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 _____ Асташёнок М.С.

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.

Далее перейдем к программе. В ней было решено использовать локальную память, которая общая в пределах рабочей группы. Это позволит переиспользовать память и уменьшить количество запросов в глобальную память.

Каждый рабочий элемент в группе ответственен за загрузку своего участка локальной памяти (которая имеет форму квадрата). Далее производится операция перемножения в пределах этого блока, после чего загружается следующий блок.

Остается добавить, что производимые вычисления в данной программе сравнивались с правильными результатами, рассчитанными с помощью прошлой работы.

Анализ.

Вычисления производились на видеокарте с 8 Гб видеопамяти и 2176 универсальными ядрами, а также на процессоре с 6 ядрами и 12 потоками. Рабочая группа выставлялась в значение 32x32 (большие значения не поддерживаются).

В качестве результата времени берется среднее арифметическое 10 испытаний. В ходе запусков программы с разными были получены следующие результаты:

Размер матрицы	Размеры рабочей группы	Время на CPU, мкс		Время на GPU, мкс
		Стандартный алгоритм, разбитый на 7 потоков	Алгоритм Штрассена	Tiling-алгоритм
128	8x8	4597	5686	993
	16x16			1074
	32x32			1084
256	8x8	34581	42296	3968
	16x16			3890
	32x32			3965
512	8x8	237096	260166	-
	16x16			16603
	32x32			17856
1024	8x8	1811846	1691837	-
	16x16			-
	32x32			78401
2048	8x8	14917610	11758702	-
	16x16			-
	32x32			400170

Из таблицы прекрасно видно, что GPU реализация выигрывает в десятки 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.