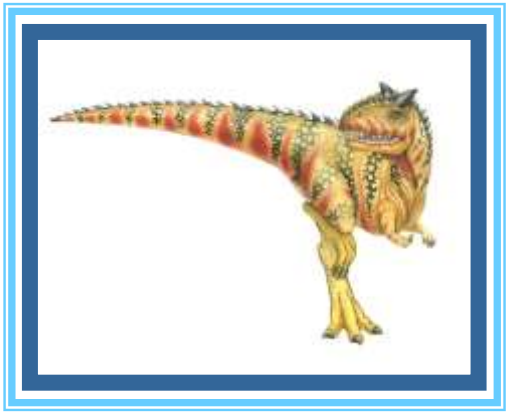
Operating System
CSC202

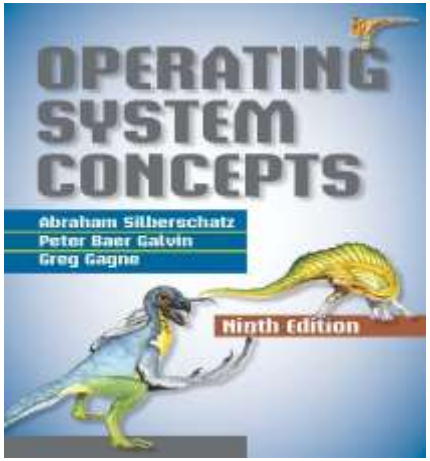


About Course

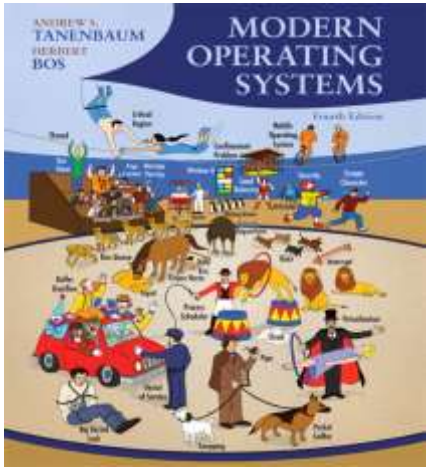
- Will Teach Operating system Concepts and Components
- Will Include Programing Assignments
- Example From real Operating System Linux
- Will Involve programming Using C Language

Text and Reference Books

- Operating System Concepts, 9th editions, Silberschatz et al. Wiley.



- Modern Operating Systems, Andrew S. Tanenbaum, latest edition, .

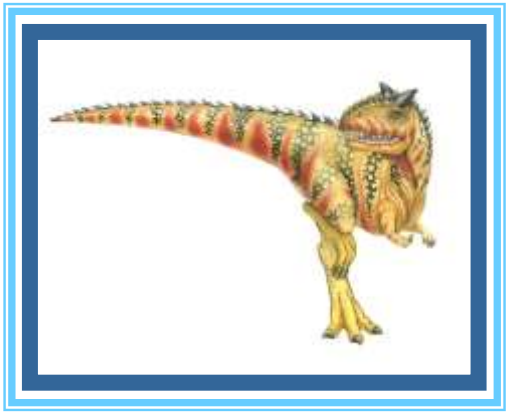


Date	Duration	Topics Covered	Evaluation Instruments used	Signature
Week 1	3 +1	**Introduction to Operating System** What Operating Systems Do , Computer-System Organization ,Computer-System Architecture ,Operating-System Structure ,Operating-System Operations ,Process Management ,Memory Management ,Storage Management ,Protection and Security ,Kernel Data Structures ,Computing Environments , Open-Source Operating Systems		
Week 2	3 +1	**Operating-System Structures.** Operating-System Services, User and Operating-System, Interface , System Calls , Types of System Calls, System Programs, Operating-System Design and Implementation, Operating-System Structure Operating-System Debugging, Operating-System Generation, System Boot		
Week 3	3 +1	**Processes** Process Concept , Process Scheduling, Operations on Processes, Interposes Communication, Communication in Client– Server Systems 136	Quiz 1 Assig 1	
Week 4	3 +1	**Threads.** Overview, Multicore Programming, Multithreading Models, Thread Libraries, Implicit Threading, Threading Issues.		
Week 5	3 +1	**CPU Scheduling**. Basic Concepts ,Scheduling Criteria, Scheduling Algorithms, Thread Scheduling, Multiple-Processor Scheduling, Real-Time CPU Scheduling, Operating-System Examples.		
Week 6	3 +1	**Deadlocks** System Model, Deadlock Characterization, Methods for Handling Deadlocks,	Quiz 2 Assig 2	
Week 7	3 +1	**Deadlocks continue..** Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock		
Week 8	3 +1	**Memory Management (Main Memory)** Background, Swapping, Contiguous Memory Allocation, Segmentation, Paging ,Structure of the Page Table		
Week 9	3 +1	**Memory Management (Virtual Memory)** Background, Demand Paging, Copy-on-Write, Page Replacement, Allocation of Frames,		
Week 10	3 +1	**Memory Management (Virtual Memory)** Thrashing, Memory-Mapped Files, Allocating Kernel Memory	Quiz 3 Assig 3	
Week 11	3 +1	**Storage Management (Mass-Storage Structure)** Overview of Mass-Storage, Structure, Disk Structure, Disk Attachment, Disk Scheduling		
Week 12	3 +1	**Storage Management (Mass-Storage Structure)** Disk Management, Swap-Space Management, RAID Structure		
Week 13	3 +1	**File-System Interface** File Concept ,Access Methods, Directory and Disk Structure, File-System Mounting, File Sharing, Protection		
Week 14	3 +1	**I/O Systems** Overview , I/O Hardware, Application I/O Interface, Kernel I/O Subsystem, Transforming I/O Requests to Hardware Operations		
Week 15	3 +1	Revision		
Week 16	3 +1	**Revision**		

Assessment

- 2 x Quiz Marks 10
- 2 x Assignment = 10
- Class Participation = 05
- Project = 15
- Mid Exam = 20
- Final Exam = 40

Chapter 1: Introduction



Outline and Objectives

Outline

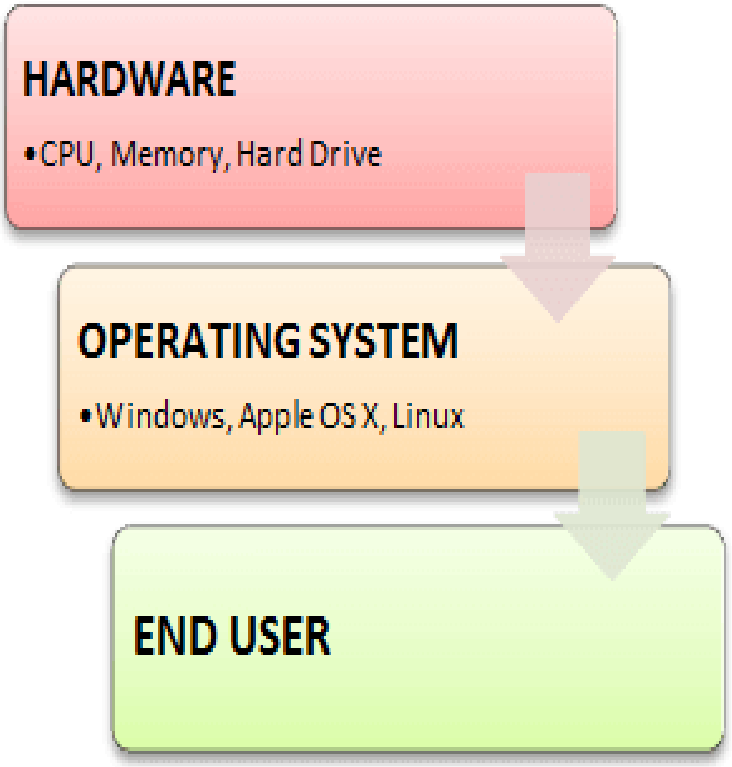
- What Operating Systems Do
- Computer-System Organization
- OS structure and operation
- Major OS Functions
  - Process Management
  - Memory Management
  - Storage Management
  - Protection and Security
- Computing Environments

Objectives

- To provide a grand tour of the major operating systems components
- To provide coverage of basic computer system

What is an operating system?

