SRI KRISHNA ARTS AND SCIENCE COLLEGE

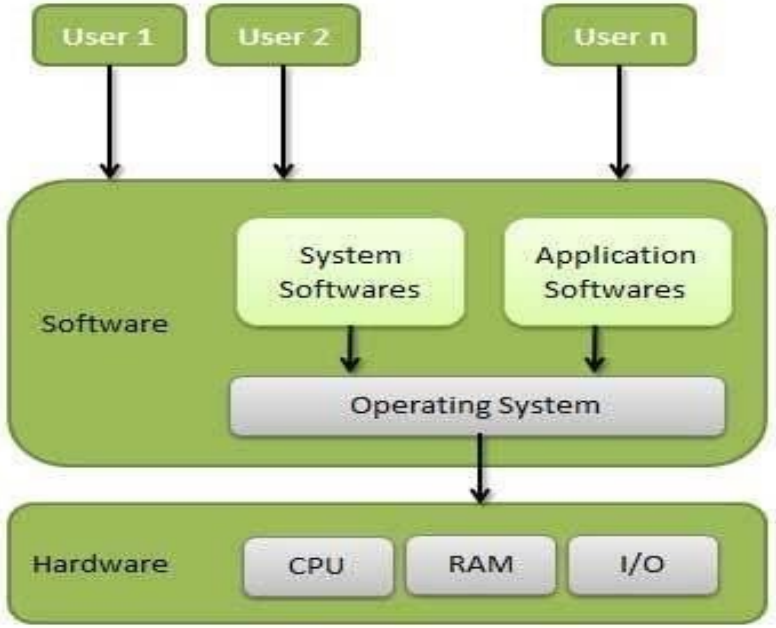
DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 1:What is Operating System?

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content



- What is an Operating system?
- Types of OS
- Benefits
- Functions of Operating System
- Example: How the Application Programs Processes

Introduction

- An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs.
- The operating system is a vital component of the system software in a computer system.
- An operating system act as an intermediary between the user of a computer and computer hardware.

- The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

- An operating system is a software that manages the computer hardware.

- The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system

- To prevent user programs from interfering with the proper operation of the system.

What is an operating system?

- An operating system can be viewed as a set of software programs normally along with the hardware for the effective and use of the machine.

- **Two benefits are**
  - Elimination of duplicate efforts
  - Provision of security and confidentiality of information to the users.
- OS is responsible for allocating and deallocating the disk.

1. What is operating system
2. Types of operating system

Types of Operating System

Windows

Windows 8

Linux

fedora

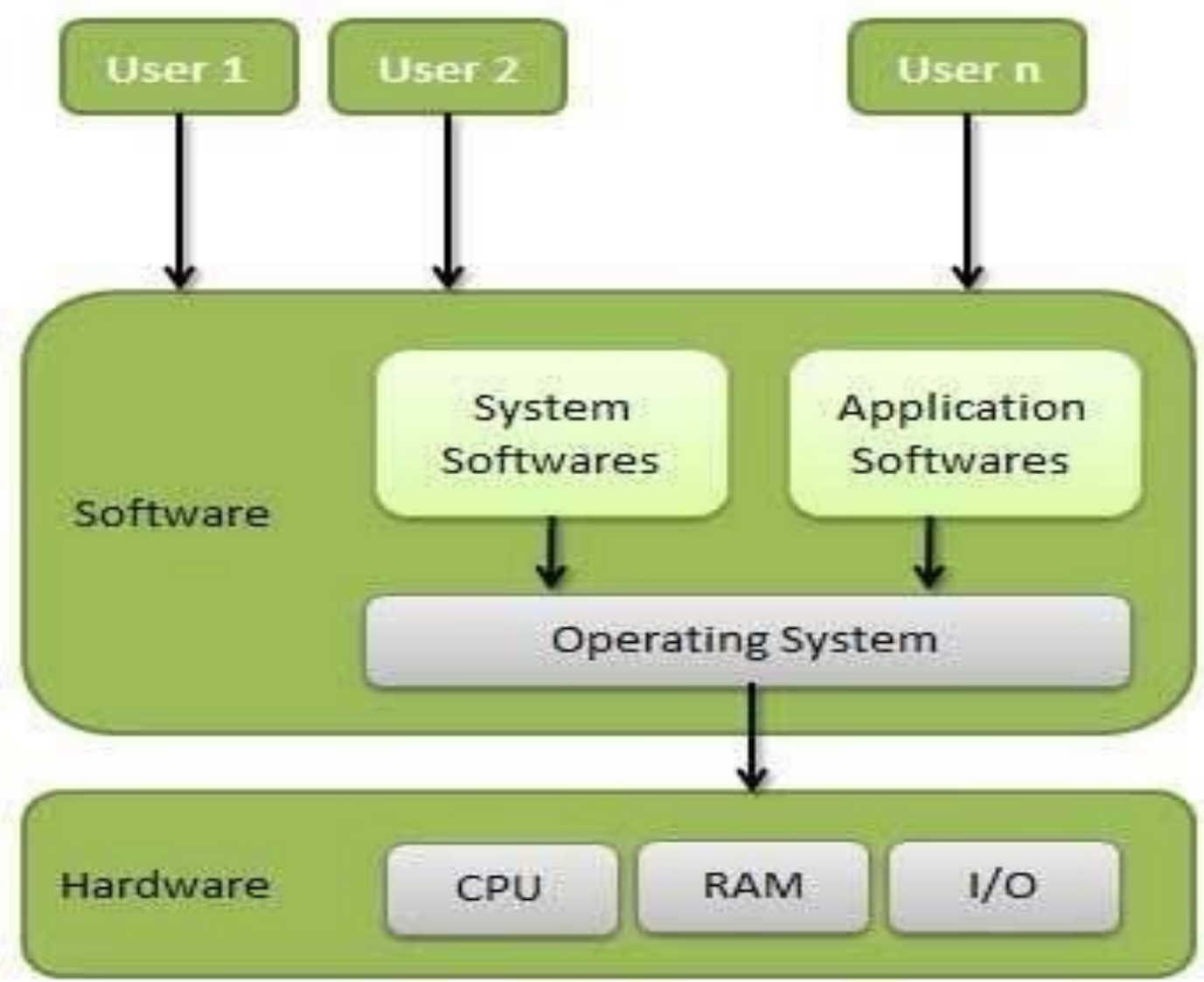
ubuntu

Mac OS

android

Definition of Operating System

- An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.
- A more common definition is that the operating system is the one program running at all times on the computer (usually called the kernel), with all else being applications programs.


User 1
User 2
User n
Software
System Softwares
Application Softwares
Operating System
Hardware
CPU
RAM
I/O

- An Operating system is concerned with the allocation of resources and services, such as memory, processors, devices and information.

- The Operating System correspondingly includes programs to manage these resources, such as a traffic controller, a scheduler, memory management module, I/O programs, and a file system.

Operations Of OS

- Start and shut down a computer
- Coordinate Tasks
- Establish An Internet Connection
- Provide a user Interface
- Configure Devices
- Control a network
- Manage Programs
- Manage Memory
- Provide Utilities

Functions of Operating System

- Process Management
- Memory Management
- Secondary Storage Management
- I/O Management
- File Management
- Protection
- Networking Management
- Command Interpretation.

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

- The creation and deletion of both user and system processes
- The suspension and resumption of processes.
- The provision of mechanisms for process synchronization
- The provision of mechanisms for deadlock handling.

Memory Management :

The operating system is responsible for the following activities in connection with memory management.

- Keep track of which parts of memory are currently being used and by whom.
- Decide which processes are to be loaded into memory when memory space becomes available.
- Allocate and deallocate memory space as needed.

Secondary Storage Management :

- The main memory is too small to permanently accommodate all data and program, the computer system must provide secondary storage to backup main memory.

- Most programs, like compilers, assemblers, sort routines, editors, formatters, and so on, are stored on the disk until loaded into memory.

- Then use the disk as both the source and destination of their processing.

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

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

I/O Management

- One of the important jobs of an Operating System is to manage various I/O devices including
  - Mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc.

I/O Management

- An I/O system is required to
  - Take an application I/O request and
  - Send it to the physical device, then
  - Take whatever response comes back from the device and
  - Send it to the application.

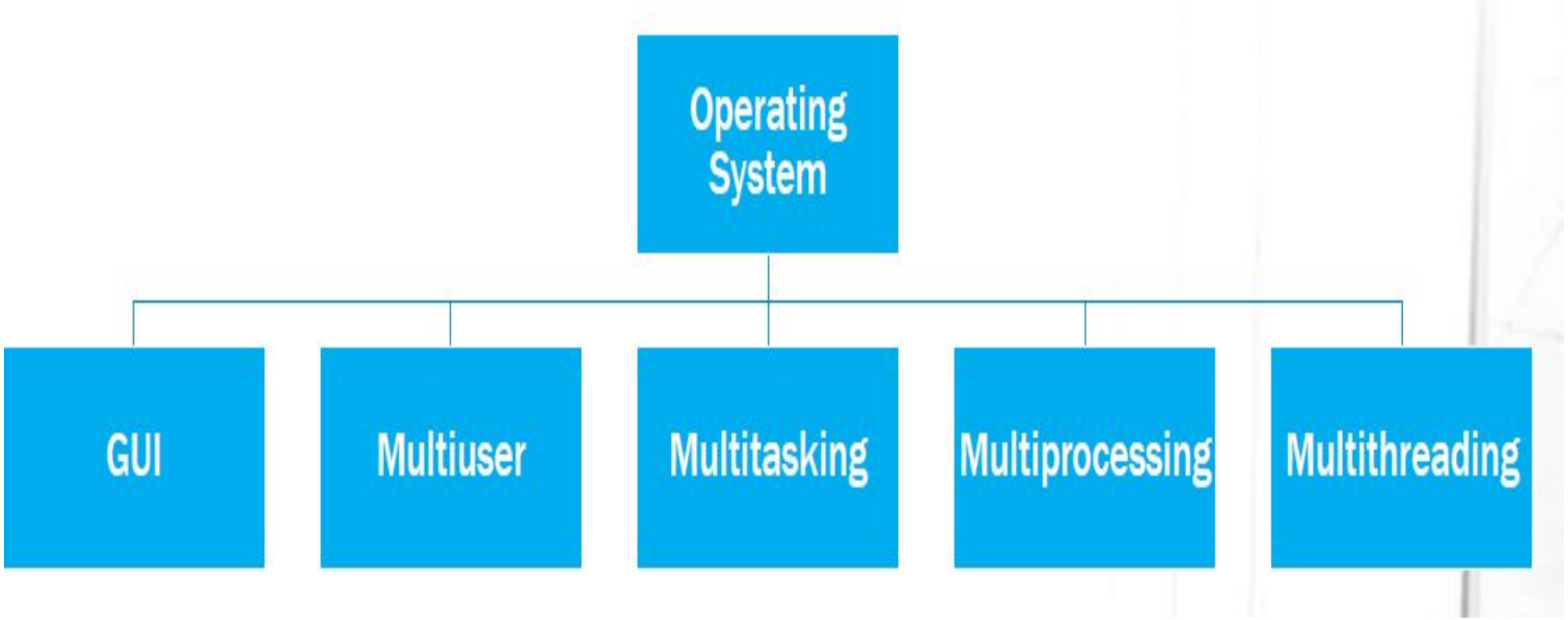
Processor Management

- The heart of managing the processor comes down to two related issues:
  - Ensuring that each process and application receives enough of the processor's time to function properly
  - Using as many processor cycles as possible for real work

System calls related to IM

- Create a file
- Create a Directory
- Open a file
- Close a file
- Read data from file to buffer
- Write data from buffer to file
- Move the file pointer
- Read and return a file status
- Create a pipe
- Create a link
- Change working directory

Features of an operating system

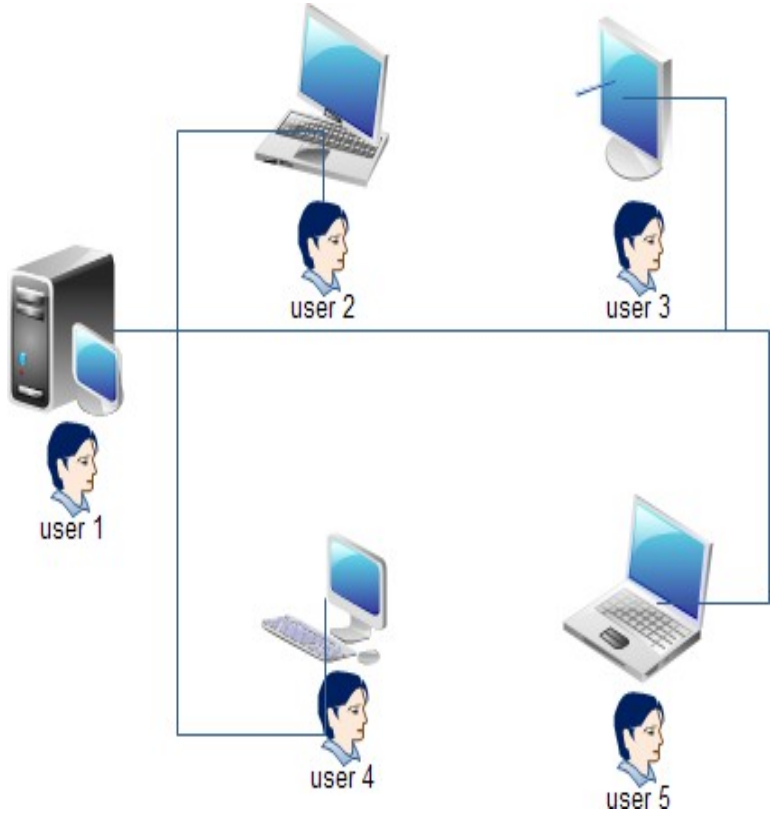


GUI

- **GUI** is “Graphical User Interface”
- Allows users to interact with electronic devices through graphical icons and visual indicators
- **GUI** is visually intuitive, users tend to learn how to use a **GUI** faster **than** a **CLI**.
- Generally provide users with immediate, visual feedback about the effect of each action

Multi-User

- **Definition**: A Multi-user operating system is a computer operating system which allows multiple users to access the single system with one operating system on it.
- **Example**: Linux, Unix, Windows 2000, Ubuntu, Mac OS etc.,
- In the multi-user operating system, different users connected at different terminals and we can access, these users through the network as shown in the diagram.



Multi-Tasking

- **Definition:** Multitasking, in an operating system, is allowing a user to perform more than one computer task (such as the operation of an application program) at a time.

- **Examples**: Microsoft Windows 2000, IBM's OS/390, and Linux
- It is easy to confuse multitasking with multithreading, a somewhat different idea.

Multi Processing

- **Definition:** Multi-processing refers to the ability of a system to support more than one processor at the same time.
- Applications in a multi-processing system are broken to smaller routines that run independently.
- The operating system allocates these threads to the processors improving performance of the system.
- **Examples:** Windows NT, 2000, XP, and Unix.

Multi Threading

- **Definition: Multithreading** is the ability of a program or an **operating system** process to manage its use by more than one user at a time
- To manage multiple requests by the same user without having to have multiple copies of the programming running in the computer.
- A **thread** is a path which is followed during a program's execution. Majority of programs written now a days run as a single thread.
- A process is a program being executed. A process can be further divided into independent units known as threads.

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 2: Computer System Organization

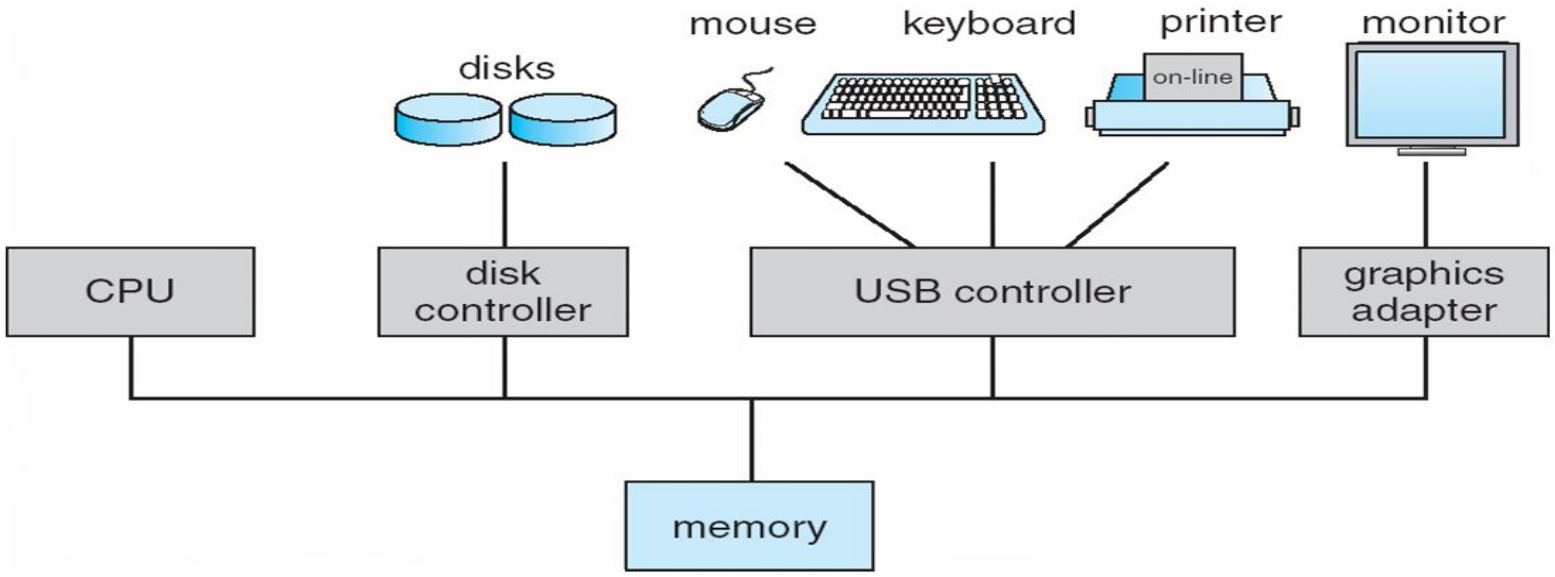
Class	**II B.Sc. IT A**
Course Code	**22ITU05**
Google Classroom Code	**x55ufio**
Map Code	**Focus on - Skill Development**

Content

- **Computer System Organization**
  - **Computer-System Operation**
  - **Interrupt Handling**
  - **I/O Structure**
  - **Storage Structure**
  - **Storage Definitions and Notation Review**
  - **Direct Memory Access Structure**

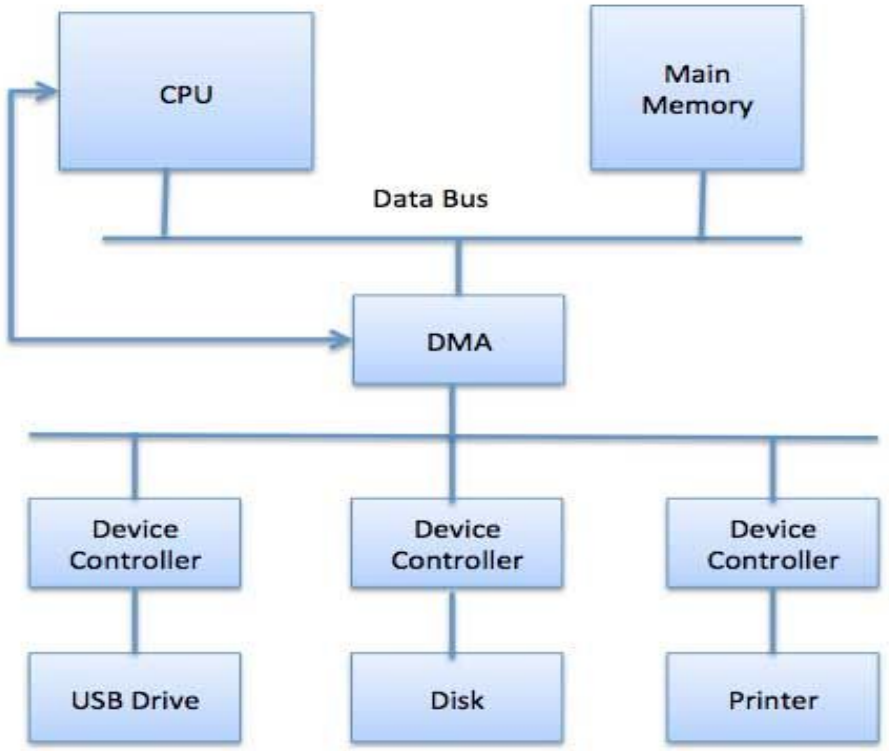
Computer System Organization

- 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 cycles



Computer-System Operation

- I/O devices and the CPU can execute **concurrently**
- Each **device controller** is in charge of a particula device type
- Each device controller has a **local buffer**
- CPU **moves data** from/to main memory to/from loc buffers
- I/O is from the device to local buffer of controller
- Device controller **informs CPU** that it has finished i operation by causing an **interrupt**



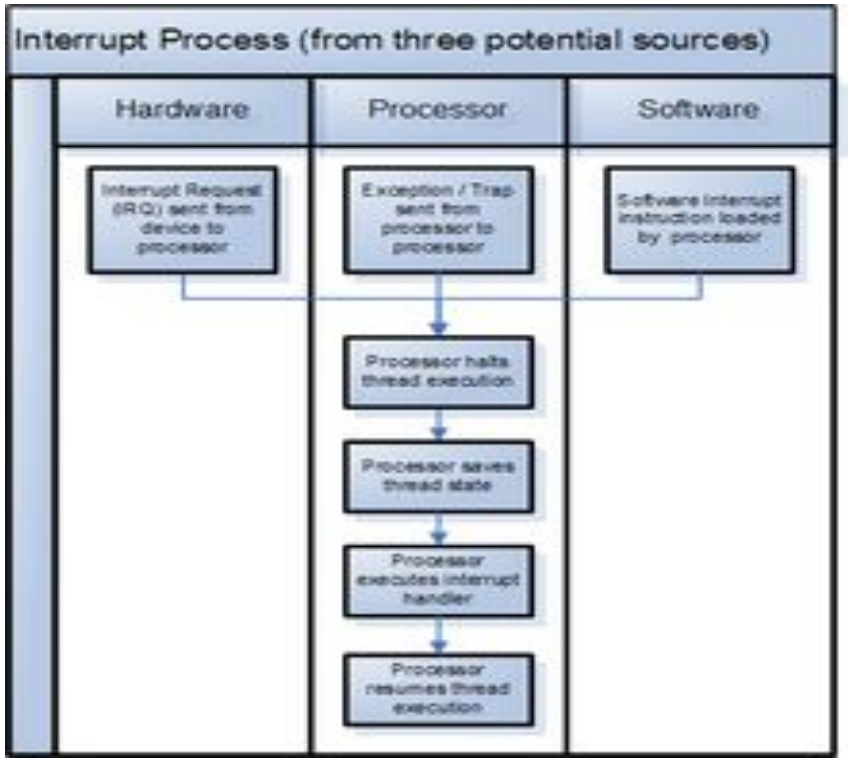
Common Functions of Interrupts

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

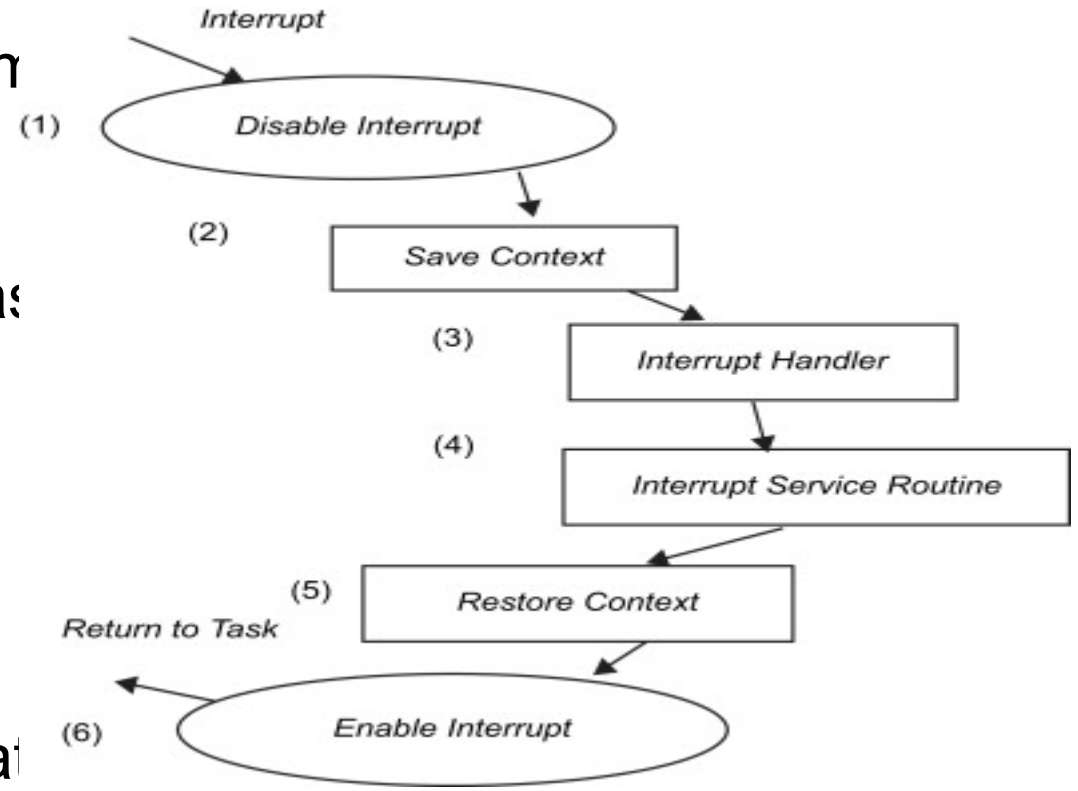
A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request

An operating system is **interrupt driven**

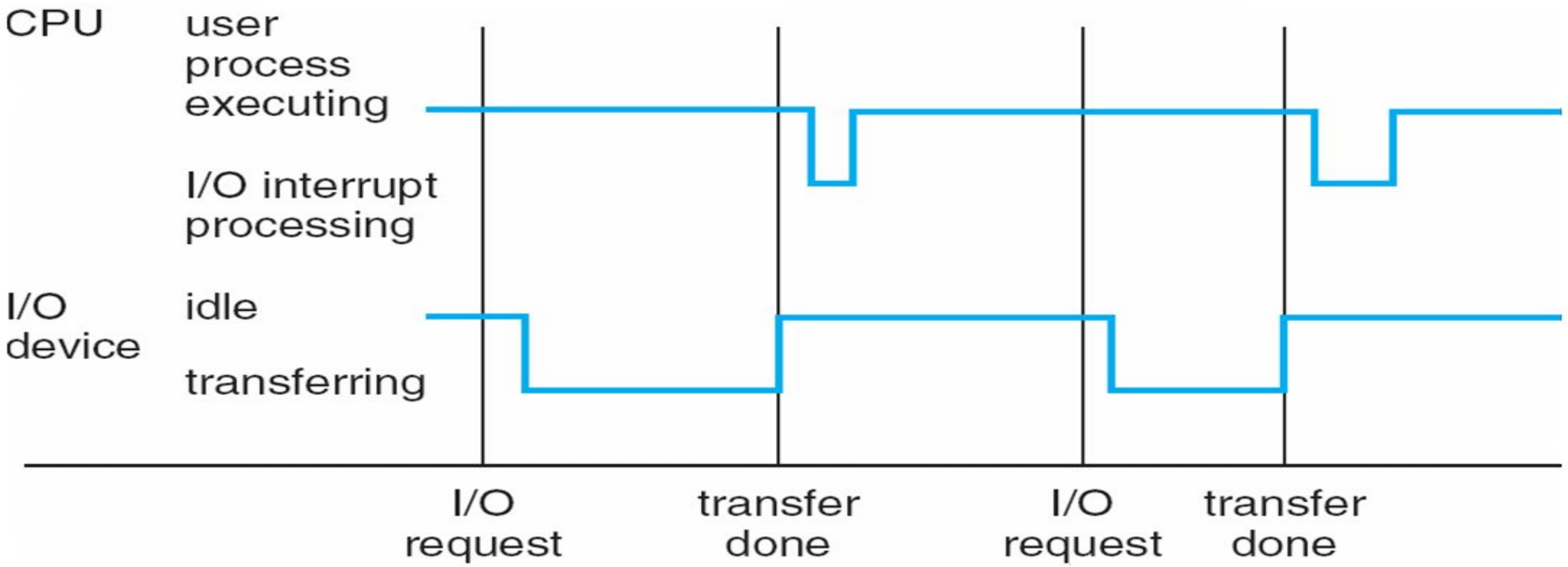


Interrupt Handling

- 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



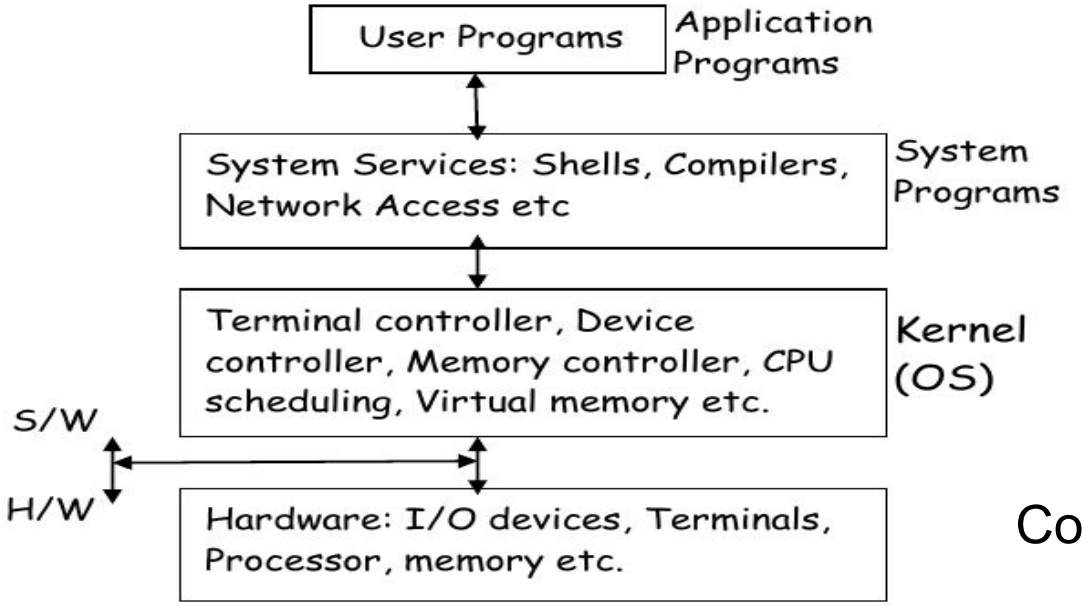
Interrupt Timeline



I/O Structure

After I/O starts, **control returns** to user program only upon I/O completion

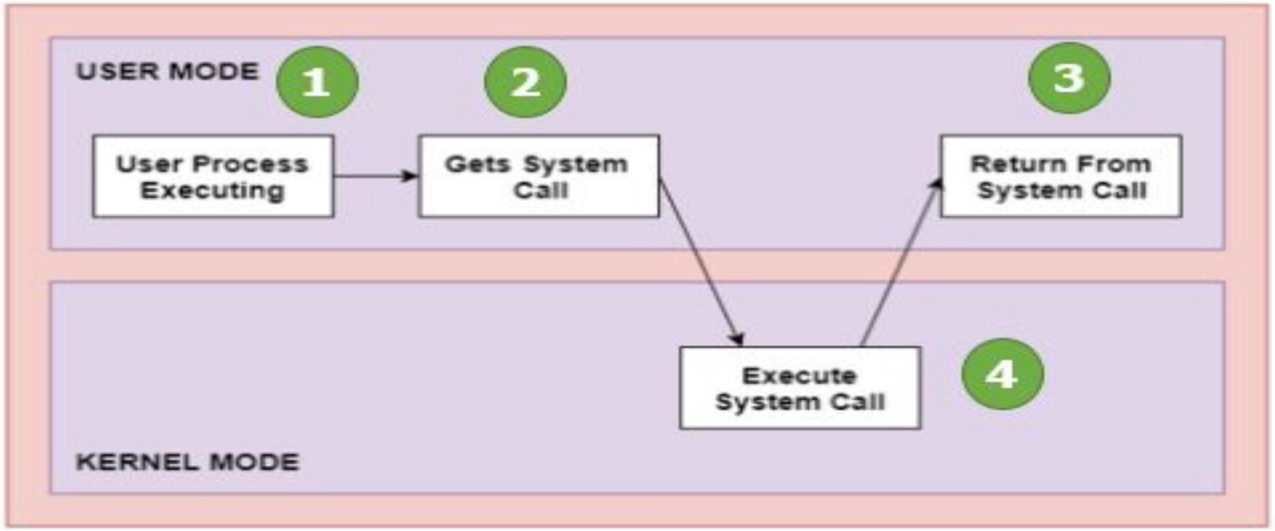
- **Wait** instruction idles the CPU until the next interrupt
- **Wait loop** (contention for memory access)
- At most one **I/O request** is outstanding at a time, no simultaneous **I/O processing**



Cont..

