# Содержание:

**Техническое задание** ……………………………………………………………………………… 2

**Аннотация** ………………………………………………………………………………………………………… 3

**Введение** …………………………………………………………………………………………………………… 5

# Глава 1. Обзор и анализ подобных программ и выбор

# языка программирования

* 1. Обзор и анализ подобных программ ……………………… 7
  2. Актуальность задачи …………………………………………………… 18
  3. Выбор языка программирования ……………………………… 19

**Глава 2. Разработка алгоритмов**

* 1. Разработка алгоритма ………………………………………………… 22
  2. Взаимодействие между модулями ………………………… 27

**Глава 3. Эксплуатационная часть**

* 1. Руководство программисту ………………………………………… 28

3.2 Руководство пользователю ………………………………………… 38

# Заключение …………………………………………………………………………………………………… 44

**Список используемой литературы** ……………………………………………… 45

**Приложение 1**  *Листинг программы*

**Приложение 2**  *Граф - листы***Техническое задание**

Разработать утилиту диагностики и тестирования основных устройств ЭВМ.

Данная утилита предназначена для определения и тестирования основных устройств ЭВМ, таких как жесткий диск, центральный процессор, оперативная память и т.д.

Системные требования к программе:

* Intel 486DX 50 MHz и выше.
* ОС Windows 95 / 98 / 2000 / ХР.
* ОЗУ 16 Mb.
* Свободного места на диске не менее 2 Mb.**Аннотация**

Данная пояснительная записка содержит информацию о выполненной выпускной работе. Выпускная работа представляет собой утилиту диагностики и тестирования основных устройств ЭВМ.

Пояснительная записка содержит:

Страниц - 45

Рисунков - 21

Граф-листов - 4

Также к выпускной работе прилагается листинг исходной программы, содержащий в своем составе 27 страниц исходного откомпилированного текста программы составленного на языке программирования Borland Delphi 6.0 Enterprise.

**Аннотация**

Бул тушундуруучу катта менин бутуруу ишим жонундо жазылган. Бутуруу ишимдин темасы бул электрондук эсептоочу машинаны диагностика жана тест кылган программа.

Катта томонкулор бар:

Барактар - 45

Суроттор - 21

Чийилген кагаздар - 4

Андан башка бутуруу ишине программанын коду кошулган. Бул программа Borland Delphi 6.0 Enterprise программалоо тилинде жазылган жана анын колому – *27* барак.

**Annotation**

This explanatory note have information about tesis.

Tesis about utility which is making diagnostic main components of PC.

Explanatory note contents:

Pages - 45

Images - 21

Flowgraph - 4

There is also source code of program, which is contains *27* pages of code which was compiled in Borland Delphi 6.0 Enterprise.**Введение**

Компьютер давно стал неотъемлемым атрибутом нашего быта - и на работе, и дома. Но достаточно ли хорошо мы знаем своего железного помощника, его сильные и слабые стороны? Давно известно, что наибольшей эффективностью обладает хорошо сбалансированная система, без явных или скрытых "бутылочных горлышек" отдельных составных частей, препятствующих достижению всей системой максимальной производительности при минимальных материальных затратах.

Вряд ли найдется знаток, умеющий по внешнему виду компьютера отличить одну «троечку» (80386) от другой (Pentium III), или же «четверку» (80486) от другой «четверки» (Pentium IV). А если корпус не серый стандартный, а оригинальный черный или с металлическим блеском, притом необычной компоновки, то задача еще более усложняется: не всегда удастся опознать даже XT. Конечно, неcколько поможет шильдик «Intel Inside» или световое табло с цифрами, которые должны показывать тактовую частоту. Да вот беда, первое может отсутствовать, а информация на втором никак не связана с реальным значением. Да и ПК — это отнюдь не только процессор, не мешает что-нибудь знать и о жестком диске или звуковой плате.

Часть данных удастся получить, если в качестве ОС установлена Windows. А что делать, если ОС не смогла определить тип устройства или вообще его обнаружить? Или установлена не Windows, а какая-либо другая система, которой следует сообщить эти данные? В этом случае на помощь могут прийти специальные утилиты, предназначенные для определения конфигурации компьютера. Большая часть из них написана для DOS. С одной стороны, это объясняется тем, что именно в DOS программа имеет полный доступ к «железу», тогда как в Windows он затруднен. С другой — DOS все-таки является неким «общим знаменателем» для многих ОС, т. е. программы для нее теоретически должны работать везде, хотя, возможно, и с некоторыми ограничениями. Да и загрузить DOS можно с одной дискеты.

Вот почему я принял решение написать данную утилиту, которая могла бы помочь многим пользователям персональных компьютеров узнать побольше информации о своем железном друге.

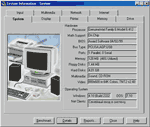
# Глава 1. Обзор и анализ подобных программ и выбор

# языка программирования

**1.1 Обзор и анализ подобных программ**

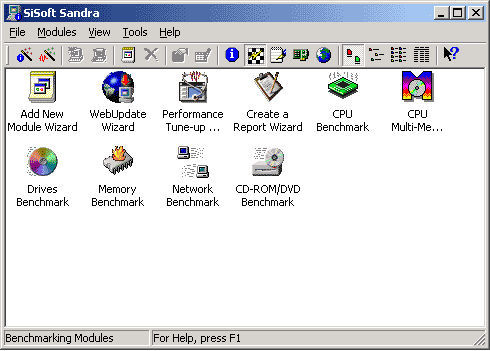
В настоящем обзоре я обозрел продукты, доступные через Сеть.

**Norton Utilities**



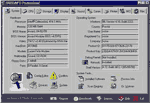
Среди пакетов сервисных утилит, без сомнения, "первым среди равных", является Norton Utilities выпускаемый фирмой Symantec и уже доросший до 2002-й версии. Входящая в пакет утилита System Information предоставляет удобно сгруппированную информацию по всем основным компонентам компьютера. Имеется возможность детализации информации по некоторым разделам, а так же сформировать отчет. Довольно наглядно и красочно, с использованием круговых диаграмм, оформлена информация об эффективности и использовании диска. Протестировать процессор можно, нажав кнопку Benchmark. Программа выдает график примерного быстродействия Вашей системы, за единицу измерения принимается производительность РС на базе процессора Intel 386SX-16MHz. Для сравнения даны рейтинги трех различных компьютеров. При всем моем уважении к действительно великолепному пакету Norton Utilities 2001, утилита System Information является весьма посредственным инструментом для получения исчерпывающей информации о компонентах системы, плюс к этому не очень наглядно, особенно в сравнении с двумя нижеописанными программами, отображаются сведения о системе, ну и совсем добивает своей примитивностью и не внушающими доверия результатами, тестовая подсистема программы. Результат - не более чем удовлетворительно.

**SiSoft SANDRA**



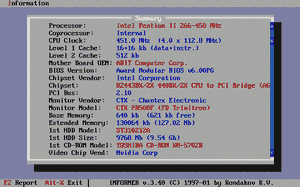
Пакет диагностических утилит SiSoft SANDRA 2001 (аббревиатура расшифровывается как System Analyzer Diagnostic and Reporting Assistant, что означает: помощник в проведении анализа и диагностики системы) является отличным решением для непрофессионального пользователя. В состав полной версии пакета входят около 70 модулей для сбора информации обо всех основных компонентах PC (включая CPU, винчестер, видео- и аудиосистемы, принтеры, коммуникационное оборудование, клавиатура, мышь и пр.), тестирования (проверяются CPU, дисковые накопители, память и сетевое оборудование, включая Burn-in Wizard - мастер проверки системы в экстремальных условиях, поочередно запускающий избранные тестовые модули) и выработки рекомендаций по улучшению их работы. Имеется возможность проверки расположения и содержимого основных конфигурационных файлов. Графический интерфейс программы достаточно нагляден и позволяет получить самую полную информацию о компьютере, включая порой и недокументированную. Главное окно программы напоминает панель управления Windows, только с большим количеством ярлыков. Каждый из них соответствует отдельной утилите, ответственной за сбор и отображение информации об определенном устройстве, входящем в систему, с предоставлением данных о производителе, версии, дате изготовления, быстродействии и т.п. В настоящее время поддерживается ОС Windows 95/98, но, по сообщениям разработчиков, в следующих версиях Sandra будет ориентирована на поддержку Windows 2000/XP, что связано с переходом программы на Unicode, который поддерживается в Windows 9х только частично. Пакет поставляется в двух версиях: профессиональной, являющейся условно-бесплатной, и требующей за регистрацию 29$ и стандартной, полностью бесплатной, но имеющей некоторые ограничения. В частности, отсутствуют ряд дополнительных диагностических модулей, но и оставшихся вполне достаточно для подробной диагностики системы. Оценка однозначная - отлично.

**HARDiNFO PRO**



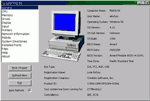
Не менее интересна программа HARDiNFO PRO, разработанная фирмой Ultimate Systems для пользователей ОС Windows 95/98. Кроме нее, существует вариант HARDiNFO 2000, выполняющий те же функции, но оптимизированный для Windows Me/2000. Интерфейс программы очень красочный и наглядный. На каждой из закладок предоставлена исчерпывающая информация о процессоре, аудио-, видео- и коммуникационной подсистемах, принтерах и накопителях. Кроме этого, оттуда же можно запустить соответствующие проверочные тесты и некоторые системные утилиты. Бесплатная оценочная версия программы работоспособна только 14 дней, стоимость регистрации составляет 29 $. Общие впечатления от программы очень хорошие, она вполне достойна наивысшей оценки - отлично.

**Informer**



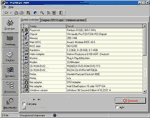
Informer 3.40 - очень неплохая отечественная разработка, предназначенная для анализа конфигурации компьютера и выдающая по ней подробный отчет. Программа предоставляет довольно широкие возможности анализа, так, по заявлениям разработчика, имеется возможность определения 110 типов процессоров и 19 - сопроцессоров, причем измерение тактовой частоты осуществляется с точностью 0.1МГц. Распознаются 2574 моделей мониторов, 324 производителей материнских плат, большое количество различных накопителей и AGP/PCI-устройств и определяются режимы их работы. Как недостаток можно отметить полное отсутствие возможности оценить производительность отдельных компонентов системы. Программа Informer предназначена для использования под управлением ОС DOS, поэтому ей присущи все издержки примитивного псевдографического интерфейса, но это в какой-то мере компенсируется очень скромными ее размерами - всего 200 Kb, что позволяет использовать утилиту с обычной дискеты. Informer, кроме DOS, прекрасно работает и под Windows 95/98/Ме. Распространяется бесплатно. Итоговую оценку можно определить как хорошо.

**Cool Info**



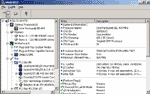
Не очень известна у нас в стране утилита Cool Info 99, последняя версия которой, 2.7i, предназначена для работы под управлением Windows 95/98 и полностью совместима с различными версиями Windows NT. Как и большинство других аналогичных утилит, Cool Info 99 легко справляется с определением большинства компонентов современных компьютеров: процессоров, мониторов, видео- и аудиокарт, дисковых накопителей, модулей памяти, принтеров, сетевых и коммуникационных систем, клавиатур, мышей. Кроме этого, выводится подробная информация о файловой системе, установленных драйверах, шрифтах, и пр. Наряду с удобным графическим интерфейсом пользователя, Cool Info 99 обеспечивает возможность записи полученных результатов в базе данных или текстовом формате, а вот замерить производительность оборудования не получится. Cool Info 99 распространяется как shareware-продукт, поэтому, после 30 дней ознакомления, производители просят заплатить 20 долларов. Программа добротная, но без изюминки, оценка - удовлетворительно.

**Dr.Hardware**



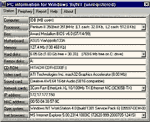
Еще со времен DOS одной из самых популярных утилит, тестирующих компьютерное "железо", была Dr.Hardware. Сейчас уже актуальна 2001-я ее версия, причем имеются варианты для Windows 95/98/Ме (Standard/Professional), для Windows NT/2000 (Professional). Стандартная версия - самая дешевая, предназначена для домашних пользователей, профессиональная - дополнена средствами автоматического создания отчетов и имеет возможность контроля мониторинга системной платы. Существует еще и версия Premium, аналогичная Professional, но поддерживающая все операционные системы семейства Windows: 95/98/Ме/NT/2000. В процессе анализа Dr. Hardware собирает сведения об установленном оборудовании не только из системного реестра, но и используя установки BIOS-а, а так же непосредственного обращаясь к портам ввода/вывода. Это позволяет, помимо созданного ОС списка оборудования, дополнительно проверить каждый компонент. В результате Dr. Hardware может предоставить информацию даже в случае некорректной установки компонента. Результаты тестирования представлены в виде круговых и линейных диаграмм, что очень удобно. Программа регулярно обновляется и поэтому может распознавать самые современные процессоры, чипсеты и другие новые компоненты системы. При всех достоинствах Доктора Харда, некоторая тяжеловесность его внешнего оформления не позволяет выставить ему наивысший балл, но хорошо - твердое.

**HWiNFO**



Рекордсменом по частоте обновлений можно признать утилиту HWiNFO (Hardware Info), обновления которой выходят ежемесячно. Под одним названием здесь, фактически, представлены две утилиты: HWiNFO для DOS (последняя версия - 4.7.6) и HWiNFO32 для WindowsNT/2000 (последняя версия - 0.9beta). HWiNFO32, как указано на сайте разработчика, поддерживает и Windows 95/98, но очень ограничено и без гарантии корректности результатов. Зачем вообще тогда декларировать такую поддержку? Как очевидное достоинство можно отметить весьма скромные, для программы такого типа, размеры, составляющие всего 516Kb. Набор анализируемых устройств достаточно широк и HWiNFO практически не уступает другим аналогичным программам. Кроме обычных бенчмарков процессора и накопителей, имеется возможность провести тестирование CPU на ошибки, кроме этого, отображаются данные, снимаемые с сенсоров мониторинга системной платы. Хотя внешний вид программы небросок, она вполне заслуживает хорошей оценки, но, из-за некорректной работы в самой распространенной ОС - Windows 9x, получается только - удовлетворительно.

**PC Information**



Информационная программа PC Information дает довольно полное представление о компьютере пользователя. Эта информация включает данные о процессоре, системной плате, BIOS-е, мониторе, памяти, дисковых накопителях, сетевых, видео- и звуковых платах, модемах, принтерах, мышке, параметрах сетевого соединения, а так же об установленных операционной системе и программном обеспечении. Как обычно, большинство данных собирается автоматически, но отдельная информация об некорректно установленном оборудовании может быть введена вручную. Программа, имея размер 416 Kb, не требует инсталляции и может быть запущена с дискеты. Большим достоинством PC Information является поддержка всех Ос Windows, начиная с версии 3.1 (кто еще помнит такую?). При желании программу можно запускать из командной строки, используя дополнительные параметры. Утилита распространяется по принципу shareware, регистрация за 15 $ позволит сохранять и распечатывать отчет о проделанной работе в текстовом, CSV или HTML-формате. Оценка - хорошо.

Конечно, рассмотренными выше утилитами все многообразие программ, отображающих "железную" начинку компьютера, не ограничивается, но вряд ли те, что остались за рамками обзора, хоть в чем-то принципиально отличаются от уже рассмотренных. По большому счету, любая из них способна дать практически исчерпывающую информацию об аппаратной конфигурации компьютера.

**1.2 Актуальность задачи**

Существуют множество других программ диагностики компьютера, и у Вас может возникнуть вполне естественный вопрос, а в какой степени моя утилита лучше существующих, или хотя бы чем она выделяется из их общего ряда? Лучший ответ Вы получите,если я опишу Вам одну конкретно взятую программу, которая считается одной из наиболее «продвинутых» в своем классе – это Sisoft Sandra 2001(System Analyzer Diagnostic and Reporting Assistant).

В состав полной версии пакета входят около 70 модулей для сбора информации обо всех основных компонентах PC (включая CPU, винчестер, видео- и аудиосистемы, принтеры, коммуникационное оборудование, клавиатура, мышь и пр.), тестирования (проверяются CPU, дисковые накопители, память и сетевое оборудование) и выработки рекомендаций по улучшению их работы. Имеется возможность проверки расположения и содержимого основных конфигурационных файлов. Графический интерфейс программы достаточно нагляден и позволяет получить самую полную информацию о компьютере, включая порой и недокументированную.

Единственный недостаток Сандры – это то, что ее профессиональная версия стоит 29 долларов.

Исходя из вышеуказанного, я пришел к выводу, что единственным достоинством моей утилиты является ее бесплатность для конечного пользователя.**1.3. Выбор языка программирования**

Потребность в разработке и применении эффективных и адекватных реальной действительности компьютерных программ и технологий сегодня возрастает. Компьютерная технология незаменима, поскольку она даёт возможность оптимизировать и рационализировать управленческую функцию за счет применения новых средств сбора, передачи и преобразования информации.

Выпускная работа написана в программной среде Delphi. Это наводит на вопрос, а почему выбран именно этот язык программирования?

Delphi обладает широким набором возможностей, начиная от проектировщика форм и кончая поддержкой всех форматов популярных баз данных. Среда устраняет необходимость программировать такие компоненты Windows общего назначения, как метки, пиктограммы и даже диалоговые панели. Работая в Windows , вы неоднократно видели одинаковые «объекты» во многих разнообразных приложениях. Диалоговые панели (например, Choose File и Save File) являются примерами многократно используемых компонентов, встроенных непосредственно в Delphi, который позволяет приспособить эти компоненты к имеющийся задаче, чтобы они работали именно так, как требуется создаваемому приложению. Также здесь имеются предварительно определенные визуальные и не визуальные объекты, включая кнопки, объекты с данными, меню и уже построенные диалоговые панели. С помощью этих объектов можно, например, обеспечить ввод данных просто несколькими нажатиями кнопок мыши, не прибегая к программированию. Та часть, которая непосредственно связана с программированием интерфейса пользователя системой, получила название визуальное программирование.

Визуальное программирование как бы добавляет новое измерение при создании приложений, давая возможность изображать эти объекты на экране монитора до выполнения самой программы. Без визуального программирования процесс отображения требует написания фрагмента кода, создающего и настрающего объект «по месту». Увидеть закодированные объекты было возможно только в ходе исполнения программы. При таком подходе достижение того, чтобы объекты выглядели и вели себя заданным образом, становится утомительным процессом, который требует неоднократных исправлений программного кода с последующей прогонкой программы и наблюдения за тем, что в итоге получилось.

Благодаря средствам визуальной разработки можно работать с объектами, держа их перед глазами и получая результаты практически сразу. Способность видеть объекты такими, какими они появляются в ходе исполнения программы, снимает необходимость проведения множества операций вручную, что характерно для работы в среде, не обладающей визуальными средствами — вне зависимости от того, является она объектно-ориентированной или нет. После того, как объект помещен в форму среды визуального программирования, все его атрибуты сразу отображаются в виде кода, который соответствует объекту как единице, исполняемой в ходе работы программы.

Размещение объектов в Delphi связано с более тесными отношениями между объектами и реальным программным кодом. Объекты помещаются в вашу форму, при этом код, отвечающий объектам, автоматически записывается в исходный файл. Этот код компилируется, обеспечивая существенно более высокую производительность, чем визуальная среда, которая интерпретирует информацию лишь в ходе исполнения программы.

**Глава 2. Разработка алгоритмов**

**2.1. Разработка алгоритма**

Рассмотрим алгоритм работы основной программы.



