

**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.

Студент: А. А. Садаков
Группа: 8О-406Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2022

Условие

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

Вариант: 5. Поэлементное нахождение максимума векторов.

Программное и аппаратное обеспечение

ГРАФИЧЕСКИЙ ПРОЦЕССОР

Имя устройства: NVIDIA GeForce GTX 1650

Compute capability: 7.5

Размер графической памяти: 4242604032

Размер разделяемой памяти: 49152

Размер константной памяти: 65536

Максимальное количество регистров на блок: 65536

Максимальное количество потоков на блок: 1024

Количество мультипроцессоров: 14

ПРОЦЕССОР

Имя устройства: Intel Core I5-10300H

Архитектура: Comet Lake-H

Количество ядер (потоков): 4(8)

Базовая (максимальная) частота: 2,5(4,5)ГГц

Кеш 1-го уровня: 64Кб (на ядро)

Кеш 2-го уровня: 256Кб (на ядро)

Кеш 3-го уровня: 6Мб (всего)

ОПЕРАТИВНАЯ ПАМЯТЬ

Объём: 8Гб

Тип: DDR4

Частота: 2933МГц

ЖЁСТКИЙ ДИСК

Имя устройства: Intel SSDPEKNW512G8L

Тип диска: SSD

Объём памяти: 512Гб

Скорость чтения/записи: 1500/1000 Мб/с

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

OS: Windows 10, Подсистема Ubuntu 20.04

IDE: Visual Studio Code

Компилятор: nvcc

Метод решения

Для нахождения поэлементного максимума векторов нужно в цикле находить максимальное значение среди всех векторов для каждого индекса и записывать его в результирующий вектор.

Описание программы

Для того, чтобы поэлементно найти максимумы двух векторов, необходимо создать вектор *arrays* длины $2n$ на *device* и заполнить его элементами первого и второго вектора. Для этого я создал 2 вектора на *host* и скопировал их на *device* при помощи функции *cudaMemcpy* и передал в *kernel*. В функции *kernel* идёт сравнение элементов i и $i + n$, после чего максимум записывается на место i . После работы *kernel* я скопировал результат в выходной вектор с помощью аналогичной функции и вывел результат на экран.

Вызов *kernel* происходит с количеством нитей на блок — 1024 и количеством блоков — 1024. В самом *kernel* я вычисляю общий индекс исполняемой нити *idx* который и будет индексом в массиве при условии $idx < n$, где n — размер векторов, и смещение *offset*, которое позволяет найти следующий обрабатываемый элемент для нити.

Аргумент *count_of_arrays* нужен для указания количества векторов для нахождения максимума, так как в моей реализации поэлементный максимум может находиться для произвольного количества векторов (для задания этот аргумент всегда будет равен двум).

Код функции:

```
1 |__global__ void kernel (double *arrays, int count_of_arrays, int n) {
2 |    int idx = blockIdx.x * blockDim.x + threadIdx.x;
3 |    int offset = blockDim.x * gridDim.x;
4 |    double tmp;
5 |    while (idx < n) {
6 |        tmp = arrays[idx];
```

```

7 |         for (int i = 1; i < count_of_arrays; ++i) {
8 |             tmp = max(tmp, arrays[idx + i * n]);
9 |         }
10 |     arrays[idx] = tmp;
11 |     idx += offset;
12 | }
13 | }

```

Результаты

1. Замеры времени работы ядер с различными конфигурациями (время указано в микросекундах).

Размер входных Конфи- гурации ядра \ данных	100	10.000	1.000.000	30.000.000	100.000.000
1, 32	6	98	14981	377551	1171093
32, 32	4	100	485	14614	48258
32, 256	6	6	141	4129	13832
256, 256	5	5	139	4031	13636
1024, 1024	22	22	139	4031	13603

2. Сравнение с CPU

Размер входных данных	100	10.000	1.000.000	30.000.000	100.000.000
GPU(256, 256)	5	5	139	4031	13636
CPU	1	59	5798	173031	607674

Выводы

При выполнении данной лабораторной работы трудностей у меня не возникло, так как это вводное задание в курс программирования графических процессоров и из материалов с лекций можно было взять пример. Использование GPU оправдано для обработки большого объёма информации, так как для маленьких объёмов использование CPU может быть даже быстрее.