Задание к проекту по курсу «Разработка компиляторов».

Необходимо разработать компилятор для учебного языка программирования.

Вход компилятора: программа на писанная на учебном языке программирования.

Выход компилятора: текст программы на языке ассемблера. Предлагается использовать эмулятор RISC-V процессора (ссылка на README в репозитории эмулятора https://github.com/asurkis/risc-emulator). По согласованию с преподавателем язык целевой машины может быть изменен.

Характеристики учебного языка программирования (императивный тьюринг полный язык), поддерживающий следующий набор операторов и выражений:

- Операторы

  1. Определения переменных. Поддерживаемые типы данных: целые числа со знаком, строки (можно ограничиться набором ASCII символов из первой половины таблицы коды от 32 до 127 и управляющими символами переноса строки и пр.).

  2. Присваивание (может быть заменено выражением в случае определения оператора-выражения).

  3. Ветвление (`if`) с факультативным `else`.

  4. Цикл `while`.

  5. Блок (составной оператор).

  6. Вывода (типа `print`).

- Выражения

  1. Арифметические (минимум + * - /).

  2. Логические not, and, or.

  3. Для чего поддерживать операции (operator) арифметические, логические сравнения.

Допускается использовать ключевые слова для определения операторов и операций отличные от общепринятых. Полный список операторов, операций и ключевых слов должен быть приведен в отчете.

Для разработанного учебного языка программирования необходимо:

- Определить лексический состав языка и описать правила определения лексем для генератора лексических анализаторов flex. Учесть обработку комментариев: однострочных и многострочных.
- Определить разработанный язык в терминах КС грамматики в формате входного файла для GNU Bison.
- Для каждого правила грамматики (в рамках синтаксически управляемой трансляции) определить семантические действия, определяющие правила построения AST.
- Разработать функцию обхода AST для его визуализации. Для построения AST использовать примеры корректных программ (небольших) для учебного языка.

На этом завершается первый этап, его результаты можно обсудить со мной, если возникнут проблемы.

Второй этап предполагает генерацию кода в процессе обхода AST.

Можно построить промежуточное представление программы в виде трехадресного кода, если вам это будет удобно. Трехадресный код не является целью проекта.

Цель второго этапа - получение ассемблерной программы эквивалентной тексту входной программы.

Проект можно выполнять в парах. Полученная на защите оценка ставится обоим участникам. Выбор языка реализации оставляю за вами.

По итогам будет нужно оформить отчет с описанием лексического и синтаксического состава, демонстрация работы компилятора для корректных и ошибочных программ. (вопросы обработки ошибок обсудим).