Lab Assignment 3: Frequently Asked Questions

1. According to the handout, we are not supposed to add Traps to the IFQ. Should we count reading the Trap as our fetch for that cycle, or should we fetch until we find a non-trap instruction?

   You should fetch until you find a non-trap instruction for that cycle.

2. Is the Dispatch stage able to dispatch multiple instructions within the same cycle if there are more than one instruction in the IFQ and the reservation stations are available?

   No, only one instruction may be dispatched per cycle.

3. Why are you restricted to only dispatching the first instruction off the IFQ? If the first instruction requires an int RS (none are available) but the second instruction in the queue requires a float RS (and say it's available) why would the second instruction wait for the first instruction? Can't the second instruction be dispatched without waiting for the first?

   Good question. Here's an example of why we should only dispatch instructions at the head of the queue:

   ```
   l.s $f1, 0($sp)
   add.s $f3, $f1, $f3
   ```

   Let's say the load instruction is at the head, and there are no integer/memory reservation stations available. But let's say there's a floating-point reservation station available. If we were to dispatch the add instruction first, then it will get the wrong value for $f1 (since the load instruction hasn't updated the map table). As you can see, instructions must be dispatched in-order since dependences can exist even across different types of reservation stations.

4. How do we handle conditional and unconditional branches in dispatch?

   Treat them the same. So since we're assuming perfect branching (both target and direction), you don't have to handle RAW hazards for branches. Branch instructions occupy the IFQ but are never dispatched to reservation stations. Branches can be removed from the IFQ once they reach the head.

5. Just to clarify, is the cycle in which an instruction enters dispatch when that instruction enters the IFQ (since fetch and dispatch are merged)? And the cycle in which it enters issue is when the instruction enters the reservation station?

   Yes, an instruction enters Dispatch when it has made it into the IFQ (this is actually still true even if F and D weren't merged, because technically, if an instruction is in the IFQ, then the fetching part is done). And yes, an instruction enters Issue when it has made it into the reservation station (since now the dispatching part is done).

6. When is an instruction removed from the IFQ?

   The instruction is removed from the IFQ when it moves to the reservation station (Dispatch stage).

7. Are we allowed to change the data structure for Instruction Fetch Queue? We are given an array by default, but could we use something else?

   Yes you can. Just make sure that your changes are marked by ECE552 Assignment 3 - BEGIN CODE and ECE552 Assignment 3 - END CODE.

8. Can instructions move from dispatch to issue out of order? Or is it always done in order?

   Moving from the Dispatch stage to the Issue stage (i.e., moving an instruction from the fetch queue to the reservation station) must always be done in order. But instructions can move from Issue to Execute out of order.

9. How is register renaming implemented in the lab for int and FP registers?

   In Tomasulo, the map table renames registers by keeping track of register "versions" (using tags) so that instructions can get their operands from the CDB instead of from the register file.

10. Why does an instruction remain in Issue if there is no functional unit available?

    When an instruction enters the issue stage, it has not yet booked a functional unit; in the dispatch stage, it booked a reservation station. It then waits here until 1) its source values are available and 2) there is a functional unit available.

11. When there are multiple instructions in reservation stations (i.e., 3 or 4 integer reservations currently used), what is the priority order for sending these reservations to the functional units?

    You should prioritize older instructions, i.e., instructions that come first in program order. Always use instruction age when dealing with structural hazards.

12. When can a RAW dependent instruction begin Execute?

    An instruction may start Execute on the cycle immediately after receiving its source value from the CDB (assuming a functional unit is available). For example, if A is waiting for a value from B, if B is in Writeback on cycle 9, then A can enter Execute on cycle 10. This is different from the textbook, which says that the instruction must stay in Issue for one more cycle (so A would enter Execute on cycle 11 instead).

13. In the textbook, an instruction reserves the CDB during Issue. Is this true for the lab?

    Unlike the textbook, during Issue, an instruction does not need to reserve the CDB; it only needs to reserve the functional unit. Be sure to take this into account. For any other discrepancies, the lecture notes take precedence over the textbook.

14. Store instructions use the integer ALU, but do not need to write to CDB. Does this mean: 1) They still take the full 4 cycles in the integer FU; 2) When they finish execute, we can remove them from the FU regardless of whether another instruction is competing for the CDB.

    Yes, they still take 4 Execute cycles, and yes, they can be removed from the FU regardless of any CDB contention.

    Does that mean the `tom_cdb_cycle` will be 0 since it never goes to that stage? Yes. Also note that the cycles in the `instruction_t struct`, e.g., `tom_dispatch_cycle, tom_issue_cycle` store the cycle an instruction *enters* a given stage.

15. When do we deallocate reservation stations and functional units?

    Reservation stations and functional units should be deallocated upon completing Execute. The functional unit shouldn't be deallocated until the instruction wins access to the CDB. But technically speaking, an instruction finishes Execute not when its result is computed but rather when it's next in line to use the CDB. So if two instructions finish Execute on cycle 16, then only the older one gets to take the CDB and proceed to Writeback on cycle 17; the younger one must wait in Execute for another cycle (so it doesn't get to deallocate its reservation station and functional unit yet). An instruction waits in Execute upon a CDB structural hazard. Only when it gets the CDB does it release its RS and FU.

16. Since stores do not write to the CDB, does that imply that up 2 instructions can leave execute (one store and one instruction that enters CDB), and that of these two the store will complete a cycle earlier? Thank you very much!

    That's right, though it's not limited to just one store; multiple store instructions can complete Execute (and retire) simultaneously since none of them will contend for the CDB.

17. Can a value be used on the cycle it is on the CDB? If an instruction is waiting for a dependency to be resolved, and that dependency enters the CDB stage on a cycle, can that instruction proceed to execute that same cycle? Or does it have to wait one more cycle, and can execute the cycle after?

    In the assignment document it says that there is no forwarding support across the CDB, and such an instruction with RAW dependency must wait until the next cycle to read its value from the CDB and then be sent to a functional unit (if there is one).