Procesorul MIPS, ciclu unic
– versiune pe 16 biți –

Student: Marcu Ariana-Mălina

Grupa: 30222

1**. Instrucțiuni alese suplimentar**

- *Instrucțiunea xor(Bitwise Exclusive OR)*

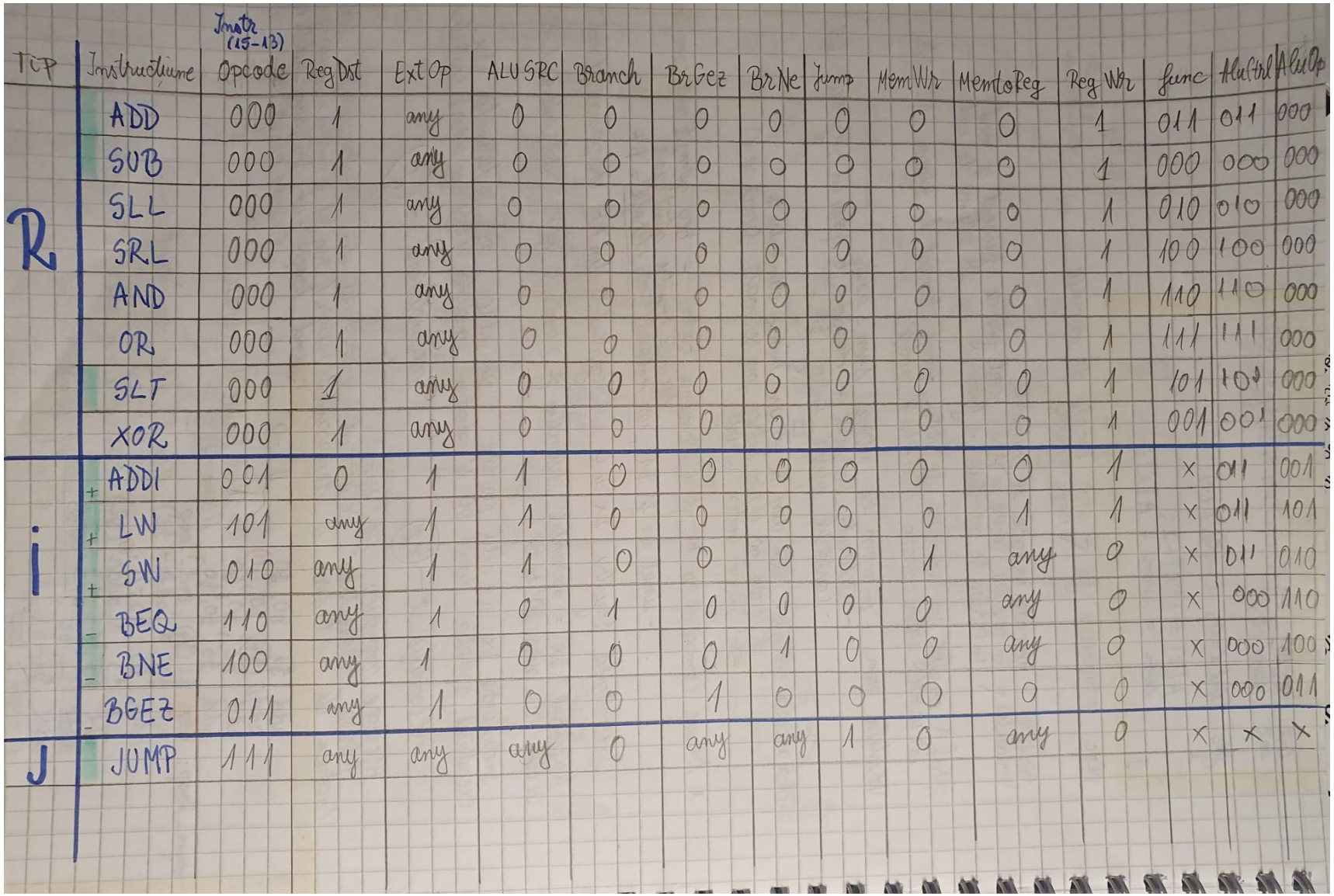
  - instrucțiune de tip R
  - realizează operația de sau-exclusiv între conținutul a două registre
  - xor $d, $s, $t
  -

- *Instrucțiunea slt (Set on Less Than)*
  - instrucțiune de tip R
  - setează registrul destinație atunci când conținutul primului registru sursă este mai mic decât conținutul celui de-al doilea registru sursă
  - slt $d, $s, $t

- *Instrucțiunea bgez (Branch on Greater Than or Equal to Zero)*
  - instrucțiune de tip I
  - efectuează un salt condiționat la o adresă relativă la adresa instrucțiunii următoare dacă registrul sursă are conținutul mai mare sau egal cu 0
  - sintaxa: bgez $s, offset

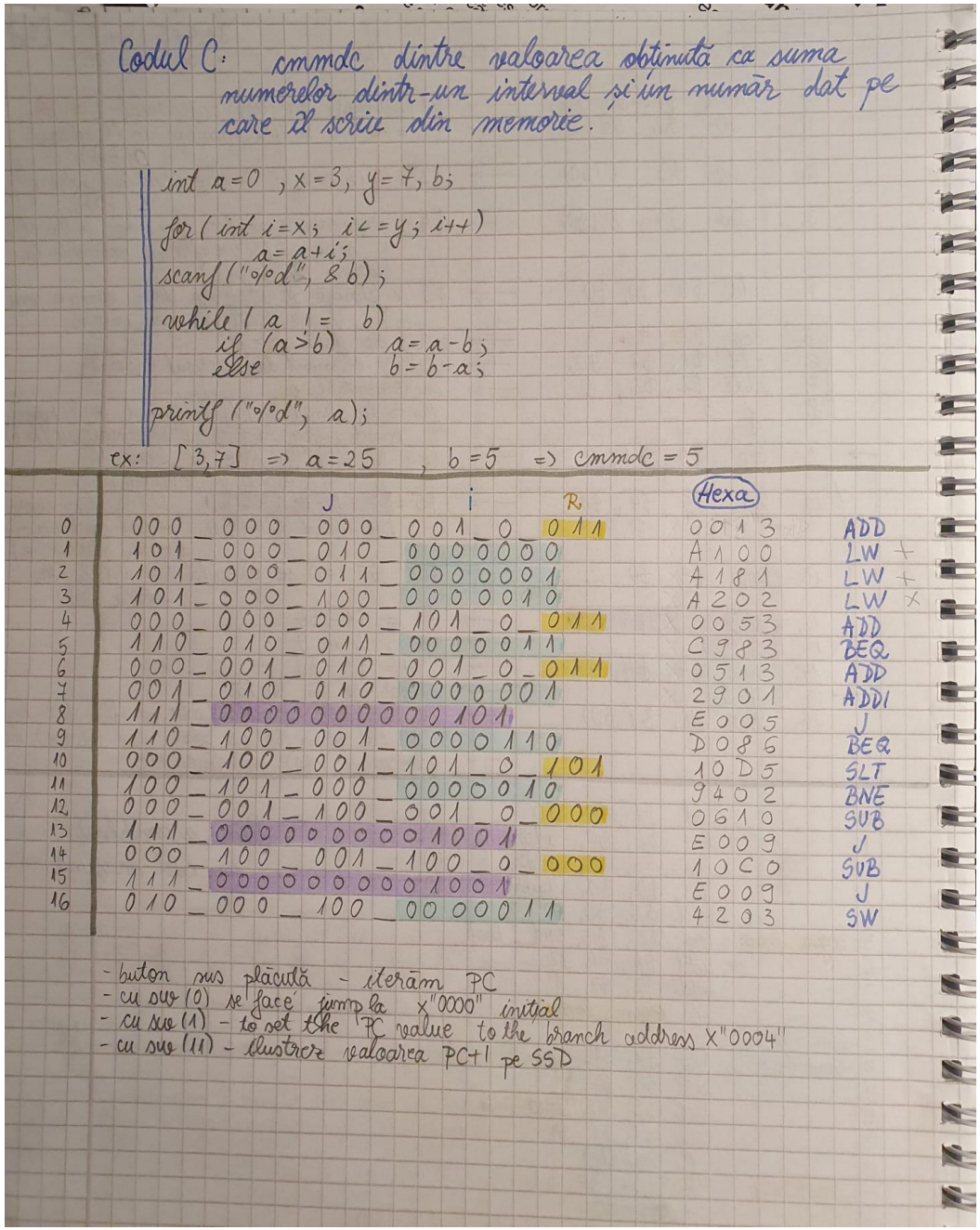
- *Instructiunea bne (Branch on not equal)*
  - instrucțiune de tip I
  - efectuează un salt condiționat la o adresă relativă la adresa instrucțiunii următoare dacă două registre sunt diferite
  - sintaxa: bne $s, $t, offset