•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

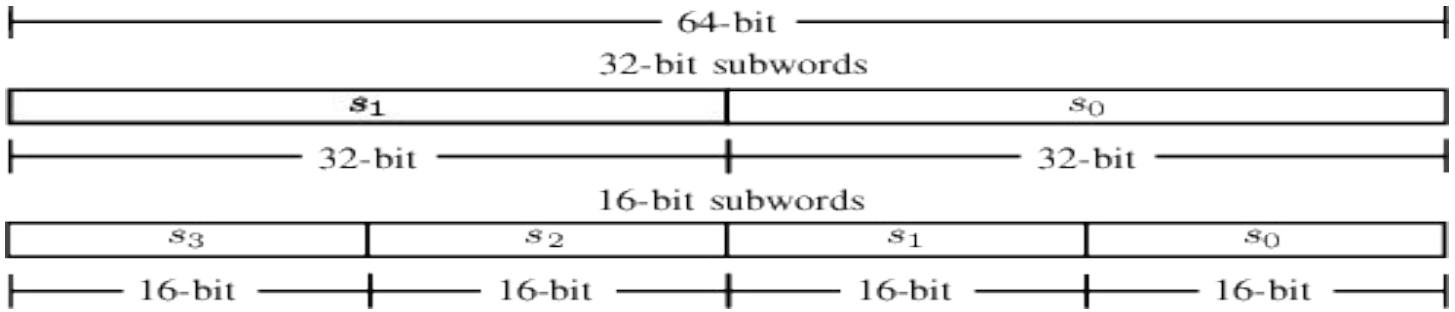


STORAGE DEFINITIONS AND NOTATION REVIEW

- The basic unit of computer storage is the **bit**.
- A bit can contain one of two values, 0 and 1.
- All other storage in a computer is based on collections of bits. Given enough bits, it is amazing how many things a computer can represent: numbers, letters, images, movies, sounds, documents, and programs, to name a few.
- A **byte** is 8 bits, and on most computers it is the smallest convenient chunk of storage.

Cont..

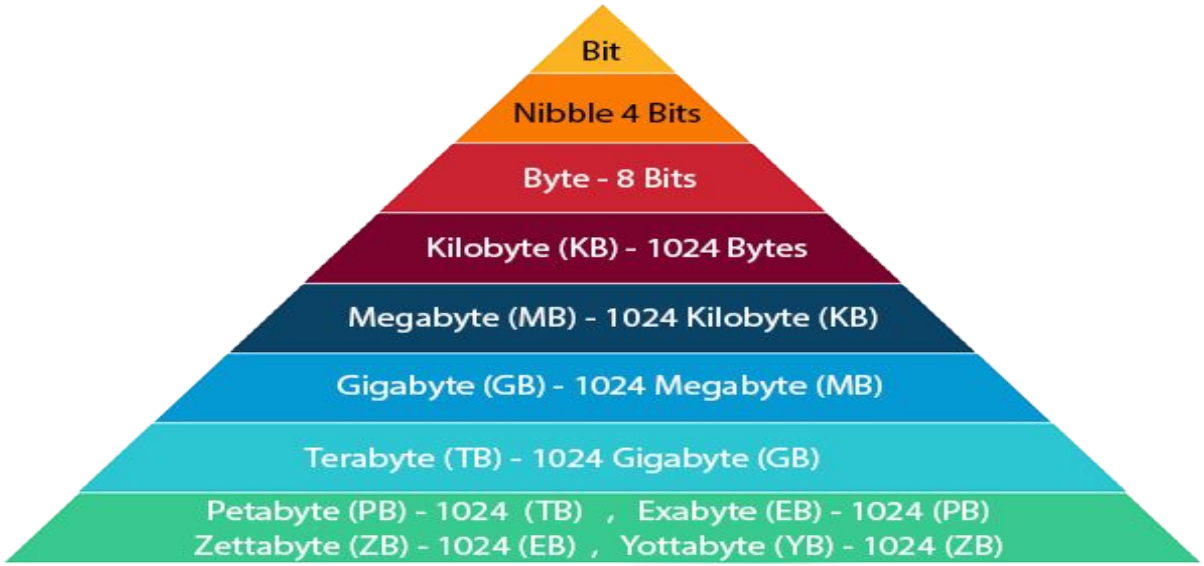
- For **example**, most computers don't have an instruction to move a bit but do have one to move a byte.
- A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes.
- For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words.
- A computer **executes** many operations in its native word size rather than a byte at a time



Cont..

Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes.

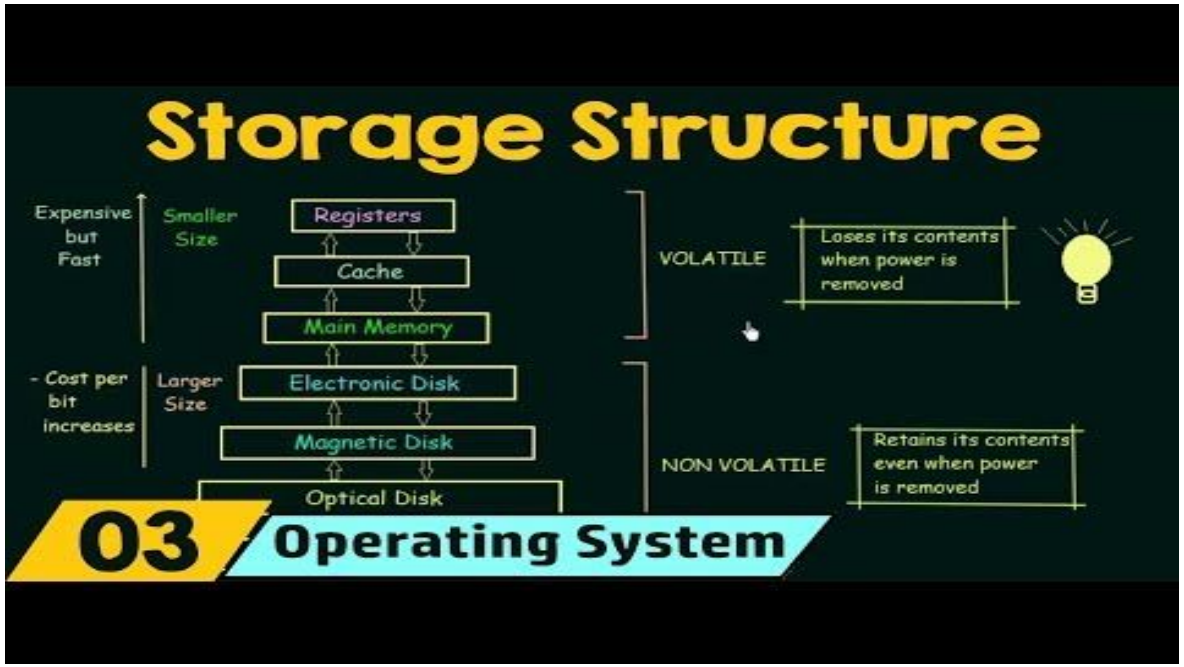
- ✔A **kilobyte**, or **KB**, is 1,024 bytes
- ✔a **megabyte**, or **MB**, is $1{,}024^2$ bytes
- ✔a **gigabyte**, or **GB**, is $1{,}024^3$ bytes
- ✔a **terabyte**, or **TB**, is $1{,}024^4$ bytes
- ✔a **petabyte**, or **PB**, is $1{,}024^5$ bytes



Computer manufacturers often round off these numbers and say that a megabyte is **1 million bytes** and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

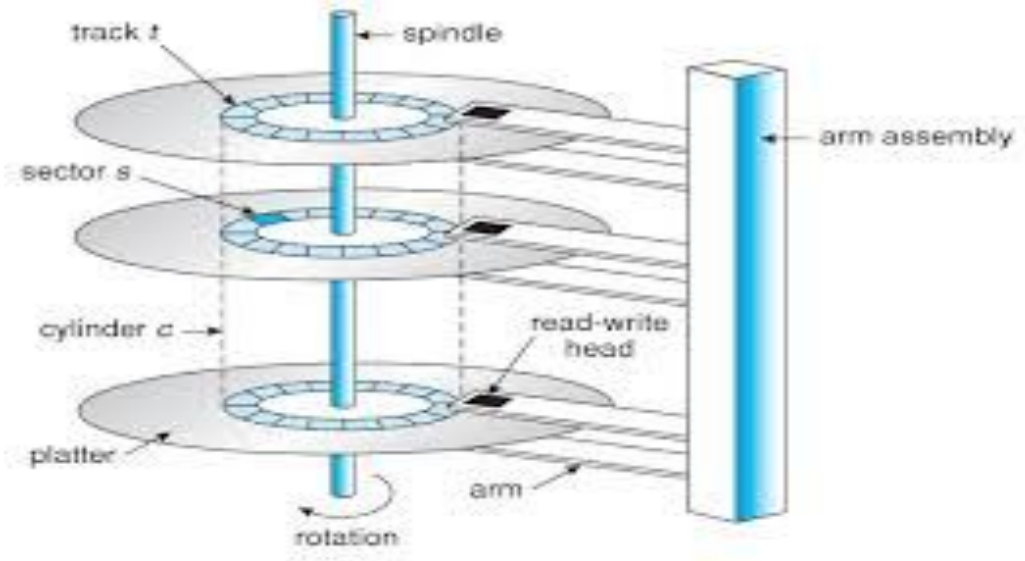
Storage Structure

- 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



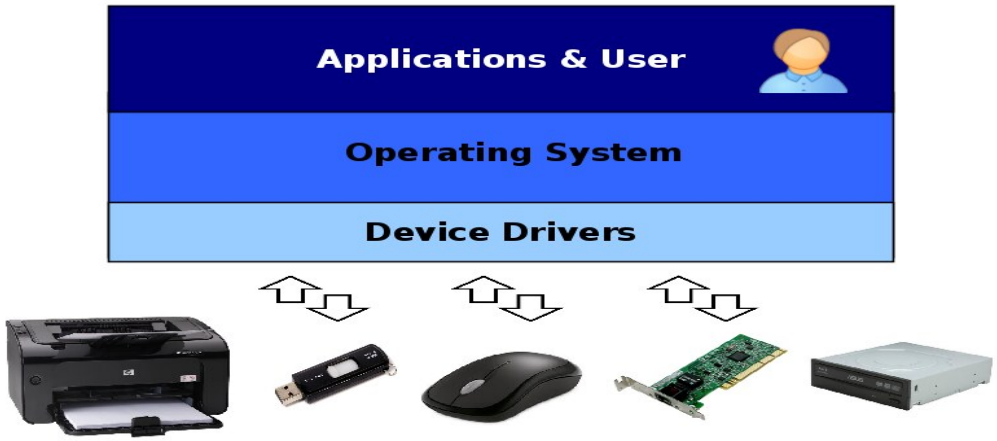
Cont..

- 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
  - Various technologies
  - Becoming more popular

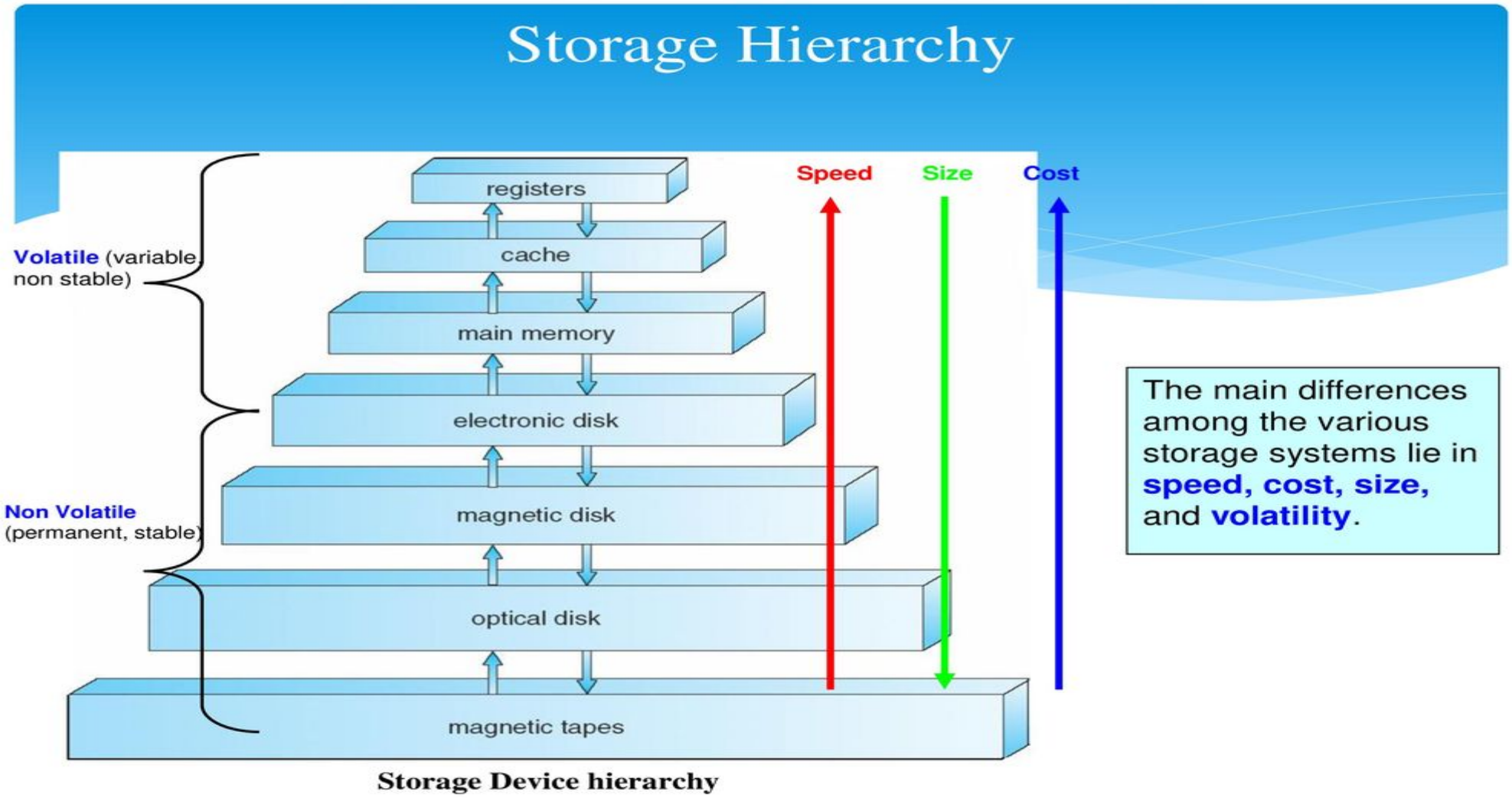


Storage Hierarchy

- 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

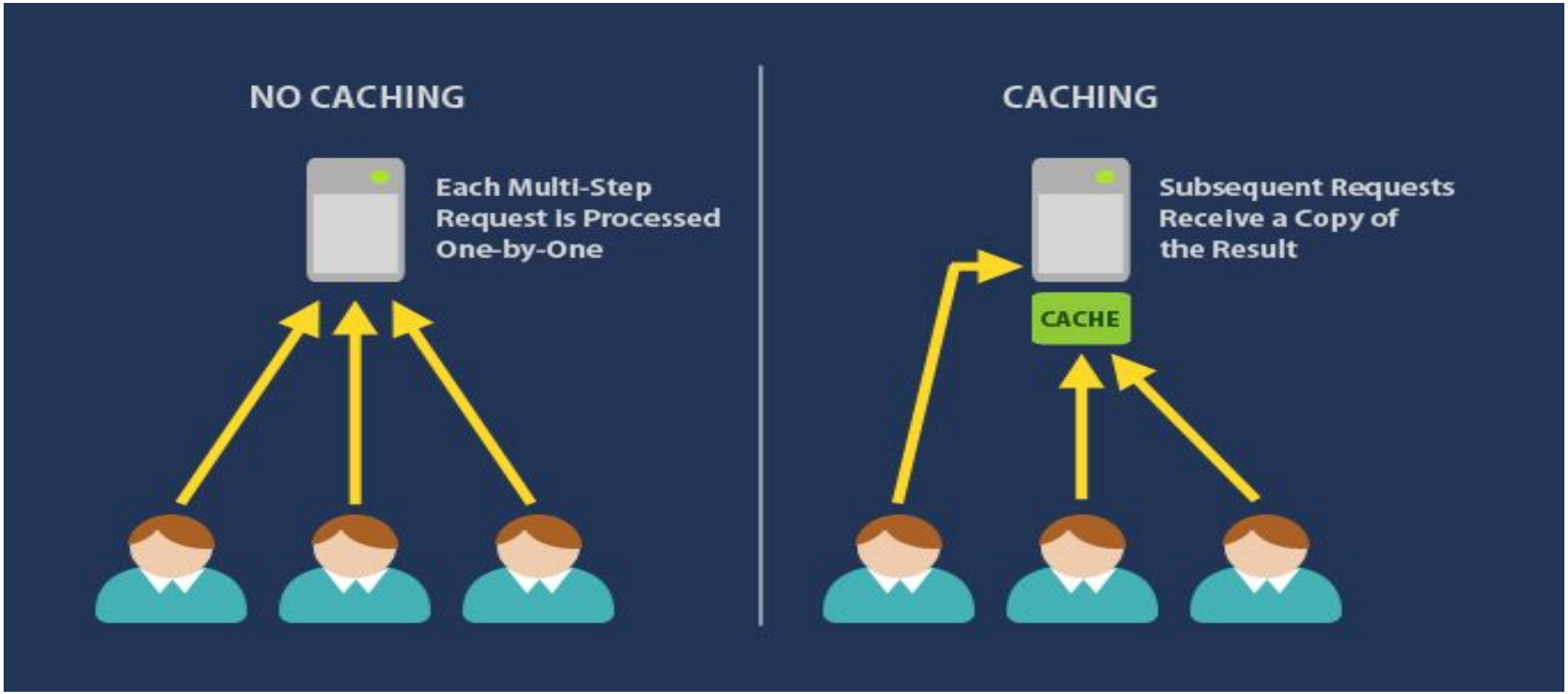


Storage Device hierarchy

Caching

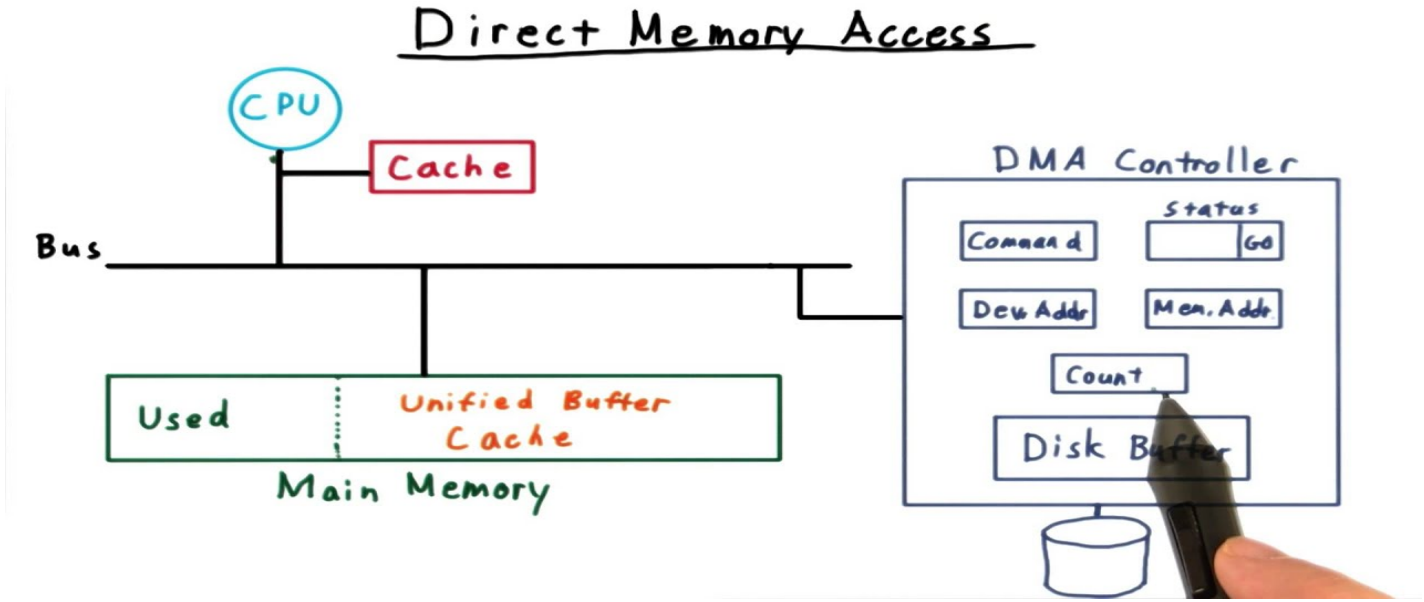
- Important **principle**, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- **Faster storage** (cache) checked first to determine if information is there
  - If it is, information used directly from 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

Cont..


NO CACHING
Each Multi-Step Request Is Processed One-by-One
CACHING
Subsequent Requests Receive a Copy of the Result
CACHE

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



Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 3:Computer System Architecture

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

Computer System Architecture

- Single Processor System
- Multiprocessor System
- Clustered Systems

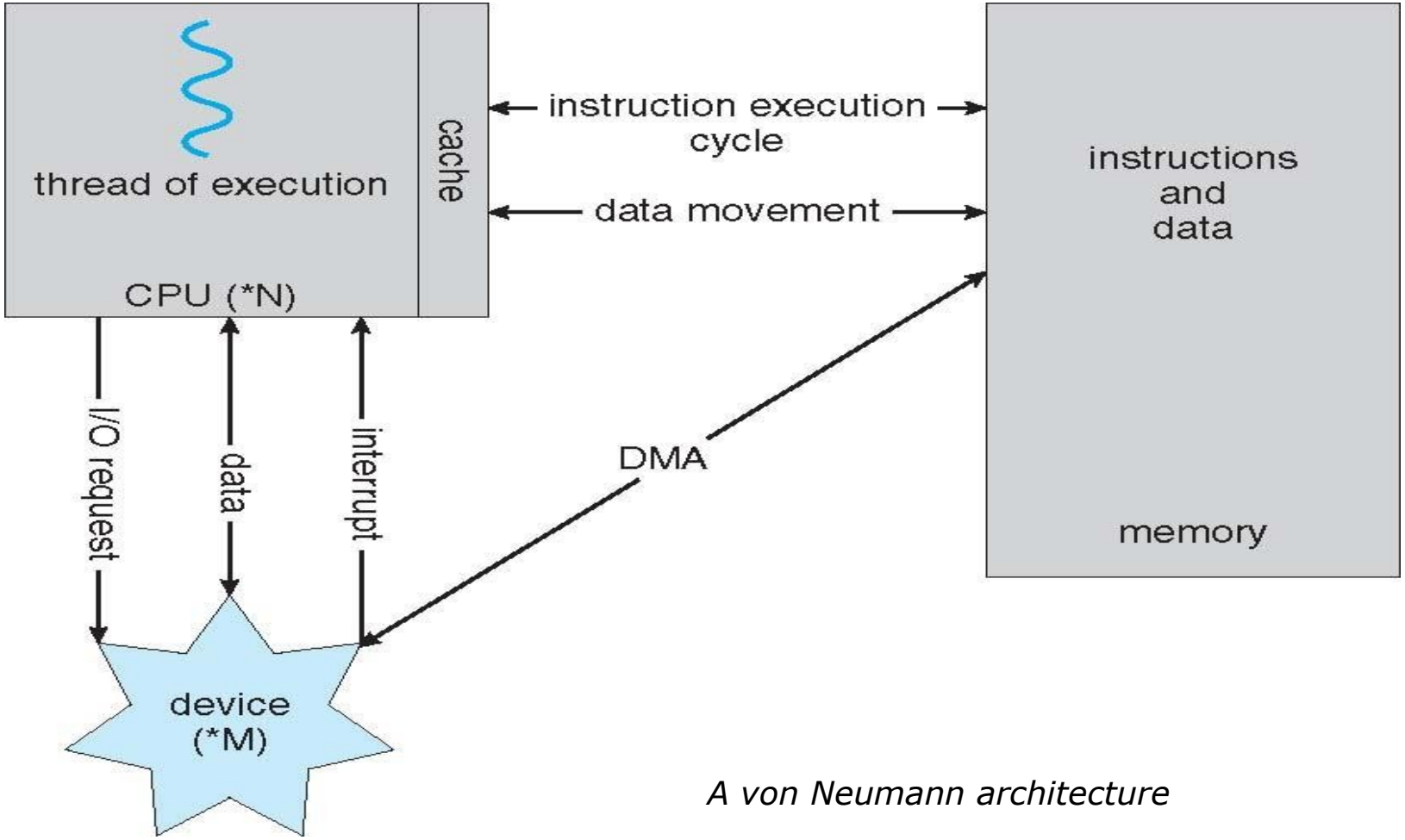
Single Processor System

- Most Systems use a single Processor.
- On a single processor system, there is a main CPU capable of executing a general-purpose instruction set, Including instructions from user processes.
- Almost all system have other special purpose processors as well. They may come in the form of device-specific processors, such as
  - Disk
  - Keyboard, and
  - Graphics controller;

Single Processor System

- All of these special-purpose processors run a limited instructions set and do not run user processes.
- Sometimes they are managed by the operating system sends them information about their next task and monitors their status,
  - **Example** : a disk-controller microprocessor receives a sequence of requests from the main CPU and implements its own disk queue and scheduling algorithm

How a Modern Computer Works



A von Neumann architecture

Multiprocessor System

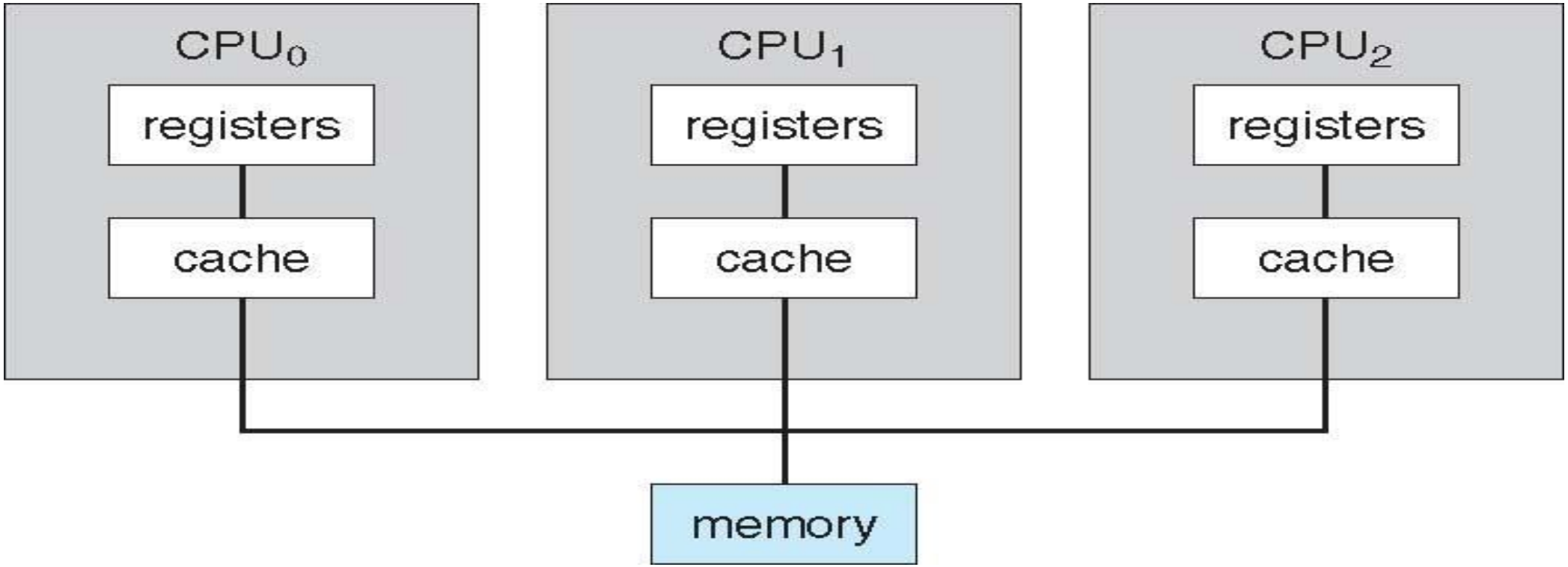
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well

- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**

Multiprocessor System

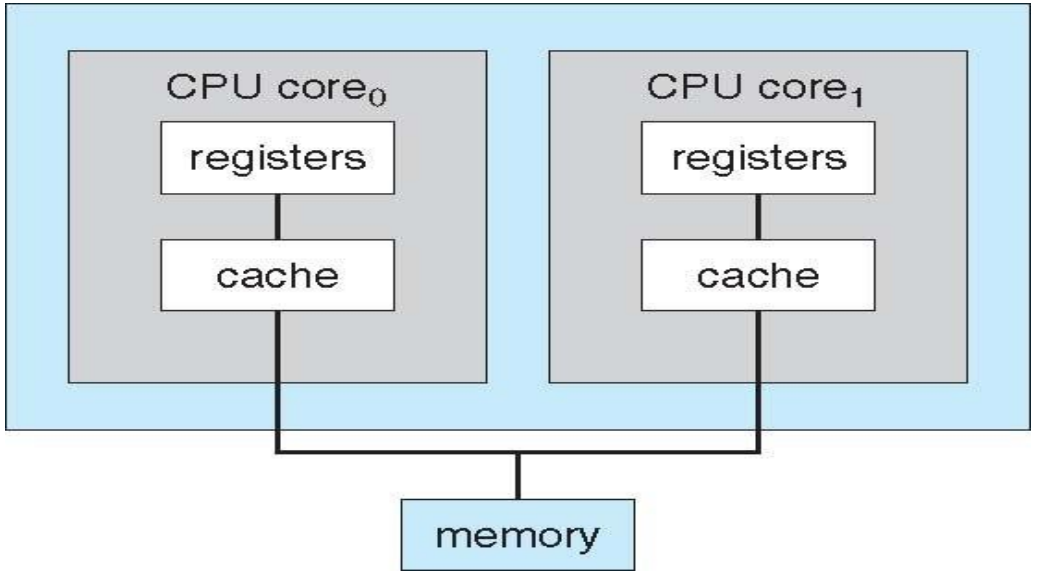
- **Advantages include**:
  1. **Increased throughput**
  2. **Economy of scale**
  3. **Increased reliability** – graceful degradation or fault tolerance
- **Two types**:
  1. **Asymmetric Multiprocessing** – each processor is assigned a specie task.
  2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture



A Dual-Core Design

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



A Dual-Core Design

- Dual-core design with two cores on the same chips.
- In this design each core has its own register set as well as its own local cache.
- Other design might use a shared cache or a combination of local and shared cache
- Aside from architectural considerations, Such as cache, memory, and bus contention, these multicore CPU appear to the operating system as N standard processors

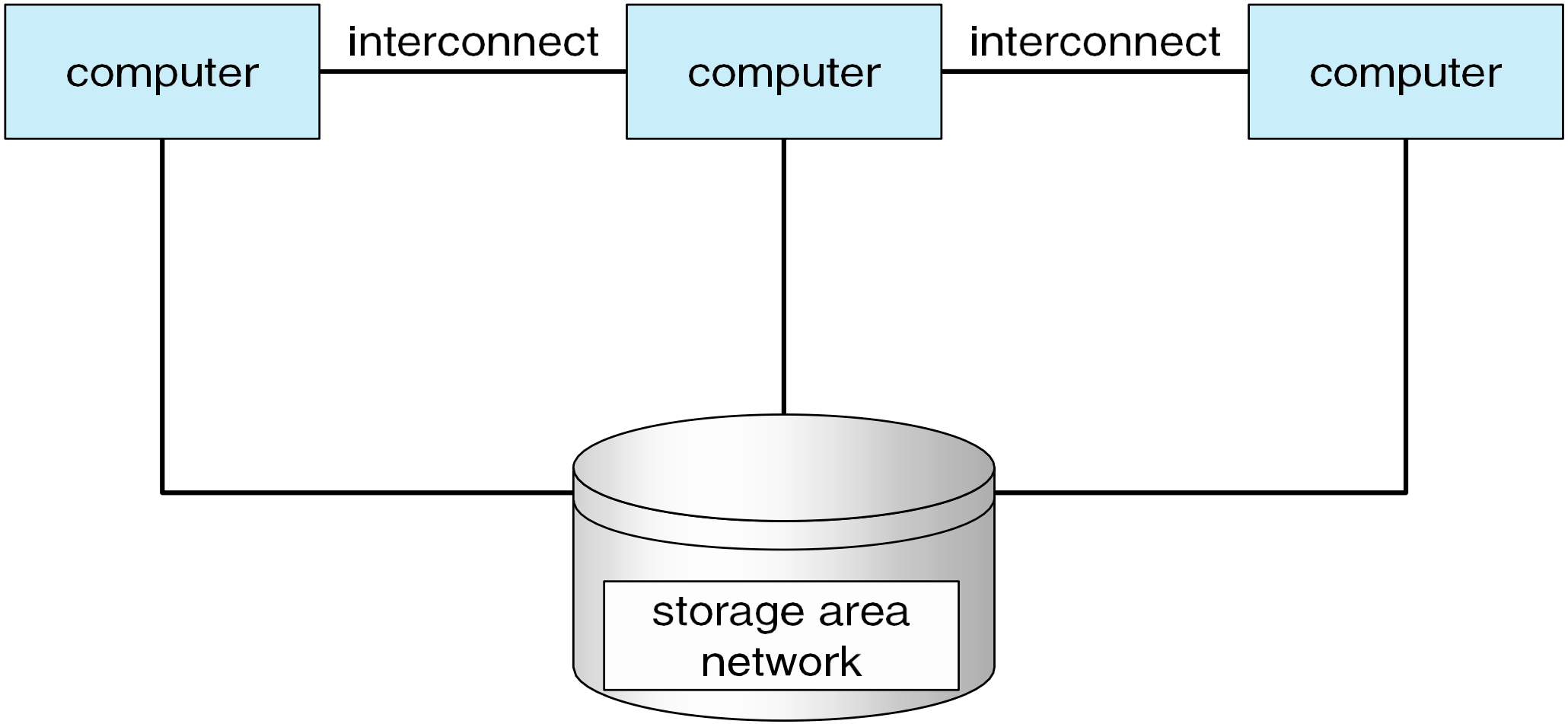
Clustered Systems

- 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

Clustered Systems

- 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



Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 4:Computer System Structure

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

Computer System Structure

- Multiprogramming
- Time sharing(multitasking)

Operating System Structure

- One of the most important aspects of operating systems is the ability to multiprogram.
- A single program cannot, in general the CPU or the I/O devices busy at all times.
- Single users frequently have multiple programs running.
- Multiprogramming increases CPU utilization by organizing jobs(code and data)
- So that the CPU always has one to execute

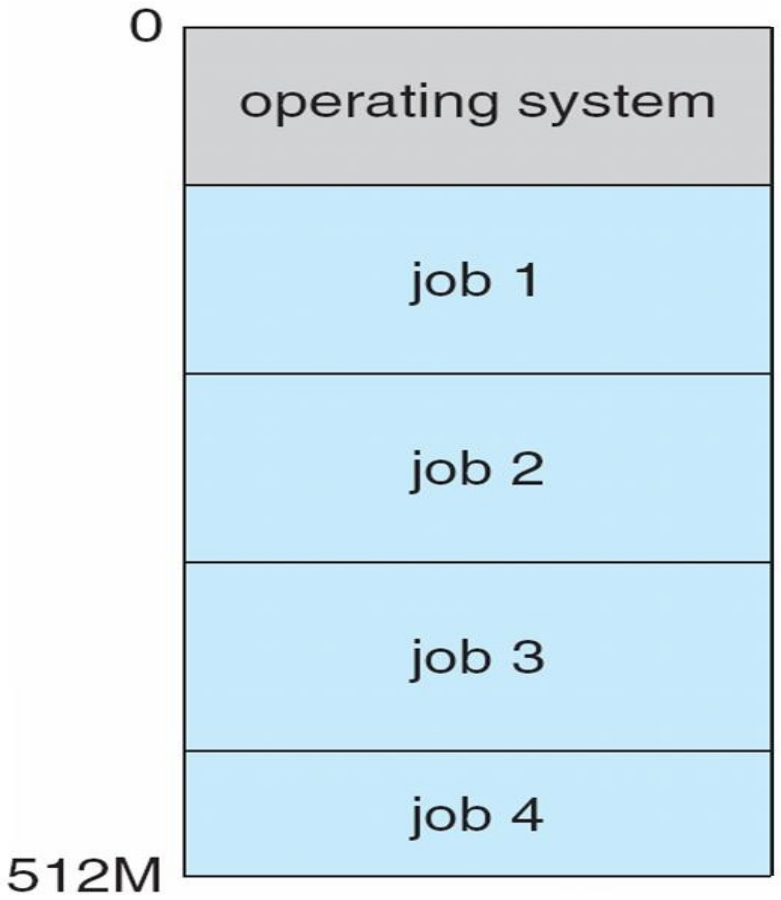
Operating System Structure

- **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

Operating System Structure

- **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

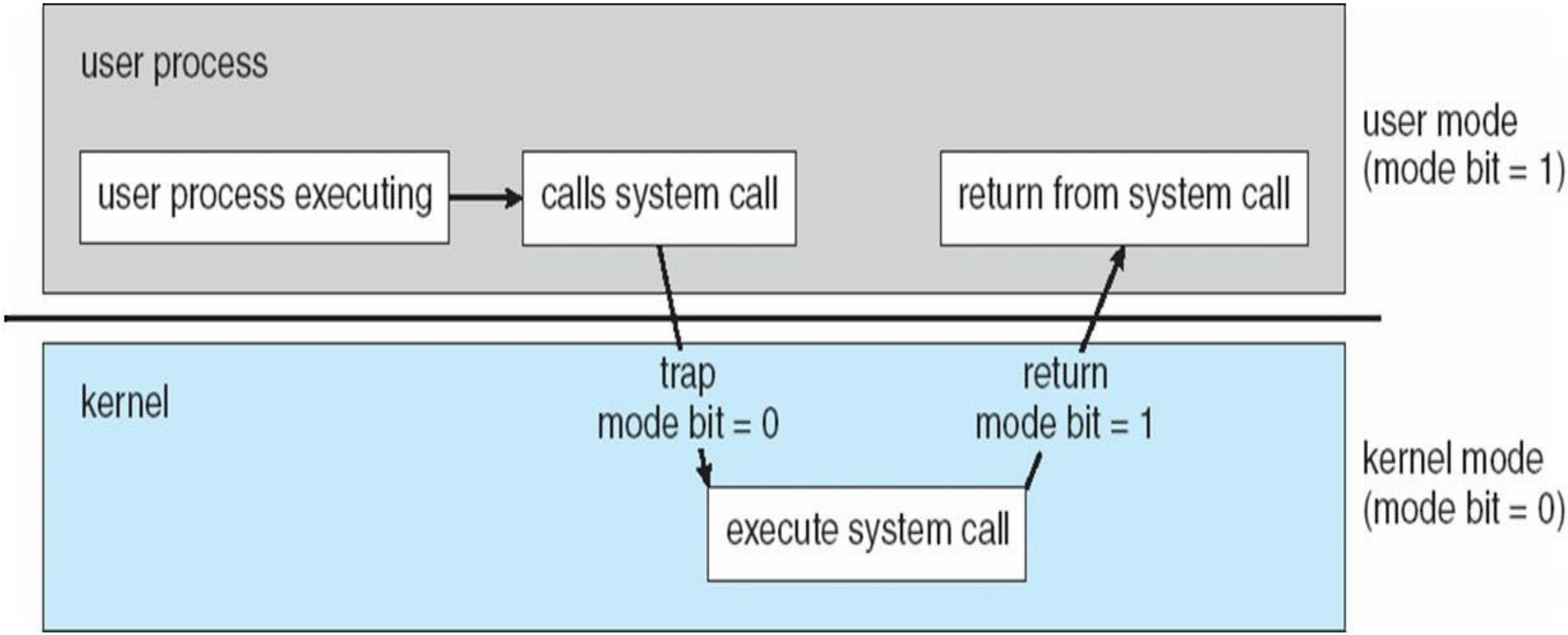
Dual-mode operations

- **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

Dual-mode operations

- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Dual-mode operations



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

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 5:Operating System Operations

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

Operating System Operations

- Dual Mode Operations
- Timer

Dual Mode operations

- In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user defined code.
- The approach taken by most computer systems is to provide hardware support that allows us to differentiate among various modes of execution.
- At the very least, we need two separate modes of operation: **user mode** and **kernel mode** (also called **supervisor mode**, **system mode**, or **privileged mode**).

- A bit, called the **mode bit**, is added to the hardware of the computer to indicate the current mode: kernel (0) or user (1).
- With the mode bit, we are able to distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the 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), it must **transition from user to kernel mode** to fulfill the request. As we shall see, this architectural enhancement is useful for many other aspects of system operation as well.
- At system boot time, the hardware starts in **kernel mode**.

- The operating system is then loaded and starts user applications in **user mode**.
- Whenever a trap or interrupt occurs, the hardware switches from **user mode to kernel mode** (that is, changes the state of the mode bit to 0).
- Thus, whenever the operating system gains control of the computer, it is in **kernel mode**.
- The system always **switches to user mode** (by setting the mode bit to 1) before passing control to a user program.

Timer

- We must ensure that the operating system **maintains control over the CPU**.
- We must **prevent** a user program from **getting stuck in an infinite loop** or not calling system services and never returning control to the operating system.
- To accomplish this goal, we can use a timer.
- A timer can be set to interrupt the computer after a specified period.
- The period may be fixed (for example, 1/60 second) or variable (for example, from 1 millisecond to 1 second).
- A variable timer is generally implemented by a fixed-rate clock and a counter.

- The **operating system** sets the counter.
- Every time the clock ticks, the counter is decremented.
- When the **counter** reaches 0, an interrupt occurs.
- Thus, we can use the timer to prevent a user program from running too long.
- A **simple technique** is to initialize a counter with the amount of time that a program is allowed to run.
- A program with a 7-minute time limit.
- for example, would have its counter initialized to 420.

- Every second, the timer interrupts and the counter is decremented by 1.
- As long as the counter is **positive**, control is returned to the user program.
- When the counter becomes **negative**, the operating system terminates the program for exceeding the assigned time limit.

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 5:Process Management , Memory Management

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

- **Process Management**
  - Process Management Activities
  - Process Architecture
  - Process Control Blocks
- **Memory Management**

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

Process Management (Cont.)

- 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

Process Architecture

Here, is an Architecture diagram of the Process

Stack: The Stack stores temporary data like function parameters, returns addresses, and local variables.

Heap Allocates memory, which may be processed during its run time.

Process Architecture

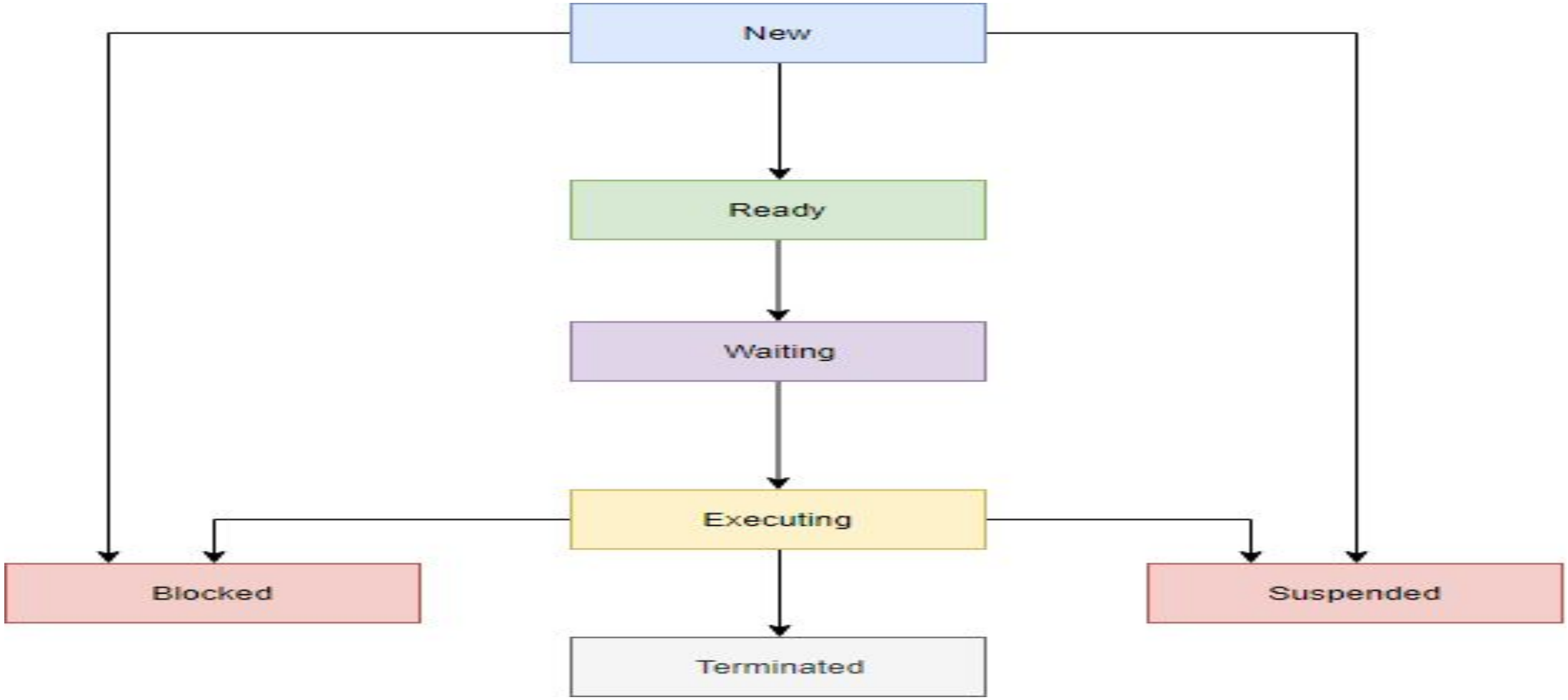
Data: It contains the variable.

