Ministry of Higher Education and Scientific Research
University of Larbi Ben M'Hidi, Oum El Bouaghi
Faculty of Exact Sciences and Natural and Life Sciences
Department of Mathematics and Computer Science

Computer Structure 1

Presented by: **Dr. NASRI/A**
nasri.ahlem1988@gmail.com
nasri.ahlem@univ-oeb.dz

2023-2024

Chapter 2: Internal Data Representation

1 **Introduction**

- introduce how information is represented in machine (Binary code)

2 **Binary coding**

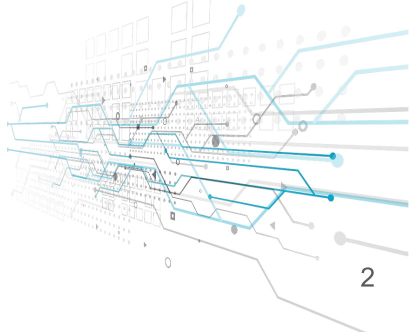
- The reflected binary code
- The DCB code
- The code exceeds three.

3 **Character representation:**

- Code EBCDIC
- Code ASCII
- Code UTF.

4 **Representation of numbers**

- Integer
- Real



Types of information processed by the computer:

Numbers, instructions, images, animated image sequences,sounds, etc, always represented in binary form.

Advantages of binary:

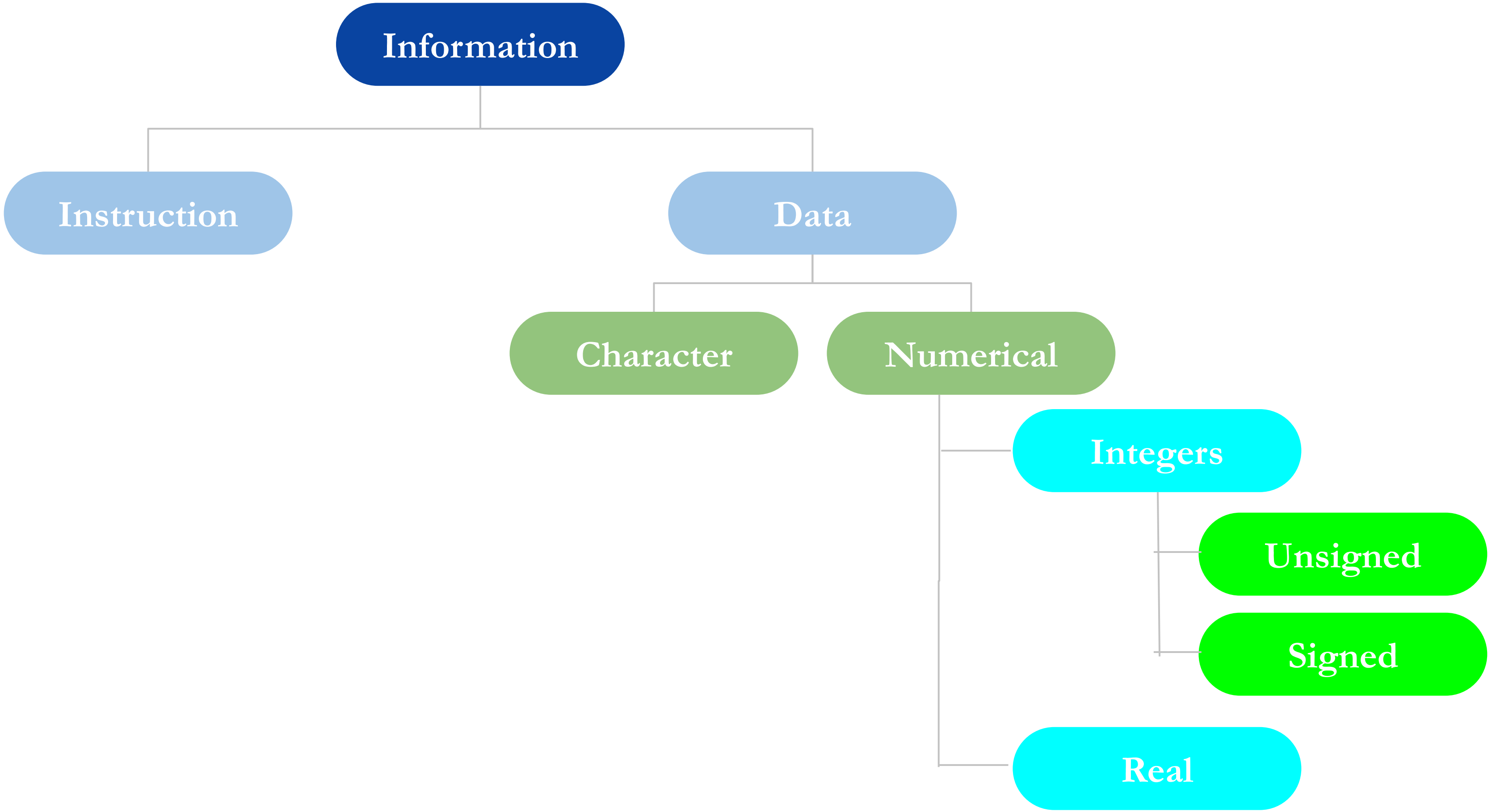
- easy to achieve technically using bistables (systems in two equilibrium states: the existence or not of an electrical voltage +5V=1 and 0V=0).
- fundamental operations that are simple to perform, in the form of logic circuits.

