Number and Base Conversion and Calculator

Rushabh Jhaveri - Rutgers University

Computer Architecture, Summer 2017

Professor Brian Russell

Synopsis

Number and Base Conversion and Calculator is a C assignment composed of two programs:

1. calc
2. format

calc takes in four arguments:

1. Operation (+ / -) as a single-character string
2. First Operand (of base binary, decimal, octal, or hexadecimal) as a string
3. Second Operand (of base binary, decimal, octal, or hexadecimal) as a string
4. Desired Output Base (as binary, decimal, octal, or hexadecimal) as a single-character string

calc then adds (or subtracts) the two arguments and returns the answer in the desired output base.

format takes in two arguments:

1. Input bit sequence of 32 bits representative of a float or an integer
2. A string that indicates what type (int or float) the input bit sequence should be interpreted as

format then interprets the input bit sequence as specified and prints out the final answer as an integer (for int type) or as a decimal in the scientific notation (for float type).

---

calc.c ReadMe

Design and Working

This program is designed to take four operands - operation, two operands, and a desired output base.

This program does not account for decimal input.

First, this program extracts the individual arguments and does an argument check, ensuring that there are correct number of arguments and displaying an error message if otherwise. Then, it stores the arguments in variables, such that the operation, each operand, and the desired output base are stored in separate variables. Then, depending on which base the argument is inputted in, it calls a function to convert the operand from that particular base to an integer array, which serves as the binary representation of the input operand. It does this for both operands. Then, depending on the operation to be performed, it calls on an addition or subtraction function to perform the arithmetic operation. Before doing so, it checks if the input operand is negative. If so, it calls on a function to convert the operand to its two's complement negative representation. It then performs the required arithmetic operation, returning an integer array containing the binary representation of the added/subtracted inputs. This is the required final answer which must now be converted into the desired output base. It does this by checking what the output base is and calling a function to convert from binary to the desired output base. It then prints the final answer in the desired output base.

Following the specifications, this is how the program handles negative numbers:

- It is assumed that the number after the base specifier is positive.
- If a number is negative it is denoted by a '-' before the base specifier.

Challenges Faced in the Making of This Program:

1. Finding a suitable type to hold 64 bit integer values.
2. Coming up with the logic for base system conversions was doable, but actually coding them and making them work took longer than expected.

Running Time Analysis:

This program runs in approximately linear time. A major part of this program involves manipulating the integer arrays which serve as binary representations of the operands. All of these manipulations take no more than linear time to run.

---

format.c ReadMe

Design and Working:

This program takes in an input bit sequence of 32 bits which is representative of a float or an integer and a string that indicates which type the input bit sequence is to be interpreted as (int / float). It first checks whether the arguments passed are valid and whether the input bit sequence is of 32 bits and whether it contains only 1's and 0's, and displays an error message if otherwise. It then takes the input

bit sequence string and bit-shifts it into a number union Once the program has bit-shifted the bit pattern into the integer part of the number union it decides if it is an integer or a float bit pattern and passes the proper section of it to its respective function. If the type is an integer, it passes n.i to convert_binary_to_int which converts the string bit sequence to an integer value, and then passes that integer value to intToASCII which converts the integer number to a string for printing. If the type is a float, it passes n.f to floatToASCII which is code written and given by professor Russell which prints it as a floating point value in the scientific notation.