--New Syllabus. Hope this Helps. From Subrina Jahan ☺

Define Computer system. What are the main components of a computer system?

A computer system is an integrated set of hardware and software components that work together to perform tasks, process data, and manage information. It consists of physical devices, known as hardware, and various software programs that control and coordinate the hardware to carry out complex functions.

Main Components of a Computer System:

1. Internal Components: Internal components of a computer system refer to the essential hardware parts located within the computer casing that work together to process data and execute instructions.
    - Processor: The brain of the computer, responsible for executing instructions and managing operations.
    - Memory (RAM): Temporary storage that provides quick access to data and instructions for the processor.
    - Storage Devices: Hard Disk Drive (HDD) and Solid-State Drive (SSD) are types of storage devices that provide long-term data storage and retain data even when the computer is turned off.
    - Motherboard: The motherboard connects the CPU, RAM, storage devices, and other internal components through buses.
2. Peripheral Components: Peripheral components are external devices that connect to the computer system, enhancing its functionality and user interaction. These devices are typically not part of the core computer architecture but are essential for input, output, and storage functions.
    - Disk: Provides long-term data storage (e.g., hard drives or SSDs).
    - Display: The visual output device, such as a monitor or screen.
    - Audio System: Manages sound output and input, including speakers and microphones.
    - Ethernet Card: Allows network connectivity for data transmission between computers.

These components interact through interfaces that connect internal and peripheral devices, allowing the computer system to function efficiently.

Peripheral Definition: Computer peripherals are hardware devices that are added to a host computer to enhance its functionality. These devices are typically optional rather than essential for the computer's basic operation. Peripherals are usually connected externally to the computer through interfaces such as USB ports, although some internal devices can also be considered peripherals depending on the context.

Types of Computer Peripherals

1. Input Devices:
    - Keyboard: Used for text input.
    - Mouse: A pointing device for navigating the user interface.
    - Trackball: A stationary pointing device with a ball that the user rotates.
    - Joystick: Used primarily for gaming and simulation applications.
    - Touch Screen: A display that detects touch input.

    - Scanner: Converts physical documents into digital format.
2. Output Devices:
    - CRT Monitor: A cathode-ray tube display used for video output.
    - Video Projector: Projects computer visuals onto a larger screen.
    - Printer: Produces physical copies of digital documents.
    - Speaker: Outputs sound from the computer.
3. Storage Devices:
    - Magnetic Disk: Hard drives and floppy disks that store data magnetically.
    - Compact Disk (CD): Optical disc used to store digital data.
    - DVD: Digital versatile disc for high-capacity data storage, including video.
    - Flash Memory: USB drives and memory cards for portable storage.
4. Other Peripherals:
    - Modem: Connects a computer to the internet through telephone lines or cable.
    - Display Adapter: Converts digital signals to display on monitors.
    - Sound Card: Enhances audio output and input capabilities.

Differentiate between offline and online peripherals.

Peripherals can be classified into two main categories: offline peripherals and online peripherals, based on their connectivity and operational requirements in relation to a computer.

Offline Peripherals: Offline peripherals are devices that can operate independently of the computer, meaning they do not require a constant connection to function.

- Examples:
    - External Hard Drives: These can store data and are connected only when needed for data transfer.
    - Printers: Many printers can store print jobs and operate offline, allowing for printing without being directly connected to the computer.
- Advantages:
    - They provide flexibility in usage, as they can be operated even when the computer is turned off or disconnected.
    - Users can utilize these devices in various environments without relying on constant computer connectivity.

Online Peripherals: Online peripherals are devices that require a continuous connection to the computer to operate effectively, relying on the computer's processing power and resources.

- Examples:
    - Mouse and Keyboard: These input devices are essential for user interaction with the computer and must be connected at all times.
- Advantages:
    - Online peripherals enable real-time processing and interaction with the computer, enhancing functionality and performance.
    - They provide immediate feedback and responsiveness to user inputs.

Lecture 2

Differentiate between *Computer Organization* and *Computer Architecture*.

Computer Organization refers to the operational aspects and physical implementation of a computer system, detailing how components like the CPU, memory, and I/O devices are arranged and interact. It focuses on the internal hardware structure, including data paths, control signals, and memory layouts, and aims to ensure components work together efficiently.

Computer Architecture, on the other hand, is concerned with the conceptual design and functionality of a computer system. It defines the instruction set, system capabilities, and performance requirements, serving as a blueprint for the overall structure. Architecture focuses on high-level design choices like instruction sets, memory hierarchy, and pipelining, which influence the system's performance and cost-effectiveness.

In essence, Computer Organization is about the "how" of implementation, while Computer Architecture is about the "what" of system capabilities and design principles.

Define *Instruction Set Architecture (ISA)* and describe its key components and layers of abstraction and Importance.

The Instruction Set Architecture (ISA) is the interface between software and hardware in a computer system. It specifies the instructions the processor can execute and serves as a blueprint for hardware design.

Key Components of ISA:

1. Assembly Language: The ISA provides an assembly language interface, defining the low-level commands the processor can execute, including operations like arithmetic (addition, subtraction), logic (AND, OR), data movement (load, store), and control flow (jumps, branches). This set of commands serves as a foundation for programming the processor directly.
2. Processor State:
    - Register File (RF): A set of accessible registers that store intermediate data, allowing quick data retrieval and manipulation during execution. These registers include both general-purpose registers, used broadly, and specialized registers for specific control or status information.
    - Memory (Mem): ISA specifies memory organization, including how data is stored, accessed, and addressed in RAM or cache. Memory design and access methods are essential for efficient data handling and overall performance.
3. Instruction Set and Encoding: The ISA defines the specific operations the processor can perform and the binary encoding that represents each instruction. This includes specifying the bit patterns for different instructions, as well as the format and length of operands, which allows the hardware to decode and execute instructions.

Layers of Abstraction in ISA:

- Above the ISA: The ISA provides a simplified, consistent programming layer for high-level languages (HLL) and the operating system (OS). This layer enables developers to write software without needing to understand the intricacies of the hardware, as the ISA handles low-level details.
- Below the ISA: Beneath the ISA, the design choices guide the processor's physical construction and implementation. Here, hardware engineers use the ISA specifications to optimize performance with techniques such as pipelining (overlapping instruction execution stages) and caching (temporary data storage to speed up access).

Importance: The *Instruction Set Architecture (ISA)* is essential in computer architecture as it defines the interface between software and hardware, specifying the processor's supported instructions and capabilities. This consistency allows software to run on any hardware implementing the same ISA, ensuring backward compatibility and fostering extensive software ecosystems. The ISA also guides hardware design, allowing for optimizations like pipelining and caching that enhance performance without changing the software interface. By balancing power, performance, and cost, the ISA enables efficient and scalable computing systems, forming the foundation of modern computer architecture.

Machine Instruction

A *machine instruction* is a low-level command in binary or assembly language that a processor directly interprets and executes. Each machine instruction performs a very specific task, such as adding two numbers, moving data between memory and registers, or making a comparison for decision-making.

Why Machine Instructions Are Considered Primitive Compared to High-Level Languages (HLLs)

Machine instructions are considered *primitive* because they provide only basic operations and lack the abstraction found in high-level languages (HLLs) like C, Python, or Java. Unlike HLLs, which can perform complex operations in a single line of code (e.g., handling loops, functions, and conditionals), machine instructions handle one small operation at a time and require multiple instructions to achieve what a single line of HLL code can do. This makes machine instructions less readable, harder to write, and more challenging to understand compared to high-level language code, which is closer to human language and thought.

What are the primary design goals of an instruction set?

1. Maximize Performance: The instruction set should allow the processor to run tasks quickly and efficiently, so programs execute faster.
2. Minimize Cost: The instructions should be simple enough to keep the hardware affordable and easier to build, making the processor cheaper.
3. Reduce Design Time: By keeping the instruction set straightforward, processors can be designed and developed faster, helping new technology come out sooner.

Explain the structure of a MIPS instruction with example.(Microprocessor without Interlocked Pipelined Stages)

A MIPS instruction is 32 bits long and follows a fixed format, making it straightforward and efficient for the processor to decode. MIPS instructions are typically organized in three main formats: R-type, I-type, and J-type, each with different fields that

serve specific purposes. Here's a breakdown of each:

R-Type (Register-Type): This format is used for arithmetic and logical operations that work directly on registers.
- Opcode (6 bits): Specifies the operation.
- rs (5 bits): The first source register.
- rt (5 bits): The second source register.
- rd (5 bits): The destination register (where the result will be stored).
- shamt (5 bits): Shift amount (used in shift instructions).
- funct (6 bits): Further specifies the operation, allowing more distinct operations within the same opcode.

