Name: _____ Section: ____

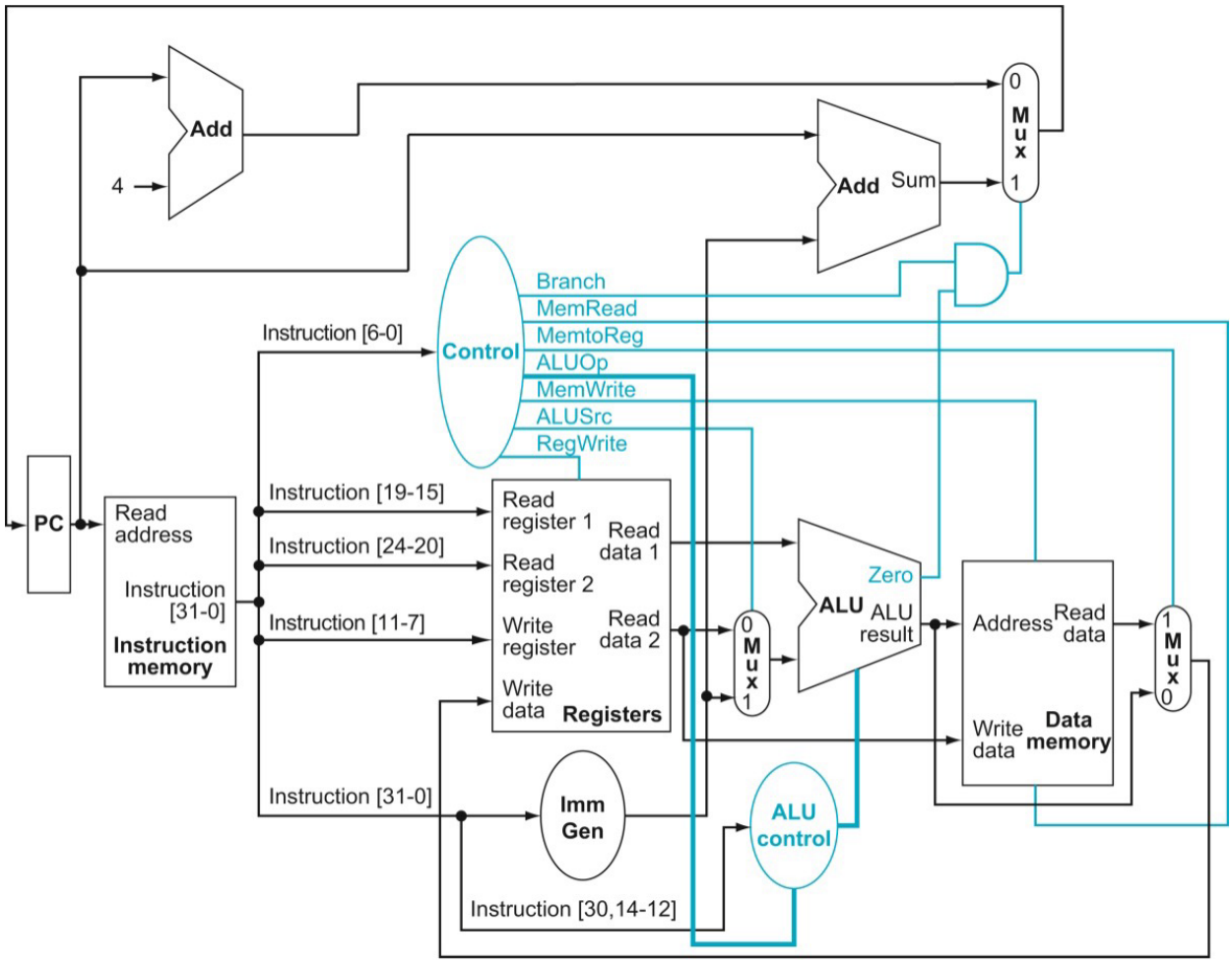
Name: _____ Section: ____

Name: _____ Section: ____

Practical 05

Read and perform the practical guide. Answer the questions after you have completed the practical. **Be sure to keep the formatting of 1-2 questions per page.**

