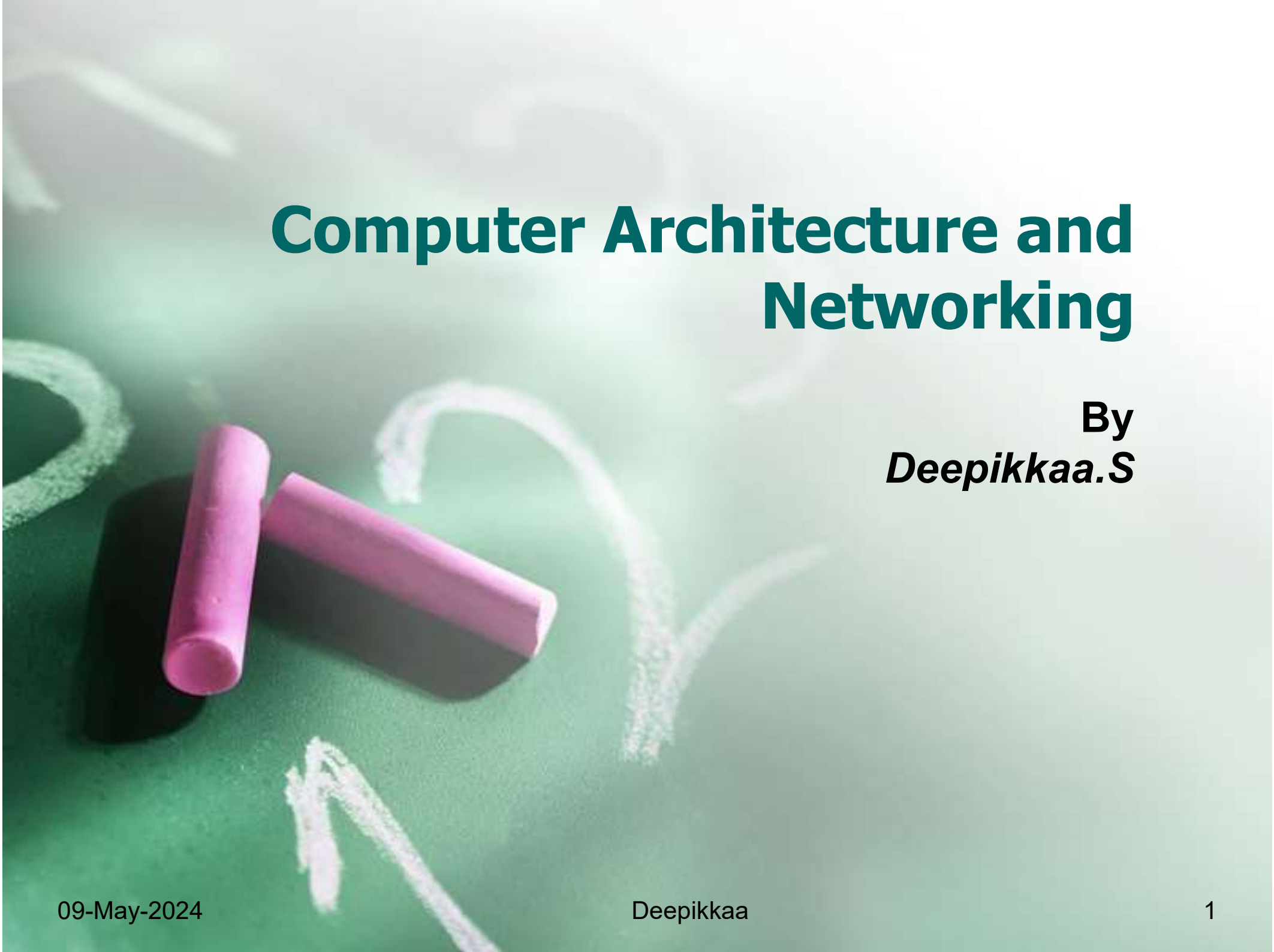


Computer Architecture and Networking

By
Deepikkaa.S

Computer architecture

- **Computer architecture is a specification describing how computer software and hardware connect and interact to create a computer network.**
- It determines the structure and function of computers and the technologies
- It is compatible with – from the central processing unit (CPU) to memory, input/output devices, and storage units

Computer architecture

- Computer scientists must build a computer with the same principles in mind as building the foundations of physical structure. The **three main pillars** they must consider are:

- **System design** This is what makes up the structure of a computer, including all hardware parts, such as CPU, data processors, multiprocessors, memory controllers, and direct memory access.

- **Instruction set architecture (ISA) –**
  - The set of all machine code instructions that a microprocessor (CPU) can execute.
  - A.K.A The language that a CPU speaks
  - This is any software that makes a computer run, including the CPU's functions and capabilities, programming languages, data formats, processor register types, and instructions used by programmers.

- **Microarchitecture** This defines the data processing and storage element or data paths. These include storage devices and related computer organisation tools.

Types of computer architecture

- **There are two main types of computer architecture:**

1) **Von Neumann architecture**
   - this features a *single memory space* for both data and instructions, which are fetched and executed ***sequentially***.

2) **Harvard architecture**
   - uses *separate memory spaces* for data and instructions, allowing for ***parallel*** fetching and execution.

Types of computer architecture

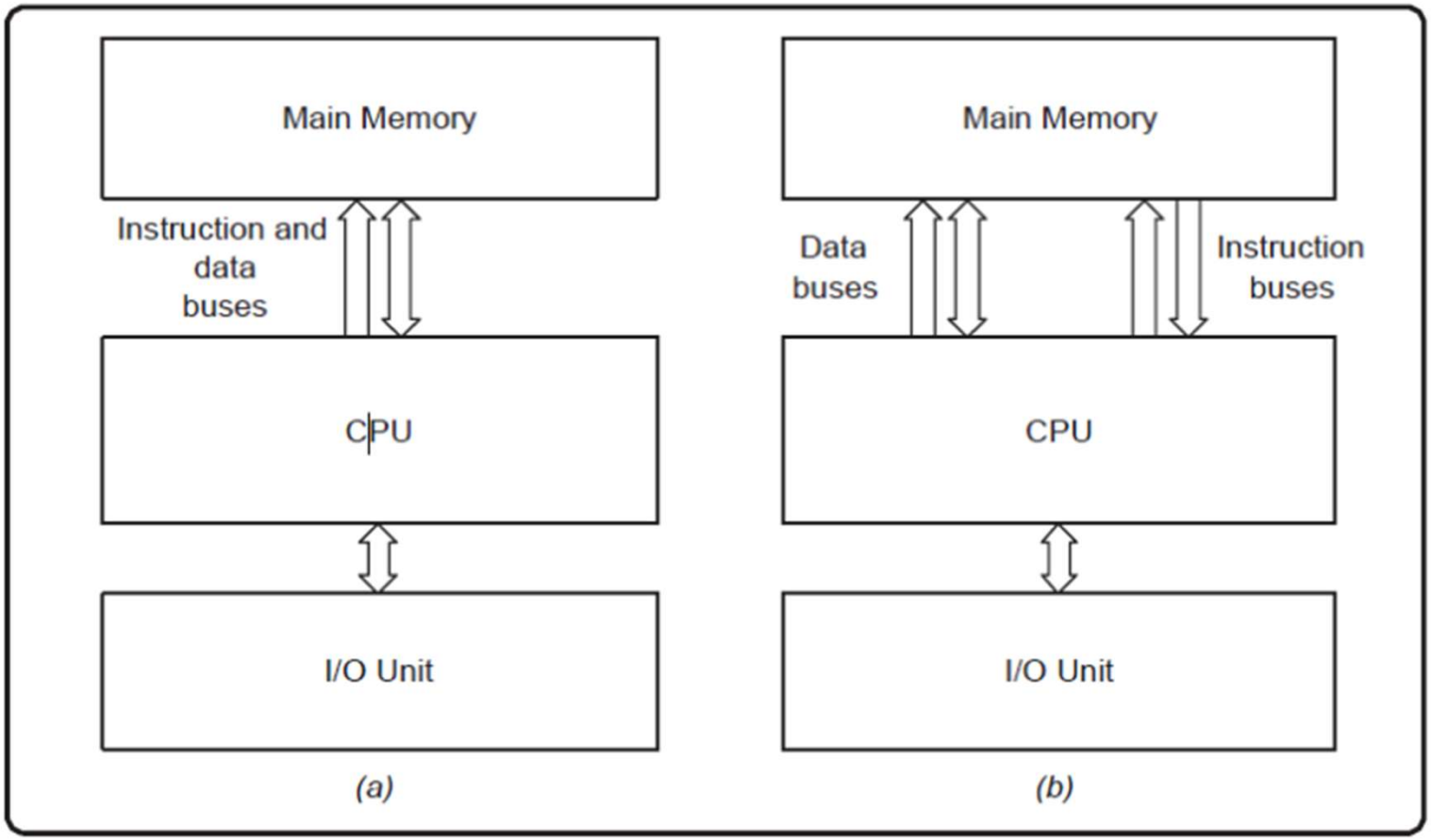


Diagram showing Von Neumann architecture (a) and Harvard architecture (b) *Source: ResearchGate*

Both types of architecture have their own advantages and trade-offs, and modern computers often use a combination of both to get the best system performance.

