COA MAY 2023 Answers

Q1

a) Discuss any five arithmetic instructions of 8086 with examples.

1. ADD (Addition)

- **Purpose**: Adds two operands and stores the result in the destination operand.
- **Syntax**: `ADD destination, source`
- **Example**:

```assembly
MOV AX, 05H    ; Load AX with 05H
ADD AX, 03H    ; Add 03H to AX (AX = AX + 03H)
```

After execution, `AX` will contain `08H`.

2. SUB (Subtraction)

- **Purpose**: Subtracts the source operand from the destination operand and stores the result in the destination.
- **Syntax**: `SUB destination, source`
- **Example**:

```assembly
MOV BX, 0AH    ; Load BX with 0AH
SUB BX, 04H    ; Subtract 04H from BX (BX = BX - 04H)
```

After execution, `BX` will contain `06H`.

3. MUL (Multiplication)

- **Purpose**: Performs unsigned multiplication of the accumulator (AX or AL) with the source operand.
- **Syntax**: `MUL source`
- **Example**:

```assembly
MOV AL, 05H     ; Load AL with 05H
MOV BL, 03H     ; Load BL with 03H
MUL BL          ; Multiply AL by BL (AX = AL * BL)
```

After execution, `AX` will contain `000FH` (15 in decimal).

4. DIV (Division)

- **Purpose**: Performs unsigned division of the accumulator (AX or DX:AX) by the source operand.
- **Syntax**: `DIV source`
- **Example**:

```assembly
MOV AX, 000FH   ; Load AX with 000FH (15 in decimal)
MOV BL, 03H     ; Load BL with 03H
DIV BL          ; Divide AX by BL (AL = AX / BL, AH = AX % BL)
```

After execution, `AL` will contain `05H` (quotient) and `AH` will contain `00H` (remainder).

5. INC (Increment)

- **Purpose**: Increments the value of the specified operand by 1.
- **Syntax**: `INC destination`
- **Example**:

```assembly
MOV CX, 07H     ; Load CX with 07H
INC CX          ; Increment CX by 1 (CX = CX + 1)
```

After execution, `CX` will contain `08H`.

d) Explain half adder and full adder with diagram.

Half Adder

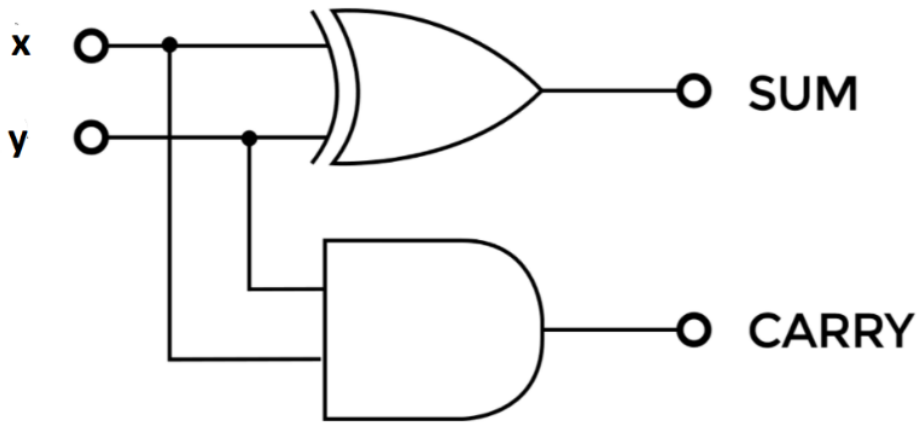
A half adder is a simple digital circuit that performs the addition of two single-bit binary numbers. It has two inputs (A and B) and two outputs (Sum and Carry). The half adder can add two bits and provide the sum and carry output.

