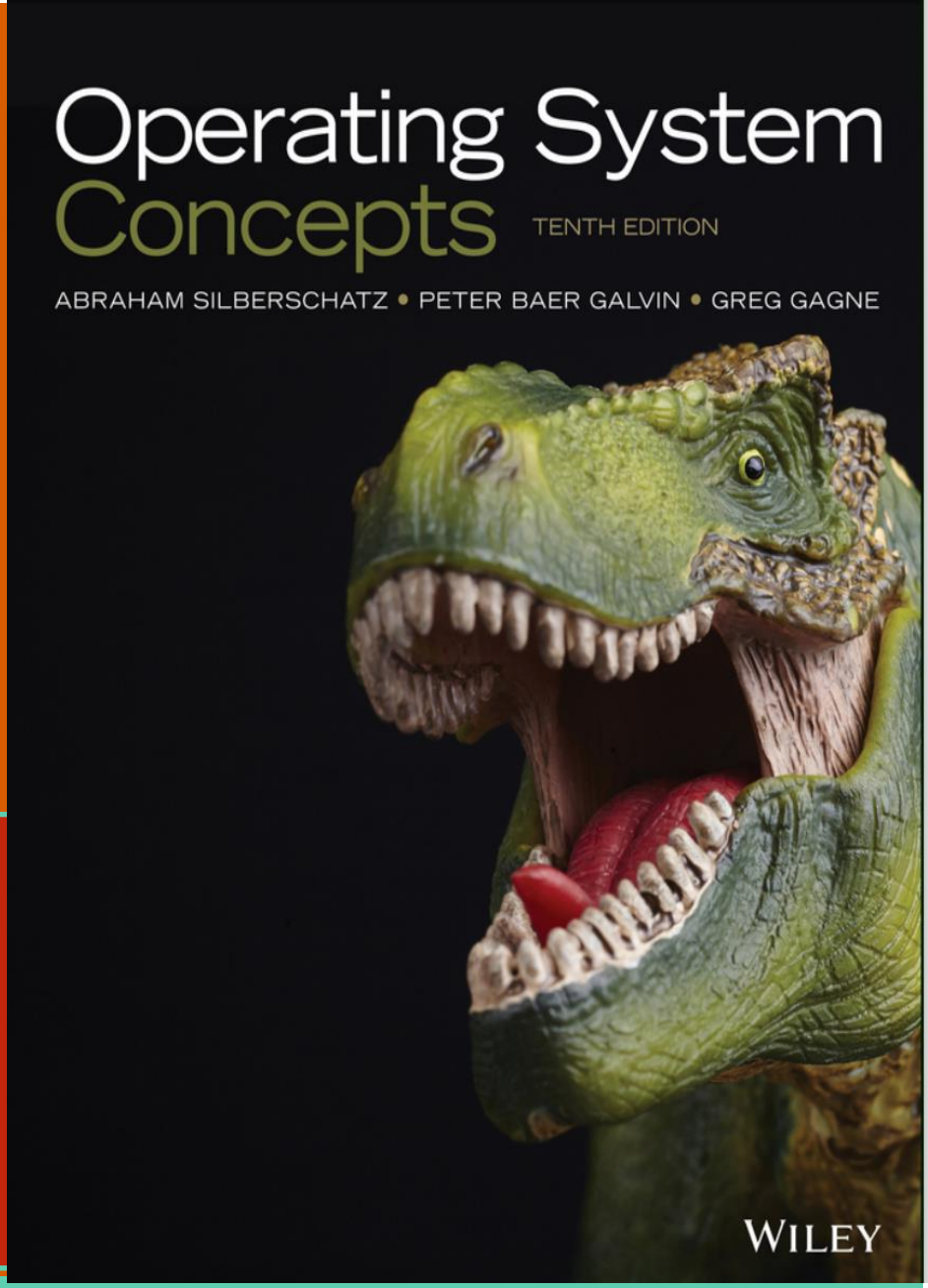
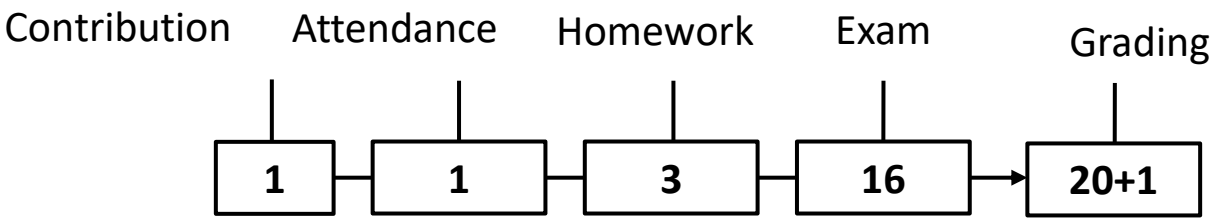
Operation Systems



Dr. A. Taghinezhad

Website: ataghinezhad.github.io, Email:a0taghinezhad@gmail.com

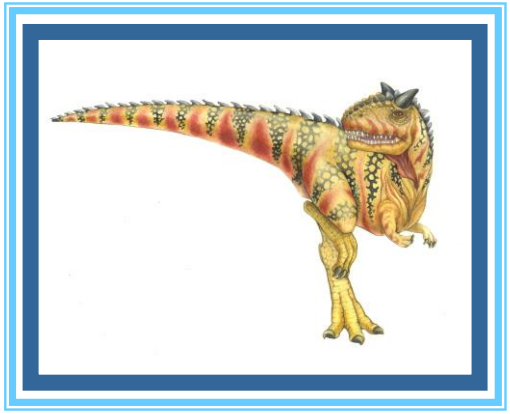
ارزیابی

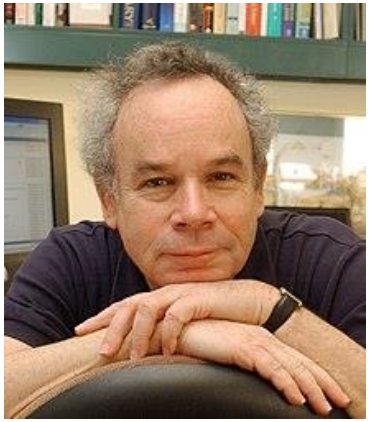


Outline of this Course

- Introduction
- Processes
- Threads and Concurrency
- CPU Scheduling
- Deadlocks
- Main Memory
- Virtual Memory
- Mass-Storage Structure

Chapter 1: Introduction





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
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems

Objectives

- To describe the **basic organization of computer** systems
- To provide a grand tour of the **major components of operating systems**
- To give an overview of the **many types of computing environments**
- To explore several open-source operating systems

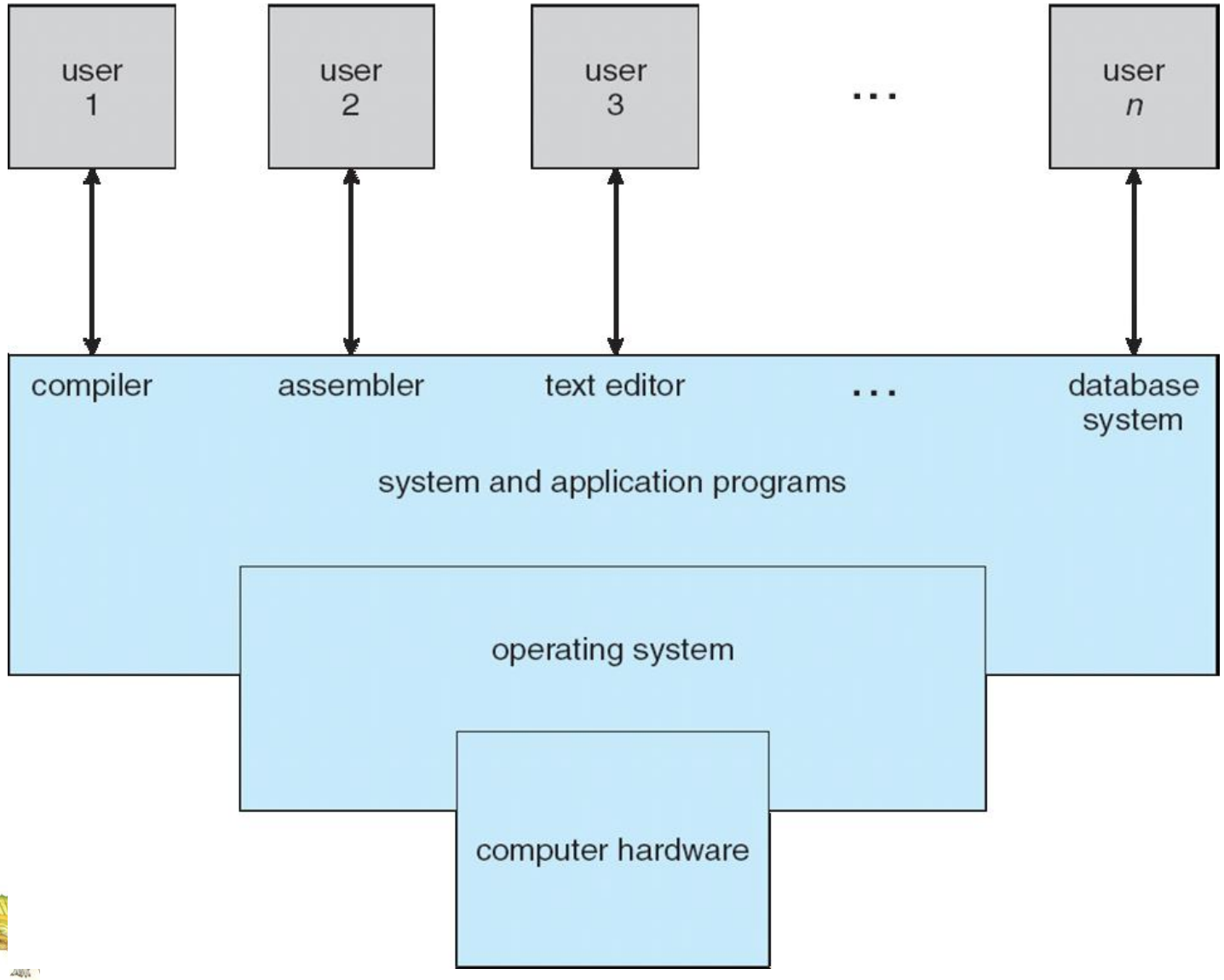
What is an Operating System?

