CS GY 6133 Homework 3

1. **[40 Points] Out-of-Order Execution Using Tomasulo's Algorithm**

   The following code sequence is executed on an out-of-order processor that implements Tomasulo's algorithm for floating point instructions. Assume that the micro-architecture has the following features:

   - 1 pipelined FP multiplier with a 4 clock cycle execution latency, and a reservation station with 2 entries.
   - 1 pipelined FP adder with a single clock cycle execution latency, and a reservation station with 2 entries.
   - A floating point register file with 6 registers f0, f1, .., f6. Before the code executes, the registers are initialized as follows:
     - R[f0]=0, R[f1]=1, R[f2]=2, R[f3]=3, R[f4]=4, R[f5]=5.
   - When 2 or more instructions can be dispatched to an FU at the same time, the older instruction is dispatched first.

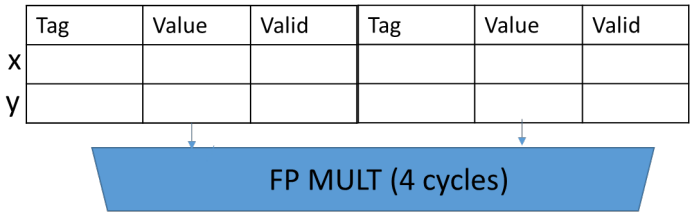
   muld f0, f1, f2
   addd f0, f1, f5
   addd f2, f3, f3
   addd f2, f4, f4
   addd f4, f5, f5

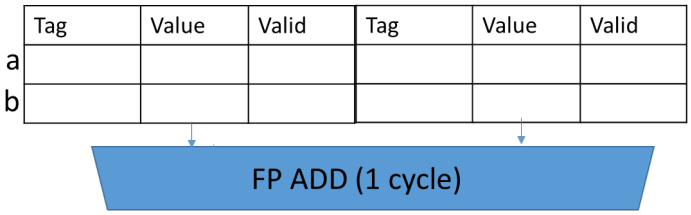
   a. [**10 Points**] Draw a data-flow graph representing the code sequence above.

   b. [**10 Points**] Simulate the execution of the code sequence above cycle-by-cycle. In each clock cycle, indicate the state of the register file and two reservation stations, and which stage of execution each instruction is in. You can use the figure below as a template. Note that an instruction could be in Fetch, Decode/Dispatch, Issue/Execute, and Write-back. An instruction could also be waiting in a reservation station, in which case indicate Waiting.

	Tag	Value	Valid
f0		0	1
f1		1	1
f2		2	1
f3		3	1
f4		4	1
f5		5	1

Instruction	Stage
muld f0, f1, f2	
addd f0, f1, f5	
addd f2, f3, f3	
addd f2, f4, f4	
addd f4, f5, f5	





c. [**10 Points**] What is the minimum number of FP multiply and FP add reservation stations you need to ensure that the code sequence above executes without stalling. (Recall that Tomasulo's algorithm stalls if the reservation station is full.) Show the cycle-by-cycle execution of Tomasulo's with the minimum number of reservation stations that you have determined.

d. [**10 Points**] For this part, assume that the number of reservation stations for both FP multiply and FP add are infinite. However, the FP add reservation stations feeds *M* parallel FP adders, where *M* can be larger than one. What is the smallest value of *M* for which there are no structural hazards in the FP adder (i.e., any FP add instruction whose operands are ready can be immediately dispatched to a FP adder). Show the cycle-by-cycle execution of Tomasulo's with the value of M that you have determined.

2. **[30 Points] Out-of-Order Execution Using Tomasulo's Algorithm + Re-order Buffer (ROB)**

The code sequence below executes on an out-of-order processor with support for precise exceptions using an ROB. Each instruction goes through five stages: Fetch, Decode/Dispatch, Issue/Execute, WB, and Commit. Further, you can assume has the following features:

- 1 pipelined FP multiplier with a 4 clock cycle execution latency, and a reservation station with infinite entries.
- 1 pipelined FP adder with a single clock cycle execution latency, and a reservation station with infinite entries.
- 1 pipelined integer ALU with a single clock cycle execution latency, and a reservation station with infinite entries. The integer ALU also has a comparator for branch instructions.
- An ROB that holds both floating point and integer instructions. The ROB can be assumed to be infinitely sized.
- A floating point register file with 4 registers f0, f1, .., f3. Before the code executes, the registers are initialized as follows:

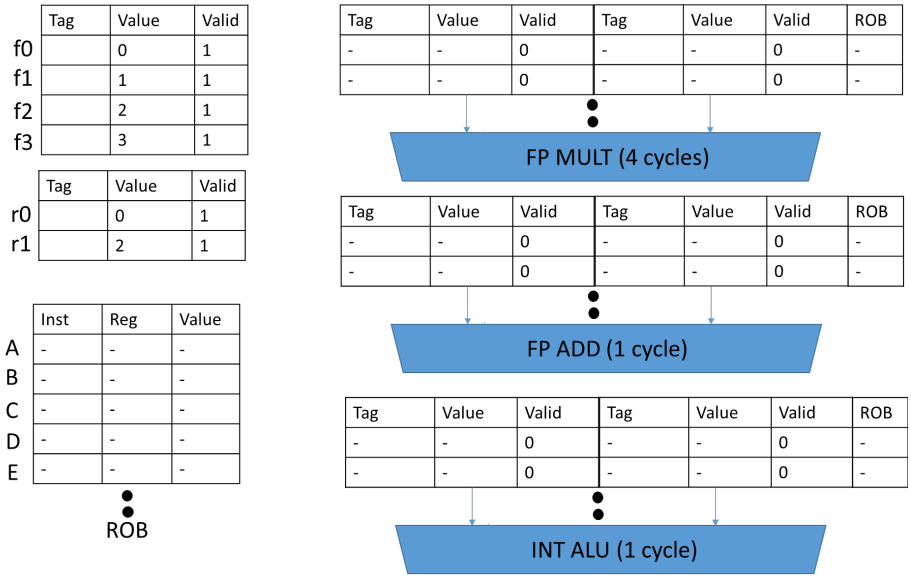
- R[f0]=0, R[f1]=1, R[f2]=2, R[f3]=3.

- An integer register file with 2 registers r0 and r1. The registers are initialized as follows:
  - R[r0] = 0, R[r1] = 2.
- When 2 or more instructions can be dispatched to an FU at the same time, the older instruction is dispatched first.

The processor speculates past branches assuming they are always Taken. The actual result of a branch, i.e., whether it is actually taken or not is communicated in the write-back stage. At this point, if the branch is mis-predicted, the processor quashes speculative state and reverts changes made to the register file by speculative instructions. Doing so takes one clock cycle. After that the processor resumes execution normally.

LOOP: subi r1, r1, 1
muld f0, f1, f1
addd f0, f2, f2
bne r1, r0, LOOP
addd f1,f2,f3

a. **[20 Points]** Simulate the execution of the code above cycle-by-cycle. You can use the template below for reference.



b. **[10 Points]** Determine the minimum size of the ROB required to guarantee that there are no stalls in the execution of the code sequence above due to structural hazards.

3. **[30 Points] Branch Prediction**

Consider the following high-level code consisting of three branches, B1, B2, and B3. The loop counter is set to a ``large" value. In the questions that follow, report the results for LARGE →∞.

```
for (i=0; i<LARGE; i++){ //B1
    if (i%4==0){ //B2
        //do stuff;
    }
    If (i%2 == 0){ //B2
        //do stuff;
    }
}
```

a. **[5 Points]** Determine the mis-prediction rate for branches B1, B2 and B3 assuming last-value prediction. That is, if a branch was Taken the previous time, it is predicted as Taken. Similarly if it is Not Taken previously, it is predicted as Not Taken.

b. **[10 Points]** Determine the mis-prediction rate for branches B1, B2 and B3 assuming that there is a separate 2-bit saturating counter for each branch. Each saturating counter is initialized to the "Strongly Predict Not Taken" state.

c. **[15 Points]** Determine the mis-prediction rate for branches B1, B2 and B3 assuming that a 2-level correlating predictor with a 2-bit global branch history register and a four 2-bit saturating counters for each branch (each of the four counters is for one of the four values of the global branch history register). Each saturating counter is initialized to the "Strongly Predict Not Taken" state.