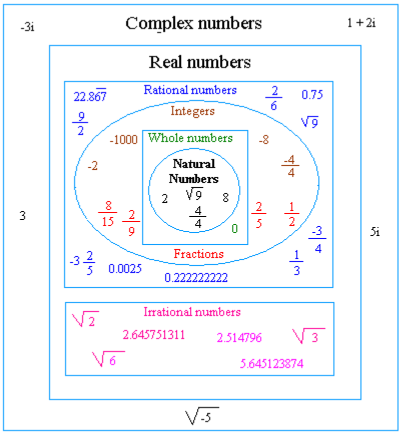
# Floating and Fixed-Point Binary

2020

Ando Ki, Ph.D.
adki@future-ds.com

## Table of contents

# Classification of numbers




- Natural number: 자연수
- Whole number: (정수)
- Integer: 정수
- Rational number: 유리수
- Irrational number: 무리수
- Real number:실수
- Complex number: 복소수

# Number Systems

- PNS (Positional Number System) – a number is represented by a string of digits. Each digit position is weighted by a power of the base or radix.
  - Position of coefficient determines its value.
- Radix - the radix or base is the number of unique digits, including zero, that a positional numeral system uses to represent numbers.
  - Decimal is natural (we count in base 10) - (Radix or Base 10 ) digits {0,1,…,8,9}
  - Binary is used in digital system - (Radix or Base 2 ) digits {0,1}
  - Octal is used for representing multi-bits (group of 3 bits) numbers in digital systems. - (Radix or Base 8=$2^3$) digits {0,1,…,6,7}
  - Hexadecimal is used for representing multi-bits (group of 4 bits) numbers in digital systems- (Radix or Base 16=$2^4$) digits {0,1,…9,A,B,…,F}

- Decimal: 십진수
  - Positional radix 10 code system
  - Coefficient in position is multiplied by radix (10) raised to the power determined by its position, *e.g.,*

$$4*10^3+8*10^2+5*10^1+6*10^0=(4{,}856)_{10}$$

- Binary: 이진수
  - Positional radix 2 code system
  - Two symbols, B = { 0, 1 }
  - Easily implemented using switches
  - Easy to implement in electronic circuitry
  - Algebra invented by George Boole (1815-1864) allows easy manipulation of symbols

$$(0101)_2=0*2^3+1*2^2+0*2^1+1*2^0=(5)_{10}$$

# Positive and negative of binary for integer number

- Positive numbers
  - Unsigned binary
- Negative numbers
  - Sign/magnitude numbers
  - Two's complement

**sign** → | | **magnitude** |

# Numbers with fractions: real number

- Floating-point
  - the binary-point (or decimal-point) floats to the right of the most significant 1
  - use a special form to represent almost fractional real numbers.

    0.00000123 → $1.23 \times 10^{-6}$
    0.1 → $1.0 \times 10^{-1}$
    124.567 → $1.24567 \times 10^{2}$
    1230000.456 → $1.230000456 \times 10^{6}$

| sign | exponent (8-bit) | significant (mantissa) (23-bit) |
|---|---|---|

- Fixed-point
  - The binary-point (or decimal-point) is fixed
  - The binary point is not a part of the representation but is implied.
  - The number of integer and fraction bits must be agreed upon by those generating and those reading the number.

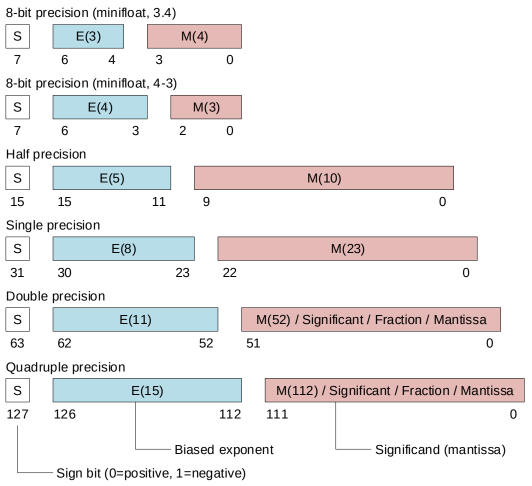
  01101100

  0110.1100

  $2^2 + 2^1 + 2^{-1} + 2^{-2} = 6.75$

- Possible errors
  - truncation error
    - round off errors using floating-point numbers because not all real numbers can be represented accurately
  - overflow error
    - attempting to represent a number that is greater than the upper bound for the given number of bits
  - underflow error
    - attempting to represent a number that is less than the lower bound for the given number of bits

# Format of floating-point: IEEE 754




- For a single-precision number,
  - the exponent is stored in the range 1~254 (0 and 255 have special meanings), and is biased by subtracting 127 to get an exponent value in the range −126~+127.
- For +0.0 of single-precision, it is {1'b0, 8'h00, 23'h0000}, where 23'h0000 means 0.0.
- When all bits of exponent is 1, it can be +/- Infinity or +/-NaN (Not a Number)
  - S=0, E=all 1, M=all 0 ➔ +Infinity.
  - S=0, E=all 1, M=not all 0 ➔ +NaN.
  - S=1, E=all 1, M=all 0 ➔ -Infinity.
  - S=1, E=all 1, M=not all 0 ➔ -NaN.



