

EE w382V: Multicore Computing

Homework 1

Instructor: Professor Vijay Garg (email: garg@ece.utexas.edu)
TA: Wei-Lun Hung (email: wlhung@utexas.edu)

Deadline: Jun. 18th, 2015

The source code must be uploaded through Canvas before the end of the due date (i.e., 11:59pm on Jun. 18th). The assignment should be done in teams of two. You should use the templates downloaded from the course github (<https://github.com/kinmener/UT-Garg-Multicore.git>). You should not change the file names and function signatures. In addition, you should not use package for encapsulation. Please zip and name the source code as [EID1_EID2].zip.

1. **(50 points)** Write a program that uses n threads, which increment a shared counter. The total number of increment operations are m . Each thread reads the value of the counter and increments it m/n times. Implement the following methods and compare the total time taken for each of the following methods for $n = 1..6$ and $m = 1,200,000$. Please include your figure in the zip file. Your figure can be drawn using any of your favorite plotting methods.
 - (a) Lamport's Fast Mutex Algorithm.
 - (b) Bakery Algorithm.
 - (c) Java's synchronized construct
 - (d) Java's Reentrant LockRemember to use `volatile` qualifier for shared variables to guarantee atomicity for parts (a) and (b).
2. **(50 points)** Write a Java class that allows parallel search in an array of integers. It provides the following `static` method:

```
public static int parallelSearch(int x, int[] A, int numThreads)
```

This method creates as many threads as specified by `numThreads`, divides the array `A` into that many parts, and gives each thread a part of the array to search for `x` sequentially. If any thread finds `x`, then it returns an index `i` such that `A[i] = x`. Otherwise, the method returns `-1`. Use a thread pool that creates a fixed number of threads and *Callable* interface for your implementation.