CSC 252: Computer Organization
Spring 2018: Lecture 4

Instructor: Yuhao Zhu

Department of Computer Science
University of Rochester

Action Items:
• Assignment 1 due Feb. 2, midnight
Announcement

• Programming Assignment 1 is out
  • Due on **Feb 2, 11:59 PM**
  • You have 3 slip days
  • Try to submit once to make sure you can submit
  • We count only the latest submission before the deadline

February 2018

<table>
<thead>
<tr>
<th>Mon 29</th>
<th>Tue 30</th>
<th>Wed 31</th>
<th>Thu Feb 1</th>
<th>Fri 2</th>
</tr>
</thead>
</table>

Announcement

• Programming Assignment 1 is out
  • Due on **Feb 2, 11:59 PM**
  • You have 3 slip days
  • Try to submit once to make sure you can submit
  • We count only the latest submission before the deadline

• TAs are better positioned to answer questions regarding assignments
Previously in 252...
Previously in 252...

• Signed vs. Unsigned Integer
  • Integer is a special case of fixed-point
  • Fractions can also be represented in fixed-point
Previously in 252…

• Signed vs. Unsigned Integer
  • Integer is a special case of fixed-point
  • Fractions can also be represented in fixed-point

• Least significant bit (byte)
  • Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  • Has nothing to do with which endianness you choose
Previously in 252…

- Signed vs. Unsigned Integer
  - Integer is a special case of fixed-point
  - Fractions can also be represented in fixed-point

- Least significant bit (byte)
  - Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  - Has nothing to do with which endianness you choose

10100011
Previously in 252…

- Signed vs. Unsigned Integer
  - Integer is a special case of fixed-point
  - Fractions can also be represented in fixed-point
- Least significant bit (byte)
  - Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  - Has nothing to do with which endianness you choose

\[10100011\]  \text{Least significant bit}
Previously in 252...

• Signed vs. Unsigned Integer
  • Integer is a special case of fixed-point
  • Fractions can also be represented in fixed-point

• Least significant bit (byte)
  • Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  • Has nothing to do with which endianness you choose

Most significant bit \[10100011\] Least significant bit
Previously in 252...

- Signed vs. Unsigned Integer
  - Integer is a special case of fixed-point
  - Fractions can also be represented in fixed-point
- Least significant bit (byte)
  - Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  - Has nothing to do with which endianness you choose

Most significant bit \(10100011\)  Least significant bit \(11\)

DEADBEEF
Previously in 252...

• Signed vs. Unsigned Integer
  • Integer is a special case of fixed-point
  • Fractions can also be represented in fixed-point

• Least significant bit (byte)
  • Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  • Has nothing to do with which endianness you choose

Most significant bit  10100011  Least significant bit

Most significant byte  DEADBEEF
Previously in 252…

• Signed vs. Unsigned Integer
  • Integer is a special case of fixed-point
  • Fractions can also be represented in fixed-point

• Least significant bit (byte)
  • Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  • Has nothing to do with which endianness you choose

**Most significant bit** 10100011 **Least significant bit**

**Most significant byte** DEADBEEF **Least significant byte**
Previously in 252...

- Signed vs. Unsigned Integer
  - Integer is a special case of fixed-point
  - Fractions can also be represented in fixed-point
- Least significant bit (byte)
  - Bit (byte) that is least significant to the numerical value of the bit stream — always the rightmost!
  - Has nothing to do with which endianness you choose

Google “Hexspeak”

- Most significant bit
  - 10100011
- Least significant bit
  - DEADBEEF
Today: Floating Point

- Background: Fractional binary numbers and fixed-point
- Floating point representation
- IEEE 754 standard
- Rounding, addition, multiplication
- Floating point in C
- Summary
Can We Represent Fractions in Binary?

- What does $10.01_2$ mean?
  - C.f., Decimal
Can We Represent Fractions in Binary?

• What does $10.01_2$ mean?
  • C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$
Can We Represent Fractions in Binary?

• What does $10.01_2$ mean?
  • C.f., Decimal

$$12.45 = 1*10^1 + 2*10^0 + 4*10^{-1} + 5*10^{-2}$$
Can We Represent Fractions in Binary?

• What does $10.01_2$ mean?
  • C.f., Decimal

$$12.45 = 1 \times 10^1 + \boxed{2} \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$
Can We Represent Fractions in Binary?

- What does $10.01_2$ mean?
  - C.f., Decimal

$$12.45 = 1 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$
Can We Represent Fractions in Binary?

• What does $10.01_2$ mean?
  • C.f., Decimal

$$12.45 = 1 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$
Can We Represent Fractions in Binary?

- What does $10.01_2$ mean?
  - C.f., Decimal

$$12.45 = 1\times10^1 + 2\times10^0 + 4\times10^{-1} + 5\times10^{-2}$$

$$10.01_2 = 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}$$
Can We Represent Fractions in Binary?

• What does \(10.01_2\) mean?
  • C.f., Decimal

\[
12.45 = 1\times10^1 + 2\times10^0 + 4\times10^{-1} + 5\times10^{-2}
\]

\[
10.01_2 = \boxed{1\times2^1} + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}
\]
Can We Represent Fractions in Binary?

- What does $10.01_2$ mean?
  - C.f., Decimal

\[
12.45 = 1\times10^1 + 2\times10^0 + 4\times10^{-1} + 5\times10^{-2}
\]

\[
10.01_2 = 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}
\]
Can We Represent Fractions in Binary?

• What does $10.01_2$ mean?
  • C.f., Decimal

$12.45 = 1 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$

$10.01_2 = 1 \times 2^1 + 0 \times 2^0 + \boxed{0 \times 2^{-1}} + 1 \times 2^{-2}$
Can We Represent Fractions in Binary?

• What does 10.01₂ mean?
  • C.f., Decimal

\[ 12.45 = 1 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} \]

\[ 10.01_2 = 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \]
Can We Represent Fractions in Binary?

• What does $10.01_2$ mean?
  • C.f., Decimal

$$12.45 = 1 \times 10^1 + 2 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

$$10.01_2 = 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$

$$= 2.25_{10}$$
Fractional Binary Numbers

\[ b_i \ b_{i-1} \ \cdots \ b_2 \ b_1 \ b_0 \ b_{-1} \ b_{-2} \ b_{-3} \ \cdots \ b_{-j} \]

\[ 2^i \]
\[ 2^{i-1} \]

\[ \frac{1}{2} \]
\[ \frac{1}{4} \]
\[ \frac{1}{8} \]

\[ 2^{-j} \]
Fractional Binary Numbers: Examples

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3/4</td>
<td>101.11</td>
</tr>
<tr>
<td>2 7/8</td>
<td>10.111</td>
</tr>
<tr>
<td>1 7/16</td>
<td>1.0111</td>
</tr>
</tbody>
</table>
## Fractional Binary Numbers: Examples

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3/4</td>
<td>101.11</td>
</tr>
<tr>
<td>2 7/8</td>
<td>10.111</td>
</tr>
<tr>
<td>1 7/16</td>
<td>1.0111</td>
</tr>
</tbody>
</table>

**Exact Same Raw Bit Stream!**
# Fractional Binary Numbers: Examples

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3/4</td>
<td>101.11</td>
</tr>
<tr>
<td>2 7/8</td>
<td>10.111</td>
</tr>
<tr>
<td>1 7/16</td>
<td>1.0111</td>
</tr>
</tbody>
</table>

- We would need to remember:
  - The raw bit stream
  - Where the binary point is

**Exact Same Raw Bit Stream!**
Fractional Binary Numbers: Examples

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3/4</td>
<td>101.11</td>
</tr>
<tr>
<td>2 7/8</td>
<td>10.111</td>
</tr>
<tr>
<td>1 7/16</td>
<td>1.0111</td>
</tr>
</tbody>
</table>

• We would need to remember:
  • The raw bit stream
  • Where the binary point is
• Makes calculations (e.g. addition) hard, and not very elegant
  • Need to first align numbers according to the binary point