- An **Operating system (OS)** is a software which acts as an interface between the end user and computer hardware.
- Every computer must have at least one OS to run other programs
- An application like Chrome, MS Word, Games, etc needs some environment in which it will run and perform its task.

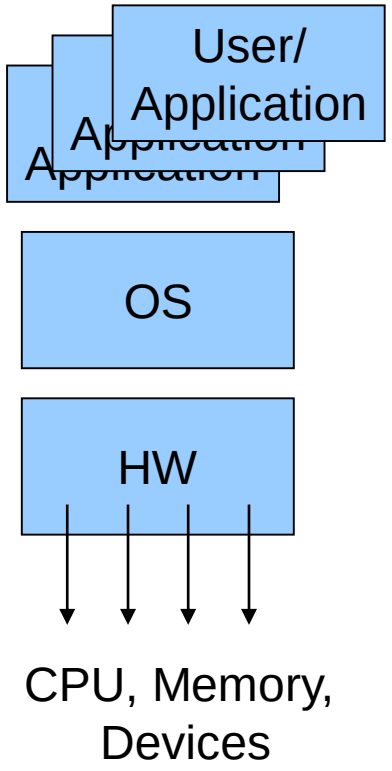


Reference: https://www.guru99.com/operating-system-tutorial.html#2

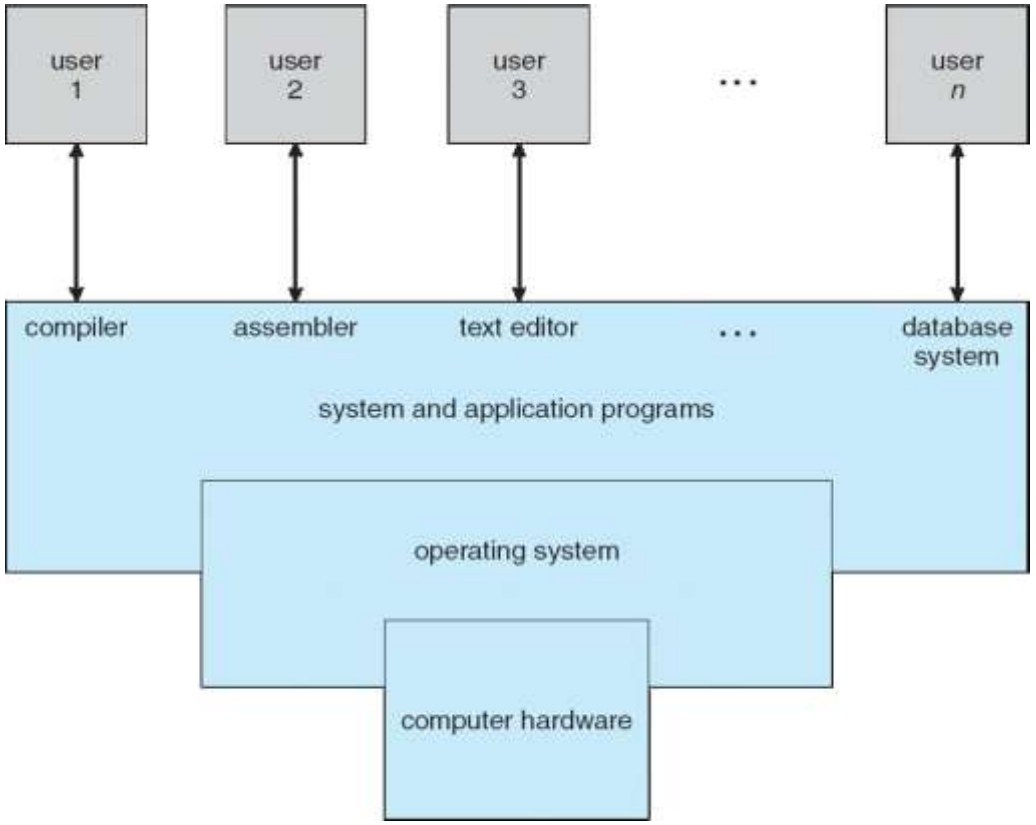
What is an operating system?

- A program that acts as an *intermediary* between a users/applications and the computer hardware

- Operating system functionalities/goals
  - Start/terminate/control executing user programs
  - Make system convenient to use
  - Control and coordinate use of hardware
    - Perform I/O; setup devices
    - **Allocate resources**
    - Use hardware efficiently
  - Implement **common services**



Basic components of a computer system: place of OS

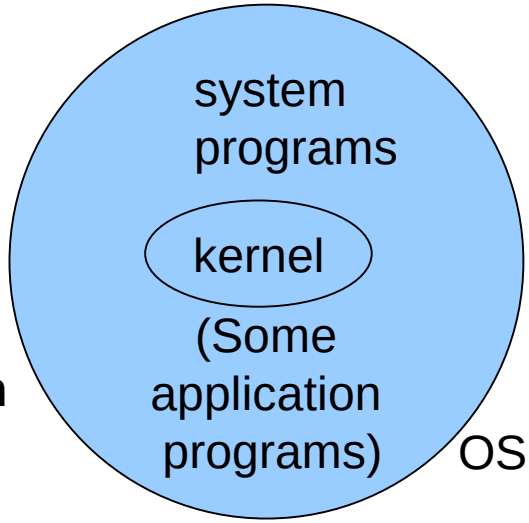


- A computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - **Controls and coordinates use of hardware among various applications and users**
  - Application programs –solve the problems of the users: use system resources
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Operating System Definition

- No universally accepted definition
  - "Everything a vendor ships when you order an operating system" is good approximation
    - But varies wildly
- **Kernel:** running all the time; having most of the functionality
- Everything else: either a **system program** (ships with the operating system) or an **application program**