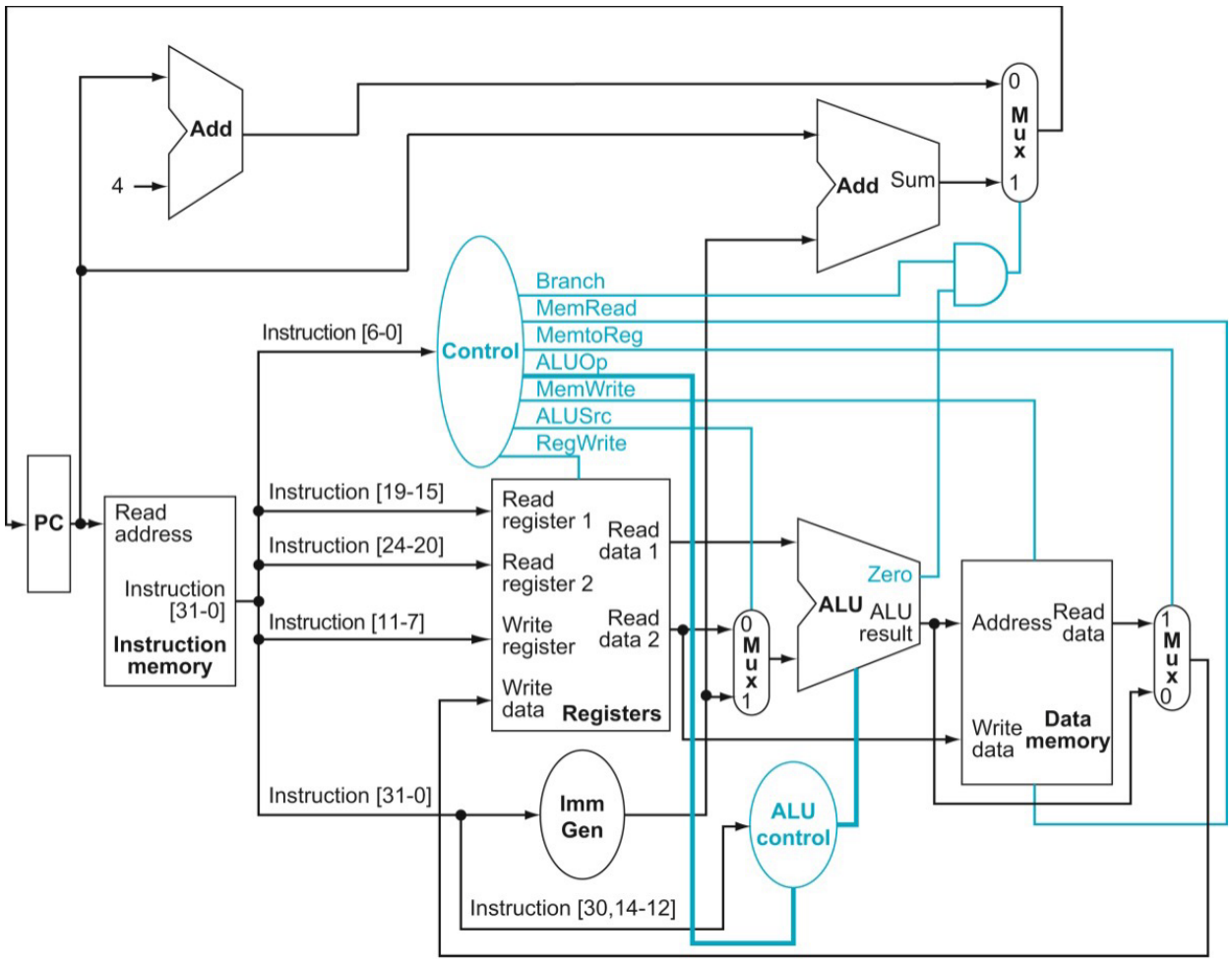
[4] (Need) Highlight the datapath for R type instructions:



[4] (Correctness/Iteration) Take a screenshot of the ModelSim running `tb_processor` for R type tests. This should include the waveform with adequate signals grouped for organization as well as the transcript documenting the instructions you are running for the test.

