the potential speedup of superpipelining the 5-stage pipeline is quite small.

Problem 3.13

a. Tomasulo algorithm--no speculation.

Table 17: Tomasulo algorithm--no speculation

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	COMMENTS
I1	L.D F0,0(R1)	1	2	(3)	3	(4)	(5)	
I2	L.D F2,0(R2)	2	3	(4)	4	(5)	(6)	
I3	L.D F4,0(R3)	3	4	(5)	5	(6)	(7)	
I4	MUL.D F6,F2,F0	4	7	(8)	12	--	(13)	wait for F2
I5	ADD.D F8,F6,F4	5	14	(15)	19	--	(20)	wait for F6
I6	ADDI R1,R1,#8	6	7	(8)	8	--	(9)	
I7	ADDI R2,R2,#8	7	8	(9)	9	--	(10)	
I8	ADDI R3,R3,#8	8	9	(10)	10	--	(11)	
I9	S.S-A F8,-8(R2)	9	11	(12)	12	--	--	wait for R2
I10	S.S-D F8,-8(R2)	10	21	(22)	22	(23)	--	wait for F8
I11	BNE R4,R2,LOOP	11	12	(13)	13	--	(14)	
I12	L.D F0,0(R1)	15	16	(17)	17	(18)	(19)	wait for I11 in dispatch

The values between parentheses show resources reserved by an instruction at the time of dispatch. Store I10 can issue to cache at clock 23 because no memory instruction is pending in the L/S queue. Load I12 can issue to cache at clock 18 because the only preceding and pending memory instruction is the store I10 and the address of the store is known at the end of clock 12. Load I12 cannot dispatch until it knows the outcome of branch I11, at the end of clock 14.

b. Tomasulo algorithm with speculation.

Table 18: Tomasulo algorithm with speculation

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	Retire	COMMENTS
I1	L.D F0,0(R1)	1	2	(3)	3	(4)	(5)	6	
I2	L.D F2,0(R2)	2	3	(4)	4	(5)	(6)	7	
I3	L.D F4,0(R3)	3	4	(5)	5	(6)	(7)	8	
I4	MULT.D F6,F2,F0	4	7	(8)	12	--	(13)	14	wait for F2
I5	ADD.D F8,F6,F4	5	14	(15)	19	--	(20)	21	wait for F6
I6	ADDI R1,R1,#8	6	7	(8)	8	--	(9)	22	
I7	ADDI R2,R2,#8	7	8	(9)	9	--	(10)	23	
I8	ADDI R3,R3,#8	8	9	(10)	10	--	(11)	24	
I9	S.D-A F8,-8(R2)	9	11	(12)	12	--	--	--	wait for R2
I10	S.D-D F8,-8(R2)	10	21	(22)	22	24	--	25	wait for F8, then wait to reach top of ROB
I11	BNE R4,R2,LOOP	11	12	(13)	13	--	(14)	26	
I12	L.D F0,0(R1)	12	13	(14)	14	(15)	(16)	27	Address of store is known since cycle 13

Store I10 reaches the L/S queue at the end of clock 22. To proceed to cache it must verify that it is at the top of the ROB. The previous instruction in process order is I8 and it retires at clock 24. Therefore the store is at the top of the ROB during clock 24 and it may issue to cache in clock 24. Note that now load I12 can dispatch before it knows the outcome of branch I11.

