Compiladores

Introdução à Compiladores

Cristiano Lehrer, M.Sc.

ybadoo

Introdução (1/2)

- O meio mais eficaz de comunicação entre pessoas é a **linguagem** (língua ou idioma).
- Na programação de computadores, uma **linguagem de programação** serve como meio de comunicação entre o indivíduo que deseja resolver um determinado problema e o computador escolhido para ajudá-lo na solução:
  - Ligação entre o pensamento humano e a precisão requerida para o processamento pela máquina.
- O desenvolvimento de um programa torna-se mais fácil se a linguagem de programação em uso estiver próxima ao problema a ser resolvido (linguagem de **alto nível**):
  - Incluir construções que refletem a terminologia e/ou os elementos usados na descrição do problema.

ybadoo

Introdução (2/2)

- Os computadores digitais aceitam e entendem somente sua própria linguagem de máquina, a qual consiste tipicamente de sequências de zeros e uns (linguagem de **baixo nível**).
- Para que se tornem operacionais, os programas escritos em linguagens de alto nível devem ser traduzidos para **linguagem de máquina**:
  - Conversão realizada através de sistemas especializados - **compiladores** ou **interpretadores** - que aceitam como entrada um representação textual da solução de um problema, expresso em uma **linguagem fonte**, e produzem uma representação do mesmo algoritmo expresso em outra linguagem, dita **linguagem objeto**.

ybadoo

Evolução das Linguagens de Programação

- Linguagens de máquina:
  - Programas em notação binária.
- Linguagens simbólicas (Assembly):
  - Linguagens simbólicas ou de montagem.
- Linguagens orientadas ao usuário:
  - Pascal, Fortran, Algol, Ada, C, C++, Java.
- Linguagens orientadas a aplicação:
  - Excel, SQL, Lotus 1-2-3, Visicalc.
- Linguagens de conhecimento:
  - Prolog.

ybadoo

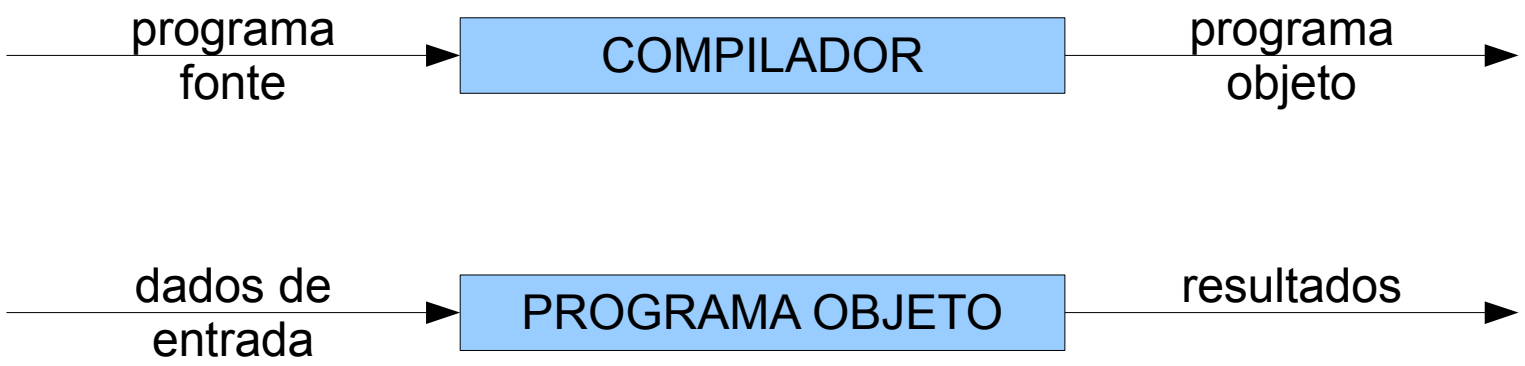
Tradutores de Linguagens de Programação (1/4)

- **Tradutor** é um sistema que aceita como entrada um programa escrito em uma linguagem de programação (**linguagem fonte**) e produz como resultado um programa equivalente em outra linguagem (**linguagem objeto**).
- Montadores (*assemblers*):
  - Tradutores que mapeiam instruções em **linguagem simbólica** (*assembly*) para instruções de **linguagem de máquina**, geralmente, numa relação de uma-para-uma, isto é, uma instrução de linguagem simbólica para uma instrução de máquina.
- Macro-assemblers:
  - Tradutores que mapeiam instruções em **linguagem simbólica** para **linguagem de máquina**, geralmente, numa relação de uma-para-várias.

ybadoo

Tradutores de Linguagens de Programação (2/4)

- Compiladores:
  - Tradutores que mapeiam programas escritos em **linguagens de alto nível** para programas equivalentes em **linguagem simbólica** ou **linguagem de máquina**.



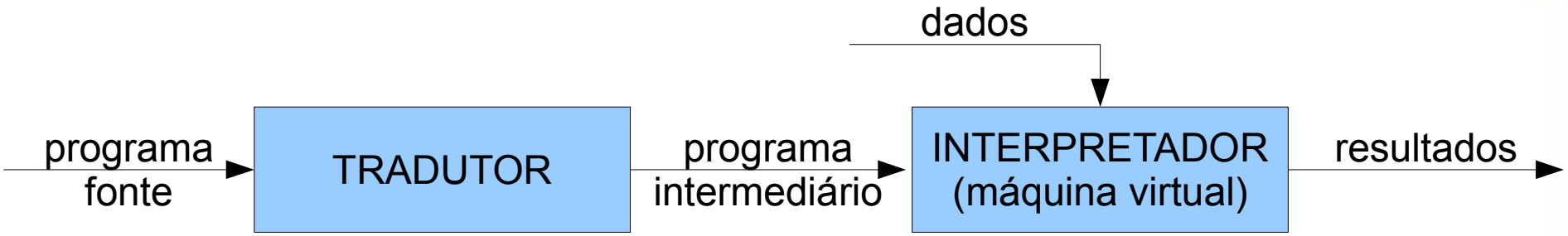
Tradutores de Linguagens de Programação (3/4)

Pré-compiladores, pré-processadores ou filtros:

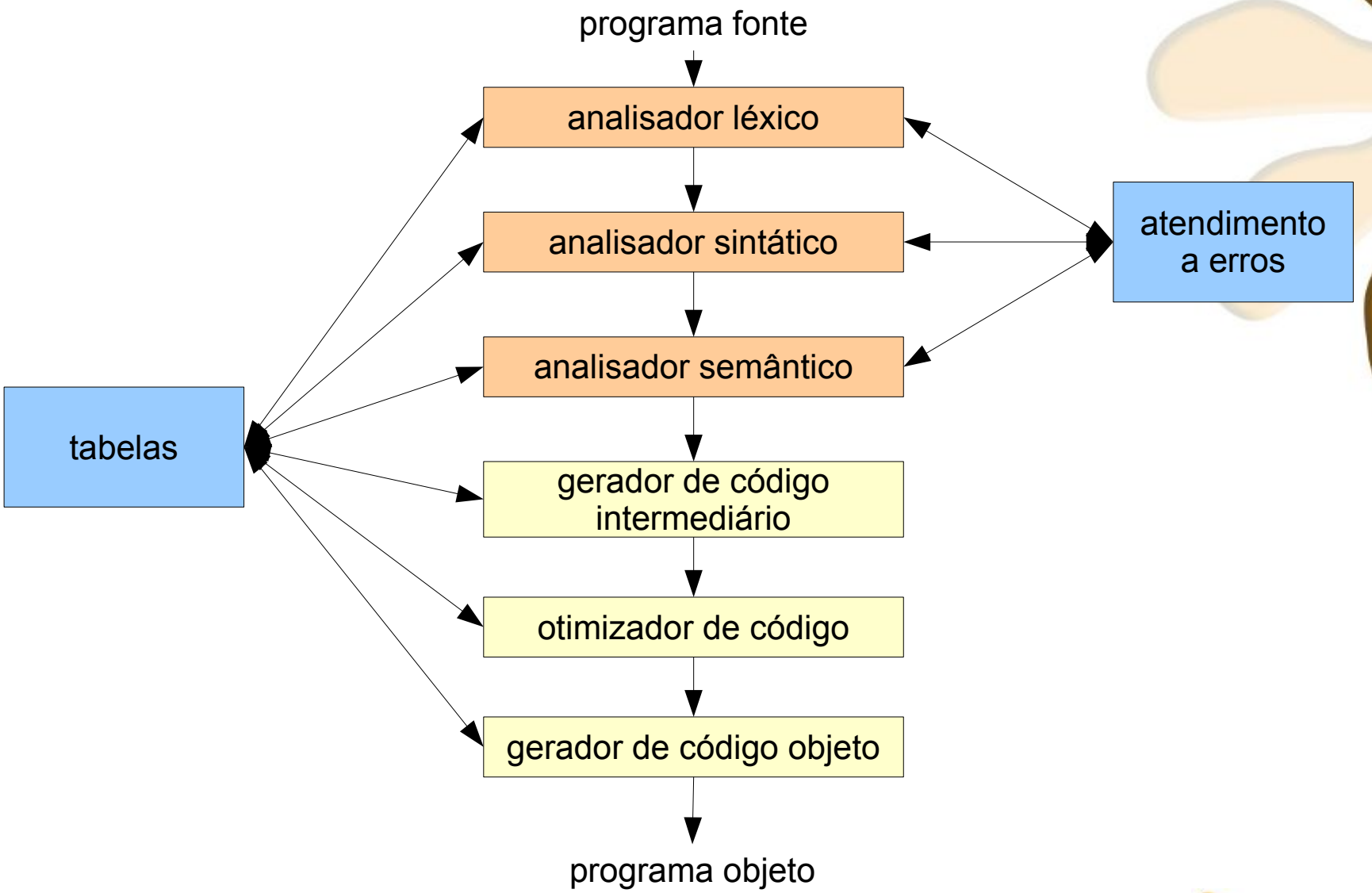
- Tradutores que mapeiam instruções escritas numa **linguagem de alto nível estendida** para instruções da **linguagem de programação original**, ou seja, são tradutores que efetuam conversões entre duas linguagens de alto nível.

Tradutores de Linguagens de Programação (4/4)

- Interpretadores:
  - Tradutores que aceitam como entrada o código intermediário de um programa anteriormente traduzido e produzem o **efeito de execução** do algoritmo original sem, porém, mapeá-lo para linguagem de máquina.



Estrutura de um Tradutor (1/7)



ybadoo

Estrutura de um Tradutor (2/7)

- Análise Léxica:
  - Identificar sequências de caracteres que constituem unidades léxicas (*tokens*).
  - O analisador léxico lê, caractere a caractere, o texto fonte, verificando se os caracteres lidos pertencem ao alfabeto da linguagem, identificando *tokens*, e desprezando comentários e brancos desnecessários.

```
while I < 100 do I := J + I;
```

```
[while, ] [id, 7] [<, ] [cte, 13] [do, ] [id, 7] [:=, ] [id, 12] [+, ] [id, 7] [;, ]
```


ybadoo

Estrutura de um Tradutor (3/7)

- Analises Sintática e Semântica:
  - A fase de análise sintática tem por função verificar se a estrutura gramatical do programa está correta, isto é, se essa estrutura foi formada usando as regras gramaticais da linguagem.
  - A fase de análise semântica tem por função verificar se as estruturas do programa irão fazer sentido durante a execução.

Estrutura de um Tradutor (4/7)

- Geração de código intermediário:
  - Esta fase utiliza a representação interna produzida pela Analisador Sintático e gera como saída uma sequência de código.
  - Vantagens da geração de código intermediário:
    - Possibilita a otimização de código intermediário, de modo a obter-se o código objeto final mais eficiente.
    - Resolve, gradualmente, as dificuldades da passagem de código fonte para código objeto, já que o código fonte pode ser visto como um texto condensado que explode em inúmeras instruções elementares de baixo nível.

Estrutura de um Tradutor (5/7)

- Otimização de código:
  - Esta fase tem por objetivo otimizar o código intermediário em termos de velocidade de execução e espaço em memória.
- Geração de código objeto:
  - Esta fase tem como objetivos:
    - Produção de código objeto.
    - Reserva de memória para constantes e variáveis.
    - Seleção de registradores.
  - É a fase mais difícil, pois requer uma seleção cuidadosa das instruções e dos registradores da máquina alvo a fim de produzir código objeto eficiente.

