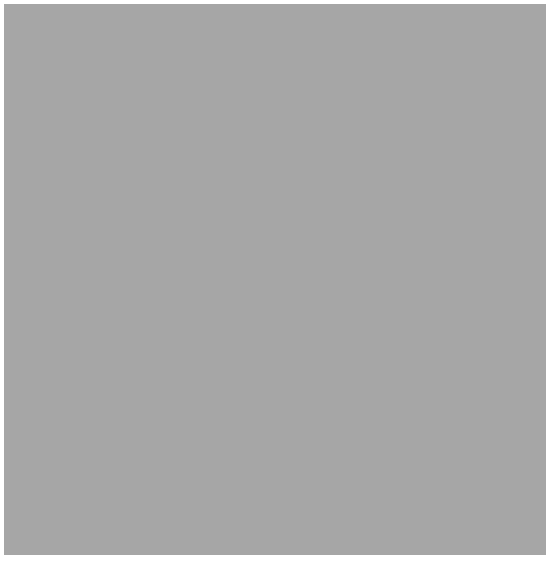
编译器构造实验

2019级计算机专业

目的

- 通过编译器相关子系统的设计，进一步加深对编译器构造的理解；
- 培养学生独立分析问题、解决问题的能力，以及系统软件设计的能力；
- 提高程序设计能力、程序调试能力



任务（题目）

- 一个简单文法的编译器的设计与实现
  - 一个简单文法的编译器前段的设计与实现
    - 定义一个简单程序设计语言文法（包括变量说明语句、算术运算表达式、赋值语句；扩展包括逻辑运算表达式、If语句、While语句等）；
    - 扫描器设计实现；
    - 符号表系统的设计实现；
    - 语法分析器设计实现；
    - 中间代码设计；
    - 中间代码生成器设计实现。
  - 一个简单文法的编译器后段的设计与实现
    - 中间代码的优化设计与实现（鼓励）；
    - 目标代码的生成（使用汇编语言描述，指令集自选）；
    - 目标代码的成功运行。

参考书

- 陈火旺.《程序设计语言编译原理》（第3版）.北京：国防工业出版社.2000.
- 美 Alfred V.Aho Ravi Sethi Jeffrey D. Ullman著.李建中，姜守旭译.《编译原理》.北京：机械工业出版社.2003.
- 美 Kenneth C.Louden著.冯博琴等译.《编译原理及实践》.北京：机械工业出版社.2002.
- 金成植著.《编译程序构造原理和实现技术》.北京：高等教育出版社. 2002.
- 有关编译原理系统分析与设计的书

一个简单文法的编译器前端的设计与实现

- 内容
  - 定义一个简单程序设计语言文法（包括变量说明语句、算术运算表达式、赋值语句；扩展包括逻辑运算表达式、If语句、While语句等）；
  - 扫描器设计实现；
  - 语法分析器设计实现；
  - 中间代码设计；
  - 中间代码生成器设计实现。

一个简单文法的编译器前端的设计与实现

- 要求
  - 给出一个源程序文件，作为编译器前端的输入
  - 输出相关编译阶段的运行结果
    - 词法分析阶段：
      - Token序列；
      - 关键字表、界符表、符号表系统。
    - 中间代码生成阶段：
      - 四元式序列；
      - 符号表系统。

一个简单文法的编译器前端的设计与实现

- 定义一个简单程序设计语言文法
- 其中包括变量说明语句、算术运算表达式、赋值语句。
  - <程序> program <标识符><分程序>.
  - <分程序> <变量说明><复合语句>
  - <变量说明> var <标识符表>：<类型>；
  - <标识符表> <标识符>，<标识符表>|<标识符>
  - <复合语句> begin <语句表> end
  - <语句表> <赋值语句>；<语句表>|<赋值语句>
  - <赋值语句> <标识符>:=<算术表达式>
  - <算术表达式> <算术表达式> ω0 <项>|<项>
  - <项> <项> ω1 <因子>|<因子>
  - <因子> <算术量>|(<算术表达式>)
  - <算术量> <标识符>|<常数>
  - <类型> integer | real | char

一个简单文法的编译器前端的设计与实现

- 定义一个简单程序设计语言文法，
- 文法中以下部分以自动机实现。
  - <标识符> <字母>|<标识符><数字>|<标识符><字母>
  - <常数> <整数>|<实数>
  - <整数> <数字>|<整数><数字>
  - <实数> <整数>.<整数>
  - <字母> A|B|C|…|Z|a|b|c|…|z
  - <数字> 0|1|2|3|4|5|6|7|8|9

