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.ffffffffffff x 2^{xx}

Example: Convert 200.6875 to normalized binary floating point form.

Step 1: Integer Part	**Step 2: Fraction Part**
Repeated division by 2	*Repeated multiplication by 2*
200 100 R **0** 50 R **0** 25 R **0** 12 R **1** 6 R **0** 3 R **0** 1 R **1** 0 R **1**	.6875 **1** .375 **0** .75 **1** .5 **1** .0

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 x 2^7

IEEE Floating Point Representations

Short Real	Java ***float***	32 bits	Range: +/- 3.40 x 10^{38} approx.
Long Real	Java ***double***	64 bits	Range: +/- 1.80 x 10^{308} approx.

IEEE Short Real Format

s	x	x	x	x	x	x	x	x	f	f	f	f	f	f	f	f	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*

<u>*Example*</u>: Derive the IEEE Short Real format of 200.6875

200.6875 = $1.10010001011_2 \times 2^7$ 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

0	1	0	0	0	0	1	1	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

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

<u>*Exercises*</u>:
1. -2.2 → xC00C CCCC
2. 0.13 → 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!

<u>Example</u>: Interpret xC0DA0000 as a real number in IEEE Short Real format

C 0 D A 0 0 0 0
1100 0000 1101 1010 0000 0000 0000 0000

Step 1: Binary	1100 0000 1101 1010 0000 0000 0000 0000
Step 2: Parse	1100 0000 1101 1010 0000 0000 0000 0000 s=1 x=10000001 f=101 1010 0000 0000 0000 0000
Step3: Floating Point	$-1.101101000\ldots \times 2^2$
Step 4: Fixed Point	-110.1101
Step 5: Interpret	- $2^2 + 2^1 + 2^{-1} + 2^{-2} + 2^{-4}$ - 4 + 2 + 0.5 + 0.25 + 0.0625 - 6.8125

<u>Exercises</u>:

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

- **Signed Infinity**

All exponent bits are 1 All fraction bits 0

s	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

- **NaN (Not-a-Number)**

All exponent bits are 1 At least one fraction bit 1

x	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	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.