

CSSE 232 – Computer Architecture I  
Rose-Hulman Institute of Technology  
Computer Science and Software Engineering Department

Quiz 5 - 35 min

## ANSWER KEY

This quiz is **closed book**. You are allowed to use the reference card from the book (attached at the back of this quiz) and one 8.5" × 11" single sided page of hand written notes. You may not use a computer, phone, etc. during the examination.

Write all answers on these pages. Be sure to **show all work** and document your code. Do not use instructions that we have not covered (e.g. no `mul` or `div` but you can use instructions like `slli`, `srl`, `xori`, etc).

RISC-V code is judged both by its correctness and its efficiency. Unless otherwise stated, you may not use RISC-V pseudoinstructions when writing RISC-V code.

Question	Points	Score
<b>Problem 1</b>	25	
<b>Problem 2</b>	25	
Total:	50	

**Problem 1.** Your team is designing a RISC-V inspired 5-stage pipelined processor targeting a device that will frequently have to lookup data in memory and manipulate it. Your team has designed the new R-type instruction ‘increment offset, calculate address’ (`ioca rd, rs1, rs2`) which behaves like this:

```
rd' = rd + rs1 ;increment the address saved in rd by value in rs1
rs2 = rd' + rs2 ;the new rd value is added it to the value in rs2
```

Here `rd'` is used to indicate the new value of `rd`, which is used in the second step. Importantly this instruction updates two registers (`rd` and `rs2`) and the second addition must happen using the result of the first addition.

The possible approaches to adding this instruction to the pipelined datapath are:

1. Adding hardware to existing pipeline stages.
2. Adding new pipeline stage(s).
3. Reuse existing pipeline stages (e.g. use stalls).

One of your team members, **Anika**, wants to take approach (1). She wants to place a new adder into the execute stage and do both additions there.

A second team member, **Kenji**, want to take approach (2). He wants to add a second excute stage after the first where a new ALU will be added to do the second addition.

In both cases the engineers suggest adding a second set of write ports to the register file, so WB will be able to write two registers at once.

- (a) (2 points) Describe in 1-2 sentences the main modifications to the pipeline needed to execute **Anika**'s design plan. Make sure to indicate any new control signals needed. You do not need to worry about hazards when you answer this, just worry about ‘normal’ execution. A reference pipelined datapath is provided after this question, you do not need to modify this datapath.

**Solution:** The new adder output would connect to a mux along with the main ALU output, a new control signal will choose which output gets sent into the EX/MEM stage register in the 'ALUout' slot. A new 'RegWrite2' control signal will be needed during WB.

- (b) (3 points) Briefly discuss the performance implications of **Anika**'s design in 1-2 sentences. What will the effects be on cycle time?

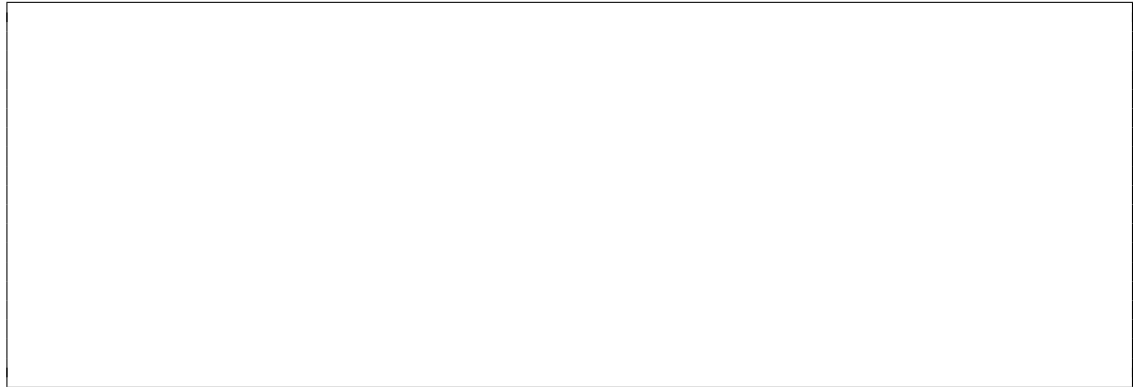
**Solution:**

This will lengthen the cycle time significantly, since we need enough time for both adders to run in series in one cycle.

- (c) (3 points) Briefly discuss the implications of **Anika**'s design for data hazards in 1-2 sentences. What will the effects be on the forwarding unit?

**Solution:** This new instruction will require modification to the forwarding unit, since both rd and rs2 can cause data hazards now. the data that will end up in rs2 will sometimes have to be forwarded to D (for branches), to X (for R/I types), and possibly to M (for sw).