Application in Operating Systems:

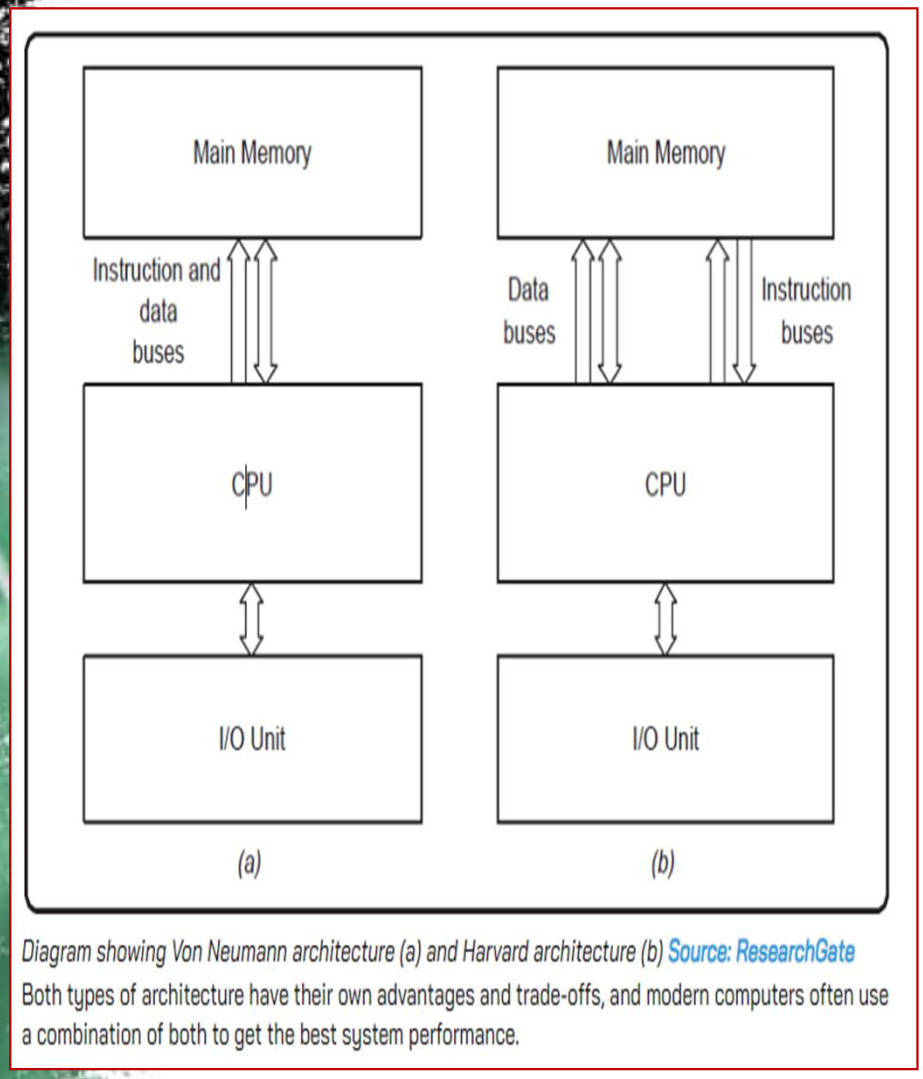


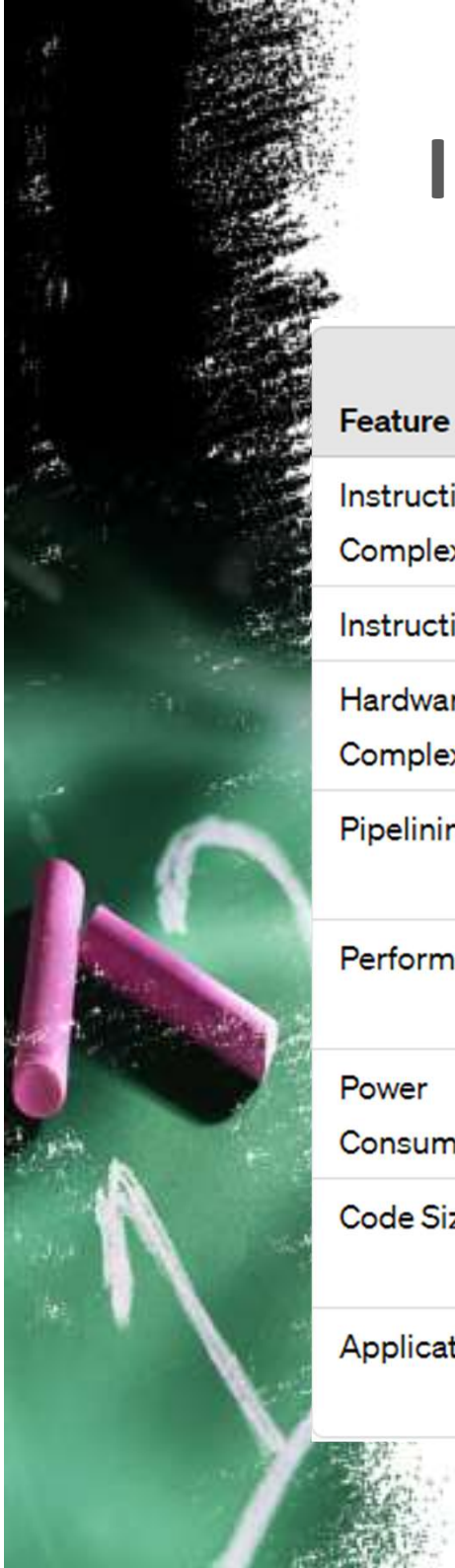
Diagram showing Von Neumann architecture (a) and Harvard architecture (b) Source: ResearchGate

Both types of architecture have their own advantages and trade-offs, and modern computers often use a combination of both to get the best system performance.

- **Von Neumann:** Commonly **used in generalpurpose computing devices like desktops, laptops, and servers**. Most modern operating systems, including Windows and Linux, are designed around Von Neumann architecture.
- **Harvard:** Often **utilized in embedded systems, microcontrollers, and specialized devices** where performance optimization and parallelism are critical. Some operating systems tailored for embedded systems may utilize Harvard architecture.

Instruction set architecture

- ***CISC***
  - **Very Complex**
  - **Complex operations can be performed using very little instructions or even just one instruction**
  - **To do this, we need a Complex CPU DESIGN, which means we need more transistors and more power**
  - **Used in Desktop, laptops, and larger systems**
  - **A CISC INSTRUCTION SET IS DEVELOPED BY INTEL.**
  - **Now INTEL with AMD are the only companies who can legally create x86 CPUs**
- ***RISC***
  - **Simplified, fewer possible instructions**
  - **Need fewer transistors, uses less power**
  - **DisAdv: Complex operation requires more instructions**
  - **Used in Mobile Devices for power efficient.**
  - **A RISC instruction set is developed by ACORN Computer and named as ARM**
  - **RISC is made popular after year 2000 when mobile devices are used more.**
  - **A company named ARM HOLDINGS sells rights/licenses to other companies.**



Instruction set architecture

Feature	CISC (Complex Instruction Set Computing)	RISC (Reduced Instruction Set Computing)
Instruction Complexity	Complex instructions performing multiple tasks.	Simple instructions focusing on basic operations.
Instruction Format	Variable-length instruction encoding.	Fixed-length instruction encoding.
Hardware Complexity	More complex hardware design.	Simpler hardware design.
Pipelining Support	Difficulties due to variable-length instructions.	Facilitated by uniform instruction format.
Performance	Potentially lower performance per clock cycle.	Generally higher performance per clock cycle.
Power Consumption	Typically higher power consumption.	Lower power consumption.
Code Size	Potentially smaller code size due to complex instructions.	Potentially larger code size due to simpler instructions.
Application	Historically dominant in general-purpose computing.	Prevalent in embedded systems, mobile devices, and specialized computing.

Central Processing Unit (CPU)

- CPU – Circuitry that controls the manipulation of data
- CPU = ALU + CU + Registers where
- ALU – Arithmetic and Logic Unit A circuit that performs operations on data
- CU – Control unit – The circuit that coordinates the activities of the CPU.
- Registers Quick, small stores of data within the CPU.