The computer's **memory is limited** => **the bit sequence** which represents the information **is also limited**

Internal Data Representation

⇓

number of bits on which we must represent this data


Information
Instruction
Data
Character
Numerical
Integers
Unsigned
Signed
Real

1. The reflected binary code (GRAY code)

- **Gray code**, also known as **reflected binary code**, is a binary numeral system with a unique property that distinguishes it from traditional binary representation.

- In Gray code, adjacent numbers differ by **only one bit**, which makes it particularly useful in applications where you want to minimize errors during transitions between values (Karnaugh's tables, input output circuit, optical encoders and analog/digital converters, …)

1. The reflected binary code (GRAY code)

Principle:

1. Start with the leftmost (most significant) bit of the Gray code, which is the same as the leftmost bit in the binary representation.
2. Move from left to right through each bit in the Gray code.
3. For each bit, calculate the corresponding bit in the binary representation using the following instructions:
   - If the current bit in the Gray code is 0, keep the previous binary bit as it is.
   - If the current bit in the Gray code is 1, flip the previous binary bit.
4. Continue this process until you have converted all bits from Gray code to binary code.

1. The reflected binary code (GRAY code)

Example: We want to represent the first 8 numbers using the gray code.

Decimal Representation	Binary Representation	a	b	c
0	000	0	0	0
1	001	0	0	1
2	010	0	1	1
3	011	0	1	0
4	100	1	1	0
5	101	1	1	1
6	110	1	0	1
7	111	1	0	0

Mirror level 1

Mirror level 2

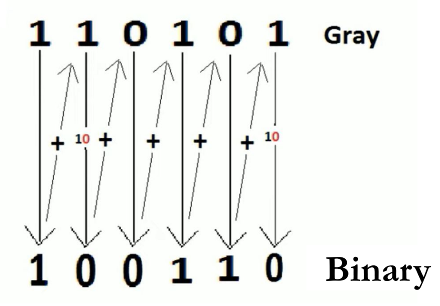
Gray Representation

1. The reflected binary code (GRAY code)

Converting GRAY to Binary

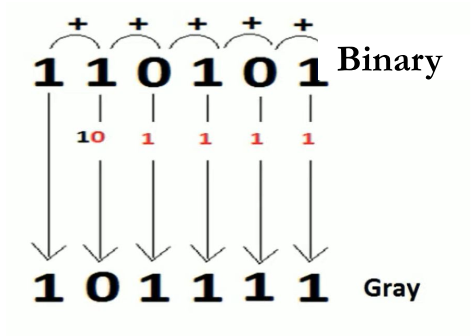
$(110101)_{\text{Gray}} = ?$

$(110101)_{\text{Gray}} = (100110)_2$



Converting Binary to GRAY

$(110101)_2 = ?$ $(110101)_2 = (101111)_{\text{Gray}}$



2. The Decimal Coded Binary code(DCB code)

Principle:

This is the most used code. Its principle is based on associating a binary code with each decimal digit on 4 bits.

Example:

$(512)_{10} = (1000000000)_2 = (0101\ 0001\ 0010)_{DCB}$

nbr of digit in decimal = 10

nbr of numbers we can represented using 4 bits = $2^4 = 16$

$\Downarrow$

There are 6 unused configurations $\longrightarrow$

Decimal	DCB	
0	0000	
1	0001	
2	0010	
3	0011	
4	0100	
5	0101	
6	0110	
7	0111	
8	1000	
9	1001	
/	1010	Not used
/	1011	
/	1100	
/	1101	
/	1110	
/	1111	

3. The code exceeds three

Principale:

Exceeds three code is very similar to the DCB code. Its principle is based on associating each decimal digit with its binary equivalent plus 3.

Example:

$$(512)_{10} = (1000000000)_2 = [(0101+ 0011)\ (0001+0011)\ (0010+0011)]_{\text{Exceeds 3}}$$

$$= (1000\ 0100\ 0101)_{\text{Exceeds 3}}$$

1. Representation of Unsigned Integers:

An unsigned integer is represented by its binary equivalent and the non-significant bits are replaced by zeros.

