CS61C Fall 2026

RISC-V Pipelining, Hazards

Discussion 9

1 Datapath Review

1.1 True or False? The single cycle datapath uses the outputs of all hardware units for each instruction.

1.2 True or False? It is possible to execute the stages of the single-cycle datapath in parallel to speed up execution of a single instruction.

1.3 Which instruction(s) are responsible for the critical path?

1.4 Why is the single-cycle datapath inefficient?

2 Performance Analysis

Register clk-to-q 30 ps	**Branch comp.** 75 ps	**DMEM write setup** 200 ps
Register setup 20 ps	**ALU** 200 ps	**Memory read** 250 ps
Register hold 10 ps	**Imm. Gen.** 15 ps	**Mux** 25 ps
RegFile read 100 ps	**RegFile setup** 20 ps	

Copied above are the same sample delays and setup times for each of the datapath components and registers. Use these, in conjunction with the pipelined datapath on the last page, to answer the questions below.

2.1 What would be the fastest possible clock time for a **single cycle** datapath? Recall that load instructions exercise the critical path.

HINT: $t_{\text{clk-cycle}} \geq t_{\text{clk-to-q}} + t_{\text{longest-combinational-path}} + t_{\text{setup}}$

2.2 What is the fastest possible clock time for a pipelined datapath?

HINT: First identify the critical path delay (longest path between two registers) for each stage.

2.3 At steady state (i.e. ignore the first few cycles), how many instructions finish every cycle in the single-cycle datapath? What about the pipelined datapath, assuming no hazards occur?

2.4 What is the fastest possible clock frequency of the single-cycle datapath compared to that of the pipelined datapath? Based on this, how do their instruction **throughputs** differ, assuming no hazards occur?

2.5 What is the speedup from the single cycle datapath to the pipelined datapath? Why is the speedup less than 5x?

3 Solving Data Hazards

One of the costs of pipelining is that it introduces pipeline hazards. Hazards, generally, are issues with something in the CPU's instruction pipeline that could cause the next instruction to execute incorrectly. Recall that **data hazards** are caused by data dependencies between instructions. In CS 61C, where we always assume that instructions go through the processor in order, we see data hazards when an instruction reads a register before a previous instruction has finished writing to that register.

For all questions, assume **no branch prediction** or **write-then-read** in one cycle for the RegFile.

Forwarding

Most data hazards can be resolved by forwarding, which is when the result of the EX or MEM stage is sent to the EX stage for a following instruction to use.

Note: There are two ways of forwarding - **MEM to EX** and **WB to EX**. How are each of these implemented in hardware? We add 2 wires: one from the beginning of the MEM stage for the output of the ALU (right after the EX/MEM pipelined register) and one from the beginning of the WB stage (right after the MEM/WB pipelined register). Both of these wires will connect to the A/B muxes in the EX stage.

3.1 Look for data hazards in the code below, and figure out how forwarding could be used to solve them.

Instruction	C1	C2	C3	C4	C5	C6	C7
`1. addi t0, a0, -1`	IF	ID	EX	MEM	WB		
`2. and s2, t0, a0`		IF	ID	EX	MEM	WB	
`3. sltiu a0, t0, 5`			IF	ID	EX	MEM	WB

3.2 Imagine you are a hardware designer working on a CPU's forwarding control logic. How many instructions after the `addi` instruction could be affected by data hazards created by this `addi` instruction?

Stalls

3.3 Identify the data hazards in the code below. One of them cannot be solved with forwarding—why? What can we do to solve this hazard?

Instruction	C1	C2	C3	C4	C5	C6	C7	C8
`1. addi s0, s0, 1`	IF	ID	EX	MEM	WB			
`2. addi t0, t0, 4`		IF	ID	EX	MEM	WB		
`3. lw t1, 0(t0)`			IF	ID	EX	MEM	WB	
`4. add t2, t1, x0`				IF	ID	EX	MEM	WB

3.4 Say you are the compiler and can re-order instructions to minimize data hazards while guaranteeing the same output. How can you fix the code above?

Control Hazards

Control hazards are caused by jump and branch instructions, because for all jumps and some branches, the next PC is not PC + 4, but the result of the ALU available after the EX stage. We could stall the pipeline for control hazards, but this decreases performance.

3.5 Identify the control hazards in the code below. How can we resolve them?

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9
`1. beq s0, s1, loop`	IF	ID	EX	MEM	WB				
`2. addi t0, t0, 4`		IF	ID	EX	MEM	WB			
`3. ori t1, t1, 7`			IF	ID	EX	MEM	WB		
`4. slli sp, sp, 2`				IF	ID	EX	MEM	WB	
`5. addi a0, t0 2`					IF	ID	EX	MEM	WB

3.6 Besides stalling, what can we do to resolve control hazards?

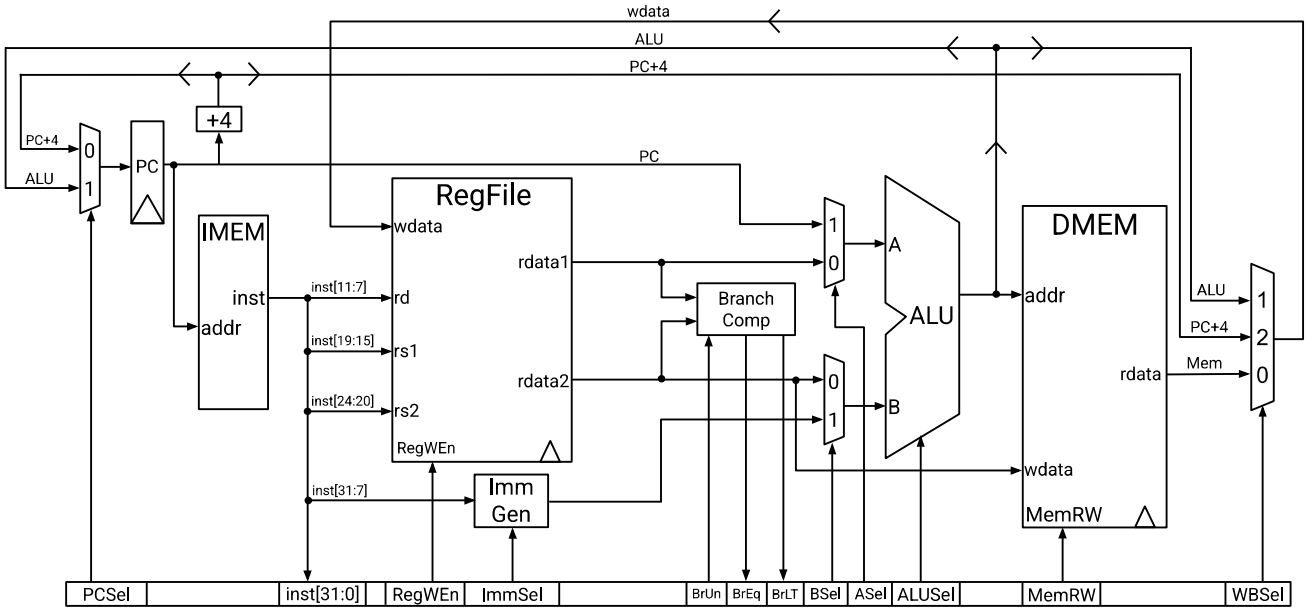
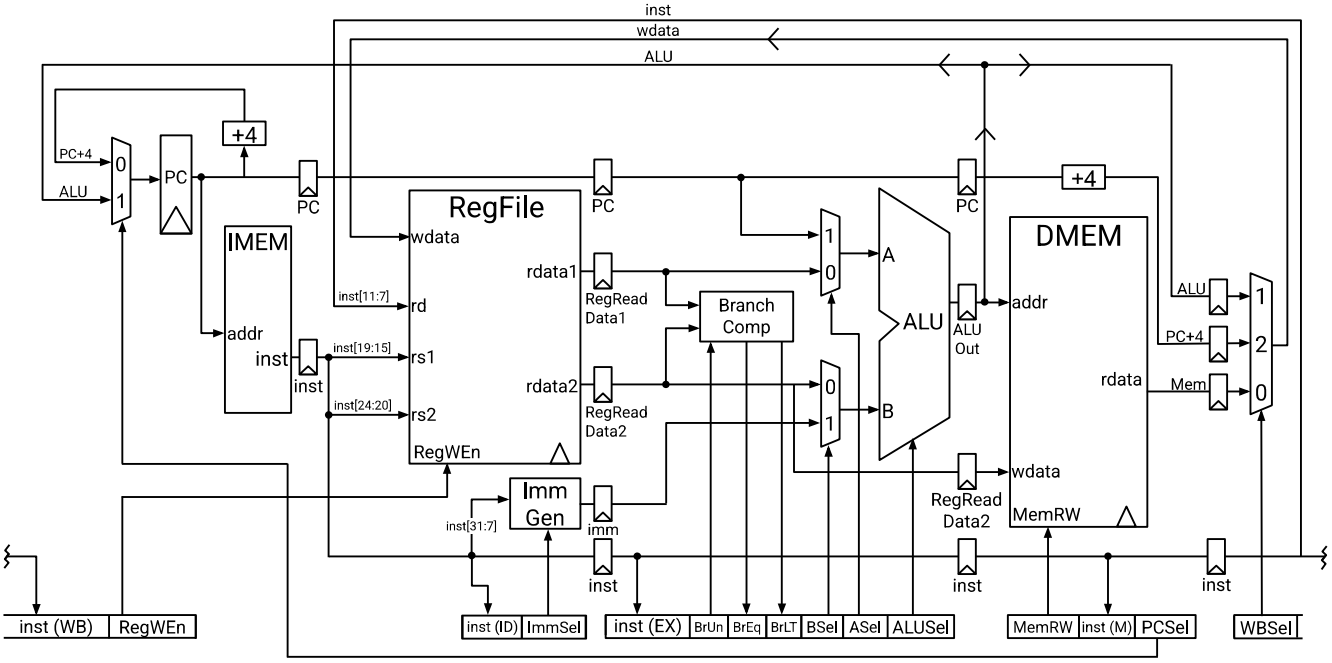
4 Hazards Practice

Given the RISC-V code below and a 5-stage pipelined CPU with no forwarding, how many hazards would there be? What types are each hazard? Consider all possible hazards between all instructions.

How many stalls would there need to be in order to fix the data hazard(s) if the RegFile supports write-then-read? What about the control hazard(s) if we use branch prediction with perfect accuracy? There is no forwarding in this question.

Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9
1. `sub t1, s0, s1`	IF	ID	EX	MEM	WB				
2. `or s0, t0, t1`		IF	ID	EX	MEM	WB			
3. `sw s1, 100(s0)`			IF	ID	EX	MEM	WB		
4. `bgeu s0, s2, loop`				IF	ID	EX	MEM	WB	
5. `add t2, x0, x0`					IF	ID	EX	MEM	WB

Single-Cycle Datapath Diagram



5-Stage Datapath Diagram