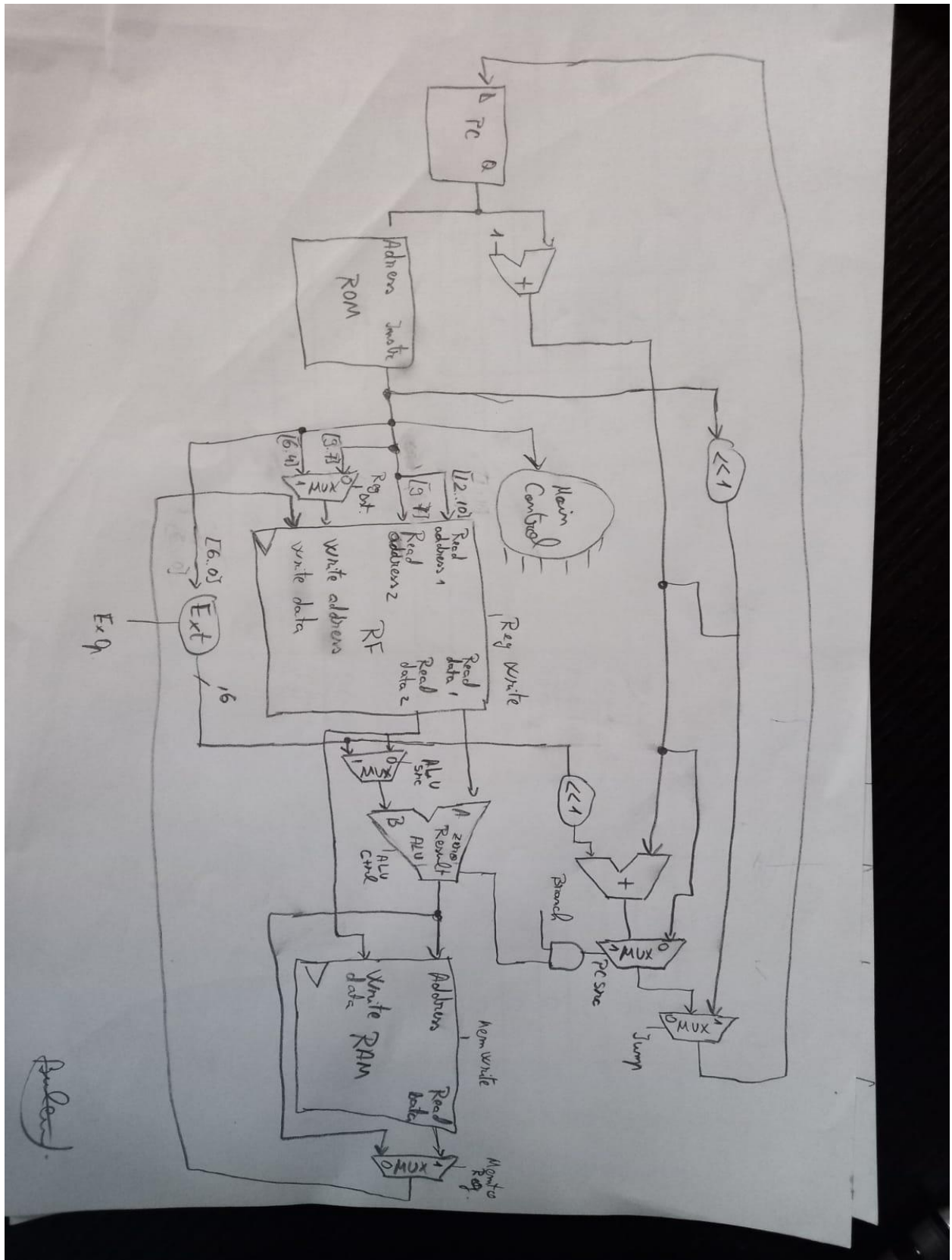


Calea de date:



A) Cele 4 instructiuni alese sunt:

Xor, nand, andi si ori

7) xor \$1, \$4, \$2
xor \$d, \$s, \$t

function: 110

=> 000-100-010-001-0-110

RTL: $RF[d] \leftarrow RF[s] \wedge RF[t]$.

