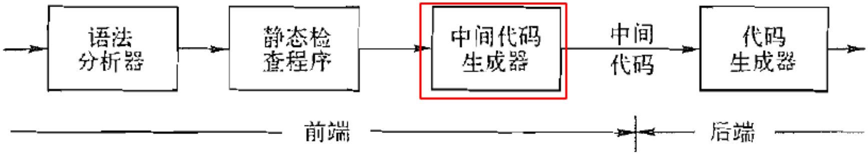
中间代码生成

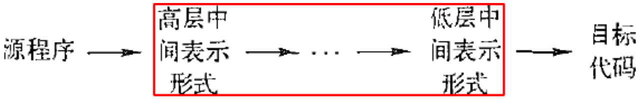
魏恒峰

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)

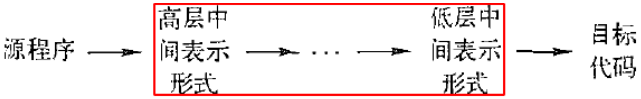
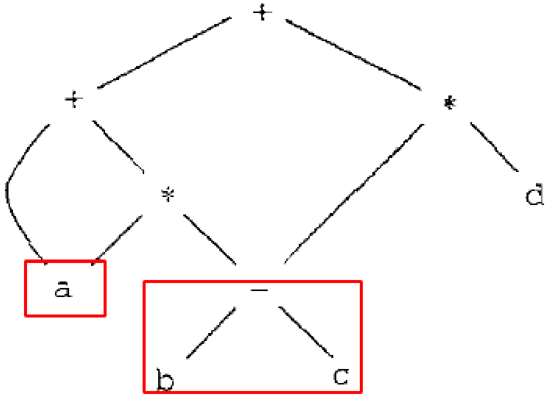
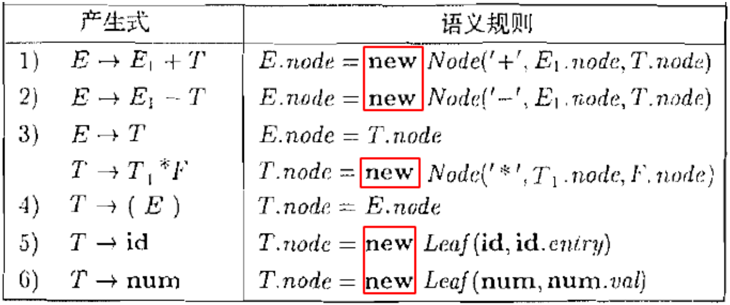


图 (抽象语法树)、**三地址代码**、C 语言

表达式的有向无环图



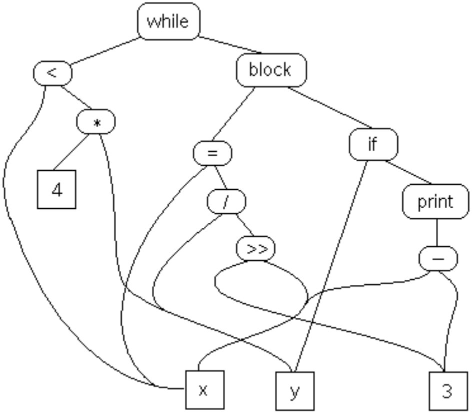
$$a + a * (b - c) + (b - c) * d$$



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。

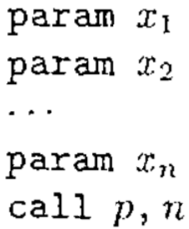
$$x = y \ \textbf{op} \ z \qquad (1)$$
$$x = \textbf{op} \ y \qquad (2)$$
$$x = y \qquad (3)$$

$$\textbf{goto} \ L \qquad (4)$$
$$\textbf{if} \ x \ \textbf{goto} \ L \qquad (5)$$
$$\textbf{if False} \ x \ \textbf{goto} \ L \qquad (6)$$
$$\textbf{if} \ x \ \textbf{relop} \ y \ \textbf{goto} \ L \qquad (7)$$

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 x_1
param x_2
...
param x_n
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。

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

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

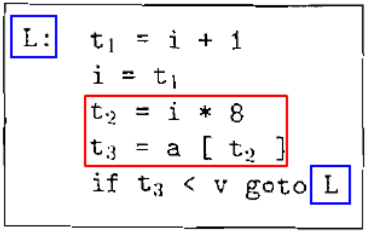
距离位置 y 处 i 个内存单元

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

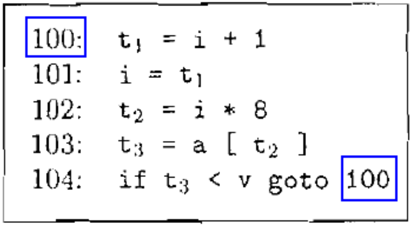
$$x = *y \tag{15}$$

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

```
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
```



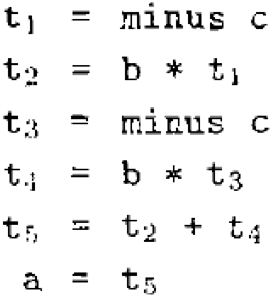
```
100:  t1 = i + 1
101:  i = t1
102:  t2 = i * 8
103:  t3 = a [ t2 ]
104:  if t3 < v goto 100
```

三地址代码的四元式表示

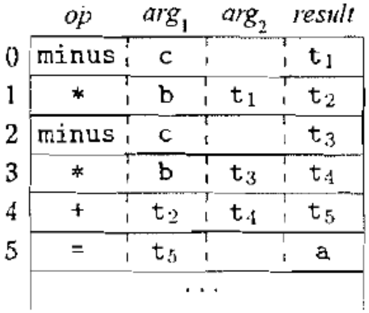
Definition (四元式 (Quadruple))

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

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



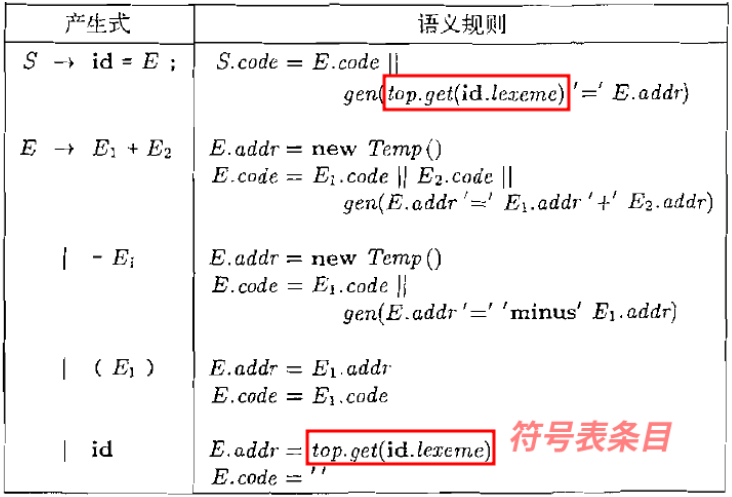
```
t1 = minus c
t2 = b * t1
t3 = minus c
t4 = b * t3
t5 = t2 + t4
 a = t5
```



	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
		…		

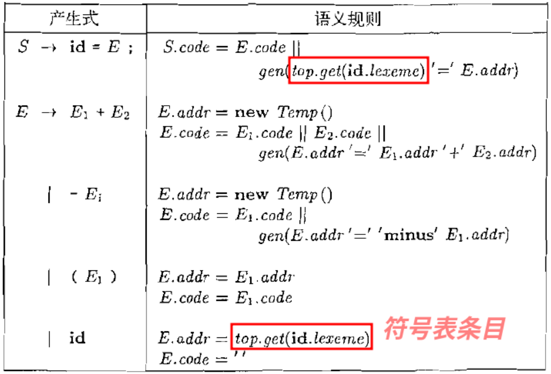
$x = y[i]$	$=[\,]$	y	i	x
$x[i] = y$	$[\,]=$	i	y	x
$x = \&y$	$=\&$	y		x
$x = *y$	$=*$	y		x
$*x = y$	$*=$	y		x

