

NAME: \_\_\_\_\_

ID: \_\_\_\_\_



## INSTRUCTIONS:

- THE EXAM IS CLOSED-BOOK/CLOSED-NOTES.
- THE DURATION OF THE EXAM IS TWO HOURS AND 30 MINUTES.
- CELL PHONES ARE NOT ALLOWED INTO THE EXAMINATION ROOM.
- WRITE YOUR NAME AND ID-NUMBER IN THE SPACE PROVIDED ABOVE.
- THE EXAM IS MULTIPLE CHOICE.
- USE THE SCANTRON SHEET TO MARK THE APPROPRIATE ANSWER FOR EACH QUESTION.
- MARK YOUR NAME, STUDENT ID, AND THE VERSION OF YOUR EXAM ON THE SCANTRON CARD.
- USE ONLY PENCIL TO MARK THE SCANTRON CARD.
- CHOOSE THE BEST ANSWER FROM THE FIVE POSSIBLE ANSWERS LISTED IN EACH QUESTION.
- IN SOME QUESTIONS, MORE THAN ONE CHOICE MAY BE A VALID ANSWER. SELECT THE BEST CHOICE YOU THINK IS THE MOST APPROPRIATE ANSWER TO THE QUESTION.
- ALL QUESTIONS ARE EQUALLY WEIGHTED.
- READ THE QUESTIONS CAREFULLY BEFORE ANSWERING.
- BUDGET YOUR TIME APPROPRIATELY.
- SUBMIT THE QUESTION SHEET TOGETHER WITH THE SCANTRON CARD AT THE END OF THE EXAM.
- CHECK YOU HAVE A TOTAL OF 11 PAGES.



**Problem 1: In-order Issue Scheduling using a Scoreboard (Answer questions Q1 to Q7)**

Consider a MIPS pipeline with a floating-point unit (FPU). The FPU contains one adder, one multiplier, and a load/store unit. The adder has a **3-cycle** latency and is fully pipelined. The multiplier has a **5-cycle** latency and is fully pipelined. Assume that loads and stores take **3 cycles** to execute (plus 1 cycle for the write-back stage for loads).

The floating-point registers are separate from the integer registers. There is a single write-back port to each register file. In case of a write-back conflict, the older instruction writes back first. FP instructions (and loads writing FP registers) must spend one cycle in the WB stage before their result can be used. Integer results are available for bypass the next cycle after issue.

Issuing of instructions to function units is done using a scoreboard. Instructions issue in order. Remember in this scheme, an instruction is issued if it does not create a WAW hazard with any previous instruction that has not written back. Recall also that the WB stage is only relevant for FP instructions (integer instructions can forward results).

**Q1.** How does the scoreboard eliminate WAR hazards?

- a) It does not. Instructions cannot issue if they create WAR hazards.
- b) WAR hazards cannot occur because instructions write back results in order.
- c) WAR hazards cannot occur because instructions issue in order.
- d) By applying register renaming
- e) None of the above choices is true, or more than one of the above choices is true.



**Q2.** In the scoreboard scheduling technique (choose one)

- a) Instructions can execute and write back results out of order.
- b) Instructions can execute out of order but must write back results in order.
- c) Integer ALU instructions execute and write back results in order.
- d) Structural hazards cannot be eliminated. The programmer must ensure structural hazards do not occur.
- e) None of the above choices is true, or more than one of the above choices is true.

Now, refer to the code segment shown in Table 1. Schedule this code using the scoreboarding technique. Use the scoreboard shown in Table 2 to fill your intermediate steps. Then answer questions **Q3 to Q7**. The first two rows have been completed for you. Assume instruction I1 is issued on cycle 0.

**Q3.** Which of the following is true about instruction I2?

- a) Issues on cycle 4; writes back on cycle 9.
- b) Issues on cycle 3; writes back on cycle 8.
- c) Issues on cycle 5; writes back on cycle 9.
- d) Issues on cycle 1; writes back on cycle 6.
- e) None of the above choices is true, or more than one of the above choices is true.

