Project Name: Build a Virtual CPU Emulator

Week 06: I/O Operations

Objective: To understand and execute all basic input and output (I/O) operations in computing which enable interaction between the user and the computer: either regarding entering data (input) or result display (output).

1. Input Operations:
   - Read data from user or external sources, for instance, a keyboard, or a file.
   - Example: by scanf() in c or by input() in Python.

2. Output Operations:
   - Display or store processed data (such as in a screen, or file).
   - Example: by printf in c and by print() in python.

3. Applications:
   - Designing user-interactive applications.
   - Data logging and retrieval from files external to the system.

4. Common Input Output Devices:
   - Keyboard, mouse (input).
   - Monitor, printer(output).

Through these means, all software applications facilitate appropriate and efficient I/O operations in user interaction and data management.

1. **Implement simulated I/O devices (keyboard, display).**
   The program simulates two I/O devices:

   1.1 **Keyboard:** Simulated using an input_buffer list. Inputs are added to the buffer using the read_input() method, mimicking keyboard input.

   1.2 **Display:** Simulated using an output_buffer list. Outputs are written to this buffer using the write_output() method, which retrieves data from the CPU's memory and displays it.

2. **Create I/O instructions and integrate them with the CPU.**
   I/O instructions are defined within the execute_instruction() method:

   2.1 **IN Instruction:** Reads data from the input_buffer and stores it in the CPU's memory at the specified address.

2.2 **OUT Instruction:** Retrieves data from a specified memory address and writes it to the output_buffer.

These instructions enable interaction between the CPU and the simulated I/O devices.

3. **Test with I/O-intensive programs.**

A test program is implemented in the run_test_program() method:

3.1 Simulated inputs are added to the input_buffer using read_input().

3.2 The IN instruction stores the inputs in memory.

3.3 The program performs an addition operation and stores the result in memory.

3.4 The OUT instruction retrieves the result from memory and writes it to the output_buffer.

This process demonstrates how the CPU handles multiple I/O operations and ensures functionality under I/O-intensive conditions.