**ВВЕДЕНИЕ**

Тест производительности, бенчмарк — контрольная задача, необходимая для определения сравнительных характеристик производительности компьютерной системы и её качества в целом. Бенчмарками также называются программы, которые тестируют время автономной работы ноутбуков и карманных персональных компьютеров, радиус действия беспроводной сети, пропускную способность каналов передачи данных, амплитудно-частотную характеристику звукового тракта и другие доступные для измерения характеристики, напрямую не связанные с производительностью. Многие компьютерные игры, благодаря их требовательности к аппаратному обеспечению и зависимости скорости работы игры от мощности компьютеров, успешно используются в качестве бенчмарков. Тестирование компьютера необходимо для быстрого выявления любых неполадок в работе системы как на программном, так и на аппаратном уровне. Если программа - тестировщик показывает результат, который отличается от нормального, это свидетельствует о том, что одно или несколько устройств работают неверно, либо же устройства не обладают техническими характеристиками, необходимыми для выдачи удовлетворяющего результата. Любой бенчмарк включает в себя два процесса: оценивание и сопоставление с образцом. Обычно за образцы принимают устройства с лучшей «начинкой». Бенчмарки необходимы конкурирующим компаниям для оценки качества выпускаемой продукции и для определения конкурентоспособности её на рынке. Но бенчмарки используются также обычными пользователями для выявления:

- соответствия их устройств характеристикам, требуемым различными приложениями, играми, программами;

- возможных неисправностей в работе системы.

**1 ПОСТАНОВКА ЗАДАЧИ**

Для реализации собственного бенчмарка следует проанализировать имеющиеся аналоги, выдвинуть требования к реализации и выбрать подходящие средства разработки. Требуется изучить программно-аппаратную архитектуру параллельных вычислений CUDA, которая позволяет существенно увеличить вычислительную производительность благодаря использованию графических процессоров фирмы Nvidia.   
CUDA позволяет реализовывать на специальном упрощённом диалекте языка программирования Си алгоритмы, выполнимые на графических процессорах Nvidia, и включать специальные функции в текст программы на Си. Архитектура CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью. Требуется изучить организацию межпроцессного взаимодействия в выбранной операционной системе, а также основы   
MMX - технологии.

На основании изученной информации необходимо предоставить системе задачи и определить время их выполнения. Определить количество данных в единицу времени, которое ядро ОС или процесс способно обработать. Затем оценить полученные при проведении теста результаты и сопоставить их с эталонными показателями.

В процессе реализации программного продукта следует минимизировать количество кода и обеспечивать удобство его понимания. В ходе работы следует пользоваться отладчиком, а по окончанию проектирования произвести тестирование программного продукта.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА СПЕЦИФИКАЦИЙ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Обзор литературы и существующих аналогов

Существует огромное количество различных видов бенчмарков. Все они различаются:

- по степени нагрузки, оказываемой на систему

- по объектам, над которыми производится тестирование

Рассмотрим наиболее популярные программы для тестирования производительности компьютеров.

PCMark - название серии компьютерных бенчмарков. Эта серия бенчмарков спроектирована для тестирования следующих компонентов персонального компьютера: центральный процессор, материнская плата, оперативная память, винчестер. Программы серии PCMark тестируют стабильность и производительность работы процессоров, скоростные характеристики и пропускную способность оперативной и постоянной памяти, а также множество других характеристик компьютерных компонентов. Для тестирования используются различные тесты, как синтетические, нагружающие определённые блоки компьютера, так и прикладные, например архивация данных, кодирование и декодирование аудио и видео, производительность физического движка и т. д.

3DMark - название серии популярнейших и широко известных компьютерных тестов производительности. Тесты данной серии ориентированы прежде всего на графические компоненты персонального компьютера с целью определения производительности системы в компьютерных играх. Основное предназначение 3DMark — тестирование производительности и стабильности графической платы (видеокарты) и оценка её производительности в условных единицах.

ATTO Disk Benchmark - средство для тестирования производительности любых дисков. Пользователь может, в течение нескольких секунд, узнать скорость записи или чтения на выбранном носителе. Для тестирования подойдут одинаково хорошо как RAID контроллеры, контроллеры памяти, HDD-диски, так и SSD-носители.

UNIGINE – бенчмарки предназначенные для тестирования графической подсистемы персонального компьютера с целью определения её производительности. Бенчмарки UNIGINE создают максимальную нагрузку на оборудование и обеспечивают полностью беспристрастные результаты на всех платформах: Windows, Linux и Mac OS.

