

Linguagem de Programação Java - Introdução

Jose.wellington@uniceub.br

Calendário

◀ agosto de 2013 ▶

D	S	T	Q	Q	S	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
-	-	-	-	-	-	-

◀ setembro de 2013 ▶

D	S	T	Q	Q	S	S
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

◀ outubro de 2013 ▶

D	S	T	Q	Q	S	S
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

◀ novembro de 2013 ▶

D	S	T	Q	Q	S	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

◀ dezembro de 2013 ▶

D	S	T	Q	Q	S	S
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Agenda

- **Paradigmas de Programação**
 - Programação Linear
 - Programação Funcional
 - Programação Procedural
 - Programação Estruturada
 - Programação Orientada por Objetos
- **Compilador**
 - Compilador – Estrutura
 - Análise Léxica
 - Análise Sintática
 - Análise Semântica
 - Geração de Código Intermediário
 - Otimização de Código
 - Geração de Código
- **Exercício**

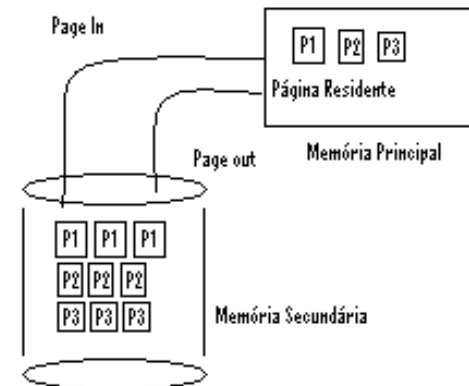
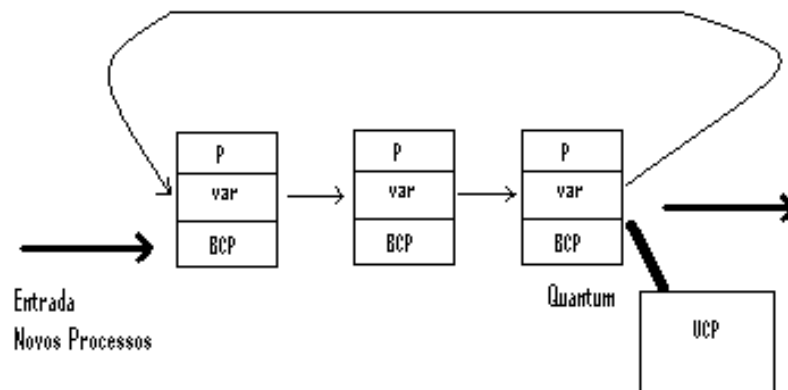
Plataforma - Java

Plataforma Java

■ Escalonamento de Processo não preemptivo

FIFO – First-In-First-out

- É o mais simples e consiste em repartir uniformemente o tempo do processador entre todos os processos prontos para execução.
- Ele consiste em organizar os processos numa fila circular alocando a cada um por sua vez uma fatia de tempo, **time slice**, do processador.



Plataforma Java

- Conjunto de classes (API, bibliotecas) disponíveis
- Basicamente 3 plataformas



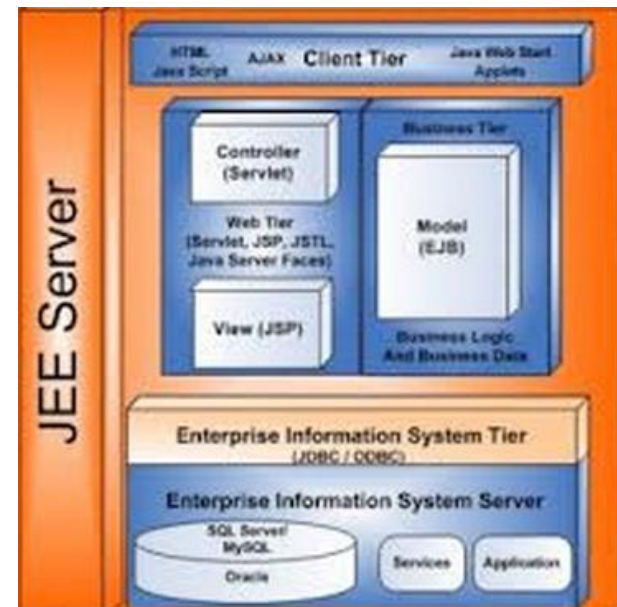
Plataforma Java

- **Plataforma Java SE - JSE**
- Java Platform, Standard Edition. É a base da plataforma. Inclui o ambiente de execução e as bibliotecas comuns
- API padrões da tecnologia: classes essenciais Classes GUI (*Graphical User Interface*).
- Gráficos: AWT, aplicações swing, applets



Plataforma Java

- • Plataforma Java EE
- API (***Application Programming Interface***) para aplicações web: páginas JSP, servlets, EJB, etc.
- Ambiente Corporativo.



Plataforma Java

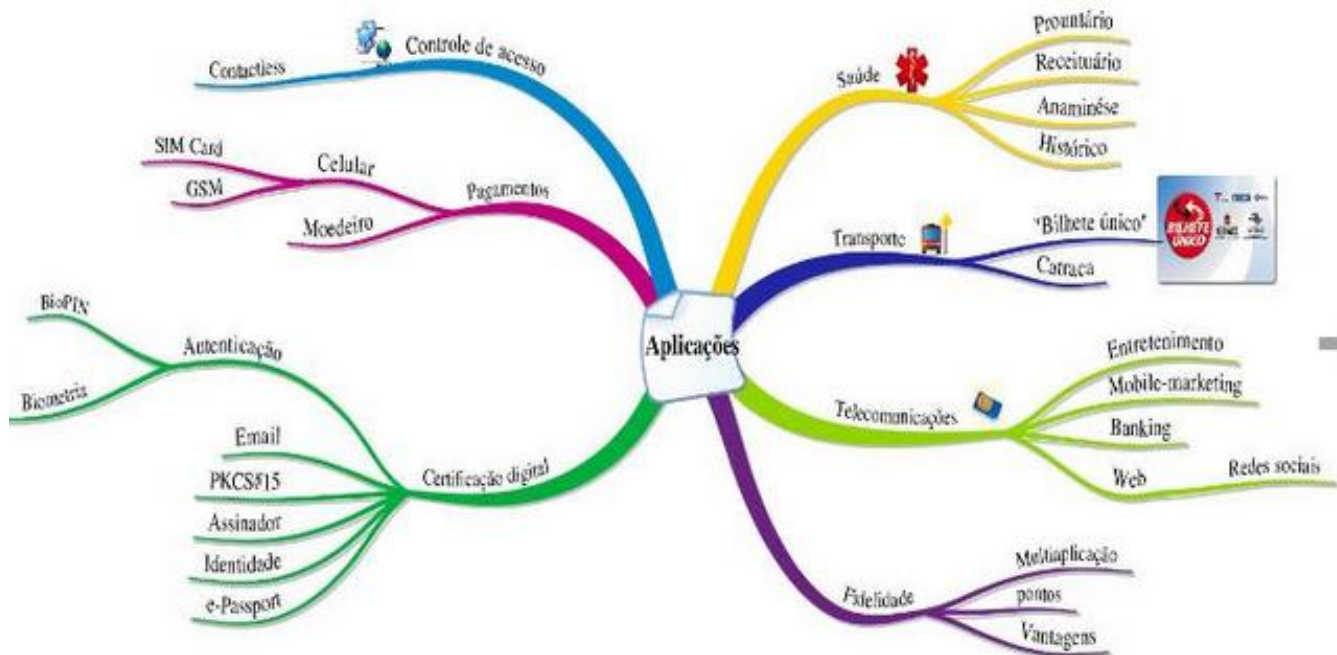
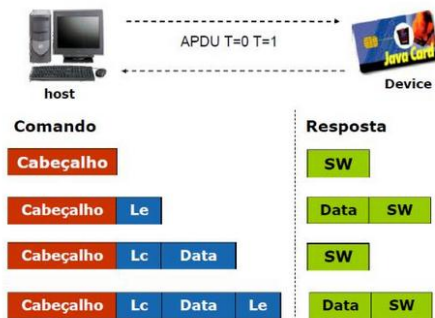
- **Plataforma Java ME**
- API para celulares, **smartphones** e PDAs

Móveis



Plataforma Java

- **Plataforma Java Card**
- Voltada para dispositivos embarcados com limitações de processamento e armazenamento, como smart cards e o **Java Ring**.



