Arhitectura sistemelor de calcul

Numele: .....

Data: ..... Grupa: ..... E-mail: .....

1. Fie $x = 109$ şi $y = 36$.

a) Convertiţi $x$ şi $y$ în baza 2.

b) Calculaţi $x - y$ lucrând în baza 2.

c) Convertiţi rezultatul din baza 2 în baza 8 şi din baza 2 în baza 16, direct (fără a trece prin baza 10).

d) Convertiţi rezultatul din baza 16 în baza 10.

e) Calculaţi $z = x - y$ folosind reprezentarea în complement faţă de 2 pe 8 biţi. Se vor explicita reprezentările lui $x$ şi $y$, complementul faţă de 1 şi cel faţă de 2 al reprezentării lui $y$, obţinerea reprezentării lui $z$, interpretarea acesteia ca număr în baza 10.

f) Determinaţi reprezentarea internă ca single a lui $t = -36$, binară (32 biţi) şi hexa (8 cifre hexa).

2. Fie $f : B_2^3 \longrightarrow B_2^2$, $f(x, y, z) = (f_1(x, y, z), f_2(x, y, z))$, unde:
$f_1(x, y, z) = \bar{z} + xy$,
$f_2(x, y, z) = \bar{x}z + xy$.

a) Construiţi tabelul de valori al lui $f$ şi scrieţi $f_1$, $f_2$ în FND şi FNC.

b) Implementaţi $f$ printr-un PROM.

c) Implementaţi $f$ printr-un codificator.

d) Implementaţi $f$ printr-un circuit cu două multiplexoare (câte unul pentru $f_1$, $f_2$) cu aceiaşi selectori $x, y, z$.

e) Implementaţi $f$ printr-un circuit care conţine doar multiplexori elementari; apoi, reduceţi la maximum numărul multiplexorilor elementari (nu este permisă adăugarea de porţi NOT).

f) Construiţi un circuit 1-DS care citeşte unul câte unul o secvenţă de biţi şi, de fiecare dată, scoate 1 / 0, după cum $f_2(x, y, z) = 1 / 0$, unde $x, y, z$ sunt antepenultimul, penultimul, respectiv ultimul bit citit.

3. a) Scrieţi un cod în limbaj de asamblare MIPS echivalent cu următorul cod în limbajul C:

```
if(x >= y) { if( x != y) z = 1;} else z = 2;
```

Se presupune că `x`, `y`, `z` sunt variabile întregi cu semn, alocate pe câte un word (`.word`) iar comparaţiile se fac cu semn.

În acest scop, completaţi codul de mai jos la un program întreg:

```
.data
  x: .word 20
  y: .word 10
  z: .space 4
.text
 main:
 ..........
 li $v0, 10
 syscall
```

(programul va face ca variabila `z` să conţină valoarea `1`).

b) Scrieţi în limbaj de asamblare MIPS o procedură `PATRAT` care primeşte ca parametri prin stivă o valoare întreagă şi adresa unei variabile word şi pune în acea variabilă pătratul întregului respectiv. Valoarea şi variabila sunt întregi de tip word (`.word`). Funcţia îşi va accesa parametrii cu $fp iar apelurile vor respecta convenţiile MIPS şi C (privind cadrul de apel, $fp, regiştrii salvaţi de apelant şi apelat, etc.).

În acest scop, completaţi codul de mai jos la un program întreg:

```
..........
x: .space 4
..........
li $t0, 2
la $t1, x
subu $sp, 8
sw $t1, 4($sp)
sw $t0, 0($sp)
jal PATRAT
addu $sp, 8
..........
```

(programul va face ca variabila `x` să conţină valoarea `4`).

Programele de la a) şi b) se vor ataşa tezei ca nişte fişiere text cu extensia 'txt' sau 's'. Rezolvarea va fi punctată doar dacă programul se poate compila (chiar dacă rezolvă incomplet sau greşit cerinţa).