# New Code

## Memory Subsystem

### Mar

A computer screen with white text

Description automatically generated

### RAM

A computer screen shot of a program

Description automatically generated  
A computer screen with white text

Description automatically generated

## Select and Encode Logic

A computer screen shot of a black screen

Description automatically generated

## Revision to R0

A screenshot of a computer program

Description automatically generated

## CON FF Logic

A computer screen shot of a program

Description automatically generated

## Input and Output Ports

### Output Port

Note that an output port is implemented using a regular register.

A computer screen shot of a black screen

Description automatically generated

### Input Port

Note that we implemented a strobe signal for the Input.

A computer screen shot of a black screen

Description automatically generated

## Updated Datapath

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of white text

Description automatically generated

# Testing

## 1 Load Instructions

### Load Testbench

`timescale 1ns/10ps

module ld\_and\_st\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

initial begin

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s7;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s1;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s11;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

delay: begin

Read <= 1; MDRin <= 1;

#10 strobe <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; MARin <= 1; // see the addition result on the bus

#10 Zlowout <= 0; MARin <= 0;

end

s7 : begin //t6

MDRin <= 1; Read <= 1;

#10 MDRin <= 0; Read <= 0;

end

s8 : begin //t7

Gra <= 1; Rin <= 1; MDRout <= 1; //see the contents of memory on the bus

#10 Gra <= 0; Rin <= 0; MDRout <= 0;

end

s9 : begin

Read <= 1; RAMout = 1; //read contents just written to RAM onto the bus

#10 Read <= 0; RAMout = 0;

end

s10 : begin

Mdatain <= 32'd50; //this is our compare value for the branch

Read <= 1; MDRin <= 1;

#10 Read <= 0; MDRin <= 0;

end

s11 : begin

IRin <= {11'b0, 2'b10, 19'b0};//branch if positive

MDRout <= 1; //put compare value on the bus

#10 MDRout <= 0;//should see a positive branch compare value

end

endcase

end

endmodule

### Load Immediate Testbench

`timescale 1ns/10ps

module ldi\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

initial begin

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s1;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s1;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s11;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

delay: begin

Read <= 1; MDRin <= 1;

#10 strobe <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0;

end

s7 : begin //t6

MDRin <= 1; Read <= 1;

#10 MDRin <= 0; Read <= 0;

end

s8 : begin //t7

Gra <= 1; Rin <= 1; MDRout <= 1; //see the contents of memory on the bus

#10 Gra <= 0; Rin <= 0; MDRout <= 0;

end

s9 : begin

Read <= 1; RAMout = 1; //read contents just written to RAM onto the bus

#10 Read <= 0; RAMout = 0;

end

s10 : begin

Mdatain <= 32'd50; //this is our compare value for the branch

Read <= 1; MDRin <= 1;

#10 Read <= 0; MDRin <= 0;

end

s11 : begin

IRin <= {11'b0, 2'b10, 19'b0};//branch if positive

MDRout <= 1; //put compare value on the bus

#10 MDRout <= 0;//should see a positive branch compare value

end

endcase

end

endmodule

### ld R2, 0x95

A screenshot of a computer

Description automatically generated

### ld R0, 0x38(R2)

A screenshot of a computer

Description automatically generated

### ldi R2, 0x95

A screenshot of a computer

Description automatically generated

### ldi R0, 0x38(R2)

A screenshot of a computer

Description automatically generated

## 2 Store Instructions

### Testbench

`timescale 1ns/10ps

module store\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101, s12 = 4'b1110, s13 = 4'b1111;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

initial begin

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s7;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s9;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s11;

s11 : #40 Present\_state = s12;

s12 : #40 Present\_state = s13;

s13 : #40 Present\_state = s7;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0;

end

s7 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s8 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s9 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s10 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s11 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s12 : begin //t5

Zlowout <= 1; MARin <= 1; // see the addition result on the bus

#10 Zlowout <= 0; MARin <= 0;

end

s13 : begin //t6

MDRin <= 1; Write <= 1; Rout <= 1; Gra <= 1;

