Operating Systems

Prepared by Dr. Hesham Abou Elfetouh

Lecture 1

Operating System Introduction

Operating System

COURSE DESCRIPTION

- The course aims to explore the importance of the operating system and its function. The different techniques used by the operating system to achieve its goals as resource manager. This course introduce also, the fundamentals of modern operating system function, design, and implementation. Topics include concurrency, synchronization, processes, threads, scheduling, memory management, file systems, device management, and security.

COURSE OBJECTIVES

By the end of this course the student will be able to:

- Recognize the importance of the operating systems.
- Recognize how the applications interact with the operating system as the latter works as an intermediary program between the machine and the application.
- Know how the operating systems transport the application requests to the hardware.
- Understand how operating systems managing resources such as processors, memory and I/O.
- Realize the efficiency or the deficiency of the different techniques used by some operating systems.

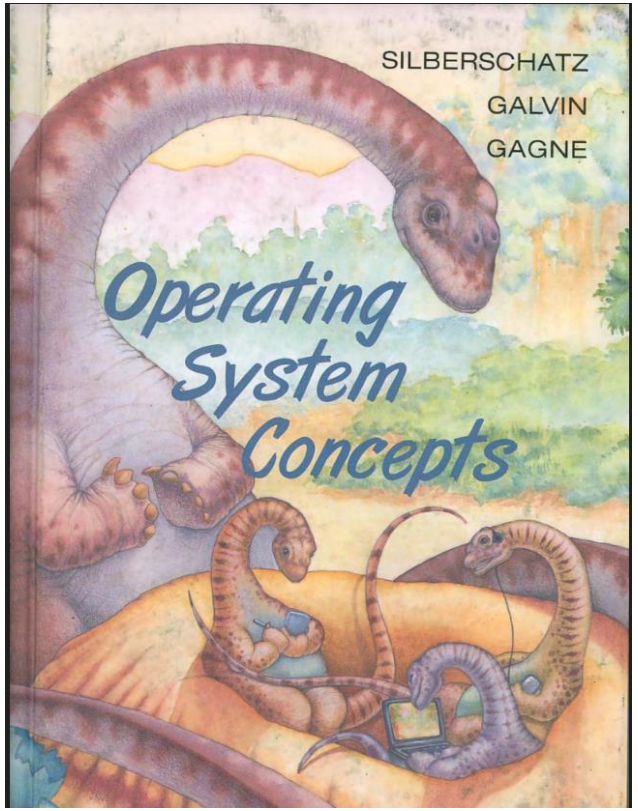
Operating System

COURSE SCHEDULING

- **Module 1: Operating System Introduction.**
- **Module 2: Processes and threads.**
- **Module 3: Concurrency.**
- **Module 4: Memory management.**
- **Module 5: Scheduling and Deadlock.**
- **Module 6: File management.**
- **Module 7: Operating System security.**
- **Module 8: Virtualization.**

TEXT BOOK

Silberschatz, Galvin, Gagne.
Operating System Concepts.
John Wiley & sons , inc



Operating System

Chapter 1: Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments
- Open-Source Operating Systems

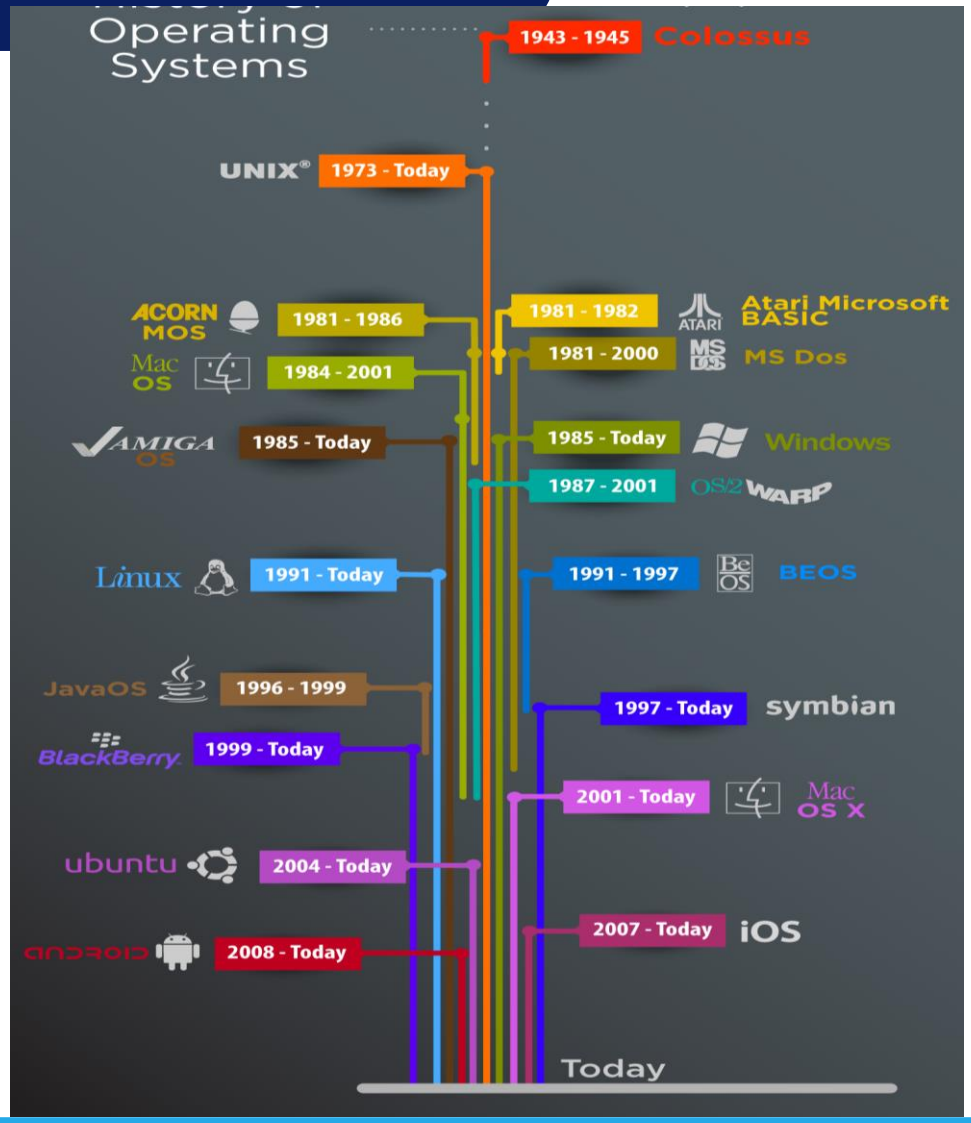
Computer Software

- Application software
  - This software is designed to solve a particular problem for users

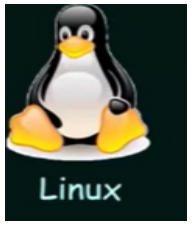
- System software
  - Provides a general working environment for developers and end users

History of Operating Systems

- First, an entire computer was dedicated to one program at a time, but this approach proved wasteful
- The first operating systems saved startup, loading, and shutdown time and made much better use of limited resources
- The first personal computers took a major step back, as they were dedicated to single users and effectively one program at a time
- Multitasking returned to the mainstream in the 1990s, and with it came all the lessons of the early shared computers



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
  - Provide environment in which applications software can execute as if they use the hardware alone.

Operating System

Major Operating system Functions:-

- Provide user interface
- Sharing hardware between users
- Sharing data between users
- Preventing users interference.
- Scheduling resources.
- Facilitating input/output
- Recovery from errors.
- Accounting for usage of resources.
- Handling networking communication.
- Organizing data in secure way and rabid access.

