OPERATING SYSTEMS
CS3500

PROF. SUKHENDU DAS

DEPTT. OF COMPUTER SCIENCE AND ENGG.,

IIT MADRAS, CHENNAI – 600036.

Email: **sdas@cse.iitm.ac.in**
URL: **//www.cse.iitm.ac.in/~vplab/os.html**
https://sites.google.com/smail.iitm.ac.in/3500-os/

Better follow: //
Office – SSB302; Ph – 4367 / 5389

July-Nov. – 2022.

INTRODUCTION

Contents to be covered

Part 1: Overview
1. Introduction
2. Operating-System Services

Part 2: Process Management
3. Processes
4. Threads and Concurrency
5. CPU Scheduling

Part 3: Process Synchronization
6. Synchronization Methods
7. Deadlocks

Part 4: Memory Management
8. Main Memory
9. Virtual Memory

Contents to be covered

Part 5: Storage Management
10. File-System Interface
11. File-System Implementation
12. Mass-Storage Structure
13. I/O Systems

Part 6: Protection and Security
14. Protection
15. Security

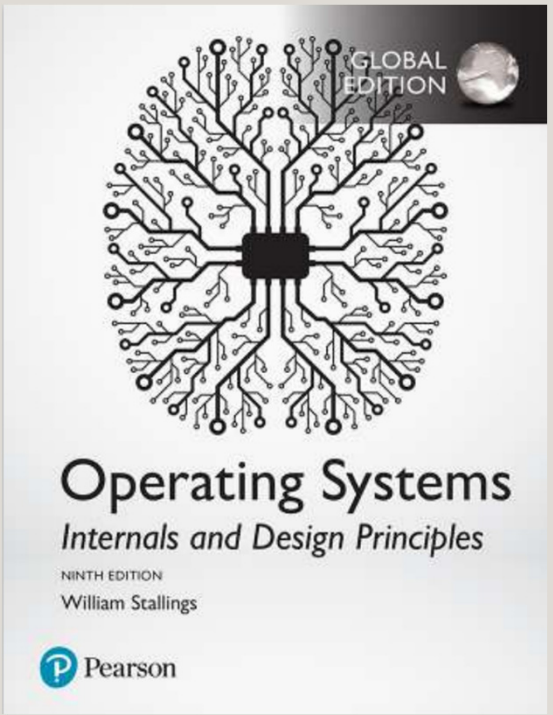
Part 7: Case Studies
17. The Linux System
18. Windows 10

Part 8: Advance Topics
19. Virtual Machines
20. Networks and Distributed Systems

REFERENCES

I. Operating System Concepts
Tenth Edition

- Avi Silberschatz
- Peter Baer Galvin
- Greg Gagne



II. Operating Systems Internals and Design Principles
Ninth Edition

- William Stallings

TENTATIVE GRADING POLICY

(SUBJECT TO APPROVAL OF CSE-CC)

Mid-Sem (Quiz)	---	15	* 1 hour
Tutorials	---	10	* best of n - 1 from n
Lab Assignments	---	25	
End- Sem	---	50	* 3 hours

TIME TABLE

May be held Online
Till end of Aug 22;
12.15 – 13.00 Hrs

Days	8.00 - 8.50	9.00 - 9.50	10.00 - 10.50	11.00 - 11.50		13.00 - 13.50	14.00 - 15.15	15.30 - 16.45	17.00 - 17.50
Mon	A	B	C	D		G	P		J/J3
							H/H1	M/M2	
Tue	B	C	D	E		A	Q		F
							M/M1	H/H2	
Wed	C	D	E	F		B	R		G
							J/J1	K/K2	
Thu	E	F	G	A		D	S		H/H3
							L/L1	J/J2	
Fri	F	G	A	B		C	T		E
							K/K1	L/L2	

Lunch recess 11.50 am - 13.00 pm

→ *TUTs; once a month*

- Lab for Assignments – starts Aug. 05

WHAT IS AN OPERATING SYSTEM?