一个简单文法的编译器前端的设计与实现

扫描器设计实现

关键字与界符表

编号	关键字	编号	界符
1	program	1	,
2	var	2	:
3	integer	3	;
4	real	4	:=
5	char	5	*
6	begin	6	/
7	end	7	+
		8	-
		9	.
		10	(
		11	)

一个简单文法的编译器前端的设计与实现

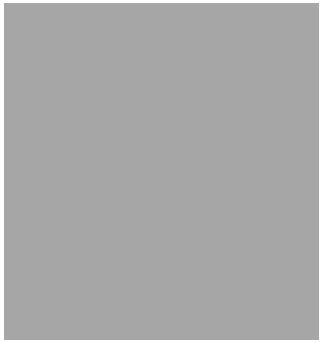
扫描器设计实现

符号表系统

符号表主表

名字	类型	种类	地址

常数表



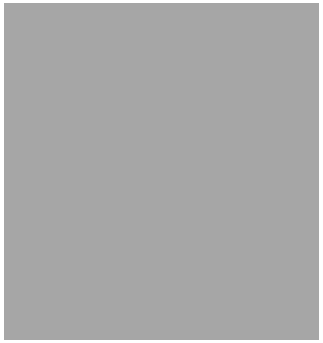
活动记录

临时变量
局部变量

0

一个简单文法的编译器前端的设计与实现

- 扫描器设计实现
  - 标识符---字母开头，后跟字母或数字字符的符号串
  - 常数---数值常数，包括整数、实数。
- 扫描器具体设计与实现参见实验一。



一个简单文法的编译器前端的设计与实现

语法分析器设计实现

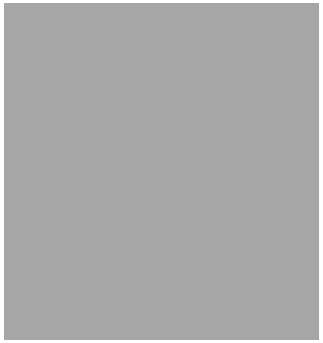
文法

PROGRAM　program id SUB_PROGRAM.
SUB_PROGRAM　VARIABLE COM_SENTENCE
VARIABLE　var ID_SEQUENCE：TYPE；
ID_SEQUENCE　id {，id}
TYPE　integer | real | char
COM_SENTENCE　begin SEN_SEQUENCE end
SEN_SEQUENCE　EVA_SENTENCE {；EVA_SENTENCE }
EVA_SENTENCE　id := EXPRESSION
EXPRESSION　EXPRESSION + TERM | EXPRESSION - TERM | TERM
TERM　TERM * FACTOR | TERM / FACTOR | FACTOR
FACTOR　id | cons | (EXPRESSION)

其中：id为标识符，cons为常数。

一个简单文法的编译器前端的设计与实现

- 语法分析器设计实现
  - 递归下降子程序（参见word文档“一个简单语言的编译实例”）
- 中间代码设计---四元式
  - 操作四元式---（op, ob1, ob2, t）
  - 赋值四元式---（:=, ob, , v）



一个简单文法的编译器前端的设计与实现

中间代码生成器设计实现

翻译文法（以变量说明语句为例）

原文法：

- **VAR　　var ID_SEQ ： TYPE ；**
- **ID_SEQ　　id { , id }**
- **TYPE　　integer | real | char**

翻译文法：

- **VAR　　var “a1” ID_SEQ ： TYPE “a6” ；**
- **ID_SEQ　　id “a2” { , id “a2” }**
- **TYPE　　integer “a3” | real “a4” | char “a5”**

