Part 2

An operating system (OS) consists of several components that work together to manage and control the operation of a computer system. Here are the key components of an operating system:

- Kernel: The kernel is the core component of the operating system. It provides essential services and manages the system's resources, including memory, processors, devices, and file systems. It acts as a bridge between the hardware and software layers of the system
- Process Management: This component is responsible for managing and executing processes or tasks. It includes features such as process scheduling, creation, termination, and synchronization to ensure efficient utilization of the CPU.

- Memory Management: The memory management component handles the allocation and deallocation of system memory to processes. It manages virtual memory, paging, swapping, and memory protection to optimize memory usage and provide a secure environment for processes.

- File System: The file system component manages the organization, storage, retrieval, and access of files on secondary storage devices such as hard drives or solid-state drives. It provides a hierarchical structure for organizing files and includes features like file permissions, directory management, and file metadata.

- Device Drivers: Device drivers are software modules that facilitate communication between the operating system and hardware devices such as printers, keyboards, monitors, and network interfaces. They enable the operating system to control and interact with these devices.

- User Interface: The user interface component provides a means for users to interact with the operating system. It can take the form of a command-line interface (CLI), where users enter text commands, or a graphical user interface (GUI), which presents a visual environment with icons, windows, and menus.

- Networking: Networking components enable the operating system to support network connectivity and communication. They include protocols, drivers, and services that allow the system to connect to networks, access the internet, and transfer data.

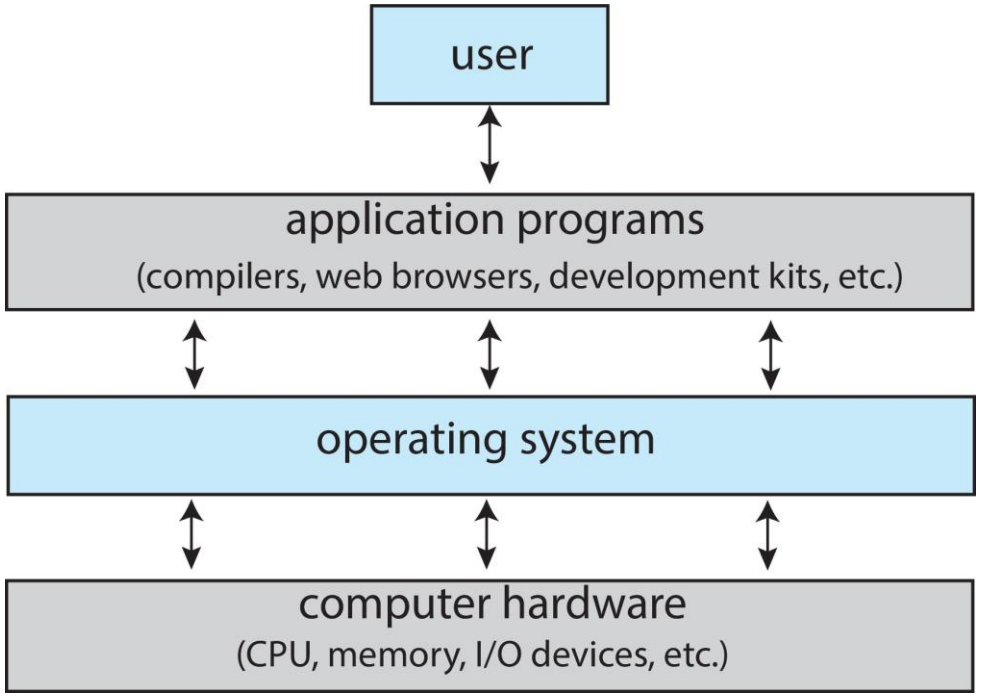
- Security: The security component ensures the protection of the system and its resources against unauthorized access, threats, and vulnerabilities. It includes features such as user authentication, access control, encryption, and firewall capabilities.

- File and Data Management: This component includes utilities and services for managing files, directories, and data on the system. It handles tasks like file organization, backup, recovery, and data integrity.

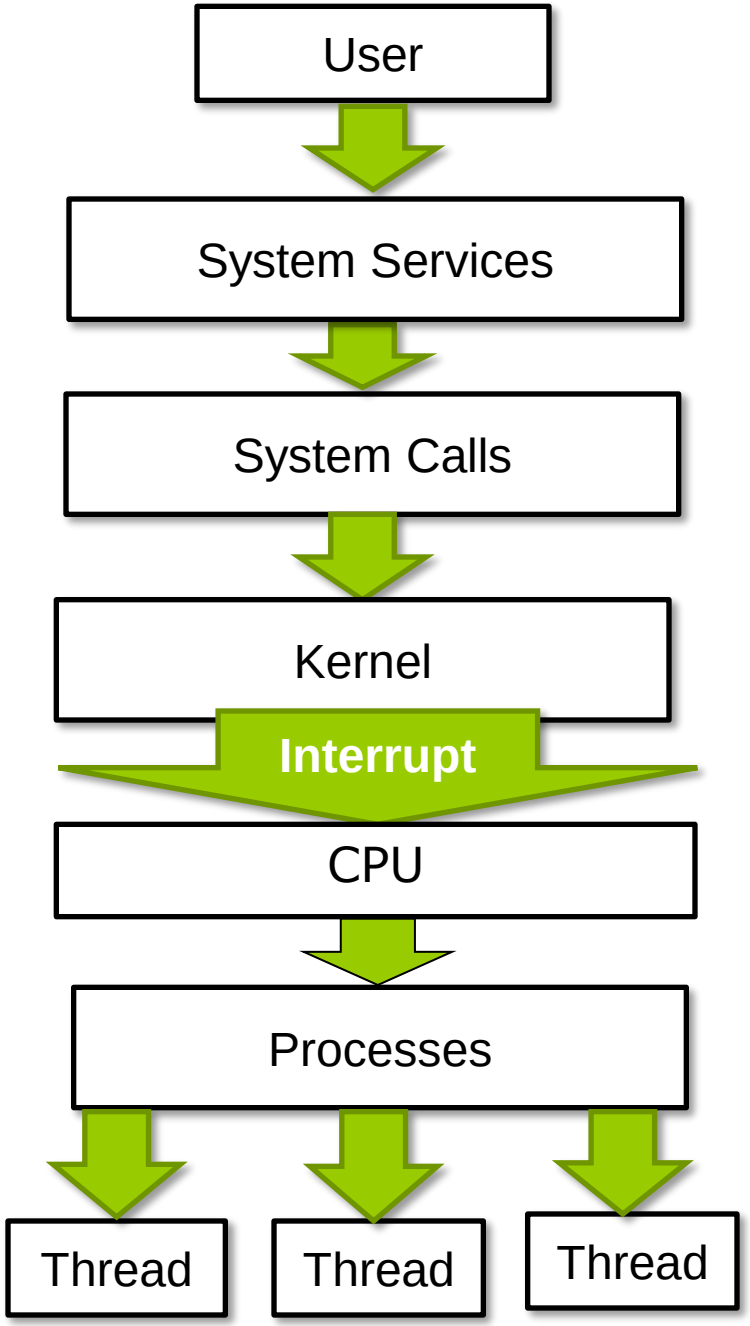
- System Libraries: System libraries are collections of precompiled code that provide common functions and services to applications. They allow programmers to leverage existing functionality without having to develop everything from scratch

