CS 203 (2019 Fall) Assignment #4

Student ID #:

Name:

1. Consider the following RISC-V instructions:

```
Loop:   LD      F1,0(X3)
        ADD.D   F2,F1,F4
        MUL.D   F1,F2,F6
        ADD.D   F1,F1,F5
        ADD.D   F7,F7,F1
        ADDI    X12,X12,-1
        BNEZ    X12,Loop
        ADDI    X16,X16,4
        LD      X3,0(X16)
```

Assume the initial value of X12 is 2. Please answer the following question.

(1) Assuming that you are given a processor using Tomasulo algorithm for instruction scheduling. The execution time and reservation stations for each functional unit is listed as below:

Functional units	Execution time	Reservation Stations
INT ALU	1	
FP ADD	2	
FP MUL	3	
Memory Access	1	4 for Address Calculation, 3 for loads, 3 for stores
Branch	1	

Please draw the pipeline diagram according to the Tomasulo handout
http://cseweb.ucsd.edu/classes/wi10/cse240a/Slides/tomasulo.pdf
for the given code.

(2) Assuming that you are using an Alpha 21264-Like processor (i.e. register renaming) that is very similar to a real Alpha 21264 except the processor has the same function unit execution time and branch predictor as the previous problem. Please draw the pipeline diagram for the given code. (You may assume the processor has the same number of functional units as the previous problem. Ignore the functional unit cluster problem and assume there are bypass paths for memory and ALU operations).

(3) Assuming R2 is very large, which of the above performs better and why?

2. Consider the following three CPU organizations:

   I. CPU CMP: 2-core superscalar processor with out-of-order issue capabilities. Each core has two functional units. Only a single thread can run on each core at a time.

   II. CPU FMT: A fine-grained multithreaded processor that allows instructions from two threads to be run concurrently on the two functional units, though only instructions from a single thread can be issued to run on any cycle.

   III.CPU SMT: An SMT processor that allows instructions from two threads to be run concurrently on the two functional units, and instructions from either or both threads can be issued to run on any cycle.

   Assume we have two threads A and B to run on these CPUs that include the following operations:

Thread A	Thread B
A1: takes 2 cycle to execute A2: depends on the result of A1 A3: conflicts for a functional unit with A2 (They must use the same one) A4: depends on the result of A2	B1: no dependencies B2: conflicts for a functional unit with B1 (They must use the same one) B3: no dependencies B4: depends on the result of B2

   How many cycles will it take to execute these two threads on each of the above three processors? How many issue slots are wasted due to hazards? (You may assume these processors have full capability of eliminating false dependencies and all the instructions are already in instruction queues. If not specified, the instruction will take one cycle to execute)

3. A simulation study shows that the average branch misprediction rate of selected SPEC92 benchmarks nearly doubles from 5% to 9.1% while going from 1 thread to 8 threads in an SMT processor. However, the wrong-path instructions fetched drops from 24% on a single-threaded processor to 7% on an 8-thread SMT processor.

   (1) Please give the reason why the branch misprediction rate increases.

   (2) Why is the number of fetched wrong-path instructions decreases? (Hint: This is related to the scheduling of instructions) Another simulation study for online transaction-processing workload, OLTP, shows a decrease in instruction cache miss rate from 14% to 9%.

   (3) When is such a decrease, called constructive cache interference, likely to happen?

4. Assume that you are running the following code segment on a symmetric multicore processor, where the inital values of both x and y are 0.

Core 1	Core 2
x = 2; w= x+y+1;	y=2; z=x+y;

(1) Please list all the possible resulting values of w, x, y, z. For each possible outcome, explain how we might arrive at those values. You may need to examine all possible interleaving of instructions. (If you feel that there are too many possibilities, you may assume the cache is coherent.)

(2) If you wanted to make the system sequentially consistent, what are the key constrains you need to impose?

(3) Assume the processor supports "relaxed memory consistency" model, which requires synchronization to be explicitly identified. To achieve "relaxed memory consistency" model, the processor provides a "barrier" instruction, which ensures that all memory operation preceding the barrier instruction complete before any memory operations following the barrier are allowed to begin. Where would you include the barrier instructions in the above code segment to get the final result as x = 2, y = 2, w = 5, and z = 4?