**Binary equivalent of a real number**

1. Convert the integer part by repeated division by 2 (or some other algorithm)
2. Convert the fraction part by repeated multiplication by 2
3. Join these two parts together to form a binary fixed point number
4. Add a zero exponent to get the number into binary floating point form
5. Normalize by moving the point to obtain a number in the form $1.fffffffffff \times 2^{xx}$

**Example:** Convert 200.6875 to normalized binary floating point form.

<table>
<thead>
<tr>
<th>Step 1: Integer Part</th>
<th>Step 2: Fraction Part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeated division by 2</td>
<td>Repeated multiplication by 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>200</th>
<th>.6875</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 R 0</td>
<td>1 .375</td>
</tr>
<tr>
<td>50 R 0</td>
<td>0 .75</td>
</tr>
<tr>
<td>25 R 0</td>
<td>1 .5</td>
</tr>
<tr>
<td>12 R 1</td>
<td>1 .0</td>
</tr>
<tr>
<td>6 R 0</td>
<td></td>
</tr>
<tr>
<td>3 R 0</td>
<td></td>
</tr>
<tr>
<td>1 R 1</td>
<td></td>
</tr>
<tr>
<td>0 R 1</td>
<td></td>
</tr>
</tbody>
</table>

Step 3: Fixed Point | 11001000.1011 |
Step 4: Floating Point | 11001000.1011 x $2^{0}$ |
Step 5: Normalize | 1.10010001011 x $2^{7}$ |

$200.6875 = _b1.10010001011 \times 2^{7}$
IEEE Floating Point Representations

<table>
<thead>
<tr>
<th>Type</th>
<th>Java</th>
<th>Bits</th>
<th>Range:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short Real</td>
<td>float</td>
<td>32</td>
<td>+/- 3.40 x 10^{38} approx.</td>
</tr>
<tr>
<td>Long Real</td>
<td>double</td>
<td>64</td>
<td>+/- 1.80 x 10^{308} approx.</td>
</tr>
</tbody>
</table>

IEEE Short Real Format

| s | x | x | x | x | x | x | x | f | f | f | f | f | f | f | f | f | f | f | f | f | f | f |

s: Sign bit 0 = positive 1 = negative  
x: 8-bit exponent in excess-127 form  
f: 23-bit fraction; whole-number 1 is not stored, referred to as the hidden bit

Example: Derive the IEEE Short Real format of 200.6875

200.6875 = 1.100100010112 x 2^7 normalized binary floating point

s: 0 positive  
x: 100 0011 0 (127 + 7) stored exponent = actual exponent + 127  
f: 100 1000 1011 0000 0000 0000 fraction bits only, whole part 1 digit is not stored

IEEE Short Real: 0100 0011 0100 1000 1011 0000 0000 0000 x4348 B000

Exercises:
1. -2.2 \rightarrow xC00C CCCC
2. 0.13 \rightarrow x3E05 1EB8
Interpreting IEEE Short-Real
1. Write the hexadecimal in binary (32 bits)
2. Parse into sign, exponent, fraction fields
3. Assemble into binary floating-point
4. Reduce the exponent to 0 to get fixed point
5. Interpret the whole and fraction parts separately to get decimal. Don’t forget the hidden bit!

**Example:** Interpret xC0DA000 as a real number in IEEE Short Real format

\[
\begin{array}{cccccccc}
\text{C} & \text{0} & \text{D} & \text{A} & \text{0} & \text{0} & \text{0} & \text{0} \\
1100 & 0000 & 1101 & 1010 & 0000 & 0000 & 0000 & 0000
\end{array}
\]

<table>
<thead>
<tr>
<th>Step 1: Binary</th>
<th>1100 0000 1101 1010 0000 0000 0000 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2: Parse</td>
<td>1100 0000 1101 1010 0000 0000 0000 0000</td>
</tr>
<tr>
<td></td>
<td>s=1                               x=110000001                               f=101 1010 0000 0000 0000 0000</td>
</tr>
<tr>
<td>Step 3: Floating Point</td>
<td>-1.101101000… x 2^2</td>
</tr>
<tr>
<td>Step 4: Fixed Point</td>
<td>-110.1101</td>
</tr>
<tr>
<td>Step 5: Interpret</td>
<td>- 2^1 + 2^-1 + 2^-2 + 2^-4</td>
</tr>
<tr>
<td></td>
<td>- 4 + 2 + 0.5 + 0.25 + 0.0625</td>
</tr>
<tr>
<td></td>
<td>- 6.8125</td>
</tr>
</tbody>
</table>

**Exercises:**
1. x40AD0000 → 5.40625
2. xC0140000 → -2.3125
3. x3EA00000 → 0.3125
**Special IEEE Short-Real Numbers**

- **Signed Zero**
  - All exponent bits 0
  - All fraction bits 0
  - s 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- **Signed Infinity**
  - All exponent bits are 1
  - All fraction bits 0
  - s 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

- **NaN (Not-a-Number)**
  - All exponent bits are 1
  - At least one fraction bit 1
  - x 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**Terminology: Precision & Range**

- **Precision**
  - Short Real has 23 fraction bits (approx. 7 decimal digits)
  - Long Real has 52 fraction bits (approx. 15 decimal digits)
  - A greater number of stored fraction bits = Greater precision (accuracy)

- **Range**
  - Short Real has 8 exponent bits
  - Long Real has 11 exponent bits
  - A greater number of stored exponent bits = Greater range of values can be stored
  - Short Real: +/- 3.40 x 10^{38} approx.
  - Long Real: +/- 1.80 x 10^{308} approx.