4a Vector Addition

#include <iostream>

#include <cuda\_runtime.h>

\_\_global\_\_ void addVectors(int\* A, int\* B, int\* C, int n) {

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

if (i < n) {

C[i] = A[i] + B[i];

}

}

int main() {

int n = 1000000;

int \*A, \*B, \*C;

int size = n \* sizeof(int);

// Allocate memory on the host

cudaMallocHost(&A, size);

cudaMallocHost(&B, size);

cudaMallocHost(&C, size);

// Initialize vectors

for (int i = 0; i < n; i++) {

A[i] = i;

B[i] = i \* 2;

}

// Allocate memory on the device

int \*dev\_A, \*dev\_B, \*dev\_C;

cudaMalloc(&dev\_A, size);

cudaMalloc(&dev\_B, size);

cudaMalloc(&dev\_C, size);

// Copy data from host to device

cudaMemcpy(dev\_A, A, size, cudaMemcpyHostToDevice);

cudaMemcpy(dev\_B, B, size, cudaMemcpyHostToDevice);

// Launch kernel

int blockSize = 256;

int numBlocks = (n + blockSize - 1) / blockSize;

addVectors<<<numBlocks, blockSize>>>(dev\_A, dev\_B, dev\_C, n);

// Copy results back

cudaMemcpy(C, dev\_C, size, cudaMemcpyDeviceToHost);

// Print results

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

std::cout << C[i] << " ";

}

std::cout << std::endl;

// Free memory

cudaFree(dev\_A);

cudaFree(dev\_B);

cudaFree(dev\_C);

cudaFreeHost(A);

cudaFreeHost(B);

cudaFreeHost(C);

return 0;

}

Command to write to run code in terminal

nvcc vector\_add.cu -o vector\_add

./vector\_add