Replace this textbox with your screenshot

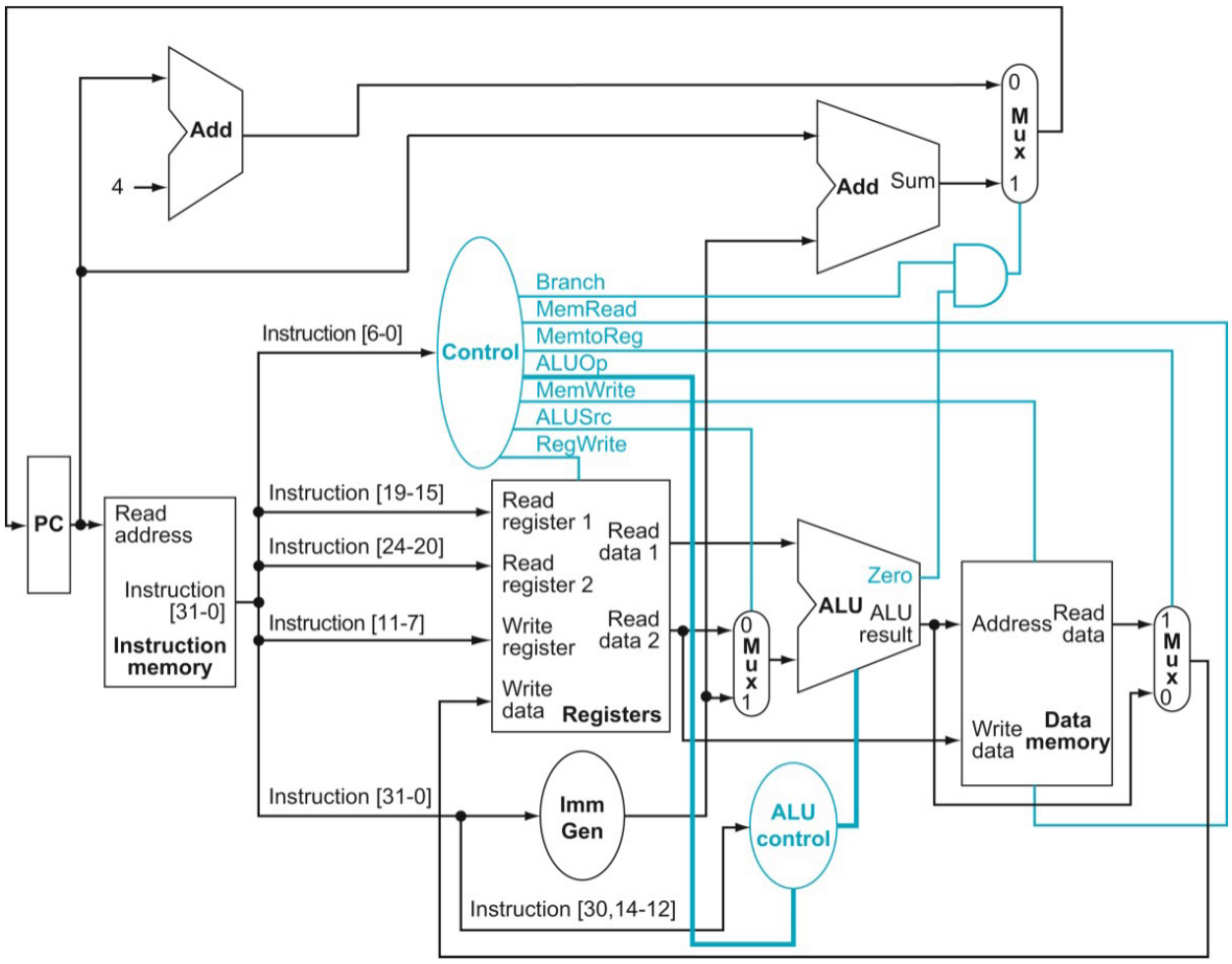
[4] (Need) Highlight the datapath for I type instructions:



[4] (Correctness/Iteration) Take a screenshot of the ModelSim running `tb_processor` for I type tests. This should include the waveform with adequate signals grouped for organization as well as the transcript documenting the instructions you are running for the test.

