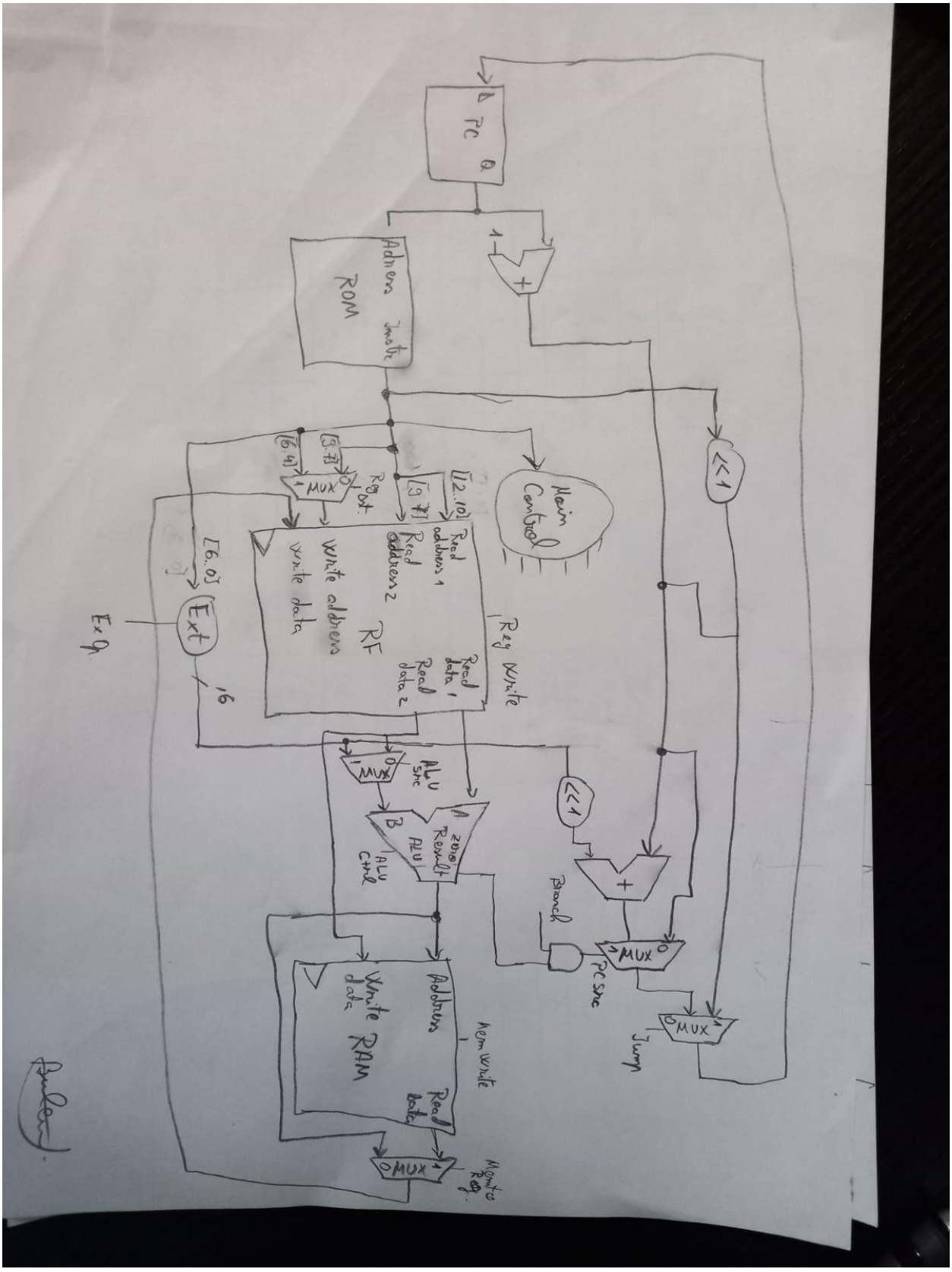
