CSE 20
Intro to Computing I

Lecture 5 – Number Systems

Announcements

- Lab #6 this week
  - Due before your next lab
- Project #1 out this Friday
  - Due 10/27 (Friday)
- Midterm Exam
  - 10/23 during lecture
  - Lectures 1-5/Lab #1-7
  - Review during labs in week of 10/16
  - Open notes
  - No electronic devices (including calculators)
- Reading assignment
  - Chapter 4.1-4.3 of textbook

Data Types

- Boolean : 1-bit
  - 2 values, range : 0-1
- Short : 16-bits (2 bytes)
  - 2^{16} values, range : -32,768 to 32,767
- Char: 16-bits (2 bytes)
  - 2^{16} values, range : 0 to 65,535
- Int : 32-bits (4 bytes)
  - 2^{32} values, range : -2,147,483,648 to +2,147,483,647
- Float: 32-bits (4 bytes)
  - Scientific format : $\pm 3.4 \times 10^{\pm 38}$
- Double : 64–bits (8 bytes)
  - $\pm 1.7 \times 10^{\pm 308}$
- String : Any length (string of characters)

Integer Data (Ch 2.4)

- Integer Data
  - 8, 16, 32, or 64 bits can be used to represent an integer
  - In 8 bits you could use:
    - 00000001 = 1, 00000010 = 2, 00000011 = 3, 01111111 = 127
  - For an n-bit integer:
    - The smallest value is -2^{n-1}
    - The largest value is $2^{n-1} – 1$

Floating Point Data (Ch 2.5)

- Floating-Point Data
  - Data stored in a kind of scientific notation, with a fractional part (mantissa) OR an exponent part:

$$Value = (-1)^{sign} \times 2^{exponent}$$

$$\underset{\text{sign}}{\mathbf{s}}\,\underbrace{\mathbf{e\,e\,e\,e\,e\,e\,e\,e}}_{\text{exponent}}\,\underbrace{\mathbf{m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m\,m}}_{\text{mantissa}}$$

Floating Point Data

- Can represent non-integer values, larger OR smaller values than an int of the same size
- ***Precision*** is limited by number of bits in mantissa
- ***Range*** is limited by number of bits in exponent

Why So Many Kinds of Data Types?

- Some take up more space than others
  - Only an issue when you store a lot of data
- Some are more complicated to do arithmetic on and therefore slower
  - Not often an issue unless you are doing scientific computing
- Some allow for a wider range of values
- Some have higher precision
- Software engineering: use the best type for the job, not something overly general
  - Not all floating point numbers are integers
  - If a variable is only going to have integer values, document this by using an integer type

Counting Numbers/Number System

- There are different systems (bases) to represent numbers: Base 2, Base 8, Base 10, Base 16
- Starts at 0

0	0	0
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 0 + 1 → 1

1	1	1
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 1 + 1 → 2

$$\begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$$

2	10	2
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 2 + 1 → 3

3	11	3
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 3 + 1 → 4

$$\begin{array}{r} 11 \\ \underline{+\ \ 1} \\ 100 \end{array}$$

4	100	4
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 4 + 1 → 5

5	101	5
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 5 + 1 → 6

6	110	6
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 6 + 1 → 7

7	111	7
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 7 + 1 → 8

```
  111
+   1
-----
 1000
```

8	1000	8
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 8 + 1 → 9

9	1001	9
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 9 + 1 → 10
  - For Base 16, there is no single digit to represent “10”.

10	1010	A
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 10 + 1 → 11

11	1011	B
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 11 + 1 → 12

12	1100	C
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 12 + 1 → 13

13	1101	D
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 13 + 1 → 14

14	1110	E
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 14 + 1 → 15

15	1111	F
Base 10	**Base 2**	**Base 16**

Counting Numbers/Number System

- Add 1… 15 + 1 → 16

16	10000	10
Base 10	**Base 2**	**Base 16**

Number System: Table

Base 10	Base 2 (Binary)	Base 16 (Hex)
0	0	0x0
1	1	0x1
2	10	0x2
3	11	0x3
4	100	0x4
5	101	0x5
6	110	0x6
7	111	0x7
8	1000	0x8
9	1001	0x9
10	1010	0xA

Number System: Table

Base 10	Base 2 (Binary)	Base 16 (Hex)
11	1011	0xB
12	1100	0xC
13	1101	0xD
14	1110	0xE
15	1111	0xF
16	1 0000	0x10
32	10 0000	0x20
64	100 0000	0x40
128	1000 0000	0x80
256	1 0000 0000	0x100
512	10 0000 0000	0x200

Numbers Conversion

- What is the value of 5029?
  - Base 10: Each position is a power of 10
    - $5 \times 10^3 + 0 \times 10^2 + 2 \times 10^1 + 9 \times 10^0$
    - $5000 + 0 + 20 + 9 \rightarrow 5029_{10}$
- What is the value of 1010?
  - Base 10: 1010_{10}
    - $1 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 0 \times 10^0$
    - $1000 + 0 + 10 + 0 \rightarrow 1010_{10}$
  - Base 2: 1010_2
    - $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
    - $8 + 0 + 2 + 0 \rightarrow 10_{10}$
  - Base 16: 1010_{16}
    - $1 \times 16^3 + 0 \times 16^2 + 1 \times 16^1 + 0 \times 16^0$
    - $4096 + 0 + 16 + 0 \rightarrow 4112_{10}$

Binary -> Hex

- Groups of 4 binary bits = Hex
- Convert : 1 0 1 0 1 1 1 0 0 1 1 0 1 1 0 0
- A E 6 C
- 0xAE6C

https://www.khanacademy.org/math/algebra-home/alg-intro-to-algebra/algebra-alternate-number-bases/v/binary-to-hexadecimal

Hex -> Dec/Binary

- Convert 0x5B6F
- To decimal:
  - $5 \times 16^3 + 11 \times 16^2 + 6 \times 16^1 + 15 \times 16^0 = 23407_{10}$
- To binary:
  - $5_{16} = 0101_2$
  - $B_{16} = 1011_2$
  - $6_{16} = 0110_2$
  - $F_{16} = 1111_2$
  - $0101\ 1011\ 0110\ 1111_2$

Numbers Conversion

- What is 5029_{10} in Binary (Base 2)?
- Find the largest power of 2 such that $2^x \leq 5029$
  - 12^{th} is largest ($2^{12} = 4096$)
  - $5029 - 4096 = 933$
- Find the largest $2^x \leq 933$
  - $9^{th} \rightarrow 2^9 = 512 \rightarrow 933 - 512 = 421$
- Find the largest $2^x \leq 421$
  - $8^{th} \rightarrow 2^8 = 256 \rightarrow 421 - 256 = 165$
- Find the largest $2^x \leq 165$
  - $7^{th} \rightarrow 2^7 = 128 \rightarrow 165 - 128 = 37$
- Find the largest $2^x \leq 37$
  - $5^{th} \rightarrow 2^5 = 32 \rightarrow 37 - 32 = 5$
- Find the largest $2^x \leq 5$
  - $2^{nd} \rightarrow 2^2 = 4 \rightarrow 5 - 4 = 1$
- Find the largest $2^x \leq 1$
  - $0^{th} \rightarrow 2^0 = 1 \rightarrow 1 - 1 = \mathbf{0}$

Put 1's in each power of 2 that we found OR 0's in others

Binary : $1\ \underbrace{0011}_{3}\ \underbrace{1010}_{A}\ \underbrace{0101}_{5}$

Hex : 0x13A5

Check:

$1 \times 16^3 + 3 \times 16^2 + 10 \times 16^1 + 5 \times 16^0$
$\rightarrow 4096 + 768 + 160 + 5 = 5029$

https://www.khanacademy.org/math/algebra-home/alg-intro-to-algebra/algebra-alternate-number-bases/v/large-number-decimal-to-binary

Numbers: Operations

Symbol	Function
*	Multiply
/	Divide
%	Remainder
+	Add
-	Subtract

Numbers: Comparison

Operator	Meaning
==	equal
!=	not equal
<	Less than
>	Greater than
<=	Less than equal

The result is either True or False

Numbers: logical operator &&(AND)

Input 1	Input 2	Expression	Output
0	0	0 && 0	0
0	1	0 && 1	0
1	0	1 && 0	0
1	1	1 && 1	1
Input 1	**Input 2**	**Expression**	**Output**
FALSE	FALSE	False AND False	FALSE
FALSE	TRUE	False AND True	FALSE
TRUE	FALSE	True AND False	FALSE
TRUE	TRUE	True AND True	TRUE

Numbers: logical operator ||(OR)

Input 1	Input 2	Expression	Output
0	0	0 || 0	0
0	1	0 || 1	1
1	0	1 || 0	1
1	1	1 || 1	1
Input 1	**Input 2**	**Expression**	**Output**
FALSE	FALSE	False OR False	FALSE
FALSE	TRUE	False OR True	TRUE
TRUE	FALSE	True OR False	TRUE
TRUE	TRUE	True OR True	TRUE

Precedence

- 1+ 2 + 3 + 4
  - (((1 + 2) + 3) + 4)
- 1 * 2 + 3 * 4
  - (1 * 2) + (3 * 4)
- 1 * 2 < 3 * 4
  - (1 * 2) < (3 * 4)
- 1 < 2 && 3 < 4
  - (1 < 2) && (3 < 4)

- () – parentheses have highest precedence
- *, /, % are next in evaluation
- +, -
- <, <=, >, >=
- ==, !=
- &&, || are last to be evaluated
- Always evaluate left to right (default)

- (int) 1 + 2.0 // 1+2.0 → 3.0
- (int) (1 + 2.0) // (int)(3.0) → 3

Numbers - Divide

- 7 / 8
  - 0
- (float) 7 / 8
  - 0.875
- (float) (7 / 8)
  - 0.0
- (float) 7 / 8.0
  - 0.875
- (int) 7/8.0
  - 0.875
- (int) (7/8.0)
  - 0

Numbers – Remainder (Modulus)

- 7 % 8
  - 7
- 8 % 7
  - 1
- 2 % 1
  - 0
- 1 % 2
  - 1
- 7 % 2
  - 1
- 8 % 2
  - 0