David Alummoottil ReadMe – PA4 Computer Architecture

Overview: This program is a Y-86 emulator, which is a simplified version of the more complex X-86 system run on many systems. There are many components and parts to this problem and I have done the best I could to replicate a Y-86 emulator. The emulator has registers and flags similar to the X-86, but only have three flags; the zero, sign, and overflow flags.

Running Time: The program ran in O(n) since it needed to go through the whole file once and compiled it all together. Many functions ran at this time in order to perform the operations given.

Program: There were many parts that needed to be executed in this program to execute the programs correctly. First, a file must be read properly with many parts in the file. The file must always have a text and size directive, but other directives can be given as well. The text directive is basically the instructions that will be carried out later in the program. After getting the directives one by one, the memory block array created is filled up accordingly, whether they are bytes, strings, or any of the other directives given. After the memory array has been filled with the directives given, the text must be decoded. The program counter starts at the beginning of the array and gradually moves up carrying out the instructions as it goes. First, you must fetch the bytes that represent the instructions, and depending on if status is AOK or not, continue with the byte given and move to decode the byte. The 1-6 byte long instructions then must be decoded. The first byte, the opcode byte, specifics what specific operation must be performed. The 0-5 other bytes after specify the registers, values, or displacements necessary to carry out the instruction. The decoding process determines the operation that must be executed and from there the program will execute the

operation. In my program, I combined the fetch, decode, and parts of the execution process into one function called “executeMemory”. I split the operations up with a switch statement and from performed many of the operations within the function as well. Each operation altered various parts of the program including the flags, registers, memory, values, status of the program, and program counter. On top of that, they also printed, scanned, and performed their respective operations with the registers and memory.

This program was the most difficult to create from all my years of programming so far, and it had to do with the multiple facets that needed to be dealt with and properly taken care of. The most difficult part to manage was the memory and how things were moving around while also making sure the program counter worked properly.