МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №*1***

**по курсу «Программирование графических процессоров»**

***Освоение программного обеспечения для работы с технологией CUDA. Примитивные операции над векторами.***

Выполнил: *А.Е. Аксенов*

Группа: *8О-408Б*

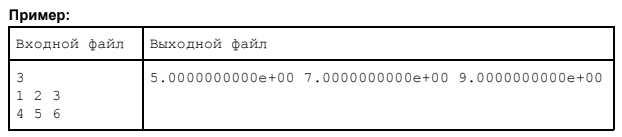
Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

Москва, 2023

**Условие**

Цель работы. Ознакомление и установка программного обеспечения для работы с программно-аппаратной архитектурой параллельных вычислений (CUDA). Реализация одной из примитивных операций над векторами. В качестве вещественного типа данных необходимо использовать тип данных double. Все результаты выводить с относительной точностью . Ограничение:

Входные данные. На первой строке задано число n - размер векторов. В следующих 2-х строках записано по n вещественных чисел - элементы векторов. Выходные данные. Необходимо вывести n чисел – результат сложения исходных векторов.

**Программное и аппаратное обеспечение**

|  |  |
| --- | --- |
| Графический процессор | |
| GPU | NVIDIA RTX 2070 |
| Compute Capability | 7.5 |
| Объем видеопамяти | 8192 МБ |
| Тип видеопамяти | GDDR6 |
| Разделяемая память | 48 КБ |
| Константная память | 64 КБ |
| Число регистров в блоке | 65636 |
| Максимальное число блоков | 16 |
| Максимальное число нитей | 1024 |
| Количество мультипроцессоров | 34 |
| Центральный процессор | |
| СPU | Intel Core i5 9600K |
| Число физических ядер | 6 |
| Максимальное число потоков | 6 |
| Базовая частота | 3.6 GHz |
| Частота Turbo Boost | 4.5 GHz |
| Объем кэша L3 | 9 МБ |
| Оперативная память | |
| Тип оперативной памяти: | DDR4 |
| Частота памяти | 2666MHz |
| Объём | 32 ГБ |
| Постоянная память | |
| Тип постоянной памяти | SSD |
| Объём постоянной памяти | 1024ГБ |
| Программное обеспечение | |
| Операционная система | Microsoft Windows 11 |
| Версия операционной системы | 2004 |
| Среда разработки | Microsoft Visual Studio 2022 |
| Версия среды разработки | 16.7.3 |
| NVIDIA CUDA Toolkit | 12.2 |

**Метод решения**

1. Считываем размер входных векторов
2. Выделяем память в основной памяти компьютера
3. Считываем входные данные.
4. Выделяем память на графическом ускорителе.
5. Копируем исходные данные на устройство.
6. Запускаем ядро с оптимальными параметрами, таким образом выполняется сложение входных векторов (зависит от возможностей графического ускорителя и конфигурации ядра).
7. Копируем результаты обратно в основную память компьютера.
8. Выводим результаты на экран.

**Описание программы**

Программа реализована в виде одного файла с исходным кодом. В ней имеется реализация одного единственного ядра, которая сопровождается квалификатором \_\_global\_\_. Используются базовые типы, определяемые NVIDIA CUDA API. В частности, для того, чтобы описать архитектуру сетки и блоков, используется тип dim3, который представляет из себя кортеж из трех целочисленных чисел, каждое из которых описывает размеры сетки или блока в одном из трех измерений. Обрабатываемые числа хранятся в вещественном типе двойной точности. Для хранения используются структуры данных из C++ STL. Реализация ядра предполагает, что конфигурация сетки и блоков выполнена в одномерном формате. Далее, каждый поток определяет свой порядковый номер относительно всей сетки и выполняет вычисление советующих элементов, с шагом равным общему числу потоков. Ниже представлена реализация ядра:

\_\_global\_\_ void sum\_kernel(double\* a, double\* b, int n) {          // ядро складывающее два вектора на GPU

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

                    i < n; i += blockDim.x \* gridDim.x){

        a[i] += b[i];

    }

}

**Результаты**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объем данных | Конфигурация — время (мкс) | | | | |
| 1,32 | 32,32 | 64,64 | 128,128 | 1024,1024 |
|  | 68 | 69 | 42 | 46 | 63 |
|  | 75 | 56 | 55 | 54 | 77 |
|  | 1103 | 418 | 396 | 783 | 427 |
|  | 40956 | 3674 | 3037 | 2877 | 2850 |
|  | 704055 | 24316 | 8584 | 5405 | 4222 |

|  |  |  |
| --- | --- | --- |
| Объём данных | Время на GPU (мкс) | Время на CPU (мкс) |
|  | 116 | 9 |
|  | 1435 | 2274 |
|  | 4261 | 570658 |

**Выводы**

Данный алгоритм может найти применение при решении различного рода математических задач, особенно тех, которые требуют обработки векторов больших размерностей. Очень простой и распространённый пример, в котором требуется возводить в квадрат компоненты векторов - нахождение евклидового расстояния для вектора. Результаты тестирования показывают значительную временную разницу между параллельными и последовательными вычислениями. Так или иначе, для довольно маленьких объёмов обрабатываемых данных выгоднее произвести обработку на центральном процессоре, т. к. копирование данных в графический процессор, планирование вычислений и их запуск требует некоторого времени. Важно сконфигурировать оптимальную конфигурацию ядра и учитывать специфику работы графической памяти, желательно повышать локальность обращений.