Example: On 8-bit sequences

8 is represented by 00001000

138 is represented by 10001010

Evaluation:

The evaluation of a number represented in binary is done by converting the representation from the binary system to the decimal system (polynomial expansion).

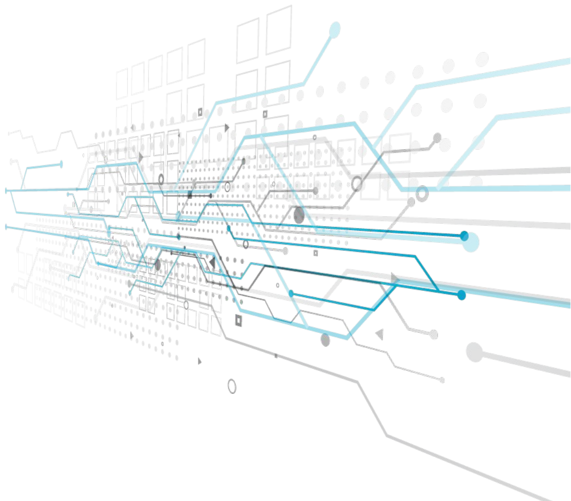
Range of values:

The range of numbers that can be represented in n-bit binary is **$[0, 2^n-1]$**

The problem is how to tell the machine that a number is positive or negative??????

An **unsigned integer** is represented by a machine like its **equivalent in binary** (as seen in the previous section).
On the other hand, to represent **a signed integer**, there are several methods:

1. Signed absolute value
2. One’s Complement
3. Two’s Complement
4. With Excess



2. Representation of Signed Integers:

A- Signed absolute value

A Signed integer is represented on n bits by the binary equivalent of its absolute value on **(n-1)** bits and **the n**th **bit** represents **the sign** of the number.

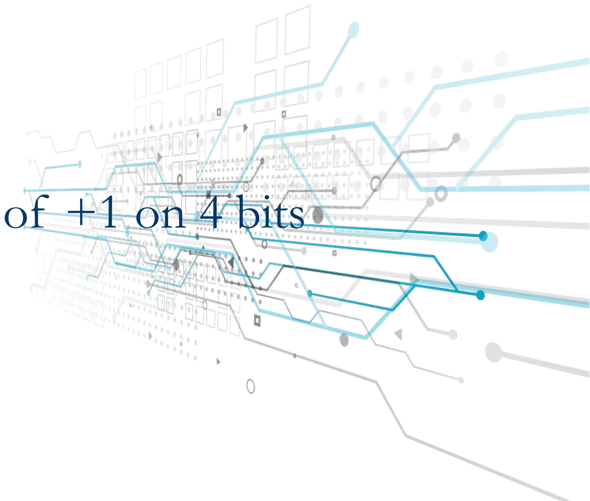
By convention, **the positive sign** is represented by **0** and **the negative sign** is represented by **1**.

Example: On 4 bits $\underbrace{1}_{\text{Sign}}\ \underbrace{\boxed{0001}}_{\text{absolute value}}$

representation of -1 on 4 bits

$0\ \boxed{0001}$

representation of +1 on 4 bits

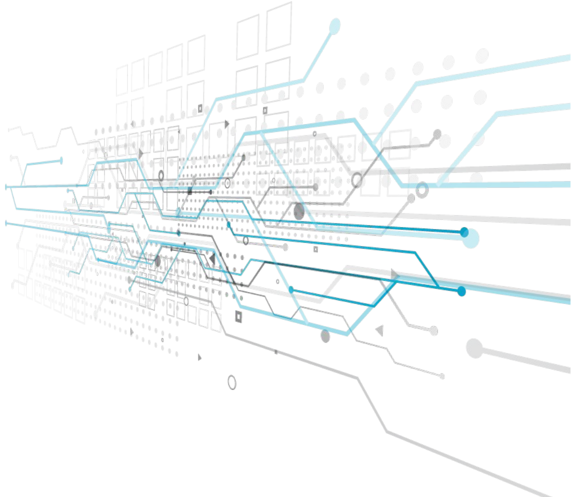


2. Representation of Signed Integers:

A- Signed absolute value

Evaluation: A (a_n, a_{n-1}, a_{n-2}…. $a_2a_1a_0$) is the SAV representation of N in decimal.
The evaluation of N is done according to the following algorithm:

```
If $a_n$=0 then (* the number is positive * )
  N $\longleftarrow$ + Decimal Conversion (A)
  Else (*the number is negative*)
  $a_n \longleftarrow 0$
  N $\longleftarrow$ - Decimal Conversion (A)
End
```



2. Representation of Signed Integers:

A- Signed absolute value

Range of values:

The range of numbers that can be represented in n-bit binary is **$[-(2^{n-1}-1), +2^{n-1}-1]$**

Example: On 3 bits, we have

The range of numbers

$-3 \leq N \leq +3$

$-(4-1) \leq N \leq +(4-1)$

$-(2^2-1) \leq N \leq +(2^2-1)$

$-(2^{(3-1)}-1) \leq N \leq +(2^{(3-1)}-1)$

NB: we have 2 representation of 0 (-0 and +0)!

