1.14.2019 (L2)

Floating point Arithmetic

Consider -321.416

$$\downarrow$$

$$-\left(3\cdot 10^2 + 2\cdot 10 + 1\cdot 10^0 + 4\cdot 10^{-1} + 1\cdot 10^{-2} + 6\cdot 10^{-3}\right)$$

Decimal representation

$$= -3.21416 \times 10^2$$

$$= -0.321416 \times 10^3$$

A similar representation is used in a computer, "floating-point arithmetic"

$$\underset{\text{sign}}{-}.\underbrace{321416}_{\text{fraction}} \times \underset{\text{base}}{10}^{\overset{\text{exponent}}{3}}$$

In general

$$\underset{\text{sign}}{\pm} f \times \beta^{e}$$

$\beta =$ 2 binary number

10 decimal "

16 hexadecimal "

f : fraction,

digits from $0, 1, \cdots \beta - 1$

e : exponent

digits from $0 \cdots \beta - 1$

Binary numbers:

$$\underbrace{b_m \cdots b_2 b_1 b_0}_{\text{integer}} . \underbrace{a_1 a_2 \cdots a_n}$$

each digit b_i, a_j takes 0 or 1

This number in base 10 is

$$b_m \cdot 2^m + b_{m-1} 2^{m-1} + \cdots + b_1 \cdot 2^1$$
$$+ b_0 2^0 + a_1 2^{-1} + a_2 2^{-2}$$
$$+ \cdots a_n 2^{-n}$$

β

Note: $(0.1101)_2$

$$= (1.101)_2 \times 2^{-1}$$

$$= (0.001101)_2 \times 2^{2}$$

β

④ To convert between binary ($\beta=2$) and decimal ($\beta=10$)

Example:

1) $x = (1.1011)_2$, convert x to a decimal number

$$x = 1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 1\cdot 2^{-3} + 1\cdot 2^{-4}$$

$$= 1 + \frac{1}{2} + 0 + \frac{1}{8} + \frac{1}{16}$$

$$= \frac{27}{16}$$

2) $x = (1.101010\cdots10\cdots)_2$

$$= (1.\overline{10})_2$$

$$x = 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-3} + \cancel{1\cdot 2^{-5}} + 1\cdot 2^{-5} + \cdots$$

Recall geometric series

$$1 + r + r^2 + r^3 + \cdots = \frac{1}{1-r}, \text{ if } |r|<1$$

$$x = 1 + \frac{1}{2} + \left(\frac{1}{2}\right)^3 + \left(\frac{1}{2}\right)^5 + \left(\frac{1}{2}\right)^7 \cdots$$

$$= 1 + \frac{1}{2}\Big(1 + \underbrace{\tfrac{1}{4}}_{r=\frac{1}{4}} + \left(\tfrac{1}{4}\right)^2 + \left(\tfrac{1}{4}\right)^3 + \cdots\Big)$$

$$= 1 + \frac{1}{2}\,\frac{1}{1-\frac{1}{4}}$$

$$= 1 + \frac{1}{2}\,\frac{1}{\frac{3}{4}} = \frac{5}{3}$$

Alternatively:

$$x = (1.\overline{10})_2$$

$$= \underbrace{1.2^0}_{=1} + \underbrace{(0.\overline{10})_2}_{=y} \quad \cdots (*)$$

$$= 1 + (10.\overline{10})_2 \cdot 2^{-2}$$

$$y = (0.\overline{10})_2 = \underbrace{(10.\overline{10})_2}\cdot 2^{-2}$$

$$= \Big\{\underbrace{(10)_2}_{=2} + \underbrace{(0.\overline{10})_2}_{y}\Big\}\, 2^{-2}$$

$$\Rightarrow \quad y = (2+y)\,\underbrace{2^{-2}}_{=\frac{1}{4}}$$

$$\Rightarrow \quad 4y = 2 + y \Rightarrow y = \frac{2}{3} \quad (**)$$

$$\Rightarrow x = \frac{5}{3}$$

Example: Convert 14.8125 to ⑥ a binary number.

we are looking for

$$14.\boxed{8125} = (b_m b_{m-1} \cdots b_1 b_0 . \boxed{a_1 a_2 \cdots a_n})_2$$

fractional part

$$0.8125 = (.a_1 a_2 \cdots a_n)_2$$

$$= a_1 \cdot 2^{-1} + a_2 2^{-2} + \cdots a_n 2^{-n}$$

$- * \cdot 2$

$$\boxed{1}6250 = \cancel{2\cdot} \boxed{a_1} + a_2 2^{-1} + \cdots + a_n 2^{-(n-1)}$$

$\boxed{a_1 = 1}$

$$\Rightarrow 0.6250 = a_2 2^{-1} + \cdots a_n 2^{-(n-1)}$$

$- * 2$

$$\boxed{1}.2500 = \boxed{a_2} + a_3 2^{-1} + \cdots a_n 2^{-(n-2)}$$

$\boxed{a_2 = 1}$

$0.25 \times 2 = \boxed{0}.50$ $\quad 0 = a_3$

$0.50 \times 2 = \boxed{1}.0$ $\quad \boxed{1 = a_4}$

$\Rightarrow .8125 = (.a_1 a_2 a_3 a_4)_2$

$= (.1101)_2$

< Collect integer part, ordered from radix point. >

Integer part

$14 = (b_m \cdots b_2 b_1 b_0)_2$

$= b_m \cdot 2^m + b_m 2^{m-1}$

$+ \cdots + b_1 2^1 + b_0$

. divided by 2

$14/2 = 7 \, R \, \boxed{0} = (b_m 2^{m-1} + \cdots + b_1) \, R \, \boxed{b_0}$

$\boxed{b_0 = 0}$

$7/2 = 3 \text{ R } \boxed{1}$ $\qquad b_1 = 1$

$3/2 = 1 \text{ R } \boxed{1}$ $\qquad b_2 = 1$

$1/2 = 0 \text{ R } \boxed{1}$ $\qquad b_3 = 1$

$\Rightarrow 14 = (1110)_2$

$\Rightarrow 14.8125 = (1110.1101)_2$

$$\underbrace{\pm f \cdot \beta^{e}}_{\text{floating point number}} \quad (\beta = 2)$$

f: (fraction): the number of digits in f determines the precision

e: (exponent): the number of digits in e determine the range of representable numbers

we follow
[IEEE 754 floating point standard]

normalized form

$f = 1.b_m b_{m-1} \cdots b_1 b_0$

leading digit is 1.

$(0.0101010111 0 \cdots 0)$

Advantage: leading 1 needs not be stored.

32-bit single precision

s	e	f
1-b	8-bit	23-bit

64-bit double precision

s	e	f
1-b	11-b	52-b

The represented number is

$$(-1)^s \cdot (1+f) \cdot 2^{e-e_0}$$

e: un-signed, e_0: exponent bias

$e-e_0$: can be either positive and negative.

↓

to represent small num.

- Let's focus on "e", or equivalently

$$2^{e-e_0}$$

single-precision

$e \in [e_{min}, e_{max}]$

e with all 0s or all 1s needs to be interpreted in a special way

$e_{min} = (\underbrace{0 \cdots 01}_{8\text{-bit}})_2 = 1$

$e_{max} = (\underbrace{11 \cdots 10}_{8\text{-bit}})_2 = 1 \cdot (2 + 2^2 + 2^3 + \cdots 2^7)$

$= 2\left(\frac{1-2^7}{1-2}\right) = 254$

$a + a^2 + a^3 + \cdots + a^n = \frac{1-a^{n+1}}{1-a}$ $a \neq 1$

$e_0 = 127$

$\Rightarrow 2^{e-e_0} \in [2^{-126}, 2^{127}]$

$\approx [10^{-38}, 10^{38}]$

Double-precision

$e \in [e_{min}, e_{max}]$

$e_{min} = 1$ $\quad e_{max} = 2^1 + 2^2 + \cdot 2^{10} = 2046$

$e_0 = 1023$

$2^{e-e_0} \in [2^{-1022}, 2^{1023}] \approx [10^{-308}, 10^{308}]$

Next : fraction f and precision

using double-precision as an example:

- How to store a number
- How to do calculation

Consider

$$x_1 = \frac{27}{16} = (1.1011)_2$$

$$x_2 = \frac{5}{3} = (1.\overline{10})_2$$

$$x_3 = \frac{2}{3} = (.\overline{10})_2 = (1.\overline{01})_2 \times 2^{-1}$$

$$x_4 = 1 = (1.0)_2$$

$$x_5 = 1 \times 2^{-52}$$

$$x_6 = 1 \times 2^{-53}$$

52-bit

Exactly

$1.\boxed{1011\,0\,0\cdots\,0}$ ... x_1

$1.\boxed{0\cdots\,0}$ ... x_4

$1\,\boxed{0\cdots\,0}$ 2^{-52} x_5

$\downarrow$ e-eo

$1.\boxed{0\cdots\,0}$ 2^{-53} x_6

Now x_2. x_3

$1\,\boxed{1010\cdots\,011}$ $\leftrightarrow$ $\boxed{fl(x_2)}$

$1\,\boxed{1010\cdots\,10}\,1010\cdots10$ $\leftarrow$ x_2

$x_3 = (1.\overline{01})_2 \times 10^{-1}$ $1.\boxed{0101\cdots01}\cdot 2^{-1}$, $fl(x_3)$

$1\,\boxed{0101\cdots\,01}\,0101\cdots$ $\swarrow$ x_3

$\cdot 2^{-1}$

$= 0 - e_0$

we follow

IEEE: rounding to

nearest rule.

$1\,\boxed{\cdots}\,1\underbrace{00000}_{0}$

following the rule.

the computer represented [illegible]

number for x: $fl(x)$

IEEE Rounding to Nearest Rule

<Double precision>

Check the 53rd bit to the right of binary point.

If it is 0, then truncate after 52nd bit (round down)

If it is 1, then

- add 1 to 52nd bit (round up)

UNLESS all digits to the right of the 1 are 0's,

in which case 1 is added to bit 52 if and only if bit 52 is 1