Seminar 2

Instructiuni cu / fara semn

* Instructiuni care nu tin cont de reprezentarea numerelor: mov, add, sub
* Instructiuni care isi interpreteaza operanzii ca fiind numere fara semn: mul, div
* Instructiuni care isi interpreteaza operanzii ca fiind numere cu semn: imul, idiv, cbw, cwd, cwde, cdq

**Instructiuni pentru inmultire si impartire (FARA SEMN)**

* **MUL**

Sintaxa: MUL *operand; operand* poate fi BYTE, WORD, DOUBLEWORD

; termen1 \* termen2 = rezultatInmultire

* Daca *operand* este BYTE atunci  
  byte \* byte = word si  
  *operand* \* AL = AX
* Daca *operand* este WORD atunci  
  word \* word = doubleword si  
  *operand* \* AX = DX:AX  
  (DX:AX este o notatie pentru dublucuvantul care are partea high in DX si partea low in AX)
* Daca *operand* este DOUBLEWORD atunci  
  doubleword \* doubleword = quadword si  
  *operand* \* EAX = EDX:EAX  
  (EDX:EAX este o notatie pentru quadword-ul care are partea high in EDX si partea low in EAX)

Observatie: Dimensiunea de reprezentare a rezultatului inmultirii este DUBLA fata de dimensiunea de reprezentare a operanzilor

|  |  |  |
| --- | --- | --- |
| *; 12\*3*  *; byte \* byte = word*  *; operand* \* AL = AX  MOV AL, 12  MOV AH, 3  *; AX=030Ch*  MUL AH  *; AH* \* AL = AX = 36 = 0024h | *; 400 \* 2*  *; word \* word = doubleword*  *; operand* \* AX = DX:AX MOV BX, 400 MOV AX, 2 MUL BX  ; *BX* \* AX = DX:AX = 800 = 320h ; DX=0000h si AX=0320h | ; *11234h \* 11234h* *; doubleword \* doubleword = quadword*  *; operand* \* EAX = EDX:EAX  MOV EAX, 11234h  MUL EAX  ; *EAX* \* EAX = EDX:EAX |

Quadword + Quadword?

Quadword – Quadword?

Pentru a efectua adunarea / scaderea a doi quadwords se aduna / se scad dublucuvintele componente. Cum?

27+  
19  
---  
46

EDX:EAX+  
EBX:ECX  
-----------  
EDX+EBX+CF:EAX+ECX (CF este 0 sau 1 in functie de existenta transportului la operatia anterioara)

In asamblare:  
ADD EAX, ECX; EAX = EAX + ECX  
ADC EDX, EBX; EDX = EDX + EBX + CF (E foarte important ca ADC sa fie imediat dupa ADD)

SUB EAX, ECX; EAX = EAX – ECX  
SBB EDX, EBX; EDX = EDX – EBX – CF

Similar, se pot aduna / scadea si dublucuvintele:  
  
DX:AX+  
CX:BX  
-------  
ADD AX, BX  
ADC DX, CX; => rezultatul adunarii este in DX:AX

PUSH DX  
PUSH AX  
POP EAX; => rezultatul adunarii este in EAX

Alternativ,

PUSH DX  
PUSH AX  
POP EAX  
  
PUSH CX  
PUSH BX  
POP EBX  
  
ADD EAX, EBX; => rezultatul adunarii este in EAX

Daca in urma ultimei adunari se seteaza CF=1 inseamna ca rezultatul adunarii nu incape pe un dublucuvant si e nevoie de un quadword pentru reprezentarea acestuia

PUSH DX  
PUSH AX  
POP EAX  
  
PUSH CX  
PUSH BX  
POP EBX

MOV EDX, 0  
ADD EAX, EBX; => rezultatul adunarii este in EAX  
ADC EDX, 0; => rezultatul adunarii este in EDX:EAX

* **DIV**

Sintaxa: DIV *operand; operand* poate fi BYTE, WORD, DOUBLEWORD

; deimpartit / impartitor = cat r rest

* Daca *operand* este un BYTE atunci:  
  WORD / BYTE = BYTE r BYTE si  
  AX / *operand* = AL rest AH
* Daca *operand* este un WORD atunci:  
  DOUBLEWORD / WORD = WORD r WORD si  
  DX:AX / *operand* = AX rest DX
* Daca *operand* este un DOUBLEWORD atunci:  
  QUADWORD / DOUBLEWORD = DOUBLEWORD r DOUBLEWORD si  
  EDX:EAX / *operand* = EAX rest EDX