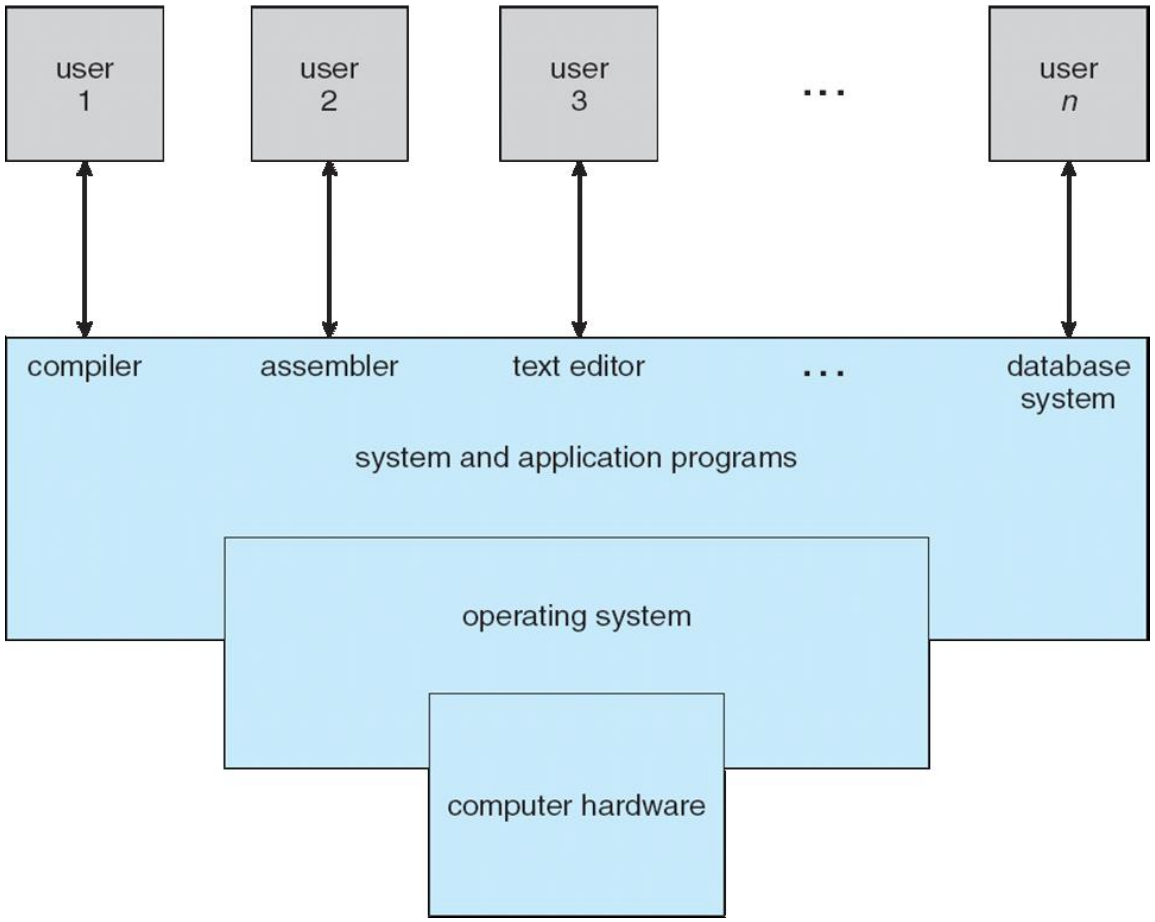
Computer System Structure

- 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

Computer System Structure

- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processors, compilers, web browsers, database systems, video games
- Users
  - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Computer Startup

- **How can an Operating system being loaded in Memory????**
- **Answer: we need a program that loads OS from Disk into memory.**

  This program is the bootstrap.
- **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

System Bootstrap or Boot-up

- BIOS (basic input output) is a Set of programs stored in ROM/EPROM contains procedures that handle the H/W devices
- BIOS bootstrap performs mainly 4 operations:

1. Power-on Self-Test (POST)
   - Tests to establish which devices are present
2. Initialization of H/W devices
   - So devices can operate without conflicts on ITQ (interrupt request) lines and I/O ports

System Bootstrap or Boot-up

3. Searches for OS to boot

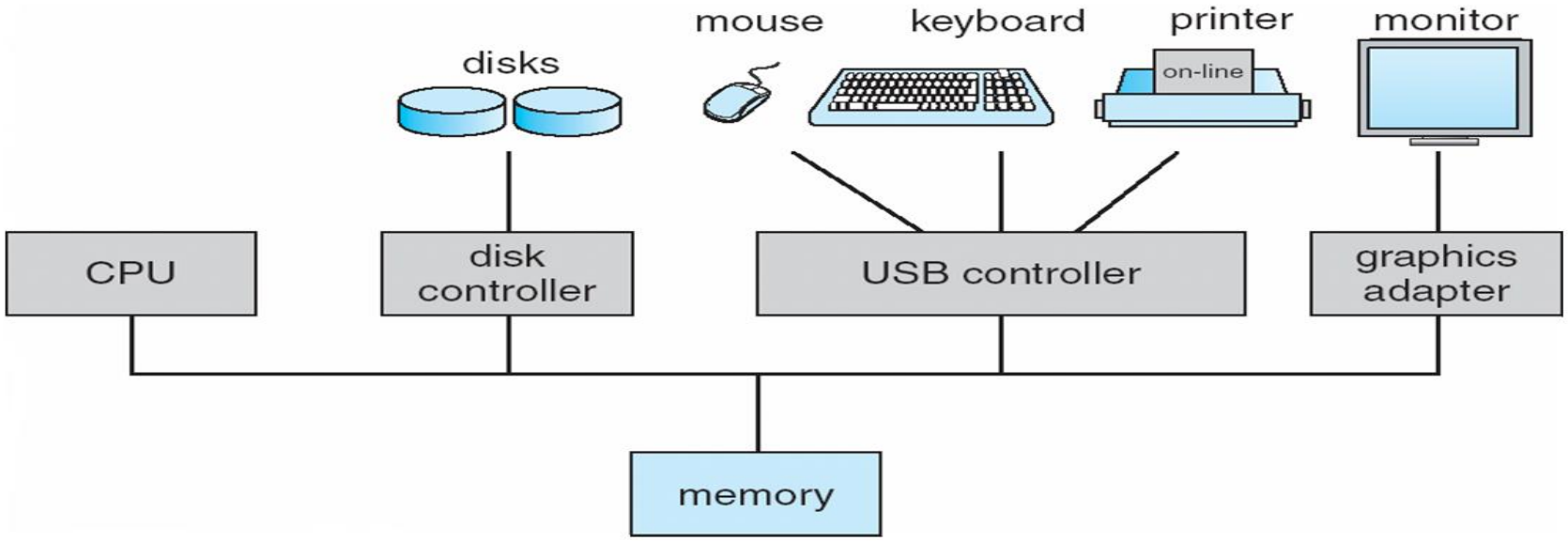
- Search order defined in BIOS.(BIOS Data part) E.g., floppy, then any hard disk, then any CD-ROM

4. If a device contains OS then sector 0 contains small Boot Loader or the first part of it

- Boot loader Copies the contents from the first sector of at specific address in memory and start executing it.

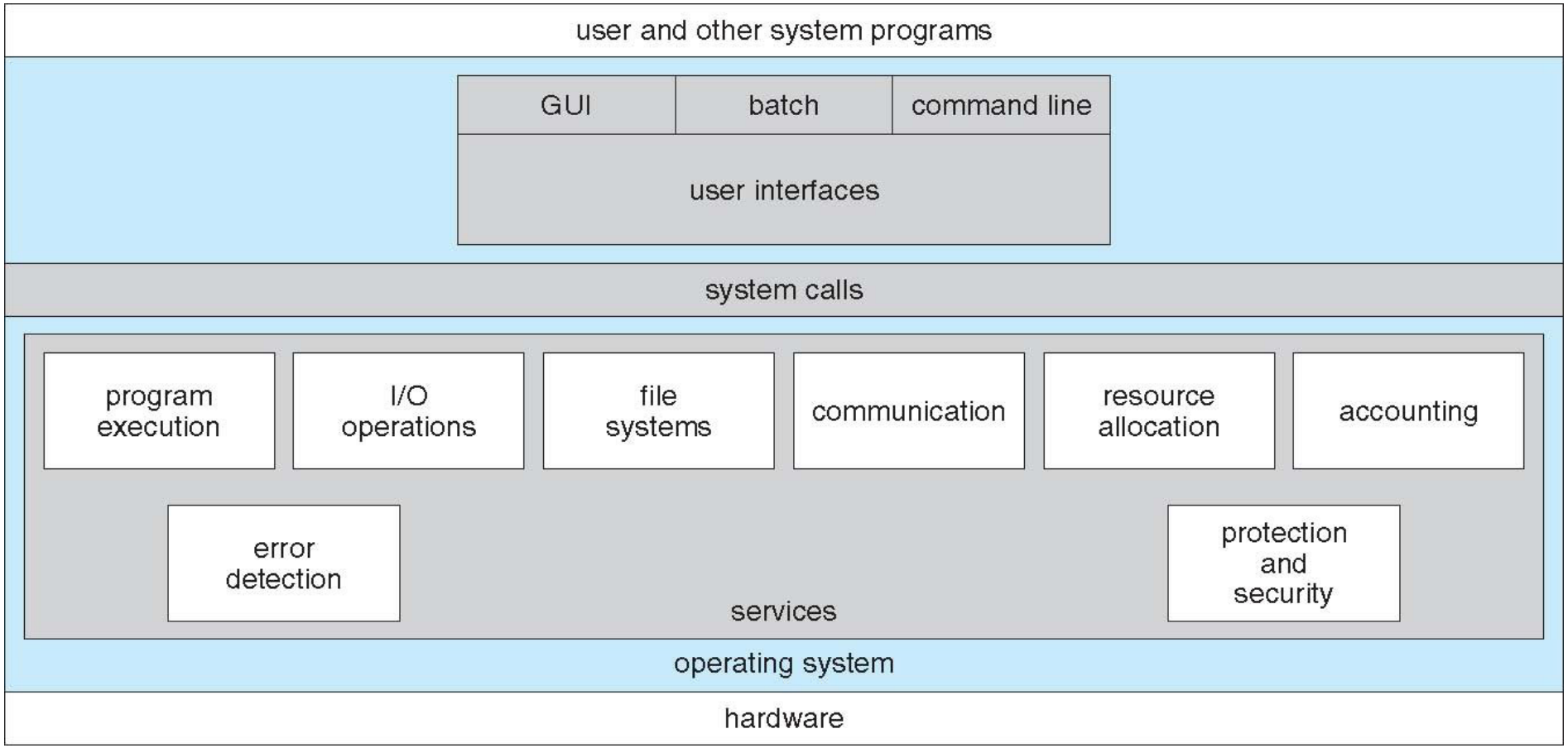
Computer System

- 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



Operating System

H/W Architecture and OS Services



Computer-System Operation

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

Some Important Terms

- **Interrupt**:-
  - The occurrence of event is usually signaled by Interrupt from hardware or software.
  - Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by the way of system bus.

