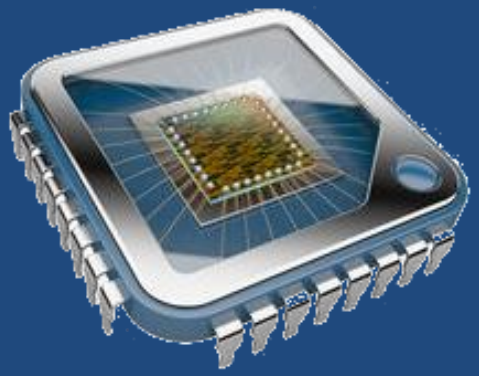


Introdução à Programação
Aula 03

Prof. Max Santana Rolemberg Farias
max.santana@univasf.edu.br
Colegiado de Engenharia de Computação

GESE
Grupo de Engenharia de Sistemas Embarcados

CEComp
Engenharia da Computação
UNIVASF

Introdução à Programação
Linguagens de Programação

- A primeira linguagem de programação foi criada por Ada Lovelace.
  - Amiga de Charles Babbagem (Calculadora mecânica)
  - Primeira programadora

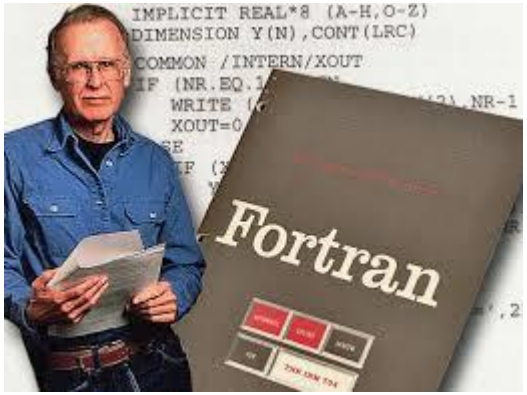
Ada Lovelace

VOCÊ SABIA ?

A linguagem de programação ADA doi batizada em homenagem a esta primeira programadora.

Introdução à Programação
Linguagens de Programação (continuação)

- A linguagem **Fortran** foi a primeira linguagem de programação de alto nível amplamente usada.
  - Foi criada na década de 1950
  - Continua sendo usada até hoje



John Backus

VOCÊ SABIA ?

O nome FORTRAN é um acrônimo da expressão *IBM Mathematical **For**mula **Tran**slation System*.

Introdução à Programação
Linguagens de Programação (continuação)

- A linguagem **Cobol** foi uma linguagem de ampla aceitação para uso comercial.
  - Foi criada na década de 1950
  - Orientada para o processamento de banco de dados.
  - Continua sendo usada até hoje, com orientação a objetos visando o .NET

Grace Hopper

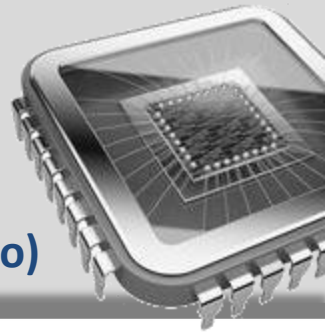
VOCÊ SABIA?

O nome COBOL é a sigla de ***Co**mmon **B**usiness **O**riented **L**anguage* (Linguagem comum orientada para os negócios).

Introdução à Programação
Linguagens de Programação (continuação)

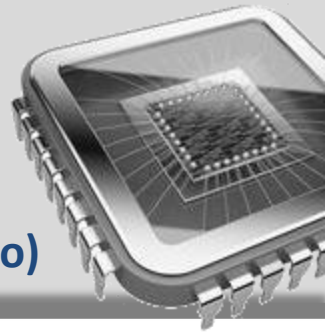
- Existem várias linguagens de programação.
  - De acordo com o índice Tiobe, as 20 mais populares são:
    - Java
    - C
    - C++
    - Python
    - C#
    - PHP
    - Visual Basic .NET
    - JavaScript
    - Perl
    - Ruby
    - Assembly
    - Visual Basic
    - Delphi/Object Pascal
    - Swift
    - Objective-C
    - MATLAB
    - Pascal
    - R
    - PL/SQL
    - COBOL

