7 Memory Consistency [65 points]

A programmer writes the following two C code segments. She wants to run them concurrently on a multicore processor, called SC, using two different threads, each of which will run on a different core. The processor implements *sequential consistency*, as we discussed in the lecture.

Thread T0		Thread T1	
Instr. T0.0	`a = X[0];`	Instr. T1.0	`Y[0] = 1;`
Instr. T0.1	`b = a + Y[0];`	Instr. T1.1	`*flag = 1;`
Instr. T0.2	`while(*flag == 0);`	Instr. T1.2	`X[1] *= 2;`
Instr. T0.3	`Y[0] += 1;`	Instr. T1.3	`a = 0;`

`X`, `Y`, and `flag` have been allocated in main memory, while `a` and `b` are contained in processor registers. A read or write to any of these variables generates a single memory request. The initial values of all memory locations and variables are 0. Assume each line of the C code segment of a thread is a *single* instruction.

(a) [15 points] What is the final value of `Y[0]` in the SC processor, after both threads finish execution? Explain your answer.

2.

Explanation. `Y[0]` is set equal to 1 by instruction T1.0. Then, it will be incremented by instruction T0.3. The sequential consistency model ensures that the operations of each individual thread are executed in the order specified by its program. Across threads, the ordering is enforced by the use of `flag`. Thread 0 will remain in instruction T0.2 until `flag` is set by T1.1, i.e., after `Y[0]` is initialized. So, instruction T0.3 must be executed after instruction T1.0, causing `Y[0]` to be first set to 1 and then incremented.

(b) [15 points] What is the final value of `b` in the SC processor, after both threads finish execution? Explain your answer.

0 or 1.

Explanation. There are *at least* two possible sequentially-consistent orderings that lead to *at most* two different values of `b` at the end:
Ordering 1: T1.0 → T0.1 - Final value = 1.
Ordering 2: T0.1 → T1.0 - Final value = 0.

With the aim of achieving higher performance, the programmer tests her code on a new multicore processor, called RC, that implements *weak consistency*. As discussed in the lecture, the weak consistency model has no need to guarantee a strict order of memory operations. For this question, consider a very weak model where there is *no* guarantee on the ordering of instructions as seen by different cores.

(c) [15 points] What is the final value of `Y[0]` in the RC processor, after both threads finish execution? Explain your answer.

1 or 2.

Explanation. Since there is no guarantee of a strict order of memory operations, as seen by different cores, instruction T1.1 could complete before or after instruction T1.0, from the perspective of the core that executes T0. If instruction T1.1 completes before instruction T1.0, from the perspective of the core that executes T0, instruction T0.3 could complete before or after instruction T1.0. Thus, there are three possible weakly-consistent orderings that lead to different values of `Y[0]` at the end:
Ordering 1 (from the perspective of T0): T1.0 $\rightarrow$ T1.1 $\rightarrow$ T0.3 - Final value = 2.
Ordering 2 (from the perspective of T0): T1.1 $\rightarrow$ T1.0 $\rightarrow$ T0.3 - Final value = 2.
Ordering 3 (from the perspective of T0): T1.1 $\rightarrow$ T0.3 $\rightarrow$ T1.0 - Final value = 1.