**MIPS Ciclu Unic**

Giesswein Alexia

Grupa 30225

Prof. Coord. Mircea Paul Muresan

**Descrierea programului scris și executat de procesor**

Programul pe care l am implementat parcurge un vector și face suma numerelor pare găsite pe poziții impare.

Programul în C:

int main()

{

int vec[10] = {3, 5, 8, 9, 12, 2, 6, 11, 4, 13}; //suma=8+12+6+4=30

int i, suma = 0, n=10;

for(i=0;i<10;i=i+2){

if(vec[i] & 1==0)

suma = suma + vec[i];

}

return 0;

}

Programul în assembly mips 16:

add $1, $0, $0 ; i=0 contorul

addi $2, $0, 10 ; n=10 (nr de iteratii)

add $3, $0, $0 ; initializam indexul locatiei de memorie (vectorul)

add $4, $0, $0 ; suma=0

addi $6, $0, 1 ; in registrul $6 avem salvat 1

bucla: beq $1, &2, final(7) ; i<10

lw $5, vec($3) ; in $5 avem elementul curent

and $7, $6, $5 ; verificam daca numarul curent e par sau impar

beq $6, $7, acolo(1) ; daca numarul e impar sarim

add $4, $4, $5 ; daca numarul e par il adaugam la suma

acolo: addi $3, $3, 2 ; indexul urmatorului element din sir

addi $1, $1, 2 ; i=i+2

j bucla(5) ; sarim inapoi la bucla

final: sw $4, suma($0) ; salvam suma in memorie la adresa suma

Codificare:

add $1, $0, $0

000\_000\_000\_001\_0\_000

addi $2, $0, 10

001\_000\_010\_0001010

add $3, $0, $0

000\_000\_000\_011\_0\_000

add $4, $0, $0

000\_000\_000\_100\_0\_000

addi $6, $0, 1

001\_000\_110\_0000001

beq $1, &2, final(7)

100\_001\_010\_0000111

lw $5, vec($3)

110\_011\_101\_0001011 --vectorul incepe de la adresa 11

and $7, $6, $5

000\_110\_101\_111\_0\_101

beq $6, $7, acolo(1)

100\_101\_111\_0000001

add $4, $4, $5

000\_100\_101\_100\_0\_000

addi $3, $3, 2

001\_011\_011\_0000010

addi $1, $1, 2

001\_001\_001\_0000010

j bucla(5)

010\_0000101

sw $4, suma($0) --suma 11 + 10\*4 =51

111\_000\_100\_0110011

**Execuția programului**

Cele 2 instrucțiuni de tip R alese suplimentar sunt: xor și sra (shift right arithmetic), iar cele 2 instrucțiuni de tip I alese suplimentar sunt: ori și bne.

Trasarea execuției programului:

add $1, $0, $0 – Instr=0010,PC\_next=1, RD1=0, RD2=0, Ext\_Imm=0010, ALURes=0,

MemData=0, WD=0

addi $2, $0, 10 – Instr=210A,PC\_next=2, RD1=0, RD2=0 (apoi se pune 0A), Ext\_Imm=0A,

ALURes=0A, MemData=0, WD=0A

add $3, $0, $0 – Instr=0030,PC\_next=3, RD1=0, RD2=0, Ext\_Imm=0030, ALURes=0,

MemData=0, WD=0

add $4, $0, $0 – Instr=0040,PC\_next=4, RD1=0, RD2=0, Ext\_Imm=0040, ALURes=0,

MemData=0, WD=0

addi $6, $0, 1 – Instr=2301,PC\_next=5, RD1=0, RD2=0 (apoi se pune 1), Ext\_Imm=1,

ALURes=1, MemData=0, WD=1

bucla: beq $1, &2, final(7) – Instr=8507,PC\_next=6, RD1=0, RD2=0A, Ext\_Imm=7,

ALURes=FFF6, MemData=0, WD=FFF6

lw $5, vec($3) – Instr=CE8B,PC\_next=7, RD1=0, RD2=4, Ext\_Imm=B, ALURes=B,

MemData=3, WD=3

and $7, $6, $5 – Instr=1AF5,PC\_next=8, RD1=1, RD2=3, Ext\_Imm=0075, ALURes=3,

MemData=0, WD=3

beq $6, $7, acolo(1) – Instr=9781,PC\_next=9, RD1=3, RD2=3, Ext\_Imm=1,

ALURes=0, MemData=0, WD=0

add $4, $4, $5 – la prima parcurgere se sare peste instrucțiunea asta

acolo: addi $3, $3, 2 – Instr=2D82,PC\_next=B, RD1=0, RD2=0, Ext\_Imm=2,

ALURes=2, MemData=0, WD=2

addi $1, $1, 2 – Instr=2482,PC\_next=C, RD1=0, RD2=0, Ext\_Imm=2, ALURes=2,

MemData=0, WD=2

j bucla(5) – Instr=4005,PC\_next=D, RD1=0, RD2=0, Ext\_Imm=5, ALURes=0,

MemData=0, WD=0

( aici prezint prima iterație: )

