Introduction to OS

Dr. Ahmed Awais

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

CHAPTER 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.

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**

Microsoft
Apple
Mac OS
Microsoft Windows Server
ubuntu
Linux
MS DOS

Desktop Operating Systems
Windows Versions
macOS
macOS Monterey
macOS Ventura
Linux Distributions
Ubuntu
Fedora
Debian
Arch Linux
Mint

Mobile Operating Systems
Android
iOS
Windows Phone (no longer supported)

Server Operating Systems
Windows Server
Linux Server Editions
CentOS
Red Hat Enterprise Linux (RHEL)
SUSE Linux Enterprise Server (SLES)
Unix
AIX
HP-UX
Solaris

Embedded Operating Systems
FreeRTOS
VxWorks
QNX

Other Notable Operating Systems
Chrome OS
OpenBSD
ReactOS (an open-source Windows alternative)

Real-Time Operating Systems (RTOS)
1. RTEMS - Real-Time Executive for Multiprocessor Systems (RTEMS)
2. Integrity
3. Nucleus

RTOS	Example Applications
RTEMS	Used in space applications, such as the **NASA's Mars Rover.**
Integrity	Commonly used in automotive systems for **safety-critical tasks, like airbag control systems.**
Nucleus	Employed in medical devices, such as **patient monitoring systems and infusion pumps.**

Type of Operating System	Examples	Key Characteristics
Desktop Operating Systems	Windows 10, Windows 11, macOS Ventura, Ubuntu, Fedora	Designed for personal computers; support a wide range of applications; user-friendly interfaces.
Mobile Operating Systems	Android, iOS, Windows Phone	Optimized for mobile devices; focus on touch interfaces; often have app stores for software distribution.
Server Operating Systems	Windows Server, CentOS, Red Hat Enterprise Linux (RHEL), SUSE Linux, AIX	Tailored for server hardware; support multiple users and processes; focus on stability and performance.
Embedded Operating Systems	FreeRTOS, VxWorks, QNX	Designed for specific hardware; often real-time systems; minimal resources and tailored for specific tasks.
Real-Time Operating Systems (RTOS)	RTEMS, Integrity, Nucleus	Provide deterministic response times; used in critical applications where timing is crucial.
Other Notable Operating Systems	Chrome OS, OpenBSD, ReactOS	Diverse functions; Chrome OS is web-centric, OpenBSD focuses on security, ReactOS aims to be Windows-compatible.

Which OS Is Mostly Used By Software Engineers?

Software engineers use different operating systems based on what they like, the type of work they do, and the industry they work in. The three most prevalent operating systems among software engineers include:

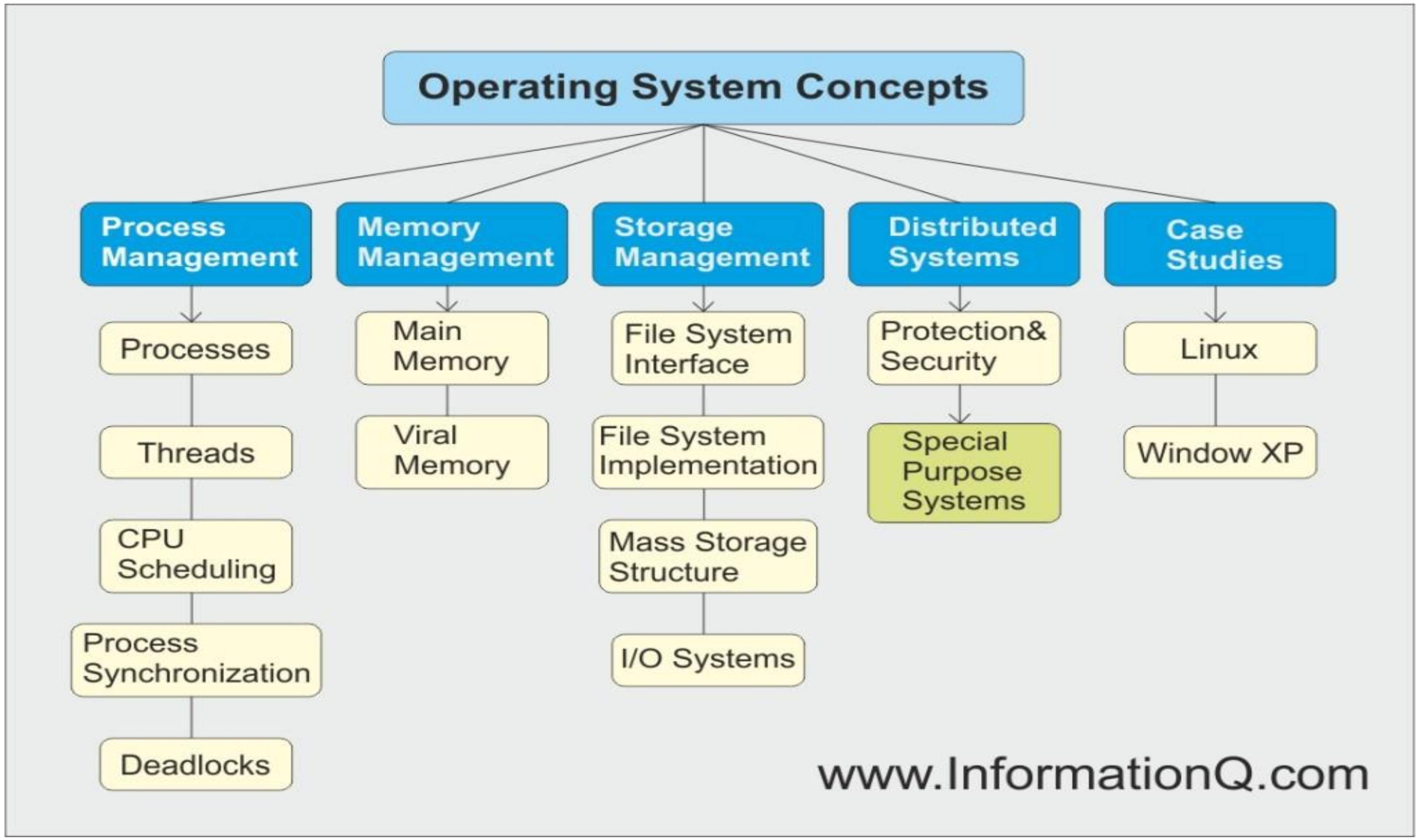
- **Windows:** Windows is widely used in corporate environments and by developers working on Microsoft technologies such as .NET framework, C#, and Visual Studio. Many software engineers find Windows convenient for its broad compatibility with commercial software and hardware.
- **Linux:** Linux is highly favored among software engineers, especially those involved in systems programming, web development, and open-source projects. Its robust command-line interface (CLI), vast array of development tools, and customization options make it popular for tasks ranging from server management to embedded systems development.
- **macOS:** macOS is preferred by many software engineers working in creative industries such as app development, UX/UI design, and multimedia production. It combines a Unix-based environment with a user-friendly interface and is known for its seamless integration with iOS development tools (Xcode).

