**Лаб 2.**

1. Какая версия compute capability у архитектуры NVIDIA Fermi?

**Ответ: 2.1**.

1. Какая максимальная размерность блока у GPU с compute capability 3.0?

**Ответ: 1024 по X, 64 по Y и Z.**

1. Предположим, вы используете одномерную сетку и блок. Если максимальная размерность сетки на устройстве 65535, а максимальная размерность блока 512, какое максимальное количество нитей может быть запущено на GPU?

**Ответ: 65535 \* 512 = 33553920.**

1. При каких условиях программист предпочтет не запускать максимальное количество нитей?

**Ответ: при недостаточном количестве данных для эффективного использования этих нитей.**

1. Что может помешать программе запустить максимальное количество нитей?

**Ответ: разделяемая память или регистры, используемые всеми потоками**

1. Что такое разделяемая память?

**Ответ: быстрая память с возможностью чтения и записи, доступная для всех нитей в блоке.**

1. Что такое глобальная память?

**Ответ: память с возможностью чтения и записи, доступная для абсолютно всех потоков.**

1. Что такое константная память?

**Ответ: быстрая память для чтения, которая не меняется во время выполнения ядра.**

1. Что характеризует размер варпа в GPU?

**Ответ: количество потоков, которые работают синхронно.**

1. Поддерживаются ли числа двойной точности в GPU версии 1.3?

**Ответ: поддержка чисел двойной точности началась с версии 1.3.**

**Лаб 3.1**

1. Сколько операций с плавающей точкой выполняется в вашем ядре? ОБЪЯСНИТЬ.

**ОТВЕТ: 3 произведения и 2 сложения.**

1. Какой формат будет более эффективным для преобразования цвета: 2D матрица, где каждый элемент представляет собой значение RGB, или 3D матрица, где каждый срез по оси Z обозначает цвет? То есть будет ли лучше чередование цветов? Вы можете назвать приложение, где противоположный вариант будет лучшим?

**ОТВЕТ: в данном случае представление с чередованием цветов лучше, так как за раз можно будет полностью считать информацию о пикселе.**

1. Сколько чтений из глобальной памяти производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: одно чтение глобальной памяти на пиксель (так как за раз будет считываться по три значения).**

1. Сколько записей в глобальную память производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: одна запись в глобальную память на пиксель. Мы пишем одно число с плавающей точкой, так как выходное изображение состоит из одиночных значений.**

1. Опишите, какие оптимизации можно сделать в вашем ядре, чтобы достичь увеличения производительности.

**ОТВЕТ: Можно производить операции в текстурной памяти.**

1. Назовите три применения преобразования цветов.

**ОТВЕТ: преобразование цветов используется во многих алгоритмах. Например: гистограммы, выделение границ, наложение различных эффектов на изображение.**

**Лаб 3.2**

1. Сколько операций с плавающей точкой выполняется в вашем ядре? ОБЪЯСНИТЬ.

**ОТВЕТ: в правильном решении каждое ядро в среднем производит одно сложение с плавающей точкой и одно деление для каждого пикселя.**

1. Сколько чтений из глобальной памяти производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: одно чтение глобальной памяти на пиксель. Так как изображение монохромное это соответствует одному чтению числа с плавающей точкой.**

1. Сколько записей в глобальную память производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: одна запись в глобальную память на пиксель. Так как изображение монохромное это соответствует одной записи числа с плавающей точкой.**

1. Опишите, какие оптимизации можно сделать в вашем ядре, чтобы достичь увеличения производительности.

**ОТВЕТ: можно представить изображение с помощью беззнаковых символов (байтов). Тогда каждый пиксель будет занимать 1 байт против 4 в числах с плавающей точкой.**

**Лаб 3.3**

1. Сколько операций с плавающей точкой выполняется в вашем ядре сложения векторов? ОБЪЯСНИТЬ.

**ОТВЕТ: N – по одному на каждую пару элементов входных векторов.**

1. Сколько чтений из глобальной памяти производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: 2N – по одному на каждый элемент каждого входного вектора.**

1. Сколько записей в глобальную память производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: N - по одному на каждый элемент результирующего вектора.**

1. Каким образом функциональный стиль кода сложения векторов в Thrust делает разработку проще или сложнее?

**ОТВЕТ: объявление данных на устройстве в качестве векторов уменьшает количество кода на хосте. Данный код, запускающий сложение векторов, является более функциональным, чем императивный код ядра CUDA.**

**Лаб 3.4**

1. Сколько операций с плавающей точкой выполняется в вашем ядре сложения векторов? ОБЪЯСНИТЬ.

**ОТВЕТ: N – по одному на каждую пару элементов входных векторов.**

1. Сколько чтений из глобальной памяти производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: 2N – по одному на каждый элемент каждого входного вектора.**

1. Сколько записей в глобальную память производится вашим ядром? ОБЪЯСНИТЬ.

**ОТВЕТ: N - по одному на каждый элемент результирующего вектора.**

1. Опишите возможный вариант оптимизации ядра для увеличения производительности.

**ОТВЕТ: разбить на несколько ядер, чтобы перекрыть передачу данных и исполнение ядра.**

1. Назовите три приложения сложения векторов.

**ОТВЕТ: анализ больших данных, статистика, быстрое преобразование Фурье.**

**4.1**

1. Сколько операций с плавающей точкой будет произведено вашим ядром? Объясните.

**ОТВЕТ: одно скалярное произведение на один элемент выходной матрицы:**

1. Сколько чтений из глобальной памяти будет произведено вашим ядром? Объясните.

**ОТВЕТ: каждая нить произведет чтений, а задействованных нитей всего .**

1. Сколько записей в глобальную память будет произведено вашим ядром? Объясните.

**ОТВЕТ: по количеству выходных элементов матрицы:**

1. Опишите, какие дальнейшие оптимизации помогли бы вам улучшить производительность ядра.

**ОТВЕТ: настройка параметров запуска ядра для оптимизации размещения в памяти.**

1. Сравните сложности реализации этой и предыдущей лабораторной. С какими сложностями вы столкнулись в данной лабораторной?

**ОТВЕТ: Ошибки можно сделать при переводе из глобального адресного пространства в адресное пространство разделяемой памяти.**

1. Представим: имеются матрицы, размерность которых превышает максимальную размерность нитей. Набросайте алгоритм, который выполняет перемножение этих матриц.

**ОТВЕТ: каждая нить должна обрабатывать 4, 9 или 16 соседних элементов матрицы.**

1. Представим: имеются матрицы, которые не умещаются в глобальной памяти. Набросайте алгоритм, который выполняет произведение таких матриц.

**ОТВЕТ: по аналогии с блочным перемножением с использованием разделяемой памяти нужно произвести разбиение на блоки, но на стороне хоста.**

**Лаб 4.2**

1. Сколько операций с плавающей точкой будет произведено вашим ядром? Объясните.

**ОТВЕТ: numCRows \* numCCols скалярных произведений = 2 \* numCRows \* numCCols \* numACols.**

1. Сколько чтений из глобальной памяти будет произведено вашим ядром? Объясните.

**ОТВЕТ: numCRows \* numCCols скалярных произведений = 2 \* numCRows \* numCCols \* numACols.**

1. Сколько записей в глобальную память будет произведено вашим ядром? Объясните.

**ОТВЕТ: по количеству выходных элементов матрицы. numCRows \* numCCols.**

1. Опишите возможный вариант оптимизации, которую можно произвести для того, чтобы увеличить производительность вашего ядра.

**ОТВЕТ: Разбить матрицу на блоки, помещая их в разделяемую память, чтобы снизить количество обращений к глобальной памяти.**

1. Назовите три применения произведения матриц.

