第六章 中间代码生成

简单赋值语句的翻译

哈尔滨工业大学 陈鄞

赋值语句翻译的任务

- 赋值语句的基本文法
  - ① $S \rightarrow \mathrm{id} = E;$
  - ② $E \rightarrow E_1 + E_2$
  - ③ $E \rightarrow E_1 * E_2$
  - ④ $E \rightarrow -E_1$
  - ⑤ $E \rightarrow (E_1)$
  - ⑥ $E \rightarrow \mathrm{id}$

- 赋值语句翻译的主要任务
  - 生成对表达式求值的三地址码

- 例
  - 源程序片段
    - $x = (a + b) * c ;$
  - 三地址码
    - $t_1 = a + b$
    - $t_2 = t_1 * c$
    - $x = t_2$

赋值语句的*SDT*

lookup(name)：查询符号表返回name 对应的记录

gen(code)：生成三地址指令code

newtemp( )：生成一个新的临时变量t，返回t的地址

$S \rightarrow \mathrm{id} = E;$ { p = lookup(id.lexeme); if p == nil then error ;
S.code = E.code ||
gen(p '=' E.addr); }

$E \rightarrow E_1 + E_2$ { E.addr = newtemp( );
E.code = E_1.code || E_2.code||
gen(E.addr '=' E_1.addr '+' E_2.addr); }

$E \rightarrow E_1 * E_2$ { E.addr = newtemp( );
E.code = E_1.code || E_2.code ||
gen(E.addr '=' E_1.addr '*' E_2.addr); }

$E \rightarrow -E_1$ { E.addr = newtemp( );
E.code = E_1.code||
gen(E.addr '=' 'uminus' E_1.addr); }

$E \rightarrow (E_1)$ { E.addr = E_1.addr;
E.code= E_1.code; }

$E \rightarrow \mathrm{id}$ { E.addr = lookup(id.lexeme); if E.addr == nil then error ;
E.code = ''; }

符号	综合属性
S	*code*
E	*code* *addr*

增量翻译 (*Incremental Translation*)

$S \to$ id $= E$; { *p = lookup*(id.*lexeme*); *if p == nil then error* ;
S.code = E.code ||
gen(p '=' *E.addr*); }

$E \to E_1 + E_2$ { *E.addr = newtemp*( );
E.code = E_1*.code* || E_2*.code*||
gen(E.addr '=' E_1*.addr* '+' E_2*.addr*); }

$E \to E_1 * E_2$ { *E.addr = newtemp*( );
E.code = E_1*.code* || E_2*.code* ||
gen(E.addr '=' E_1*.addr* '*' E_2*.addr*); }

$E \to -E_1$ { *E.addr = newtemp*( );
E.code = E_1*.code*||
gen(E.addr '=' '*uminus*' E_1*.addr*); }

$E \to (E_1)$ { *E.addr* = E_1*.addr*;
E.code= E_1*.code*; }

$E \to$ id { *E.addr = lookup*(id.*lexeme*); *if E.addr == nil then error* ;
E.code = ''; }

在增量方法中，*gen*( )不仅要构造出一个新的三地址指令，还要将它添加到至今为止已生成的指令序列之后

例

①$S \rightarrow$ id = E; { p = *lookup*(id.*lexeme*);
if p==nil then error ;
gen(p '=' E.*addr*); }

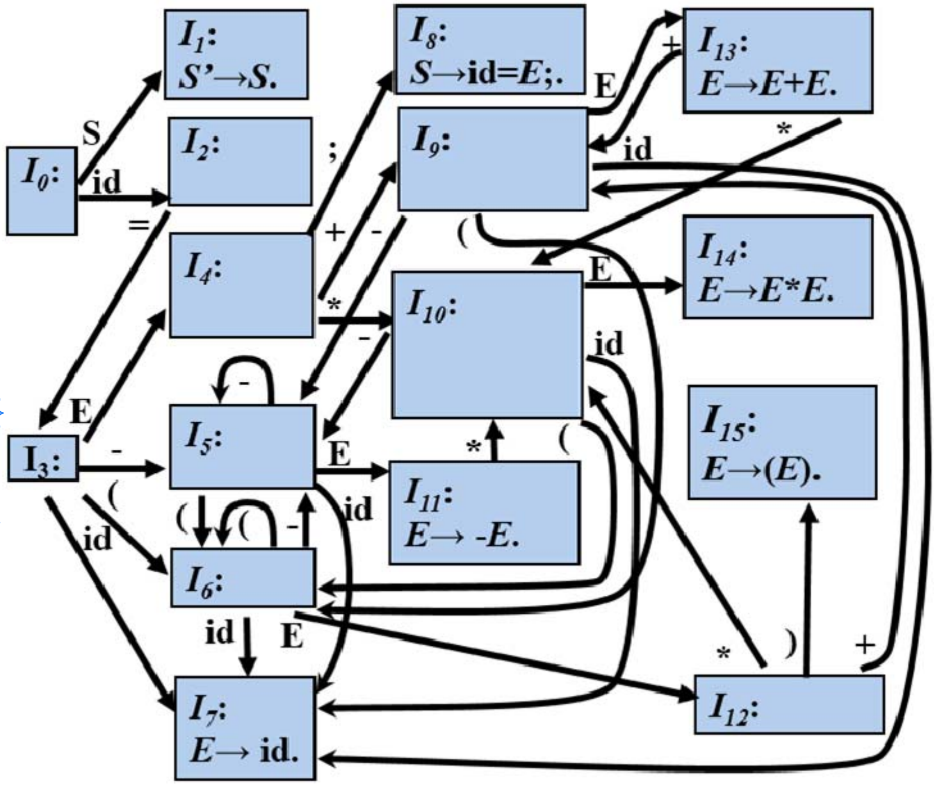
②$E \rightarrow E_1 + E_2${ E.*addr* = *newtemp*( );
gen(E.*addr* '=' E_1.*addr* '+' E_2.*addr*); }

③$E \rightarrow E_1 * E_2${ E.*addr* = *newtemp*( );
gen(E.*addr* '=' E_1.*addr* '*' E_2.*addr*); }

④$E \rightarrow -E_1${ E.*addr* = *newtemp*( );
gen(E.*addr* '=' 'uminus' E_1.*addr*); }

⑤$E \rightarrow (E_1)${ E.*addr* = E_1.*addr* ; }

⑥$E \rightarrow$ id { E.*addr* = *lookup*(id.*lexeme*);
if E.addr==nil then error ; }



例： $x = (a + b) * c$;

0	2	3	6	7
$	id	=	(	id
	x			a

例

①$S \rightarrow$ id = E; { *p = lookup(*id*.lexeme);*
if p==nil then error ;
gen(p '=' E.addr); }

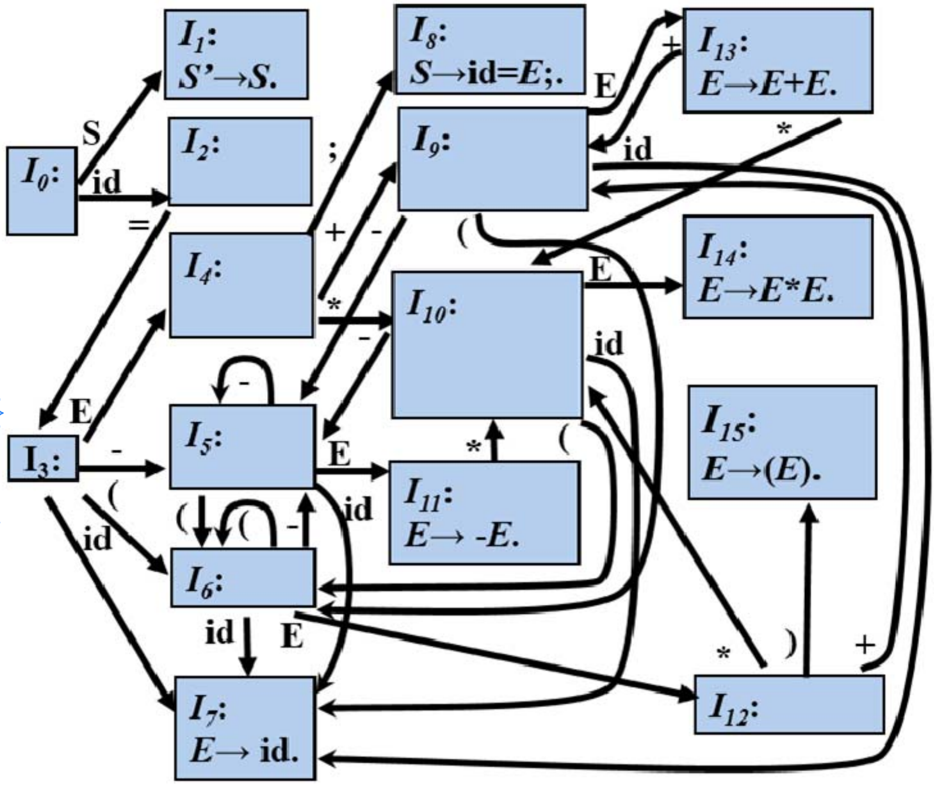
②$E \rightarrow E_1 + E_2${ *E.addr = newtemp( );*
gen(E.addr '=' E_1.addr '+' E_2.addr); }

③$E \rightarrow E_1 * E_2${ *E.addr = newtemp( );*
gen(E.addr '=' E_1.addr '' E_2.addr); }*

④$E \rightarrow -E_1${ *E.addr = newtemp( );*
gen(E.addr '=' 'uminus' E_1.addr); }

⑤$E \rightarrow (E_1)${ *E.addr = E_1.addr ; }*

⑥$E \rightarrow$ id { *E.addr = lookup(*id*.lexeme);*
if E.addr==nil then error ; }



I_0:
I_1: $S' \rightarrow S.$
I_2:
I_3:
I_4:
I_5:
I_6:
I_7: $E \rightarrow$ id.
I_8: $S \rightarrow$ id=E;.
I_9:
I_{10}:
I_{11}: $E \rightarrow -E.$
I_{12}:
I_{13}: $E \rightarrow E+E.$
I_{14}: $E \rightarrow E*E.$
I_{15}: $E \rightarrow (E).$
S
id
=
;
E
+
-
*
(
)

例： $x = (a + b) * c$;

0	2	3	6	12	9	7
\$	id	=	(	E	+	id
	x			a		b

例

①$S \rightarrow \mathbf{id} = E;$ { $p = lookup(\mathbf{id}.lexeme)$;
if p==nil then error ;
gen(p '=' E.addr); }

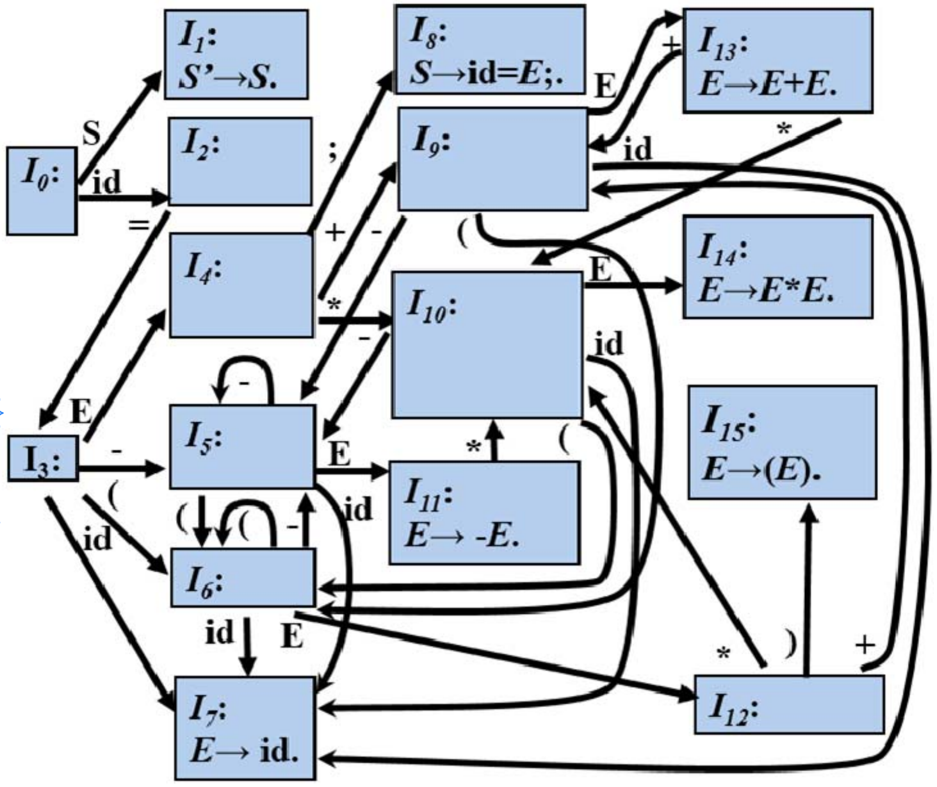
②$E \rightarrow E_1 + E_2$ { $E.addr = newtemp(\,)$;
gen($E.addr$ '=' $E_1.addr$ '+' $E_2.addr$); }

③$E \rightarrow E_1 * E_2$ { $E.addr = newtemp(\,)$;
gen($E.addr$ '=' $E_1.addr$ '*' $E_2.addr$); }

④$E \rightarrow -E_1$ { $E.addr = newtemp(\,)$;
gen($E.addr$ '=' 'uminus' $E_1.addr$); }

⑤$E \rightarrow (E_1)$ { $E.addr = E_1.addr$; }

⑥$E \rightarrow \mathbf{id}$ { $E.addr = lookup(\mathbf{id}.lexeme)$;
if E.addr==nil then error ; }



I_0:
I_1: $S' \rightarrow S.$
I_2:
I_3:
I_4:
I_5:
I_6:
I_7: $E \rightarrow \mathbf{id}.$
I_8: $S \rightarrow \mathbf{id}=E;.$
I_9:
I_{10}:
I_{11}: $E \rightarrow -E.$
I_{12}:
I_{13}: $E \rightarrow E+E.$
I_{14}: $E \rightarrow E*E.$
I_{15}: $E \rightarrow (E).$
S
id
=
;
E
+
*
-
(
)

例：$x = (a + b) * c ;$

0	2	3	6	12	9	13
\$	id	=	(	E	+	E
	x			a		b

$t_1 = a + b$

例

①$S \rightarrow \mathbf{id} = E;$ { $p = lookup(\mathbf{id}.lexeme);$
if p==nil then error ;
gen(p '=' E.addr); }

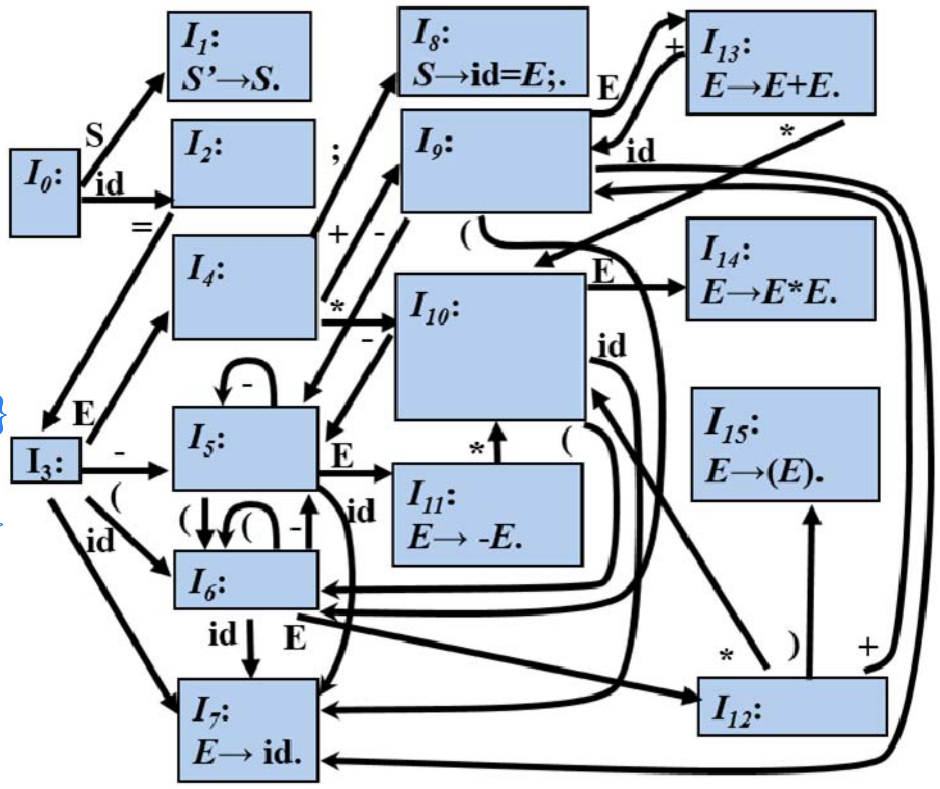
②$E \rightarrow E_1 + E_2$ { $E.addr = newtemp(\,);$
gen(E.addr '=' E_1.addr '+' E_2.addr); }

③$E \rightarrow E_1 * E_2$ { $E.addr = newtemp(\,);$
gen(E.addr '=' E_1.addr '*' E_2.addr); }

④$E \rightarrow -E_1$ { $E.addr = newtemp(\,);$
gen(E.addr '=' 'uminus' E_1.addr); }

⑤$E \rightarrow (E_1)$ { $E.addr = E_1.addr$; }

⑥$E \rightarrow \mathbf{id}$ { $E.addr = lookup(\mathbf{id}.lexeme);$
if E.addr==nil then error ; }



$t_1 = a + b$

例： $x = (a + b) * c \ ;$

0	2	3	6	12	15
\$	id	=	(	E	)
	x			t_1	

例

①$S \to \mathbf{id} = E;$ { $p = lookup(\mathrm{id}.lexeme);$
if p==nil then error ;
gen(p '=' E.addr); }

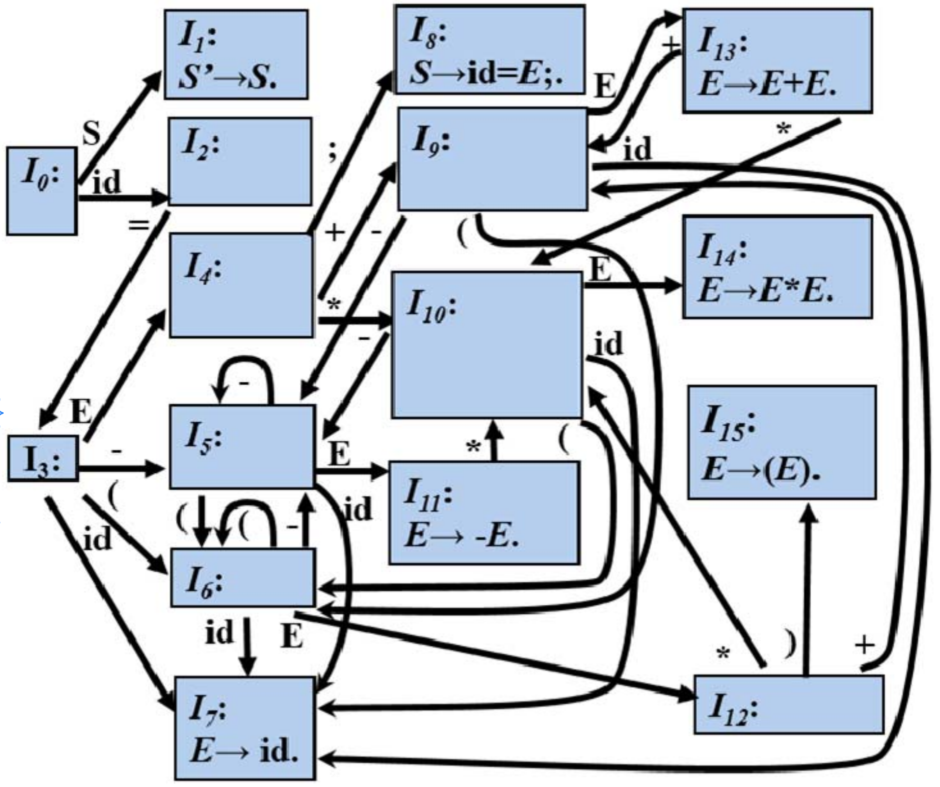
②$E \to E_1 + E_2$ { $E.addr = newtemp(\ );$
gen(E.addr '=' E_1.addr '+' E_2.addr); }

③$E \to E_1 * E_2$ { $E.addr = newtemp(\ );$
gen(E.addr '=' E_1.addr '*' E_2.addr); }

④$E \to -E_1$ { $E.addr = newtemp(\ );$
gen(E.addr '=' 'uminus' E_1.addr); }

⑤$E \to (E_1)$ { $E.addr = E_1.addr$; }

⑥$E \to \mathbf{id}$ { $E.addr = lookup(\mathrm{id}.lexeme);$
if E.addr==nil then error ; }



例： $x = (a + b) * c ;$

0	2	3	4	10	7
\$	id	=	E	*	id
	x		t_1		c

$t_1 = a + b$

例

①$S \rightarrow \text{id} = E;$ { $p = lookup(\text{id}.lexeme)$;
  if p==nil then error ;
  gen(p '=' E.addr); }

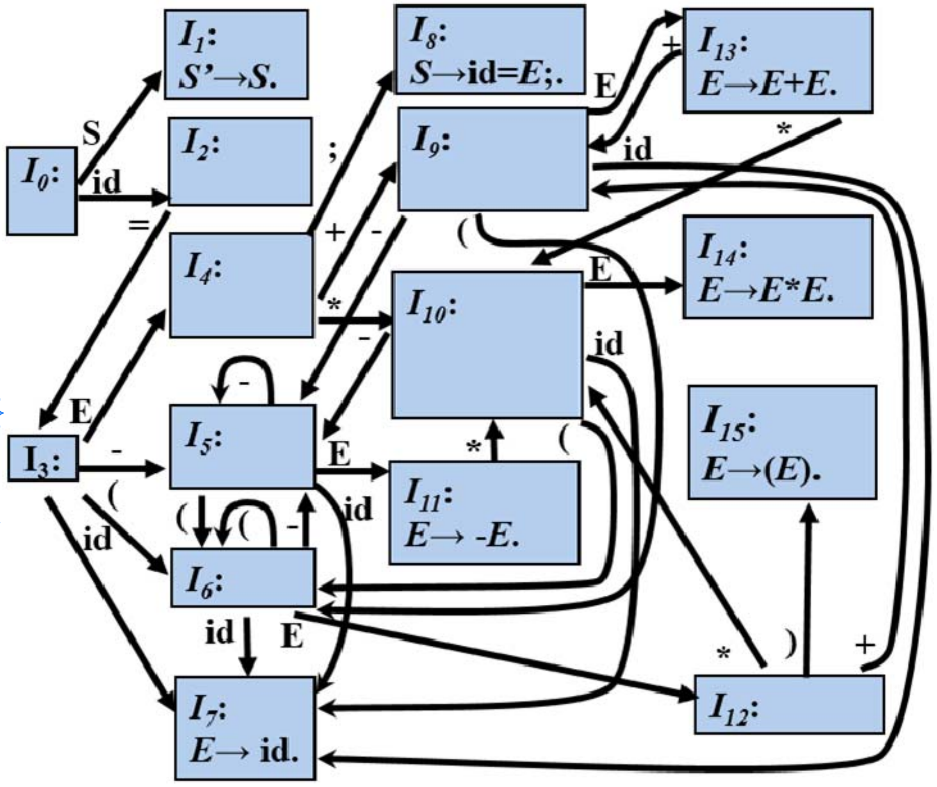
②$E \rightarrow E_1 + E_2$ { E.addr = newtemp( );
  gen(E.addr '=' E_1.addr '+' E_2.addr); }

③$E \rightarrow E_1 * E_2$ { E.addr = newtemp( );
  gen(E.addr '=' E_1.addr '*' E_2.addr); }

④$E \rightarrow -E_1$ { E.addr = newtemp( );
  gen(E.addr '=' 'uminus' E_1.addr); }

⑤$E \rightarrow (E_1)$ { E.addr = E_1.addr ; }

⑥$E \rightarrow \text{id}$ { E.addr = lookup(id.lexeme);
  if E.addr==nil then error ; }



例： $x = (a + b) * c ;$

0	2	3	4	10	14
\$	id	=	E	*	E
	x		t_1		c

$t_1 = a + b$

$t_2 = t_1 * c$

例

①$S \rightarrow$ id = E; { p = *lookup*(id.*lexeme*);
if p==nil then error ;
gen(p '=' E.*addr*); }

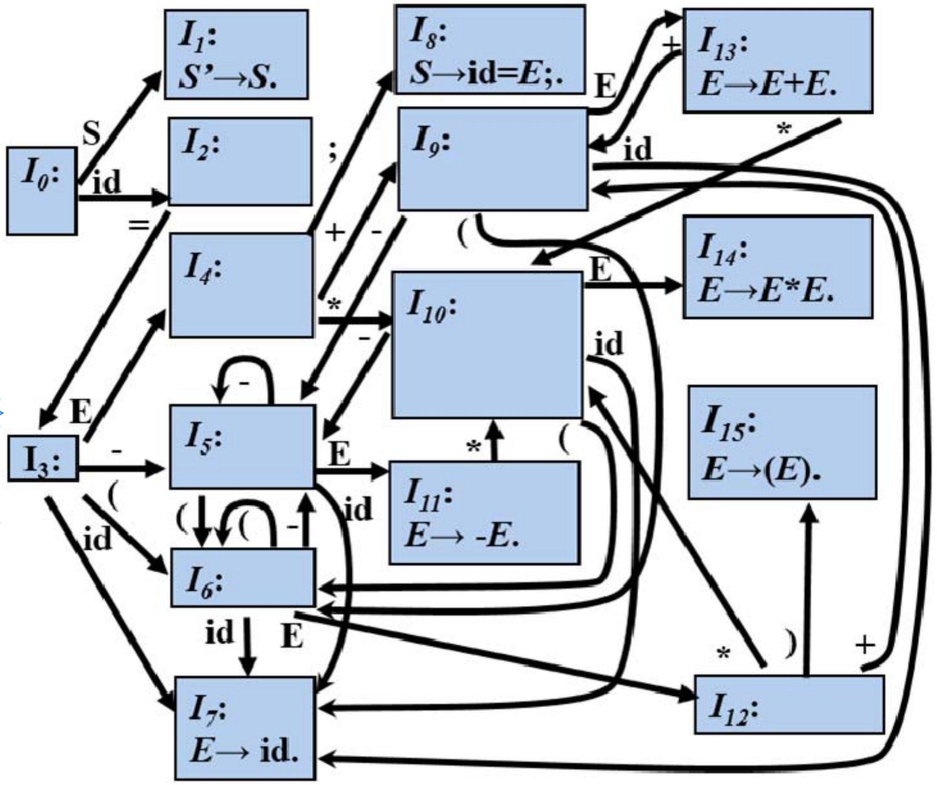
②$E \rightarrow E_1 + E_2$ { E.*addr* = *newtemp*( );
gen(E.*addr* '=' E_1.*addr* '+' E_2.*addr*); }