I-Type (Immediate-Type): Used for instructions involving constants (immediate values) or memory access.
- Opcode (6 bits): Specifies the operation.
- rs (5 bits): Source register.
- rt (5 bits): Target register (where the result will be stored).
- Immediate (16 bits): A constant value or address offset.

J-Type (Jump-Type): Used for jump instructions.
- Opcode (6 bits): Specifies the operation.
- Address (26 bits): The jump target address.

Ex-

C code: A=B+C
MIPS code: add $s0, $s1, $s2

In binary, this might look like:

000000 10001 10010 01000 00000 100000

op rs rt rd shamt funct

1. 000000 – Opcode for R-type (register) operations.
2. 10001 – rs, source register for B ($s1).
3. 10010 – rt, second source register for C ($s2).
4. 01000 – rd, destination register for A ($s0).
5. 00000 – shamt, which is 0 for non-shift operations.
6. 100000 – funct code for the addition operation.

What is the function of load and store instructions in MIPS architecture, and how do they facilitate data processing within the CPU?

In MIPS architecture, load and store instructions are critical for enabling the CPU to interact with memory, facilitating data processing during program execution.

1. Function of Load Instructions:
    - Load instructions lw (load word), are used to transfer data from a specific memory address into a CPU register. This is essential because the CPU can only perform operations on data that resides in its registers, not directly in memory.
    - For example, the instruction lw $t0, 32($s3) loads a word from memory at the address calculated by adding the offset 32 to the value in register $s3, placing the retrieved data into register $t0. This allows the CPU to access and manipulate the data stored in memory.
2. Function of Store Instructions:
    - Store instructions sw (store word), are used to transfer data from a CPU register back to a specific memory address. This is important for saving the results of computations or updating data in memory.
    - For instance, the instruction sw $t0, 32($s3) stores the contents of register $t0 into the memory address specified by the base address in $s3 plus the offset 32. This operation ensures that any computed results or updated values can be saved back to memory for future access.

Together, load and store instructions create a mechanism for the CPU to manage data efficiently. When the CPU needs to perform calculations or operations, it first loads the necessary data from memory into registers. After processing, it can store the results back into memory.

Explain the differences between RISC (Reduced Instruction Set Computing) and CISC (Complex Instruction Set Computing) architectures.

1. RISC uses a small set of simple instructions, while CISC includes a larger, more complex set of instructions capable of executing multiple operations in a single instruction.
2. In RISC, each instruction typically takes one clock cycle to execute, allowing for faster overall performance. In contrast, CISC instructions may take multiple clock cycles to execute due to their complexity, which can slow down execution.
3. RISC instructions perform basic operations and require more instructions to complete complex tasks. CISC instructions can access memory directly and perform multiple tasks in one instruction, reducing the total number of instructions needed.
4. MIPS is a classic example of a RISC architecture, using straightforward instructions like load, store, and basic arithmetic. The x86 architecture is a CISC design that features complex instructions such as MOV, which can load and store data in a single step.
5. RISC architectures generally allow for better pipelining and efficient execution due to their simplicity. CISC architectures more flexible due to their complex instructions, can suffer from slower instruction execution and more complicated decoding processes.

Describe the process of fetching and executing an instruction in the MIPS architecture.

1. The program counter (PC) provides the address of the next instruction to be executed. This address is sent to memory, where the instruction is fetched and loaded into the instruction register.
2. The fetched instruction is decoded to determine the operation type and the operands required.
3. The instruction then performs the required operation using the arithmetic-logic unit (ALU).
4. For load (lw) and store (sw) instructions, memory is accessed either to load data into a register (for lw) or to store data from a register to memory (for sw).
5. The result of the instruction (if any) is written back to the specified register. For control flow instructions like branches (beq) and jumps (j), the PC is updated to alter the program's execution path.

What are the main types of instructions in a MIPS subset, and what functions do they serve in processing data and controlling program flow?

In a MIPS subset, there are three main types of instructions:

1. Arithmetic-Logic Instructions: These instructions handles mathematical and logical operations, like adding and comparing values. They work with data in the processor's registers to perform essential calculations.
2. Memory Reference Instructions: These instructions load data from memory into registers or store data from registers back into memory. This allows the processor to access and save data during program execution.
3. Control Flow Instructions: These instructions change the order of execution in a program. They allow the program to jump to a different part or branch based on certain conditions, making loops and decision-making possible.

Describe the working process of a register in a MIPS subset.

In a MIPS subset, a register serves as a small, fast storage location that holds temporary data values for the processor. During instruction execution, the register is used to store operands for arithmetic or logical operations, as well as to hold intermediate results. When an instruction is executed, the relevant data is loaded into the register from memory or another register, the ALU performs the required operation, and the result can be stored back in the same or a different register for further use or output.

Explain how an adder functions in a MIPS subset.

The adder in a MIPS subset is primarily responsible for performing addition operations, which include incrementing the program counter (PC) to fetch the next instruction. When the PC value is passed to the adder, it adds a fixed value (usually 4 bytes for MIPS instructions) to the current PC value. The result is then sent back to the PC to update it for the next instruction to be fetched from memory. The adder can also be used in arithmetic operations within the ALU.

What is the role of the ALU (Arithmetic Logic Unit) in a MIPS subset, and how does it work?

The ALU is a critical component of a MIPS subset that performs arithmetic and logic operations. When an instruction requiring computation is executed, the ALU receives operands from the register file. It performs the specified operation, such as addition, subtraction, logical AND, or OR, based on the

instruction's opcode. The result of this operation is then sent back to a register for storage or further processing. The ALU's design allows for quick execution of mathematical and logical functions essential for program execution.

Describe the function of a multiplexer in a MIPS subset.

A multiplexer (MUX) in a MIPS subset is used to select one input from multiple sources and route it to a single output. During instruction execution, the MUX takes inputs from various registers or data paths and uses control signals to determine which input to send to the ALU or another destination. This selection capability allows for efficient data management and minimizes the number of required connections, facilitating the flow of information based on the current operation being performed.

Explain the working process of the register file in a MIPS subset.

The register file in a MIPS subset is a collection of registers that allows the processor to quickly access data. It typically contains a set number of registers (e.g., 32 in MIPS) that can be read from or written to during instruction execution. When an instruction is executed, the register file receives requests to read specific registers to obtain operands for operations or to write results back to designated registers after processing. The register file enables fast access to frequently used data, reducing the need to access slower memory.

What is the role of program memory in a MIPS subset, and how does it function?

Answer: Program memory in a MIPS subset stores the instructions of the program to be executed. During the instruction fetch phase, the program counter (PC) provides the address of the next instruction, which is sent to program memory. The memory retrieves the instruction stored at that address and sends it back to the processor for decoding and execution. Program memory allows the processor to access a sequence of instructions, enabling the execution of the program as designed.

Explain the working process of bit manipulation components in a MIPS subset.

Answer: Bit manipulation components in a MIPS subset perform specific operations at the bit level, such as shifting (left or right) and masking (using logical operations to isolate specific bits). These components can be integrated within the ALU or used as separate units. For example, when an instruction requires shifting bits to the left, the bit manipulation component takes the input value, shifts the bits accordingly, and outputs the modified value. These operations are essential for tasks like bit-level data processing, optimizing data storage, and implementing certain algorithms.

What is pipelining in the context of computer architecture, and how does it improve throughput?

Pipelining is a technique in computer architecture that allows multiple instruction phases to overlap in execution. Instead of waiting for one instruction to complete before starting the next, pipelining divides instruction processing into discrete stages (e.g., Instruction Fetch, Decode, Execute, Memory Access, and Write Back). This overlapping enables a new instruction to be fetched every clock cycle, leading to faster overall throughput. By increasing the number of instructions that can be processed simultaneously, pipelining enhances the efficiency of the CPU.

Describe the process of executing an instruction in a pipelined architecture, highlighting the key functions at each step and discussing the benefits of this approach.

In a pipelined architecture, the execution of an instruction is divided into several distinct stages that occur in overlapping cycles, allowing multiple instructions to be processed simultaneously.

