Lab 3: Matrix Multiplication with Thread Coarsening and Register Tiling

EE5355 - Parallel Algorithmic Techniques

Due: See Moodle

1 Objective

The purpose of this lab is to practice thread coarsening and register tiling optimization techniques using matrix-matrix multiplication as an example.

2 Procedure

Step 1: Transfer the tarfile lab3.tgz into your project directory and extract it to create the lab3 directory.

Step 2: Edit the file `kernel.cu`, to implement and launch a matrix-matrix multiplication kernel that uses thread coarsening and register tiling optimization techniques. The first input matrix has a column major layout and shall be tiled in the registers, the second input matrix has a row major layout and shall be tiled in shared memory, and the output matrix has a column major layout and shall be tiled in the registers. Macros have been provided to help you with accessing these matrices easily.

Step 3: Compile and test your code.

```
make
./sgemm-tiled                    # Uses the default matrix sizes
./sgemm-tiled <m>                # Uses square m x m matrices
./sgemm-tiled <m> <k> <n>        # Uses (m x k) and (k x n) input matrices
```

Your code is expected to work for varying input dimensions - which may or may not be divisible by your tile size. It is a good idea to test and debug initially with examples where the matrix size is divisible by the tile size, then try the corner cases.

Step 4: Edit the file `report.txt` to answer the following questions:

1. Assuming an output tile of size (M rows)x(N columns), where M is a multiple of N, how much on-chip memory (registers and shared memory) is needed to store each of the input and output tiles. Show your work.

2. Try to tune your tile sizes for the two input matrices and output matrix. Report the execution time for various combinations of tile sizes. Comment on the results, explaining what factors you think might have influenced the difference (or lack of difference) in performance when the tile sizes change.

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

- `kernel.cu`
- `report.txt`

3 Grading

Your submission will be graded based on the following criteria.

- Functionality/knowledge: 75%
  - Correct code and output results
  - Correct usage of register tiling and shared memory tiling in the kernel
  - Correct handling of boundary cases
- Answers to question: 25%
  - Correct answers to questions and depth of analysis
  - Sufficient work shown
  - Neatness and clarity