ME 360 Lecture 3.

- Recap from last lecture.

$(49.25)_{10}$ to binary.

$49/2 = 24R1$ $\quad 0.25\times2 = 0.5+0$
$24/2 = 12R0$ $\quad 0.5\times2 = 0.0+1$
$12/2 = 6R0$
$6/2 = 3R0$
$3/2 = 1R1$
$1/2 = 0R1$

$(49.25)_{10} = (110001.01)_2$

$(101.0\overline{101})_2$ to decimal

$(101)_2 = (5)_{10}$
$x = (0.0\overline{101})_2$
$2^4x = (101.\overline{101})_2$
$2x = (0.\overline{101})_2$
$(2^4-2)x = (101)_2 = (5)_{10}$
$x = 5/(2^4-2) = 5/14$

$(101.0\overline{101})_2 = (5\ 5/14)_{10}$

0.3 Floating Point Representation of Real Numbers

0.3.1 Floating Point Format

We use the IEEE 754 standard to represent floating point number in computers (released 1985).

Floating point number
- sign (+ or −)
- exponent (size)
- mantissa (significant bits)

Precision	sign	Exponent	Mantissa
single	1	8	23
double	1	11	52
long double	1	15	64

General normalized form:

$$\underset{\text{sign}}{\pm}\underbrace{1.bbb\ldots b}_{\text{mantissa}} \times 2^{p} \quad p \rightarrow \text{exponent}$$

Example: $(9)_{10} = (1001)_2 = +1.001\times2^3$

Let's look at double precision numbers:

*** Exponent p**

We have 11 bits, so the largest binary number we can store is: $(11111111111)_2 = 2^{11} - 1 = 2047$

so, the exponent range can be $[0, 2047]$.

But we also want negative exponents!
Using IEEE standard, we apply an "exponent bias"

$$[0, 2047] = p + 1023$$

We need "0" & "2047" for special cases, so

$$p = [-1022, 1023]$$

2047: Inf (∞) or -Inf ($-\infty$) if all mantissa bits are zero
NaN (not a number) otherwise

0: **Subnormal** floating point number

$$\pm 0.b_1 b_2 \ldots b_{52} \times 2^{-1022}$$

we extend the smallest precision from 2^{-1022} to 2^{-1074}

*** Mantissa**

- Double precision number 1 is.

$$+1.\underbrace{\boxed{00\cdots00}}_{\text{52 zeros. (mantissa)}} \times 2^0$$

Next floating point number greater than 1 is:

$$+1.\boxed{00\cdots01} \times 2^0, \text{ or } 1 + 2^{-52}$$

Machine precision ϵ_{mach}

distance between 1 and the smallest floating point number greater than 1.
For double precision: $\epsilon_{mach} = 2^{-52}$

Let's look at an example:

What is $(9.4)_{10}$ in double precision?

We have $(9.4)_{10} = (1001.\underbrace{\overline{0110}}_{\text{repeating}})_2$

so: $(9.4)_{10} = 1.\underbrace{\boxed{0010110 0110 \cdots 0110}}_{52 \text{ mantissa}}0110 \cdots \times 2^3$

How do we approximate this infinite number in 52 mantissa?

Method #1: Chopping

we just remove all the bits beyond 52nd.

Pro: simple. Con: bias results toward zero.

Method #2: Rounding.

In base 10, we round up if the next digit is 5 or higher and vice versa.

In binary, we round up if the bit is 1, with one exception.

if the bits following 52nd is 10000...., exactly half way between up & down, we round up or down according to which choice makes the final bit 52 equal to 0.

Example: $\boxed{1\ldots1}10000\ldots$ round up

$\boxed{1\ldots0}10000\ldots$ round down.

Why? to avoid bias in long computation.

Method 2 is the IEEE Rounding to Nearest Rule, for any double precision floating point number associated to x, we denote it as $fl(x)$.

Recipe for floating point representation:

1. Justify

   shift radix point to the right of the leftmost 1, and compensate with the exponent.

2. Round

   apply a rounding rule, such as the IEEE Rounding to Nearest Rule, to reduce mantissa to 52 bits.

Let's go back to $(9.4)_{10}$.

$$(9.4)_{10} = (1001.\overline{0110})_2 = 1.\boxed{0010110\ldots0110}0110 \times 2^3 \qquad \text{Justify}$$

$$fl(9.4) = 1.0010110\cdots01101 \times 2^3 \qquad \text{Round.}$$

What happened to this rounding?

- we removed the tail $(.\overline{1100})_2 \times 2^{-52} \times 2^3$

  $= (.\overline{0110})_2 \times 2^{-51} \times 2^3 = 0.4 \times 2^{-48}$

- we added $2^{-52} \times 2^3 = 2^{-49}$

so. $fl(9.4) = 9.4 + 2^{-49} - 0.4 \times 2^{-48} = 9.4 + \underbrace{0.2 \times 2^{-49}}_{\text{rounding error}}$

Error Definition

* Let x_c be a computed version of the exact quantity x. Then:

  Absolute Error is $|x_c - x|$.

  Relative Error is $\dfrac{|x_c - x|}{|x|}$

Relative Rounding Error

In the IEEE machine arithmetic model, the relative rounding error of $fl(x)$ is no more than 1/2 machine precision

$$\frac{|fl(x) - x|}{|x|} \le \frac{1}{2}\varepsilon_{mach}$$

For example, the rounding error of $x = 9.4$ is 0.2×2^{-49}, so the relative rounding error is:

$$\frac{|fl(9.4) - 9.4|}{9.4} = \frac{0.2 \times 2^{-49}}{9.4} = \frac{8}{47} \times 2^{-52} < \frac{1}{2}\epsilon_{mach}.$$

* Reminder, in double precision: $\epsilon_{mach} = 2^{-52}$

Exercise in class: