

# OPERATING SYSTEMS

---

## Introduction

**Venkatesh Prasad**

Department of Computer Science

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the following resources and persons:
  1. Slides of Operating System Concepts, Abraham Silberschatz, Peter Baer Galvin, Greg Gagne - 9<sup>th</sup> edition 2013 and some slides from 10<sup>th</sup> edition 2018
  2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9<sup>th</sup> edition 2018
  3. Some presentation transcripts from A. Frank – P. Weisberg
  4. Some conceptual text from Operating Systems: Three Easy Pieces, Remzi Arpaci-Dusseau, Andrea Arpaci Dusseau

The objective(s) of this course is to:

- Provide an insight into the basic organization of computer systems
- Familiarize students with various components of an Operating System
- Focus on fundamental problems and optimal solutions for resource management in Operating System such as Process, Threads, Disk scheduling and Memory Management
- Discuss Storage Management such as the design of a file system on secondary storage and system I/O in depth
- Discuss System Protection, System Security and Distributed Systems Basics

At the end of this course the student will be able to:

- Correlate the different components of an Operating System
- Compare and contrast various scheduling algorithms and their performance tradeoffs.
- Analyze techniques for system resource management and explore design trade-offs in designing memory components of the Operating system.
- Analyze performance issues associated with I/O devices and learn the algorithms & structures used for storage management
- Understand the goals and principles of protection in a modern computer system, explore security threats & attacks and understand the roles, types and advantages of distributed systems

- 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

# OPERATING SYSTEMS

## General Definition

- An Operating System is a program that acts as an intermediary between a user of a computer and the computer hardware
- It provides a user-friendly environment in which a user may easily develop and execute programs. Otherwise, hardware knowledge would be mandatory for computer programming.



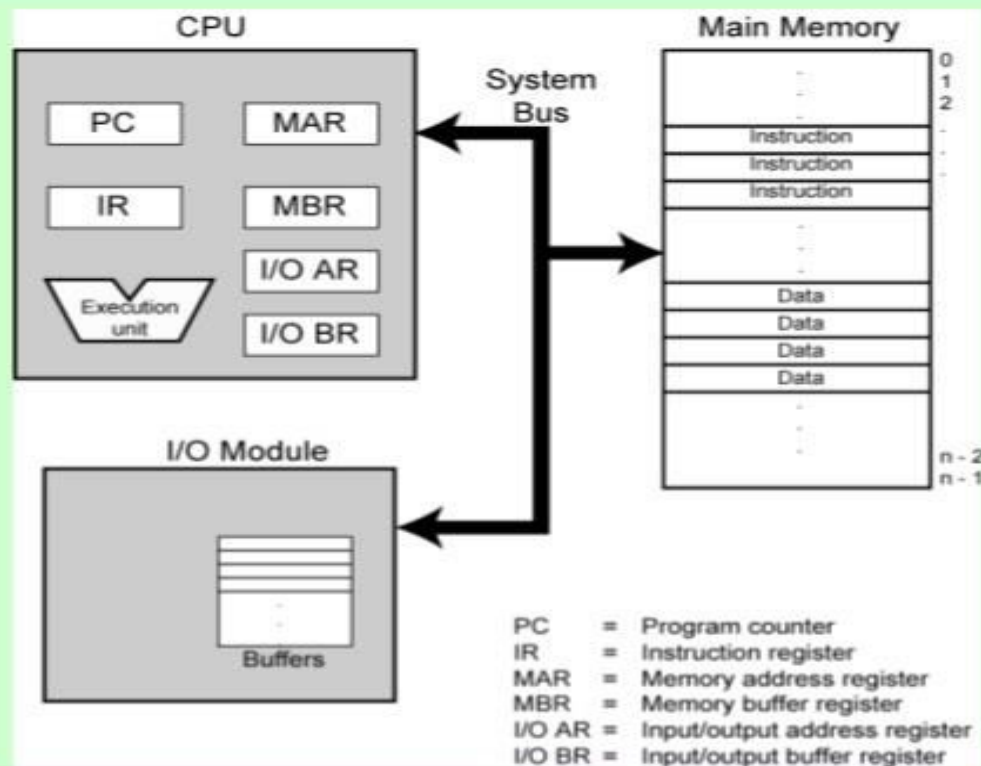
- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner
- Manage resources such as • Memory • Processor(s) • I/O Devices

- Only a small percentage of computer practitioners will be involved in the creation or modification of an operating system.
- However almost all code runs on top of an operating system, and thus knowledge of how operating systems work is crucial to **proper, efficient, effective, and secure programming**.
- Understanding the fundamentals of operating systems, how they drive computer hardware, and what they provide to applications is essential to those who program them and those who write programs on them and use them.



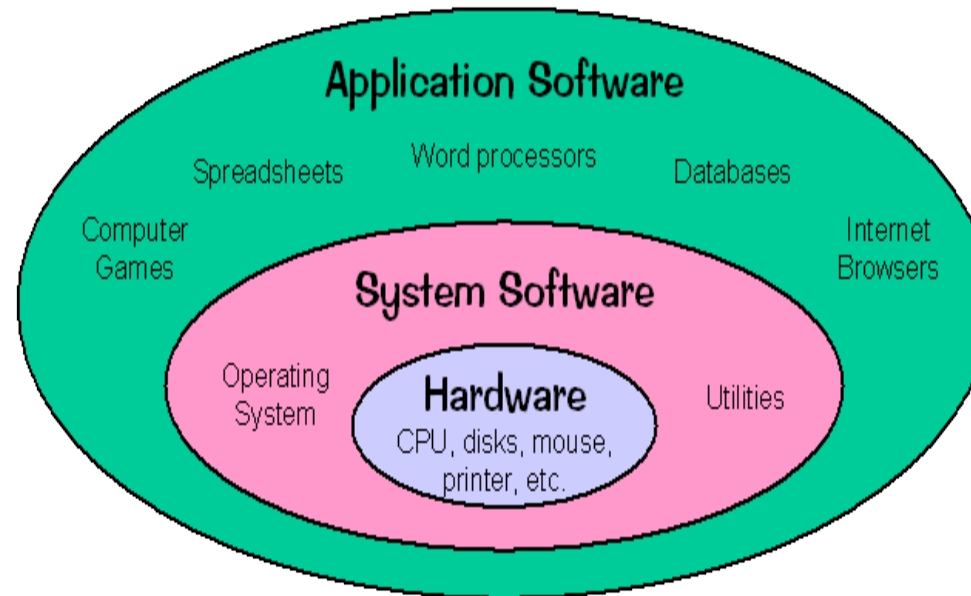
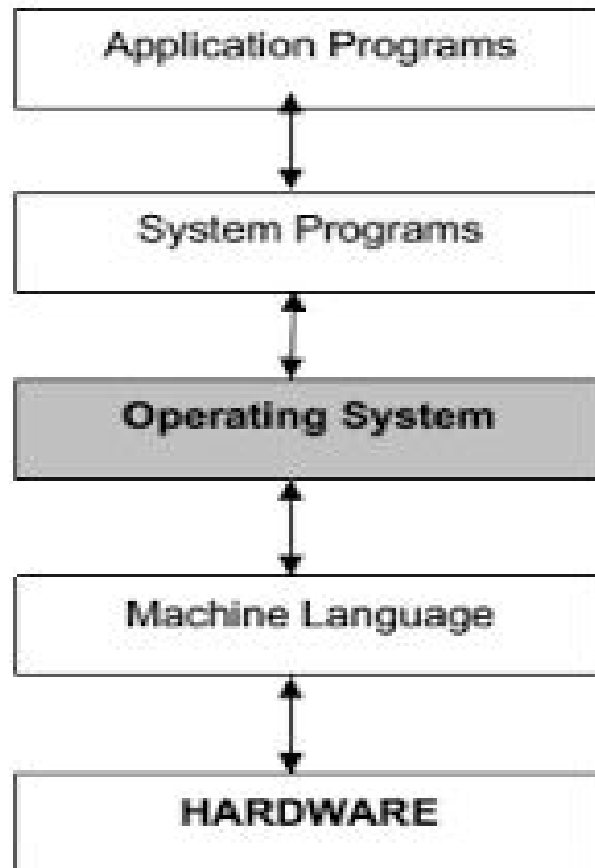
Four main structural elements: Processor, Main memory, I/o modules and System Bus

