

Name _____

1. Matrix Multiplication

Assume a tiled matrix multiplication that handles boundary conditions. Assume that we use 32X32 tiles to process square matrices of 1,000 X 1,000.

- a. How many thread blocks are launched? $\text{ceil}(1000/32) = 32$
- b. How many warps, in the entire kernel, will have control divergence due to handling boundary conditions when writing to the output matrix?
- c. What are the benefits of tiling compared to a simple matrix multiplication?
Please keep your answer short and concise.

1. Matrix Multiplication [20 points total]

- a. [5 points] For an untiled matrix-matrix multiplication kernel, assume we have an M and N matrix of size 32 x 32. How many total global memory loads are issued to matrix M?
- b. [4 points] For the output matrix P, how many stores to global memory are issued?

Name _____

- c. [5 points] For our tiled matrix-matrix multiplication kernel, if we use a 32X32 tile, how many global memory loads are *issued to matrix M* per tile?

- d. [6 points] For the tiled *single-precision (32-bit)* matrix multiplication kernel, 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 M-matrix tile by a thread block? Assume that the tile being loaded is completely within the range of the M-matrix.

- 2. [20 points] For the following, explain how it could harm performance and possible ways the program can be modified to reduce this effect. Please be specific.
 - a. [10 points] The application needs to access global memory to get one value for every operation.
How this harms performance:

Technique/change that could reduce this effect:

Name _____

2. Atomics [8 points]

Atomic operations perform a read-modify-write operation all in a single cycle.

In the following atomic instruction: `atomicAdd(&Sum, 1)`

What is the instruction:

Reading?

Modifying?

Writing?

Why do we need atomic operations? (Keep answer to 1-2 sentences.)

3. Race Conditions [10 points]

Assume I have two threads running the following instructions with `Mem[x] = 2` initially:

thread1:	<code>Old <- Mem[x]</code>	thread2:	<code>Old <- Mem[x]</code>
	<code>New <- Old + 5</code>		<code>New <- Old + 2</code>
	<code>Mem[x] <- New</code>		<code>Mem[x] <- New</code>

a. What are the possible values of `Old`?

b. What are the possible values of `Mem[x]`?

c. Assume the value to be incremented (5 and 2) is stored in a variable, `value`.
What is the corresponding `atomicAdd` call to ensure this example works?

Name _____

Multiple Choice. **Please circle your answer(s).**

4. [3 points] To perform an atomic add operation to add the value of an integer variable Partial to a global memory integer variable Total. Which one of the following statements should be used?

- a. `atomicAdd(Total, &Partial);`
- b. `atomicAdd(&Total, &Partial);`
- c. `atomicAdd(&Total, 1);`
- d. None of the above

5. Assume the following simple matrix multiplication kernel

```
__global__ void MatrixMulKernel(float* M, float* N, float* P, int Width)
{
    int Row = blockIdx.y*blockDim.y+threadIdx.y;
    int Col = blockIdx.x*blockDim.x+threadIdx.x;
    if ((Row < Width) && (Col < Width)) {
        float Pvalue = 0;
        for (int k = 0; k < Width; ++k) {Pvalue += M[Row][k] * N[k][Col];}
        P[Row][Col] = Pvalue;
    }
}
```

Assume a square 1000 x 1000 matrix. Which of the following is true?

- a. Every element in M will be accessed by 1000 different threads
- b. Every element in N will be accessed by 1000^2 different threads
- c. Every element in P will be written to 1000 times
- d. Every element in P will be written to once

Name _____

The following code block shows a histogram kernel

```
__global__ void histogram(unsigned int* input, unsigned int* histo, unsigned
int size, unsigned int num_bins) {
    __shared__ unsigned int private_histo[4096];
    int j= threadIdx.x;
    while (j < 4096) {
        __syncthreads();          <----- A
        private_histo[j] = 0;
        j += blockDim.x;
    }
    __syncthreads();              <----- B
    int i = threadIdx.x + blockIdx.x * blockDim.x;
    int stride = blockDim.x * gridDim.x;
    while (i < size) {
        __syncthreads();          <----- C
        atomicAdd( &(amp;private_histo[input[i]]), 1 );
        i += stride;
    }
    __syncthreads();              <----- D
    j = threadIdx.x;
    while (j < 4096) {
        atomicAdd(&(histo[j]), private_histo[j] );
        j += blockDim.x;
    }
    __syncthreads();              <----- E
}
```

1. In the histogram kernel above which of the sychthreads are required?

- a. A
- b. B
- c. C
- d. D
- e. E

2. In the first while loop, how many times is 0 written in to private_histo[0]?

- a. Not enough information to answer
- b. 0
- c. 32
- d. 4096
- e. 1

Name _____

3. How many atomic add operations are performed on the `private_histo[]` array?

- a. `num_bins`
- b. `4096`
- c. `stride`
- d. `size`
- e. `i`

4. How many atomic add operations are issued to `histo[2019]`?

- a. `num_bins`
- b. `gridDim.x`
- c. `blockDim.x`
- d. `size`
- e. `1`