#10 MDRin <= 0; Write <= 0; Rout <= 0; Gra <= 0;

end

endcase

end

endmodule

### st 0x87, R1

A screenshot of a computer program

Description automatically generated

### st 0x87(R1), R1

A screenshot of a computer program

Description automatically generated

## 3 ALU Immediate Instructions

### Testbench

`timescale 1ns/10ps

module alu\_instr\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, delay = 4'b1101;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

initial begin

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s1;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s1;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s11;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

delay: begin

Read <= 1; MDRin <= 1;

#10 strobe <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0;

end

s7 : begin //t6

MDRin <= 1; Read <= 1;

#10 MDRin <= 0; Read <= 0;

end

s8 : begin //t7

Gra <= 1; Rin <= 1; MDRout <= 1; //see the contents of memory on the bus

#10 Gra <= 0; Rin <= 0; MDRout <= 0;

end

s9 : begin

Read <= 1; RAMout = 1; //read contents just written to RAM onto the bus

#10 Read <= 0; RAMout = 0;

end

s10 : begin

Mdatain <= 32'd50; //this is our compare value for the branch

Read <= 1; MDRin <= 1;

#10 Read <= 0; MDRin <= 0;

end

s11 : begin

IRin <= {11'b0, 2'b10, 19'b0};//branch if positive

MDRout <= 1; //put compare value on the bus

#10 MDRout <= 0;//should see a positive branch compare value

end

endcase

end

endmodule

### addi R3, R4, -5

A screenshot of a computer

Description automatically generated

### andi R3, R4, 0x53

A screenshot of a computer

Description automatically generated

### ori R3, R4, 0x53

A screenshot of a computer

Description automatically generated

## 4 Branch Instructions

### Testbench for Branch Instructions

`timescale 1ns/10ps

module br\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, LOin, ZHighin, Zlowin, CONin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

integer flag;

initial begin

flag = 0;

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s7;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s7;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s9;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s11;

s11 : #40 Present\_state = s12;

s12 : #40 Present\_state = s13;

s13 : #40 Present\_state = (flag == 2) ? s1 : s7;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, LOin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

CONin <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

flag <= 0;

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0;

end

s7 : begin //t0

flag <= flag + 1;

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s8 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s9 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s10 : begin //t3

Gra <= 1; Rout <= 1;

#10 CONin <= 1;

#20 Gra <= 0; Rout <= 0; CONin <= 0;

end

s11 : begin //t4

PCout <= 1; Yin <= 1;

#20 PCout <= 0; Yin <= 0;

end

s12 : begin //t5

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#20 Cout<= 0; ZHighin <= 0; Zlowin <= 0;

end

s13 : begin //t6

Zlowout <= 1;//This is where I need to use the Conin logic to figure out how to get this into PCin

#20 Zlowout <= 0;

end

endcase

end

endmodule

### brzr R5, 14

A screen shot of a computer

Description automatically generated

### brnr R5, 14

A screenshot of a computer

Description automatically generated

### brpl R5, 14

A screenshot of a computer screen

Description automatically generated

### brmi R5, 14

A screenshot of a computer screen

Description automatically generated

## 5 Jump Instructions

### Testbench

`timescale 1ns/10ps

module jr\_jal\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111, s15 = 5'b10000;

reg [4:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

reg flag;

initial begin

flag = 0;

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = flag ? s11 : s7;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s9;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s1;

s11 : #40 Present\_state = s12;

s12 : #40 Present\_state = s13;

s13 : #40 Present\_state = s15;

s15 : #40 Present\_state = s14;

s14 : #40 Present\_state = s1;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0;

end

s7 : begin //t0

flag <= 1;

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s8 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s9 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s10 : begin //t3

Gra <= 1; Rout <= 1; PCin <= 1;

#20 Gra <= 0; Rout <= 0; PCin <= 0;

end

s11 : begin //t0

flag <= 0;

