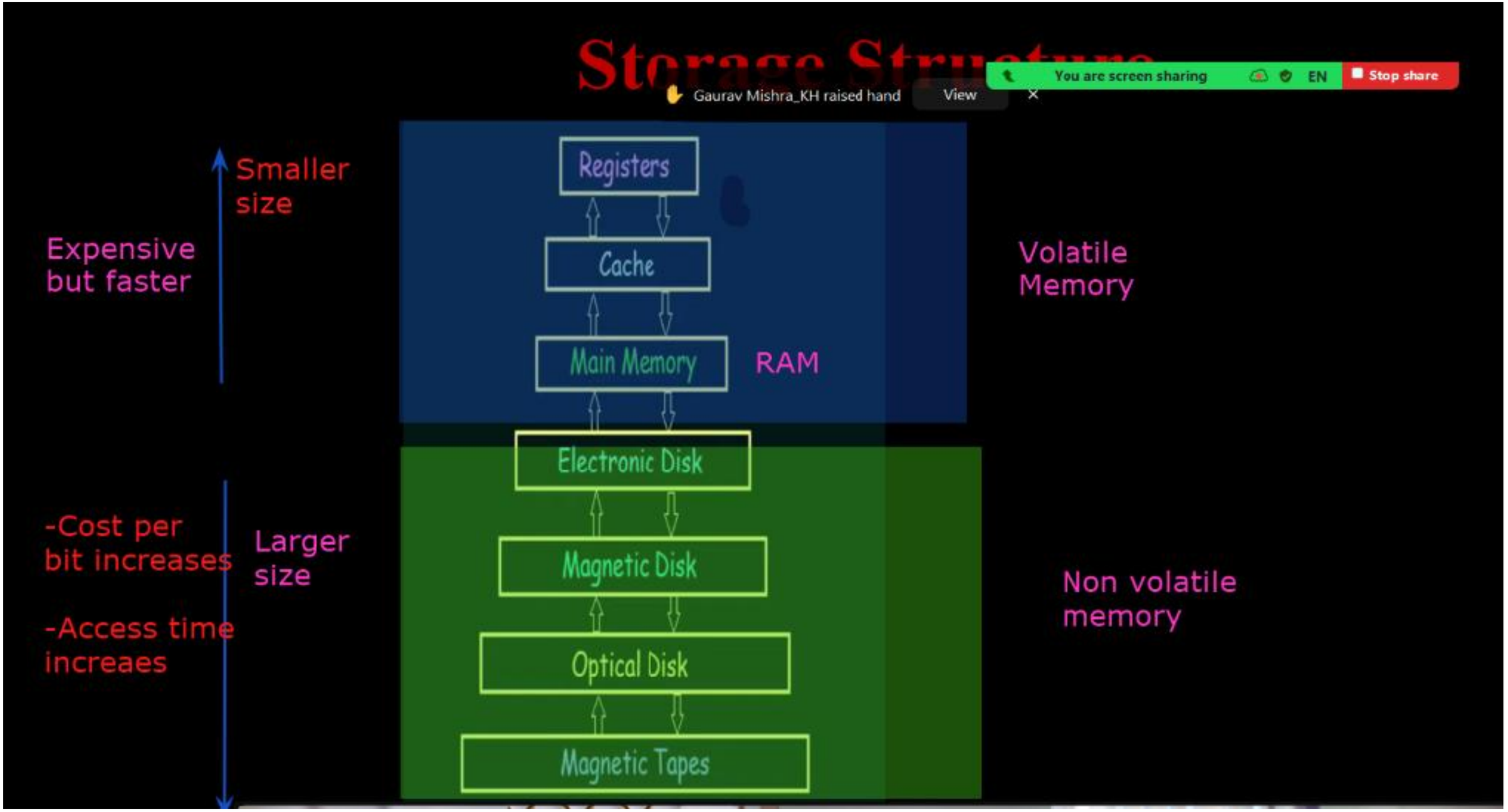

Computer Memory
Primary Memory
Secondary Memory
RAM
ROM
HDD
SSD
CD
Floppydisk
Pendrive.....

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

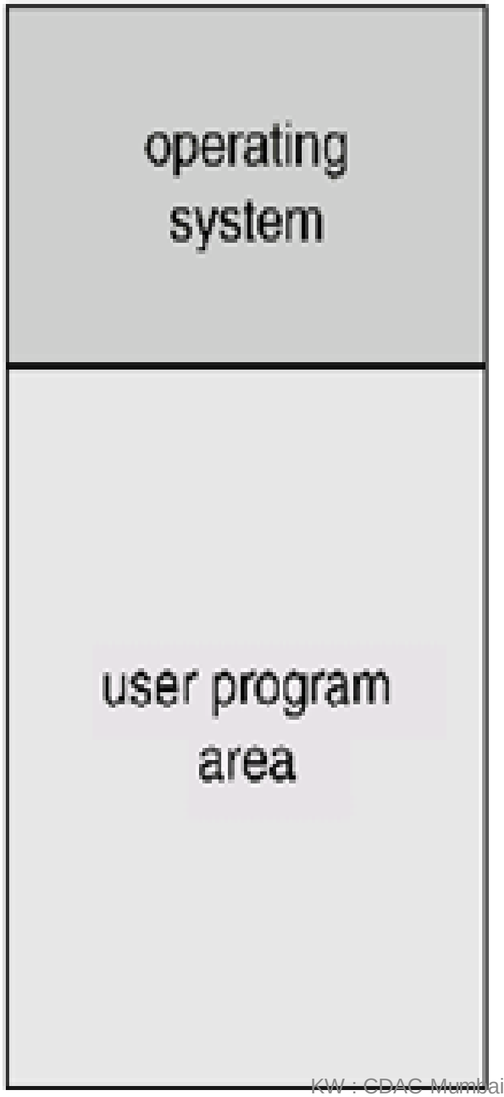


Types of Operating Systems

- **Following are some of the most widely used types of Operating system.**

  - Simple Batch System
  - Multiprogramming Batch System
  - Multiprocessor System
  - Desktop System
  - Distributed Operating System
  - Clustered System
  - Realtime Operating System
  - Handheld System

Memory Layout for a Simple Batch System



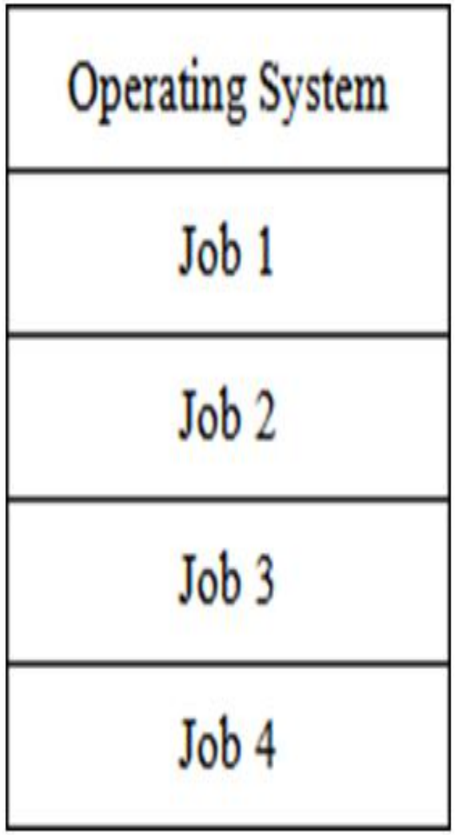
Simple Batch Systems

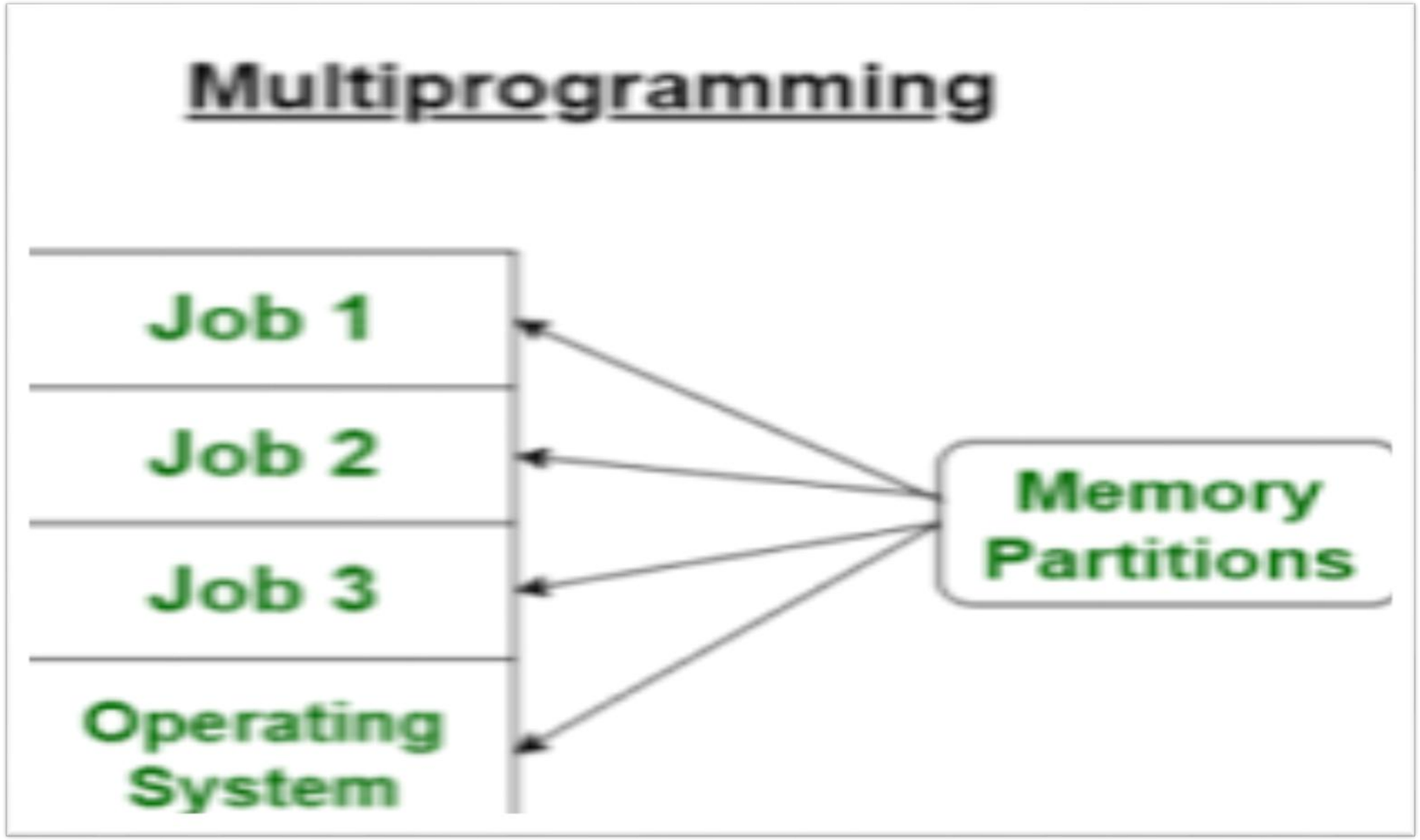
- In this type of system, there is **no direct interaction between user and the computer**.
- The user has to submit a job (written on cards or tape) to a computer operator.
- Then computer operator places a batch of several jobs on an input device.
- Jobs are batched together by type of languages and requirement.
- Then a special program, the monitor, manages the execution of each program in the batch.
- The monitor is always in the main memory and available for execution.

