МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети

Выпускная квалификационная работа бакалавра

на тему

База данных универсального дизассемблера Диздек архитектуры IA-32

Студент Щетинин Кирилл Александрович
(подпись, дата) (ФИО полностью)

Руководитель 9.06.20 О.В. Кулагин
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр А.В. Кучин
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 17.06.2020 № 14)

Секретарь ГЭК Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«30» апреля 2020г.

ЗАДАНИЕ
Выпускная квалификационная работа бакалавра

1. Студент Щетинин Кирилл Александрович гр. 16ВВ1 факультета вычислительной техники

2.Тема работы База данных универсального дизассемблера диздек архитектуры IA-32

Тема утверждена приказом ПГУ №0335 от "30" апреля 2020 г.

3. Руководитель бакалаврской работы Кулагин О.В..,

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать и отладить приложение, предназначенное для дизасемлированния задданых пользователем команд. Разработать и отладить базу данных демонстрирующее, работоспособность разработанного программного модуля.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: ассемблирование и дизасемблирование, рассмотреть основные языки программироваия для создания приложений, изучить функционал языка программирования Delphi

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	10	24.02.2020 - 04.03.2020	
2.	Анализ требований	10	05.03.2020 - 14.03.2020	
3.	Выбор технологий, применяемых при проектировании и разработке	10	15.03.2020 - 24.03.2020	
4.	Разработка структуры проекта	10	25.03.2020 - 03.04.2020	
5.	Разработка Базы данных	15	04.04.2020 - 18.04.2020	
6.	Разработка приложения	35	19.04.2020 - 22.05.2020	
7.	Оформление пояснительной записки	10	23.05.2020 - 08.06.2020	

Дата выдачи задания "13" февраля 2020г.

Руководитель бакалаврской работы 9.06.20 О.В. Кулагин
(подпись, дата)

Задание к исполнению принял студент 30.04.2020 Щетинин К.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 103 листа, 14 рисунков, 3 таблицы, 5 источников, 2 приложения.

IA-32, DELPHI, ДИЗАСЕМБЛИРОВАНИЕ, ДЕКОМПИЛЯЦИЯ

Объектом исследования является приложение, предназначенное для дизассемблирования и декопиляции.

Предметом исследования является организация алгоритмов дизассемблирования.

Цель работы – разработать приложение, предназначенное для дизассемблирования и декопиляции.

Объектом разработки является приложение, обеспечивающая выполнение следующих функций: декомпилиция, дизассмеблирование.

В результате проведенной работы было разработано приложение, удовлетворяющая поставленную задачу, а именно: декомпилиция, дизассмеблирование.

Содержание

Введение 6
1.Алгоритмы дизассемблирования 7
1.1 Базовый алгоритм дизассемблирования 7
1.2 Алгоритм линейной развертки 7
1.3 Алгоритм рекурсивного спуска. 8
2 Общая характеристика Delphi 11
3. Общая характеристика дизассемблера Диздек. 19
4. Формат машинной комманды архитектуры IA-32 20
5. Принцип построения дизассемблера Диздек. 25
6.Пример работы 28
7.Охрана труда 36
Заключение 39
Список используемых источников 40
Приложение А. Листинг программы на Delphi 41
Приложение Б.Презентация 99

Введение

Дизассемблирование – преобразование программы на машинном языке к ее ассемблерному представлению. Декомпиляция – получение кода языка высокого уровня из программы на машинном языке или ассемблере. Декомпиляция – достаточно сложный процесс. Это обусловлено следующими причинами:

Процесс компиляции происходит с потерями. В машинном языке нет имен переменных и функций, и тип данных может быть определен только по производимым над ними операциям. Наблюдая пересылку 32-х бит данных, требуется значительная работа, чтобы определить, являются ли эти данные целым числом, дробью или указателем. − Компиляция — это операция типа множество-множество. Компиляция и декомпиляция могут быть выполнены множеством способов. Поэтому результат декомпиляции может значительно отличаться от исходного кода. − Декомпиляторы в значительной степени зависимы от конкретного языка и библиотек. − Необходимо точное дизассемблирование исполняемого файла. Любая ошибка на фазе дизассемблирования размножатся в результирующем коде.

Цель инструментов дизассемблирования заключается в содействии исследованию функционирования программ, когда их исходные коды не доступны. Наиболее распространенные цели дизассемблирования: − анализ вредоносного программного обеспечения; − анализ уязвимостей программного обеспечения с закрытым исходным кодом; − анализ совместимости программного обеспечения с закрытым исходным кодом; − валидация компилятора; − отображение команд программы в процессе отладки.

1.Алгоритмы дизассемблирования

1.1 Базовый алгоритм дизассемблирования

1) Первым шагом в процессе дизассемблирования является идентификация кодового сегмента. Так как команды обычно смешаны с данными, то дизассемблеру необходимо их разграничить.

2) Получив адрес первой команды, необходимо прочитать значение, содержащееся по этому адресу (или смещению в файле) и выполнить табличное преобразование двоичного кода операции в соответствующую ему мнемонику языка ассемблера.

3) Как только команда была обнаружена и декодирована, ее ассемблерный эквивалент может быть добавлен к результирующему листингу. После этого необходимо выбрать одну из разновидностей синтаксиса языка ассемблера.

4) Далее необходимо перейти к следующей команде и повторить предыдущие шаги до тех пор, пока каждая команда файла не будет дизассемблирована.

1.2 Алгоритм линейной развертки

Алгоритм линейной развертки использует прямолинейный подход при выборе очередной команды для дизассемблирования: где завершается код одной команды - начинается код новой. В результате, наиболее сложной задачей становится определение первой команды. Дизассемблирование начинается с первого байта в сегменте кода и последовательно продвигается, обрабатывая команды одну за другой, пока не будет достигнут конец сегмента. При этом не производится попыток понять логику передач управления в программе посредством распознавания команд перехода таких, например, как условия. Главное преимущество алгоритма линейной развертки состоит в

полном покрытии кодового сегмента. Одним из основных недостатков является невозможность распознать данные, если они совмещены с кодом. Это очевидно из листинга 1.1, демонстрирующего результат дизассемблирования функции при помощи данного алгоритма. Эта функция содержит конструкцию switch. Компилятор принял решение реализовать switch как таблицу переходов. Более того, компилятор предпочел разместить таблицу переходов внутри самой функции. Конструкция jump по адресу 401250 ссылается на таблицу адресов начиная с 410257. К сожалению, дизассемблер рассматривает ее как набор команд и неверно генерирует соответствующее представление на языке ассемблера.

```
40123f:    55                       push   ebp
 401240:    8b ec                    mov    ebp,esp
 401242:    33 c0                    xor    eax,eax
 401244:    8b 55 08                 mov    edx,DWORD PTR [ebp+8]
 401247:    83 fa 0c                 cmp    edx,0xc
 40124a:    0f 87 90 00 00 00        ja     0x4012e0
❶  401250:    ff 24 95 57 12 40 00     jmp    DWORD PTR [edx*4+0x401257]
❷  401257:    e0 12                    loopne 0x40126b
 401259:    40                       inc    eax
 40125a:    00 8b 12 40 00 90        add    BYTE PTR [ebx-0x6fffbfee],cl
 401260:    12 40 00                 adc    al,BYTE PTR [eax]
 401263:    95                       xchg   ebp,eax
 401264:    12 40 00                 adc    al,BYTE PTR [eax]
 401267:    9a 12 40 00 a2 12 40     call   0x4012:0xa2004012
 40126e:    00 aa 12 40 00 b2        add    BYTE PTR [edx-0x4dffbfee],ch
 401274:    12 40 00                 adc    al,BYTE PTR [eax]
 401277:    ba 12 40 00 c2           mov    edx,0xc2004012
 40127c:    12 40 00                 adc    al,BYTE PTR [eax]
 40127f:    ca 12 40                 lret   0x4012
 401282:    00 d2                    add    dl,dl
 401284:    12 40 00                 adc    al,BYTE PTR [eax]
 401287:    da 12                    ficom  DWORD PTR [edx]
 401289:    40                       inc    eax
 40128a:    00 8b 45 0c eb 50        add    BYTE PTR [ebx+0x50eb0c45],cl
 401290:    8b 45 10                 mov    eax,DWORD PTR [ebp+16]
401293:    eb 4b                    jmp    0x4012e0
```

Рисунок 1 — Алгоритм линейной развёртки

1.3 Алгоритм рекурсивного спуска.

Алгоритм рекурсивного спуска использует концепцию передачи управления, определяющую, должна ли команда быть дизассемблирована по наличию или отсутствию на нее ссылок от других команд. Для понимания алгоритма рекурсивного спуска, полезно классифицировать команды в

зависимости от их влияния на счетчик команд CPU. Команды, не влияющие на счетчик команд. После выполнения такой команды управление переходит непосредственно к следующей команде. Примерами такой команды может послужить арифметические, такие как add; Для подобных команд процесс дизассемблирования такой же как и при линейной развертке. Команды условного перехода. Команды условного перехода, такие как x86 jnz, образуют две возможных ветви исполнения. Поскольку в статическом контексте обычно невозможно определить исход проверки условия, алгоритм рекурсивного спуска дизассемблирует обе ветви. Адрес целевой ветви добавляется в список адресов для последующего дизассемблирования. Дизассемблирование продолжается последовательно, так, как если бы условие было ложно. Команды безусловного перехода. Безусловные переходы приводят к нарушению последовательного порядка исполнения команд. Команда, получающая управление после выполнения безусловного перехода, может располагаться от него на значительном расстоянии. Кроме того, как видно из листинга 1.1, команды, следующие непосредственно за командой безусловного перехода не исполняются вообще. Таким образом, необходимость в их дизассемблировании отпадает. Алгоритм рекурсивного спуска пытается определить адрес назначения безусловного перехода и занести его в список адресов для последующего дизассемблирования. К сожалению, не все безусловные переходы могут быть корректно обработаны данным алгоритмом. Когда адрес назначения перехода зависит от параметра, получаемого в процессе исполнения, его определение методами статического анализа становится невозможным. Команды вызова функции. Команды вызова функции работают сходным образом с командами безусловных переходов (включая невозможность определить адрес назначения команды, такой как call eax), за исключением того, что после выполнения функции, управление обычно возвращается команде, следующей за ее вызовом. При

этом, как и в случае с условными переходами, образуются две ветви исполнения. Адрес назначения команды call добавляется в список адресов для последующего дизассемблирования, в то время как команда, следующая за call, дизассемблируется с использованием алгоритма линейной развертки. Алгоритм рекурсивного спуска может оказаться неэффективным в случае, если при возвращении из вызываемой функции поведение программы отклоняется от ожидаемого. Например, в коде функции может преднамеренно модифицироваться адрес возврата. Команды возврата. В некоторых случаях, алгоритм рекурсивного спуска терпит неудачу. Команда возврата из функции (например, ret x86) не предоставляет информации о том, какая команда будет выполнена далее. Если бы программа была на самом деле запущена, управление было бы передано по адресу, расположенному на вершине стека. У дизассемблера нет возможности доступа к стеку. Вместо этого дизассемблирование внезапно останавливается. В этом случае алгоритм рекурсивного спуска обращается к списку отложенных адресов, и процесс дизассемблирования возобновляется. Этот рекурсивный процесс отражает смысл названия алгоритма. Алгоритм рекурсивного спуска превосходно разделяет код и данные. Главным же недостатком такого подхода является неспособность распознавать ветви, образуемые таким командами как jump и call, использующими для адресации таблицы поиска. Несмотря на это, в сочетании с эвристиками для распознавания указателей на код, алгоритм рекурсивного спуска способен обеспечить хорошее покрытие кода в сочетании с прекрасным разделением кода и данных.

2 Общая характеристика Delphi

Delphi— императивный, структурированный, объектно-ориентированный, высокоуровневый язык программирования со строгой статической типизацией переменных. Основная область использования — написание прикладного программного обеспечения.

Этот язык программирования является диалектом языка Object Pascal. Изначально язык Object Pascal относился к несколько другому языку, который был разработан в фирме Apple в 1986 году группой Ларри Теслера[2]. Однако, начиная с Delphi 7[3] в официальных документах компании Borland название Delphi стало использоваться для обозначения языка, ранее известного как Object Pascal.

Изначально среда разработки Delphi была предназначена исключительно для разработки приложений Microsoft Windows, затем был реализован вариант для платформ Linux (под торговой маркой Kylix), однако после выпуска в 2002 году Kylix 3 его разработка была прекращена, и вскоре было объявлено о поддержке Microsoft .NET, которая, в свою очередь, была прекращена с выходом Delphi 2007.

В настоящее время, наряду с поддержкой разработки 32 и 64-разрядных программ для Windows, реализована возможность создавать приложения для Apple Mac OS X (начиная с Embarcadero Delphi XE2), iOS (включая симулятор, начиная с XE4 посредством собственного компилятора), Google Android (начиная с Delphi XE5)[4], а также Linux Server x64 (начиная с версии 10.2 Tokyo).

Независимая, сторонняя реализация среды разработки проектом Lazarus (Free Pascal, в случае компиляции в режиме совместимости с Delphi) позволяет

использовать его для создания приложений на Delphi для таких платформ, как Linux, Mac OS X и Windows CE.

Также предпринимались попытки использования языка в проектах GNU (например, Notepad GNU) и написания компилятора для GCC (GNU Pascal). При создании языка (и здесь качественное отличие от языка C) не ставилось задачи обеспечить максимальную производительность исполняемого кода или лаконичность исходного кода для экономии оперативной памяти. Изначально язык ставил во главу угла стройность и высокую читаемость, поскольку был предназначен для обучения дисциплине программирования. Эта изначальная стройность в дальнейшем, как по мере роста аппаратных мощностей, так и в результате появления новых парадигм, упростила расширение языка новыми конструкциями.

Так, сложность объектного C++, по сравнению с C, выросла весьма существенно и затруднила его изучение в качестве первого языка программирования, чего нельзя сказать об Object Pascal относительно Pascal.

Ниже перечислены некоторые отличия синтаксических конструкций Delphi от семейства C-подобных языков (C/C++/Java/C#):

- В Delphi формальное начало любой программы чётко отличается от других участков кода и должно располагаться в определённом, единственном в рамках проекта, исходном файле с расширением dpr (тогда как другие файлы исходных текстов программы имеют расширение pas).

  C-подобных языках программирования в качестве входа обычно используется глобальная функция или статический метод с именем main и определённым списком параметров, причём такая функция может быть расположена в любом из файлов исходного текста проекта.

- В Delphi идентификаторы типов, переменных, а равно и ключевые слова читаются независимо от регистра: например, идентификатор SomeVar полностью эквивалентен somevar. Регистро-зависимые идентификаторы в начале компьютерной эпохи ускоряли процесс компиляции, и кроме того, позволяли использовать очень короткие имена, порой отличающиеся лишь регистром.

И хотя к настоящему времени обе эти практики − использование нескольких идентификаторов, различающихся лишь регистром, равно как и чрезмерная их лаконичность, осуждены и не рекомендованы к применению, практически все унаследованные от C языки − C++, Java, C# − являются регистро-зависимыми, что, с одной стороны, требует достаточно большой внимательности к объявлению и использованию идентификаторов, а с другой — принуждает писать более строгий код, когда каждая переменная имеет чётко определённое имя (вариации регистра могут вызвать путаницу и ошибки).

- В Delphi в исходных файлах .pas (которые, как правило, и содержат основное тело программы) на уровне языковых средств введено строгое разделение на интерфейсный раздел и раздел реализации. В интерфейсной части содержатся лишь объявления типов и методов, тогда как код реализации в интерфейсной части не допускается на уровне компиляции. Подобное разделение свойственно также языкам C/C++, где в рамках культуры и парадигмы программирования вводится разделение на заголовочные и собственно файлы реализации, но подобное разделение не обеспечивается на уровне языка или компилятора.

В C# и Java такое разделение утрачено вовсе − реализация метода, как правило, следует сразу же после его объявления. Инкапсуляция обеспечивается лишь принадлежностью метода к той или иной области

видимости. Для просмотра одной только интерфейсной части модуля исходного кода используются специальные средства.

- В Delphi метод или функция чётко определяются зарезервированными для этого ключевыми словами procedure или function, тогда как в C-подобных языках различие обуславливается ключевым словом, определяющим тип возвращаемого значения:

- Тем сложнее в C#/C++ выглядят такие конструкции, как объявление типа «указатель на метод» В Delphi начало и конец программного блока выделяются ключевыми словами begin и end, тогда как в C-подобных языках программирования для этих целей используются фигурные скобки: {}. Таким образом, возможно, в Delphi достигается лучшая читаемость кода для лиц с ослабленным зрением. С другой стороны, фигурные скобки могут быть более интуитивными при визуальном восприятии, выполняя функцию пиктограммы. В Delphi начало и конец программного блока выделяются ключевыми словами begin и end, тогда как в C-подобных языках программирования для этих целей используются фигурные скобки: {}. Таким образом, возможно, в Delphi достигается лучшая читаемость кода для лиц с ослабленным зрением. С другой стороны, фигурные скобки могут быть более интуитивными при визуальном восприятии, выполняя функцию пиктограммы. В Delphi операция присвоения значения переменной обозначается при помощи двоеточия со знаком равенства, :=, что является заимствованием из математической нотации. Знак равенства без двоеточия — это оператор проверки равенства, возвращающий булево значение. Напротив, в C-подобных языках оператором присваивания является одинарный знак равенства, а оператором проверки равенства — двойной, ==. В силу того,

что в этих языках программирования присваивание является лишь выражением, возвращающим значение переменной слева, не так уж редки следующие неочевидные для новичка ошибки. В Delphi объектное и объектно-ориентированное программирование хоть и поощряется, однако не является единственно возможным. Так, допустимо (в отличие от C#) объявление и использование глобальных или статических функций и переменных.

Язык C# принудительно объектен. Глобальные, без привязки к классу, функции запрещены. Value-типы, наподобие структур struct, унаследованы от общего типа C#, несмотря на то, что сами по себе они не могут быть унаследованы (то есть, наследование структур в C# запрещено). Вместе с тем, экземпляры классов C# являются неявно-ссылочными типами, как и в Delphi.

Поскольку системные вызовы в Windows (как, впрочем, и в POSIX-системах наподобие Linux, Mac OS) формально необъектны, взаимодействие C#-кода с ними затруднено даже без учёта разной парадигмы управления временем жизни переменных в памяти. Delphi не имеет подобных ограничений.

Несмотря на такую акцентированную на объектность парадигму, в C# отсутствует понятие виртуального конструктора, то есть создания экземпляра класса, точный тип которого на этапе компиляции неизвестен, а известен лишь базовый класс этого экземпляра. Отчасти этот недостаток может быть скомпенсирован посредством интерфейсов или reflection, однако подобные решения не являются стандартными для языка.

- Для наиболее гибкой и эффективной реализации объектно-ориентированного подхода в Delphi, введены два механизма полиморфного вызова: классический виртуальный, а также динамический:

если в случае классического виртуального вызова, адреса всех виртуальных функций будут содержаться в таблице виртуальных методов каждого класса, то в случае с динамическим вызовом указатель на метод существует лишь в таблице того класса, в котором он был задан или перекрыт.

Таким образом, для динамического вызова из класса D метода класса A, переопределённого в B, потребуется выполнить поиск в таблицах методов классов D, A и B.

Подобная оптимизация имеет своей целью уменьшение размера статической памяти, занимаемой под таблицы методов. Экономия может быть существенна для длинных иерархий классов, с очень большим количеством виртуальных методов. В C-подобных языках динамические полиморфные вызовы не применяются.

- В отличие от C# язык Delphi допускает создание (инстанциацию) экземпляра класса, содержащего абстрактные (не имеющие реализации) методы. Чтобы исключить возможность создания экземпляра класса недостаточно объявить в нём абстрактные методы. Необходимо использовать в описании класса ключевое слово abstract. Таким образом, в настоящее время классы, имеющие абстрактные методы (в отличие от ранних реализаций Delphi), не считаются абстрактными. При помощи механизма виртуальных функций код базового класса, имеющего абстрактные методы, определяет на этапе выполнения, перекрыт ли в фактическом экземпляре класса конкретный абстрактный метод, и в зависимости от этого или вызывает перекрытый метод или создаёт исключение EAbstractError.

Также Delphi допускает перекрытие любого конкретного виртуального метода базового класса абстрактным в классе-потомке:

Прежде чем перейти к сравнению языковой реализации свойств, заметим, что сопоставление этих двух примеров наглядно показывает, что язык C# провоцирует, во-первых, злоупотребление фигурными скобками (что не так страшно в виду краткости их написания), и, во-вторых, нагромождение обязательных спецификаторов доступа к каждому члену класса; в Delphi (как и в C++) однажды объявленный спецификатор применяется ко всем последующим членам. Также, если в Delphi возможна привязка свойства к значению поля, то в C# они всегда снабжаются методами доступа, с использованием скобок составной команды (кроме автоматических свойств). Методы эти, в отличие от Delphi, не могут быть объявлены виртуальными, равно как не могут быть вызваны непосредственно.

Метод доступа в C# всегда относится к одному, и только к одному свойству, тогда как для Delphi это утверждение, вообще говоря, неверно. Более того, один и тот же метод может быть использован для реализации доступа к существенно различным свойствам. Языки Java и C# были изначально спроектированы для создания программ, работающих в управляемой среде, где управлением временем жизни объектов занимается эта среда: так что ручное управление памятью не допускается. Удобство и безопасность этого подхода негативно влияют на производительность.

