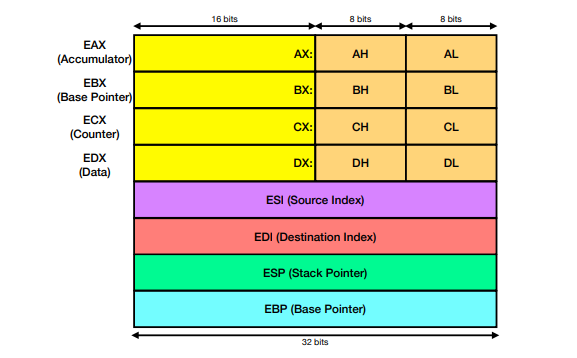
**Exercitii tutoriat 1 – REZOLVARE**

**Exercitiul 1:**

****

1. **Ce valoare va retine EAX dupa executarea urmatoarei secvente de instructiuni?**

movl $0, %eax

movb $4, %ah

movb $2, %al

Răspuns: 1026

Explicatie:

Consideram urmatoarea reprezentare pentru %eax:

AH: AL:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Dupa executia instructiunii: movl $0, %eax vom avea:

AH: AL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 |

Dupa executarea instructiunii: movb $4, %ah vom avea:

AH: AL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 1 0 0 | 0 0 0 0 0 0 0 0 |

Dupa executarea instructiunii: movb $2, %al vom avea:

AH: AL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 1 0 0 | 0 0 0 0 0 0 1 0 |

Dupa executarea instructiunilor in %eax ramane:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 1 0 0 | 0 0 0 0 0 0 1 0 |

care tradus din binar in zecimal: 21 + 210 = 2 + 1024 = 1026

1. **Ce valoare va retine registrul CH dupa executarea urmatoarei instructiuni?**

movl $553, %ecx

Raspuns: 2

Explicatie:

Dupa mutarea numarului 553 (tradus in baza 2) in registrul %ecx, obtinem urmatoarea reprezentare:

CH: CL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 1 0 | 0 0 1 0 1 0 0 1 |

Observam ca in CH avem valoarea: (0 0 0 0 0 0 1 0)2 = (2)10

1. **Ce valoare va retine BH si BL dupa executarea urmatoarei instructiuni?**

movl $484, %ebx

Analog, dupa executarea instructiunii avem:

BH: BL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 1 | 1 1 1 0 0 1 0 0 |

In BH vom avea (0 0 0 0 0 0 0 1)2 = (1)10

In BL vom avea (1 1 1 0 0 1 0 0)2 = (228)10

**Exercitiul 2:**

**Ordonati crescator in functie de spatiul ocupat in memorie urmatoarele declaratii:**

1. A: .long 10

B: .byte 50

C: .asciz „Sirul de caractere:\n”

D: .space 20

Raspuns: B A D C

Explicatie:

* A: .long 10 este declararea variabilei cu numele A, de tipul long, cu valoarea 10, asadar spatiul ocupat de ea este spatiul ocupat de un long adica 4 bytes
* B: .byte 50 este declararea variabilei cu numele B, de tipul byte, cu valoarea 50, asadar spatiul ocupat de ea este spatiul ocupat de un byte adica 1 byte
* C: asciz „Sirul de caractere:\n” Variabila C de tip asciz ocupa 21 bytes. Fiecare litera, spatiu, semn de punctiatie reprezinta un byte iar „\n”=1 byte. Adunandu-le pe toate obtinem 20 de bytes la care se mai adauga un byte datorita tipului de date asciz care adauga si terminatorul de sir => 21 bytes
* D: .space 20 inseamna ca variabilei D i se aloca in memorie un spatiu gol de 20 de bytes

1. A: quad 1

B: .asciz „h1!\n”

C: .word 50

D: .space 4

Raspuns: C D B A

Explicatie: A ocupa 8 bytes

B ocupa 5 bytes

C ocupa 2 bytes

D ocupa 4 bytes

1. A: .byte 5

B: .ascii „0123”

C: .asciz „0123”

D: .word 1

Raspuns: A D B C

Explicatie: A ocupa 1 byte

B ocupa 4 bytes (ascii nu contine terminatorul de sir)

C ocupa 5 bytes

D ocupa 2 bytes

**Exercitiul 3:**

1. **Care este valoarea maxima pe care o poate lua n in urmatoarea declaratie x: .byte n ?**

Raspuns: 255

Explicatie: x este de tipul byte => cea mai mare valoare pe care o poate lua n este cea mai mare valoare care incape pe 8 biti = 255

(binar ar arata: 1 1 1 1 1 1 1 1)

1. **Care este valoarea maxima pe care o poate lua n in urmatoarea declaratie x: .word n ?**

Raspuns: 65.535

Explicatie: analog subpunctului a) avem nevoie de cea mai mare valoare care incape pe 2 bytes (16 biti)

**Exercitiul 4:**

**Se condisdera declarate x: .word 1 si y: .word 2. Ce valoare va avea %eax dupa executarea instructiunii mov x, %eax ?**

a. 1

b. 2

c. 0002 0001

Raspuns: c)

Explicatie: Trebuie sa intelegem ca memoria este continua, iar la declarare variabilele x si y vor fi stocate unele langa altele in memorie. Instructiunea mov x, %eax va muta in %eax 32 de biti din memorie. De ce 32 biti daca x este pe 16 biti? Deoarece numarul de biti mutati trebuie sa fie compatibil cu dimensiunea lui %eax care este de 32 biti. Asadar se va muta in %eax si informatia din y deoarece, in memorie, aceasta se afla langa x.

Nota: daca am vrea sa mutam doar informatia din x, o varianta ar putea fi: movw x, %ax.

**Exercitiul 5:**

1. **Fie urmatoarea declarare in sectiunea .data:**

str1: .ascii "abc"

str2: .ascii "1"

Ce se va afisa in urma apelului WRITE urmator?

movl $4, %eax

movl $1, %ebx

movl $str1, %ecx

movl $4, %edx

int $0x80

1. abc
2. abc1
3. nimic
4. abc + o valoare reziduala

Raspuns: b

Explicatie: Asemanator exercitiului 4, stringurile str1 si str2 sunt puse in memorie unul dupa altul. Apelul WRITE cere sa afiseze 4 bytes (ne dam seama din instructiunea movl $4, %edx), iar cum stringul str1 ocupa doar 3 bytes (este declarat .ascii, adica fara terminatorul de sir), sistemul va afisa un byte “de langa” adica pe “1”

1. **Fie urmatoarea declarare in sectiunea .data:**

str: .ascii "1234"

x: .byte 97

Ce se va afisa in urma apelului WRITE urmator?

movl $4, %eax

movl $1, %ebx

movl $str, %ecx

movl $5, %edx

a.1234

b.1234a

c.12349

d.123497

e.1234a9

f.1234a97

Raspuns: b

Explicatie: Asemanator subpunctului a) apelul va afisa din memorie 5 bytes, 4 bytes din cei ai stringului str, iar al cincilea va fi x care la afisare se converteste in ascii (OBS. apelul de system WRITE poate fi folosit doar pentru afisarea stringurilor)

**Exercitiul 6:**

.data

x: .long 0x04030201

y: .long 0x08070605

.text

.global main

main:

mov x, %eax

mov y, %ah

mov $1, %eax

mov $0, %ebx

int $0x80

Acesta se compileaza si executabilul se ruleaza folosind gdb. In gdb se vor da

urmatoarele comenzi:

b main

run

stepi

stepi

Ce valoare va afisata in %eax in urma rularii comenzi i r?

a. 0x04030201

b. 0x08070605

c. 1

d. 0x04030501

e. 0x04030205

Raspuns: d.

Explicatie: Dupa instructiunea mov x, %eax registrul %eax va contine:

AH: AL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 4 | 0 3 | 0 2 | 0 1 |

(obs: numerede din desen sunt scrise in baza 16)

Instructiunea mov y, %ah va muta in %ah doar atatia biti din y cati vor incapea in %ah (primii 8 biti adica 0 5)

Astfel dupa cele 2 instructiuni, registrul va arata astfel:

AH: AL:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 4 | 0 3 | 0 5 | 0 1 |