МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Отчет
по лабораторной работе №5
по дисциплине «Параллельные алгоритмы»
Тема: Знакомство с программированием гетерогенных систем в стандарте OpenCL

Студент гр. 0304 _____ Асташёнок М.С.

Преподаватель _____ Сергеева Е.И.

Санкт-Петербург
2023

Цель работы.

На практике освоить методы работы с OpenCL.

Поставленные задачи.

Реализовать расчёт фрактала Мандельброта на OpenCL. Визуализировать результат.

Выполнение работы.

Выполнение кода на OpenCL можно разбить на два этапа:

- Настройка окружения
- Работа с индивидуальным рабочим элементом

Начнем с первого. Сперва нужно получить устройство выполнения. Для задачи создания фрактала будет достаточно использовать 1 устройство – в данном случае дискретную видеокарту.

Далее требуется создать OpenCL контекст. Он задает сколько устройств будет работать, какие это будут устройства и какие у них будут настройки. Контекст в этой задаче подразумевает использование одного устройства без дополнительных настроек.

Также нужно создать очередь исполнения. Она будет определять в каком порядке должны исполняться задачи конечным устройством (на одно устройство создается только одна очередь).

Теперь нужно создать kernel. Для него требуется получить программу .cl и скомпилировать ее. Скомпилированная программа вместе с определенным именем создает kernel. Теперь к нему можно привязать переменные.

В случае используемой программы (о ней позже) будет нужно создать переменную изображения. Для этого в OpenCL есть специальная команда clCreateImage, которой на вход подается контекст и различные конфигурации (в них различные метаданные изображения, в том числе его размер в пискелях). Одной из конфигураций здесь является флаг CL_MEM_WRITE_ONLY, который ограничивает изображение только на запись в рабочих элементах. Далее изображение нужно привязать к ранее созданному kernel с помощью команды clSetKernelArg.

Наконец, остается добавить kernel в очередь на исполнение. Но и тут есть нюанс. Т.к. мы хотим распараллелить вычисление фрактала, нам нужно

задать индексное пространство и рабочие группы. Индексное пространство – это массив (1, 2 или 3 мерный), размер которого определяет общее количество рабочий элементов. Рабочая группа – это объединение нескольких рабочих элементов в той же размерности.

Идея в том, что рабочие группы играют роль задач в thread pool, которые будут ожидать своей очереди в первом освободившемся Compute Unit. А все рабочие элементы внутри одной группы будут выполняться в concurrent режиме (не обязательно parallel).

Хорошо, программа выполнилась, но еще надо получить как-то созданную картинку. Для этого создаем еще одну задачу в очередь с помощью clEnqueueReadImage. Туда передаем очередь, переменную изображения и наш массив, в который надо записать результат.

Все, очередь успешно сформирована. Но на самом деле еще ничего не происходит, т.к. требуется сделать clFlush для начала расчетов. clFinish можно использовать, чтобы дождаться, когда все задачи из очереди завершатся.

Вот теперь точно все успешно вычислилось (если не было никаких ошибок) и результирующий массив можно передать библиотеке для формирования png-файла.

Остается рассмотреть саму программу рабочего элемента. По сути это функция, которая на вход получает `__write_only image2d_t out` переменную, доступную только на запись. С помощью команд get_global_id(0) и get_global_id(1) можно получить индекс текущего элемента в глобальном индексном пространстве. В данном случае – это координаты в пикселях, которые еще надо преобразовать в относительные координаты.

Для преобразования нужно знать общие размеры изображения. Их можно получить через get_image_dim(out).

Далее идет расчет цвета пикселя. Подробности расчета здесь на важны, важно лишь то, что проводятся вычисления в теоретически бесконечном цикле. Если в какой-то момент до достижения предельного значения итераций выполняются достаточные условия для завершения расчетов, то мы выбираем цвет такого пикселя по количеству пройденных итераций. Иначе цвет черный.

Выбранный цвет теперь записывается в изображение с помощью команды write_imageui, в которой указываются положение пикселя и его цвет.

Результат вычисления:

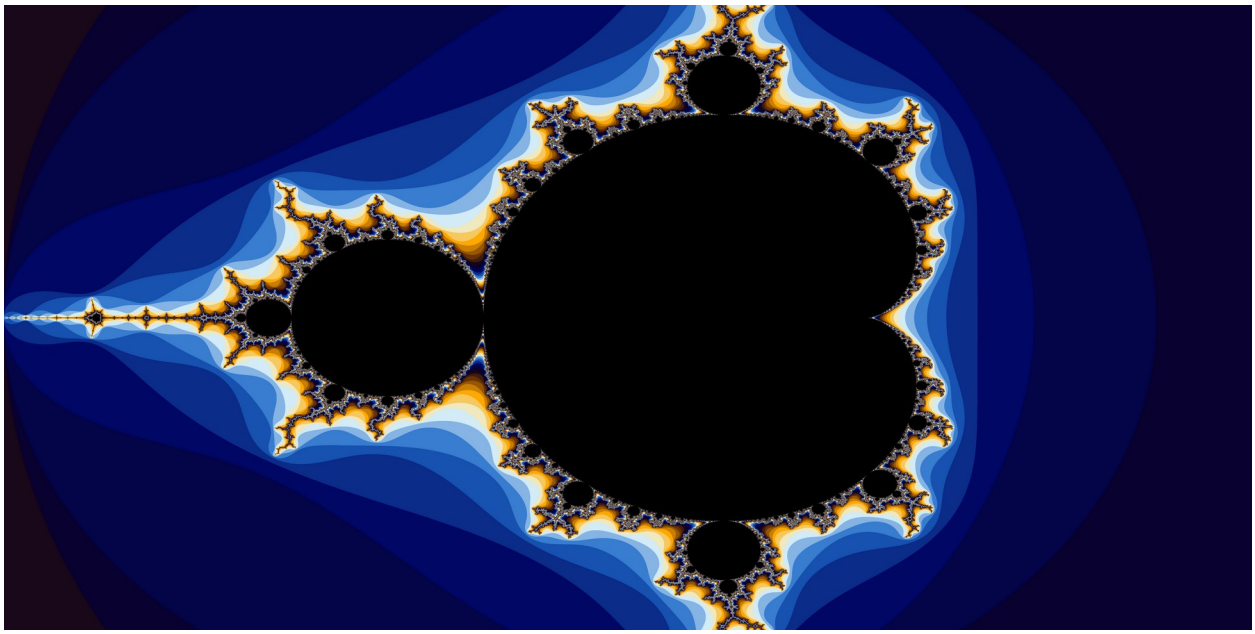

Рисунок 1 — Вычисленный с помощью OpenCL фрактал Мендельброта

Оценка производительности.

Множественные замеры показали, что при увеличении разрешения изображения время выполнения стремительно увеличивается. Можно выделить 3 источника увеличения:

- Часть подготовки OpenCL, различных данных и массивов (все задачи до вызова clEnqueueNDRangeKernel)
- Часть исполнения (clEnqueueNDRangeKernel + clFlush + clFinish)
- Часть создания png-файла (остальное)

Наибольший прирост во времени работы приходится на последнюю часть, но ее можно отбросить, т.к. основная задача – увидеть время вычислений.

Это будет часть исполнения. В ней берется три команды, где первая ставит вычисление в очередь, вторая делает flush отправляя задачу на вычисление и третяя, которая ожидает окончания исполнения. На самом деле между первой и второй задачей еще есть команда на получение результатов, но она не несет какой-либо вычислительной ценности.

Замеры части исполнения показали следующие результаты:

Таблица 1 – Замеры времени работы части исполнения в зависимости от размера изображения и размера рабочей группы.

Размеры изображения	Размер рабочей группы	Время исполнения, микросекунд
1024x1024	1x1	8563
1024x1024	8x8	1322
1024x1024	16x16	1371
1024x1024	32x32	1487
2048x2048	1x1	31115
2048x2048	8x8	4543
2048x2048	16x16	4571
2048x2048	32x32	4756
4096x4096	1x1	120988
4096x4096	8x8	37904
4096x4096	16x16	38412
4096x4096	32x32	39086

Из таблицы 1 видно, что чем больше размер изображения, тем больше времени требуется на вычисления. В то же время если для достаточно большого разрешения уменьшить размер рабочей группы (тем самым увеличив количество этих самых групп), то время исполнения также будет расти вследствие переполнения пула исполнения.

Однако слишком малый размер work-группы может привести к разительному падению производительности

Таблица 2 – Замеры времени работы части исполнения в зависимости от формы рабочей группы.

Размер изображения	Размер рабочей группы	Время исполнения
8192x8192	32x32	156634
	1x1024	157724
	1024x1	158921

Как видно из таблицы выше, квадратная форма рабочей группы имеет большую скорость исполнения, хоть изменение скорость и в пределах погрешности.

Выводы.

В данной работы были разобраны базовые принципы работы с OpenCL на примере разработки программы по вычислению фрактала Мандельброта. Было изучено, как работать с адресными пространствами, рабочими группами и элементами. Был изучен C-like язык разработки программ для рабочих элементов.