The choice of operating system often depends on factors such as the developer's specific role, the technologies they work with, and personal preferences for development workflows and tools. Many developers also use multiple operating systems through virtualization or dual-boot setups to leverage the strengths of each platform for different aspects of their work.

Why we use an Operating System?

- Because we need to interact with machines

- Have you seen one before?


Operating System Concepts
Process Management
Memory Management
Storage Management
Distributed Systems
Case Studies
Processes
Threads
CPU Scheduling
Process Synchronization
Deadlocks
Main Memory
Viral Memory
File System Interface
File System Implementation
Mass Storage Structure
I/O Systems
Protection& Security
Special Purpose Systems
Linux
Window XP
www.InformationQ.com

Computer System Structure

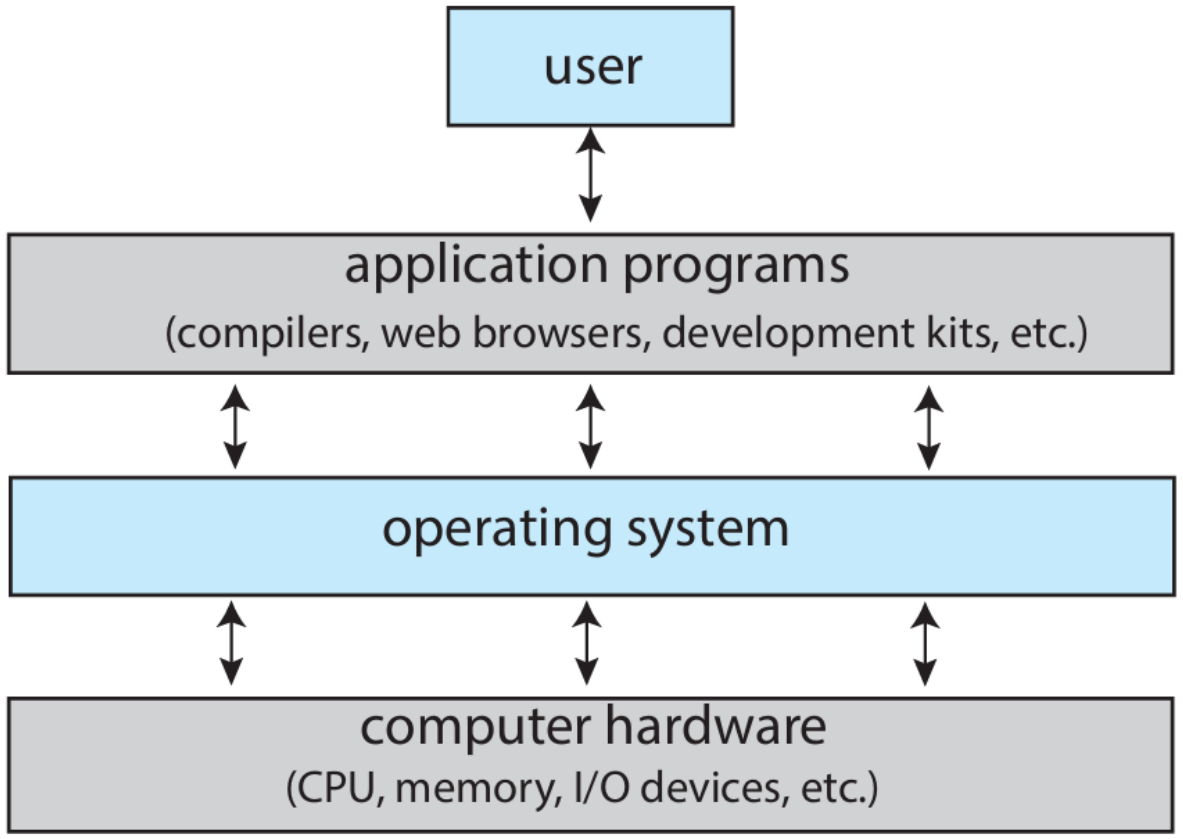

user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

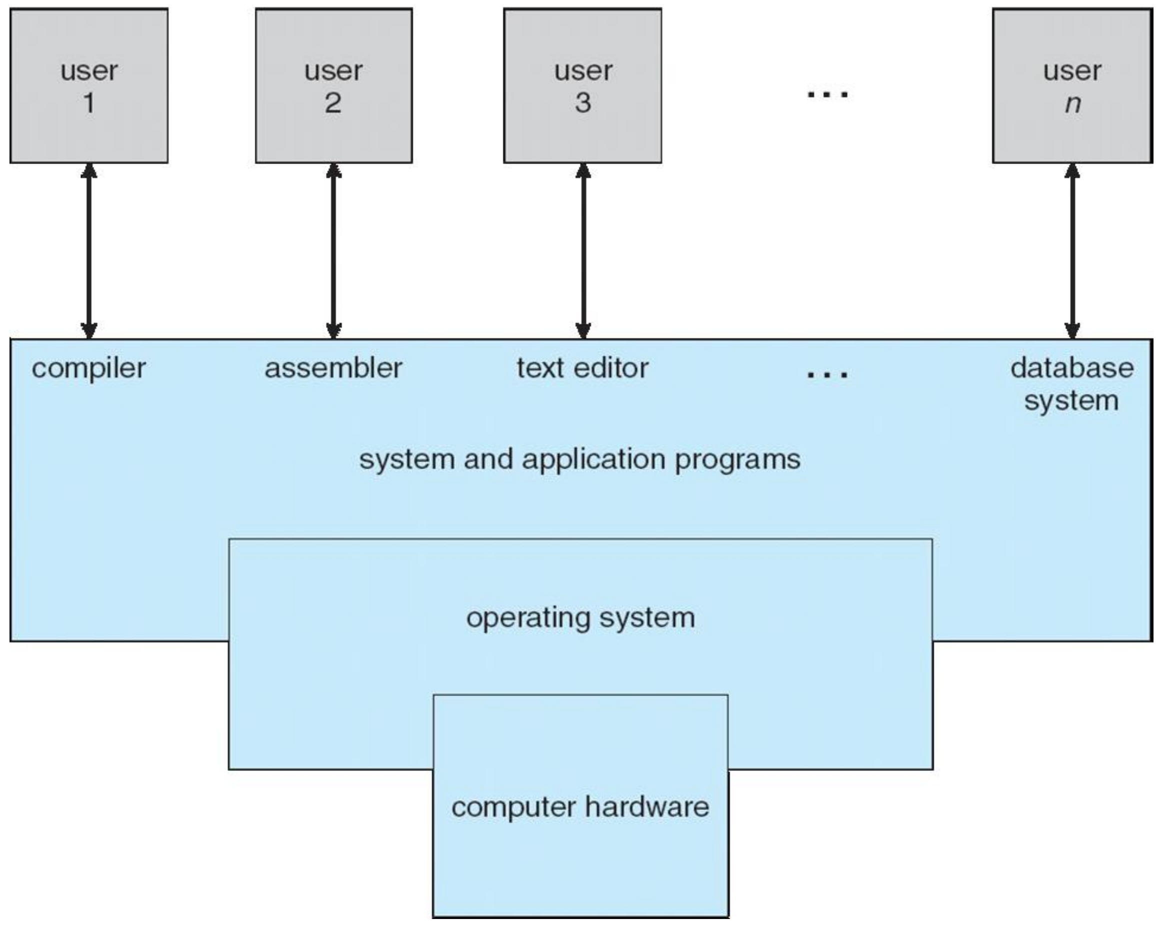
Figure 1.1 Abstract view of the components of a computer system.

Four Components of a Computer System

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

Four Components of a Computer System



What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use**
    - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

Defining OS

- Nowadays, Computers are present within toasters, cars, ships, space craft, homes, and businesses. They are the basis for game machines, cable TV tuners, and industrial control systems.
- Initially OS were fixed-purpose systems for military uses, such as code breaking and trajectory plotting, and governmental uses, such as census calculation. Those early computers evolved in to general-purpose, multifunction mainframes, and that’s when operating systems were born.

Moore’s Law about OS

- Inthe1960s, Moore’s Law predicted that the number of **transistors** on an **integrated circuit** would double every **18 months**, and that **prediction has held true.**
- Computers gained in functionality and shrank in size, leading to a vast number of uses and a vast number and variety of operating systems. (See Appendix A for more details on the history of operating systems.)

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

Operating System Definition (Cont.)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program.

Computer Startup

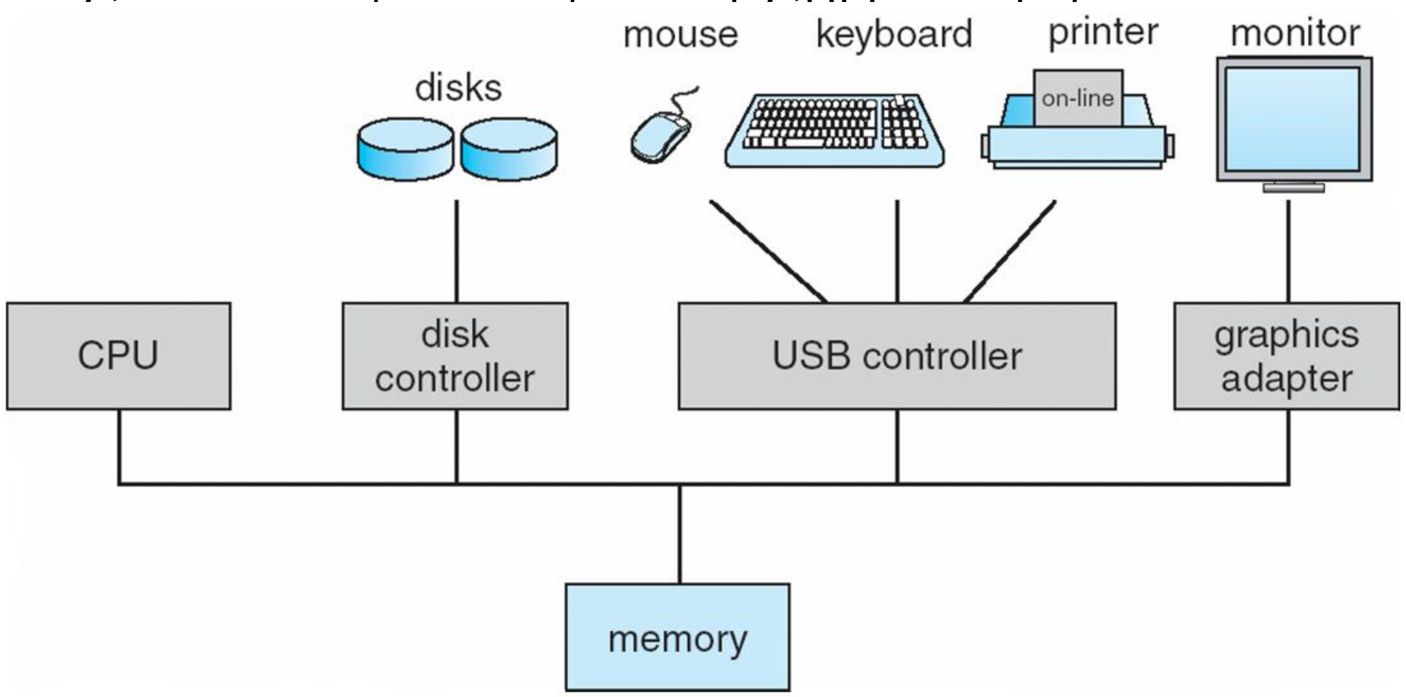
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in **ROM () or EPROM (),** generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Type	**Full Name**	**Characteristics**	**Reusability**	**Typical Use Cases**
ROM	Read-Only Memory	**Non-volatile memory** that is programmed during manufacturing; data cannot be changed.	Not reusable	Firmware, embedded systems
PROM	Programmable Read-Only Memory	**Non-volatile memory** that can be programmed once by the user; once programmed, data cannot be changed.	Not reusable	Custom firmware applications
EPROM	Erasable Programmable Read-Only Memory	**Non-volatile memory** that can be erased using UV light and reprogrammed multiple times.	Reusable (after erasing)	Development and testing of firmware

Type	Full Name	Characteristics	Typical Use Cases
DRAM	Dynamic Random Access Memory	Needs to be refreshed thousands of times per second; slower than SRAM; used for main memory.	Primary memory in computers and laptops
SRAM	Static Random Access Memory	Faster and more reliable than DRAM; does not need to be refreshed; more expensive.	Cache memory in CPUs, routers, and other high-speed applications
SDRAM	Synchronous Dynamic RAM	Synchronized with the system clock for higher performance; used in modern computers.	Main memory in PCs, laptops, and servers
DDR SDRAM	Double Data Rate Synchronous DRAM	Transfers data on both the rising and falling edges of the clock signal; faster than SDRAM.	Main memory in modern computers and gaming systems
LPDDR	Low Power DDR	Optimized for low power consumption; used in mobile devices.	Smartphones, tablets, and other portable devices

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - [illegible]



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

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
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- A *trap* is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

