Arhitectura Sistemelor de Calcul

Lect. Dr. Șotropa Diana
diana.sotropa@ubbcluj.ro

Facultatea de Matematică și Informatică
Universitatea Babeș-Bolyai

Operații pe biți

Operații logice pe biți

- Limbajul de asamblare dispune de un set de patru instrucţiuni pentru realizarea de operaţii logice la nivel de bit: **AND, OR, XOR** şi **NOT**.

AND d,s	<d> <-- <d> şi <s>	CF,OF,PF,SF,ZF modificaţi; AF - nedefinit
OR d,s	<d> <-- <d> sau <s>	CF,OF,PF,SF,ZF modificaţi; AF - nedefinit
XOR d,s	<d> <-- <d> sau exclusiv <s>	CF,OF,PF,SF,ZF modificaţi; AF - nedefinit
NOT s	<s> <-- <s> negat complement faţă de 1	

Operații logice pe biți

- Pentru doi biţi, unul din sursă, altul din destinaţie, instrucţiunile logice produc noul bit destinaţie având valoarea conform următoarelor reguli:

bitul d	bitul s	d AND s	d OR s	d XOR s
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Operații logice pe biți: AND

AND:

```
AND reg,reg
AND reg,mem
AND reg,imm
AND mem,reg
AND mem,imm
```

Exemplu:

```
mov al, 10101110b
and al, 11110110b
; AL = 10100110b
```

X and 0 = 0
X and 1 = X

bitul d	bitul s	d AND s
0	0	0
0	1	0
1	0	0
1	1	1

OF = CF = 0
SF, ZF, PF – modificate în funcție de rezultat

Operații logice pe biți: OR

OR:

```
OR reg, reg
OR reg, mem
OR reg, imm
OR mem, reg
OR mem, imm
```

Exemplu:

```
mov al,11100011b
or  al,00000100b
; AL = 11100111b
```

X or 0 = X
X or 1 = 1

bitul d	bitul s	d OR s
0	0	0
0	1	1
1	0	1
1	1	1

OF = CF = 0
SF, ZF, PF – modificate în funcție de rezultat

Operații logice pe biți: XOR

XOR:

```
XOR reg, reg
XOR reg, mem
XOR reg, imm
XOR mem, reg
XOR mem, imm
```

bitul d	bitul s	d XOR s
0	0	0
0	1	1
1	0	1
1	1	0

Exemplu:

```
mov al,11100011b
xor al,00100100b
; AL = 11000111b
```

OF = CF = 0
SF, ZF, PF – modificate în funcție de rezultat

Operații logice pe biți: NOT

NOT:

```
NOT reg
NOT mem
```

bitul d	NOT d
0	1
0	0

Exemplu:

```
mov al,11110000b
not al
; AL = 00001111b
```

Flag-urile nu sunt afectate de către instrucțiunea NOT

Operații logice pe biți

- Pentru doi operanzi instrucţiunile logice realizează aceste operaţii asupra perechilor corespunzătoare de biţi. Operanzii sursă şi destinaţie trebuie să aibă ambii aceeaşi dimensiune. Fie:

```
A    DB   10010101b
B    DB   01011010b
```

- Atunci avem

```
mov  al, [A]
and  al, [B]        ;rezultă 00010000b în AL

mov  al, [A]
or   al, [B]        ;rezultă 11011111b în AL

mov  al, [A]
xor  al, [B]        ;rezultă 11001111b în AL
```

Operații logice pe biți

- Instrucţiunea AND este indicată pentru **izolarea** unui anumit bit sau pentru forţarea anumitor biţi la valoarea 0.
- Astfel, dacă dintr-o configuraţie pe 8 biţi

  `a = x x x x x x x x b`

  dorim izolarea bitului i (0 <= i <= 7), vom crea expresia

  `AND a,2^i`

Cum izolăm biții 0,2,3 din octetul a?

Operații logice pe biți

- Instrucţiunea OR poate fi folosită printre altele pentru forţarea anumitor biţi la valoarea 1

Cum forțăm biții 0,2,3 din octetul a să aibă valoarea 1?

Operații logice pe biți

- Instrucţiunea XOR este indicată pentru schimbarea valorii unor biţi din 0 în 1 sau din 1 în 0.

Cum forțăm biții 4-7 din AL să își schimbe valorile în timp ce restul biților rămân neschimbați?

Ce face `XOR AX, AX`?