- These components work together to provide a cohesive and functional operating system, allowing users to run applications, manage files, and utilize system resources efficiently.

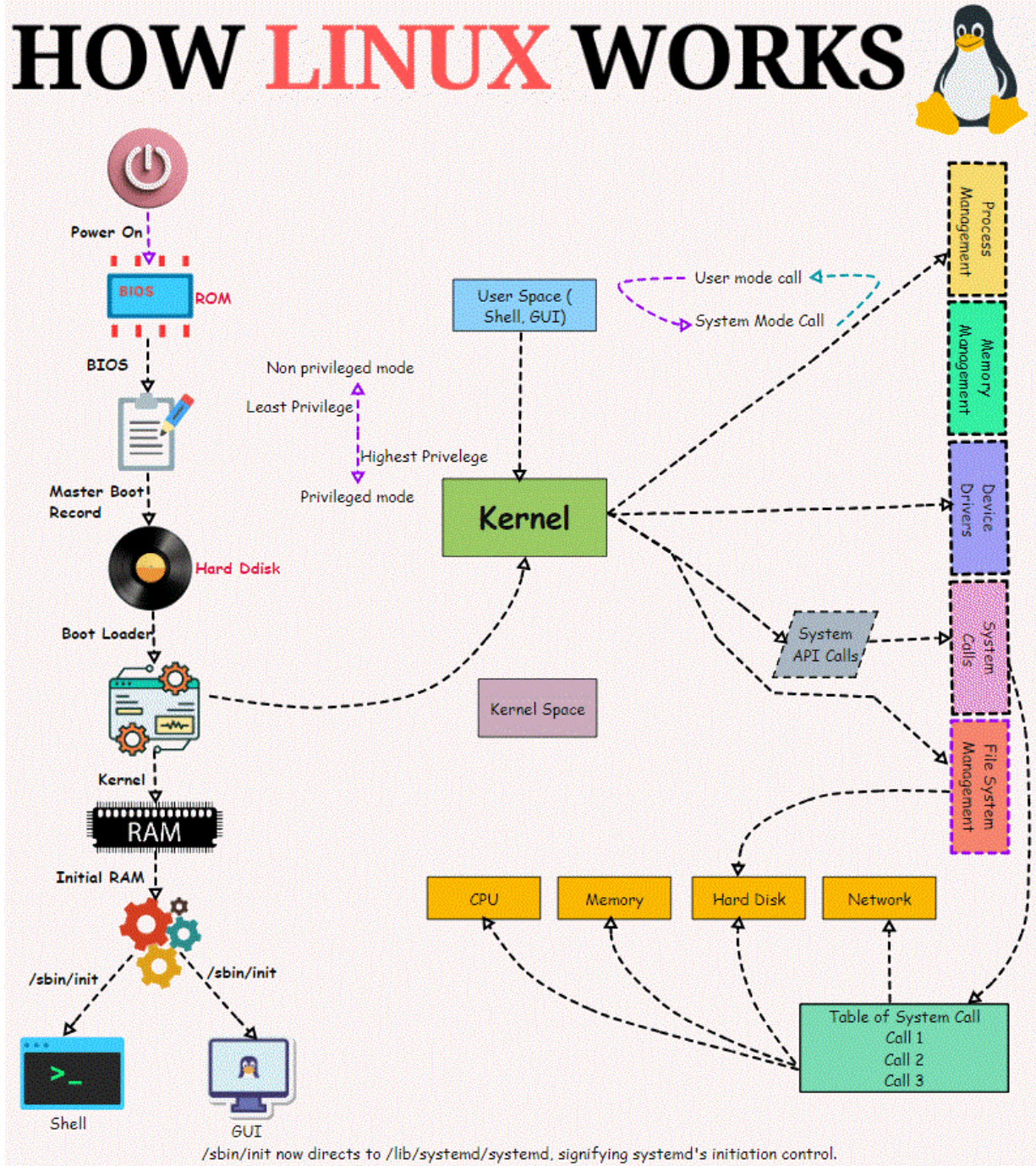
Abstract View of Components of Computer



The Big Pic of OS



HOW LINUX WORKS

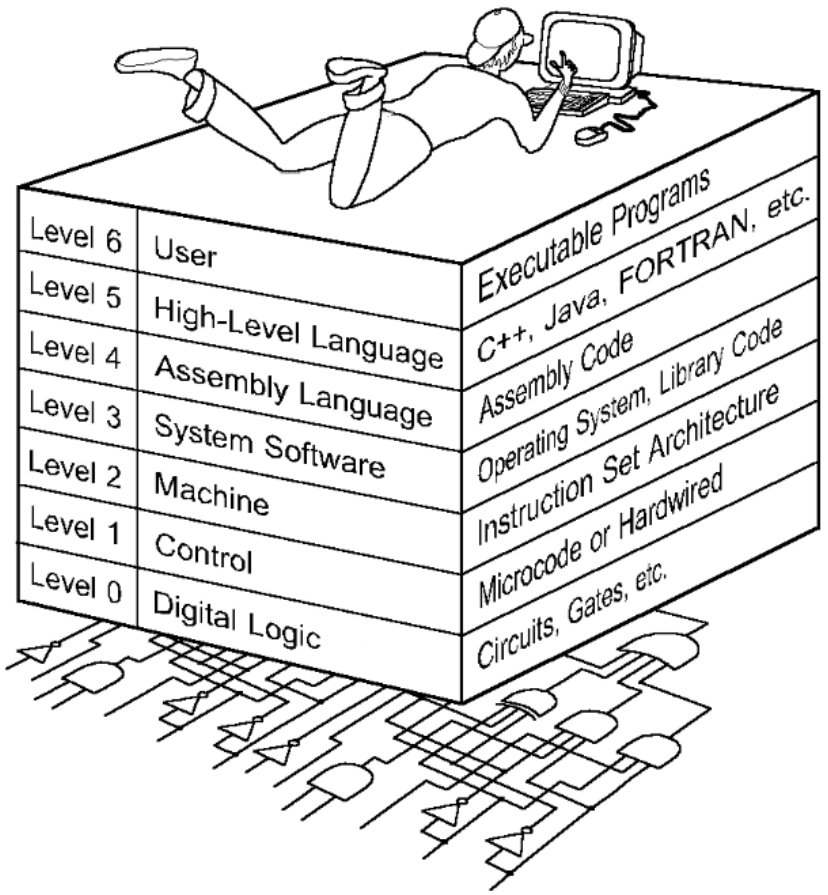


/sbin/init now directs to /lib/systemd/systemd, signifying systemd's initiation control.

