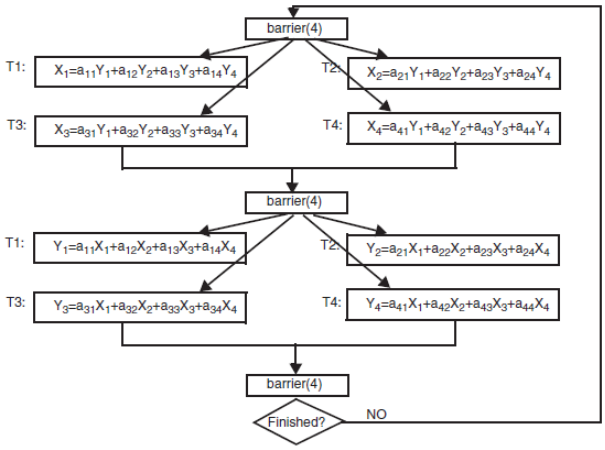
HIGH PERFORMANCE COMPUTER ARCHITECTURE 18-12-2015 **MATRICULATION NO.**_____

(FINAL TEST) **SURNAME**_____

FIRST NAME_____

1) (POINTS 20/40) Consider the four-iterate Jacobi method shown as in the following flowchart.



In this flowchart, during every iteration four threads concurrently compute the new values of a vector X_i as a linear function of Y_i values. Then the threads use a barrier synchronization.
Then the four Y_i values are concurrently computed using the X_i values generated prior to the barrier. The threads then wait again to perform a convergence test. If a specific convergence criterion is met, the program exits; otherwise it loops back to the X_i computation.
(1a) First, transform the flowchart into a pseudocode for a parallel algorithm using a barrier synchronization primitive, similar to the pseudocode shown in Figure 5.2. Assume that at the end of the second barrier the convergence test function is executed as a single thread before returning to the loop or exiting the program.
(1b) Transform the flowchart into an OpenMP parallel program using compiler directives.
(1c) Transform the flowchart into a P-Thread parallel program using the P-Thread API.
(1d) Transform the flowchart into a Cilk parallel program using the cilk_sync, cilk_spawn, cilk_for directives.

2) (points 20/40)
This problem is about the sensitivity of cache misses to actual timing and to the memory consistency model. We use the Jacobi algorithm and its overhead under various cache coherence protocols to demonstrate this point. A flowchart for this algorithm is shown above.
Assume that the caches have infinite size (i.e., no capacity misses and no conflict misses) and that the algorithm has been running for a while (i.e., no cold misses). However, because of the cache coherence protocol, we now have only coherence misses, i.e., misses due to invalidations (the fourth C in the classification) in invalidation-based protocol or updates in update-based protocols. Because matrix A is read-only, accesses to matrix A will not miss at all. We also ignore misses due to barrier synchronization. Hence, in this problem we focus on accesses to X and Y. All the components of X hold in the same cache block, and all the components of Y hold in the same cache block. Because X and Y are shared writable variables, they have been declared as volatile so that they are not allocated in register by the compiler.
The sequence of memory accesses in thread 1 (T1) to Y and X (in process order) is as follows:

rY1, wX1, rY2, wX1, rY3, wX1, rY4, wX1, rX1, wY1, rX2, wY1, rX3, wY1, rX4, wY1, . . . ,

where r means “read” and w means “write.” The sequences of accesses to X and Y by T2, T3, and T4 are similar. We consider three protocols: MSI-invalidate, MSI-update, MESI protocols.

(2a) Assume first that the system is sequentially consistent. Processors run at exactly the same speed and must globally perform their memory accesses one by one so that the four threads interleave their access to X and Y round-robin. For instance, processors each execute their read of Y1 in turn first, then they execute their write to X in turn, etc. What is the number of coherence misses in all processors for one iteration of the entire loop of Jacobi, for MSI, and MESI invalidate protocols? What is the number of updates in MSI-update?
(2b) Repeat (2a) for different timing. We assume that the system is still sequentially consistent and globally performs each access one at a time. However, the global interleaving of accesses is different. The timing is such that, in both the first and second phases of the iteration, threads execute all their memory access in turn. First T1, then T2, then T3, and finally T4. Then in the second phase we have the same behavior: T1, T2, T3, and T4.