16.482 / 16.561: Computer Architecture and Design

Spring 2015

Homework #5 Solution

(See assignment for full problem statement, description of nested loops, etc.)

1. *Dependences (40 points) Identify all true dependences, anti-dependences, and output dependences in the code above. Remember, you have a pair of nested loops, so you will likely find loop-carried dependences in the code.*

Solution: I've labeled each instruction in the code to make it easier to write the solution:

```
I0:       DADDI     R3, R0, #4
outer:    DADDI     R2, R1, #32
inner:    L.D       F0, 0(R1)
I1:       MULT.D    F6, F0, F6
I2:       S.D       F6, 8(R1)
I3:       DADDI     R1, R1, #16
I4:       BNE       R2, R1, inner
I5:       DADDI     R3, R3, #-2
I6:       BNEZ      R3, outer
```

True dependences are listed in the form
<register>: <producing inst.> → <consuming inst.>

Name dependences don't really have producing/consuming instructions, so the order doesn't matter. Loop-carried dependences are indicated by the letters (LC).

True dependences (potential RAW hazards):

R3: I0 → I5
R2: outer → I4
F0: inner → I1
F6: I1 → I2
F6: I1 → I1 (LC)
R1: I3 → I4
R1: I3 → outer (LC)
R1: I3 → I3 (LC)
R3: I5 → I6
R3: I5 → I5 (LC)

1 (continued) The re-labeled code is listed again here for reference:

```
I0:       DADDI     R3, R0, #4
outer:    DADDI     R2, R1, #32
inner:    L.D       F0, 0(R1)
I1:       MULT.D    F6, F0, F6
I2:       S.D       F6, 8(R1)
I3:       DADDI     R1, R1, #16
I4:       BNE       R2, R1, inner
I5:       DADDI     R3, R3, #-2
I6:       BNEZ      R3, outer
```

Anti-dependences (potential WAR hazards): Note that, if two instructions in a loop have a true dependence, those same two instructions have an anti-dependence if multiple loop iterations are taken into account. Such anti-dependences account for most of the entries in this list, and they're listed as loop-carried below:

```
R2: outer → I4 (LC)
F0: inner → I1 (LC)
F6: I1 → I1 (LC)
F6: I1 → I2 (LC)
R1: I3 → outer
R1: I3 → inner
R1: I3 → I2
R1: I3 → I3 (LC)
R1: I3 → I4 (LC)
R3: I5 → I5 (LC)
R3: I5 → I6 (LC)
```

Output dependences (potential WAW hazards): Most of the output dependences in this code are due to the fact that, in a loop, every instruction with a register destination has a loop-carried output dependence with the same instruction in a later iteration. Rather than using the same list format as above, let me note that the instructions in this class in this example use the following labels: **`outer, inner, I1, I3, I5`**

The lone output dependence that does not fit the description above involves **`R3`, which is written by both `I0` and `I5`**.

2. *Register renaming (20 points) Write out the instructions used in one outer loop iteration (which will contain multiple inner loop iterations) and show how the registers in that loop iteration are renamed.*

   Assume that DADDI instructions can use reservation stations named Add1-Add9, load instructions can use reservation stations named Load1-Load9, and multiply instructions use reservation stations named Mult1-Mult9. (You should not need all of these reservation stations.) Note that, although store and branch instructions are assigned reservation stations, they do not write to registers and therefore do not need to rename their destination operands.

Solution: The renamed code sequence is listed below. Note that every outer loop iteration contains 2 inner loop iterations—initially, R2 = R1 + 32, R1 is incremented by 16 every iteration, and the loop ends when R1 = R2, regardless of what the initial value of R1 is.

```
          DADDI    Add1, R0, #4
outer:    DADDI    Add2, R1, #32
inner:    L.D      Load1, 0(R1)          1st inner iteration
          MULT.D   Mult1, Load1, F6
          S.D      Mult1, 8(R1)
          DADDI    Add3, R1, #16
          BNE      Add2, Add3, inner
inner:    L.D      Load2, 0(Add3)        2nd inner iteration
          MULT.D   Mult2, Load2, Mult1
          S.D      Mult2, 8(Add3)
          DADDI    Add4, Add3, #16
          BNE      Add2, Add4, inner
          DADDI    Add5, Add1, #-2
          BNEZ     Add5, outer
```

3. *Dynamic scheduling (40 points) Assume the following latencies:*
   - *1 cycle for DADDI, BNE, and BNEZ*
   - *3 cycles (1 EX, 2 MEM) for L.D and S.D*
   - *4 cycles for MULT.D*

Write a pipeline diagram to show how long all instructions in this nested loop will take in a processor using Tomasulo's Algorithm. Assume that you do not have speculation and that you cannot fetch the target of a branch until the branch outcome is known. Assume the earliest you can determine the branch outcome is the IS stage, but note that you may have to wait for all source operands to become available before you can compare them and determine if the branch is taken.

Solution: As mentioned above, each outer loop iteration contains 2 inner loop iterations. There should be a total of 2 outer loop iterations (R3 = 4 at the start and is decremented by 2 every iteration; the loop ends when R2 = 0).

Your answer will depend on when exactly the branch is resolved, but assume that, as noted in the problem, you can determine the outcome in the IS stage if all values are available. In that case, the answer, as shown in the attached pipeline diagram, is 39 cycles**.**

Pipeline diagram for HW 5, question 3

IF = Instruction fetch, IS = Issue, EX = Execute, M = Memory, WB = Write back (complete)

Stalls due to RAW hazards shown in **red**; fetch stalls due to unresolved branches shown in **blue**

Outer iteration	Inner iteration		Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
			DADDI R3,R0,#4	IF	IS	EX	WB																																			
1		outer:	DADDI R2,R1,#32		IF	IS	EX	WB																																		
1	1	inner:	L.D F0,0(R1)			IF	IS	EX	M	M	WB																															
1	1		MULT.D F6, F0, F6				IF	IS	***S***	***S***	EX	EX	EX	EX	WB																											
1	1		S.D F6,8(R1)					IF	IS	EX	***S***	***S***	***S***	***S***	M	M																											
1	1		DADDI R1,R1,#16						IF	IS	EX	WB																														
1	1		BNE R2,R1,inner							IF	IS	EX																														
1	2	inner:	L.D F0,0(R1)								***S***	***S***	IF	IS	EX	M	M	WB																									
1	2		MULT.D F6, F0, F6										IF	IS	***S***	***S***	EX	EX	EX	EX	WB																				
1	2		S.D F6,8(R1)											IF	IS	EX	***S***	***S***	***S***	***S***	M	M																			
1	2		DADDI R1,R1,#16												IF	IS	EX	WB																							
1	2		BNE R2,R1,inner													IF	IS	EX																							
1			DADDI R3,R3,#-2														***S***	***S***	IF	IS	EX	WB																			
1			BNEZ R3,outer																		IF	IS	EX																		
2		outer:	DADDI R2,R1,#32																			***S***	***S***	IF	IS	EX	WB														
2	1	inner:	L.D F0,0(R1)																						IF	IS	EX	M	M	WB											
2	1		MULT.D F6, F0, F6																							IF	IS	***S***	***S***	EX	EX	EX	EX	WB							
2	1		S.D F6,8(R1)																								IF	IS	EX	***S***	***S***	***S***	***S***	M	M						
2	1		DADDI R1,R1,#16																									IF	IS	EX	WB									
2	1		BNE R2,R1,inner																										IF	IS	EX									
2	2	inner:	L.D F0,0(R1)																											***S***	***S***	IF	IS	EX	M	M	WB			
2	2		MULT.D F6, F0, F6																													IF	IS	***S***	***S***	EX	EX	EX	EX	WB	
2	2		S.D F6,8(R1)																															IF	IS	EX	***S***	***S***	***S***	***S***	M	M
2	2		DADDI R1,R1,#16																																IF	IS	EX	WB				
2	2		BNE R2,R1,inner																																	IF	IS	EX				
2			DADDI R3,R3,#-2																																		***S***	***S***	IF	IS	EX	WB
2			BNEZ R3,outer																																					IF	IS	EX