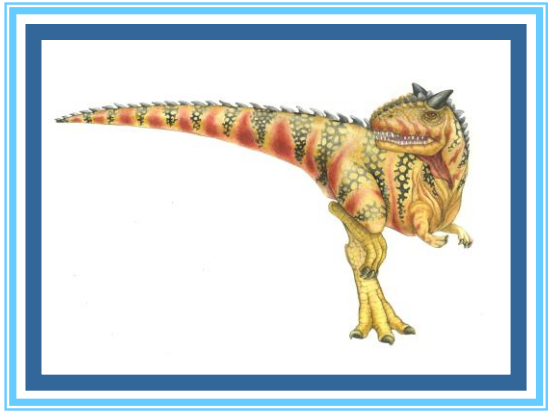
# Chapter 1: Introduction



# Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection
- Virtualization
- Distributed Systems
- Kernel Data Structures
- Computing Environments
- Free/Libre and Open-Source Operating Systems

# Objectives

Describe the general organization of a computer system and the role of interrupts

Describe the components in a modern, multiprocessor computer system

Illustrate the transition from user mode to kernel mode

Discuss how operating systems are used in various computing environments

Provide examples of free and open-source operating systems

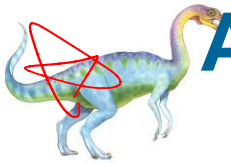
why ? mode ?

each role of operating system in each computing environment

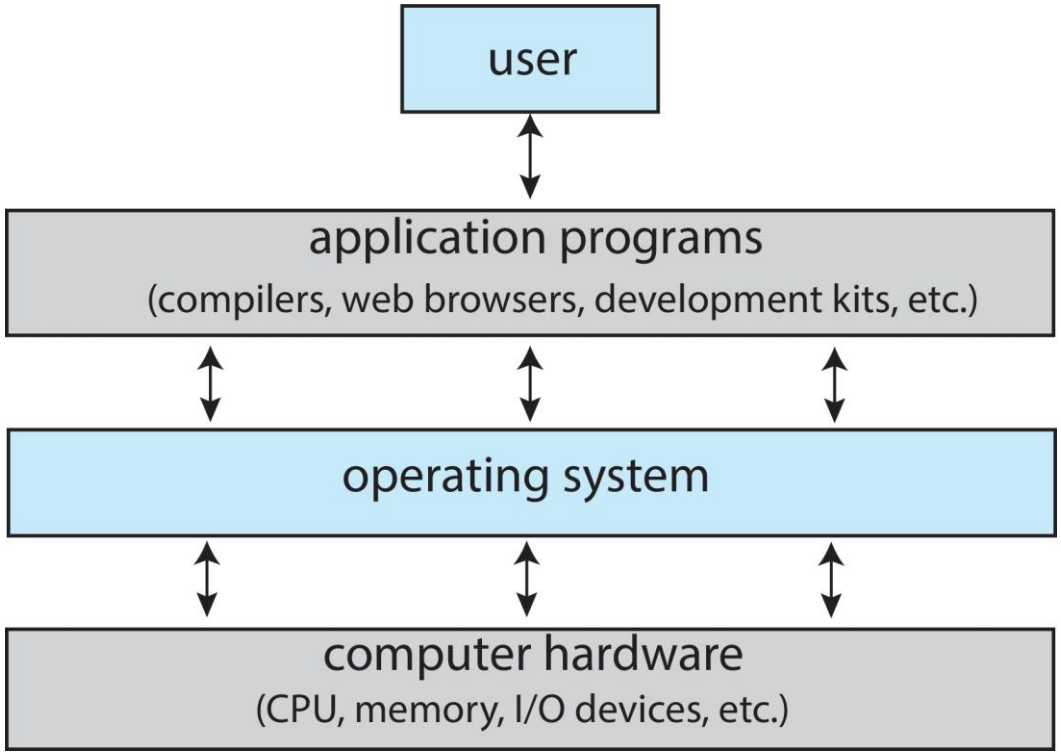
# Computer System Structure

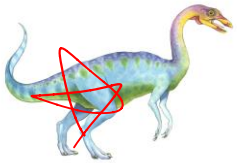
Computer system can be divided into four components:

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users + control HW instead of App
- 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



# Abstract View of Components of Computer



# What Operating Systems Do

role of OS

Depends on the point of view

Users want convenience, **ease of use** and **good performance**

good interface, "GUI"

view of user

Don't care about **resource utilization**

But shared computer such as **mainframe** or **minicomputer** must keep all users happy

view of App

Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs

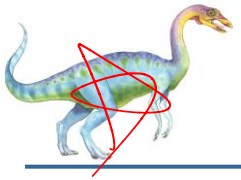
Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**

Mobile devices like smartphones and tablets 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

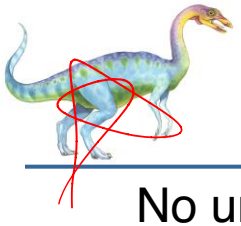
Run primarily without user intervention



# Defining Operating Systems

Term OS covers many roles

- Because of myriad designs and uses of OSes
- Present in toasters through ships, spacecraft, game machines, TVs and industrial control systems
- Born when fixed use computers for military became more general purpose and needed resource management and program control



# Operating System Definition (Cont.)

No universally accepted definition

"Everything a vendor ships when you order an operating system" is a good approximation

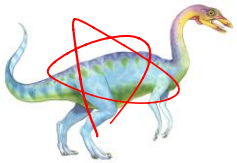
- But varies wildly

"The one program running at all times on the computer" is the **kernel,** part of the operating system

Everything else is either

- a **system program** (ships with the operating system, but not part of the kernel) , or
- an **application program**, all programs not associated with the operating system

Today's OSes for general purpose and mobile computing also include **middleware** – a set of software frameworks that provide addition services to application developers such as databases, multimedia, graphics



# Computer System Organization

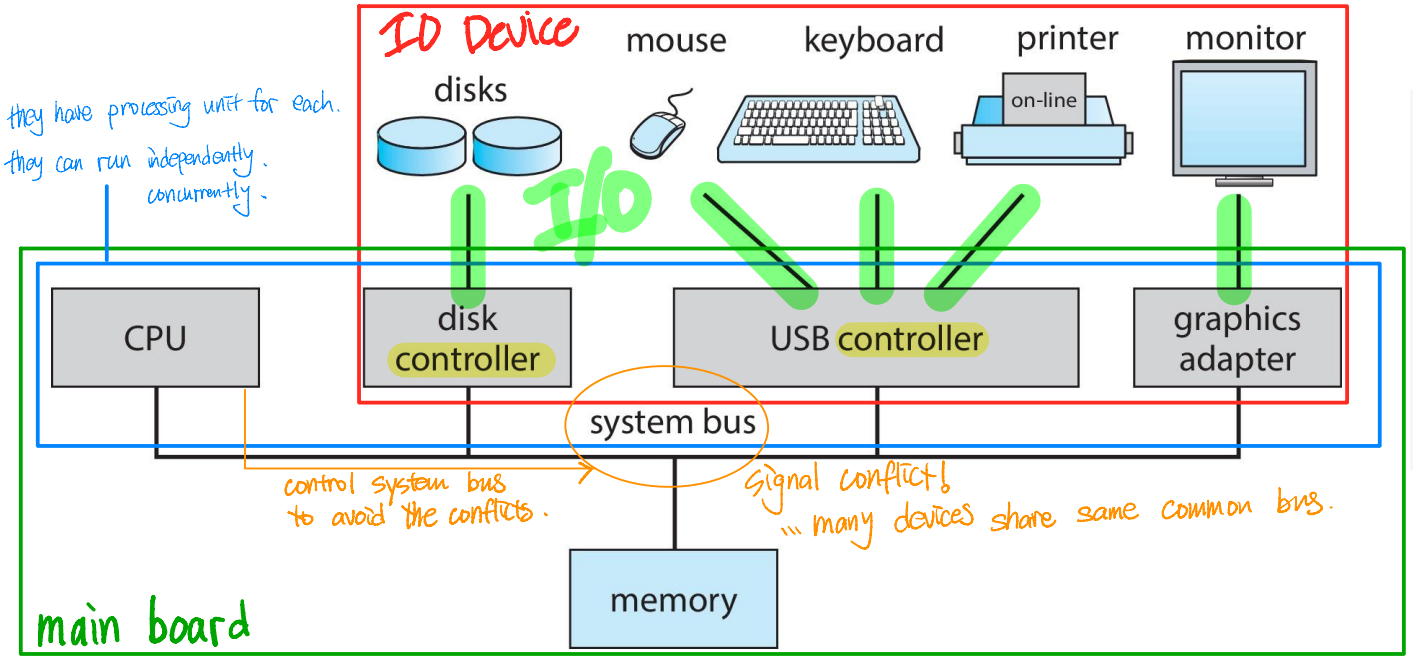
store { orders from CPU / status of devices ...

## Computer-system operation

communicate with devices. local buffers

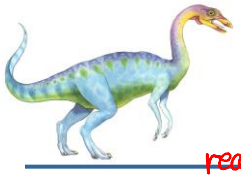
One or more CPUs, device controllers connect through common bus providing access to shared memory

Concurrent execution of CPUs and devices competing for memory cycles




```java
import java.util.Scanner;

public class Main{
    public static void main(String args[]){

    Scanner scan= new Scanner(System.in);

    //For string

    String text= scan.nextLine();

    System.out.println(text);
```

Device ↔ controller ↔ memory



# Computer-System Operation

receive request from CPU & do its task

execute code

I/O devices and the CPU can execute concurrently ∵ own processing units

Each device controller is in charge of a particular device type

Each device controller has a local buffer Store { order from CPU / result of execution

Each device controller type has an operating system **device driver** to manage it (send order to device controller (HW) / receive result of execution)

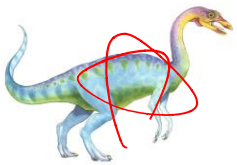
CPU moves data from/to main memory to/from local buffers

