



# 编译技术实验

---

杨海燕

2022年9月5日

# 2022 年全国大学生计算机系统能力大赛编译系统设计赛

(华为毕昇杯)

## 全国总决赛获奖结果

8 月 23 日, 根据大赛章程和技术方案, 经过大赛评审组的评审和组委会的复核, 2022 年全国大学生计算机系统能力大赛编译系统设计赛(华为毕昇杯)决赛获奖名单公布。

本届大赛决赛共 30 支参赛队入围决赛, 12 支参赛队以外卡身份入围决赛。

最终, 1 支队伍获得特等奖, 2 支队伍获得一等奖, 6 支队伍获得二等奖, 12 支队伍获得三等奖, 8 支队伍获得优胜奖。

外卡赛队共计 2 支队伍获得外卡二等奖, 3 支队伍获得外卡三等奖, 4 支队伍获得外卡优胜奖。

本届大赛全国总决赛获奖名单如下表所示。



全国高等院校计算机系统能力培养官方网站

<http://www.csc-he.com/>

2022年全国大学生计算机系统能力大赛编译系统设计赛(华为毕昇杯) 全国总决赛获奖名单 [按学校名称拼音排序, 队员排名不分先后]		
参赛队伍	学校	赛队成员
特等奖		
赫露艾斯塔	清华大学	焦景辉 王建楠 王子元 李欣隆
一等奖		
喵喵队仰卧起坐	北京航空航天大学	徐睿远 刘伟 董翰元 杨宜凡
啊对对队	清华大学	范如文 张齐颢 郝子晋 徐文博
二等奖		
HexonZ	北京理工大学	朱桂潮 陈新 林恺
邯郸路企鵝编译器	复旦大学	林立达 杜雨轩 聂绍琦 谭一凡
嘉然今天偷看乐	国防科技大学	黄子潇 熊思民 黎梓浩 贺子然
罗杨空队	哈尔滨工业大学(深圳)	梁韬 杨博康 苏亦凡
NKUER4	南开大学	时浩铭 严诗慧 林坤 梅骏逸
从容应队	西北工业大学	王翰墨 王玉佳 郑世杰 乔袁飞龙
三等奖		
(编译(编译(编译器)器)器)队	北京航空航天大学	赵文浩 杜品豪 文显庆 申浩佳
碰瓷的大白菜	北京交通大学	舒航 陈哈阳 杨欢

2022年全国大学生计算机系统能力大赛编译系统设计赛(华为毕昇杯) 外卡参赛队获奖名单 [按学校名称拼音排序, 队员排名不分先后]		
参赛队伍	学校	赛队成员
外卡二等奖		
LoveLive!Virtual Magic!	北京航空航天大学	叶颜团 刘子楠 李世昱 陈诺
编不出来不起床	北京航空航天大学	肖瑜 林璐霞 曹瑞 王岳林
外卡三等奖		
又是困难的取名时间了	北京航空航天大学	温佳昊 陈楚香 唐熙程 欧阳皓东
soredump.i)	北京航空航天大学	郭晓豪 夏雨阳 强生 蔡斌杰
自主可控编译器	华中科技大学	张家荣 秦维聪 蒋韬 郑欣觉
外卡优胜奖		
Elden Compiler	北京航空航天大学	苏俊皓 李真哲 张少龙 冯张翊
正能里荷藕队	北京理工大学	祝雪颖 张浩然 胡书杰 李坤泽
魏公村汽修厂	北京理工大学	何靖琳 王新鲁
HustSysy	华中科技大学	李谨宇 杨佳伟 朱昱洋 杨旺



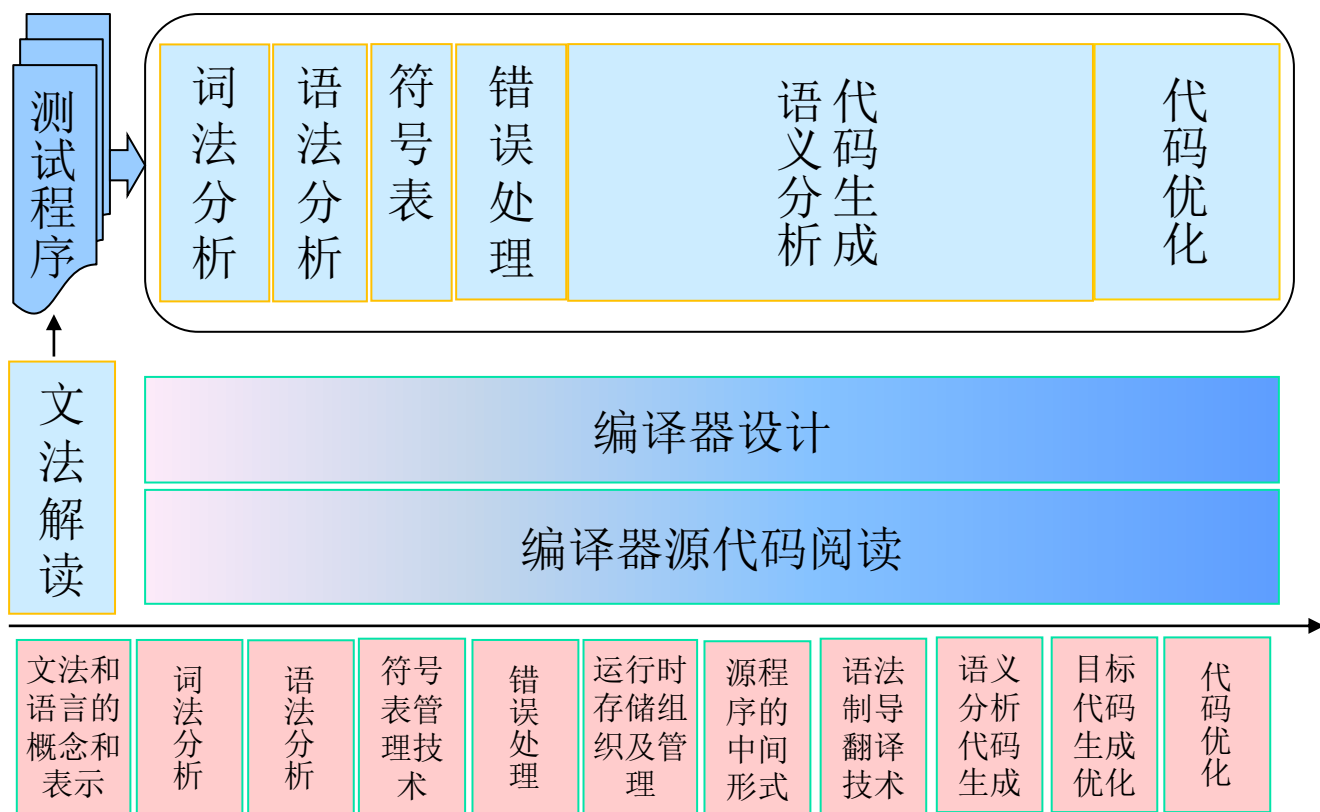
# 三次学习编译全过程

---

- 第一次：概述
  - 大致了解编译的过程和编译程序的构造
- 第二次：第3-10， 14， 15章
  - 详细学习各部分的原理和方法
- 第三次：实验
  - 基于理论学习，逐步实现一个完整编译器

深入  
应用  
巩固

# 理论课与实验作业概览





# 实验题目

---

- 根据给定的文法和要求实现编译器
  - 文法（语法定义、语义约定）
  - 中间代码
  - 目标码
  - 优化



# 文法

---

- **SysY**语言简化版，**C**语言的子集
- 如无特殊说明，语义参照**C**语言
- 具有常量、变量、整数、字符串、一维/二维数组、函数（带参数）、赋值语句、**if**语句、**while**语句、**break**语句、**continue**语句、语句块、输入输出语句等



# 中间代码

---

- 按一定要求自行设计的四元式
- LLVM IR
- 生成PCODE时可以没有中间代码



# 目标代码（三选一）

---

- PCODE
  - 可参照PASCAL-S编译器的设计
  - 需编写解释执行程序对PCODE代码解释执行
- LLVM IR
  - 完成load/store形式的LLVM IR
  - LLVM IR的运行由llc工具完成
- MIPS汇编
  - 生成32位MIPS汇编码
  - 代码生成时合理利用临时寄存器（临时寄存器池），并能生成较高质量的目标代码



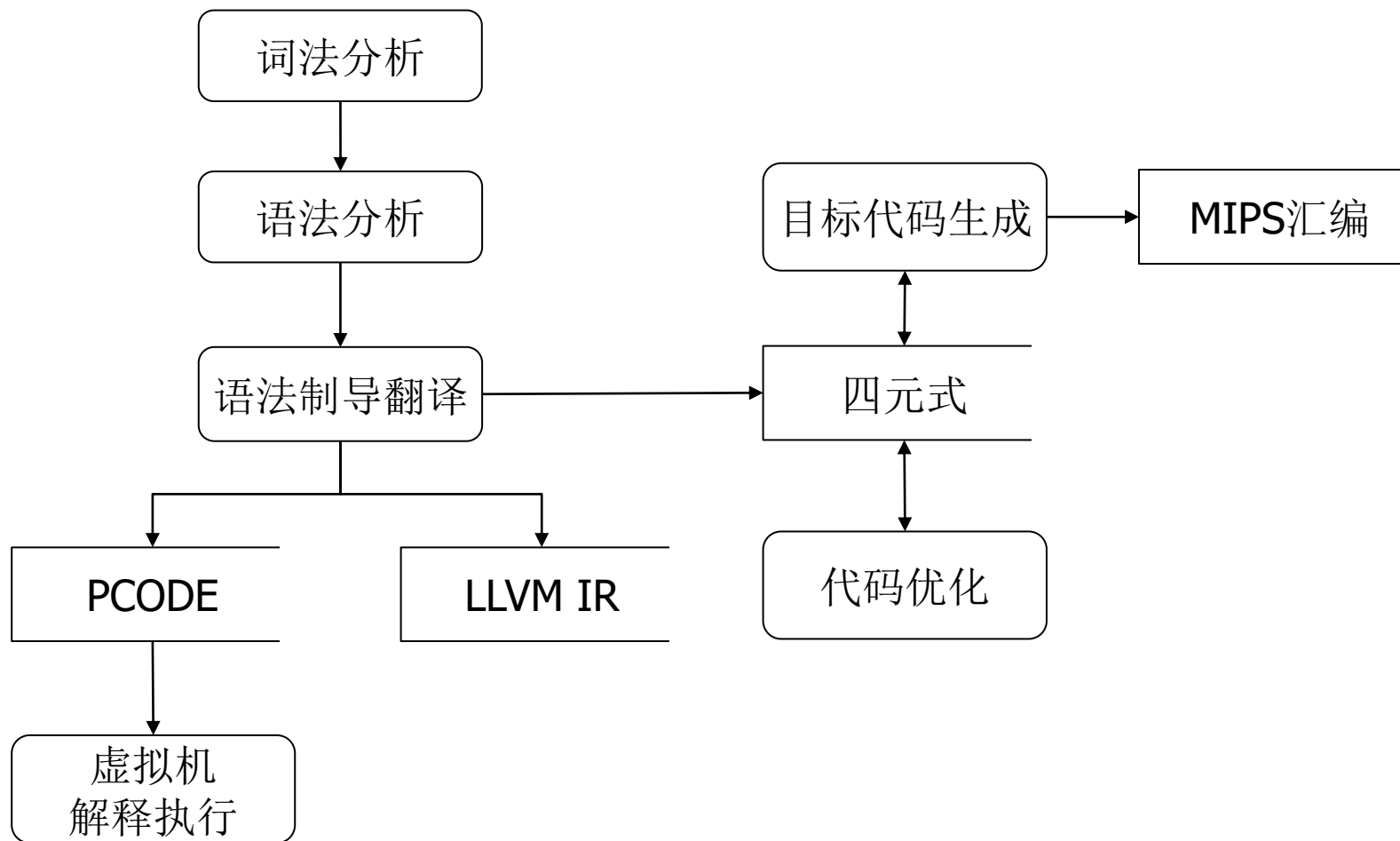


# 代码优化

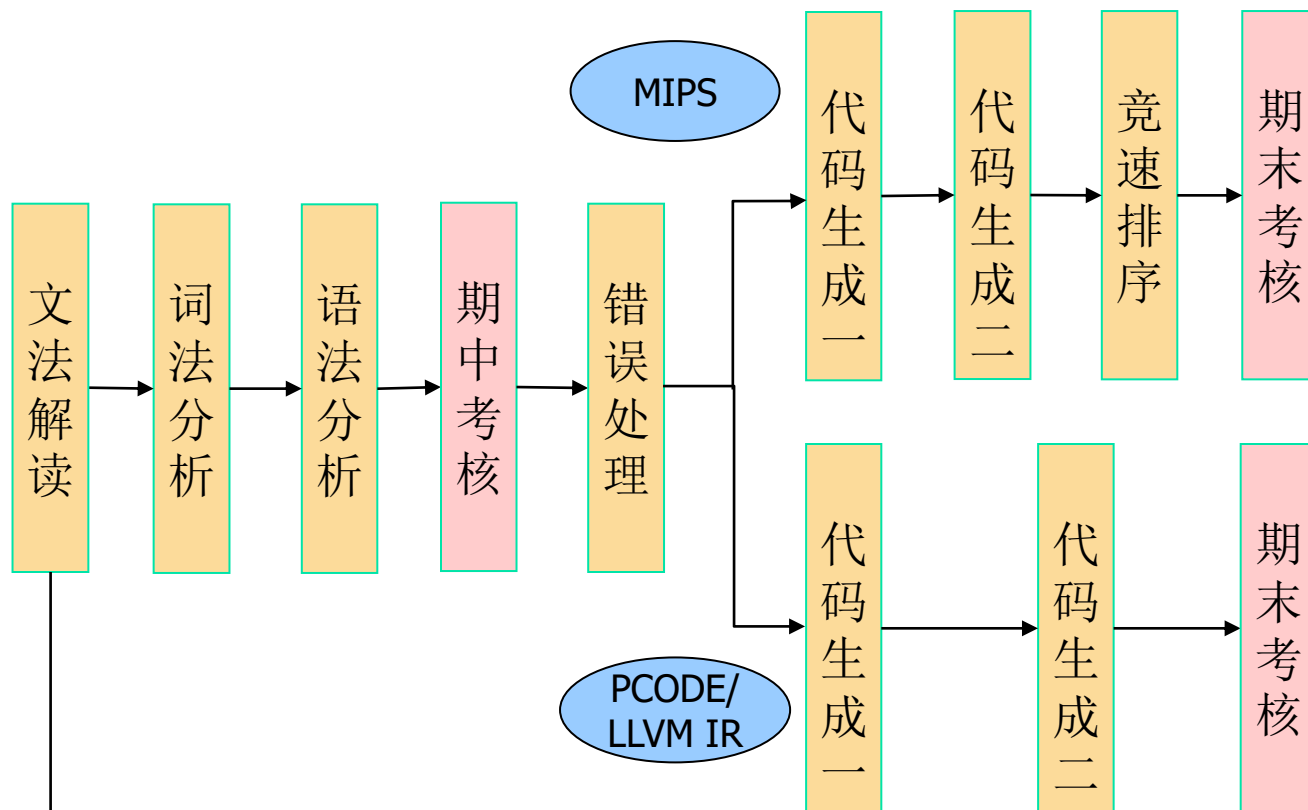
---

- 基本块内部的公共子表达式删除（**DAG**图）；
- 全局寄存器分配（引用计数或着色算法）；
- 数据流分析（通过活跃变量分析，或利用定义-使用链建网等方法建立冲突图）；
- 其它优化自选

# 任务概览



# 任务及考核步骤



建立公共测试程序库用于调试

逐步细化、完善设计文档

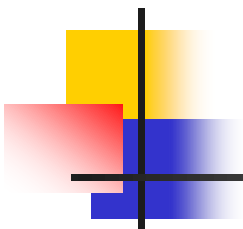
阅读PASCAL-S编译器源代码或其他开源代码



# 评分标准

---

- 生成PCODE解释执行：最高85分
- 生成LLVM IR：最高85分
- 生成MIPS汇编：最高分100
  - 参加竞速排序
  - 完成优化文章



# 分数组成

---

文法 解读	词法 分析	语法 分析	错误 处理	代码生成 一	代码生成 二	代码 优化	文档	期中 考核	期末 考核
5	5	5	5	5	15	15	5	15	25



# 文法解读

---

- 仔细阅读文法，对文法中每条规则所定义的语法成分进行分析，了解其作用、限定条件、组合情况和可能产生的出句子
- 编写**4-6**个测试程序，要求测试程序能覆盖所有语法规则与常见的组合
- 每个测试程序有**10**行输出结果
  - 第一条写语句请输出自己的学号
  - 其他写语句需尽量反映出程序定义的数据及其运算结果
- 文档中列出的需覆盖项只是最基本的



# 文法解读

---

- 请提供每个测试程序的输入数据(有<读语句>则提供，否则只需提供空的输入文件)、输出数据（若输入输出数据没有正确提供，评测时会报错），放到文件中，按下述要求为文件命名：

