Mihai Pocse

09/13/18

CIS656

**Lab 1**

Below: Tabulated results of program execution on Task A & Task B

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Number of Threads** | **Elapsed Time in Milliseconds** | **Final Counter** |
| A | 2 | 37 | 1024438 |
| A | 4 | 27 | 1029652 |
| A | 8 | 11 | 2032966 |
| A | 16 | 4 | 8531017 |
| A | 32 | 13 | 21119395 |
| B | 2 | 346 | 2000000 |
| B | 4 | 521 | 4000000 |
| B | 8 | 900 | 8000000 |
| B | 16 | 1799 | 16000000 |
| B | 32 | 4031 | 36000000 |

1. & 2. Is the counter always equal to n \* 1000000, where n is the number of threads created?  Explain why, or why not.

In the synchronized multithreading results of Task B, the n \* 1000000 value for the counter stands true and consistent in its outcomes. In Task A, the results are quite different due to the lack of synchronization causing an inaccuracy in the final counters results as the threads conflict with each other as they attempt to utilize the same resource (the counter) concurrently.

3. Analyze the differences in elapsed time between Program A and Program B.  Is there a significant difference? Explain why or why not.

The elapsed time between the two programs is significantly different, with Task A taking significantly less time than Task B to execute. Additionally, the times outputted for each run of Task A as the thread counts increased seemed to have similar or random values of elapsed time between runs regardless of the number of threads, with the overall execution times for each run being significantly less than anything outputted by Task B, whereas Task B has a consistent increase in the elapsed time in milliseconds as the number of threads increases due to the added wait time for thread synchronization to occur.