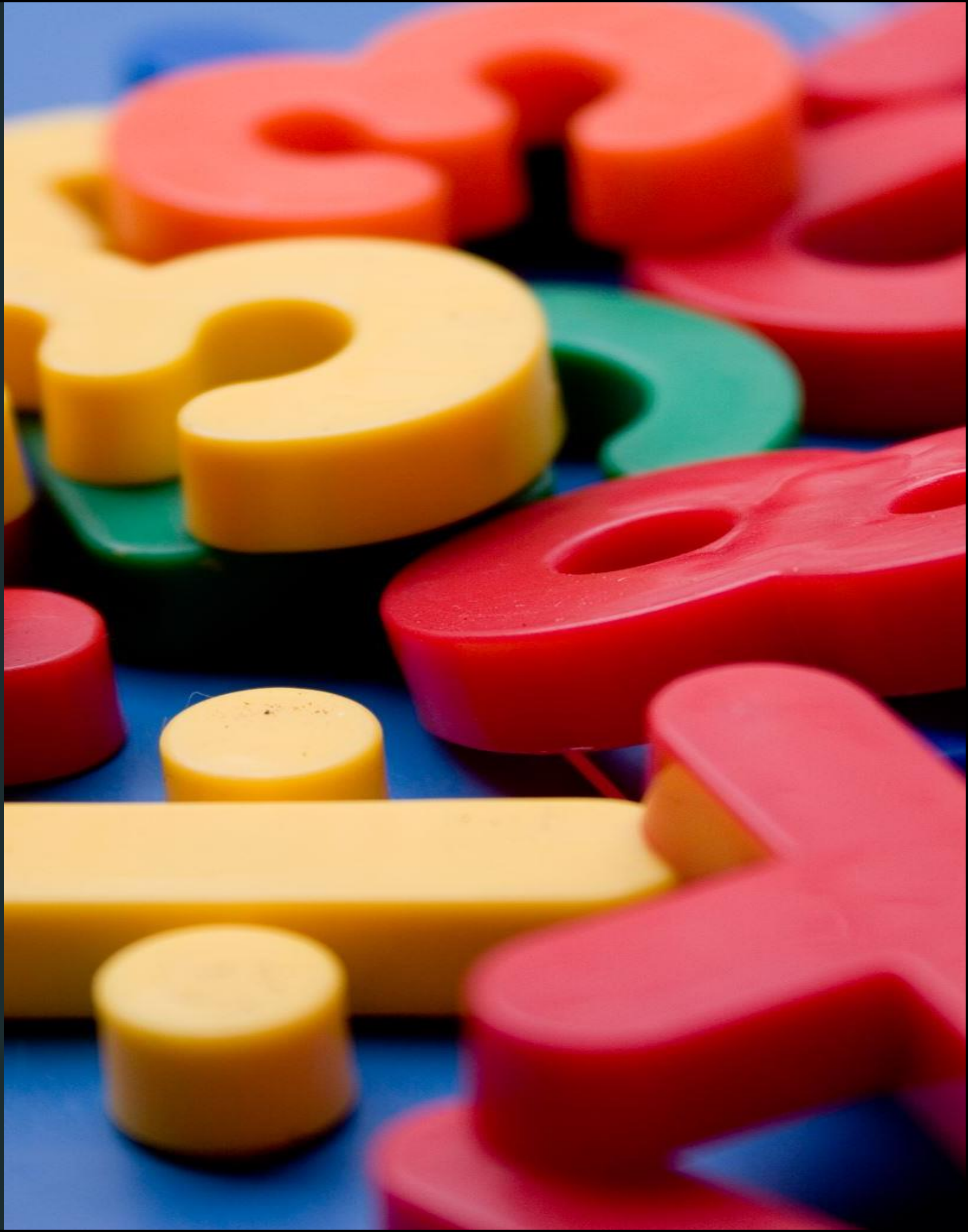
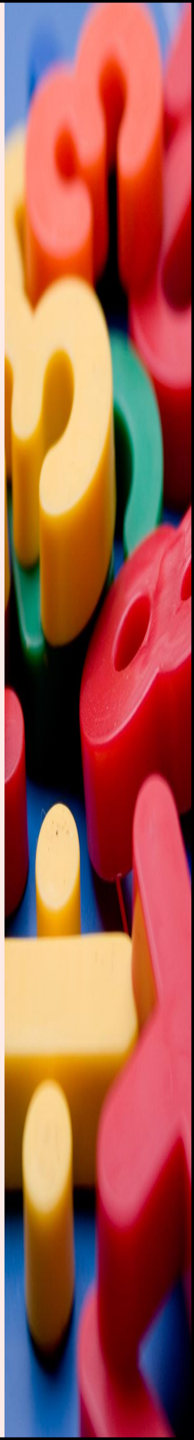
FLOATING POINT NUMBERS