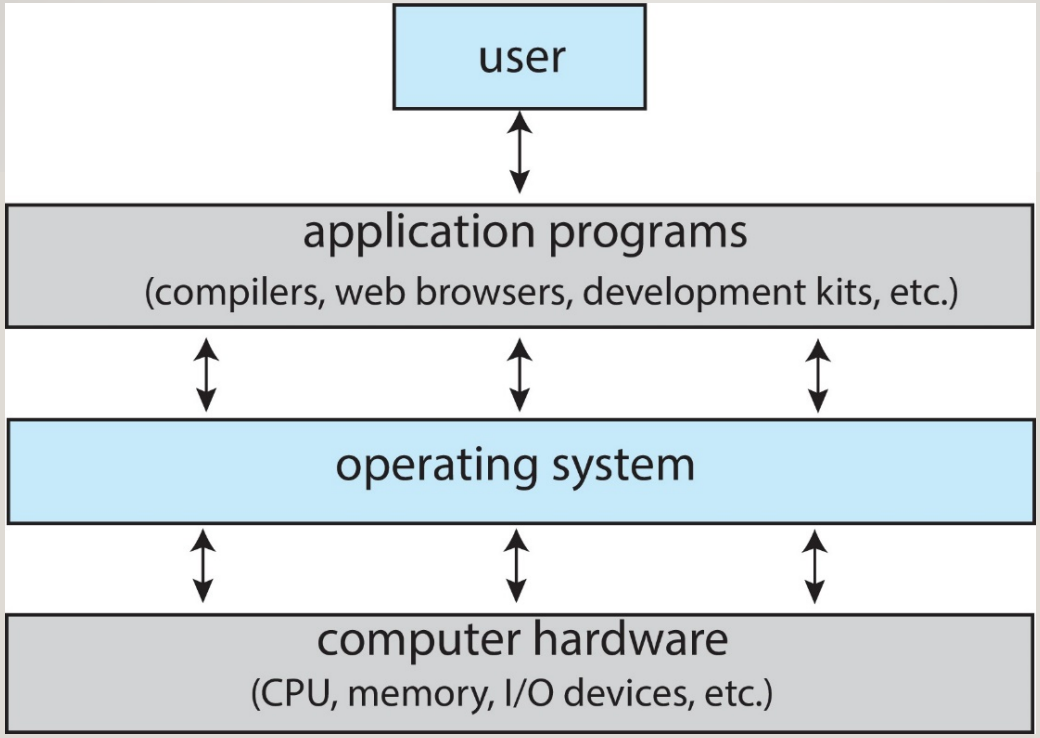
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

Computer system can be divided into four components:

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- Users
  - People, machines, other computers

Where Operating System lies?



.dll

.drv

.sys

.cab

.reg

.vga

Figure1 : Abstract View of Components of a Computer

What Operating Systems Do?

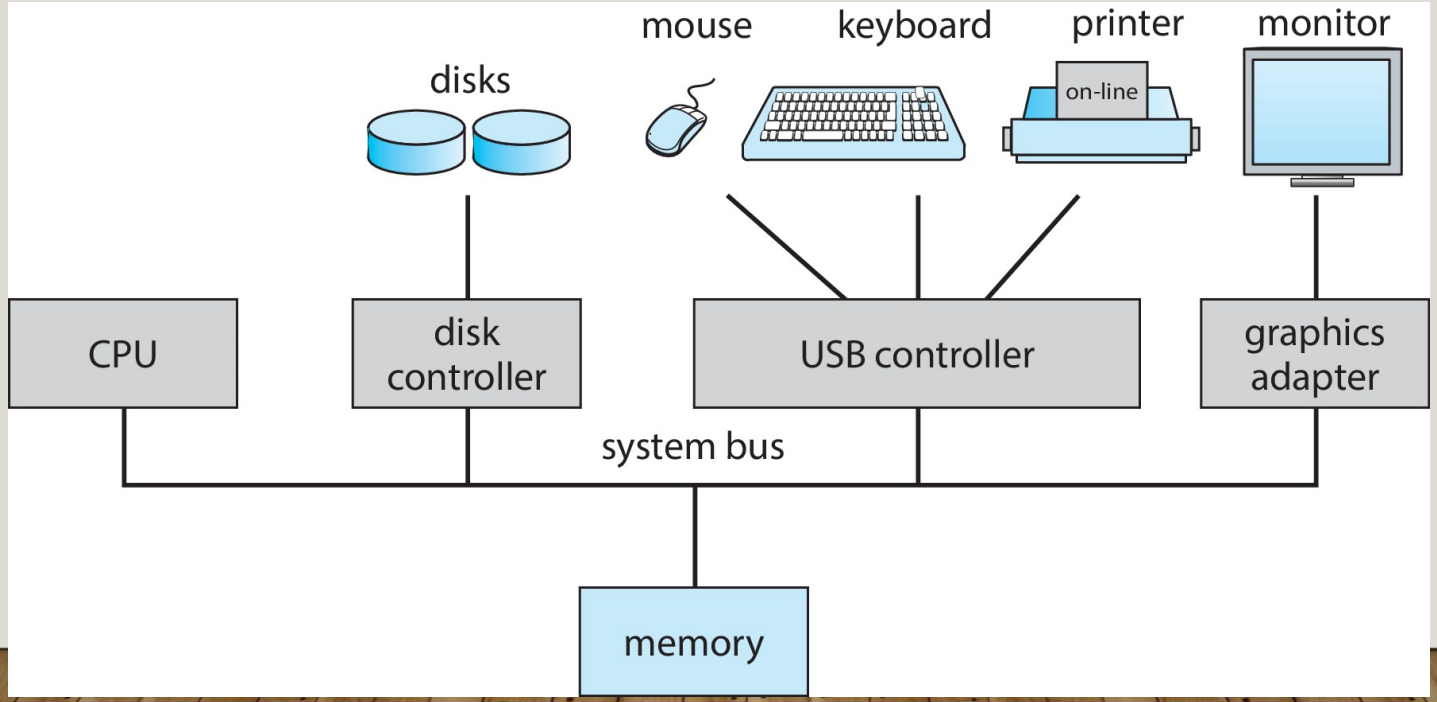
- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
  - Operating system is a **resource allocator** and **control program**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

Operating System Definition

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel,** 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 additional services to application developers such as databases, multimedia, graphics

Overview of Computer System Structure

- 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



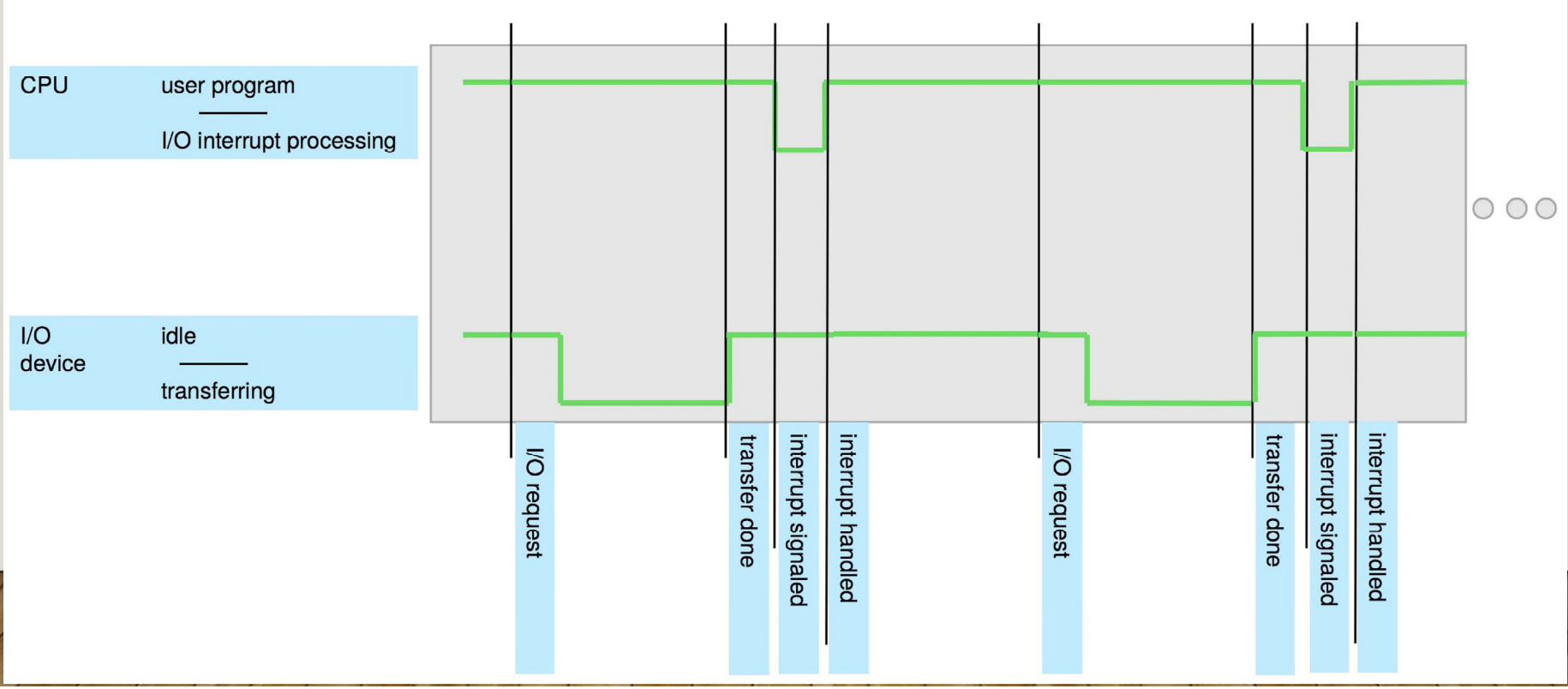
Computer-System Operation

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

Common Functions of Interrupts

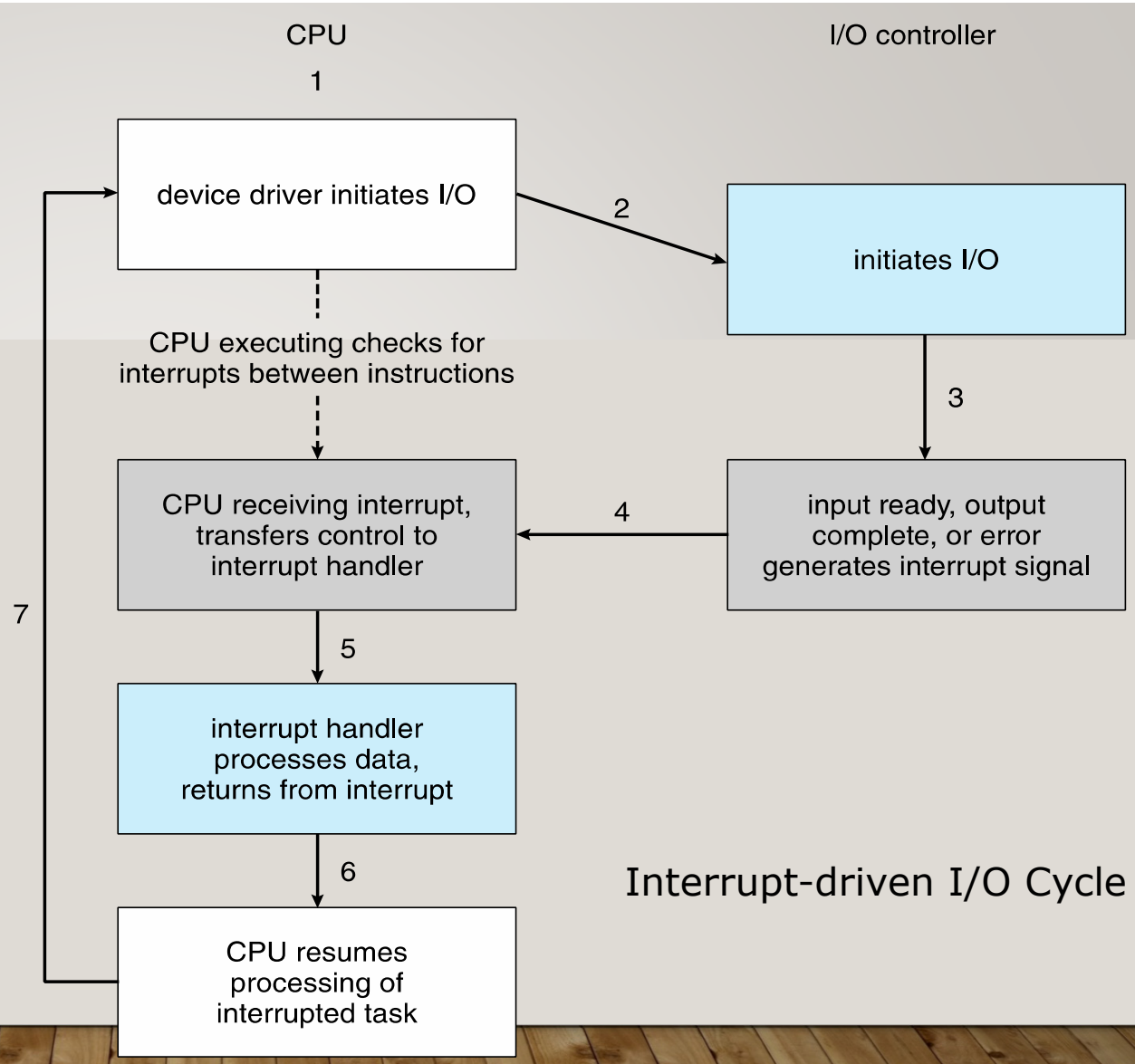
- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

Interrupt Timeline



Interrupt Handling

- The operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred
- Separate segments of code determine what action should be taken for each type of interrupt



Interrupt-driven I/O Cycle

I/O Structure

- Two methods for handling I/O
  - After I/O starts, control returns to user program only upon I/O completion
  - After I/O starts, control returns to user program without waiting for I/O completion

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing

- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the OS to allow user to wait for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Computer Startup

