Mystery.s Decoding Process

- First step, see what the code does so I ran the program on the iLab machines using various inputs until I caught on to what the program was it was a Fibonacci program.
- It took the nth Fibonacci number as the argument and printed out its result.
- Once I had this in my mind, I had to actually decode the assembly language.
- I started with the main method cause it was the easiest to understand at first, and noticed the method calls that I already knew, like atoi(), printf(), and dothething().
- At this point, I started to crackdown on what exactly was going on by drawing myself pictures of the stack for each line of code, and basically drawing myself a picture of what the code was doing.
- Eventually I was able to generate my own C code that compiled and ran, and I began comparing the assembly it generated with the assembly in mystery.s. Eventually I got it to be very similar, but with a few differences.
- The mystery.s after inputting any number higher than 46, begins to print out negatives, mine did not.
- This meant that either there was some overflow condition the mystery.s kept running into, or somewhere along the way the signed bit was being flipped and it skewed the results.

Mystery.s Optimizations

- the compiler optimized the program by storing the information into registers rather than the stack because registers are faster access than going through the stack
- The loops, and conditionals were also shrunk into smaller chunks that I'm assuming would increase efficiency