测试程序及对应的输入输出数据文件分别为

testfile1.txt input1.txt output1.txt

testfile2.txt input2.txt output2.txt

...

testfilen.txt inputn.txt outputn.txt



# 文法解读

---

- testfile1需包含A级规则
- testfile2-3需包含B级规则，不能含有A级规则
- testfile4-6包含C级规则，不能含有A、B级的规则

	Status	Coverage	Info
testfile1	RIGHT	67.0%	
testfile2	RIGHT	76.0%	
testfile3	RIGHT	63.0%	
testfile4	RIGHT	77.0%	
testfile5	RIGHT	43.0%	
Total	RIGHT	100%	
Score		100%	
Detail	恭喜你！样例已覆盖所有语法成分！		





# 词法分析

## ■ 有统一的类别码定义

单词名称	类别码	单词名称	类别码	单词名称	类别码	单词名称	类别码
Ident	IDENFR	!	NOT	*	MULT	=	ASSIGN
IntConst	INTCON	&&	AND	/	DIV	;	SEMICN
FormatString	STRCON		OR	%	MOD	,	COMMA
main	MAINTK	while	WHILETK	<	LSS	(	LPARENT
const	CONSTTK	getint	GETINTTK	<=	LEQ	)	RPARENT
int	INTTK	printf	PRINTF TK	>	GRE	[	LBRACK
break	BREAKTK	return	RETURNTK	>=	GEQ	]	RBRACK
continue	CONTINUETK	+	PLUS	==	EQL	{	LBRACE
if	IFTK	-	MINU	!=	NEQ	}	RBRACE
else	ELSETK	void	VOIDTK				



# 词法分析

---

- 输入的被编译源文件统一命名为 **testfile.txt**;  
输出的结果文件统一命名为 **output.txt**
- 按顺序和格式输出类别码和单词字符串形式

单词类别码 单词的字符/字符串形式(中间仅用一个空格间隔)

INTTK int

MAINTK main

LPARENT (

RPARENT )

LBRACE {



# 语法分析

---

- 在词法分析程序输出的基础上，输出特定语法成分的名字（非终结符）

```
INTTK int
MAINTK main
LPARENT (
RPARENT )
LBRACE {
INTTK int
IDENFR c
<VarDef>
SEMICN ;
<VarDecl>
IDENFR c
<LVal>
ASSIGN =
GETINTTK getint
```



# 错误处理

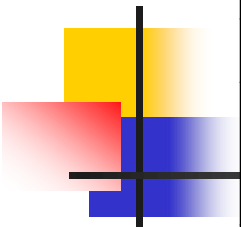
---

- 输入的被编译源文件统一命名为 **testfile.txt**；错误信息输出到命名为 **error.txt** 的结果文件中；
- 结果文件中包含如下两种信息：错误所在的行号 错误的类别码

7 b

46 e

- 其中错误类别码按下表中的定义输出，行号从**1**开始计数



错误类型	错误类别码	解释	对应文法及出错符号(...省略该条规则后续部分)
非法符号	a	格式字符串中出现非法字符报错行号 <FormatString>所在行数。	<FormatString> → “”{<Char>}””
名字重定义	b	函数名或者变量名在 <b>当前作用域</b> 下重复定义。 注意，变量一定是同一级作用域下才会判定出错，不同级作用域下，内层会覆盖外层定义。 报错行号为<Ident>所在行数。	<ConstDef> → <Ident> ... <VarDef> → <Ident> ...   <Ident> ... <FuncDef> → <FuncType> <Ident> ... <FuncFParam> → <BType> <Ident> ...
未定义的名字	c	使用了未定义的标识符报错行号为<Ident>所在行数。	<LVal> → <Ident> ... <UnaryExp> → <Ident> ...
函数参数个数不匹配	d	函数调用语句中，参数个数与函数定义中的参数个数不匹配。报错行号为函数调用语句的 <b>函数名</b> 所在行数。	<UnaryExp> → <Ident> '(' [FuncRParams ] '
函数参数类型不匹配	e	函数调用语句中，参数类型与函数定义中对应位置的参数类型不匹配。报错行号为函数调用语句的 <b>函数名</b> 所在行数。	<UnaryExp> → <Ident> '(' [FuncRParams ] '
无返回值的函数存在不匹配的return语句	f	报错行号为 <b>'return'</b> 所在行号。	<Stmt> → 'return' '{' 'Exp' '}' ';
有返回值的函数缺少return语句	g	只需要考虑函数末尾是否存在return语句， <b>无需考虑数据流</b> 。报错行号为函数 <b>结尾的'}'</b> 所在行号。	FuncDef → FuncType Ident '(' [FuncFParams] ')' Block MainFuncDef → 'int' 'main' '(' ')' Block
不能改变常量的值	h	<LVal>为常量时，不能对其进行修改。报错行号为<LVal>所在行号。	<Stmt> → <LVal> '=' <Exp> ';'   <LVal> '=' 'getint' '(' ')' ';
缺少分号	i	报错行号为分号 <b>前一个非终结符</b> 所在行号。	<Stmt>, <ConstDecl> 及 <VarDecl> 中的 ';
缺少右小括号')'	j	报错行号为右小括号 <b>前一个非终结符</b> 所在行号。	函数调用(<UnaryExp>)、函数定义(<FuncDef>)及<Stmt>中的')'
缺少右中括号']'	k	报错行号为右中括号 <b>前一个非终结符</b> 所在行号。	数组定义(<ConstDef>, <VarDef>, <FuncFParam>)和使用(<LVal>)中的']'
printf中格式字符与表达式个数不匹配	l	报错行号为 <b>'printf'</b> 所在行号。	Stmt → 'printf' '(' 'FormatString{Exp}' ');'
在非循环块中使用break和continue语句	m	报错行号为 <b>'break'</b> 与 <b>'continue'</b> 所在行号。	<Stmt> → 'break';'   'continue';'



# 代码生成

---

- 目标代码的运行结果
  - PCODE: 在解释执行程序上的运行结果
  - MIPS: 用Mars运行的结果
  - LLVM IR: 用llc工具运行的结果
- 分两次作业
  - 先快速实现一个完整编译器
  - 再扩展处理的语法成分



# 竞速排序

---

- 运行结果正确
- 对每个文件计算  $\text{FinalCycle} = 50 * \text{DIV} + 4 * \text{MULT} + 1.2 * \text{JUMP/BRANCH} + 2 * \text{MEM} + 1 * \text{OTHER}$  的值，FinalCycle 越小排名越靠前
- 每个文件根据排名和 FinalCycle 的值给分
- 多个文件则对每个文件的得分加权求和



# 任务及考核说明

---

- 每次任务对应教学平台中一道作业，作业随理论课内容依次打开，若理论课时间发生变化，会相应微调，以作业上公布的时间范围为准
- 每次作业请严格按照输入输出的要求实现，以便准确评判
- 提交后自动评判，若有错误可修改后再次提交
- 作业关闭前可多次提交，以**最后一次结果**为准
- 作业关闭后再提交会扣分，每晚交**24**小时，扣**10%**
- 期中和期末上机考核内容包括现场修改程序、新的测试程序、回答问题等



# 任务及考核日程

	一	二	三	四	五	六	日
第2周	讲解实验内容、文法解读作业打开9.5	9.6	9.7	9.8	9.9	9.10	词法分析作业打开（设计、读源代码）9.11
第3周	9.12	9.13	9.14	9.15	9.16	9.17	语法分析作业打开9.18
第4周	文法解读作业关闭9.19	9.20	9.21	9.22	9.23	9.24	9.25
第5周	9.26	9.27	9.28	9.29	9.30	10.1	错误处理作业打开10.2
第6周	词法分析作业关闭10.3	10.4	10.5	10.6	10.7	10.8	10.9
第7周	语法分析作业关闭、模拟上机考核10.10	10.11	10.12	10.13	10.14	10.15	代码生成第一次作业打开10.16
第8周	期中上机考核10.17	10.18	10.19	10.20	10.21	10.22	10.23
第9周	10.24	10.25	10.26	10.27	10.28	10.29	代码生成第二次作业10.30
第10周	10.31	11.1	11.2	11.3	11.4	11.5	竞速排序作业打开11.6
第11周	代码生成第一次作业关闭11.7	11.8	11.9	11.10	11.11	11.12	11.13
第12周	11.14	11.15	11.16	11.17	11.18	11.19	11.20
第13周	11.21	11.22	11.23	11.24	11.25	11.26	11.27
第14周	11.28	11.29	11.30	12.1	12.2	12.3	12.4
第15周	错误处理作业关闭12.5	12.6	12.7	12.8	12.9	12.10	12.11
第16周	模拟上机考核12.12	12.13	12.14	12.15	12.16	12.17	代码生成第二次作业关闭 竞速排序作业关闭（及感想、文档、文章）12.18
第17周	期末上机考核12.19	12.20	12.21	12.22	12.23	12.24	12.25



# 评测及开发环境

---

- GCC 8.1.0
- Mars 4.5
- Clion 2018.3.4
- Codeblocks 20.03
- Idea 2018.3.6
- Java JDK 1.8



# 上机安排

---

- 编译实验的上机时间：  
2-17周，周一，8-10节（15:50-18:15）  
新主楼F327、F332、F333、F334机房  
北区地下一层B区1-5号机房、曾宪梓科教楼311
- 上机时间可以到新主楼F332机房答疑，不需要答疑的同学可以自行上机完成作业
- 后续会在上机时间组织专题讨论分享
- 第8周和第17周在上机时间分别进行期中考核和期末考核，届时需要同学们到机房上机考核，会提前安排用于熟悉环境的模拟考试，具体安排另行通知

# 作业提交、课程信息获取

■ <http://judge.buaa.edu.cn>

**计算机专业课一体化平台**

[算力平台](#) [比赛](#) [OnlineJudge](#) [毕业设计](#) [GitLab](#) [教师登录](#)

**最新公告**

[数据结构与程序设计 (信息大类)] 2021级期末补考通知 2022/08/26

[高级语言程序设计 (一)] 欢迎参加2022年夏令营活动-程序设计测试, 请阅读注意事项! 2022/07/07

[数据结构与程序设计 (信息大类)] 2021级期末考试通知 2022/06/14

[编译技术] 2022年全国大学生计算机系统能力大赛编译系统设计赛(华为毕昇杯)开始报名 2022/03/22

[数据结构与程序设计 (信息大类)] 欢迎2021级同学访问本系统 2022/03/03



**学生入口**

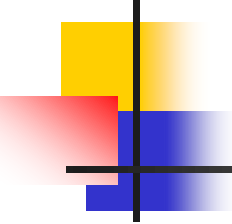
[统一认证](#) [账户登录](#) [助教登录](#)

**校园统一认证登录**

跳转至学校的统一认证平台登录

北京航空航天大学

若重置密码, 请与当前的任课教师联系

- 
- 
- 课程有关的材料请从[judge.buaa.edu.cn](http://judge.buaa.edu.cn)教学平台“课程信息” → “课件下载” 中获取
    - 文法定义及说明文件
    - 实验参考手册
    - ...
  - 所有实验作业都在教学平台发布、提交
  - 需及时关注教学平台的公告、**2022**年编译技术论坛



# 交流与沟通

---

- 现场答疑
  - 每次上机时间（新主楼**F332**机房）
- 交流
  - 组织专题分享、讨论
  - 在论坛上发起话题在线讨论
  - 汇总常见问题发布在论坛
- 联系方式

杨海燕: [compiler\\_buaa@126.com](mailto:compiler_buaa@126.com)  
82317624 G908

史晓华: [xhshi@buaa.edu.cn](mailto:xhshi@buaa.edu.cn)



# 为什么会觉得难

---

- 编译原理本身没弄清
- 需要综合运用多门课知识
  - C/C++/JAVA语言、数据结构、算法、汇编
- 编程经验不足
  - 从头开始做
  - 规模相对较大
  - 调试困难
- 需要自己独立完成



# 你该怎么做

---

- 1. 学习、复习有关知识
- 2. 确定适合自己的难度
- 3. 跟上各阶段步骤，按时完成阶段作业
- 4. 及时做，自己做、坚持做
- 5. 积极交流、沟通





# 你能得到什么

自己动手做了才会有收获  
自己动手做了一定有收获

编译原理本身没弄清

需要综合运用多门课知识

编程经验不足

需要自己独立完成

耐力  
毅力  
体力

巩固了编译原理各个知识点  
能够构造完整的编译器

复习了多门课知识  
学会了知识点在编译器及其构造中如何应用

积累了一定的编程经验  
开发了一个具有一定规模的完整系统

提高了独立解决问题的能力  
遇到问题能设法解决，激发创造性、探索能力

时间管理

.....

想，都是问题，做，才是答案



# 特别提醒

---

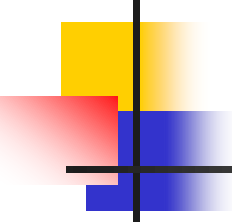
- 1. 选定题目，坚持到底，慎重换题
- 2. 了解各项要求，紧跟阶段步骤
- 3. 有疑问，及时沟通，设法解决
- 4. 相信自己，展示实力
- 5. 立足于自己思考，不要依赖于参考文档
- 6. 成绩公布前，务必自留作业备份
- 7. 确认选课（教务系统、教学平台）
- 8. 期中考试前需要到各处的机房至少各上机一次，确保代码能在各个机房环境下运行、调试



# 特别提醒

---

- 独立完成！！！！
  - 测试程序
  - 代码
  - 文档
- 不能用提交作业代替全面测试
- 误操作需自负后果



---

■ 进入教学实验室上课的学生，须进行身份查验（一卡通）和体温检测，非上课班级学生，不得进入实验室。上课过程中，一旦发现学生体温 $\geq 37.3$ 度或出现其他可疑症状（如精神萎靡、咳嗽等），则学生本人或其同学必须立刻上报辅导员，并按照学校规定的“学生发热处理流程”进行及时处理。


■ 学生就座间距须满足疫情防控要求，佩戴口罩。



# 参考资料

---

- 《编译技术》第17章 第18章及PL/0、Pascal-s编译器源代码
- 参考书
  - Compilers: Principles, Techniques, and Tools. By Alfred V. AHO, Ravi SETHI and Jeffrey D. ULLMAN
  - 中文版：编译原理，李建中，姜守旭译，机械工业出版社  
编译原理，赵建华，郑滔，戴新宇译
  - Advanced Compiler Design and Implementation. By Steven S. Muchnick.
  - 中文版：高级编译器设计与实现，赵克佳，沈志宇译，机械工业出版社



