

# Construção de Compiladores

## Capítulo 1

### Introdução

José Romildo Malaquias

Departamento de Computação  
Universidade Federal de Ouro Preto

2013.2

1 Linguagens de programação

2 Compilação

1 Linguagens de programação

2 Compilação

- ▶ Os programadores escrevem instruções em várias **linguagens de programação**:
  - algumas são diretamente compreensíveis por computadores,
  - outras requerem passos intermediários de **tradução**.
- ▶ Três **tipos gerais de linguagem**:
  - Linguagens de máquina
  - Linguagens assembly (ou de montagem)
  - Linguagens de alto nível

- ▶ Qualquer computador pode entender diretamente apenas sua própria **linguagem de máquina**.
- ▶ Essa é a *linguagem natural* do computador, definida pelo projeto de hardware.
- ▶ Em geral, as linguagens de máquina consistem em strings de números (em última instância reduzidas a 1s e 0s) que instruem os computadores a realizar suas operações mais elementares uma de cada vez.
- ▶ **Dependentes de máquina:** uma linguagem de máquina é específica e só pode ser utilizada em um tipo de computador.
- ▶ Exemplo: somar o salário base às horas extras para calcular o salário bruto:

```
+1300042774  
+1400593419  
+1200274027
```

- ▶ Abreviações em inglês que representam operações elementares formam a base das **linguagens assembly**.
- ▶ **Programas tradutores** chamados assemblers convertem programas em linguagem assembly para a linguagem de máquina.
- ▶ Exemplo: somar o salário base às horas extras para calcular o salário bruto:

```
load salarioBase  
add salarioExtra  
store salarioBruto
```

- ▶ **Instruções simples** realizam tarefas substanciais.
- ▶ **Compiladores** convertem programas em linguagem de alto nível para a linguagem de máquina.
- ▶ Permitem aos programadores escrever instruções parecidas com o inglês cotidiano e contêm notações matemáticas comumente utilizadas.
- ▶ C, C++, Java, linguagens .NET da Microsoft (por exemplo, Visual Basic, Visual C++ e C#) estão entre as linguagens de programação de alto nível mais amplamente utilizadas.
- ▶ Exemplo: somar o salário base às horas extras para calcular o salário bruto:

```
salarioBruto = salarioBase + salarioExtra
```

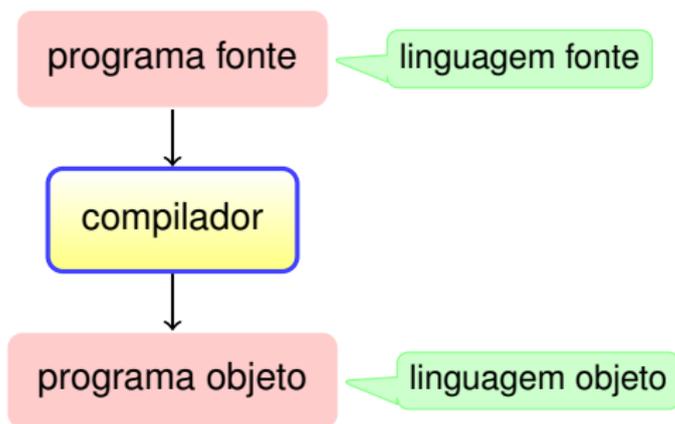
1 Linguagens de programação

2 Compilação

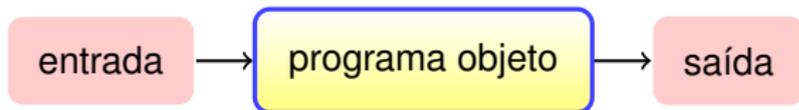
- ▶ Um **compilador** traduz (compila) um programa escrito em uma linguagem de programação de alto nível que é adequada para programadores humanos para a linguagem de máquina que é requerida pelo computador.
- ▶ Impacto do uso de uma linguagem de alto nível para programação no desenvolvimento:
  - notação mais próxima da maneira como humanos pensam sobre problemas
  - o compilador pode detectar alguns erros de programação óbvios
  - programas tendem a ser menores
- ▶ O processo de **compilação** de um programa de linguagem de alto nível em linguagem de máquina pode consumir uma quantidade considerável de tempo do computador.
- ▶ **Interpretores** executam linguagem de alto nível diretamente, embora sejam mais lentos que programas compilados.
- ▶ O Java utiliza uma combinação inteligente de compilação e interpretação para executar programas.

## Definição 1 (Compilador)

Um **compilador** é um programa de computador (ou um grupo de programas) que, a partir de um *código fonte* escrito em uma linguagem compilada, cria um programa semanticamente equivalente, porém escrito em outra linguagem, *código objeto*.

