Compiladores

Aula 2: Introducao aos Compiladores

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Fevereiro de 2026

Sumário

Objetivos da Aula

- Definir **formalmente** o conceito de compilador.
- Diferenciar modelos de execucao: **AOT**, **Interpretes** e **JIT**.
- Compreender a estrutura de **T-Diagrams**.
- Visualizar a anatomia de um compilador (Front-end vs Back-end).

Definicao Formal

Um compilador C e uma funcao que mapeia um programa P_S em linguagem fonte para um programa equivalente P_T em linguagem alvo:

$$C : L_S \rightarrow L_T$$

Preservacao Semantica

$$\forall input : Exec(P_S, input) \equiv Exec(P_T, input)$$

O comportamento deve ser **identico**, mesmo que a estrutura interna mude.

1. Compiladores Puros (AOT)

Ahead-of-Time: Traducao completa *antes* da execucao.

`Fonte (.c)` $\rightarrow$ `Compilador` $\rightarrow$ `Binario (.exe)`

- **Exemplos**: C, C++, Rust, Go.
- **Pros**: Maxima performance, otimizacao agressiva.
- **Contras**: Ciclo lento, binario preso a plataforma.

2. Interpretes Puros

Execucao direta do codigo fonte por uma Maquina Virtual, instrucao a instrucao.

`Fonte (.py)` $\rightarrow$ `Interprete (Le` $\rightarrow$ `Executa)`

- **Exemplos**: Bash, Python (conceitualmente), PHP antigo.
- **Pros**: Flexibilidade, portabilidade total.
- **Contras**: Performance baixa (overhead de decodificacao).

3. Hibridos (Bytecode + JIT)

Traducao para linguagem intermediaria (Bytecode) + Execucao por VM eficiente.

`Java` $\rightarrow$ `Bytecode (.class)` $\rightarrow$ `JVM` $\rightarrow$ `Codigo Nativo`

Just-In-Time (JIT) Compilation

A VM detecta funcoes executadas frequentemente ("hot spots") e as compila para codigo de maquina *durante a execucao*.

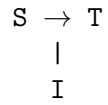
- Une portabilidade do bytecode com performance do codigo nativo.
- Exemplos: Java, C#, JavaScript (V8).

Diagramas de Lapide (T-Diagrams)

Ferramenta para descrever processos de compilacao e *bootstrapping*.

Formato em "T":

- **Topo Esq (S)**: Linguagem Fonte.
- **Topo Dir (T)**: Linguagem Alvo.
- **Base (I)**: Linguagem de Implementacao.



Exemplo: Um compilador de Java para Bytecode escrito em Java.

Modelo Analise-Sintese

Front-End (Analise)

1. **Lexico**: Tokens (palavras).
2. **Sintatico**: Arvore (estrutura).
3. **Semantico**: Tipos e Escopo.

Saida: Representacao Intermediaria (IR/AST).

Back-End (Sintese)

1. **Geracao de Codigo IR**: Codigo generico.
2. **Otimizacao**: +Rapido, -Tamanho.
3. **Cod. Alvo**: Assembly/Binario final.

Referencias

- Aho, et al. *Compilers: Principles, Techniques, and Tools* (Livro do Dragao).
- Cooper & Torczon. *Engineering a Compiler*.

Duvidas?

Obrigado!

Email: alessio@cefetmg.br