表达式的中间代码翻译

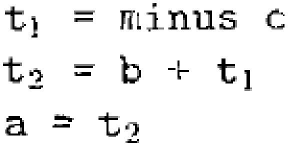


产生式	语义规则
$S \to \textbf{id} = E\ ;$	$S.code = E.code \Vert gen(\boxed{top.get(\textbf{id}.lexeme)}\ '='\ E.addr)$
$E \to E_1 + E_2$	$E.addr = \textbf{new}\ Temp()$ $E.code = E_1.code \Vert E_2.code \Vert gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid - E_1$	$E.addr = \textbf{new}\ Temp()$ $E.code =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

综合属性 $E.code$ 与 $E.addr$



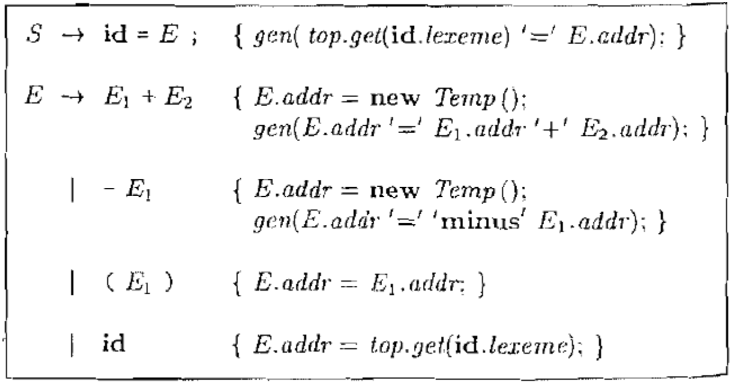
产生式	语义规则
$S \rightarrow \textbf{id} = E\ ;$	$S.code = E.code \parallel gen(\boxed{top.get(\textbf{id}.lexeme)}\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \textbf{new}\ Temp()$ $E.code = E_1.code \parallel E_2.code \parallel gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ - E_1$	$E.addr = \textbf{new}\ Temp()$ $E.code = E_1.code \parallel gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \textbf{id}$	$E.addr = \boxed{top.get(\textbf{id}.lexeme)}$ 符号表条目 $E.code =\ ''$



```
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	$\rightarrow$	**id** = E ;	{ *gen*(*top.get*(**id**.*lexeme*) '=' *E.addr*); }
E	$\rightarrow$	$E_1 + E_2$	{ *E.addr* = **new** *Temp* (); *gen*(*E.addr* '=' E_1.*addr* '+' E_2.*addr*); }
	|	$- E_1$	{ *E.addr* = **new** *Temp* (); *gen*(*E.addr* '=' '**minus**' E_1.*addr*); }
	|	(E_1)	{ *E.addr* = E_1.*addr*; }
	|	**id**	{ *E.addr* = *top.get*(**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 $a[2][3]$

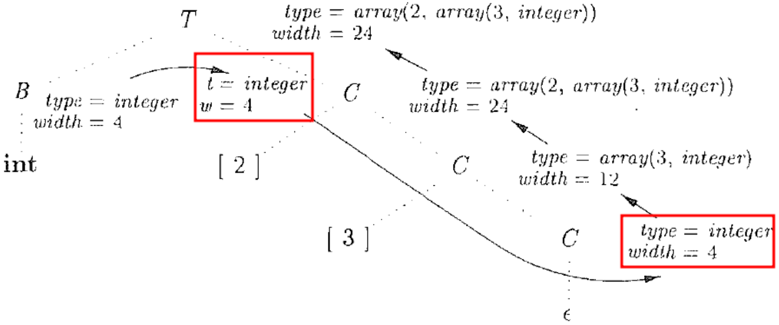


图 6-16　数组类型的语法制导翻译

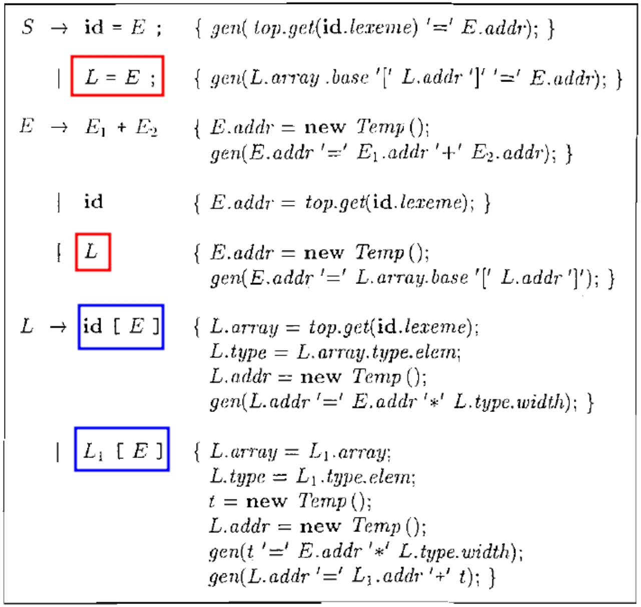
数组类型声明

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

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

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

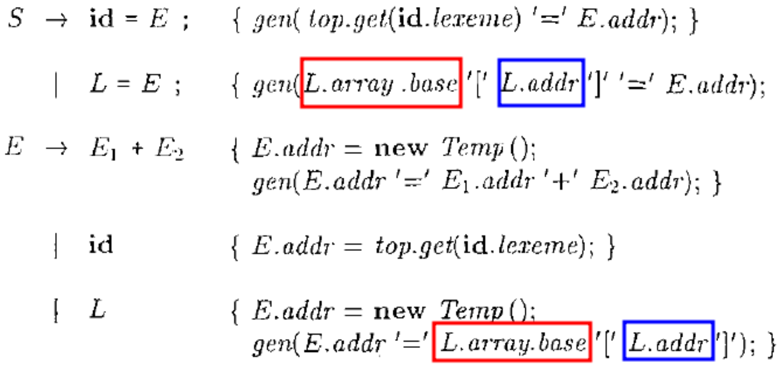
$$addr(a[1][2]) = base + 1 \times 12 + 2 \times 4$$



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

int $a[2][3]$

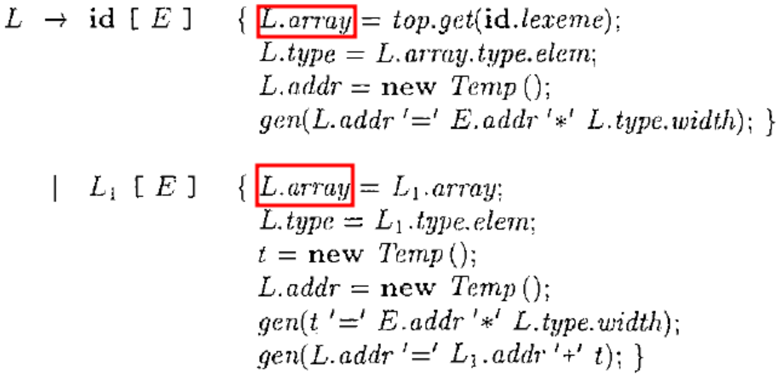
综合属性 $L.array.base$: 数组基地址 (即, 数组名)



S	$\rightarrow$	$\mathbf{id} = E\ ;$	$\{\ gen(\ top.get(\mathbf{id}.lexeme)\ '='\ E.addr);\ \}$
	$\mid$	$L = E\ ;$	$\{\ gen(\boxed{L.array.base}\ '['\ \boxed{L.addr}\ ']'\ '='\ E.addr);$
E	$\rightarrow$	$E_1 + E_2$	$\{\ E.addr = \mathbf{new}\ Temp\,();$ $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \}$