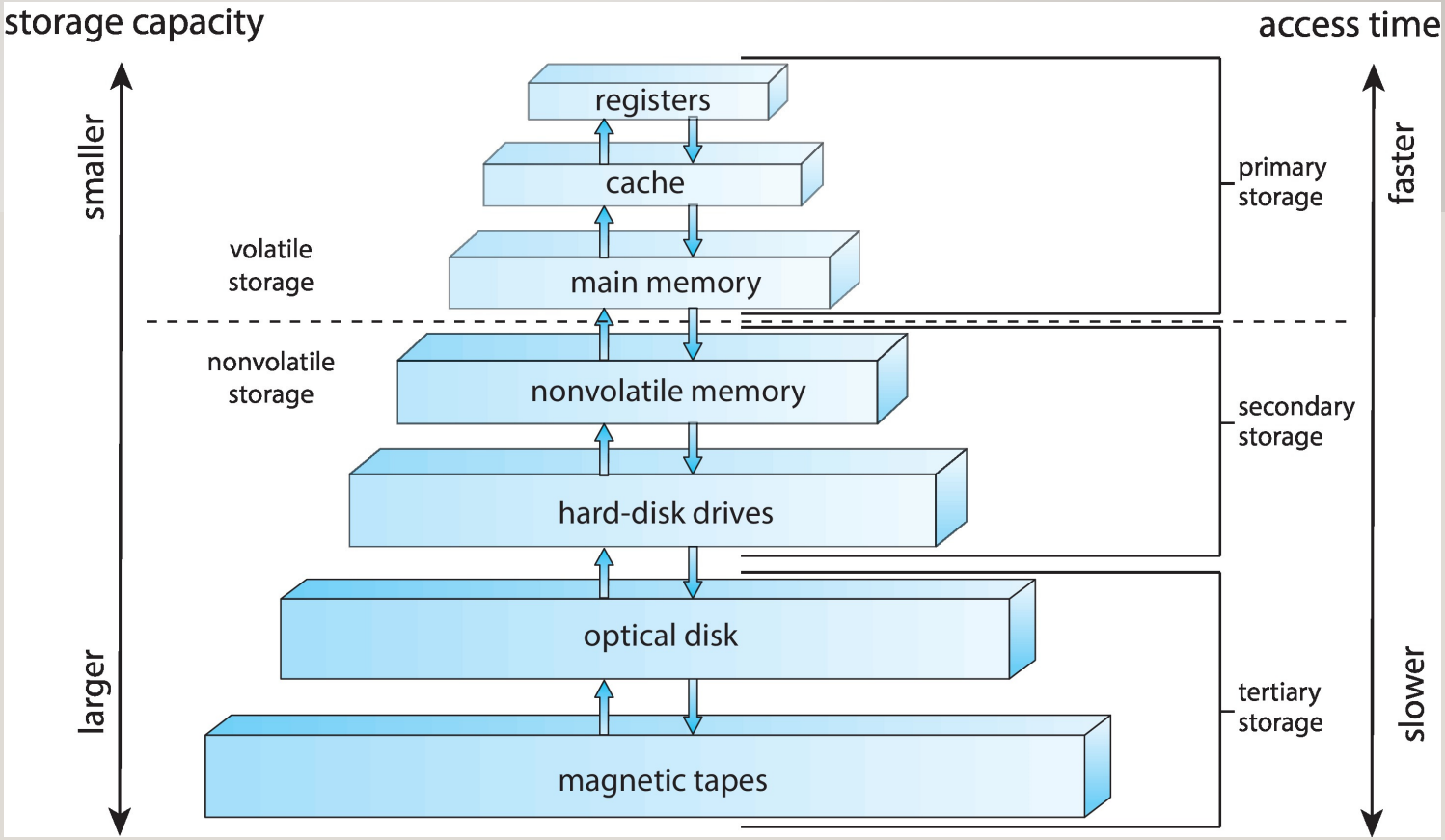
- **Bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
  - Typically **random-access memory** in the form of **Dynamic Random-access Memory (DRAM)**
- 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
  - 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
  - Becoming more popular as capacity and performance increases, price drops

