# Floating Point Numbers

## Chapter 3 of the Book of John P. Hayes

## PP-196 to 202 of David A. Patterson

# Need for Floating Point Number

- ✔ The range of number represented by a fixed-point number is insufficient for many applications, particularly, when very large and very small numbers are required.
- ✔ Example: $1.0 * 10^{18}$
- ✔ Scientific notation allows to represent such numbers using relatively few digits.

# Basic Format of Floating Point Number

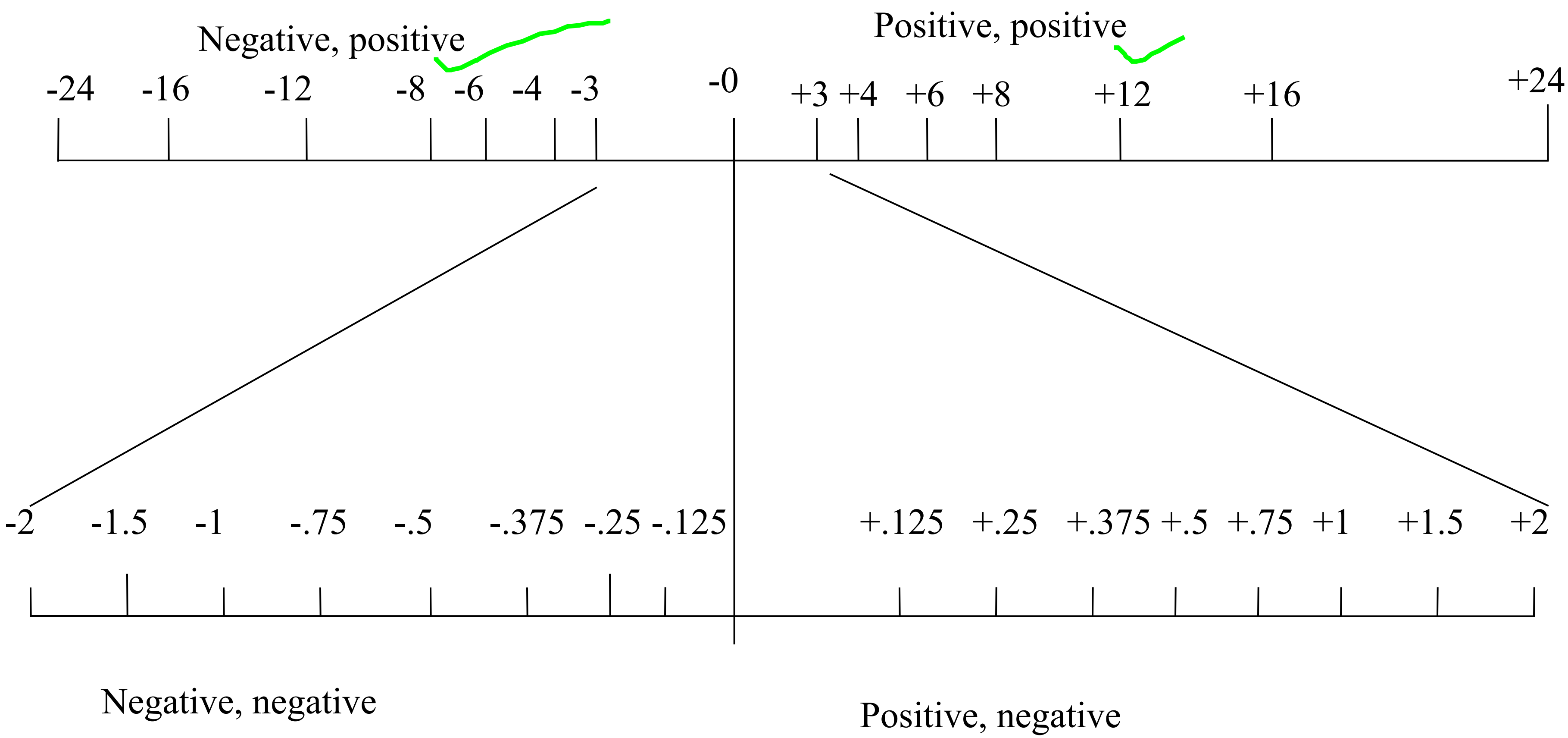
- A real number is represented as $\boxed{M \times B^E}$ where
  M= mantissa, E= exponent and B= base
- Example: $1.0 \times 10^{18}$ where 1.0 = sign magnitude mantissa
  10 = base and 18 = sign magnitude exponent.

# Representation of Floating Point Number

- A floating point number is represented as a word (M, E) consisting of a pair of fixed-point numbers: M, which is usually a fraction or integer and E, which is an integer.
- Since, B is constant, it is not stored but is simply built into the circuit that process the number.

# Example of Floating-point Number

- M,E= 3 bit sign magnitude numbers and assume the values ±0, ±1, ±2, ±3 and B=2.
- (M, E) = (x00, xxx) represent 0.



# Floating-Point Number

- The floating point representation of most real numbers is only approximate. For example, 1.25 is approximated by (011,101) representing 1.5 or by either (001, 000) or (001, 100).
- The result of most calculations with floating point arithmetic only approximate the correct result. For example, the exact result of the addition (011,001) + (011, 010) =18 which is not representable. The closest representative number of 18 is 16 (010, 011).

# Normalization

- The same number can be represented in many ways. For example: $1.0 \times 10^{18}$, $0.1 \times 10^{19}$, $1000000 \times 10^{12}$ etc.
- It is desirable to have a unique or normal form for each representable number in a floating point system.
- A binary number is normalized when it have one non-zero digit to the left of the decimal point.
- Example: for IEEE 754 the normalized form 1.M.
- An un-normalized number is normalized by shifting the mantissa to the right or left and appropriately incrementing or decrementing the exponent.

# Normalization

## Advantages:

- ✔ It simplifies the exchange of data.
- ✔ It simplifies the floating point arithmetic algorithm
- ✔ It increases the accuracy of the numbers that can be stored in a word, since the unnecessary leading 0s are replaced by real digits to the right of the binary point.

# Biasing

- If we use 2's complement or other notation in which negative exponents have 1 in MSB, a negative number will look like a big number. For example, $1.0 \times 2^{-1}$ is represented as

  1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ..................

  $1.0 \times 2^{1}$ is represented as,

  0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ..........................

- It is desirable to represent most negative exponent as 00...000 and most positive as 11....111. This convention is called biased notation.
- In biased notation, bias is the number to be subtracted from normal, unsigned representation to determine the real value.
- -1+127=126=01111110 and +1+127=128=10000000

# Biasing

| Exponent E | Unsigned value | Bias 127 | Bias 128 |
|---|---|---|---|
| 111…11 | 255 | +128 | +127 |
| 111…10 | 254 | +127 | +126 |
| …………. | ………… | ……… | ………... |
| 100…01 | 129 | +2 | +1 |
| 100…00 | 128 | +1 | 0 |
| 011…11 | 127 | 0 | -1 |
| 011…10 | 126 | -1 | -2 |
| ……………. | ……………… | ………. | ……… |
| 000…01 | 1 | -126 | -127 |
| 000…00 | 0 | -127 | -128 |

8-bit biased exponent with bias=127 (excess-127) and bias = 128 (excess-128)

# IEEE 754 Floating Point Number

| 1 | 8 | 23 |
|---|---|---|
| S | E | M |

IEEE 754 standard 32-bit floating-point number format

- ✔ S= 1 bit sign representation
- ✔ E = 8 bit excess-127. The actual exponent is E-127.
- ✔ M=23 bit mantissa [ fraction part of sign-magnitude binary significand with hidden bit]

# IEEE 754 Floating Point Number

- A real number N = $(-1)^s 2^{E-127}(1.M)$ where, $0 < E < 255$
- N=-1.5 is represented as

$$\underset{1}{1}\ \ \underbrace{01111111}_{8}\ \ \underbrace{10000000000\ldots..0}_{23}$$

$$(-1)^S\, 2^{E-127} \times 1.M$$

Double precision

# IEEE 754 Floating Point Number

| 1 | 11 | 52 |
|---|---|---|
| S | E | M |

IEEE 754 standard 64-bit floating-point number format

✔ A real number $N = (-1)^s 2^{E-1023}(1.M)$ where, $0 < E < 2047$

# Converting from Binary to Decimal Floating Point

- What is the decimal single-precision floating point number that corresponds to the bit pattern 01000100010010010000000000000000?

  E

- Use the equation

  $X = (-1)^S \times 2^{E-127} \times (1.M)$ Here, S = ? E = ? M = ?

  $S = 0$

  $E = 10001000_2 = 136_2$

  $1.M = 1.\,10010010000000000000000 = 1 + 2^{-1} + 2^{-4} + 2^{-7}$
  $= 1.5703125$

  so

  $X = (-1)^0 \times 2^{136-127} \times 1.5703125 = 804 = 8.04 \times 10^2$

# Converting from Decimal to Binary Floating Point

- What is the binary representation for the single-precision floating point number that corresponds to $X = -12.25_{10}$?
- What is the normalized binary representation for the number?

  $-12.25_{10} = -1100.01_2 = -1.10001_2 \times 2^3$
- What are the sign, stored exponent, and normalized mantissa?

  S = 1 (since the number is negative)

  $E = 3 + 127 = 130 = 128 + 2 = 10000010_2$

  $M = 10001000000000000000000_2$

  $X = 11000001010001000000000000000000_2$

Overflow:

A situation in which a positive exponent becomes to large to fit in the exponent field.

Underflow:

A situation in which a negative exponent becomes too large to fit in the exponent field.

# Handling Different Exceptional Conditions

- If E=255 and M≠0, then N=NAN
- If E=255 and M=0, then N=$(-1)^S\infty$
- If 0<E<255, then N= $(-1)^s 2^{E-127}(1.M)$
- If E=0 and M≠0, then N= $(-1)^s 2^{E-126}(0.M)$
- If E=0 and M=0, then N= $(-1)^S 0$.

$1.M \times 2^E$