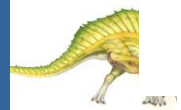
- A program that acts as an **intermediary** between a user of a computer and the computer hardware
- Operating system **goals**:
  - **Execute user programs** and make solving user problems easier
  - Make the computer system **convenient to use**
  - Use the computer **hardware** in an **efficient**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

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 application The computerns 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** and **good performance**
  - 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

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.)

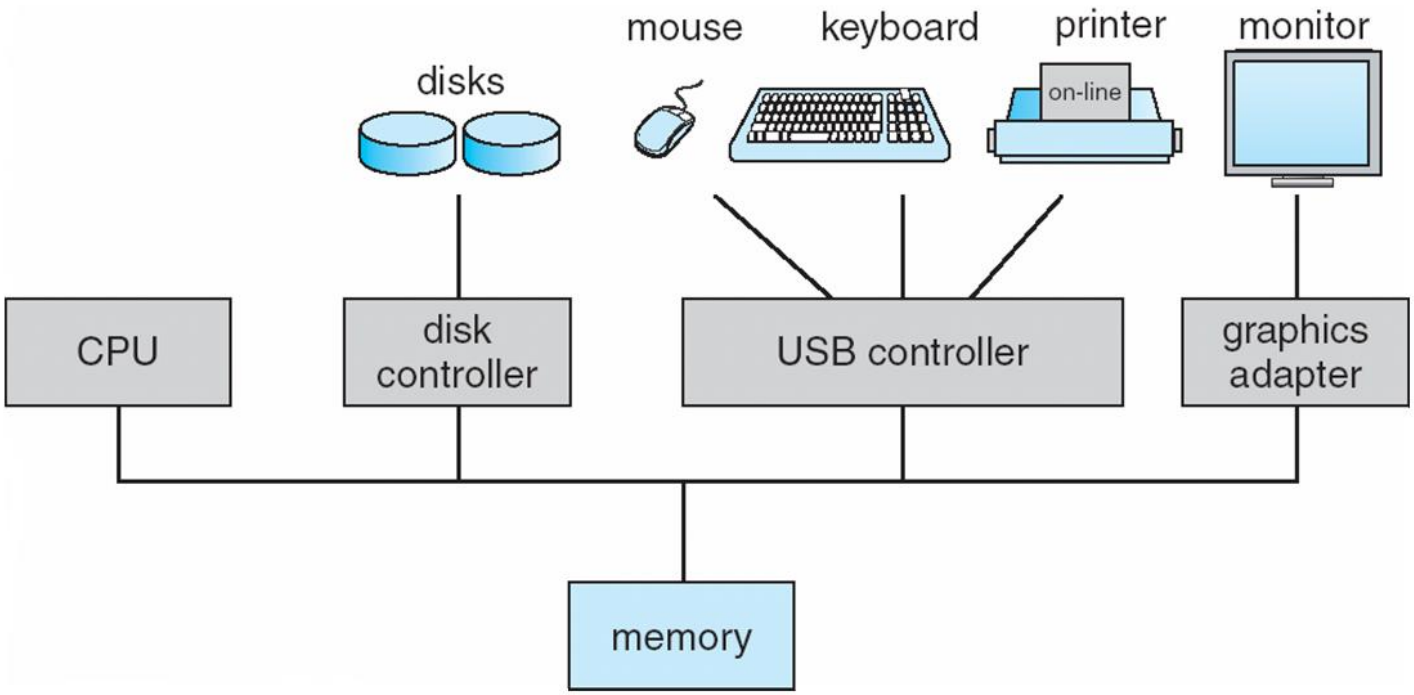
- **No universally accepted definition**
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating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

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-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

Device controller

- A **device controller** in an operating system (OS) plays a crucial role in managing communication between the computer system and input/output (I/O) devices.

1. **Definition**:
    1. A **device controller** is an **electronic component** responsible for handling the **communication** between the **CPU** (central processing unit) and various **I/O devices**.
    2. It acts as an **interface** between the computer system (operating system) and the I/O devices.
    3. Convert **Serial bits stream** to **blocks of Bytes**.

device controller

1. Controller's tasks :

1. It converts **serial bit stream** to **block** of bytes.
2. **Perform error correction** if necessary.
3. Block of bytes is first assembled bit by bit in buffer inside the controller.
4. After verification, the block has been declared to be error free and then it can be copied to main memory

Interrupt

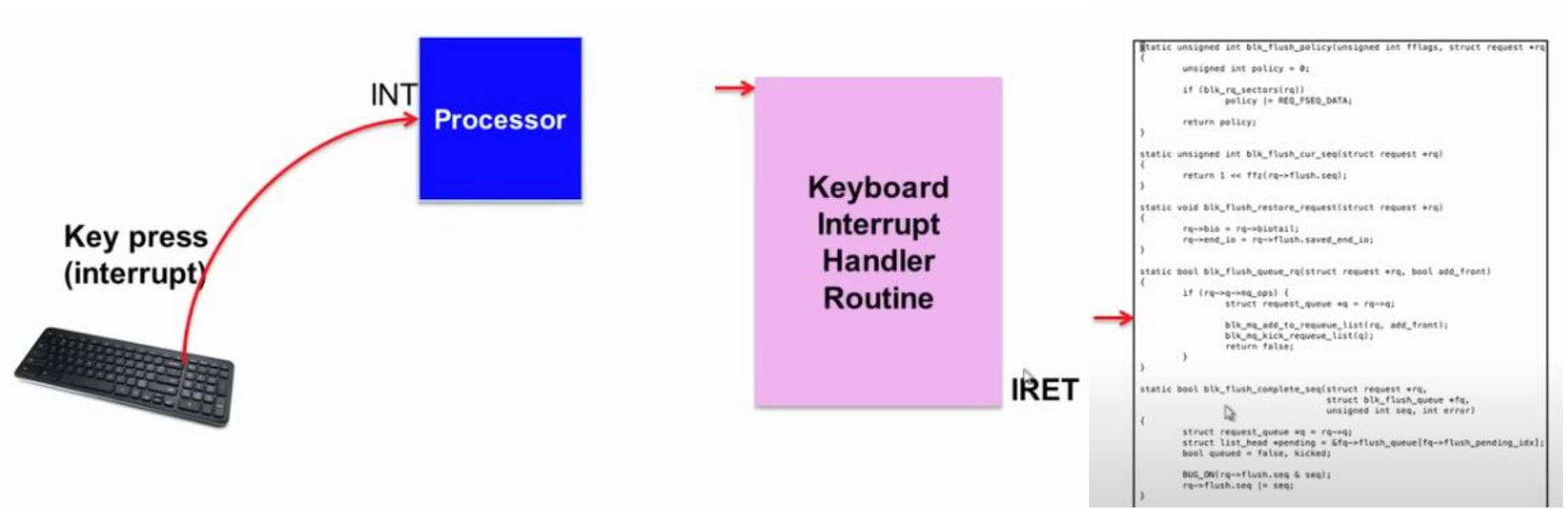
- An **interrupt** in an operating system (OS) is a **signal** emitted by either hardware or software when a process or an event requires **immediate attention (high-priority process)**.

Interrupt

- **Hardware Interrupts**: These are related to the state of hardware devices.
  - Examples include **interrupt request (IRQ) lines** on a PC or signals from devices embedded in processor logic (e.g., CPU timers).
- **Software Interrupts**: These are produced by **software** or the **system** itself (as opposed to hardware). (Traps)

1. **Interrupt Latency**: The delay between receiving an interrupt and starting the execution of the interrupt handler.

Hardware Interrupt

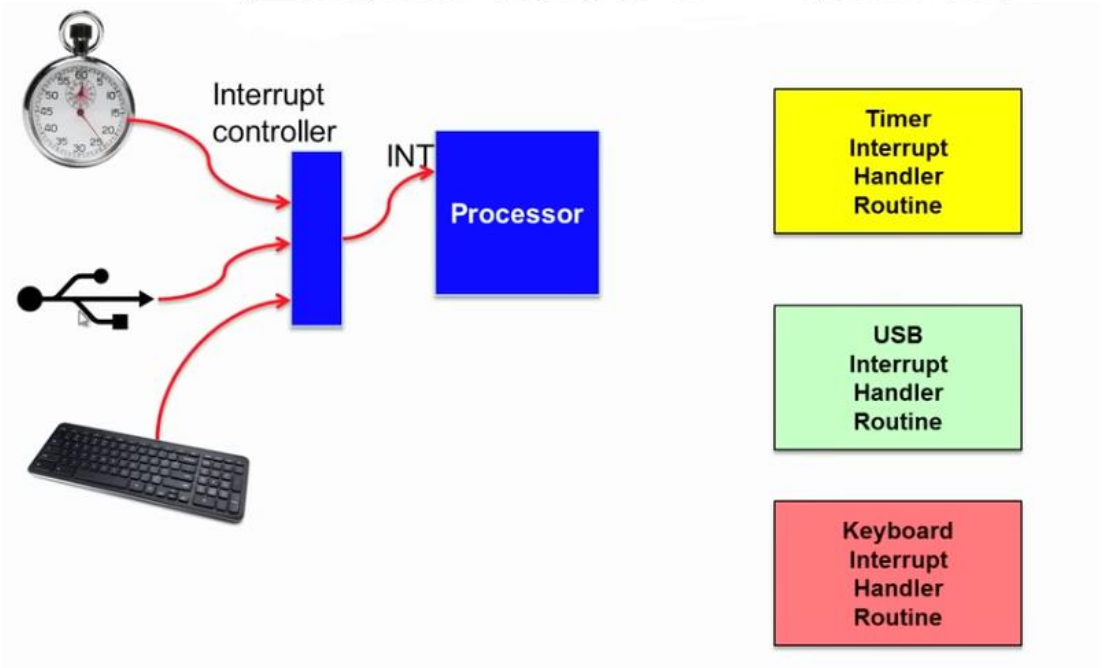


Common Functions of Interrupts

- Interrupt **transfers** control to the **interrupt service routine** generally, 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, System Calls, Synchronous Event
- An operating system is **interrupt-driven**

