Cairo University
Electronics and Communications Department

# Computer Arithmetic: Introduction to Floating Point Numbers

Hossam A. H. Fahmy



## Integers are not enough!

The *dynamic range* is the ratio of the largest magnitude to the smallest non-zero magnitude representable.

**Example 1** With four decimal digits, the numbers range from

$$9999 \longrightarrow 0000$$

The dynamic range is $9\,999 \approx 10\,000$, independent of the decimal point position.
The dynamic range of $0.9999 \longrightarrow 0.0000$ is also $\approx 10\,000$.

How can we represent both 9999 and 0.0001?



## A strange behavior

What do you expect from this code?

```c
#include<stdio.h>

int main(void)
  {
  float x=+0.0, y=-0.0;

  printf("\n\t\t The amazing results: \n");
  printf("\n 10^30 = %f", 10.0E30);
  printf("\n 10^30-10^30 = 0= %f", 10.0E30-10.0E30);
  printf("\n 10^60 = 10^30*10^30 = %f", 10.0E30*10.0E30);
  printf("\n 10^60 = 10^30*10^30 = %f", (float)(10.0E30*10.0E30))
    ;
  printf("\n 1.0/(+0.0) = %f", 1.0/x);
  printf("\n 1.0/(-0.0) = %f", 1.0/y);
  printf("\n +0.0 + (-0.0)= %f", x+y);
  printf("\n -0.0 - (-0.0)= %f", y-y);
  printf("\n 1.0/(+0.0) + 1/(-0.0)= %f", (1.0/x)+(1.0/y));
  printf("\n\n\t+-*/+-*/+-*/+-*/+-*/+-*/+-*/+-*/+-*/+-*/\n\n");
  }
```



## Simple results!

Here is the output.

```
                The amazing results:

 10^30 = 9999999999999999635896294965248.000000
 10^30-10^30 = 0= 0.000000
 10^60 = 10^30*10^30 = 99999999999999999208421814429548
2124579792562202350734542897152.000000
 10^60 = 10^30*10^30 = inf
 1.0/(+0.0) = inf
 1.0/(-0.0) = -inf
 +0.0 + (-0.0)= 0.000000
 -0.0 - (-0.0)= 0.000000
 1.0/(+0.0) + 1/(-0.0)= nan

        +-*/+-*/+-*/+-*/+-*/+-*/+-*/+-*/+-*/+-*/
```

## Scientific notation

The value of a number in scientific notation has six attributes:

$$\begin{array}{cccccc} \pm & d_0 . d_{-1} d_{-2} \cdots & d_{-t} & \times & \beta^{\pm} & {}^{exp} \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ 1 & 2 & 3 & 4 & 5 & 6 \end{array}$$

The computer representation of floating point numbers is similar.



## Floating point basics

1. The fraction is an unsigned number called the *mantissa*.
2. The sign of the entire number is represented by the most significant bit of the number.
3. The exponent is represented by a *characteristic*, a number equal to the exponent plus some positive bias.

Only mantissas of the form $0.xxx\cdots$ are fractions. When discussing both fraction and other mantissa forms (as in $1.xxx$), people tend to use the more general term *significand*.



## Normalization

$0.9 \times 10^0 = 0.09 \times 10^1 = 9.0 \times 10^{-1}$, which one do you want to represent?

A *normalized* number is represented by:

1. $d_0.d_{-1} \cdots d_{-n} \ \times \beta^{exp}$, with $d_0 \neq 0$,

   or

2. $0.d_{-1}d_{-2} \cdots d_{-n} \ \times \beta^{exp}$, with $d_{-1} \neq 0$.

By definition the number zero is represented by a string of zero bits.

If $\beta = 2$, it is either $1.d_{-1} \cdots$ or $0.1d_{-2} \cdots$. The *MSB* is certainly 1, no need to store it. $\Rightarrow$ *Hidden One*



## Why excess code?

1. Zero is represented by a string of all zeros.
2. Smaller numbers (i.e., with a negative exponent) uniformly approach zero.
3. Simplifies the comparison logic.

If $n_{exp}$ is the number of exponent digits, (usually) bias$= \frac{1}{2}\beta^{n_{exp}}$.

## Range and precision

*Range*: a pair of numbers (smallest, largest) to bound all representable numbers.

*Precision:* is the resolution of the system. Defined as the minimum difference between two mantissa representations. Equal to the value of the least significant bit of the mantissa.

$$\mathbf{max} = M_{\mathbf{max}} \times \beta^{exp_{\mathbf{max}}}$$

$$\mathbf{min} = M_{\mathbf{min}} \times \beta^{exp_{\mathbf{min}}}$$



## Overflow, underflow, and gap

Mapping from the infinite number system to a finite range may result in an unrepresentable exponent (exponent spill):

*Overflow* if $|\text{result}| > \mathbf{max}$ ($\rightarrow \pm\infty$?)

*Underflow* if $|\text{result}| < \mathbf{min}$ ($\rightarrow 0$?)

For $\pm d_0.d_{-1} \cdots d_{-t} \times \beta^{exp}$, the *gap* between two successive normalized numbers is $\beta^{-t}\beta^{exp}$.