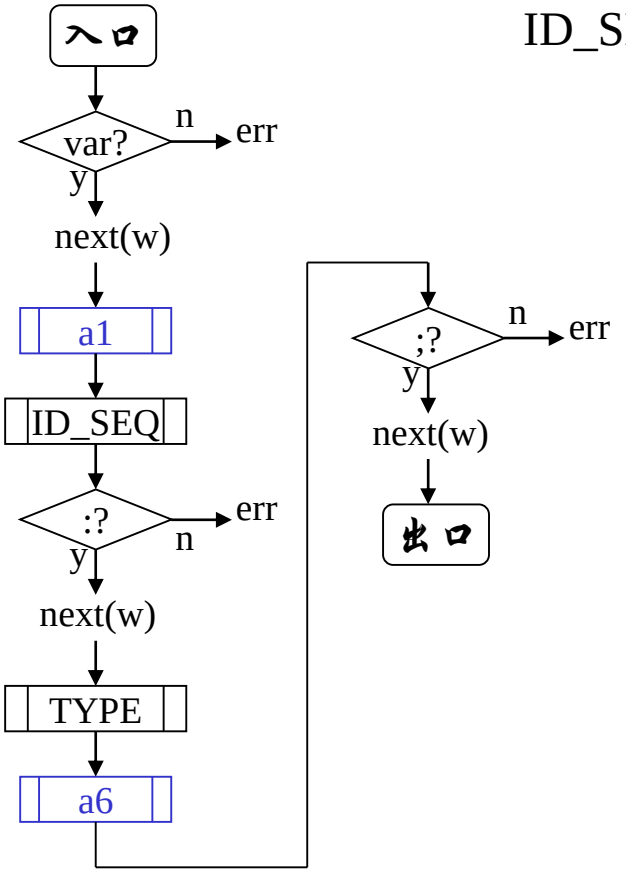
其中：a1— id.cat:=v; id.offset:=0;
a2— id.entry:=id_Token.val; push(id.entry);
a3— TYPE.type:=i; TYPE.len:=4;
a4— TYPE.type:=r; TYPE.len:=8;
a5— TYPE.type:=c; TYPE.len:=1;
a6— while (栈不空)
{ id.entry:=pop();
enter(id.entry, TYPE.type, id.cat, id.offset); //填写符号表
id.offset:=id.offset+ TYPE.len; }

一个简单文法的编译器前端的设计与实现

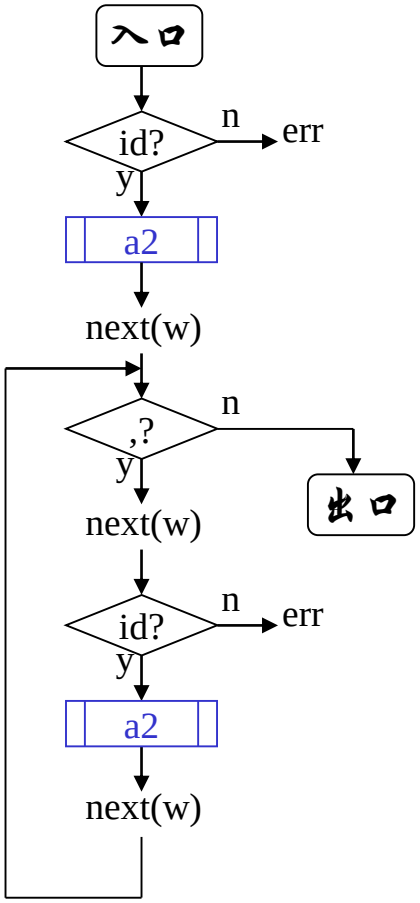
中间代码生成器设计实现

递归下降子程序实现（以变量说明语句为例）

VAR



ID_SEQ

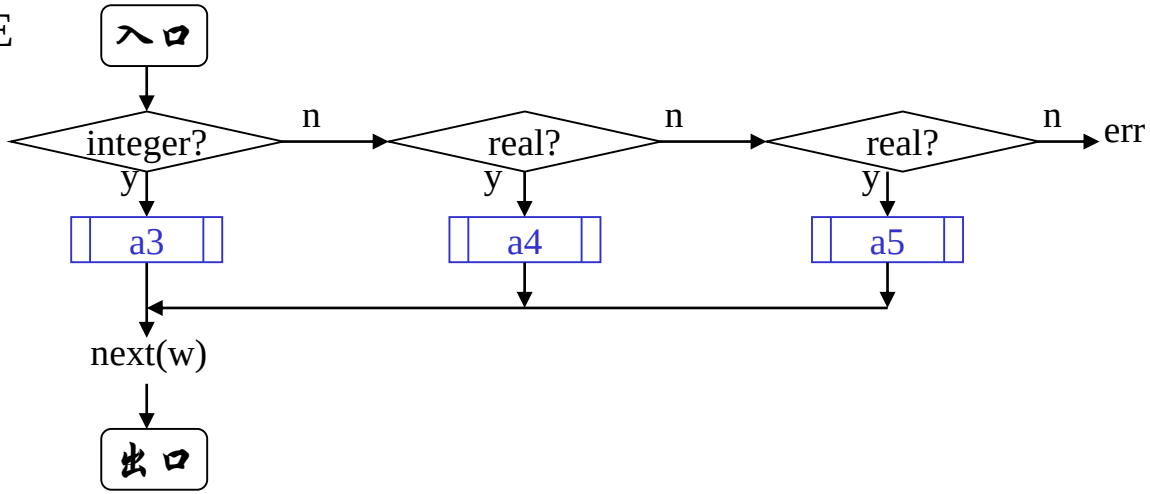


一个简单文法的编译器前端的设计与实现

中间代码生成器设计实现

递归下降子程序实现（以变量说明语句为例）

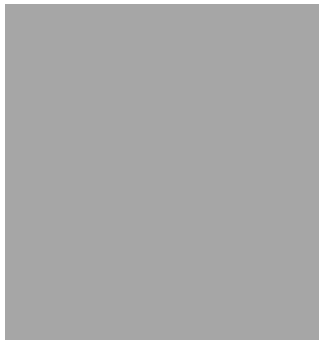
TYPE



一个实例

- 源程序：

```
program example
    var a,b:integer;
begin
    a:=2;
    b:=2*5+a
end.
```



一个实例

- **词法分析阶段输出：**

(k,1),(i,1), (k,2),(i,2),(p,1),(i,3),(p,2),(k,3),(p,3),(k,6),
(i,2),(p,4),(c,1),(p,3), (i,3),(p,4),(c,1),(p,5),(c,2),(p,7),(i,2),(k,7),(p,9),

关键字K表和界符P表

编号	关键字	编号	界符
1	program	1	,
2	var	2	:
3	integer	3	;
4	real	4	:=
5	char	5	*
6	begin	6	/
7	end	7	+
		8	-
		9	.
		10	(
		11	)

符号表I表

NAME	TYPE	CAT	ADDR
example			
a			
b			

常数C表

2
5

一个实例

- 中间代码生成阶段输出：

四元式区

(program, I1, _, _)
(:=, C1, _, I2)
(*, C1, C2, I4)
(+, I4, I2, I5)
(:=, I5, _, I3)
(end, I1, _, _)

