About Me

Engr. Muhammad Usman Malik

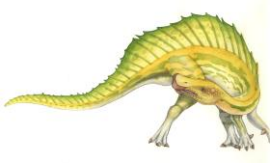
- Lecturer Computer Engineering Department Fast University Peshawar.
- BSc in Computer Systems Engineering from UET Peshawar.
- MSc in Computer Engineering from UET Lahore.
- PhD in Computer Systems Engineering from UET Peshawar (InProgress).

Area Of Interest:

- Machine Learning
- Remote Sensing

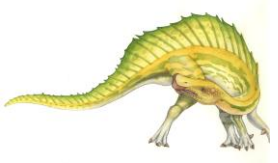
Marks Distribution

- Midterm: 30%
  - Sessional 1: 15%
  - Sessional 2: 15%
- Final: 50%
- Quizzes and Assignments: 20%



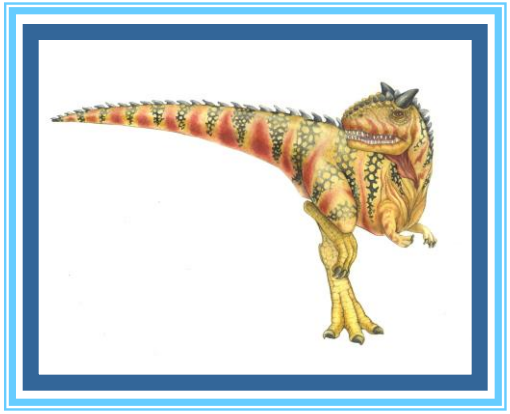
Recommended Books

- **BOOKs RECOMMENDED**
  - Abraham Silberschatz, Peter Baer Galvin and Greg Gagne: Operating System Concepts, 9th or later Edition
- **REFERENCE BOOKS**
  - UNIX Systems Programming: Communication, Concurrency, and Threads by K. A. Robbins and S. Robbins. Prentice Hall, 2003. ISBN: 0-13-042411-0.



Week	Contents
Week 1	Introduction to Operating System
Week 2	OS Structures
Week 3	Program and Processes
Week 4	Processes in UNIX (fork etc.)
Week 5	Threading in OS
Week 6	Threads in Unix (pthreads)
Week 7	Critical Sections and Semaphores
Week 8	Revision
	Mid Term Examination
Week 9,10	Deadlocks
Week 11	Main Memory
Week 12	Virtual Memory
Week 13	Input/output Devices
Week 14	Virtual Machines
Week 15	File System
Week 16	Revision
	Final Term Examination

Chapter 1: Introduction

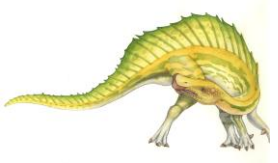


Chapter 1: Introduction

- 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
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems

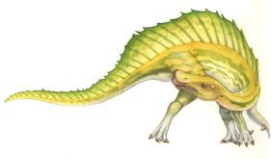
Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems



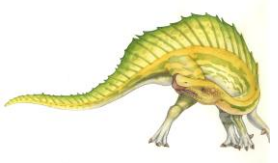
What is an Operating System?

- 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

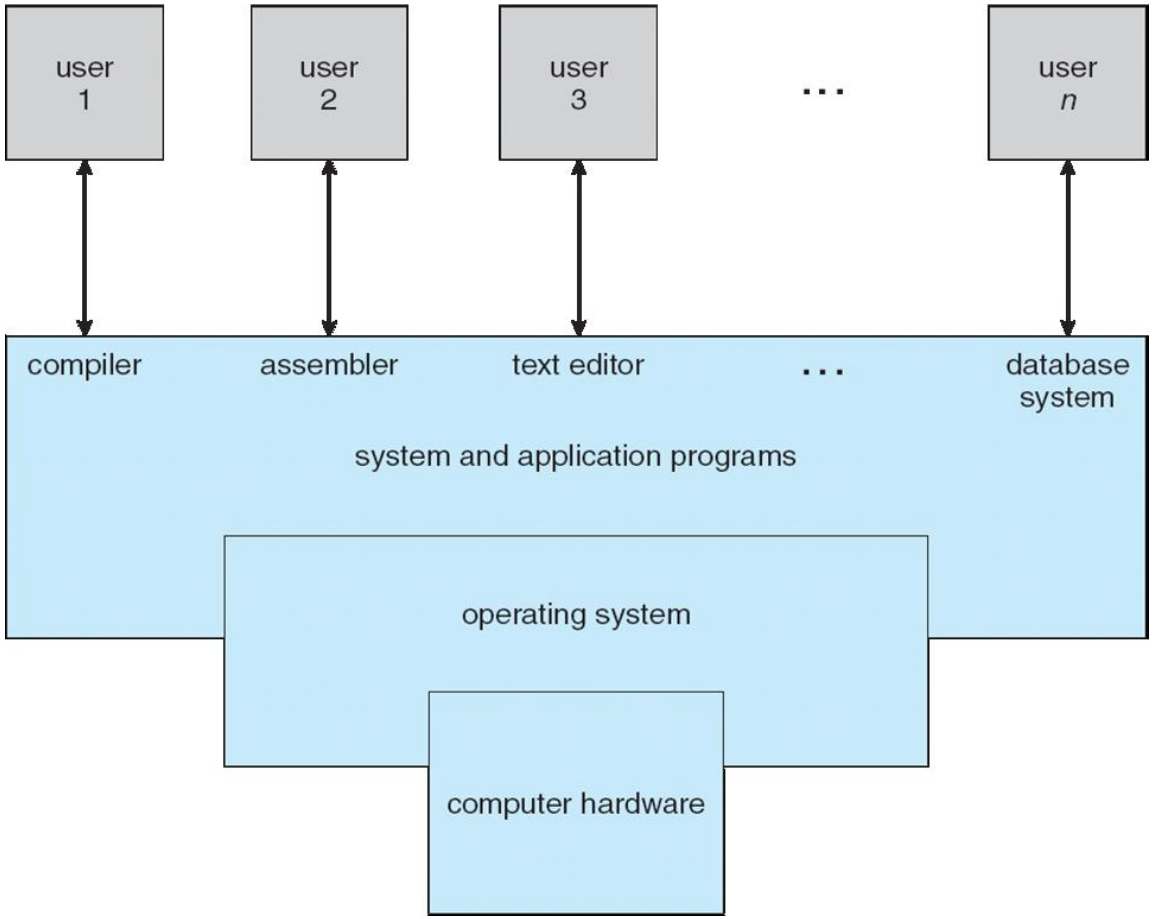


Computer System Structure

- 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

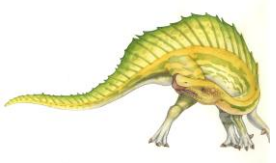


Four Components of a Computer System



What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computers such as **mainframe** or **minicomputer** must keep all users happy
  - Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs
- Users of dedicated systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tables are resource poor, optimized for usability and battery life
  - Mobile user interfaces such as touch screens, voice recognition
- Some computers have little or no user interface, such as embedded computers in devices and automobiles
  - Run primarily without user intervention

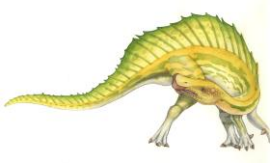


What Operating Systems Do (Contd)

- 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

Operating System Definition (Cont.)

- 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.



Operating System Definition (Cont.)

- - The kernel is described as "the one program running at all times on the computer."
- **Kernel:** It's the core part of the OS, responsible for managing system resources (like CPU, memory, and devices) and allowing different software components to communicate with the hardware.

- **System Programs and Application Programs**
- **System Programs:** These are programs that come with the operating system and perform essential functions, such as file management, device management, and system utilities.
- **Application Programs:** These are programs that the user installs and uses for specific tasks, such as word processing, web browsing, or playing games.

