Negative numbers

At the level of the 80x86 architecture, the numbers are represented in base 2. Being a binary number starting with 1, in the signed interpretation, this number is negative. Its value is given by applying 2's complement of the initial binary configuration.

We have 4 variants in oder to obtain 2's complement.

I. Subtracting the binary contents of the location from 100..00, where the numer of zera's is the same as the number of the location to be complemented).

Ex: 10000 0000 −
    1001 0 011
    ―――――――――
    0110 1101 = 6D = 96+13 = 109 => 2's complement of 147 is 109

- in signed interpretation, the value is −109

II. Reversing the values of bits of the binitial binary number after which we add 1.

1001 0011 ~> 01101100 + 1 = 0110 1101 = 109. =>

=> the value of 1001 0011 in signed interpretation is −109.

III. We left unchanged the bits starting from right until the 1^{st} bit of 1 inclusive ande we reverse the value of the remaining bits.

1001 0011 −> 0110 1101 = 109.

IV. The sum of the absolute values of the 2 complementary values is the cardinal of the set of values representable on that size.

- on 8 bits = 256 values. unsigned: [0, 255], signed: [−128, 127] byte
- on 16 bits = 65536 values. U: [0, 65535], S: [−32768, 32767] wor

on 8 bits, the complement of 1001 0011 = 147 is 256 − 147 = 109. =>
=> S = −109.

- mathematically, the two's complement representation of a negative number is the value $2^n - V$, where V - the absolute value of the represented number.

- there is a range of values: $[0,255] \cap [-128,127] = [0,127]$ that have the same value both in signed and unsiged representations, ~~that~~ and their 2's C cannot be represented as negative numbers on the same sizeof as the initial number representation

x109 is the complement of +147, but x147 is not the complement of 109, instead

- the involvment of '2's complement is unidirectional, making sense only from the S. interpretation and the representation of negative numbers.

Example:

- Addition:

```
mov al, -1
mov bl, -2
add al, bl
```

$-1 + (-2) = -3 \in [-128, 127]$ - the representable domain on a byte

$|-1| = 1 = 0000\ 0001$

$C_2(1) = 1111\ 1111 = FF$

$|-2| = 2 = 0000\ 0010$

$C_2(2) = 1111\ 1110 = FE$

$$\begin{array}{r} 1111\ 1111 + \\ 1111\ 1110 \\ \hline 1111\ 1101 \end{array} = FD = -3.$$

- Subtraction:

```
mov al, 8
mov bl, 16
sub al, bl
```

$8 - 16 = -8 \in [-128, 127)$

-||-

$$\begin{array}{r} 0000\ 1000 - \\ 0001\ 0000 \\ \hline 1111\ 1000 \end{array} \Rightarrow F8 = -8.$$

- multiplication

mov al, -1
mov bl, 100   ; AX = 100 * -1 = -100.
imul bl   bl = byte , al = byte . => bl * al = AX .

imul = signed multiplication.

- division

mov ax, 8
mov bl, -2   ; AL = AX / BL . ; ~~word~~ byte = word / byte .
idiv bl   signed division .

Ex:

mov al, -1
cmp al, 0
jle pos
  ~~st~~ sub al, -1
pos:
sub al, -1

0 is not considered negative in signed integer representatio
thus SF is not set to 1., only ZF.

OR .

"JS"- jump if SF= 1 - is negative
"JNS"- jump if SF = 0 - is positive