Arhitectura Sistemelor de Calcul **Nume student:** DONEA FERNANDO-EMANUEL
Semestrul I, an univ. 2024-2025 **Grupa student:** 143
Subiect tip B
Timp de rezolvare: 40 de minute **Data sustinerii:** 4 noiembrie 2024

Acest subiect conține **3** pagini (incluzând această pagină) și **7** întrebări.
Numărul total de puncte este de 100. Punctajul final conform cursului se va calcula astfel: $\frac{x*100}{100} * 0.01$, unde x reprezintă punctajul obținut de studentul examinat.
Orice tentativă de fraudare a acestui test de laborator se va sancționa conform regulamentului Facultății de Matematică și Informatică a Universității din București aflat în vigoare!

Tabel cu punctaje per subiect (doar pentru profesor)

Question:	1	2	3	4	5	6	7	Total
Points:	5	5	10	10	15	25	30	100
Score:	0	5	10	0	15	0	30	60

1. (5 points) Ce valoare numerică afișează comanda *print (int) a* rulată în GDB știind că programul conține în secțiunea *.data* `a: .word 0xFFFF`?:

   A. 65535

   B. -1

2. (5 points) Ce reprezintă scrierea (%edx, %eax) pentru înmulțire?:

   A. reprezentarea unui factor dintre cei doi ca fiind $2^{32} * \%edx + \%eax$

   B. reprezentarea produsului ca fiind $2^{32} * \%edx + \%eax$

3. (10 points) Ce se va întampla în urma secvenței `movl $'A', %ah`, selectați varianta corecta (cod ASCII 'A' este 65):

   A. În registrul %eax vom reține valoarea 0x00000042.

   B. Programul nu compilează deoarece %ah nu poate reține un caracter.

   C. În registrul %eax vom avea o valoare de tipul 0x...42xx.

   D. Programul nu compilează deoarece `movl` are mnemonic / sufixare greșită din punct de vedere al tipului.

4. (10 points) Care sunt valorile regiștrilor %eax, %edx după execuția `imul %ebx`, știind că inițial registrul %eax are valoarea 0x7FFFFFFF, registrul %ebx are valoarea 0xFFFFFFFE și %edx are valoarea 0x0.

   A. eax = −1, edx = 2

   B. eax = 2, edx = −1

C. eax $= 2^{32} - 1$, edx $= -1$

D. Nu se poate executa acea linie de cod cu valorile prezentate.

5. (15 points) Avem următorul program in limbajul GNU Assembly x86 sintaxa (AT&T), care este ordinea corectă a etichetelor în momentul execuției programului?:

```
.data
a: .byte 0xFF
b: .word 0xFFFF
.text
.global main
main:
    mov $0x7FFF, %ecx
    cmp a, %ecx
    jle jtb
jta:
    xorl %edx, %edx
    jz jtx
jtb:
    cmp b, %cx
    jge jtx
jtc:
    movw $0x8FFF, %cx
    cmp b, %cx
    jl jta
jtx:
    mov $1, %eax
    xor %ebx, %ebx
    int $0x80
```

A. *main, jtb, jtx*

B. *main, jtb, jtc, jtx*

C. *main, jtb, jtc, jta, jtb, ...*

D. *main, jtb, jtc, jta, jtx*

6. (25 points) În programul de mai jos se regăsesc niște greșeli de sintaxă, corectați aceste greșeli indicând rezolvările sub forma *număr_linie: sintaxa_corectă*, știind că se afișează "Hello!":

```
.data
str: .asciz "Hello, world!\n"
.text
.global main
main:
    movl $str, %edi
```

```
    mov $5, %ecx

    xor %edx, %edx
    add $'?', %edx
    mov %dx, 0(%edi, 5, 1)

    xorl %ebx, %ebx
    mov $4, %eax
    movb $1, %bl
    mov %edi, %ecx
    mov $9, %edx
    int $127
et_exit:
    mov $1, %eax
    xor %ebx, %ebx
    int $128
```

7. (30 points) Scrieți un program în limbajul GNU Assembly x86 (sintaxa AT&T) care salvează în registrul *%edx* suma elementelor aflate pe pozițiile impare dintr-un tablou unidimensional de numere întregi salvat în secțiunea *.data*.

```
.data
    n: .long 5
    v: .long 10,20,30,40,50
.text
.global main
main:
    lea v, %edi
    mov $0, %ecx
    mov $0, %edx

etloop:
    cmp n, %ecx
    jg etexit
    movl (%edi,%ecx,4), %ebx
    add %ebx, %edx
    add $2, %ecx
etexit:
    mov $1, %eax
    mov $0, %ebx
    int $0x80
```