### Computer Components: Top Level View



# OPERATING SYSTEMS

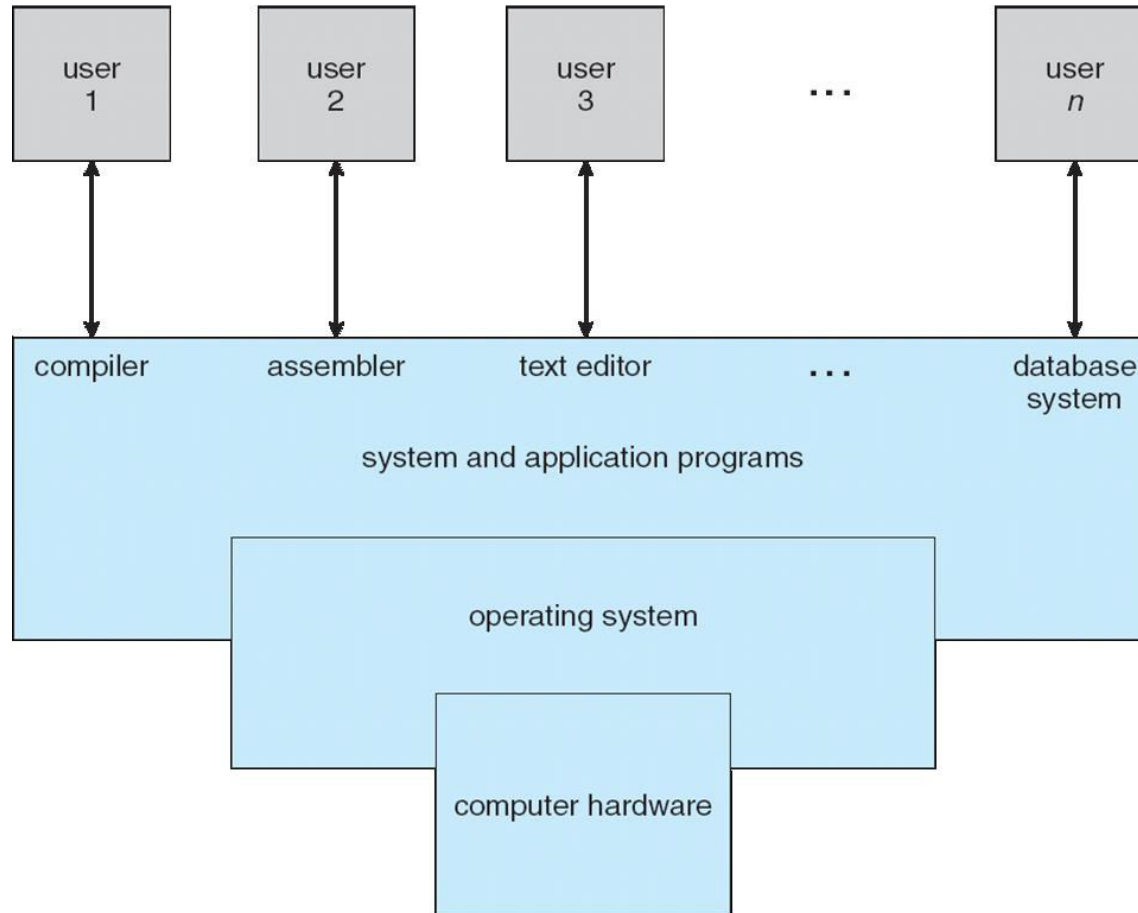
## OS as an Extended Machine



- 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

# OPERATING SYSTEMS

## Four Components of a Computer System (Abstract view)



# OPERATING SYSTEMS

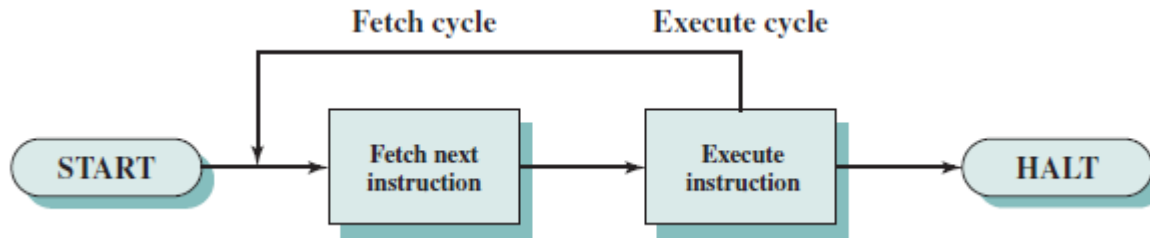
## What Operating Systems Do

---



- 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 dedicated 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

- ❑ What happens, at the fundamental level, when a program runs?

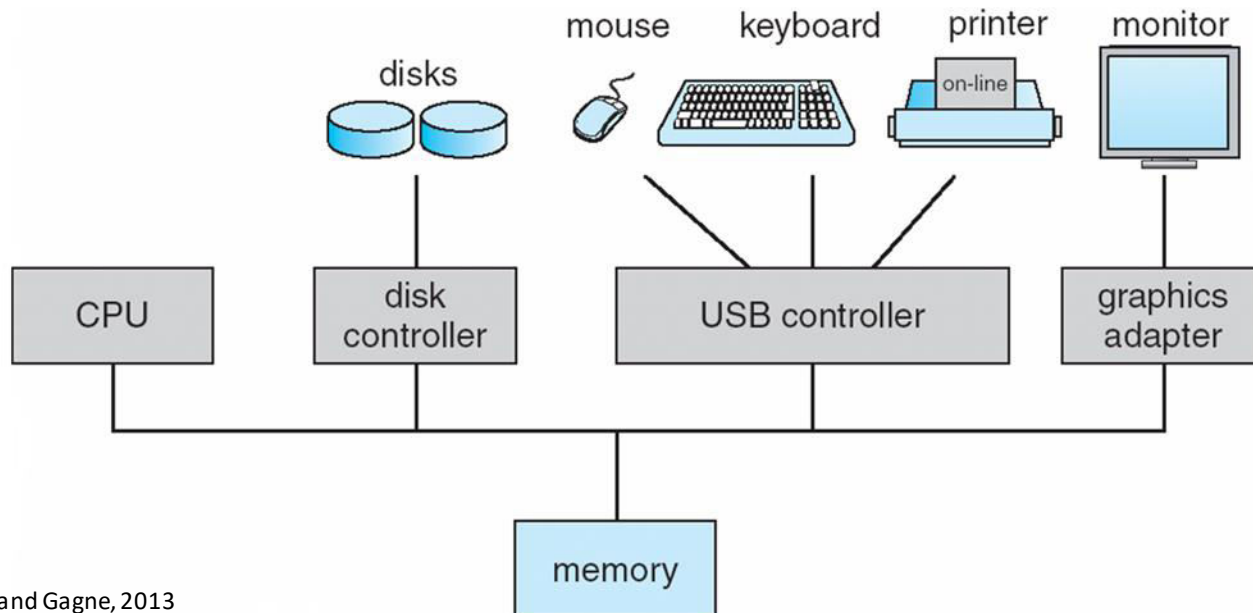


- The processor **fetches** instructions from memory, **decodes** and **executes** them.
- The Fetch and Execute cycles are repeated until the program terminates.
- This is called the **Von Neumann** model of computing.

- 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
- “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-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





- 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 has registers for action (like “read character from keyboard”) to take
- 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](#)

- 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**

# OPERATING SYSTEMS

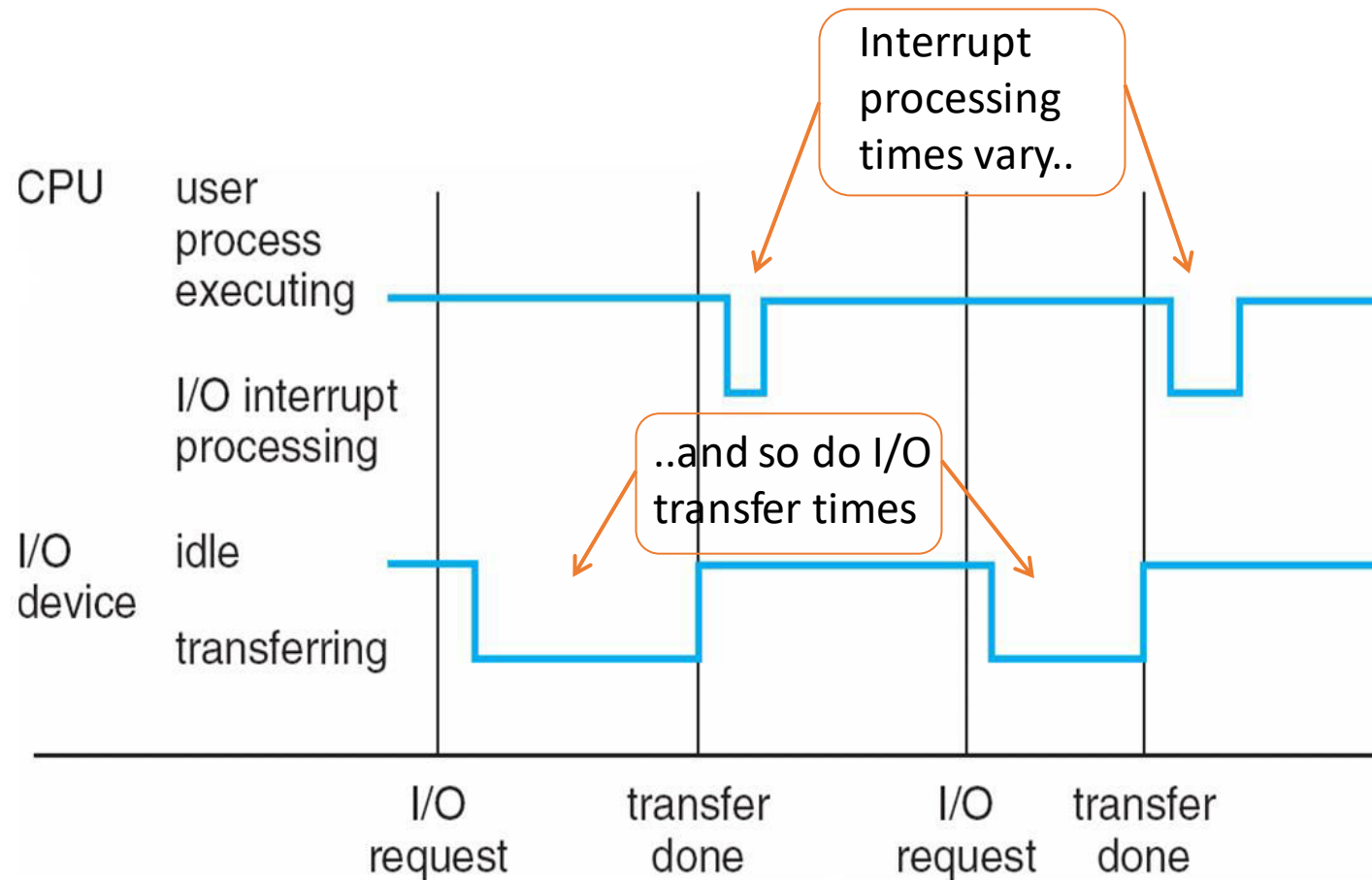
## Interrupt Handling

---

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

# OPERATING SYSTEMS

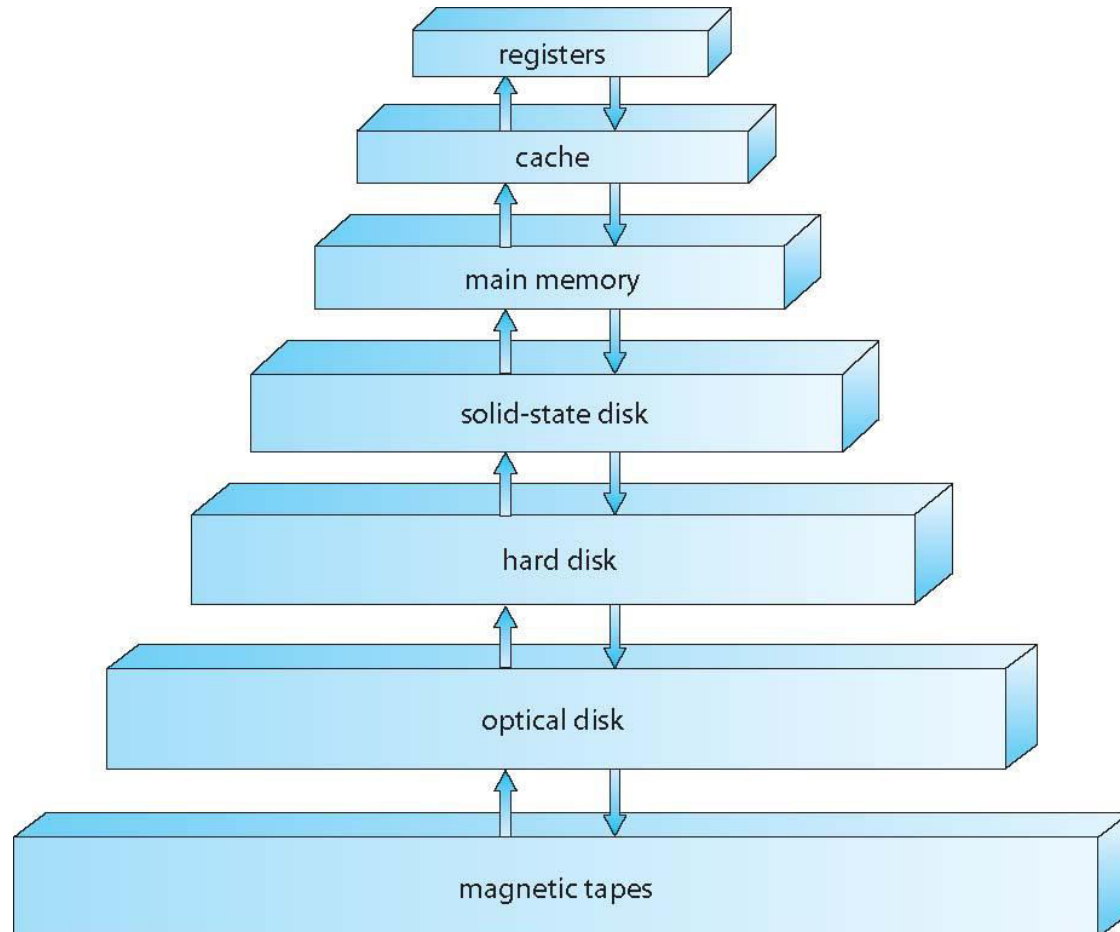
## Interrupt Timeline for a single process doing output



- 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

- Main memory – only large storage media that the CPU can access directly (**Random access and** 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** – faster than hard disks, nonvolatile
  - Various technologies and becoming more popular

