

**CS 4504 Computer Organization**  
**Prof. Kirk W. Cameron**  
**Spring 2007 Project 2: Speedup of Numerical Applications**

**Important Dates**

Assigned: April 3, 2007

Official due date: April 17, 2007

Extended due date: April 19, 2007 (-10 points for late submission)

**Summary**

In this project you are expected to speedup a naïve implementation of Matrix Multiply available as C code at the class home page. You will produce a report (< 6 pages) describing your approach and results. You may attempt additional work for extra credit (adding 2 or 3 additional pages to the report).

**Details**

You are to optimize two versions of the naive matrix multiply (matmul.c). (Tips: When you do optimization, you can use the methods we have learned, for example: blocking, prefetching, etc. In this process, you may use lmbench to get the L1 and L2 cache sizes or block sizes for analyzing the solving method)

**Version 1:** Calculates  $C = A \times B$ . A is a general  $m \times k$  matrix, B is a general  $k \times n$  matrix. For simplicity you may assume  $m = k = n$  or square matrices in this version. Compare your “hand optimized version” to the compiler’s version with optimizations turned off (the base case) and full optimizations turned on. See if you can beat the compiler through hand tuning.

**Version 2:** Modify the matrix multiply Version 1 optimized for non-square matrices

**Extra-credit (two options):**

- a. Create a naïve matrix transpose code that takes a general matrix with  $m \times k$  entries of doubles and transposes it in memory to a  $k \times m$  matrix. Use the un-optimized version as the base case and see if you can optimize the transpose to beat the compiler.
- b. Optimizing a Stencil Kernel:  
Contact the TA for access to the stencil kernel code. This code performs stencil operations in 3 dimensions. Your mission is to use optimization techniques to beat the compiler. You should also compare and contrast algorithmic optimizations with tuning optimizations in this problem.

**Experimentation**

You should compile and run the naïve Matrix Multiply with **compiler optimizations turned OFF**. How to do this will be explained in the documentation of your compiler (This typically involves specifying a command line option when invoking the compiler<sup>1</sup>.)

---

<sup>1</sup> Yes, this is possible in ALL compilers worth their salt. For example, in MS Visual C++ 6.0, see the pop up menu for Project-->Settings under Optimizations.

This is the base performance of your system for matrix multiply. Your assignment is to speed up the performance of this code by changing the algorithm implementation, thus circumventing the optimizations of the compiler. Discussions in Chapter 5 (3<sup>rd</sup> edition) of your text provide a good starting point for such techniques. You will have to use timing functions native to your platform embedded in your code to measure the performance. An example is the `getrusage()` function in Unix environments. Hardware counters can also be used. All platforms will provide some similar mechanism. You will be graded on the speedup achieved in relation to your peers and the quality of the short report describing your results.

**Required sections of paper:**

**< Failure to follow these guidelines will result in points taken off. >**

1. *Title and author info:* Your name, affiliation, address, email, phone, etc. should appear towards the top of the first page of your report. I suggest 14-point font for title, and 12-point font for the rest of your document. Document should be singlespaced, single column, 12 point font. Margins should be 1 inch from all sides. Use numeric headings and subheadings as necessary. If English is not your first language, I suggest you utilize on-campus editorial resources or your peers to ensure correct grammar. This page does not count against the 6 page limitation.

2. *Abstract:* On the first page of your document, provide an abstract that summarizes your work, approach, and findings. This should include exact numbers for your achieved speedup. This should provide the reader with an overview of your contribution without having to read the paper. This should be no longer than 200 words.

3. *Related Work:* If you utilized algorithms or approaches that you did not invent, you should cite them accordingly in your paper. This is to be your own work, where you apply optimizations by hand to the code given by the instructor. You may use other algorithms and techniques provided you understand and can explain how they work, and you implement the code yourself. Here you should simply identify others' work and discuss its relationship to your optimized matrix multiply. This section should be less than 300 words. I will use electronic means to compare your code to others in the class and the work of others. Cheating in any form will result in an F on the project and probable expulsion from the class. Save the particulars of your approach for the next section.

4. *Methodology:* This section should describe exactly the method you used to speedup the application. You should provide a step-by-step accounting of your approach. Why you did what you did, and how it might be different or the same from previous approaches. You should discuss your methods for attempting to speed up the code given and how you believe they will affect the code generally. Save exact numbers for the next section. Be succinct. Keep this section under 500 words.

5. *Results:* Here you should provide the empirical timing measurements for the matrix multiply you implemented verse the base version(s). You should summarize the details of the system upon which implementation took place (architecture, platform) and the timing

methods you used (and why you chose both arch and timing method). You should provide charts and tables as necessary to show your results and discuss the results in the context of your methodology from the last section. This section should contain mostly graphs (like those found at the end of this document) and some analysis of what worked, why you think it worked, and what didn't work.

6. *References*: IEEE Journal citation format should be followed. You must cite all references. Plagiarism is unacceptable. Formatting details can be found at:  
[http://www.ece.uiuc.edu/pubs/ref\\_guides/ieee.html](http://www.ece.uiuc.edu/pubs/ref_guides/ieee.html)