实验5 简单计算机系统-系统设计C-代码

电 25 吴晨聪 2022010311

# 1. PC.v

module PC(clk,rst\_n,branch,jump,imm,pcout);

input clk;

input rst\_n;

input branch;

input jump;

input [7:0]imm;

output reg[7:0]pcout;

initial begin

pcout=0;

end

always@(posedge clk or negedge rst\_n) begin

if (!rst\_n)

pcout=0;

else begin

if(jump) begin

pcout=imm;

end

else if(branch) begin

pcout=pcout+imm+1;

end

else if(pcout < 255) begin

pcout=pcout+1;

end

if (pcout>255) begin

pcout=0;

end

end

end

endmodule

# 2. controller2.v

module controller2(Op,zero,alucs,flagwrite,regwrite,selscrB,redges,memtoreg,wren,branch,jump);

input [3:0]Op;

input zero;

output reg [2:0]alucs;

output reg flagwrite;

output reg regwrite;

output reg selscrB;

output reg redges;

output reg memtoreg;

output reg wren;

output reg branch;

output reg jump;

initial begin

wren=0;

regwrite=0;

branch=0;

jump=0;

end

always@(\*) begin

if(Op>=0 && Op<=6) begin

regwrite=1;

selscrB=0;

redges=1;

memtoreg=0;

wren=0;

alucs=Op[2:0];

branch=0;

jump=0;

if(Op==2 || Op==3 || Op==5 || Op==6) begin

flagwrite=1;

end

else begin

flagwrite=0;

end

end

else if(Op>=8 && Op<=10) begin

regwrite=1;

selscrB=1;

redges=0;

memtoreg=0;

wren=0;

alucs=Op[2:0];

flagwrite=Op[1];

branch=0;

jump=0;

end

else if(Op==11) begin

wren=0;

regwrite=1;

selscrB=1;

redges=0;

memtoreg=1;

alucs=2;

flagwrite=1;

branch=0;

jump=0;

end

else if(Op==12) begin

wren=1;

regwrite=0;

selscrB=1;

redges=0;

memtoreg=1;

alucs=2;

flagwrite=1;

branch=0;

jump=0;

end

else if(Op==13) begin

wren=0;

regwrite=0;

selscrB=0;

redges=0;

memtoreg=0;

alucs=3;

flagwrite=0;

jump=0;

if(zero==1) begin

branch=1;

end

else begin

branch=0;

end

end

else if(Op==14) begin

wren=0;

regwrite=0;

selscrB=0;

redges=0;

memtoreg=0;

alucs=3;

flagwrite=0;

jump=0;

if(zero==1) begin

branch=0;

end

else begin

branch=1;

end

end

else if(Op==7) begin

wren=0;

regwrite=0;

selscrB=0;

redges=0;

memtoreg=0;

alucs=2;

flagwrite=0;

branch=0;

jump=1;

end

else begin

wren=0;

regwrite=0;

branch=0;

jump=0;

end

end

endmodule

# 3. cpuF.v

module cpuF(aclr,clk,rst\_n,s,zeroout);

input aclr;

wire [7:0]scrA;

wire [7:0]scrB;

reg [1:0]Rs;

reg [1:0]Rt;

reg [1:0]Rd;

reg [3:0]Op;

wire [7:0]memData;

wire selscrB;

wire redges;

wire memtoreg;

reg [7:0]imm;

wire regwrite;

wire [2:0]alucs;

input clk;

input rst\_n;

wire flagwrite;

wire [1:0]nd;

wire [7:0]di;

wire [7:0]q2;

wire carry\_out;

wire carry\_in;

wire zeroin;

wire wren;

wire branch;

wire jump;

output [7:0]s;

output zeroout;

wire [15:0]q;

wire [7:0]pcout;

always@(posedge clk or negedge rst\_n) begin

if(!rst\_n) begin

end

else begin

end

end

always@(\*) begin

Op=q[15:12];

Rs=q[11:10];

Rt=q[9:8];

Rd=q[7:6];

imm=q[7:0];

end

mux2 muxscrB(

.d0(q2),

.d1(imm),

.sel(selscrB),

.y(scrB)

);

mux1 muxnd(

.d0(Rt),

.d1(Rd),

.sel(redges),

.y(nd)

);

mux2 muxdi(

.d0(s),

.d1(memData),

.sel(memtoreg),

.y(di)

);

regfile regfile(

.clk(clk),

.rst\_n(rst\_n),

.n1(Rs),

.n2(Rt),

.nd(nd),

.di(di),

.reg\_we(regwrite),

.q1(scrA),

.q2(q2)

);

alu alu(

.data\_a(scrA),

.data\_b(scrB),

.s(s),

.zero(zeroin),

.cs(alucs),

.carry\_in(carry\_in),

.carry\_out(carry\_out)

);

flag flag(

.clk(clk),

.rst\_n(rst\_n),

.zeroin(zeroin),

.flagwrite(flagwrite),

.flagin(carry\_out),

.flagout(carry\_in),

.zeroout(zeroout)

);

PC PC(

.clk(clk),

.rst\_n(rst\_n),

.branch(branch),

.jump(jump),

.imm(imm),

.pcout(pcout)

);

cpurom cpu\_rom\_inst(

.address(pcout),

.clock(clk),

.q(q)

);

controller2 controller2(

.zero(zeroin),

.Op(Op),

.alucs(alucs),

.flagwrite(flagwrite),

.regwrite(regwrite),

.selscrB(selscrB),

.redges(redges),

.memtoreg(memtoreg),

.wren(wren),

.branch(branch),

.jump(jump)

);

cpuram cpu\_ram\_inst(

.aclr(aclr),

.wren(wren),

.address(s),

.data(q2),

.clock(clk),

.q(memData)

);

endmodule

# 4. cpuF\_tb.v

`timescale 1ns/1ps

module cpuF\_tb;

reg clk;

reg rst\_n;

reg aclr;

wire [7:0]s;

wire zeroout;

initial begin

aclr=0;

clk=1;

rst\_n=0;

#20 rst\_n=1;

end

always #10 clk = ~clk;

cpuF cpuF(

.aclr(aclr),

.clk(clk),

.rst\_n(rst\_n),

.s(s),

.zeroout(zeroout)

);

endmodule