



编译技术实验

杨海燕

2024年9月17日



2024 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		
编译系统实现赛道（ARM 后端）		
队名	学校	队员
一等奖		
喵喵队花样滑冰	北京航空航天大学	王宇奇 张杨 高揄扬 鹿煜恒
二等奖		
原末鸣初	北京航空航天大学	陈锐 冯冠霖 孟乔缘和 吴一凡
pinyinggaoshou	杭州电子科技大学	张力 潘昭宇 陈载元 林雨茹
正道的光	西北工业大学	苗潼超 谭钰蓁 袁竟程 马丁
三等奖		
OneLastCompiler	北京邮电大学	伏培赓 马志 丁嘉豪 龙帅然
牛牛向前队	国防科技大学	胡定中 刘哲浩 刘亚鹏 赖宇
Compiler_vs_Bugs	复旦大学	王浩涵 刘帝恺 王宇晖 李叔禄
优胜奖		
编译你好香	国防科技大学	刘晨希 汪诗凡 杜宇琪 施鸿润
BanGDream!It'sSYSY	南京大学	王朝晖 王陈洋 孙忆秋 余明晖
伪指令	电子科技大学	王宏飞 张玉超 郑洋 黎睿
编译成蓝色疾旋融	青岛大学	李少凡 陈冠霖 李明谦

2024 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		
编译系统实现赛道（RISC-V 后端）		
队名	学校	队员
一等奖		
NEL	北京航空航天大学	杨辛晨 杨承钊 何立群 范吴运维
八云蓝架构编译器与 RE 的橙	南开大学	华志远 张昌昊 孔德嵘
人工式生成智能	南开大学	梅骏逸 郭大玮 仇科文 冯思程
二等奖		
return_0;	清华大学	游旺 张一可 薛志宇 仇成宇
GPT5.0	电子科技大学	罗钰浩 钟震 郭闰航 王南钦
三进制冒险家	国防科技大学	汤翔晨 侯华玮 杨仰众 简泽鑫
NNVM	南京大学	陈泓宇 刘治元 刘洪源 徐天行
素履“译”往队	北京航空航天大学	张博睿 张哲 单江涵 董天振
编编又译译	电子科技大学	姚欣扬 许芳煜 龚晓阳
世界第一可爱 Fuyuki	清华大学	陈英豪 魏辰轩 于新雨 李骋昊
三等奖		
四个圣甲虫	中山大学	陈俊儒 梁爽 韩云昊 王骏越
一刻也没有为段错误而哀悼	中国科学技术大学	吕思翰 宋业鑫 周珉翔 缪言
派大星说搞优化就像光 头强抓美羊羊☺	中山大学	冯一鸣 陈溪泉 吴健强 赵文清

2024 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		
外卡参赛队		
队名	学校	队员
三等奖		
喵喵队花样滑冰	北京航空航天大学	王宇奇 张杨 高揄扬 鹿煜恒
睿睿也想打编译队	北京航空航天大学	石睿知
四元式	北京航空航天大学	杨一堂 柳政尧 李南冰 钟熙桐
逸动山楂队	北京航空航天大学	张政 张域铮 张博豪 董睿琪
海底小纵队	北京航空航天大学	陈思潮 焦子谦 郭润林 周泽同
优胜奖		
周末疯狂拼	北京航空航天大学	张俊华 严少泽 刘笑鹏 于恩泽
这是个队名队	北京航空航天大学	纪邳炀 沈铜 高鹏飞 蒋孝淳
pinyinggaoshou	杭州电子科技大学	张力 潘昭宇 陈载元 林雨茹
冲向广州队	南京大学	陈子昂 于翔 张兰 张正
YellowYaks	北京航空航天大学	刘奕哲
BanGDream!It'sMySYSY	南京大学	王朝晖 王陈洋 孙忆秋 余明晖

2024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）

主办单位：全国高等学校计算机教育研究会 系统能力培养研究专家组 系统能力培养项目发起高校

承办单位：杭州电子科技大学

协办单位：华为技术有限公司 HUAWEI “101计划”编译原理课程虚拟教研室 机械工业出版社 希冀平台 CSDN 希冀

媒体支持：CSDN SDN

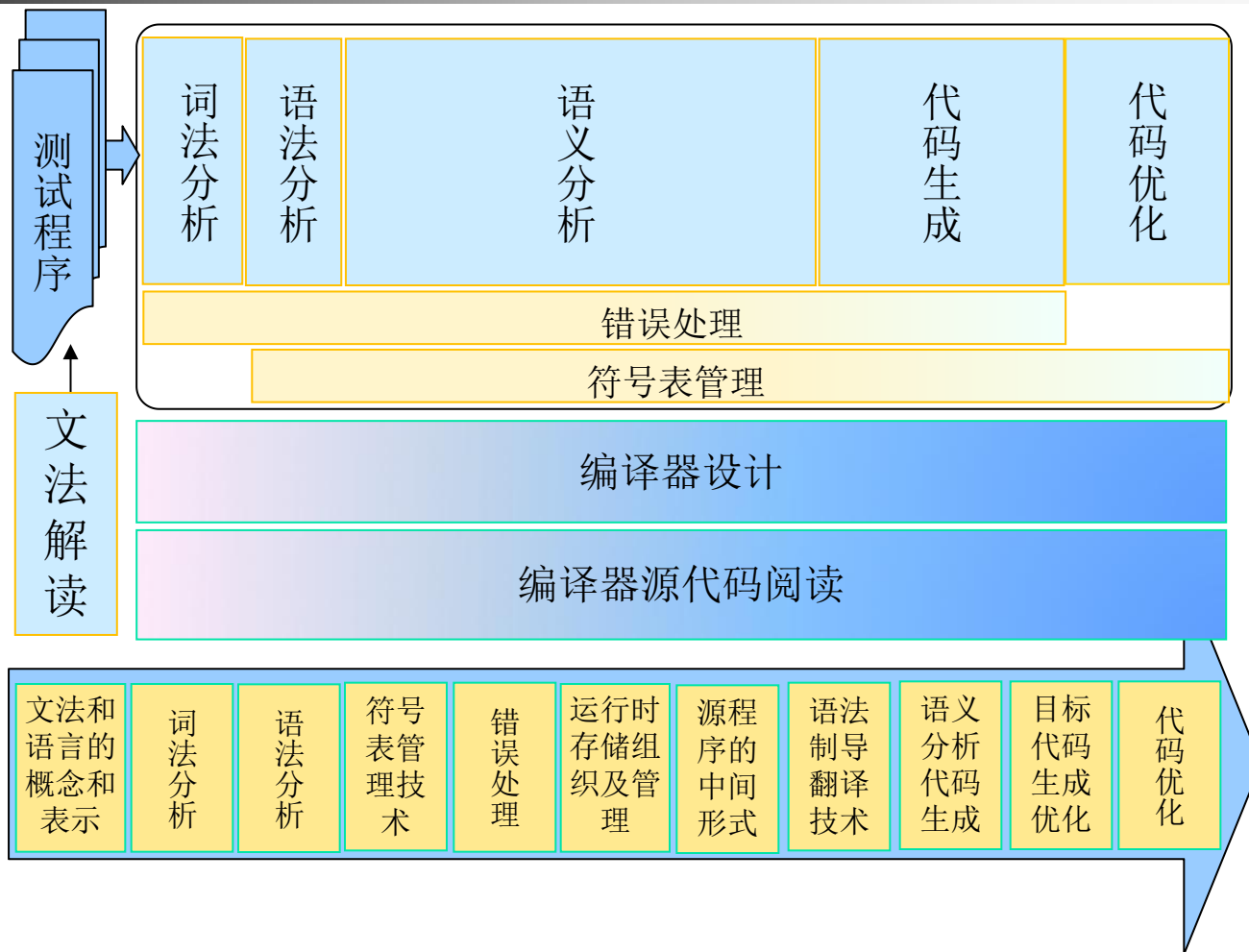


三次学习编译全过程

- 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、字符串、一维数组、函数（带参数）、赋值语句、**if**语句、**for**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
- 也可以用**LLVM IR**作为中间代码
- 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

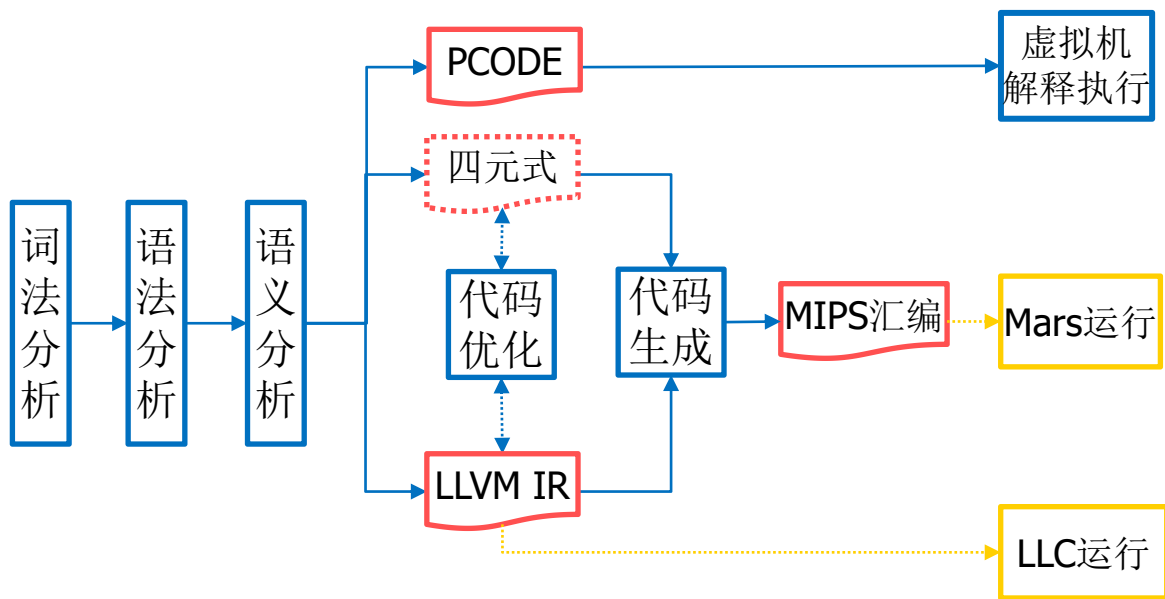
- 生成32位MIPS汇编码
- 代码生成时合理利用临时寄存器（临时寄存器池），并能生成较高质量的目标代码



