Derek Schatel
RUID: 032004123

Design Document – Formula.c

When writing the assembly code for this program I attempted to be as expeditious as possible. In other words, I avoided moving too many values around to local memory or other registers and simply worked with the initial locations.

For Factorial, I decided to go for a looping algorithm rather than recursive because recursion did not seem to have less machine instructions. Factorial passes an immediate value of 1 into the %eax register and then uses the variable passed in at 8(%ebp) and compares it to an immediate value of 1. If that variable in memory is above or equal to one (using the unsigned jump instruction because we are only concerned with positive integers), the program multiplies the current variable of the %eax register by the value in memory and then decrements the value in memory. This continues until the value in memory is less than 1, at which point the program returns because the correct value is already in the %eax register. I also included a check for the overflow flag in case the multiplication instruction overflows. **Consequently, this program will only handle up to an input of 12**, and will report an overflow error otherwise.

The nCr instructions are fairly straightforward. I chose not to use any local memory and instead stored any temporary values in registers. This function calls Factorial once the correct value has been stored in the %eax register. First, it determines the Factorial for n and stores it in %ebx. Then it determines the factorial for r and stores it in %esi. Then it subtracts r from n and calls factorial on the result, multiplies this factorial result with the value of r! stored in %esi, and finally divides n! by this result.

In the C side of the program, if the machine instructions return a 0 (from an overflow flag), then the program reports an overflow error. I also included error checking for too few or too many arguments, as well as a check if the input is not a recognized integer. Finally, I included a tag of -h for usage instructions.

In total, I tried to write the assembly in a way that would not make the stack too large with too many recursive calls (which is another reason that I opted for a loop iteration in the instructions for Factorial rather than recursive.