

Construção de Compiladores

Prof. Dr. Daniel Lucrédio

DC - Departamento de Computação

UFSCar - Universidade Federal de São Carlos

Tópico 08 - Geração de Código

Referências bibliográficas

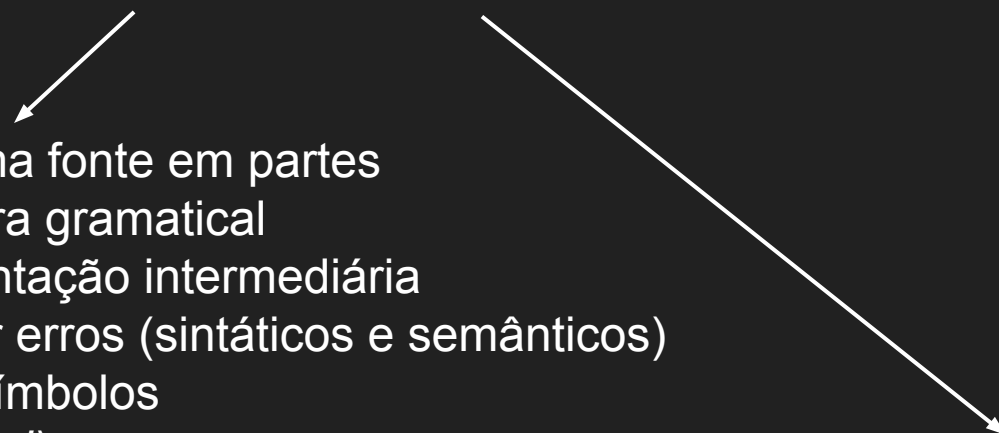
Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Compiladores: Princípios, Técnicas e Ferramentas (2a. edição). Pearson, 2008.

Kenneth C. Loudon. Compiladores: Princípios E Práticas (1a. edição). Cengage Learning, 2004.

Terence Parr. The Definitive Antlr 4 Reference (2a. edição). Pragmatic Bookshelf, 2013.

Estrutura de um compilador

Duas etapas: análise e síntese



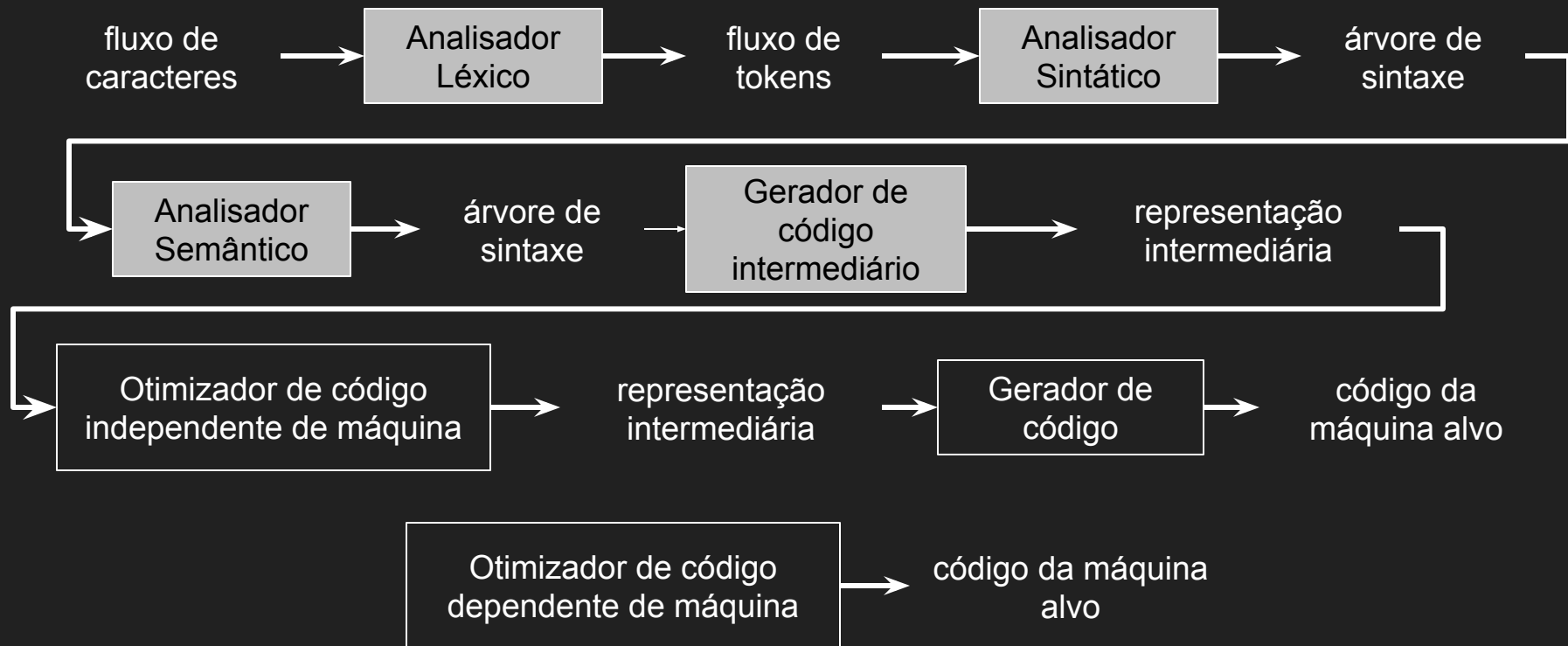
Quebrar o programa fonte em partes
Impor uma estrutura gramatical
Criar uma representação intermediária
Detectar e reportar erros (sintáticos e semânticos)
Criar a tabela de símbolos
(front-end)

