COMPILADORES

Nota de Aula (2)

Stéfano Schwenck Borges Vale Vita, Prof. Me.

O Que Veremos Hoje?

- Níveis de Linguagens
- Processadores de Linguagens de Programação
- Tradutor x Compilador x Interpretador
- Arquitetura de um compilador
- Analise e Síntese

Níveis de Linguagens

O que é uma Linguagem de Programação?

- Notação formal para expressar algoritmos
- Um método padronizado para expressar instruções para um computador.
- Permite que um programador especifique sobre quais dados o computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias

Niveis de Linguagem de Programação

Linguagem Natural

```
A area de um triângulo
é igual a b*h/2
```

Linguagem de Alto Nivel

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

Linguagem Simbolica

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

Linguagem de Maquina

```
00010010010001010010010
01110110010101101001...
```

Linguagem de Máquina

- Computadores entendem *código de máquina* ou *linguagem de máquina*
  - seqüência de instruções primitivas expressas por uma seqüência de bits
  - interpretada para executar uma determinada operação (primitiva)
    - carregar dados
    - somar registradores
    - desvios condicionais e incondicionais, etc.

Linguagem de Máquina

- Originalmente se escrevia diretamente em linguagem de máquina:
  0000 0001 0011 0010

- Problemas
  - dificuldade em ler, escrever, editar;
  - controle explícito dos endereços de memória para dados e para o próprio programa
- Limite: alguns milhares de instruções

Linguagem de Montagem (Simbólica)

- Notação simbólica para facilitar a escrita, leitura e edição:
  LOAD x
  ADD R1 R2
  JUMPZ h

- Tradução para linguagem de máquina: montagem do programa
- *Assembly language*
- Uso de um programa montador (*assembler*)

- Instruções ainda muito próximas da linguagem de máquina (relação de 1 para 1)

Linguagens de Programação de alto nível

- Maior nível de abstração:

```
let s = (a+b+c)/2
in sqrt (s*(s-a)*(s-b)*(s-c))
```

- Ao invés de:

LOAD R1 a; ADD R1 b; ADD R1 c; DIV R1 #2;
LOAD R2 R1; LOAD R3 R1; SUB R3 a;
MULT R2 R3; LOAD R3 R1; SUB R3 b;
MULT R2 R3; LOAD R3 R1; SUB R3 c;
MULT R2 R3; LOAD R0 R2; CALL sqrt;

Linguagens de alto nível devem suportar os seguintes conceitos

- Uso de ***expressões***, usando notação semelhante à matemática;
- ***tipos de dados*** primitivos e compostos;
- ***estruturas de controle*** como if-then-else, while, for etc.;
- ***declarações* de variáveis**, tipos, funções, procedimentos etc.;
- ***abstração***: o que é feito **X** como é feito;
- ***encapsulamento*** (ou abstração de dados): classes, pacotes, módulos (orientação a objetos).

Processadores de Linguagens

- Programadores precisam de meios para editar, traduzir e interpretar os programas em um computador
  - *Processadores de Linguagens de Programação*

- Sistemas que manipulam programas expressos em alguma linguagem de programação
  - Editores
  - Tradutores
  - Compiladores
  - Interpretadores

Processadores de Linguagens de Programação

- Ferramentas de software (Unix) X processadores integrados (IDEs: ***I****ntegrated* ***D****evelopment* ***E****nvironments*)
  - Exemplo: JDK x Eclipse

Tipos de Processadores de Linguagem

- Tradutor
- Compilador
- Filtro
- Pré-processador
- Montador
- Interpretador
- Interpretador / Compilador

Tradutor

- Recebe um texto expresso em uma ***linguagem fonte*** e gera um texto semanticamente equivalente em uma ***linguagem destino***.
  - Chinês para inglês
  - Java para C
  - Java para x86

Compilador

- Traduz de uma linguagem de alto nível para uma linguagem de baixo nível
  - C para Assembler
  - Java para C

O que é um compilador?

- Um dos módulos básicos do software de um computador
  - Software de aplicação x Software básico

- Função: efetuar, automaticamente, a tradução de textos, redigidos em uma determinada linguagem de programação, para alguma outra forma que viabilize sua execução pelo computador
  - em geral, em linguagem de máquina

Atividades adicionais do compilador

1. Detecção e recuperação de erros
2. Permite a inclusão de comentários no código fonte, facilitando a compreensão.

   As linhas de comentário são reproduzidas no código-objeto?

3. Comandos de controle de compilação

Filtro

- Traduz de uma linguagem de alto nível para outra linguagem de alto nível (muito semelhante à linguagem fonte)
  - C para C++

Pré-Processador

- Traduz entre 2 dialetos de uma mesma linguagem
  - Antes da compilação
- Converte um texto fora de formato para uma forma padronizada
  - Extensões da linguagem

Montador

- Traduz de uma linguagem de montagem para o código de máquina correspondente
  - Assembler para x86

Montador X Compilador

- ***Montador***
  - *1:1 → Uma instrução de máquina para cada instrução em linguagem assembler*

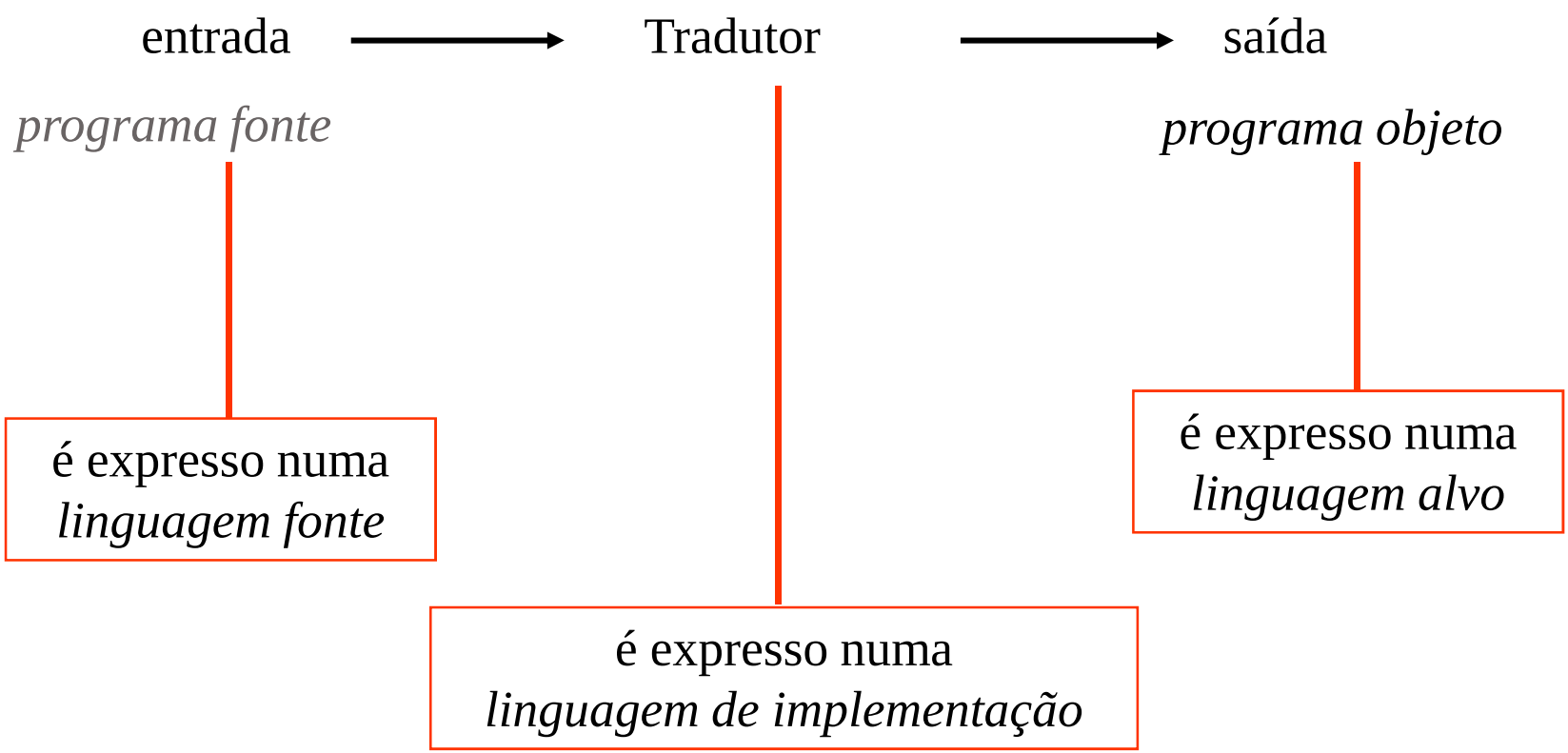
- ***Compilador***
  - *1:M → Cada linha de comando da linguagem de alto-nível produz várias instruções de máquina*

Representação do Processamento de Linguagens

Linguagens envolvidas no processamento de linguagens

- Linguagem fonte
- Linguagem destino
- Linguagem de implementação

Esquema Geral de Tradução

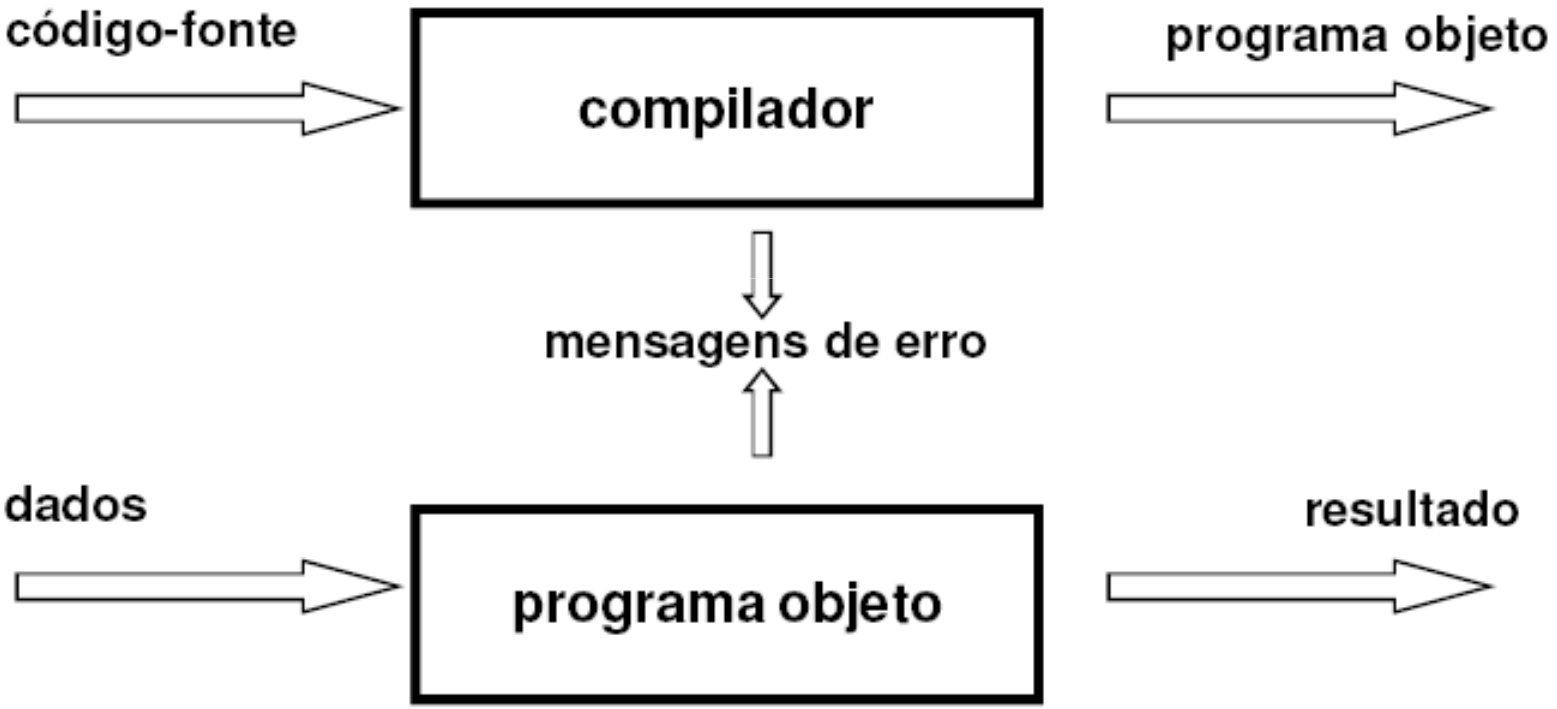


Regras de funcionamento do Tradutor

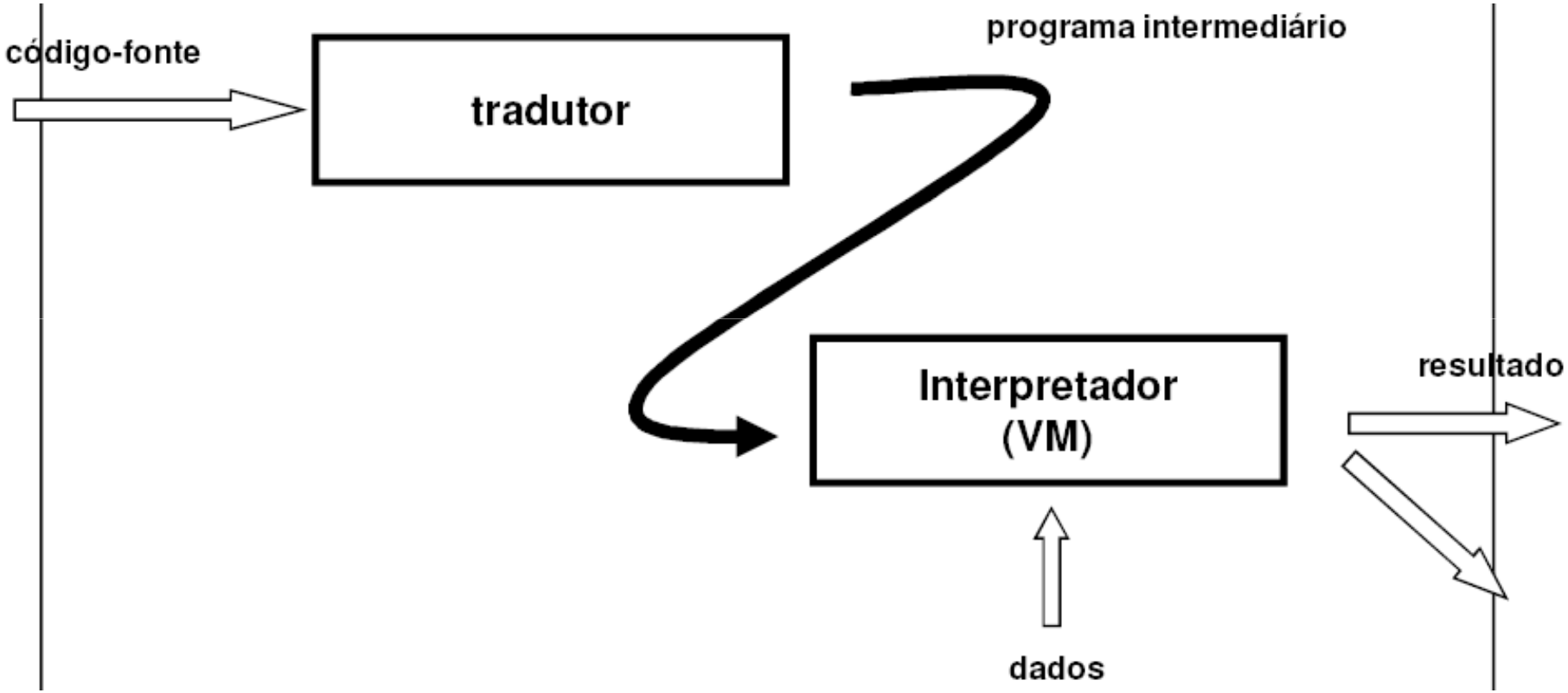
- Um programa tradutor pode rodar em uma maquina M apenas se ele for expresso em codigo da maquina M
- O programa fonte deve ser expresso na linguagem fonte do tradutor (S)
- O programa objeto deve ser expresso na linguagem objeto do tradutor (T)
- O programa objeto é **semanticamente equivalente** ao programa fonte

Compilador x Interpretador

Compilador x Interpretador



Compilador x Interpretador



Interpretador x Compilador

- ***Interpretador***
  - Programa fonte → executa imediatamente
  - Lê, analisa e executa as instruções, ***uma de cada vez***
  - Resultados imediatos, sem a tradução do programa para código objeto antes da execução
  - Execução (de linguagem de alto nível) até 100 vezes mais lenta que a versão compilada

- ***Compiladores***
  - Execução dos programas com máxima performance, em linguagem de máquina
  - Tempo de espera pela compilação

Uso de interpretadores

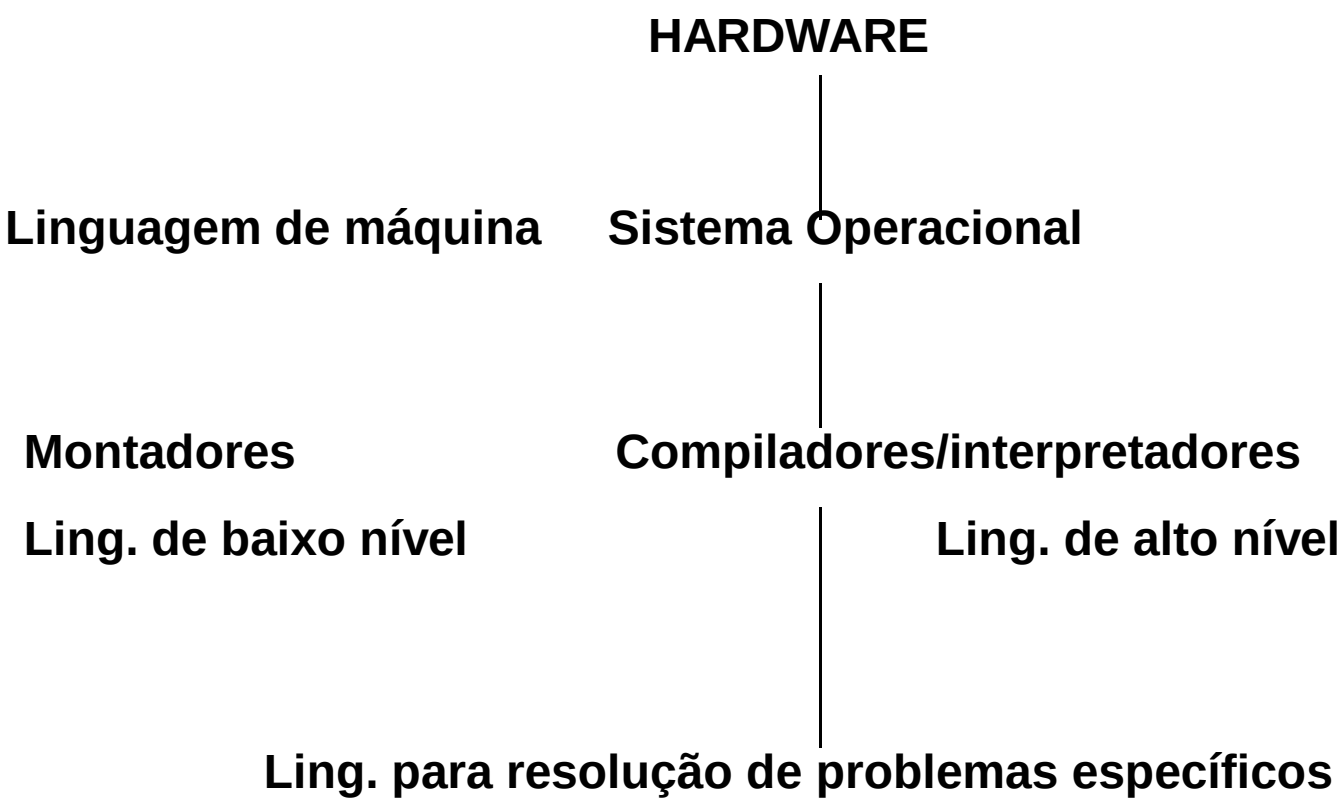
- Programador trabalha em modo interativo
  - Quer ver o resultado de uma instrução antes de entrar na próxima
- Execução de instruções apenas uma vez, ou raramente
- Formato de instruções simples, podendo ser analisada fácil e eficientemente
- Programas descartáveis
  - Em que a velocidade de execução não é tão importante

Exemplos de Interpretadores

- Basic
  - Expressões e atribuições de alto nível, mas estruturas de controle de baixo nível (desvio condicional/incondicional)
- Lisp
  - Estrutura de árvore para código e dados, podendo gerar código em tempo de execução
- Shell do Unix e do DOS
- Interpretador SQL

Arquitetura de um Compilador

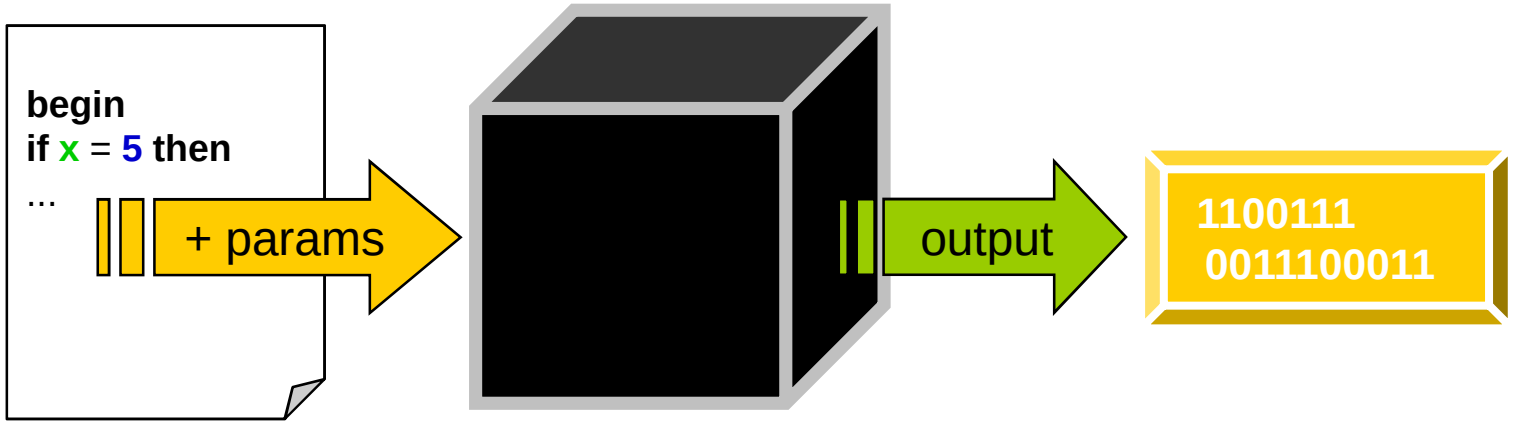
Relacionamento dos compiladores com outros elementos de um sistema de programação



Porque Estudar o Funcionamento dos Compiladores?

- Especificação / Implementação de L.P.
  - Uso geral: novas, extensões e atualizações
  - Específico: Tempo Real, robótica, descrição de HW, SO, BD, Protocolos, Interfaces
- Uso de técnicas / ferramentas em outros sistemas
  - Processamento de texto, linguagens naturais
- Entender melhor as Linguagens de Programação
  - Escolha e uso mais racional/eficiente
- Estudos avançados
  - Pesquisa e pós-graduação

O processo de Compilação

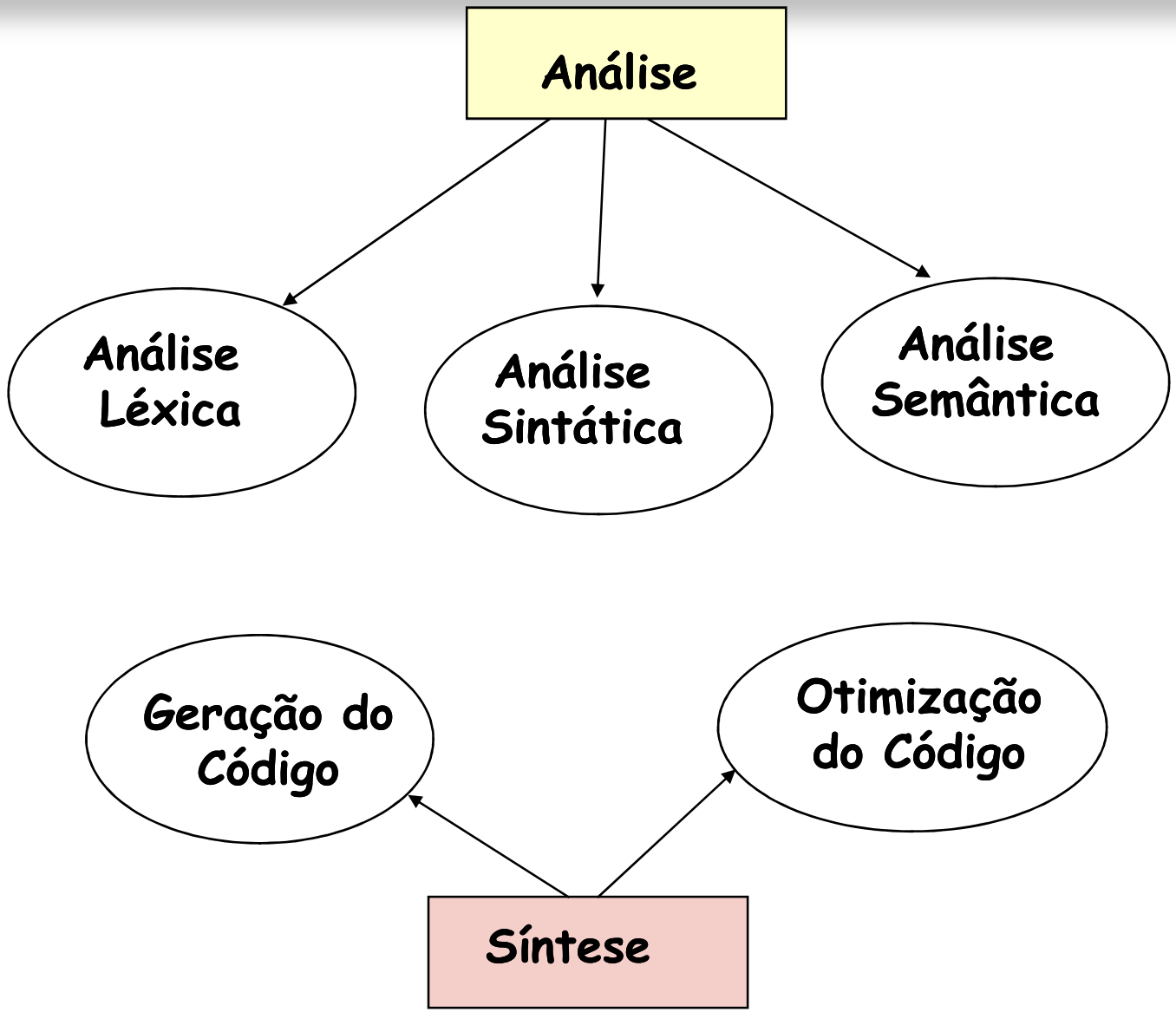


Código Fonte **Compilador** **Programa**

Etapas do processo de compilação

- Processo de compilação divide-se em duas fases:
  - **Análise** : parte o programa fonte em peças constituintes criando uma representação intermediaria do programa.
  - **Síntese** : Constrói o programa alvo desejado (em código de máquina) a partir da representação intermediaria.
- **A parte da síntese é a que requer técnicas mais especializadas.**

