FCO WINTER-2024 PAPER SOLUTION

Q.1 (a) Attempt the following (1 mark each)

1. Convert $(1101.01101)_2$ to decimal

Binary to decimal:

- Integer part: 1101_2
  $= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$
  $= 8 + 4 + 0 + 1 = \mathbf{13}$
- Fraction part: $.01101_2$
  $= 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 1\times2^{-5}$
  $= 0 + 1/4 + 1/8 + 0 + 1/32$
  $= 0.25 + 0.125 + 0.03125 = \mathbf{0.40625}$

Answer:
$(1101.01101)_2 = \mathbf{13.40625_{10}}$

2. Convert $(42.75)_{10}$ to binary

Integer part 42:

42 / 2 → remainders:

- 42 ÷ 2 = 21, r = 0
- 21 ÷ 2 = 10, r = 1
- 10 ÷ 2 = 5, r = 0
- 5 ÷ 2 = 2, r = 1
- 2 ÷ 2 = 1, r = 0
- 1 ÷ 2 = 0, r = 1

Read remainders from bottom → $\mathbf{101010_2}$

Fraction part 0.75:

- 0.75 × 2 = 1.50 → 1 (carry), fraction 0.50
- 0.50 × 2 = 1.00 → 1, fraction 0

So fraction = $\mathbf{.11_2}$

Answer:
$(42.75)_{10} = \mathbf{(101010.11)_2}$

3. Convert $(1011011001)_2$ to octal

Group bits from right side in 3–3:

- 1 011 011 001
  Pad left: 001 011 011 001

Now each group to octal:

- $001_2 = 1$
- $011_2 = 3$
- $011_2 = 3$
- $001_2 = 1$

Answer:
$(1011011001)_2 = \mathbf{(1331)_8}$

4. Convert $(54.6)_8$ to decimal

Octal → decimal:

- Integer part 54_8
  $= 5\times8^1 + 4\times8^0$
  $= 5\times8 + 4\times1 = 40 + 4 = \mathbf{44}$
- Fraction part $.6_8$
  $= 6\times8^{-1} = 6/8 = \mathbf{0.75}$

Answer:
$(54.6)_8 = \mathbf{44.75_{10}}$

5. Convert $(D2E.8)_{16}$ to binary

Each hex digit = 4-bit binary:

- $D_{16} = 13_{10} = 1101_2$
- $2_{16} = 0010_2$
- $E_{16} = 14_{10} = 1110_2$
- $8_{16} = 1000_2$

So:

$(D2E.8)_{16} = \mathbf{1101\ 0010\ 1110\ .\ 1000_2}$

Answer: $(D2E.8)_{16} = \mathbf{(110100101110.1000)_2}$

6. Perform BCD Addition : 789.6 + 516.8

First, **normal decimal addition**:

- Integer part: 789 + 516 = 1305
- Fraction part: 0.6 + 0.8 = 1.4 = 0.4 + carry 1 to integer

So total = 1305 + 1 = **1306.4**

BCD addition idea (digit-wise):

Write each decimal digit in BCD (4 bits):

- 7 = 0111
- 8 = 1000
- 9 = 1001
- 5 = 0101
- 1 = 0001
- 6 = 0110

1. Add 6 and 8 (BCD):
    - 0110 + 1000 = 1110 (14 decimal > 9)
    - Add 0110 (i.e. +6 correction) → 1 0100
    - Result digit = 0100 (4), carry 1 to integer part.
1. Add tenths carry properly with integer digits (9+6+carry etc.) similarly with BCD correction (add 6 if result > 1001_2).

Final BCD result represents **1306.4** (i.e. digits 1-3-0-6-4 in BCD).

Answer (value): 789.6 + 516.8 = **1306.4 (in BCD form)**

7. Convert $(101100.0101)_2$ to hexadecimal

Group in 4–4 bits:

- Integer: 101100_2 → pad left: 0010 1100
- Fraction: .0101 → already 4 bits

Now:

- $0010_2 = 2_{16}$
- $1100_2 = C_{16}$
- $0101_2 = 5_{16}$

Answer:
$(101100.0101)_2$ = **$(2C.5)_{16}$**

Q.1 (b) Difference between AND, OR and NOT gates (using truth tables)

Let inputs = A, B and output = Y.

1. AND Gate

- Symbol: A . B or A ∧ B
- Output is **1 only when both inputs are 1**.

A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

- Symbol: A + B or A ∨ B
- Output is **1 when at least one input is 1**.

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate

- Single input gate.
- Symbol: Y = A' (A bar).
- Output is **complement of input**.

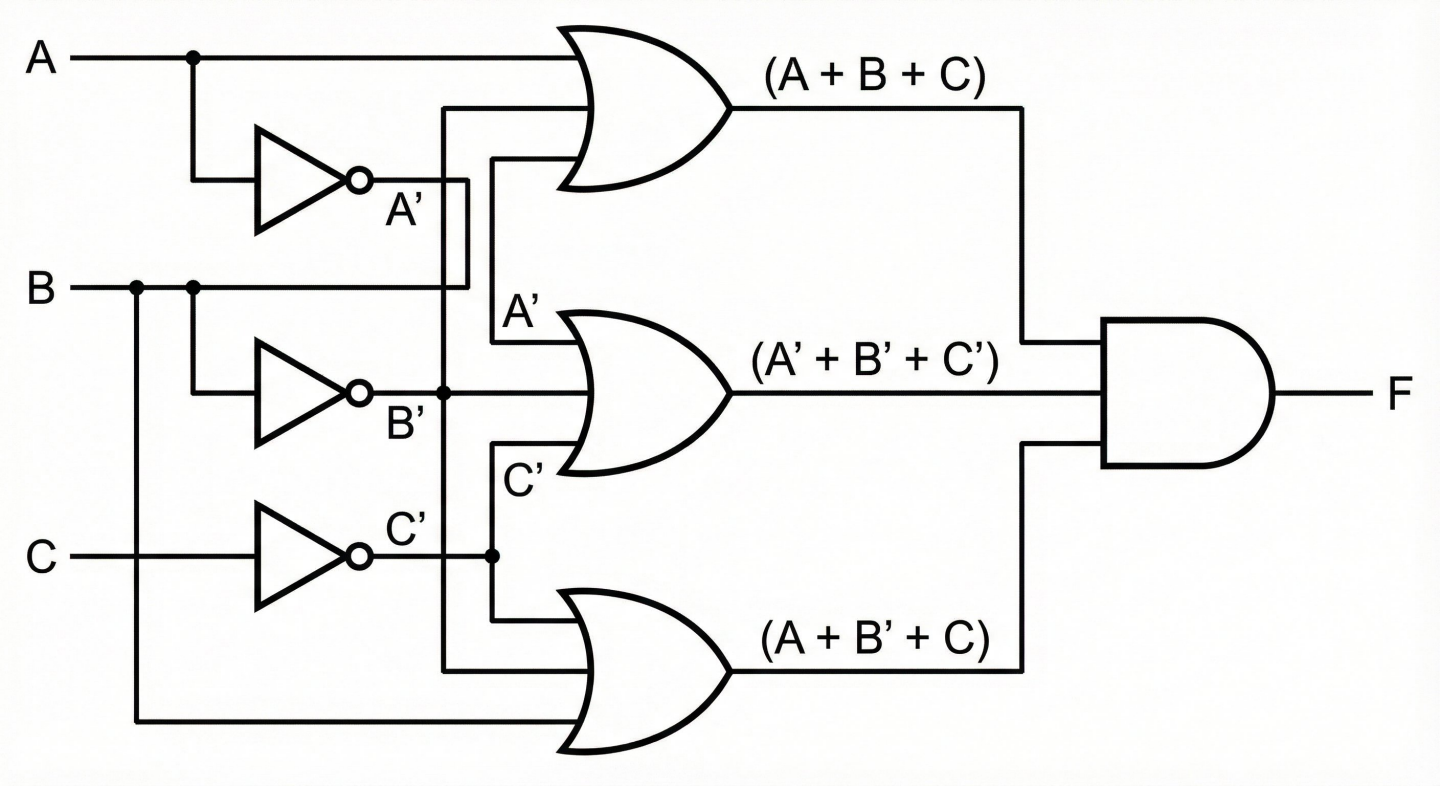
A	Y = A'
0	1
1	0

Main differences:

- **AND** → output 1 only if *all* inputs 1 (logical multiplication).
- **OR** → output 1 if *any one* input 1 (logical addition).
- **NOT** → inverts a single input (0→1, 1→0).

Q-2 (a) Draw logic gate circuit and truth table for

F = (A + B + C) * (A’ + B’ + C’) * (A + B’ + C)**



1. Expression Breakdown

The given Boolean equation consists of **three OR terms** combined by **AND operation**:

1. (A + B + C)
2. (A’ + B’ + C’)
3. (A + B’ + C)

Where:
‘+’ = OR
‘*’ = AND
‘ ’ (bar) = NOT

2. Logic Circuit Description (How to draw in exam)

Step-1: Create three OR gates

1. **First OR gate**
   Inputs: A, B, C
   Output: X = (A + B + C)
2. **Second OR gate**
   Inputs: A’, B’, C’
   Output: Y = (A’ + B’ + C’)

3. **Third OR gate**
   Inputs: A, B', C
   Output: Z = (A + B' + C)

Step-2: Insert NOT gates for complements

- NOT gate to convert A → A'
- NOT gate to convert B → B'
- NOT gate to convert C → C'

Step-3: Final AND gate

Connect outputs X, Y, Z to an AND gate to produce:

F = X * Y * Z

(Write neatly and label properly in your drawing)

3. Truth Table

Since variables A, B, C are binary, there will be **8 combinations**:

A	B	C	A+B+C	A'+B'+C'	A+B'+C	F
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	1	1	1
0	1	1	1	0	0	0
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	1	0
1	1	1	1	0	0	0

Explanation

To get F = 1, **all three OR conditions must be 1** simultaneously.
This occurs for input combinations:

- 0 0 1
- 0 1 0
- 1 0 0
- 1 0 1

Hence truth table verifies the function.

Q-2 (b) Explain single bus structure and double bus structure with diagram, advantages and disadvantages.

1. Meaning of Bus

A **bus** is a common communication pathway used to transfer data, address and control signals among different components of the computer such as CPU, memory and I/O devices.

2. Single Bus Structure

(i) Explanation

In a **single bus structure**, all major components of the computer system are connected to **one common bus**.
The same bus is used to transfer:

- data
- addresses
- control signals

between CPU, main memory and I/O devices.

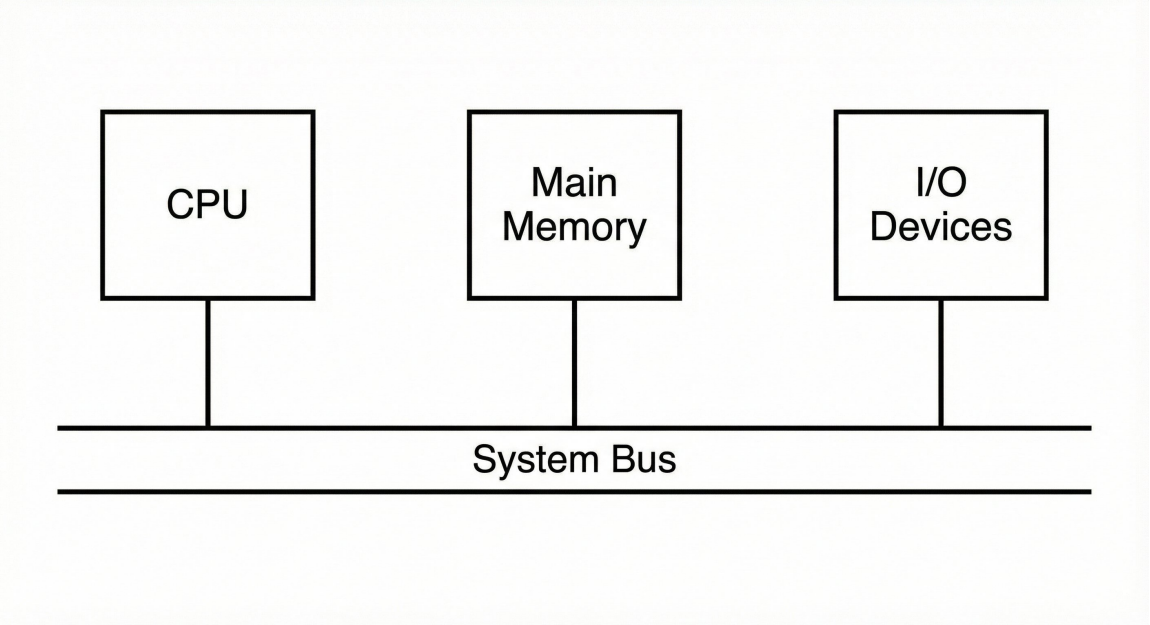
Whenever any two units want to communicate, they use the common bus. Only one transfer can occur at a time.

(ii) Diagram (how to draw)

Draw:

- One horizontal line in center labelled **System Bus**
- Connect three blocks to it:
    - CPU
    - Main Memory
    - I/O Devices

Example:



(Write labels clearly: “Single Bus Structure” under the diagram)

(iii) Advantages of Single Bus Structure

1. **Simple Design:**
   Easy to design and implement because only one bus is used.
2. **Low Cost:**
   Requires fewer wires and less hardware, so overall cost is low.
3. **Easy Expansion:**
   New devices can be added by simply connecting them to the same bus.

(iv) Disadvantages of Single Bus Structure

1. **Bus Contention:**
   Only one device can use the bus at a time.
   If many devices request the bus, they must wait.
2. **Lower Performance:**
   No parallel transfers are possible.
   As system grows, speed decreases due to sharing.
3. **Limited Bandwidth:**
   All data, addresses and control signals share the same bus,
   which limits the total data transfer rate.

3. Double Bus Structure

(i) Explanation

In a **double bus structure**, there are **two separate buses** used for data movement.
This is usually used **inside the CPU** for connecting registers and ALU.

- Registers are connected to **two internal buses** (Bus A and Bus B).
- Two operands can be read simultaneously from two registers through these buses and applied to ALU.
- The result of ALU may be stored through another path or bus.

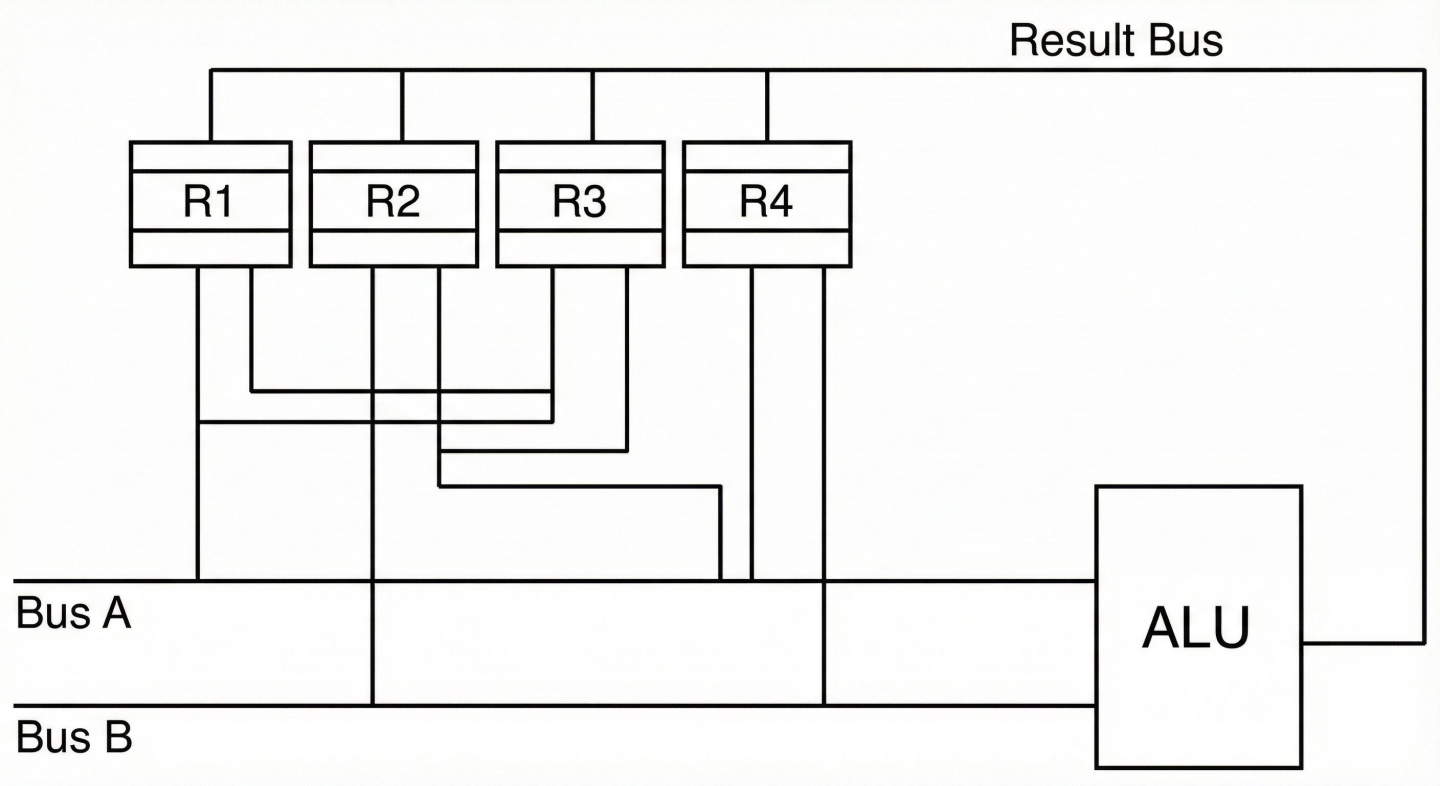
Double bus structure allows more **parallelism** and faster execution of instructions.

(ii) Diagram (internal CPU double bus)

Draw:

- Several registers on left side: R1, R2, R3, …
- Two horizontal buses labelled **Bus A** and **Bus B**
- Lines from each register to both buses
- On right side, an ALU that receives input from Bus A and Bus B
- Output of ALU going to another line/bus to registers

Example:



(Write heading: “Double Bus Structure (Internal CPU)” under the diagram.)

(iii) Advantages of Double Bus Structure

1. **Faster Execution:**
   Two operands can be fetched at the same time using two buses.
   This speeds up arithmetic and logic operations.
2. **Parallel Data Transfer:**
   Multiple data transfers can happen in a single clock cycle
   (for example, reading two registers simultaneously).
3. **Improved Performance:**
   Reduces waiting time on a single bus and increases throughput of the CPU.

(iv) Disadvantages of Double Bus Structure

1. **Higher Hardware Cost:**
   Requires more wiring, more multiplexers and more control signals.
2. **Complex Design:**
   Control unit becomes more complex because it has to manage
   multiple buses and simultaneous transfers.
3. **More Chip Area and Power:**
   Extra buses and circuits consume more physical space and power.

OR

Q.2 (b) Explain Instruction Cycle and describe steps of any two instruction cycles.

Definition of Instruction Cycle

An **Instruction Cycle** is the complete process by which a computer
fetches, decodes, executes, and **stores** the result of an instruction.

It is also called **Fetch–Decode–Execute Cycle**
and it represents **how CPU processes one instruction at a time**.

Stages of Instruction Cycle

The instruction cycle generally consists of the following phases:

(a) Fetch Cycle

- CPU reads (fetches) an instruction from **main memory**.
- Program Counter (PC) contains the address of the instruction.

- Instruction is stored temporarily in the **Instruction Register (IR)**.
- PC is incremented to point to the next instruction.

Output of this stage: Instruction fetched successfully.

(b) Decode Cycle

- The fetched instruction is interpreted by the **Control Unit**.
- CPU identifies:
    - which operation needs to be performed
    - which operands are required
    - whether memory access is needed

Output of this stage: CPU understands what must be done.

(c) Execute Cycle

- CPU performs the operation:
- It may include:
    - arithmetic/logic operation in ALU
    - reading data from memory
    - writing data to memory
    - transferring control to other instruction (branching)

Output of this stage: Task executed and result produced.

(d) Store/Write Back Cycle

- The computed result is stored:
    - in register, or
    - main memory

Output of this stage: Result committed to destination.

Two types of Instruction Cycles (write any two):

Below are two well-described instruction cycles useful for exam:

1. Fetch Cycle

Steps:

1. Program Counter sends address to memory.
2. Memory fetches instruction and sends it to CPU.
3. CPU loads instruction into Instruction Register.
4. PC increments to next instruction.

Purpose:

To bring an instruction from memory to CPU.

2. Execute Cycle

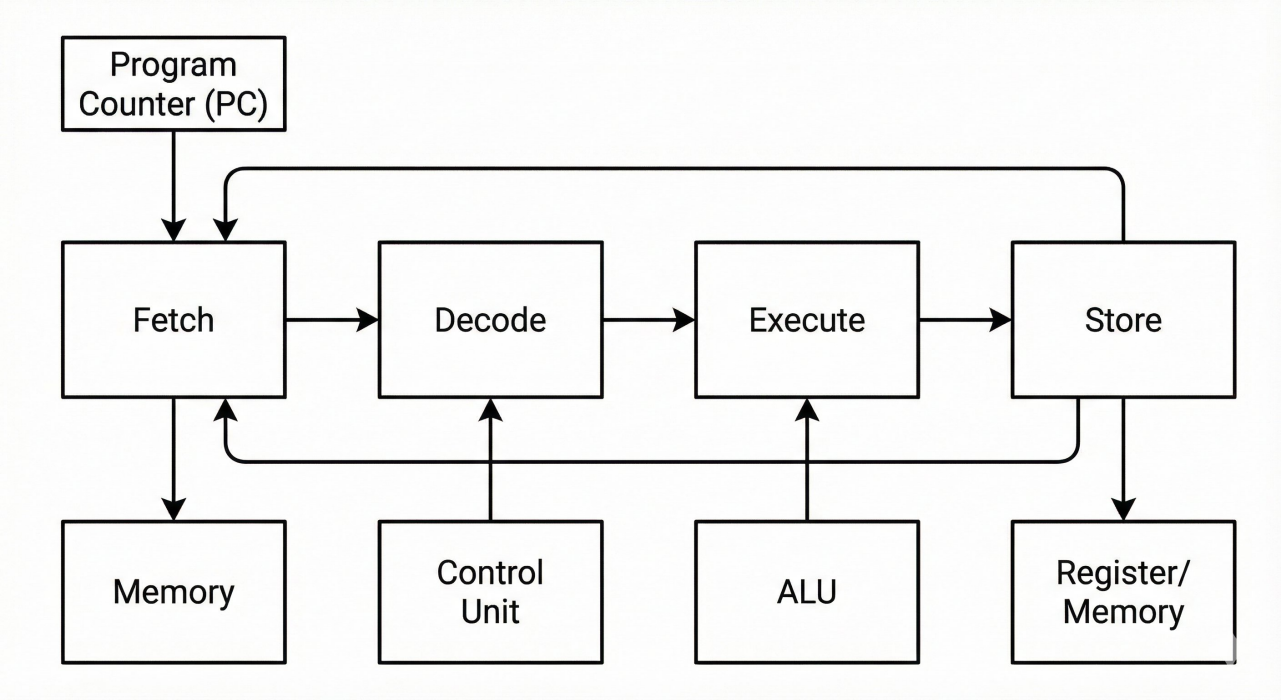
Steps:

1. ALU performs computation based on decoded instruction.
2. Registers/memory provide input operands.
3. Result is calculated.
4. The result is stored or forwarded for next instruction.

Purpose:

To perform actual work required by the instruction.

Diagram Explanation



Label it as: **Basic Instruction Cycle of CPU**

Q.3 (a) Explain the concept of Fixed Point Representation. Write the difference between fixed point and floating point representations?

Concept of Fixed Point Representation

Fixed Point Representation is a method of representing real numbers in binary format where:

- The **position of the decimal point is fixed** in the number.

- The number is divided into two parts:
  integer part and **fraction part**.
- The binary point location does not change during calculations.

Example:

Let us assume 8-bit fixed-point format where:

- 5 bits = integer part
- 3 bits = fractional part

Binary number: 10110.101

= 22.625 in decimal
(because $10110_2 = 22$
and $.101_2 = 0.625$)

Key Characteristics:

- Simple representation
- Fast arithmetic operations
- Works well for integers and small fractional ranges
- Limited dynamic range

Fixed Point Number Format

[Sign bit] [integer bits] [fraction bits]

- MSB may be used as sign
- Position of binary point is predefined and constant

Difference between Fixed Point and Floating Point Representation

Fixed Point Representation	Floating Point Representation
Decimal point position is fixed	Decimal point position is variable (floating)
Suitable for integers and small fractional values	Suitable for representing very large and small real numbers
Simple hardware design	More complex hardware control
Fast arithmetic operations	Slower due to normalization and exponent processing
Limited range and precision	Very large range and better precision
Used in embedded systems, microcontrollers	Used in scientific calculations, calculators, graphics processors

Q.3 (b) Write a short note on Multiplication Algorithms and Division Algorithms. Provide example with diagram.

Multiplication Algorithm

Definition:

Binary multiplication is performed using a method similar to decimal multiplication.
It is based on **shift and add** operations.

Steps of Binary Multiplication Algorithm:

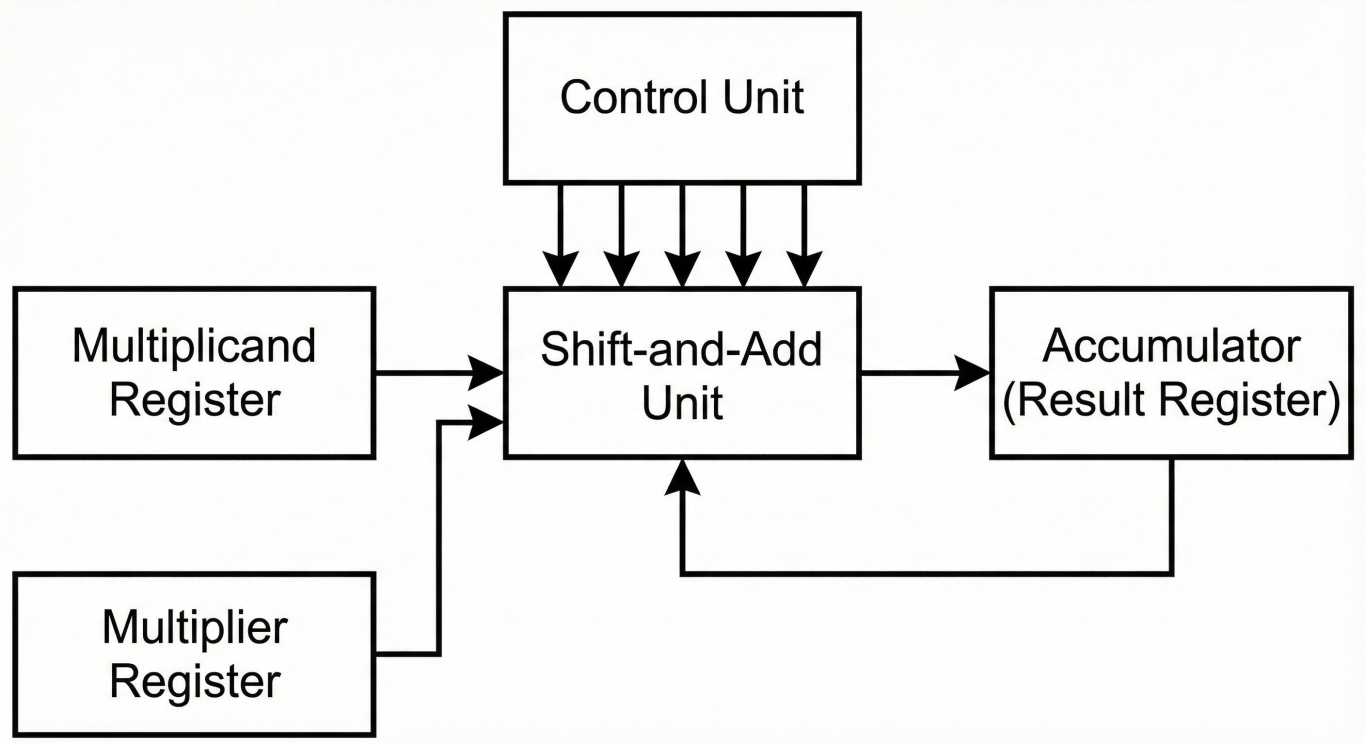
1. Multiply the multiplicand by each bit of multiplier.
2. If multiplier bit = 1 → multiplicand is added.
3. If multiplier bit = 0 → partial product is 0.
4. Each partial product is shifted left.
5. Partial products are added to get final result.

Example of Binary Multiplication:

Multiply:
1010_2 (10) × 0011_2 (3)

```
  1010
× 0011
-------
  1010        ← 1010 × 1
 1010         ← 1010 × 1 shifted left
-------
11110          ← result = 30 decimal
```

Diagram Explanation



- Control Unit manages shifts and additions
- Accumulator stores partial sums

Division Algorithm

Definition:

Binary division is performed using the **shift and subtract** method
(similar to manual long division).

Steps of Binary Division Algorithm:

1. Align divisor with most significant part of dividend.
2. If divisor ≤ current dividend portion → subtract and write quotient bit = 1.
3. If divisor > dividend portion → write quotient bit = 0.
4. Shift divisor right and repeat until all bits processed.

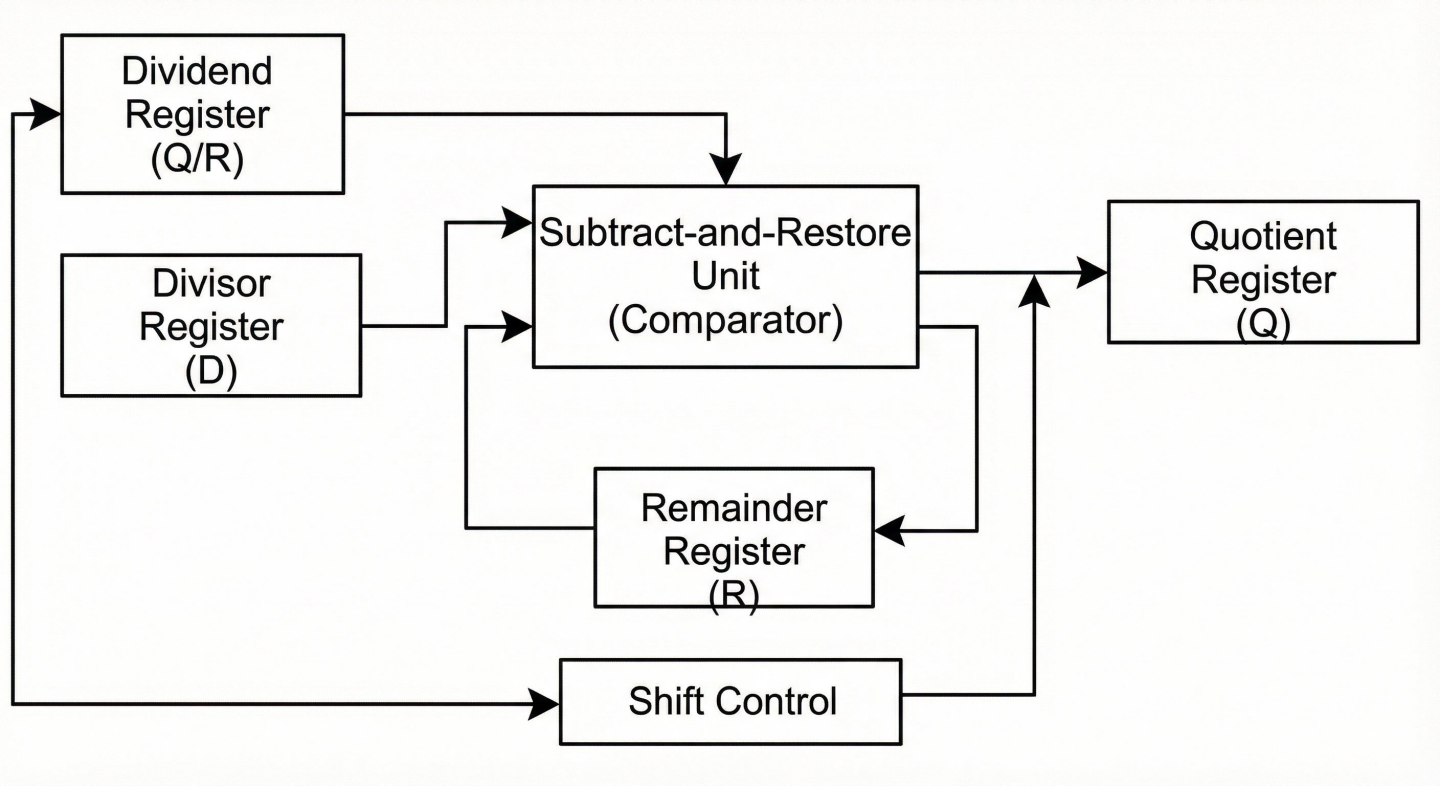
Example of Binary Division:

Divide:
11000_2 (24) ÷ 0011_2 (3)

11000 ÷ 0011 = 01000

= 8 decimal

Diagram Explanation



- Control logic shifts divisor
- Remainder is updated after each subtract
- Quotient bits collected sequentially

OR

Q.3 (a) Why use Shift Register? Explain its working with proper circuit diagram.

Definition of Shift Register

A **Shift Register** is a group of flip-flops connected in series that stores binary data and shifts it **left or right** when clock pulse is applied.

It is mainly used for:

serial-to-parallel conversion
parallel-to-serial conversion
data storage and movement
multiplication/division (by shifting)

Why do we use Shift Registers?

Shift registers are used because they:

1. **Store data bits temporarily**

2. **Shift bits left or right** for arithmetic operations
   (Left shift = ×2, Right shift = ÷2)
3. **Convert serial data into parallel form**
4. **Convert parallel data into serial form**
5. **Used in counters, digital processing, data communication**

Working of Shift Register

Basic Components

A shift register consists of:

- **Flip Flops (D flip-flops commonly used)**
- **Clock pulse**
- **Shift control logic**

Shift Operation:

When a clock pulse arrives:

Data enters first flip-flop
Previous bit moves into the next flip-flop
Final bit shifts out

Example:

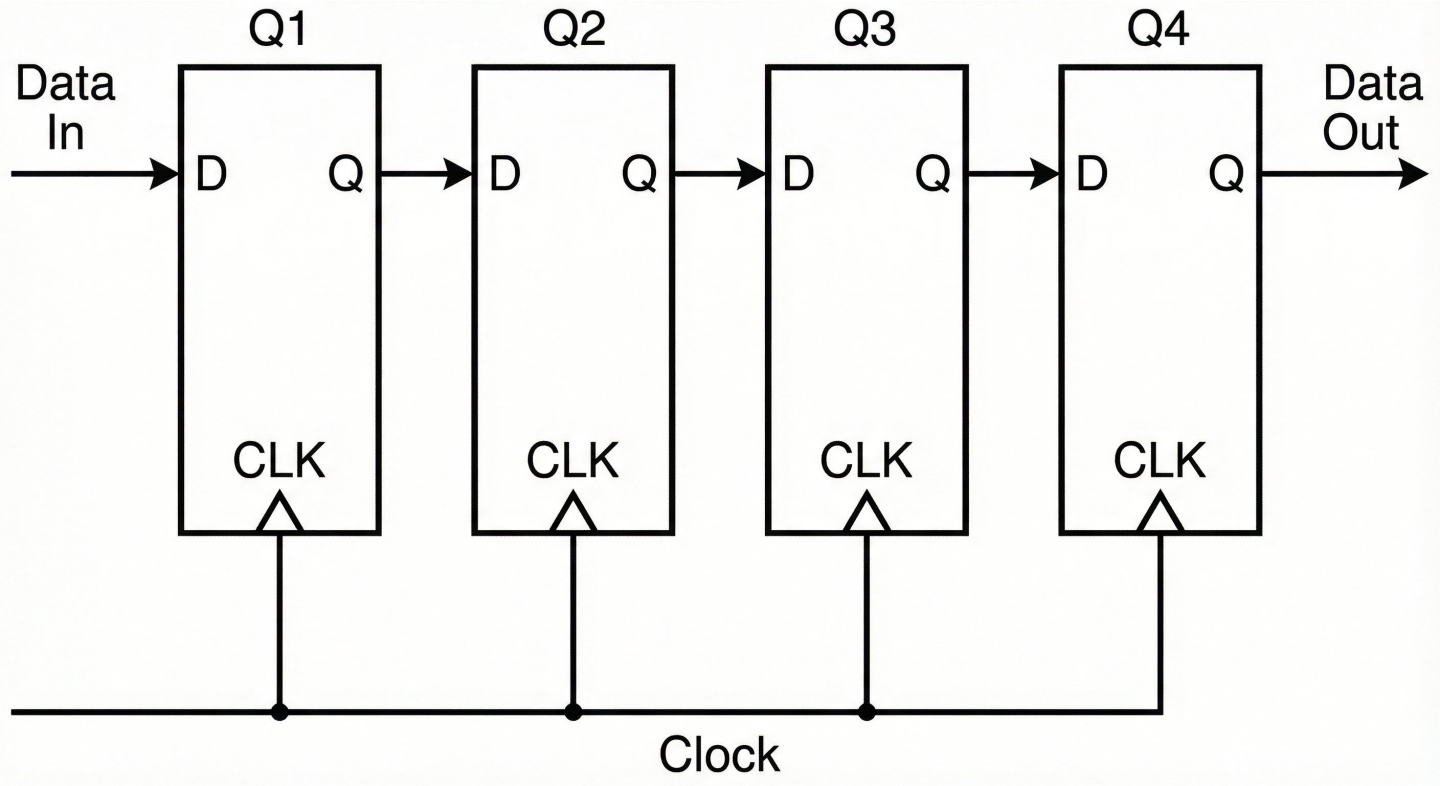
If we store **1011** inside a 4-bit register:

After one right shift → **0101**
After one left shift → **0110**

Circuit Diagram

Draw:

- 4 D flip-flops in series (Q1, Q2, Q3, Q4)
- Clock line connected to each flip-flop
- Output of each flip-flop fed into next one



Label this as **Serial-in Serial-out Shift Register**

Q.3 (b) Write the floating-point arithmetic operations with diagram.

Floating-Point Arithmetic Operations

Floating-point numbers are represented in the form:

± Mantissa × Base^Exponent

Example: $6.25 = 0.625 \times 10^1$

Floating-point arithmetic mostly includes:

Floating-point Addition
Floating-point Subtraction
Floating-point Multiplication
Floating-point Division

Floating-Point Addition/Subtraction

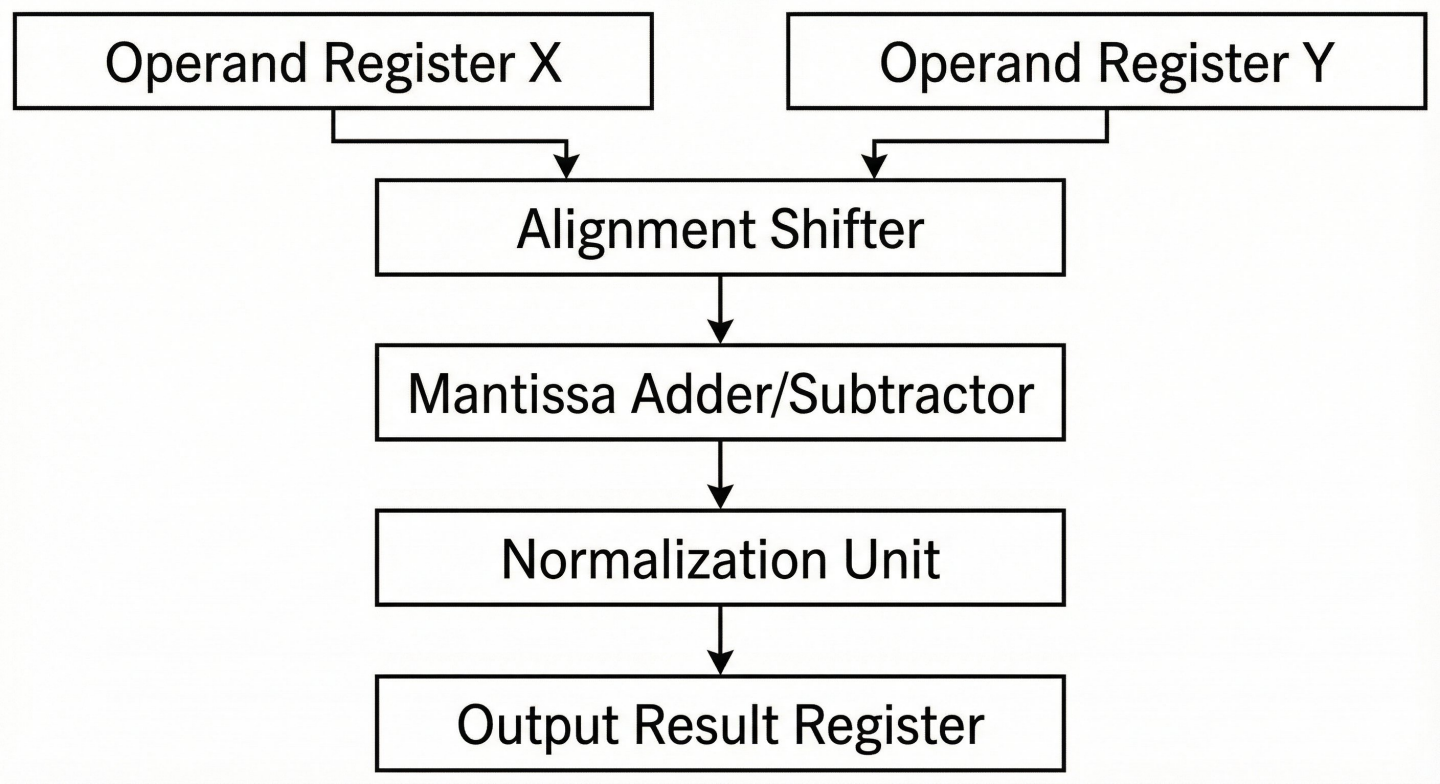
Steps:

1. **Align the exponents**
   Shift the smaller mantissa until both exponents match.
2. **Add/Subtract mantissas**
3. **Normalize the result**
   (shift mantissa so decimal is in proper position)
4. **Round the value if required**

Exam Diagram Explanation:

Draw:

- Two floating point registers: X and Y
- Alignment shifter
- Mantissa adder/subtractor
- Normalization block
- Output register



Floating-Point Multiplication

Steps:

1. **Add exponents**
2. **Multiply mantissas**
3. **Normalize the result**
4. **Round if required**

Floating-Point Division

Steps:

1. **Subtract exponents**
2. **Divide mantissas**
3. **Normalize**
4. **Round**

Q.4 (a) Why use Logic Micro-Operations? Write R1 → R2 Logic Micro-Operation with diagram and truth table.

Why do we use Logic Micro-Operations?

Logic micro-operations are used inside CPU to:

1. **Manipulate binary data at bit level**
   (AND, OR, XOR, NOT)
2. **Perform bit-wise transformations**
   Example: masking, clearing selected bits, setting bits
3. **Support arithmetic, control and data processing**
4. **Help implement ALU functionality**

Thus, logic micro-operations allow CPU to **process data efficiently and modify bit patterns** in registers.

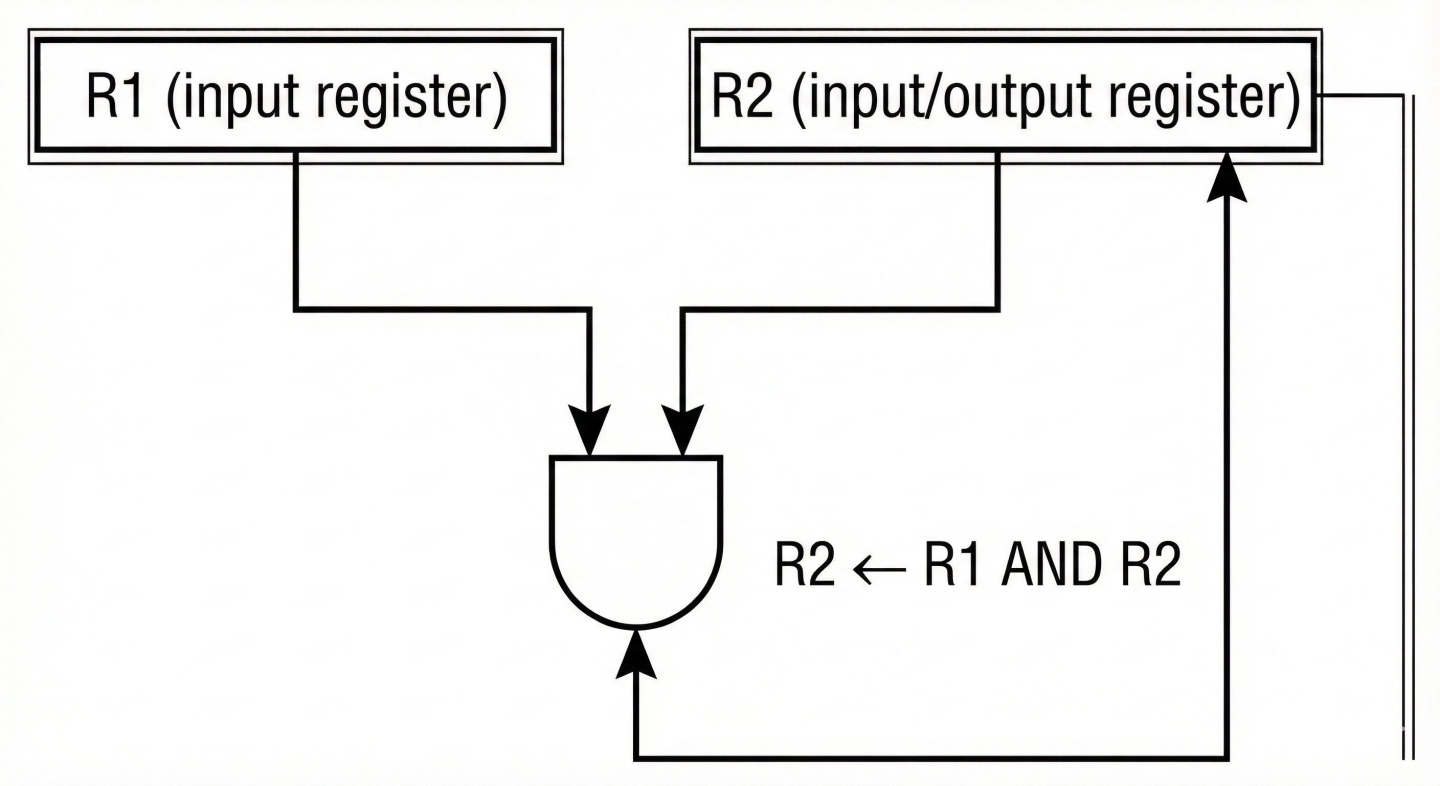
Example of Logic Micro-Operation

➡ R1 → R2 Logic Transfer with AND operation

Expression:
R2 ← R1 AND R2

Meaning:
Each bit of register R1 is AND-ed with R2 and result stored back into R2.

Diagram Representation



Label both:

Input Registers: R1, R2
Logic Circuit: AND gate
Output Register: R2

Truth Table (for one bit example)

R1	R2	R2 (after operation) = R1 · R2
0	0	0
0	1	0
1	0	0
1	1	1

➡ Truth table shows bit-wise AND logic.

Q.4 (b) Define Register Transfer Language. Write the three types of register transfer operations with diagram.

Definition of Register Transfer Language (RTL)

Register Transfer Language is a symbolic notation used to **describe operations performed between registers**.
It specifies:

data movement
arithmetic functions
logic operations

Example:
R2 ← R1 means copy content of Register R1 into R2.

Types of Register Transfer Operations

(A) Data Transfer Operation

Moves data from one register to another.

Example:
R2 ← R1

Diagram:
R1 → Bus → R2

(B) Arithmetic Operation

Uses ALU to perform arithmetic (add, subtract)

Example:
R3 ← R1 + R2

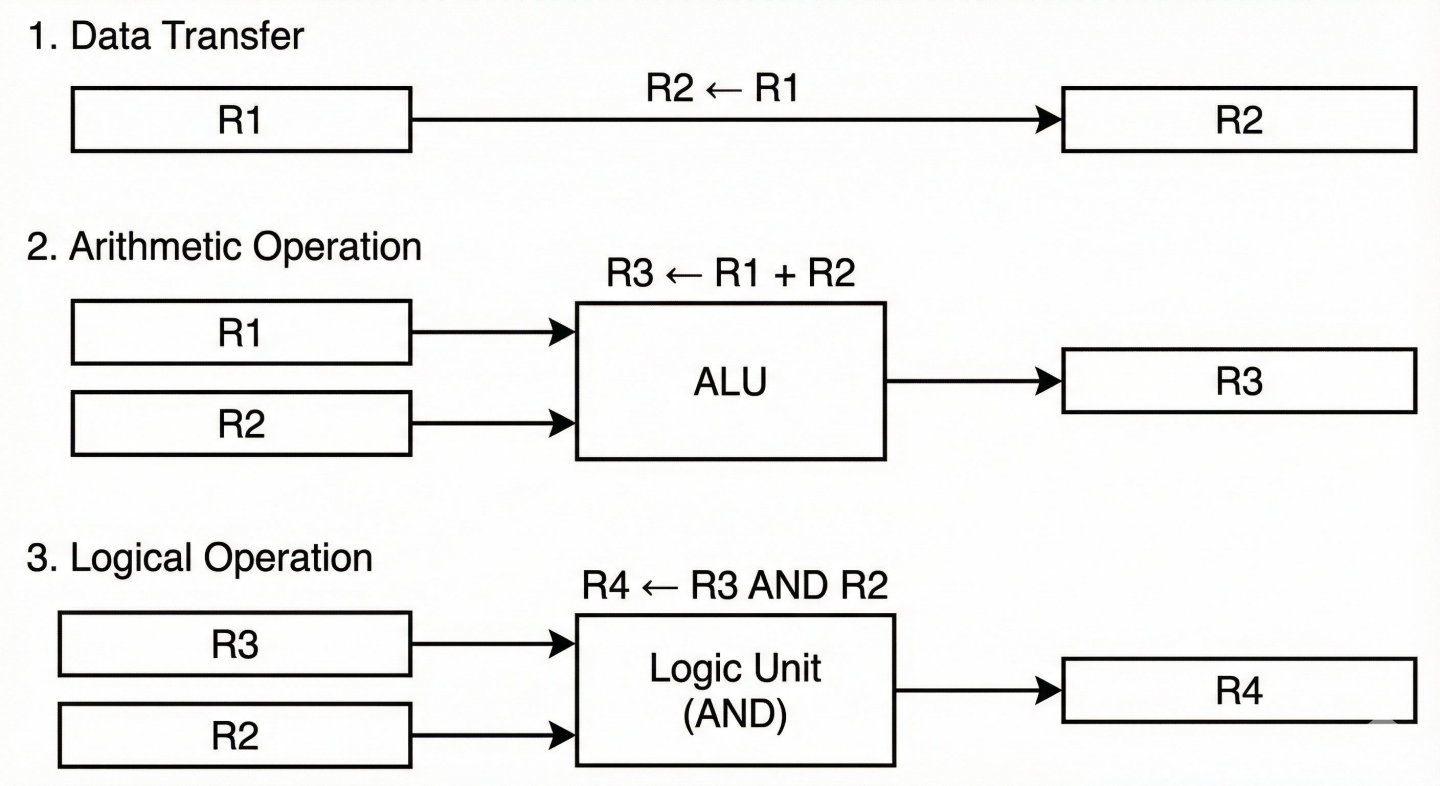
Diagram:
R1, R2 → ALU → R3

(C) Logical Operation

Performs bit-level logical operations via logic circuits.

Example:
R4 ← R3 AND R

Diagram:



OR

Q.4 (a) Why use Error Detection Codes? Name three types and explain any one with example.

Why Error Detection Codes are Used?

When data is transmitted or stored, **errors may occur** due to:

noise
signal distortion
hardware faults

To ensure accuracy, computer systems use **error detection codes** to identify corrupted data.

Thus, **error detection increases reliability** of communication and data integrity.

Types of Error Detection Codes

Three common types are:

1. **Parity Bit**
2. **Checksum**
3. **Cyclic Redundancy Check (CRC)**

(You can write any other like Hamming code, block code etc.)

Explain any one – Parity Bit Method

Definition:

Parity is a simple error detection technique where **one extra bit** (parity bit) is added to data.

Types:

- Even Parity
- Odd Parity

Even Parity Example

Suppose we send 4-bit data:

1011 (three 1s → odd count)

To make total number of 1s **even**, add a parity bit:

1011 1 → transmitted data

Receiver counts bits:

- If total 1s count is even → no error
- If odd → error detected

Q.4 (b) Explain ROM. Describe types of ROM (ROM, PROM, EPROM, EEPROM).

Definition of ROM

ROM stands for **Read Only Memory**.
It is **non-volatile** memory meaning data remains even when power is off.

ROM is used to store:

BIOS
firmware
bootstrap loaders

Contents are fixed at manufacturing or programmed once.

Types of ROM

(i) Masked ROM

- Data written permanently during manufacturing
- Cannot be modified
- Used in fixed firmware devices

(ii) PROM (Programmable ROM)

- ROM chip that can be programmed **once** after manufacturing
- Uses special PROM programmer
- Changes are permanent

(iii) EPROM (Erasable Programmable ROM)

- Can be erased and reprogrammed
- Erased using **ultraviolet light**
- Used in development boards and embedded systems

(Chip has transparent quartz window for UV exposure)

(iv) EEPROM (Electrically Erasable PROM)

- Data can be erased **electrically**
- Rewriting is possible many times
- Used in microcontrollers and BIOS updates

Comparison

ROM Type	Can be Edited?	How erased?
Masked ROM	No	Not possible
PROM	Only once	Not erasable
EPROM	Yes	UV light
EEPROM	Yes multiple times	Electrical signals

Q.5 (a) Explain the concept of Virtual Memory and Cache Memory with Secondary Storage. Discuss their advantages and disadvantages.

Virtual Memory

Meaning / Definition

Virtual memory is a memory management technique used by operating systems to **extend usable memory** beyond the physical RAM capacity.
It uses a portion of secondary storage (like hard disk or SSD) as **logical extension of main memory**.

In simple words:
Even if the computer has less RAM, virtual memory allows running large programs by storing some parts temporarily on disk.

How Virtual Memory Works?

1. Programs are broken into small blocks called **pages**.
2. Only required pages are loaded into RAM.
3. Remaining pages are stored in a **section of disk called Swap Space or Page File**.
4. When a page is needed and not in RAM → **Page Fault occurs** and OS brings it from disk.
5. A less-used page in RAM is moved back to disk.

This gives an impression that system has **more memory than installed RAM**.

Uses / Importance

- Enables large applications to run on small RAM.
- Supports multiprogramming.
- Improves convenience for user by automatic swapping.

Advantages of Virtual Memory

Allows execution of **large size programs**
Increases **multiprogramming capability**
Memory is used efficiently
Users are not limited by physical RAM size

Disadvantages of Virtual Memory

Slower than real RAM because disk access time is high
Can lead to **thrashing** (excessive swapping)
Depends heavily on disk performance

Cache Memory

Meaning / Definition

Cache memory is a **small, high-speed memory** located close to the CPU.
It stores frequently accessed instructions and data to reduce access time.

CPU first checks cache, if found → access is fast
If not found → CPU fetches from RAM (cache miss)

Why Cache Exists? (Need)

There is a huge speed gap between:

- CPU (nanoseconds)
- RAM (tens of nanoseconds)

Cache improves performance by **acting as a high-speed buffer**.

Types of Cache Memory

1. **L1 Cache** – inside CPU core, fastest but smallest
2. **L2 Cache** – slightly bigger and slower
3. **L3 Cache** – shared among cores, larger capacity

Advantages of Cache Memory

Extremely fast access
Reduces CPU waiting time
Increases system performance
Improves instruction execution speed

Disadvantages of Cache Memory

Expensive technology → increases cost
Limited capacity
Complexity of cache design logic

Relationship with Secondary Storage

- **Virtual Memory uses secondary storage** (disk/SSD) to extend RAM.
- **Cache Memory lies between CPU & RAM** to speed up access.
- Secondary storage works as:
    - Storage base for virtual memory
    - Permanent data backup

Together these improve **capacity + speed** of system.

Q.5 (b) What is Semiconductor Memory? Explain its types with advantages and disadvantages.

Definition of Semiconductor Memory

Semiconductor memory refers to memory devices made using semiconductor technology.
They are widely used in computer systems because they offer:

compact size
high speed
low power consumption

Examples include **RAM, ROM, Flash Memory**.

Classification of Semiconductor Memory

Semiconductor memory can be broadly divided as:

(A) Volatile Memory

Definition:

Stores data only while power is ON.
Data is **lost when power is removed**.

Example:

RAM (Random Access Memory)

Types of RAM:

- **Static RAM (SRAM)**
  Fast, costly, used as cache memory
- **Dynamic RAM (DRAM)**
  Cheaper, used as main memory

Advantages:

Very fast read/write
Direct access by CPU
Good for current program execution

Disadvantages:

Temporary storage
Loses data on power failure

(B) Non-Volatile Memory

Definition:

Retains data permanently even after power is removed.

Examples:

ROM, PROM, EPROM, EEPROM, Flash Memory

Uses:

BIOS, firmware, embedded systems, microcontrollers

Advantages:

Data stored permanently
No loss on shutdown
Essential for system booting

Disadvantages:

Slower data writing
Limited reprogram cycles in some types

Extra Classification (for more depth)

Semiconductor memory can also be classified as:

Read Only Memory (ROM family)

- Used for fixed data storage (BIOS, firmware)

Read/Write Memory (RAM family)

- Used during execution of programs

Flash Memory

- Used in pendrives, SSDs, SD cards

Comparison Summary

Feature	Volatile	Non-Volatile
Power effect	Data lost	Data retained
Example	RAM	ROM / Flash
Use Case	Execution	Permanent storage

OR

Q.5 (a) Write a decimal arithmetic unit with BCD adder with example.
Decimal Arithmetic Unit (DAU) Introduction

Decimal Arithmetic Unit is a special hardware unit designed to perform arithmetic operations on **decimal data using Binary Coded Decimal (BCD)** representation.

BCD represents each decimal digit (0–9) using **4-bit binary format**:

Example:
2 = 0010
9 = 1001

DAU is mainly used in:

calculators
digital clocks
financial systems
monetary applications

where **exact decimal accuracy** is required.

BCD Adder Concept

A BCD adder performs **addition of two decimal digits in BCD form**.

Procedure of BCD addition:

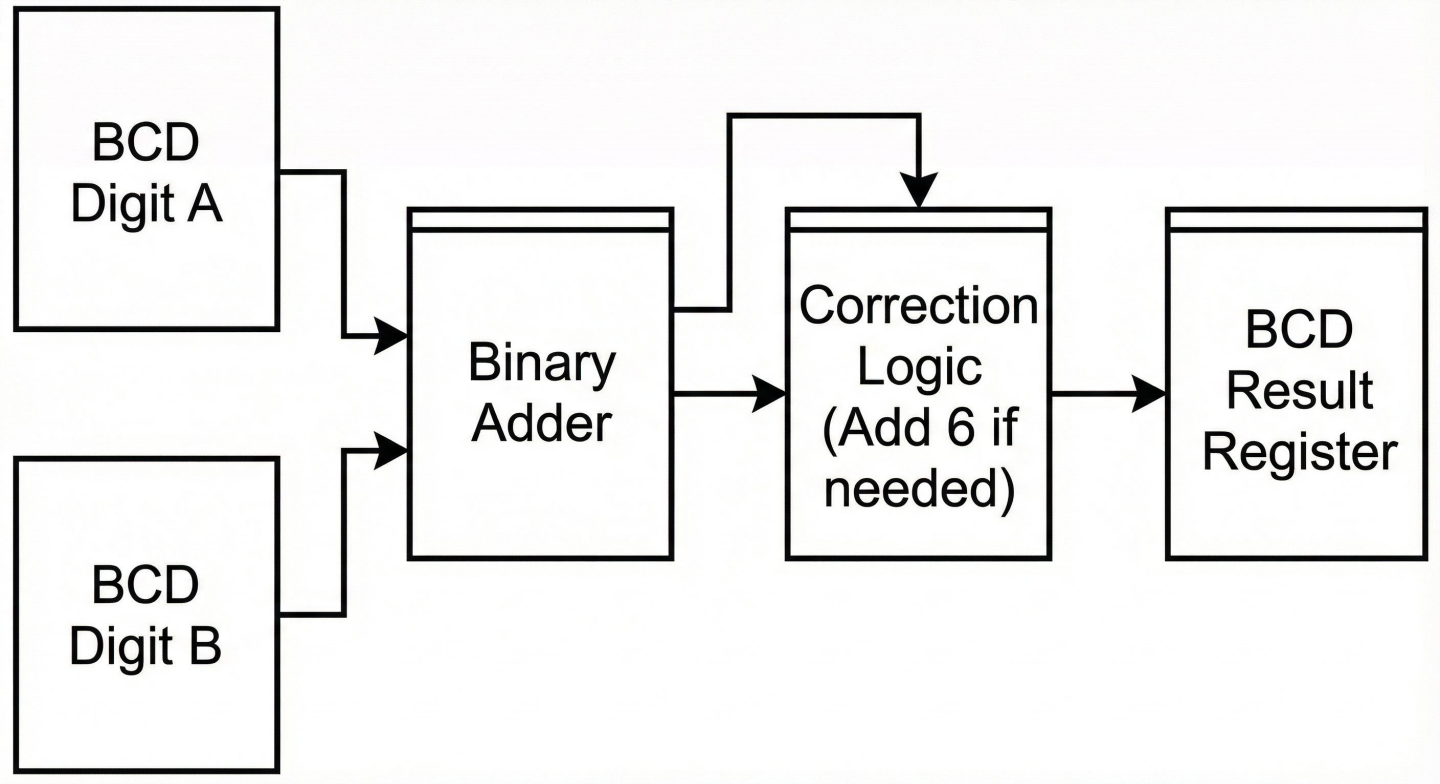
Step 1: Perform normal 4-bit binary addition

Step 2: If binary result > 9 (1001) or carry generated

Add 6 (0110) to correct value

Because in BCD, the valid digits are only 0000 to 1001.

Block Diagram Explanation (Exam Representation)



Blocks must be labelled as:

Binary adder
Correction unit
Output register

This structure ensures that decimal output is valid.

BCD Addition Example

Let us add:

9 (1001) + 5 (0101)

Step 1 — Normal binary addition:

1001 + 0101

-------- 1110 (14 decimal)

Step 2 — Apply correction (add 6)

Because result > 9:

1110 +0110

------ 1 0100

Final digit = 0100 (4)
Carry = 1

Final BCD result = **14 decimal**

This proves how decimal arithmetic unit ensures valid decimal value.

Q.5 (b) What is Memory? Explain Primary and Secondary Memory with advantages and disadvantages.

Meaning of Memory

Memory is a component inside the computer that stores:

data
instructions
intermediate results

It plays a central role in computer operation because CPU continuously requires data to execute programs.

Primary Memory

Definition:

Primary memory is directly accessed by CPU.
It holds data temporarily while processing.

Types:

- RAM (Random Access Memory)
- ROM (Read Only Memory)

Characteristics:

- RAM is volatile (data lost when power off)
- ROM is non–volatile (contents retained)

Advantages of Primary Memory

High access speed
Direct communication with processor
Required for active processing

Disadvantages of Primary Memory

Limited capacity
RAM loses data when power fails

Secondary Memory

Definition:

Secondary memory refers to storage devices used for **long-term data storage**.

Examples:

Hard Disk
SSD
Pen Drive
Optical Disk

Characteristics:

- Non–volatile
- Large capacity

Advantages of Secondary Memory

Stores large amount of data
Retains data without power
Cheaper compared to RAM

Disadvantages of Secondary Memory

Slower than primary memory
Cannot be accessed directly by CPU — data first loads into RAM

Table

Feature	Primary Memory	Secondary Memory
Access	CPU directly accesses	CPU cannot access directly
Speed	Faster	Slower
Storage	Limited	Large
Power Effect	Volatile (RAM)	Non–volatile