1. Instruction Fetch (IF): The process begins by fetching the instruction from memory using the address specified by the Program Counter (PC). The PC is then incremented to point to the next instruction for the subsequent cycle.
2. Instruction Decode (D): Once fetched, the instruction is decoded to determine its operation and the required operands. This stage involves identifying the opcode and accessing the register file to retrieve the values of the source registers.
3. Register File (RF): The register file holds general-purpose registers used during execution. The values from the source registers are read and prepared for processing in the next stage.
4. Execution / Address Generation (EX/AG): In this stage, the Arithmetic Logic Unit (ALU) performs the required operation (such as addition, subtraction, etc.) on the operands. For memory-related instructions, the effective address for memory access is calculated.
5. Memory Access (M): If the instruction involves data transfer, this stage accesses the data memory. For load instructions, data is read from the calculated address; for store instructions, data from a register is written to memory.
6. Write Back (WB): Finally, the result of the operation or the loaded data is written back to the appropriate destination register in the register file, making it available for future instructions.

Define the types of hazards that can occur in pipelining and explain how each type affects instruction execution.

In pipelined architectures, hazards are conditions that can prevent the next instruction in the pipeline from executing in the designated clock cycle. There are three primary types of hazards: control hazards, data hazards, and structural hazards.

 Control Hazards:

- Control hazards arise from branch instructions that alter the flow of execution. When a branch instruction is encountered, the processor may not immediately know the target address of the next instruction, leading to uncertainty in fetching the correct instruction. This uncertainty can cause pipeline stalls, where subsequent instructions are delayed until the branch decision is resolved. As a result, the pipeline may fetch incorrect instructions, instruction stalling, which decreases overall throughput.

 Data Hazards:

- Data hazards occur when there are dependencies between instructions that operate on the same data. The main types of data hazards include:
    - Read After Write (RAW): This occurs when an instruction needs to read a value before it is written by a prior instruction.

- Write After Read (WAR): This happens when a write operation is executed before a read operation has completed, which can lead to incorrect results.
- Write After Write (WAW): This occurs when two instructions write to the same location, and the order of writes affects the final value.

Data hazards can lead to incorrect execution results if not properly managed.

 Structural Hazards:

Structural hazards arise when multiple instructions require the same hardware resource simultaneously, such as the ALU or memory. This can occur if the pipeline stages try to access the same resource at the same time .So some instructions wait for the resource to become available, resulting in pipeline stalls. This resource contention can decrease the overall performance and efficiency of the processor.

Explain various techniques for improving branch performance in pipelined processors.

Improving branch performance in pipelined processors is essential to minimize stalls and maintain smooth execution flow. Key techniques include:

1. Branch Elimination: Branch elimination involves replacing branch instructions with other logic or instructions that achieve the same outcome, reducing the need for branching. This minimizes the disruption to the pipeline and helps avoid the delay that comes with evaluating branch conditions.
2. Branch Speed-Up: Reducing the time needed to compute condition codes (CC) and target instruction fetch (TIF) is another way to speed up branches. By making this process faster, the pipeline can determine whether a branch should be taken or not in less time, reducing the likelihood of stalls.
3. Branch Prediction: Branch prediction guesses the likely outcome of a branch (whether it will be taken or not) and allows the pipeline to proceed based on this prediction. If the prediction is correct, the pipeline continues without delay; if incorrect, the processor undoes the incorrect path and takes the correct one, minimizing the time lost to incorrect branching.
4. Branch Target Capture: Branch target capture uses historical data on previous branch behavior to predict future branches more accurately. This technique helps to improve the accuracy of branch prediction by using patterns from prior executions, allowing the processor to anticipate the correct path with fewer mispredictions.

Why are superscalar processors popular compared to other processor architectures?

A *superscalar processor* is a type of processor that can execute multiple instructions in a single clock cycle by sending them to different functional units within the processor. This ability enables faster and more efficient processing.

Reasons for Popularity:

- Binary Code Compatibility: Superscalar processors can run the same programs as scalar processors in the same family, without requiring changes to the code. This compatibility makes it easier to adopt superscalar processors for existing software.

- Shared Compiler Use: The same compiler can generate code for both scalar and superscalar processors in the same family, so developers don't need a separate compiler. This flexibility simplifies development and keeps code consistent across different processors.
- Simpler Programming than VLIW: Superscalar processors can automatically handle multiple instructions at once, without requiring programmers to organize instructions in parallel. This is simpler than VLIW (Very Long Instruction Word) processors, which need specific instruction grouping and scheduling.
- Efficient Code Density: Superscalar processors generally have higher code density than VLIW processors, meaning they use memory more efficiently. This results in smaller program sizes and better performance in terms of storage and execution.

Keyboard

A computer keyboard is a peripheral designed for the input of text and characters and also to control the operation of a computer Physically, computer keyboards are an arrangement of rectangular buttons, or "keys". Keyboards typically have characters engraved or printed on the keys; in most cases, each press of a key corresponds to a single written symbol. However, to produce some symbols requires pressing and holding several keys simultaneously or in sequence; other keys do not produce any symbol, but instead affect the operation of the computer or the keyboard itself.

How Does a Dome Switch Keyboard Work?

1. When a key is pressed, it pushes down on a rubber dome located beneath it. The underside of this dome has a conductive contact, which connects two conductive lines on the circuit board below.
2. This connection bridges the gap between the conductive lines, allowing current to flow and changing the signal from open to closed.
3. The chip within the keyboard continuously scans the circuit. When a specific line pair shows a change in signal, the chip generates a unique "make code" for the pressed key. This code is sent to the computer, either through a keyboard cable (with electrical pulses representing bits) or wirelessly, and may be repeated for confirmation.
4. Inside the computer, a decoding chip receives the signal and translates it into the corresponding keypress. The computer then interprets the command or displays the character on the screen.
5. When the key is released, the keyboard sends a "break code" (distinct from the make code) to indicate the key is no longer pressed. If the break code is missed, the keyboard may mistakenly believe the key is still pressed. Pressing and releasing the key again sends another break code to resolve this.

What is key bouncing? How a keyboard matrix is formed in keyboard interface?

When a key is pressed, the switch mechanism beneath it completes the circuit and allows a tiny amount of current to flow. However, the mechanical action of the switch causes slight vibrations, known as *bounce*. This bounce results in multiple signals being sent to the processor as the switch settles into a stable state.

The keyboard matrix is a grid of circuits beneath the keys. In most keyboards, each circuit is broken at a point under each key. Pressing a key completes the circuit, allowing current to flow through that specific spot in the grid. The processor in the keyboard constantly scans the matrix for any closed circuits, helping it detect which key has been pressed.
To identify the key press, the processor compares the key's position on the matrix with a character map stored in the keyboard’s memory. This map is a chart that tells the processor which character or function each key represents. For example, it lets the processor know that pressing "A" alone means a lowercase "a," but pressing "Shift" and "A" together means a capital "A."

What is Keyboard Interfacing? What is Scanning in a Keyboard, and What is Scan Time?

Keyboard Interfacing is the process of connecting a keyboard to a computer or microcontroller, allowing communication between them. This process involves converting key presses into electronic signals that

the computer can interpret. Keyboards interface with computers using methods like USB or Bluetooth for data input.

Scanning in a Keyboard refers to the technique used to determine which key is pressed. Most keyboards use a matrix layout, where keys are positioned at the intersections of rows and columns. The keyboard controller sends signals down each row and checks the columns for changes. When a key is pressed, it completes the circuit between a row and a column, enabling the controller to detect which key has been activated. The controller then sends a unique code, known as a "make code," to the computer.

Scan Time is the time it takes for the keyboard controller to complete one full cycle of scanning all keys. A shorter scan time leads to better responsiveness, allowing the keyboard to detect key presses more quickly. In contrast, a longer scan time may result in missed inputs, especially during fast typing.

Write short notes on the Hall effect keyboard.

A Hall effect keyboard uses magnets and Hall effect sensors instead of mechanical switches to detect key presses. When a key is pressed, it moves a magnet, and the Hall effect sensor detects this magnetic field change, generating a small electric voltage. This unique mechanism enables high durability, as there's no physical contact between parts.

Advantages:

- Hall effect keyboards can handle millions of keystrokes without failure, making them ideal for critical environments like aircraft cockpits and nuclear power plants.
- These keyboards can be made waterproof and resistant to dust and contaminants, suitable for industrial use.

Drawbacks:

- Due to the need for individual magnets, sensors, and custom control electronics, Hall effect keyboards are more expensive than standard keyboards.

Write down the working principle of Hall effect keyboard.

A Hall effect keyboard uses magnets and Hall effect sensors instead of mechanical switches to detect key presses. When a key is pressed, it moves a magnet, and the Hall effect sensor detects this magnetic field change, generating a small electric potential. This unique mechanism enables high durability, as there's no physical contact between parts.