OS Resources

1.CPU (Central Processing Unit)

2. Memory (RAM)

3. Storage Devices

4. Input/Output Devices

5. Network Interfaces

6. System Resources

- System configuration settings, environment variables, and system files.
- Managed by the operating system for systemwide settings and configurations.

OS Services

1. Process Management
2. Memory Management
3. File System Management
4. Device Management
5. Network Management
6. User Interface
7. Security Management
8. System Monitoring and Performance Management

How a CPU executes instructions

- ✓ The CPU is the central component responsible for executing programs in a computer.
- ✓ Programs consist of unambiguous instructions meant to be followed mechanically by the CPU.
- ✓ Programs are written in machine language, a simple language directly understood by the computer.
- ✓ Machine language instructions are stored in main memory (RAM) along with data.
- ✓ Main memory consists of sequentially numbered locations, each with an address.

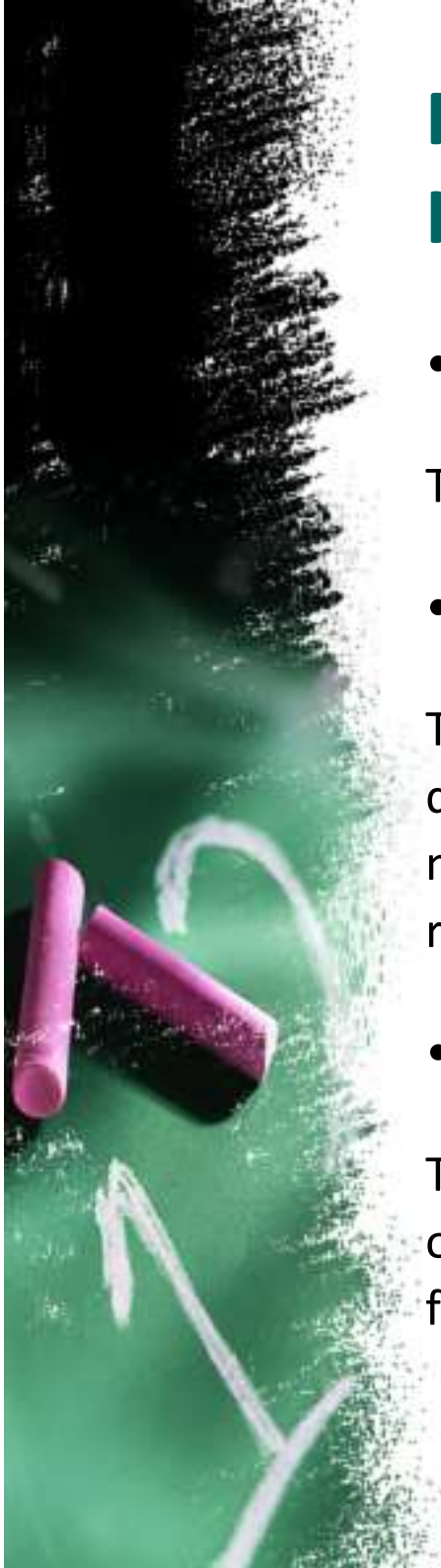
How a CPU executes instructions

- ✓ The CPU fetches instructions from memory by sending their addresses and executes them.
- ✓ This process of fetching and executing instructions is called the fetch and execute cycle.
- ✓ The CPU contains internal registers, including the program counter (PC), which keeps track of the current instruction being executed.
- ✓ Machine language instructions are binary numbers, composed of zeros and ones.
- ✓ Switches called transistors represent binary numbers, and machine language instructions are patterns of switches turned on or off.

How a CPU executes instructions

- ✓ The CPU executes instructions mechanically based on the patterns of switches, without understanding them.
- ✓ Therefore, programs must be perfect, complete, and unambiguous because the CPU executes them exactly as written.

- The CPU fetches and executes machine language instructions stored in memory, following a mechanical process without understanding the instructions' meaning.
- Animation reference – Fetch and Execute Cycle
- https://www.youtube.com/watch?v=yT-ntiGo8pc



Fetch- Decode – Execute Cycle a.k.a Instruction Cycle

- **Fetch**

The next instruction is retrived by the CPU from main memory

- **Decode**

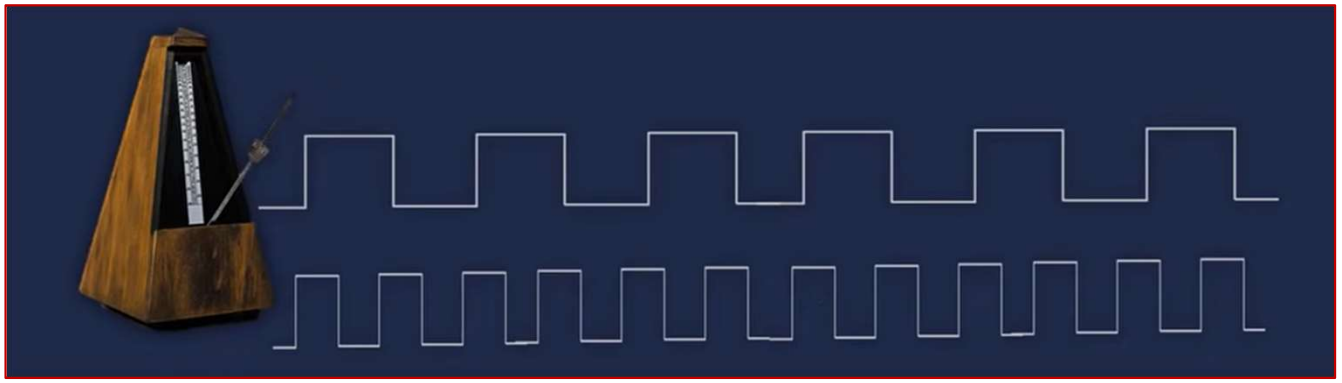
The instruction is broken down to its individual components to determine what the instruction is , and what data is being used. Decode means operator + operand. Eg: ADD R1 R5 (add contents from the registers R1 and R5)

- **Execute**

The Control Unit(CU) activates the necessary circuity / data transfers. The output of this stage is stored in a register, and data may be read / written from / to the main memory during this stage.

Clock Ticks

- Computers have a system clock that provides timing signals to synchronize circuits
- CPUs are designed to operate at a specific frequency – and the system clock is raised to this rate by the processer, giving the clock speed(Hz).
- The CPU needs a certain amount of clock ticks/ cycles per instruction



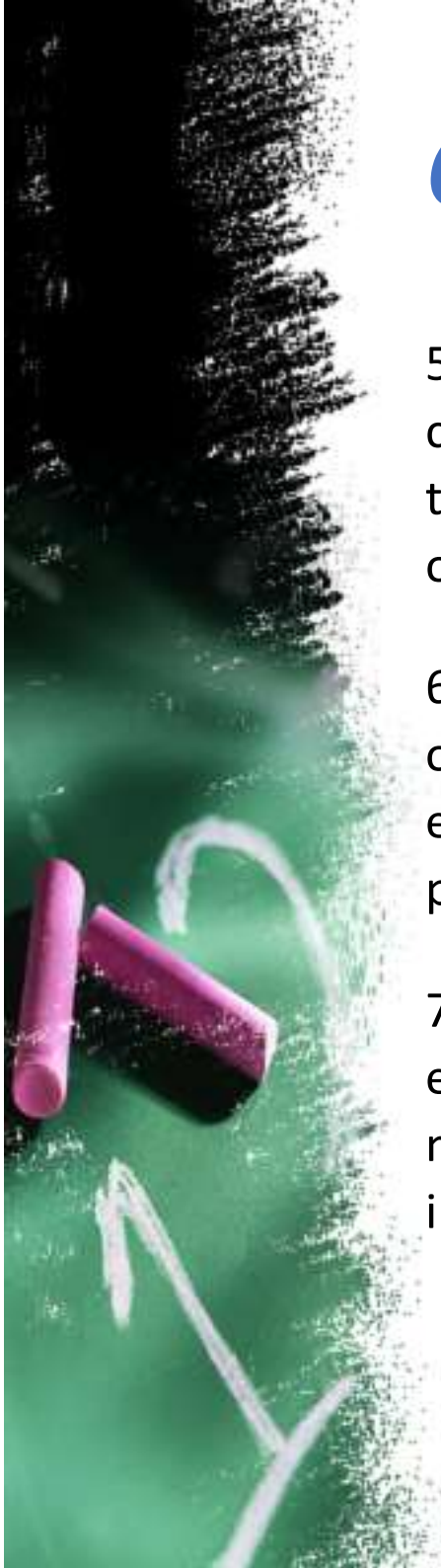
CPU performance factors

1. Clock Speed: The clock speed, measured in GHz (gigahertz), determines how many cycles per second the CPU can execute instructions. Higher clock speeds generally result in faster processing.

