**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 ПРОЦЕДУРНОГО ЯЗЫКА**

Пояснительная записка

На 19 листах

|  |  |  |
| --- | --- | --- |
| Руководитель |  | к.т.н. доцент кафедры ИЗИ Ю.М. Монахов |
| Исполнитель |  | студент гр. ИСБ-119 Д. А. Журавлев |
|  |  |  |

**Владимир 2022**

# АННОТАЦИЯ

В данном документе приведено описание компилятора подмножества процедурно-ориентированного языка safron++. Компилятор реализован на языке Java с использованием библиотек Antlr4. Основная функция компилятора – проверка принадлежности исходной цепочки входному языку и генерация выходной цепочки символов виде MIPS-кода.

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

* Реализация лексического анализатора
* Реализация синтаксического анализатора
* Реализация таблицы символов
* Реализация семантический анализатора
* Реализация генератора промежуточного кода
* Трансляция в целевой код
* Оптимизация

Реквизиты к курсовой работе:

Оглавление

[АННОТАЦИЯ 2](#_Toc104057069)

[1 ПРОЕКТИРОВАНИЕ КОМПИЛЯТОРА 4](#_Toc104057070)

[**1.1** **Основные требования** 4](#_Toc104057071)

[**1.2 Лексический анализатор** 5](#_Toc104057072)

[**1.3 Синтаксический анализатор** 7](#_Toc104057073)

[**1.4 Таблица символов** 9](#_Toc104057074)

[**1.5 Семантический анализ** 10](#_Toc104057075)

[**1.6 Построение генератора промежуточного и целевого кода** 12](#_Toc104057076)

[2 ПРОВЕРКА ВЫПОЛНЕНИЯ ТРЕБОВАНИЙ К ЯЗЫКУ 16](#_Toc104057077)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc104057078)

# 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;
4. Входная программа должна представлять собой единый модуль, но поддерживать вызов функций.

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 32 бита, вещественный 32 бита).

### **1.2 Лексический анализатор**

Лексический анализатор был реализован при помощи библиотеки Antlr4. Для его работы необходим набор зарезервированных слов и набора регулярных выражений, при помощи которых лексер будет разбивать исходный код программы на токены. В данной программе переменные описываются строчными буквами, зарезервированные слова прописные. На выходе работы лексического анализатора получаем набор токенов (рисунок 1).

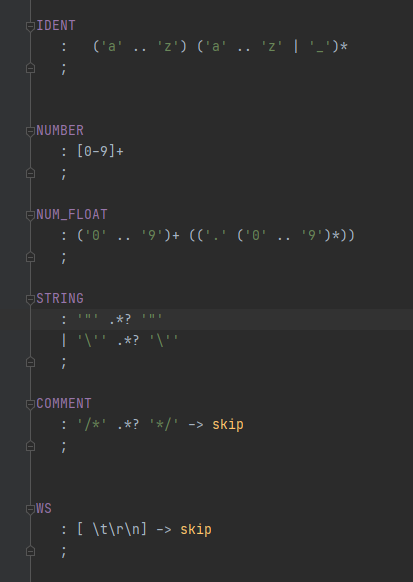
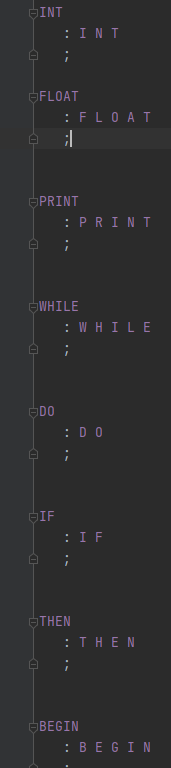


Рисунок 1 – Регулярные выражения и список зарезервированных слов.

Для хранения токенов существует специальный объект Token (номер токена, начало и конец токена в символах, текст токена, номер типа токена, номер строки и символа начала токена) рисунок 2.

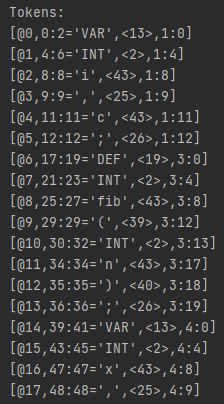


Рисунок 2 – Результат работы лексического анализатора

## **1.3 Синтаксический анализатор**

Следующим этапом курсовой работы является реализация синтаксического анализатора. Для данного этапа работы была использована библиотека Antlr4. Antlr4 генерирует лексер и парсер на основе набора правил для лексера (рисунок 1) и парсера (рисунок 3). Полный листинг КС-грамматики представлен в приложении 1 или в репозитории.

На вход синтаксическому анализатору подается исходный код, поделенный на токены лексером. Результатом работы синтаксического анализатора является синтаксическое дерево разбора. (рисунок 4).

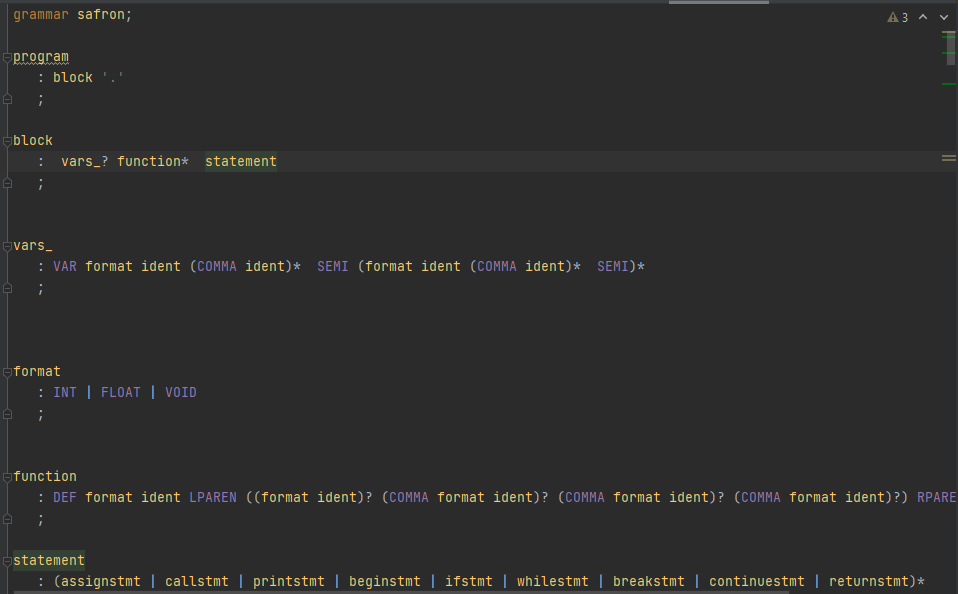


Рисунок 3 – КС-грамматика.

