Calc Documentation

Matt Klein

March 5, 2013

1 Intro and About

This program, `calc` takes two given binary, hexadecimal, octal, or decimal numbers and either adds or subtracts them with each other. The given output will be what is specified, binary, hexadecimal, octal, or decimal. The `makefile` provided will create calc from `calc.c` and `calc.h` using the GCC compiler, with the `-ansi -pedantic -Wall` flags. At the very end of the program, the program decides to use the `free` function to clear memory.

2 Data Structures

This program uses built in C ints to do the calculations. These are very fast, using built in + or -. However, they are limited to 32 bits of data.

3 Run Time Analysis

Because we are using a C ints, the worst case of each method is the following:

- `fromDec -` $O(n)$

  This is only $O(n)$ for the length of the given number. All we do is go through once, converting from base 10 to the specified output base. The function is also smart enough to use two's complement per the to flip the bits.

- `toHexChar -` $O(1)$

  This is always $O(1)$ as we are sending this function a specific integer, and using random access of an array, we can recieve the specific hexadecimal character.

- `add and sub` - $O(1)$

  This is always $O(1)$ as it is built in C arithmetic.

- `tochar` - $O(1)$

  This is always instant because we just subtract an int and recieve the given value in the ASCII table.

- `strrev` - $O(n)$

  This is always$O(n)$ where n is the length of the given string.

- `toDec` - $O(n)$

  This converts a character with the given base to base 10. If it is base 16, we cann hexCharToInt.

- `hexCharToInt`- $O(l)$

  Worst case will be 5 as that is how many hex characters we have. If there is something not in hexadecimal form, it returns -1.

- `getBase`- $O(1)$

  All this function does is check the given character, and returning the given base. If it is not `b, o, d, x` then we print that it is an invalid base.

4 Space Analysis

The program can only read up to a 32 bits per each number, as it is C ints. Afterward, we free all pointers.