Deepa Mathews



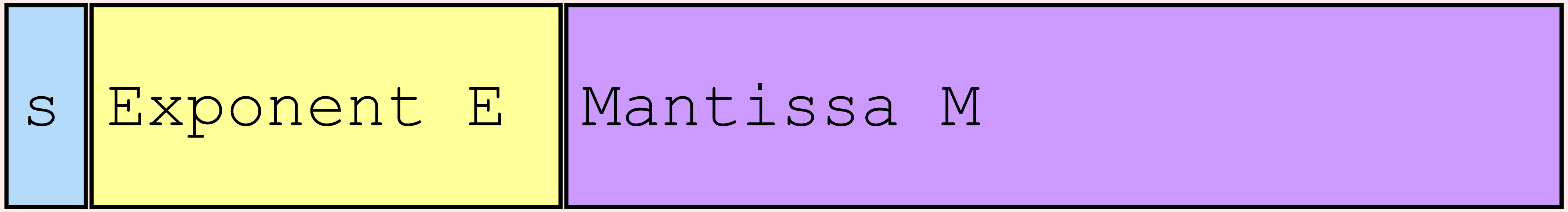
Floating Point Numbers

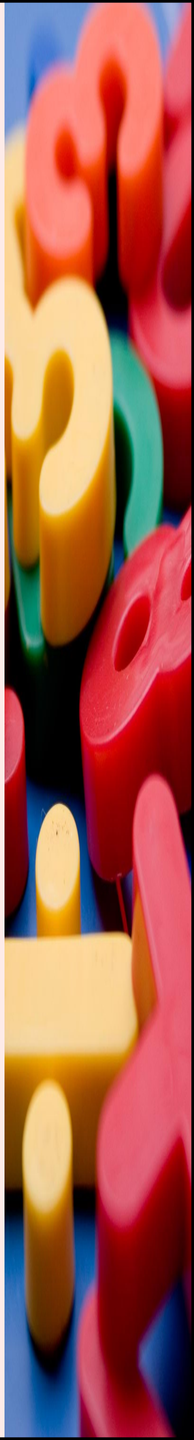
- Numbers that have decimal places are called floating point numbers or real numbers. Eg:, 4.53, 10.001, and -9345.178 are **floating point numbers**. Numbers that do not have decimal places are called **integers** (eg: 237, 9000,-23)
- As the name implies, **floating point numbers** are **numbers** that contain **floating** radix points. i.e, it can be placed anywhere relative to the significant digits of the **number.**
- Representation of floating point number is not unique. For example, the number 55.66 can be represented as 5.566×10^1, 0.5566×10^2, 0.05566×10^3, and so on.



Floating Point Representation

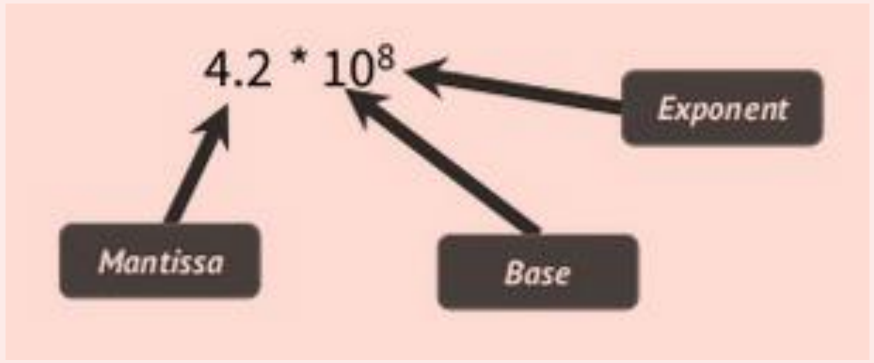
- $-1^{s} M \; 2^{E}$
  - **Sign bit** *s* determines whether number is negative(s=1) or positive(s=0)
  - **Mantissa** (Significand) *M* represents the magnitude of the number.
  - **Exponent** *E* represents the number of places that the decimal point (or binary point) is to be moved. ie E weights value by power of radix