Exact Same Raw Bit Stream!
Fixed-Point Representation
Fixed-Point Representation

- Binary point stays fixed
Fixed-Point Representation

• Binary point stays fixed

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000.</td>
</tr>
<tr>
<td>1</td>
<td>0001.</td>
</tr>
<tr>
<td>2</td>
<td>0010.</td>
</tr>
<tr>
<td>3</td>
<td>0011.</td>
</tr>
<tr>
<td>4</td>
<td>0100.</td>
</tr>
<tr>
<td>5</td>
<td>0101.</td>
</tr>
<tr>
<td>6</td>
<td>0110.</td>
</tr>
<tr>
<td>7</td>
<td>0111.</td>
</tr>
<tr>
<td>8</td>
<td>1000.</td>
</tr>
<tr>
<td>9</td>
<td>1001.</td>
</tr>
<tr>
<td>10</td>
<td>1010.</td>
</tr>
<tr>
<td>11</td>
<td>1011.</td>
</tr>
<tr>
<td>12</td>
<td>1100.</td>
</tr>
<tr>
<td>13</td>
<td>1101.</td>
</tr>
<tr>
<td>14</td>
<td>1110.</td>
</tr>
<tr>
<td>15</td>
<td>1111.</td>
</tr>
</tbody>
</table>
Fixed-Point Representation

- Binary point stays fixed

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000.</td>
</tr>
<tr>
<td>1</td>
<td>0001.</td>
</tr>
<tr>
<td>2</td>
<td>0010.</td>
</tr>
<tr>
<td>3</td>
<td>0011.</td>
</tr>
<tr>
<td>4</td>
<td>0100.</td>
</tr>
<tr>
<td>5</td>
<td>0101.</td>
</tr>
<tr>
<td>6</td>
<td>0110.</td>
</tr>
<tr>
<td>7</td>
<td>0111.</td>
</tr>
<tr>
<td>8</td>
<td>1000.</td>
</tr>
<tr>
<td>9</td>
<td>1001.</td>
</tr>
<tr>
<td>10</td>
<td>1010.</td>
</tr>
<tr>
<td>11</td>
<td>1011.</td>
</tr>
<tr>
<td>12</td>
<td>1100.</td>
</tr>
<tr>
<td>13</td>
<td>1101.</td>
</tr>
<tr>
<td>14</td>
<td>1110.</td>
</tr>
<tr>
<td>15</td>
<td>1111.</td>
</tr>
</tbody>
</table>
Fixed-Point Representation

- Binary point stays fixed

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00.00</td>
</tr>
<tr>
<td>0.25</td>
<td>00.01</td>
</tr>
<tr>
<td>0.5</td>
<td>00.10</td>
</tr>
<tr>
<td>0.75</td>
<td>00.11</td>
</tr>
<tr>
<td>1</td>
<td>01.00</td>
</tr>
<tr>
<td>1.25</td>
<td>01.01</td>
</tr>
<tr>
<td>1.5</td>
<td>01.10</td>
</tr>
<tr>
<td>1.75</td>
<td>01.11</td>
</tr>
<tr>
<td>2</td>
<td>10.00</td>
</tr>
<tr>
<td>2.25</td>
<td>10.01</td>
</tr>
<tr>
<td>2.5</td>
<td>10.10</td>
</tr>
<tr>
<td>2.75</td>
<td>10.11</td>
</tr>
<tr>
<td>3</td>
<td>11.00</td>
</tr>
<tr>
<td>3.25</td>
<td>11.01</td>
</tr>
<tr>
<td>3.5</td>
<td>11.10</td>
</tr>
<tr>
<td>3.75</td>
<td>11.11</td>
</tr>
</tbody>
</table>
Fixed-Point Representation

- Binary point stays fixed

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00.00</td>
</tr>
<tr>
<td>0.25</td>
<td>00.01</td>
</tr>
<tr>
<td>0.5</td>
<td>00.10</td>
</tr>
<tr>
<td>0.75</td>
<td>00.11</td>
</tr>
<tr>
<td>1</td>
<td>01.00</td>
</tr>
<tr>
<td>1.25</td>
<td>01.01</td>
</tr>
<tr>
<td>1.5</td>
<td>01.10</td>
</tr>
<tr>
<td>1.75</td>
<td>01.11</td>
</tr>
<tr>
<td>2</td>
<td>10.00</td>
</tr>
<tr>
<td>2.25</td>
<td>10.01</td>
</tr>
<tr>
<td>2.5</td>
<td>10.10</td>
</tr>
<tr>
<td>2.75</td>
<td>10.11</td>
</tr>
<tr>
<td>3</td>
<td>11.00</td>
</tr>
<tr>
<td>3.25</td>
<td>11.01</td>
</tr>
<tr>
<td>3.5</td>
<td>11.10</td>
</tr>
<tr>
<td>3.75</td>
<td>11.11</td>
</tr>
</tbody>
</table>
Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - Each bit represents $0.25_{10}$

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00.00</td>
</tr>
<tr>
<td>0.25</td>
<td>00.01</td>
</tr>
<tr>
<td>0.5</td>
<td>00.10</td>
</tr>
<tr>
<td>0.75</td>
<td>00.11</td>
</tr>
<tr>
<td>1</td>
<td>01.00</td>
</tr>
<tr>
<td>1.25</td>
<td>01.01</td>
</tr>
<tr>
<td>1.5</td>
<td>01.10</td>
</tr>
<tr>
<td>1.75</td>
<td>01.11</td>
</tr>
<tr>
<td>2</td>
<td>10.00</td>
</tr>
<tr>
<td>2.25</td>
<td>10.01</td>
</tr>
<tr>
<td>2.5</td>
<td>10.10</td>
</tr>
<tr>
<td>2.75</td>
<td>10.11</td>
</tr>
<tr>
<td>3</td>
<td>11.00</td>
</tr>
<tr>
<td>3.25</td>
<td>11.01</td>
</tr>
<tr>
<td>3.5</td>
<td>11.10</td>
</tr>
<tr>
<td>3.75</td>
<td>11.11</td>
</tr>
</tbody>
</table>
Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - Each bit represents $0.25_{10}$

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00.00</td>
</tr>
<tr>
<td>0.25</td>
<td>00.01</td>
</tr>
<tr>
<td>0.5</td>
<td>00.10</td>
</tr>
<tr>
<td>0.75</td>
<td>00.11</td>
</tr>
<tr>
<td>1</td>
<td>01.00</td>
</tr>
<tr>
<td>1.25</td>
<td>01.01</td>
</tr>
<tr>
<td>1.5</td>
<td>01.10</td>
</tr>
<tr>
<td>1.75</td>
<td>01.11</td>
</tr>
<tr>
<td>2</td>
<td>10.00</td>
</tr>
<tr>
<td>2.25</td>
<td>10.01</td>
</tr>
<tr>
<td>2.5</td>
<td>10.10</td>
</tr>
<tr>
<td>2.75</td>
<td>10.11</td>
</tr>
<tr>
<td>3</td>
<td>11.00</td>
</tr>
<tr>
<td>3.25</td>
<td>11.01</td>
</tr>
<tr>
<td>3.5</td>
<td>11.10</td>
</tr>
<tr>
<td>3.75</td>
<td>11.11</td>
</tr>
</tbody>
</table>

- Still need to remember the binary point, but just once for all numbers
Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - Each bit represents $0.25_{10}$

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00.00</td>
</tr>
<tr>
<td>0.25</td>
<td>00.01</td>
</tr>
<tr>
<td>0.5</td>
<td>00.10</td>
</tr>
<tr>
<td>0.75</td>
<td>00.11</td>
</tr>
<tr>
<td>1</td>
<td>01.00</td>
</tr>
<tr>
<td>1.25</td>
<td>01.01</td>
</tr>
<tr>
<td>1.5</td>
<td>01.10</td>
</tr>
<tr>
<td>1.75</td>
<td>01.11</td>
</tr>
<tr>
<td>2</td>
<td>10.00</td>
</tr>
<tr>
<td>2.25</td>
<td>10.01</td>
</tr>
<tr>
<td>2.5</td>
<td>10.10</td>
</tr>
<tr>
<td>2.75</td>
<td>10.11</td>
</tr>
<tr>
<td>3</td>
<td>11.00</td>
</tr>
<tr>
<td>3.25</td>
<td>11.01</td>
</tr>
<tr>
<td>3.5</td>
<td>11.10</td>
</tr>
<tr>
<td>3.75</td>
<td>11.11</td>
</tr>
</tbody>
</table>

- Still need to remember the binary point, but just once for all numbers
- No need to align (already aligned)
Fixed-Point Representation

- Binary point stays fixed
- Fixed interval between representable numbers
  - Each bit represents $0.25_{10}$

