编译原理课程实验方案的设计与实施

中国科学技术大学　计算机科学与技术学院

张昱　陈意云

0551-3603804，{yuzhang,yiyun}@ustc.edu.cn

http://staff.ustc.edu.cn/~yuzhang/compiler

http://staff.ustc.edu.cn/~yiyun

第二届全国高等学校计算机实验与实践教学研讨会

2009年7月25日　威海·山东大学威海分校

提纲

- 国内外编译原理实验简述
- 实验方案综述
- 部分实验支持库的设计思路
- 实验方案的特点
- 实验方案的实践及经验教训

国内外编译原理实验简述(1)

❖ 国内情况

- 清华大学、北京大学、北京航空航天大学设立单独的编译原理实践课程
- 北京航空航天大学：
  - 5种难度级别的20多种文法，
  - 学生选择级别，系统确定要实现的文法，
  - 抽查答辩
- 一部分学校：
  - PL/0及其扩展
  - 熟悉编译器自动生成工具的使用

国内外编译原理实验简述(2)

❖ 国内情况

➢ 中国科学技术大学

- **1984~1993**：实现扩展PL/0语言到扩展PL/0抽象机的编译器、实现扩展PL/0抽象机的解释器

  数组类型和函数类型

- **1994~2001**：单独的实践课

  布尔类型、exit/break语句和布尔表达式短路计算

- **2002~2006**：本科学制五年改成四年，单独的实践课被取消，仅开展熟悉Lex和Yacc的课程小实验

- **2007~**：以“源语言－抽象语法树－低级中间表示－汇编代码的内部表示－x86/MIPS汇编”为主线，每个学生完成编译器的前端(后端)，自行选择合作伙伴，分组逐个答辩

国内外编译原理实验简述(3)

❖ 国外情况

几所美国名校(Stanford, UC Berkeley, UCLA, Cornell)普遍要求:

- 实现一个小型面向对象程序设计语言的编译器;
- 与理论课程同步展开;
- 分成多个课程设计，最终完成一个完整的编译器;
- 理论课教学知识面宽;
- 实验分值占总分值的比例高。

国内外编译原理实验简述(4)

❖ 国外情况

- **Cornell: CS 412/413 Introduction to Compilers**
  - 词法分析 →语法和语义分析→代码生成→数据流分析→演示.
  - 4个书面作业 20%＋实验 45%＋ 2 次测验 35%

- **UCLA: CS 132 Compiler Construction**
  - LL(1)分析器->MiniJava的类型检查-> Piglet ->Spiglet->Kanga->MIPS
  - 期中考试15%+期末考试30%+实验55%

国内外编译原理实验简述(5)

❖ 国外情况

- UC Berkeley: CS164 Programming Languages and Compilers
  - Python子集的词法分析和语法分析→静态分析→代码生成.
  - 6个书面作业 ＋3个实验 ＋ 期中考试＋期末考试

- Stanford: CS143 Compilers
  - 词法分析→语法分析→ 语义分析 →代码生成
  - 4个书面作业4个实验 70%+期末考试30%

提纲

- 国内外编译原理实验简述
- 实验方案综述
- 部分实验支持库的设计思路
- 实验方案的特点
- 实验方案的实践及经验教训

实验方案的综述

- 实验方案设计的指导思想
- 实验方案的主线与组成要素
- 实验开发环境与工具
- 实验软件包
- 课程设计内容
- 实验方案的实施建议

计算机科学与技术学科的本科教学目标

计算机科学与技术学科的毕业生除了要掌握该学科的各个知识领域的基本知识和技术之外，还必须具有较扎实的数学功底，掌握科学的研究方法，熟悉计算机如何得以实际应用，并具有有效的沟通技能和良好的团队工作能力。

——中国计算机科学与技术教程2002(CCC2002)

专业实践及其意义

- 专业实践的形式

  课程实验、实习、毕业设计、竞赛、……

- 专业实践的意义
  - 激发学生的学习兴趣
  - 巩固和消化课堂知识
  - 提高实践技能
  - 培养创新能力
  - 增强交流能力

存在问题

- 毕业生在就业中暴露出的一些问题
  - 动手能力较差
  - 分析问题、解决问题能力较薄弱
  - 岗前技术培训：其中有些是学校可以做而没有做的
  - 岗前工程意识、质量意识和团队精神的教育：学校也可以逐步培养的
- 课程实验中存在的问题
  - 各课程的课程实验各自独立
  - 内容陈旧、覆盖面窄、综合性不高、难度低、规模小
  - 不注重对学生工程、质量、团队等意识的培养
  - 学生数增加，检查力度和深度不够
  - 学生对课程实验的热情未调动起来，拷贝风气增长

专业实践改革

❖指导思想

首先抓课程实验改革，课程实验要整体规划

- 像讨论教学计划那样来讨论技术水平的培养：语言、工具、平台的覆盖范围和相互之间的衔接。

❖其覆盖程度依赖于

- 制度的保证、学科机构的资源、教职人员的利益
- 软件类的课程实验
  - 学生至少参与完成一个有一定规模的软件项目的设计与开发，涉及对多门课程所学原理的综合运用
  - 遵循由小到大、循序渐进的原则
  - 整体规划课程实践涉及的语言、工具和环境
  - 注意培养工程意识、质量意识和团队意识

专业实践改革

软件类的课程实验

- 低年级的课程实验（如C语言、数据结构）
  - 以巩固课程知识的小实验为主
  - 训练学生基本的程序设计技能
- 高年级的课程实验（如编译原理、操作系统等）
  - 以综合运用的课程设计为主
  - 训练学生软件工程的能力

编译原理课程实验方案设计的指导思想

- 定位：综合运用的课程设计
  - (通过合作)为某个语言设计开发一个可运行的编译器
  - 为适应不同学校的要求或者适应不同学生的个性化学习，布局多个小型课程设计

不同学校可结合本校教学目标和学生情况选择合适的课程设计作为学生的课程实验内容。

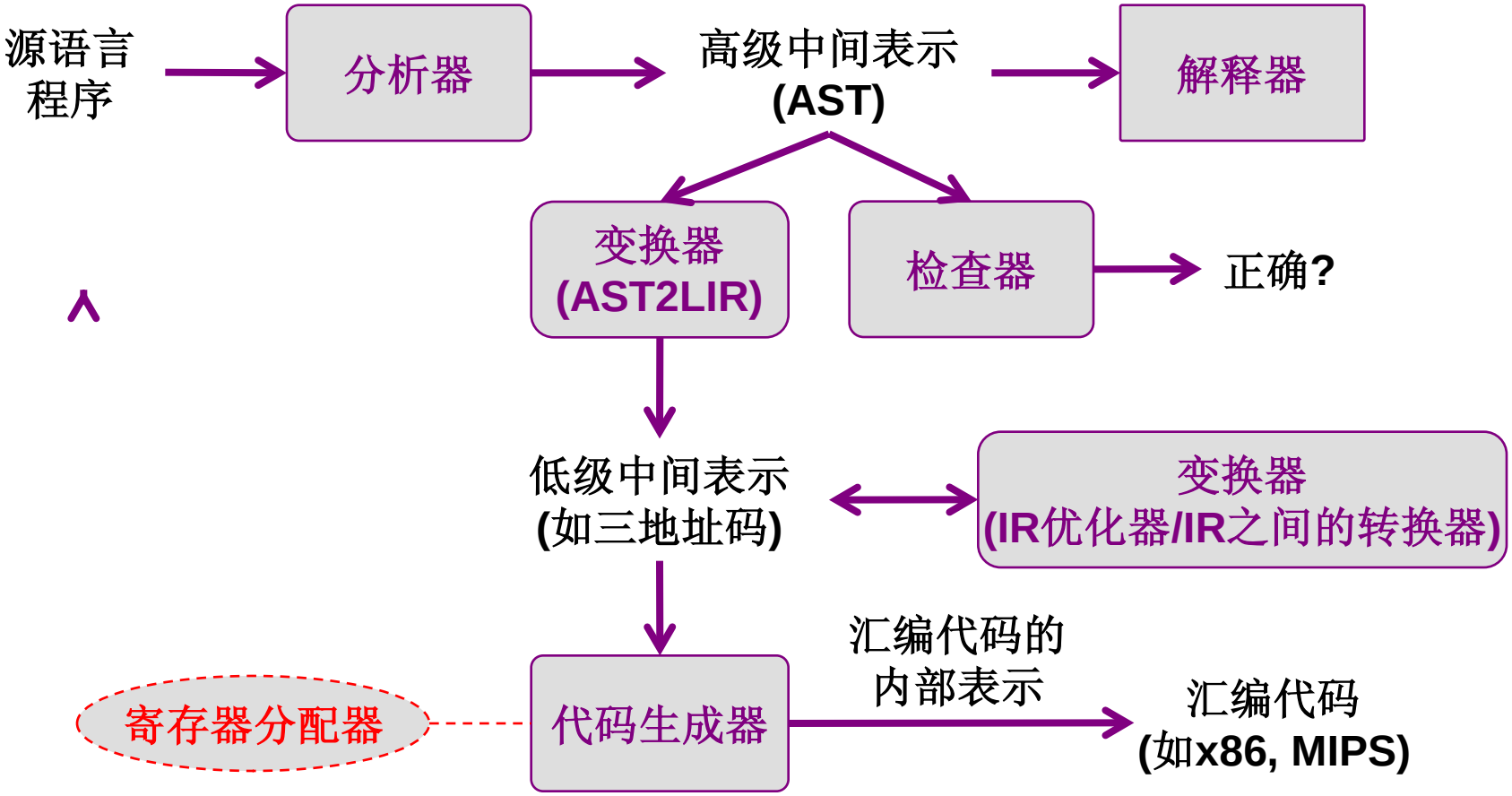
学生可以结合自己的实际情况进行选择，不同难度系数的课程设计的得分不一样。

实验方案的综述

- 实验方案设计的指导思想
- 实验方案的主线与组成要素
- 实验开发环境与工具
- 实验软件包
- 课程设计内容
- 实验方案的实施建议

实验方案的主线

以“源语言－抽象语法树－低级中间表示－汇编代码的内部表示－x86/MIPS汇编”为主线搭建的、基于组件的编译原理实验体系



实验方案的组成要素(1)

❖要实现的源语言

- **SimpleMiniJOOL语言**
  - 单函数的无类型结构化语言，变量默认为整型，if/while/continue/break
  - 通过实现该语言的一个简单的解释器，打开编译实验的大门
- **SkipOOMiniJOOL语言**
  - 多函数的强类型(整型、布尔型、字符串型以及元素类型为这三者之一的一维数组类型)结构化语言
  - 用于展开各种编译器组件的骨干实验，帮助学生理解、消化和掌握编译原理实验
- **MiniJOOL语言**
  - 强类型的面向对象语言
  - 用于展开有关面向对象编译的高级实验，帮助学生了解现代面向对象编程语言的一些编译技术和方法

实验方案的组成要素(2)

- 目标机的选择
  - x86 IA32：CISC
    - 从面向目标程序的角度来增强指令功能
    - 从面向高级语言和编译程序以及OS的优化实现的角度改进指令系统
    - 指令系统复杂，各种指令使用频率相差悬殊

    在普通PC机上，利用GCC将x86汇编码编译连接成可执行文件
  - MIPS：RISC
    - 选取使用频率最高的指令，并补充一些最有用的指令
    - 每条指令的功能应尽可能简单，并在一个机器周期内完成
    - 所有指令长度均相等，只有load和store指令才访问存储器

    SPIM模拟器上执行，或在龙芯机上利用GCC将MIPS汇编码编译连接成可执行文件

实验方案的组成要素(3)

❖ 中间表示

通过中间表示，可以将一个完整的编译器或解释器分解成若干规模适度、接口清晰的编译器组件

便于设计和实现支撑课程实验的软件库和实验运行平台

便于规划各个课程设计并提供实验的框架代码

- ➢ 高级中间表示
  - 能清楚地反映源程序的语法结构
  - Eclipse JDT中的AST—JLS3（无符号表，有声明结构）
- ➢ 低级中间表示
  - 旨在降低汇编代码生成的难度，便于开展优化
  - 提供一种以三地址码为基础的低级中间表示及其实现 LIR

提供统一的中间表示访问接口，用户可以自行实现它，扩展新的中间表示实现

实验方案的组成要素(4)

- 面向对象特征的编译
  - 分析器：MiniJOOL程序→具有面向对象特征的AST
  - AST2LIR：　AST→不具有面向对象特征的三地址码

可以与SkipOOMiniJOOL共用基于LIR三地址码的代码优化器和代码生成器

实验方案的组成要素(5)

- 汇编代码的内部表示
  - 汇编程序中的语句有伪指令、标号语句、指令
  - 不同汇编语言在具体的伪指令、指令的种类和编码上存在差异
  - 不同汇编语言在语句格式上存在一些共性之处

  开发一套统一的汇编代码内部表示库**AIR**

  其中，对于依赖于具体汇编语言的部分，主要通过编写汇编语言特征配置文件、利用配置文件自动生成程序、动态加载汇编语言特征得到运行时数据等手段来实现

实验方案的组成要素(6)

- 编译器组件类别
  - 分析器：源程序→中间表示

    如：语法分析器、带语义检查的分析器
  - 解释器：中间表示→输出执行结果
  - 检查器：中间表示→true/false

    如：语义分析器
  - 变换器：中间表示→中间表示

    如：AST到LIR的转换器、LIR优化器
  - 代码生成器：中间表示→输出目标代码(汇编代码)到指定文件

    如：x86汇编代码生成器、MIPS汇编代码生成器

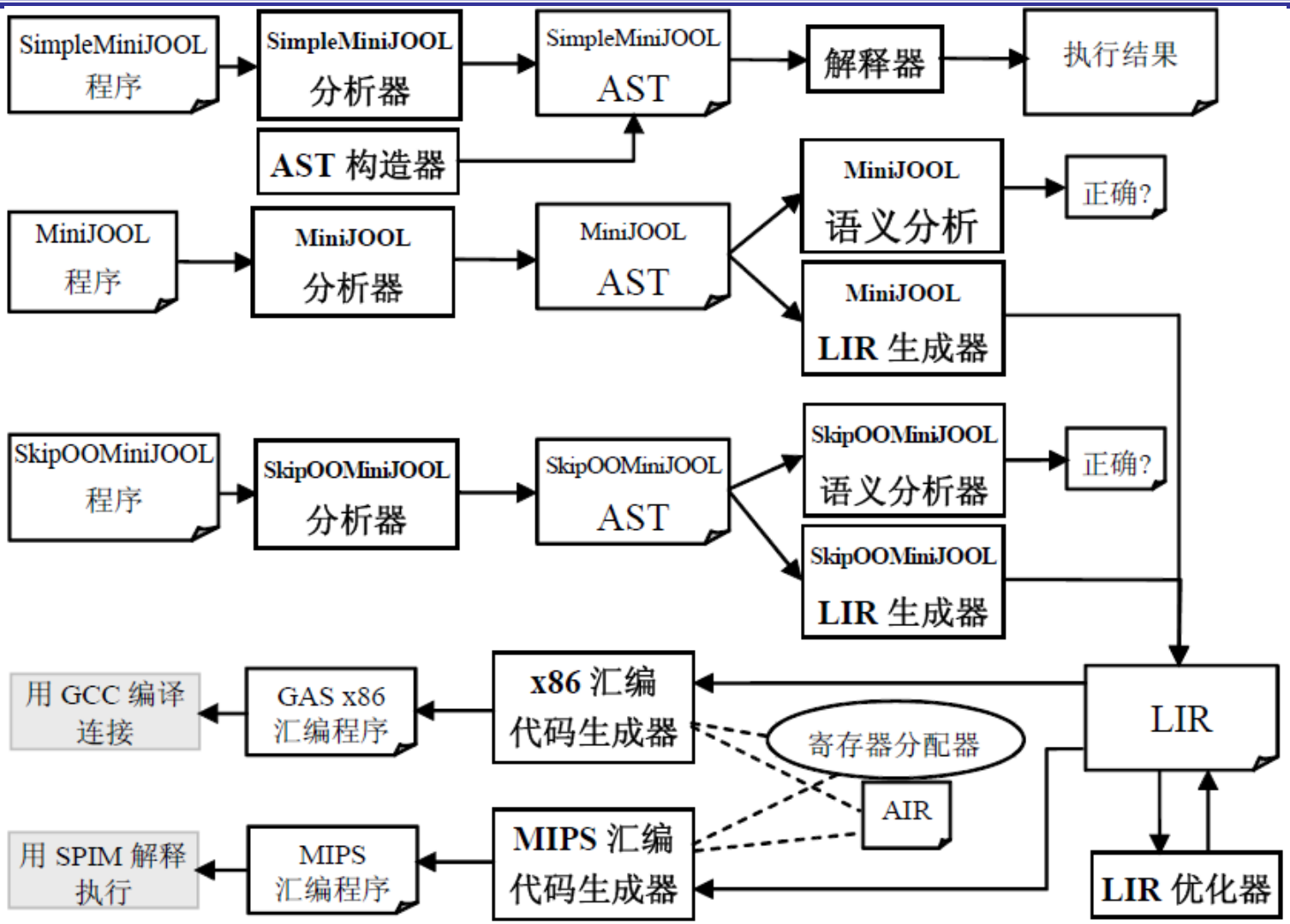
实验方案的组成要素(7)

- 实验开发环境与工具
- 支持课程实验的实验软件包

  实验教材的配套光盘
  - 实验支持库
  - 课程设计开发包
- 实施方案

以上内容在后面依次介绍

编译器实现框架


SimpleMiniJOOL 程序
SimpleMiniJOOL 分析器
SimpleMiniJOOL AST
解释器
执行结果
AST 构造器
MiniJOOL 程序
MiniJOOL 分析器
MiniJOOL AST
MiniJOOL 语义分析
正确?
MiniJOOL LIR 生成器
SkipOOMiniJOOL 程序
SkipOOMiniJOOL 分析器
SkipOOMiniJOOL AST
SkipOOMiniJOOL 语义分析器
正确?
SkipOOMiniJOOL LIR 生成器
用 GCC 编译连接
GAS x86 汇编程序
x86 汇编代码生成器
寄存器分配器
AIR
LIR
用 SPIM 解释执行
MIPS 汇编程序
MIPS 汇编代码生成器
LIR 优化器

实验方案的综述

- 实验方案设计的指导思想
- 实验方案的主线与组成要素
- 实验开发环境与工具
- 实验软件包
- 课程设计内容
- 实验方案的实施建议

实验开发环境与工具

- 编译器开发所使用的语言：Java
  - Java SDK 1.5 (J2SE 5.0)以上
  - Eclipse IDE
- Eclipse JDT中的AST，简称Eclipse AST
- 分析器的生成工具
  - 词法分析器的生成工具：JFlex
  - LALR分析器的生成工具：CUP
  - LL(k)分析器的生成工具：Java CC
- 目标机
  - x86汇编代码：PC机－GCC（MinGW），GDB
  - MIPS汇编代码：PC机－SPIM模拟器，龙芯机－GCC
- 编译工具：ant

