Derek Schatel
RUID: 032004123

Design Document – Mystery.c

In order to figure out what was going on in mystery.s, I began by systematically examining each set of assembly instructions under each function. The "add" function was very obvious, but dothething took some additional deciphering. The big breakthrough came when I realized that "num" was in fact an array being called as a global variable. After this, I was able to decipher that the program was filling this num array with "-1" and then calling a recursive fibonacci function on the input.

The 'optimized' portion of mystery.s is the fact that the algorithm uses an array with pre-filled data. This way, when using the recursive calls on the function, if the value in the array has been changed from -1 then it won't even execute the rest of the code. This is useful because if you are computing a very high fibonacci number, then you will undoubtedly come across a lot of recursive calls where the value being passed has already been seen previously. This way, the program does not need to continually evaluate the input if it has already seen it.

When evaluating the optimized vs unoptimized code generated by the assembly compiler, I noticed that the optimized code did not move values into local memory. This makes sense because the program would run faster since it would not need to repeatedly read and write between the memory and the registers. Instead, the program simply moves the relevant values into registers and manipulates them from there.