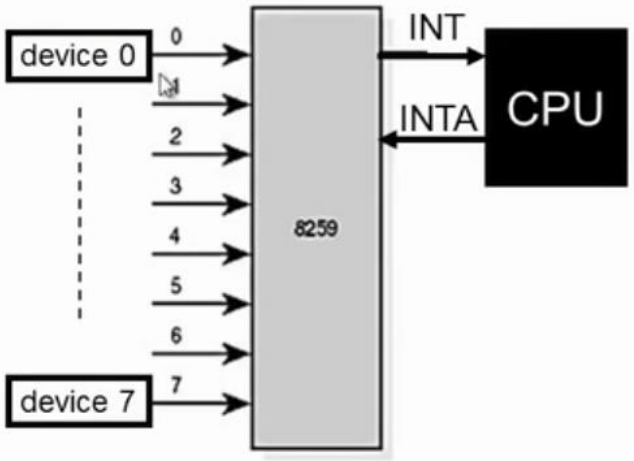
Interrupt Handling

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



Programmable Interrupt Controller

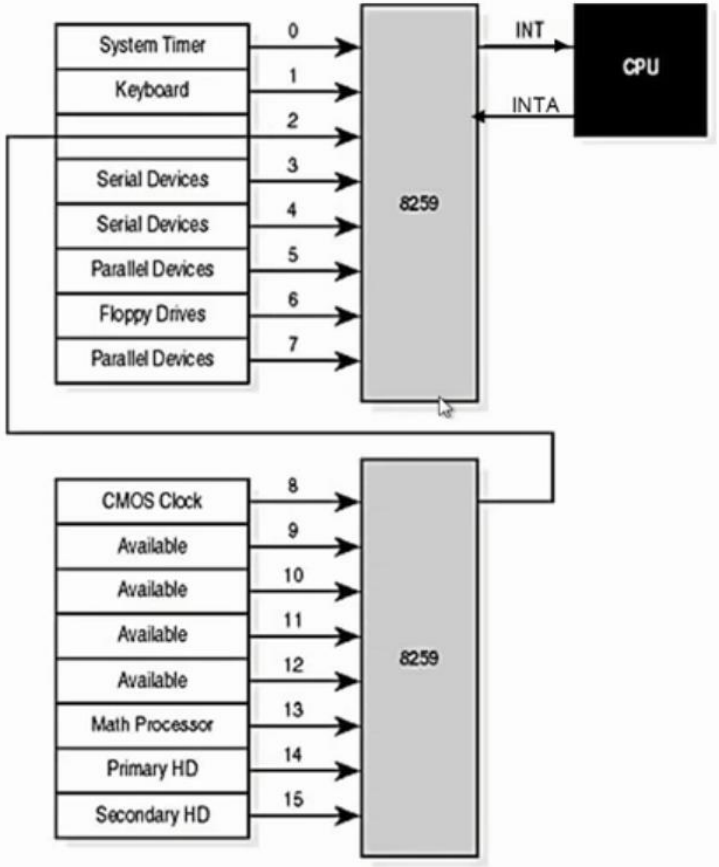
- 8259 (Programmable interrupt controller) relays upto 8 interrupt to CPU
- Devices raise interrupts by an ‘interrupt request’ (IRQ)
- CPU acknowledges and queries the 8259 to determine which device interrupted
- Priorities can be assigned to each IRQ line
- 8259s can be cascaded to support more interrupts



The Intel **8259** is a programmable interrupt controller (PIC) designed for the Intel 8085 and Intel 8086 microprocessors.

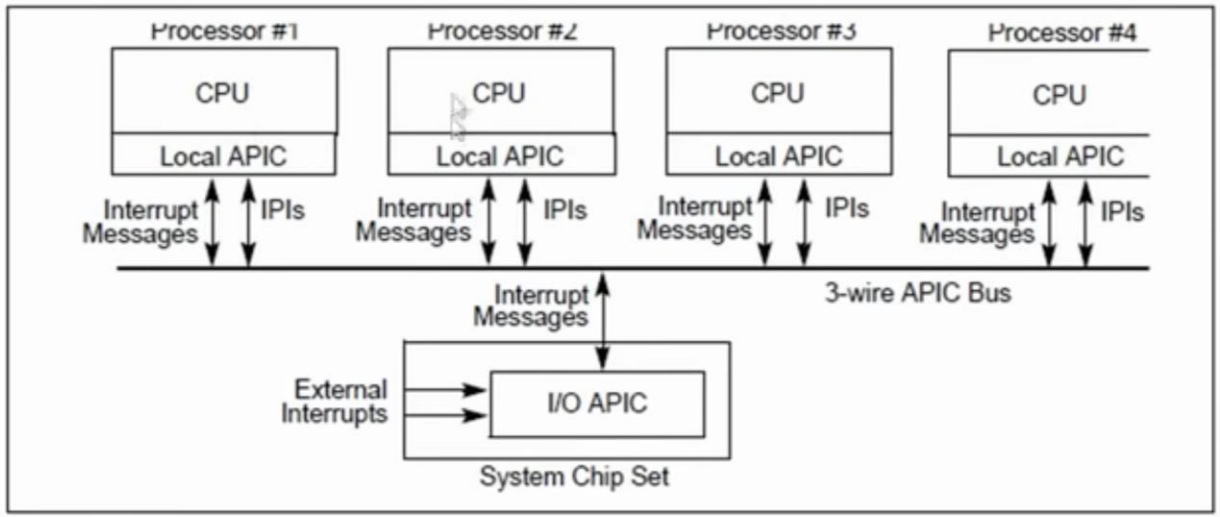
Interrupts in legacy CPUs

- 15 IRQs (IRQ0 to IRQ15), so 15 possible devices
- Limitations
  - Limited IRQs
  - Not suited for multi-processor / multi-core platforms



New systems Interrupt handler per CPU

Advanced Programmable Interrupt Controller (APIC)



- External interrupts are routed from peripherals to CPUs in multi processor systems through APIC
- APIC distributes and prioritizes interrupts to processors
- Comprises of two components
  - Local APIC (LAPIC)
  - I/O APIC
- APICs communicate through a special 3-wire APIC bus.

Interrupt Timeline

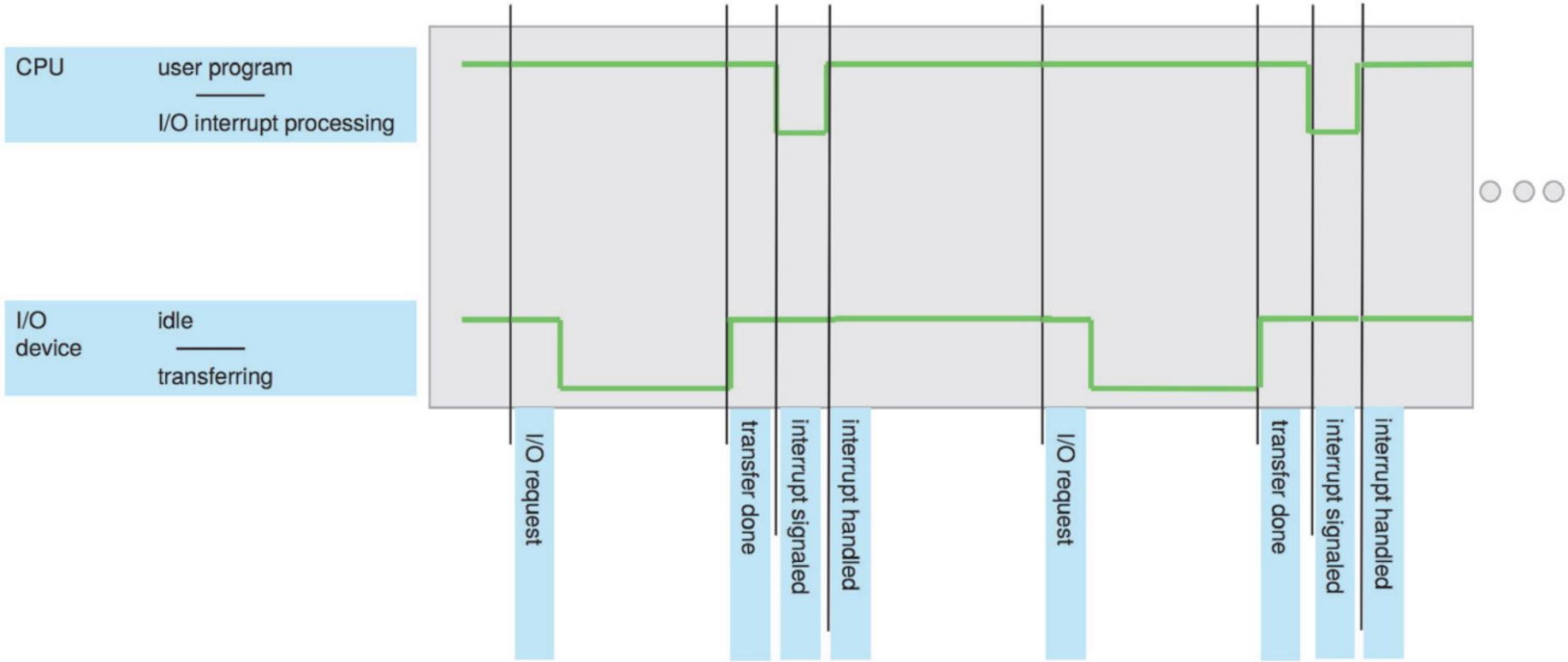


Figure 1.3 Interrupt timeline for a single program doing output.

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Figure 1.5 Intel processor event-vector table.

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 **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. A less common term is **word**, which is a given computer architecture's native unit of data.
- 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
- a **petabyte**, or **PB**, is $1,024^5$ bytes

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

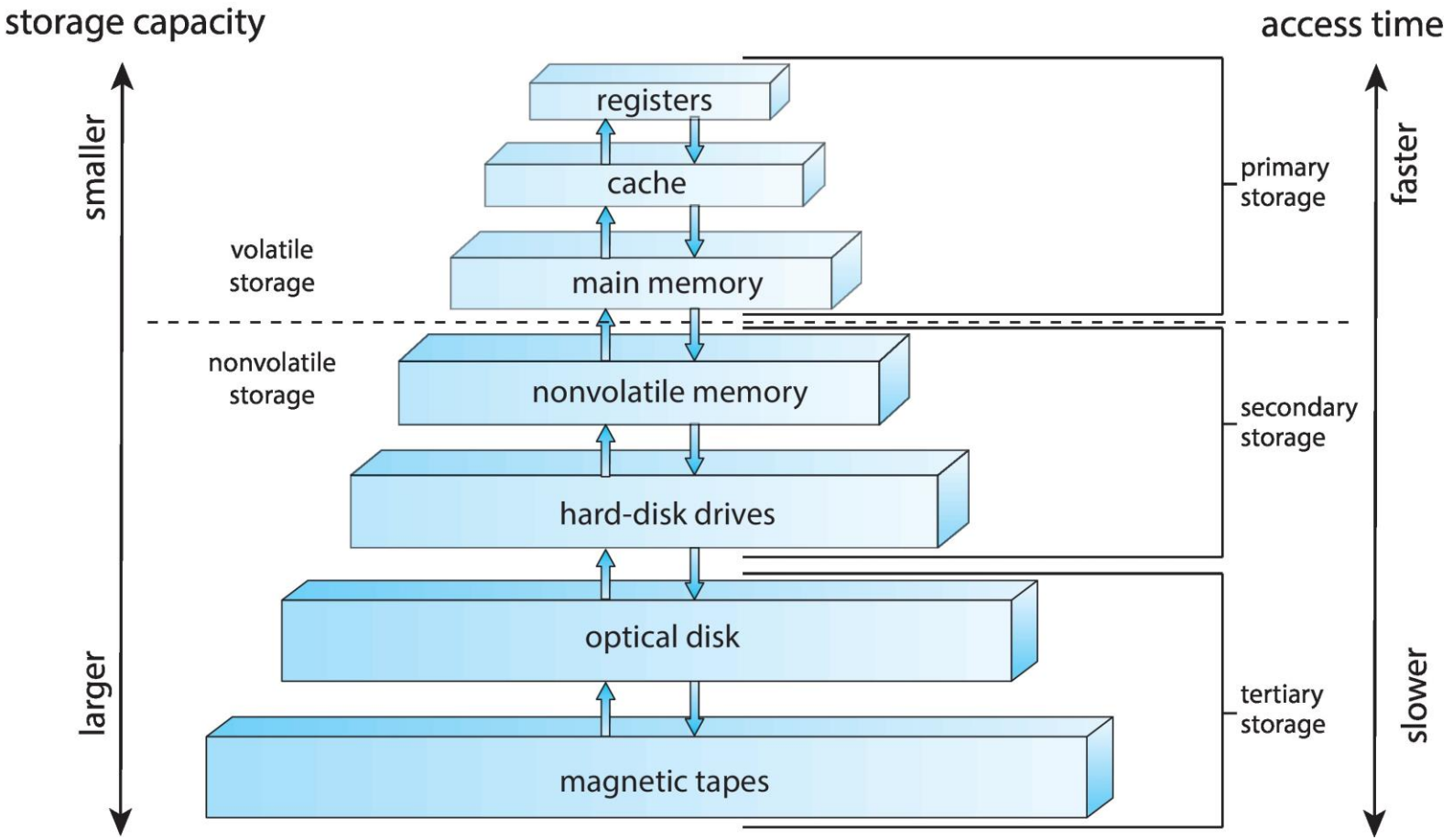
Storage Structure

- Hard 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
- **Solid-state disks** – faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular

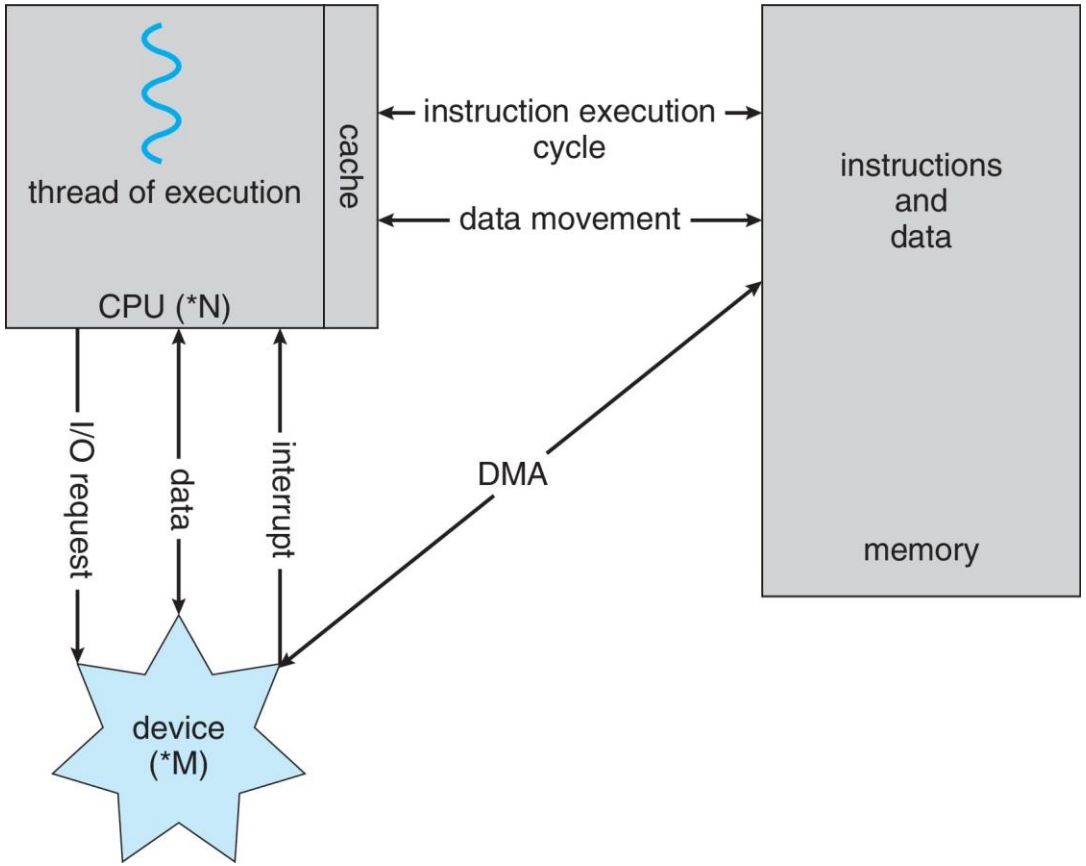
Storage Hierarchy

- 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 Structure

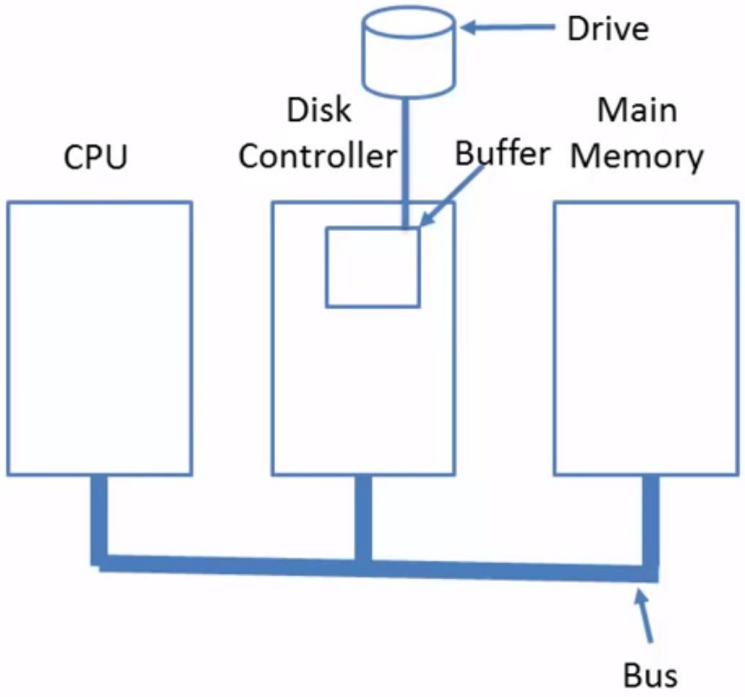
- It is a hardware device that allows certain hardware device that allows certain subsystems to **access main memory (RAM)**, **independent of the central processing unit (CPU).**
- This feature is useful when the CPU needs to perform useful work while **waiting** for a **relatively slow I/O data transfer**.
- Hardware systems that use DMA: disk drive, graphics cards, network cards, sound cards

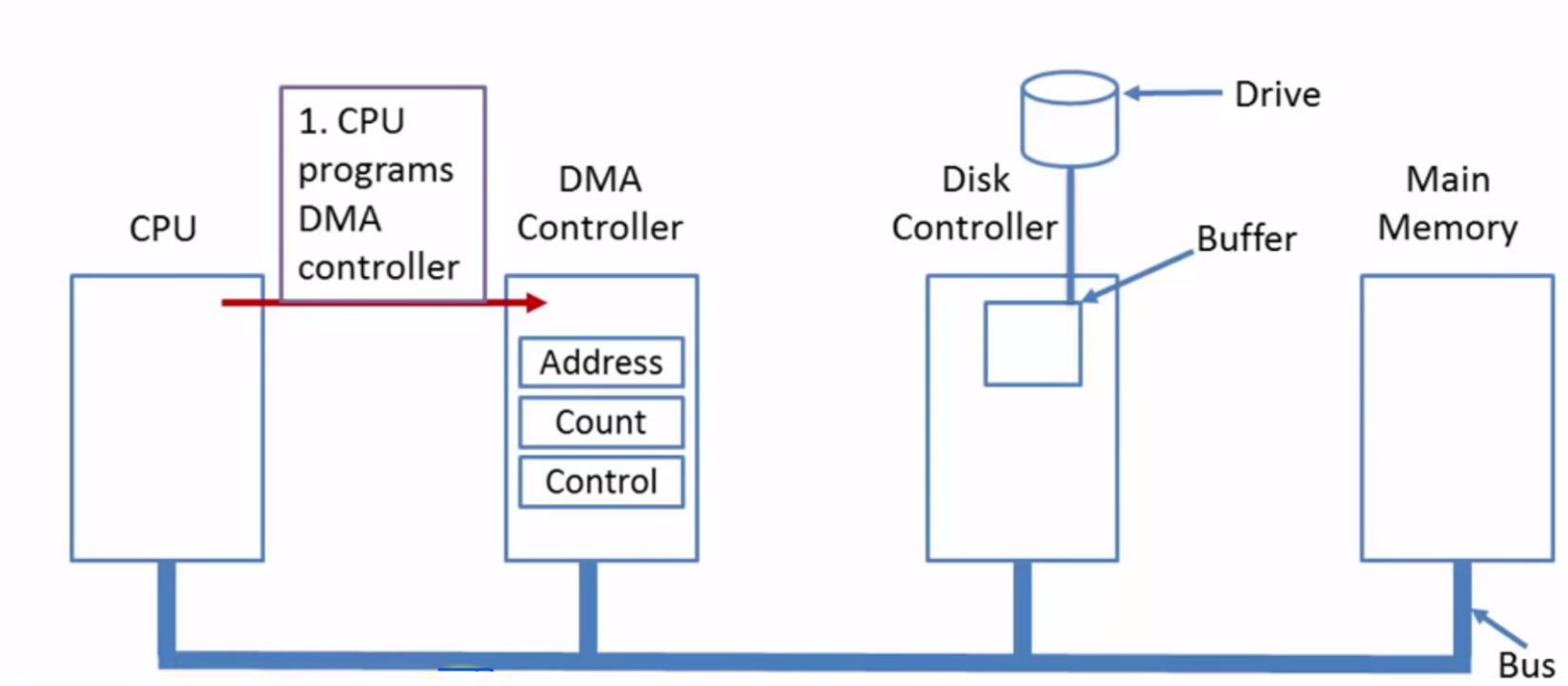
Direct Memory Access Structure

- **Without DMA**, when the CPU is using programmed input/output, it is **typically fully occupied for the entire duration** of the read or write operation, and is thus unavailable to perform other work.
- With DMA, **the CPU first initiates the transfer**, **then it does other operations** while the transfer is in progress, and it finally receives an interrupt from the DMA controller when the operation is done.

Disk read-write without a DMA

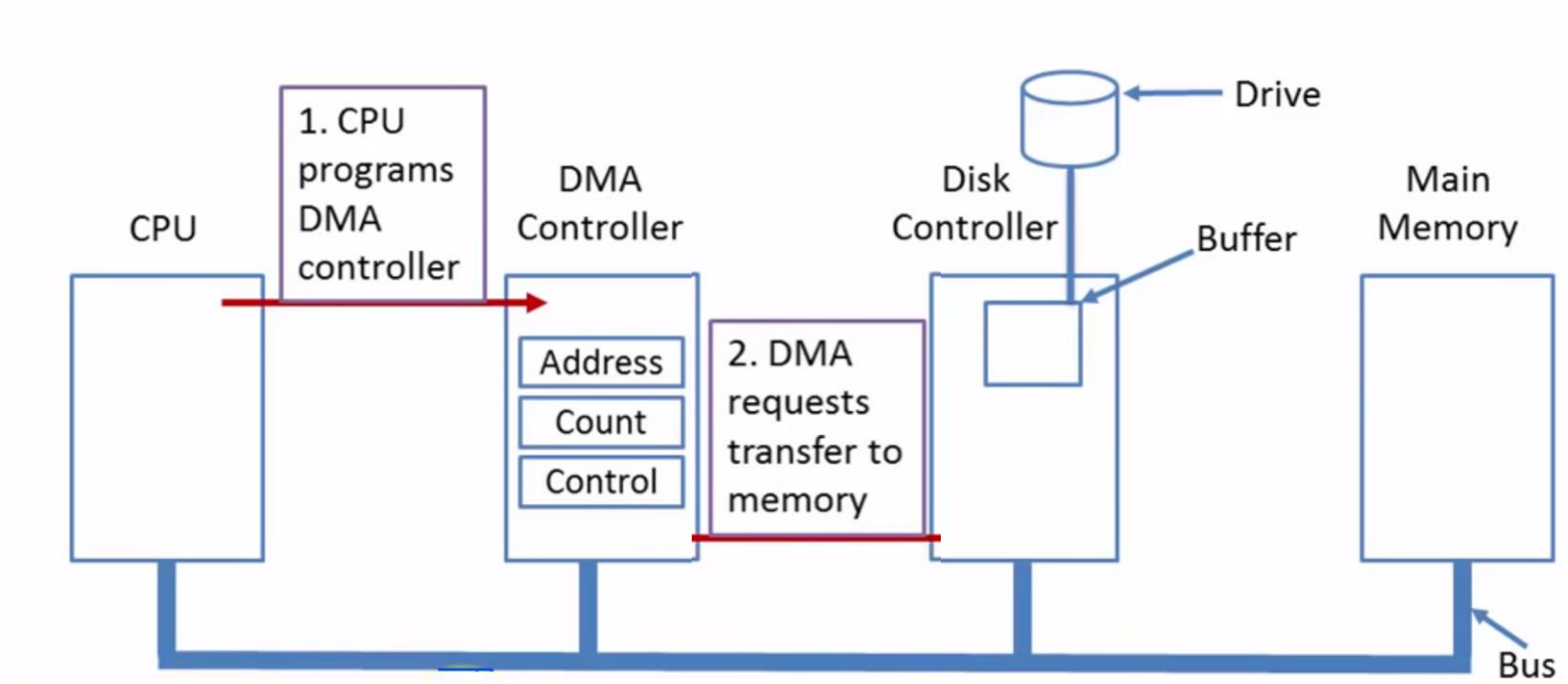
- The **disk controller reads the block** from the **drive serially**, bit by bit until the entire block is in the controller's buffer.
- It computes the **checksum** to verify that no read errors have occurred.
- Then the **controller causes an interrupt** so that OS can read the block from controller's buffer (a byte or a word at a time) by executing a loop.
- After reading every single part of the block from the controller device register, the operating system will store them in the main memory.





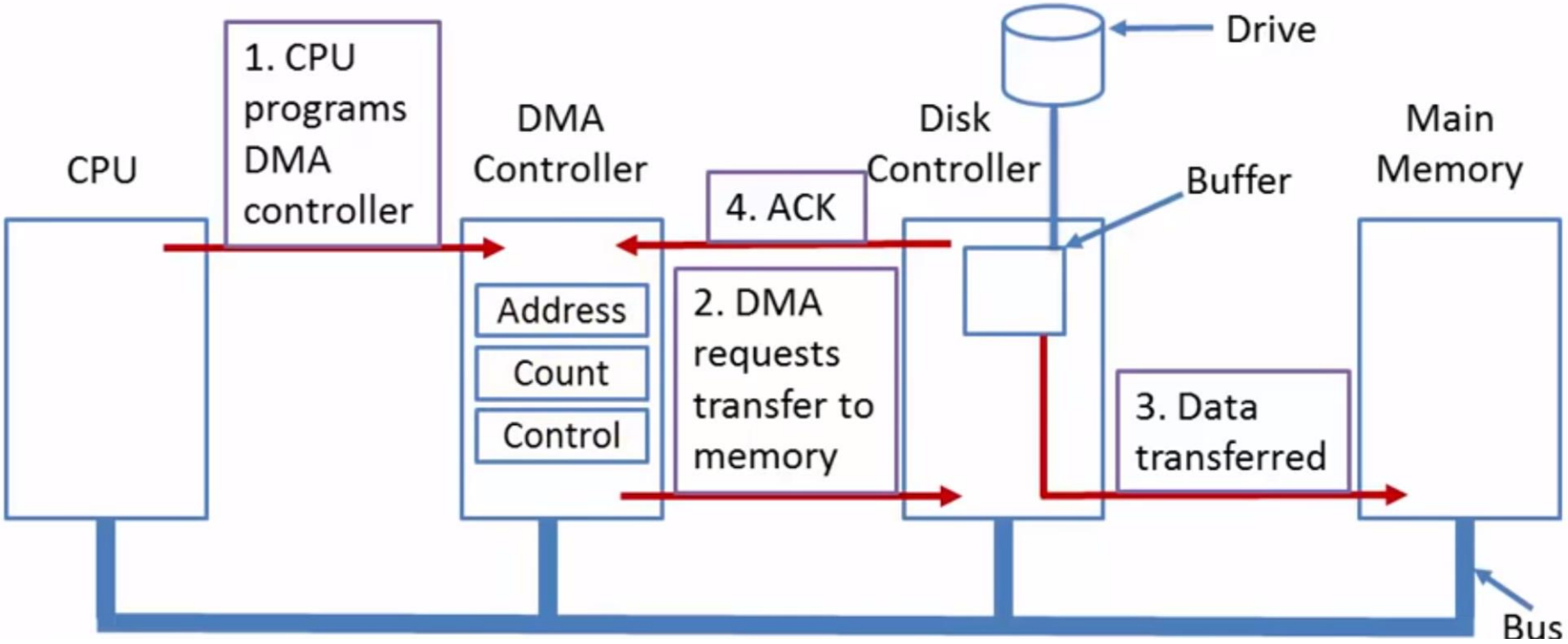
Step 1:

- First the **CPU programs the DMA** controller by setting **its registers so it knows what to transfer where.**
- It **also issues a command** to the disk controller to **read data from the disk into its internal buffer** and verify the **checksum**.
- When **valid data** are in the **disk controller's buffer**, DMA can begin.



Step 2:

- The **DMA controller initiates the transfer** by issuing a **read request** over **the bus to the disk controller.**
- The **disk controller does not know** (or care) whether it came from the CPU or a DMA controller.
- Step 3: The data from the **buffer moves to main memory**



- Step 4:
  - When the write is complete, the **disk controller sends an ASK** signal to the **DMA** controller **over** the **bus**.
  - The **DMA** controller then **increments** the **memory address in DMA control** to use and **decrements** the byte **Count**.
  - If the byte count is still greater than 0, steps 2 to 4 are repeated until it reaches 0. It lets the CPU know the transfer is completed by Interrupt.

Modes of Bus operation

- The buses can be operated in **two modes**
  - 1. **Word-at-a-time mode**:
    - Here the DMA requests for the transfer of one word and gets it.
    - If CPU wants the bus at the same time then it has to wait.
    - This mechanism is known as Cycle.
  - 2**. Block mode**:
    - Here the DMA controller tells the device to acquire the bus, issues a series of transfers, and then releases the bus.
    - This form of the operation is called Burst mode.
    - It is more efficient than cycle.

Operating-System Operations

- Bootstrap 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

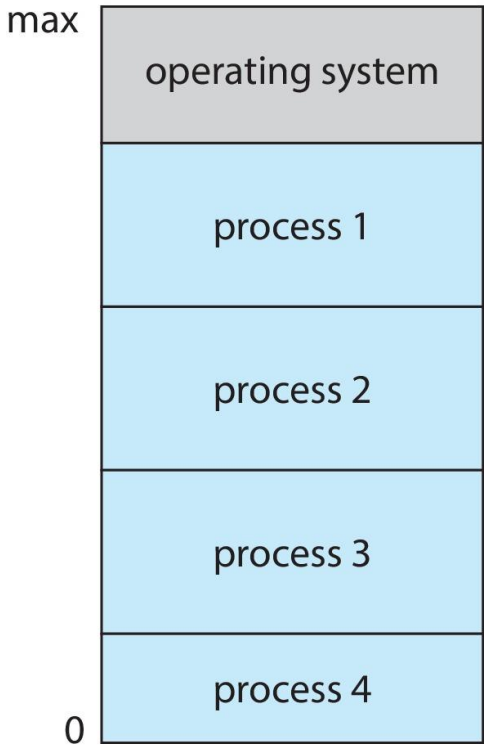
Multiprogramming (Batch system)

- 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)

- 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, which is called **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

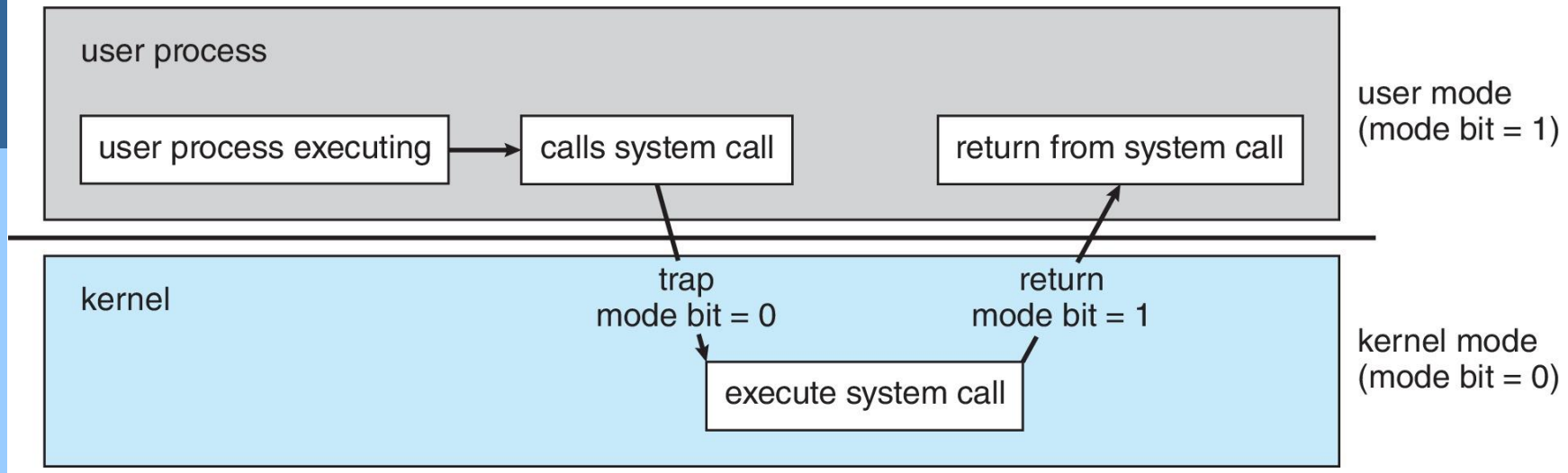
Memory Layout for Multiprogrammed System



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 u**ser 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



- At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user applications in user 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** turn to **zero**, it generate an **interrupt**
  - Set up before scheduling process **to regain control or terminate program** that exceeds allotted time

Process Management

- 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**

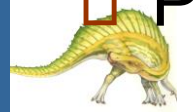
Process Management

- Single-threaded process has one **program counter** specifying **location of next instruction** to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has **one program counter per thread**
- Typically, system has many processes, some user, some operating system running concurrently on one or more CPUs
  - 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**
- Providing mechanisms for **process synchronization**
- Providing mechanisms for **process communication**
- Providing mechanisms for **deadlock handling**



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

- Memory management activities
  - **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 Management

- 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 drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

File-system Management

- File-System management
  - Files usually organized into directories
  - **Access control** on most systems to determine who can access what
  - 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

Mass-Storage Management

- Usually, disks are used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
  - **Mounting** and unmounting
  - Free-space management
  - Storage allocation
  - **Disk scheduling**
  - **Partitioning**
  - **Protection**

Caching

- 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

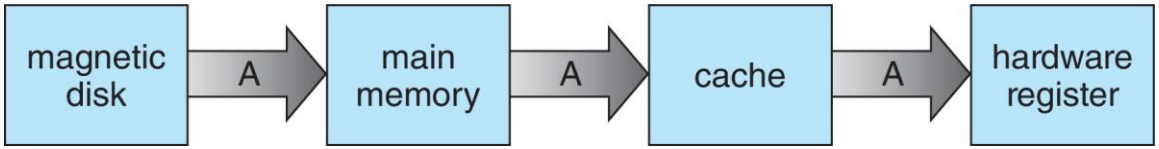
Characteristics of Various Types of Storage

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

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

Migration of data “A” from Disk to Register

- **Cache coherency** is a consistency mechanism for ensuring that all CPUs in a multiprocessor system observe a single, consistent view of the data.When one CPU updates a location in its cache, the corresponding location in other caches must be updated or invalidated to maintain coherency.
- two processors, P1 and P2, that have their own caches and are both trying to access a memory location M.
- **P1 Cache: M=10; P2 Cache: M=10 ; Main Memory: M=10**
  - Now, P1 changes the value of M to 20 in its cache. Without a cache coherency protocol, P2 would still see the old value of M (10) in its cache.
  - With a **cache coherency protocol in place, when P1 changes the value of M in its cache**, the change is propagated to the main memory and P2’s cache. So, P2 sees the updated value of M (20).

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- 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)
  - General device-driver interface
  - Drivers for specific hardware devices

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

Protection and Security

- 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

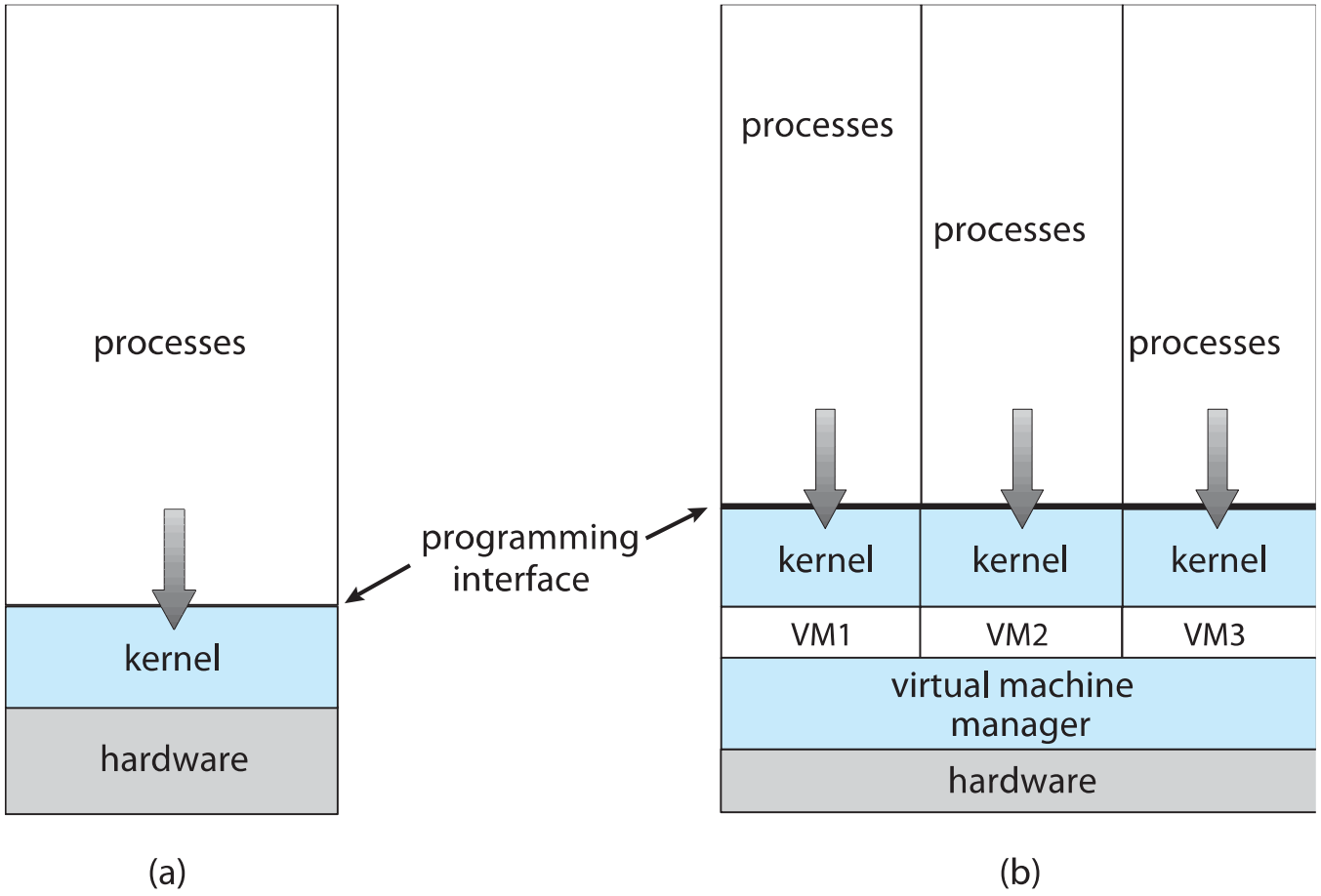
Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Emulation** used **when source CPU type different from target type** (i.e. PowerPC to Intel x86)
  - Generally slowest method
  - When computer language not compiled to native code – **Interpretation**
- **Virtualization** – **OS natively compiled for CPU**, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services

Virtualization (cont.)

- **Use cases involve laptops and desktops running multiple OSes** for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - **Developing apps for multiple OSe**s without having multiple systems
  - **Quality assurance** testing applications without having multiple systems
  - **Executing and managing compute environments** within data centers
- **VMM can run natively**, in which case they are also the host
  - There is no general-purpose host then (VMware ESX and Citrix XenServer)

Computing Environments - Virtualization



Distributed Systems

- Distributed computing
  - Collection of separate, possibly heterogeneous, systems networked together
    - **Network** is a communications path, **TCP/IP** most common
  - **Network Operating System** provides features between systems across the network
    - Communication scheme allows systems to exchange messages
    - **Illusion of a single system**

Network Operating System VS Distributed Systems

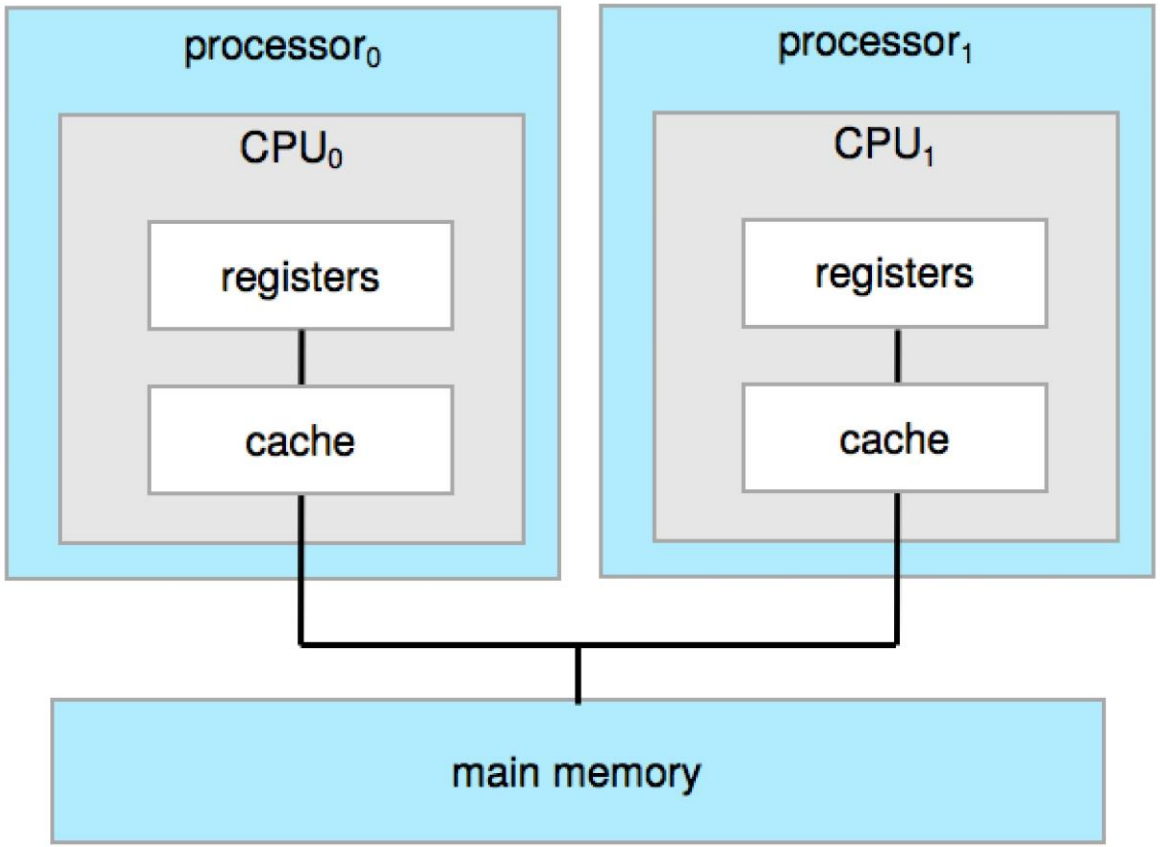
- **Windows Server:** A popular network operating system that allows centralized management of network resources like files, printers, and users. Computers on the network operate independently but can connect to the Windows server to access resources.
- **Android OS:** A distributed operating system that runs on smartphones and tablets. It communicates with other components of the operating system on other networked devices like smartwatches and smart speakers. This communication creates the illusion of a single operating system for the end user, while in reality, various tasks are distributed across different devices.

Computer-System Architecture

Computer-System Architecture

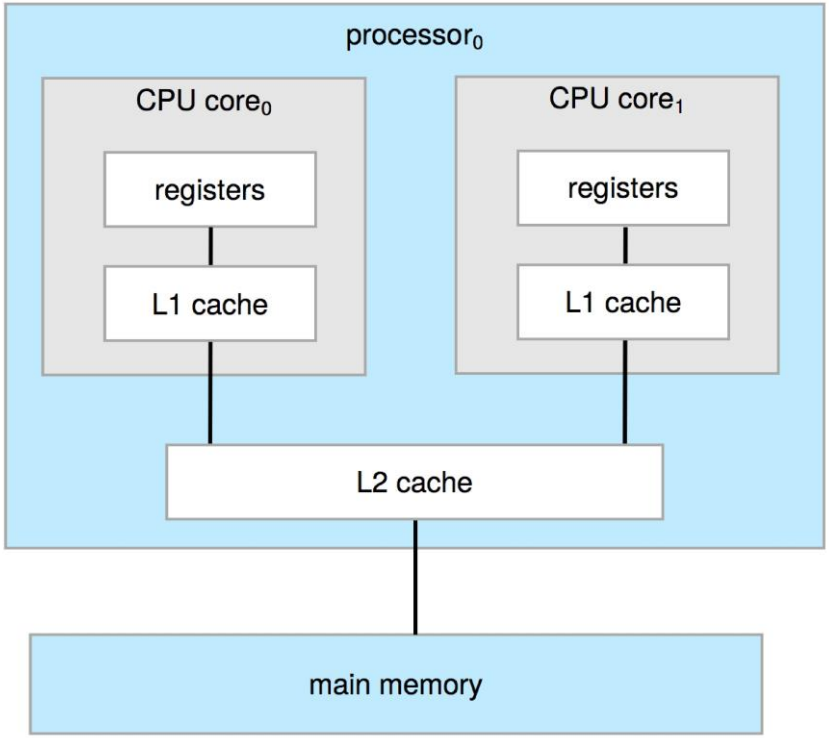
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessor's** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2. **Economy of scale**
    3. **Increased reliability** – graceful degradation or fault tolerance
  - Two types:
    1. **Asymmetric Multiprocessing (AMP)**– each processor is assigned a specie task.
    2. **Symmetric Multiprocessing (SMP)** – each processor performs all tasks

Symmetric Multiprocessing Architecture

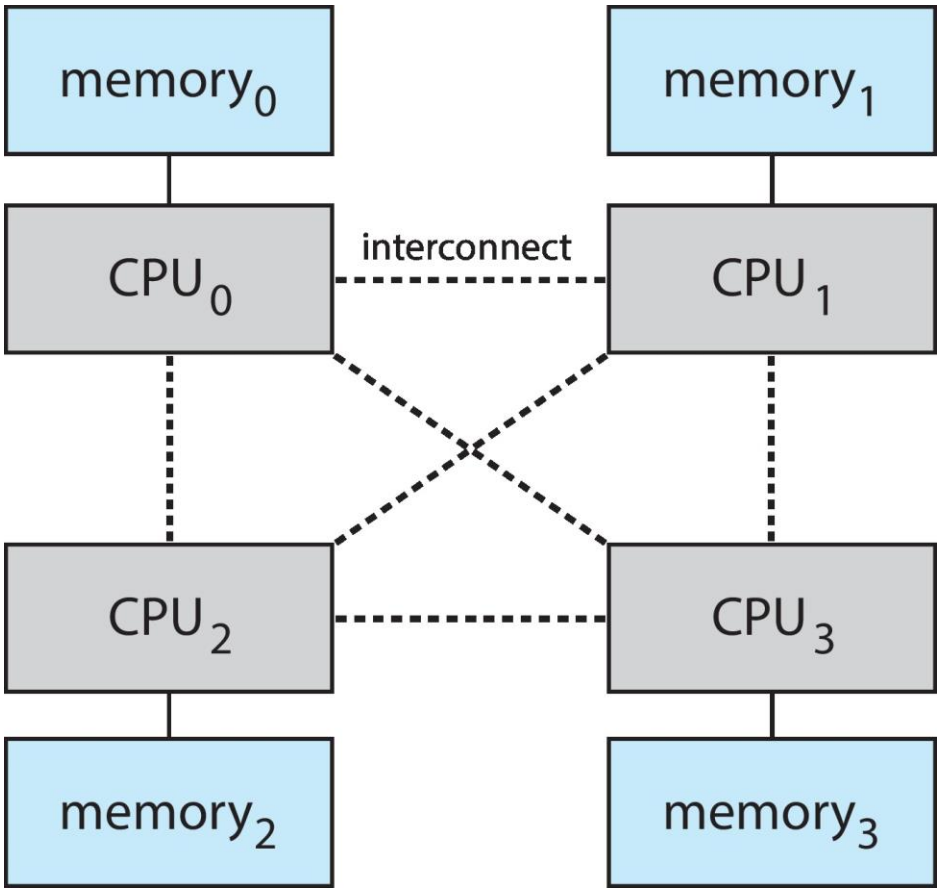


Dual-Core Design

- Multi-chip and **multicore**
- Systems containing all chips
  - Chassis containing multiple separate systems



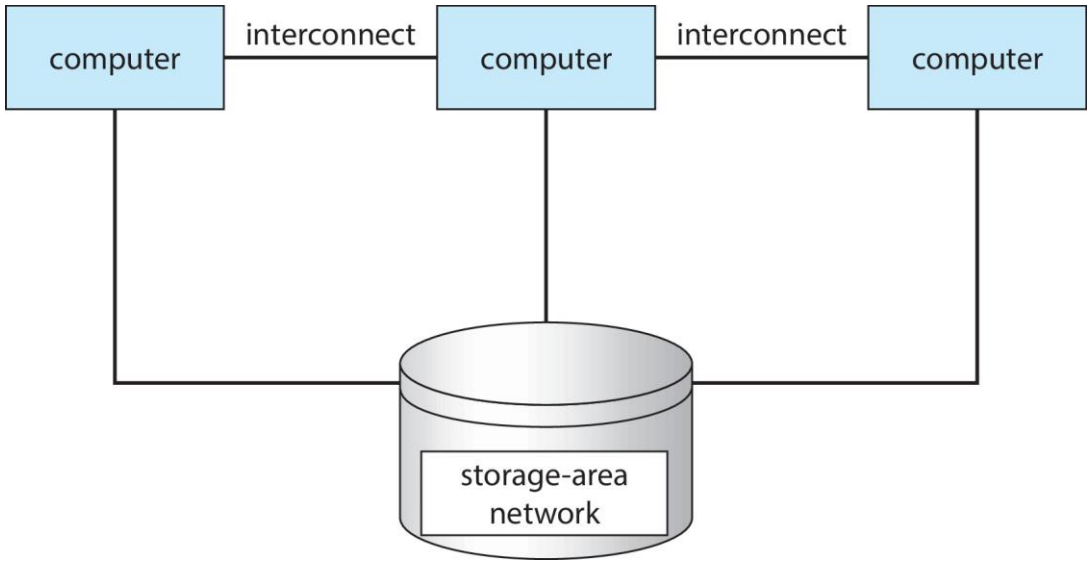
Non-Uniform Memory Access System



Clustered Systems

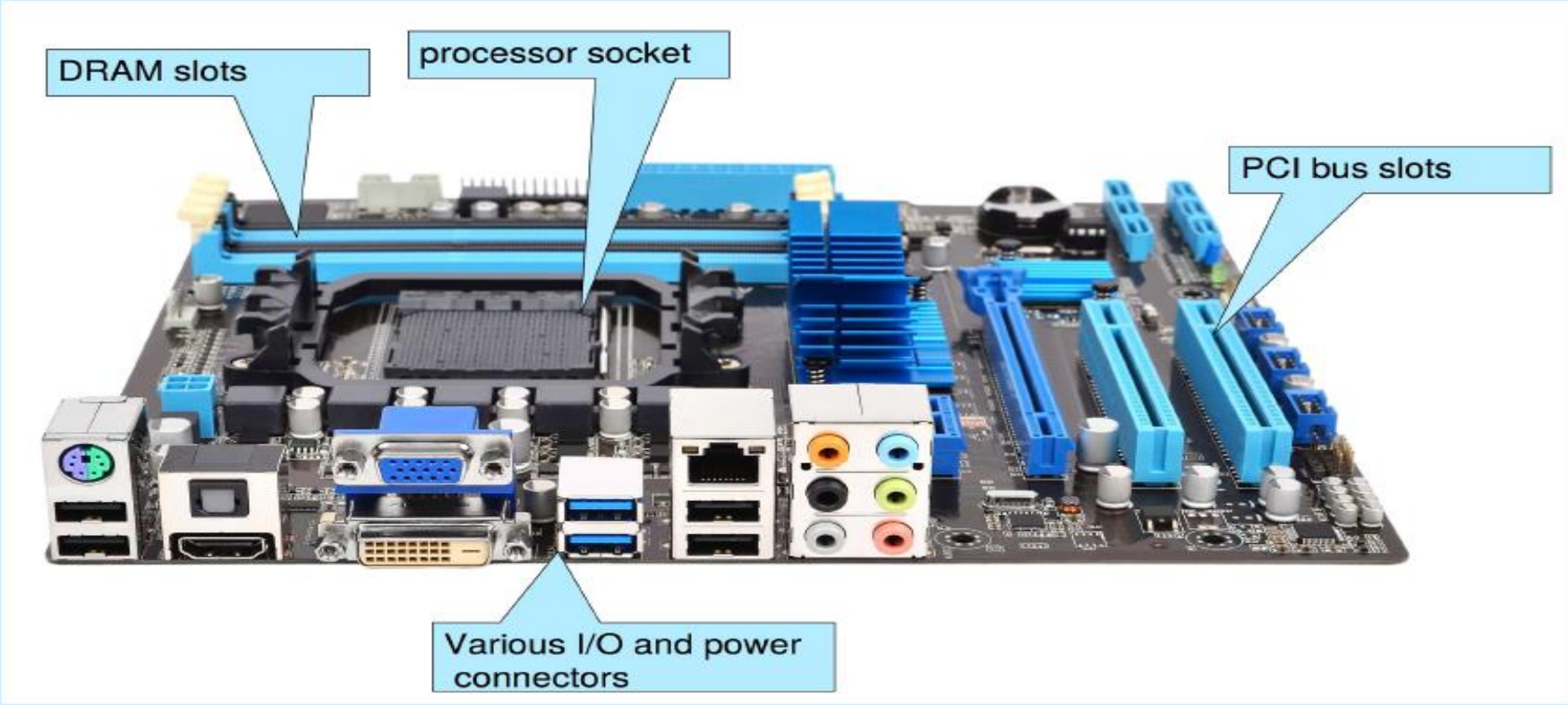
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network** (**SAN**)
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** 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

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.

Computer System Environments

Computing Environments

- Traditional
- Mobile
- Client Server
- Pear-to-Pear
- Cloud computing
- Real-time Embedded

Traditional

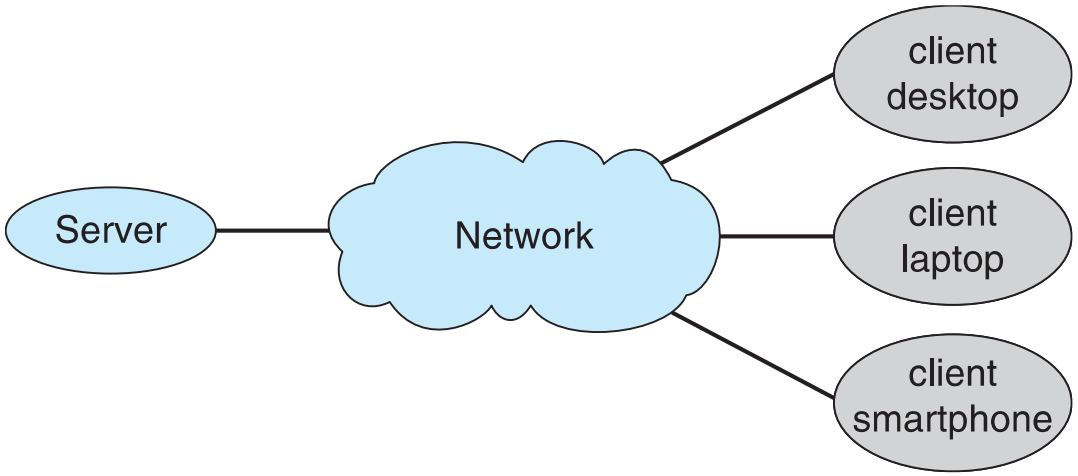
- Stand-alone general-purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- **Portals** provide web access to internal systems
- **Network computers** (**thin clients**) are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

Mobile Computing

- 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 IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

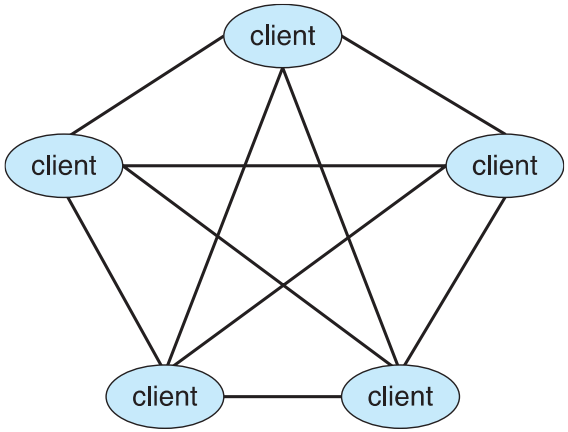
Client Server Computing

- 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



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

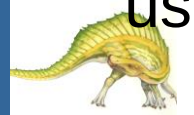


Cloud Computing

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

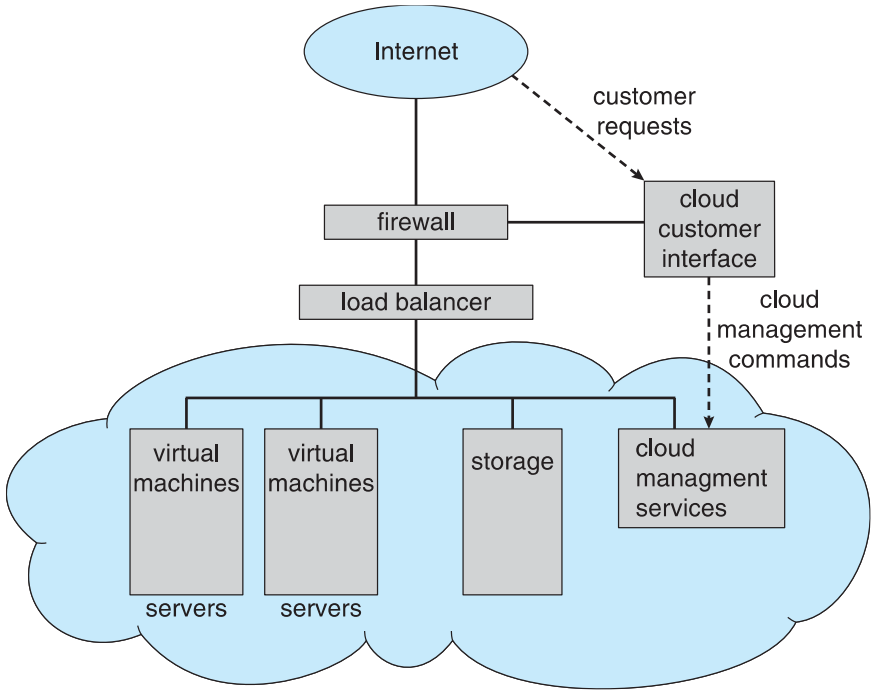
Cloud Computing – Many Types

- **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
- Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
- Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
- Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)



Cloud Computing (cont.)

- 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



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

Free and Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source** and **proprietary**
- Counter to the **copy protection** and **Digital Rights Management** (**DRM**) movement
- Started by **Free Software Foundation** (**FSF**), which has “copyleft” **GNU Public License** (**GPL**)
  - **Free software and open-source software** are two different ideas championed by different groups of people
    - http://gnu.org/philosophy/open-source-misses-the-point.html/

Free and Open-Source Operating Systems

- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

The Study of Operating Systems

- The open-source movement has overtaken operating systems, causing many of them to be made available in both source and binary (executable) format.
- The list of operating systems available in both formats includes Linux, BUSD UNIX, Solaris, and part of macOS.
- The availability of source code allows us to study operating systems from the inside out. Questions that we could once answer only by looking at documentation or the behavior of an
- operating system we can now answer by examining the code itself.
- An extensive but incomplete list of open-source operating-system projects is available from https://curlie.org/Computers/Software/Operating_Systems/Open_Source/

- In addition, the rise of virtualization as a mainstream (and frequently free) computer function
- makes it possible to run many operating systems on top of one core system.

- **The advent of open-source operating systems has also made it easier to make the move from student to operating-system developer. With some knowledge, some effort, and an Internet connection, a student can even create a new operating-system distribution.**

End of Chapter 1