山东大学
SHANDONG UNIVERSITY

编译原理

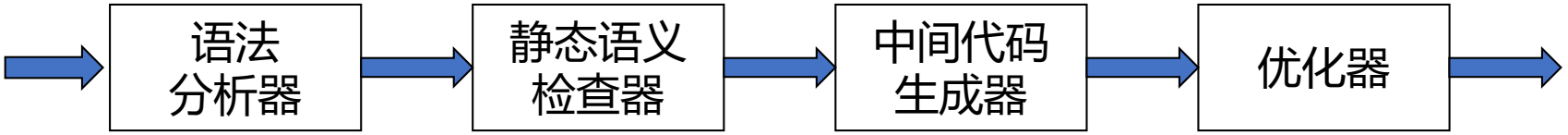
第七章 语义分析和中间代码生成

授课教师：余仲星
手机：15866821709（微信同号）
邮箱：zhongxing.yu@sdu.edu.cn

第七章 语义分析和中间代码生成

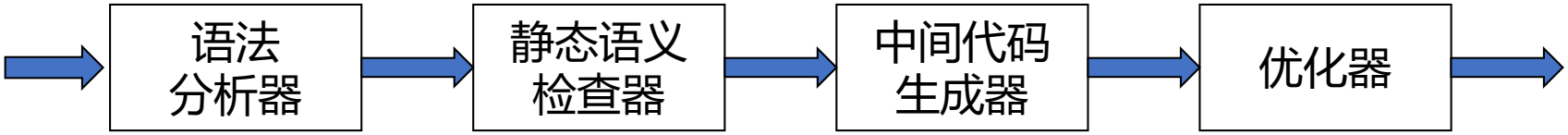
- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

第七章 语义分析和中间代码生成



- 静态语义检查通常包括：
  - 类型检查：如果操作符作用于不相容的操作数，编译程序必须报告出错信息。
  - 控制流检查：控制流语句必须使控制转移到合法的地方，如c语言中，break如果不包含在while、for或switch等语句中，则报错。
  - 一致性检查：很多场合要求对象只能被定义一次。
  - 相关名字检查：有时同一名字必须出现两次或多次，需要检测出现的名字是否相同。
  - 名字的作用域分析：确定作用域范围。

第七章 语义分析和中间代码生成

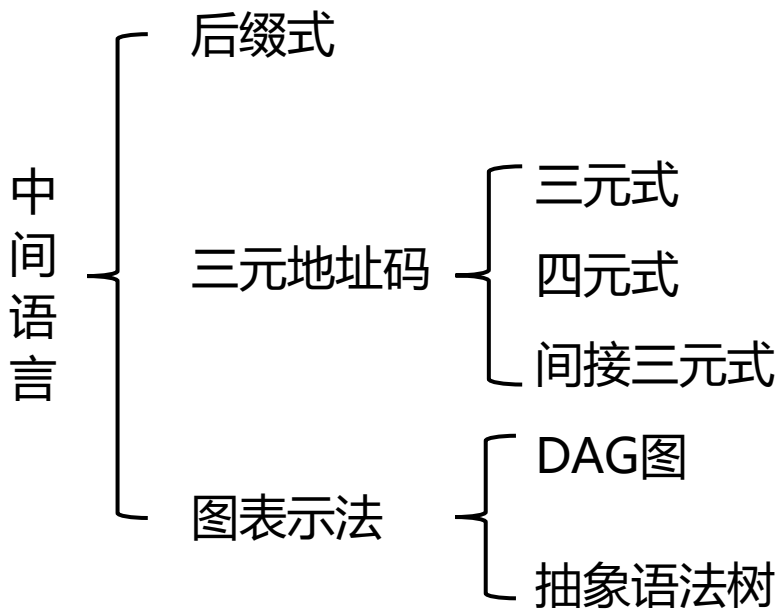


- 虽然源程序可以直接翻译为目标语言代码，但许多编译程序却采用了独立于机器的、复杂性介于源语言和机器语言之间的中间语言，这样做的好处是：
  - 便于进行与机器无关的代码优化工作；
  - 使编译程序改变目标机更容易；
  - 使编译程序的结构在逻辑上更为简单明确。

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.1 中间语言



第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.1.1 后缀式

- 后缀式表示法是波兰逻辑学家卢卡西维奇（Lukasiewicz）发明的一种表达式的表示方法，因此又称逆波兰表示法。
- 这种表示法是：把运算量写在前面，把算符写在后面。
  - $a+b$写成$ab+$，$a*b$写成$ab*$
- 用E或E_i表示中缀形式，$<E>$表示E的后缀形式，一个表达式E的后缀形式定义如下
  - ① 如果a是一个变量或常量，$<a>=a$；
  - ② 如果θ是单目运算符，$<\theta E>=<E>\theta$；
  - ③ 如果θ是双目运算符，$<E_1\theta E_2>=<E_1><E_2>\theta$；
  - ④ $<(E)>=<E>$。
- 若θ是一个k目运算符，它对运算量$e_1, e_2, \ldots, e_k$的作用结果表示成：$e_1 e_2 \ldots e_k \theta$

7.1.1 后缀式

- 与中缀及前缀表示相比：
  - 运算符个数不变；
  - 运算量的次序和个数不变。
- 后缀式优点：
  - 无括号，形式简洁；
  - 运算符的顺序与运算次序完全相同。
- 把表达式翻译成后缀式的语义规则：
  - $E \to E_1\ \theta\ E_1$　　$E.code = E_1.code \,||\, E_2.code \,||\, \theta$
  - $E \to (E_1)$　　$E.code = E_1.code$
  - $E \to id$　　$E.code = id$

Convert the infix expression into its equivalent postfix expression

Algorithm to convert an Infix notation into postfix notation

Step 1: Add ‘)” to the end of the infix expression

Step 2: Push “(“ on to the stack

Step 3: Repeat until each character in the infix notation is scanned

IF a “(“ is encountered, push it on the stack

IF an operand (whether a digit or an alphabet) is encountered, add it to the postfix expression.

IF a “)” is encountered, then;

a. **Repeatedly pop from stack and add it to the postfix expression until a “(” is encountered.**

b. Discard the “(“. That is, remove the “(“ from stack and do not add it to the postfix expression

IF an operator X is encountered, then;

a **Repeatedly pop from stack and add each operator (popped from the stack) to the postfix expression which has the same precedence or a higher precedence than X**

b. Push the operator X to the stack

Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty

Step 5: EXIT

7.1.1 后缀式

【例7.1】

$$< a*b+c*d >$$
$$= < a*b >< c*d > +$$
$$= ab*cd*+$$

【例7.2】

$$< (a+b)*(c*d+e) >$$
$$= < (a+b) >< (c*d+e) >*$$
$$=< a+b >< c*d+e >*$$
$$= ab+< c*d >< e >+*$$
$$= ab+cd*e+*$$

【例7.3】

$$< (a+b)\hat{}(c*d)\hat{}e >$$
$$= < (a+b) >< (c*d)\hat{}e > \hat{}$$
$$= < a+b >< (c*d) >< e > \hat{}\hat{}$$
$$= ab+cd*e\hat{}\hat{}$$

优先级搞错的错误做法：

$$< (a+b)\hat{}(c*d)\hat{}e >$$
$$= < (a+b)\hat{}(c*d) >< e > \hat{}$$
$$= < (a+b) >< (c*d) > \hat{}e\hat{}$$
$$= ab+cd*\hat{}e\hat{}$$

7.1.1 后缀式

【例7.4】

$< a \le b + c \wedge a > d \vee a + b \ne e >$

$= < a \le b + c \wedge a > d >< a + b \ne e > \vee$

$= < a \le b + c >< a > d > \wedge < a + b >< e > \ne \vee$

$= < a >< b + c > \le ad > \wedge ab + e \ne \vee$

$= abc + \le ad > \wedge ab + e \ne \vee$

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.1.2 图表示法

- 抽象语法树：已在前一章介绍。
- DAG图，即无循环有向图（Directed Acyclic Graph）。
  - 可以识别公共子表达式，在代码优化部分介绍。

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.1.3 三地址代码

- 三地址代码：由以下一般形式的语句构成的序列$x = y\ op\ z$。
  - x, y, z为名字、常数或编译产生的临时变量；
  - op代表运算符号。
- $x + y * z$可以翻译成如下语句序列：

$$T_1 = y * z$$

$$T_2 = x + T_1$$

其中T_1和T_2为编译时产生的临时变量。

7.1.3 三地址代码

三地址语句的种类

① $x = y\ op\ z$，其中op为二元算术算符或逻辑算符。

② $x = op\ y$，其中op为一元算符，如一元减uminus、逻辑非not、移位算符及类型转换算法。

③ $x = y$的赋值语句，将y的值赋给x。

④ $goto\ L$，无条件转移语句。

⑤ $if\ x\ relop\ y\ goto\ L$或$if\ a\ goto\ L$的条件转移语句，其中$relop$为关系运算符如<,=,>,≤,≥等， a为布尔变量或常量。若if条件为真，则执行标号为L的语句，否则执行下一条语句。

7.1.3 三地址代码

三地址语句的种类

⑥ 过程调用语句$param\ x$和$call\ p,n$，以及返回语句$return\ y$。源程序中的过程调用语句$p(x_1,x_2,\ldots,x_n)$通常产生如下的三地址代码：

$param\ x_1$

$param\ x_2$

……

$param\ x_n$

$call\ p,n$

⑦ $x=y[i]$及$x[i]=y$的索引赋值。

⑧ $x=\&y, x=*y, *x=y$的地址和指针赋值。

7.1.3 三地址代码

- 四元式：带有四个域的记录结构$(op, arg1, arg2, result)$，相当于$result = arg1\ op\ arg2$
  - op为一个代表运算符的内部码；
  - $arg1, arg2$为两个运算数；
  - $result$结果域。

【例7.5】$a = b * -c + b * -c$的四元式

$(uminus, c, -, T_1)$

$(*, b, T_1, T_2)$

$(uminus, c, -, T_3)$

$(*, b, T_3, T_4)$

$(+, T_2, T_4, T_5)$

$(=, T_5, -, a)$

7.1.3 三地址代码

- 三元式：为避免把临时变量填入到符号表，可以通过计算这个临时变量的语句位置来引用这个临时变量，这样只需要三个域$(op, arg1, arg2)$
  - op为一个代表运算符的内部码；
  - $arg1, arg2$为运算数或三元式标号。

【例7.6】$a = b * -c + b * -c$的三元式

$(0)\ (uminus, c, -)$

$(1)\ (*, b, (0))$

$(2)\ (uminus, c, -)$

$(3)\ (*, b, (2))$

$(4)\ (+, (1), (3))$

$(5)\ (assign, a, (4))$

7.1.3 三地址代码

- 间接三元式：为了便于代码优化处理，有时不直接使用三元式表，而是另设一张指示器（称为间接码表），它将按运算的先后顺序列出有关三元式在三元式表中的位置。
  - 代码优化过程中需要调整运算顺序时，只需重新安排间接码表。

【例7.7】如下语句的间接三元式

$x = (a + b) * c$

$y = d\hat{}(a + b)$

间接代码	三元式表
(1)	(1) $(+, a, b)$
(2)	(2) $(*, (1), c)$
(3)	(3) $(=, x, (2))$
(1)	(4) $(\hat{}, d, (1))$
(4)	(5) $(=, y, (4))$
(5)	

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.2.1 过程中的说明语句

- 过程中是说明语句需要处理的问题：

① 过程中的说明语句形式：$id_1, id_2, \ldots, id_n : type$

② 每个变量需要记录名字、类型、字宽信息，分别用属性name、type、width记录

③ 当新出现一个名字时，需要记录进符号表，用offset记录该名字在符号表中的地址偏移量，在识别前置初值为0 。

④ 过程enter(name, type, offset)用来把名字name填入到符号表，并给出该名字的类型type及在过程数据区中的相对地址offset。

⑤ 假定整数类型域宽为4，实型域宽为8。

7.2.1 过程中的说明语句

- 过程中的说明语句翻译模式：

$P \rightarrow \{offset = 0\}D$

$D \rightarrow D; D$

$D \rightarrow id : T \quad \{enter(id.name, T.type, offset);\ offset = offset + T.width\}$