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.

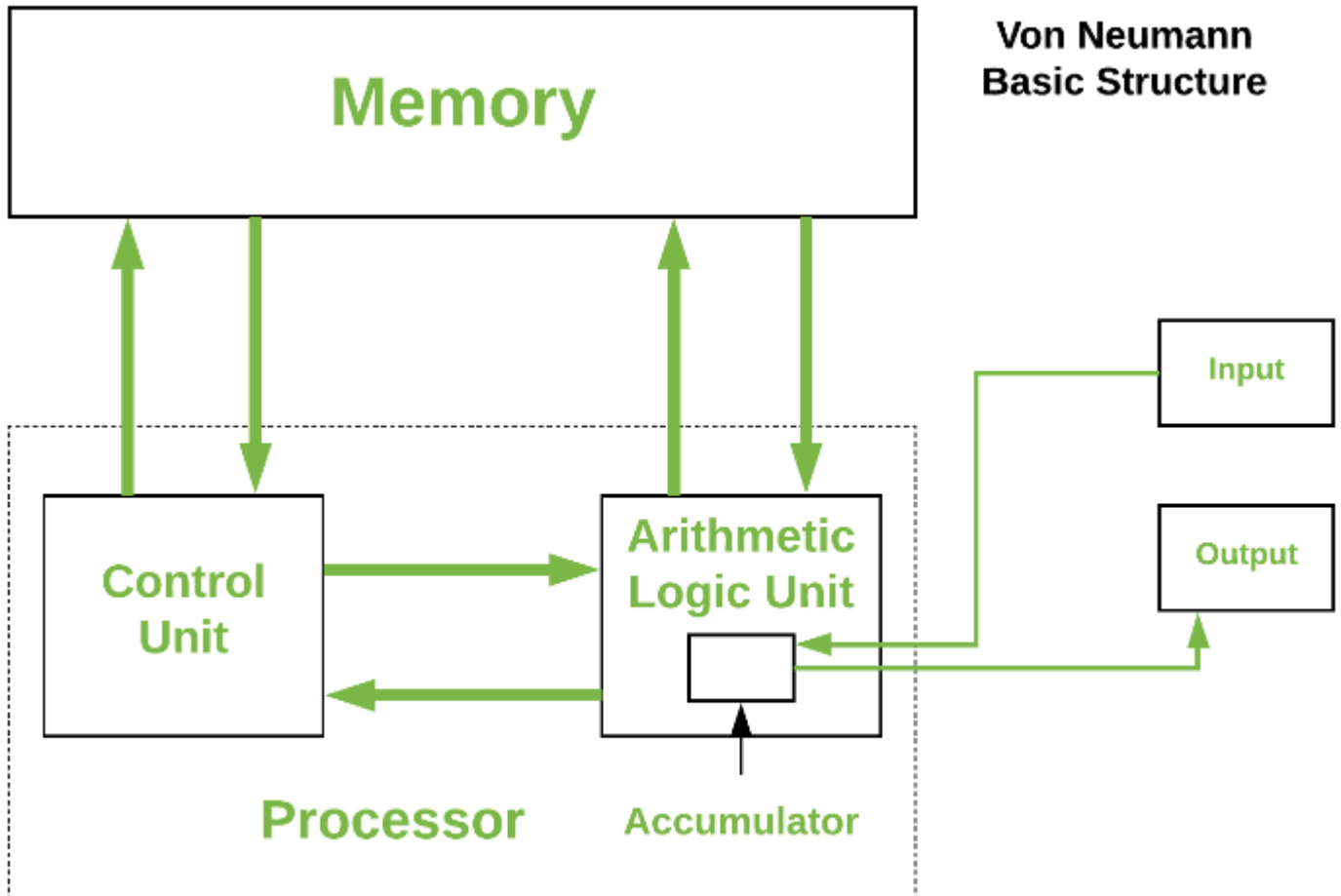
- Level 4: Assembly Language Level
  - Acts upon assembly language produced from Level 5, as well as instructions programmed directly at this level.
- Level 3: System Software Level
  - Controls executing processes on the system.
  - Protects system resources.
  - Assembly language instructions often 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.

