Massachusetts Institute of Technology
6.892: Theory of Parallel Systems
Professor Charles E. Leiserson

Handout 20
March 13, 1997

Term Project

The term project will count for 40% of your final grade and should represent effort on the order of 2 labs and 1 problem set per person. We suggest that you work in groups of one or two people. We expect the total amount of effort to scale up with the number of people involved in a group.

Term projects must be completed by the end of the term, and you will be asked to give a presentation of your work in the last few lectures of the class. Before starting the project, you must submit a project proposal by **April 3, 1997** (the first Thursday after Spring Break). Within a week or so, we will give you feedback on your proposal. We recommend that you briefly talk over ideas with Charles or Matteo before submitting your proposal.

Projects should involve programming or algorithms, and ideally, it should involve both, but that is not necessary if the content is otherwise strong. For example, if you write a program, you might try to analyze it algorithmically. If you develop and analyze an algorithm, you might code up a simple version, or at least indicate in your write-up how it might be implemented. Synthesis of theory and systems will be rewarded.

Your project proposal should include the following:

- Title of the project.
- People involved.
- Description of the problem to be solved.
- Description of your proposed approach to the problem.
- Evidence for why you believe you will be successful doing your project in the time allocated.

Project ideas

The following is a list of projects that you are free to do, or they may spark your own creativity. If you need more information on any of these projects, please talk to Charles or Matteo.

- Parallelize an existing C application.
- Parallelize Johnson's algorithm for all-pairs shortest paths or Goldberg's max-flow algorithm.
- Implement NESL on top of Cilk by building a data-parallel library for the VCODE interpreter.
- Implement a parallel "bignum" library.
- Implement Adaptive Bitonic Sort in Cilk (this is a variant of Bitonic Sort that requires $O(n \lg n)$ work, but cannot be implemented as a fixed-connection network).
- Implement a parallel QR-factorization or a parallel least-squares algorithm.
- Implement parallel primality testing, prime sieve, or factoring.
- Parallelize a heuristic for the traveling salesman problem or for graph partitioning.
- Implement a parallel algorithm for linear programming.
- Break RSA, or do an RSA challenge in Cilk (see `http://www.rsa.com/`).
- Devise a better parallelization of the MTD(f) minimax search algorithm.
- Implement a parallel convex-hull algorithm or other geometric algorithm.
- Find a good (i.e.: fast) way to compute $A^T A$ in Cilk, where A is a nonsquare matrix. Such a program will have an immediate application in computational physics.
- Parallelize one of the `matlab` toolboxes.
- Implement a more efficient parallel `malloc` for Cilk.
- Implement a "space determinator," like the Nondeterminator, which from a serial execution predicts the amount of space used by a P-processor execution of a program on a given input.