Floating Point Numbers

- **Decimal system**: floating point numbers are expressed in scientific notation by stating a number (mantissa) and an exponent of 10(base/radix).



- **Binary numbers** can also be expressed in the same notation by stating a number and an exponent of 2

$$111.01 \times 2^1$$

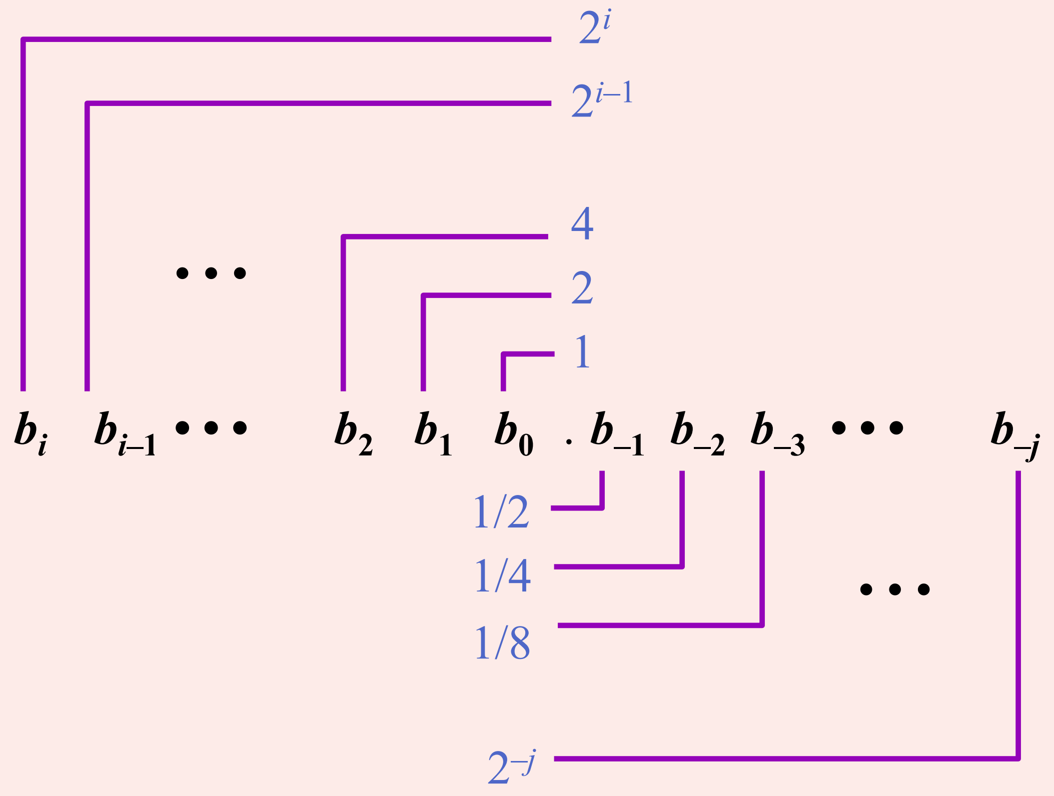
Only the mantissa and the exponent are stored

Fractional Binary Numbers

Examples

101.11_2 $\quad$ 10.111_2

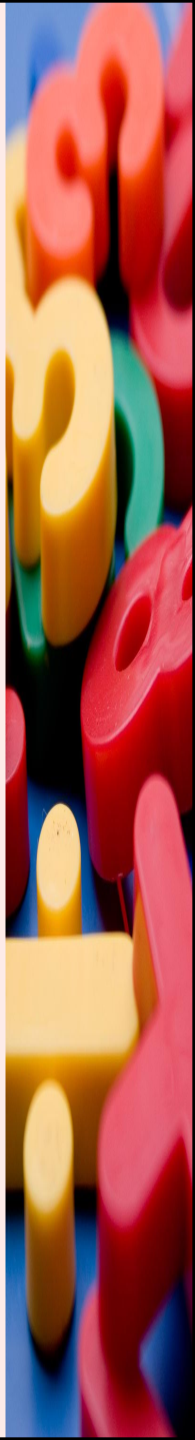
5 3/4 $\quad$ 2 7/8


2^i
2^{i-1}
4
2
1
• • •
b_i b_{i-1} • • • b_2 b_1 b_0 . b_{-1} b_{-2} b_{-3} • • • b_{-j}
1/2
1/4
1/8
• • •
2^{-j}

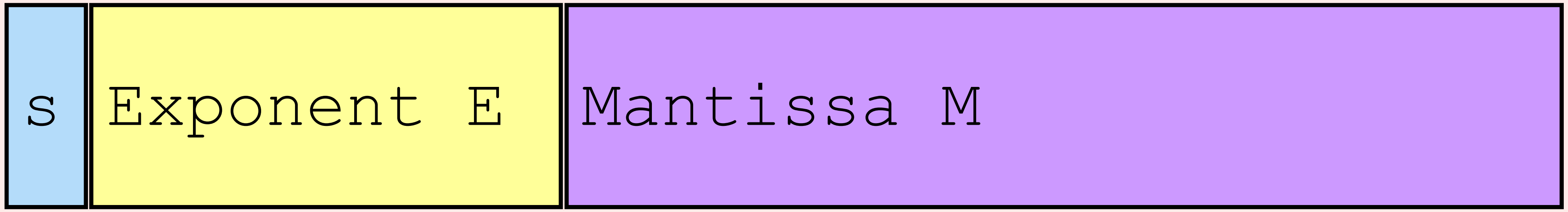
- Bits to right of “binary point” represent fractional powers of 2
- Bits to left of “binary point” represents the multiplied powers of 2

Double Precision Numbers

- For any computer, the word length is fixed. A 32-bit register can store 2^{32} different values in each register.
- If numbers greater than this are to be expressed, **two** storage locations need to be used, ie, each such number has to be stored in two registers. This is called **double precision**. Leaving the MSB which is the sign bit, this allows a 63 bit number length with two 32 bit registers.
- If still larger numbers are to be expressed, **three** registers are used to store each number. This is called **triple precision**

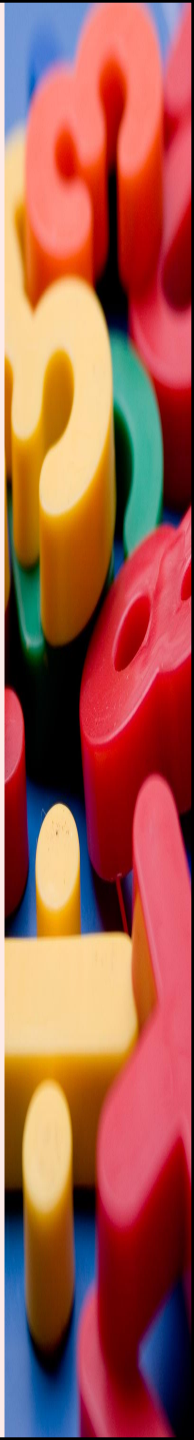


IEEE Floating Point Representations



For binary floating point numbers, the format is defined by ANSI/IEEE Standard in 754-1985 in 3 forms

- **Single precision**: 8 exp bits, 23 mantissa bits - 32 bits total
- **Double precision**: 11 exp bits, 52 mantissa bits - 64 bits total
- **Extended precision**: 15 exp bits, 63 mantissa bits - Stored in 80 bits (1 bit wasted) - Only found in Intel-compatible machines