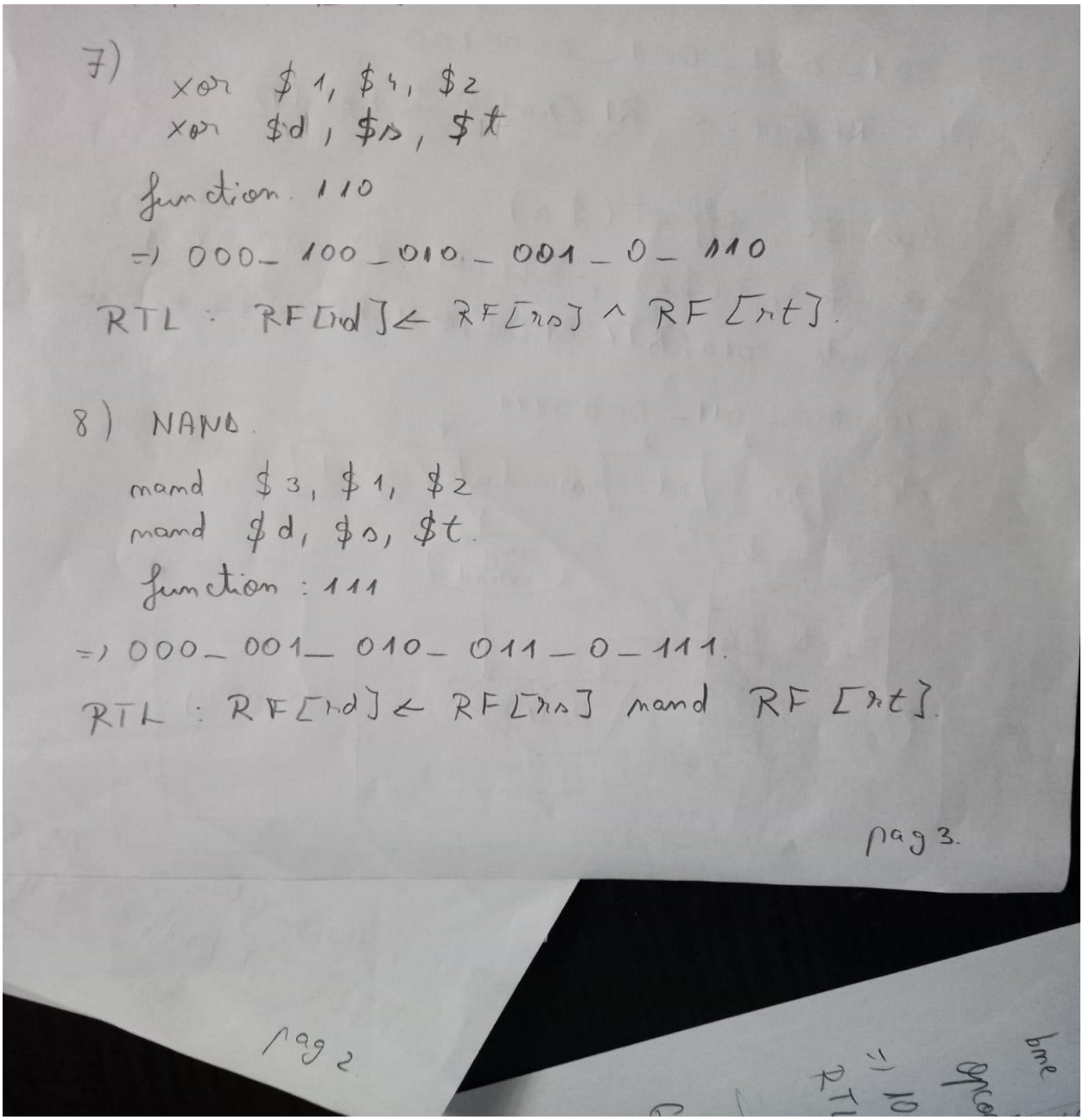
Calea de date:

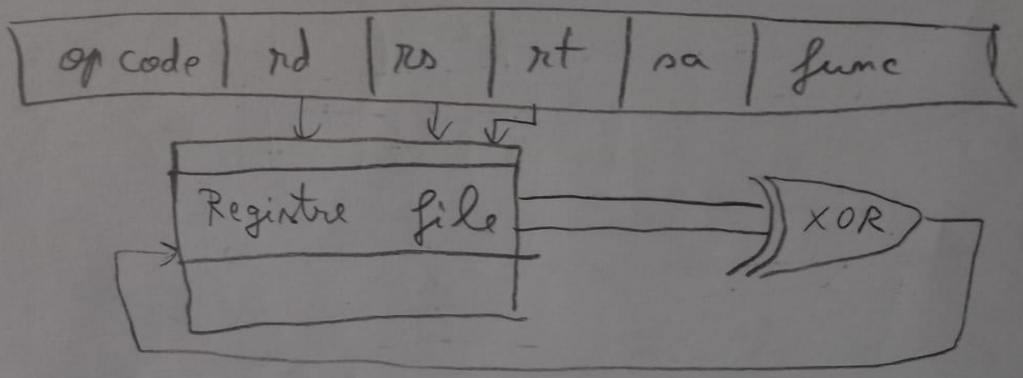


A) Cele 4 instructiuni alese sunt:

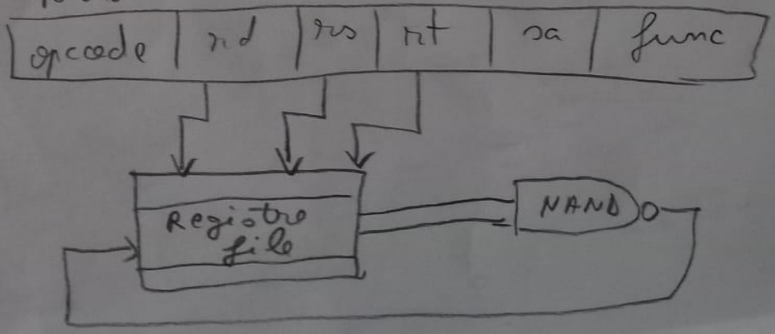
Xor, nand, andi si ori



6) XOR


op code
rd
rs
rt
sa
func
Registre file
XOR

7) Nand

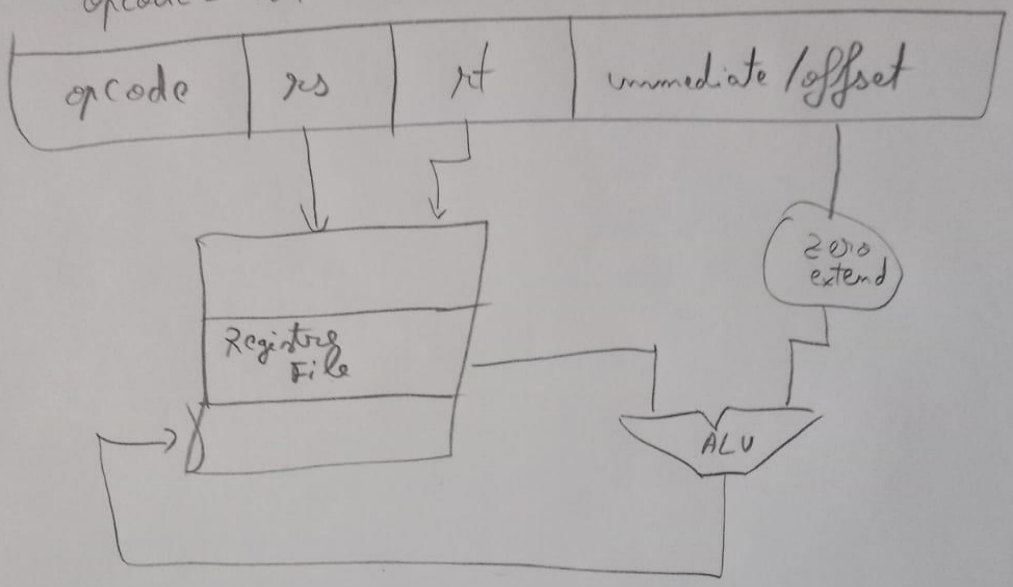

opcode
rd
rs
rt
sa
func
Registre file
NAND

Bulent

5) andi $t, $s, imm

andi $ 1, $ 0, 3

opcode = 101



101_001_000_00000 11

RTL : RF [rt] ← RF [rs] & Z_Ext (imm)

6.) ori $t, $s, imm

ori $1, $5, 7

opcode 110

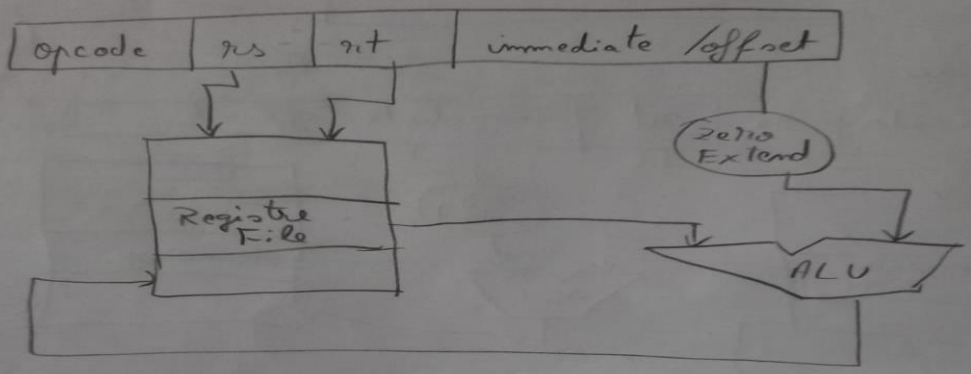
=> 110_101_001_0000111

RTL: RF[rt] ← RF[rs] | Z_Ext(imm)

