


Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Информатика
Кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

 Рубан А.И.
подпись фамилия, инициалы

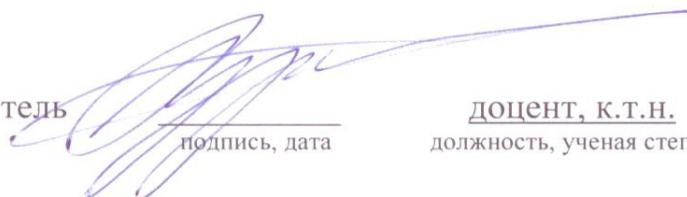
26 » сентября 2016 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

231000.62 Программная инженерия
код и наименование специальности

«Интегрированная обучающая среда разработки промежуточного
тема
представления программ»

Руководитель


подпись, дата

доцент, к.т.н.
должность, ученая степень

Кузнецов А.С.
фамилия, инициалы

Выпускник


подпись, дата

Берестюк А.В.
фамилия, инициалы

Нормоконтролер


подпись, дата

доцент, к.т.н.
должность, ученая степень

Антамошкин О.А.
фамилия, инициалы

Красноярск 2016

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Информатика
Кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

 Рубан А.И.
подпись фамилия, инициалы

«20» мая 2016 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

РЕФЕРАТ

Выпускная квалификационная работа по теме «Интегрированная обучающая среда разработки промежуточного представления программ» содержит 42 страницы текстового документа, 1 приложение, 21 иллюстрацию, 1 таблицу, 17 использованных источников.

ТРЕХАДРЕСНЫЙ КОД, БАЗОВЫЙ БЛОК, ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ, КОДОГЕНЕРАЦИЯ, СИМУЛЯТОР РАБОТЫ ПРОГРАММЫ, ТАБЛИЦА СИМВОЛОВ.

Целью работы является разработка интегрированной обучающей среды разработки промежуточного представления программ. Задачи: разработка графического интерфейса пользователя, включающего редактор кода с подсветкой синтаксиса, выделением базовых блоков, разработка подсистемы генерации ассемблерного кода целевой вычислительной системы, разработка симулятора для проверки корректности и работоспособности программы в трехадресном коде.

В работе приводится анализ проблемы проверки корректности программ в трехадресном коде при изучении промежуточного представления программ, а также, спецификация и описание сконструированной среды разработки.

Итогом работы является программная система, реализующая функции редактора кода с подсветкой синтаксиса, генерации кода ассемблера для целевых вычислительных платформ, симулятора выполнения программ в трехадресном коде, режима обучения. Система предназначена для использования в курсах разработки компиляторов при изучении промежуточного представления программ. В целом программная система отвечает поставленным требованиям, может использоваться в учебном процессе, а также обладает потенциалом для дальнейшего развития. Возможно расширение обучающего режима, добавление упражнений, добавление функций оптимизации кода на уровне базовых блоков и процедур, расширение функций редактора, отображение графа потока управления.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1 Описание предметной области	8
1.1 Анализ предметной области.....	8
1.1.1 Промежуточное представление программ.....	8
1.1.2 Трехадресный код.....	9
1.1.3 Проблема проверки корректности программ в трехадресном коде при изучении промежуточного представления.....	10
1.2 Общие требования к средам разработки программ.....	10
1.3 Существующие среды разработки программ.....	11
1.4 Цель и задачи выполняемой работы.....	13
1.5 Обобщенный вывод.....	13
1.5.1 Архитектура системы.....	13
1.5.2 Программно-аппаратные средства для реализации системы.....	14
1.5.3 Практическая применимость системы.....	15
2 Спецификация на разрабатываемое программное обеспечение	16
2.1 Техническое задание.....	16
2.1.1 Введение.....	16
2.1.2 Основание для разработки.....	16
2.1.3 Назначение разработки.....	16
2.1.4 Требования к программе.....	17
2.1.4.1 Требования к функциональным характеристикам.....	17
2.1.4.2 Требования к составу и параметрам технических средств.....	17
2.1.4.3 Требования к информационной и программной совместимости.....	17
2.1.4.4 Требования к способам поставки и комплектации.....	18
2.1.5 Требование к программной документации.....	18
2.1.6 Стадии и этапы разработки.....	18
2.1.7 Порядок контроля и приемки.....	19

3	Описание программной системы	20
3.1	Общие сведения.....	20
3.2	Функциональное назначение.....	20
3.3	Описание логической структуры.....	20
3.3.1	Модуль графического интерфейса пользователя.....	21
3.3.2	Модуль генерации ассемблерного кода.....	23
3.3.3	Модуль симулятора выполнения программы в трехадресном коде.....	23
3.4	Используемые технические средства.....	24
3.5	Вызов и загрузка.....	24
3.6	Интерфейс пользователя.....	24
3.7	Комплект поставки программной системы.....	33
3.8	Лицензирование программной системы.....	34
4	Руководство пользователя	35
4.1	Общие сведения о программном продукте.....	35
4.2	Описание установки.....	35
4.3	Описание запуска.....	35
4.4	Инструкция по работе.....	36
5	Руководство системного программиста	38
5.1	Общие сведения о программном продукте.....	38
5.1.1	Назначение и функции программы.....	38
5.1.2	Необходимы ресурсы.....	38
5.2	Комплект поставки и установки.....	38
5.3	Описание запуска.....	39
5.4	Инструкция по работе.....	39
	ЗАКЛЮЧЕНИЕ	40
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	41
	ПРИЛОЖЕНИЕ А Методическое пособие. Фрагмент (раздел 2.2 Базовые блоки)	43

ВВЕДЕНИЕ

В процессе трансляции происходит генерация промежуточного представления исходной программы, предназначенная, прежде всего, для удобства генерации кода и/или проведения различных оптимизаций. Сама форма промежуточного представления зависит от целей его использования. Одной из наиболее часто используемых форм является трехадресный код.

При изучении компиляторных курсов, и, в частности, темы промежуточного представления исходных текстов, также рассматривается генерация трёхадресного кода, однако его корректность декларируется «как есть». Это обусловлено тем, что некоторые его конструкции недостаточно формализованы. С целью устранения данного недостатка можно использовать симулятор, позволяющий проверять корректность и работоспособность полученных инструкций данного промежуточного представления.

К настоящему моменту получена реализация симулятора, который не поддерживает некоторые «стандартные» трехадресные инструкции, и работает только в терминальном режиме, не позволяя проводить полноценную проверку корректности программ. Также, в связи с режимом работы симулятора, отсутствует наглядное представление программы и возможность «удобного» редактирования кода. Таким образом актуальность выполняемой работы не вызывает сомнений.

Целью данной работы является разработка интегрированной обучающей среды разработки промежуточного представления программ.

Для достижения поставленной цели необходимо решить следующие задачи:

- разработка графического интерфейса пользователя, включающего редактор кода с подсветкой синтаксиса, выделением базовых блоков;
- разработка подсистемы генерации ассемблерного кода целевой вычислительной системы;
- разработка симулятора для проверки корректности и работоспособности

программы в трехадресном коде.

1 Описание предметной области

1.1 Анализ предметной области

1.1.1 Промежуточное представление программ

В процессе трансляции происходит генерация промежуточного представления исходной программы, предназначенная, прежде всего, для удобства генерации кода и/или проведения различных оптимизаций. Сама форма промежуточного представления зависит от целей его использования. На рисунке 1 показана схема типичного компилятора [1].