符号表I表

	NAME	TYPE	CAT	ADDR
1	example		f	16
2	a	i	v	0
3	b	i	v	4
4	t1	i	v	8
5	t2	i	v	12

活动记录映像

12	t2
8	t1
4	b
0	a

常数C表

1	2
2	5

活动记录设计

(1) 连接数据区： 0～2；

·老SP — 主调过程的活动记录首址；

·全局display地址 — 主调过程的显示区表首址；

(2) 参数个数： 3；

(3) 形参值单元区： 入口为4；

·换名形参（vn）—
分配2个单元（地址传递）；

·赋值形参（vf）— 按相应类型长度分配；

(4) 显示区表(display)： 占l+1个单元；

l为层次号，包含直接外层嵌套的l个过程的活动记录的首址，再加上本过程的活动记录首址；

(5) 局部变量区： 入口为off + l + 2；

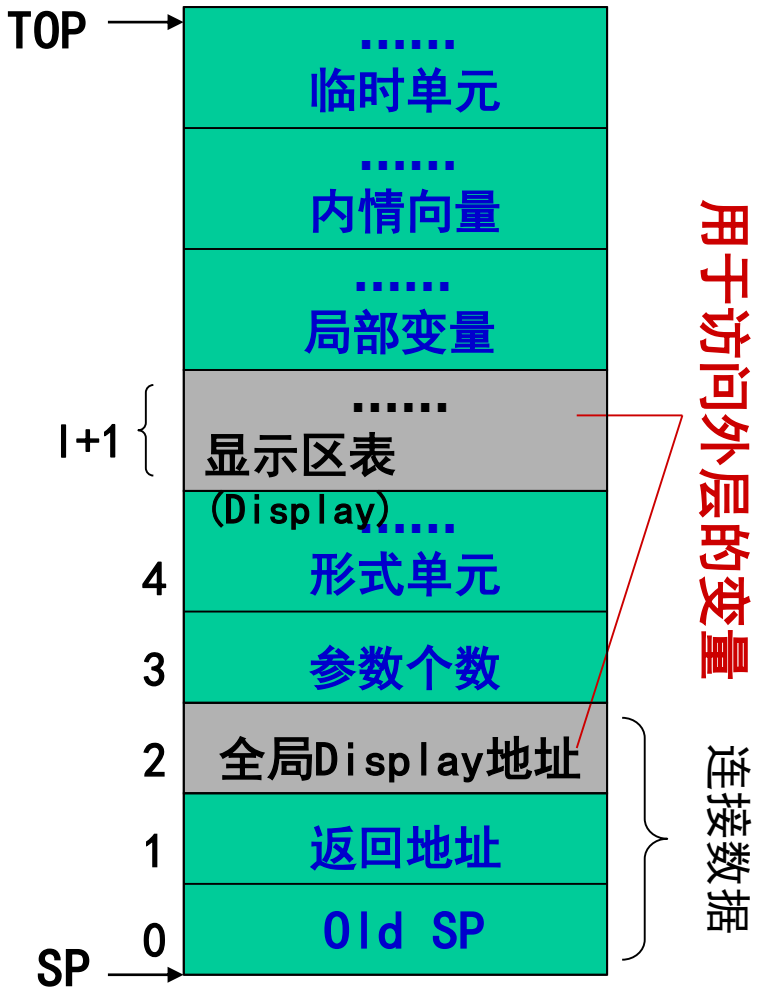
·off为形参区最后一个值单元地址；

·局部变量值单元按相应类型长度分配地址；

·类型标识符、常量标识符等不分配值单元；

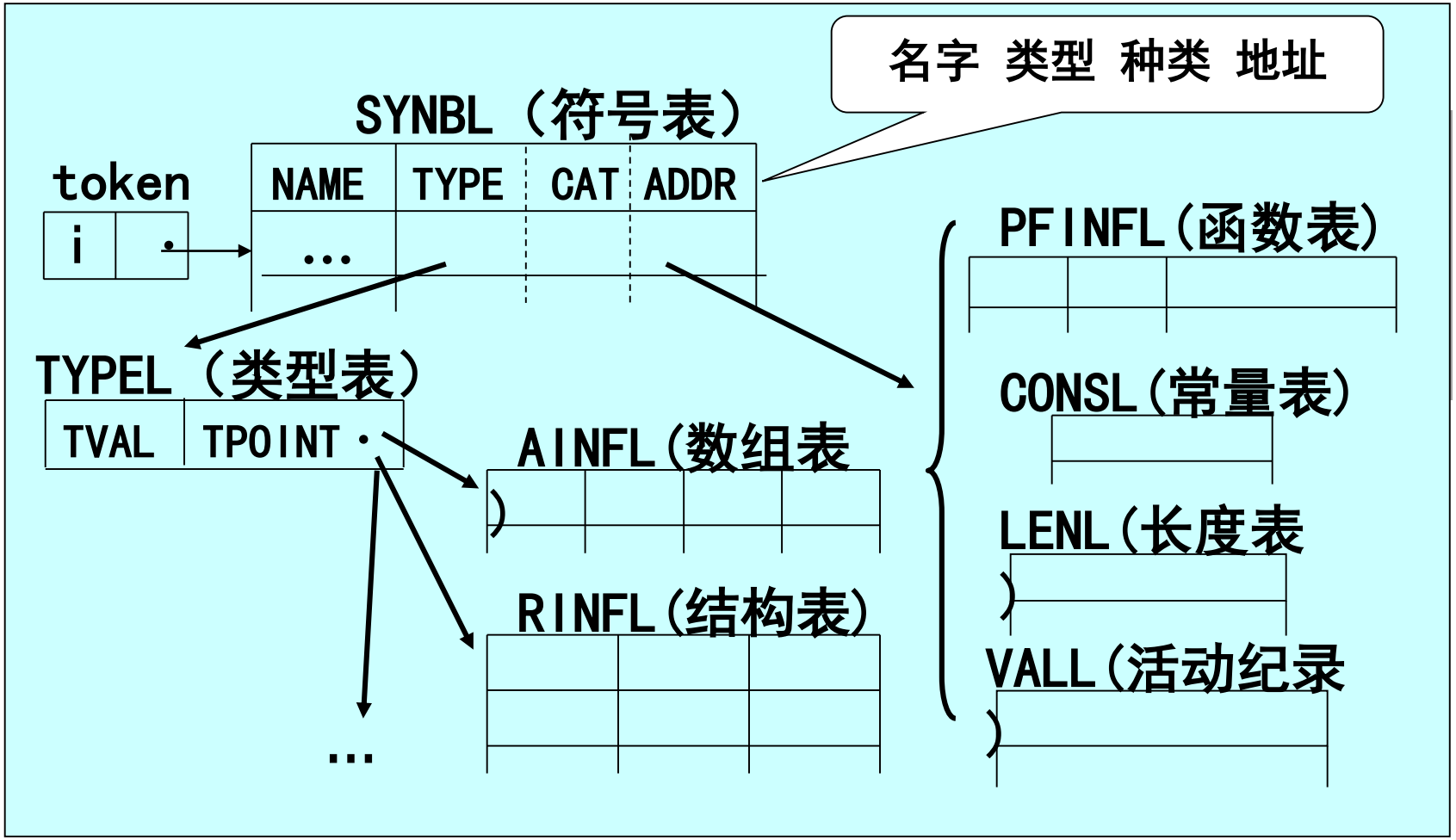
(6) 临时变量区：

编译系统定义的变量，按局部变量值单元分配原则分配地址；



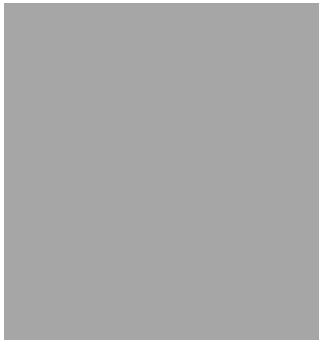
符号表系统设计

- 文法可以支持复杂数据结构和函数时的符号表：



一个简单文法的编译器后端的设计与实现

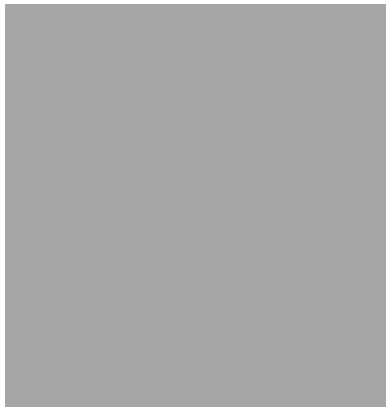
- 中间代码优化设计实现
  - 基于DAG的局部优化
  - 循环优化
- 目标代码生成
  - 选择自己熟悉的一套汇编指令集
  - 根据四元式序列结合指令集生成目标代码
    - 需要注意指令的选择
    - 寄存器的分配
    - 变量名的分配



提交相关要求

- 要求以个人为单位，提交以下内容：
  - 设计报告
  - 源代码压缩包
  - 可执行文件
  - 源码测试文件（待编译的程序，程序必须覆盖所支持的文法单元）
  - 测试输出文件（目标代码文件）
  - readme.txt（程序复现说明书）
- 提交时间
  - 截止时间为2022年7月15日





设计报告内容包括

- 作者名、学号
- 题目、内容、目的、设计方案、实现（数据结构设计、系统处理流程，可附加部分必要的源代码，禁止大篇幅贴源代码）、测试方法、测试用例和测试结果，目标代码运行结果，在报告中回答下面的问题：
  - 文法；
  - 符号表系统；
  - 活动记录；
  - 翻译文法；
  - 中间代码优化方法；
  - 目标代码指令集介绍。

设计报告内容包括

- 课程设计总结：收获、意见、建议等
- 参考文献：列出参考的主要文献资料
- 格式自拟
- 不得少于15页
- 外加封面：实验设计名、作者姓名、班级、学号…

给分标准（100分）

- 内容完整性（10%）
  - 压缩包中要求的内容
- 设计报告（50%）
  - 设计部分（25%）
    - 文法设计
    - 自动机设计
    - 翻译文法的设计
    - 符号表系统设计
    - 优化设计
    - 寄存器和变量分配
    - ……
  - 结果部分（25%）
    - 词法分析器——token序列
    - 语法分析器——程序的文法正确性判断
    - 语义分析器——中间代码生成
    - 中间代码优化——优化后的中间代码
    - 代码生成——生成的目标代码
    - 目标代码的运行成果
    - ……
- 源码功能可复现程度（30%）
- 美观性（10%）
  - 代码美观性（5%）
  - 报告美观性（5%）

说 明

- 设计的第一步是定义一个文法。样例文法是一个不错的起点，建议认真阅读。
- 符号表是一个编译器的核心数据结构，是重中之重，必须在一开始就给予足够的重视。
- 建议大家从一个简单的文法开始，完成相应的工作以后，再对你的文法进行扩充，比如增加复杂数据类型，增加If语句、While语句、子程序等，切勿一次把摊子铺的过大。
- 希望同学们认真调试，总结经验和问题。

预祝大家顺利实现自己的编译器