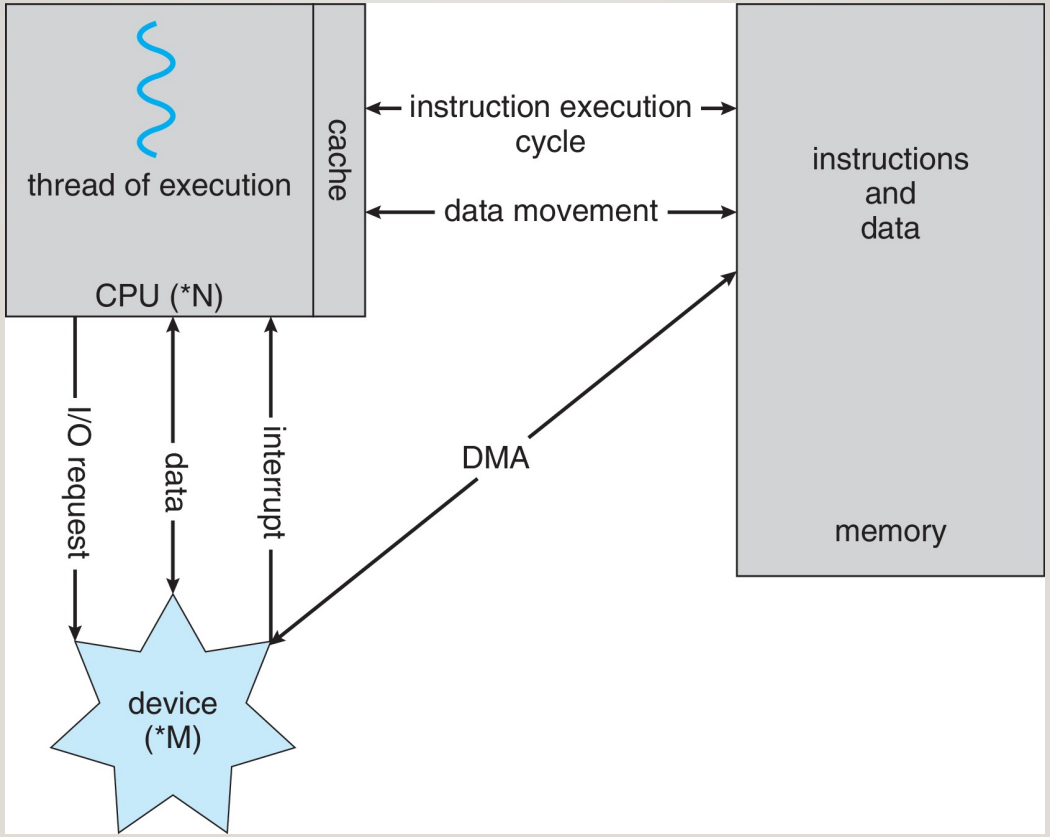
Storage Hierarchy

- 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

How a Modern Computer Works



A von Neumann architecture

Direct Memory Access (DMA) 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

Operating-System Operations

- program – simple code to initialize the system, load the kernel
- Kernel loads
- Starts **system daemons** (services provided outside of the kernel)
- 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

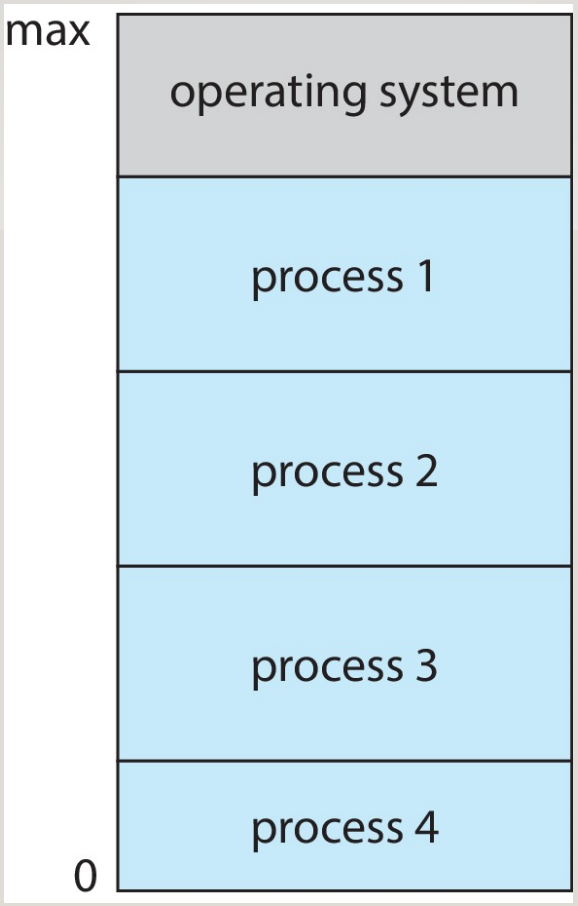
MULTIPROGRAMMING (BATCH SYSTEM)

- Single user cannot always keep CPU and I/O devices busy
- 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 job has to wait (for I/O for example), OS switches to another job

MULTITASKING (TIMESHARING)

- A logical extension of Batch systems– the 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

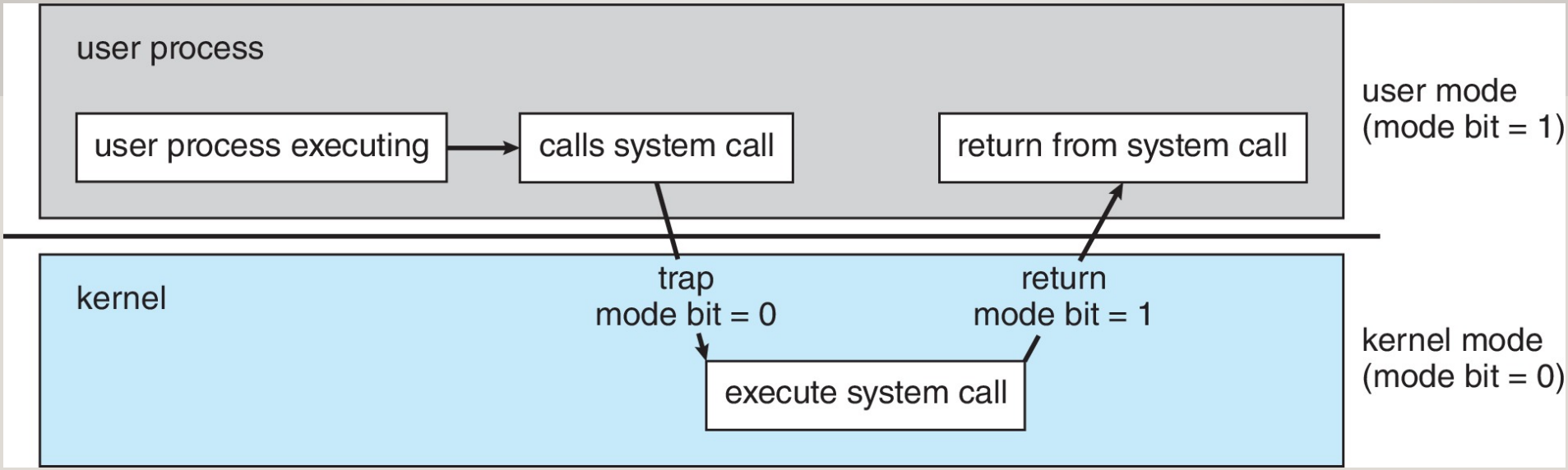
MEMORY LAYOUT FOR MULTIPROGRAMMED SYSTEM