c. Speculation with speculative scheduling

Table 19: Tomasulo algorithm with speculation and speculative scheduling

		Dispatch	Issue	Register Fetch	Exec start	Exec complete	Cache	CDB	Retire	Comments
I1	L.D F0,0(R1)	1	2	3	(4)	4	(5)	(6)	7	
I2	L.D F2,0(R2)	2	3	4	(5)	5	(6)	(7)	8	
I3	L.D F4,0(R3)	3	4	5	(6)	6	(7)	(8)	9	
I4	MULT.D F6,F2,F0	4	5	6	(7)	11	--	(12)	13	
I5	ADD.D F8,F6,F4	5	10	11	(12)	16	--	(17)	18	wait for F6
I6	ADDI R1,R1,#8	6	7	8	(9)	9	--	(10)	19	
I7	ADDI R2,R2,#8	7	8	9	(10)	10	--	(11)	20	
I8	ADDI R3,R3,#8	8	10	11	(12)	12	--	(13)	21	CDB conflict with I4
I9	S.D-A F8,-8(R2)	9	10	11	(12)	12	--	--	--	
I10	S.D-D F8,-8(R2)	10	15	16	17	17	21	--	22	wait for F8, then wait to reach top of ROB
I11	BNE R4,R2,LOOP	11	12	13	(14)	14	--	(15)	23	
I12	L.D F0,0(R1)	12	14	15	(16)	16	(17)	(18)	24	CDB conflict with I5

Because of speculative scheduling the MULT instruction I4 does not have to wait for F2 anymore because F2 is forwarded right on time through the common data bus in clock 7. By issuing I4 ahead of time, the forwarded F2 value is the input to the FP unit at the beginning of cycle 7. Instruction I8 cannot issue in clock 9, although it has no hazards with prior instructions because it cannot reserve the CDB for clock 12 (because it was reserved at clock 5 by I4). Store I10 waits for F8 to issue. F8 is put on the CDB in clock 17 and the store can execute in the AGU in clock 17 and is issued speculatively at clock 15. Then the store reaches the L/S queue where it waits to reach the top of the ROB, in clock 21. At the beginning of clock 21 the store is at the top of the ROB because the preceding instruction (I8) is in the retirement unit in clock 21.

d.

Each instruction in the data-flow graph is labeled as follows.

```
I1          L.D F0,0(R1)          /X[i] loaded in F0
I2          L.D F2,0(R2)          /Y[i] loaded in F2
I3          L.D F4,0(R3)          /Z[i] loaded in F4
I4          MUL.D F6,F2,F0        /Multiply X by Y
I5          ADD.D F8,F6,F4         /Add Z
I6          ADDI R1,R1,#8        /Update address registers
I7          ADDI R2,R2,#8
I8          ADDI R3,R3,#8
I9          S.D F8, -8(R2)       /store in Y[i]
I10         BNE R4,R2,LOOP       /(R4)-8 points to the last element of Y
```

The data-flow graph is given in Figure 2. The critical path in the data-flow graph is shown in bold (I1->I4->I5). The store I9 is off the critical path because there is no RAW memory dependencies on memory, either within one iteration or across iterations of the loop.

Table 20: Execution time comparisons

Execution time metric	Tomasulo w/o spec	Tomasulo with spec	Spec scheduling	Data-flow
Issue-to-issue	16-2=14	13-2=11	14-2=12	7
Execution-to-execution	17-3=14	14-3=11	15-3=12	7
Retirement	---	27-6=21	24-7=17	7

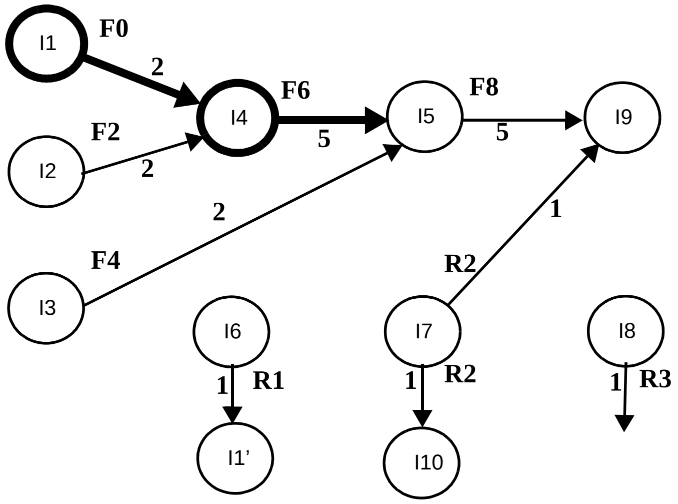

Figure 2Data-flow graph

It is a challenge to figure out the exact execution rate in a single iteration of a loop. The exercise asks to measure execution time from issue to issue of the first load. The results are shown in the first row of Table 20. Speculative execution is better than execution with no speculation. This is because of the delay needed to obtain the branch outcome. For this particular code speculative execution with speculative scheduling is slightly worse than without speculative scheduling. Looking at the schedule speculative scheduling for this particular code causes two conflicts on the CDB.
Table 20 also shows two other possible measures. The first one is between the starts of execution of the two loads. The numbers are unchanged. The second measure is the time between retirement of the two loads. This metric may be better because it only considers instructions that are non-speculative. In this case the store is included in the critical path. Under this metric, speculative scheduling improves on speculative execution without speculative scheduling.

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	Retire	COMMENTS
I22	BNE R4,R2,LOOP	28(0)	31	(32)	32	--	(33)	45	wait to get 2 ROB entries, wait for R2, and CDB conflict with I15
I23	L.D F0,0(R1)	33(1)	34	(35)	35	(36)	(37)	46	wait to get 2 ROB entries
I24	L.D F2,0(R2)	33(0)	35	(36)	36	(37)	(38)	47	wait to get 2 ROB entries and con-flict with I23

Execution Time per Iteration (issue-to-issue; second iteration) = 34 - 11 = 23 cycles. Comparing this to the execution time in part b (single dispatch) of Exercise 3.13 (11 cycles), we find that dual dispatch degrades the performance significantly. The main bottleneck is the number of ROB entries available, compounded by the fact that the machine waits until two instructions are structural hazard free to dispatch. This machine definitely needs more ROB entries.
Also, since there are a lot of data dependencies between instructions, the machine stalls instructions in issue queues while waiting for their operands and this reduces the benefits of dual dispatch. The last bottleneck is structural hazards on functional units (ALU's, cache ports, FP units), and on the single CDB.

Problem 3.15

a. Conservative Disambiguation:

Table 22: Tomasulo algorithm with speculation (two-way superscalar)-Conservative disambiguation

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	Retire	Comment
I1	L.D F2,0(R1)	1(7)	2	(3)	3	(4)	(5)	6	
I2	ADDI R1,R1,#8	1(6)	2	(3)	3	--	(4)	7	
I3	ADDI R2,R2,#8	2(5)	4	(5)	5	--	6	8	CDB conflict with I1
I4	S.D-A F2,-8(R2)	2(5)	7	(8)	8	--	--	--	wait for R2
I5	S.D-D F2,-8(R2)	3(4)	6	(7)	7	9	--	10	wait for F2 and wait for address
I6	BNEQ R1,R3,LOOP	3(3)	5	(6)	6	--	(7)	11	wait for R1
I7	L.D F2,0(R1)	4(2)	5	(6)	6	10	11	12	wait for address of previous store and CDB conflict with I9
I8	ADDI R1,R1,#8	4(1)	7	(8)	8	--	(9)	13	FU conflict with I6 and CDB conflict with I7
I9	ADDI R2,R2,#8	5(0)	8	(9)	9	--	(10)	14	CDB conflict with I7and FU conflict with I8
I10	S.D-A F2,-8(R2)	5(0)	11	(12)	12	--	--	--	wait for R2
I11	S.D-D F2,-8(R2)	7(1)	12	(13)	13	(14)	--	15	wait to get 2 ROB entries, wait for F2
I12	BNEQ R1,R3,LOOP	7(0)	10	(11)	11	--	(12)	16	wait for R1
I13	L.D F2,0(R1)	10(1)	13	(14)	14	(15)	(16)	17	wait to get 2 ROB entries and AGU conflict with I10 & I11
I14	ADDI R1,R1,#8	10(0)	11	(12)	12	--	(13)	18	wait to get 2 ROB entries

Table 22: Tomasulo algorithm with speculation (two-way superscalar)-Conservative disambiguation

