II- REPRESENTATION OF INTEGERS

II.1 Representation of a natural number

A natural number is a positive or zero integer. The choice to be made (i.e. the number of bits to be used) depends on the range of numbers to be used. To encode natural integers between 0 and 255, we only need 8 bits (one byte) because 2^8=256. In general, n-bit coding can be used to represent natural integers between 0 and 2^{n-1}.

Examples: $9 = (00000101)_2$, $128 = (10000000)_2$

II.2 Representation of a relative integer

A relative integer is an integer that can be negative. It is therefore necessary to code the number in such a way that we can know whether it is a positive number or a negative number.

- **Problem:** How to tell the machine that a number is negative or positive????

There are 3 methods to represent negative numbers:

II.2.1 Sign and Absolute Value (S/VA)

If we work on n bits , then the most significant bit is used to indicate the sign (1: negative sign, 0: positive sign) and the other bits (n -1) designate the absolute value of the number.

- **Example**: If we work on 4 bits
- $(-5)_{10}= (1\ 101)_{2;}\ (+5)_{10} = (\ 0101)_2$

Sign	VA	Values
0	**00**	**+ 0**
0	01	1
0	10	2
0	11	3
1	**00**	**-0**
1	01	-1
1	10	-2
1	11	-3

On n bits, the range of values that can be represented in S/VA:

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

Let's try to calculate 2 + (-2) = 0

	Binaire				Décimal	
	0	0	1	0		2
+	1	0	1	0	+	−2
=	1	1	0	0	=	−4

That's not what we wanted:-4 instead of 0!! **Pros:** It's a fairly simple representation. **Disadvantage:** the zero has two representations +0 and -0

II.2.2 One's complement

We call complement to one (i.e.1) of a number N another number N 'such that: $N+N' = 2^n-1$

n: is the number of bits of the representation of the number N .

Example:

Let N=1010 over 4 bits so its complement to one of N: N'= $(2^4 - 1)$-N

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

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ +\ 0\ 1\ 0\ 1 \\ \hline 1\ 1\ 1\ 1 \end{array}$$

To find the one's complement of a number, simply invert all the bits of this number: If the bit is a 0 put in its place a 1 and if it is a 1 put in its place a 0 .

Examples

$(10010011) = (01101100)_{i.e.1}$

$(11001100) = (00110011)_{i.e.1}$

In addition to one, the most significant bit indicates the sign (0: positive , 1: negative).

The one-to-one complement of the one-to-one complement of a number is equal to the number itself.

CA1(CA1(N))= N

Example:

What is the decimal value represented by the value 101010 in 1 on 6-bit complement?

- The most significant bit indicates that it is a negative number.
- Value = - CA1(101010)

= - $(010101)_2$= - ($21)_{10}$

If we work on 3 bits

One's complement	Binary	Values
000	000	**+ 0**
001	001	+ 1
010	010	+ 2
011	011	+ 3
100	-011	- 3
101	-010	- 2
110	-001	- 1
111	-000	**- 0**

Disadvantage:

In this representation the zero has a double representation. If we work on n bits, the range of values that we can represent in CA1:

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

II.2.3 The two's complement

Positive numbers are encoded in the same way as in pure binary whereas a negative number is encoded by adding the value 1 to its complement at 1. The most significant bit is used to represent the sign of the number.

The two's complement representation is the most used representation for the representation of negative numbers in the machine.

The two's complement of the two's complement of a number is equal to the number itself.

CA2(CA2(N))= N

Two's complement	Binary	Values
000	000	**+ 0**
001	001	+ 1
2010	010	+ 2
011	011	+ 3
100	- 100	- 4
101	- 011	- 3
110	- 010	- 2
111	- 001	- 1

Advantage: We note that the zero does not have a double representation

If we work on n bits, the range of values that we can represent in CA2:

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

Example

It is desired to encode the value −19 on 8 bits. Il suffit:

1. to write 19 in binary: 00010011

2. to write its complement at 1 : 11101100

3. and add 1 : 11101101

The binary representation of −19 over 8 bits is therefore 11101101.

- **Two's complement arithmetic operations**

+9	01001
+4	00100
+13	-1101

(the sign bit = 0 the number is positive)

$(01101)_2 = (13)_{10}$

+9	01001
−4	11100
+5	100101

We notice that the result is on 6 while we work on 5 bits in this case we ignore the sixth bit (called carry) so the result of the addition will be 00101(Positive result)

$(00101)_2 = (5)_{10}$

- 9	10111
-4	11100
-13	110011

We ignore the carry forward the result is 110011

(negative number) Result = - CA2 (10011)= -(01101)

-13

-9	10111
+9	01001
0	100000

$(00000)_2 = (0)_{10}$

- **Retention and overflow**

- ✓ It is said that there is a holdback if an arithmetic operation generates a carryover.
- ✓ We say that there is an overflow if the result of the operation on n bits **and** False.
- ✓ The number of bits used is insufficient to contain the result, i.e. the result exceeds the range of values on the n bits used.

