

Compiladores

Aula 1

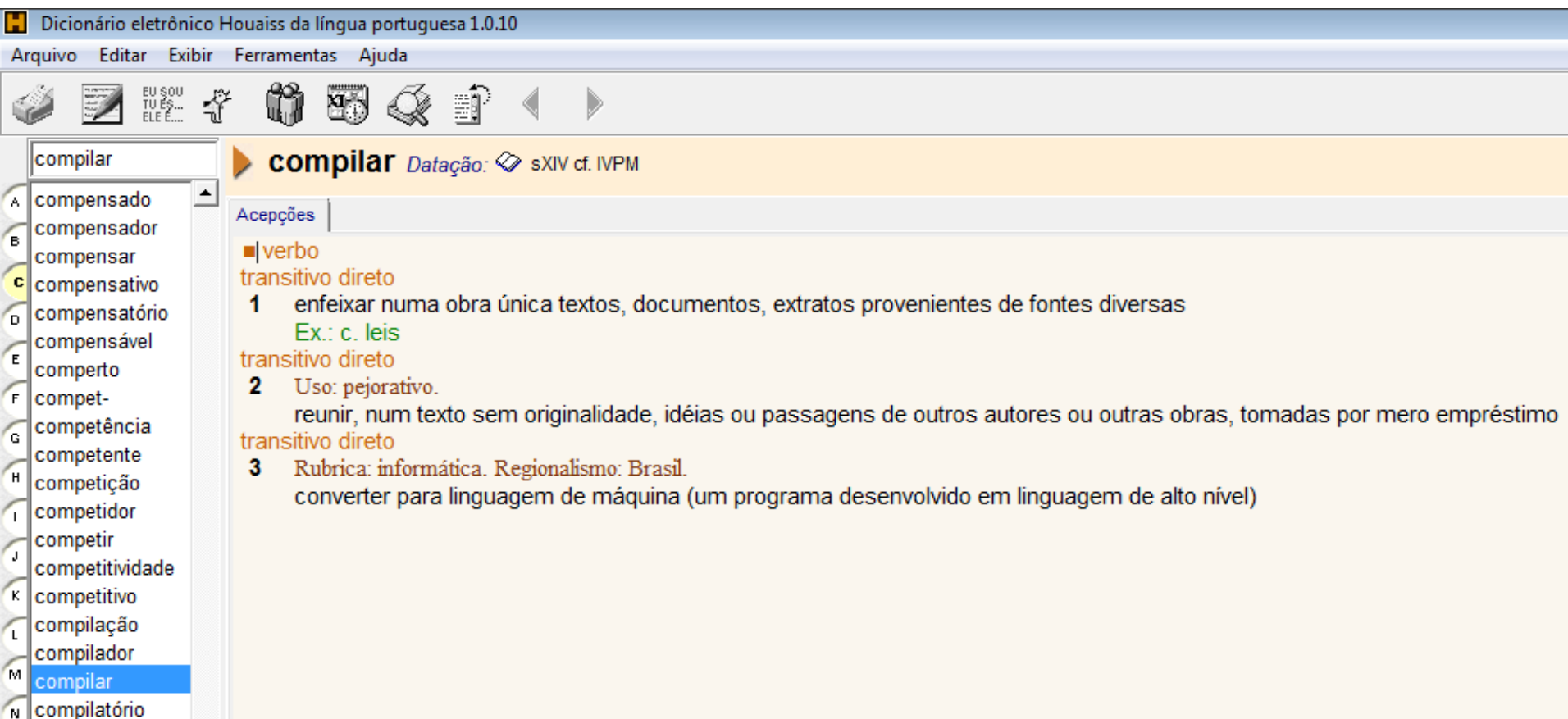
Introdução

Prof. Dr. Luiz Eduardo G. Martins



Compiladores: Introdução

- O que significa a palavra **compilar** ?



The screenshot shows the interface of the 'Dicionário eletrônico Houaiss da língua portuguesa 1.0.10'. The search term 'compilar' is entered in the search bar. The left sidebar lists related words, with 'compilar' highlighted. The main area displays the definition of 'compilar' as a verb, with three numbered entries:

compilar *Datação:* sXIV cf. IVPM

Acepções

- 1 **verbo transitivo direto**
enfeixar numa obra única textos, documentos, extratos provenientes de fontes diversas
Ex.: c. leis
- 2 **verbo transitivo direto**
Uso: pejorativo.
reunir, num texto sem originalidade, idéias ou passagens de outros autores ou outras obras, tomadas por mero empréstimo
- 3 **verbo transitivo direto**
Rubrica: informática. Regionalismo: Brasil.
converter para linguagem de máquina (um programa desenvolvido em linguagem de alto nível)

Compiladores: Introdução

- O que é um compilador ?
 - É um programa de computador que traduz uma linguagem para outra

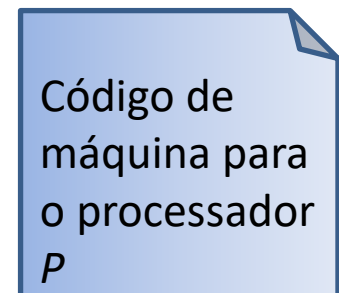
Programa escrito
na
linguagem-fonte



Compilador



Programa escrito
na
Linguagem-alvo



Compiladores: Introdução

- Importância dos Compiladores
 - Sem eles, ainda estaríamos na “idade da pedra” do desenvolvimento de *software*
 - Escrevendo programas usando linguagem de máquina (código numérico)