从我自己的开发经历而言，设计是相当重要的一环。对于码量比较大的工程而言，首先要做的不是急于上手敲代码，而是先设计好架构，从整个程序要分成哪些个逻辑部分，每个部分包含哪些功能，部分与部分之间、类与类之间的相互关系，再具体到某个功能采用何种方式实现。今年的编译器允许了使用 Java 语言开发，这样我便可以很好地将上学期面向对象课程所学的思想应用到编译器的设计中。例如语法分析阶段采用层次化设计描述语法树的结构，采用工厂模式编写递归下降解析器最终得到语法树。对于条目繁多的 SysY 文法，通过类的继承与多态，以及接口的使用，这些设计方法无疑是面向对象课程所学的一次效果很好的实践。尤其是到了编译器的中后端，完全自行设计的中间代码，从中间代码翻译为目标代码，以及中间代码的解释执行，针对中间代码和目标代码做的各种优化……每一个环节都非常考验设计能力。在设计完成得很好后，再进行编码也就并非难事了。编码完成后的调试也是一个很重要的环节，而这个环节也体现了设计的水平。如果前期有个好的设计，写出来的代码也是易于调试的，可读性好，逻辑清晰，不但降低了 bug 发生的可能性，也便于调试和修正 bug。

总之，这次编译大作业对我而言是一个巨大的挑战，我本身编程能力不强，这次算是竭尽全力逼迫自己完成。“多思考，早构思，早动手”是我这次编译课设获得的经验，相信这份经验能够在我以后的学习和工作中提供帮助。


首先，我很喜欢课程节奏，从文法解读、词法分析到最后的代码生成环环相扣。除了优化并没有极难的部分，但是比较考验架构设计和细节处理。我从设计自己的编译器的过程中更多明白了“功夫在事前”。

其次，编译课让我更大程度地熟悉了 java 语言。在编译中，经常会希望自己实现一些小功能的函数，这和 OO 是完全不同的体验。而且 OO 当年并没有这么多类，不需要进行 package 管理，这是对我的一次很珍贵地面向对象编程的训练机会。

同时，我认识到了版本管理的重要性。编译是我第一门主动使用 git 管理自己代码的课程，上百次提交悉数记录在我的 git 仓库中。尤其在优化的时候，不小心负优化了就很希望版本回退，这时候 git 就起到了很大的作用。

最重要的是，编译对我的工程能力有了极大锻炼。比如，整个编译器里庞大的变量数量，如何命名就成为了一个需要我们仔细考量的问题。我经常写着写着不知道这个名字想干啥，或者发现两个地方用了同一个名字产生了数据冲突。另外，我深知我的编译能把 OO 当年的 checkstyle 气死，经常有几百行的方法，有超长类。这样规模的编程开发还是很锻炼我的。

但编译课也不只有快乐吧，也有很多很多教训。比如让我难过的期中考试，当时我的语法分析中混入了语义分析，导致无法对维数不匹配的变量进行解读，整颗语法树直接 RE 掉了。这导致课上没有通过。现在也为竞速难过。因为刚考完理论，复习了代码优化的各个算法。我学编译的时候是快乐的，我喜欢这种精妙的设计，感觉优化很好玩。但我却因为其他各种繁杂的事儿，没有时间写了，最后就草草地做了很少的优化，寄存器分配和数据流这种经典优化手段都没能实现。



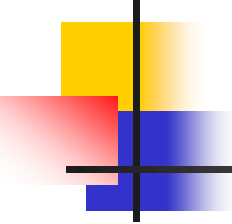
在本学期的学习中，我认为最大的收获是工程能力的提升和宏观把握项目架构的能力。在编译器的前几次作业中，词法分析器仅仅不足300行，语法分析器不足600行，这就使我感到架构开始变得不再清晰，已经出现自己看不懂的代码了。但是在后几次作业中，我对前面作业进行了小规模的重构，将重复代码抽取成为函数，将程序数据流重新设计为易于理解的模式，最后实现生成mips代码后代码量已经接近5000行，但是我仍然能够比较清晰地知道自己的每块代码在做什么。另外，编译器的架构设计使我充分认识到注释的重要性，注释可以帮助我快速理解代码的含义，从而快速掌握代码架构。

本次课设除代码优化部分全部采用文档先行的策略。在语法分析和代码生成两阶段有明显的作用。

一是让coding中的思维深度降低，更轻松。很多需要思考的事情可以直接查找文档中的说明。

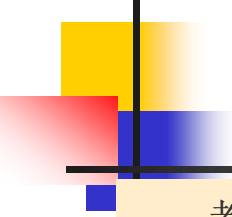
二是让错误率降低，大部分设计和思考停留在文档编写阶段，脱离coding可以使思考更全面。






上学期的00和这学期的编译，我最大的体验就是设计优先，编码其次，尤其是对于这种较大工程的设计。四元式的设计无疑是编译器设计中最为重要的环节，四元式设计的优劣，直接关系到代码生成和代码优化操作。在代码生成第一次作业里，我没能将中间代码的设计和目标代码关联起来，导致我在实现代码生成时反复重构中间代码，花费了大量时间。这一次让我改掉了先写代码再写设计文档的习惯，认识到设计文档优先的重要性。在MIPS代码生成具体实现中，存储分配让我对OS中的内存分配方案认识得更加彻底，数组存取和函数实参传地址具体实现的操作也加深了我对c语言的理解，对指针操作有更底层的认识。最后我在竞速中也取得了较为理想的排名成绩，这离不开前期花费大量时间去思考和设计。课程组提供的往届优秀设计文档也给了我很多启示，让我在代码优化中少走了许多弯路。

当然除了相关专业知识的收获，还有工程架构设计、调试、时间的任务时，有一些不知所措，然而经过设计构思、按部就班地实现功能，经过不断调试，可以正确运行、取得一定优化的效果，最后达到了3500 行的代码量（不算优化前模块的重复代码），编译课程的学已经成为了一次难忘的旅途，回想起当初每每遇到难以解决的bug时常常怀疑是不是应该选择简单一点的PCODE，现在发现选择挑战mips实际上是更正确的选择——没有突破舒适区，就不会有更大的提升。总而言之，这门课有挑战性，但过程中不乏成就感、满足感的收获，让人有“学到了东西”的充实。

- 
- 教训：1. 模块之间尽量解耦，最好可以在统一一种用于交互的数据结构后，两模块间没有任何其他联系。这样在添加功能时如果处理得当，只需对某一个或某几个模块进行修改，而不是牵一发而动全身。比如在我最初设计语法分析器时就没有生成语法树，导致错误处理和代码生成都很难展开，不得已对代码进行了重构。
2. 尽量将函数大小控制在 100 行以内，通过这种限制可以强迫自己将功能抽象为函数。虽然某些函数可能只会使用一次，但是在命名合理的前提下，函数调用可以更清楚的体现这段代码要实现的功能。而且事实上很多代码是可以实现复用的。在写代码生成部分时我就未注意到这一点，导致我写了一个长达 1500 行的巨型函数，在进行调试与修改时都困难重重。
3. 遇到自己不能解决的问题，多查文献，多问问题。
4. 在写 c/c++ 程序时尽量不要在 .h 文件中定义函数（inline 函数除外），否则一旦修改某个函数，所有引用了这个 .h 的文件都需要重新编译，效率其低。



