Construção de Compiladores

Capítulo 1

Introdução

José Romildo Malaquias

Departamento de Computação
Universidade Federal de Ouro Preto

2014.2

1 Linguagens de programação

2 Compilação

Tópicos

1 Linguagens de programação

2 Compilação

Linguagens de programação

- 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

Linguagens de máquina

- 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
```

Linguagens assembly

- 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
```

Linguagens de alto nível

- **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
```

Tópicos

1 Linguagens de programação

2 Compilação

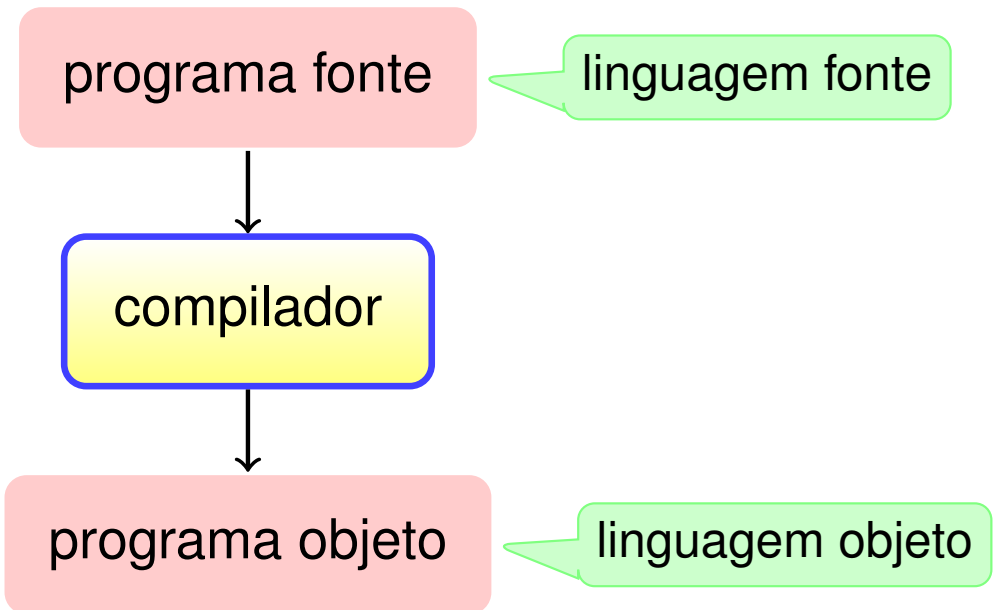
Compilação

- 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.
- **Interpretadores** 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.

Processores de linguagem: compiladores

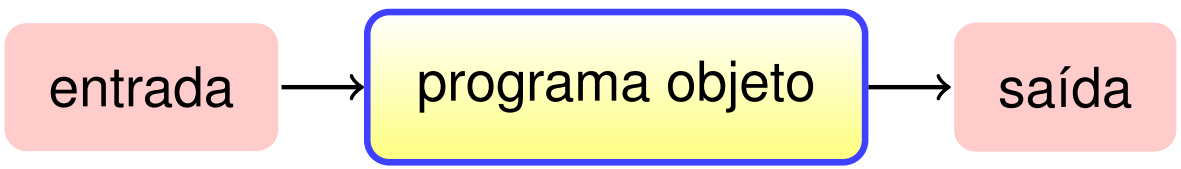
Definition 1 (Compilador)

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*.