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**

make CPU stop current task and execute special type of code

└ interrupt handler



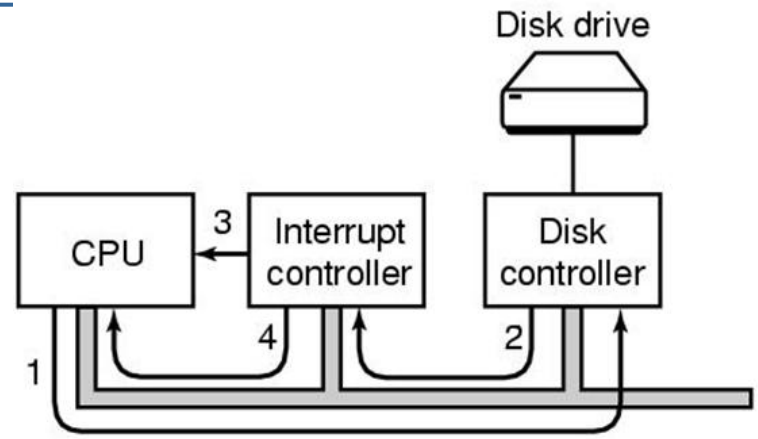
# Common Functions of Interrupts

Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines

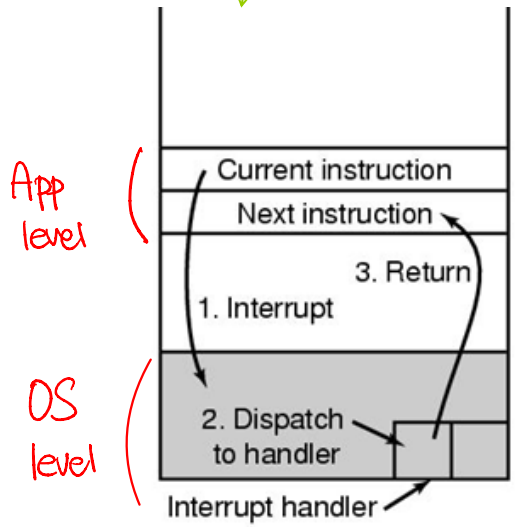
Interrupt architecture must save the address of the interrupted instruction

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

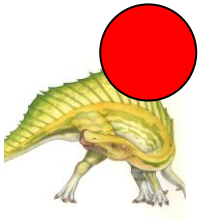
An operating system is **interrupt driven**

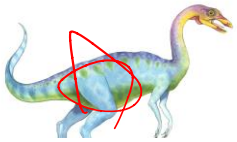



Interrupt handling example: disk drive access

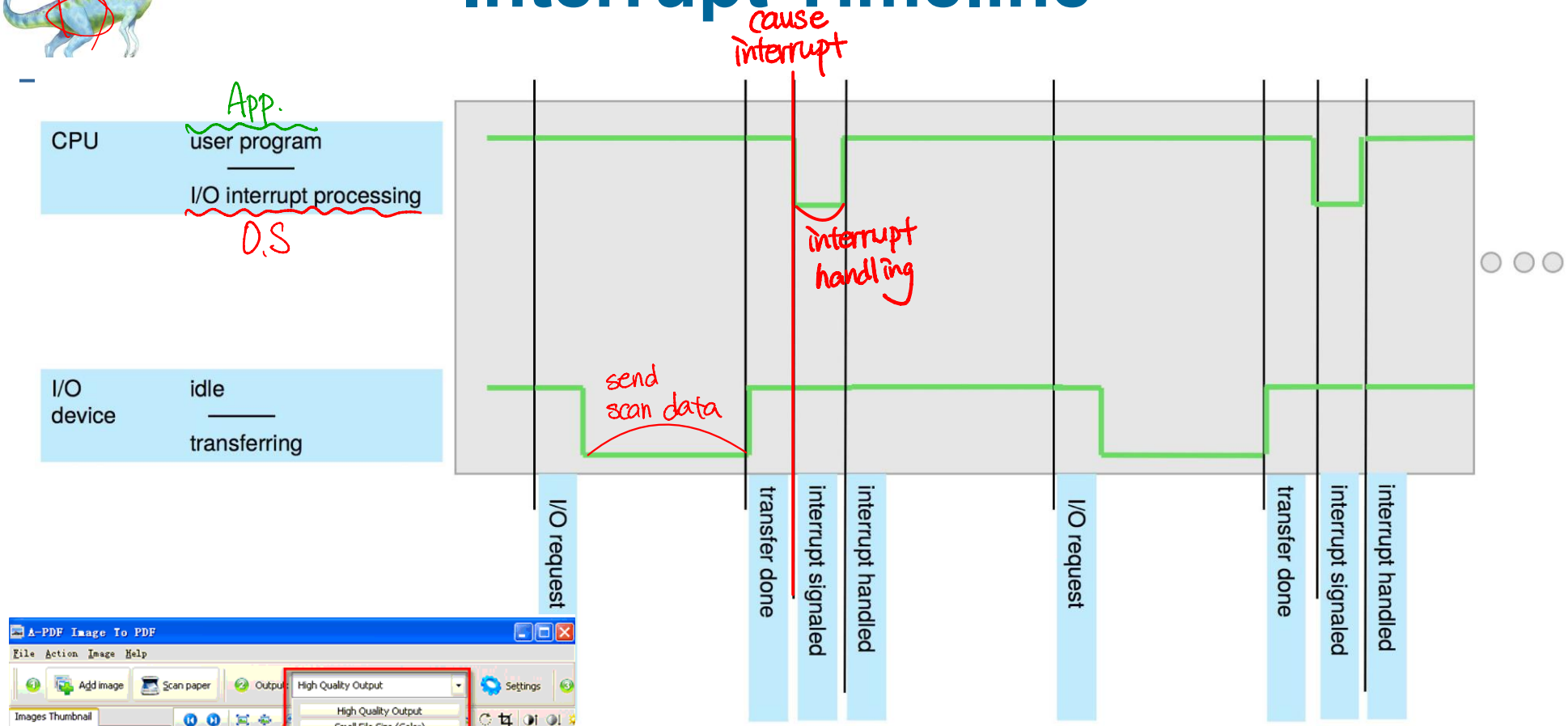


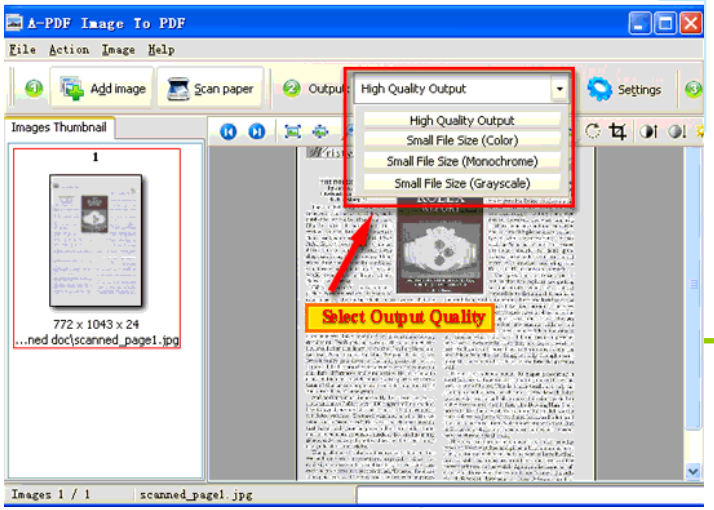

```
mov ecx, ebx
mov esp, edx
mov edx, r9d
mov rax, rdx
```

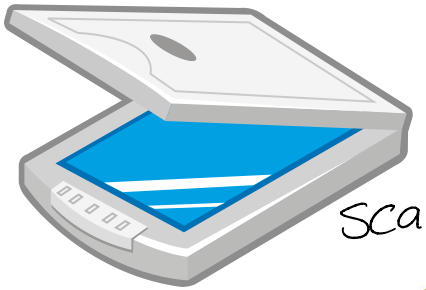




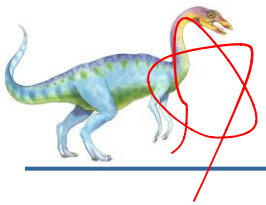
# Interrupt Timeline



scanner



# Interrupt Handling

The operating system preserves the state of the CPU by storing registers and the program counter

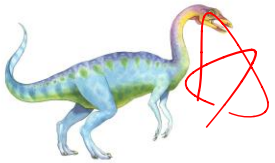
Determines which type of interrupt has occurred: efficient complex

- polling
- vectored interrupt system → OS take control of CPU back
  array of address indicate each interrupt handler

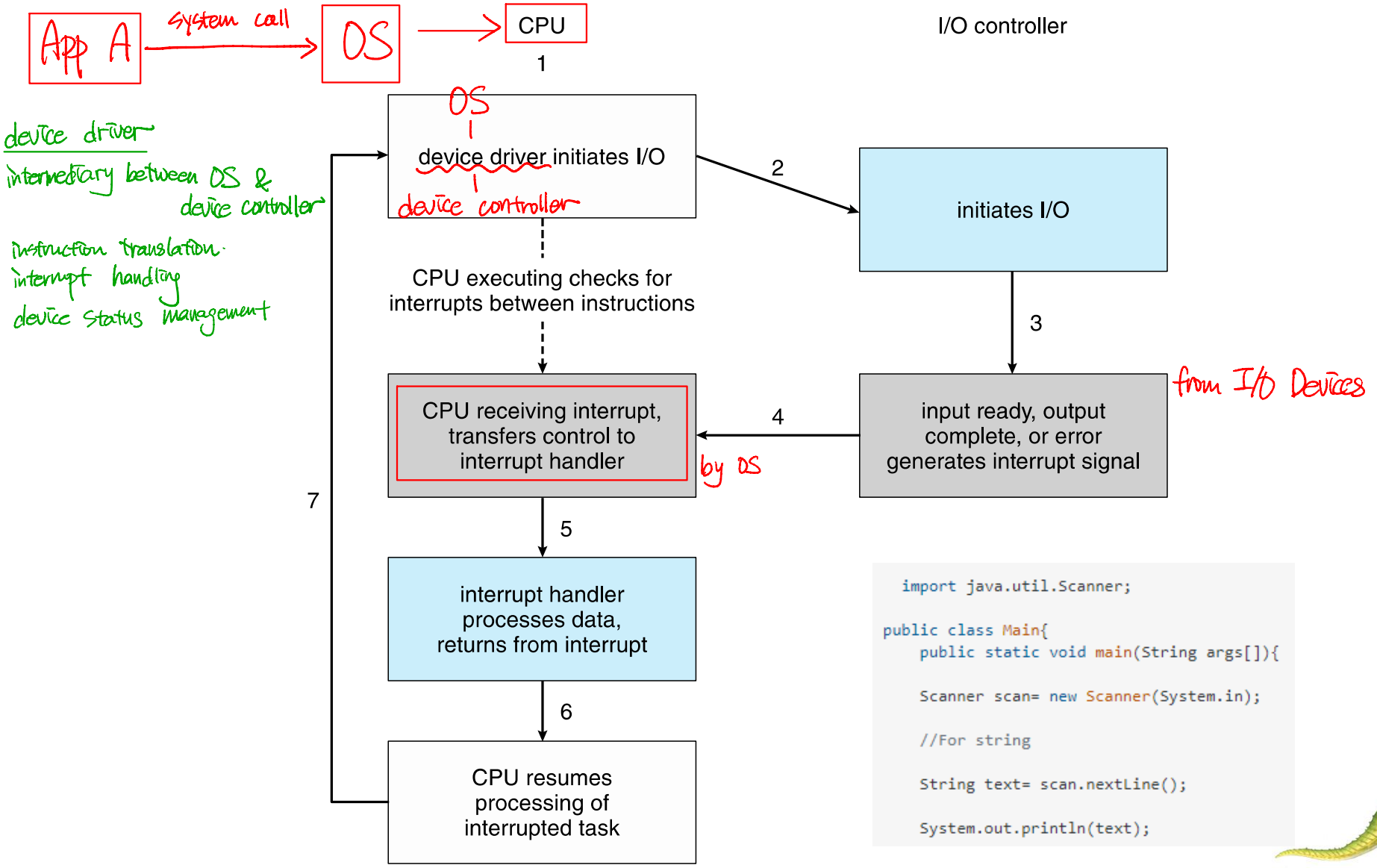
Separate segments of code determine what action should be taken for each type of interrupt

