Data Representation: integers and fixed point. PA2.

Yipeng Huang

Rutgers University

February 16, 2021
Table of contents

Announcements
   Looking ahead
   Programming assignments

Integers

Fractions and fixed point representation
Looking ahead

Class plan

2. Thursday, 2/18: Floats.
4. Tuesday, 2/23: Introduction to the software-hardware interface.
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

Announcements
   Looking ahead
   Programming assignments

Integers

Fractions and fixed point representation
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


\[ << N \text{ Left shift by } N \text{ bits} \]

- multiplies by \( 2^N \)
- \( 2 << 3 = 0000_0010_2 << 3 = 0001_0000_2 = 16 = 2 \times 2^3 \)
- \( -2 << 3 = 1111_1110_2 << 3 = 1111_0000_2 = -16 = -2 \times 2^3 \)

\[ >> N \text{ Right shift by } N \text{ bits} \]

- 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

Announcements
  Looking ahead
  Programming assignments

Integers

Fractions and fixed point representation
Unsigned fixed-point binary for fractions

Figure: Fractional binary. Image credit CS:APP
Unsigned fixed-point binary for fractions

<table>
<thead>
<tr>
<th>unsigned fixed-point char example</th>
<th>weight in decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000.0000</td>
<td>8</td>
</tr>
<tr>
<td>0100.0000</td>
<td>4</td>
</tr>
<tr>
<td>0010.0000</td>
<td>2</td>
</tr>
<tr>
<td>0001.0000</td>
<td>1</td>
</tr>
<tr>
<td>0000.1000</td>
<td>0.5</td>
</tr>
<tr>
<td>0000.0100</td>
<td>0.25</td>
</tr>
<tr>
<td>0000.0010</td>
<td>0.125</td>
</tr>
<tr>
<td>0000.0001</td>
<td>0.0625</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>signed fixed-point char example</th>
<th>weight in decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000.0000</td>
<td>-8</td>
</tr>
<tr>
<td>0100.0000</td>
<td>4</td>
</tr>
<tr>
<td>0010.0000</td>
<td>2</td>
</tr>
<tr>
<td>0001.0000</td>
<td>1</td>
</tr>
<tr>
<td>0000.1000</td>
<td>0.5</td>
</tr>
<tr>
<td>0000.0100</td>
<td>0.25</td>
</tr>
<tr>
<td>0000.0010</td>
<td>0.125</td>
</tr>
<tr>
<td>0000.0001</td>
<td>0.0625</td>
</tr>
</tbody>
</table>

**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