**Q4.** Which of the following is true about instruction I3?

- a) Issues on cycle 9; writes back on cycle 12.
- b) Issues on cycle 10; writes back on cycle 14.
- c) Issues on cycle 10; writes back on cycle 13.
- d) Issues on cycle 5; writes back on cycle 8.
- e) None of the above choices is true, or more than one of the above choices is true.



**Q5.** Which of the following is true about instruction I4?

- a) Issues on cycle 8.
- b) Issues on cycle 9.
- c) Issues on cycle 10; Registers F2 and R2 are reserved for writes on this cycle.
- d) Issues on cycle 11; Registers F3 and R2 are reserved for writes on this cycle.
- e) None of the above choices is true, or more than one of the above choices is true.

**Q6.** Which of the following is true about instruction I6?

- a) Issues on cycle 12; writes back on cycle 15.
- b) Issues on cycle 16; writes back on cycle 21.
- c) Issues on cycle 11; writes back on cycle 14.
- d) Issues on cycle 15; writes back on cycle 20.
- e) None of the above choices is true, or more than one of the above choices is true.

Q7. Instruction I7 writes back its result on cycle:

- a) Issues on cycle 12; writes back on cycle 24.
- b) Issues on cycle 16; writes back on cycle 21.
- c) Issues on cycle 22; writes back on cycle 25.
- d) Issues on cycle 15; writes back on cycle 26.
- e) None of the above choices is true, or more than one of the above choices is true.

Table 1: Code segment for Problem 1.

I1	LD	F1, 7 (R2)
I2	FMUL	F2, F1, F0
I3	FADD	F3, F2, F0
I4	ADDI	R2, R2, 8
I5	LD	F1, 7 (R2)
I6	FMUL	F2, F1, F1
I7	FADD	F2, F2, F3

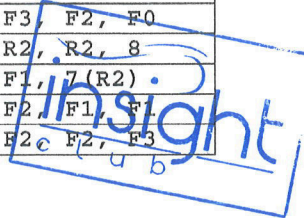
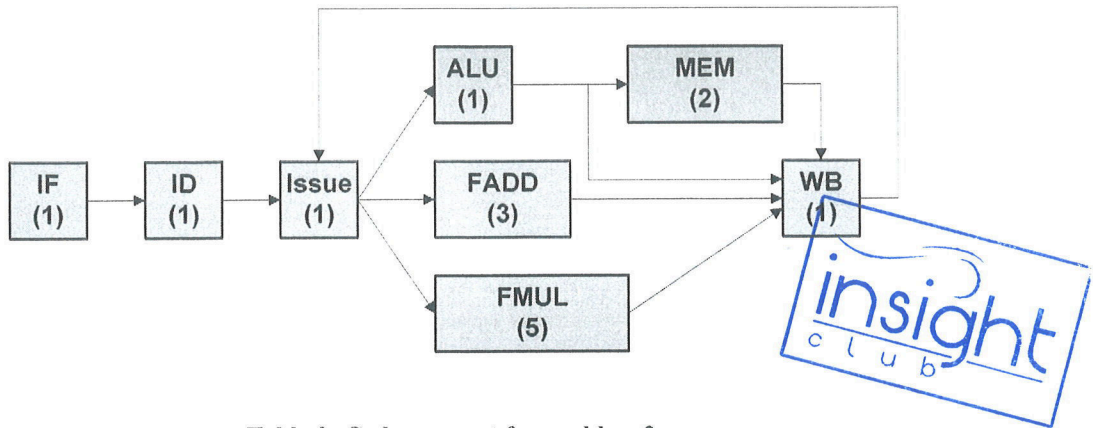


Table 2: Scoreboard for Problem 1.

Instr Issue	Time (cycles)	Functional Unit Status										FP Registers Reserved for Writes	
		INT (1)	LOAD (3)		FADD (3)			FMUL (5)			WB (1)		
I1	0		F1										F1
	1			F1									F1
	2				F1								F1
	3										F1		F1
I2	4						F2						F2
	5							F2					F2
	6								F2				F2
	7									F2			F2
	8										F2		F2
	9											F2	F2
I3	10					F3							F3
I4	11	R2					F3						F3
I5	12		F1				F3						F1, F3
	13			F1							F3		F1, F3
	14				F1								F1
	15										F1		F1
I6	16							F2					F2
	17								F2				F2
	18									F2			F2
	19										F2		F2
	20											F2	F2
	21											F2	F2
I7	22					F2							F2
	23						F2						F2
	24							F2					F2
	25										F2		F2
	26												
	27												

**Problem 2: Out-of-Order Scheduling (Questions Q8 to Q31)**

In this problem, we examine the execution of the code segment in Table 3 on the following out-of-order processor.



