编译原理
Principles of Compiler

赵银亮
2008年秋

11/24/2008 9:43 AM

任课教师情况

- 赵银亮教授
  - 联系方式：zyl9910@mail.xjtu.edu.cn
- 辅导老师：李聪、王旭昊、张长军、韦远科

- 与编译有关的研究背景：
  - Lisp机Lisp_M1系统(1987-90)
  - Common Lisp Object System实现(1990-95)
  - 并行程序性能分析(1996-98)
  - C语言交叉编译器(2000-01)
  - DSP C编译系统(2002-05)
  - SpMT编译技术（2006-）

11/24/2008 9:43 AM

教学安排

- 总学时64
  - 课内学时56，主要讲授大纲中的知识点
  - 上机实验8，将安排具体题目，提交结果
  - 辅导答疑，每周五下午3-5点西一楼411室
- 综合成绩：
  - 上机 10%
    (由各班长于计算机系实验中心登记上机时间)
  - 作业 10% 指定时间提交，过后不补
  - 考试 80%

11/24/2008 9:43 AM

教材及参考书

- 陈火旺等. 程序设计语言编译原理（第3版）. 国防工业出版社.
- Kenneth C. Louden. Compiler Construction: Principles and Practice (英文版1997). 机械工业出版社, 2002
- Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman. Compilers :principles, techniques, and tools(2001)
- 课件及参考资料下载地址：
  ftp 202.117.15.193 用户名和密码均为parallel

11/24/2008 9:43 AM

主要教学内容

- 第一章,1.1-1.5 — 引论
- 第二章,2.1-2.3 — 形式语言简介
- 第三章,3.1-3.3 3.4* — 词法分析
- 第四章,4.1-4.6 — 自顶向下语法分析
- 第五章, 5.1-5.2 5.3* — 自底向上语法分析
- 第六章,6.1-6.2 — 属性文法
- 第七章, 7.1-7.7 — 中间代码生成
- 第八章, 8.1-8.4 — 符号表
- 第九章, 9.1-9.5 — 运行时存储空间组织
- 第十章, 10.1-10.3 — 代码优化

11/24/2008 9:43 AM

几点说明

- 先修课程：离散数学、数据结构、程序设计语言。
- 学习本课程的目标：理解编译过程；能够编写编译程序；有进行程序分析、变换、评价的基础。
- 学习的途径：原理入手（掌握知识点）；掌握相关理论方法和技术方法；多做练习。
- 本课程的编程实验推荐采用C或Java编程。

11/24/2008 9:43 AM

第一章 引论

- 什么叫编译程序
- 编译过程概述
- 编译程序的结构及生成

编译程序是一个程序，该程序将用某个语言写的程序(称为源程序)作为输入，并翻译成为功能上等价的另一个语言程序(称为目标程序).

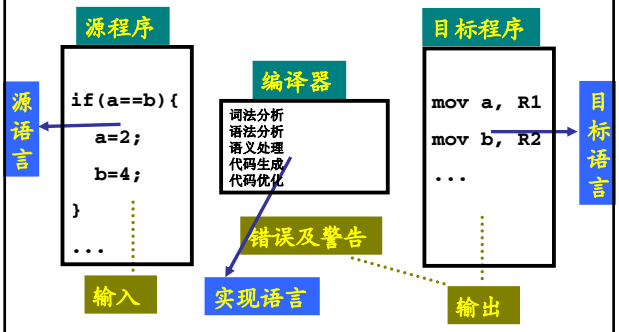
源语言通常是高级语言，如C，Fortran等；

目标语言通常是低级语言如汇编或机器语言；

随着翻译的进行，编译程序也报告错误或其他警告信息以帮助程序员改错，直到翻译通过为止.

11/24/2008 9:43 AM

基本概念



11/24/2008 9:43 AM

①源语言是高级语言，目标语言是低级语言时？源语言和目标语言任意时？

编译器/翻译程序

②书写编译器的语言可以是高级语言也可以是低级语言.

用什么语言编写编译器无关紧要

③运行编译器的机器跟运行目标程序的机器可以相同也可以不同.

编译器/交叉编译器　　宿主机/目标机

11/24/2008 9:43 AM

语言的抽象层次

- 机器语言：能够被计算机的硬件系统直接执行的指令程序
- 汇编语言：将硬件指令用一些助记符表示。如ADD表示加法操作，SUB表示减法操作等等
- 高级语言：相对于前二者而言且更接近于自然语言的程序设计语言

Properties:
–need to be precise
–need to be concise
–need to be expressive
–need to be at a high-level (lot of abstractions)

11/24/2008 9:43 AM

编译器的种类

- 诊断编译程序
  - 侧重于错误信息，帮助开发调试
- 优化编译程序
  - 侧重于提高代码效率
- 交叉编译程序
  - 宿主机和目标机不是同一个机器
- 可变目标编译程序
  - 仅定制编译器中跟机器有关部分实现针对新目标机的代码生成

11/24/2008 9:43 AM

解释器和编译器特点

- 概念上的不同。
- 基于解释执行的程序可以动态修改自身，而基于编译执行的程序动态性较弱。
- 基于解释方式有利于人机交互。
- 基于解释执行的速度较慢。
- 解释器需要保存的信息较多，空间开销大
- 二者实现技术相似。

11/24/2008 9:43 AM

1.2 编译过程概述

```
int sumcalc(int a, int b, int N)
{ int i; int x, y; x = 0; y = 0;
  for(i = 0; i <= N; i++) {
    x = x + (4*a/b)*i +
      (i+1)*(i+1);
    x = x + b*y;
  }
  return x;
}
```

```
test:
    subu   $fp, 16
    sw     zero, 0($fp)          # x = 0
    sw     zero, 4($fp)          # y = 0
    sw     zero, 8($fp)          # i = 0
labl:                            # for(i=0;i<N; i++)
    mul    $t0, $a0, 4           # a*4
    div    $t1, $t0, $a1         # a*4/b
    lw     $t2, 8($fp)           # i
    mul    $t3, $t1, $t2         # a*4/b*i
    lw     $t4, 8($fp)           # i
    addui$t4, $t4, 1             # i+1
    lw     $t5, 8($fp)           # i
    addui$t5, $t5, 1             # i+1
    mul    $t6, $t4, $t5         # (i+1)*(i+1)
    addu $t7, $t3, $t6           # a*4/b*i + (i+1)*(i+1)
    lw   $t8, 0($fp)             # x
    add  $t8, $t7, $t8           # x = x + a*4/b*i + (i+1)*(i+1)
    sw     $t8, 0($fp)
    lw     $t0, 4($fp)           # y
    mul    $t1, $t0, a1          # b*y
    lw     $t2, 0($fp)           # x
    add    $t2, $t2, $t1         # x = x + b*y
    sw     $t2, 0($fp)
    lw     $t0, 8($fp)           # i
    addui $t0, $t0, 1            # i+1
    sw    $t0, 8($fp)
    ble    $t0, $a3, labl
    lw     $v0, 0($fp)
    addu   $fp, 16
    b      $ra
```


源程序（字符流）

词法分析

单词符号（Token流）

语法分析

语法单位（语法树Parse树）

语义分析

中间表示

代码优化

优化的中间表示

代码生成

目标代码（汇编代码）

编译过程

Step1 词法分析

❖

源程序字符流：

2	3	4		*		(	1	1		+	-	2	2	)								

TOKEN流：

Num(234) mul_op lpar_op Num(11) add_op Num(-22) rpar_op

Step1 词法分析

18..23 + val#ue

不是数

变量名不能有＃字符

文法； 正规式； 有限自动机

Token例子

❖标识符(用户定义的名字); identifier
❖保留字(关键字/基本字); reserved words
❖常数(整数,双精度数,浮点数);constant
❖界符(分隔符); delimiters
❖运算符(操作符); operators

(种属，值)

例

b	e	g	i	n	△	v	a	r	△	x	1	:	r	e	a	l
;	↵	v	a	r	△	z	1	:	r	e	a	l	;	↵	x	1
:	=	0	.	5	;	↵	z	1	:	=	x	1	+	5	5	;
↵	w	r	i	t	e	(	z	1	+	5	.	5	)	;	↵	r
e	a	d	(	V	1	)	;	↵	z	1	:	=	z	1	+	x
1	↵	e	n	d	.	#										

源程序在文件中的表示

词法分析后的TOKEN表示

11/24/2008 9:43 AM

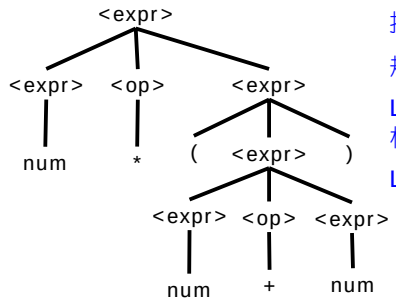
Step2 语法分析

- 依据语言的语法规则，将单词的**Token**序列转化为语法单位（语法范畴）（短语、子句、句子、程序段、程序），并确定整个输入串是否构成一个语法上正确的程序。

11/24/2008 9:43 AM

语法分析

num '*' '(' num '+' num ')'



文法；

语法树；

推导；

规约；

LL(1)分析法

LR分析法

11/24/2008 9:43 AM

语法分析

```
int * foo(i, j, k))
  int i;
  int j;
{
  for(i=0; i j) {
  fi(i>j)
  return j;
}
```

多余的括号

缺少步长

不是保留字

不是一个表达式

11/24/2008 9:43 AM

Step3 语义分析

- 对语法分析所识别出的各类语法范畴，分析其含义，查错，并产生源程序所对应的中间表示（中间代码）。

属性文法；　语义函数

11/24/2008 9:43 AM

语义分析

```
int * foo(i, j, k)
  int i;
  int j;
{
  int x;
  x = x + j + N;
  return j;
}
```

类型未说明

变量未说明

返回结果类型不匹配

使用了没有初始化的变量

11/24/2008 9:43 AM

Step4 优 化

```
int sumcalc(int a, int b, int N)
{
   int i;
   int x, y;
   x = 0;
   y = 0;
   for(i = 0; i <= N; i++) {
        x = x + (4*a/b)*i +
                (i+1)*(i+1);
        x = x + b*y;
   }
   return x;
}
```

```
int sumcalc(int a, int b, int N)
{
   int i;
   int x, t, u, v;
   x = 0;
   u = (a<<2/b);
   v = 0;
   for(i = 0; i <= N; i++) {
        t = i+1;
        x = x + v + t*t;
        v = v + u;
   }
   return x;
}
```

程序等价变换； 控制流、数据流分析

11/24/2008 9:43 AM

Constant Propagation

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, y;
  x = 0;
  y = 0;
  for(i = 0; i <= N; i++) {
     x = x + (4*a/b)*i + (i+1)*(i+1);
     x = x + b*0;
  }
  return x;
}
```

Algebraic Simplification

11/24/2008 9:43 AM

Algebraic Simplification

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, y;
  x = 0;
  y = 0;
  for(i = 0; i <= N; i++) {
     x = x + (4*a/b)*i + (i+1)*(i+1);
     x = x;
  }
  return x;
}
```

Copy Propagation

11/24/2008 9:43 AM

Copy Propagation

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, y;
  x = 0;
  y = 0;
  for(i = 0; i <= N; i++) {
     x = x + (4*a/b)*i + (i+1)*(i+1);
  }
  return x;
}
```

Common Subexpression Elimination

11/24/2008 9:43 AM

Common Subexpression Elimination

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, y, t;
  x = 0;
  y = 0;
  for(i = 0; i <= N; i++) {
     t = i+1;
     x = x + (4*a/b)*i + t*t;
  }
  return x;
}
```

Dead Code Elimination

11/24/2008 9:43 AM

Dead Code Elimination

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, t;
  x = 0;
  for(i = 0; i <= N; i++){
     t = i+1;
     x = x + (4*a/b)*i + t*t;
  }
  return x;
}
```

Loop Invariant Removal

11/24/2008 9:43 AM

Loop Invariant Removal

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, t, u;
  x = 0;
  u = (4*a/b);
  for(i = 0; i <= N; i++) {
      t = i+1;
      x = x + u*i + t*t;
  }
  return x;
}
```

Strength Reduction

11/24/2008 9:43 AM

Strength Reduction

```
int sumcalc(int a, int b, int N)
{
  int i;
  int x, t, u, v;
  x = 0;
  u = (a<<2/b);
  v = 0;
  for(i = 0; i <= N; i++) {
      t = i+1;
      x = x + v + t*t;
      v = v + u;
  }
  return x;
}
```

11/24/2008 9:43 AM

Step5 代码生成

- 中间代码变换为特定机器上的绝对指令代码或可重定位的指令代码或汇编指令代码。

指令系统

11/24/2008 9:43 AM

```
test:
   subu $fp, 16
   add  $t9, zero, zero    # x = 0
   sll  $t0, $a0, 2        # a<<2
   div  $t7, $t0, $a1      # u = (a<<2)/b
   add  $t6, zero, zero    # v = 0
   add  $t5, zero, zero    # i = 0

lab1:                      # for(i=0;i<N; i++)
   addui$t8, $t5, 1        # t = i+1
   mul  $t0, $t8, $t8      # t*t
   addu $t1, $t0, $t6      # v + t*t
   addu $t9, t9, $t1       # x = x + v + t*t

   addu $6, $6, $7         # v = v + u

   addui$t5, $t5, 1        # i = i+1
   ble  $t5, $a3, lab1

   addu $v0, $t9, zero
   addu $fp, 16
   b    $ra
```

Unoptimized Code

```
test:
    subu    $fp, 16
    sw      zero, 0($fp)
    sw      zero, 4($fp)
    sw      zero, 8($fp)
lab1:
    mul     $t0, $a0, 4
    div     $t1, $t0, $a1
    lw      $t2, 8($fp)
    mul     $t3, $t1, $t2
    lw      $t4, 8($fp)
    addui   $t4, $t4, 1
    lw      $t5, 8($fp)
    addui   $t5, $t5, 1
    mul     $t6, $t4, $t5
    addu    $t7, $t3, $t6
    lw      $t8, 0($fp)
    add     $t8, $t7, $t8
    sw      $t8, 0($fp)
    lw      $t0, 4($fp)
    mul     $t1, $t0, a1
    lw      $t2, 0($fp)
    add     $t2, $t2, $t1
    sw      $t2, 0($fp)
    lw      $t0, 8($fp)
    addui   $t0, $t0, 1
    sw      $t0, 8($fp)
    ble     $t0, $a3, lab1

    lw      $v0, 0($fp)
    addu    $fp, 16
    b       $ra
```

4*ld/st + 2*add/sub + br +
N*(9*ld/st + 6*add/sub + 4* mul + div + br)
= 7 + N*21

Execution time = 43 sec

Optimized Code

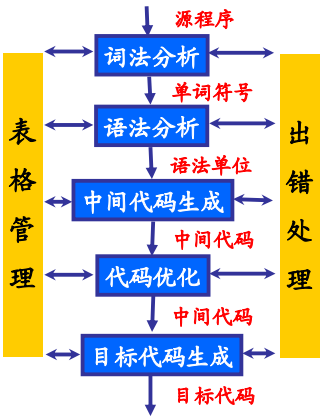
```
test:
    subu    $fp, 16
    add     $t9, zero, zero
    sll     $t0, $a0, 2
    div     $t7, $t0, $a1
    add     $t6, zero, zero
    add     $t5, zero, zero
lab1:
    addui   $t8, $t5, 1
    mul     $t0, $t8, $t8
    addu    $t1, $t0, $t6
    addu    $t9, t9, $t1
    addu    $t, $6, $7
    addui   $t5, $t5, 1
    ble     $t5, $a3, lab1

    addu    $v0, $t9, zero
    addu    $fp, 16
    b       $ra
```

6*add/sub + shift + div + br +
N*(5*add/sub + mul + br)
= 9 + N*7

Execution time = 17 sec

1.3 编译程序的结构及生成

- 表格管理
- 出错处理
- 遍
- 前端与后端
- T型图



11/24/2008 9:43 AM

表格管理

- 表格的作用和种类
  - 保存编译各阶段的有用信息
  - 符号表
  - 常数表
  - 保留字表
  - 其他
- 表格操作
  - 关注方便性和时空效率

11/24/2008 9:43 AM

出错处理

- 语法错误　不符合语法（或词法）规则的错误。单词拼写错误、括号不匹配等
- 语义错误　不符合语义规则的错误。说明错误、作用域错误、类型不匹配等

- 全　最大限度发现错误
- 准　准确指出错误的性质和发生地点
- 局部化　将错误的影响限制在尽可能小的范围内

11/24/2008 9:43 AM

一遍与多遍编译

- 是对源程序或源程序的中间结果从头到尾扫描一次，并作有关的加工处理，生成新的中间结果或目标程序。

- 注意各遍间的中间结果（内部形式、外部形式）

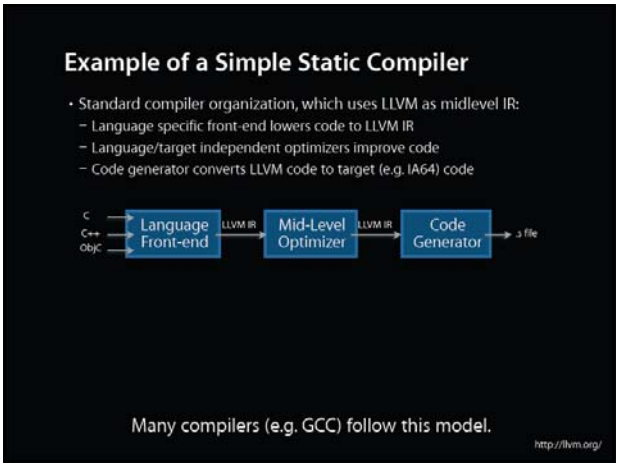
One-pass compiler: Type of software compiler that passes through the source code only once. One-pass compilers are faster, but may not generate an as efficient program. In addition, one-pass compilers cannot compile all types of source codes.
http://www.computerhope.com/jargon/o/onepassc.htm

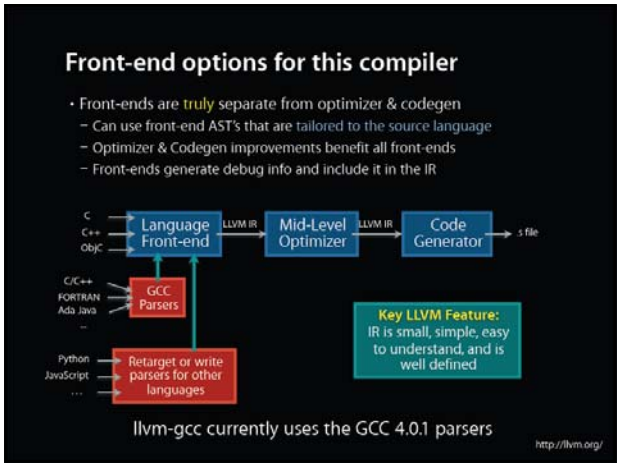
11/24/2008 9:43 AM

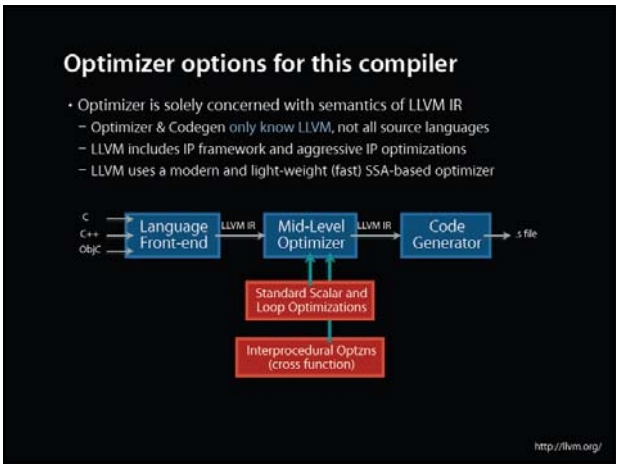
编译前端与后端

- 前端指跟目标机无关的部分，一般包括：词法、语法、语义分析；
- 后端指跟目标机有关的部分，一般包括优化和代码生成；
- 前后端之间的接口：
  - 中间表示
  - 符号表

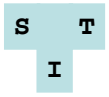
11/24/2008 9:43 AM







T型图

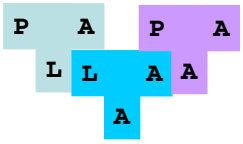


- S – Source Language
- T – Target Language
- I – Implementation Language

由用I语言写的编译程序把用S语言写的源程序翻译成用T语言写的目标程序

11/24/2008 9:43 AM

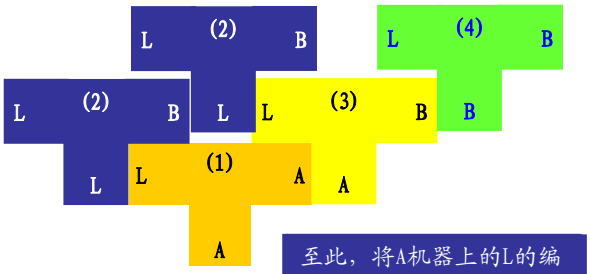
用高级语言L构造编译程序



- 将写好的语言P的编译程序用L的编译程序编译后就可得到用A机器代码实现的P编译程序

11/24/2008 9:43 AM

编译程序的移植



11/24/2008 9:43 AM

自编译方式

- 先对语言的核心部分构造一个小小的编译程序（可用低级语言实现）
- 再以它为工具构造能编译更多语言成分的较大编译程序
- 如此不断扩展，最后形成整个编译程序（滚雪球）

11/24/2008 9:43 AM

构造工具

- 构造编译程序的工具称为编译程序-编译程序、编译程序产生器或翻译程序书写系统
- 自动产生扫描器　　LEX　FLEX
- 自动产生语法分析器　YACC　BISON

11/24/2008 9:43 AM

本章小结

- 编译过程：5个阶段
- 编译理论是基础：形式语言、属性文法、数据流方程等
- 编译技术是关键：各部分的构造、编译中的数据结构等

11/24/2008 9:43 AM

课程学习指导

- 具备一定的程序设计**语言背景**
- 本课程**目标机器**采用**假想指令**，**算法描述**使用**类PASCAL伪码**
- 因为编译程序比较**复杂**，故**讨论中将其分解成各个部分**，学习中应注意**前后联系**，**多做练习**

11/24/2008 9:43 AM

本章练习

- 概要了解编译过程的内容和特点。

11/24/2008 9:43 AM