2. Tabel cu valorile semnalelor de control pentru setul de instrucțiuni selectat



Tip	Instrucțiune	Opcode (Instr (15-13))	RegDst	ExtOp	ALUSrc	Branch	BrGez	BrNe	Jump	MemWr	MemtoReg	RegWr	func	AluCtrl	AluOp
R	ADD	000	1	any	0	0	0	0	0	0	0	1	011	011	000
	SUB	000	1	any	0	0	0	0	0	0	0	1	000	000	000
	SLL	000	1	any	0	0	0	0	0	0	0	1	010	010	000
	SRL	000	1	any	0	0	0	0	0	0	0	1	100	100	000
	AND	000	1	any	0	0	0	0	0	0	0	1	110	110	000
	OR	000	1	any	0	0	0	0	0	0	0	1	111	111	000
	SLT	000	1	any	0	0	0	0	0	0	0	1	101	101	000
	XOR	000	1	any	0	0	0	0	0	0	0	1	001	001	000
I	+ ADDI	001	0	1	1	0	0	0	0	0	0	1	x	011	001
	+ LW	101	any	1	1	0	0	0	0	0	1	1	x	011	101
	+ SW	010	any	1	1	0	0	0	0	1	any	0	x	011	010
	- BEQ	110	any	1	0	1	0	0	0	0	any	0	x	000	110
	- BNE	100	any	1	0	0	0	1	0	0	any	0	x	000	100
	- BGEZ	011	any	1	0	0	1	0	0	0	0	0	x	000	011
J	JUMP	111	any	any	any	0	any	any	1	0	any	0	x	x	x

3. Descrierea în cuvinte, cod C și cod mașină a programului încărcat în memoria ROM



Codul C: cmmdc dintre valoarea obținută ca suma numerelor dintr-un interval și un număr dat pe care îl scrie din memorie.

