МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Факультет «Информационные технологии и прикладная математика»

Кафедра «Вычислительная математика и программирование»

**Лабораторная работа №1**

**по курсу «Программирование графических процессоров»**

**Освоение программного обеспечения для работы с технологией CUDA.**

Выполнил: И.И. Тишин

Группа: 8О-406Б

Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

Москва, 2019

**Условие**

Ознакомление и установка программного обеспечения для работы с программно-аппаратной архитектурой параллельных вычислений(CUDA). Реализация одной из примитивных операций над векторами.

Вариант 6. Поэлементное возведение в квадрат вектора.

Входные данные. На первой строке задано число n -- размер векторов. На

следующей строке записано n вещественных чисел -- элементы вектора.

Выходные данные. Необходимо вывести n чисел -- результат поэлементного

возведения в квадрат исходного вектора.

**Программное и аппаратное обеспечение**

Compute capability : 3.0

Name : GeForce GTX 760

Total Global Memory : 2147155968

Shared memory per block : 49152

Registers per block : 65536

Warp size : 32

Max threads per block : (1024, 1024, 64)

Max block : (65535, 65535, 65535)

Total constant memory : 65536

Multiprocessors count : 6

**Метод решения**

Cчитываем вектор на CPU. На GPU поэлементно возводим в квадрат. Каждый поток возводит в квадрат элемент с заданным индексом. После вычислений, копируем вектор с GPU памяти на CPU и выводим результат.

**Описание программы**

Всего 3 функции: main, sub и subKernel.  
В main считываются данные со стандартного ввода и выводится результат на стандартный вывод.

В sub выделяется память под GPU массивы и копируются в них данные.

В subKernel производится вычисление результата.

\_\_global\_\_ void subKernel(double\* a, double\* b, int n) {

int idx = threadIdx.x + blockIdx.x \* blockDim.x; // Индекс нити

int offset = gridDim.x \* blockDim.x; // кол-во блоков \* размер блока

while(idx < n) {

b[idx] = a[idx] \* a[idx];

idx += offset;

}

}

**Результаты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество операций | threads = 1;  blocks = 1; | threads = 16;  blocks = 16; | threads = 256;  blocks = 256; | СPU |
| 100000000 | 19733.382812 | 135.427231 | 16.832129 | 6511.93 |
| 1000000 | 234.566788 | 1.357312 | 0.176064 | 74.58 |
| 10000 | 2.168928 | 0.029408 | 0.021696 | 0.87 |

**Выводы**

Алгоритм примитивен, но позволяет в разы сократить время на выполнение операции (в суммарное количество потоков раз). Проблем при реализации не возникло. Однако можно было бы уменьшить количество затрачиваемой памяти, сохраняя результат в исходный массив.