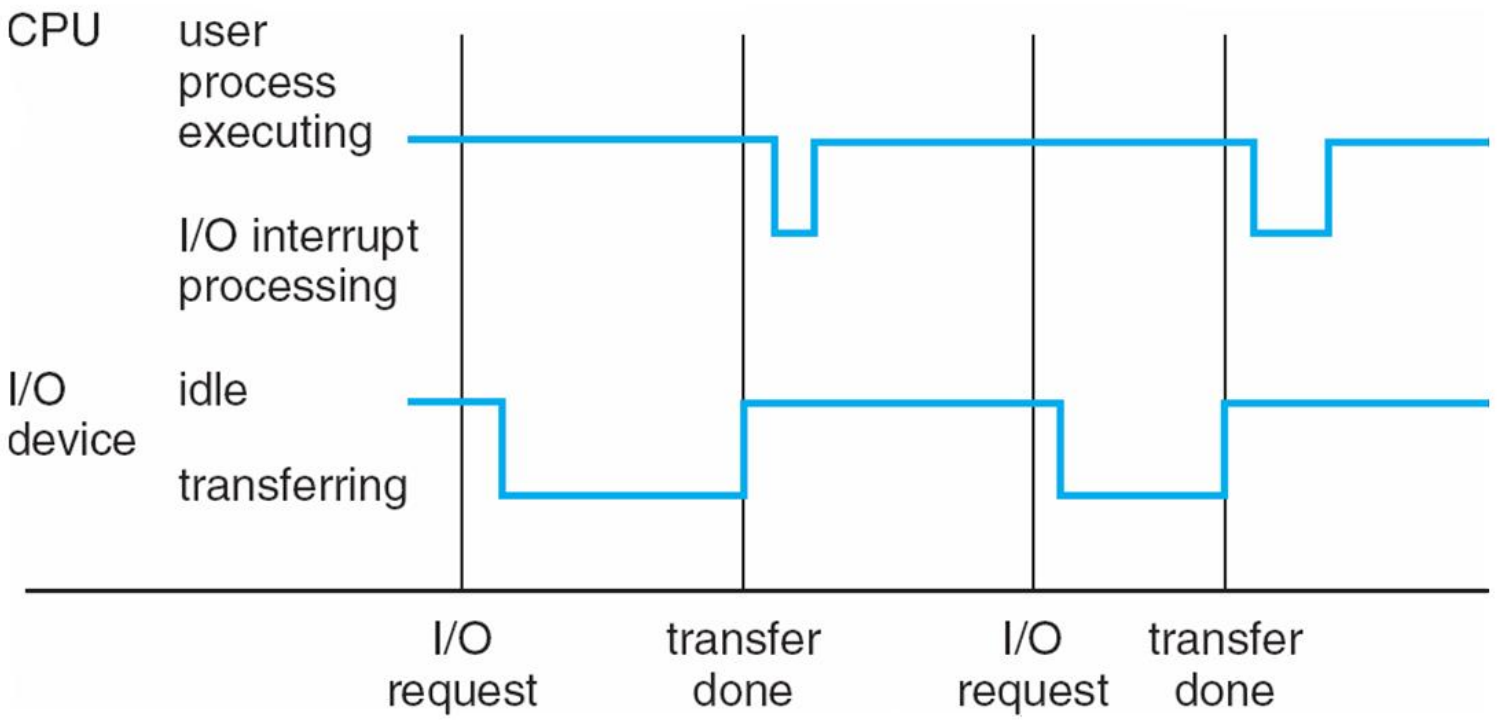
Interrupt Handling

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

- Determines which type of interrupt has occurred:
    - **polling**
    - **vectored** interrupt system

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

Interrupt Timeline



I/O Structure

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

- After I/O starts, control returns to user program without waiting for I/O completion
  - **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
  - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt

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

Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer

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

Storage-Device Hierarchy

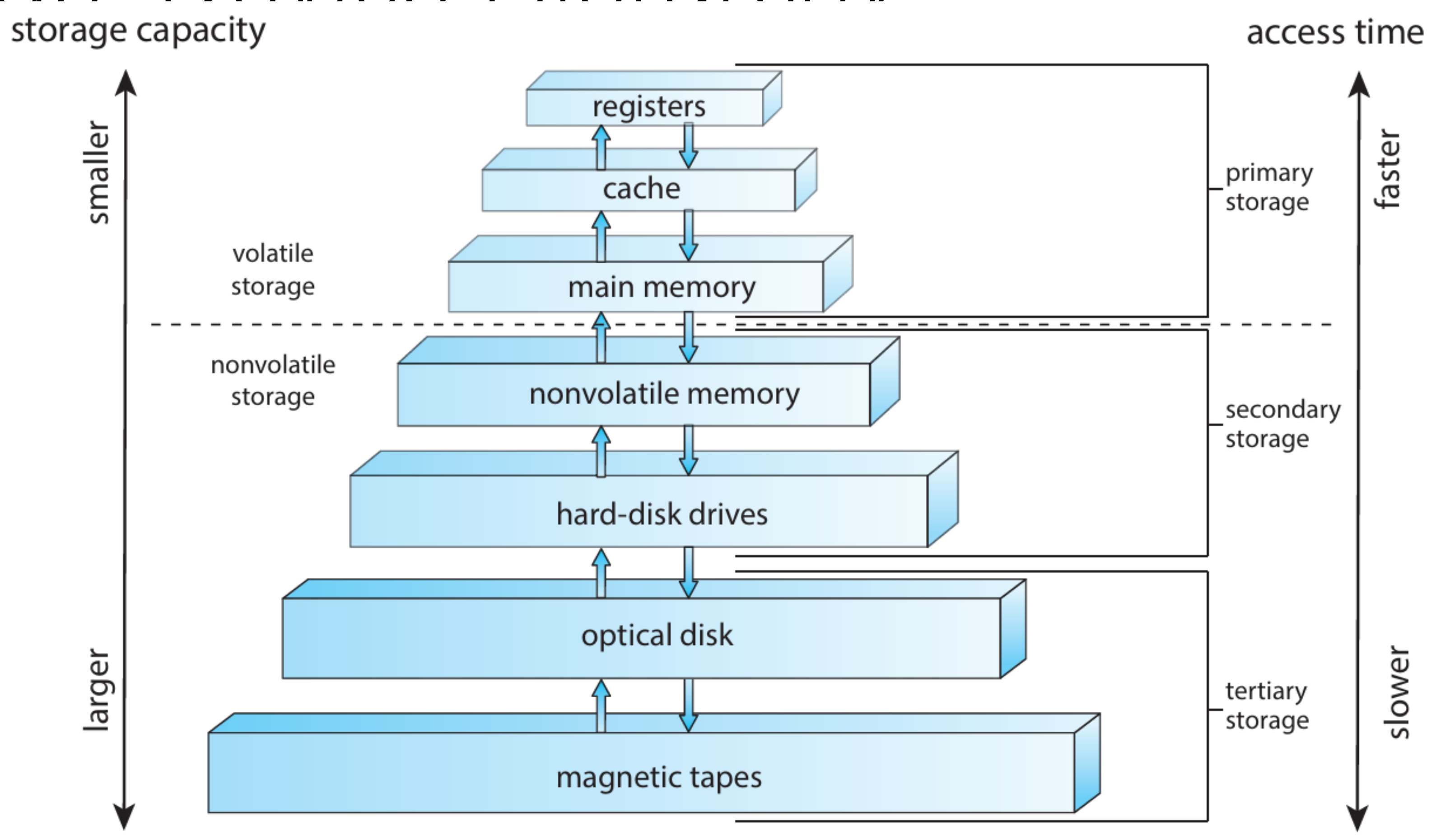


Figure 1.6 Storage-device hierarchy.

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-purpose processor (PDAs through mainframes)
  - 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:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability – graceful degradation** or **fault tolerance**
  - Two types:
    1. **Asymmetric Multiprocessing**
    2. **Symmetric Multiprocessing**

How a Modern Computer Works

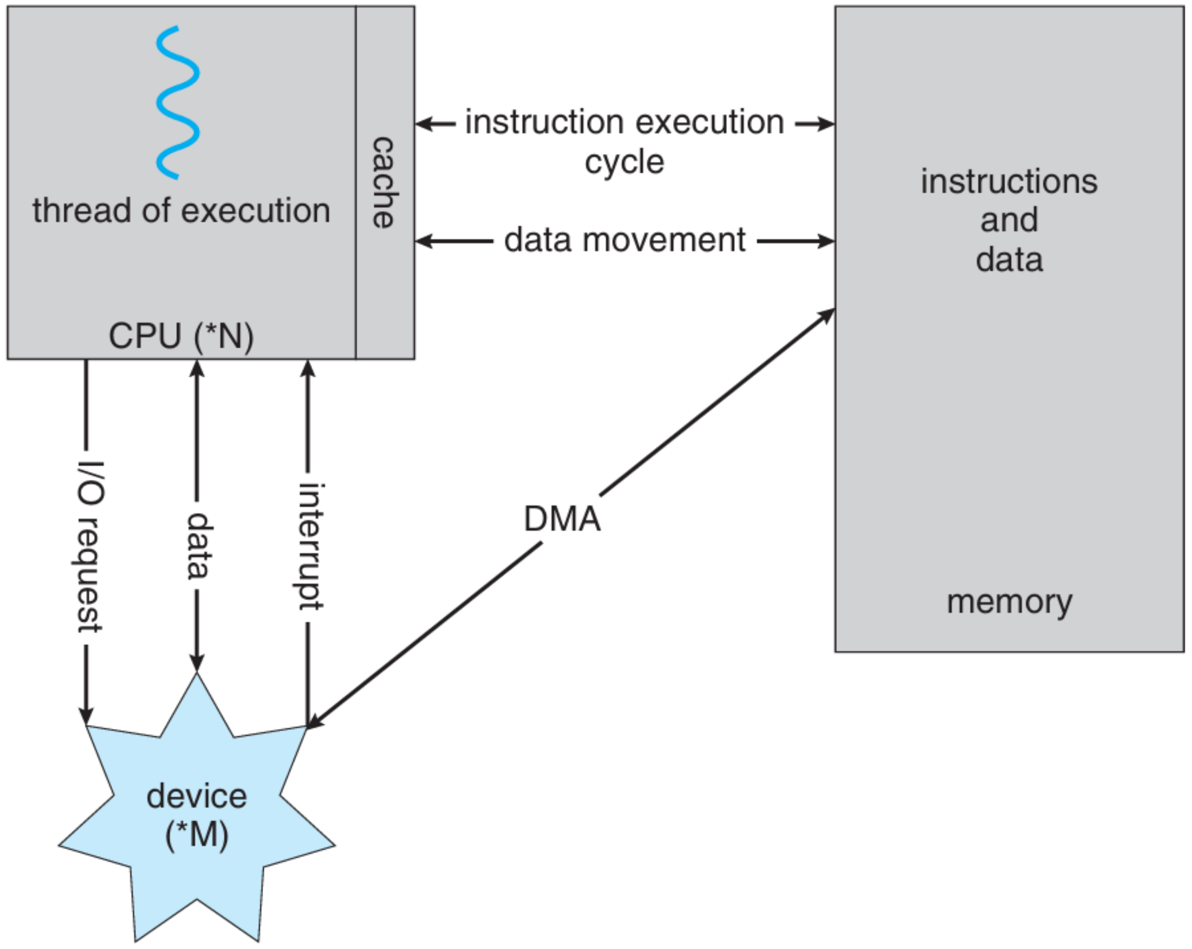


Figure 1.7 How a modern computer system works.

A von Neumann architecture

Symmetric Multiprocessing Architecture

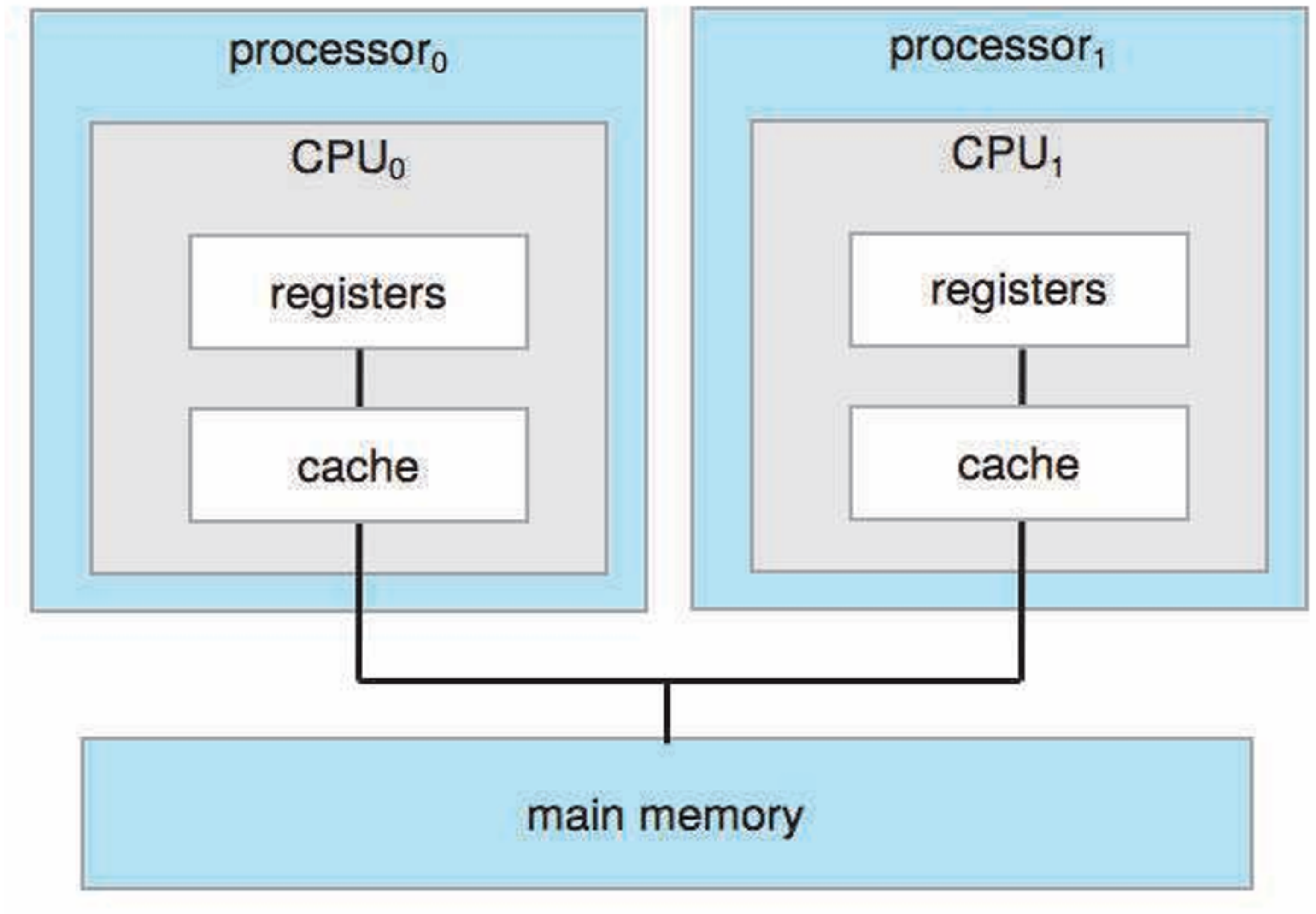


Figure 1.8 Symmetric multiprocessing architecture.

A Dual-Core Design

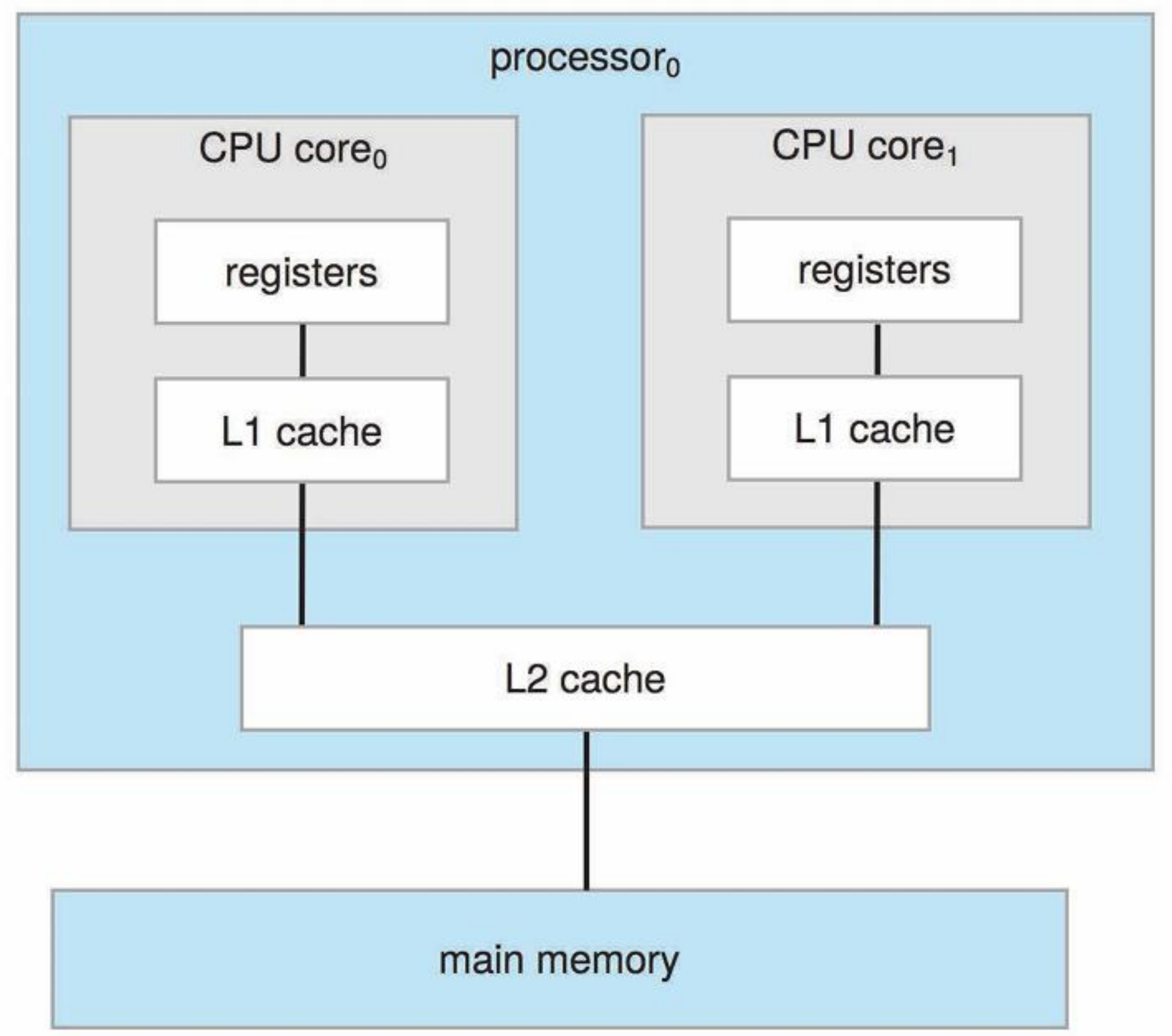


Figure 1.9 A dual-core design with two cores on the same chip.

DEFINITIONS OF COMPUTER SYSTEM COMPONENTS

- **CPU**—The hardware that executes instructions.
- **Processor**—A physical chip that contains one or more CPUs.
- **Core**—The basic computation unit of the CPU.
- **Multicore**—Including multiple computing cores on the same CPU.
- **Multiprocessor**—Including multiple processors.

Although virtually all systems are now multicore, we use the general term ***CPU*** when referring to a single computational unit of a computer system and ***core*** as well as ***multicore*** when specifically referring to one or more cores on a CPU.

Clustered Systems

A clustered system is another type of multiprocessor architecture that brings together multiple CPUs. Unlike the multiprocessor systems outlined in Section 1.3.2, clustered systems consist of two or more individual systems—referred to as nodes—each of which is typically a multicore system. These systems are generally considered loosely coupled.

It's important to note that the definition of a clustered system is not universally agreed upon; various commercial and open-source solutions strive to articulate what constitutes a clustered system and the merits of different forms. However, the widely accepted definition indicates that clustered computers share storage and are interconnected through a local-area network (LAN), or a faster interconnect such as InfiniBand.

Clustered Systems

- Clustering is usually used to provide high-availability service
- High availability provides increased reliability, which is crucial in many applications.
- Clustering can be structured asymmetrically or symmetrically.
- In asymmetric clustering, one machine is in hot-stand by mode while the other is running the applications.
- In symmetric clustering, two or more hosts are running applications and are monitoring each other.

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**

Clustered Systems

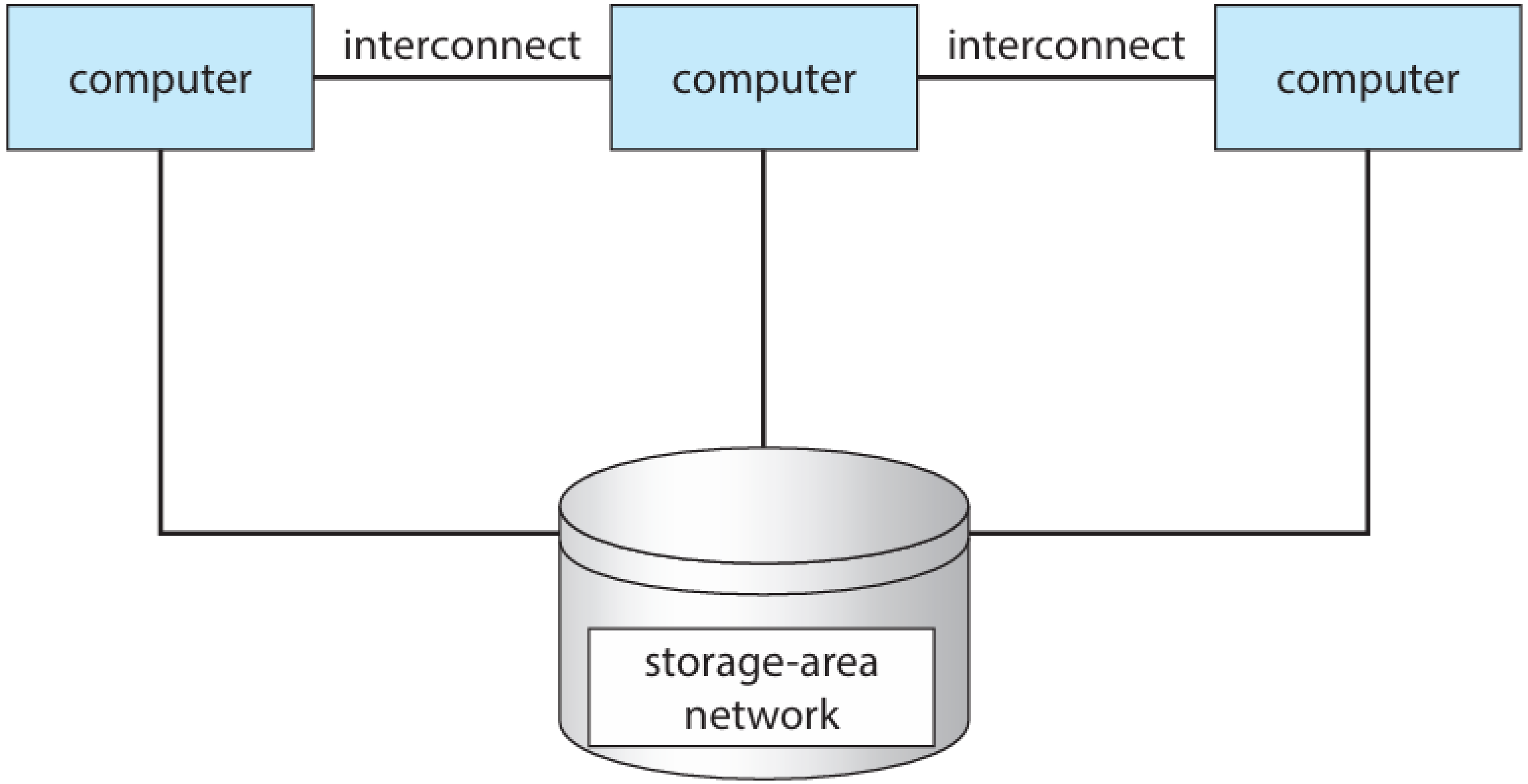


Figure 1.11 General structure of a clustered system.

Homework

How **latency** and **throughput** is important in **OS**