**课 程 设 计 报 告**

设计题目：一个简单文法的编译器的设计与实现

班 级：计算机1808

组长学号：20182853

组长姓名：王昊天

主 笔：王昊天

设计时间：2020年6月

设计分工

组长学号及姓名：20182853王昊天

分工：结构体，函数数组以及复合语句的文法设计，声明语句，函数声明，函数调用，整个程序框架的递归下降子程序的编写，四元式设计，符号表设计，活动记录设计及实现，同名变量的区分，重定义的识别；

组员1学号及姓名：20184662周滋昊

分工：目标代码指令集合定义，活跃信息生成，目标指令生成

组员2学号及姓名：20184422邱建福

分工：常数处理机，字符常量处理机，字符串常量，符号表设计，符号表底层设计

组员3学号及姓名：20184319廖真

分工：Bool运算文法的设计，翻译文法设计，常值表达式节省，公共子表达式节省，删除无用赋值

组员4学号及姓名：20184539刘荣江

分工：If，While，for，算数表达式，bool运算的设计，递归下降子程序

**摘 要**

本组设计的编译器是面向类c语言的编译器。使用c语言的语法和关键字，可以识别while语句，if语句和for语句等循环语句和条件语句，可以进行函数声明，函数调用，全局变量的声明，结构体的定义，本组设计的编译器分为词法分析，语法分析，语义分析，优化，和最终代码生成等部分，利用词法分析生成token序列，在语法分析的过程中，加入了语义动作对程序按层处理，更新标识符的token值，以区分同名变量，作用域和重定义，为避免语义分析较为繁琐复杂，在另一份语法分析中添加语义动作完成语义分析，之后对程序进行优化，语义分析形成四元式后，将四元式存入txt文件中中，然后在优化部分使用dag图优化法，将四元式转化为dag图然后进行优化，再次存入到文件中进行最终代码生成，最终代码生成为8086/8088汇编语言。

**关键词**：编译原理，前端，后端

目录

[1 概述 5](#_Toc45843006)

[2 课程设计任务及要求 5](#_Toc45843007)

[2.1 设计任务 5](#_Toc45843008)

[2.2 设计要求 5](#_Toc45843009)

[3 编译系统总体设计 6](#_Toc45843010)

[3.1 编译器结构设计 6](#_Toc45843011)

[3.2 文法设计 6](#_Toc45843012)

[3.3 符号表设计 8](#_Toc45843013)

[3.4 重定义、同名变量和作用域区分 9](#_Toc45843014)

[4 编译器前端设计 11](#_Toc45843015)

[4.1 扫描器和语法分析 11](#_Toc45843016)

[4.2 翻译文法及语义分析 11](#_Toc45843017)

[5 编译器后端设计 13](#_Toc45843018)

[5.1 优化 13](#_Toc45843019)

[5.2 目标代码生成 15](#_Toc45843020)

[6 结论 17](#_Toc45843021)

[6.1 王昊天 17](#_Toc45843022)

[6.2 邱建福 17](#_Toc45843023)

[6.3 刘荣江 17](#_Toc45843024)

[6.4 廖真 17](#_Toc45843025)

[6.5 周滋昊 18](#_Toc45843026)

[7 参考文献 18](#_Toc45843027)

[8 收获、体会和建议 18](#_Toc45843028)

[8.1 王昊天 18](#_Toc45843029)

[8.2 邱建福 19](#_Toc45843030)

[8.3 刘荣江 20](#_Toc45843031)

[8.4 廖真 21](#_Toc45843032)

[8.5 周滋昊 22](#_Toc45843033)

# 概述

本次课设所实现的类c编译器包括了结构体的定义，函数的声明调用，一维数组的实现，同名变量，重定义以及作用域的区分。分别完成了词法分析，语法分析，语义分析，优化，目标代码生成各个模块，但是遗憾的是实验过程中的突发情况，导致了课设没有顺利完成对接。目前只能各个模块分别运行。

# 课程设计任务及要求

## 设计任务

一个简单类c语言的编译器的前端和后端的设计。最终能够生成8086汇编语言，并且能正常运行。

## 设计要求

1、在深入理解编译原理基本原理的基础上，对于选定的题目，以小组为单位，先确定设计方案；

2、设计系统的数据结构和程序结构，设计每个模块的处理流程。要求设计合理；

3、编程序实现系统，要求实现可视化的运行界面，界面应清楚地反映出系统的运行结果；

4、确定测试方案，选择测试用例，对系统进行测试；

5、运行系统并要通过验收，讲解运行结果，说明系统的特色和创新之处，并回答指导教师的提问；

6、提交课程设计报告。

# 编译系统总体设计

## 编译器结构设计

包含词法分析，语法分析，语义分析，优化，目标代码生成各个模块。

## 文法设计

完全使用c语言的语法：

＜加法运算符＞ ::= +｜-

＜乘法运算符＞ ::= \*｜/

＜关系运算符＞ ::= ｜>=｜!=｜==

＜字母＞ ::= ＿｜a｜．．．｜z｜A｜．．．｜Z

＜数字＞ ::= ０｜＜非零数字＞

＜非零数字＞ ::= １｜．．．｜９

＜字符＞ ::= '＜加法运算符＞'｜'＜乘法运算符＞'｜'＜字母＞'｜'＜数字＞'

＜字符串＞ ::= "｛十进制编码为32,33,35-126的ASCII字符｝"

＜程序＞ ::= {<结构体>|＜常量说明＞|＜变量说明＞|＜有返回值函数定义＞|＜无返回值函数定义＞}＜主函数＞

<结构体> ::= typedef struct <标识符>‘{’ {<变量定义>} '}' <标识符>{,<标识符>} ‘;’| struct <标识符>‘{’ {<变量定义>} '}' <标识符>{,<标识符>} ‘;’

＜常量说明＞ ::= const＜常量定义＞;{ const＜常量定义＞;}

＜常量定义＞ ::= int＜标识符＞＝＜整数＞{,＜标识符＞＝＜整数＞}

| float＜标识符＞＝＜实数＞{,＜标识符＞＝＜实数＞}

| char＜标识符＞＝＜字符＞{,＜标识符＞＝＜字符＞}

＜无符号整数＞ ::= ＜非零数字＞｛＜数字＞｝

＜整数＞ ::= ［＋｜－］＜无符号整数＞｜０

＜小数部分＞ ::= ＜数字＞｛＜数字＞｝｜＜空＞

＜实数＞ ::= ［＋｜－］＜整数＞[.＜小数部分＞]

＜标识符＞ ::= ＜字母＞｛＜字母＞｜＜数字＞｝

＜声明头部＞ ::= <类型标识符>＜标识符＞

＜变量说明＞ ::= ＜变量定义＞;{＜变量定义＞;}

＜变量定义＞ ::= ＜类型标识符＞(＜标识符＞|＜标识符＞‘[’＜无符号整数＞‘]’){,(＜标识符＞|＜标识符＞‘[’＜无符号整数＞‘]’ )}

\*\*改（支持多维数组，好像有点难，不改也行）\*\*＜变量定义＞ ::= ＜类型标识符＞(＜标识符＞|＜标识符＞{‘[’＜无符号整数＞‘]’}){,(＜标识符＞|＜标识符＞{‘[’＜无符号整数＞‘]’ })}

＜可枚举常量＞ ::= ＜整数＞|＜字符＞

＜类型标识符＞ ::= int | float | char

＜有返回值函数定义＞ ::= ＜声明头部＞\*\*riseLayer()\*\*‘(’＜参数＞‘)’ ‘{’＜复合语句＞‘}’\*\*dropLayer()\*\*

＜无返回值函数定义＞ ::= void＜标识符＞\*\*riseLayer()\*\*‘(’＜参数＞‘)’‘{’＜复合语句＞‘}’\*\*dropLayer()\*\*

＜复合语句＞ ::= ［＜常量说明＞］［＜变量说明＞］＜语句列＞

＜参数＞ ::= ＜参数表＞

＜参数表＞ ::= ＜类型标识符＞＜标识符＞{,＜类型标识符＞＜标识符＞}| ＜空＞

＜主函数＞ ::= void main‘(’‘)’‘{’＜复合语句＞‘}’

＜表达式＞ ::= ［＋｜－］＜项＞{＜加法运算符＞＜项＞}

＜项＞ ::= ＜因子＞{＜乘法运算符＞＜因子＞}

＜因子＞ ::= ＜标识符＞｜＜标识符＞‘[’＜表达式＞‘]’|‘(’＜表达式＞‘)’｜＜整数＞|＜实数＞|＜字符＞｜＜有返回值函数调用语句＞

＜语句＞ ::= ＜条件语句＞｜＜循环语句＞| ‘{’＜语句列＞‘}’｜＜有返回值函数调用语句＞;

|＜无返回值函数调用语句＞;｜＜赋值语句＞;｜＜读语句＞;｜＜写语句＞;｜＜空＞;|＜情况语句＞｜＜返回语句＞;

＜赋值语句＞ ::= ＜标识符＞＝＜表达式＞|＜标识符＞‘[’＜表达式＞‘]’=＜表达式＞

＜条件语句＞ ::= if \*\*riseLayer()\*\* ‘(’＜条件＞‘)’＜语句＞ \*\*dropLayer()\*\*

＜条件＞ ::= ＜表达式＞＜关系运算符＞＜表达式＞｜＜表达式＞ //表达式为0条件为假，否则为真

\*\*改\*\*＜循环语句＞ ::= while \*\*riseLayer()\*\*‘(’＜条件＞‘)’＜复合语句＞ \*\*dropLayer()\*\*|for \*\*riseLayer()\*\* '('<声明语句>|<赋值语句>‘;’<条件语句>‘;'<赋值语句>’)‘<复合语句> \*\*dropLayer()\*\*

＜情况语句＞ ::= switch \*\*riseLayer()\*\* ‘(’＜表达式＞‘)’ ‘{’＜情况表＞＜缺省＞ ‘}’ \*\*dropLayer()\*\*

＜情况表＞ ::= ＜情况子语句＞{＜情况子语句＞}

＜情况子语句＞ ::= case＜可枚举常量＞：＜语句＞

＜缺省＞ ::= default : ＜语句＞|＜空＞

＜有返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’

＜无返回值函数调用语句＞ ::= ＜标识符＞‘(’＜值参数表＞‘)’

＜值参数表＞ ::= ＜表达式＞{,＜表达式＞}｜＜空＞

＜语句列＞ ::= ｛＜语句＞｝

＜读语句＞ ::= scanf ‘(’＜标识符＞{,＜标识符＞}‘)’

＜写语句＞ ::= printf ‘(’ ＜字符串＞,＜表达式＞ ‘)’| printf ‘(’＜字符串＞ ‘)’| printf ‘(’＜表达式＞‘)’

＜返回语句＞ ::= return[‘(’＜表达式＞‘)’]

## 符号表设计

使用课堂时所讲符号表系统，由于所学的符号表已经比较完善，无需更改。

符号表整体设计采用结构体数组形式，便于查找，每个数组单元包括该标识符名字、类型、种类，地址等信息。同时还有AD,FD等用于查找标识符对应表格的索引（数组表和函数表同样采用结构体数组的设计，[AD]即数组标识符对应的数组表单元，函数表同理）。数组表包括数组上下标、数组单元类型以及该数组总长度等信息，函数表包括参数表以及参数个数等信息。

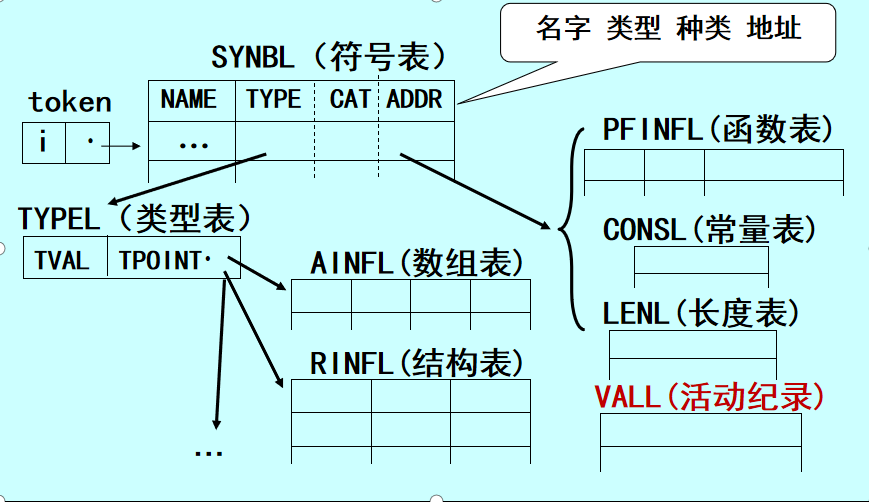
因为是C语言，所以包括了void、int、float、char等基本类型。

标识符的插入相对简单，int addIdentifier(int token,int type,int kind)。加上函数声明来说可能不那么抽象，token为标识符位序，type和kind为标识符对应类型和种类，如果没有初始化则地址指向-1,否则指向对应地址。

int addFunction(int token, int type)，函数名填写较为简单，因为类型固定为f，地址为-1。每次识别到函数定义则全局变量fd++，之后将fd赋给FD，即将函数索引指向该函数在函数表中的存储单元。

之后是参数表的填写，int addParameter(int token,int type,int ftoken)。ftoken为该形参对应的函数名称在标识符中位序，通过该值得到在符号表中中位置，进而得到FD，找到对应的形参表进行填写即可，每次填写函数表形参个数++。

之后是数组的添加，int addArry(int token,int type,int kind,int length)，length为数组元素个数，type为数组元素类型（因为数组类型固定为a），通过ad索引到对应数组表进行填写，数组下标默认为0，上标length-1，类型长度int4,float8,char1，长度则通过length和type对应长度做积得到。



## 重定义、同名变量和作用域区分

设计一个类似活动记录的结构Recording，里面存着变量的token值，全局为第零层，每当遇到函数（包括main）或者循环和条件语句，则进入下一层，当进入新的一层时，新建一个Recording压入栈中作为该层的记录，当函数（包括main）或者循环和条件语句结束时，则将当前Recording指向全局display，每当定义变量时，查看当前Recording有没有同名变量，若有，则存在重定义，若没有，判断当前token值对应的符号表中变量是否定义，若已定义则为同名变量，但是作用域不同，申请新的token值，利用相关信息进行定义，将该token值写入当前Recording，否则直接用相关信息进行定义，写入当前Recording。而遇到非定义的标识符，则根据该token对应的名称，在当前Recording中查找，若存在相同名称，则更新token为该层中此名称对应的token，托不存在，则在其全局display中查找，依次类推，直到查到为止，若第0层还没查到，则该变量未定义。

# 编译器前端设计

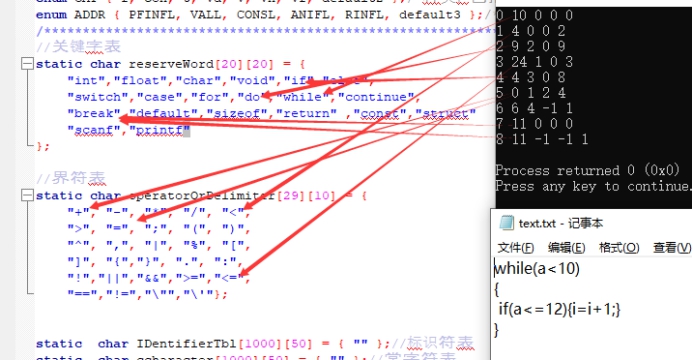
## 扫描器和语法分析

词法分析程序主要包括：接收字符串形式的源程序，按照源程序输入的次序依次扫描源程序，在扫描的同时根据语言的词法规则识别出具有独立意义的单词，并产生与源程序等价的单词序列（TOKEN）。

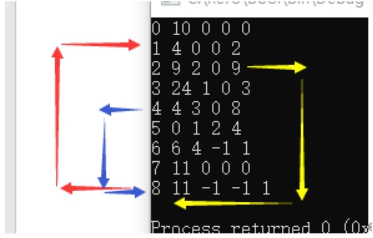
根据单词类型，词法分析主要包括六个部分，标识符IT，关键字KT，界符PT，字符串ST，常字符cT，常数CT。关键字和界符的token值由查表得到，剩下则动态生成表格，token为所在表格位序。程序核心即处理“字符串”得到目标单次序列的算法。关键字、标识符、符号串和常字符首位都是字母，故放在一起处理。具体方法为：如果识别到一个常字符，先判断前面是否有单引号或者双引号，如果有，分别识别为常字符和字符串，如果对应表格已经存在该单词则直接返回位序，否则插入后返回位序。如果没有，则读到下一个空格。之后将该单词在关键字表进行查找，如果找到返回所在位置，如果不在就是标识符，如果已经字表中返回位序，不在则插入后返回位序。遇到数字判断为常数，读到空格停止，处理同常字符。界符如果不是“=”等可作为双目运算符开始的单目运算符直接查表返回位序，否则继续判断下一单词，如果依旧是运算符则连接后进行查找，否则返回对应单目运算符位序。

## 翻译文法及语义分析

由于队友已经实现了相关的语义分析，所以翻译文法的语义动作只需在相应的位置添加即可，整体难度不大。拿实现的while循环翻译文法为例，队友实现的是while循环里可任意嵌套if，while ,for ,且这些嵌套的模块也可继续嵌套，翻译文法是只需按模块添加语义动作即可。首先四元式是一个4个int 类型的结构体，对于while循环而言，由于需要回填，故需要记录四元式的次序，所以设置一个四元式类型的数组。首先生成(while)这个无定义的四元式，然后把下一个四元式是次序记录下来，假设为n，然后生成的是判断语句，将结果存于循环控制四元式的第二个位置，同时记录该控制四元式的次序，然后生成循环体里的赋值四元式，最后将最后一条语句的下个次序回填给控制四元式的最后元素表示为0跳转，将最后一条语句的无条件跳转位置赋值为第一次记录的位置，即条件控制四元式的位置。我们小组讨论后设计出来的四元式结构上4个数字，记录在相应表中的地址，由于在四元式的第一个元素中，既可能是if，while，for等关键字，也可能是=，>,<等界符，如果按部就班的按教材，无法区分开，所以我们设计的将关键字的token值加100，如果大于一百，这在关键字表中读取，反之则在界符表中读取。



为了便于表示，将循环判断四元式的首元素设置为do,if和while结束语句设置为break；



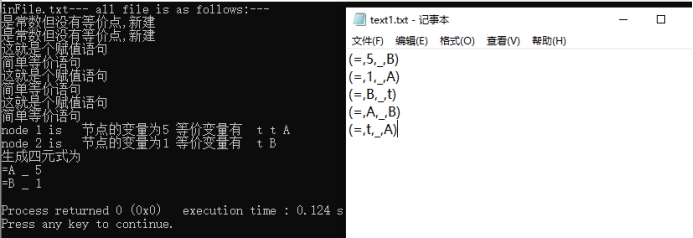
其中红线为while end的无条件跳转，黄线为while的假跳转,蓝线为if的假跳转，真的话顺序执行即可，跳转只存在于首元素为特殊的关键字（如，if，while，while end ,for）的四元式中,故容易区分出来。

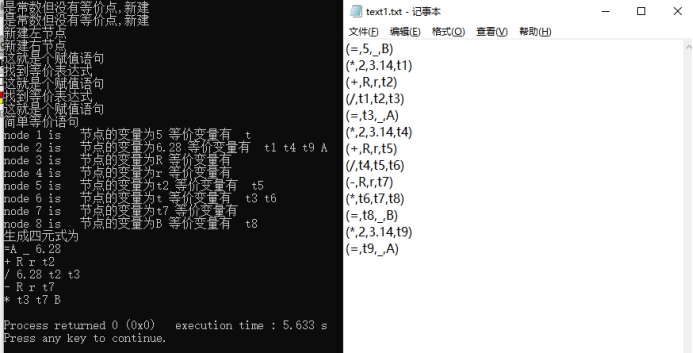
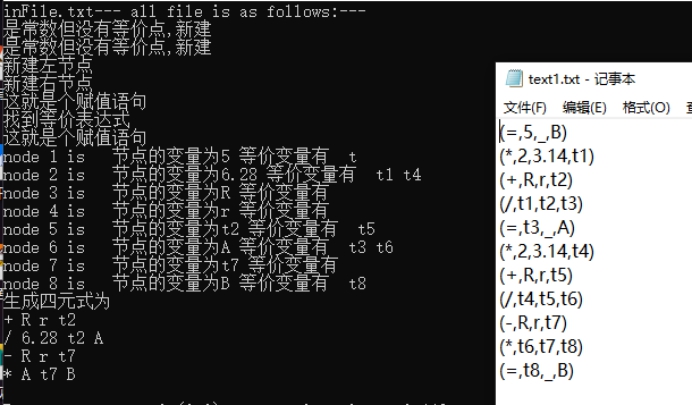
# 编译器后端设计

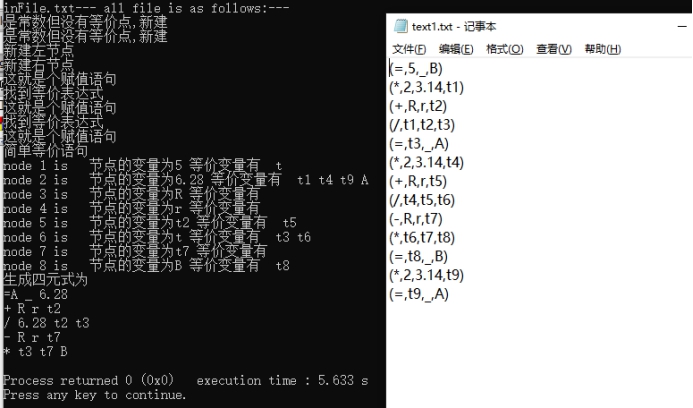
## 优化

采用的由优化方法为基于DAG的局部优化方法，对于单个节点采用邻接表的数据结构，分别由符号位，变量位，两个节点类型的左右孩子指针，一个指向等价点结构的指针，其中，等价点结构为：一个等价变量位，一个指向下一个等价点的指针。同时设置一个节点类型的数组，便于节点的计数，依次处理非常数的赋值四元式，常数表达式，和公共子表达式的四元式，而这些步骤依次可直接或简介处理删除无用赋值，常数表示式节省，和公共表达式节省，其中情况较为复杂的式公共表达式节省，我们顺序遍历节点，同时对运算符，左右孩子的变量位和等价点访问，如果完全相同，则是公共子表达式，将变量附加于父节点的等价链表上，如不存在公共表达式，则对左右孩子分别判断，若存在等价则附加相关位置，然后新建父节点并关联左右孩子。

在处理过程中还有对节点的变量为进行处理：若变量位为临时变量且等价点中存在非临时变量，则交换位置。待构建完成，则进行四元式的重组，按照教材处理方法即可，无难度。







## 目标代码生成

首先设计了目标指令集，在将寄存器，四元式，基本块，代码表等进行了数据结构的设计。将接收来的四元式进行了基本块的划分，设计规则是

第一个四元式是基本块入口。

跳转语句要跳转的四元式是基本块入口。

跳转语句的下一个四元式是基本块入口。

将基本块之间入口的语句进行集合，包括上一个基本块入口，不包括下一个，即可得到基本块。

再就进行四元式活跃信息的判断，设计了判断函数，在基本块中，从后向前遍历四元式，可得到活跃信息。通过检测活跃信息可以对寄存器中的存放值进行存取。本模块采用的时多寄存器存取，对多寄存器进行扫描。

之后即时目标代码的生成。通过目标指令集进行翻译，一个一个进行扫描，检查代码，进行翻译。

输入四元式：

(0,9,"a","b","4");

(1,1,"a","b","t1");

(2,3,"t1","c","x");

(3,7,"-1","-1","7");

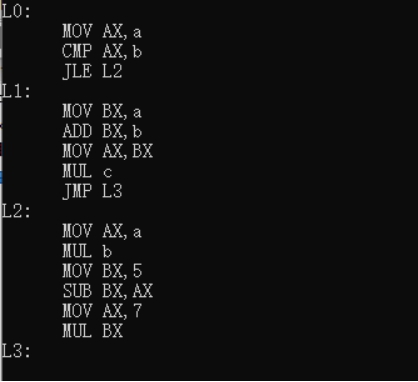
(4,3,"a","b","t2");

(5,2,"5","t2","x");

(6,3,"7","x","y");

(7,0,"-1","-1","-1");

输出：



# 结论

## 王昊天

一个简单明确的语法意味着很容易写成LL1文法，意味着文法中各个推导中重复的不多，而c语言文法是一个较复杂的LL1文法，编写过程中比较随意，在写出文法后需要进一步处理才能得到LL1文法，至此也意识到了越随意的文法实现起来越复杂。在做一件事情前，需要明确事情的难易程度，以及自己的定位，才能事半功倍。

## 邱建福

与其说是课设，不如说这门课是对于编译原理课程与编程能力、团队协作能力等的升级考核，编译器需要有高度耦合的各部分对接组成，而每部分的设计想要完成就必须对于编译原理课程相关知识了解透彻，同时如果想要做一些进阶功能的话，就需要更多的思考和研究。

## 刘荣江

这门课的各个板块高度耦合，在合作完成的时候，开始的讨论必不可少，清晰的对接要求与实现方案规划可以达到事半功倍的效果，反之则容易出现工作大量重复甚至错误的情况，导致人力物力的浪费，我们需要更加注意团队协作的方法。

## 廖真

我们组内有五个人，大家也都有能力做好自己的那部分的，分摊下来大家的任务也相对而言不在繁重，这既是优势，但若配合不当，也会变成5\*1<5的结果。由于接口众多，且实现的方式有较大区别，故花在对接的时间过多，故整体功能未对接齐全。我觉得，虽然我们是该以自己那部分的任务为主，但是我们也应该从编译系统的整体考虑，在做好自己的任务的同时，也要大概了解队友的工作，这样做是有利于我们对接工作的进行的，不会出现做无用功或则存在任务的真空地带。

## 周滋昊

本次课程设计虽说因为某些原因，导致出现了些瑕疵，但总体让我受益匪浅。对编程语言和编译原理，还有汇编语言有了更深层次的来理解，对于整体情况还算满意。

# 参考文献

1、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.

