Fixed-Point & Floating-Point Number Formats

CSC231—Fall 2014

Dominique Thiébaut
dthiebaut@smith.edu
http://cs.smith.edu/dftwiki/index.php/CSC231_An_Introduction_to_Fixed-_and_Floating-Point_Numbers
public static void main(String[] args) {

    int n = 10;
    int k = -20;

    float x = 1.50;
    double y = 6.02e23;

}
```java
public static void main(String[] args) {
    int n = 10;
    int k = -20;
    float x = 1.50;
    double y = 6.02e23;
}
```
public static void main(String[] args) {
    int n = 10;
    int k = -20;
    float x = 1.50;
    double y = 6.02e23;
}

Nasm knows what 1.5 is!
section .data

x dd 1.5

in memory, x is represented by

00111111111000000000000000000000
Fixed-Point Format

(used in very few applications where speed or memory is at a premium)
Review Decimal System

$$123.45 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$
Can we do the same in binary?

- With **unsigned numbers** first:

\[ 1101.11 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \]
Can we do the same in binary?

- With *unsigned numbers* first:

\[
1101.11 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}
\]

\[
= 8 + 4 + 1 + 0.5 + 0.25 = 13.75
\]
• The binary point is not "stored" anywhere

• A format where the binary/decimal point is fixed between 2 groups of bits is called a **fixed-point format**.
Definition

• A number format where the numbers are unsigned and where we have $a$ integer bits (on the left of the decimal point) and $b$ fractional bits (on the right of the decimal point) is referred to as a $U(a,b)$ fixed-point format.

• Value of an $N$-bit binary number in $U(a,b)$:

$$x = \left(\frac{1}{2^b}\right) \sum_{n=0}^{N-1} 2^n x_n$$
Exercises 1

• $x = 1011\ 1111 = 0\timesBF$

• What is the value represented by $x$ in $U(4,4)$

• What is the value represented by $x$ in $U(7,3)$
Exercises 2

• $z = 0000000100000000$

• $y = 0000001000000000$

• $v = 0000001010000000$

• What values do $z$, $y$, and $v$ represent in a $U(8,8)$ format?
Exercises 3

• What is 12.25 in $U(4,4)$? in $U(8,8)$?
What about **Signed** Numbers?
Observations

• In N-bit unsigned integer format, weight of MSB is $2^{N-1}$

• In N-bit signed, 2's complement, integer format, weight of MSB is $-2^{N-1}$
Fixed-Point Signed Format

- **Fixed-Point signed** format = sign bit + a integer bits + b fractional bits = N bits = $A(a, b)$

- $N = \text{number of bits} = 1 + a + b$

- Format of an $N$-bit $A(a, b)$ number:

\[
x = \left(\frac{1}{2^b}\right) \left[ -2^{N-1}x_{N-1} + \sum_{0}^{N-2} 2^n x_n \right],
\]
Examples in A(7,8)

- $00000000100000000 = 00000001 \cdot 00000000 = 1d$
- $10000000100000000 = 10000001 \cdot 00000000 = -128 + 1 = -127d$
- $0000001000000000 = 00000010 \cdot 00000000 = 2d$
- $1000001000000000 = 1000010 \cdot 00000000 = -128 + 2 = -126d$
- $0000001010000000 = 00000010 \cdot 10000000 = 2.5d$
- $1000001010000000 = 10000010 \cdot 10000000 = -128 + 2.5 = -125.5d$
Exercises

• What is -1 in A(7,8)?
• What is -1 in A(3,4)?
• What is 0 in A(7,8)?
• What is the smallest number one can represent in A(7,8)?
• The largest in A(7,8)?
Ranges

• Range = difference between most positive and most negative numbers.

• **Unsigned Range:**
  The range of $U(a, b)$ is $0 \leq x \leq 2^a - 2^{-b}$.

• **Signed Range:**
  The range of $A(a, b)$ is $-2^a \leq x \leq 2^a - 2^{-b}$. 
Precision

- **Precision** = \( N \), the number of bits
Resolution

• The resolution is the smallest non-zero magnitude representable.

• The resolution is the size of the intervals between numbers represented by the format

• Example: A(13, 2) has a resolution of 0.25.
Accuracy

• The **accuracy** is the largest magnitude of the difference between a number and its representation.

• **Accuracy** = 1/2 **Resolution**
Questions in search of answers…

• What is the accuracy of an U(7,8) number format?

• How good is U(7,8) at representing small numbers versus representing larger numbers? In other words, is the format treating small numbers better than large numbers, or the opposite?
Floating-Point Number Format
6.02 \times 10^{23}

-0.0000001

1.23456789 \times 10^{-19}

-1.0
1.230
12.30 \times 10^{-1}
123 \times 10^{-2}
0.123 \times 10^{1}
A bit of history…
• 1960s, 1970s: many different ways for computers to **represent** and **process** real numbers. Large variation in way real numbers are operated on.

• 1976: **Intel** starts design of first hardware floating-point **co-processor** for 8086. Wants to define a standard

• 1977: Second meeting under umbrella of **Institute for Electrical and Electronics Engineers** (IEEE). Mostly microprocessor makers (IBM is observer)

• Intel first to put whole **math library** in a processor
IEEE Format

• 32 bits (floats in Java)
• 64 bits (doubles in Java)
• 80 bits, extended precision (C, C++)

\[ x = +/- 1.bbbbbbb...bbb \times 2^{bbb...bb} \]
Observations

\[ x = +/- 1.bbbbbbb...bbb \times 2^{bbb...bb} \]

- +/- is the sign. It is represented by a bit, equal to 0 if the number is positive, 1 if negative.
- the part 1.bbbbbbb....bbb is called the **mantissa**
- the part bbb...bb is called the **exponent**
- 2 is the **base** for the exponent.
- the number is **normalized** so that its binary point is moved to the right of the leading 1.
- because the leading bit will always be 1, we don't need to store it. This bit will be an **implied bit**.
To be continued…