2.2 Разработка спецификации функциональных требований

Необходимыми условиями при разработке данного бенчмарка под ОС Windows являются:

* наличие видеокарты Nvidia;
* простота интерфейса;
* определение разумной нагрузки на тестируемые устройства;
* возможность играть по сети;

**3 ОБОСНОВАНИЕ ТЕХНИЧЕСКИХ ПРИЕМОВ ПРОГРАММИРОВАНИЯ**

**3.1 Выбор языка**

Язык С — компилируемый статически типизированный язык программирования общего назначения. Он уникален тем, что стал первым языком высокого уровня, потеснившим ассемблер в разработке системного программного обеспечения. Язык C является относительно простым, но мощным, этим обусловлено его широкое применение. C является основой для многих других языков программирования, и, таким образом, опыт, полученный с помощью C, может быть применен и к этим языкам. Язык C - удобное средство разработки для системного программиста. Он позволяет глубоко вникать в самые тонкие механизмы обработки информации на ЭВМ. Хотя язык требует высокой дисциплины, он не строг в формальных претензиях и допускает краткие формулировки.

C – современный язык. Он включает в себя те управляющие конструкции, которые рекомендованы теорией и практикой программирования. Его структура побуждает программиста использовать в своей работе нисходящее проектирование, структурное программирование и пошаговую разработку модулей.

Популярность C в разработке системного программного обеспечения объясняется наличием у данного языка программирования следующих достоинств:

• высокая степень структурированности языка;

• универсальность;

• модульность;

• наличие средств программирования практически на уровне ассемблера;

• большой набор операторов;

• предоставление возможностей для управления памятью.

Таким образом, выбор языка C для реализации данного проекта является оптимальным.

**3.2 Выбор среды разработки для реализации**

Для реализации поставленной задачи мной была выбрана среда разработки Microsoft Visual Studio 2013 , так как она наиболее подходит для создания данного проекта, а именно автоматизированной информационной системы, имеет понятный интерфейс и большой набор функций, инструментов.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Microsoft Silverlight.

Важную роль в выборе среды разработки сыграла возможность интеграции Nvidia CUDA Toolkit, GPU Computing SDK и Parallel Nsight в Visual Studio.

NVIDIA® CUDA® Toolkit – набор инструментальных средств для разработки GPU-приложений на C/C++. Эти инструменты включают: CUDA-компилятор, библиотеку математических функций, а также набор утилит для отладки и профилирования приложений. Помимо этого, в поставку входит подробное описание программно-аппаратной модели, руководство пользователя и документация по CUDA API.

NVIDIA® GPU Computing SDK содержит множество примеров использования CUDA, которые сопровождаются подробным описанием.

NVIDIA® Parallel Nsight – мощное расширение для Visual Studio, позволяющее осуществлять отладку, профилирование и анализ CUDA-приложений и не только.

Также плюсом в сторону выбора Visual Studio была возможность вставки кода на языке Ассемблера. С помощью этих вставок была произведена работа с дополнительным SIMD набором инструкций (MMX).

**4 РАЗРАБОТКА АЛГОРИТМОВ**

**4.1 Алгоритм тестирования графического устройства умножением элементов матриц (1024x1024)**

GPU это вычислительное устройство, которое:

- Является сопроцессором к CPU

- Имеет собственную память

- Выполняет одновременно очень много нитей

Особенностью GPU является то, что задача разбивается на подзадачи, которые можно решаются независимо друг от друга, каждая из этих подзадач решается набором взаимодействующих между собой нитей. Нити группируются в блоки фиксированного размера, блоки объединяются в сеть блоков, ядро выполняется на сетке из блоков. Как уже стало понятно, что каждая отдельная нить представляет собой поток. Но кол-во этих нитей в блоке все же ограниченное количество. Возможна 1D, 2D, 3D топология блоков. В моём случае 2D, так как производится умножение матриц, а не одномерных массивов или кубов.

Для начала производим создание матриц. Выделяем память 1024\*1024\*sizeof(int) под каждую матрицу, в том числе результирующую. Заполняю исходные матрицы рандомными числами. Для более точного результата теста умножение матриц будет производиться 20 раз. После этого нам требуется добавить матрицы в GPU. Требуется выделить память под матрицы именно на графическом устройстве. После чего копируются исходные массивы в буффер GPU с помощью функции cudaMemcpy с параметром cudaMemcpyHostToDevice.

