Chapter 03 Part 02

Real Number Encodin

Real Number Encodin

- Many applications require non-integer numbers.
- There are several ways to represent these numbers. One of them is to use fixed-point notation, which involves using integer arithmetic and simply imagining the binary point somewhere other than to the right of the least significant digit.
- Adding two numbers in this form can be done with an integer adder, while multiplication requires some additional shifts.

Real Number Encodin

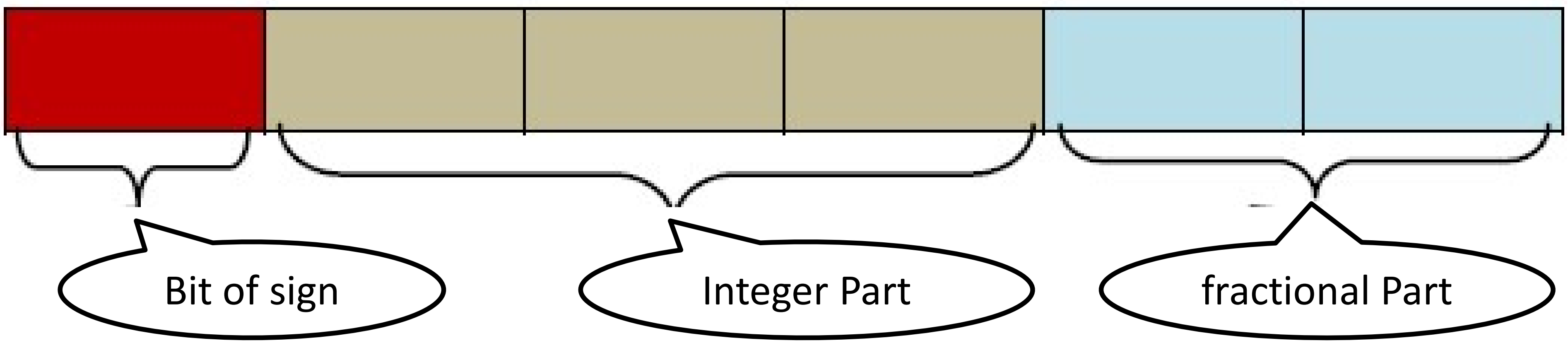
- Furthermore, there is only one non-integer representation that is widely used: the floating-point representation.
- In this system, a machine word is divided into two parts: an exponent and a mantissa.
- In order to perform arithmetic operations correctly on numbers represented by floating-point, it is necessary to normalize them

Real Number Encodin

- Fixed Point
- The representation of the decimal point that separates the integer part from the fractional part in a fractional (real) number poses a problem at the machine level. The first solution adopted was not to physically represent the decimal point and to treat the number as if it were an integer. We will say that the decimal point is fictitious (or virtual); it is managed by the programmer, who defines its position as calculations progress, which is not straightforward, hence its drawback.

Real Number Encodin

- Example: Let's consider a real number represented in 6 bits (in signed binary representation: sign + absolute value), as follows:
- 1 bit for the sign (0 for positive, 1 for negative)
- 3 bits for the integer part
- 2 bits for the fractional part



Real Number Encodin

- The largest representable fractional number on these 6 bits is:

0	1	1	1	1	1

- The largest absolute value of the integer part to be represented is equal to . $(111)_2 = 2^3\text{-}1 = 7$
- The largest absolute value of the fractional part to be represented is equal to . $(0{,}11)_2 = 0{,}75$
- Thus, the largest representable number is equal to +7.75.

Real Number Encodin

- The smallest representable fractional number on these 6 bits is:

0	1	1	1	1	1

- The largest absolute value of the integer part to be represented is equal to . $(111)_2 = 2^3-1 = 7$
- The largest absolute value of the fractional part to be represented is equal to . $(0,11)_2 = 0,75$
- Thus, the smallest representable number is equal to: -7.75.

Real Number Encodin

- The table below provides the representation of <u>some</u> fractional numbers in signed magnitude notation on 6 bits:

Fractional number in sign + absolute value representation			The decimal equivalent of the number
0	111	11	+7,75
0	111	10	+7,50
0	000	10	+0,50
0	000	01	+0,25
0	000	00	+0,00
1	000	00	-0,00
1	000	01	-0,25
1	000	10	-0,50
1	111	10	-7,50
1	111	11	-7,75

Real Number Encodin

- Floating-point
- In the real world, we often deal (problem) with numbers that belong to a very large range.
- The numbers we commonly use are in exponential notation: For example, to represent the number 127845000000, one can use one of the following representations:

- $12{,}7\,845 \,.\, 10^7$.
- $127{,}845 \,.\, 10^6$.
- $0{,}127845 \,.\, 10^9$.
- Etc.

Real Number Encodin

- We can see that exponential representations help avoid carrying many often insignificant digits. This new representation is based on the precision of a mantissa and an exponent: $X = \pm m.10^{e}$
- where 'm' is the mantissa and 'e' is the exponent. In the binary system, the exponential notion is called floating-point notation.
- It is represented as follows: where 'm' is the mantissa and 'e' is the exponent. $X = \pm m.2^{e}$

Real Number Encodin

- 1- Shift:
- To avoid having negative exponents, a shifted exponent is used as a substitute of the simple exponent.
- The value of this **S**hifted **E**xponent (**SE**) is equal to: the value of the **R**eal **E**xponent (**RE**), added to the shift value.
- The shift value must be large enough to shift all exponents with negative values.

Real Number Encodin

- Example:
- Let's assume a shift of 16; in this way, the value 16 is added to the Real Exponent value:

- Let the number be. $X = 23 \times 10^{-7}$

Real Number Encodin

- Example:
- Let's assume a shift of 16; in this way, the value 16 is added to the Real Exponent value:

- Let the number be. $X = 23 \times 10^{-7}$
- Applying the exponent shift, we get $16 + (-7) = 9$

Real Number Encodin

- Example:
- Let's assume a shift of 16; in this way, the value 16 is added to the Real Exponent value:

- Let the number be. $X = 23 \times 10^{-7}$
- Applying the exponent shift, we get $16 + (-7) = 9$
- the new representation of the number X becomes: 23×10^{9}.

Real Number Encodin

- Normalization

A number in scientific notation is said to be normalized if its integer part consists of only one digit.

- **Exemple**
  - The number $(0,715 \times 10^{3})$
  - The number $(7,15 \times 10^{2})$
  - The number $(71,5 \times 10^{1})$
  - The number $(715,0 \times 10^{0})$

Real Number Encodin

2- Normalization

A number in scientific notation is said to be normalized if its integer part consists of only one digit.

- **Exemple**
  - The number ($0,715 \times 10^{3}$) is not normalized.
  - The number ($7,15 \times 10^{2}$)
  - The number ($71,5 \times 10^{1}$)
  - The number ($715,0 \times 10^{0}$)

Real Number Encodin

2- Normalization

A number in scientific notation is said to be normalized if its integer part consists of only one digit.

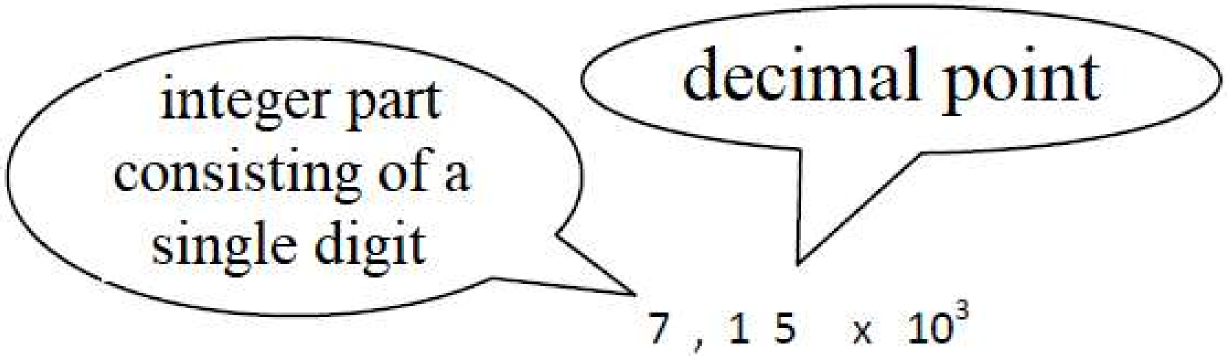
- **Exemple**
  - The number $(0,715 \times 10^{3})$ is not normalized.
  - The number $(7,15 \times 10^{2})$ is normalized.
  - The number $(71,5 \times 10^{1})$
  - The number $(715,0 \times 10^{0})$

Real Number Encodin

2- Normalization

A number in scientific notation is said to be normalized if its integer part consists of only one digit.

- **Exemple**
  - The number $(0{,}715 \times 10^3)$ is not normalized.
  - The number $(7{,}15 \times 10^2)$ is normalized.
  - The number $(71{,}5 \times 10^1)$
  - The number $(715{,}0 \times 10^0)$

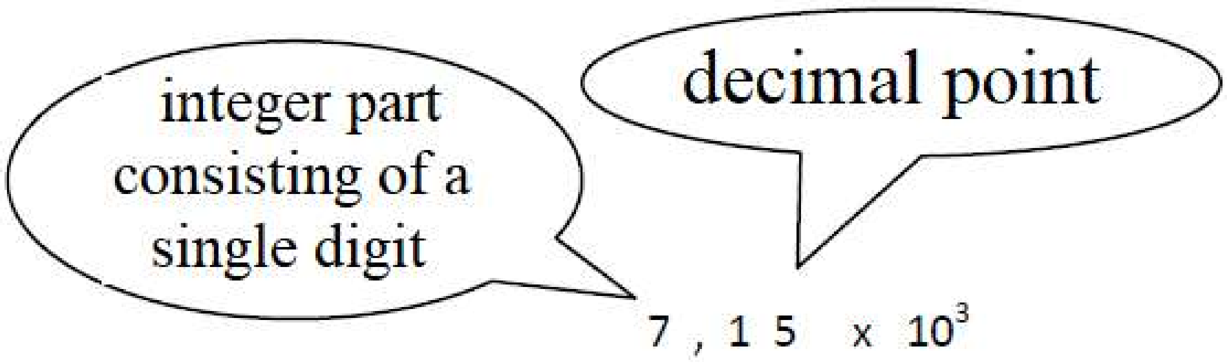


Real Number Encodin

2- Normalization

A number in scientific notation is said to be normalized if its integer part consists of only one digit.

- **Exemple**
  - The number $(0,715 \times 10^3)$ is not normalized.
  - The number $(7,15 \times 10^2)$ is normalized.
  - The number $(71,5 \times 10^1)$ is not normalized.
  - The number $(715,0 \times 10^0)$

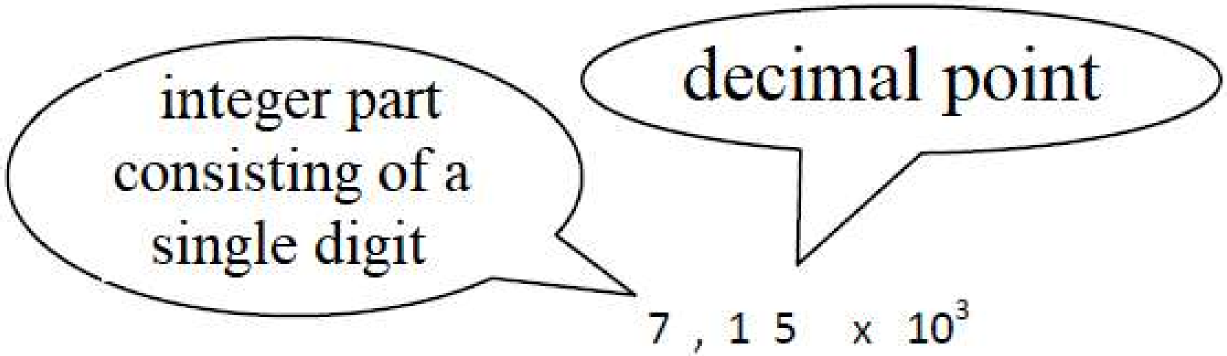


Real Number Encodin

2- Normalization

A number in scientific notation is said to be normalized if its integer part consists of only one digit.

- **Exemple**
  - The number $(0,715 \times 10^3)$ is not normalized.
  - The number $(7,15 \times 10^2)$ is normalized.
  - The number $(71,5 \times 10^1)$ is not normalized.
  - The number $(715,0 \times 10^0)$ is not normalized.



Real Number Encodin

Representation of a Floating-Point Number with Base 2

- Exponentiation

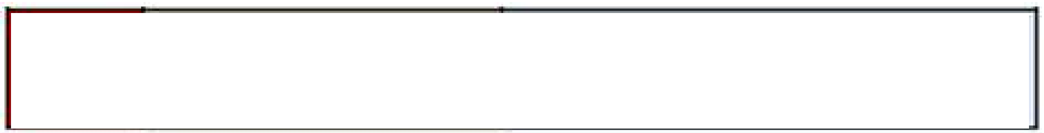
Before the 1980s, various representations of floating-point real numbers were used. After 1985, a standard was adopted by the majority of computer manufacturers, which is the IEEE 754 standard. For this reason, we will only present this representation. In general, a floating-point number is represented in computers as a sequence of bits divided into three zones:

Real Number Encodin

In general, a floating-point number is represented in computers as a sequence of bits divided into three zones:

- ✓ <u>Sign bit</u>
- ✓ <u>Exponent</u>
- ✓ <u>Mantissa</u>

In the following examples, we will adopt the following format for the representation of floating-point numbers:

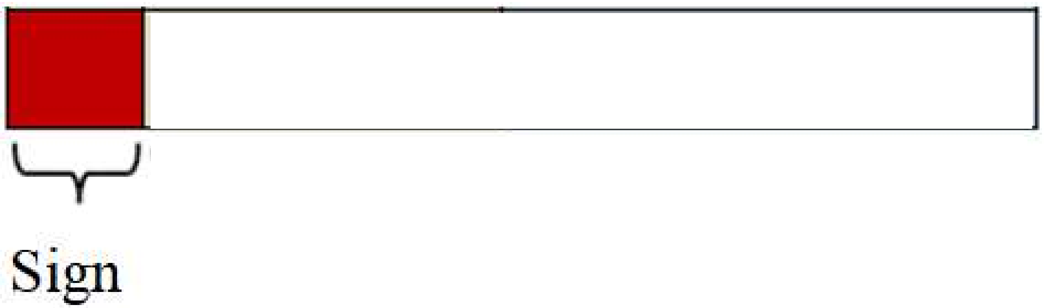


Real Number Encodin

In general, a floating-point number is represented in computers as a sequence of bits divided into three zones:

- ✓ Sign bit
- ✓ Exponent
- ✓ Mantissa

In the following examples, we will adopt the following format for the representation of floating-point numbers:

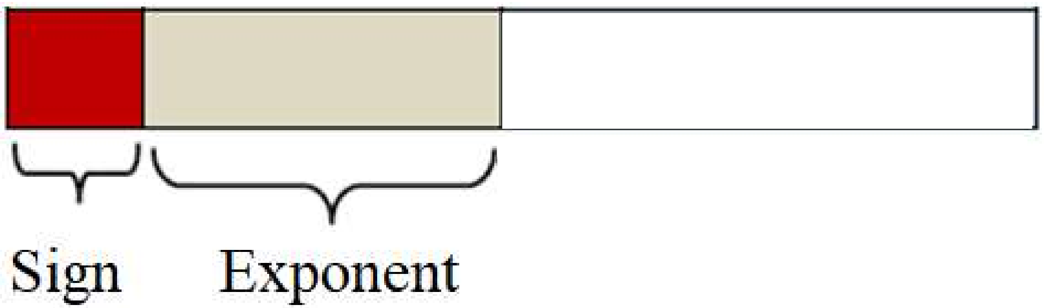


Real Number Encodin

In general, a floating-point number is represented in computers as a sequence of bits divided into three zones:

- ✓ Sign bit
- ✓ Exponent
- ✓ Mantissa

In the following examples, we will adopt the following format for the representation of floating-point numbers:

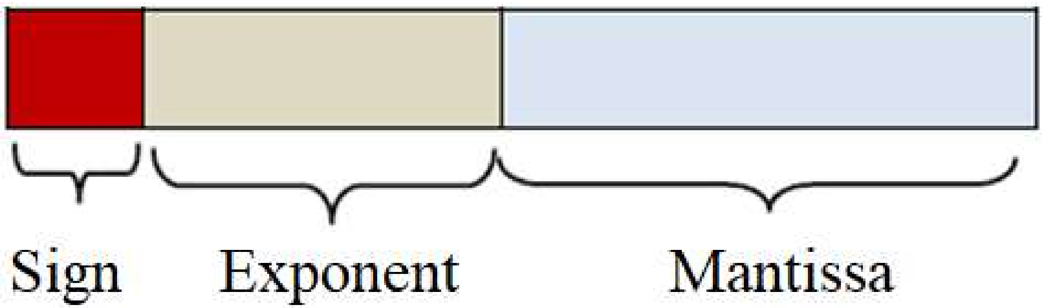


Real Number Encodin

In general, a floating-point number is represented in computers as a sequence of bits divided into three zones:

- ✓ Sign bit
- ✓ Exponent
- ✓ Mantissa

In the following examples, we will adopt the following format for the representation of floating-point numbers:



Real Number Encodin

- The IEEE 754 Representation
- This representation of real numbers is, in fact, an international standard that is widely recognized and used by the majority of computer manufacturers today. This standard defines three formats for floating-point numbers:
  - Single precision on 32 bits
  - Double precision on 64 bits
  - Extended precision on 80 bits

Real Number Encodin

- The IEEE 754 representation
- Single precision on 32 bits

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

Real Number Encodin

- The IEEE 754 representation
- Single precision on 32 bits

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

- Double precision on 64 bits

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	11 bits	52 bits

Real Number Encodin

- The IEEE 754 representation
- Single precision on 32 bits

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

- Double precision on 64 bits

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	11 bits	52 bits

- Extended precision on 80 bits

Mantissa sign	Biased exponent	Normalized mantissa
1 bits	15 bits	64 bits

Real Number Encodin

- Whether in one representation or the other, a number of conventions have been adopted:

A/- The representation order is as follows: first (1) the sign, then (2) the exponent, and finally (3) the mantissa.

B/- Exponents are balance by a shift value to avoid using two's complement representation for negative exponents.

C/- Normalization of numbers in IEEE 754 ensures that the mantissa starts with: a single digit 1 before the decimal point. This digit is implicit (does not appear in the representation).

Real Number Encodin

- First question:
- why choose the order sign + exponent followed by mantissa?
- Answer:
- In fact, this was adopted to facilitate the comparison between real numbers. Since mantissas are normalized (always start with an implied bit of 1), comparing two numbers directly comes down to comparing the exponents.

Real Number Encodin

- Second question:

Why apply a shift to the exponents?

- Answer:

In fact, this answer follows the previous one, that is, when comparing two numbers by comparing the exponents, and if we use the two's complement representation, numbers with negative exponents will appear larger than numbers with positive exponents (unless additional complementation operations are performed, etc.).

Real Number Encodin

- Third question:

Why insist that numbers always have a 1 as the only implied digit before the decimal point?

- Answer:

In fact, this is to save one bit in the representation. For example, in the IEEE 754 single-precision standard, there are 23 bits for the mantissa, whereas in reality, there are 24 (including the implied bit).

Real Number Encodin

- Fourth question:

Given that we insist on the only digit before the decimal point being 1, does this mean we cannot represent the null value 0?

- Answer:

Indeed, this is a concern, but the designers of the IEEE 754 standard addressed it by setting the exponent to zero whenever the value of the number is zero. It is worth noting that the exponent is shifted, meaning that 0 is the smallest value of the exponent (since we should not have a negative shifted exponent).

Real Number Encodin

- The IEEE 754 standard also addresses exceptional cases that may arise from calculations. Indeed, during computations, situations such as division by zero or encountering numbers approaching +∞ or -∞ can occur. For instance, in the IEEE 754 single-precision standard, a value with a mantissa=0 and an exponent=255 indicates that the number is infinite. Additionally, the value 0 is represented by a mantissa of 0 and an exponent of 0. Lastly, when the exponent is zero and the mantissa is non-zero, it indicates that the represented value is not a number.

Real Number Encodin

- Single-precision number format:
  - ✓ 1 bit for the sign
  - ✓ 8 bits for the exponent (the largest value is 127, the smallest is -126, with an offset of 127)
  - ✓ 23 bits for the mantissa
- Double-precision number format:
  - ✓ 1 bit for the sign
  - ✓ 11 bits for the exponent (the largest value is 1023, the smallest is -1022, with an offset of 1023)
  - ✓ 52 bits for the mantissa

Representation in IEEE 754 single-precision

1. Convert the number to binary
2. Write the number as: **$N = (+/-)(1,m)_2 * 2^{ER}$**
3. Calculate the shifted exponent SE = ER + 127, where the exponent is shifted by $2^{8-1} - 1 = 127$; (RE: Real Exponent)

Example: Represent the number $N = (-3.625)_{10}$ in $IEEE_{754}$ single-precision format

Note: The **1** preceding the decimal point is not encoded in the machine (referred to as the hidden bit).

Representation in IEEE 754 single-precision

- To convert a number written in IEEE 754 format:

1. Calculate the Real Exponent RE = SE – 127
2. Calculate the value = sign $(1, \text{mantissa})_2 * 2^{RE}$
   with sign= ±1

- Example: What is the decimal value of the following number represented in IEEE 754?

1	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Representation in IEEE 754 single-precision

- We can find the expression formula for real numbers as follows:

$$(-1)^s \cdot 2^{(E-127)} \cdot 1,M$$

- **Note :**

The **127** of **(E-127)** comes from $2^{\text{The nber of bits in exponent} - 1} - 1$

- If the number is positive, then: ➔ S=0
- If the number is negative, then: ➔ S=1

Representation in IEEE 754 single-precision

- Special Values

Values where all the exponent digits are either 0 or 1 are used to represent particular numbers:

- Exponent = 0 et mantissa = 0 → nomber = 0
- Exponent = 11111111 and mantissa = 0 → nomber ∞
- Exponent = 11111111 and mantissa ≠ 0 → Not a Number (NaN)
- Exponent = 0 et mantissa ≠ 0 → denormalized number

(very small absolute value – we abandon scientific notation here: $value = sign \times mantissa \times 2^{-\text{shift}+1}$

$= sign \times mantissa \times 2^{-126}$)

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows:

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm m.2^{e}$ $N = (-1)^{S} \times 1,M \times 2^{E}$

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm m.2^{e}$ $N=(-1)^{S}\times 1,M\times 2^{E}$

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm m.2^{e}$ $N=(-1)^{S}\times 1,M\times 2^{E}$

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

$$(-42,375)_{10} = (-101010,011)_2.$$

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm m.2^{e}$ $N=(-1)^{S}\times 1,M\times 2^{E}$

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

$$(-42,375)_{10} = (-101010,011)_2.$$

$$(-101010,011)_2 = (-1)^1 \times 1,01010011 \times 2^5.$$

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm m.2^{e}$ $N=(-1)^{S}\times 1,M\times 2^{E}$

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

$$(-42,375)_{10} = (-101010,011)_2.$$

$$(-101010,011)_2 = (-1)^1 \times 1,01010011 \times 2^5.$$

$$SE=E+S = 5 + 127 = (132)_{10} = (10000100)_2$$

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm m.2^e$ $N=(-1)^S \times 1,M \times 2^E$

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

$$(-42,375)_{10} = (-101010,011)_2.$$

$$(-101010,011)_2 = (-1)^1 \times 1,01010011 \times 2^5.$$

$$SE=E+S = 5 + 127 = (132)_{10} = (10000100)_2$$

1	10000100	01010011000000000000000

Representation in IEEE 754 single-precision

- **Exemple 1**:

The representation of the real number (-42.375)10 according to the IEEE 754 standard in single precision will be calculated as follows: $X = \pm\, m.2^e$ $N=(-1)^S \times 1,M \times 2^E$

Mantissa sign	Biased exponent	Normalized mantissa
1 bit	8 bits	23 bits

$$(-42,375)_{10} = (-101010,011)_2.$$

$$(-101010,011)_2 = (-1)^1 \times 1,01010011 \times 2^5.$$

$$SE=E+S = 5 + 127 = (132)_{10} = (10000100)_2$$

1	10000100	01010011000000000000000

Representation in IEEE 754 single-precision

Example 2: The decimal value represented in floating-point by the code:

(C26D0000)$_{\mathbf{IEEE754}}$ will be calculated as follows:

Representation in IEEE 754 single-precision

Example 2: The decimal value represented in floating-point by the code:

(C26D0000)$_{IEEE754}$ will be calculated as follows:

1	10000100	11011010000000000000000

Representation in IEEE 754 single-precision

Example 2: The decimal value represented in floating-point by the code:

$(C26D0000)_{\mathbf{IEEE754}}$ will be calculated as follows:

1	10000100	11011010000000000000000

$$SE = (10000100)_2 = (132)_{10} \Rightarrow E = SE - S = 132 - 127 = (5)_{10}.$$

Representation in IEEE 754 single-precision

Example 2: The decimal value represented in floating-point by the code:

$(C26D0000)_{IEEE754}$ will be calculated as follows:

1	10000100	11011010000000000000000

$SE = (10000100)_2 = (132)_{10} \Rightarrow E = SE - S = 132 - 127 = (5)_{10}.$

$N = (-1)^S \times 1,M \times 2^E = (-1)^1 \times 1,1101101 \times 2^5 = (-111011,01)_2$

Representation in IEEE 754 single-precision

Example 2: The decimal value represented in floating-point by the code:

(C26D0000)**IEEE754** will be calculated as follows:

1	10000100	11011010000000000000000

$SE = (10000100)_2 = (132)_{10} \Rightarrow E = SE - S = 132 - 127 = (5)_{10}$.

$N = (-1)^S \times 1,M \times 2^E = (-1)^1 \times 1,1101101 \times 2^5 = (-111011,01)_2$

$(-111011,01)_2 = (-59,25)_{10}$

Representation in IEEE 754 single-precision

Example 2: The decimal value represented in floating-point by the code:

(C26D0000)**IEEE754** will be calculated as follows:

1	10000100	11011010000000000000000

$$SE = (10000100)_2 = (132)_{10} \Rightarrow E = SE - S = 132 - 127 = (5)_{10}.$$

$$N = (-1)^S \times 1,M \times 2^E = (-1)^1 \times 1,1101101 \times 2^5 = (-111011,01)_2$$

$$(-111011,01)_2 = (-59,25)_{10}$$

- If the number is negative, then: S=1
- If the number is positive, then: S=0

Representation in IEEE 754 single-precision

However, certain conditions must be respected for the exponents:

- The exponent 00000000 is not allowed.
- The exponent 11111111 is not allowed.

However, it is used to signal errors; this configuration is then called NaN (Not a Number).

It is necessary to add 127 to the exponent (in the case of single precision) for a conversion from decimal to a real binary number. Thus, the exponents can range from -126 to 127.

Representation in IEEE 754 single-precision

- **Exemple 3** : Find the IEEE 754 single-precision representation of the number $(35.5)_{10}$

Representation in IEEE 754 single-precision

- **Exemple 3** : Find the IEEE 754 single-precision representation of the number $(35.5)_{10}$
- The nomber is positive, so : S=0
- $(35.5)_{10} = (\ 100011.1)_2$ ……. Fixed point
- $= 1.000111 * 2^5$.......... Floating point (M= 000111)
- Exponent : E-127 = 5 - E= 132 = (10000100)2

0	10000100	00011100000000000000000
S	E	M

Representation in IEEE 754 single-precision

- **Exemple 4:**

Find the IEEE 754 single-precision representation of the number $(-\ 525.5)_{10}$

Representation in IEEE 754 single-precision

- **Exemple 4:**

Find the IEEE 754 single-precision representation

of the number $(-\ 525.5)_{10}$

The nomber is negative, so : S=1

- $(525.5)_{10} = (1000001101.1)_2$ .... Fixed point
- $= 1.0000011011 * 2^9$ .... Floating point
  (M= 000011011)
- Exponent : E-127 = 9 - E= 136 = $(10001000)_2$, donc:

S	E	M
1	10001000	00000110110000000000000

Representation in IEEE 754 single-precision

- **Exemple 5** : Find the IEEE 754 single-precision representation of the number (-0.625)10.

Representation in IEEE 754 single-precision

- **Exemple 5** : Find the IEEE 754 single-precision representation of the number (-0.625)10.
- The nomber is negative, so : S=1
- $(0.625)_{10} = (0.101)_2$ ……. Fixed point
- $= 1.01 * 2^{-1}$ -1 .... Floating point (M= 01)
- Exponent : E-127 = -1 then E= 126 = (1111110)2

1	01111110	01000000000000000000000
S	E	M

Representation in IEEE 754 single-precision

- **Exemple 6** : Find the floating-point number with the following IEEE 754 representation:

0	10000001	11100000000000000000000

Representation in IEEE 754 single-precision

- **Exemple 6** : Find the floating-point number with the following IEEE 754 representation:

0	10000001	11100000000000000000000

- S =0 alors so the number is positive $N=(-1)^S \times 1,M \times 2^E$
- E = $(10000001)_2$ = 129 then E-127 = 129 -127 =2
- 1.M = 1.111
- so 1.111 * 2^2 = $(111,1)_2$ = $(7.5)_{10}$