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00.00</td>
</tr>
<tr>
<td>0.25</td>
<td>00.01</td>
</tr>
<tr>
<td>0.5</td>
<td>00.10</td>
</tr>
<tr>
<td>0.75</td>
<td>00.11</td>
</tr>
<tr>
<td>1</td>
<td>01.00</td>
</tr>
<tr>
<td>1.25</td>
<td>01.01</td>
</tr>
<tr>
<td>1.5</td>
<td>01.10</td>
</tr>
<tr>
<td>1.75</td>
<td>01.11</td>
</tr>
<tr>
<td>2</td>
<td>10.00</td>
</tr>
<tr>
<td>2.25</td>
<td>10.01</td>
</tr>
<tr>
<td>2.5</td>
<td>10.10</td>
</tr>
<tr>
<td>2.75</td>
<td>10.11</td>
</tr>
<tr>
<td>3</td>
<td>11.00</td>
</tr>
<tr>
<td>3.25</td>
<td>11.01</td>
</tr>
<tr>
<td>3.5</td>
<td>11.10</td>
</tr>
<tr>
<td>3.75</td>
<td>11.11</td>
</tr>
</tbody>
</table>

- Still need to remember the binary point, but just once for all numbers
- No need to align (already aligned)
- C uses fixed-point encoding only for integral data types (long, int, short, etc.)
  - Effectively, implicitly assumes the binary point is at the rightmost
Limitations of Fixed-Point (#1)
Limitations of Fixed-Point (#1)

• Can exactly represent numbers only of the form \(x/2^k\)
Limitations of Fixed-Point (#1)

• Can exactly represent numbers only of the form $x/2^k$
Limitations of Fixed-Point (#1)

• Can exactly represent numbers only of the form $x/2^k$
  • Other rational numbers have repeating bit representations
Limitations of Fixed-Point (#1)

• Can exactly represent numbers only of the form $x/2^k$
  • Other rational numbers have repeating bit representations

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/3</td>
<td>0.0101010101[01]…</td>
</tr>
<tr>
<td>1/5</td>
<td>0.001100110011[0011]…</td>
</tr>
<tr>
<td>1/10</td>
<td>0.0001100110011[0011]…</td>
</tr>
</tbody>
</table>
Limitations of Fixed-Point (#2)
Limitations of Fixed-Point (#2)

• Can’t represent very small and very large numbers at the same time
Limitations of Fixed-Point (#2)

• Can’t represent very small and very large numbers at the same time
  • To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
Limitations of Fixed-Point (#2)

• Can’t represent very small and very large numbers at the same time
  • To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers

0 .... +∞
Limitations of Fixed-Point (#2)

• Can’t represent very small and very large numbers at the same time
  • To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
Limitations of Fixed-Point (#2)

• Can’t represent very small and very large numbers at the same time
  • To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
Limitations of Fixed-Point (#2)

- Can’t represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
  - To represent very small numbers, the (fixed) interval needs to be small, making it hard to represent large numbers
Limitations of Fixed-Point (#2)

- Can’t represent very small and very large numbers at the same time
  - To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
  - To represent very small numbers, the (fixed) interval needs to be small, making it hard to represent large numbers
Limitations of Fixed-Point (#2)

• Can’t represent very small and very large numbers at the same time
  • To represent very large numbers, the (fixed) interval needs to be large, making it hard to represent small numbers
  • To represent very small numbers, the (fixed) interval needs to be small, making it hard to represent large numbers
Today: Floating Point

- Background: Fractional binary numbers and fixed-point
- Floating point representation
- IEEE 754 standard
- Rounding, addition, multiplication
- Floating point in C
- Summary
Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^E$
  - $E$ is an integer
  - Normalized form: $1 \leq |M| < 10$
Primer: (Normalized) Scientific Notation

- In decimal: $M \times 10^E$
  - $E$ is an integer
  - Normalized form: $1 \leq |M| < 10$

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Scientific Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$2 \times 10^0$</td>
</tr>
<tr>
<td>-4,321.768</td>
<td>$-4.321768 \times 10^3$</td>
</tr>
<tr>
<td>0.000 000 007 51</td>
<td>$7.51 \times 10^{-9}$</td>
</tr>
</tbody>
</table>
Primer: (Normalized) Scientific Notation

• In decimal: $M \times 10^E$
  • $E$ is an integer
  • Normalized form: $1 \leq |M| < 10$

\[
\begin{array}{|c|c|}
\hline
\text{Decimal Value} & \text{Scientific Notation} \\
\hline
2 & 2 \times 10^0 \\
-4,321.768 & -4.321768 \times 10^3 \\
0.000\ 000\ 007\ 51 & 7.51 \times 10^{-9} \\
\hline
\end{array}
\]
Primer: (Normalized) Scientific Notation

• In decimal: $M \times 10^E$
  • $E$ is an integer
  • Normalized form: $1 \leq |M| < 10$

\[
M \times 10^E
\]

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Scientific Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$2 \times 10^0$</td>
</tr>
<tr>
<td>-4,321.768</td>
<td>$-4.321768 \times 10^3$</td>
</tr>
<tr>
<td>0.000 000 007 51</td>
<td>$7.51 \times 10^{-9}$</td>
</tr>
</tbody>
</table>
Primer: (Normalized) Scientific Notation

- In decimal: \( M \times 10^E \)
  - \( E \) is an integer
  - Normalized form: \( 1 \leq |M| < 10 \)

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Scientific Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( 2 \times 10^0 )</td>
</tr>
<tr>
<td>-4,321.768</td>
<td>(-4.321768 \times 10^3)</td>
</tr>
<tr>
<td>0.000 000 007 51</td>
<td>( 7.51 \times 10^{-9})</td>
</tr>
</tbody>
</table>
Primer: (Normalized) Scientific Notation

• In decimal: $M \times 10^E$
  - $E$ is an integer
  - Normalized form: $1 \leq |M| < 10$

<table>
<thead>
<tr>
<th>Decimal Value</th>
<th>Scientific Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$2 \times 10^0$</td>
</tr>
<tr>
<td>-4,321.768</td>
<td>$-4.321768 \times 10^3$</td>
</tr>
<tr>
<td>0.000 000 007 51</td>
<td>$7.51 \times 10^{-9}$</td>
</tr>
</tbody>
</table>

Significand Base Exponent
Primer: (Normalized) Scientific Notation

- In binary: \((-1)^s M 2^E\)
- Normalized form:
  - \(1 \leq M < 10\)
  - \(M = 1.b_0b_1b_2b_3\ldots\)
Primer: (Normalized) Scientific Notation

• In binary: \((-1)^s \times M \times 2^E\)

• Normalized form:
  • \(1 \leq M < 10\)
  • \(M = 1.b_0b_1b_2b_3\ldots\)
Primer: (Normalized) Scientific Notation

• In binary: \((-1)^s M \times 2^E\)

• Normalized form:
  • \(1 \leq M < 10\)
  • \(M = 1.b_0b_1b_2b_3\ldots\)
Primer: (Normalized) Scientific Notation

• In binary: \((-1)^s M 2^E\)

• Normalized form:
  • \(1 \leq M < 10\)
  • \(M = 1.b_0b_1b_2b_3...\)
Primer: (Normalized) Scientific Notation

- In binary: \((-1)^s M 2^E\)
- Normalized form:
  - \(1 \leq M < 10\)
  - \(M = 1.b_0b_1b_2b_3\ldots\)
Primer: (Normalized) Scientific Notation

• In binary: \((-1)^s M \times 2^E\)

• Normalized form:
  • \(1 \leq M < 10\)
  • \(M = 1.b_0b_1b_2b_3\ldots\)

\((-1)^s M \times 2^E\)

Significand  Exponent

Base
Primer: (Normalized) Scientific Notation

- In binary: \((-1)^s \, M \, 2^E\)
- Normalized form:
  - \(1 \leq M < 10\)
  - \(M = 1.b_0b_1b_2b_3\ldots\)

<table>
<thead>
<tr>
<th>Binary Value</th>
<th>Scientific Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110110110110</td>
<td>((-1)^0 , 1.110110110110 \times 2^{12})</td>
</tr>
<tr>
<td>-101.11</td>
<td>((-1)^1 , 1.0111 \times 2^2)</td>
</tr>
<tr>
<td>0.00101</td>
<td>((-1)^0 , 1.01 \times 2^{-3})</td>
</tr>
</tbody>
</table>
Primer: Floating Point Representation

- In binary: \((-1)^s \, M \, 2^E\)
- Normalized form:
  - \(1 \leq M < 2\)
  - \(M = 1.b_0b_1b_2b_3\ldots\)

\[
(-1)^s \, M \times 2^E
\]
Primer: Floating Point Representation

- In binary: \((-1)^s M 2^E\)
- Normalized form:
  - \(1 \leq M < 2\)
  - \(M = 1.b_0b_1b_2b_3\ldots\)
- Encoding

