Mohit Mori (mam987)
CS 211: Computer Architecture
Assignment 4: Y86 Emulation

y86emul:

My y86emul program interprets and executes instructions from ".y86" files. The program first starts by allocating memory for the y86 program. Then, my program reads in all of the directives, storing the instructions and various values appropriately in the correct place in the memory space of the program. After every value is loaded into memory, my program then executes. Execution is done with a while loop that loops until the HALT instruction is reached or until there is an invalid instruction or address. The while loops contains an if/else if statement for every possible instruction, and each if/else if block carries out the respective instruction and then gets the next instruction before looping around.

Overall, this program was very frustrating and difficult, however that ultimately lead to a greater reward upon its completion. One of the initial, but quickly overcome challenges I faced was the conversion of values from hex to binary, from little endian to big endian, from two's complement binary to decimal, etc. As I progressed further in the assignment, one of the biggest challenges I faced was bug testing in itself. As such a large program with a lot of potential places to go wrong, bug testing was intense, tiresome, and difficult. Additionally, calling/returning and pushing/popping instructions provided me with a challenge as I did not initially understand how these instructions work, but this obstacle was overcome with a quick Google search.

y86dis:

My y86dis program interprets machine instructions from a ".y86" file like y86emul does, but instead of executing the instructions, y86dis converts them and displays them as assembly instructions. This is done in a very similar manner to y86emul. First, everything is loaded into memory, then a while loops goes through every instruction. But, instead of doing the functions of the instructions, the assembly instruction equivalent is printed to console as well as to a file called "assembly.txt".

This program was not very challenging at all because I did this program after finishing y86emul. So, all of the conversion/other necessary methods were pulled from my y86emul program. The disassemble function in y86dis was made almost exactly like the execute function in y86emul. All I had to do was remove the parts of each if/else if block that interacts with the memory and replace it with a printf and an fprintf statement, for printing to console and file respectively. Independently, this program would have been definitely more challenging, but after y86emul I was able to do it with minimal effort/obstacles.