Теперь происходит запуск самого ядра. Для этого пишется команда

<<< , >>> угловые скобки, внутри которых задаются

параметры запуска ядра:

- Кол-во блоков в сетке (в данном случае кол-во блоков будет равно кол-ву строк, деленных на 16 потоков и кол-ву столбцов на 16 потоков)

- Кол-во потоков в блоке (16,16)

Моя видеокарта не позволяет выделять больше, чем (16,16) потоков на 2D блок. При запуске ядра запускается функция с параметром \_global\_, которая означает, что вычисления будут производиться на графическом устройстве (GPU / device), а вызываться из CPU(host). Далее происходит само умножение матриц. С помощью функции cudaDeviceSynchronize(); программа ожидает завершения вычислений на GPU. После того, как GPU окончит вычисления, необходимо перенести результирующий массив обратно из GPU с помощью функции cudaMemcpy с параметром cudaMemcpyDeviceToHost.

Освобождаем выделенную на GPU под массивы память. Вызываю функцию cudaDeviceReset, которая очищает все затронутые ресурсы GPU в данном процессе. Освобождаю память, выделенную под матрицы на host.

Алгоритм тестирования скорости передачи по каналам представляет собой организацию клиент-серверных конструкций. Синхронизацию передачи данных по ним и определение времени, которое требуется для записи и считывания буфера заданного размера. После прохода по циклу, в сумме размер переданной и считанной информации будет равен примерно 6 гб.

Алгоритм тестирования MMX представляет собой несложную математическую функцию, которая будет выполнена 20 миллионов раз.

**4.2 Схема алгоритмов**

Функция void resultpoints() позволяет производить расчет набранных баллов как по отдельным тестируемым параметрам в частности, так и набранных баллов всем компьютером с учетом коэффициента важности тестируемых параметров. Алгоритм представлен в Приложении A.

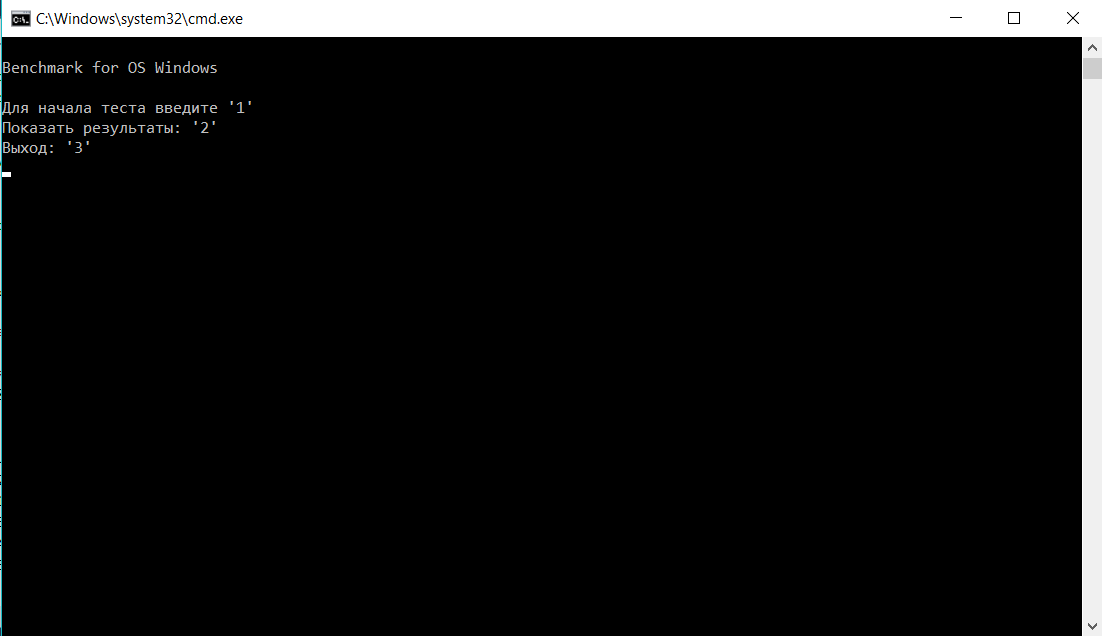
Функция int fun5() разработана для тестирования GPU с помощью CUDA. В функции запускается дочерний процесс, в котором и производится тест. На чертеже будет представлено несколько функций алгоритма тестирования GPU. Алгоритм представлен в Приложении Б.

