PCS 3216
Sistemas de Programação

João José Neto

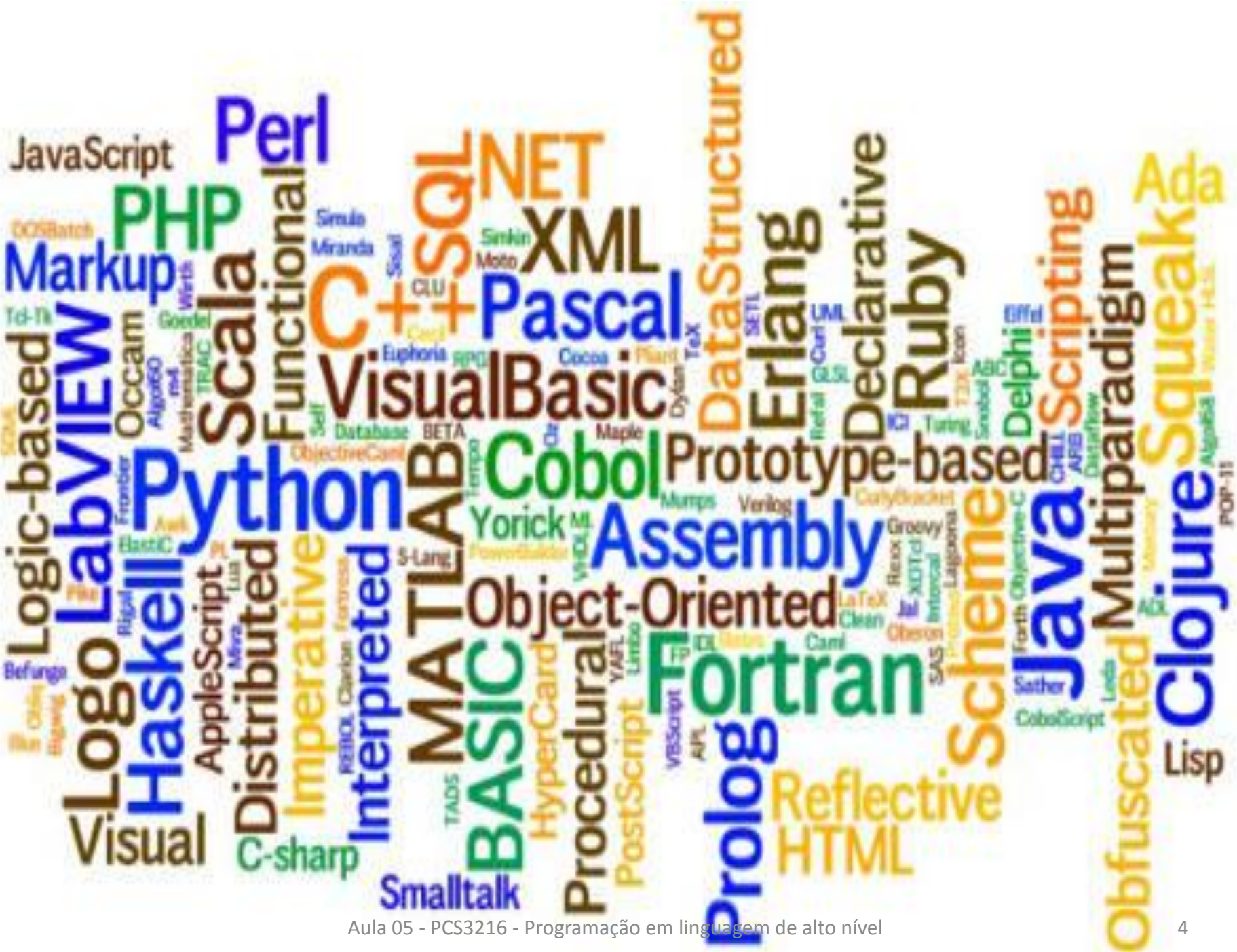
Aula 5 – Linguagem de alto nível e Compiladores

Linguagens de alto nível

- A programação em linguagens de baixo nível (linguagem de máquina binária ou de outras bases numéricas, e linguagens simbólicas) força o programador a trabalhar em um nível extremo de proximidade com a máquina.
- Isso em geral se mostra desconfortável, difícil, contraproducente e muito sujeito a falhas humanas, apesar da disponibilidade de várias ferramentas, que foram criadas e extensivamente utilizadas em suporte a essa tarefa.

Elevação do nível de abstração

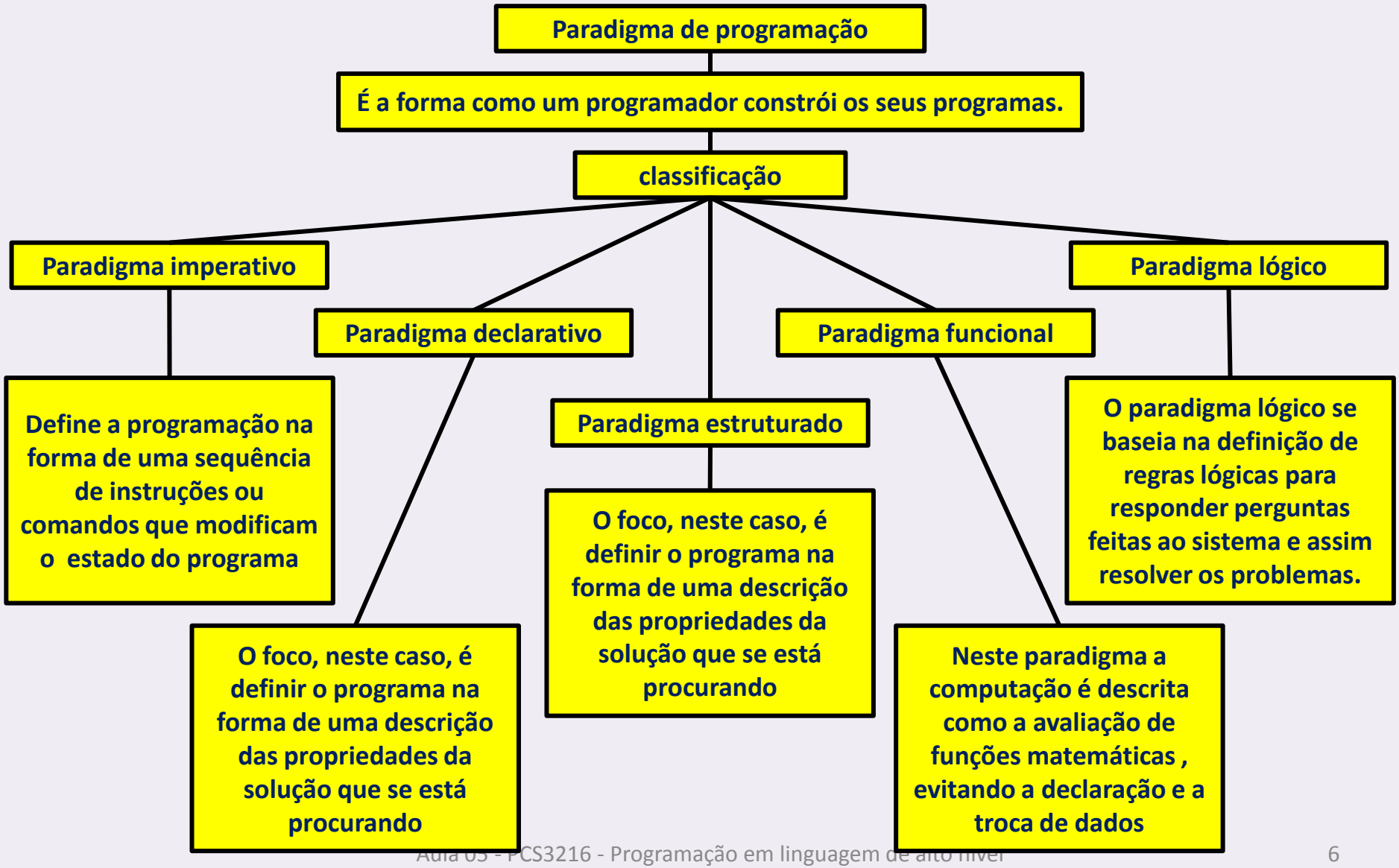
- Superou-se esse problema com a introdução de notações conhecidas como linguagens de alto nível, que elevaram o nível de abstração exigido para os trabalhos na área da programação.
- Nos dias de hoje os programadores têm à disposição uma profusão de linguagens de alto nível dos mais variados tipos, estilos e características, as quais, para a maior parte das situações normais, tendem a dispensar o uso de diversos dos recursos de programação próprios para serem utilizados com as linguagens de baixo nível.


JavaScript
Perl
PHP
Markup
Scala
Functional
C++
SQL
NET
XML
Pascal
VisualBasic
DataStructured
Erlang
Declarative
Ruby
Ada
Scripting
Squeak
Logic-based
LabVIEW
Occam
Python
MATLAB
Cobol
Prototype-based
Assembly
Yorick
Object-Oriented
Fortran
Java
Multiparadigm
Clojure
Haskell
Logo
Distributed
Imperative
Interpreted
AppleScript
BASIC
HyperCard
Procedural
PostScript
Prolog
Scheme
Reflective
HTML
Visual
C-sharp
Smalltalk
Lisp
Obfuscated

As linguagens de alto nível

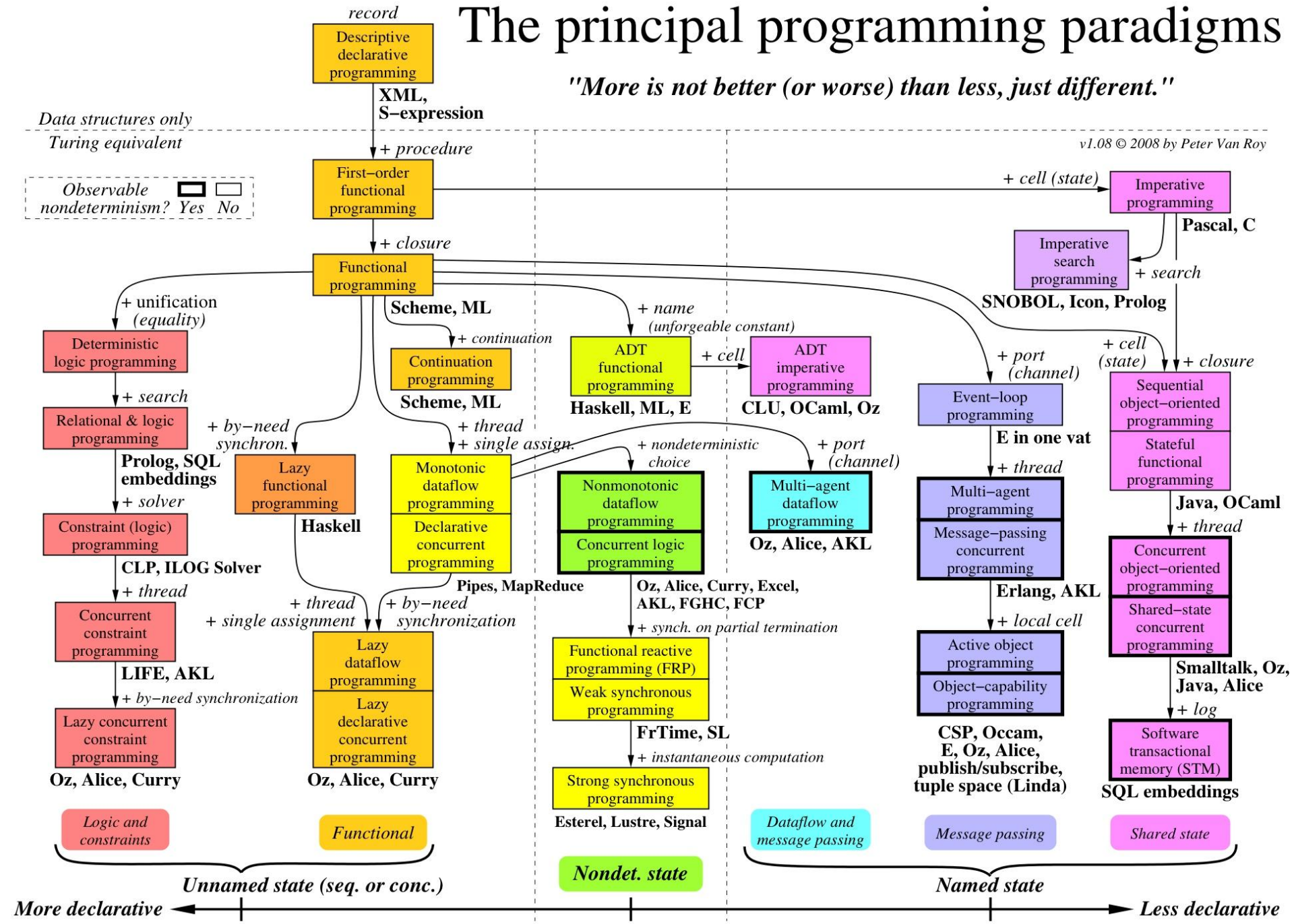
- Exprimem os programas de maneira mais **independente de máquina**
- Uso de **diagramas** e de **pseudo-código**
- Uso de **notações** mais próximas da linguagem **humana**
- Criação de **linguagens artificiais** próprias para a programação (**linguagens de alto nível**)

Paradigmas de programação

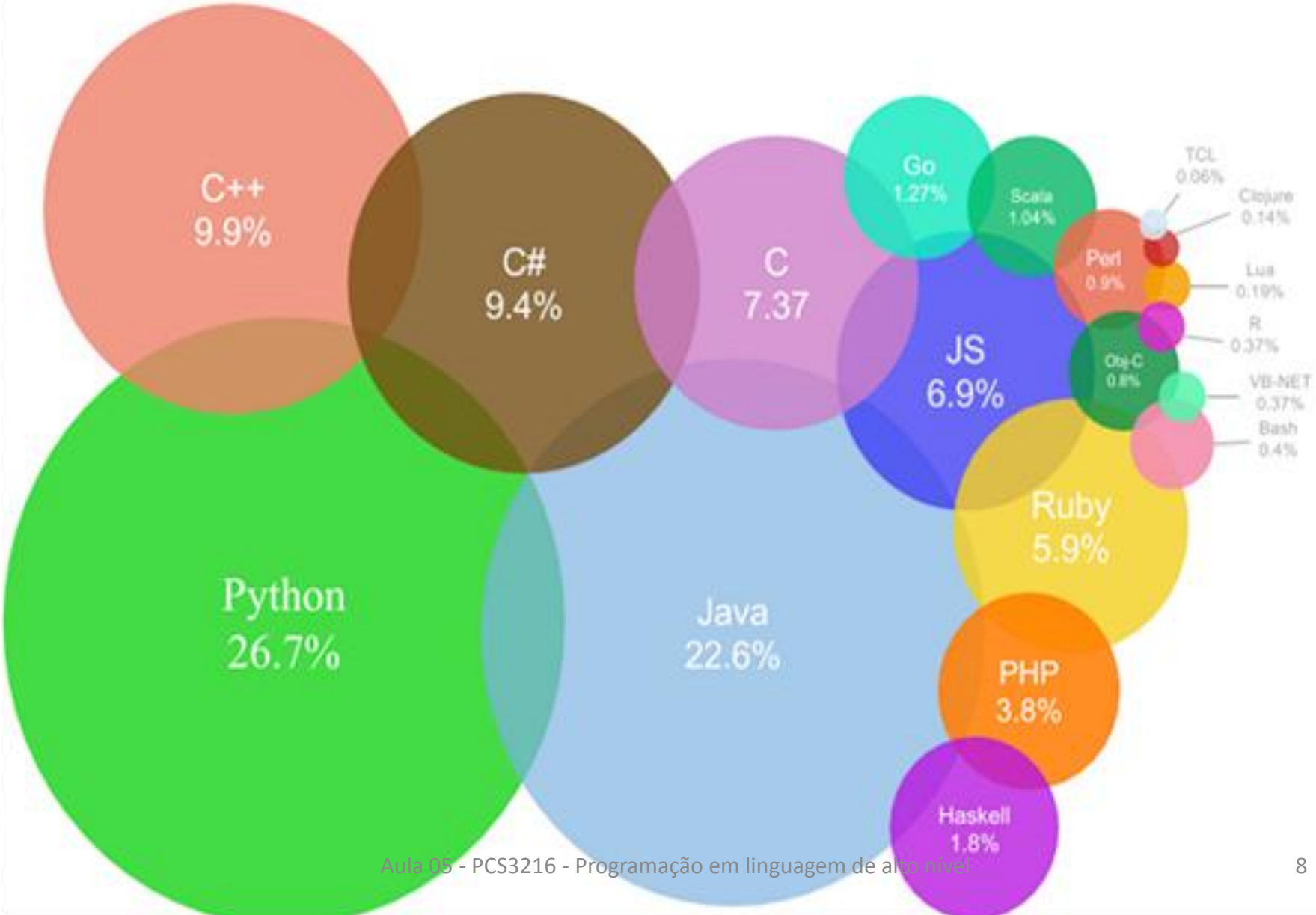


The principal programming paradigms

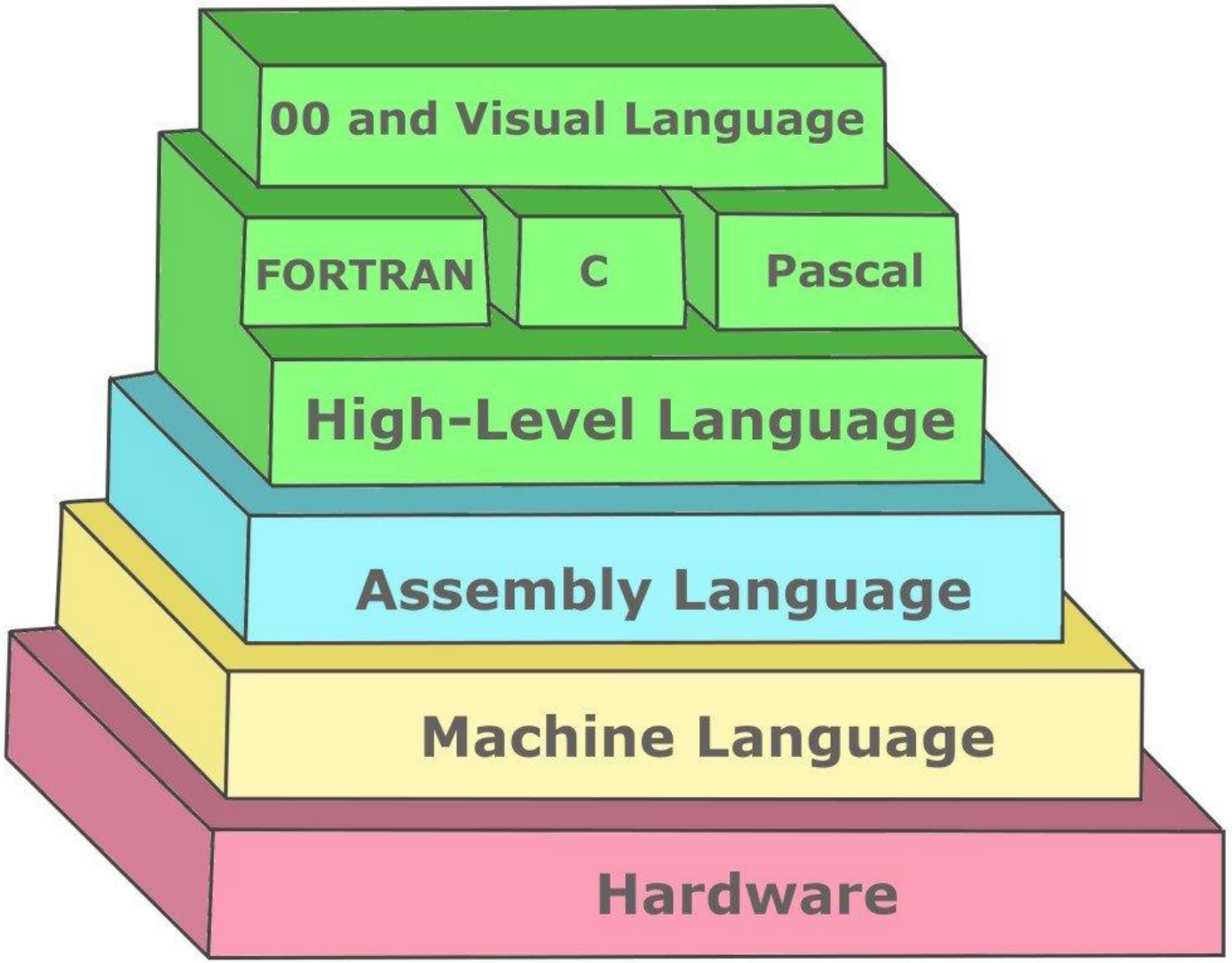
"More is not better (or worse) than less, just different."

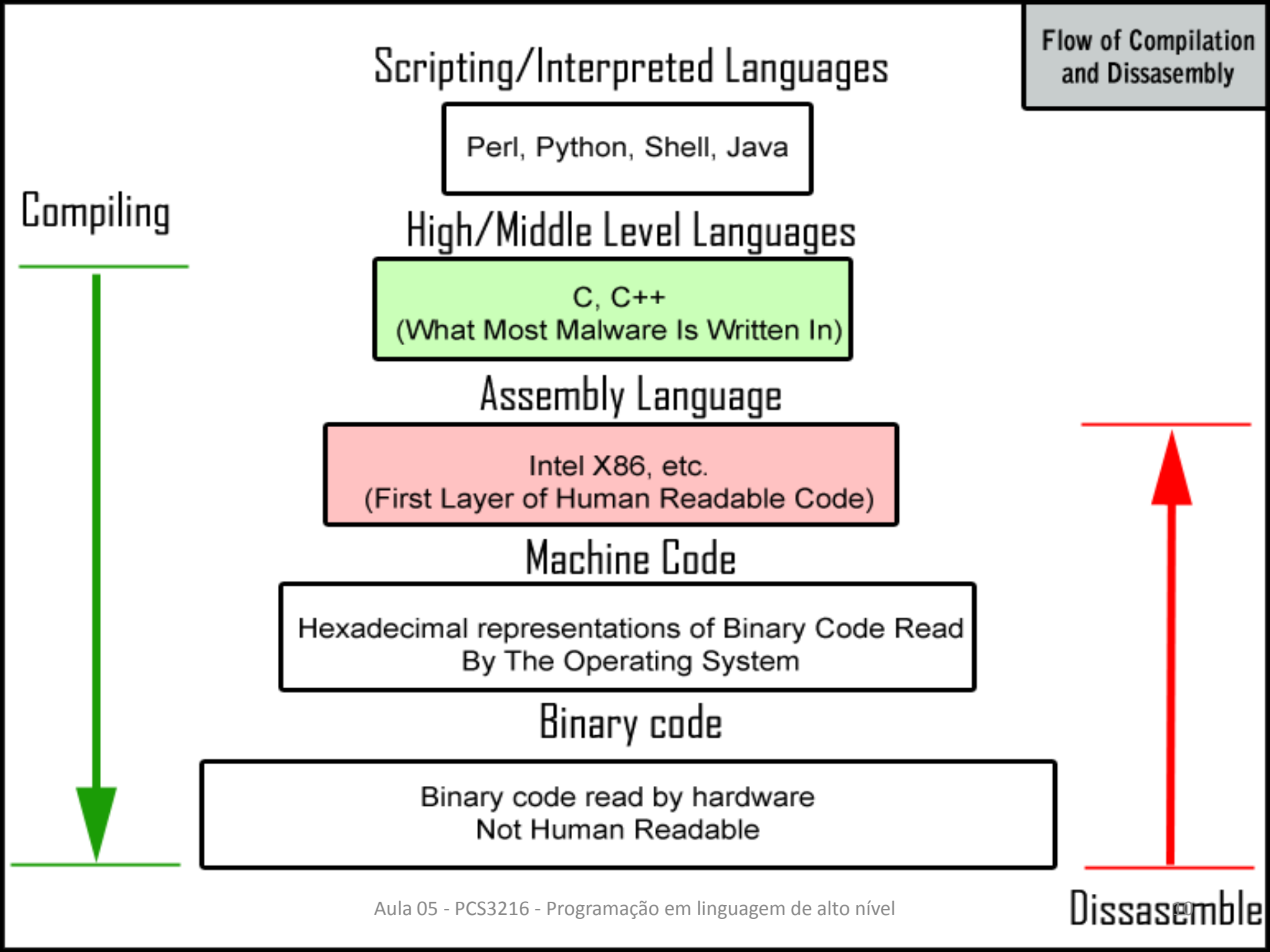


Linguagens mais populares (2016)



A escalada do nível de abstração




Flow of Compilation and Dissasembly
Scripting/Interpreted Languages
Perl, Python, Shell, Java
Compiling
High/Middle Level Languages
C, C++
(What Most Malware Is Written In)
Assembly Language
Intel X86, etc.
(First Layer of Human Readable Code)
Machine Code
Hexadecimal representations of Binary Code Read By The Operating System
Binary code
Binary code read by hardware
Not Human Readable
Dissasemble

Fórmulas

- As **primeiras linguagens** de alto nível colocavam seu foco na **codificação de fórmulas** matemáticas
- A uma das primeiras linguagens de alto nível que surgiram foi dado o nome de

  FORTRAN = **FOR**mula **TRAN**slation
- Entre outras das **linguagens** mais **antigas** citam-se:

  o Basic, o Cobol e o Lisp
- Incrivelmente, todas essas linguagens antigas continuam a ser **intensamente utilizadas** para finalidades práticas **até os dias de hoje.**

Aspecto de um antigo programa FORTRAN

```
C       CALCULATE STATISTICS ON DATA FROM LOW SPEED READER
        SUM=0
        SUMSQ=0
        TYPE 100
100     FORMAT("ENTER THE NUMBER OF VALUES TO CACULATE STATISTICS ON",/)
        ACCEPT 10,N
10      FORMAT(I)
        DO 200 I=1,N
        READ 1,110,V
110     FORMAT(E)
        SUM=SUM + V
        SUMSQ=SUMSQ + V*V
        TYPE 120,I,V
120     FORMAT("VALUE",I,"IS",E,/)
200     CONTINUE
        SAMP=N
        AVRG=SUM/SAMP
        STD=SQTF(SUMSQ/SAMP - AVRG**2)
        TYPE 300,N,AVRG,STD
300     FORMAT("NUMBER OF VALUES",I,"MEAN",E,"STANDARD DEVIATION",E,/)
        END
```

Aspecto de um antigo programa BASIC

```
 10 INPUT "What is your name: "; U$
 20 PRINT "Hello "; U$
 25 REM
 30 INPUT "How many stars do you want: "; N
 35 S$ = ""
 40 FOR I = 1 TO N
 50 S$ = S$ + "*"
 55 NEXT I
 60 PRINT S$
 65 REM
 70 INPUT "Do you want more stars? "; A$
 80 IF LEN(A$) = 0 THEN GOTO 70
 90 A$ = LEFT$(A$, 1)
100 IF (A$ = "Y") OR (A$ = "y") THEN GOTO 30
110 PRINT "Goodbye ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
150 PRINT
```

Aspecto de um antigo programa COBOL

```
IDENTIFICATION DIVISION.

PROGRAM-ID. COBOL-DEMO.
AUTHOR.     M. A. COVINGTON.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC.

DATA DIVISION.

WORKING-STORAGE SECTION.
77  SUM    PICTURE IS S999999, USAGE IS COMPUTATIONAL.
77  X      PICTURE IS S999999, USAGE IS COMPUTATIONAL.

PROCEDURE DIVISION.

START-UP.
    MOVE 0 TO SUM.
GET-A-NUMBER.
    DISPLAY "TYPE A NUMBER:" UPON CONSOLE.
    ACCEPT X FROM CONSOLE.
    IF X IS EQUAL TO 0 GO TO FINISH.
    ADD X TO SUM.
    GO TO GET-A-NUMBER.
FINISH.
    DISPLAY SUM UPON CONSOLE.
    STOP RUN.
```

Aspecto de um antigo programa Lisp

```
(DEFINE (DRIVER)
        (DRIVER-LOOP <THE-PRIMITIVE-PROCEDURES> (PRINT '|LITHP ITH LITHTENING|)))

(DEFINE (DRIVER-LOOP PROCEDURES HUNOZ)
        (DRIVER-LOOP-1 PROCEDURES (READ)))

(DEFINE (DRIVER-LOOP-1 PROCEDURES FORM)
        (COND ((ATOM FORM)
               (DRIVER-LOOP PROCEDURES (PRINT (EVAL FORM '() PROCEDURES))))
              ((EQ (CAR FORM) 'DEFINE)
               (DRIVER-LOOP (BIND (LIST (CAADR FORM))
                                  (LIST (LIST (CDADR FORM) (CADDR FORM)))
                                  PROCEDURES)
                            (PRINT (CAADR FORM))))
              (T (DRIVER-LOOP PROCEDURES (PRINT (EVAL FORM '() PROCEDURES))))))
```

Figure 1
Top Level Driver Loop for a Recursion Equations Interpreter

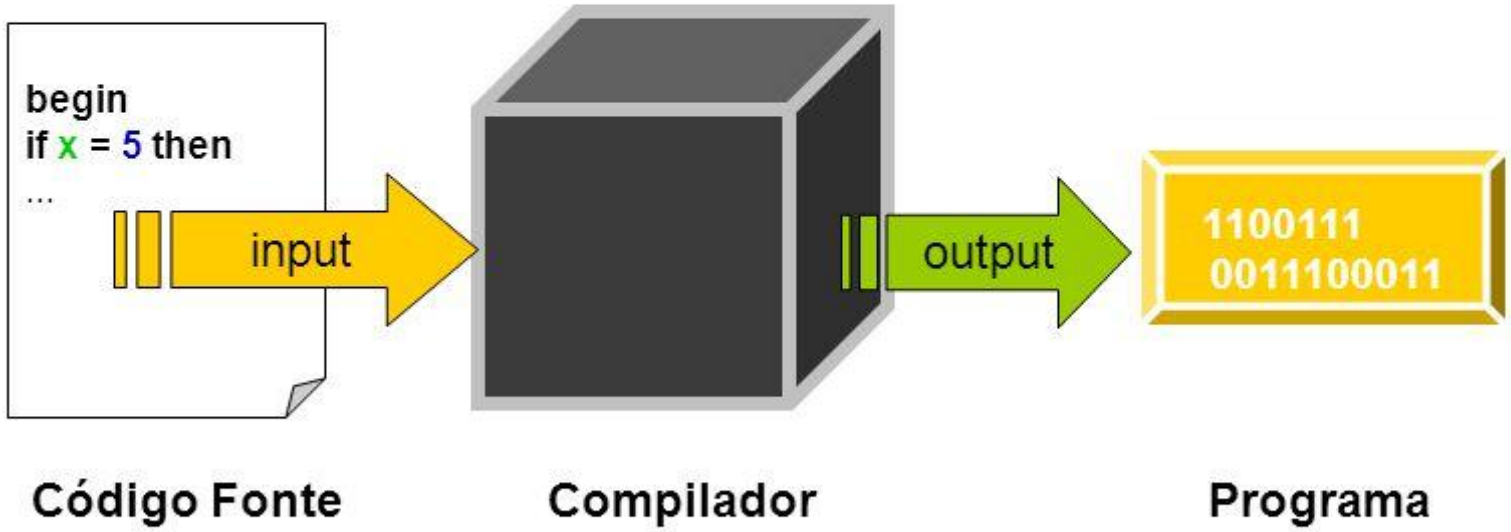
Outros comandos

- Costumam representar **simplificações** de certas construções **sintáticas** das linguagens naturais:
  - Comandos **condicionais** – if
  - Comandos de **desvio** – go to
  - Comandos **iterativos** – while, for
  - Comandos de **múltipla escolha** – case
  - Comandos de **entrada e saída** – read, print
  - Comandos de **chamada de subrotina** – call

Tradução dos programas escritos em linguagem de alto nível

- A obtenção de **códigos executáveis** a partir das **linguagens de alto nível** é complexa.
  - Apesar de toda a simplificação
  - Apesar do pequeno número de variações permitidas
- Isso exige processos **automáticos** para a sua **tradução**
- **Compiladores** são os programas que executam essa tarefa

Processo (simplificadíssimo) de compilação



Compiladores

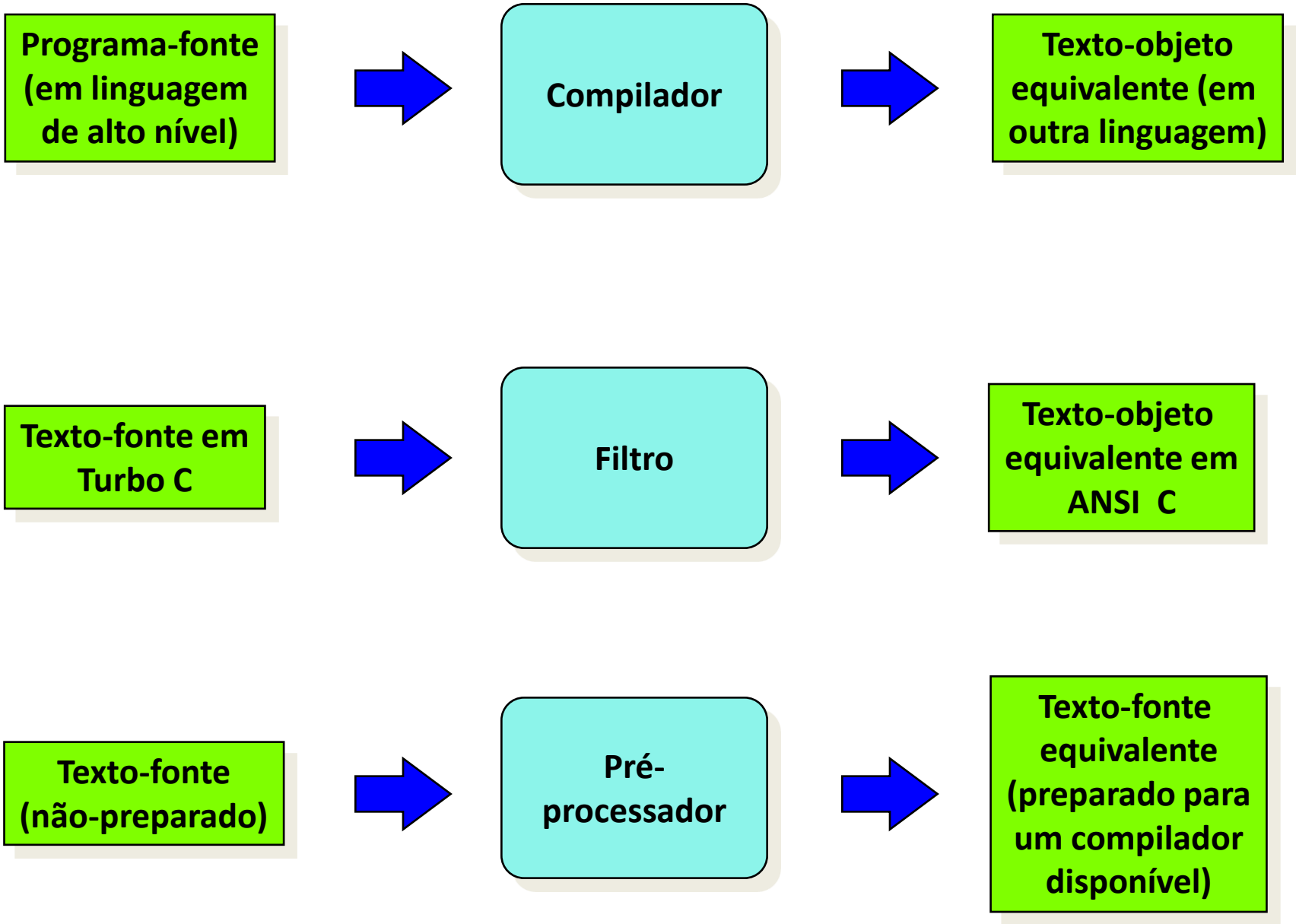
- **Traduzem** para código executável programas expressos em linguagem de alto nível (processo conceitualmente **similar ao de montagem**, embora muito mais complexo)
- Recebem como **entrada** o programa-**fonte**, em **linguagem de alto nível**
- Geram como **saída** um programa equivalente em **linguagem de máquina** (em geral **relocável**)