***System programs*: programs that are associated with the operating system**



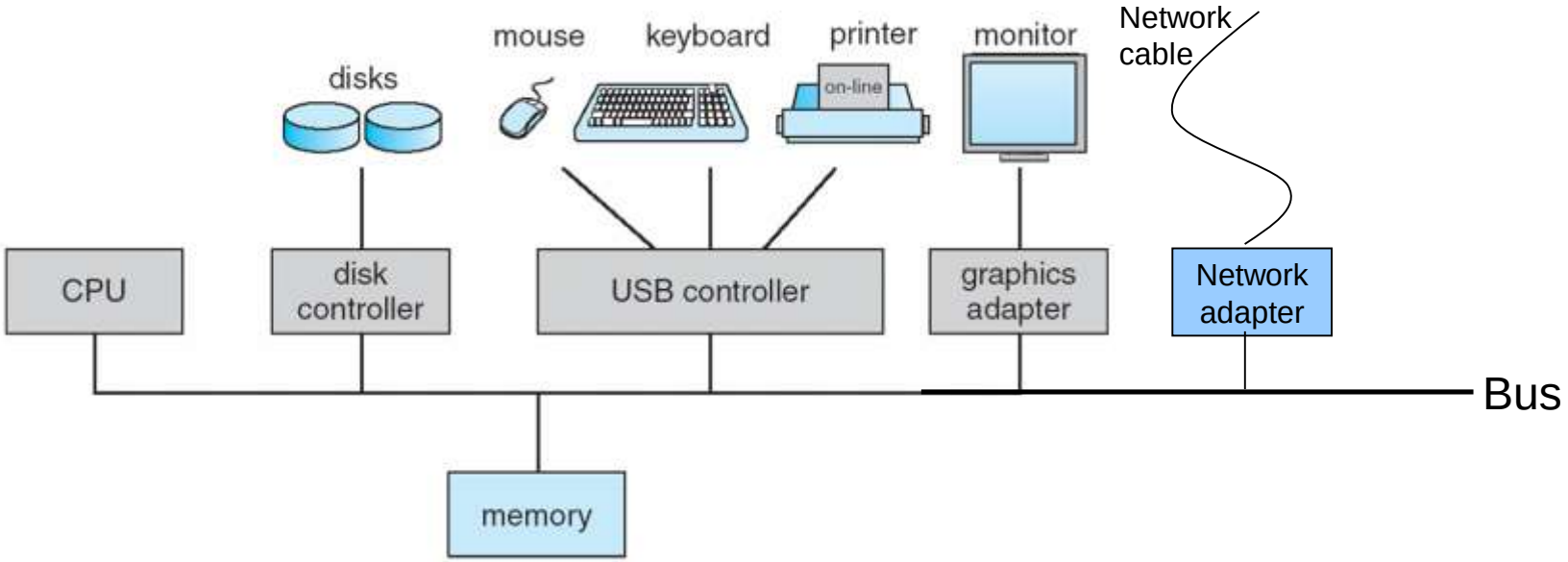
+ you can install other applications

OS CD

Computer System Organization and Operation

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



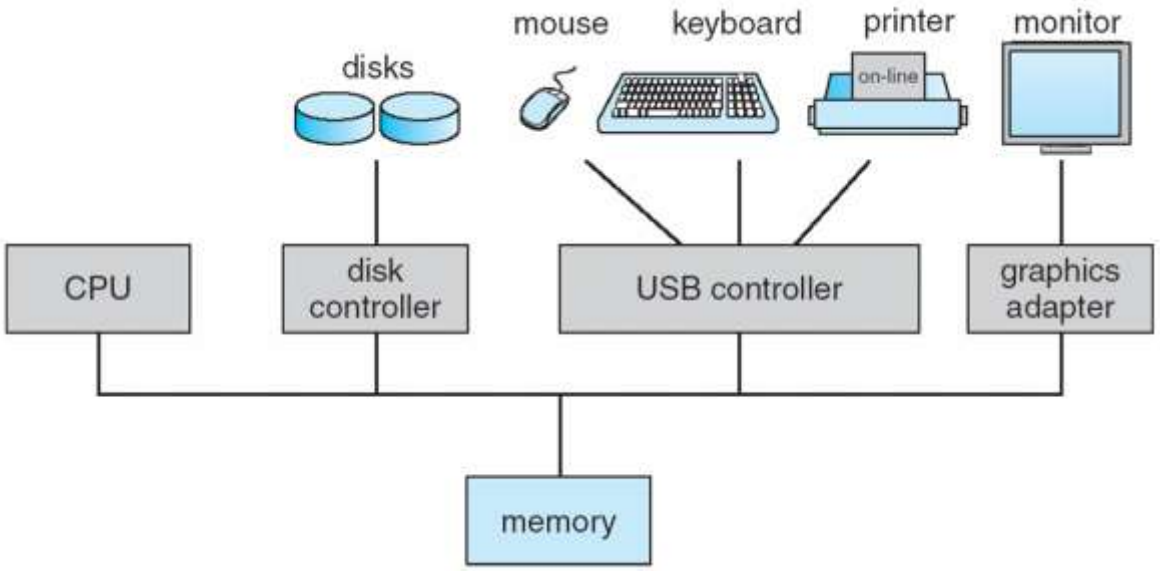
Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM,
  - generally known as **firmware**
  - Initializes all aspects of the system
  - Loads operating system **kernel** and starts execution
- Kernel runs and make the system ready for running applications
  - Kernel is always ready to run (always in memory)

Computer system operation: I/O and device interaction

- I/O devices and the CPU can execute concurrently
- Each device controller has a local buffer
  - Data movement (I/O) between device and local buffer (device
  - Data movement between memory and local buffer (by CPU)

- Device controller informs CPU that it has finished its current operation or it has something by causing an *interrupt*

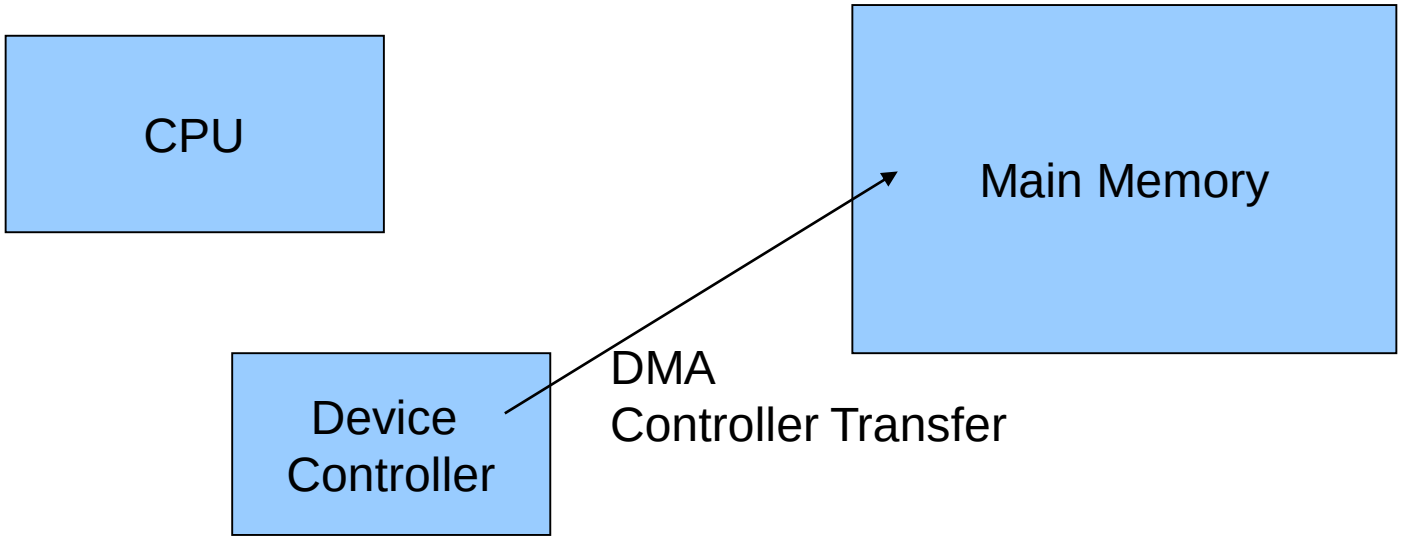


Hardware interrupts

- When interrupt occurs, hardware does the following:
  - CPU is interrupted
    - at that time application code or kernel code might be running
    - registers and the program counter saved in RAM to preserve CPU state
  - CPU starts running the respective Interrupt Service Routing (ISR)
      - (kernel routine)
    - ISR is found through **interrupt vector**
      - (table containing addresses of ISRs)

Direct Memory Access Structure

- With DMA, 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

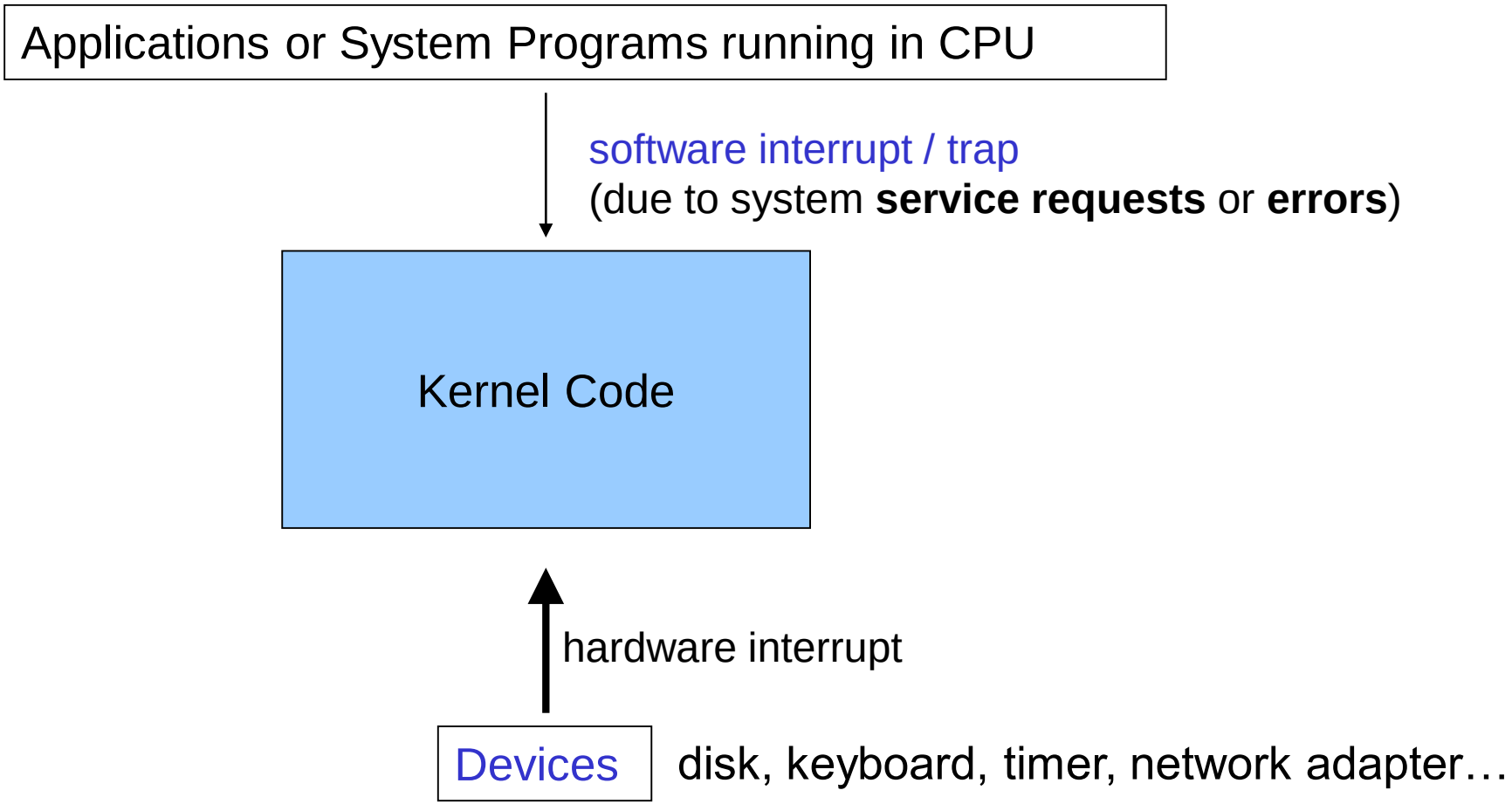


Software interrupts

- Running application software may generate interrupts as well.
  - They are called software interrupts (also called traps)
    - 1. exceptions (caused by errors)
    - 2. system calls (service request)
      - **trap** or **syscall** instruction is used

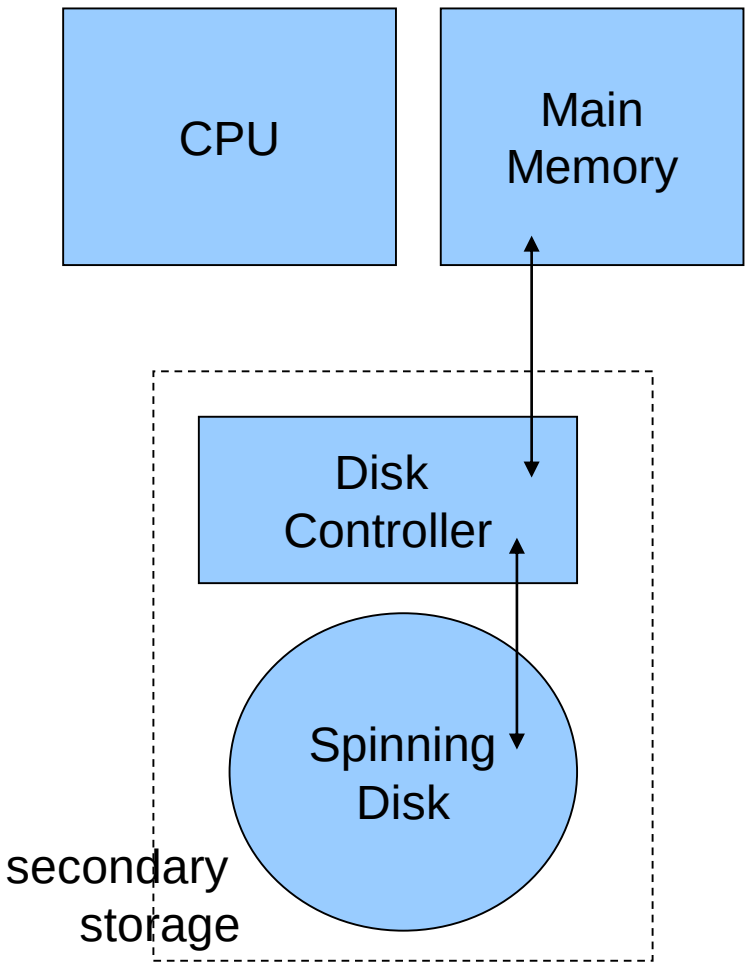
- An operating system (kernel) is interrupt-driven (event driven)

