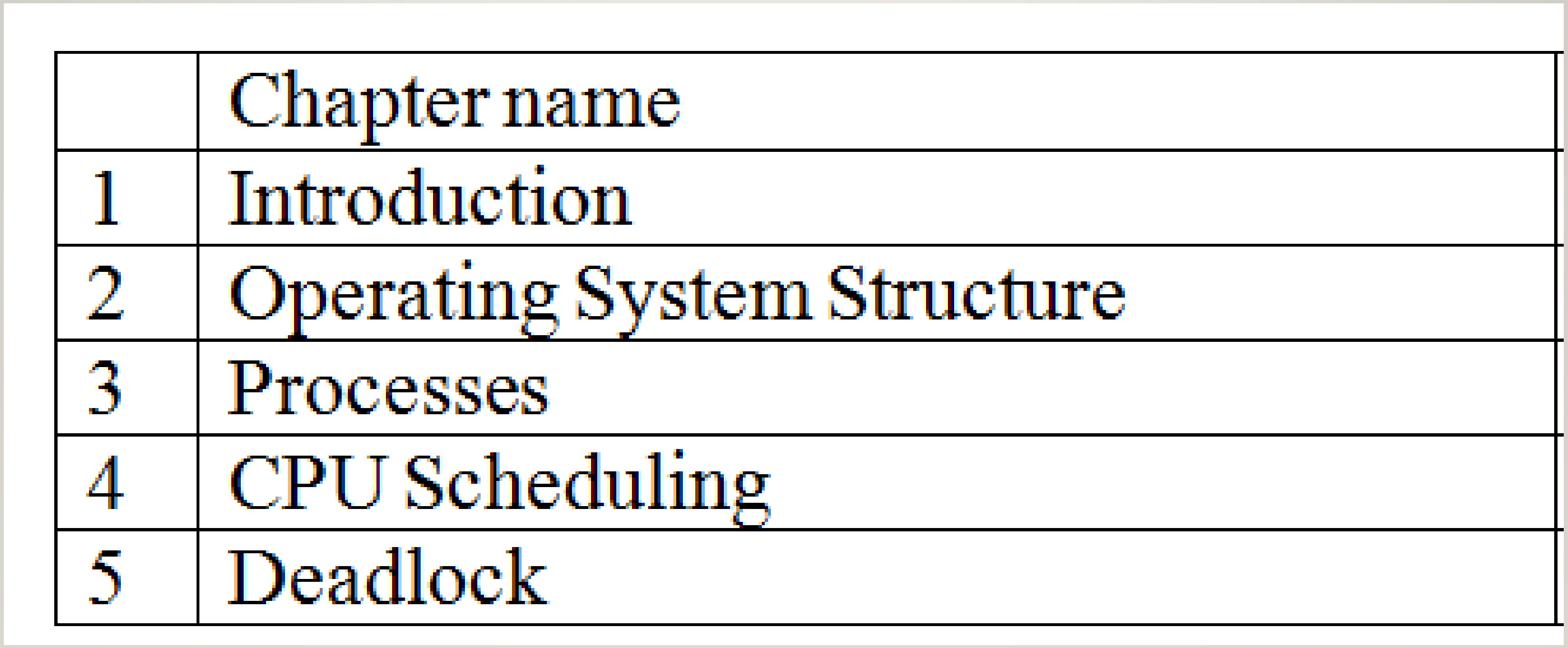
CHAPTER 1: INTRODUCTION

COURSE CONTENTS



	Chapter name
1	Introduction
2	Operating System Structure
3	Processes
4	CPU Scheduling
5	Deadlock

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

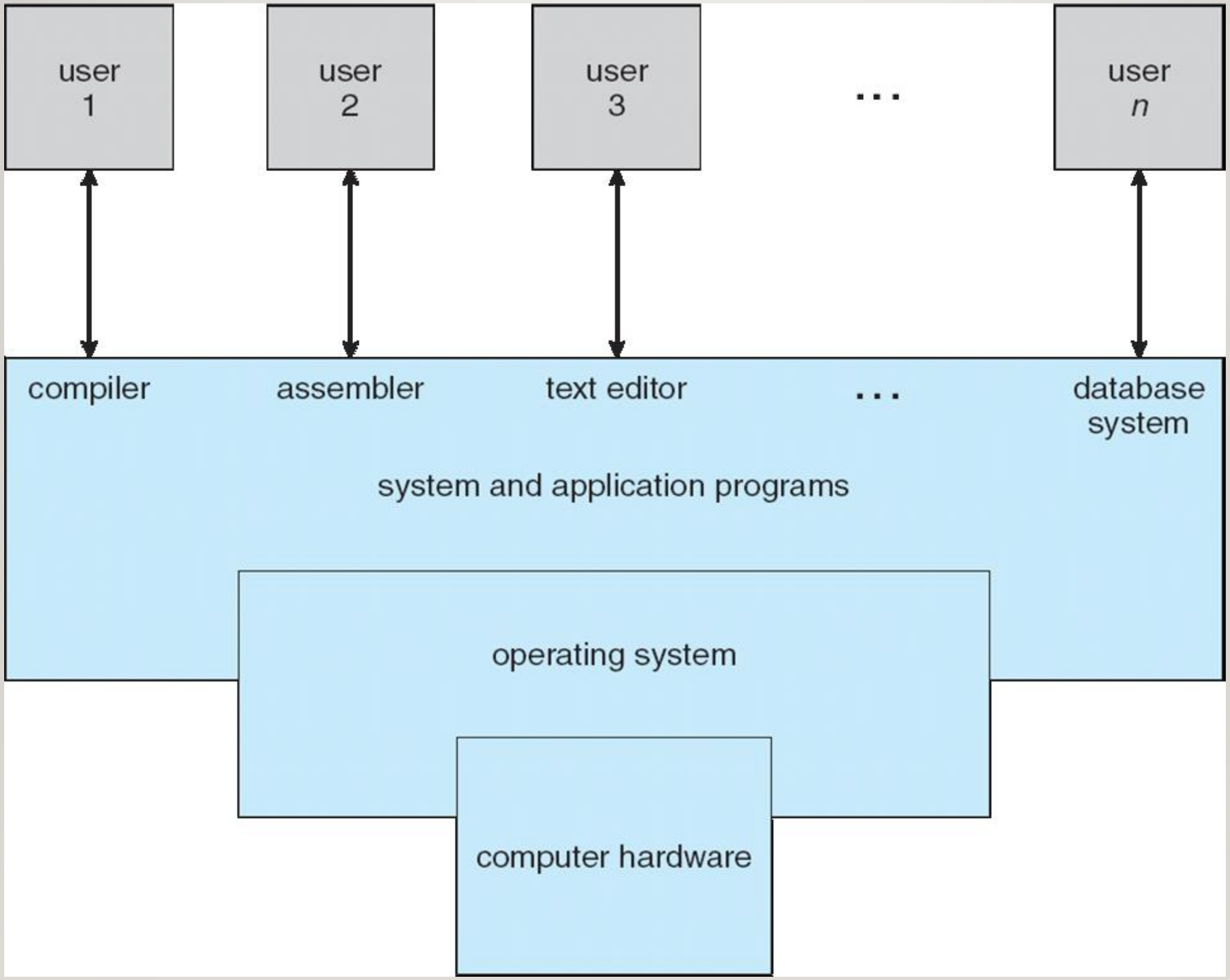
WHAT IS AN OPERATING SYSTEM?

- 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

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** 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.)

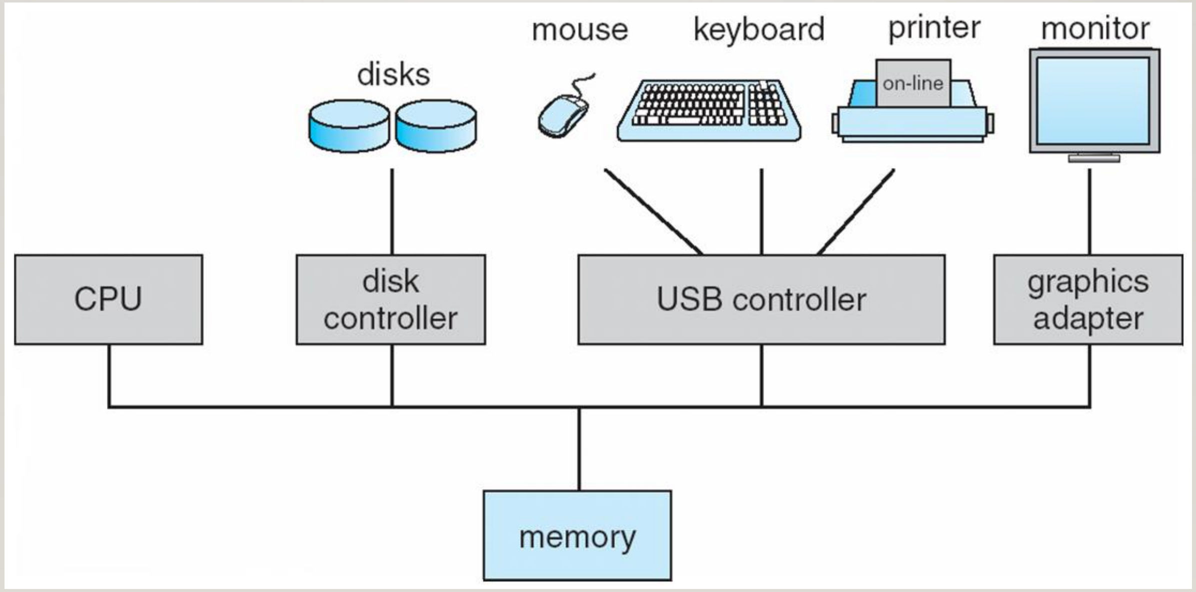
- No universally accepted definition
- The best definition is "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

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

COMMON FUNCTIONS OF INTERRUPTS

- Interrupt transfers control to the interrupt service routine (ISR) generally, through the **interrupt vector**, which contains the addresses of all the service routines
  - The **vector interrupt** have fixed memory location for transfer of control from normal execution.
  - **Non-Vectored Interrupts** don't have fixed memory location.
- 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**
- **Exception** are caused by software executing instructions
- **Interrupt** are caused by hardware devices

Interrupt Service Routine

An ISR (also called an interrupt handler) is a software process invoked by an interrupt request from a hardware device. It handles the request and sends it to the CPU, interrupting the active process. When the ISR is complete, the process is resumed.

A basic example of an ISR is a routine that handles keyboard events, such as pressing or releasing a key. Each time a key is pressed, the ISR processes the input.

How Interrupts are handled

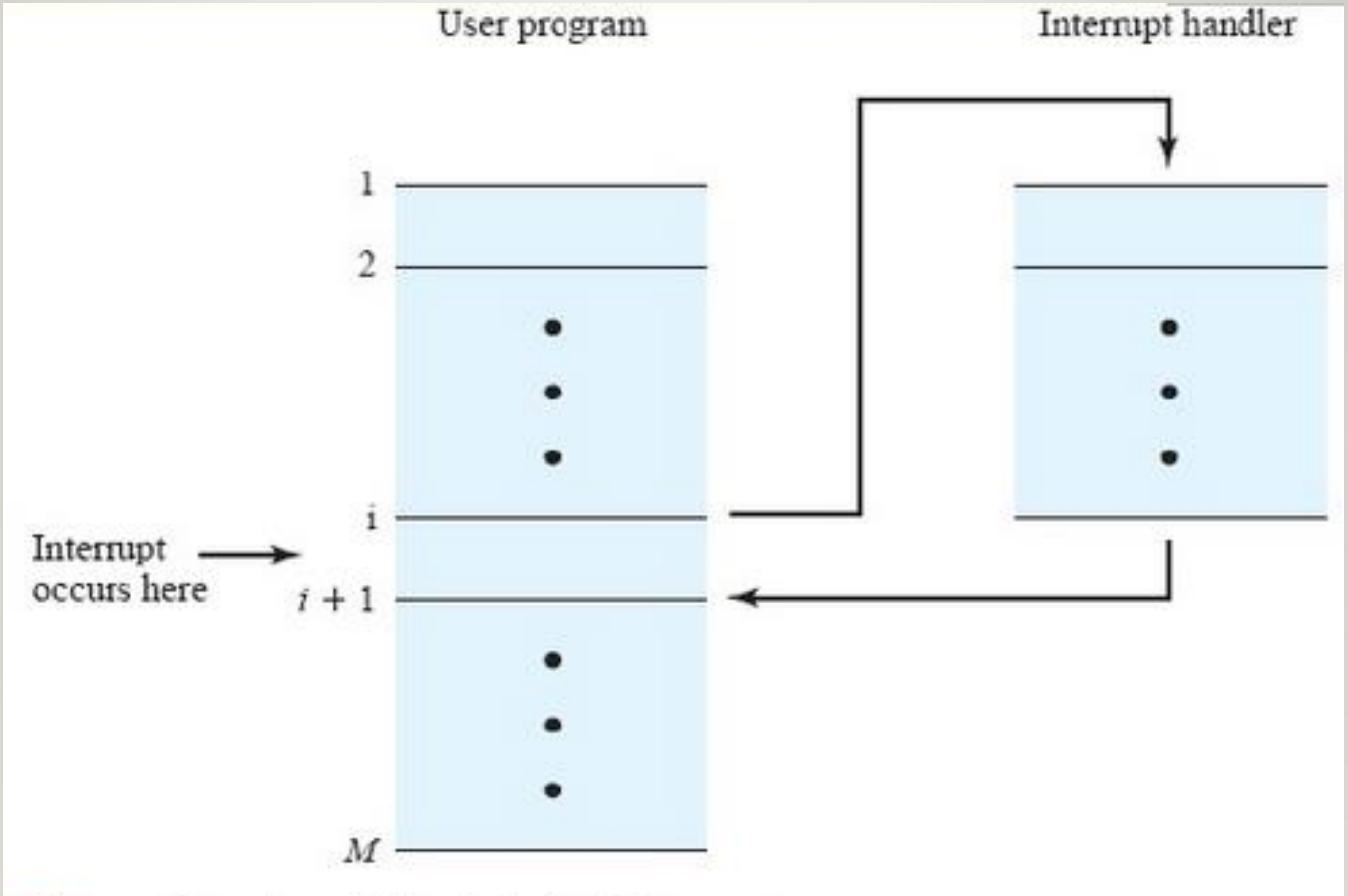


Figure Transfer of Control via Interrupts

How Interrupts are handled

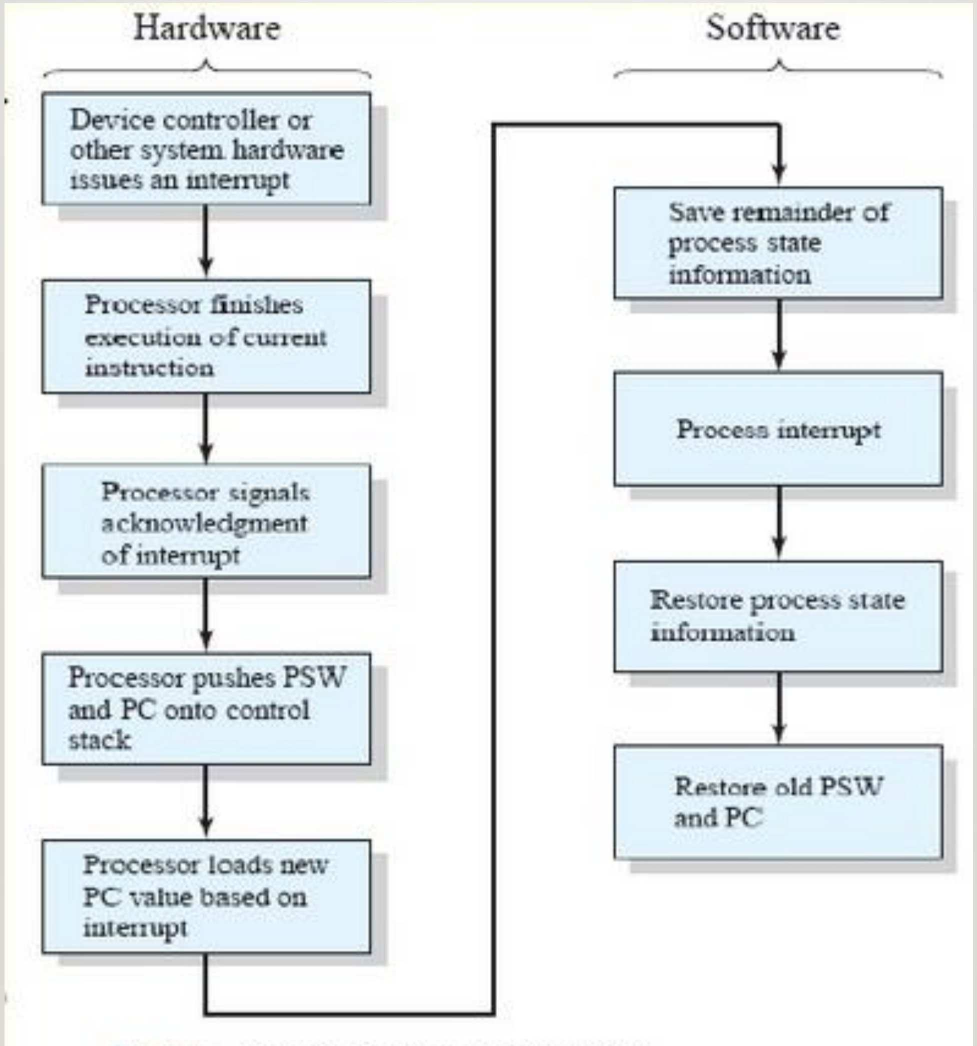
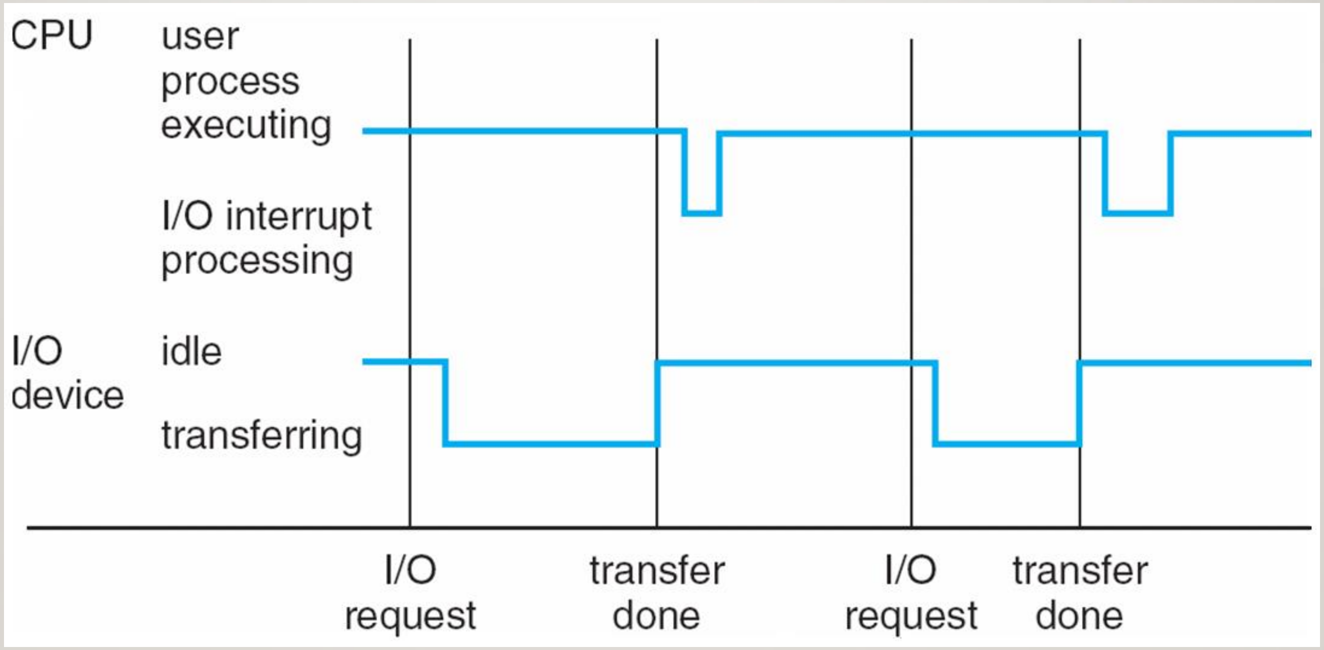

Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

Figure Simple Interrupt Processing

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:** CPU repeatedly checks whether a device needs servicing
  - **vectored** : the device notifies the CPU that it needs servicing
- 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 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 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
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).

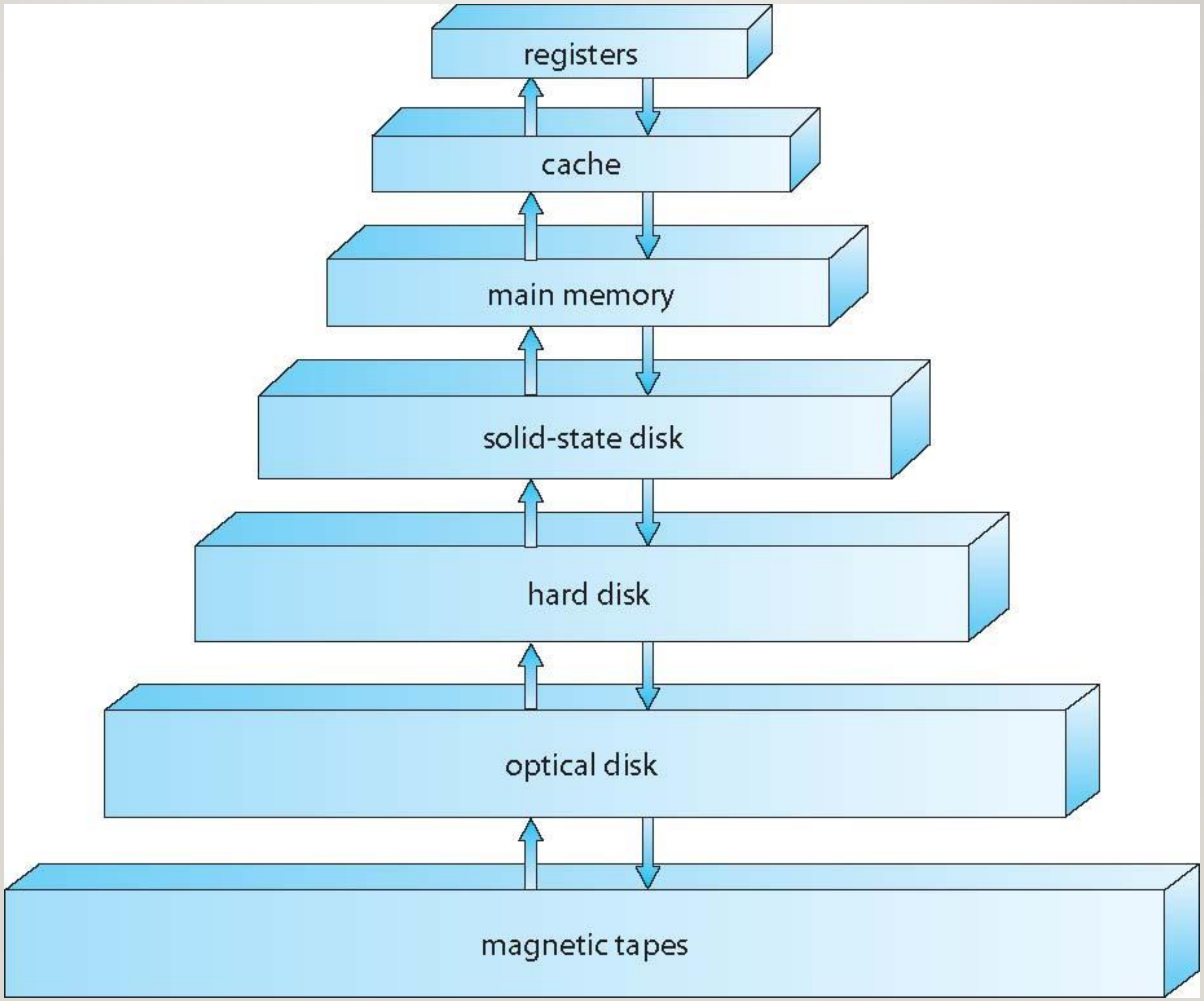
STORAGE STRUCTURE

- 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
- 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 (SSD Hard)** – 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



