Laborator 4 - Suport teoretic

Operații pe biți

AND

Sintaxă:

```
and <regd>, <regs>; <regd> ← <regd> AND <regs>
and <reg>, <mem>; <reg> ← <reg> AND <mem>
and <mem>, <reg>; <mem> ← <mem> AND <reg>
and <reg>, <con>; <reg> ← <reg> AND <con>
and <mem>, <con>; <mem> ← <mem> AND <con>
```

Semantică:

- Execută operația logică AND asupra operanzilor, punând rezultatul în primul operand.

Exemplu:

```
and EAX, 0Fh; EAX ← EAX AND 0Fh, are loc zerorizarea tuturor biților din EAX
cu excepția ultimilor patru
```

OR

Sintaxă:

```
or <regd>, <regs>; <regd> ← <regd> OR <regs>
or <reg>, <mem>; <reg> ← <reg> OR <mem>
or <mem>, <reg>; <mem> ← <mem> OR <reg>
or <reg>, <con>; <reg> ← <reg> OR <con>
or <mem>, <con>; <mem> ← <mem> OR <con>
```

Semantică:

- Execută operația logică OR asupra operanzilor, punând rezultatul în primul operand.

Exemplu:

```
or EAX, 0Fh; EAX ← EAX OR 0Fh, ultimii 4 biți din EAX vor fi setați la 1,
restul rămânând neschimbați
```

XOR

Sintaxă:

```
xor <regd>, <regs>; <regd> ← <regd> XOR <regs>
```

```
xor <reg>, <mem>; <reg> ← <reg> XOR <mem>
xor <mem>, <reg>; <mem> ← <mem> XOR <reg>
xor <reg>, <con>; <reg> ← <reg> XOR <con>
xor <mem>, <con>; <mem> ← <mem> XOR <con>
```

Semantică:

- Execută operația logică XOR asupra operanzilor, punând rezultatul în primul operand.

Exemplu:

```
xor EDX, EDX; zerorizează conținutul lui EDX
```

TEST

Sintaxă:

```
test <regd>, <regs>; <regd> AND <regs>
test <reg>, <mem>; <reg> AND <mem>
test <mem>, <reg>; <mem> AND <reg>
test <reg>, <con>; <reg> AND <con>
test <mem>, <con>; <mem> AND <con>
```

Semantică:

- Execută operația logică AND asupra operanzilor, fără să pună rezultatul în primul operand.

Exemplu:

```
TEST AL, 01h; se poate testa în acest mod dacă numărul e par sau impar
```

NOT

Sintaxă:

```
not <reg>
not <mem>
```

Semantică:

- Execută operația logică NOT asupra operandului (inversează fiecare bit).

Exemplu:

```
not BYTE PTR [var] ; se inversează fiecare bit din [var]
```

SHL

Sintaxă:

```
shl <reg>, <con8>
shl <mem>, <con8>
shl <reg>, CL
shl <mem>, CL
```

Semantică:

- Biții stocați în destinație se deplasează număr poziții (modulo 32) spre stânga. Bitul (sau biții) cel mai din dreapta se completează cu 0. Ultimul bit care iese în stânga se pastrează în CF.

Exemplu:

```
mov al, 00110011b
mov cl, 2
shl al, cl ; → al = 11001100b, CF = 0
```

SHR

Sintaxă:

```
shr <reg>, <con8>
shr <mem>, <con8>
shr <reg>, CL
shr <mem>, CL
```

Semantică:

- Biții stocați în destinație se deplasează număr poziții (modulo 32) spre dreapta. Bitul (sau biții) cel mai din stânga se completează cu 0. Ultimul bit care iese în dreapta se pastrează în CF.

Exemplu:

```
mov al, 01011110b
mov cl, 2
shr al, cl ; → al = 00010111b, CF = 1
```

SAL

Sintaxă:

```
sal <reg>, <con8>
sal <mem>, <con8>
sal <reg>, CL
sal <mem>, CL
```

Semantică:

- Biții stocați în destinație se deplasează număr poziții (modulo 32) spre stânga. Bitul (sau biții) cel mai din dreapta se completează cu 0. Ultimul bit care iese în stânga se pastrează în CF.

