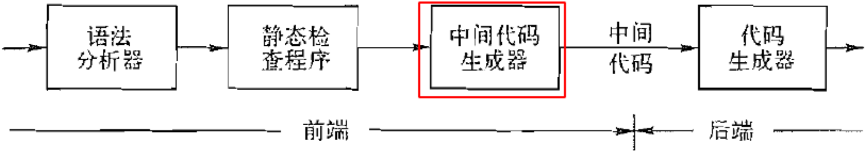
# 中间代码生成

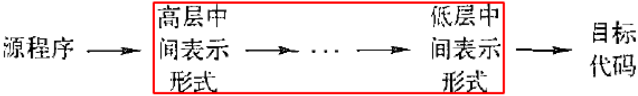
魏恒峰

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)

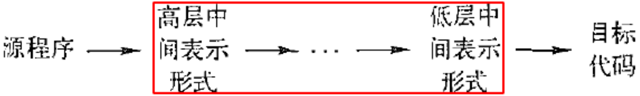
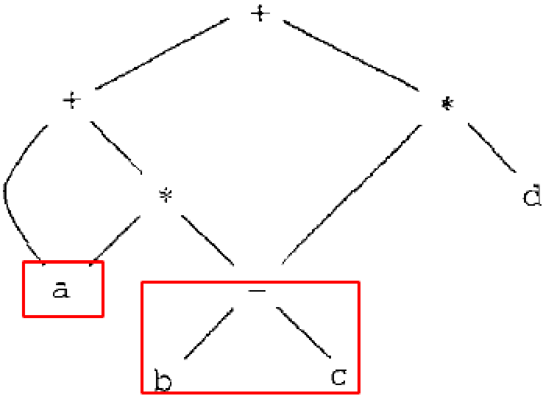



图 (抽象语法树)、**三地址代码**、C 语言

## 表达式的有向无环图

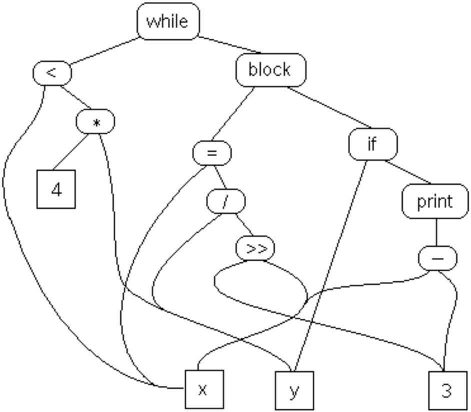



$$a + a * (b - c) + (b - c) * d$$

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

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 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 |
|---|---|
| $S \rightarrow \textbf{id} = E\ ;$ | $S.code = E.code \parallel gen(\boxed{top.get(\textbf{id}.lexeme)}\ '='\ E.addr)$ |
| $E \rightarrow E_1 + E_2$ | $E.addr = \textbf{new}\ Temp()$<br>$E.code = E_1.code \parallel E_2.code \parallel gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$ |
| $\mid\ -E_1$ | $E.addr = \textbf{new}\ Temp()$<br>$E.code = E_1.code \parallel gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr)$ |
| $\mid\ (\ E_1\ )$ | $E.addr = E_1.addr$<br>$E.code = E_1.code$ |
| $\mid\ \textbf{id}$ | $E.addr = \boxed{top.get(\textbf{id}.lexeme)}$ 符号表条目<br>$E.code = ''$ |

综合属性 $E.code$ 与 $E.addr$

| 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> $gen(E.addr\ '='\ E_1.addr\ '+'\ E_2.addr)$ |
| $\mid\ - E_1$ | $E.addr = \textbf{new}\ Temp()$ <br> $E.code = E_1.code \ \Vert$ <br> $gen(E.addr\ '='\ '\textbf{minus}'\ E_1.addr)$ |
| $\mid\ (\ E_1\ )$ | $E.addr = E_1.addr$ <br> $E.code = E_1.code$ |
| $\mid\ \textbf{id}$ | $E.addr = \boxed{top.get(\textbf{id}.lexeme)}$ 符号表条目 <br> $E.code = ''$ |

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

$$a = b + -c$$

# 表达式的中间代码翻译 (增量式)

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

**综合属性** $E.addr$

# 数组引用的中间代码翻译

int $a[2][3]$

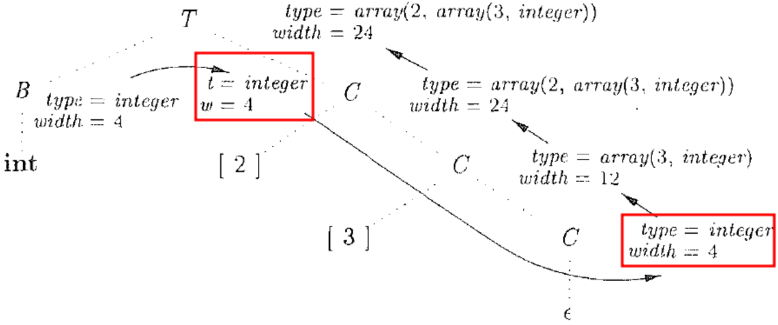



图 6-16　数组类型的语法制导翻译

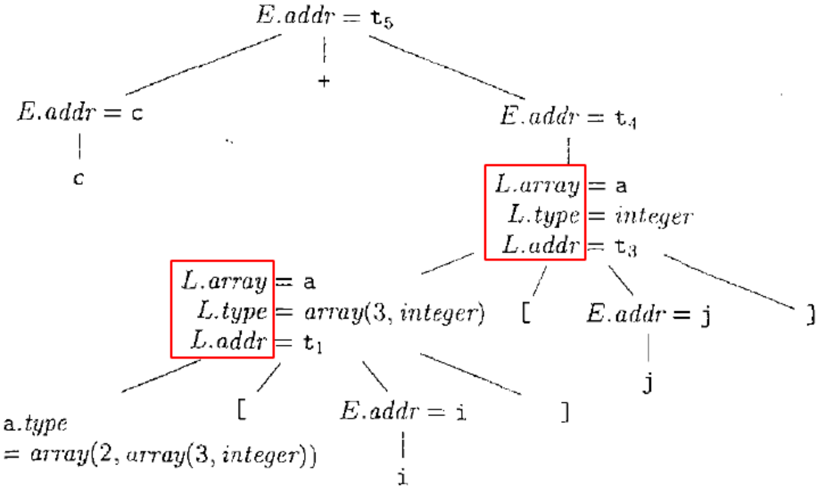
**数组类型声明**

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

int $a[2][3]$

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

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

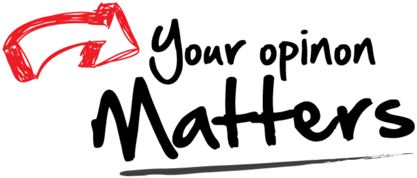

E.addr = t5
+
E.addr = c
c
E.addr = t4
L.array = a
L.type = integer
L.addr = t3
L.array = a
L.type = array(3, integer)
L.addr = t1
[
E.addr = j
]
j
a.type
= array(2, array(3, integer))
[
E.addr = i
]
i

int $a[2][3]$

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

int $a[2][3]$


Thank You!




Office 926

hfwei@nju.edu.cn