Started on	Thursday, September 15, 2022, 3:10 PM
State	Finished
Completed on	Thursday, September 15, 2022, 3:39 PM
Time taken	29 mins 28 secs

Question **1**

Complete

Points out of 1.00

For the tiled single-precision matrix multiplication kernel as shown in Lecture 4.4, assume that the tile size is 32X32 and the system has a DRAM burst size of 128 bytes. How many DRAM bursts will be delivered to the processor as a result of loading one A-matrix tile by a thread block?

Select one:

- a. 16
- b. None of the answers
- c. 128
- d. 32 ✓
- e. 64

Question **2**

Complete

Points out of 1.00

For our tiled matrix-matrix multiplication kernel, if we use a 16x16 tile, what is the reduction of memory bandwidth usage for input matrices A and B?

Select one:

- a. 1/16 of the original usage
- b. 1/64 of the original usage
- c. 1/8 of the original usage
- d. 1/32 of the original usage

Question **3**

Complete

Points out of 1.00

We want to use each thread to calculate four (adjacent) output elements of a vector addition. Assume that variable i should be the index for the first element to be processed by a thread. What would be the expression for mapping the thread/block indices to data index of the first element?

Select one:

- a. i=(blockIdx.x*blockDim.x + threadIdx.x)*4;
- b. i=blockIdx.x*blockDim.x + threadIdx.x +4;
- c. i=blockIdx.x*blockDim.x*4 + threadIdx.x;
- d. i=blockIdx.x*threadIdx.x*4;

Question **4**

Complete

Points out of 1.00

We are to process a 600x800 (800 pixels in the x or horizontal direction, 600 pixels in the y or vertical direction) picture with the PictureKernel(). That is n's value is 800 and m's value is 600.

```
__global__ void PictureKernel(float* d_Pin, float* d_Pout, int n, int m) {
    // Calculate the row # of the d_Pin and d_Pout element to process
    int Row = blockIdx.y*blockDim.y + threadIdx.y;
    // Calculate the column # of the d_Pin and d_Pout element to process
    int Col = blockIdx.x*blockDim.x + threadIdx.x;
    // each thread computes one element of d_Pout if in range
    if ((Row < m) && (Col < n)) {
        d_Pout[Row*n+Col] = 2*d_Pin[Row*n+Col];
    }
}
```

Assume that we decided to use a grid of 16x16 blocks. That is, each block is organized as a 2D 16x16 array of threads. How many warps will have control divergence?

Select one:

- a. 37 + 50*8
- b. 38*16
- c. 0
- d. None of the answers
- e. 50

Question **5**

Complete

Points out of 1.00

We are to process a 800x600 (600 pixels in the x or horizontal direction, 800 pixels in the y or vertical direction) picture with the PictureKernel(). That is n's value is 600 and m's value is 800.

```
__global__ void PictureKernel(float* d_Pin, float* d_Pout, int n, int m) {
    // Calculate the row # of the d_Pin and d_Pout element to process
    int Row = blockIdx.y*blockDim.y + threadIdx.y;
    // Calculate the column # of the d_Pin and d_Pout element to process
    int Col = blockIdx.x*blockDim.x + threadIdx.x;
    // each thread computes one element of d_Pout if in range
    if ((Row < m) && (Col < n)) {
        d_Pout[Row*n+Col] = 2*d_Pin[Row*n+Col];
    }
}
```

Assume that we decided to use a grid of 16x16 blocks. That is, each block is organized as a 2D 16x16 array of threads. How many warps will have control divergence?

Select one:

- a. 37+50*8
- b. 0
- c. 50*8
- d. 38*16
- e. None of the answers

50 * 8 warps have divergence

Question **6**

Complete

Points out of 1.00

We are to process a 600x800 (800 pixels in the x or horizontal direction, 600 pixels in the y or vertical direction) picture with the PictureKernel(). That is n's value is 800 and m's value is 600.

```
__global__ void PictureKernel(float* d_Pin, float* d_Pout, int n, int m) {
    // Calculate the row # of the d_Pin and d_Pout element to process
    int Row = blockIdx.y*blockDim.y + threadIdx.y;
    // Calculate the column # of the d_Pin and d_Pout element to process
    int Col = blockIdx.x*blockDim.x + threadIdx.x;
    // each thread computes one element of d_Pout if in range
    if ((Row < m) && (Col < n)) {
        d_Pout[Row*n+Col] = 2*d_Pin[Row*n+Col];
    }
}
```

Assume that we decided to use a grid of 16x16 blocks. That is, each block is organized as a 2D 16x16 array of threads. How many warps will be generated during the execution of the kernel?

Select one:

- a. 38*50
- b. 38*8*50
- c. 37*16
- d. 38*50*2
- e. None of the answers

Question **7**

Complete

Points out of 1.00

Assume a tiled matrix multiplication that handles boundary conditions as explained in Lecture 4.5. Assume that we use 16X16 tiles to process square matrices of 1,000X1,000. What is the total number of warps that will have control divergence due to handling boundary conditions for loading A tiles throughout the kernel execution?

Select one:

- a. 600
- b. 300
- c. 200
- d. 500
- e. None of the answers
- f. 400

1000 / 16 = 62.5
1 block = 8 warp
62.5 block is outside
62.5 * 8 = 500 warp

Question **8**

Complete

Points out of 1.00

Assume that a kernel is launched with 32 thread blocks each of which has 512 threads. If a variable is declared as a shared memory variable, how many versions of the variable will be created through the lifetime of the execution of the kernel?

Select one:

- a. 32*512
- b. 512
- c. 32
- d. 1