Processadores de linguagem: execução

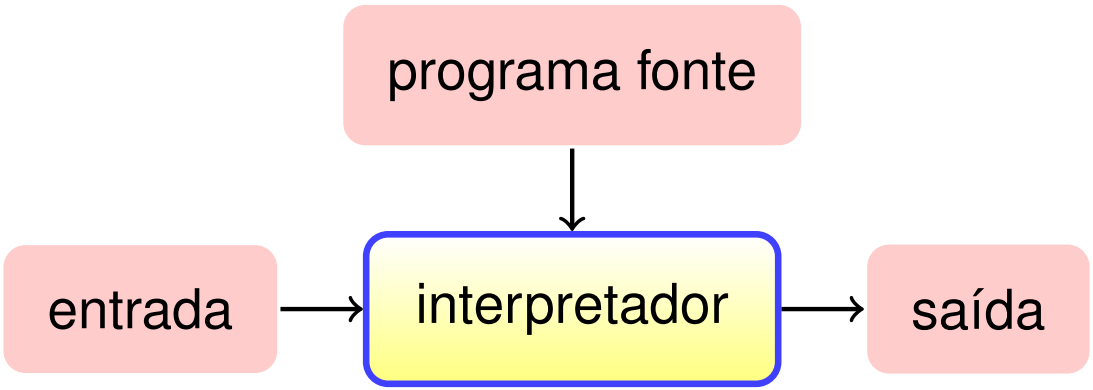
Quando o programa objeto é um programa em linguagem de máquina executável, ele poderá ser chamado pelo usuário para processar entradas e produzir saídas.



Processadores de linguagem: interpretador

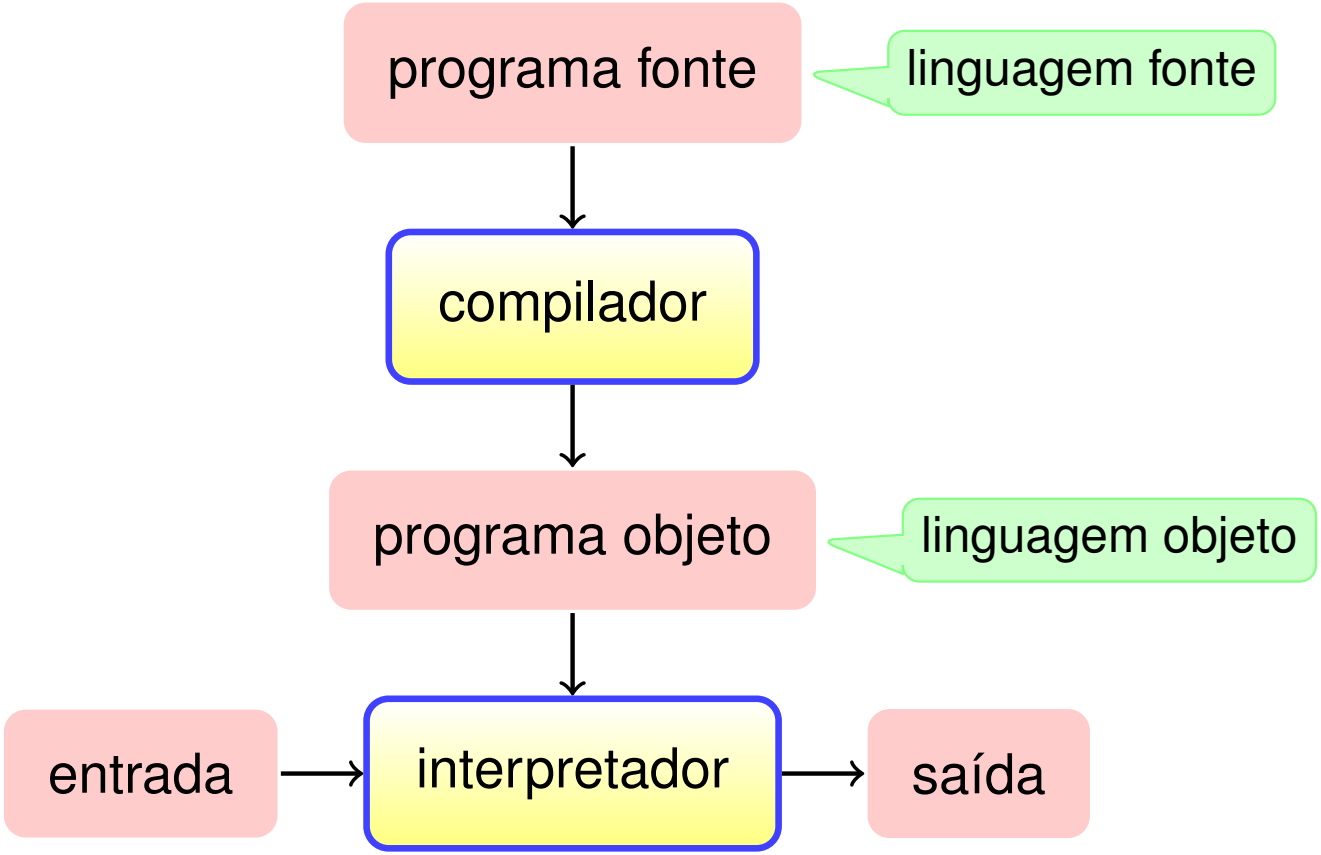
Definition 2 (Interpretador)

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 codigo intermediário gerado é então interpretado por uma máquina virtual.