Eclipse AST需要的类库文件

Eclipse 版本号	相关的 jar 文件
3.1.2	org.eclipse.core.resources_3.1.2.jar org.eclipse.core.runtime_3.1.2.jar org.eclipse.jdt.core_3.1.2.jar org.eclipse.jdt.ui_3.1.2.jar
3.2	org.eclipse.core.resources_3.2.2.jar org.eclipse.core.runtime_3.2.0.jar org.eclipse.equinox.common_3.2.0.jar org.eclipse.jdt.core_3.2.3.jar org.eclipse.jdt.ui_3.2.2.jar
3.3	org.eclipse.core.resources_3.3.0.jar org.eclipse.core.runtime_3.3.100.jar org.eclipse.equinox.common_3.3.0.jar org.eclipse.jdt.core_3.3.1.jar org.eclipse.jdt.ui_3.3.1.jar

实验环境的软件配置示例

软件名称	配置一（Windows 下）	配置二（Linux 下）
操作系统	Windows XP SP2	Ubuntu 8.04 (2.6.24-16-generic)
JDK	JavaSE 6 update 6	JavaSE 6 update 6
Eclipse	Eclipse 3.3.2	Eclipse 3.3.2
Ant	Ant 1.7.0	Ant 1.7.0
GCC	MinGW gcc 4.3.0 alpha MinGW gcc 3.4.5	gcc 4.2.3 (默认) gcc 3.4.6
SPIM	PCSpim 7.2.1	SPIM 7.3-1

实验软件配置

- 安装jdk、ant、eclipse、gcc(MinGW)、SPIM
  - 假设ant安装在D:\IDE\apache-ant-1.6.5
  - 假设jdk安装在D:\IDE\Java\jdk1.5.0
- 环境变量
  - 设置ANT_HOME 为D:\IDE\apache-ant-1.6.5
  - 设置JAVA_HOME 为D:\IDE\Java\jdk1.5.0
  - 将jdk、ant、gcc、spim可执行文件的存放路径加到环境变量PATH中

关于环境变量设置参见《编译原理实验教程》1.3.2节

演示1：用ant编译、运行Java程序

- 《编译原理实验教程》1.3.4节　XML与ant简介
- Ant的更多使用参见

  http://ant.apache.org/manual/index.html
- 示例代码：配套光盘test/SkipOOMiniJOOL/java
  - bin:
    build.xml用于编译、运行Java程序，
    compile.bat编译Java程序
  - src：一些Java源程序

在命令控制台下进入bin目录，执行

ant build

将创建classes目录，并将编译得到的class文件置于其中

执行　　ant

运行指定的Java程序

实验方案的综述

- 实验方案设计的指导思想
- 实验方案的主线与组成要素
- 实验开发环境与工具
- 实验软件包
- 课程设计内容
- 实验方案的实施建议

实验支持库

以类库文件的形式提供:

- 实验运行平台：提供平台接口以及编译器运行控制
- 以三地址码为基础的低级中间表示LIR及优化库
- 汇编代码的内部表示AIR
- 可用的编译器组件
- 实用工具：AST的图形化显示、中间表示访问类、编译器错误管理包
- LIR上的寄存器分配器

课程设计开发包(1)

目录	描述
lab1	有关 SimpleMiniJOOL 语言解释器的课程设计，参见第 2 章
lab2	有关 MiniJOOL 语言词法分析的课程设计，参见第 3 章
lab3	有关 SkipOOMiniJOOL 语言语法分析的课程设计，参见第 4 章
lab4	有关 SkipOOMiniJOOL 语言语义分析的课程设计，参见第 5 章
lab5	有关 SkipOOMiniJOOL 语言的 AST 到 LIR（或其他低级中间表示）转换的课程设计，参见第 6 章
lab6	有关 SkipOOMiniJOOL 语言的 x86 汇编代码生成的课程设计，参见第 8 章
lab7	有关 SkipOOMiniJOOL 语言的 MIPS 汇编代码生成的课程设计，参见第 8 章
lab8	有关 MiniJOOL 语言面向对象特征的编译，参见第 9 章

课程设计开发包(2)

目录	描述
lab1	有关 SimpleMiniJOOL 语言解释器的课程设计，参见第 2 章
lab2	有关 MiniJOOL 语言词法分析的课程设计，参见第 3 章
lab3	有关 SkipOOMiniJOOL 语言语法分析的课程设计，参见第 4 章
lab4	有关 SkipOOMiniJOOL 语言语义分析的课程设计，参见第 5 章
lab5	有关 SkipOOMiniJOOL 语言的 AST 到 LIR（或其他低级中间表示）转换的课程设计，参见第 6 章
lab6	有关 SkipOOMiniJOOL 语言的 x86 汇编代码生成的课程设计，参见第 8 章
lab7	有关 SkipOOMiniJOOL 语言的 MIPS 汇编代码生成的课程设计，参见第 8 章
lab8	有关 MiniJOOL 语言面向对象特征的编译，参见第 9 章

基础级实验

课程设计开发包(3)

目录	描述
lab1	有关 SimpleMiniJOOL 语言解释器的课程设计，参见第 2 章
lab2	有关 MiniJOOL 语言词法分析的课程设计，参见第 3 章
lab3	有关 SkipOOMiniJOOL 语言语法分析的课程设计，参见第 4 章
lab4	有关 SkipOOMiniJOOL 语言语义分析的课程设计，参见第 5 章
lab5	有关 SkipOOMiniJOOL 语言的 AST 到 LIR（或其他低级中间表示）转换的课程设计，参见第 6 章
lab6	有关 SkipOOMiniJOOL 语言的 x86 汇编代码生成的课程设计，参见第 8 章
lab7	有关 SkipOOMiniJOOL 语言的 MIPS 汇编代码生成的课程设计，参见第 8 章
lab8	有关 MiniJOOL 语言面向对象特征的编译，参见第 9 章

课程设计开发包(4)

学生自主选择、开发和推销某类编译器组件，并自主选择使用其他学生的组件进行装配，从而得到一个完整的编译器。参见第10章

综合级实验

独立开发局部、合作开发整体

自主推销、整体评测

课程设计开发包(5)

- 课程设计开发包各子目录的内容

  为课程设计提供框架程序以及支持快速编译运行程序的批命令等。

  - bin目录：存放ant编译文件和批处理文件等
  - config目录：存放各种配置文件
  - lib目录：存放课程设计要用到的类库文件，仅lab2和lab4目录包含该子目录
  - src目录：存放Java源程序
  - test目录：存放测试程序

配套光盘的内容

- README：光盘的说明文件
- manual.pdf：实验软件包的简要用户手册
- student目录：存放学生开展实验所需的实验支持库和课程设计开发包等
  - lab目录：存放各章课程设计的开发包
  - lib目录：存放各个实验支持库
  - config目录：存放一些配置文件、词法和文法规范描述文件
  - src目录：存放一些自动生成出来的Java源程序
  - submit目录：可作为存放各个学生实验提交内容的根目录
  - tools目录：存放JFlex、CUP、JavaCC的类库文件
- doc目录：存放实验支持库等的应用编程接口文档
- test目录：存放各种测试程序

实验支持库的构成(lib目录)

相关文件/目录	描述
AST 目录	其下按 Eclipse 版本号分子目录存放与 Eclipse AST 有关的类库文件。AST 节点的抽象基类是 org.eclipse.jdt.core.dom.ASTNode
platform.jar	实验运行平台，提供编译器组件及中间表示访问接口，它根据平台配置文件中的设置将若干编译器组件连接装配在一起并控制它们的运行。主类是 edu.ustc.cs.compile.platform.Main，详细介绍参见 2.3 节
compiler.jar	编译器组件库，包括可用的 MiniJOOL 语言及其子语言的分析器、AST 到 LIR 的转换器等组件，详细列表见附录 10
util.jar	实用工具库，包括 AST 的图形化显示包(参见 2.5 节)、中间表示访问类(参见 2.3.1 节)、编译器错误管理包(参见 4.6.1 节)等实用工具
lir.jar	提供基于三地址代码的低级中间表示。一个程序对应的低级中间表示为类 edu.ustc.cs.compile.lir.LIR 的实例。具体参见第 6 章
air.jar airCode.jar genAirCode.bat genAirCode.sh	提供汇编代码的内部表示。与具体汇编语言无关的部分打包在 air.jar，而 airCode.jar 是利用 genAirCode.bat 或 genAirCode.sh 根据存储在光盘/student/config/air 目录下的汇编语言特征配置文件 x86.xml 和 mips.xml 生成的、与具体汇编语言相关的部分。参见第 7 章
regalloc.jar	提供在 LIR 上的寄存器分配器，即为传入的 LIR 中的伪寄存器分配相应的可用硬寄存器编号（或相对编号），参见 8.5 节
MIPSGenerator.jar X86Generator.jar	分别提供 LIR 到 MIPS 汇编以及 LIR 到 x86 汇编的简单汇编代码生成器，生成器类分别为 edu.ustc.cs.compile.gen.mips.Generator 和 edu.ustc.cs.compile.gen.x86.Generator，均实现接口 edu.ustc.cs.compile.platform.interfaces.GeneratorInterface

