**CSC3004 Topics: GPU Programming Spring 2015**

#### Midterm Project: Matrix Operations

1. **Objectives**
   1. **To learn how to create a project running CUDA C**
   2. **To learn how to program CUDA code to perform parallel computing**
2. **Matrices**

**An *M × N* (pronounced "*M* by *N*") matrix *A* is a rectangular array of numbers or expressions, arranged in *M* rows and *N* columns. The individual items in a matrix are called its elements or entries.**

***A* =**

**An example of a *3* × *4* matrix (with *3* rows and *4* columns) is**

**which is the dot product of *i*-th row of A and *j*-th column of B.**

1. **Programming Assignment 1 (50 points)**

**Matrix addition**

**The matrix addition is defined for two matrices A and B of the same dimensions, i.e. same rows and columns. The sum of two *M × N* matrices *A* and *B*, denoted by *C*, is again an *M × N* matrix computed by adding corresponding elements.**

**=**

**Where the *(i, j)* element  *= +* with *i = 1, 2, … ,* *M* and *j = 1, 2, …, N*.**

**The following is an example of matrix addition:**

**+ =**

**Code requirement**

**Write a CUDA C program that does the matrix addition.**

* 1. **You should create three arrays a, b, and c of type float.**
  2. **You need to create a kernel method called matrixAdd() that is passed three two-dimensional arrays: dev\_a, dev\_b, and dev\_c, in which dev\_a and dev\_b are used to hold input matrices and dev\_c is to hold the result matrix.**

1. **Programming Assignment 2 (50 points)**

**Matrix multiplication**

**If *A* is an *M × N* matrix and *B* is an *N × P* matrix,**

***A* = , *B* =**

**the matrix product *C = AB* is defined to be the *M × P* matrix**

***C* =**

**Where the *(i, j)* element of matrix C is defined as**

**=**

**which is the dot product of *i*-th row of A and *j*-th column of B.**

**Code requirement**

**To make your life a little bit easy, we just need to implement a CUDA program that does the matrix multiplication of two N x N square matrices.**

1. **You should create three arrays a, b, and c of type float.**
2. **You need to create a kernel method called matrixMult() that is passed three two-dimensional arrays: dev\_a, dev\_b, and dev\_c, in which dev\_a and dev\_b are used to hold input matrices and dev\_c is to hold the result matrix.**
3. **You can implement in such a way that each thread computes one element of C**
   1. **Loads a row (say i-th row) of matrix A**
   2. **Loads a column (say j-th column) of matrix B**
   3. **Computes a dot product**
4. **Every value of A and B is loaded N times from global memory.**
5. **Grading:**

**Your program must compile and run to receive full credit. If a portion of your program is correct, you can receive partial credit. As noted earlier, you must implement a kernel methods for both assignments, or you will not receive any credit. The grade for your project will be determined on the following basis:**

**For matrix addition:**

1. **15 points: implement both matrixAdd() for matrix addition**
2. **15 points: and main() method**
3. **10 points: the program compiles and runs.**
4. **10 points: the program compiles and runs correctly.**

**For matrix multiplication:**

1. **15 points: implement both matrixMult() for matrix addition**
2. **15 points: and main() method**
3. **10 points: the program compiles and runs.**
4. **10 points: the program compiles and runs correctly.**