**Homework 3 Memory and Data Locality**

1. Consider the matrix addition kernel from the previous homework exercise. Can one use shared memory to reduce the global memory bandwidth consumption? Hint: analyze the elements accessed by each thread and see if there is any commonality between threads.

Shared memory will reduce the global memory bandwidth consumption as each block can contain the row of matrix\_A and the column of matrix\_B. The threads can the compute the vector multiplication and sum them to produce the output matrix without having to access global memory. This is dependent on the threads/block ratio, as the block can contain more than one row of matrix\_A and column of matrix\_B.

Can’t be used

1. What type of incorrect execution behavior can happen if one forgets to use syncthreads() after instructions for loading a tile in shared memory?

If syncthreads() isn’t used, the threads could execute out of order, meaning that if there is an operation that depends on the data calculated by previous threads, then the operation can give false data as not all the threads have finished their execution. For example, if you have an operation where you add up all the outputs of previous threads without syncing them; then the threads that haven’t executed are defaulted to zeros (if calloced) or some random value.

1. Assuming capacity were not an issue for registers or shared memory, give one case that it would be valuable to use shared memory instead of registers to hold values fetched from global memory? Explain your answer.

Registers can only be accessed by its thread, while shared memory can be accessed by all threads in the block. Using only registers can be slower than using shared memory when data has to be shared between threads. If the data is located in the register, it must be copied over to shared memory and then copied to the receiving thread’s register. This overhead will greatly slow down performance.

1. For our tiled matrix-matrix multiplication kernel, if we use a 32x32 tile, what is the reduction of memory bandwidth usage for input matrices M and N?
   1. 1/8 of the original usage
   2. 1/16 of the original usage
   3. 1/32 of the original usage
   4. 1/64 of the original usage
2. Assume that a kernel is launched with 1000 thread blocks each of which has 512 threads. If a variable is declared as a local variable in the kernel, how many versions of the variable will be created through the lifetime of the execution of the kernel?
   1. 1
   2. 1000
   3. 512
   4. 512000
3. With respect to the previous question, if a variable is declared as a shared memory variable, how many versions of the variable will be created through the lifetime of the execution of the kernel?
   1. 1
   2. 1000
   3. 512
   4. 512000
4. Consider performing a matrix multiplication of two input matrices with dimensions NxN. How many times is each element in the input matrices requested from global memory when:
   1. There is no tiling?

N

* 1. Tiles of size TxT are used?