600086 Lab Book

# Week 2 – Lab B

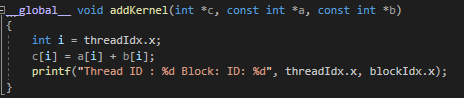
Date: 17th Feb 2022

Callum Gray

## Q1. Understand the block and thread indices

### Question:

### Solution:



In the below image each add kernel function is run separately and the others commented out .

### Text Description automatically generated

### Test data:



### Sample output:

|  |  |
| --- | --- |
| Thread Config | Output |
| addKernel <<<1, 5 >>> (dev\_c, dev\_a, dev\_b); |  |
| addKernel <<<2, 3 >>> (dev\_c, dev\_a, dev\_b); |  |
| addKernel <<<3, 2 >>> (dev\_c, dev\_a, dev\_b); |  |
| addKernel <<<6, 1 >>> (dev\_c, dev\_a, dev\_b); |  |

### Reflection:

The block number indicates which thread block the thread is located in and the thread number indicates which thread of the block is currently executing. This could be useful to identify threads when in a large-scale process where the parallel processing becomes more complex.

### Metadata:

Threadidx.x, blockidx.x

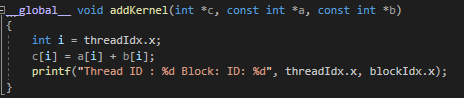
### Further information:

N/A

## Q2. Exercise 2. Find vector addition using multiple 1D thread blocks

### Question: Text Description automatically generated

### Solution:



In the below image each add kernel function is run separately and the others commented out .

### A picture containing text Description automatically generated

### Test data:



### Sample output:

|  |  |
| --- | --- |
| Thread Config | Output |
| addKernel <<<2, 3 >>> (dev\_c, dev\_a, dev\_b); |  |
| addKernel <<<3, 2 >>> (dev\_c, dev\_a, dev\_b); |  |
| addKernel <<<6, 1 >>> (dev\_c, dev\_a, dev\_b); |  |

### Reflection:

The output is limited by the number of threads generated in in each block as the calculation uses the thread ID to index each array. SO if more than 5 arrays are used then an error will be raised.

### Metadata:

Multithread

### Further information:

N/A

## Q3. Exercise 3. Understand the thread indices for 2D blocks

### Question:

A picture containing text, person, screenshot

Description automatically generated

### Solution:

Text

Description automatically generated

In the below image each add kernel function is run separately and the others commented out .

### Text Description automatically generated with medium confidence

### Test data:



### Sample output:

|  |  |
| --- | --- |
| Thread Config | Output |
| addKernel << <1, dim3(2,3) >> > (dev\_c, dev\_a, dev\_b); |  |
| addKernel << <1, dim3(3,3) >> > (dev\_c, dev\_a, dev\_b); |  |
| addKernel << <1, dim3(5,1) >> > (dev\_c, dev\_a, dev\_b); |  |

### Reflection:

The aim of the above task is to display how to Identify individual threads within a multi-dimensional block. Each thread has a unique Id based on its location in the thread which can be thought of as a coordinate to locate it e.g 2d array of threads has threadIdx.x and threadIDx.y and a 3D array of threads has threadIDx.x ,threadIdx.y and threadIdx.z to locate it.

### Metadata:

Multithread threadId.x threadId.y threadId.z

### Further information:

N/A

## Q4. Exercise 4. Find vector addition using one 2D thread block

### Question:

### Text Description automatically generatedSolution:

A screenshot of a computer

Description automatically generated with medium confidenceIn the below image each add kernel function is run separately and the others commented out.

Text

Description automatically generated

### Test data:

### 

### Sample output:

|  |  |
| --- | --- |
| Thread Config | Output |
| addKernel << <1, dim3(2,3) >> > (dev\_c, dev\_a, dev\_b); |  |
| addKernel << <1, dim3(3,2) >> > (dev\_c, dev\_a, dev\_b); |  |
| addKernel << <1, dim3(5,1) >> > (dev\_c, dev\_a, dev\_b); |  |

### Reflection:

The tasks’ goal was to perform vector addition using a multi-dimensional array of threads however the vector addition only occure3d on the column number for the thread blocks.

### Metadata:

Multithread threadId.x threadId.y threadId.z

### Further information:

Unsure how to resolve issue of only adding columns do we need to add an id checking method that will adapt to include rows as an overflow when there aren’t enough columns?

## Q5. Exercise 5. Find vector addition using multiple 2D thread blocks

## Question:

Text

Description automatically generated

### Solution:

### Text Description automatically generated

In the below image each add kernel function is run separately and the others commented out.

Text

Description automatically generated

### Test data:



### Sample output:

|  |  |
| --- | --- |
| Thread Config | Output |
| addKernel << < dim3(1, 3), dim3(3, 1) >> > (dev\_c, dev\_a, dev\_b); |  |
| addKernel << < dim3(2, 3), dim3(2, 2) >> > (dev\_c, dev\_a, dev\_b); |  |
| addKernel << < dim3(2, 2), dim3(2, 3) >> > (dev\_c, dev\_a, dev\_b); |  |
|  |  |
|  |  |
|  |  |
|  |  |

### Reflection:

This is threading 101

### Metadata:

Threads

### Further information:

Unsure of the use of mut?