# Operating System Concepts (cont'd)

Course Code: CSC 2209 Course Title: Operating Systems


Dept. of Computer Science
Faculty of Science and Technology


| | | | | | |
|---|---|---|---|---|---|
| **Lecturer No:** | **02** | **Week No:** | **02** | **Semester:** | **Fall 2024-25** |
| **Assistant Professor** | *Dr. Rajarshi Roy Chowdhury, rajarshi@aiub.edu* | | | | |

# Lecture Outline

1. I/O Structure
2. Storage Structure
3. How a Modern Computer Works
4. Direct Memory Access Structure
5. Computer-System Architecture
6. A Dual-Core Design
7. Clustered Systems
8. Operating-System Operations
9. Multiprogramming and Multitasking
10. Dual-mode and Multimode Operation
11. Process Management
12. Memory Management
13. File-system Management
14. Caching

# 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 OS to allow user to wait for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into **I/O device table** to determine device status and to modify table entry to include interrupt

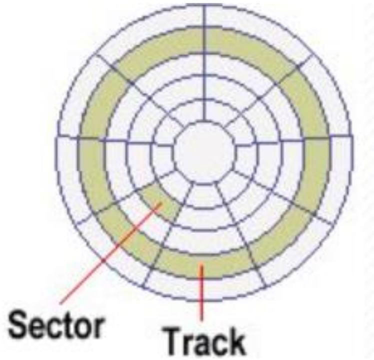
# I/O Structure

## Device-status table

| Device ID | Device Type | Address | State |
|---|---|---|---|
| 0 | Disk Drive | 0x1000 | Busy |
| 1 | Printer | 0x1001 | Idle |
| 2 | Keyboard | 0x1002 | Ready |
| 3 | Network Card | 0x1003 | Error |

