24CM401/24CS401/24DS401/24IT401

Hall Ticket Number:

II/IV B. Tech (Regular) DEGREE EXAMINATION

April, 2026 — **Common to CM, CS, DS & IT**

First Semester — **Computer Organization**

Time: Three Hours — **Maximum: 60 Marks**

Answer question 1 compulsorily. **(12X1 = 12 Marks)**

Answer one question from each unit. **(4X12 = 48 Marks)**

COMPUTER ORGANIZATION SCHEME

a) **Define 2's complement.**

2's complement is a method of representing negative binary numbers by taking the 1's complement of a number and adding 1 to it.

b) **Write the** RTL statement **for transferring data from register R1 to R2.**

RTL statement:

R2 ← R1

c) **Define** micro-operation**.**

Micro-operation is a basic operation performed on the data stored in registers.

d) **What is the function of the** Program Counter (PC)**?**

Program Counter (PC) stores the address of the next instruction to be executed.

e) **What is the** accumulator **in a basic computer?**

Accumulator is a register used to store intermediate results of arithmetic and logic operations.

f) **What are the stages of instruction cycle.**

Stages of instruction cycle: Fetch, Decode, and Execute.

g) **What are the** common fields present in an instruction format**?**

Common fields in an instruction format: Opcode, Operand (address), and Addressing mode.

h) **What is** overflow in binary arithmetic**?**

Overflow in binary arithmetic occurs when the result of an operation exceeds the range that can be represented with the given number of bits.

i) **Name any** two addressing modes **used in computers.**

Two addressing modes: Immediate addressing and Direct addressing.

j) **Why is** cache memory used **in computer systems?**

Cache memory is used to store frequently accessed data and instructions to speed up processing.

k) Differentiate interrupt-driven I/O and programmed I/O

Programmed I/O: CPU continuously checks the device status (polling).
Interrupt-driven I/O: Device interrupts the CPU when it is ready, avoiding continuous checking.

l) Define interrupt service routine (ISR).

Interrupt Service Routine (ISR) is a set of instructions executed by the CPU to handle an interrupt.

Unit –I

2 . a) Evaluate different data types used in computer organization. CO1 L2 6M

.

In computer organization, data types define how data is stored, represented, and processed. The main data types are:

1. Integer Data Type

Represents whole numbers (positive and negative).
Stored in binary form (usually using 2's complement).

Example:
+5 → 0101
−5 → 1011 (2's complement form)

2. Floating-Point Data Type

Represents real numbers with fractions using scientific notation (mantissa and exponent).

Example:
3.75 → 1.11×2^2 (binary form)

3. Character Data Type

Represents letters, digits, and symbols using encoding schemes like ASCII.

Example:
'A' → 65 (ASCII) → 01000001
'1' → 49 (ASCII)

4. Boolean Data Type

Represents logical values.

Example:
True → 1
False → 0

Used in conditions like:
IF (A > B) → True/False

5. Fixed-Point Data Type

Represents numbers with a fixed number of digits after the decimal point.

Example:
12.34 stored as 1234 (assuming 2 decimal places)