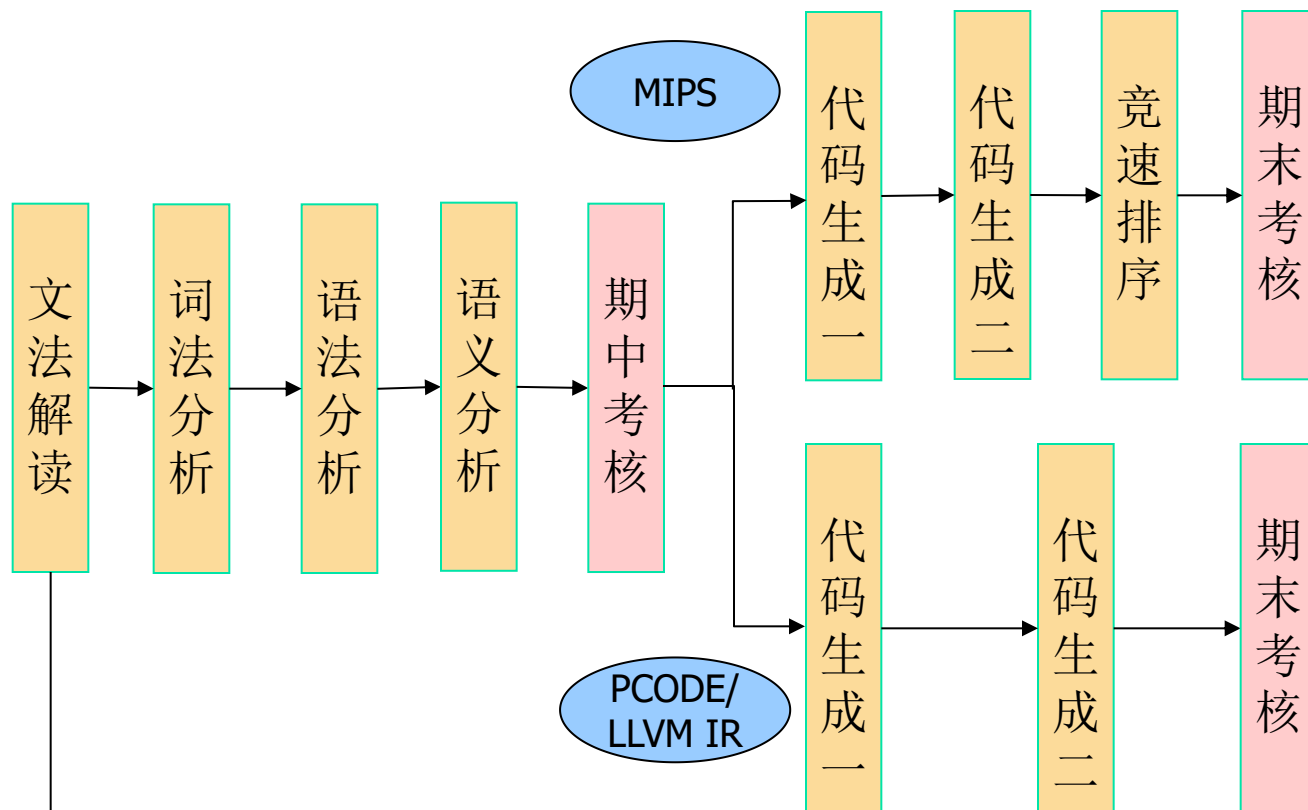
代码优化

- 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

任务架构



任务及考核步骤





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	语义 分析	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

评判结果: WrongAnswer, 本次评测得分比例: 94%。

测试点信息

测试点	得分比例	说明
testfile1	79.2%	Case Accepted
testfile2	84.4%	Case Accepted
testfile3	0%	Wrong Output
testfile4	56.2%	Case Accepted
整体覆盖率	94%	缺少const变量的字符串初始值 缺少char类型函数 缺少首forstmt与cond缺省的for循环语句 缺少cond和尾forstmt缺省的for语句 缺少getchar赋值的语句