Text: Text Section includes the current activity,

which is represented by the value of the Program

Counter.

Process Control Blocks



There are mainly seven stages of a process which are:

- **New:** The new process is created when a specific program calls from secondary memory/ hard disk to primary memory/ RAM
- **Ready:** In a ready state, the process should be loaded into the primary memory, which is ready for execution.
- **Waiting:** The process is waiting for the allocation of CPU time and other resources for execution.

- **Executing:** The process is an execution state.
- **Blocked:** It is a time interval when a process is waiting for an event like I/O operations to complete.
- **Suspended:** Suspendedstate defines the time when a process is ready for execution but has not been placed in the ready queue by OS.
- **Terminated:** Terminated state specifies the time when a process is terminated

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 (Cont.)

- **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

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 7:Storage Management, Protection and Security

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

Operating System Management Services

- Storage Management
- Protection and Security

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)

Storage Management

- **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

Mass-Storage Management

- **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)

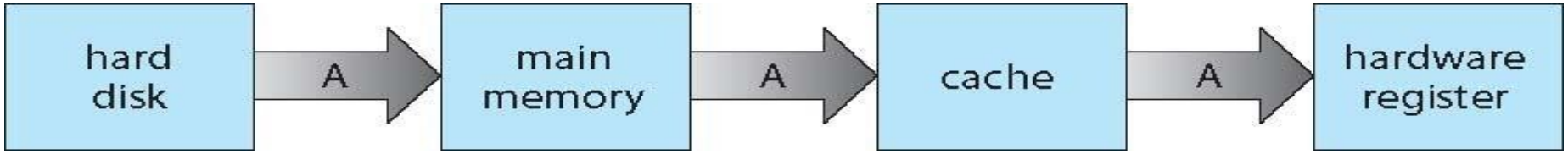
Performance of Various Levels of

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

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

Protection and Security (Cont.)

- 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

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 8:Kernal data Structure, Computing Environment, Open Source Operating Systems

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

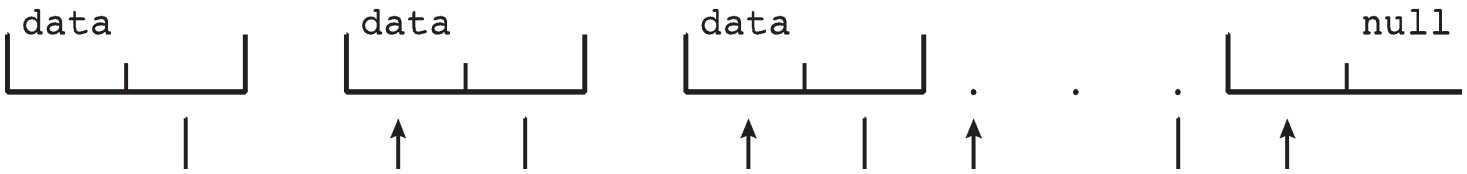
Content

- Kernal data Structures
- Computing Environment
- Open Source Operating Systems

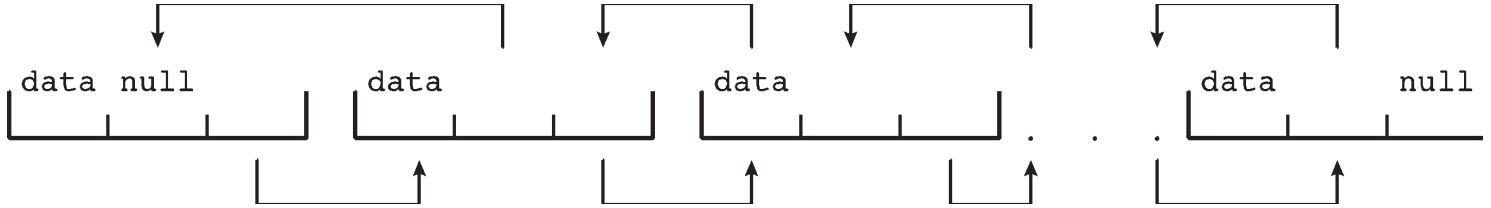
Kernel Data Structures

Many similar to standard programming data structures

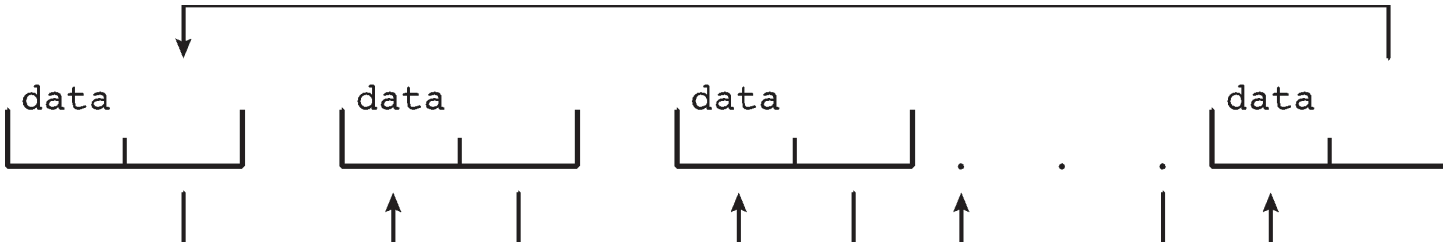
Singly linked list



Doubly linked list



Circular linked list



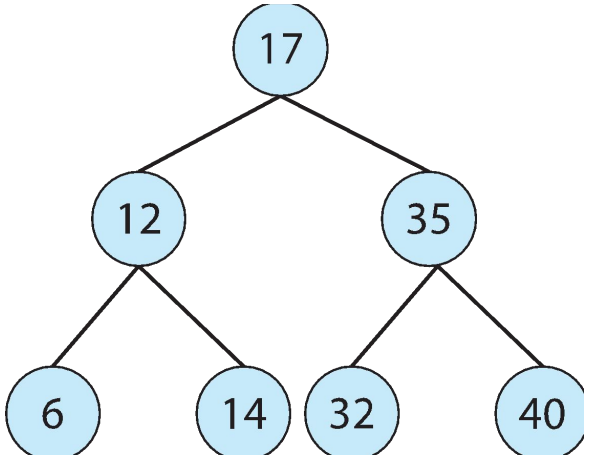
Kernel Data Structures

Binary search tree

left <= right

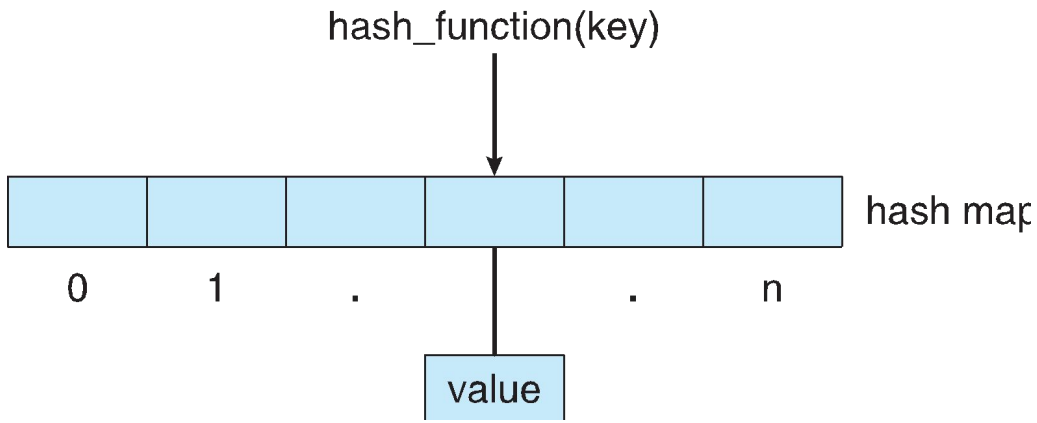
Search performance is *O(n)*

Balanced binary search tree is *O(lg n)*



Kernel Data Structures

Hash function can create a **hash map**



Bitmap – string of *n* binary digits representing the status of *n* items
Linux data structures defined in ***include*** files <linux/list.h>, <linux/kfifo.h>, <linux/rbtree.h>

Computing Environments

- Traditional
- Mobile
- Client Server
- Peer-to-Peer
- Cloud computing
- Real-time Embedded

Traditional

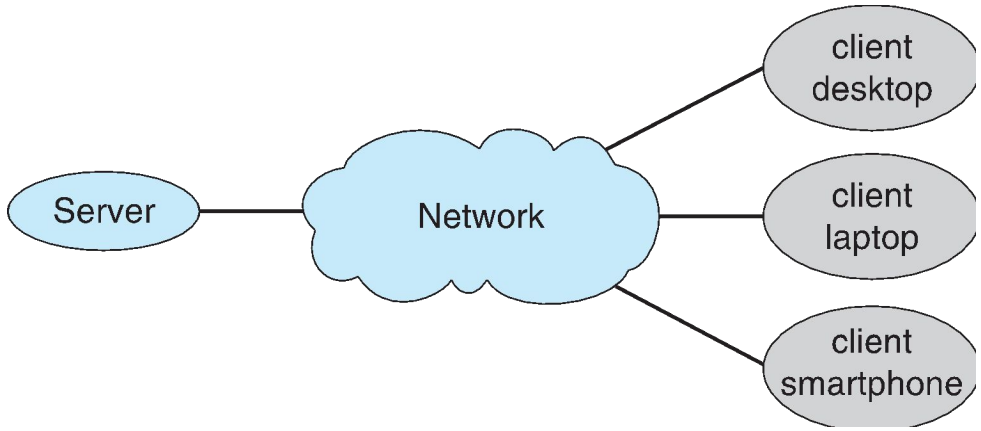
- Stand-alone general-purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Mobile

- Handheld smartphones, tablets, etc.
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

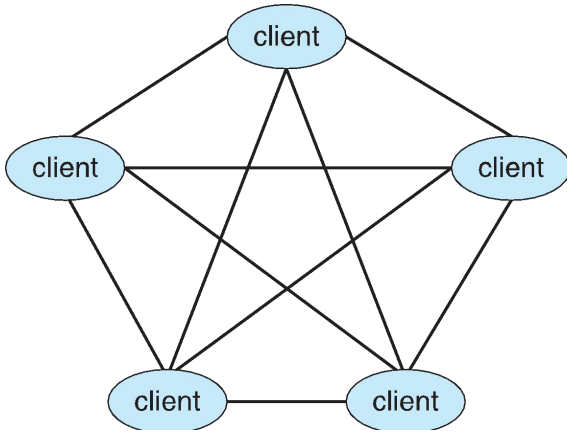
Client Server

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



Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype

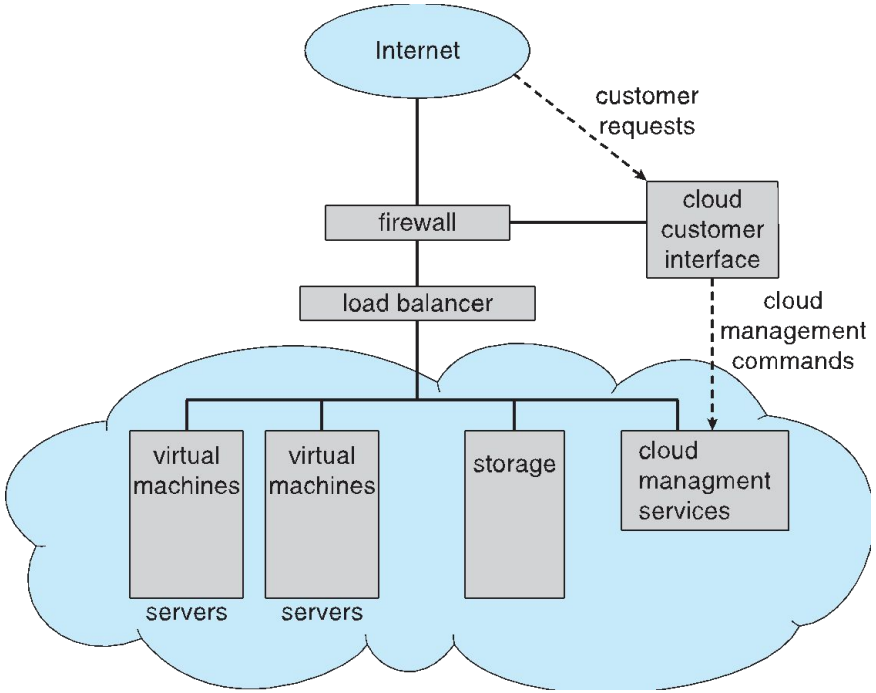


Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage

Cloud Computing (cont.)

- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



Real-Time Embedded Systems

- Real-time embedded systems most prevalent form of computers
  - Vary considerable, special purpose, limited purpose OS, **real-time OS**
  - Use expanding
- Many other special computing environments as well
  - Some have OSes, some perform tasks without an OS
- Real-time OS has well-defined fixed time constraints
  - Processing ***must*** be done within constraint
  - Correct operation only if constraints met

Free and Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source** and **proprietary**
- Counter to the **copy protection** and **Digital Rights Management** (**DRM**) movement
- Started by **Free Software Foundation** (**FSF**), which has "copyleft" **GNU Public License** (**GPL**)
  - Free software and open-source software are two different ideas championed by different groups of people
    - **https://www.gnu.org/philosophy/open-source-misses-the-point.en.html**

- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Unit 2 Lecture 1

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

- **Operating System Services**
  - User interface
  - Program execution
  - I/O operations
  - File-system manipulation
  - Communications
  - Error detection
- **Operating System Services**
  - Resource allocation
  - Accounting
  - Protection and security
- **User Operating System Interface concepts**
- **Touchscreen Interfaces**
- **The Mac OS X GUI**

Operating System Services

- Operating systems provide an environment for execution of programs and services to programs and users
- 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**

Operating System Services

- **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

Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - **Communications** – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)

Operating System Services (Cont.)

