ECE 5755 Modern Computer Systems and Architecture, Fall 2023
Assignment 2: Implementing Tiling and Blocking

1. Introduction

This lab assignment will build off the work you completed in the previous lab assignment. In particular, you will be exploring the performance gains in your kernel when you implement **loop tiling/blocking** for matmul.

2. Materials

No additional code is provided for this lab, use all of the same materials and your own code from Lab 1. Make sure to make a copy of your original lab 1 code or use a version control tool like git so that you have a known good state to revert to. You will add an additional function `matmul_blocking` which implements a tiling/blocking version of `matmul`.

3. Objective

The objective of this lab is to implement a better matrix-multiplication algorithm that uses tiling/blocking, and **to analyze the performance gains of your algorithm using microarchitectural analysis techniques**. You are tasked with implementing the following function:

```
float **matmul_blocking(float **A, float **B, int A_rows, int A_cols,
int B_rows, int B_cols)
```

The function should be put in `kernel/matrix_ops.c`. Don't forget to put the function declaration in the header file (`kernel/matrix_ops.h`) as well so the lab will compile. You'll then want to swap out your original matrix multiplication implementation for this one in your tests to verify that it's actually working, i.e. call `matmul_blocking` instead of `matmul` inside `tests/test_matrix_ops.c`

You are expected to optimize `matmul_blocking` for performance (runtime). Try a few different implementations in terms of tiling/blocking sizes, patterns, and loop order and try to find the best implementation for performance. When submitting, only submit the final/most optimized implementation. When testing for runtime, you can use the Linux `time` command; make sure to take the `real` time which is the wall-clock time. Moreover, as with profiling, make sure you are looping over the function you want to profile so that the runtime of the entire program is dominated by the function you are profiling/timing. Test with a variety of input sizes; **make sure you have some inputs that are large enough to fill up the L1 cache.**

For Lab 2, you are only required to submit `matrix_ops.c` and `matrix_ops.h` with `matmul_blocking` and any additional helper functions you choose to use inside those two files.

For this lab, we will not be grading your testing procedure, only the correctness of `matmul_blocking`. Functional correctness grading will be done with an internal grading program similar to the testing programs you were expected to submit for Lab 1. We will also be reading your code to grade for correctness in implementing tiling/blocking, so please make sure your code is readable (use comments to clarify your code).

Your profiling process must meet the following requirements:

- Use `toplev.py` from `pmu-tools` for Top-down analysis data
- Profile only the core the program is running on
- Runtime of the program-under-test should be dominated by the function-under-test (you can accomplish this by looping over your function many times)

The specific commands you use for profiling and testing and how you write your program-under-test is up to you as long as your process meets the above requirements. One option is to use the existing testing infrastructure under `tests/`.

4. Report

The report will be **3-pages max** but can be shorter if you do not need all three pages. Report requirements/prompts:

- Include a horizontal segmented bar chart for the Top-down analysis of some (~2-3) of the different implementations (i.e. tile sizes, etc.) and different test inputs of `matmul_blocking` you tried. In the same bar chart, for the same set of inputs, also include the Top-down analysis data for `matmul` from Lab 1 to serve as a baseline.
- For the same implementations (incl. `matmul` from Lab 1) and inputs, collect their performance data (runtime) and include it in a table in the report.
- Explain in detail your profiling process.
- With reference to computer architecture concepts and the Top-down analysis data, explain the performance differences between the different implementations and inputs.
- Discuss performance differences between `matmul_blocking` and `matmul`.
- Discuss any issues you had during the lab and your debugging process.

5. Deliverables

Please submit your assignment on Canvas by **Monday, Oct. 9, 11:59 pm**. You are expected to submit three files total: **matrix_ops.c, matrix_ops.h,** and your report in a file named **lab2.pdf.**

The lab will be marked out of a **total of 100 marks.**

- **40 marks** for `matmul_blocking()`
    - 20 marks for functional correctness (computes correct outputs for given set of inputs)
    - 20 marks for correct implementation of tiling/blocking
- **60 marks** for report
    - 20 marks for Top-down analysis data
    - 10 marks for performance (runtime) data
    - 30 marks for discussion (see prompts in 4. Report)