Exemplu:

```
mov al, 00110011b
mov cl, 2
sal al, cl ; → al = 11001100b, CF = 0
```

SAR

Sintaxă:

```
sar <reg>, <con8>
sar <mem>, <con8>
sar <reg>, CL
sar <mem>, CL
```

Semantică:

- Biții stocați în destinație se deplasează număr poziții (modulo 32) spre dreapta. Bitul (sau biții) cel mai din stânga se completează cu bitul de semn (bitul cel mai din stânga înainte de shiftare). Ultimul bit care iese în dreapta se pastrează în CF.

Exemplu:

```
mov al, 11011110b
mov cl, 2
sar al, cl ; → al = 11110111b, CF = 1
```

ROL

Sintaxă:

```
rol <reg>, <con8>
rol <mem>, <con8>
rol <reg>, CL
rol <mem>, CL
```

Semantică:

- Biții stocați în destinație se rotesc număr pozitii (modulo 32) spre stânga. Odată un bit ieșit în stânga el se adaugă automat în partea dreaptă a destinației. Ultimul bit rotit se păstrează în CF.

Exemplu:

```
mov al, 00110011b
mov cl, 2
rol al, cl ; → al = 11001100b, CF = 0
```

ROR

Sintaxă:

```
ror <reg>, <con8>
ror <mem>, <con8>
ror <reg>, CL
ror <mem>, CL
```

Semantică:

- Biții stocați în destinație se rotesc număr pozitii (modulo 32) spre dreapta. Odată un bit ieșit în dreapta el se adaugă automat în partea stângă a destinației. Ultimul bit rotit se pastrează în CF.

Exemplu:

```
mov al, 00111110b
mov cl, 2
ror al, cl ; → al = 10001111b, CF = 1
```

RCL

Sintaxă:

```
rcl <reg>, <con8>
rcl <mem>, <con8>
rcl <reg>, CL
rcl <mem>, CL
```

Semantică:

- Biții stocați în destinație se rotesc număr poziții spre stânga. Odată un bit ieșit în stânga el se păstrează în CF. Valoarea anterioră din CF se adaugă automat în partea dreaptă a destinației.

Exemplu:

```
stc ; CF = 1 (set carry)
mov al, 00110011b
mov cl, 2
rcl al, cl ; → al = 11001110b, CF = 0
```

RCR

Sintaxă:

```
rcr <reg>, <con8>
rcr <mem>, <con8>
rcr <reg>, CL
rcr <mem>, CL
```

Semantică:

- Biții stocați în destinație se rotesc numar poziții spre dreapta. Odatăa un bit ieșit în dreapta el se păstrează în CF. Valoarea anterioară din CF se adaugă automat în partea stângă a destinației.

Exemplu:

```
stc ; CF = 1 (set carry)
mov al, 00110011b
mov cl, 2
rcr al, cl ; → al = 11001100b, CF = 1
```

Laborator 4 – Exemple

Operații pe biți

```
; Se dau cuvintele A si B. Se cere cuvantul C format astfel:
;- bitii 0-2 ai lui C coincid cu bitii 10-12 ai lui B
;- bitii 3-6 ai lui C au valoarea 1
;- bitii 7-10 ai lui C coincid cu bitii 1-4 ai lui A
;- bitii 11-12 ai valoarea 0
;- bitii 13-15 ai lui C concid cu inverul bitilor 9-11 ai lui B

; Vom obtine cuvantul C prin operatii succesive de "izolare". Numim operatia
; de izolare a bitilor 10-12 ai lui B, pastrarea intacta a valorii acestor
; biti, si initializarea cu 0 a celorlalti biti. Operatiunea de izolare se
; realizeaza cu ajutorul operatorului and intre cuvantul B si masca
; 0001110000000000. Odata bitii izolati, printr-o operatie de rotire se
; deplaseaza grupul de biti doriti in pozitia dorita. Cuvantul final se
; obtine prin aplicarea operatorului or intre rezultatele intermediare
; obtinute in urma izolarii si rotirii.
; Observatie: rangul bitilor se numara de la dreapta la stanga

bits 32 ;asamblare si compilare pentru arhitectura de 32 biti
; definim punctul de intrare in programul principal
global  start

extern  exit ; indicam asamblorului ca exit exista, chiar daca noi nu o vom
defini
import  exit msvcrt.dll; exit este o functie care incheie procesul, este
definita in msvcrt.dll
        ; msvcrt.dll contine exit, printf si toate celelalte functii C-runtime
importante
segment  data use32 class=data ; segmentul de date in care se vor defini
variabilele
     a dw 0111011101010111b
     b dw 1001101110111110b
     c dw 0
segment  code use32 class=code ; segmentul de cod
start:

     mov  bx, 0 ; in registrul bx vom calcula rezultatul

     mov  ax, [b] ; izolam bitii 10-12 ai lui b
     and  ax, 0001110000000000b
     mov  cl, 10
     ror  ax, cl ; rotim 10 pozitii spre dreapta
     or   bx, ax ; punem bitii in rezultat

     or   bx, 0000000001111000b ; facem biti 3-6 din rezultat sa aiba valoarea
1

     mov  ax, [a] ; izolam biti 1-4 ai lui a
     and  ax, 0000000000011110b
```

```
    mov  cl, 6
    rol  ax, cl ; rotim 6 pozitii spre stanga
    or   bx, ax ; punem bitii in rezultat

    and  bx, 1110011111111111b ; facem biti 11-12 din rezultat sa aiba
valoarea 0

    mov  ax, [b]
    not  ax ; inversam valoarea lui b
    and  ax, 0000111000000000b ; izolam biti 9-11 ai lui b
    mov  cl, 4
    rol  ax, cl ; deplasam biti 4 pozitii spre stanga
    or   bx, ax ; punem biti in rezultat

    mov  [c], bx ; punem valoarea din registru in variabila rezultat

    push dword 0 ;se pune pe stiva codul de retur al functiei exit
    call [exit] ;apelul functiei sistem exit pentru terminarea executiei
programului
```

Laborator 4 - Probleme propuse

Probleme propuse

1. Se dau cuvintele A si B. Sa se obtina dublucuvantul C:
    - bitii 0-4 ai lui C coincid cu bitii 11-15 ai lui A
    - bitii 5-11 ai lui C au valoarea 1
    - bitii 12-15 ai lui C coincid cu bitii 8-11 ai lui B
    - bitii 16-31 ai lui C coincid cu bitii lui A
2. Se dau cuvintele A si B. Se cere dublucuvantul C:
    - bitii 0-3 ai lui C coincid cu bitii 5-8 ai lui B
    - bitii 4-8 ai lui C coincid cu bitii 0-4 ai lui A
    - bitii 9-15 ai lui C coincid cu bitii 6-12 ai lui A
    - bitii 16-31 ai lui C coincid cu bitii lui B
3. Se dau cuvintele A si B. Sa se obtina dublucuvantul C:
    - bitii 0-2 ai lui C coincid cu bitii 12-14 ai lui A
    - bitii 3-8 ai lui C coincid cu bitii 0-5 ai lui B
    - bitii 9-15 ai lui C coincid cu bitii 3-9 ai lui A
    - bitii 16-31 ai lui C coincid cu bitii lui A
4. Se da octetul A. Sa se obtina numarul intreg n reprezentat de bitii 2-4 ai lui A. Sa se obtina apoi in B octetul rezultat prin rotirea spre dreapta a lui A cu n pozitii. Sa se obtina dublucuvantul C:
    - bitii 8-15 ai lui C sunt 0
    - bitii 16-23 ai lui C coincid cu bitii lui B
    - bitii 24-31 ai lui C coincid cu bitii lui A
    - bitii 0-7 ai lui C sunt 1
5. Se dau octetii A si B. Sa se obtina dublucuvantul C:
    - bitii 16-31 ai lui C sunt 1
    - bitii 0-3 ai lui C coincid cu bitii 3-6 ai lui B
    - bitii 4-7 ai lui C au valoarea 0
    - bitii 8-10 ai lui C au valoarea 110
    - bitii 11-15 ai lui C coincid cu bitii 0-4 ai lui A
6. Se da cuvantul A. Sa se obtina numarul intreg n reprezentat de bitii 0-2 ai lui A. Sa se obtina apoi in B cuvântul rezultat prin rotirea spre dreapta (fara carry) a lui A cu n pozitii. Sa se obtina dublucuvantul C:
    - bitii 8-15 ai lui C sunt 0
    - bitii 16-23 ai lui C coincid cu bitii lui 2-9 ai lui B
    - bitii 24-31 ai lui C coincid cu bitii lui 7-14 ai lui A
    - bitii 0-7 ai lui C sunt 1
7. Se dau doua cuvinte A si B. Sa se obtina dublucuvantul C:
    - bitii 0-4 ai lui C au valoarea 1
    - bitii 5-11 ai lui C coincid cu bitii 0-6 ai lui A
    - bitii 16-31 ai lui C au valoarea 0000000001100101b

