Design of Digital Circuits: Lab Report		
Lab 8: Full System Integration (Session II)		
Date	27.05.2019	Grade
Names	David Zobrikofer	
	Sven Pfiffner	Lab session / lab room
		Fri: 10-12 | Room E 26.3

You have to submit this report via Moodle.

Use a zip file or tarball that contains the report and any other required material. Only one member from each group should submit the report. All members of the group will get the same grade.

The name of the submitted file should be *LabN_LastName1_LastName2.zip* (or *.tar*), where *LastName1* and *LastName2* are the last names of the members of the group.

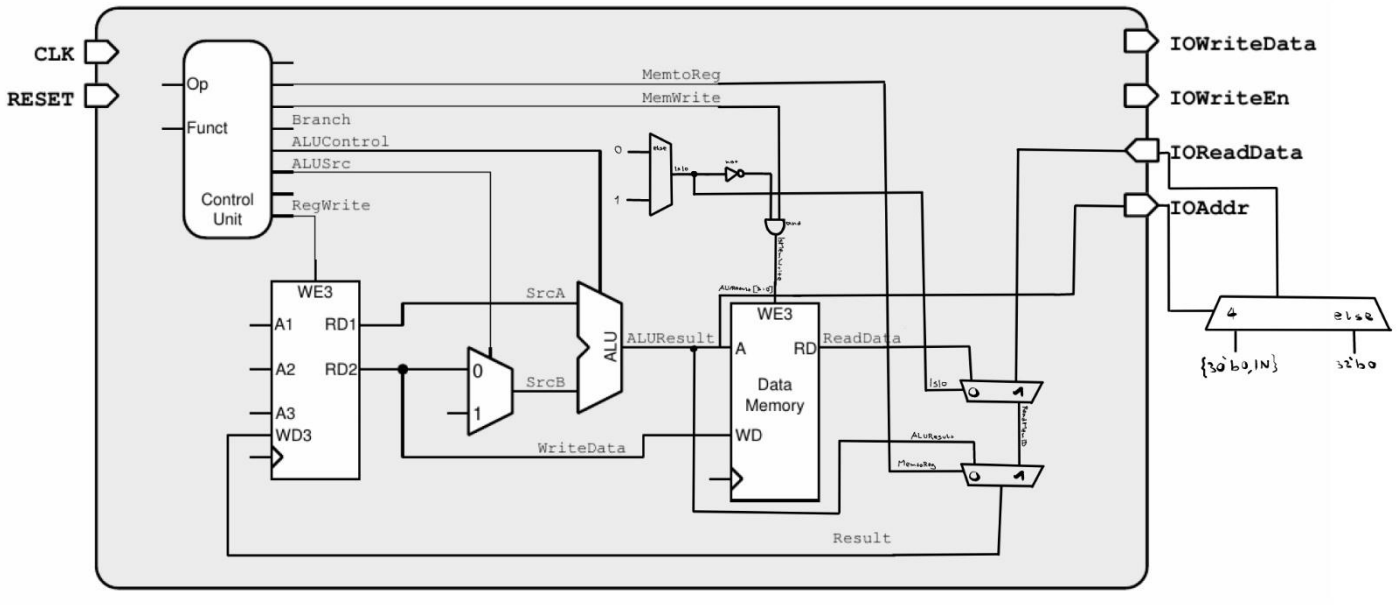
Note 1: Please include all the required material. No links/shortcuts are accepted.

Note 2: The deadline for the report is a hard deadline and it will not be extended.

Exercise 1

Below is a part of the MIPS block diagram. Draw the necessary modifications for the memory mapped I/O on this block diagram. (We are only interested in the SW and LW instructions; the rest of the block diagram has been left out on purpose. *Hint: If your circuit works you already implemented this in the* MIPS.v *module.)*

We have drawn it like we have implemented in in Verilog. Attached you can also find the code for the top.v and the mips.v module where what we have drawn can be seen in Verilog.



Exercise 2

Using Figure 1 as a reference, what additional hardware/architectural changes are needed in the top module (*top.v* file) to implement Challenge 2 described in the Manual of Lab 8, Session 2? You can either draw the additional circuitry required or write in your own words here.

First of all, the top module must be extended by a 1-bit input *"DIRECTION"* which we will use to control the direction of the snake. An additional 32-bit wire then is assigned to either *32'h0004* or *32'hFFFC* (depending on the value of *"DIRECTION"*). Finally, we assign either the default value or said wires value to *IOReadData*:

```
module top(

           ...

           input DIRECTION,

           ...

    );

...

wire [31:0] directionWire;

...

assign directionWire = DIRECTION==1'b1 ? 32'h0004: 32'hFFFC;

...

always @(*)
   begin
        case(IOAddr)
             4'b0100: IOReadData = {30'b0, IN};
             4'b1000: IOReadData = directionWire;
             default: IOReadData = 32'b0;
        endcase
   end
```

Note that, in order to work with the modified *IOReadData*, some minor changes are required in the snake-patterns.asm as well. We have gone the lazy way and just added the directionWire to our pattern address, furthermore we have added another branch if equals to check that the address never gets negative. One could have also passed the one bit direction directly to the program and then have the program do the necessary addition or subtraction of the pattern locations, but we have opted for this way since that way it is more fun to implement, we can abstract something software would have to do away into the hardware.

To see our circuit in action please see the attached GIF file.

Feedback

If you have any comments about the exercise please add them here: mistakes in the text, difficulty level of the exercise, or anything that will help us improve it for the next time.