### CU

module CU(input [31:0] c,output regdst,output jump,output regwr,output bne,output branch,output memtoreg,output [5:0]aluop,output memwr,output memrd,output alusrc);

parameter //SPECIAL OP LIST 5-0

ADDU = 6'b100001,//1

SUBU = 6'b100011,//

ADD = 6'b100000,//

SUB = 6'b100010,//

AND = 6'b100100,//

OR = 6'b100101,//

XOR = 6'b100110,//

NOR = 6'b100111,//

SLT = 6'b101010,//

SLTU = 6'b101011,//

SRL = 6'b000010,//

SRA = 6'b000011,//

SLL = 6'b000000,//

SLLV = 6'b000100,

SRLV = 6'b000110,

SRAV = 6'b000111,

JR = 6'b001000,

JALR = 6'b001001,

MULT = 6'b011000,

MULTU = 6'b011001,

DIV = 6'b011010,

DIVU = 6'b011011,//

MFHI = 6'b010000,

MFLO = 6'b010010,

MTHI = 6'b010001,

MTLO = 6'b010011,

BREAK = 6'b001101,

SYSCALL = 6'b001100,

TEQ = 6'b110100,

//SPECIAL 2 func

CLZ = 6'b100000,

MUL = 6'b000010,

//REGIMM OP LIST 20-16

BLTZ = 5'b00000,

BGEZ = 5'b00001,

//COP0 OP LIST

ERET = 6'b011000, //5-0&&25TH=1

MFC0 = 5'b00000, //20-16

MTC0 = 5'b00100,

//OPCODE FIELD 31-26

ADDI = 6'b001000,//

ADDIU = 6'b001001,//

ANDI = 6'b001100,//

ORI = 6'b001101,//

XORI = 6'b001110,//

LW = 6'b100011,

SW = 6'b101011,

BEQ = 6'b000100,

BNE = 6'b000101,

BLEZ = 6'b000110,

BGTZ = 6'b000111,

SLTI = 6'b001010,

SLTIU = 6'b001011,

LUI = 6'b001111,

J = 6'b000010,

JAL = 6'b000011,

LB = 6'b100000,// Load Byte Function=6'h24

LBU = 6'b100100,// 1Load Byte Unsigned

LH = 6'b100001,// Load high

LHU = 6'b100101,// Load High Unsigned

SB = 6'b101000,// Send Byte

SH = 6'b101001,// Send High

SPECIAL = 6'b000000,

SPECIAL2= 6'b011100,

REGIMM = 6'b000001,

COP0 = 6'b010000;

//alu

parameter \_ADDU = 4'b0000; //r=a+b unsigned

parameter \_ADD = 4'b0010; //r=a+b signed

parameter \_SUBU = 4'b0001; //r=a-b unsigned

parameter \_SUB = 4'b0011; //r=a-b signed

parameter \_AND = 4'b0100; //r=a&b

parameter \_OR = 4'b0101; //r=a|b

parameter \_XOR = 4'b0110; //r=a^b

parameter \_NOR = 4'b0111; //r=~(a|b)

parameter \_LUI = 4'b1000; //r={b[15:0],16'b0}

parameter \_SLT = 4'b1011; //r=(a-b<0)?1:0 signed

parameter \_SLTU = 4'b1010; //r=(a-b<0)?1:0 unsigned

parameter \_SRA = 4'b1100; //r=b>>>a

parameter \_SLL = 4'b1110; //r=b<<a

parameter \_SRL = 4'b1101; //r=b>>a

parameter \_DIVU = 6'b010000;

parameter \_MULTU = 6'b010001;

parameter \_SYSCALL= 4'b1000,

\_BREAK = 4'b1001,

\_TEQ = 4'b1101;

reg c1,c2,c3,c4,c5,c6,c8,c9,c14;

reg[5:0]c7;

reg [5:0]o;

reg [5:0]mo,suanshu;

initial

begin

c14=0;

c1=0;

c2=0;

c3=0;

c4=0;

c5=0;

c6=0;

c7=0;

c8=0;

c9=0;

o=0;

end

always @(\*)

begin

suanshu=0;

o[5:0]=c[31:26];

mo[5:0]=c[5:0];

if(o==0)

begin

o=mo;

suanshu=1;

end

c14=0;

if(suanshu==0)

begin

c14=0;

case(o)

LW:begin c1=0; c3=1;c4=1;c5=1;c6=0;c8=1;c9=0;c2=0;c7=\_ADD;end//over

SW,SB,SH:begin c1=1; c3=0;c4=1;c5=0;c6=1;c8=1;c9=0;c2=0;c7=\_ADD;end//over

BEQ:begin c1=0; c3=0;c4=0;c5=0;c6=0;c8=1;c9=1;c2=0;c7=\_SUB;end

BNE:begin c1=0; c3=0;c4=0;c5=0;c6=0;c8=1;c9=0;c2=0;c7=\_SUB;c14=1;end

J:begin c1=0; c3=0;c4=1;c5=0;c6=0;c8=1;c9=0;c2=1;c7=\_ADD;end

ADDI:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_ADD;end//

ANDI:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_AND;end

ORI:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_OR;end

XORI:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_XOR;end

SLTI:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SLT;end

SLTIU:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SLTU;end

ADDIU:begin c1=0; c3=1;c4=1;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_ADDU;end

endcase

end

else

begin

c14=0;

case(o)

ADD:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_ADD;end

ADDU:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_ADDU;end

SUB:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SUB;end

AND:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_AND;end

OR:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_OR;end

SLT:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SLT;end

XOR:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_XOR;end//new

NOR:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_NOR;end

SLTU:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SLTU;end

SUBU:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SUBU;end

SRAV:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SRA;end

SLLV:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SLL;end

SRLV:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_SRL;end

DIVU:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_DIVU;end

MULTU:begin c1=1; c3=1;c4=0;c5=0;c6=0;c8=0;c9=0;c2=0;c7=\_MULTU;end

endcase

end

end

assign regdst=c1;

assign jump=c2;

assign regwr=c3;

assign alusrc=c4;

assign memrd=c5;

assign memwr=c6;

assign aluop=c7;

assign memtoreg=c8;

assign branch=c9;

assign bne=c14;

endmodule

### 主模块

module test;

wire [31:0] s1,s2,s3,w1,s4,s5,s6,c1,s7,s8,s9,s10,s11,s12,c3,c4,s13,c7,s14,s15,c5,c6,c8,s16,c11,c2,c13,c15;

wire c12,c10,c9,c14;

reg clk,rst,ena\_pc;//pc

initial//pc

begin

rst=0;

clk=0;

end

always #50 clk=~clk;

assign c11=c9&c10;

assign c12=~c10;

assign c13=c14&c12;

assign c15=c11|c13;

reg [31:0]alu1\_a;//alu1

reg[3:0]alu1\_crl;

wire [31:0]alu1\_al;

wire[3:0]alu1\_crll;

initial//alu\_1&&alu\_2

begin

alu1\_a=32'b1;

alu1\_crl=4'b0000;//+

end

assign alu1\_al=alu1\_a;

assign alu1\_crll=alu1\_crl;

reg wena\_im,rena\_im;//IM

wire wena\_iml,rena\_iml;

initial

begin

wena\_im=0;

rena\_im=1;

end

assign wena\_iml=wena\_im;

assign rena\_iml=rena\_im;

ram im(.wena(wena\_iml),.rena(rena\_iml),.addr(s1),.data\_out(s4));

ram data(.rena(c5),.wena(c6),.addr(s14),.data\_in(s10),.data\_out(s15),.clk(clk),.order(s4));

pca pc1(.clk(clk),.rst(rst),.data\_in(s2),.data\_out(s1));

alu alu1(.a(alu1\_al),.b(s1),.r(s3),.aluc(alu1\_crll));

//alu alu2(.a(s3),.b(s8),.r(s9),.aluc(alu1\_crll));

alu alu2(.a(s3),.b(s5),.r(s9),.aluc(alu1\_crll));

alu alu3(.a(s11),.b(s13),.aluc(c7),.r(s14),.zero(c10));

sixteentodou se1(.a(s4),.b(s5));

MUXfour mux1(.a(s4),.b(s4),.crl(c1),.c(s6));//mux1

MUX mux2(.a(s10),.b(s5),.crl(c4),.c(s13));

MUX mux3(.a(s14),.b(s15),.crl(c8),.c(s12));

MUX mux4(.a(s3),.b(s9),.crl(c15),.c(s16));

MUX mux5(.a(s16),.b(s7),.crl(c2),.c(s2));

SHL2 shl(.a(s4),.b(s7),.pcn(s3));

//OSHL2 shl2(.a(s5),.b(s8));

RF rf(.we(c3),.raddr1(s4[25:21]),.raddr2(s4[20:16]),.waddr(s6),. wdata(s12),.rdata1(s11),.rdata2(s10),.clk(clk));

CU cu(.c(s4),.regdst(c1),.jump(c2),.regwr(c3),.alusrc(c4),.memrd(c5),.memwr(c6),.aluop(c7),.memtoreg(c8),.branch(c9),.bne(c14));

endmodule

### alu

module alu(

input [31:0] a, //OP1

input [31:0] b, //OP2

input [5:0] aluc, //controller

output [31:0] r, //result

output zero,

output carry,

output negative,

output overflow);

parameter Addu = 4'b0000; //r=a+b unsigned

parameter Add = 4'b0010; //r=a+b signed

parameter Subu = 4'b0001; //r=a-b unsigned

parameter Sub = 4'b0011; //r=a-b signed

parameter And = 4'b0100; //r=a&b

parameter Or = 4'b0101; //r=a|b

parameter Xor = 4'b0110; //r=a^b

parameter Nor = 4'b0111; //r=~(a|b)

parameter Lui1 = 4'b1000; //r={b[15:0],16'b0}

parameter Lui2 = 4'b1001; //r={b[15:0],16'b0}

parameter Slt = 4'b1011; //r=(a-b<0)?1:0 signed

parameter Sltu = 4'b1010; //r=(a-b<0)?1:0 unsigned

parameter Sra = 4'b1100; //r=b>>>a

parameter Sll = 4'b1110; //r=b<<a

parameter Srl = 4'b1101; //r=b>>a

parameter Divu = 6'b010000; //r=a/b

parameter Multu = 6'b010001;

parameter bits=31;

parameter ENABLE=1,DISABLE=0;

reg [32:0] result;

wire signed [31:0] sa=a,sb=b;

/\*Divu\*/

reg [4:0] count=0;

reg [31:0] reg\_qu=0;

reg [31:0] reg\_ru=0;

reg [31:0] reg\_bu=0;

reg busy=0;

reg r\_signu=0;

wire [31:0]rr,qq;

assign rr = r\_signu? reg\_ru + reg\_bu : reg\_ru;

assign qq = reg\_qu;

wire [32:0] sub\_add ;

assign sub\_add= r\_signu?({reg\_ru,qq[31]} + {1'b0,reg\_bu}):({reg\_ru,qq[31]} - {1'b0,reg\_bu});

/\*\*/

always@(\*)begin

case(aluc)

Addu: begin

result=a+b;

end

Subu: begin

result=a-b;

end

Add: begin

result=sa+sb;

end

Sub: begin

result=sa-sb;

end

Sra: begin

if(a==0) {result[31:0],result[32]}={b,1'b0};

else {result[31:0],result[32]}=sb>>>(a-1);

end

Srl: begin

if(a==0) {result[31:0],result[32]}={b,1'b0};

else {result[31:0],result[32]}=b>>(a-1);

end

Sll: begin

result=b<<a;

end

And: begin

result=a&b;

end

Or: begin

result=a|b;

end

Xor: begin

result=a^b;

end

Nor: begin

result=~(a|b);

end

Sltu: begin

result=a<b?1:0;

end

Slt: begin

result=sa<sb?1:0;

end

Lui1,Lui2: result = {b[15:0], 16'b0};

Divu:begin

result=a/b;

end//DIV

Multu:begin

result=a\*b;

end

default:

result=a+b;

endcase

end

assign r=result[31:0];

assign carry = result[32];

assign zero=(a==b)?1:0;

assign negative=result[31];

assign overflow=result[32];

endmodule

### PC

module pca(

input clk,//??

input rst,//??

input [31:0] data\_in,

output [31:0] data\_out

);

reg [31:0] data;

initial

data=0;

always @(posedge clk) begin

if(rst) data<=32'h00000000; //reset key

else begin

data<=data\_in; //enable ,input

end

end

assign data\_out = data;

endmodule

### JUMP连接器

module SHL2(input [31:0] a,input [31:0] pcn, output[31:0]b);//no <<2and combine

reg [31:0] an;

always@(\*)

begin

an[25:0]=a[25:0];

an[31:26]=pcn[31:26];

end

assign b=an;

endmodule

### RAM

module ram(//IM&&RAM

input wena,

input rena,

input clk,

input [31:0] order,

input [31:0] addr,

input [31:0] data\_in,

output [31:0] data\_out

);

reg [31:0]i;

reg [31:0] state [0:10000];

reg [31:0] whichtoshow;

reg [5:0]writenumber;

initial

begin

whichtoshow=32'b0;

end

parameter

SB = 6'b101000,// Send Byte

SH = 6'b101001;//

initial

begin

for(i=0;i<255;i=i+1)

begin

state[i]=32'h00000000;

$readmemb("F:/dan",state);

end

state[200]=999;

end

always @(addr)

begin

if(rena)

whichtoshow=state[addr];

else

whichtoshow=32'bz;

end

always@(posedge clk)

if(wena)

begin

if(addr!=0)

case(writenumber)

SB:state[addr][7:0]<=data\_in[7:0];

SH:state[addr][15:0]<=data\_in[15:0];

default:state[addr]<=data\_in;

endcase

end

assign data\_out=whichtoshow;

always@(order)

begin

writenumber[5:0]=order[31:26];

end

endmodule