

# Síntese

## Compiladores

### Código intermediário

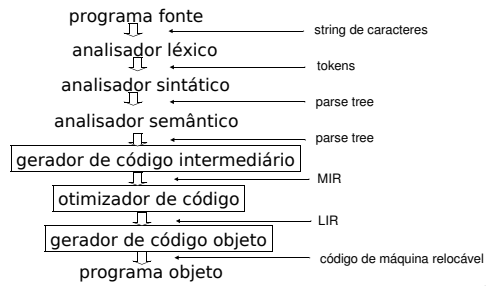
Os assuntos apresentados trazem respostas às perguntas seguintes:

1. Pode a memória ser alocada dinamicamente ?
2. Pode a memória ser liberada explicitamente ?
  - Heap vs. Pilha
3. Procedimentos podem ser recursivos ?
  - Pilha de ativação
4. O que acontece com valores de nomes quando o controle retorna de uma ativação ?
  - Ponteiros de acesso
5. Pode um procedimento referenciar nomes não locais ?
  - Ponteiros de acesso
6. Como os parâmetros são passados quando procedimentos são chamados ?
  - Passagem de parâmetros
7. Podem procedimentos ser chamados como parâmetros ?
8. Podem procedimentos ser retornados como resultados ?

## Geração de código intermediário

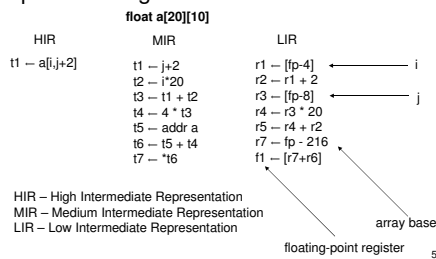
- Vantagens transformação em mais de um passo:
  - Possibilita a otimização do código intermediário para prover um código otimizado mais eficiente
  - Simplifica a implementação do compilador, resolvendo gradativamente as dificuldades de passagem de código fonte para código intermediário
  - Possibilita a tradução de código intermediário para diversas máquinas

## Modelo Clássico



## Linguagens Intermediárias

### Tipos de código intermediário



## Linguagens Intermediárias

- HIR:
  - Usada nos primeiros estágios do compilador
  - Simplificação de construções gramaticais para somente o essencial para otimização/geração de código
- MIR:
  - Boa base para geração de código eficiente
  - Pode expressar todas características de linguagens de programação de forma independente da linguagem
  - Representação de variáveis, temporários, registradores
- LIR:
  - Quase 1-1 para linguagem de máquina
  - Dependente da arquitetura

## Linguagens Intermediárias

- Tipos de código intermediário

- HIR e MIR:

- Árvore e grafo de sintaxe
  - Notações Pós-fixada e Pré-fixada:
  - Representações linearizadas

- MIR:

- Código de três endereços:
  - quádruplas
  - triplas

- LIR:

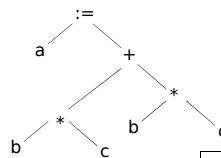
- Instruções assembler

7

## HIR: Árvore e grafo de sintaxe

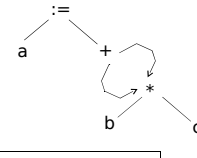
- Árvore de sintaxe

- Forma condensada da árvore de derivação
- As cadeia de produções simples ( $A \rightarrow B, B \rightarrow C$ ) não aparecem



- Grafo de sintaxe

- Inclui simplificações da árvore de sintaxe
- Faz a fatoração de sub-expressões comuns



**a := (b\*c) + (b\*c)**

8

## HIR: Notações pós e pré-fixadas

Infixada	Pós-fixada	Pré-fixada
$(a+b)*c$	$ab+c^*$	$*+abc$
$a*(b+c)$	$abc+^*$	$*a+bc$
$a+b*c$	$abc^*+$	$+a*bc$

Geração de código fácil com uma pilha!

9

## Código de três endereços (TAC)

- Cada instrução referencia, no máximo, três variáveis (endereços de memória)
  - Necessita variáveis temporárias!
- As instruções possíveis são as seguintes

$A := B \text{ op } C$   
 $A := \text{op } B$   
 $A := B$   
 goto L  
 If A op\_rel B goto L

10

## Representação de TAC para $A := B*(-C+D)$

### Quádruplas

	oper	arg1	arg2	result
(0)	-u	C		T1
(1)	+	T1	D	T2
(2)	*	B	T2	T3
(3)	:=	T3		A

### Triplas

	oper	arg1	arg2
(0)	-u	C	
(1)	+	(0)	D
(2)	*	B	(1)
(3)	:=	A	(2)

11

## TAC

- Geração de código de três endereços:
  - Expressões
  - Declarações (escopo simples)
  - Declarações (escopos aninhados)
  - Comandos de atribuição
  - Arrays e Registros
  - Expressões booleanas
  - Comandos de decisão
  - Comandos de iteração

12

## Esquema de tradução código de 3-endereços

$S \rightarrow id := E$	Exemplo:
$E \rightarrow E_1 + E_2$	$A := X + Y * Z$
$E \rightarrow E_1 * E_2$	$T1 := Y * Z$
$E \rightarrow (E_1)$	$T2 := X + T1$
$E \rightarrow id$	$A := T2$

Usa-se dois mecanismos:  
 2. O atributo 'nome' armazena o nome da variável  
 3. A função "geracod" escreve (na tela) código

13

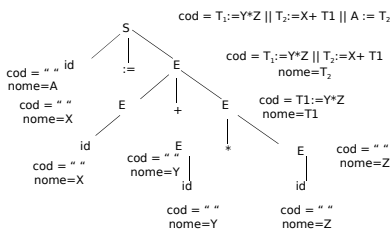
## Esquema de tradução de código de 3-endereços

$S \rightarrow id := E$	{S.cod = E.cod    geracod(id.nome := " E.nome)}
$E \rightarrow E_1 + E_2$	{E.nome = geratemp; E.cod = E <sub>1</sub> .cod    E <sub>2</sub> .cod    geracod (E.nome := " E <sub>1</sub> .nome "+" E <sub>2</sub> .nome)}
$E \rightarrow E_1 * E_2$	{E.nome = geratemp; E.cod = E <sub>1</sub> .cod    E <sub>2</sub> .cod    geracod (E.nome := " E <sub>1</sub> .nome "*" E <sub>2</sub> .nome)}
$E \rightarrow (E_1)$	{E.nome = E <sub>1</sub> .nome; E.cod = E <sub>1</sub> .cod }
$E \rightarrow id$	{E.nome = id.nome; E.cod = " }

14

## Árvore de derivação anotada

$A := X + Y * Z$



15

## (Detalhe) Reaproveitamento de temporários

- Deve-se poupar os temporários
- Usa-se um contador
  - Valor inicial: 1
  - Acrescentado a cada geração de novo temporário
  - Decrescentado a cada uso de um temporário como operando.
- Exemplo:  $X := A * B + C * D - E * F$ 
  - Necessita apenas 2 temporários.

16

## TAC

- Geração de código de três endereços:
  - Expressões
  - Declarações (escopo simples)
  - Declarações (escopos aninhados)
  - Comandos de atribuição
  - Arrays e Registros
  - Expressões booleanas
  - Comandos de decisão
  - Comandos de iteração

17

## Declarações

- Associar posições de memória para nomes locais de procedimentos
- Atualizar na tabela de símbolos:
  - endereço de memória relativo:
    - base da área estática de dados
    - dados locais no registro de ativação
- Geração de endereços pode ter máquina alvo em mente:
  - ex. inteiros consecutivos diferem de 4 bytes

18

## Declarações

- Em linguagens como C, Pascal e Fortran, variáveis de um mesmo procedimento pertencem a um mesmo grupo
  - Indexação base + deslocamento
    - Antes da primeira declaração zera o deslocamento
    - Para cada declaração processada:
      - criar entrada na tabela de símbolos com deslocamento igual ao deslocamento atual
      - incrementar o valor do deslocamento proporcional ao tamanho do objeto definido na declaração

19

## Declarações

- Usar tradução dirigida pela sintaxe:
  - atributos:
    - tipo
    - tamanho
    - deslocamento
- Rotina Suporte:
  - adSimb(nome, tipo, deslocamento):
    - cria uma nova entrada na tabela de símbolos para nome, confere o mesmo tipo e o endereço relativo deslocamento
  - array(num, tipo)
  - ponteiro(tipo)

20

## Exemplo

- Cálculo de tamanhos:
  - inteiro: 4 bytes
  - real: 8 bytes
  - array: número de elementos x tamanho do tipo
  - apontador: 4 bytes

21

## Exemplo

- $P \rightarrow \{\text{deslocamento} := 0\} D$

22

## Exemplo

- $P \rightarrow \{\text{deslocamento} := 0\} D$
- $P \rightarrow M D$
- $M \rightarrow \{\text{deslocamento} := 0\} \epsilon$

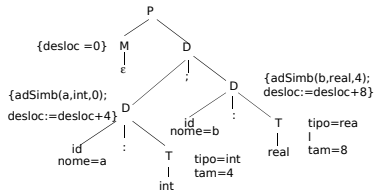
23

## Esquema de tradução tabela de símbolos bloco unitário

$P \rightarrow MD$	
$M \rightarrow \epsilon$	{desloc = 0}
$D \rightarrow D ; D$	
$D \rightarrow id : T$	{adSimb (id.nome, T.tipo, desloc); desloc = desloc + T.tam}
$T \rightarrow int$	{T.tipo = int; T.tam = 4}
$T \rightarrow real$	{T.tipo = real; T.tam = 8}
$T \rightarrow array [num] \text{ of } T_1$	{T.tipo = array(num.val, T <sub>1</sub> .tipo); T.tam = num.val * T <sub>1</sub> .tam}
$T \rightarrow ! T_1$	{T.tipo = ponteiro(T <sub>1</sub> .tipo); T.tam = 4}

24

## Árvore de derivação - a: int; b: real;



25

## TAC

- Geração de código de três endereços:
  - Expressões
  - Declarações (escopo simples)
  - Declarações (escopos aninhados)
  - Comandos de atribuição
  - Arrays e Registros
  - Expressões booleanas
  - Comandos de decisão
  - Comandos de iteração

26

## Controlando Escopo

27

## Controlando Escopo

- Como estender para múltiplos escopos ?
  - Cada procedimento possui um escopo onde endereços relativos devem ser criados
  - Usar uma tabela de símbolos para cada escopo
    - $P \rightarrow MD$
    - $M \rightarrow \epsilon$
    - $D \rightarrow D; D$
    - $D \rightarrow id : T$
    - $D \rightarrow \text{proc id}; ND; S$
    - $N \rightarrow \epsilon$

28

## Ações semânticas

- geratab(anterior):
  - cria uma tabela de símbolos retornando um apontador para a mesma. Guarda o valor anterior para a tabela criada anteriormente
- adsimb(tabela, nome, tipo, deslocamento)
  - cria uma nova entrada para nome na tabela de símbolos
- deftam(tabela, largura):
  - registra a largura acumulada de todas as entradas de tabela no cabeçalho associado a tabela de símbolos
- adproc(tabela, nome, tipo, deslocamento)
  - cria uma nova entrada para o nome de procedimento nome na tabela de símbolos

29

## Estruturas auxiliares

- Pilhas de tabela de símbolos (tabPtr)
- Pilhas de deslocamentos (desloc)

30

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

$P \rightarrow MD$   
 $M \rightarrow \epsilon$   
 $D \rightarrow D; D$   
  
 $D \rightarrow id : T$   
 $D \rightarrow proc\ id; ND; S$   
 $N \rightarrow \epsilon$   
 $T \rightarrow int$   
 $T \rightarrow real$   
 $T \rightarrow array\ [num]\ of\ T_1$   
 $T \rightarrow \wedge T_1$

31

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

$P \rightarrow MD$       {defTam(top(tabPtr),top(desloc));  
                   pop(tabPtr);  
                   pop(desloc)}  
  
 $M \rightarrow \epsilon$   
  
 $D \rightarrow D; D$   
  
 $D \rightarrow id : T$   
 $D \rightarrow proc\ id; ND; S$

32

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

$P \rightarrow MD$       {defTam(top(tabPtr),top(desloc));  
                   pop(tabPtr);  
                   pop(desloc)}  
  
 $M \rightarrow \epsilon$       {t=geraTab(nil);  
                   push(t,tabPtr);  
                   push(0,desloc)}  
  
 $D \rightarrow D; D$   
  
 $D \rightarrow id : T$   
 $D \rightarrow proc\ id; ND; S$

33

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

$P \rightarrow MD$       {defTam(top(tabPtr),top(desloc));  
                   pop(tabPtr);  
                   pop(desloc)}  
  
 $M \rightarrow \epsilon$       {t=geraTab(nil);  
                   push(t,tabPtr);  
                   push(0,desloc)}  
  
 $D \rightarrow D; D$   
  
 $D \rightarrow id : T$       {adSimb (top(tabPtr),id.nome, T.tipo, top(desloc));  
                   top(desloc)=top(desloc) + T.tam}  
  
 $D \rightarrow proc\ id; ND; S$

34

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

$P \rightarrow MD$       {defTam(top(tabPtr),top(desloc));  
                   pop(tabPtr);  
                   pop(desloc)}  
  
 $M \rightarrow \epsilon$       {t=geraTab(nil);  
                   push(t,tabPtr);  
                   push(0,desloc)}  
  
 $D \rightarrow D; D$   
  
 $D \rightarrow id : T$       {adSimb (top(tabPtr),id.nome, T.tipo, top(desloc));  
                   top(desloc)=top(desloc) + T.tam}  
  
 $D \rightarrow proc\ id; ND; S$       {t=top(tabPtr);  
                   defTam(t,top(desloc));  
                   pop(tabPtr);pop(desloc);  
                   adProc(top(tabPtr),id.nome, t)}

35

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

$N \rightarrow \epsilon$   
 $T \rightarrow int$   
 $T \rightarrow real$   
 $T \rightarrow array\ [num]\ of\ T_1$   
 $T \rightarrow \wedge T_1$

36

Exemplo: esquema de tradução para gerar uma árvore de tabelas de símbolos

```

N → ε           {t:=geraTab(top(tabPtr));
                 push(t,tabPtr);push(0,desloc)}

T → int         {T.tipo = int; T.tam = 4}

T → real       {T.tipo = real; T.tam = 8}

T → array [num] of Ti  {T.tipo = arranjo(num.val, Ti.tipo);
                       T.tam = num.val * Ti.tam}

T → ^Ti       {T.tipo = ponteiro(Ti.tipo);
               T.tam = 4}
    
```

37

Exemplo

```

a: real;
b: int;
proc p1;
  c: real;
  ---
end p1;
proc p2;
  d: array[5] of int;
  proc p3;
    e,f: real;
    ---
  end p3;
  ---
end p2;
---
    
```

38

Exemplo

```

nil
a, real, 0
    
```

```

a: real;
b: int;
proc p1;
  c: real;
  ---
end p1;
proc p2;
  d: array[5] of int;
  proc p3;
    e,f: real;
    ---
  end p3;
  ---
end p2;
---
    
```

39

Exemplo

```

nil
a, real, 0
b, int, 8
    
```

```

a: real;
b: int;
proc p1;
  c: real;
  ---
end p1;
proc p2;
  d: array[5] of int;
  proc p3;
    e,f: real;
    ---
  end p3;
  ---
end p2;
---
    
```

40

Exemplo

```

nil
a, real, 0
b, int, 8
p1,
c, real, 0
    
```

```

a: real;
b: int;
proc p1;
  c: real;
  ---
end p1;
proc p2;
  d: array[5] of int;
  proc p3;
    e,f: real;
    ---
  end p3;
  ---
end p2;
---
    
```

41

Exemplo

```

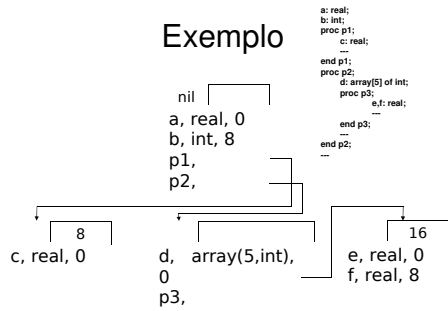
nil
a, real, 0
b, int, 8
p1,
p2,
c, real, 0
d, array(5,int),
0
p3,
    
```

```

a: real;
b: int;
proc p1;
  c: real;
  ---
end p1;
proc p2;
  d: array[5] of int;
  proc p3;
    e,f: real;
    ---
  end p3;
  ---
end p2;
---
    
```

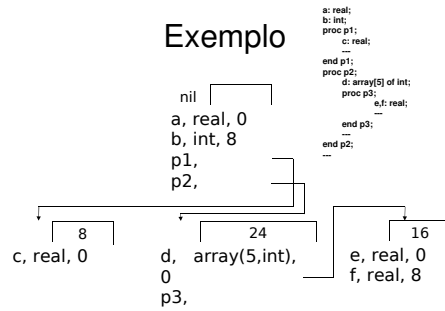
42

### Exemplo



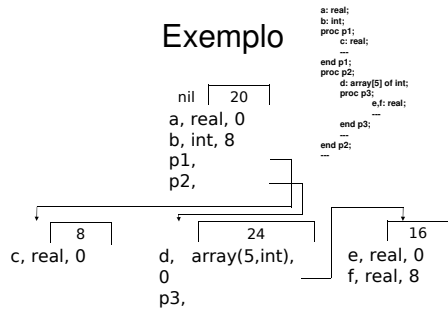
43

### Exemplo



44

### Exemplo



45

### TAC

- Geração de código de três endereços:
  - Expressões
  - Declarações (escopo simples)
  - Declarações (escopos aninhados)
  - Comandos de atribuição
  - Arrays e Registros
  - Expressões booleanas
  - Comandos de decisão
  - Comandos de iteração

46

### Comandos de atribuição

S → id := E	{ p = lookup(id.nome); if p <> NULL then geracod (p := E.ptr) else erro }
E → E1+E2	{E.ptr = geratemp; geracod (E.ptr := E1.ptr "+" E2.ptr) }
E → E1*E2	{E.ptr = geratemp; geracod (E.ptr := E1.ptr "*" E2.ptr) }
E → -E1	{E.ptr = geratemp; geracod (E.ptr := "-" E1.ptr) }
E → E1	{E.ptr = E1.ptr;}
E → id	{p=lookup(id.nome); if p<> NULL then E.ptr = p; else erro }

47

### TAC

- Geração de código de três endereços:
  - Expressões
  - Declarações (escopo simples)
  - Declarações (escopos aninhados)
  - Comandos de atribuição
  - Arrays e Registros
  - Expressões booleanas
  - Comandos de decisão
  - Comandos de iteração

48

## Arrays

- Como acessar elementos ?  $A[i]$

49

## Arrays

- Como acessar elementos ?  $A[i]$ 
  - $\text{base} + (i - \text{linf}) * w$ 
    - base: endereço de memória relativo para o array
    - linf: limite inferior do intervalo de subscritos
    - w: largura de cada elemento do array
    - i: índice que se deseja acessar

50

## Arrays

- $\text{base} + (i - \text{linf}) * w$ 
  - base: endereço de memória relativo para o array
  - linf: limite inferior do intervalo de subscritos
  - w: largura de cada elemento do array
  - i: índice que se deseja acessar
- Rearranjar:
  - $i * w + (\text{base} - \text{linf} * w)$

calculado em tempo de compilação

51

## Arrays 2D

- Como acessar elementos ?  $A[i_1, i_2]$

52

## Arrays 2D

- $A[i_1, i_2]$ 
  - base: endereço de memória relativo para o array
  - linf1, linf2: limite inferior do intervalo de subscritos
  - w: largura de cada elemento do array
  - i1, i2: índice que se deseja acessar
  - n2:  $\text{lsup2} - \text{linf2} + 1$

