ECE 408 Study Questions for Exam 2 Fall 2016

- These study questions are meant for you to use as a practice for the exam.
- For each type of question, also make sure that you can answer similar questions on related concepts. For example, you should be able to answer Question 4 for COO and other compression formats
- These questions are not meant to be comprehensive. They are meant to help you to prepare for the portion of the exam on the new material we introduced into ECE408/CS483 this semester. See the Lecture 28 for a list of important topics to fully understand before the exam.
- It would be a poor decision if you choose not use this as a practice partial exam and fully understand how to solve each problem presented before the exam.
- Good luck.

Question 1. Short Answer and Multiple Choice

1. Assume a DRAM system with a burst size of 128 bytes and a peak bandwidth of 240 GB/s. Assume a thread block size of 256 and warp size of 32 and that A is a float array in the global memory. What is the maximal memory data access throughput we can hope to achieve in the following access to A?

```
int i = 8* (blockIdx.x * blockDim.x + threadIdx.x);
float temp = A[i];
```

(A) 240 GB/s
(B) 120 GB/s
(C) 60 GB/s
(D) 30 GB/s

Answer: (D)
Explanation: Each warp is going to access 128 bytes of data that are of pattern where one out of 8 words are used. This is uncoalesced access so the maximal achievable bandwidth is 240/8 = 30 GB/s

2. Given a sparse matrix of integers with m rows, n columns, and a total of k non-zeros. How many integers are needed to represent the matrix in CSR?

(A) m+n+k
(B) 2k +m+1
(C) 2k+n

Name: _____

(D) 3k

Answer: (B)
Each non-zero element needs two integers: value and column. Each row needs an integer: start but we also need the one more at the end

3. Which of the following CUDA API call can be used to perform an asynchronous data transfer?

(A) cudaMemcpy();
(B) cudaAsyncMemcpy();
(C) cudaMemcpyAsync();
(D) cudaDeviceSynchronize();

Answer: (C)

4. For a processor that supports atomic operations in L2 cache, assume that each atomic operation takes 4ns to complete in L2 cache and 100ns to complete in DRAM. Assume that 90% of the atomic operations hit in L2 cache. What is the approximate throughput for atomic operations on the same global memory variable?

(A) 0.225G atomic operations per second
(B) 2.75G atomic operations per second
(C) 0.0735G atomic operations per second
(D) 100G atomic operations per second

Answer: (C)
The average latency is 4ns * 90% + 100ns *10% = 13.6ns. The average throughput is approximately 1/13.6 = 0.0735 G atomic operations per second

5. 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 statement should be used?

(A) atomicAdd(Total, 1);
(B) atomicAdd(&Total, &Partial);
(C) atomicAdd(Total, &Partial);
(D) atomicAdd(&Total, Partial);

Answer: (D)

Name: _____

Explanation: The first argument should be a pointer to the variable to be updated and the second argument should be the variable whose value is to be added to the global variable.

Question 2: Privatization

This question tests your understanding of parallel histogram computation and privatization. Assume that we would like to privatize a histogram that has 512 bins. Each input data value (buffer array elements) will range from 0 to 511. You should handle the case where that there are fewer than 512 threads in each block.

(A) Complete the following kernel to implement the partial privatization of the histogram.

Name: _____

```
__global__ void histo_kernel(unsigned char *buffer,  long size, unsigned int *histo)
{
  __shared__ unsigned int histo_private[1024];
  int i;
  for (                (1)                ) histo_private[i] = 0;
   __syncthreads();
  i = threadIdx.x + blockIdx.x * blockDim.x;
      // stride is total number of threads
  int stride = blockDim.x * gridDim.x;
  while (i < size) {
         atomicAdd(                (2)                );
         i += stride;
   }
  __syncthreads();
 for (                        (3)                        )
       atomicAdd(       (4)                        );
}
```

(1) i = threadIdx.x; i < 512; i+=blockDim.x
(2) &(private_histo[buffer[i]), 1
(3) i = threadIdx.x ; i < 512; i += blockDim.x
(4) &(histo[threadIdx.x]), private_histo[threadIdx.x]

Name: _____

Question 4. Sparse Matrix Multiplication

This question tests your knowledge of Sparse Matrix representation and operation. For your convenience, we are enclosing a lecture slide that illustrates the CSR format and kernel design with a small example.

