Aditya Dhiman

# COMPUTER SYSTEM ARCIETECTURE QUESTION BANK SOLUTION

## UNIT 1

### Very Short Answers (2 Marks Each)

**Q1: Illustrate the fundamental functional units of a computer system. CO1**

A computer system primarily consists of the following functional units:

1. **Central Processing Unit (CPU)**: Executes instructions and processes data.
    - **Arithmetic Logic Unit (ALU)**: Performs arithmetic and logical operations.
    - **Control Unit (CU)**: Directs the operations of the processor.
2. **Memory Unit**: Stores data and instructions.
3. **Input/Output (I/O) Units**: Manages communication between the computer and external devices.
4. **Bus**: Transfers data between the components of the computer.

**Q2: Tell the concept of a flip flop. CO1**

A flip-flop is a digital memory circuit capable of storing one bit of data. It is a bistable multivibrator, meaning it has two stable states (0 and 1). Flip-flops are fundamental building blocks for sequential logic circuits, such as registers and counters.

**Q3: Define a sequential circuit and provide an example. CO1**

A sequential circuit is a type of digital circuit whose output depends not only on the current inputs but also on the history of inputs. It has memory elements to store past inputs. Example: A simple digital clock is a sequential circuit.

**Q4: What is a combinational circuit? CO1**

A combinational circuit is a type of digital circuit whose output is solely determined by the current inputs, with no memory of past inputs. Examples include adders, multiplexers, and decoders.

**Q5: Provide the logical expressions for the sum and carry outputs of a full adder. CO1**

- **Sum (S)**: S=A⊕B⊕CinS = A \oplus B \oplus C_{in}S=A⊕B⊕Cin

- **Carry (C_{out})**: Cout=(A·B)+(Cin·(A⨁B))C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))Cout=(A·B)+(Cin·(A⨁B))

**Q6: How are logic circuits in a digital system classified? CO1**

Logic circuits in a digital system are classified into:

1. **Combinational Circuits**: Output depends only on the current input values (e.g., adders, multiplexers).
2. **Sequential Circuits**: Output depends on both current inputs and past inputs (e.g., flip-flops, counters).

**Q7: Define Multiplexer. CO1**

A multiplexer (MUX) is a combinational circuit that selects one input from multiple inputs and forwards it to a single output line. It acts as a data selector.

**Q8: What is meant by the term "level triggered"? CO1**

"Level triggered" refers to a type of circuit activation that occurs when the input signal is at a specific level (high or low) and remains responsive as long as the signal stays at that level.

**Q9: List out the applications of Multiplexers. CO1**

Applications of multiplexers include:

1. Data routing.
2. Signal multiplexing.
3. Switching networks.
4. Digital communication systems.

**Q10: Distinguish between D Flip flop & T flip flop. CO1**

- **D Flip-Flop**: Captures the value of the D input at a specific clock edge and stores it.
- **T Flip-Flop**: Toggles its output on each clock pulse if T input is high.

**Q11: Give the reason for D latch is called transparent latch? CO1**

A D latch is called a transparent latch because its output follows the input whenever the enable signal is active. This transparency ceases when the enable signal is inactive, capturing and holding the input value.

**Q12: Write minimum two advantages of counters. CO1**

1. **Synchronization**: Counters help in coordinating various operations in digital circuits.
2. **Timekeeping**: Used in clocks and timers to keep track of time or events.

### Q13: What is meant by the term “edge triggered”? CO1

"Edge triggered" refers to a type of circuit that responds to changes in the input signal at the transition (rising or falling edge) rather than the signal's level.

### Q14: Define demultiplexer. CO1

A demultiplexer (DEMUX) is a digital circuit that takes a single input and distributes it over several outputs. It performs the inverse function of a multiplexer.

### Q15: What is a truncated counter? CO1

A truncated counter is a type of counter that does not go through all possible states of its flip-flops. Instead, it resets to zero after reaching a specific count, less than its maximum possible count.

### Q16: Define the concept of BUS. CO1

A bus is a communication system that transfers data between components inside a computer or between computers. It consists of a set of parallel wires and protocols for data transmission.

### Q17: How does a decoder contribute to the overall functionality of a microcontroller or microprocessor system? CO1

A decoder in a microcontroller or microprocessor system translates encoded data or addresses into a specified output format, enabling the selection of specific memory locations or I/O devices.

### Q18: What is a Decoder? CO1

A decoder is a digital circuit that converts encoded binary input data into a specific output pattern. It is used in memory address decoding and data multiplexing.

### Q19: Difference between digital and analog computer. CO1

- **Digital Computer**: Processes discrete binary data (0s and 1s), used for general-purpose computing.
- **Analog Computer**: Processes continuous data, used for specific scientific and engineering applications.

### Q20: How is volatile memory different from non-volatile memory, and what are examples of each? CO1

- **Volatile Memory**: Requires power to maintain stored data (e.g., RAM).
- **Non-Volatile Memory**: Retains data even when power is off (e.g., ROM, SSDs).

## Medium Answers (5 Marks Each)

### Q21: Elaborate on memory organization within computers. CO1

**Memory Organization** in computers is structured hierarchically for efficiency:

1. **Registers**:
    - **Location**: Within the CPU.
    - **Function**: Store data temporarily for immediate processing.
    - **Speed**: Fastest memory type.
2. **Cache Memory**:
    - **Levels**: L1 (closest to CPU, smallest, fastest), L2, L3 (larger, slower).
    - **Function**: Stores frequently accessed data to reduce access time.
3. **Main Memory (RAM)**:
    - **Type**: Volatile memory.
    - **Function**: Holds data and instructions for currently running programs.
4. **Secondary Storage**:
    - **Types**: Hard drives, SSDs.
    - **Function**: Non-volatile storage for long-term data retention.

**Addressing**:

- **Physical Addressing**: Direct access to memory locations.
- **Logical Addressing**: CPU-generated addresses mapped to physical addresses.

### Q22: Define micro-operations and enumerate their various types. CO1

**Micro-operations** are the simplest operations performed by a CPU on data stored in registers. Types include:

1. **Arithmetic Operations**:

  - **Addition**: E.g., ADD R1, R2.
  - **Subtraction**: E.g., SUB R1, R2.
2. **Logic Operations**:
   - **AND**: E.g., AND R1, R2.
   - **OR**: E.g., OR R1, R2.
3. **Shift Operations**:
   - **Logical Shift Left/Right**: E.g., SHL R1, SHR R1.
   - **Arithmetic Shift**: Retains sign bit during shifts.
4. **Data Transfer Operations**:
   - **Move**: E.g., MOV R1, R2.
   - **Load/Store**: E.g., LOAD R1, STORE R1.

**Q23: Provide a detailed explanation of bus architecture. CO1**

**Bus Architecture** is the communication system within a computer:

1. **Data Bus**:
   - **Function**: Transfers actual data between CPU, memory, and I/O devices.
   - **Width**: Affects data transfer speed.
2. **Address Bus**:
   - **Function**: Carries memory addresses from the CPU to other components.
   - **Width**: Determines the maximum addressable memory.
3. **Control Bus**:
   - **Function**: Transmits control signals for managing data transfers and operations.
   - **Signals**: Includes read/write signals, clock signals, etc.

Buses ensure efficient and synchronized communication, enabling data transfer and control signal transmission between different parts of the system.

**Q24: Explain the operation of MOD-4 counter. CO1**

A **MOD-4 Counter** counts from 0 to 3 (four states) and then resets to 0:

1. **Components**:
    - **Flip-Flops**: Typically two, arranged to count in binary (00, 01, 10, 11).
2. **Operation**:
    - **State Sequence**: 00 → 01 → 10 → 11 → 00.
    - **Clock Pulses**: Flip-flops toggle states based on clock input.
3. **Implementation**:
    - **Logic Gates**: May be used to manage state transitions.
4. **Applications**:
    - **Digital Clocks**: For counting seconds, minutes, etc.
    - **Frequency Division**: Used in signal processing.

**Q25: Discuss the role of cache memory in computer systems. CO1**

**Cache Memory** plays a pivotal role in enhancing computer performance:

1. **Speed Enhancement**:
    - **Proximity to CPU**: Located closer than main memory, reducing access time.
    - **Levels**: L1 (fastest, smallest), L2, and L3 (larger, slower).
2. **Reducing Latency**:
    - **Frequent Access**: Stores frequently used data and instructions.
    - **Hit Rate**: High hit rate improves processing speed.
3. **Performance Improvement**:
    - **CPU Efficiency**: Reduces time CPU spends waiting for data from main memory.
    - **Bottleneck Reduction**: Minimizes delays caused by slower memory access.

**Q26: Outline the advantages of counters in computer architecture. CO1**

**Counters** offer several advantages in digital systems:

1. **Timing Control**:
    - **Synchronization**: Helps in coordinating operations within digital circuits.
    - **Pulse Counting**: Used in timers and clocks.

2. **Event Counting**:
   - **Track Events**: Counts occurrences of specific events, like clock pulses.
3. **Sequence Generation**:
   - **Finite State Machines**: Implements state sequences in control systems.
4. **Frequency Division**:
   - **Signal Processing**: Used in dividing frequencies for digital signals.