- bitii 12-15 ai lui C coincid cu bitii 8-11 ai lui B

8. Se dau doua cuvinte A si B. Sa se obtina un octet C care are:
   - pe bitii 0-5, bitii 5-10 ai cuvantului A
   - pe bitii 6-7 bitii 1-2 ai cuvantului B.

   Sa se obtina dublucuvantul D care are :
   - pe bitii 8-15, bitii lui C
   - pe bitii 0-7, bitii 8-15 din B
   - pe bitii 24-31, bitii 0-7 din A
   - iar pe bitii 16-23, bitii 8-15 din A.
9. Se de cuvantul A si octetul B. Sa se obtina dublucuvantul C astfel:
   - bitii 0-3 ai lui C coincid cu bitii 6-9 ai lui A
   - bitii 4-5 ai lui C au valoarea 1
   - bitii 6-7 ai lui C coincid cu bitii 1-2 ai lui B
   - bitii 8-23 ai lui C coincid cu bitii lui A
   - bitii 24-31 ai lui C coincid cu bitii lui B
10. Sa se inlocuiasca bitii 0-3 ai octetului B cu bitii 8-11 ai cuvantului A.
11. Se dau un octet A si un cuvant B. Sa se obtina un octet C care are pe bitii 0-3 bitii 2-5 ai lui A, iar pe bitii 4-7 bitii 6-9 ai lui B.
12. Se dau doua cuvinte A si B. Sa se obtina dublucuvantul C:
    - bitii 0-6 ai lui C au valoarea 0
    - bitii 7-9 ai lui C coincid cu bitii 0-2 ai lui A
    - bitii 10-15 ai lui C coincid cu bitii 8-13 ai lui B
    - bitii 16-31 ai lui C au valoarea 1
13. Dandu-se 4 octeti, sa se obtina in AX suma numerelor intregi reprezentate de bitii 4-6 ai celor 4 octeti.
14. Se da dublucuvantul A. Sa se obtina numarul intreg n reprezentat de bitii 14-17 ai lui A. Sa se obtina apoi in B dublucuvantul rezultat prin rotirea spre stanga a lui A cu n pozitii.
15. Se dau cuvintele A si B. Se cere dublucuvantul C:
    - bitii 0-2 ai lui C au valoarea 0
    - bitii 3-5 ai lui C au valoarea 1
    - bitii 6-9 ai lui C coincid cu bitii 11-14 ai lui A
    - bitii 10-15 ai lui C coincid cu bitii 1-6 ai lui B
    - bitii 16-31 ai lui C au valoarea 1