Floating Point

- Representing non-integer numbers
  - pi = 3.141592…
  - e = 2.71828…
  - $0.000000001 = 1.0 \times 10^{-9}$
  - $3155760000 = 3.15576 \times 10^{9}$

  } scientific notations

- Scientific notation
  - Single digit to the left of the decimal point
  - Normalized number: 1.0×10^{-9}
    - no leading zero
    - 0.1×10^{-8}, 10.0×10^{-10} are not normalized
  - Binary number in scientific notation: $1.0_2 \times 2^{-1}$
- Floating point numbers
  - Numbers in which binary point is not fixed
    - No fixed number of digits before and after the point

Floating Point Representation

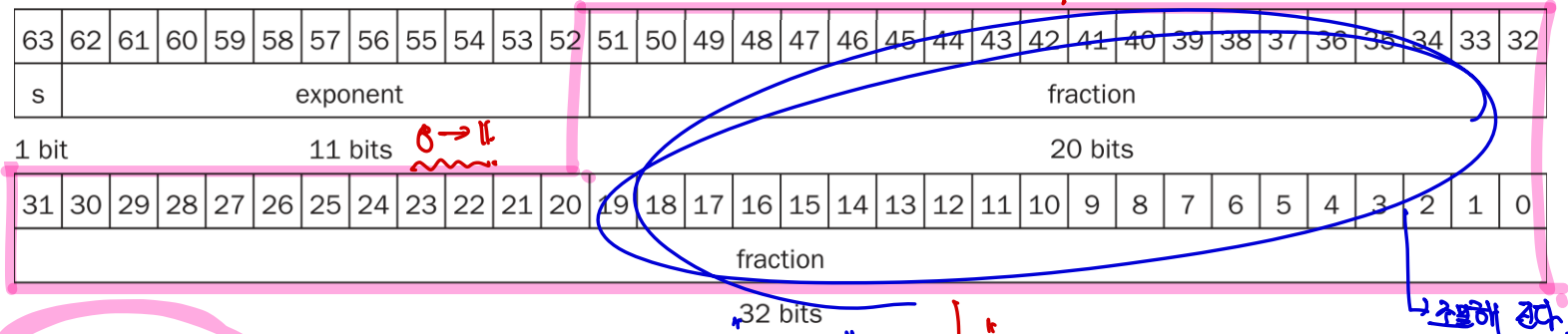
- In binary
  - $\pm 1.xxxxxxx_2 \times 2^{yyyy}$
  - Fraction and exponent
- Fraction and exponent must fit into a word
  - How many bits for fraction and exponent?
    - trade-off
- FP number representation

31	30 - 23	23 - 0
s	exponent	fraction (mantissa)
	8 bits	23 bits

  - Range in decimal: $2.0 \times 10^{-38} \sim 2.0 \times 10^{38}$
- Overflow in FP arithmetic
  - Exponent is too large for the exponent field
- Underflow in FP arithmetic
  - Negative exponent is too large to fit into the exponent field

Floating Point Representation

- Single precision: 32 bits
- Double precision: 64 bits
  - 11 bits for exponent, 52 bits for fraction
  - Range (in decimal): $2.0 \times 10^{-308} \sim 2.0 \times 10^{308}$
  - benefit: Increased precision from larger fraction bits



- IEEE 754 floating point standard
  - Exponent 00000000, Fraction 0 → 0
  - E: 11111111, F: 0 → infinity
  - E: 1-254, F: anything → normal FP number
  - Consideration for sorting
    - MSB is used as a sign bit
    - Exponent comes before fraction part

Floating Point Representation

- Biased notation
  - $X = 1.0 \times 2^{-1}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0

  - $Y = 1.0 \times 2^{+1}$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

  - In sorting, X looks larger than Y.
- Rearranging the exponent value range
  - 00000000 ~ 11111111

    most negative ~ most positive
- IEEE uses a bias of 127 for single precision (1023 for double P)
  - -1: -1 + 127 = 126 = $0111\ 1110_2$
  - 1: 1 + 127 = 128 = $1000\ 0000_2$

Precision Range

$\pm 1.00000000000000000000000 \times 2^{-126}$

$\pm 1.11111111111111111111111 \times 2^{+127}$

Floating Point Bias

Exponent		Adjusted Exponent	
127	01111111	254	11111110
.	.	.	
.	.	.	
.	.	.	
1	00000001	128	10000000
0	00000000	127	01111111
-1	11111111	126	01111110
.	.	.	
.	.	.	
.	.	.	
-126	10000010	1	00000001
-127	10000001	0	00000000
-128	10000000	-1	11111111

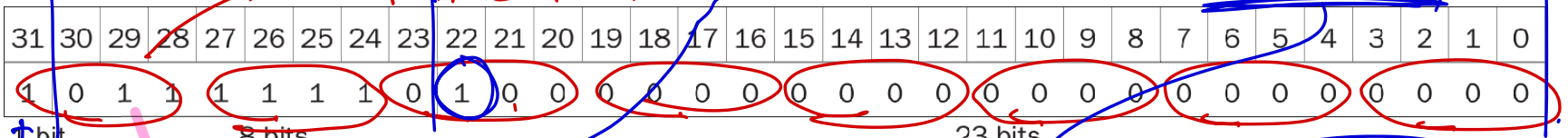
IEEE 754 Floating-Point Format

- Encoding of the single precision floating-point numbers
  - Exponent : 0, Fraction 0 → 0
  - Exponent : 255, Fraction : 0 → infinity
  - Exponent : 1-254, Fraction : anything → normal FP number
  - Exponent : 255, Fraction : Nonzero → NaN
    - NaN is a symbol for the result of invalid operations (e.g. 0/0).

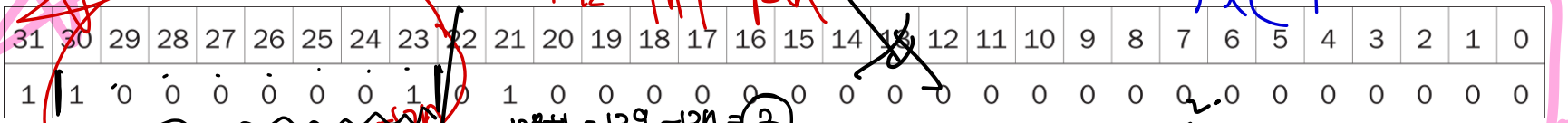
Single precision		Double precision		Object represented
Exponent	Fraction	Exponent	Fraction	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
1–254	Anything	1–2046	Anything	± floating-point number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (Not a Number)

Floating Point Example

- Represent -0.75
  - = $-3/4_{10}$ or $-3/2^2{}_{10}$
  - = $-11_2/2^2{}_{10} = -0.11_2$
  - = $-0.11_2 \times 2^0$
  - = $-1.1_2 \times 2^{-1}$
- $(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent}-127)}$
- $(-1)^1 \times (1 + 0.100\ldots000) \times 2^{126}$



- What number does this represent?



$$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent}-\text{Bias})} = (-1)^1 \times (1 + 0.25) \times 2^{(129-127)}$$
$$= -1 \times 1.25 \times 2^2$$
$$= -1.25 \times 4$$
$$= -5.0$$

Exponent - 12개.

$(-1)^S \times (1 + \text{fraction}) \times 2^{\cdot}$

-0.75

$-3/4$

$-11_{\cdot}$

$= -0.11$ ← 2진수

$= -1.1 \times 2^{-1}$

1 + 12개 = 126

1000 0001

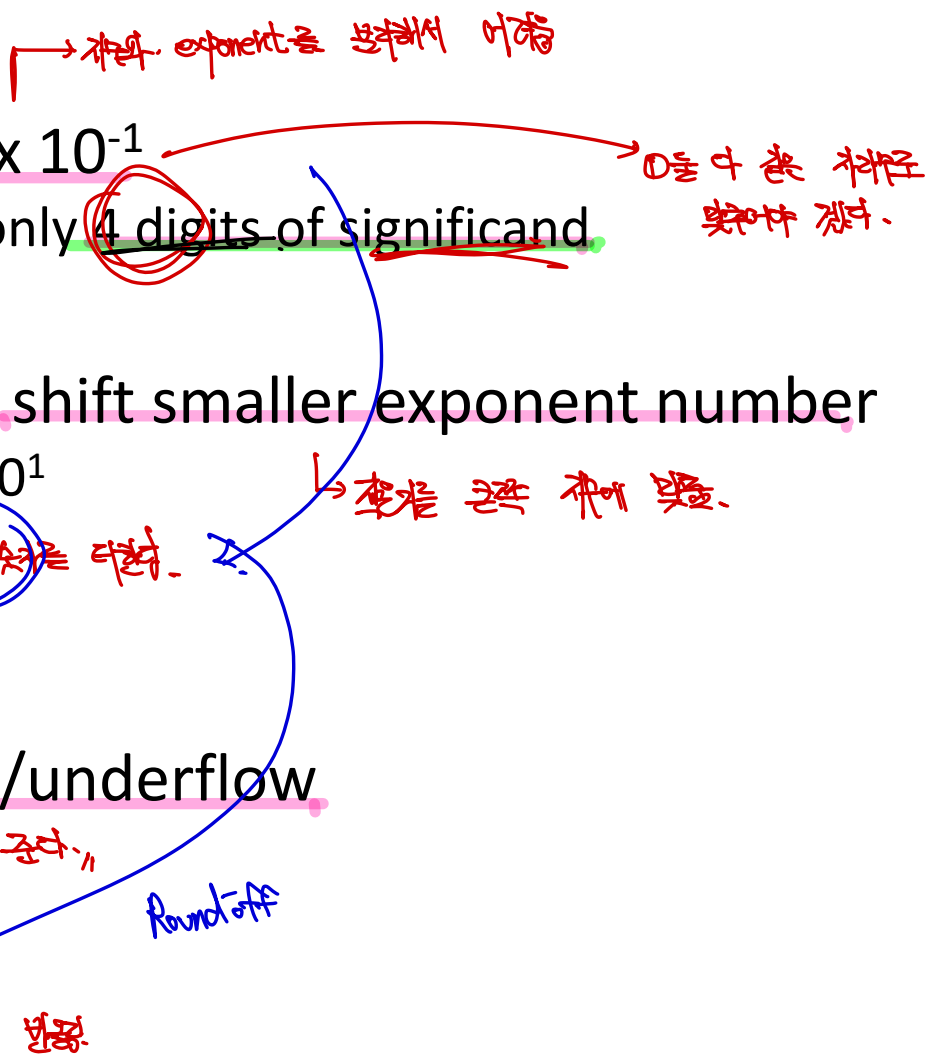
0111 1111

1000 0001

1000 0001

Floating Point Addition

- Consider:
  - $9.999_{10} \times 10^{1} + 1.610_{10} \times 10^{-1}$
    - Assume we can store only 4 digits of significand
- Steps
  - ① Align decimal points – shift smaller exponent number
    - $9.999 \times 10^{1} + 0.016 \times 10^{1}$
  - ② Add significands
    - $9.999 + 0.016 = 10.015$
    - Result: 10.015×10^{1}
  - ③ Normalize, check over/underflow
    - 1.0015×10^{2}
  - ④ Round it to 4 digits
    - 1.002×10^{2}



Binary FP Addition

- $0.5_{10} + (-0.4375_{10})$
  - $0.5_{10} = 0.1_2 = 0.1x2^0 = 1.000x2^{-1}$
  - $0.4375_{10} = 7/16 = 7/2^4 = 111x2^{-4} = 1.110x2^{-2}$
- Align
  - $1.000x2^{-1} - 1.110x2^{-2} = 1.000x2^{-1} - 0.110x2^{-1}$
- Add significands
  - $1.000x2^{-1} - 0.111x2^{-1} = 0.001 \times 2^{-1}$
- Normalize, check over/underflow
  - 1.0×2^{-4}
- Round
  - No need

FP Adder Hardware