DUAL-MODE 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.
  - When a user is running ⇨ mode bit is "user"
  - When kernel code is executing ⇨ mode bit is "kernel"
- How do we guarantee that user does not explicitly set the mode bit to "kernel"?
  - System call changes mode to kernel, return from call resets it to user
- Some instructions designated as **privileged**, only executable in kernel mode

TRANSITION FROM USER TO KERNEL MODE



TIMER

- Timer to prevent infinite loop (or 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

Operating System Tasks

- Process Management
- Memory Management
- File System Management
- Storage Management
- Protection and Security

Categories of OS (general)

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

Distributed Systems

- Collection of separate, possibly heterogeneous, systems networked together
  - **Network** is a communications path, **TCP/IP** most common
    - **Local Area Network** (**LAN**)
    - **Wide Area Network** (**WAN**)
    - **Metropolitan Area Network** (**MAN**)
    - **Personal Area Network** (**PAN**)

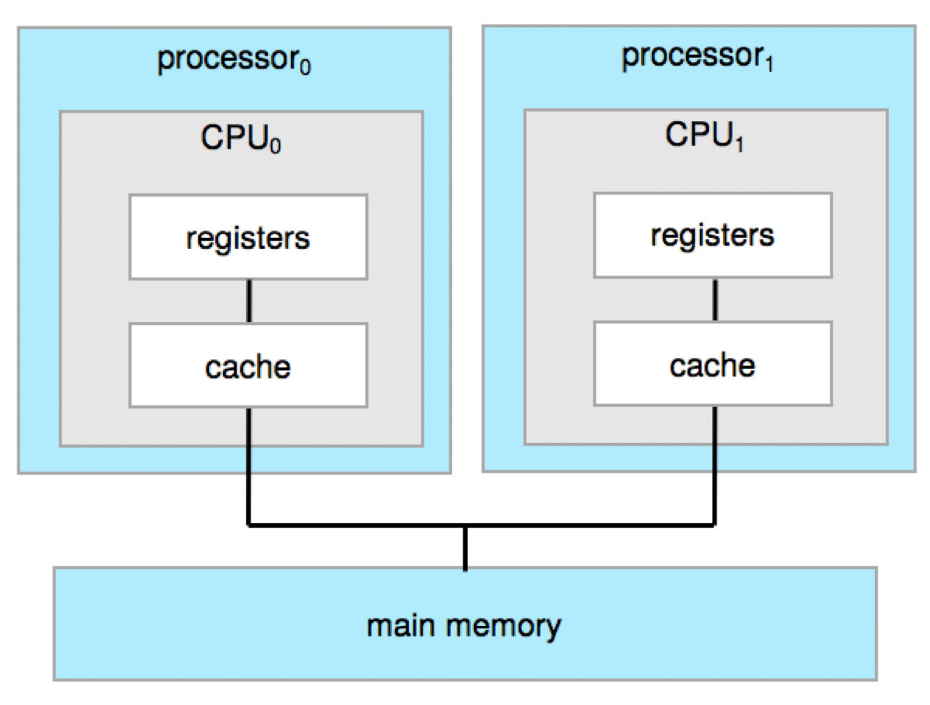
- **Network Operating System** provides features between systems across network
  - Communication scheme allows systems to exchange messages
  - Illusion of a single system