- **Overflow case**

$$\begin{array}{rr} +9 & 01001 \\ \underline{+8} & \underline{01000} \\ 17 & 10001 \end{array}$$

The result is negative whereas it must be positive Overflow!!!

$$\begin{array}{rr} -\ 9 & 10111 \\ -\ \underline{8} & \underline{11000} \\ -17 & 01011 \end{array}$$

The result is positive whereas it must be negative Overflow!!!

We have an overflow if the sum of two positive numbers gives a negative number, or the sum of two negative numbers gives a positive number

There is never an overflow if the two numbers are of different signs.

III. THE REPRESENTATION OF REAL NUMBERS

A real number consists of two parts: the integer part and the fractional part (the two parts are separated by a comma)

Problem: how to tell the machine the position of the comma? There

are two methods for representing real numbers:

III.1 Fixed point numbers

Has an ‘integer‘ part and a ‘decimal‘ part separated by a comma, used by the first machines. The position of the comma is fixed hence the name.

Example: $(11.01)_2$, $(75.23)_8$, $(E7,A4)_{16}$

III.2 Floating point numbers

Floating-point numbers are generally non-integer numbers whose numbers are written only after the decimal point and to which we

adds an exponent to specify how many positions the comma should be moved. For

example, in decimal notation, we will write:

the number 31, 41592 in the form: $0.3141592 * 10^2$

the number -0.01732 in the form: $0.1732 *10^{-1}$

In computer science, a standard has become necessary for the representation of floating points. This is the IEEE 754 standard.

➢ The IEEE 754 standard

IEEE 754 is a standard for the representation of floating-point numbers in binary. It is currently the most used for the calculation of floating-point numbers in the computer field,

In the IEEE 754 standard, a floating point number is always represented by a triplet (S,E,M)

1. The first component S determines the sign of the number represented, this sign being 0 for a positive number, and 1 for a negative number, the sign is represented by a single bit, the most significant bit (the leftmost bit)
2. the second E designates the exponent is coded on 8 bits consecutive to the sign
3. the third M designates the mantissa (the bits located after the comma) on the remaining

23 bits Thus the coding is done in the following form:

S	E	E	E	E	E	E	E	E	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M

IEEE 754 Single Accuracy:

Sign (1 bit)	Exponent (8 bit)	Mantissa (23 bit)

IEEE 754 Double Precision:

Sign (1 bit)	Exhibitor (11 bit)	Mantissa (52 bit)

However, some conditions must be met for exhibitors:

- ✓ Exhibitor 00000000 is prohibited
- ✓ Exhibitor 11111111 is prohibited. However, it is used to report errors, this configuration is called NaN (Not a Number)
- ✓ 127 must be added to the exponent (case of simple precision) for a decimal to a binary real number conversion. Exhibitors can thus range from -256 to 255.

The formula for expressing real numbers is as follows:

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

Note The 127 of the (E-127) comes from $2^{nbitsofthe\ exponent-1} - 1$

Example 1: Find the IEEE 754 single-precision representation of the number

$(35.5)_{10}$ The number is positive S=0

$(35.5)_{10} = (100011.1)_2$……. Fixed-point

$= 1,000111 * 2^5$ Floating point (M= 000111)

Exhibitor: E-127 = 5 E= 132 = $(10000100)_2$

So

0	10000100	00011100000000000000000
S	E	M

Example 2: Find the single-precision IEEE 754 representation of the number

$(-525.5)_{10}$ The number is negative S=1

$(525.5)_{10} = (1000001101.1)_2$ ..... Fixed point

$= 1.0000011011 * 2^9$ Floating point (M= 0000011011)

Exhibitor: E-127 = 9 E= 136 = $(10001000)_2$

So

1	10001000	00000110110000000000000
S	E	M

Example 3: Find the single-precision IEEE 754 representation of the number $(-0.625)_{10}$ The number is negative S=1

$(0.625)_{10} = (0.101)_2$……. Fixed-point

$= 1.01 * 2^{-1}$ Floating point (M= 01)

Exhibitor: E-127 = -1 E= 126 = $(1111110)_2$

So

1	01111110	01000000000000000000000
S	E	M

Example 4: Find the floating number with the following IEEE754 representation:

0	10000001	11100000000000000000000

S =0 Number is positive

E = (10000001)$_2$ = 129 E-127 = 129 -127 =2

1.M = 1.111

$1.111 * 2^2 = (111.1)_2 = (7.5)_{10}$