# Lab 3: Tiled Matrix Multiplication

Due Date: …

## 1. Objective

The purpose of this lab is to get you familiar with using shared memory to write optimized kernel algorithms by implementing a “tiled” version of matrix multiplication.

## 2. Procedure

**Step 1:** …

**Step 2:** Edit <lab-directory>/main.cu and <lab-directory>/kernel.cu to include the host setup code and device kernel code where indicated.

**Step 3:** Compile and test your code.

cd <lab-directory>

make

./sgemm-tiled # Uses the default matrix sizes

./sgemm-tiled <m> # Uses square m x m matrices

./sgemm-tiled <m> <k> <n> # Uses (m x k) and (k x n) input matrices

Your code is expected to work for varying input dimensions – which may or may not be divisible by your tile size. It is a good idea to test and debug initially with examples where the matrix size is divisible by the tile size, and then try the boundary cases.

**Step 4:** Answer the following question in a new file named <lab-directory>/answers.txt:

* In your kernel implementation, how many threads can be simultaneously scheduled for execution on a GeForce GTX 280 GPU, which contains 30 streaming multiprocessors? Use:

nvcc --ptxas-options="-v" kernel.cu

to see the resource usage of your kernel (although compilation will fail, it will only do so after compiling the kernel and displaying the relevant information). Show your work.

**Step 5:** Submit your assignment. You should only submit the following files:

* main.cu
* kernel.cu
* answers.txt

…

**3. Grading:**

Your submission will be graded based on the following criteria.

* Functionality/knowledge: XX%
  + Correct code and output results
  + Correct usage of shared memory in the kernel to hide global memory access latencies
  + Correct handling of boundary cases
* Answers to question: XX%
  + Correct answer to question in step 4
  + Sufficient work shown
  + Neatness and clarity