6 Memory Consistency [45 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	`X[0] = 1;`	Instr. T1.0	`X[0] = 0;`
Instr. T0.1	`X[0] += 1;`	Instr. T1.1	`flag[0] = 1;`
Instr. T0.2	`while(flag[0] == 0);`	Instr. T1.2	`b = X[0];`
Instr. T0.3	`a = X[0];`		
Instr. T0.4	`X[0] = a * 2;`		

`X` 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) [10 points] What could be possible final values of `a` in the SC processor, after both threads finish execution? Explain your answer. Provide all possible values.

0, 1, or 2.

Explanation:
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[0]`. Thread 0 will remain in instruction T0.2 until `flag` is set by T1.1. There are *at least* three possible sequentially-consistent orderings that lead to *at most* three different values of `a` at the end:
Ordering 1: T1.0 → T0.0 → T0.1 → T0.3 - Final value: `a` = 2.
Ordering 2: T0.0 → T1.0 → T0.1 → T0.3 - Final value: `a` = 1.
Ordering 3: T0.0 → T0.1 → T1.0 → T0.3 - Final value: `a` = 0.

(b) [10 points] What could be possible final values of `X[0]` in the SC processor, after both threads finish execution? Explain your answer. Provide all possible values.

0, 2, or 4.

Explanation:
The value of `X[0]` is twice the value of `a`:
Ordering 1: T1.0 → T0.0 → T0.1 → T0.3 → T0.4 - Final value: `X[0]` = 4.
Ordering 2: T0.0 → T1.0 → T0.1 → T0.3 → T0.4 - Final value: `X[0]` = 2.
Ordering 3: T0.0 → T0.1 → T1.0 → T0.3 → T0.4 - Final value: `X[0]` = 0.

(c) [10 points] What could be possible final values of `b` in the SC processor, after both threads finish execution? Explain your answer. Provide all possible values.

0, 1, 2, or 4.

Explanation:
Because there are no specific instructions to enforce the execution ordering of T1.2, `b` can have any of the values that `X[0]` can have during the execution of the two threads.

(d) [15 points] The programmer wants `a` and `b` to have the same value at the end of the execution of both threads. The final value of `a` and `b` should be the same value as in the original program (i.e., the possible final values of `a` that you found in part (a)). What *minimal* changes should the programmer make to the program?
(Hint: You can use more flags if necessary.)

She needs two more flags to enforce ordering.

Explanation:
Since the final value should be the same as in the original program, we have to maintain the `flag[0]` in T1.1 and T0.2. Then, `b` should not be updated until `X[0]` has the value that will be stored in `a`. Thus, either before or after T0.3, we need to set a new flag (`flag[1]`) that will be checked by Thread 1 before updating `b`. Finally, we cannot update `X[0]` until `b` has its final value. Thread 1 will set `flag[2]` only after `b` is updated. The modified code will be as follows:

	Thread T0
Instr. T0.0	`X[0] = 1;`
Instr. T0.1	`X[0] += 1;`
Instr. T0.2	`while(flag[0] == 0);`
Instr. T0.3	`a = X[0];`
Instr. T0.4	**flag[1] = 1;**
Instr. T0.5	**while(flag[2] == 0);**
Instr. T0.6	`X[0] = a * 2;`

	Thread T1
Instr. T1.0	`X[0] = 0;`
Instr. T1.1	`flag[0] = 1;`
Instr. T1.2	**while(flag[1] == 0);**
Instr. T1.3	`b = X[0];`
Instr. T1.4	**flag[2] = 1;**