**Lab Exercise 11.1 – Memory Objects – Global, Shared, and Local Memory in CUDA**

**Objective:**

* Understand the different types of memory in CUDA: **Global**, **Shared**, and **Local**.
* Learn how memory types impact performance.
* Write a CUDA program to observe behavior and usage of these memory spaces.

**1. CUDA Memory Types Overview**

| **Memory Type** | **Scope** | **Lifetime** | **Speed** | **Accessed By** |
| --- | --- | --- | --- | --- |
| Global | All threads | Application | Slow | All threads |
| Shared | Block | Block duration | Fast | Block threads |
| Local | Thread | Thread lifetime | Slower | Single thread |

**2. CUDA Program Demonstrating Global, Shared, and Local Memory**

#include <iostream>

\_\_global\_\_ void memoryExampleKernel(int \*globalInput, int \*globalOutput) {

// Local memory: each thread has its own copy

int localValue = globalInput[threadIdx.x];

// Shared memory: shared among threads in the same block

\_\_shared\_\_ int sharedData[256];

sharedData[threadIdx.x] = localValue;

\_\_syncthreads(); // Synchronize threads to ensure all writes to shared memory are done

// Access shared memory and store into global output

globalOutput[threadIdx.x] = sharedData[threadIdx.x];

}

int main() {

const int N = 256;

int h\_input[N], h\_output[N];

int \*d\_input, \*d\_output;

for (int i = 0; i < N; ++i)

h\_input[i] = i;

cudaMalloc((void\*\*)&d\_input, N \* sizeof(int));

cudaMalloc((void\*\*)&d\_output, N \* sizeof(int));

cudaMemcpy(d\_input, h\_input, N \* sizeof(int), cudaMemcpyHostToDevice);

memoryExampleKernel<<<1, N>>>(d\_input, d\_output);

cudaMemcpy(h\_output, d\_output, N \* sizeof(int), cudaMemcpyDeviceToHost);

std::cout << "Output from globalOutput:\n";

for (int i = 0; i < 10; ++i)

std::cout << h\_output[i] << " ";

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

cudaFree(d\_input);

cudaFree(d\_output);

return 0;

}

**3. Explanation**

* globalInput and globalOutput: Stored in **global memory** (slow but accessible by all threads).
* localValue: A local variable used **per thread**; resides in **registers or local memory**.
* sharedData[]: Allocated using \_\_shared\_\_; shared by all threads in the **same block**, very **fast** access.
* \_\_syncthreads(): Synchronizes all threads in the block to ensure consistent shared memory access.

**4. Compilation and Execution**

nvcc -o memory\_types memory\_types.cu

./memory\_types

**5. Summary and Learning Points**

* **Global memory** is flexible but slow — best used when necessary for inter-block communication or data persistence.
* **Shared memory** is extremely fast and best for inter-thread communication within a block.
* **Local memory** is private to a thread and used when registers are not sufficient.

**6. Further Exploration**

* Experiment with more blocks and shared memory size limits.
* Profile execution time when using shared memory vs only global memory.
* Observe shared memory bank conflicts with different access patterns.