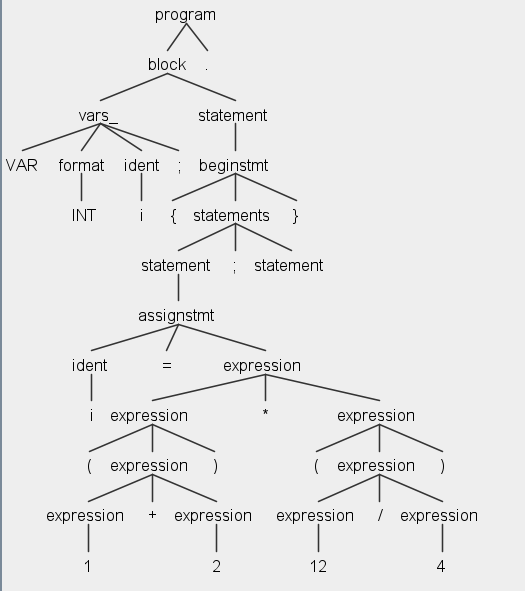


Рисунок 4 – Синтаксическое дерево разбора

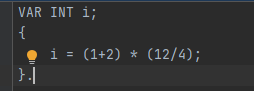


Рисунок 5 – Исходный текст программы

## **1.4 Таблица символов**

Для генерации таблицы символов используется обход синтаксического дерева, при котором тип аргументов и переменных записываются в объект symbolTable. В данной таблице присутствуют ещё 2 поля: assignVarLine и usingVarLine, необходимые для проверки использования переменных и оптимизации кода. Таблица хранится в объекте HashMap, ключом которого является имя переменной, а объект Element хранит остальные столбцы таблицы (рисунок 6). На рисунке представлена таблица символов при создании, а затем после выполнения компиляции.

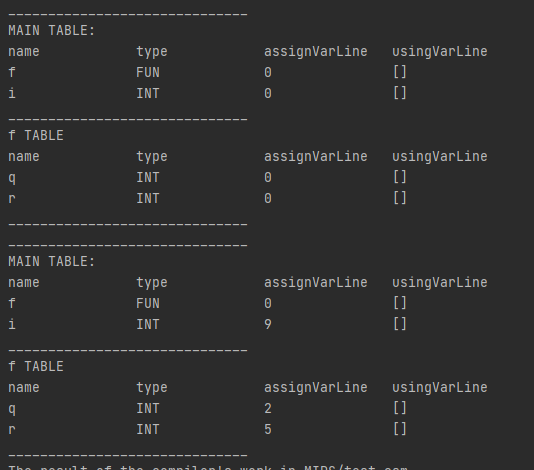


Рисунок 6 – Таблица символов

## **1.5 Семантический анализ**

Семантический анализ – это часть работы компилятора, проверяющая правильность текста исходной программы с точки зрения семантики входного языка.

Данный блок реализован с помощью обхода синтаксического дерева и проверки его с помощью семантических правил.

В их число входит:

* проверка типов переменных
* объявление переменных
* корректное расположение continue, break, return
* количество параметров вызываемой функции
* типы параметров вызываемой функции

Если синтаксический анализ и семантический анализ обнаруживают ошибки компиляция программы прекращается (рисунок 8).

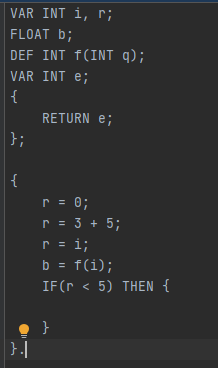


Рисунок 7 – Исходный текст программы

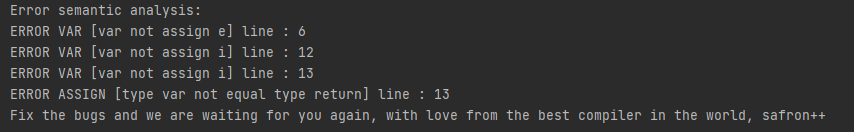


Рисунок 8 – Результат семантического анализа

## **1.6 Построение генератора промежуточного и целевого кода**

Чтобы сгенерировать промежуточный код мною был реализован алгоритм обхода синтаксического дерева, который встречая определенные узлы дерева создавал для них трехадресные инструкции. Первоначально планировалось использования стековой машины jvm, но из-за невозможности генерации java byte code, была выбрана архитектура MIPS.

MIPS - система команд и микропроцессорных архитектур, разработанных компанией MIPS Computer Systems в соответствии с концепцией проектирования процессоров RISC (то есть для процессоров с упрощенным набором команд).

Для хранения инструкций был разработан объект CodeElement (рисунок 9).

