**序言**

实验目标： 完成SPL语言编译器，输入为符合SPL语言规范的文本，输出汇编代码和可执行文件。

实验环境：

1. 操作系统：Ubuntu
2. 构建工具：CMake
3. 编译器：GCC
4. 其他工具：Bison，Flex，LLVM

实验设计：

组员分工：

1. **词法分析**
2. 实验原理

词法分析的目标是将源代码根据正则表达式转换成token序列。SPL的token包括标识符、常数值、运算符、分界符、系统函数、系统过程、系统类型、关键字等。除标识符和常数值外，其他token都是某个特定字符串。标识符可以使用（Lex）正则表达式表示：[a-zA-z][\_a-zA-Z0-9]\*，整数值表示为：[0-9]+，实数值可以表示为：[0-9]+(“.”) [0-9]+（在Lex中，.匹配任意单个字符，需要转义处理），字符值可以为表示为\‘.\’（单引号也需要使用转义字符表示）。关键字、系统函数等字符串虽然满足标识符对应的正则表达式，但是不会视为标识符。

Lex源代码主要是一个由正则表达式和相应代码片段组成的表。Lex会将表转换为对应程序，以读取输入流并且将输入划分为匹配正则表达式的字符串，并且识别出对应字符串后会执行相应程序片段。Lex是通过生成确定性有限自动机来进行表达式的识别，并且代码片段会按照输入流中出现的字符串对应表达式顺序执行。

1. **语法分析**
2. 实验原理

语法分析的目标是将token序列转换为parse树。Parse树通过

Yacc源代码主要是一个由上下文无关文法和相应代码片段组成的表，当对应结构被识别出来时会执行相应代码。Bison通过将LALR（1）文法转换为程序来解析语法。

1. **语义分析**

语义分析

1. **优化考虑**

优化考虑

1. **代码生成**

代码生成

1. **测试案例**

测试案例