Seminar II ASC

Instrucțiuni cu/fără semn. Instrucțiuni aritmetice. Conversii cu/fără semn.

Conținut

2.1. Instrucțiuni cu/fără semn

În cadrul arhitecturii IA-32, în funcție de reprezentarea cu/fără semn a numerelor, există trei clase de instrucțiuni:

1. instrucțiuni care nu țin cont de reprezentarea numerelor: *mov, add, sub*;
2. instrucțiuni care își interpretează operanzii ca fiind numere fără semn: *div, muli*;
3. instrucțiuni care își interpretează operanzii ca fiind numere cu semn: *idiv, imul, cbw, cwd, cwde*.

Este important ca programatorul să fie consistent când dezvoltă un program în limbajul de asamblare IA-32: pentru numerele interpretate fără semn trebuie folosite instrucțiunile din clasa 1 și 2 iar pentru numerele interpretate cu semn să folosească instrucțiuni din clasa 3.
Atunci când se folosesc instrucțiuni cu doi operanzi trebuie să se țină cont de următoarele:

- ambii operanzi trebuie să fie de același tip (ex. se poate aduna un octet cu un octet dar nu un octet cu un cuvânt);
- cel puțin un operand trebuie să fie un registru de uz general sau o constantă, dacă un operand este constantă acesta nu poate să fie operandul destinație.

Legat de observațiile de mai sus, să presupunem că avem următoarea secvență de cod:

Instrucțiunea add [a], [b] este o eroare la compilare (asamblare) și fișierul executab

```
a db 10
b db 11
….
add ax, [a]
add [a], [b]
```

il nu poate fi creat. Aceasta deoarece operanzii instrucțiunii nu respectă a doua constrângere de mai sus ([a] și [b] sunt referințe la memorie / variabile).

În contextul programului de mai sus instrucțiunea add ax, [a] este o eroare logică (compilatorul nu va semnala eroare de sintaxă deoarece dimensiunea operandului sursă este dedusă din dimensiunea operandului destinație). La momentul execuției la cuvântul AX nu se va adăuga octetul a ci cuvântul din memorie care începe la adresa a (cuvântul care începe la adresa a este compus din octeții 10 și 11). Acestea se întâmplă deoarece, chiar dacă variabila a a fost declarată ca un octet folosind directiva *db*, asamblorul NASM nu face legătura între tipul de dată (dimensiune) și locație de memorie în instrucțiune. Declarația „a db 10” doar rezervă un octet la adresa curentă din memorie. Se poate scrie în secțiunea de cod mov ax, [a] și asamblorul va spune CPU să copieze un cuvânt din memorie care începe de la adresa a în registrul AX, instrucțiunea mov eax, [a] va avea ca efect copierea unui dublucuvânt din memorie care începe la adresa a în registrul EAX. [a] semnalează începutul în memorie a datei – nu spune nimic legat de tipul (dimensiunea) datei definite în segmentul de date. Tipul datei este dedus din operandul destinație și prima regulă descrisă mai sus (ambii operanzi trebuie să fie de același tip).
Observație: „[ ]” este operator de adresare, efectul acestuia este explicat de următoarele exemple:

mov eax, [a]	=> copiază în registrul EAX dublucuvântul care începe la adresa variabilei a (copiază în registrul EAX valoarea variabilei a, presupunând că variabila este de tipul dublucuvânt)
mov eax, a	=> copiază în registrul EAX adresa (offset-ul) variabilei (nu valoarea stocată la acea adresă)

2.2. Instrucțiuni pentru înmulțire și împărțire (cu/fără semn)

MUL – înmulțire fără semn
Sintaxa: mul S
(unde S este un registru sau variabilă de tip octet, cuvânt sau dublucuvânt)
Efect: - dacă S este un ***octet*** => AX:=AL * S
- dacă S este un ***cuvânt*** => DX:AX:= AX * S
- dacă S este un ***dublucuvânt*** =>EDX:EAX:= EAX * S

Exemplu: instrucțiunea mul dx stochează rezultatul înmulțirii cuvântului din registrul dx cu cuvântul din registrul ax în doi registrii de 16 biți (rezultatul este un număr reprezentat pe 32 de biți), DX:AX := AX * DX. Rezultatul este stocat în doi regiștrii din motive de compatibilitate cu arhitecturile Intel 8086 precedente. Să presupunem că rezultatul înmulțirii precedente este numărul 12345678h. Cuvântul cel mai puțin semnificativ din rezultat (primii 16 biți mai puțini semnificativi) este stocat în registrul AX (AX := 5678h), cuvântul cel mai semnificativ din rezultat este stocat în registru DX (DX := 1234h).

DIV – împărțire fără semn
Sintaxa: div S
(unde S este un registru sau variabilă de tip octet, cuvânt sau dublucuvânt)
Efect: - dacă *S* este un ***octet*** => AL:=AX / S (câtul) și AH:=AX % S (restul)
- dacă *S* este un ***cuvânt*** => AX:=DX:AX / S (câtul) și DX:=DX:AX % S (restul)
- dacă *S* este un ***dublucuvânt*** =>EAX:= EDX:EAX / S (câtul) și EDX:=EDX:EAX % S (restul)

IMUL și **IDIV** – reprezintă varianta cu semn a instrucțiunilor MUL si DIV (operanzii sunt interpretați ca fiind numere cu semn).

2.2.1 Exemplu

Determinați valoarea expresiei $x = \frac{(a+b)\cdot c}{d}$ unde toate numerele sunt numere cu semn și de tipul octet.

Rezolvare:

```
;
; start program 32 biti
;
bits 32
global start
```

```
extern exit
import exit msvcrt.dll

segment data use32 class=data
   a db 3
   b db 4
   c db 2
   d db 3
   x db 0

segment code use32 class=code
start:
   mov al, [a]             ; AL:= a = 3
   add al, [b]             ; AL:= AL + b = 3 + 4 = 7
   mul byte [c]            ; AX:= AL * c = 7 * 2 = 14
                  ; pentru instructiunea mul s-a specificat tipul operandului [c]
                  ; (byte), pentru ca mul sa poate deduce cu cine sa inmulteasca.
                  ; "c" este doar o referinta, nu are asociat un tip!
   div byte [d]            ; AL:= AX / d = 14 / 3 = 4     AH:= AX % d = 14 % 3 = 2
                  ; similar cu instructiunea mul s-a specificat tipul operandului
                  ; [d] (i.e. byte)
   mov [x], AL             ; x:= AL = 4

   push    dword 0
   call    [exit]
```

2.3 Conversii cu / fără semn

2.3.1 *Exemplu*

Determinați valoarea expresiei $x = \frac{a-b \cdot c}{d}$, unde toate numerele sunt numere fără semn și a, b, c, d sunt octeți.

Rezolvare:

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

segment data use32 class=data
   a db 30
   b db 4
   c db 2
   d db 3
   x db 0

segment code use32 class=code
start:
```

```
mov al, [b]       ; AL:=b = 4
mul byte [c]      ; AX:=AL*c = 4*2 = 8
mov bl, [a]       ; BL:=a=30
; Trebuie sa se scada AX din BL, nu se poate face aceasta operatie direct datorita regulii „ambii
; operanzi trebuie sa fie de acelasi tip/dimensiune”. In astfel de cazuri trebuie convertit tipul mai
; mic la tipul mai mare (ex. se converteste valoarea din BL de la octet la cuvant). BL := 30 = 00011110b.
; Numarul 30 reprezentat fara semn pe 16 biti in baza 2 este: 0000 0000 0001 1110b. Diferenta intre
; reprezentarea fara semn pe 8 biti si 16 biti sunt cei opt biti cei mai semnificativi de valoare 0. Astfel
; conversia fara semn a unui octet/cuvant/dublucuvant (byte/word/dword) se face prin adaugarea de
; biti semnificativi de valoare 0 pana se ajunge la dimensiunea dorita.
mov bh, 0       ; BX:=0000 0000 0001 1110b
                ; conversia fara semn a lui BL in BX
sub bx, ax      ; BX:= BX-AX = 30 – 8 = 22
mov ax, bx      ; AX:=BX=22
div byte [d]    ; AL:=AX / d =22 / 3 = 7 (quotient)    AH:=AX % d =22 % 3 = 1 (remainder)
mov [x], AL     ; x:=AL=7

push    dword 0
call    [exit]
```

În exemplul de mai sus valoarea fără semn din BL este convertită la BX prin adăugarea de biți de valoare 0 în registrul BH (`mov bh, 0`).
Pentru reprezentarea cu semn dacă numărul este pozitiv, conversia la reprezentare pe 16 biți se face similar cu reprezentarea fără semn: se adaugă biți de valoare 0 ca biți semnificativi până se ajunge la dimensiunea dorită. Exemplu:

```
mov  al, 12
mov ah, 0
```

unde AX = 12 = 0000 0000 0000 1100b (AH = 00000000b și AL = 00001100b).
Dacă numărul ce se vrea convertit este negativ, trebuie adăugați opt biți de valoare 1 ca și biți semnificativi:

```
mov al, -12    ; AL:=1111 0100b
; -12 reprezentat pe 16 biti: 1111 1111 1111 0100b
```

Pentru reprezentarea cu semn conversia la un tip mai mare se face prin adăugarea de biți până la dimensiunea dorită, acești biți au valoarea bitului de semn a numărului ce trebuie convertit. Pentru a face conversia la un tip mai mare în reprezentare cu semn există instrucțiuni specializate, acestea sunt *cbw, cwd, cwde, cdq* și sunt prezentate mai jos.

CBW – conversie cu semn byte → word
Sintaxa: cbw
Efect: se convertește cu semn octetul din AL la cuvânt și e stocat în AX

CWD – conversie cu semn word → dword (dublucuvânt)
Sintaxa: cwd
Efect: se convertește cu semn cuvântul din AX la dublucuvânt și e stocat în DX:AX

CWDE – conversie cu semn word → dword extins (dublucuvânt)
Sintaxa: cwde
Efect: se convertește cu semn cuvântul din AX la dublucuvânt și e stocat în EAX

CDQ – conversie cu semn dword → qword
Sintaxa: cdq
Efect: se convertește cu semn valoarea din EAX la EDX:EAX

Conversii fără semn (nu există instrucțiuni speciale de conversie):

- conversie fără semn AL → AX mov ah, 0
- conversie fără semn AX → DX:AX mov dx, 0
- conversie fără semn EAX → EDX:EAX mov edx, 0

2.3.2 Exemplu

Să se determine valoarea expresiei $x = \frac{a \cdot b}{d} - c$ unde toate numerele sunt reprezentate cu semn și a, c, d sunt octeți iar b este un cuvânt.

Rezolvare:

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

segment data use32 class=data
   a db -3
   b dw 4
   c db 2
   d db 3
   x dw 0

segment code use32 class=code
start:
   mov al, [a]          ; AL:=a = -3
   cbw                  ; AX:=-3    (conversie cu semn a lui AL la AX)
   imul word [b]        ; DX:AX:= AX * B = -3 * 4 = -12
                        ; "d" trebuie convertit de la octet la cuvant
   mov bx, ax           ; DX:BX:=-12
   mov al, d            ; AL:=d = 3
   cbw                  ; AX:=AL=3
   mov cx, ax           ; CX:=3
   mov ax, bx           ; se copiaza BX in AX astfel incat DX:AX=-12
   idiv cx              ; AX:=DX:AX / CX = -12 / 3 = -4       DX:=DX:AX % CX = -12 % 4 = 0
                        ; "c" trebuie convertit de la octet la cuvant
                        ; se elibereaza registrul AX
   mov bx, ax           ; BX:=AX=-4
   mov al, c            ; AL := c = 2
   cbw                  ; AX := AL= 2
   sub BX, AX           ;BX := BX-AX = -4 -2 = -6
   mov [x], BX          ; x := -6
```

```
push    dword 0
call    [exit]
```

Probleme suplimentare:

1. Să se determine valoarea expresiei x=a/2+b*3-c*d+100, unde toate numerele sunt reprezentate cu semn și *a – doubleword*, *b - byte*, *c*, *d –word.*
2. Să se determine valoarea expresiei x+a*b/(c-d)+f*g, unde toate numerele sunt reprezentate cu semn și x – quadword, *a*, *d – byte*, *c - doubleword*, *b*,*g*,*g –word*