Replace this textbox with your screenshot

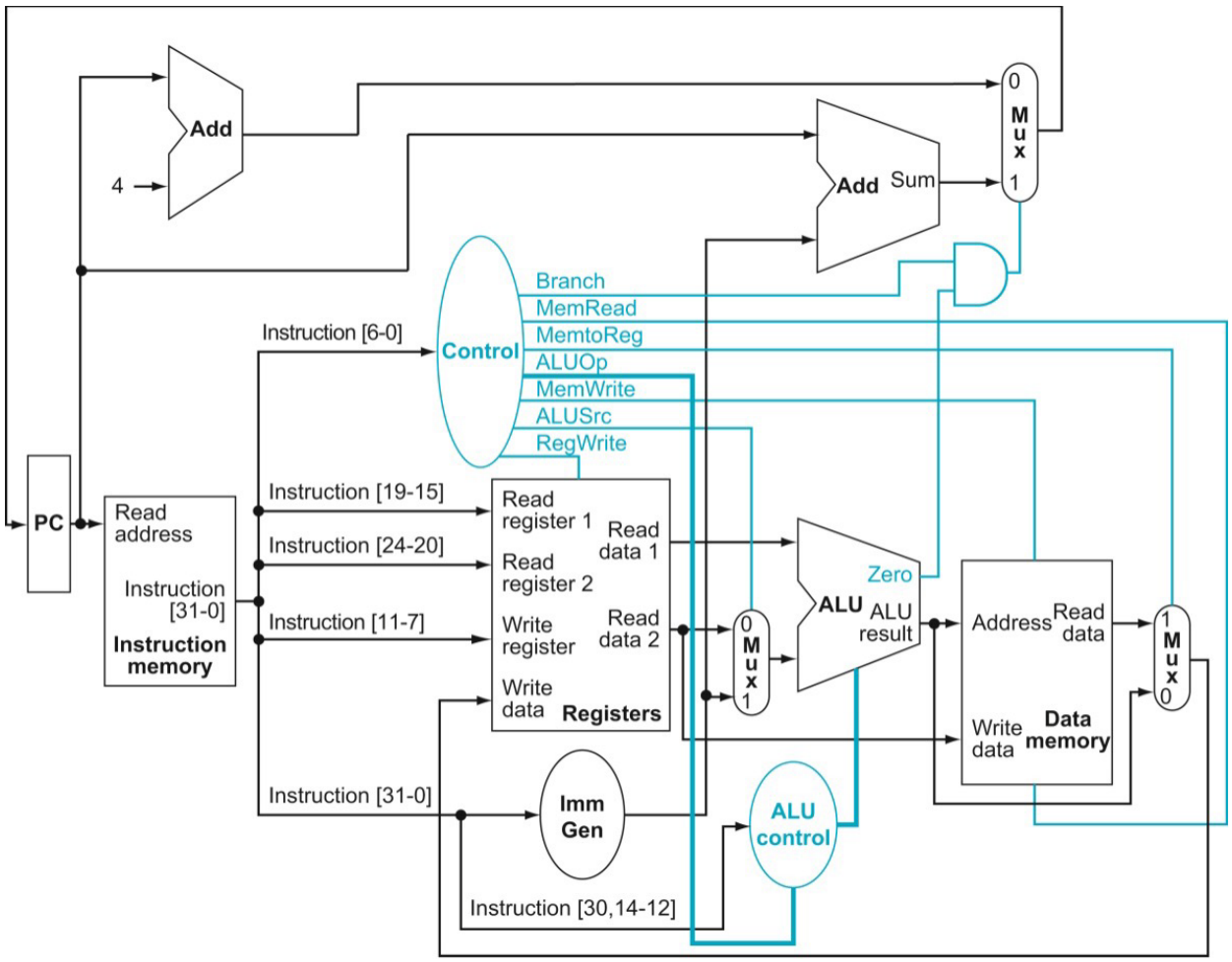
[4] (Need) Highlight the datapath for `sw` and `lw`:



[4] (Correctness/Iteration) Take a screenshot of the ModelSim running `tb_processor` for `lw` and `sw`. This should include the waveform with adequate signals grouped for organization as well as the transcript documenting the instructions you are running for the test.

Replace this textbox with your screenshot

[4] (Need) Highlight the datapath for SB type instructions:

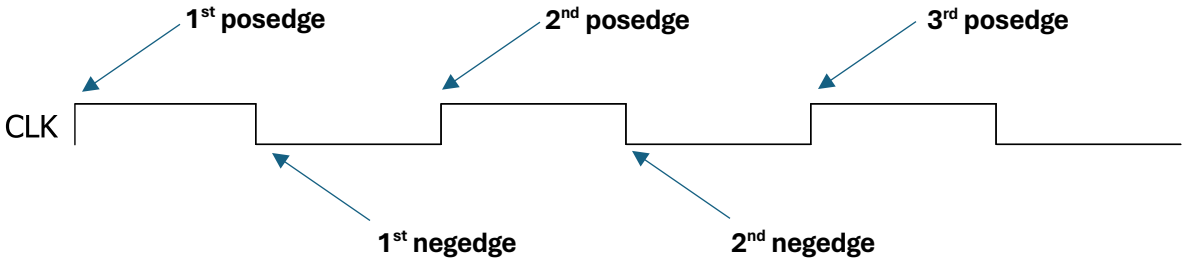


[4] (Correctness/Iteration) Take a screenshot of the ModelSim running `tb_processor` for SB type tests. This should include the waveform with adequate signals grouped for organization as well as the transcript documenting the instructions you are running for the test.

