

## Compiladores

### Código intermediário (3)

## 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

## Comandos de atribuição

```
S → id := E      { p = lookup(id.nome);  
                  if p <> NULL then S.codigo = E.codigo || geracod (p ":=" E.local)  
                  else erro }
```

```
E → E1+E2      { E.local = geratemp;  
                E.codigo = E1.codigo || E2.codigo ||  
                geracod (E.local ":=" E1.local "+" E2.local) }
```

```
E → E1*E2      { E.local = geratemp;  
                E.codigo = E1.codigo || E2.codigo ||  
                geracod (E.local ":=" E1.local "*" E2.local) }
```

```
E → -E1        { E.local = geratemp;  
                E.codigo = E1.codigo || geracod (E.local ":=" "-" E1.local) }
```

```
E → E1        { E.local = E1.local; }
```

```
E → id         { p=lookup(id.nome);  
                if p<> NULL then E.local = p; else erro }
```

## 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

## Arrays monodimensionais

- Como acessar elementos ?  $A[i]$ 
  - $\text{base} + (i - \text{linf}) * w$ 
    - base: endereço de memória relativo para o array
      - A.local
      - Determinado à compilação
    - linf: limite inferior do intervalo de subscritos
      - Definido à declaração (Fortran) ou por default (C)
    - w: largura de cada elemento do array
      - Definido por projeto do compilador
      - int: 4 Bytes, double: 8 Bytes, etc...
    - i: índice que se deseja acessar
      - Valor determinado à execução!

## Arrays monodimensionais

- $\text{base} + (i - \text{linf}) * w$ 
  - 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:
  - $i * w + (\text{base} - \text{linf} * w)$

## Arrays 2D

$A[i_1, i_2] = \text{base} + ((i_1 - \text{linf}_1) * n_2 + i_2 - \text{linf}_2) * w$

- (armazenamento por linhas!)
- **base**: endereço de memória relativo para o array
- **linf<sub>1</sub>, linf<sub>2</sub>**: limite inferior do intervalo de subscritos
- **w**: largura de cada elemento do array
- **i<sub>1</sub>, i<sub>2</sub>**: índice que se deseja acessar
- **n<sub>2</sub>**:  $\text{lup}_2 - \text{linf}_2 + 1$

- Re-ordenando:
  - $((i_1 * n_2) + i_2) * w + (\text{base} - ((\text{linf}_1 * n_2) + \text{linf}_2) * w)$

## Generalização

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

## Tradução de Arrays

- L  $\rightarrow$  id [Elist] | id
- Elist  $\rightarrow$  Elist , E | E

## Tradução de Arrays

- L  $\rightarrow$  id [ Elist ] | id
- Elist  $\rightarrow$  Elist , E | E

Transformação da gramática para facilitar a análise (atributo sintetizado ao invés de herdado):

- em cima, id é vinculado a Elist (toda a lista depende de id)
- Em baixo: id é vinculado ao arg mais à esquerda (só o 1o argumento depende)

- L  $\rightarrow$  Elist] | id
- Elist  $\rightarrow$  Elist , E | id [ E

## Exemplo de GLC com arrays

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

## Exemplo

- $x := A[i, j]$ 
  - A: array(10, 20) of integer
  - A é uma array de dimensões 10x20, com  $\text{linf}_1 = \text{linf}_2 = 1$ ,  $w = 4$
- código gerado:
  - $t1 := i * 20$
  - $t1 := t1 + j$
  - $t2 := c$  // constante  $c = \text{baseA} - 84$
  - $t3 := 4 * t1$
  - $t4 := t2 [ t3]$
  - $x := t4$

## Atributos semânticos para arrays

- **atributos:**
  - **local:** a variável.
  - **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
  - **limite(array, j)** retorna **nj**, o número de elementos da **j**-ésima dimensão

## TAC e arrays (1)

1.  $S \rightarrow L := E$ 
  - { IF L.deslocamento = NULL THEN  
// L é um identificador simples  
emitir (L.local ':=' E.local)  
ELSE  
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     |

## TAC e arrays (2)

2.  $E \rightarrow E_1 + E_2$   
{ E.local := geratemp;  
emitir (E.local := E<sub>1</sub>.local + E<sub>2</sub>.local)}
3.  $E \rightarrow (E_1)$   
{ E.local := E<sub>1</sub>.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     |

## TAC e arrays (3)

4.  $E \rightarrow L$ 
  - { IF L.deslocamento = NULL THEN  
// L é um identificador simples  
E.local := L.local  
ELSE  
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     |

## TAC e arrays (4)

5.  $L \rightarrow Elist]$  constante usada no cálculo do índice  
{ L.local := geratemp;  
L.deslocamento := geratemp;  
emitir (L.local ':=' c(Elist.array));  
emitir (L.deslocamento ':=' Elist.local '\*'  
largura(Elist.array))}

$i * w + (base - linf * w)$

w usado no cálculo do índice

|    |       |   |          |
|----|-------|---|----------|
| 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     |

## TAC e arrays (5)

6.  $L \rightarrow id$   
{ 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     |

## TAC e arrays (6)

7. Elist  $\rightarrow$  EList<sub>1</sub>, E

```
{ t := geratemp;
  m := elist1.ndim + 1;
  emitir(t := EList1.local * limite(EList1.array, m));
  emitir(t := t + E.local);
  Elist.array := EList1.array;
  Elist.local := t;
  Elist.ndim := m }
```

|    |       |               |          |
|----|-------|---------------|----------|
| 1. | S     | $\rightarrow$ | L:=E     |
| 2. | E     | $\rightarrow$ | E+E      |
| 3. | E     | $\rightarrow$ | (E)      |
| 4. | E     | $\rightarrow$ | L        |
| 5. | L     | $\rightarrow$ | Elist    |
| 6. | L     | $\rightarrow$ | id       |
| 7. | Elist | $\rightarrow$ | Elist, E |
| 8. | Elist | $\rightarrow$ | id[E     |

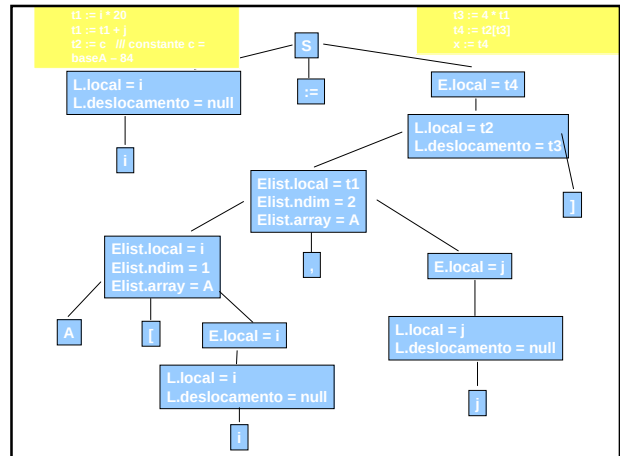
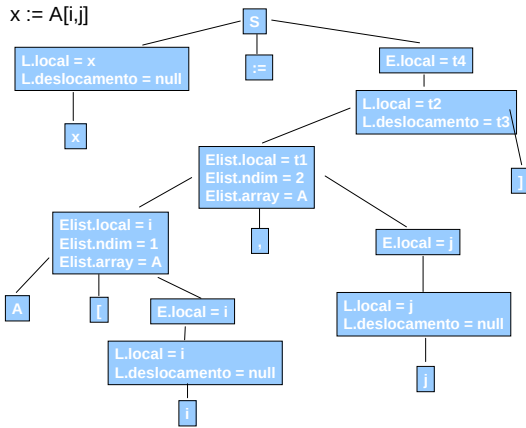
## TAC e arrays (7)

8. Elist  $\rightarrow$  id [ E

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

|    |       |               |          |
|----|-------|---------------|----------|
| 1. | S     | $\rightarrow$ | L:=E     |
| 2. | E     | $\rightarrow$ | E+E      |
| 3. | E     | $\rightarrow$ | (E)      |
| 4. | E     | $\rightarrow$ | L        |
| 5. | L     | $\rightarrow$ | Elist    |
| 6. | L     | $\rightarrow$ | id       |
| 7. | Elist | $\rightarrow$ | Elist, E |
| 8. | Elist | $\rightarrow$ | id[E     |

x := A[i,j]



## Próxima aula

Geração de código TAC (4): estruturas de controle de fluxo.