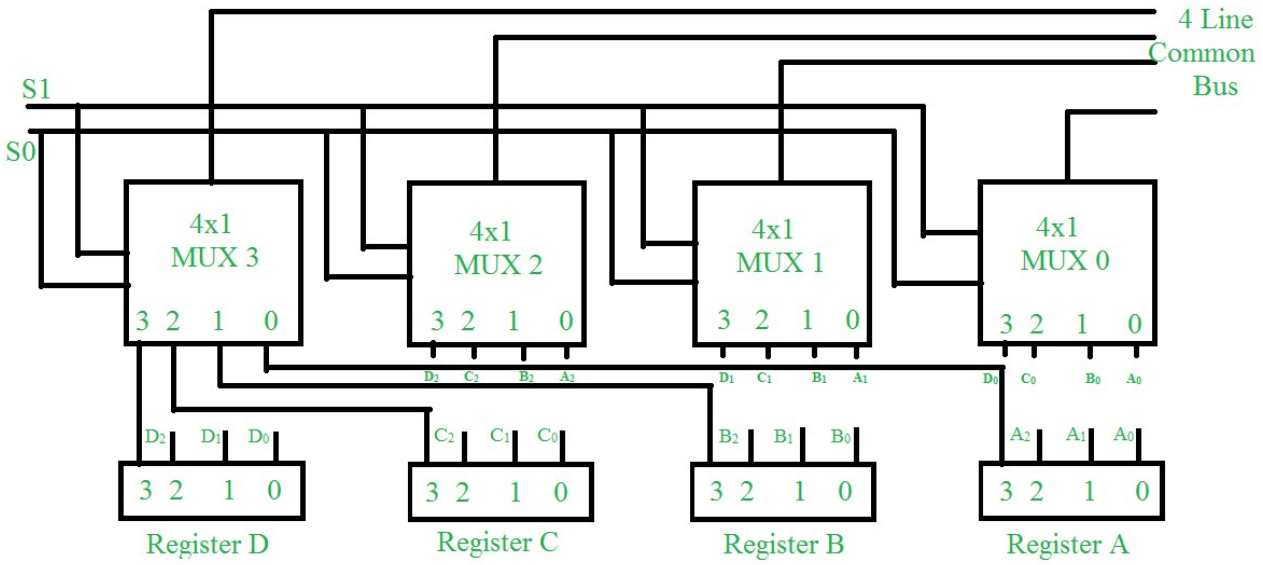
6. Address Data Type

Stores memory locations of data or instructions.

Example:
If a variable is stored at memory location 2000, then
Address = 2000

b) Construct a sequence of register transfer operations for transferring data between registers and memory through a common bus system. **CO1 L4 6M**

Common Bus System (Register–Memory Transfer)



Sequence of Register Transfer Operations

1. Register to Register Transfer

Example: Transfer data from R1 to R2
R2 ← R1
(Data from R1 is placed on the bus and loaded into R2)

2. Register to Memory (Write Operation)

MAR ← Address
MDR ← R1
Memory[MAR] ← MDR

Example: Store R1 into memory location 2000
MAR ← 2000
MDR ← R1
M[MAR] ← MDR

3. Memory to Register (Read Operation)

MAR ← Address
MDR ← Memory[MAR]
R2 ← MDR

Example: Load memory location 2000 into R2
MAR ← 2000
MDR ← M[MAR]
R2 ← MDR

(OR)

3. a) Two registers contain the following values: **CO1 L3 6M**

A = 11010110
B = 10111001
Find the results of the following logic micro-operations:

1. **A AND B**
2. **A OR B**
3. **A XOR B**
4. **NOT A**

Represent the results in binary.

Given:
A = 11010110
B = 10111001

1. A AND B

11010110

& 10111001

10010000

2. A OR B

11010110

| 10111001

11111111

3. A XOR B

11010110

⊕ 10111001

01101111

4. NOT A

A = 11010110

NOT A = 00101001

Final Answers:

- A AND B = 10010000
- A OR B = 11111111
- A XOR B = 01101111
- NOT A = 00101001

b) A computer uses 8-bit fixed-point representation with 4 bits for integer part and 4 bits for fractional part. Represent the following decimal numbers in binary fixed-point format: **CO1 L2 6M**

i) 6.375

ii) 3.625

Given: 8-bit fixed-point → 4 bits integer + 4 bits fractional

i) 6.375

- Integer part: 6 → 0110
- Fractional part: 0.375
  - 0.375 × 2 = 0.75 → 0
  - 0.75 × 2 = 1.5 → 1
  - 0.5 × 2 = 1.0 → 1
  - 0.0 × 2 = 0 → 0

Fraction = .0110

6.375 = 0110.0110

ii) 3.625

- Integer part: 3 → 0011
- Fractional part: 0.625
  - 0.625 × 2 = 1.25 → 1
  - 0.25 × 2 = 0.5 → 0
  - 0.5 × 2 = 1.0 → 1
  - 0.0 × 2 = 0 → 0

Fraction = .1010

3.625 = 0011.1010

Final Answers:

- i) 0110.0110
- ii) 0011.1010

Unit –II

4. a) Analyze the role of various registers in a basic computer organization. CO2 L2 6M

In a basic computer organization, registers are small, high-speed storage units inside the CPU used to hold data, instructions, and addresses during execution.

1. Program Counter (PC)

Holds the address of the next instruction to be executed.
Role: Ensures proper sequence of instruction execution.