В *структурной схеме* программы показаны основные взаимосвязи между отдельными модулями программы.



**Глава 3. Эксплуатационная часть**

**3.1. Руководство программистa**

В программе были применены, большое количество компонентов Windows, различные обработчик событий, процедуры и функции. При разработке программы было создано 3 модуля (Main, Diag, Example). В модуле Main былa примененa 1 процедурa.

**Краткое описание основных использованных функций**

Используем функцию GetComputerName для получения имени компьютера, функцию GetUserName для получения имени пользователя и функцию GetSystemInfo для получения информации о процессоре (наиболее полно данная функция реализована в Windows NT, где она возвращает и кол-во процессоров и их тип и т.д.).

Перейдем к параметрам экрану. Здесь мы будем использовать и Win32 API функции и стандартные объекты VCL. Так для получения разрешения экрана нам понадобится объект TScreen (его свойства Width и Height). Остальные параметры мы получим через контекст драйвера устройства DC используя функцию GetDeviceCaps.

Также будет интересна информация о памяти. Здесь нам поможет функция GlobalMemoryStatus, возвращающая информацию по объему физической и виртуальной памяти.

Узнаем информацию о ОС. Функция GetWindowsDirectory вернет путь к каталогу, где установлена система, функция GetSystemDirectory - к системному каталогу. Для определения версии ОС воспользуемся функцией GetVersionEx.

Опишем функцию BIOSInfo с параметром, характеризующем текущую ОС. Важно отметить, что способ получения информации о дате BIOS различен. Для NT получим информацию из реестра, а для Windows 95/98 из соответствующего участка памяти. Эти два способа взаимоисключаемы, так как у Windows 95/98 нет соответствующего раздела реестра, а прямой доступ к памяти в NT невозможен.

Рассмотрим функцию SystemParametersInfo, которая позволяет управлять некоторыми настройками системы. Область применения данной функции для NT и Windows 95/98 различна. Умышленно выберем некоторую общую часть для обеих систем.

Также можно позволить пользователю изменять и сохранять настройки системы по своему вкусу. Здесь можно использовать функцию SystemParametersInfo. Для компонентов tbKeyboardSpeed, tbKeyboardDelay, cbScreenSaverActive, cbSpeaker, edSSTimeOut в ObjectInspector перейдем на закладку Events и изменим событие OnChange (для tbKeyboardSpeed, tbKeyboardDelay) , OnClick (для cbScreenSaverActive, cbSpeaker) и OnExit для edSSTimeOut на Change.

Использование Delphi совместно c фунциями Microsoft Win32 API позволит программисту создать более функционально богатые и гибкие приложения.

***Процедуры модуля Main:***

procedure TForm11.Timer1Timer(Sender: TObject);

Данная процедура используется для показа сплэш-окна при начальной загрузке программы. После своего выполнения она показывает главную форму.

***Процедуры модуля Diag:***

procedure TDiadnostic.AboutClick(Sender: TObject);

Процедура выполняет функцию открытия окна или формы. Это то же можно реализовать двумя способами:

1. Form1.Show;
2. Form1.Visible:=True;

procedure GetPrName(processor1:Tlabel);

Процедура определяет тип процессора.

procedure GetRegInfoWinNT;

Процедура используется для получения информации из реестра Windows. В частности, используется для выода информации о Базовой Системе Ввода-Вывода.

    В большинстве случаев очень важной оказывается информация о типе BIOS. Строка типа BIOS хранится по адресу 0FFA68, а строка даты BIOS по адресу 0FFFF5. Это физические адреса, следовательно адреса "сегмент:смещение": 0F000:FA68 и 0F000:FFF5.  
  
    При включении компьютера BIOS инициализирует свои ресурсы и ищет, начиная с адреса 0C0000, ПЗУ установленных карт. Каждое ПЗУ имеет подпись, которая символизируется байтами 55AAh, если BIOS находит эти байты, то он узнаёт размер ПЗУ, который хранится в следующем байте и содержит число страниц по 512 байт, после чего по возможности считает контрольную сумму этого ПЗУ (она должна быть равна нулю) и передаёт управление на 4-ый байт. Также BIOS считает контрольную сумму байт CMOS, расположенных в ячейках 10h-2Dh. Так как в этих ячейках хранится важная, для продолжения работы, информация и её повреждения недопустимы.  
  
    Исходя из этого, программа определяет тип BIOS и считает контрольные суммы Видео ПЗУ и CMOS.

function GetDisplayDevice: string;

данная функция определяет основные параметры видеокарты, такие как размер памяти и строку данных производителя.

function LocalIP : string;

данная функция возвращает IP адрес текущего компьютера.

Function GetCPUSpeed: Double;

Выясняем тактовую частоту процессора.

function CheckDriveType(ch:char): String;

возвращает тип диска (сменный, жесткий)

procedure TDiadnostic.FormCreate(Sender: TObject);

создает главное окно программы

function getprintername:string;

Возвращает имя принтера, сетевого или локального.

procedure TDiadnostic.Button4Click(Sender: TObject);

выполняет обновление информации о состоянии памяти

procedure TDiadnostic.disknameChange(Sender: TObject);

выполняет смену имени диска

procedure TDiadnostic.FormClose(Sender: TObject; var Action: TCloseAction);

Процедура выполняет функцию выхода из программы. Мгновенный выход из программы.

Выход из программы можно организовать несколькими способами, такими как:

* Application.Terminate;
* Form1.Hide;
* Form1.Visible:=False;
* Form1.Close;

Эти команды практически не отличаются друг от друга. Единственным различием может быть то, что первый выход работает гораздо быстрее, т.е. приложение закрывается мгновенно.

function OpenCD(Drive : Char) : Boolean;

выполняет функцию открытия сд-рома

function CloseCD(Drive : Char) : Boolean;

выполняет функцию закрытия сд-рома.

procedure TDiadnostic.SpeedButton1Click(Sender: TObject);

Процедура выполняет функцию открытия окна или формы. В данном случае открывается окно тестирования процессора.

***Процедуры модуля Example:***

procedure TForm1.Button1Click(Sender: TObject);

выполняет измерение тактовой частоты процессора.

Следует подробнее рассказать об идентификации процессора. Как известно, в процессорах пятого поколения, а также в некоторых четвёртого поколения, появилась команда CPUID. Эта команда позволяет больше узнать об установленном в системе процессоре. Обмен информацией происходит через основные регистры (EAX, EBX, ECX, EDX). Параметры задаются через регистр EAX. Таким образом, чтобы узнать информацию о версии процессора, я записал в EAX 1 и вызвал CPUID, после чего в EAX хранился результат. Для того, чтобы узнать производителя процессора, я записал в EAX 0 и вызвал CPUID, после чего в основных регистрах находится, уникальная для каждого производителя, строка. Для AMD это "AuthenticAMD", для Intel - "GenuineIntel", для Cyrix - "CyrixInstead". Теперь расскажу об идентификации процессоров, не имеющих инструкции CPUID. Понятно, что столь подробную информацию, в этом случае получить невозможно. Т.к. программа требует процессор не ниже третьего поколения, я начал проверку именно с этого поколения. Наличие 386-го процессора определяется недоступностью для записи 18-го бита регистра флагов, если бит доступен для записи, то я проверяю наличие поддержки команды CPUID, если она не поддерживается, то процессор 486-ой. Если же CPUID поддерживается, то дальше всё определение ложится именно на эту команду.  
  
    Наличие поддержки команды CPUID определяется доступностью для записи 21-го бита регистра флагов.

procedure TForm1.pcc2PrecizeProc(Sender: TObject);

используется для тестирования скорости выполнения арифметических операций, а именно для вычисления времени сложения 100 целых чисел.

procedure TForm1.pcc3PrecizeProc(Sender: TObject);

используется для вычисления времени сложения 100 целых 64-битных чисел.

procedure TForm1.pcc4PrecizeProc(Sender: TObject);

используется для сложения 100 вещественных чисел

procedure TForm1.Button2Click(Sender: TObject);

собственно выполняет 3 вышеперечисленных процедуры

procedure TForm1.pcc5PrecizeProc(Sender: TObject);

используется для тестирования скорости системной шины

procedure TForm1.Button3Click(Sender: TObject);

при нажатии данной кнопки процессор последовательно производит вычисления над 64-мя операциями NOP, расположенными в первом случае в памчти, во втором- в кэше, и потом вычисляет задержку на передачу команд из памяти.

procedure TForm1.Button4Click(Sender: TObject);

используется для тестирования скорости вызова приложения в оперативную память (вызывается блокнот).

procedure TForm1.pcc7PrecizeProc(Sender: TObject);

здесь тестируется скорость заполнения кэша

procedure TForm1.pcc8PrecizeProc(Sender: TObject);

выясняем количество тактов, необходимых для выполнения одной операции NOP

NOP - Пустая операция.

---------------------------------------------------------

O D I T S Z A P C

----------------------------------------------------------------¬

¦Код Инструкция Такты Описание ¦

¦операции ¦

+--------T---------T-----T-----T-----T------T-------------------+

¦ ¦ ¦486 ¦386 ¦286 ¦86 ¦ ¦

+--------+---------+-----+-----+-----+------+-------------------+

¦90 ¦NOP ¦1 ¦3 ¦3 ¦3 ¦Нет операции. ¦

L--------+---------+-----+-----+-----+------+--------------------

Операция NOP не выполняет никакой операции. NOP - это одно-

байтовая инструкция, которая занимает место, но но не влияет на

содержимое машины (кроме (E)IP).

NOP - это псевдоним инструкции XCHG (E)AX, (E)AX.

procedure TProcessorClockCounter.TestPrecizeProc;

данная процедура помещает небольшой код в кэш память

procedure TProcessorClockCounter.TestPrecizeProcInCache;

данная процедура тестирует кусок кода, уже находящийся в кэш-памяти

procedure TDiadnostic.SpeedButton2Click(Sender: TObject);

данная процедура высвечивает сообщение о том, что программа тестирования памяти загружена в оперативную память.

    Тестированию подвергается память, расположенная выше первого мегабайта, во избежание проблем. Программа использует три разновидности тестирования: Бегущая Единица, Бегущий Ноль, Шахматная доска. Перед тем как начать тестирование, программа определяет объём памяти, установленной в системе. Для этого в последние четыре байта, каждого мегабайта, записывается число, затем оно читается и сравнивается. Если оно не совпадает с записанным, значит этого и последующих мегабайт памяти не существует.  
  
    Особенности применяемых тестов следующие. Тест "Бегущая Единица" или "Бегущий Ноль" состоит в том, что в каждый байт памяти записывается значение 0FFh или 0 соответственно, затем это значение сравнивается. Несовпадение говорит об ошибке. В тесте "Шахматная доска" в память записываются значения вида 10101010b, затем они сдвигаются и сравниваются. Несовпадение - ошибка.

**3.2. Руководство пользователю**

Программа функционирует по следующему принципу:

1

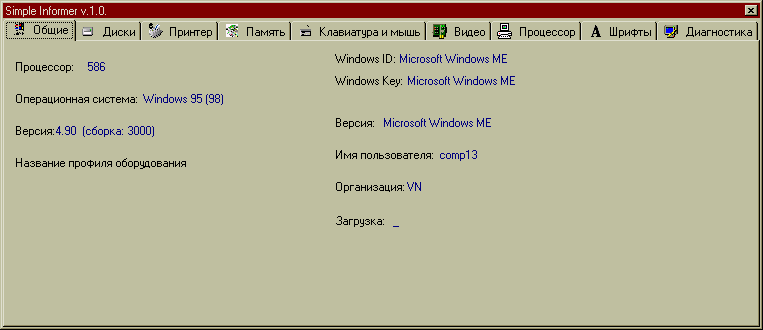
При первом запуске программы появляется окно с предложением подождать, пока программа соберет информацию о системе.



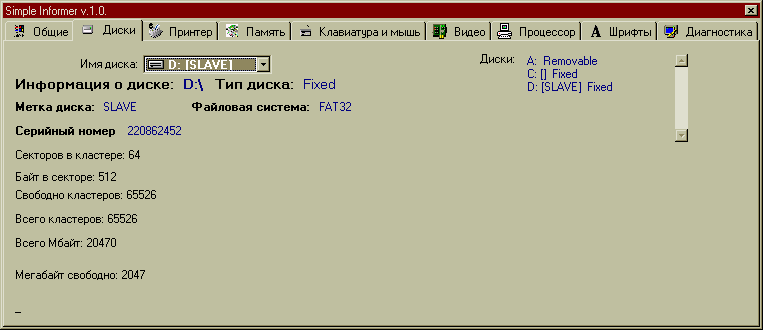
после этого появляется главное окно программы

в нем содержатся следующие закладки:

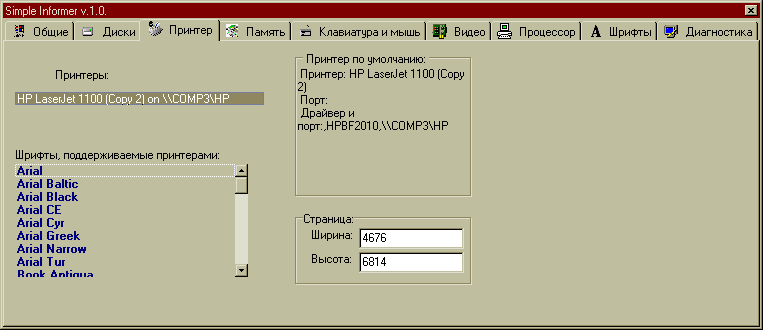
1. Общие – общая информация, т.е. тип процессора, операционная система, версия ОС, имя пользователя, организация.



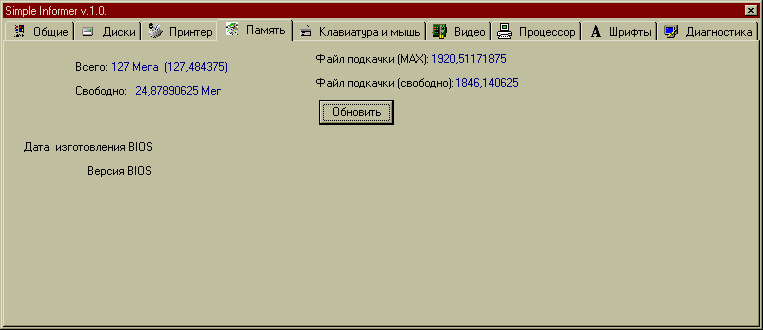
1. Диски – информация о всех дисках, установленных в системе, таких как жесткие диски, СД-РОМ, флоппи-диски, а также информация об их емкости, метке  
   тома, и файловой системе.



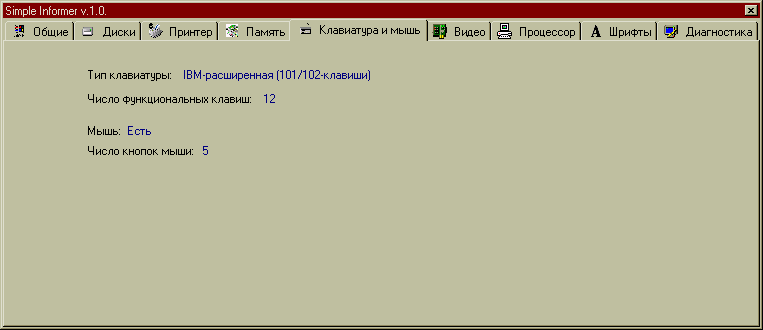
1. Принтер – информация о принтерах, установленных в системе



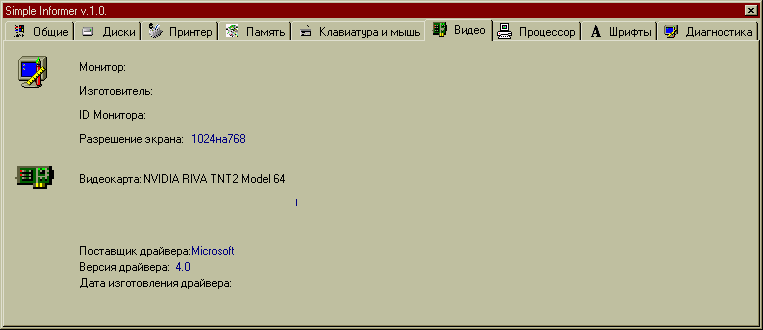
1. память – информация о системной памяти, о ее количестве, информация о файле подкачки.



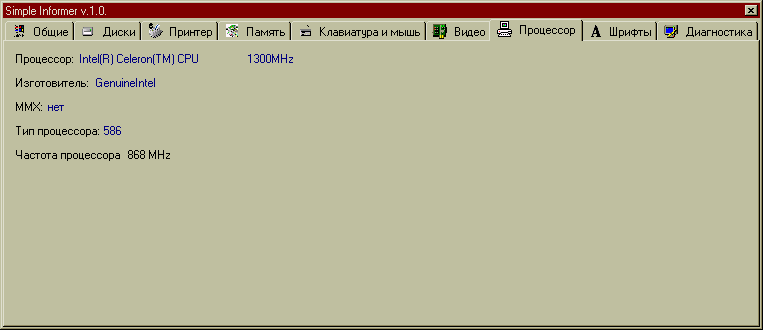
1. Клавиатура и мышь – информация о установленных в системе клавиатуре и мыши.



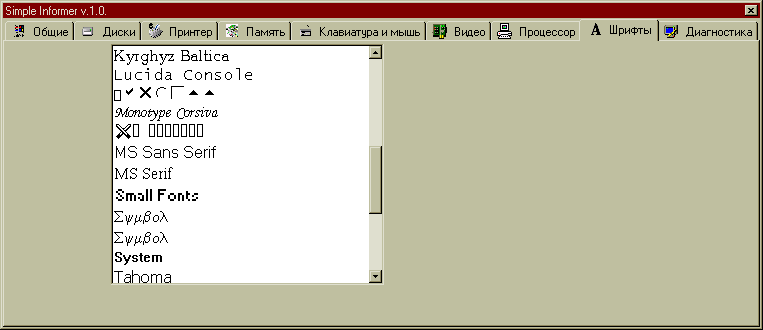
1. Видео- информация о видеокарте, разрешении монитора, поставщике драйвера и т.д.



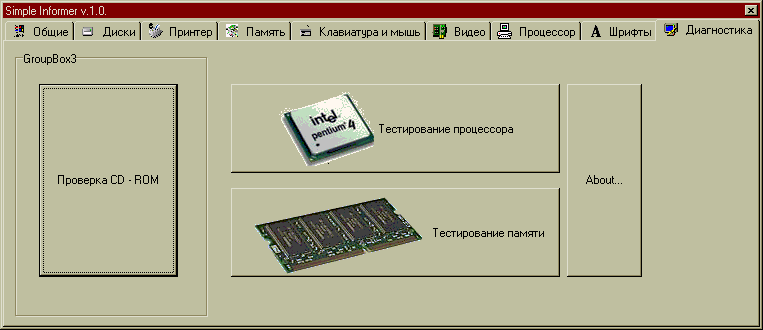
1. информация о процессоре – вендор, частота



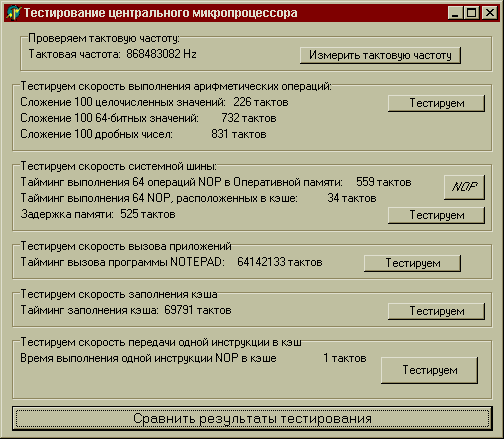
1. шрифты – информация о установленных шрифтах.



1. Диагностика – здесь собраны такие свойства, как тестирование процессора, памяти и проверка лотка СД-рома, а также информация об авторе этой маленькой программы



1. при нажатии на кнопку тестирования процессора выпадает нижеуказанное окно, где производится тестирование процессора.  
     
   здесь, можно потестировать очень много параметров. Перечислим наиболее главные из них:



-тактовая частота – здесь применяется встроенный в микропроцессор счетчик тактов, с которого мы получаем текущую тактовую частоту

-скорость выполнения арифмеических операций – здесь мы тестируем, насколько быстро процессор выполняет основные арифметические операции с различными типами данных

-скорость системной шины – здесь мы проверяем скорость шины между процессором и памятью, т.е. за сколько тактов передается информация между ними. Используется инструкции НОП, расположенные в первом случае в оперативной памяти, а во вотром случае – в кэш-памяти.

В конце программа вычисляет задержку памяти, т.е. сколько тактов требуется именно на передачу данных из / в память .

-скорость вызова приложений – здесь тестируем, за сколько тактов вызывается в оперативнюу память стандартная программа Блокнот

-скорость заполнения кэша – тама мы заполняем все 256 или 512 килобайтов кэша инструкциями, и смотрим, за сколько тактов он заполнится.

**Заключение**

Известно, что выпускная работа завершает подготовку бакалавра и показывает готовность выпускника решать теоретические и практические задачи в условиях реальной трудовой деятельности.   
Цель выпускной работы – систематизация и углубление теоретических и практических знаний студента по специальности и возможности их применения в конкретных условиях практической деятельности. Поэтому то, как студент выполнил выпускную работу, показывает, как он подготовлен.

В данной выпускной работе мною рассмотрена программа диагностики и тестирования компьютера, и в процессе ее написания я более хорошо понял назначение и принцип работы основных устройств персонального компьютера. Вышеозначенные знания, несомненно, пригодятся мне в дальнейшей моей трудовой деятельности. Я очень благодарен преподавательскому составу нашей кафедры за привитую мне способность учиться, невзирая на лень и другие обстоятельства.

Что касается социальной(общественной ценности) данной работы, то я уверен, что для меня она очень значима, так как в процессе разработки я научился терпимости по отношению к программам и вообще у меня получилась очень хорошая утилитка.

**Список используемой литературы**

1. С. Бобровский “DELPHI 5” Учебный курс Москва 2000г.
2. Справочник функций WinAPI.

**Приложение 1**  *Листинг программы*

// главный модуль

unit Main;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, jpeg, ExtCtrls;

type

TForm11 = class(TForm)

Image1: TImage;

Timer1: TTimer;

Label1: TLabel;

procedure Timer1Timer(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form11: TForm11;

implementation

uses Diag;

{$R \*.dfm}

procedure TForm11.Timer1Timer(Sender: TObject);

begin

diadnostic.show;

timer1.Enabled:=false;

end;

end.

// собственно модуль диагностики

unit Diag;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

StdCtrls, ComCtrls, Registry,Printers, ExtCtrls, AxCtrls, OleCtrls, vcf1, Tabs, Winspool,

FileCtrl, ImgList, Menus,winsock,ScktComp, Systeminfo,mmsystem, Buttons,shellapi;

type

TDiadnostic = class(TForm)

SysInfo1: TSysInfo;

Timer1: TTimer;

Button1: TButton;

SpeedButton1: TSpeedButton;

SpeedButton2: TSpeedButton;

GroupBox3: TGroupBox;

About: TButton;

procedure AboutClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure aClick(Sender: TObject);

procedure disknameClick(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure disknameChange(Sender: TObject);

procedure ListBox1DrawItem(Control: TWinControl; Index: Integer;

Rect: TRect; State: TOwnerDrawState);

procedure ListBox1MeasureItem(Control: TWinControl; Index: Integer;

var Height: Integer);

procedure ListBox1Click(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure Timer1Timer(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure SpeedButton1Click(Sender: TObject);

procedure SpeedButton2Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Diadnostic: TDiadnostic;

implementation

uses tlhelp32, about, example;

{$R \*.DFM}

function GetRootDir:string; external 'Ulandll.dll' index 1;

function getboottype:string; external 'UlanDll.dll';// index 31;

procedure TDiadnostic.AboutClick(Sender: TObject);

begin

form2.show;

end;

procedure GetPrName(processor1:Tlabel);

var SI:TsystemInfo;

begin

GetSystemInfo(SI);

Case SI.dwProcessorType of

386:Processor1.caption:='386';

486:Processor1.caption:='486';

586:Processor1.caption:='586';

686:Processor1.caption:='686';

end;

end;

procedure GetRegInfoWinNT;

var

Registryv : TRegistry;

RegPath : string;

sl,sll : TStrings;

begin

RegPath := '\HARDWARE\DESCRIPTION\System';

registryv:=tregistry.Create;

registryv.rootkey:=HKEY\_LOCAL\_MACHINE;

sl := nil;

try

registryv.Openkey(RegPath,false);

diadnostic.Label28.Caption:=(RegistryV.ReadString('SystemBiosDate'));

sl:= ReadMultirowKey(RegistryV,'SystemBiosVersion');

diadnostic.memo1.Text:=sl.Text;

except

end;

Registryv.Free;

if Assigned(sl) then sl.Free;

end;

function GetDisplayDevice: string;

var

lpDisplayDevice: TDisplayDevice;

begin

lpDisplayDevice.cb := sizeof(lpDisplayDevice);

EnumDisplayDevices(nil, 0, lpDisplayDevice , 0);

Result:=lpDisplayDevice.DeviceString;

end;

procedure getinfovideo;

var

lpDisplayDevice: TDisplayDevice;

dwFlags: DWORD;

cc: DWORD;

begin

diadnostic.memo2.Clear;

lpDisplayDevice.cb := sizeof(lpDisplayDevice);

dwFlags := 0;

cc:= 0;

while EnumDisplayDevices(nil, cc, lpDisplayDevice , dwFlags) do

begin

Inc(cc);

diadnostic.memo2.lines.add(lpDisplayDevice.DeviceString);

{Так же мы увидим дополнительную информацию в lpDisplayDevice}

end;

end;

function LocalIP : string;

type

TaPInAddr = array [0..10] of PInAddr;

PaPInAddr = ^TaPInAddr;

var

phe : PHostEnt;

pptr : PaPInAddr;

Buffer : array [0..63] of char;

I : Integer;

GInitData : TWSADATA;

begin

WSAStartup($101, GInitData);

Result := '';

GetHostName(Buffer, SizeOf(Buffer));

phe :=GetHostByName(buffer);

if phe = nil then Exit;

pptr := PaPInAddr(Phe^.h\_addr\_list);

I := 0;

while pptr^[I] <> nil do begin

result:=StrPas(inet\_ntoa(pptr^[I]^));

Inc(I);

end;

WSACleanup;

end;

Function GetCPUSpeed: Double;

const

DelayTime = 500;

var

TimerHi : DWORD;

TimerLo : DWORD;

PriorityClass: Integer;

Priority : Integer;

begin

PriorityClass := GetPriorityClass(GetCurrentProcess);

Priority := GetThreadPriority(GetCurrentThread);

SetPriorityClass(GetCurrentProcess, REALTIME\_PRIORITY\_CLASS);

SetThreadPriority(GetCurrentThread, THREAD\_PRIORITY\_TIME\_CRITICAL);

Sleep(10);

asm

dw 310Fh // rdtsc

mov TimerLo, eax

mov TimerHi, edx

end;

Sleep(DelayTime);

asm

dw 310Fh // rdtsc

sub eax, TimerLo

sbb edx, TimerHi

mov TimerLo, eax

mov TimerHi, edx

end;

SetThreadPriority(GetCurrentThread, Priority);

SetPriorityClass(GetCurrentProcess, PriorityClass);

Result := TimerLo / (1000.0 \* DelayTime);

end;

function CheckDriveType(ch:char): String;

var

DriveLetter: Char;

DriveType : UInt;

begin

DriveLetter := Ch;

DriveType := GetDriveType(PChar(DriveLetter + ':\'));

Case DriveType Of

0: Result := '?';

1: Result := 'Path does not exists';

Drive\_Removable: Result := 'Removable';

Drive\_Fixed : Result := 'Fixed';

Drive\_Remote : Result := 'Remote';

Drive\_CDROM : Result := 'CD-ROM';

Drive\_RamDisk : Result := 'RAMDisk'

else

Result := 'Unknown';

end;

end;

function GettingHWProfileName: String;

var

pInfo: TagHW\_PROFILE\_INFOA;

begin

GetCurrentHwProfile(pInfo);

Result := pInfo.szHwProfileName;

end;

procedure TDiadnostic.FormCreate(Sender: TObject);

var OsVerInfo:Tosversioninfo;

winver,build:string;

Disks:byte;

buffer:array[0..255]of char;

wd:string;

sp:array[0..max\_path-1]of char;

s:string;

memorystatus:tmemorystatus;

dwLength:DWORD; // sizeof(MEMORYSTATUS)

dwMemoryLoad:DWORD; // percent of memory in use

dwTotalPhys:DWORD ; // bytes of physical memory

dwAvailPhys:DWORD ; // free physical memory bytes

dwTotalPageFile:DWORD ; // bytes of paging file

dwAvailPageFile:DWORD ;// free bytes of paging file

dwTotalVirtual:DWORD ;// user bytes of address space

dwAvailVirtual:DWORD ; // free user bytes

ktype:integer;

R:Tregistry;

