University of Arkansas
Department of Computer Science and Computer Engineering
CSCE4643/5693 – GPU Programming

Programming Assignment 3: Tiled Matrix Multiplication

Due Date: September 26 (Tuesday), 2023 at 11:59pm
Required for all students

1. Objective

The purpose of this programming assignment 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: Download the programming assignment 3 materials from blackboard to your home folder at Karpinski cluster. Unzip it.

```
unzip p3-sgemm-tiled.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 x m matrices

./sgemm 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 the 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 5: Submit your assignment to the blackboard. You should only submit the following files:

- `main.cu`
- `kernel.cu`

Compress the files and name them as following:

```
  tar -cvf p3_<your lastname>.tar main.cu kernel.cu
```

If the last name is alan, the file name is p3_alan.tar

3. Grading:

Your submission will be graded based on the following criteria.

- Functionality/knowledge: 100 points
    - Correct code and output results (40%)
    - Correct usage of shared memory in the kernel to hide global memory access latencies (40%)
    - Correct handling of boundary cases (10%)
    - Check return values of all CUDA APIs (10%)