CS-UY 2214 — Recitation 4

Introduction

Complete the following exercises. Unless otherwise specified, put your answers in a plain text file named `recitation4.txt`. Number your solution to each question. When you finish, submit your file on Gradescope. Then, in order to receive credit, you must ask your TA to check your work. Your work should be completed and checked during the recitation session.

You may consult the E15 cheat sheet, which is available on Brightspace.

Problems

1. Read the following Verilog module. Do not enter it into your computer.

```
module mystery_module(clk, reset, A, B);
    input clk;
    input reset;
    input [3:0] A;
    output [3:0] B;

    reg[3:0] state;

    assign B = state ^ 4'b1010;

    always @(posedge clk)
        if (reset)
            state <= 4'b0000;
        else
            state <= state | A;
endmodule
```

A note on syntax: recall that `4'b1010` means a 4-bit value with binary value 1010.

Assume the module is initially reset (by setting `reset` input to 1). Then, `reset` is set to 0 and the following 4-bit inputs (here expressed as unsigned decimal numbers) are sent on input `A`, one on each subsequent clock cycle: 0, 1, 0, 9, 8, 15, 0.

Complete the following table, so that the left column shows the value of input `A` and the right column shows the value output on `B` in the same clock cycle. Keep in mind that the order of the inputs is significant, as the module stores state internally.

Write only the values of `B`; it is not necessary to reproduce the entire table. Express your answers as unsigned decimal numbers. Complete this question mentally, without entering the code into your computer.

A	state	B
0		
1		
0		
9		
8		
15		
0		

2. For each of the following E15 instructions, provide a one-sentence English description of what it does. Include in your description any effect on the value of registers, including the zero flag and the program counter.

   (a) `{mov,  Rg3, Rg1, 4'b0000}`

   (b) `{addi, RXX, Rg2, 4'b0101}`

   (c) `{cmp,  Rg1, Rg3, 4'b0000}`

   (d) `{jz,   RXX, RXX, 4'b0011}`

3. The following four instructions are encountered in sequence in an E15 program. Describe what each instruction does individually. Then, give the final value of registers Rg0 and Rg1 at the end of this sequence of instructions.

```
1. {movi, RXX, Rg0, 4'b0010}
2. {movi, RXX, Rg1, 4'b0100}
3. {add,  Rg0, Rg1, 4'b0000}
4. {subi, Rxx, Rg1, 4'b0001}
```

4. For each of the following E15 assembly instructions, represent the instruction as a 12-bit binary number. Write all 12 bits. For `RXX`, use the value `00`.

   (a) `{add,   Rg0, Rg2, 4'b1100}`

   (b) `{subi,  RXX, Rg1, 4'b0110}`

   (c) `{cmpi,  RXX, Rg3, 4'b1101}`

   (d) `{jnz,   RXX, RXX, 4'b0010}`

5. For each line of binary machine code representing an instruction, translate it to its E15 assembly form. When the source or destination register is ignored by the opcode, use `RXX` to signify this.

   (a) `101000010000`

   (b) `000000000110`

   (c) `110010110000`

   (d) `100100101011`

6. Here is a complete E15 program:

```
/*           OPCODE SRC  DST  IMMDATA */
myROM[0]   = {movi , RXX , Rg0 , 4'b0001};
myROM[1]   = {movi , RXX , Rg1 , 4'b0001};
myROM[2]   = {cmpi , RXX , Rg0 , 4'b0100};
myROM[3]   = {jz ,   RXX , RXX , 4'b0100};
myROM[4]   = {add ,  Rg1 , Rg1 , 4'b0000};
```

```
myROM[5]  = {addi, RXX, Rg0, 4'b0001};
myROM[6]  = {jmp,  RXX, RXX, 4'b1100};
myROM[7]  = {jmp,  RXX, RXX, 4'b0000};
myROM[8]  = {jmp,  RXX, RXX, 4'b0000};
myROM[9]  = {jmp,  RXX, RXX, 4'b0000};
myROM[10] = {jmp,  RXX, RXX, 4'b0000};
myROM[11] = {jmp,  RXX, RXX, 4'b0000};
myROM[12] = {jmp,  RXX, RXX, 4'b0000};
myROM[13] = {jmp,  RXX, RXX, 4'b0000};
myROM[14] = {jmp,  RXX, RXX, 4'b0000};
myROM[15] = {jmp,  RXX, RXX, 4'b0000};
```

(a) What is the final value of Rg0? What is the final value of Rg1?

(b) Let's change the instruction at address 2 to this:

```
myROM[2]  = {cmpi, RXX, Rg0, 4'b0101};
```

Now, if we run the program again, what is the final value of Rg0? What is the final value of Rg1?

7. Write an E15 program that will use Rg3 to count down from 15 to 0, then stop. You must use a loop.