实验软件包涉及的配置文件（config目录）

相关文件	描述
JFlex/java.flex	Java 语言的 JFlex 词法规范描述文件
CUP/java15.cup	Java 语言的 CUP 文法规范描述文件
JJ/java15.jj	Java 语言的 JJ 文法规范描述文件
configure.xsd configure.xml	实验运行平台配置，采用 XML 格式，configure.xsd 是这类文件的 XML Schema 定义。configure.xml 是一个样例，在各章课程设计开发包的 config 目录下有用于该课程设计的平台配置文件
error_def.xsd error_def.xml	编译器错误类型配置，采用 XML 格式，error_def.xsd 是这类文件的 XML Schema 定义。error_def.xml 是一个样例，光盘 /student/lib/util.jar 中的 edu.ustc.cs.compile.util.error.ErrorFactory 类将根据传入的错误类型配置来创建和维护这些类型的错误实例，具体参见 4.6.1 节
在 air 子目录下 air_config.xsd x86.xml mips.xml	汇编语言特征配置，采用 XML 格式，air_config.xsd 是这类文件的 XML Schema 定义。x86.xml 和 mips.xml 分别是本书所采用的 x86 和 MIPS 汇编语言的特征配置。在存放实验支持库的 lib 目录下，修改并运行 genAirCode 批处理脚本，将由传入的汇编语言特征配置文件，如 x86.xml，生成 Director.java、OpCode.java、RegName.java，这些文件提供所能处理的各种伪指令码、指令的操作码以及寄存器名对应的符号常量，它们编译后打包成类库文件 airCode.jar。

学生实验提交物目录结构组织示例

在submit目录中给出一个示例，即demo子目录

- bin目录：存放ant编译文件和批处理文件等
- config目录：存放各种配置文件
- lib目录：存放自己开发的以及所使用的由其他学生开发的编译器组件的类库文件，类库文件的命名方式可以是“学号_组件名.jar”
- doc目录：存放接口说明文档、设计实现文档以及进度计划与执行情况说明文档
- src目录：存放自己的Java源程序
- test目录：存放测试程序

实验方案的综述

- 实验方案设计的指导思想
- 实验方案的主线与组成要素
- 实验开发环境与工具
- 实验软件包
- 课程设计内容
- 实验方案的实施建议

《编译原理实验教程》书目

1. 概述- Eclipse，Ant，XML
2. 一个简单的程序解释器-lab1,Platform,AST,Visitor
3. 词法分析-lab2，JFlex
4. 语法分析-lab3，CUP，JavaCC，Util错误管理
5. 语义分析-lab4，符号表
6. 中间表示的转换-lab5，LIR
7. 汇编语言及汇编代码的内部表示-AIR
8. 汇编代码的生成-lab6(x86)，lab7(MIPS)
9. 面向对象语言的编译-lab8 MiniJOOL-I,II,III,MiniJOOL
10. 综合性课程设计- 自行建立工程

与理论课程教学同步的若干个循序渐进的课程设计

可独立开课的综合性课程设计

要实现的源语言的描述

2.1 SimpleMiniJOOL语言-非形式描述

附录4 SimpleMiniJOOL语言语法的EBNF表示-二义的

3.2 MiniJOOL语言的词法-非形式描述

附录1 MiniJOOL语言的词法记号类型及标识

4.1 SkipOOMiniJOOL语言的语法-非形式描述

5.1 SkipOOMiniJOOL语言的静态语义-非形式/形式描述

附录5 SkipOOMiniJOOL语言语法的EBNF表示-二义的

9.1 MiniJOOL语言特征-非形式描述

- -MiniJOOL-I 类、对象、单一继承，无重载、重写/隐藏
- -MiniJOOL-II 类的析构器及delete语句
- -MiniJOOL-III 域隐藏，方法的重载、重写/隐藏
- -MiniJOOL 多态

附录6 MiniJOOL语言语法的EBNF表示-二义的

示例使用的源语言及其描述

3.4.1 Block语言的词法-非形式描述

4.3.1 SimpleBlock语言-非形式描述

4.4.3 示例2：Block语言的语法分析器

1．Block语言的语法-非形式描述

5.4.1 Block语言的语义特征-非形式描述

循序渐进的课程设计(1)

- **Ch2 一个简单的程序解释器－入门级实验**
  - 课程设计1　一个简单的程序解释器　SimpleMiniJOOL
- **Ch3 词法分析**
  - 课程设计2-1　用JFlex为MiniJOOL语言构造一个词法分析器
    示例　识别由英文字母组成的单词和由数字组成的整数
  - 课程设计2-2　手工编写一个简单的词法分析器
    示例　识别十进制整数和加号
  - 课程设计2-3　编写一个非确定有限自动机(NFA)的生成器
  - 课程设计2-4　编写一个词法分析器的生成器

循序渐进的课程设计(2)

- Ch4 语法分析
  - 课程设计3-1　手工编写一个分析合法SimpleBlock程序的语法分析器
  - 课程设计3-2　用CUP生成一个能分析合法程序的语法分析器
    示例1　SimpleBlock语言的语法分析器
    示例2　Block语言的语法分析器
  - 课程设计3-3　用JavaCC生成一个语法分析器
    示例　Block语言及其子语言的分析器
  - 课程设计3-4　用CUP生成一个有错误处理能力的语法分析器
    示例　SimpleBlock语言的有错误处理能力的分析器
  - 课程设计3-5　用JavaCC生成一个有错误处理能力的语法分析器
    示例　SimpleBlock语言的有错误处理能力的分析器

循序渐进的课程设计(3)

- Ch5 语义分析
  - 课程设计4-1　为源程序对应的AST构造符号表
    示例　为Block程序对应的AST构造符号表
  - 课程设计4-2　利用AST及其符号表信息开展语义检查
    示例　利用Block程序对应的AST及其符号表信息开展语义检查
  - 课程设计4-3　对源程序关联的AST进行语义检查
    示例　对Block程序对应的AST进行语义检查
  - 课程设计4-4　在语法分析的同时构造符号表
    示例　带符号表构造的Block语言分析器
  - 课程设计4-5　在语法分析的同时开展语义检查
    示例　带语义检查的Block语言分析器

循序渐进的课程设计(4)

- Ch6 中间表示的转换
  - 课程设计5-1　SkipOOMiniJOOL语言的AST到LIR的转换器
    - 类型信息的收集
    - 全局变量的声明和访问
    - 局部变量的声明与访问
    - 方法声明
    - 参数传递
    - 返回值的传递
    - 其他语句和表达式的处理
  - 课程设计5-2　自行设计和实现一种中间表示
  - 课程设计5-3　AST到自行设计的中间表示的转换

循序渐进的课程设计(5)

- Ch8 汇编代码生成
  - 课程设计6-1　利用现有的寄存器分配器由LIR生成x86汇编代码
  - 课程设计7-1　利用现有的寄存器分配器由LIR生成MIPS汇编代码
  - 课程设计6-2　实现寄存器分配并应用于LIR到x86汇编代码的生成器中
  - 课程设计7-2　实现寄存器分配并应用于LIR到MIPS汇编代码的生成器中
  - 课程设计6-3　AST到x86汇编代码的生成
  - 课程设计7-3　AST到MIPS汇编代码的生成

提高级课程设计

❖ Ch9 面向对象语言的编译

- 课程设计8-1　构造MiniJOOL语言的词法语法分析器
- 课程设计8-2　构造MiniJOOL语言的语义检查器
- 课程设计8-3　构造MiniJOOL-I语言的AST到LIR的转换器
  类变量的创建和初始化、实例变量的创建和初始化以及访问方法的定义和调用
- 课程设计8-4　构造MiniJOOL-II语言的AST到LIR的转换器
  对析构器声明的处理、delete语句的处理
- 课程设计8-5　构造MiniJOOL-III语言的AST到LIR的转换器
  方法的重载、域隐藏、方法隐藏与方法重写
- 课程设计8-6　构造MiniJOOL语言的AST到LIR的转换器
  虚方法表的建立、动态绑定、运行时类型判断
- 课程设计8-7　构造MiniJOOL语言的LIR到汇编代码的生成器
  类变量和方法的签名、虚方法表的表示

综合级课程设计(1)

- 选做组件
  - 选做组件1　源语言程序的分析器
  - 选做组件2　基于AST的解释器
  - 选做组件3　基于LIR的解释器
  - 选做组件4　AST到LIR的转换器
  - 选做组件5　基于LIR的x86汇编代码生成器
  - 选做组件6　基于LIR的MIPS汇编代码生成器
  - 选做组件7　基于AST的x86汇编代码生成器
  - 选做组件8　基于AST的MIPS汇编代码生成器

综合级课程设计(2)

- 各编译器组件的组合方式
  - 组合方式1　选做组件1 + 选做组件2
  - 组合方式2　选做组件1 + 选做组件7 + GCC
  - 组合方式3　选做组件1 + 选做组件8 + SPIM
  - 组合方式4　选做组件1 + 选做组件4 + 选做组件3
  - 组合方式5　选做组件1 + 选做组件4 + 选做组件5 + GCC
  - 组合方式6　选做组件1 + 选做组件4 + 选做组件6 + SPIM

综合级课程设计(3)

