Introduction

Chapter 1

Basics, Computer-Organization review, Architecture review, Parallel Processing

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

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems

What is an Operating System?

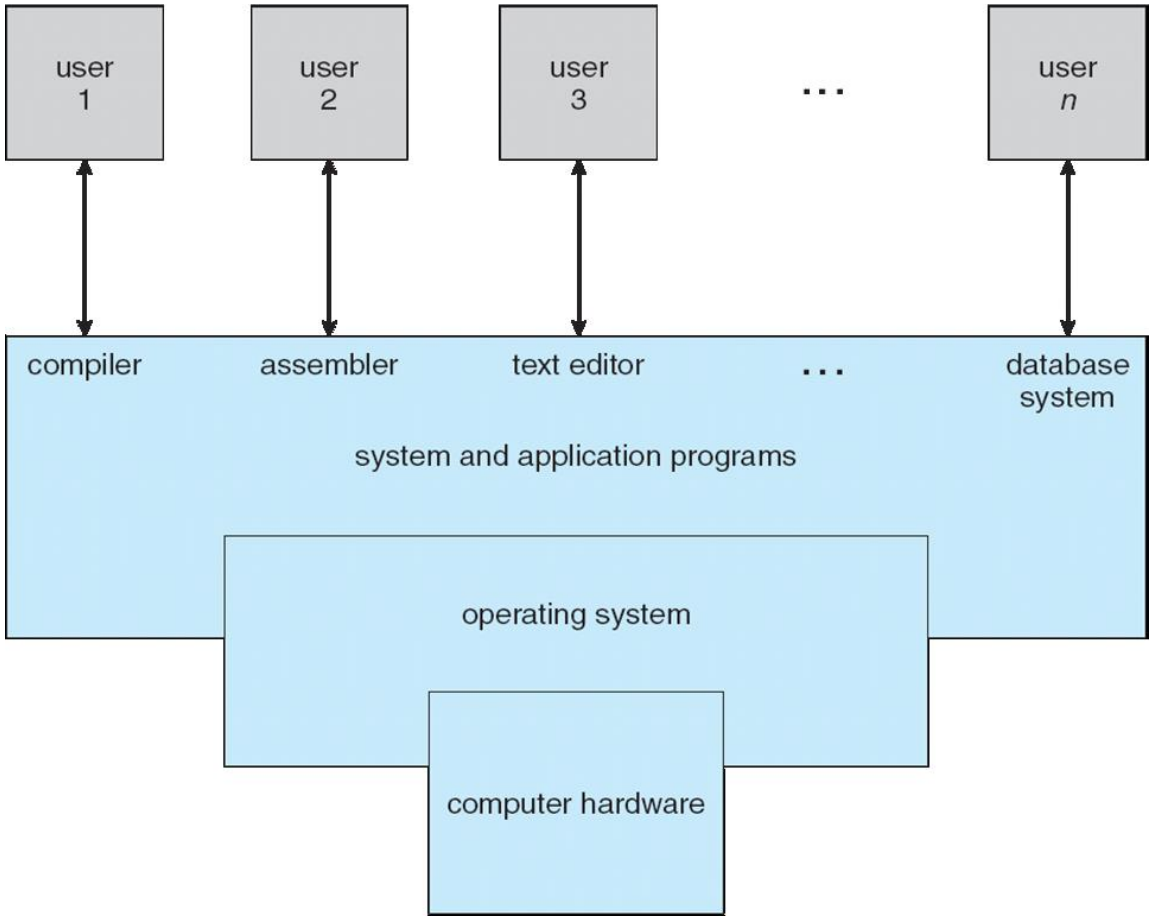
- A program that acts as an **intermediary** between a user of a computer and the computer hardware

- 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

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

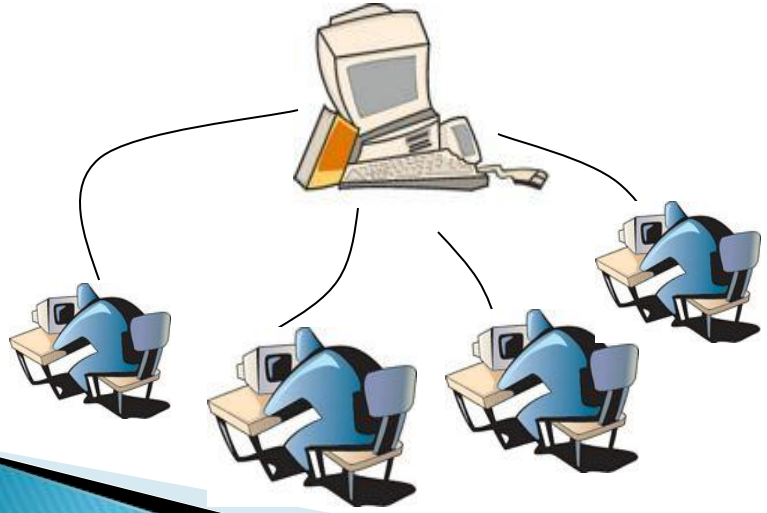


What Operating Systems Do

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



mainframe

What Operating Systems Do

- 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: A computer system has many resources that may be required to solve a problem: CPU time, memory space, file-storage space, I/O devices, and so on. The operating system acts as the manager of these 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.)

- "The one program running at all times on the computer" usually called the **kernel.**
- 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

Computer Startup: How to start everything?

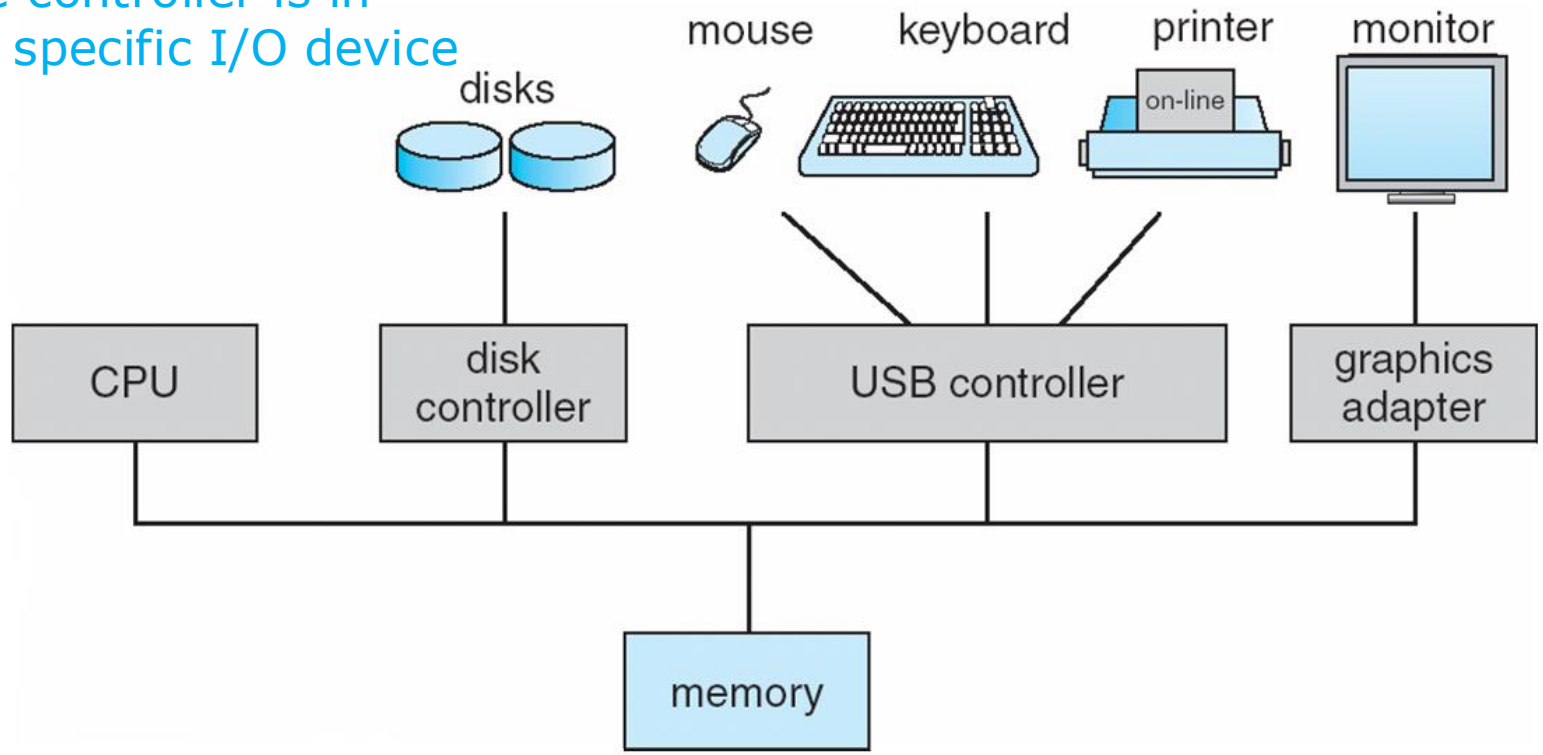
- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM (Erasable Programmable ROM) (why?), generally known as **firmware**
  - Initializes all aspects of system: from CPU registers to device controllers to memory contents.
  - The bootstrap program must know how (how?) to load the operating system and how to start executing the system.

Overview of Computer System Structure

Computer System Organization

- A general computer system consists of a group of modules, (CPU, device controllers, memory & I/O devices) interconnected by a system bus.

Each device controller is in charge of a specific I/O device type



I/O devices

- I/O devices and the CPU can execute concurrently
- Device controller
  - in charge of a particular device type
  - 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 Driver** for each device controller
  - Provides uniform interface between controller and kernel

Common Functions of Interrupts

- Interrupts are essential for the function of any OS. Interrupts from either HW or SW alert the OS when events occur so the CPU can suspend its current activity and deal appropriately with the current situation.