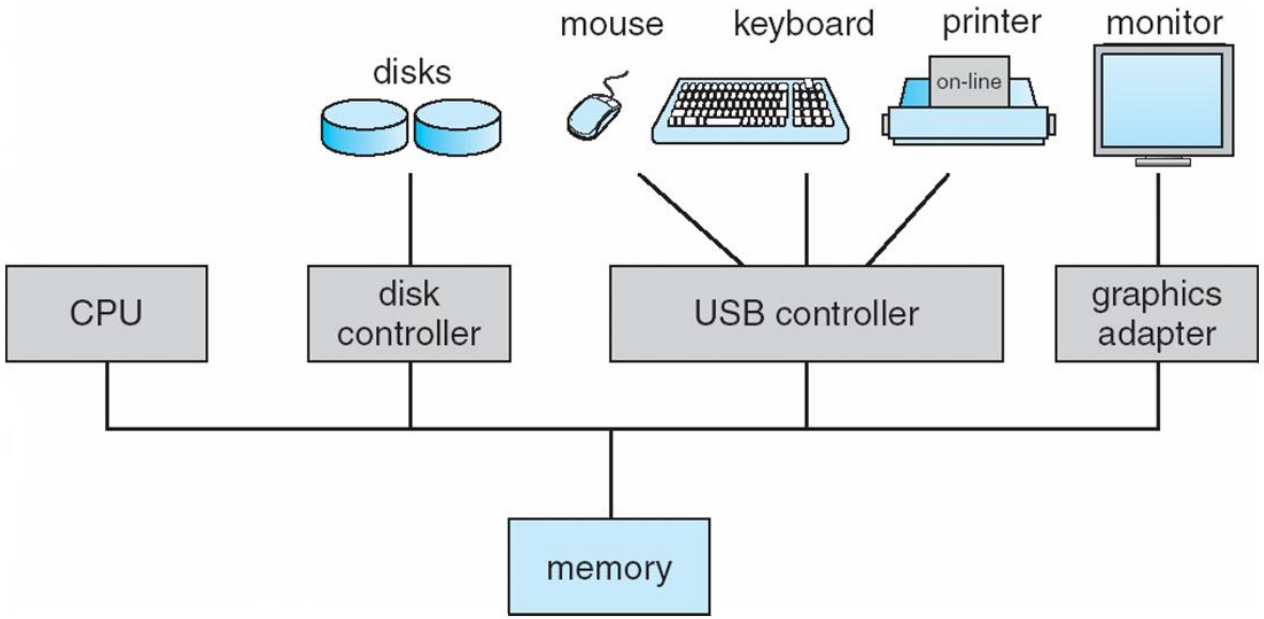
Computer Startup

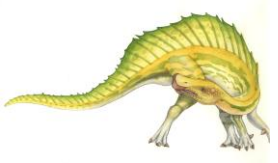
- **bootstrap program** is loaded at power-up or reboot
  - The bootstrap program is automatically loaded when the computer is turned on or restarted. It's the first piece of software that runs, and it begins the process of getting the computer ready to operate
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of the system
  - Loads operating system kernel and starts execution

Overview of Computer System Structure

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 cyclesa



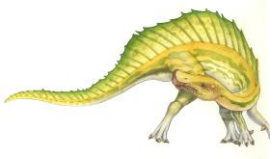


Computer-System Operation

- 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

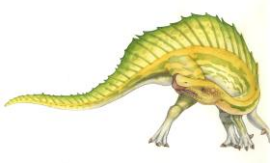
Common Functions of Interrupts

- 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**



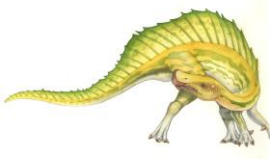
Interrupt Handling

- 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 Handling

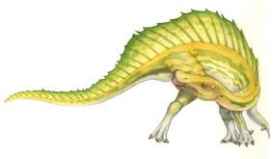
- **Preserving the State of the CPU:**
  - When an interrupt occurs, the OS needs to pause the current task being executed by the CPU. To ensure that the task can be resumed later without any issues, the OS stores the current state of the CPU, including:
- **Registers:** These are small storage areas in the CPU that hold important data and addresses being used by the current program.
- **Program Counter:** This is a special register that keeps track of the next instruction to be executed by the CPU.



Interrupt Handling

- **Determining the Type of Interrupt:**

1. **Polling:** In this method, the OS checks each device or source to see if it has generated the interrupt. This is like the OS going through a list and asking each device, "Did you cause the interrupt?“
2. **Vectored Interrupt System:** In this method, the interrupting device sends a unique identifier (an interrupt vector) to the CPU. This vector directly points to the specific interrupt handling routine, so the OS knows immediately which device or event caused the interrupt without having to check each one.