![Floating Point Representation Diagram]
Primer: Floating Point Representation

• In binary: $(-1)^s M 2^E$
• Normalized form:
  • $1 <= M < 2$
  • $M = 1.b_0b_1b_2b_3…$
• Encoding
Primer: Floating Point Representation

• In binary: \((-1)^s \, M \, 2^E\)
• Normalized form:
  • \(1 \leq M < 2\)
  • \(M = 1.b_0b_1b_2b_3\ldots\)
    Fraction
• Encoding
  • MSB \(s\) is sign bit \(s\)
Primer: Floating Point Representation

- In binary: \((-1)^s M \times 2^E\)
- Normalized form:
  - \(1 <= M < 2\)
  - \(M = 1.b_0b_1b_2b_3\ldots\)
- Encoding
  - MSB \(s\) is sign bit \(s\)
  - \(exp\) field encodes Exponent (but not exactly the same as \(E\))
Primer: Floating Point Representation

• In binary: \((-1)^s \times M \times 2^E\)
• Normalized form:
  • \(1 \leq M < 2\)
  • \(M = 1.b_0b_1b_2b_3\ldots\)
• Encoding
  • MSB \(s\) is sign bit \(s\)
  • \(exp\) field encodes Exponent (but not exactly the same as \(E\))
  • \(frac\) field encodes Fraction (but not exactly the same as Fraction)
6-bit Floating Point Example

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

\[
\begin{array}{ccc}
   s & \text{exp} & \text{frac} \\
   1 & 3 & 2 \\
\end{array}
\]
6-bit Floating Point Example

\( v = (-1)^s \ M \ 2^E \)

- \( \text{exp} \) has 3 bits, interpreted as an unsigned value
6-bit Floating Point Example

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

- \( exp \) has 3 bits, interpreted as an unsigned value
  - If \( exp \) were \( E \), we could represent exponents from \( 0 \) to \( 7 \)
6-bit Floating Point Example

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

- **exp** has 3 bits, interpreted as an unsigned value
  - If **exp** were **E**, we could represent exponents from 0 to 7
  - How about negative exponent?
6-bit Floating Point Example

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

- \( s \)  exp  frac
  1  3  2

- \( exp \) has 3 bits, interpreted as an unsigned value
  - If \( exp \) were \( E \), we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: \( E = \text{exp} - \text{bias} \) (i.e., \( \text{exp} = E + \text{bias} \))
6-bit Floating Point Example

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

- exp has 3 bits, interpreted as an unsigned value
  - If exp were E, we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: \( E = \text{exp} - \text{bias} \) (i.e., \( \text{exp} = E + \text{bias} \))
  - bias is always \( 2^{k-1} - 1 \), where k is number of exponent bits
6-bit Floating Point Example

$v = (-1)^s \times M \times 2^E$

- **exp** has 3 bits, interpreted as an unsigned value
  - If **exp** were $E$, we could represent exponents from **0 to 7**
  - How about negative exponent?
  - Add a bias term: $E = \text{exp} - \text{bias}$ (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where $k$ is number of exponent bits

- Example when $k = 3$: 

```plaintext
  1 3 2
```
6-bit Floating Point Example

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

- \( s \)  3  2

- \( exp \) has 3 bits, interpreted as an unsigned value
  - If \( exp \) were \( E \), we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: \( E = exp - bias \) (i.e., \( exp = E + bias \))
  - bias is always \( 2^{k-1} - 1 \), where \( k \) is number of exponent bits

- Example when \( k = 3 \):
  - bias = 3
6-bit Floating Point Example

\[ v = (-1)^s \: M \: 2^E \]

- \( s \) \quad \text{exp} \quad \text{frac} \n
1 \quad 3 \quad 2

- \text{exp} \) has 3 bits, interpreted as an unsigned value
  - If \( \text{exp} \) were \( E \), we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: \( E = \text{exp} - \text{bias} \) (i.e., \( \text{exp} = E + \text{bias} \))
  - bias is always \( 2^{k-1} - 1 \), where \( k \) is number of exponent bits

- Example when \( k = 3 \):
  - bias = 3
  - If \( E = -2 \), \( \text{exp} \) is 1 (001\(_2\))
6-bit Floating Point Example

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

<table>
<thead>
<tr>
<th>s</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

• \( exp \) has 3 bits, interpreted as an unsigned value
  - If \( exp \) were \( E \), we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: \( E = exp - bias \) (i.e., \( exp = E + bias \))
  - bias is always \( 2^{k-1} - 1 \), where \( k \) is number of exponent bits

• Example when \( k = 3 \):
  - bias = 3
  - If \( E = -2 \), \( exp \) is 1 (001₂)

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>
6-bit Floating Point Example

- exp has 3 bits, interpreted as an unsigned value
  - If exp were E, we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: $E = \text{exp} - \text{bias}$ (i.e., exp = E + bias)
  - bias is always $2^{k-1} - 1$, where k is number of exponent bits

- Example when $k = 3$:
  - bias = 3
  - If $E = -2$, exp is 1 (001₂)
  - Reserve 000 and 111 for other purposes (more on this later)
6-bit Floating Point Example

\[ v = (-1)^s \, M \, 2^E \]

- **exp** has 3 bits, interpreted as an unsigned value
  - If **exp** were **E**, we could represent exponents from 0 to 7
  - How about negative exponent?
  - Add a bias term: \( E = \text{exp} - \text{bias} \) (i.e., \( \text{exp} = E + \text{bias} \))
  - bias is always \( 2^{k-1} - 1 \), where \( k \) is number of exponent bits

- **Example when** \( k = 3 \):
  - bias = 3
  - If \( E = -2 \), **exp** is 1 (001₂)
  - Reserve 000 and 111 for other purposes (more on this later)
  - We can now represent exponents from \(-2 \, (\text{exp} \, 001)\) to 3 (\text{exp} \, 110)
### 6-bit Floating Point Example

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

<table>
<thead>
<tr>
<th>s</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
6-bit Floating Point Example

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

- \( s \) is the sign bit, \( exp \) is the exponent, \( frac \) is the fraction.
- \( s \) is 1, \( exp \) is 3, \( frac \) is 2.

- \( frac \) has 2 bits, append them after “1.” to form \( M \).
  - \( frac = 10 \) implies \( M = 1.10 \).
6-bit Floating Point Example

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

<table>
<thead>
<tr>
<th>s</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

• *frac* has 2 bits, append them after “1.” to form M
  • *frac* = 10 implies M = 1.10

• Putting it Together: An Example:

\[-10.1_2 = (-1)^1 1.01 \times 2^1\]
6-bit Floating Point Example

- $v = (-1)^s M 2^E$
- $s = 1$
- $exp = 3$
- $frac = 2$

• $frac$ has 2 bits, append them after “1.” to form $M$
  - $frac = 10$ implies $M = 1.10$

• Putting it Together: An Example:

$$-10.1_2 = (-1)^1 \times 1.01 \times 2^1$$
6-bit Floating Point Example

- $v = (\pm 1)^s \times M \times 2^E$

1. **$frac$** has 2 bits, append them after “1.” to form $M$
   - $frac = 10$ implies $M = 1.10$

2. Putting it Together: An Example:

   
   $-10.1_{2} = (-1)^1 \times 1.01 \times 2^1$
6-bit Floating Point Example

- Expression: \( v = (\neg1)^s \cdot M \cdot 2^E \)

  - **frac** has 2 bits, append them after “1.” to form \( M \)
    - **frac** = 10 implies \( M = 1.10 \)
  - Putting it Together: An Example:

\[
-10.1_2 = (-1)^{1} \cdot 1.01 \times 2^{1}
\]
6-bit Floating Point Example

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

- \( frac \) has 2 bits, append them after “1.” to form M
  - \( frac = 10 \) implies \( M = 1.10 \)
- Putting it Together: An Example:

\[-10.1_2 = (-1)^1 \cdot 1.01 \times 2^1\]
6-bit Floating Point Example

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

- \(\text{frac}\) has 2 bits, append them after “1.” to form M
  - \(\text{frac} = 10\) implies \(M = 1.10\)
- Putting it Together: An Example:

\[-10.1_2 = (-1)^1 \ 1.01 \times 2^1\]
6-bit Floating Point Example

\[ v = (-1)^s \, M \, 2^E \]

- **frac** has 2 bits, append them after “1.” to form M
  - **frac** = 10 implies M = 1.10
- Putting it Together: An Example:

\[ -10.1_2 = (-1)^1 \, 1.01 \times 2^1 \]
6-bit Floating Point Example

\[ v = (-1)^s \, M \, 2^E \]

- \( \frac{1}{16} \) has 2 bits, append them after “1.” to form M
  - \( \frac{1}{10} = 10 \) implies \( M = 1.10 \)

- Putting it Together: An Example:

\[ -10.1_2 = (-1)^1 \, 1.01 \times 2^1 \]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>s</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1 100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2 101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3 110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4 111</td>
</tr>
</tbody>
</table>

0

+∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0 \[\rightarrow\] +\(\infty\)
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0

+\infty
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0 \[ \infty \]
# Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

- \(\uparrow\) indicates +\(\infty\)
- \(\downarrow\) indicates 0
- \(\uparrow\uparrow\) indicates 1/4
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

-1/4
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

-∞ - 0 - 1/4 - 1/2 - +∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

-1/2

0

1/4

+∞
Representable Numbers (Positive Only)

\[ v = (-1)^s M 2^E \]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/2
0
1
1/4
+∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/2

0

1

2

+∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

\[ \begin{array}{c}
1/2 \\
0 \\
1/4 \\
0 \\
1 \\
2 \\
+\infty
\end{array} \]
Representable Numbers (Positive Only)

\( v = (-1)^s M 2^E \)

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0 1 2 4

1/4 1/2 \( + \infty \)
Representable Numbers (Positive Only)

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

\[
\begin{array}{cccc}
E & \text{exp} & E & \text{exp} \\
-3 & 000 & 1 & 100 \\
-2 & 001 & 2 & 101 \\
-1 & 010 & 3 & 110 \\
0 & 011 & 4 & 111 \\
\end{array}
\]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/2
0
1/4
0
1
2
4
8
+∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

\[ 0 \quad 1 \quad 2 \quad 4 \quad 8 \quad +\infty \]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/2 1 2 4 8 1.01 x 2^3 +\infty

0 1/4
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
</tr>
</tbody>
</table>

Graph showing representable numbers:

- 0
- 1/4
- 1
- 1/2
- 2
- 4
- 8
- 10
- +\infty
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>0</th>
<th>110</th>
<th>10</th>
</tr>
</thead>
</table>

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

\[ 0 \quad 1 \quad 2 \quad 4 \quad 8 \quad 10 \quad +\infty \]

0

1/4

0

1

2

4

8

10

+$\infty$
Representable Numbers (Positive Only)

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

\[ \begin{array}{|c|c|c|}
\hline
E & \text{exp} & E & \text{exp} \\
\hline
-3 & 000 & 1 & 100 \\
-2 & 001 & 2 & 101 \\
-1 & 010 & 3 & 110 \\
0 & 011 & 4 & 111 \\
\hline
\end{array} \]

\[ 1.10 \times 2^3 \]
Representable Numbers (Positive Only)

\[ v = (-1)^s \, M \, 2^E \]

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing representable numbers with various values of \( E \) and \( \exp \).
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

\[ \begin{align*}
0 & \quad 110 \quad 11 \\
1/2 & \quad 1 \quad 2 \quad 4 \\
1/4 & \quad 8 \quad 10 \quad 12 \\
+\infty & 
\end{align*} \]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

\[ \frac{1}{4} \quad \frac{1}{2} \quad 1 \quad 2 \quad 4 \quad 8 \quad 10 \quad 12 \quad 14 \quad +\infty \]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

![Diagram showing representable numbers with binary representation and scientific notation values]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing representable numbers along the number line with binary representations and exponent values.
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/2

0 1/4 1 2 4 8 10 12 14 +\infty

\[ 1.01 \times 2^2 \]
Representable Numbers (Positive Only)

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

-3 000 1 100
-2 001 2 101
-1 010 3 110
0 011 4 111

0 1 2 4 5 8 10 12 14 +\infty

1/4 1/2
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0 \[ \rightsquigarrow \] 1/4

0 \[ \rightsquigarrow \] 1/2

0 \[ \rightsquigarrow \] 1

0 \[ \rightsquigarrow \] 2

0 \[ \rightsquigarrow \] 4

0 \[ \rightsquigarrow \] 5

0 \[ \rightsquigarrow \] 8

0 \[ \rightsquigarrow \] 10

0 \[ \rightsquigarrow \] 12

0 \[ \rightsquigarrow \] 14

0 \[ \rightsquigarrow \] +\infty
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing representable numbers with binary representation and a range from 0 to 1/4, 1/2, 5, 1.10 x 2^2, 8, 10, 12, 14, and +∞.
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

\[ 0 \quad 101 \quad 10 \]

\[ 0.5 \quad 1 \quad 2 \quad 4 \quad 5 \quad 6 \quad 8 \quad 10 \quad 12 \quad 14 \quad +\infty \]
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/2

0

1/4

1

2

4

5

6

8

10

12

14

+\infty
Representable Numbers (Positive Only)

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

- \( M = 0, 1 \)
- \( 0 \leq E < 4 \)

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Number Line:
- 1/4
- 1/2
- 1
- 2
- 4
- 5
- 6
- 8
- 10
- 12
- 14
- +\infty

1.11 x 2^2
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

1/4 0 1 2 4 6 8 10 12 14 +∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing representable numbers with a scale from 0 to +∞.
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing numbers on a line with binary representations and exponents.
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0 1 2 4 5 6 7 8 10 12 14 +∞

0 1/4 1/2 1 2 4 5 6 7 8 10 12 14 +∞
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing representable numbers on a number line.
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

0 010 11

0 1 2 4 5 6 7 8 10 12 14 +∞
Representable Numbers (Positive Only)

\( v = (-1)^s M 2^E \)

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

Diagram showing representable numbers with markers for 0, 1/4, 1/2, 1, 2, 4, 5, 6, 7, 8, 10, 12, 14, and +∞.
### Representable Numbers (Positive Only)

\[ v = (-1)^s \, M \, 2^E \]

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

![Number Representation Diagram](image)
Representable Numbers (Positive Only)

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>

---

Diagram showing representable numbers with binary representation.
Representable Numbers (Positive Only)

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

- Uneven interval (c.f., fixed interval in fixed-point)
  - More dense toward 0, sparser toward infinite
  - Allow encoding small and large numbers at the same time
Representable Numbers (Positive Only)

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

- Uneven interval (c.f., fixed interval in fixed-point)
  - More dense toward 0, sparser toward infinite
  - Allow encoding small and large numbers at the same time
$v = (-1)^s M 2^E$

Representable Numbers (Positive Only)

- $s$  exp  frac
- $-3$  000  $E -3$
- $-2$  001  $E -2$
- $-1$  010  $E -1$
- $0$   011  $E 0$

- $E$  exp  $E$  exp
- $-3$  000  $1$  100
- $-2$  001  $2$  101
- $-1$  010  $3$  110
- $0$   011  $4$  111

0  1/4  3/8  1/2  1

$0$  5/16  7/16  1
Representable Numbers (Positive Only)

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

Unrepresented small numbers

\[ \frac{1}{4}, \frac{3}{8}, \frac{1}{2} \]

\[
\begin{array}{ccc}
E & \text{exp} & E & \text{exp} \\
-3 & 000 & 1 & 100 \\
-2 & 001 & 2 & 101 \\
-1 & 010 & 3 & 110 \\
0 & 011 & 4 & 111 \\
\end{array}
\]
Representable Numbers (Positive Only)

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

- Underflow: always round to 0 is inelegant
Representable Numbers (Positive Only)

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

- Underflow: always round to 0 is inelegant
Representable Numbers (Positive Only)

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

- Underflow: always round to 0 is inelegant

Unrepresented small numbers

0 1/8 1/4 3/8 1/2 5/16 7/16 1

-3 000 1 100
-2 001 2 101
-1 010 3 110
0 011
Representable Numbers (Positive Only)

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

- Underflow: always round to 0 is inelegant
- Using 000 for exp would only postpone the problem rather than solving it

Unrepresented small numbers

0 1/8 5/16 7/16 1/4 3/8 1/2 1
Subnormal (De-normalized) Numbers

\[ v = (-1)^s \, M \, 2^E \]

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing when \( \exp = 0 \) (subnormal numbers)
Subnormal (De-normalized) Numbers

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

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing when \( E = 0 \) (subnormal numbers)
Subnormal (De-normalized) Numbers

$v = (-1)^s M \ 2^E$

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing \textbf{when} $exp = 0$ (subnormal numbers)
- $E = (exp + 1) - bias$ (instead of $exp - bias$)
- $M = 0.frac$ (instead of $1.frac$)
Subnormal (De-normalized) Numbers

