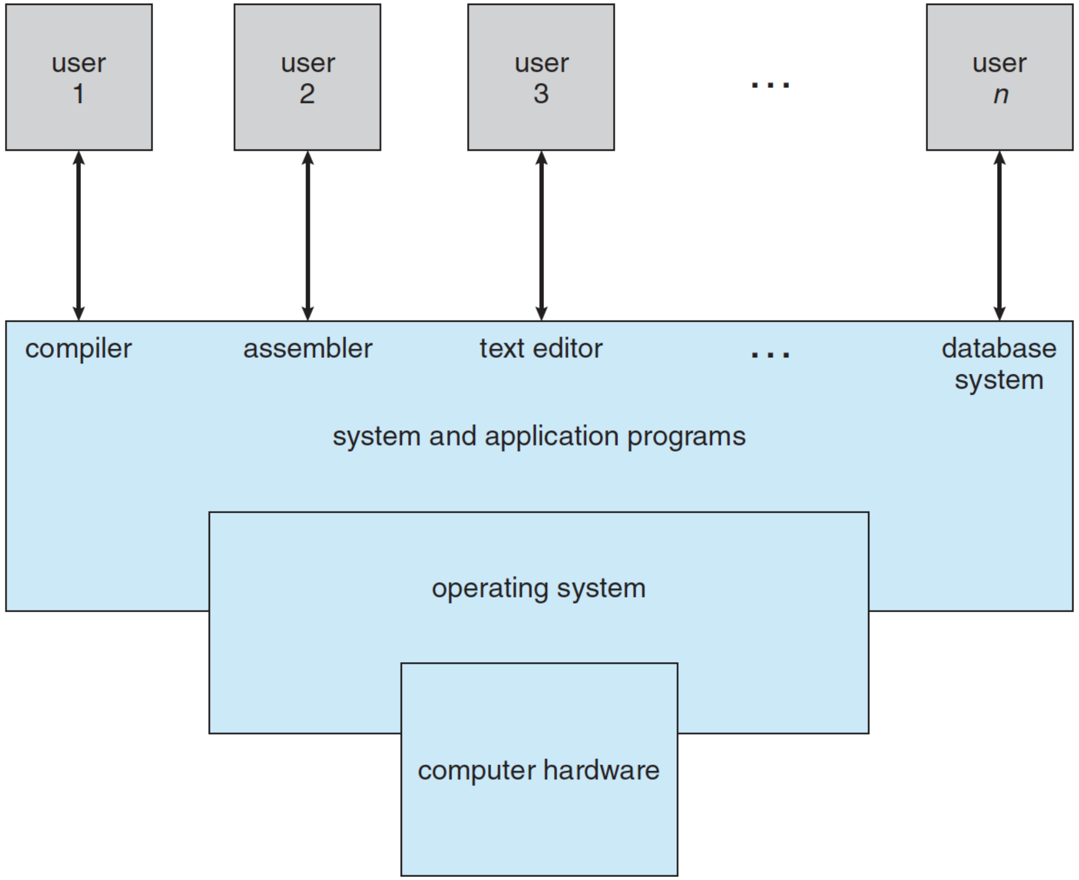
Introduction to Operating Systems

In this lesson

- Define what an operating system (OS) is.
- Understand the main goals and functions of an OS.
- Identify the types of operating systems and their use cases.
- Explore user vs. system views of an OS.
- Introduce examples of modern operating systems.

What is an Operating System?

- An **Operating System (OS)** is a software layer that acts as an intermediary between computer hardware and users, managing resources and providing an interface for executing programs.
- It makes the hardware usable by abstracting the complexity of the system, allowing users and applications to interact with it easily.



Functions of an Operating System

- **Resource Management:** Manages hardware resources like CPU, memory, storage, and I/O devices.
- **Task Scheduling:** Determines which process gets to use the CPU and for how long.
- **File Management:** Manages the organization, storage, retrieval, naming, sharing, and protection of files.
- **Security and Protection:** Ensures that unauthorized access and operations are prevented.
- **User Interface:** Provides an interface for user interaction, such as Command Line Interface (CLI) or Graphical User Interface (GUI).

OS as a Resource Allocator

- **Resource Manager:** The OS allocates resources such as CPU, memory, and disk space to various programs or users efficiently.
- **CPU Scheduling:** Determines which process will use the CPU when multiple processes are ready to run.
- **Memory Management:** Keeps track of every byte of memory and how it is allocated.
- **Disk Management:** Manages data storage on hard drives and SSDs.

OS as a Control Program

- **Managing Execution:** Controls the execution of programs to prevent errors and ensure proper use of system resources.
- **I/O Device Management:** Handles input and output operations by managing hardware devices like keyboards, printers, and monitors.
- **Protection Mechanisms:** Enforces security policies and access controls to protect resources from unauthorized use.

