1) LOGICE

op1	op2	and	or	xor
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

not ⇒

op	not op
0	1
1	0

⇒ x or 0 = x
x or 1 = 1
x and 0 = 0
x and 1 = x

and op_1, op_2
or op_1, op_2
xor op_1, op_2
} $op_1 \leftarrow op_1$ operație op_2

op_1 - poate fi registru sau variabilă

op_2 - poate fi registru, variabilă sau constantă

altfel: op_1 - registru ⇒ op_2: reg sau variab sau const

op_1 → variab ⇒ op_2: reg sau constantă

Avantaje: and - zerorizarea unor biți ⇒ izolarea unui bit

or - fixarea / constrângerea ca niște biți = 1

xor - zerorizarea completă a unui octet/cuvânt/dublu cuvânt
complementarea unor biți

not - inversare completă a variabilei biți de un anumit

Exemple:

$b_7 b_6 b_5 0 0 0 0 0$ or
$0 0 0 b_4 b_3 b_2 b_1 b_0$
$b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

$b_7 b_6 1 1 0 0 b_1 b_0$ and
$0 0 1 1 1 0 1 1$
$0 0 1 1 0 0 b_1 b_0$

$b_7 b_6 b_5 a_4 a_3 a_2 b_1 b_0$ xor
$0 0 0 1 1 1 0 0$
$b_7 b_6 b_5 \overline{a_4} \overline{a_3} \overline{a_2} b_1 b_0$

operații de deplasări de biți (shiftări)

- au forma: INSTR dest, nr

⇒ dest: byte, word, doubleW (registru sau variabilă)

nr: poate fi 1 sau reg. cl (val. max. 31)

1) SHL dest, nr

Deplasează spre stanga biții din dest. cu nr poziții iar biții rămași liberi se completează cu 0.

Ultimul bit shiftat (care iese în st) se pastrează în CF

X octet → al $x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$, cl = 2

shl al, cl ⇒ $x_5 x_4 x_3 x_2 x_1 x_0 0 0$, CF = x_6

2) SHR dest, nr - se depl spre dreapta biții din dest cu nr poziții, iar biții rămași liberi se compl. cu 0, cf - ultimul bit shift (care iese în dr).

x ⇒ al, cl = 3, shr al, cl ⇒ $0 0 0 x_7 x_6 x_5 x_4 x_3$, cf = x_2

3) SAL dest, nr = 1) SHL dest, nr.

4) SAR dest, nr - se depl. biții din dest cu nr. poz spre dreapta, iar biții rămași liberi se completează cu bitul de semn. Ultimul bit shif (care iese în dr) se pastrează în cf.

x5 bl, cl = 4, sar al, cl ⇒ $x_7 x_7 x_7 x_7 x_7 x_6 x_5 x_4$, cf = x_3

5) Rotiri de biți

ROL dest, nr - se rotesc spre stanga biții din dest cu nr poziții, iar ultimul bit rotit se adaugă în cf.

al = x = $x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0$, cl = 2, rol al, cl

=> al = $x_5 x_4 x_3 x_2 x_1 x_0 x_7 x_6$, cf = x_6

ROR dest, nr - se rot. spre dr biții din dest cu nr poz. iar în cf se adaugă ultimul bit rotit.

al = x, cl = 3 => $x_2 x_1 x_0 x_7 x_6 x_5 x_3$, cf = x_2

RCL dest, nr - se rotesc biții din dest cu nr poziții spre stanga și după pe poz. libere se adaugă conținutul din CF, iar cf se actualizează cu ultimul bit rotit.

pp cf = k, al = x, cl = 1 => rcl al, cl => $x_6 x_5 x_4 x_3 x_2 x_1 x_0 k$, cf = x_7

cl = 2 => x = $x_5 x_4 x_3 x_2 x_1 x_0 k x_7$, cf = x_6

RCR dest, nr - se rotesc biții din dest cu nr. poziții spre dreapta și după pe poz. libere se adaugă cf, iar în cf se actualiz. ultimul bit rotit.

pp cf = k al = x, cl = 1 rcr al, cl

=> al = $k x_7 x_6 x_5 x_4 x_3 x_2 x_1$, cf = x_0

cl = 2 => al = $k x_0 x_7 x_6 x_5 x_4 x_3 x_2$, cf = x_1

stc - cf = 1
clc - cf = 0

Se da un byte A: $a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0$
să se calc. un nou byte B B = ?

0-2 B = 0-2 A (mod)

3-4 B = 1

5-7 B = 2-4 A

$a_4 a_3 a_2 \,||\, a_2 a_1 a_0 = B$

```
bits 32
global start
extern exit
import exit msvcrt.dll

segment data use32 class=data
    a db 11110101b
    b db 0

segment code use32 class=code
start:
    ; bits 0-2 of B should be equal to bits 0-2 of A
    mov al, [a]             ; AL:=1111 0101b
    and al, 0000 0111b      ; AL:=0000 0101b (we isolate bits 0-2 of AL
                            ; we leave the bits 0-2 of AL unchanged and set the
                            ; other bits to zero
    or [b], al              ; b:=0000 0101b

    ; bits 3-4 of B should be set to 1
    or byte [b], 0001 1000b     ; we set the bits 3 and 4 of B to one and
                                ; leave the other bits unchanged
                                ; b:=0001 1101b

    ; bits 5-7 of B should be equal to bits 2-4 of A
    mov al, [a]             ; AL:=1111 0101b
    shl al, 3               ; shift with 3 position to the left so that bits 2-4
                            ; arrive on positions 5-7
    and al, 1110 0000b      ; AL:=1110 0000b (we isolate bits 5-7 of AL
                            ; we leave the bits 5-7 of AL unchanged and set the
                            ; other bits to zero
    or [b], al              ; b:=1111 1101b

    push dword 0
    call [exit]
```