	$\mid$	$\mathbf{id}$	$\{\ E.addr = top.get(\mathbf{id}.lexeme);\ \}$
	$\mid$	L	$\{\ E.addr = \mathbf{new}\ Temp\,();$ $gen(E.addr\ '='\ \boxed{L.array.base}\ '['\ \boxed{L.addr}\ ']');\ \}$

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



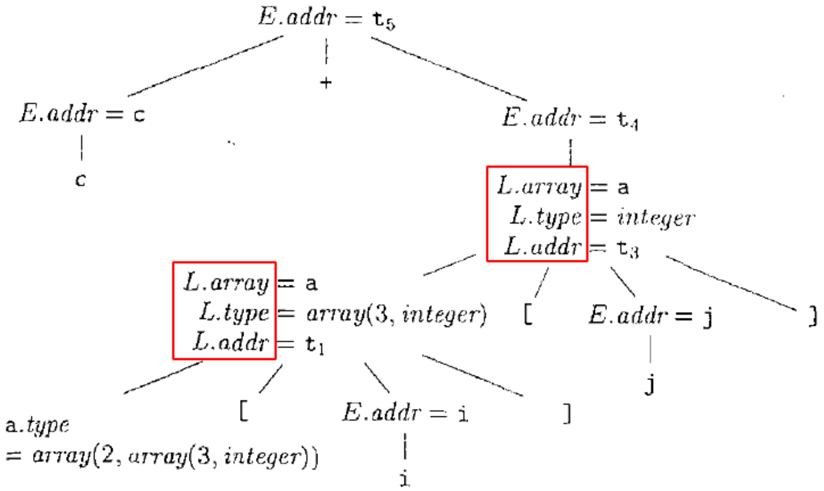
$$
\begin{array}{lll}
L \rightarrow \textbf{id}\ [\ E\ ] & \{ & \boxed{L.array} = top.get(\textbf{id}.lexeme); \\
 & & L.type = L.array.type.elem; \\
 & & L.addr = \textbf{new}\ Temp\,(); \\
 & & gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \} \\
 & & \\
\quad \mid\ \ L_1\ [\ E\ ] & \{ & \boxed{L.array}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

综合属性 $L.type$: (当前) 元素类型

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

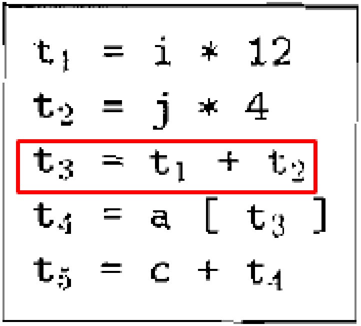
综合属性 $L.addr$：（当前）偏移地址

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



$E.addr = t_5$
$+$
$E.addr = c$
c
$E.addr = t_4$
$L.array = a$
$L.type = integer$
$L.addr = t_3$
$L.array = a$
$L.type = array(3, integer)$
$L.addr = t_1$
[
$E.addr = j$
]
j
a.type
$= array(2, array(3, integer))$
[
$E.addr = i$
]
i

int $a[2][3]$