1101 0110 1101
1001 1101 0101

Ex: processadores 8x86 – IBM PC

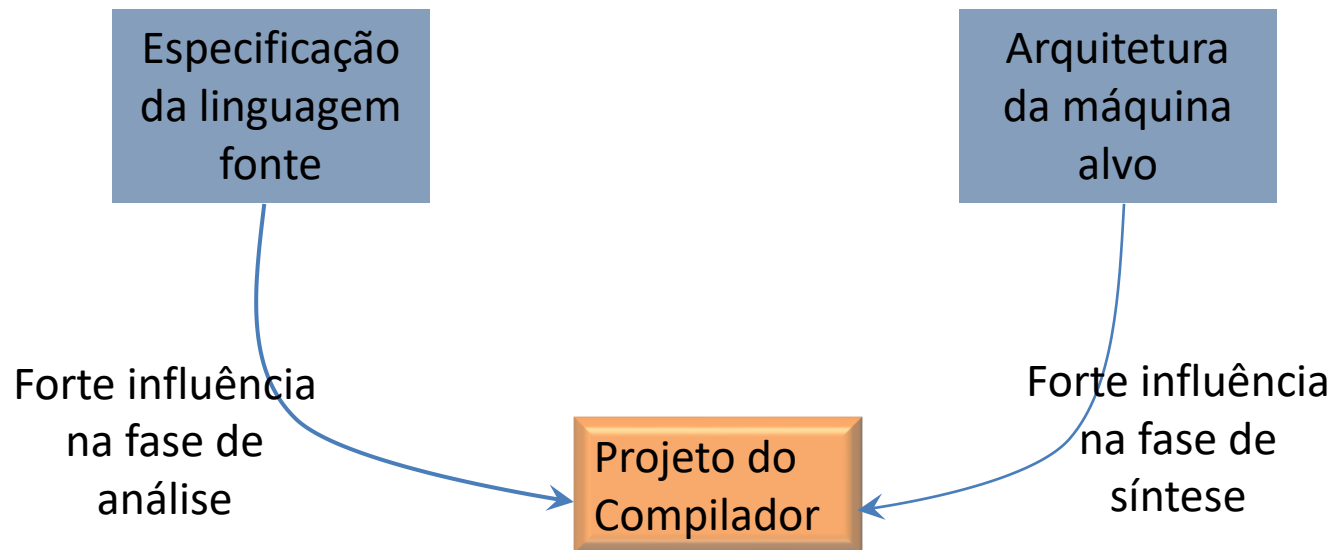
C7 06 0000 0002 (código hexadecimal)

MOV x , 2 (código de montagem)

x = 2 (código em linguagem de alto nível)

Compiladores: Introdução

- Principais fatores que influenciam o projeto de um compilador



Compiladores: Introdução

- Histórico
 - Primeiros compiladores
 - 1952: primeiro compilador (escrito por Grace Hopper), para a linguagem *A-0 System* (rodava no *UNIVAC I*)
 - 1957: primeiro compilador completo (projeto liderado por John W. Backus, na IBM), para a linguagem *FORTRAN*
 - 1960: surgimento de compiladores para múltiplas plataformas, inicialmente para a linguagem *COBOL*
 - 1962: primeiro *self-hosting compiler* (escrito com a própria linguagem que o compilador traduz), para a linguagem *LISP*
 - Da década de 70 em diante tornou-se comum implementar compiladores com a própria linguagem que ele compila -> Problema de *bootstrapping*

Compiladores: Introdução

- Programas Relacionados a Compiladores
 - Interpretador
 - Interpreta o código-fonte e executa-o imediatamente, não gera código-objeto
 - Montador (*assembler*)
 - Traduz para a linguagem de montagem (*assembly*) de um processador particular
 - Organizador (*linker*)
 - Coleta o código compilado separadamente e coloca tudo em um arquivo executável

Compiladores: Introdução

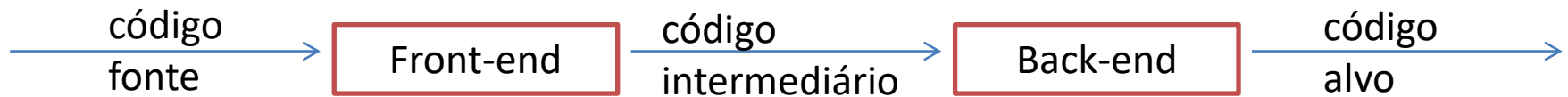
- Programas Relacionados a Compiladores
 - Carregador (*loader*)
 - Carrega o executável na memória, resolve os endereços relocáveis relativos a um dado endereço base ou inicial, torna o código executável mais flexível
 - Pré-Processador
 - Ativado pelo compilador antes do início da tradução, pode apagar comentários, incluir outros arquivos e executar substituições de macros
 - Editor
 - Oferece infraestrutura para escrever o programa fonte, gerando o arquivo a ser compilado, pode ser orientado pela estrutura ou formato da linguagem

