МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии 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: Т.А. Бердикин
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

1. Ознакомление и установка программного обеспечения для работы с программно-аппаратной архитектурой параллельных вычислений(CUDA).
   Реализация одной из примитивных операций над векторами.
2. Вариант 4: Поэлементное нахождение максимума векторов..

Программное и аппаратное обеспечение

Device: NVIDIA GeForce GTX 1050 Mobile
Размер глобальной памяти: 12143076 KB
Размер константной памяти : 65536
Размер разделяемой памяти: 49152
Регистров на блок: 32768
Максимум потоков на блок: 1024
Количество мультипроцессоров : 8
OS: Linux Ubuntu 20.04.2 LTS
Редактор: VSCode

Метод решения

Чтобы поэлементно найти минимум двух векторов, мы вызываем количество нитей, равное размеру массивов, а в качестве результата записываем максимум из двух соответствующих массивов в первый массив. Изначально был создан третий, но для экономии памяти решение лучше сделать ненормированным (то есть с удалением исходных данных).

Описание программы

Файл в первой лабораторной работе получился всего один, где и содержится весь код.
На вход принимается число n – размер векторов, и сами векторы. Затем вызывается kernel и уже там мы вычисляем индекс исполняемой нити, который (конечно же, при условии, что idx < arr_size) и будет индексом в массиве. И только уже потом мы выполняем поэлементное нахождение максимума двух векторов, записывая результат в первый.

Результаты

Конфигурация ядра (1, 32)

Количество элементов	Время (мс)
10	95
100	104
1000	106
10000	109
100000	261
1000000	1954

Конфигурация ядра (32, 32)

Количество элементов	Время (мс)
10	107
100	105
1000	100
10000	110
100000	275
1000000	1886

Конфигурация ядра (64, 64)

Количество элементов	Время (мс)
10	97
100	102
1000	105
10000	101
100000	239
1000000	1780

Конфигурация ядра (128, 128)

Количество элементов	Время (мс)
10	116
100	95
1000	102
10000	115
100000	232

1000000	1863

Конфигурация ядра (256, 256)

Количество элементов	Время (мс)
10	97
100	97
1000	115
10000	102
100000	243
1000000	1793

Конфигурация ядра (512, 512)

Количество элементов	Время (мс)
10	103
100	96
1000	101
10000	101
100000	252
1000000	1693

Конфигурация ядра (1024, 1024)

Количество элементов	Время (мс)
10	126
100	89
1000	86
10000	100
100000	240
1000000	1693

CPU

Количество элементов	Время (мс)
10	186
100	1070
1000	2188
10000	2338

100000	11165
1000000	26644

Выводы

В данной лабораторной работе я познакомился с технологией CUDA, что является основами параллельного программирования на видеокартах Nvidia.
Алгоритм сам по себе довольно простой, и особых сложностей при его написании не возникло.
Из результатов можно сделать вывод о том, что при больших объёмах данных вычисления лучше проводить на видеокарте, при малых – особенной разницы нет.