**Exercitii tutoriat 2 – REZOLVARE**

**(extrase din testul de lab 2)**

1. Fie urmatoarea declarare in sectiunea .data:

mySpace: .space 100

Acest spatiu poate fi utilizat pentru a retine ulterior:

1. un array de 30 de word-uri
2. un array de 25 de long-uri
3. un array de 50 de bytes
4. un array de 50 de word-uri

Raspuns: b) si d)

Explicatie: un space ocupa un byte => 100 space = 100 bytes

25 de long uri = 25 \* 4 bytes = 100 bytes

50 de word uri = 50 \* 2 bytes = 100 bytes

1. In apelul sistem WRITE, sirul este incarcat in %ecx cu simbolul $. De exemplu, pentru str: .asciz "Sir" incarcarea in %ecx se va face cu $str.

Care este scopul acestui simbol?

1. semnifica faptul ca sirul este constant
2. conventia WRITE, unde toate argumentele trebuie precedate de $
3. semnifica preluarea adresei din memorie pentru str
4. nu are nicio semnificatie, se poate utiliza si scrierea str pentru a obtine exact acelasi rezultat

Raspuns: c)

Explicatie: De ce nu e a) ? str nu este o constanta, ci o variabila (si poate fi modificata la fel ca in orice limbaj high-level).

De ce nu e b) ? Nu exista o astfel de conventie, celelalte argumente (ex: 4, 1) sunt precedate de $ deoarece sunt constante.

1. In instructiunea pentru intreruperea hardware, int $0x80, parametrul 0x80 este prefixat de simbolul $.

Care este semnificatia acestui simbol in contextul curent?

1. reprezinta o conventie de implementare a intreruperii hardware;
2. simbolul $ este utilizat pentru prefixarea unei constante;
3. semnifica preluarea adresei din memorie la care se regaseste codul de intrerupere 0x80;
4. nu are nicio semnificatie, se poate utiliza si scrierea int 0x80, cu acelasi rezultat.

Raspuns: b)

Explicatie: 0x80 este un numar scris in baza 16, deci o constanta.

1. Se condisdera declarate x: .word 1, y: .word 0, z: .word 2.

Ce valoare va avea eax dupa executarea instructiunii mov x, %eax ?

1. 1
2. 2
3. 0x00020001

Raspuns: a)

Explicatie: (vezi explicatia completa in exercitiile rezolvate de la tutoriatul 1) Aici instructiunea mov x, %eax va muta 32 de biti in eax, adica bitii in care sunt stocate variabilele x si y, asadar registrul eax va arata dupa mutare astfel:

|  |  |
| --- | --- |
| 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 1 |

1. Fie urmatoarea declarare in sectiunea .data

str1: .ascii "abc"

str2: .ascii "123"

Ce se va afisa in urma apelului WRITE urmator?

movl $4, %eax

movl $1, %ebx

movl $str1, %ecx

movl $5, %edx

int $0x80

1. abc
2. abc12
3. nimic
4. abc + o valoare reziduala

Raspuns: b)

Explicatie: apelul de WRITE primeste paramentrul 5 in %edx, adica trebuie sa scrie 5 bytes. Cum str1 si str2 sunt puse alaturat in memorie, functia va afisa doi bytes de langa parametrul str1, adica primii doi bytes din str2

1. Nu sunt gresit utilizate urmatoarele instructiuni mov, CU EXCEPTIA:
2. mov %eax, %ebx
3. mov $4, %eax
4. mov %ecx, $1
5. mov $4, %edx

Raspuns: c)

Explicatie: 1 este o constanta si de aceea nu ii putem modifica valoarea

1. Fie urmatorul program. Precizati secventa corecta de instructiuni in debugger, in urma careia vom obtine valoarea 8.

.data

.text

.global main

main:

movl $8, %eax

movl $2048, %ecx

et\_exit:

movl $1, %eax

movl $0, %ebx

int $0x80

1. b main; run; stepi; stepi; i r cl
2. b main; run; stepi; stepi; i r ch
3. b main; run; i r eax
4. b main; run; stepi; i r ah

Raspuns: b)

Explicatie: dupa executia instructiunii movl $2048, %ecx, registrul ecx va arata astfel:

CH: CL:

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

**Mini exercitiu:** Ce valoare va avea eax dupa a doua operatie de mai jos?

.data

.text

.globl main

main:

    movl $1, %eax

    not %eax

    movl $1, %eax

    movl $0, %ebx

    int $0x80

Raspuns: -2

Explicatie: Not %eax intoarce complementul unui numar, deci din 0x00000001 va ajunge 0xFFFFFFFE. Acest numar este reprezentat in complement fata de 2.

Daca folosim formula de la Cursul 1, facem complementul, adunam 1 si o sa ajungem la %eax = -2.