

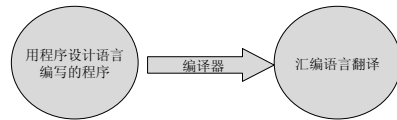
6.035 2005年秋

讲座15: 整理汇总

从语法分析到代码生成

如何让计算机理解?

- 用编程语言编写程序
- 微处理器以汇编语言执行



示例 (输入语言)

```
int expr(int n)
{
    int d;
    d = 4 * n * n * (n + 1) * (n + 1);
    return d;
}
```

示例 (输出汇编代码)

未优化代码

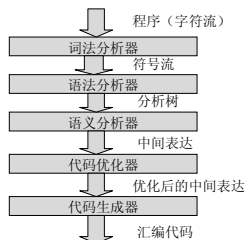
已优化代码

```

*EXPR:
.LFB1:
.LCFI0: pushq %rbp
.LCFI1: movq %rsp, %rbp
        movl %edi, -4(%rbp)
        movl -4(%rbp), %eax
        movl %eax, %edx
        imull -4(%rbp), %eax
        incl %eax
        imull %eax, %edx
        movl -4(%rbp), %eax
        incl %eax
        imull %eax, %edx
        incl %eax
        imull %eax, %eax
        incl %eax
        imull %eax, %eax
        movl %eax, -0(%rbp)
        leave
        ret

*EXPR:
.LFB2:
        movl %edi, %eax
        imull %edi, %eax
        incl %edi
        imull %edi, %eax
        imull %edi, %eax
        call $0, %eax
        ret
```

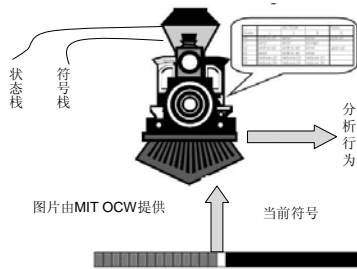
计算机理解体系



词法分析

- 词法分析器从文本流中分析出符号
- 符号通过常规表达定义

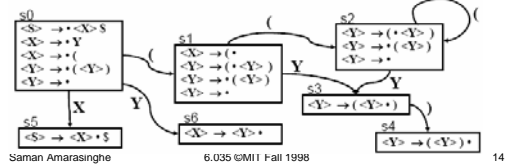
LR(K)语法分析器火车头



Saman Amarasinghe

| State | ACTION | | Goto | | |
|-------|-------------|-------------|-------------|---|---------|
| | (|) | \$ | X | Y |
| s0 | | | | | |
| s1 | shift to s2 | reduce (r1) | reduce (r1) | | goto s3 |
| s2 | shift to s2 | reduce (r2) | reduce (r2) | | goto s3 |
| s3 | error | shift to s4 | error | | |
| s4 | error | reduce (r4) | reduce (r4) | | |
| s5 | error | error | accept | | |
| s6 | error | error | reduce (r2) | | |

31



Saman Amarasinghe

14

语义分析

- 构建符号表
- 静态检查
 - 控制流检查
 - 一致性检查
 - 类型检查
- 动态检查
 - 数组边界检查
 - 空指针废弃检查

Saman Amarasinghe

6.035 ©MIT Fall 1998

15

转化成中间形式

- 目标：仍然保持很大程度上与机器的独立性，但和标准机器模式更接近
 - 从高级中间表达到低级中间表达
- 清除控制流结构
- 转换成扁平地址空间

Saman Amarasinghe

6.035 ©MIT Fall 1998

16

代码优化

- 优秀的编译程序员可以生成如同手工生成代码一样优秀的代码
- 有稳定、优秀的表现
- 无需程序员来抽象结构
 - 开发结构的优势
 - 避免程序的劣势

Saman Amarasinghe

6.035 ©MIT Fall 1998

17

代码优化

- 算法简化
- 通用子表达式删除
- 复写传播
- 常数传播
- 死代码删除
- 寄存器分配
- 指令调度

Saman Amarasinghe

6.035 ©MIT Fall 1998

18

编译器项目

- 你们已经从头创建了一个完备的编译器!!!
- 从decaf语言到运行的代码

编译器比赛

- 谁的编译器运行速度最快???
- 会提前12小时提供给学生程序
 - 测试并完成所有的优化工作
 - **不要在程序中添加特定的漏洞程序**
- 12月14日, 星期三, 上午11点在TBA
 - 提供饮料和点心

如何运用课程6.035中的知识

- 作为资深程序员
- 作为简单语言的设计者, 帮助完成其他的编程任务
- 作为处理新计算机结构的工程师
- 作为一个真正的编译器程序员

1. 资深程序员

- 现在知道编译器是做什么的了
 - 不要将它**死**黑匣子一样对待
 - 不要相信它会做正确的事情!
- 涉及
 - 性能
 - 调试
 - 正确性

1. 资深程序员

- 在课程6.035中学到了什么?
 - 优化功能如何运行或为什么他们不运行
 - 如果读懂优化后的代码

2. 语言扩展

- 在许多运用和系统中, 需要:
 - 简单语言实现
 - 处理输入
 - 定义接口
 - 命令和控制
 - 扩展语言
 - 添加新的功能
 - 修改语义
 - 帮助优化

2.语言扩展

- 在课程6.035中学到了什么
 - 通过常规表达式和CFG定义token和语言
 - 使用如jlex,lex,javacup,yacc等工具
 - 构建中间表达
 - 在中间表达上执行简单转换

3.计算机结构

- 很多特殊目的处理器
 - 在你的手机、汽车发动机、手表等
- 设计新的结构
- 使编译器后端适应新的体系结构

设计新的体系结构

- 在超大规模集成电路（VLSI）的巨大进步
 - 很快很小的转换器
 - 按比例增高到十亿转换器
 - 但是，很慢且有限的撞针和I/O口
- 计算机的体系机构是硬件和编译器的混合体
 - 需要明确什么是编译器可以做的，什么是硬件需要做的
 - 如果编译器可以完成，则不必浪费硬件资源

3.设计新的体系结构

- 在课程6.035中学到了什么？
 - 编译器的实际能力：什么实现起来简单，什么艰难
 - 如何像编译器作家一样思考

3.后端支持

- 每一个新的体系结构都需要一个新的后端
- 指令调度
 - 即使ISA总线相同（Industrial Standard Architecture，工业标准结构总线），也会有不同的资源冲突
 - 如何处理新的特征

3.后端支持

- 在课程6.035中学到了什么
 - 中间表达
 - 转换/优化中间表达
 - 将高级中间代码处理产生汇编语言
 - 汇编接口（如：调用惯例）
 - 寄存器分配
 - 代码调度

编译器编程

- 理论
- 算法
- 实现

编译器编程

- 理论：
 - 提出一般、抽象的概念
 - 证明正确性、最优性等等
- 示例
 - 语法分析理论
 - 格和数据流
 - 抽象解释
 - ML语言

4.编译器编程

- 算法
 - 对给定问题设计一个解决方案（大部分是新的优化）
 - 使用如图理论、数理论等方法
 - 可能必须要限制范围，同时找到好的启发式方法
- 示例
 - 部分冗余消除
 - 通过图着色进行寄存器分配
 - 使用多粒度（MMX，多媒体增强指令集）操作

编译器编程

- 实现
 - 开发一个新的编译器
 - 设计一个非常复杂软件
 - 将理论和算法付诸与实践
- 示例
 - Java的运行时编译技术
 - SQL查询优化引擎
 - Postscript语言的光栅化

4.编译器编程

- 在课程6.035中学到了什么？

一切!!!!

可到什么地方找寻最近的研究？

- PLDI-Programming Language Design and Implementation Conference 程序设计和实现会议
- 代码生成/机器特有
 - 小型会议
 - ASPLOS-Architecture Support for Programming Languages and Operating System 程序语言和操作系统的架构支援研讨会
 - CGO-Code Generation and Optimization 代码生成和优化
- 语言理论
 - POPL-编程语言原理
 - OOPSLA-Object Oriented Programming Systems Languages and Applications 面向对象编程系统语言和应用
- 程序分析
 - SAS-Static Analysis Symposium 静态分析会议
 - PPOPP-Principles and practice of Parallel Programming 并行程序设计原理