Project Name: Build a Virtual CPU Emulator

Week 05: Memory Management

Objective: The development of strategies and memory organization required for a virtual CPU is concerned with memory management. These processes or threads running in the virtual environment need this memory management.

- It is responsible for memory allocation and deallocation to processes efficiently.
- It provides the necessary support for virtual memory and paging as needed.
- Prevention of Memory Leak and data isolation.
- Implement First Fit, best-fit, or Worst-fit memory allocation policies.

1. **Set up a simulated memory space**

   In computing science, memory simulation provides one way to visualize and manipulate memory in a controlled and virtual environment. The concept is important for understanding how memory allocation, addressing, and operations work in real computing systems. Here, we offer a simple implementation of a simulated memory space in Python. The program initializes memory, performs the basic operations, and displays contents for debugging.

   Key Features

   1. **Initialization**:
      - 1.1. Memory is represented as a list of zeros.
      - 1.2. The size parameter defines the total number of memory slots available.
   2. **Debugging Tool**:
      - 2.1. Display memory provides a clear view of all memory slots and their values for debugging.

2. **Implement memory read/write operations**

   Key Features

   1. **Read Operation**:
      - 1.1. The read(address) method retrieves the data stored at a given memory address.
      - 1.2. Validate the address to ensure it lies within the bounds of memory size.
   2. **Write Operation**:
      - 2.1. The write (address, data) method stores data at a specified memory address.
      - 2.2. Ensures that only valid addresses are written to, preventing out-of-bounds errors.
   3. **Error Handling**:
      - 3.1. Both methods raise a Value Error if an invalid memory address is accessed.

3. **Handle address mapping and memory segmentation**
   Key Features
   1. **Memory Segmentation**:
      1.1. The memory is divided into three distinct segments:
         1.1.1. **Code Segment**: Holds program instructions.
         1.1.2. **Stack Segment**: Used for function calls and local variables.
         1.1.3. **Heap Segment**: Allocated for dynamic memory.
   2. **Address Mapping**:
      2.1. Logical addresses provided within a segment are mapped to physical addresses in the actual memory.
   3. **Error Handling**:
      3.1. Ensures segment sizes do not exceed total memory.
      3.2. Verifies that logical addresses stay within the bounds of their respective segments.