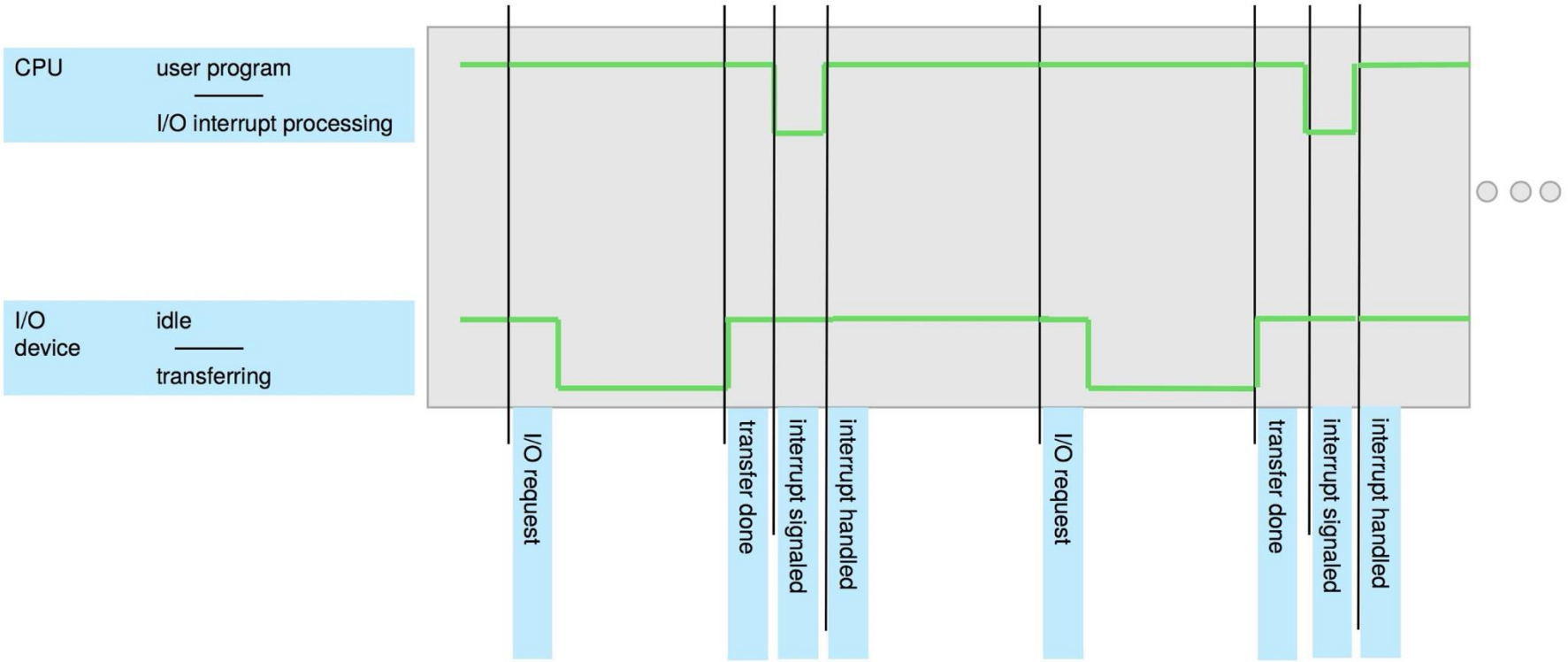
- Interrupts transfer control to a new program called *the interrupt service routine* (ISR) generally, through the *interrupt vector*, which contains the addresses of all the service routines (ISRs).
- Interrupt architecture must save the address of the interrupted instruction.
- Current operating systems are *interrupt driven*– OS waits for an event.
- Events are signaled by the occurrence of an interrupt or a trap.
- A *trap* / *Exception* is a software-generated interrupt caused either by an error or a user request.

Interrupt Handling

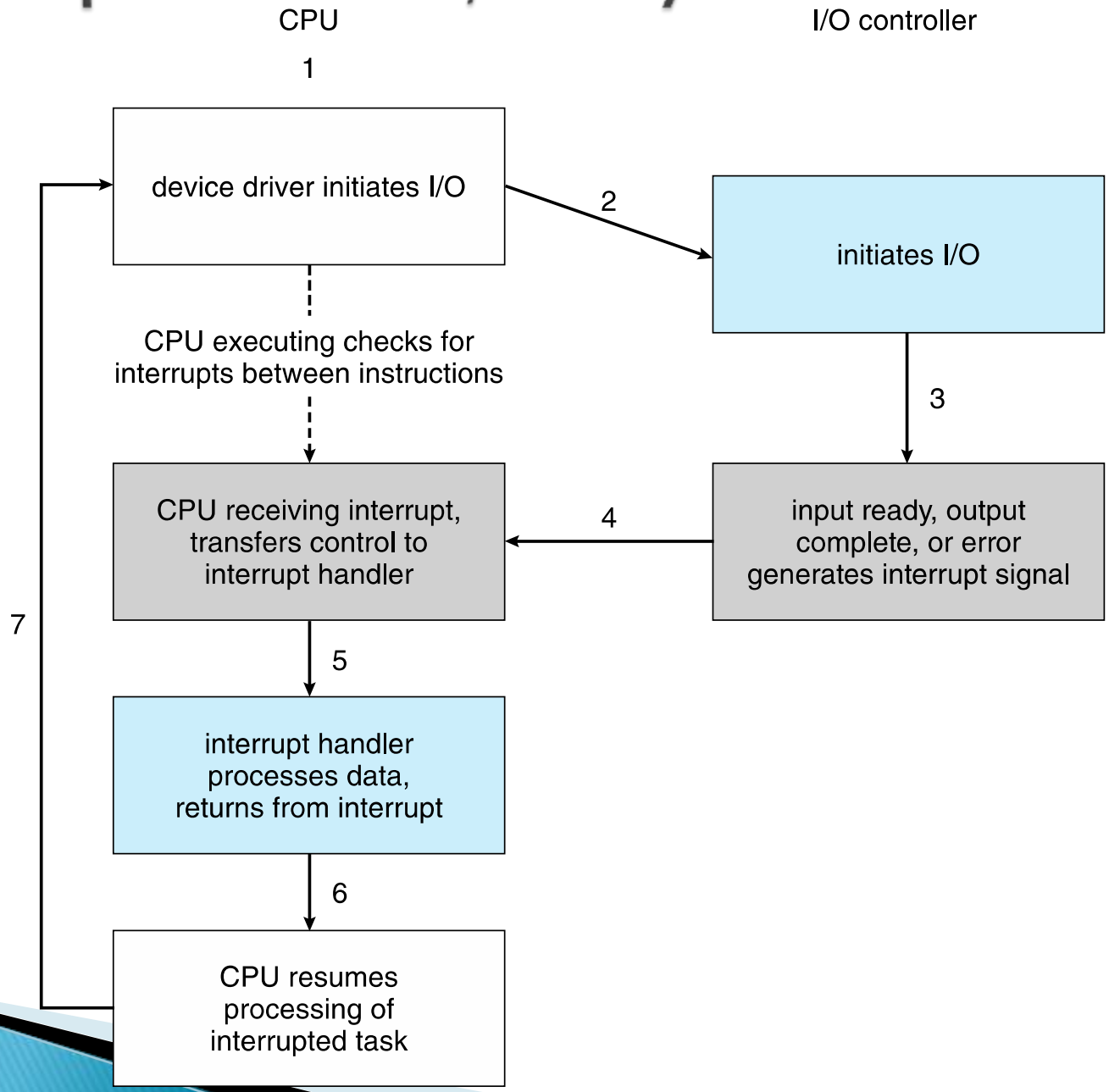
- The operating system preserves the state of the CPU by storing registers and the program counter.
- The OS determines which type of interrupt has occurred by one of two methods:
  - *Polling*– querying of all I/O devices, detect which requested services interrupted the service.
  - *vectored interrupt system* (Table of addresses)

