Mark Hirons mch165 167008833

Y86emul

In order to complete this assignment I split up the task of the program into two parts.

First, I parsed through the y86 input files and populated a char array of size dictated by the .size directive. I then stored the data from each other directive in the char array at indices calculated by converting the hex addresses into decimal and then using those values as offsets of 0. The text instructions were converted two characters at a time into a coded byte, string characters and bytes were stored at single indices, and 32-bit long ints were split into bytes, filling 4 indices of the char array each.

Second, now that the data is stored, I execute the instructions. I did this by having a program counter variable that stored the index of the first byte of the first instruction in the char array. The first byte is decoded to determine what instruction it is, and then using a state machine of functions and function pointers that correlated to a list of enum return codes, the appropriate functions corresponding to the instructions was run. Each function that emulated a y86 instruction decoded the remainder of the current instruction, performed the instruction and then set the program counter to the proper address.

The state machine runs until a function returns a INS, ADR, or HLT code which is printed out at termination. AOK tells the state machine to continue.