Other Data; C Compiling

CS 350: Computer Organization & Assembler Language Programming

Lab 2, due Fri Sep 6

[8/30: Typo Q5]

A. Why?

- Octal and hexadecimal are convenient ways to representing long bitstrings.
- We use floating-point numbers to represent non-whole numbers (numbers not evenly divisible by 1).

B. Outcomes

After this lab, you should be able to

- Translate between representations for integers in binary, octal, hex, and decimal (signed and unsigned) and to perform arithmetic using them.
- Translate floating-point numbers to/from binary, decimal and IEEE format.
- Show some of the precision problems of floating-point numbers.
- Declare, manipulate, and print signed and unsigned binary and hex integers of various lengths, in C.

C. Problems [80 points total]

1. [6 = 2*3 pts] Do Question 2.3 in the textbook (page 43): (a) Assume that there are about 400 students in your class. If every student is to be assigned a unique bit pattern, what is the minimum number of bits required to do this? (b) How many more students can be admitted to the class without requiring additional bits for each student's unique bit pattern?
2. [6 = 3*2 pts] Convert 680_{10} to binary, octal, and hexadecimal.
3. [5 pts] What is the octal representation of B25EA4D_{16} ?
4. [9 = 3*3 pts] As an 8-bit sign-magnitude number, FF_{16} (i.e., 11111111) represents -127_{10}. What decimal value does FF_{16} represent as an 8-bit number in (a) unsigned binary? (b) 2's complement? (c) 1's complement?

© James Sasaki, 2013

5. [3 = 3*3 pts] Show the results of each of the following steps: (a) Convert $4BD2_{16}$ to binary. (b) Then take the 2's complement negative. (c) Then convert back to hex. (Hint: Your answer to part (c) should equal the 16's complement of $4BD2_{16}$.)

6. [3 pts] The sequence `'a' '3'` and 6133_{16} represent the same 16-bit string. What decimal integer does this bitstring (and thus `'a' '3'` and 6133_{16}) stand for?

7. [10 = 2*5 pts] Let hex BED0 0000 represent an IEEE 32-bit floating-point number[1]. (a) What binary scientific notation value does it represent? (b) What decimal value does it represent? (You can write the answer in fractional or decimal form, your choice.)

8. [5 pts] What is the base 2 scientific notation representation of $5\,{}^{43}/_{64}$? (That's $5 + {}^{43}/_{64}$, not $5 \times ({}^{43}/_{64})$.)[2]

9. [8 = 5+3 pts] What is the IEEE 32-bit floating-point representation of $5\,{}^{43}/_{64}$, in (a) binary and (b) hex? (That's 5 + 43/64, not (5 * 43)/64.)

10. [9 = 3*3 pts] Let $X = 11.00000\ 00000\ 00000\ 00000\ 011_2$; (a) Why is it impossible to represent X exactly in 32-bit IEEE floating-point? (b) and (c) What are the two binary numbers closest to X that we *can* represent?

11. [5 pts] Say we're using 5 significant digits. To calculate (.00001 + .00001) + 1.0000 requires two additions. Do either of them cause truncation (of nonzero bits) before or after the addition? If so, specify how and when.

12. [5 pts] Repeat Question 10 on .00001 + (.00001 + 1.0000).

D. Programming Problem [20 pts]

First, compile and run the skeleton file `Lab02_skel.c.`[3] You should get the following output:

[1] In this problem and others, you can ignore any embedded blanks — they're just for readability.

[2] Have people stopped teaching mixed fractions?

[3] In general, skeleton files don't have to compile and run, but this one does.

```
CS 350 Lab 2 for ***Fill in your name and section ***

type char takes 1 byte(s), has max value of 127 = 0x7f
Adding 1 yields -128 = 0x80

type unsigned char takes 1 byte(s), has max value of 255 = 0xff
Adding 1 yields 0 = 0
```

The program output hopefully is self-explanatory, but the basic idea is that for each of the two types `char` and `unsigned char`, we print out the `sizeof` that type (the number of bytes a value of that type takes), we print out the largest value of that type in decimal and in hex, and we show what happens when you add 1 to that largest value: Since a `char` has 8 bits and uses 1 bit for the sign, the most positive `char` value is $0111\ 1111_2 = 127 = 7F_{16}$; adding `1` to this yields $1000\ 0000_2 = -127 = 80_{16}$. For an unsigned char, the largest value is $1111\ 1111_2 = 128 = FF_{16}$; adding `1` to this yields $0000\ 0000_2 = 0 = 0_{16}$.

Your Assignment

(1) Replace `***Fill in your name and section ***` with your name and section and (2) Add code so that it works for the types `short int`, `unsigned short int`, `int`, `unsigned int`, `long int`, and `unsigned long int`.

Hint: Do research as necessary to find out what the different `printf` formats and modifiers mean, because they won't be all the same. In particular, learn what `%d`, `%u`, and `%x` mean, and learn what the modifiers `#`, `hh` and `h` do.

Grading Scheme:

- A program that generates syntax errors or warnings earns ≤ 10/20 points.
- Assuming your program compiles cleanly, you earn 5 points for each of the 4 types, with 1 point each for the correct sizeof the type, the max value in decimal and hex, and the result of adding 1 in decimal and in hex.
- Include your name & section in header comments (at the top of your) `*.c` file. Also include your name & section in your program's output. As with lab 1, include your name in the file you submit. –1 point if you forget any or all of these things.