**Lab Exercise 6.1 – CUDA Programming Model**

**Objective:**  
To understand and implement a basic CUDA program that demonstrates host-device interaction, kernel launch, and thread indexing using the CUDA programming model.

Part 1: Key Concepts

| **Concept** | **Description** |
| --- | --- |
| Host | The CPU (runs regular C++ code) |
| Device | The GPU (runs CUDA kernels) |
| Kernel | A function executed on the device in parallel by many threads |
| Thread | Basic unit of execution in CUDA |
| Block | Group of threads that can share memory |
| Grid | Group of blocks that cover the overall workload |

**Part 2: Sample CUDA Program**

Filename: **hello\_cuda.cu**

#include <iostream>

// Kernel function that runs on the GPU

\_\_global\_\_ void hello\_from\_gpu() {

printf("Hello from thread [%d, %d]!\n", blockIdx.x, threadIdx.x);

}

int main() {

std::cout << "Hello from the CPU!\n";

// Launch the kernel with 2 blocks, each having 4 threads

hello\_from\_gpu<<<2, 4>>>();

// Wait for GPU to finish before proceeding

cudaDeviceSynchronize();

std::cout << "Back to CPU after GPU execution.\n";

return 0;

}

Explanation:

* \_\_global\_\_ indicates that the function is a CUDA kernel.
* <<<2, 4>>> launches the kernel with 2 blocks, each having 4 threads.
* blockIdx.x gives the block index.
* threadIdx.x gives the thread index within its block.
* cudaDeviceSynchronize() ensures that the CPU waits for GPU tasks to complete.

How to Compile and Run:

nvcc hello\_cuda.cu -o hello\_cuda

./hello\_cuda

**Expected Output:**

Hello from the CPU!

Hello from thread [0, 0]!

Hello from thread [0, 1]!

Hello from thread [0, 2]!

Hello from thread [0, 3]!

Hello from thread [1, 0]!

Hello from thread [1, 1]!

Hello from thread [1, 2]!

Hello from thread [1, 3]!

Back to CPU after GPU execution.