Test 1 will be Thursday, Oct. 11 in class and cover chapters 1 through 5. It will be closed-book and notes, except for one 8.5" x 11" sheet of paper (you can use front and back) containing any notes that you want. The test will cover the following topics (and maybe more).

Chapter 1. Introduction

Sequential Computer: fetch-execute cycle of stored-program
Moore's law and architectural improvements (cache, instruction pipelining and RISC, superscalar, multi-core)
Problems that delay/*stall* the pipeline: structural hazards, data hazards, control hazards
"Paradigm Shift" of Sequential vs. Parallel programs: pair-wise (log n) summation, parallel-prefix sum
Count 3s example concepts:

- mutual exclusion of a critical section using a mutex
- lock contention vs. local computation
- cache effects: cache coherence by cache line, false sharing

Characteristics of a "good" parallel program: correctness, performance, scalability, performance portability

Chapter 2. Understanding Parallel Computers

Diversity of parallel computers: "Six" parallel computers

- chip multiprocessor - cache coherency protocols Intel MESI vs. AMD64 MOESI
- Symmetric Multiprocessor (SMP) - snoopy caches, Sun Fire E25k characteristics
- Heterogeneous Chip Designs **-** standard processor with more specialized attached processors. Examples: GPUs, Field programmable gate arrays (FPGAs), and game cell processor
- Clusters - commodity parts: nodes connected by networking form (Gigabit ethernet, Myrinet, Infiniband, etc.) Example: HP Cluster Platform 6000 Blade server
- Supercomputer - characteristics: fancy interconnection networks and specially designed hardware Example: IBM BlueGene/L characteristics

shared address space (shared memory machines) vs. distributed address space (distributed memory machines)
Flynn' Taxonomy
Ways to abstract away from details to support performance portability: PRAM and CTA abstract models
PRAM model characteristics and limitations
CTA model characteristics: Locality Rule
Three major communication mechanisms presented to programmers (shared memory, one-sided communication, and message passing)
Types of interconnection networks: 2D and 3D torus, hypercube, fat trees, cross-bar, omega network

Chapter 3. Reasoning about Performance

Parallelism vs. Performance
Threads vs. Processes
Latency and Throughput
Categories of performance loss:

- Parallelization overheads: communication, synchronization, computation, memory
- Non-parallelizable code - Amdahl's law
- Idle processors - load balancing, memory-bound computations
- Contention for shared resources

Parallel Structure consideration for avoiding parallel performance loss:

- dependence - data dependences: flow dependence: read-after-write (RAW), anti-dependence: write-after-read (WAR), and output dependence: write-after-write (WAW); renaming usage
- granularity - frequency of interaction among the threads/processes (fine vs. course)
- locality - temporal locality and spatial locality

Performance Tradeoffs: Communication vs. Computation, Memory vs. Parallelism, Overhead vs. Parallelism

Chapter 4. First Steps Toward Parallel Programming
Broad classes of parallel computations: data parallel vs. task parallel
Peril-L notation: goals, syntax and semantics of statements, memory model
Formulating Parallelism: fixed parallelism, unlimited parallelism, scalable parallelism; Alphabetizing example

Chapter 5. Scalable Algorithmic Techniques
Blocks of Independent Computation - minimize dependencies and overheads
Scwartz' algorithm - application this guiding principle to tree operation
Generalization of the Reduce and Scan abstraction
Static allocations of data and work to processes: 1D and 2D block, cyclic, and block-cyclic allocations
Overhead issues: load balancing, communication, etc.
Dynamic allocation of work to processes via a work queue - issues (granularity, contention, etc), trees