Floating-point

Definition

$$x = \pm \left(d_0 + \frac{d_1}{\beta^1} + \frac{d_2}{\beta^2} + \ldots + \frac{d_{p-1}}{\beta^{p-1}} \right) \beta^E$$

β : base

p : precision

$[L, U]$: exponent range

$0 \leq d_i \leq \beta - 1$

$i = 0, \ldots, p - 1$

$E \in [L, U]$

Definition

- mantissa : $d_0 d_1 d_2 \ldots d_{p-1}$
- fraction : $d_1 d_2 \ldots d_{p-1}$
- sign, exponent, mantissa : stored separately

Definition

- normalisation : d_0 always non-zero unless zero
- in $\beta = 2$, $d_0 = 1$ and not stored to save space

Properties

- floating number system : finite and discrete

total number of normalized floating numbers

$$2(\beta - 1)\beta^{p-1}(U - L + 1) + 1$$

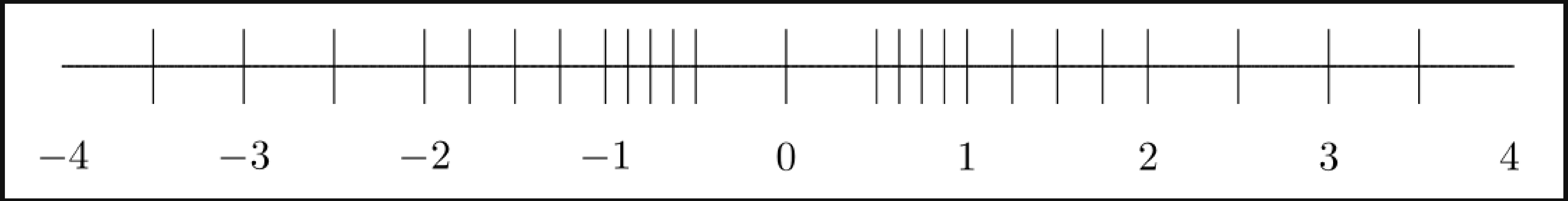
underflow level : $UFL = \beta^L$

overflow level : $OFL = \beta^{U+1}(1 - \beta^{-p})$

Properties

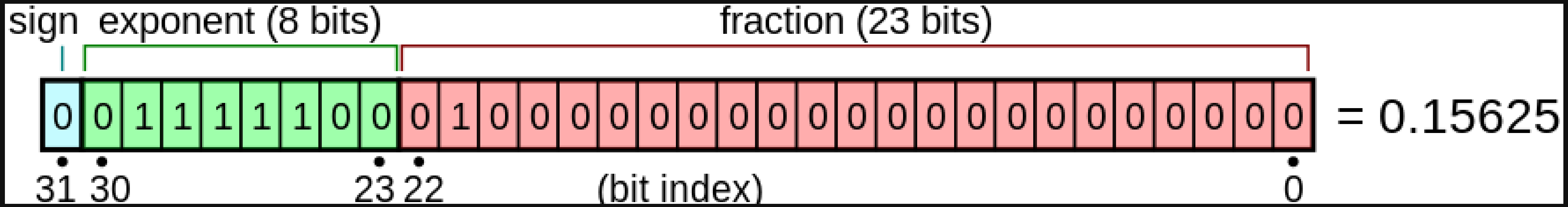
Example : toy system

$$\beta = 2,\ p = 3,\ E \in [-1, 1]$$

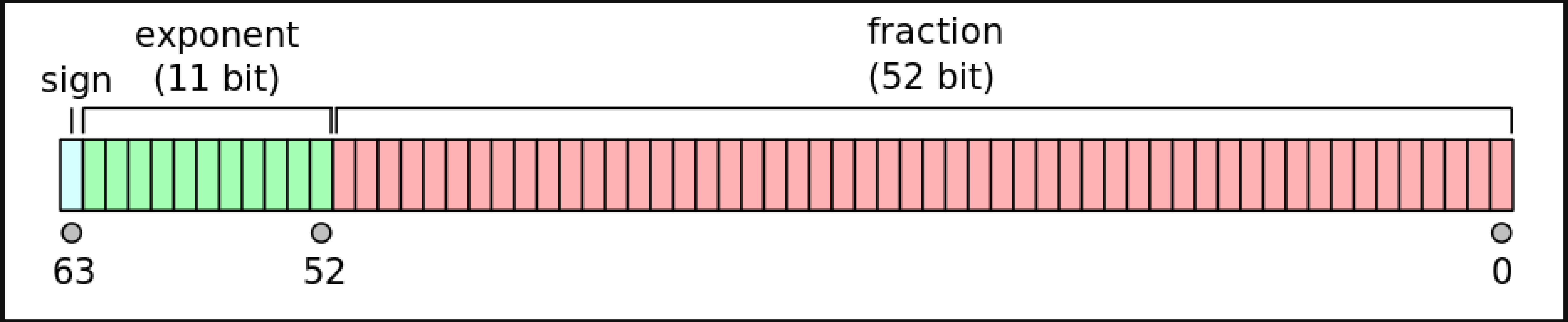


IEEE 754-2008 standard

- 32-bit base-2 format (single precision)



- 64-bit base-2 format (double precision)



Approximation

machine numbers : real number exactly representable in a floating number system

- truncation : 1.751 => 1.7
- rounding : 1.751 => 1.8

Machine Precision

the accuracy of the floating point system

- truncation : $\epsilon_{mach} = \beta^{1-p}$
- rounding : $\epsilon_{mach} = \beta^{1-p}/2$

Real Cases

```
>>> import numpy as np
>>> np.arange(0,1,0.1) == 0.6


array([False, False, False, False, False, False, False, False, False,
       False])
```

Real Cases

```
main()
{
 float x = 16777216.00 ;
 float y = 1.00;
 float z = 5.00;
 printf ("%f\t%f\t%f\n",x,x+y,x+z);
}


16777216.000000    16777216.000000    16777220.000000
```

Acknowledgement

Thanks for Your Attention

There are only 10 types of people in the world. Those who understand binary and those who don't. ☻