Practical-04

Vector Addition using CUDA

```
#include <algorithm>
#include <cassert>
#include <iostream>
#include <vector>
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

__global__ void vectorAdd(const int* __restrict a, const int* __restrict b,
    int* __restrict c, int N) {

    int tid = (blockIdx.x * blockDim.x) + threadIdx.x;


    if (tid < N) c[tid] = a[tid] + b[tid];
}

void verify_result(std::vector<int>& a, std::vector<int>& b,
    std::vector<int>& c) {
    for (int i = 0; i < a.size(); i++) {
        assert(c[i] == a[i] + b[i]);
    }
}

int main() {
    constexpr int N = 1 << 16;
    constexpr size_t bytes = sizeof(int) * N;
    std::vector<int> a;
    a.reserve(N);
    std::vector<int> b;
    b.reserve(N);
    std::vector<int> c;
    c.reserve(N);

    for (int i = 0; i < N; i++) {
        a.push_back(rand() % 100);
        b.push_back(rand() % 100);
    }

    int* d_a, * d_b, * d_c;
    cudaMalloc(&d_a, bytes);
    cudaMalloc(&d_b, bytes);
    cudaMalloc(&d_c, bytes);

    cudaMemcpy(d_a, a.data(), bytes, cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b.data(), bytes, cudaMemcpyHostToDevice);

    int NUM_THREADS = 1 << 10;

    int NUM_BLOCKS = (N + NUM_THREADS - 1) / NUM_THREADS;
    vectorAdd << < NUM_BLOCKS, NUM_THREADS >  >>(d_a, d_b, d_c, N);

    cudaMemcpy(c.data(), d_c, bytes, cudaMemcpyDeviceToHost);

    verify_result(a, b, c);

    cudaFree(d_a);
```

```
    cudaFree(d_b);
    cudaFree(d_c);

    std::cout << "COMPLETED SUCCESSFULLY\n";

    return 0;
}
```

################################ Output ##

0 3 6 9 12 15 18 21 24 27

C:\Users\DELL\source\repos\CudaRuntime1\x64\Debug\CudaRuntime1.exe (process 16848) exited with code 0.

To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.

Press any key to close this window . . .