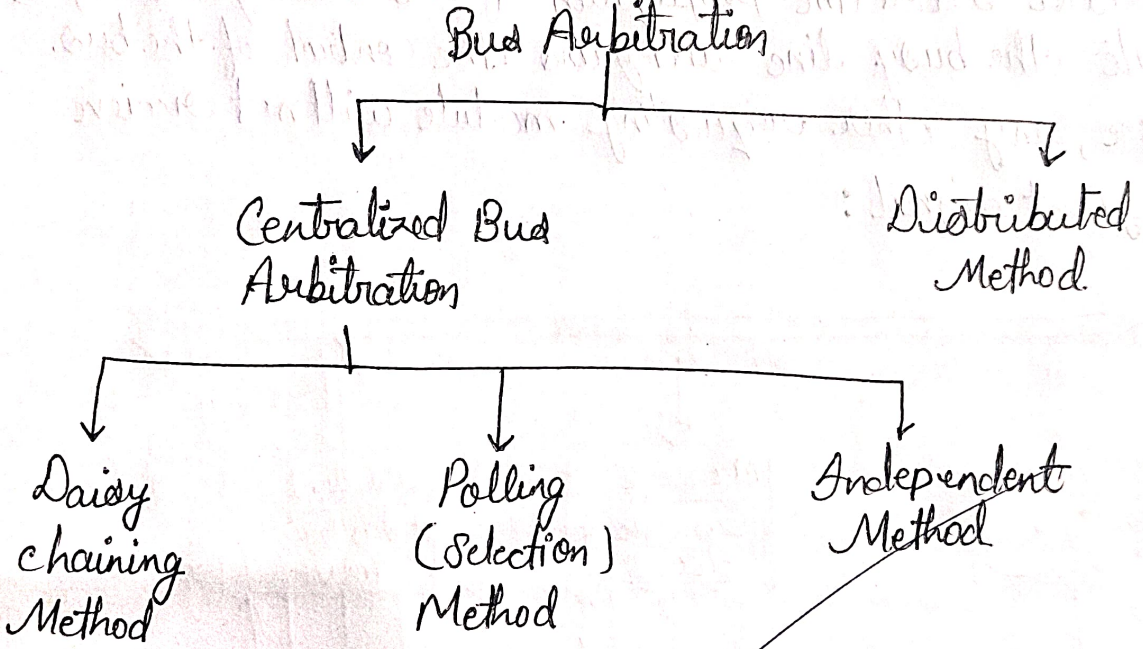
Answer-3. Bus Arbitration – The device that is allowed to initiate data transfer on the bus at any given time is called the bus master.

Bus arbitration is the proceed by which the next device to become the bus master is selected and bus mastership is transferred to it. The selection of bus master is usually done on the priority based.

Approaches to Bus Arbitration –



Centralized Bus Arbitration – In centralized bus arbitration, a single bus arbitor performs the required arbitration. The bus arbitor may be the processor or a separate controller connected to the bus.

Centralized bus approach is implemented by three methods –

Daisy Chaining method: It is a simple and cheaper method. All masters make use of the same line for bus request.

In response to a bus request, the controller sends a bus grant if the bus is free.

The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus.

The master blocks the propagation of the bits-grant signal, activate the busy line and gains the control of the bus. Therefore, any other requesting module will not recieve the grant signal:

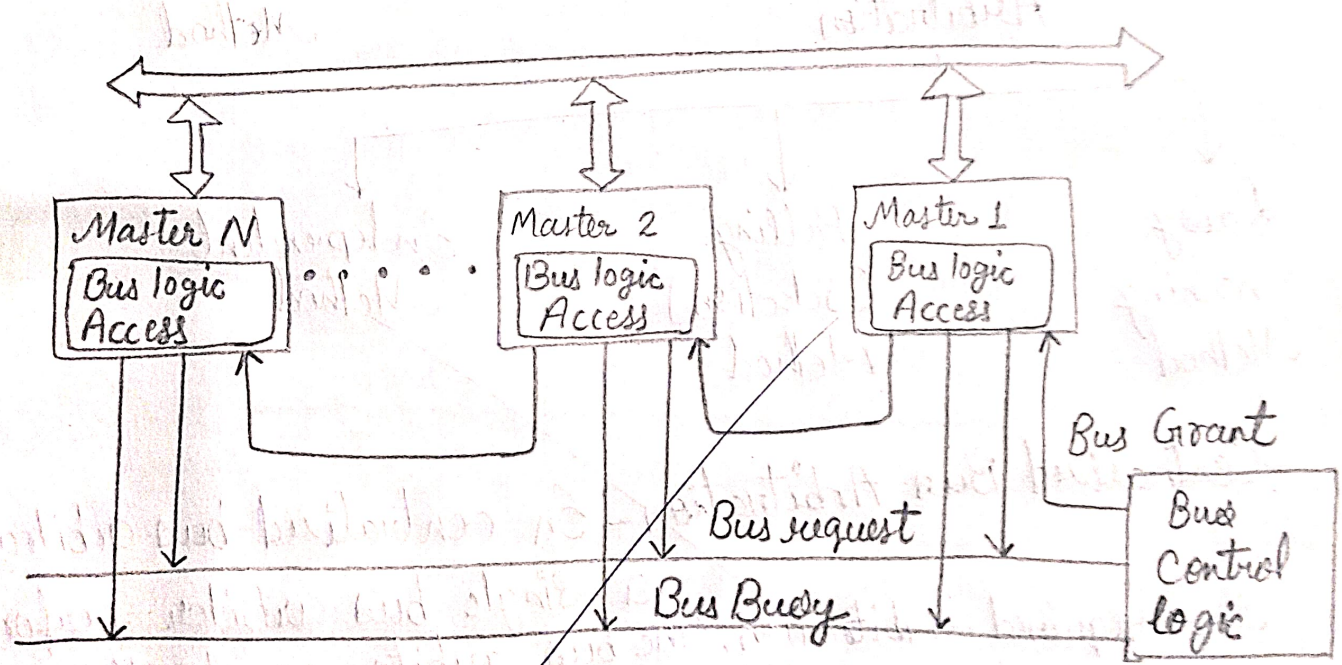


Figure: Daisy Chaining Method

Polling Method: In this, the controller is used to generate the addressed for the masters.

No. of address lines required depends on the no. of masters connected in the system. For e.g. there are eight (8) masters connected in the system, at least 3 address lines are required.

In response to a bus request, controller generates a sequence no. of master address. When a requesting master recognizes, its address; it activates the busy line and begins to use the bus.

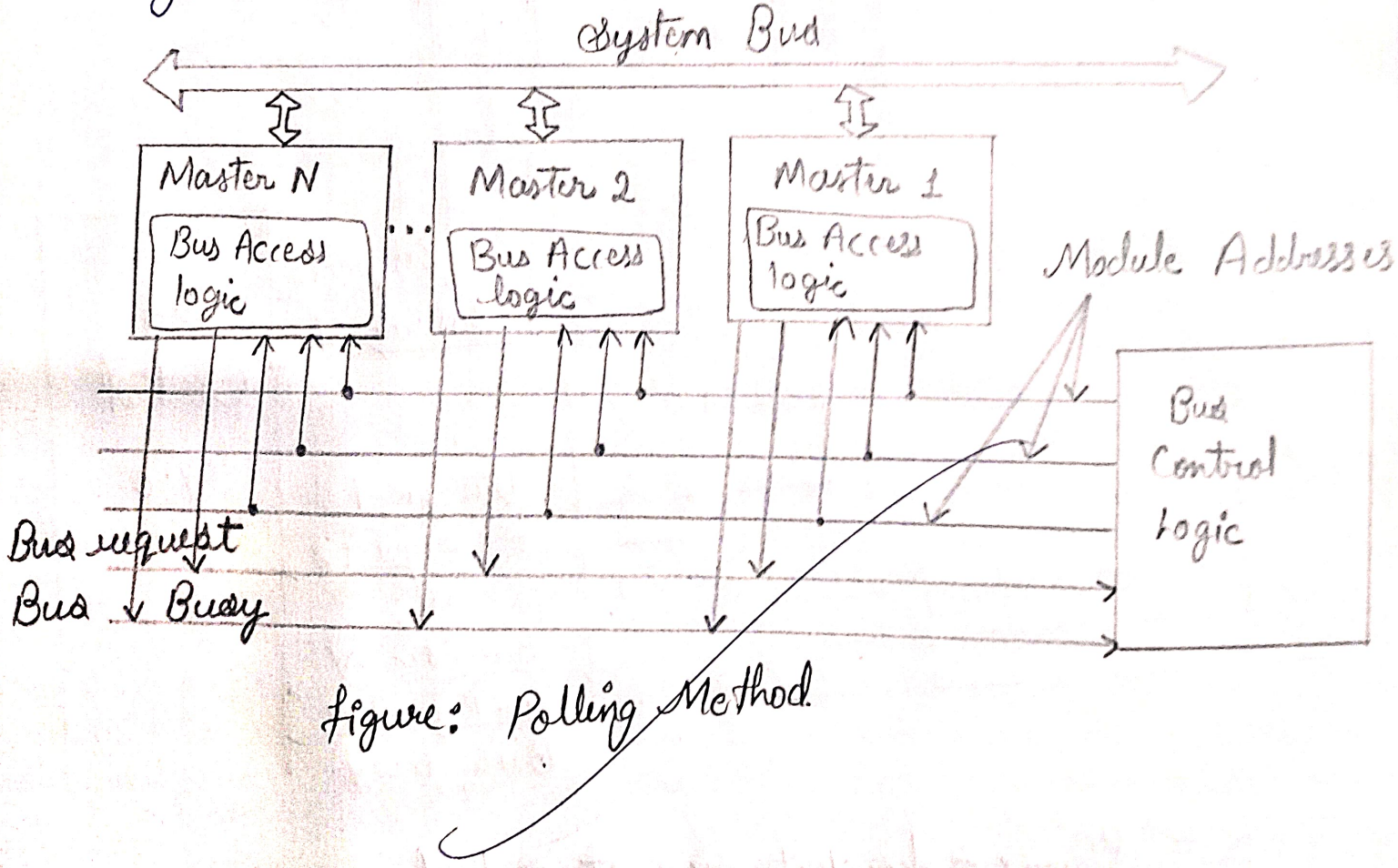


Figure: Polling Method

Independent Method: In this scheme, each master has a separate pair of bus request and bus grant lines and each pair has a priority assigned to it.
The controller selects higher priority request first and activates signal for it.

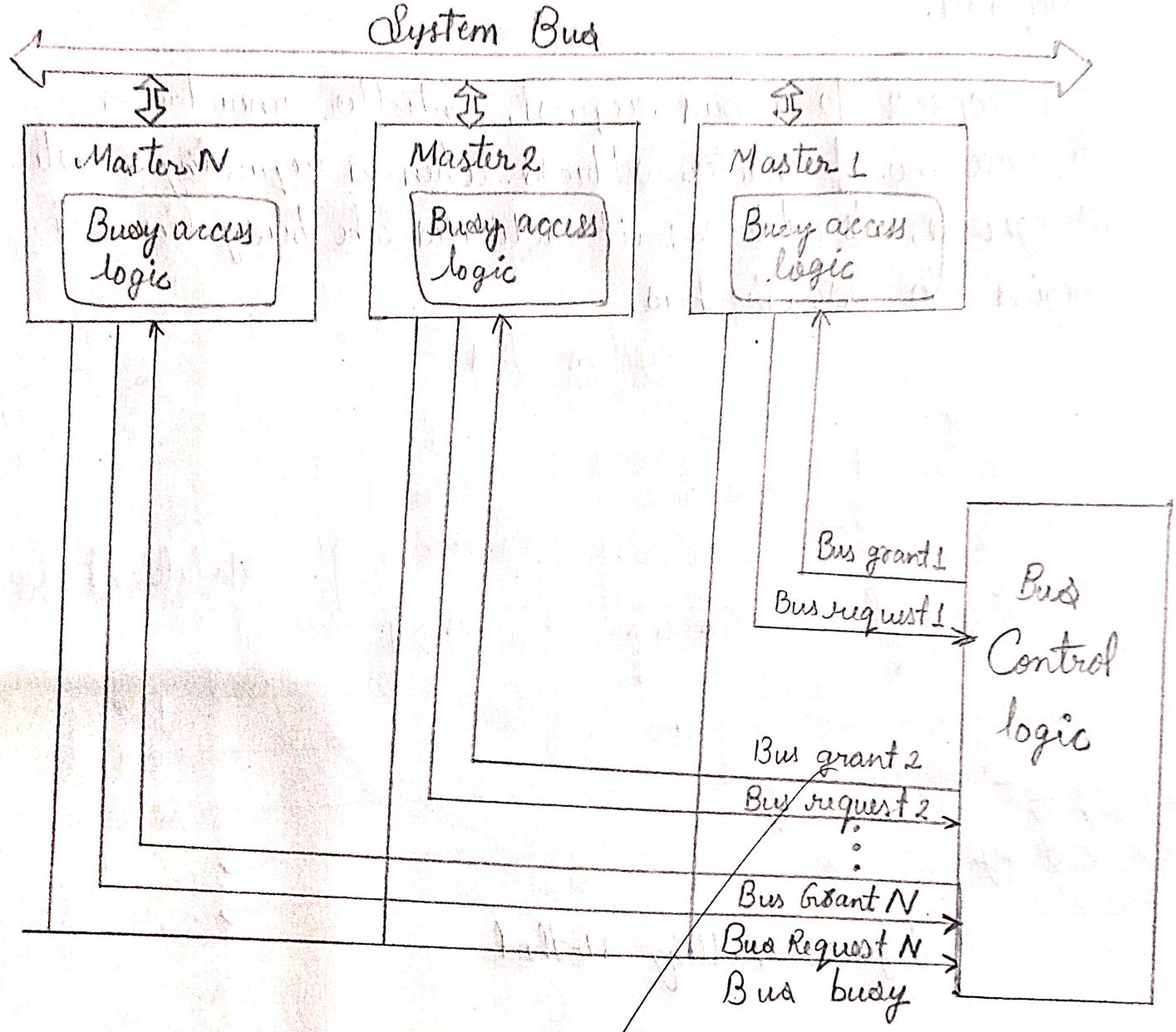


Figure: Independent request method

Distributed Bus Arbitration: In distributed bus arbitration, all devices participate in the selection of the next busmaster. In the scheme each device on the bus. When one or more devices request for the control of bus, they initiate when or (start) arbitration signal and place their 4-bit identification number. In this scheme, the device having higher no. has highest priority.

Answer-4. Representing following decimal in IEEE standard floating-point format in Single precision & double precision method represented -

Given, number $= (1486.125)_{10}$

Convert to binary..

$1486 = (10111001110)_2$

$0.125 = (0.001)_2$

So, $(1486.125)_{10} = (10111001110.001)_2$

Normalize: move point to after the first 1

$(1.0111001110001)_2 \times 2^{10}$

Sign bit = 0 (positive)

Exponent E = 10.

- Single-precision biad = 127 → exponent field = 10 + 127
  = 137 = 10001001_2 (8 bits)
- Double-precision biad = 1023 → exponent field = 10 + 1023
  = 1033 = 10000001001_2 (11 bits)

Mantissa (fraction) = the bits after the leading 1 in normalized form: 0111001110001_2 (followed by zeroes)

- Single mantissa (23 bits) = 01110011100010000000000
- Double mantissa (52 bits) = 0111001110001000000000000000000000000000000000000000

Final IEEE-754 representations:

Single precision (32-bit)

Sign | Exponent (8) | Mantissa (23)

0	10001001	01110011100010000000000

Double precision (64-bit)

Sign | Exponent (8) | Mantissa (52)

0	10000001001	0111001110001000000000000000000000000000000000000000

Answer-5 IEEE-754 Floating-Point Representation -

IEEE-754 is the standard that defines how real numbers are stored in binary inside computers so calculations stay consistent across all machines.

It represents numbers in this form:

Number = $(-1)^{sign} \times 1.fraction \times 2^{(exponent - bias)}$

Everything comes from 3 fields: Sign bit, Exponent Field, Mantissa / Fraction field.

→ The three fields:

(1) Sign bit (1 bit) -

- 0 → positive
- 1 → negative

(2) Exponent (Biased exponent) -

The exponent is stored using a ~~bias~~ bias, not as a signed integer.

- Single precision (32-bit):
  - Exponent bits: 8
  - Bias: 127
- Double precision (64-bit):
  - Exponent bits: (11)
  - Bias: 1023

Stored exponent = actual exponent + bias

(3) Mantissa / Fraction (significand) —

This stored the binary digits after the leading 1 in normalized numbers.

Example: normalized form:

1.101011 × 2^5 → mantissa = 101011

IEEE format does NOT store the leading 1 (called hidden/implicit bit), giving more precision.

→ Formats —

Single precision (32-bit)

Field	Size	Description
Sign	1 bit	+/−
Exponent	8 bits	Biased exponent
Mantissa	23 bits	Fraction digits

Double precision (64-bits)

Field	Size	Description
Sign	1 bit	+/−
Exponent	11 bits	Biased exponent
Mantissa	52 bits	Fraction digits

Answer 6- Control Unit : The control unit (CU) is the part of the CPU that directs everything. It does not do calculations — it controls who should do what and when.

It is like the CPU's traffic manager.

How it works:

It performs four core actions:

1. Fetch
   - Gets the next instruction from memory using the Program Counter (PC).
   - Sends the address to memory → receives the instructions → places it in the Instruction Register (IR).
2. Decode
   - Breaks the instruction into:
     - Operation (opcode)
     - operands
     - Addressing mode
   - Decides which hardware units will be needed (ALU? memory? Registers?)

3. Generate control signals

This the is the real work.

The CU sends precise electrical control signals to:

- ALU (which operation to perform)
- Registers (which register to read/write)
- Memory (read/write)
- Buses (which data goes where)
- I/O modules

These signals dictate every micro-operation.

4. Execute & Move to Next Instruction

- Coordinates the ALU, registers and memory to complete the instruction.
- Updates the Program Counter.
- Repeats the cycle.

Answer-7 Program Counter: A program is a series of instructions store in the memory. These instruction tells the CPU exactly how to get the desired result.

- The sequence of instructions execution is monitored by Program Counter (PC).
- It keeps track of which instruction is being executed and what the next instruction will be.

Answer-8 Addressing Mode: Addressing Mode is the method used by the CPU to specify the location of the operand (data) required by an instruction.