SAV Value	Binary Value	Decimal Value
000	+ 00	+ 0
001	+ 01	+ 1
010	+ 10	+ 2
011	+ 11	+ 3
100	- 00	- 0
101	- 01	- 1
110	- 10	- 2
111	- 11	- 3

2. Representation of Signed Integers:

B- One’s Complement

- A positive integer is represented by its binary equivalent on (n-1) bits the nth bit represents the sign of the number which is 0.
- A negative integer is represented by its **one's complement.**

We call the one's complement of a number N the number N' such that: $\mathbf{N+N'= 2^n-1}$
where n is the number of bits of the representation of the number N.

Example:

N=1010. Its One’s complement is: $N' = (2^4 - 1)-N$

$N'=(16-1)_{10}-(1010)_2= (15)_{10} - (1010)_2= (1111)_2 – (1010)_2 = (0101)_2= (5)_{10}$

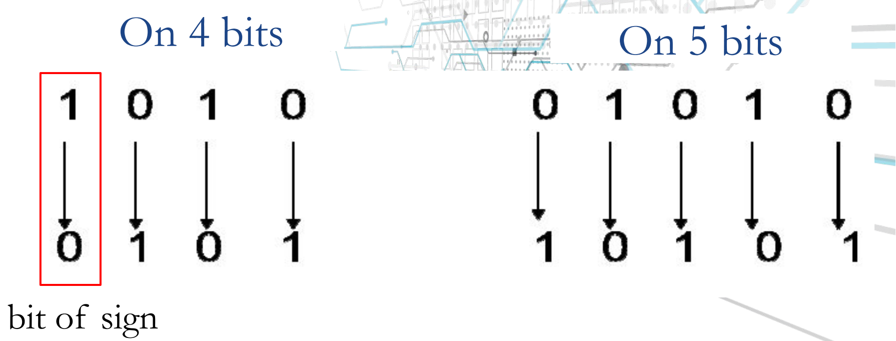
2. Representation of Signed Integers:

B- One's Complement

NB: To find the one's complement of a number, simply

1. First write the absolute value of the number (|N|) on n bits.
2. Then invert all the bits of this number: if the bit is at 1 put 0 in its place and if the bit is at 0 put in its place a 1.

CA1(CA1(N))= N

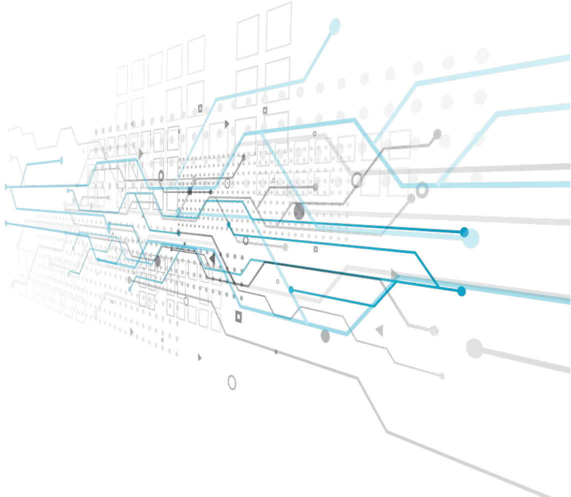


2. Representation of Signed Integers:

B- One's Complement

Evaluation: A (a_n, a_{n-1}, a_{n-2}…. $a_2a_1a_0$) is the 1C representation of decimal N.
The evaluation of N is done according to the following algorithm:

```
If $a_n$=0 then (* the number is positive * )
  N $\leftarrow$ + Decimal Conversion (A)
  Else (*the number is negative*)
 A $\leftarrow$ 1C(A)
  N $\leftarrow$ - Decimal Conversion (A)
End
```



2. Representation of Signed Integers:

B- One’s Complement

Range of values:

The range of numbers that can be represented in n-bit binary is **$[-(2^{n-1}-1), +2^{n-1}-1]$**

Example: On 3 bits, we have

The range of numbers [-3,+3]

$-3 \leq N \leq +3$

$-(4-1) \leq N \leq +(4-1)$

$-(2^2-1) \leq N \leq +(2^2-1)$

$-(2^{(3-1)}-1) \leq N \leq +(2^{(3-1)}-1)$

NB: we have 2 representation of 0 (-0 and +0)!