2. Instruction Set Architecture (ISA): The instruction set architecture defines the set of instructions that the CPU can execute. Efficient ISA design can improve performance by optimizing instruction execution.

3. Cores: Multi-core processors contain multiple processing units (cores) on a single chip. Each core can execute instructions independently, allowing for parallel processing and improved performance, especially in multi-threaded applications.

4. Cache Size and Hierarchy: CPU caches, including L1, L2, and L3 caches, store frequently accessed data and instructions closer to the CPU cores. Larger cache sizes and optimized cache hierarchies can reduce memory access latency and improve performance.

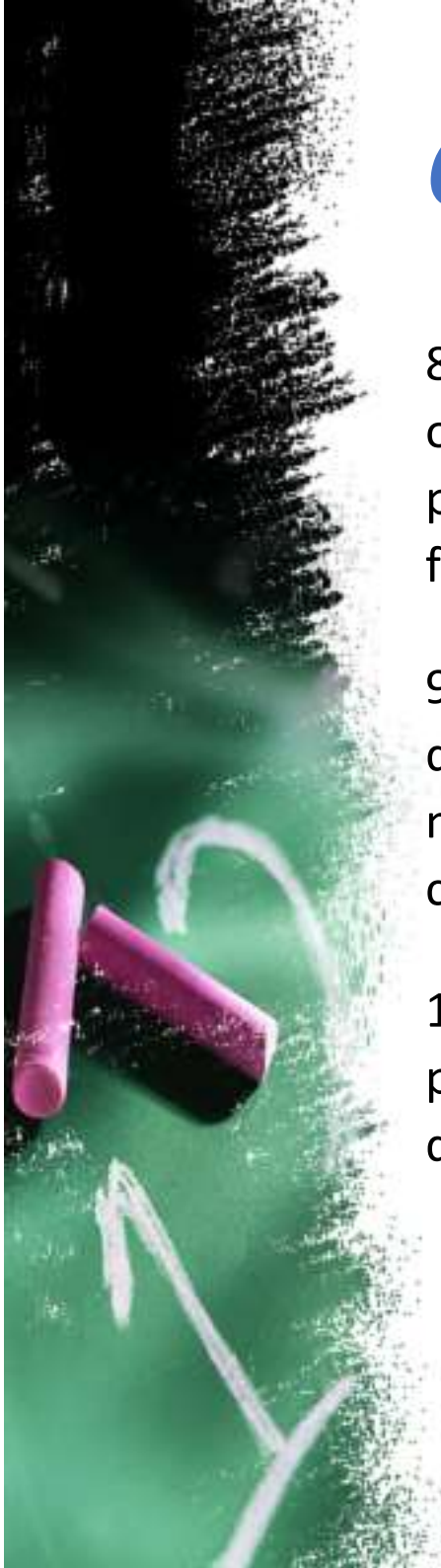


CPU performance factors

5. Memory Bandwidth: The CPU's ability to access data from memory is crucial for performance. Higher memory bandwidth allows for faster data transfer between the CPU and memory, reducing latency and improving overall system performance.

6. Pipeline Depth: Modern CPUs use pipelining to overlap the execution of multiple instructions. Deeper pipelines allow for more stages of execution, potentially increasing throughput but also introducing risks of pipeline stalls and inefficiencies.

7. Instruction-Level Parallelism (ILP): ILP techniques, such as superscalar execution and out-of-order execution, exploit opportunities to execute multiple instructions simultaneously, improving performance by increasing instruction throughput.



CPU performance factors

8. Branch Prediction: Branch prediction mechanisms anticipate the outcome of conditional branches in program execution, reducing the performance impact of branch mispredictions and improving instruction flow.

9. Power Efficiency: Efficient power management techniques, such as dynamic voltage and frequency scaling (DVFS) and idle state management, help optimize CPU performance while minimizing power consumption.

10. Instruction and Data Pipelines: The efficiency of instruction and data pipelines within the CPU affects how quickly instructions can be fetched, decoded, and executed, impacting overall processing speed.

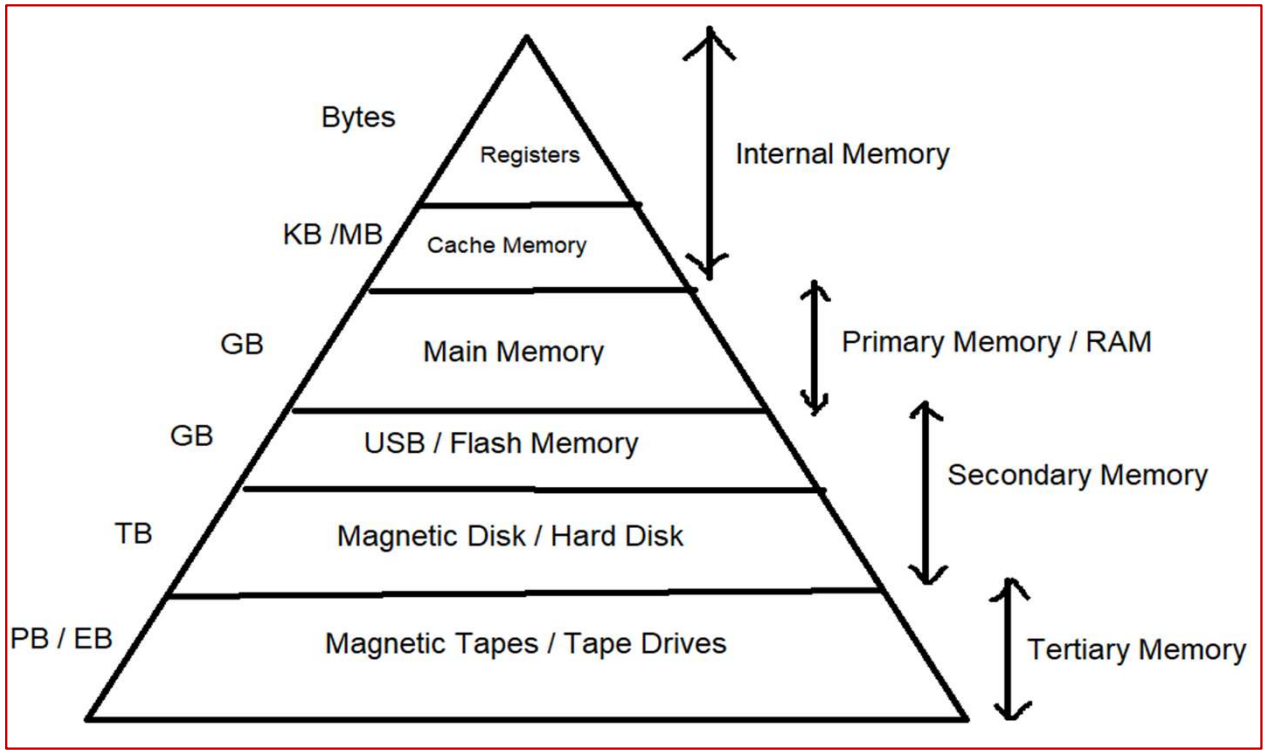
CPU performance factors

11. Memory Level Parallelism (MLP): MLP techniques, such as simultaneous multithreading (SMT) and memory prefetching, exploit parallelism at the memory level to improve data access latency and throughput.

12. Vectorization and SIMD (Single Instruction, Multiple Data): CPUs with support for SIMD instructions can perform operations on multiple data elements simultaneously, enhancing performance for tasks with data-level parallelism.

Memory

- *Types of Memory and Hierarchy*
- The memory hierarchy is a system that organizes computer memory into multiple levels, each with different characteristics in terms of speed, size, and cost. This hierarchy aims to optimize memory access and performance by providing a balance between speed and capacity.



