# Lab 2: Basic Matrix Multiplication

Due Date: …

## 1. Objective

The purpose of this lab is to implement a basic dense matrix multiplication routine.

## 2. Procedure

**Step 1:** Update your local repository to obtain the code needed for the assignment.

cd <labs-directory>

hg pull

hg update

**Step 2:** Edit the file lab2/main.cu to implement the following where indicated:

1. Allocate device memory
2. Copy host memory to device
3. Copy results from device to host
4. Free device memory

**Step 3:** Edit the file lab2/kernel.cu to initialize the thread block and kernel grid dimensions and invoke the CUDA kernel, and to implement the matrix multiplication kernel code.

**Step 4:** Compile and test your code.

cd lab2

make

./sgemm # Uses the default matrix sizes

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

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

It is a good idea to test and debug initially with small input dimensions. Your code is expected to work for varying input dimensions – which may or may not be divisible by your block size – so don’t forget to pay attention to boundary conditions.

**Step 5:** Answer the following questions in a new file named lab2/answers.txt:

1. How many times is each element of each input matrix loaded during the execution of the kernel?
2. What is the memory-access to floating-point computation ratio in each thread? Consider multiplication and addition as separate operations, and ignore the global memory store at the end. Only count global memory loads towards your off-chip bandwidth.

**Step 6:** Submit your assignment. You should only submit the following files:

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

Compress the files and name them after your student id like so:

tar -cf id.tar main.cu kernel.cu answers.txt

Send the compressed folder by email to <TA’s email address> with “ECE408 Lab 2” in the subject line. Submissions with incorrect subject lines may not be processed.

**3. Grading:**

Your submission will be graded based on the following criteria.

* Functionality/knowledge: 65%
  + Correct code and output results
  + Correct usage of CUDA library calls and C extensions
  + Correct handling of boundary cases
* Answers to questions: 35%
  + Correct answer to questions in step 5
  + Sufficient work is shown
  + Neatness and clarity