

Programming Assignment 2: Basic Matrix Multiplication

Due Date: September 19 (Tuesday) at 11:59PM

Required for ALL students

1. Objective

The purpose of this programming assignment is to implement a basic dense matrix multiplication routine **without** using tiling technique.

2. Procedure

Step 1: Download the programming assignment 2 materials from blackboard to your home folder at the Karpinski GPU cluster. Unzip it.

```
unzip p2-sgemm.assignment.zip
```

Step 2: Edit the file `main.cu` to implement the following where indicated:

- a) Allocate device memory
- b) Copy host memory to device
- c) Copy results from device to host
- d) Free device memory

Step 3: Edit the file `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.

```
make  
  
./sgemm                # Uses the default matrix sizes  
./sgemm m              # Uses square m × m matrices  
./sgemm m k n          # Uses (m × k) and (k × n) input matrices
```

It is a good idea to test and debug initially with small input dimensions. Your code is expected to work for various 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 `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 (i.e., the total number of memory accesses divided by the total number of floating-point operations) in each thread? Consider multiplication and addition as separate operations, and ignore the global memory store at the end. Only count global memory loads when calculating the total number of memory accesses.

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

- `main.cu`
- `kernel.cu`
- `answers.txt`

Compress the files and name them as following:

```
tar -cvf p2_<your lastname>.tar main.cu kernel.cu answers.txt
```

If the last name is alan, the tar file is `p2_alan.tar`

3. Grading:

Your submission will be graded based on the following criteria.

- Functionality/knowledge: 90%
 - Correct code and output results
 - Correct usage of CUDA library calls and C extensions
 - Correct handling of boundary cases
 - Check return values of CUDA APIs
- Answers to questions: 10%
 - Correct answer to questions in Step 5
 - Sufficient work is shown
 - Neatness and clarity