Compiladores: Introdução

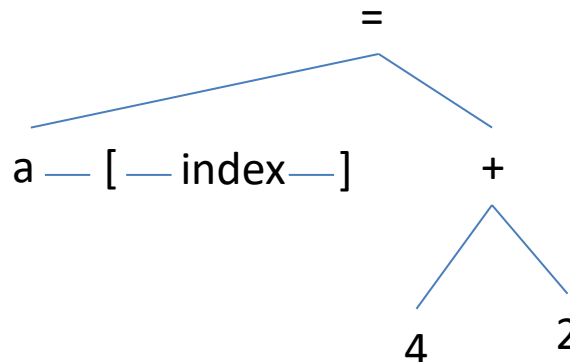
- Programas Relacionados a Compiladores
 - Depurador
 - Utilizado para determinar erros de execução em um programa compilado, costuma ser utilizado de forma integrada em um IDE (*Integrated Development Environment*)
 - Gerador de Perfil
 - Coleta estatísticas sobre o comportamento de um programa objeto durante sua execução

Compiladores: Introdução

- Visão geral do processo de tradução



$a[\text{index}] = 4 + 2$

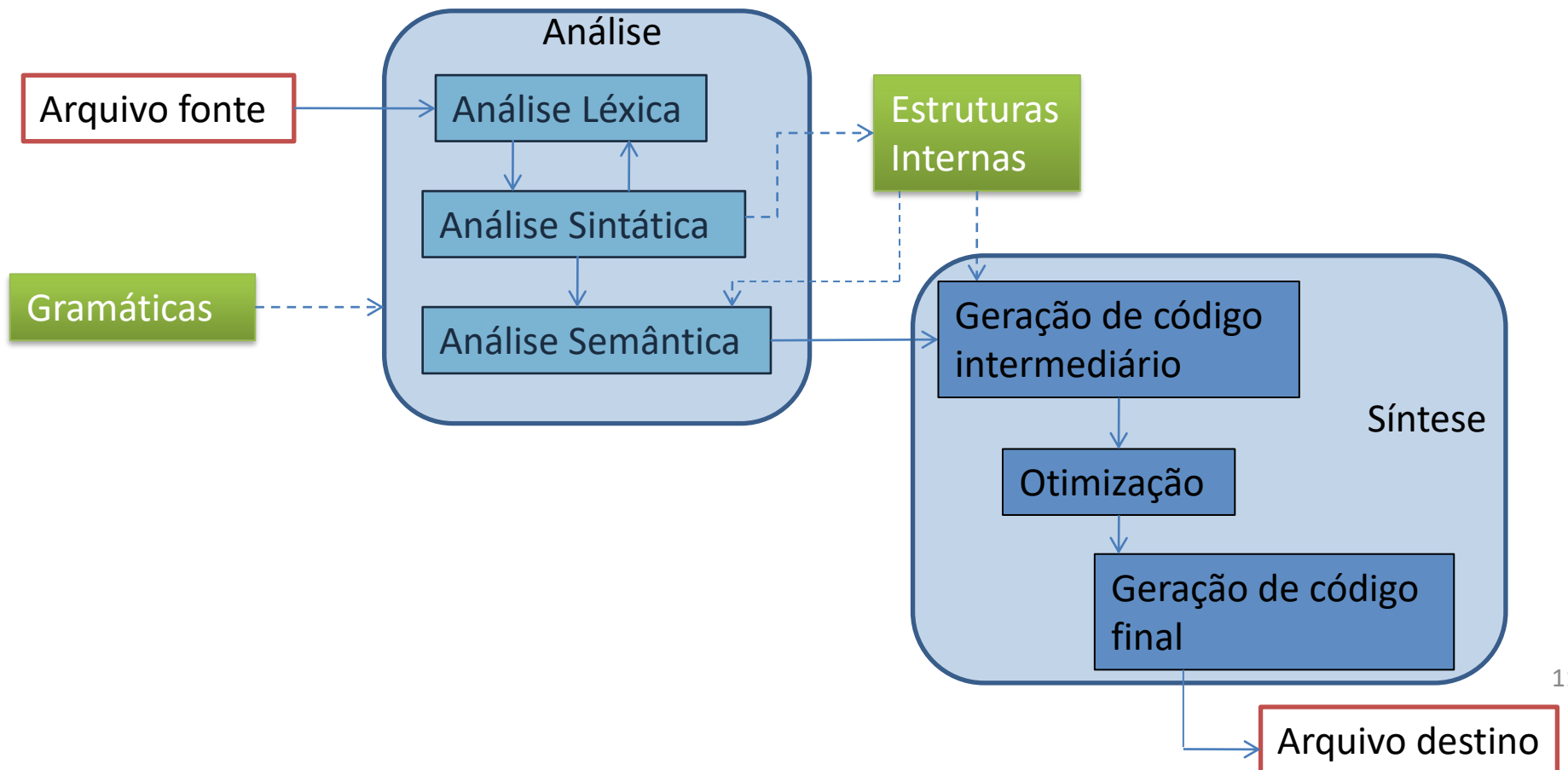


OU
 $t = 4 + 2$
 $a[\text{index}] = t$

```
MOV R0, index
MUL R0, 2
MOV R1, &a
ADD R1, R0
MOV *R1, 6
```

