BCA233: Operating System

Unit – 1

Introduction

Class policies and Guidelines

- **KINDLY LOG IN WITH YOUR CHRIST UNIVERSITY EMAIL (NOT PERSONAL EMAIL)**
- Please mute your microphones during the class sessions
- If you are joining the meeting late or in between, join the meeting after muting your microphone and video
- If you would like to say something or ask a question please use the chat box
- Switching off your video will help with better connectivity for all
- Strictly adhere to the class policies of the University / Department

Course Objectives

- This course is an introduction to the concepts behind modern computer operating systems. Topics will include what an operating system does (and doesn't) do, system calls and interfaces, processes, resource scheduling and management (of the CPU, memory, etc.), virtual memory.

Objectives of the course are

- To acquire the fundamental knowledge of the operating system architecture and its components
- To know the various operations performed by the operating system.

Course Outcomes

CO1: Demonstrate the fundamentals of an operating system.

CO2: Evaluate the importance of process and scheduling.

CO3: Analyze and Solve the issues in synchronization and memory management.

Unit-1 Teaching Hours: 8

Introduction and System Structures

Operating System Fundamentals; Computer System organization and architecture; Operating System structure and operations; Basics of process, memory and storage management and protection and security; Operating System services; User interface; System calls; System programs; Operating System structure; System boot.

Unit-2 Teaching Hours: 12

Process Management

Process concept; Process scheduling; Operations on processes; Inter Process Communication; Overview of Threads; Multi-threading models; Threading issues

Unit-3 **Teaching Hours: 12**

Process Synchronization

Need of synchronization; Critical section problems; Peterson‘s solution; Synchronization hardware; Mutex Locks; Semaphores, Classical problems of synchronization, Synchronization examples, Thread synchronization using mutex and semaphore.

Unit-4 **Teaching Hours: 6**

CPU Scheduling

CPU Scheduling concepts; Scheduling criteria; Scheduling algorithms; Overview of thread scheduling; Multi-processor scheduling

Unit-5 Teaching Hours: 12

Memory Management

Overview; Swapping; Memory allocation; Segmentation; Paging, Structure of the page table

Unit-6 Teaching Hours: 10

Virtual Memory

Overview; Demand paging; Copy on Write; Page replacement; Allocation of Frames; Thrashing

Text Books and Reference Books:

[1] Silberschatz, P.B. Galvin and G. Gagne, Operating System Concepts. 9th Edition, New Delhi: Wiley India, 2011.

Essential Reading/Recommended Reading

[1] Stalling William, Operating Systems: Internals and Design Principles. 7th Edition, Prentice Hall, 2011.

[2] Dietel et al, Operating System.3rd Edition. Pearson Education, 2004.

[3] A.S. Tanenbaum, Modern Operating Systems.3rd Ed, Prentice Hall, 2007.

Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security

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

Four Components of a Computer System

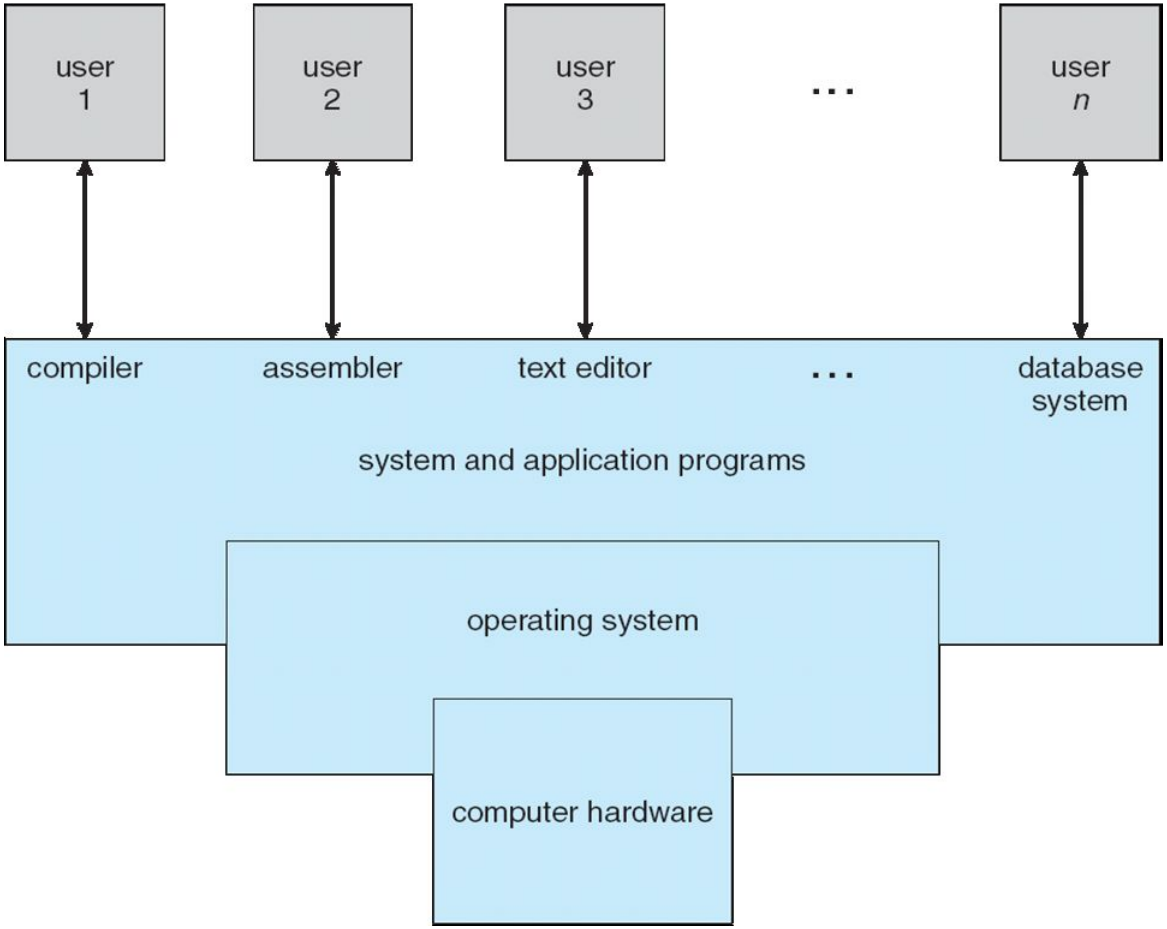


Figure 1.1 Abstract view of the components of a computer system

What Operating Systems Do

- Depends on the point of view

User 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
- Users of dedicated systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

What Operating Systems Do

System View:

- In this context, we can view an operating system as a **resource allocator**.
- A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on.
- An operating system is a control program. A **control program** manages the execution of user programs to prevent errors and improper use of the computer.

