**CS/ECE 758: PROGRAMMING MULTICORE PROCESSORS
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON**

Prof. Mark D. Hill
Assistant Jason Power

Examination
In-Class
Monday, October 29, 2012
Weight: 25%

 1:15 minutes.

**CLOSED BOOK**, etc., but one cheat sheet allowed (two-sided 8.5x11 page).

The exam is two-sided and has **EIGHT** pages, including two blank pages at the end.

Plan your time carefully, since some problems are longer than others.

NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **ProblemNumber** | **MaximumPoints** | **ActualPoints** |
| 1 | 12 |  |
| 2 | 12 |  |
| 3 | 12 |  |
| 4 | 12 |  |
| 5 | 12 |  |
| Total | 60 |  |

**Problem 1: Math, etc. (12 points)**

* *(3 points)* Consider a job that is 1/8 sequential and 7/8 perfectly parallel. What is the execution time speed up from moving from 1 processor core to (i) 4, (ii) 8, or (iii) an infinite number of cores?
* *(4 points)* Consider a CILK-like job that executes one task that spawns four tasks that each spawn three tasks and then magically completes. Assume that each task takes five time units. Write expressions for the job’s execution time on (i) 1 core, (ii) 6 cores, (iii) 12 cores, or (iv) an infinite number of cores.
* *(5 points)* For many practical problems using linear systems of equations, most matrix values are zero. Give an example of one way to represent such a *sparse matrix* wherein zero-valued elements are not explicitly stored.

**Problem 2: Locks (12 points)**

Recall that the synchronization primitive compare and swap (CAS) stores a new value in an address “a” if and only if “a” is equal to an old value:

Boolean CAS(word \*a, word old, word new):
atomic{
 t:= (\*a == old);
 if (t) \*a := new;
 return t;
}

Assume that 0 means “unlocked” and non-0 means “locked.”

(a) Using CAS and simple loads, stores, etc., implement a very simple lock(word \*L) subroutine that returns only when lock L is obtained.

* Implement unlock(word \*L).
* Discuss ways to improve the performance of lock(word \*L) vs. your answer to part (a).

**Problem 3: GPUs (12 points)**

The *CUDA programming model* seeks to enable general-purpose programming on GPUs, such as the NVidia GeForce 8800.

* Describe the *advantages* of CUDA/GPU programming relative to writing programs for the host (x86) processor(s).
* Describe the *disadvantages* of CUDA/GPU programming relative to writing programs for the host (x86) processor(s).
* List and briefly describe four different optimizations that that can improve the performance of general-purpose GPU programs.

**Problem 4: Transaction Memory, etc. (12 points)**

* Programming with locks can *deadlock* while programming with transaction memory cannot. Why? Consider presenting the conditions necessary for deadlock.

* Some hardware transaction memory systems allow only *bounded* transactions, while others enable *unbounded* transactions. Which is easier to program? Which is easier to implement in hardware? Why?

* For database transactions, what are the key ideas in *Bayer and Schkolnick’s “Concurrency of Operations on B-Trees”?*

**Problem 5: Reader (12 points)**

* *(4 points)* How does the *Sun Niagara* chip (Kongetira, et al.) differ from most multicore chips today, such as those you used in homework assignments 1-4?

*(b) (4 points)* What are the key arguments of *Edward Lee’s “The Problem with Threads”?*

* *(4 points)* What are the key ideas of *Gagan Gupta and Gurindar S. Sohi’s, “Dataflow Execution of Sequential Imperative Programs on Multicore Architectures”?*

**Scratch Sheet 1 of 2 (in case you need additional space for some of your answers)**

**Scratch Sheet 2 of 2 (in case you need additional space for some of your answers)**