- **Advantages of Simple Batch Systems**

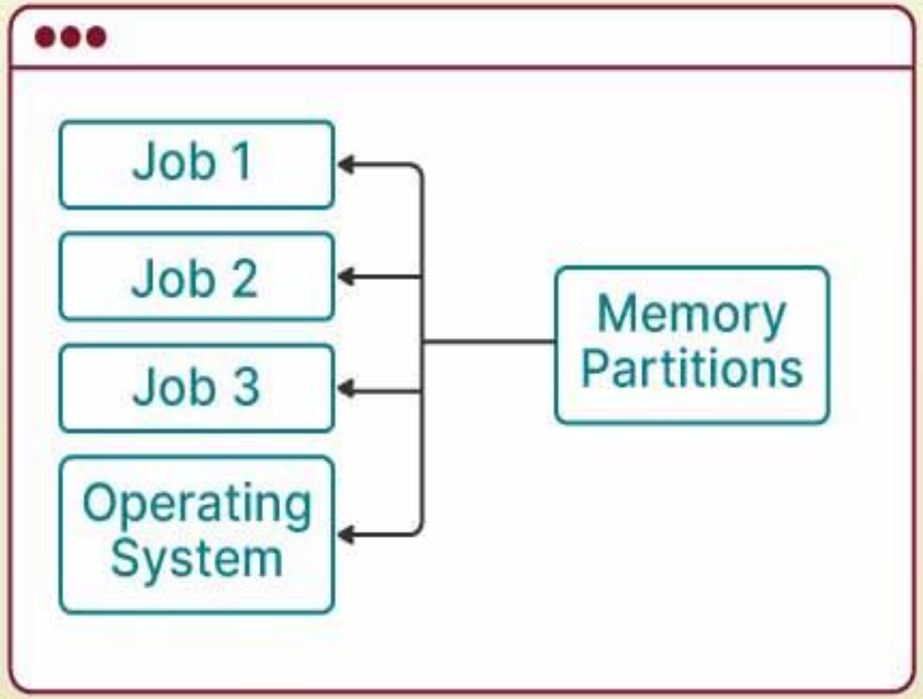
1. No interaction between user and computer.
2. No mechanism to prioritise the processes.

Multiprogramming

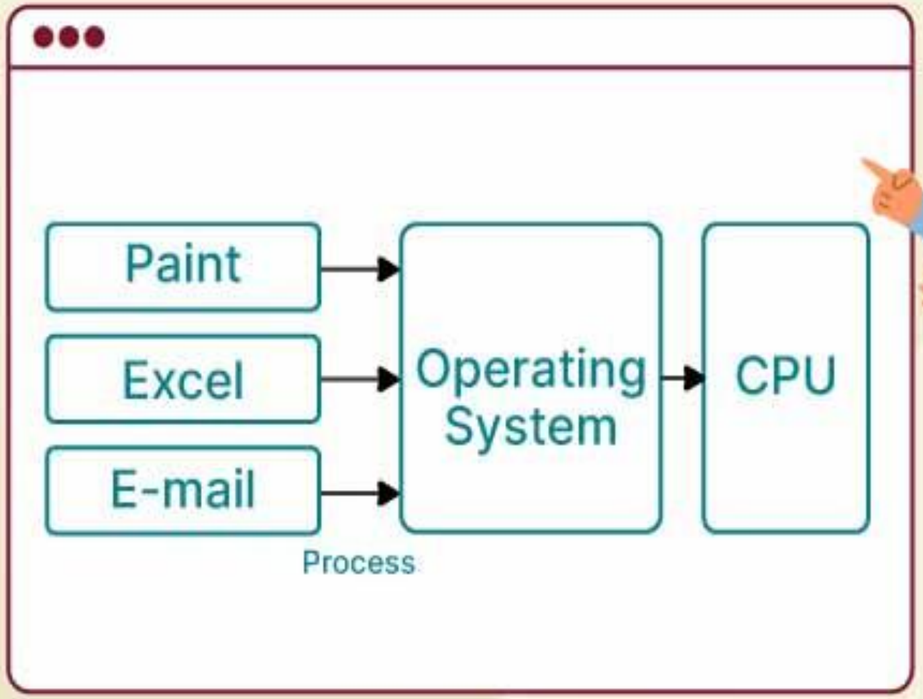




Multiprogramming and Multitasking



Multiprogramming

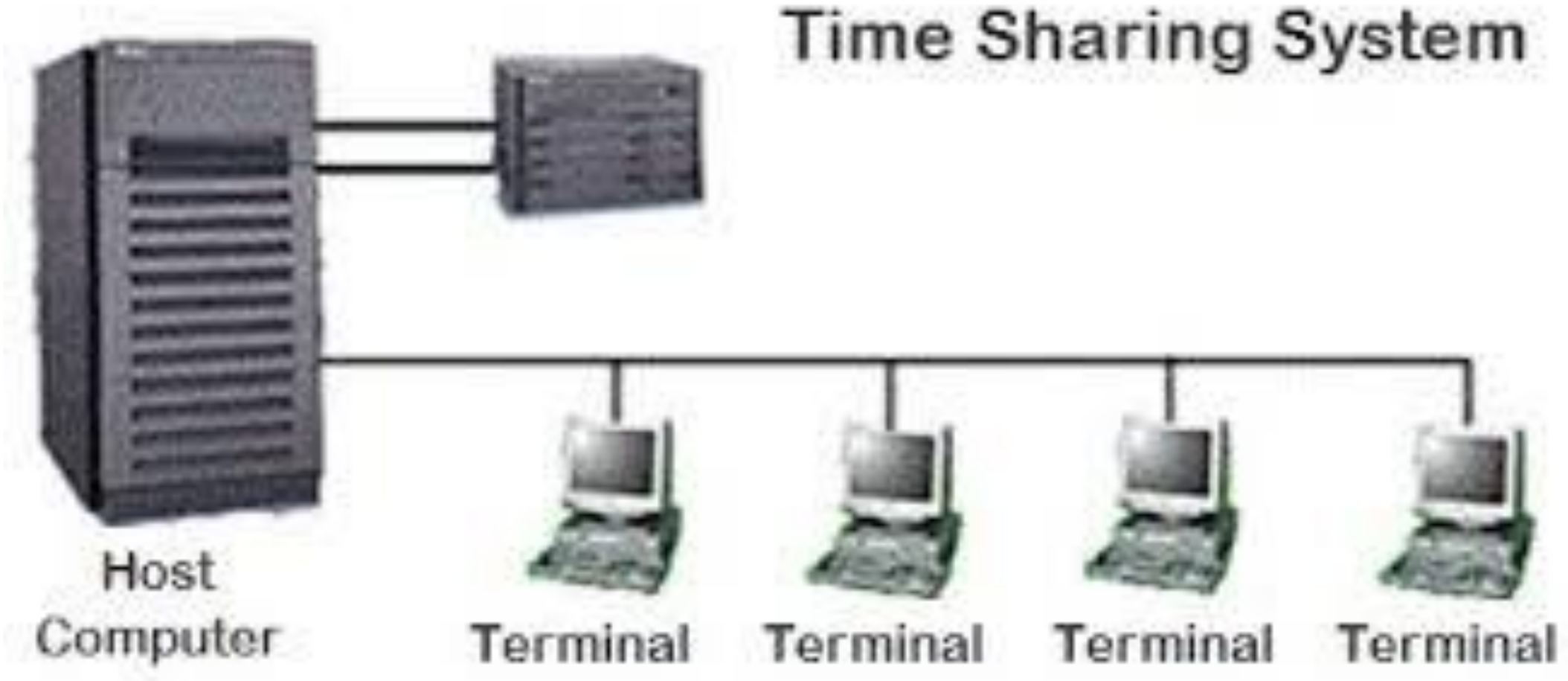


Multitasking

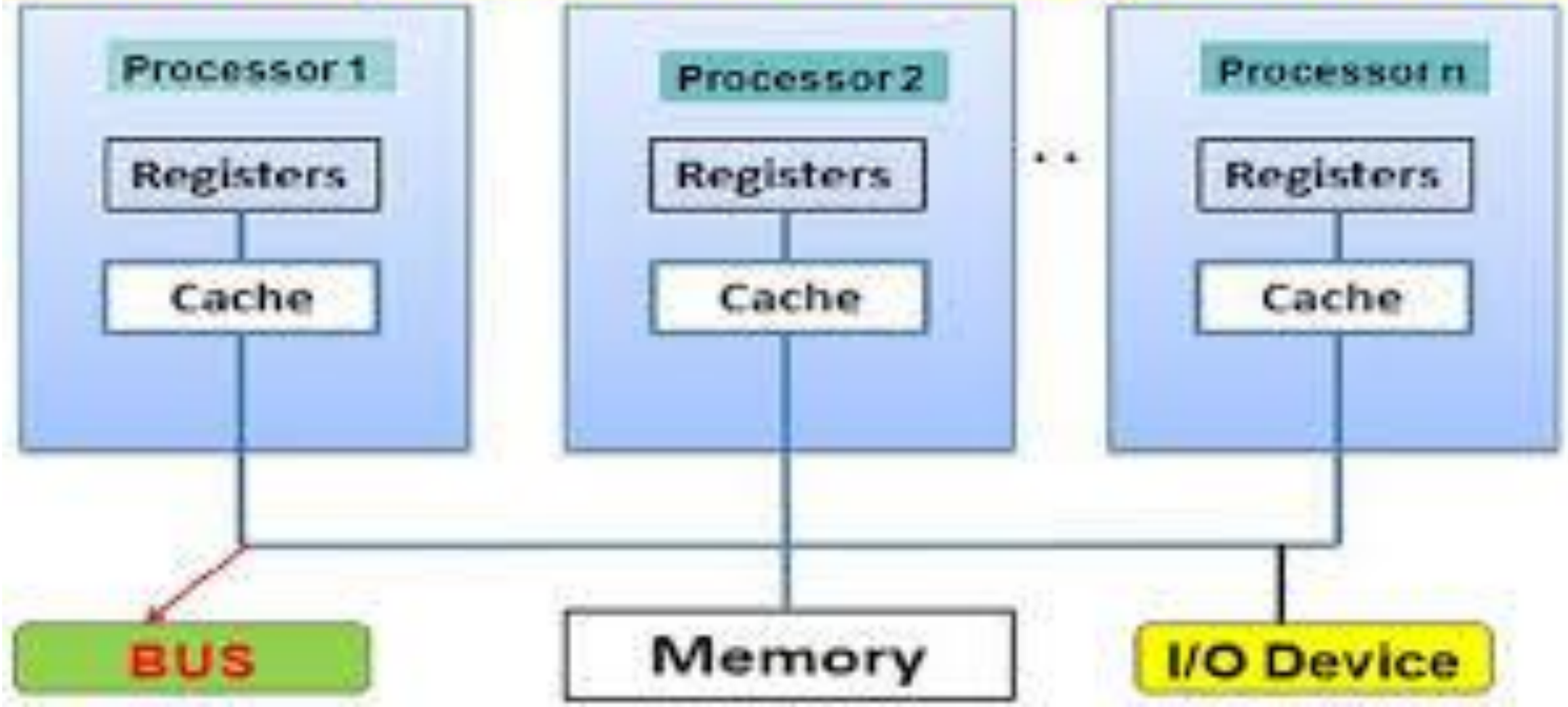
Operating System Structure

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

Time Sharing System



Multiprocessors Systems



Multiprocessor Systems

- A Multiprocessor system consists of **several processors** that share a common physical memory.
- Multiprocessor system provides **higher computing power and speed**.
- In multiprocessor system all processors **operate under single operating system**.
- **Multiplicity of the processors and how they do act together** are transparent to the others.

