CSE 471 Autumn 2001

Computer Design and Organization
Assignment #3

Due: Friday November 2

The purpose of this third assignment is to test your understanding of Tomasulo's algorithm and of its extension using a reorder buffer. This paper and pencil assignment should be done in groups of two. As usual, this should be done with different partners than before.

The code that you'll be scheduling is the well-known SAXPY loop used in linear algebra. SAXPY stands for the vector computation $S = a \times X + Y$ where S, X, Y are vectors of the same length and a is a scalar. In fact the loop we will compute is $Y = a \times X + Y$.

Your assignment is to do **Problem 4.14 (f) and (j)** in your textbook.

You'll need to make some assumptions about the machine model. Below are some we came up with. If you want to modify them, please let us now. If you need more assumptions, indicate them clearly. If the problem in the book and the directions given in this assignment differ, follow the directions given here.

Try and follow the format that we lay out below. It is similar to what is in the book and in the course slides.

Assumptions for 4.14 (f)

Many of these assumptions will also hold for 4.14 (j) (see later).

The processor model is that of Figure 4.8 with the latencies of Figure 4.63 (ignore the latencies for the FP store instructions). Assume no limitation in the number of reservation stations, load-store buffers and instruction unit queue (the instruction unit queue contains both the integer and f-p operations).

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 unit for all operations including load, store, and branch. In the case of load and store, the EX stage does both address computation and memory access (this might be unrealistic and assumes all operands are cached but this is not relevant for this assignment). Note that a store does not use the CDB.

If there is a conflict to access the CBD, priority is given in this order: F-P multiply, F-P add, Load, integer unit. The F-P add and the integer unit are considered busy if they cannot broadcast their result. The F-P units are fully pipelined though.

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

An instruction goes through 3 stages (IS,EX,WR). Show the timings for the first iteration and up to the completion of the ADDD, i.e., the fourth instruction, of the second iteration (you don't have to show the timings of the instructions following this ADDD). Show the status of non-empty reservation stations and the status of registers waiting for results at the time when the ADDD of the second iteration starts executing.

The timings should be shown in a table with the following format (in the comments column, indicate the reason for a stall of the corresponding instruction):

Instruction	Unit + Res. Station	IS	Cycle Number EX	WR	Comments
LD F2,0(r1)	Load - 1	1	2	3	

The status of non-empty reservation stations should be shown in a table with the following format:

Unit + Res. Station	Busy	V_j	V_k	Q_j	Q_k

Indicate the status of the floating-point registers awaiting a pending result with a table like:

Field	F#	F#	etc...	F#

In these tables a tag should appear without "()", e.g., Load - 1, and a value that comes from a functional unit or a register should appear within "()", e.g., (Load - 1).

How long does an iteration take (e.g., completion of the first MULTD till completion of the second MULTD)? What is the "critical path" of the computation? Are there any instructions that write their results "out-of-order"?

Additional assumptions for 4.14 (j)

The processor model now has a reorder buffer, as in Figure 4.34, but except for that feature, it has the same structure and latencies as in 4.14 (f). Now instructions will be completed in order. That is, an instruction now goes through 4 stages (IS, EX, WR, Commit).

Show the timings, status of reservation stations and of floating-point registers for the same conditions as above. Of course, your first table should have one additional column for "Commit" as in:

Instruction	Unit + Res. Station	IS	Cycle Number EX	WR	Commit	Comments
LD F2,0(r1)	Load - 1	1	2	3	4	

Show also the contents of the reorder buffer at the time when the ADDD of the second iteration starts executing.

Compare the results that you obtain for these two processor models.