Plataforma Java

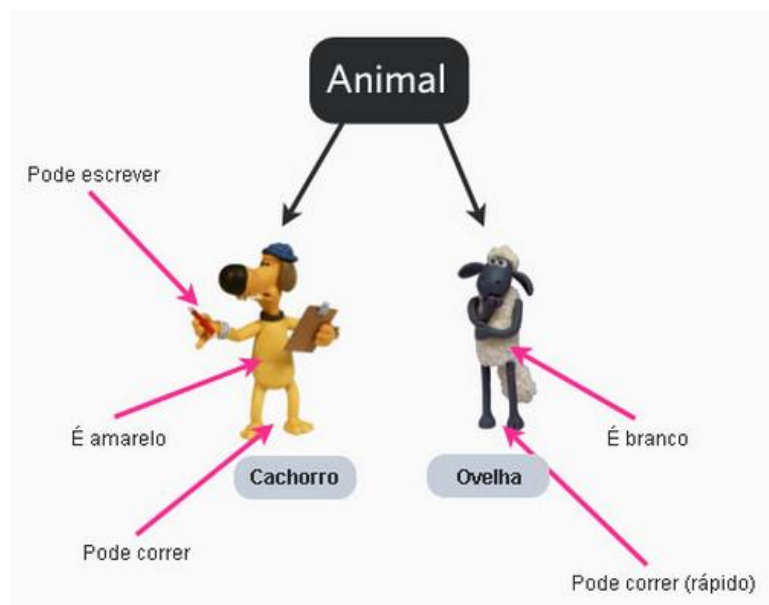
- **Plataforma Java FX**
- Plataforma para desenvolvimento de aplicações **multimídia** em desktop/web (JavaFX Script) e dispositivos móveis (JavaFX Mobile).
- Aplicações web com características de um programa tradicional de um desktop, em diversos dispositivos (**desktop, browser, telefone celulares, TVs, video-games, Blu-rays players etc.**).



Objetivo do Java

Objetivo do Java

- **Orientada a Objetos** – Há trinta anos o conceito de orientação a objetos existe na programação.
- Hoje em dia, ele é sinônimo de modernidade, **eficiência** e extensibilidade, em um universo cujas expectativas mudam muito rapidamente.



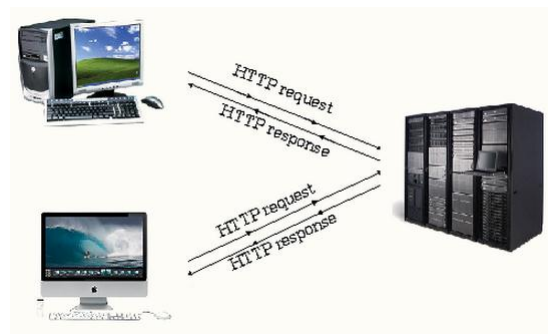
Objetivo do Java

- **Familiar** – O Java se manteve o mais perto possível do C++. Removendo suas complexidades e mantendo sua sintaxe, é possível a uma grande gama de programadores iniciar diretamente a programação nessa linguagem.



Objetivo do Java

- **Robusta** – Java foi criada para desenhar **programas confiáveis**. O interpretador verifica continuamente a execução dos programas, protegendo o sistema de erros.
- A linguagem também **evita que vícios prejudiciais por parte dos programadores** possam causar instabilidade no sistema operacional.
- Não é necessária a **alocação de memória**, e uma **série de erros de bibliotecas** podem ser descobertos imediatamente, na própria compilação.