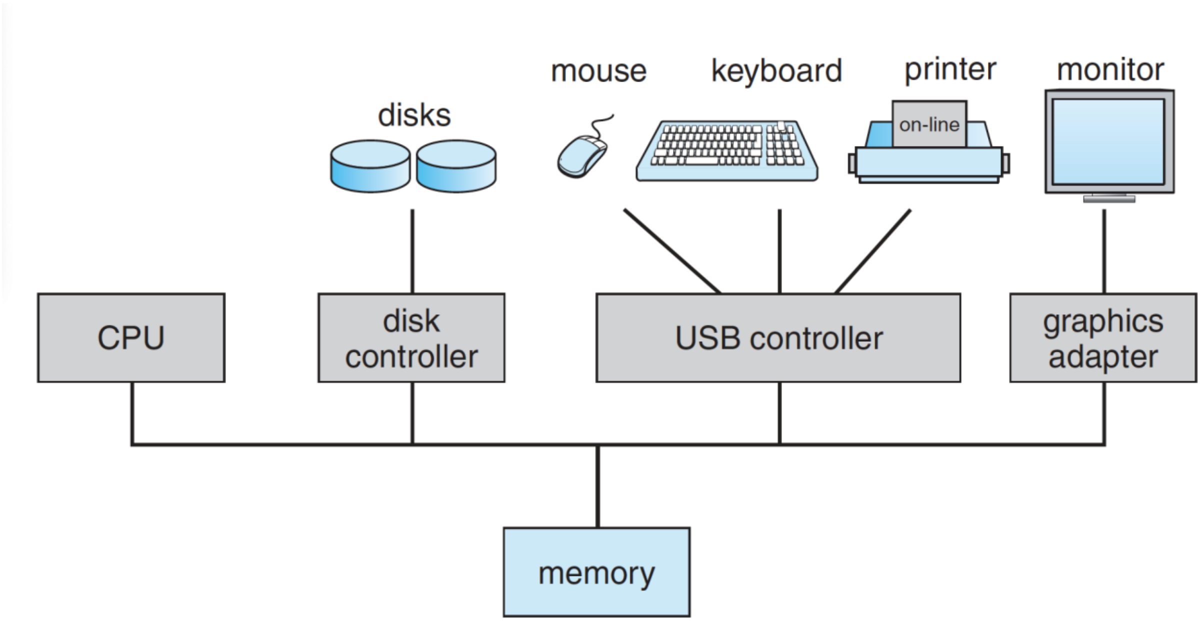
Computer Startup

- **Bootstrap Program:**
    - A small essential program loaded at power-up or reboot. Responsible for initiating the boot process. Typically resides in ROM (Read-Only Memory) or EPROM (Erasable Programmable Read-Only Memory).
- **System Initialization:**
    - Configures and initializes key hardware components, including CPU registers, memory, and I/O devices. Prepares the system for normal operations by setting up the environment.
- **Kernel Loading:**
    - The bootstrap program locates the operating system kernel on disk, loads it into memory, and begins execution. Once loaded, the kernel takes control of the system and manages resources.

Computer-System Operation

- **System Components:**
  - Consists of one or more CPUs and multiple device controllers connected through a common bus. The bus provides access to shared memory for all components.
- **Concurrent Execution:**
  - CPUs and I/O devices can execute operations simultaneously, competing for memory access.
- **Device Controllers:**
  - Each controller manages a specific type of device (e.g., disk drive, keyboard).
- **Data Movement:**
  - The CPU transfers data between the main memory and the local buffer of each device controller.

Computer-System Operation



- **I/O Process:**
  - Data from I/O devices is first moved to the local buffer of the device controller.
- **Interrupt Handling:**
  - Once an I/O operation is complete, the device controller signals the CPU by generating an interrupt, notifying it to continue processing.

Common Functions of Interrupts

- **Interrupt Handling**:
  - Control is transferred to the **Interrupt Service Routine (ISR)**, typically through an **interrupt vector**, which stores the addresses of all service routines.
- **Saving State**:
  - The interrupt architecture saves the address of the currently executing instruction to ensure the system can return to its original state after the interrupt is handled.
- **Traps and Exceptions**:
  - A **trap** or **exception** is a software-generated interrupt, triggered either by an error (e.g., divide by zero) or a user-initiated request (e.g., I/O operation).
- **Interrupt-Driven OS**:
  - Operating systems are **interrupt-driven**, meaning they respond to hardware and software interrupts to efficiently manage tasks and resources.

