Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

разработка компилятора подмножества

процедурно-ориентированного языка

Пояснительная записка

RU. 643.02068048.0001-01 81 01

На 11 листах

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н. доцент кафедры ИЗИ Ю.М. Монахов |
|  |
| Исполнитель |  | студент гр. ИСБ-117 И.Д. Койков |

**Владимир 2020**

# АННОТАЦИЯ

В данном программном документе приведён текст компилятора подмножества процедурно-ориентированного языка. Компилятор реализован на языке Java с использованием библиотеки Antlr4.8.

Разработка компилятора подмножества процедурного языка в ассемблер состоит из следующих стадий:

1) построение лексического анализатора;

2) построение синтаксического анализатора;

3) построение генератора объектного кода;

Оглавление

[Министерство науки и высшего образования Российской Федерации 1](#_Toc42068853)

[разработка компилятора подмножества 1](#_Toc42068854)

[процедурно-ориентированного языка 1](#_Toc42068855)

[АННОТАЦИЯ 2](#_Toc42068856)

[1 ПРОЕКТИРОВАНИЕ КОМПИЛЯТОРА 4](#_Toc42068857)

[1.1 Основные требования 4](#_Toc42068858)

[1.2. Создание лексического анализатора 5](#_Toc42068859)

[1.3. Разработка синтаксического анализатора 6](#_Toc42068860)

[1.4. Построение генератора объектного кода 7](#_Toc42068861)

# 1 ПРОЕКТИРОВАНИЕ КОМПИЛЯТОРА

## Основные требования

Разработка будет производиться в соответствии со следующими требованиями:

* Требования к входному языку:

1. Должны присутствовать операторные скобки;
2. Должна игнорироваться индентация программы;
3. Должны поддерживаться комментарии любой длины;

* Требования к операторам:

1. Оператор присваивания;
2. Арифметические операторы;
3. Логические операторы (И, ИЛИ, НЕ);
4. Условный оператор (ЕСЛИ);
5. Оператор цикла (while, break, continue);
6. Базовый вывод (строковой литерал, переменная);
7. Типы (целочисленный, вещественный).

* Требования к выходному языку:

1. Ассемблер.

## 1.2. Создание лексического анализатора

Лексический анализатор является первой фазой работы компилятора. Работа лексического анализатора основана на регулярных выражениях. Лексический анализатор выполняет функцию разбора входной последовательности символов на распознанные группы(лексемы) с целью получения на выходе токенами. Лексический анализатор разработан с помощью Antlr 4 под Java.

На рисунке 1 приведен список регулярных выражения для описания лексем языка.

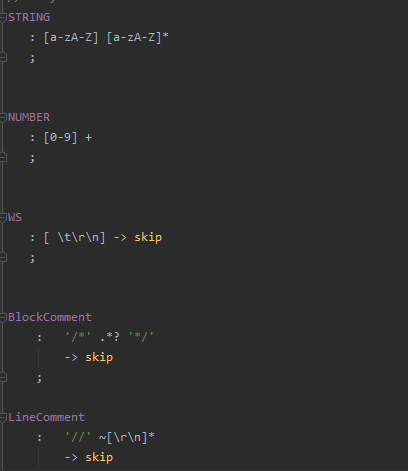


Рисунок 1 Регулярные выражения для описания лексем

## 1.3. Разработка синтаксического анализатора

Второй стадией компилятора является синтаксический анализ. На Синтаксический анализатор -это часть программы, преобразующей входные данные в структурированный формат. Все взаимодействия во время построения дерева разбора обрабатываются через класс MyVisitor. Синтаксический анализатор разработан с помощью Antlr 4 под Java.



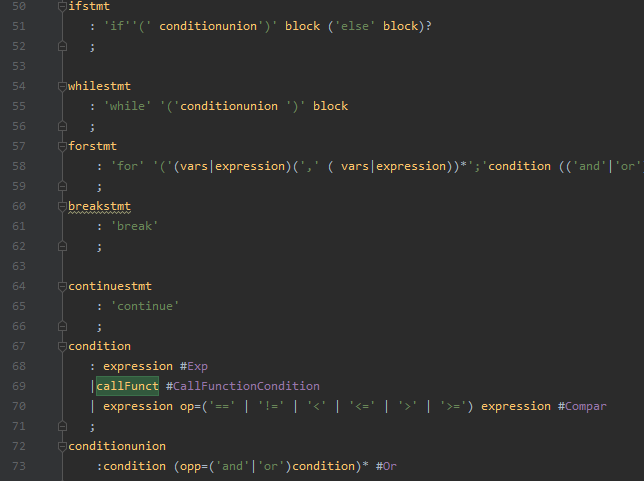
Рисунок 2 Грамматика часть 1

Рисунок 3 Грамматика часть 2

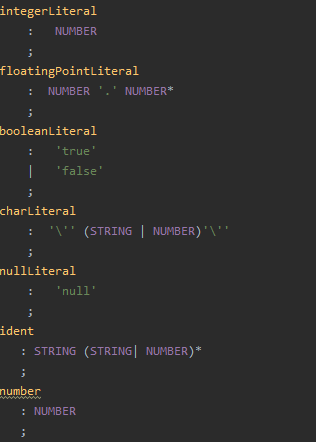


Рисунок 4 Грамматика часть 3

## 1.4. Построение генератора объектного кода

Генерация объектного кода выполняется во время обхода дерева в классе MyVisitor. Генерация осуществляется путем добавления в общий буфер объектного кода строк в объектном коде, эквивалентных исходному языку. Буфер записывается в файл с расширением ll. Все функции генерации объектного кода представлены в классе GeneratorLLVM, где реализованы следующие функции:

* Generate – вывод буфера в файл ll
* Function\_start
* Function\_end
* Call
* Функции объявления переменных (глобально и локально)
* Функции присвоения значения переменным
* Функции вывода на экран
* Функции арифметических действий
* Функции логических действий
* Функции загрузки переменных из памяти
* If\_start
* If\_end
* Функции для цикла while
* Continue
* Break

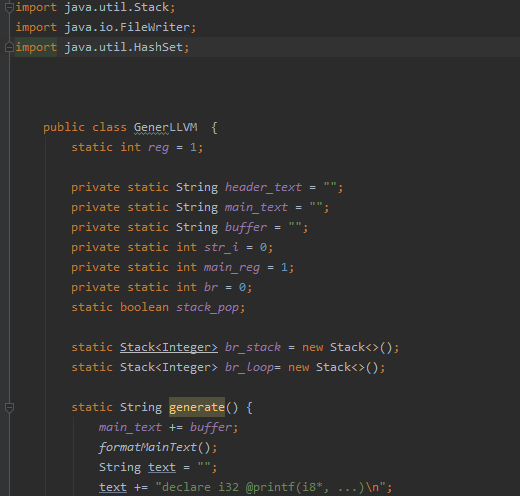


Рисунок 5 Функция generate

ПРОВЕРКА НА СООТВЕТСТВИЕ ОСНОВНЫМ ТРЕБОВАНИЯМ

При проектировании компилятора к основному языку были установлены следующие минимальные требования: наличие операторных скобок, игнорирование пробелов и идентации программы, поддержка многострочных комментариев. Наличие операторов присваивания, условных, цикла, break-continue, арифметических, логических. Должны присутствовать два типа данных. И выходная программа должна быть на ассемблере.

Далее приведено тестирование компилятора.

Рисунки 6-8 – проверка тестирование If

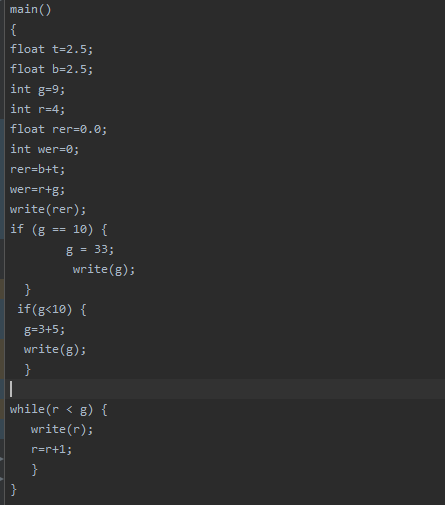


Рисунок 6 Код на исходном языке

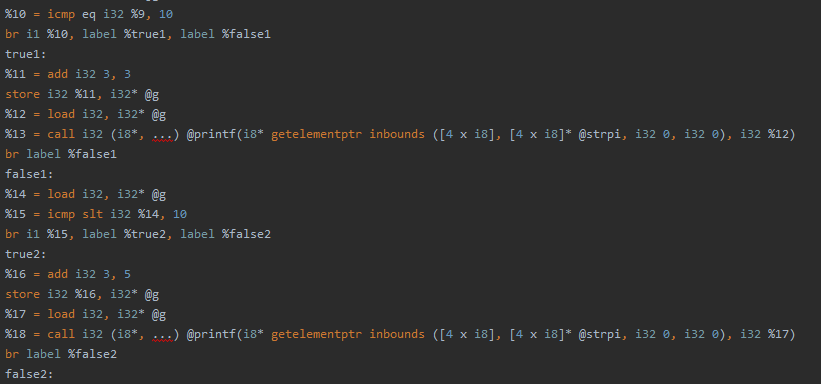


Рисунок 7 Код на языке llvm

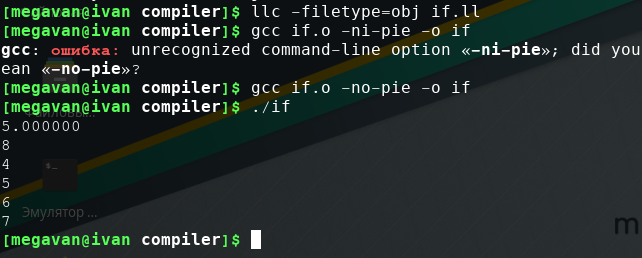


Рисунок 8 результат программы.

Рисунки 9-11 – проверка цикла while

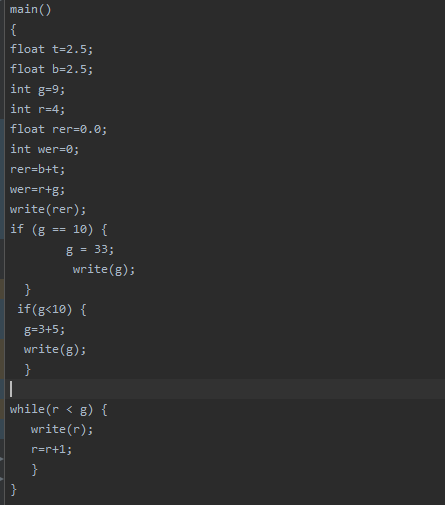


Рисунок 9 Код на исходном языке

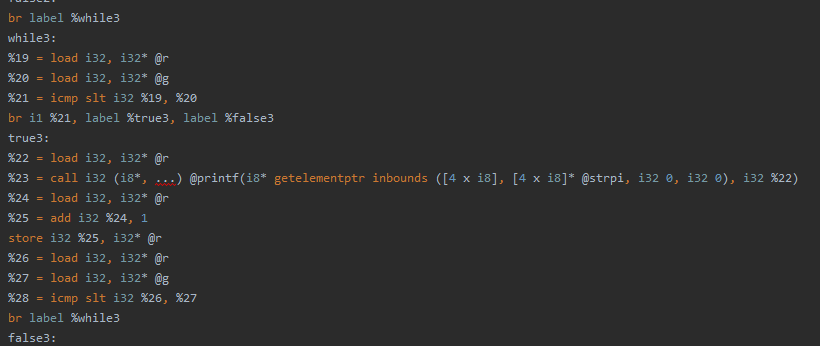


Рисунок 10 Код на языке llvm

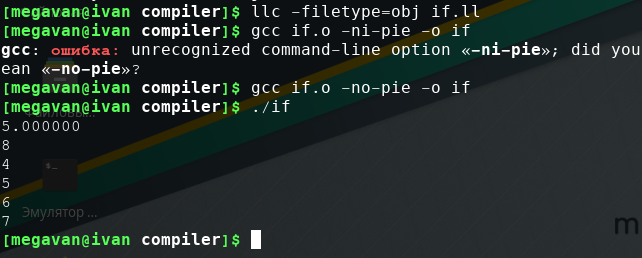


Рисунок 11 результат программы

Реквизиты к курсовой работе:

<https://github.com/IvanKoykov/compyle>