- (d) (3 points) Briefly discuss the implications of **Anika**'s design for control hazards in 1-2 sentences. What will the effects be on the hazard unit? Will this implementation cause any new kinds of stalls or flushes? If so, give an example snippet of assembly code during which the new stall would happen.



**Solution:** But there will be no new stalls possible beyond those in the base RISC-V design, since this instruction ‘finishes’ its calculation in the X stage, just like other R-types. Not a branch/jump, so no flushes.

- (e) (4 points) Briefly summarize the effects of **Kenji's** design on hazards and performance in 1-3 sentences.

**Solution:**

1. no effect on cycle time, or 'normal' CPI
2. forwarding will need to send stuff to and from the new execute stage
3. may create a new 'X2' to 'X' stall eg:

```

ioca t0, t1, t2      F D X X2 M WB
add t3, t2, t4      F D * X X2 M WB
//stall in X to get the t2 data

```

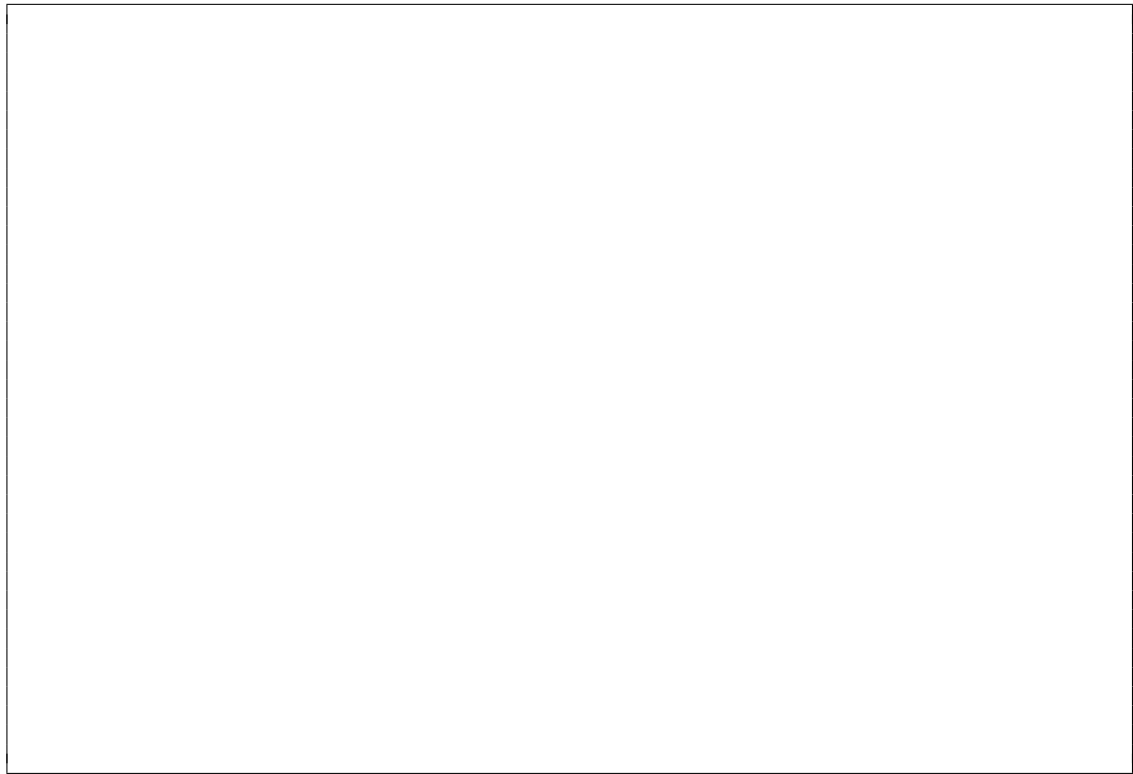
4. lengthens the lw stall

```

lw t0, 0(sp)       F D X X2 M WB
add t1, t0, t2     F D * * X X2 M WB
//stall 2 cycles for lw data

```

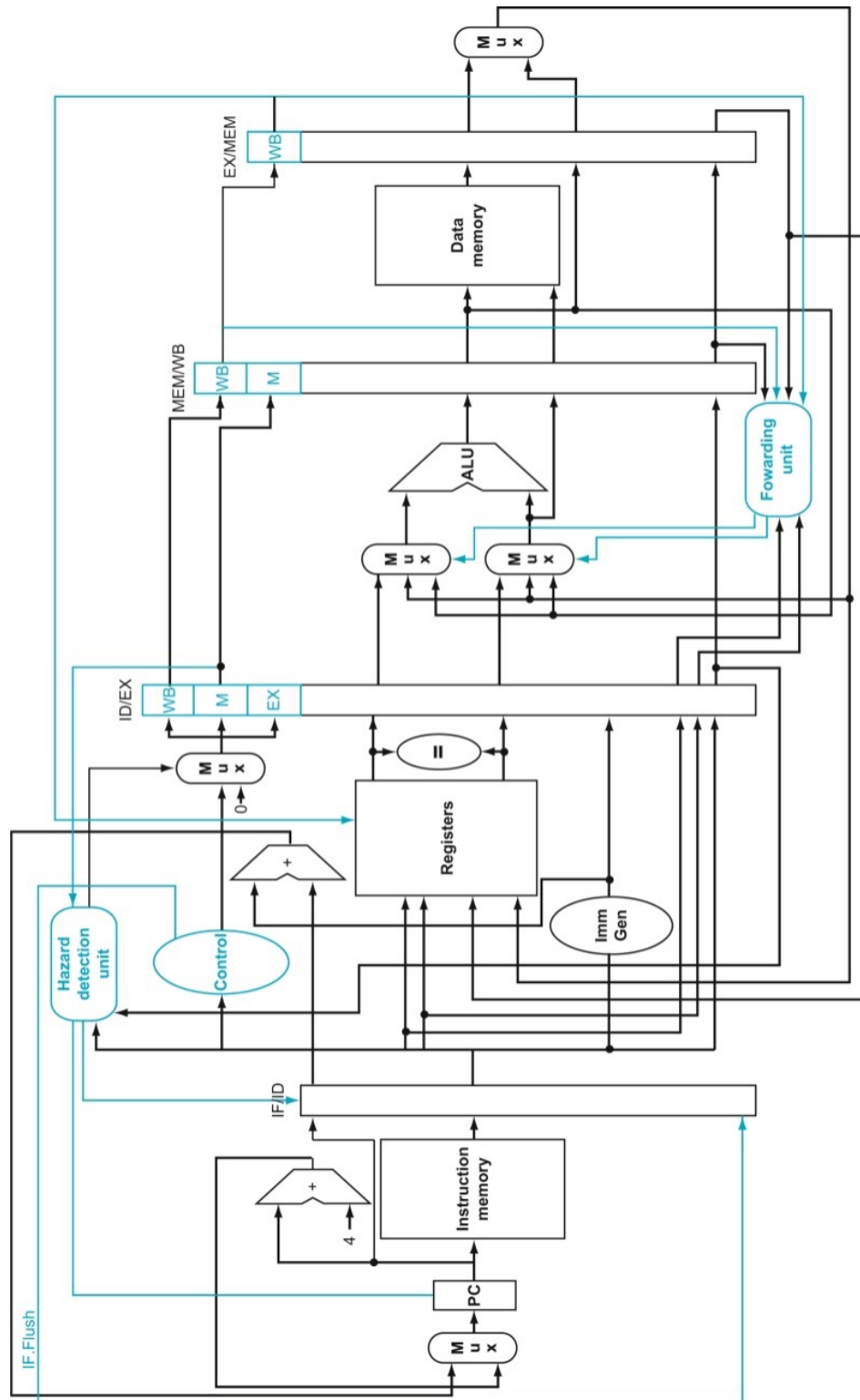
- (f) (10 points) Which approach will your team take, **Anika's** or **Kenji's**? Justify your answer with 3 strong, evidence-backed statements, these can be a positive about your chosen approach, or a negative about the alternative.

**Solution:**

1 point for explicitly picking a side

3 points per statement; 1 point for making a correct statement, 1 point for the statement that shows a contrast between the designs, 1 point for it being “strong” (actually a good and nuanced contrast).

Neither is right here, depends on the justification.





**Problem 2.** (25 points) Consider the following code running on a RISC-V processor with an advanced 5-stage pipeline that resolves hazards with forwarding wherever possible and stalls if necessary. This processor does not use a branch delay slot.

```
L: lw    x4, 4(x4)
    and  x4, x10, x4
    sw   x4, 4(x4)
    or   x4, x11, x4
    lw   x4, 4(x4)
* beq  x4, x4, L
    xor  x4, x4, x4
```

**Solution:**

5 forwards (lw→and, and→sw, and→or, or→lw(2), lw→beq) 0 write then read ()  
 3 stalls (lw→and, lw→beq (double)) 1 flush (xor)  
 7 instructions executed (one flushed)

Draw the pipeline diagram for one pass through the code plus the first instruction that is executed after the branch indicated with a \* is taken (i.e. the instruction at label L will be the last one in your diagram). Assume the \* branch is taken.

**Be sure to indicate any data forwarding, instruction flushes, and/or pipeline stalls that occur.** Remember: the final pipelined datapath from class has branch logic in the ID (decode) stage. This optimization also applies to jumps (e.g. jal, jalr). **Fit your diagram in the table below.**

instruction	pipeline stage														