- CPU periodically check status of device or App.
  when device or App. is ready, then do interrupt handling routine.
  waiting for ready = CPU idle
  no work
  just check.

inefficient simple



# Interrupt-driven I/O Cycle




```java
import java.util.Scanner;

public class Main{
    public static void main(String args[]){

    Scanner scan= new Scanner(System.in);

    //For string

    String text= scan.nextLine();

    System.out.println(text);
```

# Storage Structure

Main memory – only large storage media that the CPU can access directly

- **Random access** → efficient. doesn't matter where the data is located.
- Typically **volatile**
- Typically **random-access memory** in the form of **Dynamic Random-access Memory (DRAM)** expensive

---

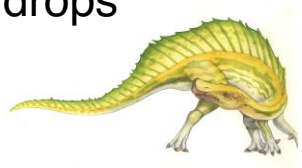
Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity

**Hard Disk Drives (HDD)** – rigid metal or glass platters covered with magnetic recording material Cheap

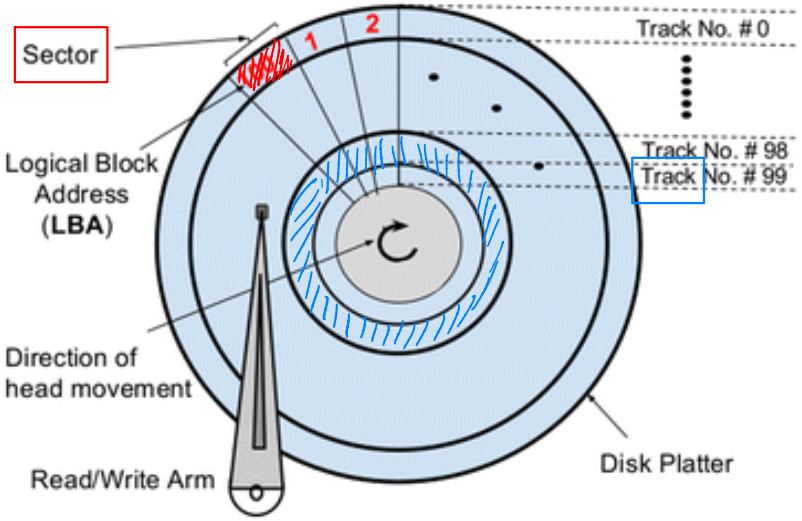
- Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
- The **disk controller** determines the logical interaction between the device and the computer

**Non-volatile memory (NVM)** devices– faster than hard disks, nonvolatile

- Various technologies like SSD, Solid State Drive.
- Becoming more popular as capacity and performance increases, price drops



# HDD Structure.




primary Storage

RAM Register Cache

Secondary Storage

HDD SSD USB ...

# 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. 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.

32bit word = 4 byte

64bit word = 8 byte

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; and 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 Hierarchy

Storage systems organized in hierarchy

- Speed
- Cost
- Volatility

efficient: temporal locality | likely be accessed again soon.
spatial locality | nearby location likely be accessed

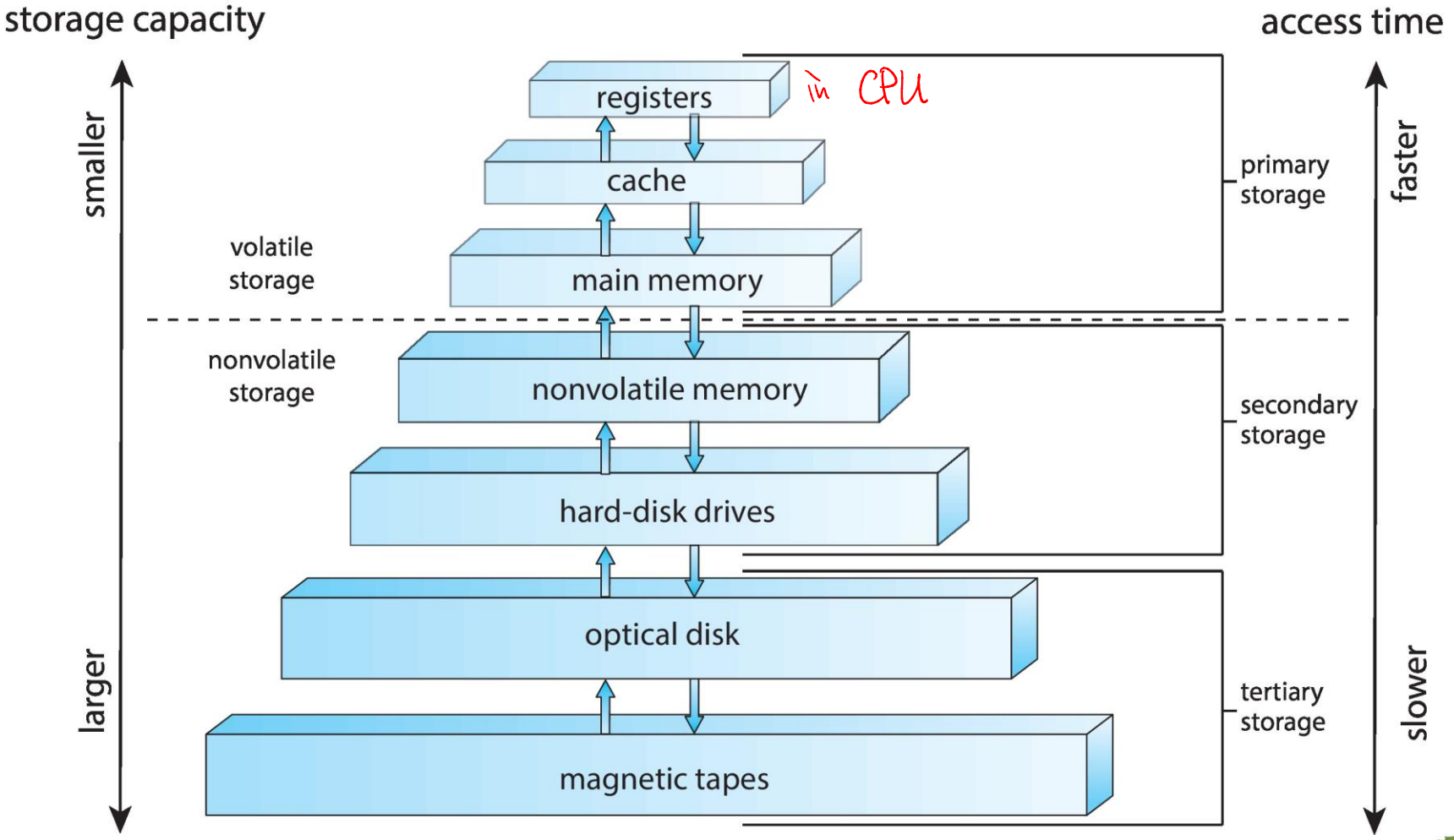
**Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

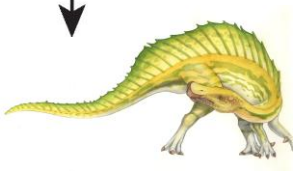
**Device Driver** for each device controller to manage I/O

