Signed zero I

- Why do we have $+0$ and -0 ?
  First, it is available (1 bit for sign)
  if no sign, $1/(1/x)) = x$ fails when $x = \pm\infty$

$$x = \infty, 1/x = 0, 1/0 = +\infty$$
$$x = -\infty, 1/x = 0, 1/0 = +\infty$$

- Compare $+0$ and -0:

```
if (x == 0)
```

  IEEE defines $+0 = -0$

Signed zero II

- IEEE:

$$3 \times (+0) = +0, +0/(-3) = -0$$

- ± 0 useful in the following situations:

$$\log x \equiv \begin{cases} -\infty & x = 0 \\ \text{NaN} & x < 0 \end{cases}$$

A small underflow negative number $\Rightarrow \log x$ should be NaN

Signed zero III

- Definition: underflow means a value smaller than the smallest floating-point number occurs
- x underflow $\Rightarrow$ round to 0, if no sign, $\log x$ is $-\infty$ but not NaN
- With ± 0, we have

$$\log x = \begin{cases} -\infty & x = +0 \\ \text{NaN} & x = -0 \\ \text{NaN} & x < 0 \end{cases}$$

Positive underflow $\Rightarrow$ round to $+0$

Signed zero IV

- Useful in complex arithmetic
  $\sqrt{1/z}$ and $1/\sqrt{z}$
  $z = -1, \sqrt{1/-1} = \sqrt{-1} = i, 1/\sqrt{-1} = 1/i = -i$
  $\Rightarrow \sqrt{1/z} \neq 1/\sqrt{z}$
- This happens because square root is multi-valued.

$$i^2 = (-i)^2 = -1$$

Signed zero V

- However, by some restrictions (or ways of calculation), they can be equal

$$z = -1 = -1 + 0i,$$
$$1/z = 1/(-1 + 0i) = -1 + (-0)i$$

so

$$\sqrt{1/z} = \sqrt{-1 + (-0)i} = -i$$

$\Rightarrow -0$ is useful

- Disadvantage of $+0$ and -0:

$x = y \Leftrightarrow 1/x = 1/y$ is destroyed

Signed zero VI

$x = 0, y = -0 \Rightarrow x = y$ under IEEE

$1/x = +\infty, 1/y = -\infty, +\infty \neq -\infty$

- There are always pros and cons for floating-point design

Denormalized number I

- Consider

$$\beta = 10, p = 3, e_{\min} = -98, x = 6.87 \times 10^{-97},$$
$$y = 6.81 \times 10^{-97}$$

- x, y are ok but $x - y = 0.6 \times 10^{-98}$ rounded to 0, even though $x \neq y$
- How important to preserve

$$x = y \Leftrightarrow x - y = 0$$

- `if (x ≠ y) {z = 1/(x-y);}`

Denormalized number II

The statement is true, but z becomes ∞

Tracking such bugs is frustrating

- IEEE uses denormalized numbers to guarantee

$$x = y \Leftrightarrow x - y = 0$$

  Details of how this is done are not discussed here

- Most controversial part in IEEE standard

  It caused long delay of the standard

- If denormalized number is used, 0.6×10^{-98} is also a floating-point number

Denormalized number III

- Remember we do not store 1 of $1.d \cdots d$
- How to represent denormalized numbers ?

  Recall for valid value, $e \geq e_{\text{min}}$ and we have $1.d \cdots d \times 2^{e}$
- For denormalized numbers, we let $e = e_{\text{min}} - 1$ and the corresponding value be

$$0.d \cdots d \times 2^{e+1} = 0.d \cdots d \times 2^{e_{\text{min}}}$$

Denormalized number IV

- Why not

$$1.d \cdots d \times 2^{e_{\min}-1}$$

Then we cannot represent

$$0.0x \ldots x \times 2^{e_{\min}}$$

Example: $6.87 \times 10^{-97} - 6.81 \times 10^{-97} \Rightarrow$ underflow due to cancellation

Denormalized number V

- An example of using denormalized numbers

$$\begin{aligned} \frac{a+bi}{c+di} &= \frac{(a+bi)(c-di)}{(c+di)(c-di)} \\ &= \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2} i \end{aligned}$$

If c or $d > \sqrt{\beta}\beta^{e_{\max}/2} \Rightarrow$ overflow

- Definition: overflow means a number larger than the maximal floating-point number occurs

Denormalized number VI

- Smith's formula

$$\frac{a+bi}{c+di} = \begin{cases} \frac{a+b(d/c)}{c+d(d/c)} + \frac{b-a(d/c)}{c+d(d/c)} i & \text{if } (|d| < |c|) \\ \frac{b+a(c/d)}{d+c(c/d)} + \frac{-a+b(c/d)}{d+c(c/d)} i & \text{if } (|d| \geq |c|) \end{cases}$$

  This avoids overflow

- However, using Smith's formula, some issues may occur without denormalized numbers

Denormalized number VII

If

$$a = 2 \times 10^{-98}, b = 1 \times 10^{-98}, c = 4 \times 10^{-98},$$
$$d = 2 \times 10^{-98}$$

then

$$d/c = 0.5, c + d(d/c) = 5 \times 10^{-98},$$
$$b(d/c) = 1 \times 10^{-98} \times 0.5 = 0$$
$$a + b(d/c) = 2 \times 10^{-98}$$

Solution $= 0.4$, wrong

Denormalized number VIII

If denormalized numbers are used, 0.5×10^{-98} can be stored,

$$a + b(d/c) = 2.5 \times 10^{-98} \Rightarrow 0.5$$

the correct answer

- Usually hardware does not support denormalized numbers directly

  Using software to simulate
- Programs may be slow if a lot of underflow