**4. Test Plan:**1~2 pages.This is a description of how you tested the program. It should also contain a description of your team’s coding standards. If you have, please include your testing files as well.

From the very beginning we started building the test code. As a consequence we learned a great deal about the various addressing modes and operation semantics of the easy68k as well as 6800 assembly language. The first week working on the project we mapped out the strategies we would use to test our disassembler. The first one to write was an easy choice, being the most familiar with the move command we entered all of the valid addressing modes that were required for our project. Once we knew the valid modes it was a process of using the 68k manual to get a sense of what was a valid command and the various opcodes and effective address modes for each. This process gave us valuable experience that we would need later on to debug and write more complex subroutines. We proceeded in this manner, writing a new test instruction and testing it by studying the list file memory output to read the instruction and convert it into binary to check if we had the correct instruction based on the opcode, opmode, and/or source and effective address.

Our testing was only as robust as was required. Maybe that’s just how it started in the beginning but we didn’t refactor or try to test the limits beyond the requirements.

Our coding standards defined a clear description of what variables and or registers would be used at the header of each method. We also had a clear expectations that the least significant 12 bits would be in registers D2, D3, D4, D5 from lowest register D2, holding the first 3 LSBs.