1C Value	Binary Value	Decimal Value
000	+000	+0
001	+001	+1
010	+010	+2
011	+011	+3
100	-011	-3
101	-010	-2
110	-001	-1
111	-000	-0

2. Representation of Signed Integers:

C- Two’s Complement

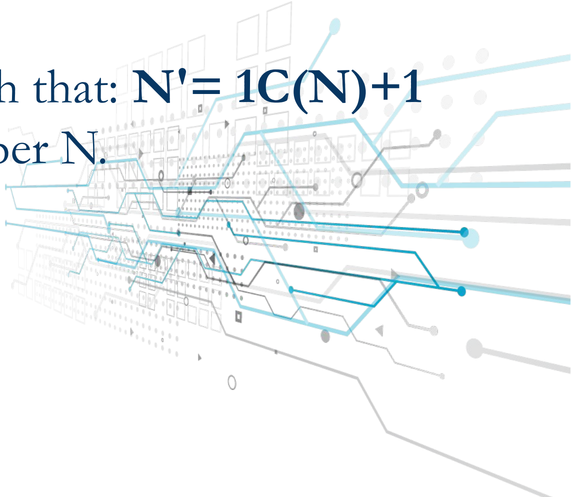
- A positive integer is represented by its binary equivalent on (n-1) bits the nth bit represents the sign of the number which is 0.
- A negative integer is represented by its **tow’s complement.**

We call the one's complement of a number N the number N' such that: **N'= 1C(N)+1** where n is the number of bits of the representation of the number N.

N=1010 on 4 bits

CA1(N)=0101

CA2(N) = **CA1(N)+1** = 0101 + 1 = 0110



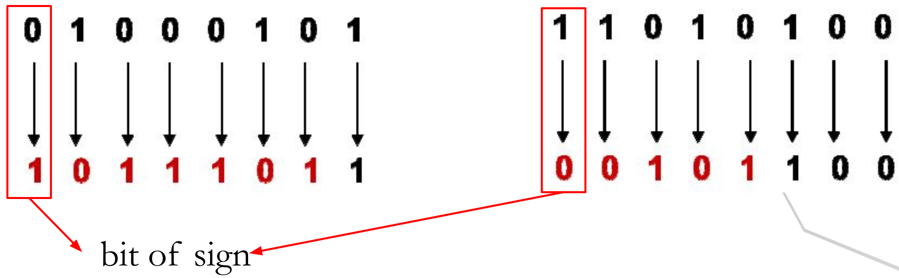
2. Representation of Signed Integers:

C- Two’s Complement

NB: To find the one's complement of a number, simply

1. First write the absolute value of the number (|N|) on n bits.
2. Next, traverse the bits of this number starting from the lowest weight bit to the first 1 and invert the other bits that come after.

CA2(CA2(N))= N

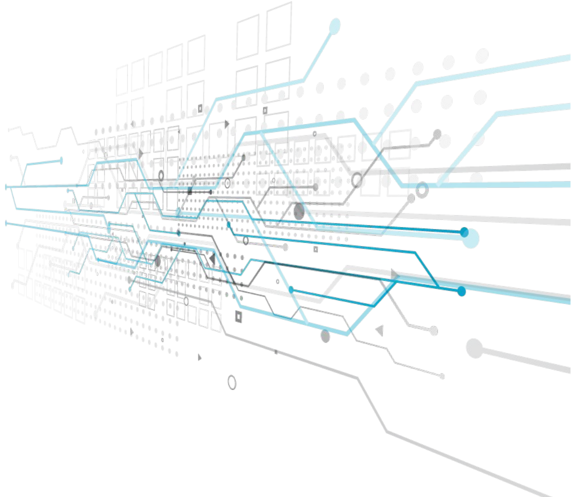


2. Representation of Signed Integers:

C- Two’s Complement

Evaluation: A (a_n, a_{n-1}, a_{n-2}…. $a_2a_1a_0$) is the 2C representation of decimal N.
The evaluation of N is done according to the following algorithm:

```
If $a_n$=0 then (* the number is positive * )
  N $\leftarrow$ + Decimal Conversion (A)
  Else (*the number is negative*)
 A $\leftarrow$ 2C(A)
  N $\leftarrow$ - Decimal Conversion (A)
End
```



Exercise:

Use these different codings to code the following relative integers on 8 bits (if possible):

Decimal	Binary	Gray	DCB	exceeds 3	SAV	C1	C2
5							
-5							
32							
-17							
222							

