CSCI-GA.3033-004

Graphics Processing Units (GPUs): Architecture and Programming

Fall 2016 – (90 minutes)

Last Name: First Name: NetID:

- If you must make assumptions to continue solving a problem, state your assumptions clearly.
- You answer on the question sheet. You can use extra white papers if you want.

(7 questions …. Total of 30 points)

1. [3 points] State three reasons why a GPU version of a code can be slower than a sequential code even though the code has data parallelism.

- Problem size not big enough to produce a lot of parallelism.
- Communication overhead is to high.
- There is no data parallelism.

2. Suppose we want to use the GPU to add two large vectors: C[i] = A[i] + B[i]
Assume that the size of the vectors is very large and they reside in global memory. We want each thread in each block to calculate two elements of the results. Now, for each thread:

a. [2 points] What will be the index of the first element the thread shall access such that the memory can be <u>coalesced</u>? That is, how will i be initialized?

i = threadIdx.x + (blockIdx.x * blockdim.x)*2

b. [2 points] What is the index of the second element to be processed by the thread (also keeping coalescing as a goal)?

i + blockDim.x

3. [2 points] State one advantage and one disadvantage of warps.

- **[Advantage]**
  - Amortize the hardware cost of instruction fetch and decode phases.
  - Simplifies the scheduling

- **[Disadvantage]**
  - Lockstep execution can cause branch divergence
  - Introduces some restrictions on block sizes.

4. 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 m = 600 and n = 800. Assume the two arrays are in the global memory.

```
__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];
   }
}
```

Suppose we want to use 16x16 blocks. That is, each block is 16x16.

a. [2 points] How many warps will be generated? To ensure full credit, show all steps.

ceil(600/16) * ceil(800/16) * (16*16)/32 = 38 * 50 * 8 = 15,200

b. [2 point] How many warps will have thread divergence?

0

c. [1 point] What is the CGMA of the last line of the kernel?

2 mem access and 1 fp operations → 1/2

d. [2 point] Is the CGMA you calculated above a good one or a bad one? Why?

Low … A good CGMA is in the order of 20 or 30 to overcome slow global memory access.

5. Suppose we have the following part of a kernel that will be executed by two threads: T1 and T2. Assume we have only those two threads in this problem.

```
__global__  int x;
__shared__  int y; // initialized to zero
int z;
if (tid == T2){ y = 1; }
if (y == 1) { z = 1; } else { z = 2; }
__syncthreads();
if (tid == T1) x = z;
```

What will be the final value (or possible values if there is more than one) of x if:

a. [2 points] T1 and T2 are in the same warp.

x = 1

b. [2 points] T1 and T2 are in the same block but not in the same warp.

x =1 or x = 2

c. [2 points] T1 and T2 are in two different blocks.

x = 2

d. [4 points] Fill in the table with the number of copies of x, y, and z in each scenario.

Scenario: T1 & T2	copies of x	copies of y	copies of z
same warp	1	1	2
same block but not warp	1	1	2
different blocks	1	2	2

6. [2 points] Do you think GPU supports virtual memory? Justify.

Yes, because two blocks, for example, assigned to the same SM will not see each other's shared memory.

7. [2 points]

a. State one advantage of CUDA over OpenCL

- More mature → more libraries and APIs
- Supports GPU to GPU communication
- Gives the programmer more control.

b. State one advantage of OpenCL over CUDA

- Can be used with other accelerators than GPUs.