Formula

Rushabh Jhaveri

Computer Architecture, Summer 2017

Professor Brian Russell

Synopsis

Assembly Language Programming is an assignment composed of two directories:

1. formula
2. mystery

Formula is a program which takes in one argument - an integer. This integer argument is the power to which the binomial expansion of the formula (1 + x) is to be performed.

Formula then performs the expansion $(1 + x)^n$ and prints the expansion to the screen.

Design and Working

This program is designed to take a single integer input.

First, this program starts the timer and checks whether the user has passed “-h” as the argument, indicating the user has passed the help flag, and the program then prints out a usage message.

After that, 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.

This integer number, power, is the degree to which the binomial expansion is to be performed.

If the power is a number greater than or equal to 13, the program prints an error message. This is because the factorial of numbers 13 onwards exceed the range that an integer can hold.

The program then proceeds to calculate and print the binomial expansion of $(1+x)^n$, where n is the power input. To do this, it calls on a function:

int nCr(int n, int r): This function takes in two integer parameters - n, which is the power input, and r, which goes from $0-r$ and is used to calculate the binomial coefficient as per the formula:

$nCr \;=\; n!\,/\,(n-r)!$

To calculate these factorials, nCr calls upon a function:

int factorial(int n): This function takes in one integer parameter - n, which is the number whose factorial is to be computed. The factorial of n is computed by multiplying n by a recursive call to factorial, and passing $n-1$ as an argument. The base condition for this recursive function is that if n is zero or one, it shall return 1.

It must be noted here that the functions nCr and factorial were to be written in assembly, in the file nCr.s. This was done by writing the two functions in a file called nCr.c, and generating the assembly code by passing the command *gcc -S nCr.c*. This generated an nCr.s file, with the assembly instructions for the two functions generated as well.

After printing the expansion, the timer ends, and the time taken to perform the computation is calculated and displayed (in microseconds).

Challenges Faced

A few of the challenges I faced in the making of this program were:

- Getting the timer to work to calculate execution time.
- Determining when the limit of integer representation will be reached.
- Determining how to access functions in a different program that have been specified as “extern”.

Running Time Analysis and Space Time Analysis

This program runs in linear time and occupies linear amount of space. The reasoning behind this analysis is that the only major contributor to runtime is the loop where each binomial coefficient is being computed and each term of the binomial expansion is being printed to output. At the worst, this takes O(*n*) time.