Homework 2

Computer 5: Fall 2022

1. In this question, we are going to investigate the difference between single-cycle, multi-cycle and pipelined implementation of the same MIPS architecture in Figure C.21 (5th edition of the text book, page C-34). The latencies associated with various computations in Figure C.21 are given in the following table. Note that for MUX, you have to consider two types delay: data input to output as well as selection input to output.

Instruction memory	Add	Mux	Registers	ALU	Zero?	Data memory	Sign-extend	Other
200ps	100ps	20ps	110ps	120ps	10ps	200ps	20ps	0

Suppose the processor only executes BEQZ.

a) **[10 point]** Suppose FigureC.21 is a single-cycle non-pipelined implementation. What is the minimum cycle length? In your computation show all possible activated data paths to perform BEQZ and select the path with the longest delay.

b) **[10 point]** What is the minimum cycle length if FigureC.21 is a multi-cycle non-pipelined implementation?

c) **[10 point]** What is the minimum cycle length if FigureC.21 is pipelined? Does it outperform (a)? Assume that the processor only executes BEQZ and there is no branch predication.

2. [40 points] For the following instructions:

```
Loop:  L.D      F1, 2048(R1)
       MUL.D    F1, F3, F1
       L.D      F0, 1024(R1)
       SUBIU    R1, R1, #8
       DIV.D    F0, F3, F0
       ADD.D    F5, F1, F0
       MUL.D    F5, F5, F6
       S.D      F5, 1024(R1)
       BNEZ     R1, Loop
```

The third column in the following table shows the number of cycles of latency between a source instruction (first column) and any subsequent instruction (of any type) consuming the result of the source instructions. The third column indicates the number of functional units available for executing the respective type of source instruction.

Function Unit	Related Instruction	Latency Cycles	Number of Units
Integer ALU	L.D S.D SUBIU BNEZ	1	2
Memory Unit	L.D S.D	4	2
FP Adder	ADD.D	2	1
FP Multiplier	MUL.D	6	1
FP Divider	DIV.D	8	1

Assume that the reservation station and the reorder buffer both have infinite size. The integer ALUs are used for effective address calculation for load/store instructions, execution of SUBIU and BNEZ instructions. Assume that you can make at most two writes to CDB in one clock cycle. Create two tables showing when each instruction issues, begins execution, accesses memory and writes its result to the CDB for the first two iterations for the following two scenarios using **Tomasulo's algorithm**.

a) Use a MIPS pipeline with **two-issue** and **without speculation**. Assume that branches are issued alone (single-issue for that time step) and branch prediction is perfect.

b) Use a MIPS pipeline with **two-issue** and **with speculation**. You also need to specify when each instruction commits. Assume that up to two instructions of any type can commit per cycle. Branches are issued alone (single-issue for that time step) and branch prediction is perfect. Note that stores will spend 4 cycles in the commit stage, because its memory access occurs during commit.

(First three rows in each table are given as examples.)

Without speculation

Iter.	Instruction		Issue	Execute	Memory	Write-CDB
1	L.D	F1, 2048(R1)	1	2	3-6	7
1	MUL.D	F1, F3, F1	1	8-13		14
1	L.D	F0, 1024(R1)	2	3	4-7	8
1	SUBIU	R1, R1, #8				
1	DIV.D	F0, F3, F0				
1	ADD.D	F5, F1, F0				
1	MUL.D	F5, F5, F6				
1	S.D	F5, 1024(R1)				
1	BNEZ	R1, Loop				
2	L.D	F1, 2048(R1)				
2	MUL.D	F1, F3, F1				
2	L.D	F0, 1024(R1)				
2	SUBIU	R1, R1, #8				
2	DIV.D	F0, F3, F0				
2	ADD.D	F5, F1, F0				
2	MUL.D	F5, F5, F6				
2	S.D	F5, 1024(R1)				
2	BNEZ	R1, Loop				

With speculation.

Iter.	Instruction		Issue	Execute	Memory	Write-CDB	Commit
1	L.D	F1, 2048(R1)	1	2	3-6	7	8
1	MUL.D	F1, F3, F1	1	8-13		14	15
1	L.D	F0, 1024(R1)	2	3	4-7	8	15
1	SUBIU	R1, R1, #8					
1	DIV.D	F0, F3, F0					
1	ADD.D	F5, F1, F0					
1	MUL.D	F5, F5, F6					
1	S.D	F5, 1024(R1)					
1	BNEZ	R1, Loop					
2	L.D	F1, 2048(R1)					
2	MUL.D	F1, F3, F1					
2	L.D	F0, 1024(R1)					
2	SUBIU	R1, R1, #8					
2	DIV.D	F0, F3, F0					
2	ADD.D	F5, F1, F0					
2	MUL.D	F5, F5, F6					
2	S.D	F5, 1024(R1)					
2	BNEZ	R1, Loop					

3. Accurate branch prediction is crucial to modern processors. Some early studies discovered that random predicators have quite good performance on some benchmarks. A random predictor is a predictor, which randomly gives results with a fixed probability: p for TAKEN and 1-p for NOT TAKEN. Due to the technology limit, we can only physically fabricate one type of random predictor, RP5, which has p=0.5.

a) **[10 points]** Is it possible to make a random predictor with p=0.4 based on RP5? If the latency of RP5 is t, what is the expected latency of your predictor?

b) **[10 points]** Suppose 40% of the branches are taken in a benchmark program. What is the miss prediction ratio of your random predictor with p=0.4 on the benchmark?

c) **[10 points]** Can you design a deterministic predictor that always has a lower mis-prediction rate than your random predictor with p=0.4 on the benchmark?