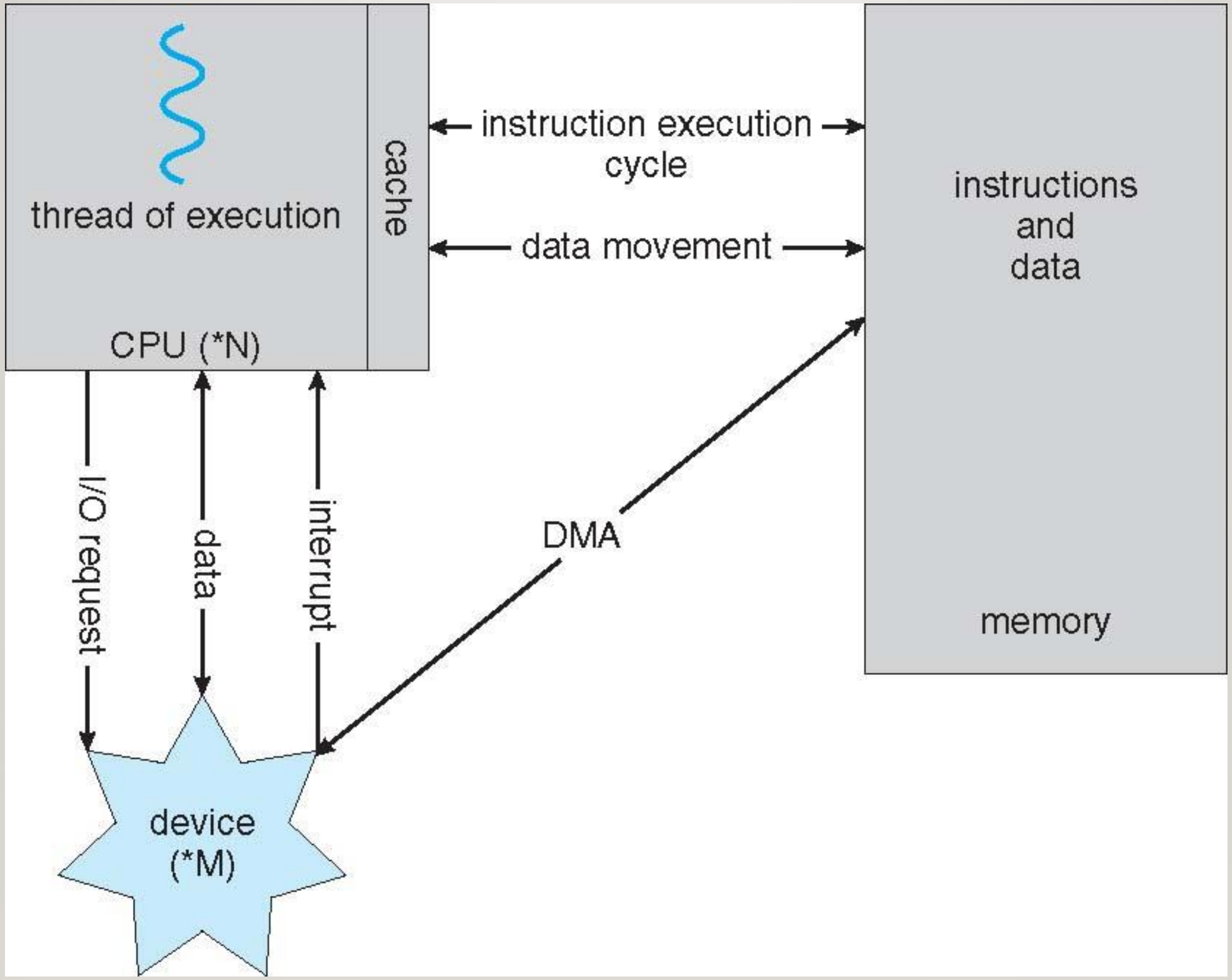
CACHING

- 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

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

HOW A MODERN COMPUTER WORKS

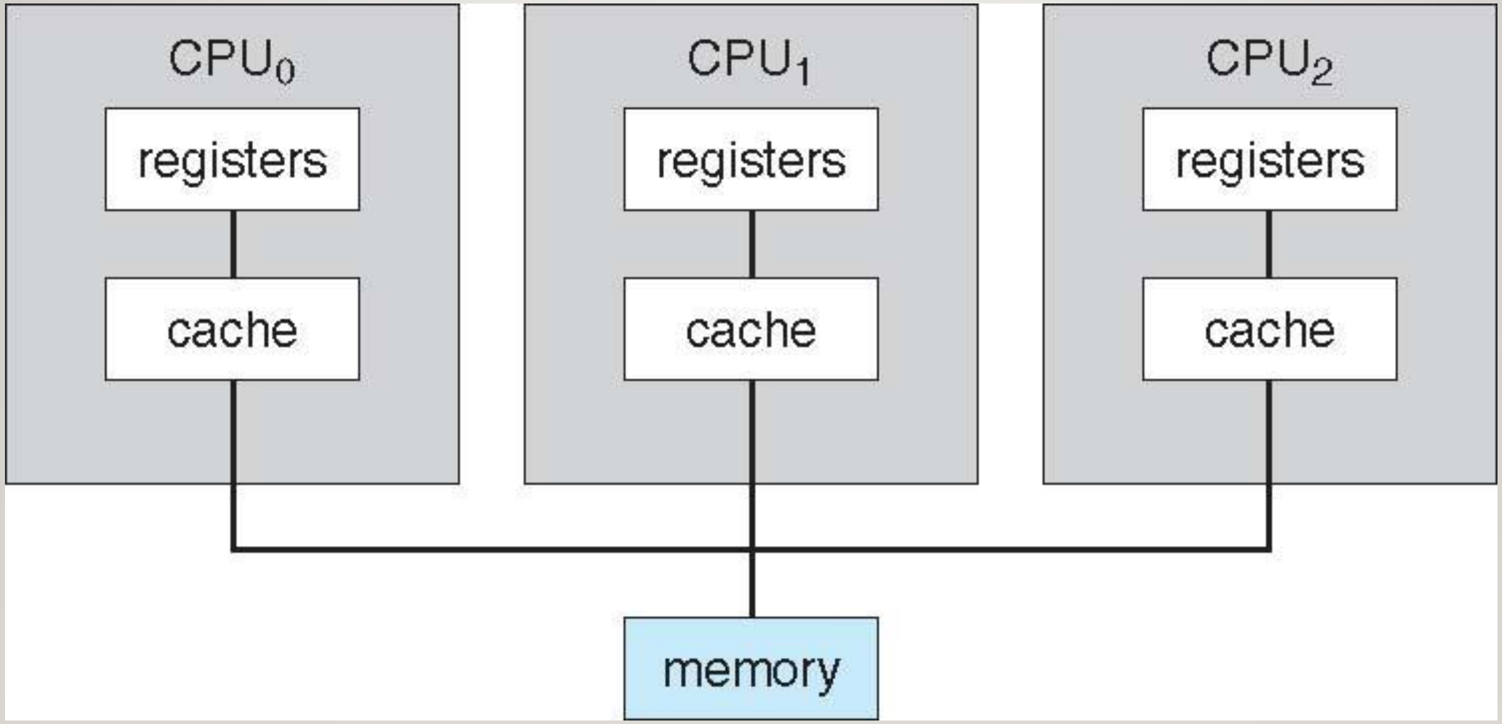


A von Neumann architecture

COMPUTER-SYSTEM ARCHITECTURE

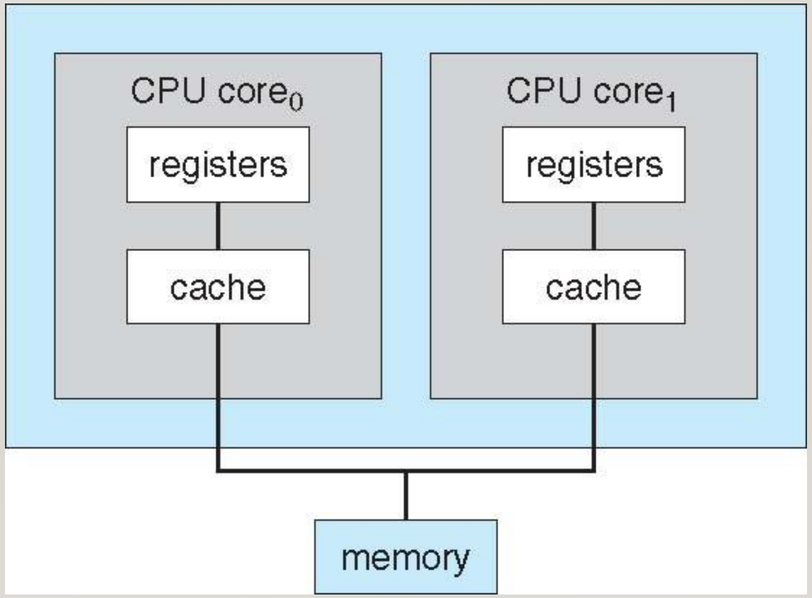
- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Advantages include:
    1. **Increased throughput**
    2.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

SYMMETRIC MULTIPROCESSING ARCHITECTURE



A DUAL-CORE DESIGN

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



