OPERATING SYSTEMS

Introduction, Computer System Organization

Chandravva Hebbi
Department of Computer Science

OPERATING SYSTEMS

Slides Credits for all PPTs of this course

- 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 - 9th edition 2013 and some slides from 10th edition 2018
2. Some conceptual text and diagram from Operating Systems - Internals and Design Principles, William Stallings, 9th 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

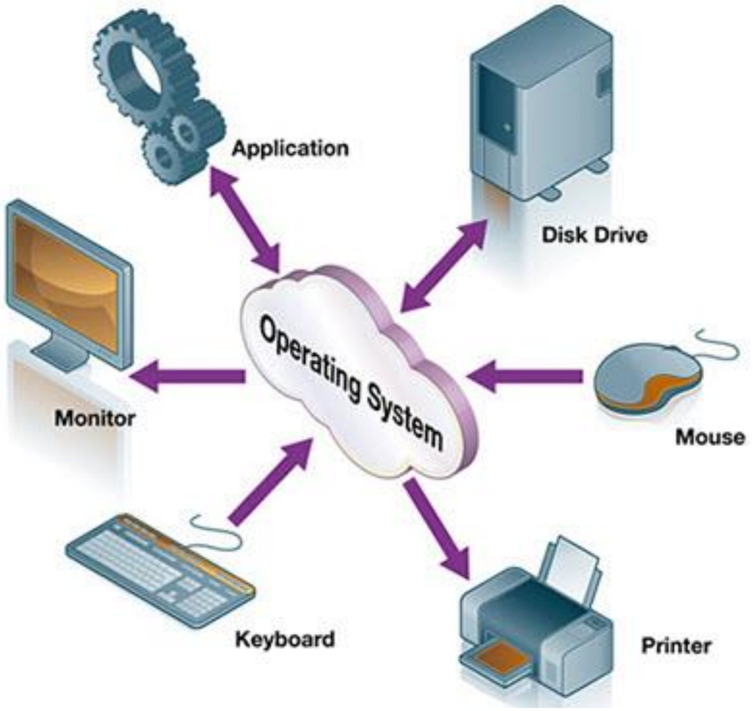
OPERATING SYSTEMS

Introduction, Computer System Organization

Chandravva Hebbi

Department of Computer Science

Need for Operating System

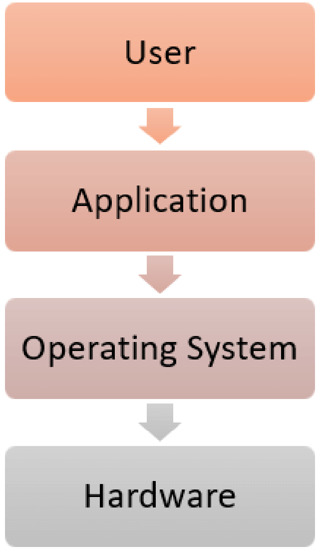


Question:

If OS, is not developed then how will the application developers access the hardware.

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.