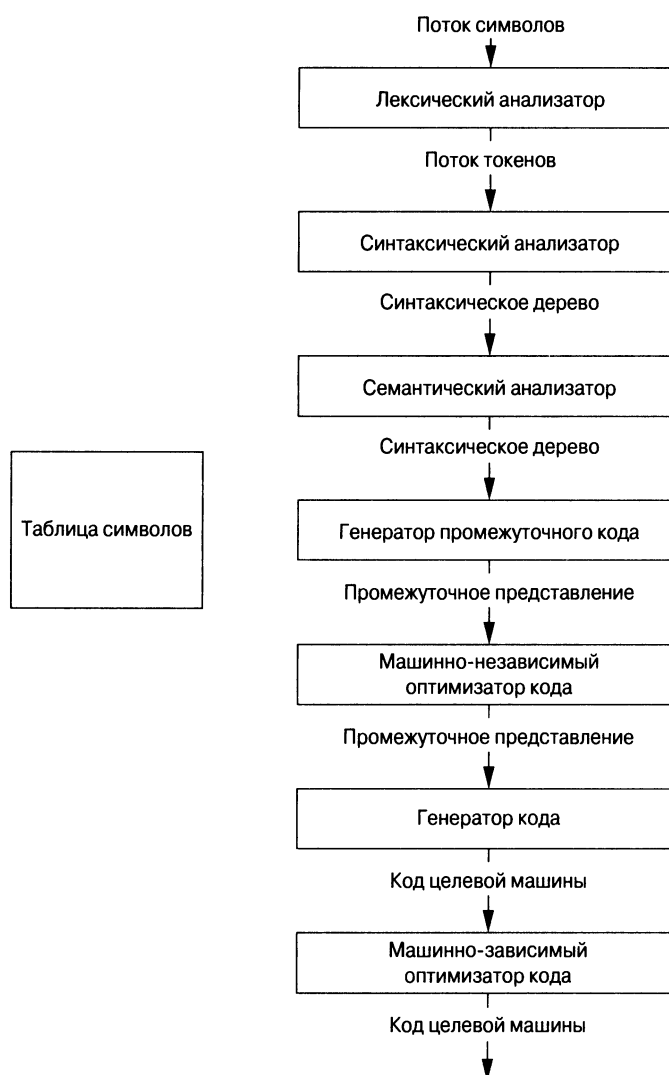


Рисунок 1 – Фазы компилятора

Одной из наиболее часто используемых форм промежуточного представления программ является трехадресный код.

1.1.2 Трехадресный код

Трехадресный код построен на основе двух концепций: адресов и команд.

В трехадресном коде в правой части команды имеется не более одного оператора, т.е. не допускаются никакие встроенные арифметические выражения. Таким образом, выражение на исходном языке наподобие $x+y*z$ может быть транслировано в трехадресные команды

$$t1 = y * z$$
$$t2 = x + t1$$

Здесь $t1$ и $t2$ — временные имена, сгенерированные компилятором. Такая развертка многооператорных арифметических операций и вложенных инструкций управления потоком делает трехадресный код особенно подходящим для генерации целевого кода и оптимизации. Использование имен для промежуточных значений, вычисляемых программой, позволяет легко выполнять перестановки в трехадресном коде.

Адрес может быть одним из следующего списка.

- Имя. Для удобства мы позволяем именам исходной программы появляться в трехадресном коде в виде адресов. При реализации исходное имя заменяется указателем на запись в таблице символов, в которой хранится вся информация об имени.

- Константа. На практике компилятор должен работать со многими различными типами констант и переменных.

- Генерируемые компилятором временные переменные. Оказывается полезным, в особенности в оптимизирующих компиляторах, создание имен, отличающихся друг от друга, всякий раз, когда требуется временная переменная. Эти временные переменные могут комбинироваться при наличии

такой возможности в процессе назначения переменным регистров и памяти [1].

1.1.3 Проблема проверки корректности программ в трехадресном коде при изучении промежуточного представления

При изучении компиляторных курсов, и, в частности, темы промежуточного представления исходных текстов, также рассматривается генерация трёхадресного кода, однако его корректность декларируется «как есть». Это обусловлено тем, что некоторые его конструкции недостаточно формализованы. С целью устранения данного недостатка можно использовать симулятор, позволяющий проверять корректность и работоспособность полученных инструкций данного промежуточного представления. Также необходимо обеспечить наглядность и удобство редактирования программ в трехадресном коде, что позволит повысить качество обучения, даст возможность контроля над промежуточными результатами разрабатываемого в рамках одноименных дисциплин компилятора.

Данные задачи можно решить двумя путями:

а) Доработка существующего программного продукта для решения поставленных задач. Программный продукт, при этом должен предоставлять возможность расширения своего функционала.

б) Разработка программной системы «с нуля». В этом случае, необходимо разрабатывать все необходимые модули системы, выбор технологий, инструментов зависит от решаемых задач.

1.2 Общие требования к средам разработки программ

На данный момент требования к интегрированным средам разработки не формализованы, поэтому ниже приведены требования, характерные для структуры типичных систем данного типа:

- расширяемость;
- наличие редактора кода;

- подсветка синтаксиса;
- мультиязычность;
- наличие компилятора;
- поддержка нескольких целевых платформ;
- наличие подсистемы автоматизированной сборки;
- наличие отладчика.

Дополнительные требования:

- кроссплатформенность.

В данном случае, наличие подсистемы автоматизированной сборки и отладчика не требуется. Вместо этого требуется реализация симулятора для выполнения программы, конструируемой в среде разработки, мультиязычность реализуется поддержкой разных «диалектов» трехадресного кода.

1.3 Существующие среды разработки программ

В процессе анализа были рассмотрены существующие среды разработки, предоставляющие возможность подключения пользовательских расширений.

Кроме того, к настоящему моменту получена реализация симулятора, который не поддерживает некоторые «стандартные» трехадресные инструкции, и работает только в терминальном режиме. Также, в связи с режимом работы симулятора, отсутствует наглядное представление программы и возможность «удобного» редактирования кода.

В таблице 1 приведено сравнение основных кандидатов для доработки – сред разработки и симулятора с точки зрения их использования для решения поставленных задач.

Таблица 1 – Сравнение возможностей сред разработки

Основные возможности	Visual Studio 2013/2015 Community Edition	Eclipse [5]	Code::Blocks[3]	Симулятор*
Возможность подключения пользовательских модулей расширения	да	да	да	нет
Наличие редактора кода	да	да	да	нет
Подсветка синтаксиса	да	да	да	нет
Основные поддерживаемые языки программирования	C, C++, C#, F#, Python (Visual Studio 2015)	C, C++, Java	C, C++	трехадресный код
Компиляция в реальном времени	нет	нет	нет	нет
Поддержка нескольких целевых платформ	да	да	да	нет
Кроссплатформенность	нет	да	да	нет
* Симулятор не является средой разработки, представлен с целью иллюстрации возможностей.				

Все рассмотренные среды, за исключением симулятора, имеют редактор исходного кода, подсветку синтаксиса языков, компилятор. В дополнение, Eclipse имеет возможность написания и подключения синтаксического анализатора пользовательского языка. Но, несмотря на наличие компилятора, данные среды ориентированы на компиляцию программ «после» их написания, поэтому, модуль генерации кода ассемблера «на лету» нуждается в разработке. Также, в случае использования указанных сред, требуется написание либо адаптация симулятора трехадресного кода. Кроме того недостатком рассматриваемых программных систем, в данном случае, является их многофункциональность, «загруженность» интерфейса, создавая, своего рода, «информационный шум», что может повредить эффективности процесса обучения. На основании данной информации, было принято решение о

разработке программной системы «с нуля» и возможной интеграции симулятора.

Таким образом актуальность выполняемой работы не вызывает сомнений.

1.4 Цель и задачи выполняемой работы

Целью данной работы является разработка интегрированной обучающей среды разработки промежуточного представления программ.

Для достижения поставленной цели необходимо решить следующие задачи:

- разработка графического интерфейса пользователя, включающего редактор кода с подсветкой синтаксиса, выделением базовых блоков;
- разработка подсистемы генерации ассемблерного кода целевой вычислительной системы;
- разработка симулятора для проверки корректности и работоспособности программы в трехадресном коде.

1.5 Обобщенный вывод

В результате анализа предметной области было сделано заключение о возможности создания системы существующими программно-аппаратными средствами.

1.5.1 Архитектура системы

Для реализации системы была выбрана модульная архитектура. Она позволяет разрабатывать, поддерживать и расширять подсистемы практически независимо друг от друга. Основные модули системы представлены ниже:

- модуль графического интерфейса пользователя;
- модуль генерации ассемблерного кода целевой вычислительной

системы;

- модуль симулятора для проверки корректности и работоспособности программы в трехадресном коде.

1.5.2 Программно-аппаратные средства для реализации системы

Базовый язык реализации – C++.

Генерация ассемблерного кода требует интеграции с какой-либо существующей компиляторной инфраструктурой. Рассматривались свободно распространяемые инфраструктуры gcc [6], LLVM [14], Parrot VM [11]. Выбор сделан в пользу LLVM, поскольку данная инфраструктура обладает наиболее широким набором компиляторов для целевых вычислительных платформ.

Инструменты проектирования – Sparx Enterprise Architect 10.0 [13].

