**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Образовательная программа «Программная инженерия»

|  |  |  |  |
| --- | --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель, профессор департамента «Программной инженерии», доктор технических наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Легалов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н. А. Павлочев  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **КРОСС-КОМПИЛЯТОРС ЯЗЫКА ПРОГРАММИРОВАНИЯ SMALL C**  **В АССЕМБЛЕР ПРОЦЕССОРА RISC-V, ДЛЯ ЭМУЛЯТОРА RARS**  **Руководство оператора**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01 34 01-1-ЛУ** | | | | | |
|  | | Исполнитель  студент группы БПИ229  \_\_\_\_\_\_\_\_\_ / Р. С. Громов/  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | | |

**Москва 2024**

УТВЕРЖДЕН

RU.17701729.10.03-01 34 01-1-ЛУ

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **КРОСС-КОМПИЛЯТОРС ЯЗЫКА ПРОГРАММИРОВАНИЯ SMALL C**  **В АССЕМБЛЕР ПРОЦЕССОРА RISC-V, ДЛЯ ЭМУЛЯТОРА RARS**  **Руководство оператора**  **RU.17701729.04.01-01 34 01-1**  **Листов 11** | |
|  |  |
|  |
|  | |

**Москва 2024**

**СОДЕРЖАНИЕ**

[**1. НАЗНАЧЕНИЕ РАЗРАБОТКИ** 3](#_Toc162889026)

[**1.1 Функциональное назначение** 3](#_Toc162889027)

[**1.2 Эксплуатационное назначение** 3](#_Toc162889028)

[**2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ** 4](#_Toc162889029)

[**2.1. Минимальный состав аппаратурных средств** 4](#_Toc162889030)

[**2.2. Минимальный состав программных средств** 4](#_Toc162889031)

[**2.3. Требования к пользователю** 4](#_Toc162889032)

[**3. ВЫПОЛНЕНИЕ ПРОГРАММЫ** 5](#_Toc162889033)

[**3.1 Установка программы** 5](#_Toc162889034)

[**3.2 Описание интерфейса** 5](#_Toc162889035)

[**3.2.1 Ввод** 5](#_Toc162889036)

[**3.2.2 Вывод** 5](#_Toc162889037)

[**4. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 8](#_Toc162889038)

[**ПРИЛОЖЕНИЕ** 9](#_Toc162889039)

[**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ** 11](#_Toc162889040)

1. НАЗНАЧЕНИЕ РАЗРАБОТКИ
   1. Функциональное назначение

Функциональное назначение «RISC-V Cross-compiler» представимо в виде списка основных функциональных групп приложения:

* Выбор исходных файлов программы, написанной на языке Small C, а также их валидация для её дальнейшей компиляции.
* Препроцессинг: получение отформатированной копии программы путём удаления лишних символов и комментариев и передача её лексическому анализатору.
* Компиляция: получение ассемблерного листинга исходной программы в ходе выполнения лексического, синтаксического, семантического анализа и генерации промежуточного кода.
* Получение информации о синтаксических и компиляционных ошибках и предупреждений
* Генерация промежуточного кода в ассемблерные инструкции архитектуры RISC-V.
* Запуск ассемблерного кода на эмуляторе RARS.
  1. Эксплуатационное назначение

Приложение «RISC-V Cross-compiler» представляет из себя полноценный кроссплатформенный компилятор для трансляции высокоуровнего кода в ассемблерные инструкции.

1. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ
   1. Минимальный состав аппаратурных средств

Испытания должны проводиться на персональном компьютере со следующими характеристиками:

1. Процессор 2-х или более ядерный с тактовой частотой 2 ГГц или выше;
2. Более 2 Гб оперативной памяти;
3. Более 1 Гб свободного дискового пространства;
4. Клавиатура;
5. Монитор.
   1. Минимальный состав программных средств

Для работы программы необходим следующий состав программных средств:

1. Операционная система Windows 7 (или более поздняя версия) или один из дистрибутивов Linux;
2. Компилятор GCC версии 6.0 или выше;
3. CMake версии 3.0 или выше.
   1. Требования к пользователю

Пользователь, должен понимать синтаксис языка Small-C, иметь базовые знания командной строки Linux или Windows (в зависимости от используемой ОС), а также знать, как установить RISC\_V Cross compiler себе на персональный компьютер и запустить его. Знакомство с CMake будет полезно в случае возникновения ошибок, но в общем случае не является обязательным.