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s12 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1; Rin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0; Rin <= 0;

end

s13 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s15 : begin

#10 PCout <= 1;

#20 PCout <= 0;

end

s14 : begin //t3

Gra <= 1; Rout <= 1; PCin <= 1;

#20 Gra <= 0; Rout <= 0; PCin <= 0;

end

endcase

end

endmodule

### jr R6

A screenshot of a computer

Description automatically generated

### jal R6

A screen shot of a computer

Description automatically generated

## 6 Special Instructions

### Testbench

`timescale 1ns/10ps

module mfhi\_mflo\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, LOin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

reg flag;

initial begin

flag = 0;

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s7;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s9;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s1;

s11 : #40 Present\_state = s12;

s12 : #40 Present\_state = s13;

s13 : #40 Present\_state = s14;

s14 : #40 Present\_state = s1;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, LOin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

if(flag == 0) HIin <= 1;

else LOin <= 1;

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0; HIin <= 0; LOin <= 0;

end

s7 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s8 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s9 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s10 : begin //t3

if (flag == 0) HIOut <= 1;

else LOout <= 1;

Gra <= 1; Rin <= 1;

#20 Gra <= 0; Rin <= 0; HIOut <= 0; LOout <= 0;

flag = 1;

end

endcase

end

endmodule

### mfhi R6

A screenshot of a computer

Description automatically generated

### mflo R7

A screenshot of a computer

Description automatically generated

## 7 Input/Output Instructions

### Testbench

`timescale 1ns/10ps

module in\_out\_tb;

reg PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout, RAMout, CONin;

wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out, R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out;

reg MARin, Zin, PCin, MDRin, IRin, OutPortin, Yin, IncPC, Read, Write, AND, HIin, Loin, ZHighin, Zlowin;

wire R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in, R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

reg Clock, clear, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, Cout;

wire branchCompare;

wire [3:0] to\_decode;

reg [31:0] Mdatain, input\_data;

parameter Default = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011,

s4 = 4'b0100, s5 = 4'b0101, s6 = 4'b0110, s7 = 4'b0111, s8 = 4'b1000, s9 = 4'b1001, s10 = 4'b1010, s11 = 4'b1011, sclear = 4'b1100, s12 = 4'b1101, s13 = 4'b1110, s14 = 4'b1111;

reg [3:0] Present\_state = Default;

reg [4:0] op;

wire [31:0] BusOut, mdrData, BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7,

BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

ZHighWire, ZLowWire;

//wire R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out, R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in;

data\_path DUT(Clock, clear, Read, Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout, CONin,

input\_data,IRin,

op,

HIOut, LOout, Zhighout, Zlowout, PCout, MDRout, InPortout, Yout, RAMout, Cout,

HIin, LOin, ZHighin, Zlowin, PCin, MDRin, OutPortin, Yin, MARin, IncPC,

BusOut, mdrData, ZHighWire, ZLowWire,

BusMuxInR0, BusMuxInR1, BusMuxInR2, BusMuxInR3, BusMuxInR4, BusMuxInR5, BusMuxInR6, BusMuxInR7, BusMuxInR8, BusMuxInR9, BusMuxInR10, BusMuxInR11, BusMuxInR12, BusMuxInR13, BusMuxInR14, BusMuxInR15,

BusMuxInZhigh, BusMuxInZlow, BusMuxInPCout, BusMuxInInPortout, BusMuxInYout, BusMuxInHI, BusMuxInLO, BusMuxInRamout, output\_data, irOut,

branchCompare,

R0out, R1out, R2out, R3out, R4out, R5out, R6out, R7out,R8out, R9out, R10out, R11out, R12out, R13out, R14out, R15out,

R0in, R1in, R2in, R3in, R4in, R5in, R6in, R7in,R8in, R9in, R10in, R11in, R12in, R13in, R14in, R15in,

to\_decode);

initial begin

Clock = 1;

end

always #5 Clock = ~Clock;

always @(negedge Clock) begin// finite state machine; if clock falling-edge so as to be offset from reg clocking

case (Present\_state)