在学期开始之初，我无论如何也想象不到自己可以用并（没）不（有）熟（用）练（过）的 c++ 写出上万行的程序，并且取得还不错的性能。编译实验对我的代码能力和设计能力都有着极大的提升。看到编译器正确运行并且输出目标代码，还是很有成就感的。

在编写大型系统时，一个清晰的设计尤其重要。因此，在编写代码前我都会给自己列出一个步骤清单，先把架构设计想清楚，从整体结构再到每步实现时的注意事项，然后严格按照之前书写的步骤进行编译器的书写。这样做虽然一开始进度可能会慢于周围的同学，但一个清晰的思路会让后面的工作轻松不少。

除此之外，代码风格和命名也十分重要。在编译器的规模逐渐增加时，一些具体的函数实现方法可能已经变得模糊，如果能有一个清晰的命名和辅助的注释，则可以让自己迅速理解函数的作用，并决定是否调用。

总的来说，虽然中间有过很痛苦的时候，但能自己写出一个基本的编译器还是很有趣的经历：)


编译课程具有极高的挑战性，不仅有许多新知识，而且有着不低的代码量。在完成编译课程的学习，尤其是完成编译器课设实验的过程中，我感觉到我的编程能力和编程意识有了不小的提高。

- 先设计，后编码

其实道理都懂，但是在具体编程的过程中往往会贪图一时的编程速度而忽略了设计的过程。然而事实上，如果一开始设计不周，之后往往会发现自己的编码工作与任务目标南辕北辙的情况，因而白白多做了很多无用功。对于编译器来说，先设计后编码的一个挑战是：之前没有做过类似的编译器项目，有些环节（尤其是代码生成和优化环节）不知道怎么写，只能边学边写，有些时候做一些无用功是难以避免的，但是在总体上仍然要坚持先设计后编码。好的设计不仅不会拖慢整个项目的进度，反而会使后面的编码过程更有章法，避免了很多无用功，节约了项目时间。

- 解耦

编译器的构造过程让我深深的体会到解耦的重要性。之前的计组和OO从来没有单个项目有过如此高的代码量。面对这么大代码量的工程，同时把程序所有部分的细节记在脑子里是不可能的，所以每次只能处理一部分。然而如果项目解耦的工作做得不好，那么如果要改动编译器的某个部分，就也会牵扯到其他部分，大大增加了代码维护的复杂度。我尽量把词法分析、语法分析（含错误处理）、符号表管理三部分单独提出，作为一个小模块。但是代码生成过程与代码优化部分（主要指数据流分析）的耦合性还是比较大，加大了编程的任务量，而且影响了任务的完成效果。



编译器的设计让我深刻体会了编译系统的理论基础、构造方法和实现技术。在编写编译器的过程中，我的动手能力能力大大提升了，自然，编译器的设计总会遇到不少困难，主要如下：


(1) 跳转指令的实现，定义了不同的标签，但在编译结束后，会通过遍历将跳转标签替换为跳转到的指令位置即pc值，因此label指令不需要做任何操作。

(2) INT指令的回填，在作用域开始时并不能确定需要令栈增加的指针数，因此需要编译该作用域结束后将增长值回填到作用域开始时。

(3) 数组传参，由于函数调用时实际上处于第1层（全局变量为第0层），因此不能通过相对地址访问到数组值，因此在传参时须计算该数组在栈中的绝对地址，具体方法为将数组的头地址在其所在作用域中的地址偏移，以及作用域的头地址的绝对地址，二者相加即可得到绝对地址。后者的取值是需要在运行时求得。在数组取值时，须区分是否为参数中的数组，如果是直接将参数值取出作为数组头地址进行之后的取值操作即可。


这些困难都是通过查阅资料、与同学交流中不断尝试解决的，我也深刻感受到与同学的交流合作、不断地尝试努力对于完成一个编译系统的巨大作用，让我受益匪浅。





《编译技术》是一门很重要的计算机专业课程，旨在帮助学生了解编译原理，并且自己设计一个小型的编译器。总体来说这门课程难度比较高，需要投入耐心与努力，进行学习，才会有所收获，提高专业知识和素养。

刚开始学习这门课的时候，我的心中其实还有一些胆怯，一个人写一个 C--（简易版的 C 语言，虽然看起来也不是很简易）编译器，这难度也太高了吧？我很担心自己不能完成课程目标，但是害怕又不能解决问题，因此只能投入精力认真对待课程。当我在理论课课堂上认真听课，课后认真完成作业之后，我渐渐发现编译的原理比没有想象中的那么复杂（就像学计组的时候觉得造 CPU 得多难的事，理解原理后就会发现，一个简单的教学用 CPU 也是可以实现的）。从词法分析，把输入的源程序拆成一个个单词，到语法分析，传说中的递归下降我也成功实现了，到这一步，一个用高级语言写的程序的“五脏六腑”就已经被解析出来了。接下来，利用在理论课上学习的语法制导翻译知识，我自己设计了一套 pcode 代码，并且把语法制导翻译加入到了我的语法分析作业中，我的编译程序输出了一套底层的 pcode 类汇编代码，当时的喜悦是无法形容的。在验证了自己的 pcode 基本没有问题之后，我开始着手设计 pcode 解释器，编写完解释器，程序跑出了正确的结果，这是令人十分欣喜的。




从学期初对编译课程的懵懵懂懂，到现在对自己设计的完整编译器结构的了如指掌，我在编译课程设计种学到了很多。

刚开始在做词法、语法分析的时候，我并没有感受到编译器设计的魅力。因为这一阶段完全是按照一定的规则对源代码的结构进行分析，不需要构思和布局，是十分枯燥且费力的工作。而这些工作又恰恰是最需要耐心和细心的，因为需要确保每一处逻辑都严丝合缝，确保每一种情况都能被覆盖并且被正确处理。这一部分让我的代码强度有了显著提升。

而从抽象AST树开始，就需要有构思和布局了。AST结构如何设计方便处理？如何设计便于处理的四元式？四元式的存储结构应该是什么样的？如何导出中间代码更方便？运行栈如何模拟？……每一个设计问题都将一定程度上地影响后续的开发和最终的实现效果，所以必须谨慎思考，以避免不必要的重构。

然而，即使前期有大量的思考和相对完整的设计，也依然难以避免重构的发生。所以，如何降低耦合度，使得每一个功能模块独立化，成为一个更为重要的问题。良好的模块化设计能让此后的重构工作”胜似闲庭信步“。



