CSE3666

Homework 6 (100 points)

Deadline: By the end of Sat, 3/30/2024.

Figure 4.21 can be downloaded in HuskyCT. Note that **the figure is copyrighted**. We use it in this course for educational purposes. Do not distribute it further. The figure posted in HuskyCT also has additional signal names.

The single-cycle (RISC-V) processor refers to the design shown in Figure 4.21. We also assume the ImmGen module handles R-Type instructions like I-type instructions, i.e., when executing an R-type instruction, the ImmGen module generates the 32-bit immediate by sign-extending the highest 12 bits in the machine code.

1. Find the following signal values when the single-cycle processor executes the following instruction. The number before the colon is the address and the one after the colon is the machine code. Explain your answers.

   ```
   0x00400200: 0xFE542023
   ```

   All signals generated by the main control.
   opcode
   rs1
   rs2
   rd
   Immediate
   ALU operation
   BranchTarget
   PCSrc
   NextPC

2. Assume register x4 is `0x0000FFF0` and register x16 is `0x09AB000C` before the execution of the following instruction. A transient fault causes MemWrite to be 1 when the single-cycle processor executes the instruction. Describe how the fault affects the state of the processor. Specifically, answer the following questions. At the beginning of the next cycle, what is the value to be stored in PC? How is the register file changed? And how is the data memory changed? Are these changes correct?

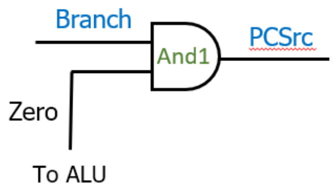
   ```
   0x0040033C: 0x01020233        # add  x4, x4, x16
   ```

3. In this problem, we improve the processor in Figure 4.21 to support JAL and JALR instructions. Assume the following two components have already been revised.

   - The main control. It has two additional outputs J and JR. J is 1 if and only if the instruction is either JAL or JALR. JR is 1 if and only if the instruction is JALR. The two signals can be generated from the opcode.
   - ImmGen. ImmGen can generate correct immediate for all types of instructions.

   The submitted diagram will include components in Figure 4.21, except for the control module, and your revisions to the diagram you will make in Tasks a) and b). Read the entire question first. It may help you to plan the diagram.

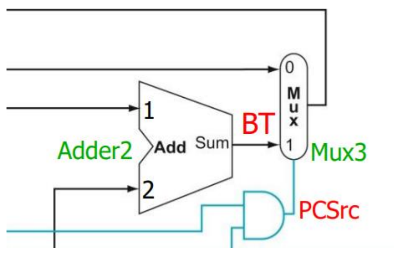
   - Although we do not need to include the control module, all control signals should be labeled. For example, Branch is labeled in the following figure, although it is not connected to the control module.



   - We also need to name every signal/gate/module we add into the diagram.
   - All wires go either horizontally or vertically, and lines are straight.
   - The input ports of MUX are labeled with 0 and 1.
   - The diagram is clean.

   Tasks:

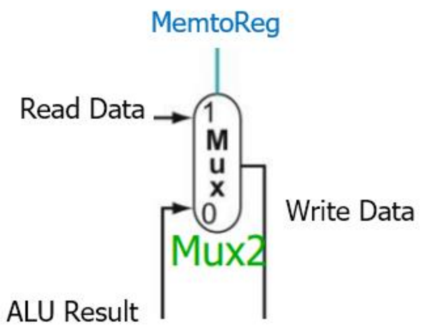
   a) We change the diagram to set the correct next PC value at the output of Mux3 for JAL, JALR and branches. The part of diagram we need to change is shown in the following figure.



   The modifications include the following.
   - Add a MUX so Adder2 can compute the correct target address for JAL, JALR and branches.
   - Change the logic that generates PCSrc, the select signal of Mux3.

Explain how the target address is computed for JAL and JALR instructions, and how the select signals of MUXes are generated. Write a logic expression if necessary.

b) We then change the diagram so the correct value can be saved in the destination register for JAL and JALR. The part of diagram we need to change is shown in the following figure. We can achieve the goal by adding a MUX before Mux2. In addition to revising the diagram, explain how to generate the select signal of the newly added MUX.



c) Specify the value of control signals for JAL and JALR. The signals controlling the MUXes can be set to don't care if they do not affect the execution of instructions. However, the signals indicating instruction types are always set. For example, the Branch signal is always set to indicate whether the instruction being executed is a branch.

Inst.	**ALU Src**	**Memto Reg**	**Reg Write**	**Mem Read**	**Mem Write**	**Branch**	**J**	**JR**
JAL								
JALR								