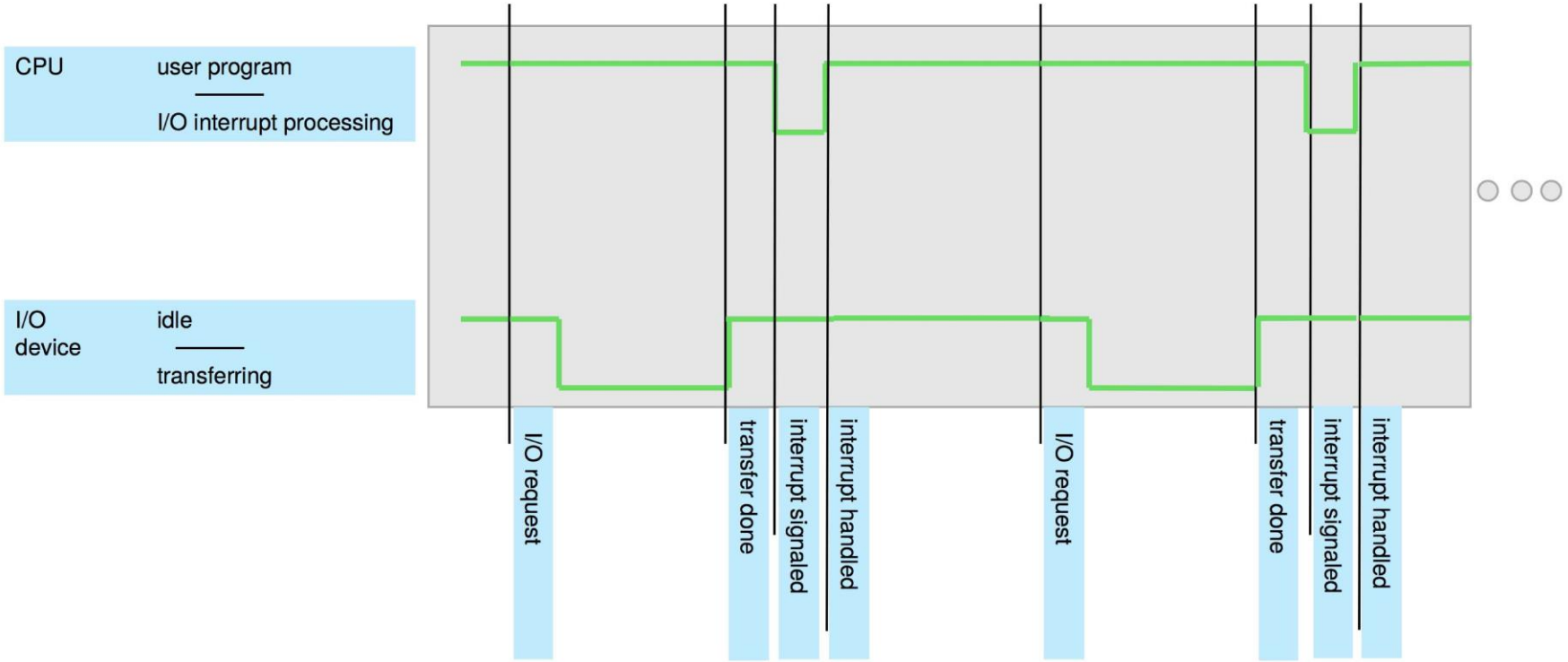
Interrupt Handling

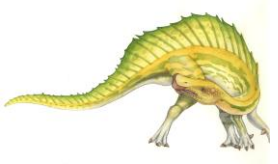
- **Action Based on the Type of Interrupt:**

  - Once the OS determines the type of interrupt, it executes the appropriate segment of code (interrupt service routine) that corresponds to that specific interrupt. Different types of interrupts might require different responses. For example:

- - A timer interrupt might prompt the OS to switch tasks in a multitasking system.
- - An I/O interrupt might prompt the OS to read data from a disk or network.

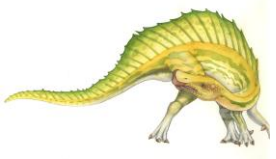
Interrupt Timeline





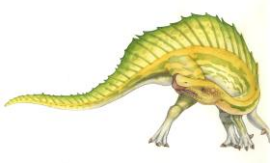
I/O Structure

- 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.



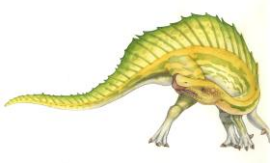
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically, **volatile**
- Secondary storage – an 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



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



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)
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

