CSM 61C Spring 2020

Pipelining and Hazards

Exam Question Compilation

This document is a PDF version of old exam questions by topic, ordered from least difficulty to greatest difficulty.

Questions:

- Fall 2019 Final Q7
- Summer 2018 Final Q5
- Summer 2018 Midterm 2 Q4
- Spring 2018 Midterm 2 Q4
- Fall 2017 Midterm 2 Q4

Q7) *RISCV Exam-isim Debug – Pipelined* (18 pts = 3 + 6 + 3 + 6)

After solving your datapath bug, you decide to introduce the traditional five-stage pipeline into your processor. You find that your unit tests with single commands work for all instructions, and write some test patterns with multiple instructions. After running the test suite, the following cases fail. You should assume registers are initialized to 0, the error condition is calculated in the fetch stage, and no forwarding is currently implemented.

Case 1: Assume the address of an array with all different values is stored in `s0`.

```
addi  t0 x0 1
slli  t1 t0 2
add   t1 s0 t1
lw    t2 0(t1)
```

Each time you run this test, there is the same incorrect output for `t2`. All the commands work individually on the single-stage pipeline.
Pro tip: you shouldn't even need to understand what the code does to answer this.

a) What caused the failure? (select ONE)	**b)** How could you fix it? (select all that apply)
◯ Control Hazard ◯ Structural Hazard ◯ Data Hazard ◯ None of the above	☐ Insert a nop 3 times if you detect this specific error condition ☐ Forward execute to write back if you detect this specific error condition ☐ Forward execute to memory if you detect this specific error condition ☐ Forward execute to execute if you detect this specific error condition ☐ Flush the pipeline if you detect this specific error condition

Case 2: *After fixing that hazard,* the following case fails:

```
      addi  s0 x0 4
      slli  t1 s0 2
      bge   s0 x0 greater
      xori  t1 t1 -1
      addi  t1 t1 1
greater:
      mul t0 t1 s0
```

When this test case is run, `t0` contains `0xFFFFFFC0`, which is not what it should have been.
Pro tip: you shouldn't even need to understand what the code does to answer this.

c) What caused the failure? (select ONE)	**d)** How could you fix it? (select all that apply)
◯ Control Hazard ◯ Structural Hazard ◯ Data Hazard ◯ None of the above	☐ Insert a nop 3 times if you detect this specific error condition ☐ Forward execute to write back if you detect this specific error condition ☐ Forward execute to memory if you detect this specific error condition ☐ Forward execute to execute if you detect this specific error condition ☐ Flush the pipeline if you detect this specific error condition

SID: _____

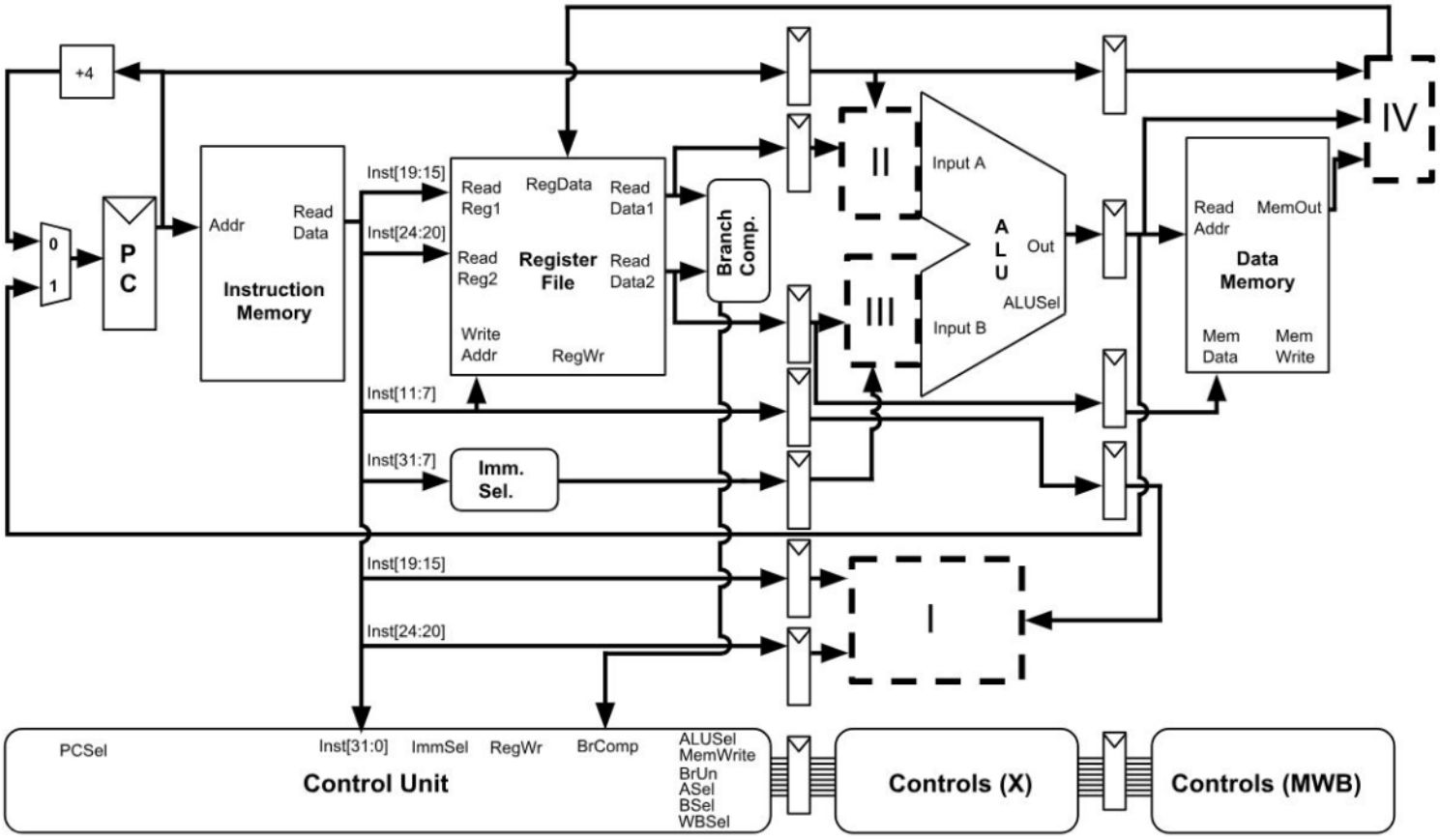
Question 5: DamonPath (8 pts)

Below we have implemented 3-stage CPU with the stages IFD, EX, and MWB. We're interested in implementing forwarding from the output of the MWB stage to the input of the ALU in the EX stage.This datapath should still implement the regular RISC-V instruction set as well as forwarding.

Make sure to read through all parts of the question (I, II, III, IV) as some variables may defined in different parts.

Assume that the control values generated by the control unit are driving their respective control inputs in the datapath.

(Note: In cases that may be ambiguous, we have marked certain values with the stage that they come from. For example, if we write ASel(X) this means the ASel coming from Controls(X) unit.)

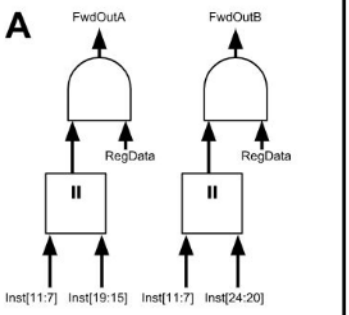


SID: _____

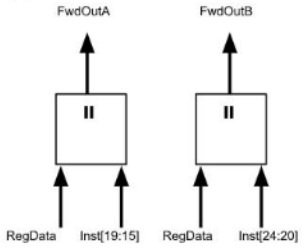
Choose the correct implementation for I - IV from the options below:

I.

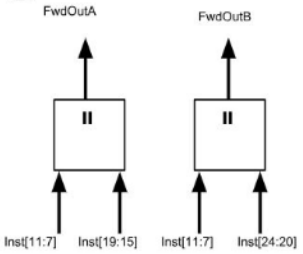
A



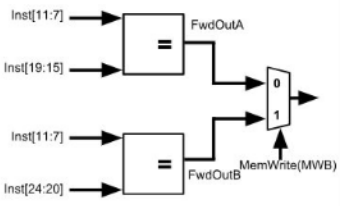
B



C

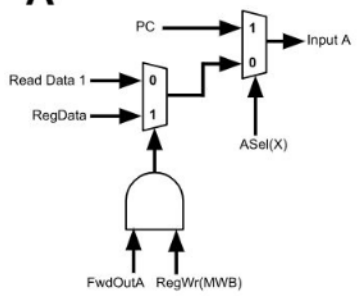


D

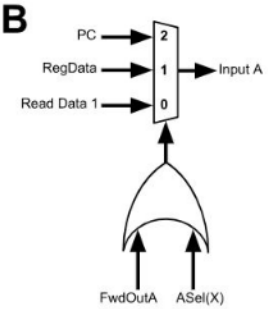


II.

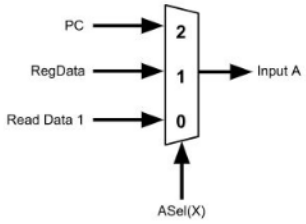
A



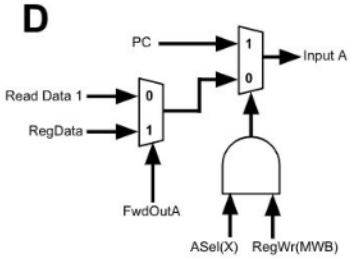
B



C

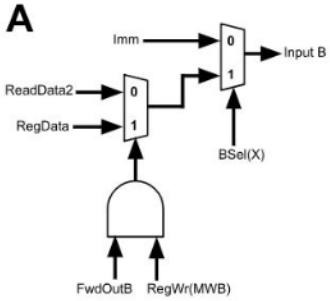


D

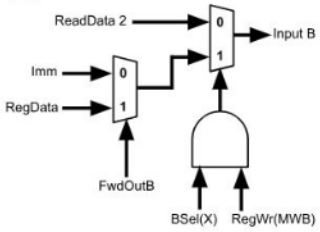


III.

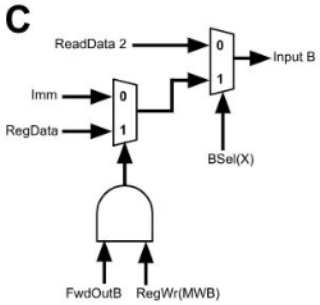
A



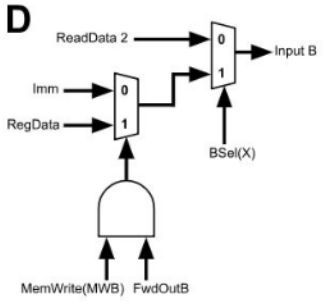
B



C

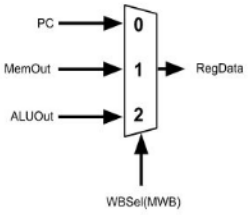


D

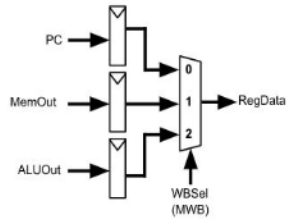


IV.

