

编译原理 实验 报告

实验名称：实验四 代码生成_____

姓名：方澳阳_____

学号：180110115_____

学院：计算机科学与技术_____

专业：计算机类_____

一、 实验目的与方法

- (1) 加深对编译器总体结构的理解与掌握
- (2) 加深对汇编指令的理解与掌握
- (3) 对指令选择，寄存器分配有较深的理解实现语言：

使用语言: C++

IDE: Clion2020.2

平台: windows10、linux

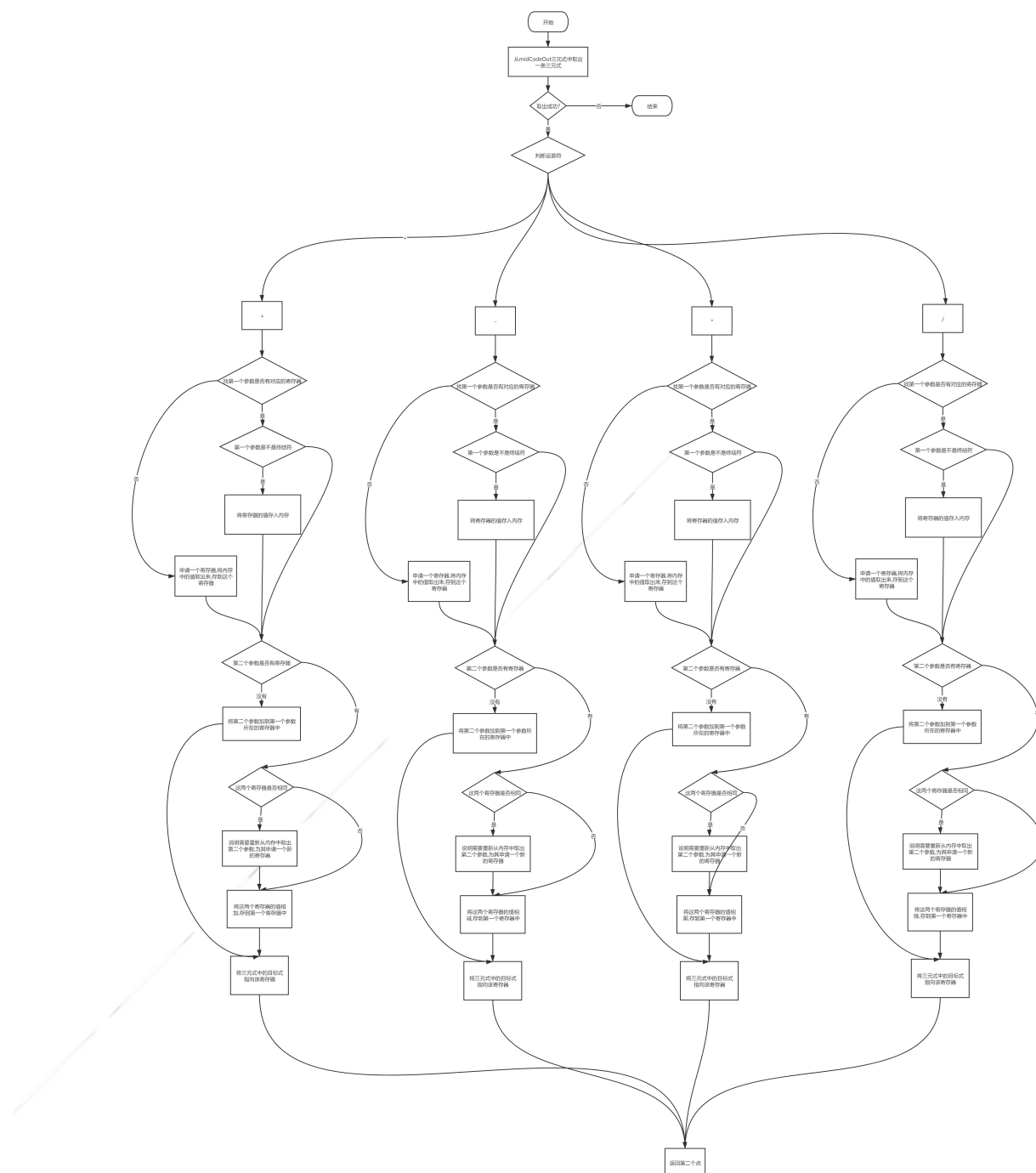
二、 实验总体流程与函数功能描述

本次实验主要在上一实验的基础上添加了以下两个函数：

```
/**
 * 以时间片轮转的方式获取一个寄存器
 * @return
 */
string Grammar_Analyzer::getReg() {
    int ans = regs.back();
    regs.pop_back();
    regs.push_front(ans);
    return "R" + to_string(ans);
}
/**
 * 将中间代码转换为汇编代码
 */
void Grammar_Analyzer::toAsm()
```

该函数的内部实现为：首先判断运算符号，从而根据运算符号生成相应的汇编代码。

主要过程如下图所示：



三、 实验内容

1. 将中间代码所给的地址码生成目标代码（汇编指令）
2. 生成汇编指令，将三地址码序列生成代码序列表（参考 P407 页）
3. 减少程序与指令的开销，进行部分优化（可选）
4. 较低完成要求：将赋值语句 $d:=(a-b)+(a-c)+(a-c)$ 翻译为三地址码序列，并将其转化为目标汇编指令

四、 实验结果与分析

输入：

```
temp = (400-2)*10#  
code = temp-12/3#  
bing = (code/10)*temp#  
ans = bing
```

实验三输出的三地址码：

```
S = 400 - 2  
temp = S * 10  
A = 12 / 3  
code = temp - A  
A = code / 10  
bing = A * temp  
S = bing + 2  
ans = S / 4
```

输出的汇编代码

```
Mov 400, R0  
Sub 2, R0  
Mul 10, R0  
Mov 12, R1  
Div 3, R1  
Mov R0, temp  
Sub R1, R0  
Mov R0, code  
Div 10, R0  
Mov temp, R2  
Mul R2, R0  
Mov R0, bing  
Add 2, R0  
Div 4, R0  
Mov R0, ans
```

寄存器与变量的映射关系为：

使用 `map<string,string> valueToReg` 该数据结构，建立一个寄存器与变量的映射。在 `toAsm` 函数中进行判断，从而实现减少寄存器的使用，加快程序运行。

分析

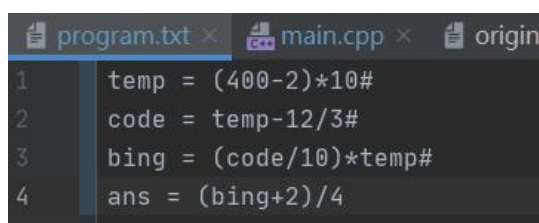
由于上个实验删减内容到只剩算数运算后，该实验中将上个实验产生的三元式转换为汇编指令的难度也大幅下降。本次实验设置了 30 个寄存器，以模拟比较真实的场景。可以从输出的汇编代码看到，以上的算术运算中用到了 3 个寄存器。

在本实验中减少访存次数的方法为多次利用寄存器，主要思想为记录一个值（可能是中间变量或者是实际的变量）对应的寄存器。在产生汇编代码时，将三地址码中的变量替换为实际的寄存器。如果寄存器有冲突，则需要访存，将内存中的值读入寄存器。

五、 实现的拓展功能

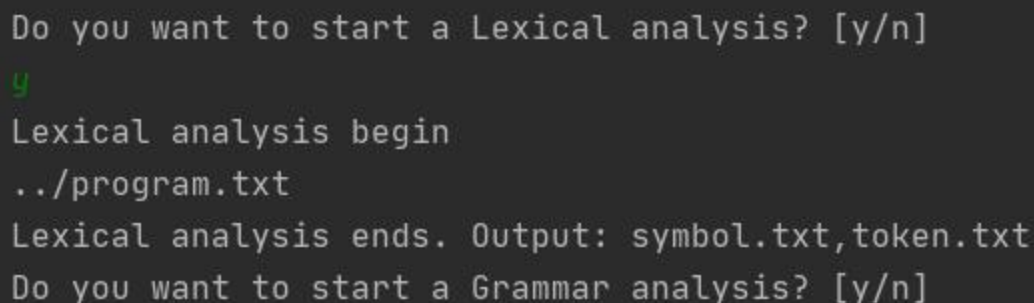
本课程全部四个实验（词法分析，语法分析，语义分析和中间代码生成，目标代码生成）整合为一个连贯的编译器程序，包括必要的功能选择窗口与各阶段的输出，并附有简单的用户说明文档。

输入文件为 program.txt



```
program.txt ×  main.cpp ×  origin
1  temp = (400-2)*10#
2  code = temp-12/3#
3  bing = (code/10)*temp#
4  ans = (bing+2)/4
```

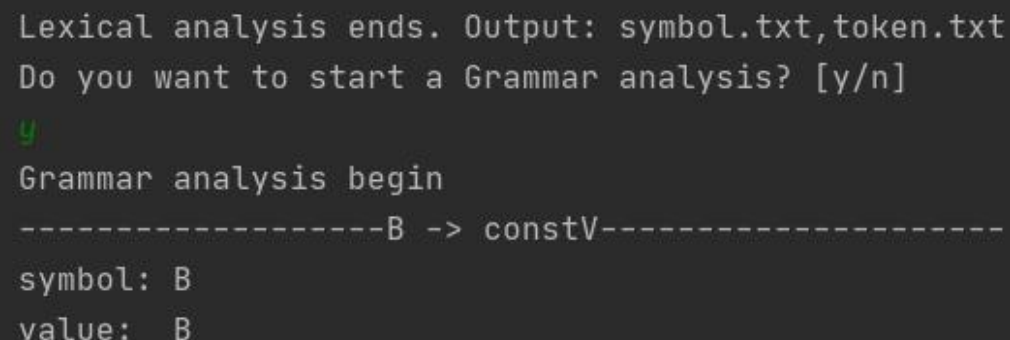
用户首先可以选择词法分析



```
Do you want to start a Lexical analysis? [y/n]
y
Lexical analysis begin
../program.txt
Lexical analysis ends. Output: symbol.txt,token.txt
Do you want to start a Grammar analysis? [y/n]
```

分析结束后产生 symbol.txt 以及 token.txt

继续选择进行语法分析,控制台输出一些 log



```
Lexical analysis ends. Output: symbol.txt,token.txt
Do you want to start a Grammar analysis? [y/n]
y
Grammar analysis begin
-----B -> constV-----
symbol: B
value:  B
```

结束之后生成了 producer.txt, 以及 midCode.txt. 继续选择生成目标代码，最后输出 asmCode.txt 目标代码文件。

```
Grammar analysis, Semantic analysis and intermediate code generation ends. Output: producer.txt, midCode.txt, symbolTable.txt
Do you want to generate target code? [y/n]
y
Success! Output: asmCode.txt
```