# Storage Structure

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

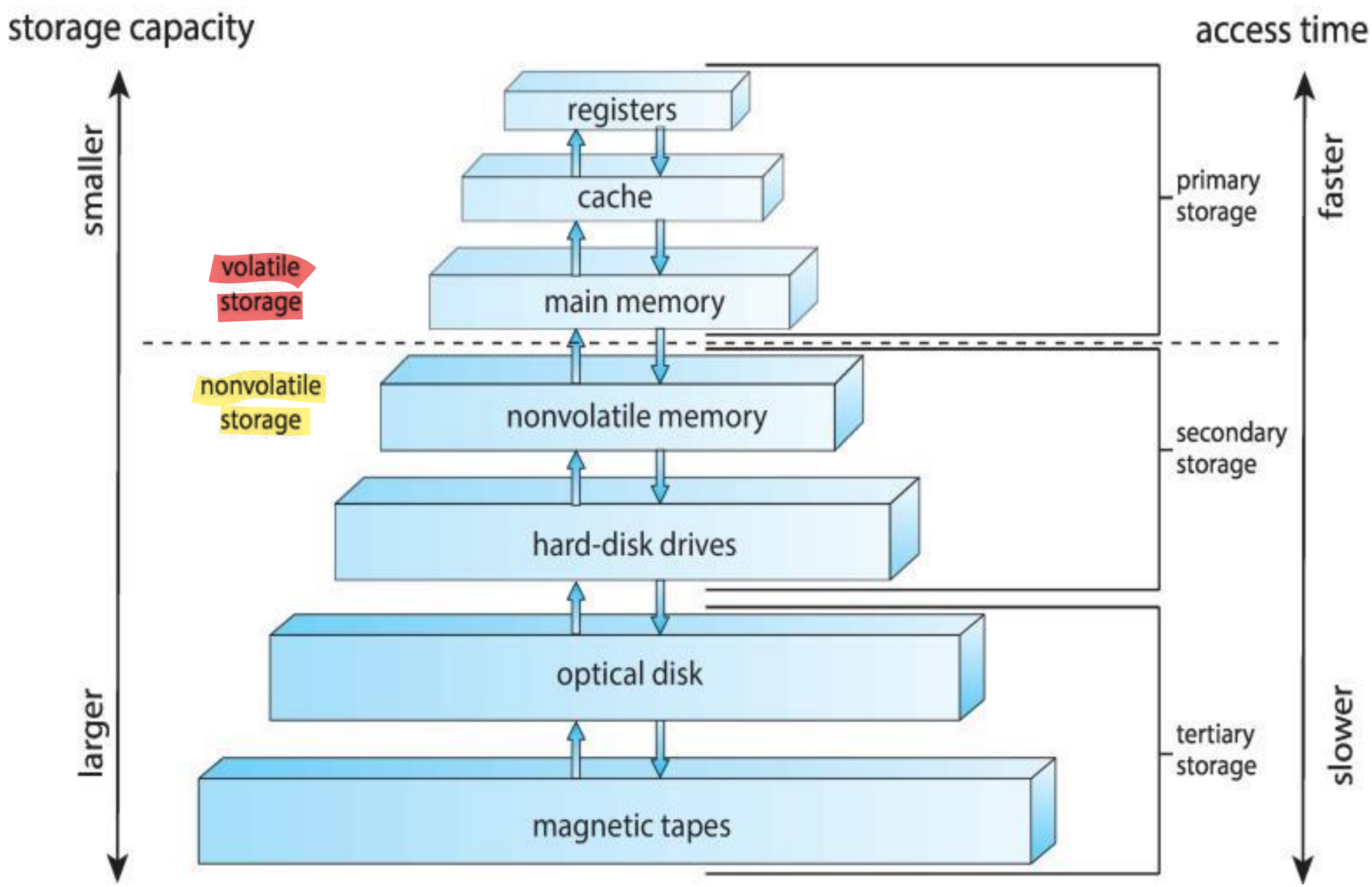



- **Non-volatile memory (NVM)** devices– faster than hard disks,
  - Various technologies
  - Becoming more popular as capacity and performance increases, price drops

