# Lab 3 Report

Zhaoze Sun
862395352
zsun114@ucr.edu

1. On Bender, compare the execution time of a 256 x 256 square matrix multiplication compared to a 1024 x 64 and 64 x 1024 rectangular matrix multiply. All input matricies have 65k entries. What do you observe? Which is faster? Can you explain the observed behavior? Tip: You may want to comment out the verify() function in main.cu when timing this question.
gpgpu_simulation_time = 0 days, 0 hrs, 1 min, 33 sec (93 sec)(256*256*256)
gpgpu_simulation_time = 0 days, 0 hrs, 9 min, 9 sec (549 sec)(1024*64*1024)
By observed the result of these two results we can find out that the 256256256 is faster than second one.
256*256*256= 16,777,216
1024*64*1024= 67,108,864
Because of the difference of operation number we can observe that the second one is fewer than the first one. So obviously we can indicate that the second one is faster one.

2. Conceptual Question: For a 64 square tiled matrix multiplication, how many times is each element of the input matrices loaded from global memory? Assume 16x16 tiles.

Answer: 4

3. Conceptual Question: For a 64 square non-tiled matrix multiplication, how many times is each element of the input matrices loaded from global memory?

Answer: 64

4. GPGPU-Sim related question: In this part, we will compare the execution of a 128x128 square tiled matrix multiplication across different tile sizes. Run ./sgemm-tiled 128 in GPGPU-Sim with TILE_SIZE of 8, 16 (default), and 32. Fill the following table:

| Tile size | 8 | 16 | 32 | Note |
|---|---|---|---|---|
| gpu_tot_sim_cycle | 43433 | 27727 | 57904 | Total cycles |
| gpu_tot_ipc | 420.6055 | 460.3144 | 390.1895 | Instruction per cycle |
| gpgpu_n_load_insn | 524,288 | 262,144 | 131,072 | Total loads to global memory |
| gpgpu_n_store_insn | 16384 | 16384 | 16384 | Total stores to global memory |
| gpgpu_n_shmem_insn | 4718592 | 4456448 | 4325376 | Total accesses to shared memory |

5. Which tile size resulted in the least number of accesses to global memory? Which tile size resulted in the most number of accesses to global memory? What is the reasoning behind this observation?

Answer: The size of tile which is the least number we access in the global memory is 32 which is 131,072. The tile size of 8 resulted in the most number of accesses to global memory is 524,288. When the tile size increases, we can get more reuse in shared memory in order to avoid global memory access.

6. Which tile size performed the fastest, which tile size performed the slowest? Why do you think that is?
Tile = 16: gpgpu_simulation_time = 0 days, 0 hrs, 0 min, 9 sec (9 sec)
Tile = 8: gpgpu_simulation_time = 0 days, 0 hrs, 0 min, 15 sec (15 sec)
Tile = 32: gpgpu_simulation_time = 0 days, 0 hrs, 0 min, 12 sec (12 sec)
The fastest one is 16 and the slowest one is 8. I think there is such a big difference between each computer. And I also think it is vary as the threads per SM depends on different GPUs.