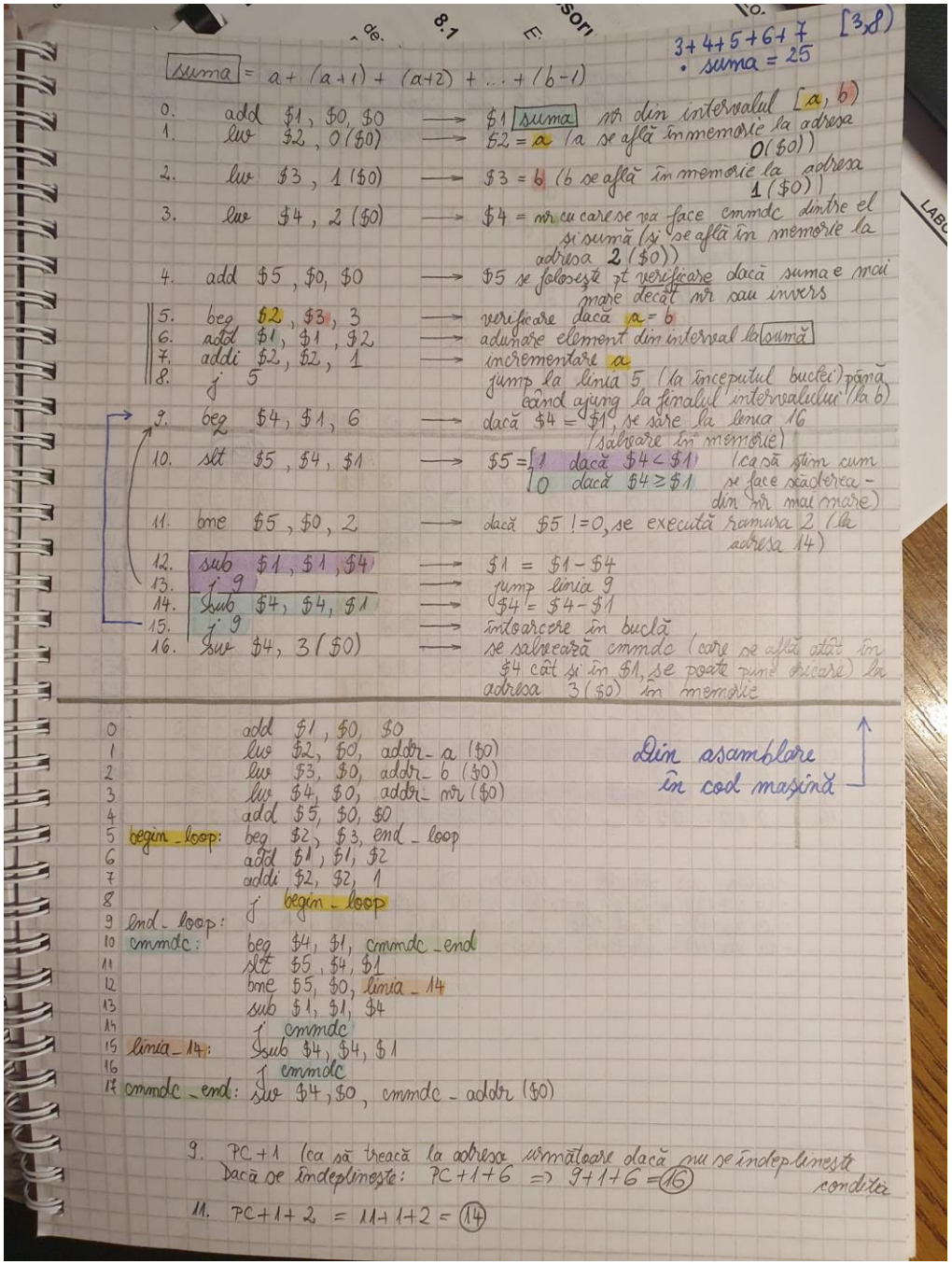
```
int a=0 , x=3, y=7, b;
for (int i=x; i<=y; i++)
    a= a+i;
scanf("%d", &b);
while ( a != b)
    if (a>b)    a=a-b;
    else        b=b-a;
printf("%d", a);
```