Comparação de compilação e interpretação

- As etapas de análise são comuns a interpretadores compiladores.
- Ao invés de gerar código a partir da árvore abstrata, um interpretador processa diretamente a árvore abstrata para avaliar expressões e executar comandos.
- Um interpretador pode precisar processar o mesmo trecho de código várias vezes (em um loop, por exemplo), tipicamente fazendo com que a interpretação seja menos eficiente do que a execução de um programa compilado.
- Escrever um interpretador frequentemente é mais simples do que escrever um compilador.
- Um interpretor pode ser mais facilmente transferido de uma plataforma para outra.
- Assim interpretadores são frequentemente usados para aplicações onde velocidade não é essencial.

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.

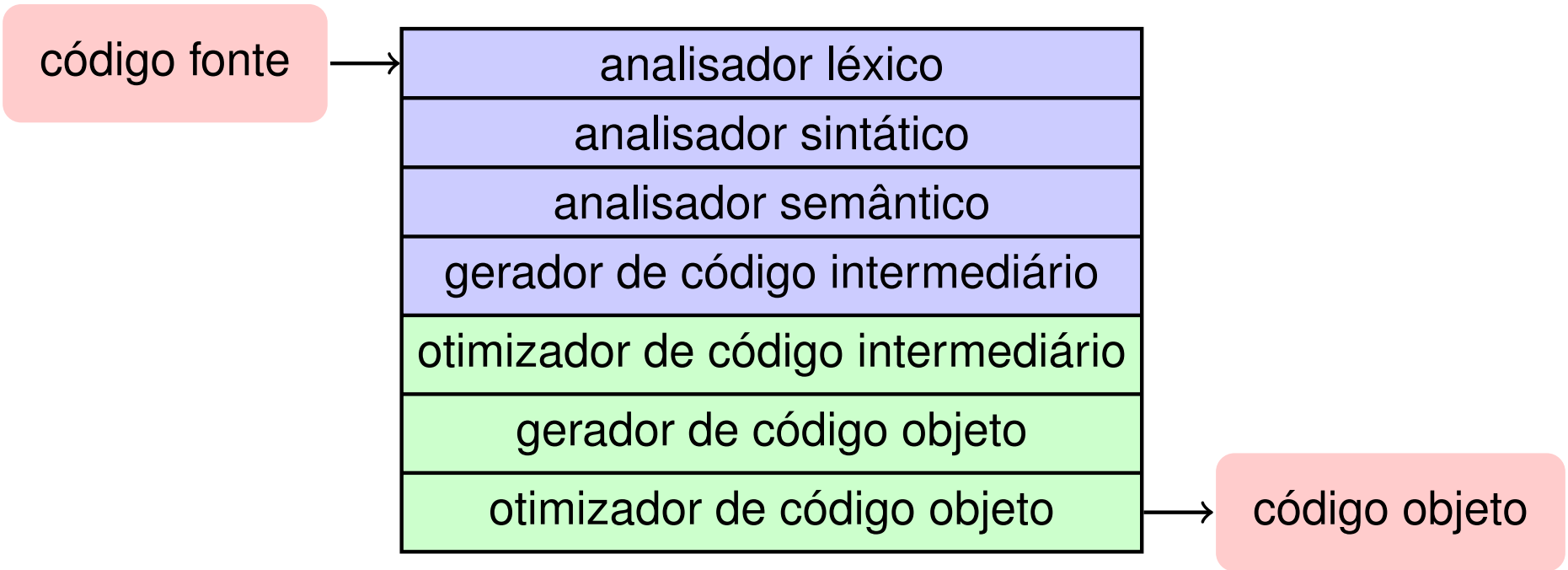
Breve histórico de compiladores

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

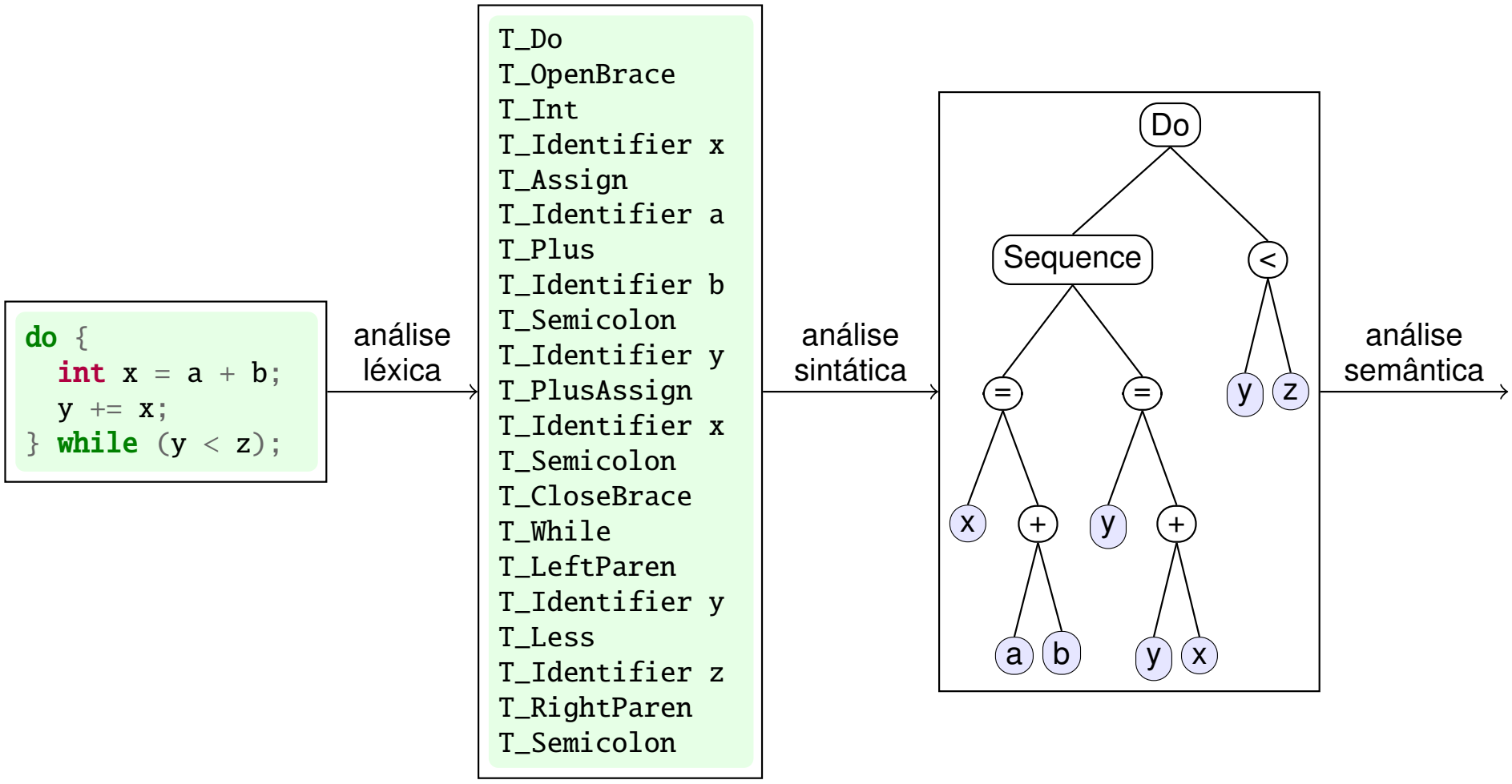