Operații logice pe biți

- Instrucţiunea **NOT** modifică biţii operandului în valoarea lor complementară (0 în 1 şi 1 în 0)

Ce instrucțiune echivalentă cu `XOR` putem scrie astfel încât efectu asupra operandului să fie același cu NOT?

Operații logice pe biți: TEST

TEST: AND fictiv (nu modifică operandul, dar modifică flag-urile corespunzător)

```
TEST reg,reg
TEST reg,mem
TEST reg,imm
TEST mem,reg
TEST mem,imm
```

Exemplu:

```
mov al, 00100101b
TEST al, 0001001b
; AL - neschimbat 00100101b AND 0001001b = 00000001b
; flag-uri schimbate corespunzător: ZF=0
```

CF = OF = 0
SF, ZF, PF – modificate în funcție de rezultat

Operații logice pe biți: TEST

TEST: AND fictiv (nu modifică operandul, dar modifică flag-urile corespunzător)

```
TEST reg,reg
TEST reg,mem
TEST reg,imm
TEST mem,reg
TEST mem,imm
```

Exemplu:

```
mov al, 00100100b
TEST al, 0001001b
; AL - neschimbat 00100100b AND 0001001b = 00000000b
; flag-uri schimbate corespunzător: ZF=1
```

CF = OF = 0
SF, ZF, PF – modificate în funcție de rezultat

Operații logice pe biți

- Cum putem seta / reseta `ZF` ?
  - `ZF = 1`, în urma operațiilor `TEST <op>, 0` sau `AND <op>, 0`
  - `ZF = 0`, în urma operației `OR <op>, 1`
- Cum putem seta / reseta `SF` ?
  - `SF = 1`, în urma operațiilor `OR <op>, 80h`
  - `SF = 0`, în urma operației `AND <op>, 7Fh`

Deplasări și rotiri de biți

- În cadrul octeţilor sau cuvintelor biţii pot fi deplasaţi aritmetic sau logic sau rotiţi la dreapta sau la stânga cu una sau mai multe poziţii. Numărul de deplasări este fie 1, fie o valoare cuprinsă între 1 şi 255
- Instrucţiunile de *deplasare* de biţi se clasifică în:
  - Instrucţiuni de deplasare logică
    - stânga - **SHL**
    - dreapta - **SHR**
  - Instrucţiuni de deplasare aritmetică
    - stânga - **SAL**
    - dreapta - **SAR**
- Instrucţiunile de *rotire* a biţilor în cadrul unui operand se clasifică în:
  - Instrucţiuni de rotire fără carry
    - stânga - **ROL**
    - dreapta - **ROR**
  - Instrucţiuni de rotire cu carry
    - stânga - **RCL**
    - dreapta - **RCR**

Deplasări și rotiri de biți

- Pentru a defini deplasările şi rotirile considerăm înainte de fiecare linie prezentată mai jos (ca şi configuraţie iniţială un octet `X = abcdefgh`, unde a-h sunt cifre binare, h este cifra binară de rang 0, a este cifra binară de rang 7, iar k este valoarea existentă în CF (CF=k)). Atunci:

```
SHL X,1    ;rezultă  X = bcdefgh0  şi  CF = a
SHR X,1    ;rezultă  X = 0abcdefg  şi  CF = h
SAL X,1    ;identic cu SHL
SAR X,1    ;rezultă  X = aabcdefg  şi  CF = h
ROL X,1    ;rezultă  X = bcdefgha  şi  CF = a
ROR X,1    ;rezultă  X = habcdefg  şi  CF = h
RCL X,1    ;rezultă  X = bcdefghk  şi  CF = a
RCR X,1    ;rezultă  X = kabcdefg  şi  CF = h
```

Important! Se observă că, în toate cazurile, bitul ce părăseşte configuraţia trece în CF.

Deplasările logice pot fi folosite pentru izolarea anumitor biţi în interiorul octeţilor (cuvintelor), iar deplasările aritmetice se pot utiliza pentru înmulţirea sau împărţirea numerelor binare cu puteri ale lui 2.

Deplasări și rotiri de biți

`SHL <op>,1` `SAL <op>,1`	deplasare logică (aritmetică) stânga cu o poziţie. CF conţine bitul cel mai semnificativ deplasat. La dreapta se introduc zerouri.	- Flag-ul AF este întotdeauna nedefinit în urma unei operaţii de deplasare. - Indicatorii PF, SF şi ZF sunt actualizaţi ca şi în cazul instrucţiunilor logice pe biţi. - Indicatorul CF conţine întotdeauna valoarea ultimului bit deplasat din cadrul operandului. - Conţinutul indicatorului OF este întotdeauna nedefinit în cazul unor operaţii de deplasare a mai multor biţi. În cazul în care se deplasează un singur bit, OF este setat la valoarea 1 dacă valoarea bitului cel mai semnificativ (semnul) a fost schimbată de operaţia de deplasare.
`SHL <op>,cl` `SAL <op>,cl`	deplasare logică (aritmetică) stânga cu cl poziţii. CF conţine ultimul bit deplasat. La dreapta se introduc zerouri.	
`SHR <op>,1`	deplasare logică dreapta cu o poziţie. La stânga se introduc zerouri. Bitul cel mai puţin semnificativ este deplasat în CF.	
`SHR <op>,cl`	deplasare logică dreapta cu cl poziţii. La stânga se introduc zerouri. CF va conţine ultimul bit deplasat.	
`SAR <op>,1`	deplasare aritmetică dreapta cu o poziţie. La stânga se extinde bitul de semn. Bitul cel mai puţin semnificativ e deplasat în CF.	
`SAR <op>,cl`	deplasare aritmetică dreapta cu cl poziţii. La stânga se extinde bitul de semn. CF va conţine ultimul bit deplasat.	
`ROL <op>,1`	rotire stânga cu o poziţie. CF va conţine bitul (cel mai semnificativ) rotit.	Instrucţiunile de rotire afectează numai flagurile CF şi OF. CF va conţine întotdeauna valoarea ultimului bit rotit. Pentru operaţiile de rotire cu mai mulţi biţi valoarea flagului OF este nedefinită. La rotirea unui singur bit OF este setat la valoarea 1 dacă bitul cel mai semnificativ îşi schimbă valoarea (schimbare de semn).
`ROL <op>,cl`	rotire stânga cu cl poziţii. CF va conţine ultimul bit rotit.	
`ROR <op>,1`	rotire dreapta cu o poziţie. CF va conţine bitul cel mai puţin semnificativ al lui s.	
`ROR <op>,cl`	rotire dreapta cu cl poziţii. CF va conţine ultimul bit rotit.	
`RCL <op>,1`	rotire stânga cu carry cu o poziţie.	
`RCL <op>,cl`	rotire stânga cu carry cu cl poziţii.	
`RCR <op>,1`	rotire dreapta cu carry cu o poziţie.	
`RCR <op>,cl`	rotire dreapta cu carry cu cl poziţii.	

Deplasări și rotiri de biți: SHL

SHL:

```
SHL reg, imm8
SHL mem, imm8
SHL reg, CL
SHL mem, CL
```

Exemplu:

```
mov al, 10010110b
SHL al, 1 ; AL = 00101100b, CF = 1
SHL al, 2 ; AL = 10110000b, CF = 0
```

Multiplicarea rapidă a operandului cu puteri ale lui 2

Deplasarea spre stânga obţinută ca efect al instrucţiunii SHL este de fapt o înmulţire cu 2 (presupunând desigur că bitul cel mai semnificativ a fost 0, în caz contrar fiind vorba de trunchiere - pierderea celei mai semnificative cifre binare).

```
shl     dx,1            ;DX*2
shl     dx,2            ;DX*4
shl     dx,3            ;DX*8
shl     dx,4            ;DX*16
```

Deplasări și rotiri de biți: SHL

Exemplu:

```
xor     ah,ah
mov     al,[var]
shl     ax,3            ;înmulţire cu opt
add     al,[var]
adc     ah,0            ;al = var * 9
add     al,[var]
adc     ah,0            ;al = var * 10
```

Deplasări și rotiri de biți: SHR

SHR:

```
SHR reg, imm8
SHR mem, imm8
SHR reg, CL
SHR mem, CL
```

Exemplu:

```
mov al, 10010110b
SHR al, 1 ; AL = 01001011b, CF = 0
SHR al, 2 ; AL = 00010010b, CF = 1
```

Exemplu:
SHR poate fi folosit pentru a transfera conținutul HIGH al lui EAX în partea LOW a lui EAX

```
mov eax, 0ABCD1234h
mov cl, 16
shr EAX, cl ; EAX = 0000ABCDh, => AX=ABCDh
```

Deplasări și rotiri de biți: SHR

Împărțirea rapidă a operandului cu puteri ale lui 2

```
mov dl,32
shr dl,1 ; AL=16

mov al,01000000b ; AL = 64
shr al,3 ; împărțire la 8, AL = 00001000b
```

Deplasări și rotiri de biți: SAL

SAL: identic cu SHL

```
SAL reg, imm8
SAL mem, imm8
SAL reg, CL
SAL mem, CL
```

Exemplu:

```
mov al, 10010110b
SAL al, 1 ; AL = 00101100b, CF = 1
SAL al, 2 ; AL = 10110000b, CF = 0
```

Deplasări și rotiri de biți: SAR

SAR:

```
SAR reg, imm8
SAR mem, imm8
SAR reg, CL
SAR mem, CL
```

Exemplu:

```
mov al,0F0h
; AL = 11110000b (-16)
sar al,1
; AL = 11111000b (-8), CF = 0
```

Exemplu:

SAR are ca efect păstrarea semnului operandului. Acest lucru face ca instrucţiunea SAR să fie indicată pentru efectuarea împărţirilor cu semn la puteri ale lui 2.

```
mov al, 10010110b
SAR al, 1 ; AL = 11001011b, CF = 0

mov    bx,-4
sar    bx,1 ; BX = -2
```

Deplasări și rotiri de biți: SAR

Exemplu:
Dacă AX conține un întreg al cărui bit de semn. Dorim să îl extindem în EAX trebuie:

```
mov ax,-128 ; EAX = ????FF80h
shl eax,16 ; EAX = FF800000h
sar eax,16 ; EAX = FFFFFF80h
```

Deplasări și rotiri de biți: ROL

ROL:

```
ROL reg, imm8
ROL mem, imm8
ROL reg, CL
ROL mem, CL
```

Exemplu:

```
mov al,40h ; AL = 01000000b
rol al,1 ; AL = 10000000b, CF = 0
rol al,1 ; AL = 00000001b, CF = 1
rol al,1 ; AL = 00000010b, CF = 0
```

Realinierea biţilor în cadrul unui operand:

```
mov al, 26h
rol al, 4 ; AL = 62h

mov al, 00100000b
mov cl, 3
rol al, cl ; CF = 1, AL = 00000001b
```

Rotirea cifrelor hexa:

```
mov ax, 6A4Bh
rol ax, 4 ; AX = A4B6h
rol ax, 4 ; AX = 4B6Ah
rol ax, 4 ; AX = B6A4h
rol ax, 4 ; AX = 6A4Bh
```

Deplasări și rotiri de biți: ROR

ROR:

```
ROR reg, imm8
ROR mem, imm8
ROR reg, CL
ROR mem, CL
```

Exemplu:

```
mov al, 01h ; AL = 00000001b
ror al, 1 ; AL = 10000000b, CF = 1
ror al, 1 ; AL = 01000000b, CF = 0
```

Realinierea biţilor în cadrul unui operand:

```
mov    si,49f1h
mov    cl,4
ror    si,cl ; si = 149Fh
```

Deplasări și rotiri de biți: RCL

RCL:

```
RCL reg, imm8
RCL mem, imm8
RCL reg, CL
RCL mem, CL
```

Exemplu:

```
clc ; CF = 0
mov bl, 88h;CF = 0, BL = 10001000b
rcl bl,1;CF = 1, BL = 00010000b
rcl bl,1;CF = 0, BL = 00100001b
```

Realizarea de deplasări de biţi implicând operanzi reprezentaţi pe mai multe cuvinte
De exemplu, secvenţa următoare multiplică cu 4 valoarea din DX:AX

```
shl     ax,1        ;bitul 15 din AX este depus în CF
rcl     dx,1        ;valoarea din CF se depune în bitul 0 din DX
shl     ax,1        ;bitul 15 din AX se depune în CF
rcl     dx,1        ;valoarea din CF se depune în bitul 0 din DX
```

Deplasări și rotiri de biți: RCR

RCR:

```
RCR reg, imm8
RCR mem, imm8
RCR reg, CL
RCR mem, CL
```

Exemplu:

```
stc ; CF = 1
mov ah, 10h; AH = 0001 0000,CF = 1
rcr ah, 1 ; AH = 1000 1000, CF = 0
```

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
UNIVERSITATEA BABEȘ-BOLYAI

Str. Mihail Kogălniceanu nr. 1
Cluj-Napoca, Cluj, România

www.cs.ubbcluj.ro