Test de laborator ASC

Exercițiul 1 (1.5 p)

Scrieți dimensiunea alocată în RAM, atât în biți cât și în bytes, pentru următoarele variabile declarate în secțiunea .data:

Declarație în date	Dimensiune în biți	Dimensiune în Bytes
str1: .ascii "Test ASC L0x04\n"		
str2: .asciz "Test ASC L0x04\n"		
myWord: .word 255		
myLong: .long 2703		
myByte: .byte '#'		

1.1) Este o declarare in sectiunea .data de forma x: y: .long 15 valida? Argumentati raspunsul

1.2) Scrieti un apel de system WRITE (codul apelului este 4) pentru sirul de caractere str2, astfel incat sa afisati, la standard output, intreg continutul sirului de caractere si nimic altceva in plus.

Exercițiul 2 (0.5 p)

Completați următorul tabel, cu valorile corecte în bazele 2, 8 și 16.

Baza 2	**Baza 8**	**Baza 16**
		41 20 00 00

Exercițiul 3 (1 p)

Fie următorul program, scris în assembly (procesor Intel x86, sintaxă AT&T):

```
.data
        x: .long 0x41200000
.text
.global main
main:
        movl $8, %eax
        mull x
et_exit:
            movl $1, %eax
            xorl %ebx, %ebx
            int $0x80
```

Se execută în **debugge**r următoarele comenzi:

b main; b et_exit; run; c; i r edx

Valoarea pe care o vom obține în registrul EDX este _____.

Exercițiul 4 (1 p)

Considerăm că, în programul de la exercițiul anterior, după declarația ***mull x*** se inserează instrucțiunile:

movl $2, %ebx
div %ebx
(restul programului rămâne identic, deci dupa cele doua instructiuni uremaza ***et_exit***)

Executam in **debugger** urmatoarele comenzi:

b et_exit; run

4.1) Descrieti ce ar trebui sa se intample in acest punct, utilizand modul in care se executa instructiunea **div**:

4.2) Ce se întâmplă dacă modificăm atribuirea în EBX sa fie:

movl $16, %ebx

(înainte de a executa div %ebx)?

Exercițiul 5 (1 p)

Fie următorul program, scris în assembly (procesor Intel 286, sintaxă AT&T):

```
.data
x: .long 18
y: .long 20

.text
.global main
main:
   movl x, %eax
   addl y, %eax
   subl $2, %eax
   xorl %edx, %edx
   divl y
   adl %edx, %eax
   mul %eax
```

```
    sub %edx, %eax
    decl %eax
    cmp $272, %eax
    je et2
et1:
movl $1, %eax
jmp et_exit
et2:
    movl $2, %ebx
et_exit:
mov $1, %eax
    xorl %ebx, %ebx
    int $0x80
```

Completați valorile registrelor în momentul în care executarea programului ajunge în dreptul etichetei et_exit (dupa ce am rulat in debugger comenzile b et_exit; run; i r).

EAX	***EBX***	***EDX***

Exercițiul 6 (1 p)

Fie următorul program, dezvoltat în limbajul de asamblare al procesorului Intel x86, sintaxă AT&T:

```
.data
.text
.global main
main:
    movl $0xa5cf05c0, %eax
    xorl %ecx, %ecx
    xorl %ebx, %ebx

    movw %ax, %cx
    movb %ch, %cl
    xorb %ch, %ch
    movb %ah, %bl
    movb %al, %bh
et_while:
```

```
    addb %bl, %bh
    loop et_while
et_exit:
    movl $1, %eax
    xorl %ebx, %ebx
    int $0x80
```

Valoarea stocată în dreptul etichetei et_exit în registrul **EBX** (în baza 16) este: _____

Exercițiul 7 (1 p)

Explicați dacă următoarele două secvențe de cod accesează elementul de pe indexul al doilea al unui array de întregi, stocat în RAM la adresa cu numele v.
Sunt secvențele de cod echivalente? Dacă da, de ce? Dacă nu, ce accesează fiecare, în parte?

a.

```
mov $v, %edi
movl $2, %ecx
movl (%edi, %ecx, 4), %ebx
```

b.

```
mov $v, %edi
addl $8, %edi
movl $0, %ecx
movl (%edi, %ecx, 4), %ebx
```

Exercițiul 8 (2 p)

Scrieți o secvență de cod în main care să calculeze **suma pătratelor elementelor** dintr-un array. Considerați că lungimea array-ului este n, iar numele lui este v. Aceste informații sunt deja în secțiunea .data.
Soluția voastră trebuie să înceapă de la eticheta main: