Geração de código

Professor: Luciano Ferreira Silva, Dr.

Geração de código

1. **Geração de código intermediário;**
   - Cria-se uma nova representação, não tão abstrata como a árvore sintática nem tão específica como a linguagem simbólica;
2. **Otimização de código;**
   - Normalmente utiliza alguma técnica de otimização para retirar fragmentos desnecessários ou redundantes e produzir um código melhor;
3. **Transferência do código para linguagem simbólica;**
   - Ocorre a tradução do código em formato intermediário para o código em linguagem simbólica de um processador específico.

Geração de código intermediário

- **Durante a análise sintática comandos elementares do código já foram reconhecidos;**
  - ✓ Os nós internos da árvore sintática obtida para as diversa expressões do programa são exemplos destes comandos elementares;
- **Logo é possível:**
  - ✓ Construir a árvore sintática por meio da análise sintática;
  - ✓ Percorrer essa árvore para o código na linguagem de destino;
- **Em geral esta produção durante as ações do analisador sintático;**
  - ✓ Esse procedimento é denominado tradução dirigida pela sintaxe.

Geração de código intermediário

- **A geração de código não se dá diretamente para a linguagem simbólica do processador-alvo;**
  - ✓ O analisador sintático gera código para uma máquina abstrata;
    - Contextualizada com a linguagem simbólica mas independente de processadores específicos;
- **Em uma segunda etapa ocorre a otimização e transferência para a linguagem simbólica;**
  - ✓ Este processo facilita a otimização;
  - ✓ E permite o reaproveitamento de uma grande parte do compilador em casos de diferentes processadores.
- **Uma das formas mais usuais para esse tipo de representação é o código de três endereços.**

Código de três endereços

- **Define uma seqüência de instruções envolvendo operações com uma atribuição ou instruções de desvio;**
- **O nome "três endereços" está associada à especificação:**
- **Pode usar no máximo três variáveis;**
  - ✓ Para operações binárias, por exemplo, duas para os operandos e uma para o resultado;
- **Expressões complexas envolvendo diversas operações devem ser decompostas em uma série de instruções elementares;**
  - ✓ Eventualmente pode ser necessário utilizar variáveis temporárias.

Código de três endereços

- **A especificação de uma linguagem de três endereços envolve quatro tipos básicos de instruções:**
  - ✓Expressões com atribuição;
  - ✓Desvios;
  - ✓Invocação de rotinas;
  - ✓Modos de endereçamento.

Instruções de atribuição

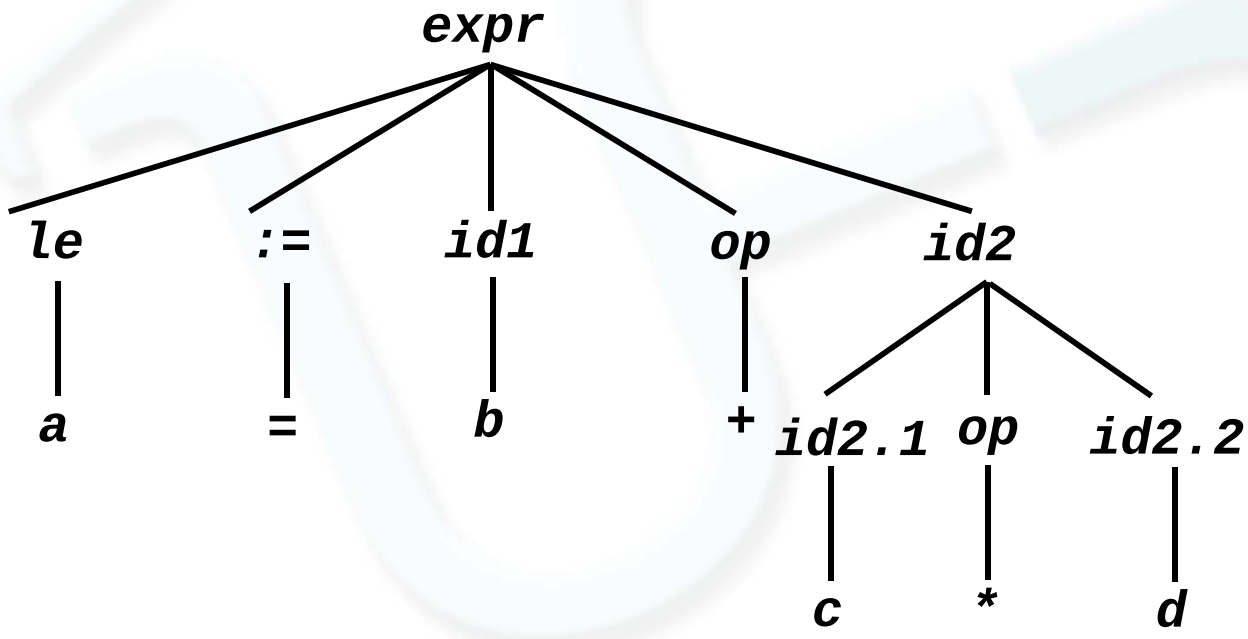
- **São aquelas nas quais:**
  - ✓ o resultado de uma operação é armazenado na variável especificada à esquerda do operador de atribuição (denotado aqui por :=);
- **São consideradas três formas básicas, nas quais o operador de atribuição pode ser usado para:**
- **Copiar o valor de uma variável para outra:**
  - ✓ ***le := id***
  - ✓ Exemplo, em C++: ***a = c;***
- **Armazenar o resultado de uma operação binária:**
  - ✓ ***le := id1 op id2***
  - ✓ Exemplo, em C++: ***b = a + d;***
- **Armazenar o resultado da aplicação de um operador unário:**
  - ✓ ***le := op id***
  - ✓ Exemplo, em C++: ***e = -b;***

Instruções de atribuição

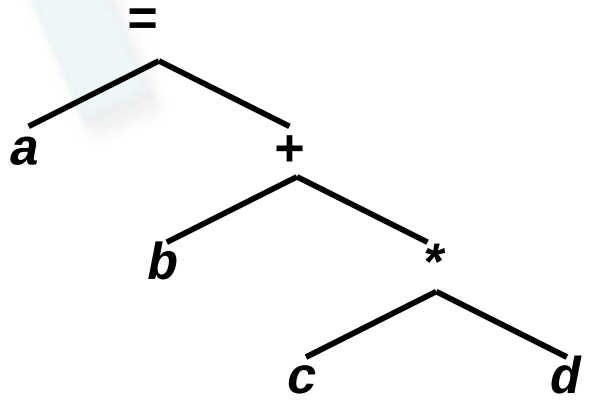
- Expressões mais complexas demandam variáveis intermediárias;
- Por exemplo, considere a expressão:

```
a = b + c * d;
```

- Sua árvore sintática pode ser:

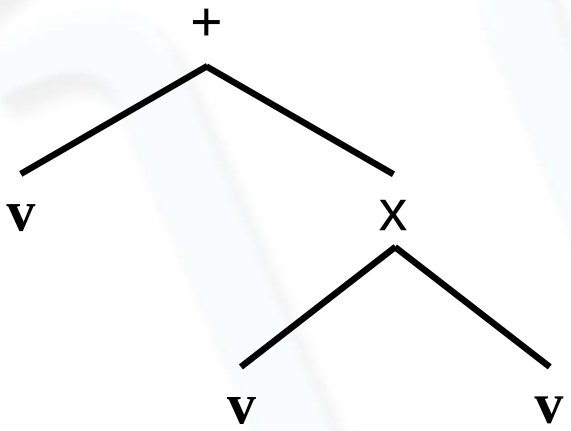


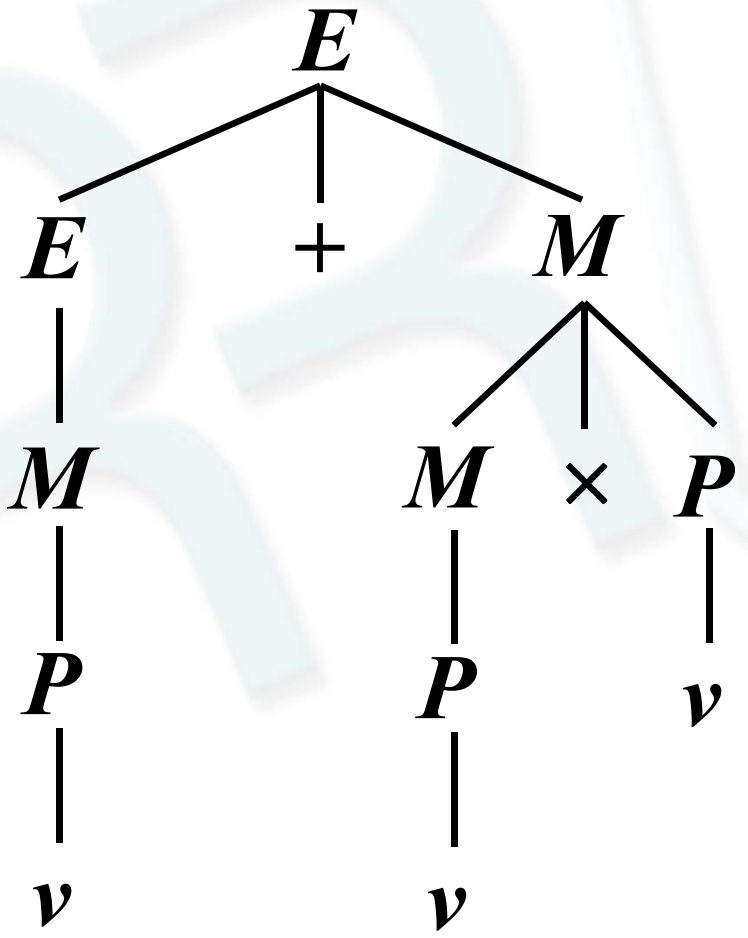
- Esta árvore é chamada de árvore simplificada:



Simplificação de árvores estudadas

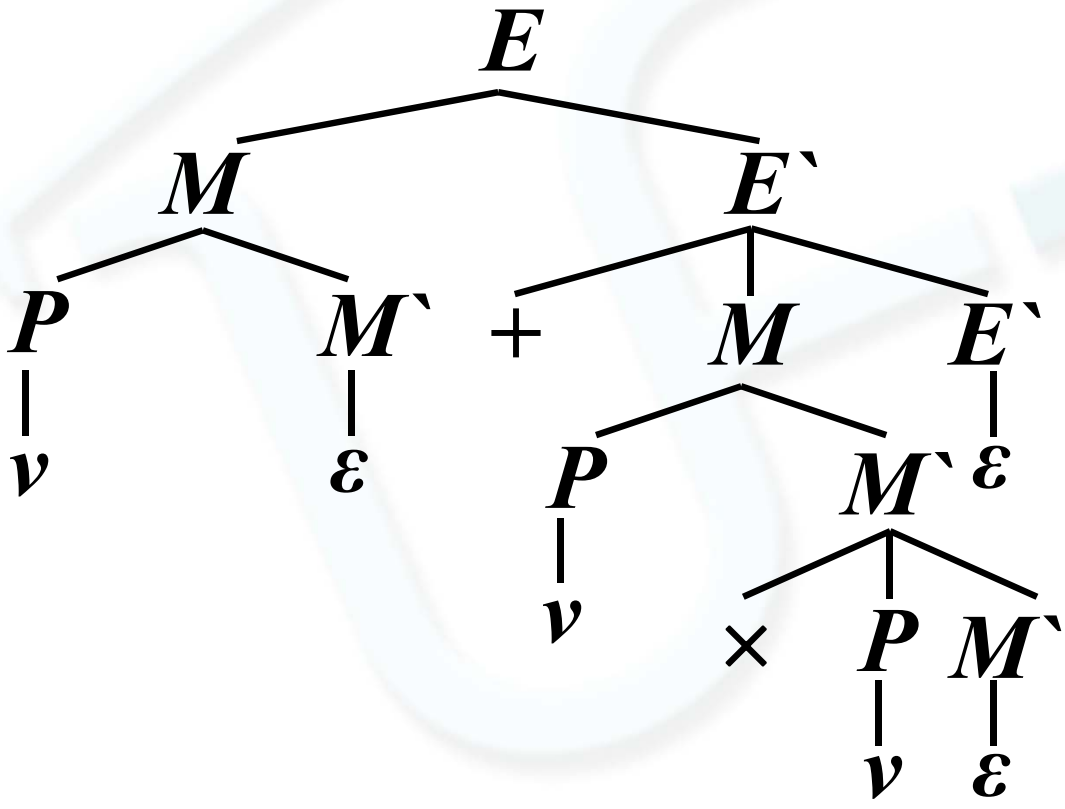
- *v + v x v;*
- **Gramática G`;**
- **Analisador de precedência fraca:**

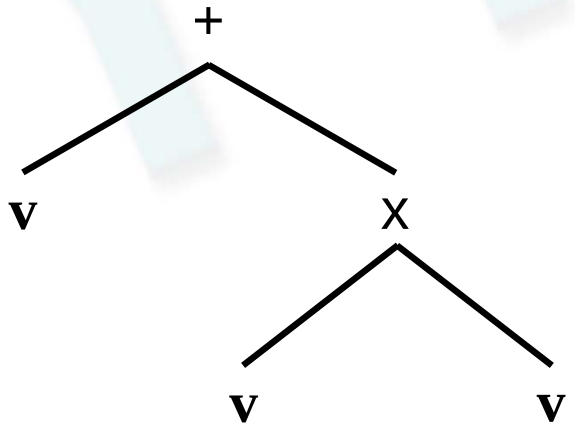




Simplificação de árvores estudadas

- *v + v x v;*
- **Gramática G``;**
- **Analisador preditivo:**

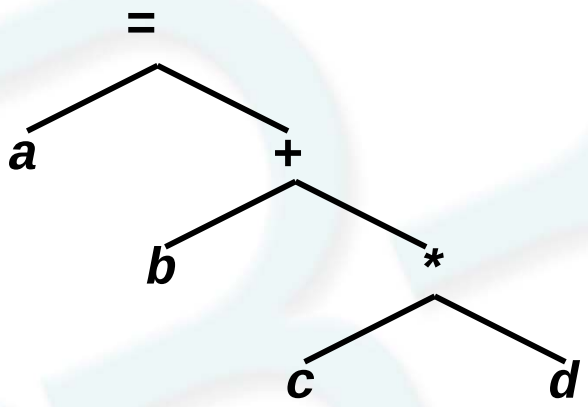




Instruções de atribuição

- Código intermediário gerado:

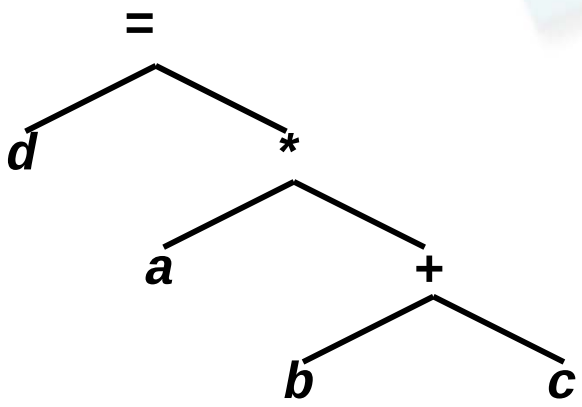
```
_t1 := c * d
 a := b + _t1
```



- Não é necessário o uso de parênteses:

```
d = a * (b + c);
```

```
_t1 := b + c
 d := a * _t1
```



Instruções de desvio

- Podem assumir duas formas básicas:
- Incondicional:

goto L

- ***L*** **é um rótulo que identifica uma linha de código;**
- Condicional:

if x Op y goto L

- ***Op*** **é um operador relacional de comparação;**
- ***L é o rótulo da linha que deve ser executada se o resultado do operador condicional for verdadeiro***

Instruções de desvio

- Exemplo, em C++:

```
while (i++ <= K)
        x[i] = 0;
x[0] = 0;
```

- Código intermediário:

```
_L1: if i > k goto _L2
        i := i + 1
        x[i] := 0
        goto _ L1
_L2: x[0] := 0
```

Instruções de invocação de sub-rotinas

- **Na linguagem simbólica é realizada por:**
  - ✓ Uma instrução que altera o ponto de execução para outra região da memória;
  - ✓ Ao mesmo tempo que preserva o ponto de execução atual para restaurá-lo quando a sub-rotina é concluída.

Instruções de invocação de sub-rotinas

- **Na linguagem de código intermediário:**
- **A instrução `call` realiza a invocação da rotina;**
  - ✓ Ela contém duas informações:
    - O nome da rotina;
    - O número de parâmetros;
- **A instrução `return` restaura o ponto de execução após a conclusão da sub-rotina;**

Instruções de invocação de sub-rotinas

- **Para identificar os parâmetros da função usa-se a instrução param na linguagem intermediária;**
  - ✓ Os quais serão tratados, de uma maneira um pouco mais complexa, como elementos de uma pilha, na linguagem simbólica;
- **Exemplo em C++:**

```
x = f(a, b, c);
```

- **Linguagem intermediária:**

```
param a
param b
param c
x := call f, 3
```

Instruções de invocação de sub-rotinas

- Outro exemplo em C++:

```
a = g(b, h(c));
```

- Código intermediário:

```
param b
param c
_t1 := call h, 1
param _t1
a := call g, 2
```

Modos de endereçamento

- **Acesso indireto e indexado;**
- **O modo de endereçamento indireto:**
  - ✓ Está associado à manipulação de variáveis que contêm endereços;
  - ✓ Na linguagem intermediária são usados os mesmos operadores da linguagem C++;
  - ✓ Exemplos:
    - Atribuir o endereço da variável `y` à variável `x`:

```
x := &y
```

    - Atribuir o valor que está armazenado no endereço `x` à variável `w`:

```
w := *x
```

Modos de endereçamento

- Atribuir o valor de *z* à posição endereçada por x:

```
*x := z
```

- A atribuição de um endereço pelo programador **não** faz sentido, portanto, a definição abaixo **não** é permitida:

```
&y := t
```

- **Sendo *p1* e *p2* ponteiros inteiros, considere o exemplo em C++:**

```
*p1 = a + *p2;
```

- **O código intermediário seria:**

```
_t1 := *p2
_t2 := a + _t1
*p1 := _t2
```

Modos de endereçamento

- **O modo de endereçamento indexado:**
  - ✓ É aquele no qual a posição do item de informação acessado é definida a partir da informação de um endereço-base e de um deslocamento (o índice).
    - Atribuir a variável x o conteúdo do endereço indicado pela soma da variável y com o valor i:

```
x := y[i]
```

      – Para linguagem simbólica as referências devem ser traduzidas para bytes;
    - Atribuir o valor de uma variável x a uma variável indexada y:

```
y[i] := x
```

Modos de endereçamento

- Sendo a, b, e s arranjos contendo 10 inteiros (4 bytes cada elemento), considere o exemplo em C++:

```
for(i=0; i<10; i++)
        s[i] = a[i] + b[i];
```

- Linguagem intermediária:

```
    i := 0
_L1: if i >= 10 goto _L2
    _t1 := 4 * i
    _t2 := a[_t1]
    _t3 := 4 * i
    _t4 := b[_t3]
    _t5 := t2 + _t4
    _t6 := 4 * i
    s[_t6] := _t5
    i := i + 1
    goto _L1
_L2:...
```

Representação interna

- **O código em formato intermediário pode ser armazenado em um arquivo em formato texto;**
- **Usualmente a sua representação é feita por meio de tabelas;**
- **Quádruplas são tabelas de quatro colunas:**

1. O operador da linguagem de formato intermediário;
2. O primeiro argumento;
3. O segundo argumento, se presente;
4. O resultado se presente;

Representação interna

- Por exemplo, o código em C++:

```
a = b + c * d;
z = f(a);
```

- Código intermediário:

```
_t1 := c * d
a := b + _t1
param a
z := call f, 1
```

- Tabela de quádruplas:

	operador	arg 1	arg 2	resultado
1	*	c	d	_t1
2	+	b	_t1	a
3	param	a		
4	call	f	1	z

Representação interna

- Considere o trecho de código C++:

```
if(x > 4)
        x = 10;
```

- Código intermediário:

```
_L1: if x <= 4 goto _L2
        x := 10
_L2:....
```

- Tabela de quádruplas:

	operador	arg 1	arg 2	resultado
1	<=	x	4	_t1
2	if	_t1	_L2	
3	:=	10		x

- Usa-se uma lista auxiliar para gravar as pendências de destino na tabela para o comando if.

Representação interna

- **Tabelas de triplas usam três colunas:**
  - ✓ O operador da linguagem de formato intermediário;
  - ✓ O primeiro argumento;
  - ✓ O segundo argumento, se presente.
- **Os resultados de expressões são indicados por meio de referências às suas posições na tabela.**

Representação interna

- **Código intermediário:**

```
_t1 := c * d
a := b + _t1
param a
z := call f, 1
```

- **Tabela de triplas:**

	operador	arg 1	arg 2
1	*	c	d
2	+	b	(1)
3	:=	a	(2)
4	param	a	
5	call	f	1
6	:=	z	(5)