CS 61C Summer 2024

RISC-V Pipelining

Discussion 8

1 Pre-Check

This section is designed as a conceptual check for you to determine if you conceptually understand and have any misconceptions about this topic. Please answer true/false to the following questions, and include an explanation:

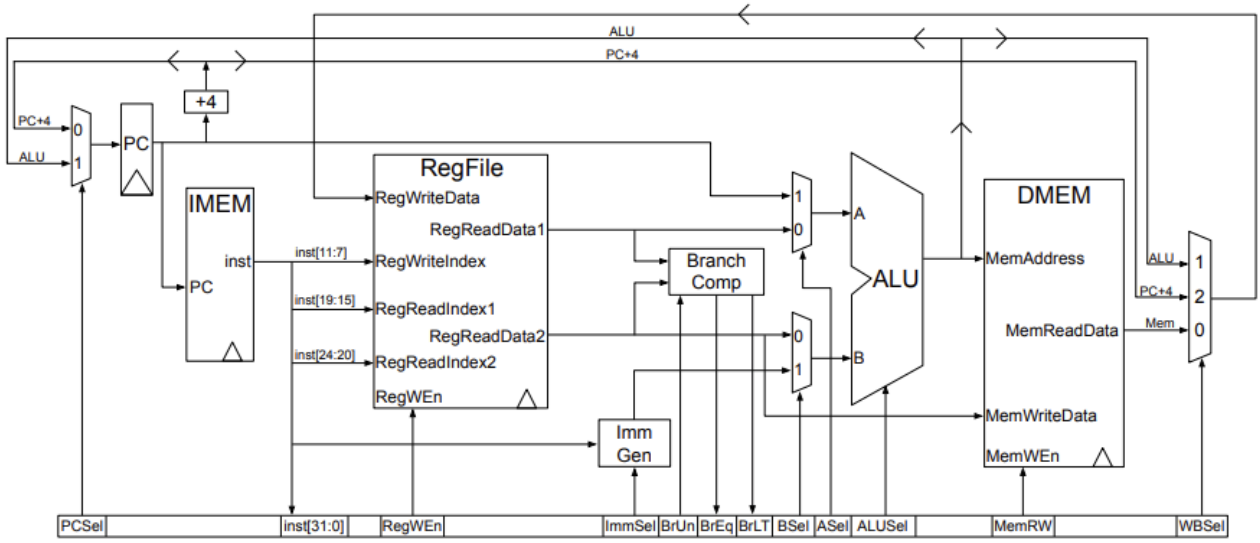
1.1 By pipelining the CPU datapath, each single instruction will execute faster (latency is reduced), resulting in a speed-up in performance.

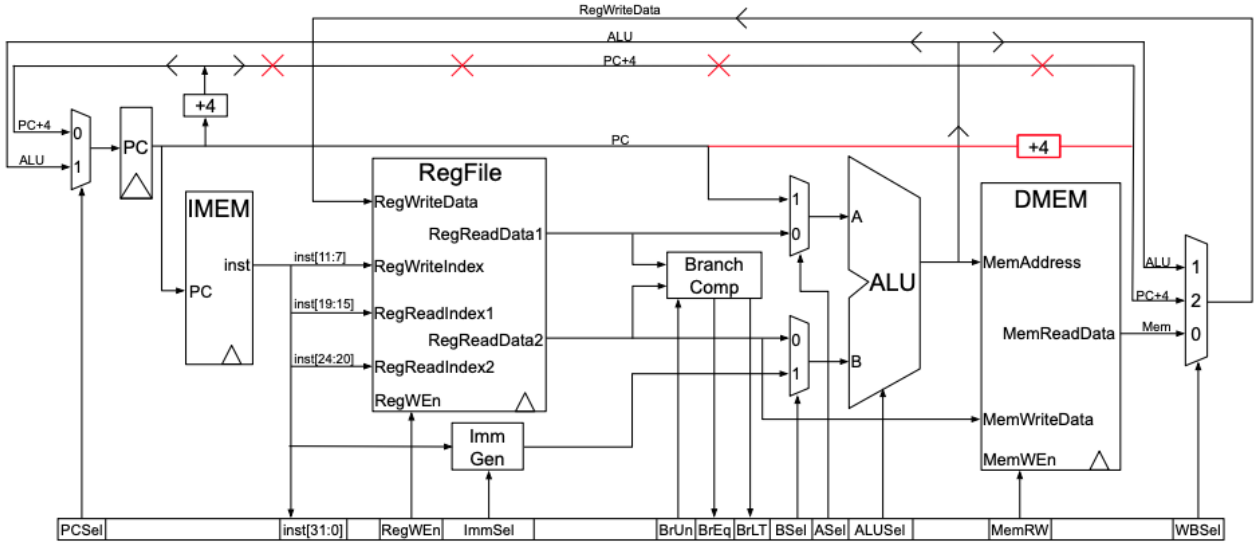
1.2 A pipelined CPU datapath results in instructions being executed with higher throughput (than the single-cycle CPU).

2 Pipelining Registers

Recall the five stages: In the **IF** stage, we use the Program Counter to access our instruction as it is stored in IMEM. Then, we separate the distinct parts we need from the instruction bits in the **ID** stage and generate our immediate, the register values from the RegFile, and other control signals. Afterwards, using these values and signals, we complete the necessary ALU operations in the **EX** stage. Next, anything we do in regards with DMEM (not to be confused with RegFile or IMEM) is done in the **MEM** stage, before we hit the **WB** stage, where we write the computed value that we want back into the return register in the RegFile.

In order to pipeline, we separate the datapath into 5 discrete stages. These 5 stages, divided by registers, allow operation of different stages of the datapath in the same clock period. Different instructions can use different stages at a time. At each clock cycle, the necessary inputs into a particular stage are sampled at the rising clock edge (and available after the clk-to-q delay). After the stage operates on the inputs, the corresponding outputs are fed into pipeline registers for the next stage. Note, pipeline registers may also be required to pass information that may not be necessary for the next immediate stage, but some future stage.





2.1 Two diagrams are provided above. The topmost one is the standard single cycle datapath. The second is a modified version. Compare these two diagrams and explain the difference.

2.2 Using the modified single-cycle datapath as reference provided above, think about the information that needs to be passed along from stage to stage. Which pipeline registers are required at the end of each stage?

IF to ID:

ID to EX:

EX to MEM:

MEM to WB:

2.3 Looking at the way PC is passed through the datapath, there are two places where +4 is added to the PC, once in the **IF** and **MEM** stage. Why do we add +4 to the PC again in the memory stage?

3 Performance Analysis

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

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

3.1 What would be the fastest possible clock time for a single cycle datapath? Recall from last week's discussion that one instruction which exercises the critical path is `lw`.
(HINT: $t_{\text{clk-cycle}} \geq t_{\text{clk-to-q}} + t_{\text{longest-combinational-path}} + t_{\text{setup}}$)

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

3.3 What is the speedup from the single cycle datapath to the pipelined datapath? Why is the speedup less than 5×?