UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Construção de Compiladores

Alterado por: prof. Rodrigo Pereira
Alterado por: prof. Marlon Oliveira
Elaborado por: Profª Christine Vieira

Compiladores

Nos computadores antigos os compiladores muitas vezes exerciam o seu papel como programas autônomos.

- Interação direta com o programador transformando o programa em formas intermediárias produzidas em meios de armazenamento externo (fitas perfuradas e cartões)

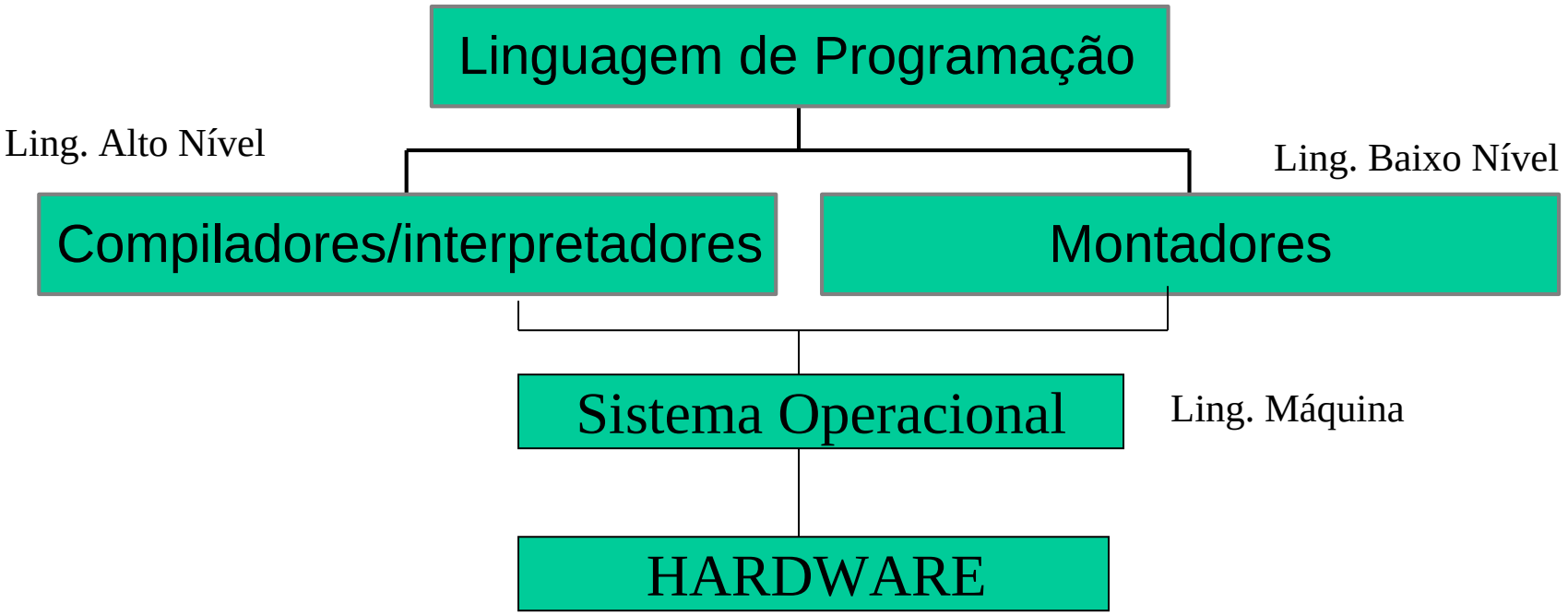
O *Código Objeto*, também se apresentava na forma de fitas e cartões.

- Continha programas objeto em linguagem de máquina realocável, ou em formato fonte de linguagem simbólica de baixo nível.

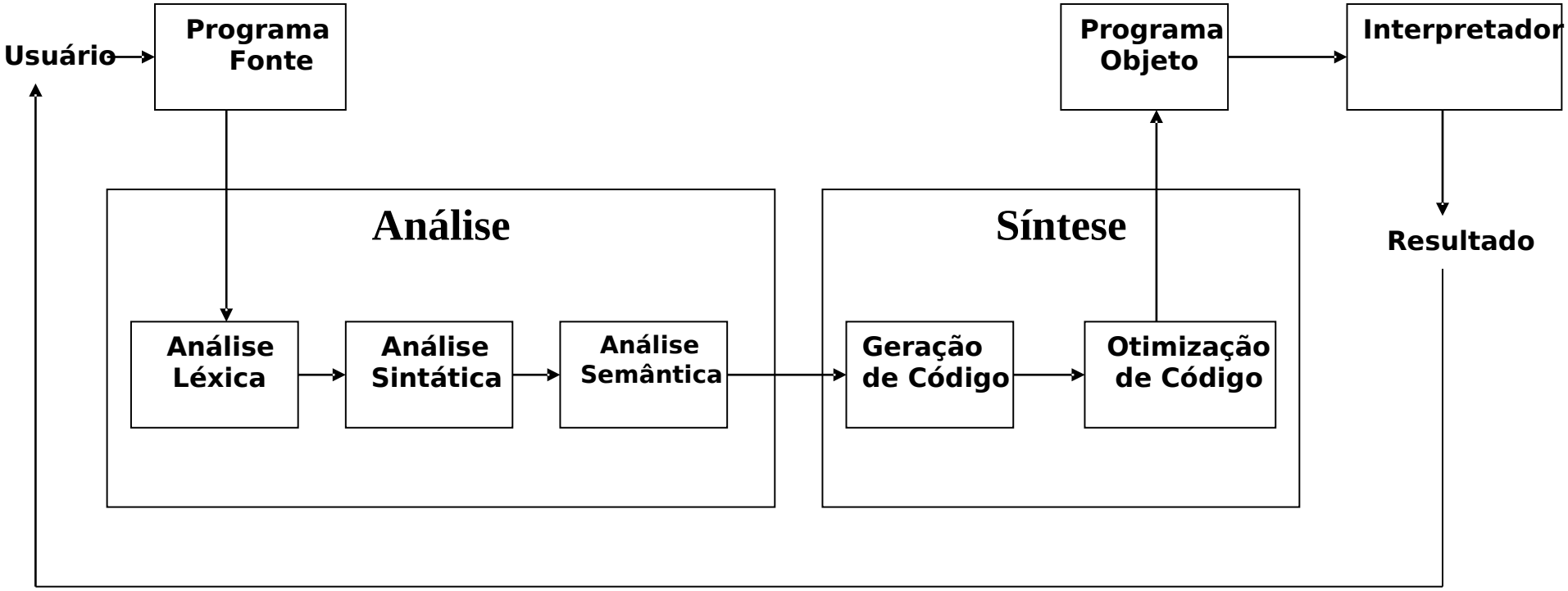
- A partir daí, através de processamento adicional, gerava-se o executável (ainda na forma de fitas e cartões)

Atualmente os compiladores operam integrados aos demais componentes do sistema de programação, fazendo parte do conjunto dos recursos oferecidos pelo sistema operacional.

Relacionamento dos compiladores com outr elementos de um sistema de programaçã



REPRESENTAÇÃO LÓGICA DE UM COMPILAD
SEGUNDO O MODELO DE ANÁLISE-SÍNTES



Análise Léxica

A entrada para um compilador é a cadeia de caracteres que representa o programa fonte.

O objetivo do bloco léxico é quebrar esta cadeia de Caracteres separando as palavras que nela estão representadas.

Por exemplo, a cadeia de entrada poderia ser:

```
RESULTADO:=RESULTADO+15
```

variável RESULTADO

comando de atribuição :=

variável RESULTADO

operação aritmética +

constante numérica 15

Assim, os 23 caracteres da cadeia de entrada são transformados em 5 novas entidades/**tokens**

Se a tabela de tokens for vista como um dicionário, então o processo léxico pode ser visto como o ato de agrupar letras em palavras e determinar suas localizações no dicionário.

Outras funções normalmente atribuídas ao bloco léxico são as de ignorar espaços em branco e comentários, além de detectar os erros léxicos.

A saída do analisadorléxico é uma sequênciade símbolos que é passada para a próxima fase, o analisador sintático.

Os símbolos nessa sequência são geralmente representados por códigos (valores inteiros).

Os símbolos reconhecidos pelo léxico se enquadram em uma das seguintes classes:

- Identificadores (do usuário, da linguagem - palavras-chaves, funções embutidas);
- Números (inteiros, reais, complexos);
- Operadores (aritméticos, relacionais, lógicos);
- Parentisadores (abre/fecha aspas, abre/fecha parênteses, abre/fecha colchetes, abre/fecha chaves, BEGIN/END);
- Sinais de pontuação (ponto, vírgula, ponto-e-vírgula).

ANÁLISE SINTÁTICA OU PARSER

- É um algoritmo
- Tem por finalidade verificar se:
  - o programa que está sendo compilado possui ou não erros de sintaxe. Verifica se as regras (produções) da gramática que definem a linguagem em questão, foram observadas na construção do programa fonte.

ANÁLISE SEMÂNTICA

O próximo passo no processo de construção de um compilador refere-se a análise semântica, normalmente composta por um conjunto de ações, as **ações semânticas**, responsáveis pela determinação do significado de cada instrução do programa fonte.

A avaliação das ações semânticas pode **criar e manter a tabela de símbolos**, gerar código-objeto e **emitir mensagens de erro**.

A análise semântica constitui-se, em linhas gerais, de um verificador de tipos. É de sua responsabilidade verificar se um operador contém operandos do mesmo tipo, se uma variável foi declarada, entre outras atribuições.

Se alguma dessas condições não for satisfeita, o sistema reportará um erro semântico.

Por exemplo, se tivéssemos declarado uma variável **x** como integer, uma variável **y** como string, e quiséssemos fazer uma atribuição do tipo **x:=x+y**, certamente seria gerado um erro semântico, porque não é possível adicionarmos um valor string a um inteiro e atribuí-lo a outro inteiro.

GERAÇÃO DE CÓDIGO

- **Síntese do programa-fonte.** Após o programa-fonte ter sido analisado e aprovado, o compilador tem condições de escrever um programa equivalente na linguagem-alvo.

- A síntese de um programa fonte tem complexidade proporcional a complexidade da estrutura da linguagem alvo.

OTIMIZAÇÃO DE CÓDIGO

Uma de suas funções pode ser a de fazer com que o programa-objeto rode o mais rápido possível.

Exemplo:

A:=2+10+B

A:=12+B