**ОТВЕТ: произведение матриц используется в почти каждом ресурсоемком приложении. Например, нейронные сети, компьютерная графика, дифференциальные уравнения в частных производных.**

**Лаб 7.1**

1. Опишите все оптимизации, которые вы попробовали выполнить, вне зависимости от того, стали ли вы их использовать в дальнейшем или нет, и вне зависимости от того, как они повлияли на производительность.

**ОТВЕТ: самая существенная оптимизация в этом примере – выполнить приватизацию гистограммы в разделяемую память. Атомарные операции выполняются в разделяемой памяти гораздо быстрее, чем в глобальной.**

1. С какими сложностями вы столкнулись во время выполнения оптимизации?

**ОТВЕТ: использование разделяемой памяти требует синхронизации.**

1. Какая оптимизация дала наибольший прирост производительности?

**Ответ: использование приватизации в разделяемой памяти дало значительный прирост производительности.**

1. Сколько чтений из глобальной памяти выполнит ваше ядро вычисления гистограммы? Объясните.

**ОТВЕТ: одно чтение на один элемент входного массива.**

1. Сколько записей в глобальную память выполнит ваше ядро вычисления гистограммы? Объясните.

**ОТВЕТ: каждый блок нитей выполняет одно атомарное сложение в глобальную память на один интервал гистограммы.**

1. Сколько атомарных операций выполнит выше ядро вычисления гистограммы? Объясните.

**ОТВЕТ: одна атомарная операция в разделяемой памяти на один входной элемент после NUM\_BINS атомарных операций в глобальной памяти на один блок нитей.**

1. Сколько конкурентных ожидается, если все элементы входного массива будут содержать одинаковые значения?

**ОТВЕТ: если все значения входного массива будут содержать одинаковое значение, между собой будут конкурировать все нити в отдельном блоке.**

1. Сколько конкурентных ожидается, если все элементы входного массива будут содержать случайные значения?

**ОТВЕТ: ожидается небольшое количество конкурентных обращений, так как одновременно каждая из 512 нитей будет выполнять атомарную операцию над одним из 4096 интервалов.**

**Лаб 7.2**

1. Опишите все оптимизации, которые вы попробовали выполнить, вне зависимости от того, стали ли вы их использовать в дальнейшем или нет, и вне зависимости от того, как они повлияли на производительность.

**ОТВЕТ: самая существенная оптимизация в этом примере – выполнить приватизацию гистограммы в разделяемую память. Атомарные операции выполняются в разделяемой памяти гораздо быстрее, чем в глобальной.**

1. С какими сложностями вы столкнулись во время выполнения оптимизации?

**ОТВЕТ: использование разделяемой памяти требует синхронизации.**

1. Какая оптимизация дала наибольший прирост производительности?

**Ответ: использование приватизации в разделяемой памяти дало значительный прирост производительности.**

1. Сколько чтений из глобальной памяти выполнит выше ядро вычисления гистограммы? Объясните.

**ОТВЕТ: одно чтение на один элемент входного массива.**

1. Сколько записей в глобальную память выполнит выше ядро вычисления гистограммы? Объясните.

**ОТВЕТ: каждый блок нитей выполняет одно атомарное сложение в глобальную память на один интервал гистограммы.**

1. Сколько атомарных операций выполнит выше ядро вычисления гистограммы? Объясните.

**ОТВЕТ: одна атомарная операция в разделяемой памяти на один входной элемент после NUM\_BINS атомарных операций в глобальной памяти на один блок нитей.**

1. Большинство текстов содержит только буквы, цифры и пробелы. Эти 95 знаков в таблице ASCII имеют последовательные значения кода: 32 - 126. Что можно сказать о ситуации (порядок выполнения) с атомарным доступом, когда более 95 нитей одновременно пытаются инкрементировать одну гистограмму?

**ОТВЕТ: Одна атомарная операция на каждый входной элемент в разделяемой памяти, затем NUM\_BINS атомарных операций на каждый блок нитей, чтобы собрать результат в итоговую гистограмму.**

**Лаб 8.1**

1. Сколько чтений из глобальной памяти выполнит программа?