Storage-Device Hierarchy

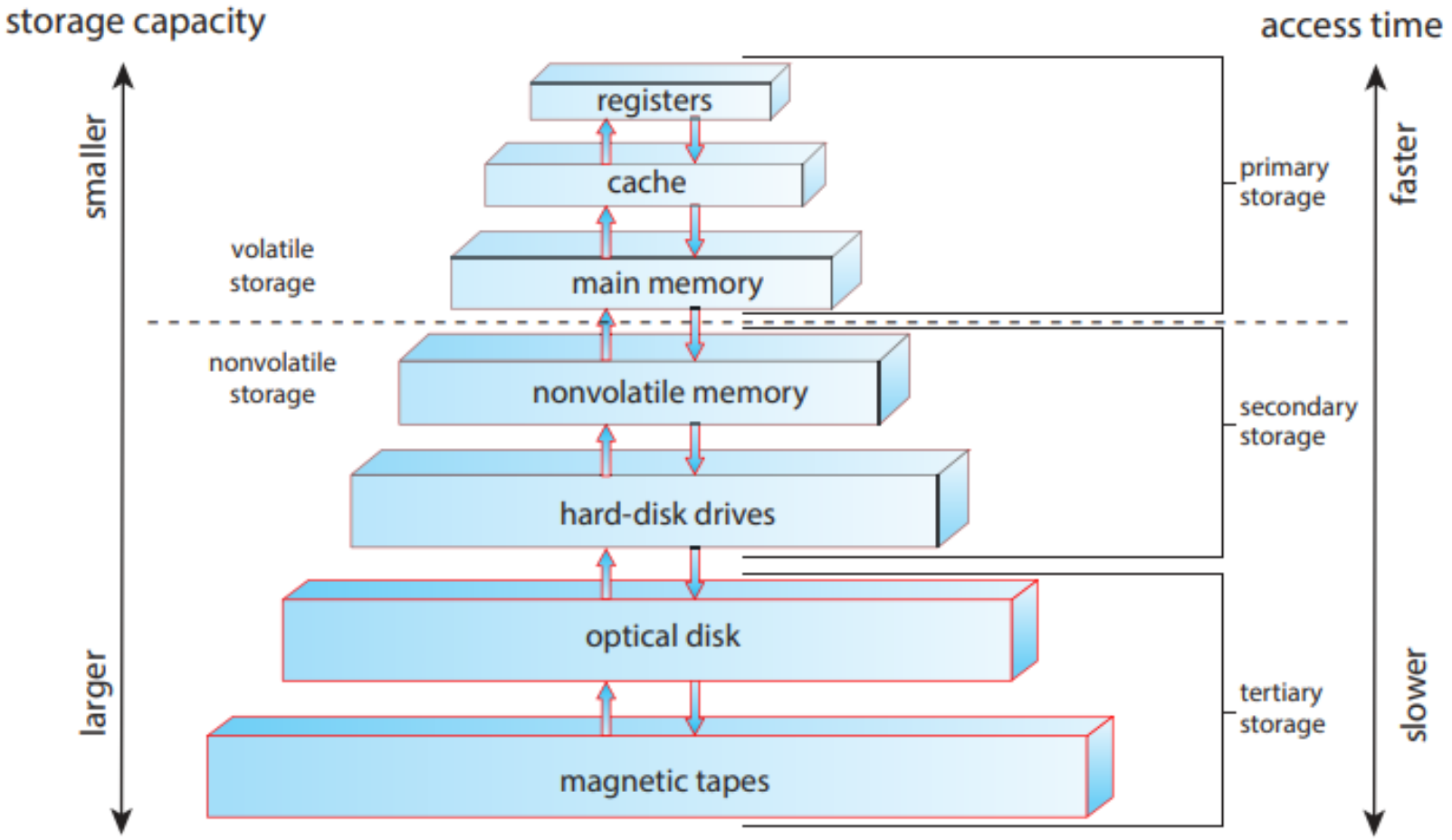
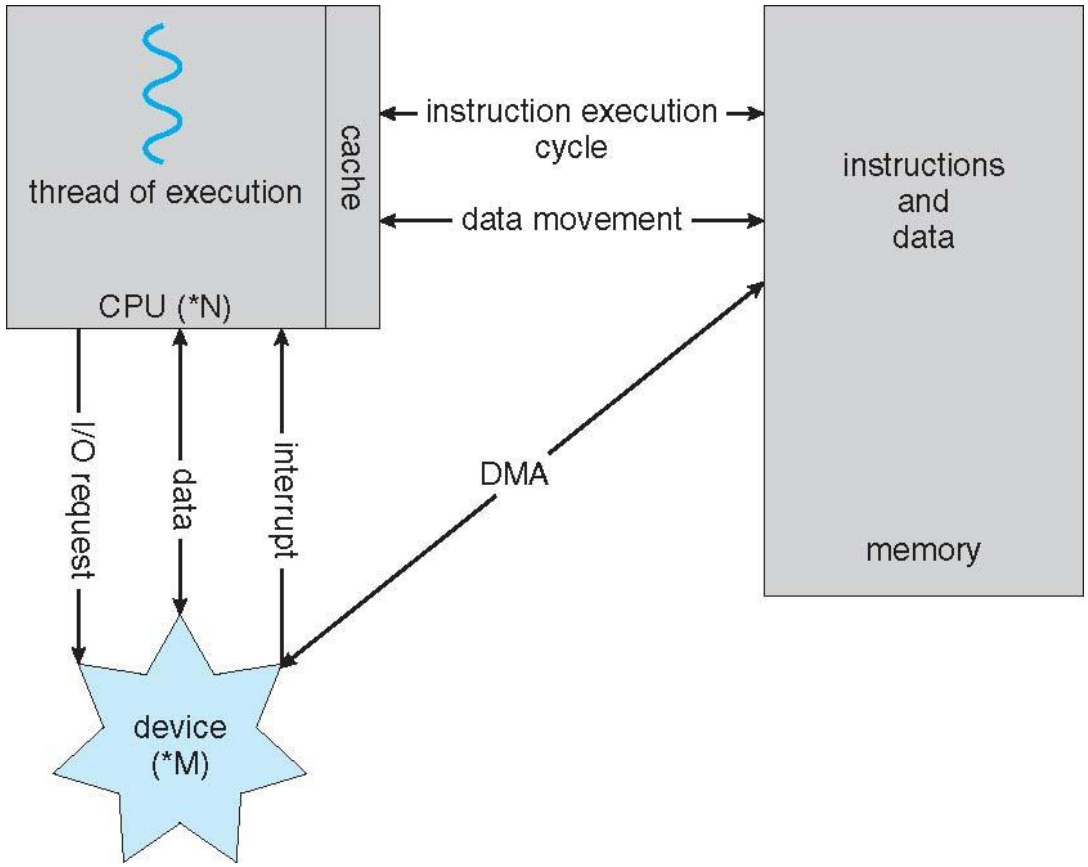


Figure 1.6 Storage-device hierarchy.