Interrupt Handling Process

- **Preserving CPU State:**
  - When an interrupt occurs, the operating system saves the current state of the CPU by storing the contents of registers and the **program counter**.
- **Identifying the Interrupt:**
  - The OS determines the type of interrupt that has occurred using one of two methods:
    - **Polling:** The CPU checks each device to identify the source of the interrupt.
    - **Vectored Interrupt System:** The device automatically sends the address of its interrupt handler, allowing direct access to the correct service routine.
- **Executing Interrupt-Specific Actions:**
  - The OS uses separate blocks of code (interrupt service routines) for each type of interrupt to define the appropriate action to take.

Storage Structure

- **Main Memory (RAM):**
  - The primary storage medium that the CPU can directly access.
    - **Random Access:** Data can be read or written in any order.
    - **Volatile:** Loses data when the system is powered off.
- **Secondary Storage:**
  - Serves as an extension of main memory, offering large, **nonvolatile** storage capacity.
  - Retains data even when the system is powered down.

Storage Structure

Hard Disks (HDDs):

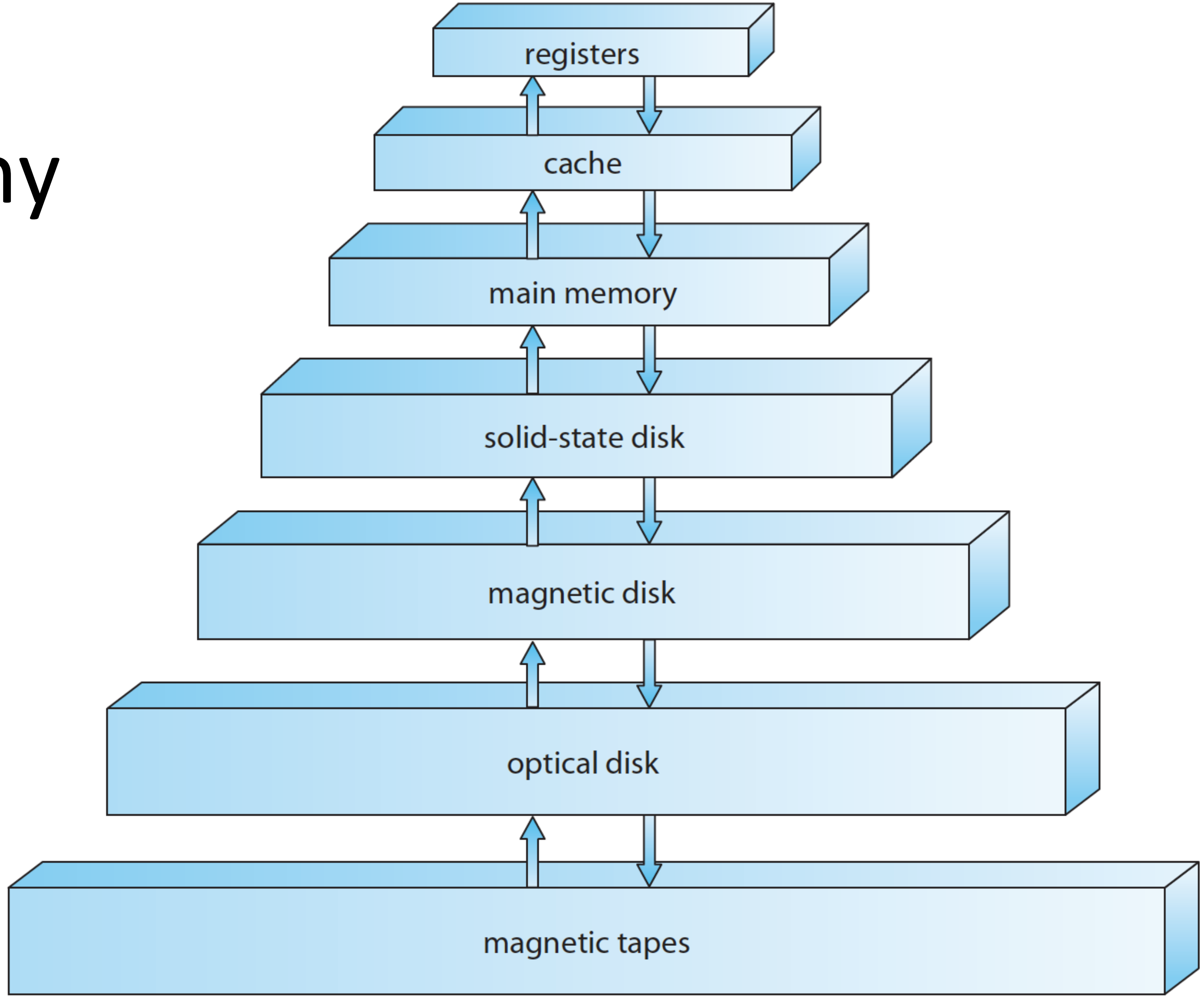
- Composed of rigid metal or glass platters coated with magnetic material.
- Data is organized into **tracks**, further divided into **sectors.**
- The **disk controller** manages communication between the device and the computer.

