Computer Design & Organization
Assignment 3
Due: Friday, October 27

This assignment is meant to test your understanding of two hardware instruction scheduling techniques (out-of-order execution with Tomasulo's algorithm and out-of-order execution with a reorder buffer), and to enable you to gauge how and why they have different effects on performance. The code you'll be scheduling is the well-known SAXPY loop used in linear algebra codes.

This project should be done in groups of **two**. As usual, this should be done with different partners than before.

The assignment is to do **Problem 4.14 (f)**, and **(j)**. If you divide up the work, each of you taking a different part of the problem, be sure to explain to your partner what you did and what you learned, because you both will be responsible for the material.

You'll need to make some assumptions about the various pipelines. Below are some we came up with; please use them so that you are all working under the same set of assumptions. If you want to modify them, please let us know **now**. If you need more assumptions, indicate them clearly on your homework. If the problem in the book differs from what is said in these pages, follow the directions in these pages.

For each part, we have suggested tables that will help you think about the problem and organize your answers. For a few of the tables, the first row has been filled in for you, just to get you going.

Assumptions for both (4.14 (f, j))

Assume the processor of Figure 4.8 (p. 253) (and ignore the statement in the book which says only 1 FP unit). Assume that the load buffer now has a load unit that will support its 6 entries, and likewise the store buffer has a store unit that can support its 3 entries. As stated above, these units have an EX time of 1 cycle for the load and store operations. The EX stage of loads and stores does both the effective address computation and the memory access. None of the functional units are pipelined.

In addition to the units shown in Figure 4.8, there is an integer unit with an unlimited number of reservation stations that has an EX time of 1 cycle for all integer operations, including branches.

The instruction queue has an infinite number of entries.

If there is a conflict to access the CDB, priority is given in this order: FP multiply, FP add, FP load, any integer operation, FP store. The FP add, FP multiply and the integer unit are considered busy if they cannot broadcast their result.

Assume that a FP multiply takes 6 cycles and a FP add takes 4 All other instructions take 1 cycle.

Whenever you need to refer to cycle numbers, label cycle #1 as the first interesting event that happens. For instance, in part (f) we ask you to begin your schedule with the cycle in which an instruction issues; label the first instruction as issuing in cycle 1.

A result broadcast at cycle *i* is read (snarfed) during the same cycle, i.e., an instruction dependent only on this result can start executing at cycle *i+1*.

Remember that there are both integer and FP register files.

Assume that the variable Done can be accessed as though it was in an integer register, although we won't show the register.

Floating point numbers are 8 bytes.

The Tomasulo processor (4.14 (f))

Show the cycles timings from when the first instruction of the first iteration issues until the fourth instruction from the second iteration completes. Express your execution profile as a table of 6 columns, where the columns are the instruction, its functional unit and reservation station number, the cycle in which it issues, executes and writes a result, and a comment field that explains why there was a stall, if any. Do not reuse functional units.

Instruction	Unit & Reservation Station	Cycle Number in:			Comment
		IS	EX	WR	
ld f2,0(r1)	Load - 1	1	2	3	
...					

Table 1: Execution profile of SAXPY on a DLX processor with Tomasulo's algorithm

Then show the Tomasulo hardware data structures at the point after which the fourth instruction in the second iteration issues (but does not execute). This can be expressed in the instruction status table, the reservation station table and the register table. Templates for all three appear below. Although the reservation station table normally contains entries for each functional unit, yours should only contain entries for busy functional units. Although the register status table contains entries for each register, yours should only contain entries for registers waiting for either results or stores still pending at this point. A tag should appear without "()", e.g., FP Mult - 1, and a value that comes from a functional unit should appear within "()", e.g., (FP Mult - 1).

Instruction	IS	EX	WR	Comment
ld f2,0(r1)	√	√	√	
...				

Table 2: Instruction status of SAXPY on a DLX processor with Tomasulo's algorithm

Functional unit	Reservation Station Status				
	Busy	V_j	V_k	Q_j	Q_k
...					

Field	Register		
	f#	...	f#
...			

Table 3: Register status after cycle 13 of SAXPY on a DLX processor with Tomasulo's algorithm.

The Tomasulo processor with speculative execution and in-order completion (4.14(j))

Show the same timings and status tables as in the last problem, but for the Tomasulo processor with in-order completion. In addition, construct a reorder buffer, a reservation station, and a register status table that reflects execution state at the same stop point (the fourth instruction in the second iteration has issued but has not executed). Although the reservation station table normally contains entire for each functional unit, yours should only contain entries for busy functional units. The register status table should only contain entries for **all** registers. A tag should appear without "()", e.g., FP Mult - 1, and a value that comes from a functional unit should appear within "()", e.g., (FP Mult - 1).Speculative execution should not enter in here.

Suggested tables are:

Instruction	Unit & Reservation Station	IS	EX	WR	Commit	Comment
ld f2,0(r1)	load - 1	1	2	3	4	
...						

Table 4: Execution profile of SAXPY on a DLX processor with Tomasulo's algorithm, speculative execution and in-order completion

Entry #	Instruction	State	Value
1			
...			

Table 5: Reorder buffer after cycle 13 of SAXPY on a DLX processor with Tomasulo's algorithm, speculative execution and in-order completion

Functional unit	Reservation Station Status					
	Busy	V_j	V_k	Q_j	Q_k	Dest
...						...

Table 6: Reservation station status of SAXPY on a DLX processor with Tomasulo's algorithm, speculative execution and in-order completion

Register	r#	...	r#	FP r#	...	FP r#
Reorder #						

Table 7: Register status of SAXPY on a DLX processor with Tomasulo's algorithm, speculative execution and in-order completion

The value field in the reorder buffer doesn't have to be filled in; it is a placeholder, just so you are aware that the value is stored there.

Determine and explain the speedups of (j) over (f). Is in-order completion worth the cost in hardware?