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**_____

КАФЕДРА _____**КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ**_____

НАПРАВЛЕНИЕ ПОДГОТОВКИ __**09.03.01 Информатика и вычислительная техника** _____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Компилятор для языка программирования на основе обратной польской записи

Студент _ИУ6-53Б_____ _____ **_____В.К. Залыгин_**
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы _____ **_____Б.И. Бычков___**
(Подпись, дата) (И.О. Фамилия)

2024 г.

РЕФЕРАТ

Расчетно-пояснительная записка состоит из 13 страниц, включающих в себя 1 рисунков, 1 таблиц, 0 источников и 2 приложения.

КОМПИЛЯТОР, СТЕКОВЫЙ ЯЗЫК, ОБРАТНАЯ ПОЛЬСКАЯ ЗАПИСЬ, LINUX, АРХИТЕКТУРА X64.

Объектом разработки является приложение-компилятор с исходного языка в машинный код архитектуры x64.

Цель работы – проектирование и реализация компилятора для стекого языка с синтаксисом на основе обратной польской записи, позволяющего создавать исполняемые файлы для целевой архитектуры x64.

Разрабатываемое программное обеспечение предназначенно для программистов, создающих программы на исходном языке. Область применения – создание программ алгоритмов обработки данных.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время существует ряд языков, синтаксис которых основан на обратной польской нотации (постфиксной нотации). Такие языки используют для описание программ для стековых машин – вычислительных устройств, которые оперируют при работе операруют стеком, в противовес регистровым машинам, оперирующим регистрами. Языки, описывающие алгоритмы для стековых машин, называют стековыми. Одна из сфер применения стековых языков – описания алгоритмов обработки данных. Стековые языки позволяют более лаконично и кратко описывать алгоритмы благодаря иной парадигме работы с контейнерами данных – в программах переменные отсуствуют и все операции последовательно выполняются над одним контейнером, стеком.

Поскольку стековые машины, в отличие от регистровых, не получили широкого распространения (TODO ссылка), существует задача компиляции кода на стековом языке под целевую регистровую архитектуру.

Таким образом, предметная область, в рамках которой ведется работа, – компиляторы для стековых языков, служащих для описания алгоритмов обработки данных.

В рамках данной курсовой работы решается задача создания компилятора для стекового языка на основе обратной польской записи (далее, исходный язык) под целевую платформу Linux x64. Целевая платформа выбрана за счет своей широкой распространенности (TODO: ссылка). К компилятору для соответствия предметной области предъявляются требования по грамматике распознаваемого языка: наличие операций ввода-вывода, полнота по Тьюрингу (иными словами – наличие условных переходов и циклов/рекурсии). Также к решению предъявляются функциональные требования:

– создание исполняемых файлов из кода исходного языка;

– сборка объектный файлов из кода исходного языка;

– составление ассемблерных листингов кодов на исходном языке.

При сравнении с существующими решениями преимуществом данной разработки является использование современных инструментов и парадигм,

что позволяет значительно снизить количество ошибок в программном обеспечении.

1 Анализ требований и уточнение спецификаций

1.1 Анализ задания и выбор технологии, языка и среды разработки

В соответствии с требованиями технического задания необходимо разработать программу, которая позволит компилировать код на исходном языке в исполнямые файлы, собирать в объектные файлы и транслировать в ассемблерные листинги. Компилятор должен обеспечивать поддержку ряда синтаксических конструкций, представляющих исходный язык и перечисленных в техническом задании. Исполняемые файлы, объектные файлы, ассемлерные листинги, являющиеся результатом работы компилятора, должны соответствовать набору команд x86-64. Программное обеспечение должно работать под управлением ОС Linux и иметь интерфейс командной строки.

Исторически к программам-компиляторам предъявляются требования по скорости работы, нативности, наличию интерфейса командной строки. Иными словами, привычный компилятор – скомпилированное нативное CLI-приложение без сборщика мусора. При разработке решения учитываются общие требования к программному обеспечению данной направленности.

Вышеперечисленные требования сужают диапазон подходящих языков программирования до нескольких штук: C, C++, Rust, Zig. В результате по совокупности факторов был выбран язык Rust. Компилятор данного языка обеспечивает автоматический контроль за состоянием памяти на этапе компиляции, сам язык обладает наиболее строгой системой типов (среди предложенных). Указанные особенности Rust позволяют писать безопасное программное обеспечение (TODO: рофл линк на новость про безопасные языки США) и недопустить ошибки при разработке программы. В таблице 1 показаны результаты сравнения языков программирования.

Для разработки на данном языке принято использовать Visual Studio Code, поэтому она выбрана в качестве среды разработки.

