CS 392: Homework Assignment 4
Due: November 18, 11:55pm

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems or programming. Use of the Internet is allowed, but should not include searching for existing solutions.

Under absolutely no circumstances code can be exchanged between students. If some code was shown in class, it can be used, but it must be obtained from Canvas, the instructor or the TA.

Assignments from previous offerings of the course must not be re-used. Violations will be penalized appropriately.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prohibit you from submitting a homework assignment in time, please e-mail me.

Problem 1. Simple Matrix Transposition (60 points) Create a simple a program that transposes a large, square matrix. The objective is to exploit cache memory to accelerate the operation. To achieve locality, the matrix should be transposed in square blocks, instead of looping over the elements of the matrix one by one. For example, each 4×4 block of the input can be selected and transposed to form the output. An example of transposition can be seen below.

$$\underset{\text{Input}}{\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}} \qquad \underset{\text{Output}}{\begin{bmatrix}1&4&7\\2&5&8\\3&6&9\end{bmatrix}}$$

The matrix to be transposed will be square and its width will be a power of 2. The width of the block will also be a power of 2, so there will be no need to check boundary conditions. The program should take two arguments: the width of the input matrix and the width of the block. There is no need to verify that they are powers of 2.

The input and output matrices can be allocated as 1D arrays by calling malloc() to allocate space for $N\times$ floats, where N is the width of the matrix. Then, the element at row i and column j can be accessed using `A[i*N + j]`. The input matrix should be initialized randomly using rand().

Try different values of the block size and record the time it takes to transpose 2048×2048, 4096×4096 and 8192×8192 matrices. Use the gettimeofday() function to record the start and end time of the transposition only, excluding the initialization. See slide 68 in the third part of the notes.

THINK IF THERE IS MORE THAN ONE WAY TO ACHIEVE LOCALITY.

Requirements.

1. The main file should be named transp.c.
2. No makefile or header files are required for this homework.
3. The matrix to be transposed will be square and its width will be a power of 2. The width of the block will also be a power of 2, so there will be no need to check boundary conditions.
4. The code should be timed using both the second and microsecond fields of the `timeval` structs returned by gettimeofday(). Report the best block width for each size of the input matrix and a few values of the block width around the best value. Either a table or a graph is acceptable for this.
5. Explain how the code becomes faster due to cache hits.

Problem 2. Arbitrary Matrix Transposition (20 points) Write a separate program for transposing matrices as above, but no longer assume that the matrices are square or that their dimensions are powers of 2. (The blocks will still be square and their width will be a power of 2.

This program should take three arguments: the height and width of the input matrix and the width of the block.

Use if statements to verify that all memory accesses are legal. Accessing the 500^{th} element of a row with 499 elements is wrong, even though it still falls within the memory allocated for the matrix.

Requirements.

1. The main file should be named transpif.c.
2. No makefile or header files are required for this homework.
3. The matrix to be transposed will not be necessarily square and can be of arbitrary dimensions that fit in memory. The width of the block will still be a power of 2. Boundary conditions must now be checked.
4. The code should be timed using both the second and microsecond fields of the `timeval` structs returned by gettimeofday(). Report the best block width for each size of the input matrix of Problem 1 and a few values of the block width around the best value. Either a table or a graph is acceptable for this.

Problem 3. Branch and Merge on GitHub (20 points) Create a repository on GitHub and push your code. Create a branch and then merge it with master. Then, include a link to the **commits** page of the repository. See for example `https://github.com/tensorflow/tensorflow/commits/master`. Links to other parts of the repository will NOT receive full credit.

The use of GitHub is strongly recommended, but not mandatory if you are already using a similar platform. Submit a link that shows the branch and merge operations on the website you were already using.

Deliverables. A zip/tar/gz file containing:

1. Source files.
2. A pdf file showing the timing results you obtained in Problems 1 and 2.
3. A link (included in the pdf file) of your commit history on GitHub as described in Problem 3.