Project Description

The program follows a pattern of recognition to ensure that when an OP code is passed in that is indeed valid before anything is printed. For example input is first tested to see if it matches a general op code it is then submitted to further test to see if it contains valid EA modes and registers for that specific operation.

An input loops loads a starting address labeled START\_INPUT to a register and also loads up END\_INPUT into a second register. The program then loads and loops threw a series of test functions which also call the corresponding print functions. Each function sets D7 if has decoded something to signal the test loop to break. Every time an address, op code, or value is loaded the address register that holds our current address is incremented. This process is continued until the current register is equal to or greater than (just in case) the ending address.

In order to achieve my standards all code was re-written to make it truly robust. Originally I decompiled and printed all of the OP codes (while working with the team) however I found my old methods flawed especially when combined together. This was split into two parts. First the op code was rigoursly tested for everything from correct sizes to correct EA addressing. This verified op code was then sent to another function which printed it assuming it was verified. Similarly the printing called a function which printed the ea which assumed it was verified. All false positives/negitives can be found in the OP\_CHECK functions. All formatting errors can be found in PRINTING and DECODE\_AS functions.

Below are flow charts that demonstrate an ideal flow of control. To see the charts in flow and all code visit this github <https://github.com/minogb/CSS-422-Decompiler>

This chart demonstrates the flow of the loop for testing op codes



This chart demonstrates the ideal flow of printing



This chart demonstrates overall workflow

