ASSIGNMENT 3

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.*

Answer:

Execution Time of 256 x 256 - 0.000181s
Execution Time of 1024 x 64 and 64 x 1024 is 0.000645s
Here as we can see Square matric 256 x 256 is faster. In my opinion, there are two possible reasons:
For 256 x 256 there are less context switches between thread blocks compared to 1024*64 and 64*1024.

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 Times

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 Times

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	43561	29425	59503	Total cycles
gpu_tot_ipc	440.2841	467.5219	397.3059	Instruction per cycle
gpgpu_n_load_insn	524288	262144	131072	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 32-size tile required the least number of global memory access. The 8-size tile resulted in the most global memory access. The number of times that each element loaded from global memory to shared memory equals the number of tiles of the row/column. As a result, a larger tile size leads to fewer tiles and fewer global memory accesses.

6. *Which tile size performed the fastest, which tile size performed the slowest? Why do you think that is?*

Answer: The 16-size tile performed the fastest, while the 32-size tile performed the slowest. 16-size tiles have a higher hardware utilization due to the size limits of shared memory and maximum threads per SM.