Interrupt-Driven OS

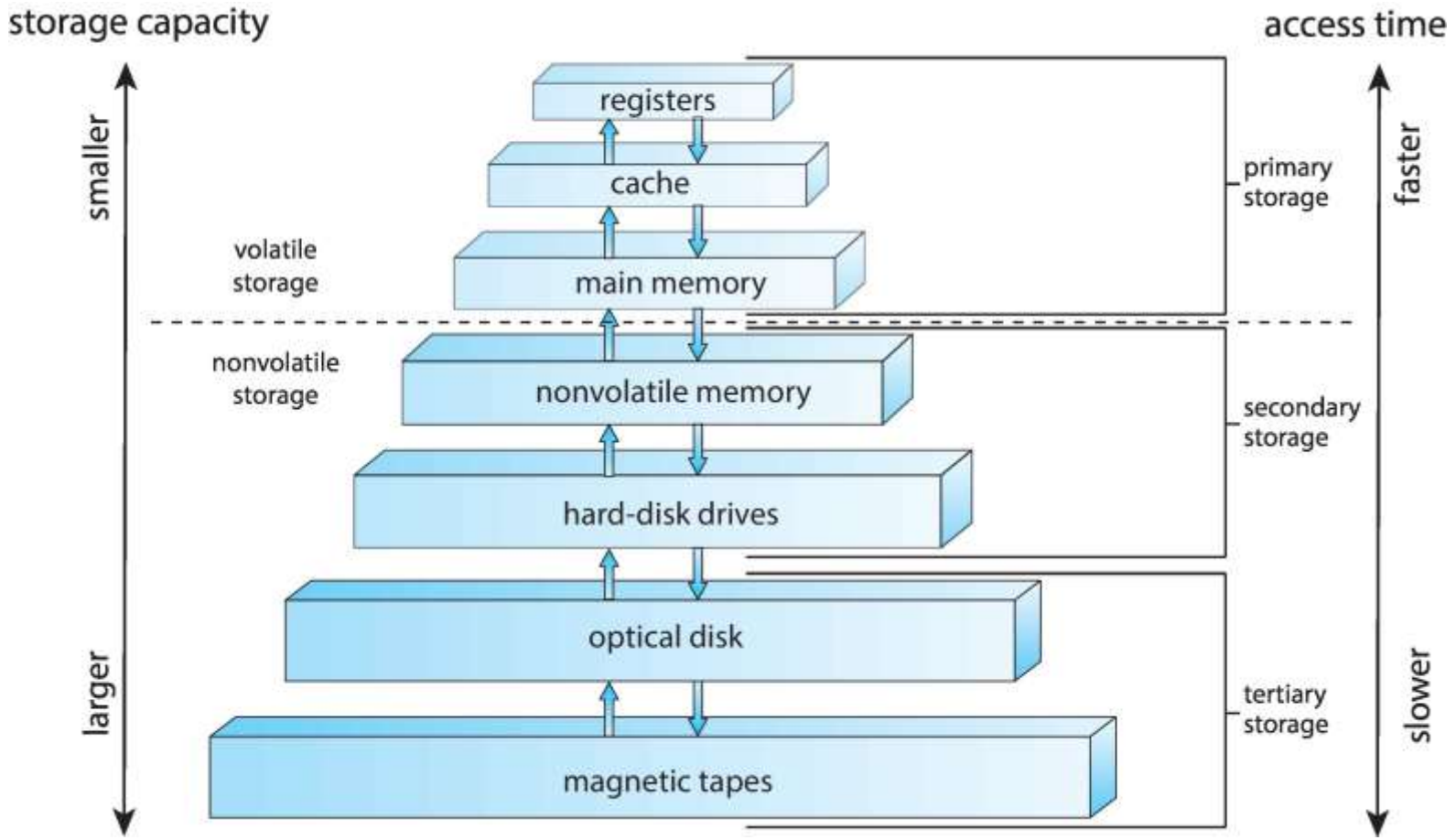


Storage Structure

- **Main memory** – CPU can access directly
- **Secondary storage** – extension of main memory that provides large nonvolatile storage capacity
  - Magnetic disks
    - platters
    - The **disk controller** determines the interaction between the device and the computer

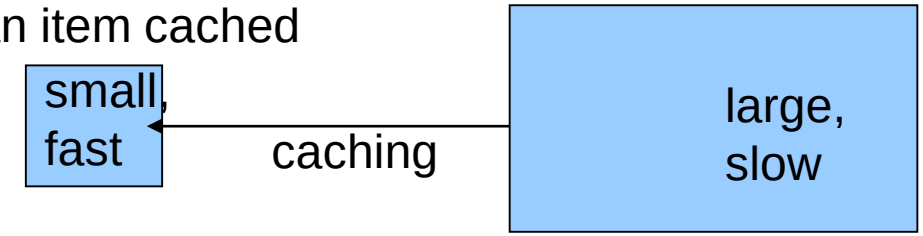


Storage-Device Hierarchy



Caching

- **Caching** – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage
  - Results from tradeoff between size and speed
- performed at **many levels** in a computer (in hardware, operating system, software)
- Cache is checked first for an item cached



Registers	Main Memory
Hardware cache L1, L2, etc	Main Memory
Main Memory	Hard Disk
Hard Disk	Tape

Performance of various levels of storage

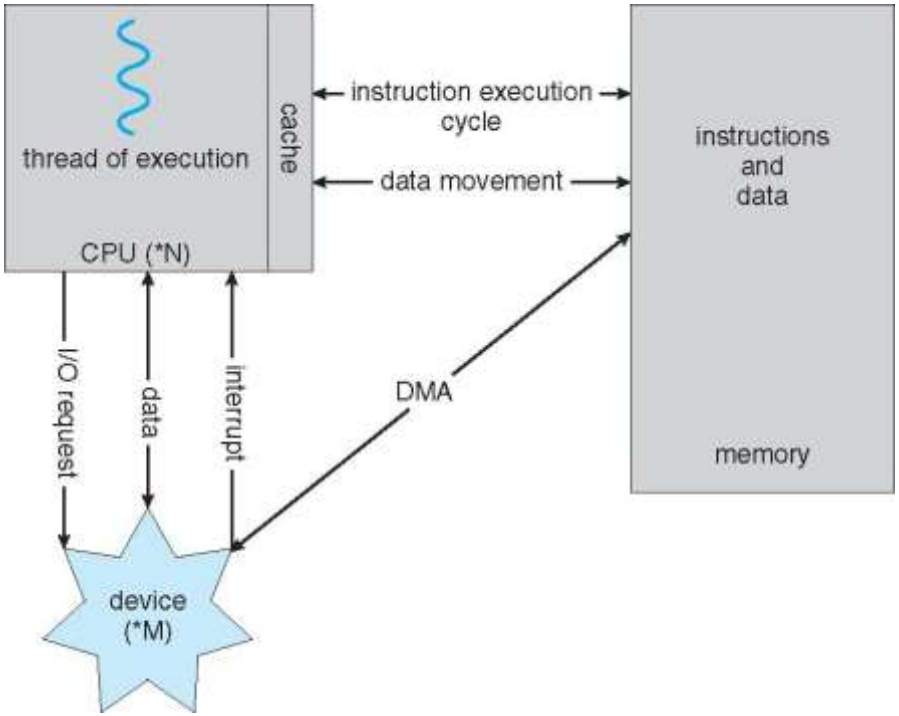
- Movement between levels of storage hierarchy can be explicit or implicit.

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Computer System Architecture

Computer System Architecture: Single processor systems

- Most systems use a **single general-purpose processor**
  - (PDAs through mainframes)
  - Most systems have special-purpose processors as well



Computer System Architecture: Multiprocessor systems

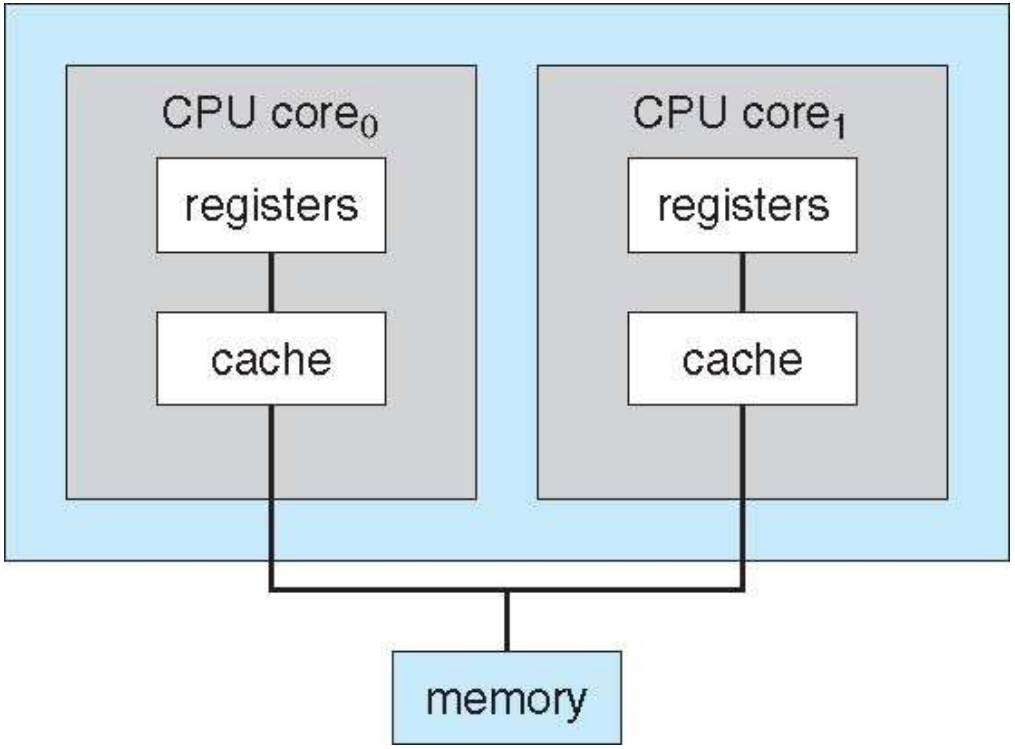
- **Multiprocessor systems** growing in use and importance
  - Also known as parallel systems, tightly-coupled systems
  - Advantages include
    1. Increased throughput
    2. Economy of scale (cheaper than using multiple computers)
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