Provides uniform interface between controller and kernel

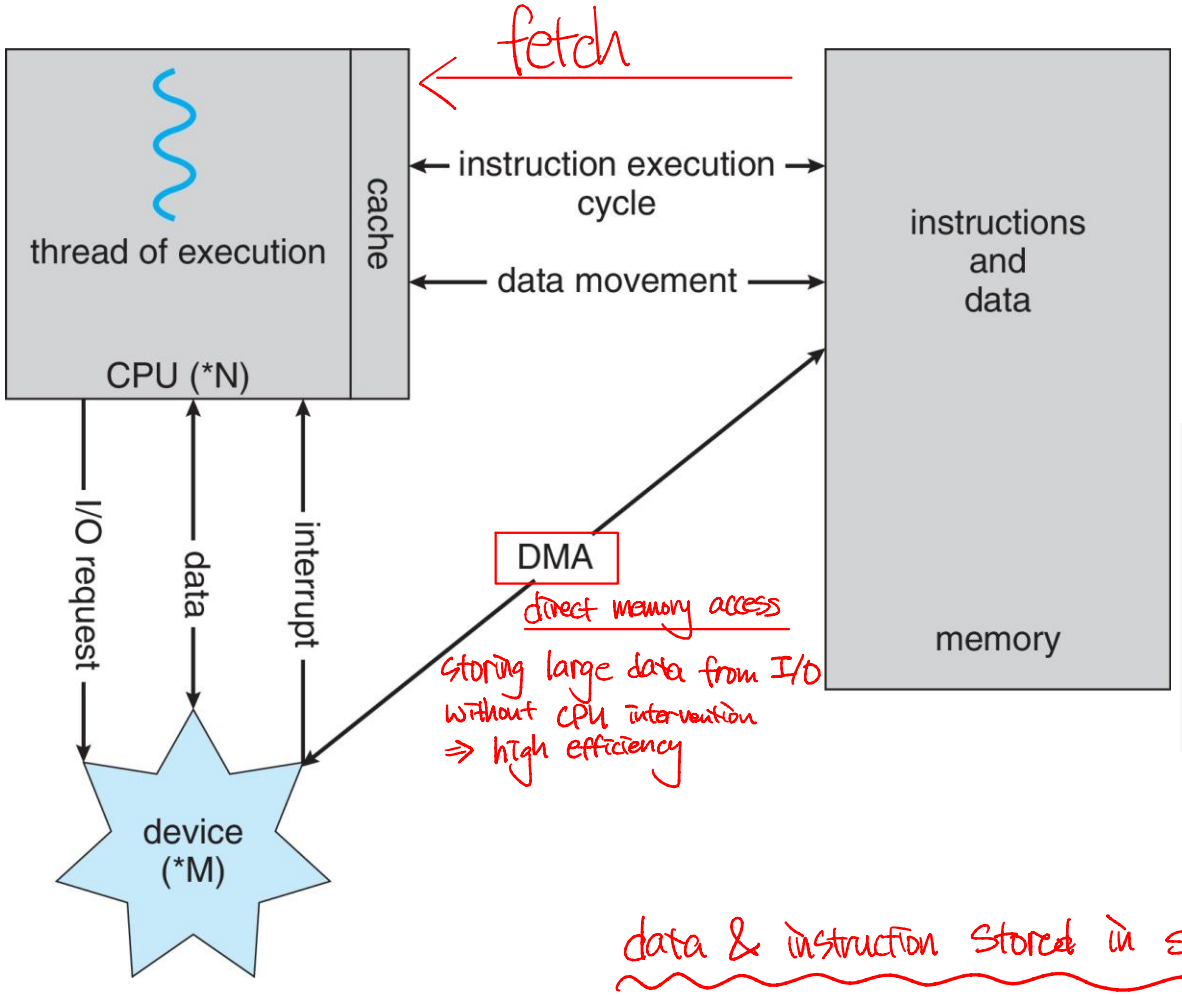
OS can ignore difference of how to work each devices.
OS can access diverse drivers by uniform way.

# Storage-Device Hierarchy



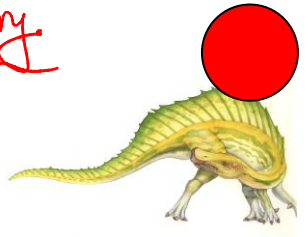
# How a Modern Computer Works




```
import java.util.Scanner;

public class Main{
    public static void main(String args[]){

    Scanner scan= new Scanner(System.in);

    //For string

    String text= scan.nextLine();

    System.out.println(text);
```

data & instruction stored in same memory.

*A von Neumann architecture*



# 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

# Computer-System Architecture

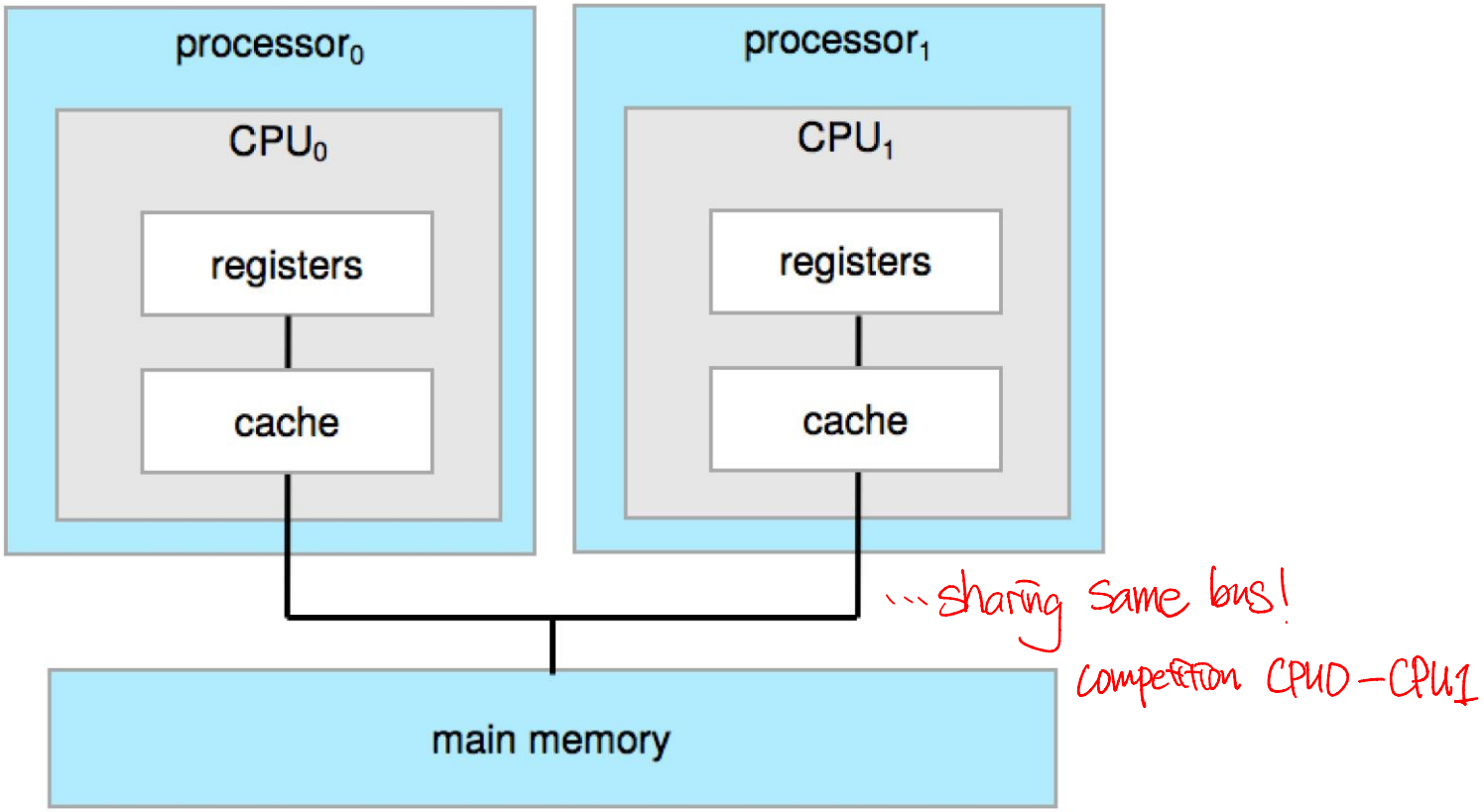
Most systems use a single general-purpose processor (PC smartphone ...)

- Most systems have special-purpose processors as well (device controller ...)

**Multiprocessors** systems growing in use and importance

- Also known as **parallel systems**, **tightly-coupled systems**
- Advantages include:
  1. **Increased throughput**
  2. **Economy of scale**
  3. **Increased reliability** – graceful degradation or fault tolerance
- Two types:
  1. **Asymmetric Multiprocessing** – each processor is assigned a specific task. (master & slave)
  2. **Symmetric Multiprocessing** – each processor performs all tasks (one type of processor)

# Symmetric Multiprocessing Architecture



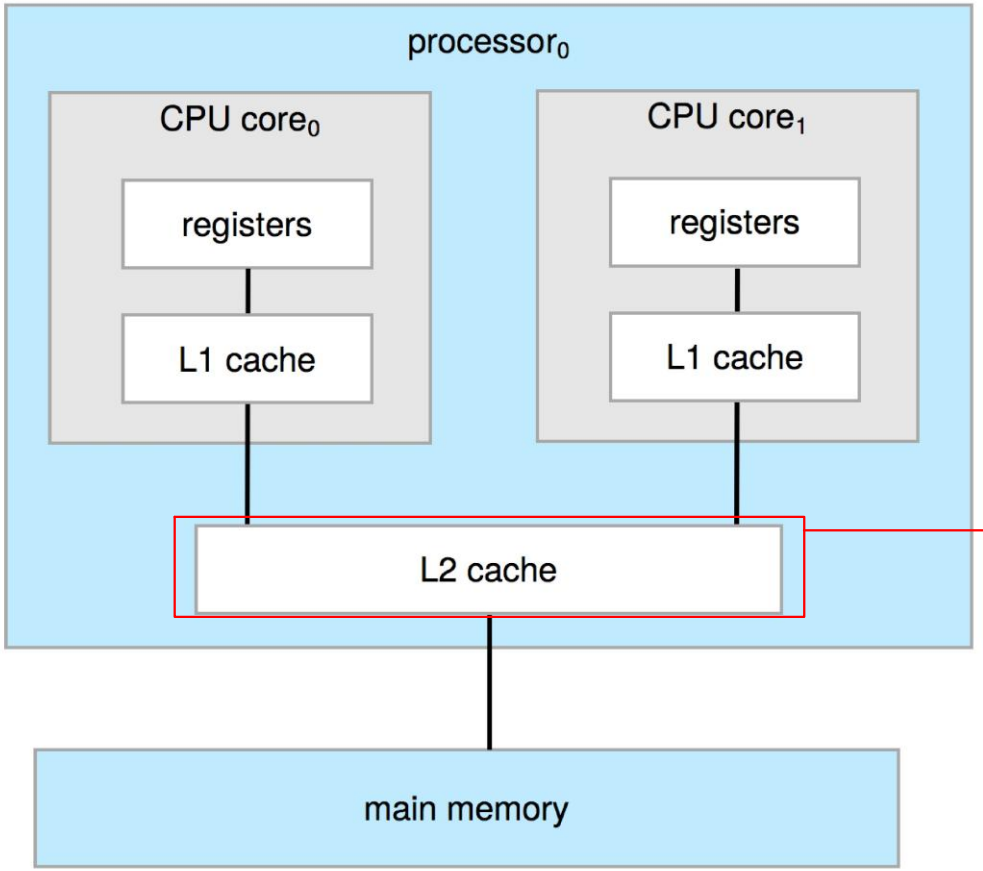

# A Dual-Core Design

Multi-chip and **multicore** ⇒ one processor, multiple core.

