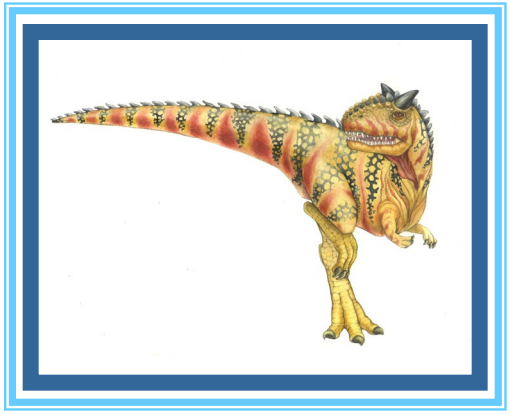
Chapter 1: Introduction (1)



Introduction

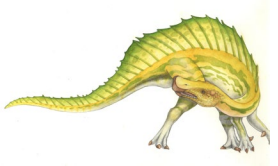
- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Operations
- Resource Management
- Security and Protection
- Virtualization
- Distributed Systems
- Kernel Data Structures
- Computing Environments
- Free/Libre and Open-Source Operating Systems

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 Does Operating System Mean

- An operating system is “fill in the blanks”
- What about:
  - Car
  - Airplane
  - Printer
  - Washing Machine
  - Toaster
  - Compiler
  - …..



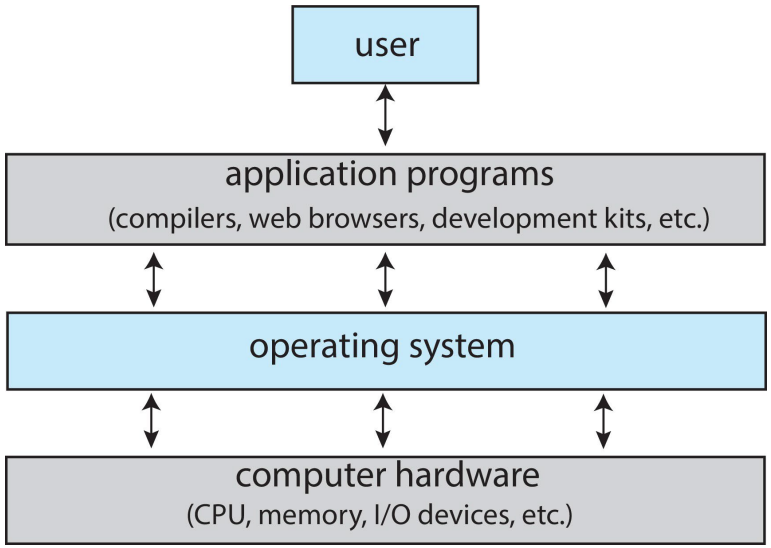
What is an Operating System

- A program that acts as an intermediary between a user of a computer and the computer hardware, including the following functions
  - Process management
  - Memory management
  - File management
  - I/O management

- Operating system goals
  - Execute user programs and make the solving of user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