R2:Tregistry;

disk1:integer;

msgtext:string;

const monitorregdir:string='\system\currentcontrolset\ENUM\Display\Default\_Monitor';

videordir:string='\System\currentcontrolset\services\class\display\0000';

processordir:string='Hardware\Description\System\Centralprocessor\0';

begin

button2.click;

Label50.Caption:=GettingHWProfileName;

listbox1.items:=screen.fonts;

numofbuttons.caption:=inttostr(getsystemmetrics(sm\_cmousebuttons));

if getsystemmetrics(sm\_mousepresent)<>0then ismouse.caption:='Есть'else

ismouse.caption:='Нет';

for disk1:=0 to diskname.items.count-1 do

begin

disk.lines.add(diskname.items[disk1]+' '+CheckDriveType(diskname.items[disk1][1]));

end;

{monitor&video}

///////

R:=tregistry.create;

R.RootKey:=HKEY\_LOCAL\_MACHINE;

R.OpenKey(monitorregdir,false);

monitortype.caption:=R.ReadString('DeviceDesc');

monitormanufacturer.caption:=R.ReadString('Mfg');

monitorid.caption:=r.readstring('HardwareID');

R.OpenKey(videordir,false);

//drvdesc.caption:=r.ReadString('DriverDesc');

driverdate.caption:=r.readstring('DriverDate');

drvprovider.caption:=r.readstring('ProviderName');

driverver.caption:=r.readstring('ver');

r.closekey;

r.closekey;

getinfovideo;

//////

{Version BIOS}

GetRegInfoWinNT;

{advanced processor info}

R2:=Tregistry.create;

R2.RootKey:=HKEY\_LOCAL\_MACHINE;

r2.OpenKey(processordir,false);

processorname.caption:=r2.readstring('Identifier');

vident.caption:=r2.readstring('VendorIdentifier');

if not (r2.readstring('MMXIdentifier')='')then

mmx1.caption:=r2.readstring('MMXIdentifier')

else

mmx1.caption:='нет';

Label48.Caption:=inttostr(Trunc(GetCPUSpeed))+' MHz';

{}

{memory}

memorystatus.dwlength:=sizeof(memorystatus);

globalmemorystatus(memorystatus);

physmemory.caption:=floattostr(memorystatus.dwtotalphys div 1024 div 1024)+' Мега '+'('+

floattostr(memorystatus.dwtotalphys / 1024 / 1024)+')';

avail.caption:=floattostr(memorystatus.dwavailphys / 1024 / 1024)+' Мег';

maxpf.caption:=floattostr(memorystatus.dwtotalpagefile / 1024 / 1024);

pffree.caption:=floattostr(memorystatus.dwavailpagefile / 1024 / 1024);

{}

{Windows info}

winid.caption:=getwinid;

winkey.caption:=getwinkey;

ver1.Caption:=getwinname;

username.caption:=getusernme;

//plusver.caption:=getplusvernum;

company.caption:=getorgname;

resolution.caption:=getscreenresolution;

{printer}

try

getprofilestring('windows','device',',,,',buffer,256);

s:=strpas(buffer);

defprn.Lines.add(' Принтер: '+copy(s,1,pos(',',s)-1));

delete(s,1,pos(',',s)-1);

defprn.lines.add(' Порт: '+copy(s,1,pos(',',s)-1));

delete(s,1,pos(',',s)-1);

defprn.lines.add(' Драйвер и порт:'+ s);

except

showmessage('Printer not found');

end;

{keyboard}

ktype:=GetKeyboardType(0);

case ktype of

1:keytype.caption:='IBM PC/XT или совместимая (83-клавииши)';

2:keytype.caption:='Olivetti "ICO" (102-клавиши)';

3:keytype.caption:='IBM PC/AT (84-клавиши) и другие';

4:keytype.caption:='IBM-расширенная (101/102-клавиши)';

5:keytype.caption:='Nokia 1050 and similar keyboards';

6:keytype.caption:='Nokia 9140 and similar keyboards';

7:keytype.caption:='Japanese keyboard';

end;

numoffunckey.Caption:=inttostr(getkeyboardtype(2));

{

typ.hide;

label14.hide;

{windir}

getwindowsdirectory(sp,max\_path);

wd:=strpas(sp);

{windir.caption:=wd;

progrfiles.caption:=getprogramfilesdir;

label13.hide;

label12.hide;

{Windows version}

OSVerInfo.dwOsversioninfosize:=sizeof(osverinfo);

getversionex(osverinfo);

case osverinfo.dwplatformid of

ver\_platform\_win32s:os.caption:='Windows 3.x';

ver\_platform\_win32\_windows:os.Caption:='Windows 95 (98)';

ver\_platform\_win32\_nt:os.caption:='Windows NT';

end;

with osverinfo do

begin

winver:=format('%d.%d',[dwmajorversion, dwminorversion]);

build:=format('%d', [LoWord(dwbuildnumber)]);

osver.caption:=winver;

osver.caption:=osver.caption+' (сборка: '+build+')';

end;

{boot}

{oottype.caption:=getboottype;

{printer}

{Prntrs.items:=Printer.Printers;}

prn.items:=Printer.Printers;

try

fnt.items:=printer.fonts;

except

end;

prn.ItemIndex:=0;

edit2.text:=inttostr(printer.pageheight);

edit1.text:=inttostr(printer.pagewidth);

GetPrName(Processor1);

GetPrName(pt);

resolution.Caption :=inttostr(Screen.Width)+'на'+inttostr(Screen.Height);

timer1.Enabled:=true;

end;

function OpenCD(Drive : Char) : Boolean;

Var

Res : MciError;

OpenParm: TMCI\_Open\_Parms;

Flags : DWord;

S : String;

DeviceID : Word;

begin

Result := False;

S := Drive + ':';

Flags := mci\_Open\_Type or mci\_Open\_Element;

With OpenParm do begin

dwCallback := 0;

lpstrDeviceType := 'CDAudio';

lpstrElementName := PChar(S);

end;

{Эта строчка необходима для правильной работы функции IntellectCD}

Res := mciSendCommand(0, mci\_Open, Flags, Longint(@OpenParm));

IF Res <> 0 Then Exit;

DeviceID := OpenParm.wDeviceID;

try

Res:=mciSendCommand(DeviceID, MCI\_SET, MCI\_SET\_DOOR\_OPEN, 0);

IF Res = 0 Then Exit;

Result := True;

finally

mciSendCommand(DeviceID, mci\_Close, Flags, Longint(@OpenParm));

end;

end;

function CloseCD(Drive : Char) : Boolean;

Var

Res : MciError;

OpenParm: TMCI\_Open\_Parms;

Flags : DWord;

S : String;

DeviceID : Word;

begin

Result := False;

S := Drive + ':';

Flags := mci\_Open\_Type or mci\_Open\_Element;

With OpenParm do begin

dwCallback := 0;

lpstrDeviceType := 'CDAudio';

lpstrElementName := PChar(S);

end;

Res := mciSendCommand(0, mci\_Open, Flags, Longint(@OpenParm));

IF Res <> 0 Then Exit;

DeviceID := OpenParm.wDeviceID;

try

Res := mciSendCommand(DeviceID, MCI\_SET, MCI\_SET\_DOOR\_CLOSED, 0);

IF Res = 0 Then

Result := True;

finally

mciSendCommand(DeviceID, mci\_Close, Flags, Longint(@OpenParm));

end;

end;

procedure Delay(msecs : Longint);

var

FirstTick : Longint;

begin

FirstTick := GetTickCount;

repeat

Application.ProcessMessages;

until GetTickCount - FirstTick >= msecs;

end;

procedure TDiadnostic.Button1Click(Sender: TObject);

var disk1:integer;

begin

for disk1:=0 to diskname.items.count-1 do

begin

if CheckDriveType(diskname.items[disk1][1])='CD-ROM'

then

begin

opencd(diskname.items[disk1][1]);

delay(5000);

closecd(diskname.items[disk1][1]);

end;

end;

end;

procedure TDiadnostic.SpeedButton1Click(Sender: TObject);

begin

form1.show;

end;

procedure TDiadnostic.SpeedButton2Click(Sender: TObject);

begin

//ShellExecute(handle,nil,'mem.exe',nil,nil,sw\_restore);

MessageDlg('Тестирующая программа загружена в оперативную память',mtInformation,[mbok],0);

end;

end.

//модуль тестирования процессора

unit ProcessorClockCounter;

interface

uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type

TClockPriority=(cpIdle, cpNormal, cpHigh, cpRealTime, cpProcessDefined);

TPrecizeProc = procedure(Sender: TObject) of Object;

TProcessorClockCounter = class(TComponent)

private

FCache:array[0..(1 shl 19) - 1] of byte; // 512 Kb NOP instructions is enough to clear cache

FStarted:DWORD;

FClockPriority:TClockPriority;

FProcessHandle:HWND;

FCurrentProcessPriority: Integer;

FDesiredProcessPriority: Integer;

FThreadHandle:HWND;

FCurrentThreadPriority: Integer;

FDesiredThreadPriority: Integer;

FCalibration:int64; //used to

FPrecizeCalibration:int64;

FStartValue:int64;

FStopValue:int64;

FDeltaValue:int64;

FPrecizeProc:TPrecizeProc;

