Michael Wexler
Assignment 3

Design:

This project is really two projects in one. One is a program which calculates numbers in the fibonacci sequence. The other is a program which calculates a summation of integers. Each program consists of a main method, which calls a function written in assembly language. This assembly function is linked with the main code during compilation. Overflow conditions are checked in the assembly code.

Implementation:

For the fibonacci program, the assembly code essentially consists of a procedure which calls itself multiple times. It uses jump conditions to see if base cases are met. For the sum2n program, the for-loop is implemented in assembly by jumping back to a label again and again. Each time, the counter is incremented, and checked to see if it is over the limit.

Space/time analysis:

The fibonacci program consists of a function which over and over again calls itself as follows: let f(x) be the fibonacci function. The fibonacci, if the base conditions are not met, will call itself by returning $f(x - 1) + f(x - 2)$. This can be visualized by drawing a binary tree which constantly branches into two trees. The number of nodes in this binary tree, which is equal to the number of constant operations in this program, is 2^n. Therefore the run-time of this program is $O(2^n)$. Regarding space considerations, the multiple recursive calls requires a large amount of stack space. The amount of stack space required is $O(2^n)$.

The sum2n program basically consists of a for-loop. This for-loop is iterated n times. Therefore, since the body of the for loop is constant-time, the run-time is O(n). Regarding space considerations, there are a finite, small amount of integer variables declared, meaning the space consumption of this program is insignificant.

Design/implementation challenges:

Writing in assembly code was quite challeninging. I was not used to working at such a low level language before. Therefore, it was quite time-consuming and tedious to write the assembly code. It gave me a much greater appreciation for the high work-efficiency C.

Linking the C with the assembly was also difficult, since I had never done that before.

Figuring out the different flags and jump conditions in the assembly code was also a challenge.