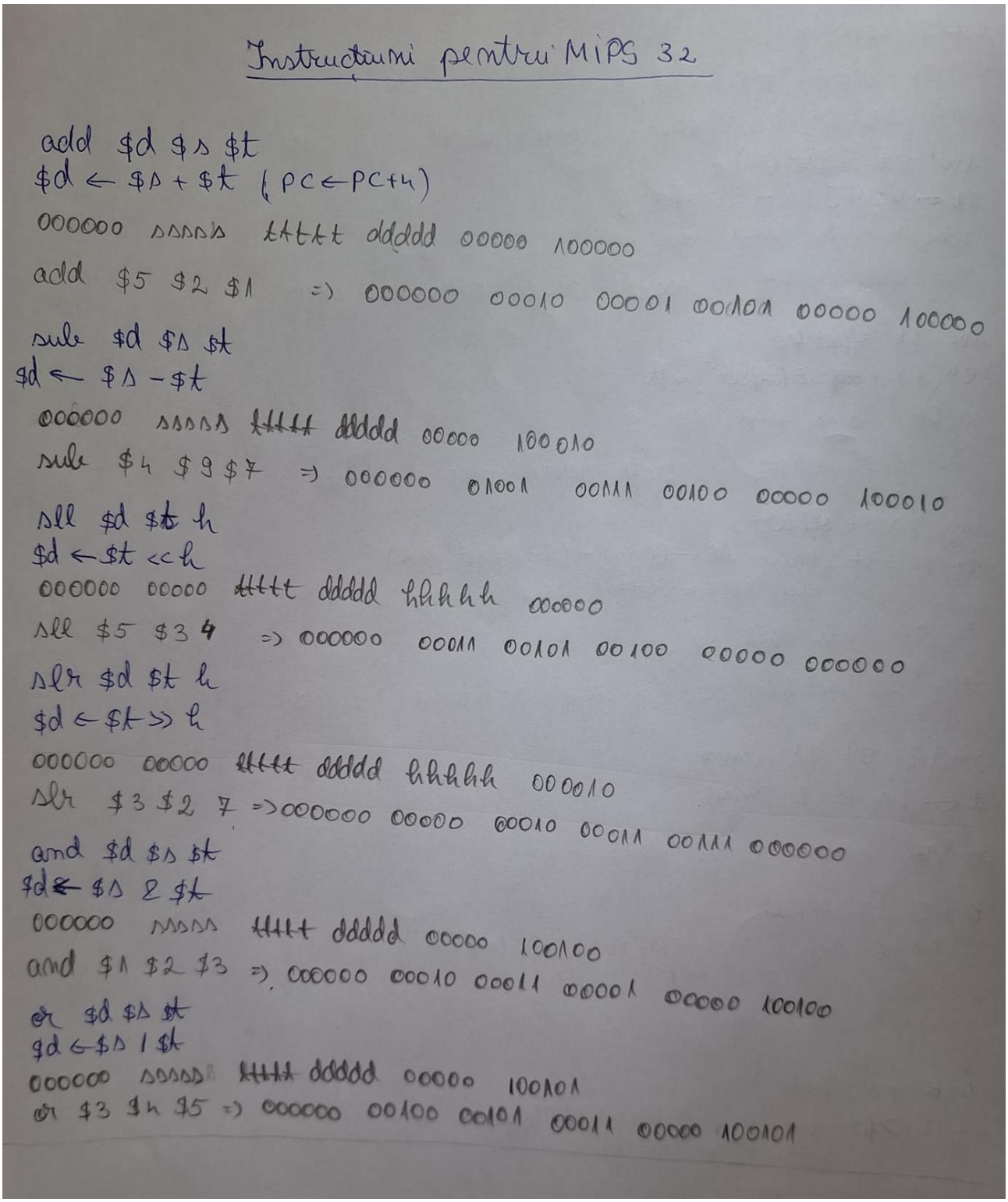
Procesor MIPS single-cycle

Cotei Ruxandra-Maria
Grupa 30228
Universitatea Tehnica din Cluj-Napoca

Cuprins:

Cuprins: 2
Instrucțiuni: 3
Tabel cu semnalele de control: 5
Cerița problemei: 5
Descrierea funcționării: 6
Posibilități de dezvoltare ulterioară: 6
Probleme întâmpinate: 6
Schema detaliată: 7
Codul C: 7
Codul RTL: 8