2. Instruction Register (IR)

Stores the current instruction being executed.
Role: Provides instruction details for decoding and execution.

3. Memory Address Register (MAR)

Holds the address of the memory location to be accessed.
Role: Acts as an interface between CPU and memory for addressing.

4. Memory Data Register (MDR) / Memory Buffer Register (MBR)

Stores data being transferred to or from memory.
Role: Temporarily holds data during read/write operations.

5. Accumulator (AC)

Stores intermediate arithmetic and logic results.
Role: Central register for ALU operations.

6. General Purpose Registers (R1, R2, …)

Used for temporary data storage during computations.
Role: Improve processing speed by reducing memory access.

7. Stack Pointer (SP)

Holds the address of the top of the stack.
Role: Used in function calls, recursion, and interrupt handling.

b) Construct the address sequencing mechanism in a microprogrammed control unit. **CO2 L2 6M**

Address

Sequencing Mechanism in a Microprogrammed Control Unit

In a microprogrammed control unit, the address sequencing mechanism determines the address of the next microinstruction to be executed.

Basic Components:

1. Control Address Register (CAR)
   Holds the address of the current microinstruction.
2. Control Memory (CM)
   Stores microinstructions.
3. Microinstruction Register (MIR)
   Holds the fetched microinstruction.
4. Sequencing Logic / Next Address Generator
   Determines the next address based on conditions.
5. Multiplexer (MUX)
   Selects the source of the next address.

Block Diagram:

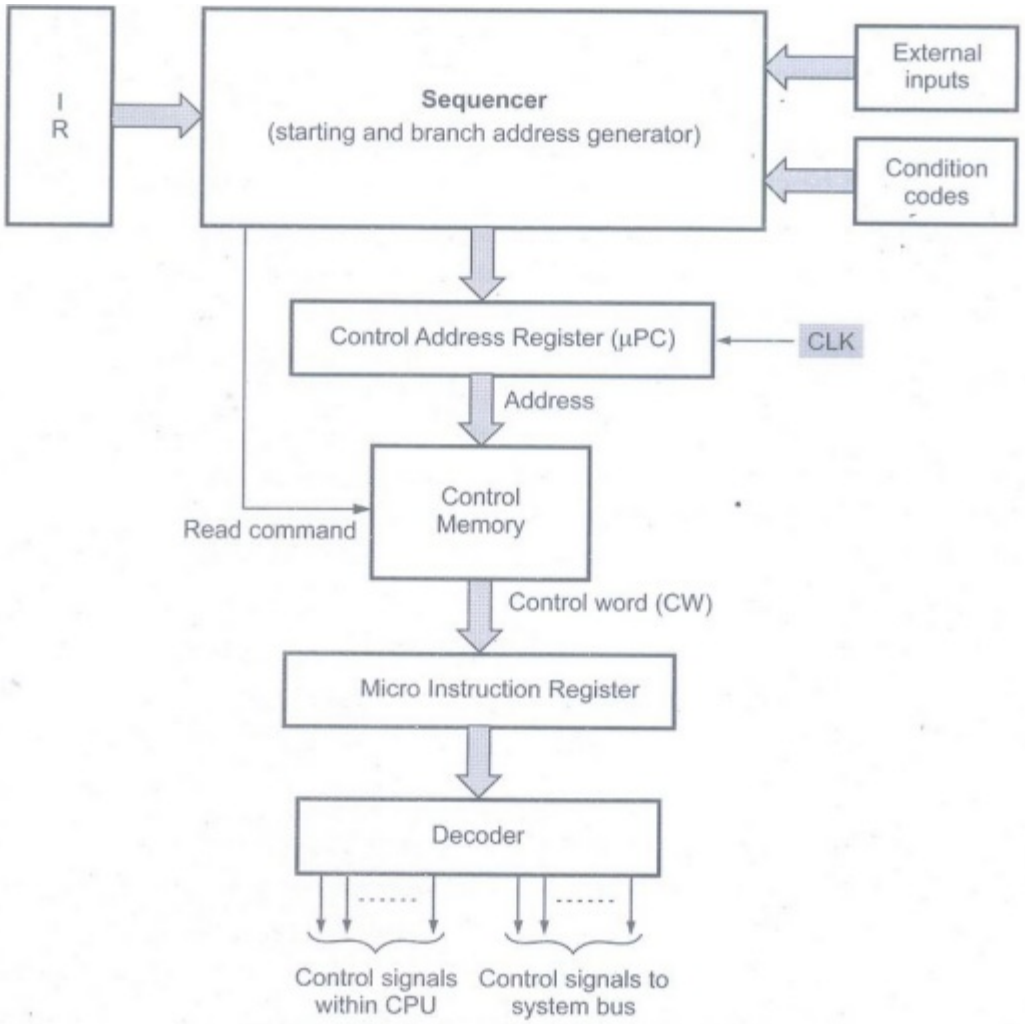


Fig. 7.6.1 Microprogrammed control unit

Working:

