IEEE 754 Floating Point Standard

In lecture slides CO2103 Chapter 03 on Background, we briefly mentioned how computer stores floating point numbers. The format used in the representation of floating point number in the computer is based on the *IEEE 754 Floating Point Standard*.

All floating point numbers will be normalized and the normalized form will be stored in the computer in accordance to *IEEE 754* standard.

> Normalized form: $\pm 1.xxxxxx... \times 2^{yyyy...}$
>
> IEEE 754 Floating Point Standard: $-1^S \times (1.0 + 0.M) \times 2^E$

The *Sign* (*S*) bit indicates if the number is positive (S=0) or negative (S=1). With normalized form, only the fractional part of the mantissa needs to be stored. The *Mantissa* (*M*) bits are the *xxxxxx...* after the radix point. *M* is stored in natural binary form. The *Exponential* (*E*) bits are the *yyyy...*, which are represented in *bias-m* to ease comparisons.

Using **normalized scientific notation**
1. Simplifies the exchange (and representation) of data that includes floating-point numbers
2. Simplifies the arithmetic algorithms to know that the numbers will always be in this form
3. Increases the accuracy of the numbers that can be stored in a word, since each unnecessary leading 0 is replaced by another significant digit to the right of the decimal point

Under IEEE 754 standard, floating point numbers can be represented in either of the two precisions: *Single-Precision* (32-bit) or *Double-Precision* (64-bit).

Bit No	Size	Field Name
31	1 bit	Sign (S)
23-30	8 bits	Exponent (E)
0-22	23 bits	Mantissa (M)

Single-Precision

Bit No	Size	Field Name
63	1 bit	Sign (S)
52-62	11 bits	Exponent (E)
0-51	52 bits	Mantissa (M)

Double-Precision

Single-Precision floating point numbers will occupy 32 bits and give approx range of $\pm 10^{-38} \ldots 10^{38}$. The *Exponent* (*E*) is represented in bias-127.

Double-Precision floating point numbers will occupy 64 bits and give approx range of $\pm 10^{-308} \ldots 10^{308}$. The *Exponent* (*E*) is represented in bias-1023.

Few examples for Single-Precision:

Number (binary)		Normalized (binary)	S	E (8-bit in bias-127)	M (23-bit)	IEEE 754 Single (32-bit)
-10.00111	=	-1.0001111×2^1	1	$1+127=128=10000000_2$	0...0001111	110000...0111
101101.111011	=	1.01101111011×2^5	0	10000100	0...01101111011	0100001...01101111011
-0.001111	=	-1.111×2^{-3}	1	01111100	0...0111	10111110...0111
0.0000101111	=	1.01111×2^{-5}	0	01111010	0...01111	001111010...01111

There are two potential errors in representing a floating numbers in IEEE 754 format:
- *Overflow* - the exponent is too large to be represented in the Exponent field
- *Underflow* - the number is too small to be represented in the Exponent field

To reduce the chances of underflow/overflow, can use 64-bit Double-Precision arithmetic

For further reference: http://babbage.cs.qc.edu/IEEE-754/References.xhtml.
The above material was prepared with reference to http://www.doc.ic.ac.uk/~ih.

Exercises:

1. Determine the normalized binary for the following decimal numbers:
   a) 234.625
   b) -890.375
   c) -0.001007080078125
   d) 0.000091552734375
   (Ans. a) 11101010.101, b) -1101111010.011, c) -0.000000000100001, d) 0.000000000000011)

2. Represent the above floating point numbers in IEEE 754 Single-Precision format. Write your answers in Hex.
   (Ans. a) 436AA000, b) C45E9800, c) BA840000, d) 38C00000)

3. Determine the decimal value of $BFC9400000000000_{16}$, which is an IEEE 754 Double-Precision number.
   (Ans. -1.9726562500000000e-1)