Chapter 4 Homework

Due November 12, 1998

Question 1 (15 points)

Exercise 4.6

Question 2 (15 points)

Exercise 4.10

Question 3 (40 points)

Use the base pipeline stages, assumptions and instruction sequence on slides 9 and 10 of Lecture 10. Assume that we have a centralized window issue queue machine explicit register-renaming and the following characteristics:

- The dispatch (D) stage of the processor described in the lecture notes is replaced with a register read (RF) stage.
- Issue queue/ reorder buffer has 10 entries
- There are 10 physical integer registers
- There are 10 physical floating point registers
- The processor maintains precise interrupts
- An instruction in the issue queue can be in one of the following states: wait, register read, execute, data bus, commit.

Show the state of the issue queue/ROB, register maps (working, commit) and register files after cycle 10.

Show the pipeline diagram for as many iterations of the loop as it takes for the machine to reach steady state.

What is the number of cycles/iteration?

What is the number of instructions per cycle?

Question 4 (60 points):

You have been asked to consult by a startup company attempting to build a new quad-issue superscalar microprocessor that uses Tomasulo's algorithm with a reorder buffer to schedule its pipeline. They have asked you to check their analysis of their processor, since their measured results do not match expectations. You are given the following specs. (See the diagram on a following page for a graphical version)

1. **IF stage:** Their processor is equipped with a 4-entry instruction queue that can fetch 2 or 4 instructions at once, but not 1 or 3. It will fetch as many as possible each cycle, but will never fetch more than can fit in the instruction queue at once (so you can have an empty spot in the queue, but you can't overfill it). It is also equipped with a BTB that can look up 1 branch target per cycle. The ISA (which is otherwise identical to DLX without delay slots) requires that only one branch appear in each pair of instructions.

2. **ID stage:** Registers are fetched from the register file and reorder buffer and sent to the reservation stations if possible. Up to 4 instructions may be issued (in-order only) as long as there are no RAW dependencies between the instructions and there are available reservation stations. At most one instruction can be issued per cycle to a functional unit's reservation station. The instruction will be issued to the lowest-numbered reservation station currently available. One of the 16 reorder buffer entries also must be available for allocation to each issued instruction during this stage. No instruction may stall in this stage — it must stall in the instruction queue or a reservation station.

3. **RS stage(s):** Instructions may sit in reservation stations as long as necessary to receive forwarded results. They must sit in a reservation station for at least one cycle, however, even if they get their results immediately. An instruction may start its first EX stage in the cycle after any forwarding instructions have gone through their C stages. However, instructions will stall for an additional cycle if they would cause a common data bus (CDB) conflict during completion. All functional units have just 2 reservation stations.

4. **EX stage(s):** Instructions actually execute in the various functional units, taking a varying amount of time to do so. The branch and integer/store address units each take 1 cycle. The load units each take 2 cycles, and are fully pipelined. The FP add/multiply unit takes 3 cycles, and is also fully pipelined. The divide unit takes 5 cycles, and is not pipelined at all.

5. **C stage:** Instruction results are put on a CDB, and the reorder buffer and reservation stations may pick up the results during this cycle — but may only use the results during the next cycle. Only 4 instructions may be in completion at a time, and excess instructions would have stalled back in RS for an additional cycle to avoid this.

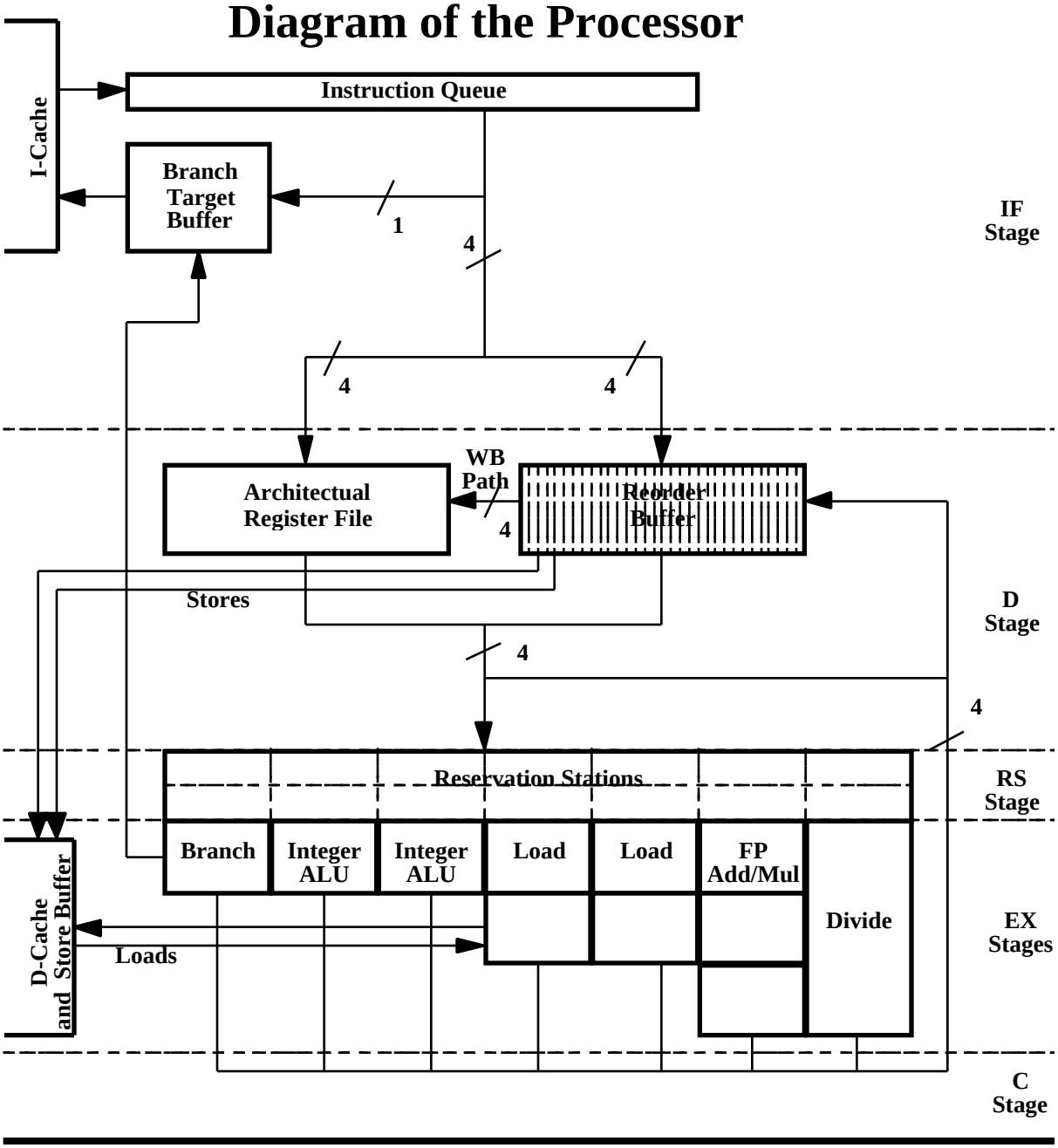
6. **RB stage(s):** These optional stages occur when an instruction must sit in the reorder buffer to allow other instructions to catch up and pass out of the reorder buffer in-order. There are a total of 16 entries here.

7. **WB stage:** Up to 4 completed and unspeculated instructions may commit back to the architectual register file per cycle. Stores are issued to the store buffer at the same time, where they complete whenever possible (assume a perfect store buffer for this problem).

The engineers give you the following test code to analyze, which averages a row and column of a 2-D, row-major, C-style array (where rows are together as arrays, but different rows are separate arrays in memory tracked by an array of pointers to the many row arrays):

```
loop: LDD   f1, 0(r1)         // Load row entry
            LD    r3, 0(r2)   // Load pointer to column's next row
            ADD   r3, r3, r4  // Add offset to the column (in r4)
            LDD   f2, 0(r3)   // Load column
            ADDD  f1, f1, f2  // Add terms
            DIV   f1, f1, f3  // Divide by 2.0000, stored in f3
            STD   0(r5), f1   // Store away
            ADD   r1, r1, #4  // Increment pointers &
            SUB   r7, r1, r6  // Loop-end test
            ADD   r2, r2, #4
            ADD   r5, r5, #4
            BNZ   r7, loop    // Loop-closing branch
```

Diagram of the Processor



An Example Pipeline: LD

IF	D	1 or more RS	EX1 (AG)	EX2 (MEM)	C	0 or more RB	WB

The engineers give you the pipeline diagram produced by their simulations of the test code (reproduced on a following page), and ask you to check the pipeline diagram in two ways:

a) *[20 points]* They are sure that the first 7 cycles of the loop have been plotted out correctly, but are not sure what the contents of the processor's buffers are at that point. They thus ask you to summarize the contents of the buffers at the end of cycle 7 on this chart, using the following notation:

- If a location has a value that was originally in the register file, put "old F*n*" or "old R*n*" there
- If a location has a value produced by an instruction, put "result #*n*" there, where n is the number of the instruction.
- If a location in the register file or reservation stations are waiting for pending results from an instruction, put the instruction's reorder buffer entry there as "pending #*n*."
- If a location in the reorder buffer is waiting for pending values, put "pending *UNIT #n*," where *UNIT #n* is the name and number of the reservation station holding the pending instruction.
- If a reorder buffer entry or reservation station is not allocated, note that it is free.

Reorder:				
	1. LDD	_____	7. STD	_____
	2. LD	_____	8. ADD	_____
	3. ADD	_____	9. SUB	_____
	4. LDD	_____	10. ADD	_____
	5. ADDD	_____	11. ADD	_____
	6. DIV	_____	12. BNZ	_____

Register File:			
	F1_____	F2_____	
	R1_____	R2_____	R3_____
	R5_____	R6_____	R7_____

		Reservation Station #1		**Reservation Station #2**	
		Op 1	**Op 2**	**Op 1**	**Op 2**
Reservation	Branch	_____	_____	_____	_____
Stations:	Integer–1	_____	_____	_____	_____
	Integer–2	_____	_____	_____	_____
	Load–1	_____	_____	_____	_____
	Load–2	_____	_____	_____	_____
	FP ALU	_____	_____	_____	_____
	Divide	_____	_____	_____	_____