$T \rightarrow integer \quad \{T.type = integer;\ T.width = 4\}$

$T \rightarrow real \quad \{T.type = real;\ T.width = 8\}$

$T \rightarrow array[num]\ of\ T_1 \quad \{T.type = array(num.val, T_1.type);\ T.width = num.val \times T_1.width\}$

$T \rightarrow \hat{}T_1 \quad \{T.type = pointer(T_1.type); T.width = 4\}$

7.2.1 过程中的说明语句

- 过程中的说明语句翻译模式：

$D \rightarrow D;D$

$D \rightarrow D:T$

$D \rightarrow id$

【例6.20】

$D \rightarrow L:T$

$T \rightarrow integer \mid char$

$L \rightarrow L,id \mid id$

【修改】

$D \rightarrow id\ L$

$L \rightarrow ,id\ L \mid :T$

$T \rightarrow integer \mid char$

- 过程中的说明语句翻译模式：

$P \rightarrow \{offset = 0\}D$

$D \rightarrow id\ L$	$\{enter(id.name, L.type, offset);$ $offset = offset + L.width\}$
$L \rightarrow, id\ L_1$	$\{enter(id.name, L_1.type, offset);$ $offset = offset + L_1.width;$ $L.type = L_1.type; L.width = L_1.width\}$
$L \rightarrow : T$	$\{L.type = T.type; L.width = T.width\}$
$T \rightarrow integer$	$\{T.type = integer;\ T.width = 4\}$
$T \rightarrow real$	$\{T.type = real;\ T.width = 8\}$
$T \rightarrow array[num]\ of\ T_1$	$\{T.type = array(num.val, T_1.type);$ $T.width = num.val \times T_1.width\}$
$T \rightarrow \hat{}T_1$	$\{T.type = pointer(T_1.type); T.width = 4\}$

- 过程中的说明语句翻译模式：

$P \rightarrow MD$

$M \rightarrow \varepsilon$ $\{offset = 0\}$

$D \rightarrow id\ L$ $\{enter(id.name, L.type, offset);$
$offset = offset + L.width\}$

$L \rightarrow, id\ L_1$ $\{enter(id.name, L_1.type, offset);$
$offset = offset + L_1.width;$
$L.type = L_1.type; L.width = L_1.width\}$

$L \rightarrow : T$ $\{L.type = T.type; L.width = T.width\}$

$T \rightarrow integer$ $\{T.type = integer;\ T.width = 4\}$

$T \rightarrow real$ $\{T.type = real;\ T.width = 8\}$

$T \rightarrow array[num]\ of\ T_1$ $\{T.type = array(num.val, T_1.type);$
$T.width = num.val \times T_1.width\}$

$T \rightarrow \hat{}T_1$ $\{T.type = pointer(T_1.type); T.width = 4\}$

过程中的说明语句

- 【例7.8】

① $P \rightarrow MD$

② $M \rightarrow \varepsilon$

③ $D \rightarrow id\ L$

④ $L \rightarrow, id\ L_1$

⑤ $L \rightarrow : T$

⑥ $T \rightarrow integer$

⑦ $T \rightarrow real$

⑧ $T \rightarrow array[num]\ of\ T_1$

⑨ $T \rightarrow \hat{}T_1$

步骤	文法符号栈	输入串	动作
1	#	$p,q,r:real\#$	初始化
2	$\#M$	$,q,r:real\#$	归约r_2
3	$\#Mp$	$,q,r:real\#$	移进
4	$\#Mp,$	$q,r:real\#$	移进
5	$\#Mp,q$	$,r:real\#$	移进
6	$\#Mp,q,$	$r:real\#$	移进
7	$\#Mp,q,r$	$:real\#$	移进
8	$\#Mp,q,r:$	$real\#$	移进
9	$\#Mp,q,r:real$	#	移进
10	$\#Mp,q,r:T$	#	归约r_7
11	$\#Mp,q,rL$	#	归约r_5
12	$\#Mp,qL$	#	归约r_4
13	$\#MpL$	#	归约r_4
14	$\#MD$	#	归约r_3
15	$\#P$	#	归约r_1
16	$\#P$	#	成功

- 过程中的说明语句翻译模式：

$P \rightarrow MD$

$M \rightarrow \varepsilon$ $\{offset = 0\}$

$D \rightarrow id\ L$ $\{enter(val[top-1].name, val[top].type, offset);$
$offset = offset + val[top].width\}$

$L \rightarrow, id\ L_1$ $\{enter(val[top-1].name, val[top].type, offset);$
$offset = offset + val[top].width;$
$val[ntop] = val[top]\}$

$L \rightarrow : T$ $\{val[ntop] = val[top]\}$

$T \rightarrow integer$ $\{val[ntop].type = integer; val[ntop].width = 4\}$

$T \rightarrow real$ $\{val[ntop].type = real; val[ntop].width = 8\}$

$T \rightarrow array[num]\ of\ T_1$ $\{val[ntop].type = array(num.val, val[top].type);$
$val[ntop].width = num.val \times val[top].width\}$

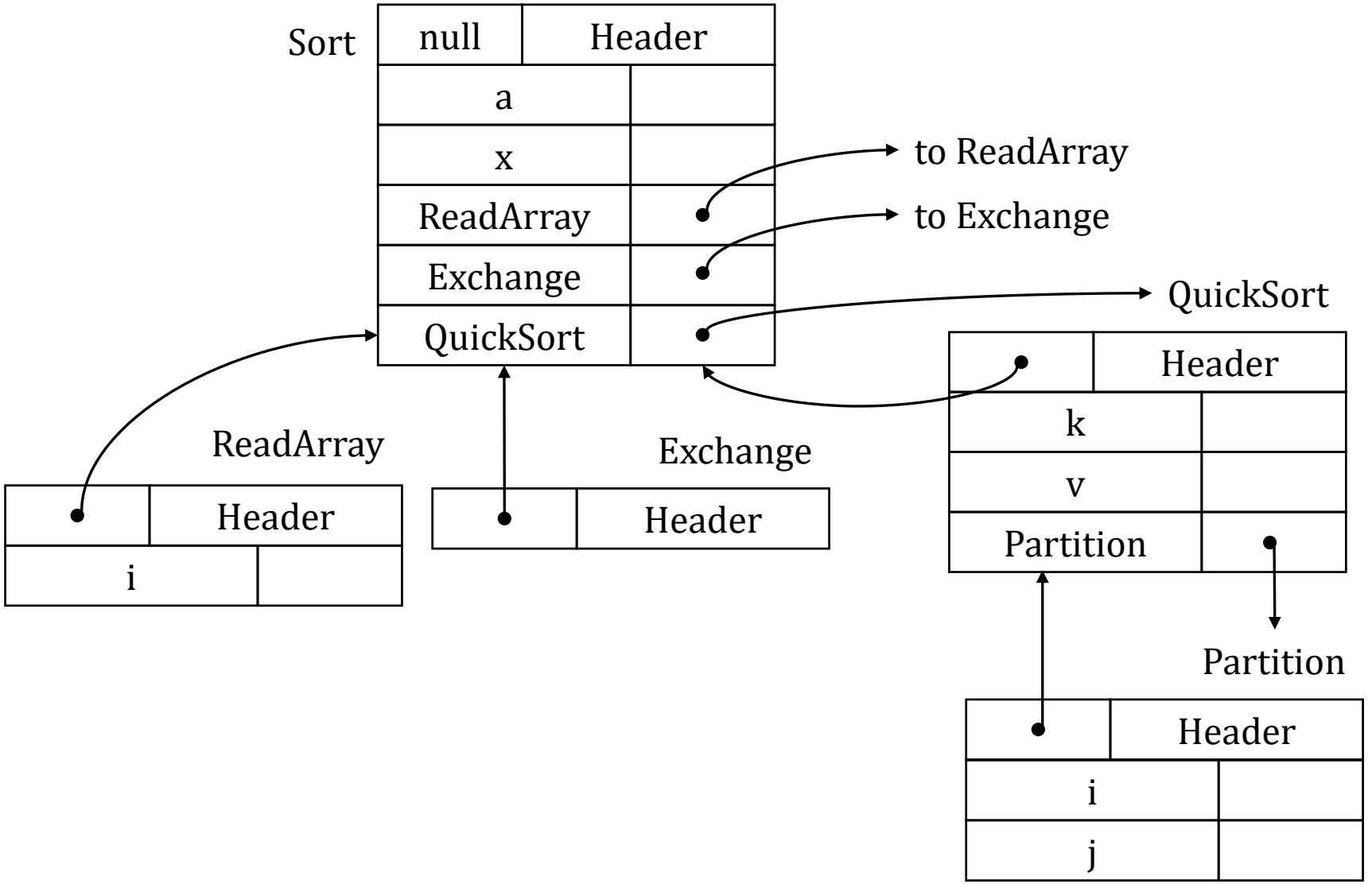
$T \rightarrow \hat{}T_1$ $\{val[ntop].type = pointer(val[top].type); val[ntop].width = 4\}$

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

```
① program Sort (input , output)
②       var a: array[0..10] of integer;
③           x: integer;
④       procedure ReadArray
⑤           var i: integer;
⑥           begin ... a... end {ReadArray}
⑦       procedure Exchange (i, j: integer)
⑧           begin x=a[i]; a[i] = a[j]; a[j] = x; end {Exchange}
⑨       procedure QuickSort (m, n: integer)
⑩           var k, v: integer;
⑪           function Partition (y, z: integer) : integer
⑫               var i, j: integer;
⑬               begin ...a...v... Exchange(i, j);... end {Partition}
⑭           begin ... end {QuickSort}
⑮       begin ... end {Sort}
```

嵌套过程的符号表


Sort
null
Header
a
x
ReadArray
Exchange
QuickSort
to ReadArray
to Exchange
QuickSort
Header
k
v
Partition
ReadArray
Header
i
Exchange
Header
Partition
Header
i
j

保留作用域的信息

- 语义规则中的操作：
  - $mktable(previous)$：创建一张新符号表，并返回指向新表的指针；参数$previous$指向先前创建的一张符号表。
  - $enter(table, name, type, offset)$：在指针$table$指向的符号表中，为名字$name$建立一个新项，并把类型$type$ 、相对地址$offset$填入到该项中。
  - $addwidth(table, width)$：在指针$table$指向的符号表表头中，记录下该表中所有名字占用的总宽度。
  - $enterproc(table, name, newtable)$：在指针$table$指向的符号表中，为名字为$name$的过程建立一个新项；参数$newtable$指向过程$name$的符号表。
  - $tblptr$是一个栈，用于存放指向嵌套外层过程的符号表指针。
  - $offset$是一个栈，用于存放变量的相对地址，当过程结束时， $offset$里记录的是过程占用的所有字节数。

- 保留作用域信息：

$P \rightarrow MD$ $\{addwith\big(top(tblptr), top(offset)\big);$
$pop(tblptr); pop(offset);\}$

$M \rightarrow \varepsilon$ $\{t = mktable(null); push(t, tblptr); push(0, offset);\}$

$D \rightarrow proc\ id\ ND; S$ $\{t = top(tblptr); addwidth\big(t, top(offset)\big);$
$pop(tblptr); pop(offset);$
$enterproc(top(tblptr), top(offset));\}$

$D \rightarrow id\ L$ $\{enter(top(tblptr), id.name, L.type, top(offset));$
$top(offset) = top(offset) + L.width\}$

$L \rightarrow, id\ L_1$ $\{enter(top(tblptr), id.name, L_1.type, top(offset));$
$top(offset) = top(offset) + L_1.width;$
$L.type = L_1.type, L.width = L_1.width\}$

$N \rightarrow \varepsilon$ $\{t = mktable(top(tblptr)); push(t, tblptr); push(0, offset);\}$

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.3.1 简单算术表达式及赋值语句

- 简单算术表达式及赋值语句的开始符号为S，来自如下部分的相应V_N：

$P \rightarrow MD$

$M \rightarrow \varepsilon$

$D \rightarrow id\ L \mid proc\ id\ ND; S$

$L \rightarrow, id\ L_1$

$N \rightarrow \varepsilon$

7.3.1 简单算术表达式及赋值语句

- 简单算术表达式及赋值语句的操作：
  - $tblptr$：是一个栈，栈顶为当前过程的符号表，所以可以取到符号信息。
  - $lookup(name)$：从$top(tblptr)$符号表寻找名字，找到即返回，找不到则转到外围（上层）符号表继续查找，直到找到或者所有外围过程都找不到为止。
  - $gen(op, arg1, arg2, result)$：生成三地址代码。
  - $newtemp$：是一个方法，生成一个临时变量。
  - $place$：是一个属性，存放文法符号的值（变量）的名字。

简单算术表达式及赋值语句：

$S \rightarrow id = E$ $\{p = lookup(id.name);$
$if\ p \neq null\ then\ gen(=, E.place, -, p); else\ error;\}$

$E \rightarrow E_1 + E_2$ $\{E.place = newtemp; gen(+, E_1.place, E_2.place, E.place)\}$

$E \rightarrow E_1 * E_2$ $\{E.place = newtemp; gen(*, E_1.place, E_2.place, E.place)\}$

$E \rightarrow -E_1$ $\{E.place = newtemp; gen(@, E_1.place, -, E.place)\}$

$E \rightarrow (E_1)$ $\{E.place = E_1.place\}$

$E \rightarrow id$ $\{p = lookup(id.name);$
$if\ p \neq null\ then\ E.place = p; else\ error;\}$

过程中的说明语句

$S \rightarrow id = E \quad \{p = lookup(id.name);$

$if\ p \neq null\ then\ gen(=, E.place, -, p);\ else\ error;\}$

$E.place)\}$

步骤	文法符号栈	输入串	动作
1	#	$x = (a + b) * -c\#$	初始
2	$\#x = (a$	$+b) * -c\#$	移进
3	$\#x = (E$	$+b) * -c\#$	归约
4	$\#x = (E + b$	$) * -c\#$	移进
5	$\#x = (E + E$	$) * -c\#$	归约
6	$\#x = (E$	$) * -c\#$	归约
7	$\#x = (E)$	$* -c\#$	移进
8	$\#x = E$	$* -c\#$	归约
9	$\#x = E * -c$	#	移进
10	$\#x = E * -E$	#	归约
11	$\#x = E * E$	#	归约
12	$\#x = E$	#	归约
13	$\#S$	#	归约
14	$\#S$	#	成功

三地址码

$(+, a, b, T1)$
$(@, c, -, T2)$
$(*, T1, T2, T3)$
$(=, T3, -, x)$

$E.place = T2$
$E.place = T3$

栈属性
(给人看的)

第七章 语义分析和中间代码生成

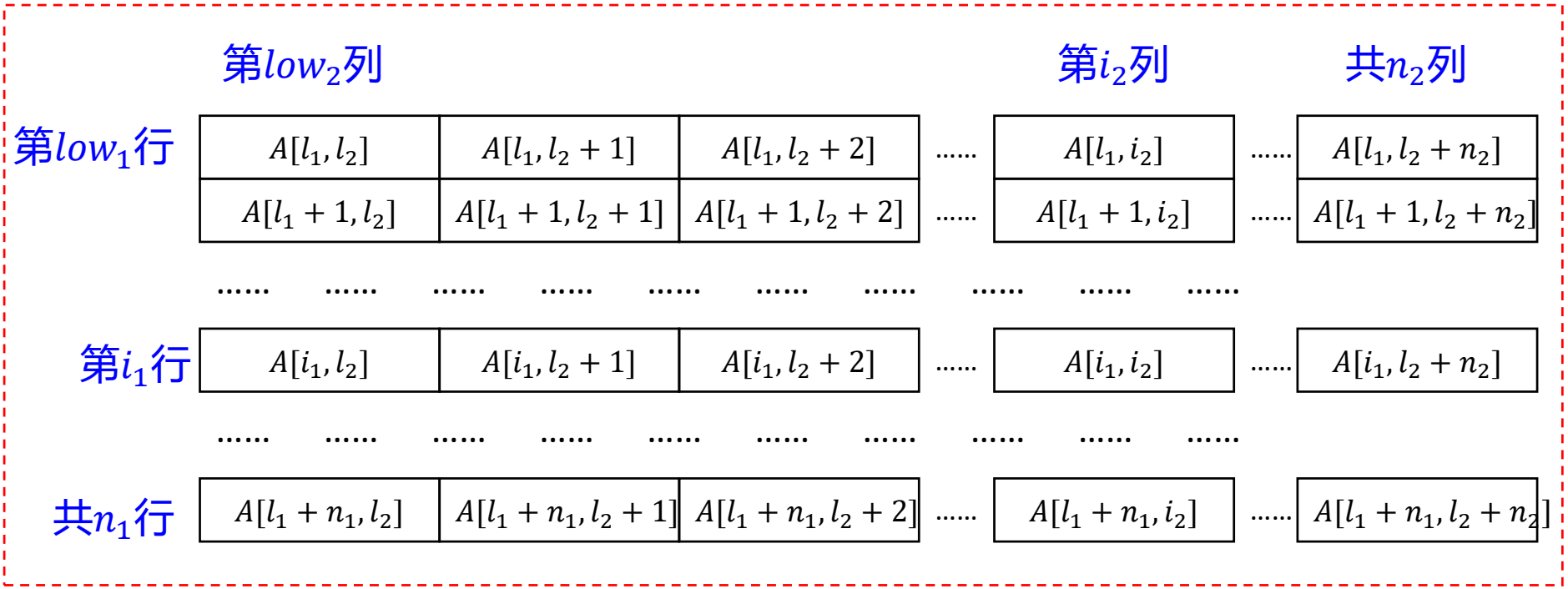
- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.3.2 数组元素的引用

- 数组连续存储，一维数组$A[i]$地址为：$base + (i - low) \times w$
  - w：数组中每个元素的宽度；
  - low：数组下标下界；
  - $base$：分配给数组的相对地址，即$base$为A的第一个元素$A[low]$的相对地址。
- 整理为：$i \times w + (base - low \times w)$
  - 令$C = base - low \times w$，可以在处理数组声明时计算出来，存放到符号表中A的对应项中；
  - $A[i]$的相对地址计算变为：$i \times w + C$。

7.3.2 数组元素的引用

- 行存储的二维数组$A[i_1, i_2]$地址：
  - $base + [(i_1 - low_1) \times n_2 + i_2 - low_2] \times w$
  - $= (i_1 \times n_2 + i_2) \times w + [base - (low_1 \times n_2 + low_2) \times w]$



7.3.2 数组元素的引用

- 行存储的二维数组$A[i_1, i_2]$地址：
  - $base + [(i_1 - low_1) \times n_2 + i_2 - low_2] \times w$
  - $= (i_1 \times n_2 + i_2) \times w + [base - (low_1 \times n_2 + low_2) \times w]$
- 行存储的多维数组$A[i_1, i_2, \dots, i_k]$地址：
  - 基准地址：$C = base - ((\dots((low_1 \times n_2 + low_2) \times n_3)\dots) \times n_k + low_k) \times w$
  - 动态地址：$(((\dots(i_1 \times n_2 + i_2) \times n_3)\dots) \times n_k + i_k) \times w$

7.3.2 数组元素的引用

- 生成数组的文法：

  $L \rightarrow id\ [Elist] \mid id$

  $Elist \rightarrow Elist, E \mid E$

- 要想知道数组的全部信息，需要有一个产生式把id（提供符号表地址）和最左下标E（提供下标值）联系起来，因此修改文法如下：

  $L \rightarrow Elist] \mid id$

  $Elist \rightarrow Elist, E \mid id[E$

7.3.2 数组元素的引用

- 最终文法：

① $S \to L = E$

② $E \to E + E$

③ $E \to E * E$

④ $E \to -E$

⑤ $E \to (E)$

⑥ $E \to L$ // E也可以是数组，如果是变量，也需要通过L过渡

⑦ $L \to Elist]$ // L是数组

⑧ $L \to id$ // L是普通变量

⑨ $Elist \to Elist, E$

⑩ $Elist \to id[E$

7.3.2 数组元素的引用

- $Elist$的属性：
  - ① $array$，记录指向符号表中相应数组名字表项的指针。
  - ② $ndim$，记录$Elist$中下标表达式的个数，即维数。
  - ③ $place$，表示临时变量，用来临时存放由$Elist$中的下标表达式计算出来的值。
- L的属性：
  - $place$，指向符号表中相应此名字表项的指针。
  - $offset$，简单名字为$null$，数组则为地址偏移量。
- 函数：
  - $limit(array, j)$，返回n_j，即由$array$所指示的数组，其第j维的长度。
- 多维数组$A[i_1, i_2, \dots, i_k]$的前m维下标：
  - 动态下标：$(\dots((i_1 \times n_2 + i_2) \times n_3) \dots) \times n_m + i_m$
  - 递归计算：$e_1 = i_1, e_2 = e_1 \times n_2 + i_2, \dots, e_m = e_{m-1} \times n_m + i_m$

翻译模式

(1) $S \rightarrow L = E$

{ if $L.offset = null$ // L是简单变量

$gen(=, E.place, -, L.place)$;

else // L是数组

$gen(=, E.place, -, L.place[L.offset])$; }

(2) $E \rightarrow E_1 + E_2$

{ $E.place = newtemp$;

$gen(+, E_1.place, E_2.place, E.place)$; }

(3) $E \rightarrow E_1 * E_2$

{ $E.place = newtemp$;

$gen(*, E_1.place, E_2.place, E.place)$; }

翻译模式

(4) $E \rightarrow -E_1$

{ $E.place = newtemp$;

$gen(@, E_1.place, -, E.place)$; }

(5) $E \rightarrow (E_1)$

{ $E.place = E_1.place$; }

(6) $E \rightarrow L$

{ if $L.offset = null$ // L是简单变量

$E.place = L.place$;

else { // L是数组，转到E后会丢失数组信息，因此此处赋值

$E.place = newtemp$;

$gen(=, L.place[L.offset], -, E.place)$; } }

翻译模式

(7) $L \rightarrow Elist]$

{ $L.place = newtemp;$

$gen(+, Elist.array, C, L.place);$ // C的计算参考前述公式，在符号表中

$L.offset = newtemp;$

$gen(*, w, Elist.place, L.offset);$ } // w在符号表中

(8) $L \rightarrow id$

{ $L.palce = id.place;$

$L.offset = null;$ }

翻译模式

(9) $Elist \rightarrow Elist_1, E$

$\{ t = newtemp;$

$m = Elist.ndim + 1;$ // 用一次维度+1

$gen(*, Elist_1.place, limit(Elist_1.array, m), t);$

$gen(+, t, E.place, t);$ // 递归计算$e_k = e_{k-1} \times n_k + i_k$

$Elist.array = Elist_1.array;$

$Elist.palce = t;$

$Elist.ndim = m; \}$

(10) $Elist \rightarrow id[E$

$\{ Elist.palce = E.place;$

$Elist.ndim = 1;$

$Elist.array = id.place; \}$

【例7.10】赋值语句$A[10,20]: integer;\ x = A[y,z];$

步骤	文法符号栈	输入串	动作
1	#	$x = A[y,z]\#$	初始
2	$\#x$	$= A[y,z]\#$	移进
3	$\#L$	$= A[y,z]\#$	归约
4	$\#L = A[y$	$,z]\#$	移进
5	$\#L = A[L$	$,z]\#$	归约
6	$\#L = A[E$	$,z]\#$	归约
7	$\#L = Elist$	$,z]\#$	归约
8	$\#L = Elist, z$	$]\#$	移进
9	$\#L = Elist, L$	$]\#$	归约
10	$\#L = Elist, E$	$]\#$	归约
11	$\#L = Elist$	$]\#$	归约
12	$\#L = Elist]$	#	移进
13	$\#L = L$	#	归约
14	$\#L = E$	#	归约
15	$\#S$	#	归约
16	$\#S$	#	成功

三地址码

$(*, y, 20, T1)$
$(+, T1, z, T1)$
$(+, A, 24, T2)$
$(*, 4, T1, T3)$
$(=, T2[T3], -, T4)$
$(=, T4, -, x)$

(9) $Elist \rightarrow Elist_1, E$

{ $t = newtemp;$

(1) $S \rightarrow L = E$

{ if $L.offset = null$

$gen(=, E.place, -, L.place);$

else

$gen(=, E.place, -, L.place[L.offset]);$ }

$E.place = z$
$E.place = T4$
$L.place = x, .offset = null$

栈属性（给人看的）

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.3.3 类型转换

- 当两个不同类型的量运算时，需要二选一：
  - 拒绝运算
  - 自动进行类型转换

【例7.11】$x = y + i * j$，其中i, j为整型，y为实型，对应四元式为：

① $(*, i, j, T1)$

② $(int2real, T1, null, T2)$

③ $(+, y, T2, T3)$

④ $(=, T3, null, x)$

翻译模式

$E \rightarrow E_1\ \theta\ E_2$

{ $E.place = newtemp;$

if $E_1.type == integer$ && $E_2.type == integer$ {

$gen(\theta^i, E_1.place, E_2.place, E.place); E.type = integer;$ }

else if $E_1.type == real$ && $E_2.type == real$ {

$gen(\theta^r, E_1.place, E_2.place, E.place); E.type = real;$ }

else if $E_1.type == integer$ && $E_2.type == real$ {

$u = newtemp; gen(int2real, E_1.place, -, u);$

$gen(\theta^r, u, E_2.place, E.place); E.type = real;$ }

else if $E_1.type == real$ && $E_2.type == integer$ {

$u = newtemp; gen(int2real, E_2.place, -, u);$

$gen(\theta^r, E_1.place, u, E.place); E.type = real;$ }

else $E.type = type\_error;$ }

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.4 布尔表达式的翻译

- 布尔表达式的作用
  - 作为控制语句的条件式；
  - 作为逻辑运算，获得逻辑值。
- 算符优先级的说明：
  - 优先级由高到低：¬,∧,∨；
  - ∧,∨服从左结合规则，¬服从右结合规则；
  - 关系表达式形如$E_1 \theta E_2$，其中关系符θ包括$<, \leq, =, \neq, >, \geq$，$E_i$为算术表达式；
  - 各关系符优先级相同，高于布尔算符，低于算术算符；
  - 关系符不得结合，如$a < b < c$为非法。

7.4 布尔表达式的翻译

- 计算方法1：如同算术表达式，一步不差的从表达式各部分值计算整个表达式的值
  - $1 \vee (\neg 0 \wedge 0) = 1 \vee (1 \wedge 0) = 1 \vee 0 = 1$
- 计算方法2：优化算法
  - $A \vee B: if\ A\ then\ true\ else\ B$
  - $A \wedge B: if\ A\ then\ B\ else\ false$
  - $\neg A: if\ A\ then\ false\ else\ true$

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.4.1 数值表示法

【例7.12】布尔表达式生成四元式：$a \vee b \wedge \neg c$

① $(\neg, c, -, T1)$

② $(\wedge, b, T1, T2)$

③ $(\vee, a, T2, T3)$

【例7.13】关系式$a < b$可以看作：$if\ a < b\ then\ 1\ else\ 0$

100. $(j <, a, b, 103)$

101. $(=, 0, -, T1)$

102. $(j, -, -, 104)$

103. $(=, 1, -, T1)$

104. ...

翻译模式

- $nextstat$：为将要生成尚未生成的四元式地址索引，每生成一条自动加1

① $E \to E_1 \vee E_2$ $\{E.place = newtemp; gen(\vee, E_1.place, E_2.place, E.place);\}$

② $E \to E_1 \wedge E_2$ $\{E.place = newtemp; gen(\wedge, E_1.place, E_2.place, E.place);\}$

③ $E \to \neg E_1$ $\{E.place = newtemp; gen(\neg, E_1.place, -, E.place);\}$

④ $E \to (E_1)$ $\{E.place = E_1.place;\}$

⑤ $E \to id$ $\{E.place = id.place;\}$

⑥ $E \to id_1\ \theta\ id_2$ $\{E.place = newtemp;$

$gen(j\theta, id_1.place, id_2.place, nextstat + 3);$

$gen(=, 0, -, E.place);$

$gen(j, -, -, nextstat + 2);$

$gen(=, 1, -, E.place);\}$

【例7.14】布尔表达式：$a < b \vee c \leq d \wedge e > f$

$E \rightarrow id_1\ \theta\ id_2$ $\quad \{E.place = newtemp;$
$E \rightarrow E_1 \vee E_2$ $\quad \{E.place = newtemp;$ $gen(\vee, E_1.place, E_2.place, E.place);\}$

步骤	文法符号栈	输入串	动作
1	#	$a < b \vee c \leq d \wedge e > f$#	初始
2	#$a < b$	$\vee c \leq d \wedge e > f$#	移进
3	#E	$\vee c \leq d \wedge e > f$#	归约
4	#$E \vee c \leq d$	$\wedge e > f$#	移进
5	#$E \vee E$	$\wedge e > f$#	归约
6	#$E \vee E \wedge e > f$	#	移进
7	#$E \vee E \wedge E$	#	归约
8	#$E \vee E$	#	归约
9	#E	#	归约
10	#E	#	成功

三地址码

100: $(j <, a, b, 103)$
101: $(=, 0, -, T1)$
102: $(j, -, -, 104)$
103: $(=, 1, -, T1)$
104: $(j \leq, c, d, 107)$
105: $(=, 0, -, T2)$
106: $(j, -, -, 108)$
107: $(=, 1, -, T2)$
108: $(j >, e, f, 111)$
109: $(=, 0, -, T3)$
110: $(j, -, -, 112)$
111: $(=, 1, -, T2)$
112: $(\wedge, T2, T3, T4)$
113: $(\vee, T1, T4, T5)$

$E.place = T3$
$E.place = T4$
$E.place = T5$

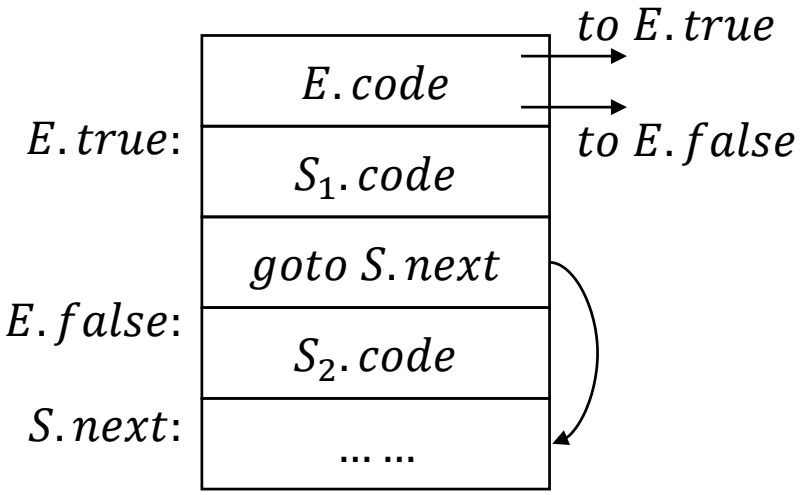
栈属性

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.4.2 作为条件控制的布尔式翻译

- 条件语句$if\ E\ then\ S_1\ else\ S_2$中的布尔表达式$E$，仅用于控制对$S_1$和$S_2$的选择，不需要同一个临时变量保留其值，因此可以为布尔式E设置两个出口：
  - 真出口：指向S_1的第一个四元式；
  - 假出口：指向S_2的第一个四元式。



7.4.2 作为条件控制的布尔式翻译

【例7.15】$if\ a > c \vee b < d\ then\ S_1\ else\ S_2$

这两个语句都跳转到L_2，但L_2需要分析到S_1才能确定，其它类似。

$if\ a > c\ goto\ L_2$

$goto\ L_1$

L_1: $if\ b < d\ goto\ L_2$

$goto\ L_3$

L_2: (关于S_1的代码序列)

$goto\ L_{next}$

L_3: (关于S_2的代码序列)

L_{next}:

100. $(j >, a, c, 104)$

101. $(j, -, -, 102)$

102. $(j <, b, d, 104)$

103. $(j, -, -, 116)$

104. S_1...

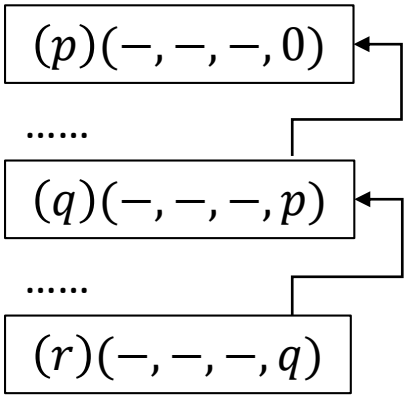
115. $(j, -, -, 120)$

116. S_2 ...

120. ...

7.4.2 作为条件控制的布尔式翻译

- 思路：
  - 生成四元式时，暂不确定跳转标号，而是把指向同一目标的四元式组成一个链表，确定目标后再回填；
  - 为非终结符号E赋予两个综合属性$E.truelist$和$E.falselist$，分别记录真、假出口链；
  - 需要回填的四元式，借助第四区块构造真、假出口链。

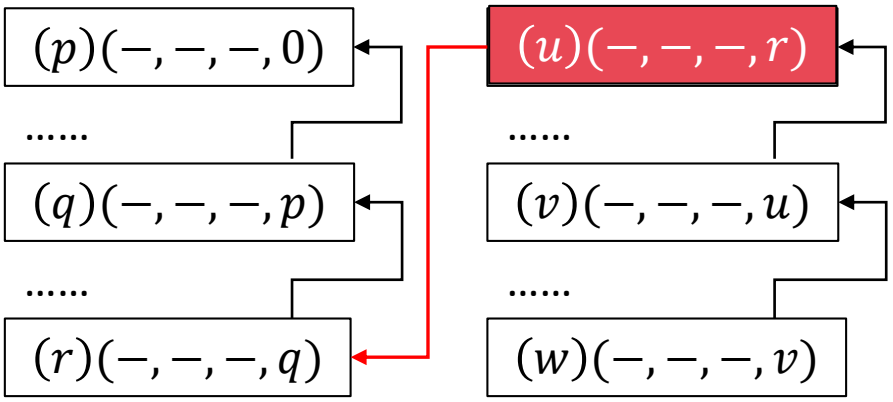


- 用到四元式：
  - $(jnz, a, -, p)$: $if\ a\ goto\ p$
  - $(j\theta, x, y, p)$: $if\ x\ \theta\ y\ goto\ p$
  - $(j, -, -, p)$: $goto\ p$

7.4.2 作为条件控制的布尔式翻译

- 需要用到的变量或函数：
  - nxq：即Next Quadruplet，指向下一条将要产生，但尚未产生的四元式地址（标号），初值为1，执行一次gen增1。
  - $mklist(i)$：创建一个链表，这个链表仅含标号为i的四元式，并返回链表指针。
  - $merge(p1,p2)$：把以$p1$和$p2$为链首的两条链合并，返回新的链首。当遇到$E_1 \wedge E_2$时，它们的$falselist$链表需要合并；当遇到$E_1 \vee E_2$时，它们的$turelist$链表需要合并。
  - $backpatch(p,t)$：回填，把p所链接的每个四元式的第四区段都填t。当遇到$E_1 \wedge E_2$时， E_2 确定了E_1的$truelist$链表需要回填的地址；当遇到$E_1 \vee E_2$时， E_2确定了E_1的$falselist$链表需要回填的地址。

Merge操作



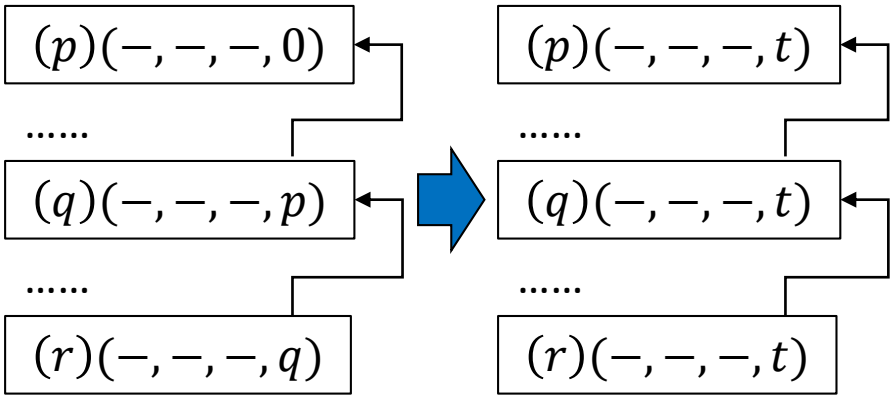
$E_1.truelist = r$
或
$E_1.falselist = r$

$E_2.truelist = w$
或
$E_2.falselist = w$

$E.truelist = w$
或
$E.falselist = w$

```
Quad * Merge(void *p1, void *p2)
{
    if (p1 == null) return p2;
    if (p2 == null) return p1;
    // 找p2链的链尾
    Quad *p = p2;
    while (p的第四区段内容 ≠ null)
        p = p的第四区段内容;
    p的第四区段内容 = p1;
    return p2;
}
```

backpatch操作



$E_1.truelist = r$
或
$E_1.falselist = r$

```
void backpatch(Quad *p, Quad *t)
{
    Quad *q;
    while (p ≠ null)
    {
        q = p的第四区段内容;
        p的第四区段内容 = t;
        p = q;
    }
}
```

翻译模式

(1) $E \to E_1 \vee M E_2$ $\{backpatch(E_1.falselist, M.quad);$

$E.truelist = merge(E_1.truelist, E_2.truelist);$

$E.falselist = E_2.falselist;\}$

(2) $E \to E_1 \wedge M E_2$ $\{backpatch(E_1.truelist, M.quad);$

$E.falselist = merge(E_1.falselist, E_2.falselist);$

$E.truelist = E_2.truelist;\}$

(3) $M \to \varepsilon$ $\{M.quad = nxq;\}$ // 在E_2之前把它记下来，E_2之后使用。

(4) $E \to \neg E_1$ $\{E.truelist = E_1.falselist;$

$E.falselist = E_2.truelist;\}$

(5) $E \to (E_1)$ $\{E.truelist = E_1.truelist;$

$E.falselist = E_2.falselist;\}$

翻译模式

(6) $E \to id_1\ \theta\ id_2$ $\{E.truelist = mklist(nxq);$

$E.falselist = mklist(nxq + 1);$

$gen(j\theta, id_1.place, id_2.place, 0);$

$gen(j, -, -, 0);\}$

(7) $E \to id$ $\{E.truelist = mklist(nxq);$

$E.falselist = mklist(nxq + 1);$

$gen(jnz, id.place, -, 0);$

$gen(j, -, -, 0);\}$

下面通过构造LR(0)分析表确定文法的归约顺序。

(0) $E' \to E$ (1) $E \to E \vee ME$ (2) $E \to E \wedge ME$ (3) $M \to \varepsilon$

(4) $E \to \neg E$ (5) $E \to (E)$ (6) $E \to id\ \theta\ id$ (7) $E \to id$

I_0: $E' \to \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_7 = Go(I_2, E)$:
$E \to \neg E \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_{11} = Go(I_6, M)$:
$E \to E \wedge M \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_1 = Go(I_0, E)$:
$E' \to E \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_4 = Go(I_0, id)$:
$E \to id \cdot \theta\ id$
$E \to id \cdot$

$I_2 = Go(I_2, \neg)$:

$I_3 = Go(I_2, ()$:

$I_4 = Go(I_2, id)$:

$I_5 = Go(I_7, \vee)$:

$I_6 = Go(I_7, \wedge)$:

$I_{12} = Go(I_8,))$:
$E \to (E) \cdot$

$I_5 = Go(I_8, \vee)$:

$I_6 = Go(I_8, \wedge)$:

$I_{13} = Go(I_9, id)$:
$E \to id\ \theta\ id \cdot$

$I_2 = Go(I_0, \neg)$:
$E \to \neg \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_8 = Go(I_3, E)$:
$E \to (E \cdot)$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_{14} = Go(I_{10}, E)$:
$E \to E \vee ME \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_2 = Go(I_{10}, \neg)$:

$I_3 = Go(I_{10}, ()$:

$I_4 = Go(I_{10}, id)$:

$I_3 = Go(I_0, ()$:
$E \to (\cdot E)$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_2 = Go(I_3, \neg)$:

$I_3 = Go(I_3, ()$:

$I_4 = Go(I_3, id)$:

$I_{15} = Go(I_{11}, E)$:
$E \to E \wedge ME \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_2 = Go(I_{11}, \neg)$:

$I_3 = Go(I_{11}, ()$:

$I_4 = Go(I_{11}, id)$:

$I_5 = Go(I_1, \vee)$:
$E \to E \vee \cdot ME$
$M \to \cdot$

$I_6 = Go(I_1, \wedge)$:
$E \to E \wedge \cdot ME$
$M \to \cdot$

$I_9 = Go(I_4, \theta)$:
$E \to id\ \theta \cdot id$

$I_{10} = Go(I_5, M)$:
$E \to E \vee M \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_5 = Go(I_{14}, \vee)$:

$I_6 = Go(I_{14}, \wedge)$:

$I_5 = Go(I_{15}, \vee)$:

$I_6 = Go(I_{15}, \wedge)$:

(0) $E' \to E$　(1) $E \to E \vee ME$　(2) $E \to E \wedge ME$　(3) $M \to \varepsilon$

(4) $E \to \neg E$　(5) $E \to (E)$　(6) $E \to id\ \theta\ id$　(7) $E \to id$

	Action								Goto	
状态	¬	∧	∨	θ	id	(	)	#	E	M
0					S_4				1	
1								acc		
2	S_2				S_4	S_3			7	
3										
4	r_7	r_7	r_7	r_4	r_7	r_7	r_7	r_7		
5										
6										11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10										
11										
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14										
15										

$I_0: E' \to \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_1 = Go(I_0, E)$:
$E' \to E \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_2 = Go(I_2, \neg)$:

$I_3 = Go(I_2, ()$:

$I_4 = Go(I_2, id)$:

$I_4 = Go(I_0, id)$:
$E \to id \cdot \theta\ id$
$E \to id \cdot$

$I_5 = Go(I_7, \vee)$:

$I_6 = Go(I_7, \wedge)$:

$I_7 = Go(I_2, E)$:
$E \to \neg E \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_{11} = Go(I_6, M)$:
$E \to E \wedge M \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_{12} = Go(I_8,))$:
$E \to (E) \cdot$

$I_5 = Go(I_8, \vee)$:

$I_6 = Go(I_8, \wedge)$:

$I_{13} = Go(I_9, id)$:
$E \to id\ \theta\ id \cdot$

(0) $E' \to E$　(1) $E \to E \vee ME$　(2) $E \to E \wedge ME$　(3) $M \to \varepsilon$

(4) $E \to \neg E$　(5) $E \to (E)$　(6) $E \to id\ \theta\ id$　(7) $E \to id$

	Action								Goto	
状态	¬	∧	∨	θ	id	(	)	#	E	M
0	S_2				S_4	S_3			1	
1								acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	r_4	r_7	r_7	r_7	r_7		
5										
6										11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

$I_2 = Go(I_0, \neg)$:
$E \to \neg \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_3 = Go(I_0, ($:
$E \to (\cdot E)$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_8 = Go(I_3, E)$:
$E \to (E \cdot)$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_2 = Go(I_3, \neg)$:

$I_3 = Go(I_3, ($:

$I_4 = Go(I_3, id)$:

$I_{14} = Go(I_{10}, E)$:
$E \to E \vee ME \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_{15} = Go(I_{11}, E)$:
$E \to E \wedge ME \cdot$
$E \to E \cdot \vee ME$
$E \to E \cdot \wedge ME$

$I_2 = Go(I_{10}, \neg)$:

$I_2 = Go(I_{11}, \neg)$:

$I_3 = Go(I_{10}, ($:

$I_3 = Go(I_{11}, ($:

$I_4 = Go(I_{10}, id)$:

$I_4 = Go(I_{11}, id)$:

(0) $E' \to E$ (1) $E \to E \vee ME$ (2) $E \to E \wedge ME$ (3) $M \to \varepsilon$

(4) $E \to \neg E$ (5) $E \to (E)$ (6) $E \to id\ \theta\ id$ (7) $E \to id$

	Action								Goto	
状态	¬	∧	∨	θ	id	(	)	#	E	M
0	S_2				S_4	S_3			1	
1		S_6	S_5					acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	S_9	r_7	r_7	r_7	r_7		
5	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		10
6	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	S_6	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

$I_5 = Go(I_1, \vee)$:
$E \to E \vee \cdot ME$
$M \to \cdot$

$I_6 = Go(I_1, \wedge)$:
$E \to E \wedge \cdot ME$
$M \to \cdot$

$I_9 = Go(I_4, \theta)$:
$E \to id\ \theta \cdot id$

$I_{10} = Go(I_5, M)$:
$E \to E \vee M \cdot E$
$E \to \cdot E \vee ME$
$E \to \cdot E \wedge ME$
$E \to \cdot \neg E$
$E \to \cdot (E)$
$E \to \cdot id\ \theta\ id$
$E \to \cdot id$

$I_5 = Go(I_{14}, \vee)$:

$I_6 = Go(I_{14}, \wedge)$:

$I_5 = Go(I_{15}, \vee)$:

$I_6 = Go(I_{15}, \wedge)$:

(0) $E' \to E$　(1) $E \to E \vee ME$　(2) $E \to E \wedge ME$　(3) $M \to \varepsilon$

(4) $E \to \neg E$　(5) $E \to (E)$　(6) $E \to id\ \theta\ id$　(7) $E \to id$

	Action								Goto	
状态	¬	∧	∨	θ	id	(	)	#	E	M
0	S_2				S_4	S_3			1	
1		S_6	S_5					acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	S_9	r_7	r_7	r_7	r_7		
5	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		10
6	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	S_6	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

步骤	状态/符号栈	输入串
1	0 $\#$	$a < b \vee c \leq d \wedge e\#$
2	04 $\#a$	$< b \vee c \leq d \wedge e\#$
3	049 $\#a <$	$b \vee c \leq d \wedge e\#$
4	04913 $\#a < b$	$\vee c \leq d \wedge e\#$
5	01 $\#E$	$\vee c \leq d \wedge e\#$
6	015 $\#E \vee$	$c \leq d \wedge e\#$
7	01510 $\#E \vee M$	$c \leq d \wedge e\#$
8	015104 $\#E \vee Mc$	$\leq d \wedge e\#$
9	0151049 $\#E \vee Mc \leq$	$d \wedge e\#$
10	01510491 3 $\#E \vee Mc \leq d$	$\wedge e\#$
11	01510 14 $\#E \vee ME$	$\wedge e\#$

(0) $E' \to E$　(1) $E \to E \vee ME$　(2) $E \to E \wedge ME$　(3) $M \to \varepsilon$

(4) $E \to \neg E$　(5) $E \to (E)$　(6) $E \to id\ \theta\ id$　(7) $E \to id$

	Action								Goto	
状态	¬	∧	∨	θ	id	(	)	#	E	M
0	S_2				S_4	S_3			1	
1		S_6	S_5					acc	7	
2	S_2				S_4	S_3				
3	S_2				S_4	S_3			8	
4	r_7	r_7	r_7	S_9	r_7	r_7	r_7	r_7		
5	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		10
6	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		11
7	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
8		S_6	S_5				S_{12}			
9					S_{13}					
10	S_2				S_4	S_3			14	
11	S_2				S_4	S_3			15	
12	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5		
13	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		
14	r_1	S_6	r_1	r_1	r_1	r_1	r_1	r_1		
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		

步骤	状态/符号栈	输入串
11	015<u>10</u> <u>14</u> #$E \vee ME$	$\wedge e$#
12	015<u>10</u> <u>14</u> 6 #$E \vee ME \wedge$	e#
13	015<u>10</u> <u>14</u> 6<u>11</u> #$E \vee ME \wedge M$	e#
14	015<u>10</u> 146<u>11</u>4 #$E \vee ME \wedge Me$	#
15	015<u>10</u> 14<u>6</u><u>11</u> <u>15</u> #$E \vee ME \wedge ME$	#
16	015<u>10</u> <u>14</u> #$E \vee ME$	#
17	01 #E	#
18	01成功 #E	#

【例7.15】布尔表达式：$a < b \vee c \leq d \wedge e$，假设$nxq = 100$

$E \to id \quad \{E.truelist = mklist(nxq);$

$E \to E_1 \vee ME_2 \quad \{backpatch(E_1.falselist, M.quad);$
$E.truelist = merge(E_1.truelist, E_2.truelist);$
$E.falselist = E_2.falselist;\}$

步骤	文法符号栈	输入串	动作
1	#	$a < b \vee c \leq d \wedge e\#$	初始
2	$\#a < b$	$\vee c \leq d \wedge e\#$	移进
3	$\#E$	$\vee c \leq d \wedge e\#$	归约
4	$\#E \vee$	$c \leq d \wedge e\#$	移进
5	$\#E \vee M$	$c \leq d \wedge e\#$	归约
6	$\#E \vee Mc \leq d$	$\wedge e\#$	移进
7	$\#E \vee ME$	$\wedge e\#$	归约
8	$\#E \vee ME \wedge$	$e\#$	移进
9	$\#E \vee ME \wedge M$	$e\#$	归约
10	$\#E \vee ME \wedge Me$	#	移进
11	$\#E \vee ME \wedge ME$	#	归约
12	$\#E \vee ME$	#	归约
13	$\#E$	#	归约
14	$\#E$	#	成功

三地址码

100: $(j <, a, b, 0)$
101: $(j, -, -, 102)$
102: $(j \leq, c, d, 104)$
103: $(j, -, -, 0)$
104: $(jnz, e, -, 100)$
105: $(j, -, -, 103)$

两个未填充四元式链，需要等到确定布尔式为真做什么、为假做什么时才能回填。

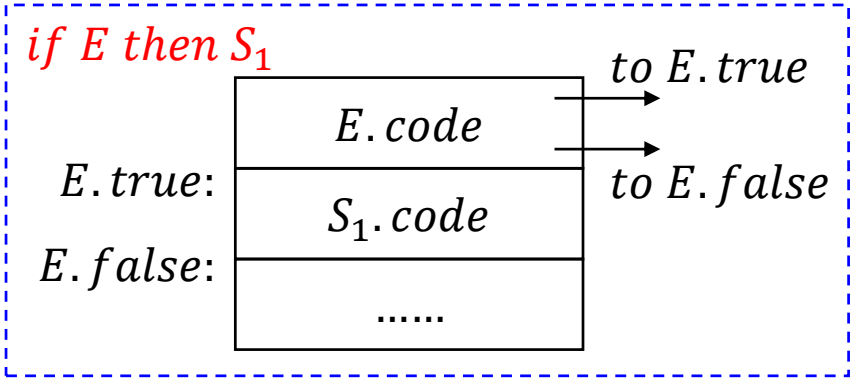
$E.truelist = 104, E.falselist = 105$
$M.quad = 104$
$E.truelist = 104, E.falselist = 105$
$M.quad = 102$
$E.truelist = 104, E.falselist = 105$

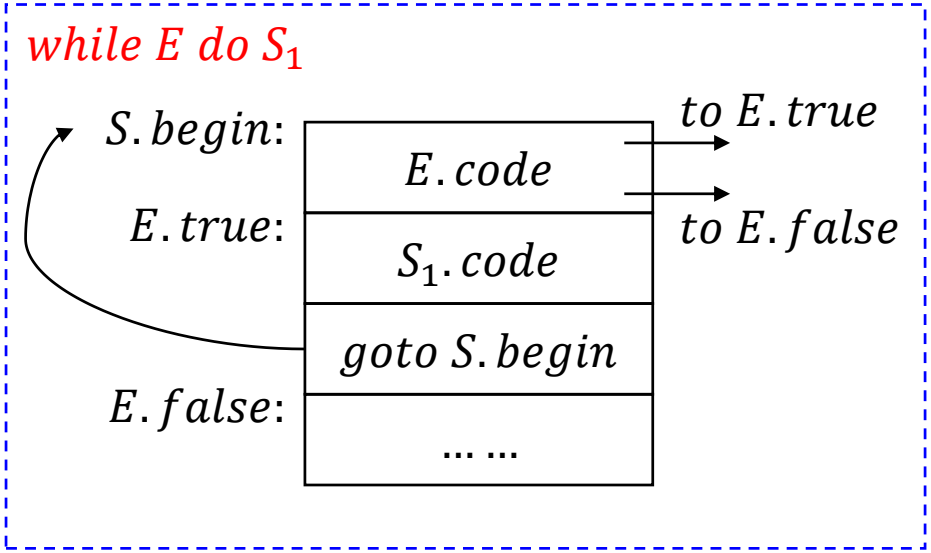
第七章 语义分析和中间代码生成

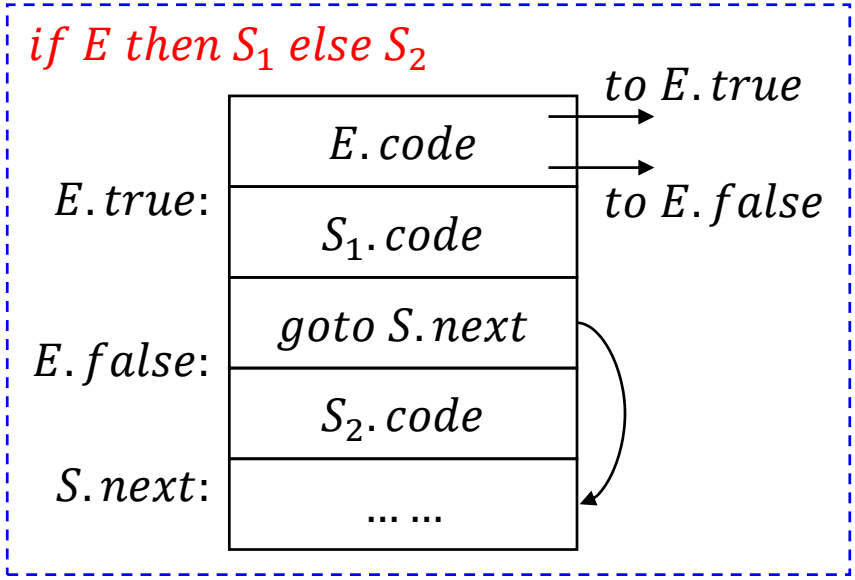
- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.5.1 控制流语句

- 控制流语句：
  - $S \rightarrow if\ E\ then\ S_1 \mid if\ E\ then\ S_1\ else\ S_2 \mid while\ E\ do\ S_1$







7.5.1 控制流语句

- 控制流语句的完整文法：

① $S \rightarrow if\ E\ then\ S$

② $\quad | if\ E\ then\ S\ else\ S$

③ $\quad | while\ E\ do\ S$

④ $\quad | begin\ L\ end$

⑤ $\quad | A$ // 赋值语句，对应7.3节的S

⑥ $L \rightarrow L; S$ // 语句表

⑦ $\quad | S$ // 语句

- 新的属性$S.nextlist$和$L.nextlist$：
  - 表示紧接语句$S(L)$之后要执行的语句。

翻译模式

(1) $S \rightarrow if\ E\ then\ M_1 S_1 N\ else\ M_2 S_2$

$\{backpatch(E.truelist, M_1.quad);$

$backpatch(E.falselist, M_2.quad);$

$S.nextlist = merge(S_1.nextlist, N.nextlist, S_2.nextlist);\ \}$

(2) $M \rightarrow \varepsilon$ $\{M.quad = nxq;\}$ // 在S之前把它记下来，S之后使用。

(3) $N \rightarrow \varepsilon$ $\{N.nextlist = mklist(nxq);$

$gen(j, -, -, 0);\ \}$ // 跳到S_2之后，也就是整个语句之后

(4) $S \rightarrow if\ E\ then\ M S_1$

$\{backpatch(E.truelist, M.quad);$

$S.nextlist = merge(E.falselist, S_1.nextlist);\ \}$

翻译模式

(5) $S \rightarrow while\ M_1 E\ do\ M_2 S_1$

$\{backpatch(S_1.nextlist, M_1.quad);$

$backpatch(E.truelist, M_2.quad);$

$S.nextlist = E.falselist;$

$gen(j, -, -, M_1.quad);\ \}$

(6) $S \rightarrow begin\ L\ end$ $\{S.nextlist = L.nextlist;\ \}$

(7) $S \rightarrow A$ $\{S.nextlist = mklist();\ \}$ // 初始化为空表

(8) $L \rightarrow L_1; MS$ $\{backpatch(L_1.nextlist, M.quad);$ // L_1的结束是S的开始

$L.nextlist = S.nextlist;\ \}$

(9) $L \rightarrow S$ $\{L.nextlist = S.nextlist;\ \}$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

I_0: $S' \to \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_1 = Go(I_0, S)$
$S' \to S \cdot$

$I_2 = Go(I_0, i)$
$S \to i \cdot EtMSNeMS$
$S \to i \cdot EtMS$

$I_3 = Go(I_0, w)$
$S \to w \cdot MEdMS$
$M \to \cdot$

$I_4 = Go(I_0, \{)$
$S \to \{ \cdot L\}$
$L \to \cdot L;MS$
$L \to \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_5 = Go(I_0, A)$
$S \to A \cdot$

$I_6 = Go(I_2, E)$
$S \to iE \cdot tMSNeMS$
$S \to iE \cdot tMS$

$I_7 = Go(I_3, M)$
$S \to wM \cdot EdMS$

$I_8 = Go(I_4, L)$
$S \to \{L \cdot\}$
$L \to L \cdot ;MS$

$I_9 = Go(I_4, S)$
$L \to S \cdot$

$I_2 = Go(I_4, i)$

$I_3 = Go(I_4, w)$

$I_4 = Go(I_4, \{)$

$I_5 = Go(I_4, A)$

$I_{10} = Go(I_6, t)$
$S \to iEt \cdot MSNeMS$
$S \to iEt \cdot MS$
$M \to \cdot$

$I_{11} = Go(I_7, E)$
$S \to wME \cdot dMS$

$I_{12} = Go(I_8, \})$
$S \to \{L\} \cdot$

$I_{13} = Go(I_8, ;)$
$L \to L; \cdot MS$
$M \to \cdot$

$I_{14} = Go(I_{10}, M)$
$S \to iEtM \cdot SNeMS$
$S \to iEtM \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_{15} = Go(I_{11}, d)$
$S \to wMEdM \cdot S$
$M \to \cdot$

$I_{17} = Go(I_{14}, S)$
$S \to iEtMS \cdot NeMS$
$S \to iEtMS \cdot$
$N \to \cdot$

$I_{16} = Go(I_{13}, M)$
$L \to L; M \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_2 = Go(I_{14}, i)$

$I_3 = Go(I_{14}, w)$

$I_4 = Go(I_{14}, \{)$

$I_5 = Go(I_{14}, A)$

$I_{18} = Go(I_{15}, S)$
$S \to wMEdMS \cdot$

$I_{19} = Go(I_{16}, S)$
$L \to L; MS \cdot$

$I_2 = Go(I_{16}, i)$

$I_3 = Go(I_{16}, w)$

$I_4 = Go(I_{16}, \{)$

$I_5 = Go(I_{16}, A)$

$I_{20} = Go(I_{17}, N)$
$S \to iEtMSN \cdot eMS$

$I_{21} = Go(I_{20}, e)$
$S \to iEtMSNe \cdot MS$
$M \to \cdot$

$I_{22} = Go(I_{21}, M)$
$S \to iEtMSNeM \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_{23} = Go(I_{22}, S)$
$S \to iEtMSNeMS \cdot$

$I_2 = Go(I_{22}, i)$

$I_3 = Go(I_{22}, w)$

$I_4 = Go(I_{22}, \{)$

$I_5 = Go(I_{22}, A)$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4						1			
1											acc				
2															
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															

I_0: $S' \to \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_1 = Go(I_0, S)$
$S' \to S \cdot$

$I_2 = Go(I_0, i)$
$S \to i \cdot EtMSNeMS$
$S \to i \cdot EtMS$

$I_3 = Go(I_0, w)$
$S \to w \cdot MEdMS$
$M \to \cdot$

$I_4 = Go(I_0, \{)$
$S \to \{ \cdot L\}$
$L \to \cdot L;MS$
$L \to \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7															
8															
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															

$I_5 = Go(I_0, A)$
$S \to A \cdot$

$I_6 = Go(I_2, E)$
$S \to iE \cdot tMSNeMS$
$S \to iE \cdot tMS$

$I_7 = Go(I_3, M)$
$S \to wM \cdot EdMS$

$I_8 = Go(I_4, L)$
$S \to \{L \cdot\}$
$L \to L \cdot ;MS$

$I_9 = Go(I_4, S)$
$L \to S \cdot$

$I_2 = Go(I_4, i)$

$I_3 = Go(I_4, w)$

$I_4 = Go(I_4, \{)$

$I_5 = Go(I_4, A)$

$I_{10} = Go(I_6, t)$
$S \to iEt \cdot MSNeMS$
$S \to iEt \cdot MS$
$M \to \cdot$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
14												17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2				
16															
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4				
18															
19															
20															
21															
22															
23															

$I_{11} = Go(I_7, E)$
$S \to wME \cdot dMS$

$I_{12} = Go(I_8, \})$
$S \to \{L\} \cdot$

$I_{13} = Go(I_8, ;)$
$L \to L; \cdot MS$
$M \to \cdot$

$I_{14} = Go(I_{10}, M)$
$S \to iEtM \cdot SNeMS$
$S \to iEtM \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_{15} = Go(I_{11}, d)$
$S \to wMEdM \cdot S$
$M \to \cdot$

$I_{17} = Go(I_{14}, S)$
$S \to iEtMS \cdot NeMS$
$S \to iEtMS \cdot$
$N \to \cdot$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16		
14	S_2			S_3		S_4				S_5		17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18			
16	S_2			S_3		S_4				S_5		19			
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4				
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5				
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8				
20															
21															
22															
23															

$I_{16} = Go(I_{13}, M)$
$L \to L;\ M \cdot S$
$S \to \cdot\, iEtMSNeMS$
$S \to \cdot\, iEtMS$
$S \to \cdot\, wMEdMS$
$S \to \cdot\, \{L\}$
$S \to \cdot\, A$

$I_2 = Go(I_{14}, i)$

$I_3 = Go(I_{14}, w)$

$I_4 = Go(I_{14}, \{)$

$I_5 = Go(I_{14}, A)$

$I_{18} = Go(I_{15}, S)$
$S \to wMEdMS \cdot$

$I_{19} = Go(I_{16}, S)$
$L \to L;\ MS \cdot$

$I_2 = Go(I_{16}, i)$

$I_3 = Go(I_{16}, w)$

$I_4 = Go(I_{16}, \{)$

$I_5 = Go(I_{16}, A)$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16		
14	S_2			S_3		S_4				S_5		17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18			
16	S_2			S_3		S_4				S_5		19			
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4			20	
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5				
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8				
20			S_{21}												
21	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		22		
22	S_2			S_3		S_4				S_5		23			
23	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1				

$I_{20} = Go(I_{17}, N)$
$S \to iEtMSN \cdot eMS$

$I_{21} = Go(I_{20}, e)$
$S \to iEtMSNe \cdot MS$
$M \to \cdot$

$I_{22} = Go(I_{21}, M)$
$S \to iEtMSNeM \cdot S$
$S \to \cdot iEtMSNeMS$
$S \to \cdot iEtMS$
$S \to \cdot wMEdMS$
$S \to \cdot \{L\}$
$S \to \cdot A$

$I_{23} = Go(I_{22}, S)$
$S \to iEtMSNeMS \cdot$

$I_2 = Go(I_{22}, i)$

$I_3 = Go(I_{22}, w)$

$I_4 = Go(I_{22}, \{)$

$I_5 = Go(I_{22}, A)$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16		
14	S_2			S_3		S_4				S_5		17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18			
16	S_2			S_3		S_4				S_5		19			
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4			20	
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5				
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8				
20			S_{21}												
21	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		22		
22	S_2			S_3		S_4				S_5		23			
23	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1				

状态/符号栈	输入串
0 #	$iEt\{A;A\}eA\#$
02 #i	$Et\{A;A\}eA\#$
026 #iE	$t\{A;A\}eA\#$
02610 #iEt	$\{A;A\}eA\#$
02610 14 #$iEtM$	$\{A;A\}eA\#$
02610 144 #$iEtM\{$	$A;A\}eA\#$
02610 1445 #$iEtM\{A$	$;A\}eA\#$
02610 1449 #$iEtM\{S$	$;A\}eA\#$
02610 1448 #$iEtM\{L$	$;A\}eA\#$
02610 144813 #$iEtM\{L;$	$A\}eA\#$
02610 144813 16 #$iEtM\{L;M$	$A\}eA\#$
02610 144813 165 #$iEtM\{L;MA$	$\}eA\#$

(0) $S' \to S$ (1) $S \to iEtMSNeMS$ (2) $M \to \varepsilon$ (3) $N \to \varepsilon$ (4) $S \to iEtMS$ (5) $S \to wMEdMS$
(6) $S \to \{L\}$ (7) $S \to A$ (8) $L \to L;MS$ (9) $L \to S$

状态	i	t	e	w	d	{	}	;	E	A	#	S	M	N	L
0	S_2			S_3		S_4				S_5		1			
1											acc				
2									S_6						
3	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		7		
4	S_2			S_3		S_4				S_5		9			8
5	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7				
6		S_{10}													
7									S_{11}						
8							S_{12}	S_{13}							
9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9	r_9				
10	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		14		
11					S_{15}										
12	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6				
13	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		16		
14	S_2			S_3		S_4				S_5		17			
15	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	18			
16	S_2			S_3		S_4				S_5		19			
17	r_4	r_4	r_3	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4			20	
18	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5				
19	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8	r_8				
20			S_{21}												
21	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2		22		
22	S_2			S_3		S_4				S_5		23			
23	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1				

状态/符号栈	输入串
02610 14813 165 #iEtM{L;MA	}eA#
02610 14813 16 19 #iEtM{L;MS	}eA#
02610 1448 #iEtM{L	}eA#
02610 144812 #iEtM{L}	eA#
02610 14 17 #iEtMS	eA#
02610 14 17 20 #iEtMSN	eA#
02610 14 17 20 21 #iEtMSNe	A#
02610 14 17 20 21 22 #iEtMSNeM	A#
02610 14 17 20 21 225 #iEtMSNeMA	A#
02610 14 17 20 21 22 23 #iEtMSNeMS	#
01 #S	#
01成功 #S	#

【例7.16】续例【7.15】$if\ a < b \vee c \leq d \wedge e\ then\ x = y + z\ else\ x = 0$

$S \rightarrow if\ E\ then\ M_1 S_1 N\ else\ M_2 S_2$

$\{backpatch(E.truelist, M_1.quad);$

$backpatch(E.falselist, M_2.quad);$

$S.nextlist = merge(S_1.nextlist, N.nextlist, S_2.nextlist);\ \}$

3	$\#iEtM$	$x = y + z\ e\ x = 0\#$
4	$\#iEtMx = y$	$+z\ e\ x = 0\#$
5	$\#iEtMx = E$	$+z\ e\ x = 0\#$
6	$\#iEtMx = E + z$	$e\ x = 0\#$
7	$\#iEtMx = E + E$	$e\ x = 0\#$
8	$\#iEtMx = E$	$e\ x = 0\#$
9	$\#iEtMA$	$e\ x = 0\#$
10	$\#iEtMS$	$e\ x = 0\#$
11	$\#iEtMSN$	$e\ x = 0\#$
12	$\#iEtMSNe$	$x = 0\#$
13	$\#iEtMSNeM$	$x = 0\#$
14	$\#iEtMSNeMx = 0$	$\#$
15	$\#iEtMSNeMx = E$	$\#$
16	$\#iEtMSNeMA$	$\#$
17	$\#iEtMSNeMS$	$\#$
18	$\#S$	$\#$

现在剩下最后一个链$S.nextlist$，此处我们手工使用nxq填充。

三地址码

100: $(j <, a, b, 106)$
101: $(j, -, -, 102)$
102: $(j \leq, c, d, 104)$
103: $(j, -, -, 109)$
104: $(jnz, e, -, 106)$
105: $(j, -, -, 109)$
106: $(+, y, z, T1)$
107: $(=, T1, -, x)$
108: $(j, -, -, 110)$
109: $(=, 0, -, x)$

$S.nextlist = null$
$M.quad = 109$
$N.nextlist = 108$
$S.nextlist = null$
$M.quad = 106$
$S.nextlist = 108$

【例7.17】$while\ a < b\ do\ if\ c < d\ then\ x = y + z$，假设$nxq = 100$

$S \to while\ M_1 E\ do\ M_2 S_1$

$\{backpatch(S_1.nextlist, M_1.quad);$

$backpatch(E.truelist, M_2.quad);$

$S.nextlist = E.falselist;$

$gen(j, -, -, M_1.quad);\ \}$

; }

1	#	$wa < bdic < dtx = y + z\#$
2	$\#w$	$a < bdic < dtx = y + z\#$
3	$\#wM$	$a < bdic < dtx = y + z\#$
4	$\#wMa < b$	$dic < dtx = y + z\#$
5	$\#wME$	$dic < dtx = y + z\#$
6	$\#wMEd$	$ic < dtx = y + z\#$
7	$\#wMEdM$	$ic < dtx = y + z\#$
8	$\#wMEdMic < d$	$tx = y + z\#$
9	$\#wMEdMiE$	$tx = y + z\#$
10	$\#wMEdMiEt$	$x = y + z\#$
11	$\#wMEdMiEtM$	$x = y + z\#$
12	$\#wMEdMiEtMx = y + z$	#
13	$\#wMEdMiEtMA$	#
14	$\#wMEdMiEtMS$	#
15	$\#wMEdMS$	#
16	$\#S$	#

此处如果用nxq填充$S.nextlist$，应为：
103: $(j, -, -, 106)$

此处用nxq手工填充$S.nextlist$

三地址码

100: $(j <, a, b, 102)$
101: $(j, -, -, 107)$
102: $(j <, c, d, 104)$
103: $(j, -, -, 100)$
104: $(+, y, z, T1)$
105: $(=, T1, -, x)$
106: $(j, -, -, 100)$

$S.nextlist = null$
$M.quad = 104$
$S.nextlist = 103$
$M.quad = 102$
$E.truelist = 100, E.falselist = 101$
$S.nextlist = 101$

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.5.2 标号与goto语句

- 标号语句形式：$L:S;$

① $S \rightarrow label\ S \mid goto\ i$

② $label \rightarrow i:$

名字	类型	…	定义否	地址
……				
L_1	标号		是	p
……				
L_2	标号		否	nxq

- 若标号已存在
  - 生成四元式：$(j,-,-,p)$

- 若标号不存在
  - 符号表中增加标号，定义否为“否”，地址为nxq；
  - 生成四元式：$(j,-,-,-)$；
  - 当遇到定义标号语句时，回填。

(1) $S \rightarrow label\ S$ {}

(2) $label \rightarrow i$: {$p = lookup(i.name)$;

if $p = null$ then $addlabel(i.name, label, true, nxq)$;

else if $p.type \neq label$ || $p.isdefined = true$ then Error;

else {

$modifylabel(p, isdefined = true)$;

$backpatch(p.address)$;} }

(3)$S \rightarrow goto\ i$ {$p = lookup(i.name)$;

if $p = null$ then {$addlabel(i.name, label, false, nxq)$;

$gen(j, -, -, -)$;}

else if $p.type \neq label$ then Error;

else if $p.isdefined = true$ then $gen(j, -, -, p.address)$;

else {$gen(j, -, -, p.address)$;

$modifylabel(p, address = nxq - 1)$;} }

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.5.3 Case语句的翻译

- Case语句形式：

switch E

{

case c_1: S_1;

case c_2: S_2;

……

case c_{n-1}: S_{n-1};

default: S_n;

}

- 文法：

// Case内部不含default部分

$I \rightarrow AS;$ // S是各类语句

$A \rightarrow case\ c: \mid I\ case\ c:$ // c是常数

// 加上switch和default

$W \rightarrow switch\ E\{ID\}$

$D \rightarrow default\ S; \mid \varepsilon$

// 考虑多个语句

$S \rightarrow S; S$

7.5.3 Case语句的翻译

- Case语句形式：

switch E

{

case c_1: S_1;

case c_2: S_2;

……

case c_{n-1}: S_{n-1};

default: S_n;

}

- 生成四元式：

$goto\ test$

L_1: S_1; $goto\ next$

……

L_{n-1}: S_{n-1}; $goto\ next$

L_n: S_n; $goto\ next$

$test$: $if\ T = c_1\ goto\ L_1$

……

$if\ T = c_{n-1}\ goto\ L_{n-1}$

$goto\ L_n$

$next$:

- 遇到switch
  - 产生标号test；
  - 产生标号next；
  - 产生临时变量T，存放E值；
  - 生成$goto\ test$。
- 遇到c_i
  - 产生标号L_i，填入符号表
  - 记下标号和符号表位置，生成$test$时使用；
  - S之后要有个四元式 $goto\ next$。

7.5.3 Case语句的翻译

- 文法：

// Case内部不含default部分

$I \rightarrow AS;$

$A \rightarrow case\ c: \mid I\ case\ c:$

// 加上switch和default

$W \rightarrow switch\ E\{ID\}$

$D \rightarrow default\ S; \mid \varepsilon$

// 考虑多个语句

$S \rightarrow S;S$

- 文法存在问题：
  - 考虑到标号名字冲突问题，可以考虑为每个Case语句创建一个符号表，这个动作在遇到switch时发生，至少不能晚于{。
  - 遇到}生成test标号里面的语句时，需要知道临时变量的名字，如果把$switch\ E$提出来，需要考虑如何传递的问题。

7.5.3 Case语句的翻译

- 新的文法：

// Case内部不含default部分

$I \rightarrow AS;$

$A \rightarrow case\ c: \mid I\ case\ c:$

// 加上switch和default

$W \rightarrow T\{ID\}$

$T \rightarrow switch\ E$

$D \rightarrow default\ MS; \mid \varepsilon$

$M \rightarrow \varepsilon$

// 考虑多个语句

$S \rightarrow S;S$

翻译模式

$T \to switch\ E$ $\{ mktable(top(tblptr)); Init(Que);$// 创建一张新的符号表和队列

$t = newtemp;\ gen(=, E.place, -, t); T.place = t;$

$addlabel(test, case, false, nxq); gen(j, -, -, -);$

$addlabel(next, case, false, null); \}$

$A \to case\ c:$ $\{ Que.add(c, nxq); \}$

$A \to I\ case\ c:$ $\{ Que.add(c, nxq); \}$

$D \to default: MS;$ $\{ D.quad = M.quad;$

$p = lookup(next); a = p.address$

$modifylabel(p, address = nxq);$

$gen(j, -, -, a);$ // 待回填链 $\}$

$D \to \varepsilon$ $\{ D.quad = null; \}$

$M \to \varepsilon$ $\{ M.quad = nxq; \}$

翻译模式

$I \rightarrow AS;$ $\{ p = lookup(next); a = p.address;$

$modifylabel(p, address = nxq);\ gen(j, -, -, a);$ // 待回填链 $\}$

$W \rightarrow T\{ID\}$ $\{ p = lookup(test); backpatch(p.address, nxq);$

$while\ (!\ Que.IsEmpty)$

$\{ (c, a) = Que.de(); gen(j =, T.place, c, a); \}$

$if\ (D.quad \neq null)\ gen(j, -, -, D.quad);$

$p = lookup(next); backpatch(p.address, nxq)\}$

【例7.18】$switch\ x\{case\ 0: x = x + 1; case\ 1: x = y; default: x = 0;\}$，假设

$I \to AS;$ $\{ p = lookup(next);$
$a = p.address;$
$modifylabel(p, address = nxq);$
$gen(j, -, -, a);\}$

$= t;$
$-);$

三地址码

100: $(=, x, -, T1)$
101: $(j, -, -, -)$
102: $(+, x, 1, T2)$
103: $(=, T2, -, x)$
104: $(j, -, -, -)$

步骤	文法符号栈	输入串
1	#	$sx\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
2	$\#sx$	$\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
3	$\#sE$	$\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
4	$\#T$	$\{c0: x = x + 1; c1: x = y; d: x = 0;\}\#$
5	$\#T\{c0:$	$x = x + 1; c1: x = y; d: x = 0;\}\#$
6	$\#T\{A$	$x = x + 1; c1: x = y; d: x = 0;\}\#$
7	$\#T\{Ax = x$	$+1; c1: x = y; d: x = 0;\}\#$
8	$\#T\{Ax = E$	$+1; c1: x = y; d: x = 0;\}\#$
9	$\#T\{Ax = E + 1$	$; c1: x = y; d: x = 0;\}\#$
10	$\#T\{Ax = E + E$	$; c1: x = y; d: x = 0;\}\#$
11	$\#T\{Ax = E$	$; c1: x = y; d: x = 0;\}\#$
12	$\#T\{AS$	$; c1: x = y; d: x = 0;\}\#$
13	$\#T\{AS;$	$c1: x = y; d: x = 0;\}\#$
14	$\#T\{I$	$c1: x = y; d: x = 0;\}\#$
15	$\#T\{Ic1:$	$x = y; d: x = 0;\}\#$

名字	类型	定义	地址
test	case	F	101
next	case	F	104

0
102

$E.place = 1$
$E.place = T2$
$T.place = T1$

【例7.18】$switch\ x\{case\ 0: x = x + 1; case\ 1: x = y; default: x = 0;\}$，假设

$W \rightarrow T\{ID\}$ $\{ p = lookup(test);$

$backpatch(p.address, nxq);$

$while\ (!Que.IsEmpty)$

$\{(c, a) = Que.de(); gen(j =, T.place, c, a);\}$
$if\ (D.quad \neq null)\ gen(j, -, -, D.quad);$
$p = lookup(next);$

$backpatch(p.address, nxq)\}$

19	$\#T\{AS$	$;d: x = 0;\}\#$
20	$\#T\{AS;$	$d: x = 0;\}\#$
21	$\#T\{I$	$d: x = 0;\}\#$
22	$\#T\{Id:$	$x = 0;\}\#$
23	$\#T\{Id: M$	$x = 0;\}\#$
24	$\#T\{Id: Mx = 0$	$;\}\#$
25	$\#T\{Id: Mx = E$	$;\}\#$
26	$\#T\{Id: MS$	$;\}\#$
27	$\#T\{Id: MS;$	$\}\#$
28	$\#T\{ID$	$\}\#$
29	$\#T\{ID\}$	$\#$
30	$\#W$	$\#$

三地址码

100: $(=, x, -, T1)$	109: $(j =, T1, 0, 102)$
101: $(j, -, -, 109)$	110: $(j =, T1, 1, 105)$
102: $(+, x, 1, T2)$	111: $(j, -, -, 107)$
103: $(=, T2, -, x)$	
104: $(j, -, -, 112)$	
105: $(=, y, -, x)$	
106: $(j, -, -, 112)$	
107: $(=, 0, -, x)$	
108: $(j, -, -, 112)$	

$E.place = 0$
$D.quad = 107$
$T.place = T1$

名字	类型	定义	地址
test	case	F	101
next	case	F	108

0	1
102	105

第七章 语义分析和中间代码生成

- 7.1 中间语言
  - 7.1.1 后缀式
  - 7.1.2 图表示法
  - 7.1.3 三地址代码
- 7.2 说明语句
  - 7.2.1 过程中的说明语句
  - 7.2.2 保留作用域的信息
- 7.3 赋值语句的翻译
  - 7.3.1 简单算术表达式及赋值语句
  - 7.3.2 数组元素的引用
  - 7.3.3 类型转换
- 7.4 布尔表达式的翻译
  - 7.4.1 数值表示法
  - 7.4.2 作为条件控制的布尔式翻译
- 7.5 控制语句的翻译
  - 7.5.1 控制流语句
  - 7.5.2 标号与goto语句
  - 7.5.3 Case语句的翻译
- 7.6 过程调用的处理
- 7.7 类型检查
  - 7.7.1 类型系统
  - 7.7.2 类型检查器的规格说明
  - 7.7.3 函数和运算符的重载
  - 7.7.4 多态函数

7.6 过程调用的处理

- Call S(A+B, Z)：

  $[+, A, B, Z]$

  $[param, -, -, T1]$

  $[param, -, -, Z]$

  $[call, -, -, S]$

$S \rightarrow call\ id\ (Elist)$

$\{\ foreach\ p\ in\ Que\ \ gen(param, -, -, p);$

$\ \ gen(call, -, -, id.place);\ \}$

$Elist \rightarrow Elist, E$

$\{\ Que.add(E.place);\ \}$

$Elist \rightarrow E$

$\{\ Que.add(E.place);\ \}$

第七章作业

【作业7-1】7.4.2节中，关系式$i^{(1)} < i^{(2)}$被翻译成相继的两个四元式：

$(j <, i^{(1)}, i^{(2)}, -)$　　// 真出口

$(j, -, -, -)$　　// 假出口

这种翻译常常浪费一个四元式。如果我们翻译成如下四元式：

$(j \geq, i^{(1)}, i^{(2)}, -)$　　// 假出口跳转，真出口自动滑到下一个四元式

那么，在$i^{(1)} < i^{(2)}$的情况下就不发生跳转（自动滑下来）。但若这个关系后有一个或运算，则另一个无条件转移指令是不可省的，例如$if\ A < B \vee C < D\ then\ x = y$

100: $(j \geq, A, B, 102)$

101: $(j, -, -, 103)$　　// 或运算前的无条件跳转不能省略

102: $(j \geq, C, D, 104)$

103: $(=, y, -, x)$

请按上述要求改写翻译布尔表达式的语义动作。