Fetch-Decode-Execute Cycle Questions

Fetch, Decode, and Execute Cycle Exercises

Exercise 1: Basic Arithmetic Operations

Objective: Write an assembly program that adds two numbers and stores the result in memory.
Instructions:

- Load the first number into register `AL`.
- Load the second number into register `BL`.
- Add the contents of `AL` and `BL`, storing the result back in `AL`.
- Store the result from `AL` into memory location `C0`.

Questions:

- How does the program counter (PC) change during each instruction fetch?
- Identify the decode and execute phases for each instruction.

Exercise 2: Data Movement and Memory Addressing

Objective: Write a program that moves a string "HELLO" into consecutive memory locations starting from `C0`.
ASCII Codes:

- 'H' → 48 (Hex) → 72 (Decimal)
- 'E' → 45 (Hex) → 69 (Decimal)
- 'L' → 4C (Hex) → 76 (Decimal)
- 'O' → 4F (Hex) → 79 (Decimal)

Instructions:

- Load the ASCII value of 'H' into `AL` and store it at memory location `C0`.
- Repeat the process for 'E', 'L', 'L', and 'O', storing each subsequent character in the next memory location.

Questions:

- How does the fetch-decode-execute cycle process the `MOV` instruction for memory operations?
- What changes occur in the instruction pointer (IP) during each instruction?

Exercise 3: Implementing a Delay Loop

Objective: Develop a delay loop using a simple counting method.
Instructions:

- Load a register (e.g., `CL`) with a specific value to set the delay duration.
- Create a loop that decrements `CL` until it reaches zero.
- Ensure the loop provides a noticeable delay before proceeding to the next instruction.

Questions:

- During the loop, how does the fetch-decode-execute cycle manage the decrement and conditional jump instructions?
- How does the microprocessor determine when to exit the loop based on the zero flag?

Exercise 4: Handling Input from a Simulated Keyboard

Objective: Write a program that reads a character from the simulated keyboard and displays it on the screen.
Instructions:

- Wait for a key press and read the character into register `AL`.
- Store the character from `AL` into the video display memory at location `C0`.

Questions:

- How does the fetch-decode-execute cycle handle input/output instructions?
- What role does the instruction pointer play when accessing hardware ports?