- **Solid-State Drives (SSDs):**
  - Nonvolatile storage devices that are **faster** than traditional hard disks.
  - Based on flash memory technologies.
  - Increasingly popular due to improved performance and reliability.

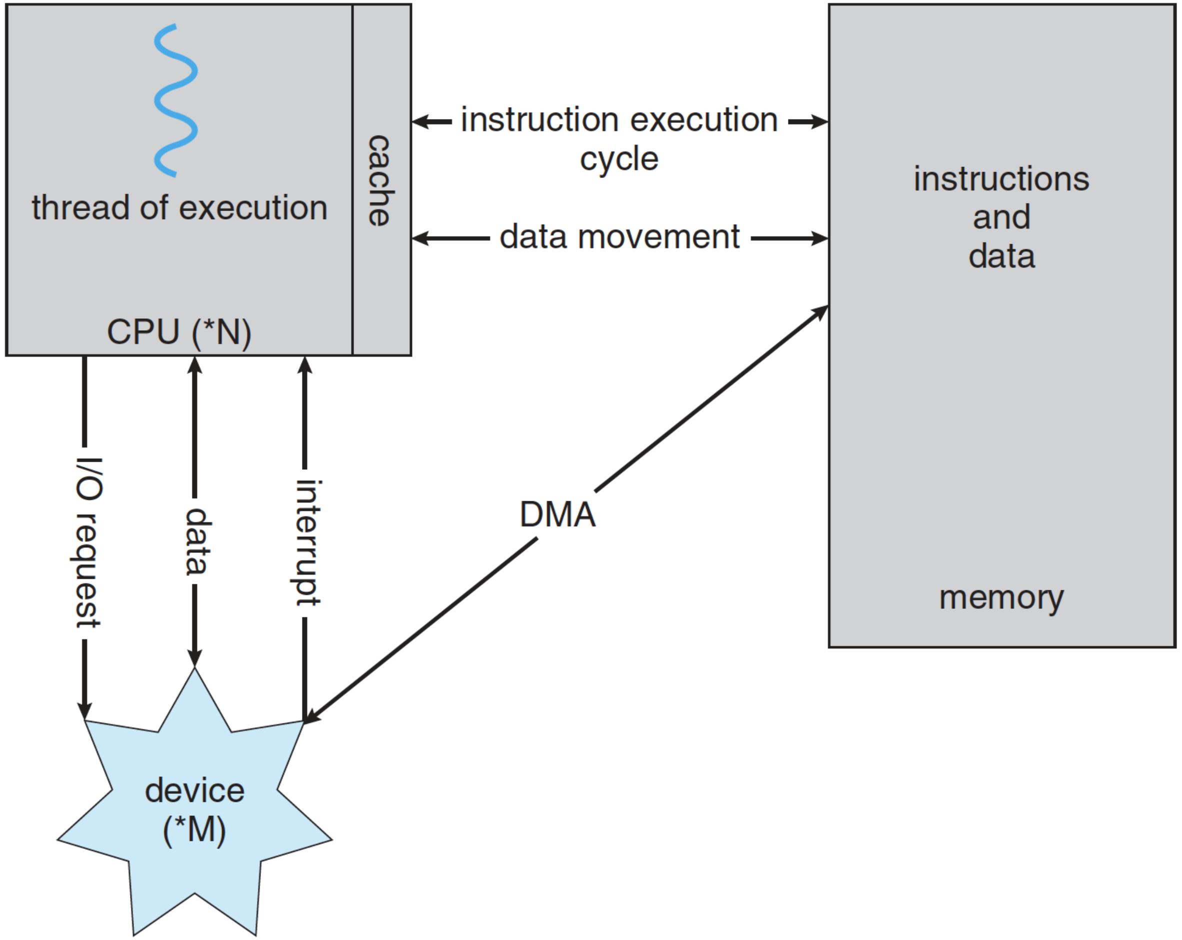
Storage Hierarchy

- **Organized by:**
  - **Speed**: Faster storage at the top (e.g., CPU registers, cache) and slower storage at the bottom (e.g., HDDs, tapes).
  - **Cost**: Higher speed storage is more expensive; slower storage is more affordable.
  - **Volatility**: Higher levels (e.g., RAM) are volatile, while lower levels (e.g., HDDs, SSDs) are nonvolatile.
- **Caching**:
  - Process of storing frequently accessed data in faster storage (cache) for quicker retrieval.
  - **Main Memory** can act as a cache for slower, secondary storage (e.g., hard disks).
- **Device Drivers**:
  - Software responsible for managing I/O between the **device controller** and the operating system.
  - Provides a consistent interface for the OS to communicate with hardware devices.

