Sep- 6

Floating point represontation.

What is a FP representation?

3.18924 {FP number}

-10.192

2×10^{-15}

-4.23×10^{31}

Fixed Point numbers : Currency

↓

7.32

2500.99

How to represent floating point numbers in binary?

$$(11) = 8 + 0 + 2 + 1$$

$$= 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$$

$$= (1011)_b$$

$$1.75 = 1 + 0.5 + 0.25$$

$$= 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$$

$$= (1.11)_b$$

$$1.5625 = 1 + 0.5 + 0\times 0.25 + 0\times .125 + 1\times 0.0625$$

$$= (1.1001)_b$$

decimal $\rightarrow$ binary

binary $\rightarrow$ decimal

$(1.0011)_b \rightarrow 1\times2^0 + 0\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 1 + 1/8 + 1/16$

$= \left[1\frac{3}{16}\right]_{decimal}$

floating point $\rightarrow \begin{cases} +/- \\ \cdot \text{ point} \\ \text{exponent } (1.37\times10^{-19}) \end{cases}$

7

IEEE 754 Floating Point Format

Standard form representation.

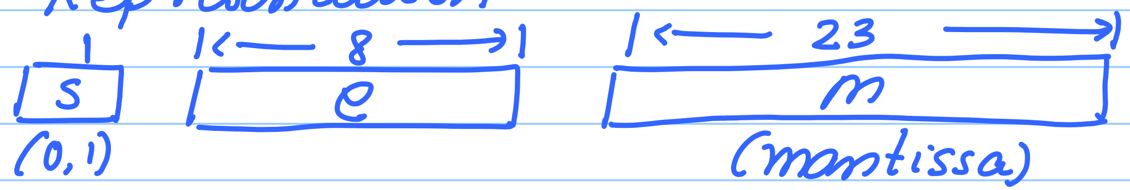
$$N = (-1)^s (1.m) \times 2^{exp}$$

$$s \in \{0, 1\}$$

$$2.5 = (-1)^0 (1.25) \times 2^1$$

$$4.5 = (-1)^0 (1.125) \times 2^2$$

IEEE 754 Representation



$\boxed{1}\ \boxed{m}$ → Mantissa (23)

Implicit

have both (+)ve and (-)ve exponents

Instead of representing (e) in 2s complement form

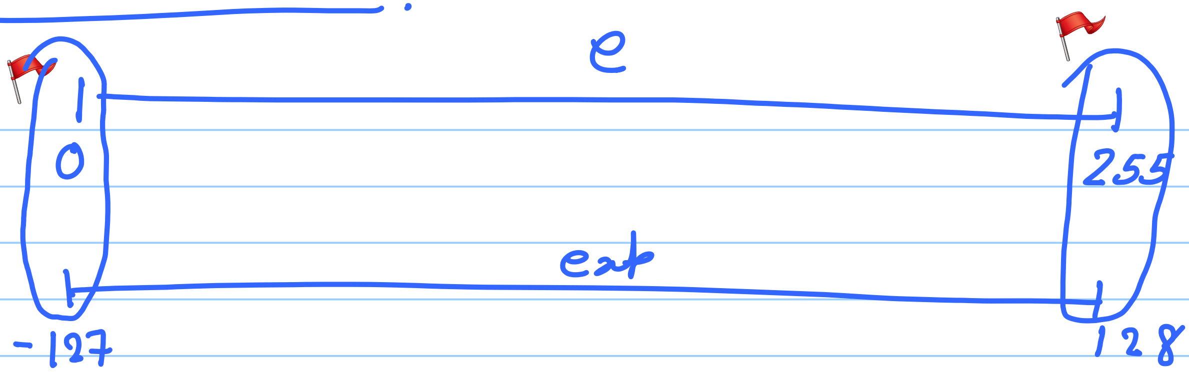
→ biased notation

$$exp = e - BIAS$$

exp: 128 to −127

e: 255 to 0 (127)

Special Cases.



The range of (e, exp) for normal FP numbers

$e \rightarrow 1 \ldots 254$

$exp \rightarrow -126 \ldots 127$

Limits of (+)ve FP (normal) numbers:

Smallest (+)ve normal FP number:

$$1.\underbrace{0 \cdots 0}_{23} \times 2^{-126} = 2^{-126}$$

Largest (+)ve normal FP number:

$$1.\underbrace{1 \cdots 1}_{23} \times 2^{127}$$

$$= 2^{127} \times (2^0 + 2^{-1} \cdots 2^{-23})$$

$$= 2^{127} \times (2^1 - 2^{-23})$$

$$= \boxed{2^{128} - 2^{104}}$$

Special Cases

$e = 0, \quad e = 255$

$e = 255, \quad m = 0, \quad S = 0 \qquad \underline{\underline{\infty}}$

$\qquad\qquad\qquad\qquad\quad S = 1 \qquad -\infty$

$e = 255, \quad m \neq 0$

NAN
(Not a Number)
$\left\{ \begin{array}{l} \sin^{-1}(5) \\ \log(-4) \end{array} \right\}$

$2 + NAN = NAN$

$NAN \times x = NAN$

$e = 0, \quad m = 0 \qquad \underline{\underline{0}}$

$e = 0, \quad m \neq 0$

[denormal numbers]

$$\left\{ \begin{array}{l} x = 2^{-126}; \\ \text{if } (x/2 == 0) \\ \quad \text{printf}(\text{" hi \textbackslash n"}); \end{array} \right\}$$

Violates Axiom

$x > 0 \Rightarrow x/2 > 0$

To protect against such potential contradictions we have some buffer in the form of denormal numbers

Denormal number: $\left[N = (-1)^s \, 0.m \times 2^{-126} \right]$

Limits of (+)ve denormal numbers:

smallest : $0.\overbrace{0 \ldots 0}^{22}1 \times 2^{-126}$

$= 2^{-23} \times 2^{-126} = 2^{-149}$

largest: $0.\underbrace{11 \cdots\cdots 1}_{23} \times 2^{-126}$

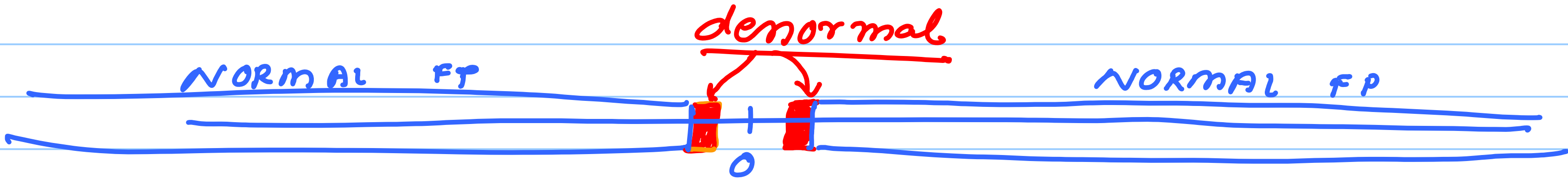
$$= 2^{-126} \times (2^{-1} + \cdots\cdots 2^{-23})$$

$$= 2^{-126} \times (2^{0} - 2^{-23})$$

$$= 2^{-126} \times (1 - 2^{-23})$$

$$= \left[2^{-126} - 2^{-149}\right]$$

NUMBER LINE



Now, this piece of code will work as expected

$$\begin{cases} x = 2^{-126}; \\ \text{if } (x/2 == 0) \\ \quad \text{printf("hi");} \end{cases}$$

REASON: $x/2 = 2^{-127}$ can be represented as a valid denormal number

$$\begin{cases} x = 2^{-149} \\ \text{if } (x/2 == 0) \\ \quad \text{printf("hi");} \end{cases}$$

output : hi

I want to solve this problem.
Instead of a float, I will use a double.

float (32) → floating point, single precision

double (64) → double precision

double

$$\text{exp} = e - \text{BIAS}$$

$$= e - 1023$$

Largest double precision number:

$$\left\{2^{emax/2} - 2^{emax/2 - 1 - (m)}\right\}$$

$$\begin{array}{r} 1023 \\ -\ 52 \\ \hline 971 \end{array}$$

$$2^{1024} - 2^{1024-1-52}$$

$$= \left\{2^{1024} - 2^{971}\right\} \approx 10^{300}$$

Range of double (64) numbers $\approx (-10^{300}$ to $10^{300})$

of float (32) numbers $\approx (-10^{40}$ to $10^{40})$

Intel machines have a format called extended precision (80 bits)

You can still have mathematical contradictions.

$$\Delta x > 0$$

$$x + \Delta x > x$$

$$\begin{cases} x = 2^{100} \\ \Delta x = 2^{-100} \\ y = 2^{100} \\ z = x + \Delta x - y \end{cases}$$

```
{ if (z > 0)
        printf ("good");
}
```

Expected Output: good

$$\left\{\begin{aligned} Z &= (x+\Delta x) - y \\ &= x - y = 0 \end{aligned}\right]$$

Comp 1

$$\left[\begin{aligned} Z &= (x-y) + \Delta x \\ &= 0 + \Delta x \\ &= \Delta x > 0 \end{aligned}\right]$$

comp 2

The result clearly depends on the order of computations.

Conclude:

1) FP arithmetic is approximate
2) Can lead to the violation of basic mathematical axioms.