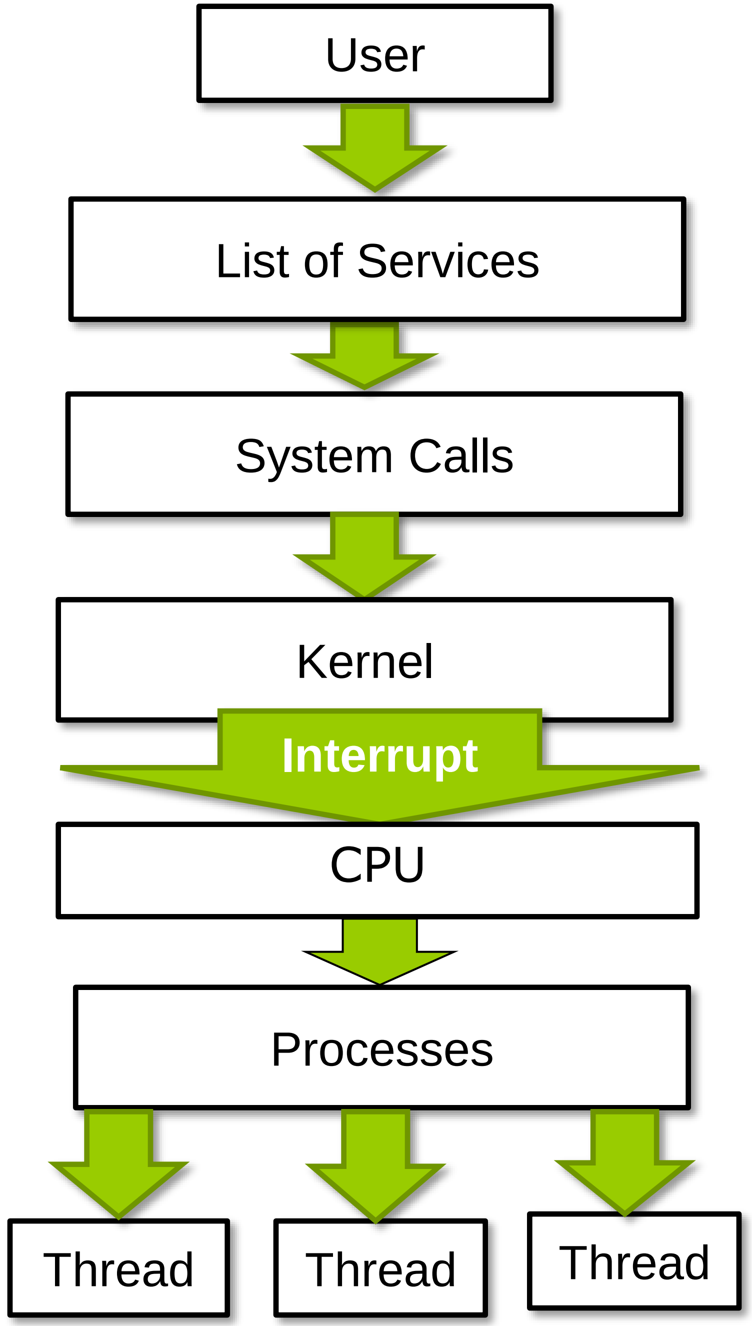
Part 2

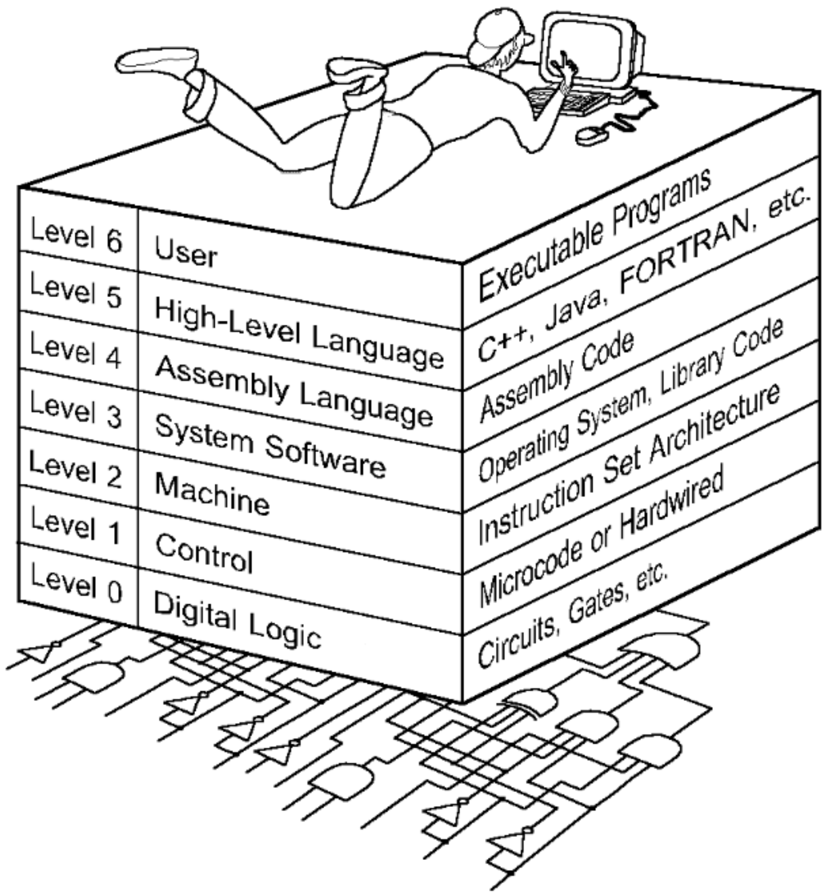
The Big Pic of OS



The Computer Level Hierarchy

- Computers consist of many things besides chips.
- Before a computer can do anything worthwhile, it must also use software.
- Writing complex programs requires a **“divide and conquer**” approach, where each program module solves a smaller problem.
- Complex computer systems employ a similar technique through a series of virtual machine layers.

- Each virtual machine layer is an abstraction of the level below it.
- The machines at each level execute their own particular instructions, calling upon machines at lower levels to perform tasks as required.
- Computer circuits ultimately carry out the work.



- Level 6: The User Level
  - Program execution and user interface level.
  - The level with which we are most familiar.
- Level 5: High-Level Language Level
  - The level with which we interact when we write programs in languages such as C, Pascal, Lisp, Python, and Java.

- Level 4: Assembly Language Level
  - Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.
- Level 3: System Software Level
  - Controls executing processes on the system.
  - Protects system resources.
  - Assembly language instructions often pass through Level 3 without modification.

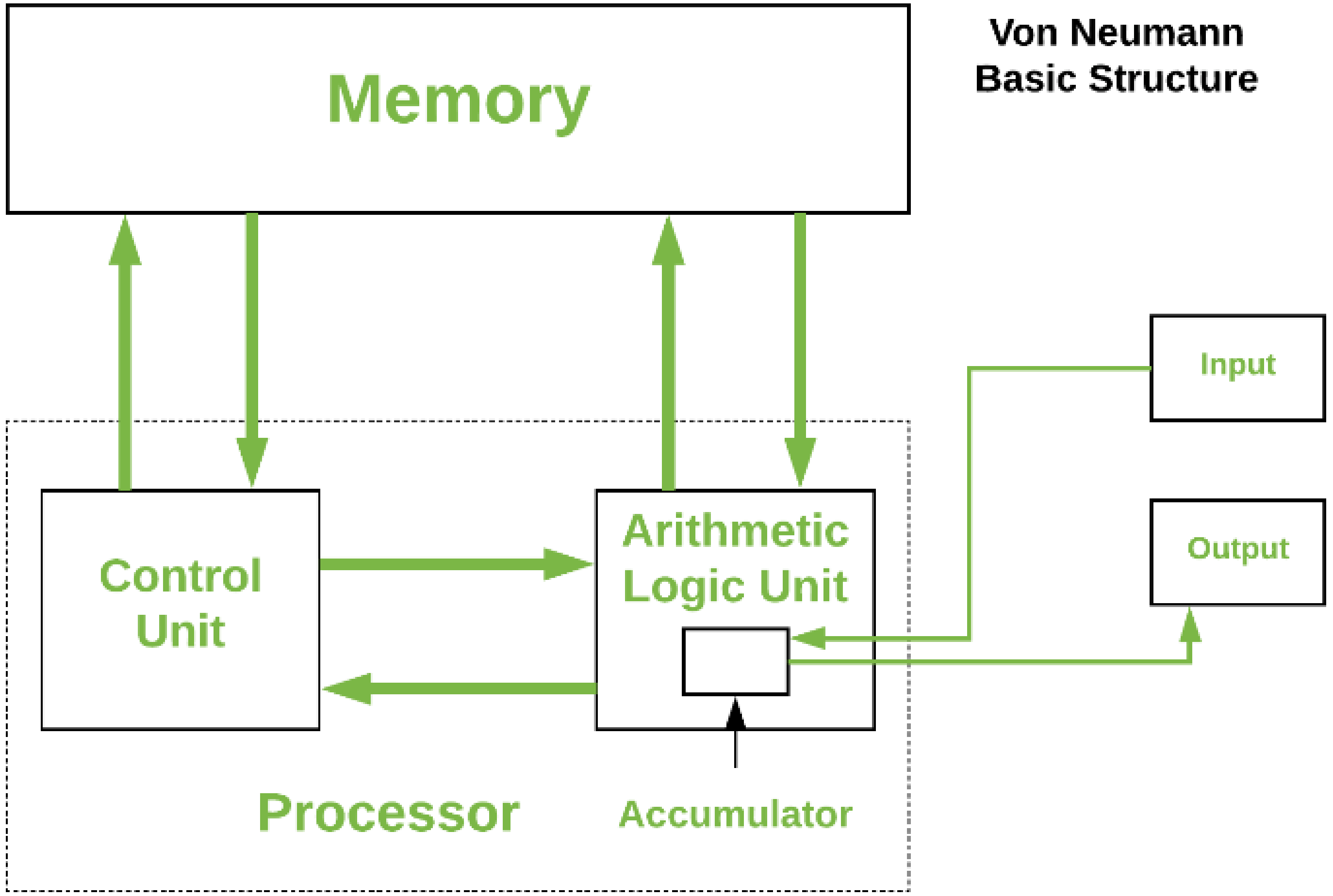
- Level 2: Machine Level
  - Also known as the Instruction Set Architecture (ISA) Level.
  - Consists of instructions that are particular to the architecture of the machine.
  - Programs written in machine language need no compilers, interpreters, or assemblers.

- Level 1: Control Level
  - A *control unit* decodes and executes instructions and moves data through the system.
  - Control units can be *microprogrammed* or *hardwired*.
  - A microprogram is a program written in a low-level language that is implemented by the hardware.
  - Hardwired control units consist of hardware that directly executes machine instructions.

- Level 0: Digital Logic Level
  - This level is where we find digital circuits (the chips).
  - Digital circuits consist of gates and wires.
  - These components implement the mathematical logic of all other levels.

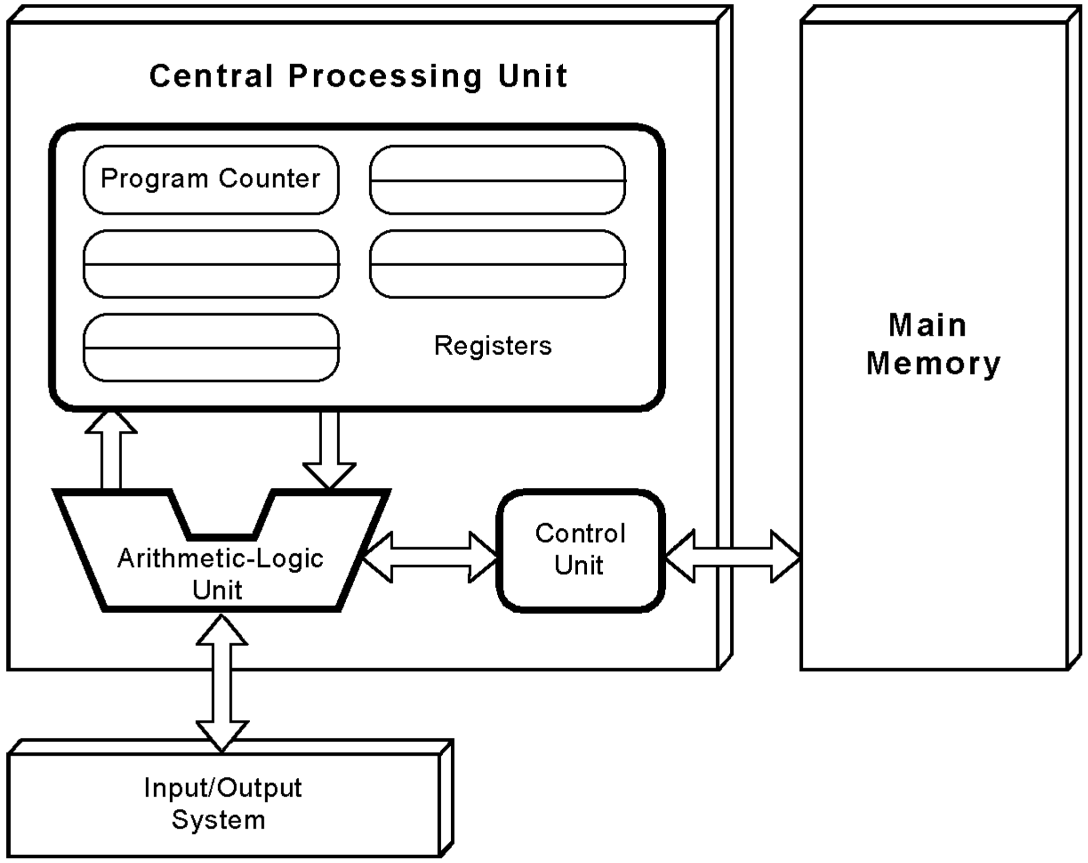
- Today’s stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system
  - The capacity to carry out sequential instruction processing.
  - A single data path between the CPU and main memory.
    - This single path is known as the *von Neumann bottleneck*.

Von Neumann
Basic Structure

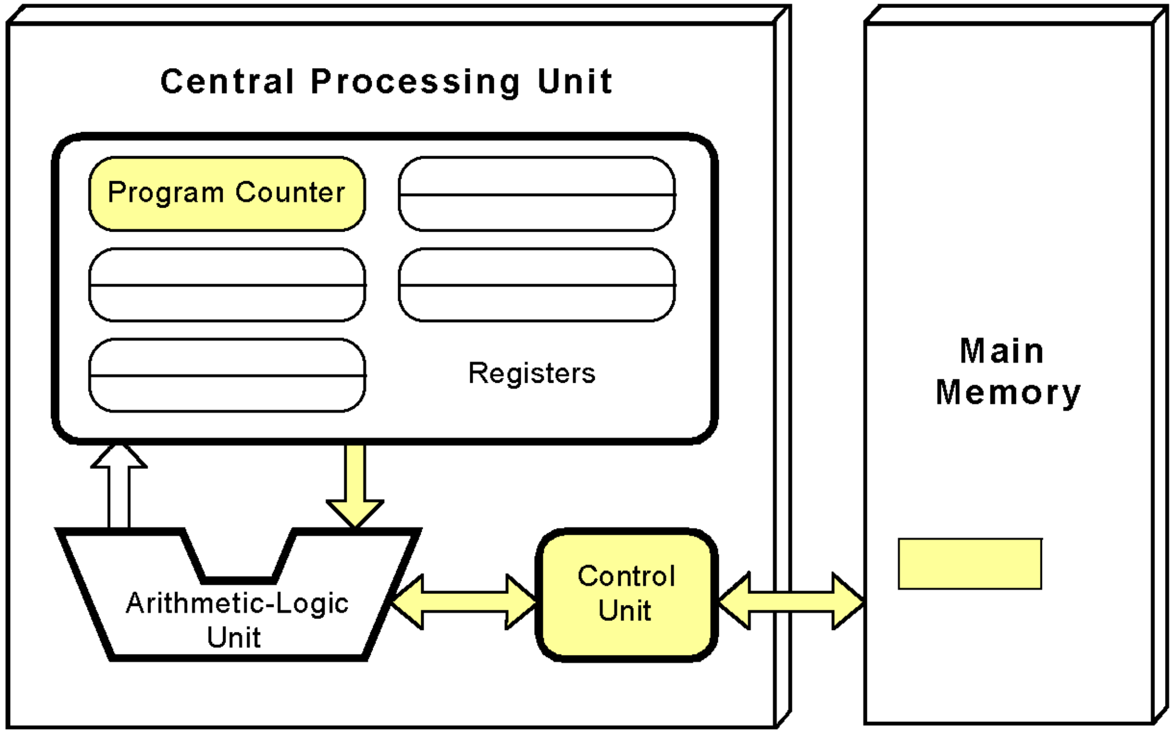


- This is a general depiction of a von Neumann system:

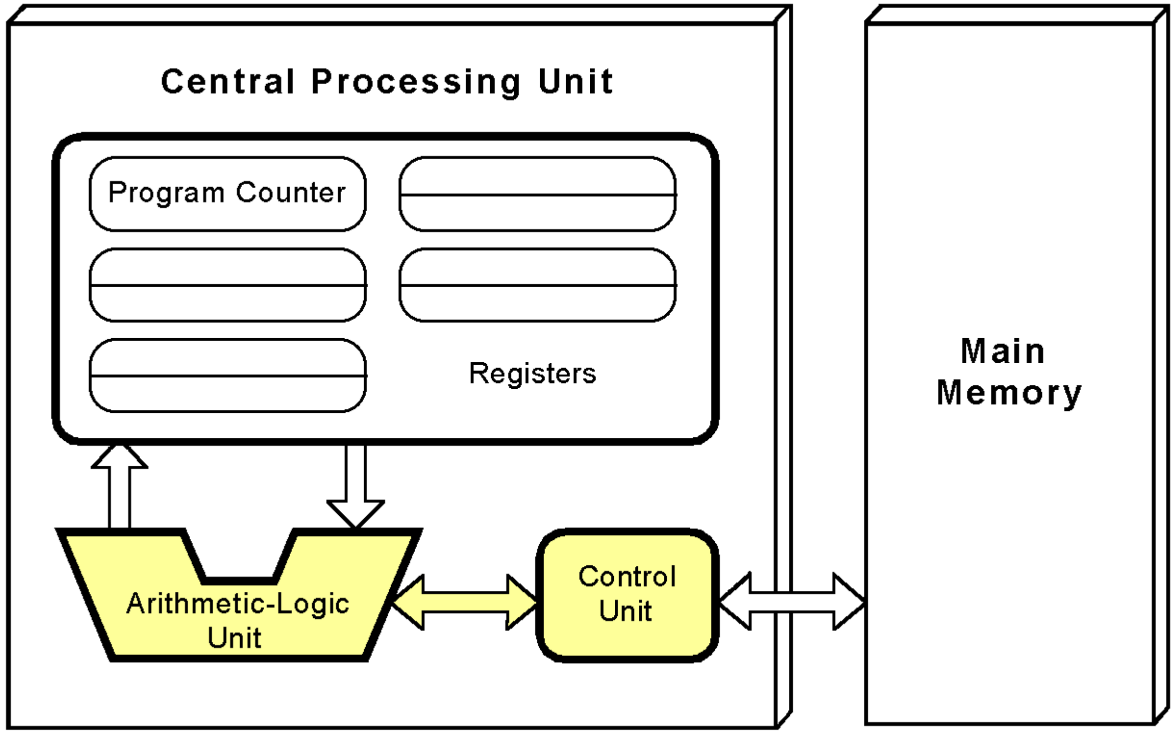
- These computers employ a fetch-decode-execute cycle to run programs as follows . . .



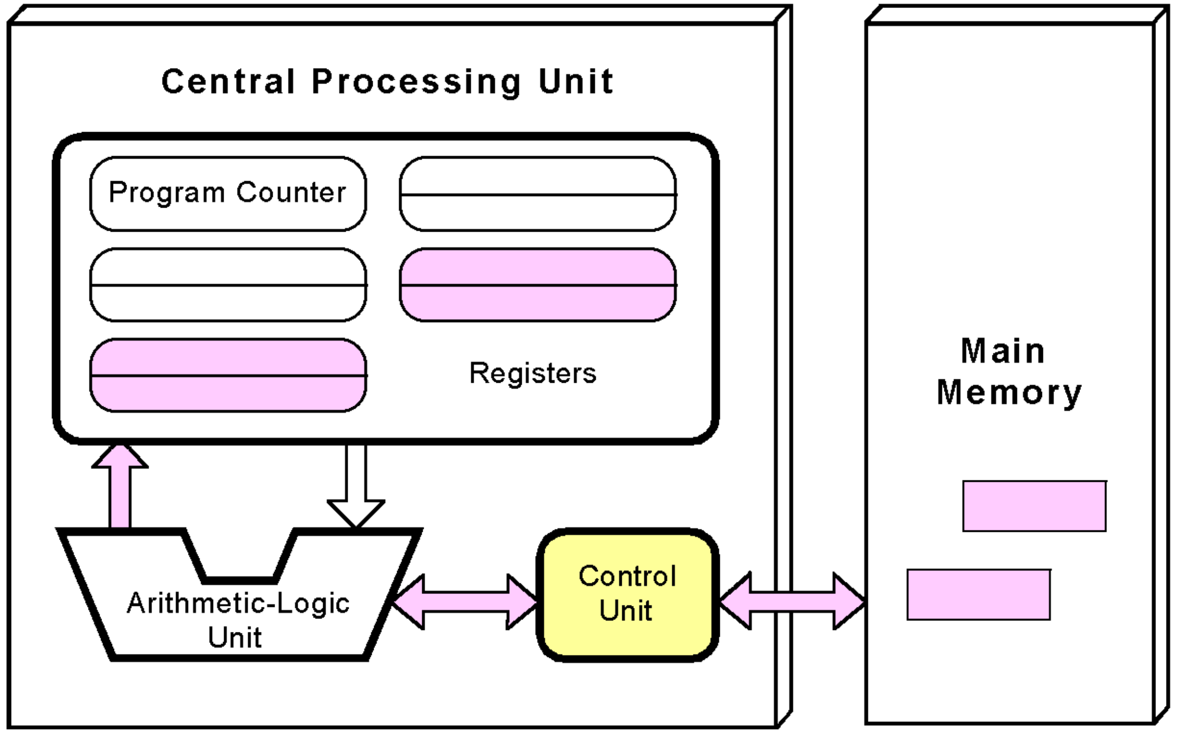
- The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.



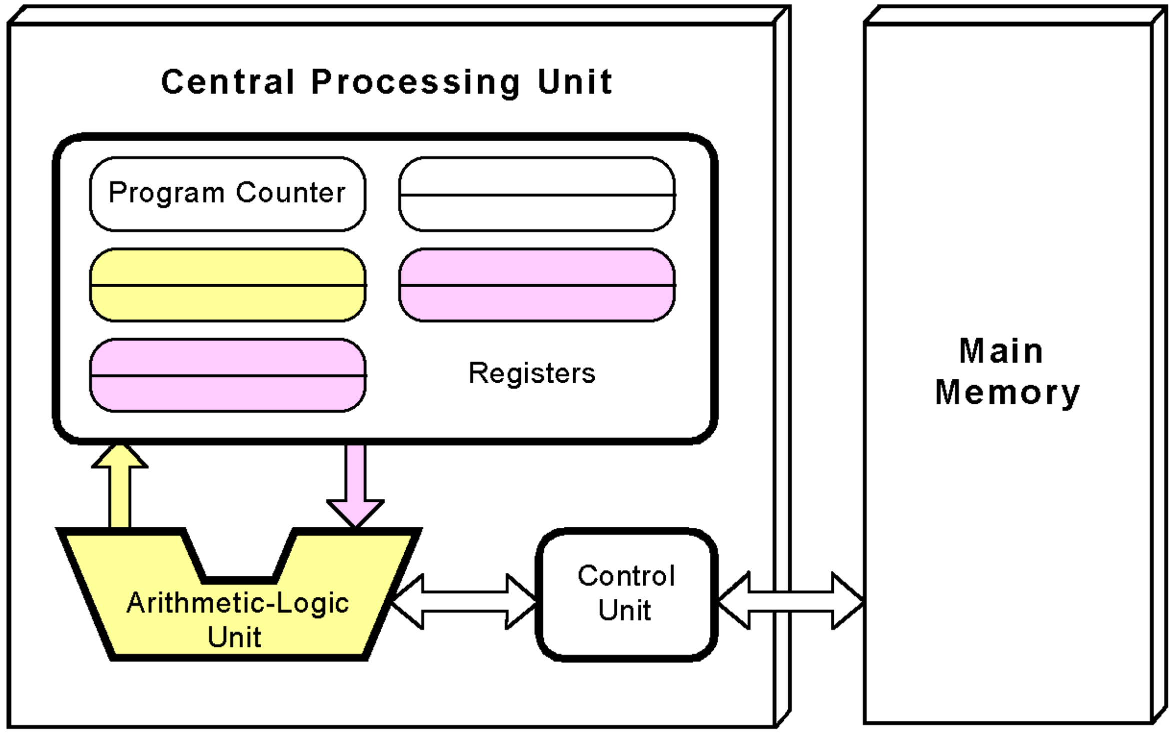
- The instruction is decoded into a language that the ALU can understand.



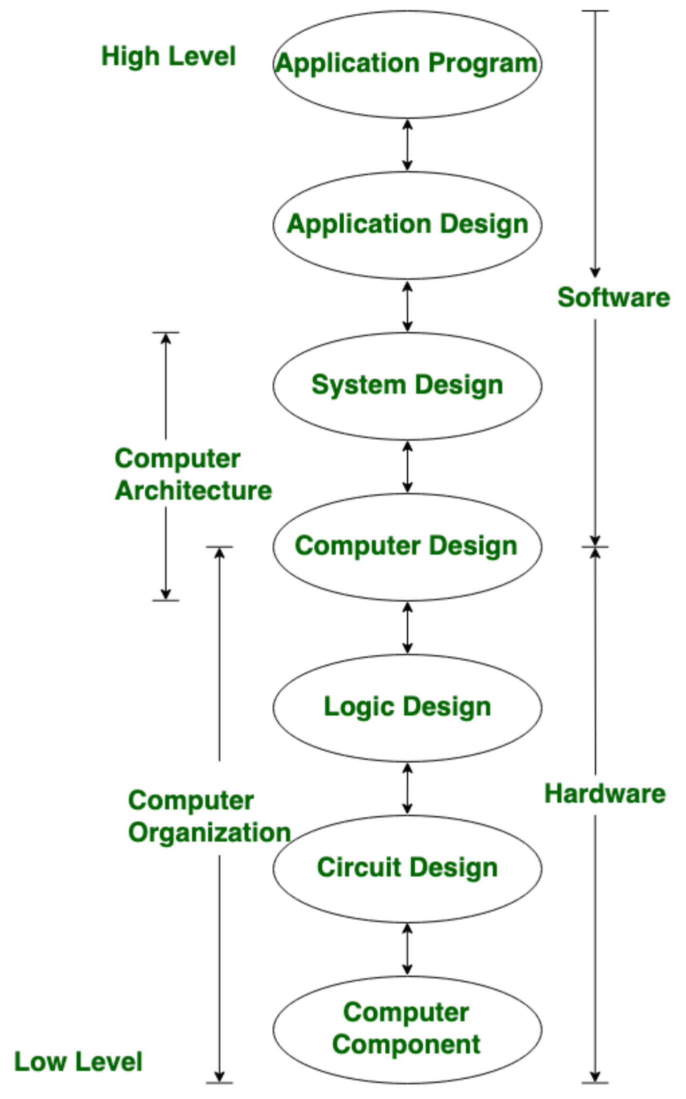
- Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.



- The ALU executes the instruction and places results in registers or memory.



- Conventional stored-program computers have undergone many incremental improvements over the years.
- These improvements include adding specialized buses, floating-point units, and cache memories, to name only a few.
- But enormous improvements in computational power require departure from the classic von Neumann architecture.
- Adding processors is one approach.



Computer System

Differences between Computer Architecture and Computer Organization

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
    - Operating system is a **resource allocator** and **control program** making efficient use of HW and managing execution of user programs

What Operating Systems Do

- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Mobile devices like smartphones and tables 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

Defining Operating Systems

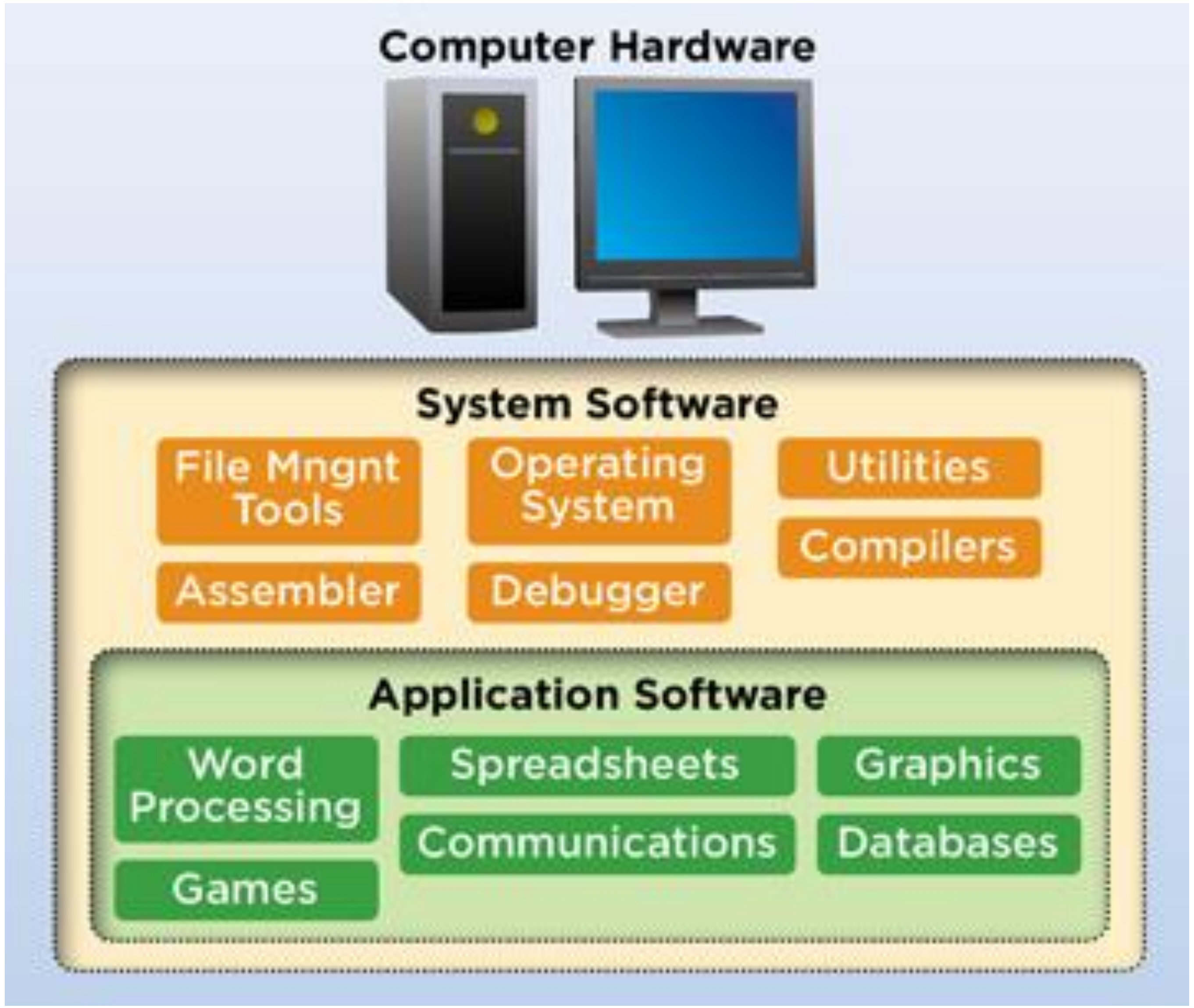
- Term OS covers many roles
  - Because of many designs and uses of OSes
  - Present in toasters through ships, spacecraft, game machines, TVs and industrial control systems
  - Born when fixed use computers for **military** became more general purpose and needed resource management and program control

Operating System Definition (Cont.)

- 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

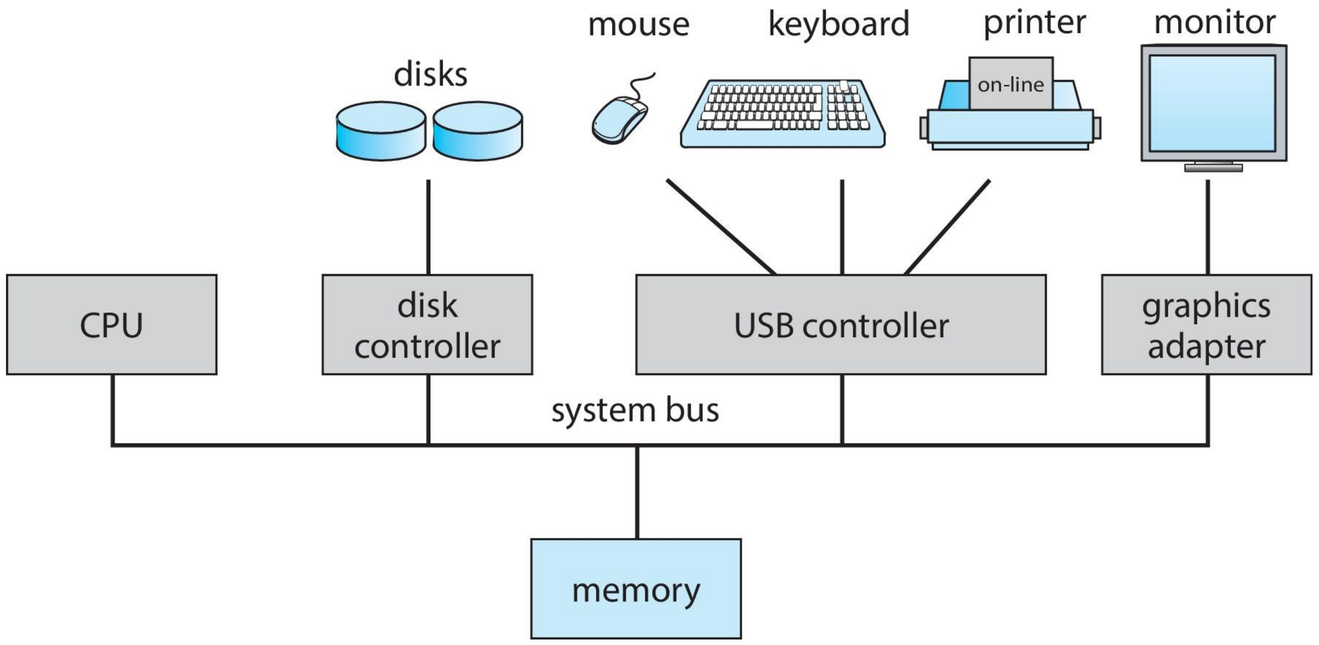
Operating System Definition (Cont.)

- 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 addition services to application developers such as databases, multimedia, graphics


Computer Hardware
System Software
File Mngnt Tools
Operating System
Utilities
Compilers
Assembler
Debugger
Application Software
Word Processing
Spreadsheets
Graphics
Communications
Databases
Games

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**
- Each device controller type has an **operating system device driver** to manage it
- 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**

Purpose of an Interrupt in Computer Organization

- **Interrupt** is the mechanism by which modules like I/O or memory may interrupt the normal processing by CPU. It may be either clicking a mouse, dragging a cursor, printing a document etc the case where interrupt is getting generated.

Why we require Interrupt?

- External devices are comparatively slower than CPU. So if there is no interrupt CPU would waste a lot of time waiting for external devices to match its speed with that of CPU.
- This decreases the efficiency of CPU. Hence, interrupt is required to eliminate these limitations.

With Interrupt:

- Suppose CPU instructs printer to print a certain document.
- While printer does its task, CPU engaged in executing other tasks.
- When printer is done with its given work, it tells CPU that it has done with its work.
  (The word ‘tells’ here is interrupt which sends one message that printer has done its work successfully.).

Advantages:

- It increases the efficiency of CPU.
- It decreases the waiting time of CPU.
- Stops the wastage of instruction cycle.

Disadvantages:

- CPU has to do a lot of work to handle interrupts, resume its previous execution of programs (in short, overhead required to handle the interrupt request.).

Difference between Interrupt and Polling

- **Interrupt:**
  Interrupt is a hardware mechanism in which, the device notices the CPU that it requires its attention.
- Interrupt can take place at any time.
- So when CPU gets an interrupt signal trough the indication interrupt-request line,
- CPU stops the current process and respond to the interrupt by passing the control to interrupt handler which services device.

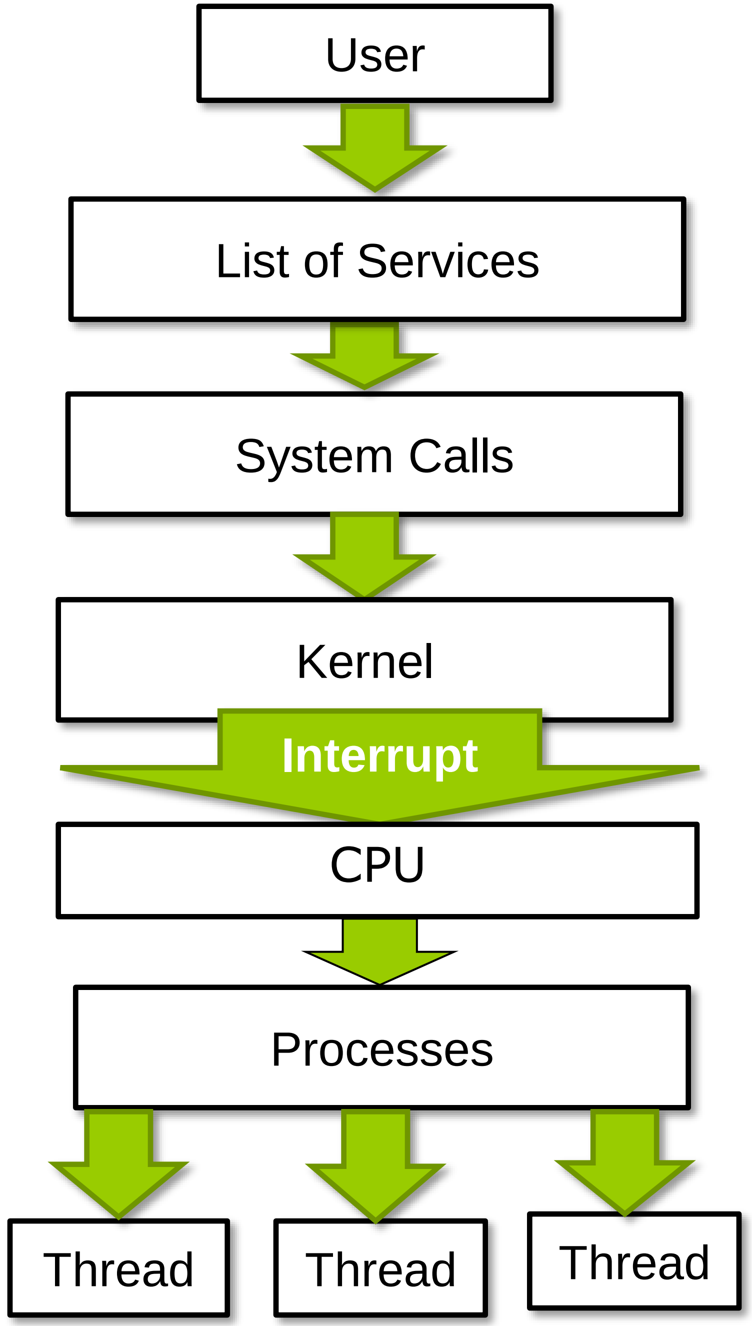
Polling:

- In polling is not a hardware mechanism, its a protocol in which CPU steadily checks whether the device needs attention.
- Wherever device tells process unit that it desires hardware processing, in polling process unit keeps asking the I/O device whether or not it desires CPU processing.
- The CPU ceaselessly check every and each device hooked up thereto for sleuthing whether or not any device desires hardware attention.

- Each device features a command-ready bit that indicates the standing of that device, i.e., whether or not it’s some command to be dead by hardware or not. If command bit is ready one, then it’s some command to be dead else if the bit is zero, then it’s no commands.

S.NO	INTERRUPT	POLLING
1.	In interrupt, the device notices the CPU that it requires its attention.	Whereas, in polling, CPU steadily checks whether the device needs attention.
2.	An interrupt is not a protocol, its a hardware mechanism.	Whereas it isn't a hardware mechanism, its a protocol.
3.	In interrupt, the device is serviced by interrupt handler.	While in polling, the device is serviced by CPU.
4.	Interrupt can take place at any time.	Whereas CPU steadily ballots the device at regular or proper interval.
5.	In interrupt, interrupt request line is used as indication for indicating that device requires servicing.	While in polling, Command ready bit is used as indication for indicating that device requires servicing.
6.	In interrupts, processor is simply disturbed once any device interrupts it.	On the opposite hand, in polling, processor waste countless processor cycles by repeatedly checking the command-ready little bit of each device.