- 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



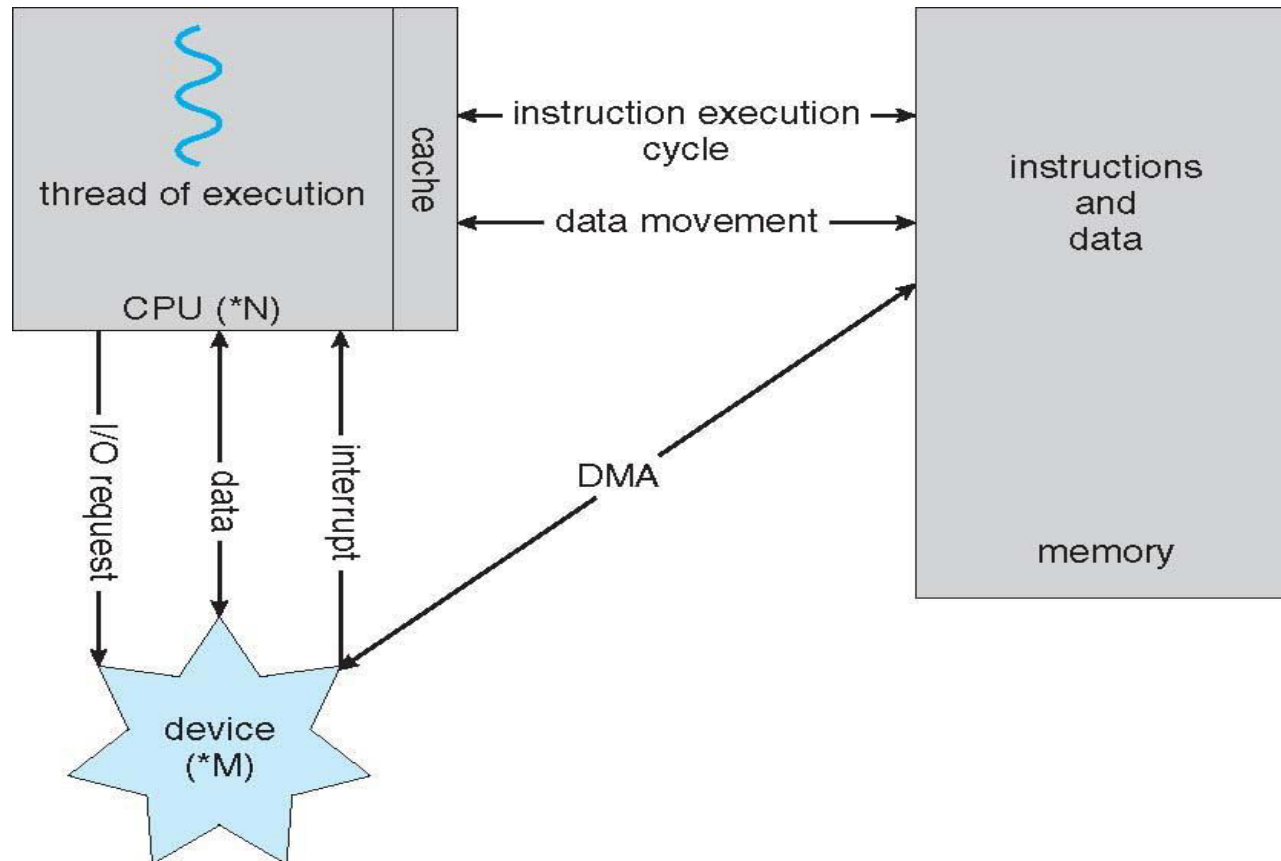


- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (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

- 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 SYSTEMS

## How a modern computer system works





**THANK YOU**

---

**Venkatesh Prasad**

Department of Computer Science Engineering

**[venkateshprasad@pes.edu](mailto:venkateshprasad@pes.edu)**