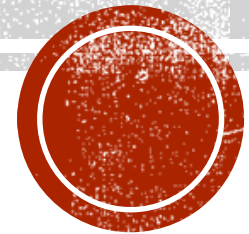


OPERATING SYSTEM

Chapter1_Introduction



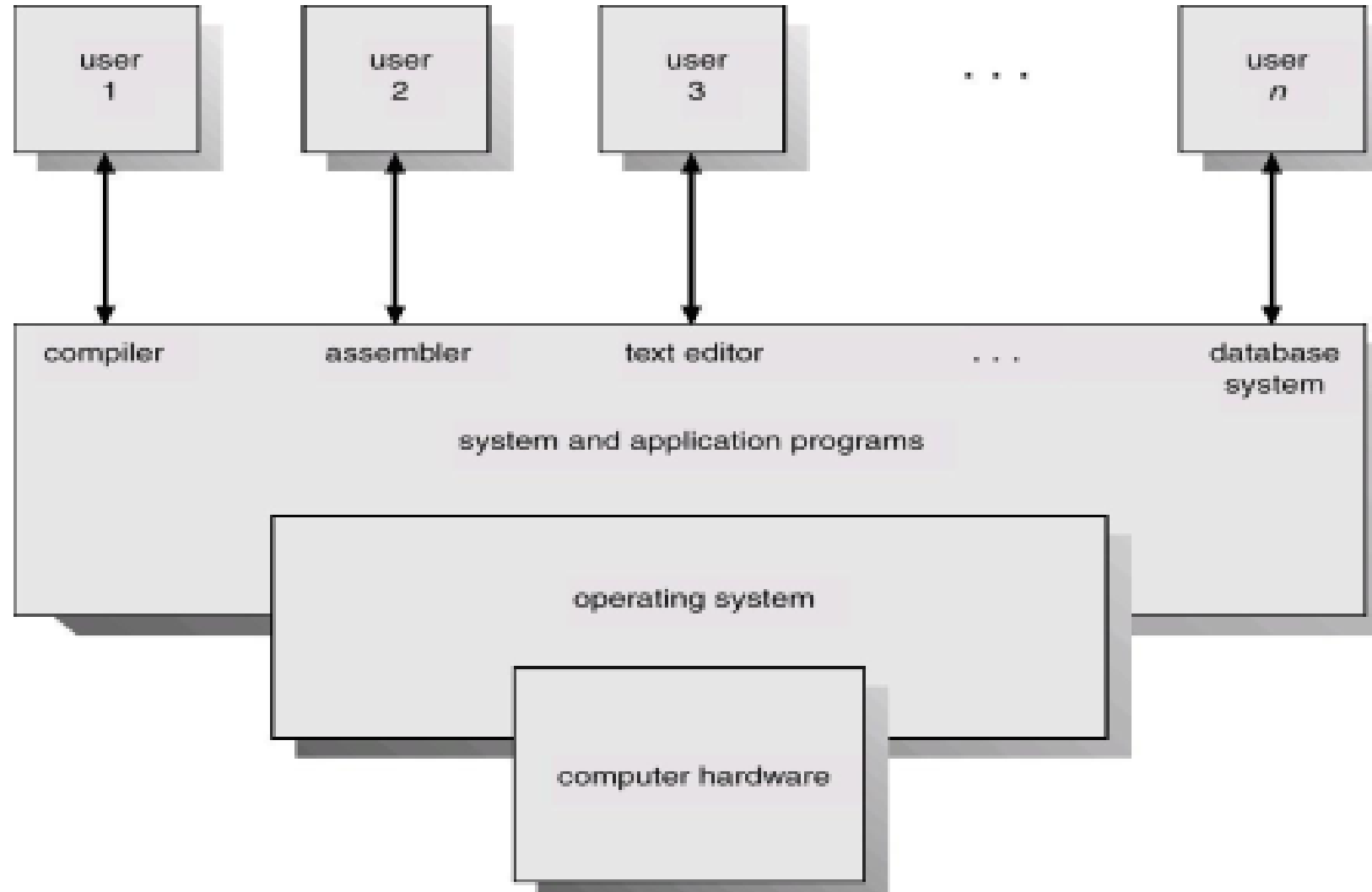
INTRODUCTION

- Computer system
- What is an operating system?
- Variations of OS
- Computer System Operation
- Memory Hierarchy
- Hardware Protection
- Common OS Concepts
- System Calls



COMPUTER SYSTEM

- Computer system provide a capability for ***gathering data, performing computations, storing information, communicating with other computer system and generating output***



*Abstract view of
computer system*



COMPUTER SYSTEM

- **1. Hardware** – provides basic computing resources (CPU, memory, I/O devices).
- **2. Operating system** – controls and coordinates the use of the hardware among the various application programs for the various users.
- **3. Applications programs** – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
- **4. Users** (people, machines, other computers).