Compiladores: Introdução

- Visão geral do processo de tradução



Compiladores: Introdução

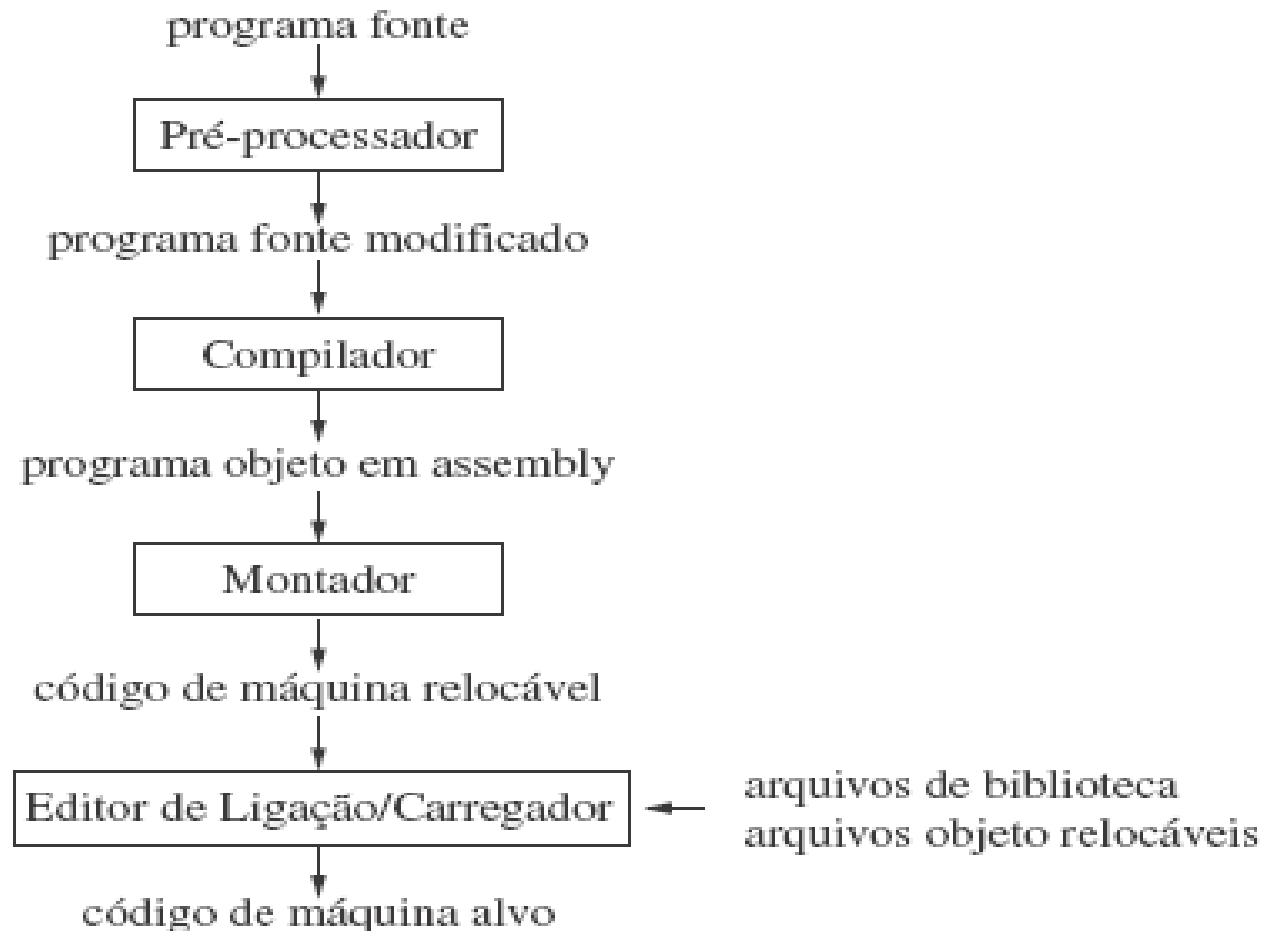


FIGURA 1.5 Um sistema de processamento de linguagem.

