Introduction to Parallel Computer Architecture
CUDA Programming Assignment 1

Instructor: Prof. Naga Kandasamy, ECE Department, Drexel University

November 11, 2015

The lab is due on November 22, 2015. You may work on the problems in teams of up to two people.

Matrix-Vector Multiplication. You will multiply a dense $n \times n$ matrix A with an $n \times 1$ vector x to yield the $n \times 1$ result vector y. The serial algorithm is shown below.

```
procedure VEC_MAT_MULT($A$, $x$, $y$)
int $i$, $j$;
for $i := 0$ to $n - 1$ do
    $y[i] := 0$;
    for $j := 0$ to $n - 1$ do
        $y[i] := y[i] + A[i, j] \times x[j]$;
    end for
end for
```

Answer the following questions:

- **(10 points)** Edit the `vec_mat_mult_on_device_using_global_memory()` function and the corresponding "naive" kernel function to complete the functionality of the vector-matrix multiplication on the GPU using global memory.
- **(15 points)** Edit the `vec_mat_mult_on_device_using_shared_memory()` function and the corresponding optimized kernel function to complete the functionality of the vector-matrix multiplication on the GPU using shared memory.

The CUDA source files for this question are available on BBLearn as a zip file. Your program should accept no arguments. The application will create a randomly initialized matrix and a vector to multiply. After the GPU-based multiplication kernel is invoked, it will then compute the correct solution using the CPU and compare that solution with the GPU-computed solutions. If the solutions match within a certain tolerance, the application will print out "Test PASSED" to the screen before exiting.

Upload all of the files needed to run your code as a single zip file via BBLearn. This question will be graded on the following parameters:

- Report the speedup achieved by the GPU kernels over the CPU implementation for the following matrix sizes: 1024×1024, 2048×2048, 4096×4096, and 8192×8192.
- Include a brief report describing how you designed your kernels (use code or pseudocode to clarify the discussion) and the amount of speedup obtained over the serial version for both GPU-based versions.
- The GTX 275 GPU can achieve a peak processing rate of about 933 GFLOPs. The memory bandwidth on the device is 141.7 Gb/s. How many floating-point operations must be performed per load operation to achieve the peak processing rate? What is the performance of your kernels (both naive as well as the one that uses shared memory), in terms of GFLOPs?