❖课程设计实施方式

- 第1~2周，教师根据本校实际情况明确课程实验目标，制订具体的综合性课程设计内容。
- 第3周，教师布置综合性课程设计内容以及时间节点。
- 第3~5周，教师指导和要求学生开展本书第2章的课程设计，旨在熟悉实验环境、AST及其访问者模式。
- 第6~7周，答疑并介绍源语言的词法、语法和语义特点，以及可利用的相关编译器组件和其他工具包。
- 第6~10周，学生自行消化理解，确定拟实现的编译器组件，并进行初步设计，将问题反馈给教师。
- 第11周左右，学生提交选做的组件以及初步设计，教师公布学生选做组件的分布情况(注意平衡性)并答疑。

综合级课程设计(4)

❖课程设计实施方式

- 第11~12周左右，学生细化组件设计，并反馈问题。
- 第13周左右，教师根据学生提问，集中答疑与指导。
- 第13~15周左右，学生独立开发编译器组件。
- 第14周左右，教师发布测试程序以及整体评测环境。
- 第15周左右，学生课程设计的第二次提交，主要提交设计文档、已开发的源代码和库等。
- 第15~17周左右，学生自行推销和选择组件，进行编译器的整体测试和修改，其间，教师可以让学生每周提交一次当前的实验结果（包括设计文档、已开发的源代码、库、测试程序等），旨在督促学生，了解学生实验状况，并根据实际决定是否安排答疑和指导。

综合级课程设计(5)

❖课程设计实施方式

- 第18周，教师规定最后的提交截止时间以及提交内容，确定考评方案并通知。
- 学生按要求在规定的截止时间之前提交最后版本。
- 学生在规定的时间和地点参加课程设计答辩与考评。

- 教师需要事先准备好网络提交环境，制订提交细则，然后通知学生。
- 学生需要按教师的规定来组织每次提交的提交物。
- 学生必须在提交的根目录下安放一个readme.txt或者是readme.doc，说明当前的进展状况。

综合级课程设计(6)

❖考评方法

- 学生分成若干组，每组约10人
- 各组用一个上午、下午或晚上的时间进行现场测试、答辩和公开评分
- 评委：教师、研究生助教、同组的所有同学
- 教师主导测试过程，学生自己动手按老师要求操作，并用投影机当众显示测试过程
- 老师和同学均可以提问，学生需当众回答
- 所提问题主要围绕完成的设计和编程，以及测试中暴露出的设计或编程错误等展开

综合级课程设计(7)

- 评分依据
  - 工程的规范性
  - 编译器的正确性
  - 错误定位与恢复能力
  - 所生成的目标代码质量
  - 回答问题时所表现出的对本课程设计所涉及知识的掌握程度
  - 对自己设计和编码的前端（后端）的熟悉程度
  - 操作的熟练程度
  - 提交物的完整性、条理性及其中反映的分析和设计的思想

综合级课程设计(8)

❖评分方法

- 每个评委当场给该组的全部同学排名次
- 由助教根据所有有效排名表，给出最终的排名
- 由老师根据本组的情况确定本组的最高分和最低分，并依据排名，主要按等间隔确定每个同学的分数

综合级课程设计(9)

❖ 其他评分细则

- 如被老师、助教和过半数同学认为所提交的不是自己的课程设计成果时，为0分;
- 未按时提交也是0分;
- 独自完成整个编译器，分组评定成绩后降10分;
- 当前后端人数比例严重失调，则抬高少数人一端分数
- 若所开发的前端（或后端）被多个同学（开发的合作伙伴除外）采用，则在分组评分的基础上加分，加分原则是:
  - 每增加两个采用者加1分
  - 课程设计和平时作业合计不超过50分

实验方案的综述

- 实验方案设计的指导思想
- 实验方案的主线与组成要素
- 实验开发环境与工具
- 实验软件包
- 课程设计内容
- 实验方案的实施建议

测试环境

- 在课程实验初期规定评测时所使用的软件版本
  →避免因软件版本不一致而出现学生提交的软件包无法支持工作，或因不熟悉评测环境而影响评测的进度和效果
- 事先明确测试环境的工作目录必须按配套光盘中的student目录结构进行组织
  →减少学生在评测时修改各种路径设置的工作量
- 评测时统一使用配套光盘student目录中的lib和tools子目录下的相关类库
  →避免因学生使用的类库不存在或与评测提供的类库相冲突
- 学生依据相对路径来使用各种类库或测试文件

课程设计提交要求

- 尽早明确学生的课程设计提交内容，学生须按规定的目录结构组织提交内容，然后将它们打包成一个文件进行电子提交
  →培养学生的工程质量意识
- 教师要反复督促和检查，才能让学生真正养成按规范办事的习惯
  →实践证明，多数学生需要被督促
- 其他的一些提交要求
  - 不得提交生成的class文件
  - 限制提交文件的大小，不要包含配套光盘中提供的类库等
  - 源程序中应该加有注释
  - 学会用jar命令只把自己编写的组件的相关类文件打包成类库文件，放到提交目录的lib子目录下
  - ……

过程管理与控制

- 督促学生尽早动手实践：边阅读、边实践、边思考、边消化理解
- 教师要结合所在学校以及学生的实际情况，尽早选择若干课程设计布置给学生
- 教师在布置实验任务时要尽可能详尽，学生也要及时反馈对任务描述的疑问
- 制定多时间节点和多次提交的过程管理与控制机制
- 对学生每次提交的内容要及时检查，并及时通报检查结果。对于无进展或进展不好的同学，要给以警示；对于进展优秀的同学，要给以表扬
- 建立日常的多种答疑和解惑机制，如email、bbs、集中讲解、公布定期的答疑时间和地点等

提纲

- 国内外编译原理实验简述
- 实验方案综述
- 部分实验支持库的设计思路
- 实验方案的特点
- 实验方案的实践及经验教训

部分实验支持库的设计思路

- 实验运行平台
- 中间表示的图形化输出
- 编译器的错误管理
- 低级中间表示
- 汇编代码的内部表示
- 寄存器分配器

实验运行平台的组成和目的

- 实验平台接口：提供各类编译器组件及组件间信息访问的接口
- 实验运行平台：能将若干个编译器组件连接装配成一个完整的编译器或解释器或它们的前端，并控制它们的执行
  - 平台配置文件或命令行参数
  - 根据配置加载、运行各编译器组件

实验平台接口

接口类名	描　　述
InterRepresent	中间表示访问接口，作为访问高级中间表示(如 AST)或低级中间表示信息的公共接口
SymTable	符号表接口
ParserInterface	分析器接口
ParserException	分析中导致编译器无法继续正确执行时抛出此类异常
InterpreterInterface	解释器接口
InterpreterException	解释执行中导致编译器无法继续正确执行时抛出此类异常
CheckerInterface	语义检查器接口
CheckerException	语义检查中导致编译器无法继续正确执行时抛出此类异常
TransformerInterface	变换器接口
TransformerException	变换器工作中导致编译器无法继续正确执行时抛出此类异常
GeneratorInterface	代码生成器接口
GeneratorException	代码生成中导致编译器无法继续正确执行时抛出此类异常

中间表示访问接口

- 接口定义：public interface InterRepresent
- 需要实现的方法

```
public Object getIR( )
public void setIR(Object ir)
public void showIR( )
public SymTable getSymTable()
public void setSymTable(SymTable symTable)
```

- 说明：旨在封装某种中间表示的实例及符号表实例，对外提供统一的访问接口
- 两种实现类
  - 高级中间表示访问类**edu.ustc.cs.compile.util.ir.HIRPack**
    以Eclipse AST为中间表示，无符号表信息
  - 低级中间表示访问类**edu.ustc.cs.compile.util.ir.LIRPack**
    以实验软件包中的edu.ustc.cs.compile.lir.LIR为中间表示，无符号表信息

分析器接口

- 接口定义：public interface ParserInterface
- 需要实现的方法

  public InterRepresent doParse(File src) throws ParserException

- 说明：
  - src：待分析的源程序文件
  - 返回：一个中间表示访问接口实例

  若分析器在分析中发现源程序有错，并且该错误影响编译器后续部分的正确运行时，就需要抛出ParserException异常。实验运行平台捕获到这个异常后，会终止运行。
- 适用于本接口的编译器组件：

  AST构造器、语法分析器、带语义检查的语法分析器等

解释器接口

- 接口定义：public interface InterpreterInterface
- 需要实现的方法

  public void interpret(InterRepresent ir) throws InterpreterException

- 说明：
  - ir：待解释执行的源程序对应的中间表示访问实例

  若解释器在执行时发现程序有错，并且该错误影响编译器后续部分的正确运行时，就需要抛出InterpreterException异常。实验运行平台捕获到这个异常后，会终止运行。
- 适用于本接口的编译器组件：

  AST解释器、LIR解释器、其他中间表示上的解释器等

检查器接口

- 接口定义：public interface CheckerInterface
- 需要实现的方法

  public boolean check(InterRepresent ir) throws CheckerException

- 说明：
  - ir：待检查的源程序的中间表示访问实例
  - 返回：若检查器认为程序语义正确，则返回**true**，否则为**false**

  若检查器在执行时发现程序有错，并且该错误影响编译器后续部分的正确运行时，就需要抛出CheckerException异常。实验运行平台捕获到这个异常后，会终止运行。
- 适用于本接口的编译器组件：

  AST的语义检查器、其他中间表示上的检查器等

变换器接口

- 接口定义：public interface TransformerInterface
- 需要实现的方法

  public InterRepresent transform(InterRepresent ir)
  throws TransformerException

- 说明：
  - ir：待变换的程序的中间表示访问实例
  - 返回：变换后的中间表示访问实例

  若变换器在执行时发现程序有错，并且该错误影响编译器后续部分的正确运行时，就需要抛出TransformerException异常。实验运行平台捕获到这个异常后，会终止运行。