Compiladores: Introdução

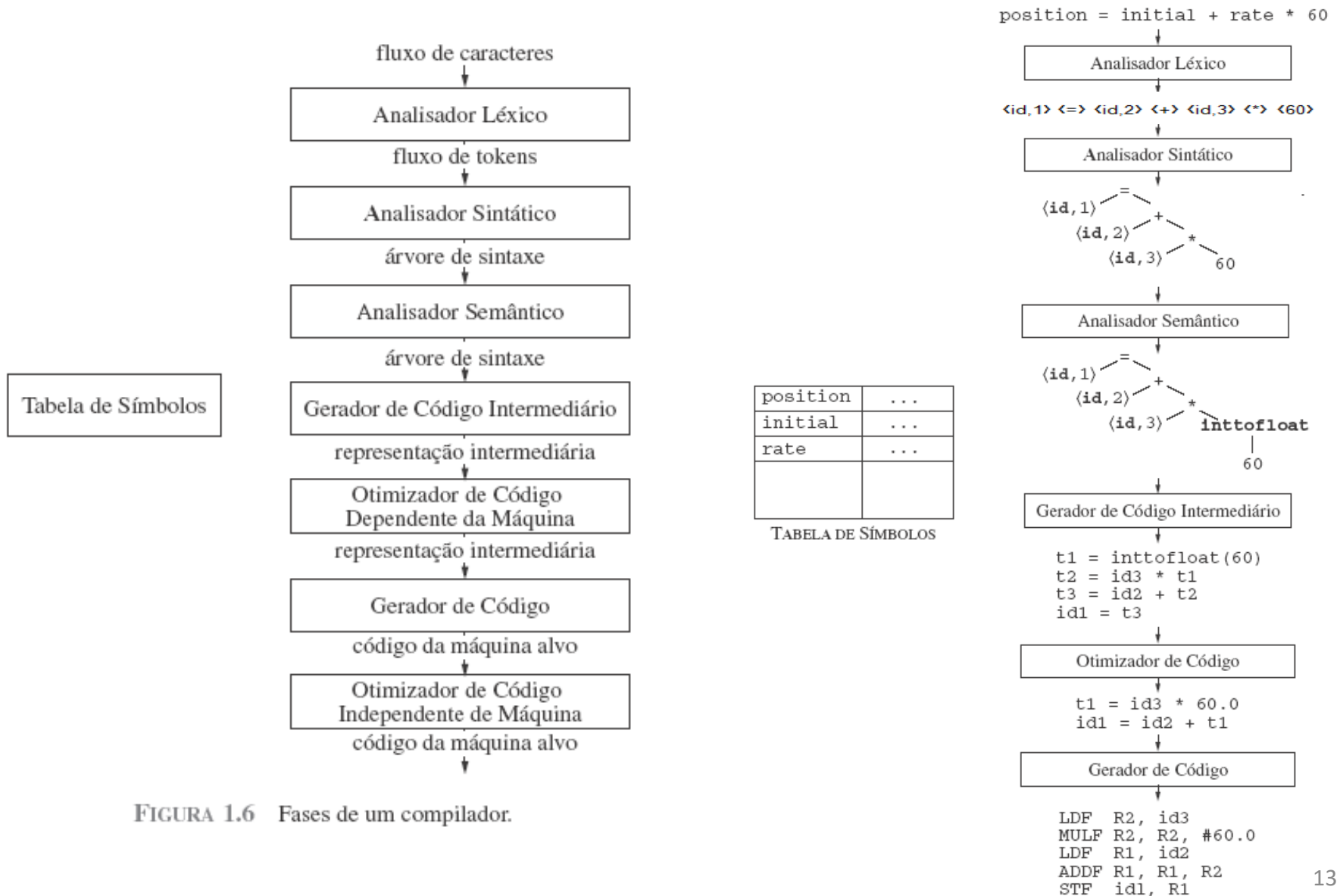


FIGURA 1.6 Fases de um compilador.

Compiladores: Introdução

- O processo de tradução
 - Análise Léxica (analizador léxico – sistema de varredura)
 - Sequências de caracteres são organizadas como unidades significativas, chamadas lexemas

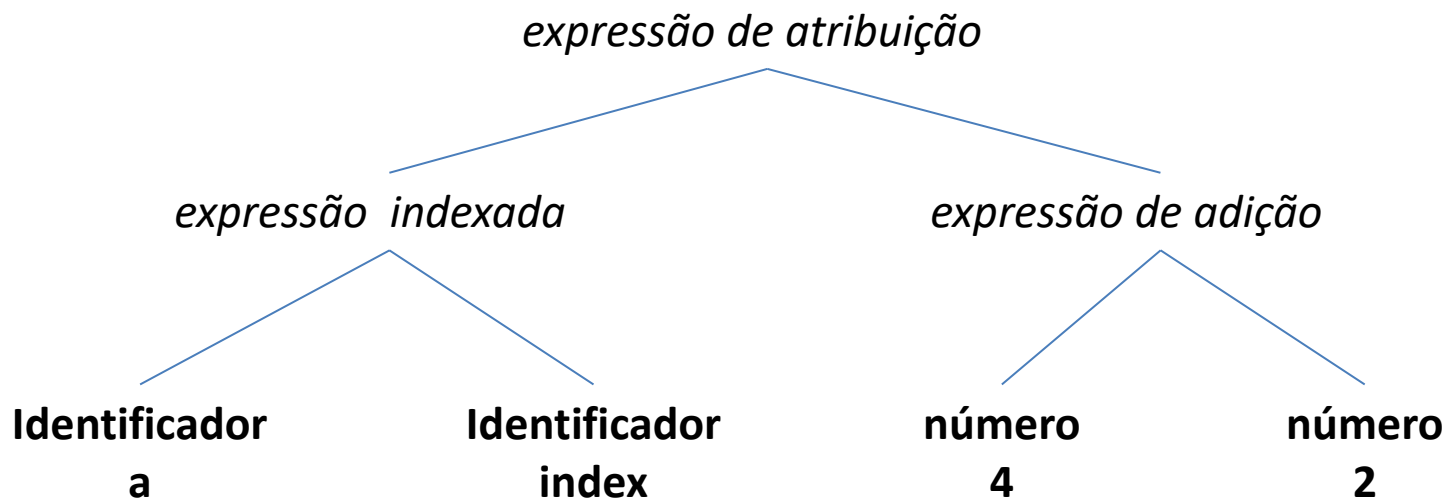
Exemplo: $a[\text{index}] = 4 + 2$

Lexemas	<i>Tokens</i>
a	Identificador
[Colchete à esquerda
index	Identificador
]	Colchete à direita
=	Atribuição
4	Número
+	Adição
2	Número