```
t1 = i * 12
t2 = j * 4
t3 = t1 + t2
t4 = a [ t3 ]
t5 = c + t4
```

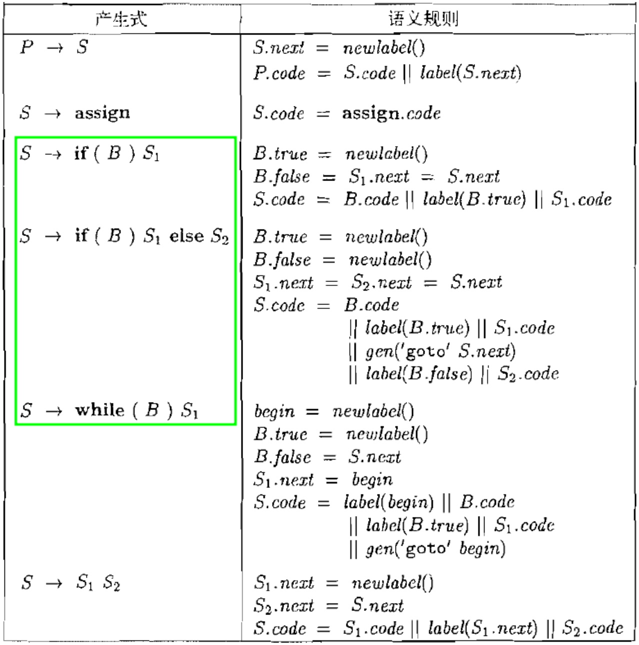
int $a[2][3]$

控制流语句的中间代码翻译

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

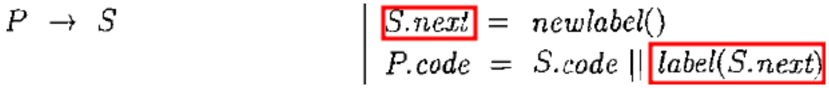
$S \rightarrow \textbf{if}\ (B)\ S_1\ \textbf{else}\ S_2$

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



产生式	语义规则
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \;\|\|\; label(S.next)$
$S \rightarrow \textbf{assign}$	$S.code = \textbf{assign}.code$
$S \rightarrow \textbf{if}\;(\;B\;)\;S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \;\|\|\; label(B.true) \;\|\|\; S_1.code$
$S \rightarrow \textbf{if}\;(\;B\;)\;S_1\;\textbf{else}\;S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\|\|\; label(B.true) \;\|\|\; S_1.code$ $\|\|\; gen('\texttt{goto}'\; S.next)$ $\|\|\; label(B.false) \;\|\|\; S_2.code$
$S \rightarrow \textbf{while}\;(\;B\;)\;S_1$	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \;\|\|\; B.code$ $\|\|\; label(B.true) \;\|\|\; S_1.code$ $\|\|\; gen('\texttt{goto}'\; begin)$
$S \rightarrow S_1\; S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \;\|\|\; label(S_1.next) \;\|\|\; S_2.code$

继承属性 $S.next$: S 的下一条指令



$P \rightarrow S$	$\boxed{S.next} = newlabel()$ $P.code = S.code \,\|\|\, \boxed{label(S.next)}$

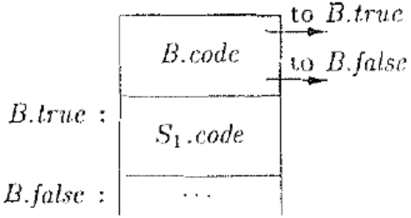
$S.next$ 为语句 S 指明了"跳出" S 的目标

$$S \rightarrow \mathbf{assign} \qquad \Big| \; S.code = \mathbf{assign}.code$$

表达式语句的翻译, 包括数组引用

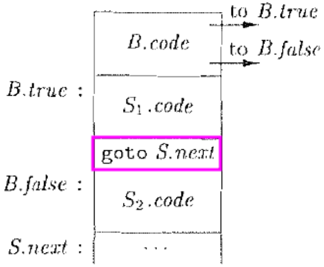
$S \rightarrow \textbf{if}\ (\ B\ )\ S_1$

$$
\begin{array}{|lcl}
B.true & = & newlabel() \\
B.false & = & \boxed{S_1.next} = S.next \\
S.code & = & B.code \ || \ label(B.true) \ || \ S_1.code
\end{array}
$$



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

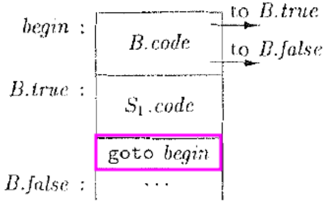
$S \rightarrow \textbf{if} (B) S_1 \textbf{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $\boxed{S_1.next = S_2.next = S.next}$ $S.code = B.code$ $\quad \| \ label(B.true) \ \| \ S_1.code$ $\quad \| \ \boxed{gen('\texttt{goto}' \ S.next)}$ $\quad \| \ label(B.false) \ \| \ S_2.code$