2. Representation of Signed Integers:

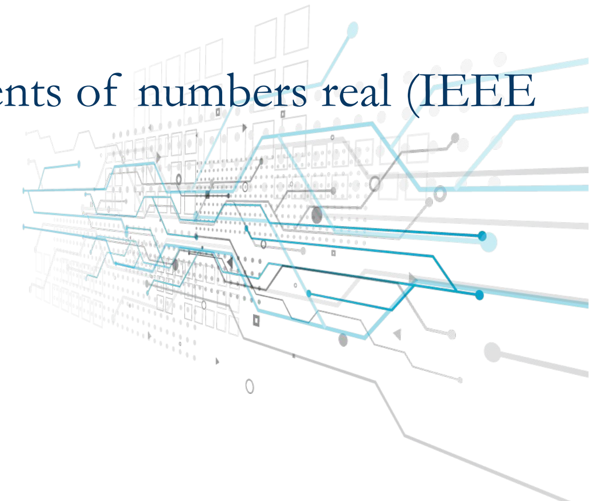
D- Representation with excess

- Every number N is represented by the binary equivalent of the characteristic **C** such that: C **= N + Excess**.
- The excess **E** is chosen so that the sum C is always positive.
- In general, this representation is used to present the exponents of numbers real (IEEE 754 standard) with Excess = 2^{n-1} -1.

Exemple: On 8 bit sequences: E= 2^7-1=127

10 is represented by 10+127= $(137)_{10}$ = $(10001001)_2$

-10 is represented by -10+127= $(117)_{10}$ = $(01110101)_2$



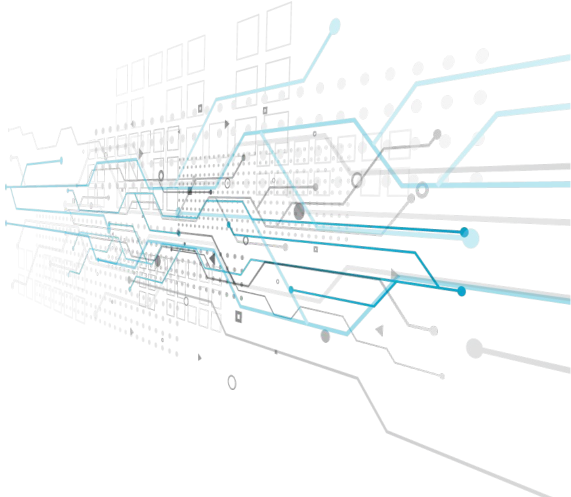
2. Representation of Signed Integers:

D- Representation with excess

Evaluation:

The evaluation of a number represented with excess is done using the following relationship:

$C = N + 2^{n-1} - 1 \leftrightarrow N = 2^{n-1} + 1$

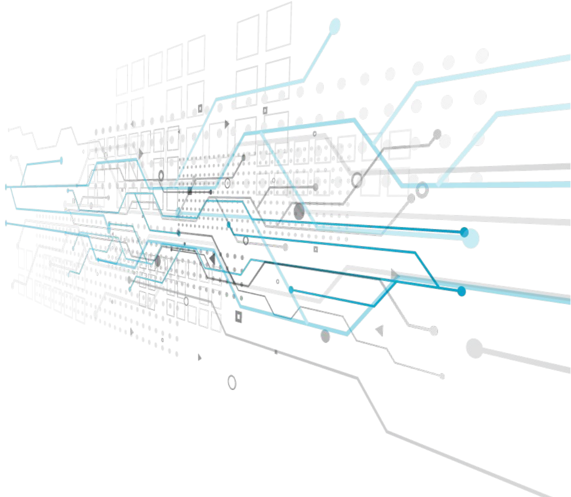


3. Representation Real numbers:

- A real number is made up of two parts: **the integer part** and the **fractional part** (the two parts are separated by a decimal point).
- The problem is how to tell the machine the position of the decimal point.??
- There are two methods for representing real numbers:

  - **Fixed point:** the position of the point is fixed

  - **Floating point:** the position of the point changes (dynamic)



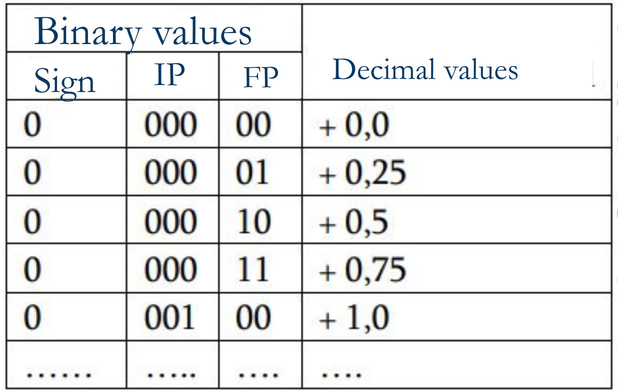
3. Representation Real numbers:

A- Fixed point:

In this representation, the Integer Part (IP) is represented on **e** bits and the Fractional Part (FP) on **f** bits, in addition one bit is used for the sign.

Sign 1 bit	Integer Part (IP) e bits	Fractional Part (FP) f bits

Example: if e=3 and f=2 we will have the following values:



Binary values			Decimal values
Sign	IP	FP	
0	000	00	+ 0,0
0	000	01	+ 0,25
0	000	10	+ 0,5
0	000	11	+ 0,75
0	001	00	+ 1,0
……	…..	….	….

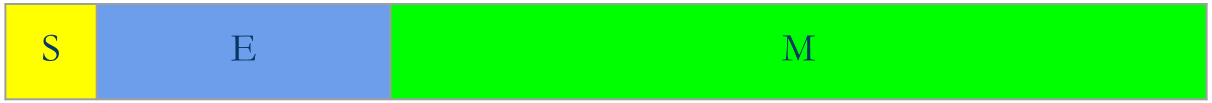
NB: In this representation the values are limited and we do not have a large precision !!

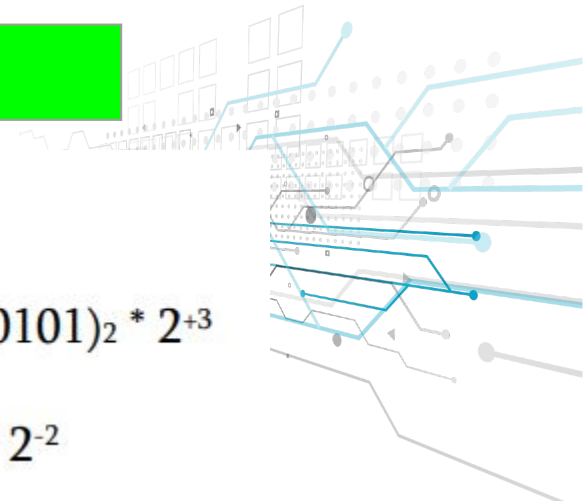
3. Representation Real numbers:

B- Floating point:

Each real number can be written as follows: $(-1)^S * (0,M) * B^E$

Where S: Sign (positive = 0, negative = 1), M: Mantissa, B: Base, E: Exponent.





Example:

$$15.6 = +\,0.156 * 10^{+2} = (-1)^0 * 0.156 * 10^{+2}$$

$$-(110{,}101)_2 = -\,(0{,}110101)_2 * 2^{+3} = (-1)^1 * (0{,}110101)_2 * 2^{+3}$$

$$(0{,}00101)_2 = +\,(0{,}101)_2 * 2^{-2} = (-1)^0 * (0{,}101)_2 * 2^{-2}$$

3. Representation Real numbers:

B- Floating point: Standard IEEE 754 (1985)

In this presentation, the real number can be written as follows:

$$(-1)^S \times (0,M) \times 2^E = (-1)^S \times (1,M_n) \times 2^E$$

S (1 bit)	E (e bits)	M_n (m bits)

Mn: Mantissa normalized to base 2 with a hidden bit. Indeed, the decimal point is placed after the most significant bit at 1,

for example: 11.01 => 1.101 => Mn= 101

- E: Exponent coded with excess 2^{e-1} -1

3. Representation Real numbers:

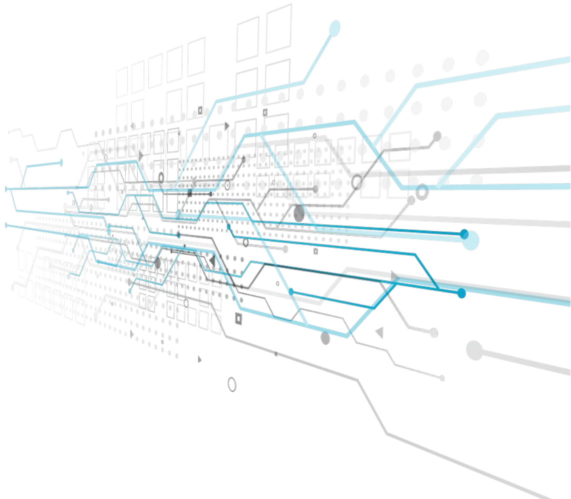
B- Floating point: Standard IEEE 754 (1985)

	32 Bits	64 Bits
Number of Sign bits (S)	1	1
Number of exponent bits (Eb)	8	11
Number of bits of the mantissa (Mn)	23	52
Exponent coding	with Excess $2^7 - 1 = 127$	with Excess $2^{10} - 1 = 1023$

3. Representation Real numbers:

B- Floating point: Standard IEEE 754 (1985)

Example: Code the number N = $(35.5)_{10}$ according to IEEE 754-32:



3. Representation Real numbers:

B- Floating point: Standard IEEE 754 (1985)

