Compiladores e Computabilidade

Prof. Leandro C. Fernandes
UNIP – Universidade Paulista, 2018

Motivação e
Conceitos fundamentais

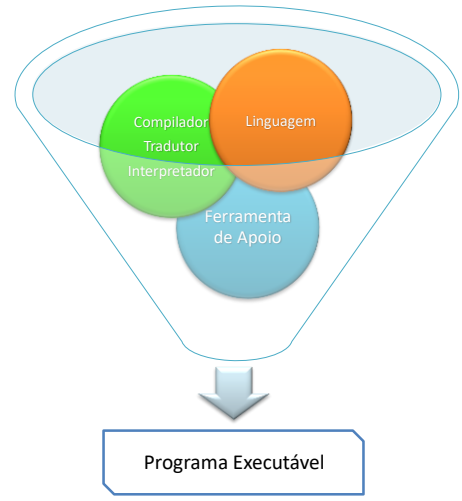
INTRODUÇÃO

Por que estudar Compiladores?

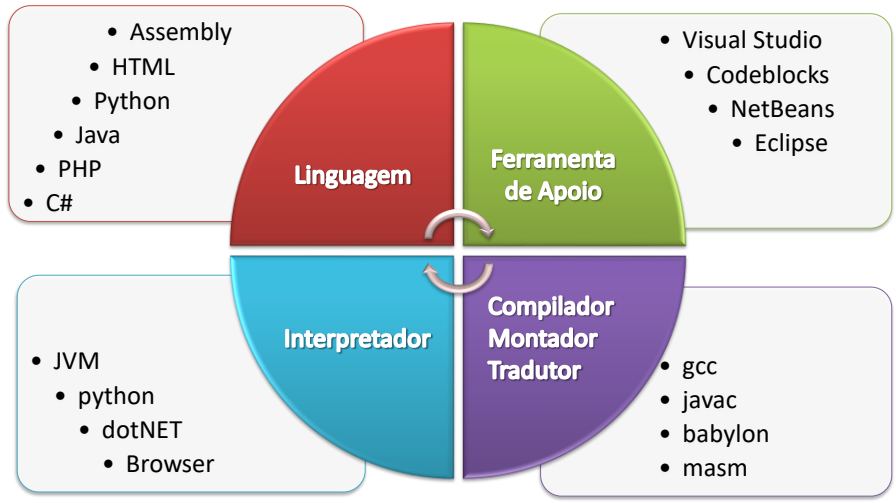
- É parte do conhecimento básico de um desenvolvedor
  - Como os compiladores funcionam?
  - Como os computadores trabalham?
    - (conjuntos de instruções, registradores, modos de endereçamento, estrutura de dados de tempo de execução, ...)
  - Quais códigos de máquina são gerados para certas construções da linguagem?
    - (consideração a cerca da eficiência)
  - O que é um bom projeto de linguagem?
  - É um projeto de programação não trivial!
- Também é útil para o desenvolvimento de softwares em geral
  - Leitura de argumentos estruturados sintaticamente
    - Linha de comando (shell S.O.)
  - Leitura de dados estruturados
    - (ex. Arquivos XML, part lists, imagens, ...)
  - Buscas em espaços de nomes hierárquicos
  - Interpretação de códigos de comandos

O que são essas coisas?

- Visual Studio
- Netbeans
- gcc
- Browser
- Eclipse
- Java
- HTML
- Assembler
- PHP
- Assembly
- Babylon
- Python
- JVM
- .NET
- Pascal Objects
- Lua



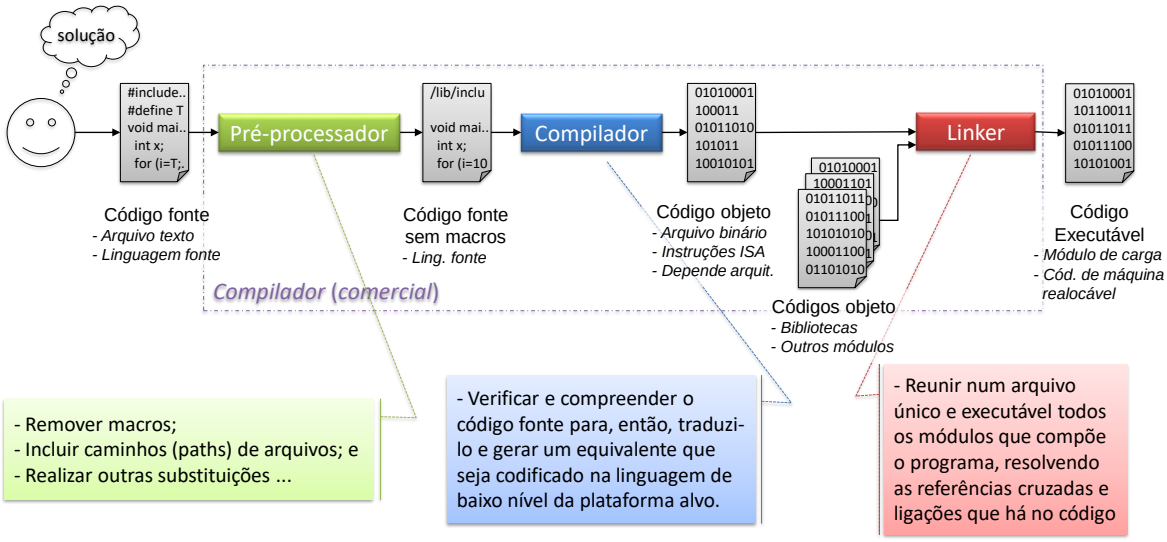
O que são essas coisas?



O que está acontecendo !?

- Qual é todo o processo que ocorre desde a codificação de um algoritmo por um programador até a sua execução em um computador?
  - Quais os elementos envolvidos neste processo?
  - Quais tarefas cabem a cada um desses elementos?
  - Quais fatores influenciam de um ou de outro modo esse processo?
  - Quais elementos podem ser reutilizados?
  - Quais elementos são específicos?

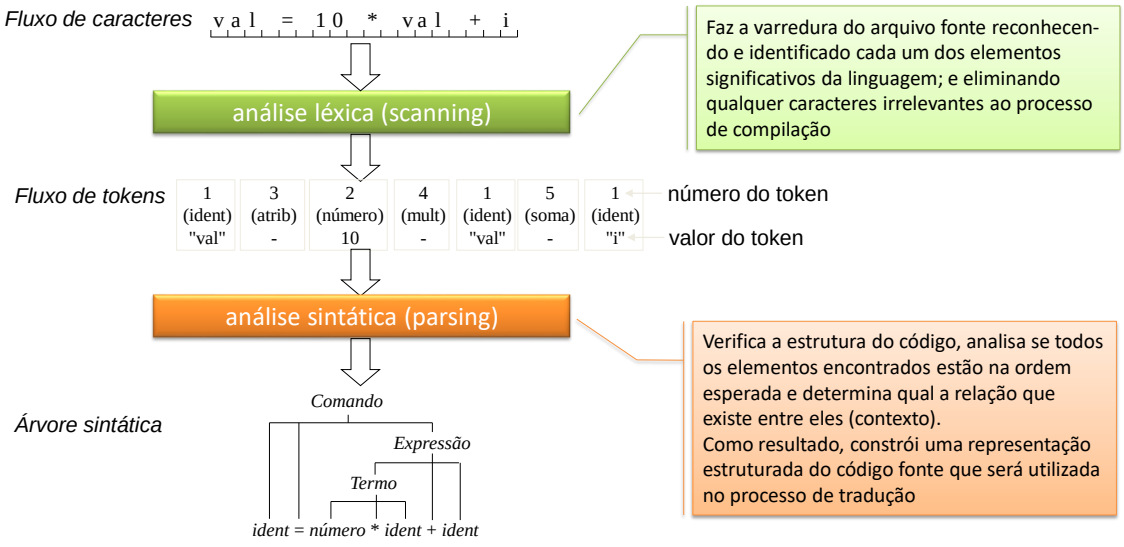
Da codificação ao executável ...



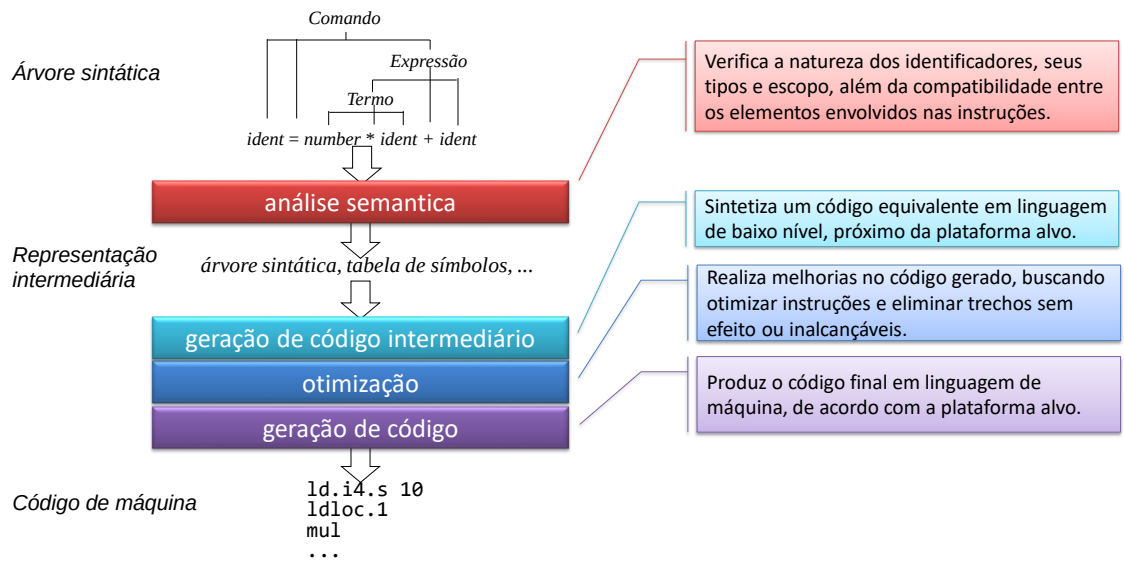
Quais tarefas são realizadas para a transformação do código?

POR DENTRO DO COMPILADOR

Estrutura Dinâmica de um Compilador

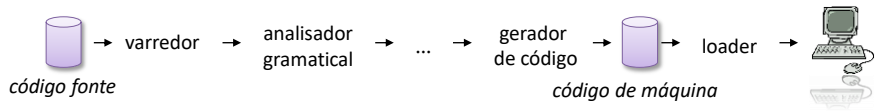


Estrutura Dinâmica de um Compilador

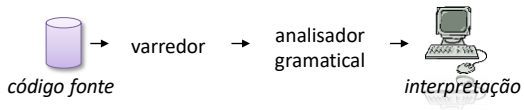


Compiladores vs Interpretadores

Compilador traduz para código de máquina

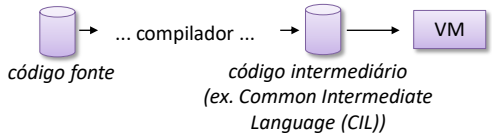


Interpretador executa o código fonte "diretamente"



- comandos em um laço são lidos e analisados gramaticalmente a cada iteração

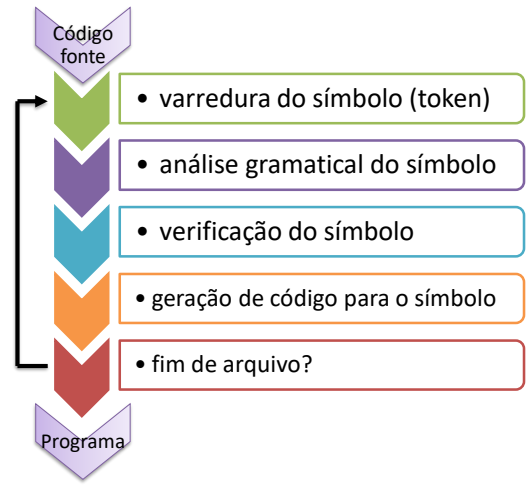
Variação: interpretação de um código intermediário



- código fonte é traduzido para o código de uma *máquina virtual*
- A máquina interpreta o código simulando a máquina real

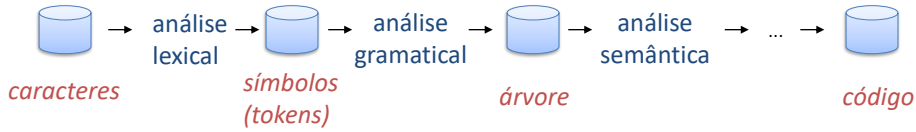
Compiladores de Uma Passagem

- As fases trabalham de maneira intercalada:
  - Cada etapa é realizada para o trecho ou seção corrente da análise;
  - Quando esta parte termina, o processo é retomado para o próximo trecho de código.
- O programa alvo está completo quando terminar a leitura do programa fonte



Compiladores de Multiplas Passagens

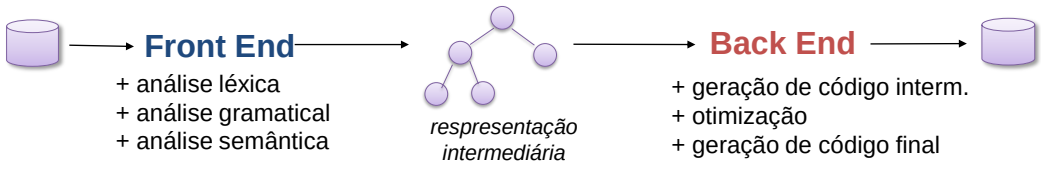
- As fases são "programas" separados, que são executados sequencialmente



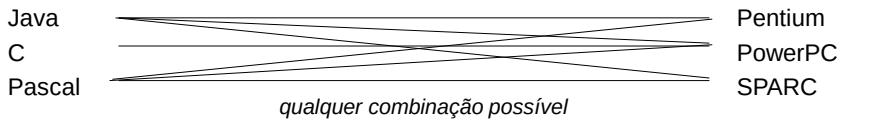
- Cada fase lê de um arquivo e escreve em um novo arquivo.

- **Por quê multiplas passagens?**
  - se a memória for escassa (irrelevante atualmente)
  - se a linguagem for complexa
  - se a portabilidade for importante

Compiladores de Duas Passagens (Atual)



dependente da linguagem **dependente da máquina**



qualquer combinação possível

Vantagens
- melhor portabilidade
- possibilidade de muitas combinações entre front-ends e back-ends
- otimizações são mais fáceis na representação intermediária do que no código fonte

Desvantagens
- lenta
- necessita mais memória

Estrutura Estática de um Compilador