Версии компиляторов – mingw32-g++ 4.8.1 [10], g++ 4.8.1, llvm compilers 3.8.0.

Система сборки – CMAKE [2].

Средства документирования:

- doxygen - документирование программы [4];

- Libre Office Writer, Microsoft Word, Excel – техническая и отчетная документация.

Среда написания исходного кода – Code::Blocks 13.12 [3].

Хранилище исходного текстов – локальный репозиторий Git [7];

Система контроля версий – Git;

Библиотека графического интерфейса пользователя – wxWidgets [15];

Требование кроссплатформенности обеспечивается генерацией проектов для сред разработки, специфичных для целевых платформ, с последующей сборкой. Целевые платформы:

- Mac OS X (начиная с версии 10.5.0), среда XCode [16];

- GNU Linux i686, среда Code::Blocks 13.12;

- Windows (начиная с версии 7), среда Visual Studio 2013.

1.5.3 Практическая применимость системы

Система может использоваться при изучении компиляторных курсов, для изучения и наглядного отображения промежуточного представления программ (трехдресного кода), а так же для проверки корректности программ в трехдресном коде.

2 Спецификация на разрабатываемое программное обеспечение

Нормативные акты, использованные при написании технического задания:

- ЕСПД ГОСТ 19.201-78 [Техническое задание, требования к содержанию и оформлению](#).

2.1 Техническое задание

2.1.1 Введение

Основная цель – разработка программного приложения в соответствии с функциональными и иными требованиями, а также составление по результатам проектирования отчета. Полное наименование программного приложения: «Интегрированная обучающая среда разработки промежуточного представления программ». Область применения: разработка программ в трехадресном коде в процессе изучения промежуточного представления программ.

2.1.2 Основание для разработки

Задание выпускающей кафедры. Тема разработки: «Интегрированная обучающая среда разработки промежуточного представления программ»

2.1.3 Назначение разработки

Требуется разработать кроссплатформенное приложение, реализующее среду разработки программ в трехадресном коде, генерирующее ассемблерный код целевой вычислительной платформы, проверяющее корректность и

работоспособность разработанной программы.

2.1.4 Требования к программе

2.1.4.1 Требования к функциональным характеристикам

Разработать приложение, реализующее среду разработки программ в трехадресном коде. Приложение должно соответствовать следующим требованиям.

Должна быть предоставлена возможность:

а) редактирования программ в трехадресном коде с подсветкой синтаксиса и другими полезными функциями (в частности, выделение базовых блоков);

б) запуска программы в симуляторе по аналогии с современными средами разработки с выдачей результатов и сообщений об ошибках;

в) поддержки разных диалектов трехадресного кода;

г) генерации целевого кода «на лету», т.е. при модификации трехадресного кода в одном окне должен изменяться ассемблерный код целевой вычислительной платформы в другом окне.

Также должна быть обеспечена кроссплатформенность приложения, то есть возможность запуска на целевых платформах:

- Mac OS X (начиная с версии 10.5.0);
- GNU Linux i686;
- Windows (начиная с версии 7).

2.1.4.2 Требования к составу и параметрам технических средств

Особых требований к техническим средствам не предъявляется.

2.1.4.3 Требования к информационной и программной совместимости

Для запуска приложения требуется установленная ОС: Mac OS X (начиная с версии 10.5.0), либо GNU Linux i686, либо Windows (начиная с версии 7).

2.1.4.4 Требования к способам поставки и комплектации

Способ поставки программного приложения или программной системы – в открытых исходных текстах (с соблюдением требований лицензирования третьесторонних программных модулей).

В состав комплекта поставки должны входить все файлы, необходимые для нормального функционирования.

2.1.5 Требование к программной документации

Сопроводительная документация должна состоять из нескольких разделов. Обязательные разделы:

- Техническое задание;
- Описание программы;
- Руководство пользователя;
- Руководство системного программиста.

2.1.6 Стадии и этапы разработки

Стадии и этапы разработки программы проводить в соответствии с ЕСПД ГОСТ 19.102-77, включающие в себя следующие стадии:

- Техническое задание;
- Эскизный проект;
- Технический проект;
- Рабочий проект;
- Внедрение.

Допускается исключить вторую стадию разработки (Эскизный проект).

Все этапы работ, включенные в состав вышеуказанных стадий разработки, допускается объединять, исключать и изменять их содержание, а также вводить другие этапы работ в технически обоснованных случаях.

2.1.7 Порядок контроля и приемки

Программный продукт считается завершенным только после проверки приложения на работоспособность и полное соответствие функциональным и иными требованиям, а также при наличии полного комплекта программной документации.

Программа и документация передаются заказчику после выполнения всех работ.

3 Описание программной системы

3.1 Общие сведения

Наименование программы – «Интегрированная обучающая среда разработки промежуточного представления программ».

Для функционирования программы на персональном компьютере должна быть установлена ОС Windows 7 или выше, либо ОС GNU Linux i686 или выше, либо ОС Mac OS X версии 10.5.0 или выше.

Программа написана на языке программирования C++, с использованием библиотеки графического интерфейса wxWidgets, компиляторов LLVM.

3.2 Функциональное назначение

Разработка программ в трехадресном коде в процессе изучения промежуточного представления программ.

3.3 Описание логической структуры

Общая архитектура программной системы отражена на рисунке 2.

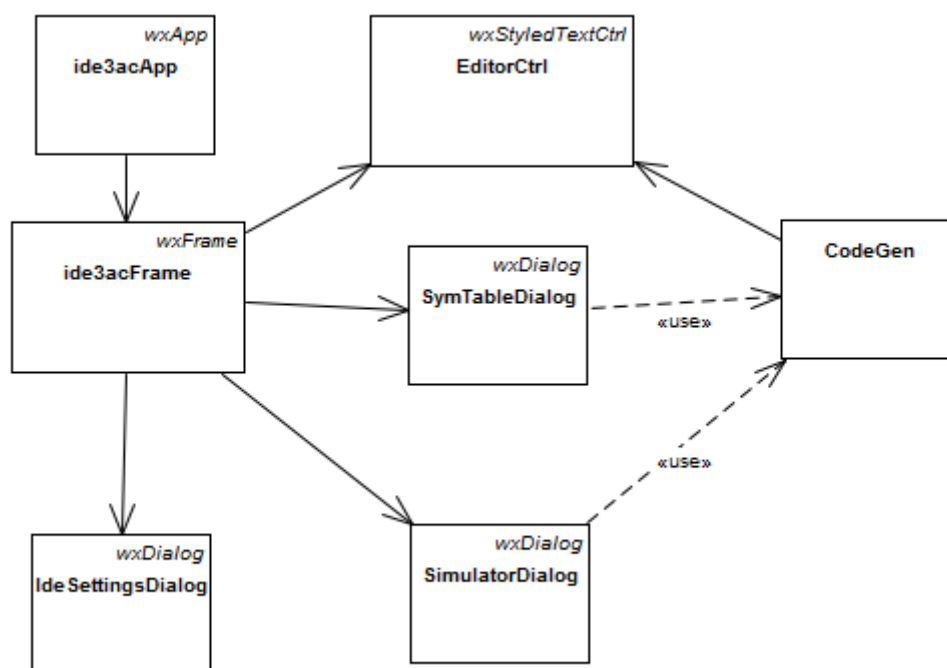


Рисунок 2 – Диаграмма классов программной системы

Программа разделена на модули согласно их функциональному назначению.

3.3.1 Модуль графического интерфейса пользователя

Классы `ide3acFrame`, `SymTableDialog`, `IdeSettingsDialog`, `EditorCtrl`.

Класс `ide3acFrame` - реализует главное окно программы. Обеспечивает посредством меню и панели быстрого доступа (Toolbar), доступ к функциональным возможностям программы, таким как: выбор целевой вычислительной платформы, редактирование таблицы символов, редактирование настроек приложения, запуск редактируемой программы в симуляторе. Также предоставляет стандартные функции редактирования документа, загрузку и сохранение файла исходного кода и т.д., обеспечивает информирование пользователя о совершенных действиях посредством строки статуса (Statusbar), отображает сгенерированный ассемблерный код, отображает информационные и сообщения об ошибках.