Example: Code the number N = $(35.5)_{10}$ according to IEEE 754-32:

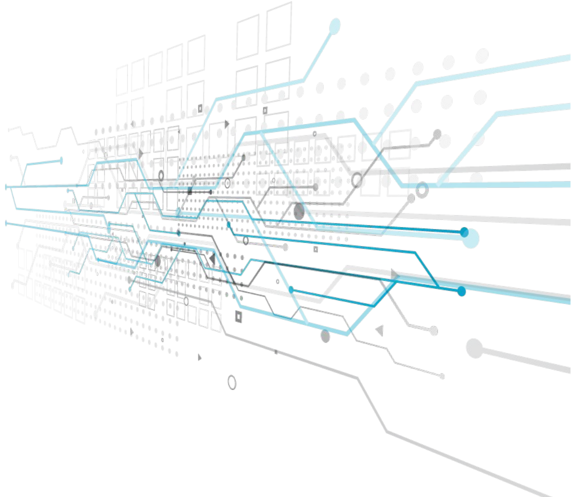
- Converting N to binary: N = 100011.1
- Exponential form of N: N = 1.000111 x 2^5
- Sign bit S = 0
  - Mantissa = 00011100000000000000000
  - Exponent coding: C = E + 127 = 5 + 127 = 132 = $(10000100)_2$
- Overall representation: (0 10000100 00011100000000000000000)

3. Representation of Characters:

The characters include : upper and lower case alphabetical letters (A..Z, a..z), numbers (0..9), punctuation (?,!, ..), special characters(%, @,..) and other symbols (>, ;, ..).

A- ASCII Code : (American Standard Code for Information Interchange)

ASCII code of 'A' is (100 0001)2 = (41)16 = (65)10



ASCII CONTROL CODE CHART

b7 b6 b5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
BITS b4 b3 b2 b1	CONTROL		SYMBOLS NUMBERS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0 0	16 DLE 10 20	32 SP 20 40	48 0 30 60	64 @ 40 100	80 P 50 120	96 ` 60 140	112 p 70 160
0 0 0 1	1 SOH 1 1	17 DC1 11 21	33 ! 21 41	49 1 31 61	65 A 41 101	81 Q 51 121	97 a 61 141	113 q 71 161
0 0 1 0	2 STX 2 2	18 DC2 12 22	34 " 22 42	50 2 32 62	66 B 42 102	82 R 52 122	98 b 62 142	114 r 72 162
0 0 1 1	3 ETX 3 3	19 DC3 13 23	35 # 23 43	51 3 33 63	67 C 43 103	83 S 53 123	99 c 63 143	115 s 73 163
0 1 0 0	4 EOT 4 4	20 DC4 14 24	36 $ 24 44	52 4 34 64	68 D 44 104	84 T 54 124	100 d 64 144	116 t 74 164
0 1 0 1	5 ENQ 5 5	21 NAK 15 25	37 % 25 45	53 5 35 65	69 E 45 105	85 U 55 125	101 e 65 145	117 u 75 165
0 1 1 0	6 ACK 6 6	22 SYN 16 26	38 & 26 46	54 6 36 66	70 F 46 106	86 V 56 126	102 f 66 146	118 v 76 166
0 1 1 1	7 BEL 7 7	23 ETB 17 27	39 ' 27 47	55 7 37 67	71 G 47 107	87 W 57 127	103 g 67 147	119 w 77 167
1 0 0 0	8 BS 8 10	24 CAN 18 30	40 (28 50	56 8 38 70	72 H 48 110	88 X 58 130	104 h 68 150	120 x 78 170
1 0 0 1	9 HT 9 11	25 EM 19 31	41) 29 51	57 9 39 71	73 I 49 111	89 Y 59 131	105 i 69 151	121 y 79 171
1 0 1 0	10 LF A 12	26 SUB 1A 32	42 * 2A 52	58 : 3A 72	74 J 4A 112	90 Z 5A 132	106 j 6A 152	122 z 7A 172
1 0 1 1	11 VT B 13	27 ESC 1B 33	43 + 2B 53	59 ; 3B 73	75 K 4B 113	91 [5B 133	107 k 6B 153	123 { 7B 173
1 1 0 0	12 FF C 14	28 FS 1C 34	44 , 2C 54	60 < 3C 74	76 L 4C 114	92 \ 5C 134	108 l 6C 154	124 | 7C 174
1 1 0 1	13 CR D 15	29 GS 1D 35	45 - 2D 55	61 = 3D 75	77 M 4D 115	93] 5D 135	109 m 6D 155	125 } 7D 175
1 1 1 0	14 SO E 16	30 RS 1E 36	46 . 2E 56	62 > 3E 76	78 N 4E 116	94 ^ 5E 136	110 n 6E 156	126 ~ 7E 176