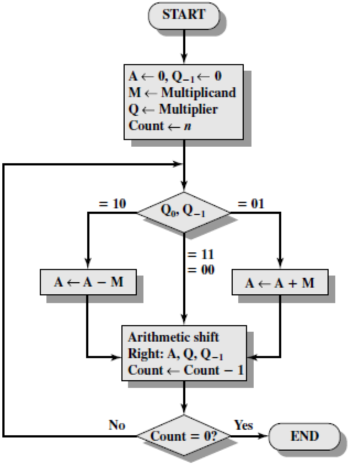
Computer Organization & Architecture

Dr. Sonu Lamba

Department of Computer Science and Engineering
The LNM Institute of Information Technology Jaipur

August 10, 2020

Booth's Multiplication Algorithm



Two major approaches to store real numbers (i.e., numbers with fractional component)

- Fixed Point Notation
- Floating Point Notation

Fixed Point Representation

Fixed-Point Representation:

- In fixed point notation, a fixed number of digits after the decimal point
- Fixed number of bits for integer part and for fractional part.
- a fixed-point number representation consists of three parts:
    - Sign field
    - Integer field
    - Fractional field

Fixed Point Representation

- In all the three schemes, the most-significant bit (msb) is called the sign bit.
  - represent the sign of the integer: 0 for positive integers and 1 for negative integers.
- The magnitude of the number is interpreted differently in different schemes.
- Example: assume number is using 32-bit format which reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part.
  - Then, -43.625 is represented as following:

1	000000000101011	1010000000000000
Sign bit	Integer part	Fractional part

Advantages and Disadvantages of Fixed Point Representation

- Advantage: performance
- Disadvantage: relatively limited range of values that they can represent.
  - smallest positive number and largest positive number which can be store in 32-bit representation as given below.

	Sign bit	Integer part	Fractional part
Smallest	0	000000000000000	0000000000000001
Largest	0	111111111111111	1111111111111111

Floating Point Representation

- What is floating point number?
- Need of floating point representation ?
- Floating-point representation:
  - Scientific notation
  - General representation:

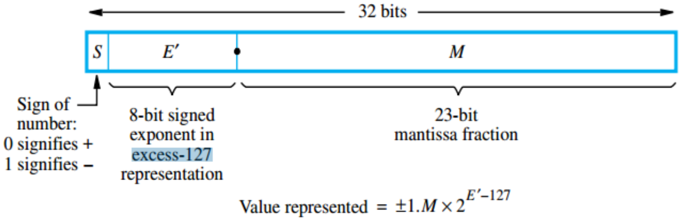
$$\pm M(significand \setminus mantissa) \times B(base)^{E(exponent)}$$

Floating-point Representation in Modern Computers

- IEEE (Institute of Electrical and Electronics Engineers) Standard 754
  1. 32-bit single-precision
  2. 64-bit double-precision.

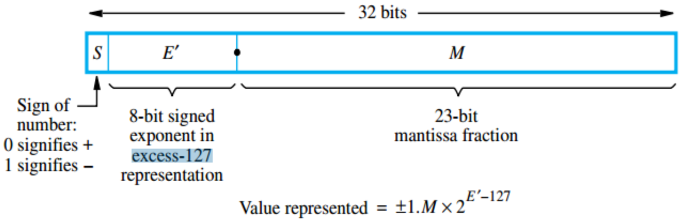
IEEE-754 32-bit Single-Precision Floating-Point Numbers

- In 32-bit single-precision floating-point representation:
  - The most significant bit is the sign bit (S)
  - The next 8 bits, $E^{'}$, represent the signed exponent of the scale factor (with an implied base of 2)
  - The remaining 23 bits, M represents mantissa fraction



(a) Single precision

IEEE (Institute of Electrical and Electronics Engineers) Standard 754



(a) Single precision

- Why the exponent field id $E^{'}$?

Single Precision

- S represents sign bit for the number.
- E represents the signed exponent of the scale factor (with an implied base of 2)
- M represents fractional part of the significant bits called mantissa, always has a leading 1
  $B = 1 \cdot M = 1 \cdot b_{-1} b_{-2} \cdots b_{-23}$
  $V(B) = 1 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} \cdots b_{-23} \times 2^{-23}$
- Normalized number?
- The end values of E are used to represent special values.

Special Values

- The end values 0 and 255 of the excess-127 exponent E are used to represent special values.
  1. When E $= 0$ and M$= 0$: represents zero. If sign bit is 0, then $+0$, else -0.
  2. When E $= 255$ and M $= 0$: the value ∞ is represented, If sign bit is 0, then $+\infty$, else $-\infty$.
  3. The sign bit is still used in these representations, so there are representations for ± 0 and $\pm\infty$
  4. When E $= 0$ and $M \neq 0$: represent denormalized number, then their value is $\pm 0 \cdot M \times 2^{-126}$
  5. When E $= 255$ and $M \neq 0$: Not a Number (NaN).

Exceptions

- A processor must set exception flags if
  1. underflow
  2. overflow
  3. divide by zero
  4. inexact
  5. invalid.

Practice Question: Single Precision

1. Suppose that IEEE-754 32-bit floating-point representation pattern is 0 10000000 110 0000 0000 0000 0000 0000.
2. Suppose that IEEE-754 32-bit floating-point representation pattern is 1 01111110 100 0000 0000 0000 0000 0000.
3. Suppose that IEEE-754 32-bit floating-point representation pattern is 1 00000000 000 0000 0000 0000 0000 0001.
4. Compute the largest and smallest positive numbers that can be represented in the 32-bit normalized form

Work Space

Work Space

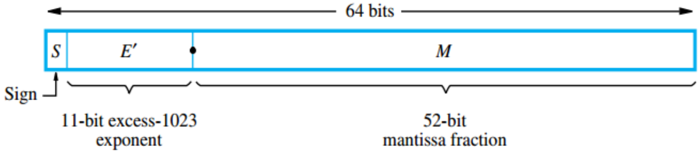
Work Space

Work Space

Work Space

IEEE-754 64-bit Double-Precision Floating-Point Numbers

- Provides more precision and range for floating-point numbers



Value represented $= \pm 1.M \times 2^{E'-1023}$

The number (N) is calculated as follows:

- Normalized form: For $1 \leq E \leq 2046$, $N = (-1)^S \times 1 \cdot F \times 2^{(E-1023)}$
- Denormalized form: For E = 0, $N = (-1)^S \times 0 \cdot F \times 2^{(-1022)}$ These are in the denormalized form
- For E = 2047, N represents special values, such as $\pm\infty$ (infinity), NaN (not a number)

Practice Questions

What about converting to decimal?

- 0011 1110 1010 1000 0000 0000 0000 0000

Arithmetic Operations on Floating-Point Numbers

- Addition and Subtraction Rule
  - Compare the magnitude of the two exponents and make suitable alignment to the number with the smaller magnitude of the exponent.
  - Set the exponent of the result equal to the larger exponent.
  - Perform addition/subtraction on the mantissas and determine the sign of the result.
  - Normalize the resulting value, if necessary.
  - Example: 1.1100×2^4 and 1.100×2^2 ?

Arithmetic Operations on Floating-Point Numbers

1. Multiplication Rule
   - Add the exponents and subtract 127 to maintain the excess-127 representation.
   - Multiply the mantissas and determine the sign of the result.
   - Normalize the resulting value, if necessary
2. Divide Rule
   - Subtract the exponents and add 127 to maintain the excess-127 representation.
   - Divide the mantissas and determine the sign of the result.
   - Normalize the resulting value, if necessary.

Example: $X = 1.000 \times 2^{-2}$ and $Y = -1.010 \times 2^{-1}$?