**Table 3: Code segment for problem 2.**

I1	LD	F1, 7 (R2)
I2	FMUL	F2, F1, F0
I3	FADD	F3, F2, F0
I4	ADDI	R2, R2, 8
I5	LD	F1, 7 (R2)
I6	FMUL	F2, F1, F1
I7	FADD	F2, F2, F3



**Assumptions:**

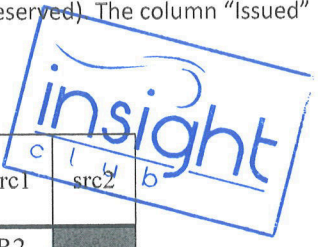
- All functional units are pipelined
- ALU operations take **1 cycle (1c)**
- Memory operations take **3 cycles (3c)** (1 cycle in ALU, followed by 2 cycles in MEM)
- Floating- point add (FADD) instructions take **3 cycles (3c)**
- Floating- point multiply (FMUL) instructions take **5 cycles (5c)**
- Floating-point registers are separate from the integer registers.
- There is a single write-back port to the FP register file and the INT register file (*one port common to both*). In the case of a WB conflict, the older instruction writes back first.
- Instructions are fetched and decoded in order
- The issue stage is a buffer (issue buffer or ROB) of unlimited length that holds instructions waiting to start execution.
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued (to function units) at a time, and in case multiple instructions are ready, the oldest one will go first
- There is no bypassing (for both integer and FP instructions). All instructions must spend one cycle in WB stage before their results can be used.

This problem consists of three parts.

- Part 1: Questions **Q8 to Q13**
- Part 2: Questions **Q14 to Q19**
- Part 3: Questions **Q20 to Q31**

**Part 1:** You are asked to schedule the code shown in Table 3. In this scheduling method, remember that an instruction enters the issue stage if it does not cause a WAR or WAW hazard. There is no register renaming. Use Table 4 to schedule the code, and then answer questions Q8 to Q13. The first row has been completed for you. In Table 4, the column 'Decode → Issue' means that the instruction has entered issue stage (operands have been read, a slot in the issue buffer has been reserved). The column "Issued" means the instruction has started to execute.

Table 4: Scheduling table



	Time (cycle)			OP	dst	src1	src2
	Decode → Issue (enter issue buffer)	Issued (start to execute)	WB				
I1	0	1	4	LD	F1	R2	
I2	1	5	10	FMUL	F2	F1	F0
I3	2	11	14	FADD	F3	F2	F0
I4	5	6	7	ADDI	R2	R2	
I5	11	12	15	LD	F1	R2	
I6	15	16	21	FMUL	F2	F1	F1
I7	22	23	26	FADD	F2	F2	F3

Latencies: ALU 1c, MEM 3c, FADD 3c, FMUL 5c, WB 1c. No forwarding.


Q8. Instruction I2 enters issue stage on cycle \_\_\_\_, starts to execute on cycle \_\_\_\_, enters WB stage on cycle \_\_\_\_:

- a) 1, 4, 9
- b) 1, 5, 11
- c) 1, 5, 10
- d) 1, 4, 10
- e) None of the above choices is true, or more than one of the above choices is true.

Q9. Instruction I3 enters issue stage on cycle \_\_\_\_, starts to execute on cycle \_\_\_\_, enters WB stage on cycle \_\_\_\_:

- a) 2, 10, 13
- b) 2, 11, 14
- c) 6, 7, 10
- d) 2, 11, 16
- e) None of the above choices is true, or more than one of the above choices is true.