Interpretadores

- Uma **alternativa** para o processamento de programas em linguagem de alto nível é a utilização de **interpretadores**
- Esses programas de sistema **percorrem o programa** a executar, analisando-o e decidindo o que fazer **passo a passo**.
- Não geram código de máquina, mas **simulam a execução** direta **dos comandos** do programa.

Conceitos Básicos

- *Compiladores* e *Interpretadores* dão ao usuário do computador acesso à utilização prática de linguagens de alto nível
- *Compiladores* geralmente efetuam a tradução de linguagens de alto nível para linguagem de máquina
- *Filtros* traduzem textos escritos em uma linguagem para outros equivalentes, em linguagem semelhante
- *Pré-processadores* preparam programas, adaptando seu formato, de forma que possam ser compilados
- *Interpretadores* executam diretamente os programas em linguagem de alto nível, sem efetuar traduções


Programa-fonte (em linguagem de alto nível)
Compilador
Texto-objeto equivalente (em outra linguagem)
Texto-fonte em Turbo C
Filtro
Texto-objeto equivalente em ANSI C
Texto-fonte (não-preparado)
Pré-processador
Texto-fonte equivalente (preparado para um compilador disponível)

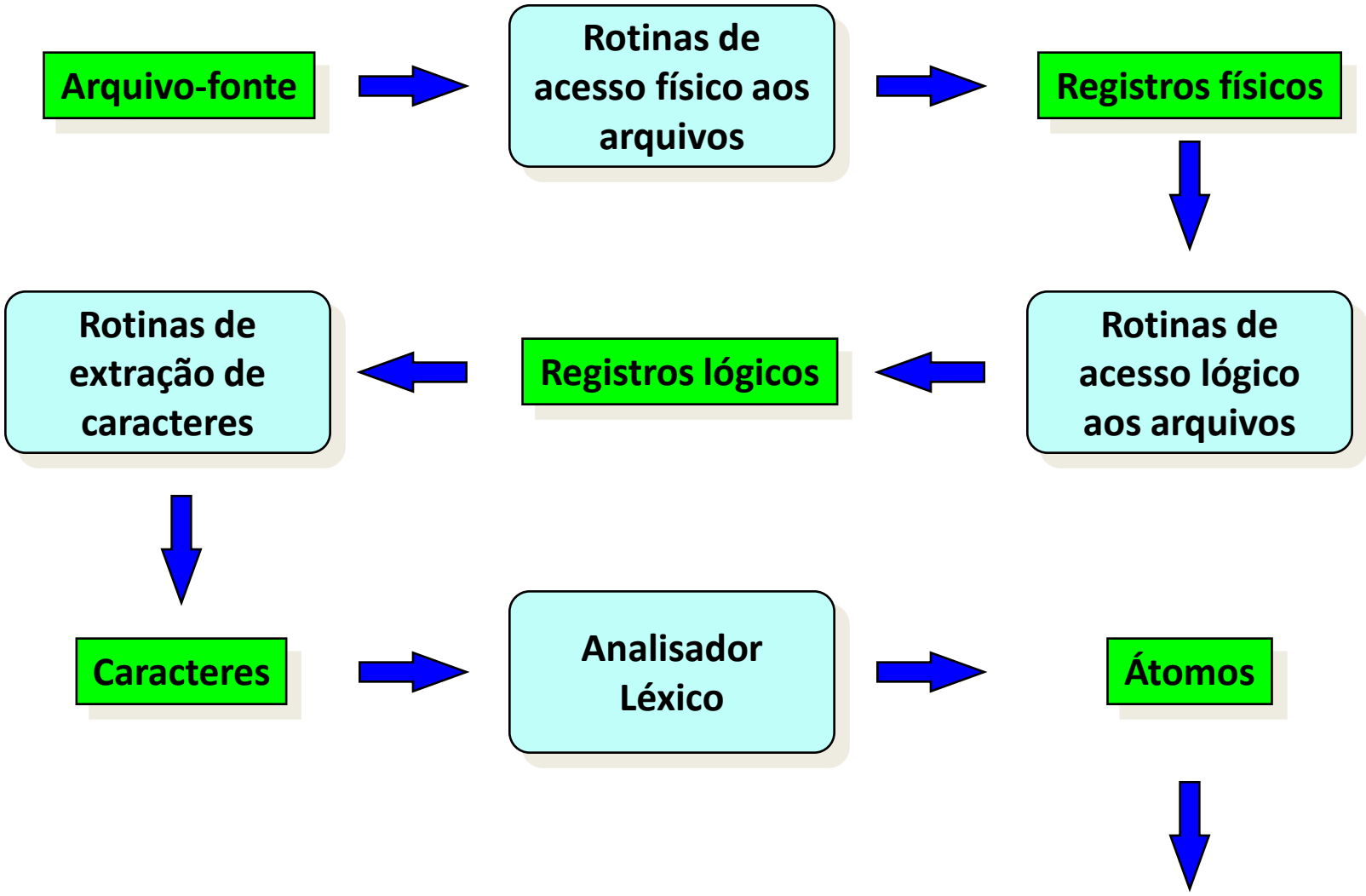
Atividades auxiliares dos compiladores

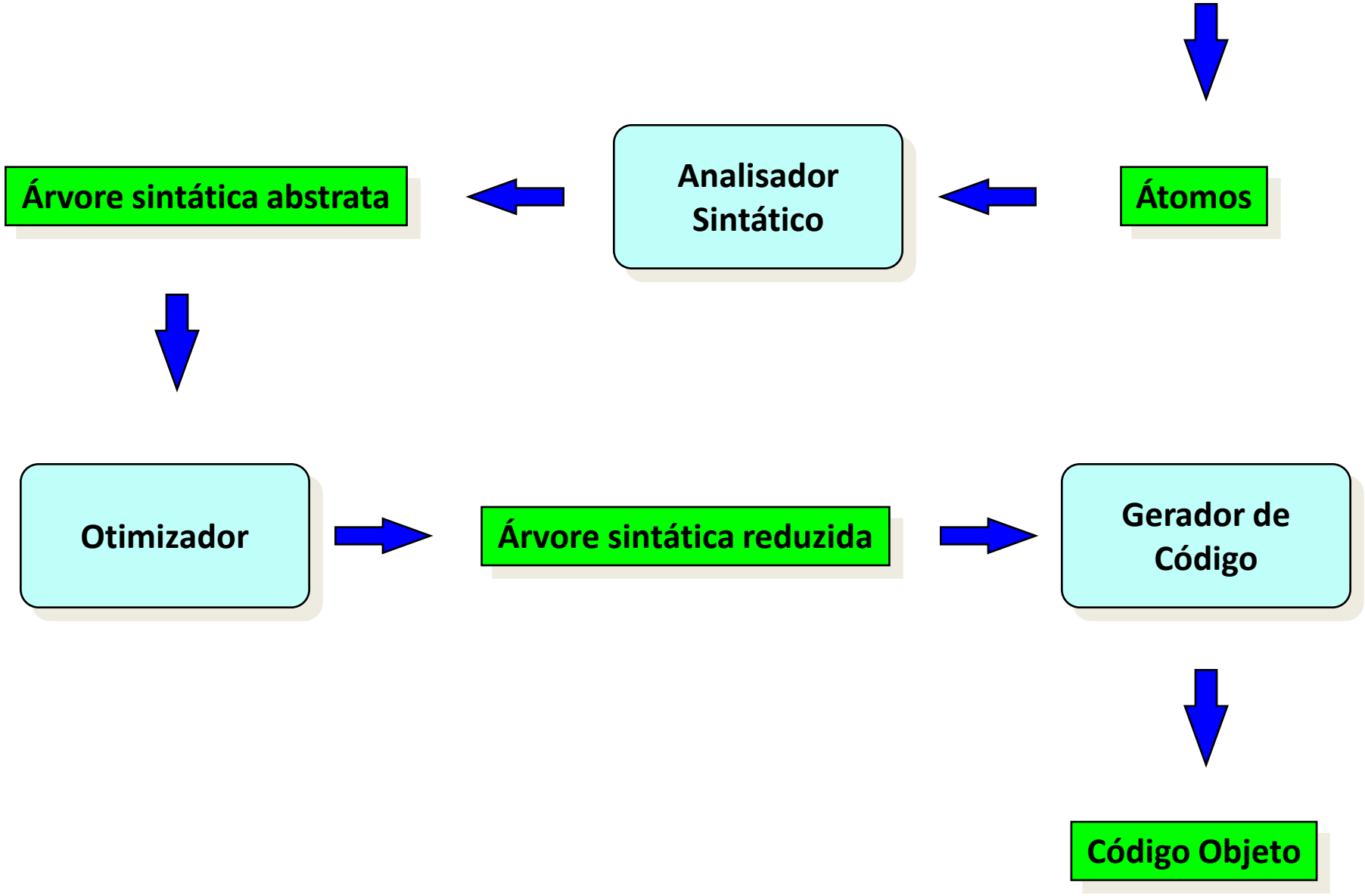
- *Detecção* de erros de sintaxe no programa
- *Recuperação* dos erros encontrados no programa-fonte
- Emissão de *mensagens de erros* detectados
- Fornecimento de recursos de *documentação*
- Fornecimento de *comandos de controle* de compilação

Formalização da sintaxe

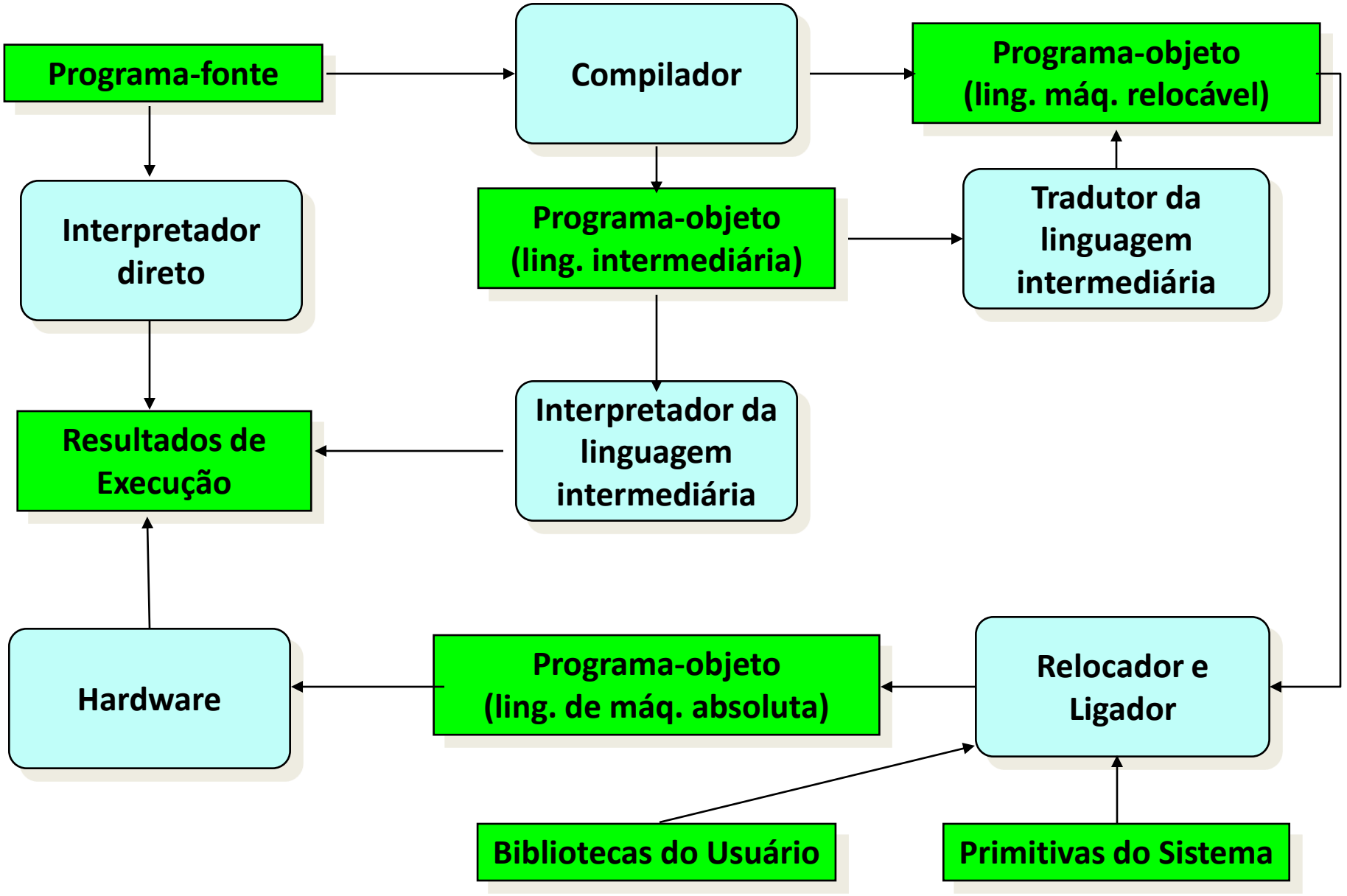
- *Descrições formais* - completas, não-ambíguas
- Inviáveis na prática para linguagens naturais
- Descrições formais para linguagens restritas:
  - generativas (gramáticas)
  - cognitivas (reconhecedores)
- *Gramáticas* representam leis de formação das linguagens
- *Reconhecedores* são dispositivos de aceitação das mesmas
- *Meta-linguagens* são linguagens usadas para a descrição formal de outras linguagens
- *BNF* (Backus-Naur Form) é até hoje uma das meta-linguagens mais populares em uso

Estrutura lógica dos compiladores




Átomos
Analisador Sintático
Árvore sintática abstrata
Otimizador
Árvore sintática reduzida
Gerador de Código
Código Objeto

Processo de Execução



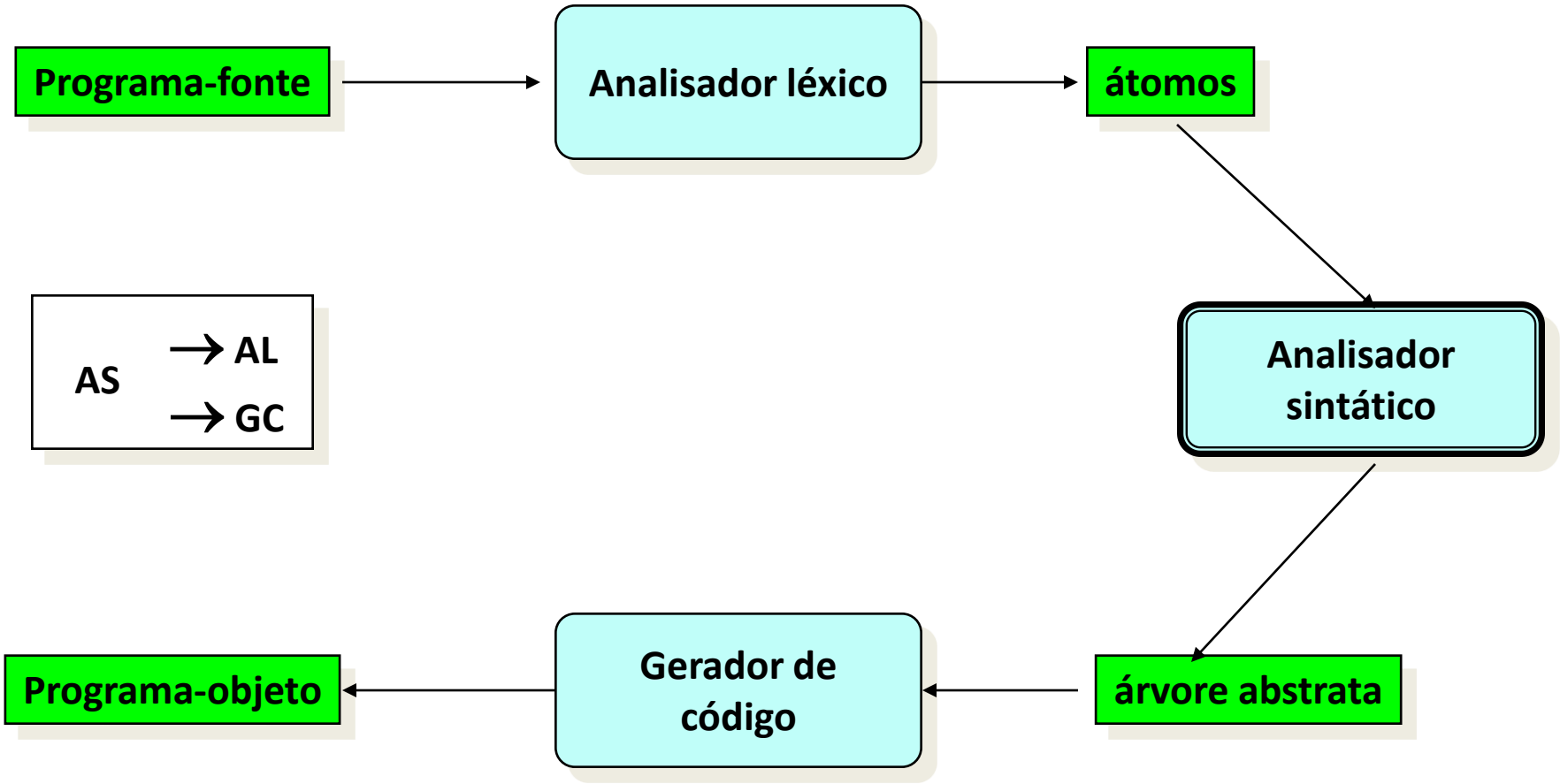
Organização Física

- Três fases macroscópicas:
  - análise léxica
  - análise sintática
  - geração de código
- A organização ideal leva em conta a aplicação
  - didática
  - desenvolvimento e depuração de software de tempo real
  - produção de versões finais para software depurado
- Compromisso entre
  - eficiência de compilação (do compilador)
  - eficiência de execução (do programa compilado)
- Otimizações para aplicações críticas

Compiladores multi-passos

- Compiladores são organizados em *passos*
- Cada passo é um processador completo, contendo analisadores léxico e sintático, e gerador de código
- Em cada passo de compilação são efetuadas transformações específicas no texto de entrada
- A saída de cada passo é entrada do passo seguinte
- A entrada do primeiro passo é o *programa-fonte*
- A saída do último passo é *programa-objeto*
- Os diversos passos do compilador se comunicam através de *linguagens intermediárias*
- As diversas versões intermediárias do programa devem ser equivalentes

Esquema de compilação guiada por sintaxe



Conclusão

- Nesta aula, procuramos caracterizar o papel desempenhado pelas linguagens de alto nível, no contexto dos sistemas de programação.
- Foi também alvo da nossa atenção indicar como, especialmente através dos compiladores e interpretadores, os sistemas de programação proporcionam ao usuário o conforto de construir os seus programas de maneira independente do seu hardware hospedeiro
- Enfim, foi apresentada também uma ligeira ideia da maneira como os compiladores operam para efetuarem a operação da tradução, para um formato executável, dos programas fornecidos em linguagem de alto nível (tema detalhado em outra disciplina)