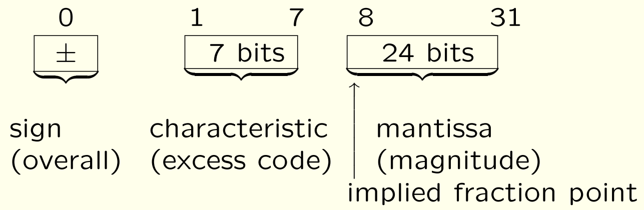
With an increase in the exponent value by one, the gap becomes $\beta$ times larger.

The precision is constant but the gap is a variable.



## Exponent range, significand width, and radix

Assume a 32-bit format:




| | Largest Number | Smallest Number | Precision | Accuracy |
|---|---|---|---|---|
| $\beta = 16$ | $7.2 \times 10^{75}$ | $5.4 \times 10^{-79}$ | $16^{-6}$ | $2^{-21}$ |
| $\beta = 2$ | $9.2 \times 10^{18}$ | $2.7 \times 10^{-20}$ | $2^{-24}$ | $2^{-24}$ |

*Accuracy* is the guaranteed or minimum number of significant mantissa bits excluding any leading zeros. Base 2 provides a better accuracy but less range.



## Representation errors

For a number $x$, $f_x \times \beta^{exp}$ is its exact (normalized) representation. The computer represents $x$ as $f_R \times \beta^{exp}$.

*MRRE* is the maximum error relative to $x$,

$$\begin{aligned} MRRE &= max\left(\frac{|\, f_x\beta^{exp} - f_R\beta^{exp}\,|}{f_x\beta^{exp}}\right) \\ &= max\left(\frac{1/2 \times 2^{-t}}{f_x}\right) \\ &= \frac{1/2 \times 2^{-t}}{1/\beta} = 2^{-t-1}\beta \end{aligned}$$

To have the same (or better) MRRE for $\beta = 2^k$ and $\beta = 2$, the gaps between two successive numbers in the larger base must be less than or equal to the gaps in the binary-base. $\Rightarrow t_k - t_1 \geq k - 1$.

## Shifting speed

**Example 2** For a 24-bit mantissa with all bits zero except the least significant bit, what is the maximum number of shifts required for each case of postnormalization.
**Binary system:** Radix $= 2$ and 23 shifts are required.
**Hexadecimal system:** Radix $= 16$ and 5 shifts are required.

Better accuracy is obtained with small base values and sophisticated round-off algorithms, while computational speed is associated with larger base values and crude round-off procedures such as truncation.



## FP does not always obey the law!

A basic law of algebra is $(A + B = A) \Rightarrow B = 0$.

**Example 3** For a system with $\beta = 2$ and 24 bits in the significand, if $A = 1.0 \times 2^{30}$ and $B = 1.0 \times 2^{-40}$ then $A + B = A$ while $B \neq 0$!

**Example 4** In a decimal system with five digits after the point, check the associativity with $1.12345 \times 10^1 + 1.00000 \times 10^4 - 1.00000 \times 10^4$.
*Solution:* Given only five decimal digits, the result of

$$\begin{aligned} (1.12345 \times 10^1 &+ 1.00000 \times 10^4) - 1.00000 \times 10^4 \\ &= 1.00112 \times 10^4 - 1.00000 \times 10^4 \\ &= 1.12000 \times 10^1. \end{aligned}$$

However, $1.12345 \times 10^1 + (1.00000 \times 10^4 - 1.00000 \times 10^4) = 1.12345 \times 10^1 + 0 = 1.12345 \times 10^1$.
Associativity fails and the first answer lost three digits of significance.



## Yet another loss

**Example 5** With $A = 0.100000 \times 16^1$ and $B = 0.FFFFFF \times 16^0$, what is $A - B$?
*Solution:*

$$\begin{array}{rccccccl} A = & 0.1 & 0 & 0 & 0 & 0 & 0 \times 16^1 & \\ B = & 0.0 & F & F & F & F & F \times 16^1 & \\ \hline A - B = & 0.0 & 0 & 0 & 0 & 0 & 1 \times 16^1 & = 0.1 \times 16^{-4}. \end{array}$$

The real answer is $0.1 \times 16^{-5}$.

Thus, the loss of significance (error) is $0.1 \times 16^{-4} - 0.1 \times 16^{-5} = 0.F \times 16^{-5} = 93.75\%$ of the correct result. Quite a large relative error!

We need to *guard* our digits.

