Basic Matrix Multiplication

Questions

1. How many floating operations are being performed in your matrix multiply kernel? Explain.

   We have 1 initialization, $2 * \texttt{numAColumns}$ from the for loop (multiplication and addition), and 1 assignment for the output matrix. Thus, for each thread, we have $2 * \texttt{numAColumns} + 2$ floating-point operations, and the kernel performs $(2 * \texttt{numAColumns} + 2) * \texttt{numCRows} * \texttt{numCColumns}$ floating-point operations. Which is obviously $O(n^3)$.

2. How many global memory reads are being performed by your kernel? Explain.

   We have 2 reads at each iteration of the for loop for the input matrices, so we have $2 * \texttt{numAColumns}$ global memory reads for each thread. So the kernel performs $2 * \texttt{numAColumns} * \texttt{numCRows} * \texttt{numCColumns}$ reads.

3. How many global memory writes are being performed by your kernel? Explain.

   Since each thread writes to one element of the output matrix, we have $\texttt{numCRows} * \texttt{numCColumns}$ writes being performed by the kernel.

4. Describe what possible optimizations can be implemented to your kernel to achieve a performance speedup.

   We can use shared memory—i.e. tiling—to reduce the number of global memory reads, and we could also optimize the matrix multiplication by using an algorithm that reduces the number of multiplications and additions needed to compute the output matrix e.g. Strassen's algorithm. Although one caveat with Strassen's algorithm is that it requires $2^n \times 2^n$ matrices, so matrices will have to be padded with zeros which could be wasteful for some matrices.

5. Name three applications of matrix multiplication.

   (a) Image processing: 3D Gaussian splatting in computer graphics requires matrix multiplication to project a '3D Guassian' onto a 2D image for real time rendering.

   (b) Solving eigenvalue problems: E.g. Diagonalizing a matrix with QR decomposition requires iterative matrix multiplication to find the eigenvalues. Furthermore, the QR decomposition also requires matrix multiplication to find the orthonormal and upper triangular matrices e.g. using Householder reflections you need to do a bunch of matrix multiplications $Q = Q_1^T Q_2^T \ldots$, $R = Q_2 Q_1 \ldots A = Q^T A$.

   (c) Image compression: JPEG compresses images using a discrete cosine transform based algorithm (DCT) this also translates to video and audio codec: I believe audio coding formats such as AAC-LP (Apple's favorite audio codec) and Opus (for the Discord users) use some form of the modified discrete cosine transform (MDCT) which tries to circumvent matrix multiplication with linear algebra tricks.

OUTPUT

```
[TIME][GPU][Allocating GPU memory.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 65-71] Elapsed t
ime: 0.245693 ms
[TIME][GPU][Copying input memory to the GPU.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 73-79]
 Elapsed time: 0.055491 ms
[TIME][Compute][Performing CUDA computation][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 86-93]
Elapsed time: 0.399591 ms
[TIME][Copy][Copying output memory to the CPU][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 95-99
] Elapsed time: 0.066102 ms
[TIME][GPU][Freeing GPU Memory][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 101-107] Elapsed tim
e: 0.1501 ms
The solution is correct
[TIME][Generic][Importing data and creating memory on host][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solut
ion.cu: 49-60] Elapsed time: 13.5406 ms
The dimensions of A are 29 x 117
The dimensions of B are 117 x 85
[TIME][GPU][Allocating GPU memory.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 65-71] Elapsed t
ime: 0.255408 ms
[TIME][GPU][Copying input memory to the GPU.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 73-79]
 Elapsed time: 0.055385 ms
[TIME][Compute][Performing CUDA computation][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 86-93]
Elapsed time: 0.346443 ms
[TIME][Copy][Copying output memory to the CPU][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 95-99
] Elapsed time: 0.044165 ms
[TIME][GPU][Freeing GPU Memory][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 101-107] Elapsed tim
e: 0.162507 ms
The solution is correct
[TIME][Generic][Importing data and creating memory on host][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solut
ion.cu: 49-60] Elapsed time: 12.4912 ms
The dimensions of A are 191 x 19
The dimensions of B are 19 x 241
[TIME][GPU][Allocating GPU memory.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 65-71] Elapsed t
ime: 0.257702 ms
[TIME][GPU][Copying input memory to the GPU.][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 73-79]
 Elapsed time: 0.133355 ms
[TIME][Compute][Performing CUDA computation][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 86-93]
Elapsed time: 0.41606 ms
[TIME][Copy][Copying output memory to the CPU][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 95-99
] Elapsed time: 0.206433 ms
[TIME][GPU][Freeing GPU Memory][/home/warehouse/junseo/cuda-code-repo-joonsuuh/Module4/BasicMatrixMultiplication/solution.cu: 101-107] Elapsed tim
e: 0.149427 ms
The solution is correct
[iunseo@r40a-09.sif bin]$
```

Figure 1: `BasicMatrixMultiplication_Solution` output