1. Consider the system shown in Figure 4.60 of the textbook.

***lw    $2, 100($1)***

***add     $3, $1, $2***

***lw    $4, 100($2)***

***and   $1, $4, $2***

***slt   $2, $1, $5***

 How many clock cycles are required to execute the instructions on the given datapath?  Justify your answer. (20 points)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| lw $2, 100($1) | IF | ID | EX | M | WB |  |  |  |  |  |  |
| add $3, $1, $2 |  | IF | IF | ID | EX | M | WB |  |  |  |  |
| lw $4, 100($2) |  |  |  | IF | ID | EX | M | WB |  |  |  |
| and $1, $4, $2 |  |  |  |  | IF | IF | ID | EX | M | WB |  |
| slt $2, $1, $5 |  |  |  |  |  |  | IF | ID | EX | M | WB |

$2 and $4 cause Hazards

Number of Cycles = S + N – 1 + Hazards

= 5 + 5 – 1 + 2 = **11 Cycles**

2. Consider the system given in Figure 4.60 of the textbook. Find the number of clock cycles required to execute the instruction  with register forwarding? Justify your answer. (20 points)

***add     $3, $1, $2***

***or       $4, $2, $3***

***and     $1, $4, $2***

***slt     $2, $1, $5***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| add $3, $1, $2 | IF | ID | EX | M | WB |  |  |  |
| or $4, $2, $3 |  | IF | ID | EX | M | WB |  |  |
| and $1, $4, $2 |  |  | IF | ID | EX | M | WB |  |
| slt $2, $1, $5 |  |  |  | IF | ID | EX | M | WB |

Number of Cycles = S + N – 1 + Hazards

= 5 + 4 – 1 + 0 = **8 Cycles**

3. Find the number of clock cycles required to execute the instruction sequence given below if the system given in Figure 4.62 of the textbook is used.  Assume branch BEQ is taken. (20 points)

***add     $3, $1, $2***

***or    $4, $2, $3***

***beq     $1, $2, 2  (Note: 2  is the offset of the BEQ )***

***and     $1, $4, $2***

***slt     $2, $1, $5***

***lw     $12, 0($18)***

***sw    $12, 4($18)***

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| add $3, $1, $2 | IF | ID | EX | M | WB |  |  |  |  |  |
| or $4, $2, $3 |  | IF | ID | EX | M | WB |  |  |  |  |
| beq $1, $2, 2 |  |  | IF | ID | EX | M | WB |  |  |  |
| and $1, $4, $2 |  |  |  | IF | FLUSHED | | | |  |  |
| slt $2, $1, $5 |  |  |  |  |  |  |  |  |  |  |
| lw $12, 0($18) |  |  |  |  | IF | ID | EX | M | WB |  |
| sw $12, 4($18) |  |  |  |  |  | IF | ID | EX | M | WB |

$12 causes a Hazard

Number of Cycles = S + N – 1 + Hazards

5 + 5 – 1 + 1 = **10 Cycles**

4) Model and verify a parameterizable register in Verilog with 32 as its default size.  This register is updated based on its input data *data\_in* on posedge of clock *clk*when *w\_enable* input is asserted.  The content of the register is always posted to its output *data\_out*. It is cleared asynchronously when active low reset signal *rst* is asserted.  You need to provide source code for the register and its testbench. (40 point)

**Parametizable.v**

//Dip Amin

//CPE 142, Sec.1

//Assignment 5

module parameterizable(data\_in, clk, w\_enable, rst, data\_out);

parameter n = 32;

output [n-1:0] data\_out;

input [n-1:0] data\_in;

input clk;

input w\_enable;

input rst;

reg [n-1:0] data\_out;

always @(posedge clk or negedge rst)

begin

if (rst)

data\_out <= 32'h00000000;

else if (w\_enable)

data\_out <= data\_in;

end

endmodule

**Parametizablestimulus.v**

//Dip Amin

//CPE 142, Sec. 1

//Assignment 5

`include "parameterizable.v"

module parameterizable\_stimulus;

reg [31:0] data\_in;

reg clk;

reg w\_enable;

reg rst;

wire [31:0] data\_out;

initial

$vcdpluson;

parameterizable uut (.data\_in(data\_in), .clk(clk), .w\_enable(w\_enable), .rst(rst), .data\_out(data\_out));

always

begin

clk = 1'b1;

#50;

clk = 1'b0;

#50;

end

initial

begin

rst = 1'b1;

data\_in = 32'hFFFFFFFF;

w\_enable = 1'b1;

#100;

rst = 1'b1;

data\_in = 32'h00083210;

w\_enable = 1'b0;

#100;

rst = 1'b0;

data\_in = 32'h12345678;

w\_enable = 1'b1;

#100;

rst = 1'b0;

data\_in = 32'h00025302;

w\_enable = 1'b0;

#100;

rst = 1'b0;

data\_in = 32'h000234FF;

w\_enable = 1'b1;

#100;

rst = 1'b1;

data\_in = 32'h00050000;

w\_enable = 1'b0;

#100; $finish;

end

initial

begin

$monitor("Data In = %h Data Out = %h Reset = %b Write Enabled = %b", data\_in[31:0], data\_out[31:0], rst, w\_enable);

end

endmodule