DIFFERENCE BETWEEN CORE I3, I5 AND I7 PROCESSORS

Model	Core i3	Core i5	Core i7
Number of cores	2	4	4
Hyper-threading	Yes	No	Yes
Turbo boost	No	Yes	Yes
K model	No	Yes	Yes

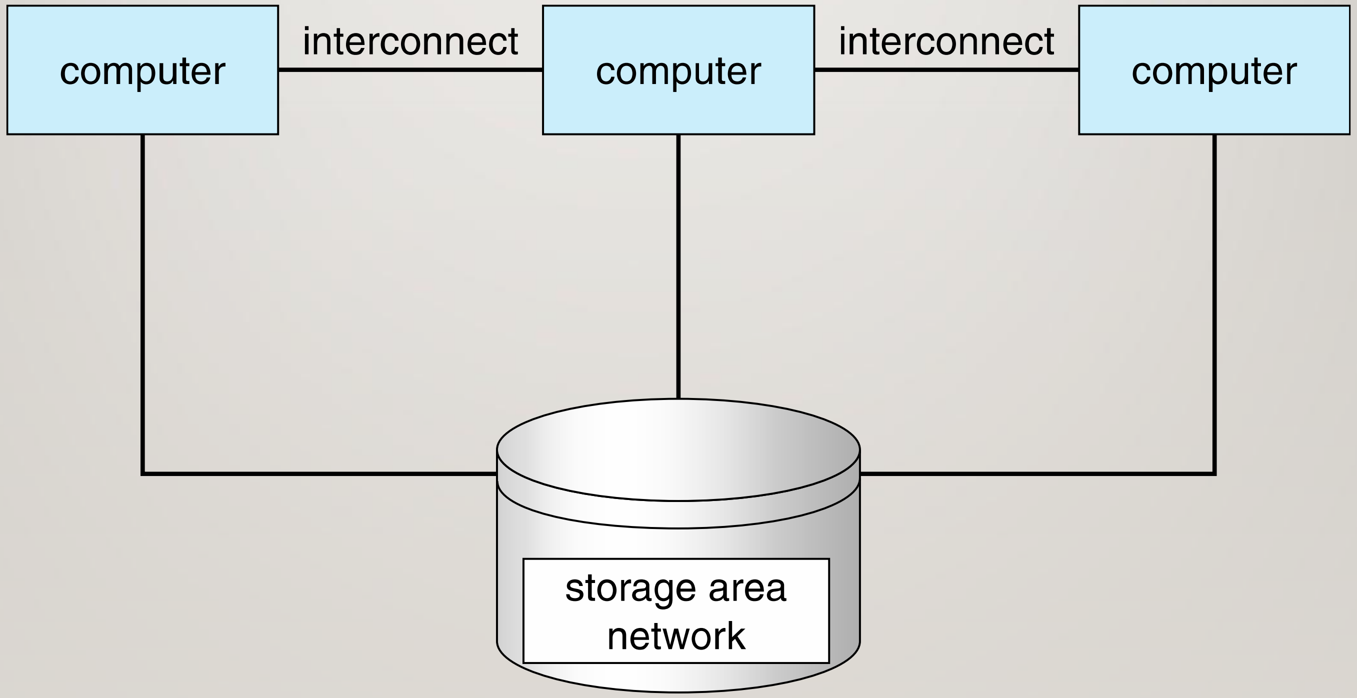
- **Hyper-Threading** is a technology used by some Intel microprocessor s that allows a single microprocessor to act like two separate processors to the operating system and the application program s that use it.

- **Turbo Boost** Technology is a way to automatically run the processor core faster than the marked frequency. The processor must be working in the power, temperature, and specification limits of the thermal design power (TDP)

