7.1 Representation of Floating-Point Numbers

1 Representation

- Representation of floating-point (real) number $N = F \times B^E$
  - base : can be any integer larger than 1 and can be implied or explicit.
  - fraction (exponent) : can be in many formats (e.g. 2's complement formats, sign-magnitude form, or other negative number representation)
- variety of floating-point number formats depend on :
  - what the base is
  - whether the base is implicit or explicit

A. A Simple Floating-Point Format Using 2's Complement

- A simple floating-point format:
  - The **base** for the exponent is 2.

    $\Rightarrow$ the value of the number is $N = F \times 2^E$.
  - The negative **exponents** and **fractions** are represented using the ***2's complement*** form.
  - **Fractional part** will have a leading ***sign bit*** and the other bits are the actual ***fraction bits***.

Sign bit	Fraction bits

    - ***Sign bit***: 0 for positive and 1 for negative
    - The implied binary point is after the first bit.
- Typical floating-point number system:
  - ***F***: 16 ~ 64 bits; ***E***: 8 ~ 15 bits

Example

Sign bit	Fraction bits (4 bits)	Exponent bits (4 bits)

- Assume that we use 4 bits for fraction part and 4 bits for exponent.
- Represent decimal 2.5 and −2.5 in 8-bit 2's complement floating-point format:

$2.5 = 10.1$

$= 1.01 \times 2^1$ (normalized representatio

$= 0.101 \times 2^2$ (4-bit 2's comp fraction)

$\Rightarrow$ F = 0.101, E = 0010 $\Rightarrow$ N = $5/8 \times 2^2$

Sign bit	Fraction bits
0	1 0 1

Exponent bits
0 0 1 0

For −2.5, the same exponent can be used, but the fraction must have a negative sign.

$\Rightarrow$ The 2's comp rep for F is 1.011.

$\Rightarrow$ F = 1.011, E = 0010 $\Rightarrow$ N = $-5/8 \times 2^2$

Sign bit	Fraction bits
1	0 1 1

Exponent bits
0 0 1 0

2 Normalization

- In order to utilize all the bits in F and have the max # (significant figures), F should be normalized so that its magnitude is as large as possible.

- If F is not normalized, left shift F until the sign bit and the next bit are different. For every shift, decrement E by 1.

5 bits		4 bits
Sign bit	Fraction bits	Exponent bits

Examples: Normalization

	5 bits	4 bits	
Unnormalized:	F = 0.0101	E = 0011	$N = 5/16 \times 2^3 = 5/2$
Normalized:	F = 0.1010	E = 0010	$N = 5/8 \times 2^2 = 5/2$
Unnormalized:	F = 1.11011	E = 1100	$N = -5/32 \times 2^{-4} = -5 \times 2^{-9}$
Shift F left:	F = 1.1011	E = 1011	$N = -5/16 \times 2^{-5} = -5 \times 2^{-9}$
Normalized:	F = 1.0110	E = 1010	$N = -5/8 \times 2^{-6} = -5 \times 2^{-9}$

* 5-bit F: −1 ~ +0.9375; 4-bit E: −8 ~ +7

Zero

- Zero cannot be normalized
  $\Rightarrow F = 0.000$ when $N = 0$.
- Any exponent could then be used; however, it is best to have a uniform representation of 0.
- In this format, associate the **negative exponent with the largest magnitude** w/ the **fraction 0**.
  - E.g.: In a 4-bit 2's complement integer number system, the most negative number is 1000, which represents −8.

    $\Rightarrow$ When F and E are 4 bits, 0 is represented by:

    $F = 0.000 \;\; E = 1000 \;\Rightarrow N = 0.000 \times 2^{-8}$

$\boxed{3}$ IEEE-754 FLP Format

- single-precision (32b), double-precision (64b)
  1, 8, 23 / 1, 11, 52
- fractional part: sign-magnitude
  exponent: biased notation
- Designers of IEEE 754 desired a format that is easy to sort.

Subfields of IEEE 754 Formats

- 3 sub-fields of the IEEE 754 FP formats:

Sign	Exponent	Fraction

1.

  - *Fractional part*: sign-magnitude representation
    $\Rightarrow$ There is an explicit sign bit (***S***) for the fraction.
    - **Sign**: 0 for positive and 1 for negative numbers
    - For ***normalized numbers***: a hidden leading 1 before the binary point

      $\Rightarrow$ **Magnitude** (**significand**) of the fraction is $\mathbf{1 + \mathit{F}}$.
  - *Exponent*: biased
    - base of the exponent: 2, is implied

$\Rightarrow$ Normalized number: $N = (-1)^s \times (1 + F) \times 2^{(E - \text{bias})}$

Biased Exponent

- ***exponent*** in the IEEE FP formats: uses a ***biased*** notation
  - For **single-precision *normalized*** FP numbers:
    - Contains the **actual exponent + 127**

      $\Rightarrow$ Converts exponents from $-126 \sim +127$
      into $1 \sim 254$.

      * 0 and 255 are reserved for special cases
  - For **double-precision *normalized*** FP numbers :
    - Contains the **actual exponent + 1023**

      $\Rightarrow$ Converts exponents from $-1022 \sim +1023$
      into $1 \sim 2046$.

      * 0 and 2047 are reserved for special cases

Ex.

Example: Single-Precision

- Represent **13.45** in IEEE single precision FP format:
  - Sign: 0 ⇐ the number is positive
  - Fraction: * .45 is a recurring binary fraction
    - Convert to binary representation:
      13.45 = 1101.01 **1100 1100 1100 1100 1100 … … …**
    - Normalize:
      13.45 = 1.10101 **1100 1100 1100 1100 1100 …** $\times 2^3$
  - Exponent: in biased notation
    - 127 + 3 = 130 or 10000010 in 8-bit binary.

Sign	**Exponent** (8 bits)	**Fraction** (23 bits)
0	1 0 0 0 0 0 1 0	1 0 1 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1

* In hex format, the 32 bits are: 4157 3333

* -13.45 can be represented by just changing the sign bit.

Example: Double-Precision

- Represent **13.45** in IEEE double precision FP format:
  - Sign: 0 ⇐ the number is positive
  - Fraction:
    - Convert to binary representation:
      13.45 = 1101.01 1100 1100 1100 1100 1100 … … …
    - Normalize:
      13.45 = 1.10101 1100 1100 1100 1100 1100 … $\times 2^3$
  - Exponent: in biased notation
    - 1023 + 3 = 1026 or 10000000010 in 11-bit binary.

Sign	**Exponent** (11 bits)	**Fraction** (52 bits)
0	1 0 0 0 0 0 0 0 0 1 0	1 0 1 0 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0

* In hex format, the 64 bits are: 402A E666 6666 6666

Largest and Smallest *Normalized* Numbers

$N = (-1)^s \times (1 + F) \times 2^{(E-127)}$

* Biased exponent: 1~254

- *Single*-precision format:
  - The ***largest*** **positive** ***normalized*** numbers:

Sign	Exponent (8 bits)	Fraction (23 bits)
0	1 1 1 1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

$$\Rightarrow N = (-1)^0 \times [1 + (1 - 2^{-23})] \times 2^{(254-127)}$$
$$= +(2 - 2^{-23}) \times 2^{127} \approx +2^{128}$$

  - The ***smallest*** **positive** ***normalized*** numbers:

Sign	Exponent (8 bits)	Fraction (23 bits)
0	0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$$\Rightarrow N = (-1)^0 \times (1 + 0) \times 2^{(1-127)} = +2^{-126}$$

J.J. Shann 7-20

- overflow: positive exponent is too large to be represented in the exponent field.

- underflow: negative exponent is too large to be represented in the exponent field.

$\boxed{4}$ Special Numbers

Special Cases

* *NaN*: Represent the result of invalid operations, e.g., 0/0.

- ***Special cases*** in IEEE 754 standard:
  - Smallest and highest exponents are used to denote these special cases.

Single Precision		Double Precision		Object Represented
Exponent (8 bits)	Fraction (23 bits)	Exponent (11 bits)	Fraction (52 bits)	
0	0	0	0	0
0	Nonzero	0	Nonzero	± denormalized number
255	0	2047	0	± infinity
255	Nonzero	2047	Nonzero	NaN (not a number)

Denormalized Numbers

* The ***smallest* positive *normalized*** number = $+2^{-126}$

* Denormalized numbers:
Exponent = 0
Fraction = Nonzero

- **Denormalized Numbers:**
  - Single precision:
    - ***Largest* denormalized** number:

      $0.11111111111111111111111 \times 2^{-126} = 2^{-126} - 2^{-149}$
    - ***Smallest* denormalized** number:

      $0.00000000000000000000001 \times 2^{-126} = 2^{-149}$

  $\Rightarrow$ allows numbers b/t 2^{-126} and 2^{-149} to be represented $(2^{-126}, 2^{-149}]$
  - Double precision: allows numbers $(2^{-1022}, 2^{-1074}]$ to be represented

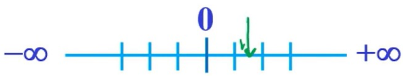
5 Rounding

Rounding

Fraction	Guard bit	Round bit	Sticky bit

- Rounding:
  - When the # of bits available is smaller than the # of bits required to represent a number, rounding is employed.
  - One has to keep more bits in intermediate representations to achieve higher accuracy.
- ***Guard*** and ***round***:
  - the two extra bits that the IEEE standard requires in intermediate representations in order to facilitate better rounding
- ***Sticky bit***:
  - the third intermediate bit used in rounding
  - It is set whenever there are ***non-zero bits*** to the right of the round bit.

Rounding Modes



- Four rounding modes of IEEE standard:
  1. ***Round up***: round towards positive infinity; round up to the next higher number
  2. ***Round down***: round towards negative infinity; round down to the nearest smaller number
  3. ***Truncate***: round towards zero; ignore bits beyond the allowable # of bits
  4. ***Unbiased***: round to nearest

	1	0	0
Fraction	Guard bit	Round bit	Sticky bit

  - If the number falls halfway, round up half the time and round down half the time.

  $\Rightarrow$ Add 1 if the lowest bit retained is 1, and truncate if it is 0.

  $\Rightarrow$ The rounded number always has a 0 in the lowest place.