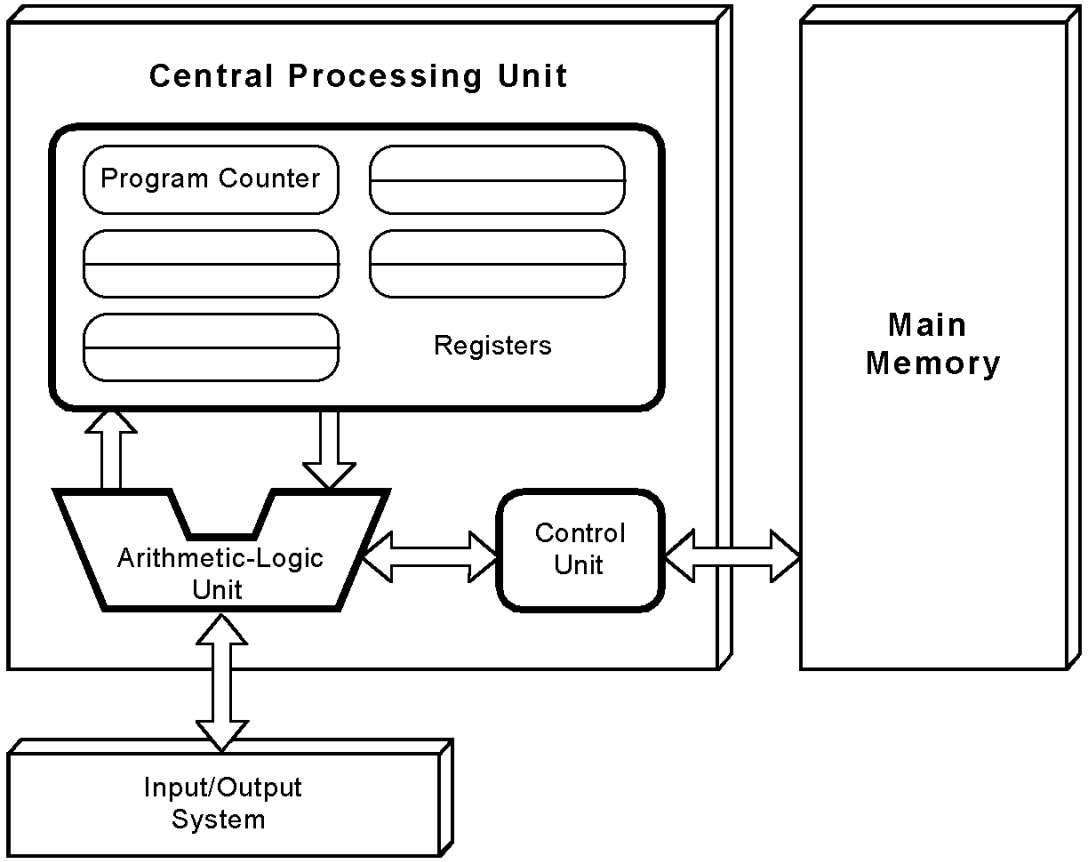
- Level 1: Control Level
  - A *control unit* decodes and executes instructions and moves data through the system.
  - Control units can be *microprogrammed* or *hardwired*.
  - A microprogram is a program written in a low-level language that is implemented by the hardware.
  - Hardwired control units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

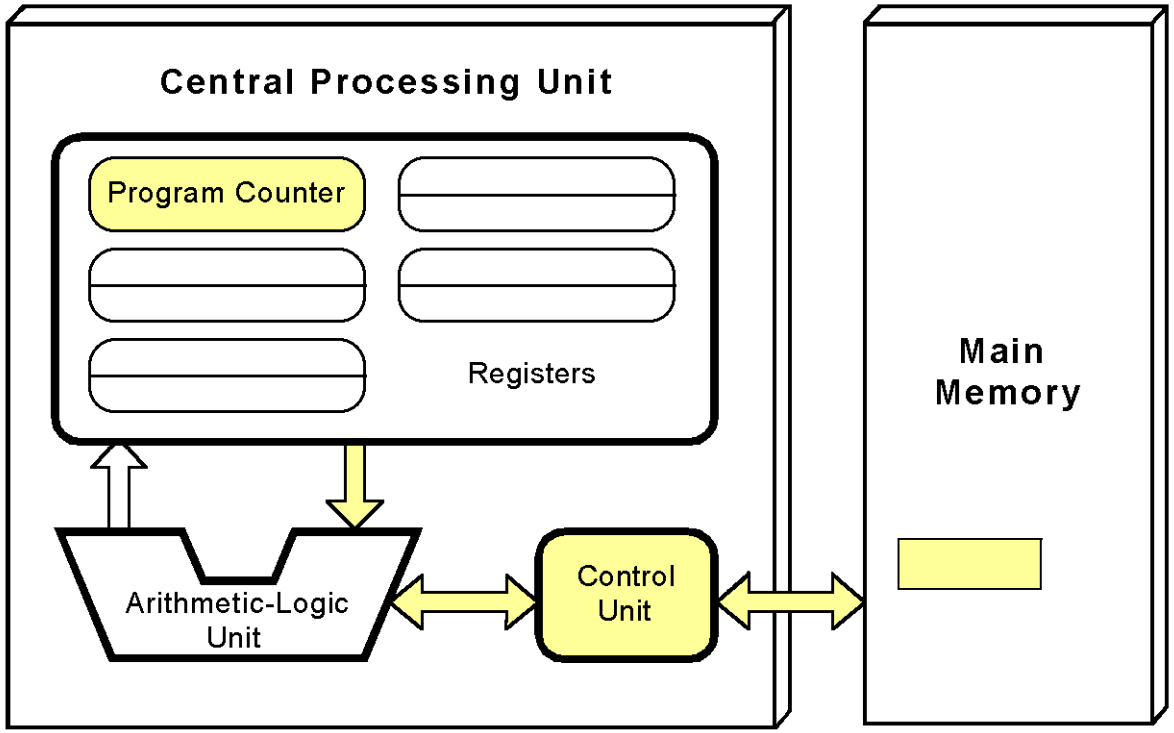
- Today's stored-program computers have the following characteristics:
  - Three hardware systems:
    - A central processing unit (CPU)
    - A main memory system
    - An I/O system
  - The capacity to carry out sequential instruction processing.
  - A single data path between the CPU and main memory.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