1. ВЫПОЛНЕНИЕ ПРОГРАММЫ
   1. Установка программы

Программа хранится в публичном репозитории, доступ к которому можно получить по следующей ссылке: *https://github.com/mrForza/RISC-V-Cross-compiler*

Чтобы установить программу себе на локальное устройство, нужно:

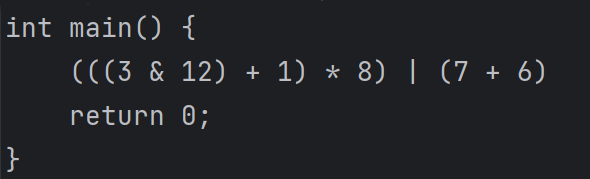
1. Склонировать репозиторий с помощью утилиты *gitbash* с помощью команды *git clone https://github.com/mrForza/RISC-V-Cross-compiler.git;*
2. Перейти в директорию, в которую был склонирован репозиторий;
3. С помощью компилятора gcc скомпилировать файл main.c в исполняемый файл;
4. Запустить исполняемый файл.
   1. Описание интерфейса
      1. Ввод

Для того, чтобы взаимодействовать с компилятором, нужно запустить исполняемый файл main.exeна Windows или main на Linux. В консольном окне пользователю будет предложен ввод пути к исходной программе, написанной на языке Small-C. Важно вводить путь относительно той директории, в которой лежит исполняемый файл программы. После ввода пути пользователь либо получит сообщение об ошибке чтения файла, либо сообщение об успешной компиляции программы. Также во время компиляции пользователь может получить синтаксическую или семантическую ошибку.