③$E \rightarrow E_1 * E_2$ { E.*addr* = *newtemp*( );
gen(E.*addr* '=' E_1.*addr* '*' E_2.*addr*); }

④$E \rightarrow -E_1$ { E.*addr* = *newtemp*( );
gen(E.*addr* '=' 'uminus' E_1.*addr*); }

⑤$E \rightarrow (E_1)$ { E.*addr* = E_1.*addr* ; }

⑥$E \rightarrow$ id { E.*addr* = *lookup*(id.*lexeme*);
if E.addr==nil then error ; }



I_0:
I_1: $S' \rightarrow S.$
I_2:
I_3:
I_4:
I_5:
I_6:
I_7: $E \rightarrow$ id.
I_8: $S \rightarrow$ id=E;.
I_9:
I_{10}:
I_{11}: $E \rightarrow -E.$
I_{12}:
I_{13}: $E \rightarrow E+E.$
I_{14}: $E \rightarrow E*E.$
I_{15}: $E \rightarrow (E).$
S
id
=
;
E
+
-
*
(
)

例： $x = (a + b) * c$;

0	2	3	4	8
\$	id	=	E	;
	x		t_2	

$t_1 = a + b$

$t_2 = t_1 * c$

$x = t_2$

例

①$S \rightarrow$ id = E; { p = *lookup*(id.*lexeme*);
if p==nil then error ;
gen(p '=' E.*addr*); }

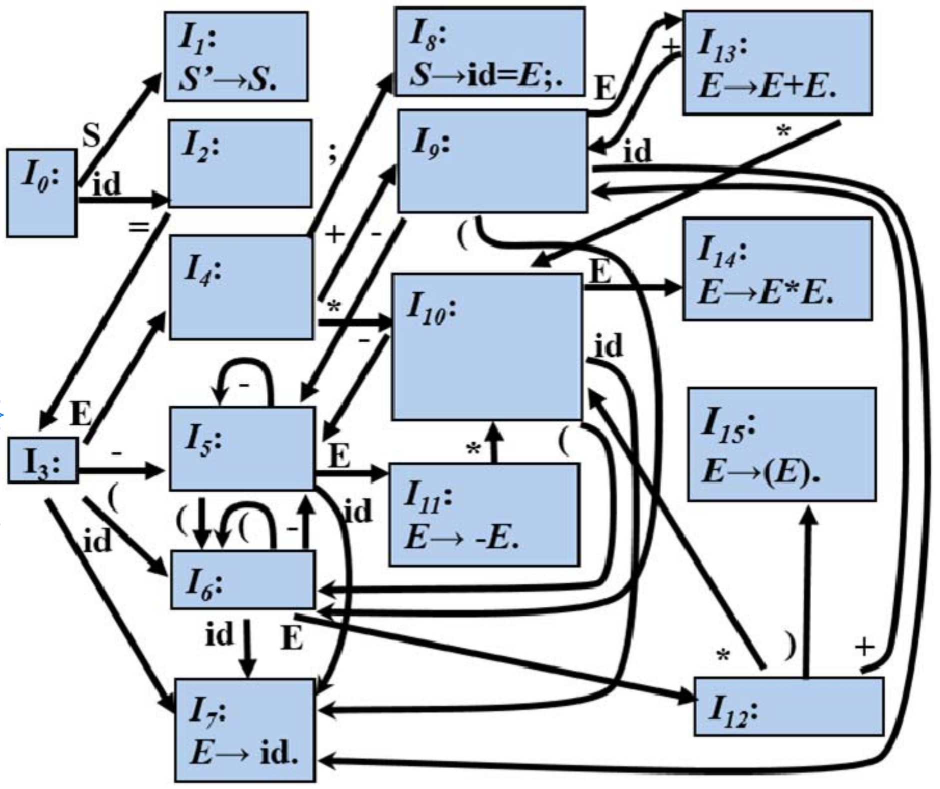
②$E \rightarrow E_1 + E_2$ { E.*addr* = *newtemp*( );
gen(E.*addr* '=' E_1.*addr* '+' E_2.*addr*); }

③$E \rightarrow E_1 * E_2$ { E.*addr* = *newtemp*( );
gen(E.*addr* '=' E_1.*addr* '*' E_2.*addr*); }

④$E \rightarrow -E_1$ { E.*addr* = *newtemp*( );
gen(E.*addr* '=' 'uminus' E_1.*addr*); }

