Data Representation: integers and fixed point. PA2.

Yipeng Huang

Rutgers University

Feburary 16, 2021

Table of contents

Looking ahead

Class plan

1. Today, Tuesday, 2/16: PA2 discussion. Integers. Floats.
2. Thursday, 2/18: Floats.
3. 2/18-2/22: Weekly short quiz on data representations.
4. Tuesday, 2/23: Introduction to the software-hardware interface.
5. Thursday, 2/25: Programming assignment 3: data representations.

Programming assignment 1: mission debriefing

Reflection survey

- Will release anonymous ungraded reflection / class feedback survey later this evening.
- Replaces weekly quiz 4.

Grading

- PA1 will be graded this week.
- The TAs responsible for PA1 grading are Neeraj and Abhinav.
- Contact them for any issues remaining about PA1.

Programming assignment 1: mission debriefing

Code review

- New mechanism for this class.
- Qualitative feedback on code (code organization, readability, commenting), beyond programming to satisfy autograder.
- Learn from fellow classmates' code.
- During the week of Feb. 22-26, you must attend your registered recitation. TAs will take attendance.
- Or, you can attend Section 09 recitation with TA Azita.
- During recitation, participate in a discussion about code quality, using anonymized examples of classmates' code from PA1.
- At the end of February we'll do another student survey to see if it's effective.

Programming assignment 2: mission briefing

Goals

- Now that we're past the logistics of using ilab and C syntax, this assignment focuses on implementing algorithms.
- Goal is to solidify skills with C programming, which is important for this class and for other CS major classes.

Quick start guide

- Graph adjacency matrices, adjacency lists, edge lists.
- Using the functions provided in the header.
- Depth-first search of a graph.

Table of contents

Don’t confuse the bitstring vs. the interpreted value

The bitstring

11111111, 377, 255, FF

Interpretation of the value

To interpret the value of a bitstring, you need to know:

1. the radix, number base: 2, 8, 10, 16.
2. the representation of signed values: two’s complement.
3. size of the data type: char, short, int, long
4. decimal point

Bit shifting

$<<$ N Left shift by N bits

- multiplies by 2^N
- $2 << 3 = 0000\_0010_2 << 3 = 0001\_0000_2 = 16 = 2 * 2^3$
- $-2 << 3 = 1111\_1110_2 << 3 = 1111\_0000_2 = -16 = -2 * 2^3$

$>>$ N Right shift by N bits

- divides by 2^N
- $16 >> 3 = 0001\_0000_2 >> 3 = 0000\_0010_2 = 2 = 16/2^3$
- $-16 >> 3 = 1111\_0000_2 >> 3 = 1111\_1110_2 = -2 = -16/2^3$

Table of contents

Unsigned fixed-point binary for fractions

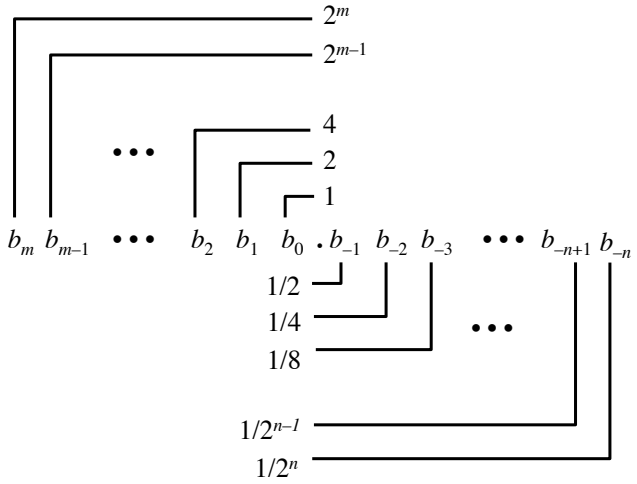

Figure: Fractional binary. Image credit CS:APP

Unsigned fixed-point binary for fractions

unsigned fixed-point char example	weight in decimal
1000.0000	8
0100.0000	4
0010.0000	2
0001.0000	1
0000.1000	0.5
0000.0100	0.25
0000.0010	0.125
0000.0001	0.0625

Table: Weight of each bit in an example fixed-point binary number

- $.625 = .5 + .125 = 0000.1010_2$
- $1001.1000_2 = 9 + .5 = 9.5$

Signed fixed-point binary for fractions

signed fixed-point char example	weight in decimal
1000.0000	-8
0100.0000	4
0010.0000	2
0001.0000	1
0000.1000	0.5
0000.0100	0.25
0000.0010	0.125
0000.0001	0.0625

Table: Weight of each bit in an example fixed-point binary number

- $-.625 = -8 + 4 + 2 + 1 + 0 + .25 + .125 = 1111.0110_2$
- $1001.1000_2 = -8 + 1 + .5 = -6.5$

Limitations of fixed-point

- Can only represent numbers of the form $x/2^k$
- Cannot represent numbers with very large or very small magnitude