Sattler College
Computer Science
CS 420 – Systems and Parallel Processing

Lab 3: Tiled Matrix Multiplication

1. Objective

The purpose of this lab is to get you familiar with using shared memory to write optimized kernel algorithms by implementing a "tiled" version of matrix multiplication.

2. Procedure

Step 1: Download the program files in Lab_3.zip

Step 2: Edit `<lab-directory>/main.cu` and `<lab-directory>/kernel.cu` to include the host setup code and device kernel code where indicated.

Step 3: Compile and test your code.

```
cd <lab-directory>

make


./sgemm-tiled                 # Uses the default matrix sizes

./sgemm-tiled <m>             # Uses square m x m matrices

./sgemm-tiled <m> <k> <n>     # Uses (m x k) and (k x n) input matrices
```

Your code is expected to work for varying input dimensions – which may or may not be divisible by your tile size. It is a good idea to test and debug initially with examples where the matrix size is divisible by the tile size, and then try the boundary cases.

The variable input arguments can be read using the C++ argc and argv variables – this site shows an example:

https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm.

Step 4: Answer the following question in a new file named `<name>answers.txt`:

- Try three tile sizes – 16, 32 and 64 – how does this effect the speed of your multiplication?
- How does the tiled solution compare to the Baseline (Lab 2) transpose variation? What is the reason for the performance differences.

Step 5: Submit your assignment. You should submit all files in a zip file, including:

- `main.cu`
- `kernel.cu`
- `answers.txt`

3. Grading:

Your submission will be graded based on the following criteria.

- Functionality/knowledge: 60%

  - Correct code and output results
  - Correct usage of shared memory in the kernel to hide global memory access latencies
  - Correct handling of boundary cases
- Answers to question: 40%
  - Correct answer to question in step 4
  - Sufficient work shown
  - Neatness and clarity