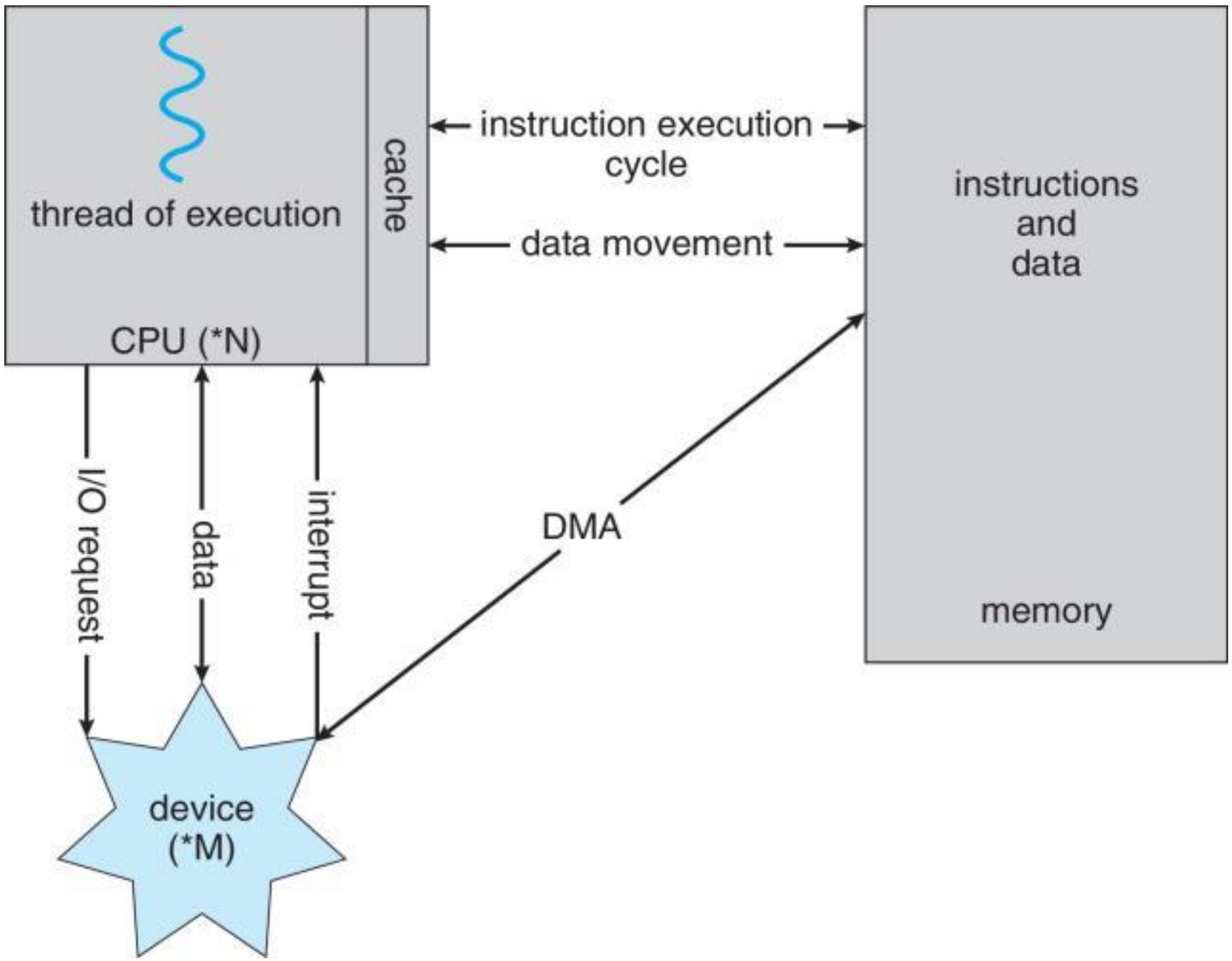
# Storage Hierarchy

- 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-Device Hierarchy



# How a Modern Computer Works



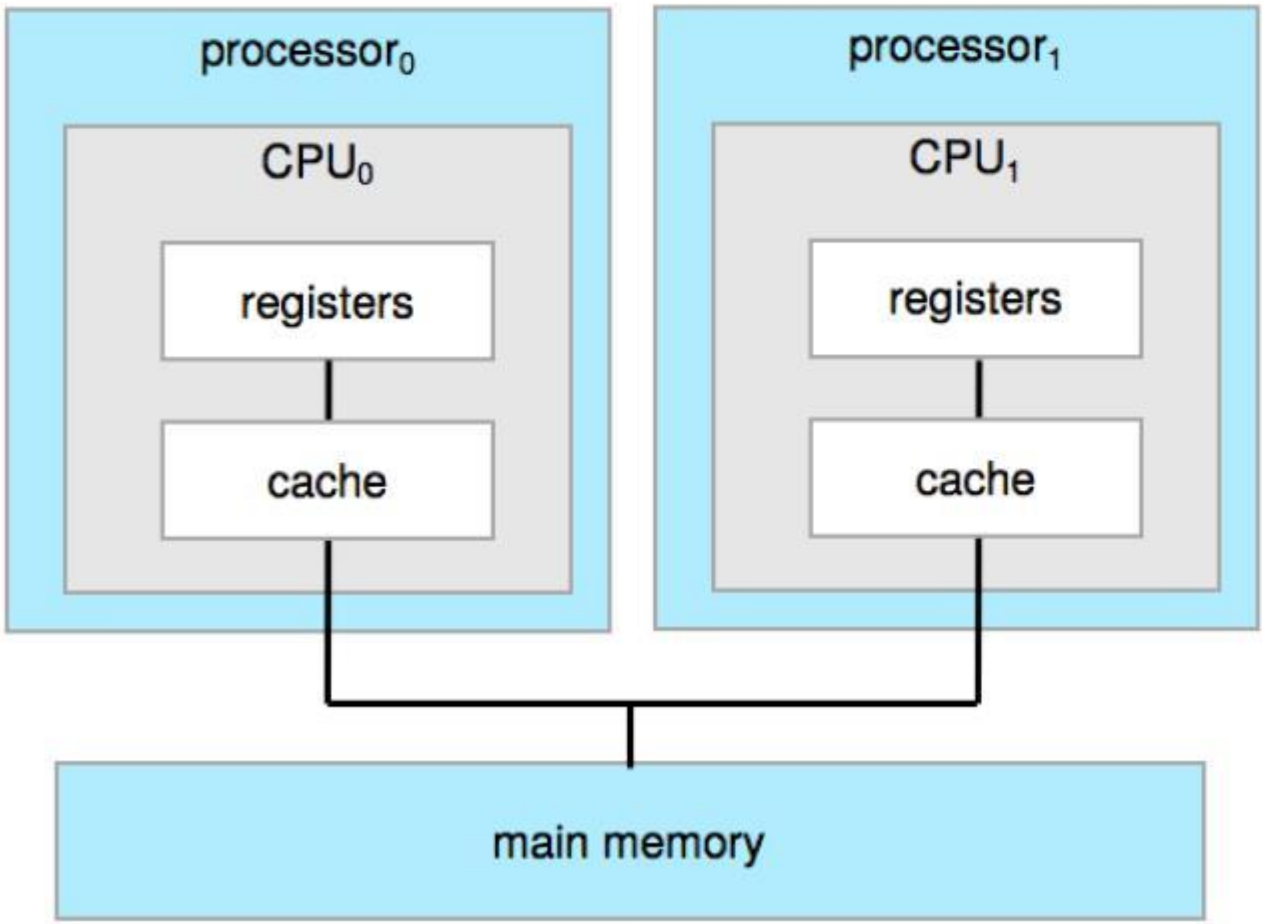
# Direct Memory Access (DMA) Structure

- 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-System Architecture

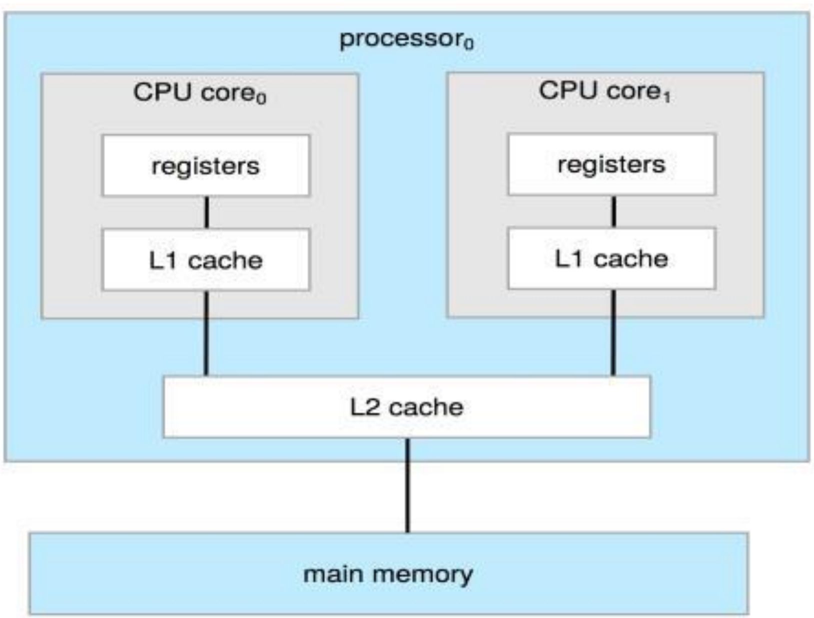
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2.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

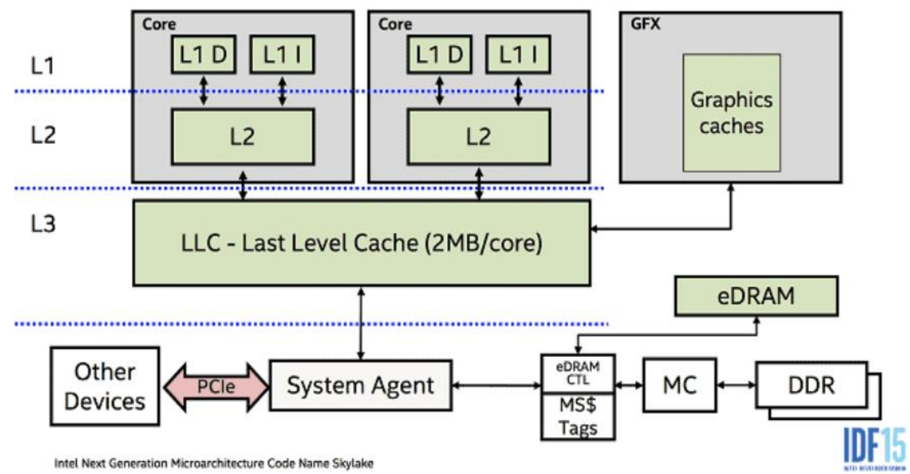
# Symmetric Multiprocessing Architecture



