7. Setul de instrucţiuni 8086 (II)

7.1 Instrucţiuni logice

Aşa cum am văzut instrucţiunile aritmetice privesc operanzii ca şi valori numerice. Instrucţiunile logice îi consideră simple şiruri de biţi. O funcţie logică se va aplica tuturor biţilor sau perechilor de biţi corespunzători. Nu există transport.

Instrucţiunea **NOT (Negare logică bit cu bit)** are ca efect negarea tuturor biţilor operandului destinaţie sau altfel spus calculează complementul faţă de 1 al acestuia. Nu modifică nici un flag.

NOT destinaţie;

Exemplu:

```
MOV AX, 1234h  ; AX=1234h =0001 0010 0011 0100b
NOT AX         ; AX=0EDCBh=1110 1101 1100 1011b
```

Restul operaţiilor au câte doi operanzi.
Instrucţiunea **AND (Şi logic bit cu bit)**

AND destinaţie, sursă; (destinaţie) = (destinaţie) AND (sursă)

Destinaţia poate fi un registru general sau o locaţie de memorie iar sursa un registru general, o locaţie de memorie sau o valoare imediată. Este des utilizată când se doreşte mascarea anumitor biţi.

Instrucţiunile **OR (Sau logic bit cu bit)** şi **XOR (Sau-exclusiv bit cu bit)** au aceiaşi operanzi ca şi instrucţiunea AND.

OR destinaţie, sursă; (destinaţie) = (destinaţie) OR (sursă)

XOR destinaţie, sursă; (destinaţie) = (destinaţie) XOR (sursă)

Bit1	Bit2	OR	XOR	AND
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

Exemplu: Presupunând AX=1234h şi BX=0F0Fh, operaţiile logice AND, OR, şi XOR (realizate independent una de alta) vor furniza

```
MOV AX, 1234h  ; AX= 1234h = 0001 0010 0011 0100b
MOV BX, 0F0Fh  ; BX= 0F0Fh = 0000 1111 0000 1111b
AND AX, BX     ; AX= 0204h = 0000 0010 0000 0100b
OR AX, BX      ; AX= 1F3Fh = 0001 1111 0011 1111b
XOR AX, BX     ; AX= 1D3Bh =0001 1101 0011 1011b
```

Instrucţiunea **TEST** realizează un AND fictiv între destinaţie şi sursă iar flagurile se modifică la fel ca şi la AND.

TEST destinaţie, sursă; (destinaţie) AND (sursă) → FLAGS

Exemplu:

```
mov AX, 1234h  ; AX=1234h=0001 0010 0011 0100b
mov BX, 0F0Fh  ; BX=0EDCBh=1110 1101 1100 1011b
test AX, BX    ; AX AND BX=0000h=0000 0000 0000 0000b ,
               ; AX=1234h, BX=0EDCBh, SF=0, ZF=1
```

7.2 Instrucţiuni pentru deplasări şi rotaţii

Instrucţiunile au structura următoare:

Mnemonica operand, contor

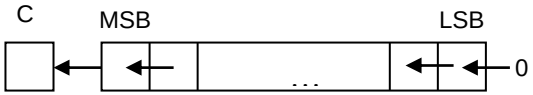
unde *operand* este un registru sau o locaţie de memorie (8 sau 16 biţi) iar *contor* poate fi constanta 1 sau registrul CL:

reg, imed
mem, imed
reg, CL
mem, CL

În cazul operaţiilor de deplasare sunt afectate toate flagurile cu excepţia lui A, iar în cazul unei rotaţii doar C şi O.

Instrucţiunea **SHL/SAL** (**Shift logic/arithmetic left)**

SHL/SAL operand, contor;



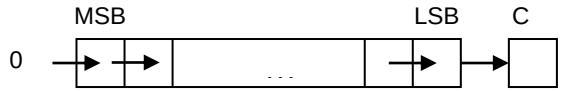
Cel mai semnificativ bit al operandului trece în C iar ceilalti biţi se deplasează la stânga cu o poziţie (înmulţire cu doi). Această operaţie se repetă de un număr de ori egal cu valoarea contorilui.

Exemplu:

```
MOV AX,5555h     ; AX=  5555h  = 0101 0101 0101 0101b
SHL AX,1         ; AX=0AAAAh=1010 1010 1010 1010b, CF=0
```

Instrucţiunea ***SHR*** **(Shift Logic Right)**

SHR operand, contor;



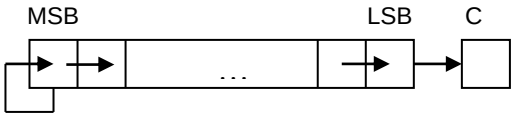
Cel mai puţin semnificativ bit al operandului trece în C iar ceilalţi biţi se deplasează la dreapta cu o poziţie (impărţire cu doi). Aceasta operaţie se repetă de un număr de ori egal cu valoarea din contor.

Exemplu:

```
MOV AX,5555h      ; AX=  5555h = 0101 0101 0101 0101b
SHR AX,1          ; AX=  2AAAh=0010 1010 1010 1010b, CF=1
```

Instrucţiunea **SAR (Shift Arithmetic Right)**—diferenţa faţă de SHR este faptul că semnul se păstrează.

SAR operand, contor;

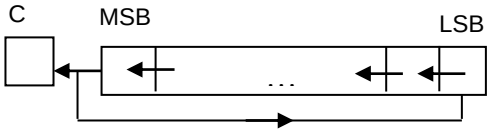


Exemplu:

```
MOV AX,5555h      ; AX=  5555h=0101 0101 0101 0101b
SAR AX,1          ; AX=  2AAAh=0010 1010 1010 1010b, CF=1
```

Instrucţiunea **ROL (Rotate Left)**

ROL operand, contor;



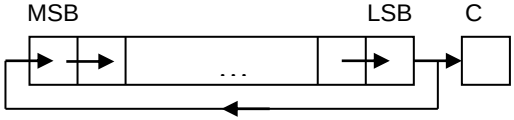
Exemplu:

```
MOV AX,5555h      ; AX=  5555h  =0101 0101 0101 0101b
ROL AX,1          ; AX=0AAAAh=1010 1010 1010 1010b, CF=0
```

Bitul MSB trece atât în C cât şi în bitul LSB din operand.

Instrucţiunea **ROR (Rotate Right)**

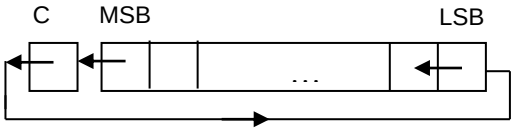
ROR operand, contor;



Exemplu:
```
MOV AX,5555h     ; AX=  5555h  =0101 0101 0101 0101b
ROR AX,1         ; AX=0AAAAh=1010 1010 1010 1010b, CF=1
```

Instrucţiunea **RCL (Rotate Left through Carry)** —C partipă efectiv la rotaţie.

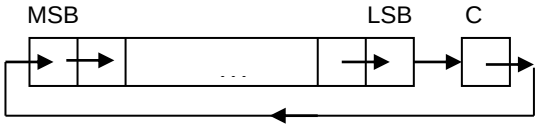
RCL operand, contor;



Exemplu:
```
CLC              ; şterge flagul CF, adică CF=0
MOV AX,5555h     ; AX=  5555h  =0101 0101 0101 0101b
RCL AX,1         ; AX=0AAAAh=1010 1010 1010 1010b, CF=0
```
Exemplu:
```
STC              ; setează flagul CF, adică CF=1
MOV AX,5555h     ; AX=  5555h=0101 0101 0101 0101b
RCL AX,1         ; AX=0AAABh=1010 1010 1010 1011b, CF=0
```

Instrucţiunea **RCR (Rotate Right through Carry)**

RCR operand, contor;



Exemplu:
```
CLC              ; şterge flagul CF, adică CF=0
MOV AX,5555h     ; AX=  5555h  =0101 0101 0101 0101b
RCR AX,1         ; AX=  2AAAh=0010 1010 1010 1010b, CF=1
```

Exemplu:
```
STC              ; setează flagul CF, adică CF=1
MOV AX,5555h     ; AX=  5555h   =0101 0101 0101 0101b
RCR AX,1         ; AX= 0AAAAh=1010 1010 1010 1010b, CF=1
```

7.3. Exerciţii şi teme

1. Analizaţi exemplele de mai jos apoi introduceţi-le pe calculator pentru a verifica corectitudinea rezultatelor.

a.
```
mov al,73h
mov bl,36h
xor al,bl
```

AL	
Sign flag	
Carry flag	
Overflow flag	
Zero flag	

b.
```
mov al,54h
not al
```

AL	
Sign flag	
Carry flag	
Overflow flag	
Zero flag	

c.
```
mov ax,1f54h
mov bx,5a36h
add al,bl
```

AL	
Sign flag	
Carry flag	
Overflow flag	
Zero flag	

d.
```
mov ax,2B54h
mov bx,0236h
mov cl,3
shr ax,1
shl bx,cl
```

AX	
BX	
Carry flag	
Overflow flag	
Zero flag	

e.
```
mov ax,2B54h
mov bx,0236h
mov cl,3
sar bx,cl
```

AX	
BX	
Carry flag	
Overflow flag	
Zero flag	

f.
```
mov ax, 2B54h
mov bx,0236h
mov cl,3
ror ax,cl
rcl bx,cl
```

AX	
BX	
Carry flag	
Zero flag	

g.
```
mov al,54h
mov bl,66h
cmp al,bl
```

AL	
BL	
Carry flag	
Overflow flag	
Zero flag	

h.
```
mov al,32
mov ah,53
mov bx,236
xor ax,bx
```

AX	
BX	
Carry flag	
Zero flag	

2. Pornind de la conţinutul regiştrilor AX=5555h şi CL=4, executaţi următoarele instrucţiuni, specificând valorile regiştrilor şi flagurilor care se modifică:

```
a) SHL AX,CL;          b) shr AX,CL;          c) sar AX,CL;

d) rol AX,CL;          e) ror AX,CL

f) CTC                 g) STC                 h) STC
   rcl AX,CL              rcl AX,CL              rcr AX,CL
```

3. Studiaţi următoarele exemple în pereche şi specificaţi diferenţa dintre ele:

```
a) mov AX,0                        Xor ax,ax

b) MOV  AX, -9                     MOV AX,-9
   MOV  BL, 2                      MOV BL, 2
   IDIV BL                         SAR AX,1
```

4. Comentaţi fiecare secvenţă şi specificaţi valorile regiştrilor şi flagurilor care se modifică:

```
a) Sir DB 2,5,6,8,4,3,75,12
MOV BX, OFFSET SIR
MOV AL,03             ; AL=3
XLAT                  ; extrage al 4-lea element din şir şi
                      ; îl salvează în  AL
SUB AH,AH             ;AH=0
MOV SI,AX             ;SI=AX
MOV AL,[BX+SI]        ;AL=?

b) var db '123456789ABCDEF'     c)sir db 1,2,3,4,5,6,7,8,9,10,11
xor AH, AH                      mov SI,2
lea BX,var                      mov BX,3
mov AL,12                       lea AX, sir[SI][BX]
xlat
```

5. Presupunând că SS=0700h şi SP=FFFEh, se dă urmatoarea secvenţă de instrucţiuni:

```
mov AX,1234h
mov BX, 5678h
push AX
push BX
```

Comentaţi fiecare instrucţiune şi specificaţi cum va arăta zona de memorie unde este alocată stiva (adresele şi conţinutul).

Răspunsuri:

2.

```
a)      shl AX,CL       ; AX=  5550h=0101 0101 0101 0000b, CF=1
b)      shr AX,CL       ; AX=  0555h=0000 0101 0101 0101b, CF=0
c)      sar AX,CL       ; AX=  0555h=0000 0101 0101 0101b, CF=0
d)      rol AX,CL       ; AX=  5555h=0101 0101 0101 0101b, CF=1
e)      ror AX,CL       ; AX=  5555h=0101 0101 0101 0101b, CF=0
f) CF=0 => rcl AX,CL    ; AX=  5552h=0101 0101 0101 0010b, CF=1
g) CF=1 => rcl AX,CL    ; AX=  555Ah=0101 0101 0101 1010b, CF=1
h) CF=1 => rcr AX,CL    ; AX=0B555h=1011 0101 0101 0101b,  CF=0
```

3. a) Cele 2 instrucţiuni sunt identice: ambele depun în registrul AX valoarea 0, dar XOR este mai rapidă.

b) IDIV furnizeaza FCh=-4, SAR furnizează FBh= -5 =>
SAR rotunjeşte numerele negative în jos iar IDIV le rotunjeşte în sus.

5.

```
mov AX,1234h    ; AX=1234h
mov BX, 5678h   ; BX=5678h
push AX         ; SP=0FFFCh, (SS:SP)=1234h
push BX         ; SP=0FFFAh ,(SS:SP)=5678h, deci stiva va arăta
                ;(începând de la adresa fizica 16FFDh, pe octet)
                ; astfel: 12,34,56,78
pop CX          ; CX=5678h, SP=0FFFCh
pop DX          ; DX=1234h, SP=FFFEh
```