FCounterSupported:boolean;

procedure PrecizeStart;

procedure PrecizeStartInCache;

procedure GetProcInf;

procedure SetClockPriority(Value: TClockPriority);

procedure ProcedureWithoutInstruction; //description is in code

function GetClock:Int64; register;

function GetStarted:Boolean;

protected

procedure AdjustPriority; virtual; // internal used in constructor to setup parameters when class is created in RunTime

function CheckCounterSupported:boolean;

public

constructor Create(AOwner: TComponent); override;

destructor Destroy; override;

procedure Calibrate;

procedure Start;

procedure Stop;

procedure EraseCache;

procedure TestPrecizeProc; virtual;

procedure TestPrecizeProcInCache; virtual;

property Counter:int64 read FDeltaValue; // contain the measured test clock pulses (StopValue - StartValue - Calibration)

property StartValue:int64 read FStartValue; // Value on the begining

property StopValue:int64 read FStopValue; // Value on test finished

property Started:Boolean read GetStarted;

property CurrentClock:int64 read GetClock; // for longer tests this could be use to get current counter

published

property ClockPriority:TClockPriority read FClockPriority write SetClockPriority default cpNormal;

property Calibration:int64 read FCalibration; // this is used to nullify self code execution timing

property OnPrecizeProc:TPrecizeProc read FPrecizeProc write FPrecizeProc; // user can define it for testing part of code inside it

property CounterSupported:boolean read FCounterSupported;

end;

procedure Register;

implementation

procedure Register;

begin

RegisterComponents('ASM Utils', [TProcessorClockCounter]);

end;

constructor TProcessorClockCounter.Create(AOwner: TComponent);

var n:integer;

begin

inherited create(AOwner);

FCounterSupported:=CheckCounterSupported;

for n:=0 to High(FCache)-1 do FCache[n]:=$90; // fill with NOP instructions

FCache[High(FCache)]:=$C3; // the last is the RET instruction

FClockPriority:=cpNormal;

FStarted:=0;

FDesiredProcessPriority:=NORMAL\_PRIORITY\_CLASS;

FDesiredThreadPriority :=THREAD\_PRIORITY\_NORMAL;

AdjustPriority;

Calibrate;

FStartValue:=0;

FStopValue:=0;

FDeltaValue:=0;

end;

destructor TProcessorClockCounter.Destroy;

begin

inherited destroy;

end;

procedure TProcessorClockCounter.GetProcInf;

begin

FProcessHandle:=GetCurrentProcess;

FCurrentProcessPriority:=GetPriorityClass(FProcessHandle);

FThreadHandle:=GetCurrentThread;

FCurrentThreadPriority:=GetThreadPriority(FThreadHandle);

end;

procedure TProcessorClockCounter.AdjustPriority;

begin

GetProcInf;

case FDesiredProcessPriority of

IDLE\_PRIORITY\_CLASS: FClockPriority:=cpIdle;

NORMAL\_PRIORITY\_CLASS: FClockPriority:=cpNormal;

HIGH\_PRIORITY\_CLASS: FClockPriority:=cpHigh;

REALTIME\_PRIORITY\_CLASS: FClockPriority:=cpRealTime;

end;

end;

procedure TProcessorClockCounter.SetClockPriority(Value: TClockPriority);

begin

if Value<>FClockPriority then

begin

FClockPriority:=Value;

case FClockPriority of

cpIdle: begin

FDesiredProcessPriority:=IDLE\_PRIORITY\_CLASS;

FDesiredThreadPriority :=THREAD\_PRIORITY\_IDLE;

end;

cpNormal: begin

FDesiredProcessPriority:=NORMAL\_PRIORITY\_CLASS;

FDesiredThreadPriority :=THREAD\_PRIORITY\_NORMAL;

end;

cpHigh: begin

FDesiredProcessPriority:=HIGH\_PRIORITY\_CLASS;

FDesiredThreadPriority :=THREAD\_PRIORITY\_HIGHEST;

end;

cpRealTime:begin

FDesiredProcessPriority:=REALTIME\_PRIORITY\_CLASS;

FDesiredThreadPriority :=THREAD\_PRIORITY\_TIME\_CRITICAL;

end;

cpProcessDefined:

begin

FDesiredProcessPriority:=FCurrentProcessPriority;

FDesiredThreadPriority :=FCurrentThreadPriority;

end;

end;

Calibrate;

end;

end;

procedure TProcessorClockCounter.TestPrecizeProc;

// This procedure is intended for testing small block of

// code when it must be put in the processor cache

begin

FDeltaValue:=0;

if FCounterSupported and assigned(FPrecizeProc) then

begin

PrecizeStart; // start test

end;

end;

procedure TProcessorClockCounter.TestPrecizeProcInCache;

// This procedure is intended for testing small block of

// code when it is already in the processor cache

begin

FDeltaValue:=0;

if FCounterSupported and assigned(FPrecizeProc) then

begin

EraseCache;

PrecizeStartInCache; // first test will fill processor cache

PrecizeStartInCache; // second test

// generate calibration value for

// code already put in the cache

end;

end;

procedure TProcessorClockCounter.ProcedureWithoutInstruction;

// this is used for calibration! DO NOT CHANGE

asm

ret

end;

procedure TProcessorClockCounter.EraseCache; register;

asm

push ebx

lea ebx,[eax + FCache]

call ebx // force call to code in array :)

pop ebx // this will fill level2 cache with NOPs (For motherboards with 1 Mb level 2 cache,

ret // size of array should be increased to 1 Mb)

// next instructions are never executed but need for proper align of 16 byte.

// Some processors has different execution times when code is not 16 byte aligned

// Actually, (on some processors), internal mechanism of level 1 cache (cache built

// in processor) filling is designed to catch memory block faster, when

// it is 16 byte aligned !!!

nop

nop

nop

nop

nop

nop

end;

function TProcessorClockCounter.GetClock: Int64; register;

asm

push edx

push ebx

push eax

mov ebx,eax

xor eax,eax // EAX & EDX are initialized to zero for

mov edx,eax // testing counter support

DW $310f // This instruction will make exception

sub eax,dword ptr [ebx+FStartValue] // or do nothing on processors wthout

sbb edx,dword ptr [ebx+FStartValue+4] // counter support

sub eax,dword ptr [ebx+FCalibration]

sbb edx,dword ptr [ebx+FCalibration+4]

mov dword ptr [esp + $10],eax

mov dword ptr [esp + $14],edx

pop eax

pop ebx

pop edx

ret

end;

procedure TProcessorClockCounter.PrecizeStartInCache; register;

asm

//this address should be 16 byte aligned

push edx

push ebx

push eax

mov ebx,eax

push eax

mov dword ptr [ebx + FStarted],1 // started:=true

DW $310f //START

mov dword ptr [ebx + FStartValue],eax // startvalue:=counter

mov dword ptr [ebx + FStartValue + 4],edx

mov edx,[ebx + FPrecizeProc + 4] //time equvialent

mov ebx,ebx

nop

nop

nop

call ProcedureWithoutInstruction // call procedure with immediate back

DW $310f //STOP

mov dword ptr [ebx + FStopValue],eax // stopvalue:=counter

mov dword ptr [ebx + FStopValue + 4],edx

sub eax,dword ptr [ebx + FStartValue]

sbb edx,dword ptr [ebx + FStartValue + 4]

mov dword ptr [ebx + FPrecizeCalibration],eax // calibration:=stopvalue - startvalue

mov dword ptr [ebx + FPrecizeCalibration + 4],edx

nop // need for proper align !!!

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

DW $310f //START

mov dword ptr [ebx + FStartValue],eax // startvalue:=counter

mov dword ptr [ebx + FStartValue + 4],edx

mov eax,[ebx + FPrecizeProc + 4]

mov edx,ebx

call [ebx + FPrecizeProc]

DW $310f //STOP

pop ebx

mov dword ptr [ebx + FStopValue],eax // stopvalue:=counter

mov dword ptr [ebx + FStopValue + 4],edx

sub eax,dword ptr [ebx + FStartValue]

sbb edx,dword ptr [ebx + FStartValue + 4]

sub eax,dword ptr [ebx + FPrecizeCalibration]

sbb edx,dword ptr [ebx + FPrecizeCalibration + 4]

mov dword ptr [ebx + FDeltaValue],eax // deltavalue:=stopvalue - startvalue - calibration

mov dword ptr [ebx + FDeltaValue + 4],edx

pop eax

pop ebx

pop edx

ret

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

end;

procedure TProcessorClockCounter.PrecizeStart; register;

asm

//this address should be 16 byte aligned

push edx

push ebx

push eax

call EraseCache // fill cache with NOPs while executing it

mov ebx,eax

push eax

mov dword ptr [ebx + FStarted],1 // started:=true

nop // need for proper align

nop

nop

nop

nop

nop

nop

nop

nop

nop

nop

DW $310f //START

mov dword ptr [ebx + FStartValue],eax // startvalue:=counter

mov dword ptr [ebx + FStartValue + 4],edx

mov edx,[ebx + FPrecizeProc + 4] //time equvivalent

mov ebx,ebx

nop

nop

nop

call ProcedureWithoutInstruction // call procedure with immediate back

DW $310f //STOP

mov dword ptr [ebx + FStopValue],eax // stopvalue:=counter

mov dword ptr [ebx + FStopValue + 4],edx

sub eax,dword ptr [ebx + FStartValue]

sbb edx,dword ptr [ebx + FStartValue + 4]

mov dword ptr [ebx + FPrecizeCalibration],eax // calibration:=stopvalue - startvalue

mov dword ptr [ebx + FPrecizeCalibration + 4],edx

mov eax,ebx

call EraseCache; // fill cache with NOPs while executing it

nop // need for proper align !!!

nop

nop

nop

nop

DW $310f //START

mov dword ptr [ebx + FStartValue],eax // startvalue:=counter

mov dword ptr [ebx + FStartValue + 4],edx

mov eax,[ebx + FPrecizeProc + 4]

mov edx,ebx

call [ebx + FPrecizeProc]

DW $310f //STOP

pop ebx

mov dword ptr [ebx + FStopValue],eax // stopvalue:=counter

mov dword ptr [ebx + FStopValue + 4],edx

sub eax,dword ptr [ebx + FStartValue]

sbb edx,dword ptr [ebx + FStartValue + 4]

sub eax,dword ptr [ebx + FPrecizeCalibration]

sbb edx,dword ptr [ebx + FPrecizeCalibration + 4]

mov dword ptr [ebx + FDeltaValue],eax // deltavalue:=stopvalue - startvalue - calibration

mov dword ptr [ebx + FDeltaValue + 4],edx

pop eax

pop ebx

pop edx

end;

end.

