Floating Point Representation

Fractional binary numbers
IEEE floating-point standard
Floating-point operations and rounding

Lessons for programmers

Many more details we will skip (it’s a 58-page standard...)
See CSAPP 2.4 for more detail.

https://cs.wellesley.edu/~cs240/
Fractional Binary Numbers

\[
\sum_{k=-j}^{i} b_k \cdot 2^k
\]
# Fractional Binary Numbers

<table>
<thead>
<tr>
<th>Value</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 and 3/4</td>
<td></td>
</tr>
<tr>
<td>2 and 7/8</td>
<td></td>
</tr>
<tr>
<td>47/64</td>
<td></td>
</tr>
</tbody>
</table>

**Observations**

- Shift left =
- Shift right =

Numbers of the form $0.111111\ldots_2$ are...?

**Limitations:**

- Exact representation possible when?

\[
1/3 = 0.333333\ldots_{10} = 0.
\]
Fixed-Point Representation

Implied binary point.

\[ b_7 \ b_6 \ b_5 \ b_4 \ b_3 \ [.\] \ b_2 \ b_1 \ b_0 \]
\[ b_7 \ b_6 \ b_5 \ b_4 \ b_3 \ b_2 \ b_1 \ b_0 \ .\]

**range:** difference between largest and smallest representable numbers

**precision:** smallest difference between any two representable numbers

**fixed point = fixed range, fixed precision**
IEEE Floating Point Standard 754
IEEE = Institute of Electrical and Electronics Engineers

Numerical form:

\[ V_{10} = (-1)^s \times M \times 2^E \]

Sign bit \( s \) determines whether number is negative or positive
Significand (mantissa) \( M \) usually a fractional value in range \([1.0, 2.0)\)
Exponent \( E \) weights value by a \((-/+\) power of two
Analogous to scientific notation

Representation:

MSB \( s = \) sign bit \( s \)
\text{exp} \text{ field encodes } E \text{ (but is not equal to E)}
\text{frac} \text{ field encodes } M \text{ (but is not equal to M)}

Numerically well-behaved, but hard to make fast in hardware
Precisions

Single precision (float): 32 bits

Finite representation of infinite range...
Three kinds of values

\[ V = (-1)^s \cdot M \cdot 2^E \]

1. **Normalized**: \( M = 1.xxx... \)
   As in scientific notation: \( 0.011 \times 2^5 = 1.1 \times 2^3 \)
   Representation advantage?

2. **Denormalized**, near zero: \( M = 0.xxx... \), smallest \( E \)
   Evenly space near zero.

3. **Special values**:

   - **0.0**: \( s = 0 \) \( \text{exp} = 00...0 \) \( \text{frac} = 00...0 \)
   - **+inf, -inf**:
     \( \text{exp} = 11...1 \) \( \text{frac} = 00...0 \)
     Division by 0.0
   - **NaN** (“Not a Number”): \( \text{exp} = 11...1 \) \( \text{frac} \neq 00...0 \)
     sqrt(-1), \( \infty - \infty \), \( \infty \times 0 \), etc.
Value distribution

-∞ - Normalized - Denormalized + Denormalized + Normalized +∞

NaN -0.0 +0.0 NaN

Floating Point
Normalized values, with float example

\[ V = (-1)^s \times M \times 2^E \]

Value: float \( f = 12345.0 \);

\[ 12345_{10} = 110000000111001_2 \]
\[ = 1.1000000111001_2 \times 2^{13} \quad \text{(normalized form)} \]

Significand:

\[ M = \begin{array}{c}
1.1000000111001_2 \\
\text{frac=} 10000001110010000000000_2
\end{array} \]

Exponent: \( E = \exp - \text{Bias} \Rightarrow \exp = E + \text{Bias} \)

\[ E = 13 \]
\[ \text{Bias} = 127 = 2^7 - 1 = 2^{k-1} - 1 \]

Splits exponents roughly \(-/+\)

\[ \exp = 140 = 10001100_2 \]

Result:

\[ 0 \begin{array}{c}
10001100 \\
s \exp \frac{K=8}{n=23} \frac{K=8}{n=23}
\end{array} 10000001110010000000000000000 \]
Denormalized Values: near zero

"Near zero": \( \exp = 000...0 \)

Exponent:

\[
E = 1 + \exp - \text{Bias} = 1 - \text{Bias} \quad \text{not: } \exp - \text{Bias}
\]

Significand: leading zero

\[
M = 0.\ xxx...\ x_2 \\
\frac{\text{frac}}{} = xxx...x
\]

Cases:

\[
\exp = 000...0, \frac{\text{frac}}{} = 000...0 \quad 0.0, -0.0 \\
\exp = 000...0, \frac{\text{frac}}{} \neq 000...0
\]
Value distribution example

6-bit IEEE-like format

Bias = $2^{3-1} - 1 = 3$

**Full Range**

- $s=1, \ exp=101$
  - $E = 5-3 = 2$
- $s=0, \ exp=110$
  - $E = 6-3 = 3$

**Zoom in to 0**

- $s=1, \ exp=010$
  - $E = 2-3 = -1$
  - Denormalized = evenly spaced
- $s=0, \ exp=001$
  - $E = 1-3 = -2$

$\frac{1}{2} = 00, \ 01, \ 10, \ 11$

$M = 1.00, \ 1.01, \ 1.10, \ 1.11$

$E = -2, \ -1, \ 0, \ 1$
Try to represent 3.14, 6-bit example

6-bit IEEE-like format

Bias = $2^{3-1} - 1 = 3$

Value: 3.14;

3.14 = 11.0010 0011 1101 0111 0000 1010 000...

= 1.1001 0001 1110 1011 1000 0101 000... $2 \times 2^1$ (normalized form)

Significand:

\[ M = \begin{array}{c}
1.10010001111010111011100001010000...
\end{array} \]

\[ \text{frac} = \frac{1}{2} \]

Exponent:

\[ E = 1 \quad \text{Bias} = 3 \quad \text{exp} = 4 = 100_2 \]

Result:

\[ 0 \ 100 \ 10 \quad = \quad 1.10_2 \times 2^1 = 3 \quad \text{next highest?} \]
Floating Point Arithmetic*

\[ V = (-1)^s \cdot M \cdot 2^E \]

1. **Compute exact result.**
2. **Fix/Round**, roughly:
   - Adjust \( M \) to fit in \([1.0, 2.0)\)...
     - If \( M \geq 2.0 \): shift \( M \) right, increment \( E \)
     - If \( M < 1.0 \): shift \( M \) left by \( k \), decrement \( E \) by \( k \)
   - Overflow to infinity if \( E \) is too wide for \( \text{exp} \)
   - Round* \( M \) if too wide for \( \text{frac} \).
   - Underflow if nearest representable value is 0.
   - ...

*complicated...

```c
double x = ..., y = ...;
double z = x + y;
```
Lessons for programmers

\[ V = (-1)^S \times M \times 2^E \]

float \neq \text{real number} \neq \text{double}

Rounding breaks associativity and other properties.

\begin{verbatim}
    double a = ..., b = ...;
    ...
    \text{x}
    if (a == b) ... 
    if (abs(a - b) < epsilon) ...
\end{verbatim}