

# 四、中间代码生成

## (12. 控制流语句的翻译 (Hard 模式))

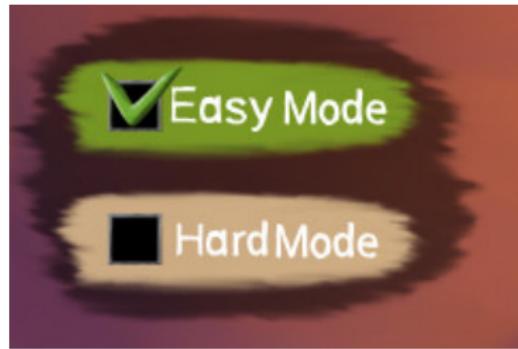
魏恒峰

hfwei@nju.edu.cn

2024 年 05 月 15 日



各个非终结符的**分工、合作**明确易懂



只需要使用**综合属性**

# 分工 合作



为布尔表达式  $B$  计算逻辑值 (假设保存在临时变量  $t1$  中)

**if**、**while** 等语句根据  $B$  的结果改变控制流

**if** ( $B$ )  $S_1$

## 如何生成更短、更高效的代码？

```
1 while (true) {  
2     if (false) {  
3         x = 1;  
4     }  
5 }
```

```
1 begin1  
2 t1 = true  
3 br t1 b.true2 b.false3  
4 b.true2:  
5 t2 = false  
6 br t2 b.true4 b.false5  
7 b.true4:  
8 t3 = 1  
9 x = t3  
10 b.false5:  
11 br begin1  
12 b.false3:
```

## 如何生成更短、更高效的代码？

```
1  t1 = true
2  br t1 or.true1 or.false2
3  or.false2:
4  t2 = false
5  t3 = OR t1 t2
6  br or.end3
7  or.true1:
8  t3 = true
9  or.end3:
10 br t3 b.true4 b.false5
11 b.true4:
12 t4 = 1
13 x = t4
14 b.false5:
```

## 直接用布尔表达式改变控制流



使用控制流跳转间接表明布尔表达式的值

# 分工      合作



父节点为子节点准备跳转指令的目标标签

子节点通过**继承属性**确定跳转目标

$$P \rightarrow S \quad S \rightarrow \mathbf{if} (B) \; S_1$$

## Control.g4

产生式
$P \rightarrow S$
$S \rightarrow \text{assign}$
$S \rightarrow \text{if} ( B ) S_1$
$S \rightarrow \text{if} ( B ) S_1 \text{ else } S_2$
$S \rightarrow \text{while} ( B ) S_1$
$S \rightarrow S_1 S_2$

产生式
$B \rightarrow B_1 \uparrow\downarrow B_2$
$B \rightarrow B_1 \&\& B_2$
$B \rightarrow ! B_1$
$B \rightarrow E_1 \text{ rel } E_2$
$B \rightarrow \text{true}$
$B \rightarrow \text{false}$

## 继承属性 $B.\text{true}$ , $B.\text{false}$ , $S.\text{next}$ 指明了控制流跳转目标

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

$P \rightarrow S$

$S.next = newlabel()$
$P.code = S.code \parallel label(S.next)$

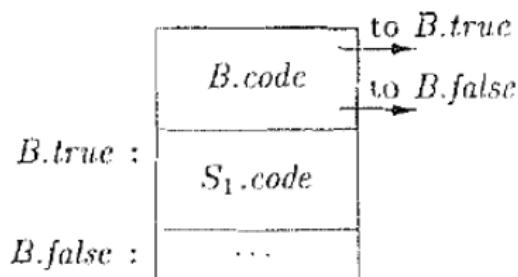
$S.next$  为语句  $S$  指明了“跳出”  $S$  的目标

$S \rightarrow \text{assign} \quad | \quad S.\text{code} = \text{assign}.\text{code}$

代表了表达式的翻译, 包括数组引用

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

$B.\text{true} = \text{newlabel}()$   
 $B.\text{false} = S_1.\text{next} = S.\text{next}$   
 $S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$

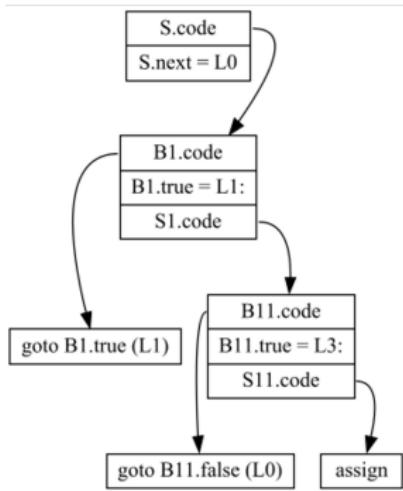


$|S \rightarrow \text{if } (B) S_1$

**B.true** = newlabel()

$$B.\text{false} = S_1.\text{next} = S.\text{next}$$

$$S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$$



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

*B* → true

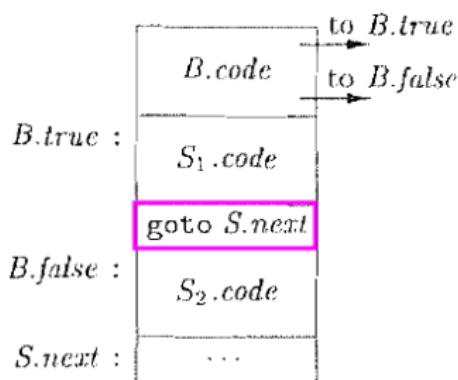
*B.code = gen('goto' B.true)*

*B* → false

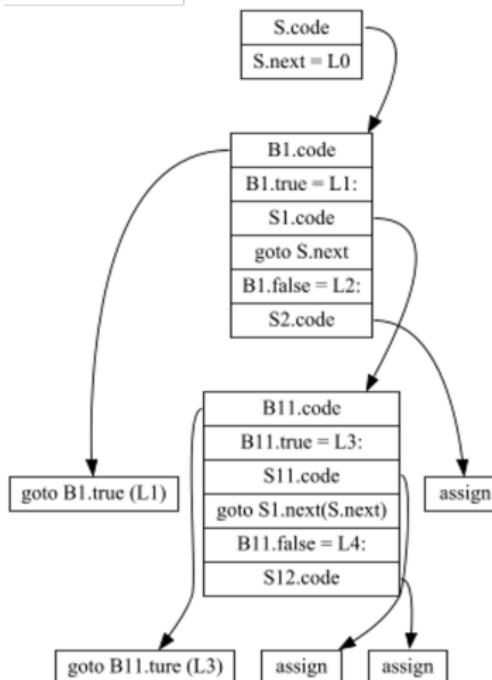
*B*.code = gen('goto' *B*.false)

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

$B.\text{true} = \text{newlabel}()$   
 $B.\text{false} = \text{newlabel}()$   
 $S_1.\text{next} = S_2.\text{next} = S.\text{next}$   
 $S.\text{code} = B.\text{code}$   
||  $\text{label}(B.\text{true}) || S_1.\text{code}$   
||  $\text{gen('goto' } S.\text{next})$   
||  $\text{label}(B.\text{false}) || S_2.\text{code}$



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



$B.\text{true} = \text{newlabel}()$

$B.\text{false} = \text{newlabel}()$

$S_1.\text{next} = S_2.\text{next} = S.\text{next}$

$S.\text{code} = B.\text{code}$

$\parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$

$\parallel \text{gen('goto' } S.\text{next})$

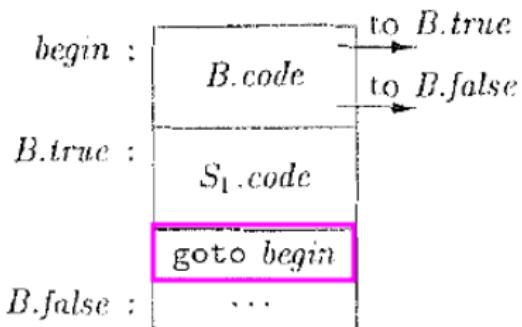
$\parallel \text{label}(B.\text{false}) \parallel S_2.\text{code}$

```

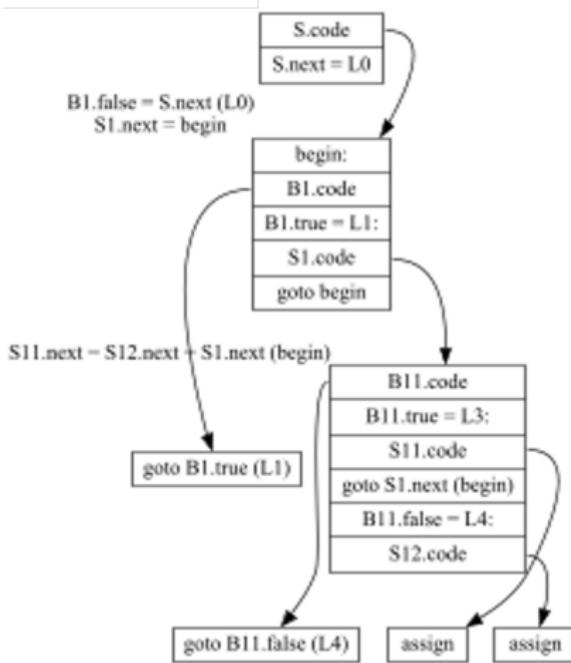
if (true)
  if (true) assign else assign
else
  assign
  
```

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

```
begin = newlabel()
B.true = newlabel()
B.false = S.next
S1.next = begin
S.code = label(begin) || B.code
|| label(B.true) || S1.code
|| gen('goto' begin)
```



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



```

begin = newlabel()
B.true = newlabel()
B.false = S.next
S1.next = begin
S.code = label(begin) || B.code
          || label(B.true) || S1.code
          || gen('goto' begin)

```

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

$S \rightarrow S_1 S_2$

$S_1.next = newlabel()$

$S_2.next = S.next$

$S.code = S_1.code || label(S_1.next) || S_2.code$

$S \rightarrow S_1 S_2$

$S_1.next = newlabel()$

$S_2.next = S.next$

$S.code = S_1.code || label(S_1.next) || S_2.code$

if (true) assign else assign assign

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

$B \rightarrow \text{true}$

$B.\underline{code} = \text{gen('goto' } B.\text{true})$

$B \rightarrow \text{false}$

$B.\underline{code} = \text{gen('goto' } B.\text{false})$

$B \rightarrow \text{true}$  $B.\text{code} = \text{gen('goto' } B.\text{true})$  $B \rightarrow \text{false}$  $B.\text{code} = \text{gen('goto' } B.\text{false})$ 

if (true) assign

 $S \rightarrow \text{if ( } B \text{ ) } S_1$  $B.\text{true} = \text{newlabel()}$  $B.\text{false} = S_1.\text{next} = S.\text{next}$  $S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$ 

if (false) assign

$B \rightarrow !B_1$

$| \cdot B_1.true = B.false$   
 $| \cdot B_1.false = B.true$   
 $| \cdot B.code = B_1.code$

$B \rightarrow !B_1$ 
$$\begin{cases} B_1.\text{true} = B.\text{false} \\ B_1.\text{false} = B.\text{true} \\ B.\text{code} = B_1.\text{code} \end{cases}$$

if (!true) assign

 $S \rightarrow \text{if} ( B ) S_1$ 
$$\begin{cases} B.\text{true} = \text{newlabel}() \\ B.\text{false} = S_1.\text{next} = S.\text{next} \\ S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code} \end{cases}$$

if (!false) assign

## 短路求值

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

## 短路求值

$B \rightarrow B_1 \text{    } B_2$	$B_1.\text{true} = B.\text{true}$
	$B_1.\text{false} = \text{newlabel}()$
	$B_2.\text{true} = B.\text{true}$
	$B_2.\text{false} = B.\text{false}$
	$B.\text{code} = B_1.\text{code} \parallel \text{label}(B_1.\text{false}) \parallel B_2.\text{code}$

if (true || false) assign

$S \rightarrow \text{if} ( B ) S_1$	$B.\text{true} = \text{newlabel}()$
	$B.\text{false} = S_1.\text{next} = S.\text{next}$
	$S.\text{code} = B.\text{code} \parallel \text{label}(B.\text{true}) \parallel S_1.\text{code}$

if (false || true) assign

## 短路求值

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

## 短路求值

$B \rightarrow B_1 \&\& B_2$	$B_1.true = newlabel()$
	$B_1.false = B.false$
	$B_2.true = B.true$
	$B_2.false = B.false$
	$B.code = B_1.code \parallel label(B_1.true) \parallel B_2.code$

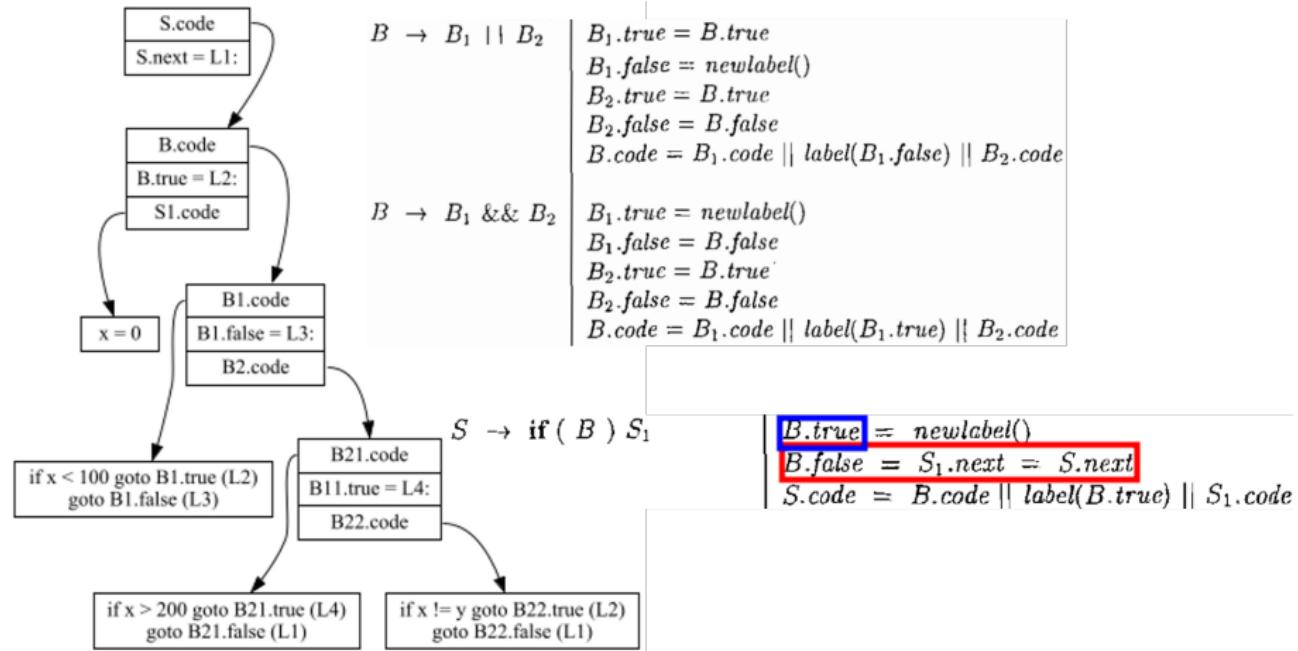
if (true && false) assign

$S \rightarrow \text{if} (B) S_1$	$B.true = newlabel()$
	$B.false = S_1.next = S.next$
	$S.code = B.code \parallel label(B.true) \parallel S_1.code$

if (false && true) assign

$$B \rightarrow E_1 \text{ rel } E_2 \quad \left| \begin{array}{l} B.code = E_1.code \parallel E_2.code \\ \parallel \boxed{\text{gen('if' } E_1.\text{addr rel.op } E_2.\text{addr 'goto' } B.\text{true)}} \\ \parallel \boxed{\text{gen('goto' } B.\text{false})} \end{array} \right.$$

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



```
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:
```

```
clang -S -emit-llvm if-boolexpr.c -o if-boolexpr-opt0.ll
```

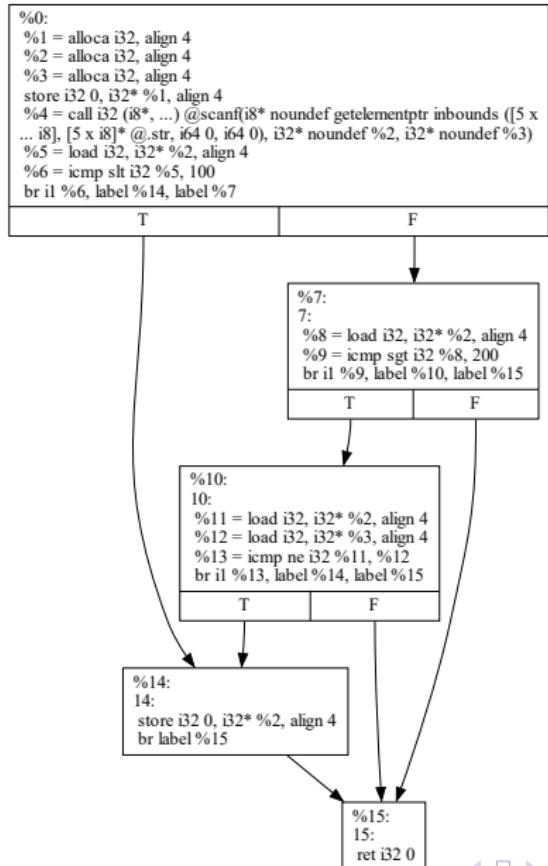
```
int main() {
    int x, y;
    scanf(Format: "%d%d", &x, &y);

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

    return 0;
}
```

```
opt -dot-cfg if-boolexpr-opt0.ll
```

```
dot -Tpdf .main.dot -o if-boolexpr-opt0-cfg.pdf
```



## CodeGenListener.java



## CodeGenListener.java



# CODEGEN

思考：如何实现增量式翻译？

```
1 while (true) {  
2     if (false) {  
3         x = 1;  
4     }  
5 }
```

```
1 L2begin:  
2 goto L3  
3 L3:  
4 goto L2begin  
5 L4:  
6 ASSIGN  
7 goto L2begin  
8 L1:
```

```
1 if (true || false) {  
2     x = 1;  
3 }
```

```
1 goto L2  
2 L3:  
3 goto L1  
4 L2:  
5 ASSIGN  
6 L1:
```

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

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

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('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('goto' begin)$
$S \rightarrow S_1 S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code    label(S_1.next)    S_2.code$

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

## 布尔表达式的作用: 布尔值 vs. 控制流跳转

$$S \rightarrow \text{id} = E ; \mid \text{if } (E) S \mid \text{while } (E) S \mid S S$$
$$E \rightarrow E \parallel E \mid E \& \& E \mid E \text{ rel } E \mid E + E \mid (E) \mid \text{id} \mid \text{true} \mid \text{false}$$

## 布尔表达式的作用: 布尔值 vs. 控制流跳转

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

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

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  $\text{jump}(t, f)$ : 生成控制流代码

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

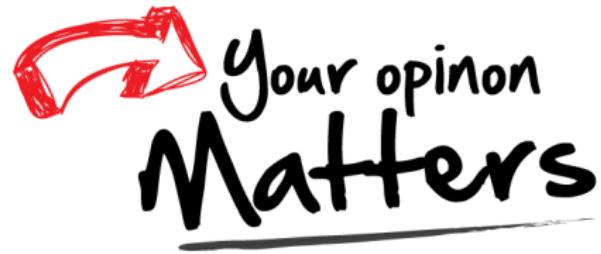
$x = a < b \&\& c < d$

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

$$E \rightarrow E_1 \&\& E_2$$

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

# Thank You!



Office 926

hfwei@nju.edu.cn