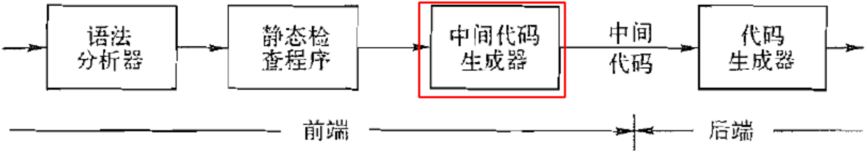
中间代码生成
(1. 表达式的翻译与控制流的翻译)

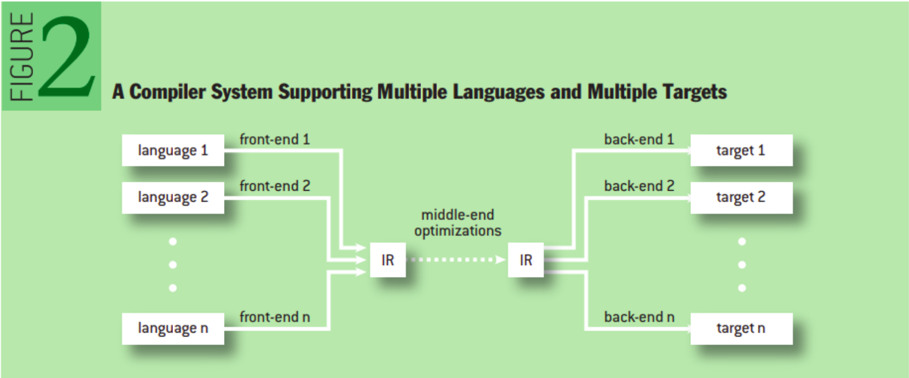
魏恒峰

hfwei@nju.edu.cn

2021 年 12 月 17 日

Intermediate Representation (IR)



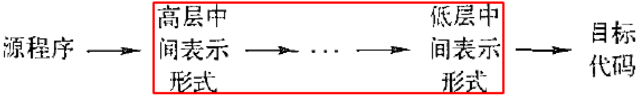


The Increasing Significance of Intermediate Representations in Compilers (Fred Chow; 2013)

LLVM 的核心就是它的 LLVM IR

(希望下一轮授课可以加入 LLVM IR 内容)

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)

华为方舟编译器的 Maple IR 采用多层设计

Intermediate Representation (IR)

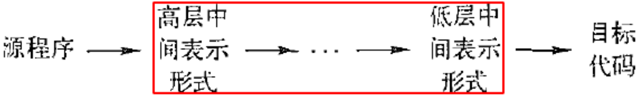


图 (抽象语法树)、**三地址代码**、C 语言

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

每个 TAC **指令**最多包含三个**地址**。

地址: 源程序中的名字、常量、编译器生成的临时变量

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

每个 TAC **指令**最多包含三个**地址**。

5 组 16 条**指令**格式 (x, y, z 为**地址**):

$$x = y \ \mathbf{op} \ z \tag{1}$$
$$x = \mathbf{op} \ y \tag{2}$$
$$x = y \tag{3}$$

$$\mathbf{goto} \ L \tag{4}$$
$$\mathbf{if} \ x \ \mathbf{goto} \ L \tag{5}$$
$$\mathbf{if\ False} \ x \ \mathbf{goto} \ L \tag{6}$$
$$\mathbf{if} \ x \ \mathbf{relop} \ y \ \mathbf{goto} \ L \tag{7}$$

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

每个 TAC **指令**最多包含三个**地址**。

5 组 16 条**指令**格式 (x, y, z 为**地址**):

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

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

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

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

$\texttt{param}\ x_1$

$\texttt{param}\ x_2$

$\dots$

$\texttt{param}\ x_n$

$\texttt{call}\ p, n$

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

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

每个 TAC **指令**最多包含三个**地址**。

5 组 16 条**指令**格式 (x, y, z 为**地址**):