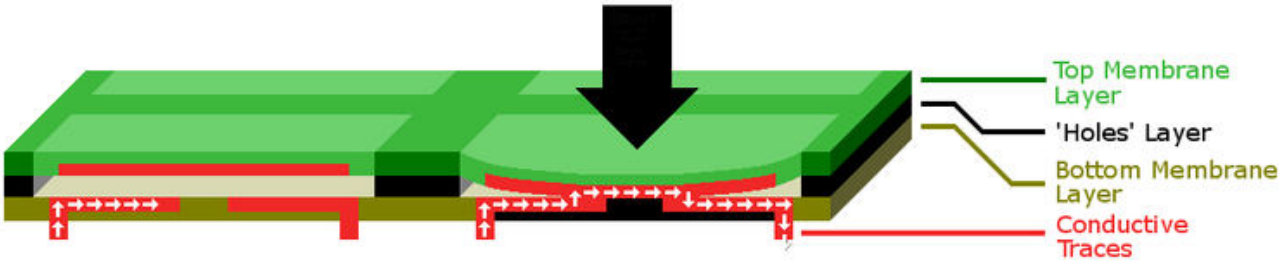
Working Principle of Hall Effect Keyboards:

1. Each key has a small magnet attached to it. When the key is pressed, the magnet moves closer to a Hall effect sensor located beneath it.
2. The Hall effect sensor detects the magnetic field produced by the magnet. According to the Hall effect, when a magnetic field is applied perpendicular to a current flowing through a conductor, a voltage is generated at a right angle to both the current and the magnetic field.

3. As the key is pressed and the magnet's proximity changes, the sensor generates a voltage signal. This signal is then processed to identify the specific key press.
4. Since the detection relies on a magnetic field and not physical contact, Hall effect keyboards experience minimal wear, allowing for millions of key presses without degradation.

Define the membrane keyboard layers

A membrane keyboard is a type of keyboard that uses a flexible membrane rather than individual mechanical switches for each key. It has a continuous surface. Due to their quiet operation, membrane keyboards are sometimes preferred for gaming or office environments where noise reduction is important. It consists of three main layers:



1. Top Membrane Layer:

   This layer has conductive traces on its underside, representing each key's position. When a key is pressed, this layer bends down to make contact with the bottom layer, completing the circuit and allowing current to flow.

2. Spacer Layer:

   The spacer layer is positioned between the top and bottom membranes and contains holes aligned with each key. It keeps the top and bottom layers apart under normal conditions, preventing accidental key presses. When a key is pressed, the top membrane moves through the spacer layer's hole to connect with the bottom layer.

3. Bottom Membrane Layer:

   The bottom layer also has conductive traces aligned with those on the top layer. When a key is pressed, the top membrane touches the bottom layer's traces, completing the circuit. This closed circuit sends a signal to the computer to register the key press.

Write down the difference between laser keyboard and dome switch keyboard

Feature	Laser Keyboard	Dome Switch Keyboard
Technology	projects a virtual keyboard onto a flat surface using a laser or beamer:	Uses a rubber dome beneath each key that completes a circuit when pressed.
Portability	Highly portable and compact, suitable for use on any flat surface.	Generally bulkier and requires a stable surface for placement.

Tactile Feedback	No tactile feedback, leading to less user comfort and more typing errors.	Provides tactile feedback due to the rubber dome, offering a better typing experience.
Typing Experience	Typing can be cumbersome, with a higher chance of accidental keystrokes.	Offers a more traditional typing experience, though often less precise than mechanical keyboards.
Durability	Vulnerable to malfunction; a failure in the laser makes the entire device unusable.	More durable overall; can still function even if some parts are damaged.
Cost	Typically more expensive due to the advanced technology.	Generally less expensive to produce, making them more budget-friendly.
Environmental Sensitivity	Sensitive to ambient light and can be disrupted by accidental movements or surface changes.	Functions well on a variety of surfaces, but may require a solid foundation to prevent key bounce.

Capacitive Keyboard

Working Principle: A capacitive keyboard operates by detecting changes in electrical capacitance when a key is pressed. Here's how it works:

1. Each key is equipped with a small plate that is spring-loaded and positioned above another plate beneath it.
2. When a key is pressed, the top plate moves closer to the bottom plate, changing the capacitance between them. This change in capacitance allows the keyboard's processor to detect the key press.
3. Unlike traditional keyboards that rely on physical contact to close a circuit, capacitive keyboards have a constant flow of current through the key matrix. This design means there is no wear from physical contact, contributing to the keyboard's longevity.

Advantages:

1. Capacitive keyboards are resistant to wear and tear, water, dirt, and foreign objects, making them suitable for various environments.
2. They typically have a longer life compared to mechanical or dome switch keyboards due to their non-mechanical design.
3. Since the plates do not physically touch, capacitive keyboards eliminate the problem of key bounce, ensuring accurate and reliable keystrokes.
4. The lack of mechanical parts can lead to a smoother and quieter typing experience.

Disadvantages:

1. Capacitive keyboards are usually more expensive than traditional keyboards due to their advanced technology and durability.
2. The keys may register a press if an object inadvertently touches the surface, which can lead to unwanted inputs.

Mouse

In computing, a mouse (plural mice or mouses) is a pointing device that detects two-dimensional motion relative to its supporting surface. Physically, a mouse consists of a small case, held under one of the user's hands, with one or more buttons. It sometimes features other elements, such as "wheels", which allow the user to perform various system-dependent operations, or extra buttons or features can add more control or dimensional input. The mouse's motion typically translates into the motion of a pointer on a display.

Varieties of Mice

Mouse come in three basic types:
1. Mechanical: The basic design is the mechanical mouse with a rubber or plastic ball on its underside that rolls in all direction. The cursor on the screen is moved as the mechanical sensors within the mouse detect the rolling ball's direction.
2. Opto-mechanical: The optomechanical mouse uses optical sensors to detect the motion of the ball.
3. Optical: In optical mice, the red glow comes from a small LED (Light Emitting Diode) inside the mouse. This LED shines onto the surface under the mouse, and a sensor captures the reflected light to detect movement, which gives the mouse a subtle high tech look.

How Does a Computer Mouse Work? / How Does Mechanical Computer Mouse Work?

The main goal of any mouse is to translate the motion of the hand into signals that the computer can use. Let's take how it works

1. A ball inside the mouse touches the desktop and rolls when the mouse moves.
2. Two rollers inside the mouse touch the ball. One of the rollers is oriented so that it detects motion in the X direction, and the other is oriented 90 degrees to the first roller so it detects motion in the Y direction. When the ball rotates, one or both of these rollers rotate as well.
3. The rollers each connect to a shaft, and the shaft spins a disk with holes in it. When a roller rolls, its shaft and disk spin.
4. On either side of the disk there is an infrared LED and an infrared sensor. The holes in the disk break the beam of light coming from the LED so that the infrared sensor sees pulses of light. The rate of the pulsing is directly related to the speed of the mouse and the distance it travels.
5. An on-board processor chip reads the pulses from the infrared sensors and turns them into binary data that the computer can understand. The chip sends the binary data to the computer through the mouse's cord.

How Does Optical Computer Mouse Work?

The optical mouse actually uses a tiny camera to take thousands of pictures every second. It able to work on almost any surface without a mouse pad. Most optical mice use a small, red
(LED) that bounces light off that surface onto a complimentary metal-oxide semiconductor (CMOS) sensor. Let's take how it works

1. The CMOS sensor sends each image to a digital signal processor (DSP) for analysis.
2. The DSP detects patterns in the images and examines how the patterns have moved since the previous image.
3. Based on the change in patterns over a sequence of images, the DSP determines how far the mouse has moved and sends the corresponding coordinates to the computer.

4. The computer cursor moves the cursor on the screen based on the coordinates received from the mouse. This happens hundreds of times each second, making the cursor appear to move very smoothly.

What do you mean by the accuracy of an optical mouse? Describe the factors that affect the quality of an optical mouse.

The accuracy of an optical mouse is its ability to precisely detect and translate movements into on-screen cursor actions. Factors Affecting the Quality of an Optical Mouse:

1. Resolution (DPI - Dots Per Inch): Resolution is the measure of how many pixels per inch the optical sensor can "see" as the mouse moves. Expressed in DPI, higher values like 1600 DPI mean greater sensitivity, so smaller movements result in larger on-screen shifts.
2. Size of the Optical Sensor: A larger optical sensor, usually between 16 x 16 and 30 x 30 pixels, can capture more detail in each movement, assuming the mouse's components can process this data efficiently.
3. Refresh Rate: The refresh rate measures how frequently the sensor samples images per second. Higher refresh rates, from 1500 to 6000 samples per second, improve the mouse's responsiveness, reducing any delay in movement detection, especially in high-speed activities.
4. Image Processing Rate: Combining sensor size and refresh rate, the image processing rate reflects how fast the mouse processes captured images. Faster processing yields smoother, more accurate cursor movements.
5. Polling Rate: The polling rate refers to how often the mouse reports its position to the computer, measured in Hertz (Hz). Common rates include 125 Hz, 500 Hz, and 1000 Hz. A higher polling rate means quicker communication between the mouse and computer, making the cursor response feel more immediate. This is particularly beneficial in gaming and fast-paced tasks.
6. Maximum Speed: Maximum speed is the fastest rate at which you can move the mouse while maintaining accurate tracking, typically ranging from 16 to 40 inches per second.

Can a mouse's sensitivity (DPI) be adjusted, and why might someone want to change it?

Yes, a mouse's sensitivity, measured in DPI (Dots Per Inch), can often be adjusted, especially on gaming and high-performance mice. DPI adjustment allows users to change how much the cursor moves in response to physical movements of the mouse.

Reasons for Adjusting DPI:

1. Higher DPI makes the cursor move faster with smaller hand movements, useful for fast-paced tasks like gaming. Lower DPI offers finer control, which is ideal for tasks requiring precision, such as graphic design or photo editing.
2. Some users prefer high sensitivity for general use, while others find lower sensitivity more comfortable and easier to control.
3. On high-resolution screens, higher DPI helps the cursor move efficiently across the screen without large hand movements, making navigation easier.

Optical vs. Mechanical Mice

Optical mice have several advantages over traditional mechanical (track-ball) mice:

1. Optical mice have no moving parts, which reduces wear and minimizes the risk of failure.
2. Without a ball to pick up dirt, optical mice remain cleaner, and there's less chance of dirt affecting the tracking sensors.
3. Optical mice typically have a higher tracking resolution, allowing for smoother and more precise cursor movements compared to mechanical mice.
4. Unlike mechanical mice, which require a mouse pad or special surface, optical mice can operate on a variety of surfaces.

Drawbacks of Optical Mice:

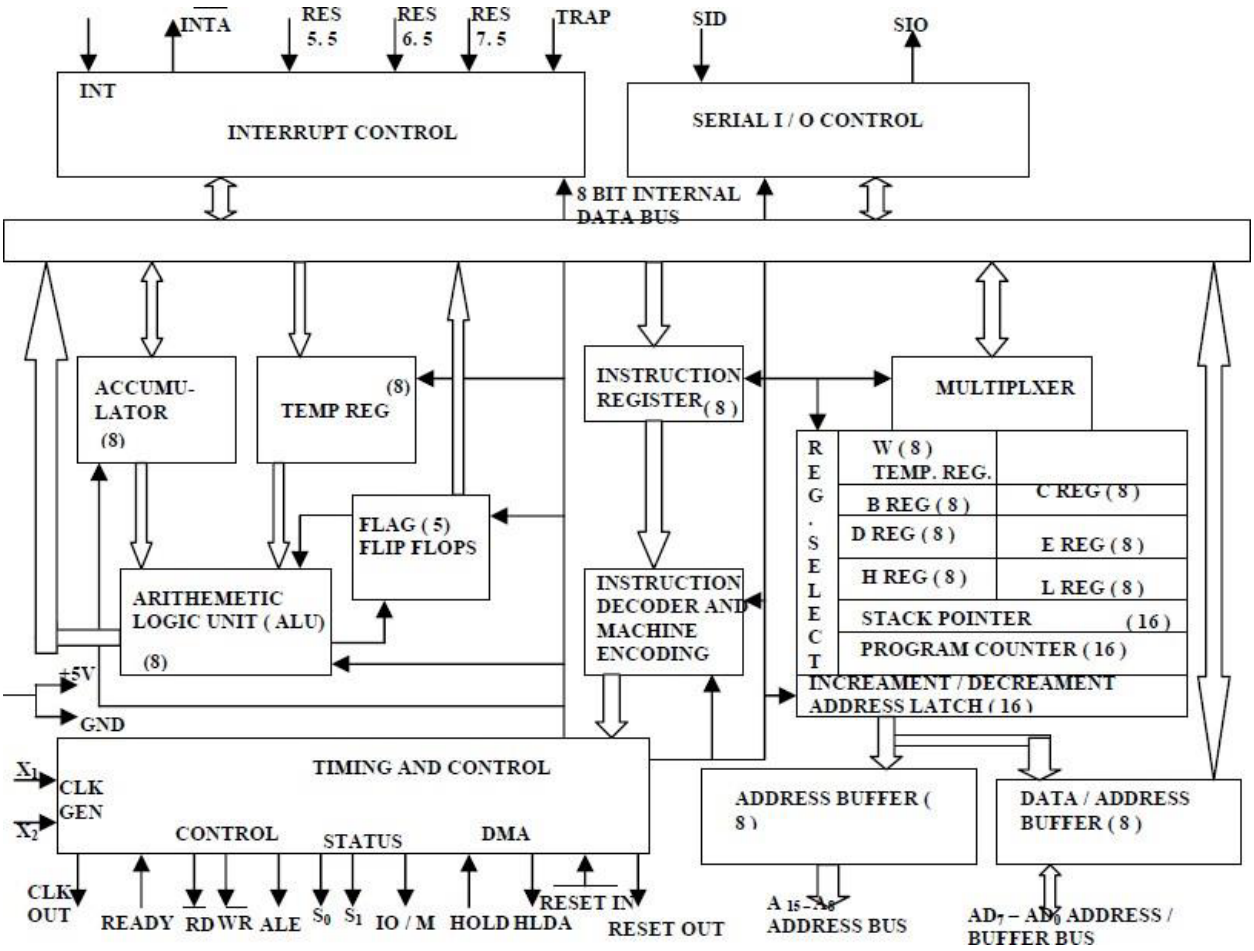
1. Optical mice can struggle on glossy or transparent surfaces, causing unpredictable cursor movement.
2. Optical mice, especially wireless models, generally consume more power than mechanical mice. While this may not affect wired models, it can reduce battery life in wireless versions, impacting their practicality for prolonged use on battery-powered devices like laptops.

8085 Microprocessor

The key features of the 8085 microprocessor include:

1. 8-bit Microprocessor: The 8085 is an 8-bit microprocessor, meaning it processes 8 bits of data at a time. It can handle operations on data that is 8 bits wide.
2. 16-bit Address Bus: It has a 16-bit address bus, which allows it to access up to 64KB of memory. The address bus is used to identify memory locations.
3. 8-bit Data Bus: The 8085 has an 8-bit data bus for data transfer between the microprocessor and peripherals or memory.
4. 40-Pin Package: It is available in a 40-pin Dual In-line Package (DIP), which is a common packaging format for ICs.
5. Clock Speed: The clock speed ranges from 3 MHz to 6 MHz, which determines the speed at which the microprocessor executes instructions.
6. 6 General-Purpose Registers: It has six 8-bit general-purpose registers (B, C, D, E, H, L), which can also be paired together as BC, DE, and HL to perform 16-bit operations.
7. Accumulator (A): It has an 8-bit accumulator, used in arithmetic and logic operations, which is a central part of its architecture.
8. Simple Instruction Set: The 8085 microprocessor uses a simple instruction set consisting of operations such as data transfer, arithmetic, logic, branch, and control instructions.
9. Interrupt System: It supports five hardware interrupt signals (TRAP, RST7.5, RST6.5, RST5.5, and INTR), which help manage asynchronous events.
10. Serial I/O Control: It includes provisions for serial input and output (SID and SOD), allowing serial communication.

Internal Architecture of 8085:



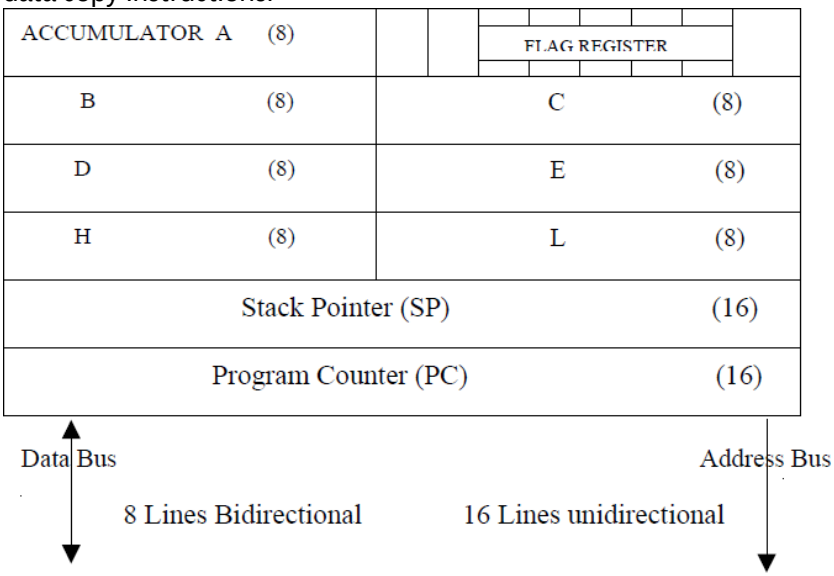
Arithmetic and Logic Unit

The ALU performs the actual numerical and logical operations such as Addition (ADD), Subtraction (SUB), AND, OR etc. It uses data from memory and from Accumulator to perform operations. The results of the arithmetic and logical operations are stored in the accumulator.