(A) In the following CSR kernel, fill in the missing indexing expressions for accessing data (input matrix), x (input vector) and y (output vector).

```
1. __global__ void SpMV_CSR(int num_rows, float *data,
   int *col_index, int *row_ptr, float *x, float *y) {
2.   int row = blockIdx.x * blockDim.x + threadIdx.x;
3.   if (row < num_rows) {
4.     float dot = 0;
5.     int row_start = row_ptr[row];
6.     int row_end =   row_ptr[row+1];
7.     for (int elem = row_start; elem < row_end; elem++) {
8.       dot += data[_____  (1) _____] * x[ ____ (2) _____ ];
       }
9.     y[ ____(3)____ ] = dot;
     }
   }
```

(1) elem
(2) col_index[elem]
(3) row

(B) Assume a matrix that has 32 original rows, 64 columns, and 10 non-zeros in every row. After we transform the matrix into CSR-transposed layout, and launch the kernel. Is there any control divergence? Why or why not?

There is no control divergence. All 32 threads will take 10 iterations.

Name: _____

(C) In (B), are the memory accesses to the matrix in the for-loop (line 6) coalesced? Why or why not?

The memory accesses to the matrix are not coalesced. All elements in the same column of the original matrix are laid out far away from each other.

Question 5. Convolution Neural Network

This question tests your understanding of the convolution layer of a CNN. We will start with a basic kernel implementation.

(A) For a 144x64 input feature map, 16x16 tiles and 5x5 convolution filters, if we use the tiled 2D convolution, what this the average number of times that each input feature map element is reused once it is loaded into the shared memory?

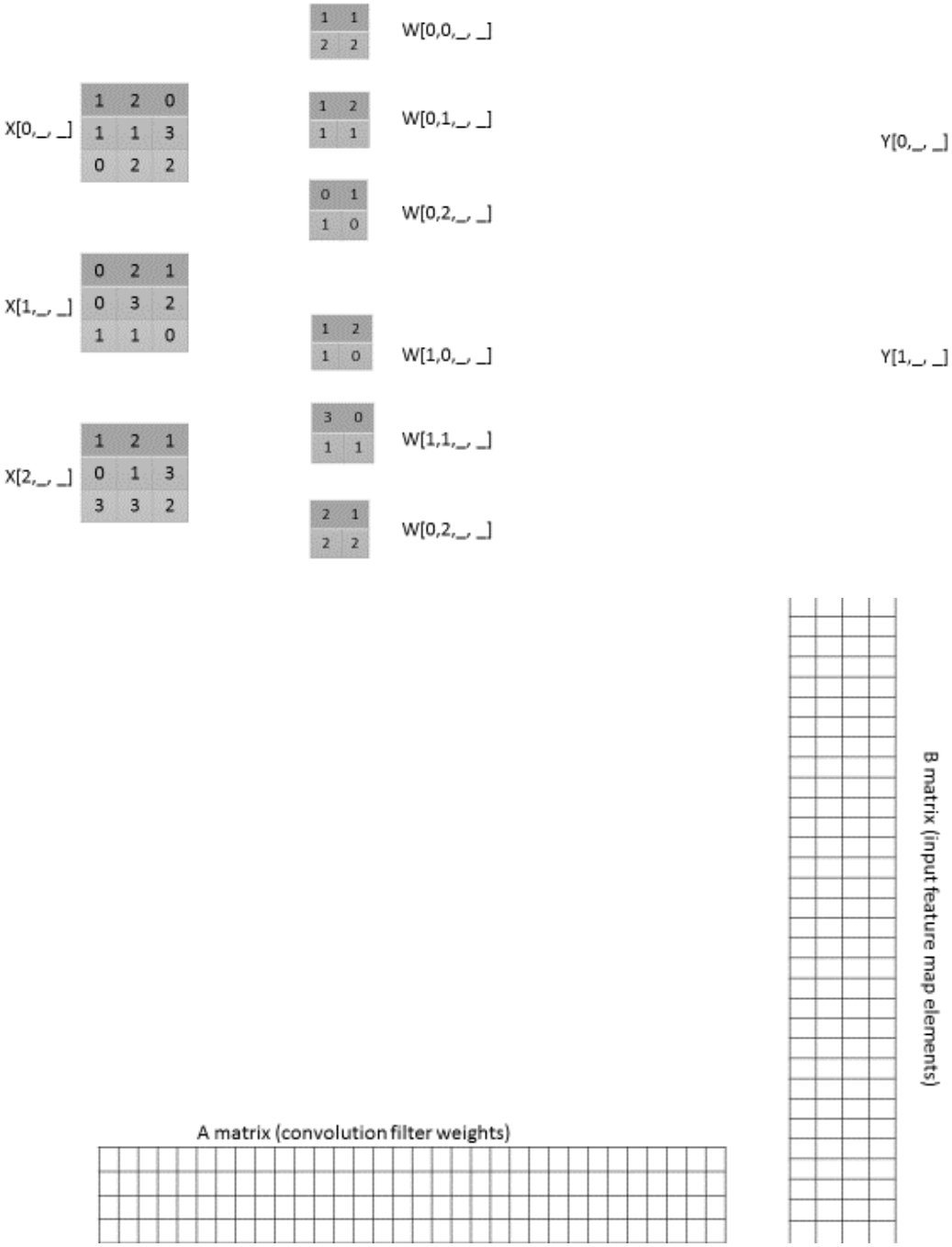
$16^2 * 5^2 / (16+5-1)^2 = 16$

(B) If we use each thread block to generate one tile of output feature map elements, how many thread blocks will be generated when we launch the kernel?

(144/16) * (64/16) = 36 thread blocks

(C) In order to use matrix-multiplication formulation, you need to convert the input feature maps into the unrolled matrix. Assume that we have three 3x3 input feature maps and convolution filter is 3x3. Please fill out the unrolled matrix below based on on the contents of the input feature maps. Use only the entries needed.

Name: _____



See lecture slide

(D) When we form the B matrix, write the kernel to do in order to convert the X tensor into the B matrix.

See lecture slide.

Name: _____

Question 6. Parallelization

Consider the following fragment of C code:

```
float *out, *in;
….
for(unsigned int x = 0; x < 512; ++x) {
   for(unsigned int y = 0; y < 512; ++y) {
      for(unsigned int z = 0; z < 512; ++z) {
         out[y*Width+x] = out[y*Width+x] <OP> in[z*Hight*Width+y*Width+x];
      }
   }
}
```

Explain how you would parallelize this code to achieve a good balance between parallelism and memory access efficiency.

You need to state to what you will assign your blocks and your threads to and why, then write out the kernel function code.

Assume out elements are initialized correctly. Do not worry about the details of variable declarations and types.

Part (a):

x is the contiguous dimension => threads assigned to x dimension
remaining dimension is y => blocks assigned to y dimension
z is the highest dimension, best to stay sequential to ensure coalesced memory access for all warps

Kernel code:

```
unsigned int y = blockIdx.x;
unsigned int x = threadIdx.x;
float tmp = out[y*Width+x]; // Use temporary to avoid global memory access
for(unsigned int z = 0; z < 512; ++z) {
        tmp = tmp <OP> in[z*Width*Height+y*Width+x];
}
 out[y*Width+x] = tmp;
```