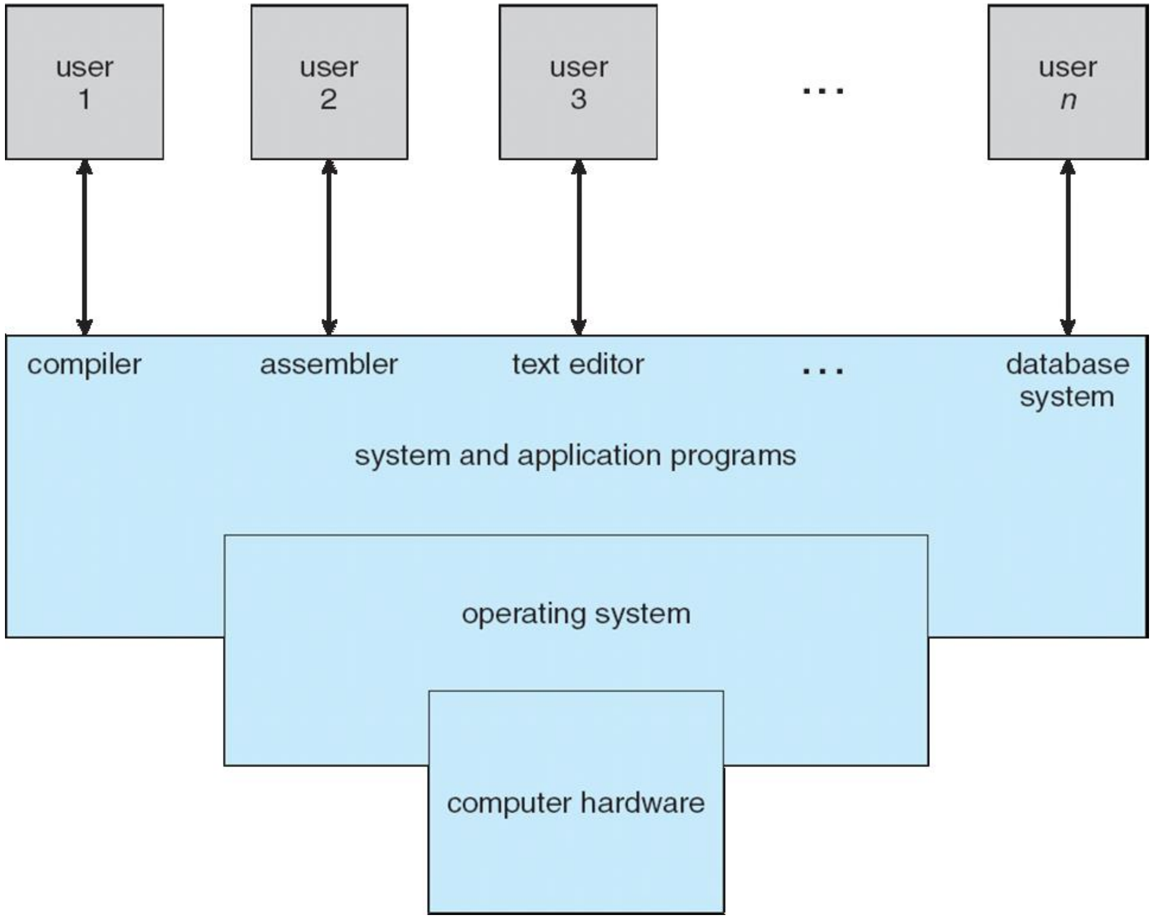
Operating System Goals

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

Why Study Operating Systems?

- 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

Four Components of a Computer System (Abstract view)



Computer System Structure

- 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

What Operating Systems Do

- Depends on the point of view user and system
- 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.
  - Maximize resource utilization.
  - Available CPU time, memory, and I/O are used efficiently and that no individual user takes more than her fair share

What Operating Systems Do

- Users of dedicated systems such as workstations have dedicated resources but frequently use shared resources from servers.
  - resources like file, compute, and print servers are shared.
  - operating system is designed to compromise between individual usability and resource utilization
- 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

System View

- 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

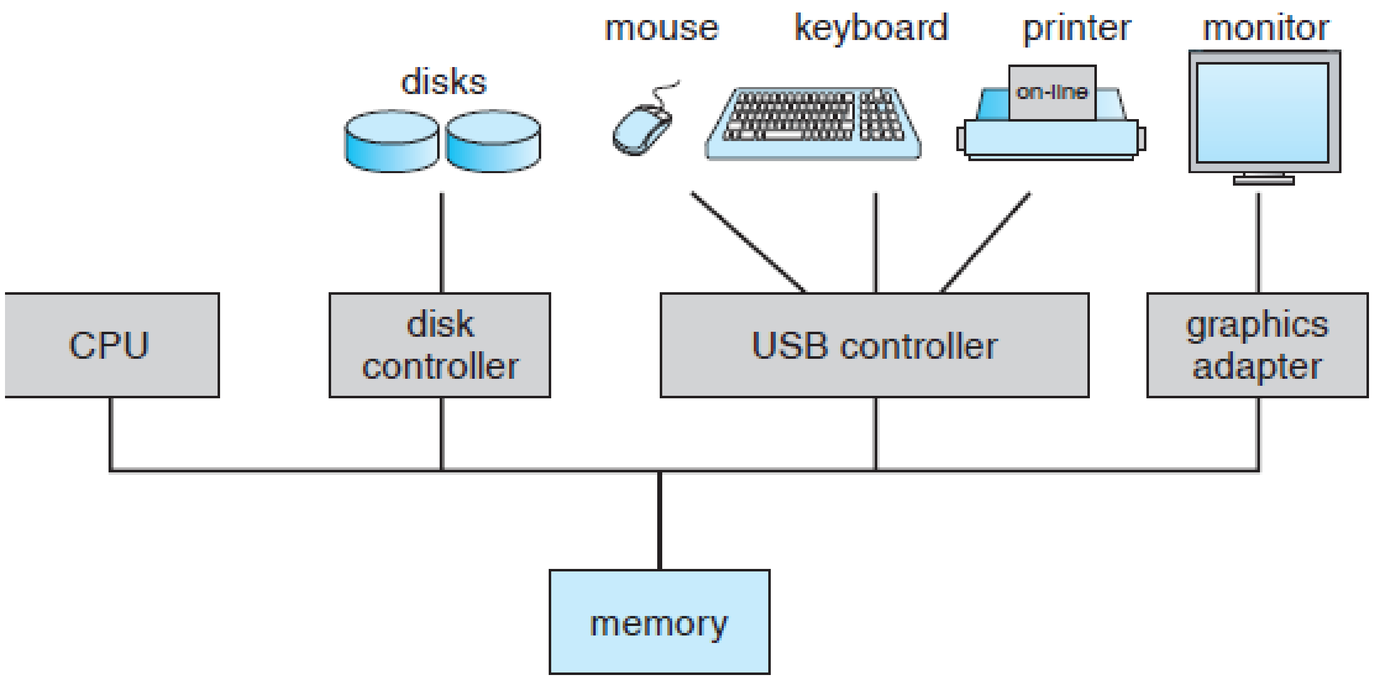
Defining Operating System

- The OS has many roles and functions
- The fundamental goal of computer systems is to execute user programs and to make solving user problems easier.
- The common functions of controlling and allocating resources are then brought together into one piece of software: the **operating system**
- "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 Organization

- Computer-system consists of,
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - The CPU and the device controllers can execute concurrently, competing for memory cycles.
    - memory controller is provided to synchronize access to the memory.

Computer System Organization



A modern computer system

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 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

Computer System Organization

- When the system is booted, the first program that starts running is a Boostrap.
- It is stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM).
- Bootstrap is known by the general term firmware, within the computer hardware.
- It initializes all aspects of the system, from CPU registers to device controllers to memory contents.
- The bootstrap program must know how to load the operating system and how to start executing that system.
- The bootstrap program must locate and load into memory the operating system kernel.
- The first program that is created is init, after the OS is booted. It waits for the occurrence of event.

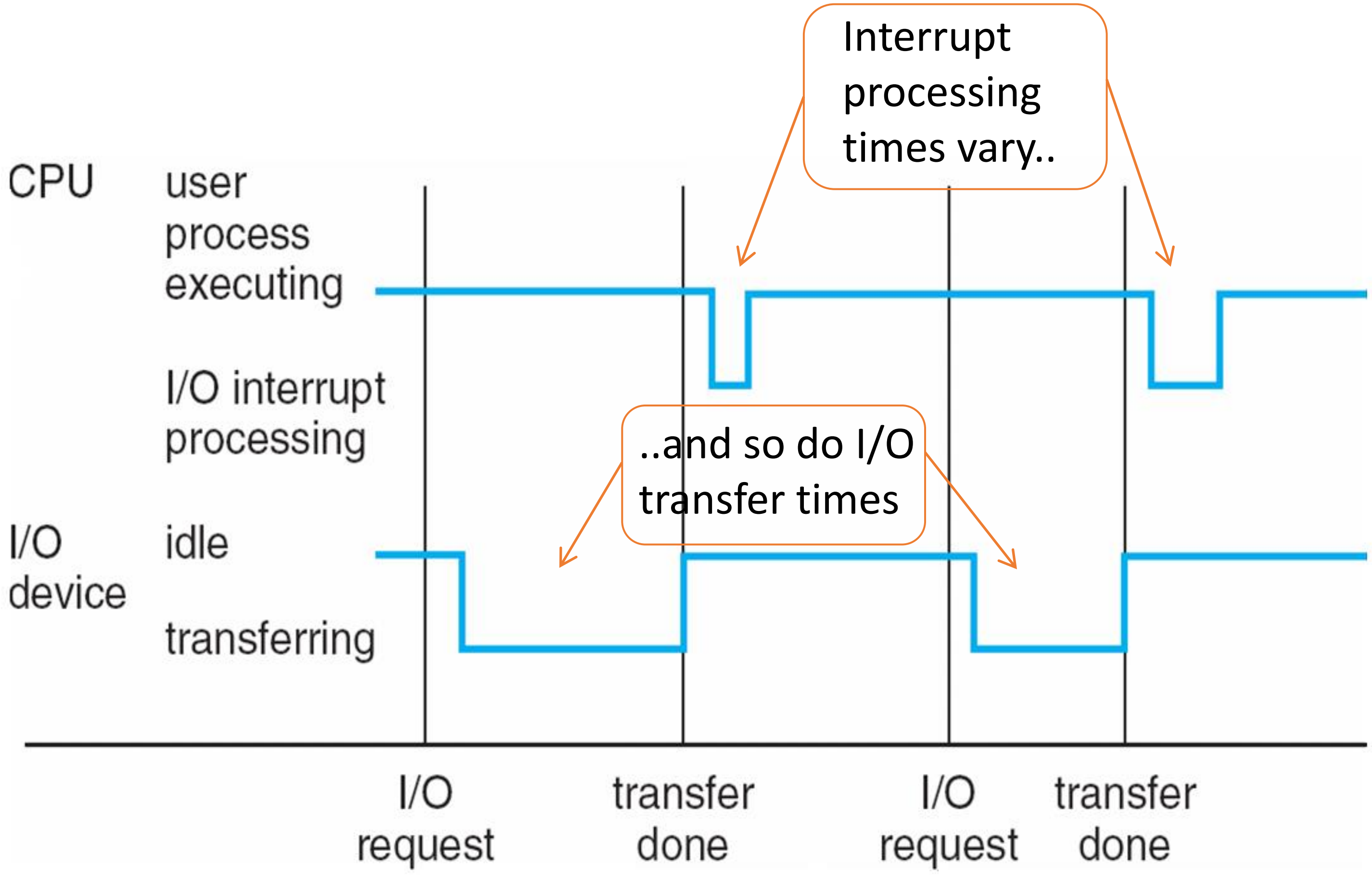
Common Functions of Interrupts

- 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 Handling

- The operating system saves 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