Observatie: Dimensiunea de reprezentare a rezultatului (CAT si REST) este egala cu dimensiunea de reprezentare a *operand*-ului, iar deimpartitul are dimensiunea dubla fata de *operand*

|  |  |  |
| --- | --- | --- |
| *; 16 / 3*  *;* WORD / BYTE = BYTE r BYTE  ; AX / *operand* = AL rest AH  MOV AX, 16  MOV BL, 3  *; AX = 0010h* DIV BL  *; AL = AX / BL = 05h* *; AH = AX % BL = 01h*  *; AX = 0105h* | *; 512 / 2 = 256 r 0*  *;* WORD / BYTE = BYTE r BYTE  *; daca alegem w/b = b r b atunci rezulta DIVISION OVERFLOW, adica rezultatul nu a incaput in dimensiunea de reprezentare*  *;* DOUBLEWORD / WORD = WORD r WORD  ; DX:AX / *operand* = AX rest DX  MOV AX, 0200h MOV DX, 0000h  MOV BX, 2  DIV BX  *; AX = DX:AX / BX = 256* *; DX = DX:AX % BX = 0* | *; 112345678h / 2*  *;* QUADWORD / DOUBLEWORD = DOUBLEWORD r DOUBLEWORD ; EDX:EAX / *operand* = EAX rest EDX  MOV EDX, 00000001h MOV EAX, 12345678h MOV ECX, 2 DIV ECX  ; EAX = EDX:EAX / ECX  ; EDX = EDX:EAX % ECX |

**Instructiuni pentru inmultire si impartire (CU SEMN)**

* **IMUL**

Sintaxa si regulile sunt IDENTICE cu cele de la MUL, cu observatia ca operanzii sunt interpretati ca fiind numere cu semn

|  |  |  |
| --- | --- | --- |
| *; -1 \* 4*  *; byte \* byte = word*  MOV AL, -1; AL=11111111b MOV AH, 4; AH=00000100b IMUL AH  ; AL \* AH = AX = -1 \* 4 = -4 | *; -1 \* 4*  *; byte \* byte = word*  MOV AL, -1; AL=11111111b MOV AH, 4; AH=00000100b MUL AH  ; AL \* AH = AX = 255 \* 4 = 1020 | *; -200 \* 4*  *; word \* word = doubleword*  MOV AX, -200 MOV DX, 4 IMUL DX  *; DX:AX=AX\*DX=-800* |

* **IDIV**

Sintaxa si regulile sunt IDENTICE cu cele de la MUL, cu observatia ca operanzii sunt interpretati ca fiind numere cu semn

|  |  |
| --- | --- |
| *; -3 / 2*  *; word / byte = byte r byte*  MOV AX, -3  MOV BL,2  IDIV BL  *; AL = AX / BL* *; AH = AX % BL* | *; 500 / (-1)*  *; doubleword / word = word r word* MOV DX,0 MOV AX,500  MOV BX, -1  IDIV BX |

**Exercitiul 1: Determinati valoarea expresiei x=(a+b)\*c/d, unde a,b,c,d sunt numere fara semn de tipul octet**

|  |  |
| --- | --- |
| **Data segment** | **Code segment** |
| A DB 2  B DB 2  C DB 2  D DB 2  X RESB 1 | MOV AL, BYTE[A]  ADD AL, BYTE[B]  ; AL = (A+B)  MOV AH, BYTE[C]  MUL AH  ; AX = (A+B)\*C  MOV DL, BYTE[D]  DIV DL  ; AL=AX/DL; AH=AX%DL  ; AL=(A+B)\*C/D; AH=(A+B)\*C%D  MOV BYTE[X], AL |

**Exercitiul 2: Determinati valoarea expresiei x=(a+b)\*c/d, unde a,b,c,d sunt numere CU semn de tipul octet**

|  |  |
| --- | --- |
| **Data segment** | **Code segment** |
| A DB -2  B DB 2  C DB -2  D DB 2  X RESB 1 | MOV AL, [A]  ADD AL, [B]  ;AL = (A+B)  IMUL BYTE [C]  ; AX = (A + B) \* C  IDIV BYTE [D]  ; AL = AX / D, AH = AX % D  MOV [X], AL |

**Exercitiul 3: Determinati valoarea expresiei x=(a-b\*c)/d, unde a,b,c,d sunt numere fara semn de tipul octet**

|  |  |
| --- | --- |
| **Data segment** | **Code segment** |
| A DB 12  B DB 2  C DB 102  D DB 2  X RESB 1 | MOV AL, [B]  MUL BYTE [C]  ; AX = AL\*C=B\*C  MOV BL, [A]  XOR BH, BH; <=> MOV BH,0  ; BX=A  SUB BX, AX  MOV AX, BX  DIV BYTE [D]  ; AL = (A-B\*C)/D si AH = (A-B\*C)%D  MOV [X], AL |
| A DB 12B DB 2  C DB 102  D DB 2  X RESB 1 | MOV AL, [B]  MUL BYTE [C]  ; AX = AL\*C=B\*C  NEG AX; AX=-B\*C  MOVZX BX,[A]; BX=A  ; BX=A  ADD AX, BX  DIV BYTE [D]  ; AL = (a-b\*c)/d SI AH = (a-b\*c)%d  MOV [X], AL |

Deoarece byte – word nu se poate realiza direct este necesara o conversie fara semn.