Computer System Structure

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



What Operating Systems Do

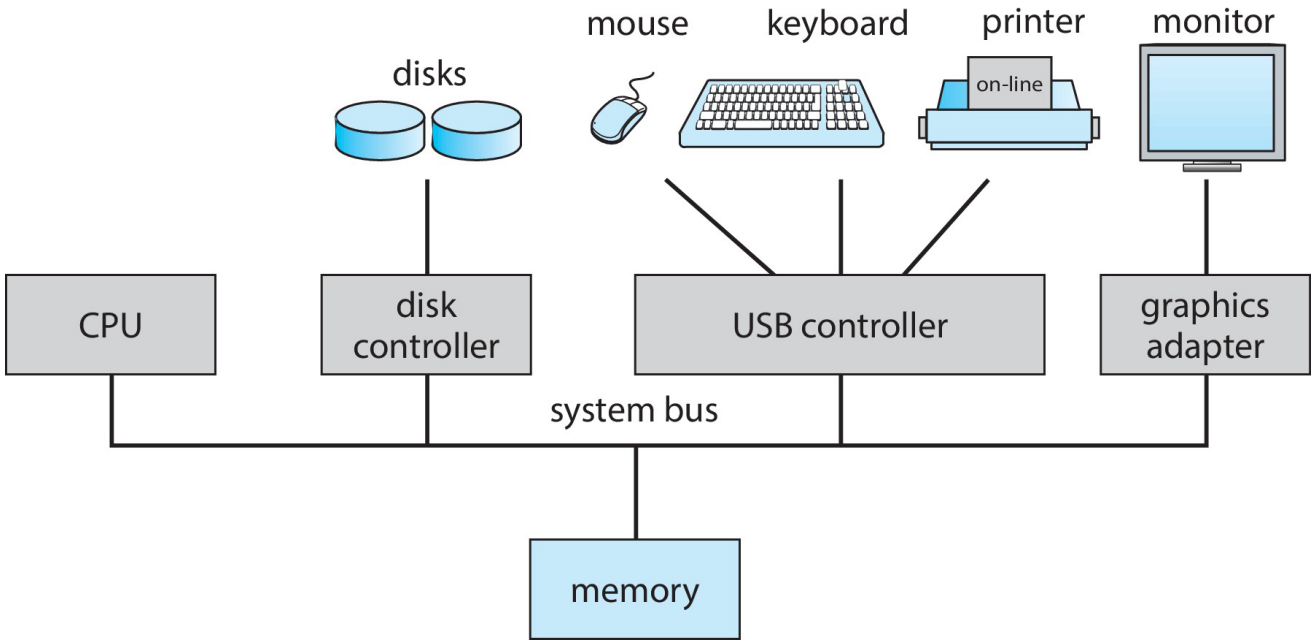
- Depends on the point of view
  - Users want convenience, **ease of use** and **good performance**
    - Don't care about **resource utilization**
  - The shared computer such as **mainframe** or **minicomputer** must keep all users happy
    - OS is a **resource allocator** and **control program** 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 tablets are resource poor, optimized for usability and battery life
    - Mobile user interfaces such as touch screens, voice recognition
  - Some computers have little or no user interface, such as embedded computers in devices and automobiles
    - Run primarily without user intervention

Operating System Definition

- No universally accepted definition
  - Everything a vendor ships when you order an operating system is a good approximation, but varies wildly
- The one program running at all times on the computer is the **kernel,** part of the operating system
- Everything else is either
  - A ***system program*** (ships with the operating system, but not part of the kernel) , or
  - An ***application program***, all programs not associated with the operating system
- Today's OSes for general purpose and mobile computing also include ***middleware*** – a set of software frameworks that provide additional services to application developers such as databases, multimedia, graphics

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

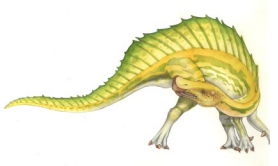


Computer-System Operations

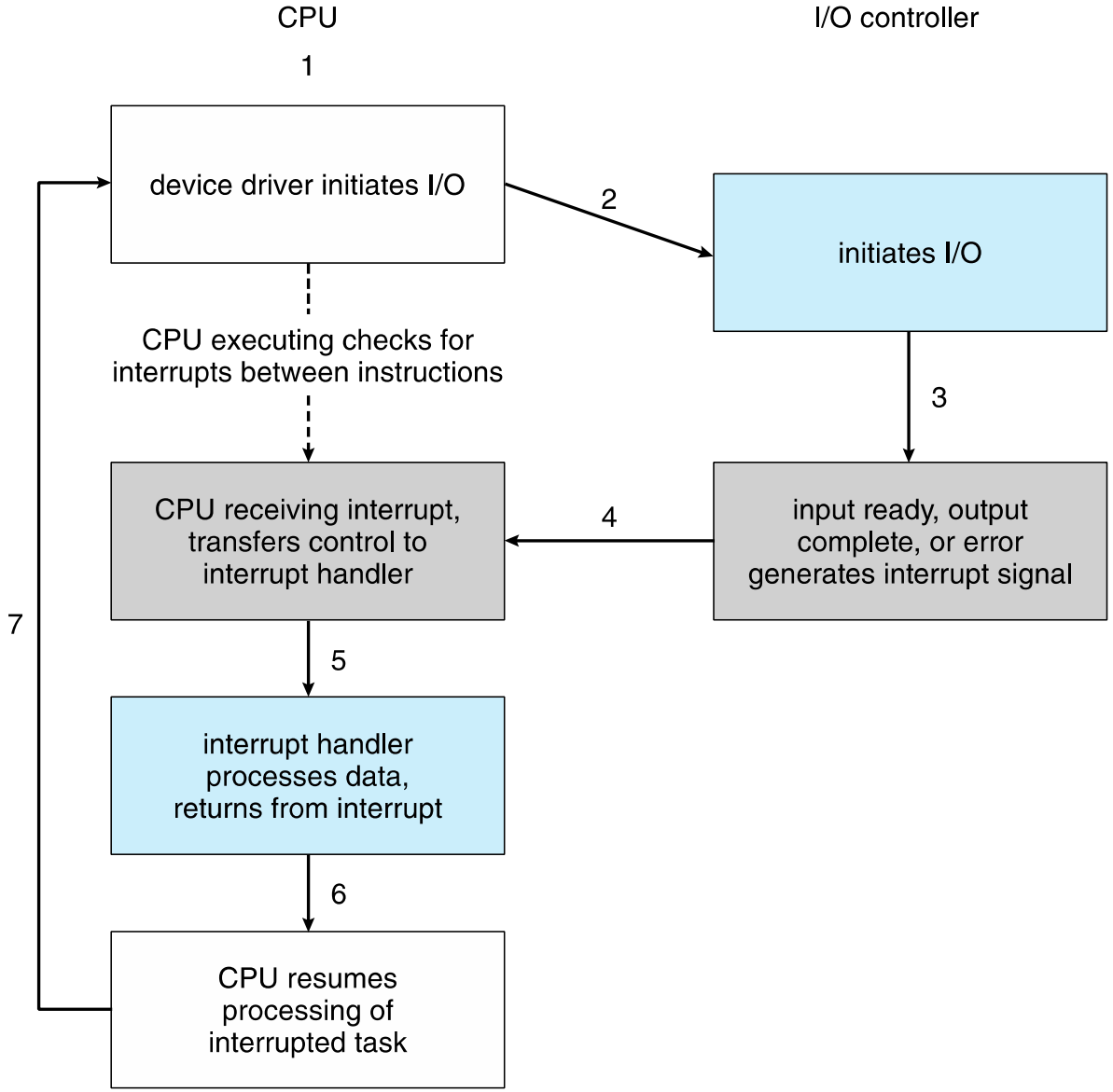
- I/O devices and the CPU can execute concurrently
  - Each device controller is in charge of a particular device type
  - Each device controller has a local buffer
  - Each device controller type has an operating system **device driver** to manage it
  - CPU moves data from main memory to local buffers of device controllers and vice versa
  - 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**

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 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-Drive I/O Cycle

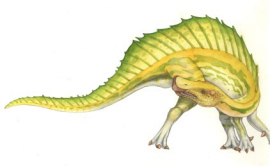


I/O Structure

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

Computer Startup

- **Bootstrap loader** - A small program that places the operating system of a computer into memory
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - When a computer powers up or restarts, the BIOS performs some initial tests and then transfers control to the master book record, where the bootstrap loader resides
    - Initialize all aspects of a system
    - Load operating system kernel and starts execution
  - Most new computers ship with boot loaders for some version of Windows or macOS

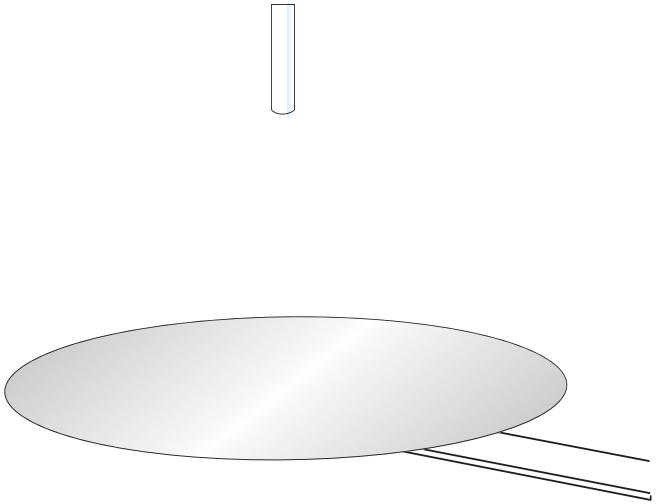


Storage Structure (1)

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - **Volatile**
  - In the form of **Dynamic Random-Access Memory (DRAM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity

Storage Structure (2)

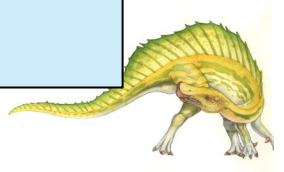
- **Hard Disk Drives** (**HDD**) – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Non-volatile memory** (**NVM**) devices– faster than hard disks, nonvolatile
  - Various technologies
  - Become 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.

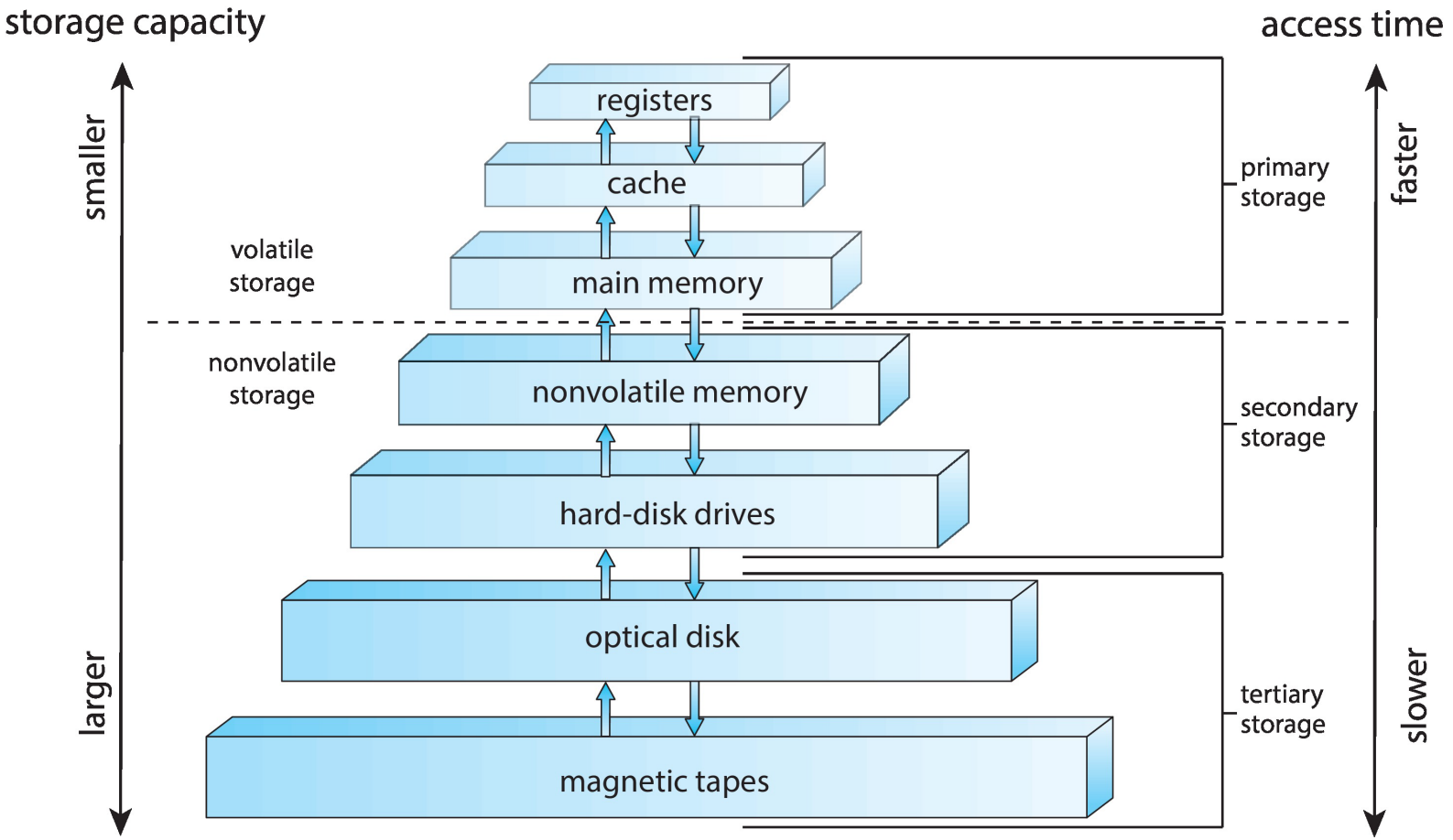
Computer storage, along with most computer throughput, is generally measured and manipulated in bytes and collections of bytes. A **kilobyte**, or KB , is 1,024 bytes; a **megabyte**, or **MB**, is $1,024^2$ bytes; a **gigabyte**, or **GB**, is $1,024^3$ bytes; a **terabyte**, or **TB**, is $1,024^4$ 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).