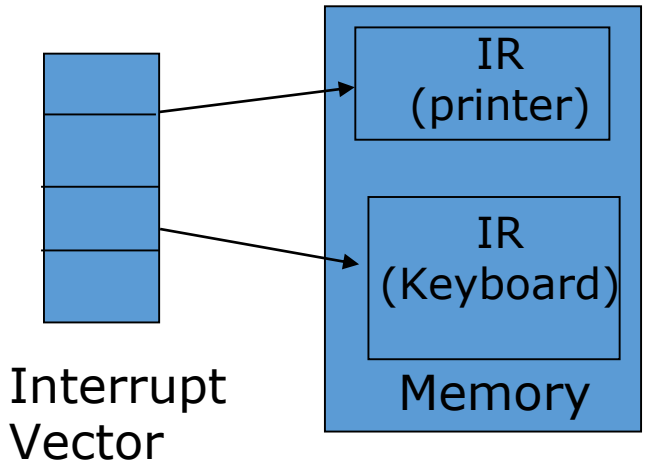
Interrupts

- Interrupt

  1. an external event raised by hardware: e.g. I/0

  2. an internal event trigger by software: e.g. **system calls** (means application needs a service from the OS. It calls a service). TRAP

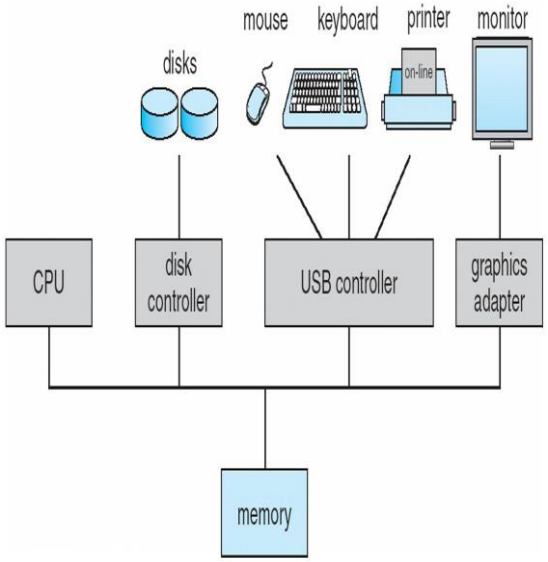
- Interrupt Handler
  - a subroutine in the operating system to process a particular type of interrupt
- Interrupt Vector
  - an array of starting addresses of interrupt handlers
  - The index for a particular interrupt to the array is provided by the hardware.



CPU/Device Interaction: Interrupts

- Interrupt transfers control to the interrupt service routine (ISR) generally, through the **interrupt vector**, which contains the addresses of all the service routines. ISR is identified by address in interrupt vector
- Interruption handling (overhead)
  - Interrupt architecture must save the address of the interrupted instruction
  - After servicing interrupt, CPU resumes execution at previously interrupted address (or “flow of control”)

OS Services: I/O Management



- Each device controller is in charge of a particular device type
- Each device controller has a local buffer/control regs.
- CPU moves data from/to main memory to/from local buffers
- Protocol: To get a byte of data from a device
  - CPU sets registers in controller with command to read e.g. character from keyboard
  - 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***
  - CPU moves data from controller to memory
- I/O devices and the CPU can execute concurrently

Interrupt Handling

- Separate segments of code determine what action should be taken for each type of interrupt
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- An operating system is **interrupt driven**

Some Important Terms

- **System Call:**
  - Software may trigger an interrupt by executing a special operation called System Call.
  - System call is the programmatic way in which a computer program request a service from the kernel of the operating system

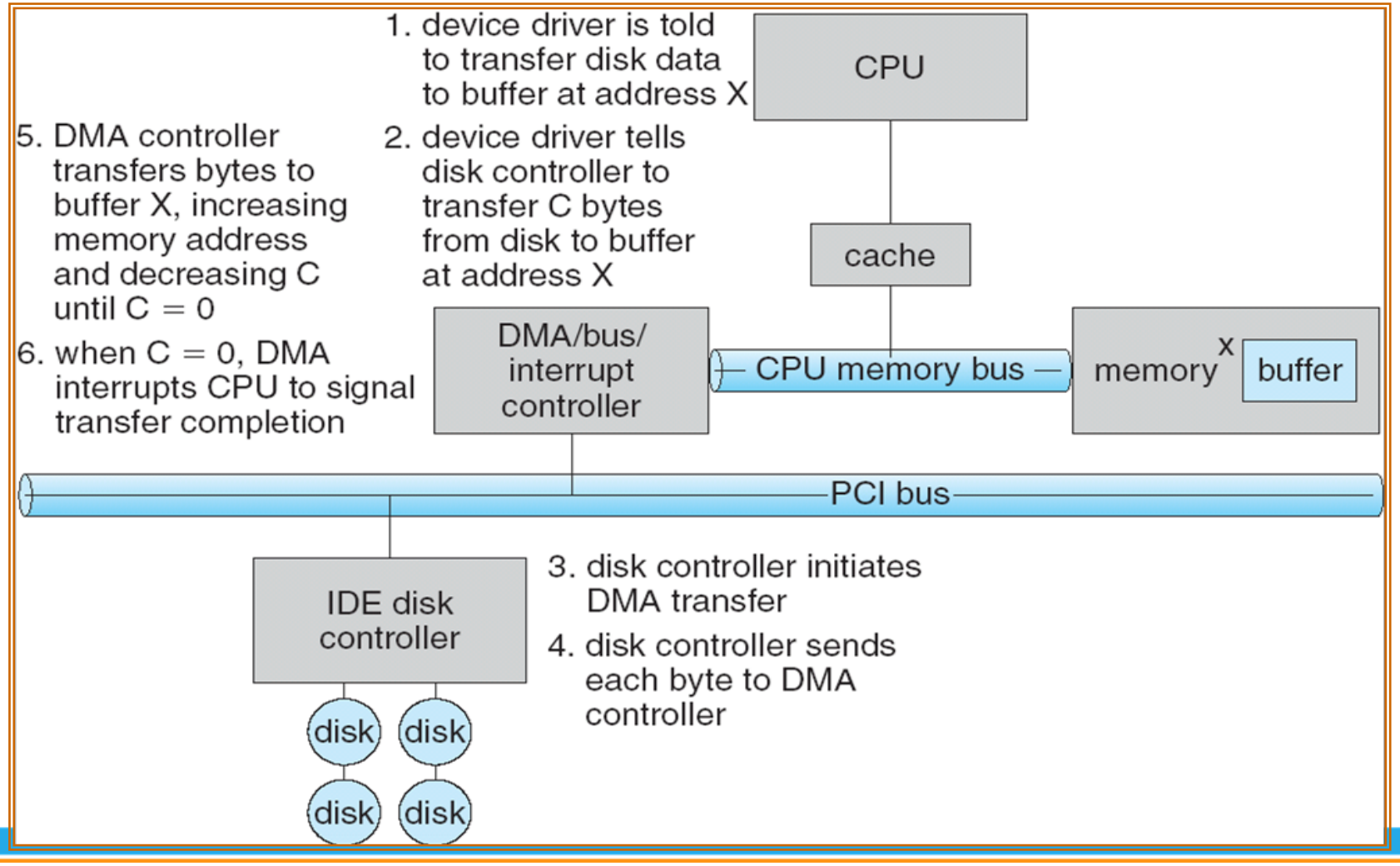
Some Important Terms

- When the CPU is interrupted, it stops what it is doing and immediately transfers execution to fixed location.
  - The fixed location usually contains the starting address where the service Routines of the interrupt is located.
- The Interrupt service Routines executes.
- On completion, the CPU resumes the interrupted computation.

I/O

- **I/O Techniques**
  - **Programmed I/O: processor interacts with I/O module directly**
    - **issues I/O commands to I/O module**
    - **interrogates status bits repeatedly (Pooling)**
    - **reads or writes: transfers the data between CPU and I/O module**
  - **Interrupt-Driven I/O: I/O module interrupts when I/O module is ready**
    - **issues I/O commands to I/O module**
    - **goes off to run other processes (doing something else)**
    - **get interrupted when I/O module is ready and reads or writes in interrupt handlers**
- **Direct Memory Access (DMA): A separate module called DMA module takes care of the interrupt from I/O and data transfer**
  - **Processor sends request to DMA (starting address, number of bytes, address of I/0 device, read or write)**
  - **DMA interacts with I/O module and completes the bulk transfer**
  - **DMA interrupts processor when done**

Six Step Process to Perform DMA Transfer



I/O Structure - I/O Device drive

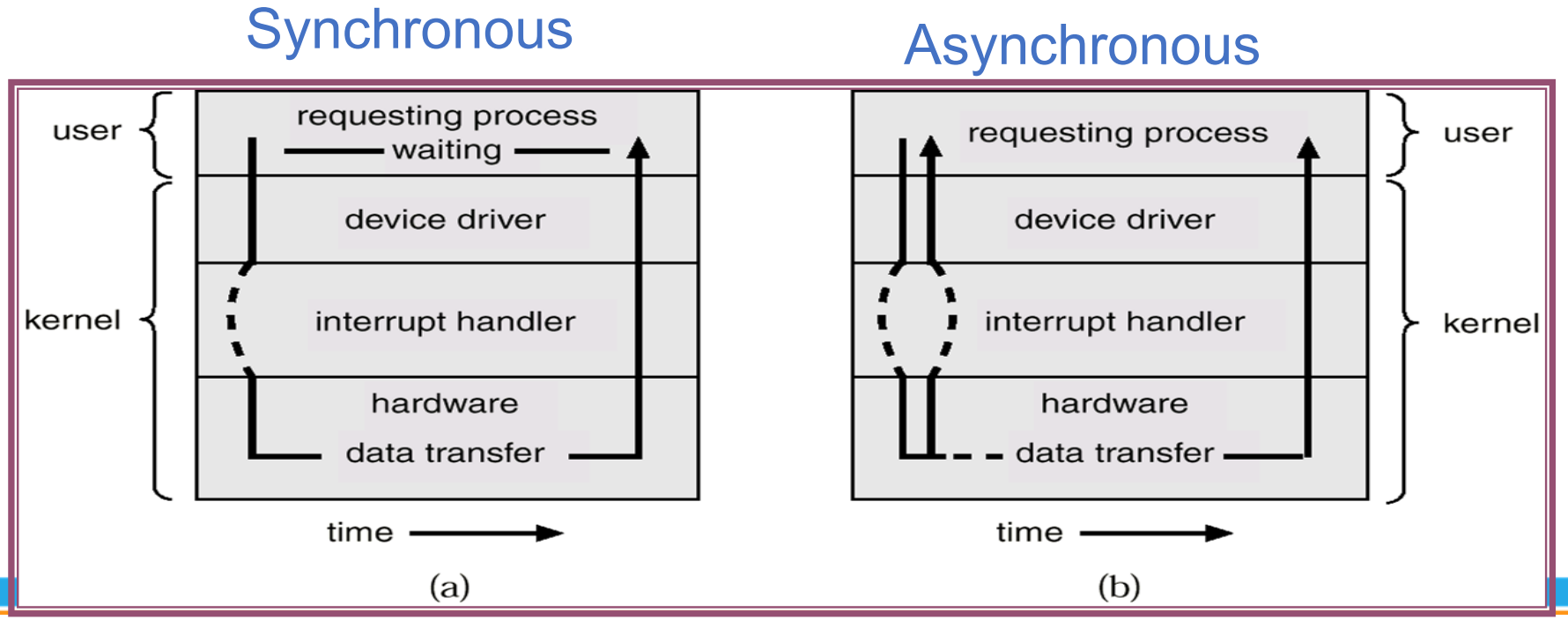
- Operating systems have a device driver for each I/O controller.
  - It understands the device controller (hide the peculiarities of the device)
  - presents a uniform interface of the device to the rest of the operating system.

I/O Structure - I/O Device driver

- **To get a service from a device driver application make**
  - **System call** – request to the operating system 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

I/O Structure- Sync. and Async. I/O

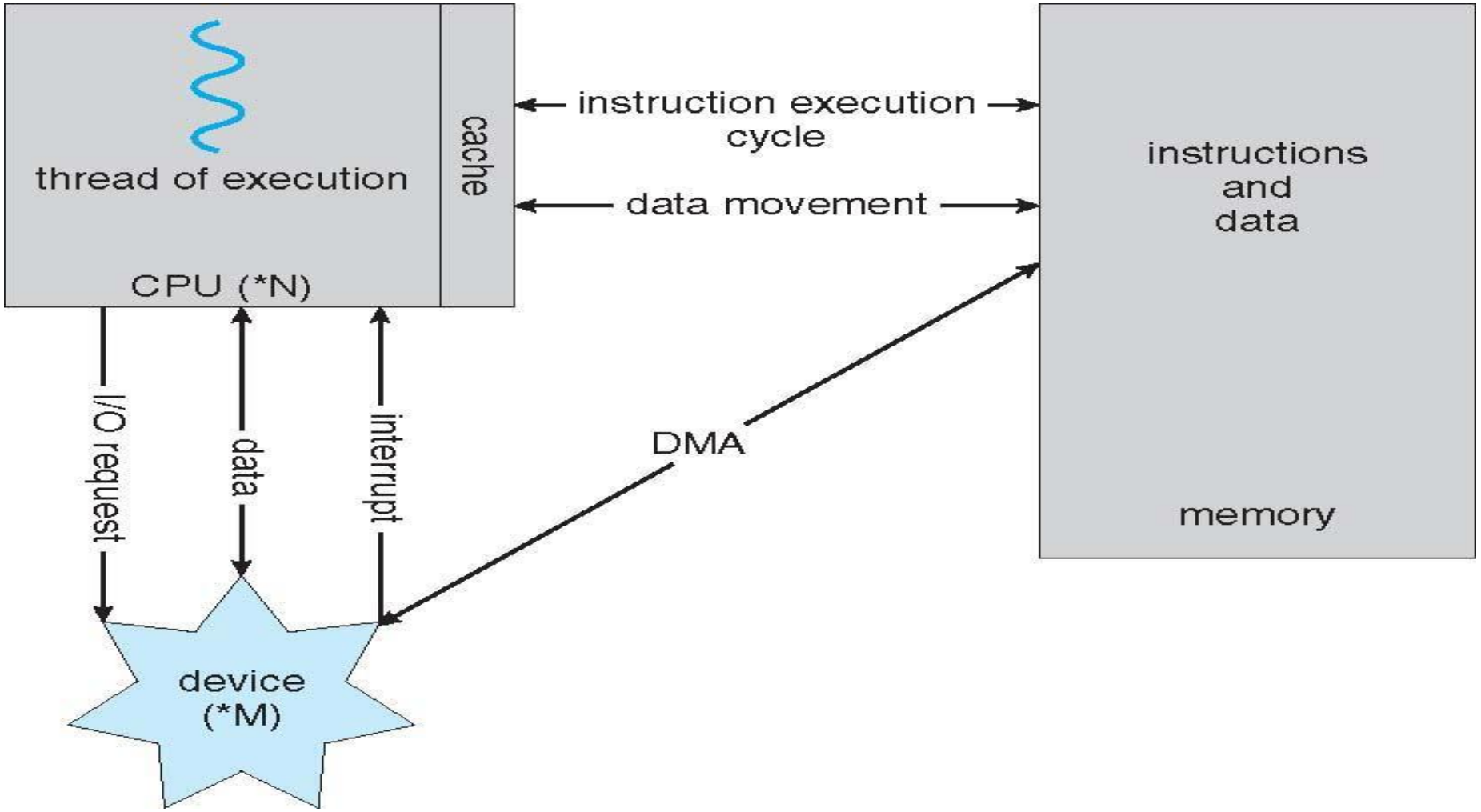
- Synchronous : After I/O starts, control returns to user program only upon I/O completion
- Asynchronous: After I/O starts, control returns to user program without waiting for I/O completion



How a Modern Computer Works

- CPU : unit where instructions are Executed
- Memory: storage where both program and data are stored
- I/O subsystem: unit to input and output data
- Instruction Execution Cycle do forever
  1. fetch next instruction
  2. execute the instruction
  3. check and process interrupt

How a Modern Computer Work



Direct Memory Access Structure

- Used for high-speed I/O (e.g. networking card 1G/sec) 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. Block size varies from device to device.