# How to convert floating to fixed-point

- Fractional decimal to fixed-point binary
  - Converting a fractional number (represented as a decimal) to a fractional binary number works by repeated multiplication by 2.
  - convert C

$$
\begin{aligned}
0.6 * 2 &= 1.2\\
0.2 * 2 &= 0.4\\
0.4 * 2 &= 0.8\\
0.8 * 2 &= 1.6 \quad \Rightarrow 0.1001100110011\ldots_{(2)}\\
0.6 * 2 &= 1.2\\
0.2 * 2 &= 0.4\\
&\vdots
\end{aligned}
$$

- Fixed-point binary to fractional decimal
  - Simply divide a fixed factor
    - It is a shifting the radix point a fixed number of places to the left.
  - Ex: 8-bit width fixed-point binary with 3-bit fractional part.
    - 0011_0101 (0x$35_{16}$, $53_{10}$)
    - 00110.101 (6.626, 53/8 = 53>>3)

0.625*2 = 1.25
0.25 *2 = 0.5
0.5 *2 = 1.0

# Negative fixed-point number

- As with integers, negative fractional numbers can be represented two ways:
  - Sign/magnitude notation
  - Two's complement notation

- Represent $-7.5_{10}$ using an 8-bit binary representation with 4 integer bits and 4 fraction bits.
  - Sign/magnitude:

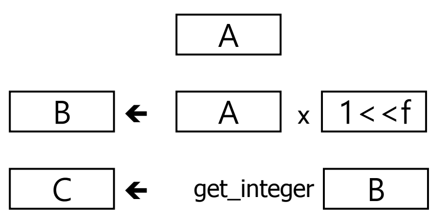
    11111000
  - Two's complement:

    1. +7.5: 01111000
    2. Invert bits: 10000111
    3. Add 1 to lsb: + 1

    10001000

# Convert floating to fixed-point

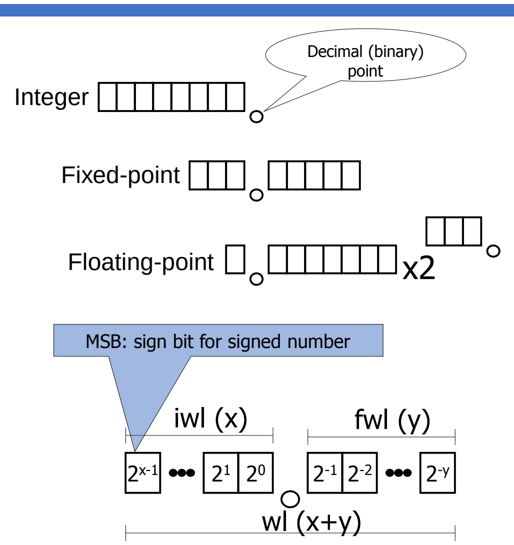
- Say 'A' is real nunber
- 1. multiply 2**f, where f is the number of bits of fractional part
- 2. take integer part (C) of the result (B)
- 3. this B is fixed-point number of 'A' with f-bit fraction



# Project: find fixed-point binary number

- Ex: Represent $6.5_{10}$ using an 8-bit binary representation with 4 integer bits and 4 fraction bits.
- Ex: Represent $6.75_{10}$ using an 8-bit binary representation with 4 integer bits and 4 fraction bits.
- Ex: Represent $7.5_{10}$ using an 8-bit binary representation with 4 integer bits and 4 fraction bits.
- Ex: Represent $-5.645_{10}$ using an 8-bit 2's complement binary representation with 4 integer bits and 4 fraction bits.