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



Operating System Structure

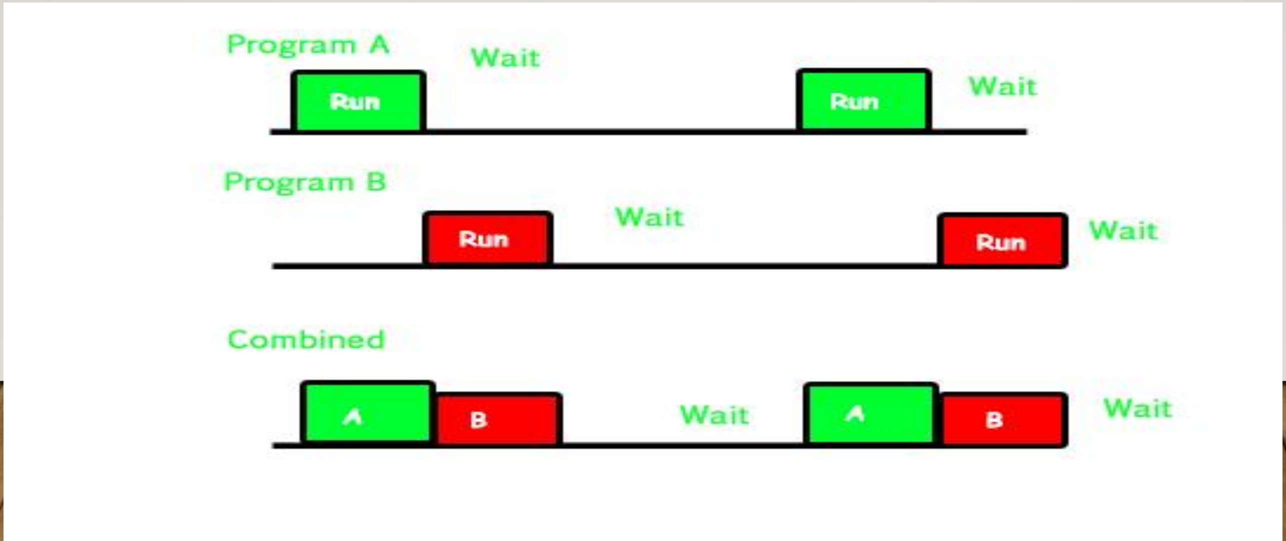
Multiprogramming is also the ability of an operating system to execute more than one program on a single processor machine. More than one task/program/job/process can reside into the main memory at one point of time. A computer running excel and firefox browser simultaneously is an **example** of **multiprogramming**.

Multiprogramming (**Batch system**) needed for efficiency

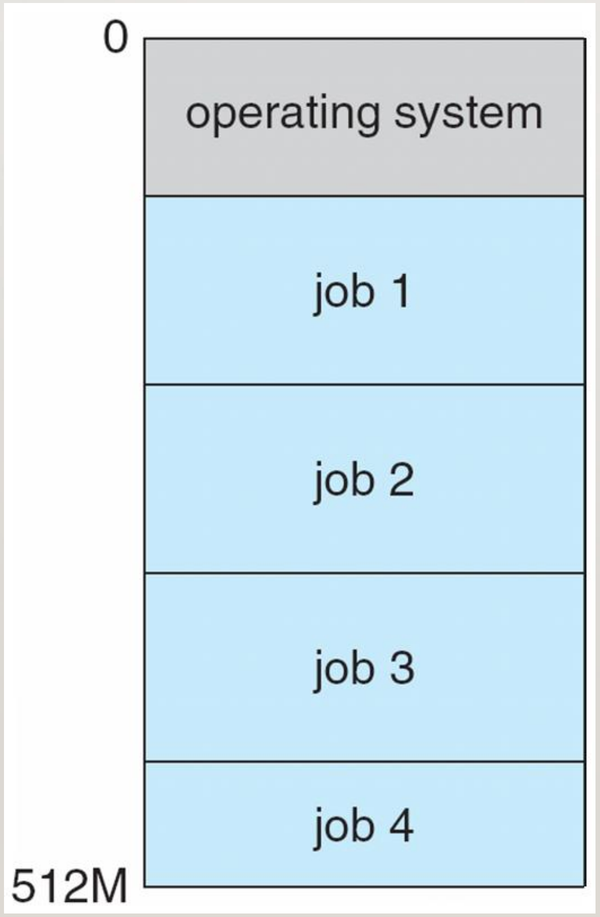
Single user cannot keep CPU and I/O devices busy at all times

Multiprogramming organizes jobs (code and data) so CPU always has one to execute

- ❑A subset of total jobs in system is kept in memory
- ❑One job selected and run via **job scheduling**
- ❑When it has to wait (for I/O for example), OS switches to another job



MEMORY LAYOUT FOR MULTIPROGRAMMED SYSTEM

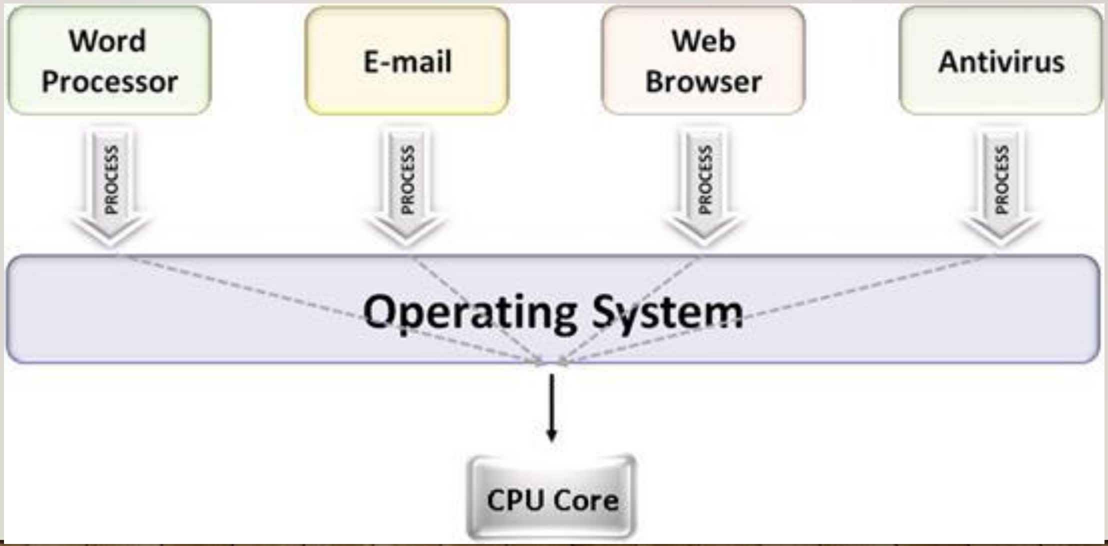


OPERATING SYSTEM STRUCTURE

- **Timesharing** (**multitasking**) is logical extension in which 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