1. Fetch:
   CAR sends address to control memory → microinstruction is loaded into MIR.
2. Next Address Selection:
   The next address is determined by:
   - Sequential increment (CAR + 1)
   - Branch address (from instruction field)
   - External inputs/conditions (status flags, interrupts)
   - Mapping from opcode
3. MUX Selection:
   Multiplexer selects one of the above sources.
4. Update CAR:
   Selected address is loaded into CAR for the next cycle.

(OR)

5. a) The program counter (PC) = 425 and the memory location M[425] contains instruction 3050. CO2 L2 6M

During the fetch cycle, determine the contents of the following registers after execution:

- **AR**
- **PC**
- **IR**
- **DR**

Show the sequence of register transfer operations.

Given:
PC = 425
M[425] = 3050

Sequence of Register Transfer Operations (Fetch Cycle):

1. AR ← PC
   AR = 425
2. IR ← M[AR], PC ← PC + 1
   IR = 3050
   PC = 426
3. AR ← IR (address part)
   IR = 3050 → Address part = 050
   AR = 050
4. DR ← M[AR] *(only if needed for further execution, optional in fetch)*

Final Contents of Registers:

- AR = 050
- PC = 426
- IR = 3050
- DR = Not changed / undefined (during fetch cycle)

b) Assume the following register values: **CO2 L3 6M**

- **PC = 450**
- **AR = 700**
- **AC = 25**
- **DR = 10**

Execute the following micro-operations:

1. **AC ← AC + DR**
2. **PC ← PC + 1**
3. **AR ← PC**

Determine the final contents of all registers.

Given initial values:
PC = 450, AR = 700, AC = 25, DR = 10

Step-by-step Execution:

1. AC ← AC + DR
   AC = 25 + 10 = 35
2. PC ← PC + 1
   PC = 450 + 1 = 451
3. AR ← PC
   AR = 451

Final Contents of Registers:

- AC = 35
- PC = 451
- AR = 451
- DR = 10 (unchanged)

Unit –III

6. a) Explain about three, two, one and zero address instruction formats. CO3 L2 6M

Instruction formats are classified based on the number of address fields present in the instruction.

1. Three-Address Instruction

Contains three address fields: two source operands and one destination.

Format:
OP A, B, C → A = B op C

Example:
ADD R1, R2, R3 → R1 = R2 + R3

Advantage: Fewer instructions required
Disadvantage: Instruction length is large

2. Two-Address Instruction

Contains two address fields: one acts as both source and destination.

Format:
OP A, B → A = A op B

Example:
ADD R1, R2 → R1 = R1 + R2

Advantage: Moderate instruction length
Disadvantage: One operand is overwritten

3. One-Address Instruction

Contains one address field; other operand is implied (usually accumulator).

Format:
OP A → AC = AC op A

Example:
ADD X → AC = AC + M[X]