- Separate segments of code determine what action should be taken for each type of interrupt.
- The OS executes the sequence of commands associated with the given interrupt.
- The OS recovers the stored information from the original process and continues execution.

Interrupt Timeline for 1 CPU doing output



Interrupt-drive I/O Cycle



Storage Structure

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 (circuit which enables the CPU to communicate with a disk)
- Solid-state disks (SSD)– faster than magnetic disks, nonvolatile
  - Various technologies
  - Becoming more popular

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

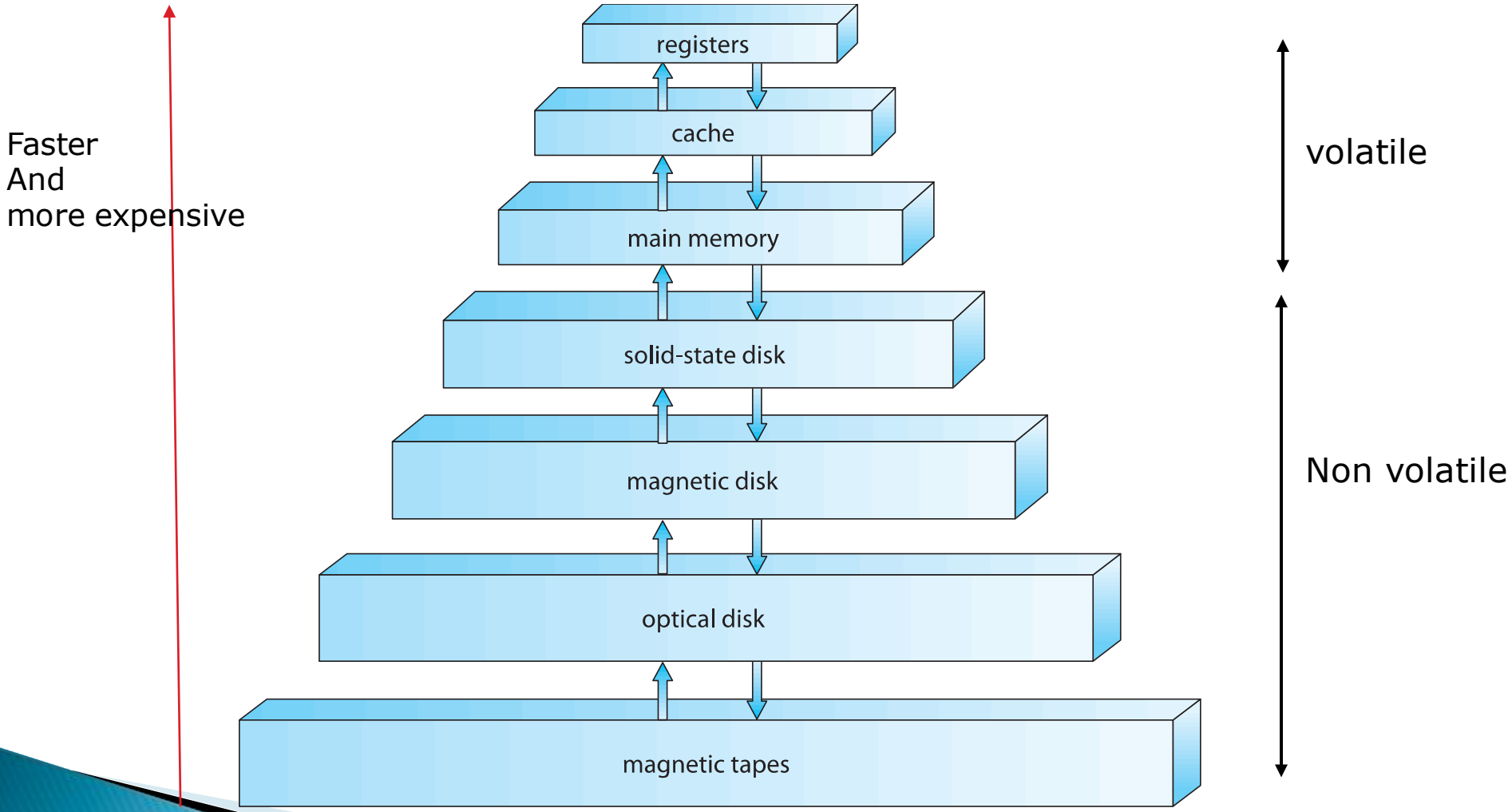
- Caching – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

