Programming Assignment 9
Due Sunday April 19

Objectives
Use standard GPU features such a shared memory to write a useful GPU program. Continue exploration of MMM.

Prerequisites (covered in class or through examples in on-line documentation)
HW – Standard GPU features: shared memory
SW – Explicitly managing data transfers in the memory hierarchy
Programming – CUDA support for shared memory, tiling, more complex kernels

---
Assignment

Implement, test, and verify MMM.
- Use single precision floating point.
- Try at least 2 different matrix sizes: 1Kx1K and 2Kx2K
- Validate with MMM on the host. Once you are sure that your GPU code is working, report the tolerance required -- that is, the largest deviation between GPU and CPU result. Report this along with the way that you generated the matrix elements.
- For the GPU execution, record two times: end-to-end including data transfers, and just the MMM execution time. Compare the performance with your best CPU MMM time from earlier in the semester.
- Base your programs on the codes presented in class.

Part 1: MMM using global memory only.

Part 2: MMM using shared memory.

Part 3: MMM using shared memory and loop unrolling.

Extra credit:
- Experiment with different ThreadBlock sizes.
- Try a few more matrix sizes.
- Find the matrix size(s), if any, where the GPU runs into problems.
- Find a GPU MMM code on the web. Get its performance. If the source code is available, explain its features.
- Now that you know about coalescing memory accesses (global) and how memory banks work (shared), change your code so that the access pattern is no longer favorable. How does this change performance?