在我看来，优化部分编译器设计中最重要的一部分，不仅因为优化能有效提高效率，节省资源，更重要的是，这一部分的众多理论与实践成果中展示着令人惊叹的智慧。当看到一些构思精巧的方法时，时常忍不住抚掌而叹一声”妙啊“。虽然构想了很多情况，但是到测试数据上，并不一定能够有很好的效果。面对这种情况，要学会调整心态，能够实现某一项优化，本身就是一项很有成就感的事，即使没有反映到分数上，也是一件值得兴奋的。

编译实验确实是很花费体力的一项工作，很多过程思考起来并不难，但是处理起来细节很多，很繁琐。能坚持下来很考验毅力，磨练意志。

想要把每一种情况都处理好，就需要在设计之初仔细思考，统筹和规划的能力尤为重要。边写边想往往会导致大量的重构，非常痛苦。不过，即使有比较全面的构思，其实也会有一定的重构，因为随着了解的深入，思考的角度发生变化。所以，在编写代码的过程中要尽可能地降低耦合度，让修改更加方便。

从“程序的编写者”到“程序的解读者“，视角的转变也使得我对计算机的理解有了更深入的理解，更能感受到，这小小的四方盒子有种种玄妙。



多与老师、同学、助教交流。（语言上设计的交流，并非互相看代码）

编译实验是我与同学、助教交流最多的一门课程，一方面是因为编译实验确实难度较大，闭门造车难以成功，另一方面也是因为我在学习的过程中感受到了交流的益处。


虽然我们的面对的文法都是相同的，但不同的同学会有不同的设计，有时差距也会很大，如果我们只是按照自己的设计往下走，往往越走越窄，出现了逻辑上的错误也难以发现，但如果我们多与同学进行交流，往往会感叹“原来还可以这么实现！”。在与多位同学交流之后，总结不同设计的优点，再对比自己设计上的不足加以改进，效果或许会更好。

在本学期的学习中，我与多位同学在编译实验的设计上进行了无数次的讨论，尽管我们从未见过彼此写的代码，但在讨论的过程中对不同的设计有了更加清晰的认识，并不断改进自己的程序。

编译原理的课程设计需要尽早选定具体的题目，中途在犹豫编写pcode还是mips的问题上浪费时间和精力并不值得。

在编译器的设计阶段，阅读pascalS编译器的源码以及github上相关的代码，帮助我在初期梳理清了词法分析、语法分析以及中间代码生成这些组成部分的结构。在此基础上，再根据现实的设计在细节上进行修改与完善，比如完善中间代码设计以及解释执行程序的编写、数据结构的选用等，从而相对顺利的完成了代码的设计与初步的实现。

合理安排架构设计与实现的时间，以免一段时间之后需要花费一定的时间来熟悉代码。同时，在代码相关部分推进的同时，也需要合理安排文档进度，一方面完善设计结构，另一方面留以记录熟悉代码。



对于像我这种没有太多编程基础的人来说，本次大作业应该是我第一次写大型的代码项目，一路走来，在作业的完成过程中，我学到了很多，也有许许多多的遗憾。

首先，最令我感触良多的是，针对一个大的项目，事先想好一个优秀的实现架构至关重要。由于代码架构决定着其可扩展性和实现难易度，因此若要在书写代码和debug上少费些力气，就必须在架构设计上多费些功夫。拿我来说，因为在词法分析中没有为错误处理留出足够的提前量，导致我在错误处理时不得不将其和词法分析耦合在一起，以补丁的方式散落在每一个递归下降的函数中，既毁掉了代码结构，也为我中间代码生成留下了祸根。由于前面错误处理纷繁复杂，在尝试MIP两周仍然无法避免大量bug的情况下，我无奈只能选择了pcode。这是我本次大作业完成过程中最大的遗憾，也是最令我吸取的教训。

其次，我学会了一个大项目的书写方法。在错误处理和中间代码生成前，我自认为我的代码结构模块化做的还可以。如果要完成一个大项目，就必须对其进行模块化。通过这次作业，我汲取了一定的经验，为今后书写类似项目积累了经验。

最后，我也对编译器的实现产生了巨大兴趣。在编写编译器的过程中，我尝试读了python的实现代码，尝试从其中得到一点启发。python对于编译器实现十分清晰和简洁，给我带来了许多启发和兴趣。

总之，本次大作业的实现是一段非常难忘的经历，希望我能总结本次作业的经验教训，从中汲取营养，继续在代码之路上前行。

首先，这门课我觉得难度是比较大的，但是学习顺序和每次作业给的时间是比较合理的。难度上我认为主要体现在一下几个方面：首先是代码量比以往的课程都要巨大。以往的课程的一次作业的代码量就是小几百行，比较多的就是 00 的一千出头，而编译这门课程每次作业的代码量有好几次都是大几百行，而代码生成的这一次作业我更是写了两千行左右。另外一点值得注意的是，编译的每次作业都是在上一次作业之上迭代开发的。以往 00 每一单元也是这样，但是并不像这次编译这么明显，而由于前期对于编译程序的整体框架缺乏认知，导致做到一半便需要重构前面的代码的情况十分普遍。如何从一开始为比较大的程序设计一个比较好的框架，为后续开发留有可扩展的空间也是这门课程想让我们体会到的事情。顺便还有一点，那就是这门课程是比较综合的一门学科，可以说是综合运用以往课程学到的知识的课程。首先，需要选择一门你比较熟悉的语言，然后运用合适的设计模式等知识来设计一个比较好的构架，而 ast 树等结构的实现也顺便可以让你复习一下数据结构的知识。而到了代码生成的阶段，你还需要拾起机组 mips 汇编相关的知识，并自己独立设计内存、寄存器分配等策略。可以说，这是一门比较综合的学科，而如果前面的知识学的不好或者有所遗忘的话，可能就会觉得有些困难。但是这些都是暂时的，只要你肯花时间复习学习相关知识，花时间思考相关设计，那么我相信这些其实都是可以克服的。

有些时候，的确有些作业有些难度，让人产生恐惧乃至抵制心理，但是根据我的经验，这些也往往是假象，关键你要敢于静下心坐下来打代码，在打代码的过程中遇到问题解决问题，这样便能让进度快起来，且往往有一种打代码过程中豁然开朗的感觉。我最后根据自身的情况选择了 pcode，没有参加竞速，还是比较可惜的，但是即使这样我也学习到了很多，也很高兴过能够在大三学习编译原理这门课程，实现自己第一个代码量比较大的程序。