Section-A: Answer All Questions

1. **Define Program Counter:**
   The Program Counter (PC) is a register in the CPU that holds the address of the next instruction to be executed.

2. **What are the primary registers present in CPU?**
   Primary CPU registers include the Program Counter (PC), Accumulator (ACC), Instruction Register (IR), Memory Address Register (MAR), and Memory Buffer Register (MBR).

3. **What does an instruction contain?**
   An instruction typically contains an opcode (operation code) and operands (data or addresses used in the operation).

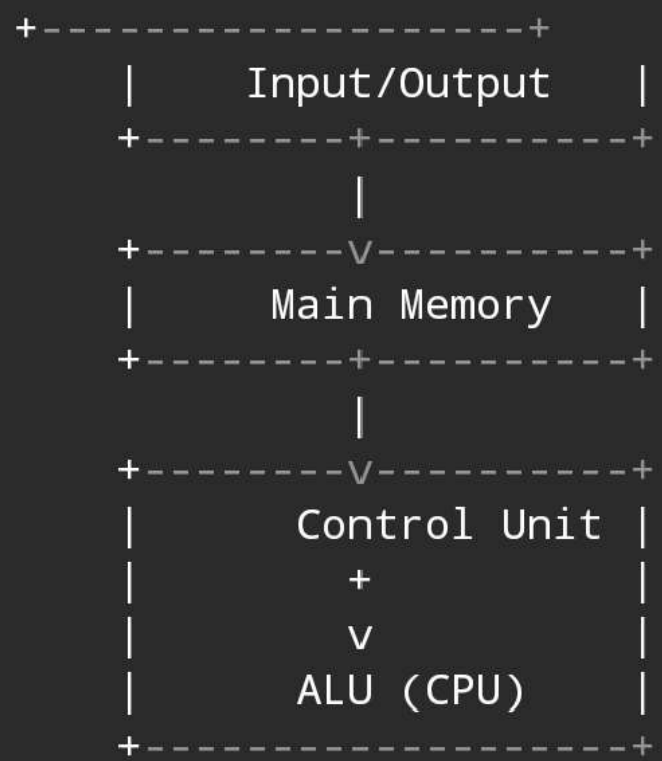
4. **Why is MAR connection with memory unidirectional?**
   The MAR (Memory Address Register) sends addresses to memory, but does not receive data back, hence the connection is unidirectional.

Section-B: Answer All Questions

1. **Describe with diagram Von Neumann Architecture:**
   The Von Neumann Architecture consists of a single memory space for both instructions and data. It includes a CPU (with ALU, CU, registers), memory, and I/O units connected via buses.

   Diagram:

   Copy code



   Instructions are fetched from memory, decoded, and executed sequentially.

2. **Differentiate Big Endian System and Little Endian System:**

Feature	**Big Endian**	**Little Endian**
Byte Order	Most significant byte stored first	Least significant byte stored first
Example (0x1234)	0x12 stored at lower address	0x34 stored at lower address
Usage	Used in some network protocols	Common in Intel processors

Section-C: Answer All Questions

1. **Discuss different types of instruction formats with example:**
    - **Zero-address:** Used in stack-based architectures. Example: `ADD` (operates on top two stack elements).
    - **One-address:** Uses accumulator and one operand. Example:

```
LOAD A
ADD B
STORE C
```

    - **Two-address:** Specifies two operands. Example: `ADD A, B` (A = A + B).
    - **Three-address:** Uses three operands. Example: `ADD A, B, C` (A = B + C).

2. **List the instructions using one-address instruction for X = (A * B) + (C * D):**
   Assuming accumulator-based one-address instructions:

Copy code

```
LOAD A       ; ACC = A
MUL B        ; ACC = A * B
STORE TEMP1  ; TEMP1 = A * B

LOAD C       ; ACC = C
MUL D        ; ACC = C * D
STORE TEMP2  ; TEMP2 = C * D

LOAD TEMP1   ; ACC = A * B
ADD TEMP2    ; ACC = (A * B) + (C * D)
STORE X      ; X = (A * B) + (C * D)
```