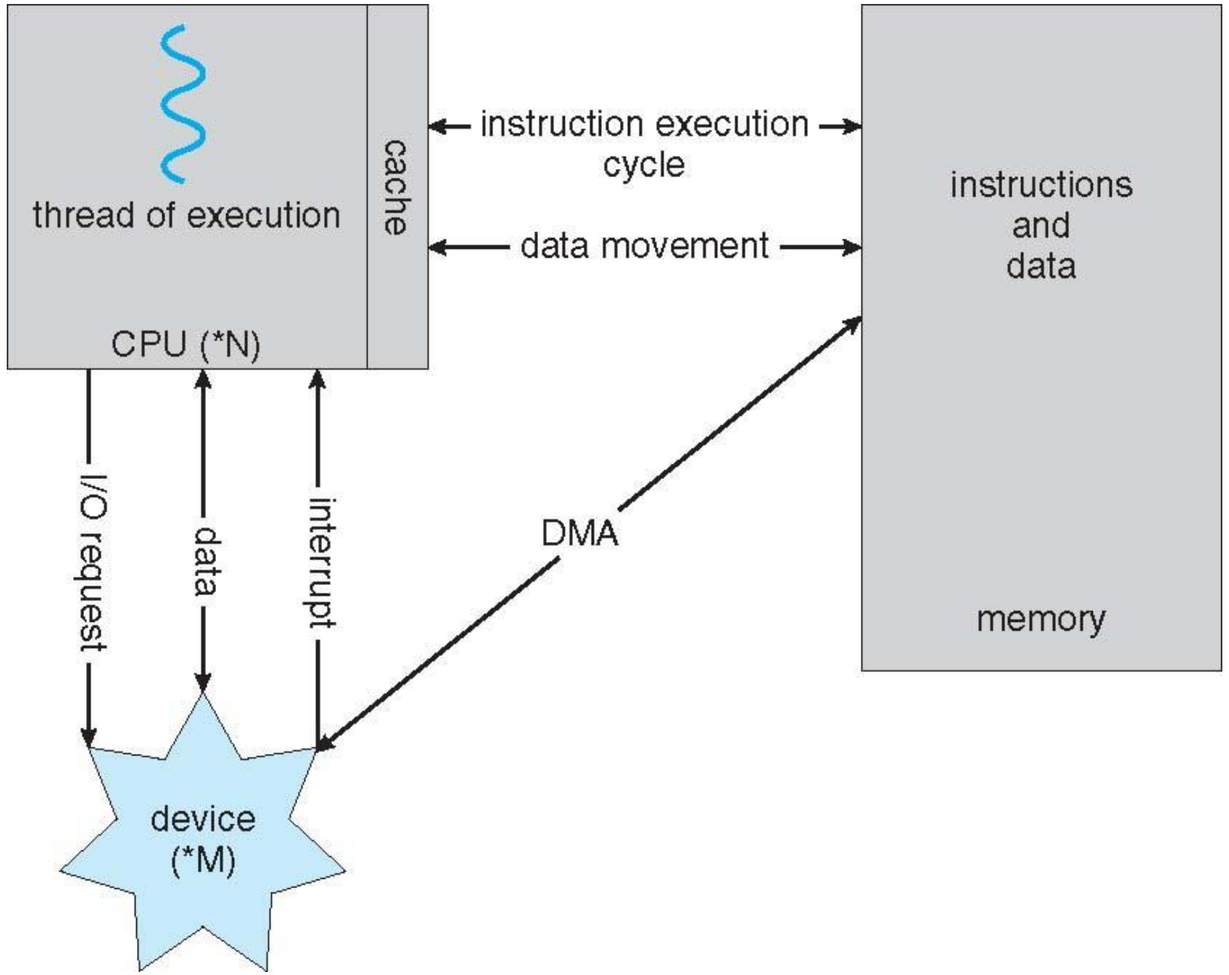
Advantages of Multiprocessor Systems

1. Enhanced performance
2. Execution of several tasks by different processors concurrently, increases the system's throughput without speeding up the execution of a single task.
3. If possible, system divides task into many subtasks and then these subtasks can be executed in parallel in different processors. Thereby speeding up the execution of single tasks.

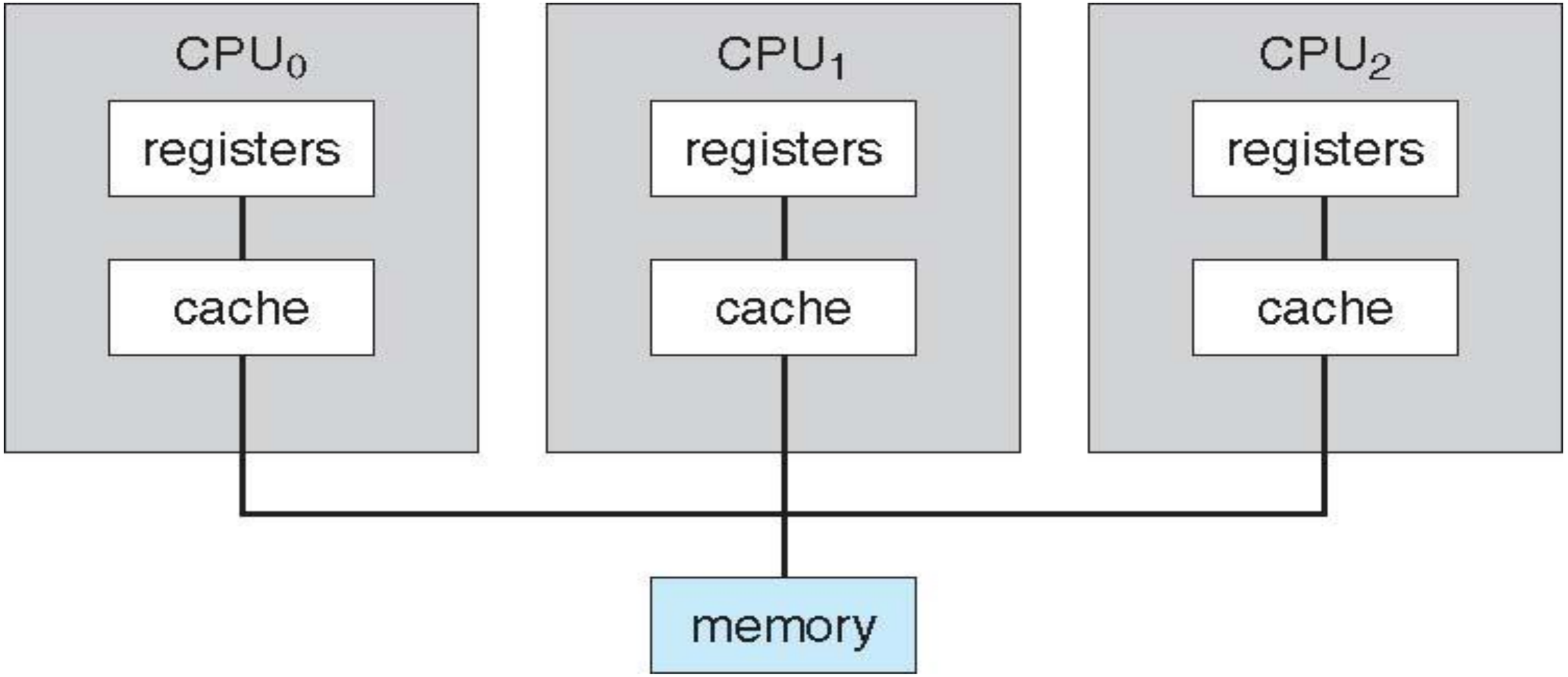
Computer-System Architecture

- **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
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