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

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing when \( exp = 0 \) (subnormal numbers)
- \( E = (exp + 1) - \text{bias} \) (instead of \( exp - \text{bias} \))
- \( M = 0.frac \) (instead of 1.frac)

\[
\begin{array}{c|c|c|c}
\text{E} & \text{exp} & \text{E} & \text{exp} \\
-3 & 000 & 1 & 100 \\
-2 & 001 & 2 & 101 \\
-1 & 010 & 3 & 110 \\
0 & 011 & 4 & 111 \\
\end{array}
\]

\[ 0.0001 \times 2^{0+1-3} = 1/16 \]
Subnormal (De-normalized) Numbers

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

- Idea: Evenly divide between 0 and 1/4 rather than exponentially decreasing \( \text{when } exp = 0 \) (subnormal numbers)
- \( E = (exp + 1) - \text{bias} \) (instead of \( exp - \text{bias} \))
- \( M = 0.frac \) (instead of \( 1.frac \))
- Subnormal numbers allow graceful underflow

\[
\begin{array}{|c|c|}
\hline
E & \text{exp} \\
\hline
-3 & 000 \\
-2 & 001 \\
-1 & 010 \\
0 & 011 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
E & \text{exp} \\
\hline
1 & 100 \\
2 & 101 \\
3 & 110 \\
4 & 111 \\
\hline
\end{array}
\]

\[
\begin{align*}
\frac{1}{16} & \quad \frac{1}{8} & \quad \frac{1}{4} & \quad \frac{3}{8} & \quad \frac{1}{2} \\
0 & \quad 1/16 & \quad 3/16 & \quad 5/16 & \quad 7/16 & \quad 1
\end{align*}
\]

\[ (-1)^0 \cdot 0.01 \times 2^{(0+1-3)} = \frac{1}{16} \]
Special Values

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

<table>
<thead>
<tr>
<th>E</th>
<th>exp</th>
<th>E</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>000</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>-2</td>
<td>001</td>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>0</td>
<td>011</td>
<td>4</td>
<td>111</td>
</tr>
</tbody>
</table>
• There are many special values in scientific computing
  • +/- $\infty$, NaNs (0 / 0, 0 / $\infty$, $\infty$ / $\infty$, ...), etc.
There are many special values in scientific computing

- +/- ∞, NaNs (0 / 0, 0 / ∞, ∞ / ∞, …), etc.

exp = 111 is reserved to represent these numbers
There are many special values in scientific computing:

- +/- ∞, NaNs (0 / 0, 0 / ∞, ∞ / ∞, …), etc.
- exp = 111 is reserved to represent these numbers.
There are many special values in scientific computing:

- +/- \(\infty\), NaNs (0 / 0, 0 / \(\infty\), \(\infty\) / \(\infty\), ...), etc.

exp = 111 is reserved to represent these numbers.

exp = 111, frac = 000

- +/- \(\infty\) (depending on the s bit). Overflow results.
- Arithmetic on \(\infty\) is exact: \(1.0/0.0 = -1.0/-0.0 = +\infty\), \(1.0/-0.0 = -\infty\)
There are many special values in scientific computing:

- \(+/- \infty\), NaNs (0 / 0, 0 / \(\infty\), \(\infty / \infty\), ...), etc.

exp = 111 is reserved to represent these numbers.

exp = 111, frac = 000

- \(+/- \infty\) (depending on the s bit). Overflow results.
- Arithmetic on \(\infty\) is exact: 1.0/0.0 = −1.0/−0.0 = \(+\infty\), 1.0/−0.0 = −\(\infty\)

exp = 111, frac \(!=\) 000

- Represent Not-a-Numbers (e.g., sqrt(−1), \(\infty - \infty\), \(\infty \times 0\))
Visualization: Floating Point Encodings

-∞  -Normalized  -Denorm  +Denorm  +Normalized  +∞

NaN

-0  +0

NaN
Visualization: Floating Point Encodings

Infinite Amount of Real Numbers
Visualization: Floating Point Encodings

Infinite Amount of Real Numbers

Finite Amount of Floating Point Numbers
Visualization: Floating Point Encodings

-∞  -Normalized  -Denorm  +Denorm  +Normalized  +∞

NaN  NaN

-0  +0

Infinite Amount of Real Numbers

Sparse

Finite Amount of Floating Point Numbers

Sparse
Visualization: Floating Point Encodings

Infinite Amount of Real Numbers

Finite Amount of Floating Point Numbers
Visualization: Floating Point Encodings

Infinite Amount of Real Numbers

Finite Amount of Floating Point Numbers

Questions?
Today: Floating Point

• Background: Fractional binary numbers and fixed-point
• Floating point representation
• IEEE 754 standard
• Rounding, addition, multiplication
• Floating point in C
• Summary
IEEE Floating Point

• IEEE Standard 754
  • Established in 1985 as uniform standard for floating point arithmetic
    • Before that, many idiosyncratic formats
  • Supported by all major CPUs (and even GPUs and other processors)

• Driven by numerical concerns
  • Nice standards for rounding, overflow, underflow
  • Hard to make fast in hardware
    • Numerical analysts predominated over hardware designers in defining standard
IEEE 754 Standard Precision Options

• Single precision: 32 bits

```
s | exp | frac
1  | 8-bit | 23-bit
```

• Double precision: 64 bits

```
s | exp | frac
1  | 11-bit | 52-bit
```
Single Precision (32-bit) Example

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

bias = \(2^{(8-1)}-1 = 127\)

<table>
<thead>
<tr>
<th>s</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8-bit</td>
<td>23-bit</td>
</tr>
</tbody>
</table>
Single Precision (32-bit) Example

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

bias = \(2^{(8-1)}-1 = 127\)

\[
\begin{array}{ccc}
  \text{s} & \text{exp} & \text{frac} \\
  1 & \text{8-bit} & \text{23-bit} \\
\end{array}
\]

\[
15213_{10} = 11101101101101_2
\]

\[
= (-1)^0 1.1101101101101_2 \times 2^{13}
\]
## Single Precision (32-bit) Example

$$v = (-1)^s M 2^E$$

### Calculation

- **Decimal to Binary**: $15213_{10}$
  - Binary: $111011011011012$
  - Normalized: $(-1)^0 1.1101101101101 \times 2^{13}$

### Format

<table>
<thead>
<tr>
<th>0</th>
<th>exp</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8-bit</td>
<td>23-bit</td>
</tr>
</tbody>
</table>

### Bias

$$bias = 2^{(8-1)-1} = 127$$
Single Precision (32-bit) Example

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

bias = \( 2^{(8-1)} - 1 = 127 \)

\[
\begin{array}{c|c|c}
0 & \text{exp} & \text{frac} \\
1 & 8\text{-bit} & 23\text{-bit} \\
\end{array}
\]

\[ 15213_{10} = 111011011011012 \\
= (-1)^0 1.11011011011012 \times 2^{13} \]
Single Precision (32-bit) Example

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

bias = \(2^{(8-1)} - 1 = 127\)

\[
\begin{array}{ccc}
0 & \text{exp} & \text{frac} \\
1 & 8\text{-bit} & 23\text{-bit} \\
\end{array}
\]

\[
15213_{10} = 11101101101101_2 \\
= (-1)^0 1.1101101101101_2 \times 2^{13} \\
\]

exp = E + bias = \(140_{10}\)
Single Precision (32-bit) Example

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

bias = \(2^{(8-1)} - 1 = 127\)

\[
15213_{10} = 11101101101101_{2}
\]

\[
= (-1)^0 \cdot 1.1101101101101_{2} \times 2^{13}
\]

exp = \(E + \text{bias} = 140_{10}\)
Single Precision (32-bit) Example

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

\[ \text{bias} = 2^{(8-1)} - 1 = 127 \]

\[ \begin{array}{ccc}
0 & 10001100 & \text{frac} \\
1 & \text{8-bit} & \text{23-bit}
\end{array} \]

\[ 15213_{10} = 11101101101101101_2 \]

\[ = (-1)^0 1.1101101101101101_2 \times 2^{13} \]

\[ \exp = E + \text{bias} = 140_{10} \]
Single Precision (32-bit) Example

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

bias = \(2^{(8-1)}-1 = 127\)

15213\(_{10}\) = 11101101101101\(_2\)

= (-1)^0 \(1.1101101101101\)\(_2\) \times 2^{13}

\exp = E + \text{bias} = 140\(_{10}\)
Today: Floating Point

- Background: Fractional binary numbers and fixed-point
- Floating point representation
- IEEE 754 standard
- Rounding, addition, multiplication
- Floating point in C
- Summary
Floating Point Operations: Basic Idea