Storage Hierarchy



How a Modern Computer Works



How a Modern Computer Works

Single General-Purpose Processor:

- Most systems are built around a single processor for general computing tasks. However, many systems also include **special-purpose processors** to handle specific tasks (e.g., GPUs for graphics).

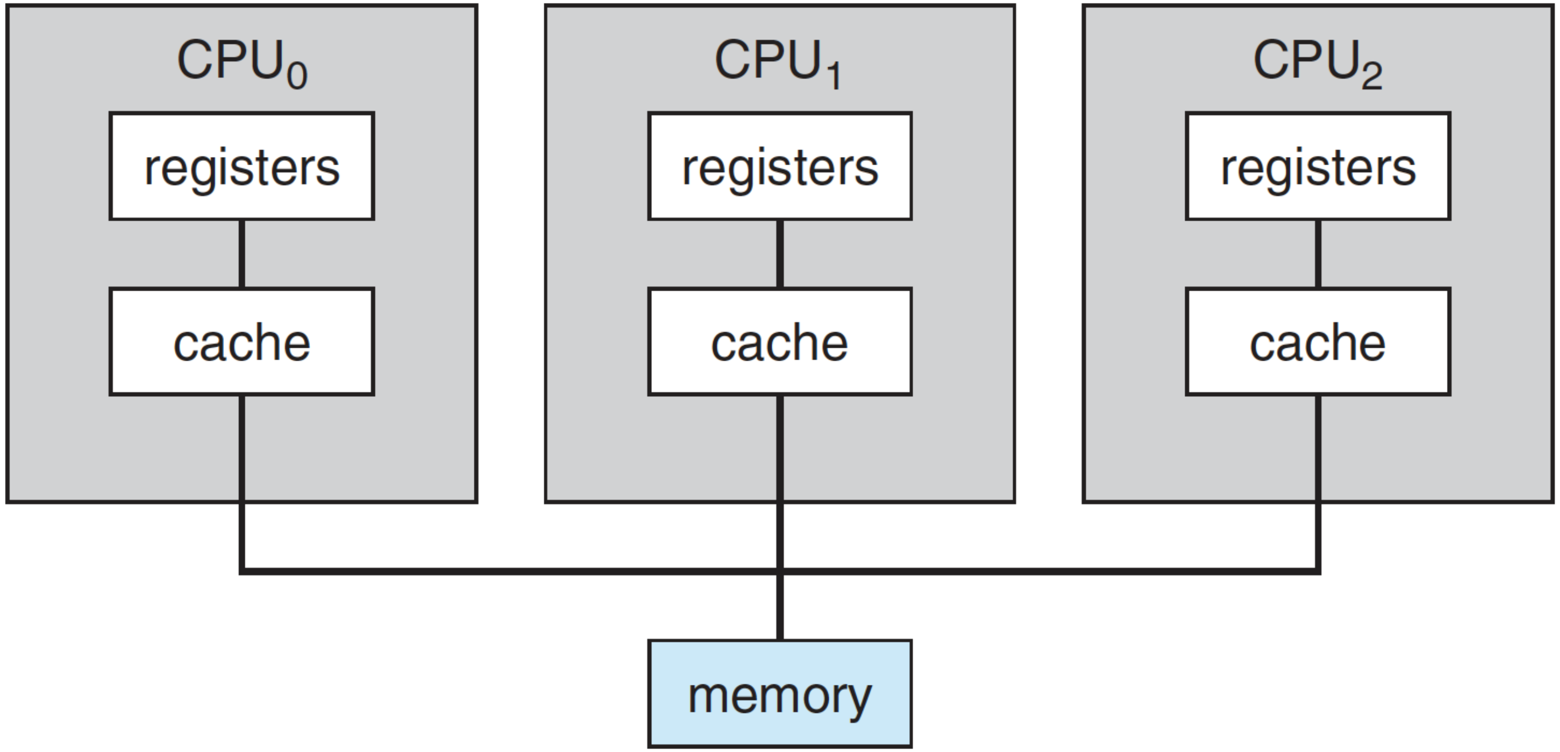
Multiprocessor Systems:

- Increasingly common and important in modern computing. Also called **parallel systems**, where multiple processors work together.
- **Advantages**:
    - **Increased Throughput**: More tasks can be processed simultaneously.
    - **Economy of Scale**: More cost-effective use of resources.
    - **Increased Reliability**: Fault tolerance through **graceful degradation**—the system continues to operate even if one processor fails.

Types of Multiprocessing:

1. **Asymmetric Multiprocessing**: Each processor is assigned a specific task or role.
2. **Symmetric Multiprocessing**: All processors perform the same tasks and share the workload.

Symmetric multiprocessing architecture



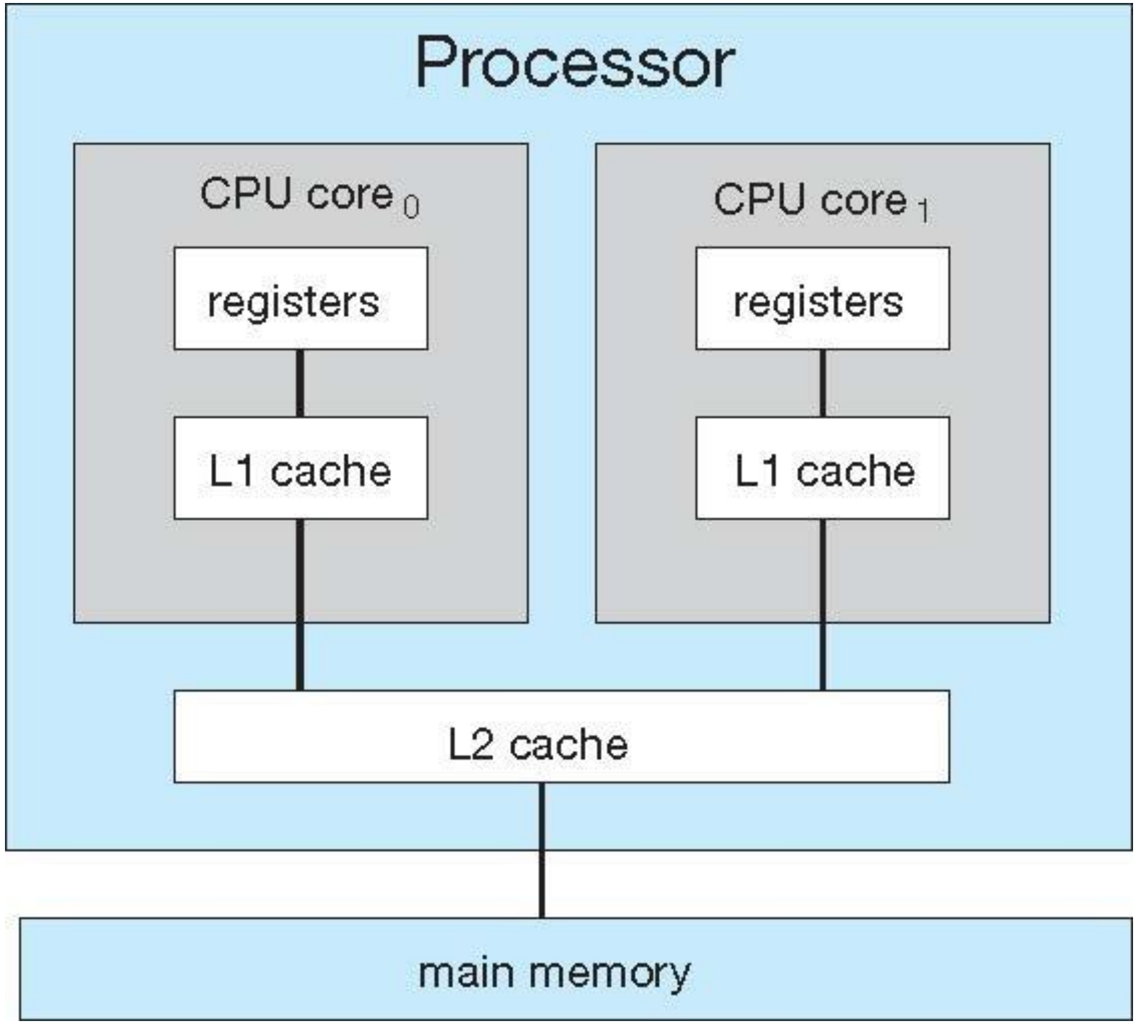
Multi-Chip or Multiprocessing

- **Multi-Chip Systems**:
  - Consist of multiple **separate processors (CPUs)** on different chips.
- **Features**:
  - Processors communicate through a shared bus or network.
  - Typically used in **high-performance computing** and servers.
  - Offers increased computational power but may have higher power consumption and slower inter-processor communication compared to multicore.

Multicore Systems

- Multiple **processing cores** integrated on a single chip (e.g., dual-core, quad-core).
  - **Advantages**:
    - **Faster Communication:** Cores communicate more efficiently within the same chip, reducing latency.
    - **Power Efficiency**: Consumes less power compared to multi-chip systems for similar performance.
    - **Improved Performance:** Enables better multitasking and parallel processing for general-purpose and mobile devices.

