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

### **Question 1**

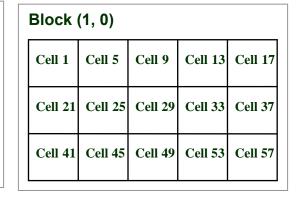
Let's consider 2 arrays of integers A and B of size *N*. Let's consider that we would like to write a C program that runs in parallel and that computes the sum of 2 arrays as following:

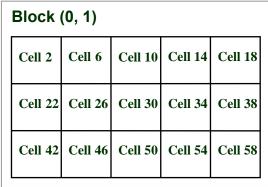
$$C[i] = A[i] + B[i]$$

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	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		





E	Block	ck (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.

### **Question 2**

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

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

We would like to run this kernel on a grid composed of **1 block** where every thread evaluates W cells as shown in the following figure (where W = 4 as a sample):

Block	(0, 0)			
Cells	Cells	Cells	Cells	<b>Cells</b> 16-19
0-3	4-7	8-11	12-15	
Cells	Cells	Cells	Cells	Cells 36-39
20-23	24-27	28-31	32-35	
Cells	Cells	Cells	Cells	<b>Cells</b> 56-60
40-43	43-47	48-51	52-55	

Write the kernel

\_\_global\_\_ void add(int \*a, int \*b, int \*c, int size, int N, int W) {

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

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

```
__global__ void kernal(int N, int *arr){
int index = blockldx.x * blockDim.x + threadIdx.x
if (index < N) doSomthing(arr[index]);
}</pre>
```

#### **Question 3**

Let's consider an array of integers of size N, denoted Data. Let's consider the following sequential iteration:
 \_\_\_device\_\_\_\_ void doSomthing(int a){...}

- a. Write the corresponding parallel code using CUDA C.
- b. Update your kernel such that every threads operates W cells.
- 2. We would like to write a kernel that (1) receives an integer A, two arrays of size N denoted X et Y respectively and (2) calculates and stores the A\*X + Y in an array of size N denoted C. We consider that the elements of the array C are calculated as follows:

$$C[i] = A * X[i] + Y[i]$$
  
So, we would like to write a kernel that receives 5 parameters:   
- An integer denoted **A** (**input**),   
- An array of integers denoted **X** (**input**),   
- An array of integers denoted **X** (**input**),   
-  $C[i] = A * X[i] + Y[i]$    
-  $C[i] = A * X[i] + Y[i]$ ;

- An array of integers, denoted **C** (**output**),

An array of integers denoted **Y** (**input**),

- An integer denoted **N** which represent the size of arrays X, Y and C (**input**),
- a. We would like to run this kernel within a grid of blocks (organized as 1-D array) each of which is a 1-D array of threads. We would like that <u>every thread</u> calculates a single cell of the result (**the array C**). Give the code of the kernel.
- b. We would like to run this kernel within a grid of blocks (organized as 1-D array) each of which is a 1-D array of threads. We would like that <u>every thread</u> <u>calculates the elements of a sub-array (of the array C) of size *width*. Give the code of the kernel.</u>