Systems containing all chips

Chassis containing multiple separate systems

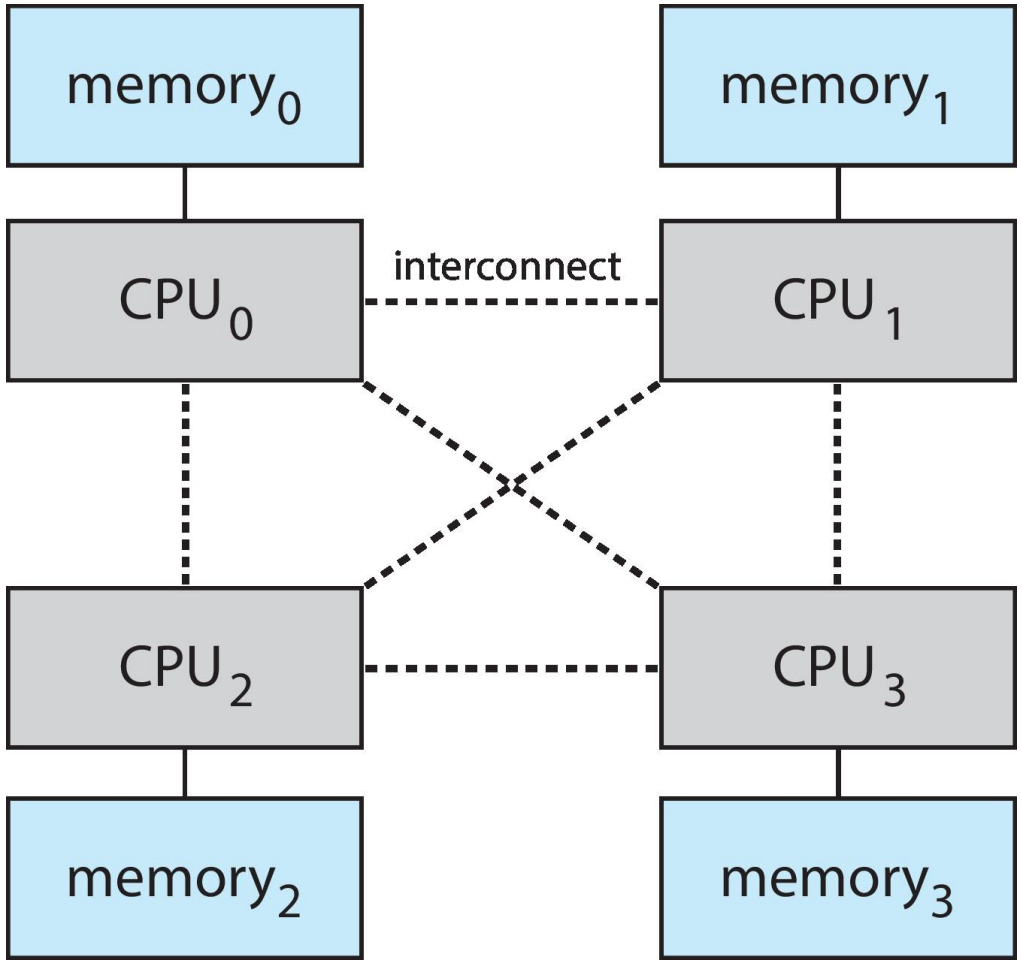
single processor act like multiprocessing system.




sharing cache!
fast communicate with other core.

# Non-Uniform Memory Access System

a.k.a. NUMA System




# Clustered Systems

Like multiprocessor systems, but multiple systems working together

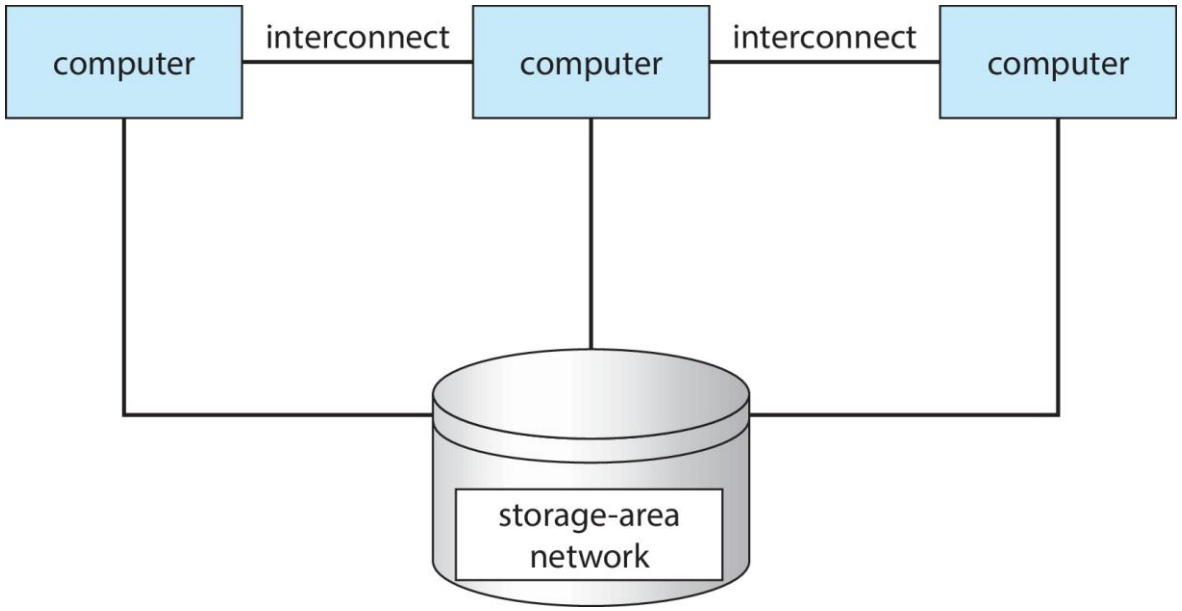
- Usually sharing storage via a **storage-area network (SAN)**
- Provides a **high-availability** service which survives failures
  - **Asymmetric clustering** has one machine in hot-standby mode
  - **Symmetric clustering** has multiple nodes running applications, monitoring each other
- Some clusters are for **high-performance computing (HPC)**
  - Applications must be written to use **parallelization**
- Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

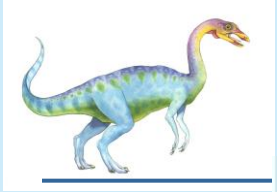
master-slave

equal nodes

↳ e.g., access same file at the same time

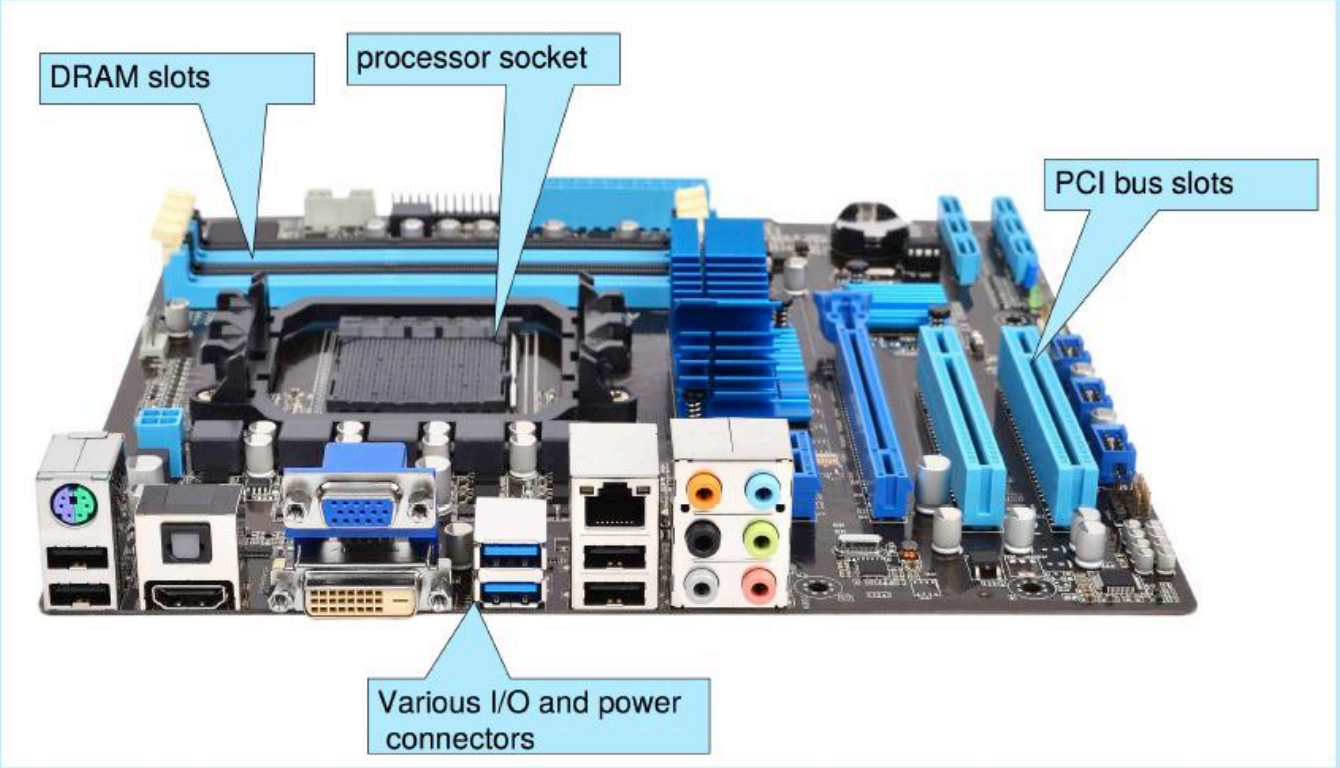
# Clustered Systems



# PC Motherboard

Consider the desktop PC motherboard with a processor socket shown below:




This board is a fully-functioning computer, once its slots are populated. It consists of a processor socket containing a CPU, DRAM sockets, PCIe bus slots, and I/O connectors of various types. Even the lowest-cost general-purpose CPU contains multiple cores. Some motherboards contain multiple processor sockets. More advanced computers allow more than one system board, creating NUMA systems.

