Y86EMULATOR

Implementation

Each instruction and operators have their own function. There is a while loop that that does through the file and checks each directive and adds it to memory depending on which directive it's currently in. Once that's done, it moves onto the execute function which is just basically a big switch statement that invokes the appropriate instruction and operator instruction. For the jump instructions, just directly used their rules with the flags. There is also a help flag which prints out the usage of the program in yellow like so ./y86emul –h

Challenges

Getting the cmpl() and movspl functions to work. Figured out that cmpl doesn't necessarily stored anything in memory, it just sets flags. Figuring out how the file defined functions work. I did by making small programs and printing their outputs. Many other challenges was also encountered.

Big-O Analysis

The space it takes up is the space that is allocated from the .size directive. This means that the size of the program is determined at execution time.

For running time, there is a loop that first goes through the entire program and checks whether it has the .text and .size directives and if there are multiple of them. If they are then the program exits with an error, if not then the file pointer returns to the top of the program, and begins space allocation and the fetch, decode, and execute process.

MakeFile

The makefile compiles the y86emul.c and y86emul.h files with the –Wall and –lm flags.