MP2

1. (Fall 2021) How many floating point operations(+ or *) are performed by your matrix multiply kernel? Answer in terms of numARows, numAColumns, numBRows, numBColumns, numCRows, and/or numCColumns and explain your answer. (1pt for correct answer and 1pt for explanation)

Answer: numCRows * numCColumns * 2 * numAColumns. The output matrix has numCRows * numCColumns elements and 2*numAColumns floating point operations are required to compute the output.

NOTE: Since numCRows = numARows, numCColumns = numBColumns and numAColumns = numBRows, if the student uses a different expression but gets the same answer, he/she should receive full points.

2. How many global memory data bytes TOTAL are being read by your kernel? How many global memory data bytes are being written? Answer in terms of numARows, numAColumns, numBRows, numBColumns, numCRows, and/or numCColumns, and explain your answer. Give the answers for reads and writes separately. **Hint**: size of float is 4 bytes. (1pt for read and 1pt for write)

Answer:
Read- (numCRows * numCColumns) * (numAColumns +numBRows) * 4. The output matrix has numCRows * numCColumns elements. For each output element, the kernel needs to read one row in A and one column in B.
Write- (numCRows * numCColumns) * 4. The output matrix has numCRows * numCColumns elements, and the kernel only needs to write once for each output.

NOTE: Since numCRows = numARows, numCColumns = numBColumns and numAColumns = numBRows, if the student uses a different expression but gets the same answer, he/she should receive full points.

3. If a certain CUDA device's SM (streaming multiprocessor) supports up to 2,048 threads and up to 32 thread blocks simultaneously, which of the following block configurations maximizes parallelism (gives the most threads) in each SM? (1pt)

(A) 16 threads/block
(B) 32 threads/block
(C) 64 threads/block
(D) 100 threads/block

Answer: C. 2048/32 = 64 threads/block

4. You need to process a 128x126 image (128 pixels in the x or horizontal direction, 126 pixels in the y or vertical direction) with the kernel called DiagonalKernel(). More specifically, numCols is 128 and numRows is 126.

```
__global__ void DiagonalKernel(float* d_Pin, float* d_Pout, int numCols, int numRows)
{
       //Calculate row # of image this thread should process
       int row = blockIdx.y*blockDim.y + threadIdx.y;

       //Calculate col # of image this thread should process
       Int col = blockIdx.x*blockDim.x + threadIdx.x;

       //Check if element processed by this thread is within bounds of the image
       if ( (row < numRows) && (col <numCols) ) {
              if (row > col) {
                     d_Pout[row*numCols+col] = d_Pin[row*numCols+col];
              }
              else {
                     d_Pout[row*numCols+col] = 0.0;
              }
       }
}
```

You decide to use a grid of 2D blocks, where each block has a dimension 16x16 threads. How many warps will be generated during the execution of the kernel? (1pt)
(A) 8*8*8
(B) 8*9*8
(C) 8*8*16
(D) 8*9*16
(E) None of the above

Answer: A. ceil((128*126)/(16*16)) = 8*8 blocks are generated, and each block consists of 8 warps.

5. Based on the DiagonalKernel() and block configuration in Qn 4, how many warps will have control divergence? (2pt)
(A) 0
(B) 63
(C) 64
(D) 120
(E) 128

Answer: B. All blocks in the diagonal will have control divergence and there are 8 warps in each block, so the total number of warps in the diagonal is 8*8=64. However, all threads in the last warp of the last block remain idle, so 64-1=63.

6. If we transposed the image in Qn 4 such that we have a 126x128 image (126 pixels in the x or horizontal direction, 128 pixels in the y or vertical direction), keeping the same block configuration, how many warps will have control divergence? (2pt)

(A) 0
(B) 63
(C) 64
(D) 120
(E) 128

Answer: D. All blocks in the diagonal will have control divergence and there are 8 warps in each block, so the number of warps in the diagonal is 8*8=64. All blocks at the right side of the matrix will also have control divergence and there are 8 such blocks, so the number of warps at the right side is 8*8=64. However, when doing the analysis, we count the bottom right block twice, so the final answer is 64+64-8=120.