Direct Memory Access 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

How a Modern Computer Works

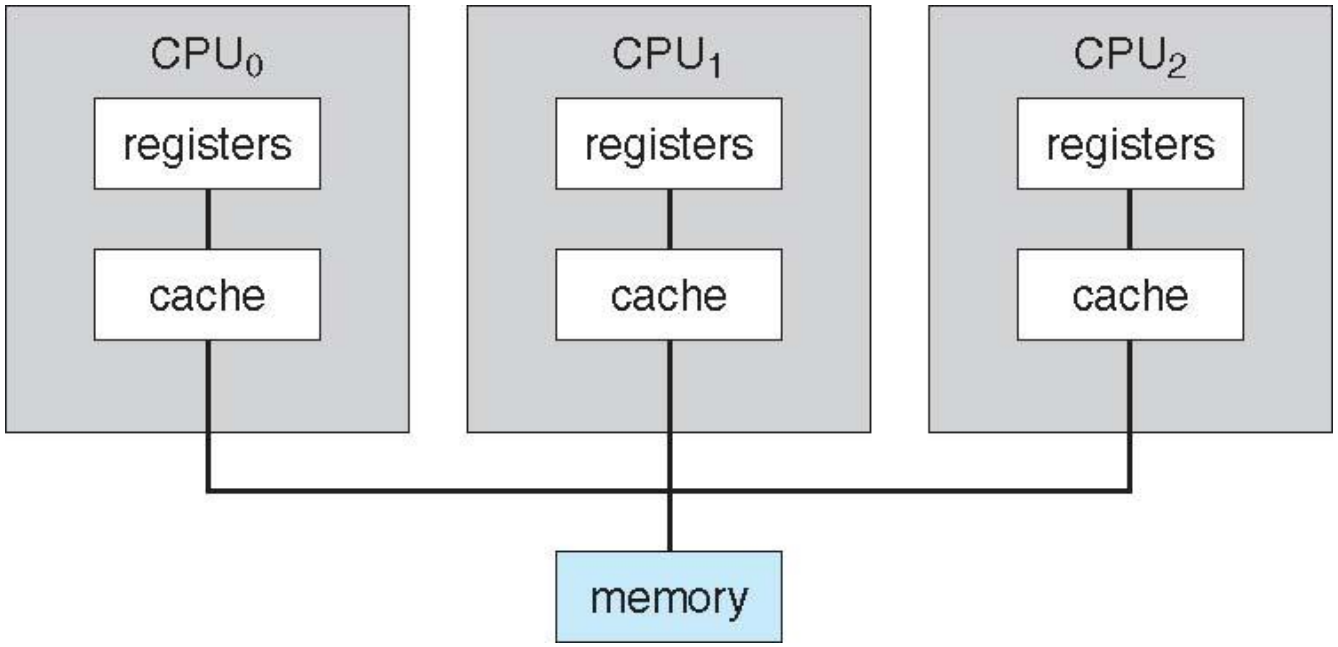


A von Neumann architecture

Computer-System Architecture

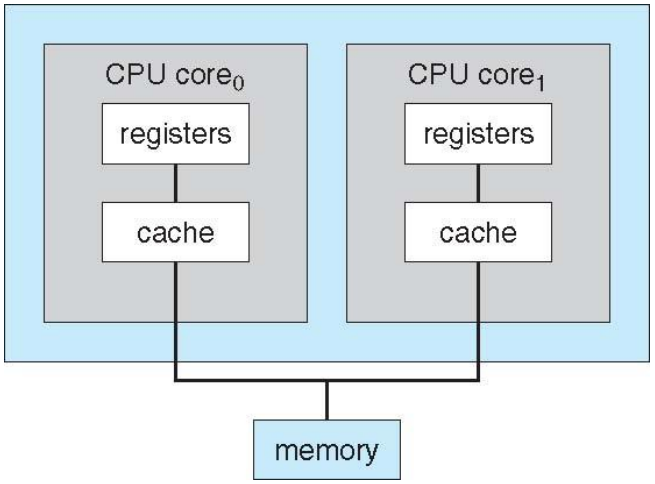
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** involve multiple CPUs that can execute processes in parallel
  - Also known as **parallel systems**,
  - Advantages include:
    1. **Increased throughput**
    2. **Economy 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

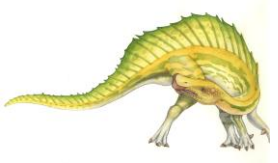
Symmetric Multiprocessing Architecture



A Dual-Core Design

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



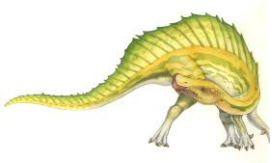


Types of operating System

1. Batch Operating System
2. Multi-programmed OS
3. Multi tasking OS
4. Real-Time OS
5. Embedded OS

Architectures

1. Distributed System
2. Clustered System



Types of operating System

1. Distributed System

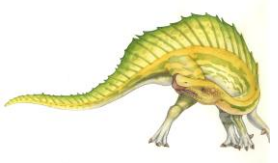
In Distributed systems multiple independent computers (often called nodes or machines) work together to achieve a common goal. These systems are designed to handle large-scale processing and data storage by distributing tasks across several machines, which can be geographically dispersed or co-located.