Operating System Definition

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

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
  - For a computer to start running—for instance, when it is powered up or rebooted—it needs to have an initial program to run.
  - This initial program, or bootstrap program, stored within the computer hardware in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM), known by the general term firmware.

Computer System Organization

- Bootstrap program initializes all aspects of the system, from CPU registers to device controllers to memory contents.
- The bootstrap program load the operating system and start executing that system.
- To accomplish this goal, the bootstrap program must locate the operating-system kernel and load it into memory.
- Once the kernel is loaded and executing, it can start providing services to the system and its users.

Computer System Organization

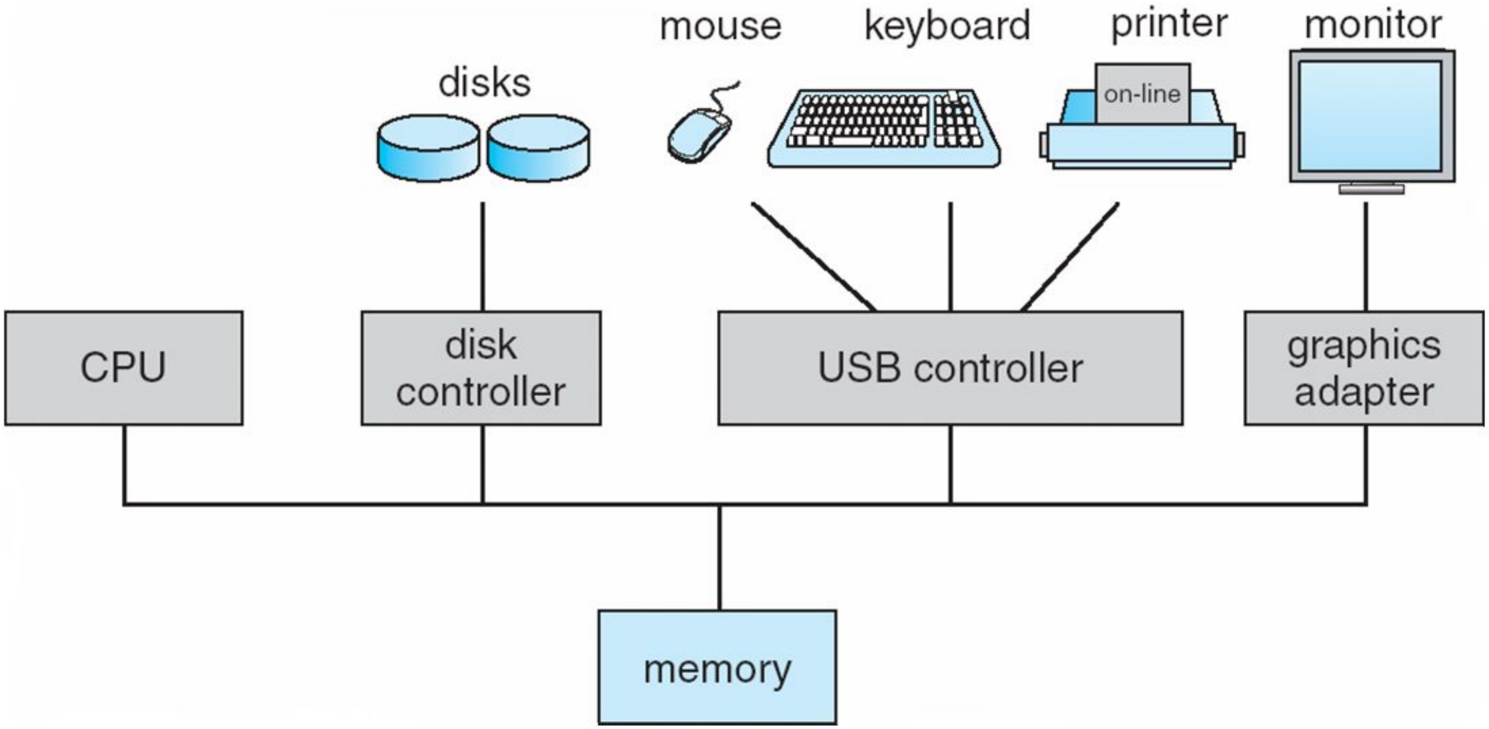


Figure: A modern computer system.

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

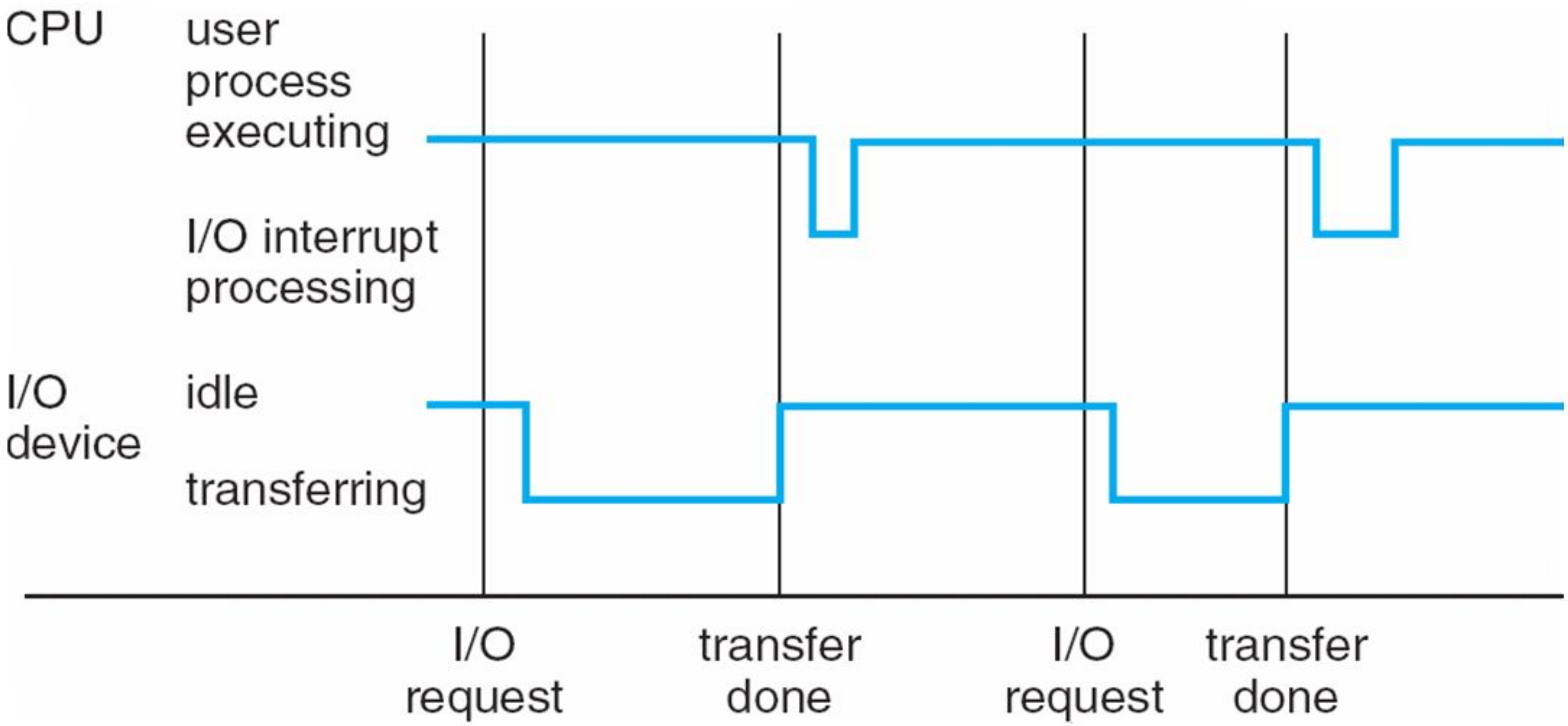
Common Functions of Interrupts

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

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

Interrupt Timeline



I/O Structure

Interrupt-driven I/O

- To start an I/O operation, the device driver loads the appropriate registers within the device controller.
- The device controller, in turn, examines the contents of these registers to determine what action to take (such as "read a character from the keyboard").
- The controller starts the transfer of data from the device to its local buffer. Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation.
- The device driver then returns control to the operating system, possibly returning the data or a pointer to the data if the operation was a read.

I/O Structure

Direct Memory Access

- Interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement such as disk I/O.
- To solve this problem, **direct memory access (DMA)** is used.
- After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with no intervention by the CPU.
- While the device controller is performing these operations, the CPU is available to accomplish other work.

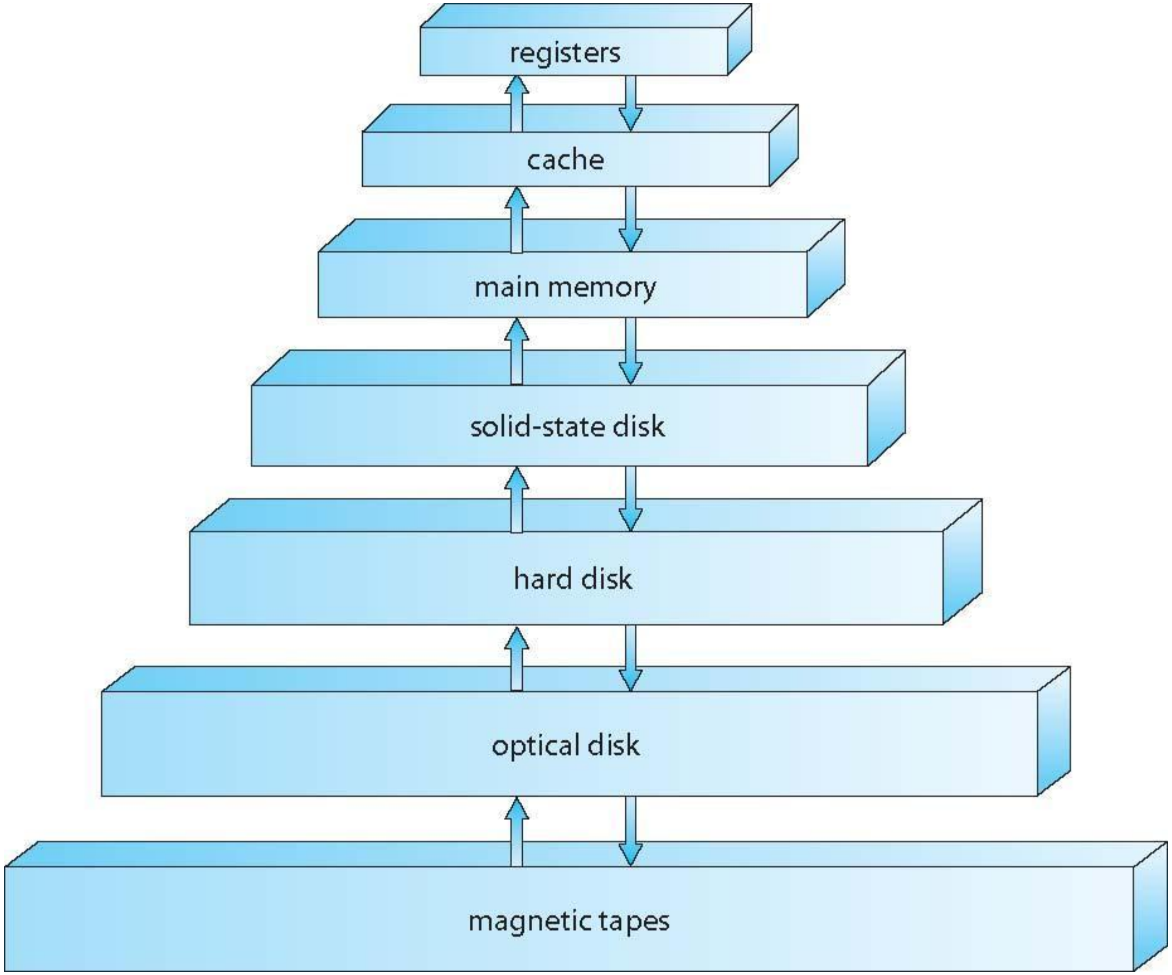
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Hard 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
- **Solid-state disks** – faster than hard disks, nonvolatile

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

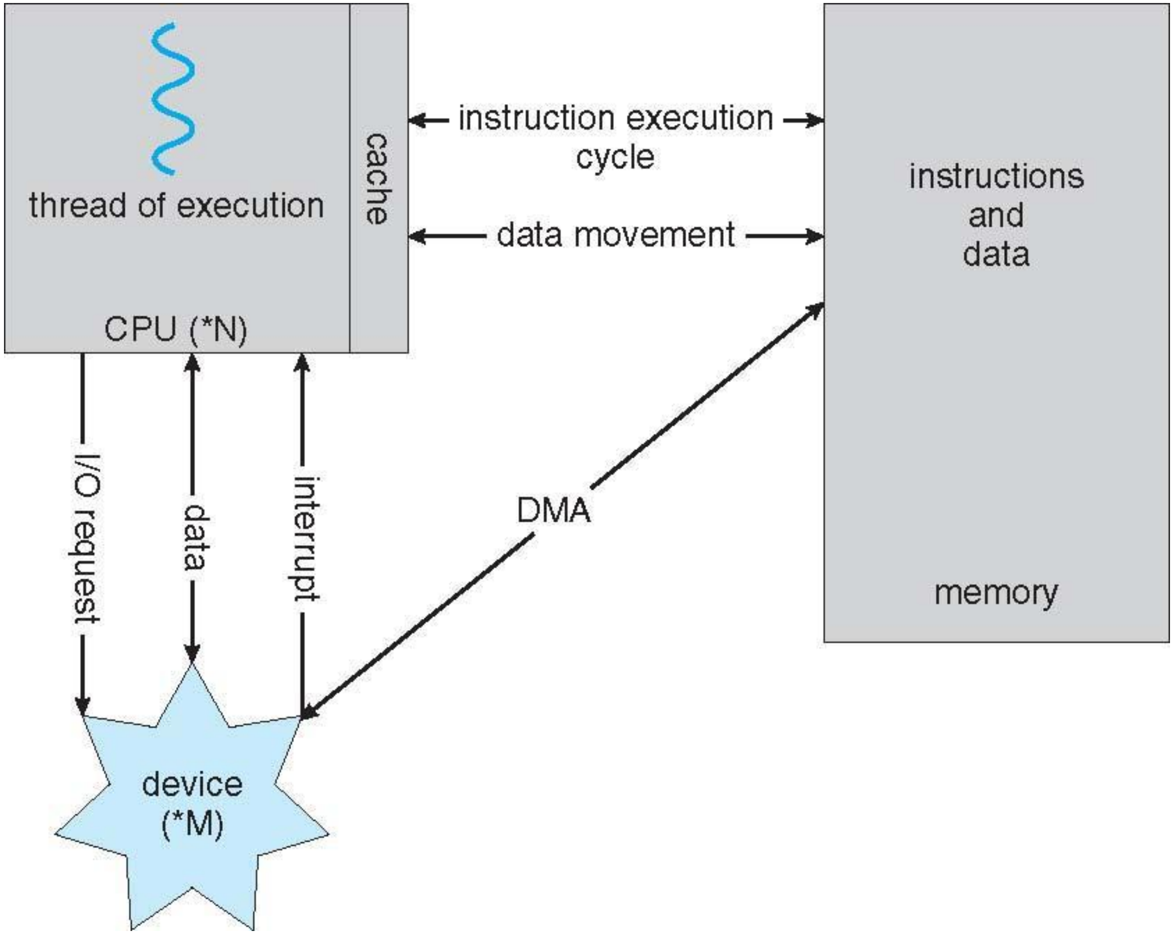
Storage-Device Hierarchy



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

How a Modern Computer Works



A von Neumann architecture

Computer-System Architecture

- **Single-Processor Systems**
  - Most systems use a single general-purpose processor
  - On a single processor system, there is one main CPU capable of executing a general-purpose instruction set, including instructions from user processes.
  - 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:

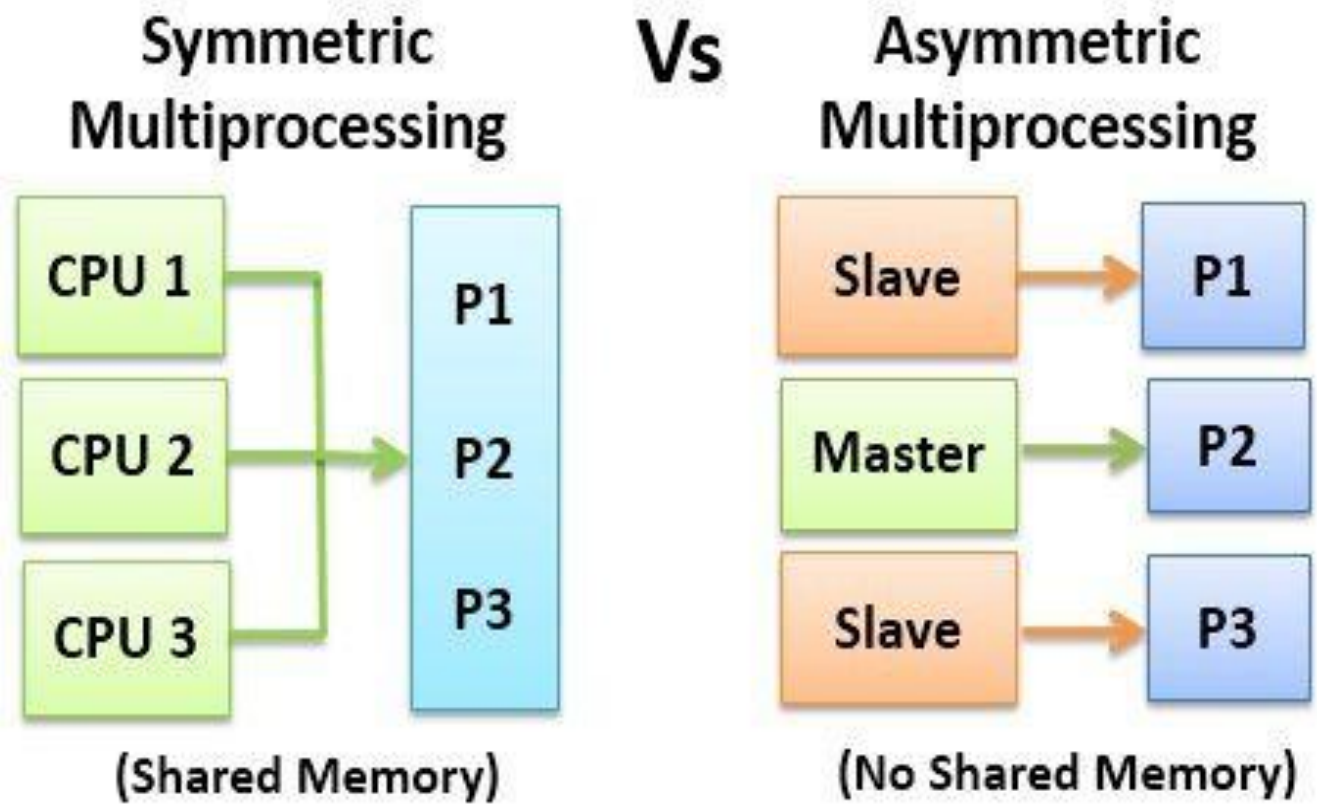
    Increased throughput

    Economy of scale

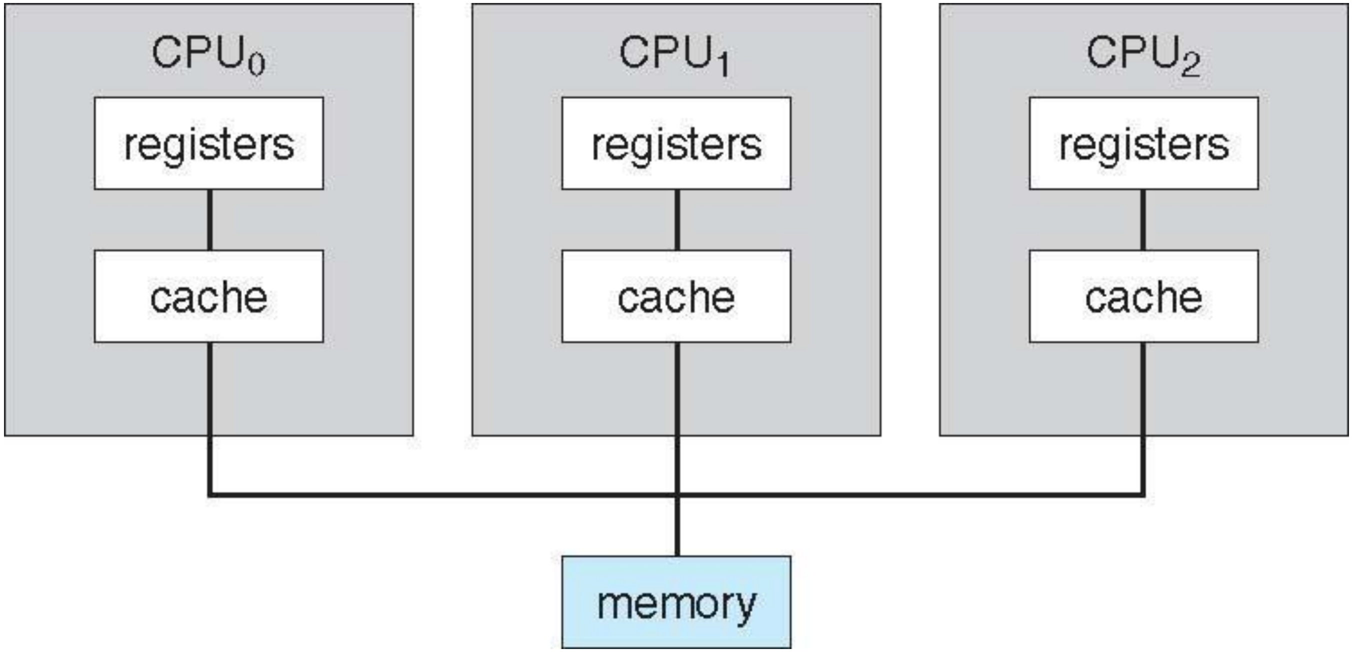
    Increased reliability – graceful degradation or fault tolerance
  - Two types:

    Asymmetric Multiprocessing – each processor is assigned a specific task.

    Symmetric Multiprocessing – each processor performs all tasks


Symmetric Multiprocessing
Vs
Asymmetric Multiprocessing
CPU 1
CPU 2
CPU 3
P1
P2
P3
(Shared Memory)
Slave
Master
Slave
P1
P2
P3
(No Shared Memory)

Symmetric Multiprocessing Architecture

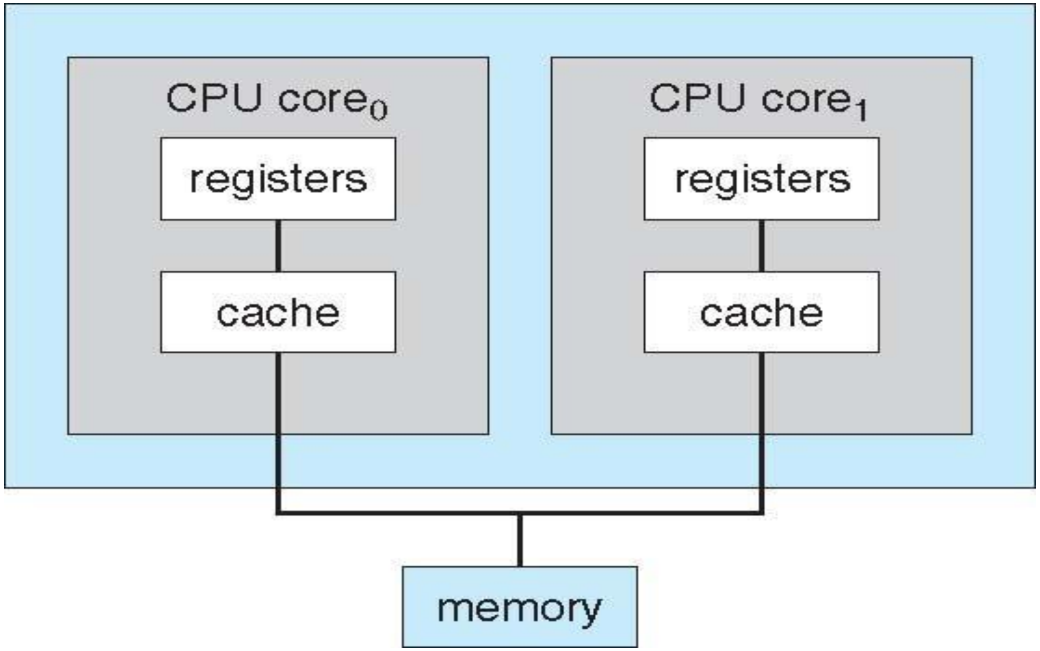


Computer System Architecture

- Multiprocessing adds CPUs to increase computing power.
- Multiprocessing can cause a system to change its memory access model from uniform memory access **(UMA)** to non-uniform memory access (**NUMA**).
- UMA is defined as the situation in which access to any RAM from any CPU takes the same amount of time.
- With NUMA, some parts of memory may take longer to access than other parts, creating a performance penalty.
- Operating systems can minimize the NUMA penalty through resource management,

