// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// Module : Memory

// Description : This is a model of a memory. The memory has two parts

// of 256 bytes each. The two memory parts are mapped at

// addresses 32'h0000\_0000 and 32'h8000\_0000

// Programmer : Ngan Ngoc Pham

// Date : Nov 11, 1998

// Reference : old description of memory by Srisai S Rao

// ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

//---

// Note: Not intended to be synthesized

// (P. Franzon)

//---

module Memory (MemData,

Address,

WriteData,

MemRead,

MemWrite);

// Port declaration

input [31:0] Address, WriteData;

input MemRead, MemWrite;

output [31:0] MemData;

// Registers

reg [31:0] MemData;

reg [7:0] Memory1[0:255];

reg [7:0] Memory2[0:255];

reg[7:0] WordBoundary;

// Integers

integer i;

integer mcd1, mcd2;

// Initialization

initial

begin

$readmemh( "testCases.data", Memory1 ); // CHANGE

$readmemh( "exception.data", Memory2 );

end

// Operation

always @(Address or MemRead or MemWrite or WriteData)

begin

MemData[31:0] = 32'hzzzz\_zzzz;

#1

WordBoundary[7:0] = Address[7:0];

WordBoundary[1:0] = 2'b00;

casex ( Address )

32'h0000\_00xx: begin

if (MemRead == 1'b1)

begin

MemData[7:0] = Memory1[ WordBoundary[7:0] + 3 ];

MemData[15:8] = Memory1[ WordBoundary[7:0] + 2];

MemData[23:16] = Memory1[ WordBoundary[7:0] + 1];

MemData[31:24] = Memory1[ WordBoundary[7:0] + 0];

end

else if (MemWrite == 1'b1)

begin

Memory1[ WordBoundary[7:0] + 3 ] = WriteData[7:0];

Memory1[ WordBoundary[7:0] + 2 ] = WriteData[15:8];

Memory1[ WordBoundary[7:0] + 1 ] = WriteData[23:16];

Memory1[ WordBoundary[7:0] + 0 ] = WriteData[31:24];

end

end

32'h8000\_00xx: begin

if (MemRead == 1'b1)

begin

MemData[7:0] = Memory2[ WordBoundary[7:0] + 3 ];

MemData[15:8] = Memory2[ WordBoundary[7:0] + 2];

MemData[23:16] = Memory2[ WordBoundary[7:0] + 1];

MemData[31:24] = Memory2[ WordBoundary[7:0] + 0];

end

else if (MemWrite == 1'b1)

begin

Memory2[ WordBoundary[7:0] + 3 ] = WriteData[7:0];

Memory2[ WordBoundary[7:0] + 2 ] = WriteData[15:8];

Memory2[ WordBoundary[7:0] + 1 ] = WriteData[23:16];

Memory2[ WordBoundary[7:0] + 0 ] = WriteData[31:24];

end

end

default: if (MemRead == 1'b1 | MemWrite == 1'b1)

begin

mcd1 = $fopen( "memoryOut.data" );

for( i=0; i<256; i=i+1 )

begin

$fwrite( mcd1,"\n@%h %h",i,Memory1[i] );

end

$fclose( mcd1 );

mcd2 = $fopen( "exceptionOut.data" );

for( i=0; i<256; i=i+1 )

begin

$fwrite( mcd2,"\n@%h %h",i,Memory2[i] );

end

$fclose( mcd2 );

if (MemRead)

$display("\n Can not read. Memory out of range");

if (MemWrite)

$display("\n Can not WRITE. Memory out of range");

$display("\n Detected End Of Memory Space");

$display("\n Saving Memory State");

$display("\n Exiting The Simulation");

$finish;

end

else begin

MemData = 32'hz;

end

endcase

end

endmodule