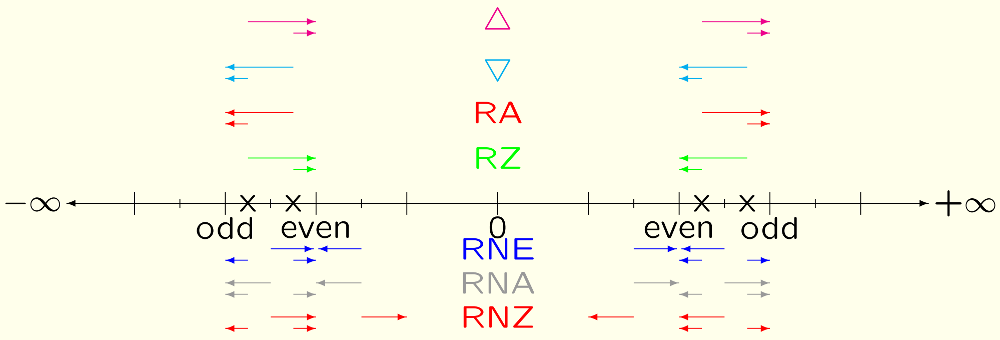


## Rounding

The rounding is a mapping from the real numbers to the machine representable numbers.

| Number | $\bigtriangledown$ | $\bigtriangleup$ | RZ | RA | RNA | RNE |
|---|---|---|---|---|---|---|
| +38.7 | +38 | +39 | +38 | +39 | +39 | +39 |
| +38.5 | +38 | +39 | +38 | +39 | +39 | +38 |
| +38.2 | +38 | +39 | +38 | +39 | +38 | +38 |
| +38.0 | +38 | +38 | +38 | +38 | +38 | +38 |
| −38.0 | −38 | −38 | −38 | −38 | −38 | −38 |
| −38.2 | −39 | −38 | −38 | −39 | −38 | −38 |
| −38.5 | −39 | −38 | −38 | −39 | −39 | −38 |
| −38.7 | −39 | −38 | −38 | −39 | −39 | −39 |

## Real numbers to floating numbers




Note the difference between RNE, RNA, and RNZ in tie cases.



## IEEE standard (binary)

| Sign | Biased exponent | Significand = 1.f (the '1' is hidden) |
|---|---|---|
| $\pm$ | e + bias | f |

| | | |
|---|---|---|
| 32 bits: | 8 bits, bias = 127 | 23 + 1 bits, single-precision or short format |
| 64 bits: | 11 bits, bias = 1023 | 52 + 1 bits, double-precision or long format |
| 128 bits: | 15 bits, bias = 16383 | 112 + 1 bits, quad-precision |

IEEE single (binary32), double (binary64), and quad (binary128) floating point number formats.

Maximum and minimum exponents in the binary IEEE formats:

| Parameter | binary32 | binary64 | binary128 |
|---|---|---|---|
| Exponent width in bits | 8 | 11 | 15 |
| Exponent bias | +127 | +1023 | 16383 |
| $exp_{\mathbf{max}}$ | +127 | +1023 | 16383 |
| $exp_{\mathbf{min}}$ | −126 | −1022 | −16382 |



## Special values

| Exponent bits | Fraction bits | Meaning |
|---|---|---|
| All ones | all zeros | $\pm\infty$ (depending on the sign bit) |
| All ones | non zero | NaN (Not a Number) |
| All zeros | all zeros | $\pm 0$ (depending on the sign bit) |
| All zeros | non zero | subnormal (denormalized) numbers |

The value of a subnormal number in the single format is equal to $(-1)^{sign} \times 2^{-126}(0.f)$.

**Example 6** According to this definition the following bit string

$$\underbrace{\overset{0}{\boxed{0}}}_{\text{sign}}\ \underbrace{\overset{1 \qquad 8}{\boxed{0\ldots\ldots 0}}}_{(\text{bias}=\ 127)}\ \underbrace{\overset{9 \qquad 31}{\boxed{010\ldots\ldots 0}}}_{\text{significand}}$$

is equal to $2^{-126} \times 0.01 = 2^{-128}$.

Those *subnormal* numbers provide the *gradual underflow* property.



## Prior formats

| | IBM S/370 | DEC PDP-11 | CDC Cyber 70 |
|---|---|---|---|
| | S = Short<br>L = Long | S = Short<br>L =Long | |
| Word length | S: 32 bits<br>L: 64 bits | S: 32 bits<br>L: 64 bits | 60 bits |
| Exponent | 7 bits | 8 bits | 11 bits |
| Significand | S: 6 digits<br>L: 14 digits | S: (1)+23 bits<br>L: (1)+55 bits | 48 bits |
| Bias of exponent | 64 | 128 | 1024 |
| Radix | 16 | 2 | 2 |
| Hidden '1' | No | Yes | No |
| Radix point | Left of Fraction | Left of hidden '1' | Right of MSB of Fraction |
| Range of Fraction (F) | $(1/16) \leq F < 1$ | $0.5 \leq F < 1$ | $1 \leq F < 2$ |
| F representation | Signed magnitude | Signed magnitude | One's complement |
| Approximate max. positive number* | $16^{63} \simeq 10^{76}$ | $2^{126} \simeq 10^{38}$ | $2^{1023} \simeq 10^{307}$ |
| Precision | S: $16^{-6} \simeq 10^{-7}$<br>L: $16^{-14} \simeq 10^{-17}$ | S: $2^{-24} \simeq 10^{-7}$<br>L: $2^{-56} \simeq 10^{-17}$ | $2^{-48} \simeq 10^{-14}$ |

Approximate maximum positive number for the DEC PDP-11 is $2^{126}$, as 127 is a reserved exponent.