- **Error detection** – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

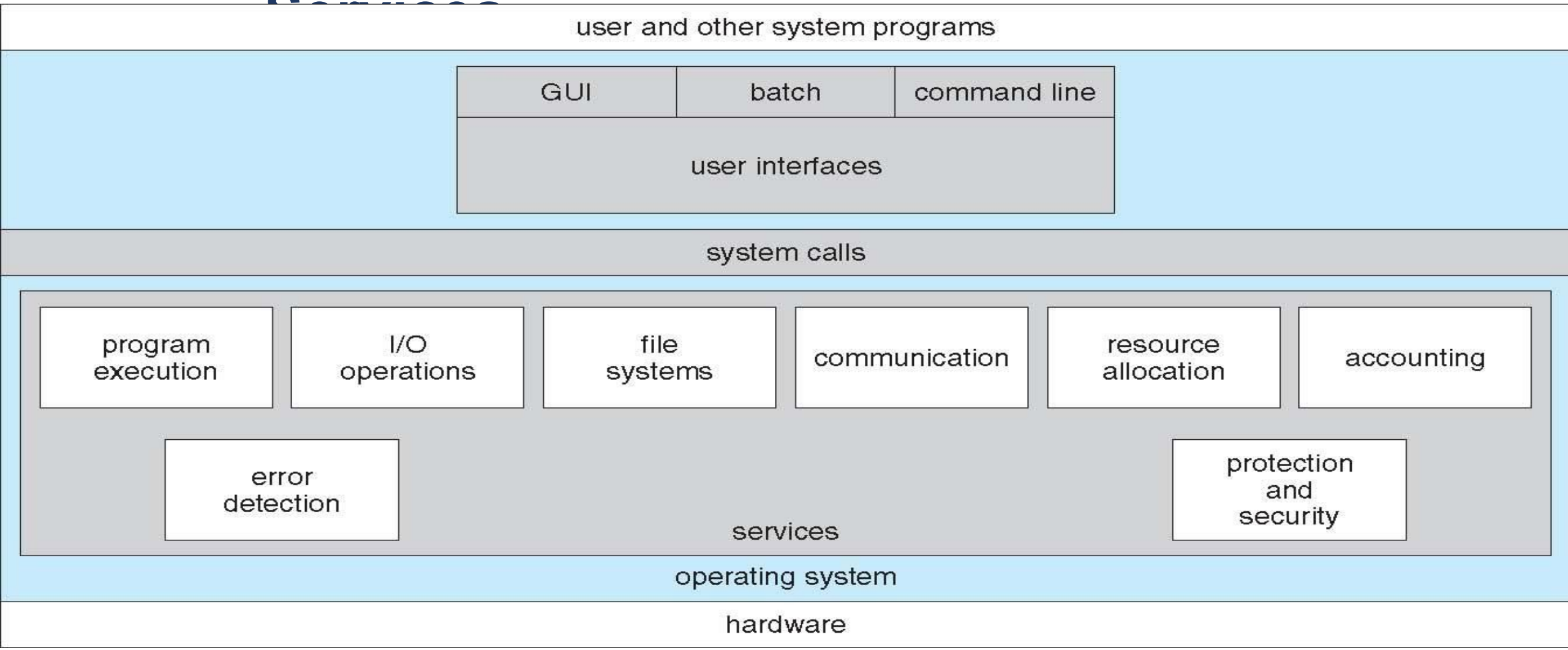
Operating System Services (Cont.)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - CPU cycles, main memory, file storage, I/O devices.
  - **Accounting** - To keep track of which users use how much and what kinds of computer resources

Operating System Services (Cont.)

- **Protection and security** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
  - **Protection** involves ensuring that all access to system resources is controlled
  - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

A View of Operating System Services

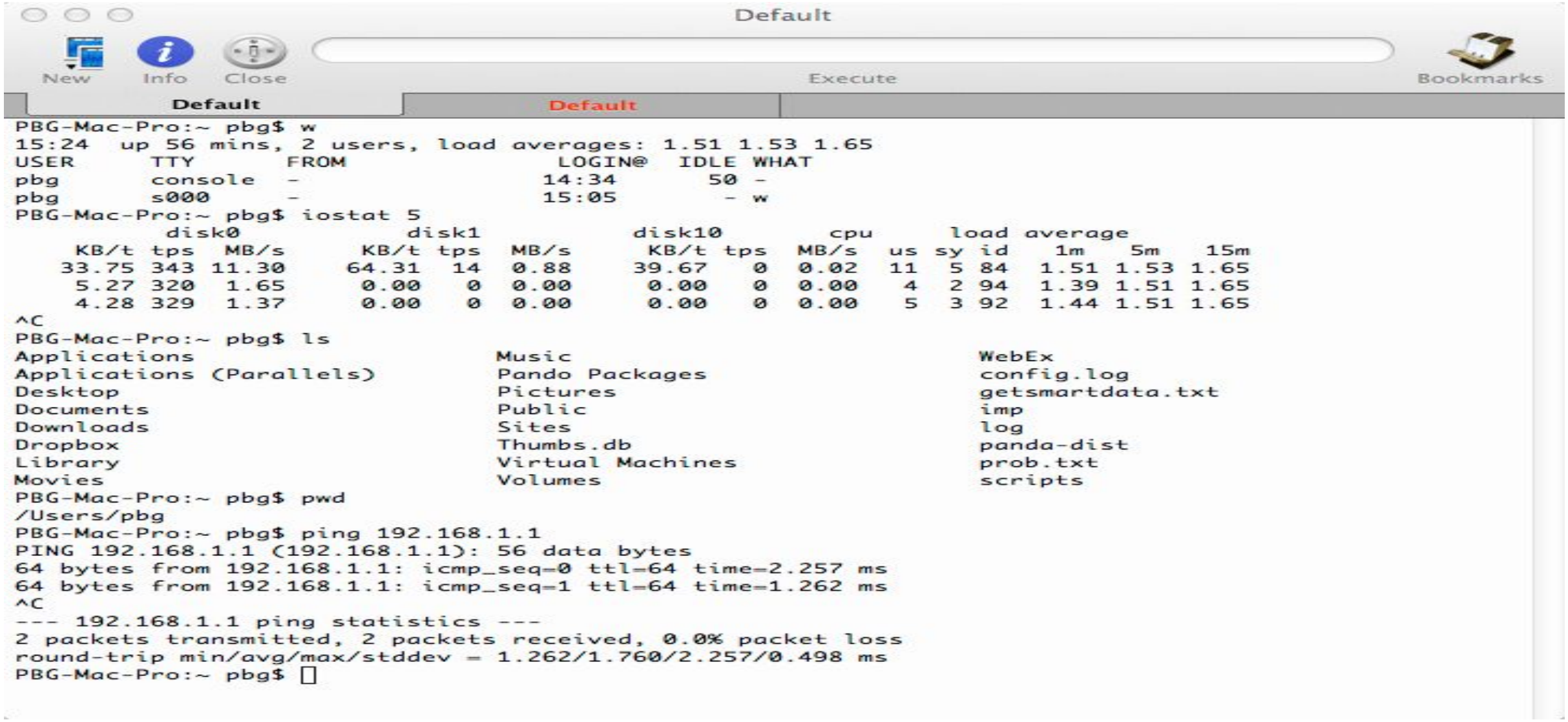


User Operating System Interface - CLI

CLI or **command interpreter** allows direct command entry

- Sometimes implemented in kernel, sometimes by systems program
- Sometimes multiple flavors implemented – **shells**
- Primarily fetches a command from user and executes it
- Sometimes commands built-in, sometimes just names of programs
  - If the latter, adding new features doesn't require shell modification

Bourne Shell Command Interpreter



User Operating System Interface - GUI

- User-friendly **desktop** metaphor interface
  - Usually mouse, keyboard, and monitor
  - **Icons** represent files, programs, actions, etc
  - Various mouse buttons over objects in the interface cause various actions(provide information, options, execute function, open directory (known as a **folder**)
  - Invented at Xerox PARC

User Operating System Interface - GUI

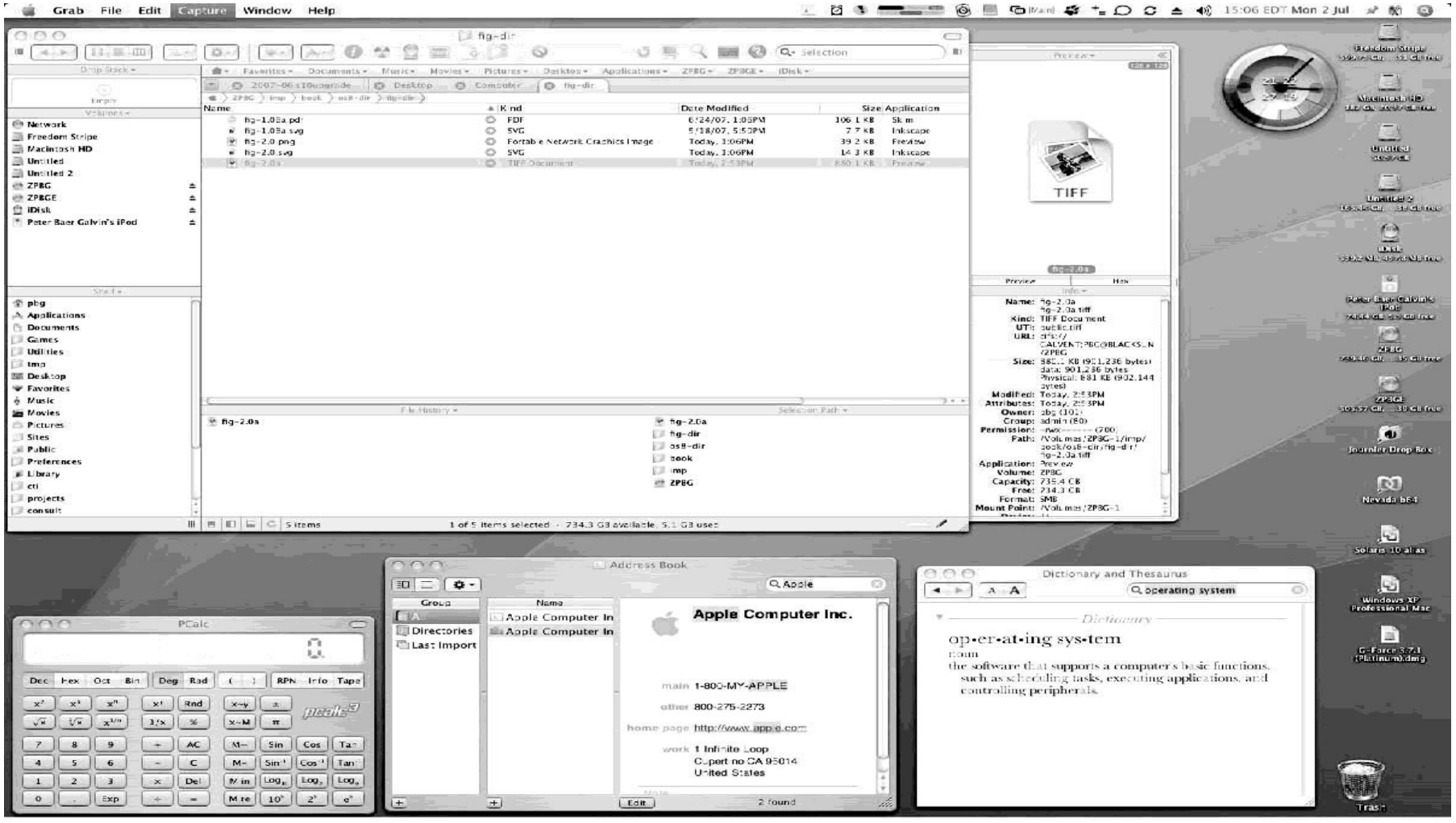
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI “command” shell
  - Apple Mac OS X is “Aqua” GUI interface with UNIX kernel underneath and shells available
  - Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

Touchscreen Interfaces

Touchscreen devices require new interfaces

- Mouse not possible or not desired
- Actions and selection based on gestures
- Virtual keyboard for text entry
- Voice commands.

The Mac OS X GUI



Summary

- Operating systems provide an environment for execution of programs and services to programs and users
  - Error detection – OS needs to be constantly aware of possible errors
  - Protection involves ensuring that all access to system resources is controlled
- Security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
  - Sometimes commands built-in, sometimes just names of programs
  - User-friendly desktop metaphor interface
  - Many systems now include both CLI and GUI interfaces

Test Your knowledge

CLICK HERE

THANK YOU

SRI KRISHNA ARTS AND SCIENCE COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY AND COGNITIVE SYSTEMS

OPERATING SYSTEM

Lecture 2:System Call

Class	II B.Sc. IT A
Course Code	22ITU05
Google Classroom Code	x55ufio
Map Code	Focus on - Skill Development

Content

System Calls

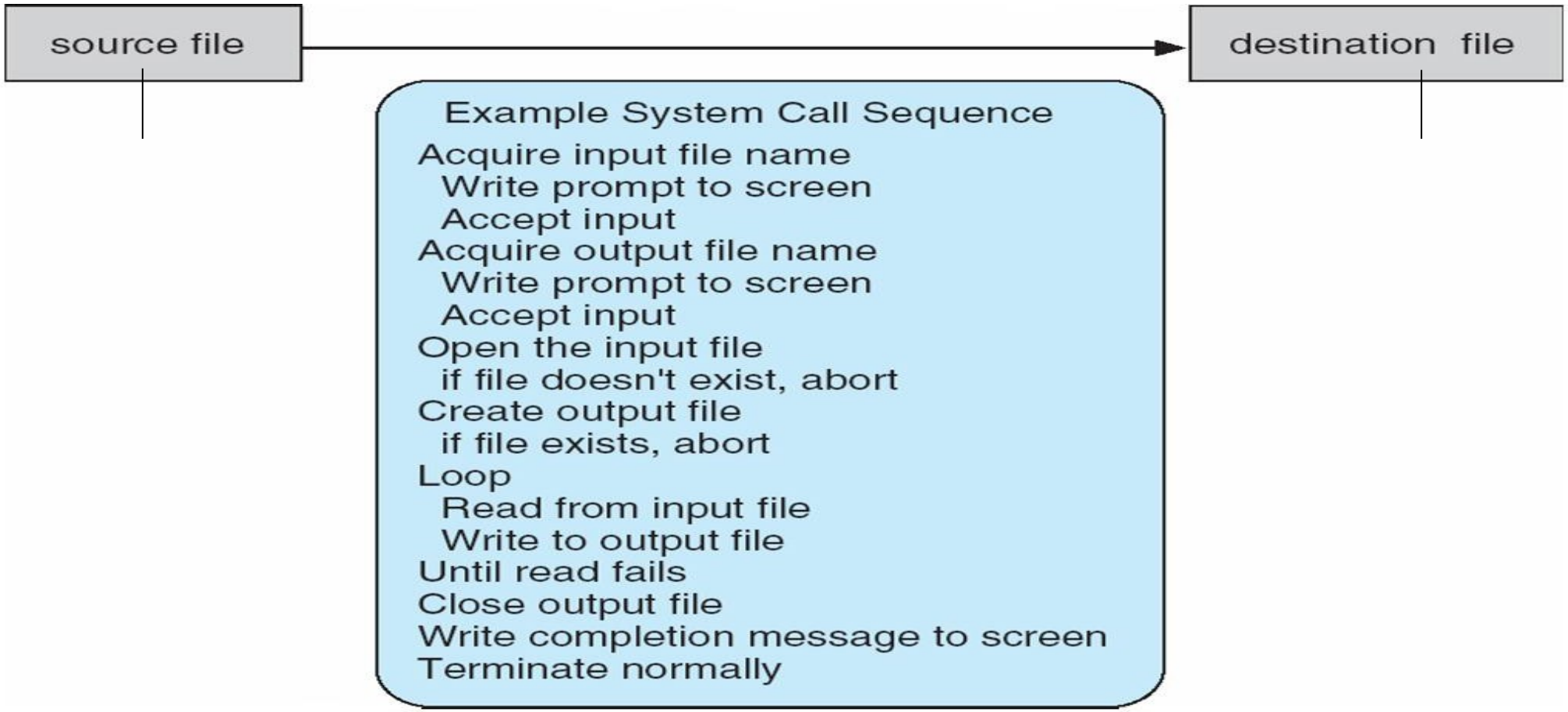
- API – System Call – OS Relationship
- System Call Parameter Passing
- Parameter Passing via Table
- Types of System calls

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 Programming 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)

Example of System Calls

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



Example of Standard API

EXAMPLE OF STANDARD API

As an example of a standard API, consider the `read()` function that is available in UNIX and Linux systems. The API for this function is obtained from the `man` page by invoking the command

```
man read
```

on the command line. A description of this API appears below:

```
#include <unistd.h>

ssize_t     read(int fd, void *buf, size_t count)
```

return value — function name — parameters

A program that uses the `read()` function must include the `unistd.h` header file, as this file defines the `ssize_t` and `size_t` data types (among other things). The parameters passed to `read()` are as follows:

- `int fd`—the file descriptor to be read
- `void *buf`—a buffer where the data will be read into
- `size_t count`—the maximum number of bytes to be read into the buffer

On a successful read, the number of bytes read is returned. A return value of 0 indicates end of file. If an error occurs, `read()` returns −1.

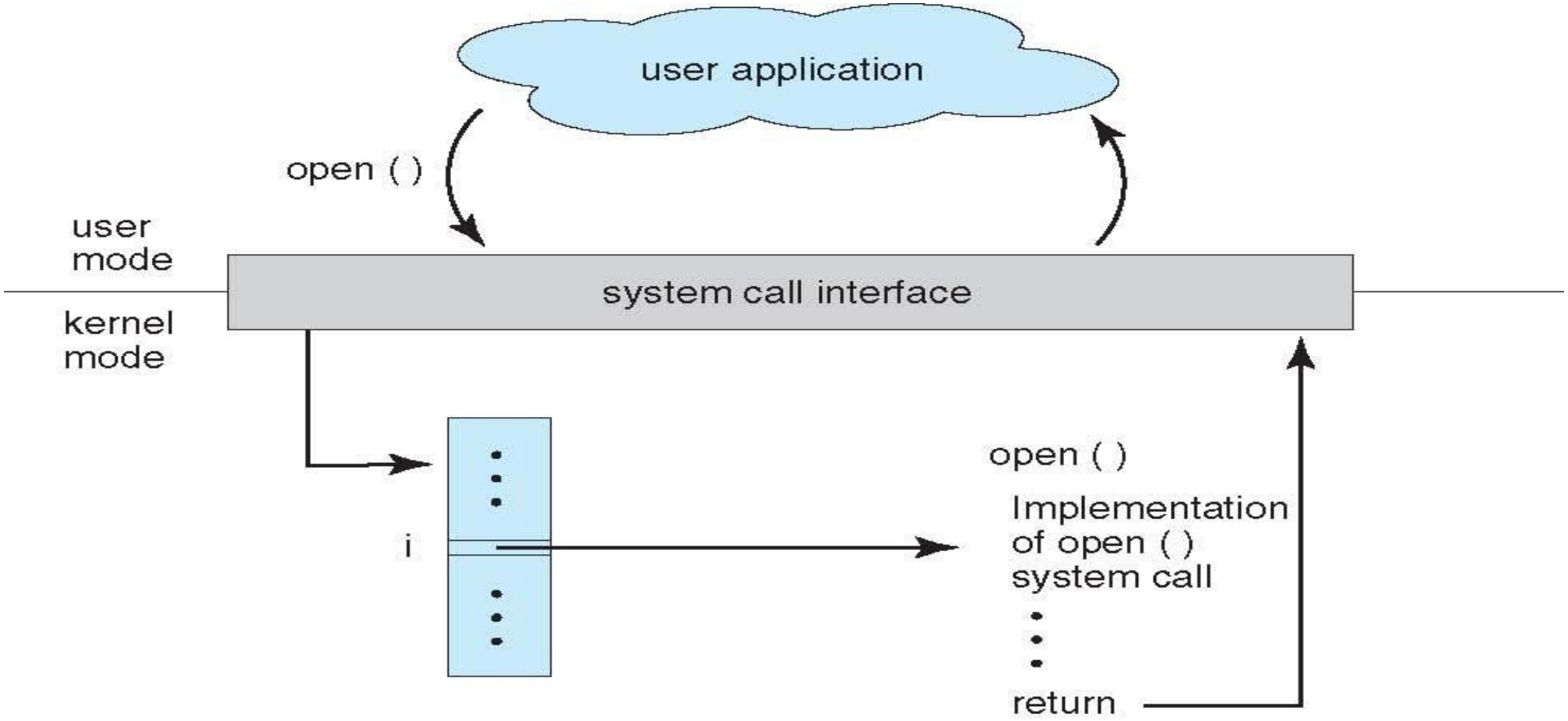
System Call Implementation

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers
- The system call interface invokesthe intended system call in OS kernel and returns status of the system call and any return values

System Call Implementation

- The caller need know nothing about how the system call is implemented
  - Just needs to obey API and understand what OS will do as a result call
  - Most details of OS interface hidden from programmer by API
    - Managed by run-time support library (set of functions built into libraries included with compiler)

API – System Call – OS Relationship



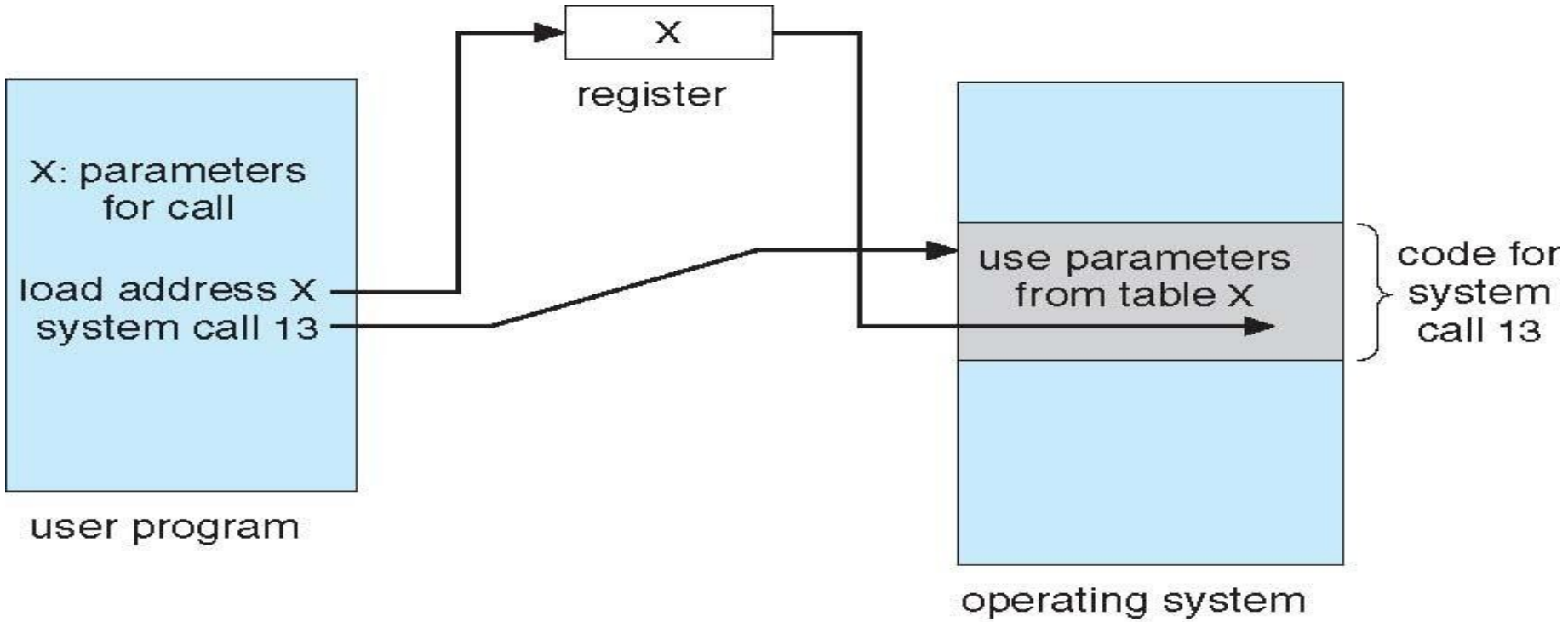
System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in registers
    - In some cases, may be more parameters than registers

System Call Parameter Passing

- Parameters stored in a block, or table, in memory, and address of block passed as a parameter in a register
  - This approach taken by Linux and Solaris
- Parameters placed, or **pushed**, onto the **stack** by the program and **popped** off the stack by the operating system
- Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table



Types of System Calls

- **Process control**
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
  - Dump memory if error
  - **Debugger** for determining **bugs, single step** execution
  - **Locks** for managing access to shared data between processes

Types of System Calls

- **File management**
  - create file, delete file
  - open, close file
  - read, write, reposition
  - get and set file attributes
- **Device management**
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices

Types of System Calls (Cont.)

- **Information maintenance**
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes
- **Communications**
  - create, delete communication connection
  - send, receive messages if **message passing model** to **host name** or **process name (**From **client** to **server)**
  - **Shared-memory model** create and gain access to memory regions
  - transfer status information
  - attach and detach remote devices

Types of System Calls (Cont.)

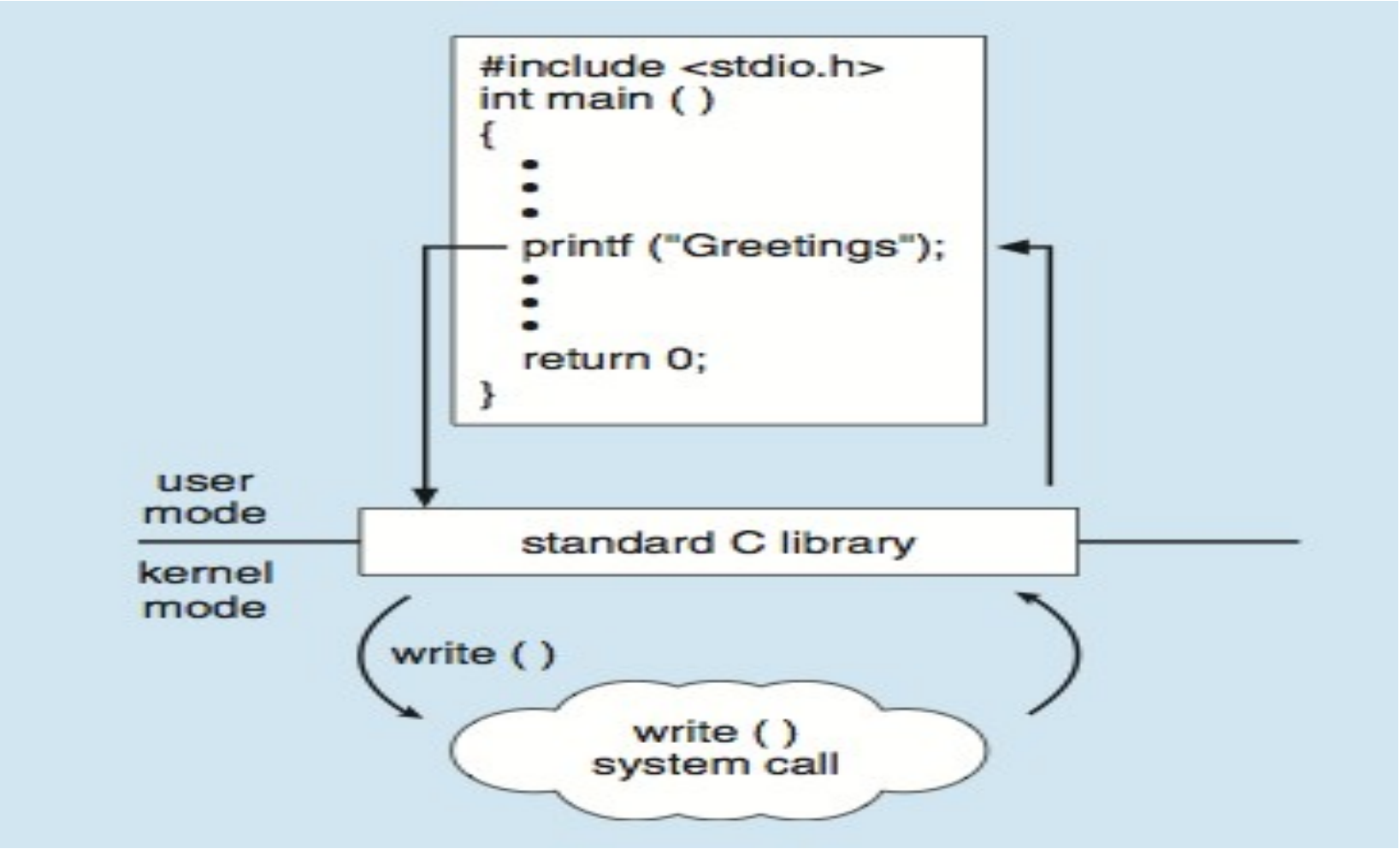
- **Protection**
  - Control access to resources
  - Get and set permissions
  - Allow and deny user access

Examples of Windows and Unix System

	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Communication	CreatePipe()	pipe()
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitlializeSecurityDescriptor()	umask()
	SetSecurityDescriptorGroup()	chown()

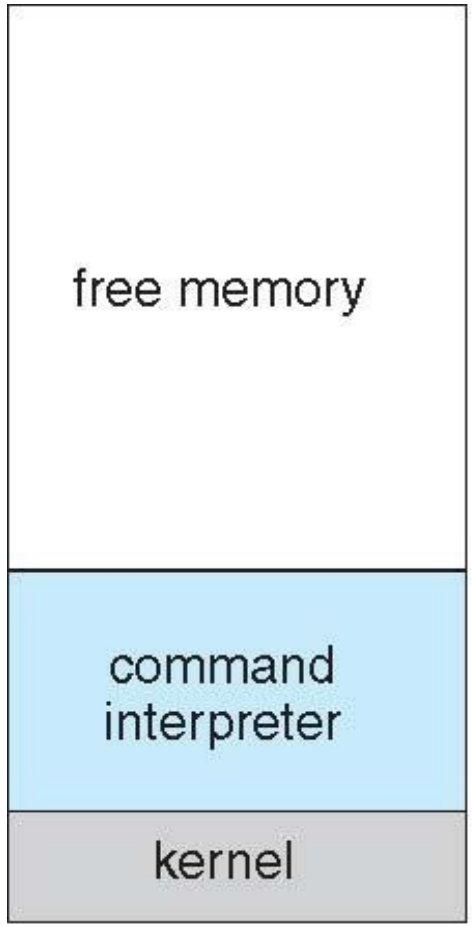
Standard C Library Example

- C program invoking printf() library call, which calls write() system call



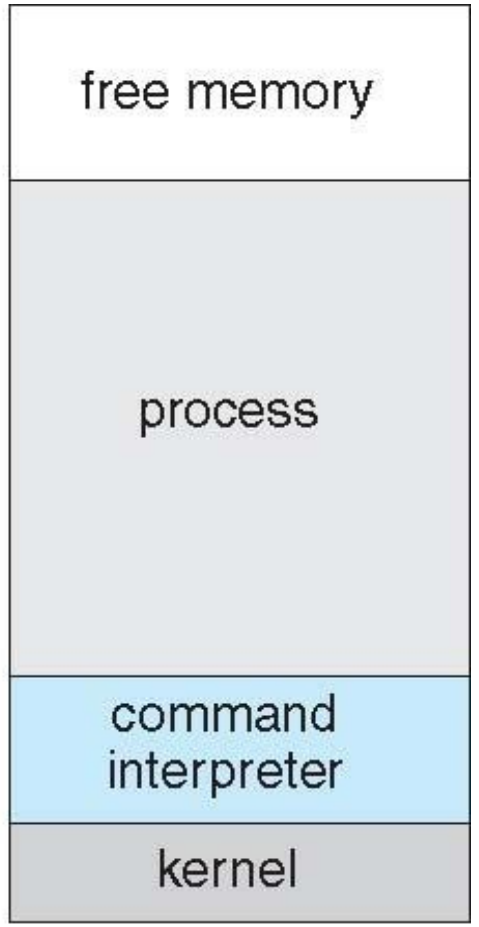
Example: MS-DOS

- Single-tasking
- Shell invoked when system booted
- Simple method to run program
  - No process created
- Single memory space
- Loads program into memory, overwriting all but the kernel
- Program exit -> shell reloaded



(a)

At system startup



(b)

running a program

Test Your knowledge

CLICK HERE

THANK YOU