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	Retire	Comment
I15	ADDI R2,R2,#8	11(0)	12	(13)	13	--	(14)	19	
I16	S.D-A F2,-8(R2)	11(0)	15	(16)	16	--	--	--	wait for R2
I17	S.D-D F2,-8(R2)	13(1)	17	(18)	18	(19)	--	20	wait to get 2 ROB entries, wait for F2
I18	BNEQ R1,R3,LOOP	13(0)	15	(16)	16	--	(17)	21	wait to get 2 ROB entries and CDB conflict with I13
I19	L.D F2,0(R1)	15(1)	16	(17)	17	(18)	(19)	22	wait to get 2 ROB entries

At the time of issue, load I7 reserved the cache for clock 7 and the CDB for clock 8. However, the load must wait in the L/S queue until the address of the previous store is known, at the end of clock 8. However if the load issues to cache in clock 9, it will conflict with I9 at clock 10. Therefore I7 must wait one more clock to issue to cache, at clock 10. Meanwhile I8 cannot issue at clock 5 because I6 has already reserved the integer unit for clock 6. It also cannot issue at clock 6 because load I7 has reserved the CDB for clock 8 (a false conflict and a CDB cycle that is lost). Similarly I9 fails to issue at first in clock 6 because of the CDB conflict with I7. Then it cannot issue at clock 7 because of the conflict with I8 on the integer unit.

b. Speculative Disambiguation:

Table 23: Tomasulo algorithm with speculation (two-way superscalar)-Speculative disambiguation

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	Retire	Comment
I1	L.D F2,0(R1)	1(7)	2	3	3	4	5	6	
I2	ADDI R1,R1,#8	1(6)	2	3	3	--	4	7	
I3	ADDI R2,R2,#8	2(5)	4	5	5	--	6	8	CDB conflict with I1
I4	S.D-A F2,-8(R2)	2(5)	7	8	8	--	--	--	wait for R2
I5	S.D-D F2,-8(R2)	3(4)	6	7	7	9	--	10	wait for F2 and wait for address
I6	BNEQ R1,R3,LOOP	3(3)	5	6	6	--	7	11	wait for R1
I7	L.D F2,0(R1)	4(2)	5	6	6	7	8	12	
I8	ADDI R1,R1,#8	4(1)	7	8	8	--	9	13	FU conflict with I6 and CDB conflict with I7
I9	ADDI R2,R2,#8	5(0)	8	9	9	--	10	14	CDB conflict with I7 and FU conflict with I8
I10	S.D-A F2,-8(R2)	5(0)	11	12	12	--	--	--	wait for R2
I11	S.D-D F2,-8(R2)	7(1)	9	10	10	14	--	15	wait to get 2 ROB entries, wait for F2, then wait to reach top of ROB
I12	BNEQ R1,R3,LOOP	7(0)	10	11	11	--	12	16	wait to get 2 ROB entries then wait for R1
I13	L.D F2,0(R1)	10(1)	12	13	13	14	15	17	wait to get 2 ROB entries and AGU conflict with I10
I14	ADDI R1,R1,#8	10(0)	11	12	12	--	13	18	wait to get 2 ROB entries
I15	ADDI R2,R2,#8	11(0)	12	13	13	--	14	19	
I16	S.D-A F2,-8(R2)	11(0)	15	16	16	--	--	--	wait for R2

Table 23: Tomasulo algorithm with speculation (two-way superscalar)-Speculative disambiguation

		Dispatch	Issue	Exec start	Exec complete	Cache	CDB	Retire	Comment
I17	S.D-D F2,-8(R2)	13(1)	16	17	17	19	--	20	wait to get 2 ROB entries, then wait for F2
I18	BNEQ R1,R3,LOOP	13(0)	14	15	15	--	16	21	wait to get 2 ROB entries
I19	L.D F2,0(R1)	15(1)	17	18	18	19	20	22	wait to get 2 ROB entries and AGU conflict with I17

Looking again at load I7, the load is (successively) issued to cache speculatively at clock 7. This avoids load delays, and eliminates the wasted cycle on the CDB.

