CS 520: Computer Architecture and Organization
Fall 2015: Prof. Timothy N. Miller
Homework 3
Deadline: Monday October 19

Problem 1

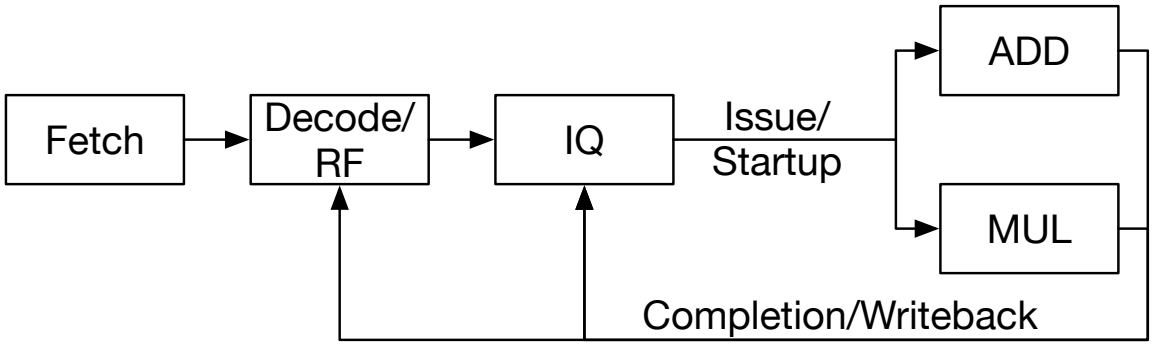

Figure 1

Definitions:
Startup – when an instruction is send from the issue queue into execution.
Completion – when an instruction's results are available for forwarding to other instructions and writeback to the register file.

For subproblems below, refer to Figure 1. In this simplified architecture, we stipulate the following:

- Instructions with valid operands spend at least one cycle in the IQ.
- For a single cycle instruction (ADD), execution occurs on one cycle, and write back will occur on the following cycle.
- If an instruction's dependencies become resolved by writeback, that instruction will be issued (start executing) immediately. This means that if you have two consecutive single-cycle instructions, and there is a dependency, they will execute on consecutive cycles.
- We only support ADD and MUL instructions. Adds take have **one** cycle of latency, while multiplies take **four**.

Problem 1.1

In class, we discussed the following fact: If an out-of-order (OOO) processor has a centralized issue queue (IQ), then there must be multiple paths from the IQ to functional units. Otherwise, instruction startup will be serialized.

What does it mean to serialize instruction startup, and why might this have an effect on performance?

Problem 1.2

If an OOO processor has a single shared write-back bus, what might this do to the performance of instruction completion?

Problem 1.3

Refer to Figure 1 and make the following assumptions:

- Startup is serialized
- Any number of writebacks can occur simultaneously

Construct a sequence of five instructions (only ADD and MUL) such that execution would be at least two cycles shorter if startup were not serialized. Draw timelines to support your answer.

Problem 1.4

Refer to Figure 1 and make the following assumptions:

- Any number of startups can occur simultaneously
- Writebacks are serialized

Construct a sequence of five instructions (only ADD and MUL) such that execution would be at least two cycles shorter if writeback were not serialized. Draw timelines to support your answer.

Problem 2

Refer to Figure 1. Assume that any number of startups and writebacks can occur simultaneously. Construct a sequence of three instructions such that out-of-order completion would cause a false dependency to corrupt a flow dependency.

Problem 3

Problem 3.1

Draw a dataflow graph for the following instruction sequence. Be sure to include flow dependencies, antidependencies, and output dependencies through both registers and memory.

```
I1: LOAD R1, R2, #100      // R1 ← MEM[R2+100]
I2: ADDI R5, R1, #10       // R5 ← R1 + 10
I3: LOAD R3, R1, #10       // R3 ← MEM[R1+10]
I4: MULI R4, R2, #4        // R4 ← R2 * 4
I5: STORE R4, R5, #0       // MEM[R5+0] ← R4
I6: MUL R1, R3, R4         // R1 ← R3 * R4
I7: ADD R4, R2, R3         // R4 ← R2 + R3
I8: SUBI R4, R4, #3        // R4 ← R4 - 3
```

Problem 3.2

There is a false dependency between I7 and I8 that can be ignored. Why?

Problem 3.3

Draw a timeline for a 5-stage APEX pipeline that shows all instances of forwarding of register values within the pipeline.