53

## Arrays 2D

- $A[i_1, i_2]$   $\text{base} + ((i_1 - \text{linf1}) * n_2 + i_2 - \text{linf2}) * w$ 
  - base: endereço de memória relativo para o array
  - linf1, linf2: limite inferior do intervalo de subscritos
  - w: largura de cada elemento do array
  - i1, i2: índice que se deseja acessar
  - n2:  $\text{lsup2} - \text{linf2} + 1$

54

## Tradução de Arrays 2D

- $base + ((i1 - \text{linf1}) * n2 + i2 - \text{linf2}) * w$ 
  - base: endereço de memória relativo para o array
  - $\text{linf1}, \text{linf2}$ : limite inferior do intervalo de subscritos
  - w: largura de cada elemento do array
  - $i1, i2$ : índice que se deseja acessar
  - $n2: \text{lsup2} - \text{linf2} + 1$
- Rearranjar:
  - $((i1 * n2) + i2) * w + (base - ((\text{linf1} * n2) + \text{linf2}) * w)$

calculado e armazenado em c(array)

55

## Generalização

- Fórmula geral:
  - $((\dots((i_1 n_2 + i_2) n_3 + i_3) \dots) n_k + i_k) * w +$   
 $base - ((\dots((\text{linf}_1 n_2 + \text{linf}_2) n_3 + \text{linf}_3) \dots) n_k + \text{linf}_k) * w$
- Recorrência:
  - $e_1 = i_1$
  - $e_m = e_{m-1} * n_m + i_m$

56

## Tradução de Arrays

- L → id [Elist] | id
- Elist → Elist , E | E

57

## Tradução de Arrays

- L → id [Elist] | id
- Elist → Elist , E | E
  
- L → Elist] | id
- Elist → Elist , E | id [ E

58

## Tradução de Arrays

- L → id [Elist] | id
- Elist → Elist , E | E
  
- L → Elist] | id
- Elist → Elist , E | id [ E

Facilita atrelar o nome do array à expressão de índice mais a esquerda (atributo sintetizado ao invés de herdado)

59

## Exemplo

1. S → L:=E
2. E → E+E
3. E → (E)
4. E → L
5. L → Elist]
6. L → id
7. Elist → Elist, E
8. Elist → id[E

60

## Exemplo

- $x := A[y,z]$ 
  - A é uma array de 10x20 com  $\text{linf1}=\text{linf2}=1$ ,  $w=4$
- código gerado:
  - $t1 := y * 20$
  - $t1 := t1 + z$
  - $t2 := c$  // constante  $c = \text{baseA} - 84$
  - $t3 := 4 * t1$   $\text{base} - ((\text{linf1} * n2) + \text{linf2}) * w$
  - $t4 := t2 [ t3]$
  - $x := t4$

61

## Atributos semânticos

- Atributos:
  - local: ponteiro para um dado nome contendo um valor
  - deslocamento: deslocamento dentro do array (nulo quando for identificador simples)
  - array: ponteiro para a tabela de símbolos contendo a entrada do array
- Variáveis, funções:
  - ndim: registra o número de dimensões em uma lista de índice
  - $\text{limite}(\text{array}, j)$  retorna  $n_j$ , o número de elementos da  $j$ -ésima dimensão

62

## Exemplo

S	→	L:=E
{ IF L.deslocamento = NULL		
THEN // L é um identificador simples		
emitir (L.local := ' E.local)		
<b>ELSE</b>		
emitir (L.local [' L.deslocamento '] := 'E.local))		
1.	S	→ L:=E
2.	E	→ E+E
3.	E	→ (E)
4.	E	→ L
5.	L	→ Elist]
6.	L	→ id
7.	Elist	→ Elist, E
8.	Elist	→ id[E