pag 6

pag 2

④ ORi

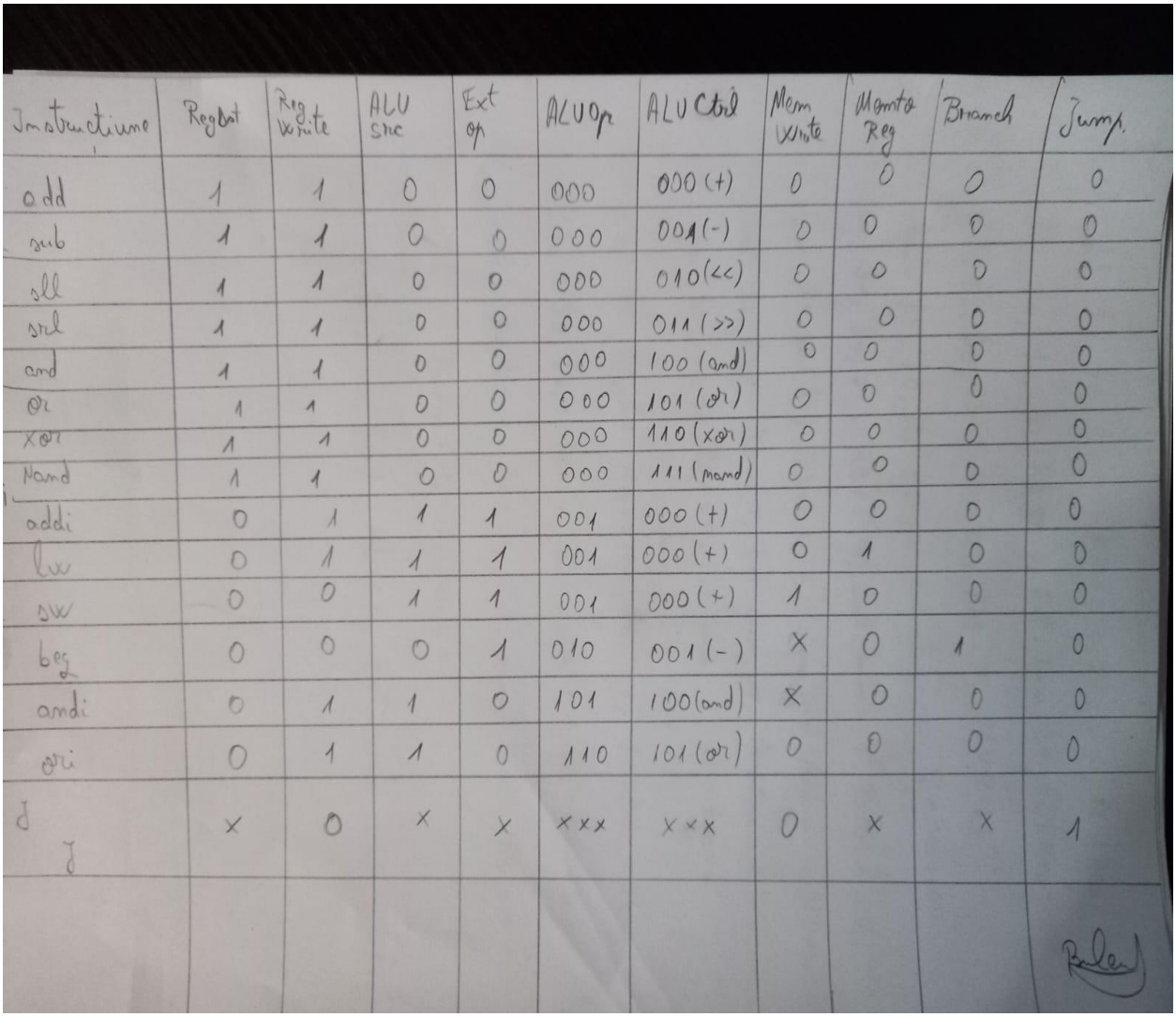


pag

0_111

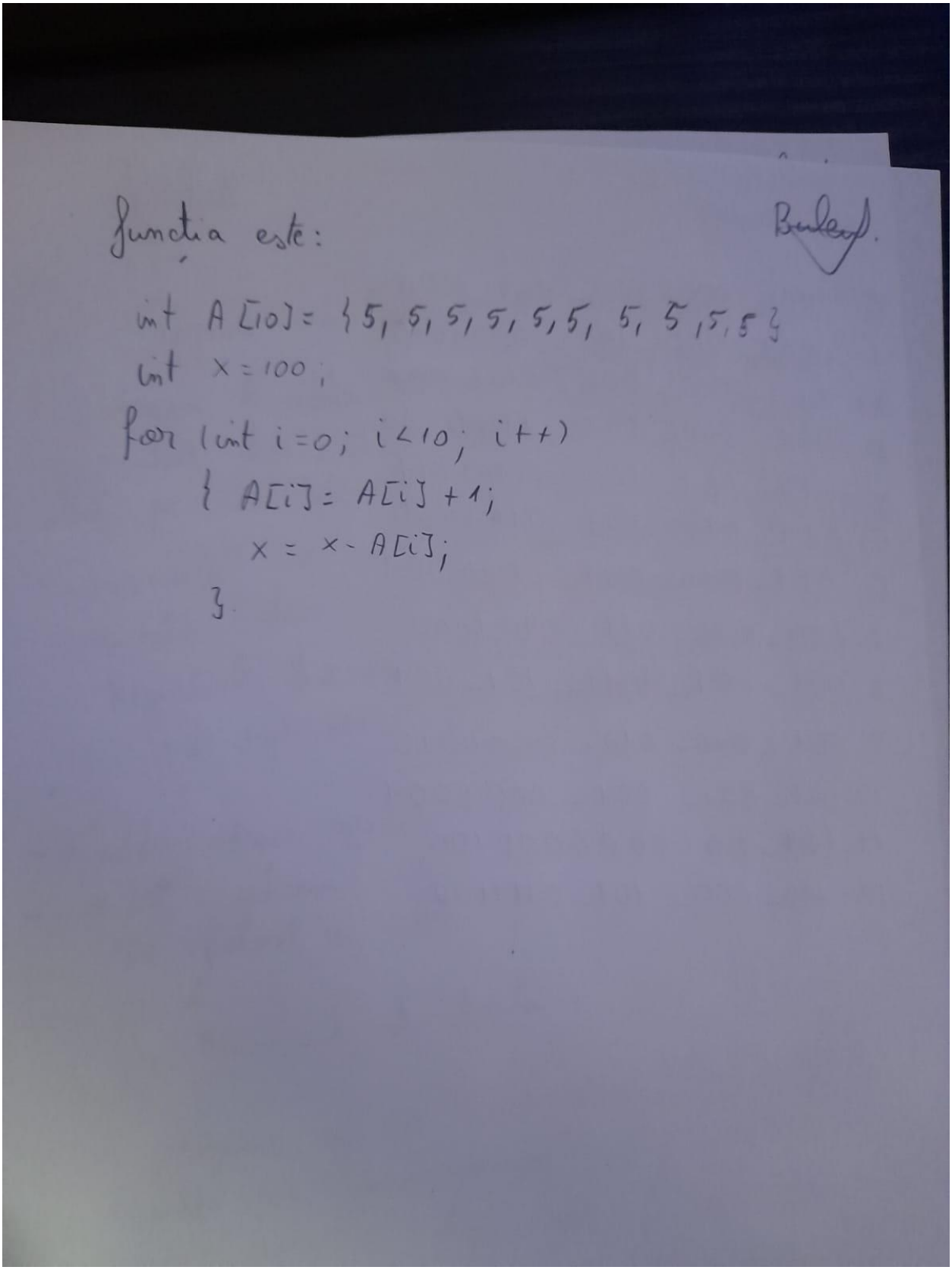
and RF[rt]

B) Tabelul cu valorile semnalelor de control pentru toate instructiunile



Instructiune	RegDst	Reg Write	ALU src	Ext op	ALUOp	ALUCtrl	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
Nand	1	1	0	0	000	111 (nand)	0	0	0	0
addi	0	1	1	1	001	000 (+)	0	0	0	0
lw	0	1	1	1	001	000 (+)	0	1	0	0
sw	0	0	1	1	001	000 (+)	1	0	0	0
beq	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	X	0	X	X	XXX	XXX	0	X	X	1

C) Descrierea codului



Cod de asamblare.

Rulem.

```
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:  beq $1, $4, 7       : verifică dacă i=10
5:  lw $3, 40($2)       : în $3 se aduce elementul curent din șir
                          $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 șir. A[0] = $3 = 6
8.  sub $5, $5, $3      : x = x - $3 (=) 100 = 100 - 6
9.  addi $2, $2, 1      Indexul următorului element din șir.
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.
```

Barbu.

4.

Se presupune că șirul 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 șirului A.
- @$5: adresa x
- $2: folosit ca index.
- $1: i
- $4: Numărul de iterații
- A are 10 locații de câte 2 octeți fiecare.
- se presupune că A are adresa de început A_addr = 40 iar variabila x se află în memorie imediat după șir, x_addr = 60
- begin_loop se află la pseudo-adresa = 4
- end_loop = 7 (12 - 5) unde 5 este indexul instrucțiunii lw.

Cod Mașină

Butea.

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_000 0001
11: 101_00000000000100
12: 100_000_101_0111100

D) Trasarea executiei programului