Mystery

Rushabh Jhaveri

Computer Architecture, Summer 2017

Professor Brian Russell

Synopsis

Assembly Language Programming is an assignment composed of two directories:

1. formula
2. mystery

Mystery is a program which takes in one argument - an integer. This integer argument is the n^{th} Fibonacci number.

Formula then computes the value of the n^{th} Fibonacci number and prints it to the screen.

Design and Working

This program is designed to take a single integer input.

First, the program does an argument check, ensuring that there are correct number of arguments and displaying an error message if otherwise.

Then, the program takes the string argument and converts it to an integer number. If the argument passed as a string is not a digit, an appropriate error message is displayed, and the program exits.

The program then initializes an array of size 200, declared as a global variable, initializing the value at every array index to -1.

The program then calls on the function compute_fibonacci, and stores the return value in a variable called result.

int compute_fibonacci(int number): This function takes an integer argument, which is the n^{th} Fibonacci number, and returns the value of the n^{th} Fibonacci number. It does this by first checking whether the index numbered number in the array is -1, thus ensuring that value has not been computed yet. It then assigns the zeroth index of the array as zero, and the first index as one, as per the rule of the Fibonacci sequence. It then calculates any subsequent Fibonacci number through the recursive formula:

$$F_n = F_{n-1} + F_{n-2}$$

It is important to note here that this function has been coded based on the given mystery.s assembly code, and has been interpreted and translated to C code. Thus, the following was concluded based on observation:

- To calculate the sum, there was another function, **int add(int x, int y)**, which returned the sum of the two integer parameters passed to it.
- There existed an array called num which was used for calculation of the Fibonacci sum.
- Recursive implementation
- Multiple if conditions to check for recursive conditions

Challenges Faced

- Understanding how assembly initializes and allocates registers.
- Determining that there was an array being used.
- Understanding the compiler-optimized code.

Compiler Optimization

It is highly probable that there were more variables than there were blocks of memory claimed for use by the program. This is probably because the compiler knows ahead of time how many variables and registers will be used, and how many will be required. It thus uses minimum amount of registers and reuses registers for multiple variables. It’s also possible that the compiler reuses values that have already been computed and stores them elsewhere for later use, instead of re-computing them.