# Operating-System Operations

Bootstrap program – simple code to initialize the system, load the kernel

1. Kernel loads
2. Starts **system daemons** (services provided outside of the kernel)
3. Kernel **interrupt driven** (hardware and software)

   Hardware interrupt by one of the devices

   Software interrupt (**exception** or **trap**):

   - Software error (e.g., division by zero)
   - Request for operating system service – **system call**
   - Other process problems include infinite loop, processes modifying each other or the operating system

```
import java.util.Scanner;

public class Main{
    public static void main(String args[]){

    Scanner scan= new Scanner(System.in);

    //For string

    String text= scan.nextLine();

    System.out.println(text);
```

# Multiprogramming and Multitasking

multiple programs work in concurrent manners.

```
import java.util.Scanner;

public class Main{
    public static void main(String args[]){

    Scanner scan= new Scanner(System.in);

    //For string

    String text= scan.nextLine();

    System.out.println(text);
```

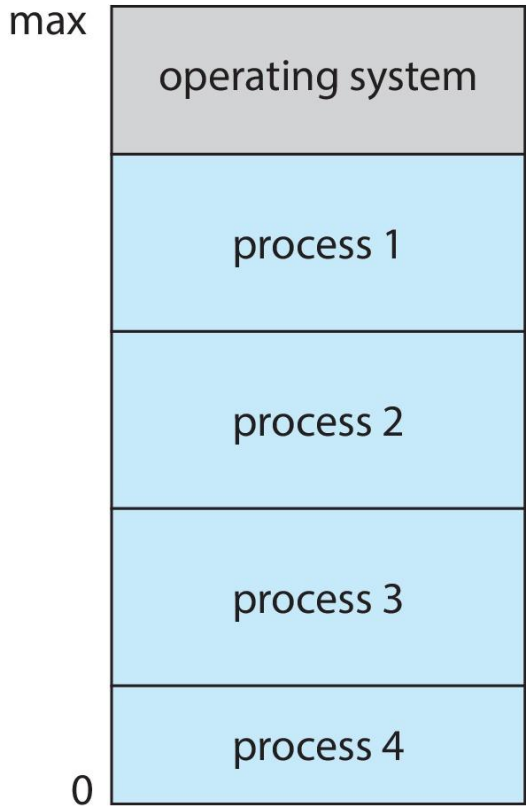
**Multiprogramming** (**Batch system**) needed for efficiency

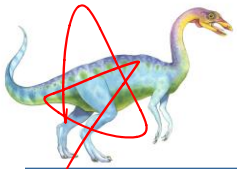
- Single user cannot keep CPU and I/O devices busy at all times
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- When it has to wait (for I/O for example), OS switches 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



# Dual-mode and Multimode Operation

**Dual-mode** operation allows OS to protect itself and other system components

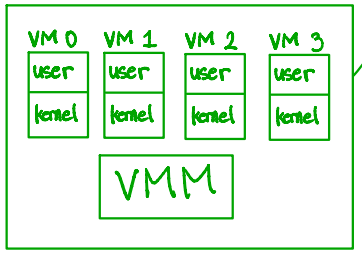
**User mode** and **kernel mode**

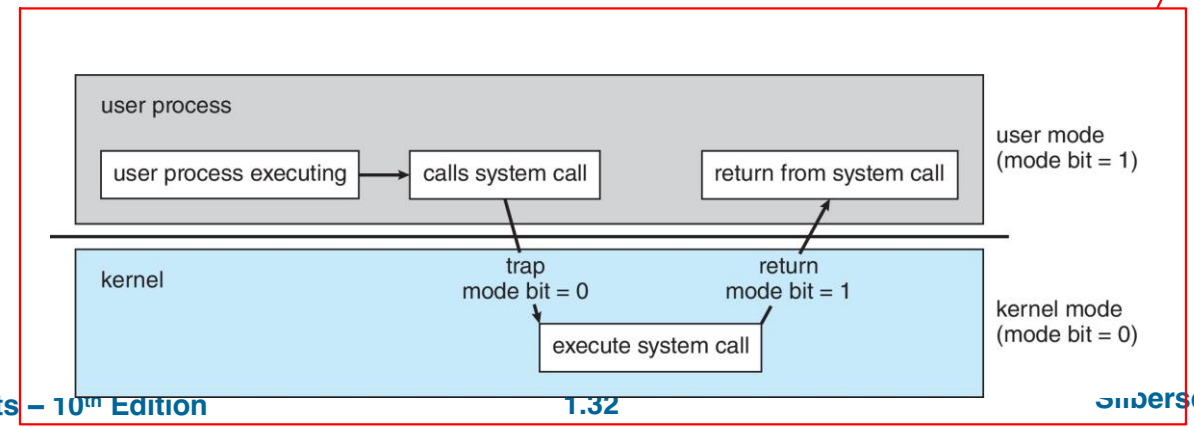
**Mode bit** provided by hardware

- Provides ability to distinguish when system is running user code or kernel code
- Some instructions designated as **privileged**, only executable in kernel mode
- System call changes mode to kernel, return from call resets it to user

Increasingly CPUs support multi-mode operations

i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**



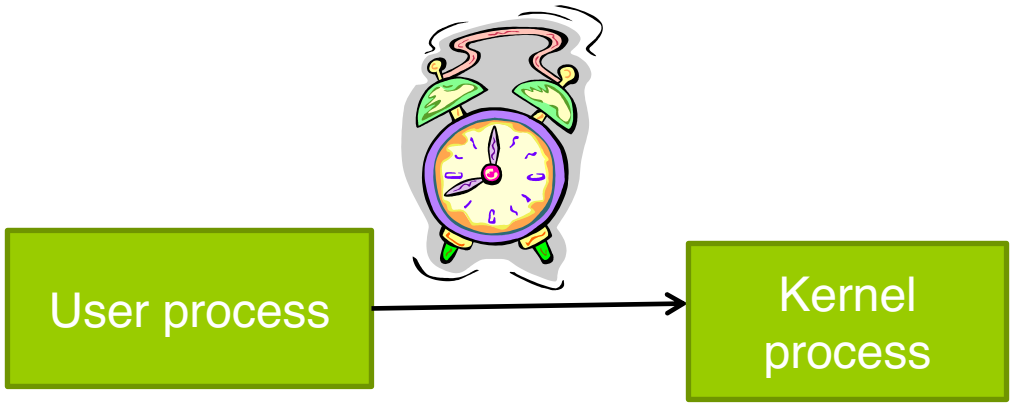




# Transition from User to Kernel Mode

Timer Interrupt

Timer to prevent infinite loop / process hogging resources + resource sharing.

- 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




System privilege

# Process Management

load in memory, provide CPU resource.

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

execute sequential manner ...

single process!

Single-threaded process has one **program counter** specifying location of next instruction to execute

maintained by process

- Process executes instructions sequentially, one at a time, until completion

Multi-threaded process has one program counter per thread

multiprogramming

Typically system has many processes, some user, some operating system running concurrently on one or more CPUs

multi - process ⇕ mapping multi CPUs

maximize usage of process

- 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 → keep data consistency / avoid crash of processes.
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication with data
- Providing mechanisms for deadlock handling
  shortage of resources.
  ⋯ stop working processes.

# Memory Management

To execute a program all (or part) of the instructions must be in memory

All (or part) of the data that is needed by the program must be in memory

Memory management determines what is in memory and when

↳ Optimizing CPU utilization and computer response to users

Memory management activities

e.g.,

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes (or parts thereof) and data to move into and out of memory
- Allocating and deallocating memory space as needed

# File-system Managemen

regardless of media

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 dr
  - Varying properties include access speed, capacity, data-access method (sequential or random)

user consider nothing !

OS do.

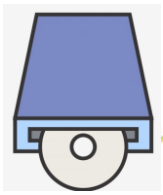
File-System management

- Files usually organized into directories
- Access control on most systems to determine who can access
- 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

in multi-user system
and what.

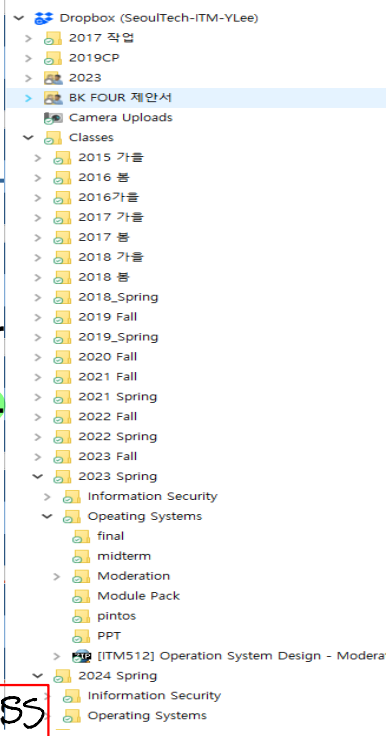
... appending, (re)naming, setting attributes, ...





logical file structure
↕ mapping
physical storage structure.

... OS keep metadata of file.
information of mapping.



secondary storage

# 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 ... non-volatile
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - Mounting and unmounting – (un) install of storage
  - Free-space management
  - Storage allocation
  - Disk scheduling – handle multiple request of reading/writing files
  - Partitioning – make multi logical disk in single physical disk
  - Protection
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications

# 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

# Characteristics of Various Types of Storage

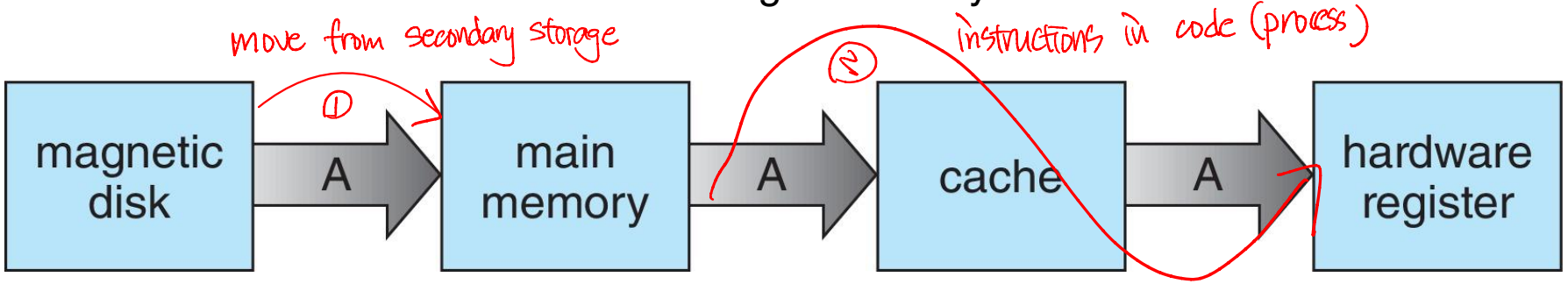
| Level | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Name | registers | cache | main memory | solid-state disk | magnetic disk |
| Typical size | < 1 KB | < 16MB | < 64GB | < 1 TB | < 10 TB |
| Implementation technology | custom memory with multiple ports CMOS | on-chip or off-chip CMOS SRAM | CMOS 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

