**College of Computer Science and Information Technology**
**Universiti Tenaga Nasional (UNITEN)**
**CSNB594: Parallel Computing**
**Lab 10 – CUDA Part 2**

**Student ID:** **Student Name:**

Topics: Compiling C program with CUDA enabled on Windows using MS Visual Studio. In this session, we are going to learn to write parallel program using C language.

| # | Course Outcome | Coverage |
|---|---|---|
| CO 1 | Design parallel code to solve a given problem, determine computational bottlenecks and optimize the performance of the code. | ✓ |
| CO 2 | Describe different parallel architectures, programming models, and algorithms for parallel programming. | |
| CO 3 | Analyze the time complexity of parallel algorithms as a function of the problem size and number of processors. | ✓ |
| CO 4 | Implement parallel solutions using Pthreads, OpenMP, MPI, Hybrid or GPU. | ✓ |

Marks Allocation:

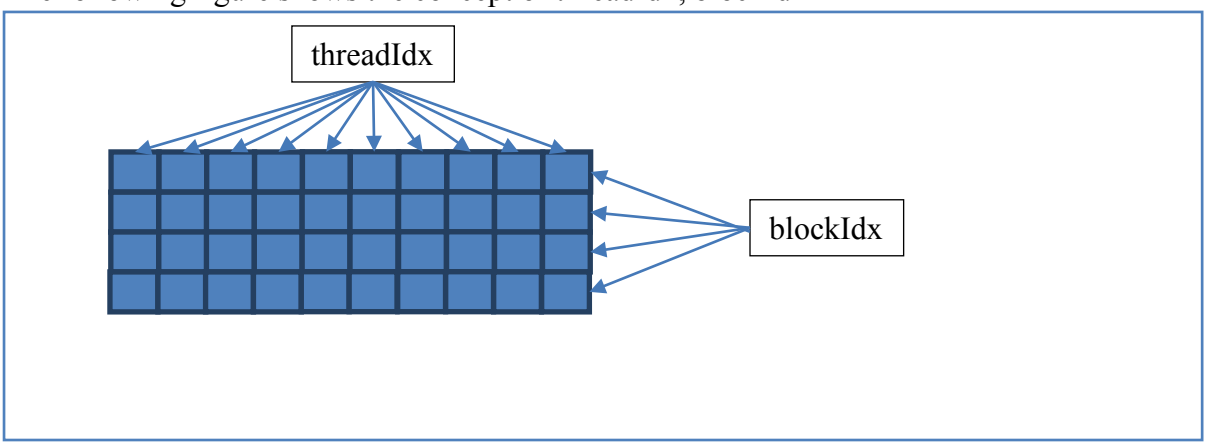
| Question/Activity | Marks Allocation |
|---|---|
| ACTIVITY 1 | 2 |
| ACTIVITY 2 | 7 |
| ACTIVITY 3 | 7 |
| ACTIVITY 4 | 6 |
| **Total** | **22** |

## WINDOWS

Here are some quick tips on CUDA:

1. One (1) block can consists up to 512 threads (old GPU) and 1024 threads (new GPU).
2. Kernel launcher,

   <<< NUM_OF_BLOCKS, NUM_OF_THREADS_PER_BLOCK >>>
3. The following figure shows the concept of threadIdx, blockIdx



## ACTIVITY 1: SIMPLE CUDA PROGRAM

You are given the following source code.

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>
#include <stdlib.h>
#include <time.h>


__global__ void addKernel(int *d_out, int *d_in)
{

	int i = threadIdx.x;
	int newValue = d_in[i];
	d_out[i] = newValue + 1;

}

int main()
{
	const int arraySize = 10;
	const int arrayByte = arraySize * sizeof(int);

	int host_input[arraySize];
	int host_output[arraySize];

	int i;

	int *dev_in = 0;
	int *dev_out = 0;

	/*Start:keep track execution duration*/
	time_t begin, end;
	begin = time(NULL);

	for (i = 0; i < arraySize; i++)
	{
		host_input[i] = i;
	}

	printf("initial value:\n");
	for (i = 0; i < arraySize; i++)
	{
		printf("host[%d]: %d\n", i, host_input[i]);
	}

	// Allocate GPU buffers for three vectors (two input, one output)    .
	cudaMalloc((void**)&dev_in, arrayByte);
	cudaMalloc((void**)&dev_out, arrayByte);

	// Copy input vectors from host memory to GPU buffers.
	cudaMemcpy(dev_in, host_input, arrayByte, cudaMemcpyHostToDevice);

	// Launch a kernel on the GPU with one thread for each element.
	addKernel << <1, arraySize >> >(dev_out, dev_in);

	// Copy output vector from GPU buffer to host memory.
```

```
        cudaMemcpy(host_output, dev_out, arrayByte, cudaMemcpyDeviceToHost);

        printf("----- After GPU -----\n");
        for (i = 0; i < arraySize; i++)
        {
                printf("%d\n", host_output[i]);
        }

        /*End:keep track execution duration*/
        end = time(NULL);
        printf("Duration %f seconds\n", difftime(end, begin));

        cudaFree(dev_out);
        cudaFree(dev_in);

    // cudaDeviceReset must be called before exiting in order for profiling and
    // tracing tools such as Nsight and Visual Profiler to show complete traces.
   cudaDeviceReset();

    return 0;
}
```

You may compile and see the output of this program.

**Activity 1, Question 1**

How many block will be created in this program?

Answer:

**Activity 1, Question 2**

How many thread will be created in each block?

Answer:

## ACTIVITY 2: EXTEND PROBLEM

Modify the source code in Activity 1 to meet the following criteria:

1. Create a new array.
2. The value of each element in this array (1) should be the current array index + 10.
3. The array created in (1) should be passed during the kernel launcher too.
4. The addKernel function should receive one (1) more value, which is the array created in (1).
5. Perform addition between each element of the both arrays.

Modified Source code:

Screen shot output:

## ACTIVITY 3: MULTIPLE BLOCKS

Modify the source code in Activity 2 to meet the following criteria:

1. Increase the elements of array to 1000.
2. Assume that we are using the old version of GPU.
3. Based on our data size, determine:
    a. How many block should be created?

    Answer:

    b. How many threads should be created in each block?

    Answer:
4. To access data in each index for multiple blocks and threads, you need to use this formula:

```
int i = threadIdx.x + blockIdx.x * blockDim.x;
```

5. Modify your source code to meet the new array size and its elements, the number of block, and thread.

| Modified Source code: |
| --- |
| Screen shot output: |

## ACTIVITY 4 : INCREASE DATA

Modify the source code in Activity 2 to meet the following criteria:

1. Increase the elements of array to 30000.
2. Assume that we are using the new version of GPU.
3. Based on our data size, determine:
    a. How many block should be created?

    Answer:

    b. How many threads should be created in each block?

    Answer:
4. Modify your source code to meet the new array size and its elements, the number of block, and thread.

| Modified Source code: |
| --- |
| Screen shot output: |

**Submission:**
Complete today's activity by submission of this file thru Google Classroom. Name your file as:
**CSNB594_Lab10 YourStudentID YourFullName.docx**