For this assignment, the design of the actual coding aspect: mystery.c, was simple. Once you discover the pattern that is outlined in mystery.s, it is easy to see that the “mystery” algorithm is actually the highly popular Fibonacci sequence.

The code itself is an if statement inside a for loop. So there are constant comparisons, along with three assignments for each iteration of the loop and one addition. Every aspect of the code is entirely simple and linear. The run time ends up being n*(comparison + assignments + addition), where n is the number of iterations through the loop. As n approaches infinity, we can drop the constants being multiplied by n as they become insignificant (they only total to 5). This leaves us with just “n”. Since this program is entirely linear, our big O notation and runtime is O(n).

The design implementation was quick and simple. The real challenge I ran into was deciphering the assembly language. However, once I figured out exactly what every command and instruction meant, running through the code was very straightforward. It was interesting to see how -20(%rbp) ended up being the stop condition, whereas the final %eax was the answer, and %edx was the Fibonacci sequence up until that final answer. I ran into no problems designing the code to determine the nth term in the Fibonacci sequence.