ECE 4750 Computer Architecture
Example Pipeline Diagram

revision: 2025-09-18-17-05

Problem 1. Resolving Data Hazards

In this problem, you will identify architectural data dependences, and then resolve data hazards through (1) only stalling and (2) a combination of bypassing and stalling.

Part 1.A Architectural Data Dependencies

Consider the following sequence of TinyRV1 instructions. Draw arrows to indicate the architectural RAW dependencies between instructions.

```
addi   x1,    x2,    1

lw     x3,    0(x1)

lw     x4,    0(x3)

add    x5,    x3,    x1
```

Part 1.B Pipeline Diagram Illustrating Stalling

Draw a pipeline diagram that illustrates how these four instructions would execute on a pipelined TinyRV1 processor that uses only stalling to resolve data hazards. You should assume the processor includes absolutely no bypassing. Add arrows to your pipeline diagram to indicate all microarchitectural RAW dependencies. For a processor without bypassing, the arrows will be exclusively those that result in passing values through registers. Ensure that your diagram does not have any data hazards.

`addi x1, x2, 1`																		
`lw x3, 0(x1)`																		
`lw x4, 0(x3)`																		
`add x5, x3, x1`																		

Part 1.C Pipeline Diagram Illustrating Bypassing and Stalling

Draw a pipeline diagram that illustrates how these four instructions would execute on a pipelined TinyRV1 processor that uses both stalling and full bypassing to resolve data hazards. Add arrows to your pipeline diagram to indicate all microarchitectural RAW dependencies (those that result in passing values through registers **and** those that result in using the bypass network). Ensure that your diagram does not have any data hazards.

`addi x1, x2, 1`																		
`lw x3, 0(x1)`																		
`lw x4, 0(x3)`																		
`add x5, x3, x1`																		

Note: You should be able to draw the pipeline diagram for any microarchitecture we can reasonably invent on the fly. What if the processor has only X to D bypassing but not M to D or W to D bypassing? What would that look like? You can create new problems for yourself by executing the same instruction sequence on a wide variety of real or made-up processor designs.

Problem 2. Pipeline diagram for simple assembly sequence

[From T02 Lecture notes page 46]

First, draw arrows over the instruction sequence on the left, indicating the architectural RAW dependencies between instructions.

Next, draw a pipeline diagram on the right, illustrating how the following assembly sequence would execute on a fully bypassed pipelined TinyRV1 processor. Include microarchitectural dependency arrows over the pipeline stages to illustrate how data is transferred via registers and along various bypass paths.

`lw x1, 0(x2)`																		
`lw x3, 0(x4)`																		
`add x5, x1, x3`																		
`sw x5, 0(x6)`																		
`addi x2, x2, 4`																		
`addi x4, x4, 4`																		
`addi x6, x6, 4`																		
`addi x7, x7, -1`																		
`bne x7, x0, loop`																		