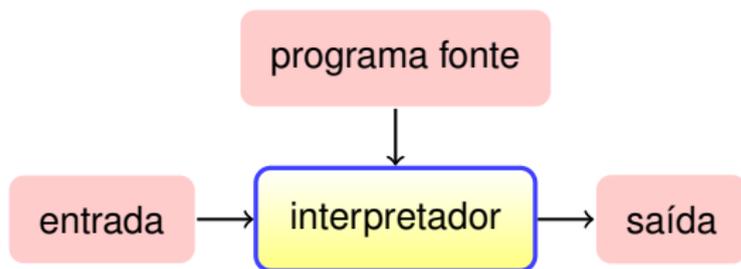


Quando o programa objeto é um programa em linguagem de máquina executável, ele poderá ser chamado pelo usuário para processar entradas e produzir saídas.



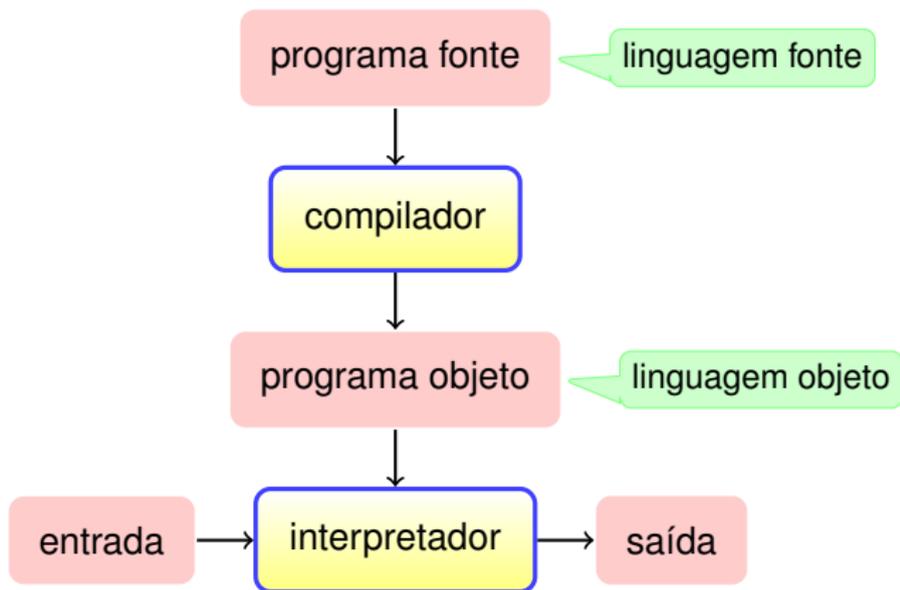
## Definição 2 (Interpretador)

**Interpretador** é um processador de linguagem que recebe como entrada um programa em uma linguagem de programação e executa diretamente as operações especificadas no programa sobre as entradas fornecidas pelo usuário.



# Processadores de linguagem: compilação + interpretação

Um programa fonte pode ser primeiramente compilado para uma linguagem intermediária, e o código intermediário gerado é então interpretado por uma máquina virtual.



# Porque estudar compiladores?

- ▶ Construir um grande e ambicioso sistema de software.
- ▶ Ver a aplicação da teoria na prática.
- ▶ Aprender a projetar linguagens de programação.
- ▶ Aprender como os processadores de linguagens funcionam.

- ▶ Inicialmente não havia nada.
- ▶ Então, criou-se linguagens de máquinas.
- ▶ Então, criou-se linguagens de montagem (*assembly*).
- ▶ Então, criou-se linguagens de alto nível.

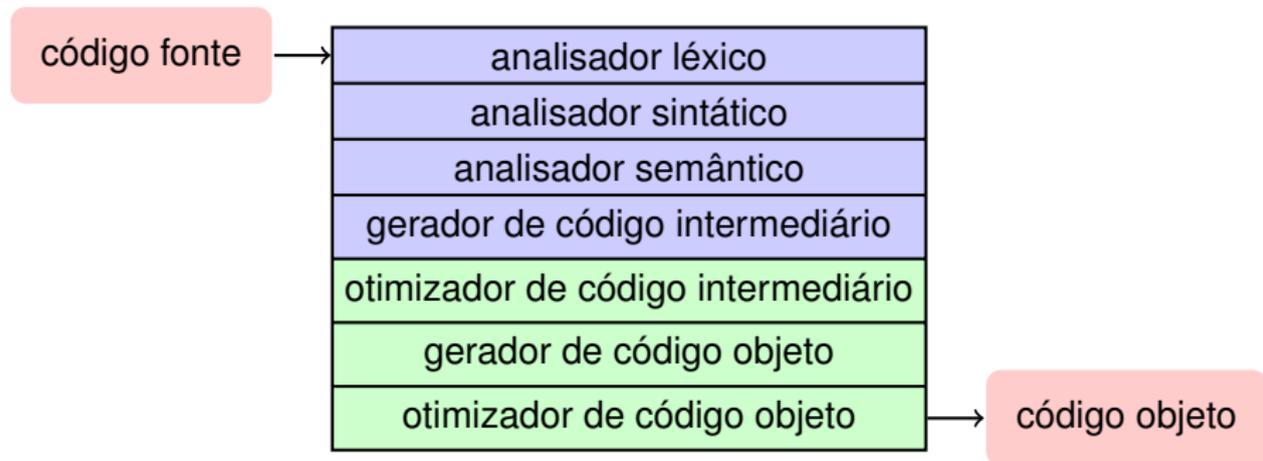
## ▶ **Front-end:**

- **Análise léxica:** identifica símbolos que compõem o programa.
- **Análise sintática:** identifica como estes símbolos se relacionam entre si.
- **Análise semântica:** identifica o significado destas relações.
- **Geração de código intermediário:** produz a estrutura na representação intermediária.

## ▶ **Back-end:**

- **Otimização do código intermediário:** simplifica as estruturas intermediárias geradas.
- **Geração de código objeto:** produz a estrutura na linguagem objeto.
- **Otimização de código objeto:** melhora a estrutura produzida.

# Estrutura de um compilador moderno (cont.)



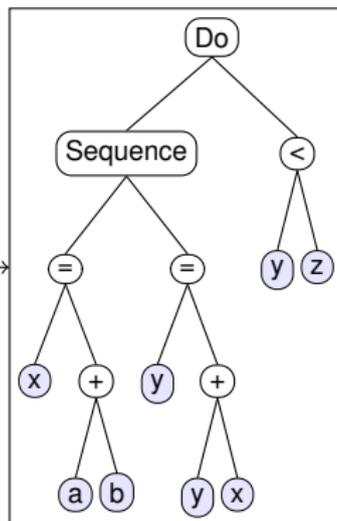
# Estrutura de um compilador moderno (cont.)

```
do {  
  int x = a + b;  
  y += x;  
} while (y < z);
```

análise  
léxica

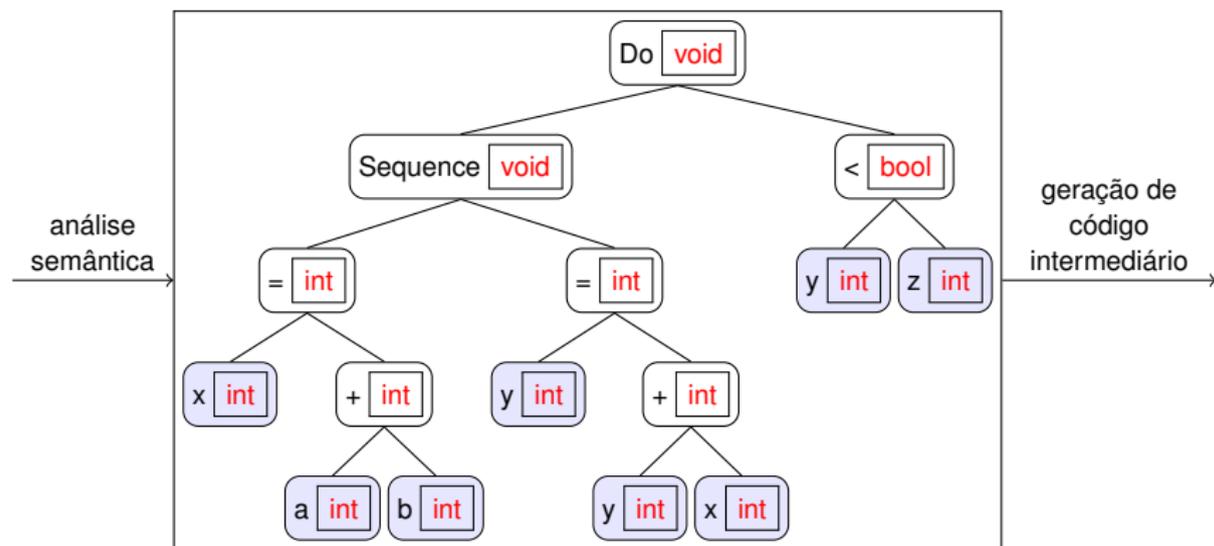
```
T_Do  
T_OpenBrace  
T_Int  
T_Identifier x  
T_Assign  
T_Identifier a  
T_Plus  
T_Identifier b  
T_Semicolon  
T_Identifier y  
T_PlusAssign  
T_Identifier x  
T_Semicolon  
T_CloseBrace  
T_While  
T_LeftParen  
T_Identifier y  
T_Less  
T_Identifier z  
T_RightParen  
T_Semicolon
```

análise  
sintática



análise  
semântica

# Estrutura de um compilador moderno (cont.)



# Estrutura de um compilador moderno (cont.)