Advantage: Shorter instruction
Disadvantage: More instructions needed

4. Zero-Address Instruction

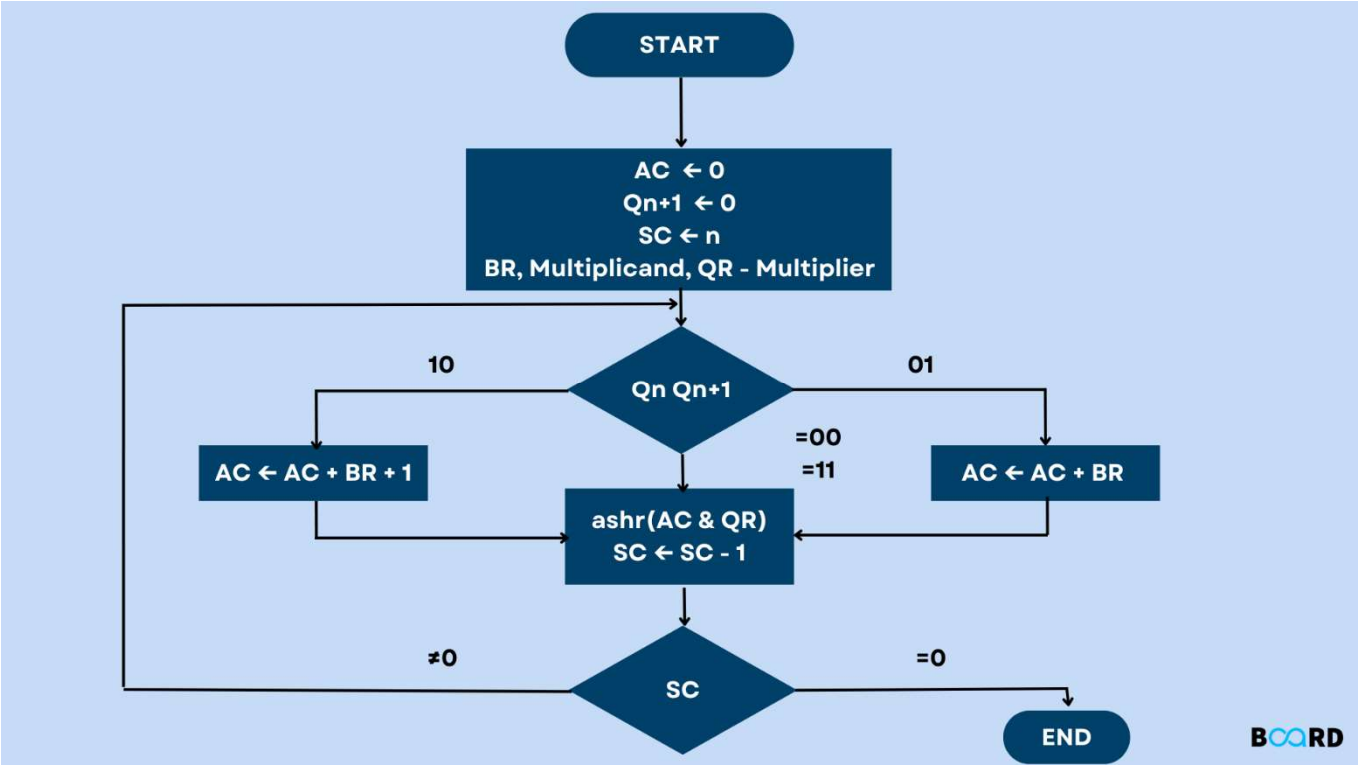
Contains no address field; operands are implied (stack-based).

Format:
OP

Example:
ADD → Top two stack elements are added

Advantage: Very short instruction format
Disadvantage: Requires stack organization

b) Explain booth's algorithm with an example. CO3 L2 6M



Booth's Algorithm

Booth's algorithm is used to perform signed binary multiplication (in 2's complement form). It reduces the number of additions and subtractions by encoding the multiplier.

Registers Used:

- M → Multiplicand
- Q → Multiplier
- A → Accumulator (initially 0)
- Q_{-1} → Extra bit (initially 0)
- n → Number of bits

Rules:

Check the pair (Q_0 , Q_{-1}):

- 00 → No operation
- 11 → No operation
- 01 → A = A + M
- 10 → A = A − M

After operation → Perform Arithmetic Right Shift of (A, Q, Q_{-1})

Repeat for n times

Example: Multiply (5 × 3)

Take 4-bit representation:
M = 0101 (+5)
Q = 0011 (+3)
A = 0000, Q_{-1} = 0, n = 4

Step-wise Execution:

Step	Q_0	Q_{-1}	Operation	A	Q	Q_{-1}
Init	-		Initial	0000	0011	0
1	1	0	A = A − M	1011	0011	0
			Shift	1101	1001	1
2	1	1	No operation	1101	1001	1
			Shift	1110	1100	1
3	0	1	A = A + M	0011	1100	1
			Shift	0001	1110	0
4	0	0	No operation	0001	1110	0
			Shift	0000	1111	0

Final Result:

(A, Q) = 0000 1111 = 15 (Decimal)

(OR)

7. a) Perform restoring division method on 14 ÷ 6. Consider Accumulator A as 5 bits, Register Q as 4 bits, dividend as 4 bits and divisor as 5 bits **CO3 L2 6M**

i) Find out the value of accumulator A at the end of the first step, i.e. when SC=3?

ii) Determine the value of accumulator A and register Q at the end of the second step, i.e. when SC= 2?