**ОТВЕТ: данный код выполнит чтений.**

1. Сколько чтений из разделяемой памяти выполнит ваша программа?

**ОТВЕТ: каждая нить сделает 7 чтений, всего нитей.**

1. Как бы вы использовали готовый код, выполняющий свертку, для расчета данной разностной схемы?

**ОТВЕТ: для данной схемы потребуется свертка с ядром 3x3, где центральный элемент равен 6, смежные к нему – 1. Угловые элементы равны 0.**

1. Ваш код, рассчитывающий разностную схему, делает столько же, меньше или больше обращений к памяти, чем эквивалентный код расчета свертки 3x3?

**ОТВЕТ: данный код выполнит меньшее количество обращений. Эквивалентный код, проводящий вычисления через свертку, также будет загружать значения, соответствующие нулевым угловым элементам маски.**

**Лаб 8.2**

1. Назовите 3 применения свертки.

**ОТВЕТ: обработка изображений, обработка сигналов, разностные схемы.**

1. Сколько операций с плавающей точкой будет выполнено вашим ядром свертки? Объясните.

**ОТВЕТ: для каждого выходного элемента требуется MASK\_WIDTH \* MASK\_WIDTH входных элементов, а всего выходных элементов *width\*height\*channels*.**

1. Сколько чтений из глобальной памяти произведет ваше ядро? Объясните.

**ОТВЕТ: каждый блок нитей считает элементов. Всего блоков нитей, что в сумме дает . Это включает фантомные элементы, которые не будут считаны. Нужно вычесть .**

1. Сколько записей в глобальную память произведет ваше ядро? Объясните.

**ОТВЕТ: одна запись на один выходной элемент, или *width\*height\*channels*.**

1. Какими будут максимальное, минимальное и среднее значения количества *реальных* операций, которые произведет одна нить? Под *реальными* подразумеваются операции, которые напрямую влияют на финальный результат.

**ОТВЕТ: все нити выполнят одну и ту же работу во время свертки, которая исчисляется KERNEL\_WIDTH \* KERNEL\_WIDTH операциями.**

1. Какой была бы скорость выполнения операций с плавающей точкой для ядер CPU и GPU в данном приложении? Как они зависят от размера входных данных?

**ОТВЕТ: Это зависит от конкретной машины. Производительность реализации на CPU будет падать с ростом размера входных данных, так как они станут слишком большими для кэша. Реализация на GPU будет иметь относительно постоянную скорость выполнения операций с плавающей точкой.**

1. Сколько дополнительного времени понадобится в качестве накладных расходов на вычисления на GPU? Учтите, весь код выполняется на хосте, за исключением самого ядра. Как будут изменяться эти накладные расходы при увеличении размера входных данных?

**ОТВЕТ: ответ зависит от конкретной машины. Количество данных, которые нужно переслать, и выполнение свертки растут с одинаковым темпом, поэтому для больших входных данных эти показатели тоже будут расти с одинаковым темпом**.

1. Как вы думаете, что случится, если вы увеличите размер маски (скажем, до 1024), в то время как размер блока будет 16x16? На что вы потратите больше всего времени? Наложит ли это новые ограничения, которые нужно будет учесть в алгоритме (подумайте о размере разделяемой/константной памяти)?

**ОТВЕТ: если маска свертки слишком большая, потребуется слишком много места для обеспечения параллелизма в блоке нитей. Для больших масок большую часть времени будет производиться чтение фиктивных граничных элементов из глобальной памяти в разделяемую, что нивелирует все преимущества алгоритма.**

1. Обязательно ли вам иметь отдельный буфер для результата? Или, с другой стороны, почему нельзя записывать результат свертки в то же самое место?

**ОТВЕТ: если вести запись результата в то же самое место, можно изменить значение, необходимое другой нити, что может привести к некорректному результату.**

1. Что такое тождественная маска?

**ОТВЕТ: все нули, кроме единицы в центре.**