Lecture 10

Floating-point representations

Methods of representing real numbers (1)

1. Fixed-point number system

100101010 1111010

100101010
1111010

• limited range and/or limited precision
• results must be scaled

2. Rational number system

\[
\frac{1001010101}{0110001011}
\]

1001010101
numerator

0110001011
denominator

\[
\text{Number} = \frac{1001010101}{0110001011}
\]
Methods of representing real numbers (2)

3. Logarithmic number system

\[ \begin{array}{c|c}
\text{sign} & \text{logarithm} \\
\hline
10010101010 & 1001010.1010_2 \\
\end{array} \]

- wide range
- low precision

Number = - 2

4. Floating-point number systems

Fixed-point vs. floating point representation (1)

**Fixed-point**

<table>
<thead>
<tr>
<th>ulp</th>
<th>Relative representation error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000 . 0000 1001_2</td>
<td>large</td>
</tr>
<tr>
<td>1001 0000 . 0000 0000_2</td>
<td>small</td>
</tr>
</tbody>
</table>

**Absolute representation error**

- ulp with truncation
- ulp/2 with rounding

<table>
<thead>
<tr>
<th>Absolute representation error constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative representation error variable</td>
</tr>
<tr>
<td>- large for small numbers</td>
</tr>
<tr>
<td>- small for large numbers</td>
</tr>
</tbody>
</table>
Fixed-point vs. floating point representation (2)

Floating-point

\[ 1.001 \cdot 2^{-5} \]
\[ 1.001 \cdot 2^{7} \]

**Absolute representation error variable**
- large for large numbers
- small for small numbers

**Relative representation error constant**

Floating point number (1)

number = ± significand \cdot base^{exponent}

\[ x = \pm s \cdot b^{e} \]

Typical value of the base
- \( b=2 \)
- but in the past
  - \( b=2, 8, 16, 256 \)

Typical range of the significand
- \( s \in [1, 2) \)
- Other common choice
  - \( s \in [0, 1) \)
  - Such \( s \) is called *mantissa*
Floating point number (2)

Exponent $e$ typically represented using biased representation

Exponent stored as

$$e + \text{bias}$$

where $\text{bias} = 2^{h-1} - 1$, $h$ number of bits of the exponent

Extra cost compared to the 2’s complement representation is small because of relatively small sizes of exponents

Range of floating point numbers

$$[-\text{max}, -\text{min}] \text{ and } [\text{min}, \text{max}]$$

$$-\infty \quad 0 \quad \infty$$

where

$\text{max} = \text{largest significand largest exponent}$

$$\text{max} = s_{\text{max}} b^{e_{\text{max}}}$$

$\text{min} = \text{smallest significand smallest exponent}$

$$\text{min} = s_{\text{min}} b^{e_{\text{min}}}$$
Current standard

ANSI / IEEE Std 754 - 1985

- Single-precision or short format, 32 bits
- Double-precision or long format, 64 bits

Operations on special numbers

<table>
<thead>
<tr>
<th>$\infty$</th>
<th>0</th>
<th>$\infty$</th>
<th>NaN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>not-a-number</td>
</tr>
</tbody>
</table>

- $0 / 0 = \text{NaN}$
- $+\infty \cdot 0 = \text{NaN}$
- $\pm\infty \cdot \text{Ordinary Number} = \pm\infty$
- $\text{NaN} + \text{Ordinary Number} = \text{NaN}$
Denormals

\[ 0.f \cdot 2^{-126} \]

hidden 1
replaced by 0

smallest exponent

Representation  
\[ e + \text{bias} = 0 \]
\[ s = f \neq 0 \]

Implementation of denormals is optional

Extended formats

Used *internally* to reduce the effect of accumulated errors

<table>
<thead>
<tr>
<th>Representation</th>
<th>Number of bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>normal</td>
</tr>
<tr>
<td>Single-precision</td>
<td></td>
</tr>
<tr>
<td>• Exponent</td>
<td>8</td>
</tr>
<tr>
<td>• Significand</td>
<td>23+1</td>
</tr>
<tr>
<td>Double-precision</td>
<td></td>
</tr>
<tr>
<td>• Exponent</td>
<td>11</td>
</tr>
<tr>
<td>• Significand</td>
<td>52+1</td>
</tr>
</tbody>
</table>
Floating-point operations
Unpacking & packing numbers

Unpacking
1. Separate sign, exponent, significand
2. Reinstate hidden 1
3. Convert to internal extended format

Packing
1. Test for special values
2. Remove hidden 1
3. Combine sign, exponent, and significand

Floating-point operations
Addition (1)

\[(\pm s_1 \cdot b^{e_1}) + (\pm s_2 \cdot b^{e_2}) = \]

= assuming \(e_1 > e_2 = \)

\[= (\pm s_1 \cdot b^{e_1}) + (\pm \frac{s_2}{b^{e_1-e_2}} \cdot b^{e_1}) = \]

\[= (\pm s_1 \pm \frac{s_2}{b^{e_1-e_2}}) b^{e_1} \]

Alignment shift or preshift

\[e_1 > e_2\]

\[
\begin{array}{c}
\text{s}1 \\
1.01110101 \\
\text{s}2 \\
1.01110101 \\
\text{e}1-\text{e}2 \text{ positions} \\
0.00001011
\end{array}
\]
Floating-point operations

Addition (2): Normalization shift or post shift

\[ s = \pm s_1 \pm \frac{s_2}{b^{e_1-e_2}} \quad s_1, s_2 \in [1, 2) \]

For the same sign of operands

\[ 1 \leq s < 4 \]

A single right shift of \( s \) and adding 1 to the exponent might be required

For the different signs of operands

\[ 0 \leq s < 2 \]

Multiple left shifts of \( s \) and decrementing exponent might be necessary

Floating-point operations

Addition (3)

If a common case of \( e_1-e_2 \geq 2 \)

\[ s_1 \in [1, 2) \]
\[ s_2' \in [0, 1/2) \]

\[ s_1-s_2' \in (1/2, 2) \]

\( s_1-s_2' \) might need to be shifted left by one position
Floating-point operations
Addition (4)

Each significand represented in two’s complement notation using at least the following bits

\[ \text{c}_{\text{out}} z_1 z_0 \cdot z_{-1} z_{-2} \ldots z_{-l} \ G \ R \ S \]

- \( z_1 \) - sign bit
- \( G \) - guard bit: protects against the loss of precision
- \( R \) - round bit: basic bit used for rounding
during left shift by one position
- \( S \) - sticky bit: auxiliary bit used for rounding
during left shift by one position

Floating-point operations
Addition (4)

R - used for rounding in case of one bit left shift performed
during normalization
- if \( R=0 \), set \( z'_{-l} \) to 0, discarded part < ulp/2
- \( R=1 \) and \( S \neq 0 \) set \( z'_{-l} \) to 1, discarded part < ulp/2
- \( R=1 \) and \( S=0 \) round to the nearest even number
discarded part = ulp/2,

S - logical OR of all bits shifted through the given position
during preshift (alignment shift)
Floating-point operations

Multiplication

\[(\pm s_1 \cdot b^{e_1}) \cdot (\pm s_2 \cdot b^{e_2}) = \]
\[= (\pm s_1 \cdot s_2) \cdot b^{e_1+e_2} = s \cdot b^e\]

\[1 \leq s_1 \cdot s_2 < 4\]

A single right shift of s and adding 1 to the exponent might be required

If e1 and e2 are of the same sign, multiplication may lead to overflow or underflow

Floating-point operations

Division

\[\frac{\pm s_1 \cdot b^{e_1}}{\pm s_2 \cdot b^{e_2}} = \pm \frac{s_1}{s_2} \cdot b^{e_1-e_2}\]

\[\frac{1}{2} < \frac{s_1}{s_2} < 2\]

A single left shift of s and subtracting 1 from the exponent might be required

If e1 and e2 are of the opposite signs, multiplication may lead to overflow or underflow
## Comparisons and exceptions

<table>
<thead>
<tr>
<th>Valid comparisons</th>
<th>Comparisons that generate exceptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-\infty &lt; 0$</td>
<td>$\text{NaN} &lt; +\infty$</td>
</tr>
<tr>
<td>$+ 0 = - 0$</td>
<td>$\text{NaN} &gt; -\infty$</td>
</tr>
<tr>
<td>$\text{NaN} \neq \text{NaN}$</td>
<td></td>
</tr>
<tr>
<td>$\text{NaN} \neq \pm \infty$</td>
<td></td>
</tr>
</tbody>
</table>

Operations that generate exceptions

$(-\infty) + (+\infty)$

$0 \cdot \infty$

$0 / 0$

$\infty / \infty$

## Rounding schemes (1)

### Truncation

Signed-magnitude representation

2’s complement representation
Rounding schemes (2)
Round to the nearest (rtn)

Signed-magnitude representation

-5  -4  -3  -2  -1  0  1  2  3  4

2’s complement representation

-5  -4  -3  -2  -1  0  1  2  3  4

Rounding schemes (3)
Round to the nearest (rtn)

Rounding errors

<table>
<thead>
<tr>
<th>Value</th>
<th>Rounded</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.01</td>
<td>0</td>
<td>-0.25</td>
</tr>
<tr>
<td>0.10</td>
<td>1</td>
<td>+0.5</td>
</tr>
<tr>
<td>0.11</td>
<td>1</td>
<td>+0.25</td>
</tr>
</tbody>
</table>

Average error +0.125
Rounding schemes (4)
Round to the nearest even (rtne)

Signed-magnitude representation

2’s complement representation