Construir o programa objeto
com base na representação intermediária
e na tabela de símbolos
(back-end)

Fases de um compilador

front-end

back-end



Geração de código

- O que é?
 - Etapa na qual o programa-fonte (estático) é transformado em código-alvo (executável)
 - Etapa mais complexa de um compilador, pois depende de
 - Características da linguagem-fonte
 - Informações detalhadas da arquitetura-alvo
 - Estrutura do ambiente de execução
 - Sistema operacional da máquina-alvo
- Envolve também tentativas de otimizar ou melhorar a velocidade e/ou tamanho do código-alvo

Geração de código intermediário

- O que é?
 - Etapa na qual é gerada uma interpretação intermediária explícita para o programa fonte
- Código intermediário X Código alvo
 - O código intermediário não especifica detalhes
 - Quais registradores serão usados, quais endereços de memória serão referenciados etc.

Código intermediário

- Exemplo: C pode ser usada como R.I.
 - O compilador C é o back-end
 - Reutilização de back-end
 - Evita a necessidade de se construir um novo
 - O compilador C++ original era assim
 - Mas depois evoluiu, permitindo um processo mais otimizado



Gerador de código C

Vamos gerar código C para a linguagem ALGUMA

Demonstração 1

Código intermediário

- Veremos 2 exemplos:
 - Código de três endereços
 - P-código

Código de 3-endereços

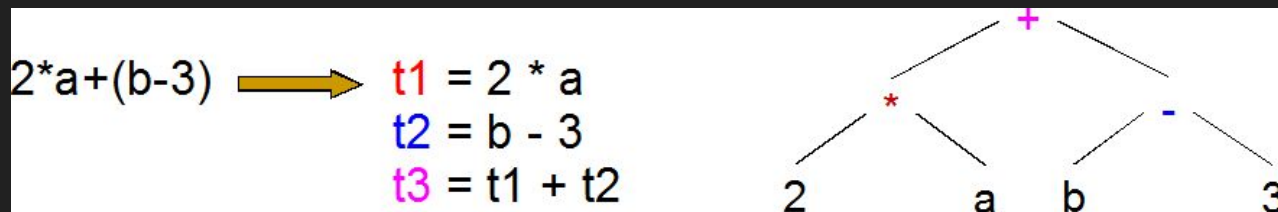
- Instrução básica
- Avaliação de expressões aritméticas

$$x = y \text{ op } z$$

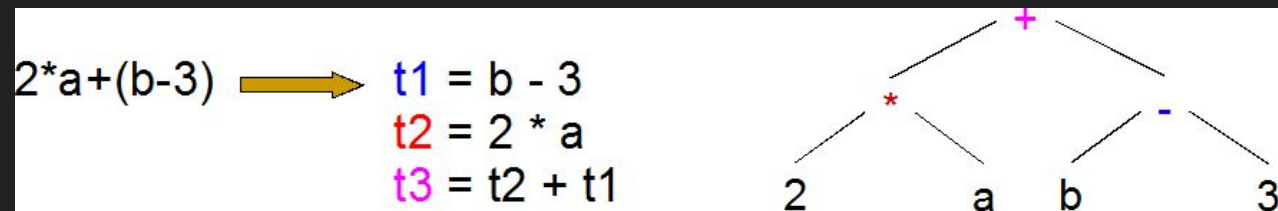
- **op** é um operador aritmético (+ ou -, por exemplo)
- O nome advém dessa forma de instrução na qual ocorre a manipulação de até 3 endereços na memória: x, y e z
- Nem sempre as instruções seguem esse formato
Ex: **a = - b**

Código de 3-endereços

- Exemplo

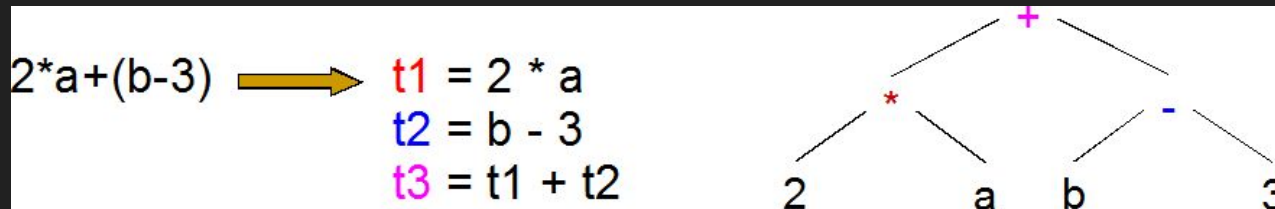


- Código de três endereços obtido da árvore sintática
 - Por meio da linearização da esquerda para a direita
 - Percurso em pós-ordem
- Outra possibilidade



Código de 3-endereços

- Exemplo



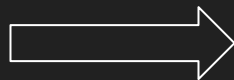
- Temporários

- Os nomes (t1, t2 e t3) devem ser diferentes dos nomes existentes no código-fonte
- Correspondem aos nós interiores da árvore sintática e representam seus valores computados
- O temporário final (t3) representa o valor da raiz
- Podem ser mantidos na memória ou em registradores

Código de 3-endereços

- Ex: Programa que computa o fatorial (em uma linguagem fictícia) e um possível código de três endereços

```
{ Programa exemplo
  -- computa o fatorial
}
read x; { inteiro de entrada }
if 0 < x then { não computa se x <= 0 }
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact { fatorial de x como saída }
end
```



```
read x
t1 = x > 0
if_false t1 goto L1
fact = 1
label L2
t2 = fact * x
fact = t2
t3 = x - 1
x = t3
t4 = x == 0
if_false t4 goto L2
write fact
label L1
halt
```

P-código

- Surgiu como um código de montagem-alvo padrão
 - Produzido pelos compiladores Pascal
- Foi projetado como código de uma máquina hipotética baseada em pilhas
 - P-máquina
 - Com interpretador para diversas máquinas reais

P-código

- Características
 - Projetado para ser executado diretamente
 - Então, contém uma descrição implícita de um ambiente de execução
 - Informações específicas da P-máquina (tamanho de dados, formação da memória etc.)
- Representação e implementação similares ao código de três endereços

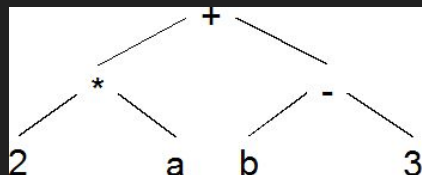
P-código

- Iremos considerar uma versão abstrata simplificada da P-máquina
 - Memória de código
 - Memória de dados
 - não especificada para variáveis com nomes
- Pilha de dados temporários
 - Registradores para manter a pilha e dar suporte à execução

P-código

- Exemplo

$$2*a+(b-3)$$



```
ldc 2      ; carrega constante 2
lod a      ; carrega valor da variável a
mpi        ; multiplicação de inteiros
lod b      ; carrega valor da variável b
ldc 3      ; carrega constante 3
sbi        ; subtração de inteiros
adi        ; adição de inteiros
```

P-código

- Exemplo

$x := y + 1$

```
lda x      ; carrega endereço de x
lod y      ; carrega valor de y
ldc 1      ; carrega constante 1
adi        ; adição
sto        ; armazena topo no endereço
           ; abaixo do topo & retira os dois
```

P-código

- Exemplo

$x := y + 1$

```
lda x      ; carrega endereço de x
lod y      ; carrega valor de y
ldc 1      ; carrega constante 1
adi        ; adição
sto        ; armazena topo no endereço
           ; abaixo do topo & retira os dois
```

Diferença
entre lda e lod

P-código

Exemplo - programa
que computa o fatorial
(em uma linguagem
fictícia) e seu P-código

```
{ Programa exemplo
  -- computa o fatorial
}
read x; { inteiro de entrada }
if 0 < x then { não computa se x <= 0 }
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact { fatorial de x como saída }
end
```

```
lda x      ; carrega endereço de x
rdi        ; lê um inteiro, armazena no
           ; endereço no topo da pilha (& o retira)
lod x      ; carrega o valor de x
ldc 0      ; carrega a constante 0
grt        ; retira da pilha e compara os dois valores do topo
           ; coloca na pilha o resultado booleano
fjp L1     ; retira o valor booleano, salta para L1 se falso
lda fact   ; carrega endereço de fact
ldc 1      ; carrega constante 1
sto        ; retira dois valores, armazena primeiro
           ; em endereço representado pelo segundo
lab L2     ; definição do rótulo L2
lda fact   ; carrega endereço de fact
lod fact   ; carrega valor de fact
lod x      ; carrega valor de x
mpi        ; multiplica
sto        ; armazena topo em endereço do segundo & retira
lda x      ; carrega endereço de x
lod x      ; carrega valor de x
ldc 1      ; carrega constante 1
sbi        ; subtrai
sto        ; armazena (como no caso anterior)
lod x      ; carrega valor de x
ldc 0      ; carrega constante de 0
equ        ; teste de igualdade
fjp L2     ; salta para L2 se falso
lod fact   ; carrega valor de fact
wri        ; escreve topo da pilha & retira
lab L1     ; definição do rótulo L1
stp
```

Geração de código

- Veremos agora alguns exemplos de como traduzir o programa-fonte
 - Já analisado sintaticamente e semanticamente
- Para código intermediário
 - Pronto para otimização / geração de código-alvo
- Veremos apenas algumas situações
 - Para outras, consulte as referências da disciplina
 - A ideia geral é a mesma
 - Mas os detalhes para cada situação são muitos

Declarações de controle

- Ex:

```
if ( E ) S1 else S2
```

- Código de 3-endereços

```
<código para t1 = avaliação de E>
```

```
if_false t1 goto L1
```

```
<código para S1>
```

```
goto L2
```

```
label L1
```

```
<código para S2>
```

```
label L2
```

Obs:

if_false = testa se t1 é falso

goto = salto incondicional

Declarações de controle

- Ex:

```
if ( E ) S1 else S2
```

- P-código

```
<código para avaliar E>
```

```
fjp L1
```

```
<código para S1>
```

```
ujp L2
```

```
lab L1
```

```
<código para S2>
```

```
lab L2
```

Obs:

fjp = salta se valor no topo da pilha é falso

ujp = salto incondicional

Declarações de controle

- Ex:

```
while ( E ) S
```

- Código de 3-endereços

```
label L1  
<código para t1 = avaliação de E>  
if_false t1 goto L2  
<código para S>  
goto L1  
label L2
```

Obs:

if_false = testa se t1 é falso
goto = salto incondicional

Declarações de controle

- Ex:

```
while ( E ) S
```

- P-código

```
lab L1  
<código para avaliar E>  
fjp L2  
<código para S>  
ujp L1  
lab L2
```

Obs:

fjp = salta se valor no topo da pilha é falso

ujp = salto incondicional

Geração de código

Outras tarefas:

- Expressões lógicas
- Cálculo de endereços
- Referências de matrizes
- Estrutura de registros
- Referências de ponteiros
- Geração de rótulos
- Ajuste retroativo (backpatching)
- Chamadas de procedimentos e funções

Demonstração

- Agora veremos um processo completo
 - Geração de código
- Usaremos um ambiente de execução simples
 - Baseado em P-código
 - Totalmente estático
 - Sem procedimentos
- Durante a demonstração, tente imaginar o que seria necessário para
 - Procedimentos
 - Recursividade



Linguagem ALGUMA

Demonstração 2

Uma outra abordagem



Xtext



Syntax Coloring
Semantic Coloring
Error Checking
Auto-Completion
Formatting
Hover Information
Mark Occurences
Go To Declaration

Rename Refactoring
Debugging
Toggle Comments
Outline / Structure View
Quick Fix Proposals
Find References
Call Hierarchy
Type Hierarchy
Folding

<https://www.eclipse.org/Xtext>

Fim