• Basic idea
  • We perform the operation & produce the infinitely precise result
  • Make it fit into desired precision
    • Possibly overflow if exponent too large
    • Possibly round to fit into frac

• \( x +_f y = \text{Round}(x + y) \)

• \( x \times_f y = \text{Round}(x \times y) \)
Rounding Modes
Rounding Modes

• Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
Rounding Modes

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)

- Directed rounding:
  - Towards zero (chop)
  - Round down (-∞)
  - Round up (+∞)
Rounding Modes

• Default: To nearest; if equally near, then to the one having an even least significant digit (bit)

• Directed rounding:
  • Towards zero (chop)
  • Round down (-∞)
  • Round up (+∞)

<table>
<thead>
<tr>
<th>Rounding Mode</th>
<th>1.40</th>
<th>1.60</th>
<th>1.50</th>
<th>2.50</th>
<th>-1.50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Towards zero</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>Round down (-∞)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>-2</td>
</tr>
<tr>
<td>Round up (+∞)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>Nearest even (default)</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-2</td>
</tr>
</tbody>
</table>
Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for \( \text{frac} \)
Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for $frac$

<table>
<thead>
<tr>
<th>Precise Value</th>
<th>Rounded Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000011</td>
<td>1.000</td>
<td>1.000 is the nearest (down)</td>
</tr>
<tr>
<td>1.000110</td>
<td>1.001</td>
<td>1.001 is the nearest (up)</td>
</tr>
<tr>
<td>1.000100</td>
<td>1.000</td>
<td>1.000 is the nearest even (down)</td>
</tr>
<tr>
<td>1.001100</td>
<td>1.010</td>
<td>1.010 is the nearest even (up)</td>
</tr>
</tbody>
</table>
### Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for $frac$

<table>
<thead>
<tr>
<th>Precise Value</th>
<th>Rounded Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000011</td>
<td>1.000</td>
<td>1.000 is the nearest (down)</td>
</tr>
<tr>
<td>1.000110</td>
<td>1.001</td>
<td>1.001 is the nearest (up)</td>
</tr>
<tr>
<td>1.000100</td>
<td>1.000</td>
<td>1.000 is the nearest even (down)</td>
</tr>
<tr>
<td>1.001100</td>
<td>1.010</td>
<td>1.010 is the nearest even (up)</td>
</tr>
</tbody>
</table>
Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for frac

<table>
<thead>
<tr>
<th>Precise Value</th>
<th>Rounded Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000011</td>
<td>1.000</td>
<td>1.000 is the nearest (down)</td>
</tr>
<tr>
<td>1.000110</td>
<td>1.001</td>
<td>1.001 is the nearest (up)</td>
</tr>
<tr>
<td>1.000100</td>
<td>1.000</td>
<td>1.000 is the nearest even (down)</td>
</tr>
<tr>
<td>1.001100</td>
<td>1.010</td>
<td>1.010 is the nearest even (up)</td>
</tr>
</tbody>
</table>
Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for $frac$

<table>
<thead>
<tr>
<th>Precise Value</th>
<th>Rounded Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000011</td>
<td>1.000</td>
<td>1.000 is the nearest (down)</td>
</tr>
<tr>
<td>1.000110</td>
<td>1.001</td>
<td>1.001 is the nearest (up)</td>
</tr>
<tr>
<td>1.000100</td>
<td>1.000</td>
<td>1.000 is the nearest even (down)</td>
</tr>
<tr>
<td>1.001100</td>
<td>1.010</td>
<td>1.010 is the nearest even (up)</td>
</tr>
</tbody>
</table>
Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for $frac$

<table>
<thead>
<tr>
<th>Precise Value</th>
<th>Rounded Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000011</td>
<td>1.000</td>
<td>1.000 is the nearest (down)</td>
</tr>
<tr>
<td>1.000110</td>
<td>1.001</td>
<td>1.001 is the nearest (up)</td>
</tr>
<tr>
<td>1.000100</td>
<td>1.000</td>
<td>1.000 is the nearest even (down)</td>
</tr>
<tr>
<td>1.001100</td>
<td>1.010</td>
<td>1.010 is the nearest even (up)</td>
</tr>
</tbody>
</table>
Rounding Modes (Binary Example)

- Default: To nearest; if equally near, then to the one having an even least significant digit (bit)
- Assuming 3 bits for $frac$

<table>
<thead>
<tr>
<th>Precise Value</th>
<th>Rounded Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000011</td>
<td>1.000</td>
<td>1.000 is the nearest (down)</td>
</tr>
<tr>
<td>1.000110</td>
<td>1.001</td>
<td>1.001 is the nearest (up)</td>
</tr>
<tr>
<td>1.000100</td>
<td>1.000</td>
<td>1.000 is the nearest even (down)</td>
</tr>
<tr>
<td>1.001100</td>
<td>1.010</td>
<td>1.010 is the nearest even (up)</td>
</tr>
</tbody>
</table>
Floating Point Addition
Floating Point Addition

\[ (-1)^{s_1}M_1 \ 2^{E_1} \ + \ (-1)^{s_2}M_2 \ 2^{E_2} \]

\[ 1.000 \times 2^{-1} + 11.10 \times 2^{-3} \]
Floating Point Addition

\[ (-1)^{s_1} M_1 \ 2^{E_1} + (-1)^{s_2} M_2 \ 2^{E_2} \]

\[ 1.000 \times 2^{-1} + 11.10 \times 2^{-3} \]

align

\[ 1.000 \times 2^{-1} + 0.111 \times 2^{-1} \]
Floating Point Addition

\[ (-1)^{s_1} M_1 \ 2^{E_1} \ + \ (-1)^{s_2} M_2 \ 2^{E_2} \]

\[ 1.000 \times 2^{-1} \ + \ 1.110 \times 2^{-3} \]

\[ 1.000 \times 2^{-1} \ + \ 0.111 \times 2^{-1} \]

\[ \text{add} \]

\[ 1.111 \times 2^{-1} \]
Floating Point Addition

- \((-1)^{s_1} M_1 \ 2^{E_1} + (-1)^{s_2} M_2 \ 2^{E_2}\)

- Exact Result: \((-1)^s M \ 2^E\)
  - Sign \(s\), significand \(M\):
    - Result of signed align & add
  - Exponent \(E\): \(E_1\)
    - Assume \(E_1 > E_2\)

\[
1.000 \times 2^{-1} + 11.10 \times 2^{-3}
\]

\[
1.000 \times 2^{-1} + 0.111 \times 2^{-1}
\]

\[
1.111 \times 2^{-1}
\]
Floating Point Addition

• \((-1)^s_1 \times M_1 \times 2^{E_1} + (-1)^s_2 \times M_2 \times 2^{E_2}\)

• Exact Result: \((-1)^s \times M \times 2^E\)
  • Sign \(s\), significand \(M\):
    • Result of signed align & add
  • Exponent \(E\): \(E_1\)
    • Assume \(E_1 > E_2\)

• Fixing
  • If \(M \geq 2\), shift \(M\) right, increment \(E\)
  • If \(M < 1\), shift \(M\) left \(k\) positions, decrement \(E\) by \(k\)
  • Overflow if \(E\) out of range
  • Round \(M\) to fit \(frac\) precision

\[
1.000 \times 2^{-1} + 11.10 \times 2^{-3}
\]

\[
1.000 \times 2^{-1} + 0.111 \times 2^{-1}
\]

\[
1.111 \times 2^{-1}
\]
Mathematical Properties of FP Add
Mathematical Properties of FP Add

- Commutative?
Mathematical Properties of FP Add

- Commutative? Yes
Mathematical Properties of FP Add

• Commutative? Yes
• Associative?
Mathematical Properties of FP Add

- Commutative? Yes
- Associative? No
  - Overflow and inexactness of rounding
    - \((3.14+1e10)-1e10 = 0\), \(3.14+(1e10-1e10) = 3.14\)
Mathematical Properties of FP Add

• Commutative? Yes
• Associative? No
  • Overflow and inexactness of rounding
    • $(3.14+1e10)-1e10 = 0$, $3.14+(1e10-1e10) = 3.14$
• 0 is additive identity?
Mathematical Properties of FP Add

- Commutative? Yes
- Associative? No
  - Overflow and inexactness of rounding
    - $(3.14+1e10)-1e10 = 0$, $3.14+(1e10-1e10) = 3.14$
- 0 is additive identity? Yes
Mathematical Properties of FP Add

- Commutative? Yes
- Associative? No
  - Overflow and inexactness of rounding
    - $(3.14+1e10)-1e10 = 0, 3.14+(1e10-1e10) = 3.14$
- $0$ is additive identity? Yes
- Every element has additive inverse (negation)?
Mathematical Properties of FP Add

• Commutative? Yes
• Associative? No
  • Overflow and inexactness of rounding
  • \((3.14+1e10)-1e10 = 0\), \(3.14+(1e10-1e10) = 3.14\)
• 0 is additive identity? Yes
• Every element has additive inverse (negation)? Almost
  • Except for infinities & NaNs
Mathematical Properties of FP Add

- Commutative? Yes
- Associative? No
  - Overflow and inexactness of rounding
    - 
      \[(3.14+1e10)-1e10 = 0, \ 3.14+(1e10-1e10) = 3.14\]
- 0 is additive identity? Yes
- Every element has additive inverse (negation)? Almost
  - Except for infinities & NaNs
- Monotonicity: \[a \geq b \Rightarrow a+c \geq b+c\]?
Mathematical Properties of FP Add

- Commutative?  Yes
- Associative?  No
  - Overflow and inexactness of rounding
    - $(3.14+1e10)-1e10 = 0$, $3.14+(1e10-1e10) = 3.14$
- 0 is additive identity?  Yes
- Every element has additive inverse (negation)?  Almost
  - Except for infinities & NaNs
- Monotonicity: $a \geq b \Rightarrow a+c \geq b+c$?  Almost
  - Except for infinities & NaNs
Floating Point Multiplication
Floating Point Multiplication

\[ (-1)^{s_1} M_1 \ 2^{E_1} \times (-1)^{s_2} M_2 \ 2^{E_2} \]
Floating Point Multiplication

• \((-1)^{s_1} M_1 \ 2^{E_1} \times (-1)^{s_2} M_2 \ 2^{E_2}\)

• Exact Result: \((-1)^s M \ 2^E\)
  - Sign s: \(s_1 \land s_2\)
  - Significand M: \(M_1 \times M_2\)
  - Exponent E: \(E_1 + E_2\)
Floating Point Multiplication

- $(-1)^{s_1} M_1 2^{E_1} \times (-1)^{s_2} M_2 2^{E_2}$
- Exact Result: $(-1)^s M 2^E$
  - Sign $s$: $s_1 \land s_2$
  - Significand $M$: $M_1 \times M_2$
  - Exponent $E$: $E_1 + E_2$
- Fixing
  - If $M \geq 2$, shift $M$ right, increment $E$
  - If $E$ out of range, overflow
  - Round $M$ to fit frac precision
Floating Point Multiplication

• \((-1)^{s_1} M_1 \ 2^{E_1} \times (-1)^{s_2} M_2 \ 2^{E_2}\)

• Exact Result: \((-1)^s M \ 2^E\)
  • Sign s: \(s_1 \ ^\land \ s_2\)
  • Significand M: \(M_1 \times M_2\)
  • Exponent E: \(E_1 + E_2\)

• Fixing
  • If \(M \geq 2\), shift M right, increment E
  • If E out of range, overflow
  • Round M to fit frac precision

• Implementation
  • Biggest chore is multiplying significands
Mathematical Properties of FP Mult
Mathematical Properties of FP Mult

• Multiplication Commutative?
Mathematical Properties of FP Mult

• Multiplication Commutative?  Yes
Mathematical Properties of FP Mult

• Multiplication Commutative? Yes
• Multiplication is Associative?
Mathematical Properties of FP Mult

• Multiplication Commutative?  
  Yes

• Multiplication is Associative?  
  No

  • Possibility of overflow, inexactness of rounding
  • Ex: \((1e20*1e20)*1e-20= \text{inf}, \ 1e20*(1e20*1e-20)= 1e20\)
Mathematical Properties of FP Mult

- Multiplication Commutative? Yes
- Multiplication is Associative? No
  - Possibility of overflow, inexactness of rounding
  - Ex: \((1e20 \times 1e20) \times 1e-20 = \text{inf}\), \(1e20 \times (1e20 \times 1e-20) = 1e20\)
- 1 is multiplicative identity?
Mathematical Properties of FP Mult

- Multiplication Commutative? Yes
- Multiplication is Associative? No
  - Possibility of overflow, inexactness of rounding
  - Ex: \((1e20*1e20)*1e-20= \inf\), \(1e20*(1e20*1e-20)= 1e20\)
- 1 is multiplicative identity? Yes
Mathematical Properties of FP Mult

• Multiplication Commutative? Yes
• Multiplication is Associative? No
  - Possibility of overflow, inexactness of rounding
  - Ex: \((1e20*1e20)*1e-20= \text{inf}, 1e20*(1e20*1e-20)= 1e20\)
• 1 is multiplicative identity? Yes
• Multiplication distributes over addition?
Mathematical Properties of FP Mult

• Multiplication Commutative?  \(\text{Yes}\)
• Multiplication is Associative?  \(\text{No}\)
  • Possibility of overflow, inexactness of rounding
  • Ex: \((1e20*1e20)*1e-20= \text{inf}, 1e20*(1e20*1e-20)= 1e20\)
• 1 is multiplicative identity?  \(\text{Yes}\)
• Multiplication distributes over addition?  \(\text{No}\)
  • Possibility of overflow, inexactness of rounding
  • \(1e20*(1e20-1e20)= 0.0, 1e20*1e20 - 1e20*1e20 = \text{NaN}\)
Mathematical Properties of FP Mult

- Multiplication Commutative?  Yes
- Multiplication is Associative?  No
  - Possibility of overflow, inexactness of rounding
  - Ex: \((1e20 \times 1e20) \times 1e-20 = \text{inf}, 1e20 \times (1e20 \times 1e-20) = 1e20\)
- 1 is multiplicative identity?  Yes
- Multiplication distributes over addition?  No
  - Possibility of overflow, inexactness of rounding
  - \(1e20 \times (1e20 - 1e20) = 0.0, 1e20 \times 1e20 - 1e20 \times 1e20 = \text{NaN}\)
- Monotonicity: \(a \geq b \text{ and } c \geq 0 \Rightarrow a \times c \geq b \times c?\)
Mathematical Properties of FP Mult

- Multiplication Commutative? \[ \text{Yes} \]
- Multiplication is Associative? \[ \text{No} \]
  - Possibility of overflow, inexactness of rounding
  - Ex: \((1e20 \times 1e20) \times 1e-20 = \text{inf}, 1e20 \times (1e20 \times 1e-20) = 1e20\)
- 1 is multiplicative identity? \[ \text{Yes} \]
- Multiplication distributes over addition? \[ \text{No} \]
  - Possibility of overflow, inexactness of rounding
  - \(1e20 \times (1e20 - 1e20) = 0.0, 1e20 \times 1e20 - 1e20 \times 1e20 = \text{NaN}\)
- Monotonicity: \(a \geq b \& c \geq 0 \implies a \times c \geq b \times c? \quad \text{Almost}\)
  - Except for infinities & NaNs
Today: Floating Point

• Background: Fractional binary numbers and fixed-point
• Floating point representation
• IEEE 754 standard
• Rounding, addition, multiplication
• Floating point in C
• Summary
Fixed point (implicit binary point)

SP floating point

DP floating point

<table>
<thead>
<tr>
<th>C Data Type</th>
<th>Bits</th>
<th>Max Value</th>
<th>Max Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>8</td>
<td>$2^7 - 1$</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16</td>
<td>$2^{15} - 1$</td>
<td>32767</td>
</tr>
<tr>
<td>int</td>
<td>32</td>
<td>$2^{31} - 1$</td>
<td>2147483647</td>
</tr>
<tr>
<td>long</td>
<td>64</td>
<td>$2^{31} - 1$</td>
<td>~$9.2 \times 10^{18}$</td>
</tr>
<tr>
<td>float</td>
<td>32</td>
<td>$(2 - 2^{-23}) \times 2^{127}$</td>
<td>~$3.4 \times 10^{38}$</td>
</tr>
<tr>
<td>double</td>
<td>64</td>
<td>$(2 - 2^{-52}) \times 2^{1023}$</td>
<td>~$1.8 \times 10^{308}$</td>
</tr>
</tbody>
</table>
Floating Point in C

• C Guarantees Two Levels
  • `float` single precision
  • `double` double precision

• Conversions/Casting
  • Casting between `int`, `float`, and `double` changes bit representation
  • `double/float → int`
    • Truncates fractional part
    • Like rounding toward zero
    • Not defined when out of range or NaN: Generally sets to TMin
  • `int → double`
    • Exact conversion, as long as int has ≤ 53 bit word size
  • `int → float`
    • Will round according to rounding mode