bucla: beq $1, &2, final(7) – Instr=8507,PC\_next=6, RD1=2, RD2=0A, Ext\_Imm=7,

ALURes=8, MemData=0, WD=8

lw $5, vec($3) – Instr=CE8B,PC\_next=7, RD1=2, RD2=3, Ext\_Imm=B, ALURes=D,

MemData=8, WD=8

and $7, $6, $5 – Instr=1AF5,PC\_next=8, RD1=1, RD2=8, Ext\_Imm=0075, ALURes=9,

MemData=0, WD=9

beq $6, $7, acolo(1) – Instr=9781,PC\_next=9, RD1=8, RD2=9, Ext\_Imm=1,

ALURes=1, MemData=0, WD=1

add $4, $4, $5 – Instr=12C0,PC\_next=0A, RD1=0, RD2=8, Ext\_Imm=0040, ALURes=8,

MemData=0, WD=8

acolo: addi $3, $3, 2 – Instr=2D82,PC\_next=B, RD1=2, RD2=2, Ext\_Imm=2,

ALURes=4, MemData=0, WD=4

addi $1, $1, 2 – Instr=2482,PC\_next=C, RD1=2, RD2=2, Ext\_Imm=2, ALURes=4,

MemData=0, WD=4

j bucla(5) – Instr=4005,PC\_next=D, RD1=0, RD2=0, Ext\_Imm=5, ALURes=0,

MemData=0, WD=0

final: sw $4, suma($0) – Instr=E233,PC\_next=E, RD1=0, RD2=1E, Ext\_Imm=0033,

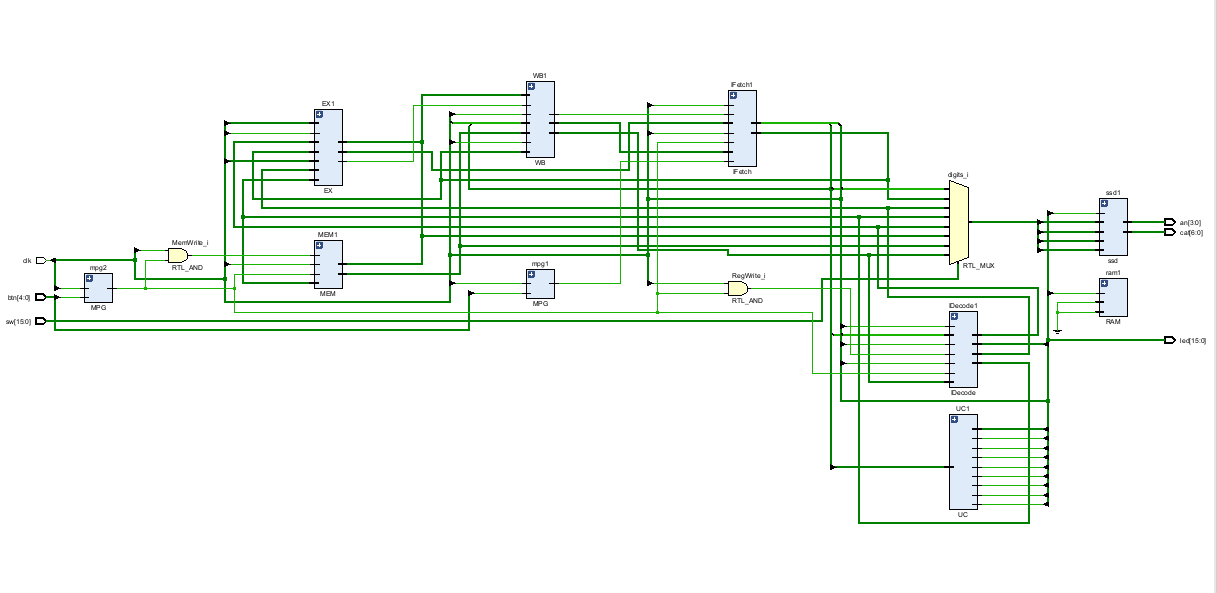
ALURes=0033, MemData=1E, WD=0033 // suma=1E

**Codul în VHDL**

Am implementat toate activitățile din laboratoarele 4-7, am testat codul pe plăcuța Basys3 și funcționează corect. Nu există erori de sintaxă, se generează bitstreamul, iar programul a fost testat pe placă și este funcțional complet.

Programul afișează pe plăcuță (în funcție de switchurile active):

* sw(2:0) = 000 – se afișează instrucțiunea pe ssd
* sw(2:0) = 001 – se afișează următoarea valoare a lui PC (PC + 1) pe ssd
* sw(2:0) = 010 – se afișează RD1 pe ssd
* sw(2:0) = 011 – se afișează RD2 pe ssd
* sw(2:0) = 100 – se afișează Ext\_Imm pe ssd
* sw(2:0) = 101 – se afișează ALURes pe ssd
* sw(2:0) = 110 – se afișează MemData pe ssd
* sw(2:0) = 111 – se afișează WD pe ssd



**Tabel cu valorile semnalelor de control pentru instrucțiuni**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instructiune | Func Instr (2:0) | Opcode Instr (15:13) | ALUOp (1:0) | ALUCtrl (2:0) | RegDst | ExtOp | AluSrc | Branch | BranchNE | Jump | Mem Write | Memto Reg | Reg Write |
| Add | 000 | 000 | 10 | 000 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Sub | 100 | 000 | 10 | 100 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Sll | 010 | 000 | 10 | 010 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Srl | 011 | 000 | 10 | 011 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| And | 001 | 000 | 10 | 001 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Or | 101 | 000 | 10 | 101 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Xor | 110 | 000 | 10 | 110 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Sra | 111 | 000 | 10 | 111 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Addi | - | 001 | 00 | 000 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Lw | - | 110 | 00 | 000 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| Sw | - | 111 | 00 | 000 | X | 1 | 1 | 0 | 0 | 0 | 1 | X | 0 |
| Beq | - | 100 | 01 | 100 | X | 1 | 0 | 1 | 0 | 0 | 0 | X | 0 |
| Ori | - | 011 | 11 | 010 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Bne | - | 101 | 01 | 100 | X | 1 | 0 | 0 | 1 | 0 | 0 | X | 0 |
| J | - | 010 | 01 | XXX | X | 1 | X | X | X | 1 | 0 | X | 1 |

**Schema microprocesorului**