Класс `EditorCtrl` - реализует окно редактирования исходного кода

разрабатываемой программы в трехадресном коде. Обеспечивает функции редактирования файла с исходным кодом, стандартные функции редактирования документа, реализует подсветку синтаксиса трехадресного кода, выделение и подсветку базовых блоков. Базовый блок – максимальная последовательность следующих друг за другом трехадресных команд, обладающая следующими свойствами:

- поток управления может входить в блок только через первую инструкцию блока;

- управление покидает блок без останова и ветвления, возможное исключение – последняя инструкция в блоке. Базовые блоки становятся узлами графа потока управления, дуги которого показывают порядок следования блоков.

Реализация механизма подсветки синтаксиса кода основана на wxScintilla – адаптированному для wxWidgets свободно распространяемому компоненту Scintilla [12], компоненту обеспечения редактирования исходных кодов. В качестве контейнера использовался компонент wxStyledTextCtrl, входящий в библиотеку wxWidgets. Для поддержки синтаксиса трехадресного кода реализован лексический анализатор языка трехадресного кода. С целью уменьшения времени, необходимого для обновления текста в окне редактора, обрабатывается только область измененного текста. Также, анализатор фиксирует и сохраняет номера строк лидеров – первых инструкций базовых блоков, после завершения стадии подсветки синтаксиса выполняется проход по строкам текста программы, инструкция-лидер каждого базового блока помечается латинским символом «B» в области - слева от текста. Далее, для лучшего восприятия, фон блоков окрашивается двумя цветами в чередующемся порядке.

Класс SymTableDialog - реализует окно редактирования таблицы символов разрабатываемой программы в трехадресном коде. Обеспечивает установку и выбор типа и размера (типа) символов (переменных).

Класс IdeSettingsDialog - реализует окно редактирования настроек

программы. Обеспечивает настройку шрифта, палитры основных и фоновых цветов для подсветки элементов синтаксиса языка (ключевых слов, идентификаторов, меток, числовых и строковых констант), базовых цветов текста и фона редактора.

3.3.2 Модуль генерации ассемблерного кода

Модуль генерации ассемблерного кода целевой вычислительной платформы представлен классом CodeGen.

Обеспечивает трансляцию трехадресного кода в представление IR [8] с последующей генерацией ассемблерного кода (посредством компилятора LLVM) целевой вычислительной платформы.

Хранит внутренне представление трехадресной программы в виде четверок. Четверки – форма представления трехадресного кода в виде записи, имеет четыре поля с именами op, arg1, arg2, result. Поле op содержит внутренний код оператора. Например, трехадресная команда $x = y + z$ представлена путем размещения + в op, y в arg1, z в arg2, x в result.

После проведения операций обновления стилей текста (подсветка синтаксиса) и выделения базовых блоков, метод OnStyleNeeded() класса EditorCtrl вызывает метод CodeGen::parse(), который, в данной реализации, проводит синтаксический анализ разрабатываемой программы, заполняет внутреннее представление программы, таблицу символов, вызывает метод CodeGen::generate(), который осуществляет генерацию IR представления, кода ассемблера целевой вычислительной платформы.

3.3.3 Модуль симулятора выполнения программы в трехадресном коде

Модуль симулятора выполнения программы в трехадресном коде реализован в классе SimulatorDialog.

Осуществляет выполнение программы в трехадресном коде.

Обеспечивает проверку корректности и работоспособности программы.

Хранит размещение переменных в виртуальном адресном пространстве, единица адресации принята равной одному байту. Отображает список символов, их адреса и значения на каждом шаге выполнения программы.

Успешное завершение выполнения программы свидетельствует о корректности программы. В случае ошибки, выводится соответствующее сообщение.

3.4 Используемые технические средства

Для работы программы необходим IBM-совместимый компьютер с установленной ОС Windows 7 или выше, либо ОС GNU Linux i686 или выше, либо ОС Mac OS X версии 10.5.0 или выше.

3.5 Вызов и загрузка

Вызов программы производится:

ОС Windows: путем запуска запускаемого файла ide3ac.exe.

ОС GNU Linux i686: путем запуска запускаемого файла ide3ac.

ОС Mac OS X: путем запуска запускаемого файла ide3ac.

3.6 Интерфейс пользователя

Данные, запрашиваемые программой в процессе выполнения, и результаты работы программы обрабатываются посредством графического интерфейса пользователя.

При запуске программы отображается главное окно программы (рисунок 3).

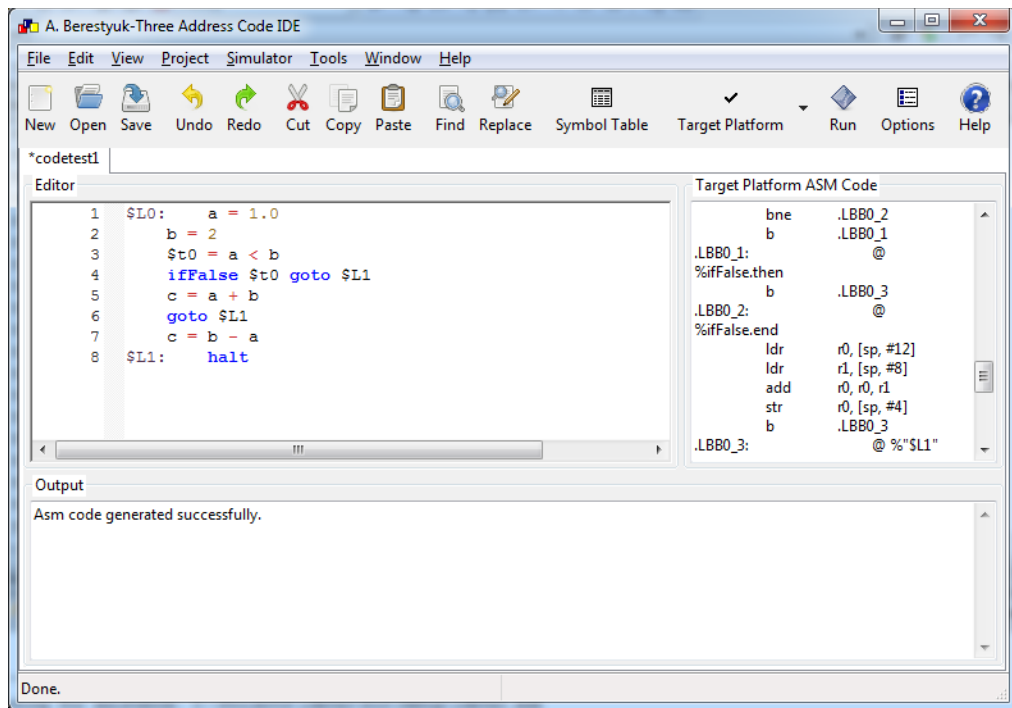


Рисунок 3 – Главное окно программы

Основные элементы главного окна:

Заголовок окна – содержит системное меню, заголовок программы, системные кнопки управления программой.

Главное меню (рисунок 4) – содержит разделы меню, обеспечивающие доступ к функциям программы:

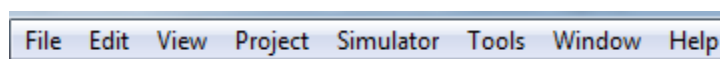


Рисунок 4 – Главное меню программы

Раздел меню «File» (рисунок 5) - содержит пункты, поддерживающие операции с файлами:

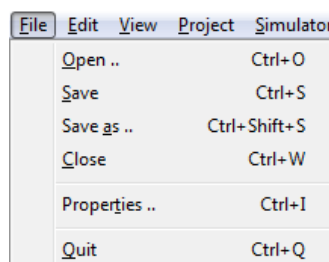


Рисунок 5 – Раздел меню «File»

- «New» - создать новый файл.
- «Open» - открыть диалоговое окно «Открытие файла».
- «Save» - сохранить файл.
- «Save as...» - открыть диалоговое окно «Сохранение файла».
- «Close» - закрыть файл.
- «Properties» - открыть диалоговое окно «Свойства файла» с информацией о файле.
- «Exit» - выйти из программы.