**5 ТЕСТИРОВАНИЕ, ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

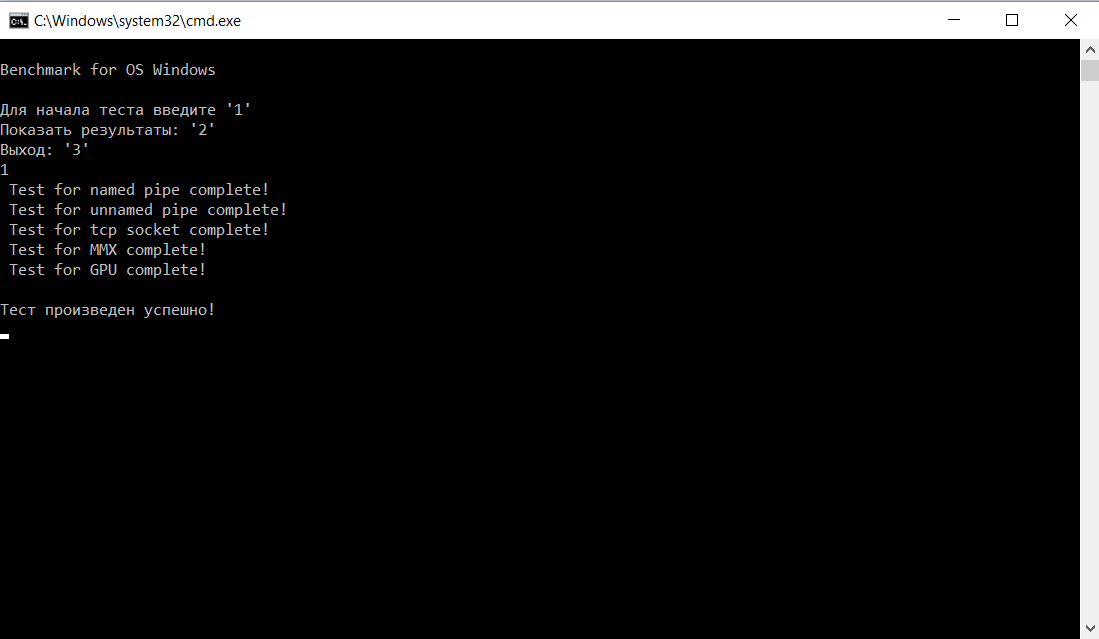
Для обеспечения надлежащего качества приложения необходимо провести полное функциональное тестирование приложения. В ходе тестирования должны быть обнаружены и по возможности исправлены все возможные ошибки, присутствующие в коде.

Целью моей курсовой работы была реализация программы тестирования производительности компьютера. Соответственно, я не ставил перед собой задачу создать графический интерфейс. Меню представляет собой 3 пункта:

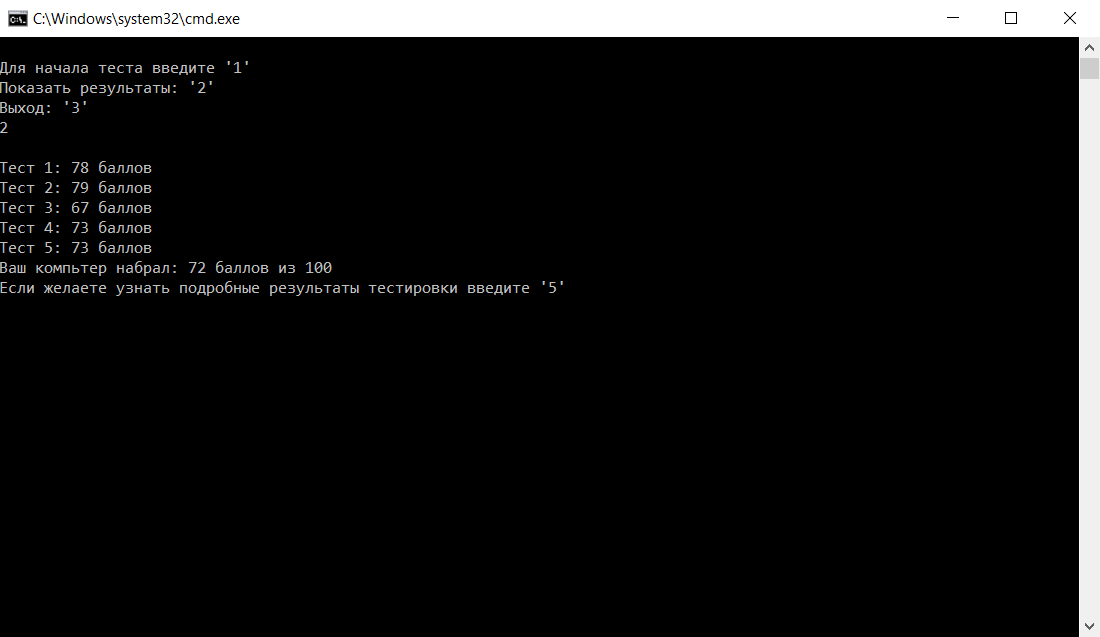
1. Начать тест
2. Показать результаты теста
3. Выход из программы