# A Dual-Core Design

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



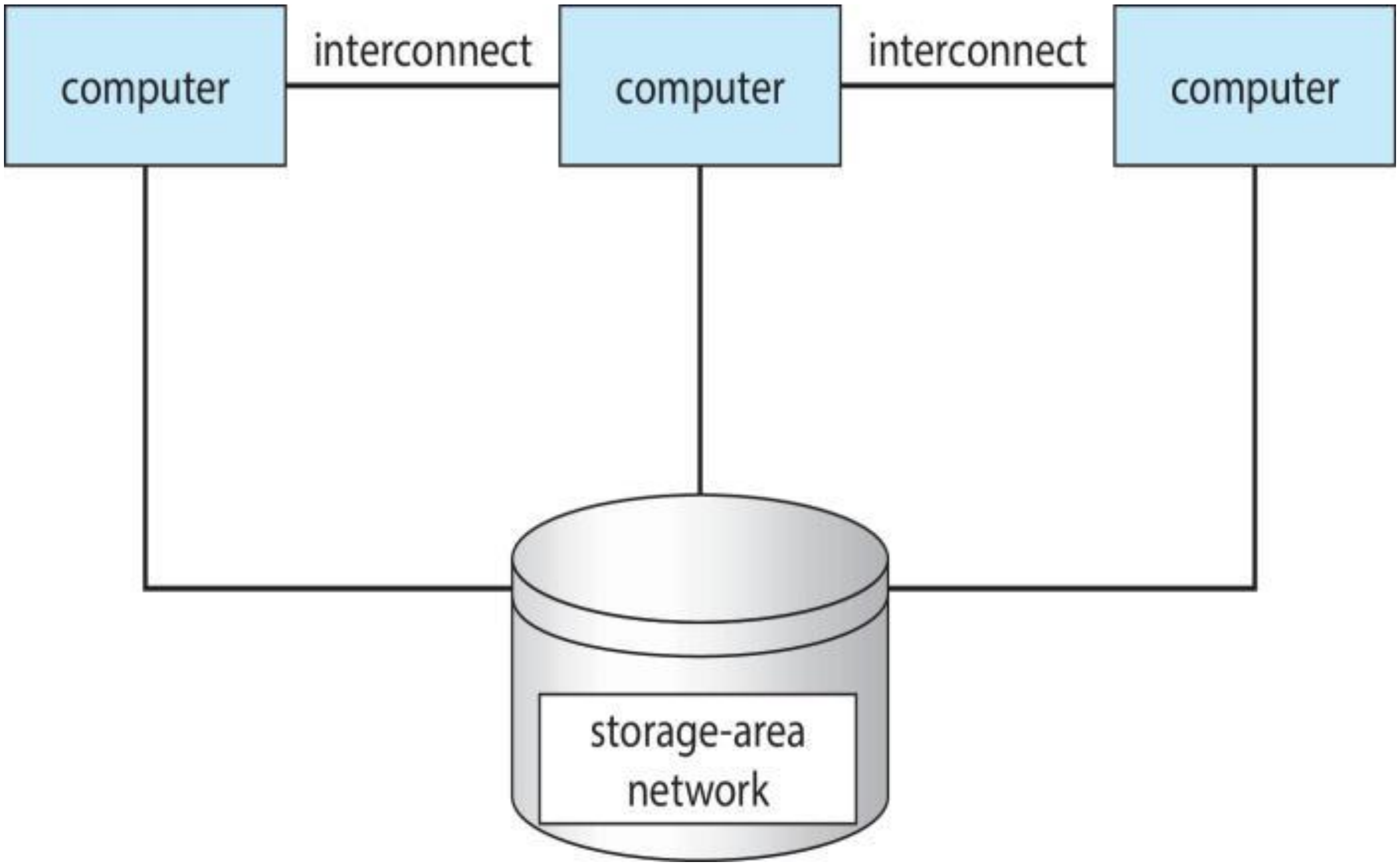




| Cache Level | Speed | Size | Volatility | Purpose |
|---|---|---|---|---|
| **L1 Cache** | Fastest | 16 KB - 128 KB | Volatile | Immediate access to data and instructions |
| **L2 Cache** | Fast | 256 KB - 2 MB | Volatile | Buffer between L1 and L3/main memory |
| **L3 Cache** | Moderate | 2 MB - 32 MB+ | Volatile | Shared cache for multi-core CPUs |

# 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



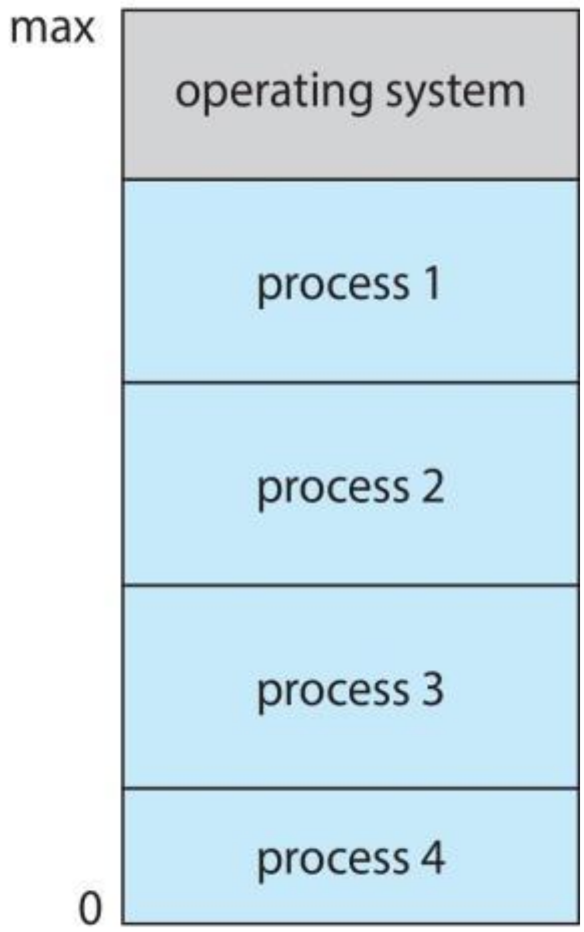
# Operating-System Operations

- 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 and Multitasking

- **Multiprogramming** (**Batch system**) 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 doesn'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



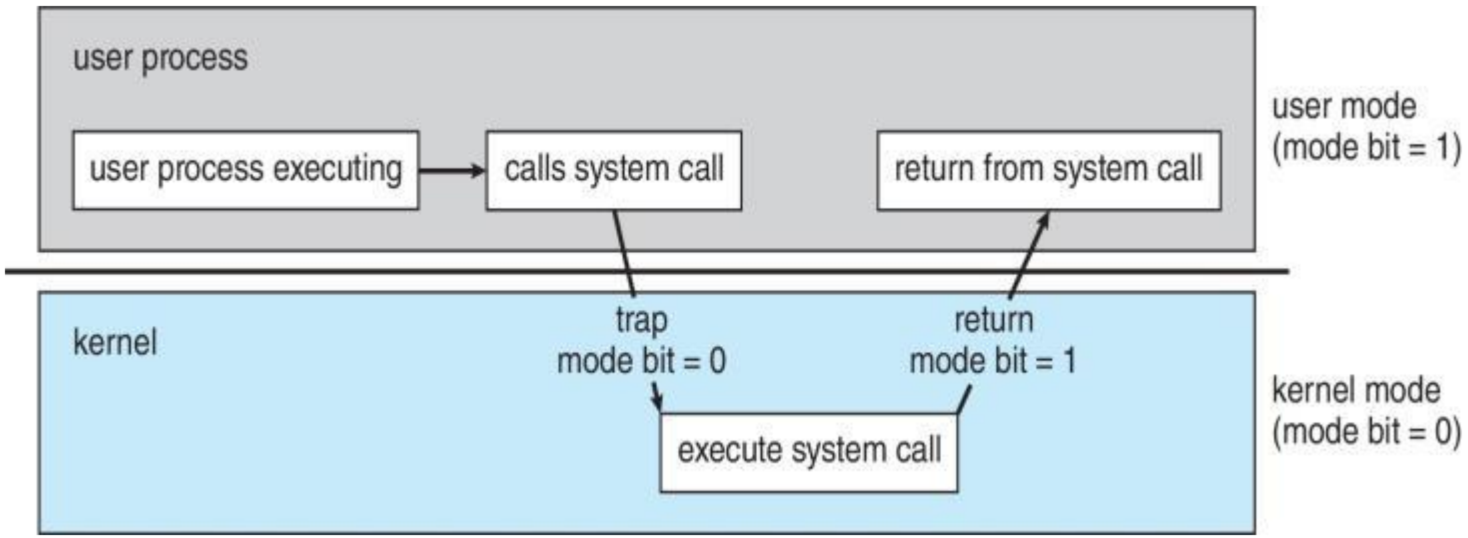
# Dual-mode and Multimode Operation

- **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
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

# Transition from User to Kernel Mode

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

# Transition from User to Kernel Mode



# 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

- 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

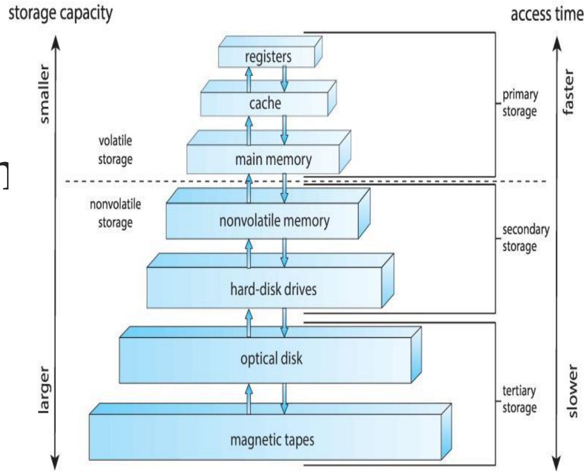
# File-system Management

- 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

# 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



## Books

- Operating Systems Concept
  - Written by Galvin and Silberschatz
  - Edition: 9th

## Caching

Caching is a crucial technique used at various levels in a computer system to temporarily store frequently accessed data in faster storage for quicker access. It involves checking the cache first for data, and if not found, copying it from slower storage to the cache. Effective cache management, including size and replacement policies, is essential for optimal performance.

# References

- Operating Systems Concept
  - Written by Galvin and Silberschatz
  - Edition: 9th