## Exemplo

E	→	$E_1 + E_2$
{ E.local := geratemp;		
emitir (E.local := $E_1.local + E_2.local$ )}		
E	→	( $E_1$ )
{ E.local := $E_1.local$ }		
1.	S	→ L:=E
2.	E	→ E+E
3.	E	→ (E)
4.	E	→ L
5.	L	→ Elist]
6.	L	→ id
7.	Elist	→ Elist, E
8.	Elist	→ id[E

## Exemplo

E	→	L
{ IF L.deslocamento = NULL		
THEN // L é um identificador simples		
E.local := L.local		
<b>ELSE</b>		
E.local := geratemp;		
emitir (E.local := ' L.local [' L.deslocamento'] )}		
1.	S	→ L:=E
2.	E	→ E+E
3.	E	→ (E)
4.	E	→ L
5.	L	→ Elist]
6.	L	→ id
7.	Elist	→ Elist, E
8.	Elist	→ id[E

## Exemplo

L	→	Elist ]
{ L.local := geratemp;		
L.deslocamento := geratemp;		
emitir (L.local := ' c(Elist.array));		
emitir (L.deslocamento := ' Elist.local '*' largura(Elist.array))}		
1.	S	→ L:=E
2.	E	→ E+E
3.	E	→ (E)
4.	E	→ L
5.	L	→ Elist]
6.	L	→ id
7.	Elist	→ Elist, E
8.	Elist	→ id[E

$i * w + (\text{base} - \text{linf} * w)$   
 constante usada no cálculo do índice  
 w usado no cálculo do índice

### Exemplo

L → id  
 { L.local := id.local;  
 L.deslocamento := null }

1.	S	→	L:=E
2.	E	→	E+E
3.	E	→	(E)
4.	E	→	L
5.	L	→	Elist
6.	L	→	id
7.	Elist	→	Elist, E
8.	Elist	→	id E

### Exemplo

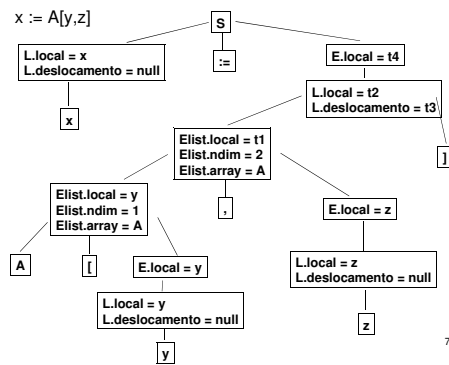
Elist → Elist, E  
 { t := geratemp;  
 m := Elist.ndim + 1;  
 emitir(t := Elist.local \*\* limite(Elist.array,m));  
 emitir(t := t + E.local);  
 Elist.array := Elist.array;  
 Elist.local := t;  
 Elist.ndim := m }

1.	S	→	L:=E
2.	E	→	E+E
3.	E	→	(E)
4.	E	→	L
5.	L	→	Elist
6.	L	→	id
7.	Elist	→	Elist, E
8.	Elist	→	id E

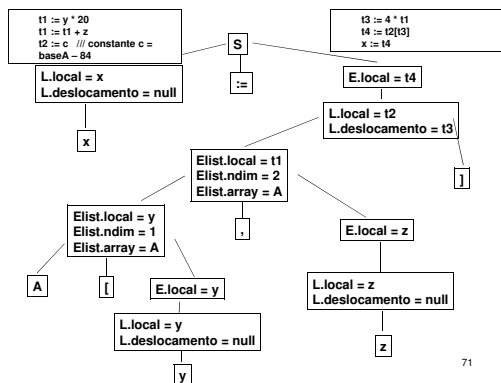
### Exemplo

Elist → id [ E  
 { Elist.array := id.local;  
 Elist.local := E.local;  
 Elist.ndim := 1 }

1.	S	→	L:=E
2.	E	→	E+E
3.	E	→	(E)
4.	E	→	L
5.	L	→	Elist
6.	L	→	id
7.	Elist	→	Elist, E
8.	Elist	→	id E



70



71