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



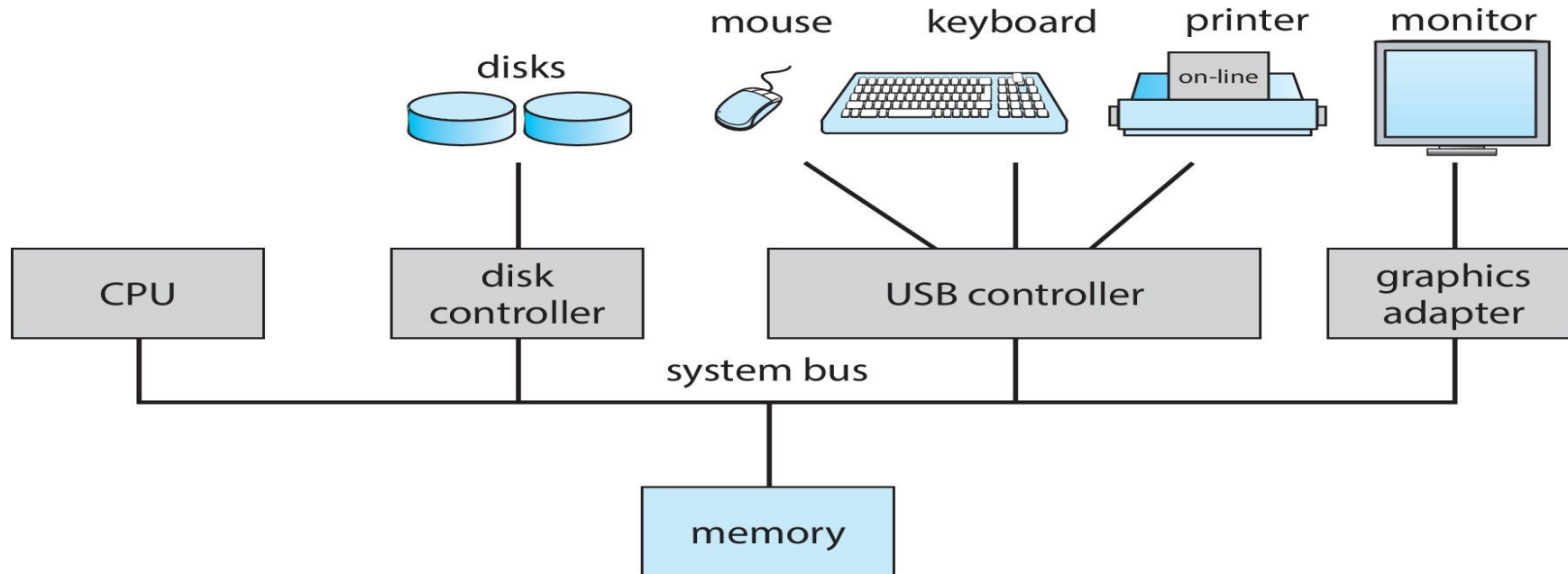
OPERATING SYSTEM DEFINITION

- 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 OSe for general purpose and mobile computing also include **middleware** – a set of software frameworks that provide addition services to application developers such as databases, multimedia, graphics



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
- 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**

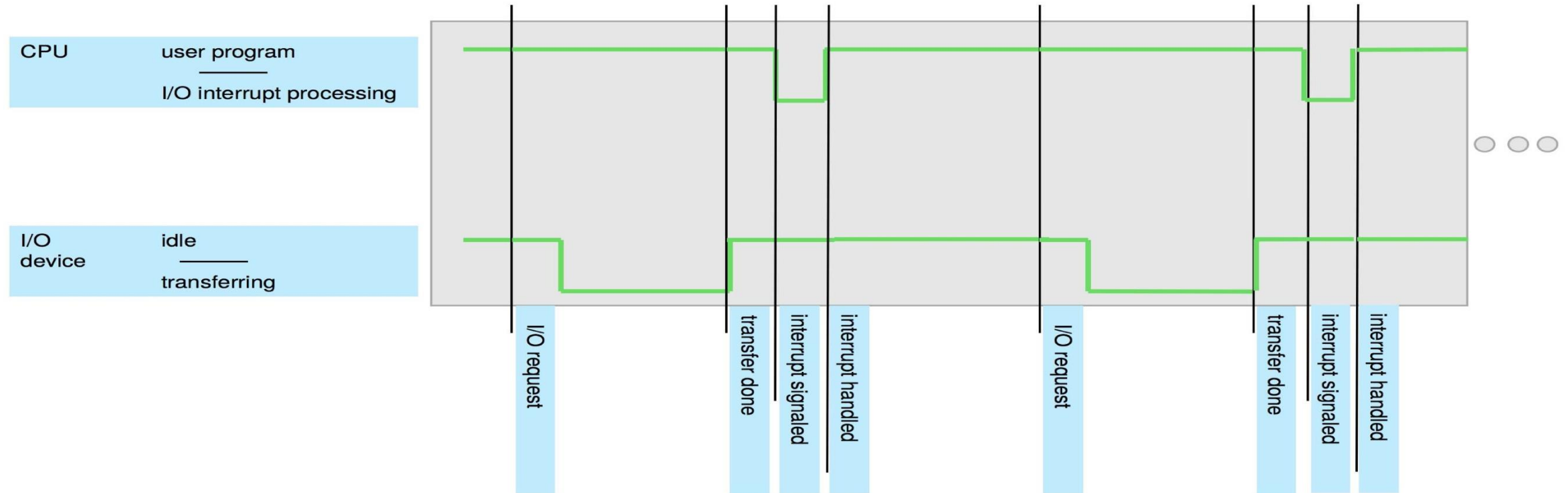


COMMON FUNCTIONS OF INTERRUPTS

- 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

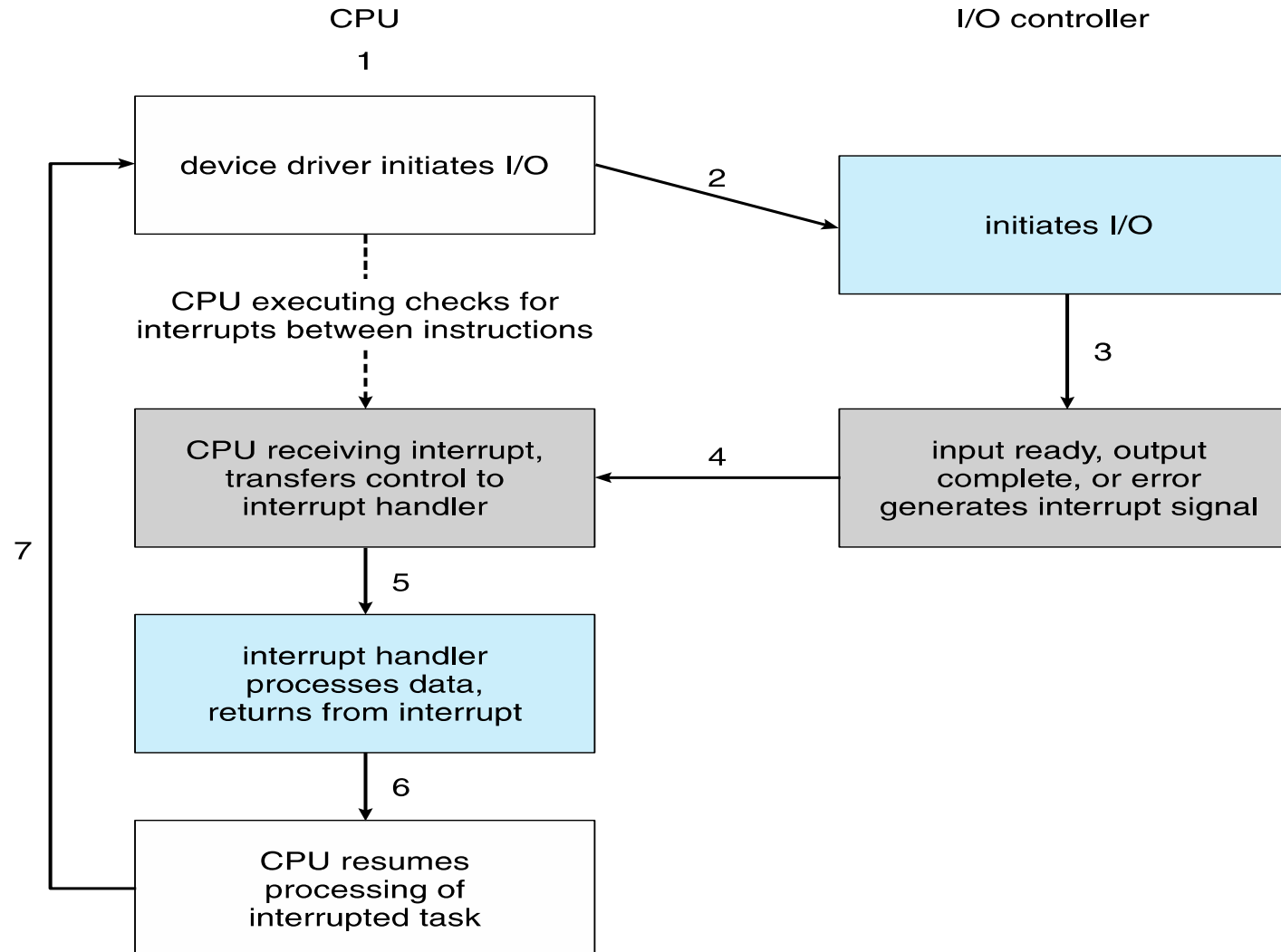


INTERRUPT HANDLING

- 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-DRIVE 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



I/O STRUCTURE (CONT.)

- 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



Storage Structure



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



STORAGE STRUCTURE (CONT.)

- **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 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 is amazing how many things a computer can 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. For example, most computers don't have an instruction to move a bit but do have one to move a byte. A less common term is **word**, which is a given computer architecture's native unit of data. A word is made up of one or more bytes. For example, a computer that has 64-bit registers and 64-bit memory addressing typically has 64-bit (8-byte) words. A computer executes many operations in its native word size rather than a byte at a time.

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; and a **petabyte**, or **PB**, is $1,024^5$ bytes. Computer manufacturers often round off these numbers and say that a megabyte is 1 million bytes and a gigabyte is 1 billion bytes. Networking measurements are an exception to this general rule; they are given in bits (because networks move data a bit at a time).

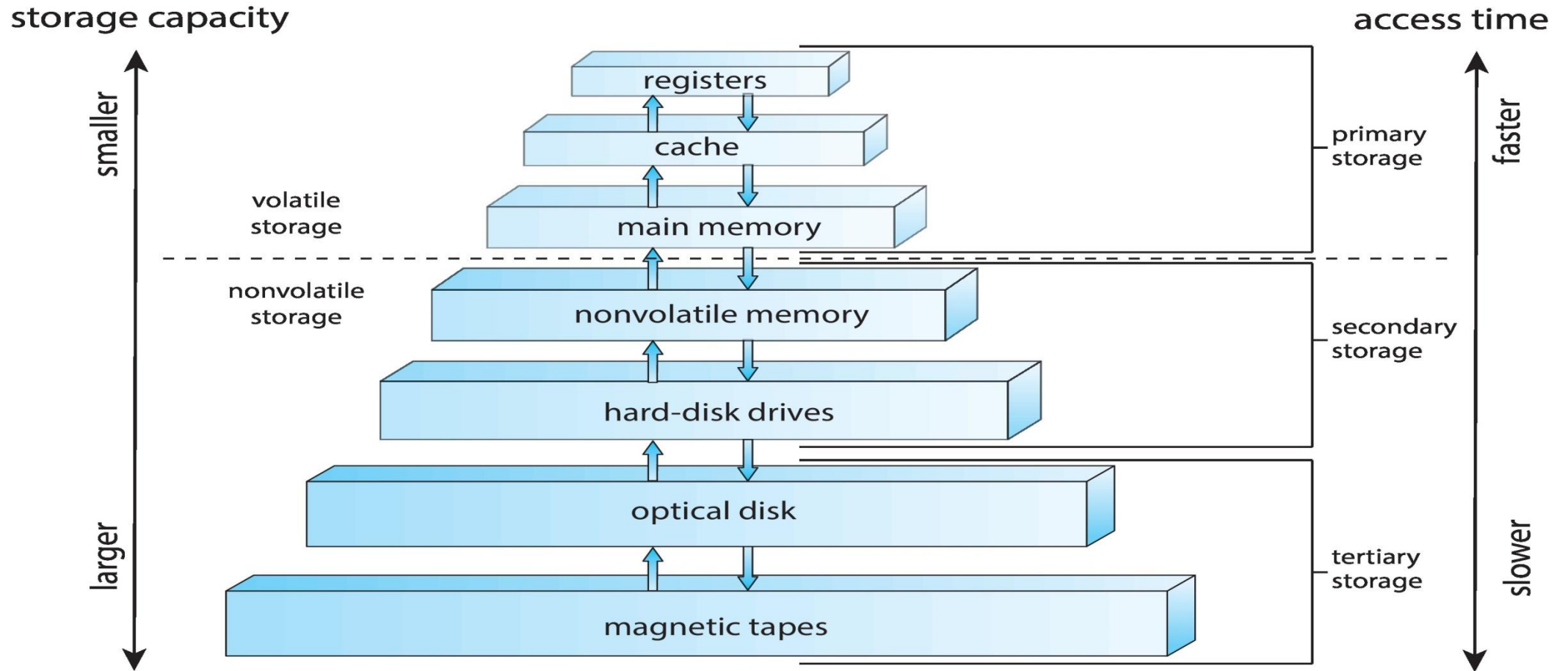


STORAGE HIERARCHY

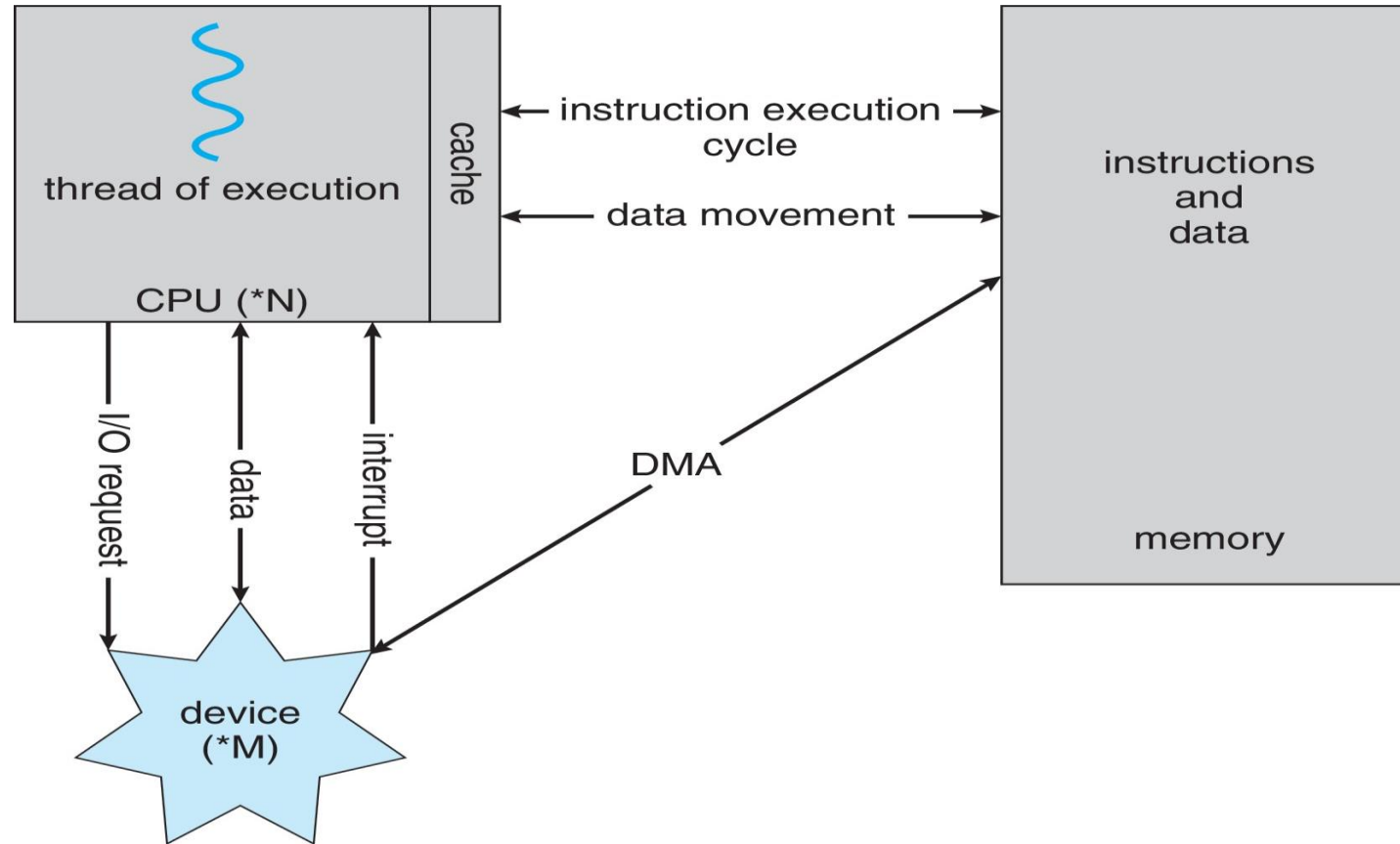
- Storage systems organized in hierarchy
 - Speed
 - Cost
 - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O
 - Provides uniform interface between controller and kernel