1. Clustered System

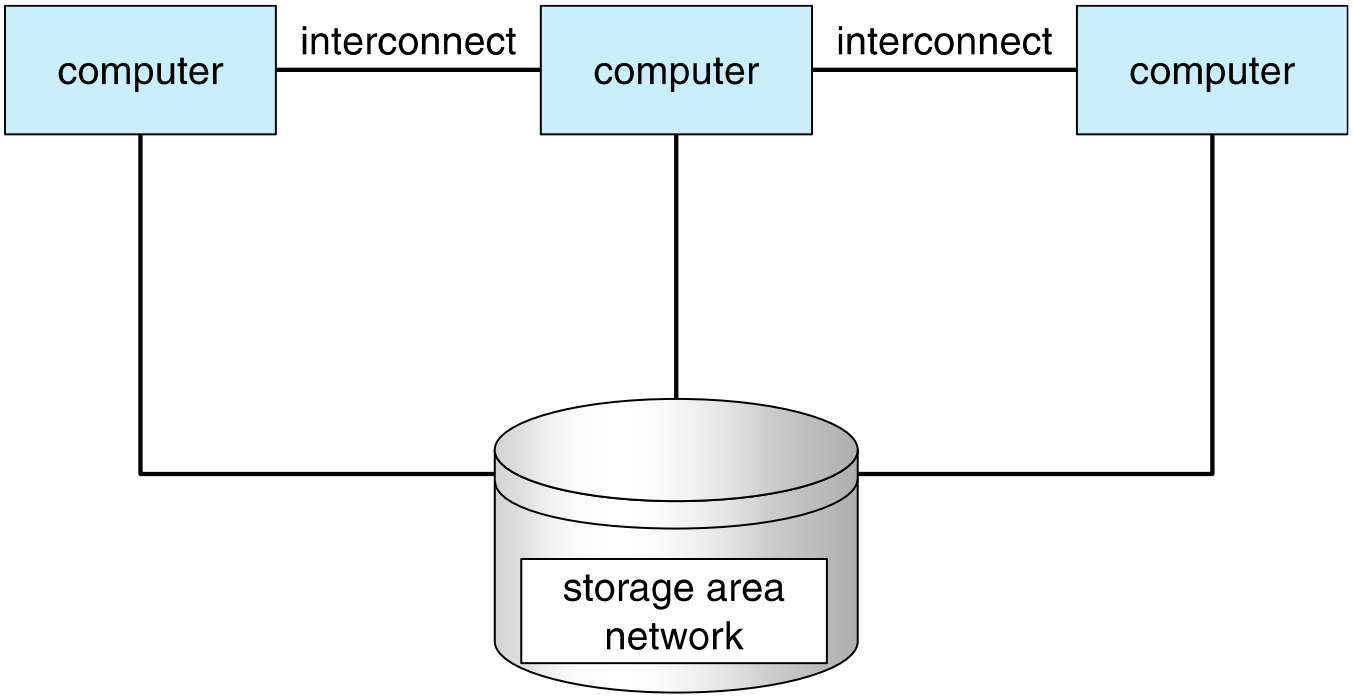
A **clustered system** is a type of distributed system where a group of closely connected computers (nodes) work together as a single unit to provide higher availability, performance, or scalability.

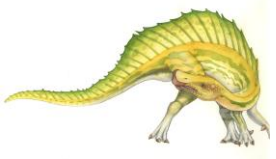
Clustered Systems

- 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**



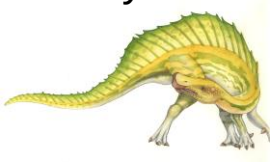
Clustered Systems



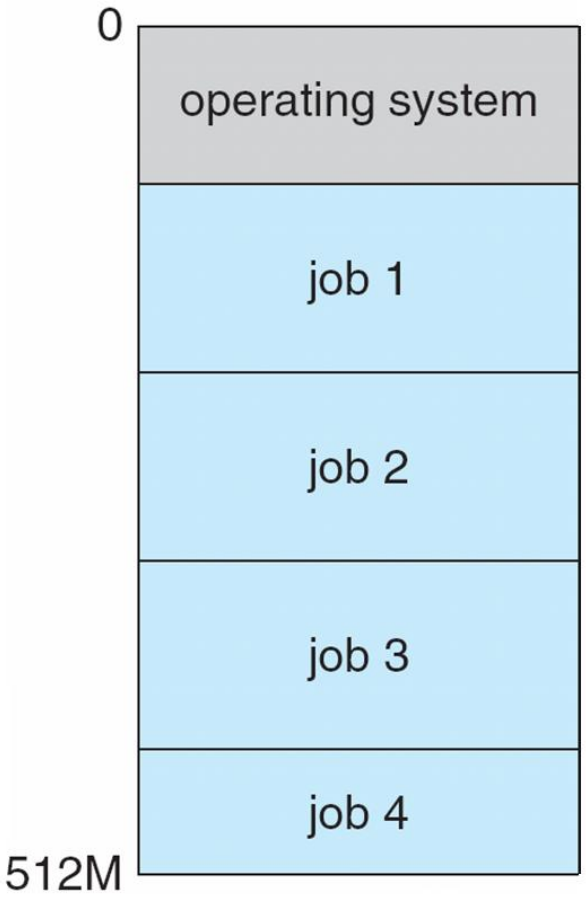


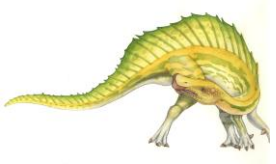
Operating System Structure

- **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



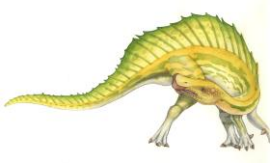


Operating-System Operations

- **Interrupt driven** (hardware and software)
  - 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

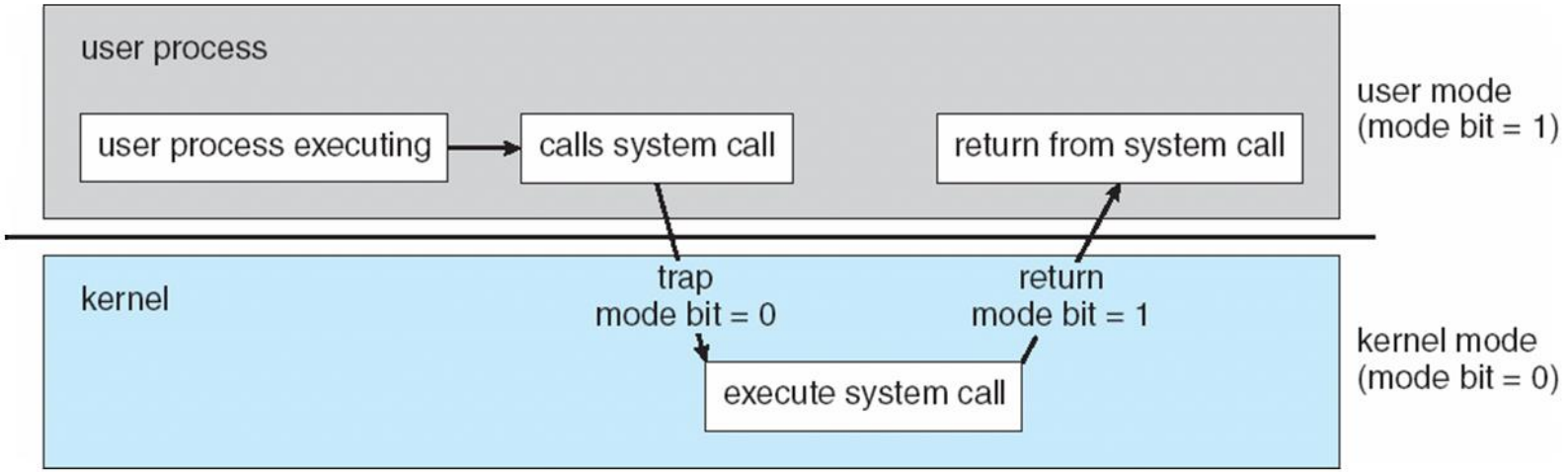
Operating-System Operations (cont.)

- **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



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
- A multithreaded process has multiple program counters, each pointing to the next instruction to execute for a given thread.

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

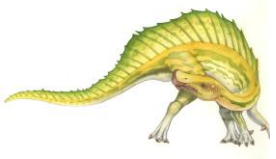
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- Main memory is a large array of bytes, ranging in size from hundreds of thousands to billions. Each byte has its own address.
- Main memory is a repository of quickly accessible data shared by the CPU and I/O devices.
- The CPU reads instructions from main memory during the instruction-fetch cycle and both reads and writes data from main memory during the data-fetch cycle

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
  - To improve both the utilization of the CPU and the speed of the computer's response to its users, general-purpose computers must keep several programs in memory, creating a need for memory management.

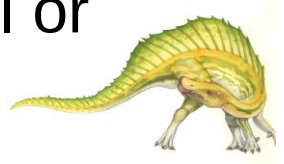


Memory Management

- 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

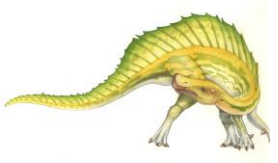
Storage Management

- To make the computer system convenient for users, the operating system provides a uniform, logical view of information storage.
  - Abstracts physical properties to logical storage unit - **file**
  - A file is a container that holds data or information in a structured format.
  - Files can store a variety of data types, such as text, images, videos and executable code etc.
  - 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)



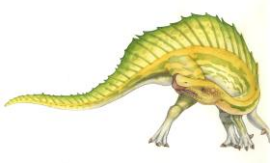
Storage Management

- File-System management
  - Files are fundamental units of data storage in a computer System
  - Files are 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

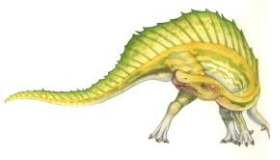
- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)



Performance of Various Levels of Storage

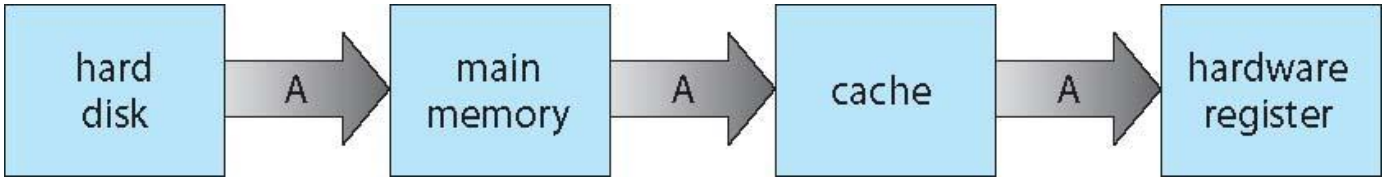
Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 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

Movement between levels of storage hierarchy can be explicit or implicit



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



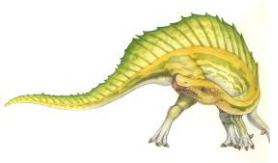
- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

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), 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



End of Chapter 1