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	else	ELSETK	void	VOIDTK	;	SEMICN
IntConst	INTCON	!	NOT	*	MULT	,	COMMA
StringConst	STRCON	&&	AND	/	DIV	(LPARENT
CharConst	CHRCON		OR	%	MOD)	RPARENT
main	MAINTK	for	FORTK	<	LSS	[LBRACK
const	CONSTTK	getint	GETINTTK	<=	LEQ]	RBRACK
int	INTTK	getchar	GETCHARTK	>	GRE	{	LBRACE
char	CHARTK	printf	PRINTF TK	>=	GEQ	}	RBRACE
break	BREAKTK	return	RETURNTK	==	EQL		
continue	CONTINUETK	+	PLUS	!=	NEQ		
if	IFTK	-	MINU	=	ASSIGN		



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 **lexer.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)

LBACE {



词法分析

- 有非法单词时输出至error.txt
- 输出非法单词所在的行号以及错误类别码

```
int main(){  
    if(1 & 2){ // 非法单词 &  
        printf("2024 Compiler\n");  
    }  
    return 0;  
}
```

error.txt:
2 a



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
IDENFR c
<VarDef>
SEMICN ;
<VarDecl>
IDENFR c
<LVal>
ASSIGN =
GETINTTK getint
```

- 有非法单词和语法错误时输出出错行号和错误类别码



语义分析

- 输入的被编译源文件统一命名为 **testfile.txt**;
- 在语法错误的基础上增加**语义错误**;
- 对于错误的源程序，输出出错行号和错误类别码
- 对于正确的源程序，输出编译完成后的**符号表**

```
int a;  
int foo() {  
    a = a + 1;  
    return 1;  
}  
int main() {  
    int a = 1;  
    printf("%d",a + foo());  
    return 0;  
}
```

```
1 a Int  
1 foo IntFunc  
3 a Int
```

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	出现了 '&' 和 ' ' 这两个符号，应该将其当做 '&&' 与 ' ' 进行处理，报错行号为 '&' 或 ' ' 所在的行号。	LAndExp → LAndExp '&&' EqExp LOrExp → LOrExp ' ' LAndExp
名字重定义	b	函数名或者变量名在 当前作用域 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 函数名 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 函数名 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'return'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， 无需考虑数据流 。报错行号为函数 结尾的'}' 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 前一个非终结符 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 前一个非终结符 所在行号。	函数调用(<UnaryExp>)、函数定义(<FuncDef>)、基础表达式(<PrimaryExp>)及<Stmt>中的']'
缺少右中括号']'	k	报错行号为右中括号 前一个非终结符 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'printf'所在行号。	Stmt → 'printf'('(StringConst{,Exp})')';'
在非循环块中使用break和continue语句	m	报错行号为'break'与'continue'所在行号。	<Stmt>→'break'; 'continue';'



代码生成

- 考核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
 - 再处理扩展的语法成分
- 应始终具备错误处理能力



竞速排序

- 运行结果正确
- 对每个文件计算 $\text{FinalCycle} = \text{DIV} * 50 + \text{MULT} * 3 + \text{JUMP/BRANCH} * 3 + \text{MEM} * 4 + \text{OTHER} * 1$ 的值，在运行正确的前提下，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(9.9)					文法解读作业打开 讲解实验内容		
第3周(9.16)				词法分析作业打开（设计、读源代码）	词法分析讲座		
第4周(9.23)				语法分析作业打开	文法解读作业关闭 语法分析讲座		
第5周(9.30)					词法分析作业关闭		
第6周(10.7)				语义分析作业打开	语义分析讲座		
第7周(10.14)					语法分析作业关闭 模拟上机考核		
第8周(10.21)				代码生成第一二次作业打开	期中上机考核		
第9周(10.28)					四元式，LLVM讲座		
第10周(11.4)				竞速排序作业打开	语义分析作业关闭 PCODE，MIPS讲座		
第11周(11.11)					代码优化讲座		
第12周(11.18)							
第13周(11.25)					代码生成第一次作业关闭		
第14周(12.2)							
第15周(12.9)							
第16周(12.16)				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）	模拟上机考核		
第17周(12.23)					期末上机考核		



评测及开发环境

- Clang12.0.0
- Java JDK 17
- Mars 2024(编译专用)
- Clion 2019.3.6
- Idea(机房现有版本)
- 可以使用CMake进行项目管理



上机安排

- 编译实验的上机时间：
2-17周，周五，8-10节（15:50-18:15）
新主楼F327、F332、F333、F334机房
北区地下一层B区1,3,4号机房
- 上机时间可以到机房答疑，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讲座
- 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

- <https://judge.buaa.edu.cn/>
- 帐号为学号，以前的密码

The screenshot shows the homepage of the '计算机专业课一体化平台' (Computer Professional Course Integrated Platform) for Beijing University of Aeronautics and Astronautics (BUAA). The page features a navigation bar with links to '公开课' (Open Course), '实训' (Practical Training), '算力平台' (Computing Power Platform), '比赛' (Competition), 'OnlineJudge', '毕业设计' (Graduation Design), 'GitLab', and '教师登录' (Teacher Login). On the left, there are three data panels: '评测总次数' (Total Evaluation Times) with a value of 10,387,811, '总代码行 不包括云实验' (Total Lines of Code, excluding Cloud Experiments) with a value of 