Anirudh Tunoori

netid: at813

RUID: 165003175

pa3: formula

Note: Please excuse the excessive comments in nCr.s they were used to help me understand the steps that I was taking in each function and I left them in (unlikely event) case others are lost when following my code, Thanks!

My formula program consisted of 5 files: nCr.s, nCr.h, formula.c, formula.h and the makefile. My makefile contained 4 rules, formula which built the formula executable, formula.o which built formula.o, nCr.o which built nCr.o, and clean. My nCr.h file was left unchanged (from the skeleton that was provided) and contained two external functions. My nCr.s file contained the assembly code for my factorial and my nCr functions. The remaining two functions: the main function and the checkInput function can be found in formula.c (whose prototypes are defined in formula.h). The main function calls the check input function to ascertain that the input is a single integer. It also determines the elapsed/runtime of the entire operation, which typically varied around 30 to less than 50 microseconds, using the timeval struct. The main function also provides information if the user enters a help flag (-h). Finally the main function contains the output structure and obtains the (nCr/binomial) coefficients as each factor is printed. My function does not detect overflow (using the JO flag) although if Factorial returned a 0 (at power > 34)it would print overFlow detected. The overflow should occur for any int higher than 14 and the expansion of any power greater than 14 would be incorrect. The nCr function (written in assembly) makes the required three Factorial calls when necessary. It also performs signed multiplication and division. Compared to the Factorial function this function was not particularly problematic for me. The recitation ppt on Sakai helped me better understand the different addressing (particularly the different movl operands). Sometimes I mixed up my use of %esp and %ebp, and initially I was sure how to access the local variables (and general purpose registers) of the function. Factorial, gave me more difficulties as I initially tried a recursive approach for a long time (in order to optimize runtime). Although i had a decent understanding of the stack frame I would loose the information from the previous recursive call. I tried coming up with a separate, simple recursive C function to look at the assembly counterpart but in the end I switched to the iterative approach. The other difficulty with this function was “deciding the appropriate space” for addl and subl ($12, $8, $24, etc.).

This program does not use any high level data structures and primarily only primitive int datatypes (2^31 -1).

Runtime Analysis: as I mentioned above the runtime varies (30-50 microseconds), checkInputs (which isn't super essential to the functionality of the program) runs at O(n) as each digit of the input is checked. The main function runs at O(n) time as each factor requires a call to nCr and there are n factors. Factorial runs at O(n) as it it determines the factorial iteratively (a1*a2*… an). Finally the nCr function runs at O(n + r): it calls factorial three times through the process. It finds the three factorial components of the formula and returns the coefficient.

Space Analysis: The program can only take an int (2^31 -1)as an input. Anything larger than 14 will provide an incorrect expansion (this is because pointers pointing to the numbers are freed).