```
# 3 BITS RANGE

111 = 7
110 = 6
101 = 5
100 = 4
011 = 3
010 = 2
001 = 1
000 = 0     000 =  0        ( Unsigned )      0  -->  2^n-1
            111 = -1        (  Signed  ) -2^n/2  -->  2^n-1
            110 = -2
            101 = -3
            100 = -4



# OPERATIONS OVER 8 BITS NUMBERS

Op:       0 - 1
$0b00000000 - $0b1

Res:
$0b11111111     N :  255    CF = 1  ( False over Unsigned. )
                Z :   -1    OF = 0  ( True  over   Signed. )

Op:     127 + 1
$0b01111111 + $0b1

Res:
$0b10000000     N :  128    CF = 0  ( True  over Unsigned. )
                Z : -128    OF = 1  ( False over   Signed. )


# Ops summary.

# Multiplication on integers.

    # Carry & Overflow flags set if ah,*d* have some Bits set to one.

    mulb <8  Bits> : %al  * <> ->      %ax

    mulw <16 Bits> : %ax  * <> ->  %dx:%ax

    mull <32 Bits> : %eax * <> -> %edx:%eax

    mulq <64 Bits> : %rax * <> -> %rdx:%rax


# Division on integers

    # RFLAGS Are unaffected by those ops. But if overflow : Generates DE
    #  ( Divided by Zero ) exception.

    divb <8  Bits> :      %al  / <> ->  %al (q) &  %ah (r)

    divw <16 Bits> :  %dx:%ax  / <> ->  %ax (q) &  %dx (r)

    divl <32 Bits> : %edx:%eax / <> -> %eax (q) & %edx (r)

    divq <64 Bits> : %rdx:%rax / <> -> %rax (q) & %rdx (r)


# Signed versions : imul, idiv
```

```
        .data
mystr:      .string "This is a test ! \n"
        .globl main
        .text
main:
    movl    $4,         %eax        # Function code: Write
    movl    $1,         %ebx        # Channel: Stdout
    movl    $mystr,     %ecx        # Source address.
    movl    $18,        %edx        # Number of chars to display.

    int     $0x80   # --> "This is a test ! "
    movl    $4,         %eax        # Has to be set again.
    movb    $51,        2(%ecx)
    int     $0x80   # --> "Th3s is a test ! "
    ret
```