PHYS405 - Advanced Computational Physics
Assignment #9
Due: Friday, December 11

Purpose: Learn the CUDA language.
Note: Please identify all your work.

Part A

This assignment consists in multiplying two square matrices of identical size.

$$P = MxN \tag{1}$$

The matrices **M** and **N**, of size *MatrixSize*, could be filled with random numbers.

Step 1

Write a program to do the matrix multiplication assuming that the matrices **M** and **N** are small and fit in a CUDA block. Input the matrix size via a command line argument. Do the matrix multiplication on the GPU *and* on the CPU and compare the resulting matrices. Make sure that your code works for arbitrary block size (up to 512 threads) and (small) matrix size.

Use one-dimensional arrays to store the matrices **M**, **N** and **P** for efficiency.

Step 2

Modify the previous program to multiply arbitrary size matrices. Make sure that your code works for arbitrary block size (up to 512 threads) and matrix size (up to memory limitation). Instrument your program with calls to *gettimeofday()* to time the matrix multiplication on the CPU and GPU. Plot these times as a function of matrix size (up to large matrices, 4096) and guess the matrix size dependence of the timing.

Step 3

Optimize the previous code to take advantage of the very fast share memory. To do this you must tile the matrix via 2-D CUDA grid of blocks (as above). All matrix elements in a block within **P** will be computed at once. The scalar product of each row of **M** and each column of **N** within the block can be calculated by scanning over the matrices in block size tiles. The content of **M** and **N** within the tiles can then be transfered into the share memory for speed.

References: Book in draft form by David Kirk (NIDIA) and Wen-Mei Hwu (U. of Illinis) for their course ECE 498 (see NVIDIA web page). The link is quoted on the assignment web page. A skeleton code is also provided. See also the in-class exercise on array reversal.