Arhitectura sistemelor de calcul

Numele: .....

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

1. Fie $x = 125$ şi $y = 78$.

a) Convertiţi x şi y în baza 8.

b) Convertiţi mai departe x şi y din baza 8 în baza 2 şi din baza 2 în baza 16, direct (fără a trece prin baza 10).

c) Calculaţi $x + y$ lucrând în baza 8 şi $x - y$ lucrând în baza 16.

d) Convertiţi rezultatele din bazele 8, respectiv 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) Interpretaţi ca număr în baza 10 reprezentarea internă hexa în format single OxC29A0000.

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:
$f_1(x, y, z) = \bar{x}\,\bar{z} + (\bar{y} \oplus z)$,
$f_2(x, y, z) = (\bar{x} + y)(\bar{y} \oplus z)$.

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.

g) Implementaţi poarta NAND printr-un circuit care conţine doar porţi NOR.

3. Considerăm implementarea procesorului MIPS cu 1 ciclu / instrucţiune (vezi verso). Fie fragmentul de program:

```
li $t1, 1              et:
li $t2, 0xFFFF         sll $t2, $t2, 16
li $t3, -1             sub $t2, $t2, $t1
                       slt $t3, $t2, $t3
                       beq $t1, $t3, et
```

Presupunem că în memorie instrucţiunea `slt` din program are adresa α.

a) Pentru instrucţiunile `slt` şi `beq` din program scrieţi câmpurile din reprezentarea lor internă (ex: op/rs/rt/imm, valorile se scriu hexa); pentru `beq` din program scrieţi şi reprezentările ei binară (32 biţi) şi hexa (8 cifre hexa).

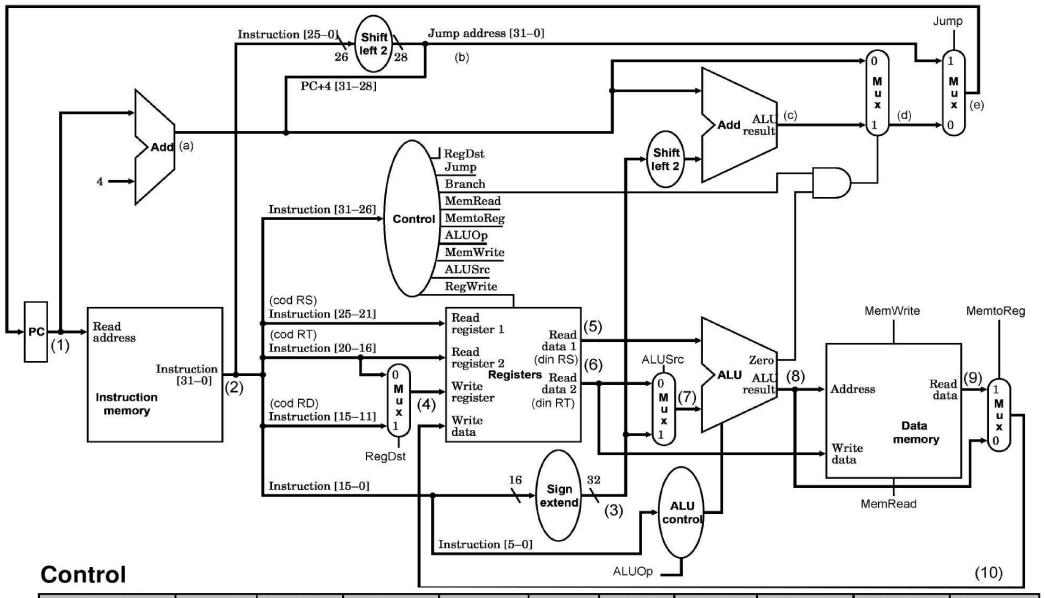
b) Completaţi tabelul următor cu valorile obţinute la prima executare a instrucţiunilor `slt` şi `beq` din program; valorile se vor scrie hexa/formulă, iar dacă valoarea este necunoscută/nedefinită se va nota "?"; în coloanele PC şi $t3 se vor trece valorile noi, de la sfârşitul executării instrucţiunilor respective:

	1	4	5	7	8	ALU zero	10	(d)	(e)	Branch	Mem To Reg	ALU Op (2b)	ALU Ctrl (3b)	Reg Write	PC	$t3
Iniţial	—	—	—	—	—	—	—	—	—	—	—	—	—	—	α	FFFFFFFF (-1)
slt	α															
beq																

c) Adăugaţi procesorului implementarea instrucţiunii: `msw rs, rt`
care scrie în memorie la adresa obţinută calculând diferenţa valorilor din regiştrii `rs` şi `rt` valoarea aflată în registrul `rt` (adică efectuează `mem[rs - rt] := rt`); se va adapta formatul I, cu op = - (nerelevant). Pentru implementare, este suficientă adăugarea unei linii cu valori `0, 1, X` tabelului "Control":

Instruction [31 - 26]	RegDst	ALU Src	Mem To Reg	Reg Write	Mem Read	Mem Write	Branch	Jump	ALU Op1	ALU Op0
—										

Implementarea cu un ciclu pe instructiune



Control

	Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	Jump	ALUOp1	ALUp0
0x0	R-format	1	0	0	1	0	0	0	0	1	0
0x23	lw	0	1	1	1	1	0	0	0	0	0
0x2b	sw	X	1	X	0	0	1	0	0	0	0
0x4	beq	X	0	X	0	0	0	1	0	0	1
0x2	j	X	X	X	0	0	0	0	1	X	X

ALU Control

	ALUOp		Camp functie						Operatie	
	$ALUOp_1$	$ALUOp_0$	F5	F4	F3	F2	F1	F0		
lw/sw	0	0	X	X	X	X	X	X	010	(+)
beq	X	1	X	X	X	X	X	X	110	(-)
add	1	X	X	X	0	0	0	0	010	(+)
sub	1	X	X	X	0	0	1	0	110	(-)
and	1	X	X	X	0	1	0	0	000	(and)
or	1	X	X	X	0	1	0	1	001	(or)
slt	1	X	X	X	1	0	1	0	111	(slt)

R-format

ALU Operation

ALU control input	Function
000	and
001	or
010	add
110	subtract
111	set on less than

```
add/sub/slt rd,rs,rt # rd := rs+rt, rd := rs-rt, rd := (rs<rt)?1:0
# |   0    |   rs    |   rt    |    rd    |   0   | 0x20/0x22/0x2a |
# --------------------------------------------------------------------
#  31-26 25-21 20-16 15-11  10-6   5----------0
#    6 b    5 b     5 b     5 b     5 b      6 b
```

```
lw/sw rt,imm(rs) # rt :=/=: mem[(rs)+imm]
  # | 0x23/0x2b |   rs    |   rt    |    imm    |
  # ---------------------------------------------------
  #   31-------26  25-21 20-16   15---0
  #         6 b        5b       5b       16 b
```

```
beq rs,rt,et
# if rs=rt then goto et
# if rs=rt then PC:=PC+4+imm*4 else PC:=PC+4
# |  0x4  |   rs    |    rt    | imm= (et-PC-4)/4  |
# ------------------------------------------------------
#  31-26  25-21 20-16 15------------------0
#     6 b      5 b     5 b              16 b
```

```
j et
# goto et
# PC:=(PC+4) & 0xf0000000 + imm*4
# |  0x2  |     imm     |
# -------------------------
#  31-26  25-------0
```

Registri: $t0 (8) - $t7 (15)