⑤$E \rightarrow (E_1)$ { E.*addr* = E_1.*addr* ; }

⑥$E \rightarrow$ id { E.*addr* = *lookup*(id.*lexeme*);
if E.addr==nil then error ; }



I_0:
I_1: $S' \rightarrow S.$
I_2:
I_3:
I_4:
I_5:
I_6:
I_7: $E \rightarrow$ id.
I_8: $S \rightarrow$ id=E;.
I_9:
I_{10}:
I_{11}: $E \rightarrow -E.$
I_{12}:
I_{13}: $E \rightarrow E+E.$
I_{14}: $E \rightarrow E*E.$
I_{15}: $E \rightarrow (E).$
S
id
=
;
E
+
*
-
(
)

例： $x = (a + b) * c$;

0	1
\$	S

$t_1 = a + b$

$t_2 = t_1 * c$

$x = t_2$

第六章 中间代码生成

简单赋值语句的翻译

哈尔滨工业大学 陈鄞

第六章 中间代码生成

数组引用的翻译

哈尔滨工业大学 陈鄞

数组引用的翻译

- 赋值语句的基本文法

$S \rightarrow \mathbf{id} = E; \mid L = E;$

$E \rightarrow E_1 + E_2 \mid -E_1 \mid (E_1) \mid \mathbf{id} \mid L$

$L \rightarrow \mathbf{id}\ [E] \mid L_1\ [E]$

将数组引用翻译成三地址码时要解决的主要问题是确定数组元素的存放地址，也就是数组元素的寻址

数组元素寻址 (*Addressing Array Elements*)

- 一维数组
  - 假设每个数组元素的宽度是w，则数组元素$a[i]$的相对地址是：

$$base+i\times w$$

其中，$base$是数组的基地址，$i\times w$是偏移地址

- 二维数组
  - 假设一行的宽度是w_1，同一行中每个数组元素的宽度是w_2，则数组元素$a[i_1]\,[i_2]$的相对地址是：

$$base+\underbrace{i_1\times w_1+i_2\times w_2}_{\text{偏移地址}}$$

- k维数组
  - 数组元素$a[i_1]\,[i_2]\ldots[i_k]$的相对地址是：

$$base+\underbrace{i_1\times w_1+i_2\times w_2+\ldots+i_k\times w_k}_{\text{偏移地址}}$$

$w_1 \rightarrow a[i_1]$ 的宽度
$w_2 \rightarrow a[i_1]\,[i_2]$ 的宽度
$\ldots$
$w_k \rightarrow a[i_1]\,[i_2]\ldots[i_k]$的宽度

例

➢ 假设$type(a) = array(3, array(5, array(8, int)))$，
一个整型变量占用4个字节，
则$addr(a[i_1][i_2][i_3]) = base + i_1 * w_1 + i_2 * w_2 + i_3 * w_3$

$$= base + i_1 * \underbrace{160}_{a[i_1]\text{的宽度}} + i_2 * \underbrace{32}_{a[i_1][i_2]\text{的宽度}} + i_3 * \underbrace{4}_{a[i_1][i_2][i_3]\text{的宽度}}$$

带有数组引用的赋值语句的翻译

- 例1

  假设 $type(a)=array(n, int)$,

  - 源程序片段
    - $c = a[i]$;

  $addr(a[i]) = base + i*4$

  - 三地址码
    - $t_1 = i * 4$
    - $t_2 = a [t_1]$
    - $c = t_2$

带有数组引用的赋值语句的翻译

- 例2

  假设 $type(a) = array(3, array(5, int))$，

  - 源程序片段
    - $c = a[i_1][i_2];$ $\boxed{addr(a[i_1][i_2]) = base + \underbrace{i_1 * 20}_{t_1} + \underbrace{i_2 * 4}_{t_2}}$
  - 三地址码
    - $t_1 = i_1 * 20$
    - $t_2 = i_2 * 4$
    - $t_3 = t_1 + t_2$
    - $t_4 = a[t_3]$
    - $c = t_4$