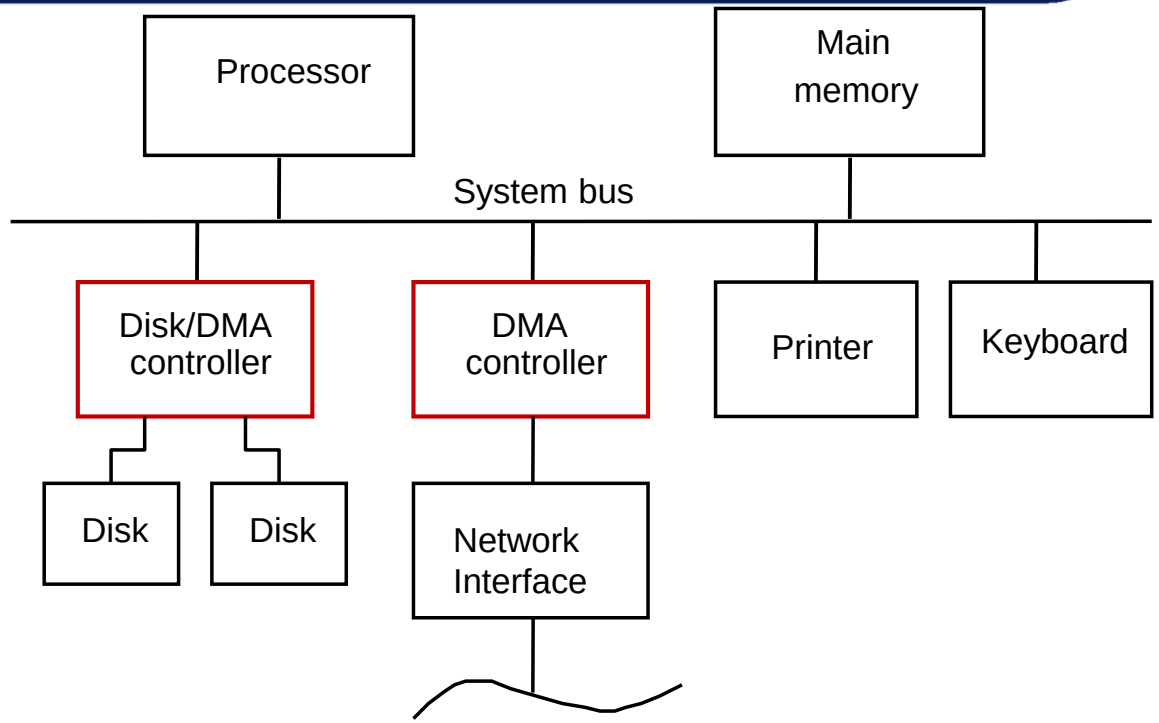
Direct Memory Access DMA

- Direct Memory Access (DMA):
  - A special control unit may be provided to transfer a block of data directly between an I/O device and the main memory, without continuous intervention by the processor.
- Control unit which performs these transfers is a part of the I/O device's interface circuit. This control unit is called as a DMA controller.
- DMA controller performs functions that would be normally carried out by the processor:
  - For each word, it provides the memory address and all the control signals.
  - To transfer a block of data, it increments the memory addresses and keeps track of the number of transfers.

DMA

- DMA controller can transfer a block of data from an external device to the processor, without any intervention from the processor.
  - However, the operation of the DMA controller must be under the control of a program executed by the processor. That is, the processor must initiate the DMA transfer.
- To initiate the DMA transfer, the processor informs the DMA controller of:
  - Starting address,
  - Number of words in the block.
  - Direction of transfer (I/O device to the memory, or memory to the I/O device).
- Once the DMA controller completes the DMA transfer, it informs the processor by raising an interrupt signal.

Direct Memory Access



- *DMA controller connects a high-speed network to the computer bus.*
- *Disk controller, which controls two disks also has DMA capability. It provides two DMA channels.*
- *It can perform two independent DMA operations, as if each disk has its own DMA controller. The registers to store the memory address, word count and status and control information are duplicated.*

Direct Memory Access (contd..)

- Processor and DMA controllers have to use the bus in an interwoven fashion to access the memory.
  - DMA devices are given higher priority than the processor to access the bus.
  - Among different DMA devices, high priority is given to high-speed peripherals such as a disk or a graphics display device.
- Processor originates most memory access cycles on the bus.
  - DMA controller can be said to “steal” memory access cycles from the bus. This interweaving technique is called as “cycle stealing”.
- An alternate approach is the provide a DMA controller an exclusive capability to initiate transfers on the bus, and hence exclusive access to the main memory. This is known as the block or burst mode.

Operating Systems Groups

Three groups

Batch.

Runs batch jobs, no online users or tasks

Time-sharing

Supports online tasks and online users

Most of the OS you know is in this category

Real-Time **(RTOS)**

Real Time Operating Systems are used in computers that controls critical systems such as: control machinery, scientific instruments, Satellites, nucellar reactors and industrial systems.

Storage Structure

- Main memory: is the only large storage media that the CPU can access directly
- Secondary storage: is an extension of main memory that provides large nonvolatile storage capacity

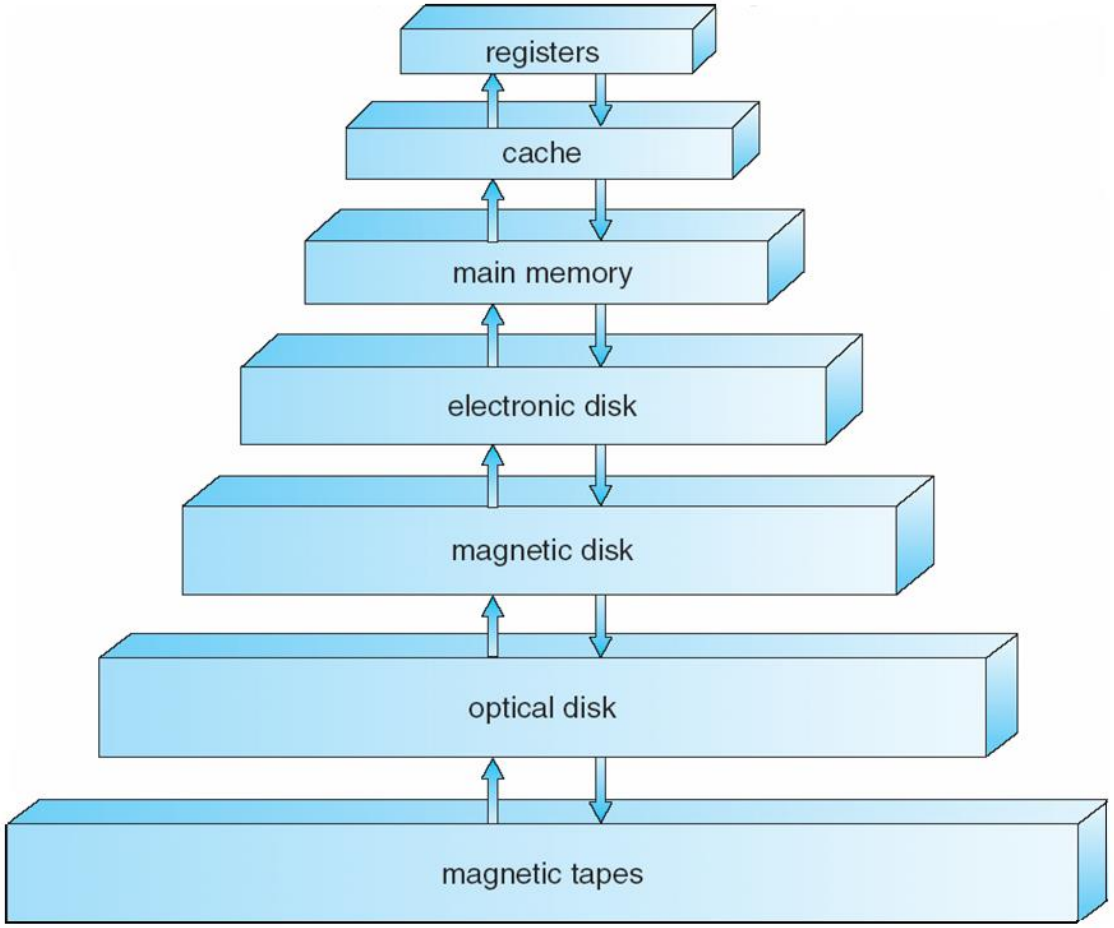
Storage Structure

- A magnetic disk is a storage device that uses a magnetization process to write, rewrite and access data.
- Electronic Disks: is data storage device functionally similar to a hard disk drive but using flash memory
- Cache memory: is a small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data.

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
    - (Volatile: Looses its contents when power is removed)
    - (Non Volatile: Retains its contents even when power is removed)