Astfel, conversia fara semn a unui octet / word / doubleword se face prin adaugarea de 0-uri nesemnificative pana se ajunge la dimensiunea dorita

AL -> AX <=> MOV AH,0  
AX -> DX:AX <=> MOV DX,0  
AX -> EAX <=> MOV [aux], AX <=> PUSH word 0  
 MOV EAX, 0 PUSH AX  
 MOV AX, [aux] POP EAX  
EAX -> EDX:EAX <=> MOV EDX,0

**Exercitiul 4: Determinati valoarea expresiei x=(a-b\*c)/d, unde a,b,c,d sunt numere cu semn de tipul octet**

|  |  |
| --- | --- |
| **Data segment** | **Code segment** |
| A DB 12  B DB 2  C DB 102  D DB 2  X RESB 1 | MOV AL, BYTE [B]  MOV AH, BYTE [C]  IMUL AH  ; AX = B \* C  MOV BX, AX  ; BX = B \* C  MOV AL, BYTE [A]  CBW  ; AX = A  SUB AX, BX  ; AX = A – B \* C  IDIV BYTE [D]  ; AL = (A – B \* C) / D  ; AH = (A – B \* C) % D  MOV BYTE [X], AL |

In interpretarea cu semn la conversie trebuie sa se tina cont de bitul de semn

MOV AL, +12  
Daca +12 vrem sa il convertim la word atunci +12 = 00001100b => adaug 0-uri nesemnificative => 00000000 00001100b

MOV AL, -12  
Daca –12 vrem sa il convertim la word atunci –12 = 11110100b => adaug 1-uri nesemnificative => 11111111 11110100b

**Instructiuni de conversie in interpretarea cu semn**

* **CBW - convert byte to word**

Sintaxa: CBW

EFECT: se converteste cu semn octetul din registrul AL la cuvant si se stocheaza in registrul AX

* **CWD – convert word to doubleword**

Sintaxa: CWD

EFECT: se converteste cu semn cuvantul din registrul AX la dublucuvant si se stocheaza in registrul DX:AX

* **CWDE – convert word to doubleword extended**

Sintaxa: CWDE

EFECT: se converteste cu semn cuvantul din registrul AX la dublucuvant si se stocheaza in registrul EAX

* **CDQ – convert doubleword to quadword**

Sintaxa: CDQ

EFECT: se converteste cu semn dublucuvantul din registrul EAX la quadword si se stocheaza in registrul EDX:EAX

**Exercitiul 5: Determinati valoarea expresiei x=a\*b/d-c, unde a,c,d sunt octeti, b este cuvant iar numerele sunt interpretate ca numere fara semn**

|  |  |
| --- | --- |
| **Data segment** | **Code segment** |
| A DB 2  B DW 2  C DB 2  D DB 2  X resw 1 | MOV AL, [A]  MOV AH, 0  ; AX = A  MUL WORD[B]  ; DX:AX= A\*B  MOV BL, [D]  MOV BH, 0  ; BX = D  DIV BX  ; AX = A\*B/D  MOV CL, [C]  MOV CH, 0  SUB AX, CX  MOV [X], AX |

**Exercitiul 6: Determinati valoarea expresiei x=a\*b/d-c, unde a,c,d sunt octeti, b este cuvant iar numerele sunt interpretate ca numere CU semn**

|  |  |
| --- | --- |
| **Data segment** | **Code segment** |
| A DB -2  B DW 2  C DB 2  D DB -2  X resw 1 | MOV AL, [A]  CBW  IMUL WORD [B]  ; DX:AX = A \* B  MOV BX, AX  ; DX:BX = A \* B  MOV AL,[D]  CBW  ; AX = D  MOV CX, AX  MOV AX, BX  IDIV CX  ; AX = DX:AX / CX  ; AX = A \* B / D  MOV BX, AX  MOV AL, [C]  CBW  NEG AX  ADD AX, BX  ; AX = A \* B / D – C  MOV [X], AX |