Adit Patel

Program:
This program was one of the most difficult assignments in this course. We were required to write the equivalent C code for the y86 emulator. I approached this program by following the y86 instructions that were given to us for this assignment and writing the functions accordingly. I went about by creating functions for each of the instructions based on groups i.e. I put all of this in a switch case, and once i got an overflow message. I put all the transfers involving registers in one function, a separate function for all the operations (add, subtract, xor, and, multiply and compare)a function for all the jump statements, separate functions for push, pop, return, call ,movsbl and the read and write instructions. I also created a function to input the data into the memory block based on the corresponding directives. The main challenges that I faced were getting the instructions to run smoothly and without any errors as it was difficult to check the functionality of the code without writing the entire program. My program works correctly for both prog.1 and prog.2. Prog.1 prints out the entire messages (congrulations) that it required to print according to the instructions and the program ends with the status getting set to HLT and an appropriate message being printed out. Prog.2 asks the user to input an integer and finds the corresponding factorial of the number. And ends when ctrl+d is entered and an appropriate message is outputted with status getting set to HLT. My program does all the required instructions for the y86 emulator including reading and writing as well as storing all the instructions in the memory block and accessing it based on the program counter. Running time of my program was O(n). As the program needs to complete all the instructions in the text file to executed properly and this would take O(n) time.

Algorithm:
My program gets the text file from the terminal and first checks if a valid file has been entered. If no errors are present, it sends the file into another method where it gets separated based on the directives that are present in the file and the instructions in the directives are stored in the memory block in the corresponding address. Each instruction is sent to the decode function where based on the switch statement, it sends the instruction to the require y86 instruction function. Within the functions the memory block is access and the method is executed. In the functions the flags as well as the program counter are constantly updated. The

registers array is the array that store the value of each register throughout the program and this is also updated according to the function. If there is either an address error or an instruction error, the status which is of type enum is updated to either INS or ADR with an appropriate message getting otherwise it remains in AOK which means the no errors have been encountered. If the program is executed correctly without any error, the status is set to HLT and an appropriate message is printed with the program coming to an end. This was the algorithm upon which my code is based on. All the functions are based on the y86 instructions that was given to us and code is written in the order of those instructions.