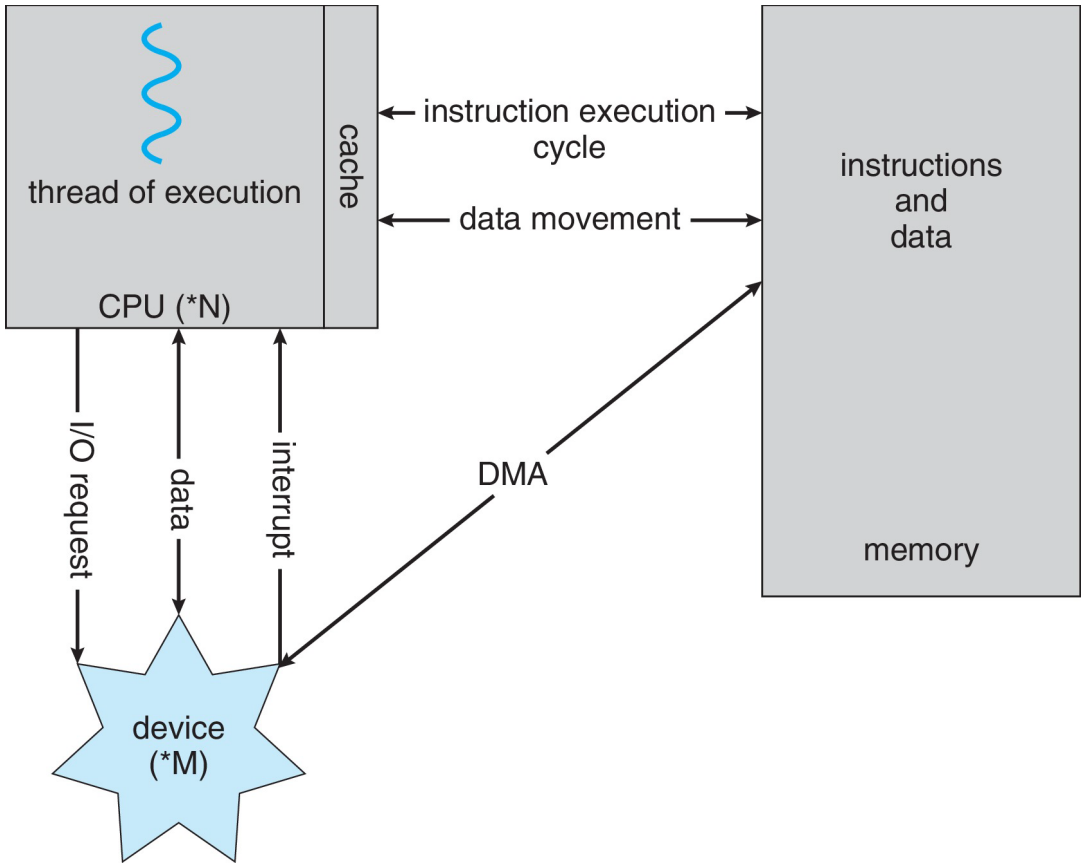
Storage Hierarchy

- 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



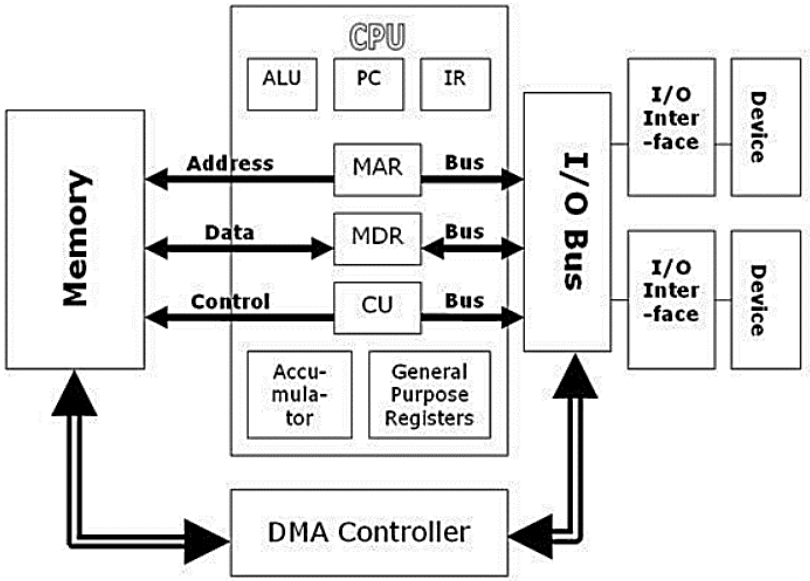
How a Modern Computer Works



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
  - A simple code to initialize the system, and load the kernel
- Starts **system daemons** (services provided outside of the kernel)
- Kernel **interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap**)
    - Software error (e.g., division by zero)
    - Request for operating system service – **system call**
    - Other process problems include infinite loop, processes modifying each other or the operating system

