Memory Management

Objective:
To implement memory management for the virtual CPU by setting up a simulated memory space, performing read/write operations, and handling address mapping with segmentation.

Key Concepts

1. **Simulated Memory Space:**
   A contiguous block of memory allocated in the program to represent the CPU's memory.
2. **Memory Segmentation:**
   Dividing the memory into logical segments (e.g., Code, Data, Stack), each with defined size and boundaries, for structured memory access.
3. **Logical to Physical Address Mapping:**
   Translating logical addresses (relative to a segment) into physical addresses (absolute within memory) to perform operations.
4. **Memory Operations:**
   - **Write:** Store a value at a specific logical address in a defined segment.
   - **Read:** Retrieve a value from a specific logical address in a defined segment.

Algorithm

1. Initialize Memory:

- Input: Total memory size and segment definitions (name and size).
- Process:
  1. Create a memory array of the given total size.
  2. Divide memory into segments with base and limit addresses based on segment sizes.
  3. Validate that total segment sizes do not exceed memory size.
- Output: Memory structure with segments defined.

2. Write Operation:

- Input: Segment name, logical address, value to write.
- Process:
  1. Retrieve the base and limit of the specified segment.
  2. Calculate the physical address as the sum of the base address and the logical address.

3. Check if the physical address exceeds the segment limit.
4. Write the value to the calculated physical address.

- Output: Value stored in the memory array.

3. Read Operation:

- Input: Segment name, logical address.
- Process:
    1. Retrieve the base and limit of the specified segment.
    2. Calculate the physical address as the sum of the base address and the logical address.
    3. Check if the physical address exceeds the segment limit.
    4. Retrieve the value from the calculated physical address.
- Output: Value retrieved from the memory array.

4. Display Memory:

- Input: None.
- Process:
    1. Return the entire memory array for debugging or analysis.
- Output: Full memory state.

Sample Execution

1. **Input:**
    - Create memory with a size of 100.
    - Define segments:
        - **Code:** 30 units.
        - **Data:** 40 units.
        - **Stack:** 30 units.
    - Write 42 to logical address 15 in the "Data" segment.
    - Read value from logical address 15 in the "Data" segment.
2. **Process:**
    - Memory is initialized with three segments.
    - The "Data" segment starts at physical address 30.

    - Logical address 15 maps to physical address 45 in memory.
    - Value 42 is written to address 45.
    - The value is retrieved from address 45.

3. **Output:**
    - **Write Operation:** Value 42 stored in "Data" segment at logical address 15.
    - **Read Operation:** Value retrieved is 42.

Conclusion

This week’s implementation achieves memory management for the virtual CPU by efficiently segmenting memory, enabling logical-to-physical address translation, and performing essential read/write operations. This functionality lays the foundation for handling complex memory interactions in later stages of the project.