Introdução à Programação
Linguagens de Programação (continuação)



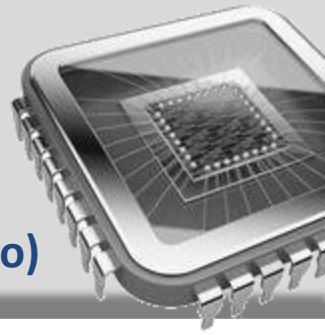
- As linguagens de programação podem ser classificadas em gerações.
  - Primeira geração
  - Segunda geração
  - Terceira geração
  - Quarta geração
  - Quinta geração

Introdução à Programação
Linguagens de Programação (continuação)



- **Linguagens de programação de 1ª geração**
  - São linguagens onde suas estrutruras de controle são aparentemente orientadas a máquina.
  - As instruções condicionais não são aninhadas e dependem fortemente de instruções de desvio como GOTO
  - Uma linguagem típica desta geração é a linguagem Fortran.

Introdução à Programação
Linguagens de Programação (continuação)



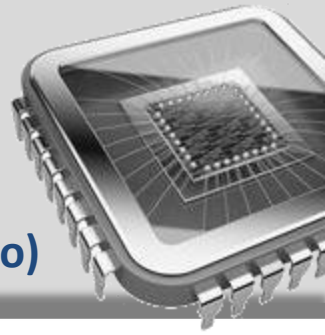
- **Linguagens de programação de 2ª geração**
  - São linguagens onde suas estrutruras de controle são estruturadas de forma a minimizar ou dispensar o uso do GOTO.
  - Uma das grandes contribuições desta geração foi sua estrutura de nomes.
    - Que permitiu melhor controle de espaços de nomes e uma eficiente alocação dinâmica de memória.
  - Uma linguagem típica desta geração é o Algol 60.

Introdução à Programação
Linguagens de Programação (continuação)

- **Linguagens de programação de 3ª geração**
  - São linguagens que dão ênfase a simplicidade e eficiência.
    - As estruturas de controle sao mais simples e eficientes.
  - Uma linguagem típica desta geração é a linguagem Pascal.

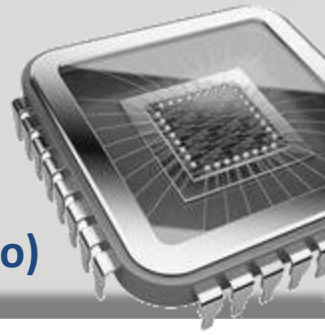
Niklaus Wirth
Criador da linguagem Pascal

Introdução à Programação
Linguagens de Programação (continuação)



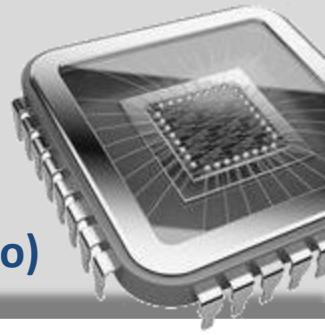
- **Linguagens de programação de 4ª geração**
  - A maioria das linguagens desta geração focam na modularização e no encapsulamento.
  - Uma linguagem típica desta geração é a linguagem ADA.

Introdução à Programação
Linguagens de Programação (continuação)



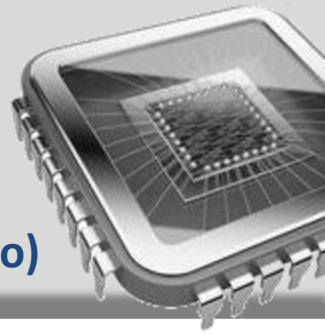
- **Linguagens de programação de 5ª geração**
  - Nesta geração, são agrupadas diversas linguagens com diversos paradigmas como:
    - Paradigma orientado a objetos
    - Paradigma funcional
    - Paradigma lógico

Introdução à Programação
Linguagens de Programação (continuação)



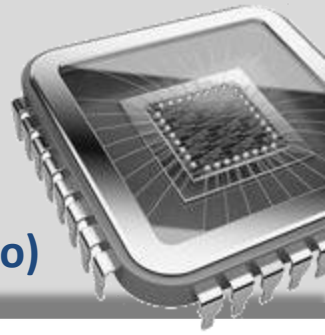
- As linguagens de programação podem ser agrupadas segundo o paradigma que seguem para abordar a sua sintaxe e semântica.
- Os paradigmas se dividem em dois grandes grupos:
  - Imperativo
  - Declarativos

Introdução à Programação
Linguagens de Programação (continuação)



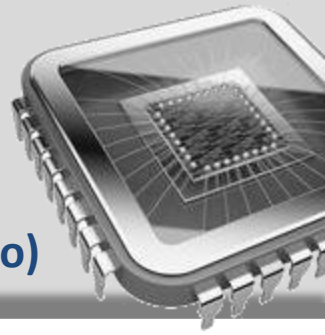
- **Paradigmas Imperativos**
  - Os paradigmas imperativos são aqueles que facilitam a computação por meio de mudanças de estado.
    - O paradigma procedural
    - O paradigma de estruturas de blocos
    - O paradigma de orientação a objetos
    - O paradigma da computação distribuída

Introdução à Programação
Linguagens de Programação (continuação)



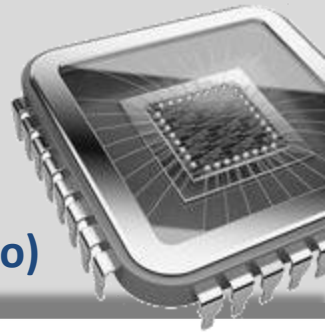
- **Paradigmas Imperativos: Procedural**
  - Neste paradigma, os programas são executados através de chamadas sucessivas a procedimentos separados.
  - Exemplos de linguagens deste paradigma são:
    - Fortran
    - Basic

Introdução à Programação
Linguagens de Programação (continuação)



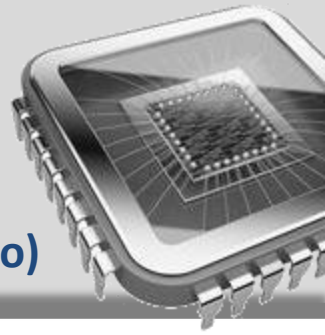
- **Paradigmas Imperativos: Estrutura de Blocos**
  - As características marcante deste paradigma são os escopos aninhados.
  - Exemplos de linguagens deste paradigmas são:
    - Algol 60
    - Pascal
    - C

Introdução à Programação
Linguagens de Programação (continuação)



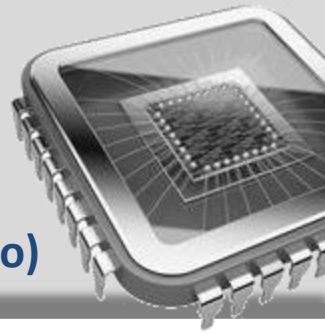
- **Paradigmas Imperativos: Orientação a Objetos**
  - Este paradigma descreve linguagens que suportam a interação entre objetos.
  - Exemplos de linguagens deste paradigmas são:
    - C++
    - Linguagem D
    - Java
    - Python
    - Ruby

Introdução à Programação
Linguagens de Programação (continuação)



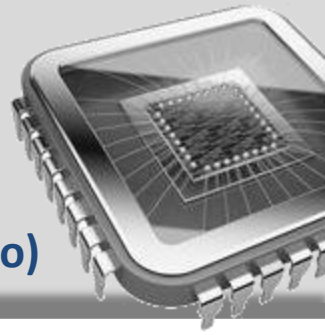
- **Paradigmas Imperativos: Computação Distribuída**
  - Este paradigma suporta que mais de uma rotina possa executar independentemente.
  - Um exemplo de linguagens deste paradigma é a linguagem Ada.

Introdução à Programação
Linguagens de Programação (continuação)



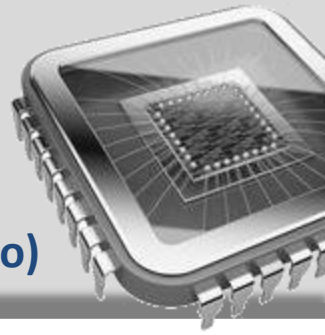
- **Paradigmas Declarativos**
  - Os paradigmas declarativos são aqueles nos quais um programa especifica uma relação ou função. Se dividem em:
    - O paradigma funcional
    - O paradigma da programação lógica

Introdução à Programação
Linguagens de Programação (continuação)



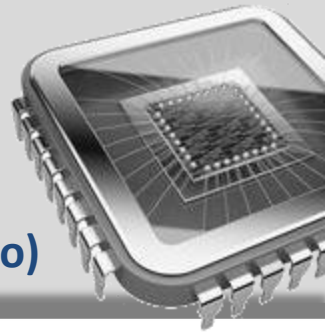
- **Paradigmas Declarativos: Funcional**
  - Na programação funcional, o mapeamento entre os valores de entrada e saída são alcançados mais diretamente.
  - Um programa é uma função tipicamente constituída de outras funções mais simples
  - Exemplos de linguagens deste paradigmas são:
    - Lisp
    - Scheme
    - Haskell

Introdução à Programação
Linguagens de Programação (continuação)

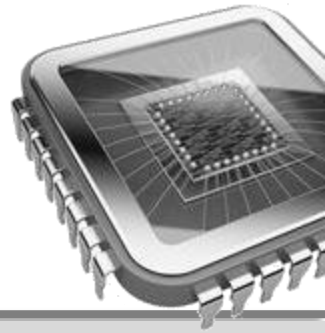


- **Paradigmas Declarativos: Programação Lógica**
  - Este paradigma se baseia na noção de que um programa implementa uma relação ao invés de um mapeamento.
  - Exemplos de linguagens deste paradigma são:
    - Prolog
    - Gödel

Introdução à Programação
Linguagens de Programação (continuação)

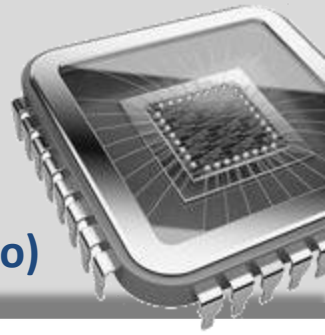


- As linguagens de programação podem ser definidas quanto a estrutura de tipos.
  - Fracamente tipada, onde o tipo da variável muda dinamicamente conforme a situação
    - PHP e Smalltalk
  - Fortemente tipada, onde o tipo da variável, uma vez atribuído, se mantém o mesmo até ser descartada da memória.
    - Java e Ruby
  - Dinamicamente tipada, onde o tipo da variável é definido em tempo de execução.
    - SNOBOL, APL, Awk, Perl e Python
  - Estaticamente tipada, onde o tipo da variável é definido em tempo de compilação
    - Java e C



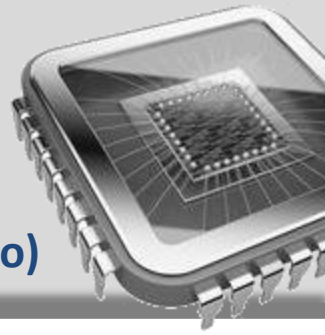
COMO OS SISTEMAS COMPUTACIONAIS ENTENDEM AS LINGUAGENS DE PROGRAMAÇÃO?

Introdução à Programação
Linguagens de Programação (continuação)



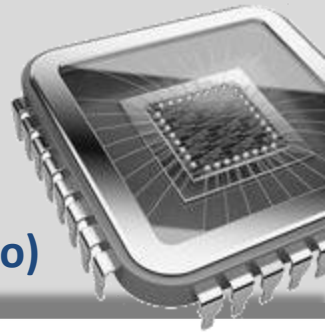
- As linguagens de programação podem ser convertidas, ou traduzidas, em códigos de máquina por **compiladores** ou **interpretadores**.
  - Em ambos os processos ocorre a tradução do código fonte para código de máquina

Introdução à Programação
Linguagens de Programação (continuação)



- **Linguagens compilada**
  - Se a linguagem traduz todo o texto do programa para só depois executar, então diz-se que o programa foi compilado.
    - O mecanismo utilizado para a tradução é um compilador
  - A versão compilada do programa pode ser executado um número indefinido de vezes sem que seja necessária nova compilação.
    - Compensa o tempo gasto na compilação

Introdução à Programação
Linguagens de Programação (continuação)



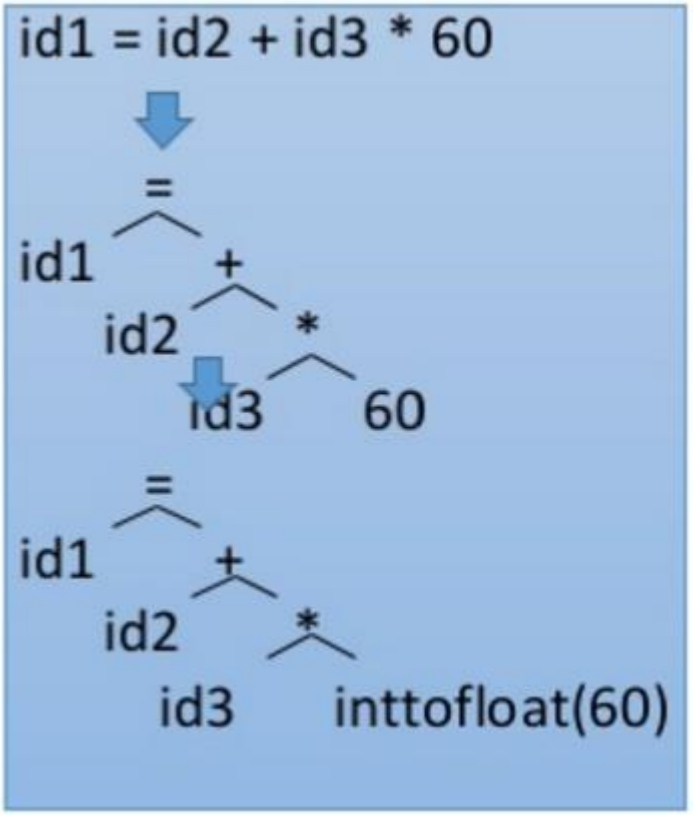
- **Linguagens Compilada: Análise léxica**
  - Reconhece as sequências de símbolos (cadeia de caracteres) que representam uma unidade (sentença).
  - Com o objetivo de separar os símbolos adequadamente.
  - As sentenças reconhecidas são:
    - Identificadores, palavras-chaves, variável, constantes, operadores, delimitadores e palavras de instrução (while, for, etc)
  - Cada símbolo isolado pelo analisador léxico é chamado de **token**.

pos = ini + val * 60

id1 = id2 + id3 * 60

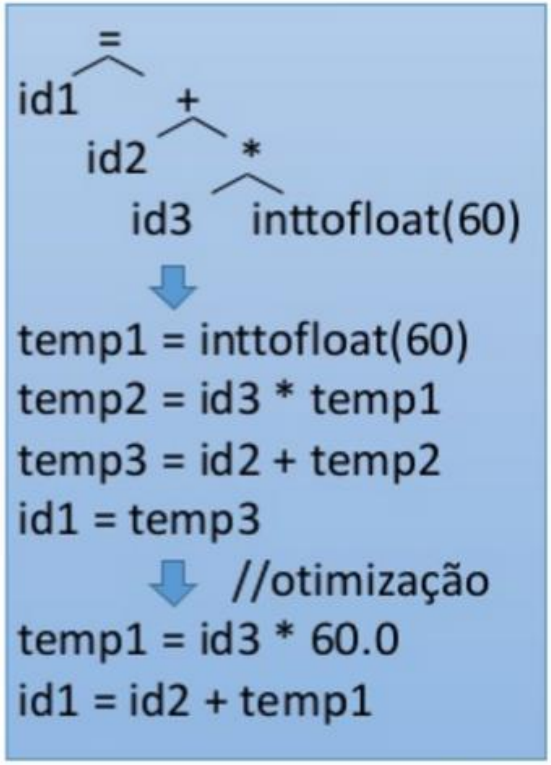
Introdução à Programação
Linguagens de Programação (continuação)

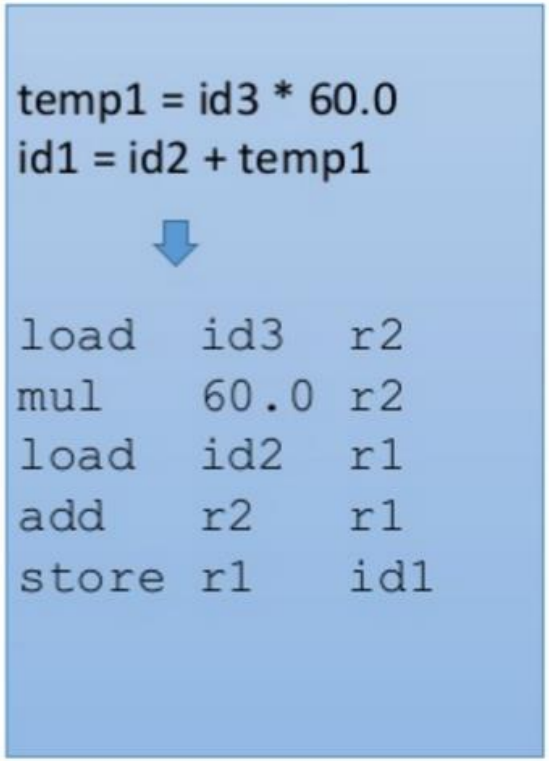
- **Linguagens Compilada: Análise sintática**
  - O analisador sintático recebe do analisador léxico uma sequência de símbolos (tokens) e determina se estão dispostos conforme especifica a gramática da linguagem.
    - Identifica a estrutura gramatical do programa e o papel de cada componente.
  - É construída uma árvores sintática (binária) e uma tabela de símbolos que representam as variáveis.



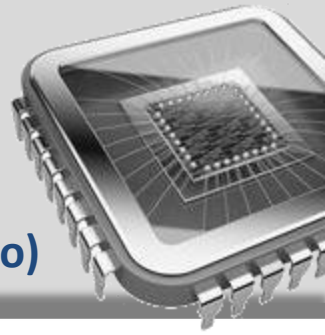
Introdução à Programação
Linguagens de Programação (continuação)

- **Linguagens Compilada: Geração de código**
  - Processo de construir instruções da linguagem de máquina (em *assembly*)
    - Simulam as instruções reconhecidas na análise sintática.





Introdução à Programação
Linguagens de Programação (continuação)

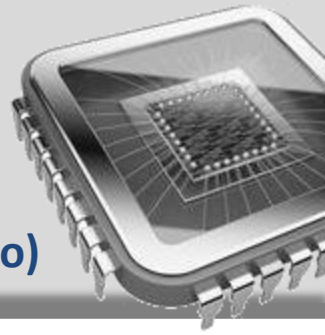


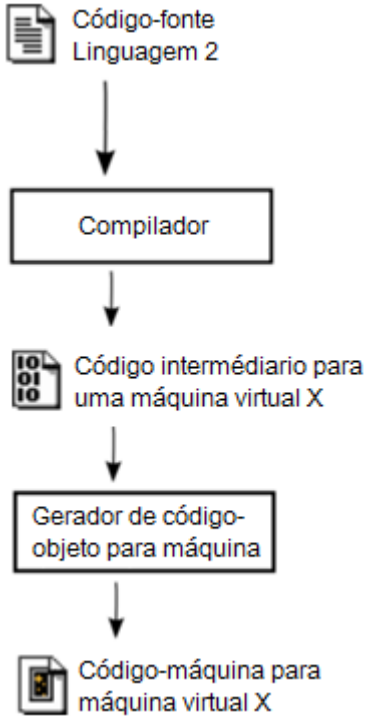
- **Linguagens Compiladas**

- Execução rápida, o código já está traduzido.
- Os executáveis são pequenos.

Só roda na aquitetura em que for compilado.

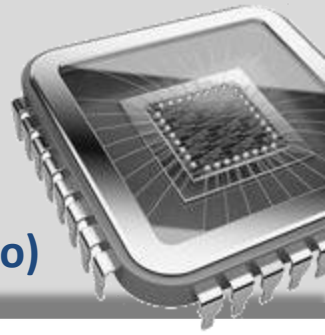
Introdução à Programação
Linguagens de Programação (continuação)





- **Linguagens interpretada**
  - Se a linguagem traduz todo o texto do programa para uma linguagem intermediária (bytecode) que só depois será traduzida para linguagem de máquina que será executada, então diz-se que o programa foi interpretado.
    - Utiliza uma máquina virtual para executar o programa
  - A tradução dos comandos da linguagem intermédia para linguagem de máquina é feita em tempo de execução.
    - A máquina virtual reconhece toda a arquitetura do hardware e traduz de acordo com o equipamento.

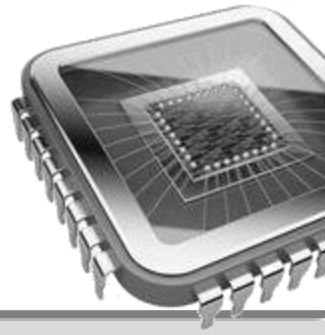
Introdução à Programação
Linguagens de Programação (continuação)



- **Linguagens Interpretada**

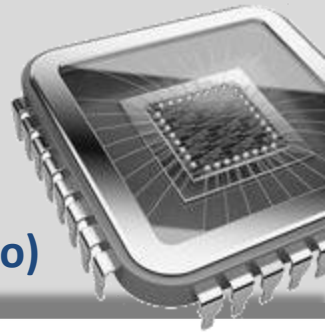
Independe de arquitetura (desde que suporte a máquina virtual.

Desempenho inferior à linguagem compilada por causa da tradução, mas nada alarmante.
Consome bastante hardware.



COMO ESPECIFICAR O CONJUNTO DE REGRAS DE UMA LINGUAGEM DE PROGRAMAÇÃO?

Introdução à Programação
Linguagens de Programação (continuação)



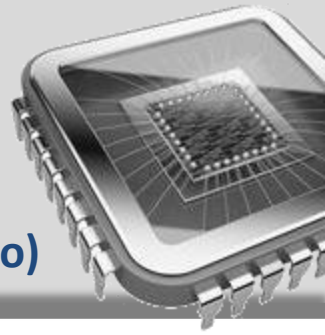
FORMAL vs INFORMAL

- As linguagens de programa têm uma especificação formal para especificar de forma estruturada seus conjuntos de regras.

Os métodos informais por sua vez denotam a inexistência das características associadas ao formalismo.

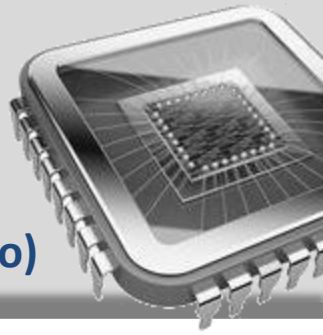
- Apresentam frases de significado duvidoso
- Geração de manuais imprecisos
- Não existe técnicas que auxiliem a implementação

Introdução à Programação
Linguagens de Programação (continuação)



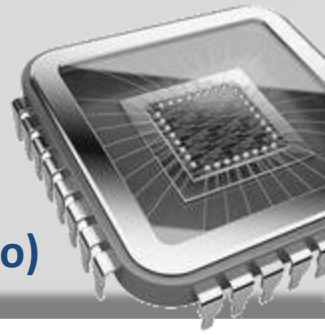
- **Especificação Formal**
  - Precisa e não ambígua
  - Automatizável
  - Permite construir provadores de propriedades de programas.
  - Com a utilização de especificação formal temos:
    - Especificação léxica
    - Especificação sintaxe
    - Especificação semântica

Introdução à Programação
Linguagens de Programação (continuação)



- **Especificação Formal: Sintática**
  - Na especificação sintática são especificadas as regras para determinar se uma dada cadeia de entrada pertence ou não à linguagem definida por uma gramática.
  - Produz uma estrutura comumente denominada *Abstract Syntax Tree* (AST).
  - A Backus-Naur Form (BNF) foi criada para a descrição sintática. (notação mais utilizada)

Introdução à Programação
Linguagens de Programação (continuação)



Especificação Léxica em BNF

<operador> ::= + | - | * | / | = | != | < | > | <= | >=

<identificador> ::= **<identificador> <letra>** | **<identificador> <dígito>** | **<letra>**

<número> ::= **<dígito> <número>** | **<dígito>**

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letra> ::= a | b | c | d | e | f | g | h | i | j | l | m | n | o | p | q | r | s | t | u | v | x | z

Especificação Sintática em BNF

<programas> :: = { **<comandos>** }

<comandos> ::= **<comando> <comandos>** | **<comando>**

<comando> ::= **<atribuição>** | **<condicional>** | **<loop>**

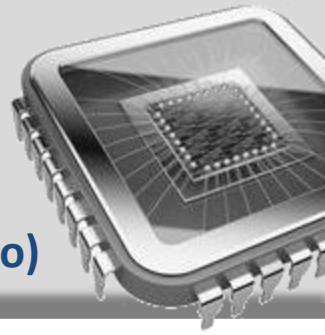
<atribuição> ::= **<identificador>** = **<expressão>**

<condicional> ::= if **<expressão>** { **<comandos>** } | if **<expressão>** { **<comandos>** } else { **<comandos>** }

<loop> ::= while **<expressão>** { **<comandos>** }

<expressão> ::= **<identificador>** | **<número>** | (**<expressão>**) **<expressão> <operador> <expressão>**

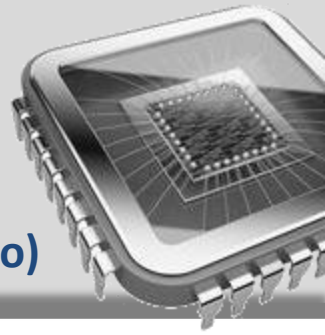
Introdução à Programação
Linguagens de Programação (continuação)



- **Especificação Formal: Semântica**
  - Descreve o significados das estruturas do programa expressos na sua sintaxe
  - A semântica pode ser:
    - Semântica estática: Descreve as características de um programa válido
    - Semântica dinâmica: Descreve os resultados da execução do programa

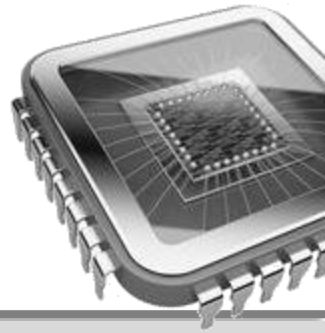
Não existe ainda um formalismo aceito globalmente para descrever a semântica da linguagem.

Introdução à Programação
Linguagens de Programação (continuação)



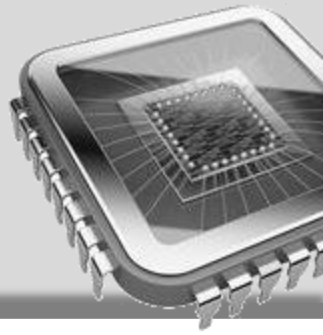
- Exemplos de formalismos semânticos
  - Semântica operacional Estrutural
  - Máquina de Estado Abstrato
  - Semântica Denotacional
  - Semântica de Ações

Mais detalhes, só na disciplina de compiladores.



POR QUE PROGRAMAR?

Introdução à Programação
Por que programar



É útil

- Programação com uma ferramenta.
- Pode ser aplicado a quase qualquer atividade
  - Arte
  - Ciência
  - Filosofia
  - Entretendimento

É divertido

- Melhor que usar programas prontos
- Fazer programas é como resolver quebra-cabeças
- Programação é como uma arte