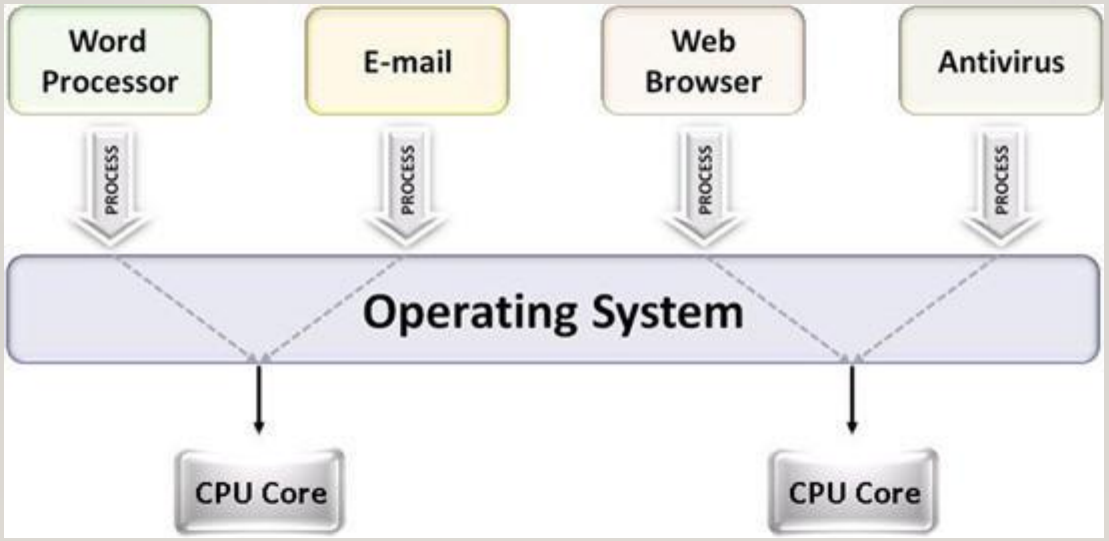
Multitasking

Multitasking is the ability of an operating system to execute more than one task simultaneously on a single processor machine. **Though we say so but in reality no two tasks on a single processor machine can be executed at the same time. Actually CPU switches from one task to the next task so quickly that appears as if all the tasks are executing at the same time.** More than one task/program/job/process can reside into the same CPU at one point of time.



Multiprocessing

Multiprocessing is the ability of an operating system to execute more than one process simultaneously on a multi processor machine. In this, a computer uses more than one CPU at a time.



OPERATING-SYSTEM OPERATIONS

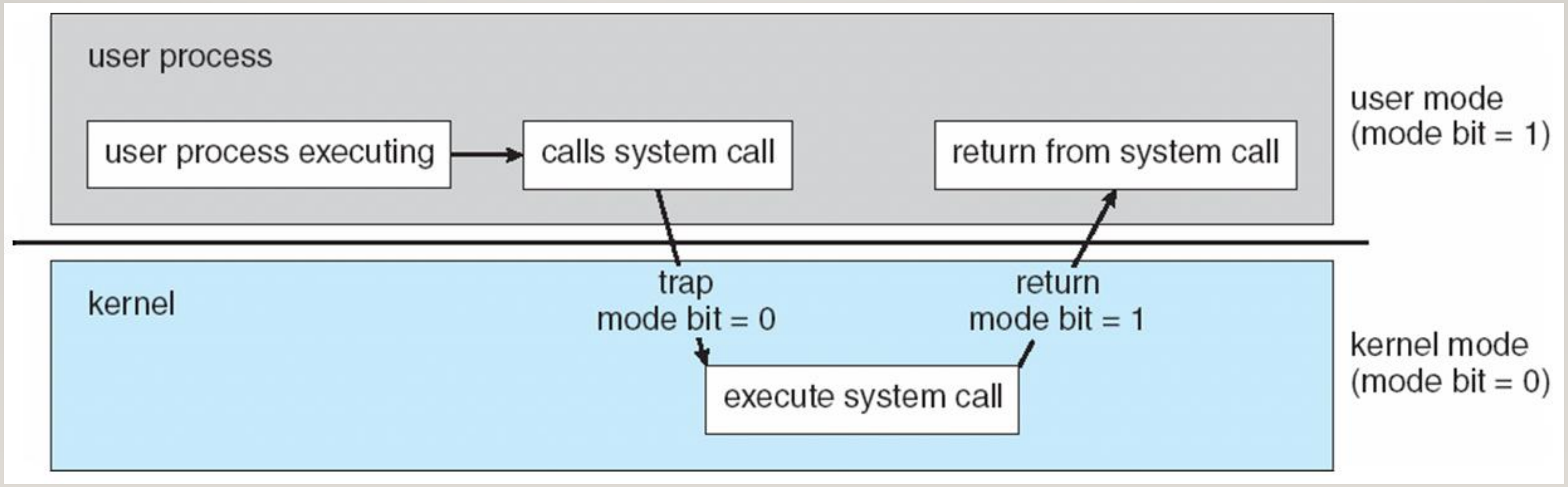
- **Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service
    - Other process problems include infinite loop, processes modifying each other or the operating system

OPERATING-SYSTEM OPERATIONS (CONT.)

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - When a computer application is running, it is in the user mode. Some examples are word application, PowerPoint.
  - When the process is in user mode and requires any hardware resource, that request is sent to the kernel.
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

TRANSITION FROM USER TO KERNEL MODE

- Timer to prevent infinite loop / 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



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
- 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 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

- **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**
  - 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 used to store data that does not fit in main memory or data that must be kept for a "long" period of time
- Proper management is of central importance
- Entire speed of computer operation depend on disk subsystem and its algorithms
- OS activities
    - Free-space management
    - Storage allocation
    - Disk scheduling
- Some storage need not be fast
    - Tertiary storage includes optical storage, magnetic tape
    - Still must be managed – by OS or applications
    - Varies between WORM (write-once, read-many-times) and RW (read-write)

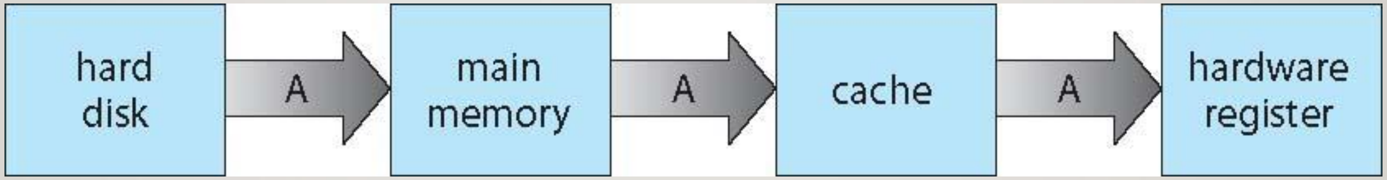
PERFORMANCE OF VARIOUS LEVELS OF STORAGE

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

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