

**编译原理实验课程报告**

**题目：MiniC语言编译器设计与实现**

**专业班级： 网安2002班**

**学 号： U202012043**

**姓 名： 范启航**

**指导教师： 刘铭**

**报告日期： 2022年12月10日**

**网络空间安全学院**

**要 求**

1、实验代码及报告为本人独立完成，内容真实。如发现抄袭，成绩无效；如果引用资料，需将资料列入报告末尾的参考文献，参考文献格式按华中科技大学本科毕业论文规范，并在正文中标注参考文献序号；

2、按编译原理实验任务，内容应包含：工具入门、词法分析、语法分析、语义分析及中间代码生成、目标代码生成；

3、报告中简单说明解决问题的思路，特别是与众不同的、独特的部分；对设计实现中遇到的问题、解决进行记录；每个任务进行归纳、小结；最后，对整个实验进行总结，列出主要优点、不足；

4、评分标准：5个主要实验环节按任务要求完成；采用的方法合适、设计合理；能体出分析问题、灵活运用知识解决实际问题的能力；报告条理清晰、语句通顺、格式规范；

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **工具**  **入门** | **词法**  **分析** | **语法**  **分析** | **语义分析及中间代码生成** | **优化及目标代码生成** | **格式**  **规范** | **总分** |
| **10** | **20** | **30** | **20** | **10** | **10** | **100** |
|  |  |  |  |  |  |  |

目录

[一、 实验过程 4](#_Toc29324)

[1.1 Flex & Bison工具入门 4](#_Toc31832)

[1.2 MiniC词法分析 6](#_Toc20314)

[1.3 MiniC语法分析及语法树生成 7](#_Toc10774)

[1.4 MiniC语义分析及中间代码生成 10](#_Toc28618)

[1.5 MiniC代码优化及目标代码生成 11](#_Toc26982)

[二、 实验心得 13](#_Toc30492)

[三、 实验内容和过程的建议 14](#_Toc3656)

[参考文献 15](#_Toc14346)

# 实验过程

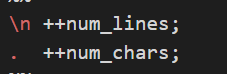
## Flex & Bison工具入门

（1）实验内容

Lex是Lexical Compiler的缩写，是Unix环境下非常著名的工具，主要功能是生成一个词法分析器(scanner)的C源码，描述规则采用正则表达式(regular expression)。本次实验使用Lex对minic进行词分析。

（2）实验过程task101: 对输入文件进行行数，字符树分析。

实现方法：规则段：匹配字符/n 则行数+1, 其他字符则，字符数+1，规则段如下图所示：



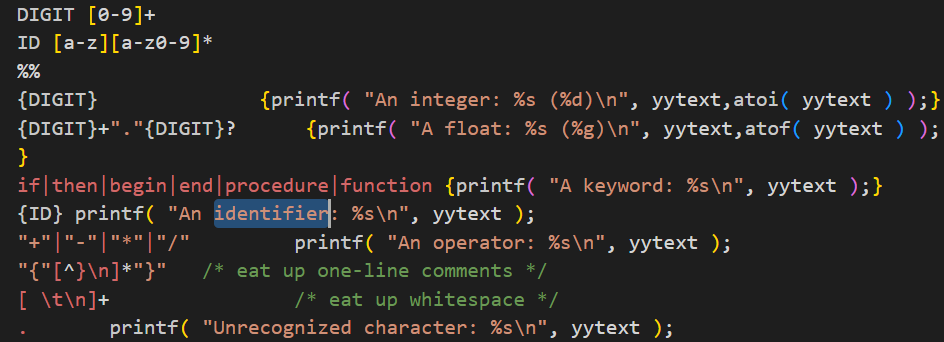
* + - 1. 匹配规则

分析结束时输出字符数与行数，程序输出部分如下图所示：



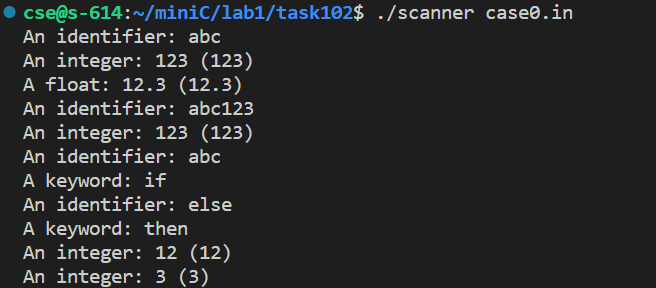
* + - 1. 程序分析输出

task102：实现简单的toy语言的词法分析，实现对整数，小数，部分关键字，标识符，操作符号的识别，输出不能识别的符号。匹配规则如下图所示：



* + - 1. 匹配规则

测试结果如下图所示：



* + - 1. 测试结果

task103：了解flex规则在匹配时，如果多条规则都匹配成功， yylex 会选择匹配长度最长的那条规则，如果有匹配长度相等的规则，则选择排在最前面的规则。

预测结果：132311132测试结果如下图所示：



* + - 1. 测试结果

task104：利用flex工具生成PL语言的词法分析器

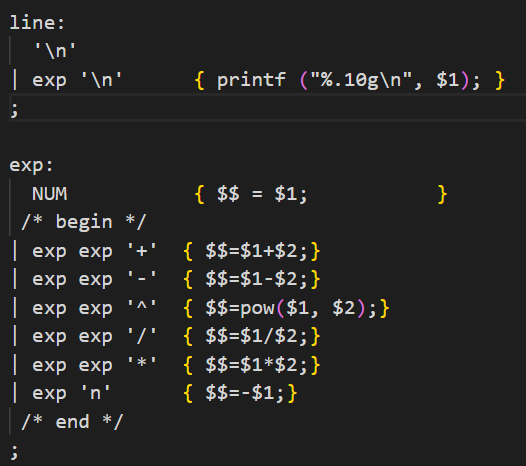
实现方法：编写对应的匹配规则，匹配规则部分如下图所示：



* + - 1. 匹配规则（部分）

task105：利用YACC/Bison构建一个逆波兰符号计算器。

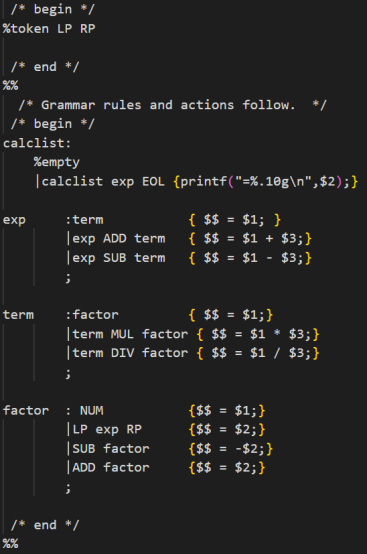
实现方法：根据逆波兰式计算规则编写bison BNF范式。语法规则如下图所示：



* + - 1. 语法规则

task106：继续使用Bison，完成中缀式计算器的语法规则设计。

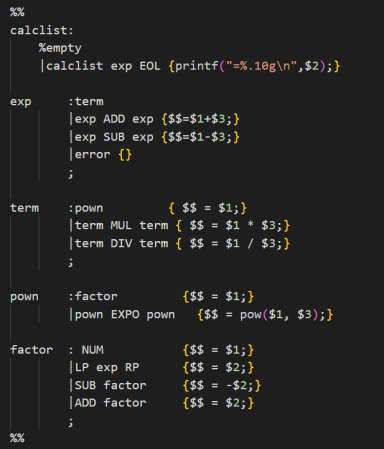
语法规则如下图所示：



* + - 1. 语法规则

task107:联合Bison+flex 的模式，完成中缀表达式计算器。

语法规则如下图所示：



* + - 1. 语法规则

（3）小结

通过实验一，我学习到了使用Flex和正则表达式进行简单的词法分析，使用Bison进行简单的语法分析，并将两者结合起来实现简单的计算器功能。

不足之处：词法分析部分存在较多的代码冗余。

## MiniC词法分析

（1）实验内容

Flex与Bison工具后，我们将利用两种工具，逐步完成对Mini-C语言的结构分析。

（2）实验过程

task201：识别参考资料对应的单词。

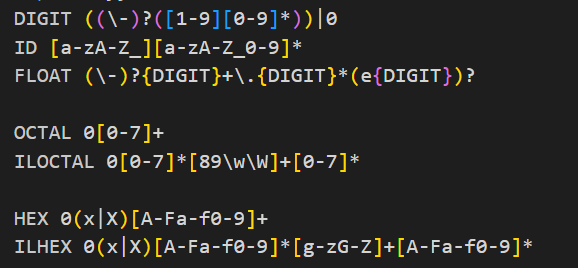
实现方法：词法匹配规则如下图所示：



* + - 1. 匹配规则

task202：增加了对保留关键字的测试；能够识别简单浮点数，例如1.2，1.05e5，八进制数、十六进制数等。同时，能做到一定程度的容错功能：识别非法八进制如08、非法十六进制数字如0xGF2。

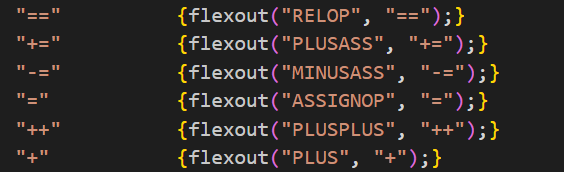
实现方法：词法匹配规则如下图所示：



* + - 1. 匹配规则

task203：掌握规则顺序对词法分析的影响，修改词法规则，完成以下四个运算符的识别：++，--，+=，-=。

实现方法：词法匹配规则如下图所示：



* + - 1. 匹配规则

（3）小结

本次实验实现了对minic中的关键字、符号、整数、小数、八进制和十六进制数的分析，以及相应的错误检测机制。

不足之处：匹配规则存在冗余，鲁棒性不强，可能存在bug。

## MiniC语法分析及语法树生成

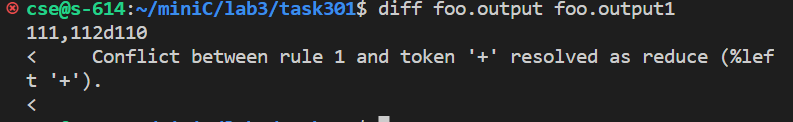
（1）实验内容

学习词法、语法分析识别程序的编写方法，使用flex，完成Mini-C语言词法分析。采用语法制导的方法，完成语法树的输出。

（2）实验过程

task301：了解Bison系统自动解决冲突的能力以及手动解决移进规约冲突与规约规约冲突，并动手消除状态中的移进规约冲突。

Bison自动解决冲突：



* + - 1. Bison解决冲突

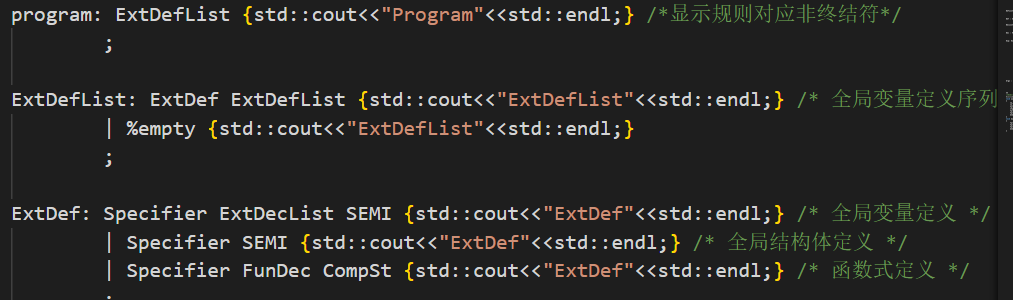
消除状态中的移进规约冲突：



* + - 1. 消除冲突

task302：输入Mini-C的源文件，找到最左规约序列，判断其语法是否正确，输出错误信息，并按最左规约的顺序，打印出规约过程中使用的非终结符名称。

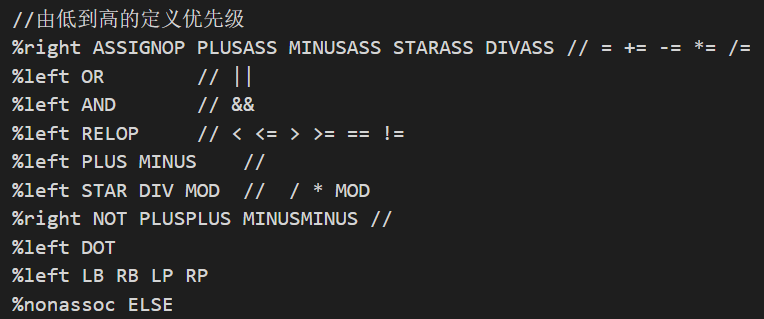
实现方法：根据Mini-C语法，编写语法规则，输出非终结符名称。语法规则部分如下图：



* + - 1. 语法规则

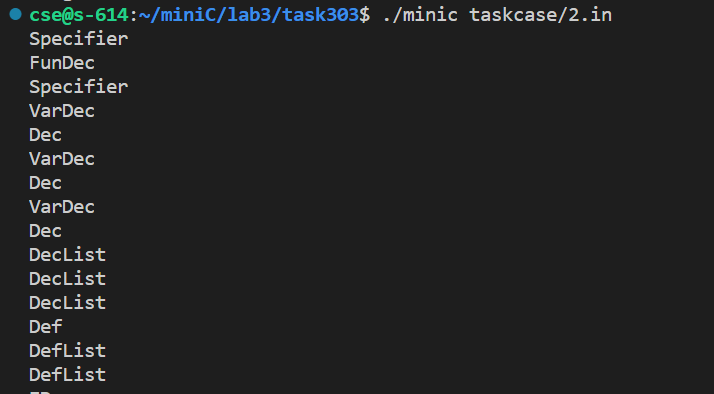
task303：利用Bison -v参数等解决编译过程中的移进规约冲突。

实现方法：添加优先级规则，如下图所示：



* + - 1. 优先级规则

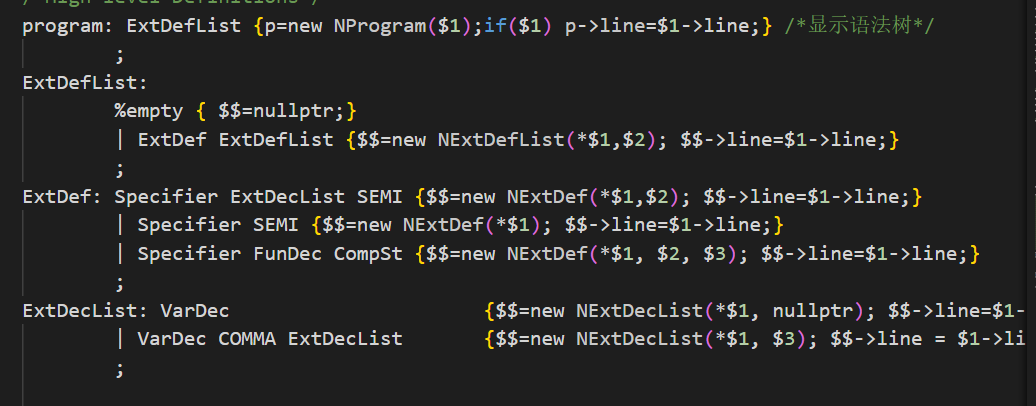
测试结果如下图所示：



* + - 1. 测试结果

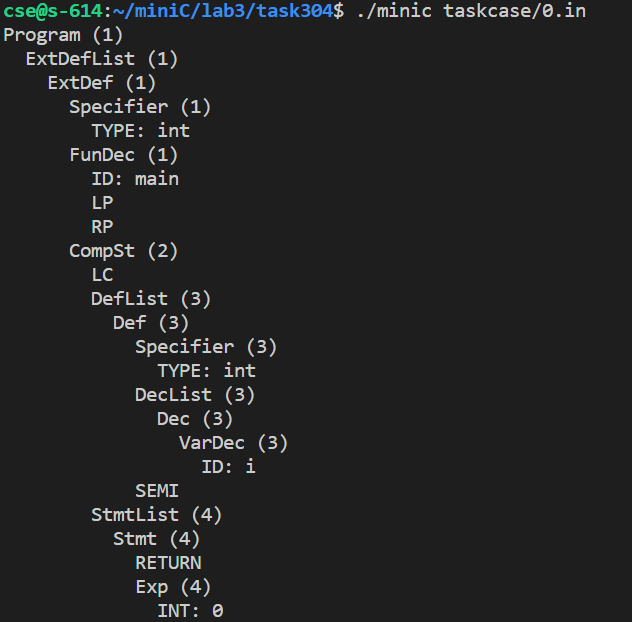
task304：对Mini-C的简单样例进行分析，按实验指导书要求，输出其对应的语法树。

实现方法：利用语法分析构建语法树，语法规则部分如下图所示：



* + - 1. 语法规则

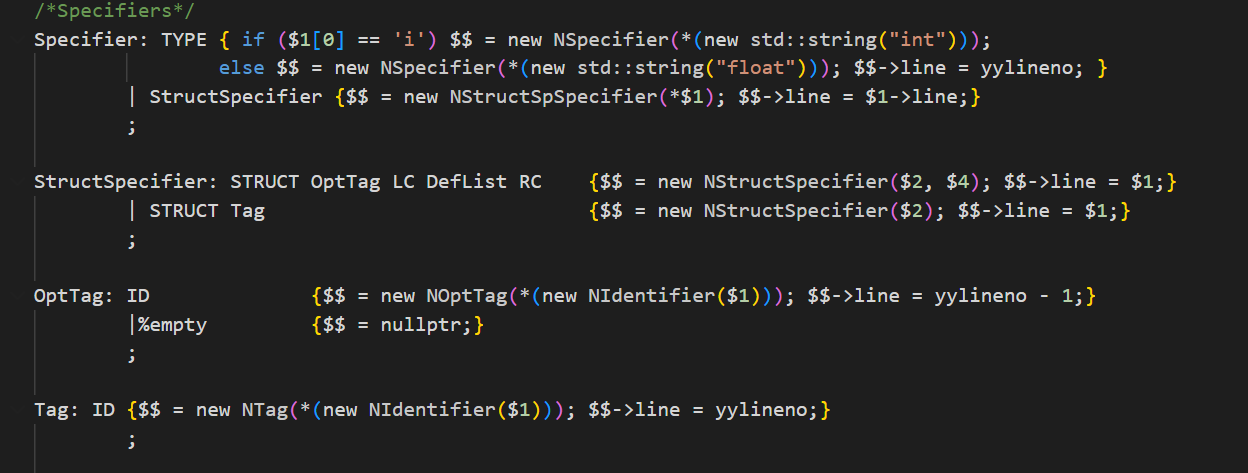
测试结果如下图所示：



* + - 1. 测试结果

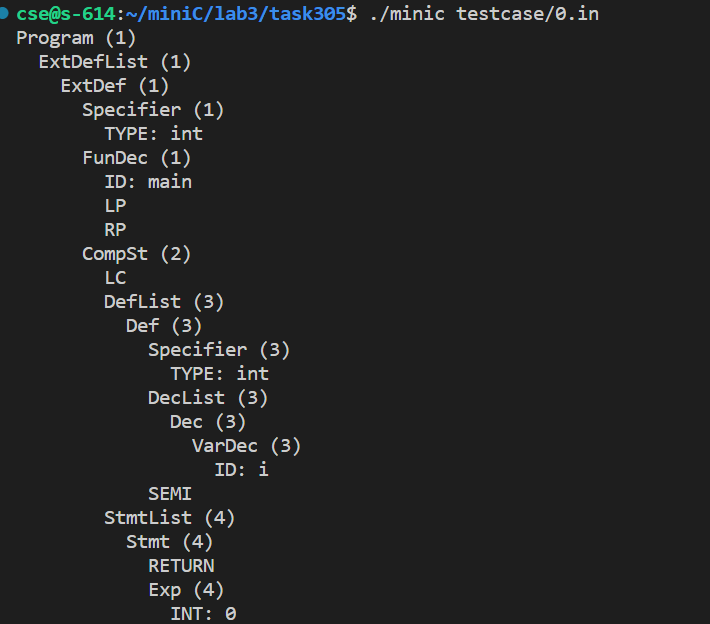
task305：完善语法树的输出。

实现方法：完善语法树的语法规则，并编写对应语法规则。部分语法规则如下图所示：



* + - 1. 语法规则

测试结果如下图所示：



* + - 1. 测试结果

（3）小结

本次实验中，完成了对Mini-C的语法树构建，能够输出完整的语法树，并对其行号的输出以及错误提示。

遇到的困难：在语法树的构建过程中，有几个特殊的类需要单独构建与判断。

不足之处：语法树中有两处的行号问题。

## MiniC语义分析及中间代码生成

（1）实验内容

编写程序，对输入的Mini-C源文件进行分析，输出对应的IR程序段。

（2）实验过程

task401: 手工编写LLVM IR程序实现多个输出与分支等功能。

实验一：输出HUSTCSE\n。

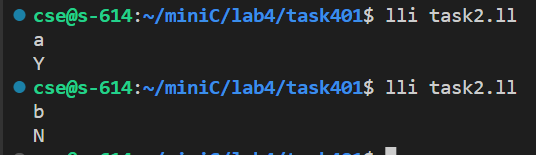
实验二：输入单个字符，如果是a则输出Y，否则输出N。

实现方法：一、编写代码，先将字符对应ASCII码存入内存中，后将其取出放入参数内。最后call putchar函数输出。代码如下图所示：

二、编写代码，获取输入字符，将其与a字符比较，若结果相等则跳转到输出Y，否则跳转到输出N，测试结果如下图所示：



* + - 1. 实验一测试结果



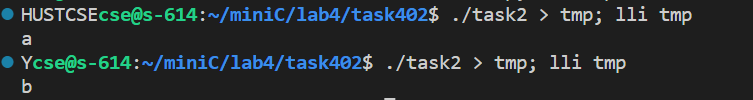
* + - 1. 实验二测试结果

task402：使用C++代码与LLVM提供的接口函数，动态生成LLVM IR。

实现方法：实验一，获取字母对应ASCII常量，存入内存，加载常数，存入参数数组中，调用函数输出对应字符。实验二，创建基本块进行跳转，部分代码测试结果如下图所示：



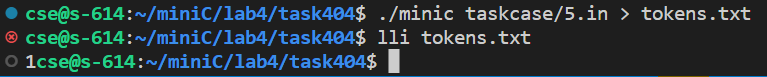
* + - 1. 实验一测试结果



* + - 1. 实验二测试结果

task404: 编写编译器前端，分阶段（子任务）将minic源代码翻译到中间代码；并分析其中的语义错误。

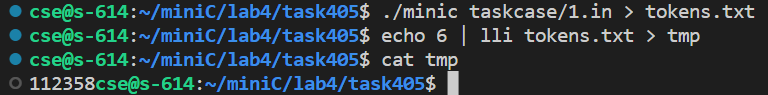
实现方法：根据IR规则与语义分析，生成相应的代码与错误解析。测试结果如下图所示：



* + - 1. 测试结果

task405：编写编译器前端，分阶段（子任务）将minic源代码翻译到中间代码；并分析其中的语义错误。

实现方法：根据IR规则与语义分析，生成相应的代码与错误解析，测试结果如下图所示：



* + - 1. 测试结果

（3）小结

本实验实现了对Mini-C源文件的LLVM IR实现，完成了动态编译的目的。

遇到的困难：变量表的建立，维护和查找需要定义编写相关内容

不足之处：部分功能尚未实现。

## MiniC代码优化及目标代码生成

（1）实验内容

简单的使用LLVM代码优化框架及命令行，完成相关任务。

（2）实验过程

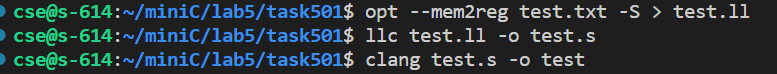
关卡相关的主要实现过程。

task501：调用LLVM中的优化函数，观察IR代码发生的变化。

实验结果：由于编写的问题，代码无变化。

task502：将文件test.txt中的IR进行优化，并最终能够产生可执行的二进制程序test。

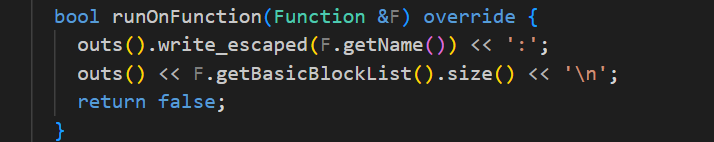
测试结果：



* + - 1. 测试结果

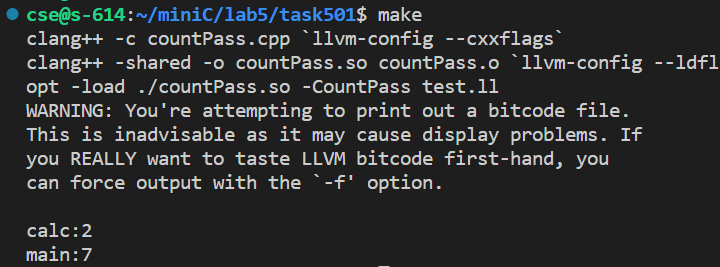
task503：我们还可以构造自己的PASS实现输出函数名字及对应函数的基本块数量。

实现方法：重写runOnFunction函数，如下图所示：



* + - 1. 修改代码

测试结果如下图所示：



* + - 1. 测试结果

（3）小结

通过本实验对LLVM本身的优化函数的调用，自己编写优化函数以及可执行文件的生成等。

不足之处：由于代码问题LLVM优化效果不明显。

# 实验心得

在本次实验中，我学习到了对于Mini-C语言编译器的基本实现，能够对基本的函数调用，运算，语句等具体的实现，复习了编译原理中所学的一些知识。

在实验一中我学习到了对Flex,Bison的基本使用，能够对文本中的词法进行简单的分析以及一些简单的语法分析。

在实验二中我学习到了更为复杂的词法分析，能够对八进制数，十六进制数以及小数等进行具体的识别。

在实验三中我学习到了Bison冲突的解决方法以及语法树的构建。

在实验四中我学习到了LLVM IR的编写以及如何依据语法树来动态生成LLVM IR文件。

在实验五中我学习到了LLVM中的一些代码优化函数，以及编写了自己的一些函数。

# 实验内容和过程的建议

本实验内容循序渐进，由简单到复杂来实现了一个编译器的构建。

优点：使用云平台，省略了本地配置环境的时间，同时使用linux无桌面端，流畅不卡顿。

使用云平台自测试，有一个统一的评分标准，方便学生去发现错误。

前三个实验难度适中，且说明较详细。

缺点：第三个实验4，5。第四个实验3，4，5难度较大。且说明内容较少，需要学生自己补充的内容过多，示例较少，做起来力不从心，只能借鉴网络上的部分参考，才能实现相关内容。建议这部分加入一些提示，示例部分，同时减少需要自己补充的部分。

# 参考文献

[1] 许畅 等编著. 《编译原理实践与指导教程》.机械工业出版社

[2] Jobn Levine著，陆军 译. 《Flex与Bison》.东南大学出版社