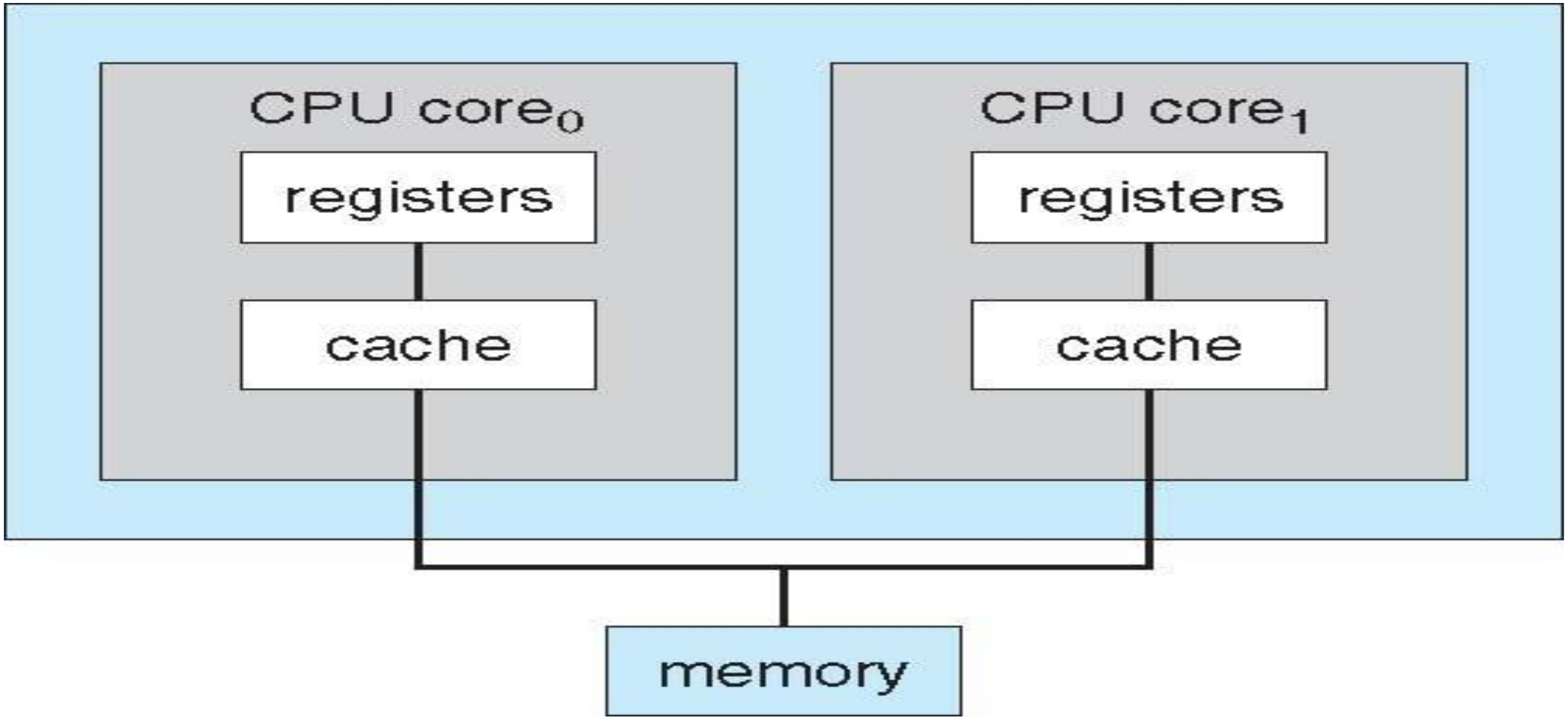
How a Modern Computer Works



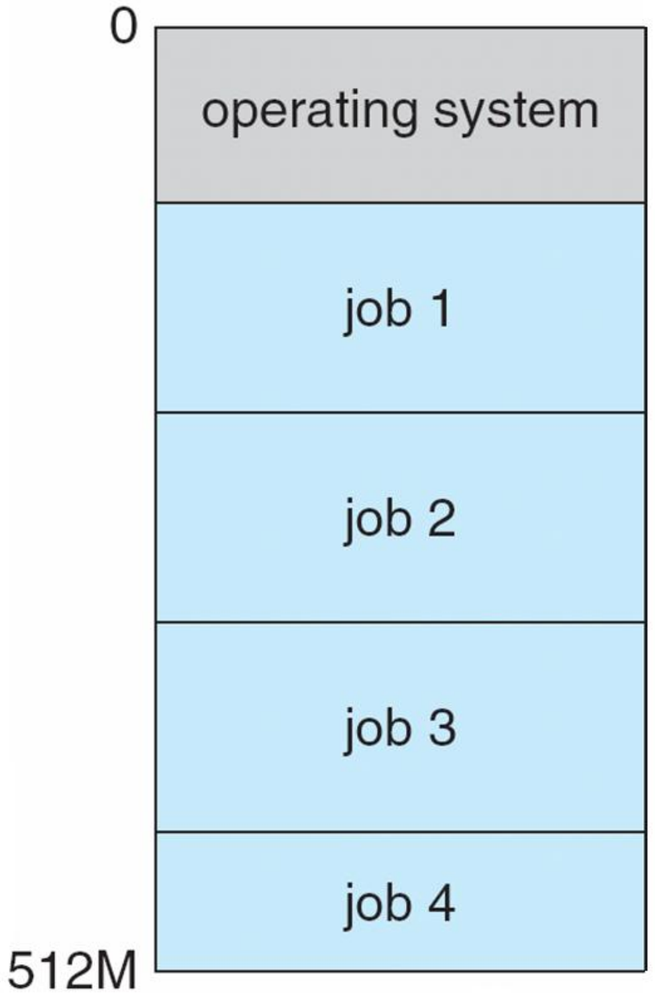
Symmetric Multiprocessing Architecture



A Dual-Core Design

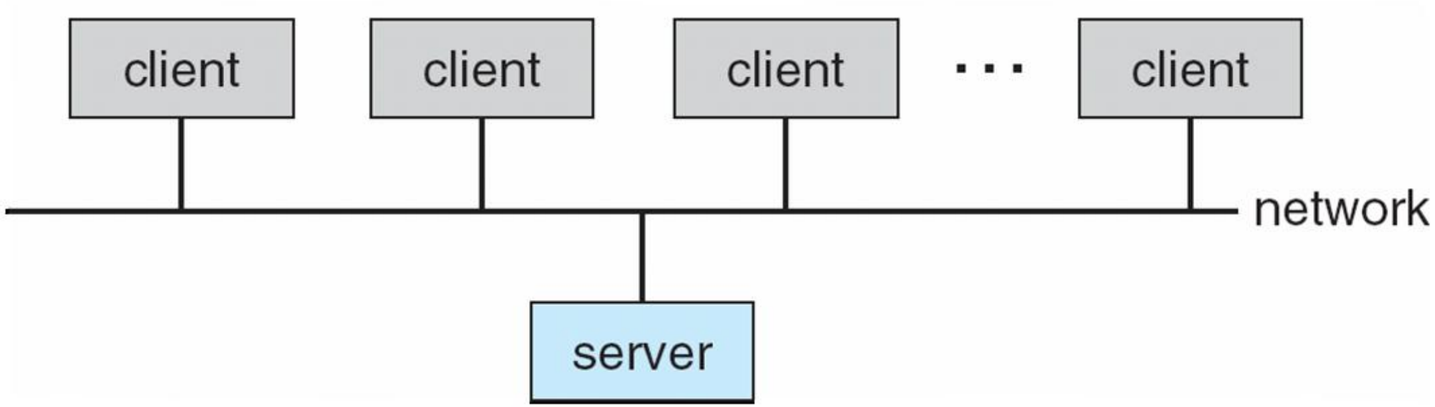


Memory Layout for Multiprogrammed System



Computing Environments (Cont)

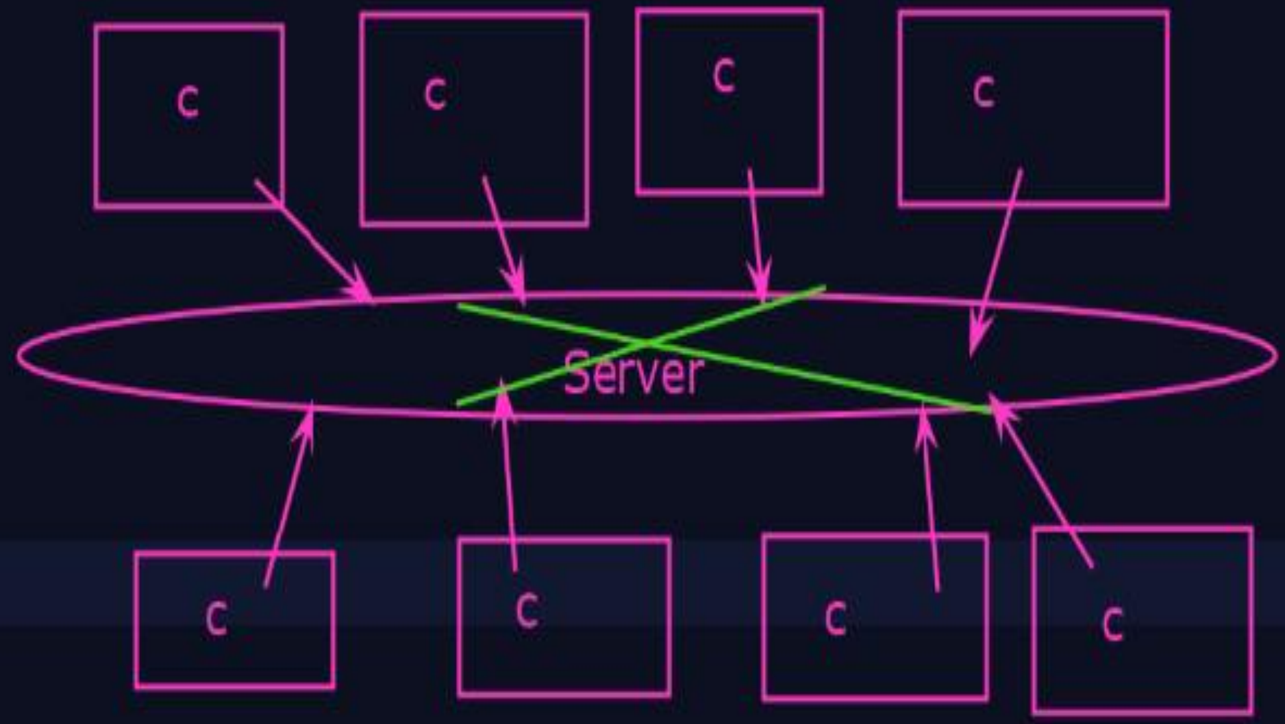
- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server** provides an interface to client to request services (i.e. database)
    - **File-server** provides interface for clients to store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

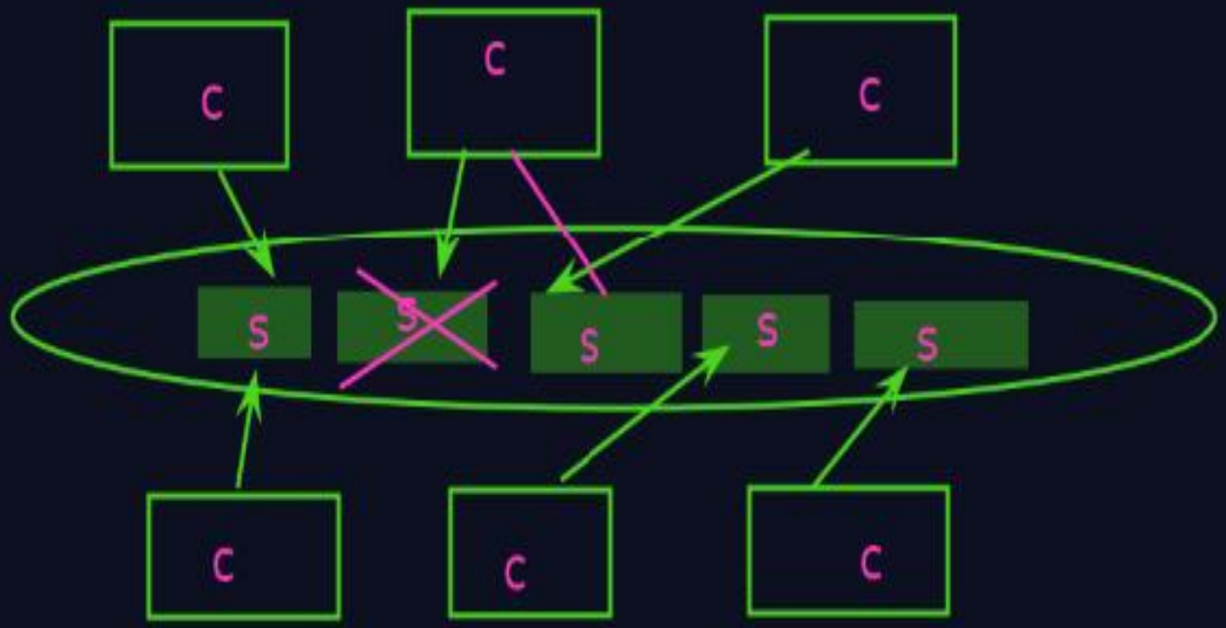


Multiprocessor System:
---

1. Client-server architecute

2. Peer to Peer architecture





Multiprocessor System:

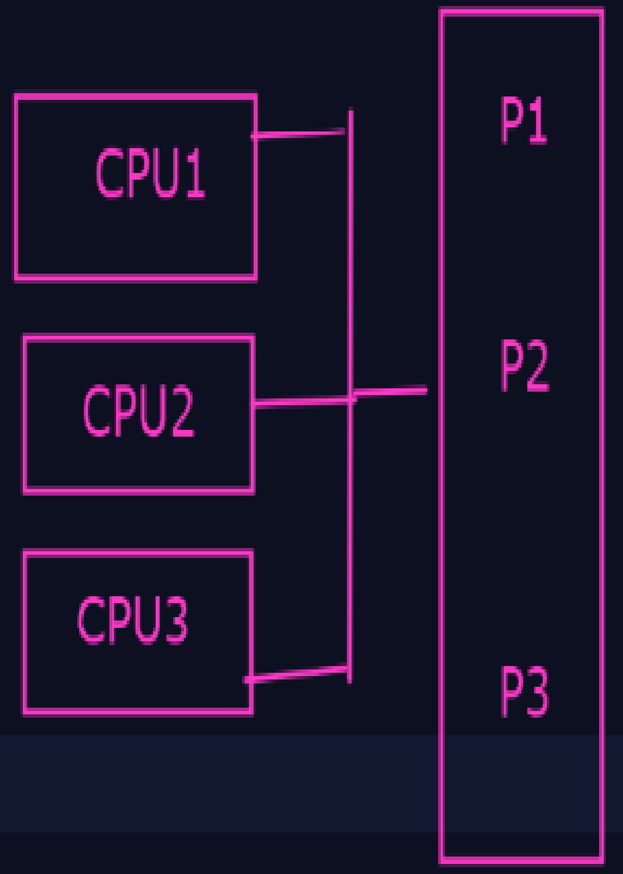
---

1. Client-server architecute

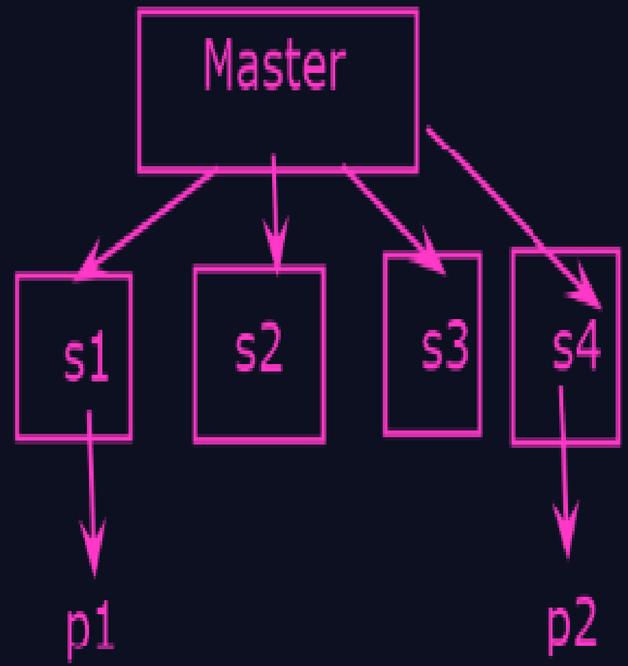
2. Peer to Peer architecture

3. Symmetric Multiprogramming

4. Asymmeteric Multiprogramming

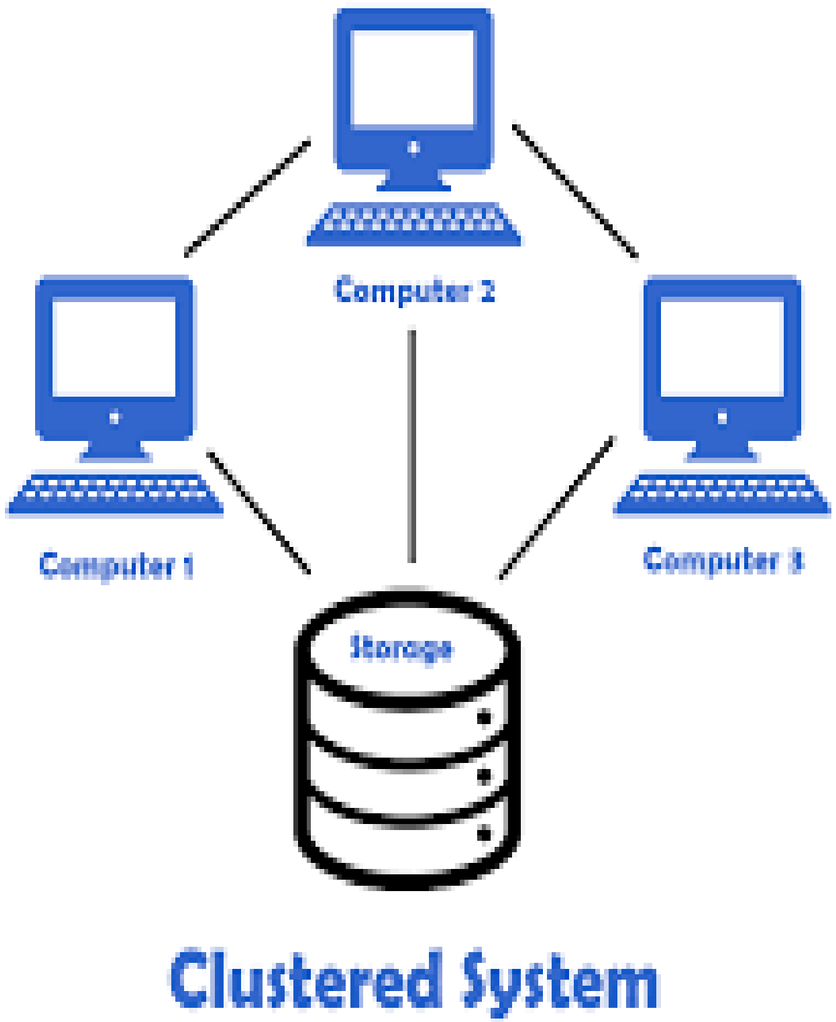


Symmetric Multiprogramming



Asymmeteric Multiprogramming

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

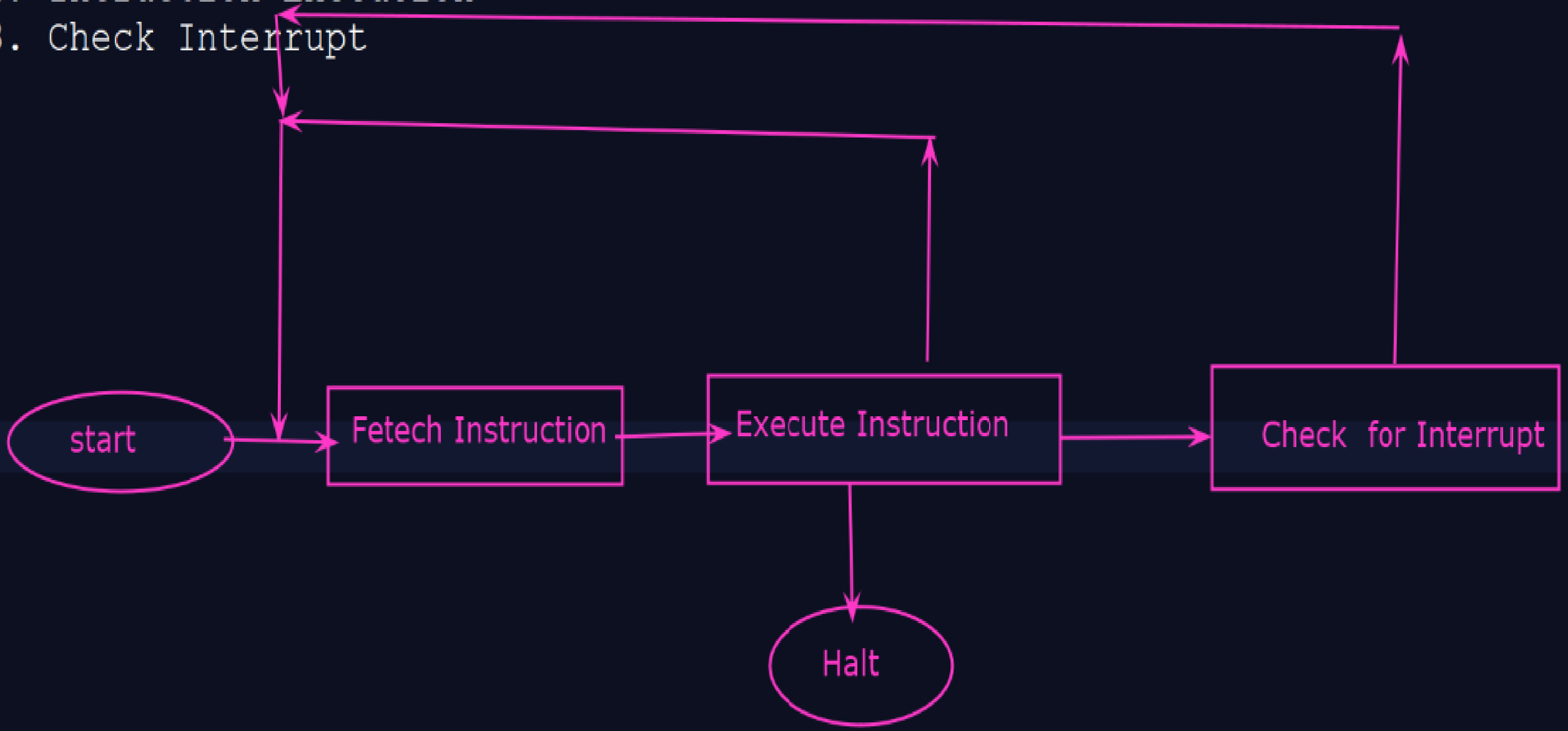
Real Time Operating System

- **The Real-Time Operating system which guarantees the maximum time for critical operations and complete them on time are referred to as Hard Real-Time Operating Systems.**

- **While the real-time operating systems that can only guarantee a maximum of the time, i.e. the critical task will get priority over other tasks, but no assurity of completing it in a defined time. These systems are referred to as Soft Real-Time Operating Systems.**

