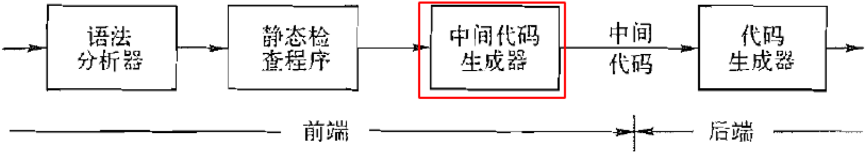
# 中间代码生成

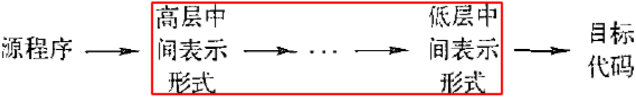
魏恒峰

hfwei@nju.edu.cn

2020 年 12 月 21 日


语法
分析器
静态检
查程序
中间代码
生成器
中间
代码
代码
生成器
前端
后端

## Intermediate Representation (IR)



**精确:** 不能丢失源程序的信息

**独立:** 不依赖特定的源语言与目标语言

(如, 没有复杂的寻址方式)

## Intermediate Representation (IR)

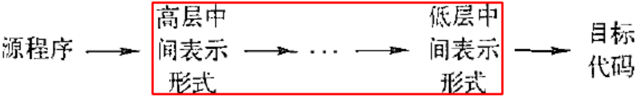
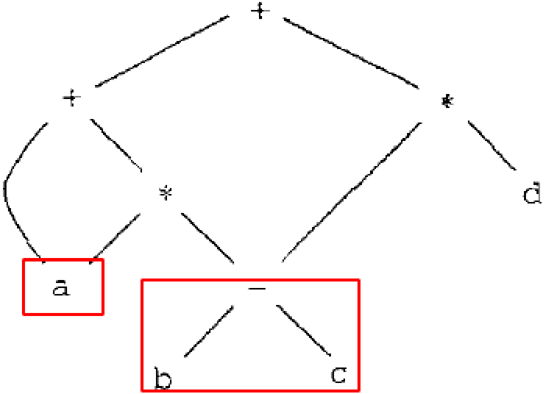
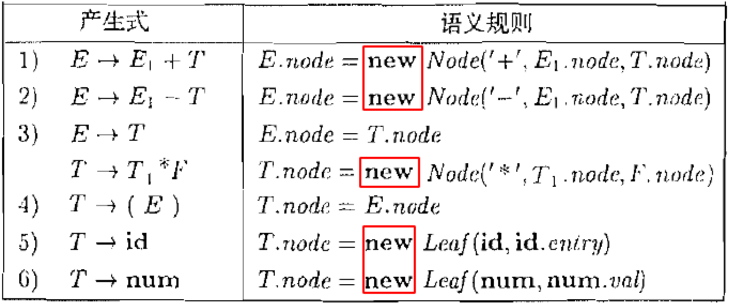


图 (抽象语法树)、**三地址代码**、C 语言

## 表达式的有向无环图



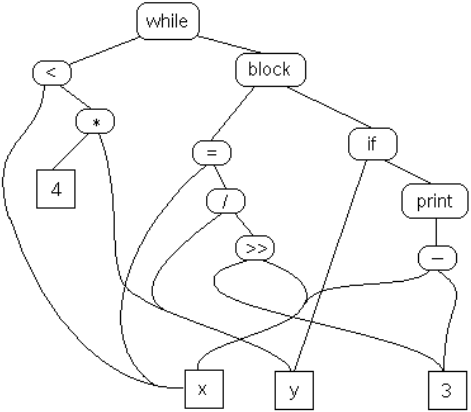
$$a + a * (b - c) + (b - c) * d$$



| 产生式 | 语义规则 |
|---|---|
| 1) $E \to E_1 + T$ | $E.node = \boxed{\textbf{new}}\ Node('+', E_1.node, T.node)$ |
| 2) $E \to E_1 - T$ | $E.node = \boxed{\textbf{new}}\ Node('-', E_1.node, T.node)$ |
| 3) $E \to T$ | $E.node = T.node$ |
| $T \to T_1 * F$ | $T.node = \boxed{\textbf{new}}\ Node('*', T_1.node, F.node)$ |
| 4) $T \to (\ E\ )$ | $T.node = E.node$ |
| 5) $T \to \textbf{id}$ | $T.node = \boxed{\textbf{new}}\ Leaf(\textbf{id}, \textbf{id}.entry)$ |
| 6) $T \to \textbf{num}$ | $T.node = \boxed{\textbf{new}}\ Leaf(\textbf{num}, \textbf{num}.val)$ |

在创建节点之前, 先判断是否已存在 (哈希表)

```
while (x < 4 * y) {
    x = y / 3 >> x;
    if (y) print x - 3;
}
```



Definition (三地址代码 (Three-Address Code (TAC; 3AC)))

每个 **TAC** 指令**最多**包含三个操作数。

$$x = y\ \mathbf{op}\ z \tag{1}$$

$$x = \mathbf{op}\ y \tag{2}$$

$$x = y \tag{3}$$

$$\mathbf{goto}\ L \tag{4}$$

$$\mathbf{if}\ x\ \mathbf{goto}\ L \tag{5}$$

$$\mathbf{if\ False}\ x\ \mathbf{goto}\ L \tag{6}$$

$$\mathbf{if}\ x\ \mathbf{relop}\ y\ \mathbf{goto}\ L \tag{7}$$

Definition (三地址代码 (Three-Address Code (TAC; 3AC)))

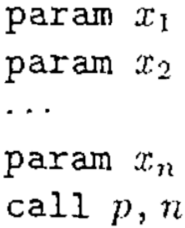
每个 **TAC** 指令**最多**包含三个操作数。

$$\textbf{param}\ x \tag{8}$$

$$\textbf{call}\ p, n \tag{9}$$

$$y = \textbf{call}\ p, n \tag{10}$$

$$\textbf{return}\ y \tag{11}$$



```
param x1
param x2
...
param xn
call p, n
```

$p(x_1, x_2, \ldots, x_n)$

Definition (三地址代码 (Three-Address Code (TAC; 3AC)))

每个 **TAC** 指令**最多**包含三个操作数。

$$x = y[i] \tag{12}$$

$$x[i] = y \tag{13}$$

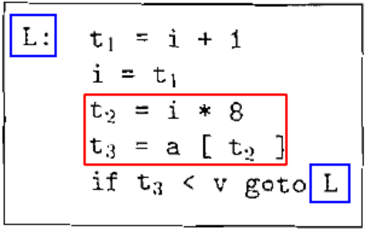
**距离位置 $y$ 处 $i$ 个内存单元**

$$x = \&y \tag{14}$$

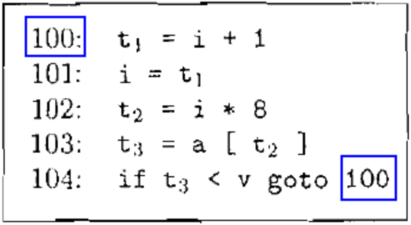
$$x = *y \tag{15}$$

$$*x = y \tag{16}$$

```
do i = i + 1; while (a[i] < v);
```



```
L:   t1 = i + 1
     i = t1
     t2 = i * 8
     t3 = a [ t2 ]
     if t3 < v goto L
```



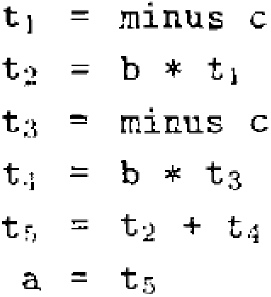
```
100:  t1 = i + 1
101:  i = t1
102:  t2 = i * 8
103:  t3 = a [ t2 ]
104:  if t3 < v goto 100
```

# 三地址代码的四元式表示

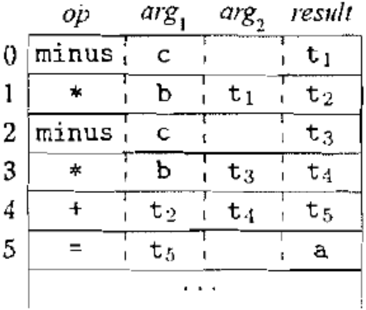
Definition (四元式 (Quadruple))

一个四元式包含四个字段, 分别为 $op$、$arg_1$、$arg_2$ 与 $result$。

$$a + a * (b - c) + (b - c) * d$$



```
t1 = minus c
t2 = b * t1
t3 = minus c
t4 = b * t3
t5 = t2 + t4
 a = t5
```

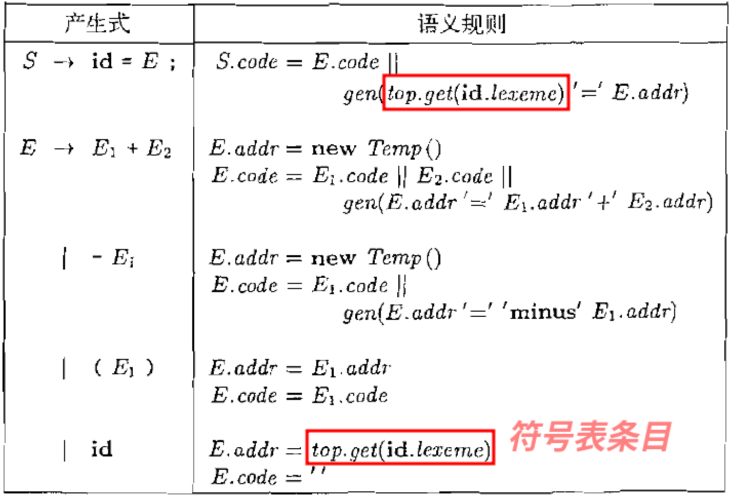


|   | $op$ | $arg_1$ | $arg_2$ | $result$ |
|---|---|---|---|---|
| 0 | minus | c |  | $t_1$ |
| 1 | * | b | $t_1$ | $t_2$ |
| 2 | minus | c |  | $t_3$ |
| 3 | * | b | $t_3$ | $t_4$ |
| 4 | + | $t_2$ | $t_4$ | $t_5$ |
| 5 | = | $t_5$ |  | a |
|   |  | … |  |  |

| | | | | |
|---|---|---|---|---|
| $x = y[i]$ | $=[\ ]$ | $y$ | $i$ | $x$ |
| $x[i] = y$ | $[\ ]=$ | $i$ | $y$ | $x$ |

| | | | | |
|---|---|---|---|---|
| $x = \&y$ | $=\&$ | $y$ | | $x$ |
| $x = *y$ | $=*$ | $y$ | | $x$ |
| $*x = y$ | $*=$ | $y$ | | $x$ |

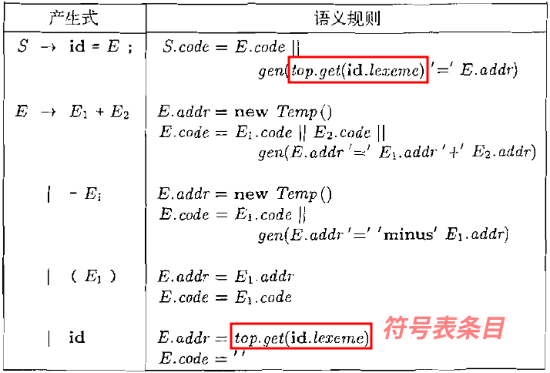
# 表达式的中间代码翻译



| 产生式 | 语义规则 |
|---|---|
| $S \to \textbf{id} = E\ ;$ | $S.code = E.code \Vert$ <br> $gen(\boxed{top.get(\textbf{id}.lexeme)}\ '='\ E.addr)$ |
| $E \to E_1 + E_2$ | $E.addr = \textbf{new}\ Temp()$ <br> $E.code = E_1.code \Vert E_2.code \Vert$ <br> $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$ |
| $\quad \mid\ - E_1$ | $E.addr = \textbf{new}\ Temp()$ <br> $E.code = E_1.code \Vert$ <br> $gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr)$ |
| $\quad \mid\ (\ E_1\ )$ | $E.addr = E_1.addr$ <br> $E.code = E_1.code$ |
| $\quad \mid\ \textbf{id}$ | $E.addr = \boxed{top.get(\textbf{id}.lexeme)}$ <br> $E.code = ''$ |

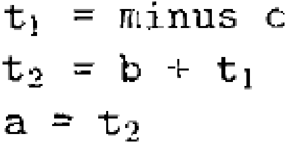
符号表条目

综合属性 $E.code$ 与 $E.addr$



| 产生式 | 语义规则 |
|---|---|
| $S \rightarrow \mathbf{id} = E\ ;$ | $S.code = E.code \ \Vert$ <br> $gen(\boxed{top.get(\mathbf{id}.lexeme)}\ '=' \ E.addr)$ |
| $E \rightarrow E_1 + E_2$ | $E.addr = \mathbf{new}\ Temp\,()$ <br> $E.code = E_1.code \ \Vert\ E_2.code \ \Vert$ <br> $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$ |
| $\mid\ -\ E_1$ | $E.addr = \mathbf{new}\ Temp\,()$ <br> $E.code = E_1.code \ \Vert$ <br> $gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr)$ |
| $\mid\ (\ E_1\ )$ | $E.addr = E_1.addr$ <br> $E.code = E_1.code$ |
| $\mid\ \mathbf{id}$ | $E.addr = \boxed{top.get(\mathbf{id}.lexeme)}$ <br> $E.code = ''\,''$ |

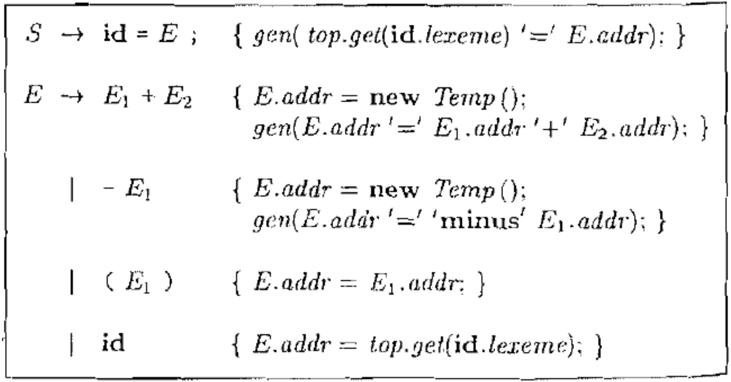
符号表条目



```
t1 = minus c
t2 = b + t1
a = t2
```

$$a = b + -c$$

## 表达式的中间代码翻译 (增量式)



| | |
|---|---|
| $S \rightarrow \textbf{id} = E \;;$ | $\{\ gen(\ top.get(\textbf{id}.lexeme)\ '='\ E.addr);\ \}$ |
| $E \rightarrow E_1 + E_2$ | $\{\ E.addr = \textbf{new}\ Temp\,();$<br>$gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \}$ |
| $\quad \mid\ - E_1$ | $\{\ E.addr = \textbf{new}\ Temp\,();$<br>$gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr);\ \}$ |
| $\quad \mid\ (\ E_1\ )$ | $\{\ E.addr = E_1.addr;\ \}$ |
| $\quad \mid\ \textbf{id}$ | $\{\ E.addr = top.get(\textbf{id}.lexeme);\ \}$ |

综合属性 $E.addr$

## 数组引用的中间代码翻译

声明 : $\text{int } a[2][3]$

数组引用 : $x = a[1][2];\ a[1][2] = x$

需要计算 $a[1][2]$ 的相对于**数组基地址** $a$ 的**偏移地址**

# 数组引用的中间代码翻译

int $a[2][3]$

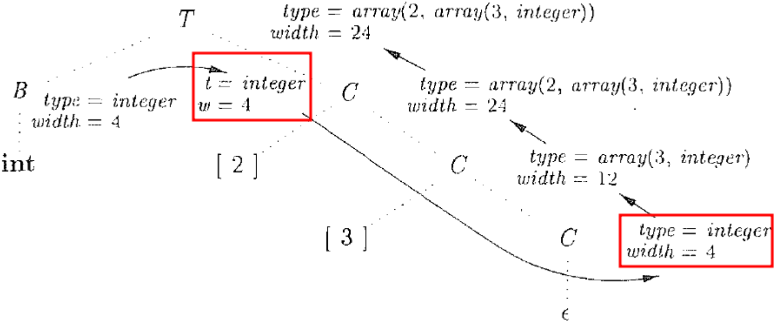


图 6-16 数组类型的语法制导翻译

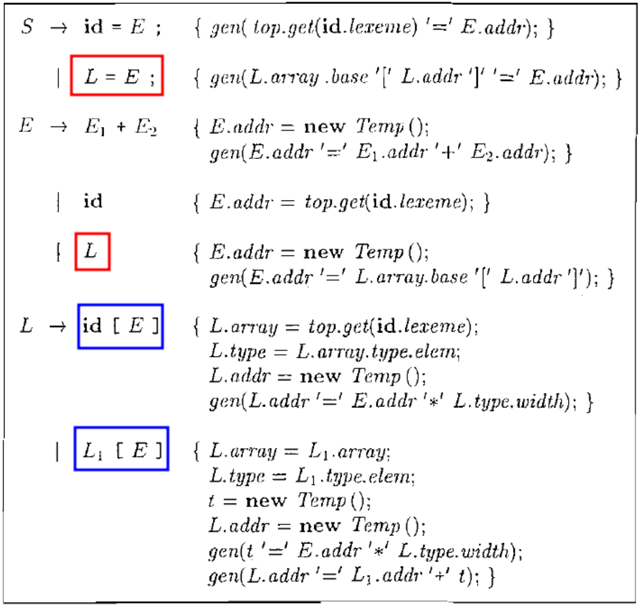
**数组类型声明**

$$\text{int } a[2][3]$$

$$array(2, array(3, \text{integer}))$$

| | 元素类型 | 元素宽度 |
|---|---|---|
| $a[\,]$ | $array(3, \text{integer})$ | 12 |
| $a[\,][\,]$ | integer | 4 |

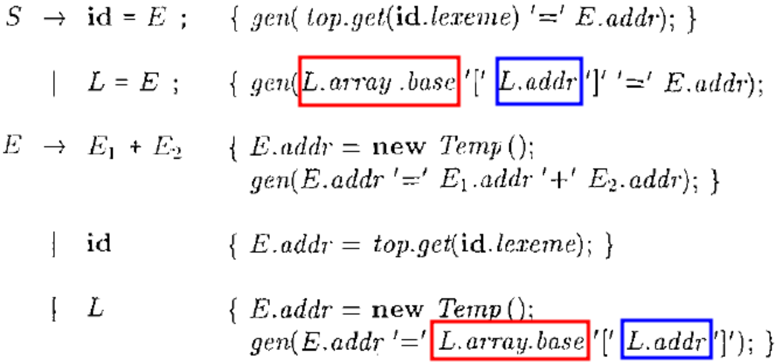
$$addr(a[1][2]) = base + 1 \times 12 + 2 \times 4$$



| | | | |
|---|---|---|---|
| $S$ | $\rightarrow$ | **id** = $E$ ; | { *gen*( *top.get*(**id**.*lexeme*) '=' $E.addr$); } |
| | \| | $\boxed{L = E\ ;}$ | { *gen*($L.array.base$ '[' $L.addr$ ']' '=' $E.addr$); } |
| $E$ | $\rightarrow$ | $E_1 + E_2$ | { $E.addr$ = **new** *Temp* ();<br>*gen*($E.addr$ '=' $E_1.addr$ '+' $E_2.addr$); } |
| | \| | **id** | { $E.addr$ = *top.get*(**id**.*lexeme*); } |
| | \| | $\boxed{L}$ | { $E.addr$ = **new** *Temp* ();<br>*gen*($E.addr$ '=' $L.array.base$ '[' $L.addr$ ']'); } |
| $L$ | $\rightarrow$ | $\boxed{\textbf{id}\ [\ E\ ]}$ | { $L.array$ = *top.get*(**id**.*lexeme*);<br>$L.type$ = $L.array.type.elem$;<br>$L.addr$ = **new** *Temp* ();<br>*gen*($L.addr$ '=' $E.addr$ '*' $L.type.width$); } |
| | \| | $\boxed{L_1\ [\ E\ ]}$ | { $L.array$ = $L_1.array$;<br>$L.type$ = $L_1.type.elem$;<br>$t$ = **new** *Temp* ();<br>$L.addr$ = **new** *Temp* ();<br>*gen*($t$ '=' $E.addr$ '*' $L.type.width$);<br>*gen*($L.addr$ '=' $L_1.addr$ '+' $t$); } |

int $a[2][3]$

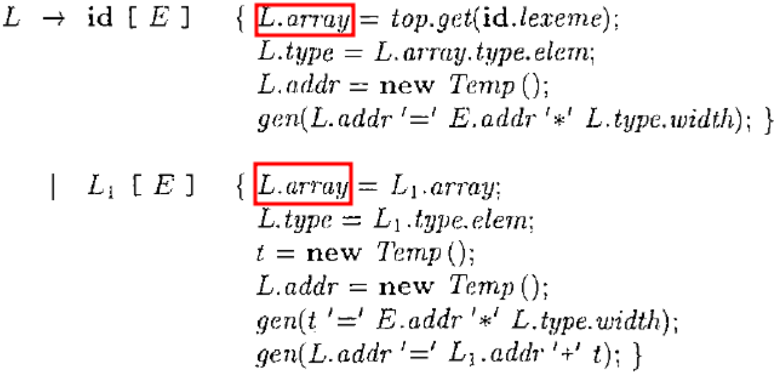
## 综合属性 $L.array.base$ : 数组基地址 (即, 数组名)



$$
\begin{array}{lcll}
S & \rightarrow & \textbf{id} = E \;; & \{\ gen(\ top.get(\textbf{id}.lexeme)\ '='\ E.addr);\ \} \\
& | & L = E \;; & \{\ gen(\boxed{L.array.base}\ '['\ \boxed{L.addr}\ ']'\ '='\ E.addr); \\
E & \rightarrow & E_1 + E_2 & \{\ E.addr = \textbf{new}\ Temp\,(); \\
& & & \quad gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \} \\
& | & \textbf{id} & \{\ E.addr = top.get(\textbf{id}.lexeme);\ \} \\
& | & L & \{\ E.addr = \textbf{new}\ Temp\,(); \\
& & & \quad gen(E.addr\ '='\ \boxed{L.array.base}\ '['\ \boxed{L.addr}\ ']');\ \}
\end{array}
$$

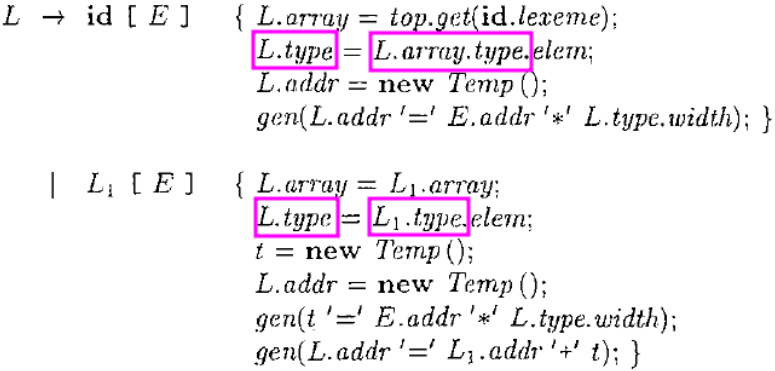
## 综合属性 $L.addr$ : 偏移地址

## 综合属性 $L.array$ : 数组名 id对应的符号表条目



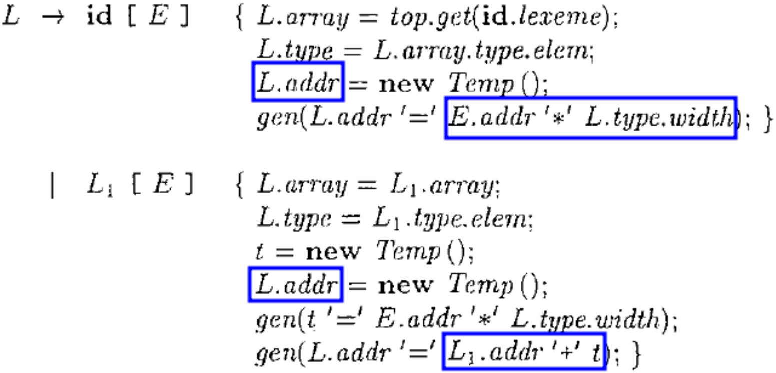
$$
\begin{array}{llll}
L & \rightarrow & \mathbf{id}\ [\ E\ ] & \{\ \boxed{L.array} = top.get(\mathbf{id}.lexeme); \\
 & & & \quad L.type = L.array.type.elem; \\
 & & & \quad L.addr = \mathbf{new}\ Temp\,(); \\
 & & & \quad gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \} \\
 & & & \\
 & \mid & L_1\ [\ E\ ] & \{\ \boxed{L.array} = L_1.array; \\
 & & & \quad L.type = L_1.type.elem; \\
 & & & \quad t = \mathbf{new}\ Temp\,(); \\
 & & & \quad L.addr = \mathbf{new}\ Temp\,(); \\
 & & & \quad gen(t\ '='\ E.addr\ '*'\ L.type.width); \\
 & & & \quad gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}
\end{array}
$$

## 综合属性 $L.type$ :（当前）元素类型

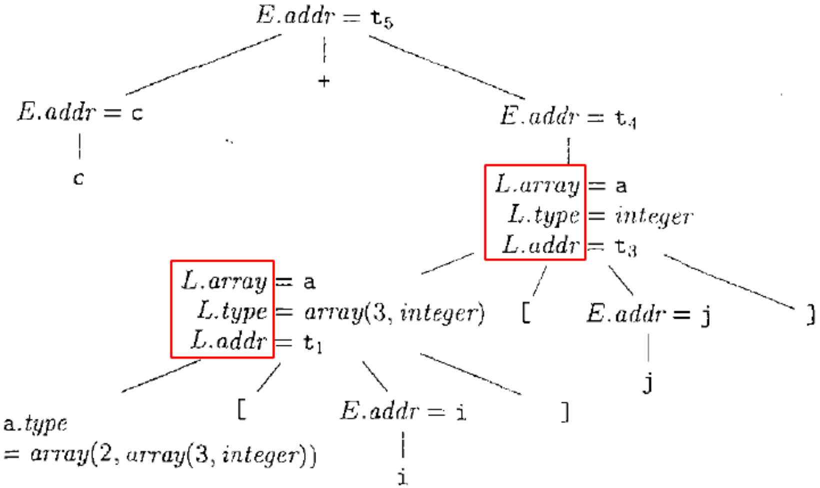


$$
\begin{array}{llll}
L & \rightarrow & \mathbf{id}\ [\ E\ ] & \{\ L.array = top.get(\mathbf{id}.lexeme); \\
 & & & \ \ \boxed{L.type} = \boxed{L.array.type.}elem; \\
 & & & \ \ L.addr = \mathbf{new}\ Temp\,(); \\
 & & & \ \ gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \} \\
 & | & L_1\ [\ E\ ] & \{\ L.array = L_1.array; \\
 & & & \ \ \boxed{L.type} = \boxed{L_1.type.}elem; \\
 & & & \ \ t = \mathbf{new}\ Temp\,(); \\
 & & & \ \ L.addr = \mathbf{new}\ Temp\,(); \\
 & & & \ \ gen(t\ '='\ E.addr\ '*'\ L.type.width); \\
 & & & \ \ gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}
\end{array}
$$

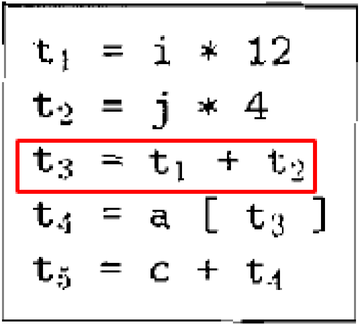
# 综合属性 $L.addr$：（当前）偏移地址



$$
\begin{array}{llll}
L & \rightarrow & \textbf{id}\ [\ E\ ] & \{\ L.array = top.get(\textbf{id}.lexeme); \\
 & & & \quad L.type = L.array.type.elem; \\
 & & & \quad \boxed{L.addr} = \textbf{new}\ Temp\,(); \\
 & & & \quad gen(L.addr\ '='\ \boxed{E.addr\ '*'\ L.type.width});\ \} \\
 & & & \\
 & \mid & L_1\ [\ E\ ] & \{\ L.array = L_1.array; \\
 & & & \quad L.type = L_1.type.elem; \\
 & & & \quad t = \textbf{new}\ Temp\,(); \\
 & & & \quad \boxed{L.addr} = \textbf{new}\ Temp\,(); \\
 & & & \quad gen(t\ '='\ E.addr\ '*'\ L.type.width); \\
 & & & \quad gen(L.addr\ '='\ \boxed{L_1.addr\ '+'\ t});\ \}
\end{array}
$$


E.addr = t5
+
E.addr = c
c
E.addr = t4
L.array = a
L.type = integer
L.addr = t3
[
E.addr = j
j
]
L.array = a
L.type = array(3, integer)
L.addr = t1
a.type
= array(2, array(3, integer))
[
E.addr = i
i
]

int $a[2][3]$



$t_1 = i * 12$

$t_2 = j * 4$

$t_3 = t_1 + t_2$

$t_4 = a [ t_3 ]$

$t_5 = c + t_4$

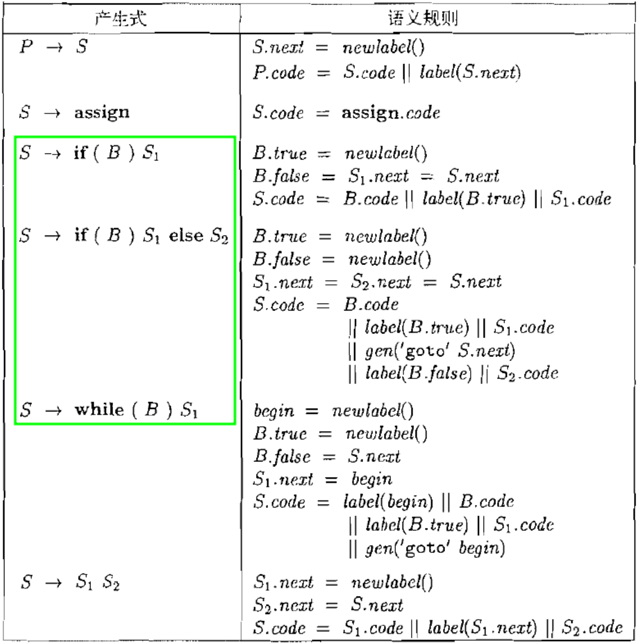
int $a[2][3]$

## 控制流语句的中间代码翻译

$$S \rightarrow \textbf{if}\ (B)\ S_1$$

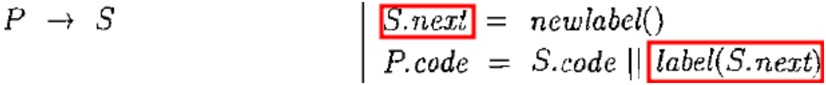
$$S \rightarrow \textbf{if}\ (B)\ S_1\ \textbf{else}\ S_2$$

$$S \rightarrow \textbf{while}\ (B)\ S_1$$



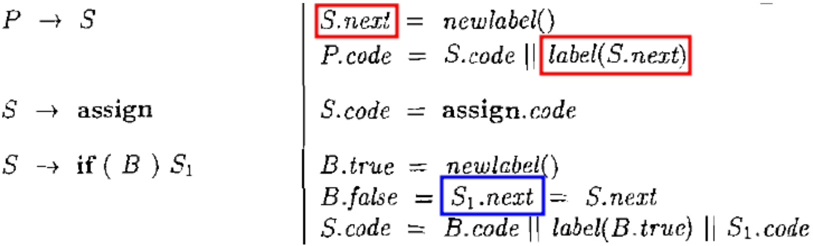
| 产生式 | 语义规则 |
|---|---|
| $P \to S$ | $S.\mathit{next} = \mathit{newlabel}()$<br>$P.\mathit{code} = S.\mathit{code} \Vert \mathit{label}(S.\mathit{next})$ |
| $S \to \textbf{assign}$ | $S.\mathit{code} = \textbf{assign}.\mathit{code}$ |
| $S \to \textbf{if} \ ( \ B \ ) \ S_1$ | $B.\mathit{true} = \mathit{newlabel}()$<br>$B.\mathit{false} = S_1.\mathit{next} = S.\mathit{next}$<br>$S.\mathit{code} = B.\mathit{code} \Vert \mathit{label}(B.\mathit{true}) \Vert S_1.\mathit{code}$ |
| $S \to \textbf{if} \ ( \ B \ ) \ S_1 \ \textbf{else} \ S_2$ | $B.\mathit{true} = \mathit{newlabel}()$<br>$B.\mathit{false} = \mathit{newlabel}()$<br>$S_1.\mathit{next} = S_2.\mathit{next} = S.\mathit{next}$<br>$S.\mathit{code} = B.\mathit{code}$<br>$\Vert \mathit{label}(B.\mathit{true}) \Vert S_1.\mathit{code}$<br>$\Vert \mathit{gen}(\texttt{'goto'} \ S.\mathit{next})$<br>$\Vert \mathit{label}(B.\mathit{false}) \Vert S_2.\mathit{code}$ |
| $S \to \textbf{while} \ ( \ B \ ) \ S_1$ | $\mathit{begin} = \mathit{newlabel}()$<br>$B.\mathit{true} = \mathit{newlabel}()$<br>$B.\mathit{false} = S.\mathit{next}$<br>$S_1.\mathit{next} = \mathit{begin}$<br>$S.\mathit{code} = \mathit{label}(\mathit{begin}) \Vert B.\mathit{code}$<br>$\Vert \mathit{label}(B.\mathit{true}) \Vert S_1.\mathit{code}$<br>$\Vert \mathit{gen}(\texttt{'goto'} \ \mathit{begin})$ |
| $S \to S_1 \ S_2$ | $S_1.\mathit{next} = \mathit{newlabel}()$<br>$S_2.\mathit{next} = S.\mathit{next}$<br>$S.\mathit{code} = S_1.\mathit{code} \Vert \mathit{label}(S_1.\mathit{next}) \Vert S_2.\mathit{code}$ |

继承属性 $S.next$ : $S$ 的下一条指令

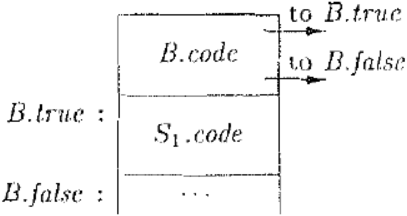


$$P \to S \qquad \boxed{S.next} = newlabel()$$
$$P.code = S.code \;||\; \boxed{label(S.next)}$$

$S.next$ 为语句 $S$ 指明了“跳出” $S$ 的目标

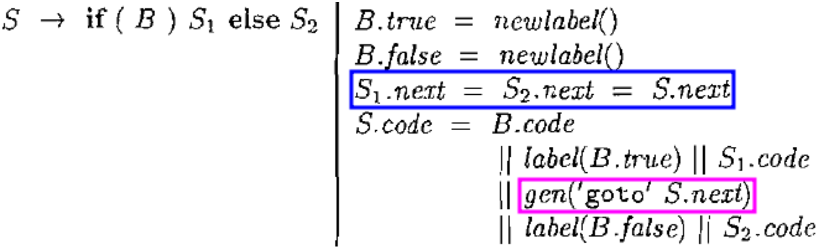


| | |
|---|---|
| $P \rightarrow S$ | $\boxed{S.next} = newlabel()$<br>$P.code = S.code \mid\mid \boxed{label(S.next)}$ |
| $S \rightarrow \textbf{assign}$ | $S.code = \textbf{assign}.code$ |
| $S \rightarrow \textbf{if}\ (\ B\ )\ S_1$ | $B.true = newlabel()$<br>$B.false = \boxed{S_1.next} = S.next$<br>$S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$ |



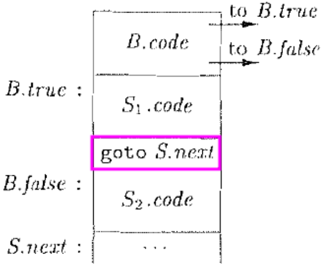
if (B)

  if （B） **assign**

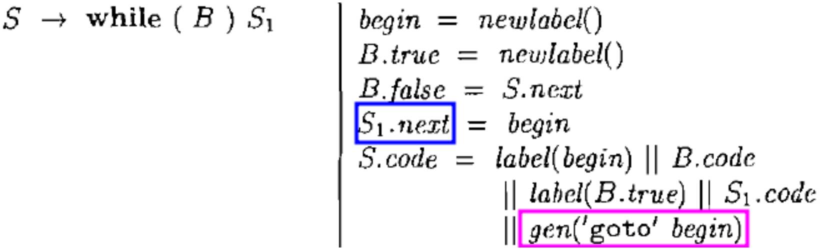


$$S \rightarrow \textbf{if} \; ( \; B \; ) \; S_1 \; \textbf{else} \; S_2$$

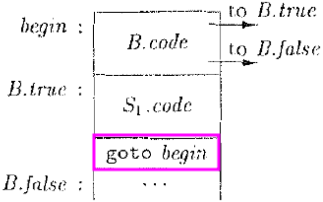
$$\begin{aligned}
B.true &= newlabel() \\
B.false &= newlabel() \\
&\boxed{S_1.next \; = \; S_2.next \; = \; S.next} \\
S.code &= B.code \\
&\quad || \; label(B.true) \; || \; S_1.code \\
&\quad || \; \boxed{gen('\texttt{goto}' \; S.next)} \\
&\quad || \; label(B.false) \; || \; S_2.code
\end{aligned}$$



```
if (B)
  if (B) assign else assign
else
  assign
```

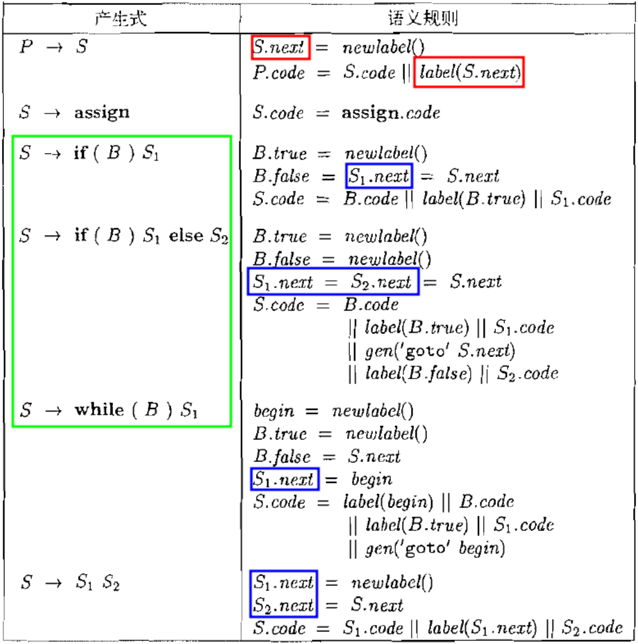


| | |
|---|---|
| $S \rightarrow$ **while** $(\ B\ )\ S_1$ | $begin = newlabel()$<br>$B.true = newlabel()$<br>$B.false = S.next$<br>$\boxed{S_1.next} = begin$<br>$S.code = label(begin) \parallel B.code$<br>$\parallel label(B.true) \parallel S_1.code$<br>$\parallel \boxed{gen(\texttt{'goto'}\ begin)}$ |



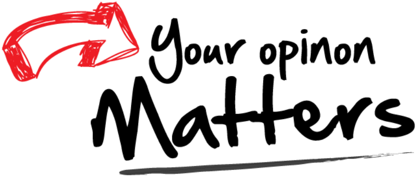
```
while (B)
  if (B) assign else assign
```

$$S \rightarrow S_1\ S_2 \qquad \begin{array}{l} \boxed{\begin{array}{l} S_1.next \\ S_2.next \end{array}} \begin{array}{l} = \ newlabel() \\ = \ S.next \end{array} \\ S.code \ = \ S_1.code \ || \ label(S_1.next) \ || \ S_2.code \end{array}$$



| 产生式 | 语义规则 |
|---|---|
| $P \rightarrow S$ | $\boxed{S.next} = newlabel()$<br>$P.code = S.code \,\|\|\, \boxed{label(S.next)}$ |
| $S \rightarrow \textbf{assign}$ | $S.code = \textbf{assign}.code$ |
| $S \rightarrow \textbf{if}\ (\ B\ )\ S_1$ | $B.true = newlabel()$<br>$B.false = \boxed{S_1.next} = S.next$<br>$S.code = B.code \,\|\|\, label(B.true) \,\|\|\, S_1.code$ |
| $S \rightarrow \textbf{if}\ (\ B\ )\ S_1\ \textbf{else}\ S_2$ | $B.true = newlabel()$<br>$B.false = newlabel()$<br>$\boxed{S_1.next = S_2.next} = S.next$<br>$S.code = B.code$<br>$\|\| \ label(B.true) \,\|\|\, S_1.code$<br>$\|\| \ gen(\text{'goto'}\ S.next)$<br>$\|\| \ label(B.false) \,\|\|\, S_2.code$ |
| $S \rightarrow \textbf{while}\ (\ B\ )\ S_1$ | $begin = newlabel()$<br>$B.true = newlabel()$<br>$B.false = S.next$<br>$\boxed{S_1.next} = begin$<br>$S.code = label(begin) \,\|\|\, B.code$<br>$\|\| \ label(B.true) \,\|\|\, S_1.code$<br>$\|\| \ gen(\text{'goto'}\ begin)$ |
| $S \rightarrow S_1\ S_2$ | $\boxed{S_1.next} = newlabel()$<br>$\boxed{S_2.next} = S.next$<br>$S.code = S_1.code \,\|\|\, label(S_1.next) \,\|\|\, S_2.code$ |

Thank
You!



Office 926

hfwei@nju.edu.cn