Objetivo do Java

- **Segura** – A tecnologia do Java foi desenhada para utilizar extensivamente a rede e os ambientes distribuídos. Nessas arquiteturas, segurança é um dos parâmetros principais. **Um aplicativo em Java não pode ser invadido via rede**, pois suas restrições de segurança não permitem acessos não autorizados.



Objetivo do Java

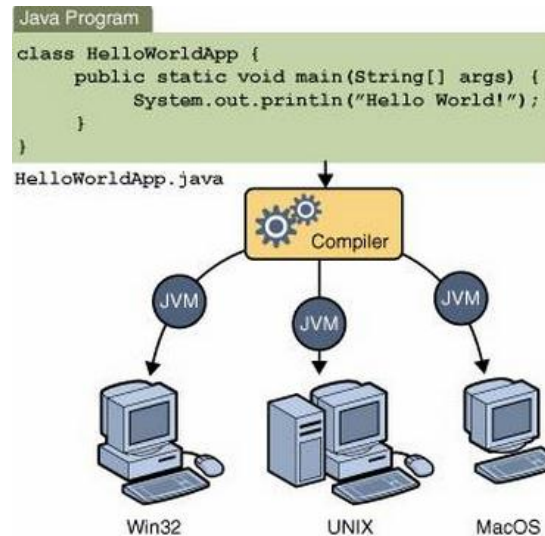
- **Neutralidade** – Java foi criada para funcionar em uma grande variedade de **plataformas de hardware**. seus **bytecodes permitem a criação de um programa em qualquer plataforma** e sua execução em qualquer plataforma.

Java está em todo lugar



Objetivo do Java

- **Portabilidade** – A neutralidade de arquitetura é apenas um dos pontos que indicam a **portabilidade de um sistema**. Além disso, o Java uniformiza os tipos de dados nas diferentes arquiteturas, de modo que um inteiro num PC representa a mesma quantidade de bits em uma estação de trabalho. Assim, um programa Java é totalmente independente de hardware e software.



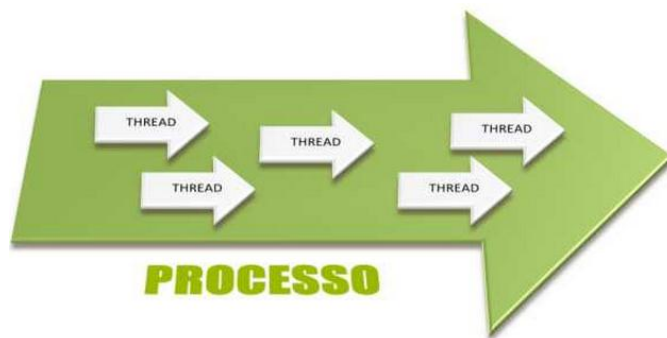
Objetivo do Java

- **Alta Performance** – A performance de um programa Java é relacionada estritamente à **performance do interpretador**.
- A JVM Java permite que se execute o código do usuário na máxima velocidade possível; todas as outras tarefas ficam em segundo plano.
- Ainda assim, se for necessário uma performance ainda maior, **é possível compilar o bytecode para código nativo da máquina**.



Objetivo do Java

- **Multi-Tarefa** – Sistemas orientados à rede necessitam **executar várias tarefas ao mesmo tempo**. Java permite a construção de um modelo onde podem ser executadas *threads* concorrentes.



Paradigmas de Programação

Paradigmas de Programação

■ Programação Linear

- Em matemática, problemas de Programação Linear são problemas de otimização nos quais a função objetivo e as restrições são todas lineares.