Given:
Dividend = 14 → Q = 1110 (4 bits)
Divisor = 6 → M = 00110 (5 bits)
Accumulator A = 00000 (5 bits)
SC = 4

Step 1 (SC = 4 → after 1st iteration, SC = 3)

Left shift (A, Q):
A Q = 00000 1110 → 00001 1100

So, A = 00001

Subtract M:
A = 00001 − 00110 = 11011 (negative)

Since negative → restore A
A = 11011 + 00110 = 00001
$Q_0 = 0$

Answer (i): A = 00001

Step 2 (SC = 3 → after 2nd iteration, SC = 2)

Left shift (A, Q):
A Q = 00001 1100 → 00011 1000

So, A = 00011

Subtract M:
A = 00011 − 00110 = 11101 (negative)

Since negative → restore A
A = 11101 + 00110 = 00011
$Q_0 = 0$

Answer (ii):
A = 00011
Q = 1000

Final Answers:

i) A = 00001 (at SC = 3)
ii) A = 00011, Q = 1000 (at SC = 2)

b) Write down the sequence of steps required to perform the following operation using single cycle data path architecture. CO3 L2 6M
Add R1, R2, (R3)

Instruction:

ADD R1, R2, (R3)
Meaning: R1 ← R2 + M[R3]

Sequence of Steps (Single Cycle Data Path):

1. Fetch Instruction:
   IR ← M[PC]
   PC ← PC + 1
2. Decode Instruction:
   Identify R1, R2, R3
3. Read Registers:
   A ← R2
   MAR ← R3
4. Memory Access:
   MDR ← M[MAR]
5. Execute Operation:
   ALU performs addition
   Result = A + MDR
6. Write Back:
   R1 ← Result

Summary:

- Operand 1 → R2
- Operand 2 → Memory location pointed by R3
- Result stored in → R1

Final Operation:

R1 ← R2 + M[R3]

Unit –IV

8.	**a)**	**A user program with hundreds of instruction is executing on a processor in your computer. Here some of the instructions are found in the TLB due to frequent access of the instructions. Assume that the processor uses TLB to access the on-going instruction. 40% is the hit ratio for the instruction in TLB. Processor takes 50ns to search the instruction in TLB. When the instruction is not found in the TLB processor takes 100ns to access the main memory. Determine the effective main memory access time.**	**CO4**	**L3**	**6M**

Given:

- TLB hit ratio (h) = 0.4
- TLB access time = 50 ns
- Main memory access time = 100 ns

Formula:

Effective Access Time (EAT) =
h × (TLB time) + (1 − h) × (TLB time + Memory time)

Calculation:

EAT = 0.4 × 50 + 0.6 × (50 + 100)
EAT = 20 + 0.6 × 150
EAT = 20 + 90
EAT = 110 ns

Final Answer:

Effective Main Memory Access Time = 110 ns

	b)	**Consider there are different cache memories in three different computers. These cache memories are directed mapped cache, fully associate cache and two-way set associative cache. You have to assume that all these cache memories have four one-word blocks. Processor is requesting the following block addresses from the cache memory.** **2,4,6,8,2,6,4,4,8.** **Find out the number of misses for the above three different cache organizations.**	**CO4**	**L2**	**6M**

Given:
Cache size = 4 blocks (1 word each)
Block sequence: 2, 4, 6, 8, 2, 6, 4, 4, 8

1. Direct Mapped Cache

Mapping: Block number mod 4

Address	Cache Index	Hit/Miss
2	2	Miss
4	0	Miss
6	2	Miss (replaces 2)
8	0	Miss (replaces 4)
2	2	Miss (replaces 6)
6	2	Miss (replaces 2)
4	0	Miss (replaces 8)
4	0	Hit
8	0	Miss

Total Misses = 8

2. Fully Associative Cache

(Any block can go anywhere, use FIFO)

