`timescale 1ns/1ps

//here we will define the instruction register(IR)

`define opr\_type IR[31:27]

`define rdst IR[26:22]

`define rsrc1 IR[21:17]

`define mode IR[16]

`define rsrc2 IR[15:11]

`define isrc IR[15:0]

//here we will define arithmatic and logical operations

`define movesgpr 5'b00000

`define mov 5'b00001

`define add 5'b00010

`define sub 5'b00011

`define mul 5'b00100

`define kor 5'b00101

`define kand 5'b00110

`define kxor 5'b00111

`define kxnor 5'b01000

`define knand 5'b01001

`define knor 5'b01010

`define knot 5'b01011

module top();

reg [31:0] IR;

reg [15:0] GPR [31:0];

reg [15:0] SGPR;

reg [31:0] mul\_reg;

always@(\*)

begin

case(`opr\_type)

///////////////////////////////////////

`movesgpr : begin

GPR[`rdst]=SGPR;

end

///////////////////////////////////////

`mov:begin

if(`mode)

GPR[`rdst]=`isrc;

else

GPR[`rdst]=GPR[`rsrc1];

end

///////////////////////////////////////

`add:begin

if(`mode)

GPR[`rdst]=GPR[`rsrc1]+`isrc;

else

GPR[`rdst]=GPR[`rsrc1]+GPR[`rsrc2];

end

///////////////////////////////////////

`sub:begin

if(`mode)

GPR[`rdst]=GPR[`rsrc1]-`isrc;

else

GPR[`rdst]=GPR[`rsrc1]-GPR[`rsrc2];

end

///////////////////////////////////////

`mul:begin

if(`mode)

mul\_reg=GPR[`rsrc1]\*`isrc;

else

mul\_reg=GPR[`rsrc1]\*GPR[`rsrc2];

GPR[`rdst]=mul\_reg[15:0];

SGPR= mul\_reg[31:16];

end

///////////////////////////////////////

`kor:begin

if(`mode)

GPR[`rdst]=GPR[`rsrc1] | `isrc;

else

GPR[`rdst]=GPR[`rsrc1] | GPR[`rsrc2];

end

///////////////////////////////////////

`kand:begin

if(`mode)

GPR[`rdst]=GPR[`rsrc1] & `isrc;

else

GPR[`rdst]=GPR[`rsrc1] & GPR[`rsrc2];

end

///////////////////////////////////////

`kxor:begin

if(`mode)

GPR[`rdst]=GPR[`rsrc1] ^ `isrc;

else

GPR[`rdst]=GPR[`rsrc1] ^ GPR[`rsrc2];

end

///////////////////////////////////////

`kxnor:begin

if(`mode)

GPR[`rdst]=~(GPR[`rsrc1] ^ `isrc);

else

GPR[`rdst]=~(GPR[`rsrc1] ^ GPR[`rsrc2]);

end

///////////////////////////////////////

`knand:begin

if(`mode)

GPR[`rdst]=~(GPR[`rsrc1] & `isrc);

else

GPR[`rdst]=~(GPR[`rsrc1] & GPR[`rsrc2]);

end

///////////////////////////////////////

`knor:begin

if(`mode)

GPR[`rdst]=~(GPR[`rsrc1] | `isrc);

else

GPR[`rdst]=~(GPR[`rsrc1] | GPR[`rsrc2]);

end

///////////////////////////////////////

`knot:begin

if(`mode)

GPR[`rdst]= ~(`isrc);

else

GPR[`rdst]=~(GPR[`rsrc1]);

end

///////////////////////////////////////

endcase

end

///////////////////////logic for condition flag

reg sign = 0, zero = 0, overflow = 0, carry = 0;

reg [16:0] temp\_sum;

always@(\*)

begin

/////////////////sign bit

if(`opr\_type == `mul)

sign = SGPR[15];

else

sign = GPR[`rdst][15];

////////////////carry bit

if(`opr\_type == `add)

begin

if(`mode)

begin

temp\_sum = GPR[`rsrc1] + `isrc;

carry = temp\_sum[16];

end

else

begin

temp\_sum = GPR[`rsrc1] + GPR[`rsrc2];

carry = temp\_sum[16];

end end

else

begin

carry = 1'b0;

end

///////////////////// zero bit

if(`opr\_type == `mul)

zero = ~((|SGPR[15:0]) | (|GPR[`rdst]));

else

zero = ~(|GPR[`rdst]);

//////////////////////overflow bit

if(`opr\_type == `add)

begin

if(`mode)

overflow = ( (~GPR[`rsrc1][15] & ~IR[15] & GPR[`rdst][15] ) | (GPR[`rsrc1][15] & IR[15] & ~GPR[`rdst][15]) );

else

overflow = ( (~GPR[`rsrc1][15] & ~GPR[`rsrc2][15] & GPR[`rdst][15]) | (GPR[`rsrc1][15] & GPR[`rsrc2][15] & ~GPR[`rdst][15]));

end

else if(`opr\_type == `sub)

begin

if(`mode)

overflow = ( (~GPR[`rsrc1][15] & IR[15] & GPR[`rdst][15] ) | (GPR[`rsrc1][15] & ~IR[15] & ~GPR[`rdst][15]) );

else

overflow = ( (~GPR[`rsrc1][15] & GPR[`rsrc2][15] & GPR[`rdst][15]) | (GPR[`rsrc1][15] & ~GPR[`rsrc2][15] & ~GPR[`rdst][15]));

end

else

begin

overflow = 1'b0;

end

end

endmodule

module tb();

integer i=0;

top dut();

//updating value of all the register in GPR to 2

initial begin

for(i=0;i<32;i=i+1)

begin

dut.GPR[i]=2;

end

end

initial begin

//immediate add op

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=1;

dut.`opr\_type=2;

dut.`rsrc1=0;

dut.`rdst=1;

dut.`isrc=4;

#10;

$display("OPERATION:ADD\_imm Rsrc1=%0d Rsrc2=%0d Rdst=%0d",dut.GPR[0],dut.`isrc,dut.GPR[1]);

$display("-------------------------------------------");

//immediate add operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=0;

dut.`opr\_type=2;

dut.`rsrc1=0;

dut.`rdst=9;

dut.`rsrc2=1;

#10;

$display("OPERATION:ADD Rsrc1=%0d Rsrc2=%0d Rdst=%0d",dut.GPR[0],dut.GPR[1],dut.GPR[9]);

$display("-------------------------------------------");

//immediate mov operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=1;

dut.`opr\_type=1;

dut.`rdst=1;

dut.`isrc=90;

#10;

$display("OPERATION:MOV\_imm Rdst=%0d imm\_data=%0d",dut.GPR[1],dut.`isrc);

$display("-------------------------------------------");

//mov operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=0;

dut.`opr\_type=1;

dut.`rdst=7;

dut.`rsrc1=1;

#10;

$display("OPERATION:MOV Rdst=%0d rsrc1=%0d",dut.GPR[7],dut.GPR[1]);

$display("-------------------------------------------");

//immediate mul operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=1;

dut.`opr\_type=4;

dut.`rsrc1=1;

dut.`rdst=3;

dut.`isrc=88;

#10;

$display("OPERATION:MUL\_imm Rsrc1=%0d Rsrc2=%0d Rdst=%0d",dut.GPR[1],dut.`isrc,dut.GPR[3]);

$display("-------------------------------------------");

//mul operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=0;

dut.`opr\_type=4;

dut.`rsrc1=1;

dut.`rsrc2=3;

dut.`rdst=4;

#10;

$display("OPERATION:MUL\_imm Rsrc1=%0d Rsrc2=%0d Rdst=%0d",dut.GPR[1],dut.GPR[3],dut.mul\_reg);

$display("-------------------------------------------");

//immediate and operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=1;

dut.`opr\_type=6;

dut.`rsrc1=1;

dut.`isrc=3;

dut.`rdst=4;

#10;

$display("OPERATION:immediate AND Rsrc1=%8b Rsrc2=%8b Rdst=%8b",dut.GPR[1],dut.`isrc,dut.GPR[4]);

$display("-------------------------------------------");

//and operation

$display("-------------------------------------------");

dut.IR=0;

dut.`mode=0;

dut.`opr\_type=6;

dut.`rsrc1=1;

dut.`isrc=3;

dut.`rdst=4;

#10;

$display("OPERATION:immediate AND Rsrc1=%8b Rsrc2=%8b Rdst=%8b",dut.GPR[1],dut.`isrc,dut.GPR[4]);

$display("-------------------------------------------");

/////////////////////////// zero flag

dut.IR = 0;

dut.GPR[0] = 0;

dut.GPR[1] = 0;

dut.`mode = 0;

dut.`rsrc1 = 0;//gpr[0]

dut.`rsrc2 = 1;//gpr[1]

dut.`opr\_type = 2;

dut.`rdst = 2;

#10;

$display("OP:Zero Rsrc1:%0d Rsrc2:%0d Rdst:%0d",dut.GPR[0], dut.GPR[1], dut.GPR[2] );

$display("------------------------------------------");

//////////////////////////sign flag

dut.IR = 0;

dut.GPR[0] = 16'h8000; /////1000\_0000\_0000\_0000

dut.GPR[1] = 0;

dut.`mode = 0;

dut.`rsrc1 = 0;//gpr[0]

dut.`rsrc2 = 1;//gpr[1]

dut.`opr\_type = 2;

dut.`rdst = 2;

#10;

$display("OP:Sign Rsrc1:%0d Rsrc2:%0d Rdst:%0d",dut.GPR[0], dut.GPR[1], dut.GPR[2] );

$display("------------------------------------------");

////////////////////////carry flag

dut.IR = 0;

dut.GPR[0] = 16'h8000; /////1000\_0000\_0000\_0000 <0

dut.GPR[1] = 16'h8002; /////1000\_0000\_0000\_0010 <0

dut.`mode = 0;

dut.`rsrc1 = 0;//gpr[0]

dut.`rsrc2 = 1;//gpr[1]

dut.`opr\_type = 2;

dut.`rdst = 2; //////// 0000\_0000\_0000\_0010 >0

#10;

$display("OP:Carry & Overflow Rsrc1:%0d Rsrc2:%0d Rdst:%0d",dut.GPR[0], dut.GPR[1], dut.GPR[2] );

$display("-----------------------------------------");

#20;

$finish;

end

endmodule