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

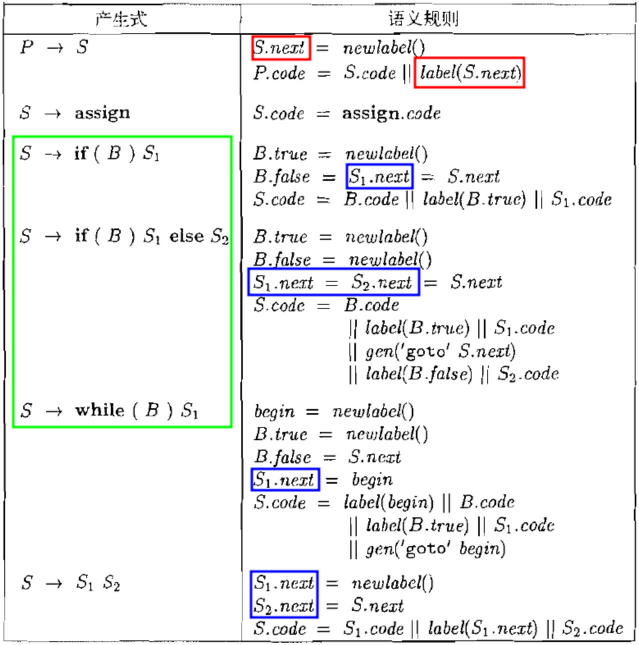
$S \rightarrow \textbf{while} \ (\ B \ ) \ S_1$

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



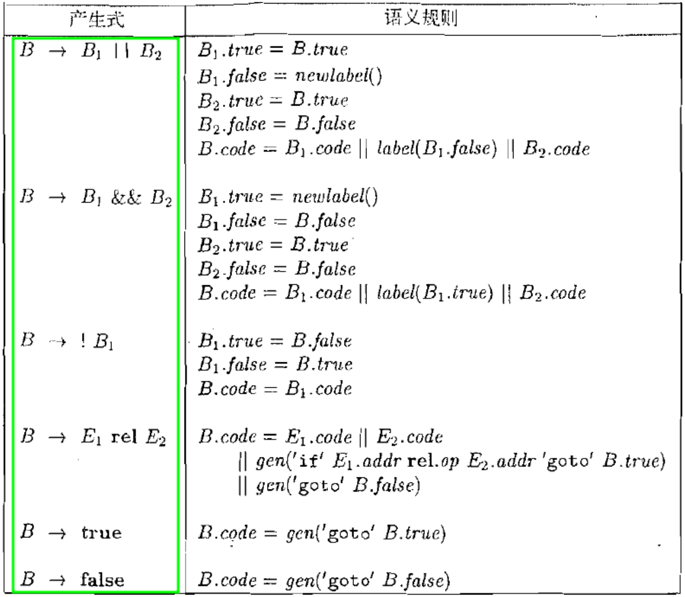
```
while (B)
  if (B) assign else assign
```

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



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

布尔表达式的中间代码翻译



产生式	语义规则
$B \to B_1 \,\|\|\, B_2$	$B_1.true = B.true$ $B_1.false = newlabel()$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \,\|\|\, label(B_1.false) \,\|\|\, B_2.code$
$B \to B_1 \,\&\&\, B_2$	$B_1.true = newlabel()$ $B_1.false = B.false$ $B_2.true = B.true$ $B_2.false = B.false$ $B.code = B_1.code \,\|\|\, label(B_1.true) \,\|\|\, B_2.code$
$B \to\, !\, B_1$	$B_1.true = B.false$ $B_1.false = B.true$ $B.code = B_1.code$
$B \to E_1 \;\textbf{rel}\; E_2$	$B.code = E_1.code \,\|\|\, E_2.code$ $\|\| \; gen('\texttt{if}'\; E_1.addr\; \textbf{rel}.op\; E_2.addr\; '\texttt{goto}'\; B.true)$ $\|\| \; gen('\texttt{goto}'\; B.false)$
$B \to \textbf{true}$	$B.code = gen('\texttt{goto}'\; B.true)$
$B \to \textbf{false}$	$B.code = gen('\texttt{goto}'\; B.false)$

$$
\begin{array}{lll|l}
B & \rightarrow & \textbf{true} & B.code = gen('\texttt{goto}'\ \boxed{B.true}) \\
B & \rightarrow & \textbf{false} & B.code = gen('\texttt{goto}'\ \boxed{B.false})
\end{array}
$$

`if (true)` **assign**

$$
S \rightarrow \textbf{if}\ (\ B\ )\ S_1 \quad
\left|
\begin{array}{lcl}
B.true & = & newlabel() \\
B.false & = & \boxed{S_1.next} = S.next \\
S.code & = & B.code \ || \ label(B.true) \ || \ S_1.code
\end{array}
\right.
$$

`if (false)` **assign**

$$B \rightarrow \; ! \; B_1 \qquad \left| \begin{array}{l} \boxed{\begin{array}{l} B_1.true \\ B_1.false \end{array}} \begin{array}{l} = B.false \\ = B.true \end{array} \\ B.code = B_1.code \end{array} \right.$$