Main Memory (RAM)

Characteristics:

- *Larger capacity compared to cache memory (several gigabytes to terabytes).*
- *Slower access time compared to cache memory (measured in nanoseconds to microseconds).*
- *Volatile memory, requiring power to maintain data integrity.*

Purpose:

- *Stores program instructions and data required by the CPU during program execution.*
- *Acts as a bridge between high-speed cache memory and slower secondary storage devices.*

Secondary Storage

Characteristics:

- *Large capacity (terabytes to petabytes).*
- *Much slower access time compared to main memory (milliseconds to seconds).*
- *Non-volatile memory, retaining data even when power is turned off.*

- **Purpose:**
  - *Stores data and programs for long-term storage.*
  - *Includes storage devices such as hard disk drives (HDDs), solid-state drives (SSDs), and magnetic tape.*

Cache Memory

- L1 Cache
- L2 Cache
- L3 Cache

L1 Cache

- - Characteristics:
  - *- Located closest to the CPU, typically integrated directly into the CPU core.*
  - *- Very small in size (a few kilobytes).*
  - *- Extremely fast access time, with latency measured in nanoseconds.*
- - Purpose:
  - *- Stores frequently accessed data and instructions to minimize access time.*
  - *- Acts as a buffer between the CPU and slower levels of memory.*

L2 Cache

- - Characteristics:
  - *- Larger than L1 cache but still smaller than main memory.*
  - *- Located on the CPU chip or as a separate module.*
  - *- Slightly slower access time compared to L1 cache.*
- - Purpose:
  - *- Provides additional cache space for storing frequently accessed data and instructions.*
  - *- Helps reduce memory latency by holding a larger working set of data closer to the CPU.*

L3 Cache

- - Characteristics:
  - *- Larger and slower than L2 cache.*
  - *- Typically shared among multiple CPU cores in multi-core processors.*
  - *- Can be located on the CPU chip or as a separate module shared among multiple CPU cores.*
- - Purpose:
  - *- Serves as a shared cache for multiple CPU cores, facilitating data sharing and coherence.*
  - *- Helps improve overall system performance by reducing memory access latency for shared data.*

RAM - ROM

READ-ONLY MEMORY

CONTAINS PREPROGRAMMED CODE

RANDOM ACCESS MEMORY

CONTAINS TEMPORARY DATA

ROM – RAM – Cache - HDD

Feature	Memory)	Memory)	Cache Memory	(HDD)
Volatility	Non-volatile	Volatile	Volatile	Non-volatile
Purpose	Stores firmware, configuration data, and permanent data	Stores program instructions and data temporarily during program execution	Stores frequently accessed data and instructions for quick access by the CPU	Stores data and programs for long-term storage
Access Time	Typically slower access compared to RAM	Faster access compared to ROM, but slower than cache	Much faster access than RAM, but slower than cache	Slower access compared to RAM, cache, and SSDs
Speed	Slow	Fast	Very fast	Relatively slow
Size	Smaller	Larger	Smaller	Larger
Type of Storage	Permanent or semi-permanent	Temporary	Temporary	Permanent
Location	Typically on the motherboard or embedded in devices	On the motherboard or in memory modules	On or near the CPU (L1, L2, L3 caches)	Installed externally in the computer case
Examples	BIOS, firmware, microcode	DDR4, DDR5, SRAM, DRAM	L1, L2, L3 cache	Magnetic hard disk drives (HDDs)

Memory management concepts

- Memory Partitioning
- Memory Allocation
- Memory Paging and Segmentation
- Virtual Memory
- Memory Protection and Sharing
- Garbage Collection

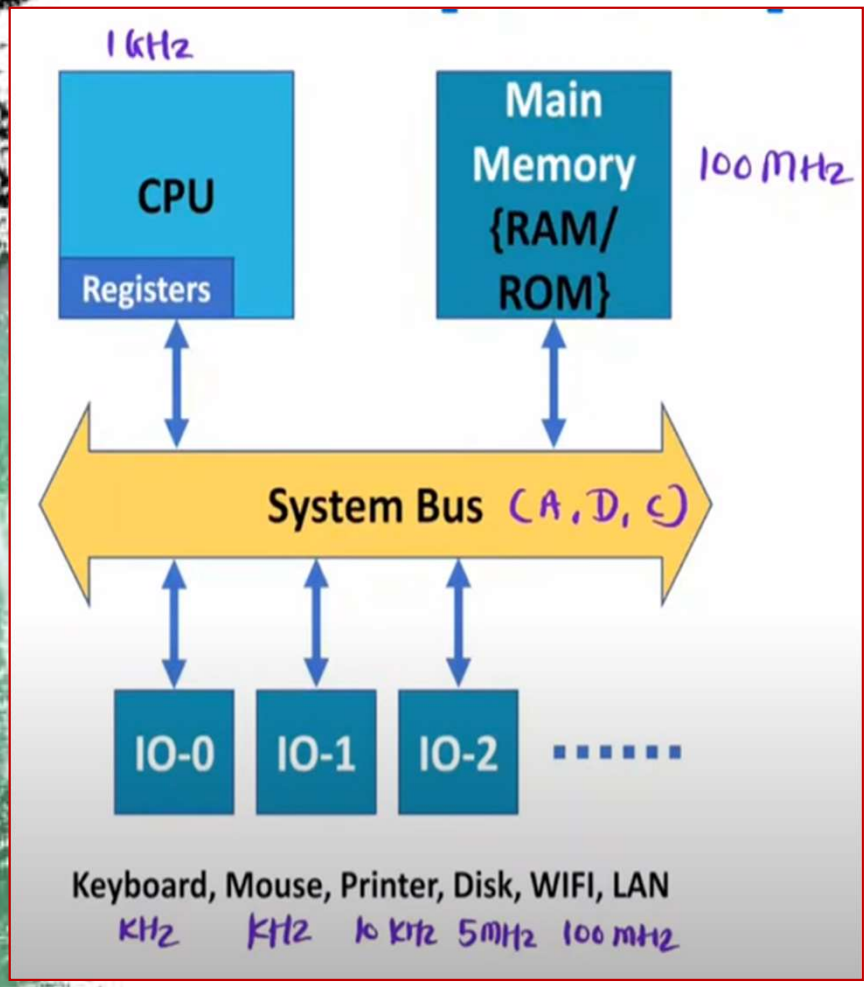
Keypoints - Memory

- *The memory hierarchy provides a balance between speed, capacity, and cost by organizing memory into multiple levels.*
- *Cache memory (L1, L2, L3) is designed for fast access by storing frequently accessed data and instructions.*
- *Main memory (RAM) acts as a bridge between cache memory and slower secondary storage devices.*
- *Secondary storage devices provide large-capacity storage for long-term data storage but have slower access times compared to main memory.*

Input and Output System

- ✓ I/O systems facilitate communication between the CPU, memory, and various I/O devices, enabling data transfer and device control.
- ✓ Communication between these components can occur via memory-mapped I/O, I/O ports, programmed I/O, interrupt-driven I/O, or direct memory access (DMA).
- ✓ Buses serve as communication pathways, allowing data transfer between the CPU, memory, and I/O devices, with data transfer occurring in parallel or serial modes.

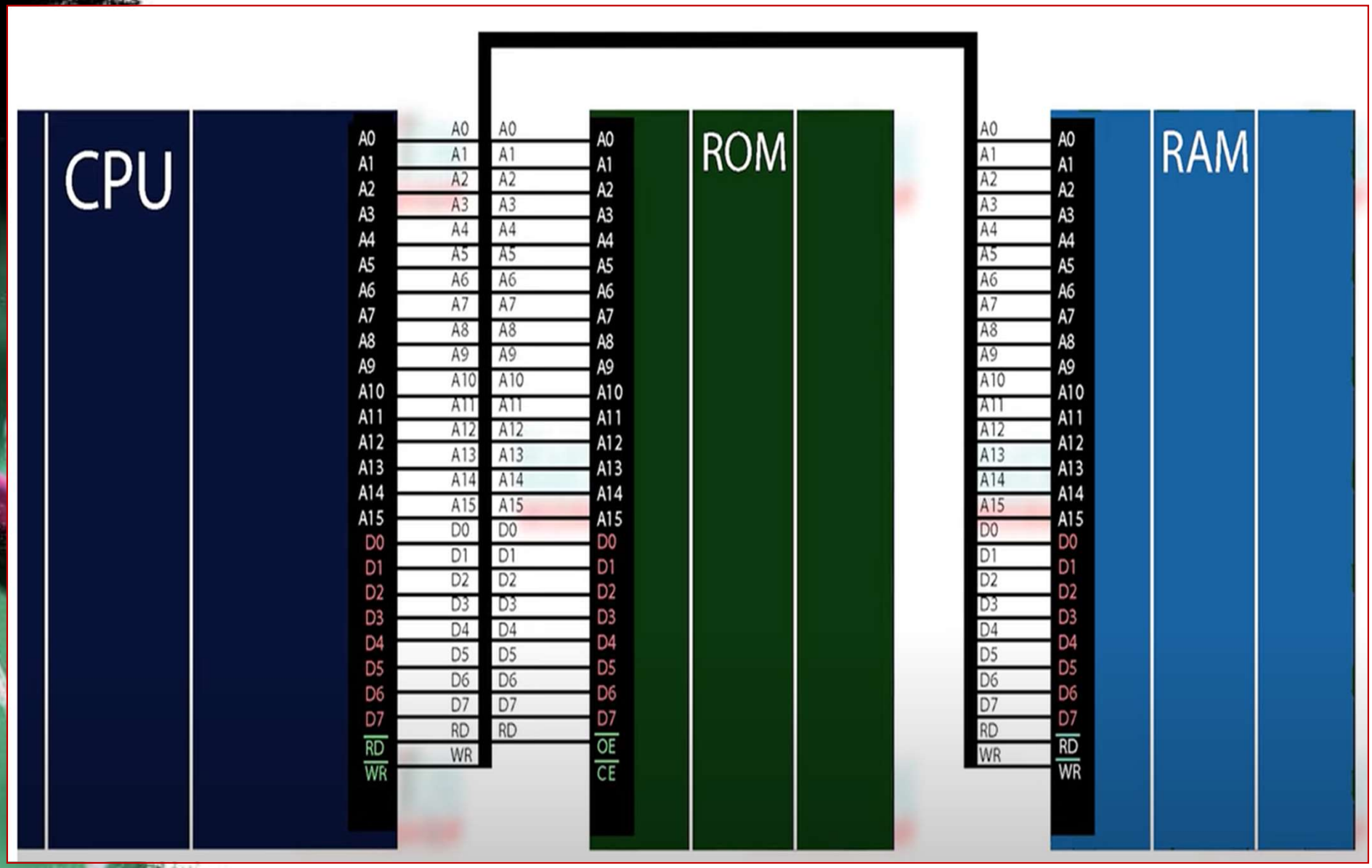
Architecture



System Bus is a combination of Address Bus, Data Bus and Control Bus.

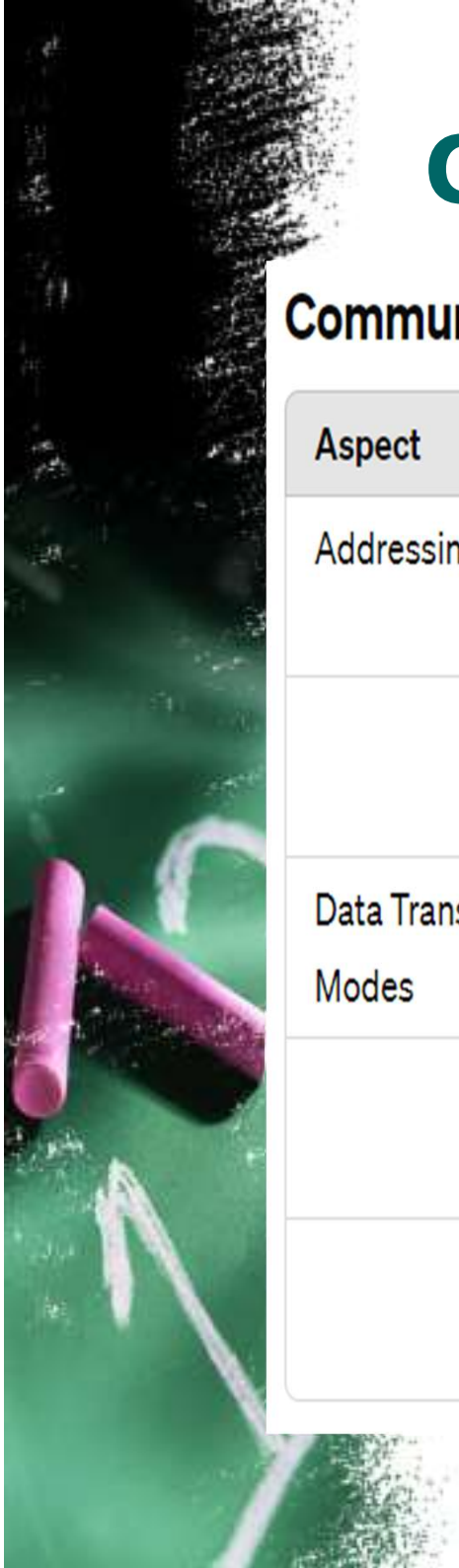
The frequency in which the CPU function is in GHz. Memory works in MHz. But the I/O works in KHz which is very slower .

Address Bus, Data Bus, Control Bus



Address Bus, Data Bus, Control Bus

Aspect	Address Bus	Data Bus	Control Bus
Function	Transmits memory addresses from CPU to memory and I/O devices.	Transmits data between CPU, memory, and I/O devices.	Carries control signals between CPU, memory, and I/O devices.
Direction	Uni-directional (CPU to memory and I/O devices).	Bi-directional (CPU <-> Memory <-> I/O devices).	Uni-directional (CPU to memory and I/O devices).
Size	Determines the maximum memory capacity the CPU can address.	Determines the number of bits transferred in parallel.	Not determined by size but by specific control signals.
Width	Width depends on the CPU architecture and memory addressing capabilities.	Width depends on the CPU architecture and memory bus design.	Width depends on the specific control signals required.
Examples	Sending a memory address to read/write data from/to a specific location.	Transferring binary data, instructions, or metadata between components.	Signaling read/write commands, interrupt requests, and timing signals.



Communication

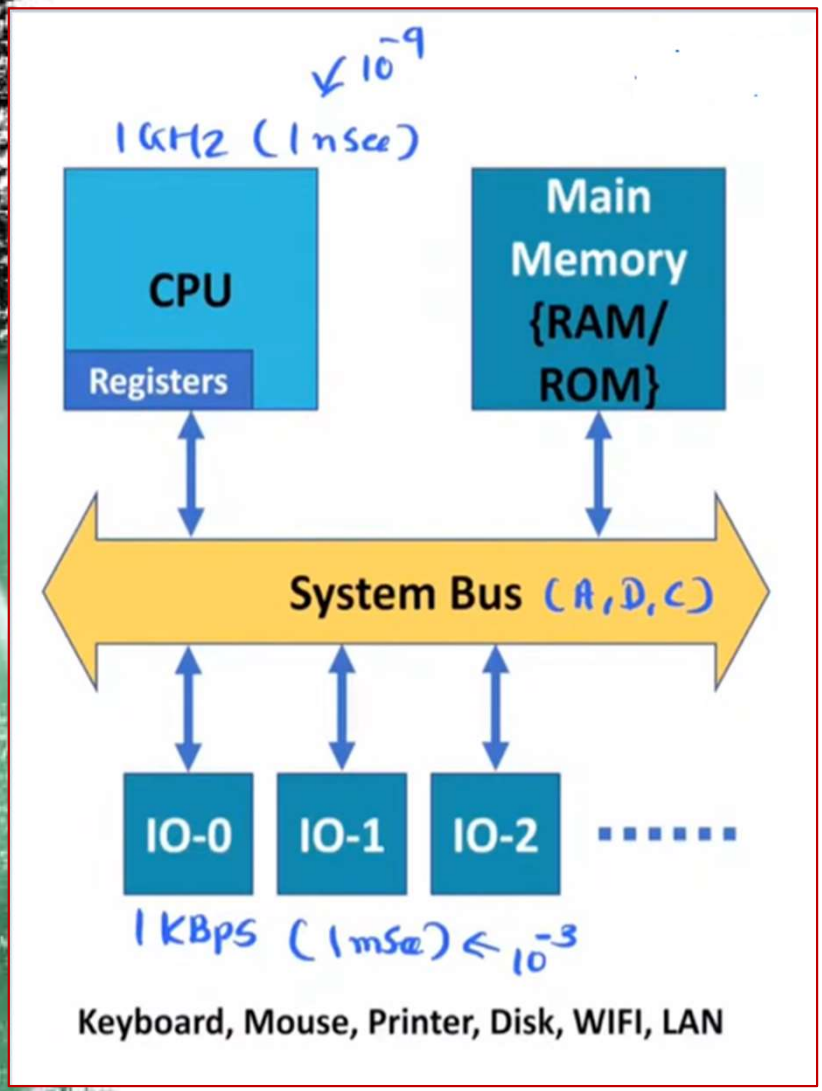
Communication between CPU, Memory, and I/O Devices:

Aspect	Description
Addressing	Memory-mapped I/O: Treats I/O devices as memory locations, allowing direct access via memory addresses.
	I/O Ports: Uses dedicated I/O ports for device communication, separate from memory addresses.
Data Transfer Modes	- Programmed I/O: CPU manually controls data transfer between memory and devices via instructions.
	- Interrupt-driven I/O: Devices trigger interrupts to notify the CPU when data is ready for transfer or when an operation is completed.
	- Direct Memory Access (DMA): Allows devices to transfer data directly to/from memory without CPU intervention, reducing CPU overhead.

IO Transfer Mode

1) Programmed I/O
2) Interrupt Driven IO
3) DMA

Programmed IO

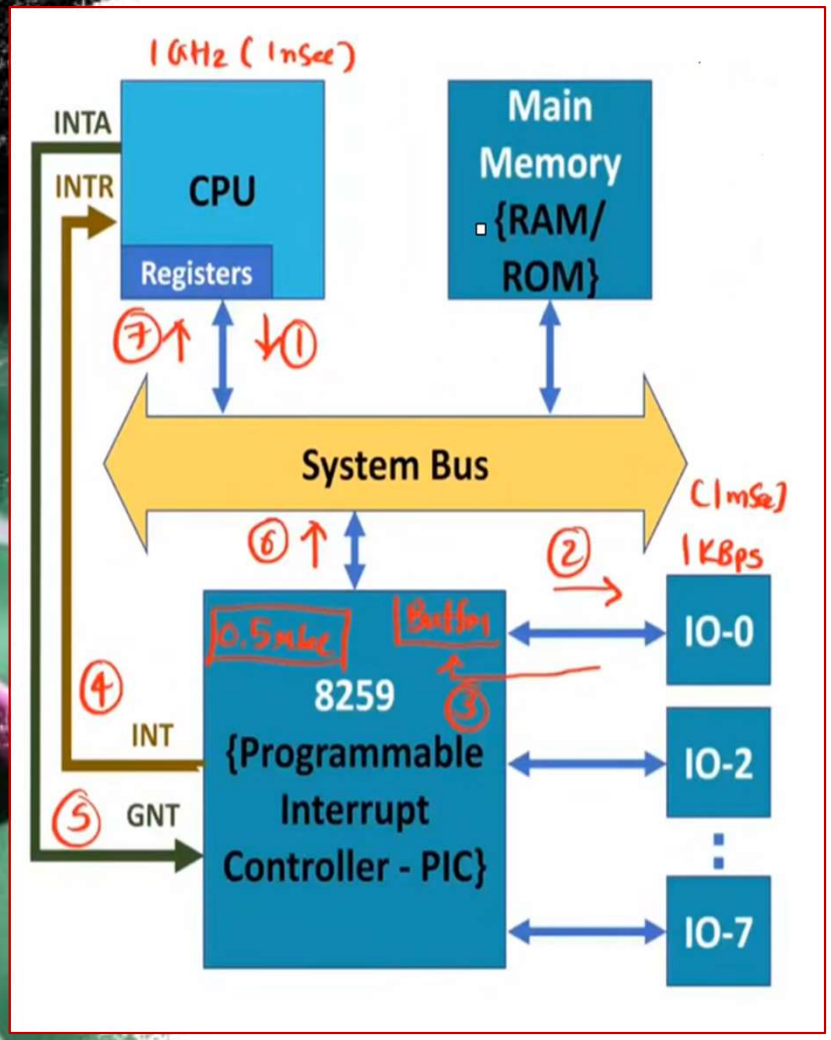


- IO Device and CPU directly interacts with each other.
- CPU works very fast but IO works very slow.
- **DisAdv**:
  - *CPU has to wait for a long time. So the Performance of the CPU will come down.*

Types of Programmed IO

1) Memory-mapped IO
2) IO mapped IO

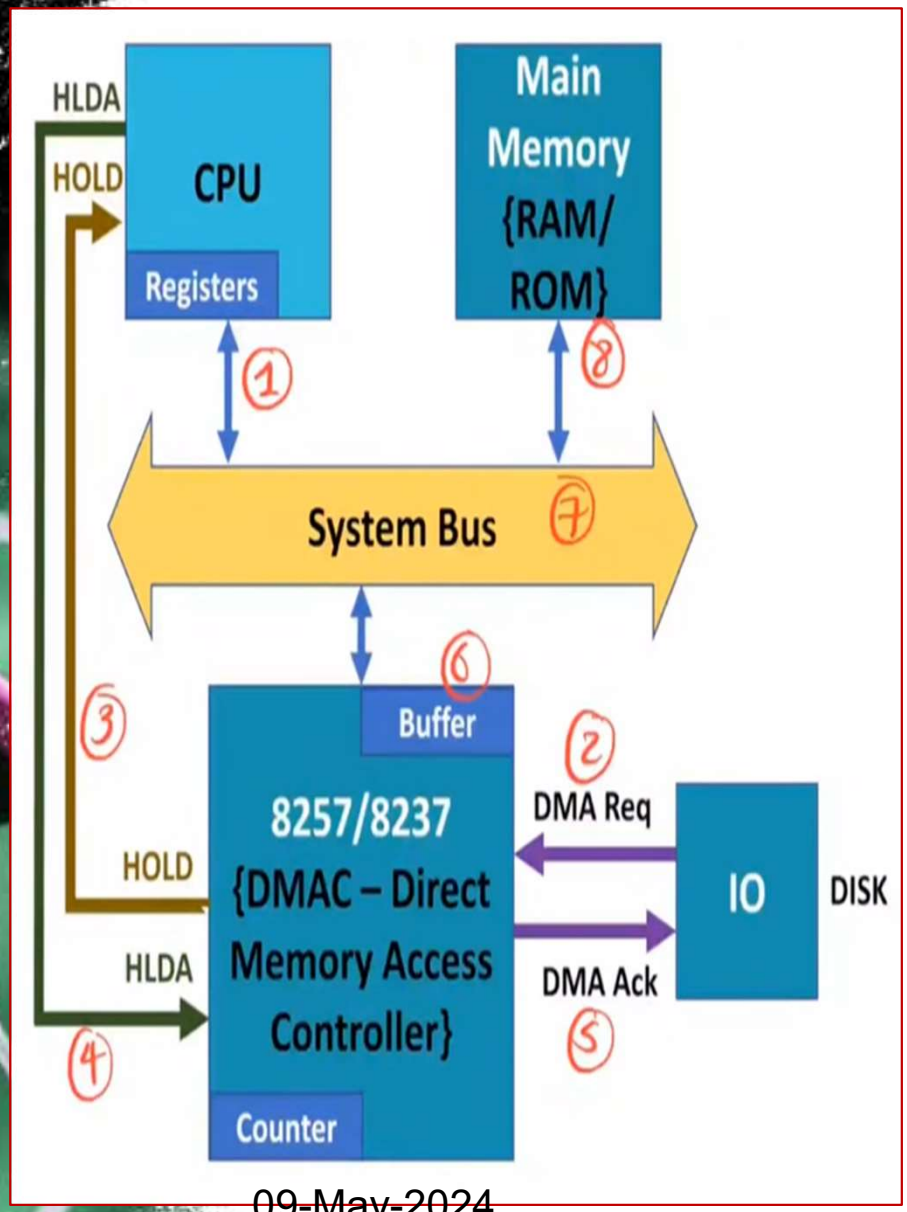
Interrupt Driven IO



1. CPU request the data as IO Command to System bus via PIC to IO devices
2. PIC will request data from IO
3. IO sends data to PIC and it will get stored in Buffer within PIC.
4. Once the buffer is full, PIC will send an interrupt signal to the CPU
5. If the CPU is free, it will send Interrupt Acknowledgement to PIC.
6. PIC will send the data in the buffer to System bus
7. System bus will transfer the data to Registers in the CPU. The CPU Continue will other jobs till the buffer is full. So Less BLOCK Time

- **DisAdv** : For data transfer between IO and memory, CPU stays busy.

Direct Memory Access



1. CPU requests the data as IO Command to the System bus. IO Command has
2. {Port number, Memory Address, Count, Control Signal}
3. DMAC will wait for the DMA request from the Disk. The disk is ready to transfer the data.
4. DMAC will give a hold signal to the CPU because it needs control over the system bus to transfer the data to the main memory. The system bus is now under use by the CPU.
5. CPU gives HOLD ACK to the DMAC
6. After receiving the HLDA from the CPU, it gives DMA ACK to the Disk.
7. Now Disk transfers the data to the DMAC Buffer.
8. Now DMAC will transfer data to System Bus
9. Main Memory reads the data from the system bus.

Direct Memory Access

- IO Command has

➢ Port number in the disk through which data transfer is going to happen

➢ Memory Address where the data finally should get stored,

➢ Count will load in the counter. For eg, If we assume 1 word = 1byte and we need to transfer 32 bytes, the counter will be updated with 32. Once the disk transferred 1 byte of data to main memeory the counter will get decremented by 1 and it continues till the counter value reaches 0. Means data transfer is completed without the involvement of CPU.

➢ Control Signal will let us know which operation is performed now (READ or WRITE)

Memory Mapped IO and IO Mapped IO in 8086

Memory Mapped IO	IO Mapped IO
❖ Here IO devices treated as Memory	❖ Here IO devices treated as IO.
❖ 20 bits addressing ($A_0 - A_{19}$)	❖ 8 or 16 bits addressing ($A_0 - A_{15}$)
❖ It can address = 2^{20} = 1MB Address	❖ It can address = 2^{16} = 64K Address
❖ Number of devices can be = 2^{20}	❖ Number of devices can be = 2^{16}
❖ Decoding is more complex as total 20 Address lines are used for full decoding.	❖ Decoding is Less complex as total 16 Address lines are used for full decoding.
❖ Decoding is Expensive.	❖ Decoding is cheaper.
❖ Here we use $\overline{MEMR}$ and $\overline{MEMW}$ control signals.	❖ Here we use $\overline{IOR}$ and $\overline{IOW}$ control signals.
❖ IO can be accessed by any memory instructions.	❖ IO can be accessed ONLY by IN and OUT Instructions.
❖ Data transfer happens between any registers and IO.	❖ Data transfer only between AX [AH & AL] and IO.
❖ Works slower due to more gates and circuits.	❖ Works faster due to less delay.

Parallelism and Hardware Acceleration

- *Introduction to Parallel Computing:*
  - *- **Definition**: Parallel computing is a computing model where multiple tasks or parts of a task are executed simultaneously to increase computational speed and efficiency.*
  - *- **Purpose**: Parallel computing aims to solve complex problems faster by dividing them into smaller tasks that can be processed concurrently.*
  - *- **Techniques**: Parallel computing techniques include task parallelism, data parallelism, and pipeline parallelism.*

Multi-core Processors and GPUs

- **- Multi-core Processors:**
  - *- Definition: Multi-core processors contain multiple processing units (cores) on a single chip.*
  - *- Purpose: Enable parallel execution of tasks by distributing workload across multiple cores.*
  - *- Examples: Intel Core i7, AMD Ryzen processors.*
- **- Graphics Processing Units (GPUs):**
  - *- Definition: GPUs are specialized processors designed for parallel computation of graphical and non-graphical tasks.*
  - *- Purpose: Accelerate tasks such as rendering, image processing, machine learning, and scientific simulations.*
  - *- Examples: NVIDIA GeForce, AMD Radeon GPUs.*

Parallelism and Hardware Acceleration

- *Introduction to Parallel Computing:*
  - *- **Definition**: Parallel computing is a computing model where multiple tasks or parts of a task are executed simultaneously to increase computational speed and efficiency.*
  - *- **Purpose**: Parallel computing aims to solve complex problems faster by dividing them into smaller tasks that can be processed concurrently.*
  - *- **Techniques**: Parallel computing techniques include task parallelism, data parallelism, and pipeline parallelism.*

Parallelism and Hardware Acceleration

- *Introduction to Parallel Computing:*
  - *- **Definition**: Parallel computing is a computing model where multiple tasks or parts of a task are executed simultaneously to increase computational speed and efficiency.*
  - *- **Purpose**: Parallel computing aims to solve complex problems faster by dividing them into smaller tasks that can be processed concurrently.*
  - *- **Techniques**: Parallel computing techniques include task parallelism, data parallelism, and pipeline parallelism.*

Real-world Applications and Examples

- **Scientific Computing:** - Simulations of weather patterns, fluid dynamics, and molecular dynamics.
- **Data Analytics and Machine Learning:** - Training and inference of deep learning models, processing large datasets.
- **Image and Video Processing:** Real-time video encoding/decoding, image rendering, and computer vision tasks.
- **Financial Modeling:** Risk analysis, option pricing, algorithmic trading.
- **High-Performance Computing (HPC):** Computational fluid dynamics, finite element analysis, genome sequencing.
- **Gaming and Virtual Reality:** Realistic graphics rendering, physics simulations, immersive experiences.

Benefits of Parallelism and Hardware Acceleration

- *- **Improved Performance:** Parallelism increases computational speed and efficiency, allowing tasks to be completed faster.*
- ***- Scalability:** Parallel computing systems can scale to handle larger workloads and datasets.*
- ***- Cost-effectiveness**: Hardware acceleration reduces the time and resources required to process complex tasks, leading to cost savings.*
- ***- Enhanced User Experience:** Faster processing enables smoother and more responsive applications and services.*

Challenges

- ***- Concurrency Management:*** *Ensuring proper synchronization and coordination between parallel tasks to avoid race conditions and deadlocks.*
- ***- Programming Complexity:*** *Developing parallel algorithms and software requires specialized knowledge and tools.*
- ***- Resource Utilization:*** *Efficiently utilizing hardware resources to maximize performance while minimizing energy consumption.*
- ***- Data Dependencies:*** *Identifying and managing dependencies between data elements to enable effective parallelization.*

Understanding Computer Networking

- Definition: Networking involves the interconnection of multiple computers and devices to facilitate communication and resource sharing.
- Components:
  - ✓ ***Devices***: *Computers, routers, switches, modems, and other network devices.*
  - ✓ ***Protocols***: *TCP/IP, Ethernet, WiFi, HTTP, DNS, etc.*
  - ✓ ***Topology***: *Physical (e.g., star, bus, ring) and logical (e.g., clientserver, peertopeer) network configurations.*
  - ✓ ***Communication***: *Data transmission occurs via packets across networks, with each packet containing source and destination addresses.*

Importance in Software Development and Communication

- ✓ **Facilitates Collaboration**: Networking enables realtime communication and collaboration among individuals and teams, essential for software development projects.
- ✓ **Distributed Systems:** Many modern software applications are built as distributed systems, where multiple components communicate over a network to perform tasks.
- ✓ **ClientServer Architecture:** Networking enables the clientserver model, where clients request services from servers over a network, a fundamental architecture in web development and many other software systems.
- ✓ **Internet Connectivity:** The internet relies on networking protocols and infrastructure to connect billions of devices worldwide, enabling access to information, services, and resources.

Examples:

1. Web Applications
2. Messaging Apps
3. Cloud Computing
4. Online Gaming
5. Video Conferencing

Networking plays a crucial role in software development and communication by enabling connectivity, collaboration, and the exchange of data and resources over distributed systems and the internet.

Network Models and Protocols

- Network models play a crucial role in the design, implementation, and management of modern computer networks, contributing to their reliability, efficiency, and functionality.
- They provide a common language and framework for network engineers, administrators, and developers to collaborate effectively in building and maintaining robust network infrastructures.

Aspect	OSI Model	TCP/IP Model
Layers	1. Physical 2. Data Link 3. Network 4. Transport 5. Session 6. Presentation 7. Application	1. Network Interface 2. Internet 3. Transport 4. Application
Development	Developed by the International Organization for Standardization (ISO) as a conceptual framework for network communication.	Developed by the Department of Defense (DoD) for ARPANET, the precursor to the modern internet.
Approach to Protocols	Defines each layer's functions and protocols, focusing on conceptual clarity and inter-operability.	Implements a simplified model with fewer layers, focusing on practical implementation and scalability.
Implementation	Less commonly used as a reference model in networking. More theoretical.	Widely adopted as the de facto standard model for internet communication.
Popularization	Popularized in academic and educational settings for teaching network architecture.	Became the basis for the internet's protocol suite and dominates practical network implementations.
Flexibility	Offers a more comprehensive and flexible framework for understanding network communication.	Offers simplicity and scalability, making it easier to implement and manage in real-world scenarios.

IP Addressing and Subnets

- **- IPv4 vs. IPv6:**
  - ***- IPv4 (Internet Protocol version 4):*** *Uses 32-bit addresses, allowing for approximately 4.3 billion unique addresses. Widely used but facing exhaustion of address space.*
  - ***- IPv6 (Internet Protocol version 6):*** *Introduces 128-bit addresses, providing a virtually unlimited supply of unique addresses. Designed to replace IPv4 and support the growing number of devices connected to the internet.*

Subnetting Basics

- - Divides a larger network into smaller, manageable subnetworks (subnets).
- - Improves network performance, security, and scalability by segmenting traffic and controlling broadcast domains.
- - Involves dividing the IP address space into network and host portions, typically using subnet masks to identify the boundaries.

Wireless and Wired Networks

- - Comparison of Wired and Wireless Networking:
- - Wired Networks: Use physical cables (e.g., Ethernet cables) to connect devices. Typically offer higher reliability and bandwidth compared to wireless networks.
- - Wireless Networks: Utilize radio waves to transmit data between devices. Provide mobility and flexibility but may experience interference and signal degradation.

WiFi, Ethernet, and Emerging Technologies

- - WiFi (Wireless Fidelity): Enables wireless connectivity to local area networks (LANs) and the internet. Commonly used for home and business networking.
- - Ethernet: Standardized wired networking technology commonly used in LANs. Offers high speed and reliability for data transmission.
- - Emerging Technologies: Include advancements such as 5G cellular networks, Internet of Things (IoT) devices, and fiber optic broadband, which aim to enhance network performance and connectivity.

Network Security Basics

- Networks face threats such as unauthorized access, data breaches, malware, and denial-of-service (DoS) attacks.
- Security risks are compounded by the increasing complexity and interconnectedness of modern networks.

Basic Security Measures

- Firewalls: Monitor and control incoming and outgoing network traffic based on predetermined security rules. Protect against unauthorized access and malicious activity.
- VPNs (Virtual Private Networks): Securely extend private networks over public networks like the internet. Encrypt data to ensure confidentiality and privacy for remote users and branch offices.

Thank You