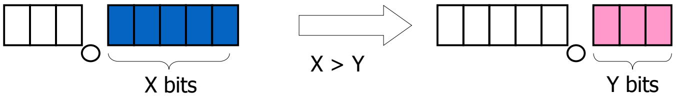
# Integer, floating-point, fixed-point



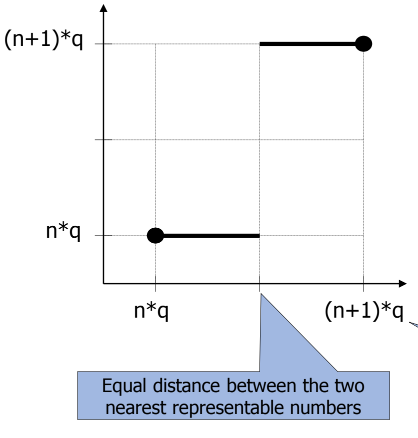

- Integer
  - a kind of fixed point type
  - Manipulation is fast and cheap
  - Poor for modeling continuous real-world behavior
- Floating-point
  - Better approximation to real number
  - Good for modeling continuous behavior
  - Manipulation is slow and expensive
  - Requires more hardware to implement the functionality
- Fixed-point
  - Used in many signal processing applications
  - Requires less hardware to implement the functionality than floating-point

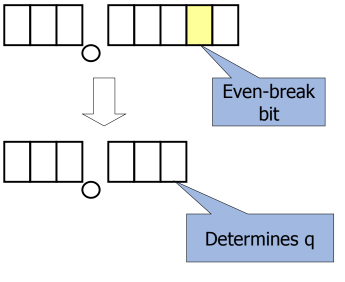
# Fixed-point quantization

- Operations performed on fixed-point data types are done using arbitrary precision.
- The resulting operand is cast to fit the fixed-point data type object.
- The quantization (rounding/truncation) is applied by the casting operation.
- Quantization modes:
  - Truncation
  - Rounding



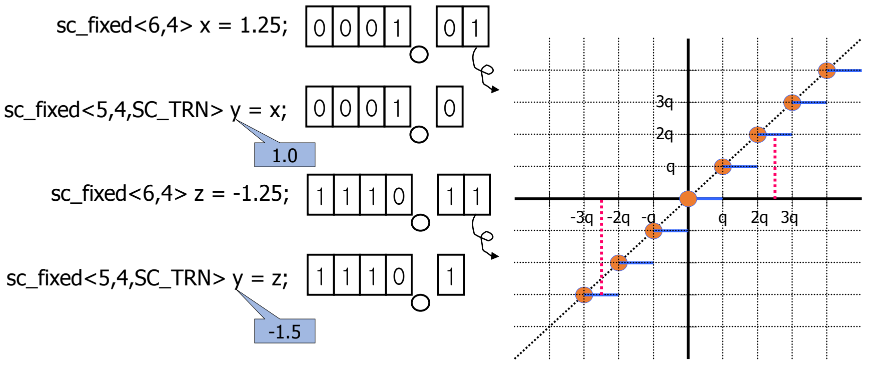

# Fixed-point rounding



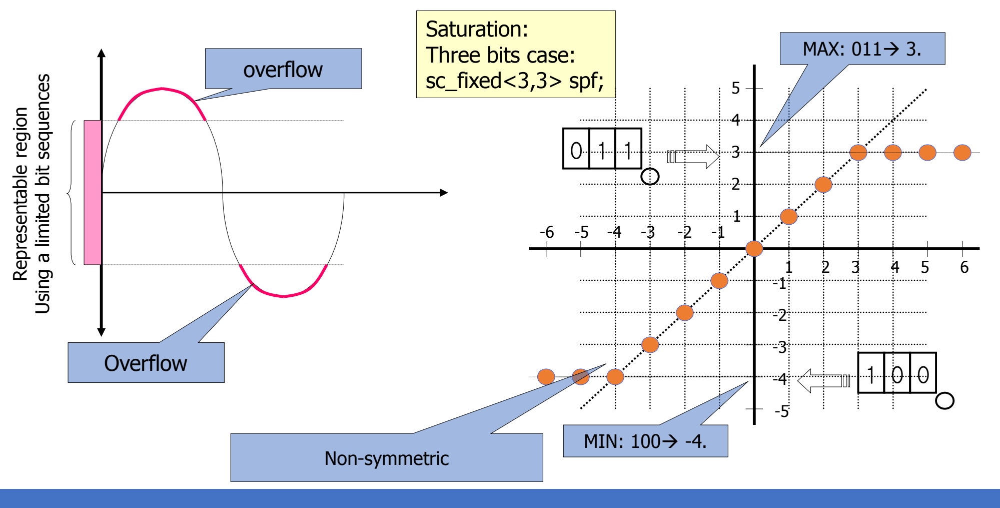
# Quantization: truncation

- Always rounded towards minus infinity
- The redundant bits are always deleted.

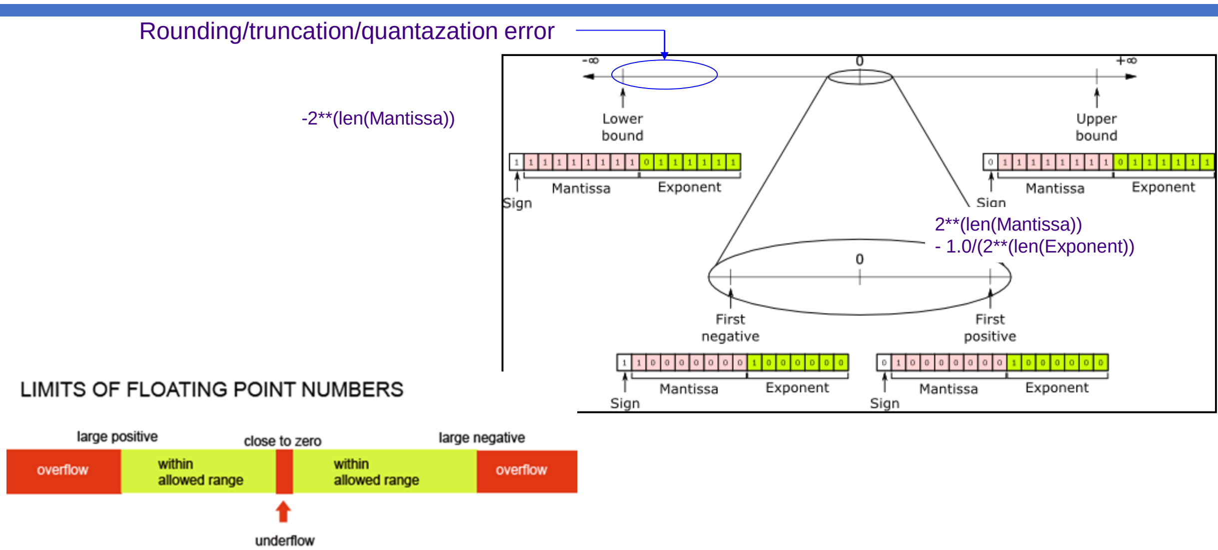



# Overflow

- Overflow/underflow if the result is too positive or too negative to fit in the result.

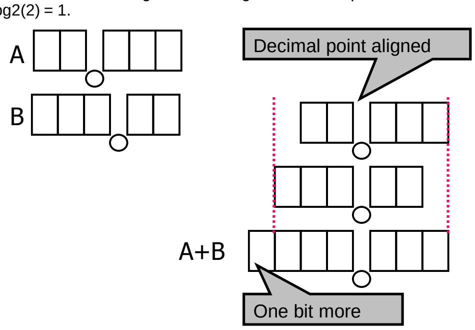


# Overflow/underflow

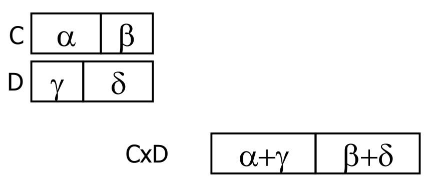



# Fixed-point math techniques

- Addition or subtraction: Radix point must be aligned.
- Addition or subtraction of N numbers requires a word length is log2(N) bit more than the maximum aligned word length.
  - Addition or subtraction requires a word length that is one bit more than the maximum aligned word length of the two operands. I.e., log2(2) = 1.

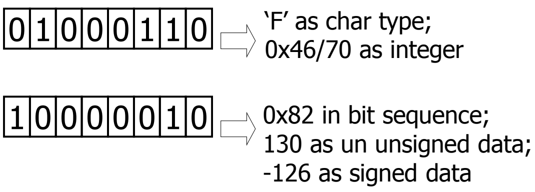



- Multiplication requires a word length that is the sum of the word length of the two operands.



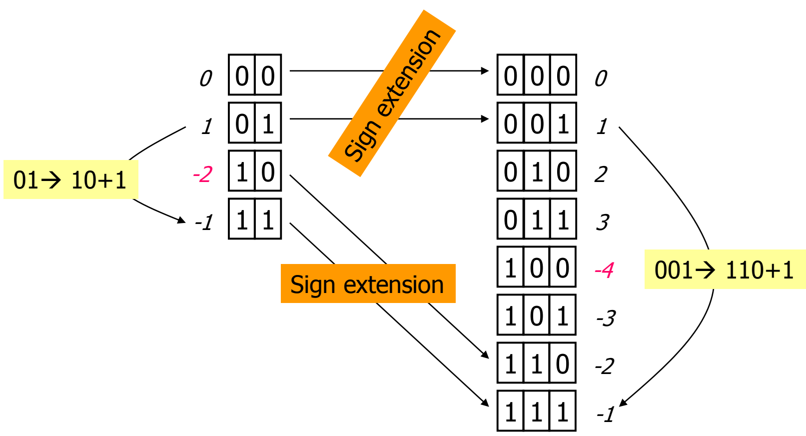
# What is data type

- Data type is the way how we interpret the meaning of bit sequences.




# Signed and unsigned

- Signed data types
  - 2's complement signed
  - In 2's complement notation, one more negative value than positive value can be represented.



# References