Q10. Instruction I4 enters issue stage on cycle \_\_\_\_, starts to execute on cycle \_\_\_\_, enters WB stage on cycle \_\_\_\_:

- a) 15, 16, 17
  - b) 4, 6, 7
  - c) 15, 16, 19
  - d) 5, 6, 7
  - e) None of the above choices is true, or more than one of the above choices is true.
- 

Q11. Instruction I5 enters issue stage on cycle \_\_\_\_, starts to execute on cycle \_\_\_\_, enters WB stage on cycle \_\_\_\_:

- a) 11, 12, 15
- b) 8, 9, 12
- c) 8, 9, 13
- d) 18, 19, 22
- e) None of the above choices is true, or more than one of the above choices is true.

Q12. Instruction I6 enters issue stage on cycle \_\_\_\_, starts to execute on cycle \_\_\_\_, enters WB stage on cycle \_\_\_\_:

- a) 14, 15, 20
- b) 16, 17, 22
- c) 15, 16, 21
- d) 12, 16, 21
- e) None of the above choices is true, or more than one of the above choices is true.

Q13. Instruction I7 enters issue stage on cycle \_\_\_\_, starts to execute on cycle \_\_\_\_, enters WB stage on cycle \_\_\_\_:

- a) 22, 23, 28
- b) 21, 22, 25
- c) 23, 24, 27
- d) 22, 23, 26
- e) None of the above choices is true, or more than one of the above choices is true.

**Part 2:** Repeat Part 1 above now with register renaming (both integer and FP registers). Assume you are given an unlimited number of renaming resources (size of issue buffer/ROB is unlimited). The instructions are committed in order and only one instruction may be committed per cycle. The earliest time an instruction can be committed is one cycle after write back.

First rename the appropriate registers in Table 3 then perform scheduling. Use the register names T0, T1, T2, etc. (in ascending order) when renaming the registers as they appear in program order. Since you have an unlimited number of register names, you should use a new register name each time a register is written. Use the table below to write your schedule. The column 'committed' refers to the cycle when the instruction actually commits. The first row has been completed for you. Answer questions Q14 to Q19.

	Time (cycle)				OP	dst	src1	src2
	Decode → Issue (enter issue buffer)	Issued (start to execute)	WB	Committed				
I1	0	1	4	5	LD	T0	R2	
I2	1	5	10	11	FMUL	T1	T0	F0 5
I3	2	11	14	15	FADD	T2	T1	F0 3
I4	3	4	5	16	ADDI	T3	R2	
I5	4	6	9	17	LD	T4	T3	
I6	5	10	15	18	FMUL	T5	T4	T4 5
I7	6	16	19	20	FADD	T6	T5	T2 3

Latencies: ALU 1c, MEM 3c, FADD 3c, FMUL 5c, WB 1c. No forwarding.

**Q14.** I2 enters issue stage on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 1, 5, 10, 11
- b) 1, 6, 11, 12
- c) 1, 2, 7, 8
- d) 1, 4, 9, 10

e) None of the above choices is true, or more than one of the above choices is true.

**Q15.** I3 enters issue stage on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 12, 13, 16, 17
- b) 2, 12, 15, 16
- c) 2, 11, 14, 15
- d) 2, 10, 13, 14

e) None of the above choices is true, or more than one of the above choices is true.

**Q16.** I4 enters issue stage on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 3, 16, 17, 18
- b) 3, 4, 15, 16
- c) 16, 17, 18, 19
- d) 3, 4, 5, 16

e) None of the above choices is true, or more than one of the above choices is true.

**Q17.** I5 enters issue stage on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 4, 17, 20, 21
- b) 4, 6, 9, 17
- c) 5, 6, 9, 17
- d) 4, 5, 8, 17

e) None of the above choices is true, or more than one of the above choices is true.

**Q18.** I6 enters issue stage on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 5, 10, 15, 18
- b) 5, 7, 10, 18
- c) 5, 7, 10, 11
- d) 5, 12, 15, 18

e) None of the above choices is true, or more than one of the above choices is true.

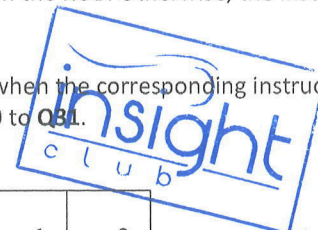
I7 enters issue stage on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- 15, 18, 19
- 16, 19, 20
- 11, 14, 15
- 19, 22, 23

e) None of the above choices is true, or more than one of the above choices is true.

