ME 759
High Performance for Engineering Applications
Assignment 4
Due Friday 02/21/2020 at 9:00 PM

Submit responses to all tasks which don't specify a file name to Canvas in a file called assignment4.txt, docx, pdf, rtf, odt (choose one of the formats). Also all plots should be submitted on Canvas. All *source files* should be submitted in the `HW04` subdirectory on the `master` branch of your homework git repo with no subdirectories.

All commands or code must work on *Euler* with only the `cuda` module loaded unless specified otherwise. Commands and/or code may behave differently on your computer, so be sure to test on Euler before you submit.

Please submit clean code. Consider using a formatter like clang-format.
* Before you begin, copy the provided files from `HW04` of the ME759-2020 repo. Do not change any of the provided files because we will write clean copies over them when grading.

1. (a) Implement in a file called `matmul.cu` the `matmul` and `matmul_kernel` functions as declared and described in `matmul.cuh`.

   (b) Write a program `task1.cu` which does the following:

   - Creates matrices (as 1D row major arrays) `A` and `B` of size `n*n` in *managed (aka unified) memory*.
   - Fills those matrices however you like.
   - Calls your `matmul` function.
   - Prints the last element of the resulting matrix.
   - Prints the time taken to perform the multiplication in *milliseconds* using CUDA events.
   - Compile: `nvcc task1.cu matmul.cu -Xcompiler -O3 -Xcompiler -Wall -Xptxas -O3 -o task1`
   - Run (where `n` and `threads_per_block` are positive integers): `./task1 n threads_per_block`
   - Example expected output:
     ```
     11.36
     1.23
     ```

   (c) On an Euler *compute node*, run `task1` for each value $\texttt{n} = 2^5, 2^6, \cdots, 2^{15}$ and generate a plot `task1.pdf` which plots the time taken by your algorithm as a function of `n` when `threads_per_block` $= 1024$. Overlay another plot which plots the same relationship with a different choice of `threads_per_block`.

2. (a) Implement in a file called `stencil.cu` `stencil` and `stencil_kernel` functions as declared and described in `stencil.cuh`. These functions should produce the 1D convolution of `image` and `mask`:

$$\texttt{output}[i] = \sum_{j=-R}^{R} \texttt{image}[i+j] * \texttt{mask}[j+R] \qquad i = 0, \cdots, \texttt{n}-1$$

Assume that $\texttt{image}[i] = 0$ when $i < 0$ or $i > \texttt{n} - 1$. Pay close attention to what data you are asked to store and compute in shared memory.

(b) Write a program `task2.cu` which does the following:

- Creates arrays `image` (length `n`), `output` (length `n`), and `mask` (length `2 * R + 1`) all in *managed memory*.
- Fills those arrays however you like.
- Calls your `stencil` function.
- Prints the last element of the resulting array.
- Prints the time taken to perform the convolution in *milliseconds* using CUDA events.
- Compile: `nvcc task2.cu stencil.cu -Xcompiler -O3 -Xcompiler -Wall -Xptxas -O3 -o task2`
- Run (where `n`, `R`, and `threads_per_block` are positive integers):
  `./task2 n R threads_per_block`
- Example expected output:
  `11.36`
  `1.23`

(c) On an Euler *compute node*, run `task2` for each value $\texttt{n} = 2^{10}, 2^{11}, \cdots, 2^{31}$ and generate a plot `task2.pdf` which plots the time taken by your algorithm as a function of `n` when `threads_per_block` $= 1024$ and `R` $= 128$. Overlay another plot which plots the same relationship with a different choice of `threads_per_block`.