A Dual-Core Design

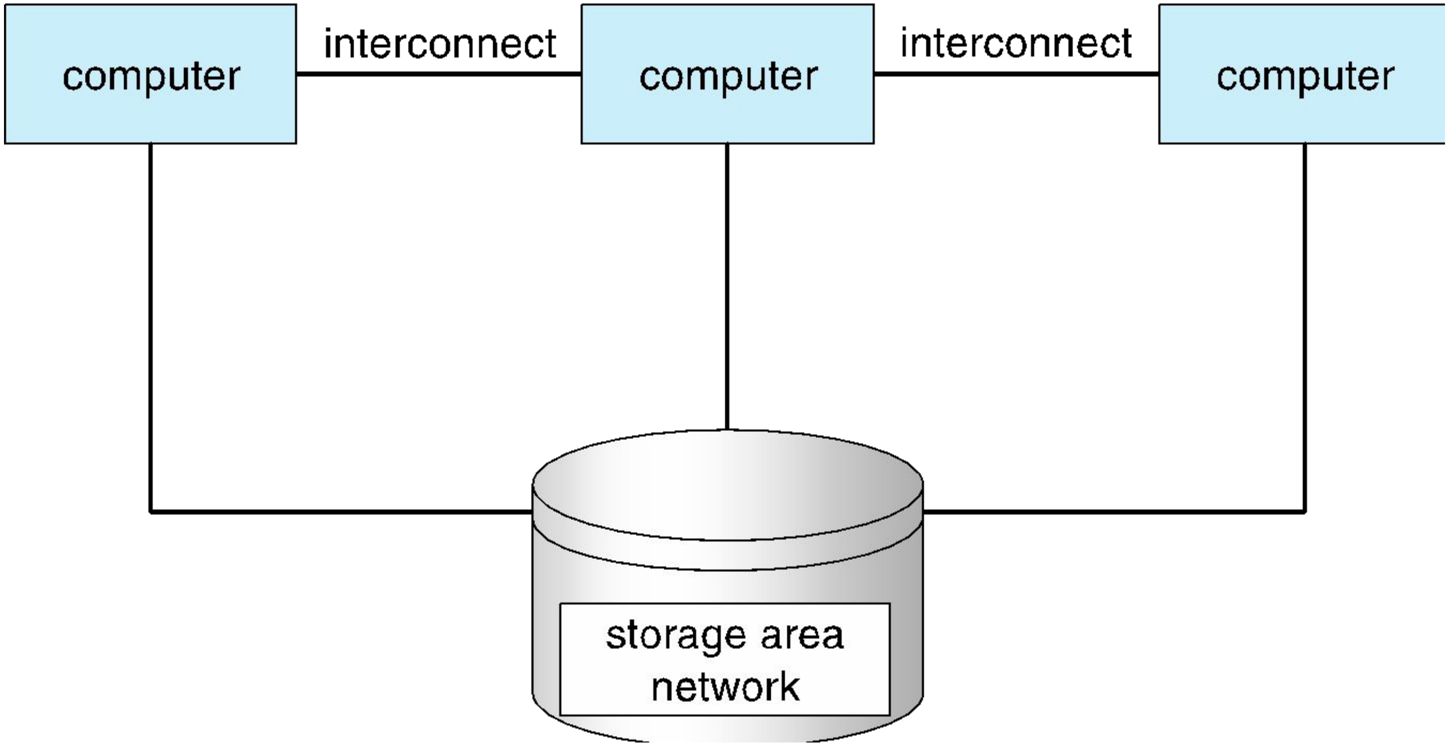
- Multi-chip and **multicore**
- Systems containing all chips
  - Chassis containing multiple separate systems