**Part 3:** Repeat Part 2 above but now assuming that the ROB has only two entries. The first entry in the ROB has tag T0, while the second entry has tag T1. An instruction cannot enter the issue stage if there is no empty slot in the ROB. Otherwise, the instruction is reserved a slot its destination register is renamed with the appropriate tag.

Use the table below to write your schedule. In the last column, record the renamed register when the corresponding instruction enters the ROB. The first row in the table has been completed for you. Answer questions Q20 to Q31.



	Time (cycle)				OP	dst	src1	src2	comments
	Decode → Issue (enter issue buffer)	Issued (start to execute)	WB	Committed					
I1	0	1	4	5	LD	T0	R2		F1 renamed T0
I2	1	5	10	11	FMUL	T1	T0	F0	F2 renamed T1
I3	6	11	14	15	FADD	T0	T1	F0	F3 renamed T0
I4	12	13	15	16	ADDI	T1	R2		R2 renamed T1
I5	16	17	20	21	LD	T0	T1		R1 renamed T0
I6	17	21	26	27	FMUL	T1	T0	T0	F2 renamed T1
I7	22	27	30	31	FADD	T0	T1	F3	F2 renamed T0

Latencies: ALU 1c, MEM 3c, FADD 3c, FMUL 5c, WB 1c. No forwarding.

**Q20.** When **Instruction I2** enters the ROB, its registers are renamed as:

- a) FMUL T0, T0, F0.
- b) FMUL T1, T0, F0.
- c) FMUL T1, F1, F0.
- d) FMUL T0, F1, F0.
- e) None of the above choices is true, or more than one of the above choices is true.

**Q21.** I2 enters ROB on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 1, 6, 11, 12
- b) 5, 6, 11, 12
- c) 6, 7, 12, 13
- d) 1, 5, 10, 11
- e) None of the above choices is true, or more than one of the above choices is true.

**Q22.** When **Instruction I3** enters the ROB, its registers are renamed as:

- a) FADD T0, F2, F0.
- b) FADD T1, T1, F0.
- c) FADD T0, T1, F0.
- d) FADD T1, T0, F0.
- e) None of the above choices is true, or more than one of the above choices is true.



**Q23.** I3 enters ROB on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 12, 13, 16, 17
- b) 6, 11, 14, 15
- c) 5, 11, 14, 15
- d) 6, 12, 15, 16
- e) None of the above choices is true, or more than one of the above choices is true.

**Q24.** When **Instruction I4** enters the ROB, its registers are renamed as:

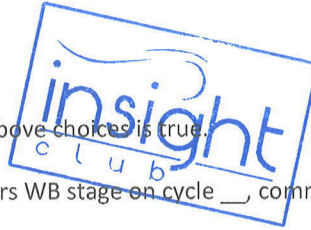
- a) ADDI T1, T1, 8.
- b) ADDI T0, R2, 8.
- c) ADDI T0, T0, 8.
- d) ADDI T1, R2, 8.
- e) None of the above choices is true, or more than one of the above choices is true.

**Q25.** I4 enters ROB on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 12, 13, 15, 16
- b) 12, 13, 14, 16
- c) 12, 13, 14, 15
- d) 16, 17, 18, 19
- e) None of the above choices is true, or more than one of the above choices is true.

**Q26.** When **Instruction I5** enters the ROB, its registers are renamed as:

- a) LD T1, 5(T0).
- b) LD T0, 5(R2).
- c) LD T0, 5(T1).
- d) LD T1, 5(R2).
- e) None of the above choices is true, or more than one of the above choices is true.



**Q27.** **I5** enters ROB on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 17, 18, 21, 22
- b) 16, 17, 20, 21
- c) 13, 17, 20, 21
- d) 13, 14, 17, 20
- e) None of the above choices is true, or more than one of the above choices is true.

**Q28.** When **Instruction I6** enters the ROB, its registers are renamed as:

- a) FMUL T1, T0, T0.
- b) FMUL T0, T1, T1.
- c) FMUL T1, F1, F1.
- d) FMUL T0, F1, F1.
- e) None of the above choices is true, or more than one of the above choices is true.

