**Lab Exercise 6.2 – Increase the Number of Blocks and Threads in a CUDA Program**

**Objective:**  
To modify a CUDA kernel to use more blocks and threads, understand how thread and block indexing work, and observe how the thread hierarchy scales.

**Sample CUDA Program**

**Filename:** **scaled\_threads.cu**

#include <iostream>

// CUDA kernel

\_\_global\_\_ void identify\_thread() {

int global\_id = blockIdx.x \* blockDim.x + threadIdx.x;

printf("Hello from Block %d, Thread %d, Global ID: %d\n", blockIdx.x, threadIdx.x, global\_id);

}

int main() {

std::cout << "Launching more blocks and threads...\n";

int numBlocks = 4;

int threadsPerBlock = 8;

identify\_thread<<<numBlocks, threadsPerBlock>>>();

cudaDeviceSynchronize();

std::cout << "Execution complete.\n";

return 0;

}

**Explanation:**

* numBlocks = 4: Four blocks will be created.
* threadsPerBlock = 8: Each block will contain eight threads.
* global\_id = blockIdx.x \* blockDim.x + threadIdx.x: Computes a unique thread ID across the entire grid.
* Output includes each thread's block, thread index within the block, and global thread ID.

**How to Compile and Run:**

nvcc scaled\_threads.cu -o scaled\_threads

./scaled\_threads

**Expected Output (shortened):**

Launching more blocks and threads...

Hello from Block 0, Thread 0, Global ID: 0

Hello from Block 0, Thread 1, Global ID: 1

...

Hello from Block 3, Thread 7, Global ID: 31

Execution complete.