- **Programação sequencial** - Sequências os comandos podendo ter GOTO.

```
10 INPUT A$
20 GOTO 200
30 PRINT A$,B
40 GOTO 1000
100 GOTO 30
200 INPUT B
210 IF B>=0 GOTO 30
220 IF B<0 GOTO 100
500 GOTO 3000
1000 INPUT C$
1200 INPUT D
2000 IF D>0 GOTO 500
3000 PRINT A$, "+", C$, "=", B+D
5000 END
```

Paradigmas de Programação

■ Programação Funcional

- Trata a computação como uma avaliação de **funções** matemáticas e que evita estados ou dados mutáveis. Ela enfatiza a aplicação de funções (Linguagem Haskell).

```
(%i32) map("=", [a,b,c], [1,2,3]);
```

```
(%o32) [ a = 1 , b = 2 , c = 3 ]
```

```
(%i33) f : lambda([x,y], (x+y)*(x-y));
```

```
(%o33) lambda( [ x , y ] , (x+y)(x-y))
```

```
(%i34) f(a,b);
```

```
(%o34) (a-b)(b+a)
```

```
(%i35) map(f, [a,b,c], [1,2,3]);
```

```
(%o35) [(a-1)(a+1),(b-2)(b+2),(c-3)(c+3)]
```

```
(%i36) expand(%);
```

```
(%o36) [a2-1, b2-4, c2-9]
```

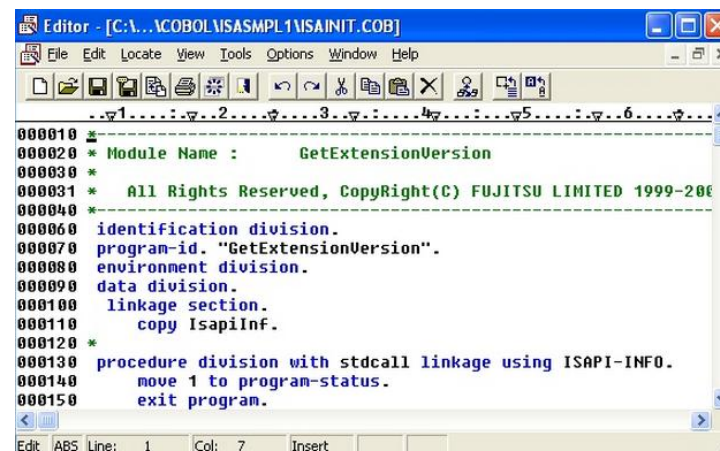
```
(%i37) factor(%);
```

```
(%o37) [(a-1)(a+1),(b-2)(b+2),(c-3)(c+3)]
```


Paradigmas de Programação

■ Programação Procedural

- Baseada no conceito de chamadas a **procedimento** (linguagens: C, C++, Fortran, Pascal, MATLAB).
- Deve suportar o conceito de procedimentos, e possuir uma sintaxe para defini-los.
- Idealmente, ela deve suportar a especificação de tipos de **argumentos**, variáveis locais, chamadas recursivas e o uso de procedimentos em módulos distintos de um programa. Ela também pode suportar a distinção entre argumentos de **entrada e de saída**.

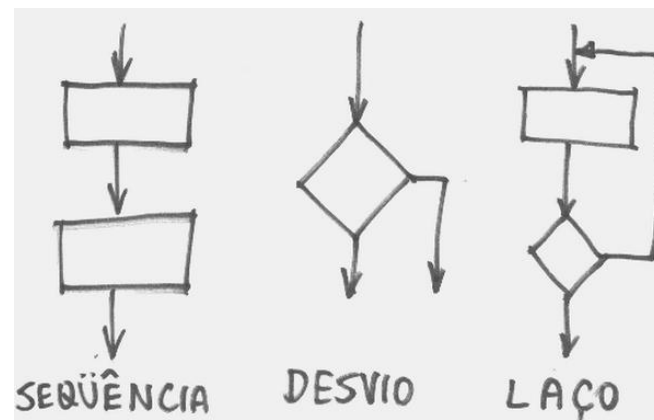


```
Editor - [C:\V...\ACOBOLVISASMP1\VSAINIT.COB]
File Edit Locate View Tools Options Window Help
--1--2--3--4--5--6--
000010
000020 * Module Name :      GetExtensionVersion
000030 *
000031 *  All Rights Reserved, Copyright(C) FUJITSU LIMITED 1999-200
000040 *-----
000060 identification division.
000070 program-id. "GetExtensionVersion".
000080 environment division.
000090 data division.
000100 linkage section.
000110     copy IsapiInf.
000120 *
000130 procedure division with stdcall linkage using ISAPI-INFO.
000140     move 1 to program-status.
000150     exit program.
```


Paradigmas de Programação

■ Programação Estruturada

- É uma forma de programação de computadores que preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas: **sequência**, **decisão** e **iteração**.
- Também chamada de **Programação Modular**.



Paradigmas de Programação

■ Programação Orientada por Objetos

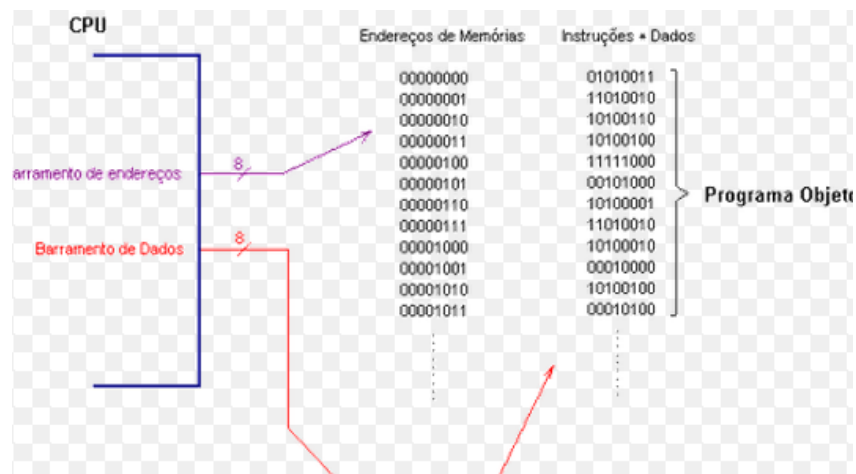
- Aproximar o mundo real do mundo virtual: a idéia fundamental é tentar simular o mundo real dentro do computador.
- Para isso, nada mais natural do que utilizar Objetos.
- Permite re-uso de código e flexibilidade no desenvolvimento.



Compilador

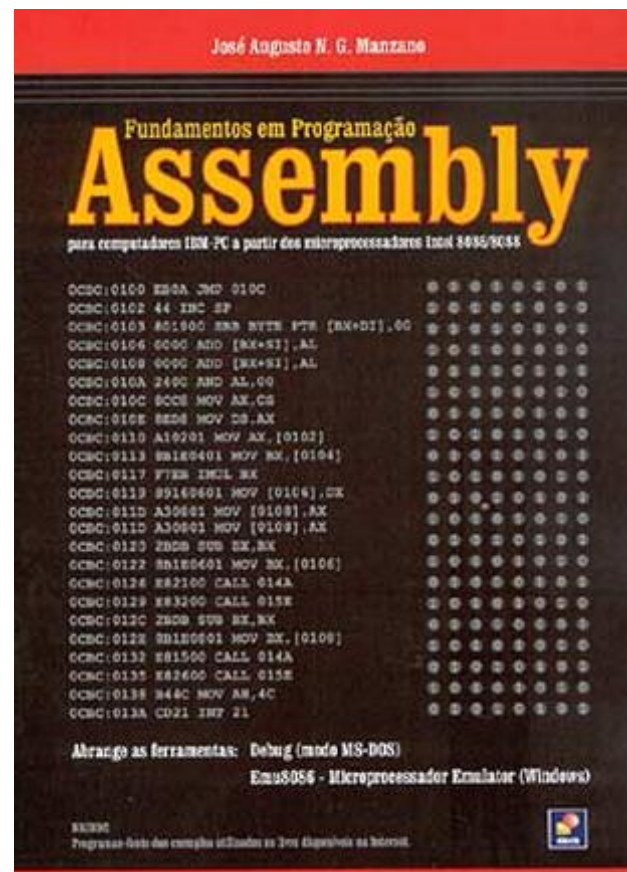
Programa em Linguagem de Máquina

- Uma linguagem de programação é um conjunto de ferramentas, **regras de sintaxe e símbolos ou códigos** que nos permitem escrever programas de computador.
- A primeira e **mais primitiva linguagem** de computador é a própria linguagem máquina (0's e 1's).
- Um programa era **difícil, longo e principalmente caro** de o construir.
- Era também **difícil de ser entendido** por outros programadores.
- Essa **complexidade** levou a **necessidade de desenvolver novas técnicas e ferramentas**.