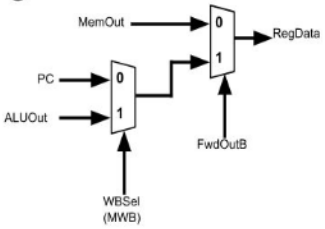
A



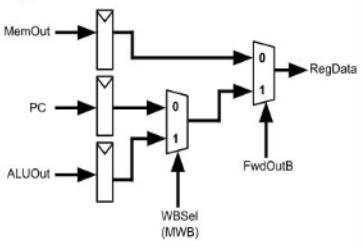
B



C



D



SID: _____

Question 4: RISC IS HAZARDOUS (22 pts)

Consider a RISC-V pipeline in 3 stages with the following specification:

Stage 1	Stage 2	Stage 3
Instruction Fetch/Instruction Decode (IFD)	Execute (EX)	Memory/Write Back (MWB)

It also has the following details:

- No Forwarding
- No reading and writing to the same register in the same cycle
- Reading and writing to the same register is considered a data hazard

Consider the following piece of code

```
1.        add t1 x0 x0
2.        add t2 x0 x0
3.        addi a0 x0 2
4.        slli a0 a0 2
5.  L2:   bge t1 a0 End
6.        add t3 sp t1
7.        lw t3 0(t3)
8.        add t2 t2 t3
9.        addi t1 t1 4
10.       j L2
    .
    .
    .
End:
```

1) Assume that instead of branch prediction, the CPU always stalls to resolve a control hazard. How many stall(s) are necessary for each control hazard **each time** it is encountered? You may not need all boxes.

Line Number	# Stalls/Encounter

SID: _____

2)

a) Where are the data hazards that produce stall(s)? You should describe each hazard as a tuple, (A, B), where instruction A is the the instruction that triggered a data hazard in instruction B. Place A's line number in the left box and B's line number in the right box You may not need all boxes.

Line of Instruction that first accesses the data	Line of instruction that must be stalled

b) How many **total cycles** will it take to complete the code above? Assume the pipeline is cleared upon reaching End. In addition, assume that there is perfectly accurate branch and jump prediction in the IFD stage. Thus, **ONLY CONSIDER STALLS DUE TO DATA HAZARDS. ASSUME THERE ARE NO STALLS FOR STRUCTURAL OR CONTROL HAZARDS.** Remember to **calculate the total number of cycles for fully executing the code**. A workspace was provided with this exam for you to show your work, which will only be graded if your final answer is not correct.

Total Cycles: _____

SID: _____

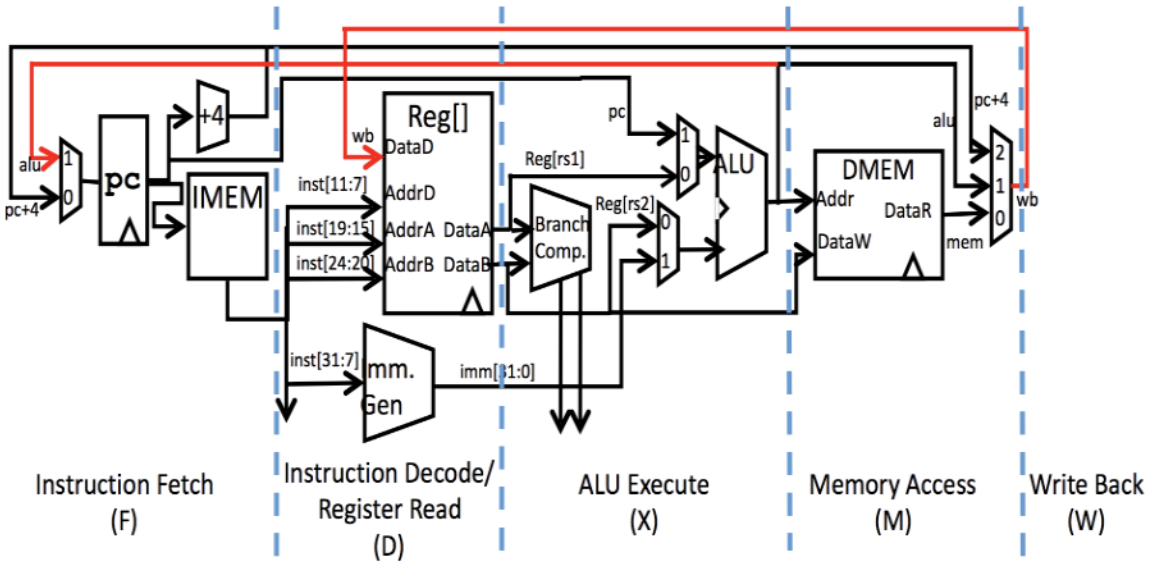
Now assume that our pipeline has full forwarding along with perfect branch/jump prediction done in the IFD stage.

3) What are the data hazards that remain? How many cycles must be stalled **each time** this hazard is encountered? If there are two instructions with the same data hazard and both are resolved by the first stall then only list the first instruction as needing to be stalled. The left box(es) should be a subset of the value(s) you had in the rightmost box(es) in 2a. You may not need all boxes.

Line of instruction that must be stalled	# Stalls/Encounter

Problem 4 *More or Less* **(20 points)**

Consider a typical 5-stage (Fetch, Decode, EXecute, Memory, WriteBack) pipeline. Assume pipeline registers exist where the dotted lines are.



For this question, consider the following parameters:

- Forwarding is not implemented
- The branch predictor always predicts the branch is not taken. Flush the pipeline if prediction is wrong.
- We cannot read and write from the same registers or memory address in the same clock cycle.
- No other optimizations are implemented in this datapath.

(a) Fill in the corresponding pipeline stages for the code sequence below for the 5-stage pipeline. The first instruction is done for you. If you need to stall a cycle, write "*" in that cycle.

```
begin:
        ori s1, x0, 0xF
        andi s2, x0, 0
        beq s1, s2, exit
        lw s1, 0xc(s0)
        xor s1, s1, s2
exit:
        lw s1, 0xc(s0)
```

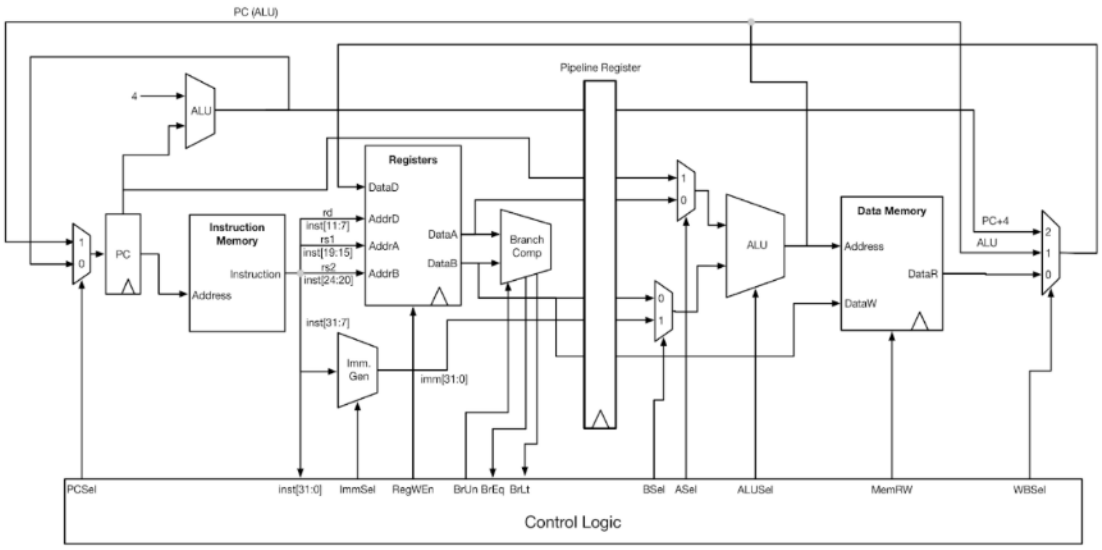
Instructions	Cycles																			
	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	C19	C20
ori s1 x0 0xf	F	D	E	M	W															
andi s2 x0 0																				
beq s1 s2 exit																				
lw s1 0xc(s0)																				
xor s1 s1 s2																				
lw s1 0xc(s0)																				

(b) Assume the maximum delays through each stage are: `F: 200ns, D: 150ns, EX: 100ns, MEM: 300ns, WB: 250ns`.

Assume the delays for the pipeline registers are factored into the pipeline stage delays. What is the latency and best case throughput of this 5-stage pipelined CPU? You may leave your answers as fractions. Don't forget the units!

Latency:_____ Throughput: _____

You decide to combine certain stages to make a two-stage pipeline by removing certain registers. The two-stage pipelined datapath looks like the following:



Again, consider the following parameters:

- Forwarding is not implemented
- The branch predictor always predicts the branch is not taken. Flush the pipeline if prediction is wrong.
- We cannot read and write from the same registers or memory address in the same clock cycle.
- No other optimization is implemented on this datapath.

(c) Fill in the corresponding pipeline stages for the same code sequence for the 2-stage pipelined CPU. The code sequence is reproduced below. Use "A" to denote stage 1, "B" for stage 2. The first instruction is done for you. If you need to stall a cycle, write "*" in that cycle.

```
begin:
        ori s1, x0, 0xF
        andi s2, x0, 0
        beq s1, s2, exit
        lw s1, 0xc(s0)
        xor s1, s1, s2
exit:
        lw s1, 0xc(s0)
```

Instructions	Cycles													
	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
ori s1 x0 0xf	A	B												
andi s2 x0 0														
beq s1 s2 exit														
lw s1 0xc(s0)														
xor s1 s1 s2														
lw s1 0xc(s0)														

(d) Suppose we want to execute three instructions on this two-stage pipelined CPU. The first instruction begins executing at the start of Cycle C0, the second begins at the start of Cycle C1, and the third, C2.
Fill in the correct control signals on each clock cycle in order to execute these instructions correctly:

- Any signals set by earlier instructions (before the first) should be set to "E".
- Any signals set by later instructions (after the third) should be set to "L".
- Indicate `Enable` or `Disable` for write enable signals.
- ImmSel should be set as `I, S, SB, U` or `UJ`.
- All other signals should be set as 0, 1, an ALU operation, or X (doesn't matter).
- The list of available ALU operations are `ADD, AND, OR, SRL, SRL, SLT`.

You may assume that there are no structural or data hazards.

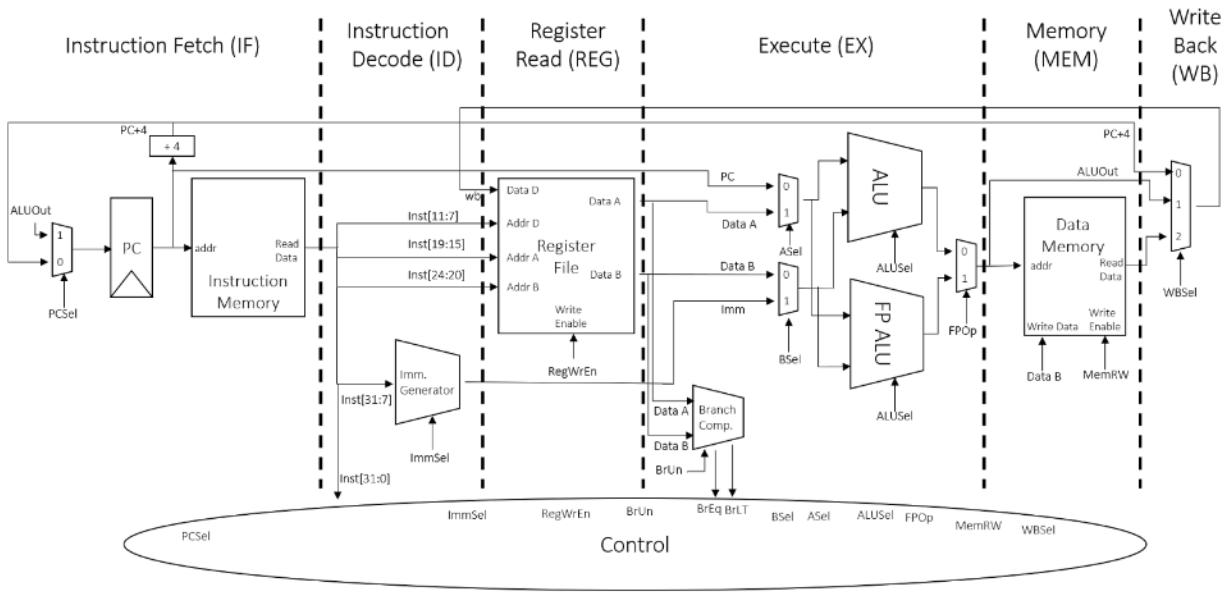
```
Program:
    srl t1, t2, t3
    sw t0, 4(a0)
    bltu s0, t2, 44
```