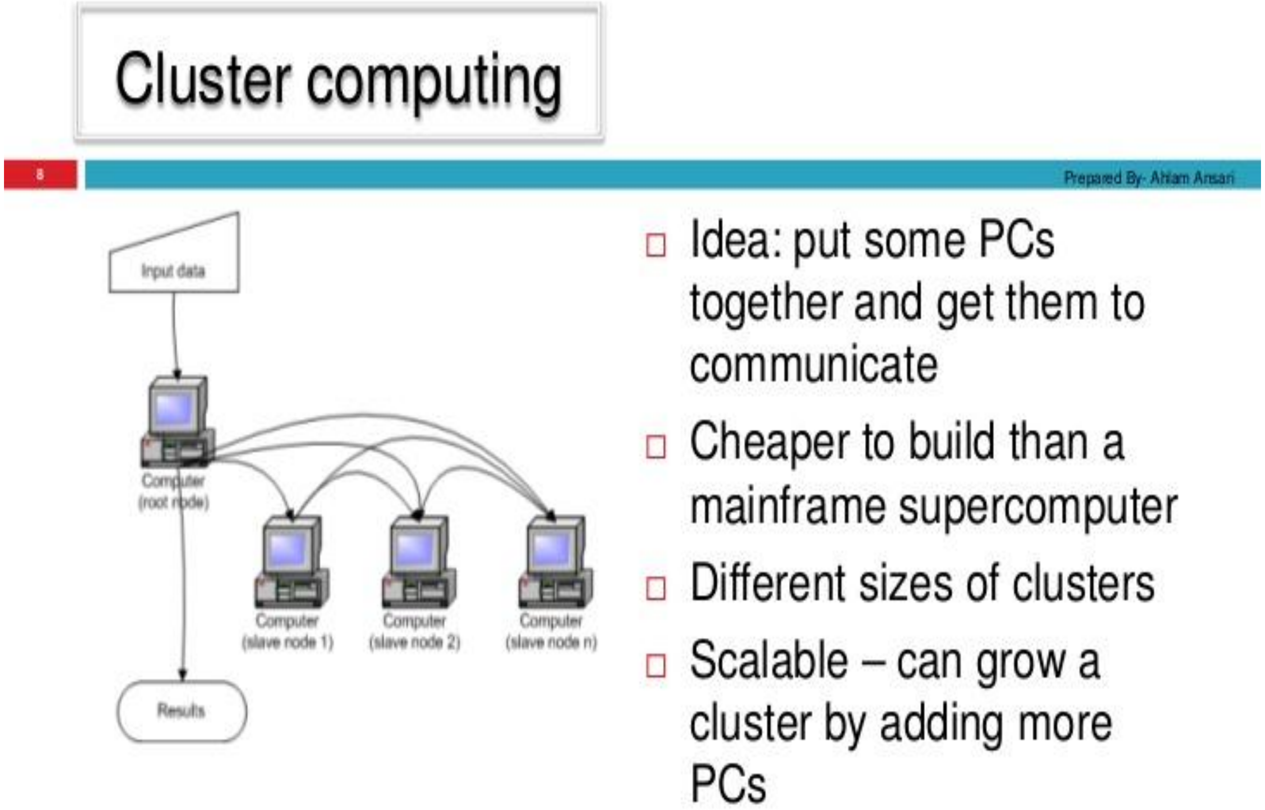


Clustered Systems

- 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





Cluster computing

8

Prepared By- Ahlam Ansari

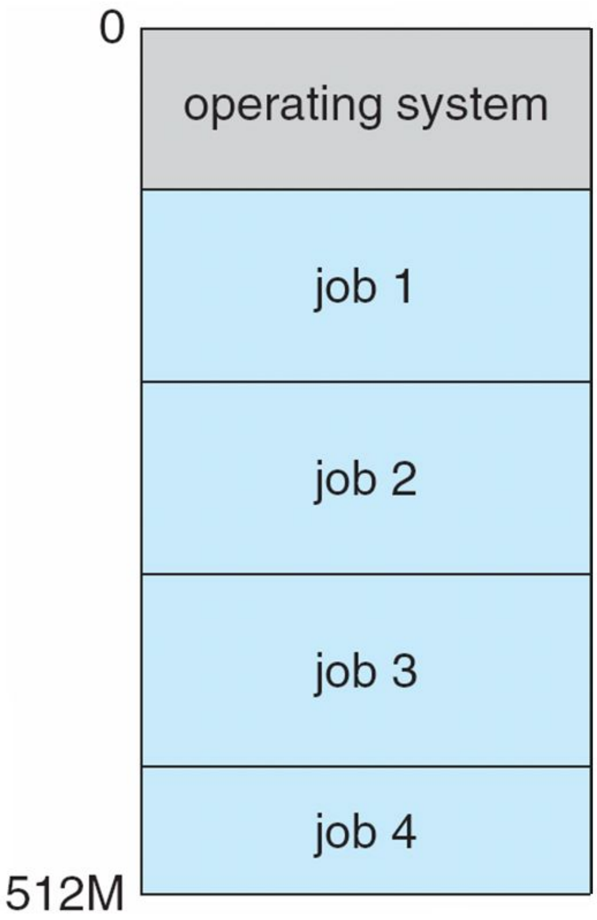
- □ Idea: put some PCs together and get them to communicate
- □ Cheaper to build than a mainframe supercomputer
- □ Different sizes of clusters
- □ Scalable – can grow a cluster by adding more PCs

Operating System Structure

- **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 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



Operating-System Operations

- **Interrupt driven** (hardware and software)
  - Events are almost always signaled by the occurrence of an interrupt or a trap.
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service
    - Other process problems include infinite loop, processes modifying each other or the operating system
- For each type of interrupt, separate segments of code in the operating system determine what action should be taken.
- An interrupt service routine is provided to deal with the interrupt

Operating-System Operations

- **Dual-Mode and Multimode operation**
- **Dual-mode** operation allows OS to protect itself and other system components
- **User mode** and **kernel mode (supervisor mode, system mode, or privileged 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
- When the computer system is executing on behalf of a user application, the system is in user mode.
- However, when a user application requests a service from the operating system (via a system call), the system must transition from user to kernel mode to fulfill the request.
- 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

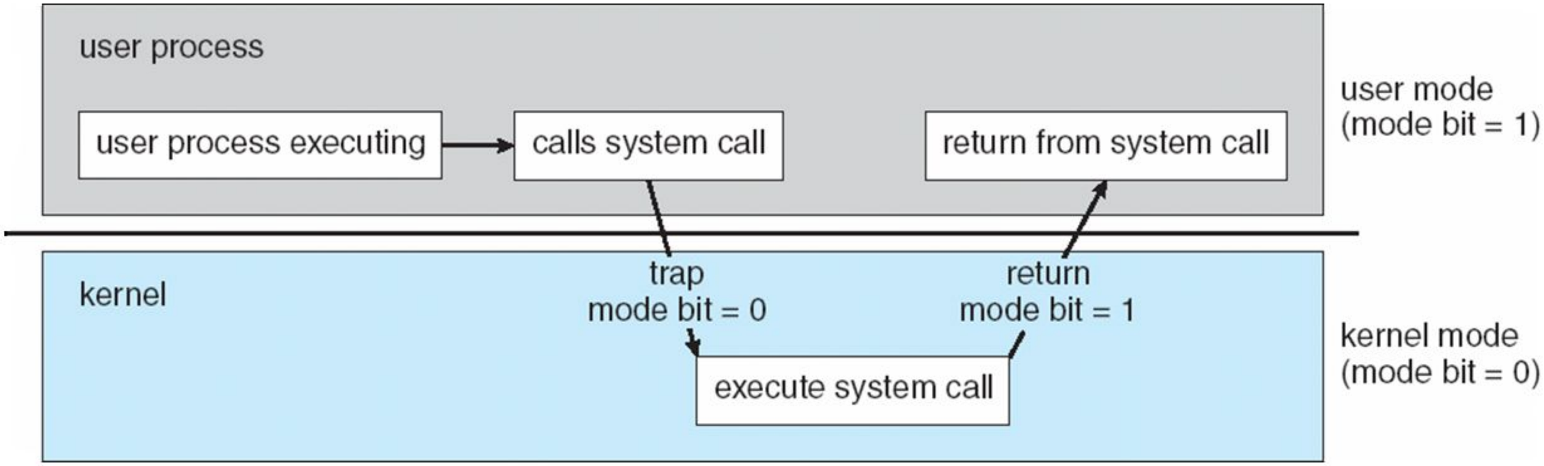


Figure: Transition from user to kernel mode.

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 of 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

