CS152 Worksheet 6

Q1. Tomasulo's Algorithm

Simulate the execution of the following code using Tomasulo's algorithm. Show the contents of the reservation station entries and the register file for each cycle. Also indicate which instructions are in which pipeline stages of each functional unit.

Assume that at most one instruction can be dispatched per cycle, and two results can be broadcasted over the result bus simultaneously. Both functional units are fully pipelined; the floating-point add latency is 2 cycles, and the multiply latency is 3 cycles. Broadcasting the result to consumers takes another cycle. An instruction can begin execution in the same cycle that is dispatched, assuming all operands are present.

```
A: fmul f1, f0, f2    # produces value V3
B: fadd f0, f3, f1    # produces value V4
C: fmul f3, f2, f3    # produces value V5
D: fadd f3, f3, f1    # produces value V6
```

Cycle 1
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
f0	1	V0
f1		
f2	1	V1
f3	1	V2

Adder

Stage	Instruction
1	
2	

Multiplier

Stage	Instruction
1	
2	
3	

Cycle 2
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
`f0`		
`f1`		
`f2`		
`f3`		

Multiplier

Stage	Instruction
1	
2	
3	

Adder

Stage	Instruction
1	
2	

Cycle 3
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
`f0`		
`f1`		
`f2`		
`f3`		

Multiplier

Stage	Instruction
1	
2	
3	

Adder

Stage	Instruction
1	
2	

Cycle 4
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
`f0`		
`f1`		
`f2`		
`f3`		

Multiplier

Stage	Instruction
1	
2	
3	

Adder

Stage	Instruction
1	
2	

Cycle 5
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
`f0`		
`f1`		
`f2`		
`f3`		

Multiplier

Stage	Instruction
1	
2	
3	

Adder

Stage	Instruction
1	
2	

Cycle 6
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
`f0`		
`f1`		
`f2`		
`f3`		

Multiplier

Stage	Instruction
1	
2	
3	

Adder

Stage	Instruction
1	
2	

Cycle 7
Dispatched instruction:

	p	tag/data	p	tag/data
T0				
T1				
T2				

	p	tag/data	p	tag/data
T3				
T4				

Register File

	p	tag/data
`f0`		
`f1`		
`f2`		
`f3`		

Multiplier

Stage	Instruction
1	
2	
3	

Adder

Stage	Instruction
1	
2	

Q2 Register Renaming

Complete the following table, assuming that the X registers are renamed from the a pool of physical registers. The initial map table and free list is given below. Assume that the free list is organized as a FIFO – a physical register is dequeued from the top of the list when allocated, and a physical register is added back to the bottom of the list when reclaimed. For each instruction, label the following:

- Which physical register is assigned to the instruction as the destination
- On commit, which physical register is added back to the free-list

FreeList:

p2	p4	p10	p7	p1	p0	p9	p8	p13	p14

Architectural Register	Physical Register
x1	p12
x2	p11
x3	p5
x4	p6
x5	p3

Instruction	**Architectural Destination Register**	**Physical Destination Register**	**(To be) Freed Register**
ld x2, 0(x4)	x2	p2	p11
sw x2, 0(x3)			
addi x4, x4, 0x4			
addi x3, x3, 0x4			
bne x4, x1, loop			
ld x2, 0(x4)			
sw x2, 0(x3)			
addi x4, x4, 0x4			
addi x3, x3, 0x4			
bne x4, x1, loop			

Precise exceptions:

Suppose the second ld instruction raised an exception. Show the state of the map table and free list before jumping to the exception handler.