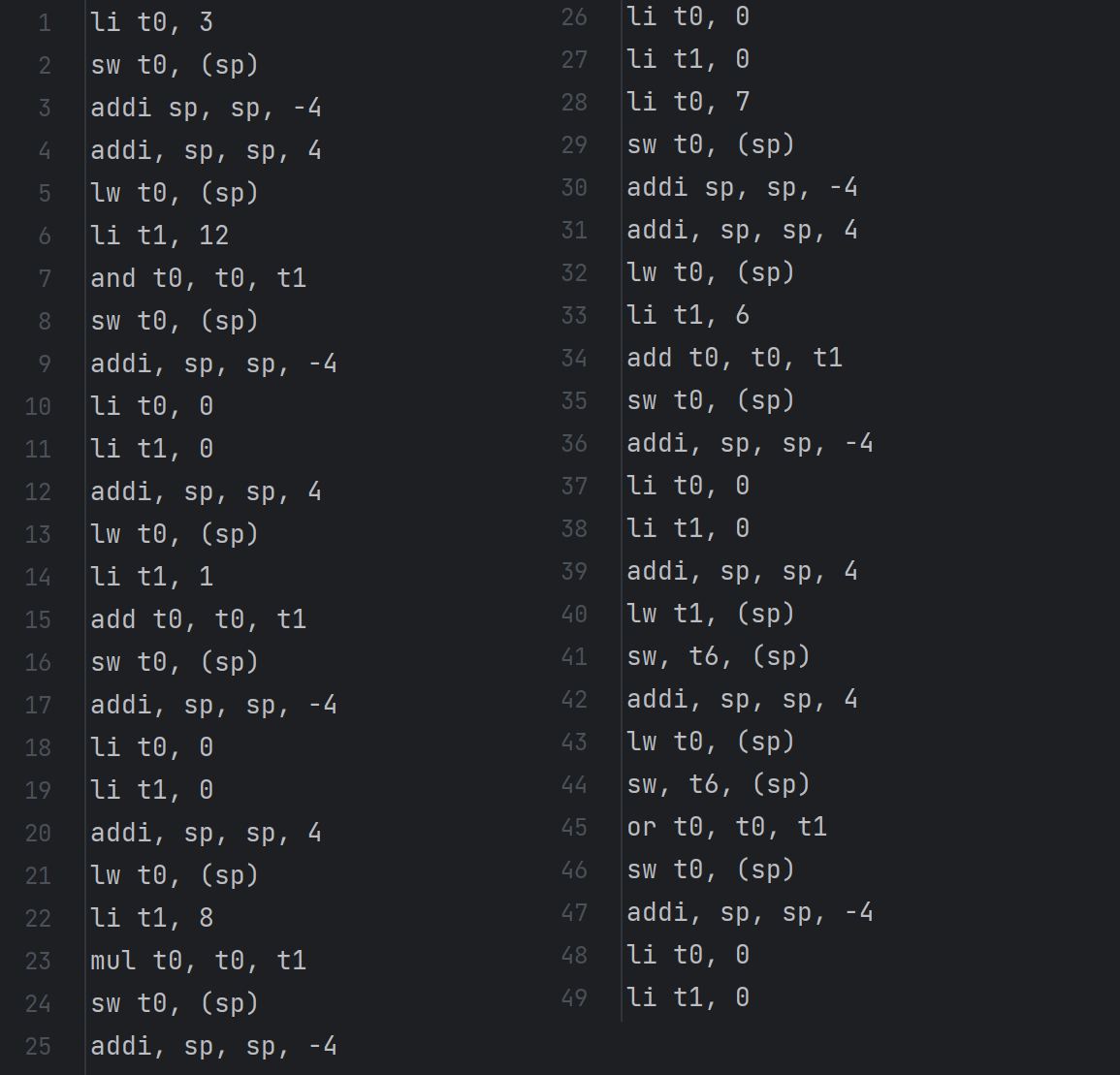
* + 1. Вывод

При чтении исходного файла, компилятор выводит ассемблерный листинг в консоль и в файл, имеющий название: assembly.asm.

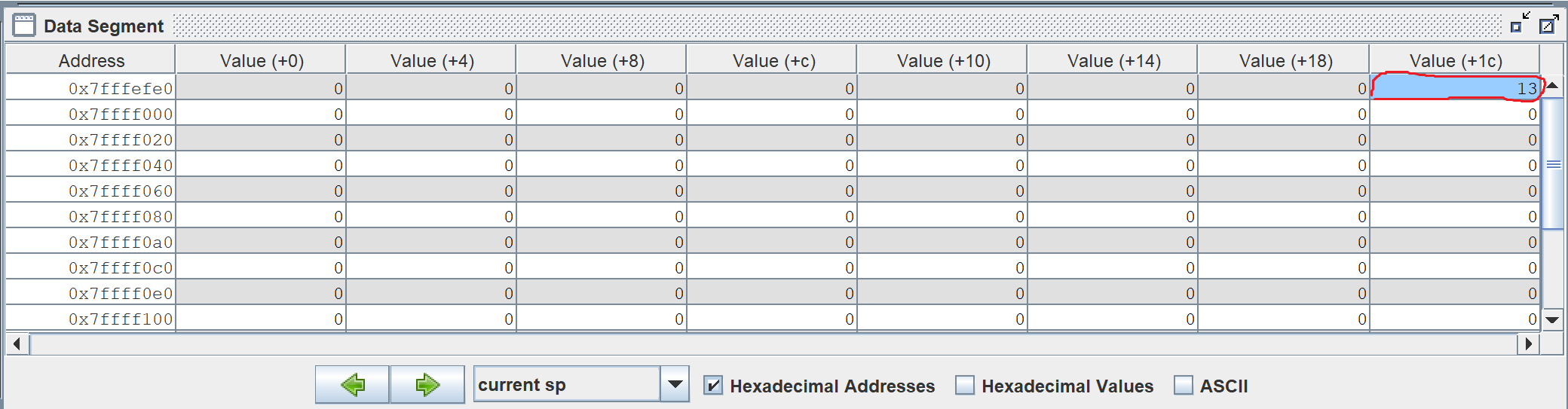
*Рисунок 1. Исходный текст программы на языке Small-C. Арифметическое выражение*