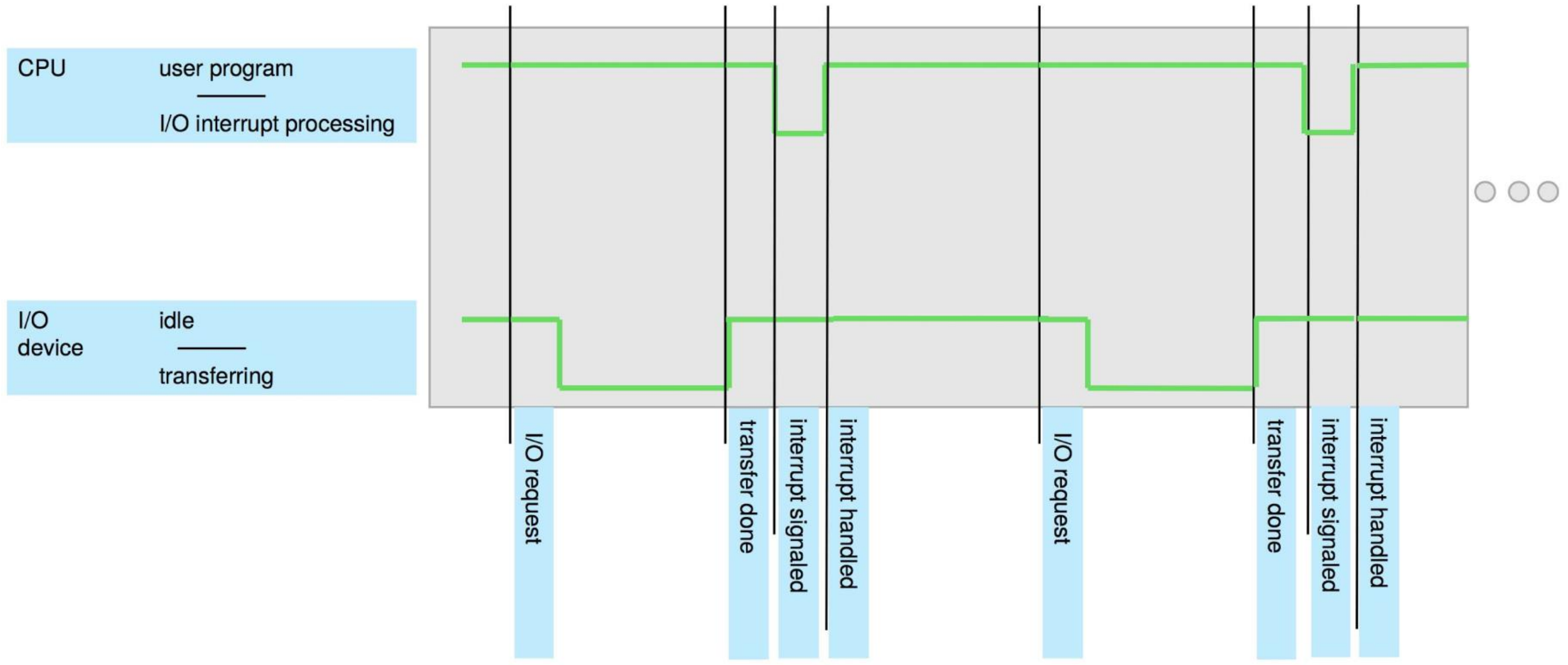
The Big Pic of OS



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**

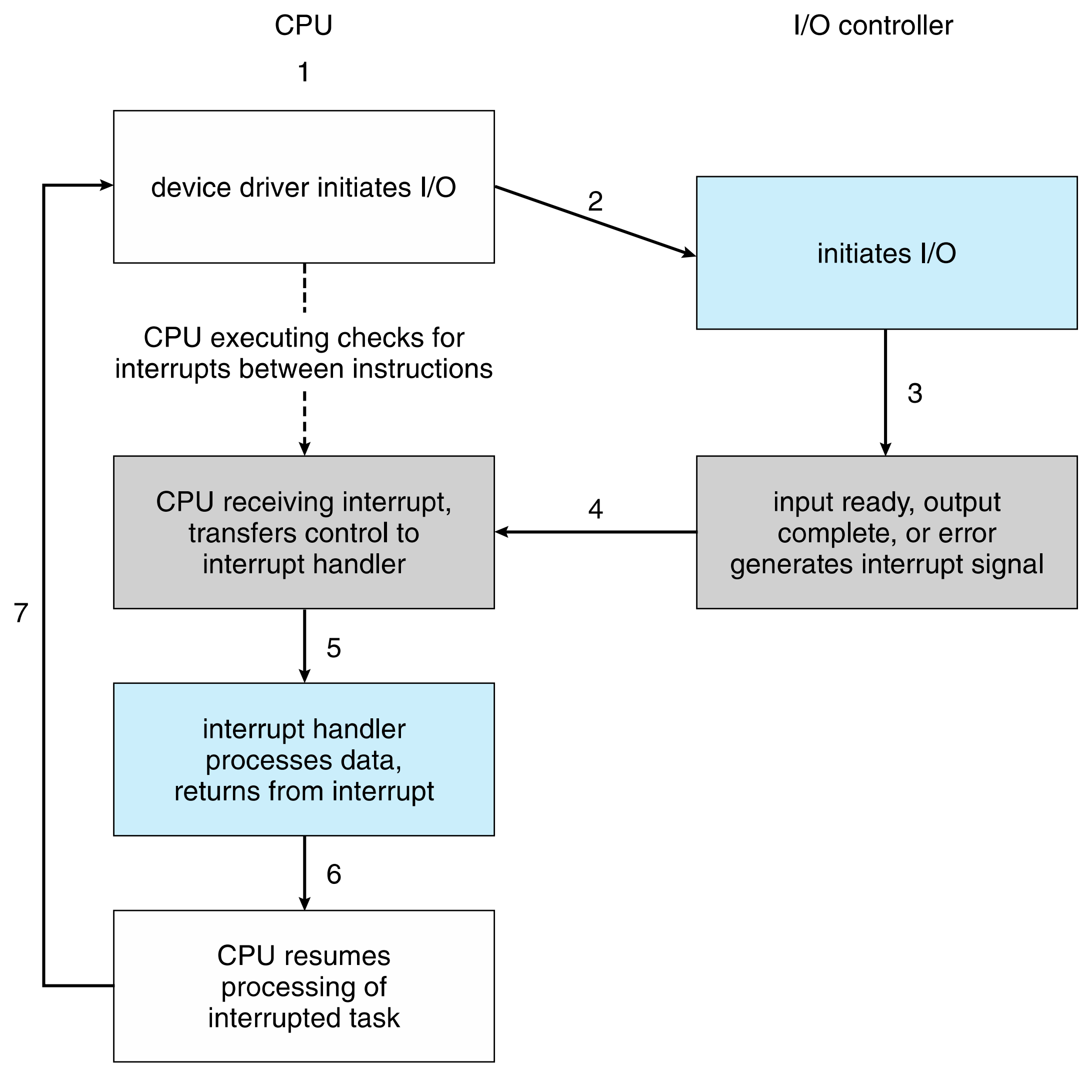
Interrupt Timeline



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

Interrupt-drive I/O Cycle


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

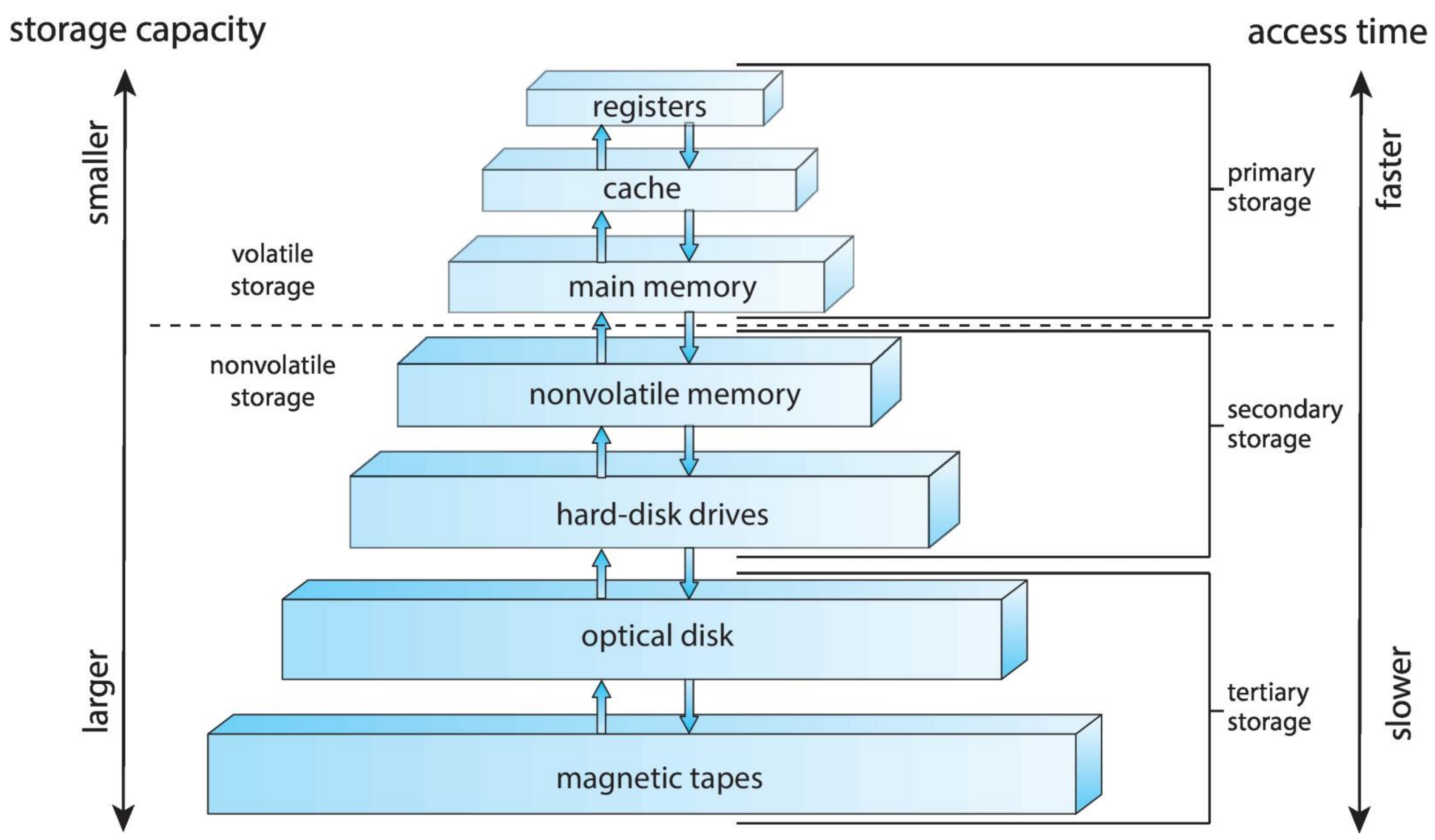
I/O Structure

- 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

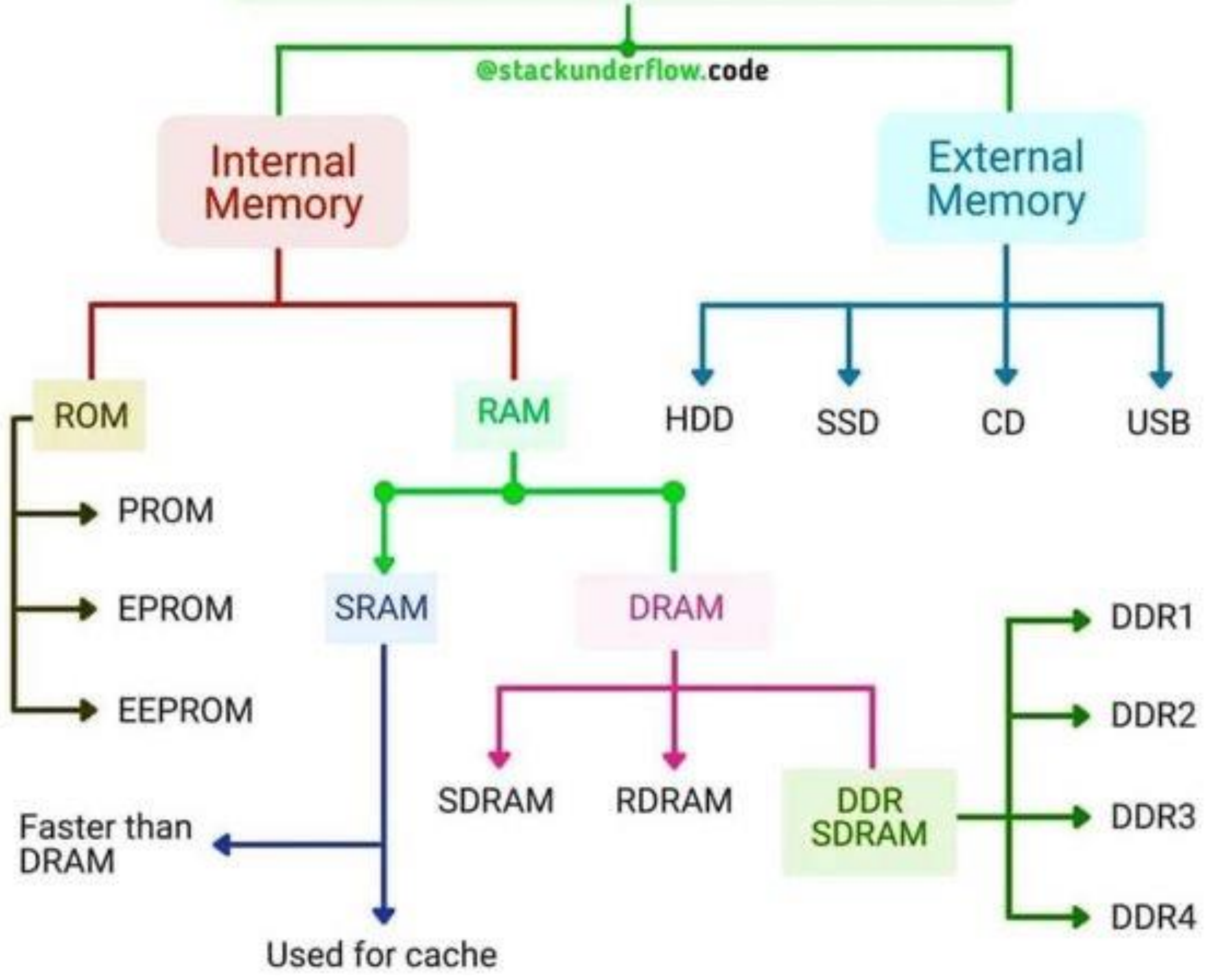
Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access**
  - Typically **volatile**
  - Typically **random-access memory** in the form of **Dynamic Random-access Memory (DRAM)**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
- **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
  - Becoming more popular as capacity and performance increases, price drops

Storage-Device Hierarchy



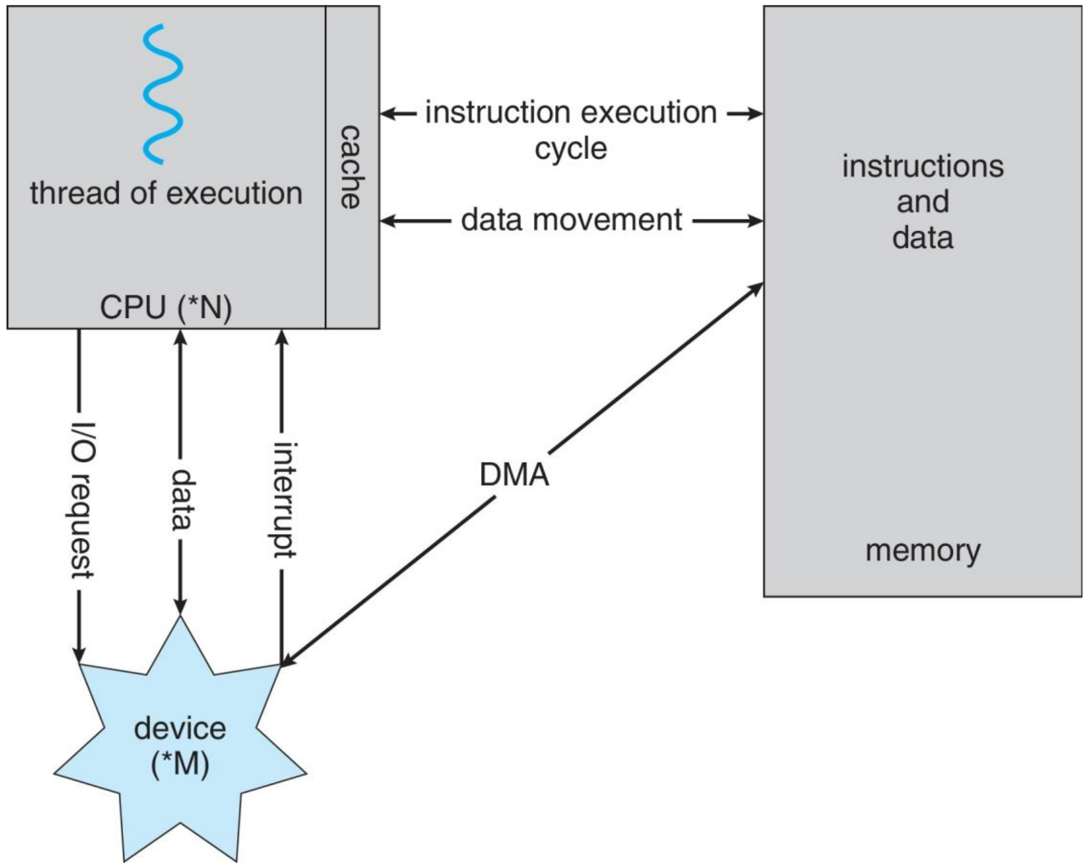
TYPES OF Computer Memory



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

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

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

RAM vs. ROM

- RAM (Random Access Memory) and ROM (Read-Only Memory) are two types of computer memory that serve different purposes in a computer system. Here are the key differences between them:

Volatility:

1. RAM is volatile memory, which means it loses its data when the computer is powered off or restarted. It is used for temporary data storage while the computer is running.
2. ROM is non-volatile memory, meaning it retains its data even when the computer is turned off. It stores essential data and instructions that are required for booting up the computer and initializing hardware components.

Function:

- RAM is used for temporary storage of data that the CPU (Central Processing Unit) actively uses during the execution of programs and tasks. It provides fast read and write access and is crucial for the computer's performance.
- ROM contains firmware or permanent software instructions that are integral to the computer's operation, such as the BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface). It is responsible for the initial boot-up of the computer and the loading of the operating system.

Read/Write Access:

- RAM is read-write memory, allowing data to be both read from and written to it. This makes it suitable for tasks like running applications, storing temporary files, and managing the computer's active processes.
- ROM is read-only memory, and its contents are typically programmed during the manufacturing process and cannot be easily altered or updated by the user. It is meant for storing permanent data and instructions.