It tells the processor where to find the data – in a register, in memory, inside the instruction itself, or at an address computed at runtime.

S. No	Addressing mode	Example
1.	Register mode	MOV R_1 R_2
2.	Absolute mode	MOV A_L 2000
3.	Immediate mode	MOV A_L #20
4.	Register indirect	MOV A_1 (R_0)
5.	Auto-Increment	MOV (R_2)$_1$ + R_0
6.	Index Mode	MOV 20[R_1]$_1$ R_0
7.	Relative Mode	JN$_Z$ Back
8	Auto-Decrement	MOV (R_1)$_1$ – R_0
9.	Implied mode	CMA

Answer-9 Types of Instruction formats:

Intruction formats define how an instruction is arranged in bits - mainly how the opcode, operands, and address fields are placed.

The standard major types are:

1. Zero-Address Format
   - No operand Field.
   - Used in stack-based machines.
   - Operands are implicitly on top of the stack.
2. One-Address Format
   - Has one explicit operand.
   - The other operand is implied (usually the accumulator - AC).
3. Two-Address Format
   - Instruction contains two operands.
   - Typically one operand is both source and destination.
4. Three-Address Format
   - Has three explicit operands:

     destination = operand1 op operand2

Answer-20 Difference between Memory stack & Register stack:

Memory stack	Register Stack
1. Implemented in main memory (RAM).	1. Implemented using CPU registers.
2. Slower because memory access takes more time.	2. Very fast because register are inside the CPU.
3. Can grow large → big stack size	3. Limited size → only a few registers available.
4. Used for storing large data, subroutine activation records, return addresses, local variables.	4. Used for holding temporary values, intermediate results, top-of-stack operations.
5. Managed using stack Pointer (SP) stored in a register.	5. Managed using small counter or pointer inside register file.
6. More flexible but slower	6. Very limited but extremely fast.

Answer-11 Buses: A group of wires, called bus is used to provide necessary signals for communication between modules.

→ Bus is a share transmission medium -

1. Must only be used by one device at a time.
2. When used to connect major components of computer system [CPU, memory, input output] is called a system bus.

→ The system bus is seperated into three functional groups:

1. Data bus
2. Address bus
3. Control bus

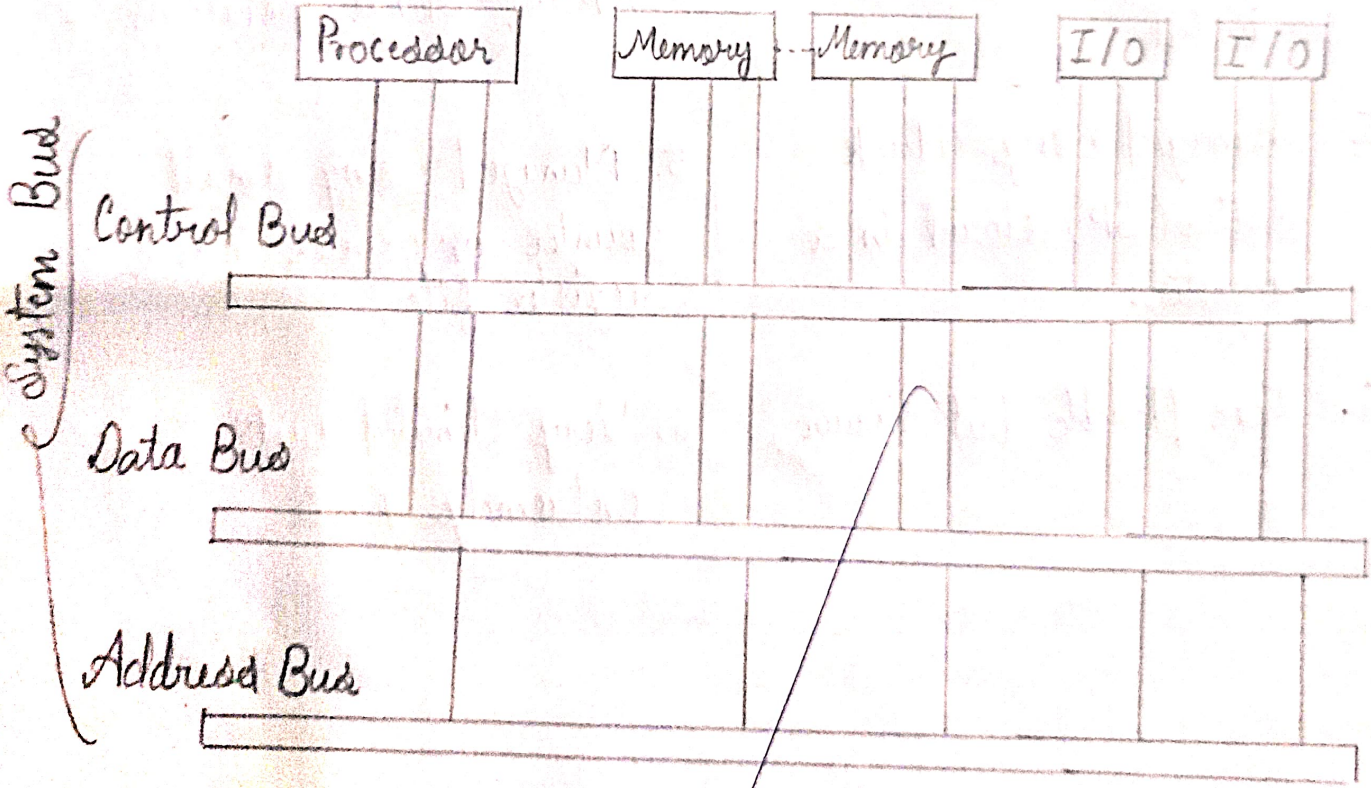


Figure: Bus Interconnection System.

Answer-12 Types of Buses: A bus is a shared communication pathway used to transfer data between computer components.

There are three primary types:

1. Data Bus -
   - Carries actual data between CPU, memory, and I/O devices.
   - Bidirectional (data can go both ways).
   - Width (8, 16, 32, 64 bits) decides how much data can be transferred at once.
2. Address Bus -
   - Carries the memory address or I/O address that the CPU wants to access.
   - Unidirectional (CPU → memory I/O).
   - Width determines max addressable memory (e.g., 32 bits bus → 4 GB address space).
3. Control Bus -
   - Carries control and timing signals such as:
     - Read / Write
     - Interrupt
     - Clock
     - Memory enable
   - Determines how and when data transfer occurs.

Answer-184 Processor Organization: Processor Organization refers to the internal arrangement of CPU components and the way the datapath, registers, ALU, and control signals interact to execute instructions. It focuses on the hardware implementation details such as register structure, internal buses, and execution flow.

Answer-185 Steps in an instruction cycle-

1. Fetch – CPU reads the next instruction from memory using the Program Counter (PC).
2. Decode – Control unit interprets the fetched instructions and identifies the operation and operands.
3. Fetch operands – CPU gets the required data from registers or memory (if needed).
4. Execute – ALU or other units perform the operations (arithmetic, logic, jump, etc.).
5. Store result – The output of the operation is written back to a register or memory.
6. Update PC – PC is updated to point to the next instruction (or modified for branch/jump).

Answer-186 Data Register: A Data Register is a CPU register used to temporarily store data being transferred between the processor and main memory or between the processor and the ALU.

Answer-17. Bus: A bus is a common communication pathway that transfers data, addresses, or control signals between different components of a computer system (CPU, memory, I/O devices).

It reduces wiring by letting multiple units share the same set of lines.

Memory transfer: Memory transfer refers to the process of moving data between the CPU and main memory.

It involves two basic operations:

- Memory Read – data is transferred from memory to CPU.
- Memory Write – data is transferred from CPU to memory.

During memory transfer, the CPU uses the address bus to specify the location, the data bus to move the data, and the control bus to signal read/write.

Answer. 18- Functional Units of a computer system -.

1. Input Unit - Bring raw data and instructions from the outside world into the computer.
   → Accept data from input devices (keyboard, mouse, scanner).
   → Converts it into a binary form the system can understand.
   → Sends it to memory for storage or CPU for processing.

2. Output Unit - Sends processed information back to the user.
   → Takes binary results from the computer.
   → Converts them into human-readable form (text, images, audio).
   → Delivers through devices like monitors, printers, speakers.

3. Memory Unit - (a) Primary Memory (Main memory - RAM, ROM).
   → Temporarily stores data and instructions currently being executed.
   → Fast but limited in size.
   → RAM is volatile; ROM is non-volatile.

   (b) Secondary memory (Hard Disk, SSD).
   → Long-term storage for files, programs, and OS.
   → Slower but much larger.

Functions:

→ Stores instructions before execution.
→ Holds intermediate results.
→ Provides data to CPU when requested.

4. Arithmetic Logic Unit (ALU) – Performs all arithmetic and logical operations.

→ Arithmetic: add, subtract, multiply, divide.
→ Logic: AND, OR, NOT, comparison (<, >, =).
→ Output results to registers or memory.

ALU is the actual calculator of the computer.

5. Control Unit (CU) – Direct the entire system – like a traffic controller.

→ reads and decodes ~~controller~~ instructions.
→ generates control signals to manage memory, ALU, and I/O.
→ Ensures operations happen in the correct sequence.
→ Manage the instruction cycle (fetch → decode → execute).