数组引用的*SDT*

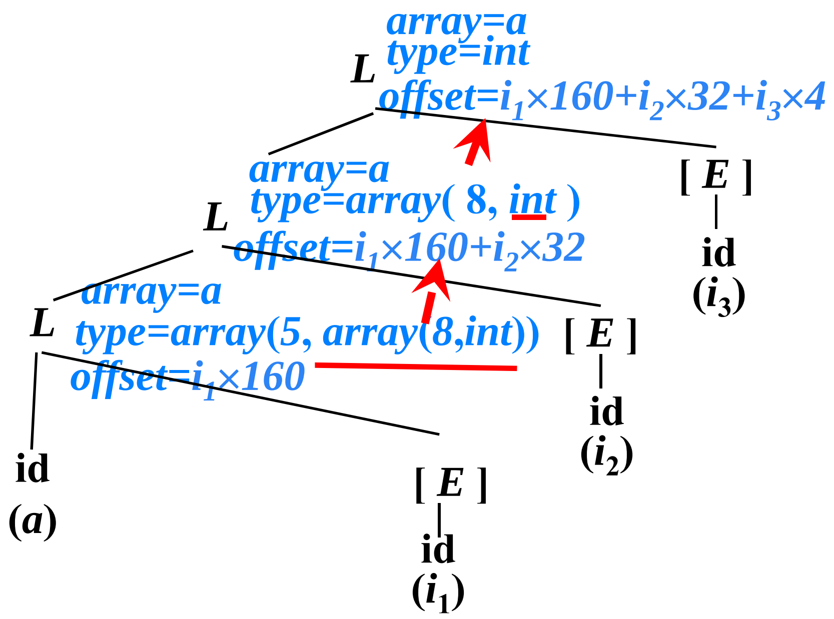
- 赋值语句的基本文法

$S \rightarrow \mathrm{id} = E; \mid L = E;$

$E \rightarrow E_1 + E_2 \mid -E_1 \mid (E_1) \mid \mathrm{id} \mid L$

$L \rightarrow \mathrm{id}\ [E] \mid L_1\ [E]$

$base + \underline{i_1 \times w_1} + \underline{i_2 \times w_2} + \ldots + i_k \times w_k$



假设 *type*(*a*)= *array*(3, *array*(5, *array*(8, *int*)))，翻译语句片段“*a*[*i*₁][*i*₂][*i*₃]”

- *L*的综合属性
  - *L.type*：*L*生成的数组元素的类型
  - *L.offset*：指示一个临时变量，该临时变量用于累加公式中的$i_j \times w_j$项，从而计算数组引用的偏移量
  - *L.array*：数组名在符号表的入口地址

数组引用的*SDT*

假设 *type*(*a*)= *array*(3, *array*(5, *array*(8, *int*)))，翻译语句片段 “*a*[i_1][i_2][i_3]”

$$addr(a[i_1][i_2][i_3]) = base + \underbrace{i_1 \times w_1}_{t_1} + \underbrace{i_2 \times w_2}_{t_2} + \underbrace{i_3 \times w_3}_{t_4}$$

$S \rightarrow$ id = E;

  | L = E; { *gen*(*L.array* '[' *L.offset* ']' '=' *E.addr*); }

$E \rightarrow E_1 + E_2$ | $-E_1$ | (E_1) | id

  | L { *E.addr* = *newtemp*(); *gen*(*E.addr* '=' *L. array* '[' *L.offset* ']'); }

$L \rightarrow$ id [E]{ *L.array* = *lookup*(id.*lexeme*); *if L.array*==*nil then error* ;

L.type = *L.array.type.elem* ;

L.offset = *newtemp*();

gen(*L.offset* '=' *E.addr* '*' *L.type.width*); }

  | L_1[E]{ *L.array* = L_1. *array*;

L.type = L_1.*type.elem* ;

t = *newtemp*();

gen(*t* '=' *E.addr* '*' *L.type.width*);

L.offset = *newtemp*();

gen(*L.offset* '=' L_1.*offset* '+' *t*); }

三地址码

$t_1 = i_1$*160

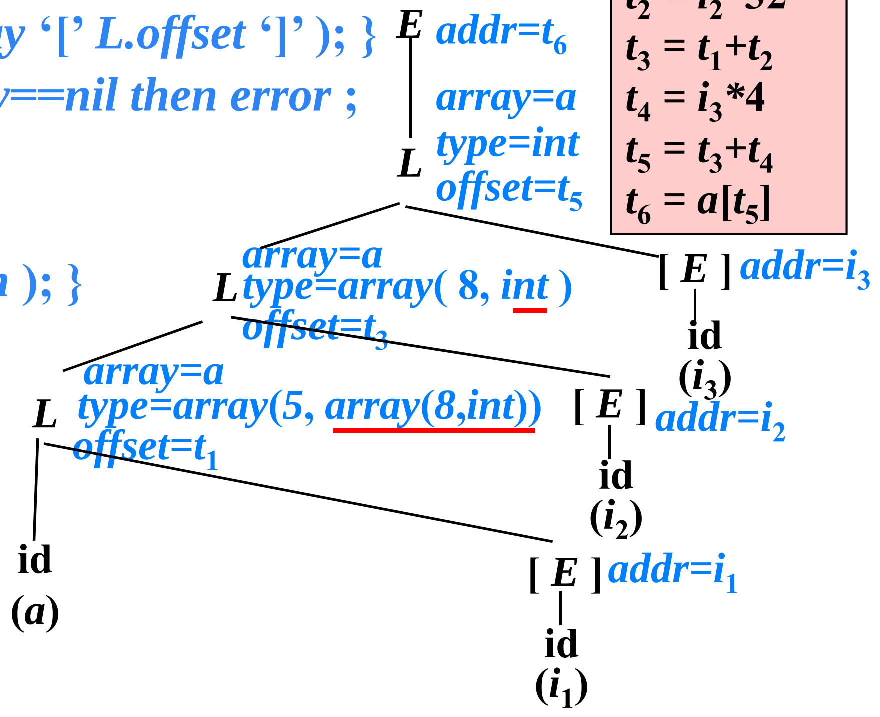
$t_2 = i_2$*32

$t_3 = t_1 + t_2$

$t_4 = i_3$*4

$t_5 = t_3 + t_4$

$t_6 = a[t_5]$



E addr=t6
L array=a type=int offset=t5
L array=a type=array(8, int) offset=t3
[E] addr=i3
id (i3)
L array=a type=array(5, array(8,int)) offset=t1
[E] addr=i2
id (i2)
id (a)
[E] addr=i1
id (i1)

第六章 中间代码生成

数组引用的翻译

哈尔滨工业大学 陈鄞