151,952,562, and '云实验提交人次' (Cloud Experiment Submission Times) with a value of 2,651. Each panel includes a small bar chart showing trends over time. The center of the page features a large image of a BUAA campus scene with a pond and trees, overlaid with the text '84 在线课程' (84 Online Courses). On the right, there is a '学生入口' (Student Entrance) section with buttons for '统一认证' (Unified Authentication), '账户登录' (Account Login), and '助教登录' (Teaching Assistant Login), followed by a prominent blue button for '校园统一认证登录' (Campus Unified Authentication Login). The footer contains the text '北京航空航天大学' (Beihang University), a password reset notice, and a copyright notice for CourseGrading 7.5.13.

计算机专业课一体化平台

公开课 实训 算力平台 比赛 OnlineJudge 毕业设计 GitLab 教师登录

评测总次数
10,387,811
过去一周新增 101,067 过去一天新增 605

总代码行 不包括云实验
151,952,562
过去一年新增 45,821,071 最近学期新增 16,258,962

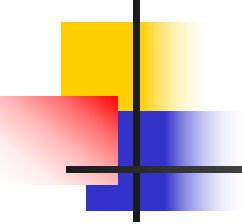
云实验提交人次
2,651
当前学期提交人次 6 当前学期实验总时长 0 小时

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

84
在线课程

北京航空航天大学
若重置密码，请与当前的任课教师联系

Copyright © CourseGrading 7.5.13 旗舰版

- 
-
- 课程有关材料请从实验平台中获取
 - 文法说明文件
 - 实验教程
 -
 - 所有实验作业都在实验平台发布、提交
 - 需及时关注平台的公告、论坛



实验教程

- 介绍编译器开发完整流程
 - 前端部分：词法分析、语法分析、语义分析
 - 中间代码：LLVM、四元式、Pcode
 - 代码优化：寄存器分配、SSA.....
 - 目标代码生成：MIPS
- 提供配套示例编译器 **tolangc**
- 旨在为编译器的架构设计和实现提供参考
- 教程位置：
 - 希冀平台 - 在线教程
 - 项目地址：<https://github.com/wokron/tolangc>



对大模型的使用要求

- 辅助学习和理解理论知识



- 辅助编程和编译器开发



- 严禁使用LLM生成各种文档或文档的部分章节



具备完善判断机制，违反者可能被取消实验成绩！



查重要求

- 出现项目代码整体相似度过高，或其他高度相似的情况的，视为代码相似。
 - 抄袭者与被抄袭者同等处理
 - 被查出代码相似的学生有权提出申诉
- 以任何非正当方式获取测试用例、直接输出结果的，均视为作弊行为。
- 编译技术课程组保留关于作弊行为的解释、调查和处置的权力。



交流与沟通

- 现场答疑
 - 每次上机时间（新主楼F332机房，北区1号机房）
- 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

82317624 G908

史晓华: 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



特别帮助

尽管大家在之前的课程中积累了许多编程经验，但是独立设计一个编译器仍可能是不小的挑战，最终的代码量可能有上万行，因此非常考验大家的编程能力和项目管理能力。

需要帮助？

- 觉得自己独立完成编译器有困难？
- 没有思路，无从下手？
- 很难通过评测题目？

帮助别人！

- 学有余力，对完成编译器很有信心
- 乐于分享自己的学习经验
- 愿意抽出时间帮助其他同学

联系我们

不论你需要帮助，还是想加入我们帮助其他同学，随时联系自己班级的助教，我们会尽可能地提供全面的帮助！



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. 期中考试前需要到机房至少上机一次，确保代码能在机房环境下运行、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



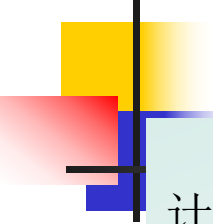
参考资料

- 《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


在本学期实验中我觉得最重要的是对一个**大型项目的整体设计**，花在设计上的时间不会白费，而是为了更好的编码。因为前期写的比较匆忙因此在代码编写中我也出现了很多问题，如前期没有设计好接口或者没有预留错误处理所需要的部分从而导致后续代码十分难编写，甚至进行了几次小规模的重构，白白花费了很多无用的时间，这更让我意识到了**设计的重要性**。课程的层层递进教学能让我能够认真了解每个阶段的任务后进行合理的设计，同时预留了充足的时间让我们进行设计和反思，相信在本次实验中所学到的知识一定能应用在我今后的学习中。

在本学期实验中我获得的另外一点收获就是要提高自己**代码的规范性**，无论是代码风格还是参数的命名都十分重要，这直接影响了后期对代码调试的效率。因为整体代码量较大，后期调试时对于前期所写的函数或者需要使用的变量如果没有一个清晰的命名或者注释，就要花费很多时间去重新理解，十分影响效率。这让我想起了 OO 课上的 **checkstyle**，当时十分不在意如今却被自己的代码风格所影响导致效率低下，这也让我得到了教训，也坚定了我之后提升代码风格的决心。

