

Name: _____ Section: _____

Name: _____ Section: _____

Name: _____ Section: _____

Practical 09

Read and perform the practical guide. Answer the questions after you have completed the practical. **Be sure to keep the formatting in this file (questions should not change pages).**

[4] (Need) Why is forwarding needed in a pipelined processor? Explain this to someone who is familiar with processor architectures (they understand the 5 pipeline stages) but does not know the details of the RISC-V instruction set (e.g., you cannot reference specific instructions in your explanations).

[4] (Need/Iteration) The testbench provides you with a sequence of tasks for you to progressively implement forwarding:

1. `test_no_hazard_detection()`
2. `test_write_then_read_hazard_detection()`
3. `test_WB_to_EX_fwd()`
4. `test_MEM_to_EX_fwd()`

Did you follow this sequence of tests to implement your forwarding unit, or did you skip some steps? Is it necessary to follow this sequence of 5 test? Explain your team's progression through implementing the forwarding unit.

[4] (Iteration) Did your forwarding unit work the first time you implemented it? What revisions did you have to make to your stall detection logic? Explain what the error was and how you fixed it.

[4] (Need) Why is stalling needed for a pipelined processor? Explain this to someone who is familiar with processor architectures (they understand the 5 pipeline stages) but does not know the details of the RISC-V instruction set (e.g., you cannot use `lw` in your explanation).

[4] (Iteration) Did your stall/hazard detection unit work the first time you implemented it? What revisions did you have to make to your stall detection logic? Explain what the error was and how you fixed it.

[4] (Correctness) A teammate argues that implementing the logic for stall detection is too tedious, and instead to unconditionally stall every single instruction once. They argue that the processor will still work and that's all they care about. Is this teammate technically correct? Why or why not?

[4] (Correctness/Performance) As discussed in lecture, there are two approaches to resolve `lw` into `sw` forwarding:

`lw x14, 24(x2)`

`sw x14, 100(x2)`

The first is to implement `WB` into `MEM` forwarding, and the other is to stall `sw` by one cycle. Which approach did your team decide to use? Explain the advantage/disadvantage of the approach your team has chosen.

[4] (Correctness/Performance) The instructions hint at two approaches to resolve forwarding into branches:

```
add    x14, 24(x2)
```

```
beq    x14, x0, L
```

How did your team approach solving this data hazard? Explain the advantage/disadvantage of the approach your team has chosen.

[4] (Correctness) Take a screenshot of ModelSim running a test to verify the correctness of the case described in the previous question. Annotate the screenshot to show how you know the tests are passing. Then answer the questions below.

Replace this textbox with your screenshot

How many test variations did you and your team decide to create and test in total? Did you reach a “substantial” check? Why or why not? *For your answer “time” is a legitimate resource limitation you are allowed to mention.*

[4] (Need/Correctness) The instructions direct you to a forwarding datapath problem when trying to forward an instruction such as `lui` from **MEM** to `addi` in **EX**:

```
lui  x10, 0x00ABD
addi x5, x10, 0xDEF
```

Explain why this problem exists, and how you and your team resolved this forwarding problem.

[4] (Correctness) Take a screenshot of ModelSim running a test to verify the correctness of the case described in the previous question. Annotate the screenshot to show how you know the tests are passing. Then answer the questions below.

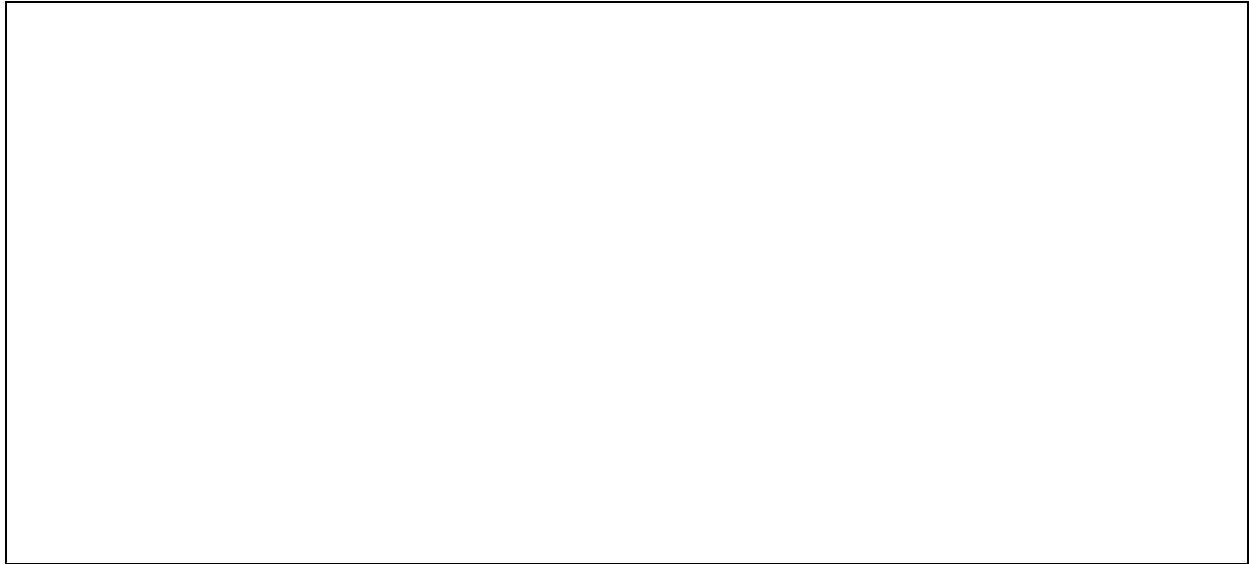
Replace this textbox with your screenshot

How many test variations did you and your team decide to create and test in total? Did you reach a “substantial” check? Why or why not? *For your answer “time” is a legitimate resource limitation you are allowed to mention*

[4] (Correctness/Performance) What purpose of running `relPrime` on your processor aside from calculating a relative prime?

[4] (Need) Describe your new instruction. Make sure you include its name, command, syntax, and purpose.

[4] (Need) Draw out your new instructions' instruction format. Be sure to label each field of bits and explain what each field does (including how it is being addressed if applicable). If you need to create a new addressing mode, describe it here as well.

A large, empty rectangular box with a thin black border, intended for the student to draw out their new instruction format. The box is currently blank.

[4] (Correctness) How did you account your new instruction for flushing, stalling, and/or forwarding? Discuss hazards that your instructions may cause and what you had to update to your processor to resolve these hazards.

[4] (Performance) Explain how you expect this new instruction to affect the performance of your processor.

[4] (Performance) If applicable, compare the runtime (in number of cycles) `relPrime` (5040) under the base RISC-V ISA and with your new instruction.

[8] What was the biggest challenge in implementing and testing your new instruction? Explain in 100 words or less.

[10] What is the single biggest thing you learned from designing and implementing your new instruction? Explain in 100 words or less.

[0] What is the git commit ID for your final commit of your code. This is required to pass the assignment. Check Practical 1 for instructions on how to get the correct commit ID.