MIPS CICLU UNIC

- Raport -

Nume student: Matiș Oana-Antonia

Grupa: 30228

CUPRINS

1. Instrucțiuni…………………………………………………………………...3

2. Valorile semnalelor de control…………………………………………….....7

3. Descrierea programului ales………………………………………………....7

4. Trasarea execuției programului.....................................................................10

5. Schema procesorului MIPS pe 16 biți...........................................................11

6. Concluzii.......................................................................................................12

1. **Instrucțiuni**
2. Instrucțiuni de tip R

ADD: $d ß$s + $t; PC ß PC + 1;

add $d, $s, $t

000 sss ttt ddd 0 000

Ex. add $1, $2, $5

000 010 101 001 0 000

SUB: $d ß$s - $t; PC ßPC + 1;

sub $d, $s, $t

000 sss ttt ddd 0 001

Ex. sub $1, $5, $2

000 101 010 001 0 001

SLL: $d ß$t << h; PC ßPC + 1;

sll $d, $t, h

000 sss ttt ddd h 010

Ex. sll $1, $3, 1

000 000 011 001 1 010

SRL: $d ß$t >> h; PC ßPC + 1;

srl $d, $t, h

000 000 ttt ddd h 011

Ex. srl $1, $2, 1

000 000 010 001 1 011

OR: $d ß$s | $t; PC ßPC + 1;

or $d, $s, $t

000 sss ttt ddd 0 101

Ex. or $3, $2, $1

000 010 001 011 0 101

AND: $d ß$s & $t; PC ßPC + 1;

and $d, $s, $t

000 sss ttt ddd 0 100

Ex. and $2, $1, $4

000 001 100 010 0 100

XOR: $d ß$s ^ $t; PC ßPC + 1;

xor $d, $s, $t

000 sss ttt ddd 0 110

Ex. xor $1, $2, $3

000 010 011 001 0 110

SLLV: $d ß$t << $s; PC ßPC + 1;

sllv $d, $t, $s

000 sss ttt ddd 0 110

Ex. sllv $2, $1, $5

000 101 001 010 0 111

1. Instrucțiuni de tip I

ADDI: $t ß$s + imm; PC ßPC + 1;

addi $t, $s, imm

001 sss ttt iiiiiii

Ex. addi $2, $1, 3

001 001 010 0000011

LW: $t ßMEM[$s + offset]; PC ßPC + 1;

lw $t, offset($s)

010 sss ttt iiiiiii

Ex. lw $1, 4($3)

010 011 001 0000100

SW: MEM[$s + offset] ß$t; PC ßPC + 1;

sw $t, offset($s)

011 sss ttt iiiiiii

Ex. sw $5, 3($2)

011 010 101 0000011

BEQ: if $s == $t then PC ßPC + 1 + (offset << 2); else PC ßPC + 1;

beq $s, $t, offset

100 sss ttt iiiiiii

Ex. beq $1, $3, 3

100 001 011 0000011

BGTZ: if $s > 0 PC ßPC + 4 + (offset << 2); else PC ßPC + 1;

bgtz $s, offset

101 sss 000 iiiiiii

Ex. bgtz $2, 6

101 010 000 0000110

BNE: if $s != $t PC ßPC + 1 + (offset << 2); else PC ßPC + 1;

bne $s, $t, offset

110 sss ttt iiiiiii

Ex. bne $2, $3, 1

110 010 011 0000100

1. Instrucțiuni de tip J

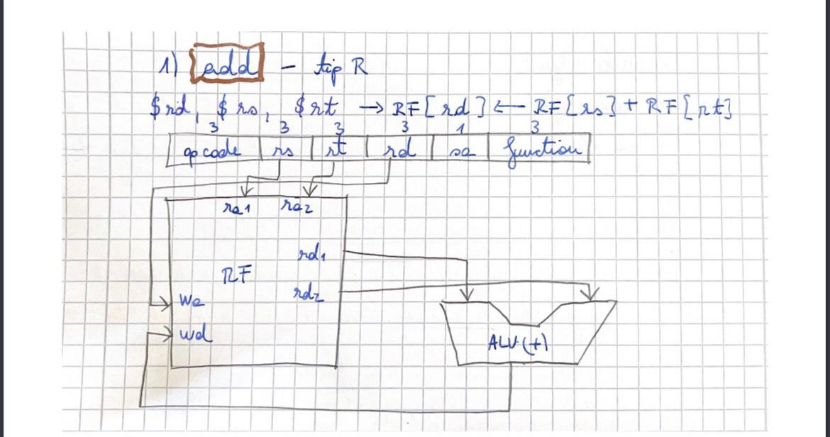
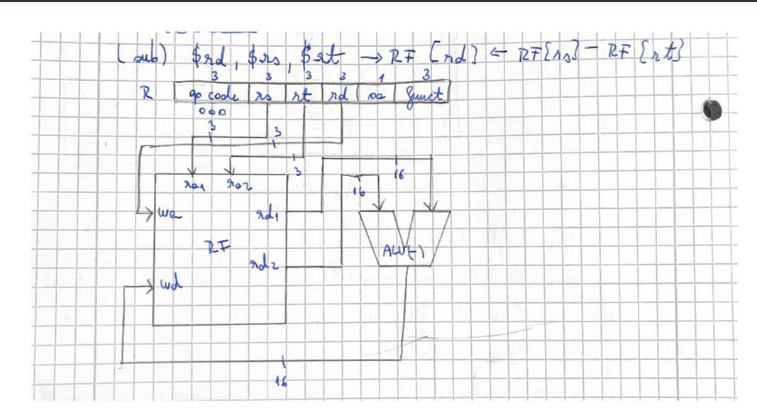
J: PC ß((PC +4) & 0xf0000000) | (target << 2);

j target

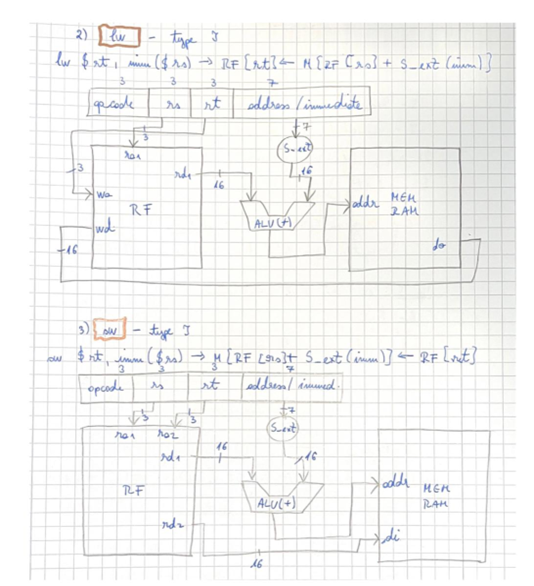
111 iiiiiii

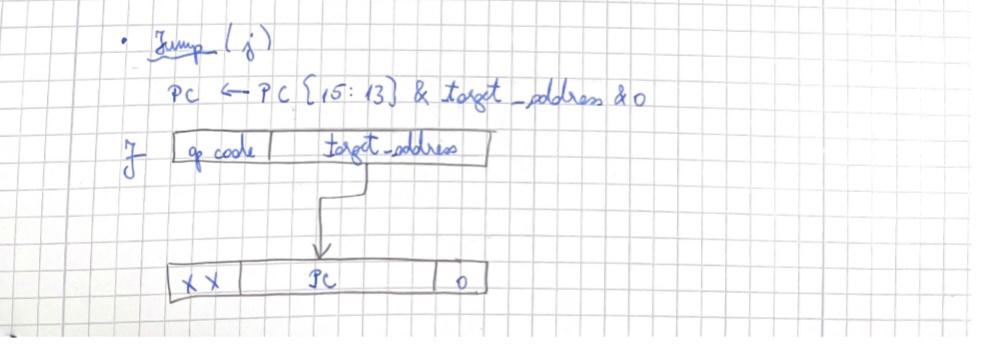
Ex. j 1

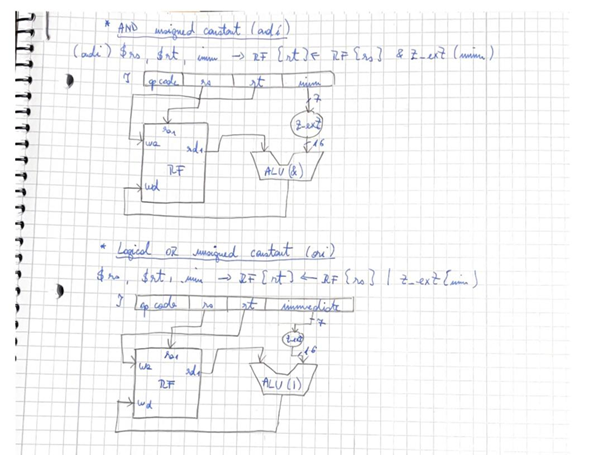
111 0000000000001



O imagine care conține diagramă

Descriere generată automat

O imagine care conține diagramă

Descriere generată automat

1. **Valorile semnalelor de control**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | Reg  dest | Reg  write | Alu  src | Ext  Op | Alu  Ctrl | Mem  write | Mem  to reg | Branch  equal | Branch  not equal | Branch  greater  than zero | Jump |
| ADD | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| SUB | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| SLL | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| SRL | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| AND | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| OR | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| XOR | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| SLLV | 1 | 1 | 0 | X | 00 | 0 | 0 | 0 | 0 | 0 | 0 |
| ADDI | 0 | 1 | 1 | 1 | 01 | 0 | 0 | 0 | 0 | 0 | 0 |
| LW | 0 | 1 | 1 | 1 | 01 | 0 | 1 | 0 | 0 | 0 | 0 |
| SW | X | 0 | 1 | 1 | 01 | 1 | X | 0 | 0 | 0 | 0 |
| BEQ | X | 0 | 0 | 1 | 10 | 0 | X | 1 | 0 | 0 | 0 |
| BGTZ | X | 0 | 0 | 1 | 10 | 0 | X | 0 | 0 | 1 | 0 |
| BNE | X | 0 | 0 | 1 | 10 | 0 | X | 0 | 1 | 0 | 0 |
| J | X | 0 | X | X | 11 | 0 | X | X | X | X | 1 |

1. **Descrierea programului ales**

Programul în limbajul C pe care l-am ales, pentru a fi testat cu ajutorul procesorului, are rolul de a calcula suma elementelor diferite de cifra 3 dintr-un vector de 10 elemente.

Programul parcurge fiecare element dintr-un vector folosind un for. În interiorul acestui for, se testează condiția ca elementul curent să fie diferit de 3 cu ajutorul unui if. În cazul în care elementul curent îndeplinește această condiție, acesta este adăugat la o variabilă suma, care a fost declarată și inițializată la începutul programului. În caz contrar, dacă elementul este egal cu 3, acesta nu este adăugat la sumă.

În vederea implementării programului pe FPGA, acesta trebuie convertit în limbaj de asamblare și apoi în cod mașină.

* **Programul în limbaj C**

int A[10]={1,3,2,4,3,1,7,3,9,2};

int sum=0;

for(int i=0; i<n; i++)

{

If(A[i]!=3)

{

sum=sum+A[i];

}

}

* **Programul în limbaj de asamblare**

add $1, $0, $0

addi $4, $0, 10

addi $6, $0, 3

add $2, $0, $0

add $5, $0, $0

beq $1, $4, 6

lw $3, 10($2)

bne $3, $6, 1

add $5, $5, $3

addi $2, $2, 1

addi $1, $1, 1

j 5

sw $5, 20($0)

* **Programul in cod mașină**

000 000 000 001 0 000 => 0010

001 000 100 0001010 => 220A

001 000 010 0000011 => 2103

000 000 000 010 0 000 => 0020

000 000 000 101 0 000 => 0050

100 001 100 0000110 => 8606

010 010 011 0001010 => 498A

110 011 110 0000001 => 8F01

000 101 011 101 0 000 => 15D0

001 010 010 0000001 => 2901

001 001 001 0000001 => 2481

111 0000000000101 => E005

011 000 101 0010100 => 6294

1. **Trasarea execuției programului**

add $1, $0, $0 RD1=0, RD2=0, Ext\_imm=0, AluRes=0

addi $4, $0, 10 RD1=0, RD2=0, Ext\_imm=10, AluRes=10

addi $6, $0, 3 RD1=0, RD2=0, Ext\_imm=3, AluRes=3

add $2, $0, $0 RD1=0, RD2=0, Ext\_imm=0, AluRes=0

add $5, $0, $0 RD1=0, RD2=0, Ext\_imm=0, AluRes=0

beq $1, $4, 6 RD1=0, RD2=10, Ext\_imm=0, AluRes=0

lw $3, 10($2) RD1=2, RD2=3, Ext\_imm=0, AluRes=0, wd=1, mem\_data=1 (A[0])

