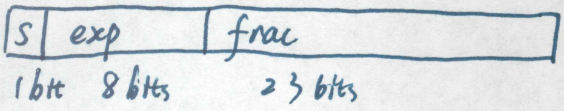
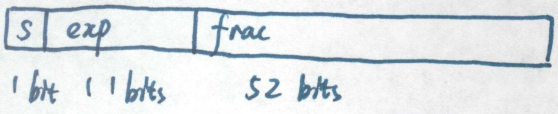
deprecated note on floats

IEEE floating point, based on IEEE Standard 754

Single precision: 32 bits

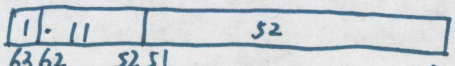


Double precision: 64 bits



S: 符号位 sign　exp: 指数 exponent　frac: 小数 fraction

以Double precision为例



$value = (-1)^{sign} \times fraction \times 2^{exponent}$

$exponent = C - 1023$ ← 偏移量 exponent bias　单精度浮点数偏移量为127　$bias = 2^{k-1} - 1$, k = # of exp bits

$fraction = 1 + m$ ← 尾数 mantissa

$C = \sum_{i=0}^{10} c_i 2^i$

62位到52位存的是C，共11位，以表示exponent，range [0, 204]

所以 exponent = C − 1023 (encodes exponent) 的range为 [−1023, 1024]，不考虑C=0和C=204的情况(特殊含义)

exponent = C − 1023 的range为 [−1022, 1023]，指数exponent的range为 $[2^{-1022}, 2^{1023}]$

$m = \sum_{i=1}^{52} m_i 2^{-i}$

51位到0位存的是m，共52位，以表示fraction，range $[0, 1-2^{-52}]$

所以小数fraction (encodes) 的range为 $[1, 2-2^{-52}]$

当 $C = 2046$, $m = 1 - 2^{-52}$ 时，双精度浮点数达到可以表示的最大正数 $\approx 2 \times 10^{308}$

当 $C = 1$, $m = 0$ 时，最小正数 $\approx 2 \times 10^{-308}$

最小负数 $\approx -2 \times 10^{308}$

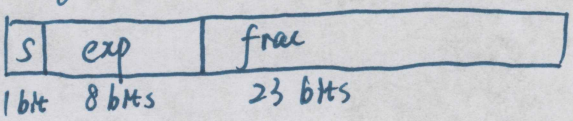
最大负数 $\approx -2 \times 10^{-308}$

$C = 0$ { $m = 0 \Rightarrow \pm 0$; $m \neq 0$ denormalized num

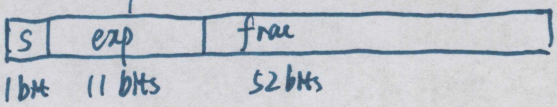
$C = 204$ { $m = 0 \Rightarrow \pm\infty$; $m \neq 0 \Rightarrow$ NaN

IEEE floating point, based on IEEE Standard 754

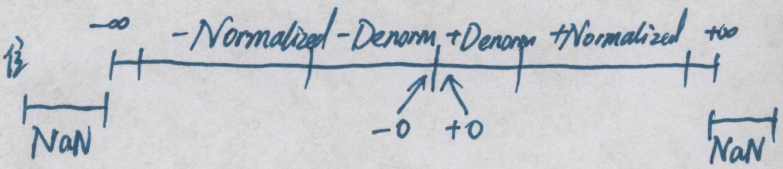
Single precision: 32 bits 有效位数: 7位



Double precision: 64 bits 有效位数: 16位



Visualization: Float Point Encodings



s: sign bit 符号位 exp: exponent 指数 frac: fraction 小数

$$value = (-1)^{sign} \times fraction \times 2^{E}$$

$$E = \begin{cases} exp - bias & normalized \\ 1 - bias & denormalized \end{cases}$$

$bias = 2^{k-1} - 1$, k: # of exp bits 偏移量

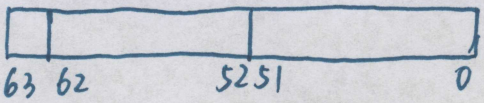
$$fraction = \begin{cases} 1+m & normalized \\ 0+m & denormalized \end{cases}$$

mantissa 尾数

always have implied 1

no implied 1

example: double precision normalized values



63: sign bit

62-52: stores exp to encode E. 存储 exp 以表示 E; $exp = \sum_{i=0}^{10} exp_i 2^i \in [0, 2047]$

去掉 exp=0 和 exp=2047 的情况 (denormalized values), $E = exp - bias \in [-1022, 1023]$

$2^E \in [2^{-1022}, 2^{1023}]$

51-0: stores frac to encode fraction. 存储 frac 以表示 fraction; $frac = \sum_{i=1}^{52} frac_i 2^{-i} \in [0, 1-2^{-52}]$

when exp = 2046 ($E = 1023$), frac $1-2^{-52}$ 时. 可表示最大正数 $\approx 1.8 \times 10^{308}$

when exp = 1 ($E = -1022$), frac = 0 时 可表示最小正数 $\approx 2.2 \times 10^{-308}$

2's complement : overflow in modulo way

denormalized floating point : overflow to infinity

denormalized : equal-spaced

normalized : once exp increment by 1, the spacing doubles
numbers with same exp field are equally spaced

Rounding : Nearest Even is the default method

less than a half : round down
more than a half : round up
in the middle : round to nearest even { if truncate is odd : round up, add 1 and propagate
if truncate is even : round down, that is, just truncate } statistically unbiased

Rounding Binary Fractional Numbers :

if bits to right of rounding position start with 0 : round down
start with 1 { 100...0 舍为0 half way
1...0...1... 舍1 round up

floating point puzzles

int x = ... ; x == (int)(float)x : false, we don't have enough frac bits in float to represent int, so we lose some bits by rounding

x == (int)(double)x : true, we have enough frac bits in double to represent int

float f = ... ; f == (float)(double)f : true
neither d nor f is NaN

double d = ... ; d == (double)(float)d : false

f == -(-f) : true just toggling a bit

2/3 == 2/3.0 : false

if d < 0.0 d*2 < 0.0 : true (even 2*d causes overflow, because overflow goes to negative infinity)

if d > f -f > -d : true monocity

d*d ≥ 0.0 : ~~false~~ true (overflow goes to positive infinity)

(d+f)-d == f : false not associative