Storage Device Hierarchy



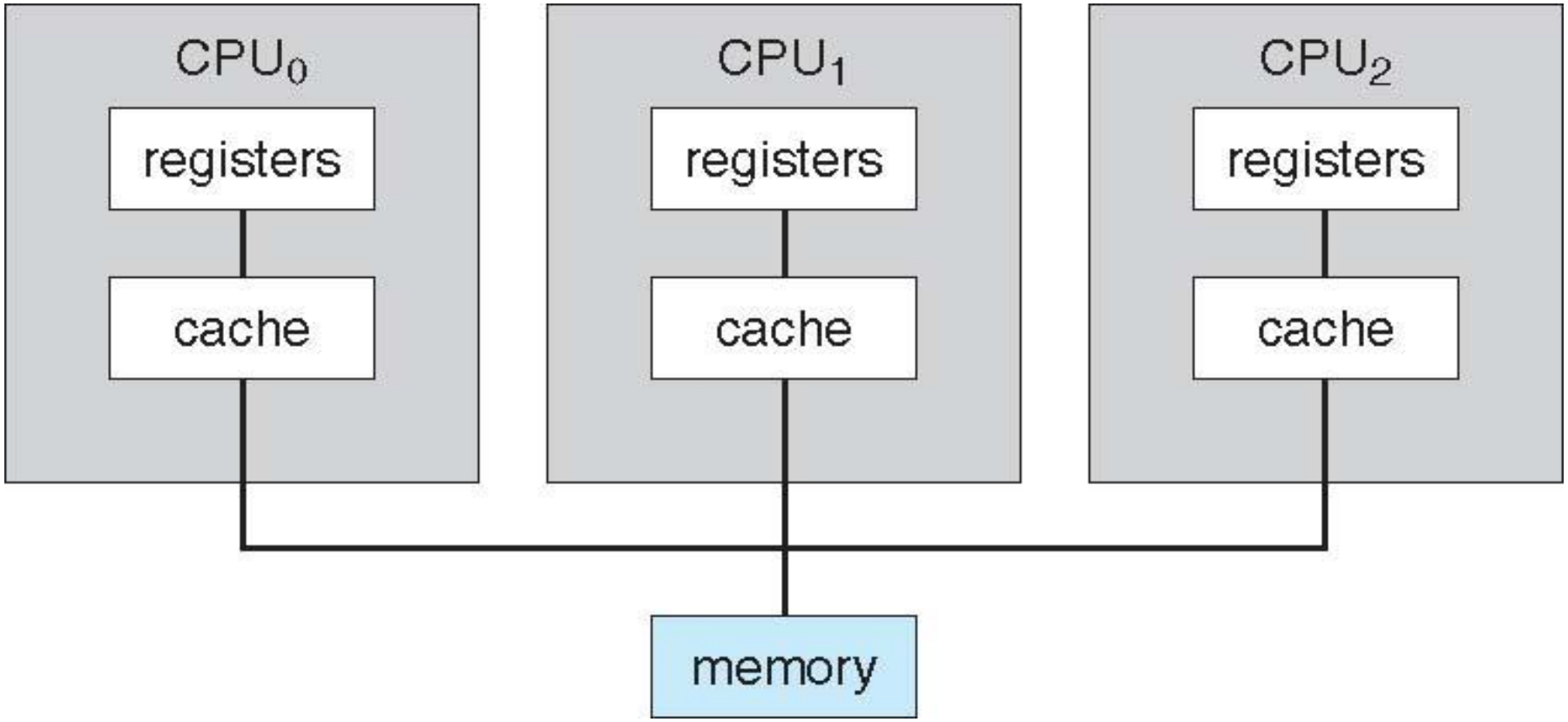
Caching

- 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

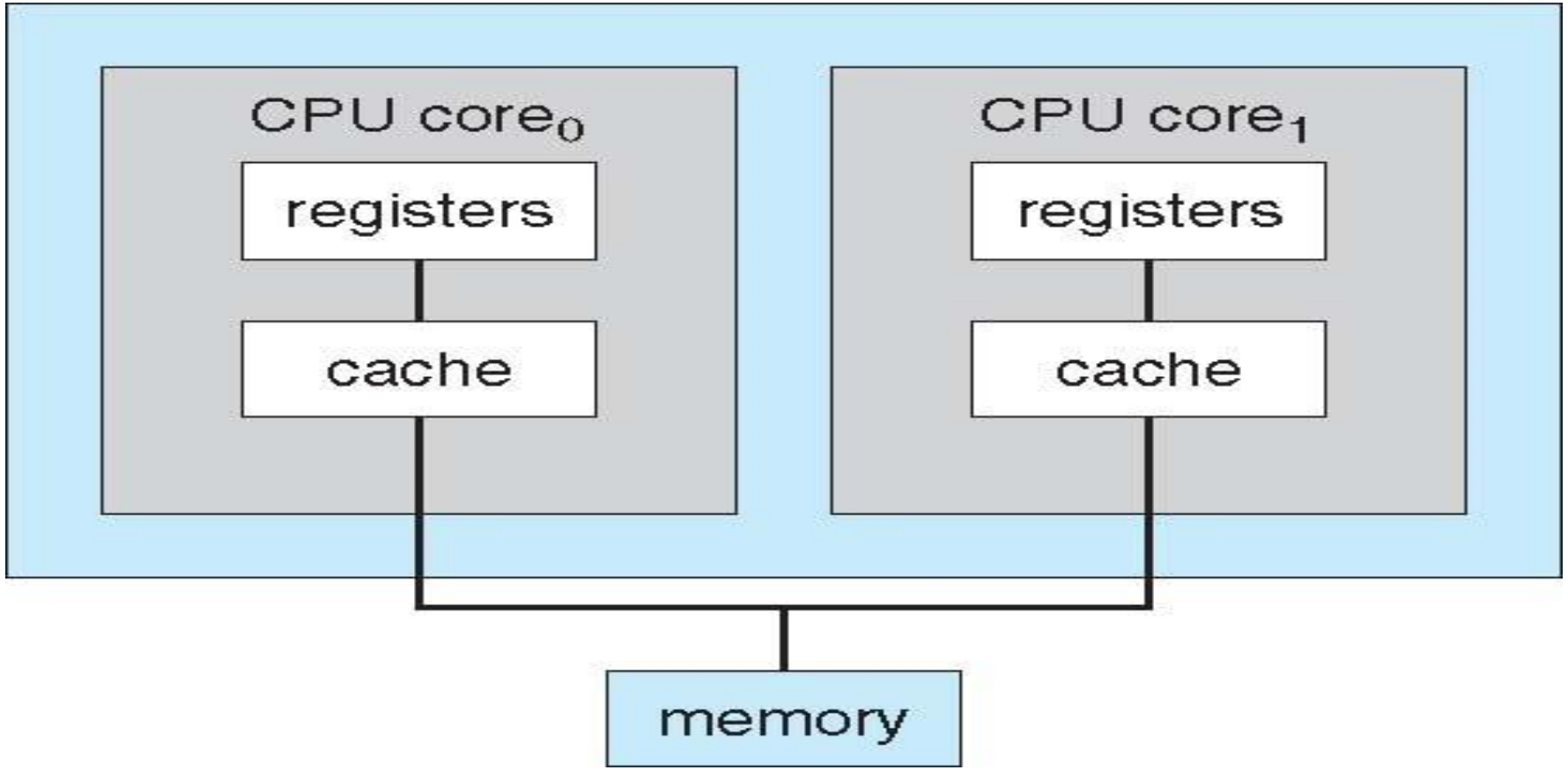
Computer-System Architecture

- Most systems use a single general /special-purpose processor
- Multiprocessors systems growing in use & importance
  - Advantages
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability
  - Two types
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing

Symmetric Multiprocessing Architecture



A Dual-Core Design



Basic of Operating Systems (I/O Structure)

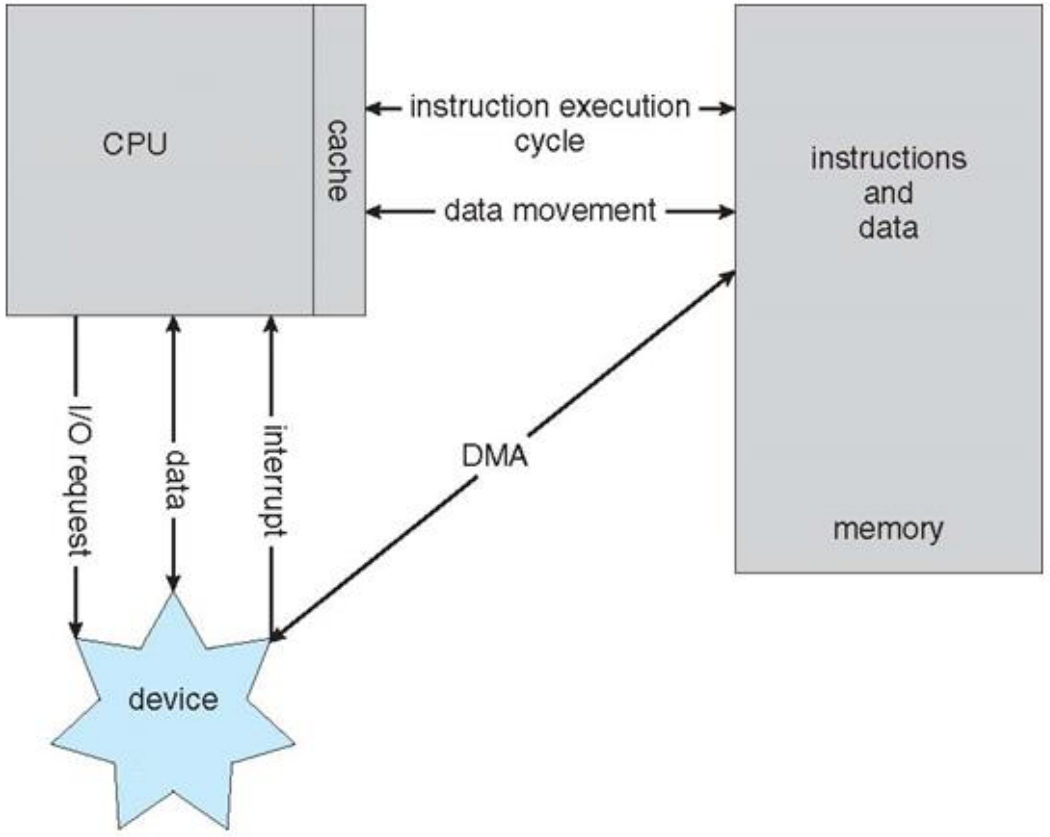
- Storage is only one of many types of I/O devices within a computer.
- A large portion of operating system code is dedicated to managing I/O, both because of its importance to the reliability and performance of system and because of the varying nature of the devices.
- A general-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.

Basic of Operating Systems (I/O Structure)

- Typically, Operating systems have a device driver for each device controller.