Além de reconhecer os *tokens*, o analisador léxico pode inserir identificadores na tabela de símbolos

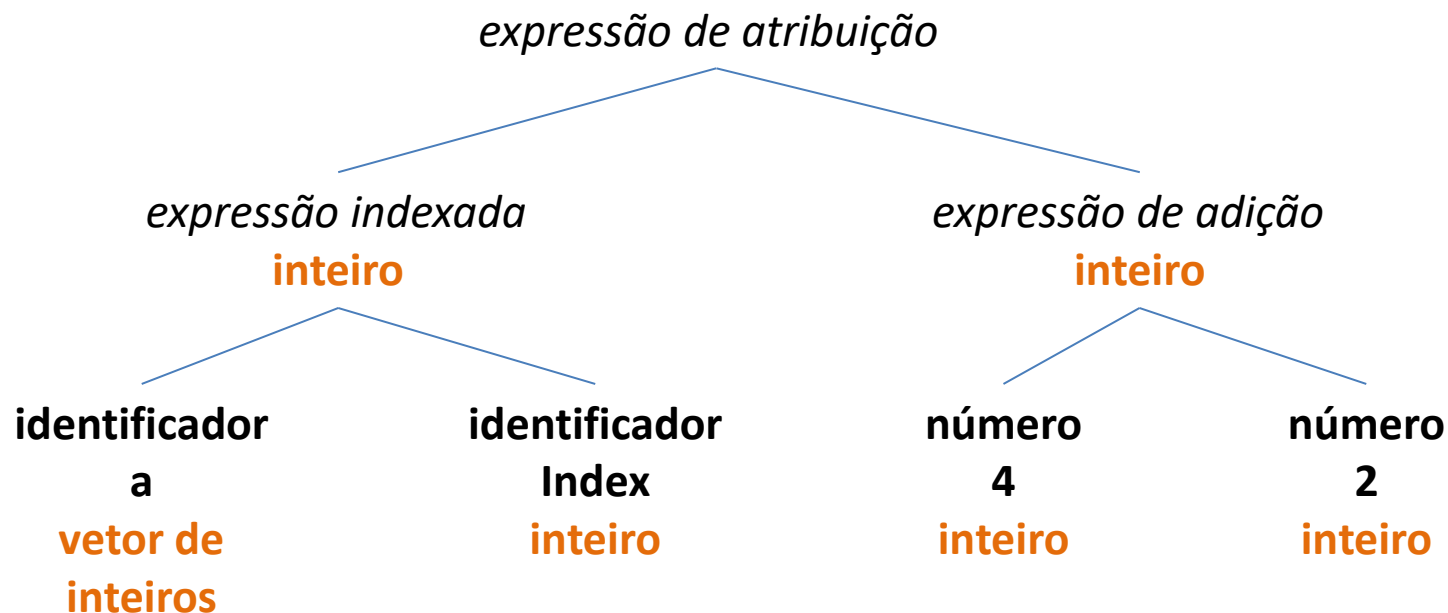
Compiladores: Introdução

- O processo de tradução
 - Análise Sintática (*analisador sintático – Parser*)
 - Determina os elementos estruturais do programa e seus relacionamentos
 - Os resultados da análise sintática geralmente são representados como uma *árvore de análise sintática* ou uma *árvore sintática*



Compiladores: Introdução

- O processo de tradução
 - Análise Semântica (**analisador semântico**)
 - A semântica de um programa é seu significado, contrastando com sintaxe ou estrutura
 - O analisador semântico faz a verificação de tipos e declarações (semântica estática), e frequentemente adiciona novas informações (atributos) na árvore sintática



Compiladores: Introdução

- O processo de tradução
 - A fase de análise

Análise léxica	Verifica se a palavra está bem formada
Análise sintática	Verifica se a sentença está bem formada
Análise semântica	Verifica se o texto (análise de tipos) está coerente

Compiladores: Introdução

- O processo de tradução
 - Gerador de Código Intermediário
 - Gera uma representação intermediária linearizada, próxima do código de montagem
 - Essa representação intermediária deve ser:
 - Facilmente produzida
 - Facilmente traduzida para a máquina alvo
 - Uma representação intermediária muito utilizada é o **código de três endereços**

Exemplo: $a[\text{index}] = 4 + 2$



$t = 4 + 2$

$a[\text{index}] = t$

Compiladores: Introdução

- O processo de tradução
 - Otimizador de Código Intermediário
 - Transforma o código intermediário com o objetivo de produzir um código objeto melhor
 - Código objeto melhor pode significar: um código mais rápido, menor ou que consuma menos energia

