实验6 简单计算机系统-系统设计D-代码

电 25 吴晨聪 2022010311

# 1. PC2.v

module PC2(clk,rst\_n,branch,jump,imm,pcout);

input clk;

input rst\_n;

input branch;

input jump;

input [15:0]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. mux12.v

module mux12(d0,d1,sel,y);

input [4:0]d0;

input [4:0]d1;

input sel;

output reg [4:0]y;

always@(\*) begin

if(!sel) begin

y=d0;

end

else begin

y=d1;

end

end

endmodule

# 3. mux22.v

module mux22(d0,d1,sel,y);

input [31:0]d0;

input [31:0]d1;

input sel;

output reg [31:0]y;

always@(\*) begin

if(!sel) begin

y=d0;

end

else begin

y=d1;

end

end

endmodule

# 4. moveImm.v

module moveImm(imm,immout);

input [15:0]imm;

output reg[31:0]immout;

always@(\*) begin

immout[15:0]=imm;

immout[31:16]=0;

end

endmodule

# 5. cpuG.v

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

input aclr;

wire [31:0]scrA;

wire [31:0]scrB;

reg [4:0]Rs;

reg [4:0]Rt;

reg [4:0]Rd;

reg [5:0]Op;

reg [5:0]funct;

reg [5:0]shamt;

reg [25:0]addr;

wire [31:0]memData;

wire selscrB;

wire redges;

wire memtoreg;

reg [15:0]imm;

wire [31:0]immout;

wire regwrite;

wire [3:0]alucs;

input clk;

input rst\_n;

wire flagwrite;

wire [4:0]nd;

wire [31:0]di;

wire [31:0]q2;

wire carry\_out;

wire carry\_in;

wire zeroin;

wire wren;

wire branch;

wire jump;

output [31:0]s;

output zeroout;

wire [31:0]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[31:26];

Rs=q[25:21];

Rt=q[20:16];

Rd=q[15:11];

shamt=q[10:6];

funct=q[5:0];

imm=q[15:0];

addr=q[25:0];

end

moveImm moveImm(

.imm(imm),

.immout(immout)

);

mux22 muxscrB(

.d0(q2),

.d1(immout),

.sel(selscrB),

.y(scrB)

);

mux12 muxnd(

.d0(Rt),

.d1(Rd),

.sel(redges),

.y(nd)

);

mux22 muxdi(

.d0(s),

.d1(memData),

.sel(memtoreg),

.y(di)

);

regfile2 regfile(

.clk(clk),

.rst\_n(rst\_n),

.n1(Rs),

.n2(Rt),

.nd(nd),

.di(di),

.reg\_we(regwrite),

.q1(scrA),

.q2(q2)

);

alu2 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)

);

PC2 PC2(

.clk(clk),

.rst\_n(rst\_n),

.branch(branch),

.jump(jump),

.imm(imm),

.pcout(pcout)

);

cpurom2 cpu\_rom\_inst(

.address(pcout),

.clock(clk),

.q(q)

);

controller22 controller22(

.zero(zeroin),

.Op(Op),

.funct(funct),

.alucs(alucs),

.flagwrite(flagwrite),

.regwrite(regwrite),

.selscrB(selscrB),

.redges(redges),

.memtoreg(memtoreg),

.wren(wren),

.branch(branch),

.jump(jump)

);

cpuram2 cpu\_ram\_inst(

.aclr(aclr),

.wren(wren),

.address(s),

.data(q2),

.clock(clk),

.q(memData)

);

endmodule

# 6. cpuG\_tb.v

`timescale 1ns/1ps

module cpuG\_tb;

reg clk;

reg rst\_n;

reg aclr;

wire [31:0]s;

wire zeroout;

initial begin

aclr=0;

clk=1;

rst\_n=0;

#20 rst\_n=1;

end

always #10 clk = ~clk;

cpuG cpuG(

.aclr(aclr),

.clk(clk),

.rst\_n(rst\_n),

.s(s),

.zeroout(zeroout)

);

endmodule

# 7. controller22.v

module controller22(Op,zero,funct,alucs,flagwrite,regwrite,selscrB,redges,memtoreg,wren,branch,jump);

input [5:0]Op;

input zero;

input [5:0]funct; //多了个输入funct

output reg [4:0]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) begin //Op=0表示R-Type

regwrite=1;

selscrB=0;

redges=1;

memtoreg=0;

wren=0;

branch=0;

jump=0;

if(funct==32) begin //R-add

flagwrite=1;

alucs=2;

end

else if(funct==34) begin //R-sub

flagwrite=1;

alucs=3;

end

else if(funct==42) begin //R-sel

flagwrite=1;

alucs=4;

end

else begin

flagwrite=0;

alucs=0;

end

end

else if(Op==8) begin //I-add

regwrite=1;

selscrB=1;

redges=0;

memtoreg=0;

wren=0;

alucs=2;

flagwrite=1;

branch=0;

jump=0;

end

else if(Op==9) begin //I-add-unsign

regwrite=1;

selscrB=1;

redges=0;

memtoreg=0;

wren=0;

alucs=2;

flagwrite=1;

branch=0;

jump=0;

end

else if(Op==12) begin //I-and-unsign

regwrite=1;

selscrB=1;

redges=0;

memtoreg=0;

wren=0;

alucs=0;

flagwrite=1;

branch=0;

jump=0;

end

else if(Op==4) begin //BEQ相等时跳转

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==5) begin //BNE不等时跳转

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==2) begin //跳转

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