16.482 / 16.561: Computer Architecture and Design

Summer 2015

Homework #5 Solution

1. *Dynamic scheduling (30 points) Given the loop below:*

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

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*

How long would this nested loop take without speculation? Remember, without speculation, you cannot fetch past a branch until the outcome of the branch is known.

Solution: First of all, note that 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), and every outer loop iteration contains 2 inner loop iterations (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).

Your answer will depend on when exactly the branch is resolved, but assume you don't know until the end of the EX stages. In that case, the answer, as shown in the attached pipeline diagram, is 39 cycles**.**

2. *Speculation (30 points) How many cycles will the sequence in Question 1 take if we do allow speculation and assume every branch prediction—including the predicted target from the BTB—is correct?*

Solution: Allowing speculation removes the fetch stall cycles; as shown in the diagram, there are 10 such cycles in the loop. However, some instructions must wait to commit, a problem not present in part (a). As shown in the diagram, the loop takes 36 cycles with speculation and perfect branch prediction.

Pipeline diagram for HW 5, question 1

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

Pipeline diagram for HW 5, question 2

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

Stalls due to RAW hazards shown in **red**

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
			DADDI R3,R0,#4	IF	IS	EX	WB	C																														
1		outer:	DADDI R2,R1,#32		IF	IS	EX	WB	C																													
1	1	inner:	L.D F0,0(R1)			IF	IS	EX	M	M	WB	C																									
1	1		MULT.D F6, F0, F6				IF	IS	***S***	***S***	EX	EX	EX	EX	WB	C																					
1	1		S.D F6,8(R1)					IF	IS	EX	***S***	***S***	***S***	***S***	M	M	C																					
1	1		DADDI R1,R1,#16						IF	IS	EX	WB						C																			
1	1		BNE R2,R1,inner							IF	IS	EX							C																		
1	2	inner:	L.D F0,0(R1)								IF	IS	EX	M	M	WB				C																	
1	2		MULT.D F6, F0, F6									IF	IS	***S***	***S***	EX	EX	EX	EX	WB	C																
1	2		S.D F6,8(R1)										IF	IS	EX	***S***	***S***	***S***	***S***	M	M	C																
1	2		DADDI R1,R1,#16											IF	IS	EX	WB						C														
1	2		BNE R2,R1,inner												IF	IS	EX							C													
1			DADDI R3,R3,#-2													IF	IS	EX	WB						C												
1			BNEZ R3,outer														IF	IS	EX							C											
2		outer:	DADDI R2,R1,#32															IF	IS	EX	WB						C										
2	1	inner:	L.D F0,0(R1)																IF	IS	EX	M	M	WB				C									
2	1		MULT.D F6, F0, F6																	IF	IS	***S***	***S***	EX	EX	EX	EX	WB	C								
2	1		S.D F6,8(R1)																		IF	IS	EX	***S***	***S***	***S***	***S***	M	M	C							
2	1		DADDI R1,R1,#16																			IF	IS	EX	WB						C						
2	1		BNE R2,R1,inner																				IF	IS	EX							C					
2	2	inner:	L.D F0,0(R1)																					IF	IS	EX	M	M	WB				C				
2	2		MULT.D F6, F0, F6																						IF	IS	***S***	***S***	EX	EX	EX	EX	WB	C			
2	2		S.D F6,8(R1)																							IF	IS	EX	***S***	***S***	***S***	***S***	M	M	C		
2	2		DADDI R1,R1,#16																								IF	IS	EX	WB						C	
2	2		BNE R2,R1,inner																									IF	IS	EX							C	
2			DADDI R3,R3,#-2																										IF	IS	EX	WB						C
2			BNEZ R3,outer																											IF	IS	EX						

3. *Speculation & branch prediction (40 points) Now, assume the processor has a 2-bit BHT to predict branch outcomes. On a mispredicted branch, the correct instructions are fetched starting with the cycle after the misprediction is recognized (EX). Assume that all BHT entries are initially equal to 00, and that the two branches in this example use separate BHT entries. Also, assume the BTB correctly predicts all targets for taken branches. How long will the loop in Question 1 now take?*

Solution: Once again, the pipeline diagram is attached. Note that, with a mispredicted branch, incorrect instructions are fetched until the branch is resolved in the EX stage. In two cases, we actually have incorrectly fetched branches that access the BHT (shown in red in the table below), but they do not affect the operation of the program.

Branch	Prediction	Actual	Updated BHT entry	Cycle updated
BNE	NT	T	01	9
BNEZ	NT	T	--	Branch is squashed before updating BHT
BNE	NT	NT	00	16
BNEZ	NT	T	01	18
BNE	NT	T	01	26
BNEZ	NT	NT	--	Branch is squashed before updating BHT
BNE	NT	NT	00	33
BNEZ	NT	NT	00	35

Also, note that we don't actually know the instructions to be executed after the mispredicted BNEZ; those instructions are indicated using question marks in the pipeline diagram.

We can see that the code takes 42 cycles to execute.

Pipeline diagram for HW 5, question 3

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

Stalls due to RAW hazards shown in **red**

Mispredicted branches and incorrectly fetched instructions (and the associated squashed operations) are marked 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	40	41	42
			DADDI R3,R0,#4	IF	IS	EX	WB	C																																			
1		outer:	DADDI R2,R1,#32		IF	IS	EX	WB	C																																		
1	1	inner:	L.D F0,0(R1)			IF	IS	EX	M	M	WB	C																														
1	1		MULT.D F6, F0, F6				IF	IS	*S*	*S*	EX	EX	EX	EX	WB	C																										
1	1		S.D F6,8(R1)					IF	IS	EX	*S*	*S*	*S*	*S*	M	M	C																									
1	1		DADDI R1,R1,#16						IF	IS	EX	WB						C																								
1	**1**		**BNE R2,R1,inner**							**IF**	**IS**	**EX**							**C**																							
1			**DADDI R3,R3,#-2**								**IF**	**IS**																														
1			**BNEZ R3,outer**									**IF**																														
1	2	inner:	L.D F0,0(R1)										IF	IS	EX	M	M	WB		C																						
1	2		MULT.D F6, F0, F6											IF	IS	*S*	*S*	EX	EX	EX	EX	WB	C																		
1	2		S.D F6,8(R1)												IF	IS	EX	*S*	*S*	*S*	*S*	M	M	C																	
1	2		DADDI R1,R1,#16													IF	IS	EX	WB						C																
1	2		BNE R2,R1,inner														IF	IS	EX							C															
1			DADDI R3,R3,#-2															IF	IS	EX	WB						C														
1			**BNEZ R3,outer**																**IF**	**IS**	**EX**							**C**													
?			**??**																	**IF**	**IS**																			
?			**??**																		**IF**																			
2		outer:	DADDI R2,R1,#32																			IF	IS	EX	WB				C												
2	1	inner:	L.D F0,0(R1)																				IF	IS	EX	M	M	WB		C											
2	1		MULT.D F6, F0, F6																					IF	IS	*S*	*S*	EX	EX	EX	EX	WB	C								
2	1		S.D F6,8(R1)																						IF	IS	EX	*S*	*S*	*S*	*S*	M	M	C							
2	1		DADDI R1,R1,#16																							IF	IS	EX	WB					C							
2	**1**		**BNE R2,R1,inner**																								**IF**	**IS**	**EX**							**C**					
2			**DADDI R3,R3,#-2**																									**IF**	**IS**												
2			**BNEZ R3,outer**																										**IF**												
2	2	inner:	L.D F0,0(R1)																											IF	IS	EX	M	M	WB		C				
2	2		MULT.D F6, F0, F6																												IF	IS	*S*	*S*	EX	EX	EX	EX	WB	C		
2	2		S.D F6,8(R1)																													IF	IS	EX	*S*	*S*	*S*	*S*	M	M	C	
2	2		DADDI R1,R1,#16																														IF	IS	EX	WB						C	
2	2		BNE R2,R1,inner																															IF	IS	EX							C	
2			DADDI R3,R3,#-2																																IF	IS	EX	WB					C	
2			BNEZ R3,outer																																	IF	IS	EX						C

4. Multithreading (50 points) Given the three threads shown below, determine how long they take to execute using (a) fine-grained multithreading, (b) coarse-grained multithreading, and (c) simultaneous multithreading.

For coarse-grained multithreading, switch threads on any stall longer than 1 cycle. (Note that you must determine the number of stall cycles based on dependences between instructions.) For simultaneous multithreading, treat thread 1 as the preferred thread, followed by thread 2 and thread 3.

Assume you are using a processor with the following characteristics:

- *6 functional units: 3 ALUs, 2 memory ports (load/store), 1 branch*
- *In-order execution*
- *The following instruction latencies:*
  - *L.D/S.D: 4 cycles (1 EX, 3 MEM)*
  - *ADD.D/SUB.D: 2 cycles*
  - *All other operations: 1 cycle*

Thread 1:

```
L.D F0, 0(R1)
L.D F2, 8(R1)
ADD.D F4, F0, F2
SUB.D F6, F2, F0
S.D F4, 16(R1)
S.D F6, 24(R1)
DSUBUI R1, R1, #32
BNEZ R1, loop
```

Thread 2:

```
DADDUI R1, R1, #24
ADD.D F2, F0, F4
ADD.D F4, F6, F8
ADD.D F6, F0, F6
S.D F2, -24(R1)
S.D F4, -16(R1)
S.D F6, -8(R1)
BEQ R1, R7, end
```

Thread 3:

```
L.D F6, 0(R1)
ADD.D F8, F8, F6
S.D F8, 8(R1)
DADDUI R1, R1, #16
BNE R1, R2, loop
L.D F6, 0(R1)
ADD.D F8, F8, F6
S.D F8, 8(R1)
DADDUI R1, R1, #16
BNE R1, R2, loop
```

Solution: When dealing with these threads, the first step is really to identify the dependences and the latency of the producing instructions. Doing so allows you to figure out both what instructions are independent, and how many cycles are required between dependent instructions.

Note that just writing the stalls as you normally would using in-order execution isn't sufficient. You may run into cases in which stalls that are hidden in a single issue processor show up in multithreading (or any multiple issue machine, for that matter) because you're executing multiple instructions in each cycle.

The breakdown starts on the next page. Note that each instruction has been numbered to make it easier to list the dependences. The number of cycles shown after each dependence is the number of cycles required between the producing and consuming instructions. If no other instructions are available during this time, the thread will stall.

Thread 1:

```
(1)     L.D F0, 0(R1)
(2)     L.D F2, 8(R1)
(3)     ADD.D F4, F0, F2
(4)     SUB.D F6, F2, F0
(5)     S.D F4, 16(R1)
(6)     S.D F6, 24(R1)
(7)     DSUBUI R1, R1, #32
(8)     BNEZ R1, loop
```

Dependences:

(1) → (3)	3 cycles
(2) → (3)	3 cycles
(3) → (5)	1 cycle
(4) → (6)	1 cycle
(7) → (8)	0 cycles

Thread 2:

```
(1)     DADDUI R1, R1, #24
(2)     ADD.D F2, F0, F4
(3)     ADD.D F4, F6, F8
(4)     ADD.D F6, F0, F6
(5)     S.D F2, -24(R1)
(6)     S.D F4, -16(R1)
(7)     S.D F6, -8(R1)
(8)     BEQ R1, R7, end
```

Dependences:

(1) → (5)	0 cycles
(1) → (6)	0 cycles
(1) → (7)	0 cycles
(1) → (8)	0 cycles
(2) → (5)	1 cycle
(3) → (6)	1 cycle
(4) → (7)	1 cycle

Thread 3:

```
(1)     L.D F6, 0(R1)
(2)     ADD.D F8, F8, F6
(3)     S.D F8, 8(R1)
(4)     DADDUI R1, R1, #16
(5)     BNE R1, R2, loop
(6)     L.D F6, 0(R1)
(7)     ADD.D F8, F8, F6
(8)     S.D F8, 8(R1)
(9)     DADDUI R1, R1, #16
(10)    BNE R1, R2, loop
```

Dependences:

(1) → (2)	3 cycles
(2) → (3)	1 cycle
(4) → (5)	0 cycles
(4) → (6)	0 cycles
(6) → (7)	3 cycles
(7) → (8)	1 cycle
(9) → (10)	0 cycles

*We can now considering the scheduling of these threads under each multithreading scheme, starting with **(a) fine-grained multithreading**, in which we alternate threads every cycle. This technique takes 20 cycles to execute all three threads.*

Cycle	ALU1	ALU2	ALU3	Mem1	Mem2	Branch	
1				T1: L.D	T1: L.D		
2	T2: DADDUI	T2: ADD.D	T2: ADD.D				
3				T3: L.D			
4							T1 stall
5	T2: ADD.D			T2: S.D	T2: S.D		
6							T3 stall
7	T1: ADD.D	T1: SUB.D					
8				T2: S.D		T2: BEQ	
9	T3: ADD.D						
10	T1: DSUBUI			T1: S.D	T1:S.D		
11	T3: DADDUI			T3: S.D			
12						T1: BNEZ	
13				T3: L.D		T3: BNE	
14							T3 stall
15							T3 stall
16							T3 stall
17	T3: ADD.D						
18							T3 stall
19	T3: DADDUI			T3: S.D			
20						T3: BNE	

*Next, **(b) coarse-grained multithreading**, where we switch threads on any stall requiring more than 1 cycle—namely, the three-cycle stalls in both Thread 1 and Thread 3. Note that, with this technique, Thread 2 will run to completion without being switched out. However, we do have to be careful when scheduling that thread, as there are dependences between the ADD.D and S.D instructions that must be satisfied. Overall, coarse-grained and fine-grained multithreading perform similarly, as the threads take a total of 21 cycles to complete.*

Cycle	ALU1	ALU2	ALU3	Mem1	Mem2	Branch	
1				T1: L.D	T1: L.D		
2	T2: DADDUI	T2: ADD.D	T2: ADD.D				
3	T2: ADD.D						
4				T2: S.D	T2: S.D		
5				T2: S.D		T2: BEQ	
6				T3: L.D			
7	T1: ADD.D	T1: SUB.D					
8							T1 stall
9	T1: DSUBUI			T1: S.D	T1:S.D		
10						T1: BNEZ	
11	T3: ADD.D						
12							T3 stall
13	T3: DADDUI			T3: S.D			
14				T3: L.D		T3: BNE	
15							T3 stall
16							T3 stall
17							T3 stall
18	T3: ADD.D						
19							T3 stall
20	T3: DADDUI			T3: S.D			
21						T3: BNE	

*Our last technique is **(c) simultaneous multithreading**. Thread 1 is the preferred thread, followed by Threads 2 and 3. This method allows for the best overall usage of functional units and takes only 16 cycles for all three threads to complete.*

Cycle	ALU1	ALU2	ALU3	Mem1	Mem2	Branch	
1	T2: DADDUI	T2: ADD.D	T2: ADD.D	T1: L.D	T1: L.D		
2	T2: ADD.D			T3: L.D			
3				T2: S.D	T2: S.D		
4				T2: S.D		T2: BEQ	
5	T1: ADD.D	T1: SUB.D					
6	T3: ADD.D						
7	T1: DSUBUI			T1: S.D	T1:S.D		
8	T3: DADDUI			T3: S.D		T1: BNEZ	
9				T3: L.D		T3: BNE	
10							T3 stall
11							T3 stall
12							T3 stall
13	T3: ADD.D						
14							T3 stall
15	T3: DADDUI			T3: S.D			
16						T3: BNE	