James McCaffrey

I received some help from Professor Daniels and used his implementation of bitpack.

I believe everything has been correctly implemented

There aren't really any significant changes from my design, rather some small changes like removing some redundant functions.

The architecture of my solution is as follows:

main.rs: uses the other modules to read in the um program file and extract information from each 32 bit word and call the right instruction function with the values pulled from the word.

memory.rs: this is for storing and manipulating the um's memory in the form of a Vec<Vec<u32>>. Each subvector is a memory segment made up of 32 bit words. There is also the unmapped memory vector Vec<u32> that functions more like a stack, that holds what segments are currently unmapped.

instructions.rs: this is for executing each instruction and holding the cpu register values in an array of u32s of length 8. The main just calls each instruction function and passes through the memory and register object so they can be manipulated.

It would take around 1 seconds for my implementation to execute 50 million instructions running on release mode. I noticed that output causes quite the slowdown, so I wrote this test that just moves around some memory and registers

```
for x in 0..50000{
        asm(load_v(0, 6)); // 6 into register 0
    asm(load_v(2, 80)); // 80 into register 2
    asm(load_v(3, 5)); // 5 into register 3
    asm(load_v(4, 1)); // 1 into register 4

    asm(map(1, 0)); //map 6 from reg a. 1 is pointer
    asm(store(4, 3, 2)); //stores value 80 in mem[1][5]
    asm(load(5, 4, 3)); //stores value mem[1][5] into register 5

    asm(unmap(1)); //unmap previous segment 1 is now null pointer
    asm(map(6, 0)); //map another segment of length 6 and pointer in reg 6
    asm(store(4, 3, 3)); //stores value 5 into mem[1][5]
    asm(load(7, 4, 3)); //stores mem[1][5] into register 7
    }
    asm(halt());
}
```

Using this, I got 550001 instructions in 10.23ms which is about 1 second per 50 million instructions.

Spent about 2 hours analyzing the assignment

Spent about 4 hours preparing the design

Spent about 8 hours actually implementing RUM