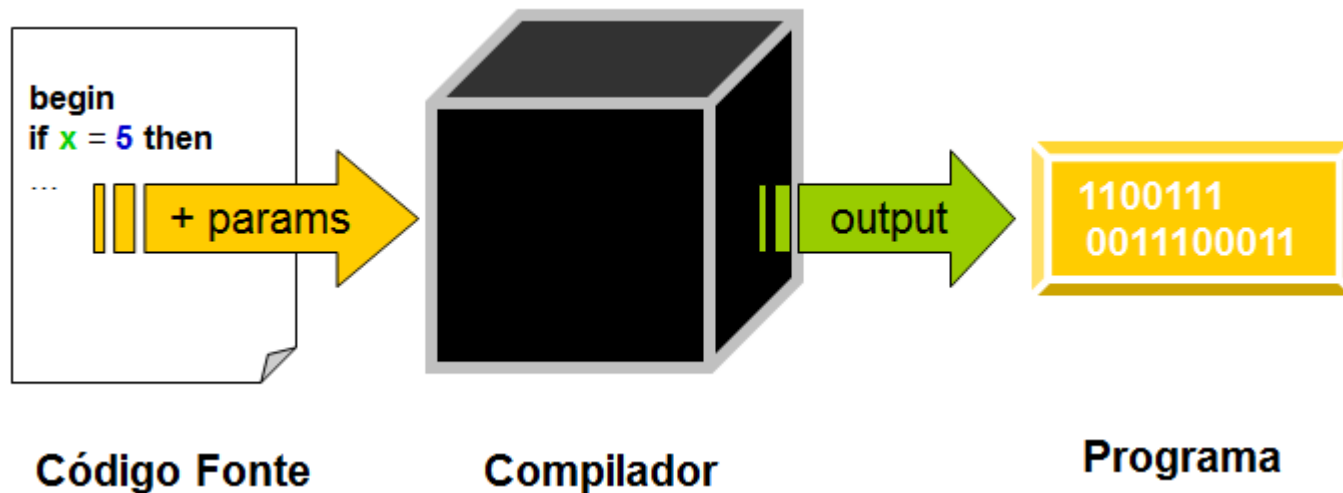
Linguagem de Montagem

- A resolução do problema passou pela criação de uma linguagem em que os códigos numéricos foram substituídos por **mnemônicos**.
- O nome dessa linguagem é **ASSEMBLY LANGUAGE**.
- Então será necessário um outro programa que leia o programa escrito nessa linguagem alternativa e o traduza para a linguagem nativa do computador!!!
- O processo de tradução da linguagem de montagem para a linguagem de máquina é realizada por um programa chamado **ASSEMBLER**.
- Essa complexidade levou à necessidade de desenvolver novas técnicas e ferramentas.



