ใบงานการทดลองที่ 5 Memory Management in CUDA

วัตถุประสงค์การเรียนรู้ (Learning Outcomes)

1. เข้าใจความแตกต่างระหว่าง Global Memory, Shared Memory, Register Memory ของ GPU
2. เขียนโปรแกรม CUDA ที่ใช้ memory แบบต่าง ๆ
3. วัดและเปรียบเทียบ Execution Time ของ kernel ที่ใช้ memory ต่างชนิดกัน
4. วิเคราะห์ผลกระทบของ memory type ต่อ performance และ latency

โจทย์หลัก (Problem Statement)

นักศึกษาทดลองเขียนโปรแกรม **การบวกเวกเตอร์ (Vector Addition)** ขนาดใหญ่ โดยใช้ **memory แบบต่าง ๆ** ของ CUDA

โจทย์: “เปรียบเทียบเวลาในการประมวลผลของ kernel ที่ใช้ Global Memory, Shared Memory และ Register Memory สำหรับ Vector Addition”

โดยกำหนดขนาดข้อมูล (Data Size)

- Vector A, B, C เท่ากับ 10^6 elements (1000000)
- Data type: float

ตัวอย่าง Kernel

1 ใช้ **Global Memory**

```
__global__ void vectorAddGlobal(float* A, float* B, float* C, int N) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    if (idx < N)
        C[idx] = A[idx] + B[idx];
}
```

2 ใช้ **Shared Memory**

```
__global__ void vectorAddShared(float* A, float* B, float* C, int N) {
    extern __shared__ float temp[];
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    int tid = threadIdx.x;

    if (idx < N) {
        temp[tid] = A[idx] + B[idx];
        __syncthreads(); // sync threads in the block
        C[idx] = temp[tid];
    }
}
```

*** **หมายเหตุ: extern __shared__ กำหนดขนาด shared memory เมื่อ launch kernel**

3 ใช้ **Register Memory**

```
__global__ void vectorAddRegister(float* A, float* B, float* C, int N) {
   int idx = blockIdx.x * blockDim.x + threadIdx.x;
   if (idx < N) {
      float a = A[idx];  // stored in register
      float b = B[idx];  // stored in register
      C[idx] = a + b;
   }
}
```

** ตัวแปรใน kernel local จะถูกเก็บใน **register** หาก resource เพียงพอ

การรันโปรแกรม ** ใช้ cudaEventRecord() วัดเวลา kernel execution

```
int threadsPerBlock = 256;
int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;

// Global Memory
vectorAddGlobal<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);

// Shared Memory
vectorAddShared<<<blocksPerGrid, threadsPerBlock, threadsPerBlock *
sizeof(float)>>>(d_A, d_B, d_C, N);

// Register Memory
vectorAddRegister<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
```

Kernel สำหรับการตรวจวัดเวลา

```
cudaEventRecord(start);
// kernel launch
cudaEventRecord(stop);
cudaEventSynchronize(stop);
cudaEventElapsedTime(&milliseconds, start, stop);
```

ตารางบันทึกผลการทดลอง

Memory Type	Threads/Block	Blocks	Execution Time (ms)
Global Memory	256	1000	
		2000	
		3097 (คำนวณ)	
		8000	
Shared Memory	256	1000	
		2000	
		3097 (คำนวณ)	
		8000	
Register Memory	256	1000	
		2000	
		3097 (คำนวณ)	
		8000	

สรุปและวิเคราะห์การทดลอง

.....

.....

.....

.....

คำถามวิเคราะห์ (Analysis Questions)

1. Kernel ไหนทำงานเร็วที่สุด? เพราะอะไร?
2. Global Memory ช้าเมื่อเทียบกับ Shared Memory / Register Memory อย่างไร?
3. ขนาด block และ threads ส่งผลต่อ performance อย่างไร?
4. ทำไมการใช้ Shared Memory ถึงช่วยลด latency และเพิ่ม bandwidth?
5. ถ้า kernel ใช้ Register มากเกินไป จะเกิดอะไรขึ้น? (Spill to local memory)

สรุปหลักการและวิธีการใช้งาน Memory แต่ละประเภทใน CUDA

.....

.....

.....

.....

.....

.....

.....

.....