Lab title

your names

October 23, 2017

1 Introduction

Introduction with problem overview, your design procedure, and rationale. Be as brief as possible to let me know the big picture of the lab.

2 Interface

This section should identify the inputs and outputs of each stage. To do this, rather than explaining them in paragraph form, please take the datapath diagram in Figure 1 and add your signal names to the diagram. This will give you a graphical representation of your system that can be quickly evaluated to determine the meaning of each signal. For any additional signals that appear on your simulation results, put the signals in a table with a short description of that signal.

3 Design

Most of the design for the datapath is pre-determined by Figure 1. The item that is not pre-determined is the timing. For this Design section, you do not need to add any substantial text. Please just reference a timing diagram that shows the timing between each stage. See Figure 2 for an example of a simple timing diagram.

4 Implementation

The Verilog code and explanations of why you implemented this way. Code should be included in the report like Listing 1 on page 1. For this lab, you should describe iMemory and iWriteback (and any submodules you feel are necessary).

Listing 1: Verilog code for implementing a register.

```
`include "definitions.vh"
```

Figure 1: Full Non-Pipelined Datapath

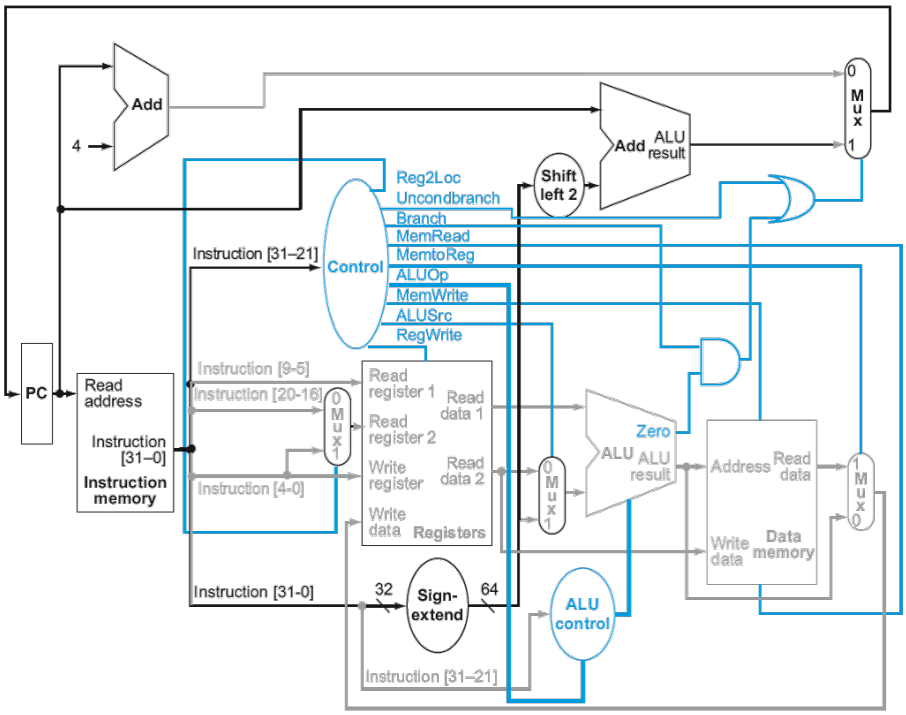
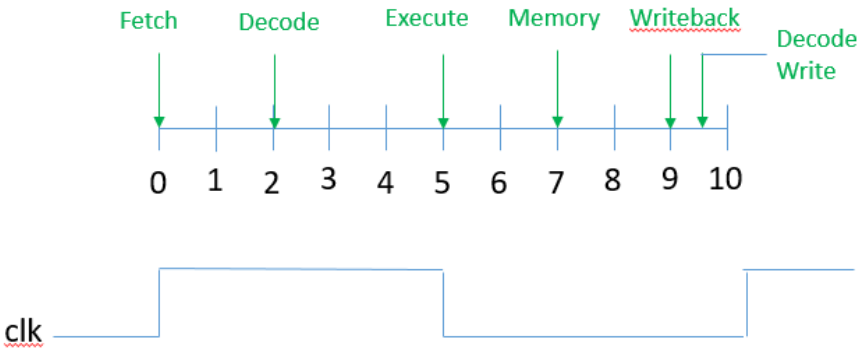


Figure 2: Timing Diagram Example



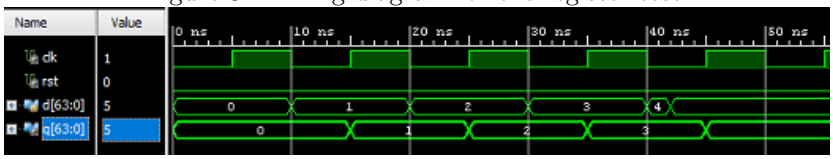
```
module register(
    input clk,
    input reset,
    input  [`WORD-1:0] D,
    output reg [`WORD-1:0] Q=`WORD'b0
    );

    always @(posedge(clk),posedge(reset))begin
        if (reset==1'b1)
            Q<=`WORD'b0;
        else
            Q <= D;
    end

endmodule
```

5 Test

This is where you discuss the test cases you wrote, and what they were designed to test. You should discuss expected errors as well as random errors. Please include:

1. Testbench code (in this case, datapath.v because it is the top-level module)
2. List of assembly commands that you have been using to test your datapath since the iDecode stage
3. Binary for the commands you have been using to test your datapath since the iDecode stage
4. Expected Results Table
5. Simulation Results from these commands (that should match your expected results). A sample simulation diagram is in Figure 3 on page 4.
6. Assembly code for the "Division Problem" C code that I gave you
7. Instruction Data File for the Division Problem
8. Register Data File for the Division Problem
9. Memory Data File for the Division Problem
10. Simulation Results for the Division Problem

Figure 3: Timing diagram for the register test.



Listing 2: Verilog code for testing a register.

```
`include "definitions.vh"


module test_regs;

wire clk;
wire rst=0;
reg [`WORD - 1:0] d;
wire [`WORD - 1:0] q;

oscillator clk_gen(clk);

register UUT(
    .clk(clk),
    .reset(rst),
    .D(d),
    .Q(q)
    );

initial
begin
    d<='WORD'd0; #`CYCLE;
    d<='WORD'd1; #`CYCLE;
    d<='WORD'd2; #`CYCLE;
    d<='WORD'd3; #`CYCLE;
    d<='WORD'd4; #(`CYCLE/5);
    d<='WORD'd5; #(`CYCLE*4/5);
end

endmodule
```

6 Conclusions

Overview the main points you want to stick in peoples minds and answer key questions you want to stick in peoples minds. Did it work? How well? What would you have done differently? What did you learn?