The Control Unit doesn't execute calculations — it orchestrates them.

6. CPU Registers — Registers are ultra-fast storage locations inside the CPU.

→ Program Counter (PC): Holds address of next instruction.
→ Instruction Register (IR): Holds the current instruction.
→ Accumulator (ACC): Stores intermediate arithmetic results.
→ General Purpose registers: Temporarily hold data, addresses, etc.
→ Flags Register: Stores status bits (zero flag, carry, sign).

Registers give the CPU high-speed access to critical data.

7. Buses — A bus is a communication pathway inside the computer.

→ Data Bus: Carries actual data.
→ Address Bus: Carries memory addresses.
→ Control Bus: Carries signal like Read, Write, Interrupt.

These buses like CPU ⟷ Memory ⟷ I/O.

8. Output / Input Unit (I/O interface) —
Handle communication between CPU / memory and I/O devices.
→ Converts device-specific signals to computer-friendly format.
→ Use controllers and ports to manage data flow.
→ Handle interrupts to notify CPU of events.

Answer-19. Stack Organization in Processors:

A stack is a memory area where the last item pushed is the first item popped.

Operations happen using two stack-specific instructions:

- PUSH → insert data on top of the stack.
- POP → remove data from the top.

The CPU typically has a Stack Pointer (SP) register that always points to the top of the stack.

How stack-based processing works:

Instead of referencing registers explicitly in instructions, the processor executes operations like this:

1. PUSH operands onto the stack.
2. ALU uses top elements of the stack as inputs.
3. After execution, the result is pushed back on the stack.

This removes the need to specify operand addresses in every instructions — the CPU always knows the operands are at the top of the stack.

Example: for A + B × C:

```
PUSH B
PUSH C
MUL   ; C×B → result pushed
PUSH A
ADD   ; A + (B×C) → result pushed.
```

Why it used:

→ Simplified instruction format → no need for operand fields.
→ Reduced hardware complexity.
→ Efficient for expression evaluation (especially postfix/Reverse Polish Notation).
→ Useful for procedure calls and interrupts (store return address, parameters, local variables).

Stack Pointer (SP):

→ A dedicated register pointing to the top of stack.
→ Automatically updated during PUSH / POP.
→ Helps CPU track where to store and retrieve data.

Used in Processors:

1. Expression Evaluation - Stack processors use postfix expressions, making calculations easier and faster for nested expressions.
2. Subroutine calls - → Return address is pushed
   → Parameters and local variables stored on stack.
   → stack unwinds during returns.
3. Interrupt handling -
→ system state is saved onto the stack so the CPU can resume correctly.

Answer-20: General Purpose Registers (GPRs)-

General Purpose Registers are high-speed storage locations inside the CPU used to hold temporary data, operands, memory addresses, and intermediate results during program execution. Unlike special purpose registers, GPRs are not restricted to a single predefined function — the processor can use them for arithmetic operations, logical operations, address calculations, looping, and data transfer.

Key Features —

1. Fast Access: They provide much faster access compared to main memory, reducing execution time.

2. Flexible Usage: They can store —
   - Operands for ALU operations
   - Intermediate results
   - Memory addresses
   - Loop counters / hidden values

3. Programmable: GPRs can be directly used and manipulated by instructions

4. Reduced memory Access: By holding frequently used data, they minimize the number of memory reads / writes.

Examples:

- In 8086: AX, BX, CX, DX.
- In RISC processors (like MIPS): 32 registers (R_0 - R_{31})

More general purpose registers improve CPU efficiency by allowing more data to be kept inside the processor, reducing dependency on slower memory and enabling faster instruction execution.

Answer-21 Role of ALU in a Processor:

The Arithmetic Logic Unit (ALU) is the core computational component of the CPU responsible for performing all arithmetic and logical operations required during instruction execution. It works closely with the control unit and registers to complete the actual data processing tasks of a program.

1. Perform Arithmetic Operations –

The ALU executes basic and complex arithmetic functions such as:

- Addition
- Subtraction
- Multiplication
- Division
- Increment / Decrement

These operations are essential for almost all instructions in any program.

2. Perform Logical Operations –

It handles logical comparisons and bit level operations like:

- AND, OR, NOT, XOR.
- Bit shifts & rotates.
- Comparisons (equal, less than, greater than).

These operations determine program decisions and branching.

3. Works with Registers to Process Data:
The ALU takes operands from general-purpose registers, processes them, and writes the result back into registers. This makes execution fast and reduces memory access.

4. Generates Status / Flag information –
After every operation, the ALU updates flags in the status register (zero, carry, sign, overflow).
These flags are essential for conditional instruction like jumps or loops.

5. Supports Address Calculation –
ALU performs address related calculations such as:
→ Effective address computation.
→ Pointer arithmetic.
→ Indexing in arrays (crucial for memory access instructions).

6. Enables Decision Making in Programs –
Logical results and flag updates allow the CPU to decide: → which instruction to execute next, whether to repeat a loop, whether a condition is true / false.

7. Backbone of Execution Unit –
The ALU forms the main part of the CPU's execution unit. Without it, the processor cannot perform any meaningful computation.

Answer-23 Different Modes of Data Transfer:

Data transfer modes define how data moves between the CPU and peripheral devices or memory. Different methods are used depending on the speed of devices and efficiency requirements.

The main data transfer modes are:

1. Programmed I/O (PIO) – In this mode, the CPU controls and manages the entire data transfer.
   - The CPU repeatedly checks (polls) the device status to know whether it is ready.
   - Once ready, the CPU reads/writes data directly.
2. Interrupt-Driven I/O – Here, the device interrupts the CPU when it is ready for data transfer.
   - CPU performs other tasks instead of polling.
   - When the device signals an interrupt, CPU stops current execution, handles the device, completes transfer, and resumes.
3. Direct Memory Access (DMA) – In DMA, a special controller (DMA controller) transfers data directly between I/O device and memory without involving the CPU for each word/byte.
   - CPU only initializes: source, destination, and block size.
   - DMA handles the transfer independently.

4. Memory-Mapped I/O - In this method, I/O devices share the same address space as memory.
   - Devices registers are accessed using normal load/store instructions.
   - No special I/O instructions required.

5. Isolated I/O (Port-Mapped I/O) - I/O devices use a separate I/O address space.
   - CPU uses special instructions like IN and OUT to access devices.

6. Serial and Parallel Data Transfer -
   - Serial: Data transferred one bit at a time. Suitable for long-distance communication.
   - Parallel: Data transferred multiple bits at once (e.g. 8-bit, 16-bit). Faster but used for short distances.

Answer-24. Addressing Modes define how the CPU identifies the location of operands during instruction execution. Each mode gives flexibility in accessing data stored in registers, memory, or inside the instruction itself.

Below are the major addressing modes used in typical processors:

1. Immediate Addressing Mode -
   The operand (data) is given directly inside the instructions.
   Example: MOV R1, #10
   Here, the value 10 is the operand itself.
   Use: Fastest way to load constants.
2. Register Addressing Mode -
   The operand is located in a register, not in memory.
   Example: ADD R1, R2
   This means: R1 = R2 + contents of R2.
   Use: Very fast because registers are inside CPU.
3. Direct (Absolute) Addressing Mode - The instruction contains the memory address of the operand.
   Example: MOV R1, 5000H
   This loads the contents of memory location 5000H into R1.

4. Indirect Addressing Mode – Instruction specifies a register or memory location that contains the address of the actual operand.

Example: MOV R1, (R2).

R2 contains an address; operand is found at that address.

Use: Useful for accessing variables stored dynamically.

5. Register Indirect Addressing Mode – A register hold the memory address of the operand.

Example (8086): MOV AX, [BX].

BX contains the address where data is stored.

6. Indexed Addressing Mode – Effective address = Base address + Index register. Used heavily in arrays and tables.

Example: MOV AX, [BASE + SI]

Accessed array element using SI index.

7. Base Addressing Mode –

Effective address = Base register + displacement

Displacement is a constant in the instruction.

Example: MOV, AX [BX + 10].

8. Base Scale and Addressing Mode :- Uses base register + index register to compute address.

Example: MOV AX, [BX + SI]

Common for multi-dimensional arrays.

9. Relative Addressing Mode :- Effective address = PC + Displacement.

Used for jumps.

Example: JMP LOOP

Address = current PC + offset to LOOP

10. Implicit (Inherent) Addressing mode:

Operand is not mentioned in the instruction; it is implied.

Example: CLC (Clear Carry Flag)

Operand is predefined (Carry flag)

11. Auto-Increment / Auto-Decrement Addressing Mode :-

Register points to operand and is automatically updated.

Auto-increment example: MOV R1, (R2)+

Use data at address in R2, then R2 = R2 + 1 (or +2 for words).

Auto-decrement example: MOV -(R2), R1

First decrement R2, then access memory.

Useful for stack operations and array traversal.

Answer-25 Booth's Algorithm: Booth's Algorithm is a multiplication algorithm used to multiply signed binary numbers in 2's complement representation. It's main purpose is to reduce the number of addition/subtraction operations during multiplication, especially when the multiplier contains long sequences of 1s.

Booth's method scans the multiplier bit-pair wise and decides whether to add, subtract, or do nothing, based on transitions in the multiplier bits.

Flowchart: