

CINVESTAV

Alumno: Efraín Arrambide Barrón

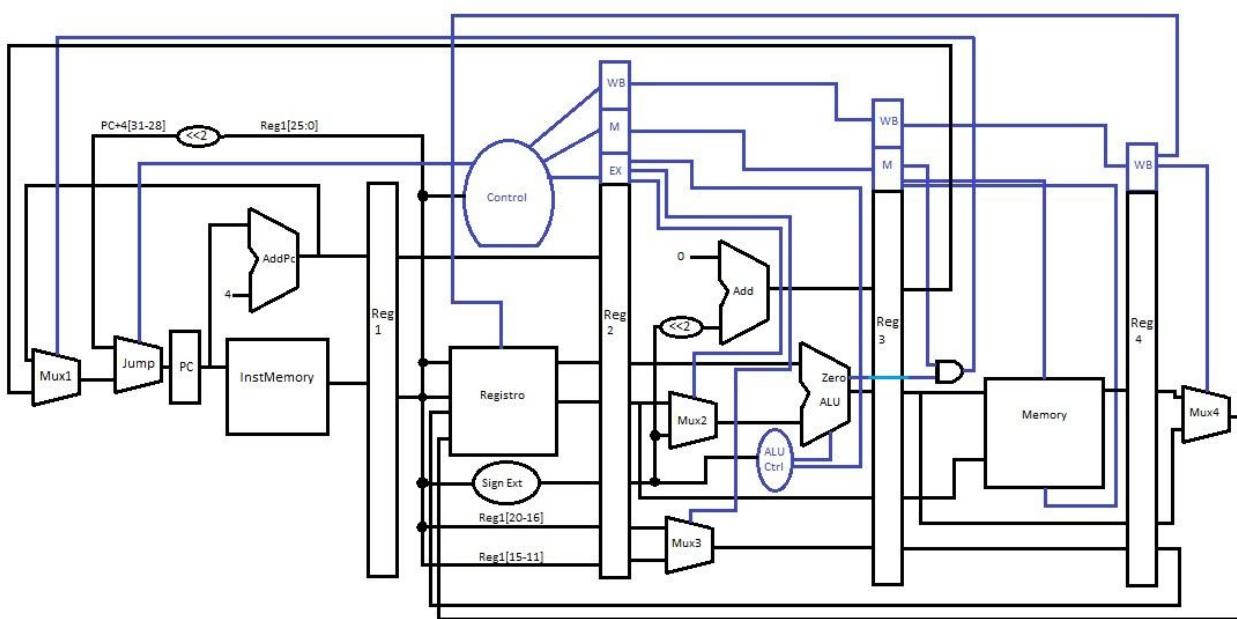
Tarea 4: Pipeline

Docente: Doc. Mariano Aguirre

PADTS 19



a) Diagrama a bloques de la arquitectura implementada, realizado por el alumno.



ADD RD, RS, RT

RD = RS + RT

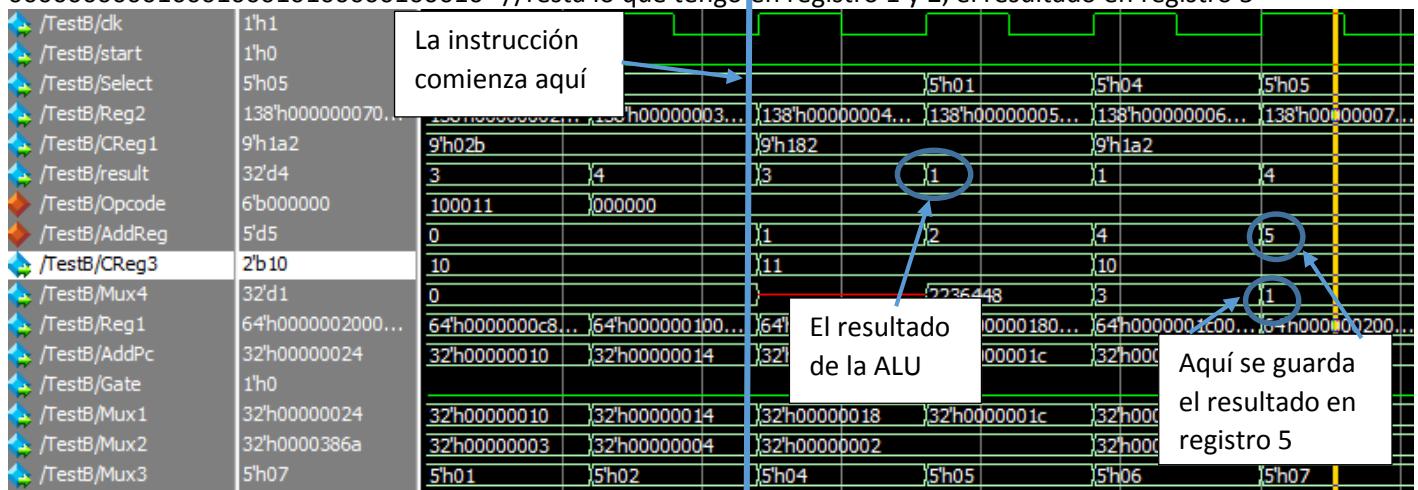
00000000001000100010000000100000 //suma lo que tengo en registro 1 y 2, el resultado en registro 4



SUB RD, RS, RT

RD = RS - RT

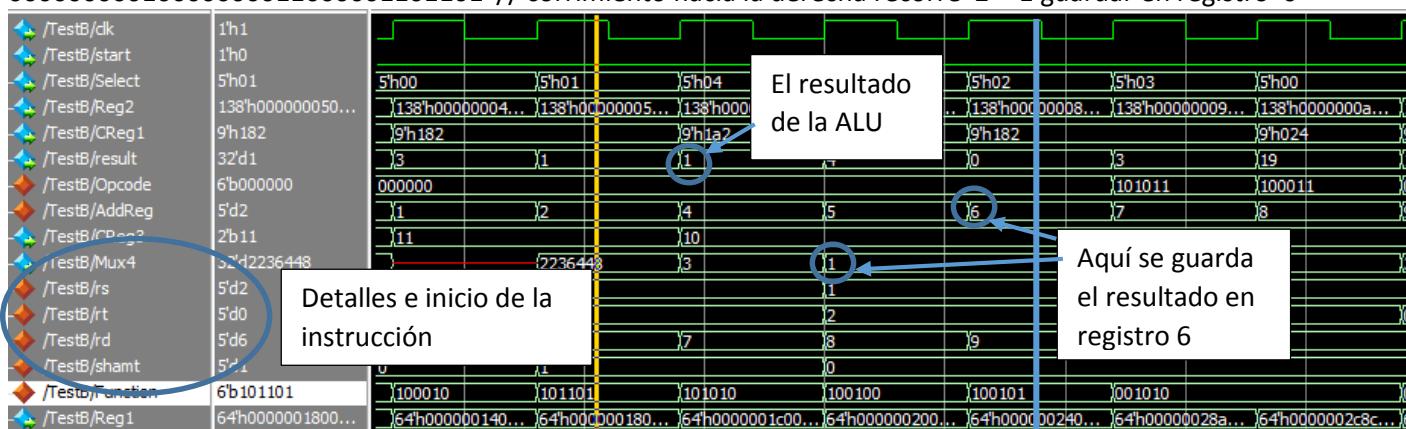
00000000001000100010100000100010 //resta lo que tengo en registro 1 y 2, el resultado en registro 5



SRL RD, RS, SHIFTS

RD = RS >> SHIFT5

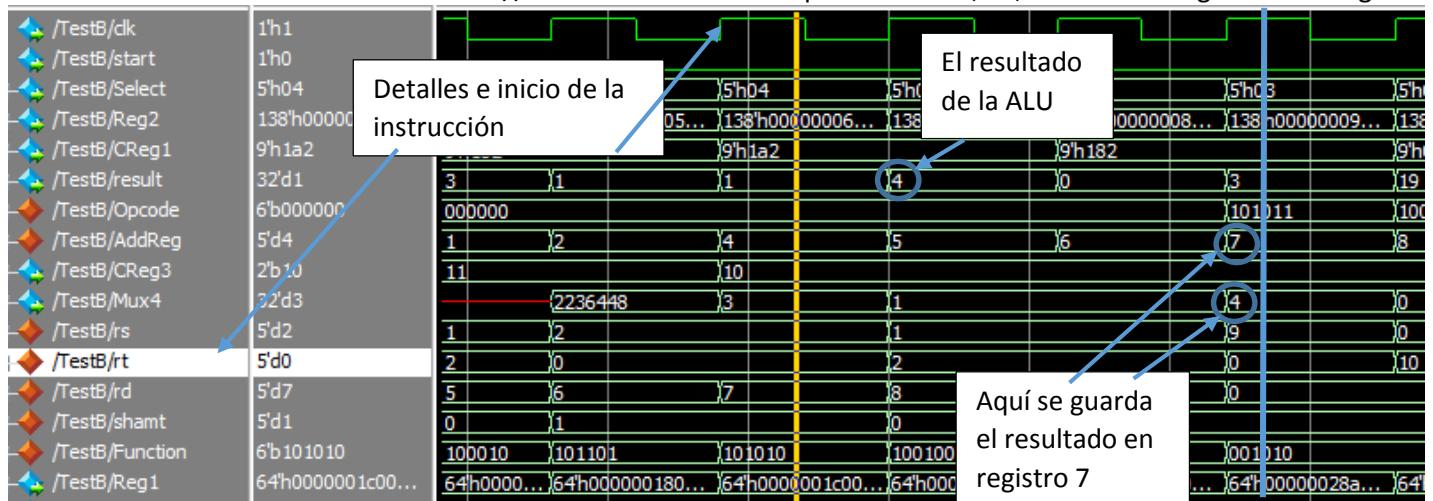
00000000001000000011000001101101 // corrimiento hacia la derecha recorre 2>>1 guardar en registro 6



