1. 预备阶段

实验任务：针对一个自定义类C语言，基于LLVM框架实现该语言的编译器。要求：

1. 实现词法分析，并输出词法信息，至少包括（单词，term类型）
2. 实现语法分析，并输出语法分析树
3. 实现代码生成，并输出对应的LLVM IR代码
4. 增加JIT支持
5. 利用LLVM后端，生成ARM机器代码
6. 支持优化
   1. 熟悉实验环境
      1. LLVM介绍
7. Llvm.org
8. Ppt
   * 1. LLVM的安装
9. Linux

参考官网手册

1. Windows

参考官网手册

* + 1. Kaleidoscope项目

一个入门教程：如何基于LLVM实现一个简单的自定义语言的编译器。

1. 手动实现词法分析器
2. 手动实现递归下降语法分析器
3. 生成LLVM IR代码，并借助LLVM后端，生成机器代码
4. 调试信息支持
5. JIT支持和优化
   * 1. LLVM IR

<https://llvm.org/docs/LangRef.html>

建议：不必详细审读，仅作参考用

* 1. 类C语言：VSL

VSL(Very Simple Language)是一个最简单的过程式程序设计语言,它具有一般过程式程序设计语言的特点: 块结构, 支持函数的递归调用和具有基本控制结构(如, while语句, if语句等).

* + 1. 词法

comment "//".\*

delimiter [ \t\n]

whitespace {delimiter}+

uc\_letter [A-Z]

lc\_letter [a-z]

letter {lc\_letter}|{uc\_letter}

ascii\_char [^\"\n]

escaped\_char \\n|\\\"

digit [0-9]

variable {lc\_letter}({lc\_letter}|{digit})\*

integer {digit}+

text \"({ascii\_char}|{escaped\_char})\*\"

* + 1. 语法

program : function\_list

;

function\_list : function

| function\_list function

;

function : FUNC VARIABLE '(' parameter\_list ')'

statement

;

parameter\_list : variable\_list

|

;

variable\_list : VARIABLE

| variable\_list ',' VARIABLE

;

statement : assignment\_statement

| return\_statement

| print\_statement

| null\_statement

| if\_statement

| while\_statement

| block

;

assignment\_statement : VARIABLE ASSIGN\_SYMBOL expression

;

expression : expression '+' expression

| expression '-' expression

| expression '\*' expression

| expression '/' expression

| '-' expression

| '(' expression ')'

| INTEGER

| VARIABLE

| VARIABLE '(' argument\_list ')'

;

argument\_list :

| expression\_list

;

expression\_list : expression

| expression\_list ',' expression

;

print\_statement : PRINT print\_list

;

print\_list : print\_item

| print\_list ',' print\_item

;

print\_item : expression

| TEXT

;

return\_statement: RETURN expression

; ;

null\_statement : CONTINUE

;

if\_statement : IF expression THEN statement FI

| IF expression THEN statement

ELSE statement FI

;

while\_statement : WHILE expression DO statement DONE

;

block : '{' declaration\_list statement\_list '}'

;

declaration\_list:

| declaration\_list declaration

;

declaration : VAR variable\_list

;

statement\_list : statement

| statement\_list statement

;

* + 1. 一个例子程序

本程序通过一个函数的递归调用计算10的阶乘:

FUNC main()

{

VAR i

i := 0

WHILE 11 - i

DO

{

PRINT "f( ", i, " ) = ", f(i), "\n"

i := i + 1

}

DONE

}

FUNC f(n)

{

IF n

THEN

RETURN n \* f(n - 1)

ELSE

RETURN 1

FI

}

* + 1. VSL的语义

VSL程序的执行将从第一个定义的函数开始, 如上例中, 程序将从main()

开始执行, VSL所有的变量都是隶属于函数中申明该变量的块结构block的

局部变量, 函数的参数用传值的方法进行传递, 函数在遇到返回语句

return或函数体结束时返回.

表达式的计算和通常的表达式的计算相同. if和while中的条件表达式在

表达式的值是非零时表示真, 为零时表示假.

* 1. 实验方式

鼓励合作，小于等于5人

* 1. 实验结果
     1. 代码

提供代码版本网址，可访问下载使用代码管理（建议github or similar），需要看到所有成员的代码提交，以及代码版本日志。

* + 1. 文档

1. 设计文档
2. 测试文档
3. 代码配置、安装、运行方法的说明文档
4. 各成员的分工和贡献
5. 词法分析
6. 词法分析
7. 语法分析
8. IR代码生成