ECE 4960

Spring 2017

Lecture 4

Exception Handling: Soft Landing

Edwin C. Kan
School of Electrical and Computer Engineering
Cornell University

Signed Zero

- Zero is represented by the zero exponent *e* and the zero mantissa *f*. The sign field in IEEE standards actually makes a difference.
- With the "bias" in the floating point representation, the *e* field for zero in the normal expression of $(-1)^s \cdot (1.f) \cdot 2^{e-1023}$ corresponds to $e_{min} - 1$ or $(11111111110)_2$ or $-(1022)_{10}$.
- Remember that $(e)_2$ for $(11111111111)_2$ is reserved for exception of NaN and INF.
- +0 == –0, rather than –0 < +0
- As we distinguish INF and NINF, we need to distinguish –0 and +0 to make $1/(1/x) = x$, when x is INF or NINF.
- log(+0) is NINF and log(–0) is NaN.

Hacker Practice

- ❑ Write a small function to test +0 and –0.

In an upper-level function, use

+1.0; –1.0; DBL_MAX; – 1.0*DBL_MAX; +0; –0; INF, NINF; NaN

to test and generate report!

At home:

- ❑ Write a small function to test INF and NINF.
- ❑ Write a small function to test NaN.

If you do not know how to link with math.h or python built-in, use DBL_MAX = 10^{308}

Needs to Handle Underflowing

- With the "normal" or "normalized" expression when the e field represents a negative number and the mantissa = $(1.f)_2 > 1$, the smallest number representable in double precision is 2^{-1022}.
- For $x = (1.1011)_2 \times 2^{-1020}$ and $y = (1.1010)_2 \times 2^{-1020}$ both are representable, legal floating-point numbers.
- They have a strange arithmetic property without exception handling: $x - y = 0$ even though $x \neq y$!!!
- A programmer can easily write:

```
if (x != y) { z = 1.0/(x - y)};
```

- This can have surprises when underflow happens!

Denormals

- Define in double precision $e_{\min} = -1022$
- When $e > e_{\min} - 1$, the number is $1.b_1b_2...b_{p-1} \times 2^e$
- When $e = e_{\min} - 1$, the number is $0.b_1b_2...b_{p-1} \times 2^{e+1}$
- A convention called gradual underflow or soft landing.
- We can prove that $x = y \Leftrightarrow x - y = 0$ always holds when denormals are used.

Hacker Practice

- Observe the exception handling on your platform:

```
// Make x with easily observable precision
//
double x = 1.234567890123456;
int i = 1;

// The normalized number is above 4.9407*10^(-324)
x *= 10^(-307);

// Decrease the normalized number to the range of denormals
for (i=1; i<20; i++) {
      x /= 10.0;
      print(x);
}
```

- Suggest another way to observe the soft landing behavior