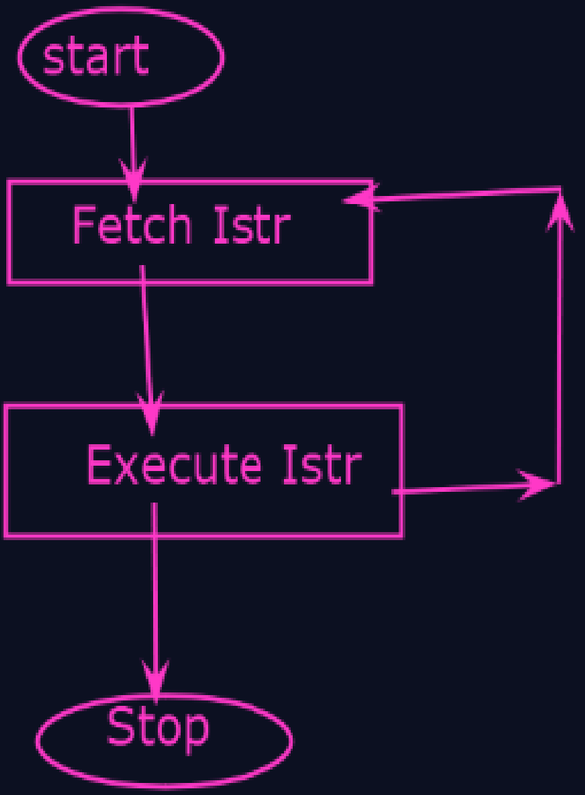
With Interrupt:

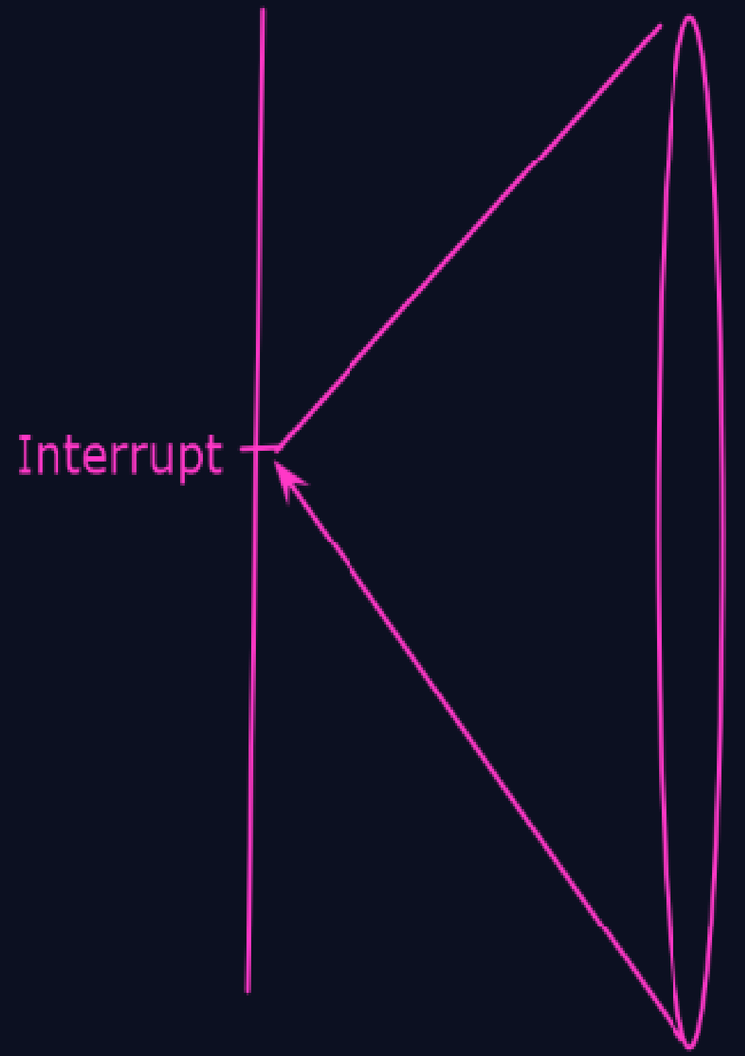
1. Instruction Fetch
2. Intruction Execution
3. Check Interrupt



Intrrupt Handling Technique:

1. Polling
2. Vectored interrupted system





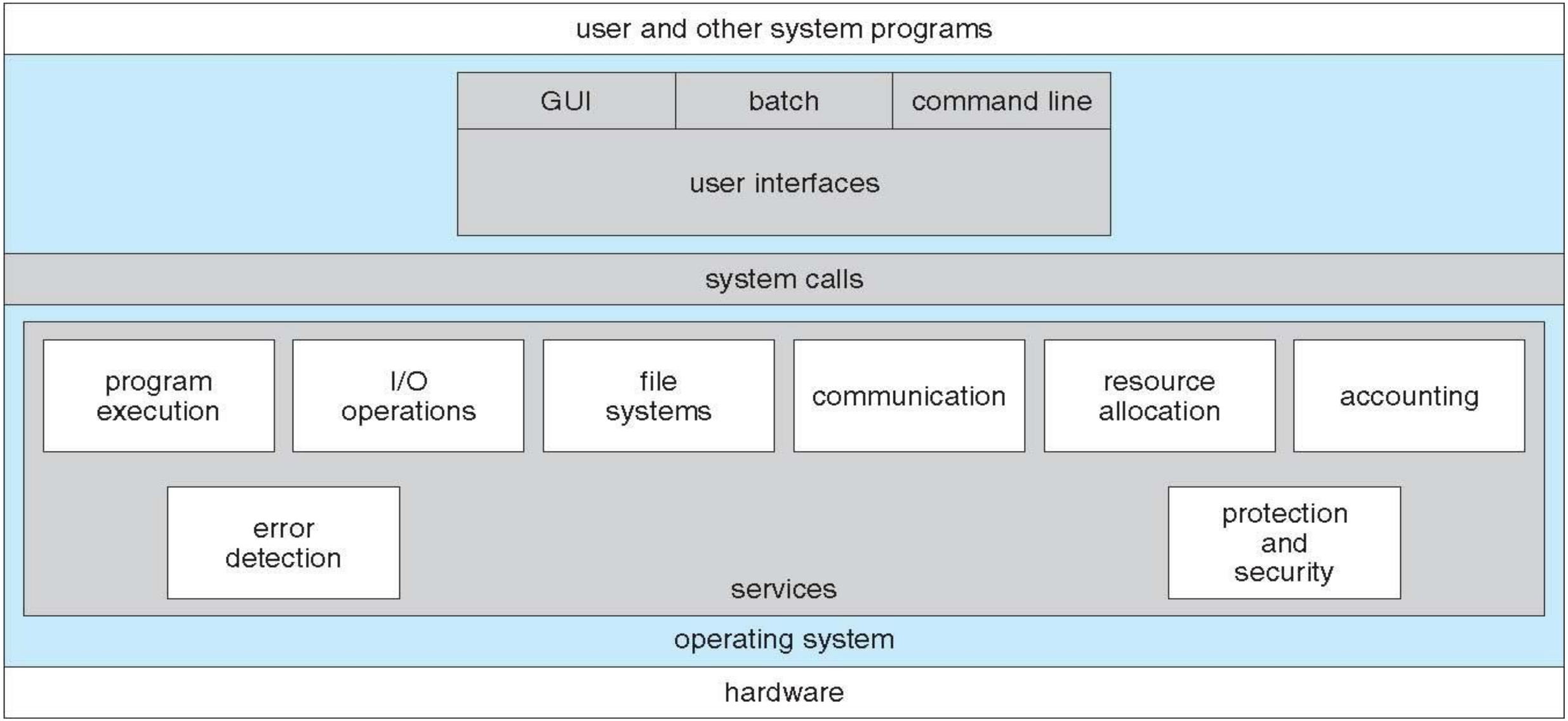
Interrupt Handling

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

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.

A View of Operating System Services



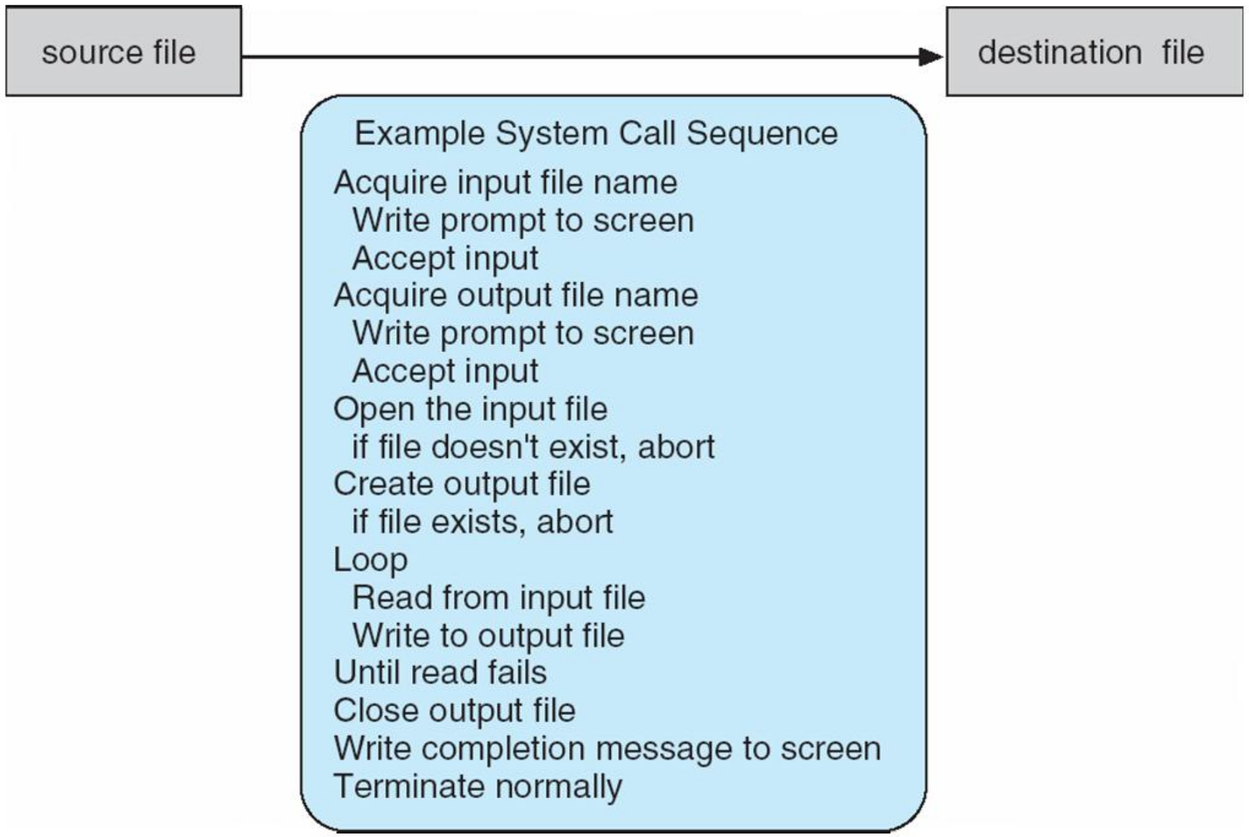
System Calls

- **Programming interface to the services provided by the OS**
- **Typically written in a high-level language (C or C++)**
- **Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use**
- **Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)**
- **Why use APIs rather than system calls?**