Replace this textbox with your screenshot

[8] (Correctness) How did you choose what instructions to include for your testbenches? What strategy/thinking process did you follow in order to ensure your set of testbench instructions sufficiently checks the validity of your processor (make sure to discuss this with specific reference to each instruction type implemented)? Discuss how you ensured adequate coverage of the variety of instructions, registers, immediates, etc.

[8] (Correctness) Below is a timing diagram:



Consider two instructions that runs on the single-cycle processor:

1. `add a0, t1, t2`
2. `lw  t3, 8(t4)`

Fill out the following table indicating which of the following clocked operations are performed:

- **`add`'s instruction read**
- **`add`'s register read**
- **`add`'s register write**
- **`add`'s ALU operation**
- **`add`'s PC write**
- **`lw`'s instruction read**
- **`lw`'s register read**
- **`lw`'s register write**
- **`lw`'s ALU operation**
- **`lw`'s memory read**
- **`lw`'s PC write**

Clock Timing	**Operation(s)**
1st posedge	`add`'s instruction read
1st high level	
1st negedge	
1st low level	
2nd posedge	`lw`'s instruction read
2nd high level	
2nd negedge	
2nd low level	
3rd posedge	

[8] Need I asked ChatGPT the following prompt: "For the components of PC, instruction memory, register file, and data memory in a RISC-V processor, how should these components be clocked for a single-cycle architecture?". ChatGPT explained that all components must complete their operations within one clock cycle, and thus suggested the following clock timings:

Component	Clocked?	Operation Timing
PC	Yes	Updated at clock edge (start of cycle)
Instruction Memory	No	Combinational read (fetch phase)
Register File	Yes (write)	Read: combinational Write: clocked
Data Memory	Yes	Read/Write usually clocked

Ignoring the fact that ChatGPT did not detail posedge and negedge timing, will this proposed set of clock timing work? Why or why not?

Full ChatGPT transcript can be found at the end of the worksheet.

[8] (Performance) Explain how this single-cycle implementation might have a bunch of "unused time" during a clock cycle.

[8] (Correctness) Describe the most important thing your team had to test in the process of implementing branches.

[8] (Iteration) Explain whether implementing `beq` first before implementing the entire branch datapath was or was not necessary for your group's implementation process.

[8] What was the biggest challenge in implementing and testing the single cycle datapath? Explain in 100 words or less.

[10] What is the single biggest thing you learned from writing tests for this practical? 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.

For the components of PC, instruction memory, register file, and data memory in a RISC-V processor, how should these components be clocked for a single-cycle architecture?

In a **single-cycle RISC-V architecture**, all components must complete their operations **within one clock cycle**. This means the entire instruction fetch, decode, execution, memory access, and write back phases happen in one cycle. For this to work, certain timing conventions must be followed:

1. Program Counter (PC)
- **Clocked**: Yes.
- **Why**: The PC needs to update at the **start of every cycle** with the address of the next instruction. This is typically triggered on the **rising edge** of the clock.
- **Update timing**: New PC value is computed during the cycle and loaded into the PC at the next clock edge.

2. Instruction Memory
- **Clocked**: No.
- **Why**: It behaves like a **combinational read-only memory (ROM)** in single-cycle designs. As soon as the PC is provided as input, the corresponding instruction is immediately available (with some access delay).
- **Read timing**: Combinational; instruction is read as soon as the PC value is known.

3. Register File
- **Clocked**: Yes (for **write operations**).
- **Why**:
    - **Reads**: Are **combinational**; register data is available as soon as register numbers are provided.
    - **Writes**: Happen on the **rising edge** of the clock. The register to be written, write data, and write-enable signal must be stable before the clock edge.
- **Write timing**: Occurs at the **end of the cycle**, allowing the write-back stage to complete its computation.

4. Data Memory
- **Clocked**: Yes.
- **Why**: Like the register file, **reads and writes** are typically performed on the **clock edge**, though reads may be combinational depending on implementation.
- **Read timing**:
    - If **combinational**: Output appears once address is stable (usually assumed in single-cycle).
    - If **synchronous**: Read occurs at clock edge (less common for single-cycle).
- **Write timing**: On the **rising edge**, controlled by a write-enable signal.