Introdução à Programação

Linguagens de Programação:
sintaxe e semântica de linguagens de programação e conceitos de linguagens interpretadas e compiladas

Engenharia da Computação

Professor: Críston Pereira de Souza

Web: www.univasf.edu.br/~criston.souza/algoritmos.html

email: criston.souza@univasf.edu.br

Sumário

- Sintaxe
- Semântica
- Linguagens interpretadas
- Linguagens compiladas
- Máquina virtual

Sintaxe

- A sintaxe de uma LP é a forma de suas expressões, de suas instruções e de suas unidades de programa.
- Ex.: sintaxe comando *if* (*se*) da linguagem C.
  - if <expr> <cmd>
  - *se o valor atual da expressão for verdadeiro, a instrução incorporada será selecionada para execução*.

Sintaxe

- As regras sintáticas especificam quais sequências de caracteres do alfabeto da linguagem estão nela.
- As descrições formais da sintaxe das LPs são chamadas de lexemas.
- Os lexemas de uma LP incluem seus identificadores, palavras reservadas, literais e operadores.

Sintaxe

- Um *token* (simbolo) de uma LP é uma categoria de seus lexemas.
- Exemplo instrução em linguagem C:
  - index = 2 * cont + 17;
  - Os lexemas e os símbolos dessa instrução são:

Lexemas	Símbolos
index	Identificador
=	sinal igual
2	literal
*	operador de multiplicação
cont	identificador
+	operador de adição
17	Literal
;	ponto e vírgula

Sintaxe

- Métodos formais para descrever a sintaxe
  - O lingüista Noam Chomsky, na década de 50, descreveu quatro classes de gramáticas que definem quatro classes de linguagens.
  - Destas, as gramáticas livres de contexto e regulares, passaram a ser úteis para descrever a sintaxe de LPs.
  - BNF – *Backus-Naur Form – Forma de Backus-Naur*, é uma notação muito utilizada para descrever a sintaxe de LPs.

Sintaxe

- Métodos formais para descrever a sintaxe
  - A BNF usa abstrações para estruturas sintáticas.
  - Exemplo:
    - <atribuição> -> <var> = <expressão>
  - O símbolo a esquerda da seta, chamado de lado esquerdo LE, é a abstração que está sendo definida.
  - O texto a direita da seta é a definição do LE, chamado de lado direito LD.
  - A definição é chamada de *regra* ou *produção*.
  - Exemplo:
    - soma = valor1 + valor2

Sintaxe

- Métodos formais para descrever a sintaxe
  - As abstrações em uma descrição BNF são chamadas *símbolos não-terminais*.
  - Os lexemas e o símbolos das regras são chamados *símbolos terminais*.
  - Exemplo:
    - <instr_if> -> **if** <expr_logica> <instrução> |
      if <expr_logica> <instrução> **else** <instrução>
  - A BNF é suficientemente poderosa para descrever a grande maioria das sintaxes das linguagens de programação.

Sintaxe

- Métodos formais para descrever a sintaxe
  - A BNF é um dispositivo generativo para definir linguagens.
  - As sentenças da linguagem são geradas por uma sequência de aplicações das regras, iniciando-se com um símbolo não-terminal, chamado símbolo de início.
  - Uma geração da sentença é chamada derivação.

Sintaxe

- Métodos formais para descrever a sintaxe
  - Exemplo 1:
    - <programa> -> inicio <lista_de_comandos> fim
      <lista_de_comandos> -> <instrução>
      | <instrução>; <lista_de_comandos>
      <instrução> -> <var> = <expressão>
      <var> -> A | B | C
      <expressão> -> <var> + <var>
      | <var> - <var>
      | <var>

Sintaxe

- Métodos formais para descrever a sintaxe
  - Exemplo 2:
    - <atribuição> -> <id> + <expressão>
      <id> -> A | B | C
      <expressão> -> <id> + <expressão>
      | <id> * <expressão>
      | (<expressão>)
      | <id>

Sintaxe

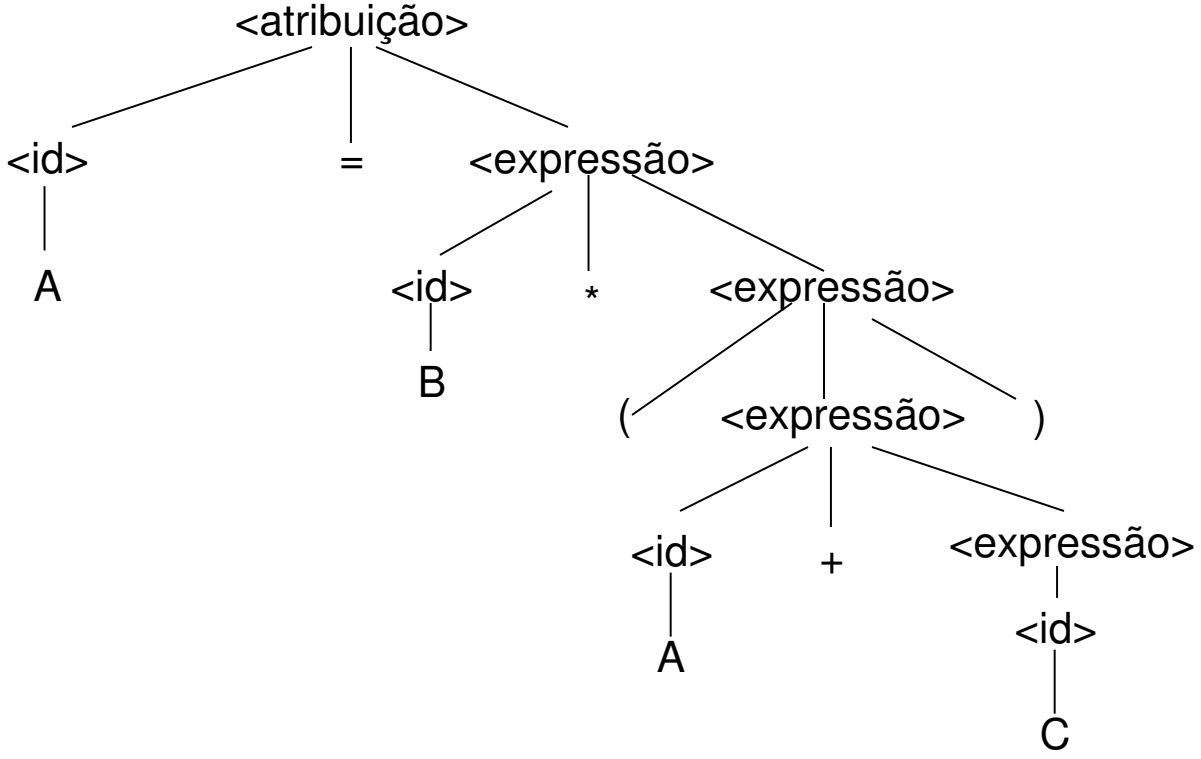
- Métodos formais para descrever a sintaxe
  - Exemplo 2: A = B * (A + C)
    - <atribuição> = > <id> = <expressão>
      = > A = <expressão>
      = > A = <id> * <expressão>
      = > A = B * <expressão>
      = > A = B * (<expressão>)
      = > A = B * (<id> + <expressão>)
      = > A = B * (<A> + <expressão>)
      = > A = B * (<A> + <id>)
      = > A = B * (A + C)

Sintaxe

- Métodos formais para descrever a sintaxe
  - As gramáticas descrevem naturalmente a estrutura sintática hierárquica das linguagens que definem.
  - As estruturas definidas pelas gramáticas são chamadas de *árvores de análise*.
  - Todo vértice interno de uma árvore de análise é rotulada com um símbolo não-terminal; toda folha é rotulada com um símbolo terminal.
  - Toda subárvore de uma árvore de análise descreve uma instância de uma abstração na instrução.

Sintaxe

- Métodos formais para descrever a sintaxe
  - Exemplo 2: A = B * (A + C)



Semântica

- A semântica trata da análise do significado das expressões, das instruções e das unidades de programa.
- A semântica é importante para que os programadores saibam precisamente o que as instruções de uma linguagem fazem.

Semântica

- Semântica operacional
  - Descreve o significado de um programa ao executar suas instruções em uma máquina, real ou simulada.
  - As alterações que ocorrem no estado de uma máquina, quando determinada instrução é executada, define o significado desta.
  - O estado de um computador é definido pelos valores de todos os seus registradores e de suas localizações de memória.

Semântica

- Semântica operacional
  - Para avaliar a semântica operacional de um programa é necessário o uso de um simulador (computador hipotético) e então traduzir a linguagem de alto nível, descrita pela LP, em linguagem de alto nível, que será interpretada pelo simulador.

Semântica

- Semântica operacional

Exemplo: Intrução em linguagem C:

```
for (expr1; expr2; expr3) {
  ...
}
```

Semântica operacional:

```
expr1;
loop: if expr2 = 0 goto out
   ...
   expr3;
   goto loop
out: ...
```

Semântica

- Semântica axiomática
  - Método para provar a exatidão dos programas que mostra a computação descrita por sua especificação.
  - Cada instrução de um programa tanto é precedida como seguida de uma expressão lógica que especifica restrições a variáveis.
  - As expressões lógicas são usadas para especificar o significado das instruções.
  - As restrições são descritas pela notação do cálculo de predicados.

Semântica

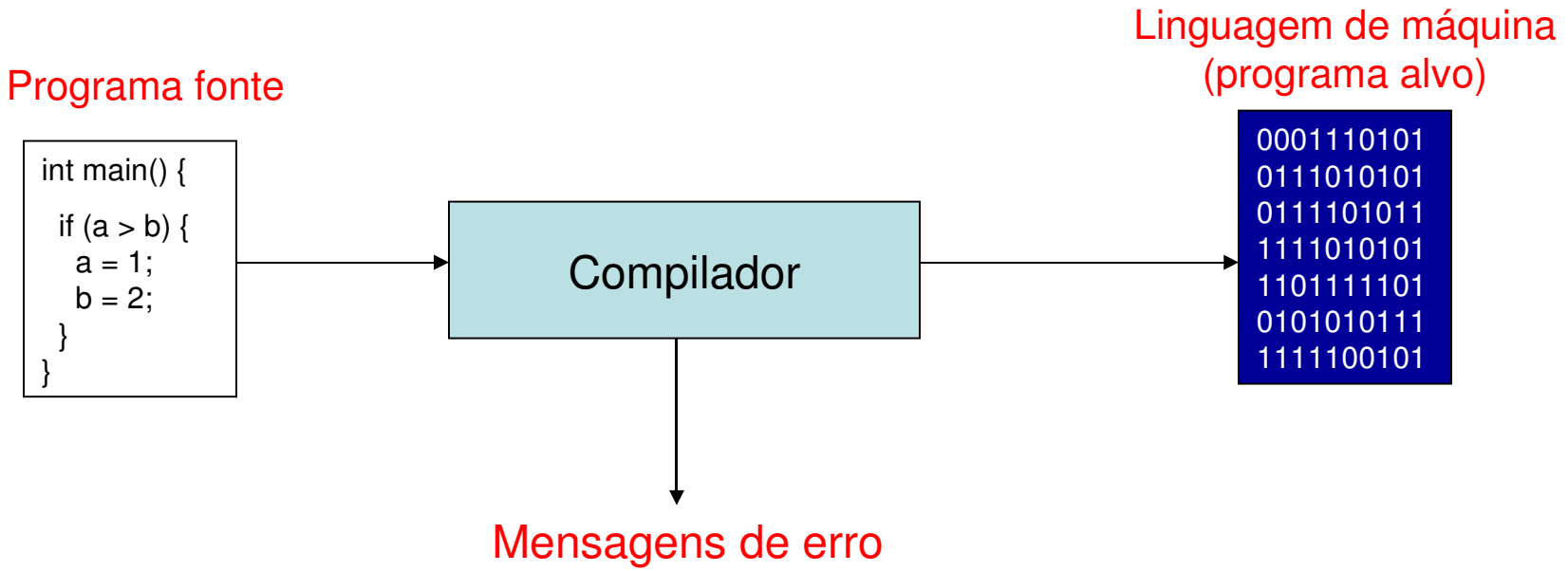
- Semântica denotacional
  - Baseia-se na teoria da função recursiva.
  - Para cada entidade da linguagem deve ser definido tanto um objeto matemático como uma função que relacione instâncias daquela entidade com as deste.
  - Os objetos representam o significado exato de suas entidades correspondentes.
  - A dificuldade no uso deste método está em criar os objetos e as funções de correspondências.
  - Os objetos matemáticos denotam o significado de suas entidades sintáticas correspondentes.

Linguagens Compiladas

- Programas escritos em linguagens de alto nível devem ser traduzidos para linguagens de máquina.
- Um compilador implementa uma linguagem fonte traduzindo programas escritos nessa linguagem para a linguagem objeto da máquina alvo onde os programas vão ser executados.

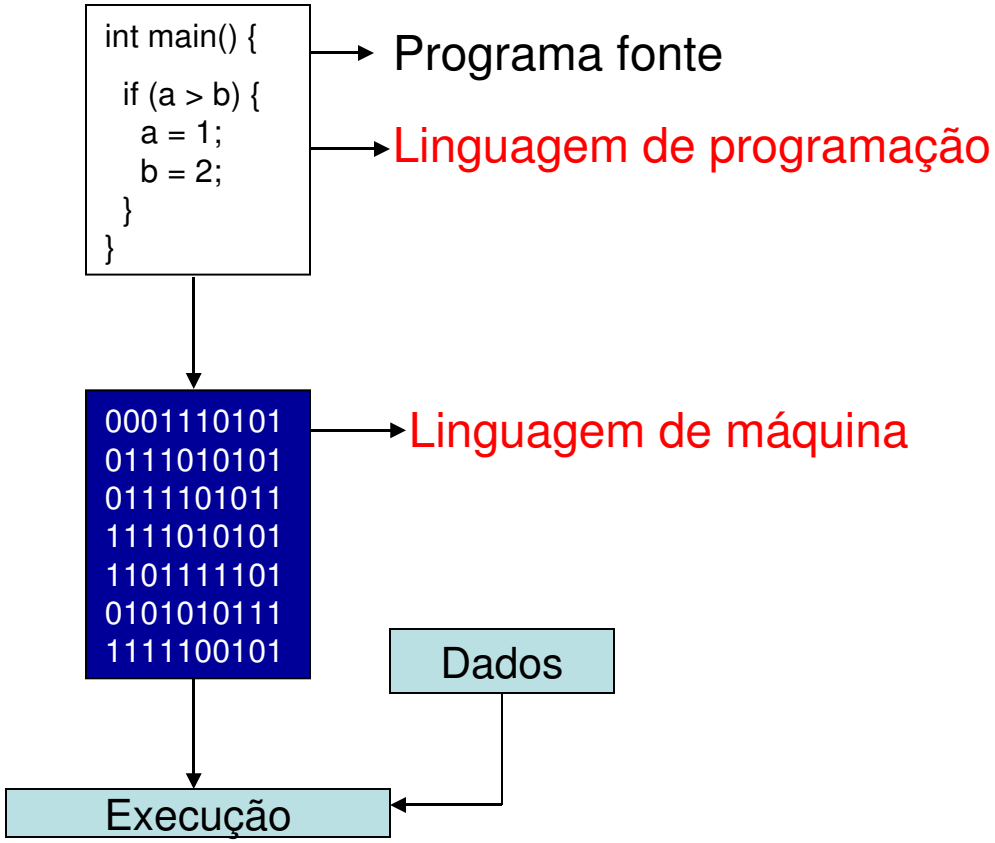
Linguagens Compiladas

- Compilador: visão geral



Linguagens Compiladas

- Compilador: visão geral

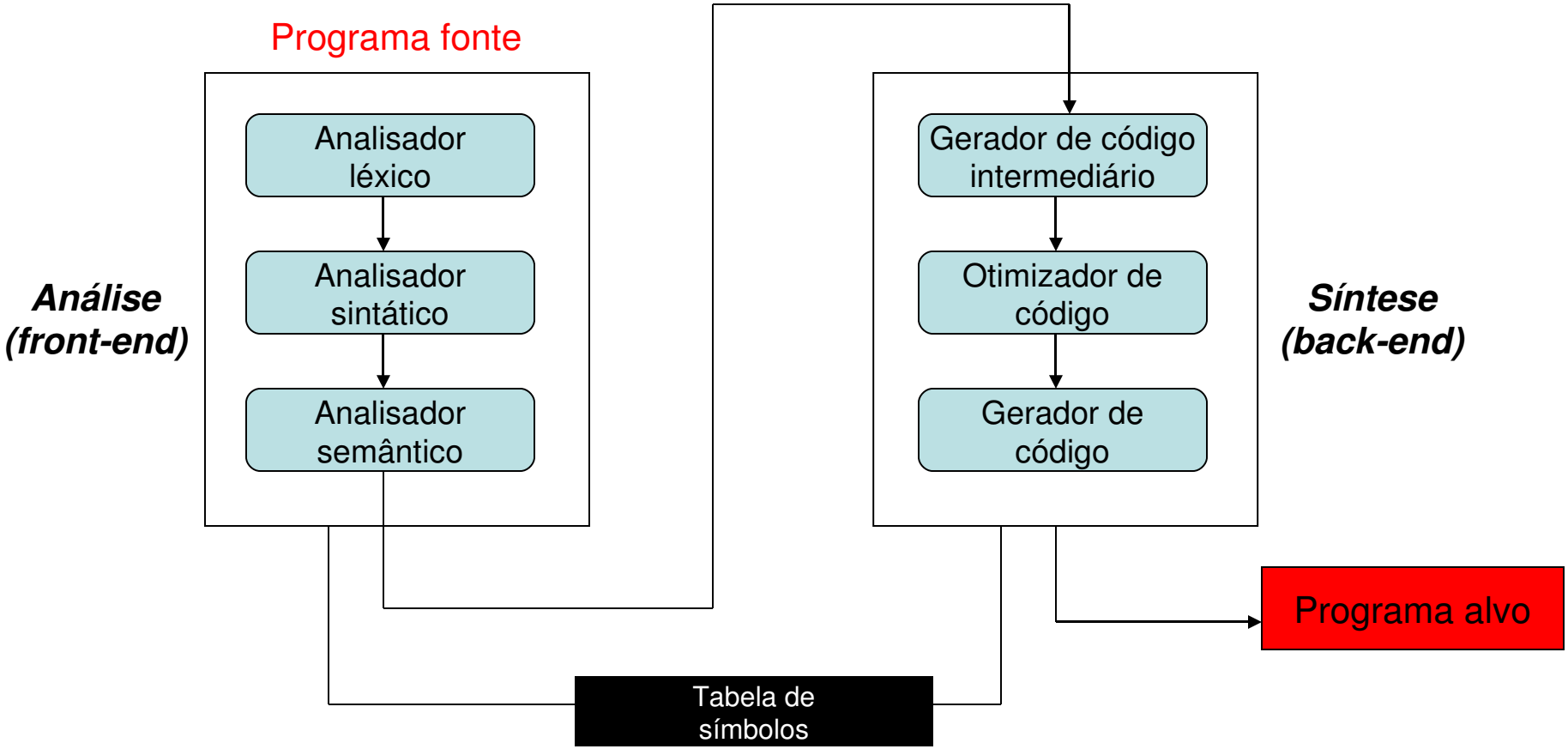


Linguagens Compiladas

- Fases de um compilador
  - Análise
    - Reconhece e divie e programa em partes.
    - Cria uma representação intermediária do programa.
    - Também chamada de fase *front end*.
  - Síntese
    - A partir da representação intermediária, constrói o programa alvo desejado.
    - Também chamada de fase *back end*.

Linguagens Compiladas

- Fases de um compilador



Linguagens Compiladas

- Fases de um compilador
  - Análise
    - Análise Léxica
      - O programa é *scaneado* e agrupado em *tokens.*
    - Análise Sintática
      - O *tokens* que foram reconhecidos na fase de análise léxica são agrupados hierarquicamente em uma árvore sintática.
    - Análise Semântica
      - O programa é verificado para garantir que seus componentes se combinam de forma significativa

Linguagens Compiladas

- Fases de um compilador
  - Síntese
    - Geração de Código Intermediário
      - Gera uma representação intermediária do código final.
      - O código intermediário é de fácil produção, otimização, e tradução para o código de máquina.
      - Facilita a portabilidade e a otimização do código final.
    - Otimização de Código
      - Fase que visa melhorar o código intermediário.
      - É capaz de fazer transformações de tipo, por exemplo de inteiro para real, que seja necessário.
    - Geração de Código
      - Fase final da compilação onde o código de máquina é gerado.
      - Nesta fase as instruções do código intermediário são traduzidas para código de máquina.

Linguagens Compiladas

- Fases de um compilador
  - Tabela de Símbolos
    - Estrutura de dados que contém um registro para cada identificador, com campos contendo seus atributos.
    - Permite encontrar rapidamente cada registro, armazenar e recuperar dados.
    - A tabela é preenchida e usada por todas as fases.
    - Os identificadores são inseridos na tabela durante a
    - análise léxica, mas o tipo não é conhecido no momento em que é inserido.

Linguagens Compiladas

- Fases de um compilador
  - Detecção de Erros
    - Cada fase do processo de compilação pode gerar mensagens de erro.
    - O compilador é capaz de não parar todo o processo de compilação logo que encontre o primeiro erro.
    - Os erros são agrupados e identificados para, após o processo de compilação, serem apresentados ao usuário do compilador (programador).

Linguagens Interpretadas

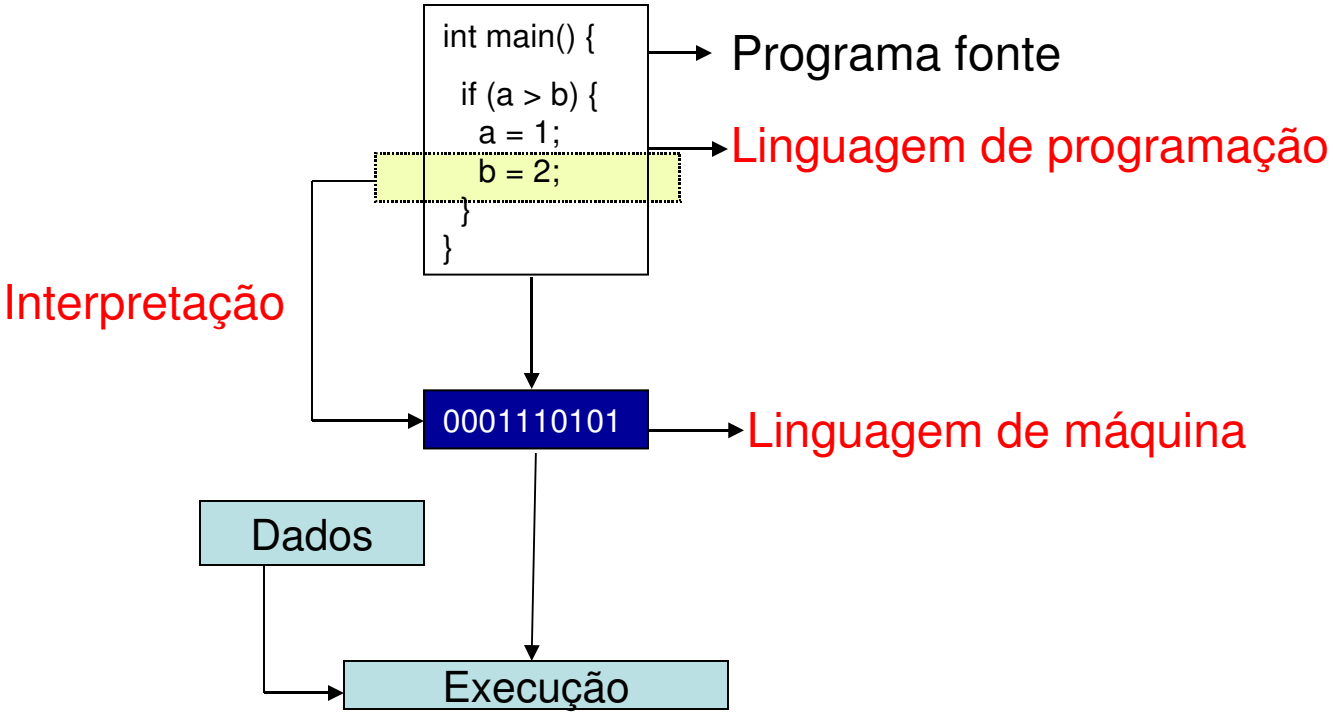
- Gera código intermediário que é executado, por um programa ou máquina virtual, sem gerar código de máquina.
- Um interpretador simula a execução de cada instrução ou comando de um programa, de forma que o seu efeito seja reproduzido corretamente, à medida que essa execução se torna necessária.

Linguagens Interpretadas

- O tempo de execução de um programa interpretado é maior que o tempo de execução de um programa compilado.
- As mensagens de erro são mais amigáveis do que as mensagens produzidas por um compilador.

Linguagens Interpretadas

- Visão geral



Linguagens Interpretadas

- Visão geral: Máquina Virtual