Cycle	Signals								
	PCSel	ImmSel	RegWEn	BrUn	BSel	ASel	ALUSel	MemRW	WBSel
C0									
C1									
C2									
C3									

Q4: Danger: Hazardous Material

In this question, you will be working with a modified RISC-V CPU. As opposed to the traditional 5-stage pipeline, this altered CPU has split the second phase into two distinct stages: instruction decode and register read. Furthermore, this CPU has a floating-point ALU (FP ALU) along with a traditional ALU (no floating point Register File is added). The added control signal, FPOp, determines which ALU to use for a given instruction. This floating-point ALU works by interpreting its 32-bit operands in IEEE 754 floating-point format. Unlike in standard RISC-V (and on the green sheet), assume floating point operations use the same registers as normal, non-floating point operations.

A diagram of the modified CPU and its corresponding stages are shown below. To simplify the diagram, any label at the very beginning of a wire acts like a tunnel in Logism. Along with the diagram above, you are given the following delays incurred by the circuit elements in the table below, as well as the delays incurred for some of the datapath stages.



Element	Register CLK-to-Q	Register Setup	MUX	ALU	FP ALU	Mem Read	Mem Write	RegFile Read	RegFile Setup	Imm. Gen.
Delay (ps)	20	25	10	150	215	225	230	100	35	75

a. Consider the floating point instruction, `faddi`, that is similar to `addi` except it considers its operands in floating-point format and executes a floating-point add operation. Assuming that this CPU is **NOT Pipelined** (i.e. it is a single-cycle CPU), what is the shortest clock period possible to execute the instruction `faddi t0, s0, 2.71` corectly? Write your answer as a single, simplified number (no summations or other expressions) on your answer sheet.

b. What is the shortest possible clock period at which we can run this pipelined CPU? Write your answer as a single, simplified number (no summations or other expressions) on your answer sheet.

c. Keeping the modified pipeline in mind, consider the program below with the following assumptions:
   - **There are no pipelining optimizations** (no forwarding, load delay slot, branch prediction, pipeline flushing, etc.)...
   - **We cannot read and write from registers in the same clock cycle.**
   - **An integer 100 is stored at memory address 0x61C61C61, and that R[a0] = 0x61C61C61**.
   - A hazard between two instructions should be counted as only 1 hazard.

```
lw t0, 0(a0)          # R[a0] = 0x61C61C61
srli s0, t0, 4
faddi s1, t0, 1.7
beq a0, s1, Label
add a1, t2, t3
Label ...
```

Write your answers to the questions below on your answer sheet.

i. How many cycles would the program take to execute correctly on the pipelined machine?

ii. How many stalls would need to be added for the program to be executed correctly on the pipelined machine?

iii. How many data hazards are present in the program?

iv. How many control hazards are present in the program?

d. For both parts below, reorder the instructions to minimize the number of cycles needed to execute the program, then answer the questions below. Your reordered instructions should produce the same results for all involved registers as the original instructions do.

i. How many cycles does the program take to execute with reordered instructions?

ii. How many stalls are required?