Поскольку процессы в рамках предметной области (создание исполняемых файлов, объектных файлов, ассемблерных листингов) можно описать как последовательность вызовов функций, постепенно преобразующих код от исходного языка до исполняемого файла, рационально использовать структурный

Таблица 1 — Сравнение свойств языков программирования

	C	C++	Zig	Rust
Работа с памятью	Ручная	Ручная	Ручная	Автоматическая
Компиляция в нативный код	Да	Да	Да	Да
Зрелость и стабильность	Да	Да	Нет	Скорее да
Современные методы разработки	Нет	Да	Да	Да

подход. Структурный подход также является идиоматичным при разработке на Rust.

1.2 Разработка фунциональной диаграммы

В соответствии с техническим заданием программное решение должно обеспечивать создание различных выходных файлов.

Для разработки решения необходимо разложить процесс создания выходных файлов на этапы.

Процесс создания ассемблерного листинга можно разбить на 2 этапа:

– разбор кода программы;

– трансляция в языка ассемблера (компиляция).

В случае, если необходимо собрать объектный файл, то к 2 этапам создания ассемблерного листинга добавляется еще один этап – «компоновка объектного файла».

В случае, если необходимо сделать исполняемый файл, то к 3 этапам сборки объектного файла добавляется еще один этап – «компоновка исполняемого файла».

TODO: обновить диаграммы и вставить

1.3 Разработка диаграммы вариантов использования

Поскольку техническое задание предполагает реализацию различные варианты использования программы, целесообразно показать их на диаграмме вариантов использования. Рисунок 1 показывает возможности использования программного обеспечения.

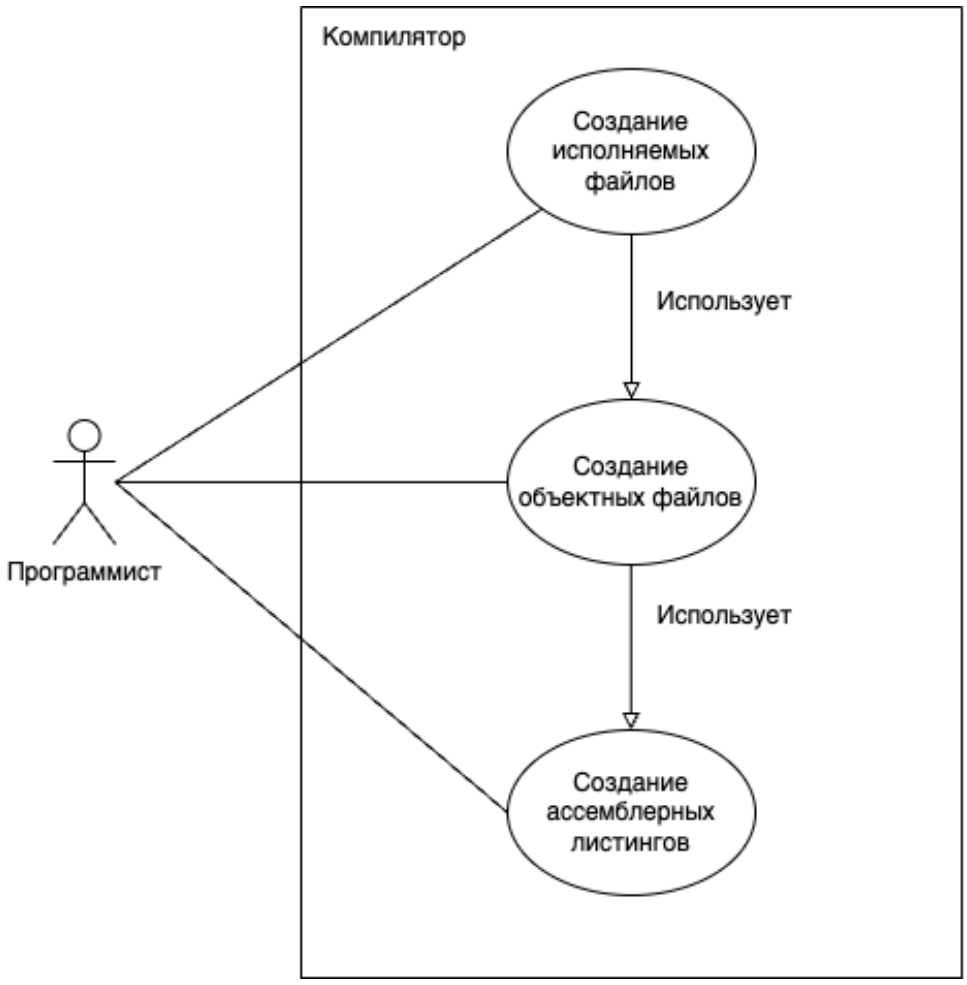

Рисунок 1 — Диаграмма вариантов использования

1.4 Разработка синтаксических диаграмм

2 Проектирование структуры и компонентов программного продукта

3 Выбор стратегии тестирования и разработка тестов

ЗАКЛЮЧЕНИЕ

ПРИЛОЖЕНИЕ А

ПРИЛОЖЕНИЕ Б