Раздел меню «Edit» (рисунок 6) - содержит пункты, поддерживающие операции редактирования файла:

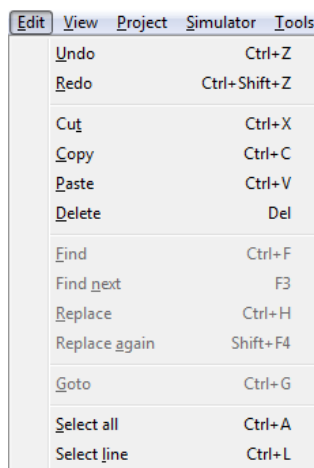


Рисунок 6 – Раздел меню «Edit»

- «Undo» - отменить последнее совершенное действие.
- «Redo» - повторить последнее отмененное действие.
- «Cut» - «вырезать» выделенный диапазон текста в буфер обмена.
- «Copy» - скопировать выделенный диапазон текста в буфер обмена.
- «Paste» - вставить содержимое буфера обмена в текст, начиная с позиции курсора.
- «Delete» - удалить выделенный диапазон текста.
- «Find» - открыть диалоговое окно «Поиск» для поиска необходимого фрагмента текста в файле.

- «Find next» - произвести повторный поиск фрагмента текста в файле.
- «Replace» - открыть диалоговое окно «Поиск и замена» для поиска и замены необходимого фрагмента текста в файле.
- «Find next» - произвести повторный поиск и замену фрагмента текста в файле.
- «Goto» - открыть диалоговое окно «Переход» для перехода на необходимую позицию в тексте файла.
- «Select all» - выделить все содержимое файла.
- «Select line» - выделить строку текста, на которой находится курсор.

Раздел меню «View» (рисунок 7) - содержит пункты, обеспечивающие управление внешним видом интерфейса программы:

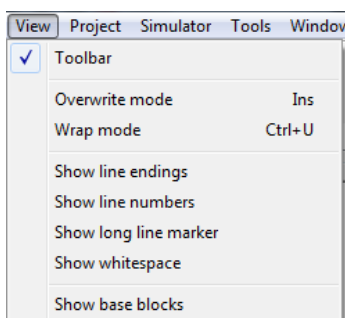


Рисунок 7 – Раздел меню «View»

- «Toolbar toggle» - включить/выключить отображение панели быстрого доступа.
- «Overwrite mode» - включить/выключить режим редактирования текста «перезаписью».
- «Wrap mode» - включить/выключить перенос слов.
- «Show line endings» - включить/выключить отображение символов конца строки.
- «Show line numbers» - включить/выключить отображение номеров строк.
- «Show long line marker» - включить/выключить отображение маркера границы страницы.
- «Show whitespace» - включить/выключить отображение пробельных

СИМВОЛОВ.

- «Show basic blocks» - включить/выключить отображение базовых блоков.

Раздел меню «Project» (рисунок 8) - содержит пункты, относящиеся к текущему редактируемому файлу:

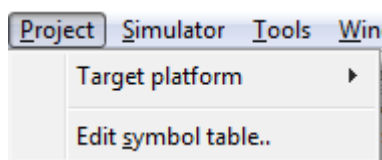


Рисунок 8 – Раздел меню «Project»

- «Edit symbol table» - открыть диалоговое окно редактирования таблицы символов редактируемой программы.

Подраздел меню «Target platform» (рисунок 9) - содержит пункты, обеспечивающие выбор целевой вычислительной платформы:

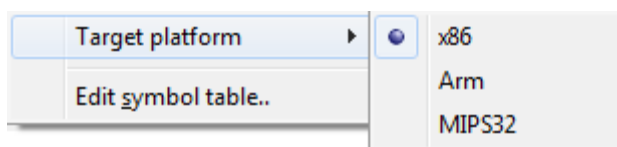


Рисунок 9 – Подраздел меню «Target platform»

- «x86» - выбрать целевую вычислительную платформу x86.

- «Arm» - выбрать целевую вычислительную платформу arm.

- «MIPS32» - выбрать целевую вычислительную платформу MIPS32.

Раздел меню «Simulator» (рисунок 10) - содержит пункты, обеспечивающие функции, относящиеся к симулятору работы программы в трехадресном коде:

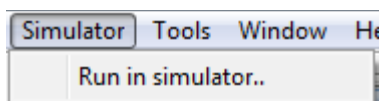


Рисунок 10 – Раздел меню «Simulator»

- «Run in simulator» - запустить редактируемую программу в симуляторе.

Раздел меню «Tools» (рисунок 11) - содержит пункты, обеспечивающие доступ к настройкам и дополнительным функциям программы:

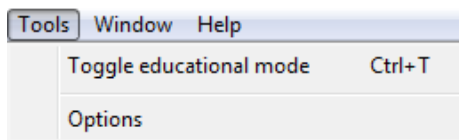


Рисунок 11 – Раздел меню «Tools»

- «Options» - открыть диалоговое окно редактирования настроек программы.

- «Toggle educational mode» - включить/выключить режим обучения.

Раздел меню «Window» (рисунок 12) - содержит пункты, обеспечивающие доступ к открытым файлам с программами в трехадресном коде:

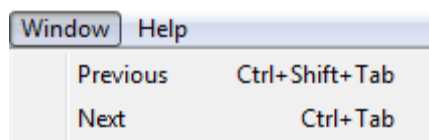


Рисунок 12 – Раздел меню «Window»

- «Previous» - активировать предыдущее окно редактирования программы.

- «Next» - активировать следующее окно редактирования программы.

Раздел меню «Help» (рисунок 13) - содержит пункты, обеспечивающие доступ к справке и информации о программе:

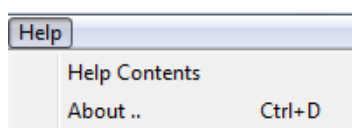


Рисунок 13 – Раздел меню «Help»

- «Help Contents» - открыть окно справки программы.

- «About...» - открыть окно с информацией о программе.

«Панель быстрого доступа» (рисунок 14) - обеспечивает быстрый доступ к часто используемым функциям программы, дублирует пункты меню, соответственно: «New», «Open», «Save», «Undo», «Redo», «Cut», «Copy», «Paste», «Find», «Replace», «Symbol Table», «Target Platform», «Run», «Options», «Help».

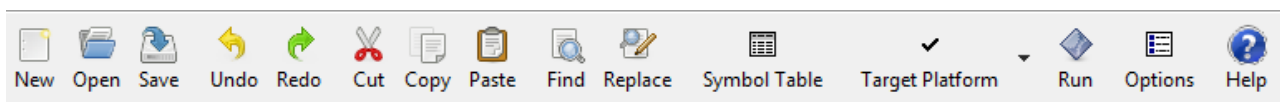
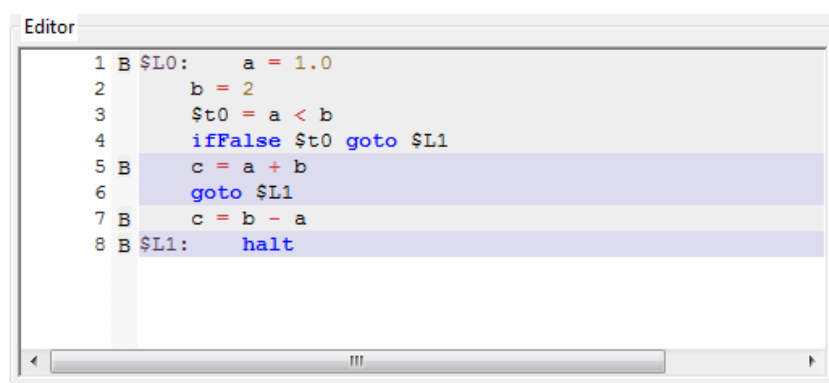


Рисунок 14 – Панель быстрого доступа

Все вышеперечисленные объекты являются кнопками, кроме «Target Platform», данный объект представляет собой раскрывающийся список, содержащий строки с наименованиями целевых вычислительных платформ, эквивалентный содержимому подраздела меню «Target platform».