Memory
Von Neumann
Basic Structure
Input
Output
Control
Unit
Arithmetic
Logic Unit
Processor
Accumulator

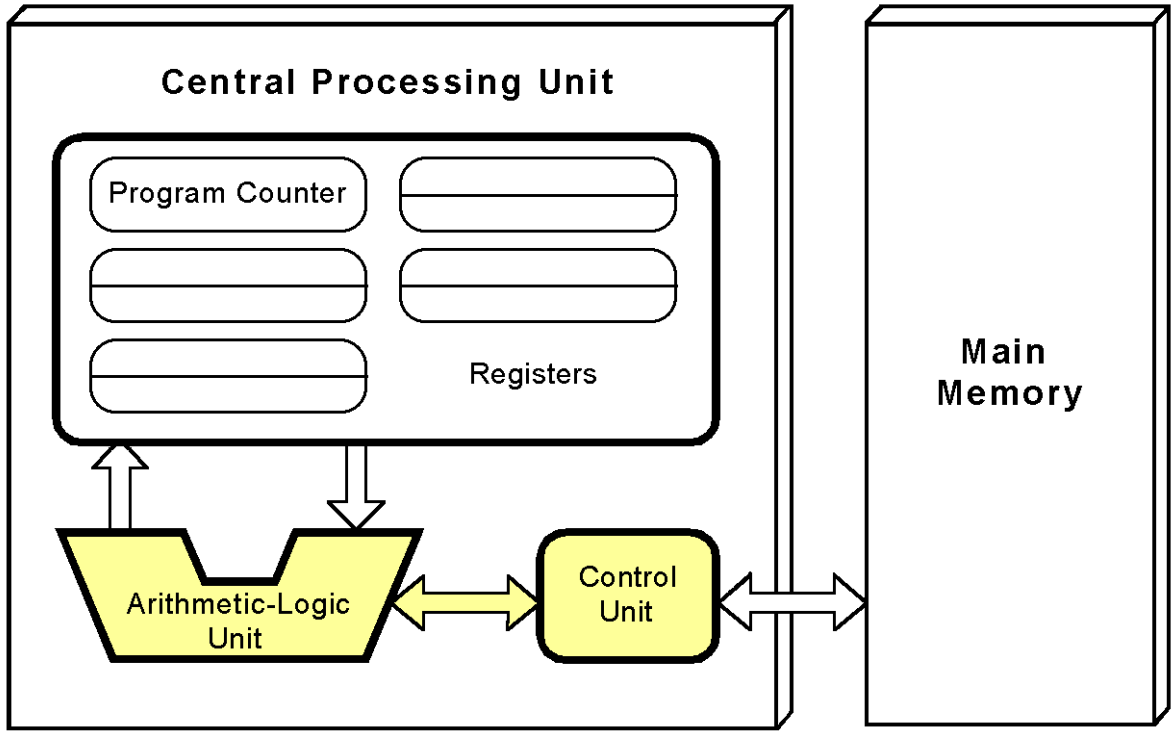
- **This is a general depiction of a von Neumann system:**
- **These computers employ a fetch-decode-execute cycle to run programs as follows . . .**



