//------------------------------código-top-riscv--em-sv-------------------------------------

// -----------------------------------------------------------------------------

// FILE NAME : riscv\_top

// AUTHOR : voo

// AUTHOR'S EMAIL : {voo}@cin.ufpe.br

// -----------------------------------------------------------------------------

// RELEASE HISTORY

// VERSION DATE AUTHOR DESCRIPTION

// 2.0 2017-01-30 voo version sv

// -----------------------------------------------------------------------------

`timescale 1ns/1ps

module riscv\_top(

input logic CLK,

input logic RST,

//-----------PARA PERIFERICOS--------------//

output logic [15:0] pio\_0\_out,

input logic [15:0] pio\_0\_in,

//------------------------DEBUG-------------------//

//output logic [1:0] cmp,

//output logic [31:0] sp,

//output logic [31:0] ra,

//output logic [31:0] ir,

// output logic [31:0] result,

//output logic [31:0] instruction\_o,

//----------------Periférico UART----------------//

input logic uart\_rx,

output logic uart\_tx,

output logic uart\_full,

output logic uart\_empty

);

//-------------------------SINAIS-UTILIZADOS---------------------//

logic ACK\_I\_M0; //wshbn\_master wbus

//--------------------------------------------------------//

logic [7:0] ADR; // ADR = ADR\_O\_M0 wshbn\_master wbus

logic [7:0] ADR\_O\_M0; //wshbn\_master

//--------------------------------------------------------/;

logic ACK\_O\_S0;

logic ACK\_O\_S1;

//--------------------------------------------------------//

logic ACMP0;

logic ACMP1;

logic ACMP2;

//--------------------------------------------------------//

logic CYC ;

logic CYC\_O\_M0 ; //wshbn\_master wbus

logic [31:0] DAT\_O\_M0 ; //wshbn\_master wbus

logic [31:0] DAT\_O\_S0 ; //wshbn\_pio SLV0

logic [31:0] DAT\_O\_S1 ; //wshbn\_timer

logic [31:0] DAT\_O\_S2 ; //UART

//--------------------------------------------------------//

logic [31:0] DRD ;

logic [31:0] DWR ;

//--------------------------------------------------------//

logic STB ; //STB = STB\_O\_M0 wshbn\_master wbus

logic STB\_O\_M0; //wshbn\_master wbus

logic STB\_I\_S0; //wshbn\_pio SLV0

logic STB\_I\_S1; //wshbn\_timer SLV1

//--------------------------------------------------------//

logic STB\_I\_S2;

logic WE ;

logic WE\_O\_M0;

logic [1:0] st;

//--------------------------------------------------------//

logic [31:0] inst\_cache\_add; //inst\_mem\_ctrl inst\_mem

logic [31:0] inst\_cache\_data; //i\_cache M0 -> inst\_mem\_ctrl inst\_mem

logic inst\_cache\_rden ; //inst\_mem\_ctrl inst\_mem -> i\_cache M0

//--------------------------------------------------------//

logic [31:0] data\_cache\_data\_o;

logic [31:0] data\_cache\_data\_i;

logic [29:0] data\_cache\_addr\_i;

logic data\_cache\_wren;

logic data\_cache\_rden;

logic data\_cache\_busy;

//--------------------------------------------------------//

logic [31:0] ext\_ram\_data\_o;

logic [31:0] ext\_ram\_data\_i;

logic [29:0] ext\_ram\_addr\_i;

logic ext\_ram\_wren;

logic ext\_ram\_rden;

logic ext\_ram\_busy;

//--------------------------------------------------------//

logic [31:0] cpu\_data\_add;

logic [31:0] cpu\_data\_i;

logic [31:0] cpu\_data\_o;

logic cpu\_data\_wren;

logic cpu\_data\_rden;

//--------------------------------------------------------//

logic [31:0] wshbn\_data\_o;

logic [31:0] wshbn\_data\_i;

logic wshbn\_data\_wren;

logic wshbn\_data\_rden;

//--------------------------------------------------------//

logic wshbn\_stall\_w ;

logic wshbn\_busy ;

logic mem\_busy ;

logic data\_av ;

//--------------------------------------------------------//

logic tmr\_interrupt;

logic interrupt\_req;

//--------------------------------------------------------//

logic [29:0] wshbn\_addr;

reg rst\_eval\_regs = 1'b0;

reg en\_eval\_regs = 1'b1;

logic [63:0] clk\_counter;

logic [63:0] inst\_counter ;

logic [63:0] branch\_counter ;

//--------------------------------------------------------//

logic [31:0] inst\_add;

logic [31:0] inst;

logic [31:0] ext\_inst\_add;

logic [31:0] ext\_inst\_data;

logic inst\_rden

logic ext\_inst\_rden;

logic mem\_clken;

//----------------------INSTANCIAS----------------------------------//

inst\_mem\_ctrl inst\_mem (

.clk(CLK),

.rst(RST),

.rd(inst\_rden),

.addr\_i(inst\_add),

.instr\_o(inst),

.cmp(cmp),

.mem0\_rd(inst\_cache\_rden),

.mem0\_addr\_o(inst\_cache\_add),

.mem0\_data\_i(inst\_cache\_data)

);

//

i\_cache M0 (

.address(inst\_cache\_add[12:2]),

.clock(CLK), .rden(inst\_cache\_rden),

.q(inst\_cache\_data)

);

//--------------------------------------------------------//

d\_cache M1(

.address(data\_cache\_addr\_i[11:0]),

.clock(CLK), .data(data\_cache\_data\_i),

.wren(data\_cache\_wren), .q(data\_cache\_data\_o));

//--------------------------------------------------------//

riscv\_cpu\_no cpu (

.clk\_i(CLK),

.rst\_i(RST), .inst\_cache\_rden(inst\_rden),

.inst\_cache\_add(inst\_add),

.inst\_cache\_data(inst), .ra(ra),

.sp(sp),

.ir\_o(ir),

.mem\_clken(mem\_clken),

.interrupt\_req(interrupt\_req),

.mem\_busy(mem\_busy),

.data\_av(1'b0),

.data\_add(cpu\_data\_add),

.data\_o(cpu\_data\_o),

.data\_i(cpu\_data\_i),

.data\_rden(cpu\_data\_rden),

.data\_wren(cpu\_data\_wren),

.rst\_eval\_regs(rst\_eval\_regs),

.en\_eval\_regs(en\_eval\_regs),

.clk\_counter\_o(clk\_counter),

.inst\_counter\_o(inst\_counter),

.branch\_counter\_o(branch\_counter),

.result(result),

.instruction\_o(instruction\_o)

);

//--------------------------------------------------------//

mem\_ctrl mem\_controller (

.clk(CLK),

.rst(RST),

.rd(cpu\_data\_rden),

.wr(cpu\_data\_wren),

.addr\_i(cpu\_data\_add),

.data\_i(cpu\_data\_o),

.data\_o(cpu\_data\_i),

.hold\_cpu(mem\_busy),

.wshbn\_rd(wshbn\_data\_rden),

.wshbn\_wr(wshbn\_data\_wren),

.wshbn\_addr\_o(wshbn\_addr),

.wshbn\_data\_o(wshbn\_data\_i),

.wshbn\_data\_i(wshbn\_data\_o),

.wshbn\_busy(wshbn\_busy),

.wshbn\_data\_av(data\_av),

.mem0\_rd (data\_cache\_rden),

.mem0\_wr(data\_cache\_wren),

.mem0\_addr\_o(data\_cache\_addr\_i),

.mem0\_data\_o(data\_cache\_data\_i),

.mem0\_data\_i(data\_cache\_data\_o),

.mem0\_busy(data\_cache\_busy)

);

//--------------------------------------------------------//

interrupt\_controller int\_ctrl (

.CLK\_I(CLK),

.RST\_I(RST),

.interrupt0(tmr\_interrupt),

.interrupt1(1'b0),

.interrupt2(1'b0),

.interrupt3 (1'b0),

.inti(interrupt\_req)

);

//--------------------------------------------------------//

wshbn\_master wbus (

.CLK\_I(CLK),

.RST\_I(RST),

.ADR\_O(ADR\_O\_M0),

.DAT\_I(DRD),

.DAT\_O(DAT\_O\_M0),

???.WE\_O(WE\_O\_M0),

.STB\_O(STB\_O\_M0),

.ACK\_I(ACK\_I\_M0),

.CYC\_O(CYC\_O\_M0),

.busy(wshbn\_busy),

.data\_av(data\_av),

.stall\_cpu(wshbn\_stall\_w),

.add\_i(wshbn\_addr[7:0]),

.data\_i(wshbn\_data\_i),

.data\_o(wshbn\_data\_o),

.wr(wshbn\_data\_wren),

.rd(wshbn\_data\_rden),

.st(st)

);

//----------------------SLAVE-BUS-UART--------------------------------//

wshbn\_pio SLV0(

.CLK\_I(CLK),

.RST\_I(RST),

.ADR\_I(ADR[3:0]),

.DAT\_I(DWR),

.DAT\_O(DAT\_O\_S0),

.WE\_I(WE),

.STB\_I(STB\_I\_S0),

.ACK\_O (ACK\_O\_S0),

.CYC\_I(CYC),

.pio\_0\_in(pio\_0\_in),

.pio\_0\_out(pio\_0\_out)

);

//------------------------DECODER-BUS-ADDRESS------------------------------//

always @(ADR) begin

ACMP2 = ( ~ADR[7] & ~ADR[6] & ADR[5] & ~ADR[4] ); //0010

ACMP1 = ( ~ADR[7] & ~ADR[6] & ~ADR[5] & ADR[4] ); //0001

ACMP0 = ( ~ADR[7] & ~ADR[6] & ~ADR[5] & ~ADR[4] ); //0000

end

//--------------------------DECODER-BUS-CONTROL----------------------------//

always @(ACMP2,ACMP1, ACMP0, CYC, STB)

begin

STB\_I\_S2 = CYC & STB & ACMP2;

STB\_I\_S1 = CYC & STB & ACMP1;

STB\_I\_S0 = CYC & STB & ACMP0;

end

//-----------------------CONTROL-BUS-MASTER-------------------------------//

assign ACK\_I\_M0 = ACK\_O\_S0; //ACK\_I\_M0 wshbn\_master wbus

assign CYC = CYC\_O\_M0; //wshbn\_master wbus

assign ADR = ADR\_O\_M0; //wshbn\_master wbus

assign WE = WE\_O\_M0; //wshbn\_master wbus

assign DWR = DAT\_O\_M0; //wshbn\_master wbus

assign STB = STB\_O\_M0; //wshbn\_master wbus

//-----------------------DECODER-BUS-DATA-------------------------------//

always @(DAT\_O\_S1, DAT\_O\_S0, ADR )

begin

case ( ADR[7:4] )

4'b0000: DRD <= DAT\_O\_S0; // IO wshbn\_pio SLV0

4'b0001: DRD <= DAT\_O\_S1; // wshbn\_timer

4'b0010: DRD <= DAT\_O\_S2; // UART

default : DRD <= DAT\_O\_S0; // wshbn\_pio SLV0

endcase

end

endmodule