Sangini Shah (sms591)

Programming Assignment 4- `y86emul`
Description of Program and Runtime Analysis

Emulator

This emulator creates an y86 image which is an unsigned char array that contains all the data and instructions involved in executing this program. Initially the program begins with reading the individual directives and storing them onto the image. Errors are thrown if the directives are not found or are incorrectly formed. As the text directive is approached, each byte is converted to an integer and stored onto the stack. This way each instruction can be accessed easily. Then the program runs through the fetch decode and execute cycle, reading each instruction, translating it, performing the action, updating the registers, and updating the program counter.

There were many difficulties approached when working on this assignment. Initially, I was having trouble determining how I would be storing each value onto the image, or how I should represent the data. Once that was overcome, it was a matter of interpreting the bytes in the proper manner, or checking that everything was updated properly. I kept on making the mistake of taking only a byte from the stack and storing it into a long, causing some incorrect mathematical computations that required aggressive debugging to solve. Then there was the confusion of the read and write methods where I was not aware of what to be outputting. In the end I decided that readb will read a hex value from 00 - ff, readl will read in a long decimal value, writeb will print out a character, and writel will print out a long decimal value. The program also prints out an overflow message if the OF flag is thrown.

However, before I could solve these problems, I wanted to test if the registers were being loaded properly and the computations were being done correctly and so I wrote prog3.y86 which simply stores two numbers, 10 and 2, into two registers, adds the two together, and prints out the sum which is stored in memory on the image.

Overall this assignment was very educational but also extremely time consuming. It allowed me to better understand how the fetch decode and execute cycle works as well as how data is stored in memory/registers and how important size casting is.

Disassembler

Because of my trouble with writing the emulator, I was not able to begin writing the disassembler.