CS 357 - 04 Floating point representation

Boyang Li (boyangl3)

Base-β system: For a base-β system, a number can be expressed as

$$(a_n \dots a_{n-1} a_3 a_2 a_1 a_0 . b_1 b_2 b_3 b_4 \dots)_\beta = \sum_{k=0}^{n} a_k \beta^k + \sum_{k=1}^{\infty} b_k \beta^{-k}$$

Commonly used system: Binary ($\beta = 2$), Octal ($\beta = 8$), Decimal ($\beta = 10$) and Hexadecimal ($\beta = 16$).

Convert integers between different systems:

- **Base n to decimal:** Use definition above.
- **Decimal to base-n:** Divide by n until get quotient 0 and take the remainders in reverse direction.

Convert decimals between different systems:

- **Base n to decimal:** Use definition above. There is a shortcut to convert the "decimal" parts:
  1. Convert the "decimal" part into decimal (base-10) store the result in a
  2. The number of "decimal" digits is m
  3. Calculate $\frac{a}{n^m}$, the result is the value of "decimal" parts in the base-10 system.
- **Decimal to base-n:** Following the steps below:
  1. Convert the integer to base-n system.
  2. For the decimal part, multiplying by n
  3. Take the decimal part, repeat step 2 until get 0 in the decimal part.
  4. Now the base-n number, we already get the integer part, the "decimal" part are the **digits of integer part of each multiplication** in the forward sequence.

Example from textbook: Convert 23.375 to binary number.

$$\begin{aligned} 23.375 &= (10111)_2 \\ 2 \cdot 0.375 &= 0.75 \\ 2 \cdot 0.75 &= 1.5 \\ 2 \cdot 0.50 &= 1.0 \end{aligned}$$

So the final result is $23.375 = (10111.011)_2$

Note: Not all fractions can be expressed into a binary. Because you might get repeat result using the technique above. (Example: 0.1)

Floating point numbers: Similar to scientific notation, the floating point number is usually binary. Here is the standard form:

$$x = \pm q \cdot 2^m$$

- $\pm$ describes the sign
- q is the **significand**
- m is the exponent

Note: Usually we use the **normalized** floating number, where $1 \leq q < 2$.

Properties of normalized floating point:

$$x = \pm 1.b_1 b_2 b_3 \ldots b_n \times 2^m = 1.f \times 2^m$$

- **Digits of fraction:** Either 0 or 1
- **Exponent range:** $m \in [L, U]$
- **Precision:** $p = n + 1$
- **Smallest positive normalized:** 2^L
- **Largest positive normalized:** $2^{U+1}\left(1 - \frac{1}{2^{n+1}}\right)$

IEEE-754 standards:

Properties	**IEEE-754 Single Precision**	**IEEE-754 Double Precision**
Number of bits	32	64
Size	4 Bytes	8 Bytes
Sign bit(s)	Index 31 to the right (1 bit)	Index 63 to the right (1 bit)
Exponent bits(c)	Index 30 - 23 to the right (8 bits)	Index 62 - 52 to the right (11 bits)
Fraction bits(f)	Index 22 - 0 to the right (23 bits)	Index 51 - 0 to the right (52 bits)
Expression	$x = (-1)^s 1.f \times 2^m$	$x = (-1)^s 1.f \times 2^m$
Exponent (m)	$c = m + 127$	$c = m + 1023$
Mach. Epsilon (ϵ)	2^{-23}	2^{-52}
Smallest NFP (UFL)	2^{-126}	2^{-1022}
Largest NFP (OFL)	$2^{128}(1 - 2^{-24})$	$2^{1024}(1 - 2^{-53})$

Corner cases of IEEE-754: Here are some corner cases of IEEE-754.

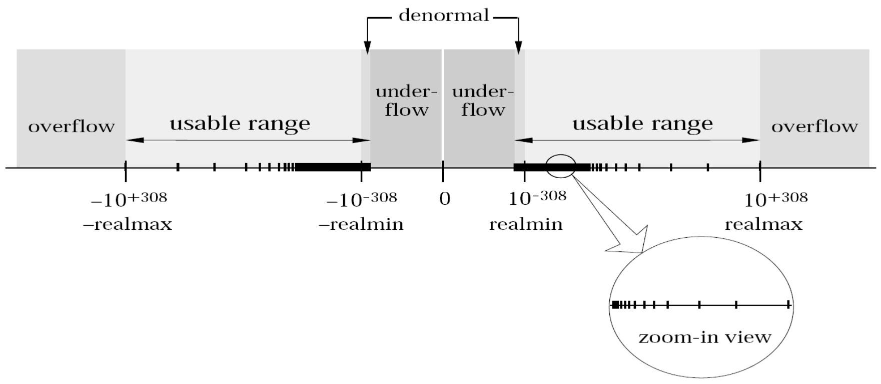
- **Zero:** In our definition of floating point numbers above, we said that there is always a leading 1 assumed. This is true for most floating point numbers. A notable exception is zero. In order to store zero as a floating point number, we store all zeros for the exponent and all zeros for the fractional part. Note that there can be both +0 and -0 depending on the sign bit.
- **Infinity:** If a floating point calculation results in a number that is beyond the range of possible numbers in floating point, it is considered to be infinity. We store infinity with all ones in the exponent and all zeros in the fractional. $+\infty$ and $-\infty$ are distinguished by the sign bit.
- **NaN:** Arithmetic operations that result in something that is not a number are represented in floating point with all ones in the exponent and a non-zero fractional part.

Machine Epsilon(ϵ): defined as the distance (gap) between 1 and the next largest floating point number.

Subnormal numbers: However, we can go even smaller than this by removing the restriction that the first number of the significand must be a 1. These numbers are known as subnormal, and are stored with all zeros in the exponent. Technically, zero is also a subnormal number. **It is important to note that subnormal numbers do not have as many significant digits as normal numbers.**

Properties	**IEEE-754 Single Precision**	**IEEE-754 Double Precision**
Exponent bits(c)	$(00000000)_2$	$(00000000000)_2$
Exponent(m) set to	$m = -126$	$m = -1022$
Smallest SNFP (UFL)	$2^{-23} \times 2^{-126} = 2^{-149}$	$2^{-52} \times 2^{-1022} = 2^{-1074}$

Floating point number line: Figure from the textbook.


denormal
overflow
usable range
under-flow
under-flow
usable range
overflow
-10^{+308}
–realmax
-10^{-308}
–realmin
0
10^{-308}
realmin
10^{+308}
realmax
zoom-in view