ybadoo

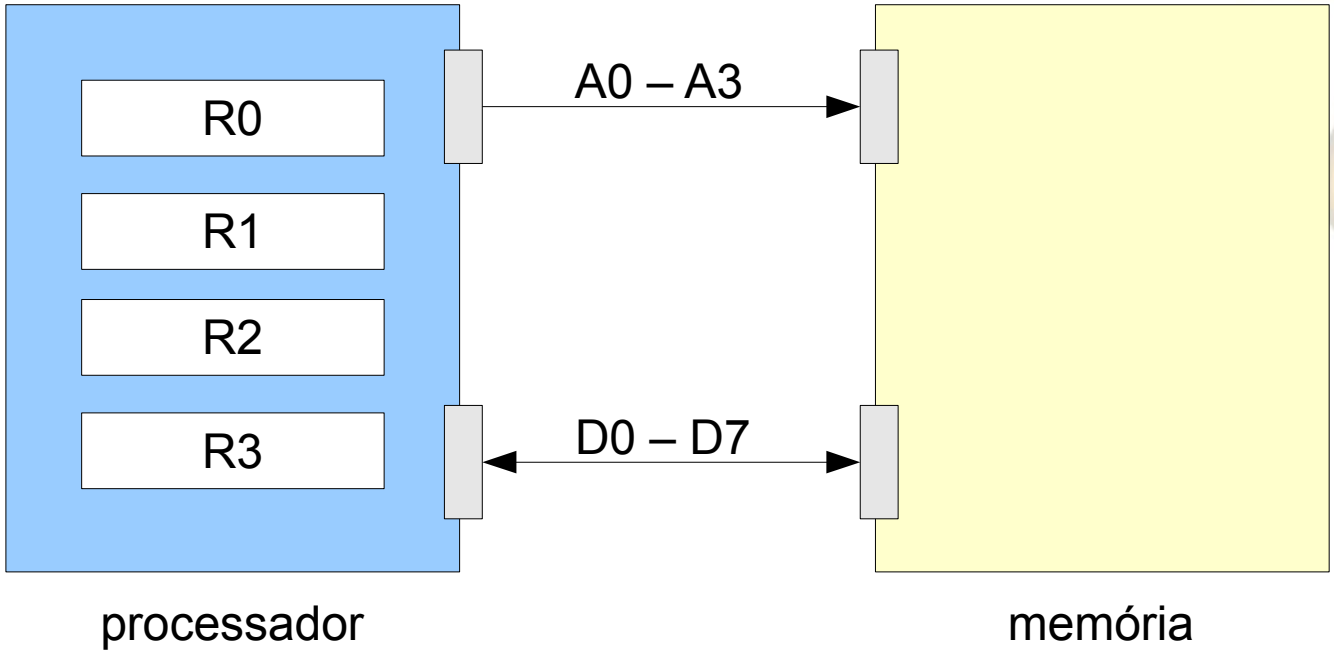
Estrutura de um Tradutor (6/7)

- Gerência de tabelas:
  - Algumas das tabelas usadas são fixas para cada linguagem, por exemplo, a tabela de palavras reservadas, tabelas de delimitadores.
  - Entretanto, a estrutura que possui importância fundamental é aquela que é montada durante a análise do programa fonte, com informações sobre:
    - Declarações de variáveis.
    - Declarações dos procedimentos ou subrotinas.
    - Parâmetros de subrotinas.

Estrutura de um Tradutor (7/7)

- Atendimento a erros:
  - Tem por objetivo tratar os erros que são detectados em todas as fases de análise do programa fonte.
  - Qualquer fase analítica deve prosseguir em sua análise, ainda que erros tenham sido detectados.
  - É fundamental que o tradutor prossiga na tradução, após a detecção de erros, de modo que o texto seja totalmente analisado.

Arquitetura de um Processador Hipotético



ybadoo

Registradores

- Presentes na figura:
  - Quatro registradores de dados, usados como área de armazenamento temporário para os dados envolvidos na execução das operações:
    - R0, R1, R2 e R3
- Ocultos na figura:
  - Registrador de instruções:
    - Armazena a instrução que é executada.
  - Contador de programas:
    - Armazena o endereço da próxima instrução que deve ser executada.
  - Registrador de controle:
    - Armazena o estado associado ao resultado da última instrução.

Barramento

- Processador de oito bits:
  - Dimensão do barramento de dados:
    - D0 a D7
- Memória:
  - Capacidade de endereçamento de memória:
    - 16 posições:
      - A0 a A3.

Instruções (1/2)

- LOAD – 00
  - Para transferir o conteúdo de uma posição de memória para um registrador.
  - Por exemplo, a instrução **LOAD 10, R1** é usada para carregar o conteúdo da posição 10 de memória para o registrador R1.
- STORE – 01
  - Para transferir o conteúdo de um registrador para uma posição de memória.
  - Por exemplo, a instrução **STORE R2, 5** é usada para transferir o conteúdo do registrador R2 para a posição 5 de memória.

ybadoo

Instruções (2/2)

- ADD – 10
  - Para adicionar o conteúdo de dois registradores e armazenar o resultado em outro registrador.
  - Por exemplo, a instrução **ADD R1, R2, R3** soma o conteúdo de R1 e R2 e coloca o resultado em R3.
- BZERO – 11
  - Para mudar o conteúdo do contador de programas para a posição de memória especificada se o conteúdo do registrador indicado for igual a zero.
  - Por exemplo, a instrução **BZERO R3, 7** define que o contador de programas será alterado para o valor 7 somente se o conteúdo de R3 for igual a zero.

ybadoo

Representação Binária

- LOAD 5, R3 – 00010111
  - LOAD – 00
  - 5 – 0101
  - R3 – 11
- BZERO R3, 7 – 11110111
  - BZERO – 11
  - R3 – 11
  - 7 – 0111

Exemplo

Linguagem de Alto Nível

```
c = a + b;
```

Linguagem Simbólica	Linguagem de Máquina
`LOAD 10, R1`	`00101001`
`LOAD 11, R2`	`00101110`
`ADD R1, R2, R0`	`10011000`
`STORE R0, 12`	`01001100`

ybadoo