Week 3: Fixed and Floating Point

Real Numbers:

e.g. 5.139

- 5 → $\frac{1}{10}$'s place
- 1 → $\frac{1}{100}$'s place
- 9 → $\frac{1}{1000}$'s place

What About Other Bases?

- N-digit non-negative number with radix r:

$$D_{N-1} \cdots D_1 D_0 . D_{-1} D_{-2} \cdots = (D_{N-1} \times r^{N-1}) + \cdots + (D_1 \times r^1) + (D_0 \times r^0) + (D_{-1} \times r^{-1}) + (D_{-2} \times r^{-2}) \cdots$$

- The (radix point) separates the integer and the fraction

Fixed-Point Binary:

- The "fixed" position for the radix point is called fixed-point

e.g. $XXXX.XX$

2^{-1} 2^{-2}

$$1001.01_{(2)} = 8 + 1 + \frac{1}{4} = 9.25_{(10)}$$

$$0000.11_{(2)} = \frac{1}{2} + \frac{1}{4} = 0.75_{(10)}$$

Another way to Look at it....

- It is same as a b-bit integer shifted right by 2 bits ★★★

(Dividing by 2^2)

$$\therefore XXXX.XX_{(2)} = XXXXXX_{(2)} / 4$$

e.g. $1001.01_{(2)} = 100101_{(2)} / 4 = 37/4 = 9.25_{(10)}$

Converting to Fixed-Point Binary

e.g. Convert 6.625 to an 8-bit fixed-point binary number with 4 fraction bits

$6.625 - 4 = 2.625 \quad (2^2)$

$2.625 - 2 = 0.625 \quad (2^1)$

$0.625 - 0.5 = 0.125 \quad (2^{-1})$

$0.125 - 0.125 = 0 \quad (2^{-2})$

$\therefore 0110.1010_{(2)}$

☆ The location of the radix-point is defined by the particular format

- It is NOT visible
- It is NOT encoded in the number itself
- You must be told where it is

Fixed V.S Floating-Point

- Fixed-Point: The location of the radix point is defined by the number of format

- Floating-Point: The format defines a way to encode the radix point location into the number
  (The position is not "fixed" in one place)

IEEE 32-Bit Floating-Point

- Divide up the bits of a number into three fields
  (field is a range of bits for a specific purpose)

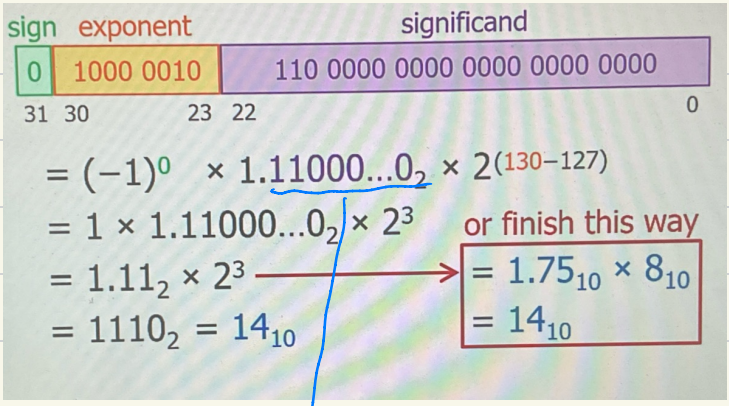
☆ Interpret the number according to the equation:

$$(-1)^{sign} \times 1.significand \times 2^{(exponent - bias)}$$

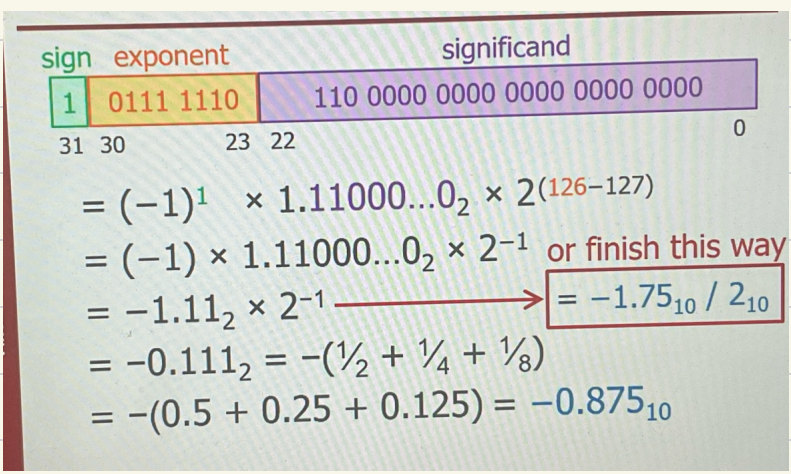
Normalized

Implicit leading 1

e.g



★ Put the significand behind the 1.



Discussion of IEEE Floating-Point

- standardized formats facilitate exchange of data
- The implicit leading one ensures normalization, but prevents representation of 0. ☆☆☆

IEEE formats have special cases for:

- zero (All bits 0)
- positive and negative infinity (+INF / -INF)
- various not-a-number (NaN) values

Format Comparison

- Three **unsigned** 8-bit binary formats:
  - Integer XXXXXXXX
  - Fixed-point with 4 fractional bits
    XXXX.XXXX
  - Floating-point with 4 exponent bits
    - The exponent is unsigned and there is no bias
    - It does have an implicit leading 1 (hidden bit)

XXXX XXXX → 1.significand × $2^{exponent}$

exponent significand

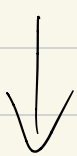
Format Comparison

		XXXXXXXX	XXXX.XXXX	$1.XXXX \times 2^{XXXX}$
		Integer	**Fixed**	**Float**
Range	**From**	0_{10}	0_{10}	1_{10}
	To	255_{10}	15.9375_{10}	$63{,}488_{10}$
Precision		8 bits	8 bits	**5** bits
Unique Values		256	256	256

- **Range:** from the leftmost to the rightmost representable values on the number line
- **Precision:** number of significant digits

ASCII

We need to encode information that humans can understand binary and computers can understand us!



How do computers handle text characters?

Encoding standards

ASCII Text Encoding Standard

- The American Standard Code for Information Interchange (ASCII) was established in 1963
  - ASCII is one of the most common text encodings
- It uses 7 bits to represent 128 different characters
  - Alphanumeric characters A-Z, a-z, 0-9, encoded to allow easy sorting and transformation

100 1100 → 'L'	110 1100 → 'l'	011 0110 → '6'
100 1101 → 'M'	110 1101 → 'm'	011 0111 → '7'

  - Punctuation: #, %, @, etc.
  - Control characters: backspace, end of file, tab, etc.
- It was extended to 8 bits by different companies to support more characters

[To convert to / from ASCII, we use ASCII table]

Other Text Encoding Standards

- 128 characters is not always enough...
  - The Internet is global, and the world has a variety of languages the require a variety of symbols
- Coding standards commonly used today include:
  - Unicode (UTF-16)
    - Originally 16-bit coding scheme – extended to 32 bits (UTF-32)
    - First 128 code values are the ASCII character set
    - A very sophisticated encoding scheme that efficiently represents characters, glyphs, sequences, and more
      - a, á, â, ấ, ä, å, æ, ă, ą, ā, ǎ, ạ, a̋, ā, ǣ, ắ, ằ, ậ
  - UTF-8 (**U**nicode **T**ransformation **F**ormat **8**-bit)
    - Encodes each Unicode character as 1-4 bytes
    - ASCII characters represented by single byte (codes 0-127)
    - Default encoding for XML and dominant Web character set

Additional Notes on Educational Objectives:

- How exponent-bias is used in floating-point expression?

↓

☆ The bias is used to ensure that the exponent is stored as a non-negative value. This simplifies the hardware implementation of floating-point arithmetic.