**Q29.** **I6** enters ROB on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 17, 21, 26, 27
- b) 22, 23, 28, 29
- c) 18, 19, 24, 25
- d) 17, 18, 23, 24
- e) None of the above choices is true, or more than one of the above choices is true.

**Q30.** When **Instruction I7** enters the ROB, its registers are renamed as:

- a) FADD T1, T0, F3.
- b) FADD T1, T0, T0.
- c) FADD T0, F2, F3.
- d) FADD T0, T1, F3.
- e) None of the above choices is true, or more than one of the above choices is true.



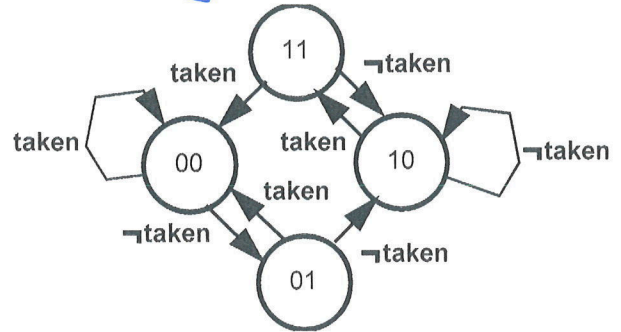
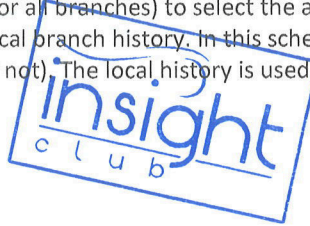
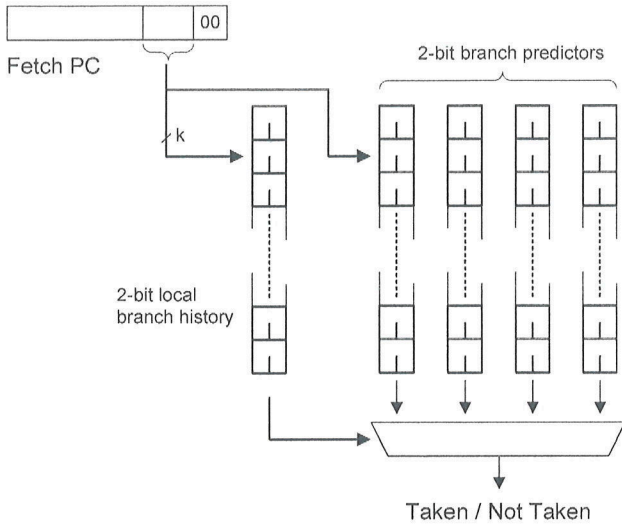
**Q31.** **I7** enters ROB on cycle \_\_, starts to execute on cycle \_\_, enters WB stage on cycle \_\_, commits on cycle \_\_:

- a) 22, 23, 26, 28
- b) 22, 23, 26, 27
- c) 22, 27, 30, 31
- d) 23, 27, 30, 31
- e) None of the above choices is true, or more than one of the above choices is true.



**Problem 3: Branch Prediction (Questions Q32 to Q38)**

In this problem, we study the performance of a 2-level branch prediction scheme when executing a loop (single branch). The prediction scheme is a 2-level predictor shown below. It employs 2-bit branch predictor state machines for local prediction as discussed in class. In state 1X, we predict 'not taken', while in state 0X, we predict taken (see FSM below). However, instead of using a 2-bit shift register for global branch history (common for all branches) to select the appropriate 2-bit predictor to use, a 2-bit shift register is maintained for every branch to act as a 2-bit local branch history. In this scheme, the last two outcomes of each branch are tracked in a two-bit shift register (1 for taken, 0 for not). The local history is used to select which 2-bit predictor to use.



Consider the loop shown below. Assume that every time this loop executes, it ends up taking two iterations. Column 1 in Table 5 below shows the values of register R1 to be used to evaluate the branch instruction on each iteration. The history register for B1 is initialized to '00' (row 1, column 2), while all local branch predictors are initialized to '10' (columns 3 to 6). Column 7 shows the predicted value of the branch while the final column shows that actual outcome of the branch.

```

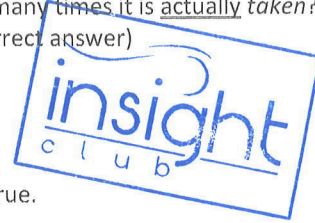
LOOP: LW    R1, 4(R2)
      SUBI  R2, R2, 8
B1:    BNEQZ R1, LOOP
    
```



**Table 5: Branch prediction table. (N: Not taken; T: Taken)**

	System state		Branch Predictor				Behavior	
	R1	History	B1 bits				Predicted	Actual
		way 00	way 01	way 10	way 11			
1	56	00	10	10	10	10	N	T
2	38		11	10	10	10		
3	0							
4	19							
5	22							
6	0							
7	88							
8	96							
9	0							
10	11							
11	12							
12	0							

- Q32.** Start by filling all rows in the History column (column 2) and the final column in Table 5 based on the values of R1 listed in column 1. In updating the History value, the contents of the shift register are shifted to the left, and the new branch outcome value is inserted from the right. Out of the 12 times the branch is executed, how many times it is actually taken? The value of the History register after the branch executes 12 times is \_\_\_\_: (choose the correct answer)
- a) Taken 4 times; value of History register is '11'.
  - b) Taken 8 times; value of History register is '11'.
  - c) Taken 8 times; value of History register is '01'.
  - d) Taken 8 times; value of History register is '10'.
  - e) None of the above choices is true, or more than one of the above choices is true.



Next, fill out columns 3 to 7 in Table 5. The first row has been completed for you. For row 1, the History register in column 2 is '00', so way '00' is selected. The stored prediction bits are '10' (bold). The state machine predicts "not taken", which is recorded in column 7, row 1. Since the branch is actually taken, the prediction bits for way '00' are updated in row 2, column 3 (italic). Fill the remaining rows in the table in this fashion then answer questions **Q33** to **Q38**.

- Q33.** Refer to row 4 in the table. The stored prediction bits for way '01' are \_\_\_\_; The predicted value in column 7 is \_\_\_\_:
- a) '11'; Taken.
  - b) '11'; Not Taken.
  - c) '10'; Taken.
  - d) '10'; Not Taken.
  - e) None of the above choices is true, or more than one of the above choices is true.

- Q34.** Refer to row 7 in the table. The stored prediction bits for way '01' are \_\_\_\_; The predicted value in column 7 is \_\_\_\_:
- a) '00'; Taken.
  - b) '10'; Not Taken.
  - c) '00'; Not Taken.
  - d) '11'; Not Taken.
  - e) None of the above choices is true, or more than one of the above choices is true.

- Q35.** Refer to row 12 in the table. The stored prediction bits for way '10' are \_\_\_\_; The predicted value in column 7 is \_\_\_\_:
- a) '00'; Taken.
  - b) '10'; Not Taken.
  - c) '11'; Not Taken.
  - d) '00'; Not Taken.
  - e) None of the above choices is true, or more than one of the above choices is true.



- Q36.** How many times does the predictor mispredict?
- a) 4 times
  - b) 5 times
  - c) 6 times
  - d) 7 times
  - e) None of the above choices is true, or more than one of the above choices is true.

- Q37.** After training, does the predictor catch the behavior of the loop (i.e. that it runs only two iterations every time it executes)?
- a) Yes
  - b) No
  - c) Can't tell
  - d) It is a coincidence in this example. In other cases, the predictor will not be able to catch the 2-iterations pattern.
  - e) None of the above choices is true, or more than one of the above choices is true.

- Q38.** To achieve high accuracy on loops that run for 3 iterations instead of 2, how would you modify this prediction scheme?
- a) Make the local branch history 3 bits; Keep 4 two-bit local branch predictors.
  - b) Keep the local branch history 2 bits; Include 8 three-bit local branch predictors.
  - c) Make the local branch history 3 bits; Include 8 two-bit local branch predictors.
  - d) Make the local branch history 3 bits; Include 4 three-bit local branch predictors.
  - e) None of the above choices is true, or more than one of the above choices is true.

