UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROCESORUL MIPS PIPELINE- 16 BIȚI

Student: Hădărău Elena

Grupa:30223

a) Cele 4 instrucțiuni suplimentare

1. Tipul R

- xor $rd, $rs, $rt RTL: RF[rd] ←RF[rs] ^ RF[rt] //Exclusive OR

- sra $rd, $rs, sa RTL: RF[rd] ←RF[rs] >> sa //Shift right arithmetic

2. Tipul I

- bne $rs, $rt, imm RTL: If(RF[rs] !=RF[rt]) then PC ←PC+1 + S_Ext(imm)

Else PC ←PC+1 // Branch on not equal

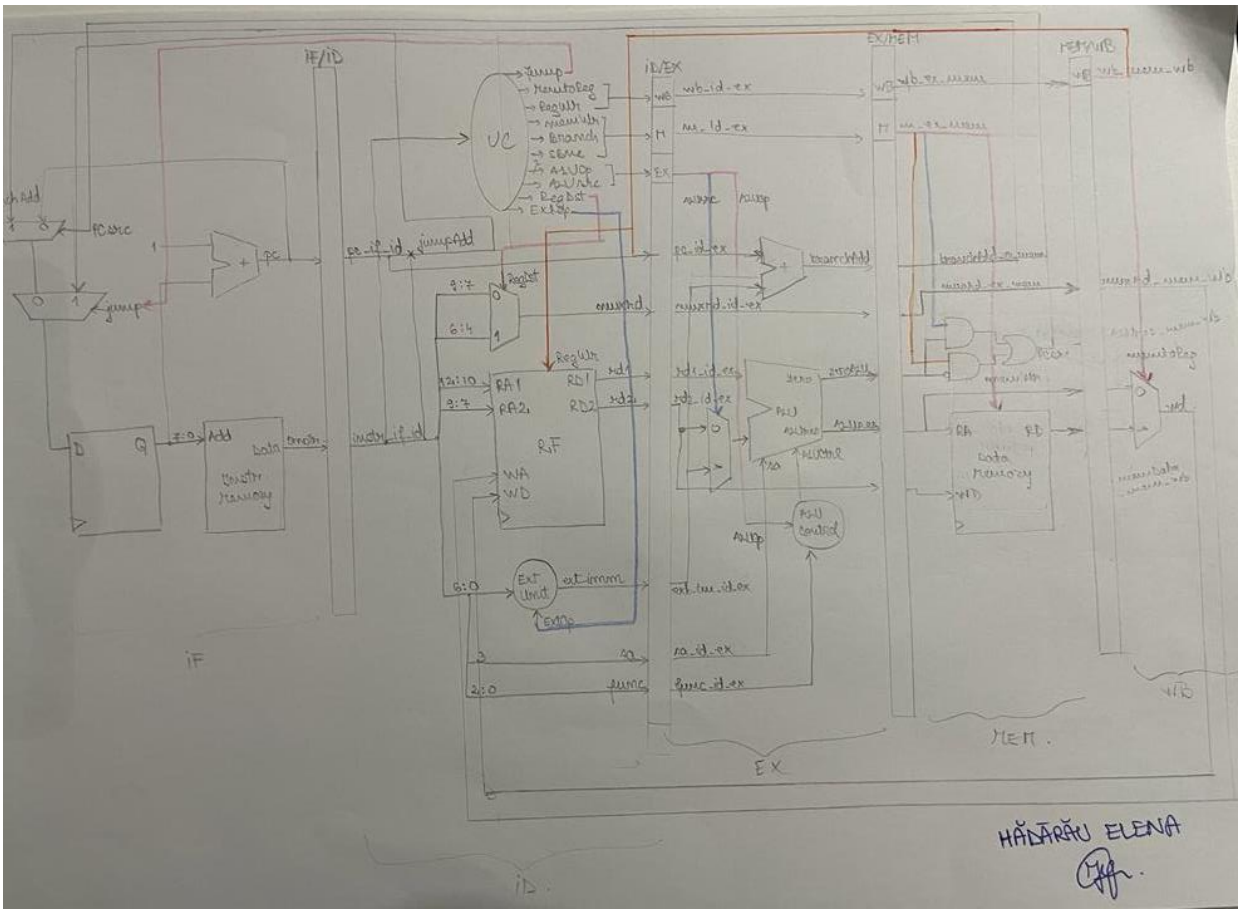
* s-au făcut modificări doar pentru instrucțiunea bne. Mai exact, am adăugat o poartă AND între semnalul Zero de la ieșirea ALU negat și semnalul sbne, iar între rezultatul acestei porți și al celui dintre Zero și semnalul branch am folosit o poartă SAU pentru a determina valoarea semnalului pcSrc.