Address	Cache Content	Hit/Miss
2	2	Miss
4	2,4	Miss
6	2,4,6	Miss
8	2,4,6,8	Miss
2	-	Hit
6	-	Hit
4	-	Hit
4	-	Hit

Address Cache Content Hit/Miss

8 - Hit

Total Misses = 4

3. Two-Way Set Associative Cache

- 4 blocks → 2 sets, each with 2 blocks
- Mapping: Block mod 2

Set Mapping:

- Set 0 → even blocks
- Set 1 → odd blocks (not used here)

All blocks go to Set 0

Address Set 0 Content Hit/Miss

2 2 Miss

4 2,4 Miss

6 4,6 Miss (2 removed)

8 6,8 Miss (4 removed)

2 8,2 Miss (6 removed)

6 2,6 Miss (8 removed)

4 6,4 Miss (2 removed)

4 - Hit

8 4,8 Miss

Total Misses = 8

Final Answers:

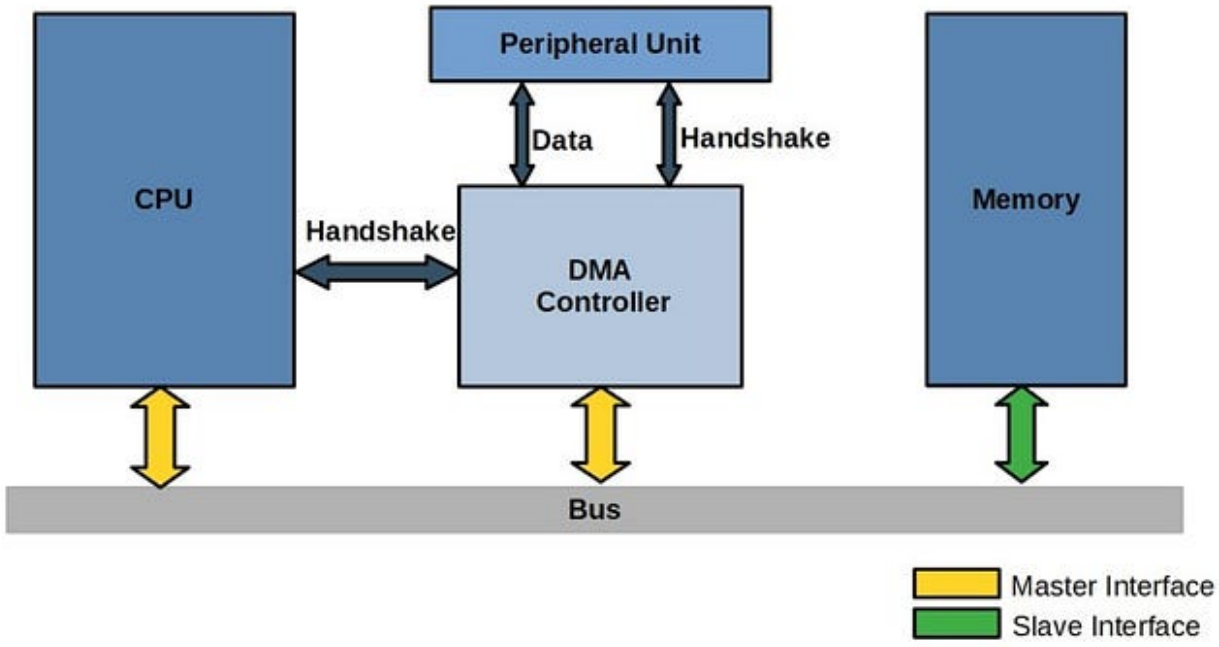
- Direct Mapped Cache Misses = 8
- Fully Associative Cache Misses = 4
- Two-Way Set Associative Cache Misses = 8

9. a) With neat diagram discuss how DMA access memory directly. CO4 L2 6M

Direct Memory Access (DMA)

DMA allows an I/O device to transfer data directly to/from memory without continuous CPU involvement, improving system efficiency.

Neat Diagram:



Working Steps:

1. Initialization:
   CPU provides starting address, transfer size, and direction to DMA controller.
2. Bus Request:
   DMA requests control of the system bus from CPU.
3. Bus Grant:
   CPU grants control and releases the bus.
4. Data Transfer:
   DMA transfers data directly between memory and I/O device (bypassing CPU).