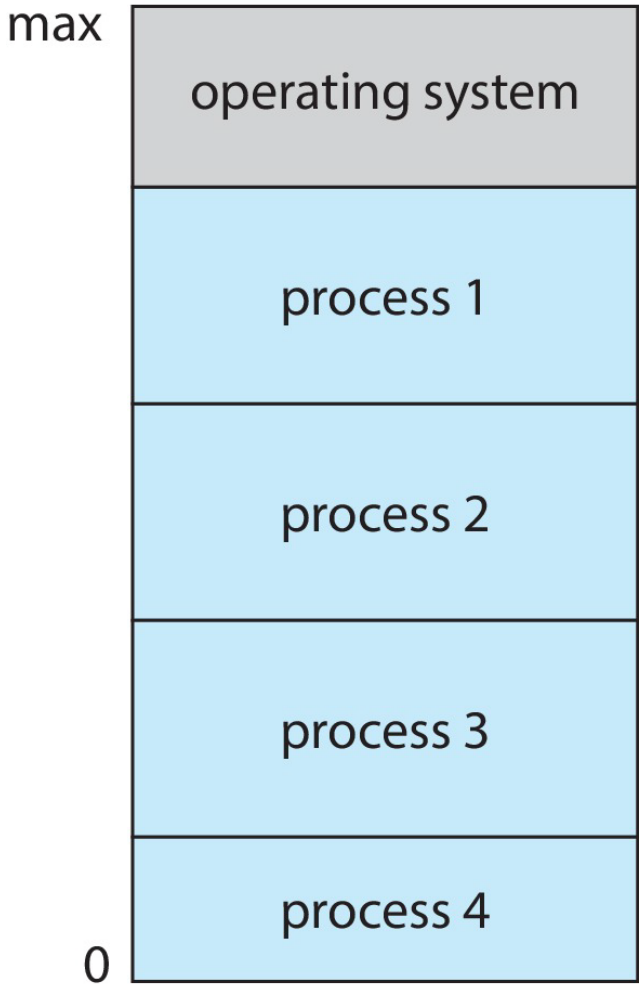
Multiprogramming (Batch System)

- Single user cannot always keep CPU and I/O devices busy
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via **job scheduling**
- When job has to wait (for I/O for example), OS switches to another job

Multitasking (Timesharing)

- A logical extension of Batch systems– the CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨ **process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System



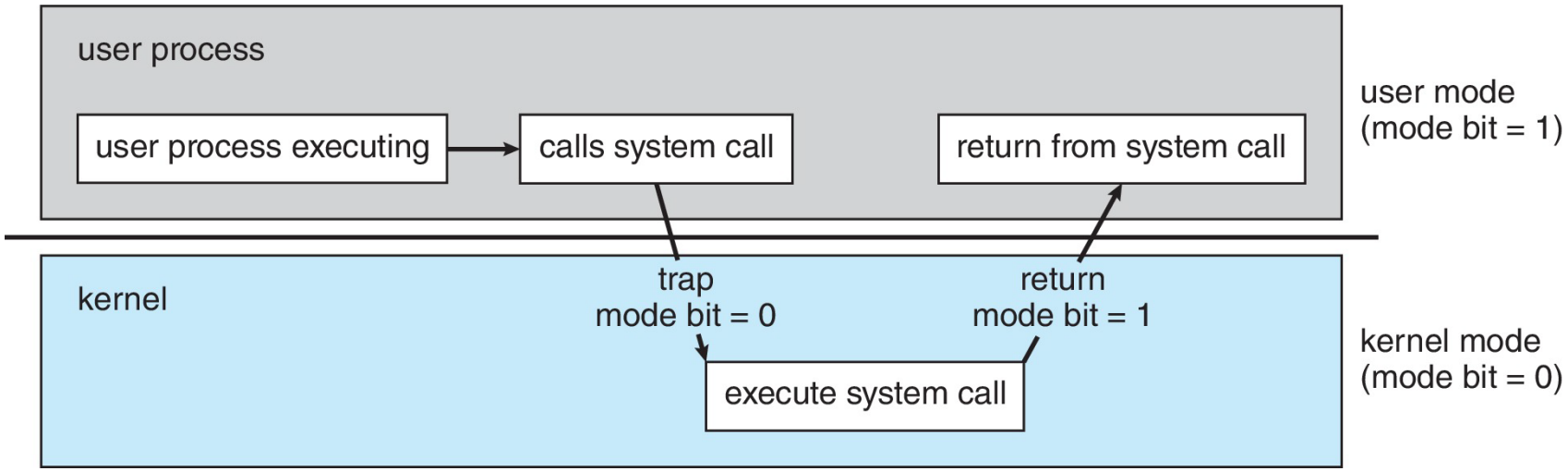
Dual-Mode Operation (1)

- **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”

Dual-Mode Operation (2)

- 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



Timer

- Timer to prevent infinite loop (or process hogging resources)
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time

Take a Break