Instrucțiuni:



Instrucțiuni pentru MIPS 32

add $d $s $t
$d ← $s + $t (PC←PC+4)
000000 sssss ttttt ddddd 00000 100000
add $5 $2 $1 => 000000 00010 00001 00101 00000 100000

sub $d $s $t
$d ← $s - $t
000000 sssss ttttt ddddd 00000 100010
sub $4 $9 $7 => 000000 01001 00111 00100 00000 100010

sll $d $t h
$d ← $t << h
000000 00000 ttttt ddddd hhhhh 000000
sll $5 $3 4 => 000000 00011 00101 00100 00000 000000

slr $d $t h
$d ← $t >> h
000000 00000 ttttt ddddd hhhhh 000010
slr $3 $2 7 => 000000 00000 00010 00011 00111 000000

and $d $s $t
$d ← $s & $t
000000 sssss ttttt ddddd 00000 100100
and $1 $2 $3 => 000000 00010 00011 00001 00000 100100

or $d $s $t
$d ← $s | $t
000000 sssss ttttt ddddd 00000 100101
or $3 $4 $5 => 000000 00100 00101 00011 00000 100101

addi $t $s imm

$t ← $s + SE(imm)

001000 sssss tttt iiii...i (16)

addi $3 $4 9 => 001000 00100 00011 0...01001

lw $t, offset($s)

$t ← MEM[$s + SE(offset)]

100011 sssss tttt 00...0 (16)

lw $7, 11($4) => 100011 00100 00111 0...01011

sw $t, offset($s)

MEM[$s + SE(offset)] ← $t

101011 sssss tttt 00...0 (16)

sw $5, 9($3) => 101011 00011 00101 0...01001

beq $s $t offset

if $s = $t then PC ← (PC+4) + (SE(offset) << 2)

else PC ← (PC+4)

000100 ssss tttt 00...0 (16)

beq $5 $6 8 => 000100 00101 00110 0...01000

---

andi $t $s imm

$t ← $s & ZE(imm)

bne2 100100 sssss tttt iii...i (16)

bne2 $s, offset

if $s ≠ 0 then PC ← (PC+4) + (SE(offset) << 2)

else PC ← PC+4

000101 sssss 000...0 (21)

$t

bgez $s, offset

if $s ≥ $t then PC ← (PC+4) + (SE(offset) << 2)

else PC ← (PC+4)

000111 ssss tttt 00...0 (16)

---

jaddr

PC ← (PC+4)[31:28] || (addr << 2)

000010 aa...a (26)

j 31 => 000010 0...011111

Tabel cu semnalele de control:

Instr	Reg Dst	Reg Write	ALU Src	Ext Op	ALU Op	Mem Write	MemTo Reg	Branch	Jump	Alu Ctrl	Branch	Branch Not Equal Zero	Branch Greater
Bnez	0	0	0	1	011	0	0	0	0	001	X	1	X
Bg	0	0	1	1	011	0	0	0	0	001	X	X	1
Beq	0	0	0	1	011	0	0	0	0	000	1	X	X
Sub	1	1	0	0	000	0	0	0	0	001	X	X	X
Andi	0	1	1	0	010	0	0	0	0	000	X	X	X
Lw	0	1	1	1	001	0	1	0	0	000	X	X	X
Sw	0	0	1	1	001	1	0	0	0	000	X	X	X
j	0	0	0	0	000	0	0	0	1	xxx	X	X	X

Cerința problemei:

Să se determine valoarea pară maximă dintr-un șir de N numere stocate în memorie începând cu adresa A (A≥12). A și N se citesc de la adresele 0, respective 4. Rezultatul se va scrie în memorie la adresa 8. (problema 12 din anexa 7).

Descrierea funcționării:

1. Se declară o variabilă întreagă `Max` și se inițializează cu prima valoare din vectorul `v`.
2. Programul intră într-o buclă `for` care parcurge fiecare element al vectorului `v`.
3. În interiorul buclei, se verifică dacă elementul curent al vectorului `v` este par sau nu. Pentru a face aceasta, se utilizează operatorul bit cu bit & între valoarea curentă `v[i]` și 1 (`v[i] & 1`). Dacă rezultatul este zero, înseamnă că ultimul bit al lui `v[i]` este zero, ceea ce înseamnă că `v[i]` este un număr par.
4. În continuare, se verifică dacă elementul curent este mai mare decât valoarea maximă (`Max`) găsită până acum. Dacă da, valoarea lui `Max` este actualizată cu valoarea curentă `v[i]`.
5. La final, se memorează valoarea maximă găsită la adresa 8.