整体来说，虽然整体在实现编译器的过程中有十分多的困难，但最后完成时收获了很多，也得到了满满的成就感。相信带着这份收获，吸取教训，一定能够在今后的学习上不断进步。



编译课是一门非常有挑战性的一门课。它需要我们自己一个人完成架构设计，最终完成一个几千行的编译器。值得庆幸的是，它的**理论部分与实验部分的联系**是非常紧密的（对比计组和操作系统，狠狠表扬），理论课的学习使得我们在实验的早期，对编译器有**宏观的认识**，非常有利于整体架构。要知道，对于大型系统的编写，架构设计可谓重中之重，如果没有设计好就草率地编码，那很难使一个几千行的系统正常运行，完成编译功能。回顾这一学期与编译相处的时光，既有苦涩也有喜悦。很多时候，坐在电脑前，一写就是一整天，很多个晚上，独坐电脑前被 bug 折磨到深夜；但每一次改完 bug，程序能够正常运行时，内心的喜悦也是其他任何时候都无法获得的。从刚开始的不知从何下手，到完成整个编译系统，我能清晰地感受到编译带给我的成长，轻舟已过万重山！



面向对象设计


现在觉得面向对象在编译技术之前上确实是很有道理的，因为编译实验中涉及到了大量的对编译器的“组件”进行抽象，然后进行面向对象的封装。比如说在进行语法分析的时候，需要为代码生成一个语法生成树，这里便需要面向对象的设计思想为非终结符结点设计相对应的类。在编译技术实验课程中，每一次迭代开发也是锻炼了我的面向对象的设计思想。同时，很多在OO中学到的设计模式，也在编译技术实验中得到了实践，比如说访问者模式和工厂模式。

模块化开发

模块化指的是把一个大项目进行功能划分，细分为不同的小模块进行开发。其实无论是理论课中学到的编译器的七个关键步骤还是实验每次布置的迭代任务，都蕴含了模块化开发的想法。一个编译器最终可以分为不同的组件，比如词法分析器、语法生成器、中间代码生成器等等。这样模块化开发的方法在工程中也具有很大的用处，不同的后端代码生成组件就可以让编译器的目标代码生成别的汇编语言。只要保证组件之间的接口是一样的，就可以很轻松地更换编译器所使用的具体组件。

在理论学习部分，我**认真参与了每一节课**，并且**及时复习准备小测**，它既让我没有在期末的时候手忙脚乱，也指导我一步步完成了实验，但是只学理论知识，学习深度是远远不够的，自己动手做一个编译器才能让人真正学懂编译。从最开始简单的词法分析、语法分析，到之后难度骤升的代码生成和优化，这期间我遇到过很多困难，但是我**从来没有想过要放弃**。我自己设计了中间代码，这让我有一种这个编译器属于我自己的幸福感，这是**自我创造和思考**带来的快感。之后为了优化代码，我一遍遍地看指导书，看不懂指导书又去看相关论文和博客即使是实现之后也遇到很多bug，每次交上去一片红都让我觉得离成功差之千里，最开始我在每一个样例中都能找到不一样的错误，优化后又不断暴露更多的错误。在一遍遍的对照、调整、修改之后终于完成了mips生成和优化。

除了代码设计层面，还有就是对实验部分的实践深度的感慨。**代码优化**能做的事情太多了，而且对于每一种优化方法，能做到的程度又有不同。就例如**冲突图**的构建，**活跃变量分析**看活跃范围是一种。看基本块级别的变量定义点的活跃变量是一种。通过**定义使用链**，构建网然后进行语句级别的定义处活跃进行判断又是一种。其中又以网的效果最好。还有在进行冲突图着色的时候，怎么选取一个"合适"的变量，以及对不能着色的变量，转变为新的活跃范围很小的临时变量，然后进一步进行活跃分析，**寄存器分配**，这些都是很有门道的。总之，**编译实验的深度是非常深的**，而且可以接触到许多比较前沿的理论，例如SSA等。



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。

虽然课程难度不小，但是能感受到**编译实验课和理论课设计**的巧妙，达到了**相辅相成**的效果，用理论课知识完成实验内容，用实验内容实践去巩固所学知识。

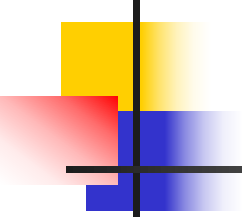
我的编程水平相当不行，在做编译每个阶段的实验前都有不小的畏难情绪，对自己能做出这件事很没有底，可以说每阶段的实验在读完实验要求后都有点无从下手的感觉，不过经历千辛万苦每阶段都能完成，完成后都会比较欣慰。

在每次完成阶段性任务后我都会习惯性的回望一下完成前的自己，当时还是差点连题都读不懂，但是现在已经做出来了，不免会感到令人满足的充实感，能看到自己是实打实地进步了。

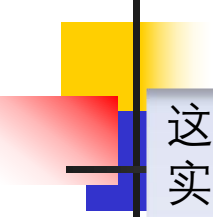
首先就是编码时**对于整体架构的规划与设计**，这一点在之前的算法和 java 等课程可能体现不出来， 因为之前面对的问题规模较小， 但是在编译实验中， 虽然划分成了好几个实验， 但其实这些实验之间是紧密联系的， 每一个部分的编写都是在搭建整个架构的一部分， 会对之后的编码产生很大的影响， 因此在编写之前的设计显得十分重要， 我也因为前面架构的问题在错误处理以及代码生成时遇到了很大困难， 最后进行了重构才成功完成。 一个好的架构对于大型的工程项目时十分重要的， 我在今后的学习过程中也一定会谨记这一点。

第二点就是在编码过程中， **对于细节的处理**十分重要。 因为无论是在语法树生成还是语义分析的时候， 由于文法众多， 我们需要讨论很多情况， 有很多重复的代码块， 也有类似但细节上有不同的代码， 在编码过程中， 我们很难保证一遍就实现 0 差错， 所以在写代码时， 在编写每一部分的时候， 要对每一步的处理都有过分析以及检查， 把代码的结构理清晰， 注释什么的都要写好， 这样我们在之后的debug 以及优化时可以节省很多时间， 也可以规避一些重大 bug 的产生， 减少大重构的几率。


第三点是在面对这么庞大的编程任务时， 我们虽然要对总体进行设计， 但是如果一直停留在构思和设计上， 我们会不断产生畏难情绪， 并且对于进一步的深入设计也会越来越难， 因为人的思考能力是有限的， 但是编写过程中可能产生的细节问题是数不清的， 因此我们有时候不一定非要把所有部分全部想明白再开始编写， 而是可以想一部分写一部分， 再保证这部分的正确性之后， 就可以投入到下一步的编写中， 一次只考虑未来几步， 一步一步来， 最后完成整个部分的编码。 很多问题的产生本质上是因为想的多做的少， 即使想不明白的问题也可以先试着写写， **在实践中不断思考**， 才能逐渐把问题分析透彻。



我认为这次编译实验给我带来最大的收获就是让我对计算机科学与技术学科的认识更加清晰了，建立了更加完整的知识结构。在之前我学习计算机组成、操作系统等科目，我了解到很多计算机硬件和底层软件的交互，了解到了计算机只能运用0和1进行运算；在我学习数据结构、程序设计、面向对象思想等科目的时候，我学习到了该怎么样设计开发程序，一个程序和软件系统的基本构成，了解到了人类可以通过一些高级语言“告诉”机器该怎么运转。但是，这两者之间究竟是怎么联系起来的，我一直不能完全理解，这也是我觉得计算机这一学科非常有魅力的地方，人说的话究竟是怎样准确无误的编程0和1字串的、怎么就能够正确执行.....这次编译实验告诉我，我可以通过一些文法约束，完成词法分析、语法分析、语义分析、代码生成等等的工作而完成一套形式化的翻译过程，这就完成了我之前学习知识的关键连接部分，让我感觉到非常的通透。



这一学期的编译实验让我**从零开始搭建了一个编译器**，在实验的过程中确实遇到了许多困难，但解决之后对编译原理的理解也更深了一些。从词法分析、语法分析到符号表管理、错误处理以及之后的中间代码生成和解释执行，前期的工作还是较为简单，到后期考验也是越来越多。在这几次实验中我最大的感受便是**要动手去做**，很多时候总是希望自己能够想清楚再动手去做，却发现越想越难以入手，越想感觉问题越多，但当我真正开始写代码后才发现很多问题并没有之前想的那么抽象，在已经成型的代码基础上解决一些问题会比空想好做很多。同时在实验过程中**保持冷静与耐心**也十分重要，很多bug并不能很快找出来，评测一个点过不去自己写测试文件找问题找了好几天也是家常便饭，这中间的过程无非是痛苦的，但真正找出来问题之后也能有真实的收获，并且这些问题往往也是逻辑硬伤，找到之后对于自己的理解也有很大的提升。一个更大的体会是对**实践出真知**的感悟，在写实验时虽然并没有对此有很大的感触，但结合到理论发现真正写过这些内容对于理论知识的理解有着巨大的帮助，还记得理论课上第一次听可能有些困惑的问题，在实验完成过程中或者实验完成后会有恍然大悟的感觉，再回顾理论知识之时也觉得感悟至深。




在本次编译实验中，我收获良多。首先，我意识到了要对编译实验**充满耐心**。编译实验代码量庞大，每一个细节都可能影响整个系统的正确性，与其急于求成，不如花费更多时间仔细审视代码，保证其功能的正确性。在调试过程中，我们会遇到各种各样的问题，只有耐心地结合测试数据一点点排查，更好地理解代码的执行过程，才能解决遇到的问题。其次，设计在此次编译实验中显得尤为关键。这算是我第一次接触一个相对庞大的项目，由于经验不足，一开始我的代码结构和模块划分并不理想，在中期几乎是进行了一次重构。这次实验让我认识到，**好的设计**能够事半功倍。在下一次编写类似规模的程序时，我将更加注重整体架构的设计，以便更加高效地完成任务。在此次任务中，我也意识到了交流的重要性。自己死磕问题容易陷入陷阱，适当的与同伴交流可能会有意想不到的收获，能帮助我从另一个视角看待问题。

总的来说，这次编译实验是一次充满挑战但也充满收获的经历。通过这次实验，我不仅提升了自己的编程能力，还深刻体会到了坚持、设计和交流的重要性，受益匪浅。

设计和文档很重要，设计和文档很重要，设计和文档很重要！由于前几次作业在设计上下的功夫不够，不得已在语法分析作业时从头到尾做了重构。而在早早完成语法分析作业，大概有一个月没有继续编译实验后，再回头完成代码生成作业时，看到我写的代码竟然没几个注释、设计文档里也没有详细说明时一脸茫然。在花费大功夫理清架构之后，由于时间紧张仍没有仔细设计就开始进行代码生成的任务。虽然最终艰难地完成了作业任务，但基本是缝缝补补、漏洞百出，代码丑陋得不忍直视。

一开始在九月实验的词法分析刚开放时，我发现难度并不高，以为编译实验并不像大家讲的那样困难；直到十一月底我才将MIPS代码生成完成，并没有加入任何优化时，我才深刻意识到了学长学姐们在感想中提到的**早点开始**的含金量。因此在这里我也希望未来将要进行编译实验的同学们，如果你想参与竞速环节，请不要认为时间很充裕，最好在十一月之前完成LLVM IR代码的编写，并尽早开始代码优化。



这门课程是非常系统的。这门课程从文法出发，给我们讲述了一个编译器的完整流程。不管是编译器的过程中的某个环节还是编译器的优化以及相关文法的介绍。通过这门课的学习能让我们对编译技术有了整体的把握。而编译课设，也是一环套一环，让我们最终完成了一个完整的编译器。

这门课程的学习过程是有挑战性的。毋庸置疑，这门课程的难度是比较大的，从最简单的文法分析，到一个几千行的编译器，各个组件的相互协同，各个细节的把握都需要我们花很多时间去调整。由于代码数量较多，完成代码后的debug过程也是繁杂而痛苦的。但是在这样的挑战中也进一步磨练了我们的编程能力和大工程的能力。

大型代码能力

这学期的编译实验的项目复杂度对我来说相当的高，特别是代码生成开始，任务量和设计量便陡增，开始我不知所措，不过不开始做就永远不可能做出来，按照LLVM中部分的划分一步一步推进后，将任务拆分成多个难题，终究还是能过举步维艰的前进的。



1. 理论知识的掌握

在本学期的编译技术课程中，本人深入学习了编译器的构造原理，掌握了从词法分析，语法分析，中间代码生成，目标代码生成，最后进行中端后端代码优化的整个编译流程。通过实际编写编译器，我更深刻地理解了理论知识，并且在编译器实现时也时常会想起理论所学，并尝试结合起来融会贯通。

2. 编程能力的提升

通过使用JAVA语言编写编译器，我的系统级编程能力和面向对象思维得到了极大的提升，不得不说像是经历了一次oopromax大作业，从架构设计到功能实现应有尽有。

3. 优化技术的了解

我认为代码优化称得上编译的灵魂所在，我进行了mem2reg、图着色寄存器分配，局部公共子表达式删除，死代码删除，乘除法优化，基本块合并等诸多常见优化。在优化过程中，大大增进了我查询资料，构建算法的能力，也使我的图论等基础知识进一步巩固增强，不仅使我学习到了编译器优化的基本方法，还了解了提高程序运行效率的重要性。即使仍和第一梯队有所差距，最后的结果也足以让自己满意。非常好优化，使我的晚饭时间消失！