Default : #40 Present\_state = sclear;

sclear : #40 Present\_state = s1;

s1 : #40 Present\_state = s2;

s2 : #40 Present\_state = s3;

s3 : #40 Present\_state = s4;

s4 : #40 Present\_state = s5;

s5 : #40 Present\_state = s6;

s6 : #40 Present\_state = s7;

s7 : #40 Present\_state = s8;

s8 : #40 Present\_state = s9;

s9 : #40 Present\_state = s10;

s10 : #40 Present\_state = s11;

s11 : #40 Present\_state = s12;

s12 : #40 Present\_state = s13;

s13 : #40 Present\_state = s14;

endcase

end

always @(Present\_state) begin // do the required job in each state

case (Present\_state) // assert the required signals in each clock cycle

Default: begin

/\*PCout <= 0; Zlowout <= 0; MDRout <= 0; // initialize the signals

R2out <= 0; R3out <= 0; MARin <= 0; Zin <= 0;

PCin <=0; MDRin <= 0; IRin <= 0; Yin <= 0;

IncPC <= 0; Read <= 0; AND <= 0;

R1in <= 0; R2in <= 0; R3in <= 0; Mdatain <= 32'h00000000;\*/

{Write, strobe, BAOut, Gra, Grb, Grc, Rin, Rout} <= 8'b0;

{PCout, Zhighout, Zlowout, MDRout, HIOut, LOout, InPortout, Yout} <= 8'b0;

{MARin, Zin, PCin, MDRin, IRin, Yin, IncPC, Read, AND, HIin, InPortout, Loin, ZHighin, Zlowin} <= 13'b0;

clear<=0;

Mdatain <= 32'h00000000;

BAOut <= 0;

end

sclear : begin

clear <= 1;

#10 clear <= 0;

end

s1 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s2 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s3 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s4 : begin //t3

Grb <= 1; BAOut <= 1;

#10 Yin <= 1; //see the value of the reg to be added

#10 Grb <= 0; BAOut <= 0; Yin <= 0;

end

s5 : begin //t4 see C on the bus

Cout<= 1; ZHighin <= 1; Zlowin <= 1;

#10 op <= 5'b00011;

#20 Cout<= 0; op <= 5'b00000; ZHighin <= 0; Zlowin <= 0;

end

s6 : begin //t5

Zlowout <= 1; Gra <= 1; Rin <= 1;// see the addition result on the bus

#10 Zlowout <= 0; Gra <= 0; Rin <= 0;

end

s7 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s8 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s9 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s10 : begin //t3

Gra <= 1; Rout <= 1; OutPortin <= 1;

#20 Gra <= 0; Rout <= 0; OutPortin <= 0;

end

s11 : begin //t0

PCout <= 1; MARin <= 1; IncPC <= 1; ZHighin <= 1; Zlowin <= 1; //see the PC on the bus

#10 PCout <= 0; MARin <= 0; IncPC <= 0; ZHighin <= 0; Zlowin <= 0;

end

s12 : begin //t1

Zhighout <= 0; Zlowout <= 1; PCin <= 1; //see the value of PC plus one on the bus

#10 Read <= 1; MDRin <= 1;

#20 Zhighout <= 0; Zlowout <= 0; PCin <= 0;Read <= 0; MDRin <= 0;

end

s13 : begin //t2

MDRout <= 1; Read <= 0; MDRin <= 0; //the the instruction on the bus

#10 IRin <= 1;

#20 MDRout <= 0; IRin <= 0;

end

s14 : begin //t3

input\_data <= 32'habba;

strobe <= 1;

#5 strobe <= 0; Gra <= 1; Rin <= 1; InPortout <= 1;

#20 Gra <= 1; Rin <= 1; InPortout <= 0;

end

endcase

end

endmodule

### out R3

A screenshot of a computer

Description automatically generated

### in R4

A screenshot of a computer

Description automatically generated