#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cuda\_runtime.h>

#include <chrono>

#include <ctime>

#include <math.h>

using namespace std;

\_\_global\_\_ void Min(float\* InputArray, int ArraySize) {

int block = blockIdx.x;

float stddevf = InputArray[0];

for (int i = block; i < ArraySize; i++) {

stddevf = min(InputArray[i], stddevf);

}

InputArray[0] = stddevf;

}

\_\_global\_\_ void Max(float\* InputArray, int ArraySize) {

int block = blockIdx.x;

float stddevf = 0.0;

for (int i = block; i < ArraySize; i++) {

stddevf = max(InputArray[i], stddevf);

}

InputArray[0] = stddevf;

}

\_\_global\_\_ void Average(float\* InputArray, int ArraySize) {

int block = blockIdx.x;

float stddevf = 0.0;

for (int i = block; i < ArraySize; i++) {

stddevf += InputArray[i];

}

InputArray[0] = stddevf/ArraySize;

}

\_\_global\_\_ void Sum(float\* InputArray, int ArraySize) {

int block = blockIdx.x;

float stddevf = 0.0;

for (int i = block; i < ArraySize; i++) {

stddevf += InputArray[i];

}

InputArray[0] = stddevf;

}

\_\_global\_\_ void stddev(float\* InputArray, int ArraySize, float mean) {

int block = blockIdx.x;

float stddevf=0.0;

for (int i = block; i < ArraySize; i++) {

stddevf += (InputArray[i] - mean) \* (InputArray[i] - mean);

}

stddevf = stddevf/ ArraySize;

InputArray[0] = stddevf;

}

int main() {

//Read Array Size From User

int ArraySize = -1;

printf("Enter The Number Of Elements: : ");

scanf("%d", &ArraySize);

if (ArraySize <= 0)

return 0;

//Declare The Float Array

float\* h\_Array = new float[ArraySize];

printf("The random elements in the array are-\n");

srand(time(0));

for (int i = 0; i < ArraySize; i += 1) {

h\_Array[i] = rand();

printf("%f ", h\_Array[i]);

}

std::chrono::time\_point<std::chrono::system\_clock> start, end;

int ArrayMemory = ArraySize \* sizeof(float);

//int ThreadBlockSize = (ArraySize + 1) >> 1;

float\* d\_Array;

float result;

float stdev;

cudaMalloc(&d\_Array, ArrayMemory);

start = std::chrono::system\_clock::now();

// Copy Array To GPU For Minimum Function

cudaMemcpy(d\_Array, h\_Array, ArrayMemory, cudaMemcpyHostToDevice);

Min << <ArraySize,1 >> > (d\_Array, ArraySize);

cudaMemcpy(&result, d\_Array, sizeof(float), cudaMemcpyDeviceToHost);

printf("The Minimum Value In The Array: : %f\n", result);

// Copy Array To GPU For Maximum Function

cudaMemcpy(d\_Array, h\_Array, ArrayMemory, cudaMemcpyHostToDevice);

Max << <ArraySize,1 >> > (d\_Array, ArraySize);

cudaMemcpy(&result, d\_Array, sizeof(float), cudaMemcpyDeviceToHost);

printf("The Maximum Value In The Array: : %f\n", result);

// Copy Array To GPU For Sum Function

cudaMemcpy(d\_Array, h\_Array, ArrayMemory, cudaMemcpyHostToDevice);

Sum << <ArraySize,1 >> > (d\_Array, ArraySize);

cudaMemcpy(&result, d\_Array, sizeof(float), cudaMemcpyDeviceToHost);

printf("The Sum Of Numbers In The Array: : %f\n", result);

// Copy Array To GPU For Average Function

cudaMemcpy(d\_Array, h\_Array, ArrayMemory, cudaMemcpyHostToDevice);

Average << <ArraySize,1 >> > (d\_Array, ArraySize);

cudaMemcpy(&result, d\_Array, sizeof(float), cudaMemcpyDeviceToHost);

printf("The Average Of Numbers In The Array: : %f\n", result);

// Copy Array To GPU For std deviation Function

cudaMemcpy(d\_Array, h\_Array, ArrayMemory, cudaMemcpyHostToDevice);

stddev << <ArraySize,1 >> > (d\_Array, ArraySize, result);

cudaMemcpy(&stdev, d\_Array, sizeof(float), cudaMemcpyDeviceToHost);

printf("The Standard deviation Of Numbers In The Array: : %f\n", sqrt(stdev));

end = std::chrono::system\_clock::now();

std::chrono::duration<double> elapsed\_seconds = end - start;

//std::time\_t end\_time = std::chrono::system\_clock::to\_time\_t(end);

std::cout<< "elapsed time: " << elapsed\_seconds.count() << "s\n";

return 0;

}