This project simulates a simplified CPU with an MMU, OS, memory, and peripherals to model the fetch, decode and execute cycle of the CPU.

1. CPU (Central Processing Unit)

- Fetches instructions from memory.

- Decodes the instruction

- Executes the instruction fetched from memory.

- Communicates with the MMU for memory access.

- Supports basic arithmetic and logical operations.

- Contains registers for temporary data storage.

- Program Counter, which points to the next instruction, in the code section in memory.

- Accumulator, which the logical operations work on.

- Instruction Register, holds the instruction in use.

- Memory Address Register, which holds the address for the memory.

- Memory Data Register, which hold either the instruction or data from memory.

2. MMU (Memory Management Unit)

- Translates virtual addresses to physical addresses.

- Reads from memory.

- Writes to memory.

1. OS (Operating System)

- The entity of that govern the other entities, like cpu, memory, scheduler, process controller.

- Initialize the required components:

- CPU

- Memory

- ProcessTable

- CPU Controller

- Scheduler

- FreeList

- Start/Stop/Pause/Resumes the simulation.

- Schedules processes.

1. Memory (RAM)

- Stores code, stack and heap, of various processes.

- Accessible by the CPU via the MMU.

5. Scheduler

- Manages the process queue, for which to run.

6. Process Controller

- Create a new process.

- Control the allocation of memory space.

- Store instructions

-

MMU functions:

func (mmu \*MMU) TranslateAddress(virtualAddr uint32) (int, error) {}

Takes in a virtual address, and returns the physical address given after a translation. This physical address is used to access the physical memory.

func (mmu \*MMU) Read(physicalAddr uint32) (int, error) {}

Takes in a physical address, reads from memory at that address, and returns the stored value at this address.

func (mmu \*MMU) Write(physicalAddr uint32, value uint32) error {}

Takes in a physical address and a uint32 value, goes to that memory address, and replaces the currently stored value, with the new one.