STORAGE-DEVICE HIERARCHY



HOW A MODERN COMPUTER WORKS



A von Neumann architecture



DIRECT MEMORY ACCESS STRUCTURE

- 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

- 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

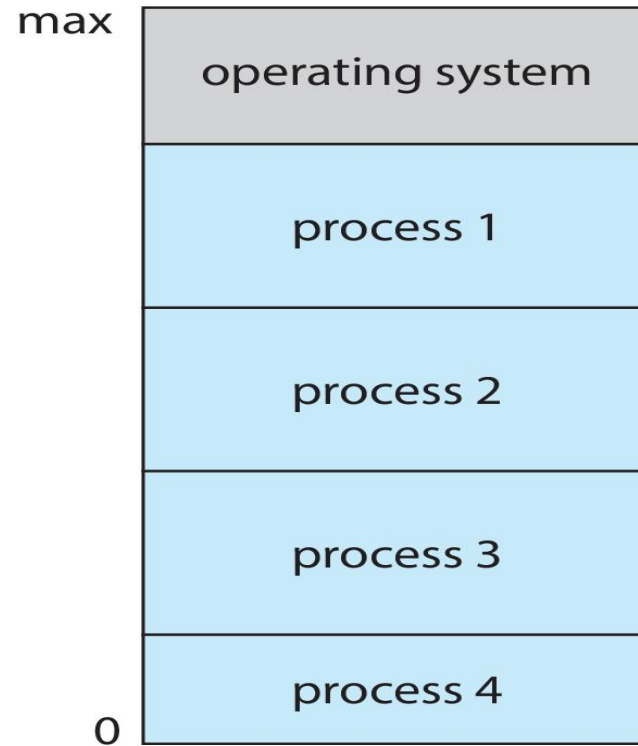


MULTITASKING (TIMESHARING)

- 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 \Rightarrow **process**
 - If several jobs ready to run at the same time \Rightarrow **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

