

Compiladores

Geração de Código
Prof. Tales Viegas

Geração de Código

- ▶ Traduzir para a linguagem-alvo a representação da árvore gramatical obtida para as diversas expressões do programa
- ▶ A este procedimento chamamos “Tradução dirigida pela Sintaxe”

Geração de Código

- ▶ Em geral a geração de código se dá para uma “máquina abstrata”, em uma linguagem intermediária próxima do Assembly.
- ▶ Isto permite que seja gerado um código independente de processadores específicos.
- ▶ Após este processo o código é traduzido para o Assembly desejado.

Geração de Código Intermediário

- ▶ Formato intermediário entre a linguagem de alto-nível e a linguagem Assembly
- ▶ Geralmente são utilizadas duas formas para este tipo de representação
 - Código de 3 Endereços
 - Notação pós-fixada

Código de Três Endereços

- ▶ Sequência de operações envolvendo operações binárias ou unárias e uma atribuição
- ▶ No máximo 3 variáveis: 2 para os operadores e 1 para o resultado
- ▶ Instruções envolvendo mais diversas operações são decompostas em uma série de instruções neste formato

Código de Três Endereços

- ▶ Quatro tipos básicos de instruções:
 - Expressões com atribuição
 - Desvios
 - Invocação de rotinas
 - Acesso indexado e indireto

Instruções de Atribuição

- ▶ Instruções onde o resultado de uma operação é armazenado na variável especificada
 - $x := y \text{ op } z$
 - $x := \text{op } y$
 - $x := y$

Instruções de Atribuição

- ▶ Por exemplo: $a = b + c * d$

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

Instruções de Desvio

- ▶ Desvio incondicional

goto L

- Onde L é um rótulo simbólico que representa uma linha de código

- ▶ Desvio condicional

if x opr y goto L

Instruções de Desvio

- ▶ Por exemplo:

```
while (i++ < k)
```

```
    x[i] = 0;
```

```
x[0] = 0;
```

- ▶ Seria traduzido para:

```
_L1: if i > k goto _L2
```

```
    i := i + 1
```

```
    x[i] := 0
```

```
    goto _L1
```

```
_L2: x[0] := 0
```

Invocação de Rotinas

- ▶ Ocorre em duas etapas
 - Os argumentos são registrados com a instrução param
 - A instrução call completa a execução da rotina.
- ▶ A instrução return indica o fim de execução (opcionalmente com valor de retorno)

Invocação de Rotinas

- ▶ Por exemplo:

f (a, b, c)

- ▶ Seria traduzido para:

param a

param b

param c

_t1 := call f, 3

Acesso Indexado e Indireto

► Atribuições indexadas

- $x := y[i]$
- $x[i] := y$

► Atribuições indiretas

- $x := \&y$
- $w := *x$
- $*x := z$

Tipos de Representação

- ▶ Quádruplas (Tabelas com 4 colunas)
- ▶ Ex: $a = b + c * d$

	Operador	arg1	arg2	resultado
1	*	c	d	_t1
2	+	b	_t1	a

Tipos de Representação

- ▶ Triplas (Tabelas com 3 colunas)
- ▶ Ex: $a = b + c * d$

	Operador	arg1	arg2
1	*	c	d
2	+	b	(1)

Notação Pós-Fixada

- ▶ Também conhecida como “Notação Polonesa”
- ▶ Dispensa o uso de parênteses

Notação Pós-Fixada

- ▶ Exemplos

$a * b + c$

$a * (b + c)$

$(a + b) * c$

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

- ▶ Ficaria como:

$a\ b\ * c\ +$

$a\ b\ c\ +\ *\;$

$a\ b\ +\ c\ *\;$

$a\ b\ +\ c\ d\ +\ *\;$

Instruções de Desvio

L jmp

x y L jcc

► Onde jcc pode ser:

- jeq (Jump equal)
- jne (Jump not equal)
- jlt (Jump less than)
- jle (Jump less or equal than)
- jgt (Jump greater than)
- jge (Jump less or greater than)

Notação Pós-Fixada

- ▶ Baseada em pilhas (máquinas de zero endereços)
- ▶ Operandos são incluídos ou removidos da pilha com as instruções push e pop, respectivamente

Exemplo

▶ $a * (b + c)$

push a

push b

push c

add

mult

Otimização de Código

- ▶ Traduções dirigidas pela sintaxe contemplam expressões independentes, logo acabam gerando sequências não tão eficientes
- ▶ A etapa de otimização consiste em aplicar um conjunto de heurísticas para detectar as sequências ineficientes e substituí-las por outras mais eficientes

Otimização de Código

- ▶ Devem manter o significado do programa original
- ▶ Devem capturar a maior parte das melhorias do código, dentro de limites razoáveis de esforço gasto para tal fim
- ▶ Geralmente os compiladores permitem especificar o grau de esforço desejado no processo de otimização

Otimização de Código

- ▶ Eliminação de desvios desnecessários

- ▶ O código:

```
<a>
```

```
  goto _L1
```

```
_L1: <b>
```

- ▶ Pode ser traduzido para:

```
<a>
```

```
_L1: <b>
```

Otimização de Código

- ▶ Eliminação de código redundante
- ▶ Eliminação de código não-alcançável

goto _L1

x := y

_L1: ...

Otimização de Código

- ▶ Eliminação de Expressões Comuns

- ▶ Exemplo:

$a = b + c$

$e = c + d + b$

- ▶ Original:

$a := b + c$

$_t1 := c + d$

$e := _t1 + b$

- ▶ Poderia ser traduzido para:

$a := b + c$

$e := a + d$