Posibilități de dezvoltare ulterioară:

Deși proiectul a fost dezvoltat conform specificațiilor și cerințelor stabilite, este important de menționat că nu am avut încă oportunitatea de a testa design-ul pe o placă hardware dedicată. Testarea pe placă hardware este un pas esențial pentru validarea funcționalității și performanței în mediul real.

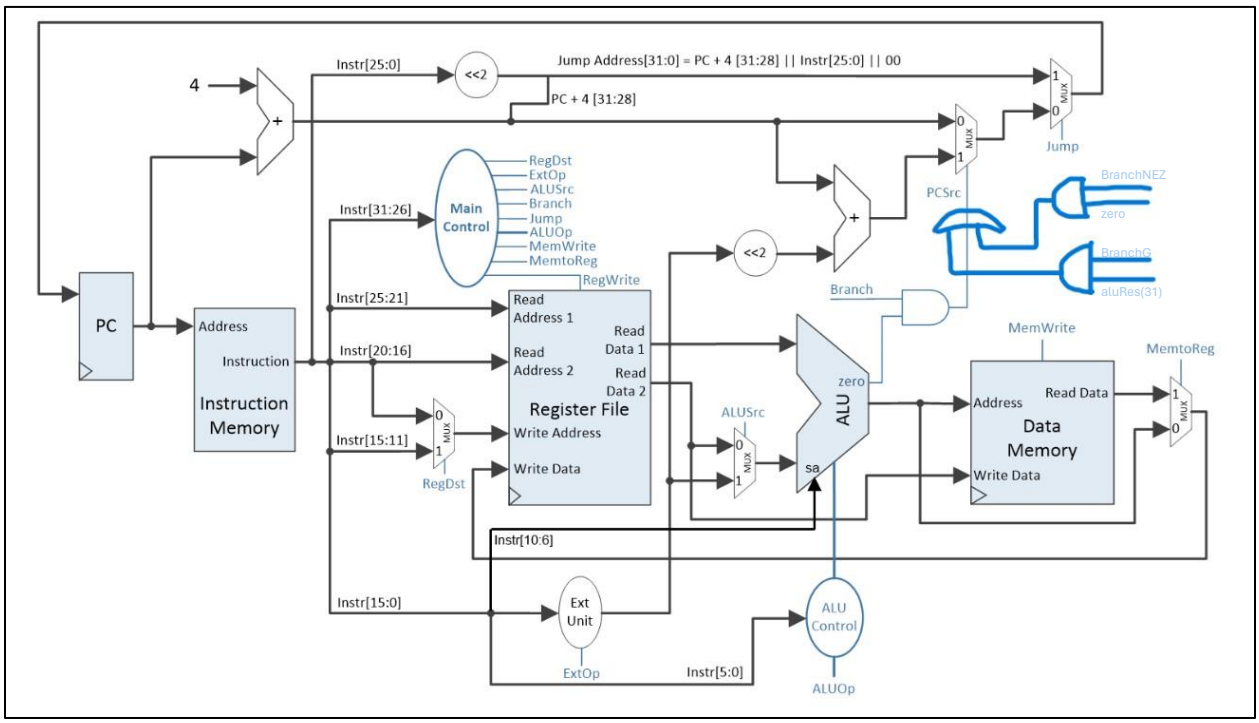
Având în vedere complexitatea și importanța proiectului, recomandăm efectuarea testelor pe placă hardware pentru a confirma corectitudinea și stabilitatea design-ului în condiții reale de utilizare, deși principial ar trebui să funcționeze.

Probleme întâmpinate:

În timpul procesului de dezvoltare, am întâmpinat anumite dificultăți în abordarea problemei și în redactarea codului necesar. În special, am constatat că definirea unei strategii logice adecvate și transpunerea acesteia în cod VHDL a reprezentat o provocare majoră. Lipsa experienței anterioare în proiectarea de procesoare și în gândirea logică asociată acestei domenii a amplificat aceste dificultăți.

