Test Plan

DifficultWare, CSS 422 A

This document contains a description of how we tested our program at each phase and the techniques we leveraged.

Input/Output

First we wanted to determine if we were able to correctly read data from anywhere in memory. To test this, we loaded a simple x68 program into memory and pointed our disassembler at it. Then, we iterated through that memory and confirmed that it was what we were expecting to see. Next we needed to receive that data's starting and ending addresses from a file. Using a configuration file that contained those addresses, we implemented some code that would read the file and store them in address registers. By manually stepping through our disassembler and checking that the correct addresses were loading into memory, we were able to observe that the input/output portion of our project was functioning correctly.

Operation Codes

We started our project by creating a master-list of all the possible OP codes that we'd need to disassemble (listed in TestAll.x68). By analyzing that list we discovered a pattern where almost all of the instructions could be grouped by their first four bits and then determined by the next four bits. Our first test of the OP codes consisted of tracing through our disassembler to make sure we branched into the group with the correct first four bits. Next our test was to keep tracing and see if we moved into the portion of code for one specific instruction. At this point, we could guarantee the instruction to disassemble and move on in that process. As long as we get there we can deduce that each preceding step was working as expected.

Effective Addresses

The tests for the effective addresses were the most involved for us. We created a test file with the instructions our disassembler could handle, ran the disassembler, and manually compared our program output to what was in the file. This method was very effective because it quickly showed us the bugs in our code. After we detected each bug, we would fix it, run it, check it, then move on to the next instruction and if that resulted in a bug, repeat. By doing this we were eventually able to identify and resolve (hopefully) all of the bugs in our project, which led to a functional disassembler.