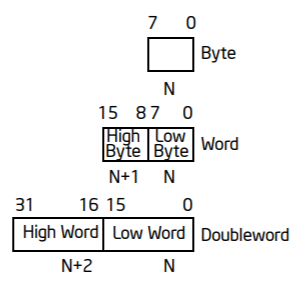
Introducere în limbajul de asamblare IA-32

* Identificarea solutiilor ecuatiei de gradul 2: ax^2+bx+c=0
  + Secventa de pasi
    - Calculam delta
    - Daca delta >=0 calculam x1 si s2 solutiile ecuatiei
  + Set de date / entitati care sunt folosite de algoritm
    - a,b,c – coeficienti si x1,x2 – solutii
* Tipuri de date IA-32
  + Byte (octet) - tip de data reprezentat pe 8 biti
    - b7b6b5b4b3b2b1b0
    - b0 – bitul cel mai putin semnificativ (bit low)
    - b7 – bitul cel mai semnificativ (bit high)
  + Word (cuvant) - tip de date reprezentat pe 16 biti (2 octeti)
    - o1o0 = b15b14....b8 b7b6b5...b0
    - b0 – bitul cel mai putin semnificativ (bit low)
    - b15 – bitul cel mai semnificativ (bit high)
    - o0 – octetul cel mai putin semnificativ (octet low, octet inferior)
    - o1 – octetul cel mai semnificativ (octet high, octet superior)
  + Double word (dublu cuvant) - tip de date reprezentat pe 32 biti (2 cuvinte => 4 octeti)
    - w1w0 = o11o10 o01o00
    - w0 – cuvantul cel mai putin semnificativ (cuvantul low)
      * o00 – octetul low din cuvantul low
      * o01 – octetul high din cuvantul low
    - w1 – cuvantul cel mai semnificativ (cuvantul high)
      * o10 – octetul low din cuvantul high
      * o11 – octetul high din cuvantul high
  + Quadword – tip de date reprezentat pe 64 de biti (2 dublucuvinte => 4 cuvinte => 8 octeti)
    - d1d0 = …
    - d0 – dublucuvantul cel mai putin semnificativ (low)
    - d1 – dublucuvantul cel mai semnificativ (high)



Date

* + In timpul executiei programului nu isi modifica valoarea: **CONSTANTE** (date constante)
    - 3 tipuri
      * Numerica (numere natural / intregi) - poate fi scrisa in baza 2, 10 sau 16
        + In baza 2: 1010b, 011b, 11b
        + In baza 10: 10, 8 ,- 5, 2
        + In baza 16: 3Fh, 2A3Bh, 0FAh

Valoarea numerica se poate stoca in memorie ca byte, word, doubleword sau quadword

* + - * Caracterul
        + ‘a’, ‘B’, ‘!’

Caracterul se poate stoca in memorie ca byte

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

String-ul se poate stoca in memorie ca secvente de bytes, cuvinte de dublucuvinte sau de quadwords

* + - DECLARARE CONSTANTE
      * **zece EQU 10**
      * **doi EQU 2**
  + In timpul executiei programului isi modifica valoarea: **VARIABILE** (date variabile)
    - O variabila are un nume, un tip de date, o valoare curenta si o locatie de memorie
    - 2 tipuri:
      * Variabile predefinite
        + REGISTRII de uz general ai procesorului

De tip dublucuvant (registrii de uz general)

EAX = registrul acumulator – el este folosit de marea majoritate a instructiunilor ca operand implicit

EBX = registru g

eneral

ECX = registru de numarare (contor) - el este folosit de instructiuni care au nevoie de index numeric

EDX = regsitru de date – impreuna cu EAX este folosit in calculele ale caror rezultate depasesc un dublucuvant

- Pot fi utilizati ca si tip de data reprezentat pe dublu cuvant, cuvant sau octet

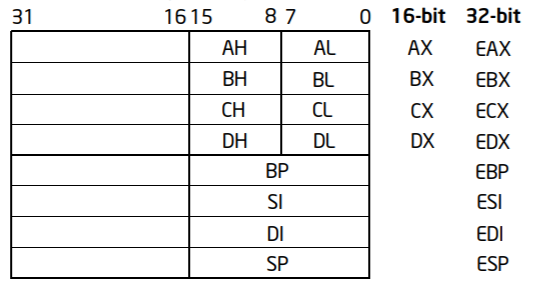
|  |  |  |  |
| --- | --- | --- | --- |
| EAX | | | |
| Cuvantul high al lui EAX | | Cuvantul low al lui EAX **= AX** | |
| Octetul high al cuvantului high din EAX | Octetul low al cuvantului high din EAX | Octetul high al cuvantului low din EAX **= AH** | Octetul low al cuvantului low din EAX  **= AL** |

ESP, EBP, ESI, EDI

- sunt utilizati ca si date de tip dublu cuvant sau cuvant

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

- EDI, ESI – sunt registrii de index si se folosesc de obicei pentru a accesa elemente din siruri de octeti, cuvinte, ...



* + - * + Registrii de segment (de tipul cuvant): CS (code segment) , DS (data segment), SS (stack segment), ES (extra segment), FS, GS

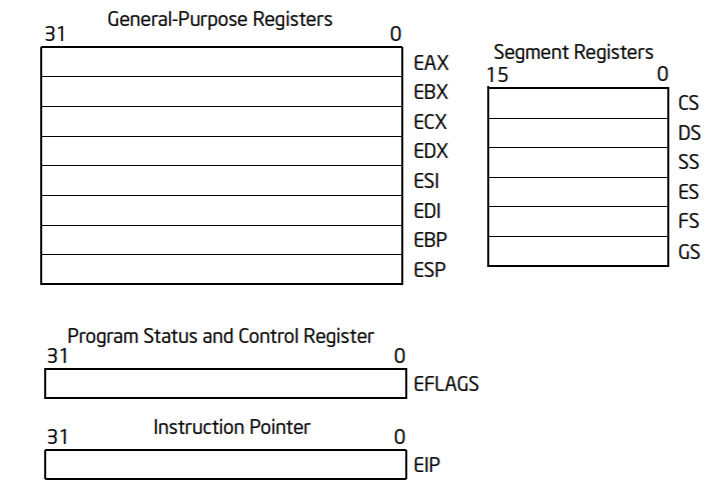
Arhitectura x86 permite folosirea a patru tipuri de segmente

Segment de cod – contine instructiuni

Segment de date – contine datele gestionate de instructiuni

Segment de stiva

Segment suplimentar de date



* + - * + Alti registri: EIP , EFLAGS (de tip dublucuvant)
      * Variabile definite de catre utilizator
        + VARIABILE CU VALOARE INITIALA

**a db 7 ; variabila a are tipul byte si valoarea 7**

**b dw 10**

**c dd –8**

**d dq 89h**

* + - * + VARIABILE FARA VALOARE INITIALA

**a resb 1**

**b resb 64**

**c resw 1; c este o variabila de tip word fara valoare initiala pentru care s-a rezervat 1 word**

* **a-> adresa variabilei (offset)**
* **[a] -> valoarea variabilei, valoarea de la adresa a**
* **!!! In memorie “casutele” sunt de tip octet**

INSTRUCTIUNI:

numeinstructiune destinatie, sursa

* O instructiune are maxim 2 operanzi
* Dintre cei doi operanzi maxim unul se poate afla in memoria RAM; celalalt se afla fie intr-un registru, fie este o constanta intreaga

1. **MOV dest, sursa** (dest:=sursa – operatie de atribuire)

!!! Dest, sursa – registrii, variabile sau constante, ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant sau dublucuvant); dest – nu poate sa fie constanta!

MOV ax,2; ax:=2

MOV [a], eax; a:=eax

1. **ADD dest, sursa** (dest := dest + sursa)

!!! Dest, sursa – registrii, variabile sau constante, ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant sau dublucuvant); dest – nu poate sa fie constanta!

ADD bx,cx; bx:= bx+cx

ADD [a], 101b; a:= a+ 101b

1. **SUB dest, sursa** (dest:= dest - sursa)

!!! Dest, sursa – registrii, variabile sau constante, ambii operanzi trebuie sa fie de acelasi tip (octet, cuvant sau dublucuvant); dest – nu poate sa fie constanta!

2.

SUB dx, 2; dx:= dx-2

SUB [a], cx; a:= a – cx

Exemple:

* 1+2

|  |  |
| --- | --- |
| Segment de date | Segment de cod |
|  | MOV AL,1; AL=1  ADD AL, 2; AL=1+2=3 |

* 1-2

|  |  |
| --- | --- |
| Segment de date | Segment de cod |
|  | MOV AX,1; AX=1  SUB AX,2; AX=-1 |

* a,b,c – dublu cuvinte; a+(b-c)

|  |  |
| --- | --- |
| Segment de date | Segment de cod |
| a dd 3  b dd 4  c resd 1 | MOV EAX,2  MOV [c],EAX  ; c=2 = 00000000 00000000 00000000 00000010b = 00 00 00 02h  MOV EAX,[b] ; eax=b; ax=4  SUB EAX,[c] ; eax=b-c; ax=4-2=2  ADD [a],EAX ; a=a-(b-c) = 3 – 2 = 1 |

* (a+b) – (c+10), a,b,c – qw !!! TEMA OPTIONALA

Intrebare:

1. registrii generali ai procesorului sunt:

EAX (DA, NU?), EBX, CS, ESP, EDP

1. Este corecta urmatoarea operatie?

MOV EAX, BX? (daca da, de ce, daca nu, de ce)