Am întâmpinat dificultăți și în înțelegerea și aplicarea corectă a conceptelor de proiectare hardware. Aceste provocări m-au determinat să acordăm o atenție sporită documentației și resurselor disponibile pentru a ne perfecționa cunoștințele și abilitățile în acest domeniu. Cu toate acestea, am învățat din aceste experiențe și am dobândit o mai mare înțelegere a procesului de dezvoltare hardware, contribuind astfel la creșterea nivelului nostru de competență și la îmbunătățirea abilităților noastre în domeniul proiectării de procesoare.

Schema detaliată:



Codul C:

```c
#include <stdio.h>

int main() {
    int n = 7;
    int v[] = {3, 8, 4, 5, 10, 2, 7};
    int Max = v[0];

    for(int i = 0; i<n; i++){
        if(v[i] % 2 == 0){
            if(v[i] > Max) {
                Max = v[i];
            }
        }
    }

    int a = Max;

    return 0;
}
```

Codul RTL:

```
100011_00000_00001_0000000000000000
100011_00000_00010_0000000000000100
000000_00000_00000_00011_00000_100000
000000_00000_00010_00100_00000_100000
000100_00011_00001_0000000000001001
        100011_00011_00101_0000000000000011
        100100_00110_00101_0000000000000001
        000101_00110_00000000000000001011
                001000_00011_00011_0000000000000001
                000010_00000000000000000000000101
        000111_00101_00100_0000000000001101
                000000_00100_00000_00101_00000_100000
        001000_00011_00011_0000000000000001
        000010_00000000000000000000000101
101011_00000_00100_0000000000001000
```

```
 -- lw n, 0($0)          -- extrag N
 --2     -- lw a, 4($0)          -- extrag adresa de inceput
 --3     -- add i, $0, $0     -- i = 0, contor
 --4     -- add max, $0, a   -- initializez Max
 --5     -- beq i, n, 9
 --6        -- lw x, a(i)              -- aduc in x elementul curet
     --7        -- andi aux, x, 1           -- aux = x & 1
     --8        -- bnez aux, 11                -- verific daca numarul e par
1",      --9             -- add i, i, 1            -- i++
,        --10            -- j 5                          -- conditia nu e indeplinita => urmatoarea iteratie a buclei
1",      --11       -- bg x, max, 13           --verific daca  x > Max
000",    --12            -- add x, max, $0            -- x = max
 --13       -- add i, i, 1                  -- i++
     --14       -- j 5                          -- sar la inceputul buclei
 --15    -- sw max, 8($0)   -- stochez max la adresa 8
```

```
B"100011_00000_00001_0000000000000000",      --1     -- lw n, 0($0)          -- extrag N
B"100011_00000_00010_0000000000000100",      --2     -- lw a, 4($0)          -- extrag adresa de inceput
B"000000_00000_00000_00011_00000_100000",    --3     -- add i, $0, $0     -- i = 0, contor
B"000000_00000_00010_00100_00000_100000",    --4     -- add max, $0, a   -- initializez Max
B"000100_00011_00001_0000000000001001",      --5     -- beq i, n, 9
B"100011_00011_00101_0000000000000011",      --6        -- lw x, a(i)              -- aduc in x elementul curet
    B"100100_00110_00101_0000000000000001",      --7        -- andi aux, x, 1           -- aux = x & 1
    B"000101_00110_00000000000000001011",        --8        -- bnez aux, 11                -- verific daca numarul e par
        B"001000_00011_00011_0000000000000001",      --9            -- add i, i, 1            -- i++
        B"000010_00000000000000000000000101",        --10           -- j 5                          -- conditia nu e indeplinita => urmatoarea iteratie a buclei
        B"000111_00101_00100_0000000000001101",      --11       -- bg x, max, 13           --verific daca  x > Max
        B"000000_00100_00000_00101_00000_100000",    --12           -- add x, max, $0            -- x = max
B"001000_00011_00011_0000000000000001",      --13       -- add i, i, 1                  -- i++
    B"000010_00000000000000000000000101",        --14       -- j 5                          -- sar la inceputul buclei
B"101011_00000_00100_0000000000001000",      --15    -- sw max, 8($0)   -- stochez max la adresa 8
others => x"0000"
```