7.1 中间代码的形式

- 中间代码的作用
  - 过渡：经过语义分析被译成中间代码序列
- 中间代码的形式
  - 中间语言的语句
- 中间代码的优点
  - 形式简单、语义明确、独立于目标语言
  - 便于编译系统的实现、移植、代码优化
- 常用的中间代码
  - 语法树(6.3.5节)
  - 逆波兰表示、三地址码(三元式和四元式)、DAG图表示

7.1.1 逆波兰表示

- 中缀表达式的计算顺序不是运算符出现的自然顺序，而是根据运算符间的优先关系来确定的，因此，从中缀表达式直接生成目标代码一般比较麻烦。
- 波兰逻辑学家J. Lukasiewicz于1929年提出了后缀表示法，其优点为：表达式的运算顺序就是运算符出现的顺序，它不需要使用括号来指示运算顺序。

7.1.1 逆波兰表示

- 例7.1 下面给出的是一些表达式的中缀、前缀和后缀表示。

中缀表示	前缀表示	后缀表示
a+b	+ab	ab+
a*(b+c)	*a+bc	abc+*
(a+b)*(c+d)	*+ab+cd	ab+cd+*
a:=a*b+c*d	:=a+*ab*cd	abc*bd*+:=

7.1.2 三地址码

- 所谓三地址码，是指这种代码的每条指令最多只能包含三个地址，即两个操作数地址和一个结果地址。
- 如$x+y*z$三地址码为：$t_1 := y*z$　　$t_2 := x+t1$
- 三地址码中地址的形式：
  - 名字、常量、编译器生成的临时变量。

7.1.2 三地址码

- 例7.2 赋值语句a:=(-b)*(c+d)-(c+d)的三地址码如图7.1所示

t_1 := minus b
t_2 := c+d
t_3 := t_1*t_2
t_4 := c+d
t_5 := t_3-t_4
a := t_5

图7.1 a:=(-b)*(c+d)-(c+d)的三地址码

四元式

- 四元式是一种比较常用的中间代码形式，它由四个域组成，分别称为op、arg1、arg2和result。op是一个一元或二元运算符，arg1和arg2分别是op的两个运算对象，它们可以是变量、常量或编译器生成的临时变量，运算结果则放入result中。

	op	arg_1	arg_2	result
0	minus	b		t_1
1	+	c	d	t_2
2	*	t_1	t_2	t_3
3	+	c	d	t_4
4	-	t_3	t_4	t_5
5	assign	t_5		a
		...		

图7.2(a) 图7.1中三地址码的四元式表示

三元式

- 为了节省临时变量的开销，有时也可以使用只有三个域的三元式来表示三地址码。三元式的三个域分别称为op，arg1和arg2，op，arg1和arg2的含义与四元式类似，区别只是arg1和arg2可以是某个三元式的编号(图7.2(b)中用圆括号括起来的数字)，表示用该三元式的运算结果作为运算对象。

	op	arg_1	arg_2
0	minus	b	
1	+	c	d
2	*	(0)	(1)
3	+	c	d
4	-	(2)	(3)
5	assign	a	(4)
		...	

图7.2(b) 图7.1中三地址码的三元式表示

生成三地址码的语法制导定义

产生式	语义规则
S→id:=E	S.code:=E.code‖gencode(id.addr':='E.addr)
$E \rightarrow E_1+E_2$	E.addr:=newtemp;E.code := E_1.code ‖ E_2.code ‖ gencode(E.addr':='E_1.addr'+' E_2.addr)
$E \rightarrow E_1*E_2$	E.addr:=newtemp;E.code := E_1.code ‖ E_2.code ‖ gencode(E.addr':='E_1.addr'*' E_2.addr)
$E \rightarrow -E_1$	E.addr:=newtemp; E.code := E_1.code ‖ genode(E.addr':=''uminus' E_1.addr)
$E \rightarrow (E_1)$	E.addr:= E_1.addr; E.code := E_1.code
E→ id	E.addr:= id.addr; E.code := ‘’
E→ num	E.addr:= num.val; E.code := ‘’

属性 code 表示生成的代码

7.3 赋值语句的翻译

翻译的需求

- 充分了解各种语言现象的语义
  - 包括：控制结构、数据结构、单词
  - 充分了解它们的实现方法
- 目标语言的语义
  - 了解中间代码的语义
  - 了解运行环境

辅助子程序与语义属性设置

- 辅助子程序
  - gencode(code)，emit(code)：产生一条中间代码
  - newtemp：产生新的临时变量
  - lookup：检查符号表中是否出现某名字
- 语义属性设置
  - 中间代码序列：code
  - 地址：addr
  - 下一条四元式序号：nextquad

7.3.1 简单赋值语句的翻译

$S \rightarrow$ id := E {p := lookup(id.name);
if p ≠ nil then
gencode(p, ':=', E.addr)
else error }

$E \rightarrow E_1 + E_2$ {E.addr := newtemp;
emit(E.addr,':=',E_1.addr,'+

$E \rightarrow -E_1$ {E.addr := newtemp;
emit(E.addr,':=','uminu

$E \rightarrow (E_1)$ {E.addr := E_1.addr }

$E \rightarrow$ id {p := lookup(id.name);
if p ≠ nil then E.addr := p else error }

x:=-a*b+c翻译为以下三地址码序列：
t1:=minus a
t2:=t1 * b
t3:=t2 + c
t4:=t3