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

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

距离位置 x/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
```

为 (某些) 控制语句生成**带有位置号**的中间代码时, 需要采用**回填技术**

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	arg1	arg2	result
0	minus	c		t1
1	*	b	t1	t2
2	minus	c		t3
3	*	b	t3	t4
4	+	t2	t4	t5
5	=	t5		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
$S \rightarrow \textbf{id} = E\ ;$	$S.code = E.code \ \Vert$ $gen(\boxed{top.get(\textbf{id}.lexeme)}\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \textbf{new}\ Temp\,()$ $E.code = E_1.code \ \Vert\ E_2.code \ \Vert$ $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ -\ E_1$	$E.addr = \textbf{new}\ Temp\,()$ $E.code = E_1.code \ \Vert$ $gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	$E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \textbf{id}$	$E.addr = \boxed{top.get(\textbf{id}.lexeme)}$ 符号表条目 $E.code =\ ''$

综合属性 $E.code$ 与 $E.addr$

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

$$
\begin{array}{llll}
S & \rightarrow & \mathbf{id} = E \;; & \{\ gen(\ top.get(\mathbf{id}.lexeme)\ '='\ E.addr);\ \} \\
E & \rightarrow & E_1 + E_2 & \{\ E.addr = \mathbf{new}\ Temp\,();\ \\
 & & & \quad gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr);\ \} \\
 & | & -\,E_1 & \{\ E.addr = \mathbf{new}\ Temp\,(); \\
 & & & \quad gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr);\ \} \\
 & | & (\ E_1\ ) & \{\ E.addr = E_1.addr;\ \} \\
 & | & \mathbf{id} & \{\ E.addr = top.get(\mathbf{id}.lexeme);\ \}
\end{array}
$$

综合属性 $E.addr$

假想一个全局指令缓冲区, 对 gen 的连续调用将生成一个指令序列

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**

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

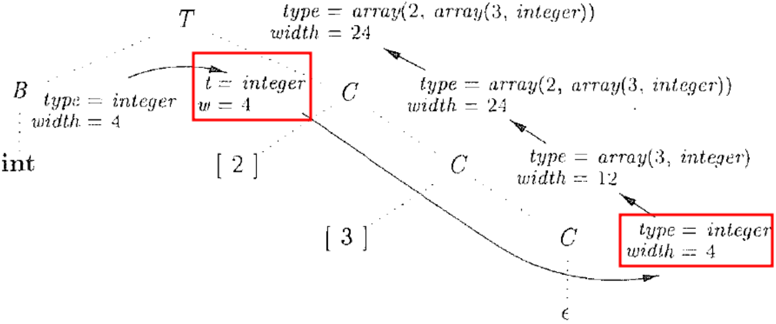


图 6-16　数组类型的语法制导翻译

数组类型声明

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

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

	类型	宽度
a	$array(2, array(3, \text{integer}))$	24
$a[i]$	$array(3, \text{integer})$	12
$a[i][j]$	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	$\rightarrow$	$\boxed{\mathbf{id}\ [\ E\ ]}$	$\{\ L.array = top.get(\mathbf{id}.lexeme);$ $L.type = L.array.type.elem;$ $L.addr = \mathbf{new}\ Temp\,();$ $gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \}$
	$\mid$	$\boxed{L_1\ [\ E\ ]}$	$\{\ L.array = L_1.array;$ $L.type = L_1.type.elem;$ $t = \mathbf{new}\ Temp\,();$ $L.addr = \mathbf{new}\ Temp\,();$ $gen(t\ '='\ E.addr\ '*'\ L.type.width);$ $gen(L.addr\ '='\ L_1.addr\ '+'\ t);\ \}$

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

综合属性 $L.array.base$：数组基地址（即，数组名）

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

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

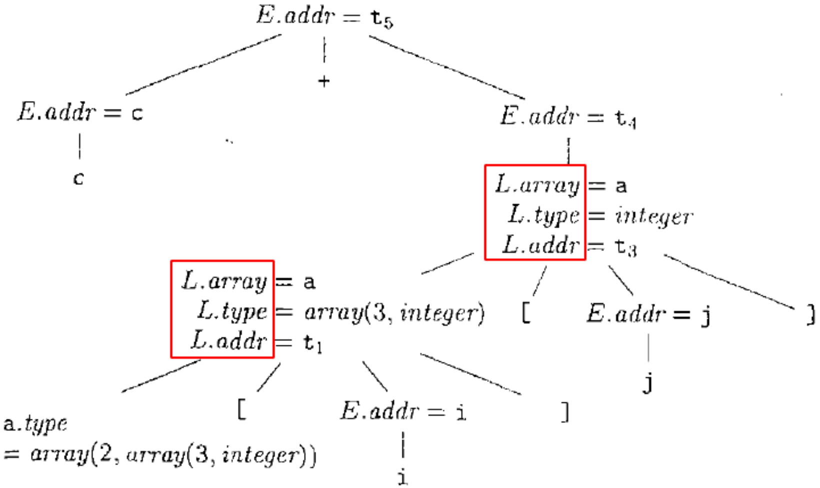
$$
\begin{array}{lll}
L \rightarrow \mathbf{id}\ [\ E\ ] & \{ & \boxed{L.array} = top.get(\mathbf{id}.lexeme); \\
 & & L.type = L.array.type.elem; \\
 & & L.addr = \mathbf{new}\ Temp\,(); \\
 & & gen(L.addr\ '='\ E.addr\ '*'\ L.type.width);\ \} \\
\\
\quad \mid\ L_1\ [\ E\ ] & \{ & \boxed{L.array} =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

综合属性 $L.type$：(当前) 元素类型

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

综合属性 $L.addr$: (当前) 偏移地址

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


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

$$c + a[2][3]$$

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

$c + a[2][3]$

控制流语句与布尔表达式的中间代码翻译

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

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

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

布尔表达式的作用：布尔值 *vs.* 控制流跳转

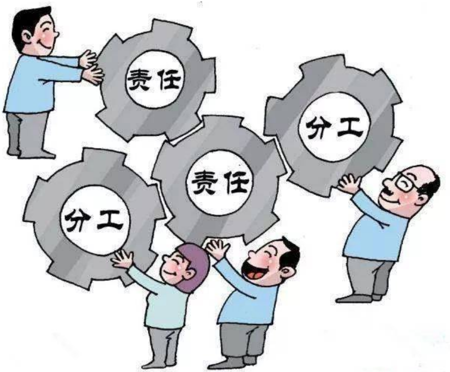
$$S \to \boxed{\textbf{id} = E;} \mid \boxed{\textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S}$$

$$E \to \boxed{E \parallel E \mid E \&\& E \mid E\ \textbf{rel}\ E} \mid E + E \mid \boxed{(E)} \mid \textbf{id} \mid \boxed{\textbf{true} \mid \textbf{false}}$$

我们先关注“控制流跳转”

控制流语句与布尔表达式的中间代码翻译

分工明确　　各司其职



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

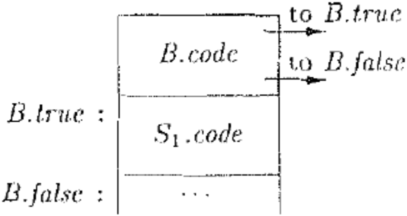
$$P \rightarrow S \qquad \begin{array}{|lcl} \boxed{S.next} & = & newlabel() \\ P.code & = & S.code \,||\, \boxed{label(S.next)} \end{array}$$

$S.next$ 为语句 S 指明了“跳出” S 的目标

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

代表了表达式的翻译, 包括数组引用

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

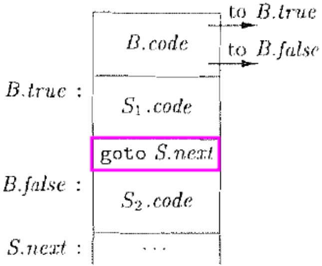



```
if (true)
  if (false) assign
```

$$P \to S$$

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

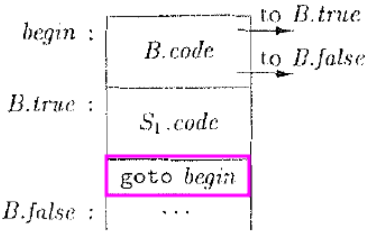
$S \rightarrow \textbf{if} \ (\ B \ ) \ S_1 \ \textbf{else} \ S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $\boxed{S_1.next = S_2.next = S.next}$ $S.code = B.code$ $\quad \| \ label(B.true) \ \| \ S_1.code$ $\quad \| \ \boxed{gen(\text{'goto'} \ S.next)}$ $\quad \| \ label(B.false) \ \| \ S_2.code$




```
if (true)
  if (true) 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 (true)
  if (false) assign else assign
```

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

`if (true)` **assign** `else` **assign** **assign**

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

布尔表达式的中间代码翻译

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

$$B \to \textbf{true} \qquad B.code = gen('\text{goto}'\ \boxed{B.true})$$

$$B \to \textbf{false} \qquad B.code = gen('\text{goto}'\ \boxed{B.false})$$

`if (true)` **assign**

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

`if (false)` **assign**

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

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

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

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

短路求值

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

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

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

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

短路求值

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

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

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

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

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

```
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 :
```

布尔表达式的作用: 布尔值 $vs.$ 控制流跳转

$$S \rightarrow \boxed{\textbf{id} = E;} \mid \boxed{\textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S}$$

$$E \rightarrow \boxed{E \parallel E \mid E \&\& E \mid E\ \textbf{rel}\ E} \mid E + E \mid \boxed{(E)} \mid \textbf{id} \mid \boxed{\textbf{true} \mid \textbf{false}}$$

根据 E 所处的上下文判断 E 所扮演的角色, 调用不同的代码生成函数

函数 $jump(t, f)$: 生成控制流代码

函数 $rvalue()$: 生成计算布尔值的代码, 并将结果存储在临时变量中

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 $E.code = ''$

$$E \to E_1 \&\& E_2$$

为 E 生成**跳转代码**, 在**真假出口处**将 **true** 或 **false** 存储到临时变量

```
x = a < b && c < d
```

```
        ifFalse a < b goto L1
        ifFalse c < d goto L1
        t = true
        goto L2
L1:     t = false
L2:     x = t
```

布尔表达式的作用: 布尔值 *vs.* 控制流跳转

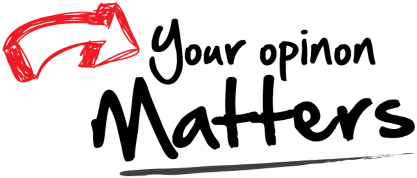
$$S \rightarrow \boxed{\textbf{id} = E;} \mid \boxed{\textbf{if}\ (E)\ S \mid \textbf{while}\ (E)\ S \mid S\ S}$$

$$E \rightarrow \boxed{E \parallel E \mid E \&\& E \mid E\ \textbf{rel}\ E} \mid E + E \mid \boxed{(E)} \mid \textbf{id} \mid \boxed{\textbf{true} \mid \textbf{false}}$$

为什么要区分布尔表达式的这两种角色?

可以不区分布尔表达式的这两种角色吗?


Thank
You!



Office 926

hfwei@nju.edu.cn