beq $3, $6, 1 RD1=1, RD2=3, Ext\_imm=1, AluRes=0

add $5, $5, $3 RD1=0, RD2=1, Ext\_imm=0, AluRes=1

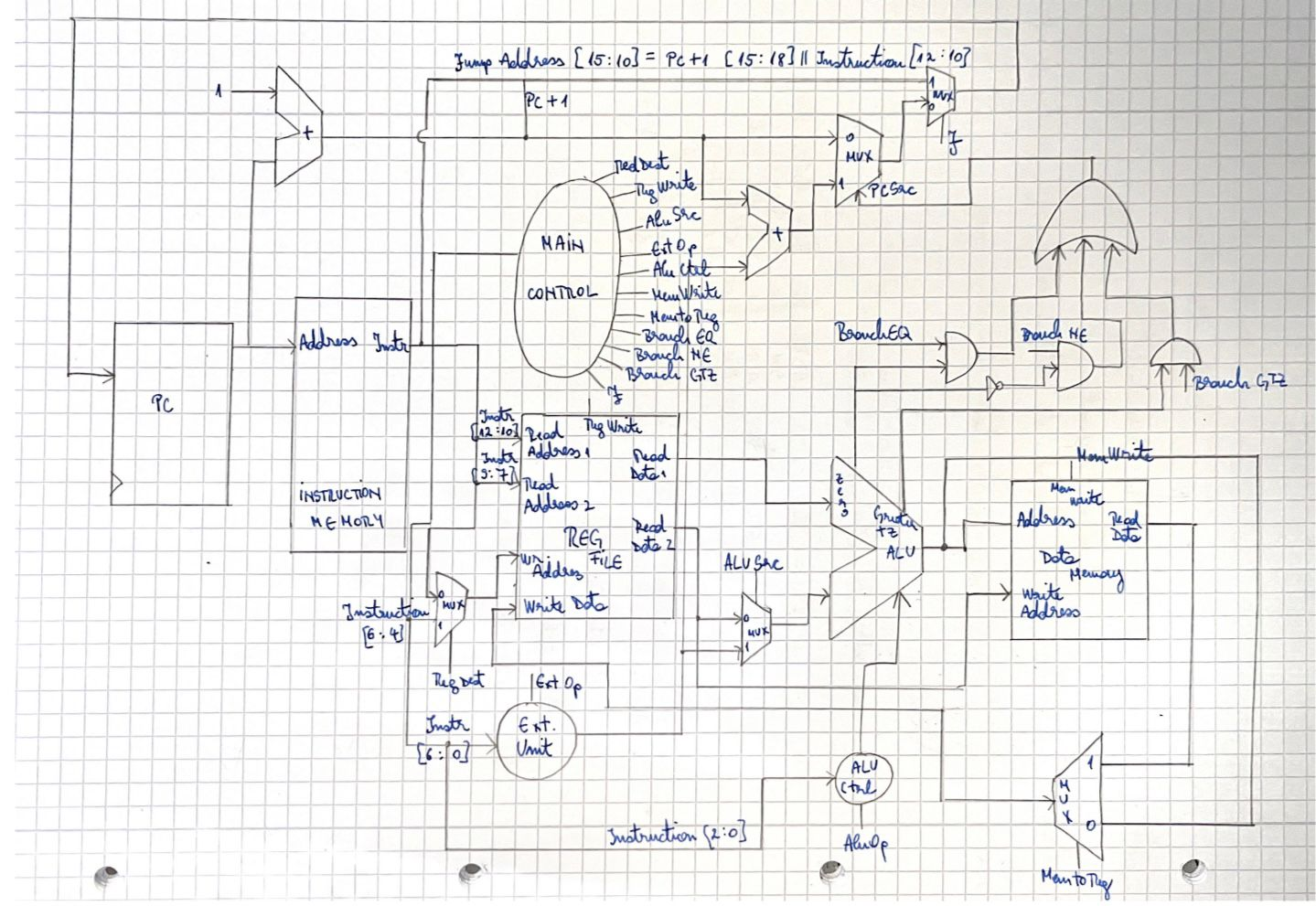
addi $2, $2, 1 RD1=0, RD2=0, Ext\_imm=1, AluRes=1

addi $1, $1, 1 RD1=0, RD2=0, Ext\_imm=1, AluRes=1

j 5 RD1=0, RD2=0, Ext\_imm=5, AluRes=0

sw $5, 20($0) RD1=0, RD2=5, Ext\_imm=0, AluRes=0

1. **Schema procesorului MIPS pe 16 biți**

****

1. **Concluzii**

În ceea ce privește descrierea procesorului MIPS în VHDL, toate componentele acestuia au fost implementate. Proiectul a fost testat pe placa FPGA și este parțial funcțional. Bucla princpală a programului se oprește după 10 iterații, dar nu execută al doilea Branch(bne) în cazul în care se găsește un element egal cu 3.