**Q27: Explain the process of bus and memory transfer comprehensively. CO1**

**Bus and Memory Transfer** involves:

1. **Addressing**:
   - **CPU Sends Address**: Via the address bus to specify memory location.
2. **Control Signals**:
   - **Read/Write Operations**: Control bus signals initiate data read or write.
3. **Data Transfer**:
   - **Data Bus**: Transfers data between CPU and memory.
   - **Data Read**: Data moved from memory to CPU.
   - **Data Write**: Data moved from CPU to memory.
4. **Coordination**:
   - **Synchronization**: Ensures error-free and timely data exchange.

**Q28: Details about various micro-operations in computers. CO1**

**Micro-operations** are fundamental operations performed by the CPU:

1. **Arithmetic Operations**:
   - **Addition**: E.g., ADD R1, R2.
   - **Subtraction**: E.g., SUB R1, R2.
2. **Logic Operations**:
   - **AND**: E.g., AND R1, R2.
   - **OR**: E.g., OR R1, R2.

3. **Shift Operations**:
   - **Logical Shifts**: Shift bits left/right (SHL, SHR).
   - **Arithmetic Shifts**: Preserve sign bit during shifts.
4. **Data Transfer Operations**:
   - **Move**: E.g., MOV R1, R2.
   - **Load/Store**: E.g., LOAD R1, STORE R1.

**Q29: Illustrate the connection between the processor and memory, highlighting the functions of each component in this connection. CO1**

**Processor and Memory Connection**:

1. **Address Bus**:
   - **Function**: CPU sends memory addresses to access specific locations.
2. **Data Bus**:
   - **Function**: Transfers data between CPU and memory.
3. **Control Bus**:
   - **Function**: Manages read/write operations through control signals.

**Components**:

- **CPU**:
  - **ALU**: Performs arithmetic and logical operations.
  - **Control Unit**: Directs operations of the CPU.
- **Memory**:
  - **RAM**: Stores data and instructions for current operations.
  - **Cache**: Provides faster data access to the CPU.

**Q30: Explain what an instruction signifies in computing. CO1**

An **Instruction** in computing is a command for the CPU to perform a specific operation:

1. **Opcode**:
   - **Definition**: Specifies the operation to be performed (e.g., ADD, MOV).

2. **Operands**:
    - **Definition**: Data or memory addresses involved in the operation.
3. **Execution Cycle**:
    - **Fetch**: CPU fetches instruction from memory.
    - **Decode**: Control unit interprets the instruction.
    - **Execute**: ALU or other CPU units perform the operation.

## Detailed Answers (10 Marks Each)

### Q31: Elucidate the main components of a computer using a block diagram. CO1

**Main Components of a Computer:**

1. **Central Processing Unit (CPU)**:
    - **Control Unit (CU)**: Coordinates activities of other components.
    - **Arithmetic Logic Unit (ALU)**: Performs arithmetic and logic operations.
2. **Memory**:
    - **Primary Memory**: Stores data and instructions temporarily.
        - **RAM**: Volatile memory for active programs.
        - **ROM**: Non-volatile memory for firmware and boot-up instructions.
    - **Secondary Storage**: Stores data permanently.
        - **HDD, SSD**: Examples.
3. **Input/Output (I/O) Devices**:
    - **Keyboard, Mouse, Monitor, Printer**: Examples.
4. **System Bus**:
    - **Data Bus**: Transfers data between CPU, memory, and I/O devices.
    - **Address Bus**: Transmits memory addresses.
    - **Control Bus**: Manages data transfer and control signals.
5. **Motherboard**:

  - **Connects Components**: Provides physical connections between CPU, memory, and I/O devices.
  - **Chipsets**: Control data flow between CPU, memory, and peripherals.

6. **Power Supply**:
   - **Converts AC to DC**: Supplies power to all components.

**Block Diagram**: [Include a visual representation of the block diagram showing the interconnections between CPU, memory, I/O devices, and buses.]

**Q32: Illustrate and elucidate the functioning of a binary adder-subtractor circuit. CO1**

**Binary Adder-Subtractor Circuit** combines addition and subtraction functionality:

- **Components**:
  - **Full Adder**: Adds binary numbers.
  - **Subtractor Circuit**: Performs binary subtraction.
- **Operation**:
  - **Addition**: Performs bitwise addition of two binary numbers.
  - **Subtraction**: Utilizes 2's complement to perform subtraction.
- **Implementation**:
  - **Combination**: Uses a multiplexer to select between addition and subtraction modes.
- **Applications**:
  - **Arithmetic Operations**: Addition and subtraction of binary numbers.
  - **Data Processing**: Used in digital calculators, computers, etc.

[Include a schematic diagram of the binary adder-subtractor circuit and explain its operation step by step.]

**Q33: Provide the diagram for 4*1 multiplexer and tell the importance used in it. CO1**

**4x1 Multiplexer Diagram**: [Include a visual representation of the 4x1 multiplexer circuit.]

**Importance**:

- **Data Selection**: Allows selection of one input from four data inputs based on the select lines.

- **Resource Optimization**: Saves hardware by reducing the number of gates required for data selection.
- **Versatility**: Can be used in various applications like data routing, signal selection, etc.

[Explain the significance of each input/output and how the selection lines control data routing.]

**Q34: Expound on Arithmetic, Logic, and Shift micro-operations. CO1**

**Micro-operations** in CPU:

1. **Arithmetic Operations**:
    - **Addition**: Performs binary addition of operands.
    - **Subtraction**: Utilizes 2's complement to perform subtraction.
2. **Logic Operations**:
    - **AND, OR, XOR**: Perform bitwise logical operations on operands.
    - **NOT**: Performs bitwise negation.
3. **Shift Operations**:
    - **Left Shift**: Moves bits leftward, filling vacated bits with zeroes.
    - **Right Shift**: Moves bits rightward, filling vacated bits based on operation type (arithmetic or logical).

[Provide examples and explain the significance of each micro-operation in data processing.]

**Q35: Explain the execution of a complete instruction given to the computer CO1**

**Execution of an Instruction**:

1. **Fetch**: CPU retrieves instruction from memory based on the program counter (PC).
2. **Decode**: Control unit interprets the opcode and determines the instruction type.
3. **Execute**:
    - **Data Fetch**: Fetches operands from memory or registers.
    - **Operation**: Performs specified operation (e.g., arithmetic, logic) using ALU.
    - **Write Back**: Stores result in memory or registers based on instruction.

[Describe each step in detail, including relevant control signals and data flow.]

## Q36: Elaborate on the different buses in computer architecture with its advantages and disadvantages. CO1

**Types of Buses**:

1. **Data Bus**:
    - **Advantages**: High-speed data transfer.
    - **Disadvantages**: Limited bandwidth, susceptible to data corruption.
2. **Address Bus**:
    - **Advantages**: Provides direct memory addressing.
    - **Disadvantages**: Limited address range, slower than data bus.
3. **Control Bus**:
    - **Advantages**: Coordinates data transfer and operations.
    - **Disadvantages**: May introduce delays, complexity.

[Discuss the role of each bus, their advantages, disadvantages, and impact on system performance.]

## Q37: Discuss Memory hierarchy and memory types in detail. CO1

**Memory Hierarchy**:

1. **Registers**: Fastest and smallest, located within CPU.
2. **Cache Memory**: Intermediate between registers and main memory, used for faster data access.
3. **Main Memory (RAM)**: Volatile memory for active programs.
4. **Secondary Storage**: Non-volatile memory for long-term storage.

**Memory Types**:

- **RAM (Random Access Memory)**:
    - **Types**: SRAM (Static RAM), DRAM (Dynamic RAM).
    - **Volatile**: Requires power to retain data.
- **ROM (Read-Only Memory)**:
    - **Types**: PROM, EPROM, EEPROM, Flash.

  - **Non-volatile**: Retains data even without power.

[Explain the characteristics, advantages, and applications of each memory type in the hierarchy.]

**Q38: Show that a JK flip-flop can be converted to a D flip flop with an inverter between J and K. CO1**

**Conversion Process**:

1. **JK Flip-Flop**:
   - **Inputs**: J, K.
   - **Operation**: Toggles output based on inputs.
2. **D Flip-Flop**:
   - **Inputs**: D (data), CLK (clock).
   - **Operation**: Captures data at clock edge.

**Conversion Method**:

- Connect both J and K inputs of the JK flip-flop to D input via an inverter.
- When the clock signal arrives, the JK flip-flop behaves like a D flip-flop.

[Include a diagram illustrating the connection and explain how the conversion works.]

**Q39: Explain different logic gates with their truth tables and diagram. CO1**

**Logic Gates**:

1. **AND Gate**:
   - **Operation**: Output is high (1) only when all inputs are high (1).
   - **Truth Table**:

| A | B | Out |
|---|---|-----|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

2. **OR Gate**, **NOT Gate**, **NAND Gate**, **NOR Gate**, **XOR Gate**, **XNOR Gate**:
    - Provide similar explanations, truth tables, and diagrams.

[Include diagrams, truth tables, and explanations for each type of logic gate.]

**Q40: Execute the encoder and decoder function with suitable diagram and truth table. CO1**

**Encoder and Decoder Functions**:

- **Encoder**: Converts multiple input lines into a smaller set of output lines.
- **Decoder**: Converts binary information from input lines into a specific output configuration.

**Encoder Truth Table**:

| Input (A, B) | Output (Y) |
|-------------|------------|
| 0 0 | 00 |
| 0 1 | 01 |
| 1 0 | 10 |
| 1 1 | 11 |

**Decoder Truth Table**:

yaml

Copy code

| Input (A, B) | Output (Y0, Y1, Y2, Y3) |
|-------------|------------------------|
| 0 0 | 1000 |
| 0 1 | 0100 |
| 1 0 | 0010 |
| 1 1 | 0001 |

[Include diagrams illustrating the encoder and decoder circuits along with their respective truth tables and explain their operation.]

These detailed responses should cover the requirements of each question comprehensively. Let me know if you need further clarification on any of the topics!

# UNIT 2

### Q37: What is a register, and what is its significance in computer architecture? CO2

A register is a high-speed storage unit within the CPU. It holds data temporarily for immediate processing. Registers are crucial in computer architecture for fast access to data, enhancing CPU performance.

### Q38: Write the concept of register transfer in computer architecture. CO2

Register transfer involves the movement of data between registers within a computer system. It's fundamental to instruction execution, enabling data manipulation and control flow within the CPU.

### Q39: Describe the functioning of a control unit. CO2

The control unit orchestrates the activities of other CPU components. It interprets program instructions, generates control signals, and coordinates data transfer between registers, ALU, and memory.

### Q40: Define Computer Architecture. CO2

Computer architecture refers to the design and organization of computer systems, including hardware components and their interconnections. It defines the structure, behavior, and operation of CPUs, memory, I/O devices, and system buses.

### Q41: What is the relation between memory size and their speeds? CO2

Generally, smaller memory sizes have faster access times compared to larger ones. However, technology advancements have improved both size and speed, aiming for a balance between larger capacities and faster access times in memory systems.

### Q42: What are the four types of computer architecture? CO2

The four types are: Von Neumann Architecture, Harvard Architecture, Modified Harvard Architecture, and Pipeline Architecture.

**Q43: What is the purpose of a clock signal in register transfer? CO2**

The clock signal synchronizes register transfers, ensuring data moves at precise intervals. It divides CPU operations into discrete steps, regulating data flow and enabling coordinated execution of instructions.

**Q44: Mention various types of computer memory. CO2**

Types include RAM (Random Access Memory), ROM (Read-Only Memory), Cache Memory, and Secondary Storage (e.g., HDDs, SSDs).

**Q45: Difference between MAR and PC. CO2**

MAR (Memory Address Register) holds the memory address being accessed, while PC (Program Counter) stores the address of the next instruction to be fetched.

**Q46: Why is Register Transfer Language (RTL) preferred for describing the internal organization of digital computers? CO2**

RTL offers a precise description of digital systems' internal structure, focusing on data transfers between registers. It's hardware-independent and facilitates design and analysis of digital systems.

**Q47: Explain the role of a clock signal in register transfer. CO2**

The clock signal synchronizes register operations, ensuring data moves in discrete steps. It controls the timing of register updates, coordinating CPU activities and maintaining system integrity.

**Q48: Tell the function of a data bus in register transfer. CO2**

The data bus transfers data between registers, memory, and other components. It carries binary data signals in parallel, facilitating high-speed data transfer and bidirectional communication.

**Q49: What is the role of an address bus in register transfer? CO2**

The address bus transmits memory addresses between the CPU and memory devices. It specifies the location in memory from which data should be fetched or to which data should be written during register transfer operations.

**Q50: Discuss the significance of a control bus in register transfer. CO2**

The control bus carries control signals that regulate data transfer and CPU operations. It coordinates activities within the CPU, ensuring proper execution of instructions and maintaining system integrity.

### Q51: Define RTL (Register Transfer Language) and its purpose. CO2

RTL describes digital systems' internal organization by specifying data transfers between registers. It's used for hardware design and analysis, offering a concise representation of digital circuits' behavior.

### Q52: List some common RTL operators. CO2

Common RTL operators include MOVE, ADD, SUBTRACT, AND, OR, SHIFT, and LOAD. They describe data transfer and manipulation operations between registers in digital systems.

## Level B (5 Marks Each)

### Q53: What are some common types of shift micro-operations, and how do they differ from one another? CO2

**Common Types of Shift Micro-operations**:

1. **Logical Shifts**:
    - **Left Shift**: Moves bits to the left, filling vacant bits with zeros.
    - **Right Shift**: Moves bits to the right, filling vacant bits based on shift type (logical or arithmetic).
2. **Arithmetic Shifts**:
    - **Left Arithmetic Shift**: Similar to logical left shift, but preserves sign bit.
    - **Right Arithmetic Shift**: Preserves sign bit and shifts bits to the right.

**Differences**:

- Logical shifts treat numbers as unsigned, shifting all bits including the sign bit.
- Arithmetic shifts consider numbers as signed, preserving the sign bit during shifting.

### Q54: Differentiate between bus transfer and memory transfer. CO2

**Bus Transfer**:

- **Definition**: Movement of data between CPU and peripherals via the system bus.

- **Purpose**: Facilitates communication between CPU, memory, and I/O devices.
- **Examples**: Data transfer between CPU and RAM, I/O devices.

**Memory Transfer**:

- **Definition**: Movement of data between CPU and memory (RAM).
- **Purpose**: Involves loading or storing data into/from memory for program execution.
- **Examples**: Fetching instructions from memory, storing variables into memory.

**Q55: Explain the rounding mechanism in the IEEE standard for floating-point numbers. CO2**

**Rounding Mechanism in IEEE Floating-Point Standard**:

- **Purpose**: Ensures accurate representation of real numbers within the limited precision of floating-point formats.
- **Steps**:
    1. Determine the rounding mode (e.g., round to nearest, round up, round down).
    2. Check the bits beyond the precision limit.
    3. Apply rounding based on the rounding mode and the value of the first discarded bit.

**Q56: Discuss the role of cache memory in computer systems. CO2**

**Role of Cache Memory**:

- **Purpose**: Cache memory serves as a high-speed buffer between the CPU and main memory (RAM), storing frequently accessed data and instructions.
- **Functionality**:
    - Reduces the average access time to data by providing faster access than main memory.
    - Improves overall system performance by minimizing CPU idle time waiting for data from main memory.
    - Enhances CPU efficiency by exploiting temporal and spatial locality in program execution.
- **Types**: Includes level 1 (L1), level 2 (L2), and level 3 (L3) caches, organized hierarchically based on proximity to the CPU and size.

## Q57: Describe the IEEE standard for floating-point numbers. CO2

**IEEE Standard for Floating-Point Numbers**:

- **Format**: Follows the IEEE 754 standard, defining formats for representing floating-point numbers in binary.
- **Components**: Consists of sign bit, exponent, and mantissa (fractional part).
- **Formats**: Includes single precision (32-bit) and double precision (64-bit) formats.
- **Representation**: Uses scientific notation to represent real numbers, allowing for a wide range of values with varying precision.
- **Special Values**: Includes representations for positive and negative infinity, NaN (Not a Number), and denormalized numbers.

## Q58: Provide a depiction of the register transfer mechanism for P: R2 R1 with relevant diagrams. CO2

**Register Transfer Mechanism**:

- **Description**: Represents the transfer of data from register R1 to register R2 using a temporary register P.
- **Diagram**:

css

Copy code

```
P <- R1

R2 <- P
```

- **Explanation**: Data from R1 is first transferred to temporary register P. Then, the content of P is transferred to register R2.

## Q59: Explain the process of storing a word into memory. CO2

**Storing a Word into Memory**:

1. **Address Calculation**: Calculate the memory address where the word will be stored.
2. **Data Transfer**:
    - Send the memory address to the memory unit (MAR).
    - Transfer the word to be stored to the data bus.

3. **Write Operation**:
    - Enable the write control signal on the control bus.
    - Transfer the word from the data bus to the specified memory address.
4. **Verification**:
    - Verify the successful storage of the word by reading it back from memory if necessary.

**Q60: Explain different types of Register Transfer Operations. CO2**

**Types of Register Transfer Operations**:

1. **Data Transfer Operations**: Involve moving data between registers or between registers and memory.
2. **Arithmetic Operations**: Perform arithmetic computations on data stored in registers.
3. **Logic Operations**: Execute logical operations (e.g., AND, OR, XOR) on data within registers.
4. **Control Operations**: Manage the flow of instructions and data within the CPU, coordinating execution.

**Q61: Why is RTL preferred for describing the internal organization of digital computers? CO2**

**Reasons for RTL Preference**:

- RTL provides a concise and hardware-independent representation of digital systems.
- It focuses on data transfers between registers, closely aligning with the CPU's hardware implementation.
- RTL facilitates design, analysis, and verification of digital circuits by abstracting complex hardware details.
- It enables efficient communication and documentation of digital system architectures.

**Q62: Define system bus? Explain the different buses in computer architecture. CO2**

**System Bus Definition**:

- The system bus is a communication pathway that connects various components within a computer system, facilitating data transfer and control signals exchange.

**Different Buses in Computer Architecture**:

1. **Data Bus**:
    - Transfers data between CPU, memory, and I/O devices.
    - Bidirectional communication channel.
2. **Address Bus**:
    - Transmits memory addresses from CPU to memory.
    - Unidirectional communication.
3. **Control Bus**:
    - Carries control signals to coordinate data transfer and operations.
    - Manages activities within the CPU, memory, and I/O devices.

**Level C (10 Marks Each)**

**Q63: Describe the different components of a floating-point number as defined by the IEEE standard. CO2**

**Components of a Floating-Point Number (IEEE Standard)**:

1. **Sign Bit**: Indicates the sign of the number (positive or negative).
2. **Exponent**: Represents the scale factor applied to the mantissa, determining the number's magnitude.
3. **Mantissa (Significand)**: Represents the significant digits of the number in normalized form.
4. **Base (Radix)**: Typically binary (base-2) or decimal (base-10) representation.
5. **Precision**: Determines the number of significant digits in the mantissa.
6. **Range**: Defines the minimum and maximum values representable by the floating-point format.

**Q64: Provide a detailed explanation of how the Arithmetic Logic Shift Unit operates. CO2**

**Arithmetic Logic Shift Unit Operation**:

- Performs logical and arithmetic shift operations on binary numbers.
- **Logical Shift**: Moves all bits left or right, filling vacant positions with zeros.
- **Arithmetic Shift**: Preserves the sign bit during right shifts, ensuring sign extension.

- **Operation**:
  1. Receives input from registers or memory.
  2. Executes shift operation based on control signals.
  3. Outputs shifted result to destination register or memory.

**Q65: Explain Booth's algorithm for multiplication. CO2**

**Booth's Algorithm**:

- A multiplication algorithm used to multiply signed binary numbers efficiently.
- **Operation**:
  1. Perform sign extension on both multiplicand and multiplier.
  2. Initialize the product register with the multiplicand and shift register with zeros.
  3. Repeat the following steps until the multiplier becomes zero:
     - If the last two bits of the multiplier are 01, subtract the multiplicand from the product.
     - If the last two bits are 10, add the multiplicand to the product.
  4. Right shift the product and the multiplier.
  5. Repeat steps 3-4 until all bits of the multiplier are processed.
- **Advantages**: Reduces the number of additions and subtractions compared to traditional multiplication methods.

**Q66: What is the IEEE standard for floating-point numbers? Discuss the significance of the exponent and mantissa in the IEEE standard for floating-point numbers, and how do they affect the precision and range of floating-point values? CO2**

**IEEE Standard for Floating-Point Numbers**:

- IEEE 754 standard defines formats for representing floating-point numbers in binary.
- **Significance**:
  - **Exponent**: Determines the scale factor applied to the mantissa, affecting the number's magnitude and range.
  - **Mantissa**: Represents the significant digits of the number, influencing precision and accuracy.

- **Precision and Range**:
    - Increasing the exponent range expands the range of representable values but reduces precision.
    - Increasing the mantissa size improves precision but reduces the range of representable values.
- **Trade-off**: Balancing exponent range and mantissa size optimizes precision and range for a given floating-point format.

**Q67: Difference between computer organization and architecture CO2**

**Computer Organization**:

- Focuses on how hardware components are interconnected and operate to execute instructions.
- Concerned with the design and implementation of CPU, memory, I/O systems, and system buses.
- Emphasizes the structural aspects of computer systems, including CPU organization, memory hierarchy, and instruction execution.

**Computer Architecture**:

- Focuses on the design principles and attributes that define a computer system's logical structure and behavior.
- Concerned with the instruction set architecture (ISA), addressing modes, and CPU design.
- Emphasizes the functional aspects of computer systems, including instruction set design, addressing modes, and CPU operation.

**Key Difference**: Computer organization deals with the physical aspects and interconnection of hardware components, while computer architecture focuses on the logical structure and behavior of a computer system.

**Q68: Booth's algorithms is different from other multiplication algorithms in terms of speed and efficiency? Compare. CO2**

**Booth's Algorithm**:

- **Speed**: Generally faster for large multiplicands due to reduced number of additions and subtractions.

- **Efficiency**: Efficient for both positive and negative multiplicands, minimizing hardware complexity.

**Other Multiplication Algorithms**:

- **Standard Multiplication**: Slower for large multiplicands due to the need for repeated additions.
- **Array Multiplier**: Faster for small multiplicands but requires more hardware resources and longer propagation delays.
- **Shift and Add Algorithm**: Slower due to the need for multiple shift and add operations.

**Comparison**:

- Booth's algorithm offers superior speed and efficiency for large multiplicands compared to traditional multiplication algorithms.
- It minimizes the number of arithmetic operations required, resulting in faster computation and reduced hardware complexity.

**Q69: In conforming to the IEEE standard mention any four situations under which a processor sets exception flag Give examples for the situations. CO2**

**Situations Leading to Exception Flags**:

1. **Overflow**: Occurs when the result of an arithmetic operation exceeds the range representable by the floating-point format.
    - Example: Adding two large numbers results in a value exceeding the maximum representable value.
2. **Underflow**: Occurs when the result of an arithmetic operation is too small to be represented accurately.
    - Example: Subtracting a small number from a large number results in a value close to zero.
3. **Invalid Operation**: Occurs when an arithmetic operation produces an undefined or unsupported result.
    - Example: Performing square root operation on a negative number.
4. **Division by Zero**: Occurs when attempting to divide a number by zero.
    - Example: Dividing a number by zero results in an undefined result.

## Q70: Discuss in detail about floating point addition and subtraction CO2

**Floating-Point Addition**:

- Align the exponents of the operands by shifting the mantissa of the operand with the smaller exponent.
- Add or subtract the aligned mantissas, considering the sign bit of the operands.
- Normalize the result if necessary and round it to the specified precision.

**Floating-Point Subtraction**:

- Align the exponents of the operands by shifting the mantissa of the operand with the smaller exponent.
- Add or subtract the aligned mantissas, considering the sign bit of the operands.
- Normalize the result if necessary and round it to the specified precision.

**Considerations**:

- Handle special cases such as zero, infinity, and NaN (Not a Number) appropriately.
- Apply rounding rules as per IEEE 754 standard to ensure accuracy and precision.

## Q71: Demonstrate the registers and its functions used in computer architecture. CO2

**Registers Used in Computer Architecture**:

1. **Program Counter (PC)**: Holds the address of the next instruction to be fetched from memory.
2. **Instruction Register (IR)**: Stores the currently fetched instruction.
3. **Memory Address Register (MAR)**: Holds the address of the memory location being accessed.
4. **Memory Buffer Register (MBR)**: Temporarily stores data fetched from or to be written into memory.
5. **General-Purpose Registers (GPR)**: Used for storing data temporarily during program execution.
6. **Status Register (Flag Register)**: Stores condition flags indicating the result of arithmetic and logic operations.

**Functions**:

- **PC**: Sequentially fetches instructions from memory for execution.
- **IR**: Holds the currently executing instruction, decoded by the CPU.
- **MAR and MBR**: Facilitate data transfer between CPU and memory during read and write operations.
- **GPR**: Temporarily stores operands, results, and intermediate values during program execution.
- **Status Register**: Stores flags such as zero, carry, and overflow, reflecting the outcome of arithmetic and logic operations.

**Q72: Tell the merits and demerits of having clock signals in register transfer CO2**

**Merits**:

- **Synchronization**: Ensures orderly and synchronized execution of operations within the CPU.
- **Control**: Divides operations into discrete steps, facilitating precise control over data movement and processing.
- **Timing**: Regulates the timing of register transfers, preventing data corruption and ensuring data integrity.
- **Performance Optimization**: Allows for pipelining and parallel execution of instructions, enhancing overall system performance.

**Demerits**:

- **Clock Skew**: Clock signals may experience propagation delays, leading to timing issues and potential data hazards.
- **Power Consumption**: Continuous clock signal generation consumes additional power, contributing to overall system power consumption.
- **Complexity**: Designing and implementing clock distribution networks adds complexity to the CPU architecture and layout.
- **Sensitivity to Frequency**: Clock signals are sensitive to frequency variations, requiring careful design considerations for high-speed operation.

**Q73: Illustrate the practical applications used by the help of Arithmetic logic unit CO2**

**Practical Applications of Arithmetic Logic Unit (ALU)**:

1. **Mathematical Computations**: Performs arithmetic operations such as addition, subtraction, multiplication, and division required in scientific and engineering calculations.
2. **Data Processing**: Executes logical operations (AND, OR, XOR, NOT) necessary for data manipulation and decision making in digital signal processing and data analysis.
3. **Control Unit Operations**: Assists the control unit in executing program control instructions, branching, and conditional jumps within the CPU.
4. **Graphics Processing**: Supports vector and matrix operations essential for rendering and processing graphics in gaming and multimedia applications.
5. **Cryptography**: Implements complex mathematical algorithms required for encryption and decryption processes in secure communication protocols and digital signatures.
6. **Artificial Intelligence**: Executes mathematical operations and logical comparisons in machine learning algorithms, neural networks, and pattern recognition tasks.

# UNIT 3

1. **Control Unit**:
   - The control unit manages the execution of instructions and coordinates the activities of other hardware components in a computer system.
   - Its primary function is to decode instructions fetched from memory and generate control signals to execute them.
2. **Direct Memory Access (DMA)**:
   - DMA is a feature that allows peripheral devices to transfer data directly to and from the memory without involving the CPU.
   - It improves overall system performance by offloading data transfer tasks from the CPU.
3. **Handshaking in Asynchronous Data Transfer**:
   - Handshaking refers to the exchange of control signals between the sender and receiver to synchronize data transmission.
   - It ensures that both devices are ready to send and receive data, preventing data loss or corruption.

4. **Communication of Control Unit with Other Components**:
   - The control unit communicates with other components through control signals sent via the control bus.
   - It coordinates the activities of the CPU, memory, and I/O devices by generating control signals based on the instructions being executed.
5. **Main Components of a Control Unit**:
   - Instruction Register (IR), Instruction Decoder, Control Logic, Timing and Control Signals Generator.
6. **Hardwired vs. Microprogrammed Control Unit**:
   - Hardwired Control Unit: Uses fixed logic circuits to execute instructions directly.
   - Microprogrammed Control Unit: Uses microinstructions stored in control memory to execute instructions, offering flexibility and easier modification.
7. **Role of Instruction Register (IR)**:
   - The IR holds the currently fetched instruction, which is decoded by the control unit to generate control signals for execution.
8. **Purpose of Control Bus**:
   - The control bus carries control signals between various components of the computer system, facilitating communication and coordination.
   - It enables the control unit to synchronize and manage the activities of the CPU, memory, and I/O devices.
9. **Interrupts in 8085 Microprocessors**:
   - Interrupts are signals generated by external devices to request the CPU's attention.
   - They temporarily suspend the CPU's current execution to handle urgent tasks, improving system responsiveness.
10. **Determining Next Instruction to be Executed**:
    - The control unit determines the next instruction based on the instruction pointer (such as the Program Counter) and the current execution state.

11. **Instruction Cycle**:
    - The instruction cycle, also known as the fetch-decode-execute cycle, is the basic operation performed by a CPU to execute instructions.
    - It involves fetching an instruction from memory, decoding it to determine the operation to be performed, and executing the operation.
12. **Four Stages of the Instruction Cycle**:
    - **Fetch**: The CPU retrieves the instruction from memory.
    - **Decode**: The instruction is decoded to determine the operation to be performed.
    - **Execute**: The decoded instruction is executed, which may involve accessing data from memory or performing arithmetic/logical operations.
    - **Write Back**: The results of the operation are written back to memory or registers if necessary.
13. **Asynchronous Data Transfer**:
    - Asynchronous data transfer is a method of data transfer where the sender and receiver operate independently without a common clock signal.
    - Data is transmitted in bursts or packets, with each packet containing synchronization information to ensure correct reception.
14. **Decode and Execution Stage in the Instruction Cycle**:
    - The decode stage deciphers the instruction fetched from memory to determine the operation to be performed and the operands involved.
    - The execution stage executes the decoded instruction, performing the specified operation such as arithmetic, logic, or data transfer.
15. **Input-Output Processor (IOP)**:
    - An Input-Output Processor (IOP) is a specialized processor dedicated to managing input and output operations in a computer system.
    - It offloads I/O tasks from the CPU, allowing it to focus on computational tasks, thereby improving system performance.
16. **Accumulator in 8085 Microprocessors**:

- The accumulator is a register in the 8085 microprocessor used to store intermediate results of arithmetic and logic operations.
- It is the primary register for performing arithmetic operations and holds the final result of such operations.

17. **Role of Timing and Control Unit**:
    - The timing and control unit generates and synchronizes the timing signals required for the operation of various components within the computer system.
    - It ensures that instructions are executed in the correct sequence and at the appropriate time, maintaining the system's overall integrity and functionality.
18. **Software Interrupt**:
    - A software interrupt is a signal generated by software to request the CPU to perform a specific function or service.
    - It is initiated by executing a special instruction (e.g., INT) in the program code to trigger a predefined interrupt service routine.
19. **Microcontroller**:
    - A microcontroller is a compact integrated circuit that combines a microprocessor core with memory, input/output peripherals, and other essential components.
    - It is commonly used in embedded systems for controlling various devices and performing specific tasks.
20. **Synchronous Data Transfer**:
    - Synchronous data transfer is a method of data transfer where the sender and receiver operate synchronously with a common clock signal.
    - Data is transmitted in a continuous stream, with each bit synchronized to the clock signal for accurate reception.
21. **Clock Signal and Timing Control**:
    - The clock signal is a periodic signal generated by the system clock that synchronizes the operations of various components within a computer system.
    - It dictates the timing of instruction execution, data transfer, and other system activities, ensuring proper coordination and synchronization.
22. **Multiple-Bus Organization**:

  - Multiple-bus organization improves the performance of a computer system by increasing the bandwidth and reducing contention for data transfer.
  - It involves segregating different types of data transfers (e.g., instruction fetch, data transfer) onto separate buses, allowing simultaneous transfer of multiple data streams.

## LEVEL B SHORT ANSWERS (5 MARKS)

23. **Steps in the Instruction Cycle of a Control Unit**:

- **Fetch**: The control unit retrieves the instruction from memory using the program counter (PC).
- **Decode**: The instruction is decoded to determine the operation to be performed and the operands involved.
- **Execute**: The decoded instruction is executed by the CPU, which may involve data manipulation or transfer.
- **Write Back**: If necessary, the results of the operation are written back to memory or registers.

24. **Purpose of the Program Counter (PC) in a Control Unit**:

- The program counter holds the memory address of the next instruction to be fetched and executed.
- It increments automatically after each instruction fetch to point to the next sequential instruction in memory.

25. **Role of the Control Bus in a Computer System**:

- The control bus carries control signals between various components of the computer system, such as the CPU, memory, and I/O devices.
- It differs from other buses (data bus and address bus) as it carries signals related to system control rather than data or memory addresses.

26. **Data Transfer in a Computer System**:

- Data transfer involves moving data between different components within a computer system.

- **Parallel Data Transfer**: Involves transferring multiple bits simultaneously over separate lines, offering high data transfer rates but requiring more hardware resources.
- **Serial Data Transfer**: Involves transferring one bit at a time over a single line, offering lower data transfer rates but requiring fewer hardware resources.

27. **Methods Involved in Asynchronous Data Transfer**:

- **Start-Stop Method**: Involves sending start and stop bits before and after each data byte to delineate the data frame.
- **Baud Rate Method**: Involves synchronizing data transmission using a predefined baud rate, with data bits sent continuously at regular intervals.

28. **Advantages of DMA**:

- **Improved Performance**: DMA allows for direct data transfer between peripherals and memory, reducing CPU involvement and improving overall system performance.
- **Efficiency**: DMA minimizes CPU overhead by offloading data transfer tasks, allowing the CPU to focus on other critical tasks.
- **Resource Utilization**: DMA optimizes system resources by efficiently managing data transfer between peripherals and memory.

29. **Difference between Synchronous and Asynchronous Data Transfer**:

- **Synchronous Transfer**: Data is transferred at a fixed rate synchronized by a common clock signal, ensuring data integrity and timing consistency.
- **Asynchronous Transfer**: Data is transferred without a common clock signal, relying on start and stop signals or predefined baud rates for synchronization, offering flexibility but potentially lower data transfer rates.

30. **Significance of Buffer in an I/O Output Interface**:

- A buffer temporarily stores data being transferred between the CPU and I/O devices, smoothing out any differences in data rates between the two.
- It prevents data loss or corruption by providing a temporary storage area for data during I/O operations, ensuring reliable and efficient data transfer.

## LONG ANSWERS 10 MARKS

31. **Differences Between Hardwired and Microprogrammed Control Units**:

**Hardwired Control Unit**:

- **Implementation**: Uses fixed logic circuits to control the CPU's operation.
- **Advantages**: Faster execution due to direct hardware implementation, simpler design, and lower cost.
- **Disadvantages**: Limited flexibility, difficult to modify or update, requires more hardware components.

**Microprogrammed Control Unit**:

- **Implementation**: Utilizes microinstructions stored in control memory to control the CPU's operation.
- **Advantages**: Greater flexibility, easier modification and updates, allows for complex instruction sets.
- **Disadvantages**: Slower execution compared to hardwired control units, requires additional control memory.

32. **Direct Memory Access (DMA) in a Computer System**:

- DMA allows peripherals to transfer data directly to and from memory without CPU intervention.
- The DMA controller manages data transfer requests from peripherals, coordinates with the CPU to grant access to memory, and controls data movement between memory and peripherals.
- DMA operations are initiated by the CPU, which configures the DMA controller and provides necessary parameters for data transfer.

33. **Interrupt**:

- An interrupt is a signal generated by hardware or software to request the CPU's attention.
- **Types**: Hardware interrupts (e.g., I/O interrupts, timer interrupts) and software interrupts (e.g., system calls, exceptions).
- **Benefits**: Enables multitasking, improves system responsiveness, facilitates handling of asynchronous events, and allows for error handling and recovery.

34.**Execution of Instructions in 8085 Microprocessor**:

- The 8085 microprocessor fetches instructions from memory using the program counter (PC), decodes them, and executes them sequentially.
- The flag register stores status flags indicating the result of arithmetic and logic operations, while general-purpose registers (e.g., accumulator) hold operands and results during instruction execution.

35.**Input-Output Processor (IOP)**:

- An IOP is a specialized processor dedicated to managing I/O operations in a computer system.
- Features include I/O instruction set, data buffering, interrupt handling, and I/O device control.
- Advantages: Offloads I/O tasks from the CPU, improves system performance, and provides efficient handling of I/O operations.
- Disadvantages: Adds complexity and cost to the system, requires additional hardware resources.

36.**Multiple Bus Structure**:

- Multiple bus structure involves segregating different types of data transfers (e.g., instruction fetch, data transfer) onto separate buses.
- **Single Bus**: Uses a single bus for all data transfers, limiting bandwidth and causing contention.
- **Double Bus**: Utilizes separate buses for instruction fetch and data transfer, improving bandwidth and reducing contention, enhancing system performance.

37.**Serial Communication**:

- Serial communication involves transmitting data one bit at a time over a single communication line.
- **Types**: Asynchronous serial communication (e.g., RS-232), synchronous serial communication (e.g., SPI, I2C).
- **Transmission Modes**: Simplex, Half-duplex, Full-duplex.

38.**Asynchronous Data Transfer**:

- Asynchronous data transfer involves sending data without a shared clock signal, relying on start and stop bits for synchronization.
- **Methods**: Start-stop method, Baud rate method.
- **Advantage**: Flexibility, compatibility with devices operating at different speeds, simplicity of implementation.

# UNIT 4

 **Shift Instructions**:

- Shift instructions are operations that move the bits of a binary number to the left or right, either preserving the sign bit or shifting in zeros.
- Examples include logical shift left (LSL), logical shift right (LSR), arithmetic shift left (ASL), and arithmetic shift right (ASR).

 **Processor Organization**:

- Processor organization refers to the structure of a CPU, including its main components such as the Arithmetic Logic Unit (ALU), Control Unit (CU), Registers, and Bus Interface.

 **Addressing Mode in 8085**:

- Addressing mode in 8085 microprocessor determines how the operand is specified in an instruction. Examples include direct addressing, indirect addressing, and immediate addressing.

 **Single Accumulator Organization**:

- In single accumulator organization, the CPU has only one general-purpose register known as the accumulator, which is used for most arithmetic and logic operations.

 **Direct Addressing Mode**:

- Direct addressing mode specifies the operand directly in the instruction. For example, MOV A, M in 8085 assembly language moves the content of memory location addressed by HL register pair to the accumulator.

 **Data Transfer Instructions**:

  - Data transfer instructions move data between memory and registers, or between different registers within the CPU, without altering the data itself.

- **Opcode and Operand**:
  - Opcode is a part of the instruction that specifies the operation to be performed, while the operand specifies the data on which the operation is to be performed. For example, in the instruction MOV A, B, MOV is the opcode, and B is the operand.

- **Examples of Data Transfer Instructions**:
  - Examples include MOV (move), LDA (load accumulator), STA (store accumulator), LXI (load immediate), etc.

- **Data Manipulation Instructions**:
  - Data manipulation instructions perform operations on data stored in registers, such as addition, subtraction, logical AND, logical OR, etc.

- **General Register-Based CPU Organization**:
  - In general register-based CPU organization, the CPU has multiple general-purpose registers that can be used to store operands and intermediate results during program execution.

- **Reduced Instruction Set Computer (RISC)**:
  - RISC is a type of microprocessor architecture that emphasizes a small and highly optimized set of instructions, resulting in simpler instruction decoding and faster execution.

- **Advantages of Using Stack-Based CPU Organization**:
  - Advantages include efficient use of memory, support for nested function calls, and simplified program control flow.

- **Move Instruction in Microprocessor**:
  - Move instruction copies data from one location to another without altering the original data.

- **Direct Addressing Mode**:
  - Direct addressing mode specifies the memory address directly in the instruction to access the operand.

- **Complex Instruction Set Computer (CISC)**:

- CISC is a type of microprocessor architecture that supports a large and diverse set of complex instructions, often requiring multiple clock cycles to execute.

 **Addressing Modes in Microprocessors**:

- Addressing modes define how the operand is specified in an instruction, such as direct addressing, indirect addressing, register addressing, etc.

 **Push Operations in Stack-Based CPU Organization**:

- Push operations store data onto the stack. For example, the PUSH instruction in assembly language pushes the content of a register onto the stack.

 **Advantages and Disadvantages of General Register-Based CPU Organization**:

- Advantages include flexibility in data manipulation and efficient use of registers. Disadvantages may include increased hardware complexity and cost.

 **Two Address Instruction in Assembly Programming**:

- A two-address instruction involves two operands, where one operand serves as the source and the other as the destination for the operation.

 **Program Control Instructions**:

- Program control instructions alter the sequence of program execution, such as branching instructions (JMP, JZ), subroutine call instructions (CALL), and return instructions (RET).

## SHORT ANSWERS 5 MARKS

23. **General Register-Based CPU Organization**:

- General register-based CPU organization utilizes multiple general-purpose registers for storing data and operands during program execution.
- Types include Single Accumulator, Multiple Accumulator, and Stack-based architectures.

24. **RISC vs. CISC**:

- **RISC (Reduced Instruction Set Computing)**: Emphasizes a small and highly optimized set of instructions, with each instruction performing a simple task.

- **CISC (Complex Instruction Set Computing)**: Supports a large and diverse set of complex instructions, often requiring multiple clock cycles to execute.

25. **Implicit vs. Explicit Addressing Modes**:

- **Implicit Addressing**: The operand is implicitly specified by the instruction mnemonic. Example: INR A (increment accumulator).
- **Explicit Addressing**: The memory address of the operand is explicitly provided in the instruction. Example: MOV A, M (move data from memory to accumulator).

26. **One-Address vs. Zero-Address Instructions**:

- **One-Address Instruction**: Operates on data stored in memory and one register operand. Example: ADD operand (add the contents of memory operand to accumulator).
- **Zero-Address Instruction**: Operates directly on data stored in the CPU's registers. Example: POP (pop the top value from the stack into the accumulator).

27. **Role of Relative Addressing Mode**:

- Relative addressing mode specifies the operand as a displacement relative to the current instruction's memory address.
- Example: JMP label (unconditional jump to a memory address calculated as current address + displacement).

28. **Implied vs. Immediate Mode of Addressing**:

- **Implied Addressing**: The operand is implicitly specified by the instruction mnemonic without explicitly providing an operand.
- **Immediate Addressing**: The operand value is directly provided in the instruction. Example:

5 marks each these are

ChatGPT

Apologies for the oversight. Here are the responses to the Level B questions with 5 marks each:

23. **General Register-based CPU Organization**:

- In a general register-based CPU organization, multiple general-purpose registers are used to store data and operands during program execution. This architecture allows for more efficient data manipulation and enables a wider range of operations to be performed without accessing memory frequently. Types of register-based CPU organizations include Single Accumulator, Multiple Accumulator, and Stack-based architectures.

24. **RISC vs. CISC**:

- **RISC (Reduced Instruction Set Computing)**: RISC architectures emphasize simplicity and efficiency by employing a small and optimized set of instructions. Each instruction performs a single task, resulting in faster execution and better pipelining. Examples include MIPS and ARM.
- **CISC (Complex Instruction Set Computing)**: CISC architectures support a larger and more complex set of instructions, including multi-step operations. These architectures aim to reduce the number of instructions required to perform a task, often sacrificing execution speed for versatility. Examples include x86 and x86-64.

25. **Implicit vs. Explicit Addressing Modes**:

- **Implicit Addressing**: In implicit addressing modes, the operand is implied by the instruction mnemonic itself. The operation acts directly on a specified register or memory location without explicitly specifying the operand's address. For example, the INR A instruction increments the contents of the accumulator register without explicitly specifying the memory address.
- **Explicit Addressing**: Explicit addressing modes involve specifying the operand's memory address directly within the instruction. The instruction explicitly states the memory location from which the operand is to be fetched or stored. For example, the MOV A, M instruction moves the contents of the memory location pointed to by the HL register pair into the accumulator.

26. **One-Address vs. Zero-Address Instructions**:

- **One-Address Instruction**: One-address instructions operate on data stored in memory and one register operand. These instructions typically involve performing an operation between the accumulator and a memory location. An example is the ADD operand instruction, which adds the contents of a memory operand to the accumulator.
- **Zero-Address Instruction**: Zero-address instructions operate directly on data stored in the CPU's registers. These instructions manipulate data within the registers themselves without referencing memory locations. An example is the POP instruction, which retrieves data from the top of the stack into the accumulator.

27. **Role of Relative Addressing Mode**:

- Relative addressing mode is commonly used in branching instructions within the 8085 microprocessor to specify the operand's memory address relative to the current instruction's address. This mode allows for the efficient execution of conditional and unconditional branch instructions by specifying the branch target as an offset or displacement from the current instruction's address. For example, the JMP instruction with a relative address specifies the target location by adding an offset to the current instruction's address.

28. **Implied vs. Immediate Mode of Addressing**:

- **Implied Addressing**: In implied addressing mode, the operand is implicitly determined by the instruction mnemonic itself, without explicitly specifying the operand's address or value. The instruction operates directly on the CPU's registers or internal data paths. An example is the CLC (Clear Carry) instruction, which clears the carry flag without specifying any operands.
- **Immediate Addressing**: Immediate addressing mode involves specifying the operand's value directly within the instruction itself. The operand is a constant or immediate value rather than a memory address. An example is the MOV A, #data instruction, which moves an immediate data value into the accumulator.

29. **Interrupt Instructions**:

- Interrupt instructions are used to initiate interrupt service routines in response to external events or conditions. These instructions facilitate the handling of interrupts by the CPU, allowing it to respond promptly to asynchronous events without disrupting the normal flow of program execution. Examples of interrupt instructions include RST (Restart), INT (Interrupt), and CALL (Call Subroutine), which transfer control to predefined interrupt service routines based on the interrupt type or source.

30. **Difference between Register and Memory**:

- **Register**: Registers are small, high-speed storage locations located within the CPU. They are used to store temporary data, operands, and addresses during program execution. Registers offer fast access times and are directly accessible by the CPU for arithmetic and logic operations. However, the number of registers is limited, and they have a smaller storage capacity compared to memory.
- **Memory**: Memory is a larger storage area that holds program instructions and data during program execution. It provides a larger storage capacity compared to registers but has slower access times. Memory is organized into addressable storage locations,

each of which can store data permanently or temporarily. Unlike registers, memory is external to the CPU and accessed via the memory bus.

## LONG ANSWERS 10 MARKS

31. **Various Types of Addressing Modes in 8085**:

- The 8085 microprocessor supports several addressing modes to access memory locations:
    1. Immediate Addressing
    2. Direct Addressing
    3. Register Addressing
    4. Indirect Addressing
    5. Register Indirect Addressing
    6. Implicit Addressing
    7. Relative Addressing

32. **Various Types of Instruction Formats in Computer Architecture**:

- Instruction formats define the layout and structure of instructions. Common types include:
    1. **Register Operand**: The instruction contains one or more register operands.
    2. **Immediate Operand**: The instruction contains an immediate value as an operand.
    3. **Direct Operand**: The instruction specifies a memory address directly.
    4. **Indirect Operand**: The instruction specifies a memory address indirectly through a register.
    5. **Indexed Operand**: The instruction uses an index register to specify a memory address.

33. **Various Types of Data Manipulation Instructions**:

- Data manipulation instructions perform arithmetic and logical operations on data. Types include:
    1. **Arithmetic Instructions**: ADD, SUB, INC, DEC.

2. **Logical Instructions**: AND, OR, XOR, NOT.
3. **Shift and Rotate Instructions**: SHL, SHR, ROL, ROR.
4. **Compare Instructions**: CMP, TEST.

34. **Various Types of Registers in 8085 Microprocessors**:

- 8085 microprocessors have several types of registers:
  1. **Accumulator (A)**: Stores one operand for arithmetic and logic operations.
  2. **General-Purpose Registers (B, C, D, E, H, L)**: Used for general data storage and manipulation.
  3. **Special-Purpose Registers (PC, SP)**: Program Counter (PC) holds the address of the next instruction, Stack Pointer (SP) points to the top of the stack.
  4. **Flag Register**: Stores status flags indicating the result of arithmetic and logic operations.

35. **Microcontroller vs. Microprocessor**:

- **Microcontroller (µC)**: Integrated circuit containing a CPU, memory, and I/O peripherals on a single chip. Designed for embedded systems and specific tasks.
- **Microprocessor (µP)**: Central processing unit (CPU) of a computer. Requires external memory and peripherals for operation. Used in general-purpose computing devices.

36. **Different Types of Registers in 8085 Microprocessors**:

- 8085 microprocessors include various registers:
  1. **Accumulator (A)**: Stores one operand for arithmetic and logic operations.
  2. **General-Purpose Registers (B, C, D, E, H, L)**: Used for general data storage and manipulation.
  3. **Special-Purpose Registers (PC, SP)**: Program Counter (PC) holds the address of the next instruction, Stack Pointer (SP) points to the top of the stack.
  4. **Flag Register**: Stores status flags indicating the result of arithmetic and logic operations.

37. **Various Types of Program Control Instructions**:

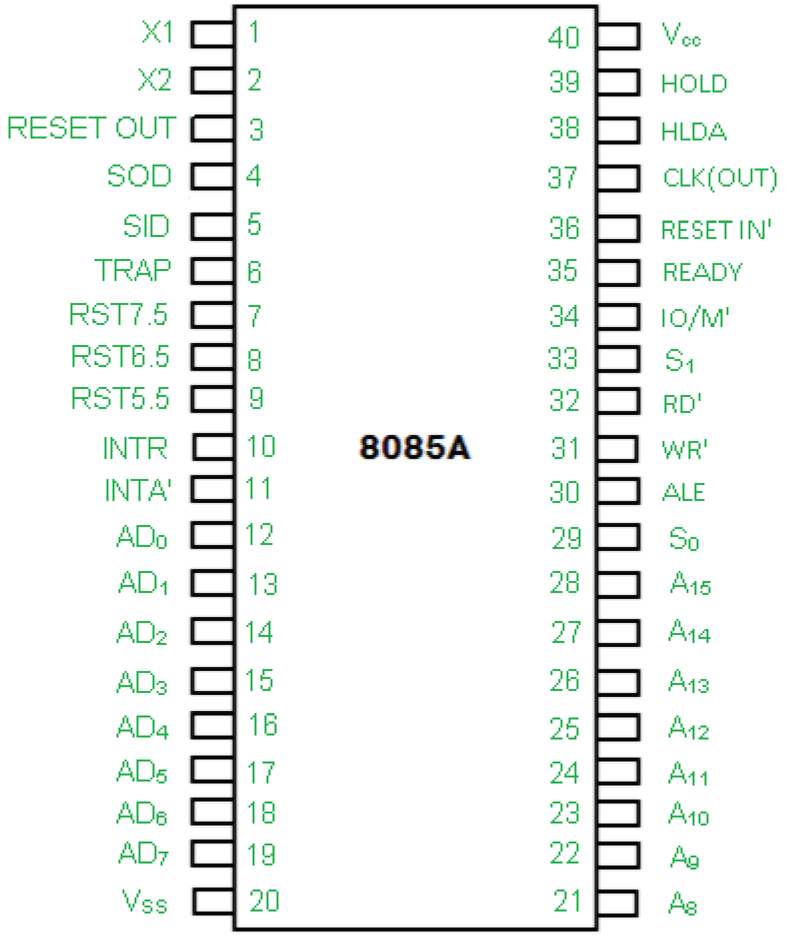
- Program control instructions alter the sequence of program execution. Types include:

1. **Branching Instructions**: JMP (unconditional jump), JZ (jump if zero), JC (jump if carry), etc.
2. **Subroutine Instructions**: CALL (call subroutine), RET (return from subroutine).
3. **Loop Instructions**: LOOP (loop until CX = 0), JCXZ (jump if CX = 0).

38.**Stack Organization in Computer Systems**:

- Stack organization involves using a data structure called a stack to store data temporarily during program execution. Push operation adds data onto the stack, while pop operation retrieves data from the stack. The stack follows the Last In, First Out (LIFO) principle.

39.**Pin Diagram of 8085 Microprocessor**:



- The pin diagram of the 8085 microprocessor illustrates the connections of various pins on the chip to external components. It includes address bus pins, data bus pins, control signal pins, power supply pins, and clock signal pins.

# UNIT 5

## VERY SHORT ANSWERS 2 MARKS

 **Pipeline in a CPU**:

- A pipeline in a CPU is a technique used to enhance processing efficiency by dividing the execution of instructions into multiple stages. Each stage performs a specific task, and multiple instructions can be processed simultaneously, overlapping different stages of execution.

 **Parallel Processing vs. Serial Processing**:

- **Parallel Processing**: In parallel processing, multiple tasks or instructions are executed simultaneously using multiple processing units. This approach increases throughput and reduces processing time.
- **Serial Processing**: In serial processing, tasks or instructions are executed one after the other, sequentially. Only one task is processed at a time, leading to a slower overall execution compared to parallel processing.

 **Difference between Volatile and Non-Volatile Memory**:

- **Volatile Memory**: Volatile memory requires continuous power to retain data. It loses its stored data when power is turned off. Examples include RAM (Random Access Memory).
- **Non-Volatile Memory**: Non-volatile memory retains stored data even when power is turned off. It does not require continuous power to maintain data integrity. Examples include ROM (Read-Only Memory) and flash memory.

 **Arithmetic Pipelining**:

- Arithmetic pipelining is a technique used to enhance arithmetic operation throughput by breaking down arithmetic operations into smaller stages. Each stage performs a specific arithmetic operation, allowing multiple arithmetic operations to be executed simultaneously in different stages of the pipeline.

- **MIMD in Parallel Processing**:
  - MIMD (Multiple Instruction, Multiple Data) is a parallel processing architecture where multiple processing units execute different instructions on different sets of data simultaneously. Each processing unit operates independently, enabling parallel execution of diverse tasks.
- **Role of RAM in Processing**:
  - RAM (Random Access Memory) plays a crucial role in processing by providing fast access to data and instructions required by the CPU during program execution. It serves as a temporary storage medium for actively used data and program instructions.
- **Assembler & Macros**:
  - **Assembler**: An assembler is a program that translates assembly language code into machine code, allowing the CPU to execute the instructions. It converts mnemonic instructions and symbolic addresses into binary code.
  - **Macros**: Macros are predefined sequences of assembly language instructions that can be reused multiple times within a program. They allow programmers to define and use custom instructions to simplify coding and improve code readability.
- **Different Levels of Memory in Memory Hierarchy**:
  - Memory hierarchy typically consists of multiple levels, including:
    - Registers
    - Cache memory
    - Main memory (RAM)
    - Secondary storage (Hard disk drive, SSD)
    - Tertiary storage (Tape drive)
- **Role of Secondary Storage in a Computer System**:
  - Secondary storage serves as a long-term storage medium for data and programs that need to be retained when the system is powered off. It provides larger storage capacity compared to primary memory (RAM) but with slower access speeds.
- **Common Applications of Parallel Processing**:
  - Scientific simulations

- Weather forecasting
- Image and video processing
- Financial modeling
- Data mining and analysis

 **Pipeline Hazards**:

- Pipeline hazards are situations in which the pipeline's efficiency is compromised, leading to delays or incorrect execution of instructions. Types of pipeline hazards include data hazards, structural hazards, and control hazards.

 **Auxiliary Memory**:

- Auxiliary memory refers to secondary storage devices used for long-term data storage, such as hard disk drives, solid-state drives, optical discs, and magnetic tapes.

 **Data Hazard in Computer System Architecture**:

- Data hazard occurs when there is a dependency between instructions regarding the data they use. It can lead to conflicts and incorrect results if instructions are executed out of order or if the required data is not available when needed.

 **Cache Memory and Its Role**:

- Cache memory is a small but fast memory unit located between the CPU and main memory. It stores frequently accessed data and instructions to reduce the time taken to fetch them from main memory, thereby improving overall system performance.

 **Static Linking**:

- Static linking is the process of combining and incorporating all necessary library routines and external functions into the executable file during compilation. This results in a standalone executable that contains all the required code, making it independent of external libraries.

 **Common Types of Secondary Storage Devices**:

- Hard disk drives (HDDs)
- Solid-state drives (SSDs)
- Optical discs (CDs, DVDs, Blu-ray discs)
- USB flash drives

- Magnetic tapes

▯ **SIMD in Parallel Processing**:

- SIMD (Single Instruction, Multiple Data) is a parallel processing architecture where a single instruction is applied to multiple data sets simultaneously. It is commonly used in multimedia processing and scientific computing applications.

▯ **Loading**:

- Loading refers to the process of transferring a program from secondary storage (such as a hard disk drive) into main memory (RAM) for execution by the CPU. It involves copying the program's instructions and data into memory addresses allocated for its execution.

▯ **Macro in Microprocessor**:

- A macro in a microprocessor is a predefined sequence of instructions or operations that can be used repeatedly within a program. Macros help streamline coding and improve code readability by allowing complex operations to be encapsulated and invoked with a single instruction.

▯ **Role of the Scheduler in Parallel Processing**:

- The scheduler in parallel processing is responsible for allocating processing resources (such as CPU cores) to different tasks or processes. It ensures efficient utilization of available resources, minimizes idle time, and maximizes overall system throughput by scheduling tasks for execution based on priority, resource availability, and task depend

## SHORT ANSWERS 5 MARKS

23. **Difference between RAM and ROM**:

- **RAM (Random Access Memory)**:
    - Volatile memory used for temporary data storage.
    - Allows both read and write operations.
    - Data is lost when power is turned off.
- **ROM (Read-Only Memory)**:
    - Non-volatile memory used for permanent data storage.
    - Typically contains firmware and software instructions.

  - Read-only; data cannot be modified or erased.

24. **Memory Hierarchy and its Importance**:

- **Memory Hierarchy**: Memory hierarchy refers to the arrangement of different types of memory in a computer system, organized based on access speed, capacity, and cost. It includes registers, cache memory, main memory (RAM), and secondary storage devices (HDD, SSD).
- **Importance**: Memory hierarchy is crucial for optimizing system performance and efficiency. It allows faster access to frequently used data and instructions by storing them closer to the CPU, reducing memory access latency and improving overall system responsiveness.

25. **Hazards of Parallel Processing**:

- **Data Hazards**: Occur when instructions in parallel execution require access to the same data simultaneously, leading to conflicts and incorrect results.
- **Control Hazards**: Arise when the execution order of instructions is disrupted due to conditional branching or dependencies between instructions.
- **Structural Hazards**: Result from resource conflicts, such as multiple instructions attempting to access the same hardware component simultaneously.

26. **Differences between MISD and SIMD**:

- **MISD (Multiple Instruction, Single Data)**:
  - Multiple processing units execute different instructions on the same data.
  - Rarely used in practice due to limited applicability.
- **SIMD (Single Instruction, Multiple Data)**:
  - Single processing unit executes the same instruction on multiple data sets simultaneously.
  - Commonly used in multimedia processing and scientific computing applications.

27. **Difference Between Compiler and Assembler**:

- **Compiler**:
  - Translates high-level programming languages (e.g., C, C++) into machine code.
  - Generates executable files that can be directly executed by the CPU.

- **Assembler**:
    - Translates assembly language code into machine code.
    - Converts mnemonic instructions and symbolic addresses into binary code.

28. **Performance Difference between Parallel Processing and Serial Processing**:

- **Parallel Processing**:
    - Offers higher throughput and faster execution of tasks by utilizing multiple processing units simultaneously.
    - Suitable for tasks that can be divided into parallel subtasks.
- **Serial Processing**:
    - Executes tasks sequentially, one after another.
    - Slower compared to parallel processing, especially for tasks with parallelizable components.

29. **Types of Mapping Used for Cache Memory**:

- **Direct Mapping**: Each block of main memory maps to exactly one cache line.
- **Associative Mapping**: Each block of main memory can map to any cache line.
- **Set-Associative Mapping**: A compromise between direct and associative mapping, where each block of main memory maps to a set of cache lines.

30. **Flash Memory and Its Types**:

- **Flash Memory**: Non-volatile memory technology used for data storage in devices like USB drives, SSDs, and memory cards.
- **Types**:
    - **NOR Flash**: Offers faster read times and random access, commonly used in firmware storage.
    - **NAND Flash**: Provides higher storage densities and lower costs, commonly used in SSDs, USB drives, and memory cards.

## LONG ANSWERS 10 MARKS

###  **Different Levels of Memory in Memory Hierarchy**:

- Memory hierarchy typically consists of several levels:
    - **Registers**: Fastest and smallest memory located within the CPU.
    - **Cache Memory**: Small-sized but faster than main memory, used to store frequently accessed data and instructions.
    - **Main Memory (RAM)**: Larger than cache memory, stores data and instructions currently being processed by the CPU.
    - **Secondary Storage**: Larger capacity but slower access compared to RAM, used for long-term data storage (e.g., HDDs, SSDs).
    - **Tertiary Storage**: Even larger storage devices (e.g., tape drives), used for archival purposes.

###  **Deadlock in Parallel Processing**:

- Deadlock refers to a situation where two or more processes are unable to proceed because each is waiting for the other to release a resource, resulting in a deadlock state where no progress can be made.
- In parallel processing, deadlock can occur when multiple processes compete for shared resources such as memory, I/O devices, or communication channels.
- Deadlocks can severely impact system performance and must be avoided or resolved using deadlock prevention or recovery mechanisms.

###  **Register in Processor and Types of CPU Registers**:

- A register is a small, high-speed storage location within the CPU used to hold data temporarily during processing.
- Types of CPU registers include:
    - **Data Registers**: Hold data being processed by the CPU.
    - **Address Registers**: Store memory addresses of data or instructions.
    - **Control Registers**: Control various CPU operations and execution modes.
    - **Status Registers**: Hold flags indicating CPU status, such as zero flag, carry flag, etc.

###  **Types of RAM and Their Differences**:

  - **Static RAM (SRAM)**: Faster and more expensive than DRAM, uses flip-flops to store data, requires more power, and is commonly used in cache memory.
  - **Dynamic RAM (DRAM)**: Slower and cheaper than SRAM, uses capacitors to store data, requires refreshing to maintain data integrity, and is commonly used as main memory.

- **Little Endian and Big Endian**:
  - **Little Endian**: Stores the least significant byte of a word at the lowest memory address, while the most significant byte is stored at the highest memory address.
  - **Big Endian**: Stores the most significant byte of a word at the lowest memory address, while the least significant byte is stored at the highest memory address.
  - The difference lies in the byte ordering within multi-byte data types.
- **Hazards in Computer Architecture**:
  - Hazards are conditions that prevent the next instruction in a pipeline from executing during its designated clock cycle, leading to stalls or incorrect program behavior.
  - Types of hazards include:
    - **Data Hazards**: Arise when an instruction depends on the result of a previous instruction that has not yet completed.
    - **Control Hazards**: Occur due to changes in the control flow, such as branches or jumps, affecting the execution sequence.
    - **Structural Hazards**: Result from resource conflicts, such as attempting to access the same hardware component simultaneously.
- **Pipelining and Its Types**:
  - Pipelining is a technique used in CPU design to improve performance by overlapping the execution of multiple instructions.
  - Types of pipelining include:
    - **Instruction Pipelining**: Divides the instruction execution into stages, allowing multiple instructions to be processed simultaneously.
    - **Arithmetic Pipelining**: Breaks down arithmetic operations into smaller stages, enabling multiple arithmetic operations to be executed concurrently.
    - **Superscalar Pipelining**: Utilizes multiple execution units within the CPU to execute multiple instructions in parallel.

- **Types of Parallel Processing**:
  - **Bit-level Parallelism**: Processing multiple bits or binary digits simultaneously.
  - **Instruction-level Parallelism**: Concurrent execution of multiple instructions within a single processor.
  - **Task-level Parallelism**: Distribution of tasks or processes across multiple processors or computing nodes for parallel execution.
  - **Data-level Parallelism**: Simultaneous processing of multiple data elements or data streams.