Y86 Emulator: Read Me
Michael Russo
Mr880

y86emul.c:

- The implementation of the fetch decode execute cycle is a main feature of the program.
- We start the program with several error check and messages and for the help command (-h). Checking for…
  - Help Command
  - Empty Files
- Once the file is retrieved and error checked and line by line, using c functions, getting the lines separated in order to store them into an array where we can begin directing the array for types.
  - .size
  - .string
  - .long
  - .byte
  - .text
- We error check these for non multiple uses and store the info into memory (emem). Branching off into their selective types, we determined which info to store into memory by way of global types.
- Once we are set up, we run the program using counters and executing instructions we are able to figure out what values to stop the processes.
- We used integer arrays instead of char arrays. This allows us to save characters of the .text file, saving every two characters into a single place. This leads into the use of a function that can get a certain byte of memory from integer into a hex value, giving access to the two characters. These values lead our instructions.

- We fetch by reading the 4 byte value from memory two chars at a time converting the final result to an int via little endian persuasion.
- Many error checks are in place throughout the program.
- Bit shifting (32 bits) via shiftTheBits() is used.
- Jump instructions were performed during our arithmetic operations keeping in mind the correct flags were initiated using our flag array.
- Push and pop functions were instituted as functions as well.

Test Cases:

- prog1.y86:

Message was correctly printed out ending with a Halt instruction.

• prog2.y86:

Program calculated factorials after promoting the user to “Enter an integer>>”.