8) NAND.

nand \$3, \$1, \$2

nand \$d, \$s, \$t.

function: 111

=> 000-001-010-011-0-111.

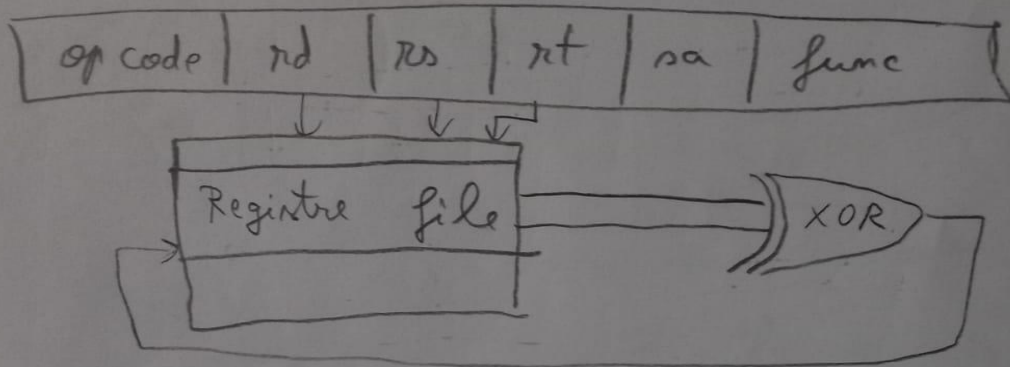
RTL: $RF[d] \leftarrow RF[s] \text{ nand } RF[t]$.

pag 3.

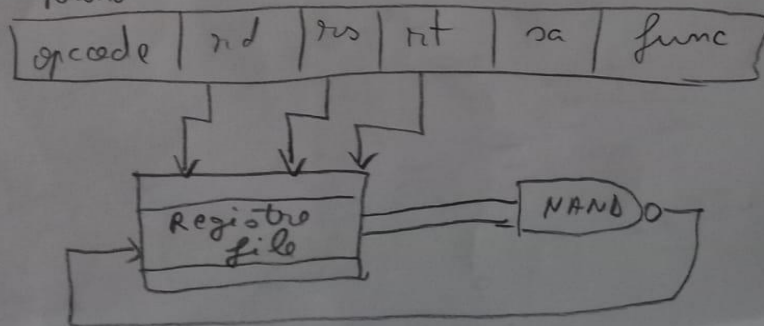
pag 2

pag 10
pag 11
pag 12

6) xor



7) Nand



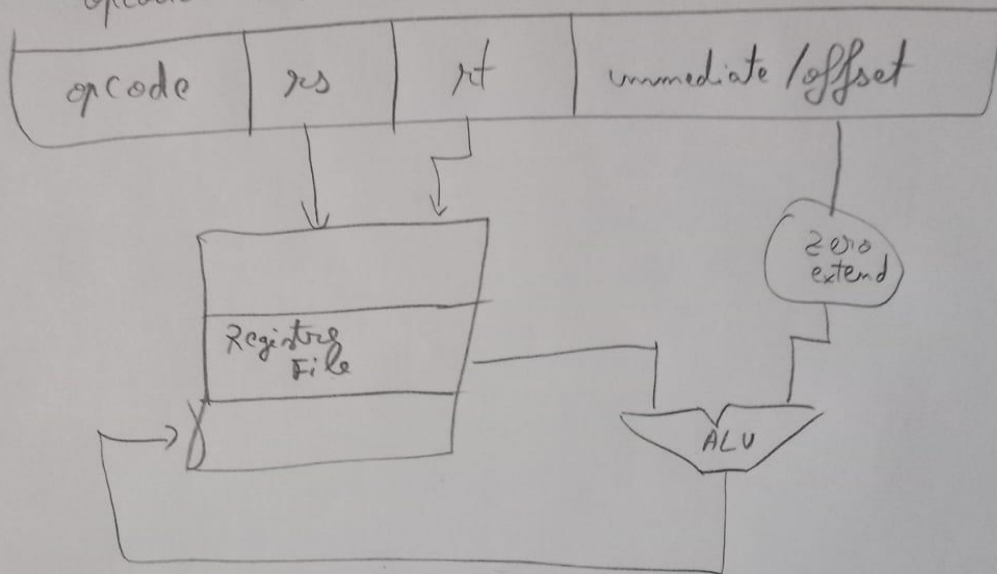
Handwritten notes at the bottom of the page include:

- $010-011-0$
- $RF[ra]$
- $\$2$
- $\$t$
- $RF[ra]$
- $0-0-110$
- $RF[ra]$
- Ent

Butant

51 andi \$t, \$0, imm
andi \$1, \$0, 3

opcode = 101



101-001-000-0000011

RTL: $RF[rt] \leftarrow RF[rs] \& Z_Ext(imm)$

6.) ori \$t, \$s, imm

ori \$1, \$5, 7

opcode: 110

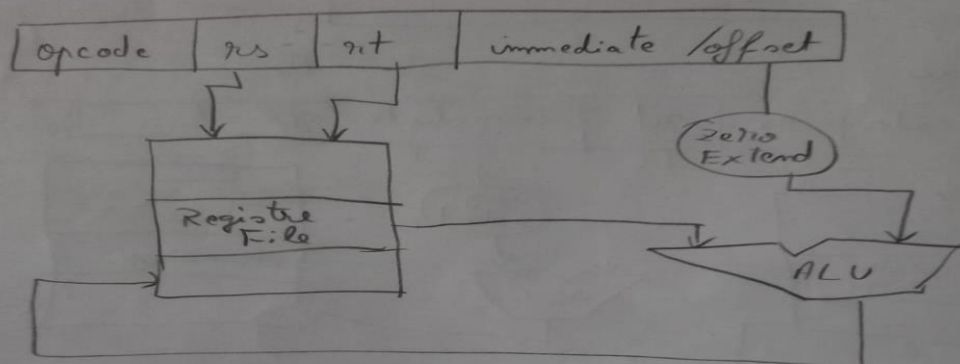
=> 110-101-001-0000111

RTL: $RF[rt] \leftarrow RF[rs] \mid Z_Ext(imm)$

pag 6

pag 2

④ ORI



pag

0-111
and $RF[rt]$

3

B) Tabelul cu valorile semnalelor de control pentru toate instructiunile

Instruction	Regdst	Reg write	ALU src	Ext op	ALU op	ALU ctrl	Mem write	Memto Reg	Branch	Jump
add	1	1	0	0	000	000(+)	0	0	0	0
sub	1	1	0	0	000	001(-)	0	0	0	0
sll	1	1	0	0	000	010(<<)	0	0	0	0
srl	1	1	0	0	000	011(>>)	0	0	0	0
and	1	1	0	0	000	100 (and)	0	0	0	0
or	1	1	0	0	000	101 (or)	0	0	0	0
xor	1	1	0	0	000	110 (xor)	0	0	0	0
lmand	1	1	0	0	000	111 (lmand)	0	0	0	0
addi	0	1	1	1	001	000(+)	0	0	0	0
lwi	0	1	1	1	001	000(+)	0	1	0	0
swi	0	0	1	1	001	000(+)	1	0	0	0
beg	0	0	0	1	010	001(-)	X	0	1	0
andi	0	1	1	0	101	100 (and)	X	0	0	0
ori	0	1	1	0	110	101 (or)	0	0	0	0
J J	X	0	X	X	xxx	xxx	0	X	X	1

Riley

C) Descrierea codului

functia este:

Bulex.

```
int A[10] = {5, 5, 5, 5, 5, 5, 5, 5, 5, 5};
```

```
int x = 100;
```

```
for (int i = 0; i < 10; i++)
```

```
{ A[i] = A[i] + 1;
```

```
  x = x - A[i];
```

```
}
```

Cod de asamblare.