Interrupt Timeline for a single process doing output

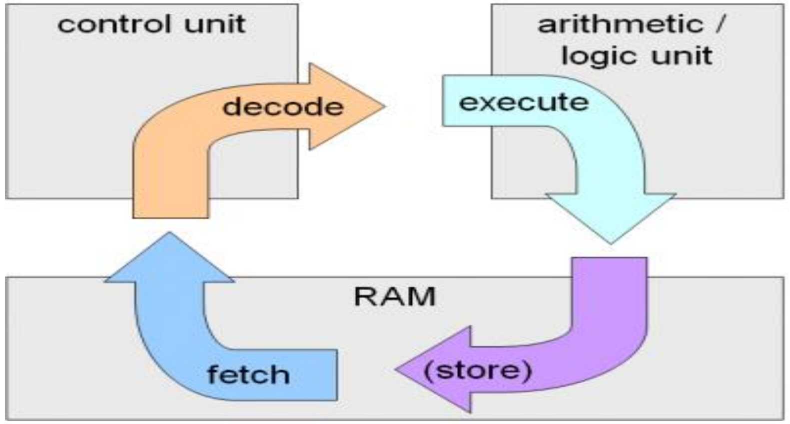


Storage Structure

- Main memory – only large storage media that the CPU can access directly (**Random access memory** and typically **volatile)**
  - Implemented with semiconductor technology called DRAM
- Computers use other forms of memory like ROM, EEPROM
- Smart phones have EEPROM to store factory installed programs.

Program Execution Model

- Typical instruction execution



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

Image source :https://computersciencewiki.org/index.php/The_machine_instruction_cycle

Storage Structure

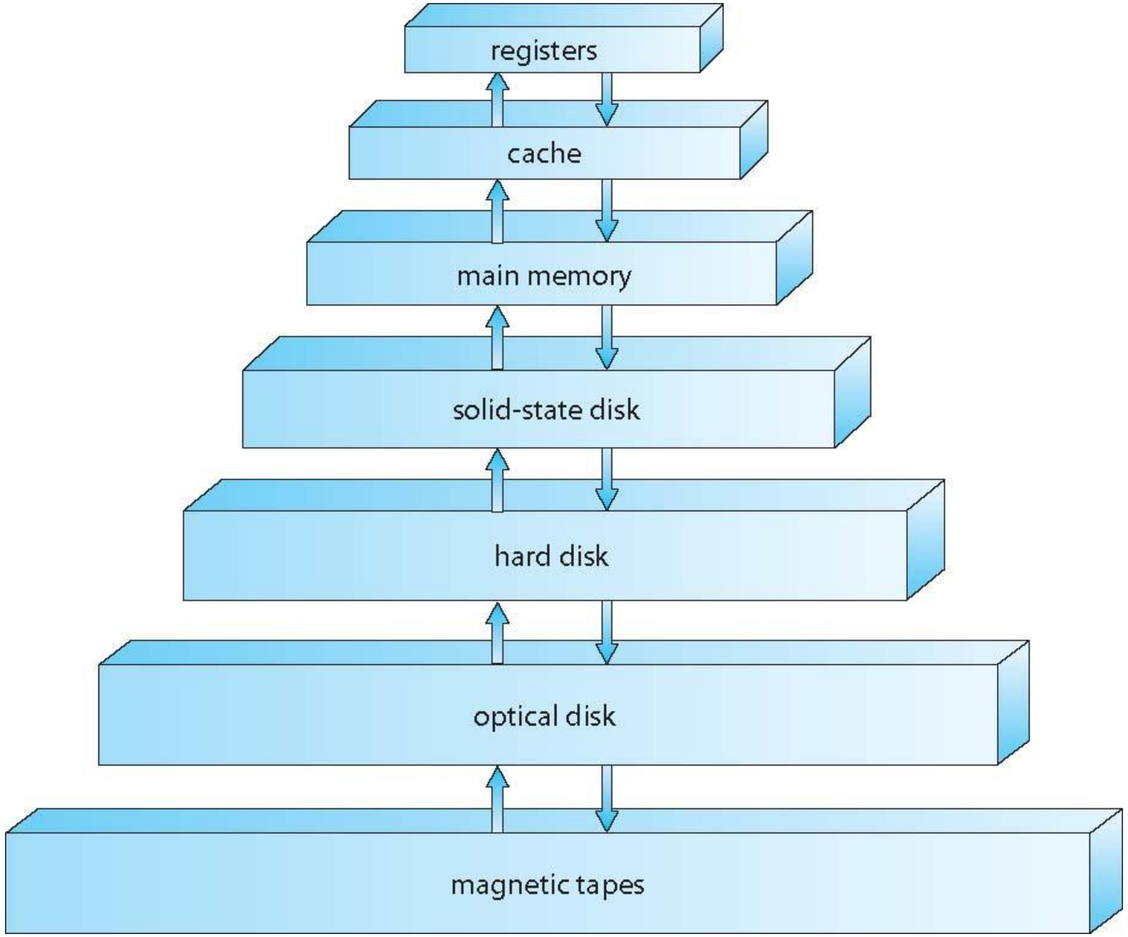
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- Hard disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Solid-state disks** – faster than hard disks, nonvolatile
  - Various technologies and becoming more popular
  - Flash memory used in camera's PDA's