`if (!true)` **assign**

$$S \rightarrow \textbf{if} \, (\, B \,) \, S_1 \qquad \left| \begin{array}{lcl} B.true & = & newlabel() \\ B.false & = & \boxed{S_1.next} = S.next \\ S.code & = & B.code \, || \, label(B.true) \, || \, S_1.code \end{array} \right.$$

`if (!false)` **assign**

短路求值

$$B \rightarrow B_1 \mid\mid B_2 \quad \left| \begin{array}{l} \boxed{B_1.true} = B.true \\ B_1.false = newlabel() \\ \boxed{B_2.true} = B.true \\ B_2.false = B.false \\ B.code = B_1.code \mid\mid label(B_1.false) \mid\mid B_2.code \end{array} \right.$$

`if (true || false)` **assign**

$$S \rightarrow \textbf{if}\ (\ B\ )\ S_1 \quad \left| \begin{array}{l} B.true = newlabel() \\ B.false = \boxed{S_1.next} = S.next \\ S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code \end{array} \right.$$

`if (false || true)` **assign**

短路求值

$$
B \rightarrow B_1 \;\&\&\; B_2 \quad
\begin{array}{|l}
B_1.true = newlabel() \\
\boxed{B_1.false} = B.false \\
B_2.true = B.true \\
\boxed{B_2.false} = B.false \\
B.code = B_1.code \;||\; label(B_1.true) \;||\; B_2.code
\end{array}
$$

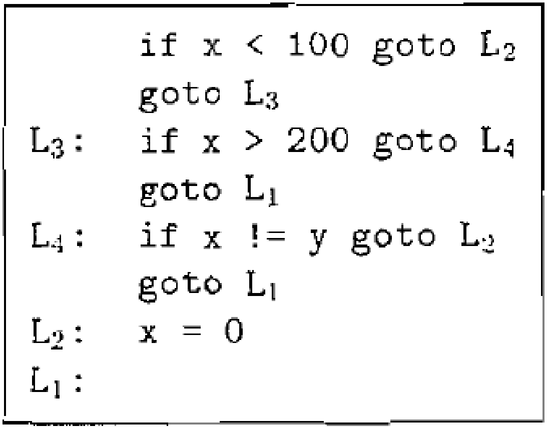
```
if (true && false) assign
```

$$
S \rightarrow \textbf{if} \; (\; B \;) \; S_1 \quad
\begin{array}{|l}
B.true = newlabel() \\
B.false = \boxed{S_1.next} = S.next \\
S.code = B.code \;||\; label(B.true) \;||\; S_1.code
\end{array}
$$

```
if (false && true) assign
```

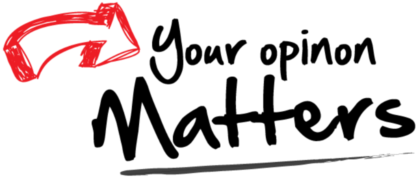
$$
\begin{array}{lcl|l}
B & \to & E_1 \ \textbf{rel} \ E_2 & B.code = E_1.code \ || \ E_2.code \\
 & & & \quad || \ \boxed{gen}(\texttt{'if'} \ E_1.addr \ \textbf{rel}.op \ E_2.addr \ \texttt{'goto'} \ B.true) \\
 & & & \quad || \ \boxed{gen}(\texttt{'goto'} \ B.false)
\end{array}
$$

```
if (x < 100 || x > 200 && x != y) x = 0;
```



```
         if x < 100 goto L2
         goto L3
L3 :     if x > 200 goto L4
         goto L1
L4 :     if x != y goto L2
         goto L1
L2 :     x = 0
L1 :
```

Thank You!



Office 926

hfwei@nju.edu.cn