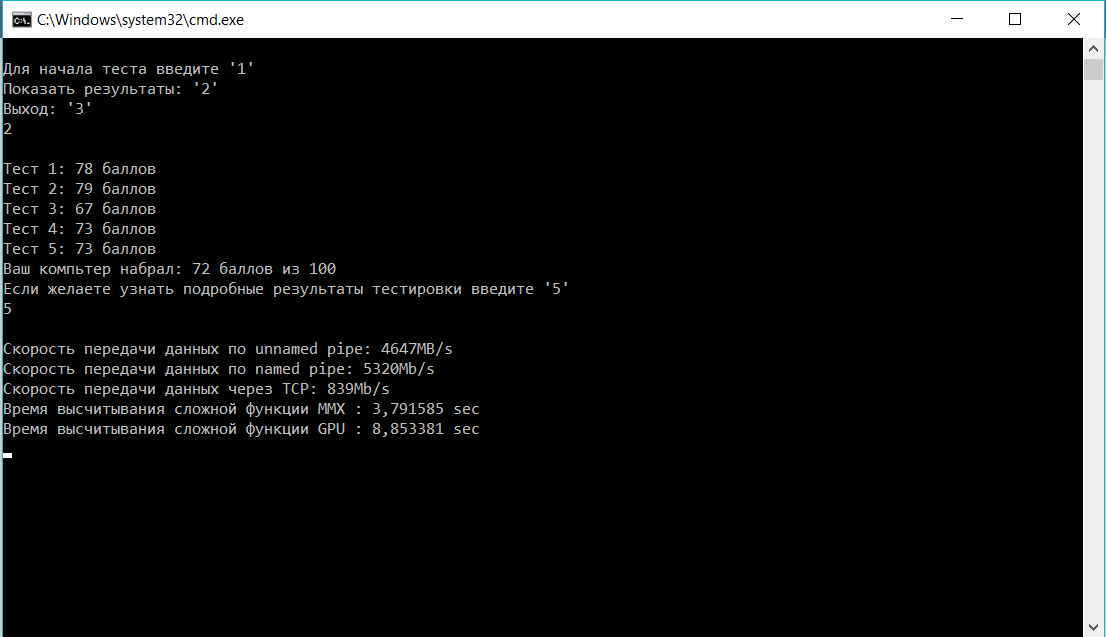
Во время теста на экран будут выводиться сообщения об успешном завершении отдельных тестов.



При выборе второго пункта меню, вам будет представлена информация о набранных баллах в отдельных тестах и компьютера в целом.



После того, как будет получен результат теста производительности компьютера в баллах, вы можете узнать подробные результаты введя цифру 5.



Результаты тестирования моего ноутбука вы можете наблюдать на скриншоте.

Для выхода из программы нужно ввести 3.

Ошибка может возникнуть, если вы захотите увидеть результаты тестирования, предварительно не пройдя её. В этом случае вы будете уведомлены об этом.

**ЗАКЛЮЧЕНИЕ**

Данный курсовой проект является результатом работы, включающей анализ поставленной задачи, исследование существующих аналогов, работу с литературой, разработку алгоритмов, тестирование созданного приложения, составление руководства для пользователя.

Мой бенчмарк предоставляет пользователю возможность протестировать свой персональный компьютер быстро и просто при условии наличия необходимых библиотек.

К достоинствам программы можно отнести простой и понятный интерфейс, что в свою очередь обеспечивает удобство эксплуатации для обычных пользователей.

В процессе программной реализации приложения использовались интеграция Nvidia CUDA Toolkit в Visual Studio, алгоритмы, наиболее подходящие для каждого конкретного случая.

Приложение было подвергнуто множественным тестам, в ходе которых явные неисправности обнаружены не были. Количество тестов было достаточным для того, чтобы утверждать, что приложение будет стабильно работать при любых возможных входных данных.

При создании программного средства преследовались цели воссоединения в себе простоты, функциональности и возможностей. Все это удалось реализовать при помощи литературы.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

[1] Документация по архитектуре клиент-сервер [Электронный ресурс] – Режим доступа: <http://www.4stud.info/networking/lecture5.html>

[2] Документация по установке и использованию Nvidia CUDA Toolkit

Режим доступа: <https://developer.nvidia.com/cuda-toolkit>

[3] Доманов А.Т. Предварительный стандарт предприятия. Дипломные проекты(работы): общие требования / А.Т. Доманов, Н.И. Сороко. – Мн.: БГУИР, 2009.

[4] Системное программное обеспечение ЭВМ в 2-ух частях. / В.А. Супонев, А.А. Уваров, В.А. Прытков

[5] Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. – СПб.: Питер, 2017. – 1120с