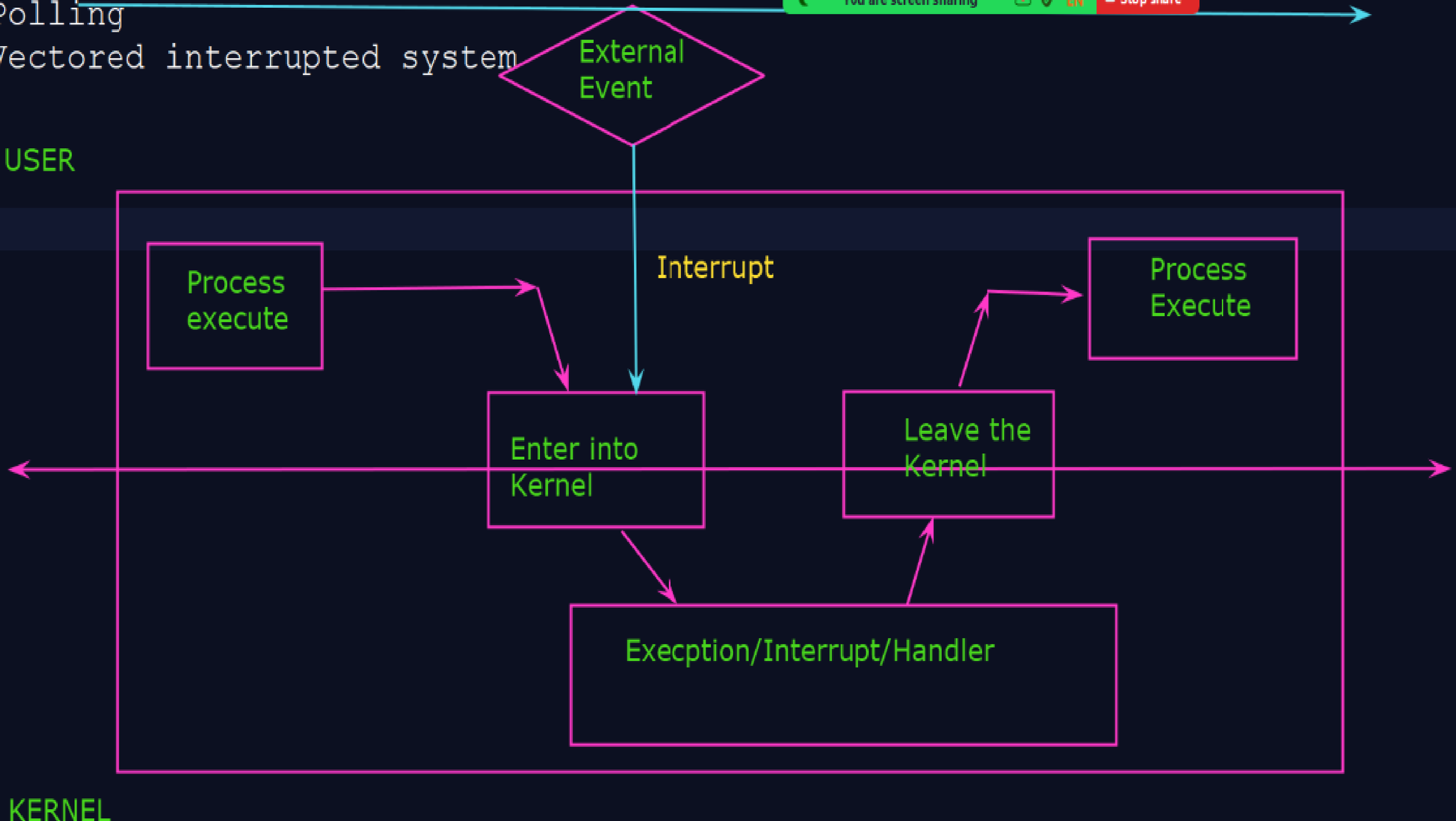
(Note that the system-call names used throughout this text are generic)

Example of System Calls

- **System call sequence to copy the contents of one file to another file**



1. Polling
2. Vectored interrupted system

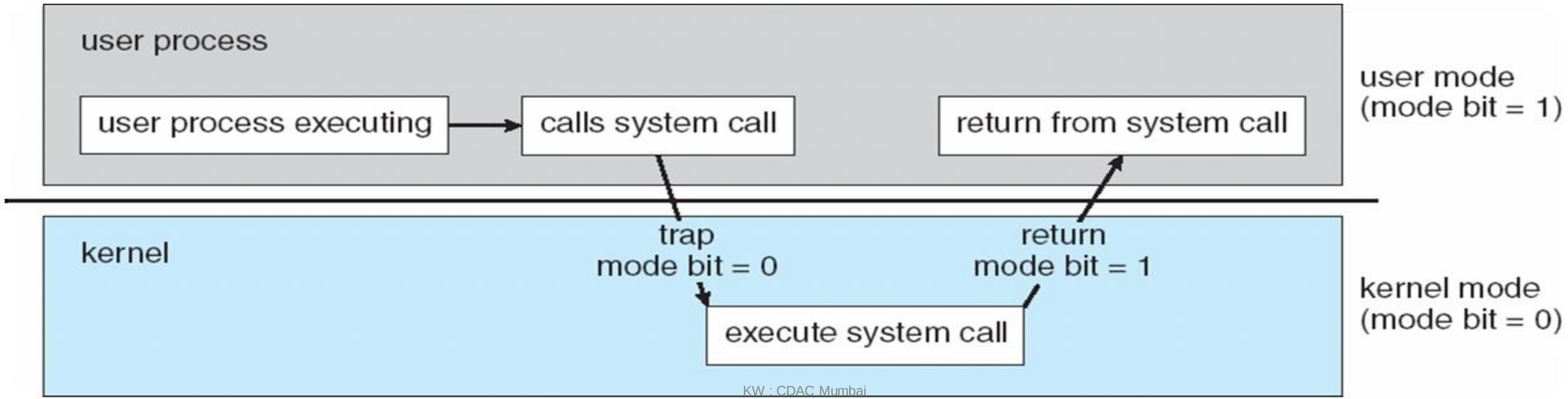


Operating-System Operations

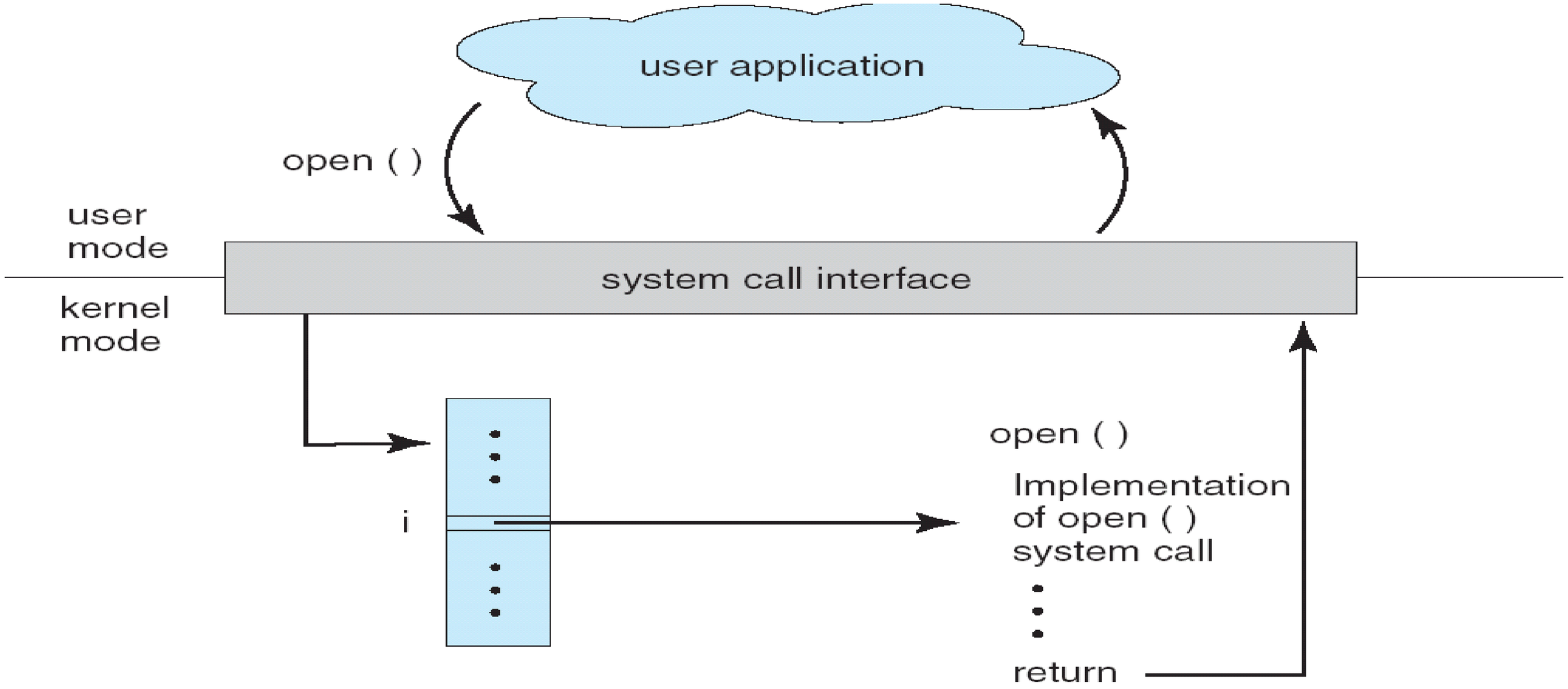
- **Interrupt driven by hardware**
- **Software error or request creates exception or trap**
  - Division by zero, request for operating system service
- **Other process problems include infinite loop, processes modifying each other or the operating system**
- **Dual-mode operation allows OS to protect 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

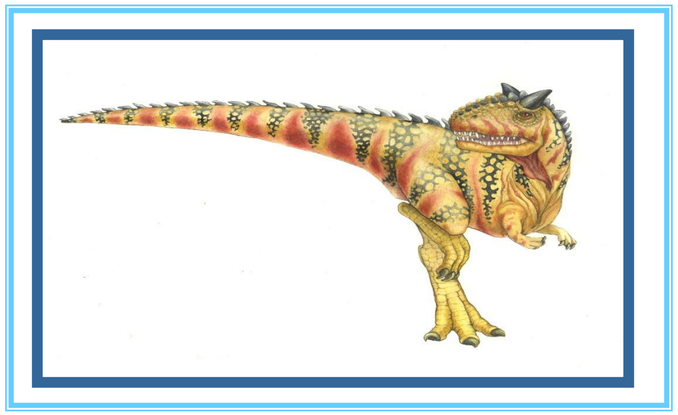
- **Timer to prevent infinite loop / process hogging resources**
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



