For this assignment you will also turn in a maximum 3-page report on your findings, in PDF format. Your report should be written in the style of a research paper. Please use **full sentences and graphs/tables to convey results**. Describe **interesting and unexpected finding**, using **evidence to conjecture why the system behaves the way it does**. The page limit does not include a cover-page and references, and large figures/graphs. Please describe what you have found, complete with data, interpretations and insights. Also, convey the results as clearly as possible using graphs or tables where appropriate. Further, you must **briefly explain your measurement choices** and justifications. We will evaluate this assignment based on the quality of the report, quality of the implementation and a proper testing methodology.

This report analyses my implementation of three locks: a simple spinlock, an exponential backoff, and a queue lock. Each implementation is recursive so I have included tests to support both the iterative and recursive usage of each of the locks. For instance, for the spinlock, there is a test for mySpinLockTAS (iterative test method using recursive lock) and myRecursiveSpinLockTAS (recursive test method using recursive lock). All tests are performed in an environment with multicore processors.

My main priority was ensuring the results are correct. For each of the tests, the iterative and recursive test methods’ expected total count is equal to numItterations \* numThreads. For the rest of this report, when I refer to parameters, I am talking about the variables under test, which includes threads, number of iterations, operations outside critical section and operations inside critical section. The numItterations variable is fixed at 130000 initially to avoid segmentation fault when using the recursive test methods. This is due to the stack overlapping with the heap.

The measurements are taken based on a few considerations. I want to understand how all locks compare relatively to each other given fixed parameters. For each of the parameters, I will increase it one at a time while leaving all other parameters fixed at the default value and perform an analysis on the behavior. The default values are set to (4, 130000, 0, 1) for number of threads, number of iterations, work outside critical section and work inside critical section respectively.

A comparison is made between all test methods when parameters are fixed at a certain default value. Figure 1 shows that regardless of which lock is used (spinlock/mutex locks), a recursive approach take much less time to increment a counter than an iterative one. This is due to caching of instructions for that function. MySpinlock TTAS performs better than MySpinlock TAS as a value does not need to be fetched from main memory every time. Instead, the value can be read directly from the cache. However, MySpinlock TTAS does not perform as well as MyMutex TTAS because there is still contention for the lock. The problem is that after the lock is released, there is an invalidation storm. Upon all of those threads’ next access to their cache, they will encounter a cache miss. This results in higher traffic on the bus as each of the threads attempt to acquire the new value from main memory. MyMutex TTAS implements a backoff strategy, which attempts to avoid the prior situation of bus contention from happening, as can be seen from the results. The threads are backed off for a random period of time before trying to acquire the lock again. Thus, MyMutex TTAS has a much smaller test time than the other locks.

[Figure 1]