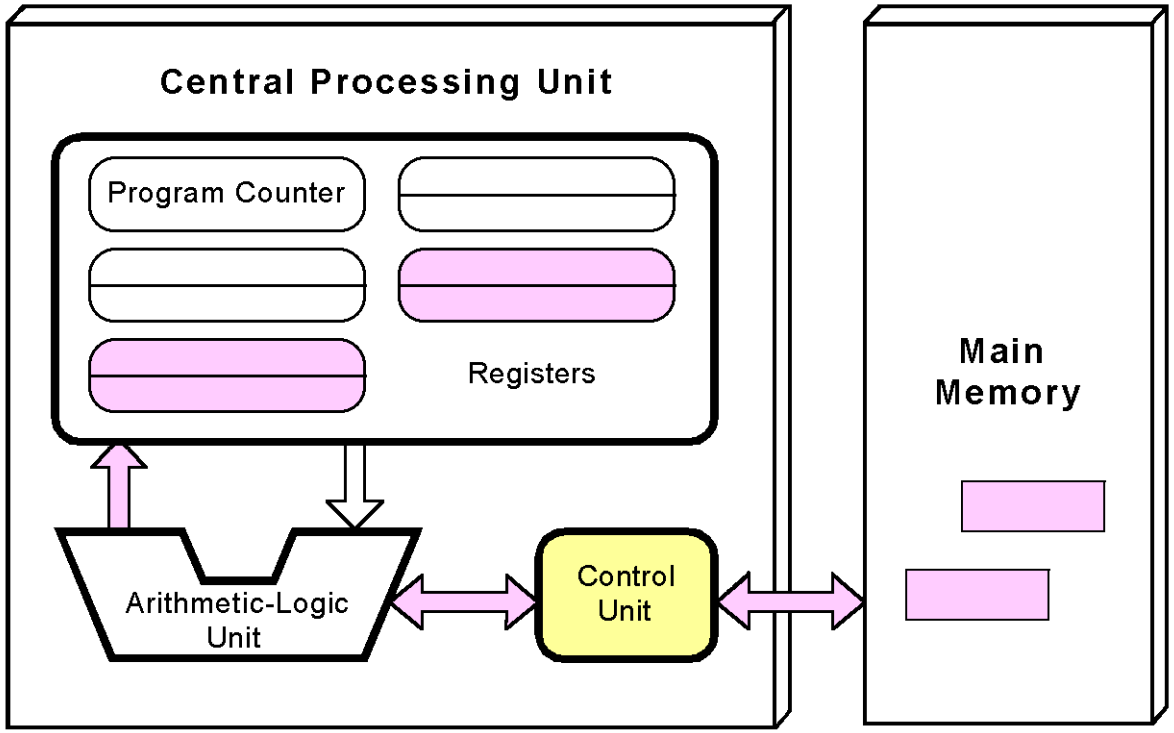
- **The control unit fetches the next instruction from memory using the program counter to determine where the instruction is located.**



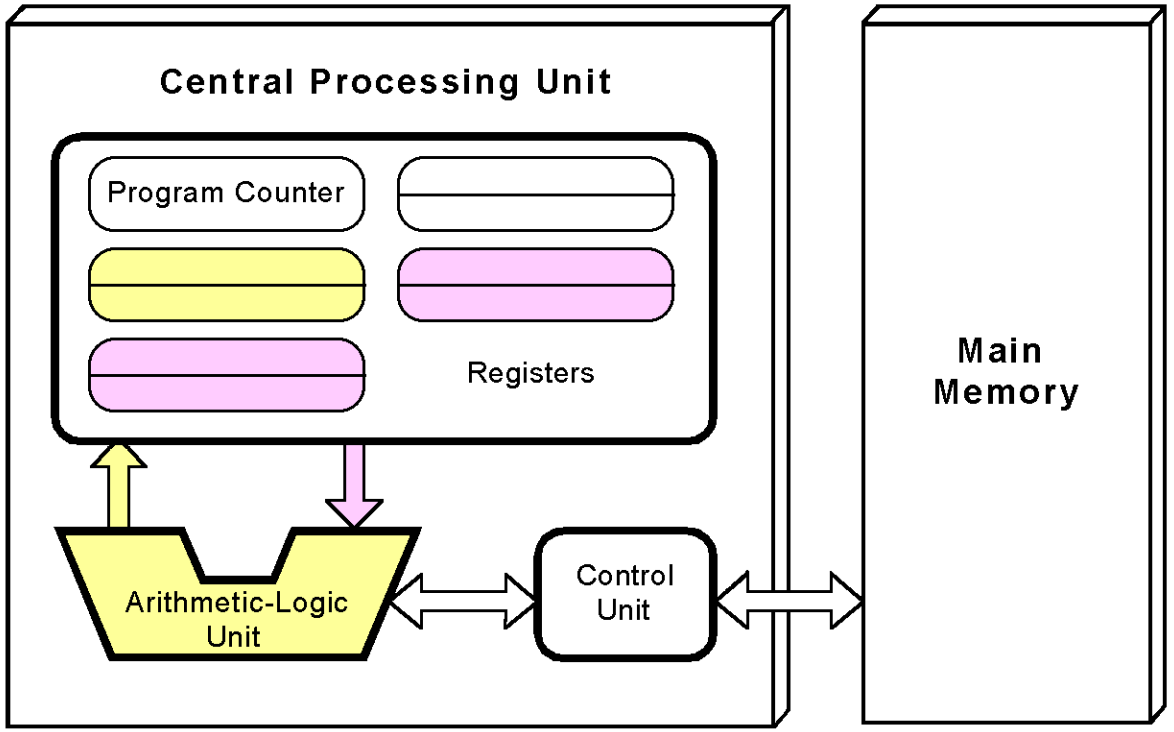
- **The instruction is decoded into a language that the ALU can understand.**



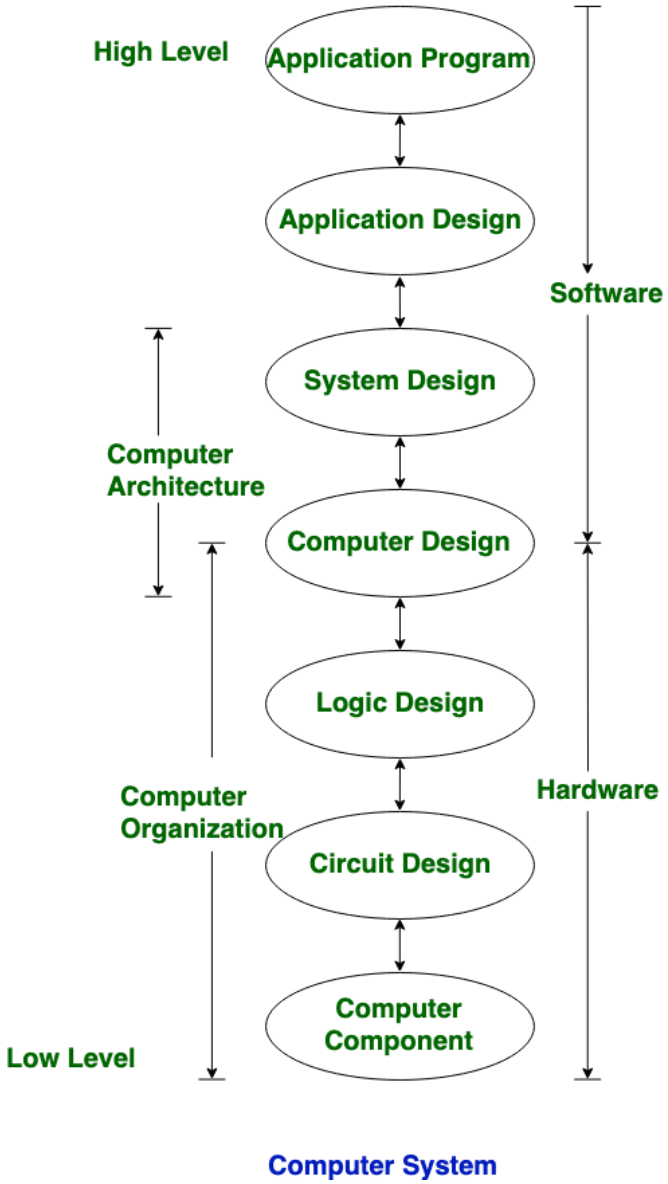
- **Any data operands required to execute the instruction are fetched from memory and placed into registers within the CPU.**