- Device Driver for each device controller to manage I/O
  - Provides uniform interface between controller and kernel

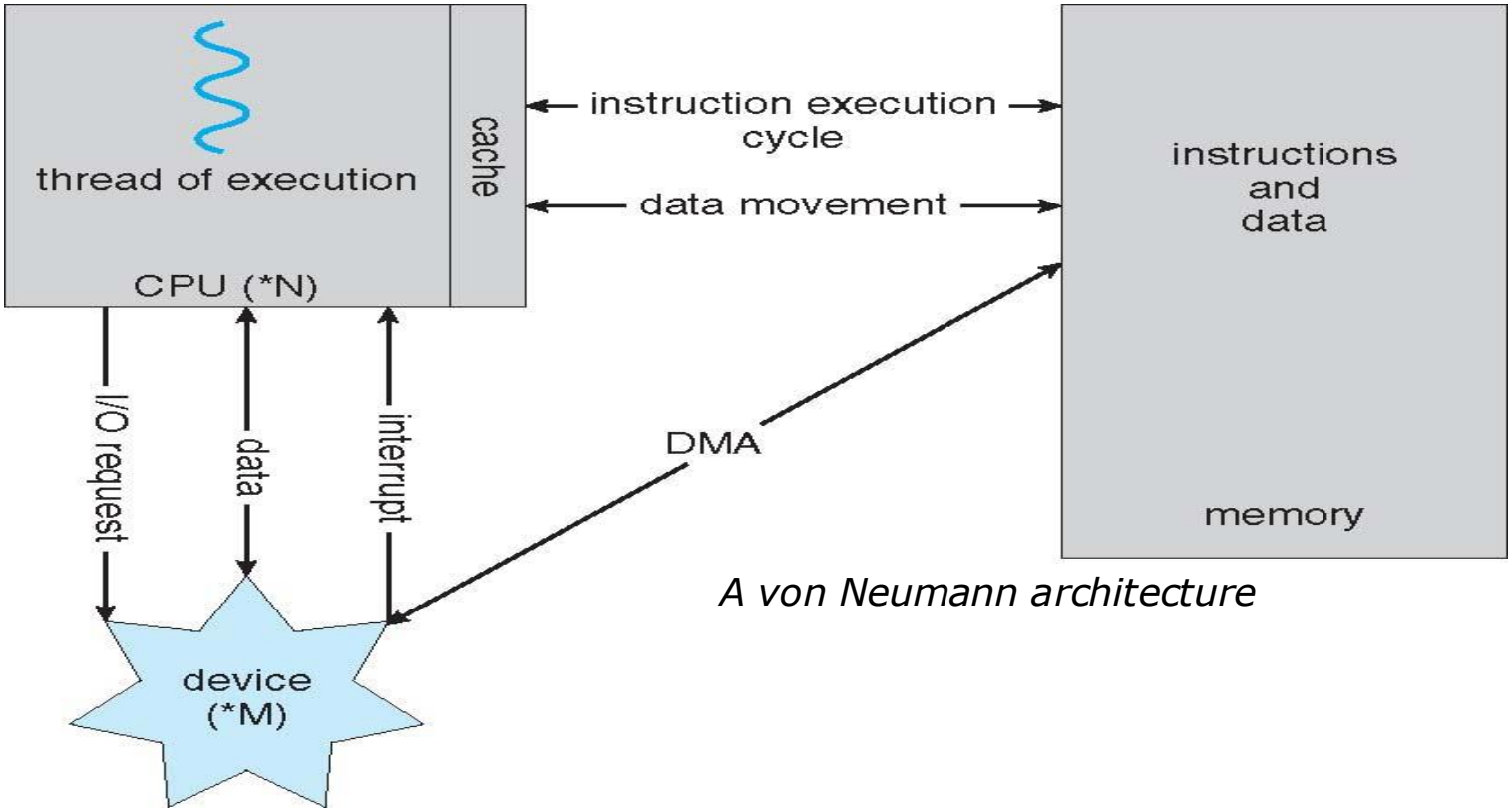
Caching

- Fetching instructions from the RAM is time consuming – slows down the system.
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Idea of caching is to copy instructions and data into the cache for faster access, when a process needs data it first checks and tries to access the cache otherwise it accesses data from the source.
- Cache smaller than storage being cached