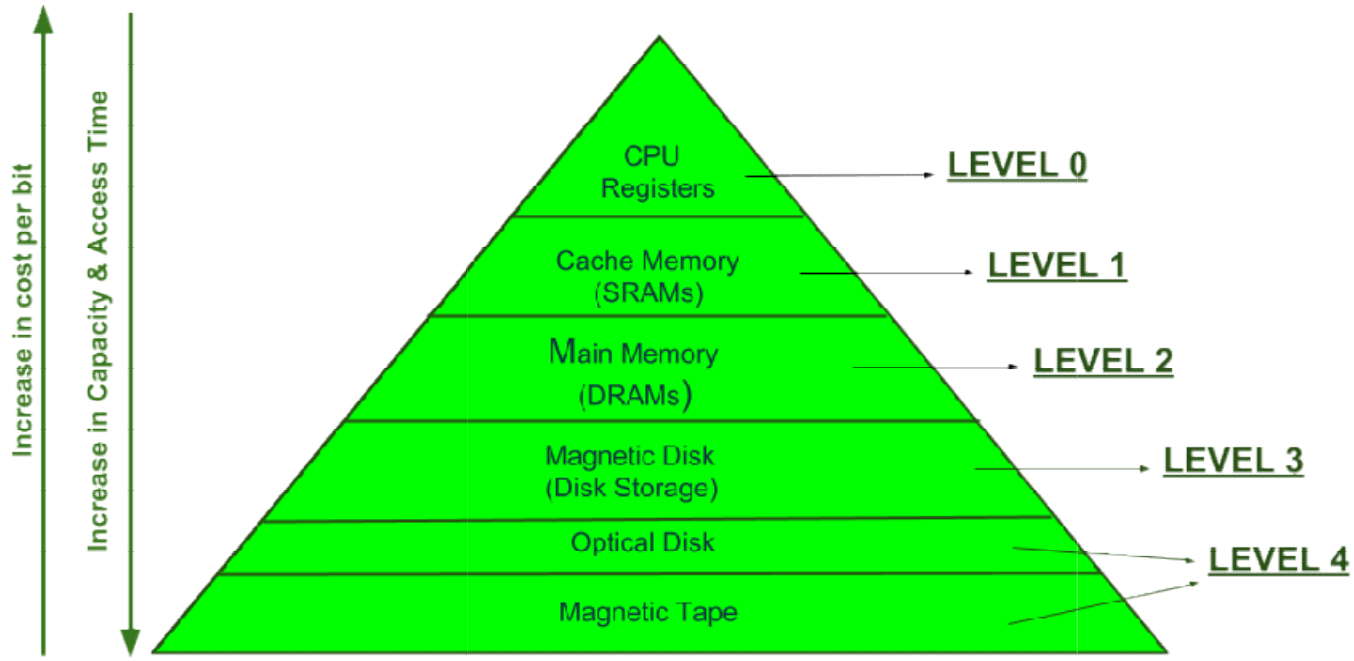
5. Completion:
   DMA sends an interrupt to CPU after transfer is complete.

(b) Explain memory hierarchy in detail. CO4 L2 6M

Memory Hierarchy

Memory hierarchy is the organization of different types of memory in a computer system based on speed, cost, and capacity to achieve efficient performance.

Neat Diagram:



MEMORY HIERARCHY DESIGN

Levels of Memory Hierarchy:

1. Registers

- Located inside CPU
- Very high speed
- Stores immediate data for processing
  Example: Accumulator, Program Counter

2. Cache Memory

- Small, fast memory between CPU and main memory
- Stores frequently used data
  Example: L1, L2, L3 cache

3. Main Memory (RAM)

- Stores programs and data currently in use
- Slower than cache but larger in size

4. Secondary Memory

- Non-volatile storage
- Large capacity but slower
  Example: Hard Disk, SSD

5. Tertiary Memory

- Used for backup and archival
- Very large capacity, very slow
  Example: Magnetic tapes

Characteristics:

Level	Speed	Cost	Capacity
Registers	High	High	Low
Cache	High	High	Low
Main Memory	Medium	Medium	Medium
Secondary	Low	Low	High

Working Principle:

- Frequently used data is kept in faster memory (top levels)
- Less frequently used data is stored in lower levels
- Based on locality of reference (temporal & spatial)