3666 hw 6

Casey Provitera

March 2024

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

1

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
   0xFE542023 to binary
   = 0b 1111 1110 0101 0100 0010 0000 0010 0011
   opcode of 010 0011 denotes S type instruction
   funct3 of 010 denotes sw
   imm = 1111 111 0000 0 = 1111 1110 0000
   rs1 = 01000
   rs2 = 00101

   All signals generated by the main control.
   opcode = 0b 010 0011
   rs1 = 0b 01000
   rs2 = 0b 00101
   rd = not generated so default to 0b00000
   Immediate = 0b 1111 1110 0000
   after sign extension = 0xFFFFFFE0
   ALU operation = lw/sw instruction = 0b 0010

BranchTarget = current address plus immediate
0x00400200 + 0xFFFFFFE0, change to unsigned versions for easier math
0b0000 0001 1111 (flipped bits) +1 = 0b 100000 = 32 in decimal
back to hex 32 = 0x020
0x00400200 - (need minus since the sign version was negative) 0x20
0x00400200 - 0x20 = 0x004001E0
PCSrc = not a branch call so is 0
NextPC = current addres + 4_{10}
0x00400200 + 4_{10} = 0x00400204

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

1. At the beginning of the next cycle, what is the value to be stored in PC?
The value stored in PC is 0x00400340, this is because the mux that either chooses PC+4 or PC plus the branch offset chooses PC+4 since this is not a branch call.

2. How is the register file changed? And how is the data memory changed? Are these changes correct?
The register files is changed to now have the out put of the "add x4, x4, x16" call in the Write data register. But the memory data is not changed since the mux that tells us to either us strictly the ALU output or the data from the "Data memory" block chooses to use the strict ALU output for this call since the value of 'MemtoReg' by default is set to 0 and is not changed in this problem.
These changes are in fact correct. Even though 'MemWrite' was set to 1 by accident, the "add x4, x4, x16" call still evaluate correctly, it does slow down the function call since instead of simply skipping the "Data memory" block we have to calculate within it. But other than a slight slow down no errors arise from the fault.

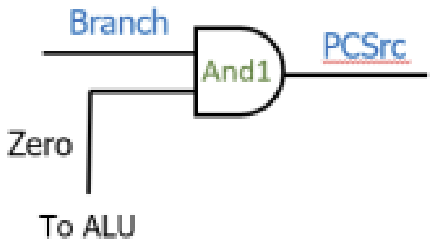
3

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:

1. 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.

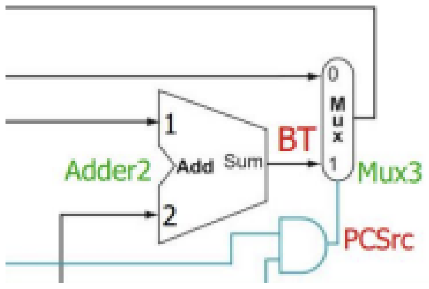


Figure 1: Original

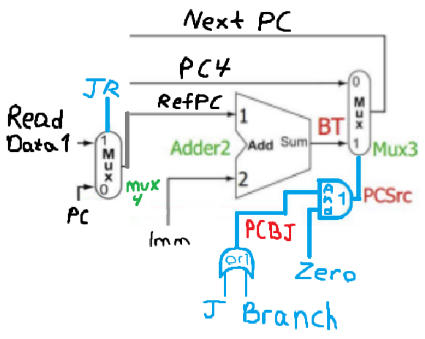


Figure 2: Updated

First we needed to add the MUX input for the adder since the JAL and JALR calls work differently from branch and none branch calls when calculating the next instruction address.
For Jal calls we go to the label name address and store PC+4 in rd, so this can work like a branch call, by just having the PC + the immediate. JALR is different, for JALR we go to a memory location calculated using an immediate (or offset) and rs1. This is why we have the mux4 in place. If it sees the call is a JR type then it uses read data 1 or rs1 to be added to the immediate instead of PC.
Second we needed to add the or gate before the and gate that goes as the input for the mux telling the program where to get the Next PC address from. This is because we not only need to use the immediate for branch calls now but also for J type calls which is why we added the or gate.
the logic becomes the following PCSrs = Zero and (J or Branch)

2. 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.

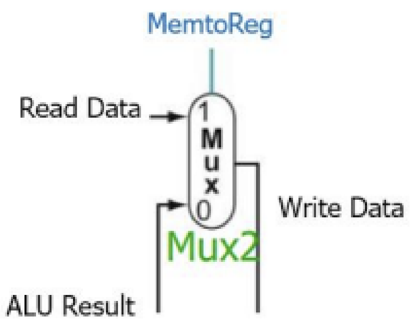


Figure 3: Original

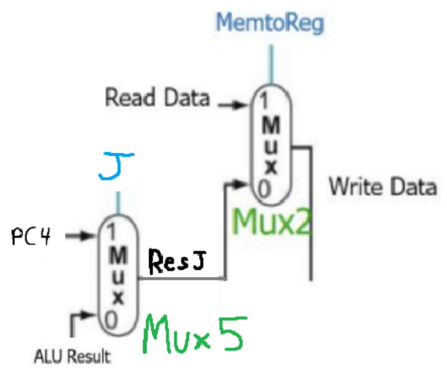


Figure 4: Updated

The J command comes from the generated opcode from both JAL and JALR calls. This is the only new signal in the updated diagram, meaning this addition to the single cycle processor is very easy.

3. 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.	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	J	JR
JAL	X	0	1	0	0	0	1	0
JALR	X	0	1	0	0	0	1	1

The JAL and JALR instructions have identical signals besides the JR signal since that is used to calculate next PC for only JALR. ALUSrc can be either 1 or 0 since the ouptut from the ALU is not used in the process call. Instead the adder output is used.