**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. 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 a 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 a valid command was 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. We took a general approach of specifying all the things that we would like the instruction to do, and eliminating the EA or OP codes that the instruction was incapable of handling.

We also made sure to test some instructions that were not required, to make sure that our program could handle them. We carefully compared the output to the test code to ensure that generic “DATA” messages and the address location were printed in those locations.

As far as coding standards, we agreed upon formatting standards early on. Once we had an agreed upon format, it became much easier to combine our work later on. Additionally, whenever there was a subroutine that we knew would be shared by other members of the group, we made sure to make clear comments about what variables would be used by the subroutine. This was important so that we could avoid unnecessary conflicts between data or address registers overwriting one another.