Other Systems For Representing Numbers

Binary Coded Decimal

- BCD uses four bits to encode each decimal digit.
- 479 could be encoded in two bytes as
  0000 0100 0111 1001

	4	7	9
0000	0100	0111	1001

- The Intel architecture has only a few instructions to do arithmetic on BCD numbers.

Floating Point

- Floating-point numbers allow an arbitrary number of decimal places to the right of the decimal point.
  - For example: $0.5 \times 0.25 = 0.125$
- They are often expressed in scientific notation.
  - For example:

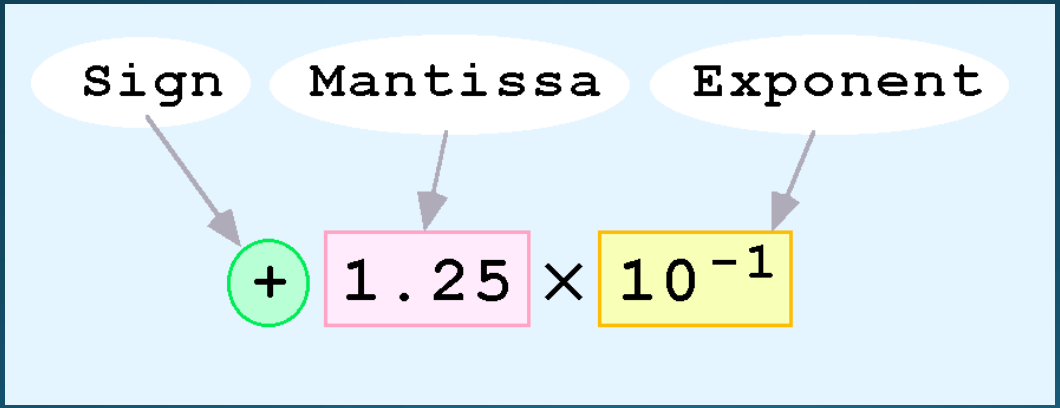
  $0.125 = 1.25 \times 10^{-1}$

  $5{,}000{,}000 = 5.0 \times 10^{6}$

Floating Point

- Computers use a form of scientific notation for floating-point representation
- Numbers written in scientific notation have three components:

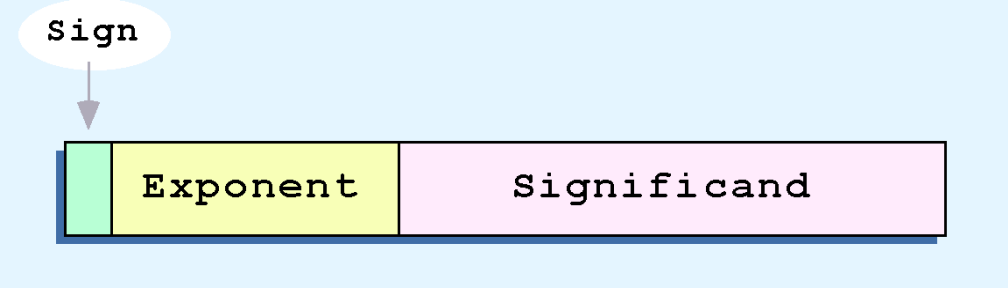
$$0.125 = 1.25 \times 10^{-1}$$



*** This is to introduce the concept. More accurate definition will be provided in the floating point discussion

Floating Point

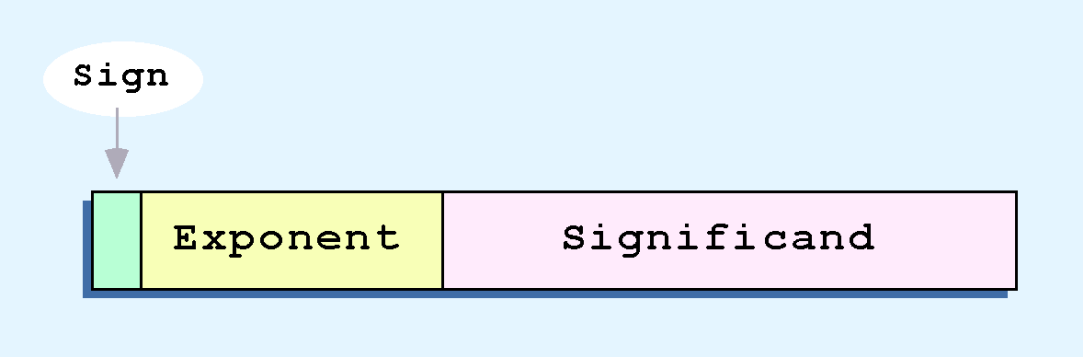
- Computer representation of a floating-point number consists of three fixed-size fields:



- This is the standard arrangement of these fields.

  Note: Although "significand" and "mantissa" do not technically mean the same thing, many people use these terms interchangeably. We use the term "significand" to refer to the fractional part of a floating point number.

Floating Point



- The one-bit sign field is the sign of the stored value.
- The size of the exponent field determines the range of values that can be represented.
- The size of the significand determines the precision of the representation.

Floating Point format

IEEE-754 single precision is a popular 32-bit format

- Write the number in base 2 "scientific notation"
  - sign bit, 8-bit exponent, 23-bit mantissa/significand
  - normalized as 1.xxxxx
  - leading 1 is hidden
- Sign bit (0 positive, 1 negative)
- 8-bit exponent in excess(or bias) 127 format
  - NOT excess 128
  - 0000 0000 and 1111 1111 are reserved
  - +0 and -0 is zero exponent and zero mantissa
  - 1111 1111 exponent and zero mantissa is infinity
- 23 bits for fraction (omitting the leading 1)

Floating Point format

Significand Convert to Binary

Convert X.Y

- Represent X as :

  $?.2^{n} + ?.2^{n-1} + \ldots ?.2^{1} + ?.2^{0}$

- Represent Y as :

  $?.2^{-1} + ?.2^{-2} + \ldots ?.2^{-(p-1)} + ?.2^{-p}$

- Arrange the coefficients as :

  <??..??>.<??..??>

Floating Point format

Exponent: Range of values

8-bit Exponent	Calculation	Representation
0000 0000	NA	Reserved
0000 0001	1-127 = -126	-126_{10}
0000 0010	2-127 = -125	-125_{10}
……	……	……
0111 1111	127 – 127 = 0	0_{10}
……	……	……
1111 1110	254-127=127	127_{10}
1111 1111	NA	Reserved

Floating Point Example

- $78.375_{10} = 1001110.011_2$
- $1001110.011_2 = 1.001110011 \times 2^6$
- 0 10000101 00111001100000000000000