Rule:

- 0: add \$1, \$0, \$0 $i = 0$
- 1: addi \$4, \$0, 10 : se salvează nr de iterații $\$4 = 10$
- 2: add \$2, \$0, \$0 : inițializarea indexului locației de memorie
- 3: addi \$5, \$0, 100 : $x = 100$
- 4: beg \$1, \$4, 7 : verifică dacă $i = 10$
- 5: lw \$3, 40(\$2) : în $\$3$ se aduce elementul curent din sir
 $\$3 = A[0] = 5$
- 6: addi \$3, \$3, 1 : $\$3 = \$3 + 1 = 6$
- 7: sw \$3, 40(\$2) : se salvează noua valoare a lui $\$3$ în
elementul curent din sir. $A[0] = \$3 = 6$
- 8: sub \$5, \$5, \$3 : $x = x - \$3$ ($\Rightarrow 100 = 100 - 6$)
- 9: addi \$2, \$2, 1 : indexul următorului element din sir.
- 10: addi \$1, \$1, 1 : $i = i + 1$
- 11: j 4 : salt la începutul buclei.
- 12: sw \$5, 60(\$0) : salvarea sumei în memorie
la adresa 60.

Buland

4.

Se presupune că sirul A se află în memoria la adresa A-addr iar următoarele la adrese consecutive din 2 în 2 octeți (fiecare element este întreg pe 16 biți).

- variabila x se află în memorie la adresa x-addr dar este stocată temporar în \$5
- registrul \$2 este folosit ca index.
- i este reprezentat de registrul \$1:

- A-addr: adresa sirului A.

- \$5: adresa x

- \$2: folosit ca index.

- \$1: i

- \$4: Numărul de iterații

- A are 10 locații sau câte 2 octeți fiecare.

- se presupune că A are adrese de început
 $A\text{-addr} = 40$ iar variabila x se află în memorie imediat după sir, $x\text{-addr} = 60$

- begin-loop se află la pseudo-adresa = 4

- end-loop = 7 (12-5) unde 5 este indexul instrucțiunii lui.

Cod Maxima.

Bulen.

- 0: 000_000_000_001_000
- 1: 001_000_100_0001010
- 2: 000_000_000_010_000
- 3: 001_000_101_1100100
- 4: 010_001_100_0000111
- 5: 011_010_011_0101000
- 6: 001_011_011_0000001
- 7: 100_010_011_0101000
- 8: 000_101_011_101_001
- 9: 001_010_010_0000010
- 10: 001_001_001_0000001
- 11: 101_0000000000100
- 12: 100_000_101_0111100

D) Trasarea executiei programului

- 0) add \$1, \$0, \$0 : RD1 = 0, RD2 = 0, ALURes = 0
- 1) addi \$4, \$0, 10 : RD1 = 0, Ext-Imm = 10, ALURes = 10
- 2) add \$2, \$0, \$0 : RD1 = 0, RD2 = 0, ALURes = 0
- 3) addi \$5, \$0, 100 : RD1 = 0, Ext-Imm = 100, ALURes = 100
- 4) beq \$1, \$4, 7 : RD1 = 10, Ext-Imm = 7, Branch = 0
- 5) lw \$3, 40(\$2) : RD1 = 0, Ext-Imm = 40, Address = 40, ALURes = 40
- 6) addi \$3, \$3, 1 : RD1 = 5, Ext-Imm = 1, ALURes = 6
- 7) sw \$3, 40(\$2) : RD1 = 0, Ext-Imm = 6, ALURes = 6
- 8) sub \$5, \$5, \$3 : RD1 = 100, RD2 = 6, ALURes = 94
- 9) addi \$2, \$2, 1 : RD1 = 0, Ext-Imm = 1, ALURes = 1
- 10) addi \$1, \$1, 1 : RD1 = 0, Ext-Imm = 1, ALURes = 1
- 11) j 4 : Jump = 1, Address = 4
- 12) sw \$5, 60(\$0) : RD1 = 0, Ext-Imm = 60, ALURes = 60

Pulcinella