- andi $rt, $rs, imm RTL: RF[rt] ←RF[rs] & Z_Ext(imm) // Logical AND unsigned constant

b) Configurare registre MIPS16 Pipeline

IF/ID	ID/EX	EX/MEM	MEM/WB
instr_if_id(16b)<=instr	wb_id_ex(2b)<=memtoReg&regWrite	Wb_ex_mem(2b)<=wb_id_ex	Wb_mem_wb(2b)<= wb_ex_mem
pc_if_id(16b)<=pc	m_id_ex(3b)<=memWrite&branch&sbne	M_ex_mem(3b)<=m_id_ex	memData_mem_wb(16b)<=memData
	ex_id_ex(3b)<=ALUop&ALUsrc	Rd2_ex_mem(16b)<=rd2_id_ex	ALUres_mem_wb<=ALUres_ex_mem
	pc_id_ex(16b)<=pc_if_id	ALUres_ex_mem(16b)<=ALUres	Muxrd_mem_wb<=muxrd_ex_mem
	rd1_id_ex(16b)<=rd1	branchAdd_ex_mem(16b)<=branchAdd	
	rd2_id_ex(16b)<=rd2	Zero_ex_mem(1b)<=zeroALU	
	Ext_imm_id_ex(16b)<=ext_imm	Muxrd_ex_mem(3b)<=muxrd_id_ex	
	Sa_id_ex(1b)<=sa		
	Func_id_ex(3b)<=func		
	Muxrd_id_ex(3b)<=muxrd		

c) Schema procesorului MIPS16 pipeline



d) Programul în limbaj de asamblare

Varianta cu hazarduri

```
0: add $1, $0, $0          #i=0, contorul buclei
1: addi $2,$0,10           #se pune în registrul 2 numărul de iterații (10)
2: add $3, $0,$0           #inițializare indexului locației de memorie
3: add $4,$0,$0            #catepare=0 – în registrul 4 se va reține nr. elementelor pare
4: beq $2,$1,7             #verific dacă s-au făcut 10 iterații
5: lw $5,0($3)             #în carc în registrul 5 elementul curent din șir
6: andi $6, $5, 1          #instr. 6 și 7 – verific dacă elementul e par sau nu; dacă e impar
7: bne $6,$0,1              #se face salt la instr. 9
8: addi $4,$4,1            #dacă elementul verificat e par, atunci se incrementează reg4
9: addi $3, $3, 1          #incrementez indexul locației pt urm. element din vector
10: addi  $1, $1, 1        #actualizare contor buclă – i++
11: j 4                    #salt la începutul buclei
```

12: sw $4,12($0) #salvarea în memorie la adresa 12 a numărului de elemente pare

Varianta fără hazarduri

0: add $1, $0, $0

1: addi $2,$0,10

2: add $3, $0,$0

3: add $4,$0,$0

4: beq $2,$1,18

5: NoOp

6: NoOp

7: NoOp

8: lw $5,0($3)

9: NoOp

10: NoOP

11: andi $6, $5, 1

12: NoOp

13: NoOp

14: bne $6,$0,4

15: NoOp

16: NoOp

17: NoOp

18: addi $4,$4,1

19: addi $3, $3, 1

20: addi $1, $1, 1

21: j 4

22: NoOp

23: sw $4,12($0)

*Am adăugat în arhivă un fișier de tip excel cu diagrama pipeline.

Programul în cod mașină

0: b”000_000_000_001_0_000” # instr. in hexa: x”0010”

1: b”001_000_010_0001010” #x”210A”

2: b”000_000_000_011_0_000” #x”0030”

3: b”000_000_000_100_0_000” #x”0040”

4: b”000_000_000_000_0_000” #x”0000”

5: b”000_000_000_000_0_000” #x”0000”

6: b”100_010_001_0010001” #x”8891”

7: b”000_000_000_000_0_000” #x”0000”

8: b”000_000_000_000_0_000” #x”0000”

9: b”000_000_000_000_0_000” #x”0000”

10: b”010_011_101_0000000” #x”4E80”

11: b”000_000_000_000_0_000” #x”0000”

12: b”000_000_000_000_0_000” #x”0000”

13: b”101_101_110_0000001” #x”B701”

14: b”000_000_000_000_0_000” #x”0000”

15: b”000_000_000_000_0_000” #x”0000”

16: b”110_110_000_0000100” #x”D804”

17: b”000_000_000_000_0_000” #x”0000”

18: b”000_000_000_000_0_000” #x”0000”

19: b”000_000_000_000_0_000” #x”0000”

20: b”001_100_100_0000001” #x”3201”

21: b”001_011_011_0000001” #x”2D81”

22: b”111_0000000000100” #x”E004”

23: b”001_001_001_0000001” #x”2481”

24: b”011_000_100_0001100” #x”620C”

*Pentru trasarea programului am adăugat un fișier de tip Excel

e) Nu există părți din procesor incomplete.
f) Nu există erori la implementarea în VHDL.

g) Programul a fost testat pe plăcuță FPGA BASYS 3 și funcționează conform așteptărilor.