Storage-Device Hierarchy



How a Modern Computer Works



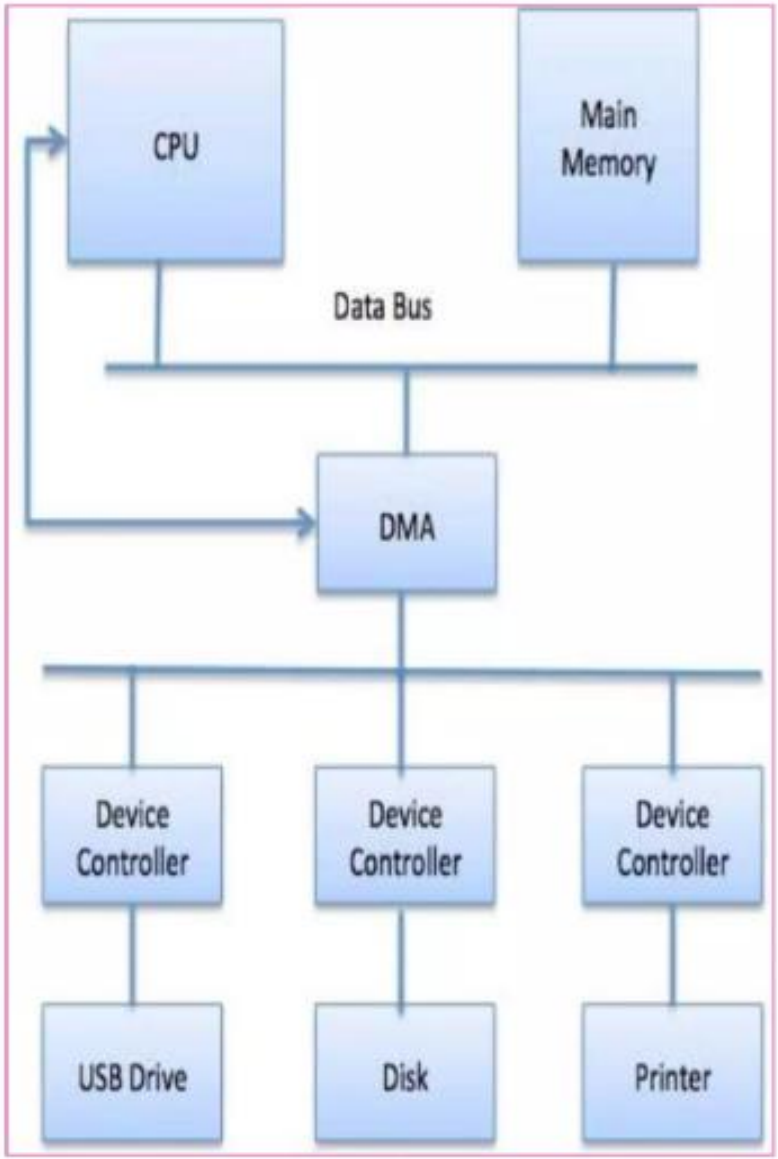
A von Neumann architecture

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

Direct Memory Access (DMA) Structure

- Slow devices like keyboards will generate an interrupt to the main CPU after each byte is transferred. If a fast device such as a disk generated an interrupt for each byte, the operating system would spend most of its time handling these interrupts. So a typical computer uses ***direct memory access (DMA)*** hardware to reduce this overhead.
- Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement. DMA module itself controls exchange of data between main memory and the I/O device. CPU is only involved at the beginning and end of the transfer and interrupted only after entire block has been transferred.



Computer System Architecture

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.

Computer-System Architecture

- **Single processor** systems
  - Most systems have special-purpose processors as well (device-specific processor, i.e. disk. KB…) run a limited instruction set & do not run user processes
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput** – more work done in less time
    2. **Economy of scale** – cost less than multiple single-processors as they share peripherals, power supplies, storage.
    3. **Increased reliability – graceful degradation** or **fault tolerance**
  - Two types:
    1. **Asymmetric Multiprocessing**
    2. **Symmetric Multiprocessing**

Multiprocessor Systems

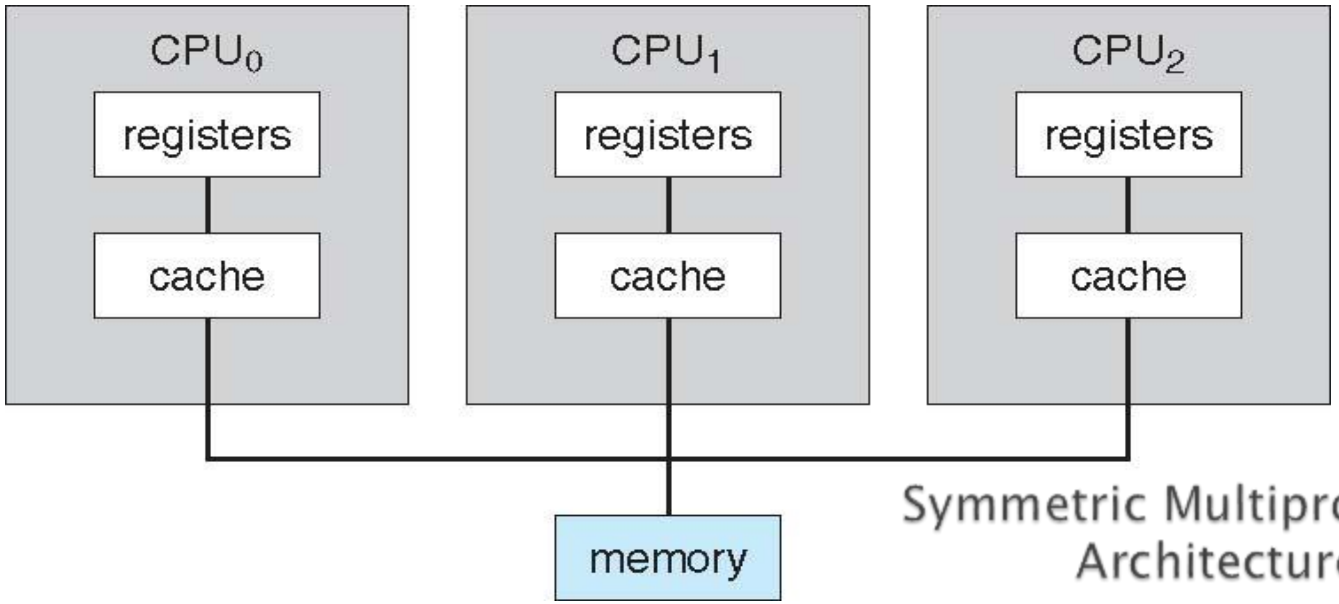
Within the past several years, **multiprocessor systems** (also known as **parallel systems** or **multicore systems**) have begun to dominate the landscape of computing. Such systems have two or more processors in close communication, sharing the computer bus and sometimes the memory, and peripheral devices.