- **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.



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

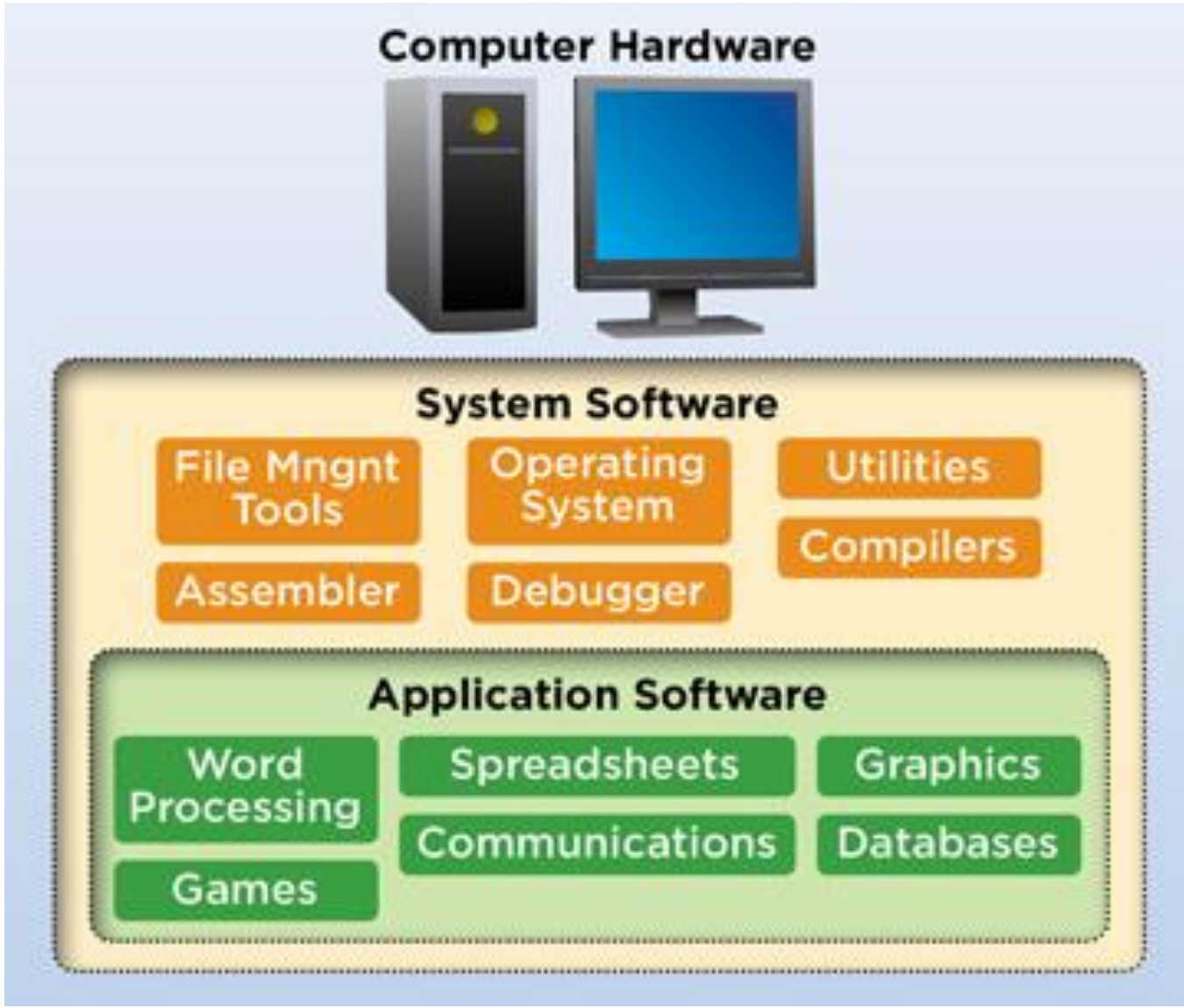
- Term OS covers many roles
  - Because of many designs and uses of OSes
  - Present in toasters through ships, spacecraft, game machines, TVs and industrial control systems
  - Born when fixed use computers for **military** became more general purpose and needed resource management and program control