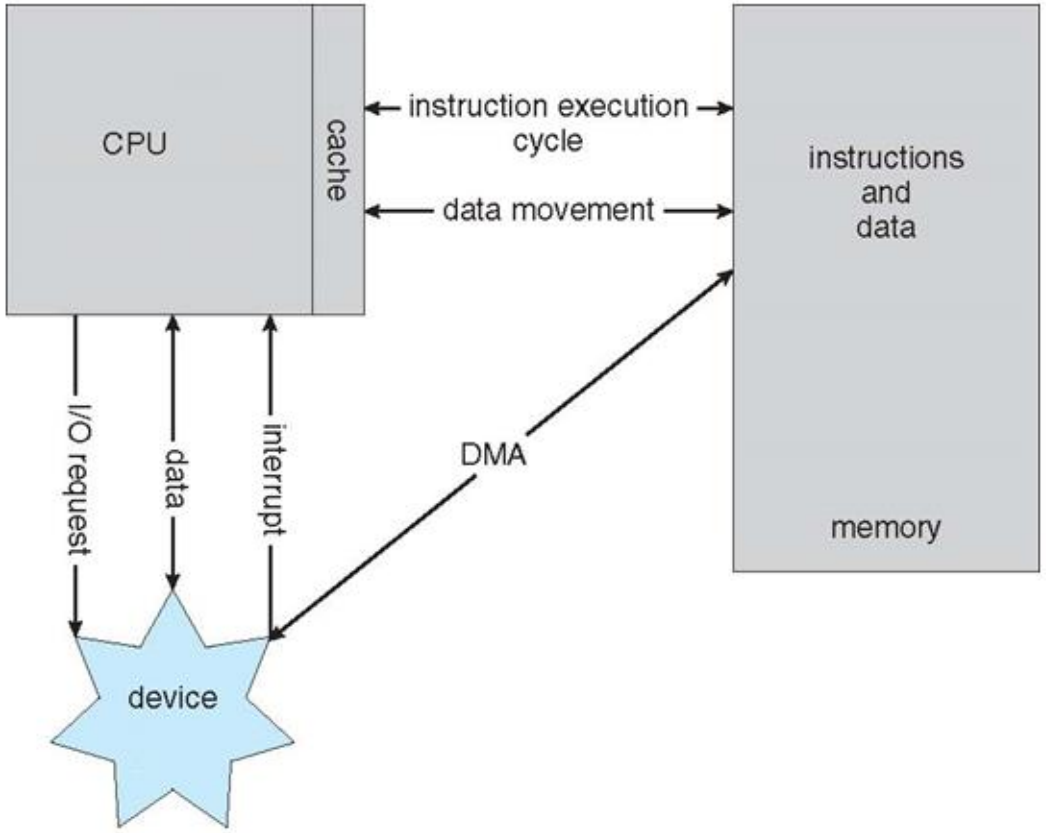
- This device driver understands the device controller and presents a uniform interface to the device to the rest of the operating system.

Working of an I/O operation



- To start in I/O operation, the device driver loads the appropriate registers within the device controller which in turn determine what action to take.
- The controller starts the transfer of data from the device to the local buffer.

Working of an I/O operation



- Once data transfer is completed, the device controller informs device driver via interrupt that it has finished its operation and the control is returned to the operating system

Basic of Operating Systems (I/O Structure)

- The form presented in the last slide is called interrupt-driven I/O. it is suitable when moving small amounts of data but can produce high overhead when used for bulk of data.
- To solve this problem, Direct Memory Access **(DMA)** is used.
- Device controller with **DMA** capability, 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

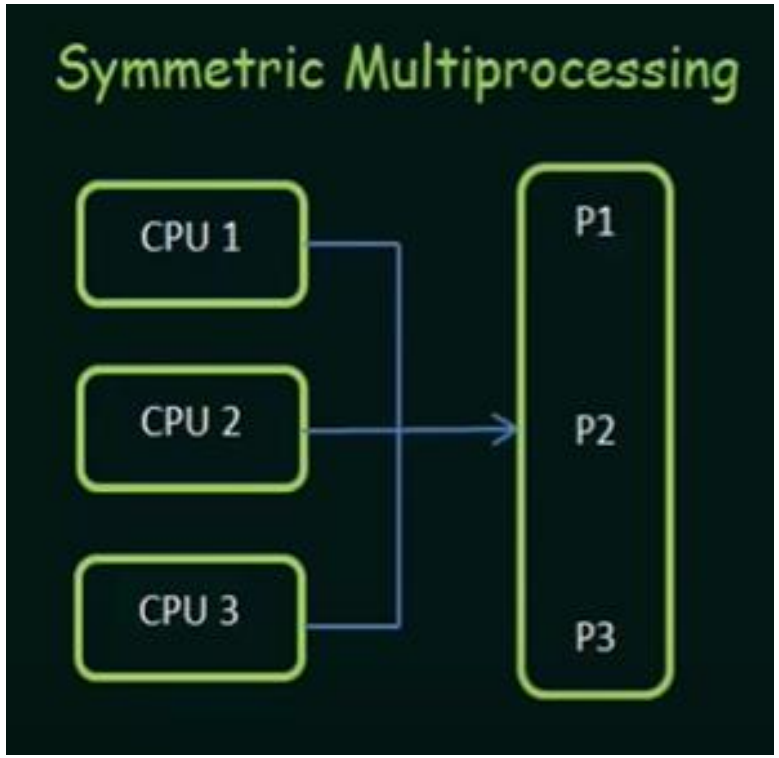
- Types of Computer Systems are based on the number of General-Purpose Processors:
  - Single Processor Systems
  - Multi Processor Systems
  - Clustered Systems

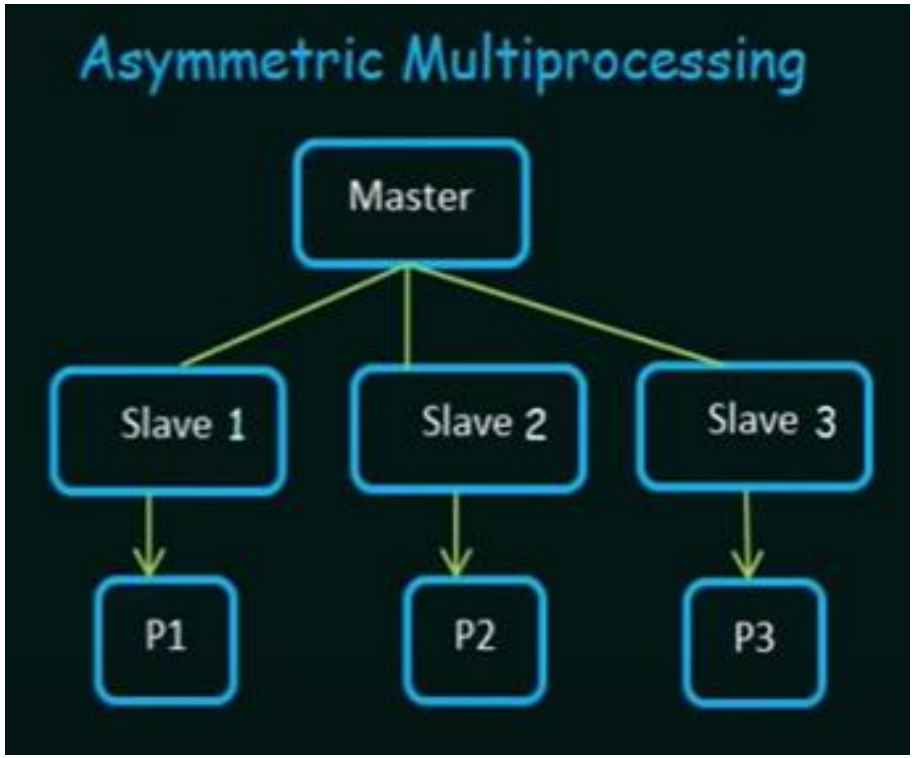
- Single Processor Systems: one main CPU capable of executing a general-purpose instruction set.
- Other special purpose processors (device controllers) are also present to perform specific task.

Computer-System Architecture

- 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
    2. Symmetric Multiprocessing

Computer-System Architecture





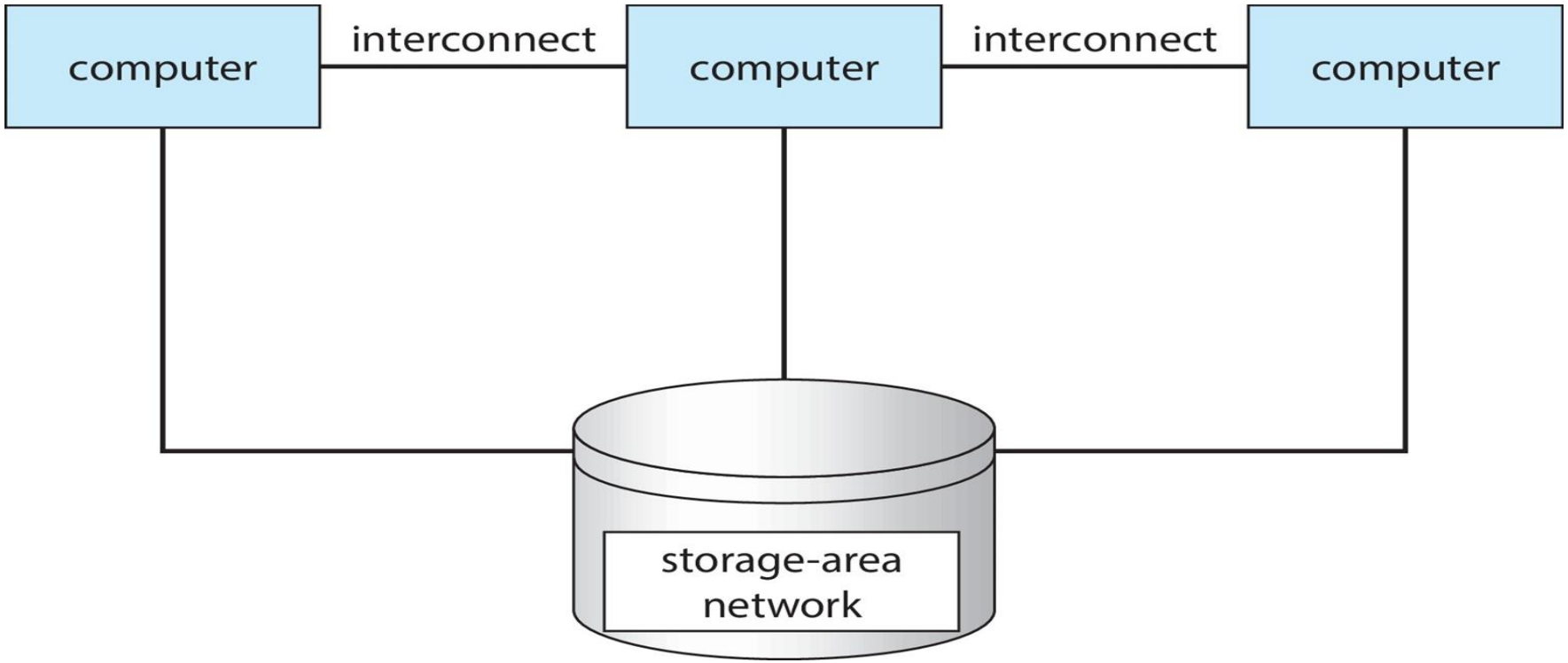
Computer-System Architecture

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

Computer-System Architecture



Operating Systems Structure

Operating Systems vary greatly in their makeup internally:

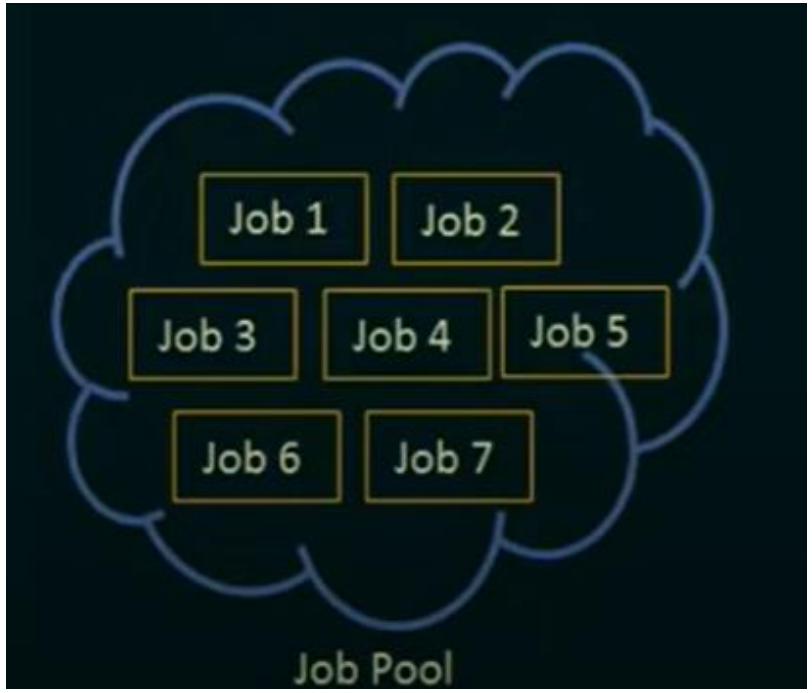
- Multiprogramming
- Time Sharing (Multitasking)

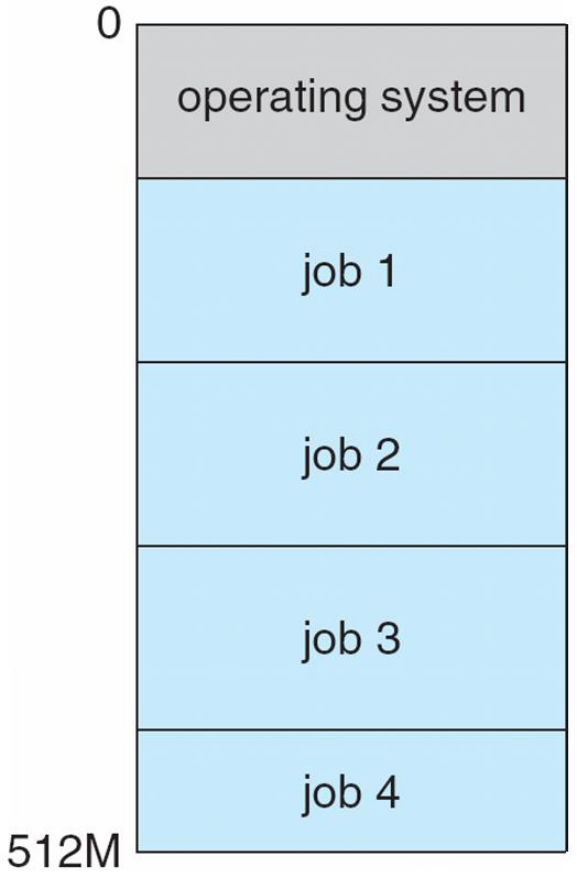
Operating Systems Structure

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

Operating Systems Structure





Memory layout of multiprogramming systems

Operating Systems Structure

Timesharing (multitasking) is a logical extension of multiprogramming 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**

Operating Systems Structure

- 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

Operating Systems Operations

- Operating systems are considered as an Interrupt driven mostly by hardware
- A *trap* is a software-generated interrupt caused either by an error or a user request
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service

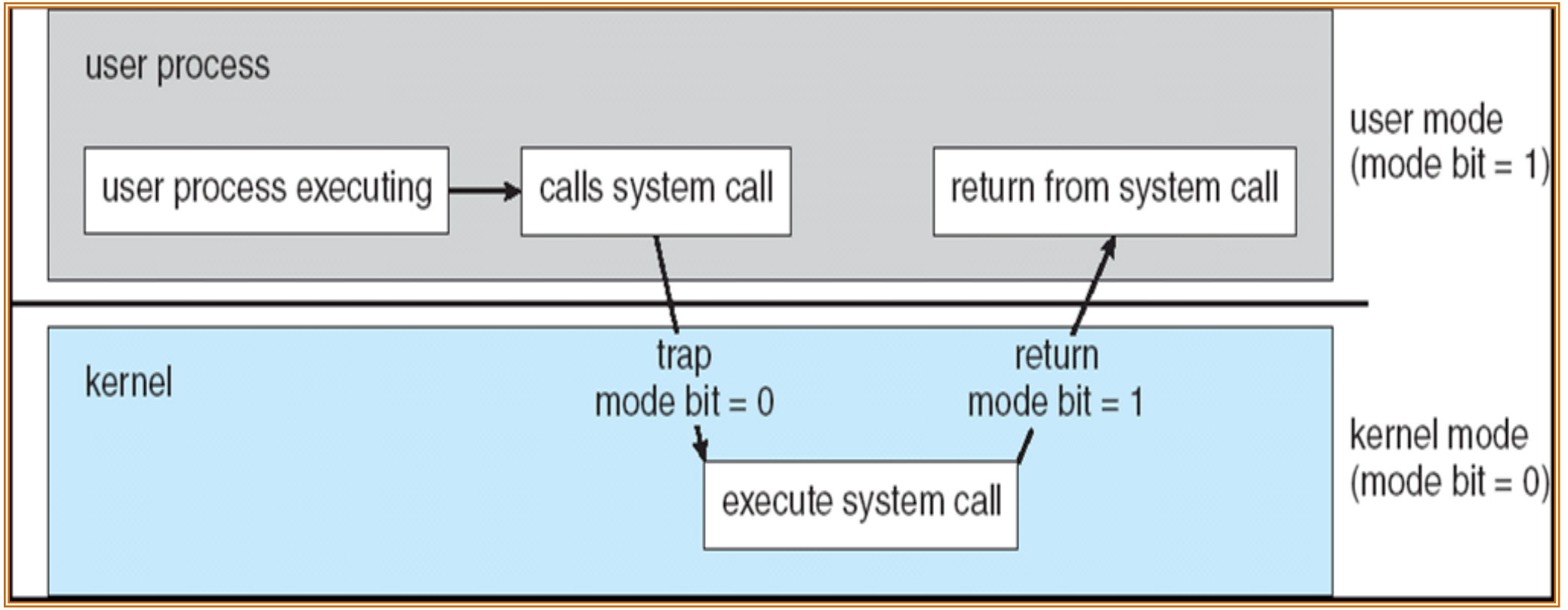
Operating Systems Operations

- **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

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Transition from User to Kernel Mode



Transition from User to Kernel Mode

- User's code issues a system call
- Mode is changed to kernel mode
- OS first checks that everything (e.g., parameter values) is in order and legal
- Then, OS executes system call which may contain multiple privileged instructions
- mode is set again to user mode

End of Lecture