Problem 3.16

a. With one-bit state machine, 0 is not taken (NT) and 1 is taken (T).
The values are: 8,9,10,11,7,20,29,30,31
b1: T,U,T,U,U,T,U,T,U
b2: T,T,U,T,T,U,T,U,T

Table 24: 1-bit predictor

Iteration	1	2	3	4	5	6	7	8	9
B1(Prediction/Actual Branch Direction)	0/1	1/0	0/1	1/0	**0/0**	0/1	1/0	0/1	1/0
B2(Prediction/Actual Branch Direction)	0/1	**1/1**	1/0	0/1	**1/1**	1/0	0/1	1/0	0/1

If a prediction and the actual branch direction are the same in a given entry of Table 24, it means that the branch prediction is correct, and these cases are in bold in the table above.
Also the branch direction in the iteration is the prediction in the next iteration.
Therefore, the prediction accuracy for b1 is 1/9 = 11%
The prediction accuracy for b2 is 2/9 = 22%.
The overall prediction accuracy for both branches is 3/18 = 1/6 = 16.67%

b. 2-level branch prediction scheme.
If the previous branch, either b1 or b2, is taken, g is set to 1. If the previous branch is not taken, then g is 0 in the history register. If a prediction and an actual branch direction are same, they are shown in bold in Table 25.
Thus, the prediction accuracy for b1 is 4/9 = 44.4%.
The prediction accuracy for b2 is also 6/9=66.7%.
The overall accuracy is 55.5%.

c.
The prediction success rate for b2 when g=0 is 4/5 = 80%.
For b2, g is equal to 0 when the previous branch, b1, is not taken. That means the value is odd. In the given 9 values, there is no odd number which is also a multiple of 5. Therefore, once the branch

predictor is warmed-up, b2 branch with g=0 is highly predictable.

Table 25: Two-level branch prediction scheme

Iteration	1	2	3	4	5	6	7	8	9
B1 (g/Prediction/ Actual Branch Direction)	0/0/1	**1/0/0**	1/0/1	0/1/0	1/1/0	1/0/1	**0/0/0**	**1/1/1**	**0/0/0**
B2 (g/Prediction/ Actual Branch Direction)	1/0/1	0/0/1	1/1/0	**0/1/1**	**0/1/1**	**1/0/0**	**0/1/1**	**1/0/0**	**0/1/1**

The values are: 8,9,10,11,7,20,29,30,31

b1:	T,U,T,U,U,T,U,T,U
b2:	T,T,U,T,T,U,T,U,T

Problem 3.17

First, let's think about the possible aliasing for private table access. Branch1 and Branch2 are separated by 1 instruction and their PC values differ by 8. Branch2 and Bbranch3 are separated by 2 instructions and their PC values differ by 12. Branch1 and Branch3 are separated by 4 instructions and their PC values differ by 20. Since 10 bits of the branch PC are used, there is no possibility of aliasing of branch addresses for the private tables for this piece of code.

1. GAg
The history pattern is global and the predictors are shared. In the following table, the first column indicates the loop iteration and the 2nd column indicates branch. The pattern column shows the global history and the value of the predictor. The action column shows the prediction, the actual branch action, and the updated predictor value.

Table 26:

Iteration	Branch	Pattern	Action
1	BNEZ R2, LAB1	0/0	NT/NT/0 (Success)
1	BEQZ R0, LAB2	0/0	NT/T/1
1	BNEZ R1, LOOP	1/0	NT/T/1
2	BNEZ R2, LAB1	1/1	T/T/1 (Success)
2	BNEZ R1, LOOP	1/1	T/T/1 (Success)
3	BNEZ R2, LAB1	1/1	T/NT/0
3	BEQZ R0, LAB2	0/1	T/T/1 (Success)
3	BNEZ R1, LOOP	1/0	NT/T/1
4	BNEZ R2, LAB1	1/1	T/T/1 (Success)
4	BNEZ R1, LOOP	1/1	T/T/1 (Success)
5	BNEZ R2, LAB1	1/1	T/NT/0
5	BEQZ R0, LAB2	0/1	T/T/1 (Success)