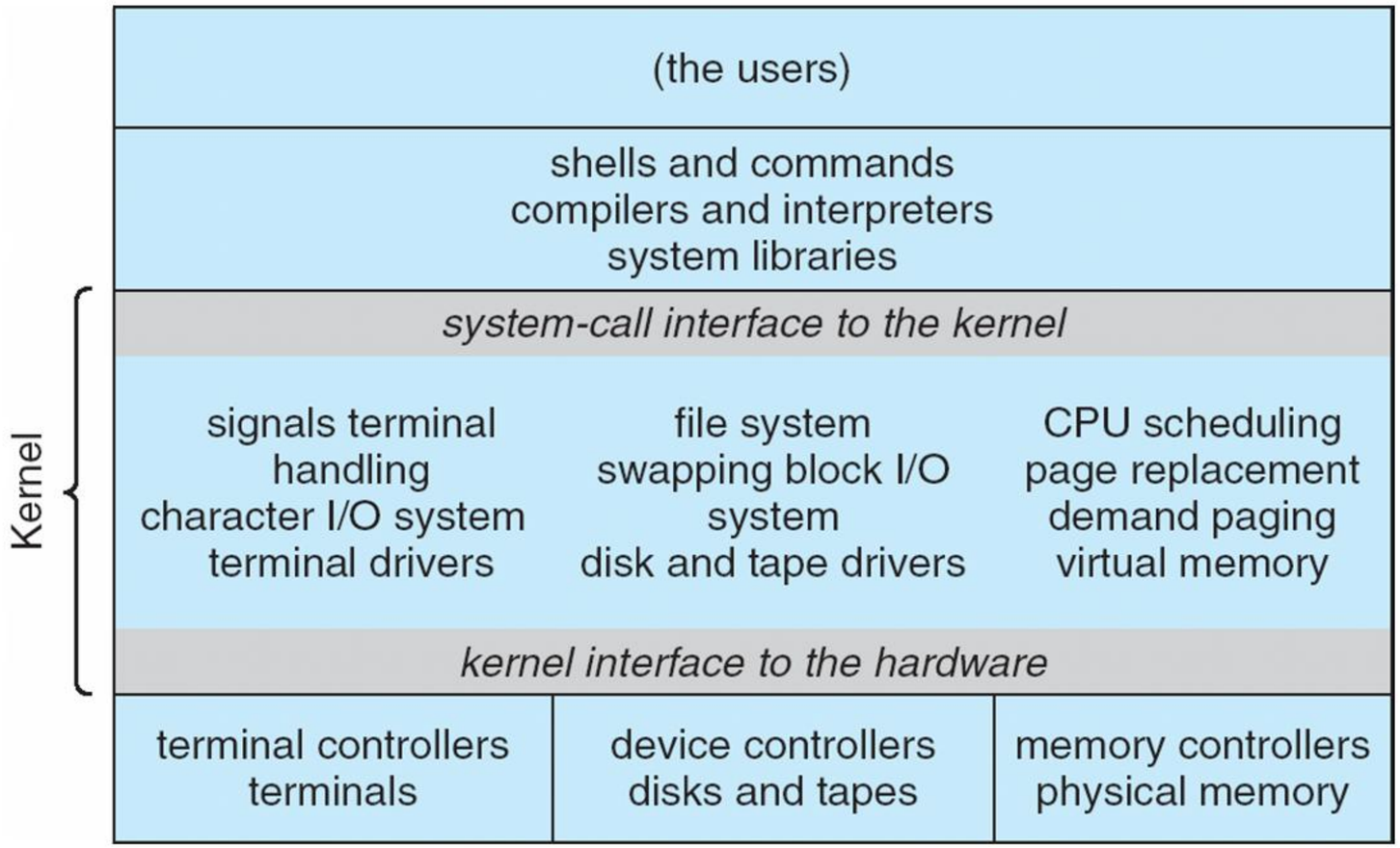
API – System Call – OS Relationship



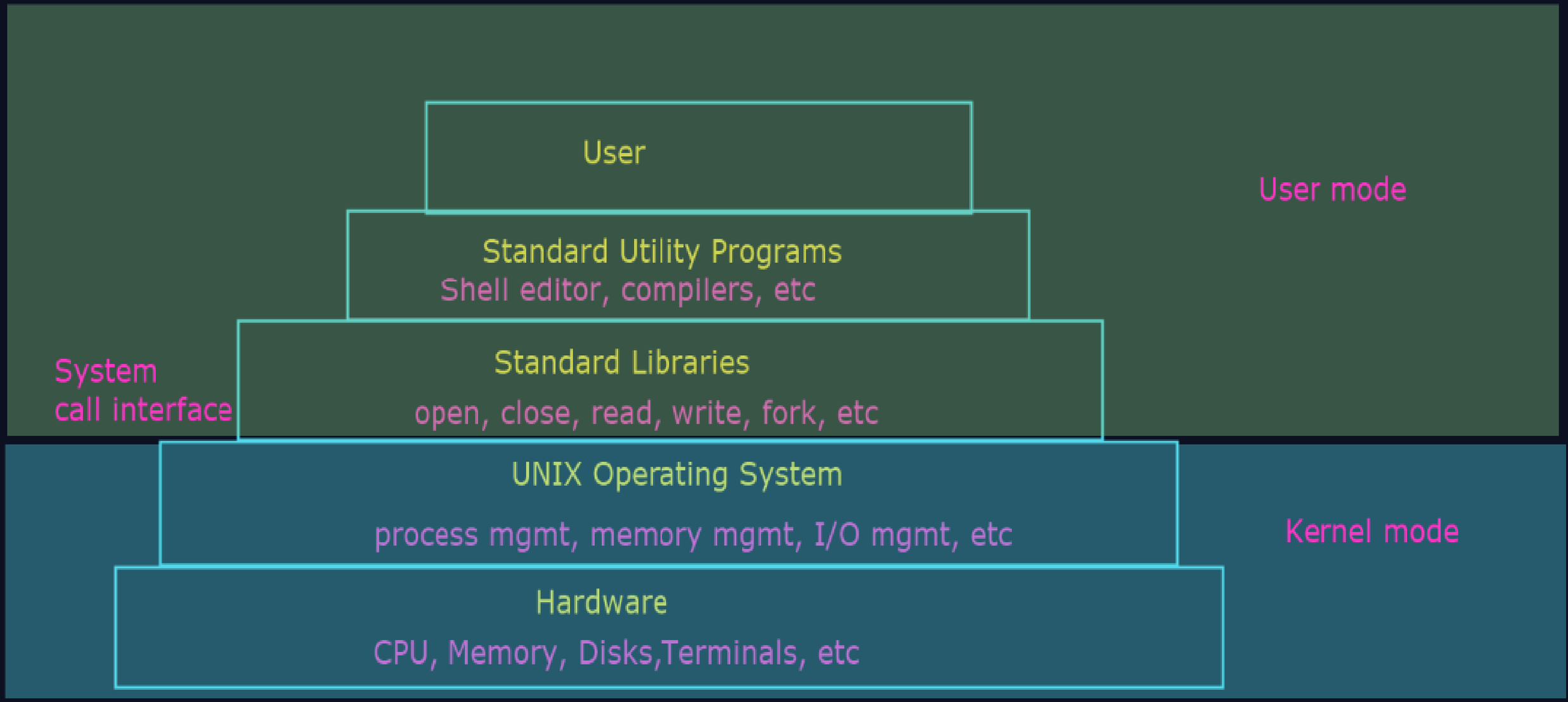
Introduction to Linux

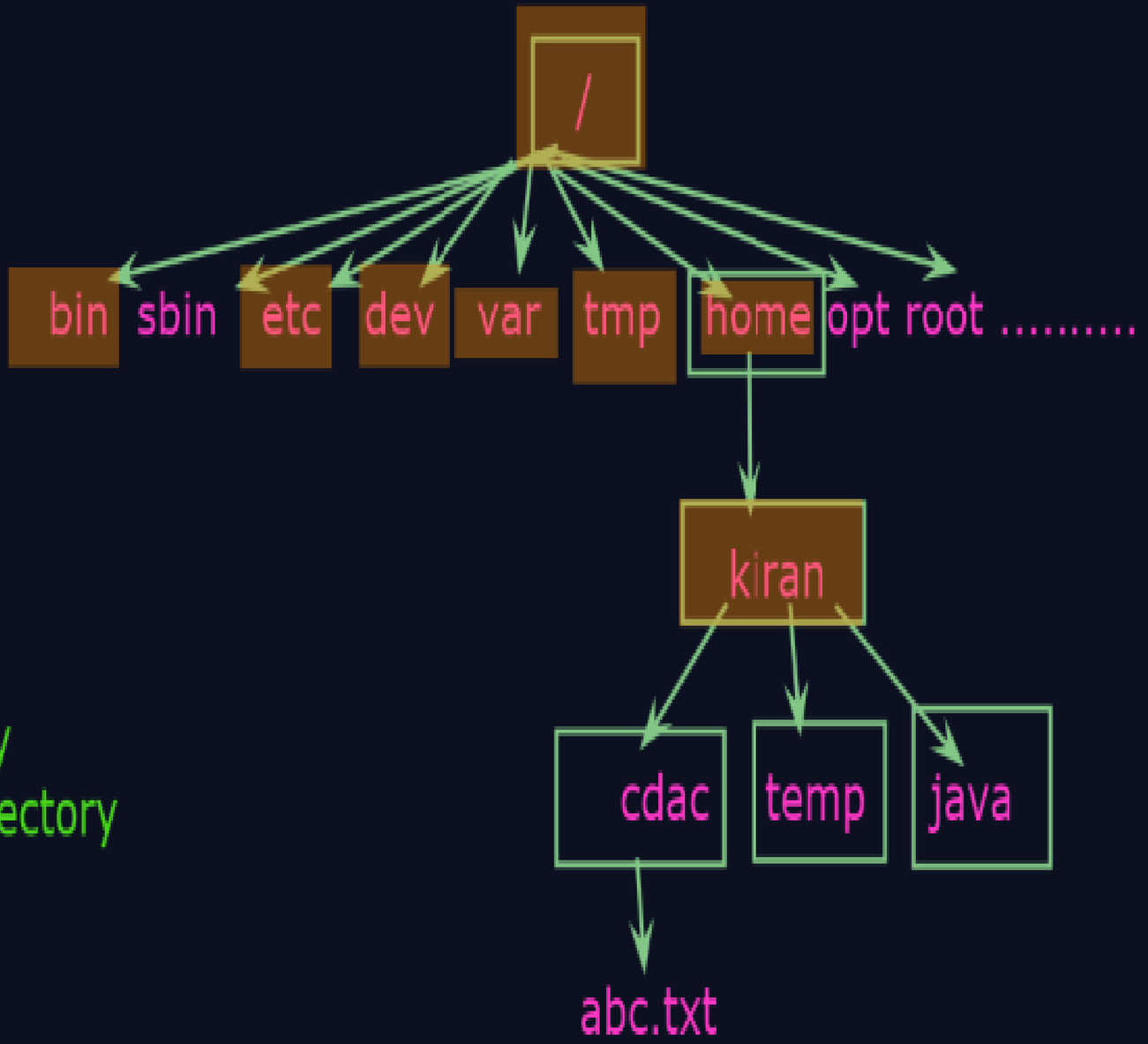


Traditional UNIX System Structure



Layers of Linux/Unix:





Directory :

. : Directory
.. : Parent directory
~ : User's home directory