-Introducere în limbajul de asamblare IA-32

ax^2+bx+c=0

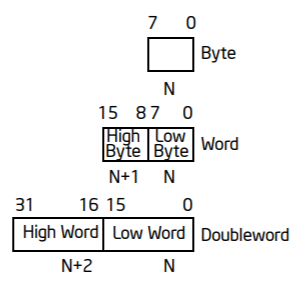
Secventa de pasi / operatii

* P1: calcul delta
* P2: delta >=0 atunci x1, x2

Set de date: a,b,c, x1, x2

Tipuri de date:

* Byte (Octet) - tip de data reprezentat pe 8 biti
  + B0 – bitul cel mai putin semnificativ (bitul low)
  + B7 – bitul cel mai semnificativ (bitul high)
* Word (Cuvant) - tip de data reprezentat pe 16 biti
  + W=O1O0
  + O0 – octetul cel mai putin semnificativ (octet low)
  + O1 – octetul cel mai semnificativ (octet high)
* Double word (dublu cuvant) - tip de data reprezentat pe 32 biti
  + D=W1W0=O11O10 O01O00
  + W0 – cuvantul cel mai putin semnificativ (cuvantul low)
    - O00 – octetul low al cuvantului low
    - O01 – octetul high al cuvantului low
  + W1 – cuvantul cel mai semnificativ (cuvantul high)
    - O10 – octetul low al cuvantului high
    - O11 – octetul high al cuvantului high
* Quad word - tip de data reprezentat pe 64 biti



Date:

* CONSTANTE: Date care nu isi modifica valoarea pe parcursul executiei programului
  + **Numere naturale sau intregi**: pot fi stocate in memorie ca byte, word, doubleword sau quadword
    - Baza 2: 10101b, 110b
    - Baza 10: 7, +15, -200
    - Baza 16: 35h, 3Ah, 0A4Eh
  + **Caracter:** poate fi stocat in memorie ca byte
    - ‘a’,’B’,’!’
  + **Secvente de caractere**: poate fi stocate in memorie ca secvente de octeti, cuvinte, dublu cuvinte sau quadwords
    - ‘asc’, ‘Arhitectura Calculatoarelor’, ‘Ana are mere.’

**zece EQU 10**

* VARIABILE: Date care isi modifica valoarea pe parcursul executiei programului
  + O variabila are NUME, TIP DE DATA (byte, word, doubleword sau quadword), O VALOARE CURENTA si O LOCATIE
  + Predefinite
    - Registrii de uz general al procesorului (sunt de tipul dublucuvant)
      * EAX, EBX, ECX, EDX – pot fi utilizati ca tip de data dublucuvant, cuvant sau octet

|  |  |  |  |
| --- | --- | --- | --- |
| **EAX** – 32 de biti | | | |
| Cuvantul High coresp. lui EAX | | Cuvantul Low coresp. lui EAX: **AX** | |
| Octetul high al cuvantului high coresp lui EAX | Octetul low al cuvantului high coresp lui EAX | Octetul high al cuvantului low coresp lui EAX: **AH** | Octetul low al cuvantului low coresp lui EAX: **AL** |

EAX = registru acumulator; este folosit in majoritatea instructiunilor ca unul dintre operanzi

EBX = registru general

ECX = registru de numarare (contor); este folosit de instructiuni care au nevoie de indicatii numerice

EDX = registru de date; este folosit impreuna cu EAX pentru calcule ale caror rezultate se depaseste un dublucuvant

* + - * ESI, EDI, ESP, EBP - pot fi utilizati ca tip de data dublucuvant sau cuvant

ESI, EDI = registrii de index; sunt utilizati pentru a accesa elementele din siruri de octeti, de cuvinte, …

ESP, EBP = registrii destinati lucrului cu stiva (o zona de memorie in care se pot depune succesiv valori, extragerea acestora facandu-se in ordinea inversa depunerii); ESP = pointeaza catre ultimul element introdus in stiva (varful stivei); EBP = pointeaza catre primul element introdus in stiva (baza stivei)

* + - Registrii de segment (de tip cuvant): CS, DS, SS, ES, FS, GS
      * Se folosesc pentru a retine valorile selectorilor segmentelor active
      * Se pot folosi 4 tipuri de segmente
        + Segment de cod => CS
        + Segment de date => DS
        + Segment de stiva => SS
        + Segment suplimentar de date (extrasegment) => ES
    - Alti registrii: EIP si Eflags
  + Definite de catre utilizator
    - Cu valoare initiala
      * **a DB 7; a=07h**
      * **b DW –1; b=FFFFh**
      * **c DD 4567h; c=0000 4567h**
      * **d DQ 101010b; d=0000 0000 0000 002Ah**
    - Fara valoare initiala
      * **a RESB 1; se rezerva 1 octet**
      * **b RESW 2; se rezerva 2 cuvinte = se rezerva 4 octeti**
      * **c RESB 64; se rezerva 64 de octeti**

INSTRUCTIUNI:

O instructiune poate avea maxim 2 operanzi

!!! Maxim unul din operanzi se poate afla in memoria RAM, celalalt trebuie sa fie intr-un registru sau o constanta intreaga

**nume\_instructiune destinatie, sursa**

1. **MOV dest, sursa**
   * Asignare / atribuire
   * dest <- sursa
   * **dest si sursa pot fi registrii, variabile sau constante**
   * **ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant, dublucuvant)**
   * **dest nu poate fi constanta**

MOV AX, 2; AX=0002h

MOV [a], EAX; a=EAX

MOV [a], BX; a=BX

MOV BYTE [a], 2

1. **ADD dest, sursa**
   * Adunare
   * dest<-dest+sursa
   * **dest si sursa pot fi registrii, variabile sau constante**
   * **ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant, dublucuvant)**
   * **dest nu poate fi constanta**

ADD AL, AH

ADD [a], EAX

ADD WORD [a], 3Ah

1. SUB dest, sursa
   * Scadere
   * dest<-dest – sursa
   * **dest si sursa pot fi registrii, variabile sau constante**
   * **ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant, dublucuvant)**
   * **dest nu poate fi constanta**

SUB BX,2

SUB AL,[a]

Exercitii:

1. 1+2

|  |  |
| --- | --- |
| Data segment | Code segment |
|  | mov AL, 1; AL= 01h  add AL, 2; AL = 03h |

2. a+b, a,b – octeti

|  |  |
| --- | --- |
| Data segment | Code segment |
| A DB 1  B DB 1 | mov AL, [a] ;a=01h, AL=01h  add AL,[b] ; AL=01h+01h=02h |

3. a-b+c, a,b,c – cuvinte

|  |  |
| --- | --- |
| Data segment | Code segment |
| A dw 1  B dw 1  C resw 1 | mov word [c], 4 ;c = 0004h  mov AX, [a] ;AX = 0001h  sub AX, [b] ;AX = 0000h  Add AX, [c] ; AX = 0004h |

4. (a+b) - (x+y) +10, a,b,x,y, - dublucuvinte

|  |  |
| --- | --- |
| Data segment | Code segment |
| A DD 3  B DD 4  X DD 1  Y DD 2 | MOV EAX, [A] EAX = 00000003h  ADD EAX,[B] EAX = 00000007h  Mov ebx, [x]; ebx = 00000001h  Add ebx, [y]; ebx = 00000003h  Sub eax, ebx; eax = 00000004h  Add eax, 10; eax = 0000000Eh |