SLL RD, RS, SHIFT5

RD = RS << SHIFT5

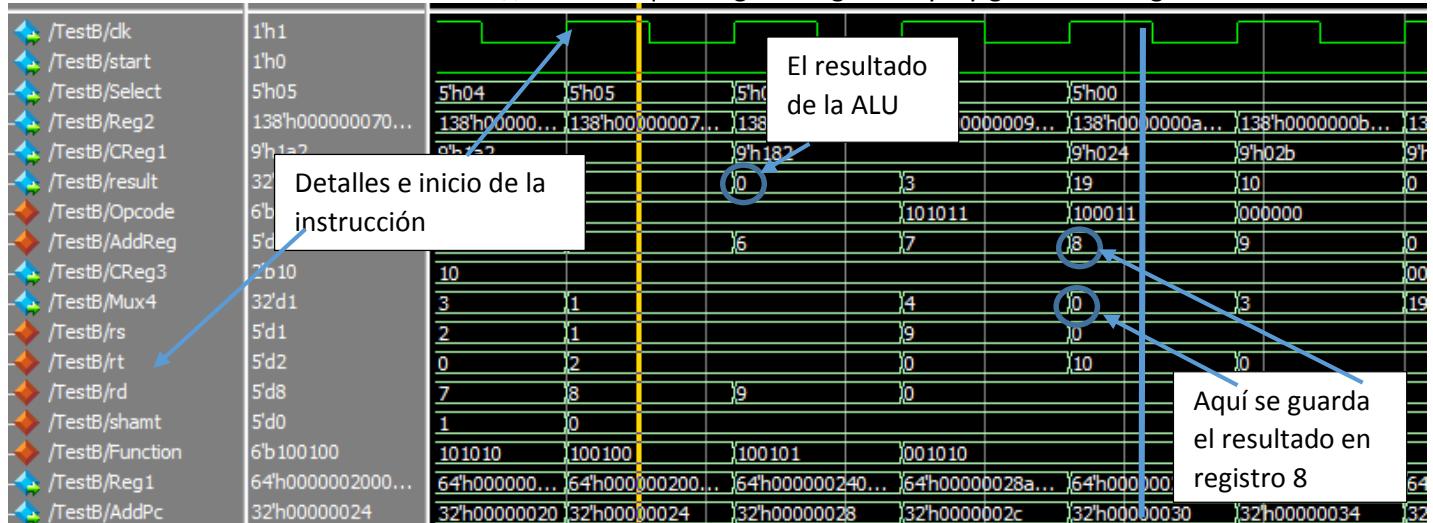
00000000010000000011100001101010 // corrimiento hacia la izquierda SLL RD, RS, SHIFT5 2<<1 guardar en registro 7



AND RD, RS, RT

RD = RS & RT

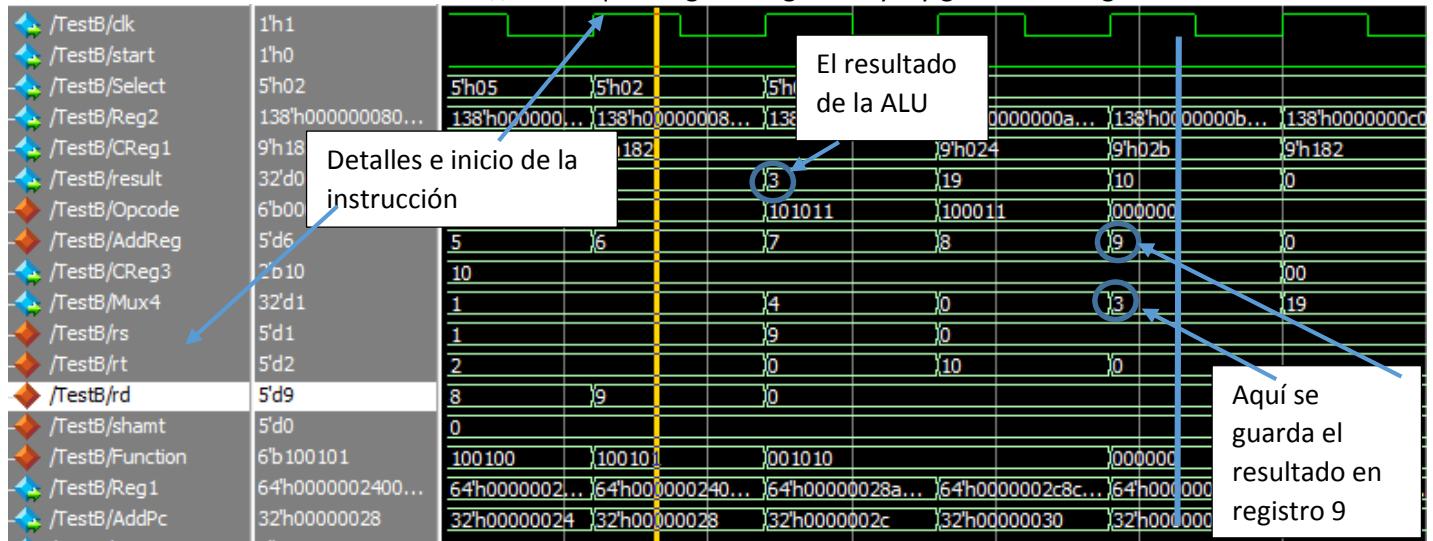
00000000001000100100000000100100 // and lo que tengo en registro 1 y 2 y guardar en registro 8



XOR RD, RS, RT

RD = RS □□ RT

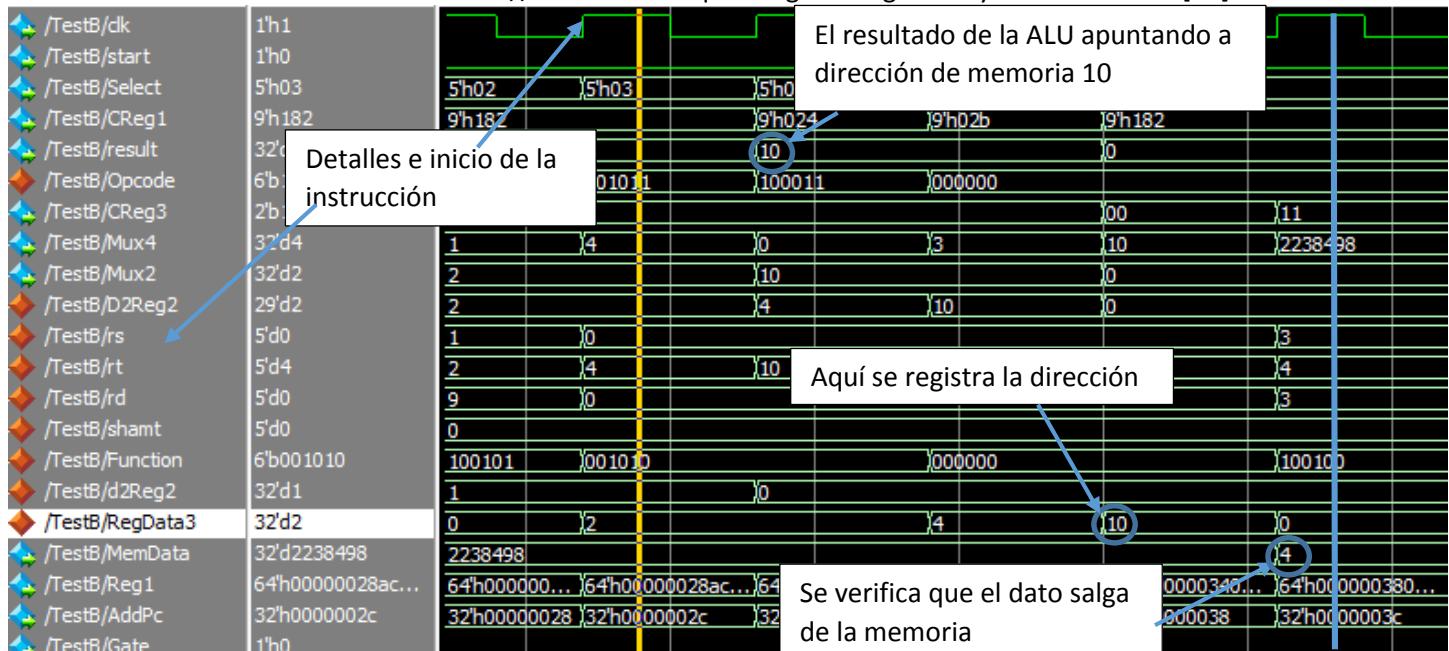
000000000010001001000000100101 // xor lo que tengo en registro 1 y 2 y guardar en registro 9



SW RS, OFF16(RT)

$$\text{MEM32(RT + OFF16}_\pm\text{)} = \text{RS}$$

101011000000010000000000000001010 //almacenar lo que tengo en registro 4 y colocarlo mem[10]

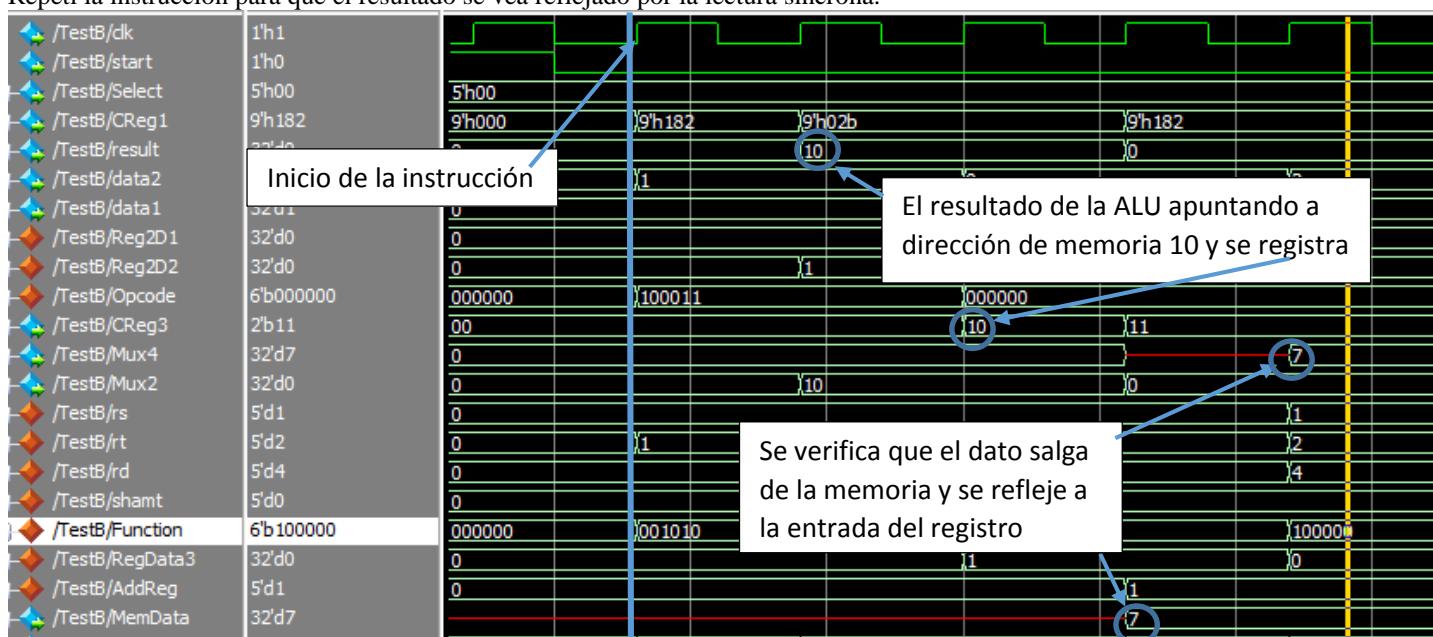


LW RD, OFF16(RS)

$$RD \equiv MEM32(RS + OFF16_{\pm})$$

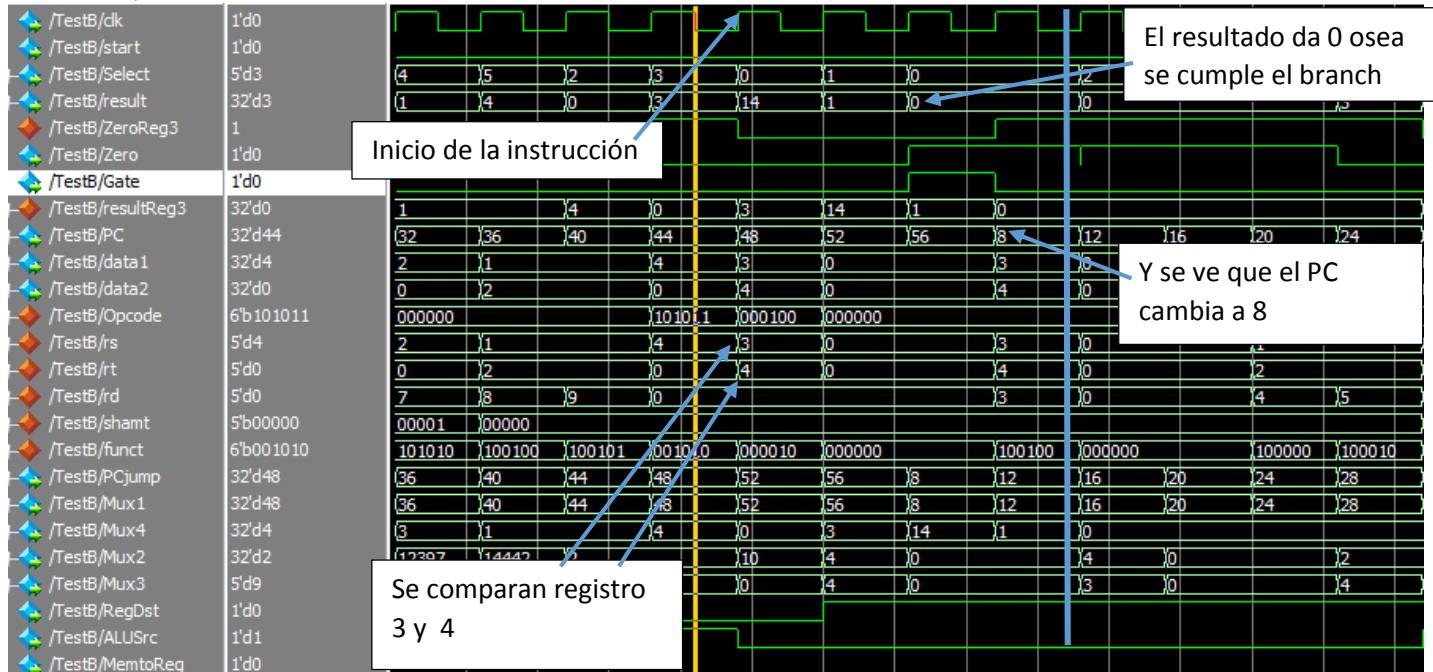
100011000000000000000000000000001010 // carga lo que tengo en localidad de memoria 10 al registro 1.

Repetí la instrucción para que el resultado se vea reflejado por la lectura síncrona.



BEQ RS, RT, OFF18

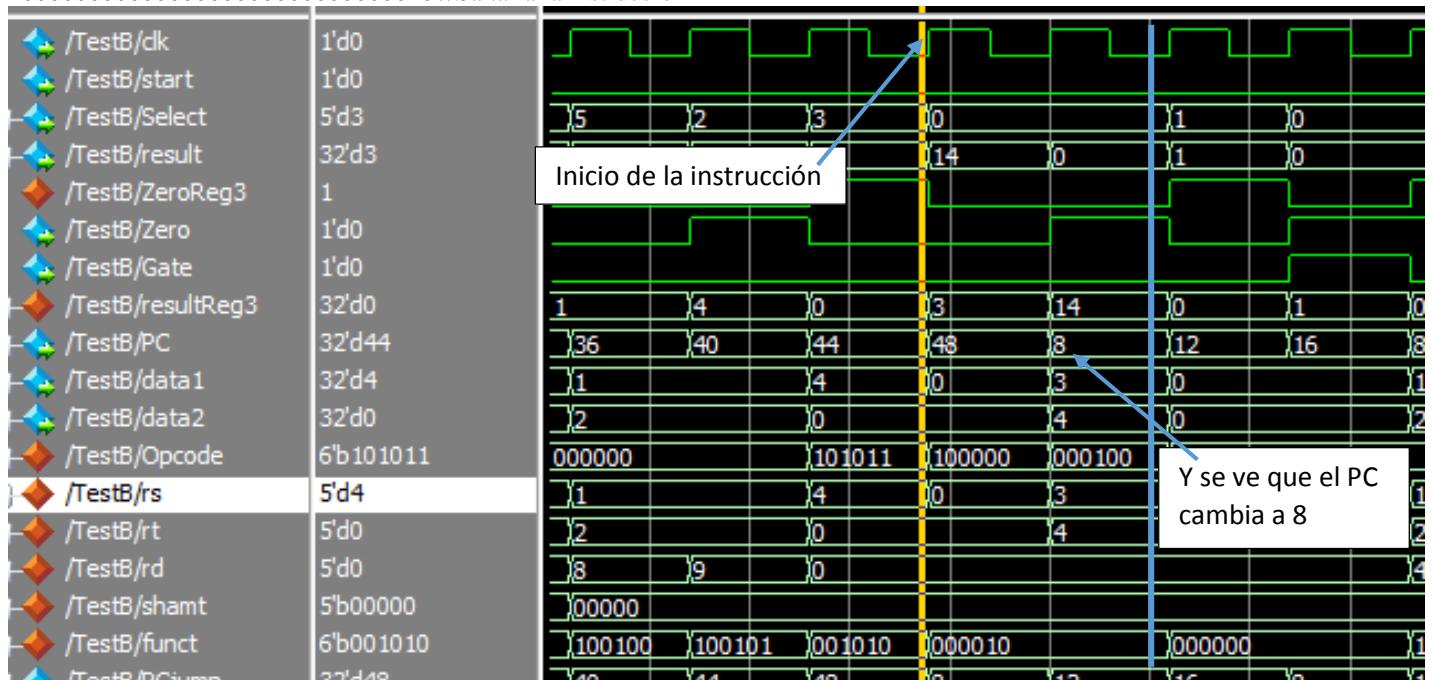
IF RS = RT, PC += OFF18 \pm



J ADDR28

PC = PC[31: 28] :: ADDR28

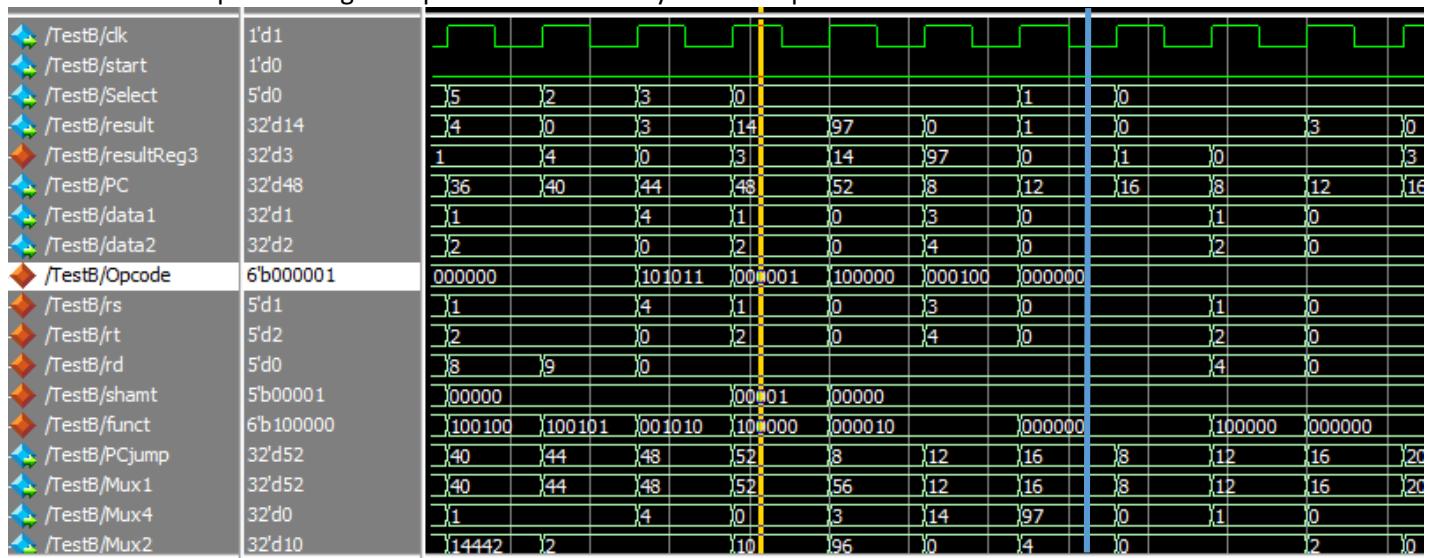
1000000000000000000000000000000010 //Saltar a la instrucion 2



ADDI RD, RS, CONST16

RD = RS + CONST16₁₆

000001000010001000000000001100000 //suma lo que hay en registro 1 y offset(96), el resultado colócalo en registro 2
Si le se verifica el primer diagrama podrá entender muy bien este paso



Slow 1200mV 0C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	114.44 MHz	114.44 MHz	clk	

Flow Summary

Flow Status	Successful - Sat Oct 25 02:30:10 2014
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	Pipeline
Top-level Entity Name	Pipeline
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	1,122 / 114,480 (< 1 %)
Total combinational functions	984 / 114,480 (< 1 %)
Dedicated logic registers	544 / 114,480 (< 1 %)
Total registers	544
Total pins	292 / 529 (55 %)
Total virtual pins	0
Total memory bits	2,048 / 3,981,312 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

```

module Mux1(input [106:0]Reg3,input [31:0]AddPc, input Gate, output [31:0]Mux1);

assign Mux1=(Gate)?Reg3[101:70]:AddPc;

endmodule

module PC(input [31:0]PCjump, input clk,start, output logic [31:0]PC);
    always@(posedge clk or posedge start)
        if(start)
            PC <= 0;
        else
            PC <= PCjump;
Endmodule

module AddPc(input[31:0]PC,output logic [31:0]AddPc);

    always@(PC)
        AddPc=PC+4;

Endmodule

module InstMem (output [31:0]Instruction,input [31:0] PC);

    reg [31:0] InstMemory [0:31];

        initial $readmemb("Instruction.dat",InstMemory);

        assign Instruction = InstMemory [PC[6:2]];

endmodule

module Reg1(input [31:0]AddPc,Instruction, input clk,start, output logic [63:0]Reg1);

    always@(posedge clk or posedge start)
        if(start)Reg1<=0;
        else           Reg1<={AddPc,Instruction};

endmodule

module Registro (output [31:0]data1,data2,input [31:0]Mux4,input [63:0]Reg1,input [70:0]Reg4,input clk,input [31:0]R30In,output [31:0]R31Out);
    reg [31:0] Registro [0:31];

    initial $readmemb("Registro.dat",Registro);

        assign R31Out = Registro[31];
        assign data1 = Registro[Reg1[25:21]];
        assign data2 = Registro[Reg1[20:16]];

    always@(posedge clk) begin
        if ((Reg4[70]==1) && (Reg4!=0) && (Reg4!=30))
            Registro[Reg4] <= Mux4;
        else if(Reg4==0)
            Registro[Reg4] <= 0;
        else if(Reg4==30)

```

```

    Registro[Reg4] <= R30In;
end
endmodule
module Reg2(module Add(module Mux2(module Mux3(module ALU (

```

```

module ALUControl (input [146:0]Reg2,output logic[4:0]Select);
logic [1:0]ALUOp;
assign ALUOp={Reg2[145],Reg2[144]};

always@(ALUOp or Reg2[15:10])
begin
    if ( ALUOp == 2'b10)
        begin
            case(Reg2[15:10])
                6'b100000: Select = 0;//Suma ALU
                6'b100010: Select = 1;//Resta ALU
                6'b100100: Select = 2;//AND
                6'b100101: Select = 3;//XOR
                6'b101101: Select = 4;//Corrimiento Dere
                6'b101010: Select = 5;//Corrimiento Izq
                default: Select = 0;
            endcase

        end
    else if (ALUOp == 2'b01)
        Select = 1;
    else
        Select = 0;
end

endmodule

module Reg3(input [31:0]result,Add,input [4:0]Mux3, input Zero, input [146:0]Reg2, input clk,start,
output logic [106:0]Reg3);

always@(posedge clk or posedge start)

    if(start)Reg3<=0;
    else           Reg3<={Reg2[142:138],Add,Zero,result,Reg2[73:42],Mux3};

endmodule
module Memory(input clk, output logic [31:0]MemData, input [106:0]Reg3);
reg [31:0] mem [0:31];
initial $readmemb("memoria.dat",mem);

    always@(posedge clk)
        if(Reg3[104]) mem[Reg3[41:37]] <= Reg3[36:5];
    always@(posedge clk)
        if(Reg3[105]) MemData <=mem[Reg3[41:37]];

endmodule

```

```

module Reg4(module Mux4(module Control (

```

```

        ALUOp0=0;
        Jump=0;
        MemRead=0;
        end
6'b000100: begin//Opcion de branch
        RegDst=0;
        ALUSrc=0;
        MemtoReg=0;
        RegWrite=0;
        MemWrite=0;
        Branch=1;
        ALUOp1=0;
        ALUOp0=1;
        Jump=0;
        MemRead=0;
        end
6'b100000: begin //Opcion de Jump
        RegDst=0;
        ALUSrc=0;
        MemtoReg=0;
        RegWrite=0;
        MemWrite=0;
        Branch=0;
        ALUOp1=0;
        ALUOp0=0;
        Jump=1;
        MemRead=0;
        end
6'b000001: begin // R-format
        RegDst=0;
        ALUOp1=1;
        ALUOp0=0;
        ALUSrc=1;
        Branch=0;
        MemtoReg=0;
        RegWrite=1;
        MemWrite=0;
        Jump=0;
        MemRead=0;
        end

endcase
end
endmodule

```

```

module Gate(input [106:0]Reg3,input Zero, output logic Gate);
assign Gate=Zero&Reg3[106];
endmodule

```

```

module IJump(endmodule

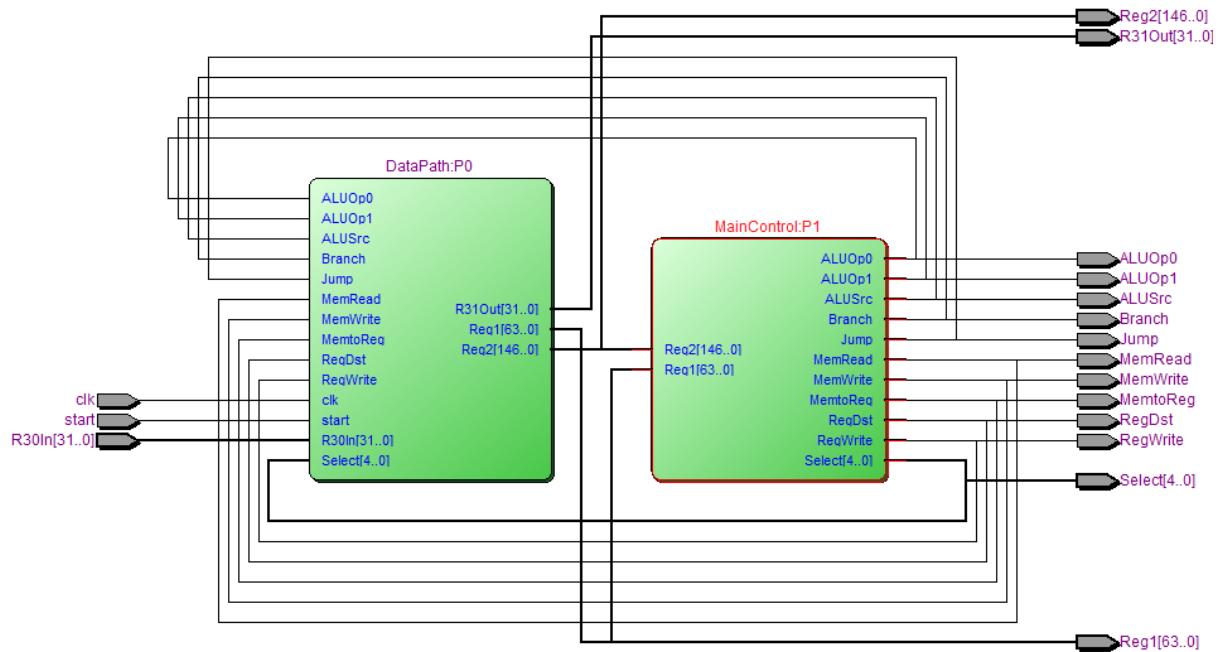
module DataPath(Endmodule

module MainControl(Endmodule

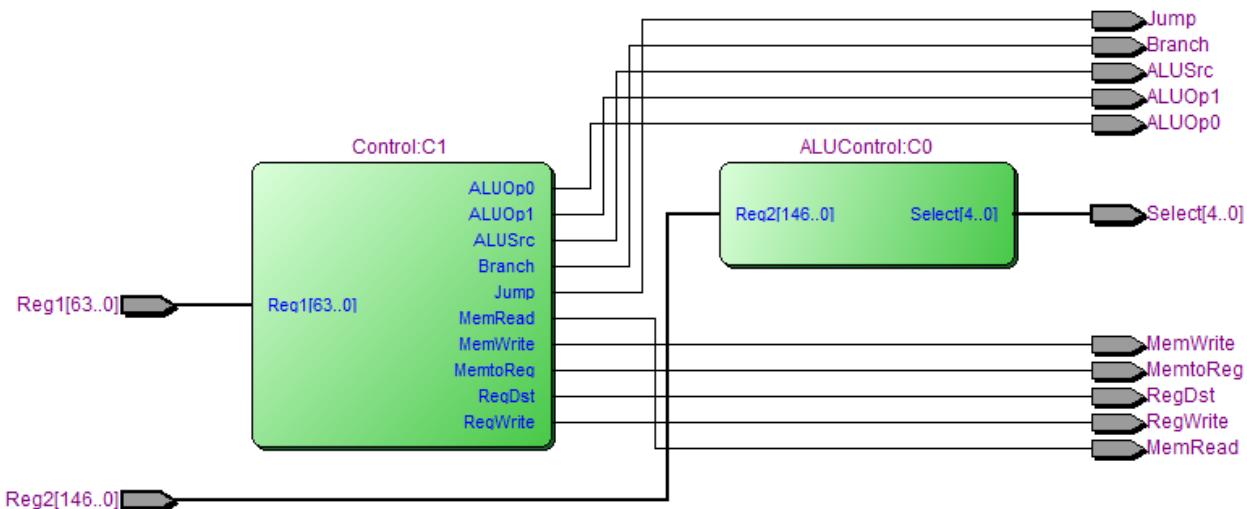
module Pipeline(Endmodule

```

Herarquia Pipeline



Main Control



DataPath