Truth Table for Half Adder:

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic Equations for Half Adder:

- Sum = A XOR B
- Carry = A AND B



Full Adder

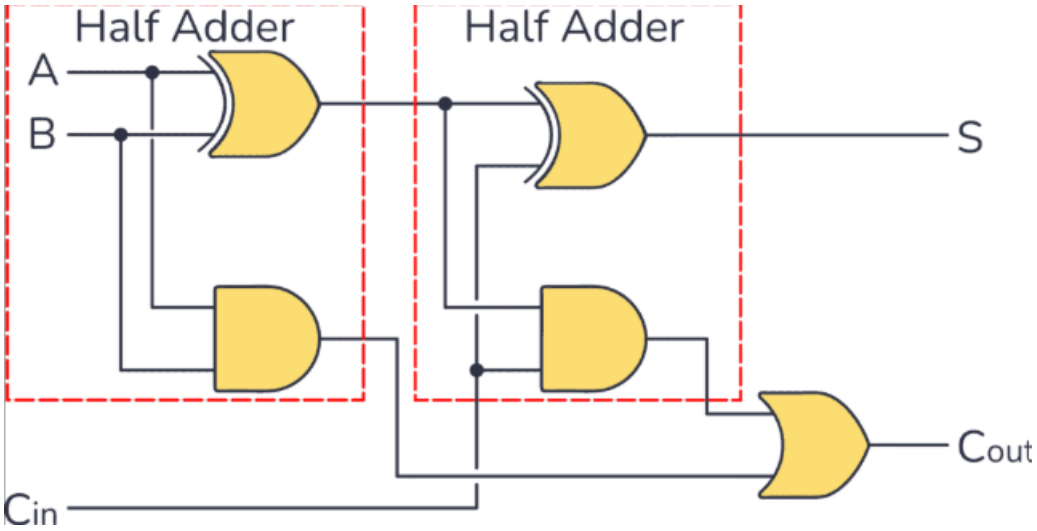
A full adder is a more complex digital circuit that performs the addition of three single-bit binary numbers: two significant bits (A and B) and an input carry bit (Cin). It has three inputs (A, B, and Cin) and two outputs (Sum and Carry). The full adder can add two bits along with an input carry from a previous addition.

Truth Table for Full Adder:

A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Logic Equations for Full Adder:

- Sum = (A XOR B) XOR Cin
- Carry = (A AND B) OR (Cin AND (A XOR B))



Q2

b) Discuss various cache memory mapping techniques with advantages and disadvantages of it.

Cache memory mapping techniques determine how data from the main memory is placed into the cache memory for faster access by the CPU. There are **three main mapping techniques**: **Direct Mapping**, **Associative Mapping**, and **Set-Associative Mapping**.

1. Direct Mapping

- **Description:** In direct mapping, each block of main memory is mapped to exactly one cache line.

- **Mechanism:**
  - The main memory address is divided into three parts: **Tag**, **Cache Line Number**, and **Block Offset**.
  - The Cache Line Number determines the line in the cache where the block will be stored.

- **Advantages:**
  - Simple implementation.
  - Requires less hardware (low cost).

- **Disadvantages:**
  - **Collision Problem:** If two blocks map to the same cache line, the previous data is overwritten, leading to frequent replacements (conflicts).

- **Example:** Suppose a system has 8 cache lines and 32 memory blocks. Each memory block can only map to one specific cache line (e.g., Block 0, 8, 16 map to Line 0).

2. Associative Mapping

- **Description:** In associative mapping, any block of main memory can be stored in any line of the cache memory.

- **Mechanism:**
  - The main memory address is divided into two parts: **Tag** and **Block Offset**.
  - The cache stores the tag of the memory block along with the data.
  - When accessing data, the entire cache is searched to find the matching tag.

- **Advantages:**
  - No restriction on which cache line a block can occupy, solving collision problems.

- **Disadvantages:**
  - More complex hardware and higher cost due to the need for a fully associative search.
  - Slower access compared to direct mapping because of the overhead of tag searching.

- **Example:** Any of the 32 memory blocks can be stored in any of the 8 cache lines.

3. Set-Associative Mapping

- **Description:** A compromise between direct and associative mapping. The cache is divided into sets, and each memory block is mapped to a specific set but can occupy any line within that set.

- **Mechanism:**
  - The main memory address is divided into three parts: **Tag**, **Set Number**, and **Block Offset**.
  - The Set Number determines the set in the cache where the block will be stored.

- **Advantages:**
  - Reduces collision problems compared to direct mapping.
  - Lower hardware complexity compared to associative mapping.

- **Disadvantages:**
  - Slightly more complex hardware compared to direct mapping.

- **Example:** Suppose the cache has 4 sets and each set has 2 cache lines. Memory blocks are assigned to sets, but they can occupy any line within their assigned set.

Q3

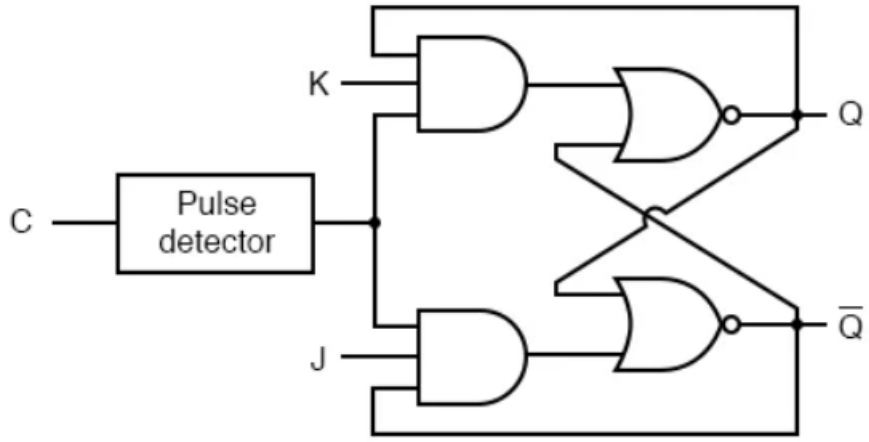
b) Explain JK flip with diagram and Compare SR and JK flip flop.

A JK flip-flop is a versatile and widely used type of flip-flop in digital electronics. It is an improvement over the SR flip-flop by eliminating the "invalid" state where both inputs are high. The JK flip-flop has two inputs (J and K) and two outputs (Q and Q bar), as well as a clock input that synchronizes its operation.

Key Characteristics:

- **J (Set):** When J is high, the flip-flop sets the output Q.
- **K (Reset):** When K is high, the flip-flop resets the output Q.
- **Clock (CLK):** The flip-flop's outputs change state on the triggering edge (rising or falling) of the clock pulse.
- **Toggle:** When both J and K are high, the flip-flop toggles its output.

Circuit Diagram:



Characteristics table and Truth table:

CP	J	K	Q	Q_{+1}	State
1	0	0	0	0	NO CHANGE
1	0	0	1	1	
1	0	1	0	0	RESET
1	0	1	1	0	
1	1	0	0	1	SET
1	1	0	1	1	
1	1	1	0	1	TOGGLES
1	1	1	1	0	

Truth Table of JK flip flop

CP	J	K	Q_{+1}	State
1	0	0	Q_n	NO CHANGE
1	0	1	0	RESET
1	1	0	1	SET
1	1	1	$\overline{Q}_n$	TOGGLES

Simplified Truth Table of JK flip flop

Aspect	SR Flip-Flop	JK Flip-Flop
Inputs	S (Set), R (Reset)	J (Set), K (Reset)
Outputs	Q (stored value), $\bar{Q}$ (complement of Q)	Q (stored value), $\bar{Q}$ (complement of Q)
Functionality	- Set (S=1, R=0): Sets Q to 1.	- J=1, K=0: Sets Q to 1.
	- Reset (S=0, R=1): Resets Q to 0.	- J=0, K=1: Resets Q to 0.
	- Invalid state (S=1, R=1): Indeterminate output	- J=1, K=1: Toggles Q (switches Q to opposite value).
Clock Dependency	Operates on clock input	Operates on clock input
Invalid Condition	When S=1 and R=1 (undefined state)	No invalid condition (always well-defined).
Toggle Functionality	Not available	Available (J=1, K=1 toggles Q).
Applications	Basic storage, simple control systems	Used in counters, registers, and advanced systems.
Complexity	Simpler design	More complex due to toggle functionality.

Q4

a) Write 8086 Assembly Language Program to count the number of 0's and 1's in given 8-bit numbers.

Code:

```
.model small
.stack 100h
.data
num db 13h
ones db 0
zeros db 0
.code
main proc
mov ax, @data
mov ds, ax
mov al, num
mov cx, 08h
back: ror al, 1
jnc zerinc
inc ones
jmp next
zerinc: inc zeros
next: dec cx
jnz back
mov ah, 4ch
int 21h
main endp
end main
```

b) Discuss concept of DMA and its various data transfer techniques

What is DMA?

- **DMA** stands for **Direct Memory Access**.
- It allows **peripheral devices** (like hard disks, sound cards) to **directly transfer data to/from memory without CPU involvement**.
- The **DMA controller (DMAC)** manages the data transfer.

Why DMA is Needed?

- To **reduce CPU load** during large data transfers.
- To **speed up** input/output operations.
- CPU only initiates the transfer and is **free for other tasks** during the actual data movement.

How DMA Works:

1. CPU gives control to the **DMA controller**.
2. DMA controller **transfers data** between memory and I/O device.
3. After the transfer, DMA **informs CPU** via an **interrupt**.

Various DMA Data Transfer Techniques:

1. Burst Mode (Block Transfer Mode)

- DMA controller **transfers a block of data** continuously without releasing control of the bus.
- CPU is **locked** (cannot access memory) during the transfer.

Use: Fast transfer of large blocks (e.g., disk data).

2. Cycle Stealing Mode

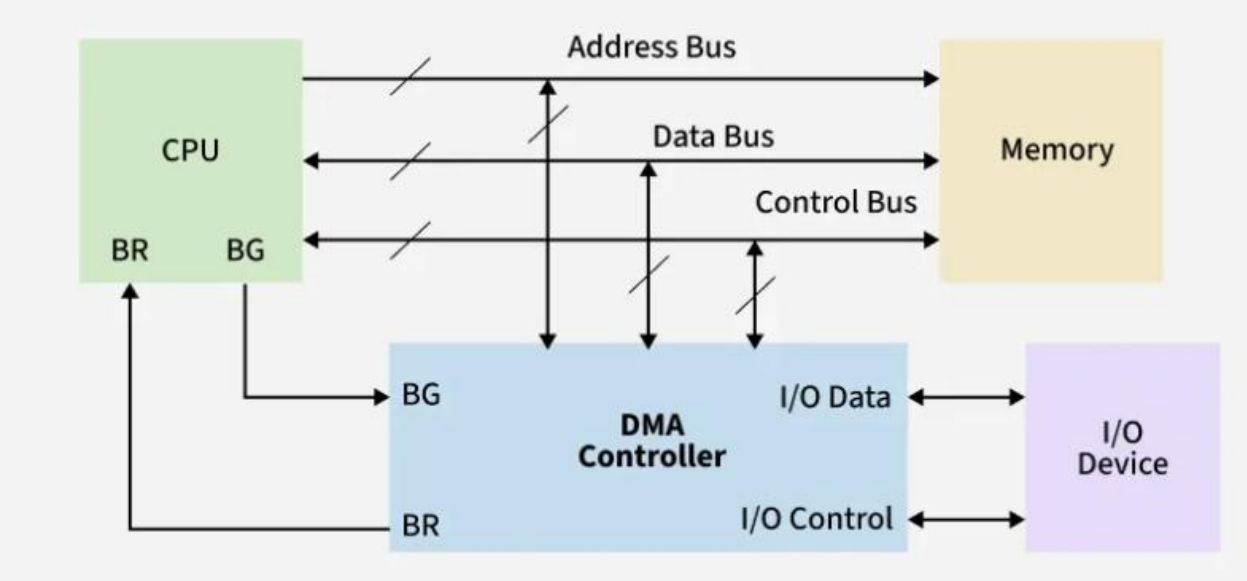
- DMA controller **steals** one memory cycle at a time.
- CPU and DMA **share** the bus.
- CPU experiences **small delays**, but can continue working.

Use: When CPU must remain active while DMA works.

3. Transparent Mode

- DMA controller transfers data **only when CPU is not using the bus**.
- **No noticeable delay** for CPU.
- Data transfer is **slower** compared to other modes.

Use: When CPU operation must not be disturbed at all.



Q6

a) Write short note on decoder and encoder.

Decoder:
A **decoder** is a combinational circuit that converts **binary input** signals into a unique output. It is used in systems where information needs to be **decoded** from one format to another.

Example Applications:
- **Memory Address Decoding**: Identifies the specific memory location.
- **Binary to Decimal Conversion**: Used in display systems (e.g., seven-segment displays).

Encoder:
An **encoder** performs the opposite function of a decoder—it takes multiple inputs and converts them into a smaller binary representation. Encoders are useful in data compression and communication systems.

Example Applications:
- **Priority Encoders**: Assign priority to inputs and generate a binary output.
- **Data Compression**: Used in multimedia encoding formats (e.g., MP3, JPEG).