Abdulellah Abualshour

Computer Architecture

6 March 2016

Numeric Base Conversion and Calculator

The program takes four arguments which are the operand, two values, and an output base and performs the suitable operations to the two values after converting them to binary first and then converting the final result to the output base (e.g. binary to hexadecimal).

Design:

Most of the work in my implementation is done in the Adding and Subtracting part of the **main** function, and the **binary_operation** function as well. If statements are there to check if the input contains any negative values in addition to error checking for non-valid bases in any of the arguments. In **binary_operation**, a basic binary value addition or subtracting operation is done (same as strategy in which is done by hand). I also have helper functions to make the program more readable rather than doing everything in the main function:

- **strrev**: the usual reverse string function (since it's not available in C).
- **bitflip**: flips the bits (0s to 1s and vice versa)
- **padding**: adjusts the binary value passed as an argument in order to match the operation given.
- **bin1_is_bigger**: checks if the first binary argument is longer or greater than the second binary argument by checking the length, and if the lengths are equal we check the binary value itself (since a longer binary number of digits means a greater value).
- **oct_to_bin**
- **dec_to_bin**
- **hex_to_bin**
- **bin_to_dec**
- **one_bin_to_oct**
- **one_bin_to_hex**
- **one_oct_to_bin**
- **one_hex_to_bin**
- **base_adjust**: adjusts the output string to match the output base.
- **final_conversion**: final printouts and checks after completion of operations.

In addition, **calc.h** includes all heading necessary for the program to function properly (called at the top of **calc.c**).

The **makefile** includes all flags necessary for the program to run and also to allow the use of the **math.h** library in particular using the flag **–lm.**

Challenges:

The most challenging obstacle that I ran into was figuring out a way of manipulating the string arguments to do the suitable operations on them without casting them directly. Thankfully, I came out with idea of converting the two values I have to binary no matter what and then checking the binary string to see whether it is negative or positive and finding the actual value and difference between the two values (which one is bigger). Also, the multiplication function was too hard for me and I ended up not doing it as a result of this.