Multiprocessor systems have three main advantages:

- Increased throughput: By increasing the number of processors, we expect to get more work done in less time
- Economy of scale: Multiprocessor systems can cost less than equivalent multiple single-processor systems
- Increased reliability: If functions can be distributed properly among several processors, then the failure of one processor will not halt the system, only slow it down.

Types of Multiprocessor Systems

- Symmetric multiprocessing
  - Each processor performs all tasks within OS
  - No master-slave relationship exists between processors

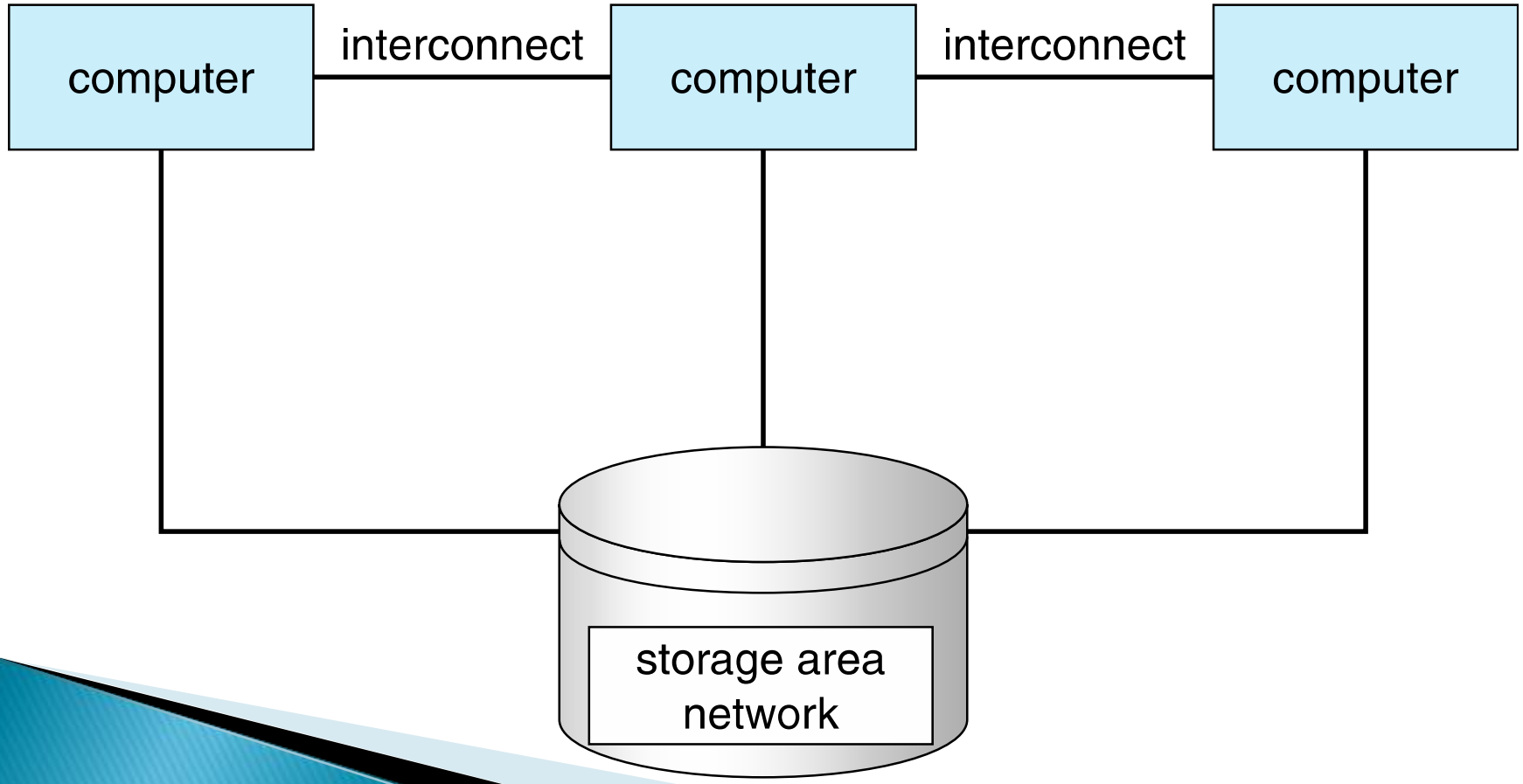


Symmetric Multiprocessing Architecture

- Asymmetric multiprocessing
  - Master processor controls and allocates work to the slave processors
  - More common in extremely large systems

Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)** & closely linked via a **LAN**