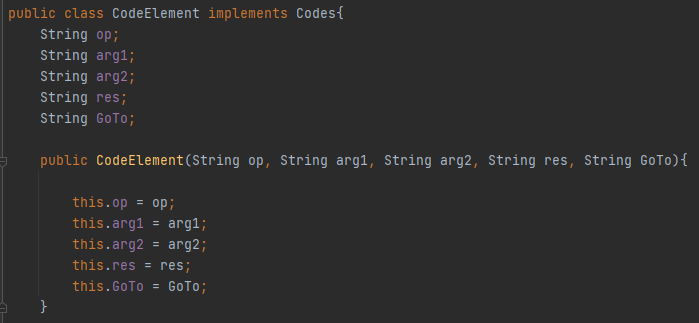


Рисунок 9 – Объект CodeElement

Промежуточные инструкции создавались с целью перевода их в инструкции под архитектуру MIPS, поэтому генератор промежуточного кода превратился в генератор целевого кода. При генерации добавлялись операции переноса значения переменной в регистр и наоборот, а так же сохранения значений локальны переменных и регистра $ra (адрес возврата) в стек, для того чтобы они не были перезаписаны при вызове функции. Так же в целевой код записываются значения строковых и нецелочисленных констант, для дальнейшей корректной работы.

Пример работы генератора представлен на рисунках 10 – 11:

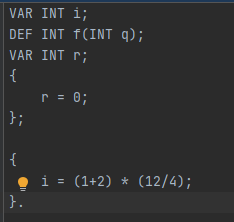


Рисунок 10 – Исходный текст

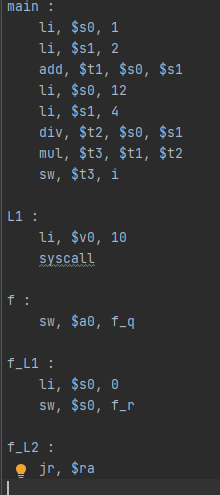


Рисунок 11 – Сгенерированный код

В результате работы компилятора получается файл с расширение .asm. Для запуска полученых команд был использован эмулятор MARS. После открытия файла ассемблируем его и запускаем. В результат компиляции программы рисунок 12 представлен на рисунке 14:

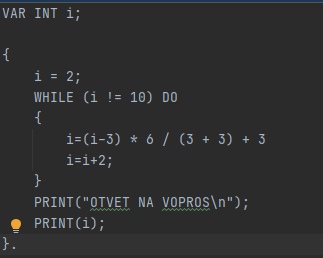


Рисунок 12 – Исходный код программы на языке safron++

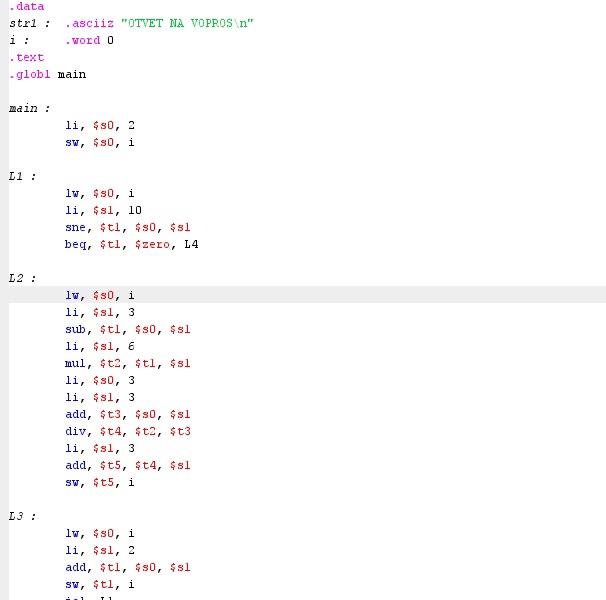


Рисунок 13 – Список инструкции на MIPS

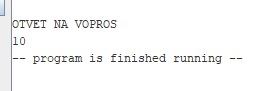


Рисунок 14 – Результат работы программы

# 2 ПРОВЕРКА ВЫПОЛНЕНИЯ ТРЕБОВАНИЙ К ЯЗЫКУ

Пример выводит первые 10 чисел последовательности Фибоначчи показывает выполнение следующих требований:

- Операторы присваивания

- Арифметика

- Операторы цикла (while)

- Условный оператор

- Базовый вывод переменных и строковых литералов

- Поддержка вызова функций и процедур

- Игнорирование индентации программы

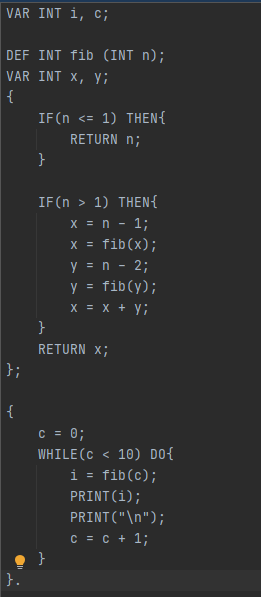
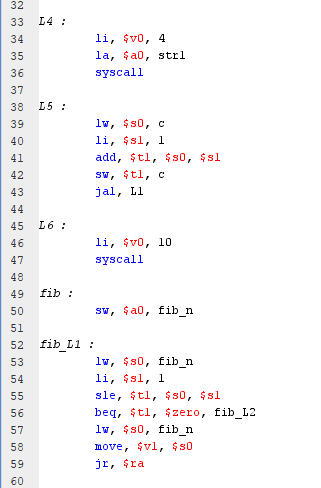
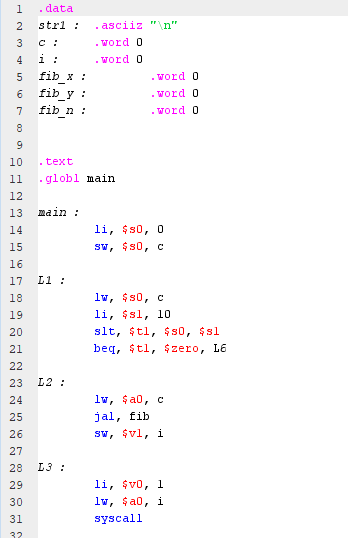
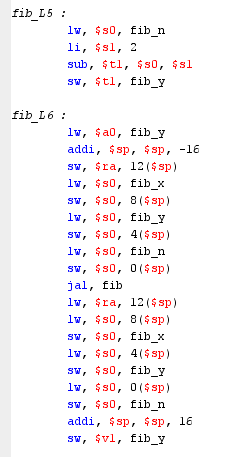
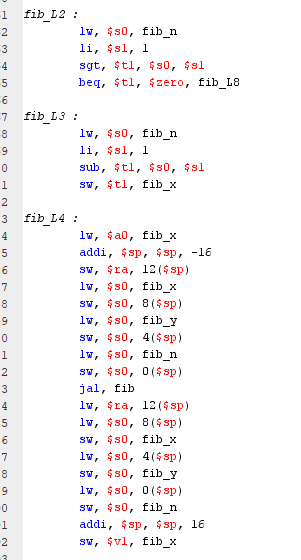


Рисунок 15 – Исходный код программы fib.s





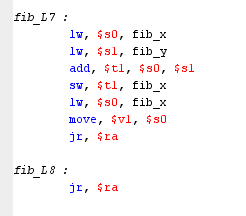


Рисунок 16 – MIPS код

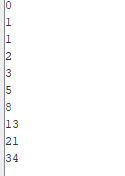


Рисунок 17 – Результат

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ульман Д. Д. Компиляторы: принципы, технологии и инструментарий, 2-е изд./ Ульман Д. Д. , Ахо А.В., Лам С.М., Сети Р.-М. :ООО «И.Д. Вильямс», 2008.- 1184 с.
2. Appplemac «Изучаем MIPS-ассемблер» /Хабр [Электронный ресурс] Доступ: URL - https://habr.com/ru/post/147685/
3. MIPS Technologies, Inc. «MIPS Instruction Set» / MIPS Technologies, Inc. [Электронный ресурс] Доступ: URL -https://www.dsi.unive.it/~gasparetto/materials/MIPS\_Instruction\_Set.pdf