Week 14
Fixed & Floating Point Formats

CSC231—Assembly Language
Week #14

Dominique Thiébaut
dthiebaut@smith.edu
In C, strings are terminated by a byte containing 0 decimal, or 0000 0000 binary. In C, we express this quantity as '\0'.

In assembly, 0 as a byte is expressed as 0

'\0' in C = 0000 0000 = 0

'0' in assembly = 0011 0000 = 0x30

Cmsg    db    "hello", 0
cmp     al, 0
http://cs.smith.edu/dftwiki/index.php/CSC231_An_Introduction_to_Fixed-_and_Floating-Point_Numbers
```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;
}

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!
in memory, $x$ is represented by

```
00111111 11000000 00000000 00000000
```

or

```
0x3FC00000
```
Outline

• Fixed-Point Format

• Floating-Point Format
Fixed-Point Format

• Used in very few applications, but programmers know about it.

• Some micro controllers (e.g. Arduino Uno) do not have Floating Point Units (FPU), and must rely on libraries to perform Floating Point operations (VERY SLOW)

• Fixed-Point can be used when storage is at a premium (can use small quantity of bits to represent a real number)
Review Decimal Real Numbers

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

• Let's do it with **unsigned numbers** first:

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

Binary Point
Can we do the same in binary?

• Let's do it with **unsigned numbers** first:

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

\[ = 8 + 4 + 1 + 0.5 + 0.25 \]

\[ = 13.75 \]
• If we know where the binary point is, we do not need to "store" it anywhere. (Remember we used a bit to represent the +/- sign in 2’s complement.)

• A format where the binary/decimal point is fixed between 2 groups of bits is called a **fixed-point format**.
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 $U(a,b)$:

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

x = 1011 1111 = 0xBF

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

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

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

- $z = 00000001 \ 00000000$
- $y = 00000010 \ 00000000$
- $v = 00000010 \ 10000000$
Exercise 3

• What is 12.25 in $U(4,4)$? In $U(8,8)$?
What about Signed Fixed-Point Numbers?
Observation #1

- In an N-bit, \textit{unsigned} integer format, the weight of the MSB is $2^{N-1}$
<table>
<thead>
<tr>
<th>nybble</th>
<th>Unsigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>+0</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+2</td>
</tr>
<tr>
<td>0011</td>
<td>+3</td>
</tr>
<tr>
<td>0100</td>
<td>+4</td>
</tr>
<tr>
<td>0101</td>
<td>+5</td>
</tr>
<tr>
<td>0110</td>
<td>+6</td>
</tr>
<tr>
<td>0111</td>
<td>+7</td>
</tr>
<tr>
<td>1000</td>
<td>+8</td>
</tr>
<tr>
<td>1001</td>
<td>+9</td>
</tr>
<tr>
<td>1010</td>
<td>+10</td>
</tr>
<tr>
<td>1011</td>
<td>+11</td>
</tr>
<tr>
<td>1100</td>
<td>+12</td>
</tr>
<tr>
<td>1101</td>
<td>+13</td>
</tr>
<tr>
<td>1110</td>
<td>+14</td>
</tr>
<tr>
<td>1111</td>
<td>+15</td>
</tr>
</tbody>
</table>

\[
N = 4 \\
2^{N-1} = 2^3 = 8
\]
• In an N-bit **signed** 2's complement integer format, the weight of the MSB is $-2^{N-1}$
<table>
<thead>
<tr>
<th>nybble</th>
<th>2's complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>+0</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+2</td>
</tr>
<tr>
<td>0011</td>
<td>+3</td>
</tr>
<tr>
<td>0100</td>
<td>+4</td>
</tr>
<tr>
<td>0101</td>
<td>+5</td>
</tr>
<tr>
<td>0110</td>
<td>+6</td>
</tr>
<tr>
<td>0111</td>
<td>+7</td>
</tr>
<tr>
<td>1000</td>
<td>-8</td>
</tr>
<tr>
<td>1001</td>
<td>-7</td>
</tr>
<tr>
<td>1010</td>
<td>-6</td>
</tr>
<tr>
<td>1011</td>
<td>-5</td>
</tr>
<tr>
<td>1100</td>
<td>-4</td>
</tr>
<tr>
<td>1101</td>
<td>-3</td>
</tr>
<tr>
<td>1110</td>
<td>-2</td>
</tr>
<tr>
<td>1111</td>
<td>-1</td>
</tr>
</tbody>
</table>

\[ N=4 \]
\[ -2^{N-1} = -2^3 = -8 \]
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)

- $00000001\ 00000000 = 00000001\ .\ 00000000 = ?$
- $10000001\ 00000000 = 10000001\ .\ 00000000 = ?$
- $00000010\ 00000000 = 0000010\ .\ 00000000 = ?$
- $10000010\ 00000000 = 1000010\ .\ 00000000 = ?$
- $0000010\ 00000000 = 000010\ .\ 00000000 = ?$
- $1000010\ 00000000 = 100010\ .\ 00000000 = ?$
- $0000010\ 10000000 = 0000010\ .\ 10000000 = ?$
- $1000010\ 10000000 = 1000010\ .\ 10000000 = ?$
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 $A(7,8)$?
Exercises

• What is the largest number representable in $U(a, b)$?

• What is the smallest number representable in $U(a, b)$?

• What is the largest positive number representable in $A(a, b)$?

• What is the smallest negative number representable in $A(a, b)$?
• **Fixed-Point Format**

• **Definitions**
  
  • Range
  
  • Precision
  
  • Accuracy
  
  • Resolution

• **Floating-Point Format**
• 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** = \( b \), the number of fractional bits
  - [https://en.wikibooks.org/wiki/Floating_Point/Fixed-Point_Numbers](https://en.wikibooks.org/wiki/Floating_Point/Fixed-Point_Numbers)

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

2 different definitions
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.
$A(13, 2) \rightarrow \text{sbbbb bbbbb bbbbb bb . bb}$

$2^{-2} = 0.25$

$2^{-1} = 0.5$
$A(13, 2) \rightarrow \text{ sbb\textbf{b} bb\textbf{b} bb\textbf{b} bb . bb}$

Resolution

$2^{-2} = 0.25$

$2^{-1} = 0.5$
• The **accuracy** is the largest magnitude of the difference between a number and its representation.

• **Accuracy** = 1/2 **Resolution**
A(13, 2) \rightarrow \text{sbbb bbbb bbbb bb . bb}

\[
\begin{align*}
2^{-2} &= 0.25 \\
2^{-1} &= 0.5
\end{align*}
\]

Real quantity we want to represent
A(13, 2) $\rightarrow$ sbbb bbbb bbbb bb bb

Real quantity we want to represent

Representation

$2^{-2} = 0.25$

$2^{-1} = 0.5$
A(13, 2) \rightarrow \text{sbbbb bbbbb bbbbb bb . bb}

Error

\begin{align*}
2^{-2} &= 0.25 \\
2^{-1} &= 0.5
\end{align*}
A(13, 2) $\rightarrow$ sbbb bbbbb bbbbb bb . bb

Largest Error = Accuracy

$2^{-2} = 0.25$
$2^{-1} = 0.5$
Exercise

• 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?