Clustered Systems

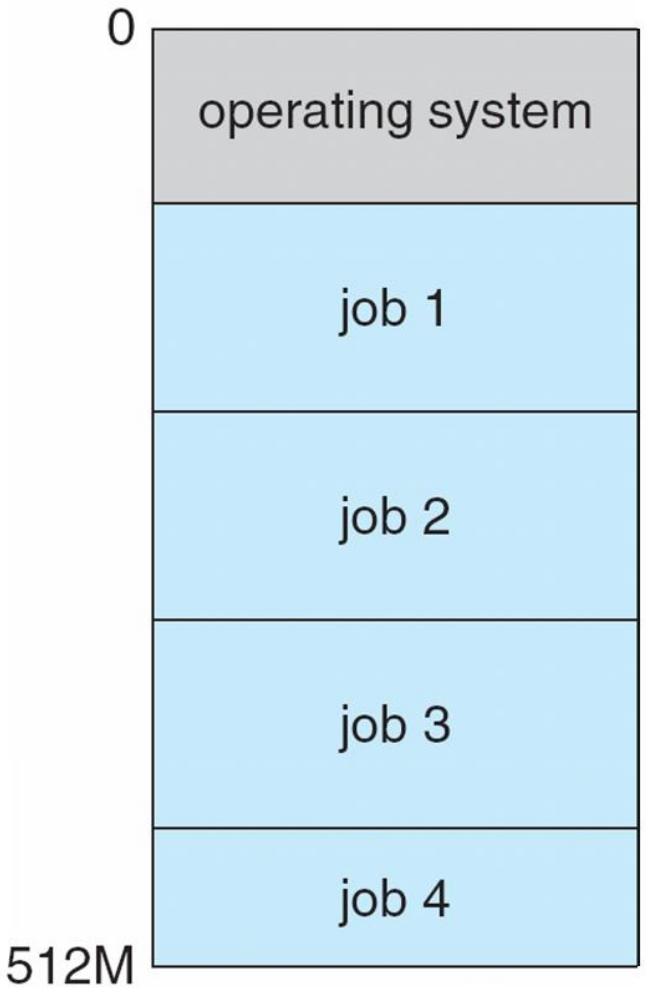
- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a storage-area network (SAN) & closely linked via a LAN
  - Provides a high-availability service which survives failures
    - Asymmetric clustering has one machine in hot-standby mode monitoring the active server while the other is running the applications
    - 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 (dividing a program into separate components that run in parallel on individual PCs)
  - Some have distributed lock manager (DLM) to avoid conflicting operations between shared accessed data

Operating System Structure & Operations

Operating System Structure

- Different OSs vary greatly internally but have many common features.
- First computer systems were batch systems. low CPU utilization.
- 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
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking), many users can share the computer.
- is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing.

Memory Layout for Multiprogrammed System

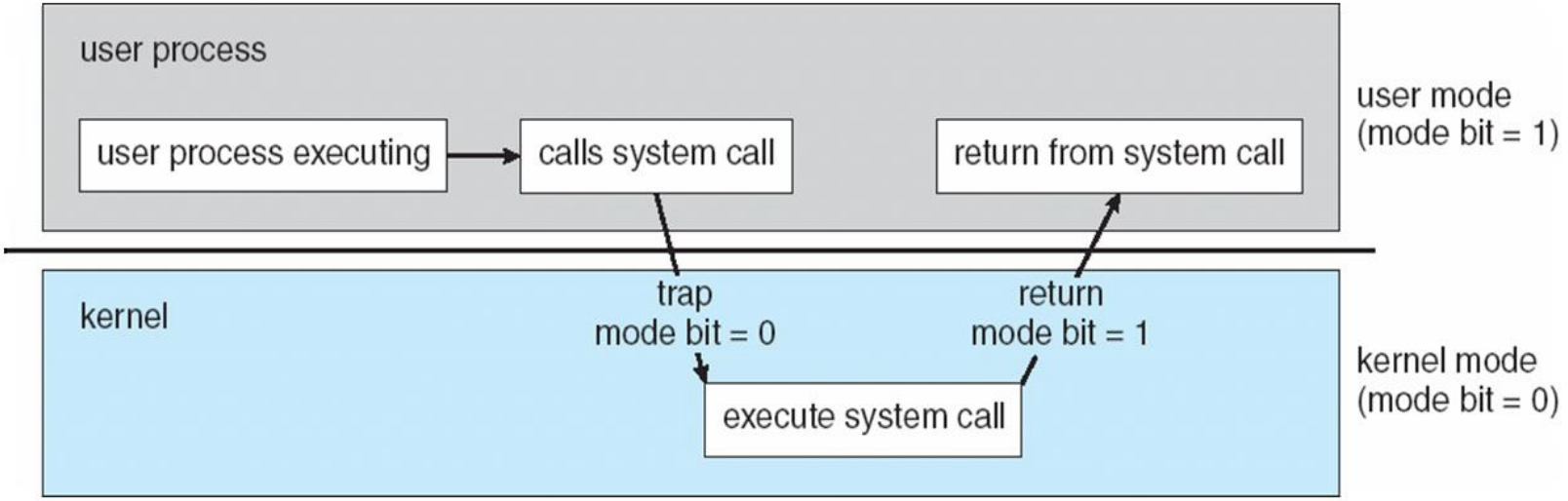


Operating-System Operations

- Dual-mode operation allows OS to protect itself and other system components
  - User mode – computer system is executing on behalf of a user app
  - kernel mode (also *kernel/system/supervisor mode*) protects the OS when it is running – execution done on behalf of operating system.
  - Mode bit provided by hardware (kernel (0) or user (1))
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions whose execution can harm the system are 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

- The Operating System must protect the CPU from being taken over by a user program (e.g. in an infinite loop) -> Timer
  - 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



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
- 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

- A program's instructions and data must be in the main memory for it to be executed by the CPU.
- To improve speed of execution and CPU utilization many programs must reside in the memory at the same time.
- 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

Storage Management

- Users do not see bits and bytes. OS enables them to see data/programs as files and folders.
- File-System management
  - A file is a collection of related information e.g. program files, data files.
  - OS file management activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and dirs
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

Storage Management. Cont.

- Mass-Storage Management
  - **Secondary storage** – hard disks
  - Need hard disks as main memory is too small and volatile.
  - Programs and data are loaded from the hard disk and results stored back on the hard disk.

- OS disk management activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - **Tertiary storage** includes tape drives, CDs, DVDs

Performance of Various Levels 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