2、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.

3、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.

4、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.

# 收获、体会和建议

## 王昊天

在完成这次课设的过程中，遇到了各种各样的问题，究其原因还是自己没有做过，更没有带领过人数这么多，模块之间的耦合性这么强的项目。没有意识到统一各项内容的重要性，在本次实验中，小看了本次课设的难度，高估了自己和队友的能力，自以为只要按照书上的来，按照c语言现有的结构来，就能很轻松的完成本次课设，而没有讨论细节，在前期自己没有确定符号表的接口，只想着负责符号表的同学根据其他人的需求先设计一下发给我们就好，觉得负责语法分析的同学自己根据c语言确定下文法就可以，而没有在一起讨论，导致了本次实验走了很多弯路，也意识到无论怎样，在项目初期都要有一个足够详细的设计，做好充分的讨论，统一整个队伍的想法，才能在最后实现无偏差的对接，这次队长当的很失败，最重要的两点都没做到，这也是导致了最后没有完成对接的一个原因。

在本次课设中，自己的一些想法也没能实现，没有实现自己所设计的四元式，可能由于自己没有表达清楚，队友初期并没有按照自己设计的四元式来做（应该是为了方便debug）。通过本次课设也意识到，一定要把自己的程序调试好，保证没有bug了再交给队友，否则带给队友的则是成倍的工作量（深有感触）。

当然，虽然没有完成对接，但是由于作为队长，对整个编译器的原理有了一个宏观全面的认识是本次课设最大的收获，思考解决各种各样的问题，遇招拆招也使自己的能力提高了不少。虽然这次课设有点难度，但是自己还是喜欢这种有点挑战的课设的，有挑战，就意味着能学到很多东西。

## 邱建福

这门课设说难也是特别难，主要是各个部分的高度耦合非常考验团队协作，麻烦点在于对接以及怎么把理论变为实践。刚开始写的时候一度非常迷茫，我是负责符号表设计和词法分析的。设计词法分析倒还好，到了设计符号表以及接口函数时，就毫无头绪了，因为我根本不知道队友需要什么函数，甚至功能都不知道，根本没办法写，摸索着写好之后，却发现自己写的和队友需要的大相径庭，甚至完全没办法对接。我觉得很有必要在刚开始重点强调这门课程的高度协作性（我们做过的课设基本都是你写一块我写一块，不那么需要协作，当着手这门课设时候很容易按照以往思维去写。

再来说说收获，虽然只是“线上教学”模式下的课设，但这次课设不止让我感悟了许多，更让我学到了很多——从刚开始分定大致工作“分道扬镳”，到后来意识到各个部分之间的高度耦合，再到后来将自己辛苦思考设计出来的东西全部推倒重做。最后对接发现问题之后学会了怎么合作，学会了了解团队成员的需求，学会了阐释自己的想法与沟通，不只是学到了很多，也非常感谢这门课让我遇到一起努力的小伙伴。

建议的话，凭借自身体会谈谈，不客观也不具有代表性，如果能对老师教学有什么帮助的话，当然是再好不过。

课设验收的优秀评选以及平时对于课设进度的报告都能够看出老师的良苦用心，但课设进度的报告频率可能有些多了。说没有压力当然不可能，在家里很多时候会有各种琐事，写代码可能集中于某一天两天，很多时候隔天的汇报都是几乎什么都没做，还有很多时候计划用来编程结果汇报之后时间最好的半上午过去了，也没有心思写代码了，白白浪费时间。如果非要有进度汇报的话，可以稍微减少些频率，好比两三天一次或者让改变形式——不是让别人监督完成，而是学生自己写，每天如果做什么工作啊，遇到什么问题或者感受都写到日结里就好。

## 刘荣江

因为经历了电脑损坏导致数据丢失的情况，经过这次的坎坷我意识到了随时做好备份是多么的重要，这样即使是工作到一半发生意外，丢失的也只是最近的一次更新，此前的所有进度都会保留下来。编译原理课程设计其实在人生之中只会是一个很小很小的项目，未来会有更多大得多的工程，如果没有良好的备份意识的话，出现意外将会是难以承受之痛。

同时我意识到了，这门课的各个模块确实是高度耦合的，所以对于团队的配合能力要求非常高，一不小心就会出现推到重来的情况，这都是非常浪费人力物力的。以后在遇到这种类型的项目的时候，我们应该不仅仅局限于口头的交流，更应该对于接口的形式和具体的需求实现办法进行文字对接，阐述清楚明白，这样可以更加有效率，避免能力不足之外的问题。

收获的话，对于具体的实现有了一些经验，无论是大概架构还是debug都有了更好的经验，不会出现完全不知道如何下手，停留在课堂理论的情况。对于编译原理这门课也有了更深刻的认识，对我以后学习工作都是有很大的帮助的。

至于建议， 可能是这个线上会议的进度汇报吧，虽然老师这边是没有别的意思，但是客观上会让同学们花费不少时间去考虑如何面对老师的提问，而且线上学习，很多同学在家有一些家务琐事要做，进度大多留在晚上进行，早上开会时间上也会比较麻烦。

最后，辛苦老师们的教学和指导，这个小学期，你们安排的会议、讲座、抽奖活动、优秀评比等等活动我们都看在眼里，真的是非常用心了，谢谢老师们的付出！

## 廖真

这次课程设计相比于前面几次难度加大，同时队友也有4个，但我们人多反而并没有达到预计的效果。其中团队协作意识是我们所或缺的。在这次课设中，我相信我们所走的冤枉路是可以避免的，首先是队友之间的沟通尤为重要，阐述自己的实现方式，尤其是对接部分，大家都要想明白自己的实现和队友的实现，大家都有义务为对接多多考虑，才能发现存在哪些差别，便于最后的对接。其次我们也有及时反馈进度，让后面的队友有准备。如果每个人都清楚的明白自己的具体任务是什么，并积极主动与大家讨论对接问题及实现上的缺陷，那么我相信我们每个人都能人尽其才。

不可否认的是，编译原理课程设计是存在较大的挑战性的。这不仅因为课程本身是一门理论性较强的课程，而且模块较多，在学习的时候容易忘记模块间的联系，在课设时容易带来对接困难，任务分工模糊的问题。同时这也会带来一些实现细节上的小问题，例如在翻译文法的设计上，我的观点时将翻译文法设计成个一个整体，后来拿到语义分析时，却不得不把语义动作拆开，同时队友的实现与自己的实现差距较大，为了契合队友代码不得不做出较大改动，这是我对接上存在的问题，主要也得怪我没搞清楚情况。

十分幸运的遇见了这么负责任而要平易近人的老师，朱老师在群里的话让我感触很多，让我感受到了老师们为了让我们学好的良苦用心，同时教会了我们要学会自己解决问题。这次课真的是十分有深度的课程，学会了很多，感触了良久。

但还是有说一下我对关于进度报告的事，我明白老师的本意是让我们多思考，但是时间有点紧，导致有时两次报告间几乎没啥进度，同时还有其他的课设，时间不好安排，但大家还是会比较在意这些进度报告，会导致浪费大上午的时间，所我建议还是适当降低报告的频率会好点。同时我建议老师在课堂上能多从理论往实践延申，毕竟理论是人脑总结出来的，但是在课程设计实现时也许比较困难，其他的例如最后的优秀评比也是十分有意思的，认清自己的不足。

## 周滋昊

我的任务分配是目标代码生成相关的，所形成的目标代码是汇编语言。最开始我是先将目标代码指令集设定好，但发现负责符号表的同学，还没完成。所以，能是进行初步的设定。然后我进行了对寄存器数据结构的设定，每个寄存器都有其名字和存储数，还完成了寻址函数等函数的设计。接下来的任务要进行程序的主体，对每个基本块进行翻译。

我们主要的交流办法有两个： 1.QQ群进行文字交流和信息收集包括初期的学号手机，分工表填写，之后的四元式展示，会议之前的一些核对 2.腾讯会议进行语音交流和屏幕共享 腾讯会议一般是组长王昊天发起的，我们参与，主要是涉及一些详细内容的讨论

设计整体感觉下来比以往的课设量都要多很多，很多内容都没赶上进度，GitHub的使用也很少。在一开始发现任务量这么重的情况下，确实心里会有一些不耐。也是后来的与团队之间多多交流，之后又听了肖老师的一番话，整体对未来的期待还是有所转晴。希望在之后的环节里好好努力吧。这次课程设计下来，有明显的感觉比以往的课程设计的工作量要大，设计起来也更加繁琐，但能克服下来，这是一件令人满意的事情。 当然，团队协作的密切感也是前所未有的，工作比本身也有着较强的团队密切性，是组员间的交流程度更强。

希望能再之后的学习与工作中，再接再厉。