CME341 Assignment 6

```
Revised Oct. 24, 2014: The test bench was changed to catch up with
                       the change made to microprocessor a year ago
                       when the sync_reset flip/flop was moved outside
                       the program sequencer.
Revised Oct. 22, 2020: Added a $stop statement to the testbench
Revised Dec. 13, 2020: Removed a line from the testbenc
Revised Oct. 24, 2022: Corrected typo in testbench. Added a line ending
                       semicolon to initial procedure: "initial #25 $stop;"
```

1. A top level block diagram of the microprocessor is provided by the instructor (as part of notes on the web). The diagram of interest contains blocks like "program sequencer", "instruction decoder" etcetera. This block diagram will be translated to a Verilog structural description, which ends up as a module containing an instantiation for each block in the block diagram. This block diagram is fully annotated, which means the wires that interconnect the instantiated blocks have all been named and properly sized and the signals inside the blocks (that connect to the wires) are also be named. Any fully annotated block diagram can be converted to structural description in Verilog HDL.

   Create a Verilog HDL description (structural) of the top module of the microprocessor. Be sure to declare all the wires that run between blocks as wires.

2. Design a prototype for the program sequencer. Use a behavioral description. Do not use a second level of hierarchy (i.e. do not instantiate anything in the program sequencer module). Include behavioral descriptions of both the program counter and the next address logic in the one module.

   Note that the conditional jump instruction will execute as a "jump" if and only if the zero flag is **not** set at the time the instruction is to be executed. Since the zero flag is only written when an ALU instruction is executed, the conditional jump instruction is only executed as a "jump" if the last ALU instruction, no matter how long ago it was executed, produced a result that is **not zero**.

3. The Verilog description below is supposed to be a test bench that tests the program sequencer.

```
‘timescale 1us / 1ns
module test_program_sequencer();

reg clk;

reg sync_reset;
reg jump;
reg zero_flag;
reg conditional_jump;
reg correct_address;
reg [3:0] count;
reg [3:0] LS_nibble_ir;
wire [7:0] pm_address;

initial clk = 1’b0;
always #0.5 clk = ~clk;

initial #25 $stop; // suspend the simulation after 25 clock cycles

initial count = 4’h0;
always @ (posedge clk)
if (count == 4’b1001)
    #0.005 count <= 4’b0000; // hold count for 5 ns
                             // after clock edge to model
                             // a hardware counter
else
    #0.005 count <= count + 4’h1; // hold count for 5 ns
                             // after clock edge to model
                             // a hardware counter


always @ *
case (count)
4’b0000:
```

```
     begin
     sync_reset = 1'b1;
     jump = 1'b0;
     zero_flag = 1'b1;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'h0;
     // on first time through the loop pc = 0
     if (pm_address == 8'h00) correct_address =1'b1;
     else correct_address =1'b0; // pm_address in wrong
     end
4'b0001:
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b1;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'hA;
     if (pm_address == 8'h01) correct_address =1'b1;
     else correct_address =1'b0;
     end
4'b0010:
     begin
     sync_reset = 1'b0;
     jump = 1'b1;
     zero_flag = 1'b1;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'hA;
     if (pm_address == 8'hA0) correct_address =1'b1;
     else correct_address =1'b0;
     end
4'b0011:
     begin
     sync_reset = 1'b0;

     jump = 1'b0;
     zero_flag = 1'b0;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'hA;
```

```
     if (pm_address == 8'hA1) correct_address =1'b1;
     else correct_address =1'b0;
     end

4'b0100:
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b0;
     conditional_jump = 1'b1;
     LS_nibble_ir = 4'h1;
     if (pm_address == 8'h10) correct_address =1'b1;
     else correct_address =1'b0;
     end
4'b0101:
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b1;
     conditional_jump = 1'b1;
     LS_nibble_ir = 4'h1;
     if (pm_address == 8'h11) correct_address =1'b1;
     else correct_address =1'b0;
     end

4'b0110:
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b1;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'hB;
     if (pm_address == 8'h12) correct_address =1'b1;
     else correct_address =1'b0;
     end
4'b0111:
     begin
     sync_reset = 1'b0;
```

```
     jump = 1'b0;
     zero_flag = 1'b0;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'h3;
     if (pm_address == 8'h13) correct_address =1'b1;
     else correct_address =1'b0;
     end
4'b1000:
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b0;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'h3;
     if (pm_address == 8'h14) correct_address =1'b1;
     else correct_address =1'b0;
     end
4'b1001:
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b0;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'h3;
     if (pm_address == 8'h15) correct_address =1'b1;
     else correct_address =1'b0;
     end
default: // this should never happen
     begin
     sync_reset = 1'b0;
     jump = 1'b0;
     zero_flag = 1'b0;
     conditional_jump = 1'b0;
     LS_nibble_ir = 4'h0;
     correct_address = 1'b0;
     end

endcase
```

```
program_sequencer prog_sequencer (
       .clk(clk),
       .sync_reset(sync_reset)
       .dont_jump(zero_flag),
       .jmp(jump),
       .jmp_nz(conditional_jump),
       .jump_addr(LS_nibble_ir),
       .pm_addr(pm_address)
       );
endmodule
```

(a) Draw a block diagram for the test bench.

(b) Does this test bench fully test the program sequencer?

(c) It is pointed out in the comment under the case item for `count == 4'b0000` that on the first time through the loop the `PC` is zero.

Does the initial value of the pc (i.e. the value the first time count is zero) affect the result?

4. Design a test system (or test bench) for the program sequencer (hopefully different and a bit better than the one given above). Please hand in a block diagram of the test system and a Verilog HDL description of each block. Test the system and comment on the results.