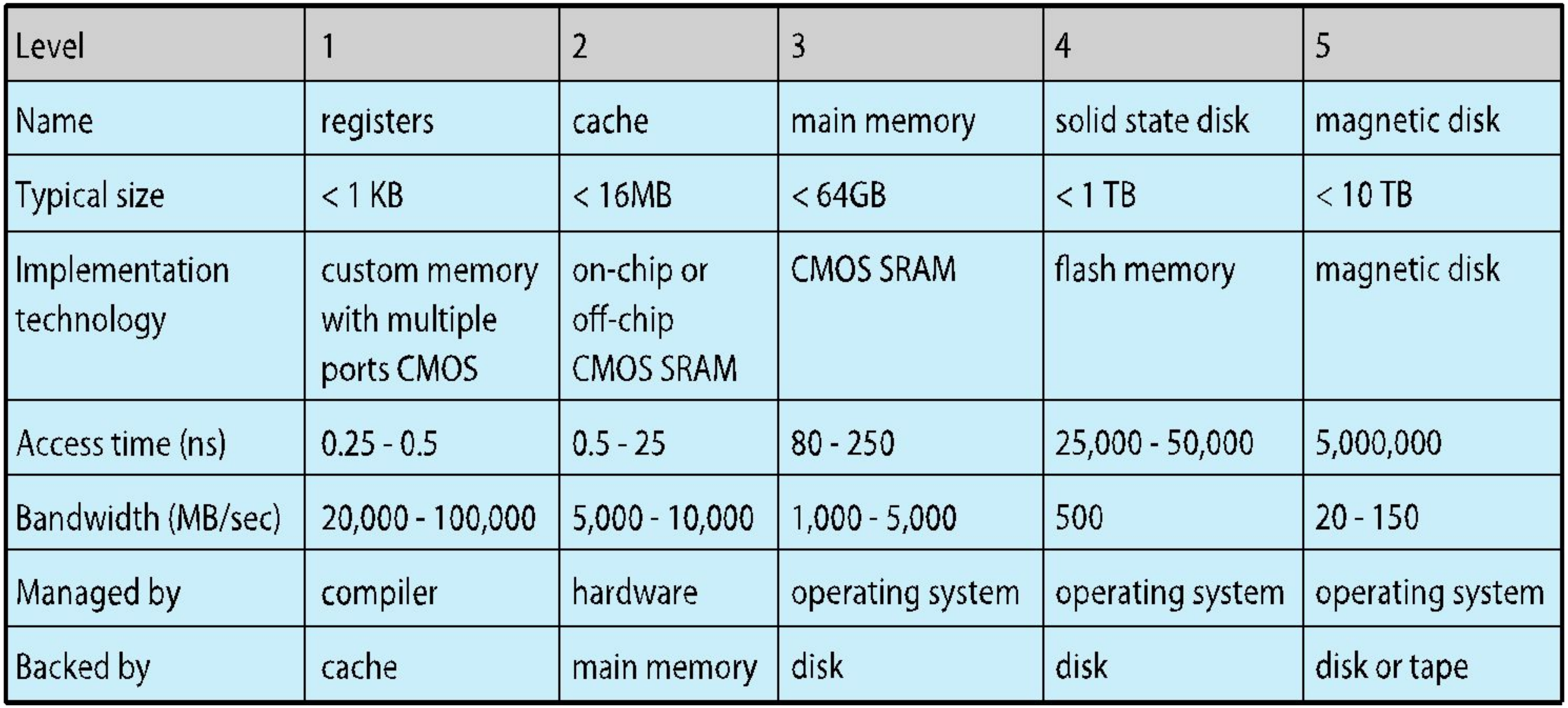
Storage Management

- 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 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
  - Free-space management
  - Storage allocation
  - Disk scheduling

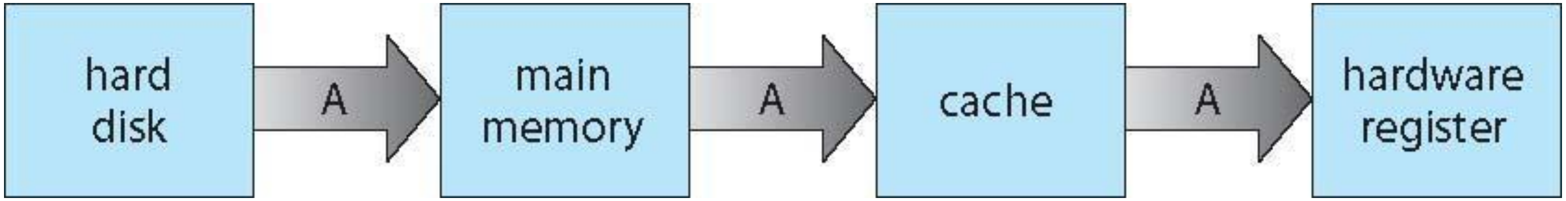
Performance of Various Levels of Storage



Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

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
- Distributed environment situation even more complex
  - Several copies of a datum can exist

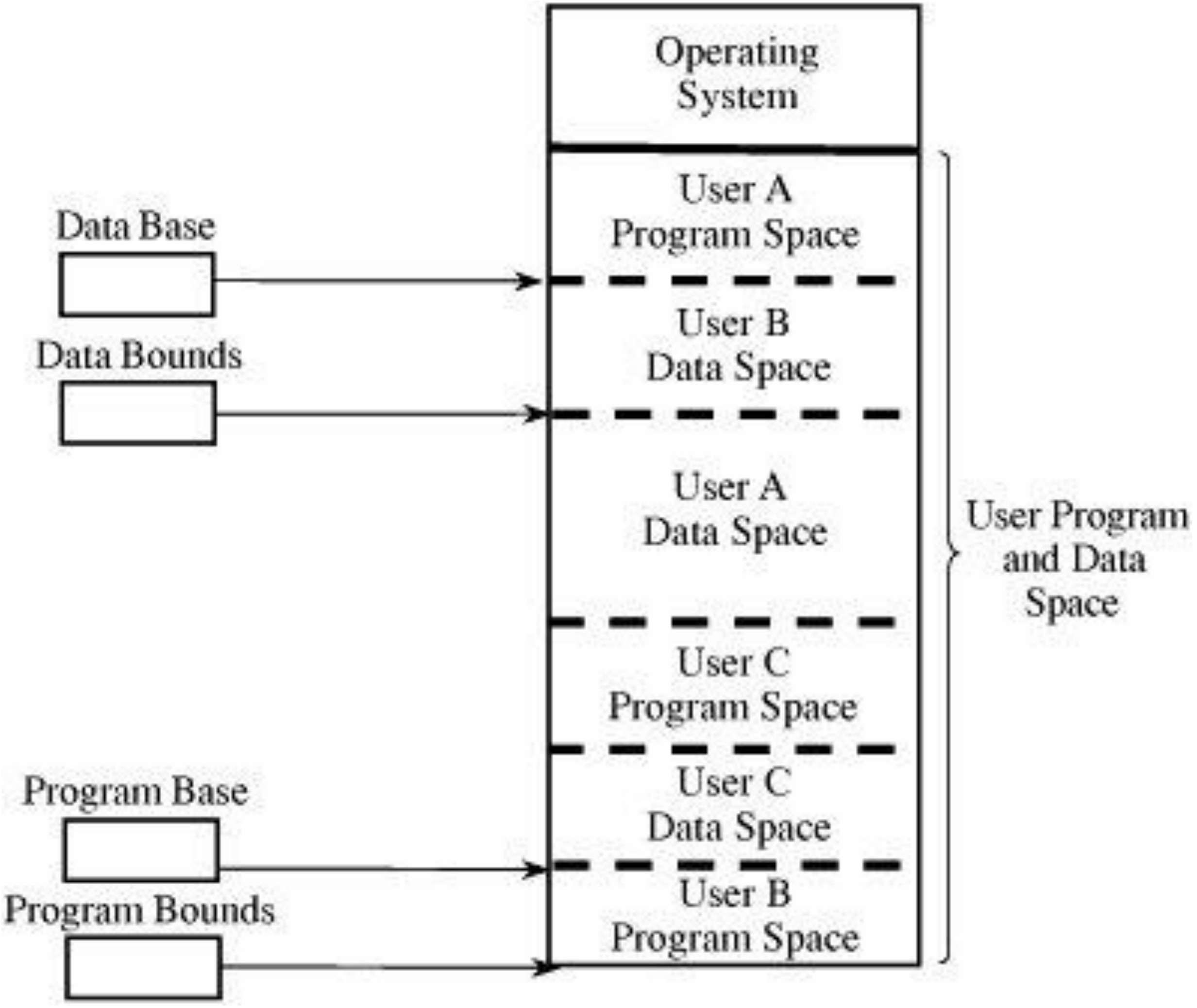


I/O Subsystem

- 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


Operating System
User A Program Space
User B Data Space
User A Data Space
User C Program Space
User C Data Space
User B Program Space
User Program and Data Space
Data Base
Data Bounds
Program Base
Program Bounds

End of Chapter 1