Size and Capacity:

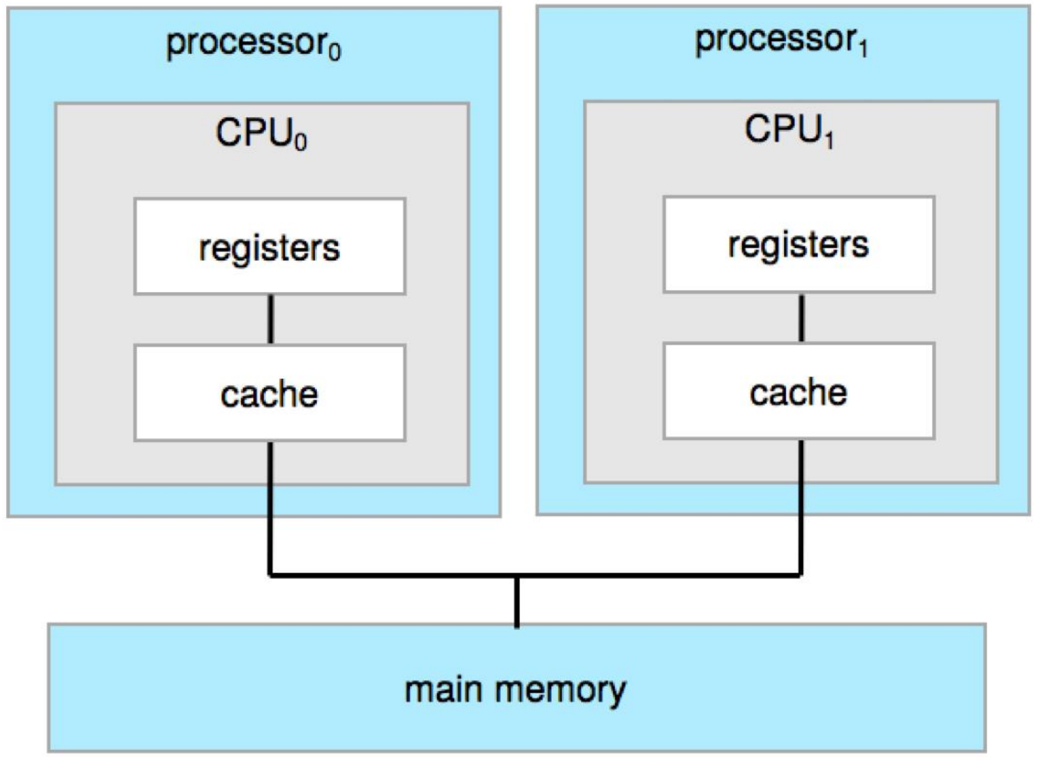
- RAM is usually larger in terms of capacity than ROM in modern computers. RAM capacity can range from a few gigabytes to several terabytes.
- ROM is typically smaller in capacity compared to RAM, as it stores essential firmware and boot-up instructions. It is usually measured in megabytes or gigabytes.

Access Speed:

- RAM offers much faster access speeds compared to ROM. This high-speed access allows the CPU to quickly read and write data during program execution.
- ROM provides slower access speeds compared to RAM because it contains firmware that is not meant to be frequently accessed during normal operation.

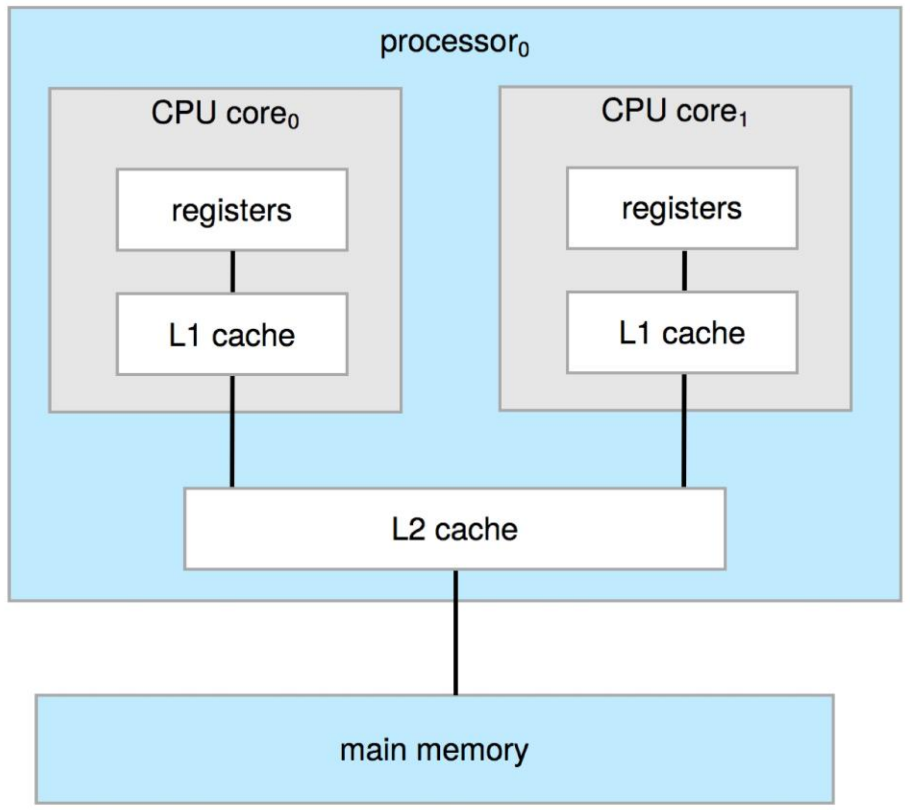
- RAM and ROM serve distinct roles in a computer system.
- RAM provides temporary storage for actively running programs and data, while ROM contains permanent instructions and firmware essential for booting up the computer and initializing hardware components.
- These differences in functionality, volatility, and access speeds make them essential components in modern computing devices.

Symmetric Multiprocessing Architecture

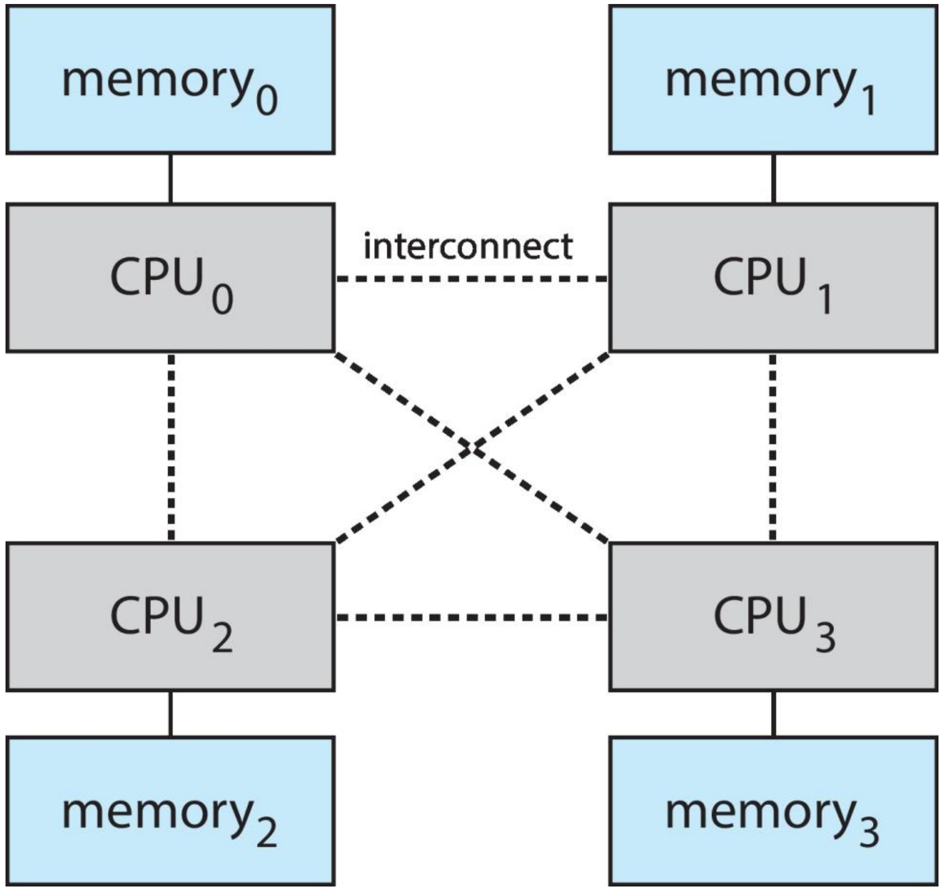


A Dual-Core Design

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



Non-Uniform Memory Access System



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