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

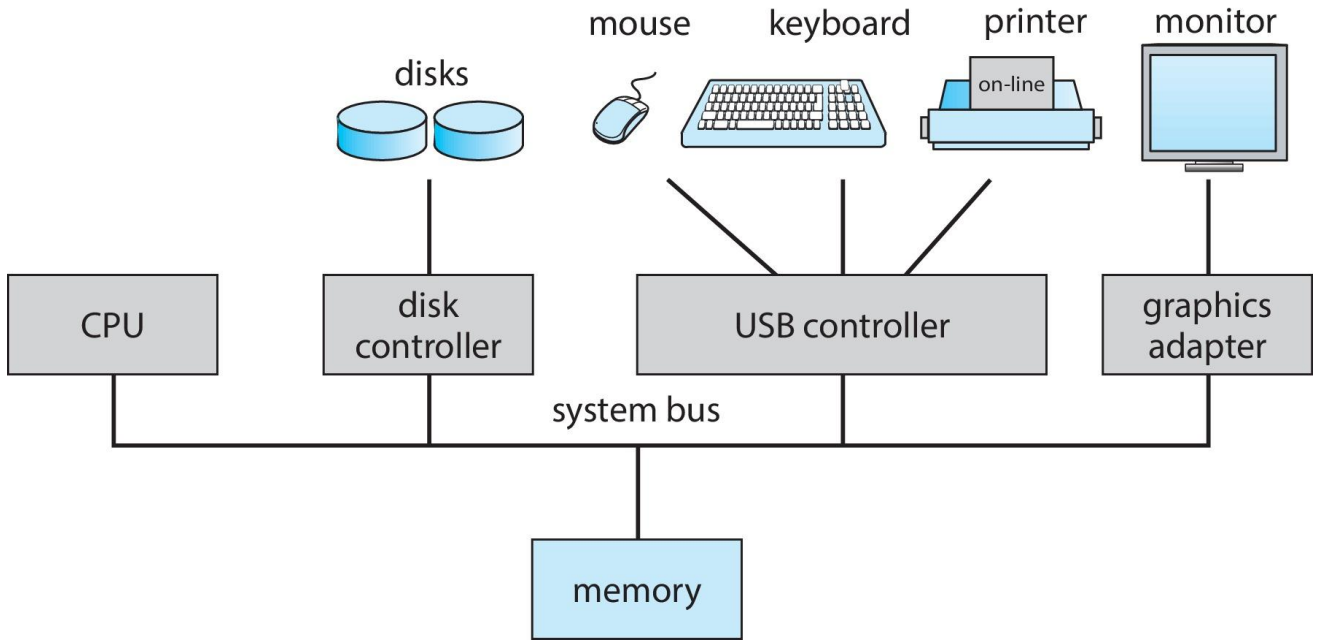
Operating System Definition (Cont.)

- 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.

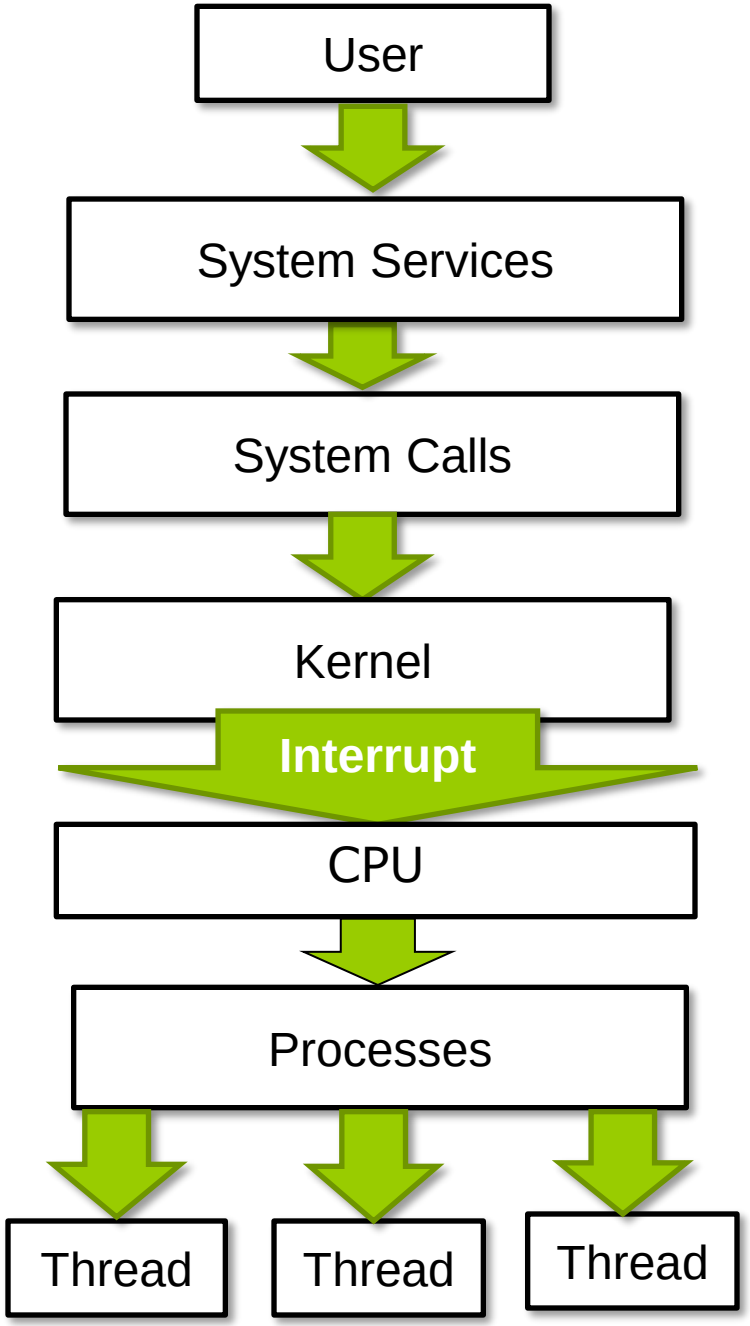
Polling:

- 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



Interrupt Timeline