CS239 programming exercise 2

1 Objectives

Review and demonstration of the following:

- Theory and practice of the memory hierarchy in parallel programming. In particular
  - Improving performance of parallel programs by exploiting other types of memory other than global memory (e.g. shared memory), improving access times, patterns, and latency tolerance.
  - The ability to reason out how to design parallel programs based on memory types and hardware limits.

The objectives are obtained through theoretical and practical experiments from understanding and application of well-known concepts fitted to the GPU. Specificaly, your choice(s) of parameters, e.g. size or dimension of your input(s), blocks in the grid, block of threads. The design, execution, and analyses of your experiments are the main highlights of the exercise and report.

2 Exercises

Main references I suggest is [Kirk and Hwu, 2012] and [NVIDIA, 2025], but you can consult other sources as long as you cite precisely which part(s) is your own work, and which part(s) are not (including where you take those part(s) that are not yours). Perform the citations in the short report.

For this exercise, main references are Chapters 4 and 5 of [Kirk and Hwu, 2012] or their corresponding section(s) in [NVIDIA, 2025].

1. Again, begin your code by querying the properties of the device(s), i.e. GPU(s), that are installed in the system you are using. You can use the `cudaDeviceProp` type, then loop and print all device properties.
2. In Chapters 4 and 5 of [Kirk and Hwu, 2012] two matrix multiplication versions were presented: a version using global memory only and a version using global and shared memory. Both versions assumed that the input matrices are square matrices.
   - In your source code, create a kernel `matmul_rec_glob` which is a matrix multiplication for rectangular (i.e. not necessarily square) input matrices using global memory (for now). Recall that if matrix `A` is n rows by k columns, and matrix `B` is k rows by m columns where n is not necessarily equal to m or k, then `C` $=$ `A` $\times$ `B` is an n rows by m columns product matrix.

     Your `blockDim` preferably is a multiple of 32 (for performance reasons, check Chapters 4 and 5 of [Kirk and Hwu, 2012]) so you may choose to still create square 2D blocks for your thread blocks, while your grid of blocks is not necessarily square. The design considerations are up to you *as long as you CLEARLY* provide the dimensions of your design and considerations. For example, for a `blockDim` of 32 and with a square block, you can choose to launch $ceil(n/32)$ rows and $ceil(m/32)$ columns compute for `C`. Please do not use padding in order to transform `A`, `B`, and `C` into square matrices as in [Kirk and Hwu, 2012].

In your kernel functions, make sure that n, k, m, and block width are variables.

- Create a second kernel `matmul_rec_shar` which performs matrix multiplication as in your kernel `matmul_rec_glob`, but this time using a tiling algorithm and shared memory. Again, CLEARLY indicate the design considerations in your report.
- Analyse the pros and cons of your kernels. Include computations for CGMA as in Chapter 5 of [Kirk and Hwu, 2012]. Preferably include also the averaged run times (shown as graphs, charts etc) using (pseudo) randomly generated single-precision float values and various matrix sizes. Argue the pros and cons of your design based on values obtained from the device (using `cudaDeviceProp`) and tile size (or the lack of tiling), block or grid dimensions, etc.

3 Report

Precisely cite in your report which parts of your work are yours, and which parts are not, whether the part(s) are in code form or design form. Remember that despite the apparent simplicity or straightforwardness of the exercise, the *point of the exercise* is not to insult your programming skills or understanding of concepts; the point is to gain *insights* from the experiments, and NOT just report beautiful tables, numbers, charts and the like. Aside from almost certainly citing [Kirk and Hwu, 2012] and [NVIDIA, 2025] in your report, you can cite other sources e.g. articles, books, videos. Please typeset your report, including snippets of your code, using a recommended maximum of 4 pages and using `article` template of LaTeX e.g. `https://www.overleaf.com/latex/templates/a-simple-article-template/gdsdkccmjnxg`.

When citing code in your report, make sure you use the `typewriter font` to make reading and identification easier. Email me the PDF file of your exercise report. The filename of your report for exercise number `X` as a PDF file should be

`lastname-exerX.pdf`

with **email subject** as `lastname-exerX`. Those who prefer offline, i.e. download and setup required but not constant Internet connection afterwards, you can download LaTeX for Linux, Mac, or Windows. Those who prefer online, i.e. no download and setup required but constant Internet connection is required, I suggest `https://www.overleaf.com`.

References

[Kirk and Hwu, 2012] Kirk, D. and Hwu, W. (2012). *Programming Massively Parallel Processors: A Hands-on Approach (2nd ed).* Morgan Kaufmann.

[NVIDIA, 2025] NVIDIA (February 2025). CUDA C programming guide. `http://docs.nvidia.com/cuda/cuda-c-programming-guide`.