Test de laborator
Arhitectura Sistemelor de Calcul

Seria 13

8 Ianuarie 2021

Cuprins

1 Informatii generale

1. Se acorda 4p din oficiu, iar nota maxima ce poate fi obtinuta este 10.
2. Subiectul e impartit in doua: o parte de implementare si o parte de analiza de cod si de intrebari generale.
3. Timpul efectiv de lucru este de 1h si 10 de minute din momentul in care subiectele sunt transmise. Rezolvarile vor fi completate in Google Form-ul asociat:
   (a) **Grupele 131 - 132**: https://forms.gle/1QP1sAvtxRrASLb1A
   (b) **Grupele 133 - 134**: https://forms.gle/YpyeeHySRFMP7s5A6
4. Este permis accesul la orice fel de materiale, insa orice incercare de fraudare atrage, dupa sine, notarea examenului cu nota finala 4 si sesizarea *Comisiei de etica* a *Universitatii din Bucuresti*.
5. In cazul suspiciunilor de frauda, studentii vizati vor participa si la o examinare orala.
6. In timpul testului de laborator, toate intrebarile privind subiectele vor fi puse pe Teams, pentru a avea toata lumea acces la intrebari si raspunsuri.

2 Subiectul I - Implementare (3.5 p)

2.1 Problema 1 (1p / 3.5p)

Sa se implementeze **procedura recursiva** `sumOfDigits` care primeste ca argument un numar natural x si care returneaza, prin intermediul registrului `$v0`, suma cifrelor numarului x.

2.2 Problema 2 (1p / 3.5p)

Sa se implementeze procedura `divisors` care, primind ca argumente doua numere naturale x si y, afiseaza toti divizorii produsului $x \cdot y$ doar daca suma cifrelor numarului $x \cdot y$ este para. Pentru implementare, se va face un apel imbricat din `divisors` la `sumOfDigits`.

2.3 Problema 3 (1.5p / 3.5p)

Fie dat un *array* `v` in memorie, continand elemente intregi. Sa se apeleze procedura `divisors` pentru fiecare doua elemente `v[i]` si `v[i+1]`; pentru ultimul element din *array*, `v[i+1]` va fi considerat ca fiind `v[0]`.

Important

1. Procedurile vor fi implementare respectand conventiile prezentate in cadrul laboratorului, referitoare la constructia cadrului de apel si la restaurarea registrilor.
2. Pentru implementarea corecta a problemei, fara utilizarea procedurilor, se acorda cel mult 1.5p.

3 Subiectul II - Analiza codului si intrebari generale (2.5p)

Fie urmatorul program, dezvoltat in limbajul de asamblare MIPS.

```
.data
    x: .space 4
    string1: .asciiz "string1 "
    string2: .asciiz "string2 "
    string3: .asciiz "string3 "
.text

main:
    la $t0, main
    jal f
    addi $t0, $t0, 20
    jr $t0
    li $v0, 4
    la $a0, string1
    syscall
    li $v0, 4
    la $a0, string2
    syscall
    li $v0, 4
    la $a0, string3
    syscall
    li $v0, 10
    syscall

f:
    subu $sp, 4
    sw $fp, 0($sp)
    addiu $fp, $sp, 4
    li $1, 4
    li $t0, 0
    add $t0, $t0, $1
    sw $t0, x
    lw $fp, -4($fp)
    addu $sp, 4
    jr $ra
```

Programul de mai sus nu functioneaza corect.

1. **0.6p** Care sunt modificarile ce trebuie efectuate pentru a elimina erorile? Explicati. Programul trebuie sa afiseze la final **string1 string 2 string3** si sa **memoreze** in **x** valoarea **4**. Nu se accepta modificari care sa schimbe structura programului.

2. **0.6p** Realizati o modificare pe o singura linie de cod astfel incat sa se afiseze, la *standard output*, rezultatul **string2 string3**. Explicati. Atentie la pseudoinstructiuni!

3. **0.6p** Fie o instructiune reprezentata in cod masina, avand campul de operatie egal cu `000000`, un cod de functie pe 5 biti valid; in plus, nu se efectueaza shiftare. Putem afirma ca instructiunea este valida? Dar daca impunem ca toti registrii sa aiba o codificare binara diferita de `00000`? Explicati.

4. **0.7p** Sa se scrie o secventa scurta in MIPS care sa provoace un *stack overflow*. Utilizati un numar minim de registri si explicati cum ati gandit.