Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «**Информатика и системы управления (ИУ)**»

КАФЕДРА «**Программное обеспечение ЭВМ и информационные технологии (ИУ7)**»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Разработка компилятора языка tinyc»

Студент группы **ИУ7-21М** _____ **В.А. Иванов**
(Подпись, дата) (И.О. Фамилия)

Руководитель _____ **А.А. Ступников**
(Подпись, дата) (И.О. Фамилия)

***2023** г.*

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине Конструирование компиляторов

Студент группы ИУ7-21М

Иванов Всеволод Алексеевич
(Фамилия, имя, отчество)

Тема курсового проекта Разработка компилятора языка tinyc

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) Кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Описать грамматику языка tinyc, расширить её поддержкой массивов с помощью грамматики языка Си. Разработать компилятор расширенного языка tinyc. В качестве фронтенда компилятора использовать утилиту ANTLR4 для преобразования исходного кода в синтаксическое дерево в соответствии с описанной грамматикой. В качестве бекенда компилятора использовать LLVM для генерации промежуточного представления на основе составленного синтаксического дерева.

Оформление курсового проекта:

Расчетно-пояснительная записка на 20-30 листах формата А4.
Расчетно-пояснительная записка должна содержать постановку задачи, введение, аналитическую, конструкторскую, технологическую части, заключение и список литературы.

Дата выдачи задания « 2 » марта 2023 г.

Руководитель курсового проекта _____ А.А.Ступников
(Подпись, дата) (И.О.Фамилия)

Студент _____ В.А.Иванов
(Подпись, дата) (И.О.Фамилия)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Компилятор — программа, переводящая написанный на языке программирования текст в набор машинных кодов[1].

Целью данного курсового проекта является разработка компилятора для языка программирования tinyc. В данной курсовой работе грамматика данного языка будет расширена за счёт элементов грамматики языка Си. Это делается для удволетворения требований к курсовому проекту по наличию более сложных элементов языка, чем имеющиеся в данной грамматике.

Основные задачи, которые необходимо выполнить в рамках данного проекта:

1) Провести анализ существующей грамматики языка tinyc, и расширить ее поддержкой массивов, используя грамматику языка Си.
2) Разработать лексический и ситнтаксический анализатор с использованием утилиты ANTLR4.
3) Разработать семантический анализатор для генерации промежуточного представления LLVM.
4) Провести тестирование компилятора.

1 Аналитическая часть

1.1 Составляющие компилятора

Компилятор состоит из следующих составляющих подпрограмм:

- Frontend компилятора отвечает за первичную обработку исходного кода и создание внутреннего представления программы. Он состоит из следующих частей:
  - препроцессор;
  - лексический анализатор;
  - синтаксический анализатор;
  - семантический анализатор;
  - генератор промежуточного представления;
- Middle-end компилятора занимается оптимизацией и преобразованием промежуточного представления программы.
- Backend компилятора отвечает за генерацию целевого кода, который может быть выполнен на конкретной аппаратной платформе или виртуальной машине.

В данной работе функции Middle-end и Backend компилятора будут осуществляться библиотекой LLVM, поэтому рассмотрим более подробно Frontend составляющих компилятора.

1.1.1 Препроцессор

Препроцессор компилятора - это компонент компилятора, который выполняет предварительную обработку исходного кода перед фазой фронтенда. Его задача заключается в обработке директив препроцессора и внесении соответствующих изменений в исходный код.

Препроцессор предоставляет набор директив, которые позволяют включать или исключать определенные части исходного кода, задавать макросы для замены текста и включать заголовочные файлы. Примером директив языка Си являются **include**, **define**, **pragma**. После работы препроцессора изменённый

исходный код программы подаётся на вход лексический анализатора.

В данном проекте препроцессор не используется ввиду его избыточности.

1.1.2 Лексический анализатор

Лексический анализатор выполняет первичную обработку исходного кода, разбивая его на лексемы. Лексема - минимальный элемент исходного кода. Примеры: ключевые слова, идентификаторы, операторы, константы и символы пунктуации.

Задачи лексического анализатора:

- разбиение исходного кода на лексемы;
- идентификация типов лексем;
- удаление незначащих символов;
- формирование потока токенов для синтаксического анализатора;

1.1.3 Синтаксический анализатор

Синтаксический анализатор выполняет построение синтаксического дерева из полученного потока токенов, которое представляет иерархическую структуру программы. Обычно это представление выражается в виде абстрактного синтаксического дерева (AST), где каждый внутренний узел является оператором, а дочерние его аргументами.

Задачи синтаксического анализатора:

- проверка синтаксической корректности (соответствие грамматике);
- построение синтаксического дерева;
- обработка ошибок.

Полученное представление программы в виде синтаксического дерева используется на следующем этапе.

1.1.4 Семантический анализ

Семантический анализатор выполняет проверку семантики исходного кода, включая правильное использование типов данных, правила области видимости и согласованность операций.

Задачи семантического анализатора:

- установить семантическую связь между различными частями программы;
- выявить потенциальные ошибки и несоответствия типов.

Семантический анализатор составляет таблицу символов, описывающую хранящиеся типы данных.

1.1.5 Генерация кода

Генерация кода - это фаза компиляции, в которой основываясь на синтаксическом дереве программы и системных таблиц создаётся её код.

Получение машинного кода осуществляется в два этапа.

1) Генерация промежуточного кода - относится к последней фазе frontend компилятора.
2) Генерация машинного кода - относится к middle-end и backend компилятора.

Основные этапы генерации кода включают:

- оптимизация промежуточного представления;
- выбор инструкций целевой платформы, соотвествующие промежуточному представлению;
- связывание данных с именами переменных;
- собственно генерация кода.

Результатом этого этапа является исполняемый на целевой платформе код.

1.2 ANTLR4

В качестве лексического и синтаксического анализатора будет использован ANTLR4 (ANother Tool for Language Recognition). Выбор обосновывается рядом преимуществ и особенностей данного инструмента:

- поддержка генерацию лексических и синтаксических анализаторов для широкого спектра языков программирования;
- удобный и интуитивно понятный синтаксис для описания грамматик языков программирования;
- автоматическая генерация синтаксического дерева;

- широкая и активная пользовательская база и развитое сообщество разработчиков.

1.3 LLVM

В качестве генератора кода используется LLVM (Low Level Virtual Machine). Его выбор обосновывается следующими факторами:

- Поддержка большого количества целевых платформ.
- Поддержка библиотек на различных языках (C, C++, Rust, Python и другие).
- Поддержка основных типы данных: целые числа, числа с плавающей точки различных точностей, массивы, структуры, функции.
- Автоматическая оптимизация сгенерированного промежуточного представления.
- Широкая и активная пользовательская база и развитое сообщество разработчиков.
- Имеется интерпретатор промежуточного представления.

Выводы

В данном разделе был проведён обзор основных фаз компиляции. Обоснованы выборы средств лексического и синтаксического анализа - ANTLR4 и генератора машинного кода - LLVM. Метод работы компилятора будет заключаться в генерации синтаксического дерева и генерации по нему промежуточного представления кода (LLVM IR).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19781-83 // Вычислительная техника. Терминология: Справочное пособие. Выпуск 1 / Рецензент канд. техн. наук Ю. П. Селиванов. — М.: Издательство стандартов, 1989. — 168 с. — 55 000 экз. — ISBN 5-7050-0155-X.;