Y Introducere în limbajul de asamblare IA-32

ax^2+bx+c=0

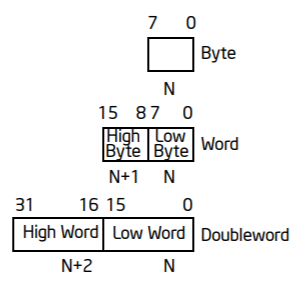
Pasi:

1. Calculam delta
2. Daca delta >=0 x1,x2

Date: a,b,c, x1,x2

Tipuri de date:

* Byte (octet) - tip de date reprezentat pe 8 biti
  + B0 – cel mai putin semnificativ (bit low)
  + B7 – cel mai semnificativ (bit high) => bitul de semn
* Word (cuvant) - tip de date reprezentat pe 16 biti
  + W=o1o0
  + O0 – octetul cel mai putin semnificativ (octetul low)
  + O1 – octetul cel mai semnificativ (octetul high)
* Double word (dublu cuvant) - tip de date reprezentat pe 32 biti
  + D=W1W0=O11O10 O01O00
  + W0 – cuvantul cel mai putin semnificativ (cuvant low)
    - O00 – octetul low din cuvantul low
    - O01 – octetul high din cuvantul low
  + W1 – cuvantul cel mai semnificativ (cuvant high)
    - O10 – octetul low din cuvantul high
    - O11 – octetul high din cuvantul high
* Quad word - tip de date reprezentat pe 64 biti
  + …



Date:

* CONSTANTE: Date care nu isi modifica valoarea pe parcursul executiei programului
  + Numere naturale / intregi
    - In baza 2: 1010b, 111000111b
    - In baza 10: 5, 67, -7, -89
    - In baza 16: 1Dh, 0AFh, 1234h, 0A0h

- pot fi stocate pe un byte, word, double word sau quad word

* + Caracter
    - ‘a’, ‘B’, ‘!’

- poate fi stocat pe un byte

* + Secvente de caractere (String)
    - ‘asc’, ‘Arhitectura Sistemelor de Calcul’

- poate fi stocat ca secvente de bytes, words, doublewords sau quadwords

**zece EQU 10**

* VARIABILE: Date care isi modifica valoarea pe parcursul executiei programului
  + Variabilele au un nume, un tip de data, o valoare curenta si o locatie de memorie
  + Variabile predefinite (Registrii )
    - Locatii de memorie pe procesor utilizati pentru diverse calcule
    - Registrii generali (de tip dublucuvant)
      * EAX, EBX, ECX, EDX – pot sa fie utilizati ca si tip de data dublucuvant, cuvant sau octet

|  |  |  |  |
| --- | --- | --- | --- |
| **EAX** | | | |
| Cuvantul high al dublucuvantului EAX | | Cuvantul low al dublucuvantului 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 si este folosit de majoritatea instructiunilor ca fiind unul din operanzi

EBX = registru general

ECX = registru de numarare (registru contor)

EDX = registru de date care este folosit de cele mai multe ori impreuna cu EAX in calcule ale caror rezultate depasesc un dublu cuvant

* + - * ESI, EDI, ESP, EBP – pot sa fie utilizati ca si tip de data dublucuvant sau cuvant

ESP, EBP = registrii destinati lucrului cu stiva (stiva = o zona de memorie in care se depun succesiv valori, extragerea lor facandu-se in ordinea inversa depunerii)

ESP = pointeaza spre ultimul element introdus in stiva (varf)

EBP = pointeaza spre primul element introdus in stiva (baza)

EDI si ESI = registrii de index utilizati pentru a accesa elemente din siruri de octeti sau de cuvinte

* + - Registrii de segment
      * CS, DS, SS, ES, FS, GS
      * Arhitectura x86 permite folosirea a 4 tipuri de segmente
        + Segment de cod
        + Segment de date
        + Segment de stiva
        + Segment suplimentar de date
    - Alti registri
      * EIP
      * EFLAGS
  + Variabile definite de utilizator
    - Cu valoare initiala

**A db 3**

**B dw 23h**

**C dd 1**

**D dq -1**

* + - Fara valoare initiala

**A resb 1**

**B resw 2**

**C resb 64**

INSTRUCTIUNI

O instructiune are maxim 2 operanzi

!!! Dintre cei doi operanzi maxim unul se poate afla in memoria RAM. Celalalt operand este fie un registru fie o constanta intreaga

1. MOV dest , sursa

* Asignarea / atribuirea: dest <- sursa
* Dest si sursa sunt registrii, variabile sau constante
* Ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant sau dublucuvant)
* Dest nu poate fi o constanta

Ex.

MOV AX,2; AX=2=0002h

MOV [a], EAX

1. ADD dest , sursa

* Adunare: dest <- dest + sursa
* Dest si sursa sunt registrii, variabile sau constante
* Ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant sau dublucuvant)
* Dest nu poate fi o constanta

Ex.

ADD BX, CX

ADD BYTE [a],8

1. SUB dest , sursa

* Scadere: dest <- dest – sursa
* Dest si sursa sunt registrii, variabile sau constante
* Ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant sau dublucuvant)
* Dest nu poate fi o constanta

Exercitii:

1. 1-2

|  |  |
| --- | --- |
| Data segment | Code segment |
|  | MOV EAX, 2  MOV EBX, 1  SUB EBX, EAX |

1. a+b, a,b – byte

|  |  |
| --- | --- |
| Data segment | Code segment |
| a db 8  b resb 1 | MOV byte [b], 5; MOV AL,5 [\\](file:///\\) MOV [b], AL  MOV AH, [a]  ADD AH, [b] |

1. a-b, a,b – doubleword

|  |  |
| --- | --- |
| Data segment | Code segment |
| a dd –1  b dd 7 | MOV EAX, [a]  MOV EBX, [b]  SUB EAX, EBX |

1. (a+b)-(x+10), a,b,x – word

|  |  |
| --- | --- |
| Data segment | Code segment |
| a dw 3  b dw 5  x dw 7 | MOV AX, [a]  ADD AX, [b] ; AX=a+b  MOV BX, [x]  ADD BX, 10 ; BX = x+10  SUB AX, BX |

-10 (reprezentat pe un octet)

10 / 2 = 5 r 0

5 / 2 = 2 r 1

2 / 2 = 1 r 0

1 / 2 = 0 r 1

10 = 00001010b

1. Complementul fata de 2  
   P1: 11110101b

1  
P2: 11110101b+  
 1b  
---------------------

11110110b => -10 = 11110110b

1. Reguli alternative
   1. 000010**10b** => 11110110b = -10
   2. 1111112  
      100000000b-  
       00001010b  
      -----------------  
       11110110b = -10
   3. 0 15 16  
      1 0 0h-  
       Ah  
      -------------  
       F6h = 1111 0110b = -10

0 9 10  
1 0 0-  
 3  
--------  
 97

Un octet (n=8)

* Interpretarea fara semn: [0,255] = [0, 2^n - 1]
* Interpretarea cu semn: [-128, +127] = [-2^(n-1) , 2^(n-1) -1]

Interpretarea fara semn:

10 = 00001010b (un octet)

= 00000000 00001010b (un cuvant)

= 00000000 00000000 00000000 00001010b (un dublu cuvant)

Interpretarea cu semn:

+10 = 00001010b (un octet)

= 00000000 00001010b (un cuvant)

= 00000000 00000000 00000000 00001010b (un dublu cuvant)

-10 = 1111 0110b (un octet)

= 11111111 1111 0110b (un cuvant)

= 11111111 11111111 11111111 1111 0110b (un dublu cuvant)