Registers

The 8085 includes six registers, one accumulator and one flag register, as shown in Fig. 3. In addition, it has two 16-bit registers: stack pointer and program counter. They are briefly described as follows.
The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L. they can be combined as register pairs - BC, DE and HL to perform some

16- bit operations. The programmer can use these registers to store or copy data into the register by using data copy instructions.

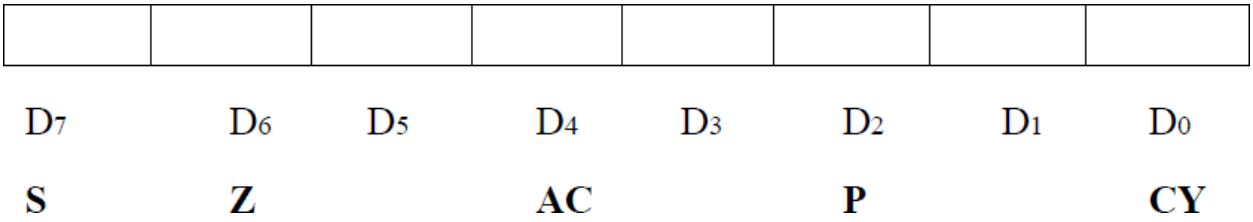


Accumulator

The accumulator is an 8-bit register that is a part of ALU. This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flag register

The ALU includes five flip-flops, which are set or reset after an operation according to data condition of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags. Their bit positions in the flag register are shown in Fig. 4. The microprocessor uses these flags to test data conditions



Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

Stack Pointer (SP)

The stack pointer is also a 16-bit register, used as a memory pointer. It points to a memory location in R/W memory, called stack. The beginning of the stack is defined by loading 16- bit address in the stack pointer.

Instruction Register/Decoder
It is an 8-bit register that temporarily stores the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and decodes or interprets the instruction. Decoded instruction then passed to next stage.

Interrupt control:
Whenever a microprocessor is executing the main program and if suddenly an interrupt occurs, the microprocessor shifts the control from the main program to process the incoming request. After the request is completed, the control goes back to the main program. There are 5 interrupt signals in 8085 microprocessors: INTR, TRAP, RST 7.5, RST 6.5, and RST 5.5.
Priorities of Interrupts: TRAP > RST 7.5 > RST 6.5 > RST 5.5 > INTR

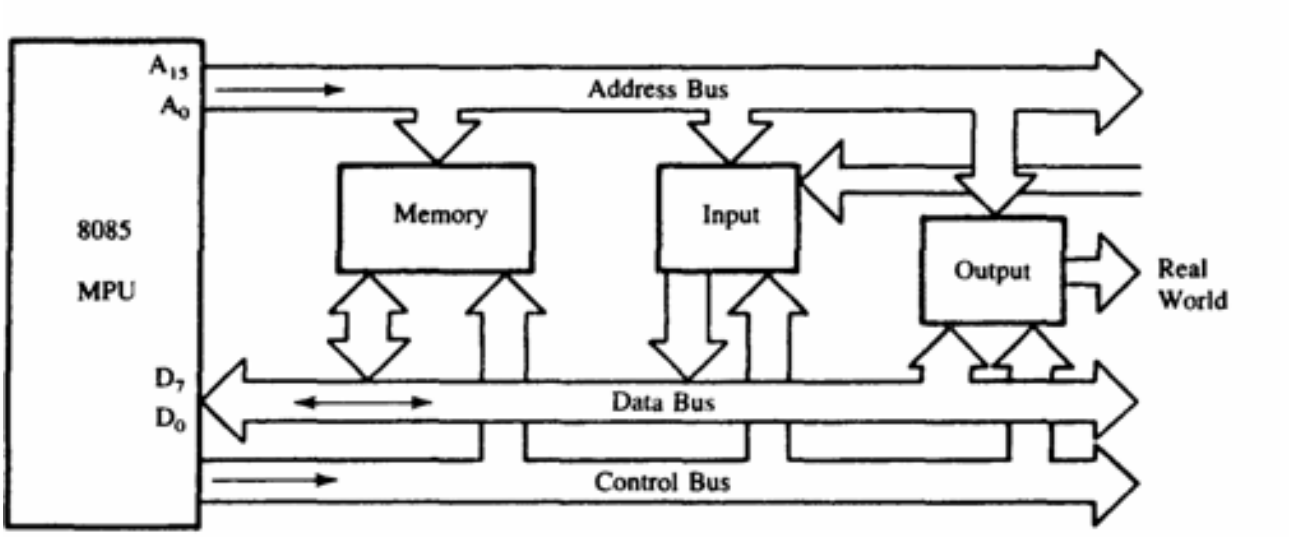
Serial Input/output control:
It controls the serial data communication by using Serial input data and Serial output data.
Serial Input/Output control in the 8085 microprocessor refers to the communication of data between the microprocessor and external devices in a serial manner, i.e., one bit at a time. The 8085 has a serial I/O port (SID/SOD) for serial communication. The SID pin is used for serial input and the SOD pin is used for serial output. The timing and control of serial communication is managed by the 8085's internal circuitry.

Timing and control unit:
The timing and control unit comes under the CPU section, and it controls the flow of data from the CPU to other devices. It is also used to control the operations performed by the microprocessor and the devices connected to it. There are certain timing and control signals like Control signals, DMA Signals, RESET signals and Status signals.

Status Flag

- ✓ Sign Flag: It is used to indicate whether the result is positive or negative. It will set (SF=1) if the result is –ve and if the result +ve then SF=0.
- ✓ Zero Flag: It is used to indicate whether the result is a Zero or non-zero. It will set (ZF=1) if the result is zero else ZF=0.
- ✓ Auxiliary carry Flag: It is used to indicate whether or not the ALU has generated a carry/Borrow from D3 bit position to D4 bit. It will set if there was a carry out from bit 3 to bit 4 of the result else AF=0. The auxiliary carry flag is used for binary coded decimal (BCD) operations.
- ✓ Parity Flag: It is used to indicate parity (Even or Odd) of the result. It will set if the parity is even else PF =0.
- ✓ Carry Flag: It is used to indicate whether a carry/Borrow has been generated /occurred during addition/subtraction It will set if there was a carry is generated from the MS-bit during addition, or borrow during subtraction/comparison else CF=0.

BUS Structure:



Data Bus:

Data bus carries data in binary form between microprocessor and other external units such as memory. It is used to transmit data i.e. information, results of arithmetic etc between memory and the microprocessor. Data bus is bidirectional in nature. The data bus width of 8085 microprocessor is 8-bit i.e. 28 combination of binary digits and are typically identified as D0 – D7. Thus size of the data bus determines

what arithmetic can be done. If only 8-bit wide then largest number is 11111111 (255 in decimal). Therefore, larger numbers have to be broken down into chunks of 255. This slows microprocessor.

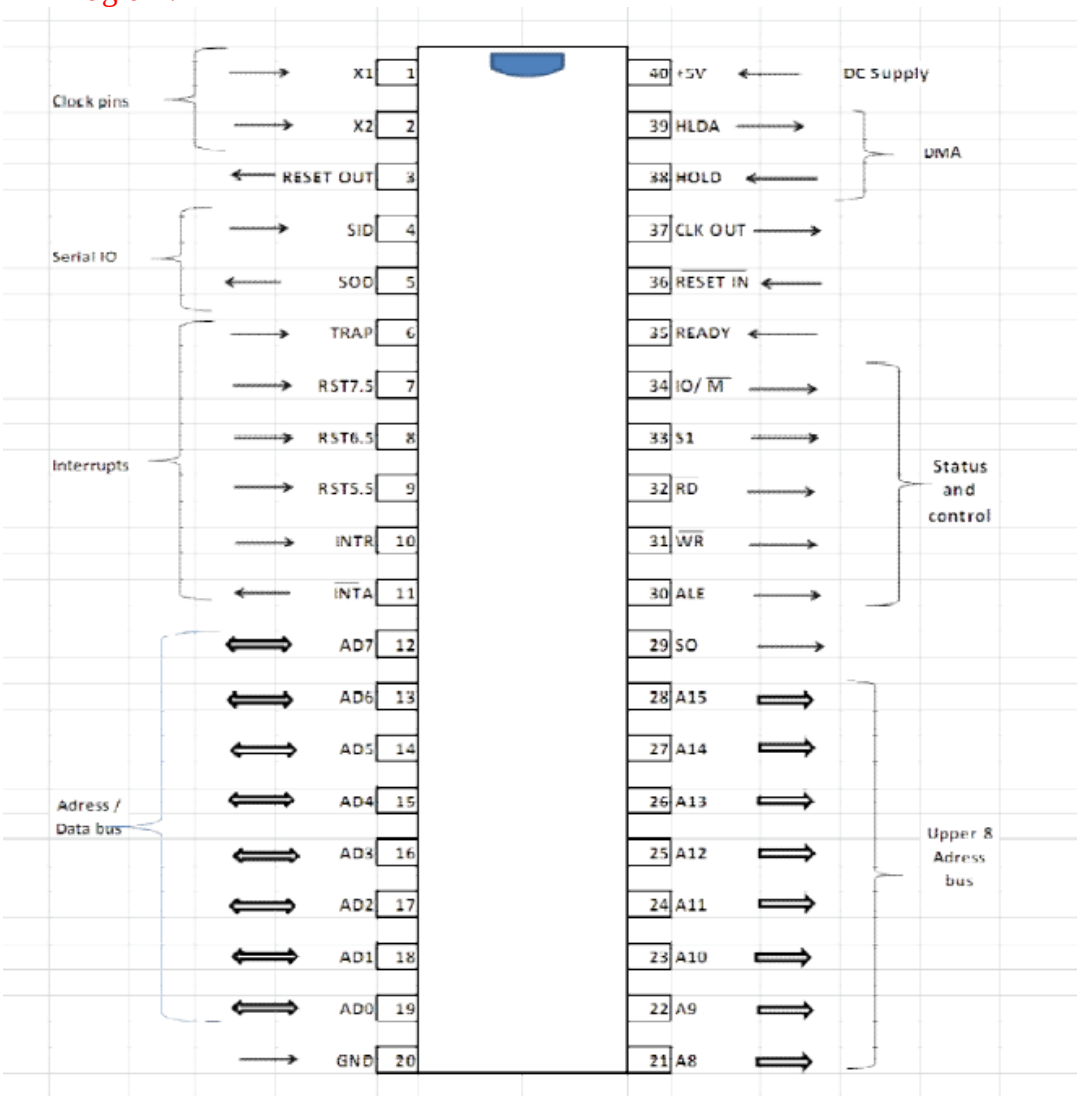
• Address Bus:

The address bus carries addresses and is one way bus from microprocessor to the memory or other devices. 8085 microprocessor contain 16-bit address bus and are generally identified as A0 - A15. The higher order address lines (A8 – A15) are unidirectional and the lower order lines (A0 – A7) are multiplexed (time-shared) with the eight data bits (D0 – D7) and hence, they are bidirectional.

• Control Bus:

The control bus is a bidirectional bus that is used to carry control signals between the microprocessor and other components such as memory and I/O devices. It is used to transmit commands to the memory or I/O devices

Pin Diagram:



X1 And X2 Pin:

It is also called Oscillator Pins to which crystal of max 6.14 Mhz should be connected so that 8085 can generate clock signals internally.The internal Clock pulses will be ½ times crystal value i.e, 3.07 Mhz.

RESET IN and RESET OUT

RESET IN is used to reset the microprocessor by making the pin Low. When the signal on this pin is low for at least 3 clock cycles, it forces the microprocessor to reset . Thereby ,

1. Clear the PC and IR.
2. Disable all the interrupts (except TRAP) and the SOD pin.
3. HIGH output pulse to RESET OUT pin.

Reset OUT is used to reset the external peripheral devices and ICs on the circuit. It is active high signal. The output on this pin goes high whenever RESET IN pin is made low .

SID and SOD:

SID (Serial Input Data): It receives 1-bit from external device and Stores the bit at the MSB of the Accumulator. RIM (Read Interrupt Mask) instruction is used to transfer the bit from SID MS Bit of Acc.

SOD (Serial Output Data): It transmits MSB of Accumulator through this pin. SIM (Set Interrupt Mask) instruction is used to transfer the MS bit of Acc through SOD.

Interrupt Pins:

Interrupt: (INTR,RST5.5,RST6.6,RST7.5 and TRAP pins)

It allows external devices to i*nterrupt* the normal program execution of the microprocessor. When microprocessor receives interrupt signal, it temporarily stops current program and starts executing new program indicated by the interrupt signal.

Interrupt signals are generated by external peripheral devices like keyboard , sensors, printers etc. After execution of the new program, microprocessor returns back to the previous program.

The 5 Hardware Interrupt Pins are TRAP , RST 7.5 , RST6.5 , RST 5.5 and INTR. Interrupts can be classified as,

1. Maskable and Non-Maskable
2. Vectored and Non-Vectored
3. Priority Based Interrupts

Maskable and Non-Maskable

Maskable interrupts are those interrupts which can be *enabled* or *disabled*. Enabling and Disabling can be done by software instructions like EI , DI and SIM. The interrupt pins RST7.5, RST6.5 ,RST5.5 and INTR are Maskable.

The interrupts which cannot be disa*bled* are called non-maskable interrupts. These interrupts can never be disabled by any software instruction. TRAP is a non-maskable interrupt. Such pins are normally used for emergency cases like fire alarming , fire extinguisher system ,intruder detector etc.

Vectored and Non-Vectored

- Vectored interrupts which have particular memory location where program control is transferred when interrupt occur. Each vectored interrupt points to the particular location in memory. RST 7.5 , RST 6.5 ,RST 5.5, TRAP are vectored Interrupts.
- Non-Vectored interrupts don't have fixed memory location for transfer of program control. The address of the memory location is given by interrupting device to the processor along with the interrupt. INTR is a non-vectored interrupt.

Priority Based Interrupts

- When there is a simultaneous interrupt request at two or more interrupt pins then the microprocessor will execute program of that pin that has higher priority. To avoid confusion in such cases all microprocessor assigns priority level to each interrupt pins. Priority is considered by microprocessor only when there are simultaneous requests.

Address and Data Pins

Address Bus pins A_{15}–A_8 and AD_7–AD_0 : The address bus are used to send address for memory or IO device. It selects one of the many locations in memory or a particular IO port. 8085 has 16-bit bus.

Data Bus pins AD_7–AD_0 : It is used to transfer data between microprocessor and memory /IO Device. 8085 Data bus is of 8-bit.

ALE Pin:

It indicates whether the bus has address or data .Since the bus AD_7 – AD_0 is used for Adress as well as data therefore it is known as Multiplexed bus. Due to multiplexing 8085 has less no of pins.

Status S1 ,So

S_0 and S_1 are called Status Pins.They indicate the status of current operation which is in progress by 8085.The 4 status indicated by 8085 are

S_0	S_1	Operation
0	0	Halt
0	1	Write
1	0	Read
1	1	Opcode Fetch

IO/M:
This pin indicates whether I/O or memory operation is being performed by microprocessor.

- If IO/M = 1 then IO being performed. Otherwise memory operation.

RDPin
It is a control signal used to perform Read operation from memory or from Input device. It is active low signal. A low signal indicates that data on the data bus must be placed by the selected memory location or input device.

WR
It is a control signal used to perform Write operation into memory location or to output device. It is also active low signal. A low signal indicates that data on the data bus must be written into selected memory location or to output device.

READY Pin

This pin is used to synchronize slower peripheral devices with high speed of microprocessor. A low pulse in T2 causes the microprocessor to enter into *wait state*. The microprocessor remains in wait state until the input at this pin goes high.

HOLD

HOLD pin is used to request the microprocessor for DMA transfer. A high signal on this pin is a request to microprocessor, by external device, to relinquish the hold on buses. The request is sent by external device through DMA controller

HLDA Pin

The HLDA signal is send to DMA Controller as acknowledgement to DMA controller to indicate that microprocessor has relinquished the system bus. After data transfer When HOLD is made low by DMA controller, HLDA is also made low by 8085 so that the microprocessor takes control of the buses again.

V_{SS} and V_{CC} Pin

+5V DC power supply is connected to V_{CC} to bias internal circuit. Ground signal is connected to V_{SS}.

The flow of an Instruction Cycle in 8085 Architecture :

1. Execution starts with Program Counter. It starts program execution with the next address field. it fetches an instruction from the memory location pointed by Program Counter.
2. For address fetching from the memory, multiplexed address/data bus acts as an address bus and after fetching instruction this address bus will now acts as a data bus and extract data from the specified memory location and send this data on an 8-bit internal bus. For multiplexed address/data bus Address Latch Enable(ALE) Pin is used. If ALE = 1 (Multiplexed bus is Address Bus otherwise it acts as Data Bus).
3. After data fetching data will go into the Instruction Register it will store data fetched from memory and now data is ready for decoding so for this Instruction decoder register is used.
4. After that timing and control signal circuit comes into the picture. *It sends control signals all over the microprocessor to tell the microprocessor whether the given instruction is for READ/WRITE and whether it is for MEMORY/I-O Device activity.*
5. Hence according to timing and control signal pins, logical and arithmetic operations are performed and according to that data fetching from the different registers is done by a microprocessor, and mathematical operation is carried out by ALU. And according to operations Flag register changes dynamically.
6. With the help of Serial I/O data pin(SID or SOD Pins) we can send or receive input/output to external devices .in this way execution cycle is carried out.
7. While execution is going on if there is any interrupt detected then it will stop execution of the current process and Invoke Interrupt Service Routine (ISR) Function. Which will stop the current execution and do execution of the current occurred interrupt after that normal execution will be performed.

Explain the concept of "T-state" in the context of the 8085 microprocessor./ What is meant by "instruction delay" in the 8085 microprocessor, and how does it work?

In the 8085 microprocessor, instruction delay refers to the time taken for an instruction to execute, measured in clock cycles or T-states. Different instructions require different amounts of time to execute, depending on their complexity and the number of operations involved. Instruction delay is essential for applications that require precise timing, such as generating delays in embedded systems, where specific timing intervals are needed.

A T-state (or clock period) in the 8085 microprocessor is the smallest unit of time in which the processor performs a single internal operation. Each T-state corresponds to one clock cycle of the system clock, and it determines the time taken to execute each part of an instruction.

MVI C, FFH	F,R	7 T-State
LOOP: DCR C	F	4 T-State
JNZ LOOP	F, R, R	7/10 T-State

The 8085 microprocessor breaks down each instruction into machine cycles, such as fetch, memory read, and memory write. Each machine cycle is further divided into multiple T-states. For example, the fetch

cycle in the 8085 might require 4 T-states. The number of T-states an instruction requires can be used to calculate its total execution time based on the system clock period TTT.

T-states are essential for timing analysis and performance optimization in the 8085. They are especially important in applications requiring precise timing, such as generating delays or controlling real-time systems (e.g., traffic lights).

- A time delay loop in the 8085 microprocessor is a program structure that creates a controlled delay by executing a set of instructions repeatedly for a specific amount of time.

Define the Stack Pointer (SP) and explain its role in stack memory management in the 8085 microprocessor.

The Stack Pointer (SP) is a 16-bit register in the 8085 microprocessor that holds the address of the top of the stack in memory. The stack is a Last-In-First-Out (LIFO) structure used for temporarily storing data. The SP helps manage the stack by pointing to the memory location where the next data item will be pushed (stored) or popped (retrieved).

In the 8085, the stack typically grows backwards (from higher memory addresses to lower ones). To use the stack, the programmer sets the initial address of the bottom of the stack using the LXI SP instruction. For example, LXI SP, FFFFH sets the stack pointer to the memory location FFFFH, often used as the end of memory in the 8085, so that the stack has space to grow downward.

Describe how information is saved on and retrieved from the stack in the 8085 microprocessor.

In the 8085 microprocessor, information is saved on the stack using the PUSH instruction and retrieved from the stack using the POP instruction. Both instructions operate on register pairs (e.g., BC, DE, HL).

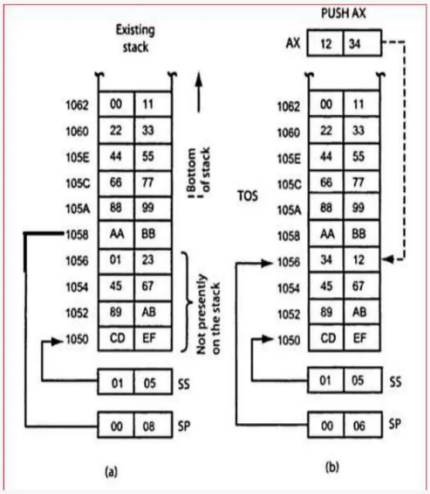
STACK STRUCTURE OF 8086: PUSH OPERATION

AX initially contains the number 1234H.

Execution of push instruction causes the stack pointer to be decremented by two.

Therefore the next stack access is to the location corresponding to address 1056H. This location is where the value in AX is pushed.

The MSB of AX (ie,12H) resides in memory address 1057H and LSB of AX (ie, 34H) is held in memory address 1056H

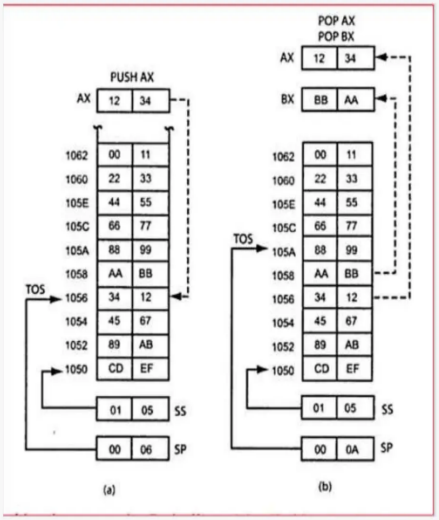


STACK STRUCTURE OF 8086: POP OPERATION

The execution of the first instruction POP AX, causes the 8086 to read the value from the Top of the stack and put it in to AX register as 1234H

SP is incremented to give 0008H and the other read operation POP BX, causes the value BBAAH to be loaded into the BX register.

SP is incremented once more and now equals 000AH. Therefore , the new top of stack is at address 105AH



Explain how the stack is used in CALL and RET instructions.

The CALL and RET instructions use the stack to handle subroutine calls in the 8085 microprocessor.

- CALL: When a subroutine is called, the 8085 pushes the current Program Counter (PC) address (the address of the instruction immediately after CALL) onto the stack. This allows the program to return to the exact location in the main program after the subroutine execution is completed.

- RET: At the end of a subroutine, the RET instruction pops the return address (stored during the CALL) from the stack back into the Program Counter (PC), allowing the program to continue execution from the instruction after the original CALL.

What is the Program Status Word (PSW) in the 8085 microprocessor?

The Program Status Word (PSW) in the 8085 microprocessor is an 8-bit register that combines the Accumulator (A) and the Flag Register (F). The PSW stores both the current result of an operation (in the Accumulator) and the status of certain conditions in the form of flags (e.g., zero, carry, sign, parity, auxiliary carry) in the Flag Register. This register pair (PSW) can be pushed onto the stack with PUSH PSW and retrieved with POP PSW, allowing temporary storage and restoration of the processor's state during subroutine calls or interrupt handling.

What is a subroutine, and why is it used in programming with the 8085 microprocessor?

A subroutine is a set of instructions designed to perform a specific task that can be called and reused multiple times within a program. Instead of writing the same set of instructions repeatedly, the programmer groups these instructions into a subroutine, which can then be called from different points in the program. This approach saves memory space and simplifies code management.

In the 8085 microprocessor:

- The CALL instruction is used to transfer control to the subroutine.
- The RET instruction is used to return control back to the point in the program where the subroutine was called.

Explain "call by reference" in the context of subroutines in the 8085 microprocessor.

In the 8085 microprocessor, call by reference refers to the process of allowing a subroutine to directly modify the data stored in registers or memory, so that any changes made in the subroutine will affect the calling program. Since the data in registers is manipulated directly, any modifications within the subroutine persist when control returns to the main program.

For instance, if a subroutine is designed to manipulate data in the B register, the main program calling this subroutine will see the modified value of B after the subroutine completes, because B was directly altered within the subroutine.

To prevent unintended changes to important registers, it is common practice to save the contents of necessary registers on the stack before the subroutine runs and restore them afterward. This approach uses the PUSH and POP instructions, ensuring that temporary changes in the subroutine do not permanently alter data needed by the calling program.

Flowchart to multiply two number