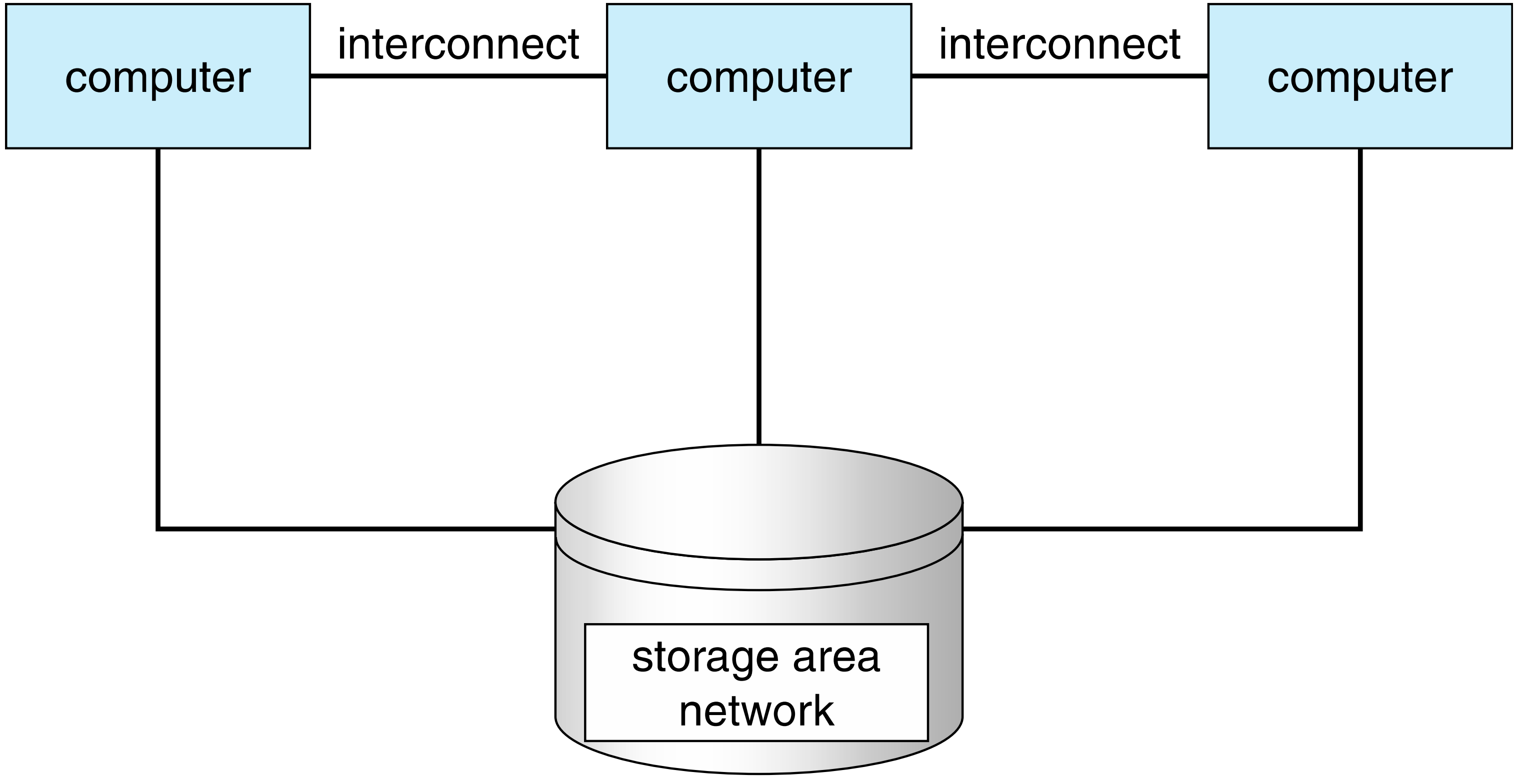
Multicore Systems Architecture



Clustered Systems

- Similar to multiprocessor systems, but consists of multiple **independent systems** working together, often connected via a high-speed network.
- **Shared Storage**:
  - Systems typically share storage through a **Storage-Area Network (SAN)**, allowing multiple nodes to access the same data efficiently.
- **High Availability**:
  - Provides **fault tolerance** and **high availability** by continuing operations even when one system fails.
- **High-Performance Computing (HPC)**:
  - Tasks are parallelized to achieve faster computation. Applications must be specially written to support parallel processing.
- **Distributed Lock Manager (DLM)**:
  - In some clusters, a **DLM** is used to manage access to shared resources, preventing conflicts during concurrent operations.