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

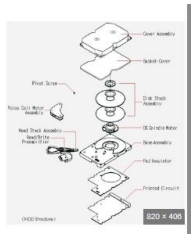
- Several copies of a datum can exist
- Various solutions covered in Chapter 19

# I/O Subsystem

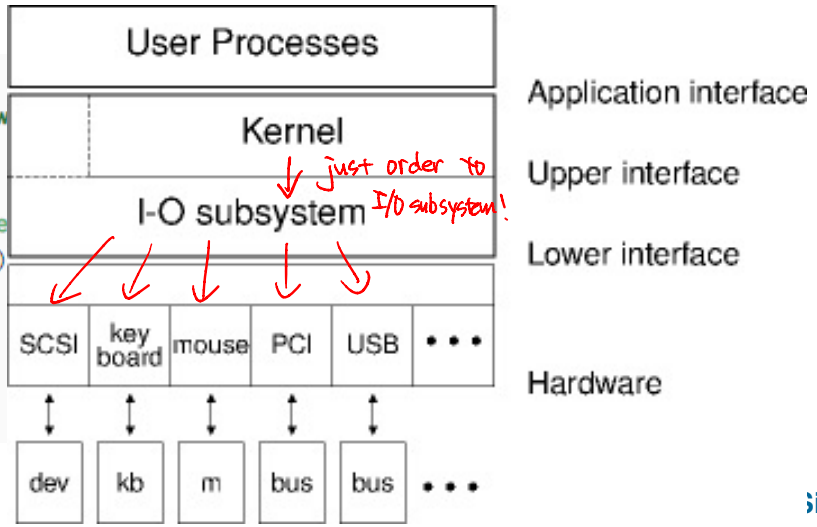
One purpose of OS is to hide peculiarities of hardware devices from the user → access different type of storage in uniform way.

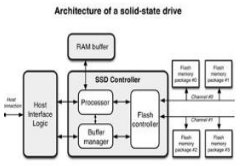
I/O subsystem responsible for

- 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)
  - buffering: store data in buffer and send the data to actual device at regular intervals.
  - spooling: before one job is finished, the output of the job is stored in queue, an input from another job can be processed in meantime
- General device-driver interface
- Drivers for specific hardware devices



```
@Test
public void givenWritingStringToFile_w
  throws IOException {
    String str = "Hello";
    FileOutputStream outputStream = ne
    byte[] strToBytes = str.getBytes()
    outputStream.write(strToBytes);

    outputStream.close();
}
```




# 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

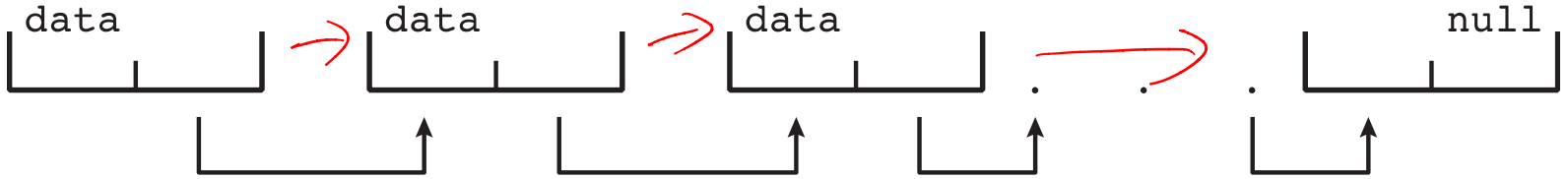
Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file

**Privilege escalation** allows user to change to effective ID with more rights

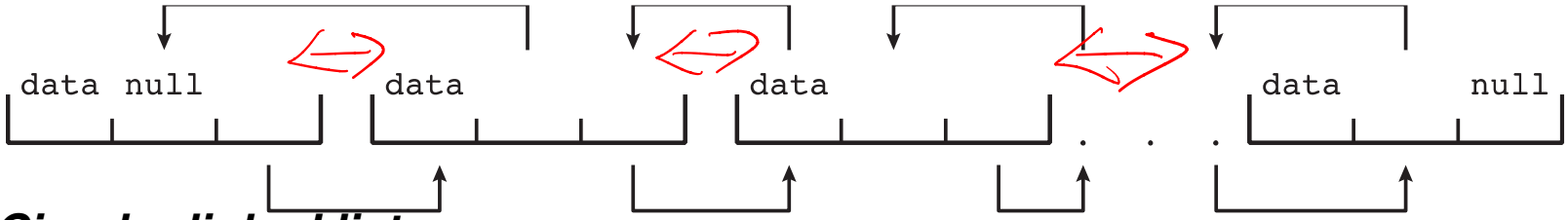
user change their password,
changing pw is right of root.
so user request privilege escalation

# Kernel Data Structures

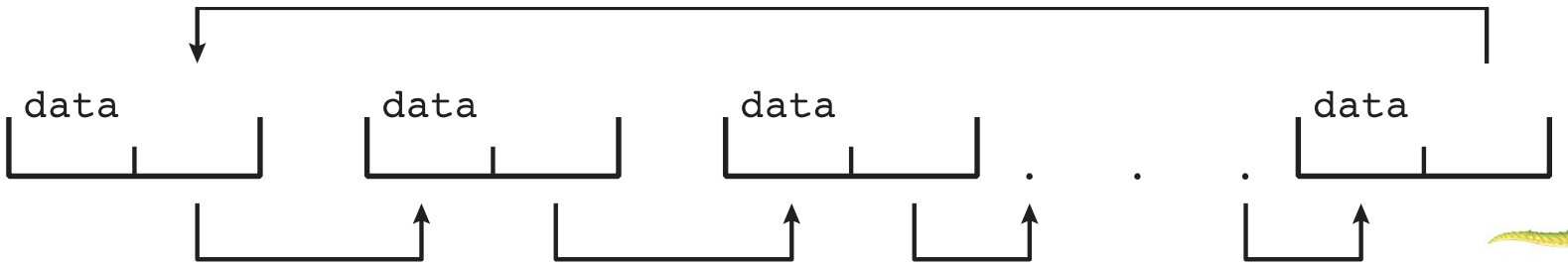
- Many similar to standard programming data structures
- ***Singly linked list***

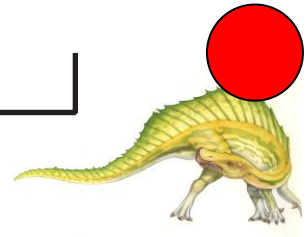



- ***Doubly linked list***




- ***Circular linked list***



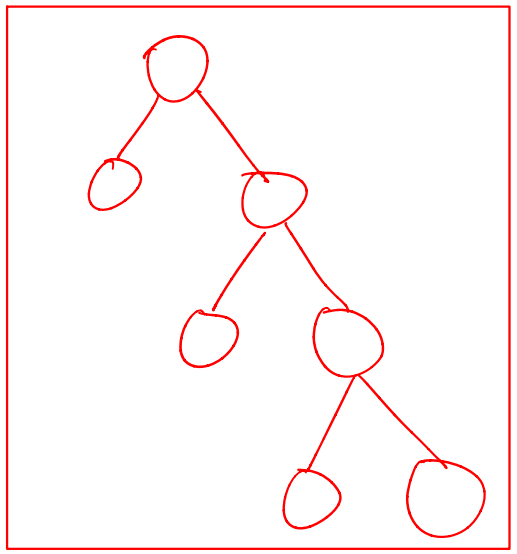
# Kernel Data Structures

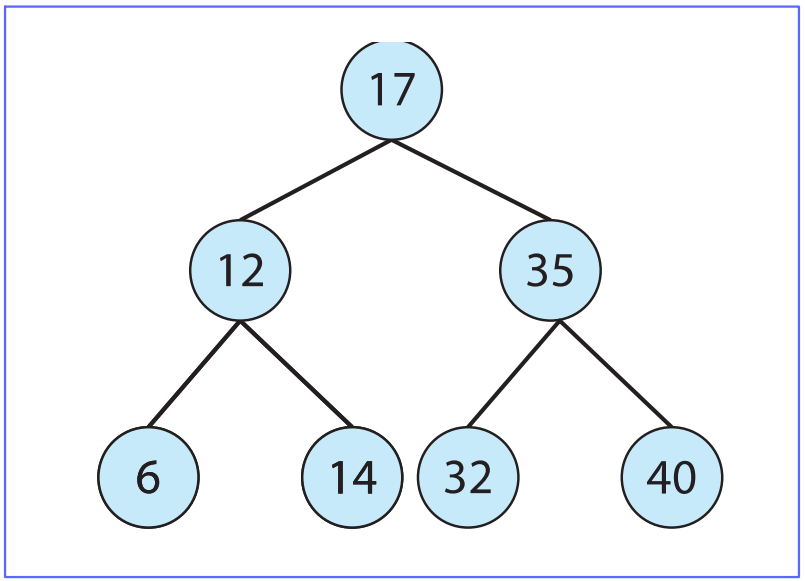
**Binary search tree**

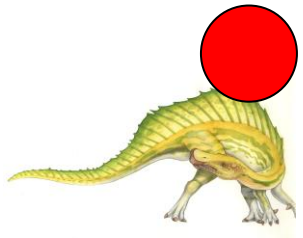
left <= right

Search performance is *O(n)*

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



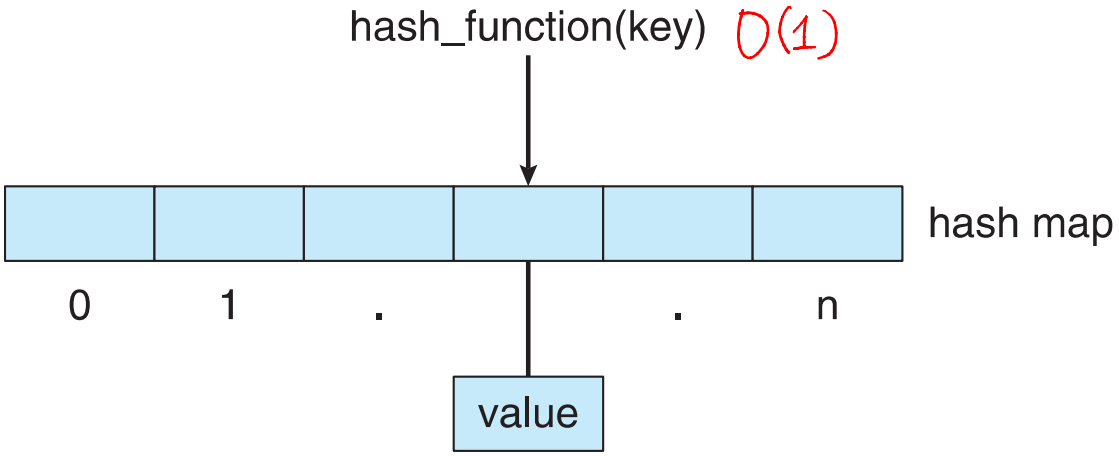


# Kernel Data Structures

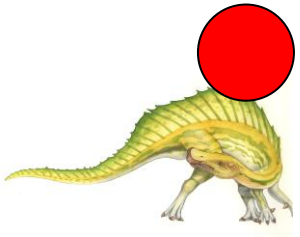
efficient storage usage!

**Hash function** can create a **hash map** (key, value)




**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

Stand-alone general purpose machines like PC

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

role of I/O device.
everything is done in server.

# Computing Environments - 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 ss, or cellular data networks for connectivity

Leaders are **Apple iOS** and **Google Android**

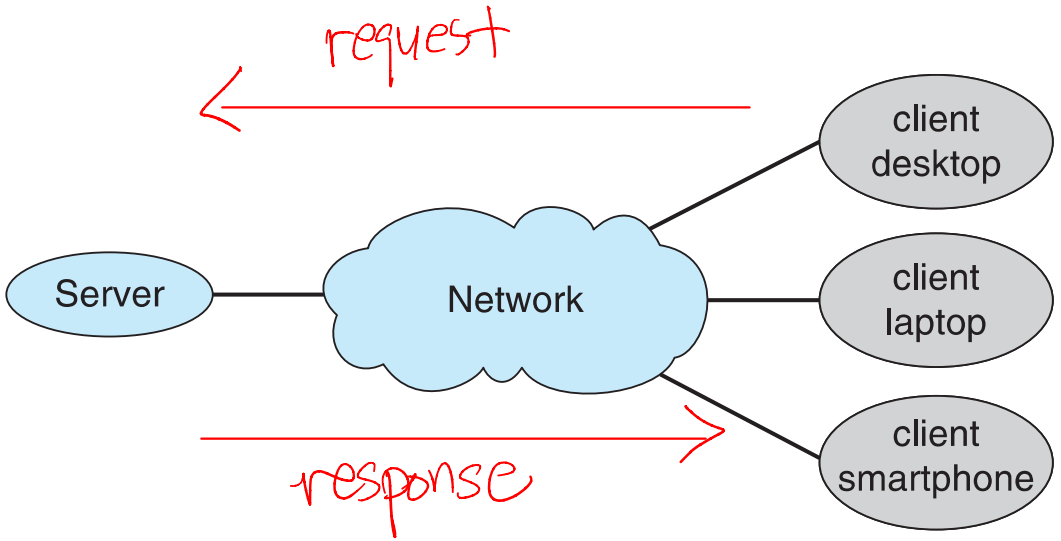
# Computing Environments – Client-Server

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

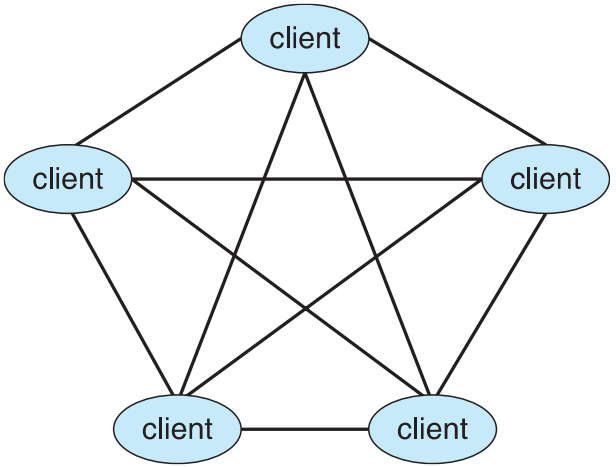



# Computing Environments - Peer-to-Peer

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




# Computing Environments – Cloud Computing

Delivers computing, storage, even apps as a service across a network virtually

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

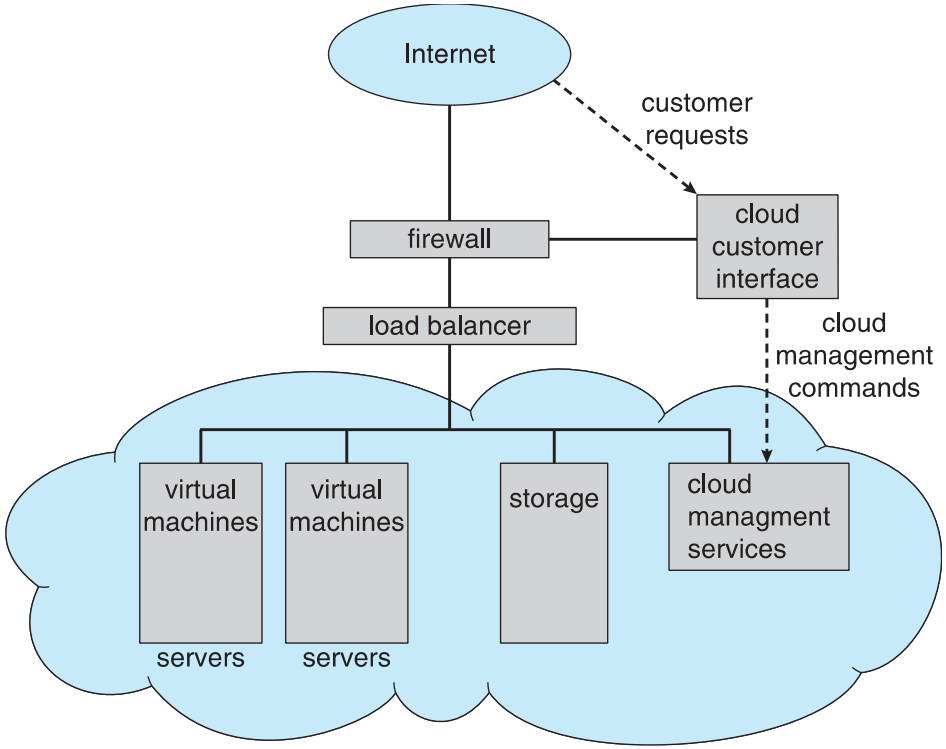
Many types

- **Public cloud** – available via Internet to anyone willing to pay
- **Private cloud** – run by a company for the company's own use
- **Hybrid cloud** – includes both public and private cloud components (popular)
- Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor) ⇒ Google Workspace SW App
- Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server) ⇒ Azure App Service App development platform
- Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use) ⇒ AWS Azure Google Cloud Platform virtual server, storage, network

# Computing Environments – Cloud Computing

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




# Computing Environments – 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

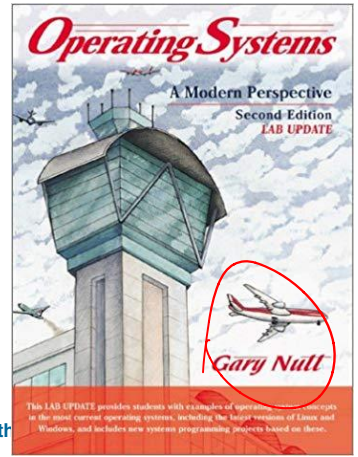
Many other special computing environments as well

- 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

decision should be made in very short time.





# Summary

An Operating System is software that manages the computer hardware, as well as proving an environment for applications programs to run

Interrupts are a key way in which hardware interacts with the operating system. A hardware device triggers an interrupt by sending a signal to the CPU to alert the CPU that some event requires attention. The interrupt is managed by the interrupt handler

For a computer to do its job of executing programs, the programs must be in main memory, which is the only large storage area that the processor can access directly.

The wide variety of storage systems in a computer system can be organized in a hierarchy according to speed and cost. The higher levels are expensive, but they are fast.

# Summary (Cont'd)

Modern computer architectures are multiprocessor systems in which each CPU contains several computing cores

To best utilize the CPU, modern operating systems employ multiprogramming, which allows several jobs to be in memory at the same time, thus ensuring that the CPU always has a job to execute

Multitasking is an extension of multiprogramming wherein CPU scheduling algorithms rapidly switch between processes, providing users with a fast response time

To prevent user programs from interfering with the proper operation of the system, the system hardware has two modes: user mode and the kernel mode

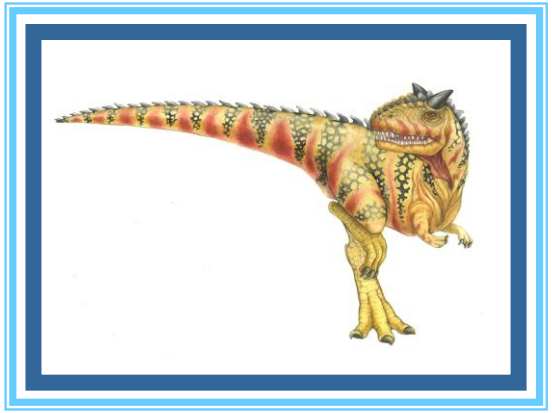
# Summary (Cont'd)

Various instructions are privileged and can executed only in kernel mode. Examples include the instruction to switch to kernel mode, I/O control, timer management, and interrupt management

A Process is the fundamental unit of work in an operating system. Process management includes creating/deleting processes and providing mechanisms for processes to communicate and synchronize with each other.

Computing takes place in a variety of environments, including traditional computing, mobile computing, client-server systems, peer-to-peer systems, cloud computing, and real-time embedded systems

# End of Chapter 1



# C language material

Reference Materials

https://faculty.washington.edu/jstraub/dsa/Master_2_7a.pdf

http://www.tutorialspoint.com/ansi_c/index.htm

http://edu.goorm.io/lecture/201/%EB%B0%94%EB%A1%9C-%EC%8B%A4%ED%96%89%ED%95%B4%EB%B3%B4%EB%A9%B4%EC%84%9C-%EB%B0%B0%EC%9A%B0%EB%8A%94-c%EC%96%B8%EC%96%B4 (in Korean)

https://dojang.io/mod/page/view.php?id=644 (Unit 74) (in Korean)