`timescale 1ns / 1ps

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* C E C S 4 4 0 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* File Name: TopLevel\_tb.v

\* Project: lab 4

\* Designer: Benjamin Santos and Naoaki Takatsu

\* Email: benjamin.santos@student.csulb.edu

\* naoaki.takatsu@student.csulb.edu

\* Rev. No.: Version 1.0

\* Rev. Date: 10/1/2018

\*

\* Purpose: This is the integer datapath module testbench.

\*

\* Notes: This module will test for the following cases:

\* Logical OR, Subtraction, Logical Shift Right,

\* Logical Shift Left, Division, Multiplication using DT,

\* Memory Read, 1's Complement, 2's Complement,

\* Addition, and Memory Write.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

module TopLevel\_tb;

// Inputs

reg clk, reset, D\_En, T\_Sel, HILO\_LD, dm\_cs, dm\_wr, dm\_rd;

reg [2:0] Y\_Sel;

reg [4:0] D\_Addr, S\_Addr, T\_Addr, FS;

reg [31:0] DT, PC\_in;

// Outputs

wire N, Z, C, V;

integer i;

// Instantiate the Unit Under Test (UUT)

TopLevel uut ( .clk(clk), .reset(reset), .D\_En(D\_En), .D\_Addr(D\_Addr),

.S\_Addr(S\_Addr), .T\_Addr(T\_Addr), .T\_Sel(T\_Sel), .FS(FS),

.HILO\_LD(HILO\_LD), .Y\_Sel(Y\_Sel), .PC\_in(PC\_in), .DT(DT),

.dm\_cs(dm\_cs), .dm\_wr(dm\_wr), .dm\_rd(dm\_rd),

.N(N), .Z(Z), .C(C), .V(V) );

task Reg\_Dump;

begin

//Read initial regfile data through ALU\_OUT

for( i = 0; i < 16; i = i + 1 )

begin

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b0, 5'h00, 5'h00, i };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h01, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd} =

{ 1'b0, 1'b0, 1'b0};

@ (negedge clk)

$display( "t=%t, alu\_out%0d = 0x%h, reg%0d = 0x%h, mem%0d = 0x%h ",

$time, i, uut.IDP.regfile.T, i, uut.IDP.regfile.regs[i], i, uut.data\_mem.M[i]);

end

end

end

endtask

// Create a 10 ns clock

always

#5 clk = ~clk;

initial begin

// Initialize Inputs

clk = 0;

reset = 1'b1;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b0, 5'h00, 5'h00, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd} =

{ 1'b0, 1'b0, 1'b0};

// store .dat file data to regfile

$readmemh( "./IntReg\_Lab4.dat", uut.IDP.regfile.regs );

$readmemh( "./dMem\_Lab4.dat", uut.data\_mem.M );

$timeformat( -9, 1, " ps", 9 ); //Display time in nanoseconds

// Wait 100 ns for global reset to finish

#100;

reset = 0;

// Read initialized regfile data through ALU\_OUT

$display("\*\*\*\*\*\*Contents of initialized registers\*\*\*\*\*\*");

Reg\_Dump();

// OR $r1 <- $r3 | $r4

// 1. $RS <- $S(r3), $RT <- $T(r4)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b0, 5'h00, 5'h03, 5'h04 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// Signed SUB $r2 <- $r1 - $r14

// 1. $RS <- $S(r1), $RT <- $T(r14)

// OR $r1 <- $r3 | $r4

// 2. $ALU\_OUT <- $RS(r3) | $RT(r4)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b0, 5'h00, 5'h01, 5'h0E };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h09, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// SRL $r3 <- SRL $r4

// 1. $RT <- $T(r4)

// Signed SUB $r2 <- $r1 - $r14

// 2. $ALU\_OUT <- $RS(r1) - $RT(r14)

// OR $r1 <- $r3 | $r4

// 3. $D(r1) <- $ALU\_OUT(r3 | r4)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h01, 5'h00, 5'h04 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h03, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// SLL $r4 <- SLL $r5

// 1. $RT <- $T(r5)

// SRL $r3 <- SRL $r4

// 2. $ALU\_OUT <- $RT(r14) >> 1

// Signed SUB $r2 <- $r1 - $r14

// 3. $D(r2) <- $ALU\_OUT(r1 - r14)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h02, 5'h00, 5'h05 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h0D, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// DIV $r5 <- $r15 / $r14, $r6 <- $r15 % $r14

// 1. $RS <- $S(r15), $RT <- $T(r14)

// SLL $r4 <- SLL $r5

// 2. $ALU\_OUT <- $RT(r14) << 1

// SRL $r3 <- SRL $r4

// 3. $D(r3) <- $ALU\_OUT(r14 >> 1)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h03, 5'h0F, 5'h0E };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h0C, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// MPY

// $r7 <- $r11 \* 0xFFFF\_FFFB (lower 32 bit)

// $r8 <- $r11 \* 0xFFFF\_FFFB (upper 32 bit)

// 1. $RS <- $S(r11)

// DIV $r5 <- $r15 / $r14, $r6 <- $r15 % $r14

// 2. $HI <- $RS(r15) % $RT(r14)

// $LO <- $RS(r15) / $RT(r14)

// SLL $r4 <- SLL $r5

// 3. $D(r4) <- $ALU\_OUT(r5 << 1)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h04, 5'h0B, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h1F, 1'b1 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// MPY

// $r7 <- $r11 \* 0xFFFF\_FFFB (lower 32 bit)

// $r8 <- $r11 \* 0xFFFF\_FFFB (upper 32 bit)

// 1. $RS <- $S(r11), $RT <- $DT(0xFFFF\_FFFB)

// Wait for DIV to finish storing

// 2. do nothing

// DIV $r5 <- $r15 / $r14, $r6 <- $r15 % $r14

// 3. $D(r6) <- $HI(r15 % r14)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h06, 5'h0B, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b1, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'hFFFF\_FFFB, 32'h0000\_0000, 3'h1 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// Memory read $r12 <- M[r15]

// 1. $RS <- $S(r12)

// MPY

// $r7 <- $r11 \* 0xFFFF\_FFFB (lower 32 bit)

// $r8 <- $r11 \* 0xFFFF\_FFFB (upper 32 bit)

// 2. $HI <- $RS(r11) \* $RT(0xFFFF\_FFFB) (upper 32 bits)

// $LO <- $RS(r11) \* $RT(0xFFFF\_FFFB) (lower 32 bits)

// DIV $r5 <- $r15 / $r14, $r6 <- $r15 % $r14

// 3. $D(r5) <- $LO(r15 / r14)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h05, 5'h0C, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h1E, 1'b1 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h2 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// Memory read $r12 <- M[r15]

// 1. $RS <- $S(r12)

// Wait for MPY to finish storing

// 2. do nothing

// MPY

// $r7 <- $r11 \* 0xFFFF\_FFFB (lower 32 bit)

// $r8 <- $r11 \* 0xFFFF\_FFFB (upper 32 bit)

// 3. $D(r8) <- $HI($r11 \* 0xFFFF\_FFFB)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h08, 5'h0C, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h1 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// NOR (1's complement) $r11 <- $r0 NOR $r11

// 1. $RS <- $S(r0), $RT <- $T(r11)

// Memory read $r12 <- M[r15]

// 2. $ALU\_OUT <- $RS(r15)

// MPY

// $r7 <- $r11 \* 0xFFFF\_FFFB (lower 32 bit)

// $r8 <- $r11 \* 0xFFFF\_FFFB (upper 32 bit)

// 3. $D(r7) <- $LO($r11 \* 0xFFFF\_FFFB)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h07, 5'h00, 5'h0B };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h2 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// NOR (1's complement) $r11 <- $r0 NOR $r11

// 1. $RS <- $S(r0), $RT <- $T(r11)

// Wait for Memory read

// 2. do nothing

// Memory read $r12 <- M[r15]

// 3. $DY <- M[$ALU\_OUT(r15)]

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b0, 5'h00, 5'h00, 5'h0B };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b1, 1'b0, 1'b1 };

end

// SUB (2's complement) $r10 <- $r0 - $r10

// 1. $RS <- $S(r0), $RT <- $T(r10)

// NOR (1's complement) $r11 <- $r0 NOR $r11

// 2. $ALU\_OUT <- ~( $RS(r0) | $RT(r11) )

// Memory read $r12 <- M[r15]

// 3. $D <- $DY(M[r15])

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h0C, 5'h00, 5'h0A };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h0B, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h3 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// ADD $r9 <- $r10 + $r11

// 1. $RS <- $S(r10), $RT <- $T(r11)

// SUB (2's complement) $r10 <- $r0 - $r10

// 2. $ALU\_OUT <- $RS(r0) - $RT(r10)

// NOR (1's complement) $r11 <- $r0 NOR $r11

// 3. $D <- $ALU\_OUT(~( r0 | r11 ))

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h0B, 5'h0A, 5'h0B };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h03, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// SW immediate (PC\_Load) $r13 <- 0x1001\_00C0

// 1. do nothing

// ADD $r9 <- $r10 + $r11

// 2. $ALU\_OUT <- $RS(r0) - $RT(r10)

// SUB (2's complement) $r10 <- $r0 - $r10

// 3. $D <- $ALU\_OUT($r0 - $r10)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h0A, 5'h00, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h02, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// write memory M[14] <- R12

// 1. $RS <- $S(r12)

// SW immediate (PC\_Load) $r13 <- 0x1001\_00C0

// 2. do nothing

// ADD $r9 <- $r10 + $r11

// 3. $D <- $ALU\_OUT($r10 + $r11)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h09, 5'h0C, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h1001\_00C0, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// write memory M[14] <- R12

// 2. $ALU\_OUT <- $RS(r12)

// SW immediate (PC\_Load) $r13 <- 0x1001\_00C0

// 3. $D <- $PC\_in(0x1001\_00C0)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b1, 5'h13, 5'h00, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h1001\_00C0, 3'h4 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b0, 1'b0, 1'b0 };

end

// write memory M[14] <- R12

// 3. $M[14] <- $ALU\_OUT(r12)

@ (negedge clk)

begin

reset = 1'b0;

// Regfile related control signals

{ D\_En, D\_Addr, S\_Addr, T\_Addr } =

{ 1'b0, 5'h00, 5'h00, 5'h00 };

// ALU related control signals

{ T\_Sel, FS, HILO\_LD } =

{ 1'b0, 5'h00, 1'b0 };

// Y-MUX related control signals

{ DT, PC\_in, Y\_Sel } =

{ 32'h0000\_0000, 32'h0000\_0000, 3'h0 };

// Memory related control signals

{ dm\_cs, dm\_wr, dm\_rd } =

{ 1'b1, 1'b1, 1'b0 };

end

// Read updated regfile data through ALU\_OUT

$display("\*\*\*\*\*\*Contents of updated registers\*\*\*\*\*\*");

Reg\_Dump();

end

endmodule