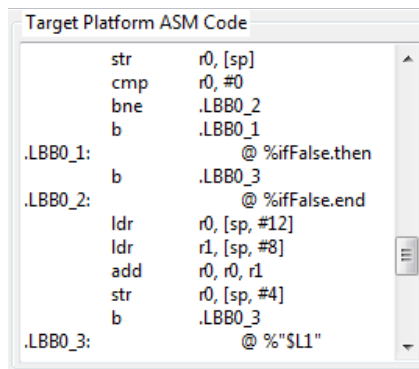
Область редактора кода «Editor» (рисунок 15) – обеспечивает отображение, редактирование программы в трехадресном коде.



```
1 B $L0:  a = 1.0
2      b = 2
3      $t0 = a < b
4      ifFalse $t0 goto $L1
5 B     c = a + b
6      goto $L1
7 B     c = b - a
8 B $L1:  halt
```

Рисунок 15 – Область редактора кода с выделенными базовыми блоками

Область кода целевой вычислительной платформы «Target Platform ASM Code» (рисунок 16) – обеспечивает отображение полученного кода ассемблера целевой вычислительной платформы.



```
Target Platform ASM Code
str    r0, [sp]
cmp    r0, #0
bne    .LBB0_2
b      .LBB0_1
.LBB0_1:      @ %ifFalse.then
b      .LBB0_3
.LBB0_2:      @ %ifFalse.end
ldr    r0, [sp, #12]
ldr    r1, [sp, #8]
add    r0, r0, r1
str    r0, [sp, #4]
b      .LBB0_3
.LBB0_3:      @ %"SL1"
```

Рисунок 16 – Область кода целевой вычислительной платформы

Область вывода сообщений системы «Output» (рисунок 17) – обеспечивает отображение сообщений системы.

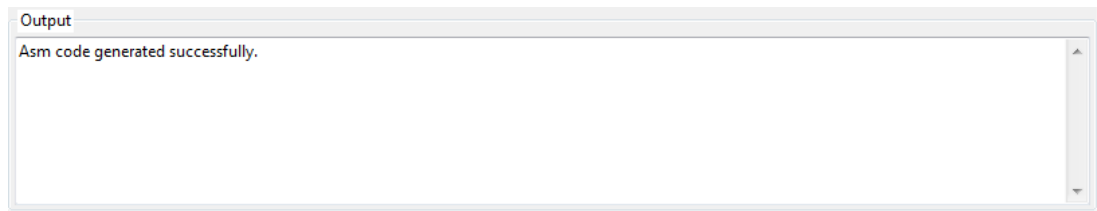


Рисунок 17 – Область вывода сообщений системы

Строка статуса (рисунок 18) – отображает приветственное сообщение, сообщение программы о последнем произведенном действии.



Рисунок 18 – Строка статуса

При выборе пункта меню «Edit symbol table» отображается диалоговое окно редактирования таблицы символов (рисунок 19).

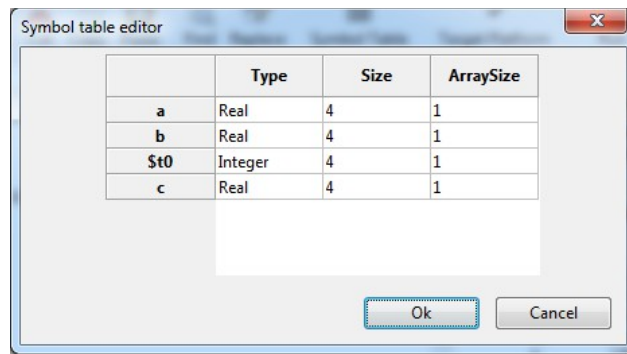


Рисунок 19 – Диалоговое окно редактора таблицы символов

Редактор таблицы символов обеспечивает отображение и функции редактирования типа, размера символов программы.

При выборе пункта меню «Run in simulator» отображается диалоговое окно симулятора выполнения программы в трехадресном коде (рисунок 20).

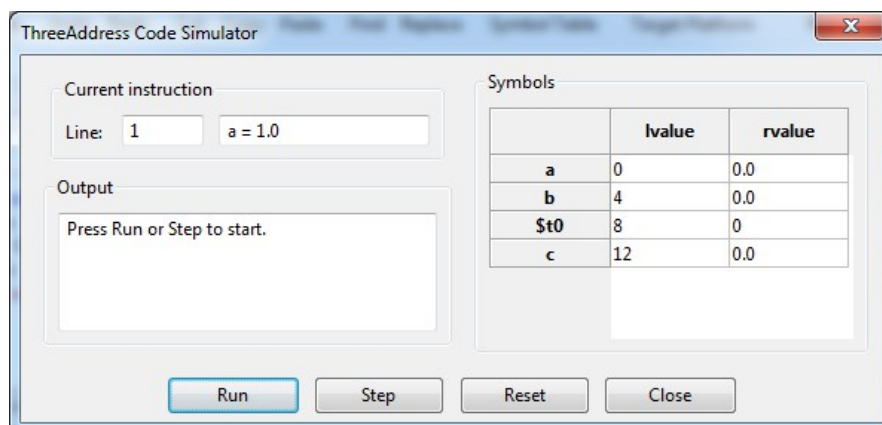


Рисунок 20 – Диалоговое окно симулятора выполнения программы

Отображает список символов, их адреса и значения на каждом шаге выполнения программы, поле вывода сообщений, элементы управления.

Успешное завершение выполнения программы свидетельствует о корректности программы. В случае ошибки, в область «Output» выводится соответствующее сообщение.

Содержит элементы управления:

- кнопка «Run» - выполнение программы;
- кнопка «Step» - выполнение текущей инструкции и отображение следующей;

- кнопка «Reset» - сброс состояния симулятора, атрибутов символов к начальным значениям.

- кнопка «Close» - закрытие окна симулятора.

В программе предусмотрен режим обучения, при активации которого, в области редактора демонстрируется пример с пояснениями по тематике генерации промежуточного кода (рисунок 21).

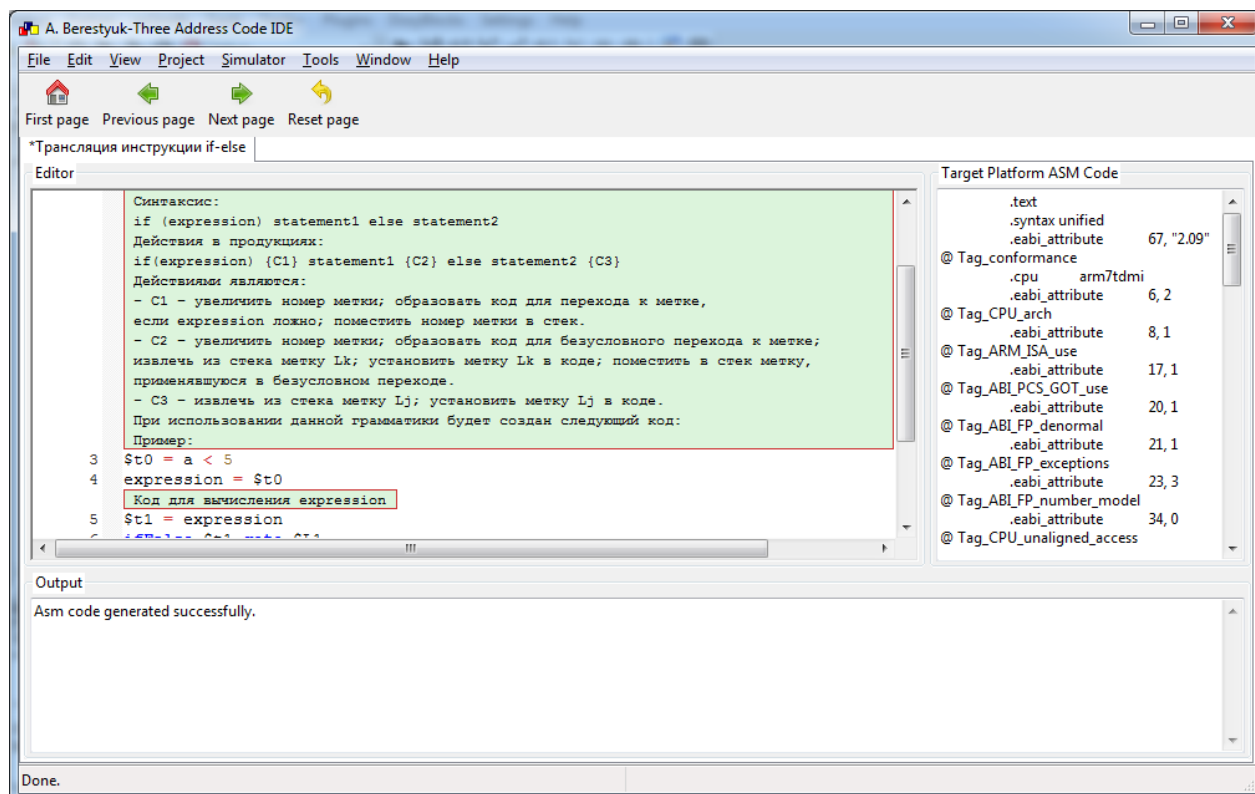


Рисунок 21 – Главное окно программы в режиме обучения

Пользователю предоставляется возможность внести изменения в пример и изучить влияние данных изменений на выполнение программы. Навигация по примерам осуществляется с помощью кнопок дополнительной панели.

