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 matrices 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

.

256\*256 kernel execution time is 0.000091s, 1024\*64 and 64\*1024 kernel execution time is 0.000192s. As a result, multiplying a 256\*256 square matrix is faster. Because the 1024\*64 and 64\*1024 rectangularmatrices have more context shifts between thread blocks than the 256\*256 square matrix.

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.

For 64 squaretiled matrix multiplication, each element of the input matrices was loaded four times from global memory.

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?

For 64 squarenon-tiled matrix multiplication, each element of the input matrices was loaded 64 times from global memory.

4. Which tile resulted in the least number of accesses to global memory. What is the reason of this observation.

The 32-tile size resulted in the fewest global memory accesses. The tile size of 8 resulted in the greatest global memory accesses. The number of global memory accesses equals the number of tiles in a row divided by the number of tiles in a column. As a result, if the tile size is higher, we will have fewer tiles and fewer global memory accesses.

5. Which tile perfomed the fastest and which tile performed the slowest

The 16 tile size was the fastest, while the 32 tile size was the slowest. 16 tile size has fewer cycles and may execute more instructions per cycle due to shared memory and maximum thread limits.