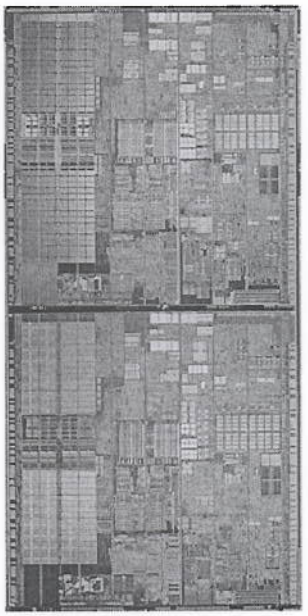
ECE 5984
Multiprocessor Programming

Course
Syllabus



Instructor:
Dr. Binoy Ravindran
Associate Professor
ECE Dept., Virginia Tech
Office: 2040-C Torgersen Hall
Blacksburg, VA 24061
540-231-3777, binoy@vt.edu
http://www.ece.vt.edu/faculty/ravindran.html
http://www.real-time.ece.vt.edu/

Course Objectives:

The computer industry is undergoing a paradigm shift, as chip manufacturers are increasingly investing in, and manufacturing a new generation of multi-processor chips called multicores, as it is becoming increasingly difficult to increase clock speeds by packing more transistors in the same chip without increasing power consumption and overheating. Consequently, application software performance can no longer be improved by simply relying on increased clock speeds of single processors; software must explicitly be written to exploit the hardware parallelism of multiprocessors. Programming multi-processors is fundamentally different from programming single-processors, due to the need to understand how concurrent computations on separate processors coordinate with one another, in contrast with understanding how concurrent computations on the same processor coordinate with one another. This is a complex and intricate problem, and requires new abstractions, algorithms, and programming tools.

The course will approach multiprocessor programming from two complimentary directions: *principles* and *practice*. In the principles part, the course will illustrate how to reason about concurrency by discussing the classical mutual exclusion problem, correctness properties (e.g., fairness, deadlock, linearizability), properties of shared memory (e.g., register constructions, atomic snapshots), and synchronization primitives for implementing concurrent data structures, ranging from simple ones (e.g., consensus protocols) to powerful universal constructions (e.g., universality of consensus). The course will illustrate the practice of multiprocessor programming through programming patterns such as spin locks, monitor locks, the work-stealing paradigm, and barriers. Concurrent data structures (e.g., concurrent linked lists, queues, stacks, hash maps, skiplists) will be discussed through synchronization patterns ranging from coarse-grained locking to fine-grained locking to lock-free structures, atomic synchronization primitives, elimination, parallel search, and transactional memory.

Upon completion of the course, the student should be able to:

- Demonstrate fundamental principles of multiprocessor programming including:
  - concurrency correctness properties such as consistency, linearizability, progress, fairness, and deadlock-freedom;
  - properties of shared memory such as register constructions and atomic snapshots; and
  - synchronization primitives for implementing concurrent data structures, ranging from simple ones (e.g., atomic registers, consensus protocols, FIFO queues) to powerful universal constructions (e.g., universality of consensus).
- Write multiprocessor programs using:
  - programming patterns including spin locks and contention, monitor locks and waiting, work-stealing and parallelism, and barriers;
  - concurrent data structures including concurrent linked lists, concurrent queues, concurrent stacks, concurrent hash maps, and concurrent skiplists;
  - synchronization patterns including coarse-grained locking, fine-grained locking, optimistic locking, lazy synchronization, non-blocking synchronization, atomic synchronization primitives, elimination, parallel search; and
  - transactional memory abstractions including software transactional memory.
- Establish properties of multiprocessor programs including their correctness, fairness, consistency, linearizability, progress, and deadlock-freedom.

Prerequisites:

Graduate Standing. Students are assumed to have taken an operating systems course during their undergraduate curriculum (students without an OS course should discuss with instructor). Knowledge of concurrent programming in Java, or in C++ (augmented with a threads package) is assumed.

Required Text:

The Art of Multiprocessor Programming, Maurice Herlihy and Nir Shavit, paperback, 528 pages, ISBN-13: 978-0-12-370591-4, ISBN-10: 0-12-370591-6, Morgan Kauffman, February 2008

Papers from the literature will be made available on the course website.

Software packages that implement synchronization abstractions and algorithms discussed in the course will be made available on the course website.

Grading (tentative):

Homeworks	20%	(five to six homeworks)
Projects	50%	(two projects)
Midterm Exam	15%	
Final Exam	15%	

Course Topic Outline (tentative):

Topic	*Percentage*
Introduction to shared objects and synchronization, and the mutual exclusion problem	5
Correctness properties of concurrent programs (consistency, linearizability, progress, fairness, deadlock-freedom)	10
Foundations of shared memory (register constructions, atomic snapshots)	10
Synchronization operations for concurrent data structures (atomic registers, consensus protocols, FIFO queues, universality of consensus)	15
Programming patterns: Spin locks and contention, monitor locks and waiting, work-stealing and parallelism, barriers	10
Concurrent data structures (concurrent linked lists, concurrent queues, concurrent stacks, concurrent hash maps, concurrent skiplists)	20
Synchronization patterns: coarse-grained locking, fine-grained locking, optimistic locking, lazy synchronization, non-blocking synchronization	15
Atomic synchronization primitives, elimination, parallel search	10
Transactional memory	5