//简单计算机核设计 2009-4-29 解释权姜咏江 Email:accsys@126.com

//参考书:姜咏江.PMC计算机设计与应用.清华大学出版社.2008-5

//说明：这里给出的简单计算机核设计，是初学计算机设计的最好实例。

//基本输入时钟clock

//复位控制：reset\_n,低电位有效

//基本输出：o

//程序存储器iram,16位，高5位是类指令代码,用imem16\_1.mif初始化

//数据存储器dram,16位，不用数据文件初始化

//用lpm存储器地址数据信号要稳定1拍，才可以读写数据

//指令格式:高5位指令代码,11位地址码,16位立即数(分高低8位)

module jdcpu

(

clock, //系统输入时钟

reset\_n, //复位信号，低电平有效

o, //数据输出端口

//调试输出以观察CPU内部变化，设计完成后删除：

opc, //程序计数器观察变量

omar, //数据地址寄存器观察变量

ojp, //CPU节拍观察变量

oqw, //程序存储器输出观察变量

oda, //累加器变化观察

ozf, //累加器为零标志观察

osp //堆栈指针变化观察

);

input clock;

input reset\_n;

output [15:0] o;

output [15:0] oqw,oda;

output [10:0] opc,omar,osp;

output [2:0] ojp;

output ozf;

//CPU内部器件和导线

//定义16位的程序存储器输出连接导线和数据存储器输入导线

wire [15:0] q\_w,q\_data;

//定义保持写数据存储器和写堆栈存储器控制信号的寄存器

reg dwren,swren;

//定义16位用于保持取出指令进行分析的指令寄存器

reg [15:0] ir;

//定义16位的运算器前端寄存器a、b,累加器da，输出寄存器oo和暂存输入数据寄存器ddata

reg [15:0] b,a,da,oo,ddata;

//定义11位的程序计数器pc，地址残存寄存器pc\_back，数据存储地址寄存器mar，堆栈指针sp和堆栈输出寄存器q\_s

reg [10:0] pc,pc\_back,mar,sp,q\_s;

//定义3位的CPU节拍寄存器

reg [2:0] jp;

//指令:

reg lda, //取数:从数据单元取数到da

add, //加:da与数据单元相加，结果放入da

out, //输出:将数据单元内容输出到输出寄存器

sdal, //低8位立即数:将8位立即数扩充为16位送da

sdah, //高8位立即数:将8位立即数作为高8位，与原da低8位连接成16位放在da中

str, //da送数据存储单元:

sub, //减:da与数据单元相减，结果放入da

jmp, //跳转

jz, //da为0跳转

jn, //da为负跳转

call, //调用子程序

ret, //返回

mult, //

divi, //

stp; //停止

//仿真信号输出:

assign o = oo;

assign opc = pc;

assign osp = sp;

assign omar = mar;

assign ojp = jp;

assign oqw = q\_w;

assign olda=lda;

assign oadd=add;

assign osub=sub;

assign oout=out;

assign ojmp=jmp;

assign ostr=str;

assign osdal=sdal;

assign osdah=sdah;

assign ocall=call;

assign oret=ret;

assign ojz=jz;

assign ojn=jn;

assign oda=da;

assign oir=ir;

assign ozf=~|da;

//指令存储器:

lpm\_rom iram(.address(pc),.inclock(clock),.q(q\_w)); //程序存储器

defparam iram.lpm\_width = 16;

defparam iram.lpm\_widthad = 11;

defparam iram.lpm\_outdata = "UNREGISTERED";

defparam iram.lpm\_indata = "REGISTERED";

defparam iram.lpm\_address\_control = "REGISTERED";

defparam iram.lpm\_file = "imem16\_2013.mif"; //初始化文件,放置程序

//数据存储器:

lpm\_ram\_dq dram(.data(ddata),.address(mar),.we(dwren),.inclock(clock),.q(q\_data)); //数据存储器

defparam dram.lpm\_width = 16;

defparam dram.lpm\_widthad = 10;

defparam dram.lpm\_outdata = "UNREGISTERED";

defparam dram.lpm\_indata = "REGISTERED";

defparam dram.lpm\_address\_control = "REGISTERED";

lpm\_ram\_dq sram(.data(pc\_back),.address(sp),.we(swren),.inclock(clock),.q(q\_s)); //堆栈

defparam sram.lpm\_width = 11;

defparam sram.lpm\_widthad = 10;

defparam sram.lpm\_outdata = "UNREGISTERED";

defparam sram.lpm\_indata = "REGISTERED";

defparam sram.lpm\_address\_control = "REGISTERED";

always @(posedge clock or negedge reset\_n)

begin

if (!reset\_n)

begin

pc <= 0;

sp <= 0;

lda <= 0;

add <= 0;

out <= 0;

sdal <= 0;

sdah <= 0;

str <= 0;

sub <= 0;

jmp <= 0;

jz <= 0;

jn <= 0;

call <= 0;

ret <= 0;

mult <= 0;

divi <= 0;

jp <= 0;

end

else

begin

// 节拍jp指出的状态：

case (jp)

0: begin //空拍，稳定地址寄存器数据需要

jp <= 1; //转到1拍

end

1: begin //依指令前5位编码来识别指令，并将指令标识置位

case (q\_w[15:11])

5'b00001: lda <= 1; //lda:00001

5'b00010: add <= 1; //add:00010

5'b00011: out <= 1; //out:00011

5'b00100: sdal <= 1; //低8位，扩充有符号16位

5'b00101: sdah <= 1; //高8位，与前面低8位输入合成16位

5'b00110: str <= 1; //da送数据单元

5'b00111: sub <= 1;

5'b01000: jmp <= 1;

5'b01001: if (da==0) jz <= 1; //累加器da是0，跳转

5'b01010: if (da[15]==1) jn <= 1; //累加器da为负，跳转

5'b01011: call <= 1;

5'b01100: ret <= 1;

5'b01101: mult <= 1;

5'b01110: divi <= 1;

5'b11111: stp <= 1;

default: jp <= 0;

endcase //节拍区分指令结束

jp <= 2; //转到jp=2的状态

end

2: begin //CPU进入jp=2的状态

case (q\_w[15:11]) //用指令编码确定指令

5'b00001: begin //lda <= 1;

mar<=q\_w[10:0]; //数据地址给到数据地址寄存器

jp <= 3; //转到jp=3的状态

end

5'b00010: begin //add <= 1;

mar<=q\_w[10:0];

jp <= 3;

end

5'b00011: begin //out <= 1;

mar<=q\_w[10:0];

jp <= 3;

end

5'b00100: begin //sdal <= 1;

da <= {{8{q\_w[7]}},q\_w[7:0]}; //将指令中写的8位立即数扩充成16位有符号数送到累加器da

sdal<= 0; //sdal指令执行完成

pc <= pc+1; //准备取下一条指令

jp<= 0; //节拍状态复位

end

5'b00101: begin //sdah <= 1;

da[15:0] <= {q\_w[7:0],da[7:0]}; //将指令中写的8位数放入累加器的高8位，累加器低8位数不变

sdah <= 0; //sdal指令执行完成

pc <= pc+1; //准备取下一条指令

jp<= 0; //节拍状态复位

end

5'b00110: begin //str <= 1;

mar<=q\_w[10:0];

ddata <= da; //累加器da送数据存储器

jp <= 3; //指令str未执行完，转jp=3

end

5'b00111: begin //sub <= 1;

mar<=q\_w[10:0];

jp <= 3;

end

5'b01000: begin //jmp <= 1;

pc <= q\_w[10:0];//将跳转程序地址送程序计数器

jmp <=0; //跳转指令完成

jp <= 0;

end

5'b01001: begin //jz <= 1;

if (jz) pc <= q\_w[10:0];//如果da=0则跳转

else pc <= pc+1; //不然执行下一条指令

jz <=0;

jp <= 0;

end

5'b01010: begin //jn <= 1;

if (jn) pc <= q\_w[10:0];

else pc <= pc+1;

jn<=0;

jp <= 0;

end

5'b01011: begin //call <= 1;

pc\_back <= pc+1;//保存下一条指令的地址

jp <= 3;

end

5'b01100: begin //ret <= 1;

jp <= 3;

end

5'b01101: begin //mult<= 1;

mar<=q\_w[10:0];

jp <= 3;

end

5'b01110: begin //divi <= 1;

mar<=q\_w[10:0];

jp <= 3;

end

5'b11111: jp<=0; //stp指令，返回jp=0状态

default: jp <= 0; //其他情况一律节拍返回jp=0状态

endcase

end

3: begin

case (q\_w[15:11])

5'b00001: begin //lda <= 1;

jp <= 4;

end

5'b00010: begin //add <= 1;

jp <= 4;

end

5'b00011: begin //out <= 1;

jp <= 4;

end

5'b00110: begin //str <= 1;

dwren <= 1;

jp <= 4;

end

5'b00111: begin //sub <= 1;

jp <= 4;

end

5'b01011: begin //call <= 1;

pc <= q\_w[10:0];//pc接收子程序地址

swren <= 1; //发出写堆栈信号

jp <= 4;

end

5'b01100: begin //ret <= 1;

sp <= sp-1;

jp <= 4;

end

5'b01101: begin //mult <= 1;

jp <= 4;

end

5'b01110: begin //divi <= 1;

jp <= 4;

end

default: jp <= 0;

endcase

end

4: begin

case (q\_w[15:11])

5'b00001: begin //lda <= 1;

da<=q\_data; //存储单元数据送累加器

pc <= pc+1;

jp <= 0;

lda<= 0; //lda指令执行完成

end

5'b00010: begin //add <= 1;

b<=q\_data; //存储单元数据送前端寄存器b

a<=da; //累计器da内容送前端寄存器a

jp <= 5;

end

5'b00011: begin //out <= 1;

oo <= q\_data; //将数据存储单元输出

pc <= pc+1;

jp <= 0;

out<= 0;

end

5'b00110: begin //str <= 1;

dwren <= 1; //发出写数据寄存器信号

jp <= 5;

end

5'b00111: begin //sub <= 1;

b<=q\_data;

a<=da;

jp <= 5;

end

5'b01011: begin //call <= 1;

sp <= sp+1; //写完堆栈之后，堆栈指针前移一位

swren <= 0; //停止写堆栈信号

jp <= 5;

end

5'b01100: begin //ret <= 1;

pc <= q\_s; //返回地址送到程序计数器

ret <= 0;

jp <= 0;

end

5'b01101: begin //mult <= 1;

b<=q\_data;

a<=da;

jp <= 5;

end

5'b01110: begin //divi <= 1;

b<=q\_data;

a<=da;

jp <= 5;

end

default: jp <= 0;

endcase

end

5: begin

case (q\_w[15:11])

5'b00010: begin //add <= 1;

da <= a+b; //相加结果送累加器da

pc <= pc+1;

add <=0;

jp <= 0;

end

5'b00110: begin //str <= 1;

dwren <= 0; //结束写存储器信号

pc <= pc+1;

str <=0;

jp <= 0;

end

5'b00111: begin //sub <= 1;

da <= a-b; //将减法运算结果送累加器da

pc <= pc+1;

sub<=0;

jp <= 0;

end

5'b01011: begin //call <= 1;

swren <= 0; //结束写堆栈信号

call<=0;

jp<=0;

end

5'b01101: begin //mult <= 1;

da <= a\*b; //将乘法运算结果送累加器

pc <= pc+1;

mult <=0;

jp <= 0;

end

5'b01110: begin //divi <= 1;

da <= a/b; //将除法运算结果送累加器

pc <= pc+1;

divi <=0;

jp <= 0;

end

default: jp <= 0;

endcase

end

endcase

end

end

endmodule

////// 仿真实例: 求64\*8且输出循环次数 ////////

//

// 汇编 编译

// sdal 1 2001

// str one 3001

// sub one 3801

// str result 3002

// str n 3005

// sdal 64 2040

// str x 3003

// sdal 8 2008

// str y 3004

//loop: lda y 0804

// jz exit 4812

// sub one 3801

// str y 3004

// lda result 0802

// add x 1003

// str result 3002

// call loopno 5814

// jmp loop 4009

//exit: out result 1802

// stp ffff

//loopno: lda n 0805

// add one 1001

// str n 3005

// out n 1805

// ret 6000

//

// 将编译的16进制数写入imem16\_1.mif

//

/////// 16进制结果输出:0200 //////////////////

//

////验证CPU的实例3: 求8！且输出结果。

//

// 汇编 编译

// sdal 1 2001

// str one 3001

// str result 3002

// sdal 8 2008

// str x 3003

//loop: lda x 0803

// jz exit 480D

// mult result 6802

// str result 3002

// lda x 0803

// sub one 3801

// str x 3003

// jmp loop 4005

//exit: out result 1802

// stp ffff

//将编译的16进制数写入imem16\_2013.mif

/////// 如果下载到开发板检查运行情况，各输出之间要加延时子程序。

//

//毕业设计：扩充设计一个功能完备的计算机，并在PMC110计算机开发板上下载运行。