//модуль диагностики

unit Systeminfo;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,

Dialogs,extctrls;

type TDialupAdapterInfo = record //Информация о Dialup адаптере

alignment:dword;

buffer:dword;

bytesrecieved:dword;

bytesXmit:dword;

ConnectSpeed:dword;

CRC:dword;

framesrecieved:dword;

FramesXmit:dword;

Framing:dword;

runts:dword;

Overrun:dword;

timeout:dword;

totalbytesrecieved:dword;

totalbytesXmit:dword;

end;

type TKernelInfo = record

CpuUsagePcnt:dword;

Numthreads:dword;

NumVMS:dword;

end;

type TFATInfo = record

BreadsSec:dword;

BwritesSec:dword;

Dirtydata:dword;

ReadsSec:dword;

WritesSec:dword;

end;

type TVMMInfo = record

CDiscards:dword;

CInstancefaults:dword;

CPageFaults:dword;

cPageIns:dword;

cPageOuts:dword;

cpgCommit:dword;

cpgDiskCache:dword;

cpgDiskCacheMac:dword;

cpgDiskCacheMid:dword;

cpgDiskCacheMin:dword;

cpgfree:dword;

cpglocked:dword;

cpglockedNoncache:dword;

cpgother:dword;

cpgsharedpages:dword;

cpgswap:dword;

cpgswapfile:dword;

cpgswapfiledefective:dword;

cpgswapfileinuse:dword;

end;

type

TSysInfo = class(TComponent)

private

fDialupAdapterInfo:TDialupAdapterInfo;

fKernelInfo:TKernelInfo;

fVCACHEInfo:TVCACHEInfo;

fFATInfo:TFATInfo;

fVMMInfo:TVMMInfo;

ftimer:TTimer;

fupdateinterval:integer;

tmp:dword;

vsize:dword;

pkey:hkey;

regtype:pdword;

fstopped:boolean;

procedure fupdatinginfo(sender:tobject);

procedure fsetupdateinterval(aupdateinterval:integer);

protected

fsysInfoChanged:TNotifyEvent;

public

constructor Create(Aowner:Tcomponent);override;

destructor Destroy;override;

property DialupAdapterInfo: TDialupAdapterInfo read fDialupAdapterInfo;

property KernelInfo: TKernelInfo read fKernelInfo;

property VCACHEInfo: TVCACHEInfo read fVCACHEInfo;

property FATInfo: TFATInfo read fFATInfo;

property VMMInfo: TVMMInfo read fVMMInfo;

procedure StartRecievingInfo;

procedure StopRecievingInfo;

published

property SysInfoChanged:TNotifyEvent read fsysInfoChanged write

fsysInfoChanged;//Это событие вызывается после определённого интервала времени.

property UpdateInterval:integer read fupdateInterval write

fsetupdateinterval default 5000;

end;

procedure TSysInfo.startrecievingInfo;

var

res:integer;

begin

res:=RegOpenKeyEx(HKEY\_DYN\_DATA,'PerfStats\StartStat',0,KEY\_ALL\_ACCESS,pkey);

if res<>0 then

raise exception.Create('Could not open registry key');

fstopped:=false;

// Для Dial Up Адаптера

RegQueryValueEx(pkey,'Dial-Up Adapter\Alignment',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\Buffer',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\Framing',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\Overrun ',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\Timeout',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\CRC',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\Runts',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\FramesXmit',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\FramesRecvd',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\BytesXmit',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\BytesRecvd',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\TotalBytesXmit',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\TotalBytesRecvd',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\ConnectSpeed',nil,regtype,@tmp,@vsize);

// Для VCACHE

RegQueryValueEx(pkey,'VCACHE\LRUBuffers',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\FailedRecycles',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\RandomRecycles',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\LRURecycles',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\Misses',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\Hits',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\cMacPages',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\cMinPages',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VCACHE\cCurPages',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'Dial-Up Adapter\BytesXmit',nil,regtype,@tmp,@vsize);

//Для VFAT

RegQueryValueEx(pkey,'VFAT\DirtyData',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VFAT\BReadsSec',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VFAT\BWritesSec',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VFAT\ReadsSec',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VFAT\WritesSec',nil,regtype,@tmp,@vsize);

//Для VMM

RegQueryValueEx(pkey,'VMM\cpgLockedNoncache',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgCommit',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgSharedPages',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgDiskcacheMid',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgDiskcacheMac',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgDiskcacheMin',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgDiskcache',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgSwapfileDefective',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgSwapfileInUse',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgSwapfile',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cDiscards',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cPageOuts',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cPageIns',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cInstanceFaults',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cPageFaults',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgOther',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgSwap',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgLocked',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'VMM\cpgFree',nil,regtype,@tmp,@vsize);

//Для KERNEL

RegQueryValueEx(pkey,'KERNEL\CPUUsage',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'KERNEL\VMs',nil,regtype,@tmp,@vsize);

RegQueryValueEx(pkey,'KERNEL\Threads',nil,regtype,@tmp,@vsize);

RegCloseKey(pkey);

ftimer.enabled:=true;

end;

destructor tsysinfo.Destroy;

begin

StopRecievingInfo;

ftimer.Destroy;

inherited;

end;

procedure Register;

begin

RegisterComponents('Samples', [TSysInfo]);

end;

end.

// модуль диагностики процессора

unit example;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, ProcessorClockCounter, StdCtrls;

type

TForm1 = class(TForm)

GroupBox1: TGroupBox;

StaticText1: TStaticText;

Button7: TButton;

Button8: TButton;

procedure pcc1PrecizeProc(Sender: TObject);

procedure pcc2PrecizeProc(Sender: TObject);

procedure pcc3PrecizeProc(Sender: TObject);

procedure pcc4PrecizeProc(Sender: TObject);

procedure pcc5PrecizeProc(Sender: TObject);

procedure pcc7PrecizeProc(Sender: TObject);

procedure pcc8PrecizeProc(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

// Тактовая частота

procedure TForm1.pcc1PrecizeProc(Sender: TObject);

begin

sleep(1000); //wait 1 s

end;

procedure TForm1.Button1Click(Sender: TObject);

begin

button1.Caption:='Wait';

button1.Enabled:=false;

pcc1.TestPrecizeProcInCache;

label1.Caption:=IntToStr(pcc1.Counter)+' Hz';

button1.Caption:='Измерить тактовую частоту';

button1.Enabled:=true;

end;

// скорость выполнения арифметических операций

procedure TForm1.pcc2PrecizeProc(Sender: TObject);

var n:integer;

m:integer; // integer variable

begin

for n:=0 to 99 do m:=m+1;

end;

procedure TForm1.pcc3PrecizeProc(Sender: TObject);

var n:integer;

m:Int64; // Int64 variable

begin

for n:=0 to 99 do m:=m+1;

end;

procedure TForm1.pcc4PrecizeProc(Sender: TObject);

var n:integer;

m:single; // single type variable

begin

for n:=0 to 99 do m:=m + 1.0001;

end;

procedure TForm1.Button2Click(Sender: TObject);

begin

pcc2.TestPrecizeProcInCache;

label2.Caption:=IntToStr(pcc2.Counter)+' тактов';

pcc3.TestPrecizeProcInCache;

label3.Caption:=IntToStr(pcc3.Counter)+' тактов';

pcc4.TestPrecizeProcInCache;

label4.Caption:=IntToStr(pcc4.Counter)+' тактов';

end;

// скорость системный шины

procedure TForm1.pcc5PrecizeProc(Sender: TObject);

begin

asm

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; nop;

nop; nop; nop; nop; nop; nop; nop; ret;

end;

end;

procedure TForm1.Button3Click(Sender: TObject);

var cInRAM, cInCache:int64;

begin

pcc5.TestPrecizeProc; // Code is in RAM and will be pulled in cache

cInRAM:=pcc5.Counter;

label5.Caption:=IntToStr(cInRAM)+' тактов';

pcc5.TestPrecizeProcInCache; // Code is already in cache

cInCache:=pcc5.Counter;

label6.Caption:=IntToStr(cInCache)+' тактов';

label7.Caption:=IntToStr(cInRAM-cInCache)+ ' тактов';

end;

// скорость вызова приложений

procedure TForm1.Button4Click(Sender: TObject);

begin

pcc6.Start;

WinExec(PChar('Notepad.exe'),SW\_SHOWNORMAL);

pcc6.Stop;

label8.Caption:=IntToStr(pcc6.Counter)+' тактов';

end;

// Example 5

procedure TForm1.pcc7PrecizeProc(Sender: TObject);

begin

refresh;

end;

procedure TForm1.Button5Click(Sender: TObject);

begin

pcc7.TestPrecizeProcInCache;

label9.Caption:=IntToStr(pcc7.Counter)+ ' тактов';

end;

// скорость заполнения кэша

procedure TForm1.pcc8PrecizeProc(Sender: TObject);

begin

asm nop end;

end;

procedure TForm1.Button6Click(Sender: TObject);

begin

pcc8.TestPrecizeProcInCache;

label10.Caption:=IntToStr(pcc8.Counter)+ ' тактов';

end;

procedure TForm1.Button7Click(Sender: TObject);

begin

MessageDlg('NOP - Пустая операция'#13 +

'это псевдоним инструкции XCHG (E)AX, (E)AX',

mtInformation,[mbok],0);

end;

procedure TForm1.Button8Click(Sender: TObject);

begin

MessageDlg('процессор Pentium IV'#13 +

'с частотой системной шины 400 МГц',

mtInformation,[mbok],0);

end;

end.

Министерство Образования и Культуры

Кыргызской Республики

Кыргызский Технический Университет

им. И. Раззакова.

Кафедра Информатики и Вычислительной Техники



Выпускная Работа

на тему: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил: ст. гр. ЭВМ-1-99

Ыйсаев У.Б.

Принял(а): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Бишкек, 2003 г.