# Lab 4: Tiled Convolution

Due Date: …

## 1. Objective

The purpose of this lab is to get you more familiar with shared memory tiling techniques, handling complex boundary conditions, and using constant memory.

## 2. Procedure

**Step 1:** Update your local repository to obtain the code needed for the assignment.

cd <labs-directory>

hg pull

hg update

**Step 2:** Edit lab4/main.cu to add the constant memory copy and set the block and grid dimensions correctly. Edit lab4/kernel.cu to implement the shared memory tiled convolution. To handle halo cells, treat them as having a value of zero.

**Step 3:** Compile and test your code.

cd lab4

make

./convolution # Uses the default input image size

./convolution <m> # Uses a square m x m input image

./convolution <m> <n> # Uses an (m x n) input image matrix

**Step 4:** Answer the following questions in a new file named lab4/answers.txt:

1. What is the floating-point computation rate for the GPU kernel in this application? How does it scale with the size of the input image? To answer this question, try multiple sized inputs and calculate the rate for each using the timing measurements provided in the code. Make sure to justify your choice of input sizes.
2. What percentage of time is spent as overhead for using the GPU? Consider as overhead: device memory allocation time and memory copy time to and from the device. Do not include problem setup time or result verification time in your calculations of overhead or total execution time. Try this with multiple input sizes and explain how the overhead scales with the size of your input?

**Step 5:** Submit your assignment. You should only submit the following files:

* main.cu
* kernel.cu
* answers.txt

Compress the files and name them after your student id like so:

tar -cf id.tar main.cu kernel.cu answers.txt

Send the compressed folder by email to <TA’s email address> with “ECE408 Lab 4” in the subject line. Submissions with incorrect subject lines may not be processed.

**3. Grading:**

Your submission will be graded based on the following criteria.

* Functionality/knowledge: 65%
  + Correct code and output results
  + Correct usage shared and constant memory to cover global memory access latency
  + Correct handling of boundary cases
* Answers to question: 35%
  + Correct answer to questions in step 4
  + Sufficient work shown
  + Neatness and clarity