- 适用于本接口的编译器组件：

  AST到LIR的转换器、AST或LIR上的各种优化变换器、其他中间表示上的优化变换器、中间表示之间的转换器等

代码生成器接口

- 接口定义：public interface GeneratorInterface
- 需要实现的方法

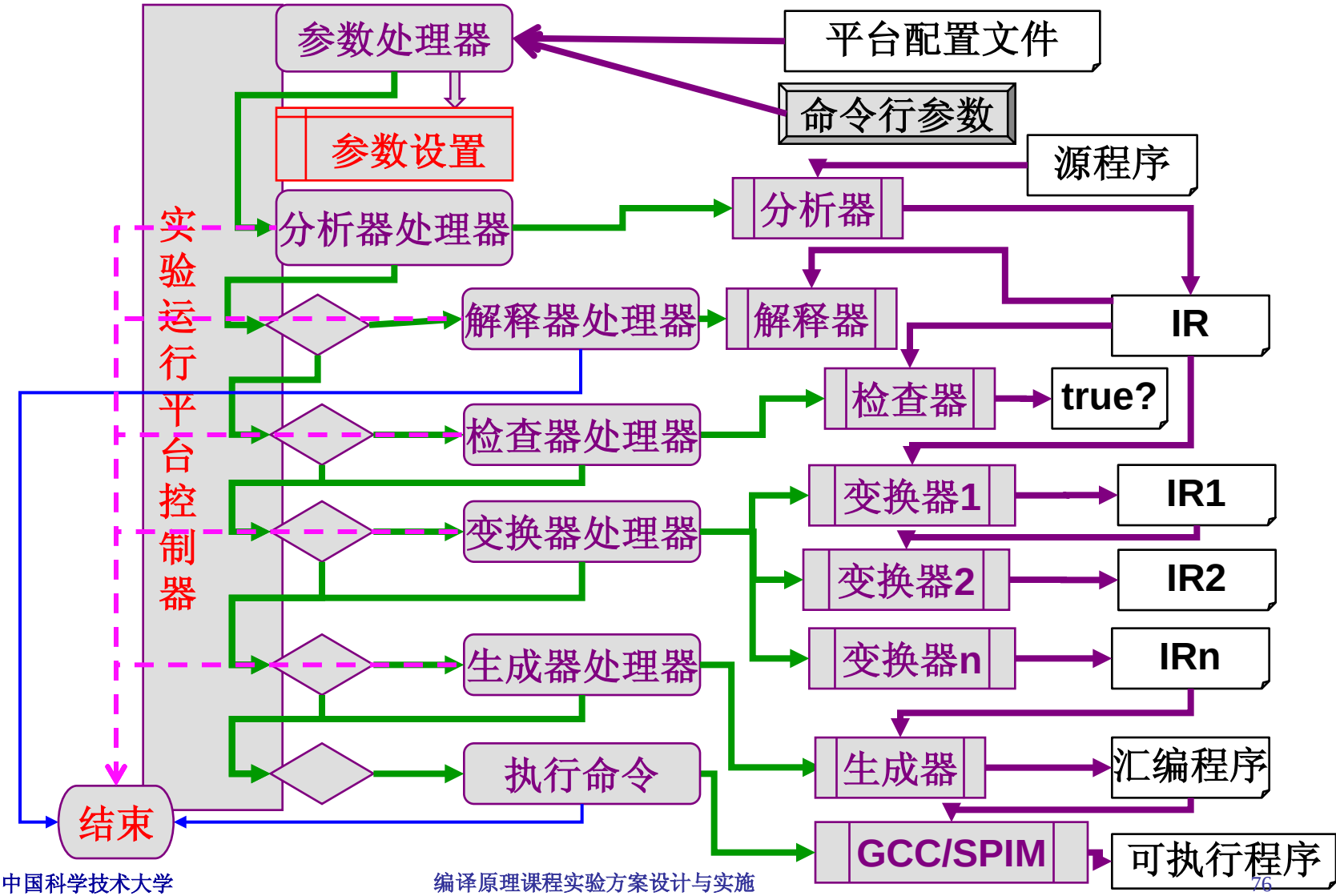
```
public void generate(File cfgFile, File outFile,
        InterRepresent ir) throws GeneratorException
```

- 说明：
  - **cfgFile**：目标语言的特征配置文件
  - **outFile**：生成的目标程序的输出文件
  - **ir**：待进行代码生成的程序的中间表示访问实例

  若生成器在执行时发现程序有错，并且该错误影响编译器后续部分的正确运行时，就需要抛出GeneratorException异常。实验运行平台捕获到这个异常后，会终止运行。
- 适用于本接口的编译器组件：

  由某种中间表示到**x86**或**MIPS**汇编代码的生成器等

实验运行平台的工作机制



实验运行平台的使用

❖使用简介

➢ java命令行

java -classpath <classpath>
edu.ustc.cs.compile.platform.Main [options] [source-file]

➢ 批命令脚本

在课程设计开发包的bin目录下提供 run.bat或run.sh

./run.sh [options] [source-file]　　Linux/Unix 平台

run.bat [options] [source-file]　　Windows平台

<classpath> 指定本次执行所依赖的除JRE（Java Runtime Environment）提供的类库之外的类库文件或class文件的位置，包括实验运行平台类库文件、所涉及的编译器组件的类库文件或class文件的位置等。

[options] 指定实验运行平台的各种参数

[source-file] 指定待编译的源程序文件的位置

实验运行平台的命令行参数(1)

- --help（或-h）：输出帮助信息
- --cfg-file <file>（或-cf <file>）：指定平台配置文件的位置
- --debug（或-d）：打开调试模式
- --parser-class <classname>（或-P <classname>）：指定分析器的全称类名
- --disp-ir=[yes|no](或-di=[yes|no]): 指定是否显示分析器、变换器执行后所得到的中间表示,默认设为no
- --do-interp=[yes|no](或-i=[yes|no]): 指定是否调用解释器，默认设为no
- --interp-class <classname>（或-I <classname>）：指定解释器的全称类名，缺省类名为edu.ustc.cs.compile.interpreter.Interpreter，该选项在--do-interp=yes（或-i=yes）时生效
- --do-check=[yes|no]（或-c=[yes|no]）：指定是否调用检查器。默认设为no

实验运行平台的命令行参数(2)

- --checker-class <classname>（或-C <classname>）：指定检查器的全称类名，默认类名为edu.ustc.cs.compile.checker.skipoominijool.Checker该选项在--do-check=yes（或-c=yes）时生效
- --do-trans =[yes|no]（或-t=[yes|no]）：指定是否调用变换器。默认设为no
- --trans-classes <classname list>(或-T <classname list>):指定待执行的一组变换器的全称类名，各个类名之间以冒号来分隔。该参数在--do-trans =yes（或-t=yes）时生效，默认为edu.ustc.cs.compile.ast2lir.skipoominijool.AST2LIR
- --do-gen=[yes|no]（或-g=[yes|no]）：指定是否调用代码生成器，默认设为yes
- --gen-class <classname>（或-G <classname>）：指定你的代码生成器的全称类名，默认类名为edu.ustc.cs.compile.generator.Generator。该选项在--do-gen=yes（或-g=yes）时生效

实验运行平台的命令行参数(3)

- --asmcfg-file <file>（或-af <file>）：指定目标语言特征配置文件的位置，默认为空串。该选项在--do-gen=yes（或-g=yes）时生效
- 用--arch=[x86|mips]或-a=[x86|mips]：指定生成的汇编代码是Intel x86架构上的还是MIPS R2000/R3000架构上的。默认为x86
- -S <file>：指定汇编代码文件名file，汇编代码生成器将把生成的汇编代码输出到该文件中。<file>默认为a.s。该选项在--do-gen=yes（或-g=yes）时生效
- --exec=[yes|no]（或-e=[yes|no]）：指定是否运行生成的汇编代码。如果生成的是x86架构上的汇编代码，实验运行平台会先调用gcc将汇编代码编译成可执行文件，然后再执行。如果生成的是MIPS架构上的汇编代码，实验运行平台会调用spim执行这些汇编代码。该选项默认设为yes，并且在 --do-gen =yes（或-g=yes）时生效

实验运行平台的命令行参数(4)

- --gcc-path <path>或-gcc <path>：指定你的系统中gcc的绝对路径。该选项默认为/usr/bin/gcc，并且在 --do-gen=yes（或-g=yes）且 --arch=x86（或-a=x86）时生效
- -o <file>：指定可执行文件名file，实验运行平台将调用gcc对生成的汇编代码编译链接得到可执行代码，并将其输出到所指定的文件中。可执行文件名默认设为a.out。该选项在 --do-gen=yes（或-g=yes）且 --arch=x86（或-a=x86）时生效
- --spim-path <path>或-spim <path>：指定你的系统中spim的绝对路径。该选项默认为/usr/local/bin/spim，它在--do-gen=yes（或-g=yes）且--arch= mips（或-a= mips）时生效