Symmetric Multiprocessing Architecture

In Symmetric Multiprocessing, processors shares the same memory

In Asymmetric Multiprocessing, processors do not shares the same memory

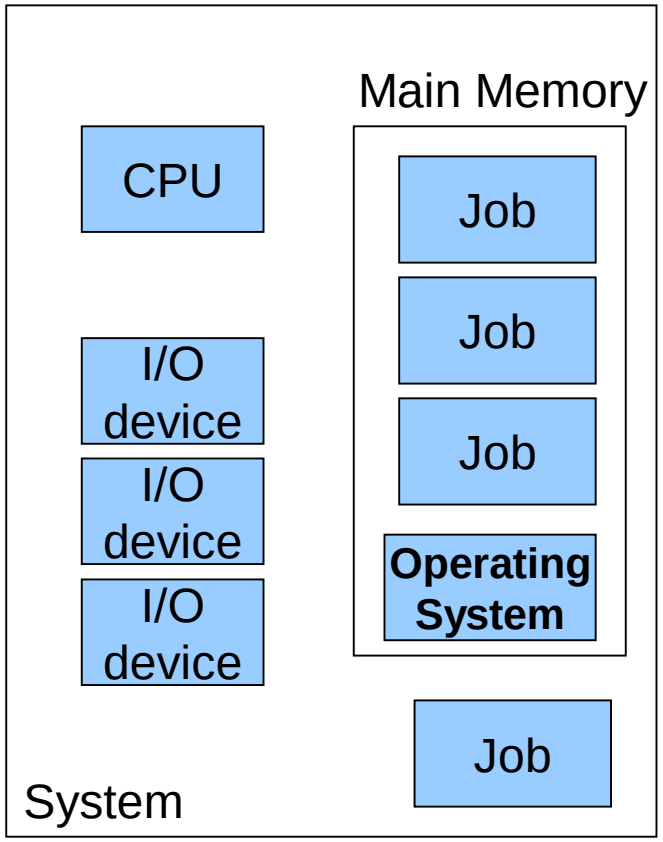
A Dual Core Design



Operating System and Functionalities

Operating Systems: providing multiprogramming

- **Multiprogramming** 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**
    - OS selects which job
  - When it has to wait (for I/O for example), OS switches to another job

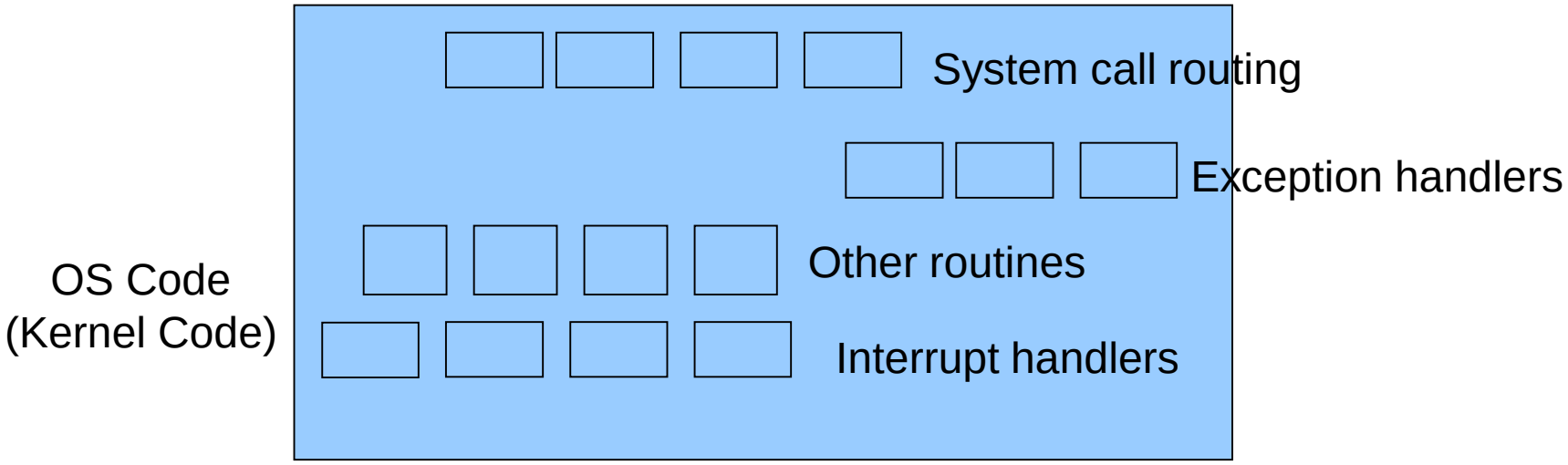


Operating Systems: providing time sharing

- **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
  - program loaded in memory ⇨**process**
  - If several processes ready to run at the same time ⇨ **CPU scheduling**

Operating System: how operates

- is interrupt driven
  - Hardware interrupt causes ISR to run (which is a routine of OS)
  - Software error or request creates **exception** or **trap**
    - Division by zero, for example (exception)
    - request for an operating system service (trap)



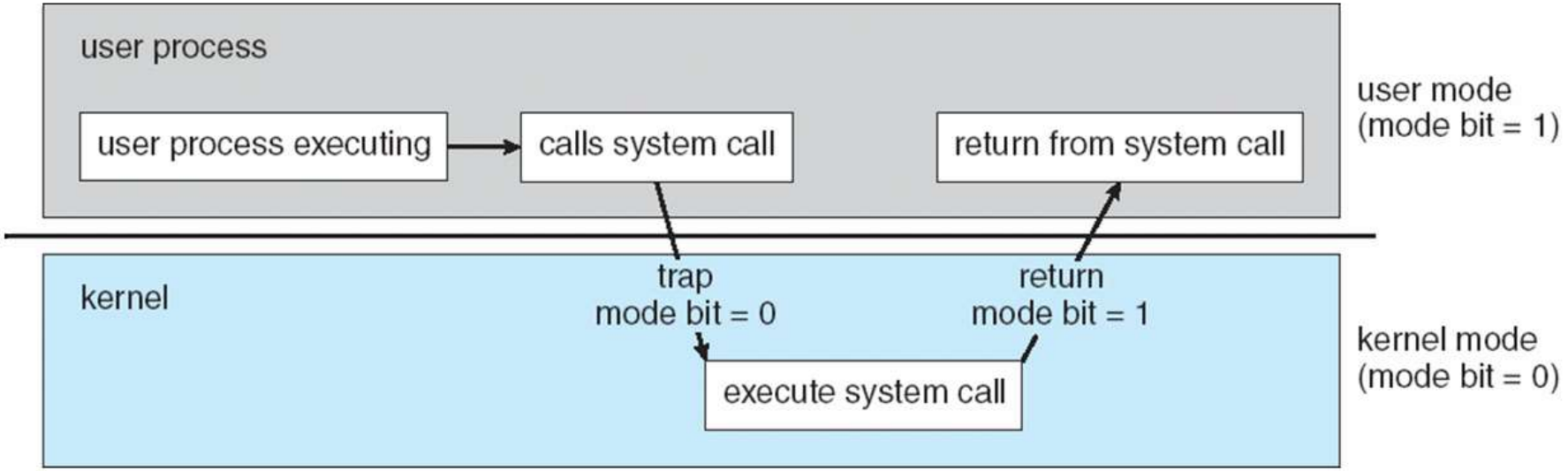
Operating System: how operates

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - user code or kernel code in different modes
    - Some machine instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user mode

Operating System: how operates

Dual mode system operation

Transition from User to Kernel Mode and Vice Versa



Operating System: how operates

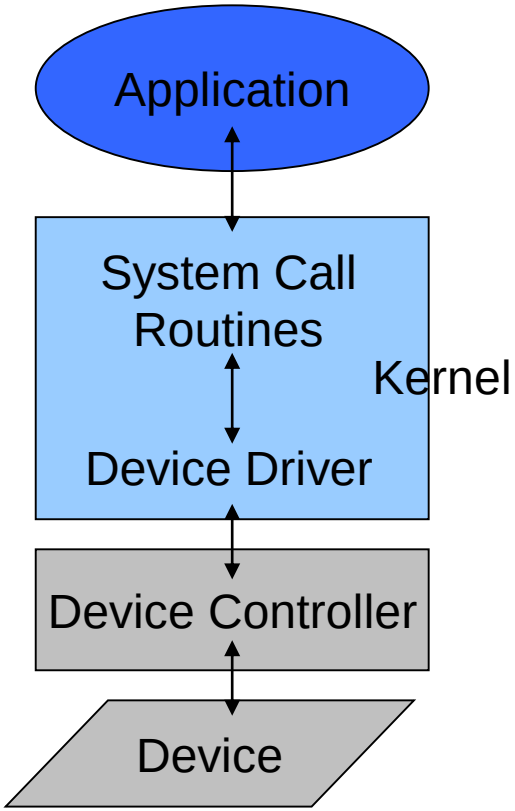
- **Timer device** to prevent infinite loop / process hogging resources
  - 1) Set the timer device to interrupt after a while
    - Can be a fixed or variable time period
  - 2) CPU executes a program (a process)
  - 3) Timer device sends an interrupt after that period
  - 4) CPU starts executing timer handler: OS gains control
  - 5) OS can schedule the same process or other process
  - 6) OS sets the timer again before giving the CPU to the scheduled process

Major OS Functionalities

Reference: https://www.guru99.com/operating-system-tutorial.html#2

I/O Structure

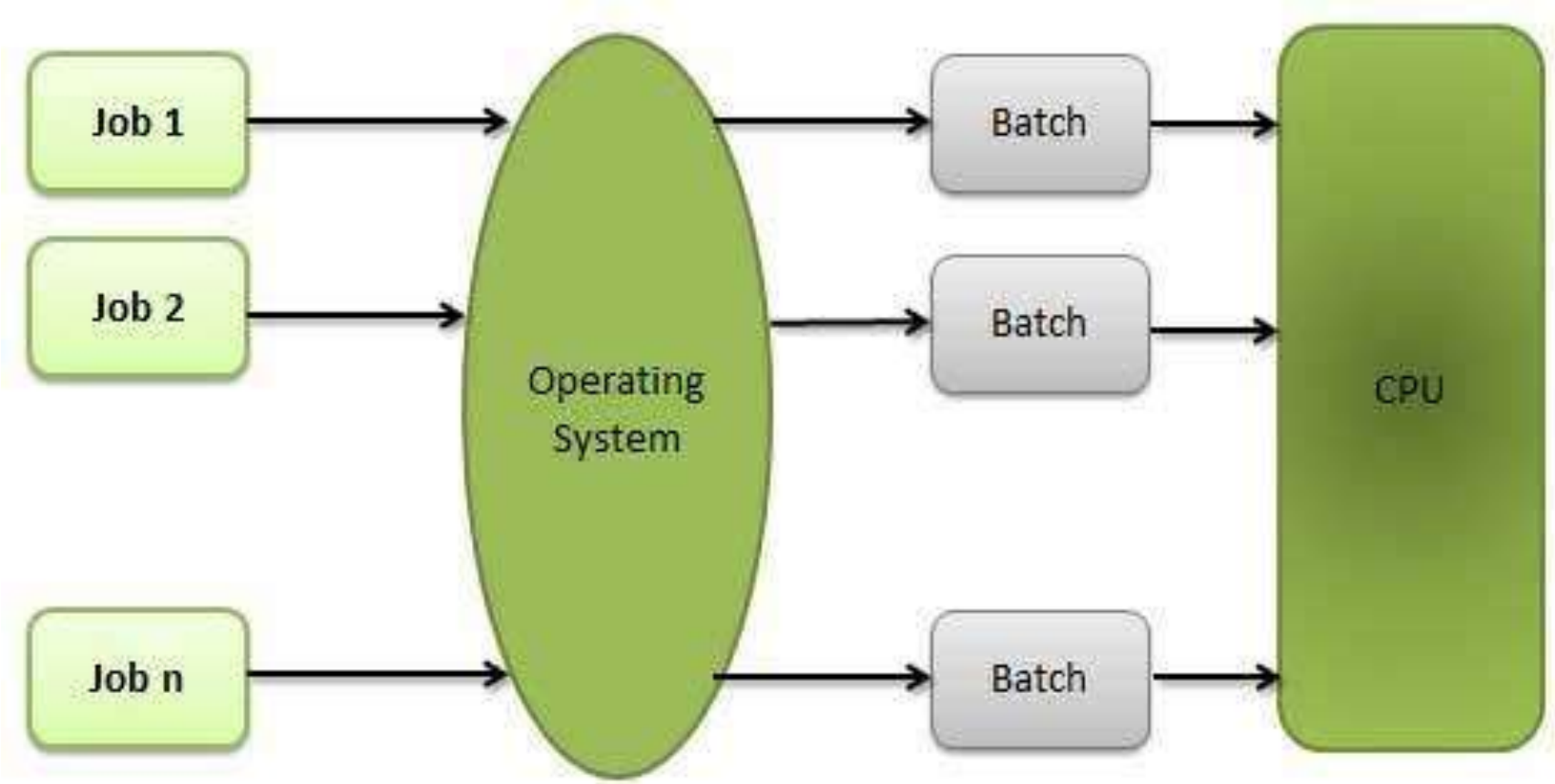
- Application programs do I/O via OS
  - The request is done by calling a System Call (OS routine)
  - System call routine in OS performs the I/O via the help of device driver routines in OS.
  - After issuing a system call, an application may wait for the call to finish (blocking call) or may continue to do something else (non-blocking call)



Types of Operating system

- Batch Operating System
- Multitasking/Time Sharing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