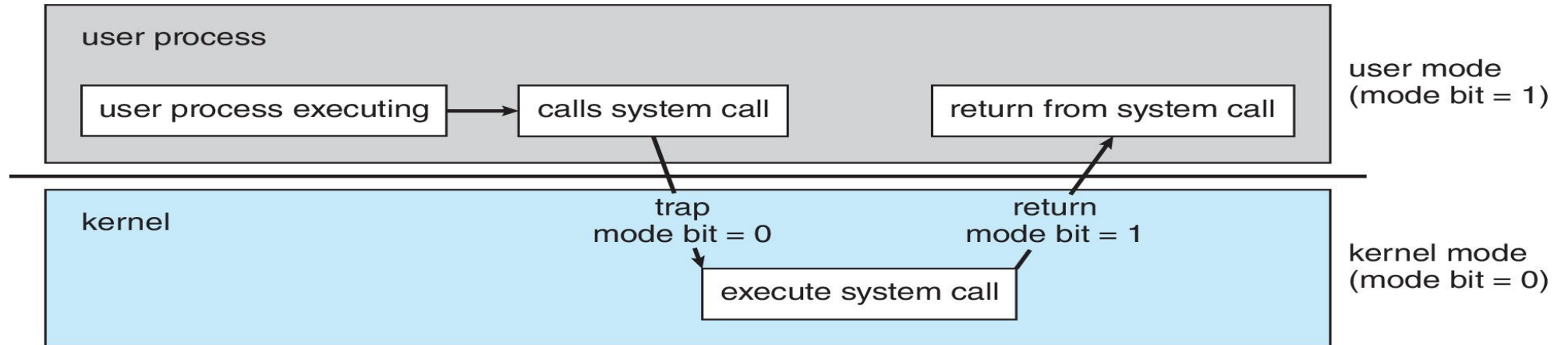


DUAL-MODE OPERATION

- **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



Computer System Architecture

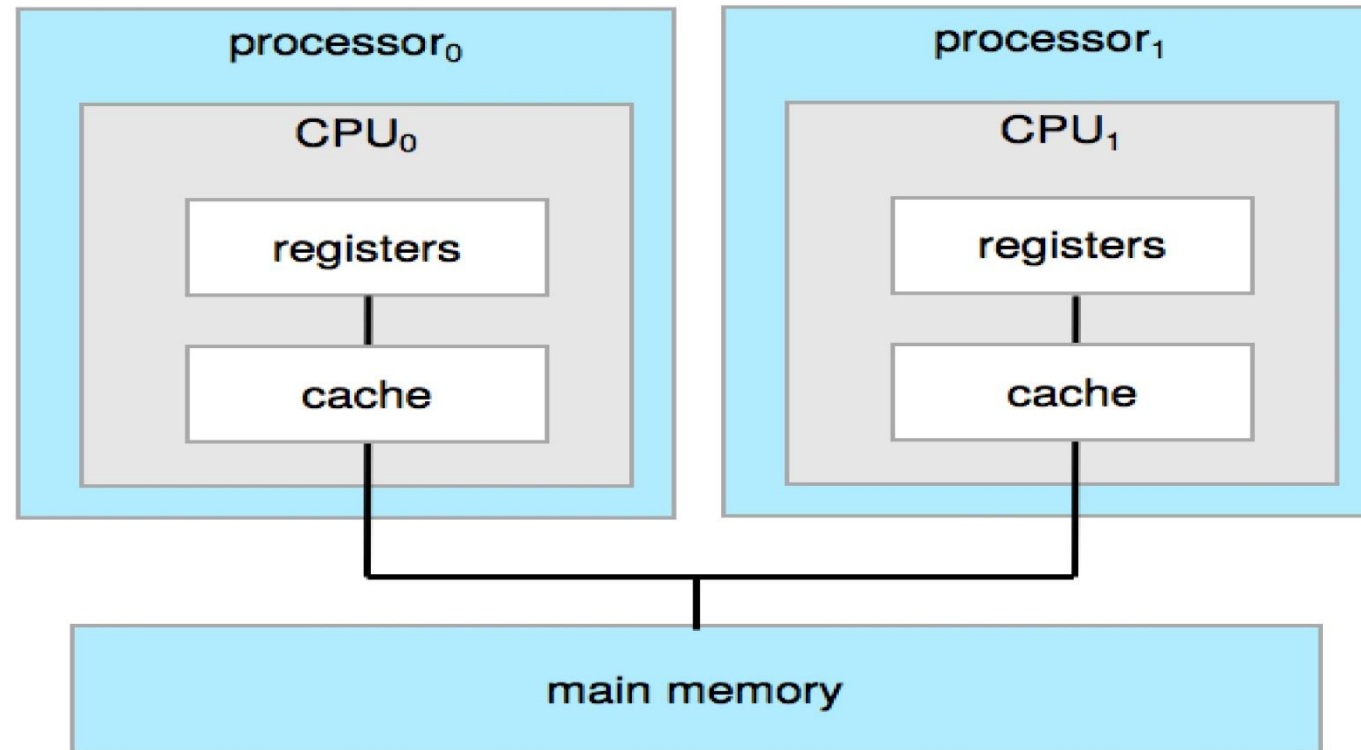


COMPUTER-SYSTEM ARCHITECTURE

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

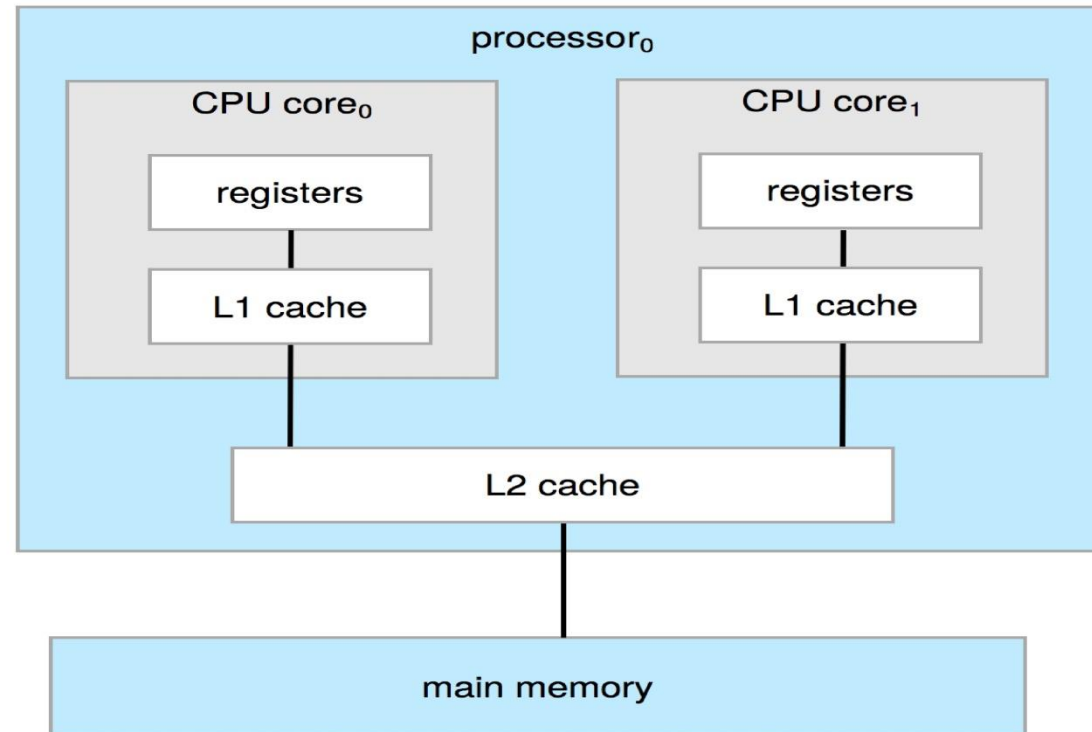


SYMMETRIC MULTIPROCESSING ARCHITECTURE



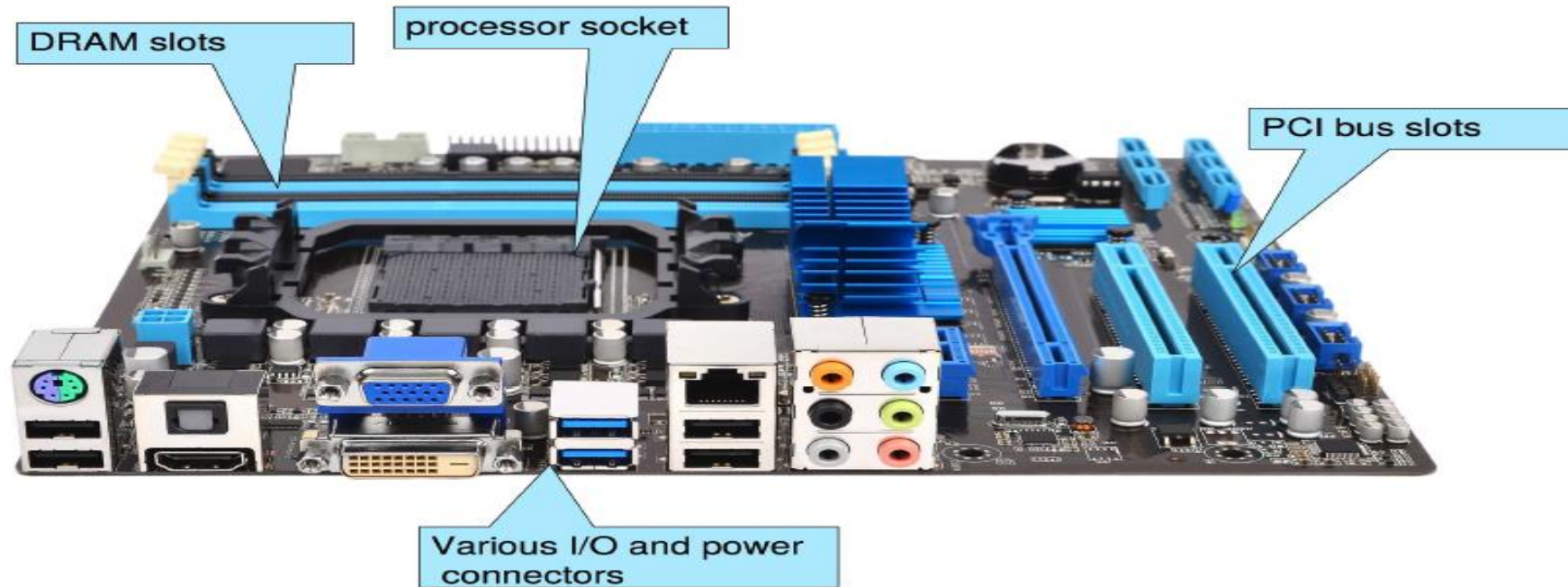
DUAL-CORE DESIGN

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



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/>
- 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

There has never been a more interesting time to study operating systems, and it has never been easier. 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, BSD 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.

Operating systems that are no longer commercially viable have been open-sourced as well, enabling us to study how systems operated in a time of fewer CPU, memory, and storage resources. 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. For example, VMware (<http://www.vmware.com>) provides a free “player” for Windows on which hundreds of free “virtual appliances” can run. Virtualbox (<http://www.virtualbox.com>) provides a free, open-source virtual machine manager on many operating systems. Using such tools, students can try out hundreds of operating systems without dedicated hardware.

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. Just a few years ago, it was difficult or impossible to get access to source code. Now, such access is limited only by how much interest, time, and disk space a student has.



END OF CHAPTER 1