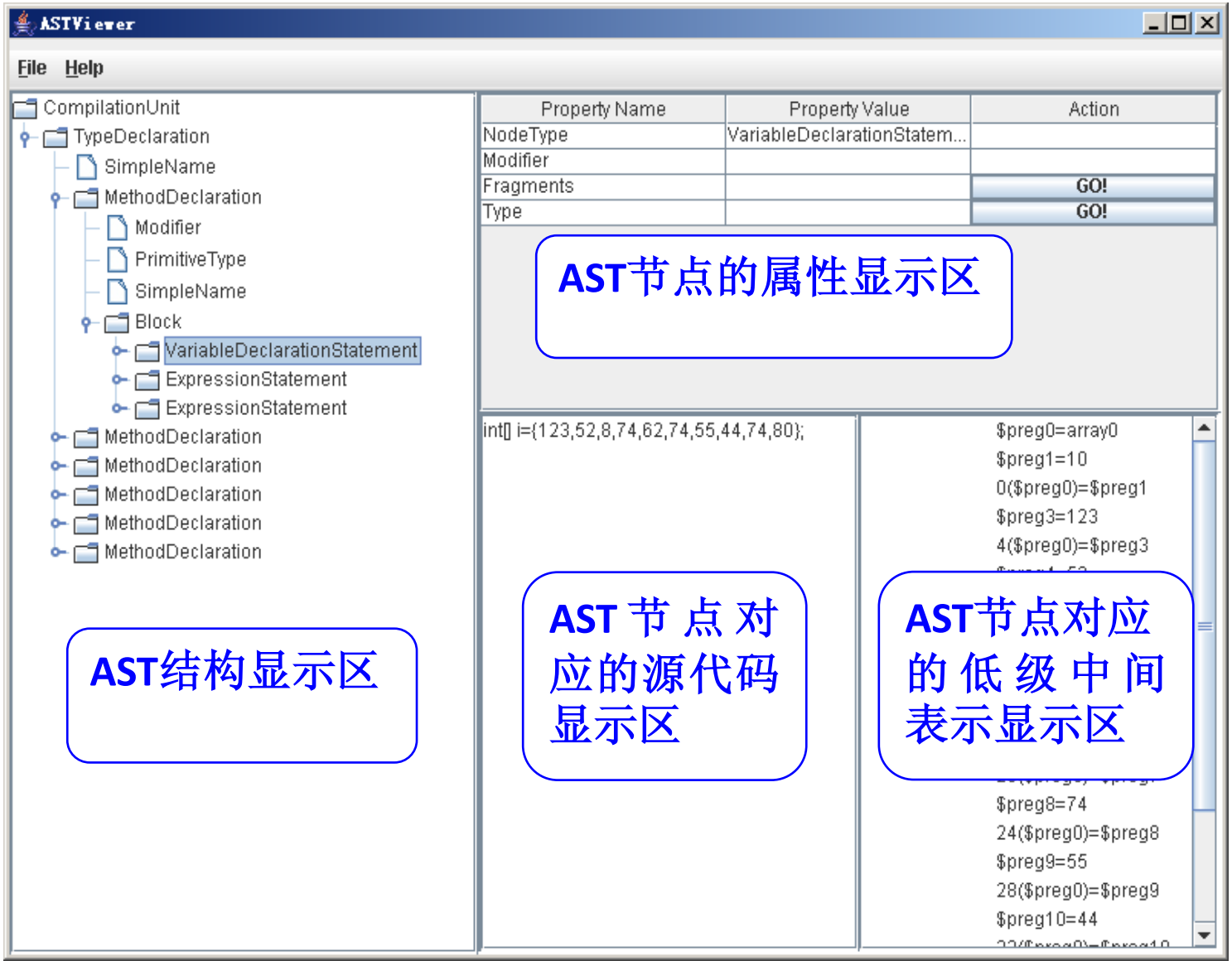
实验平台配置文件

```
1  <?xml version="1.0"?>
2
3  <configs xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:noNamespaceSchemaLocation="configure.xsd">
5    <boolCfgs>
6      <boolCfg name="debug" value="true"/>
7      <boolCfg name="dispIR" value="true"/>
8      <boolCfg name="doInterp" value="true"/>
9      <boolCfg name="doGen" value="false"/>
10     <boolCfg name="exec" value="false"/>
11   </boolCfgs>
12
13   <strCfgs>
14     <strCfg name="srcFile" value="../test/syntax.mj"/>
15     <strCfg name="parserClass"
16         value="edu.ustc.cs.compile.parser.simpleminijool.Parser"/>
17     <strCfg name="interpClass"
18         value="edu.ustc.cs.compile.interpreter.Interpreter"/>
19   </strCfgs>
20 </configs>
```

部分实验支持库的设计思路

- 实验运行平台
- 中间表示的图形化输出
- 编译器的错误管理
- 低级中间表示
- 汇编代码的内部表示
- 寄存器分配器

中间表示的图形化显示


ASTViewer
File Help
CompilationUnit
TypeDeclaration
SimpleName
MethodDeclaration
Modifier
PrimitiveType
SimpleName
Block
VariableDeclarationStatement
ExpressionStatement
ExpressionStatement
MethodDeclaration
MethodDeclaration
MethodDeclaration
MethodDeclaration
MethodDeclaration
Property Name
Property Value
Action
NodeType
VariableDeclarationStatem...
Modifier
Fragments
GO!
Type
GO!
AST节点的属性显示区
int[] i={123,52,8,74,62,74,55,44,74,80};
$preg0=array0
$preg1=10
0($preg0)=$preg1
$preg3=123
4($preg0)=$preg3
$preg8=74
24($preg0)=$preg8
$preg9=55
28($preg0)=$preg9
$preg10=44
AST结构显示区
AST节点对应的源代码显示区
AST节点对应的低级中间表示显示区

Eclips AST简介

参见《编译原理实验教程》2.4

Eclipse帮助页面：窗口左边的目录“JDT Plug-in Developer Guide” → “Reference” → “API Reference” → “org.eclipse.jdt.core.dom”

❖相关的类

- ASTNode类及其派生类：描述各种AST节点的类，每个AST节点表示Java源程序中的一个语法结构。
- AST类：创建AST节点的工厂类，类中包含许多创建各类AST节点的工厂方法。
- ASTVisitor类：AST的访问者抽象类，类中声明了一组访问各类AST节点的visit( )方法、endVisit( )方法、preVisit( )方法和postVisit( )方法。

演示2：Eclipse,实验运行平台和AST

- 《编译原理实验教程》1.3.3、2.3～2.7
- 示例代码：配套光盘student/lab/lab1
  - Eclipse的使用
    - 按 1.3.3 节的3(5)建立工程
    - 修改 src/edu/ustc/cs/compile/interpreter/Main.java,注释57～65行
    - 读src/edu/ustc/cs/compile/interpreter/TestCase.java 尝试自己写一个创建AST的方法
  - 在命令控制台的使用
    - 修改config/lab1-*.xml，将doInterp的值改为false
    - 在命令控制台下进入bin目录，执行

      run ..\config\lab1-testcase.xml

      或 run ..\config\lab1-parser.xml

AST的图形化显示

- 软件包
  - 在edu.ustc.cs.compile.util.ASTView包中
    - core.ASTViewer　AST的图形化显示器
    - core.ASTViewPropertyItem　一条输出属性
    - core.ASTViewPropertyDump 输出属性的定制基类
    - plugin.GenericPropertyDump 默认的输出属性定制类
- 使用

```
ASTViewer astviewer = new ASTViewer(root, null);
  // 创建ASTViewer实例，root为AST的根，null表示用默认定制类实例处理节点的属性显示
astviewer.show();      // 启动对AST树的图形化显示
```

AST的图形化显示

❖AST节点输出属性的定制

- 自行编写一个类，它由类ASTViewPropertyDump或GenericPropertyDump派生
- 该类中根据需要重写部分或全部的dump( )方法。

  public List dump(T node)

  返回一个元素类型为ASTViewPropertyItem的序列

参见《编译原理实验教程》2.5

部分实验支持库的设计思路

- 实验运行平台
- 中间表示的图形化输出
- 编译器的错误管理
- 低级中间表示
- 汇编代码的内部表示
- 寄存器分配器

错误类型与错误信息管理

《编译原理实验教程》 4.6.1节

- 目的
  - 便于管理各类错误，统一每类错误的错误输出信息
- 组成
  - 错误类型配置文件
    示例：配套光盘student/config/err_def.xml/xsd
  - 应用编程接口
    - 类ErrorFactory
- 演示3：错误信息管理
  - 示例代码：配套光盘
    student/lab/lab3/config/CUP/sblock_err.cup

部分实验支持库的设计思路

- 实验运行平台
- 中间表示的图形化输出
- 编译器的错误管理
- 低级中间表示
- 汇编代码的内部表示
- 寄存器分配器

低级中间表示

《编译原理实验教程》 6.2节

- LIR组成 （lir.jar）
  - 三地址代码
  - 为支持数据的存储布局所需保存的类型信息
  - 基本优化信息(用于优化和代码生成的基础结构)
- 演示4: AST到LIR的转换
  - 示例代码: 配套光盘
    student/lab/lab5/bin/demo.bat demo.sh
    student/lab/lab5/config/demo.xml

部分实验支持库的设计思路

- 实验运行平台
- 中间表示的图形化输出
- 编译器的错误管理
- 低级中间表示
- 汇编代码的内部表示
- 寄存器分配器

汇编代码的内部表示

《编译原理实验教程》 7.4节

❖ AIR的组成

- 固定部分：lib/air.jar
- 可变部分
  - 汇编语言特征配置文件
    示例代码：配套光盘student/config/air
  - 符号编码类的代码生成
    示例代码：配套光盘student/lib/genAirCode.bat .sh
    生成出的代码位于：student/src
    对生成出的代码编译打包成 student/lib/airCode.jar
  - 运行时动态加载特征配置

部分实验支持库的设计思路

- 实验运行平台
- 中间表示的图形化输出
- 编译器的错误管理
- 低级中间表示
- 汇编代码的内部表示
- 寄存器分配器

寄存器分配器

《编译原理实验教程》 8.5节

- 算法
  - 线性扫描寄存器分配算法(Linear scan register allocation, TOPLAS99)
- 实验支持库中的实现：regalloc.jar
  - edu.ustc.cs.compile.gen.regalloc.RegAllocator
    - RegAllocator( )
    - RegAllocator(int availRegNum) // 指定可用寄存器数目
    - RegAllocator(int[ ] availRegIDArray) //指定一组可用寄存器编号
    - Integer getRegID(int pseudoRegID)
    - Integer getOffset(int pseudoRegID)

提纲

- 国内外编译原理实验简述
- 实验方案综述
- 部分实验支持库的设计思路
- 实验方案的特点
- 实验方案的实践及经验教训

实验方案的特点(1)

- 规范性
  - 目录结构
  - 编译运行
    - 批处理脚本、配置文件
    - Eclipse IDE工程文件
- 指导性
  - 提供样例及其说明

    如：以SimpleBlock和Block语言示意语法、语义分析的实现
  - 提供一般性的处理规则

    如：语法结构与汇编代码的映射规则
  - 提供框架代码

实验方案的特点(2)

❖整体性

每个课程设计的实验内容着眼于局部点，
但是课程设计的运行完成一个相对完整的功能，例如一个编译器、一个分析器的生成器，等等。

❖灵活性

- 对课程设计的选择，可根据实际情况灵活确定
  - 与课堂教学同步进行：选择各个独立的课程设计
  - 综合性课程实验：前端/后端
- 对编译、调试、运行方法的灵活选择
  - Eclipse工程、命令控制台脚本、实验平台脚本
- 组件实现方法的灵活选择
- 编译器实现方法的灵活选择

实验方案的特点(3)

❖开拓性

实验方案主线以及实验运行平台支持学生开展更多开拓创新型实验，例如，设计实现新的中间表示、与之有关的转换器、优化变换器等。

❖语言的描述

- ➢非形式描述与形式描述相结合
- ➢让学生初步接触语言的形式描述

提纲

- 国内外编译原理实验简述
- 实验方案综述
- 部分实验支持库的设计思路
- 实验方案的特点
- 实验方案的实践及经验教训

实验方案的实践(1)

http://staff.ustc.edu.cn/~yuzhang/compiler

- PB04011：综合性实验，2007年春
  - 以SkipOOMiniJOOL语言作为要实现的源语言
  - 对语言仅有简单的非形式描述，一些语言特征未描述清楚
  - 每个学生独立完成编译器的前端或后端
  - 每个学生自行选择完成后端或前端的合作伙伴

  前端要求：
  - 词法分析、语法分析、语义检查并生成AST

  后端要求：
  - 由AST生成x86汇编代码，不要求代码优化，但要求考虑寄存器分配等问题

  提交要求：
  - 一个完整的编译器……

实验方案的实践(2)

http://staff.ustc.edu.cn/~yuzhang/compiler

- PB05011：综合性实验，2008年春
  - 以SkipOOMiniJOOL语言作为要实现的源语言
  - 对语言有非形式的描述和程序举例
  - 每个学生独立完成编译器的前端或后端
  - 每个学生自行选择完成后端或前端的合作伙伴

  前端要求：
  - 词法分析、语法分析、语义检查并生成AST

  后端要求：
  - 由AST生成x86汇编代码或MIPS汇编代码，不要求代码优化，但要求考虑寄存器分配等问题

  提交要求：
  - 一个完整的编译器……

实验方案的实践(3)

http://staff.ustc.edu.cn/~yuzhang/compiler

- PB06011：综合性实验，2009年春
  - 以SkipOOMiniJOOL语言作为要实现的源语言
  - 对语言有非形式的描述和程序举例、形式描述了类型系统
  - 每个学生独立完成编译器的前端或后端
  - 每个学生自行选择完成后端或前端的合作伙伴
  - 提供了丰富的实验支持库和课程设计开发包

  前端要求：
  - 词法分析、语法分析、语义检查并生成AST或LIR

  后端要求：
  - 由AST或LIR生成x86汇编代码或MIPS汇编代码，不要求代码优化，但要求考虑寄存器分配等问题

  提交要求：
  - 一个完整的编译器……

2009年春季编译实验实践的过程管理

- 提交检查
  - 5月15 提交系统设计书，包括进度表和拟开发的组件
  - 5月22前 经老师认可学生可调整拟开发的组件
  - 5月31 提交已完成源代码和前后端接口描述
  - 6月10 再次提交已完成源代码和进度报告
  - 6月20 提交源文件、类库文件、测试程序、设计文档等
  - 6月22 提交课程实践的收获、体会和建议（发邮件给张昱老师）
- 发布节点
  - 6月13 发布测试程序
  - 6月16 发布测试环境，对最终提交文档的规定

每次提交时需要说明当前的执行进度与计划中的出入与原因，以及对计划的调整（如果有的话）！

实践效果

❖2007春实践反响

- BBS CompilerTech版：07.5.10(978)　07.6(+300)
- 学生1（张昊中）

收益

- 加深了对LALR分析的理解；
- 大概了解了一个编译器（特别是前端）的结构和工作流程
- 积累了一些做工程的经验。

存在问题

- 对实验中要处理的语言的定义不清晰；
- 实验指导不足。

实践效果

- 学生2（李勋浩）

  收益

  - 了解了编译器的基本运行方式，以及经典程序语言运行的基本方式
  - 一些一直以来未能理解的概念和疑问，在自己的摸索中明朗了
  - 在这样相对较为开放的实验环境中，一定程度上激发了我创造的热情。实现中我会不断假想，假如用另一种方法去实现，会怎么样？假如增加某种功能，是否容易？如果不容易，需要加入什么样的内容？

  不足之处

  - 大三下课程较紧，实验周期不长，学生不易在已完成结果上扩展
  - 实验中能够找到的非常相关的阅读材料不足，能找到的也大多是英文的，这对大多数学生来说是个障碍;
  - 抄袭现象较严重，这也是跟本次实验相对其它课程实验来说更难入门有关。

实践效果

❖2008春实践效果

- 学生1（徐奎）

总的来说，我觉得这门实验是非常有意义，有必要，并且应该大力推广下去的，让以后的同学也能有这样一次宝贵的经历。

一、正面的：与其他课程的实验相比，这门实验的难度和所需要投入的时间与精力都要高很多，这是一种新的要求，相对与一些不疼不痒的小实验，更具有挑战性，也能给人更多的锻炼与完成后的成就感。……

二、负面的：1、有些同学借用别人的代码，检查之前突击一下，最后也能通过，甚至得分还不错。相反有些同学自己花时间做了但可能做的不是太好。2、在做的中期过程中，与其他组的同学沟通交流不够，同时与老师和助教的交流也太少，很多时候都是自己在摸索。

实践效果

2008春实践效果

- 学生2（赵增）

一个个人小小的建议：这个实验涉及的内容以及技术实在很多，或许学校能将其作为一门课程设立，至少，可一多给学生上几节课程。大实验拆分成几个阶段性的小实验来提交，可能比较有助与大家熟悉实验。

2009春实践效果

- http://fbbs.ustc.edu.cn/cgi/bbscon?bn=CompilerTech&fn=M4A3E6C8B&num=2490
- http://fbbs.ustc.edu.cn/cgi/bbscon?bn=CompilerTech&fn=M4A538F62&num=2502
- http://fbbs.ustc.edu.cn/cgi/bbsdoc?board=CompilerTech

经验教训

❖ 经验

- 提供程序框架和文档说明：既有挑战性又有好的效果
- 以中间表示作为编译器组件间信息传递的接口：既控制了学生开发的规模又允许有自行设计的空间
- 提供AST图形显示并要求生成汇编码，便于测试和考评
- 合作开发、自主推销和选择、整体评测，既培养了团队精神，又增强了质量意识
- 规定了统一的版本提交截止时间，既有公平性和工程性，又易于评测
- 教师主导的集体公开评分方式，既有公平性又易评测
- 由学生参与评分，既能弥补教师对学生实际情况了解的局限性，又能调动学生的参与热情

经验教训

❖教训及改进之处

- 多种开发工具和环境加宽了学生的技术层面，但导致学生不能集中精力到和编译有关的技术上来。
  前导软件课程实践中逐步熟悉掌握其中的部分工具，提供对这些工具的文档说明和样例。
- 2007,2008对要实现的语言描述不够=>2009形式化语言规范
- 2007,2008对提交环境目录和编写能编译运行编译器的批处理文件等要求发布太迟。
  增加相关的要求与指南，平时注意对学生强调它们。
- 2007只有提交截止时间=>多时间节点和多次提交的过程管理与控制
- 需要配备技术水平高的软件实验师，研究生助教承担不好这样的实验指导

经验教训

❖教训及改进之处

- 增加实验学时或独立开实验课，强化指导和督促。
- 抄袭、拷贝问题

  对SkipOOMiniJOOL的语言特征再分解，确定并定义多种难度级别的语言，让学生进行级别选择；

  AST等中间表示上做限制或扩展
- 进一步完善优化方面的实验
- 丰富可以使用的编译器组件
- 针对学生对类库文件的反编译问题

  字节码的混淆调研与应用

谢 谢 ！