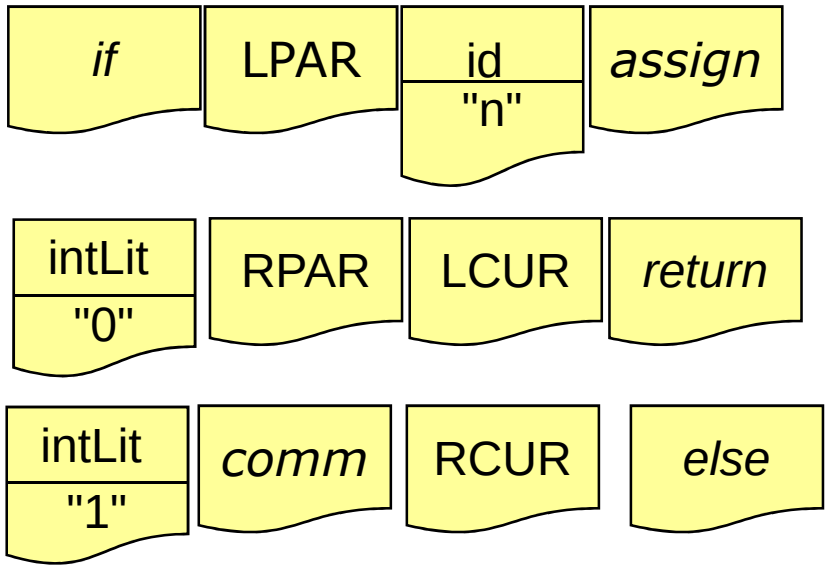
Analise e Sintese



Análise Léxica (*Scanning*)

- Código fonte -> sequência de *tokens*
  - Símbolos como identificadores, literais, operadores, palavras-chave, pontuação etc.

```
if (n == 0) {
    return 1;
} else {
    ...
}
```



Análise Sintática

- Agrupa caracteres ou *Tokens* em uma estrutura hierárquica com algum significado
- Determina se uma dada *cadeia de entrada* pertence ou não à linguagem definida por uma *gramática*

A seguinte construção é válida?

```
int y = 0,k = 0;
int x = y+++k;
```

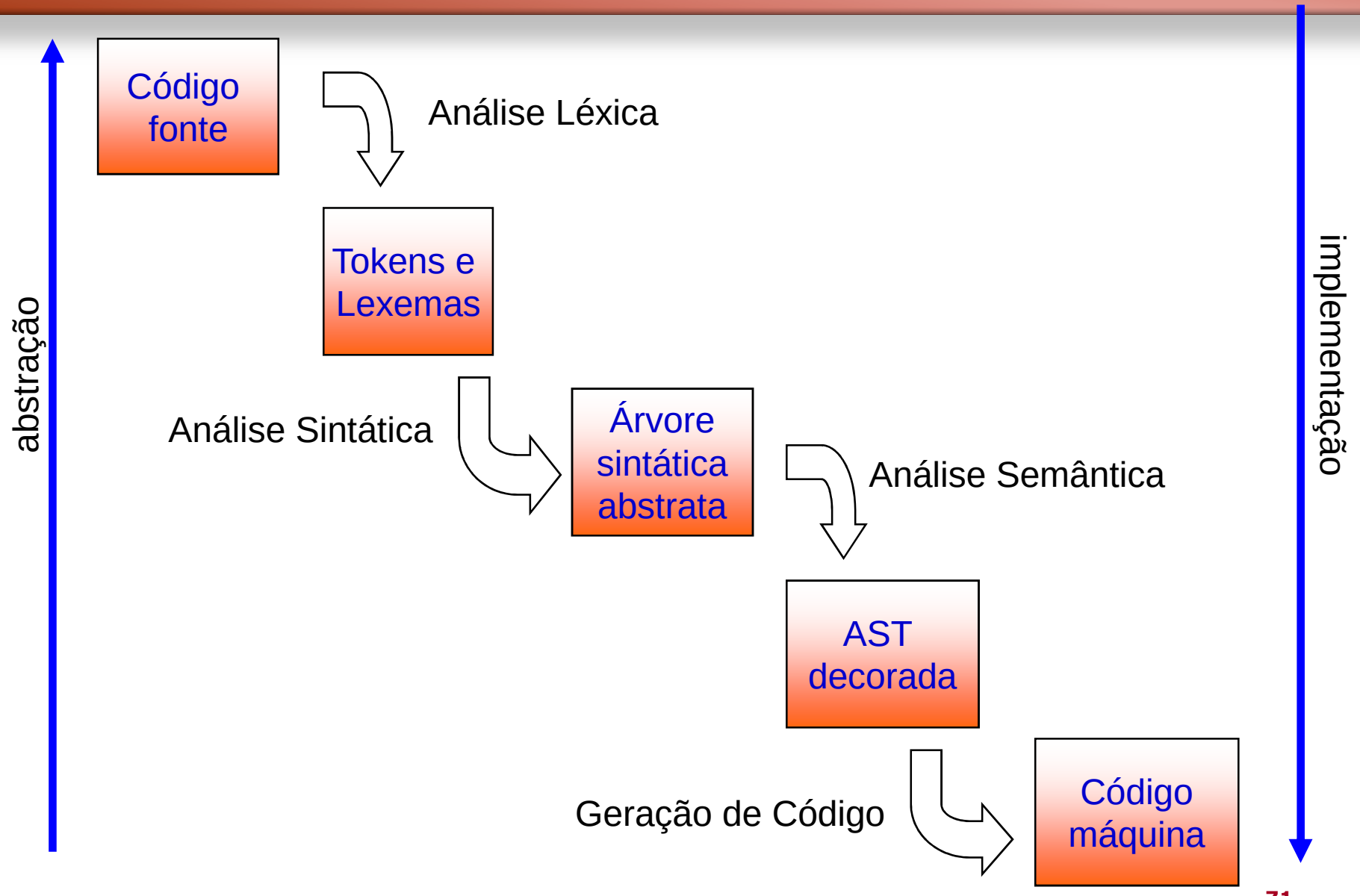
Análise Semântica (Contextual)

- Verifica se o programa está de acordo com as restrições contextuais da linguagem fonte

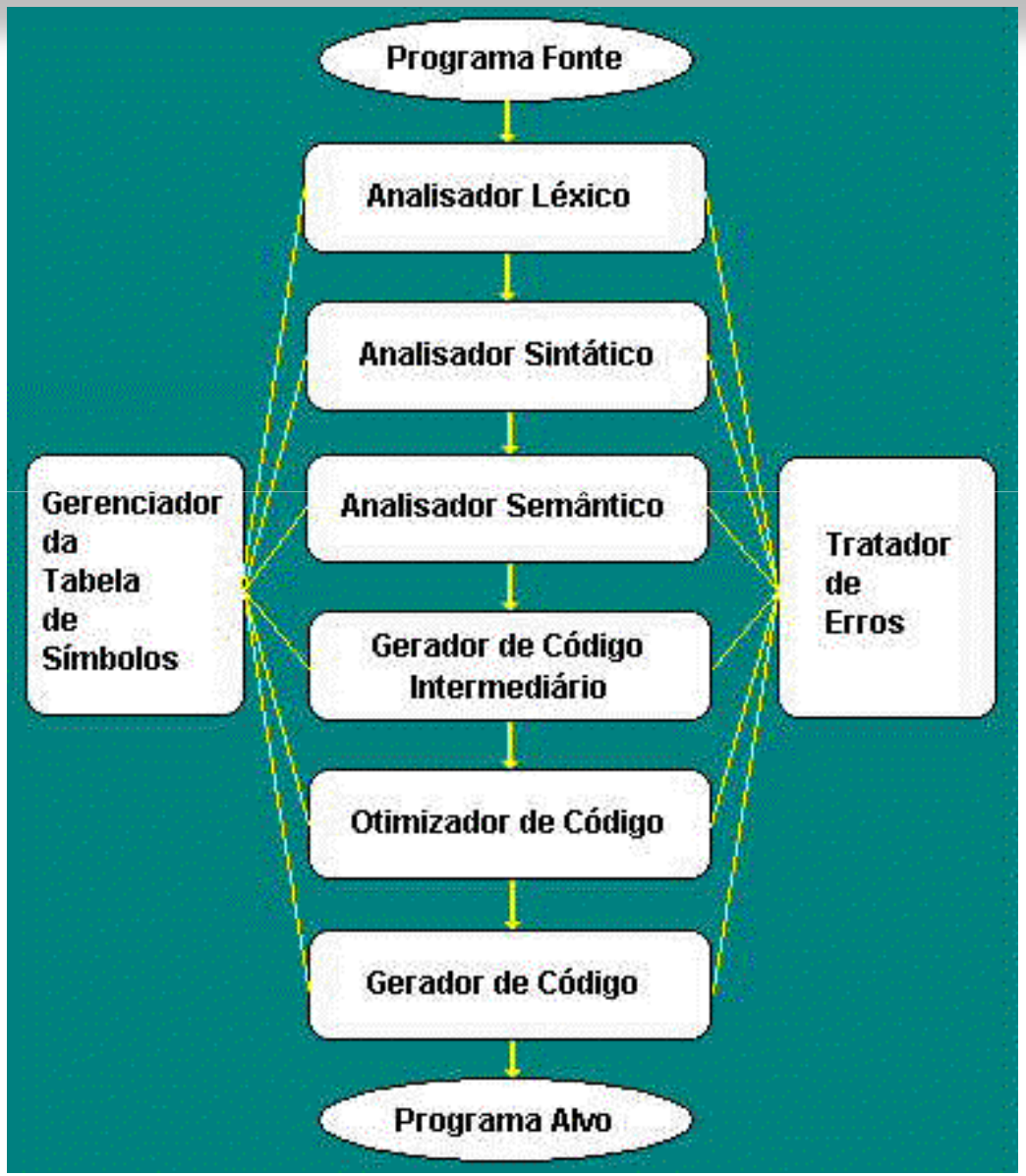
- Em uma linguagem com tipos estáticos e ligação estática:
  - Verifica regras de escopo
  - Verifica regras de tipos

- Duas fases:
  - Identificação
  - Verificação

Fases de Compilação



Fases de um compilador



Atividades para a proxima aula

- Pesquisa e resumo
  - Linguagens compiladas x interpretadas
  - Trazer exemplos para discussão