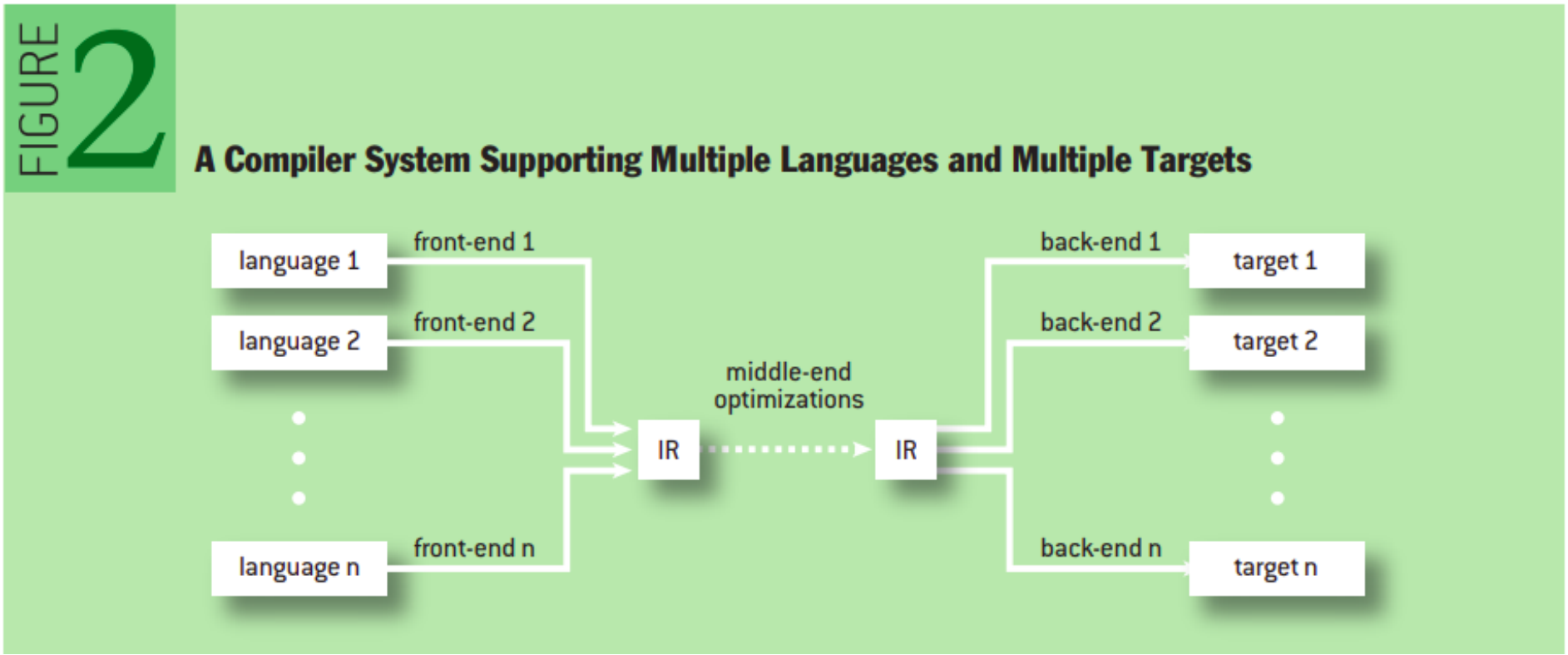
中间代码生成
(1. 表达式的翻译与控制流的翻译)

魏恒峰

hfwei@nju.edu.cn

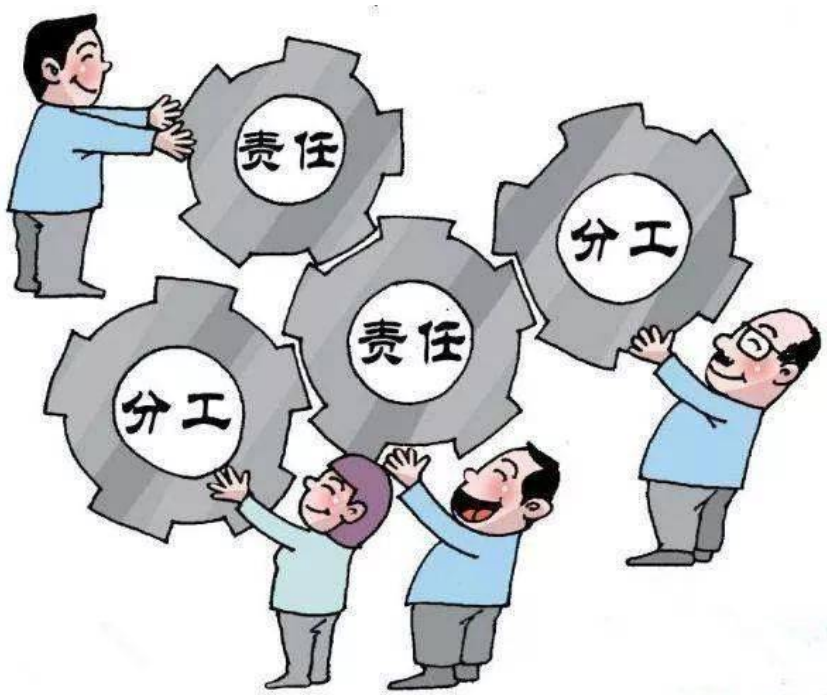
2022 年 12 月 26 日



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


腾讯视频 出品
大局观
好羊奶 选蓝河
对有大局观对对对
知乎 @Genoa

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

在自顶向下的分析过程中

为右部的每个 B 计算 B.true 与 B.false

为右部的每个 S 计算 S.next

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

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

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

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

表达式的中间代码翻译

综合属性 $E.code$: 中间代码

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

$$a = b + -c$$

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

综合属性 $E.addr$: 变量名 (包括临时变量)、常量

```c
int main() {
  int a = 0, b = 1, c = 2;

  a = b + -c;

  return 0;
}
```

```
%7 = sub nsw i32 0, %6
%8 = add nsw i32 %5, %7
store i32 %8, i32* %2, align 4
```

数组引用的中间代码翻译

声明：int $a[2][3]$

数组引用：$x = a[1][2];\ a[1][2] = x$

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

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

	类型	宽度
a	$array(2, array(3, \text{integer}))$	24
$a[i]$	$array(3, \text{integer})$	12
$a[i][j]$	integer	4

int $a[2][3]$

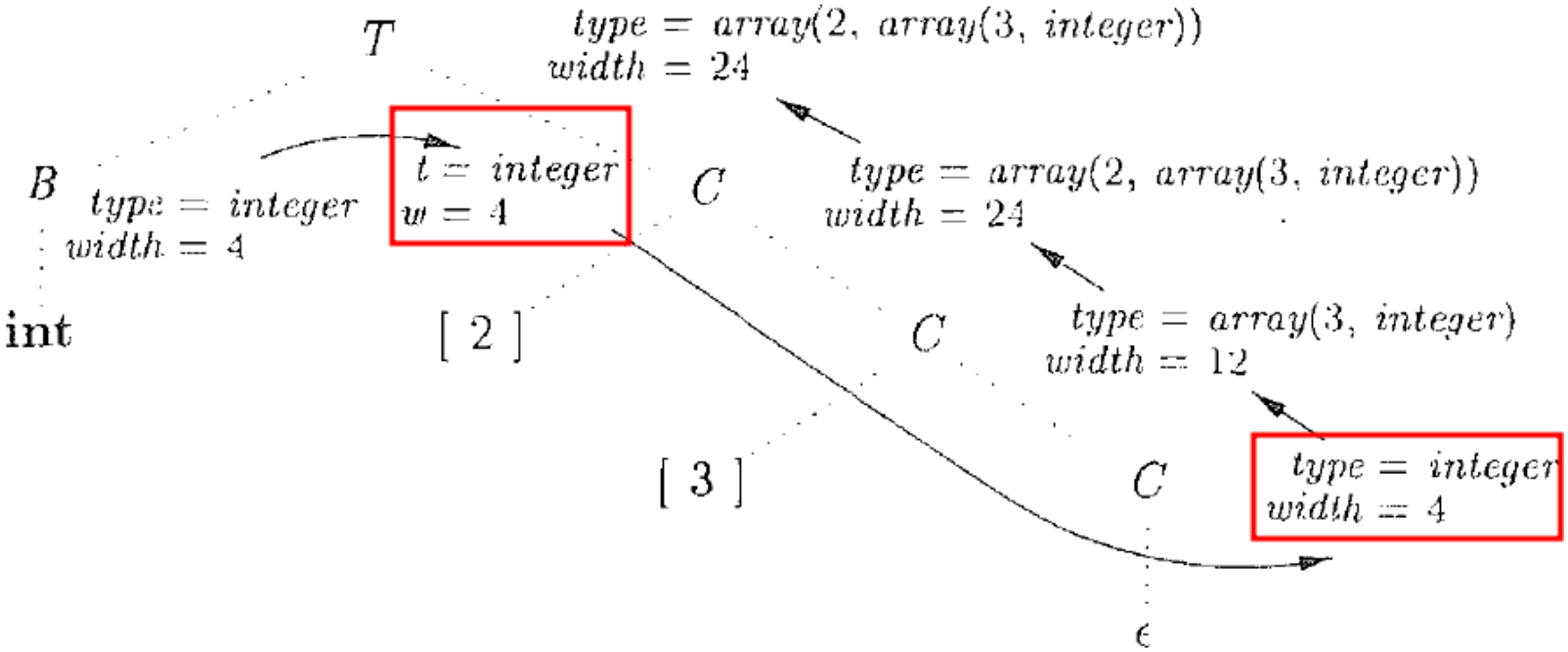
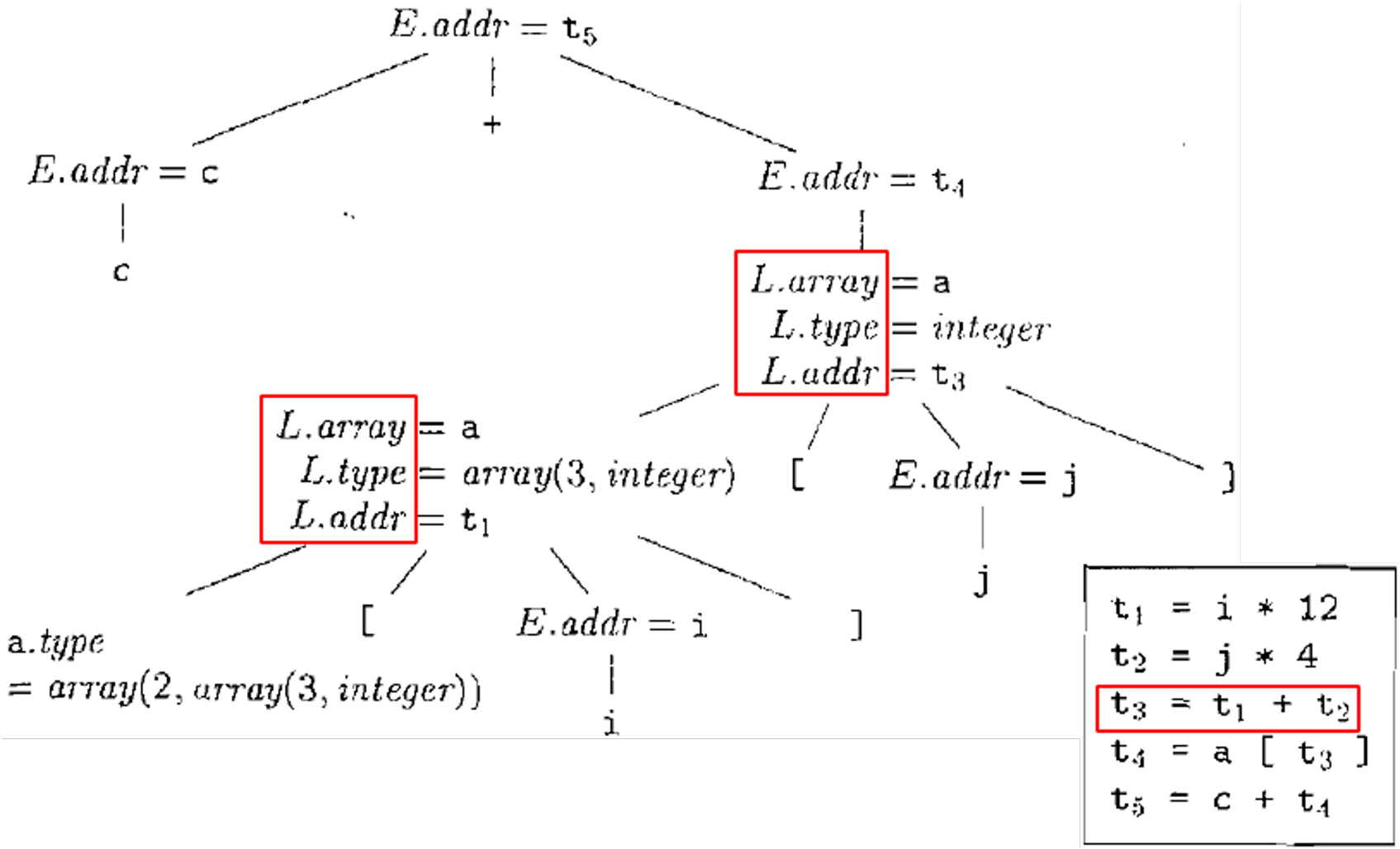


图 6-16　数组类型的语法制导翻译

综合属性 $L.array(.base)$：数组基地址（即，数组名）

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

综合属性 $L.addr$：偏移地址


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

$c + a[i][j]$

```
%2 = alloca [2 x [3 x i32]], align 16
```

```c
int main() {
  int a[2][3] = { 0 };

  int i = 1, j = 2;
  int c = 10, d = 20;

  d = c + a[i][j];

  return 0;
}
```

```
%8 = load i32, i32* %5, align 4 %8: c
%9 = load i32, i32* %3, align 4 %9: i
%10 = sext i32 %9 to i64
%11 = getelementptr inbounds [2 x [3 x i32]], [2 x [3 x i32]]* %2, i64 0, i64 %10
%12 = load i32, i32* %4, align 4 %12: j
%13 = sext i32 %12 to i64
%14 = getelementptr inbounds [3 x i32], [3 x i32]* %11, i64 0, i64 %13
%15 = load i32, i32* %14, align 4 %15: a[i][j]
%16 = add nsw i32 %8, %15
store i32 %16, i32* %6, align 4
```

The Often Misunderstood GEP Instruction

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

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

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

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

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

继承属性 $S.next$

$$P \rightarrow S \quad \left| \begin{array}{l} S.next = newlabel() \\ P.code = S.code \,||\, label(S.next) \end{array} \right.$$

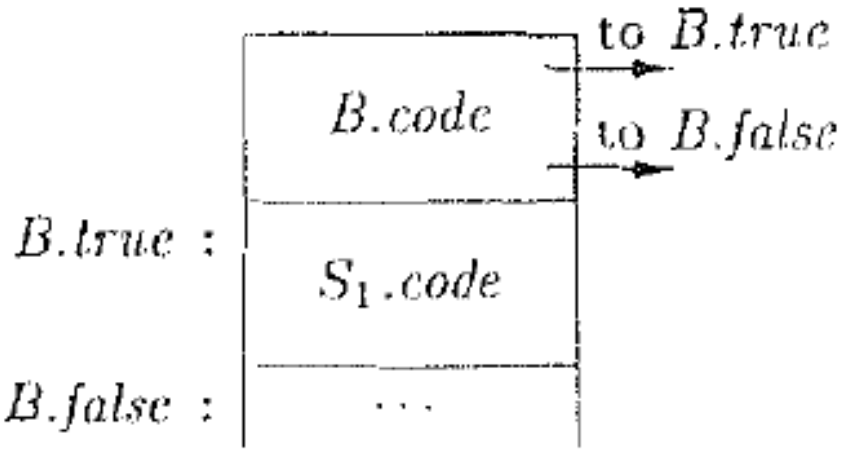
$S.next$ 为语句 S 指明了“跳出” S 的目标

$$S \rightarrow \textbf{assign} \quad \left| \quad S.code = \textbf{assign}.code \right.$$

代表了表达式的翻译, 包括数组引用

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

$B.true = newlabel()$
$B.false = S_1.next = S.next$
$S.code = B.code \,||\, label(B.true) \,||\, S_1.code$



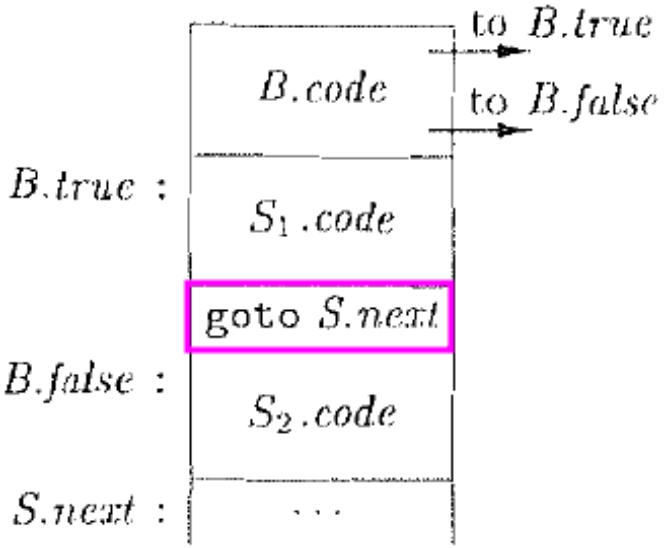

```
if (true)
  if (false) assign
```

$P \rightarrow S$

$B \rightarrow \textbf{true}$ | $B.code = gen('\texttt{goto}'\ B.true)$

$B \rightarrow \textbf{false}$ | $B.code = gen('\texttt{goto}'\ B.false)$

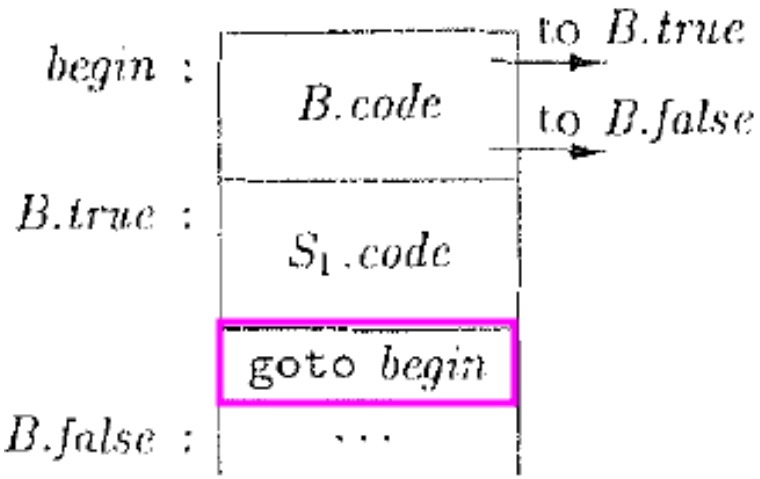
$S \rightarrow$ **if** (B) S_1 **else** S_2	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\quad \| \| label(B.true) \| \| S_1.code$ $\quad \| \| gen('\texttt{goto}'\ S.next)$ $\quad \| \| label(B.false) \| \| S_2.code$




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

$S \rightarrow$ **while** (B) S_1

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




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

$$
\begin{array}{lll}
S \rightarrow S_1\ S_2 & \boxed{S_1.next} = newlabel() \\
& \boxed{S_2.next} = S.next \\
& S.code = S_1.code \mid\mid label(S_1.next) \mid\mid S_2.code
\end{array}
$$

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

布尔表达式的中间代码翻译

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

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

if (true) assign

S	$\rightarrow$ **if** (B) S_1	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \mid\mid label(B.true) \mid\mid S_1.code$

if (false) assign

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

if (!true) assign

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

if (!false) assign

短路求值

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

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

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

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

短路求值

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

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

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

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

$$
\begin{array}{lll}
B \rightarrow E_1 \ \mathbf{rel} \ E_2 & \Big| & B.code = E_1.code \ || \ E_2.code \\
& & \quad || \ gen(\text{'}\mathtt{if}\text{'} \ E_1.addr \ \mathbf{rel}.op \ E_2.addr \ \text{'}\mathtt{goto}\text{'} \ B.true) \\
& & \quad || \ gen(\text{'}\mathtt{goto}\text{'} \ B.false)
\end{array}
$$

```
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

$S \rightarrow$ **id** $= E$; | **if** (E) S | **while** (E) S | S S

$E \rightarrow E \,\|\, E \mid E \,\&\&\, E \mid E$ **rel** $E \mid E + E \mid (E) \mid$ **id** | **true** | **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

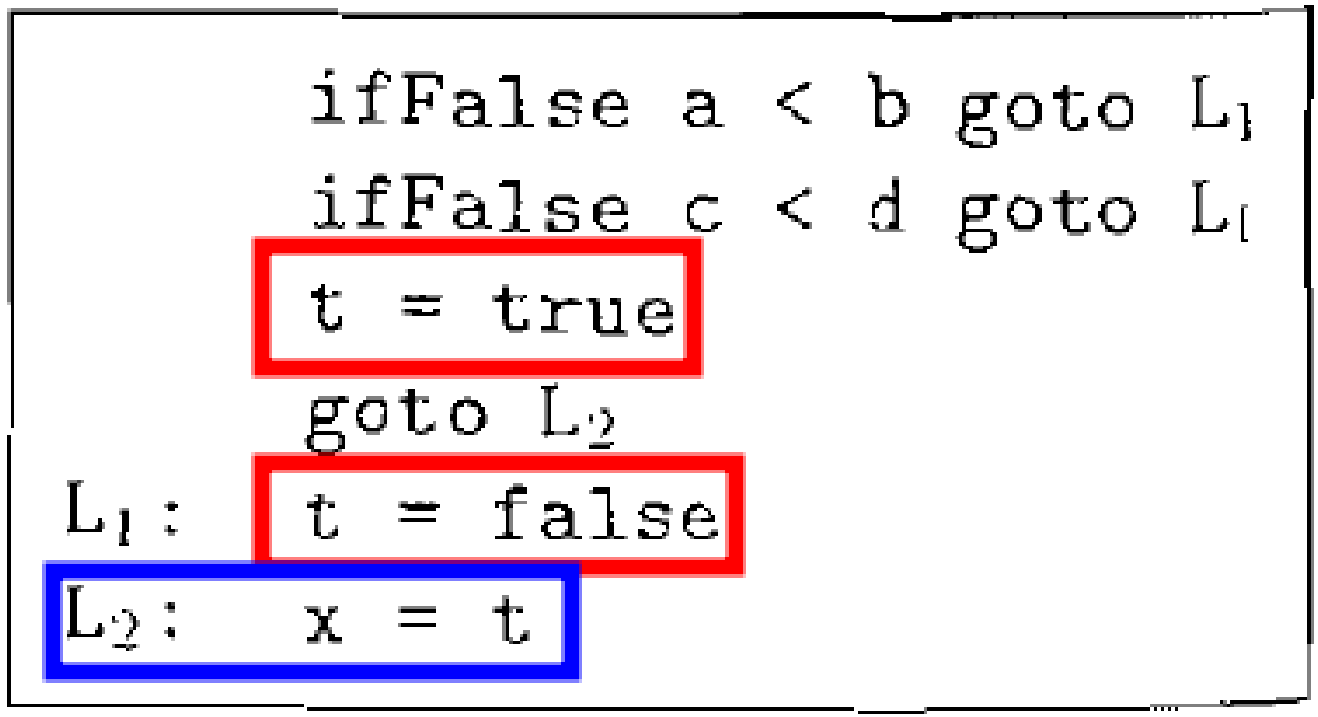
函数 $rvalue()$: 生成计算布尔值的代码, 并将结果存储在临时变量中

产生式	语义规则
$S \rightarrow \mathbf{id} = E\ ;$	$S.code = E.code \mid\mid gen(top.get(\mathbf{id}.lexeme)\ '='\ E.addr)$
$E \rightarrow E_1 + E_2$	$E.addr = \mathbf{new}\ Temp()$ 临时变量：虚拟寄存器 $E.code = E_1.code \mid\mid E_2.code \mid\mid gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$
$\mid\ -E_1$	$E.addr = \mathbf{new}\ Temp()$ $E.code = E_1.code \mid\mid gen(E.addr\ '='\ '\mathbf{minus}'\ E_1.addr)$
$\mid\ (\ E_1\ )$	生成代码 $E.addr = E_1.addr$ $E.code = E_1.code$
$\mid\ \mathbf{id}$	$E.addr = top.get(\mathbf{id}.lexeme)$ 符号表条目 $E.code = ''$

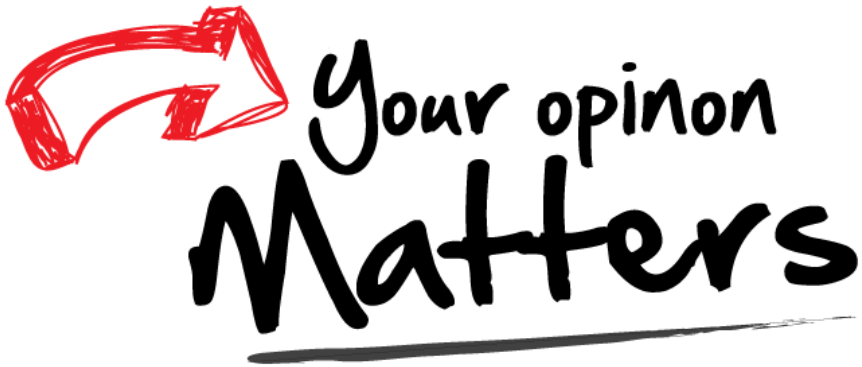
$$E \rightarrow E_1 \&\& E_2$$

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

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


Thank
You!



Office 926

hfwei@nju.edu.cn