*Рисунок 2. Вывод ассемблерного листинга*

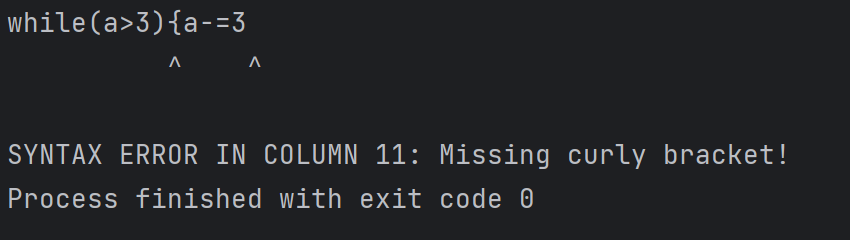


*Рисунок 3. Запуск ассемблера в эмуляторе RARS*

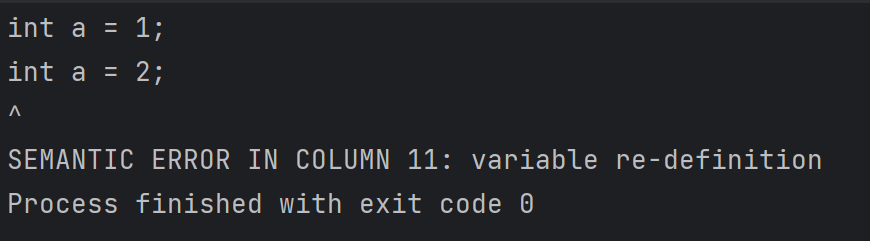


Вывод как на *рисунке 2* появляется, когда программа не содержит ошибок. В противном случае будут выведены сообщения об ошибке. Пример сообщения о синтаксической ошибке представлен на *рисунке 4*. В данном случае была пропущена фигурная скобка в теле цикла while. Сообщение содержит тип ошибки, её описание, а также номер колонки в строке, в которой была допущена ошибка. Пример сообщения о семантической ошибке представлен на *рисунке 5*. В конце генерации низкоуровнего кода компилятора, продемонстрирована работа запуска этого кода на эмуляторе RARS. При генерации арифметических выражений ассемблер создаётся таким образом, чтобы результат этого арифметического выражения лежал на вершине стка. Как видно из *рисунка 3*, на вершине стэка лежит число 13, что как раз является истинным результатом данного выражения.

*Рисунок 4. Синтаксическая ошибка.*



*Рисунок 5. Семантическая ошибка.*



1. **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**
2. Small-C Wikipedia.
3. J.G. Harstone. // C compilers for the BBC Computer.
4. Д. Хендрикс. // Компилятор языка Си для микроЭВМ.
5. Альфред В. Ахо, Моника С. Лам, Рави Сети, Джеффри Д. Ульман. // Компиляторы: принципы, технологии и инструментарий.
6. Свердлов С. // Конструирование компиляторов, учебное пособие.

# **ПРИЛОЖЕНИЕ**

**ТЕРМИНОЛОГИЯ**

**Компилятор** — программа, переводящая написанный на языке программирования текст в набор машинных кодов.

**Лексический анализ** (токенизация) – это процесс аналитического разбора входной последовательности символов на распознанные группы — лексемы — с целью получения на выходе идентифицированных последовательностей, называемых «токенами».

**Синтаксический анализ** (парсинг) в лингвистике и информатике - это процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора (синтаксическое дерево). Обычно применяется совместно с лексическим анализом.

**Синтаксис** языка программирования — набор правил, описывающий комбинации символов алфавита, считающиеся правильно структурированной программой (документом) или её фрагментом.

**Препроцессор** — компьютерная программа, принимающая данные на входе и выдающая данные, предназначенные для входа другой программы.

**Метаязык** — язык, предназначенный для описания другого языка, называемого объектным языком.

**Терминальные** **символы** – это символы, которые могут появляться в выходных данных производственных правил формальной грамматики и которые не могут быть изменены с помощью правил грамматики.

**Нетерминал** (нетерминальный символ) — объект, обозначающий какую-либо сущность языка (например: формула, арифметическое выражение, команда) и не имеющий конкретного символьного значения.

**Формальная грамматика** или просто грамматика в теории формальных языков — способ описания формального языка, то есть выделения некоторого подмножества из множества всех слов некоторого конечного алфавита.

**Регулярные выражения** – это формальный язык, используемый в компьютерных программах, работающих с текстом, для поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.

**Лексема** – это последовательность символов исходной программы, которая соответствует шаблону токена и идентифицируется лексическим анализатором как экземпляр токена.

**Шаблон (pattern)** — это описание вида, который может принимать лексема токена. В случае ключевого слова шаблон представляет собой просто последовательность символов, образующих это ключевое слово.

**Токен** – это пара, состоящая из имени токена и необязательного атрибута.

# **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |