King Saud University College of Computer and Information Sciences Department of Computer Science CSC453 – Parallel Processing – Tutorial No 5bis – Quarter 3 2023

Question

Let's consider 2 arrays of integers A and B of size *N*. Let's consider that we would like to write a C program that runs in parallel and that computes the sum of 2 arrays as following:

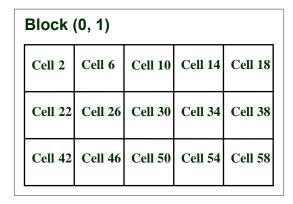
$$C[i] = A[i] + B[i]$$

Let's consider the following kernel:

1. We would like to run this kernel on **2-D grid of blocks** each of which is of **2-D matrix of threads**. Every thread evaluates a single cell as shown in the following figure:

Block (0, 0)				
Cell 0	Cell 4	Cell 8	Cell 12	Cell 16
Cell 20	Cell 24	Cell 28	Cell 32	Cell 36
Cell 40	Cell 44	Cell 48	Cell 52	Cell 56

Block (1, 0)				
Cell 1	Cell 5	Cell 9	Cell 13	Cell 17
Cell 21	Cell 25	Cell 29	Cell 33	Cell 37
Cell 41	Cell 45	Cell 49	Cell 53	Cell 57



Block (1, 1)				
Cell 3	Cell 7	Cell 11	Cell 15	Cell 19
Cell 23	Cell 27	Cell 31	Cell 35	Cell 39
Cell 43	Cell 47	Cell 51	Cell 55	Cell 59

King Saud University College of Computer and Information Sciences Department of Computer Science CSC453 – Parallel Processing – Tutorial No 5bis – Quarter 3 2023

 Give the formula that allows every thread to compute the cell_id of the cell he is going to process.

2. We would like to run this kernel on grid composed of a single 2-D thread block (1 block where threads are organized as a 2-D matrix). Every thread evaluates a single cell as shown in the following figure:

Block (0, 0)				
Cell 03	Cell 3	Cell 6	Cell 9	Cell 12
Cell 17	Cell 4	Cell 7	Cell 10	Cell 13
Cell 3	Cell 5	Cell 8	Cell 11	Cell 14

 Give the formula that allows every thread to compute the cell_id of the cell he is going to process.

King Saud University College of Computer and Information Sciences Department of Computer Science

CSC453 - Parallel Processing - Tutorial No 5bis - Quarter 3 2023

```
mapping
for (int i = 0; i < N; i++)
  doSomthing(arr[i])
when calling a func in kernal u should __device__
 _device__ void doSomthing(int x ) {}
int main(void){
int count;
cudaGetDeviceCount(&count);
cudaDeviceProp prop;
int max = 0;
int current = 0;
for(int i = 0; i < count; i++){
  cudaGetDeviceProperties(&prop, i);
  if (prop.multiProcessorCount > max)
     max = prop.multiProcessorCount;
     current = i;
}
cudaSetDevice(current);
kernal<<<N, N>>>(d_a, d_b, d_c);
}
```

King Saud University College of Computer and Information Sciences Department of Computer Science CSC453 – Parallel Processing – Tutorial No 5bis – Quarter 3 2023