ex: [3,7] => a=25 , b=5 => cmmdc = 5

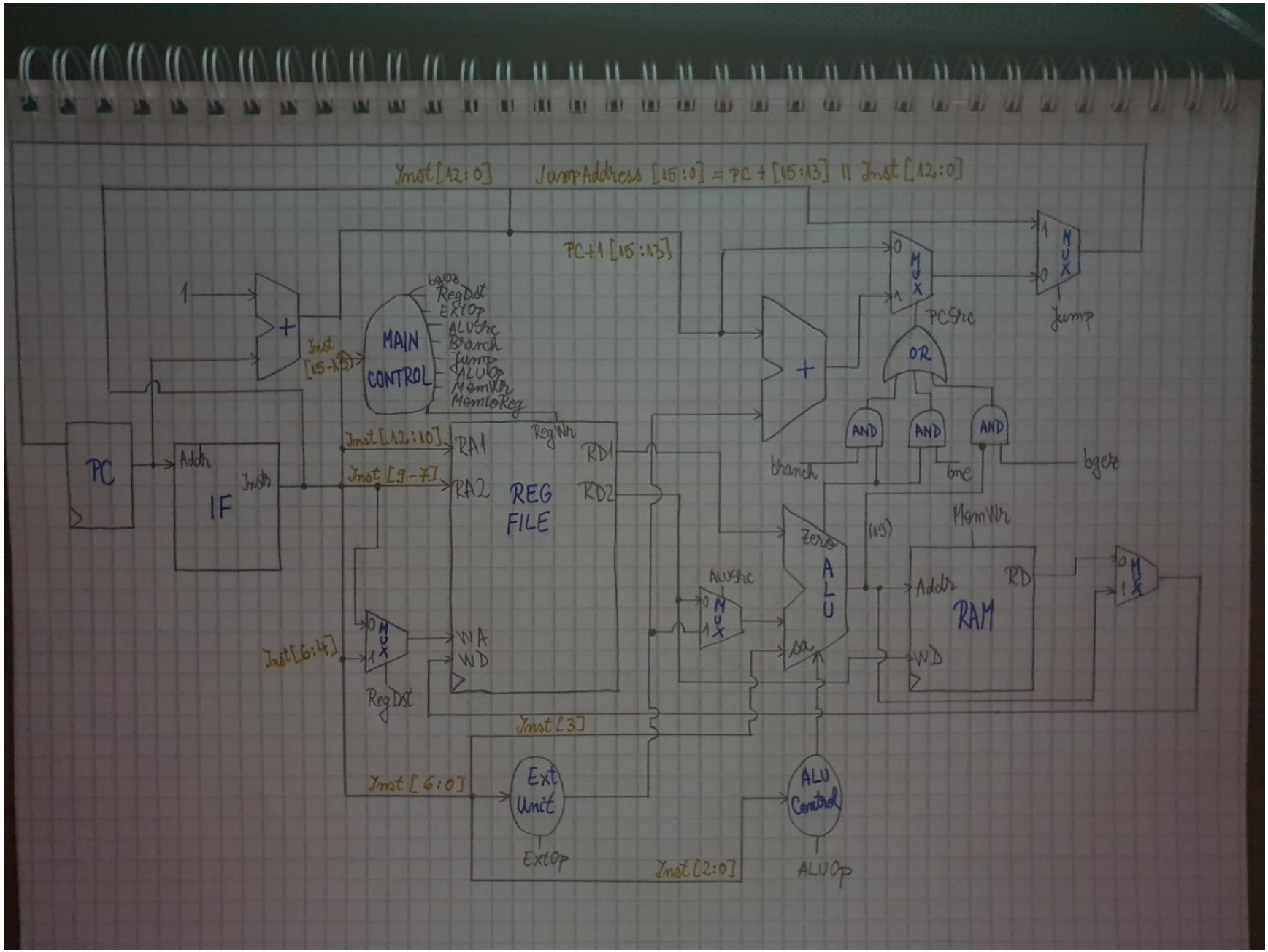
		J		i		Rs	Hexa	
0	000_	000_	000_	001_	0_	011	0013	ADD
1	101_	000_	010_	0000000			A100	LW +
2	101_	000_	011_	0000001			A181	LW +
3	101_	000_	100_	0000010			A202	LW x
4	000_	000_	000_	101_	0_	011	0053	ADD
5	110_	010_	011_	0000011			C983	BEQ
6	000_	001_	010_	001_	0_	011	0513	ADD
7	001_	010_	010_	0000001			2901	ADDI
8	111_	0000000000101					E005	J
9	110_	100_	001_	0000110			D086	BEQ
10	000_	100_	001_	101_	0_	101	10D5	SLT
11	100_	101_	000_	0000010			9402	BNE
12	000_	001_	100_	001_	0_	000	0610	SUB
13	111_	0000000001001					E009	J
14	000_	100_	001_	100_	0_	000	10C0	SUB
15	111_	0000000001001					E009	J
16	010_	000_	100_	00000011			4203	SW

- buton sus placuță - iterăm PC
- cu sus (0) se face jump la x"0000" initial
- cu sus (1) - to set the PC value to the branch address x"0004"
- cu sus (11) - afișăm valoarea PC+1 pe SSD

4. Trasarea execuției programului



5. RTL schematic – in PDF-ul atașat + imaginea alaturata



6. Trasarea execuției programului de test pentru MIPS16

- sw(7:5) = 000 - se afișează instrucțiunea pe SSD
- sw(7:5) = 001 - se afișează următoare valoare secvențială a PC (PC + 1), pe SSD
- sw(7:5) = 010 - se afișează RD1 pe SSD
- sw(7:5) = 011 - se afișează RD2 pe SSD
- sw(7:5) = 100 - se afișează Ext_Imm pe SSD
- sw(7:5) = 101 - se afișează ALURes pe SSD
- sw(7:5) = 110 - se afișează MemData pe SSD
- sw(7:5) = 111 - se afișează WD pe SSD.