Exemplo: o código de três endereços abaixo

$t = 4 + 2$

$a[index] = t$

pode ser otimizado para

$t = 6$ (empacotamento constante)

$a[index] = t$

e depois para

$a[index] = 6$

Compiladores: Introdução

- O processo de tradução
 - Gerador de Código Alvo
 - A partir do código intermediário otimizado, gera o código para a máquina alvo
 - Nessa fase as propriedades da máquina alvo se tornam o fator principal
 - Conjunto de instruções
 - Modos de representação de dados
 - Modos de endereçamento
 - Conjunto de registradores

Exemplo: $a[\text{index}] = 6$

Linguagem de
montagem
hipotética



MOV R0, index	;; move valor de index para R0
MUL R0, 2	;; dobra o valor em R0 (inteiro ocupando 2 bytes)
MOV R1, &a	;; move endereço de <i>a</i> para R1
ADD R1, R0	;; adiciona R0 a R1
MOV *R1, 6	;; move o valor 6 para o endereço apontado em R1

Compiladores: Introdução

- O processo de tradução
 - Otimizador de Código Alvo
 - O compilador tenta melhorar o código alvo gerado
 - Melhorias mais comuns:
 - Escolher outros modos de endereçamento (melhora desempenho)
 - Substituições de instruções lentas por outras mais rápidas (eliminação de operações redundantes ou desnecessárias)

Exemplo: $a[\text{index}] = 6$

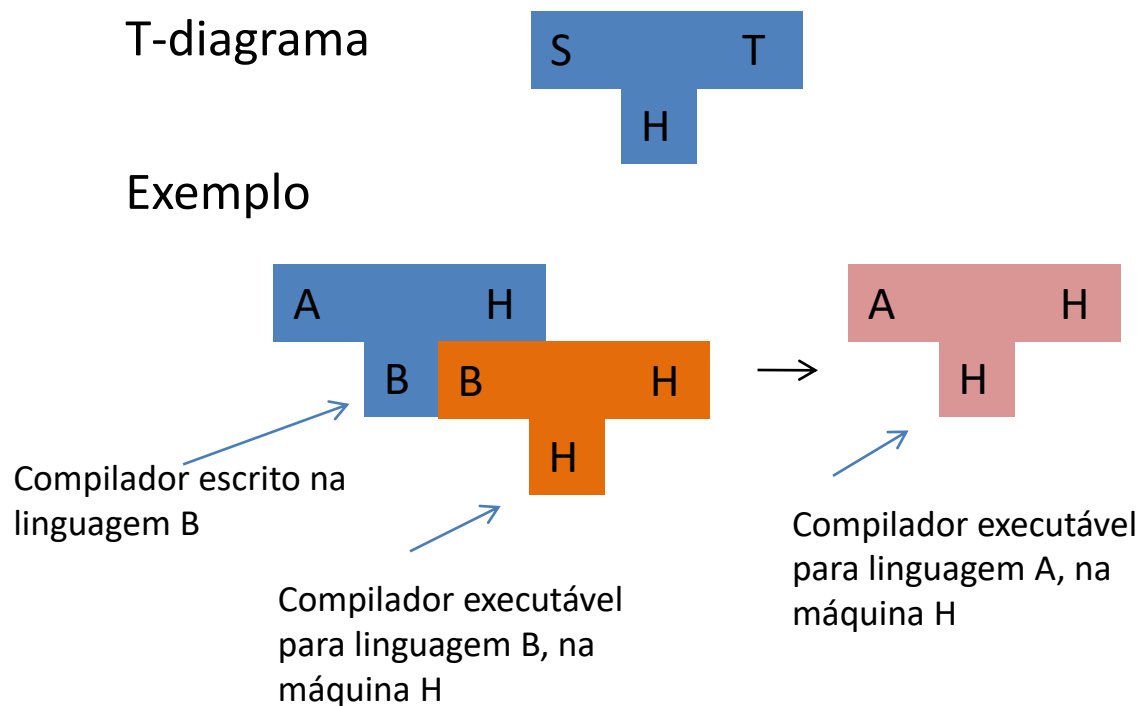
MOV R0, index	MOV R0, index	;; move valor de index para R0
MUL R0, 2	SHL R0	;; instrução de deslocamento (dobra valor em R0)
MOV R1, &a	MOV &a[R0], 6	;; endereçamento indexado
ADD R1, R0		
MOV *R1, 6		

Compiladores: Introdução

- Passadas
 - É comum um compilador processar um programa fonte diversas vezes
 - Essas repetições são chamadas de **passadas**
 - A passada inicial
 - constrói a árvore sintática ou o código intermediário
 - Demais passadas
 - consistem em acrescentar informações, alterando a estrutura ou produzindo uma representação diferente

Compiladores: Introdução

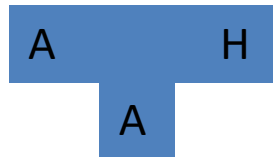
- Compilador do Compilador
 - Como criamos o compilador do compilador ?
 - Abordagem 1: escrever o compilador em linguagem diferente da qual ele deve compilar (para qual já exista um compilador executável)



Compiladores: Introdução

- Compilador do Compilador
 - Como criamos o compilador do compilador ?
 - Abordagem 2: escrever o compilador na mesma linguagem que ele deve compilar (para qual não exista um compilador executável)

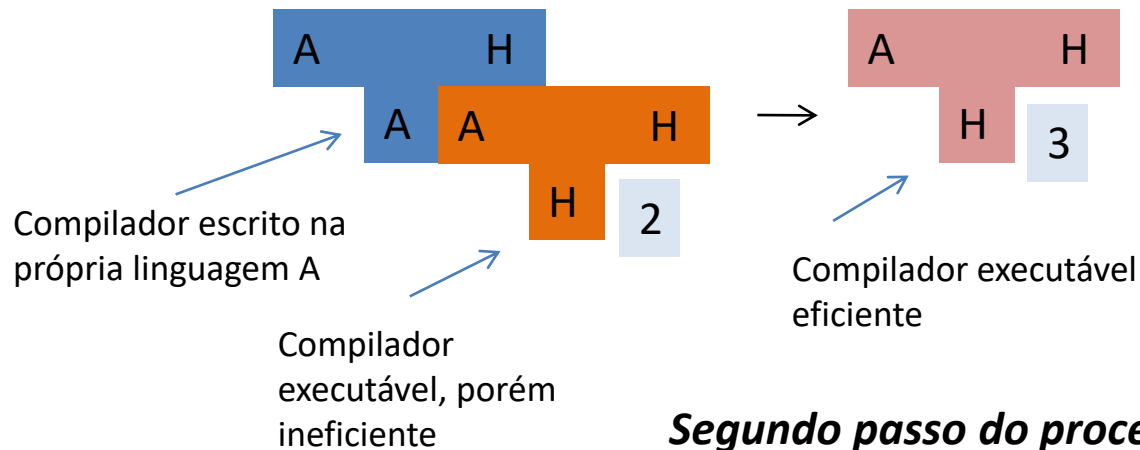
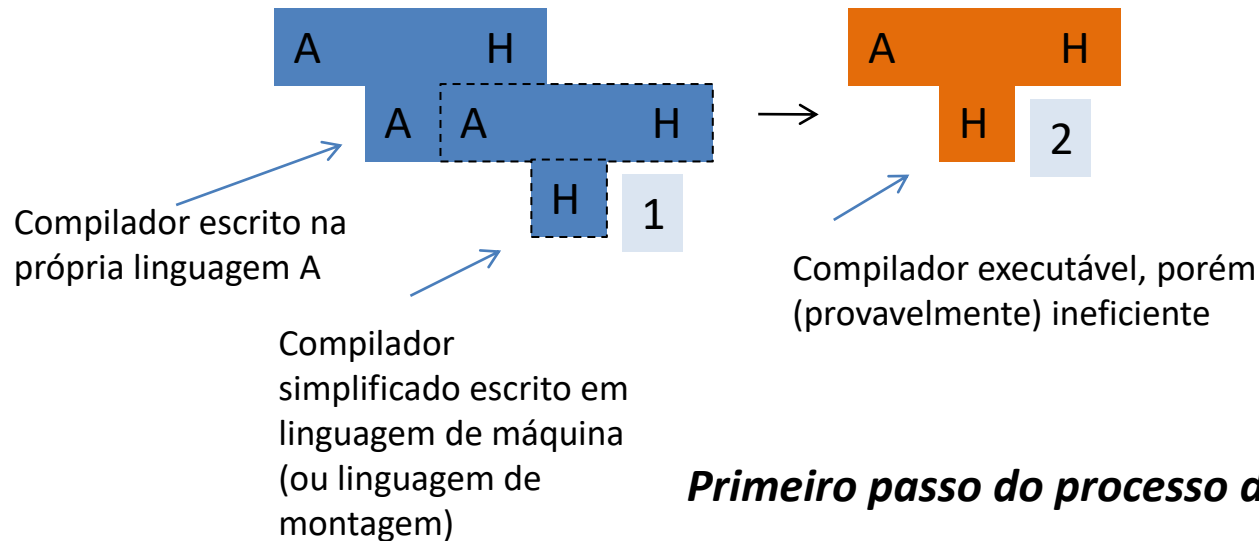
Exemplo



- Processo de **partida rápida** (*bootstrapping*)

Compiladores: Introdução

- Partida Rápida (*Bootstrapping*)



Compiladores: Introdução

- Partida Rápida (*Bootstrapping*)
 - Vantagens:
 - Testar a capacidade da linguagem fonte do compilador (linguagem nova criada)
 - Facilitar a transposição do compilador para outro computador hospedeiro, requerendo apenas reescrever o módulo de síntese do código-fonte do compilador

Compiladores: Introdução

- Bibliografia consultada

LOUDEN, K. C. **Compiladores: princípios e práticas.**
São Paulo: Pioneira Thompson Learning, 2004
(Cap. 1)

AHO, A. V.; LAM, M. S.; SETHI, R. e ULLMAN, J. D.
Compiladores: princípios, técnicas e ferramentas.
2ª edição – São Paulo: Pearson Addison-Wesley,
2008 (Cap. 1)