LSU EE 7722 **Homework 3** **Due: 30 April 2020**

The code for this assignment can be found in the `hw03` *directory. It is a 2-dimensional version of the convolution code used in Homework 1.*

Problem 1: Modify the code in `conv_prob1` so that there is overlap within a block of `d_in` elements accessed from one `h` iteration to the next, with the goal of increasing the chances of finding `d_in` values in the level 1 cache. (This should be run on a CC 7.X GPU, since they have a level 1 cache.) For example, in the first `h` iteration block 0 may update a rectangular section of `d_out` including columns 0 through 31 and rows 0 through 4. In the block's second `h` iteration it might access columns 0 through 31 and rows 5 through 9.

Think about which shape (maximum number of columns, maximum number of rows, equal number of columns and rows, etc.) would provide the best performance accounting for overlap, utilization and other issues.

Problem 2: In the NVidia assembler code for `conv_wbuf` there is one load instruction (`LDG` on Turing) for each `FFMA`. This forces execution time to be at least four times the lower bound based on the number of `FP32` units. Modify the code in `conv_prob2` so that a value is loaded once and used `n_per_thread` multiple times, where `n_per_thread` is a constant in the kernel. Inspect the SASS code and make sure that there are fewer `LDG` instructions and also make sure that the `FFMA` instructions directly access constant memory, rather than rely on separate `LDC` (load constant) instructions.

Solve this by having one `h` iteration compute `n_per_thread` values of `d_out`. First make sure that the code is correct and using each loaded value `n_per_thread` times, then worry about performance.

Problem 3: The bar graphs printed by the code show FP utilization counting only FP instructions. Modify the code so that the `+` segments in the bar graph show utilization including `FFMA` and memory instructions. (Most of the code is written. Search for `num_ops_ls`.) Set a value appropriate for each kernel, especially your solution to Problem 2.