Valorile se completează în hexazecimal așa cum trebuie să apară pe SSD. Succesiunea pașilor reprezintă ordinea de execuție în timp la apăsarea butonului ENable. **Pasul 0 corespunde stării inițiale a circuitului (PC = 0), iar pasul *N* caracterizează starea după apăsarea de *N* ori a butonului ENable**. Inițial registrele vor avea valoarea 0 (care se atribuie automat în lipsa unei inițializări explicite a RF), iar memoria de date RAM poate fi inițializată cu valori dorite. Tabelul se completează pentru tot programul sau dacă are buclă până la finalul primei iterații.

Pas	**SW(7:5)**	“000”	“001”	“010”	“011”	“100”	“101”	“110”	“111”	**De completat numai pentru instrucțiuni de salt**	
	Instr (*în asamblare*)	**Instr** (*hexa*)	**PC+1**	**RD1(rs)**	**RD2(rt)**	**Ext_Imm**	**ALURes**	**MemData**	**WD**	**BranchAddr**	**JumpAddr**
0	add $1, $0, $0	X"0013"	X”0001”	X”0000”	X”0000”	-	X”0000”	X”0000”	X”0000”		
1	lw $2, 0($0)	X"A100"	X”0002”	X”0000”	X”0000”	X”0000”	X”0000”	X”0000”	X”0000”		
2	lw $3, 1($0)	X"A181"	X”0003”	X”0000”	X”0000”	X”0001”	X”0001”	X”0000”	X”0000”		
3	lw $4, 2($0)	X"A202"	X”0004”	X”0000”	X”0000”	X”0002”	X”0002”	X”0003”	X”0003”		
4	add $5, $0, $0	X"0053"	X”0005”	X”0000”	X”0000”	-	X”0000”	X”0000”	X”0000”		
5	beq $2, $3, 3	X"C983"	X”0006”	X”0003”	X”0008”	X”0003”	X”0003”	X”0008”	X”0003”	X”0009”	
6	add $1, $1, $2	X"0513"	X”0007”	X”0000”	X”0003”	-	X”0003”	X”0008”	X”0003”		
7	addi $2, $2, 1	X"2901"	X”0008”	X”0003”	X”0004”	X”0001”	X”0003”	X”0008”	X”0003”		
8	j 5	X"E005"	X”0009”	X”0000”	X”0000”	-	-	-	-		X”0005”

9	beq $4, $1, 6	X"D086"	X”000A”	X”0005”	X”0019”(25)	X”0006”	X”0005”	X”0000”	X”0005”	X”0010”	
10	slt $5, $4, $1	X"10D5"	X”000B”	X”0001”	X”0005”	-	X”0006”	-	X”0006”		
11	bne $5, $0, 2	X"9402"	X”000C”	X”0001”	X”0000”	X”0002”	X”0003”	X”0008”	X”0003”	X”000E”	
12	sub $1, $1, $4	X"0610"	X”000D”	X”0019”	X”0005”	-	X”0014”	-	X”0014”		
13	j 9	X"E009"	X”000E”	X”0000”	X”0000”	-	-	-	-		X”0009”
14	sub $4, $4, $1	X"10C0"	X”000F”	X”0005”	X”0019”	X”0000”	X”0014”	-	X”0014”		
15	j 9	X"E009"	X”0010”	X”0000”	X”0000”	-	-	-	-		X”0009”
16	sw $4, 3($0)	X"4203"	X”0011”	X”0000”	X”0005”(rez	X”0003”	X”0005”	X”0001”	X”0001”		

Programul a fost testat si functioneaza bine totul pana la InstructionDecode inclusiv, insa la testarea finala, de la primul beq, nu mai functioneaza corect, doar partial.