3.7 Комплект поставки программной системы

Программная система поставляется в открытых исходных кодах. Основные элементы файловой структуры комплекта поставки представлены

ниже, пути к файлам указаны относительно корневого каталога программной системы в алфавитном порядке:

`/include` – каталог, содержит файлы заголовков модулей программной системы;

`/src` – каталог, содержит файлы исходных кодов модулей программной системы;

`LICENCE.txt` – файл, содержит текст лицензии на программную систему;

`makefile` – основной файл сборки программной системы, содержит скрипт сборки для утилиты сборки `make` [9];

`makefile.linux` – файл сборки программной системы, содержит скрипт сборки для семейства операционных систем GNU Linux;

`makefile.mac` – файл сборки программной системы, содержит скрипт сборки для операционной системы MacOSX;

`makefile.win` – файл сборки программной системы, содержит скрипт сборки для операционной системы Windows;

`README.md` – файл, содержит общее описание и инструкцию по сборке программной системы.

Исходные коды программной системы размещены в общественном репозитории Github, режим доступа: <https://github.com/elsonriente/ide3ac>.

3.8 Лицензирование программной системы

Программная система распространяется в открытых исходных кодах под лицензией «zlib»[17]. В связи с объемом исходных кодов библиотеки графического интерфейса `wxWidgets`, а также библиотек инфраструктуры компиляторов LLVM, было принято решение не включать данные исходные файлы в комплект поставки, таким образом, дополнительное лицензирование не требуется.

4 Руководство пользователя

4.1 Общие сведения о программном продукте

Название программного продукта «ide3ac». Исходные файлы на языке программирования C++ для ОС Windows, GNU Linux, MacOSX.

Применяется для разработки программ в трехадресном коде при изучении компиляторных курсов.

Программная система обеспечивает редактирование файлов программ с подсветкой синтаксиса, выделением базовых блоков, генерацию соответствующего программе ассемблерного кода целевой вычислительной платформы. Поддерживает запуск редактируемой программы в симуляторе.

4.2 Описание установки

Программный продукт поставляется в архиве в открытых исходных текстах. Сначала архив надо распаковать в отдельный каталог.

Программный продукт является исходными файлами. Поэтому перед запуском программы необходимо провести процедуру компиляции. Для этого на компьютере должна быть установлена библиотека графического интерфейса wxWidgets 3.0.2, программное обеспечение LLVM 3.8.0. Для компиляции в ОС Windows, должно быть установлено программное обеспечение MinGW. Сборка выполняется с помощью make.

4.3 Описание запуска

Вызов программы производится:

- ОС Windows: путем запуска запускаемого файла ide3ac.exe.
- ОС GNU Linux i686: путем запуска запускаемого файла ide3ac.

- ОС OS Mac OS X: путем запуска запускаемого файла ide3ac.

4.4 Инструкция по работе

Общее описание графического интерфейса программы изложено в разделе 3.6 «Интерфейс пользователя».

После запуска программы отображается главное окно. Функции:

а) управление файлами – создание нового, открытие, закрытие, просмотр свойств файла;

б) редактирование - отмена последнего совершенного действия, повтор последнего отмененного действия, перемещение выделенного диапазона текста в буфер обмена, копирование выделенного диапазона текста в буфер обмена, вставка содержимого буфера обмена в текст, удаление выделенного диапазона текста, поиск необходимого фрагмента текста в файле, поиск и замена необходимого фрагмента текста в файле, переход на необходимую позицию в тексте файла, выделение всего содержимого файла, выделение строки текста на позиции курсора;

в) управление отображением – сверка/развертка фрагмента текста, соответствующего базовому блоку, включение/выключение отображения панели быстрого доступа, режима редактирования текста «перезаписью», переноса слов, отображение символов конца строки, отображение номеров строк, отображение маркера границы страницы, отображение пробельных символов;

г) редактирование таблицы символов – для выполнения данного действия, необходимо выбрать пункт меню «Project/Edit symbol table» нажать кнопку «Symbol Table» на панели быстрого доступа, откроется диалоговое окно редактирования таблицы символов, далее следует выбрать тип, размер переменных в таблице и нажать кнопку «Ok» для применения изменений;

д) выбор необходимой целевой вычислительной платформы –

осуществляется выбором соответствующего пункта подраздела меню «Target platform», либо выбором пункта в выпадающем списке панели быстрого доступа;

е) запуск симулятора – осуществляется выбором пункта меню «Simulator/Run in simulator», либо нажатием кнопки «Run» в панели быстрого доступа, откроется диалоговое окно симулятора, отображающее список символов, их адреса и значения на каждом шаге выполнения программы, поле вывода сообщений, элементы управления, далее нажать кнопку «Run» для выполнения программы, либо кнопку «Step» для выполнения текущей инструкции и отображение следующей, либо кнопку «Reset» для сброса состояния симулятора, атрибутов символов к начальным значениям, кнопку «Exit» для закрытия окна симулятора;

ж) Доступ к режиму обучения – осуществляется выбором пункта меню «Tools/Toggle educational mode», в области редактора отобразится пример с пояснениями по тематике генерации промежуточного кода, навигация по примерам производится с помощью кнопок панели навигации;

з) Доступ к настройкам программы – осуществляется выбором пункта меню «Tools/Options», либо нажатием кнопки «Options» в панели быстрого доступа;

и) доступ к справочной системе программы – осуществляется выбором пункта меню «Help/Help Contents», либо нажатием кнопки «Help» в панели быстрого доступа;

к) Доступ к справочной системе программы – осуществляется выбором пункта меню «Help/About..».

5 Руководство системного программиста

5.1 Общие сведения о программном продукте

5.1.1 Назначение и функции программы

Название программного продукта «ide3ac». Исходные файлы на языке программирования C++ для ОС Windows, GNU Linux, MacOSX. Применяется для разработки программ в трехадресном коде при изучении компиляторных курсов. Программная система обеспечивает редактирование файлов программ с подсветкой синтаксиса, выделением базовых блоков; генерацию соответствующего программе ассемблерного кода целевой вычислительной платформы; поддерживает запуск редактируемой программы в симуляторе.

5.1.2 Необходимы ресурсы

Для функционирования программы на персональном компьютере должны быть установлены ОС Windows 7 или выше, либо ОС GNU Linux i686, либо ОС Mac OS X 10.5.0 или выше.

5.2 Комплект поставки и установки

Программный продукт поставляется в архиве в открытых исходных текстах. Сначала архив надо распаковать в отдельный каталог.

Программный продукт является исходными файлами. Поэтому перед запуском программы необходимо провести процедуру компиляции. Для этого на компьютере должна быть установлена библиотека графического интерфейса wxWidgets 3.0.2, программное обеспечение LLVM 3.8.0. Данные инструменты должны быть собраны в виде статических библиотек с поддержкой кодировки

юникод. Для компиляции в ОС Windows, должно быть установлено программное обеспечение MinGW. Сборка выполняется с помощью make.

5.3 Описание запуска.

Вызов программы производится:

- ОС Windows: путем запуска запускаемого файла ide3ac.exe.
- ОС GNU Linux i686: путем запуска запускаемого файла ide3ac.
- ОС OS Mac OS X: путем запуска запускаемого файла ide3ac.

5.4 Инструкция по работе.

Инструкция по работе описана в разделе 4.4.

ЗАКЛЮЧЕНИЕ

При изучении компиляторных курсов, и, в частности, темы промежуточного представления исходных текстов, также рассматривается генерация трёхадресного кода, однако его корректность декларируется «как есть». Это обусловлено тем, что некоторые его конструкции недостаточно формализованы. С целью устранения данного недостатка, а также, для обеспечения наглядного представления программы и возможности «удобного» редактирования кода было проведено исследование существующих сред разработки на предмет целесообразности добавления необходимого функционала. В ходе анализа было принято решение о разработке программной системы «с нуля» существующими программно-аппаратными средствами.

В рамках данной работы была разработана программная система, реализующая функции редактора кода с подсветкой синтаксиса, генерации кода ассемблера для целевых вычислительных платформ, симулятора выполнения программ в трёхадресном коде, демонстрации примеров кода. Система предназначена для использования в курсах разработки компиляторов при изучении промежуточного представления программ. Также написано методическое пособие по генерации кода для использования совместно с программой, фрагмент пособия представлен в приложении А. В целом, программная система отвечает поставленным требованиям, также обладает потенциалом для дальнейшего развития. Возможно расширение обучающего режима, добавление упражнений, добавление функций оптимизации кода на уровне базовых блоков и процедур, расширение функций редактора, отображение графа потока управления.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ахо А. В. Компиляторы: принципы, технологии и инструментарий /А. В. Ахо, М. С. Лам, Р. Сети, Д. Д. Ульман. - Москва : Изд. “Вильямс”, 2008.
2. CMake [Электронный ресурс]. - Режим доступа: <https://cmake.org> (дата обращения 10.03.2016).
3. Code::Blocks [Электронный ресурс]. - Режим доступа: <http://www.codeblocks.org> (дата обращения 17.02.2016).
4. Doxygen: Main Page [Электронный ресурс]. - Режим доступа: <http://www.stack.nl/~dimitri/doxygen/index.html> (дата обращения 29.04.2016).
5. Eclipse – The Eclipse Foundation open source community website. [Электронный ресурс]. - Режим доступа: <https://eclipse.org/> (дата обращения 17.02.2016).
6. GCC, the GNU Compiler Collection [Электронный ресурс]. - Режим доступа: <https://gcc.gnu.org> (дата обращения 17.02.2016).
7. Git [Электронный ресурс]. - Режим доступа: <https://git-scm.com> (дата обращения 02.05.2016).
8. LLVM Language Reference Manual [Электронный ресурс]. - Режим доступа: <http://llvm.org/docs/LangRef.html> (дата обращения 18.02.2016).
9. GNU Make [Электронный ресурс]. - Режим доступа: <https://www.gnu.org/software/make/> (дата обращения 15.05.2016).
10. MinGW. Minimalist GNU for Windows [Электронный ресурс]. - Режим доступа: <http://www.mingw.org> (дата обращения 14.03.2016).
11. Parrot, speaks your language [Электронный ресурс]. - Режим доступа: <https://parrot.org> (дата обращения 17.02.2016).
12. Scintilla and SciTE [Электронный ресурс]. - Режим доступа: <http://www.scintilla.org/> (дата обращения 15.05.2016).
13. Sparx Systems Product Family [Электронный ресурс]. - Режим доступа:

- <http://www.sparxsystems.com.au/products/index.html#EA> (дата обращения 12.04.2016).
14. The LLVM Compiler Infrastructure [Электронный ресурс]. - Режим доступа: <https://llvm.org> (дата обращения 17.02.2016).
15. wxWidgets. Cross-Platform GUI Library [Электронный ресурс]. - Режим доступа: <http://www.wxwidgets.org> (дата обращения 17.02.2016).
16. Xcode [Электронный ресурс]. - Режим доступа: <https://developer.apple.com/xcode/> (дата обращения 12.05.2016).
17. Zlib Licence [Электронный ресурс]. - Режим доступа: http://www.gzip.org/zlib/zlib_license.html (дата обращения 14.06.2016).

ПРИЛОЖЕНИЕ А

Методическое пособие. Фрагмент (раздел 2.2 Базовые блоки)

Мы введем представление промежуточного кода на основе графа, полезное при рассмотрении кодогенерации. При этом не столь важно, строит ли алгоритм кодогенерации граф явным образом или нет. Кодогенерация получает большие преимущества, имея сведения о контексте.

Представление на основе графа строится следующим образом.

а) Промежуточный код разделяется на **базовые блоки** (*basic blocks*, далее – ББ), представляющие собой максимальные последовательности следующих друг за другом трехадресных команд. Эти последовательности обладают следующими свойствами:

- 1) поток управления может входить в ББ только через первую инструкцию блока, т.е. переходы в середину блока отсутствуют;
- 2) управление покидает ББ без останова или ветвления. Исключением может стать только последняя инструкция в блоке.

б) ББ становятся узлами **графа потока управления** (*control flow graph*, далее – CFG), чьи дуги показывают порядок следования блоков.

При рассмотрении вопросов оптимизации кода мы будем изучать трансформации CFG, которые преобразуют исходный промежуточный код в «оптимизированный» промежуточный код, позволяющий генерировать более качественный целевой код. «Оптимизированный» промежуточный код трансформируется в целевой код с помощью рассматриваемых нами методов генерации кода.

Итак, в первую очередь мы должны обеспечить разбиение последовательности трехадресных инструкций на ББ. Каждый новый ББ начинается с первой инструкции, а остальные инструкции добавляются до тех пор, пока не встретится инструкция перехода или метка следующей инструкции. При отсутствии переходов и меток управление последовательно

проходит от одной инструкции к другой. Эту идею мы развиваем в следующем алгоритме.

Алгоритм разбиения трехадресных инструкций на ББ

ВХОД: последовательность трехадресных инструкций.

ВЫХОД: список ББ для данной последовательности, в которой каждая инструкция принадлежит ровно одному ББ.

МЕТОДИКА: сначала определяются инструкции промежуточного кода, являющиеся **лидерами** (*leader*), т.е. первыми элементами в некоторых ББ (лидер в ББ всегда только один). Инструкция сразу после окончания промежуточной программы в лидеры не включается. Для поиска лидеров используются следующие правила:

а) Первая трехадресная инструкция промежуточного кода является лидером.

б) Любая инструкция, являющаяся целевой для условного или безусловного перехода, является лидером.

в) Любая инструкция, следующая непосредственно за условным или безусловным переходом, является лидером.

После этого делают еще один проход по промежуточной программе, и каждый ББ содержит самого лидера и все инструкции до следующего лидера или до конца промежуточной программы.

Промежуточный код, показанный ниже, превращает матрицу 10 x 10 в единичную. При генерации промежуточного кода предполагается, что элементы массива являются числами с плавающей точкой и имеют размер 8 байт.

- 1) $i = 0$
- 2) $\$L0: j = 0$
- 3) $\$L1: \$t1 = 10 * i$
- 4) $\$t2 = \$t1 + j$
- 5) $\$t3 = 8 * \$t2$
- 6) $\$t4 = \$t3 - 88$

- 7) `a[$t4] = 0.0`
- 8) `j = j + 1`
- 9) `$t5 = j < 10`
- 10) `ifTrue $t5 goto $L1`
- 11) `i = i + 1`
- 12) `$t6 = i < 10`
- 13) `ifTrue $t6 goto $L0`
- 14) `i = 1`
- 15) `$L2: $t7 = i - 1`
- 16) `$t8 = 88 * $t7`
- 17) `a[$t8] = 1.0`
- 18) `i = i + 1`
- 19) `$t9 = i < 10`
- 20) `ifTrue $t9 goto $L2`

Хотя происхождение этого кода не важно, он может быть результатом трансляции С-кода:

```
for (i=0; i < 10; ++i)
    for (j=0; j < 10; ++j)
        a[i][j] = 0.0;
for (i=0; i < 10; ++i)
    a[i][i] = 1.0;
```

Инструкция 1 является лидером в соответствии с первым правилом нашего алгоритма. Для поиска других лидеров, мы сначала находим инструкции переходов. В данном примере имеются три такие инструкции – 10, 13 и 20. Согласно второму правилу целевые инструкции этих переходов являются лидерами; это инструкции 3, 2 и 15 соответственно. По третьему правилу каждая инструкция, следующая за переходом, является лидером, у нас это 11 и 14. В коде нет инструкций, следующих за 20 инструкцией, но если бы такая инструкция была, то инструкция 21 также стала бы лидером.

В результате мы приходим к выводу, что лидерами являются инструкции

1, 2, 3, 11, 14 и 15. ББ каждого лидера содержат все инструкции от него самого до инструкции перед следующим лидером включительно. Таким образом, ББ лидера 1 состоит только из одной инструкции 1, блок лидера 2 – из одной инструкции 2. Блок лидера 3 состоит из инструкций с 3 по 10 включительно. Элементами блока лидера 11 являются инструкции 11, 12 и 13. Блок лидера 14 состоит из единственной инструкции 14, а в блок лидера 15 входят команды 15-20.