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

OPERATING SYSTEMS

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

I/O Structure

- Storage is a type of I/O device
- A large portion of operating-system code is dedicated to managing I/O
    - As reliability and performance of a system is the main concern.
- General-purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- Each device controller is in charge of a specific type of device.
- **Small Computer-Systems Interface (SCSI)** controller enables to connect more devices.

I/O Structure

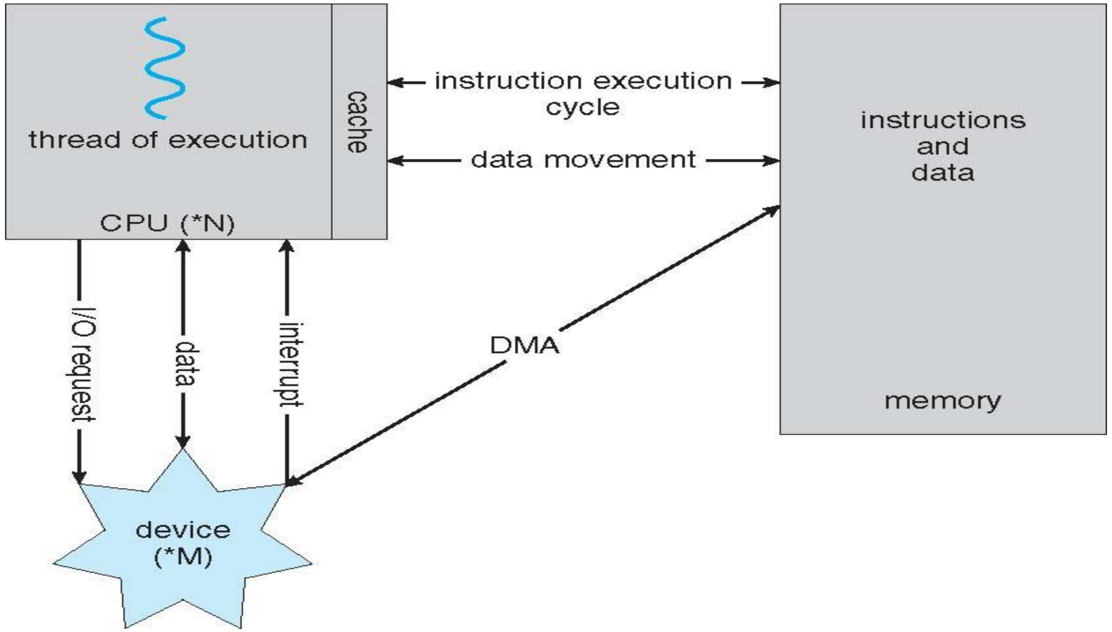
- A device controller maintains some local buffer storage and a set of special-purpose registers.
- The device controller is responsible for moving the data between the peripheral devices.

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.

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



THANK YOU

Chandravva Hebbi
Department of Computer Science Engineering
chandravvahebbi@pes.edu