Charle Misback, Declan Casey, and Nathan Loria

11/12/20

Dr. Mohan

CS 200 Project Proposal

INTRODUCTION OF PROBLEM

For our final project, we aim to solve the problem of completing mathematical functions and operations without having an implemented function to do so. More simply, how does the computer complete the function when we tell a C program to add 2 and 2? At a hardware level, the simplest mathematical operators (+, -, /, *) are actually high level due to several reasons. The way data is expressed internally is different than it is expressed externally to the user, developed, etc. As we already know, machines read binary and humans read decimals, and the data conversion done between the two, as well as hexadecimal, is near instantaneous. Since the data is expressed differently at high and low levels, it must also be processed differently, using operators such as the shift operator, for example, to do so. Using only low level hardware solutions in C, we aim to implement a data convertor and calculator, as well as a CLI that allows the user to choose the operations they'd like to perform.

SOLUTION TO PROBLEM, WHY IT IS GOOD

In order to solve the previously outlined problems, the group proposes a calculator program. Rather than using classic arithmetic operators (+, -, /, *) the calculator program will make use of the same arithmetic algorithms that are utilized at a hardware level in computers. Along with the ability to perform basic mathematical operations, the program will allow a user to input either a hexadecimal, decimal, or binary value into the program in order for the execution to take place. The ability to allow the user these options will come from methods that can convert between all three data types that are presented (binary, decimal, and hexadecimal). Once the user enters their values for the computation and chooses an operator, the program will convert both inputted values into binary and proceed with calculations. As previously stated, within the addition function, the program will implement binary addition. This will be done by the indices of a malloc array as bits and using these bits to calculate a result & carryout value for each index. The subtraction algorithm will work in a similar manner, however the value that is being subtracted from another value will be inverted using the 2's complement rule and the addition procedure can then be run on this "negative" binary value. In order to implement the division and multiplication algorithms required for this project, the program will closely follow the steps of the algorithms that were in our class slides. Much like with addition and subtractions, the binary values used for these computations will be stored in malloc arrays. This program will also provide the user with a user friendly interface in order to facilitate efficient use. In order to avoid the (+, -, /, *) operators all together, bitwise operators will be utilized to perform math operations in methods other than the mathematical functions.

PROJECT TIMELINE, DIVISION OF WORK, POSSIBLE EXTRAPOLATIONS OF THE PROJECT

We will be dividing the work among us with Nate working on the calculator portion; making sure it can simulate the arithmetic operators without using any itself; Declan working on the converter, making sure it can convert back and forth between base two, ten, and sixteen; and Charles working on the command line UI and using bitwise operators to simulate the arithmetic operators. All the above duties will ideally be working at the bare minimum before our progress report is due December 1st. Our focus will be to polish the final project during the last few days before the final due date, December 4th. This project has many possible extrapolations; obvious instances like creating more mathematical functions like exponents or logarithms and complex instances such as creating a GUI and attempting to make a traditional calculator.