Clustered Systems Structure



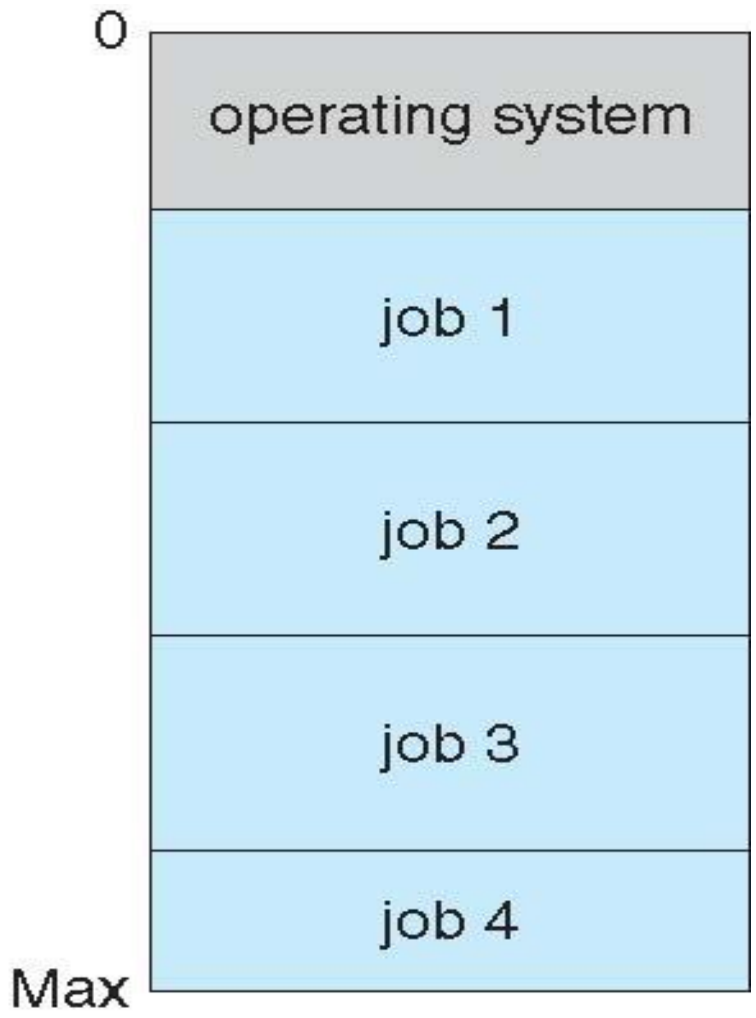
Multiprogramming (Batch System)

- **Multiprogramming** Improves efficiency by keeping the CPU and I/O devices busy.
  - **Single User Limitation:** A single user cannot utilize the CPU and I/O devices at all times.
  - **Job Organization:** Multiple jobs (programs) are organized so that the CPU always has a job to execute.
  - **Job Scheduling:** A subset of jobs is kept in memory, and the OS selects one job to run. If the job is waiting (e.g., for I/O), the OS switches to another job.

Timesharing (Multitasking)

- A logical extension of multiprogramming where the CPU switches between jobs frequently, allowing multiple users to interact with their jobs while they run.
- **Interactive Computing**: Users can interact with running programs, and response times are typically less than 1 second.
- **Multiple Programs:** Each user has at least one program (process) running in memory.
- **CPU Scheduling:** If several jobs are ready, the OS schedules CPU time for each job.
- **Swapping**: If processes exceed available memory, the OS swaps them in and out of memory to ensure all jobs run.
- **Virtual Memory**: Enables execution of processes that don’t fit entirely in memory by storing parts of the process on disk.

Memory Layout for Multiprogramming System



Modes of Operation

- **Dual-Mode Operation** enables the OS to protect itself and other system components by distinguishing between user and system-level operations.
- **The Dual-Mode are:**
  - **User Mode**: For running user applications.
  - **Kernel Mode**: For running critical system tasks and accessing hardware directly.

Modes of Operation

- **Mode Bit:**
  - A hardware-provided bit that indicates whether the CPU is in user mode or kernel mode.
  - **Privileged Instructions:** Certain critical instructions can only be executed in kernel mode to prevent unauthorized access or misuse.
- **System Calls:**
  - When a user program requests an OS service (e.g., I/O), a system call is made, switching the CPU to **kernel mode.**
  - After the system call is completed, the mode is reset back to **user mode.**

Process Management

- Process: A process is a program in execution, representing a unit of work in the system. While a program is passive, a process is an active entity.
- Resources: To complete its task, a process requires resources such as:
  - CPU, memory, I/O, files, etc.
  - Initialization data.
- Process Termination: When a process terminates, any reusable resources must be reclaimed.

Thread

- **Thread**: The smallest unit of CPU utilization within a process.
- **Single-threaded Process**: Executes instructions sequentially, one at a time, with a single program counter tracking the next instruction.
- **Multi-threaded Process**: Contains multiple threads, each with its own program counter.
- **System Overview**: Typically, multiple processes, including both user and operating system processes, run concurrently on one or more CPUs.

Thank you