Estrutura de um compilador moderno

- ***Front-end***:
  - **Análise léxica**: identifca símbolos que compõem o programa.
  - **Análise sintática**: identica 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.

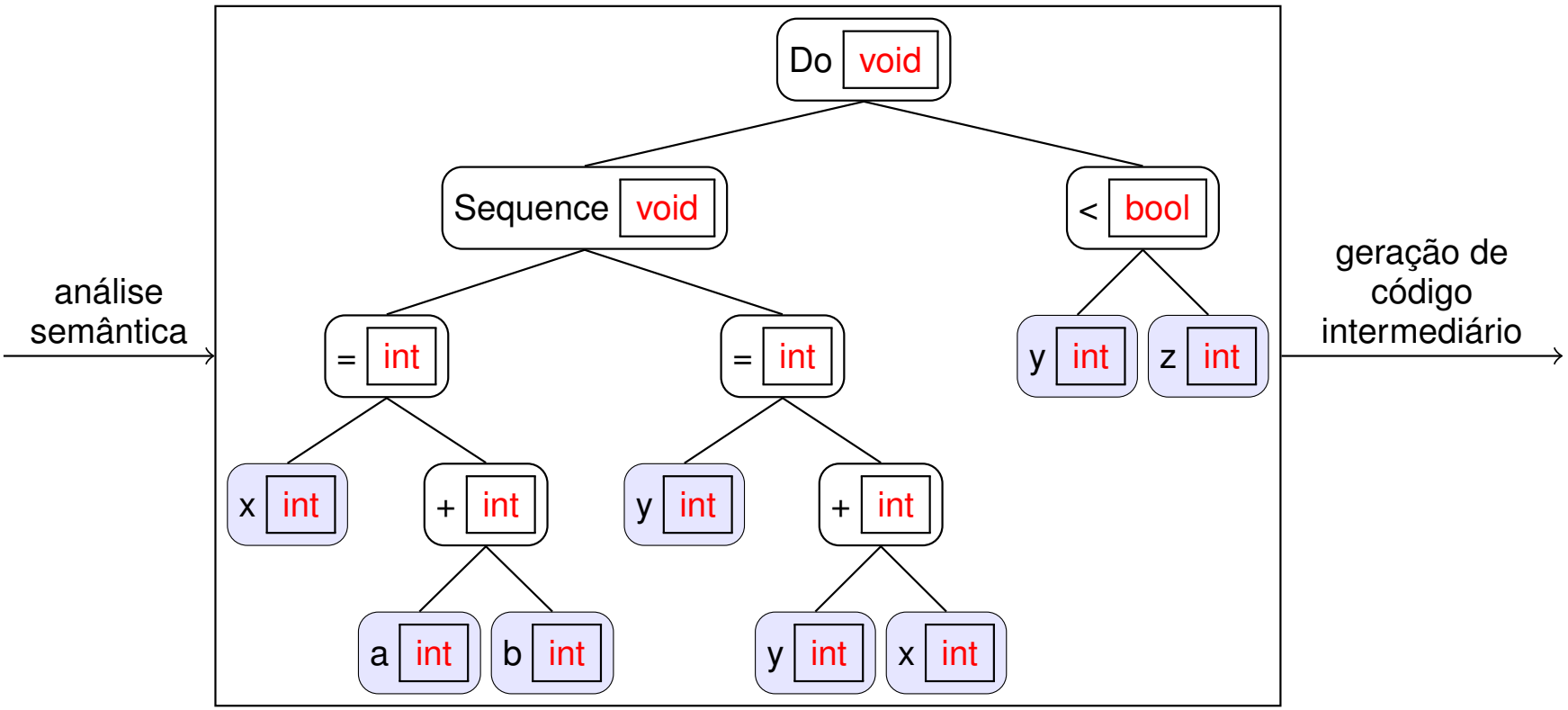
Estrutura de um compilador moderno (cont.)



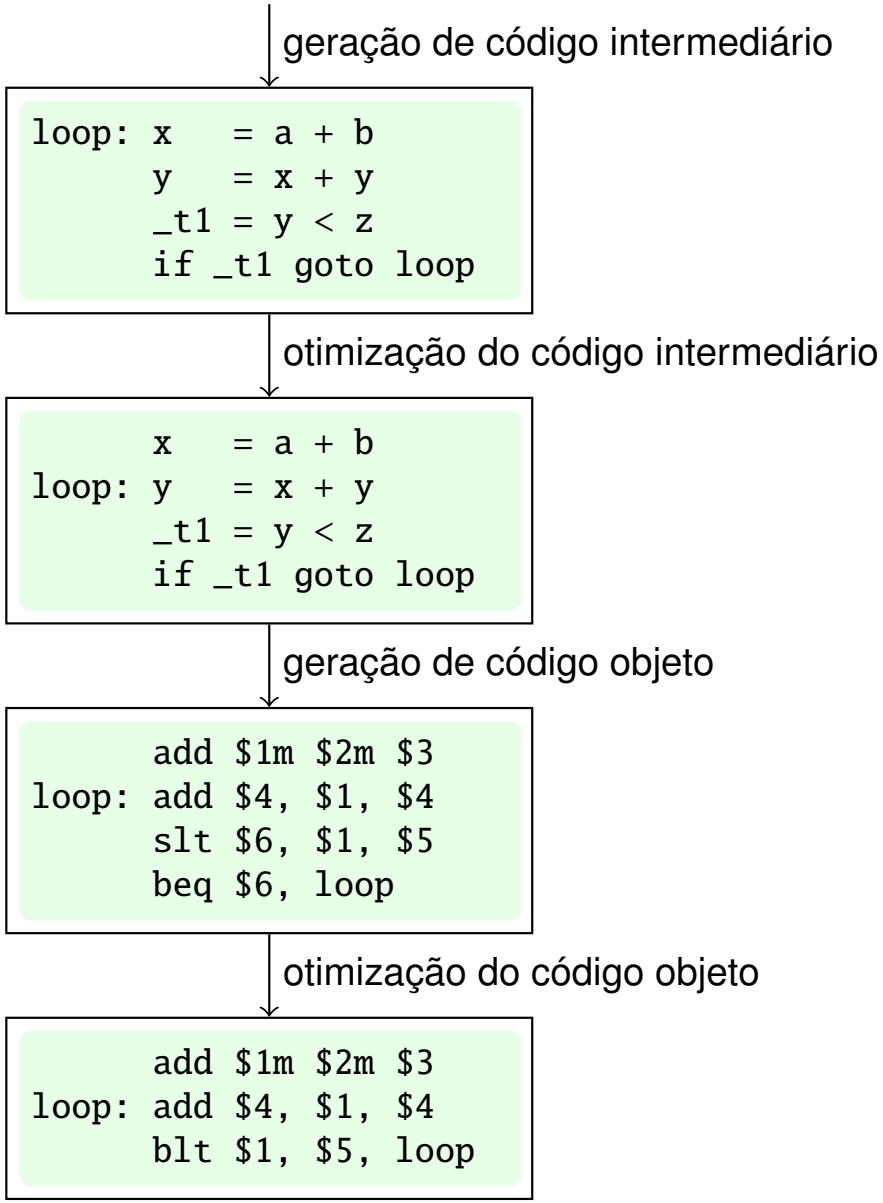
Estrutura de um compilador moderno (cont.)



Estrutura de um compilador moderno (cont.)



Estrutura de um compilador moderno (cont.)



Fim