Попытки снизить издержки сборки мусора могут привести к существенному искажению стиля программирования

В Delphi не существует автоматического управления памятью: (в классических компиляторах языка) экземпляры классов создаются и удаляются вручную, тогда как для некоторых типов — интерфейсов, строк и динамических массивов задействован механизм подсчёта ссылок. Ни один из этих подходов, вообще говоря, не гарантирует отсутствия утечек памяти, но,

с другой стороны, проблема отзывчивости не актуальна, временные издержки управления памяти малы, и, что важнее, очевидны. Также, при отсутствии утечек, общий объём задействованной оперативной памяти существенно меньше, чем у аналогичных приложений, полагающихся на сборщик мусора.

3. Общая характеристика дизассемблера Диздек.

Диздек предназначен для дизассемблирования исполняемых модулей программ разных процессоров с формированием для этих модулей:

1) ассемблерных текстов и единообразных математизированных низкоуровневых описаний программ, пригодных для дальнейших равносильных преобразований;
2) математических описаний алгоритмов программ на основе оригинального математического аппарата – алгебры последовательностей индексных объектов-соответствий (АПИОС) [3-6], позволяющего исключить операторы адресных переходов из описания программы;
3) текстов программ на языках программирования высокого уровня (ЯПВУ): Паскаль в версии Дельфи 6.0 (далее – Паскаль-Дельфи), и Си.
4) описаний схем узлов ПЛИС на языке описания аппаратуры Verilog для аппаратной реализации алгоритма программы в виде узла на ПЛИС.

4. Формат машинной комманды архитектуры IA-32

Префиксы - необязательные однобайтные элементы машинной команды. Назначение префиксов - изменить действия, выполняемые командой. Префиксы могут указываться программистом явно при написании исходного текста программы, либо их, по определенным соображениям, может вставить ассемблер. Процессор распознает префиксы по их значениям. Машинная команда может иметь до четырех префиксов одновременно. В памяти префиксы предшествуют команде. Порядок их следования при этом может быть любым.

Код операции - обязательный элемент, описывающий операцию, выполняемую командой. Код операции может занимать от одного до трех байт. Для некоторых машинных команд часть битов кода операции может находиться в байте mod r/m.

Многим командам соответствует несколько кодов операций, каждый из которых определяет нюансы выполнения операции. Отметим, что поле кода операции не имеет однозначной структуры. В зависимости от конкретных команд, не обязательно разных с точки зрения языка ассемблера, оно может иметь в своем составе от одного до трех элементов. Один из этих трех элементов является непосредственно кодом операции или ее частью, остальные уточняют детали операции. Такое строение поля кода операции усложняет, в частности, процесс дизассемблирования. Для определения размера и границ очередной команды необходимо полностью проанализировать ее поле кода операции.

Рассмотрим назначение дополнительных битов поля кода операции:

Поле	Назначение
d	Определяет направление передачи данных: 0 - передача данных из регистра reg в память (или регистр), адресуемую полем r/m;1 - передача данных из памяти (или регистра), адресуемой полем r/m, в регистр reg. При наличии байта sib адрес операнда в памяти формируется с учетом содержимого этого байта
s	Задает необходимость расширения 8-разрядного непосредственного операнда до 16 или 32 бита. Старшие биты при этом заполняются значением старшего (знакового) бита исходного 8-разрядного операнда
w	Определяет размер данных, которыми оперирует команда: байт, слово, двойное слово: 0 - 8 битов; 1 - 16 битов для 16-разрядного размера операндов или 32 бита для 32-разрядного размера операндов
reg	Определяет регистр, используемый в команде. Значение поля зависит от поля w, в том числе если поле w отсутствует

Таблица 1 ─ дополнительные коды операций

Последующие поля машинной команды определяют характеристики и местоположение операндов, участвующих в операции, и особенности их использования.

Байт режима адресации mod r/m – иногда называемый постбайтом, несет информацию об операндах и режиме адресации. Большинство команд процессора Intel -двухоперандные. Операнды могут находиться в памяти, а также в одном или двух регистрах. Архитектура IA-32 не допускает, чтобы оба операнда команды находились в памяти. Если операнд находится в памяти, то байт mod r/m определяет компоненты (смещение, базовый и индексный регистры), используемые для вычисления его эффективного адреса. Байт mod r/m состоит из трех полей.

─ Поле mod (два бита) определяет способ адресации и количество байтов, занимаемых в команде адресом операнда (поле смещения в команде). Поле mod используется совместно с полем r/m, которое определяет способ модификации адреса операнда полем смещения в команде. Поле mod в комбинации с полем r/m образует 32 возможных значения, обозначающих один из восьми регистров и 24 режима адресации.

Способs адресации
Поле смещения в команде отсутствует и адрес операнда определяется содержимым базового и/или индексного регистра. Какие именно регистры потребуются для вычисления эффективного адреса, определяется значением этого байта.
Поле смещения в команде присутствует, занимает один байт и модифицируется содержимым базового и/или индексного регистра.
Поле смещения в команде присутствует, занимает два или четыре байта (в зависимости от значения, действующего по умолчанию или определяемого префиксом размера адреса) и модифицируется содержимым базового и/или индексного регистра.

Операндов в памяти нет - они находятся в регистрах. Это же значение байта mod используется в случае, когда команда работает с непосредственным операндом

Таблица 2 — Способы адресации

— Поле reg (3 бита) определяет либо регистр, находящийся в команде на месте второго операнда, либо возможное расширение кода операции.

— Поле r/m используется совместно с полем mod и определяет либо регистр, находящийся в команде на месте первого операнда (если mod = 11), либо базовые и индексные регистры, применяемые для вычисления эффективного адреса (совместно с полем смещения в команде).

В архитектуре Intel один из операндов обязательно находится в регистре, и он может быть первым или вторым. Расположение первого и второго операндов в формате команды фиксировано. Но, например, команда mov может выполнять пересылку как из регистра в память, так и из памяти в регистр. В машинном представлении это одна и та же команда. В ее поле reg будет содержаться код регистра, а в поле r/m - код режима адресации. Эти две команды будут различаться только одним битом d, который определяет направление передачи. Если в команде участвуют два регистра, то в этом случае вступает в силу правило: поле reg определяет второй операнд, а поле r/m - первый.

Байт масштаба, индекса и базы (Scale-Index-Base - sib) используется для расширения возможностей адресации операндов. На наличие байта sib в машинной команде указывает сочетание одного из значений 01 или 10 поля mod и значения поля r/m= 100. Байт sib состоит из трех элементов.

— В поле масштаба (ss) размещается масштабный множитель для индексного компонента index, занимающего следующие три бита байта sib. В

поле ss может содержаться значение 1, 2, 4 или 8. При вычислении эффективного адреса на это значение будет умножаться содержимое индексного регистра.

– Поле index позволяет хранить номер индексного регистра, содержимое которого применяется для вычисления эффективного адреса операнда.

– Поле base требуется для хранения номера базового регистра, содержимое которого также применяется для вычисления эффективного адреса операнда. В качестве базового и индексного регистров могут использоваться большинство регистров общего назначения.

Поле смещения в команде - это 8-, 16- или 32-разрядное целое число со знаком, представляющее собой полностью или частично значение эффективного адреса операнда.

5. Принцип построения дизассемблера Диздек.

Основные подходы к построению дизассемблеров можно условно поделить на два типа. Первый тип предполагает построение дизассемблеров с жестким алгоритмом работы, который пользователь никак не может изменить. По такому типу построено большинство существующих дизассемблеров, включая, например, известный дизассемблер Sourcer. Второй тип предполагает построение интерактивных дизассемблеров, алгоритм работы которых конечный пользователь может менять в выгодную для себя сторону. Так построен дизассемблер IDA, в котором пользователь может при помощи встроенного Си-подобного языка практически полностью переписать его функции. С одной стороны, интерактивность дизассемблера позволяет опытному пользователю исправлять недостаки создателей дизассемблера и модифицировать его под свои нужды. Но с другой стороны, использование дизассемблера IDA при верификационных испытаниях любой программы вынудит экспертную организацию одновременно верифицировать две программы: 1) исследуемую программу, и 2) Си-подобную программу, модифицирующую функции дизассемблера.

Поэтому, указанный Си-подобный язык является недостатком дизассемблера IDA, а не его достоинством.

Другими неприятными особенностями IDA являются: 1) его и 2) полное отсутствие технической документации на него. Подстройка инструментального программного обеспечения (ПО) под нужды конечного пользователя действительно является ценным свойством. В Диздеке возможность подстройки пакета под нужды конечного пользователя решается совершенно по-другому, чем в IDA. А автоматическая реконструкция управляющих программных конструкций делает ненужным такие, использованные в IDA «красивости», как стрелки, обозначающие связи в

программе, и всплывающие окна с текстами подпрограмм. Однако, возможность перехода в дизассемблированной программе по указанному в команде адресу и возможность возврата на один ход назад в Диздеке существуют.

В Диздеке создан своего рода инструментарий и однородные информационные структуры (списки), с помощью которых можно решать самые разные задачи по дизассемблированию. Эти компоненты названы объектами. К ним относятся:

1) подпрограмма, дизассемблирующая очередную команду,
2) список косвенных переходов,
3) список точек входа,
4) список безусловных переходов.

Кроме объектов существуют также переменные, к которым можно получить доступ извне. Они названы элементами. К ним относятся:

1) адрес дизассемблируемой команды,
2) адрес команды, следующей за дизассемблируемой командой,
3) адрес перехода из управляющей команды,
4) адрес команды, подлежащей дизассемблированию,
5) начальный адрес интервала дизассемблирования,
6) конечный адрес интервала дизассемблирования,
7) номер используемой точки входа в программу по списку точек входа,
8) 32-разрядный указатель на регистр, определяющий косвенный переход,
9) переменная-признак косвенного перехода.

В процессе дизассемблирования могут встречаться разные ситуации, которые в дальнейшем будут названы условиями. Это прежде всего следующие ситуации:

1) обнаружение косвенного адресного перехода,

2) обнаружение косвенного вызовы подпрограммы,

3) обнаружение прямого адресного перехода,

4) обнаружение прямого вызовы подпрограммы,

5) достижение конца пути дизассемблирования,

6) начало дизассемблирования.

При необходимости могут пользователем могут быть введены и дополнительные условия.

Работа программируемой настройки Диздека сводится к подбору для каждого условия соответствующей управляющей последовательности, которые названы управляющими записями. В них могут выполняться различные действия с объектами и элементами. При возникновении соответствующего условия в процессе своей работы, Диздек обращается к набору управляющих записей – словарю – и извлекает для исполнения оттуда нужную запись. Так управляющие записи определяют конкретный алгоритм работы Диздека. Это может быть и самый примитивный алгоритм, дизассемблирующий все команды подряд без какого-либо их анализа, и алгоритм дизассемблера Sourcer, в котором производится анализ точек входа и отслеживаются адресные переходы. В Диздеке не решена проблема дизассемблирования самомодифицирующегося кода. Ее решение требует постановки соответствующей работы.

Таким образом Диздек предоставляет пользователю некоторый инструментарий, а порядок его использования определяет конечный пользователь при помощи словаря из 29-40 управляющих записей (СУЗ). Исследовать СУЗ можно и без знания правил их написания – Диздек сам сформирует их разъяснения на русском языке.

6.Пример работы

После дизассемблирования программа разбивается на базовые блоки и управляющие команды. Последние преобразуются при помощи АПИОС в конструкции ЯПВУ.

Исходная программа	**Декомпилированная программа (главный модуль)**

.asect “vect“, 0000h
B INIT
.asect “TEKCT“, 0020h
INIT SETC INTM
MAR *,AR7
LAR AR3,#8000h
LDP #0H
LACC #0017H
LACC #000FH
SAMM 2AH
LACC #0FFFFH
SAMM 28H
LACC #0FFFFH
SAMM 28H
ZAP
SACL *
XOP7 LAR AR7,#8000h
LACC #0FFFFH
SAMM 19H
TOZU SAR AR7,*+
CMPR 3
BCND TOZU, TC
LAR AR7,#8001h
LACC #5a51h
SACL *
	unit KPM2OZUC;
	interface
	uses SysUtils, GLOBAL, KPM2OZUC_SBB;
	procedure GLAV_KPM2OZUC;
	implementation
	procedure GLAV_KPM2OZUC;
	var i:word;
	adres:longword;
	begin
	adres:=$00000000;
	repeat
	case adres of
	$00000041:begin {Блок № 1}
	KPMBazblok3;
	adres:=$00000044;
	end;
	$00000000:begin {Блок № 2}
	KPMBazblok1;
	repeat
	KPMBazblok2;
	until not(((regm[$0005] shr 11) and $00000001)=1);
	adres:=$00000041;
	end;
	else begin
	writeln(perekod(' - Сформирован непредусмотренный адрес:'),inttohex(adres,8),'h.');

XOP5 NOP
 	
CLRC XF
 	
.end
	end;
	
	end;
	
	until adres=$00000044;
	
	end;
	
	end.

Таблица 3 – Пример декомпилированной программы.

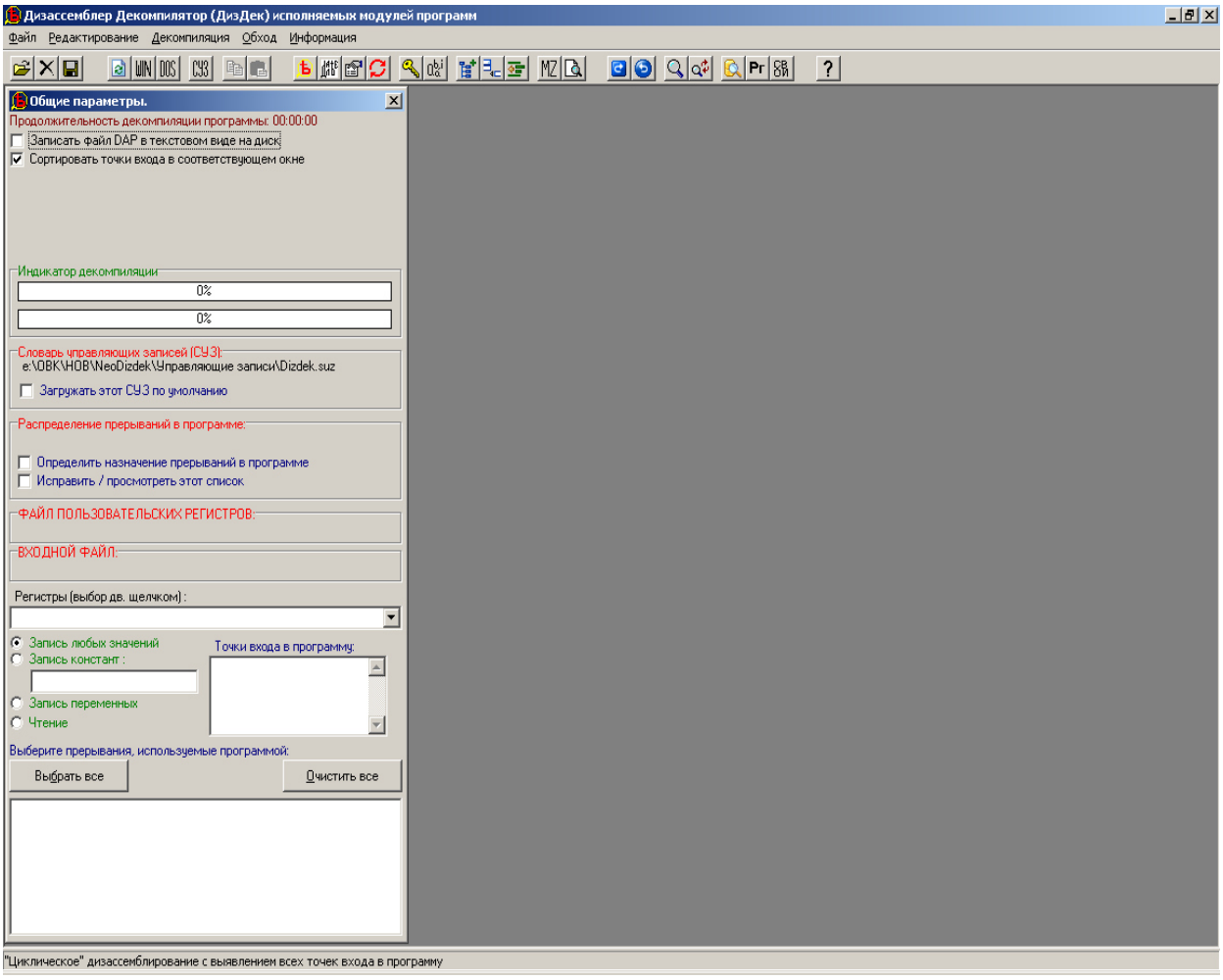


Рисунок.2 - Основное окно программы Диздек.

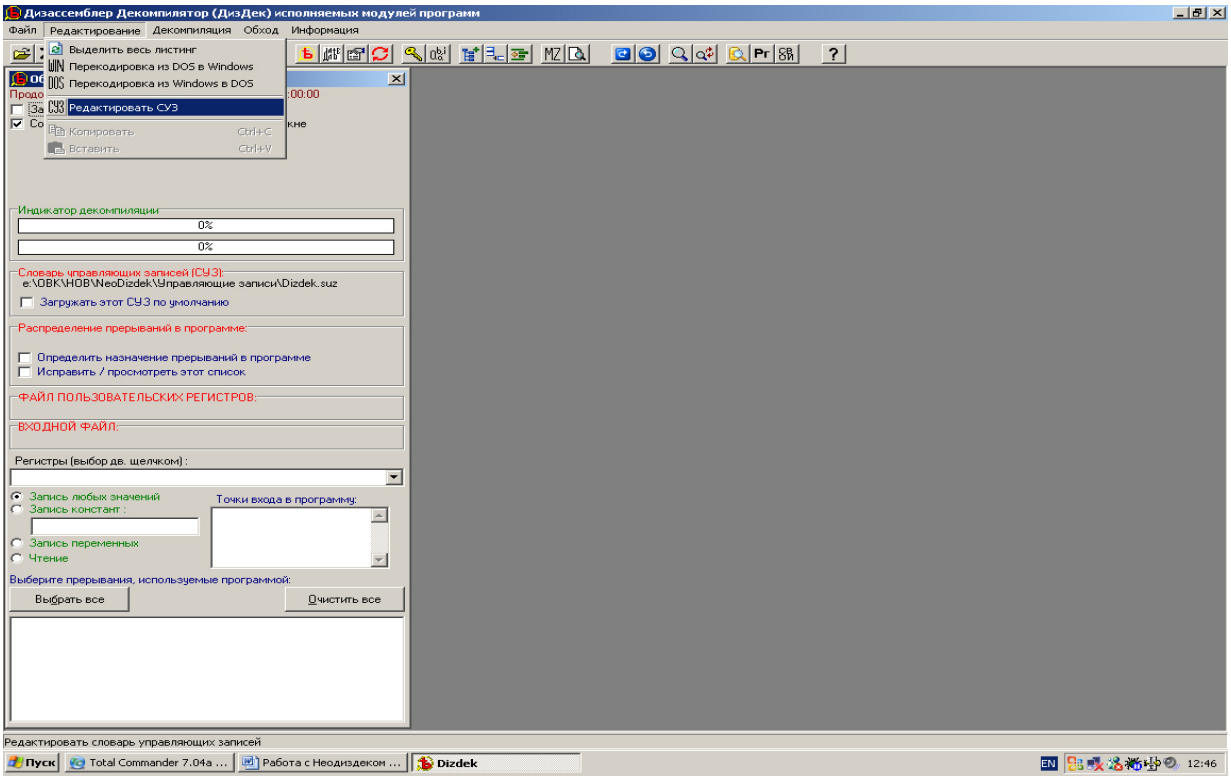


Рисунок.3 - Подменю «Редактирование».

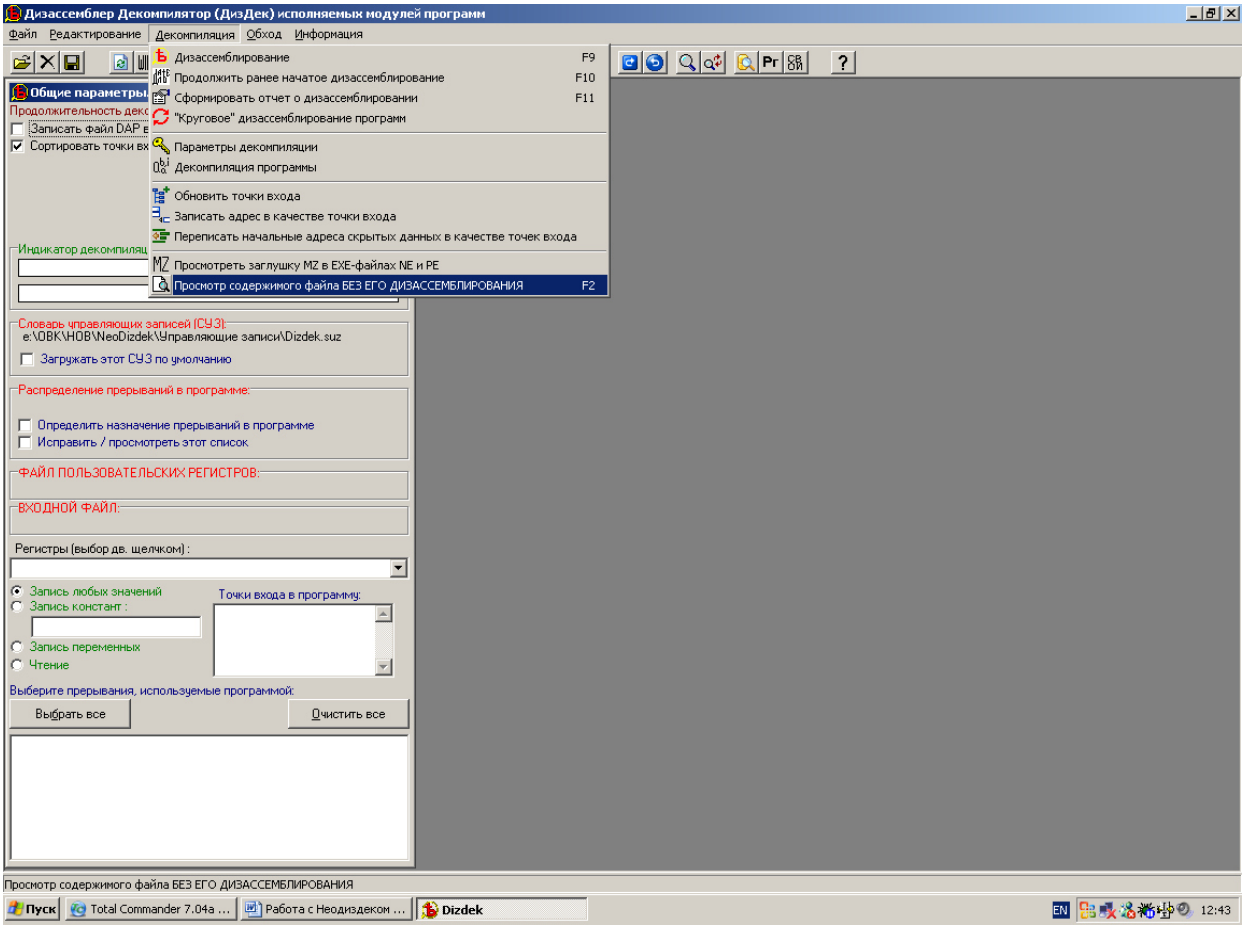


Рисунок.4 - Подменю «Декомпиляция».

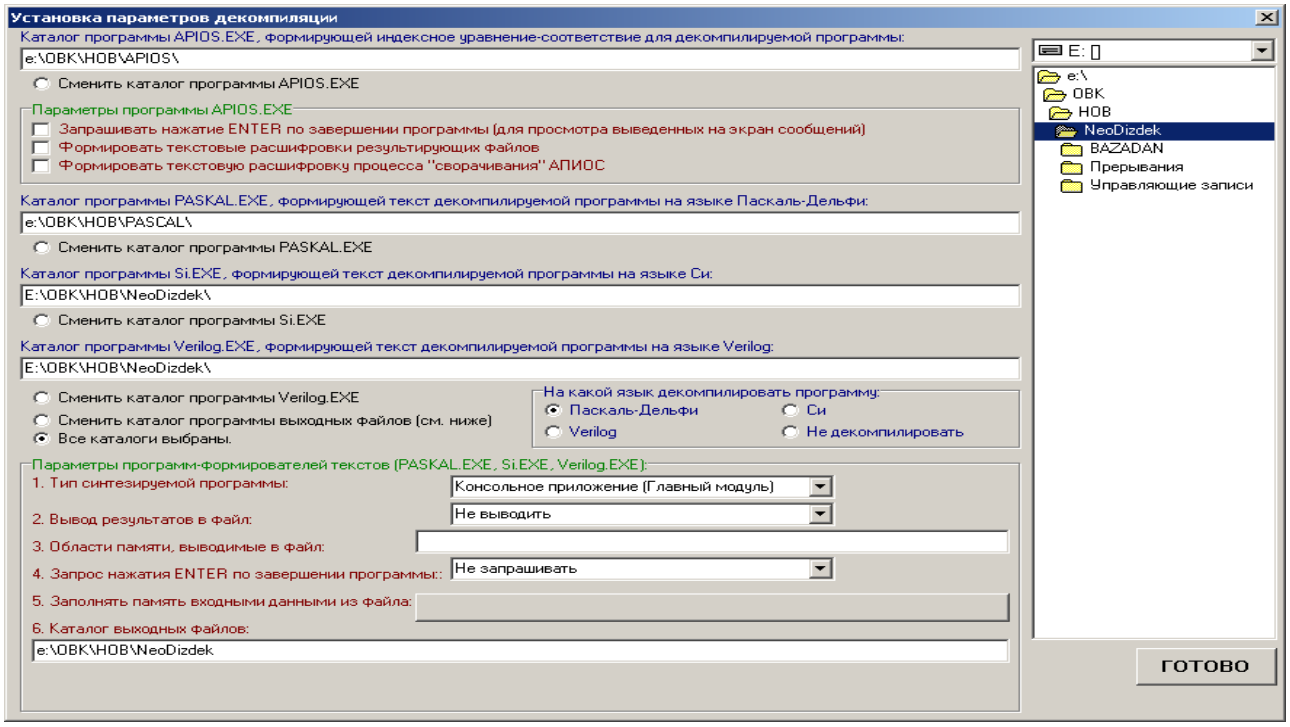


Рисунок.5 - Параметры декомпиляции.

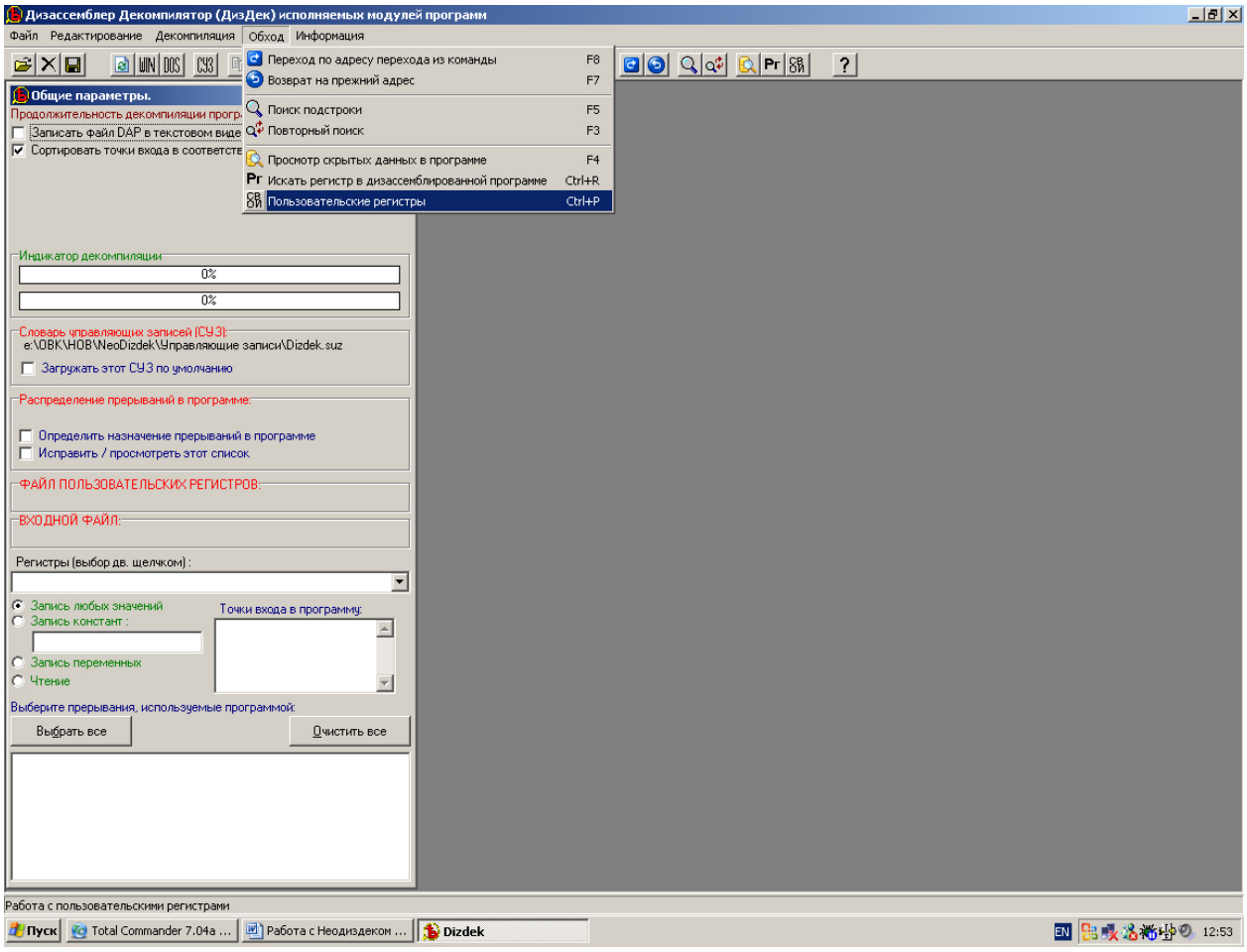


Рисунок.6 - Подменю «Обход».

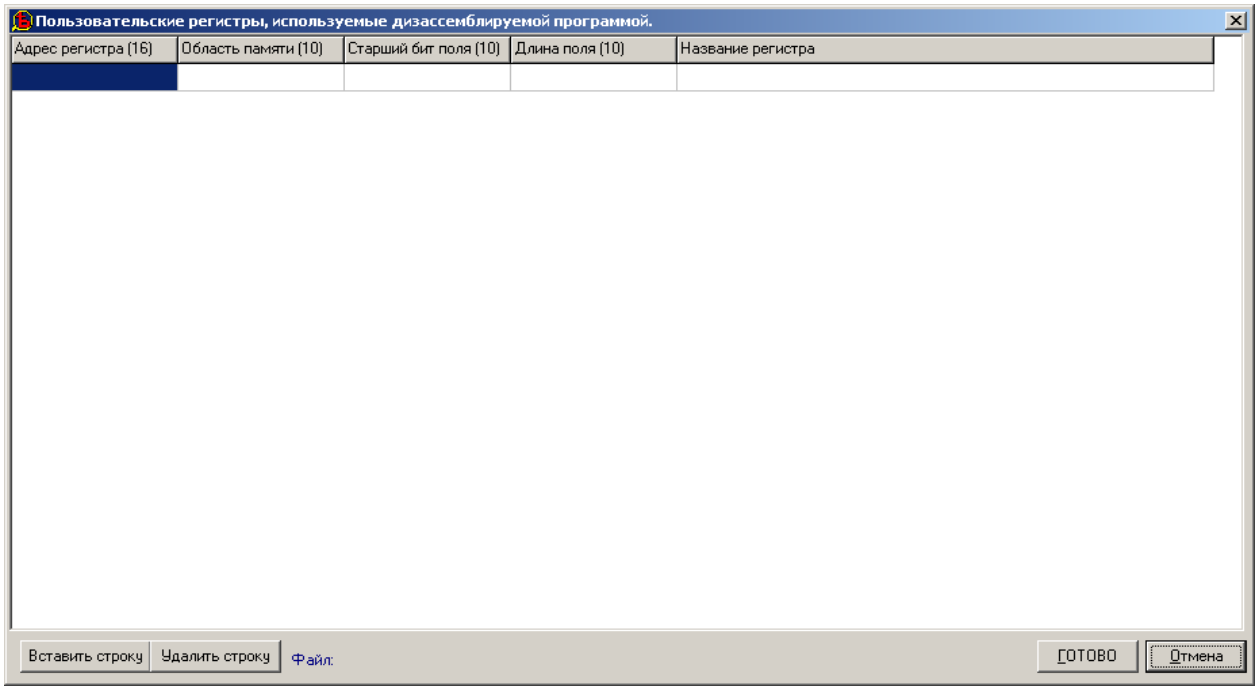


Рисунок.7 - Окно для формирования списка пользовательских регистров.

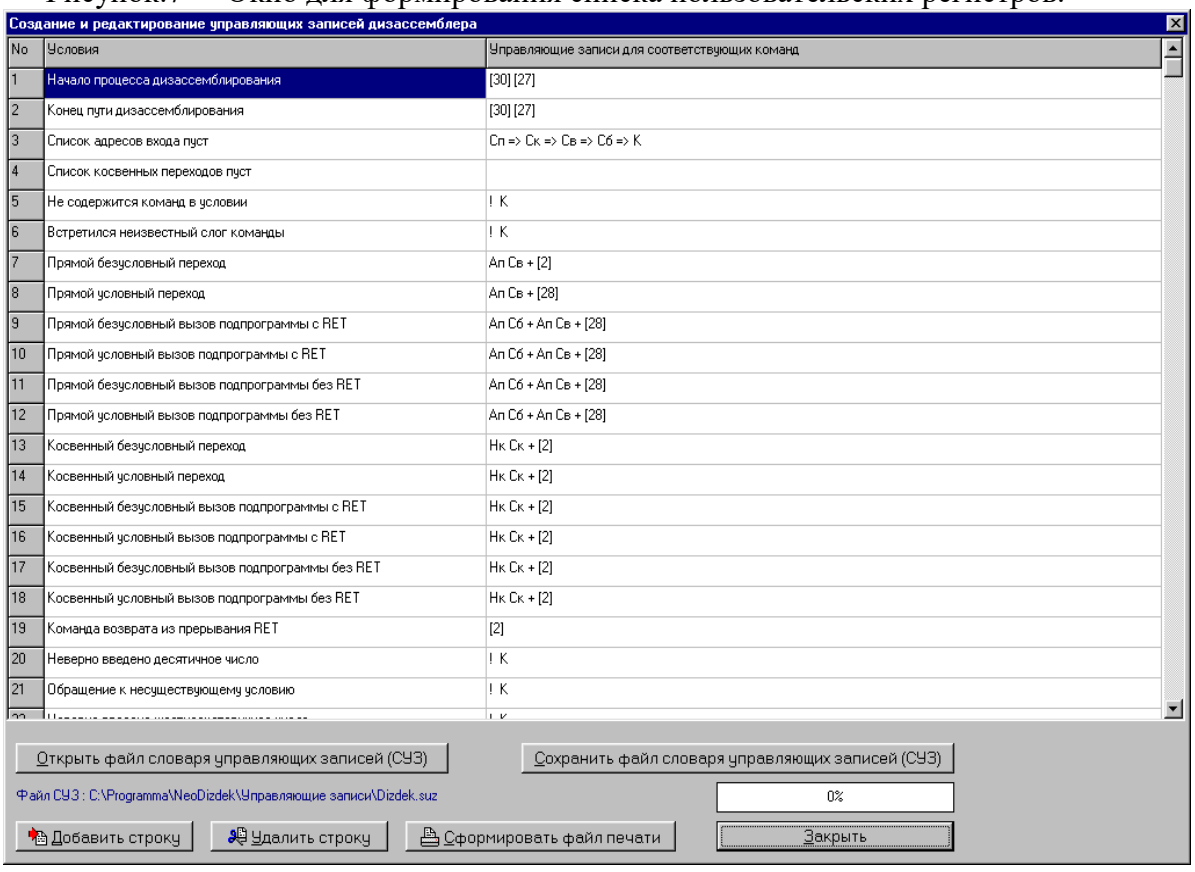


Рисунок.8 - Окно редактирования СУЗ программы Диздек.

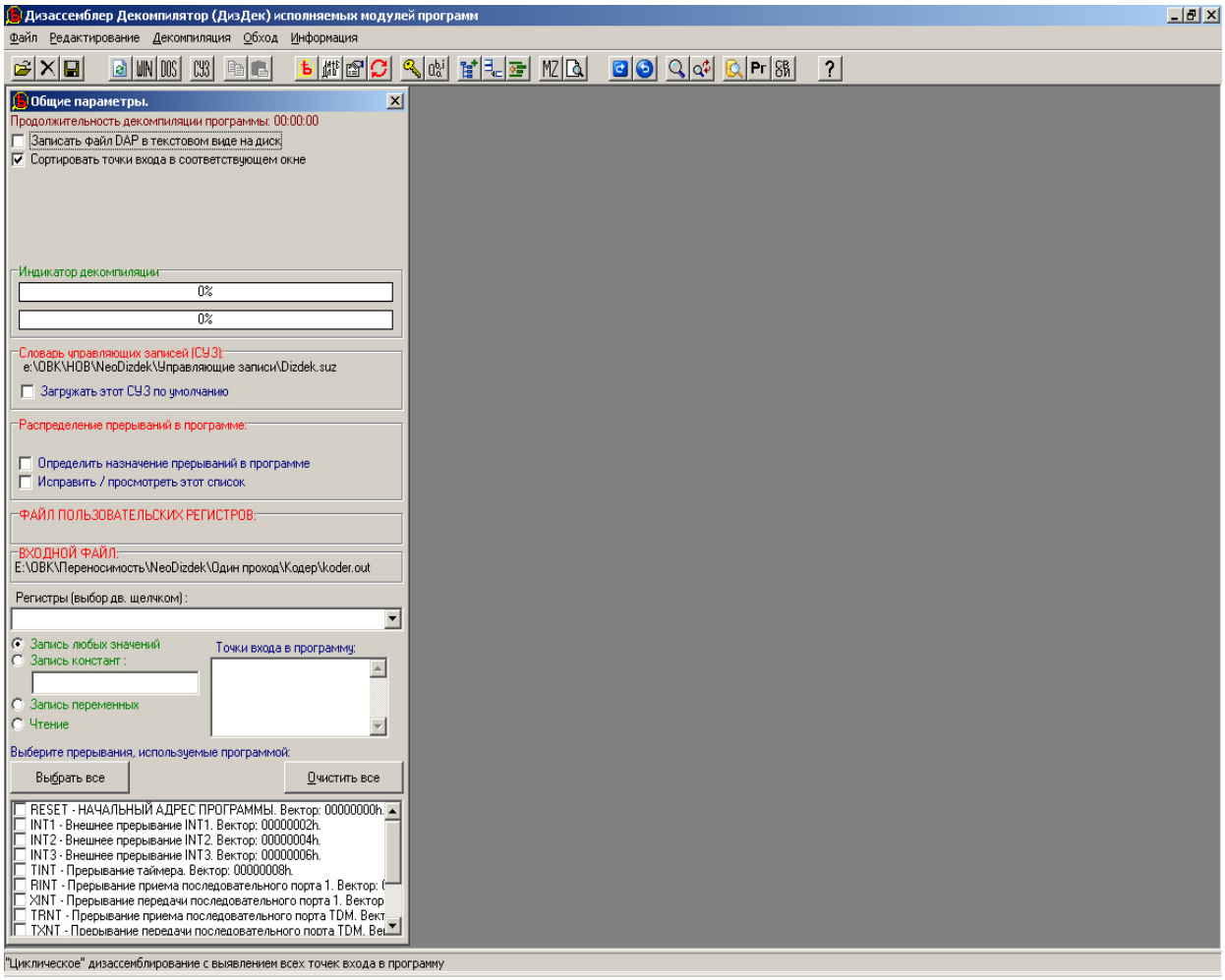


Рисунок.9 - Основное окно Диздека с выбранным входным файлом и прерываниями.

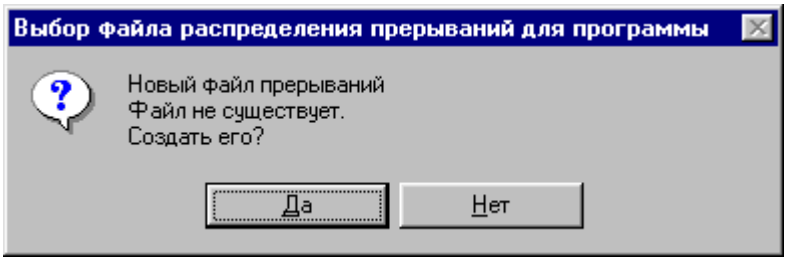


Рисунок.10 - . Окно, предлагающее создать файл с именем «Новый файл прерываний».

По нажатии кнопки «**Да**» в программе сохранится имя вновь введенного файла и появится окно редактирования назначений прерываний, представленное на рис. 11.

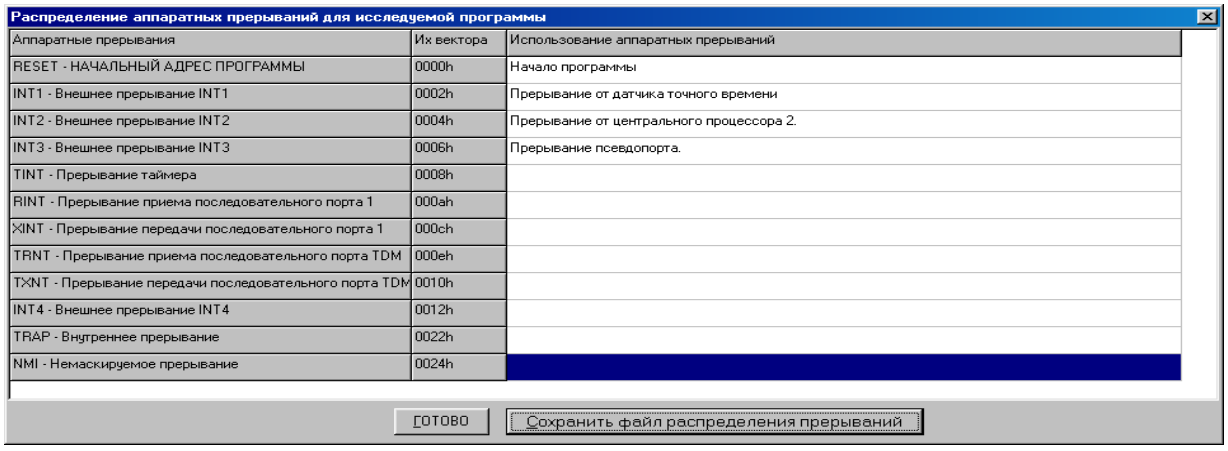


Рисунок.11 - Окно редактирования назначений аппаратных прерываний.

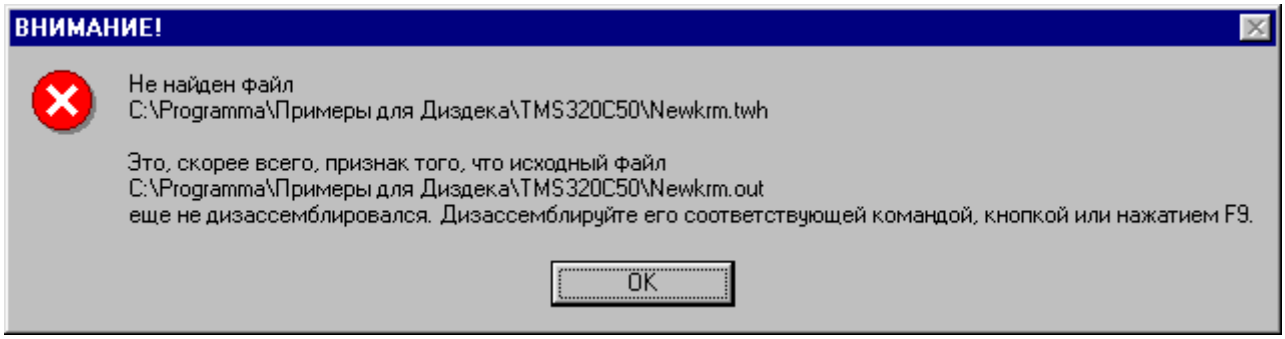


Рисунок.12 - Окно, предупреждающее о невозможности продолжения дизассемблирования.

После загрузки ранее дизассемблированного файла можно продолжить его дизассемблирование обычным порядком. На рис. 13 представлено окно дизассемблера по окончании дизассемблирования.

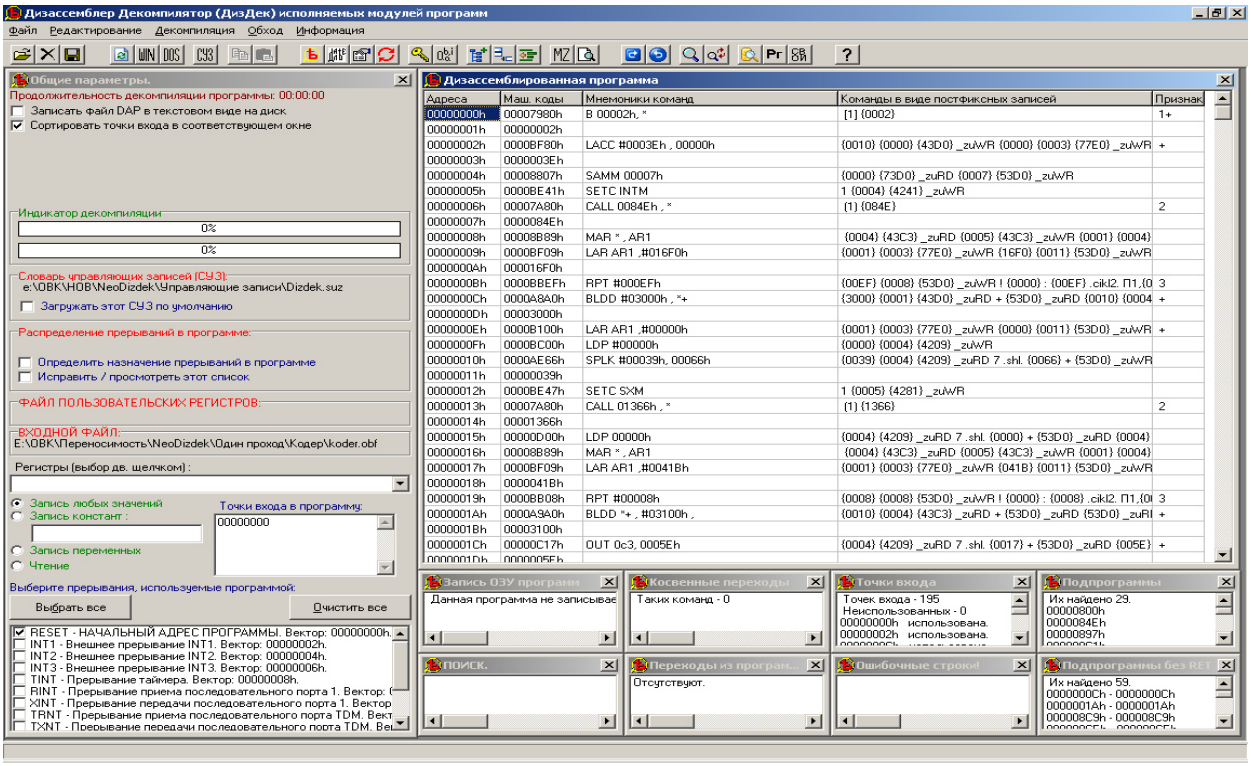


Рисунок.13 - Вид программы по окончании процесса дизассемблирования.

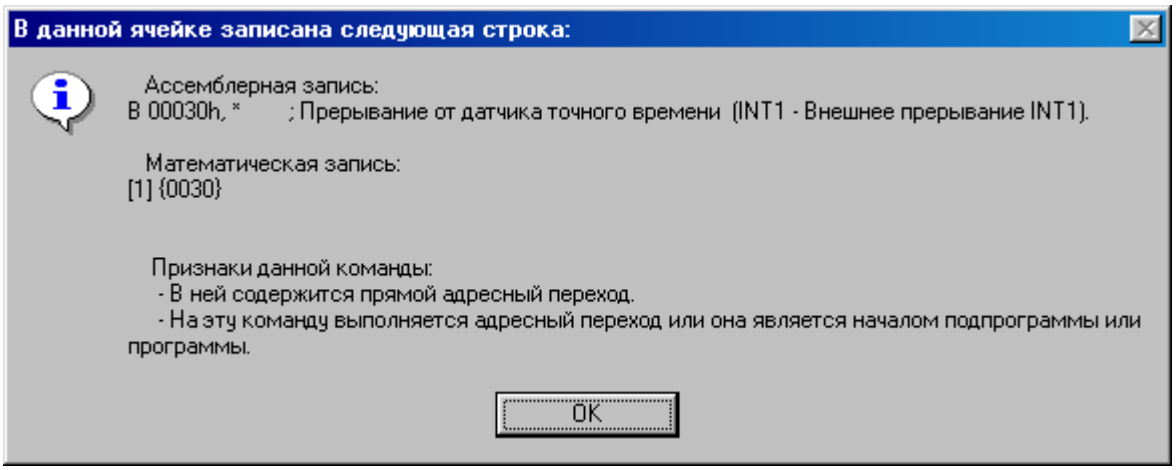


Рисунок.14 - Вид длинных описаний команды в отдельном окне.

7.Охрана труда

Появление современных компьютеров произвело техническую революцию во всех сфера жизни человека. Но не смотря на огромный вклад, внесённый данными устройствами, нельзя отрицать и негативных сторон данного технического прогресса. Главный недостаток современных компьютеров заключается в их негативном влиянии на здоровье человека. Для минимизации негативного влияния компьютера были разработаны требования по безопасности, при работе с ПВЭМ.

В состав данных требований входят следующие пункты:

- ПЭВМ должны соответствовать требованиям настоящих санитарных правил, и каждый их тип подлежит санитарно-эпидемиологической экспертизе с оценкой в испытательных лабораториях, аккредитованных в установленном порядке;
- допустимые уровни звукового давления и уровней звука, создаваемых ПЭВМ, не должны превышать установленных норм;
- временные допустимые уровни электромагнитных полей, создаваемых ПЭВМ, не должны превышать установленных норм;
- концентрации вредных веществ, выделяемых ПЭВМ в воздух помещений, не должны превышать предельно допустимых концентраций, установленных для атмосферного воздуха;
- мощность экспозиционной дозы мягкого рентгеновского излучения в любой точке на расстоянии 0,05 м от экрана и корпуса ВДТ (на электронно-лучевой трубке) при любых положениях регулировочных устройств не должна превышать 1 мкЗв/час (100 мкР/час);

- документация на проектирование, изготовление и эксплуатацию ПЭВМ не должна противоречить требованиям настоящих Санитарных правил.

Также необходимо чтобы используемое ПЭВМ, поддерживало возможность поворота корпуса и его фиксации в установленном положении. Расцветка ПЭВМ должна состоять из спокойных цветов, не отвлекающих внимание.

При организации условий труда также следует учитывать, требования к освещённости рабочего места сотрудника:

- помещения, предназначенные для размещения рабочих мест пользователей персональных компьютеров, должны иметь естественное и искусственное освещение;
- естественное освещение помещений, предназначенных для размещения рабочих мест, должно осуществляться через световые проемы, ориентированные преимущественно на север и северо-восток;
- в случае иной ориентации световых проемов, необходимо предусматривать эффективные средства регулирования интенсивности естественного освещения;
- оконные проемы в помещениях, предназначенных для использования компьютеров, необходимо оборудовать устройствами регулирования интенсивности естественного освещения типа жалюзи, занавесей, внешних козырьков и др.

Уровень шума в рабочем помещении, также регулируется санитарными нормами. Зачастую главным источником шума в рабочем помещение является офисное оборудование. Для снижения подобного шума необходимо предусмотреть специализированные амортизирующие прокладки,

предусмотренные нормативными документами. Уровень шума на рабочих местах не должен превышать 50 дБА.

Соблюдение описанных ранее мер является обязательным требованием для любого рабочего места, так как оно позволяет снизить уровень негативного влияния на здоровье человека при работе с ПЭВМ.

Заключение

В данной работе было разработано выполняющее, дизассемблирование и декомпиляцию. Приложение имеет интуитивно понятный интерфейс, в котором пользователь может выбирать необходимые ему процедуры. После чего можно наблюдать за ходом выполнения работы.

Приложение разрабатывалась с использованием объектно ориентированного языка программирования Delphi. Язык удовлетворяет требованию строгой статистической типизации переменных, что повышает наглядность и читаемость кода. Также средства языка позволили относительно просто анализировать заданные алгоритмы.

В дальнейшем приложение можно улучшить, добавив новую функцию, позволяющую выполнить дизассемблирование и декомпиляцию динамического кода.

В ходе выполнения работы, были получены навыки о приложения. Улучшены навыки разработки в среде Delphi 2007. Получен дополнительный опыт и расширены знания по языку Delphi.

Список используемых источников

1. Владислав Пирогов. Ассемблер и дизасемблирование. — СПб.: «БХВ-Петербург», 2006.

2. В.И. Юров. ASSEMBLER, 2-е издание. — СПб.: «Питер», 2003.

3. М.Ю.Гусенко. Вопросы обратной трансляции программного обеспечения специального назначения — Информационные технологии, №4, 2002.

4. М.Ю.Гусенко. декомпиляция типов данных исполняемых программ —

Приложение А. Листинг программы на Delphi

```
unit MAIN;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls,
Menus,
  StdCtrls, Dialogs, Buttons, Messages, ExtCtrls, ComCtrls,
StdActns,
  ActnList, ToolWin, ImgList, struktura, Grids, Clipbrd;

type
  TMainForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    FileOpenItem: TMenuItem;
    FileCloseItem: TMenuItem;
    Help1: TMenuItem;
    N1: TMenuItem;
    FileExitItem: TMenuItem;
    HelpAboutItem: TMenuItem;
    FileSaveItem: TMenuItem;
    FileSaveAsItem: TMenuItem;
    Edit1: TMenuItem;
    CopyItem: TMenuItem;
    PasteItem: TMenuItem;
    StatusBar: TStatusBar;
    ActionList1: TActionList;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    FileNew1: TAction;
    FileSave1: TAction;
    FileExit1: TAction;
    FileOpen1: TAction;
    FileSaveAs1: TAction;
    WindowCascade1: TWindowCascade;
    WindowTileHorizontal1: TWindowTileHorizontal;
    WindowArrangeAll1: TWindowArrange;
```

```
WindowMinimizeAll1: TWindowMinimizeAll;
HelpAbout1: TAction;
FileClose1: TWindowClose;
WindowTileVertical1: TWindowTileVertical;
ToolBar2: TToolBar;
ToolButton1: TToolButton;
ToolButton2: TToolButton;
ToolButton3: TToolButton;
ToolButton5: TToolButton;
ToolButton6: TToolButton;
ToolButton7: TToolButton;
ImageList1: TImageList;
ToolButton8: TToolButton;
N2: TMenuItem;
N3: TMenuItem;
MZNEPE1: TMenuItem;
N4: TMenuItem;
ToolButton9: TToolButton;
ToolButton10: TToolButton;
ToolButton11: TToolButton;
ToolButton12: TToolButton;
ToolButton13: TToolButton;
ToolButton14: TToolButton;
DOS: TToolButton;
ToolButton15: TToolButton;
WindowsDOS1: TMenuItem;
N5: TMenuItem;
DOSWindows1: TMenuItem;
ToolButton16: TToolButton;
N6: TMenuItem;
N7: TMenuItem;
N8: TMenuItem;
N9: TMenuItem;
ToolButton17: TToolButton;
ToolButton18: TToolButton;
N11: TMenuItem;
ToolButton19: TToolButton;
N12: TMenuItem;
ToolButton20: TToolButton;
N13: TMenuItem;
N14: TMenuItem;
```

```
N10: TMenuItem;
ToolButton4: TToolButton;
ToolButton21: TToolButton;
N15: TMenuItem;
ToolButton22: TToolButton;
N16: TMenuItem;
ToolButton23: TToolButton;
ToolButton25: TToolButton;
ToolButton26: TToolButton;
N17: TMenuItem;
ToolButton24: TToolButton;
N18: TMenuItem;
ToolButton27: TToolButton;
N19: TMenuItem;
N20: TMenuItem;
ToolButton28: TToolButton;
ToolButton29: TToolButton;
N21: TMenuItem;
ToolButton30: TToolButton;
N22: TMenuItem;
N23: TMenuItem;
ToolButton31: TToolButton;
ToolButton32: TToolButton;
KrugDiz: TToolButton;
KRGDiz: TMenuItem;
N24: TMenuItem;
Apios: TMenuItem;
ToolButton33: TToolButton;
ToolButton34: TToolButton;
N25: TMenuItem;
ToolButton35: TToolButton;
N26: TMenuItem;
ToolButton36: TToolButton;
N27: TMenuItem;
N28: TMenuItem;
N29: TMenuItem;
ToolButton37: TToolButton;
procedure FileOpen1Execute(Sender: TObject);
procedure HelpAbout1Execute(Sender: TObject);
procedure FileExit1Execute(Sender: TObject);
procedure Nachalo(Sender: TObject);
```

```
        procedure FormResize(Sender: TObject);
        procedure N3Click(Sender: TObject);
        procedure N4Click(Sender: TObject);
        procedure MZNEPE1Click(Sender: TObject);
        procedure UMobf(var Msg: TMessage);
        message UM_obf;
        procedure UMMain(var Msg: TMessage);
        message UM_Main;
        procedure DOSClick(Sender: TObject);
        procedure ToolButton2Click(Sender: TObject);
        procedure ToolButton15Click(Sender: TObject);
        procedure ToolButton16Click(Sender: TObject);
        procedure N9Click(Sender: TObject);
        procedure FileCloseItemClick(Sender: TObject);
        procedure N11Click(Sender: TObject);
        procedure N12Click(Sender: TObject);
        procedure N10Click(Sender: TObject);
        procedure Ret(Sender: TObject);
        procedure N16Click(Sender: TObject);
        procedure N17Click(Sender: TObject);
        procedure N18Click(Sender: TObject);
        procedure N20Click(Sender: TObject);
        procedure N21Click(Sender: TObject);
        procedure N23Click(Sender: TObject);
        procedure KrugDizClick(Sender: TObject);
        procedure KRGDizClick(Sender: TObject);
        procedure ApiosClick(Sender: TObject);
        procedure N25Click(Sender: TObject);
        procedure N26Click(Sender: TObject);
        procedure N28Click(Sender: TObject);
        procedure N29Click(Sender: TObject);
      private
        { Private declarations }
        procedure CreateMDIChild(const Name: string);
        procedure   CreateTSP(const   Name,FN:   string;   const
X,Y,SHir,Vis: integer);
        procedure   CreateGSP(const   Name,FN:   string;   const
X,Y,SHir,Vis: integer);
        procedure CreateMZF;
        procedure CreatePEF;
        procedure CreateNEF;
        procedure CreateOBF(const FN:String);
```

```
      procedure CreateOUTosf;
      procedure CreateTDPRER;
      procedure vivodDIZ;
      procedure VydeliDannye;
      procedure ZapRegis;
      procedure ZapRegisPol;
    public
      { Public declarations }
    end;

var
  MainForm: TMainForm;


implementation

{$R *.dfm}

uses CHILDWIN, about, Unit1, outTOosf, SUZRed, MZFail, PEFail,
NEFail, RPR,
     SLUZHfun, FailOBF, Dizassembler, Spisok, GlavSpisok,
OKCANCL1, SRGP,
  Pardek;

procedure TMainForm.VydeliDannye;
var f,fosp:file of byte;
    a,b,c,d,opersis:byte;
    zag1,zag2:longword;
    fa,a1,a2:longint;
    j,j1:longint;
    fpr:file of promfile;
    zpr:promfile;
    mc:masblok;
begin
 assignfile(f,ChangeFileExt(failv,'.obf'));
 reset(f);
 assignfile(fosp,ChangeFileExt(failv,'.osp'));
 rewrite(fosp);
 read(f,a,b,c,d,opersis);
 tipslova:=b and $03;
 write(fosp,a,b,c,d,opersis);
 read(f,a,b,c,d);
```

```
      write(fosp,a,b,c,d);
      closefile(f);
      assignfile(fpr,ChangeFileExt(failv,'.dap'));
      reset(fpr);
      dlin:=filesize(fpr);
      dlt:=0;
      Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,2,55);

      for j:=1 to ub do begin
       mc:=mbb[j];
       a1:=mc.sec2;
       a2:=mc.sec4;
       seek(fpr,mc.fp);
       zag2:=0;
       for j1:=a1 to a2 do begin
        read(fpr,zpr);
        if zpr.pd=0 then begin
         if zag2=0 then begin
          d:=0;
          zag1:=j1;
          fa:=filepos(fosp);
          write(fosp,d,d,d,d,d,d,d,d);
         end;
         zag2:=zag2+1;
         case tipslova of
          0:write(fosp,zpr.k4);
          1:write(fosp,zpr.k3,zpr.k4);
          2:write(fosp,zpr.k1,zpr.k2,zpr.k3,zpr.k4);
         end;
        end else if zag2>0 then begin
         seek(fosp,fa);
         LongWordToByte(zag1,a,b,c,d);
         write(fosp,a,b,c,d);
         LongWordToByte(zag2-1,a,b,c,d);
         write(fosp,a,b,c,d);
         seek(fosp,filesize(fosp));
         zag2:=0;
        end;
        dlt:=dlt+1;
        Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,1,(100*dlt) div dlin);
```

```
      end;
      if zag2>0 then begin
       seek(fosp,fa);
       LongWordToByte(zag1,a,b,c,d);
       write(fosp,a,b,c,d);
       LongWordToByte(zag2-1,a,b,c,d);
       write(fosp,a,b,c,d);
       seek(fosp,filesize(fosp));
       zag2:=0;
      end;
     end;
     fa:=filesize(fosp);
     closefile(fosp);
     if fa<12 then erase(fosp);
     closefile(fpr);
    end;

    procedure TMainForm.UMMain(var Msg: TMessage);
    var f2:file of zaptvh;
        f4:file of bezret;
        fa:longword;
        szp,s2,s,stv,skp,sbp,s3:string;
        j,j1,i,z,a,b1:longint;
        kpr:kosreg;
        adrs,kl:longword;
        b:bezret;
        f3:file of kosreg;
    begin
     s2:='            ';
     case Msg.WParam of
      1:begin
         szp:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись    ОЗУ
программ')]).FileName;
         if fileexists(szp) then begin
          assignfile(f2,szp);
          reset(f2);
          fa:=filesize(f2)-1;
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись        ОЗУ
программ')]).ListBox1.Items.Add('      Таких      команд      -
'+IntToStr(fa+1));
          closefile(f2);
          for j:=0 to fa do begin
```

```
           s2:=vibTVH(szp,j);
           TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись       ОЗУ
программ')]).ListBox1.Items.Add(copy(s2,1,pos('h',s2)));
           dlt:=dlt+5;
          end;
          end else begin
           TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись       ОЗУ
программ')]).ListBox1.Items.Add(' Данная программа не записывает
память программ!');
          end;
        end;
      2:begin
         skp:=TTSP(Screen.Forms[OknoPoImeni(Screen,    'Косвенные
переходы')]).FileName;
         if fileexists(skp) then begin
          assignfile(f3,skp);
          reset(f3);
          fa:=filesize(f3)-1;
          closefile(f3);
          s:='Пр-к : П.Р.К        Адрес команды : Точка входа, на
которой она обнаружена       Регистр (адрес:модификатор (Название
регистра))';
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Косвенные
переходы')]).ListBox1.Items.Add('      Таких      команд      -
'+IntToStr(fa+1));
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Косвенные
переходы')]).ListBox1.Items.Add(s);
         end else TTSP(Screen.Forms[OknoPoImeni(Screen,'Косвенные
переходы')]).ListBox1.Items.Add(' Таких команд - 0');
         if fileexists(skp) then for j:=0 to fa do begin
          chitKP(skp,j,j1,kpr);
          s:=IntToHex(kpr.pr,2)+'h: ';
          if (kpr.pr and $04)<>0 then s:=s+'П.' else s:=s+'-.';
          if (kpr.pr and $02)<>0 then s:=s+'P.' else s:=s+'-.';
          if (kpr.pr and $01)<>0 then s:=s+'K' else s:=s+'-';
          s:=s+copy(s2,1,12-
length(s))+IntToHex(ByteToLongword(kpr.adres1,kpr.adres2,kpr.adr
es3,kpr.adres4),8)+'h                                    :
'+IntToHex(ByteToLongword(kpr.t1,kpr.t2,kpr.t3,kpr.t4),8)+'h
'+ImRegistra(TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).ListBox1,ByteToWord(kpr.areg1,kpr.areg2),ByteToWo
rd(kpr.am1,kpr.am2));
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Косвенные
переходы')]).ListBox1.Items.Add(s);
```

```
         end;
        end;
      3:begin
         stv:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).FileName;
         if FileExists(stv) then begin
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Items.Clear;
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Sorted:=srttvh;
          assignfile(f2,stv);
          reset(f2);
          fa:=filesize(f2)-1;
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Items.Add(' Точек входа - '+IntToStr(fa+1));
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Items.Add(' Точек входа - '+IntToStr(fa+1));
          closefile(f2);
          i:=0;
          b1:=0;
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Переходы       из
программы!')]).ListBox1.Items.Clear;
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Переходы       из
программы!')]).ListBox1.Items.Add('Отсутствуют.');
          for j:=0 to fa do begin
           s2:=vibTVH(stv,j);
           if pos('НЕ ИСПОЛЬЗ',s2)<>0 then i:=i+1;
           s3:=copy(s2,1,pos('h',s2)-1);
           z:=heks(s3,16,a);
           a:=perehod(z,0);
           if kons.usl=26 then begin
            b1:=b1+1;
            TTSP(Screen.Forms[OknoPoImeni(Screen,'Переходы     из
программы!')]).ListBox1.Items.Add(s3+'h');
            kons.usl:=0;
           end;
           TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Items.Add(copy(s2,1,pos('h',s2))+'
'+copy(s2,pos(':',s2)+1,255));
          end;
          if                        b1>0                        then
TTSP(Screen.Forms[OknoPoImeni(Screen,'Переходы                  из
программы!')]).ListBox1.Items.Strings[0]:='Таких    переходов    -
'+IntToStr(b1);
```

```
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Items.Strings[1]:='     Неиспользованных     -
'+IntToStr(i);
         end;
        end;
      5:begin

sbp:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).File
Name;
         if FileExists(sbp) then begin
          assignfile(f2,sbp);
          reset(f2);
          fa:=filesize(f2)-1;

TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBox1.
Items.Add(' Их найдено '+IntToStr(fa+1)+'.');

TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBox1.
Sorted:=true;
          closefile(f2);
          for j:=0 to fa do begin
           s2:=vibTVH(sbp,j);

TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBox1.
Items.Add(copy(s2,1,pos('h',s2)));
          end;
          erase(f2);
          for j:=1 to fa+1 do begin

s:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBo
x1.Items.Strings[j];
           delete(s,pos('h',s),255);
           adrs:=heks(s,16,i);
           zapisTVH(sbp,adrs);
          end;

TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBox1.
Sorted:=false;
         end                                                 else
TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBox1.
Items.Add(' Отсутствуют.');
        end;
      6:begin
```

```
         sbp:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы
без RET')]).FileName;
         if FileExists(sbp) then begin
          assignfile(f4,sbp);
          reset(f4);
          fa:=filesize(f4)-1;
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы  без
RET')]).ListBox1.Items.Add(' Их найдено '+IntToStr(fa+1)+'.');
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы  без
RET')]).ListBox1.Sorted:=true;
          for j:=0 to fa do begin
           read(f4,b);
           kl:=ByteToLongword(b.d1,b.d2,b.d3,b.d4);
           adrs:=ByteToLongword(b.a1,b.a2,b.a3,b.a4);
           TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы без
RET')]).ListBox1.Items.Add(IntToHex(adrs,8)+'h                    -
'+IntToHex(adrs+kl-1,8)+'h');
          end;
          closefile(f4);
          rewrite(f4);
          for j:=1 to fa+1 do begin
           s:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы
без RET')]).ListBox1.Items.Strings[j];
           s2:=copy(s,1,pos('h',s)-1);
           delete(s,1,pos('h',s)+3);
           delete(s,pos('h',s),255);
           adrs:=heks(s2,16,i);
           kl:=heks(s,16,i)-adrs+1;
           LongwordToByte(adrs,b.a1,b.a2,b.a3,b.a4);
           LongwordToByte(kl,b.d1,b.d2,b.d3,b.d4);
           write(f4,b);
          end;
          closefile(f4);
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы  без
RET')]).ListBox1.Sorted:=false;
         end                                                 else
TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы           без
RET')]).ListBox1.Items.Add(' Отсутствуют.');
        end;

4:TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBox
1.Items.Add('Информационное сообщение: '+mSUZ[1,Msg.LParam]);
     end;
```

```
    end;

    procedure TMainForm.vivodDIZ;
    var i:longword;
        s,s1,s2,sfp:string;
        zpf:promfile;
        fp:file of promfile;
    begin
         TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge2.Progress:=0;
         sfp:=ChangefileExt(failv,'.dap');
{промежуточный файл}
         if FileExists(sfp) then begin
          assignfile(fp,sfp);
          reset(fp);
          dlin:=FileSize(fp);

TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.RowCount:=dlin+1;
          Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,2,60);
          for i:=0 to dlin-1 do begin
            read(fp,zpf);

s1:=IntToHex(ByteToLongword(zpf.a1,zpf.a2,zpf.a3,zpf.a4),8)+'h';
            s2:=s1;

TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.Cells[0,i+1]:=s1;
            s:='                  ';
            if (zpf.p and $80)>0 then s:=s+'7';
            if (zpf.p and $40)>0 then s:=s+'6';
            if (zpf.p and $20)>0 then s:=s+'5';
            if (zpf.p and $10)>0 then s:=s+'4';
            if (zpf.p and $08)>0 then s:=s+'3';
            if (zpf.p and $04)>0 then s:=s+'2';
            if (zpf.p and $02)>0 then s:=s+'1';
            if (zpf.p and $01)>0 then s:=s+'+';

s1:=IntToHex(ByteToLongword(zpf.k1,zpf.k2,zpf.k3,zpf.k4),8)+'h';

TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.Cells[1,i+1]:=s1;
```

```
TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.Cells[4,i+1]:=copy(s,18,255);

TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.Cells[2,i+1]:=zpf.mnem;
            if          StrPos(zpf.mnem,'***')<>nil          then
TTSP(Screen.Forms[OknoPoImeni(Screen,'Ошибочные
строки!')]).ListBox1.Items.Add(s2);

TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.Cells[3,i+1]:=zpf.mat;
            Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,1,(100*i) div (dlin-1));
          end;
          closefile(fp);
          TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
          TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge2.Progress:=0;
         end;
    end;

    procedure TMainForm.UMobf(var Msg: TMessage);
    var s:string;
        i:integer;
    begin
     daiobf:=1;
     case Msg.WParam of
      0:CreateOUTosf;
      1:begin
         s:=TipEXE(failv);
         if s='MZ' then CreateMZF;
         if s='NE' then CreateNEF;
         if s='PE' then CreatePEF;
        end;
      2:begin
         case Msg.LParam of
          0:begin
             for i:=MDIChildCount-1 downto 0 do
              if               (not(MDIChildren[i]               =
Nil))and(not(MDIChildren[i] is TMDIChild)) then
                  MDIChildren[i].Destroy;
```

```
              TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).sfail.Caption:=failv;
            end;
          1:begin
             i:=MDIChildCount;
             repeat
              if MDIChildren[i] is TOUTosf then begin
               MDIChildren[i].Destroy;
               i:=-1;
              end else i:=i-1;
             until i=-1;
            end;
         end;
        end;
      3:CreateTDPRER;
     end;
     //daiobf:=0;
    end;

    procedure TMainForm.CreateMDIChild(const Name: string);
    var
      Child: TMDIChild;
    begin
      Child := TMDIChild.Create(Application);
      Child.Caption := Name;
    end;

    procedure  TMainForm.CreateTSP(const  Name,FN:  string;  const
X,Y,SHir,Vis: integer);
    var Child: TTSP;
        s:string;
        f4:textfile;
        j:integer;
        dlt,dlin:longword;
    begin
      TSPotkr:=true;
      Child := TTSP.Create(Application);
      Child.FileName:=FN;
      Child.Caption := Name;
      if Name='Точки входа' then Child.ListBox1.Sorted:=true
                            else Child.ListBox1.Sorted:=false;
      Child.Top:=Y;
```

```
      Child.Left:=X;
      Child.Verh:=Y;
      Child.Levo:=X;
      Child.Width:=SHir;
      Child.Height:=Vis;
      Child.ListBox1.Width:=Child.Width-10;
      Child.ListBox1.Height:=Child.Height-30;
      Child.Constraints.MaxWidth:=Child.Width;
      Child.Constraints.MaxHeight:=Child.Height;
      if Name='Ассемблерный текст программы' then begin
       listing:=0;
       dlin:=SizeByte(FN);
       assignfile(f4,FN);
       reset(f4);
       dlt:=0;
       repeat
        readln(f4,s);
        dlt:=dlt+length(s);
        j:=pos(#9,s);
        if j<>0 then repeat
         insert('        ',s,j);
         delete(s,pos(#9,s),1);
         j:=pos(#9,s);
        until j=0;
        Child.ListBox1.Items.Add(s);
        Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,1,(100*dlt) div dlin);
       until eof(f4);
       closefile(f4);
       Child.ListBox1.ItemIndex:=listing;
      end;
    end;

    procedure TMainForm.CreateGSP(const Name,FN: string; const
X,Y,SHir,Vis: integer);
    var
      Child: TGSP;
    begin
      Child := TGSP.Create(Application);
      Child.Top:=Y;
      Child.Left:=X;
      Child.Caption := Name;
```

```
  Child.Verh:=Y;
  Child.Levo:=X;
  Child.Width:=SHir;
  Child.Height:=Vis;
  Child.FileName:=FN;
  Child.Shirina:=SHir;
  Child.StringGrid1.Width:=Child.Width-10;
  Child.StringGrid1.Height:=Child.Height-30;
  Child.Constraints.MaxWidth:=Child.Width;
  Child.Constraints.MaxHeight:=Child.Height;
end;

procedure TMainForm.CreatePEF;
begin
  TPEF.Create(Application);
end;

procedure TMainForm.CreateTDPRER;
begin
  TDPRER.Create(Application);
end;

procedure TMainForm.CreateOBF(const FN:string);
begin
  imobf:=FN;
  TOBFF.Create(Application);
end;

procedure TMainForm.CreateOUTosf;
begin
  TOUTosf.Create(Application);
end;

procedure TMainForm.CreateNEF;
begin
  TNEF.Create(Application);
end;

procedure TMainForm.CreateMZF;
begin
  TMZF.Create(Application);
```

```
    end;

    procedure TMainForm.FileOpen1Execute(Sender: TObject);
    begin
     FileCloseItemClick(Sender);
     zptvh:=false;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Clear;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).failClick(Sender);
     ZapRegis;
    end;

    procedure TMainForm.HelpAbout1Execute(Sender: TObject);
    begin
      AboutBox.ShowModal;
    end;

    procedure TMainForm.FileExit1Execute(Sender: TObject);
    begin
      Close;
    end;

    procedure TMainForm.Nachalo(Sender: TObject);
    var s:string;
    begin
     ischipoisk:=false;
     APIOSpar:='';
     failRGP:='';
     RGSTZap.nomer:=-1;
     RGSTZap.stroka:='';
     RGSTZap.WR:=true;
     RGSTZap.RD:=false;
     krugom:=false;
     koneckruga:=false;
     krugpervij:=true;
     KrugUk:=0;
     KrugUTVH:=0;
     ProgDekom:='';
     ischistroku:='';
     listing:=0;
     ischibit:=false;
```

```
 TSPotkr:=false;
 zakokno:=false;
 bobf:=false;
 failv:='';
 s:=paramstr(1);
 katprog:=ExtractFilePath(ExpandFileName('Dizdek.exe'));
 katAPIOS:=katprog;
 katPas:=katprog;
 katSi:=katprog;
 katVerilog:=katprog;
 katVyh:=katprog;
 delete(katVyh,length(katVyh),1);
 ChDir(katprog);
 katsbd:=katprog+'BAZADAN\';
 failproc:=katprog+'PROCESOR.KOD';
 failv:='';
 vsegoform:=Screen.FormCount;
 zptvh:=false;
 srttvh:=true;
 if FileExists(s) then failv:=s;
end;

procedure TMainForm.FormResize(Sender: TObject);
var k:integer;
begin
 PX:=MainForm.ClientWidth-5;
 PY:=MainForm.ClientHeight;
 k:=OknoPoImeni(Screen,'Общие параметры.');
 if k<0 then CreateMDIChild('Общие параметры.');
end;

procedure TMainForm.N3Click(Sender: TObject);
var f3:file of zaptvh;
    ztvh:zaptvh;
    a,c:byte;
    i,j:longint;
    s,stv,s1,skp,szp,sbp:string;
    f2:file of zprer;
    z:zprer;
    f4:textfile;
    x,y,ya1,ya2,sa:integer;
```

```
        b:file of byte;
        k,fprr:boolean;
        tv:longword;
        t1,t2:TDateTime;
    begin
     if   pos('.obf',lowercase(ExtractFileName(failv)))<>0   then
AdresaOBF(failv);
     KrugUTVH:=0;
     vysota:=3*(TSH div 4)-3;
     ya1:=3*(TSH div 4);
     sa:=(TSH div 4) div 2;
     ya2:=ya1+sa;
     ChDir(katprog);
     ucetadresa:=true;
     fprr:=false;
     s:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).sprer.Caption;
     fprer:=ChangeFileExt(failv,'.spr');
     if (fprer<>s)and(s<>'') then begin
      fprer:=s;
      fprr:=true;
     end else if FileExists(fprer) then DeleteFile(fprer);
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Clear;
     if zptvh then begin
      assignfile(f3,ChangeFileExt(failv,'.twh'));
      reset(f3);
      repeat
       read(f3,ztvh);
       tv:=ByteToLongword(ztvh.a1,ztvh.a2,ztvh.a3,ztvh.a4);
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Add(IntToHex(tv,8));
      until eof(f3);
      closefile(f3);
      reset(f3);
      i:=0;
      j:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Count;
      repeat
       read(f3,ztvh);
       tv:=ByteToLongword(ztvh.a1,ztvh.a2,ztvh.a3,ztvh.a4);
       s:='. Вектор: '+IntToHex(tv,8)+'h';
```

```
       s1:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Strings[i];
       if                    pos(s,s1)<>0                    then
TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Checked[i]:=true;
       i:=i+1;
      until (i=j)or(eof(f3));
      predprer:=tv;
      closefile(f3);
     end;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge2.Progress:=0;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
     s:=ChangeFileExt(failv,'.dap');
     if FileExists(s) then DeleteFile(s);
     s:=ChangeFileExt(failv,'.das');
     if FileExists(s) then DeleteFile(s);
      if not zptvh then begin
       stv:=ChangeFileExt(failv,'.tvh');
       DeleteFile(stv);
       DeleteFile(ChangeFileExt(failv,'.twh'));
      end else stv:=ChangeFileExt(failv,'.tvh');
     s1:=ChangeFileExt(failv,'.lst');
     if not FileExists(s1) then s1:=ChangeFileExt(failv,'.asm');
     if FileExists(s1) then begin
      vysota:=2*(TSH div 4)-3;
      ya1:=vysota+3;
      ya2:=ya1+sa;
      CreateTSP('Ассемблерный    текст    программы',s1,(PX    div
3),3*(TSH div 4),2*(PX div 3)-2,(TSH div 4));
     end;

     daiobf:=1;
     daimz:=0;
     bobf:=false;
     if mSUZ[1,1]='1' then begin
      MessageBox(0,'Не выбран файл словаря управляющих записей,
необходимый                    для                    работы
программы.','ВНИМАНИЕ!',mb_Ok+mb_iconstop);
      exit;
     end else
     if failv<>'' then begin
```

```
     if krugom then koneckruga:=false
               else koneckruga:=true;
    t1:=Time;
    repeat
     s:=ChangeFileExt(failv,'.kp');
     if FileExists(s) then DeleteFile(s);
     s:=ChangeFileExt(failv,'.bp');
     if FileExists(s) then DeleteFile(s);
     s:=ChangeFileExt(failv,'.brt');
     if FileExists(s) then DeleteFile(s);
     s:=ChangeFileExt(failv,'.ret');
     if FileExists(s) then DeleteFile(s);
     s:=ChangeFileExt(failv,'.osp');
     if FileExists(s) then DeleteFile(s);
     s:=ChangeFileExt(failv,'.zp');
     if FileExists(s) then DeleteFile(s);
      for i:=MDIChildCount-1 downto 0 do
              if               (not(MDIChildren[i]               =
Nil))and(not(MDIChildren[i] is TMDIChild)) then
                   if      (MDIChildren[i].Caption<>'Ассемблерный
текст программы') then MDIChildren[i].Destroy;
      if        (TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Count<>0)and(not fprr) then begin
       kolp:=0;
       for                          i:=0                          to
TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Count-1 do
                if
TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Checked[i] then kolp:=kolp+1;
       if (not FileExists(fprer))and(kolp>0) then begin
        assignfile(f2,fprer);
        rewrite(f2);
       end;
       for                          i:=0                          to
TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Count-1 do begin
        if       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Checked[i] then begin
         s1:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Strings[i];
         delete(s1,1,pos('. Вектор: ',s1)+9);
         delete(s1,pos('h',s1),255);
```

```
          a:=heks(s1,16,j);
          if
(FileExists(fprer))and(TMDIChild(Screen.Forms[OknoPoImeni(Screen
,'Общие параметры.')]).sprer.Caption='') then begin
           LongwordToByte(a,z.a1,z.a2,z.a3,z.a4);
           z.p:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Strings[i];
           delete(z.p,pos('. Вектор: ',z.p),255);
           z.s:='';
           reset(f2);
           seek(f2,filesize(f2));
           write(f2,z);
           closefile(f2);
          end;
          zapisTVH(stv,a);
          zapisTVH(ChangeFileExt(failv,'.twh'),a);
         end;
        end;

      end;
      zakokno:=true;
      if pos('.out',lowercase(failv))<>0 then begin
       failv:=ChangeFileExt(failv,'.obf');
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).sfail.Caption:=failv;
      end;
      if
(pos('.exe',lowercase(failv))<>0)or(pos('.dll',lowercase(failv))
<>0) then begin
       s:=TipEXE(failv);
       if s='MZ' then begin
        daimz:=0;
        CreateMZF;
       end;
       if s='NE' then CreateNEF;
       if s='PE' then CreatePEF;
       failv:=ChangeFileExt(failv,'.obf');
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).sfail.Caption:=failv;
      end;
      if pos('.obf',lowercase(failv))<>0 then begin
       assignfile(b,failv);
```

```
        reset(b);
        read(b,a,c);
        closefile(b);
        kodim(ByteToWord(a,c) div 4,failproc,processor);
        kons.obekt:=0;
        kons.perem:=0;
        kons.a1:=0;
        kons.a2:=0;
        kons.a3:=0;
        kons.a4:=0;
        kons.a5:=0;
        kons.a6:=0;
        kons.NTV:=0;
        kons.NKPa:=0;
        kons.NKPm:=0;
        kons.usl:=0;

{================================================================
======}
         s:=ChangeFileExt(failv,'.dap');
         skp:=ChangeFileExt(failv,'.kp');
         szp:=ChangeFileExt(failv,'.zp');
         sbp:=ChangeFileExt(failv,'.bp');
         stv:=ChangeFileExt(failv,'.tvh');
         CreateTSP('Запись ОЗУ программ',szp,(PX div 3),ya1,(PX div
6),sa);
         CreateTSP('ПОИСК.','',(PX div 3),ya2,(PX div 6),sa);
         CreateTSP('Косвенные переходы',skp,(PX div 3)+(PX div
6),ya1,(PX div 6),sa);
         CreateTSP('Переходы из программы!',skp,(PX div 3)+2*(PX
div 6),ya1,(PX div 6),sa);
         CreateTSP('Подпрограммы',ChangeFileExt(failv,'.ret'),(PX
div 3)+2*(PX div 6),ya2,(PX div 6),sa);
         CreateTSP('Подпрограммы                                  без
RET',ChangeFileExt(failv,'.brt'),(PX div 3)+3*(PX div 6),ya2,(PX
div 6)-2,sa);
         CreateTSP('Точки входа',stv,(PX div 3)+(PX div 6),ya2,(PX
div 6),sa);
         CreateTSP('Ошибочные строки!',stv,(PX div 3)+3*(PX div
6),ya1,(PX div 6)-2,sa);
         CreateGSP('Дизассемблированная программа',s,PX div
3,0,2*(PX div 3)-3,vysota+3);
         vivodDIZ;
```

```
        vydeliDannye;

{================================================================
============================}
       if KrugDiz.Down then begin
        s:=ChangeFileExt(failv,'.osp');
        if FileExists(s) then AdresaOBF(s)
                         else KrugUTVH:=0;

KolvoOshibok:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ошибочные
строки!')]).ListBox1.Items.Count;
        if KolvoOshibok>0 then begin
         if krugpervij then begin
          KrugDiz.Down:=false;
          KRGDiz.Checked:=false;
          koneckruga:=true;
          MessageBox(0,'Обнаружены ошибочные строки в программе
(см. окошко "Ошибочные строки!"). Дальнейшее "круговое"
дизассемблирование невозможно.','Сообщение программы
Диздек.',mb_Ok+mb_iconinformation);
         end else begin
          i:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Count-1;
          s:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines[i];
          TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines[i]:='';
          TTSP(Screen.Forms[OknoPoImeni(Screen,'Ошибочные
строки!')]).ListBox1.Items.Clear;
          KrugUk:=KrugUk+1;
          KrugMas[KrugUk]:=Heks(s,16,j);
          N9.Click;
         end;
         krugpervij:=false;
        end else begin
         krugpervij:=false;
         i:=0;
         repeat
          i:=i+1;
          k:=Koddannyh(KrugTVH[i],koneckruga);
         until (not k)or(i=KrugUTVH);
         if KrugUTVH=1 then DeleteFile(s);
         if k then begin
```

```
          krugom:=false;
          if koneckruga then begin
           krugom:=false;
    {       k2:=true;}
           KrugDiz.Down:=false;
           s:='"Круговое"      дизассемблирование      программы
завершено.'+#10+#13;
           if KrugUk=0 then s:=s+'Скрытых данных в программе нет.'
                       else begin
                        s:=s+'В программе имеются скрытые данные.
Их начальные адреса:'+#10+#13;
                        for   i:=1   to   KrugUk   do   s:=s+'   -
'+IntToHex(KrugMas[i],8)+'h'+#10+#13;;
                       end;
           StrPCopy(proms,s);
           MessageBox(0,proms,'Сообщение                  программы
Диздек.',mb_Ok+mb_iconinformation);
          end;
         end else begin
          if KrugUTVH>0 then begin
           TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Add(IntToHex(KrugTVH[i],8));
           N9.Click;
          end else begin
           koneckruga:=true;
           StrPCopy(proms,'"Круговое"          дизассемблирование
программы завершено.'+#10+#13+'Скрытых данных в программе нет.');
           MessageBox(0,proms,'Сообщение                  программы
Диздек.',mb_Ok+mb_iconinformation);
          end;
         end;
        end;
       end;

{================================================================
============================}
      end;
      KrugDiz.Down:=krugom;
      KRGDiz.Checked:=krugom;
    until koneckruga;
     N18Click(Sender);
     t2:=Time;
```

```
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Vrem.Caption:='Продолжительность      декомпиляции
программы: '+TimeToStr(t2-t1);
     end else MessageBox(0,'Не выбран входной файл!','Ошибка
программы Диздек',mb_Ok+mb_iconstop);
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge2.Progress:=0;
    end;

    procedure TMainForm.N4Click(Sender: TObject);
    var s:string;
    begin
     daiobf:=0;
     if pos('.obf',lowercase(failv))<>0 then begin
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Clear;
       CreateOBF(failv);
     end else
     if pos('.out',lowercase(failv))<>0 then begin
       CreateOUTosf;
     end else
     if
(pos('.exe',lowercase(failv))<>0)or(pos('.dll',lowercase(failv))
<>0) then begin
       s:=TipEXE(failv);
       if s='MZ' then begin
        daimz:=0;
        CreateMZF;
       end;
       if s='NE' then CreateNEF;
       if s='PE' then CreatePEF;
     end;
    end;

    procedure TMainForm.MZNEPE1Click(Sender: TObject);
    var s:string;
        i:integer;
    begin
     for i:=MDIChildCount-1 downto 0 do
      if  (not(MDIChildren[i]  =  Nil))and(not(MDIChildren[i]  is
TMDIChild)) then
```

```
          MDIChildren[i].Close;
     if failv='' then MessageBox(0,'Необходимо сначала открыть EXE-файл формата PE или NE.','Сообщение программы Диздек',mb_Ok+mb_iconexclamation)
     else                                                    if (pos('.exe',lowercase(failv))<>0)or(pos('.dll',lowercase(failv))<>0) then begin
       s:=TipEXE(failv);
       if s='MZ' then daimz:=0;
       if s='NE' then daimz:=1;
       if s='PE' then daimz:=1;
       CreateMZF;
     end else MessageBox(0,'Заглушку MZ можно посмотреть только у исполняемых EXE-файлов форматов PE и NE. У других файлов ее нет.','Сообщение программы Диздек',mb_Ok+mb_iconexclamation);
    end;

    procedure TMainForm.DOSClick(Sender: TObject);
    var s:string;
        z,x,y,n,k:integer;
    begin
    if TSPotkr then begin
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие параметры.')]).Gauge2.Progress:=0;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие параметры.')]).Gauge.Progress:=0;
     if  TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст программы')]).ListBox1.SelCount=0 then begin
      n:=0;
      k:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст программы')]).ListBox1.Items.Count-1;
     end else begin
      x:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст программы')]).ListBox1.SelCount;
      y:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст программы')]).ListBox1.ItemIndex;
      if y=0 then begin
       n:=0;
       k:=x-1;
      end else
      if TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст программы')]).ListBox1.Selected[y-1] then begin
       k:=y;
       n:=y-x+1;
```

```
       end else begin
        n:=y;
        k:=y+x-1;
       end;
      end;
      dlin:=k-n+1;
      x:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.maxValue;
      y:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.minValue;
      for z:=n to k do begin
       dlt:=z-n;

s:=perekod(TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный
текст программы')]).ListBox1.Items[z],'dos');
       TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный    текст
программы')]).ListBox1.Items[z]:=s;
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=dlt*(x-y) div dlin;
      end;
      TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
    end else MessageBox(0,'Кодировку шрифта можно менять только в
листинге (исходном тексте программы) в окне "Ассемблерный текст
программы".','Сообщение                                    программы
Диздек',mb_Ok+mb_iconexclamation);
    end;

    procedure TMainForm.ToolButton2Click(Sender: TObject);
    var f:textfile;
        i,x,y:integer;
    begin
     if TSPotkr then begin
      TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
       x:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.maxValue;
       y:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.minValue;
       dlin:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный
текст программы')]).ListBox1.Count-1;

assignfile(f,TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный
текст программы')]).FileName);
```

```
      rewrite(f);
      for i:=0 to dlin do begin

writeln(f,TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный
текст программы')]).ListBox1.Items[i]);
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=i*(x-y) div dlin;
      end;
      closefile(f);
      TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
     end else MessageBox(0,'Сохранить можно только листинг
(исходный текст программы) с модифицированной кодировкой из окна
"Ассемблерный текст программы".','Сообщение программы
Диздек',mb_Ok+mb_iconexclamation);
    end;

    procedure TMainForm.ToolButton15Click(Sender: TObject);
    var s:string;
        z,x,y,n,k:integer;
    begin
    if TSPotkr then begin
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge2.Progress:=0;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
     if  TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст
программы')]).ListBox1.SelCount=0 then begin
      n:=0;
      k:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст
программы')]).ListBox1.Items.Count-1;
     end else begin
      x:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст
программы')]).ListBox1.SelCount;
      y:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст
программы')]).ListBox1.ItemIndex;
      if y=0 then begin
       n:=0;
       k:=x-1;
      end else
      if TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный текст
программы')]).ListBox1.Selected[y-1] then begin
       k:=y;
       n:=y-x+1;
```

```
      end else begin
       n:=y;
       k:=y+x-1;
      end;
     end;
     dlin:=k-n+1;
     x:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.maxValue;
     y:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.minValue;
     for z:=n to k do begin
      dlt:=z-n;

s:=perekod(TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный
текст программы')]).ListBox1.Items[z],'win');
      TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный    текст
программы')]).ListBox1.Items[z]:=s;
      TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=dlt*(x-y) div dlin;
     end;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Gauge.Progress:=0;
    end else MessageBox(0,'Кодировку шрифта можно менять только в
листинге (исходном тексте программы) в окне "Ассемблерный текст
программы".','Сообщение                                    программы
Диздек',mb_Ok+mb_iconexclamation);
    end;

    procedure TMainForm.ToolButton16Click(Sender: TObject);
    var i:integer;
    begin
    if TSPotkr then begin
     TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный     текст
программы')]).ListBox1.ItemIndex:=0;
     for                          i:=0                          to
TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный          текст
программы')]).ListBox1.Items.Count-1 do
        TTSP(Screen.Forms[OknoPoImeni(Screen,'Ассемблерный  текст
программы')]).ListBox1.Selected[i]:=true;
    end else MessageBox(0,'Не открыто окно "Ассемблерный текст
программы".','Сообщение                                    программы
Диздек',mb_Ok+mb_iconexclamation);
    end;
```

```
    procedure TMainForm.N9Click(Sender: TObject);
    var stv:string;
        a:Longint;
        i,j,k,m:longint;
        s1:string;
        f2:file of zaptvh;
        fa:longword;
    begin
    if failv<>'' then begin
     stv:=ChangeFileExt(failv,'.tvh');
     if FileExists(stv) then begin
      assignfile(f2,stv);
      erase(f2);
     end;
     if FileExists(ChangeFileExt(failv,'.twh')) then begin
      assignfile(f2,ChangeFileExt(failv,'.twh'));
      erase(f2);
     end;
     for i:=0 to TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Count-1 do begin
      if         TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Checked[i] then begin
       s1:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Strings[i];
       delete(s1,1,pos('. Вектор: ',s1)+9);
       delete(s1,pos('h',s1),255);
       a:=heks(s1,16,j);

       zapisTVH(stv,a);
       zapisTVH(ChangeFileExt(failv,'.twh'),a);
      end;
     end;
     k:=0;
     j:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Count;
     i:=0;
     repeat
      s1:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines[i];
      if s1<>'' then begin
       a:=heks(s1,16,m);
       if m<>0 then begin
```

```
        TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.SelStart:=k;
        TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.SelLength:=length(s1);
        StrPCopy(proms,'В шестнадцатеричном числе: '+s1+' имеется
недопустимый символ.');
        MessageBox(0,proms,'Внимание!',mb_OK+mb_iconexclamation);
       end else begin
        zapisTVH(stv,a);
        zapisTVH(ChangeFileExt(failv,'.twh'),a);
       end;
      end;
      k:=k+length(s1)+2;
      i:=i+1;
     until (i=j)or(m<>0);
     if m=0 then begin
      zptvh:=true;
      assignfile(f2,stv);
      reset(f2);
      fa:=filesize(f2)-1;
      closefile(f2);
     end;
    end else MessageBox(0,'Выбирать и сохранять точки входа можно
только              после              открытия              входного
файла.','ВНИМАНИЕ!',mb_OK+mb_iconinformation);
    end;

    procedure TMainForm.FileCloseItemClick(Sender: TObject);
    begin
     Perform(UM_obf,2,0);
    end;

    procedure TMainForm.N12Click(Sender: TObject);
    var i,a,b,c,j:longint;
        s,s1:string;
        G:TGridRect;
        ass:array [0..255] of char;
    begin
     if not ischibit then s1:=ansilowercase(ischistroku)
                     else s1:=ischistroku;
     if s1='' then MessageBox(0,'Для поиска введена пустая строка,
а должна быть введена непустая.','ВНИМАНИЕ!',mb_OK+mb_iconerror)
```

```
     else begin

TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Clear;

TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Sorted
:=true;
      if ActiveMDIChild is TGSP then begin
       b:=TGSP(ActiveMDIChild).StringGrid1.RowCount;
       a:=TGSP(ActiveMDIChild).Stolbec;
       i:=TGSP(ActiveMDIChild).Stroka;
       c:=i;
       repeat
        s:=TGSP(ActiveMDIChild).StringGrid1.Cells[a,i];
        if not ischibit then s:=ansilowercase(s);
        i:=i+1;
        j:=pos(s1,s);
        if ischipoisk then begin
         if j<>0 then begin

s:=TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.Cells[0,i-1];

TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Items.
Add(s);
         end;
         j:=0;
        end;
       until (i=b)or(j<>0);
       i:=i-1;
       if not ischipoisk then
        if pos(s1,s)<>0 then begin
        TGSP(ActiveMDIChild).StringGrid1.LeftCol:=0;
        TGSP(ActiveMDIChild).StringGrid1.TopRow:=i;
        G.Left:=a;
        G.Right:=a;
        if   i=TGSP(ActiveMDIChild).StringGrid1.RowCount-1   then
G.Bottom:=i
                                                          else
G.Bottom:=i+1;
        G.Top:=i;
        TGSP(ActiveMDIChild).StringGrid1.Selection:=G;
        TGSP(ActiveMDIChild).Stroka:=G.Bottom;
       end else begin
```

```
        StrPCopy(ass,'Подстрока
:'+#10+#13+#10+#13+ischistroku+#10+#13+#10+#13+'в         столбце
'+IntToStr(a)+'    ниже    строки    '+IntToStr(c)+'    нигде    не
встречается.');
        MessageBox(0,ass,'ВНИМАНИЕ!',mb_OK+mb_iconexclamation);
       end;
      end;
      if                    (ActiveMDIChild                    is
TTSP)and(TTSP(ActiveMDIChild).Caption='Ассемблерный         текст
программы') then begin
       b:=TTSP(ActiveMDIChild).ListBox1.Count;
       i:=TTSP(ActiveMDIChild).ListBox1.ItemIndex;
       c:=i;
       repeat
        s:=TTSP(ActiveMDIChild).ListBox1.Items.Strings[i];
        if not ischibit then s:=ansilowercase(s);
        i:=i+1;
       until (i=b)or(pos(s1,s)<>0);
       i:=i-1;
       if pos(s1,s)<>0 then begin
        TTSP(ActiveMDIChild).ListBox1.Selected[i]:=true;
        if                        b>i+1                        then
TTSP(ActiveMDIChild).ListBox1.ItemIndex:=i+1
                else TTSP(ActiveMDIChild).ListBox1.ItemIndex:=i;
        listing:=TTSP(ActiveMDIChild).ListBox1.ItemIndex;
       end else begin
        StrPCopy(ass,'Подстрока
:'+#10+#13+#10+#13+ischistroku+#10+#13+#10+#13+'в         столбце
'+IntToStr(a)+'    ниже    строки    '+IntToStr(c)+'    нигде    не
встречается.');
        MessageBox(0,ass,'ВНИМАНИЕ!',mb_OK+mb_iconexclamation);
       end;
      end;
     end;
    end;

    procedure TMainForm.N11Click(Sender: TObject);
    begin
     DlgPoisk.Showmodal;
     if ischistroku<>'' then N12Click(Sender);
    end;

    procedure TMainForm.N10Click(Sender: TObject);
```

```
    var p:byte;
        a,b,i:integer;
        s,s1:string;
        c:longint;
    begin
     if ActiveMDIChild is TGSP then begin
      chislok:=0;
      a:=0;
      b:=0;

s:=TGSP(ActiveMDIChild).StringGrid1.Cells[3,TGSP(ActiveMDIChild)
.Stroka];

s1:=TGSP(ActiveMDIChild).StringGrid1.Cells[0,TGSP(ActiveMDIChild
).Stroka];
      delete(s1,length(s1),1);
      c:=heks(s1,16,a);
      StrPCopy(mp,s);
      p:=0;
      p:=p or zapadresa(7);                  {адресный переходъ}
      StrPCopy(mp,s);
      p:=p or zapadresa(9);                  {подпрограмма с RET}
      StrPCopy(mp,s);
      p:=p or zapadresa(11);                {подпрограмма без RET}
      case chislok of
       0:MessageBox(0,'В выбранной команде нет ни адресных
переходов,                          ни                          вызовов
подпрограмм','ВНИМАНИЕ!',mb_OK+mb_iconinformation);
       1:begin
          prk:=kuk[1];
          i:=prk.perehod;
          prk.adres:=c;
          kuk[1]:=prk;
          a:=i div $10000;
          b:=i mod $10000;
          Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')].Perform(UM_Pereh,a,b);
         end
       else begin
        CreateTSP('В данной команде несколько адресных переходов.
Выберите нужный из них двойным щелчком мыши.','Пустое
место',0,vysota+3,PX-5,(TSH div 4));
```

```
        TTSP(Screen.Forms[OknoPoImeni(Screen,'В данной команде несколько адресных переходов. Выберите нужный из них двойным щелчком мыши.')]).ListBox1.Items.Add(s);
        for i:=1 to chislok do begin
         prk:=kuk[i];
         prk.adres:=c;
         kuk[i]:=prk;
         if prk.p=0 then TTSP(Screen.Forms[OknoPoImeni(Screen,'В данной команде несколько адресных переходов. Выберите нужный из них двойным щелчком мыши.')]).ListBox1.Items.Add(IntToHex(prk.perehod,8));
        end;
       end;
      end;
     end else MessageBox(0,'Для перемещения по адресным переходам необходимо выбрать команду в окне "Ассемблерный текст программы"','ВНИМАНИЕ!',mb_OK+mb_iconinformation);
    end;

    procedure TMainForm.Ret(Sender: TObject);
    var a,b,i:integer;
    begin
     if failv<>'' then begin
      prk:=kuk[1];
      i:=prk.adres;
      a:=i div $10000;
      b:=i mod $10000;
      Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная программа')].Perform(UM_Pereh,a,b);
     end;
    end;

    procedure TMainForm.N16Click(Sender: TObject);
    var s:string;
        i:integer;
    begin
     if ActiveMDIChild is TGSP then begin
      i:=TGSP(ActiveMDIChild).Stroka;
      s:=TGSP(ActiveMDIChild).StringGrid1.Cells[0,i];
      delete(s,length(s),1);
      Clipboard.Open;
      Clipboard.AsText:=s;
      Clipboard.Close;
```

```
      TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Add(s);
      N9Click(Sender);
     end else MessageBox(0,'Для записи адреса дизассемблируемой
команды в качестве точки входа необходимо выбрать команду в окне
"Ассемблерный                                                текст
программы"','ВНИМАНИЕ!',mb_OK+mb_iconinformation);
    end;

    procedure TMainForm.N17Click(Sender: TObject);
    var s1,s,skp,szp,sbp,stv,sf:string;
        j:longint;
        f4:textfile;
        x,y,ya1,ya2,sa:integer;
    begin
    if   pos('.obf',lowercase(ExtractFileName(failv)))<>0   then
begin
     vysota:=3*(TSH div 4)-3;
     if   pos('.obf',lowercase(ExtractFileName(failv)))<>0   then
AdresaOBF(failv);
     KrugUTVH:=0;
     sf:=ChangeFileExt(failv,'.twh');
     if failv<>'' then begin
     if FileExists(sf) then begin
      zptvh:=true;
      Perform(UM_obf,2,0);
      ucetadresa:=false;
      Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Soob,0,0);
      s1:=ChangeFileExt(failv,'.lst');
      if not FileExists(s1) then s1:=ChangeFileExt(failv,'.asm');
      ya1:=3*(TSH div 4);
      sa:=(TSH div 4) div 2;
      ya2:=ya1+sa;
      if FileExists(s1) then begin
       CreateTSP('Ассемблерный   текст   программы',s1,(PX   div
3),3*(TSH div 4),2*(PX div 3)-2,(TSH div 4));
       vysota:=2*(TSH div 4)-3;
       ya1:=vysota+3;
       ya2:=ya1+sa;
       dlin:=SizeByte(s1);
      end;
      s:=ChangeFileExt(failv,'.dap');
```

```
      skp:=ChangeFileExt(failv,'.kp');
      szp:=ChangeFileExt(failv,'.zp');
      sbp:=ChangeFileExt(failv,'.bp');
      stv:=ChangeFileExt(failv,'.tvh');
      CreateTSP('Запись ОЗУ программ',szp,(PX div 3),ya1,(PX div
6),sa);
      CreateTSP('ПОИСК.','',(PX div 3),ya2,(PX div 6),sa);
      CreateTSP('Косвенные переходы',skp,(PX div 3)+(PX div
6),ya1,(PX div 6),sa);
      CreateTSP('Переходы из программы!',skp,(PX div 3)+(PX div
6),ya2,(PX div 6),sa);
      CreateTSP('Подпрограммы',ChangeFileExt(failv,'.ret'),(PX
div 3)+3*(PX div 6),ya1,(PX div 6)-2,sa);
      CreateTSP('Подпрограммы                                  без
RET',ChangeFileExt(failv,'.brt'),(PX div 3)+3*(PX div 6),ya2,(PX
div 6)-2,sa);
      CreateTSP('Точки входа',stv,(PX div 3)+2*(PX div 6),ya1,(PX
div 6),sa);
      CreateTSP('Ошибочные строки!',stv,(PX div 3)+2*(PX div
6),ya2,(PX div 6),sa);
      CreateGSP('Дизассемблированная программа',s,PX div
3,0,2*(PX div 3)-3,vysota+3);
      vivodDIZ;
     end else begin
      StrPCopy(proms,'Не                                       найден
файл'+#10+#13+sf+#10+#13+#10+#13+'Это, скорее всего, признак
того, что исходный файл'+#10+#13+failv+#10+#13+'еще не
дизассемблировался. Дизассемблируйте его соответствующей
командой, кнопкой или нажатием F9.');
      MessageBox(0,proms,'ВНИМАНИЕ!',mb_OK+mb_iconerror);
     end;
     end else MessageBox(0,'Необходимо сначала открыть файл
программы,                                  подлежащей
дизассемблированию.','ВНИМАНИЕ!',mb_OK+mb_iconinformation);
    end else MessageBox(0,'Для продолжения дизассемблирования
необходимо открыть файл с расширением
OBF.','ВНИМАНИЕ!',mb_OK+mb_iconinformation);
    end;

    procedure TMainForm.N18Click(Sender: TObject);
    var sfl,s,tvhobf,tvhprer,shrft:string;
        f:file of byte;
        kodp,i,ctv,cp:word;
        a,b,c,d,opersis:byte;
```

```
        zag1:longword;
        f2:file of zaptvh;
        fa,a1,a2,ap,av:longint;
        szp,s2,stv,skp,sbp:string;
        j,j1,il:longint;
        kpr:kosreg;
        f3:file of kosreg;
        neis,samod,kosp,zd,skd:boolean;
        fpr:file of promfile;
        zpr:promfile;
        mc:masblok;
        lst:TComboBox;
        lss,lso:string;
    begin
    if ub>0 then begin
     ctv:=0;
     cp:=0;
     neis:=false;
     skd:=false;
     sfl:=ChangeFileExt(failv,'.rtf');
     samod:=false;
     kosp:=false;
     shrft:='24';
     assignfile(f,ChangeFileExt(failv,'.obf'));
     reset(f);
     zag1:=filesize(f);
     closefile(f);
     if FileExists(sfl) then DeleteFile(sfl);
     rtf(0,sfl);
     StrPCopy(proms,'Отчет     о     дизассемблировании     файла
'+ExtractFileName(failv)+'.');
     stp(sfl,1,'ц','32','жн');
     StrPCopy(proms,' ');
     stp(sfl,1,'ш',shrft,'н');
     StrPCopy(proms,'1. Имя файла: ');
     stp(sfl,1,'ш',shrft,'ж');
     StrPCopy(proms,ExtractFileName(failv)+'.');
     stp(sfl,1,'ш',shrft,'н');
     StrPCopy(proms,'2. Объем файла: ');
     stp(sfl,1,'ш',shrft,'ж');
     StrPCopy(proms,TochkiVChislo(zag1)+' байт.');
```

```
stp(sfl,1,'ш',shrft,'н');

StrPCopy(proms,'3. Каталог, в котором хранится файл: ');
stp(sfl,1,'ш',shrft,'ж');
StrPCopy(proms,ExtractFilePath(failv)+'.');
stp(sfl,1,'ш',shrft,'н');

assignfile(f,failv);
reset(f);
read(f,a,b);
c:=b and $03;
kodp:=ByteToWord(a,b) div 4;
{kodim(kodp,failproc,s);}
StrPCopy(proms,'4. Процессор: ');
stp(sfl,1,'ш',shrft,'ж');
StrPCopy(proms,processor+'.');
stp(sfl,1,'ш',shrft,'н');
StrPCopy(proms,'5. Виды слов исполняемого модуля: ');
stp(sfl,1,'ш',shrft,'ж');
ed:=$0001 shl c;
ce:=c;
tipslova:=c;
case c of
 0:s:='байты.';
 1:s:='слова (16 бит).';
 2:s:='двойные слова (32 бита).';
 3:s:='четырехкратные слова (64 бита).';
end;
StrPCopy(proms,s);
stp(sfl,1,'ш',shrft,'н');
read(f,opersis);
case opersis of
 0:s:=' Никакая не используется.';
 1:s:=' MS-DOS, формат файлов COM.';
 2:s:=' MS-DOS, формат файлов MZ.';
 3:s:=' MS Windows 3.x (формат файлов NE).';
 4:s:=' MS Windows 95/98/XP/NT 3.5 (формат файлов PE).';
 5:s:=' OS/2.';
end;
s:=s;
StrPCopy(proms,'6. Используемая операционная система: ');
```

```
     stp(sfl,1,'ш',shrft,'ж');
     StrPCopy(proms,s);
     stp(sfl,1,'ш',shrft,'н');

     read(f,a,b);
     zag1:=ByteToLongword(0,0,a,b);
     av:=0;
     StrPCopy(proms,'7.   Адресное   пространство,   занимаемое
программой.');
     stp(sfl,1,'ш',shrft,'жн');
     for j:=1 to ub do begin
      mc:=mbb[j];
      a1:=mc.sec2;
      a2:=mc.sec4;
      av:=av+a2+1-a1;
      StrPCopy(proms,'  7.'+IntToStr(j)+'.  '+IntToHex(a1,8)+'h  -
'+IntToHex(a2,8)+'h.');
      if j<>ub then stp(sfl,1,'ш',shrft,'н')
               else stp(sfl,1,'ш',shrft,'');
     end;
     StrPCopy(proms,' - Всего слов в программе: '+IntToStr(av)+'
или '+IntToHex(av,8)+'h.');
     stp(sfl,1,'ш',shrft,'жкн');
     StrPCopy(proms,'8. Точки входа в программу.');
     stp(sfl,1,'ш',shrft,'жн');
     s:='  8.1.  Указанные  в  самом  исполняемом  модуле  (их
'+IntToStr(zag1)+' шт.): ';
     StrPCopy(proms,s);
     stp(sfl,1,'ш',shrft,'ж');
     tvhobf:='';
     for i:=1 to zag1 do begin
      read(f,a,b,c,d);
      s:=IntToHex(ByteToLongword(a,b,c,d),8);
      tvhobf:=tvhobf+Uppercase(s)+' ';
      s:=s+'h';
      if zag1=1 then s:=s+'.' else begin
       if i<>zag1 then s:=s+', ' else s:=s+'.';
      end;
      StrPCopy(proms,s);
      stp(sfl,1,'ш',shrft,'');
     end;
     StrPCopy(proms,' ');
```

```
     stp(sfl,1,'ш',shrft,'н');
     tvhprer:='';
     StrPCopy(proms,' 8.2. Вектора прерываний, выбранные вручную
или автоматически при дизассемблировании.');
     stp(sfl,1,'ш',shrft,'жн');
     a:=1;
     for i:=0 to TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Count-1 do begin
      if         TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Checked[i] then begin
       s:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).PRER.Items.Strings[i];
       s2:=s;
       delete(s2,1,pos('. Вектор: ',s2)+9);
       delete(s2,pos('h',s2),255);
       tvhprer:=tvhprer+Uppercase(s2)+' ';
       StrPCopy(proms,'   7.2.'+IntToStr(a)+'. '+s);
       stp(sfl,1,'ш',shrft,'н');
       a:=a+1;
      end;
     end;
     cp:=a-1;

     StrPCopy(proms,' 8.3. Введенные вручную или автоматически
после анализа частично дизассемблированной программы: ');
     stp(sfl,1,'ш',shrft,'ж');
     a:=0;
     if          TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Count>0 then begin
      zag1:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Count-1;
      for i:=0 to zag1 do begin

s:=Uppercase(TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Strings[i]);
       if (pos(s,tvhobf)=0)and(pos(s,tvhprer)=0) then begin
        s:=s+'h';
        a:=a+1;
        if zag1=1 then s:=s+'.' else begin
         if i<>zag1 then s:=s+', ' else s:=s+'.';
        end;
        StrPCopy(proms,s);
        stp(sfl,1,'ш',shrft,'');
```

```
       end;
      end;
     end;
     if a=0 then begin
              StrPCopy(proms,'        Точки входа вручную не
вводились.');
              stp(sfl,1,'ш',shrft,'н');
             end else begin
              StrPCopy(proms,'   ');
              stp(sfl,1,'ш',shrft,'н');
              StrPCopy(proms,'ВНИМАНИЕ:');
              stp(sfl,1,'ш',shrft,'пк');
              StrPCopy(proms,'   Использован режим "кругового
дизассемблирования" - режим поиска скрытых точек входа в программу
по критерию отсутствия недопустимых кодов команд в блоке
дизассемблируемых команд.');
              stp(sfl,1,'ш',shrft,'кн');
              s:='   Всего введено таким образом '+IntToStr(a);
              s2:=copy(s,length(s),1);
              ctv:=a;
              if s2='1' then s:=s+' точка входа.'
                 else if (s2='2')or(s2='3')or(s2='4') then s:=s+'
точки входа.'
                 else s:=s+' точек входа.';
              StrPCopy(proms,s);
              stp(sfl,1,'ш',shrft,'н');
             end;

     s2:='            ';
     szp:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись        ОЗУ
программ')]).FileName;
     fa:=-1;
     StrPCopy(proms,'9.  Адреса команд, записывающих память
программ.');
     stp(sfl,1,'ш',shrft,'ж');
     if fileexists(szp) then begin
      samod:=true;
      fa:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись        ОЗУ
программ')]).ListBox1.Count-1;
      s:=' (их '+IntToStr(fa)+' шт.): ';
      StrPCopy(proms,s);
      stp(sfl,1,'ш',shrft,'ж');
      for j:=1 to fa do begin
```

```
       s:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Запись        ОЗУ
программ')]).ListBox1.Items.Strings[j];
       s:=copy(s,1,pos('h',s));
       if fa=1 then s:=s+'.' else begin
       if j<>fa then s:=s+', ' else s:=s+'.';
       StrPCopy(proms,s);
       stp(sfl,1,'ш',shrft,'');
      end;
      end;
      StrPCopy(proms,' ');
      stp(sfl,1,'ш',shrft,'н');
     end else begin
      StrPCopy(proms,': Таких команд в данной программе нет.');
      stp(sfl,1,'ш',shrft,'н');
     end;

     StrPCopy(proms,'10. Адреса команд косвенных переходов.');
     stp(sfl,1,'ш',shrft,'ж');
     skp:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Косвенные
переходы')]).FileName;
     if fileexists(skp) then begin
      lst:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).ListBox1;
      kosp:=true;
      fa:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Косвенные
переходы')]).ListBox1.Count-2;
      s:=' (их '+IntToStr(fa)+' шт.) ';
      StrPCopy(proms,s);
      stp(sfl,1,'ш',shrft,'ж');
      StrPCopy(proms,' представлены в табл. 1.');
      stp(sfl,1,'ш',shrft,'жн');
      StrPCopy(proms,'Таблица 1.');
      stp(sfl,1,'п',shrft,'жн');
      StrPCopy(proms,' ');
      stp(sfl,1,'ш','16','н');
      StrPCopy(proms,'Преобразуйте текст в таблицу под строкой
"Таблица 1" следующей командой:');
      stp(sfl,1,'ш','16','кн');
      StrPCopy(proms,' - Для редактора Word версий 2003 или ниже:
"Таблица/Преобразовать/Текст в таблицу" с разделителем % и
автоподбором по содержимому.');
      stp(sfl,1,'ш','16','кн');
```

```
      StrPCopy(proms,' - Для редактора Word версий 2007 или выше:
"Вставка/Таблица/Преобразовать в таблицу" с разделителем % и
автоподбором по содержимому.');
      stp(sfl,1,'ш','16','кн');
      StrPCopy(proms,' ');
      stp(sfl,1,'ш','16','н');
      StrPCopy(proms,'Признак:П.Р.К%Адрес команды%Точка входа, на
которой она обнаружена%Регистр, определяющий косвенный переход
(адрес:модификатор (Название регистра))');
      stp(sfl,1,'ш',shrft,'н');
      for j:=0 to fa-1 do begin
       chitKP(skp,j,j1,kpr);
       s:=IntToHex(kpr.pr,2)+'h: ';
       if (kpr.pr and $04)<>0 then s:=s+'П.' else s:=s+'-.';
       if (kpr.pr and $02)<>0 then s:=s+'Р.' else s:=s+'-.';
       if (kpr.pr and $01)<>0 then s:=s+'К' else s:=s+'-';
       lss:=' - {'+IntToHex(ByteToWord(kpr.areg1,kpr.areg2),4)+'}
{'+IntToHex(ByteToWord(kpr.am1,kpr.am2),4)+'}';
       il:=0;
       repeat
        lso:=lst.Items[il];
        il:=il+1;
       until (il=lst.Items.Count)or(pos(lss,lso)<>0);
       if pos(lss,lso)<>0 then begin
        delete(lso,pos(lss,lso),255);
        lss:=IntToHex(ByteToWord(kpr.areg1,kpr.areg2),4)+'h     :
'+IntToHex(ByteToWord(kpr.am1,kpr.am2),4)+'h'+' ('+lso+')';
       end                                                  else
lss:=IntToHex(ByteToWord(kpr.areg1,kpr.areg2),4)+'h            :
'+IntToHex(ByteToWord(kpr.am1,kpr.am2),4)+'h';

s:=s+'%'+IntToHex(ByteToLongword(kpr.adres1,kpr.adres2,kpr.adres
3,kpr.adres4),8)+'h%'+IntToHex(ByteToLongword(kpr.t1,kpr.t2,kpr.
t3,kpr.t4),8)+'h%'+lss;
       StrPCopy(proms,s);
       stp(sfl,1,'ш',shrft,'н');
      end;
     end else begin
      StrPCopy(proms,' ');
      stp(sfl,1,'ш',shrft,'н');
      StrPCopy(proms,'   Таких команд в данной программе нет.');
      stp(sfl,1,'ш',shrft,'н');
     end;
```

```
     stv:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).FileName;
     StrPCopy(proms,'11. Точки входа, обнаруженные в программе при
дизассемблировании.');
     stp(sfl,1,'ш',shrft,'ж');
     if fileexists(stv) then begin
      fa:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Count-2;
      StrPCopy(proms,' (их '+IntToStr(fa)+' шт.): ');
      stp(sfl,1,'ш',shrft,'ж');
      s2:='';
      for j:=2 to fa+1 do begin
       s:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Точки
входа')]).ListBox1.Items.Strings[j];
       if pos('HE',s)<>0 then begin
        s2:='жп';
        neis:=true;
       end;
       StrPCopy(proms,copy(s,1,pos('h',s)));
       stp(sfl,1,'ш','20',s2);
       if fa=1 then s:='.' else
               if j<>fa then s:=', ' else s:='. ';
       StrPCopy(proms,s);
       stp(sfl,1,'ш','20','');
      end;
      StrPCopy(proms,' ');
      stp(sfl,1,'ш',shrft,'н');
      if neis then begin
       StrPCopy(proms,'Жирным шрифтом и подчеркиванием выделены
точки входа, не использованные при дизассемблировании.');
       stp(sfl,1,'ш',shrft,'н');
      end;
     end;


sbp:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).File
Name;
     StrPCopy(proms,'12. Адреса подпрограмм, обнаруженные при
дизассемблировании.');
     stp(sfl,1,'ш',shrft,'ж');
     if fileexists(sbp) then begin
```

```
fa:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListB
ox1.Count-1;
      s:=' (их '+IntToStr(fa)+' шт.): ';
      StrPCopy(proms,s);
      stp(sfl,1,'ш',shrft,'ж');
      for j:=1 to fa do begin

s:=TTSP(Screen.Forms[OknoPoImeni(Screen,'Подпрограммы')]).ListBo
x1.Items.Strings[j];
       s:=copy(s,1,pos('h',s));
       if fa=1 then s:=s+'.' else begin
        if j<>fa then s:=s+', ' else s:=s+'.';
        StrPCopy(proms,s);
        stp(sfl,1,'ш',shrft,'');
       end;
      end;
      StrPCopy(proms,' ');
      stp(sfl,1,'ш',shrft,'н');
     end else begin
      StrPCopy(proms,' ');
      stp(sfl,1,'ш',shrft,'н');
      StrPCopy(proms,'   Таких команд в данной программе нет.');
      stp(sfl,1,'ш',shrft,'н');
     end;

     s:=ChangeFileExt(failv,'.osp');
     if FileExists(s) then begin
      skd:=true;
      StrPCopy(proms,'13. В этой программе обнаружены скрытые
данные.');
      stp(sfl,1, 'ш',shrft,'жн');
      StrPCopy(proms,' 13.1. Имя файла, куда они помещены -
'+s+'.');
      stp(sfl,1,'ш',shrft,'н');
      StrPCopy(proms,' 13.2. Начальные и конечные адреса этих
данных.');
      stp(sfl,1,'ш',shrft,'н');
      assignfile(f,s);
      reset(f);
      seek(f,9);
```

```
      i:=1;
      repeat
       read(f,a,b,c,d);
       zag2:=ByteToLongword(a,b,c,d);
       read(f,a,b,c,d);
       zag3:=ByteToLongword(a,b,c,d);
       StrPCopy(proms,'           13.2.'+IntToStr(i)+'.      Блок
'+IntToStr(i)+':              '+IntToHex(zag2,8)+'h                -
'+IntToHex(zag3+zag2,8)+'h.');
       stp(sfl,1,'ш',shrft,'н');
       if ed=1 then seek(f,filepos(f)+zag3+1)
               else seek(f,filepos(f)+(zag3+ed-1)*ed);
       i:=i+1;
       dlt:=dlt+1;
      until eof(f);
      closefile(f);
      s:='14. ';
     end else begin
      s:='13. ';
     end;


StrPCopy(proms,s+TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общи
е параметры.')]).Vrem.Caption+'.');
     stp(sfl,1,'ш',shrft,'жн');
     StrPCopy(proms,' ');
     stp(sfl,1,'ш',shrft,'н');

     StrPCopy(proms,'Заключение.');
     stp(sfl,1,'ц',shrft,'жпн');
     StrPCopy(proms,'Дизассемблирование программы ');
     stp(sfl,1,'ш',shrft,'');
     if (ctv=0)and(cp>1) then s:='было полностью автоматическим'
        else begin
         s:='';
         if   ctv>0   then   s:='потребовало   ручного   ввода
дополнительных точек входа в количестве '+IntToStr(ctv)+' штук';
         if cp>0 then
            if s<>'' then s:=s+' и указания аппаратных прерываний
в количестве '+IntToStr(cp)+' штук'
                     else  s:='потребовало  указания  аппаратных
прерываний в количестве '+IntToStr(cp)+' штук';
```

```
        end;
      s:=s+'. ';
      StrPCopy(proms,s);
      stp(sfl,1,'ш',shrft,'');
      if not(samod or kosp) then begin
       StrPCopy(proms,'Программа не содержит косвенных переходов и
элементов самомодификации. ');
       stp(sfl,1,'ш',shrft,'');
      end else begin
       s:='';
       if samod then s:='Программа является самомодифицирующейся
(см. п. 9). ';
       if kosp then s:=s+'Программа содержит косвенные переходы
(см. п. 10). ';
       StrPCopy(proms,s);
       stp(sfl,1,'ш',shrft,'');
      end;
      if neis then begin
       StrPCopy(proms,'Дизассемблер использовал не все точки входа
(см. п. 11). ');
       stp(sfl,1,'ш',shrft,'');
      end;
      if skd then begin
       StrPCopy(proms,'В программе обнаружены исходные данные или
скрытый программный код (см. п. 13). ');
       stp(sfl,1,'ш',shrft,'н');
      end;
      StrPCopy(proms,' ');
      stp(sfl,1,'ш',shrft,'н');
      zd:=false;

      rtf(2,sfl);
      StrPCopy(proms,'Сформирован отчет в
файл:'+#10+#13+sfl+#10+#13+#10+#13+'Преобразуйте текст под
строкой "Таблица 1", если она есть, в таблицу командой редактора
Word "Таблица/Преобразовать/Текст в таблицу" с разделителем % и
автоподбором по содержимому.');
      MessageBox(0,proms,'ВНИМАНИЕ!',mb_OK+mb_iconinformation);
    end else MessageBox(0,'Для формирования отчета о
дизассемблировании необходимо сначала либо дизассемблировать файл
(по F9), либо открыть ранее дизассемблированный файл (по
F10).','ВНИМАНИЕ!',mb_OK+mb_iconexclamation);
    end;
```

```
    procedure TMainForm.N20Click(Sender: TObject);
    var tbs:array [0..255] of char;
        s:string;
    begin
     s:=ChangeFileExt(failv,'.osp');
     if FileExists(s) then begin
      CreateOBF(s);
     end else begin
       StrPCopy(tbs,'Не                                    найден
файл:'+#10+#13+s+#10+#13+#10+#13+'Скрытых данных в этой программе
нет.');
       MessageBox(0,tbs,'ВНИМАНИЕ!',mb_OK+mb_iconinformation);
     end;
    end;

    procedure TMainForm.N21Click(Sender: TObject);
    var i,a:integer;
        s:string;
    begin
     i:=OknoPoImeni(Screen,'Файл промежуточного формата с кодами
команд или данных');
     if i>=0 then begin
      a:=TOBFF(Screen.Forms[OknoPoImeni(Screen,'Файл
промежуточного     формата     с     кодами     команд     или
данных')]).ListBox2.Items.Count-1;
      for i:=0 to a do begin
       s:=TOBFF(Screen.Forms[OknoPoImeni(Screen,'Файл
промежуточного     формата     с     кодами     команд     или
данных')]).ListBox2.Items.Strings[i];
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Memo1.Lines.Add(s);
      end;
      N9Click(Sender);
     end else MessageBox(0,'Сначала необходимо открыть окно "Файл
промежуточного формата с кодами команд или данных" командой
"Обход/Просмотр скрытых данных в программе" или нажатием клавиши
F4.','ВНИМАНИЕ!',mb_OK+mb_iconexclamation);
    end;

    procedure TMainForm.N23Click(Sender: TObject);
    begin
     SUZRd.Showmodal;
```

```
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).zSUZ.Caption:=failSUZ;
    end;

    procedure TMainForm.KrugDizClick(Sender: TObject);
    begin
     KRGDiz.Checked:=KrugDiz.Down;
     krugom:=KrugDiz.Down;
     krugpervij:=true;
     koneckruga:=false;
    end;

    procedure TMainForm.KRGDizClick(Sender: TObject);
    begin
     KrugDiz.Down:=KRGDiz.Checked;
     krugom:=KRGDiz.Checked;
     krugpervij:=true;
     koneckruga:=false;
    end;

    procedure TMainForm.ApiosClick(Sender: TObject);
    var s,s1,s2,sf:string;
        o:integer;
        f:file of byte;
        a,b:byte;
        kodp:word;
    begin
     if    (failv<>'')and(pos('.obf',lowercase(failv))<>0)    then
begin
      N29Click(Sender);
      if PPardek then begin
       s:=katAPIOS+'APIOS.exe';
       if FileExists(s) then begin
        DeleteFile(ChangeFileExt(failv,'.ext'));
        StrPCopy(proms,s+' '+APIOSpar);
        o:=WinExec(@proms,sw_ShowNormal);
        if o<=31 then case o of
                        0:MessageBox(0,'Для    запуска    программы
APIOS.exe системе не хватает памяти или ресурсов.','Сообщение
программы Диздек.',mb_Ok+mb_iconstop);
                        ERROR_BAD_FORMAT:begin
```

```
                                        StrPCopy(proms,'Ошибка в
файле:'+#10+#13+s+#10+#13+#10+#13+'Либо он не является EXE-файлом
формата PE, либо у него искажен образ.');

MessageBox(0,proms,'Сообщение                                   программы
Диздек.',mb_Ok+mb_iconstop);
                                      end;
                    end;
      end else begin
       StrPCopy(proms,'Не          найдена          программа
'+#10+#13+#10+#13+s+#10+#13+#10+#13+'Найдите          программу
"APIOS.exe" и повторите это действие снова.');
       MessageBox(0,proms,'Сообщение                    программы
Диздек.',mb_Ok+mb_iconstop);
      end;

      repeat
      until Fileexists(ChangeFileExt(failv,'.ext'));

      if progDekom<>'' then
      if FileExists(ProgDekom) then begin
       StrPCopy(proms,ProgDekom+'
"'+ChangeFileExt(failv,'.kfg')+'"');
       o:=WinExec(@proms,sw_ShowNormal);
       if o<=31 then case o of
                    0:MessageBox(0,'Для   запуска   выбранной
программы синтеза текста (PASKAL.EXE, Si.EXE, Verilog.EXE) системе
не   хватает   памяти   или   ресурсов.','Сообщение   программы
Диздек.',mb_Ok+mb_iconstop);
                    ERROR_BAD_FORMAT:begin
                                        StrPCopy(proms,'Ошибка в
файле:'+#10+#13+progDekom+#10+#13+#10+#13+'Либо  он  не  является
EXE-файлом формата PE, либо у него искажен образ.');

MessageBox(0,proms,'Сообщение                                   программы
Диздек.',mb_Ok+mb_iconstop);
                                      end;
                    end;
      end else begin
       StrPCopy(proms,'Не          найдена          программа
'+#10+#13+progDekom+#10+#13+#10+#13+'Найдите  ее  и  повторите  это
действие снова.');
       MessageBox(0,proms,'Сообщение                    программы
Диздек.',mb_Ok+mb_iconstop);
```

```
       end;
      end;
     end else MessageBox(0,'Сначала надо выбрать входной файл с
расширением OBF.','Сообщение программы
Диздек.',mb_Ok+mb_iconstop);
    end;

    procedure TMainForm.ZapRegis;
    var f:file of rabp;
        z:rabp;
        b:file of byte;
        c,d:byte;
        e,i,j:word;
        s:string;
        n,k:longword;
    begin
     if (pos('.obf',failv)<>0)and(failv<>'') then begin
      assignfile(b,failv);
      reset(b);
      read(b,c,d);
      e:=ByteToWord(c,d) div 4;
      closefile(b);
      s:=katSBD+'Proc'+IntToHex(e,4)+'.rfp';
      assignfile(f,s);
      reset(f);
      for i:=3 to 8 do begin
       seek(f,filesize(f)-9+i);
       read(f,z);
       n:=StrToInt(z.mat);
       k:=StrToInt(z.mnem);
       if not((n=0)and(k=0)) then
          for j:=n to k do begin
           seek(f,j);
           read(f,z);
           if (i>3)and(i<8) then
delete(z.mat,pos(':',z.mat),255);
           registr:=z.mat+' - '+z.mnem;
           Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,3,0);
          end;
      end;
      closefile(f);
```

```
 end;
end;

procedure TMainForm.ZapRegisPol;
var f:file of ZapisRGP;
    z:ZapisRGP;
    f1:file of dpf;
    z1:dpf;
    b:file of byte;
    c,d,r:byte;
    e,i,j:word;
    s:string;
    n,k:longword;
begin
 if FileExists(failRGP) then begin
  assignfile(b,failv);
  reset(b);
  read(b,c,d);
  e:=ByteToWord(c,d) div 4;
  closefile(b);
  s:=katSBD+'Proc'+IntToHex(e,4)+'.rff';
  assignfile(f1,s);
  reset(f1);
  repeat
   read(f1,z1);
  until (z1.im)='разрядность адресов';
  closefile(f1);
  razriadAdresa:=StrToInt(IntToHex(z1.zn1,8));
  r:=razriadAdresa div 4;

  assignfile(f,failRGP);
  reset(f);
  for i:=1 to filesize(f) do begin
   read(f,z);
   registr:=z.imia+'       -       {'+IntToHex(z.adres,r)+'}
{'+IntToHex(((z.zona*$1000)+(z.StarBit*$40)+z.dlina),4)+'}';
   Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,3,0);
  end;
  closefile(f);
 end;
end;
```

```
    procedure TMainForm.N25Click(Sender: TObject);
    var k:integer;
        s,s1,sk:string;
        a,b:word;
        z,i:longword;
        zpr:promfile;
        fp:file of promfile;
    begin
     k:=OknoPoImeni(Screen,'Дизассемблированная программа');
     if k<>-1 then begin
      Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,4,0);
      k:=RGSTZap.nomer;
      if k<>-1 then begin
       RGSTZap.znachenie:=0;
       s:=RGSTZap.stroka;
       s1:=s;
       delete(s1,1,pos('{',s1));
       delete(s1,pos('}',s1),255);
       RGSTZap.adres:=heks(s1,16,k);
       delete(s,1,pos('{',s));
       delete(s,1,pos('{',s));
       delete(s,pos('}',s),255);
       a:=heks(s,16,k);
       RGSTZap.segment:=a div $1000;
       a:=a mod $1000;
       RGSTZap.nachalo:=a div $40;
       b:=a mod $40;
       RGSTZap.konec:=RGSTZap.nachalo-b+1;
       if RGSTZap.WR then begin
        sk:='_zuWR';
        if RGSTZap.Proverka=1 then begin
         s:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).Chislo.Text;
         if s<>'' then begin
          if copy(s,length(s),1)<>'h' then try
           RGSTZap.Proverka:=1;
           RGSTZap.znachenie:=StrToInt(s);
          except
```

```
          on EConvertError do MessageBox(0,'Ошибка преобразования
десятичного              числа!','Сообщение              программы
Диздек.',mb_Ok+mb_iconstop);
         end else begin
          RGSTZap.znachenie:=heks(copy(s,1,length(s)-1),16,k);
          if k=0 then begin
           RGSTZap.Proverka:=1;
          end    else    MessageBox(0,'Ошибка    преобразования
шестнадцатеричного          числа!','Сообщение          программы
Диздек.',mb_Ok+mb_iconstop);
         end;
        end else MessageBox(0,'Не введено число (десятичное или
шестнадцатеричное),  которое  надо  искать.'+#10+#13+'  Признак
шестнадцатеричного числа - буква h на конце.','Сообщение программы
Диздек.',mb_Ok+mb_iconstop);
       end;

       if RGSTZap.Proverka=2 then begin

       end;
      end;
      if RGSTZap.RD then sk:='_zuRD';

z:=TGSP(Screen.Forms[OknoPoImeni(Screen,'Дизассемблированная
программа')]).StringGrid1.RowCount-1;
      assignfile(fp,ChangefileExt(failv,'.dap'));
      reset(fp);

TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Items.
Clear;

TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Sorted
:=true;
      for i:=1 to z do begin
       read(fp,zpr);

s1:=IntToHex(ByteToLongword(zpr.a1,zpr.a2,zpr.a3,zpr.a4),8)+'h';
       StrCopy(proms,' ');
       StrCopy(proms,zpr.mat);
       StrCopy(mp,proms);
       if                VyborRegistra(0,sk)                then
TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Items.
Add(s1);
      end;
```

```
k:=TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Cou
nt;

TTSP(Screen.Forms[OknoPoImeni(Screen,'ПОИСК.')]).ListBox1.Items.
Add(' Найдено '+IntToStr(k)+' строк.');
       closefile(fp);
      end else MessageBox(0,'В окошке "Регистры (выбор дв.
щелчком)" окна "Общие параметры" не выбран регистр. Поиск
невозможен.','Сообщение программы
Диздек.',mb_Ok+mb_iconexclamation);
     end else MessageBox(0,'Не обнаружено окно с
дизассемблированной программой. Поиск невозможен.','Сообщение
программы Диздек.',mb_Ok+mb_iconexclamation);
    end;

    procedure TMainForm.N26Click(Sender: TObject);
    begin
     RGP.Showmodal;
     TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).LabelRGP.Caption:=failRGP;
     if (fileexists(failRGP))and(failv<>'') then begin
      Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,5,0);
      ZapRegis;
      ZapRegisPol;
     end;
    end;

    procedure TMainForm.N28Click(Sender: TObject);
    begin
     if          TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).LoadRGP.Execute then begin
      failRGP:=TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).LoadRGP.FileName;
      if pos('.rgp',failRGP)=0 then failRGP:=failRGP+'.rgp';
      TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).LabelRGP.Caption:=failRGP;
      if (fileexists(failRGP))and(failv<>'') then begin
       Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')].Perform(UM_Glav,5,0);
       ZapRegis;
       ZapRegisPol;
      end;
```

```
      end else begin
       failRGP:='';
       TMDIChild(Screen.Forms[OknoPoImeni(Screen,'Общие
параметры.')]).LabelRGP.Caption:=failRGP;
      end;
    end;

    procedure TMainForm.N29Click(Sender: TObject);
    begin
     Form2.Showmodal;
    end;

    end.
```

Приложение Б.Презентация

База данных универсального дизассемблера диздек архитектуры IA-32

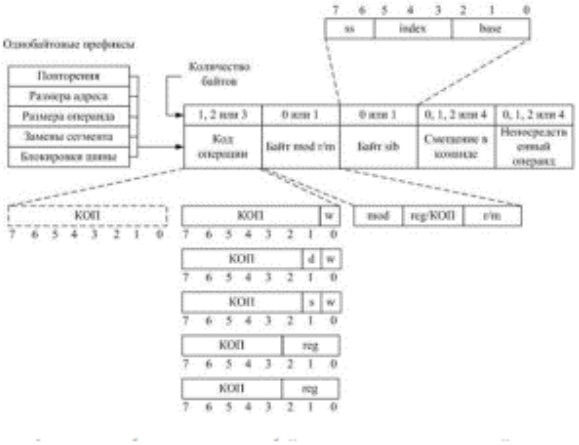
Автор работы: студент группы 16ВВ1 Щетинин К.А.
Научный руководитель: к.т.н. Кулагин О.В.

Пенза 2020

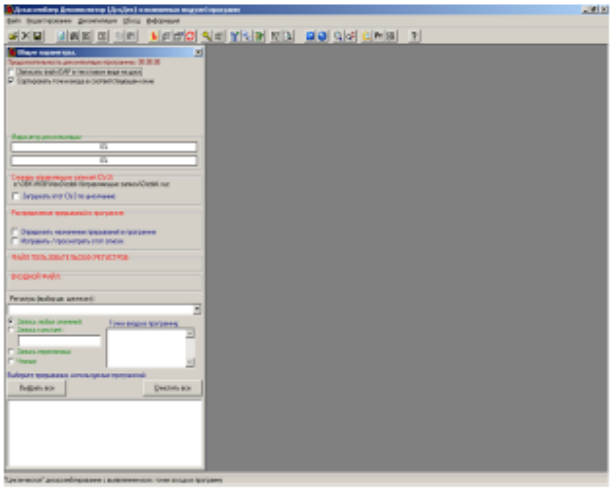
Цель и объекты работы

- Цель работы – разработать базу данных для приложения диздек.
- Объектом разработки является приложение, обеспечивающая дизассемблирование.
- В результате проведенной работы было разработано приложение, рещающее поставленную задачу.

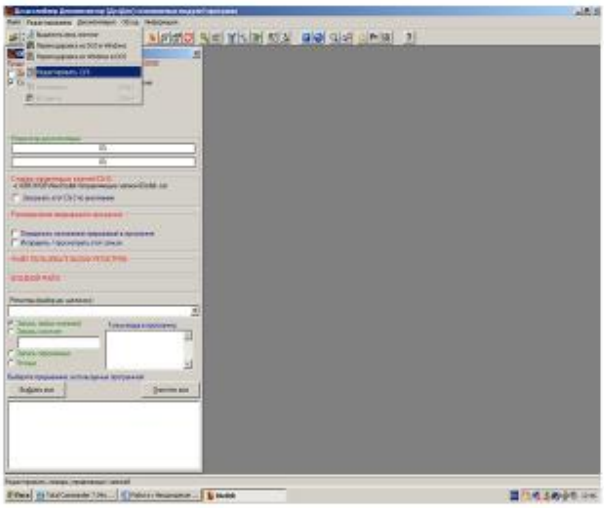
Формат команды в IA-32



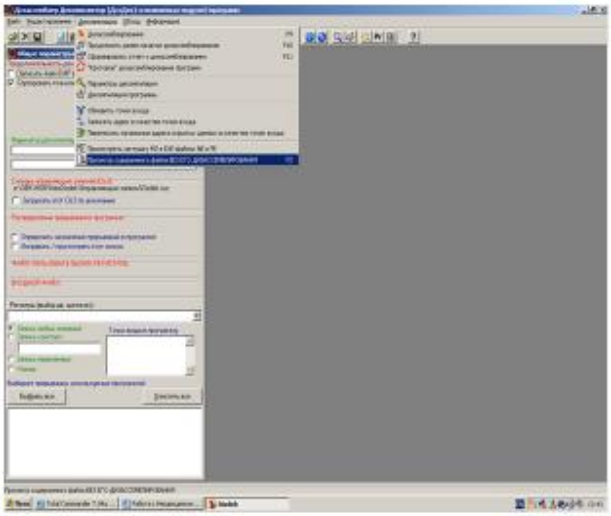
Основное окно программы Диздек.



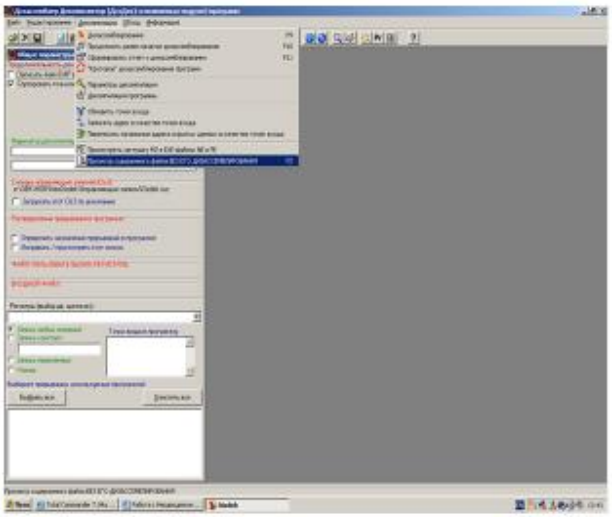
Подменю «Редактирование».



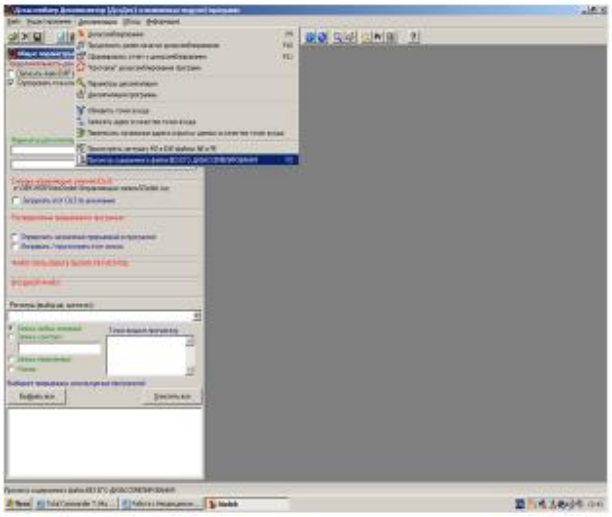
Подменю «Декомпиляция».



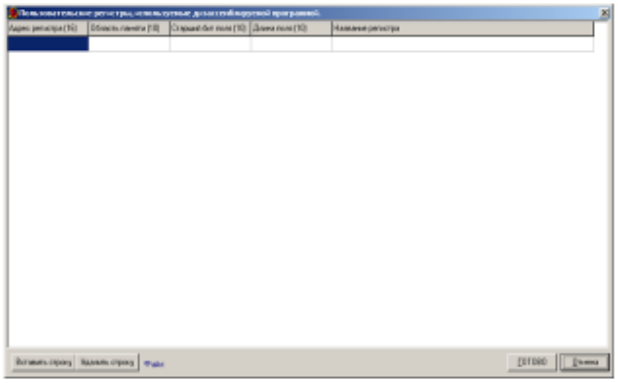
Подменю «Декомпиляция».



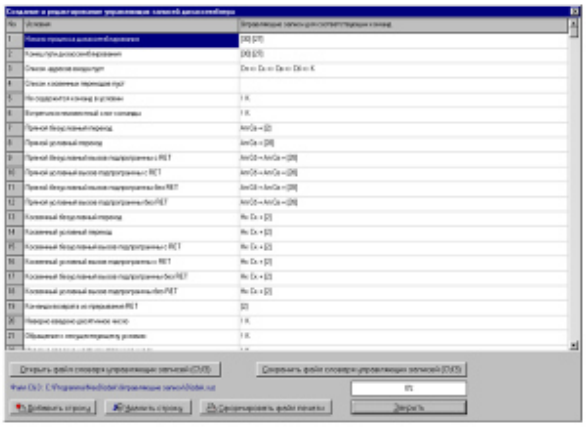
Подменю «Декомпиляция».



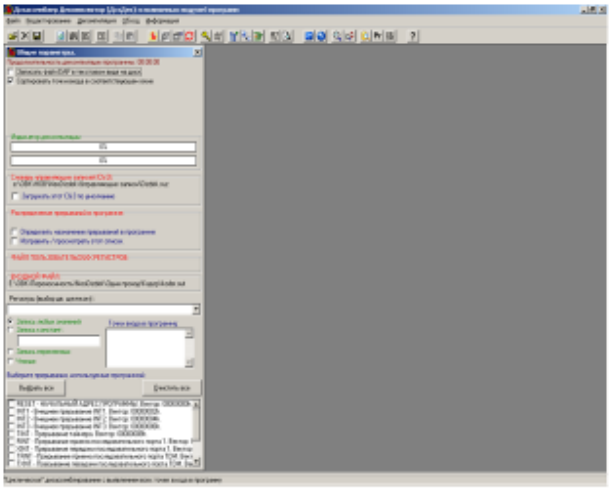
Окно для формирования списка пользовательских регистров.



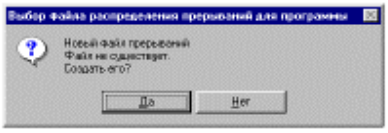
Окно редактирования СУЗ программы Диздек.



Основное окно Диздека с выбранным входным файлом и прерываниями.

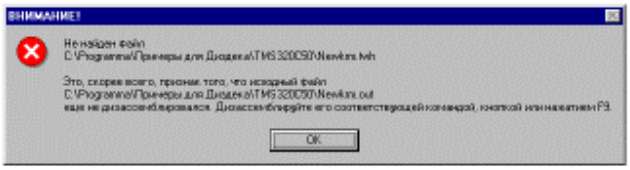


Окно, предлагающее создать файл с именем «Новый файл прерываний».

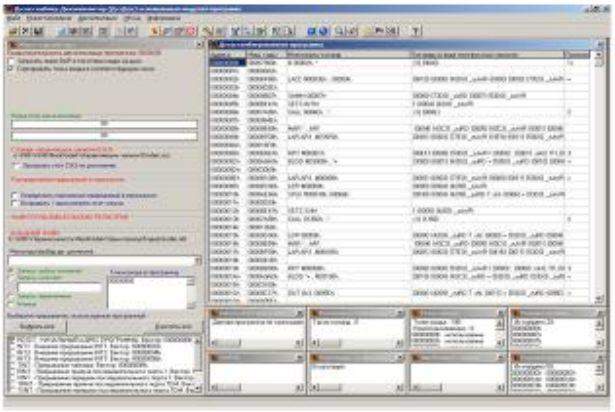


Окно редактирования назначений аппаратных прерываний.

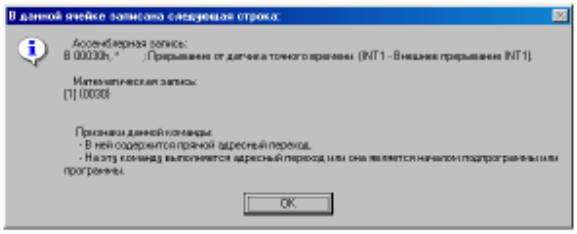
Окно, предупреждающее о невозможности продолжения дизассемблирования.



Вид программы по окончании процесса дизассемблирования.



Вид длинных описаний команды в отдельном окне.



Вывод

- В ходе данной работы было реализовано приложение для дизассемблирования, и были приобретены навыки работы с языком программирования Delphi. При реализации поставленной задачи были получены следующие результаты:
- проведён анализ современных способов и алгоритмов дизассемблирования;
- разработана база данных, позволяющая провести проверку на правильность дизассемблирования ;

Спасибо за внимание