**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**Отчет**

**по лабораторной работе №6**

**по дисциплине «Параллельные алгоритмы»**

**Тема: Оптимизация доступа к памяти в модели OpenCL**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0304 |  | Аристархов И.Е. |
| Преподаватель |  | Сергеева Е.И. |

Санкт-Петербург

2023

**Цель работы.**

Используя OpenCL реализовать параллельное умножение матриц.

**Выполнение работы.**

Общий принцип работы с OpenCL был описан в предыдущей работе, поэтому данная часть будет описана кратко с подробностями в различиях.

Сперва получается устройство (в данном случае GPU), создается на него контекст и очередь команд. Затем загружается исходный код для kernel, он собирается и с ним формируется Kernel. Далее идет формирование переменных. Именно здесь и лежит основное отличие с предыдущей работой, где использовалось 2D изображение, когда как тут требуется создать 3 двумерных массива. Это можно сделать, создав одномерные буферы с ручным контролем перебора индексов у них. Для первых двух буферов (для матриц перемножения) передается не только размер, но и начальное значение, когда последний (для результата) остается без последнего.

Теперь буферы и переменная размера привязываются к Kernel и создается задача на вычисление в очереди. Остается дождаться конца вычислений и очистить память.

Сама программа устройства не представляет особого интереса, т.к. все, что там проиходит – это перемножение элементов из обоих матриц по положению рабочего элемента.

**Анализ.**

Вычисления производились на Nvidia GTX 1650 4 Гб. Рабочая группа выставлялась в значение 32x32 (большие значения не поддерживаются).

Были получены следующие результаты:

|  |  |  |
| --- | --- | --- |
| Размер матрицы | Время на CPU, сек | Время на GPU, сек |
| 128 | 0,064 | 0,004 |
| 256 | 0,486 | 0,015 |
| 512 | 1,902 | 0,062 |
| 1024 | 14,267 | 0,258 |
| 2048 | 119,558 | 0,953 |

CPU реализация брала количество потоков по размеру матрицы (в одной из предыдущих работ был сделан вывод, что это лучший подход по скорости выполнения).

Из таблицы прекрасно видно, что GPU реализация выигрывает в десятки и в сотни раз, относительно CPU реализации.

**Выводы.**

В результате выполнения лабораторной работы была разработана программа производящая умножение матриц, с использованием GPU на OpenCL. Сравнение данной программы с ранней реализацией на CPU показало, что GPU во много раз быстрее производит умножение.