Batch Operating System

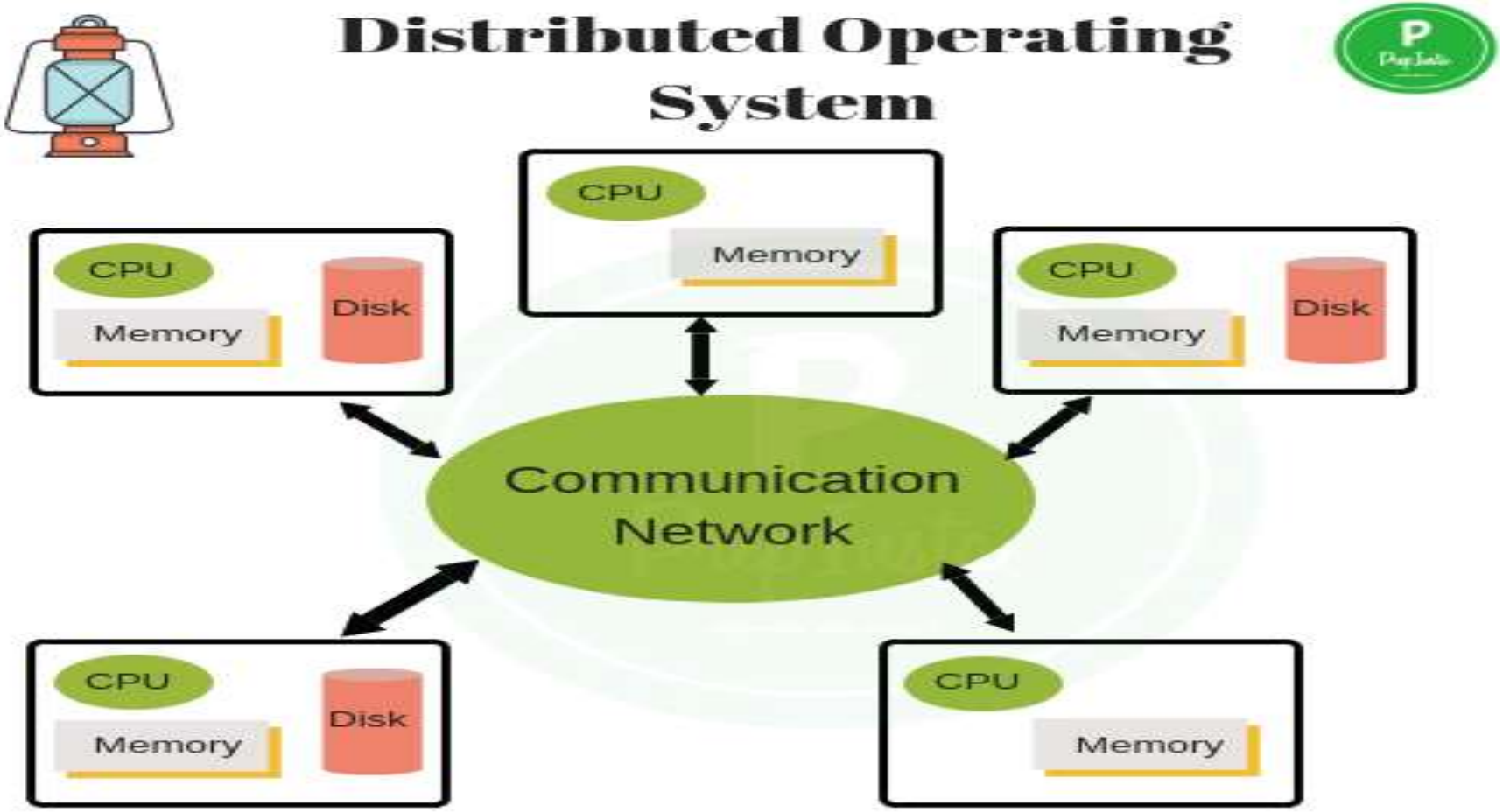


Multitasking/Time Sharing OS

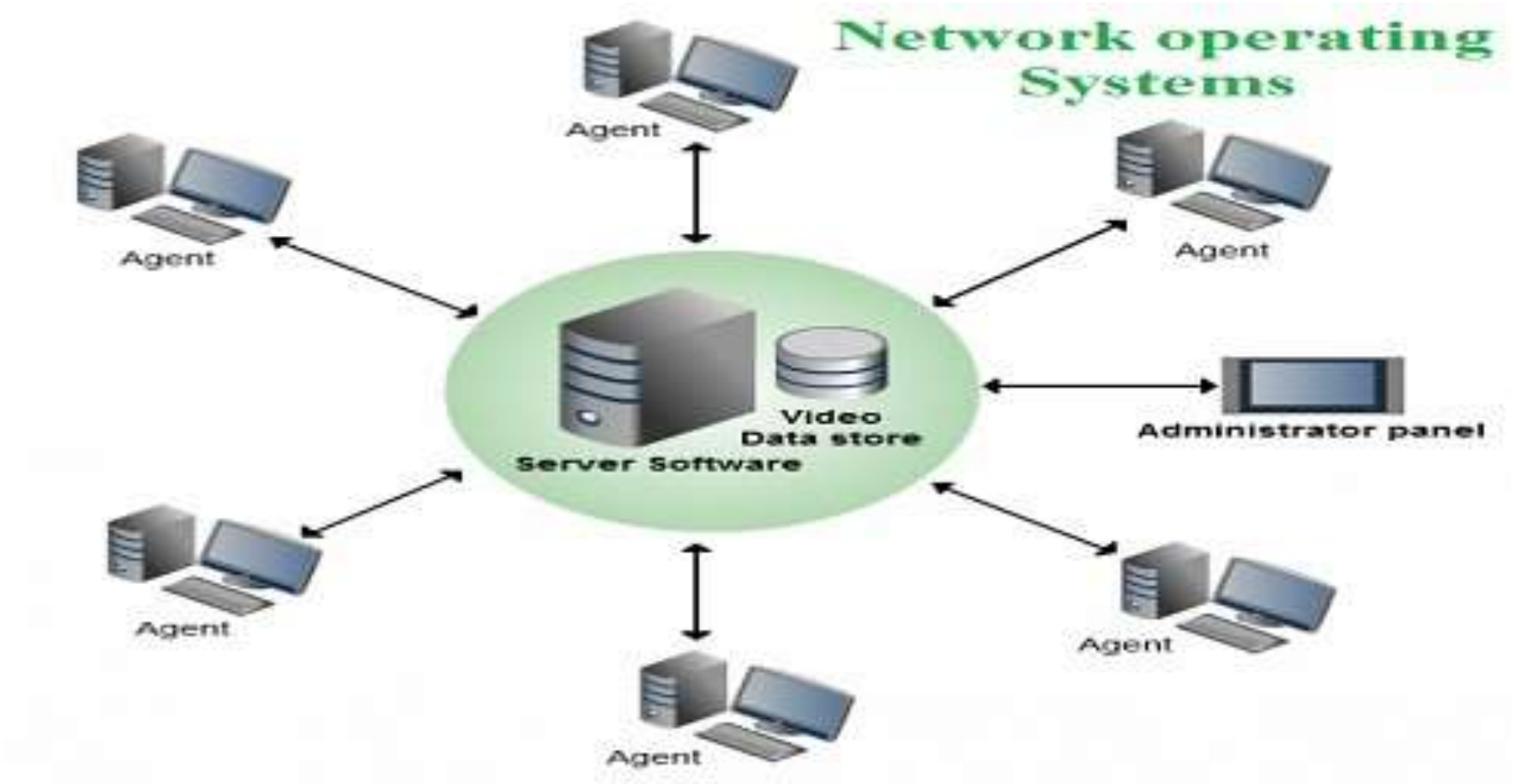
Real Time OS

- A real time operating system time interval to process and respond to inputs is very small. Examples: Military Software Systems, Space Software Systems.

Distributed OS



Network OS


Network operating Systems
Agent
Agent
Agent
Video
Data store
Server Software
Administrator panel
Agent
Agent
Agent

Mobile OS

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to the copy protection movement
- Examples include
  - GNU/Linux,
  - BSD UNIX (FreeBSD, etc.)
  - Sun Solaris

References

- Operating System Concepts, 7th and 8th editions, Silberschatz et al. Wiley.
- Modern Operating Systems, Andrew S. Tanenbaum, 3rd edition, 2009.
- These slides are adapted/modified from the textbook and its slides:
  Operating System Concepts, Silberschatz et al., 7th and 8th editions, Wiley.
- **Reference: https://www.guru99.com/operating-system-tutorial.html#2**

Additional Study Material