Multiprocessing Architectures

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

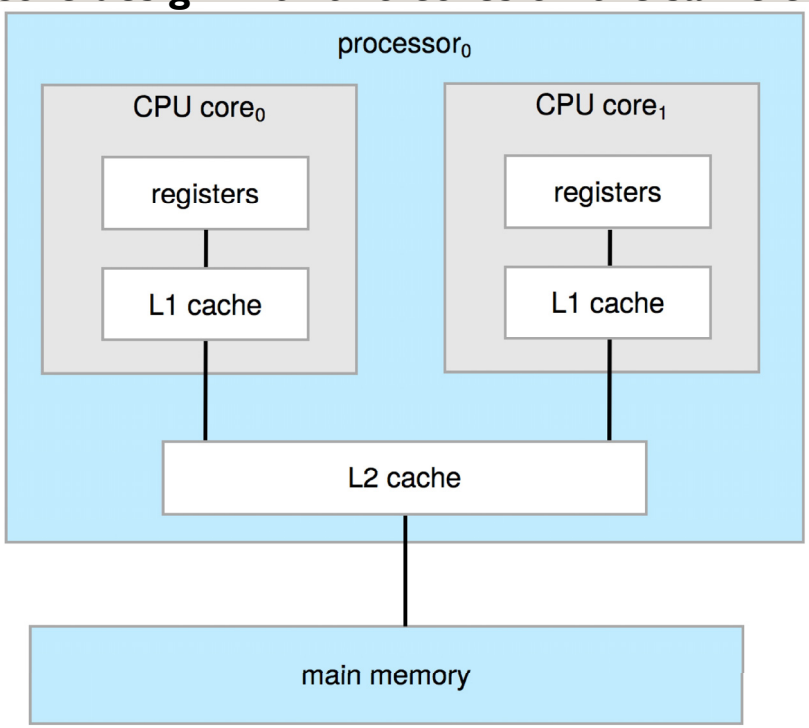
Symmetric Multiprocessing Architecture



Multiprocessor & Multicore

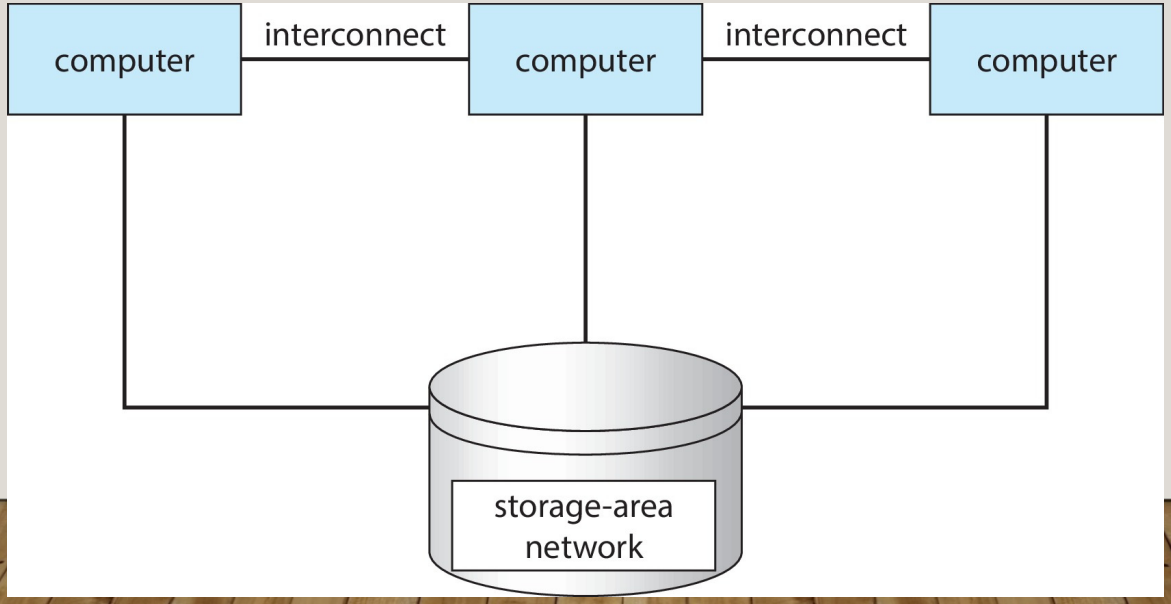
- Definition of multiprocessor has evolved over time and now includes multicore systems, in which multiple computing cores reside on a single chip.
- Multicore systems can be more efficient than multiple chips with single cores because on-chip communication is faster than between-chip communication

A dual-core design with two cores on the same chip.



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



Computing Environments

- Traditional: Stand-alone general-purpose machines
- Mobile: Handheld smartphones, tablets, etc. Leaders are Apple iOS and Google Android
- Client Server: Client requests resource/service and server provides that respective resource/service. A server can provide service to multiple clients at a time and here mainly communication happens through computer network.
- Peer-to-Peer: All nodes are considered peers. May each act as client, server or both. Examples include Napster and Gnutella, **Voice over IP** (**VoIP**) such as Skype
- Cloud computing: On demand availability of computer system resources like processing and storage are availed. Here computing is not done in individual technology or computer rather it is computed in cloud of computers where all required resources are provided by cloud vendor.
- Real-time Embedded: Real-time OS has well-defined fixed time constraints. Processing must be done within constraint.