Spring 2018 **ECE 8823/CS8803: GPU Architectures**

Assignment 1

Due: 11:59 pm, Wednesday, January 31st, 2018

Purpose: The primary purpose of this assignment is for you to i) become familiar with the steps of compiling and executing a simple CUDA program, and ii) get comfortable with the thread hierarchy and the grid model.

Target Machines: For this assignment you will use two GPUs made available in ECE. Instructions for accessing these GPUs can found on the via the Resources page and at http://ece8823-sy.ece.gatech.edu/resources/accessing-gpus-on-the-ece-machines/.

Part I: Warmup - Executing CUDA Programs

Purpose: Understand the basic runtime and kernel calling conventions and resolve compilation and execution steps in accessing the GPUs.

Assignment: For the warmup, implement and execute the vector-add kernel from the text for two integer arrays. Have the vectors be at least 1K elements and the kernel parametric in the number of thread blocks and the number of threads per block. You can use the code from the text or the template listed on the class repository (see Class Resources page link on the class page). The latter is more elaborate in terms of error checking, device query, etc., so you might find this a more useful example. You might vary the i) size of the thread blocks, ii) number of thread blocks, or iii) the number of elements that each thread will add. You do not need to turn in any code for this part.

Part II: Matrix Addition

Purpose: Become comfortable with threading and data addressing in a 2D grid.

Assignment: Perform the addition of two matrices using a 2D grid of thread blocks. Like vector addition, each thread will add one element from each matrix. The kernel code must be parametric in the matrix size, grid, and block parameters. The kernel must function for matrix sizes of at least 64x64 elements (they can be integers). Note: there will be a limit on the total number of threads in a kernel depending on the specific GPUs that you will have access to (you can use the device query functions to determine the maximum number of threads).

1. Your kernel should be parametric in the matrix dimensions, the number of thread blocks in each dimension, and the size of each thread block. Each thread block should be a 2D array of threads. To keep it simple, the thread block dimensions can be square.
2. Your kernel should work for integer matrices of varying input sizes. To keep it simple you can support square matrices only.
3. Use CUDA event timers to record the kernel execution time. Report the execution time for any three grid and matrix configurations and print to the output file. CUDA event timers are documented at http://devblogs.nvidia.com/parallelforall/how-implement-performance-metrics-cuda-cc/.
4. The input files should provide the parameters and matrices in the following format. The input file will be named *A1Input.txt*.

Sample Input file:

```
// grid dimensions
1	1	1
// block dimensions
2	2	1
// matrix dimensions
2	2
// matrix 1 (in row major form)
1	1
1	1
// matrix 2 (in row major form)
2	2
2	2
```

5. Print the output matrix to the file *A1Output.txt* in the following format

Sample Output file:

```
// matrix sum (in row major format).
3	3
3	3
```

6. Your programs will be executed with the following command line.

Sample command line: ./a.out < A1Input.txt

Grading Guidelines

For your information here are the grading guidelines

- Program compiles without errors (and appears to be correct): 35 points
- Program executes correctly for one set of inputs: (additional) 35 points
- Program executes correctly for across a set of valid parameter values: 15 points
- Documentation (comments): 15 points
    - Have a commented program header with your name, class, and assignment number
    - Commented sections of code

Submission Guidelines:

All program submissions should be electronic. Submissions must be time stamped by midnight on the due date. Submissions will be via Tsquare.

Note: **No late assignments will be graded**. Remember, you are expected to make a passing grade on the assignments to pass the course!