"Normalized" Numeric Values

Mantissa is coded with implied leading 1

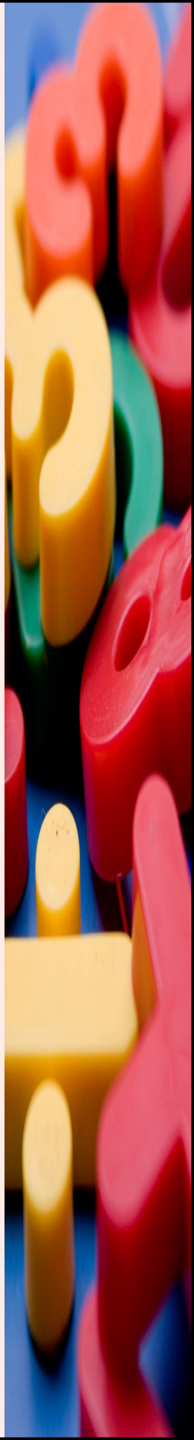
$M = 1.xxx\ldots x_2$

xxx…x bits of mantissa

Leading bit is always 1

Exponent is coded as *biased* value, $E = Exp - Bias$

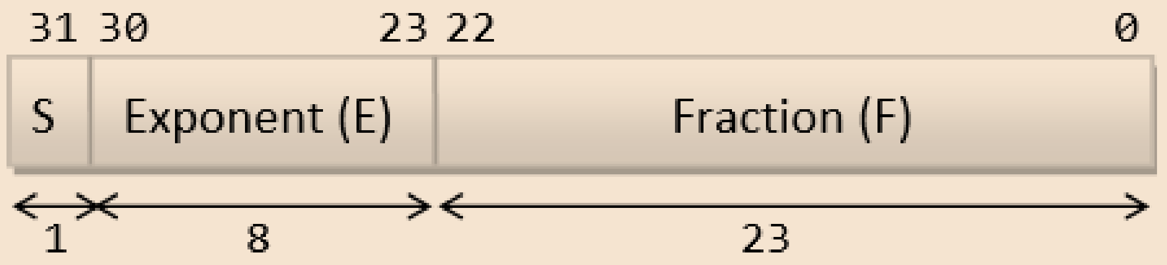
- **Single precision**: 127 (*Exp*: 1…254, *E*: -126…127)
- **Double precision**: 1023 (*Exp*: 1…2046, *E*: -1022…1023)
- In general: $Bias = 2^{e-1} - 1$, where e is number of exponent bits



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

In **32-bit single-precision** floating-point representation:

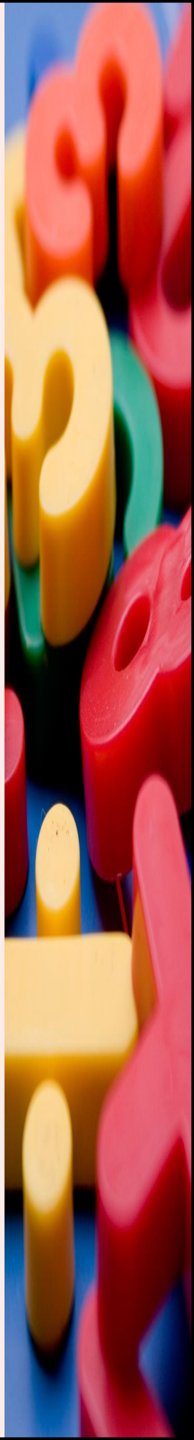
- MSB is the *sign bit* (S), with 0 for positive and 1 for negative numbers.
- The next 8 bits represent *exponent* (E).
- The remaining 23 bits represents ***fraction* (F)**, also called as **mantissa part**



32-bit Single-Precision Floating-point Number

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

- In the mantissa or fractional part, the binary point is understood to be the left of the 23 bits. Effectively there are 24 bits in the mantissa because in any binary number the MSB is always a 1. Therefore this 1 is understood to be there although it does not occupy an actual bit position.
- The 8 bits in the exponent represents a **biased exponent**, which is obtained by adding **127** to the actual exponent. It is also called as **excess 127 notation and** allows a range of actual exponent values from -126 to +127. It's purpose is to allow very large or small numbers without requiring a separate sign bit for the exponent.

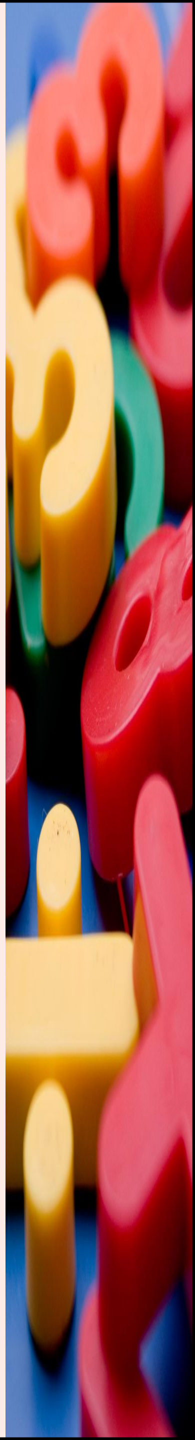


IEEE-754 32-bit Single-Precision Floating-Point Numbers

- Eg: 1011010010001 can be expressed by moving the binary point 12 places to the left and then multiplying by the appropriate power of two.

$$1011010010001 = 1.011010010001 \times 2^{12}$$

- The exponent, 12, is expressed as a **biased exponent** by adding 127 to it. (12+127=139). The binary equivalent of 139 is 10001011. So, the biased exponent (E) is 10001011.

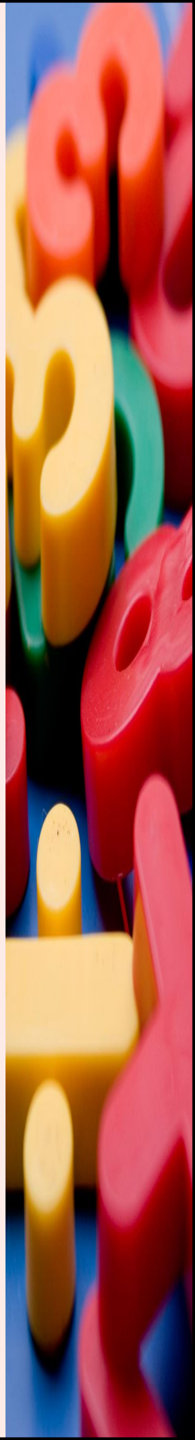


IEEE-754 32-bit Single-Precision Floating-Point Numbers

Express the binary no **1011010010001** in floating point format ?

- Normalize the given number: i.e 1.011010010001* 2^{12}
- Mantissa, M (23 bits) = **01101001000100000000000**
- S = **0** as it is a positive number
- Exponent 12 is expressed as biased exponent by adding 127 (12+127 = 139) i.e exponent E = 139 = $(\mathbf{10001011})_2$
- So, the floating point no is ,

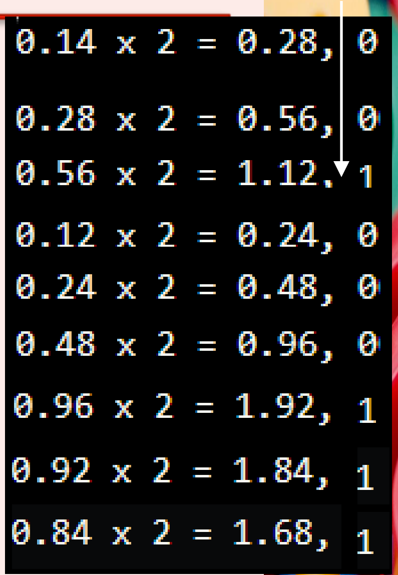
0	**10001011**	**01101001000100000000000**
S	**E**	**M**



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

Represent the number **3.14** in Single Precision Format

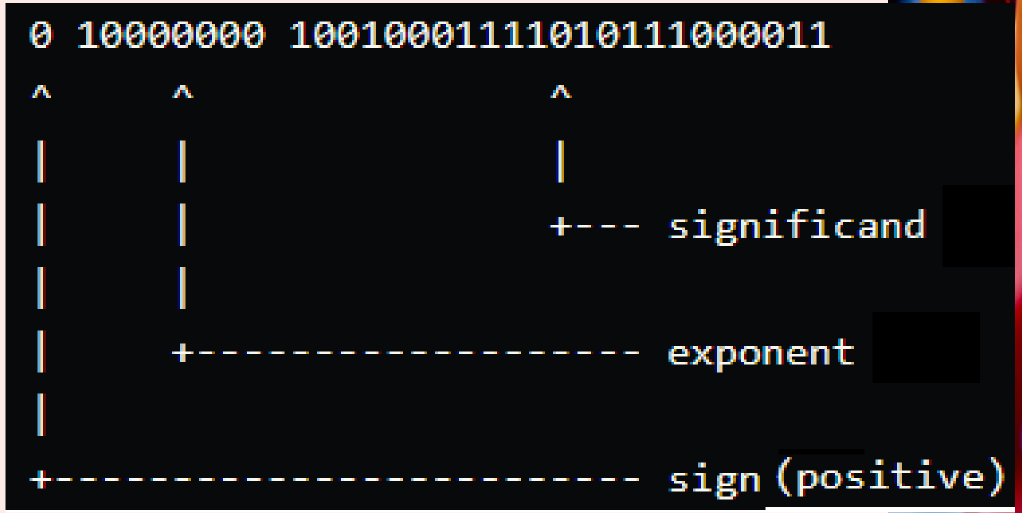
- **Sign bit is 0** as it is a positive number
- **Mantissa**
  - Binary of 3 is 0011 and of .14 is 001000111…, so Mantissa = 11.001000111…
  - Normalize the Mantissa and adjust the exponent accordingly, **1.1001000111… x 2^1**
  - **Mantissa in 23 bits : 10010001111010111000011**
- **Exponent**
  - Add the bias of 127 to the exponent 1 and store it
  - **Biased Exponent, E =1+127=$(128)_{10}$=$(1000\ 0000)_2$**



```
0.14 x 2 = 0.28, 0

0.28 x 2 = 0.56, 0
0.56 x 2 = 1.12, 1
0.12 x 2 = 0.24, 0
0.24 x 2 = 0.48, 0
0.48 x 2 = 0.96, 0
0.96 x 2 = 1.92, 1
0.92 x 2 = 1.84, 1
0.84 x 2 = 1.68, 1
```

So 3.14 can be represented as



```
0 10000000 10010001111010111000011
^      ^                  ^
|      |                  |
|      |                  +--- significand
|      |
|      +------------------- exponent
|
+------------------------- sign (positive)
```

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

<u>Represent the number -15213 in Single Precision Format</u>

Sign bit is 1 as it is a negative number

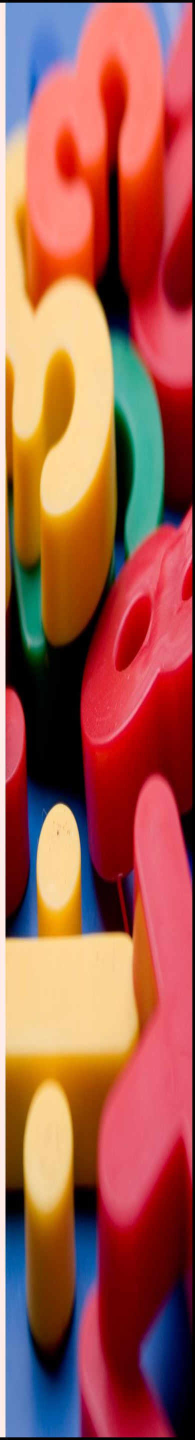
$-15213_{10} = \mathbf{11101101101101}_2 = 1.1101101101101_2 \text{ X } 2^{13}$

Mantissa, $M = 1.\underline{1101101101101}_2$

$= \mathbf{\underline{1101101101101}0000000000}_2$

Exponent , $E = 13$, $Bias = 127$, $Exp = 13+127 = 140 = \mathbf{10001100}_2$

15213: ***1*** **10001100** **11011011011010000000000**



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

Convert decimal no $3.248 * 10^4$ to a single precision floating point binary number

- Convert the decimal no to binary, $3.248 * 10^4 = 32480 = (111111011100000)_2$
- Normalize, $= 1.\mathbf{11111011100000} * 2^{14}$

  So, Mantissa is fractional 23-bit binary no 11111011100000000000000
- Biased exponent $= 14 + 127 = 141 = (10001101)_2$

The complete floating point no is,

0	10001101	11111011100000000000000