**Problem 4: General Questions (Questions Q39 to Q43)**

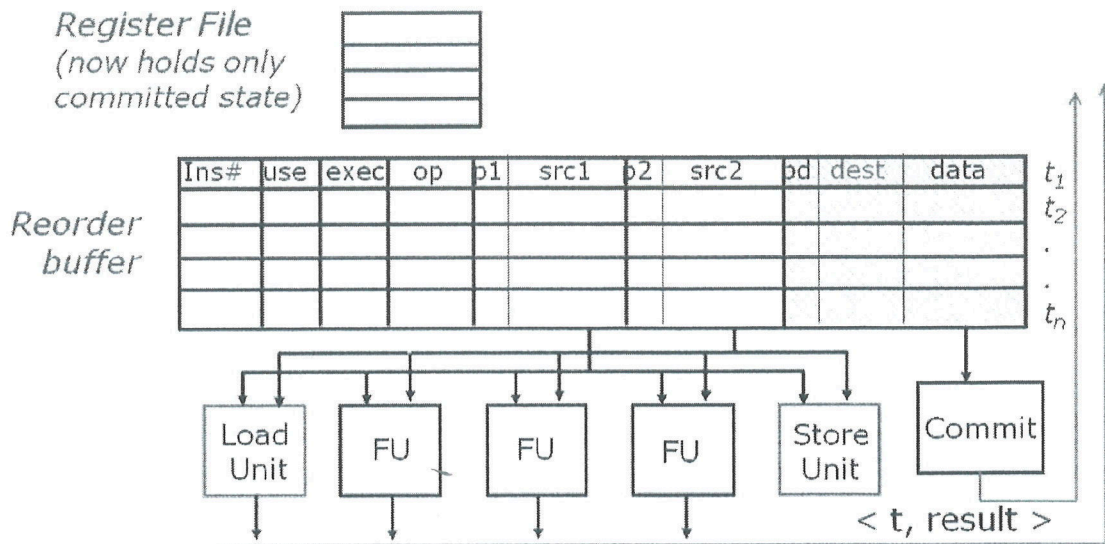


Figure 1: ROB

- Q39.** Consider an out-of-order pipeline that performs speculative execution using data-in-ROB mechanism as shown above. The register file contains only committed data. Assume there is no renaming table. How does the decode stage find the tag of a source register?
- By searching the 'src1', 'src2', and 'dest' fields in the ROB
  - By searching the 'src1' and 'src2' fields in the ROB
  - By searching the 'dest' field in the ROB
  - Consulting the renaming table inside the register file.
  - None of the above choices is true, or more than one of the above choices is true.
- Q40.** Referring to Figure 1, where does the output from the 'Commit' block go?
- To 'src1', 'src2', 'data' fields in ROB
  - To 'data' field in ROB
  - To the register file only
  - To the register file and the 'data' field in ROB
  - None of the above choices is true, or more than one of the above choices is true.
- Q41.** Referring to Figure 1, where does the output from the Load Unit and the Function Unit blocks go?
- To 'src1', 'src2', 'data' fields in ROB
  - To 'data' field in ROB
  - To the register file only
  - To the register file and the 'data' field in ROB
  - None of the above choices is true, or more than one of the above choices is true.
- Q42.** Referring to Figure 1, a renaming table can be added in order to
- Speed up finding the tag of a source register in decode stage; in this case, the 'dest' field is removed from ROB
  - Perform register renaming
  - Speed up finding the tag of a source register in decode stage.
  - Speed up finding the tag of a destination register in decode stage
  - None of the above choices is true, or more than one of the above choices is true.
- Q43.** Referring to Figure 1 and Q42, how does the ROB recover from a branch misprediction?
- By taking snapshots of the entire ROB at each predicted branch; recover earlier ROB snapshot if branch is mispredicted.
  - By taking snapshots of the renaming table at each mispredicted branch; recover earlier snapshot if branch is mispredicted; rollback corresponding entry in ROB right before the mispredicted branch.
  - By clearing the ROB and the renaming table, and restarting execution
  - The 'commit' block always guarantees perfect recovery from mispredicted branches. ROB does not do anything to recover.
  - None of the above choices is true, or more than one of the above choices is true.