Compilador

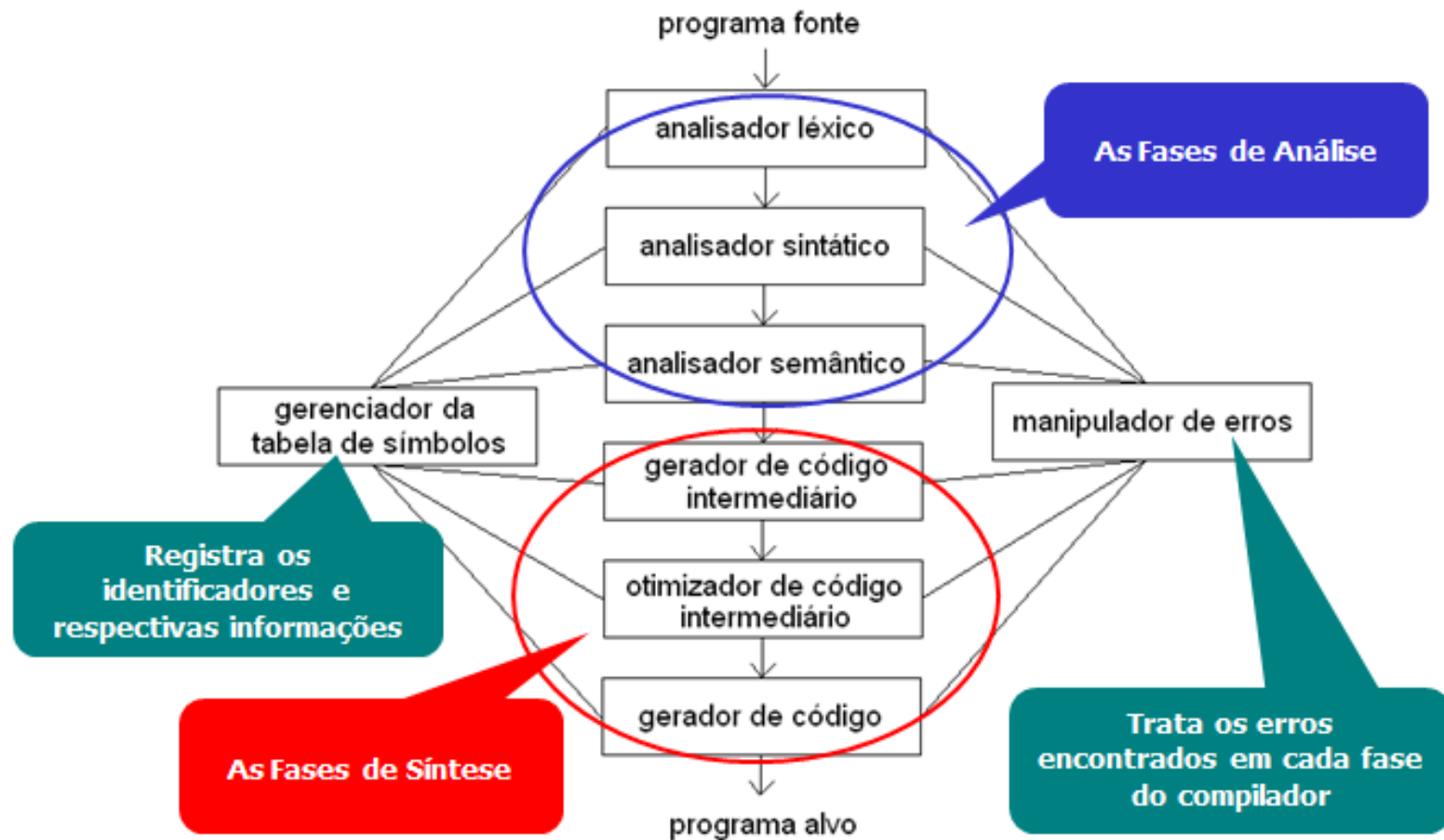
- Transforma Linguagem de alto nível em linguagem de máquina.



Compilador

- Um **compilador** tem a finalidade de **converter** uma linguagem – Linguagem Fonte – de fácil escrita e leitura para os programadores, numa linguagem – **Linguagem alvo ou objeto** – que possa ser executada pelas máquinas.
- O **código executável** gerado pelo **compilador** é **dependente do sistema operacional e da linguagem de máquina para o qual o código fonte foi traduzido**.
- A enorme variedade de compiladores existentes é bem vinda, visto que **existem milhares de linguagens fonte**, e as linguagens alvo são também muito variadas.

Compilador - Estrutura



Compilador

■ Análise Léxica

- Ler o arquivo com o programa-fonte
- Transforma o caractere do programa fonte em tokens (símbolos).
- Identificar os tokens correspondentes
 - “um token se estende até que seja encontrado um caractere que não faça parte dele”
- Relatar erros léxicos

Tokens e lexemas

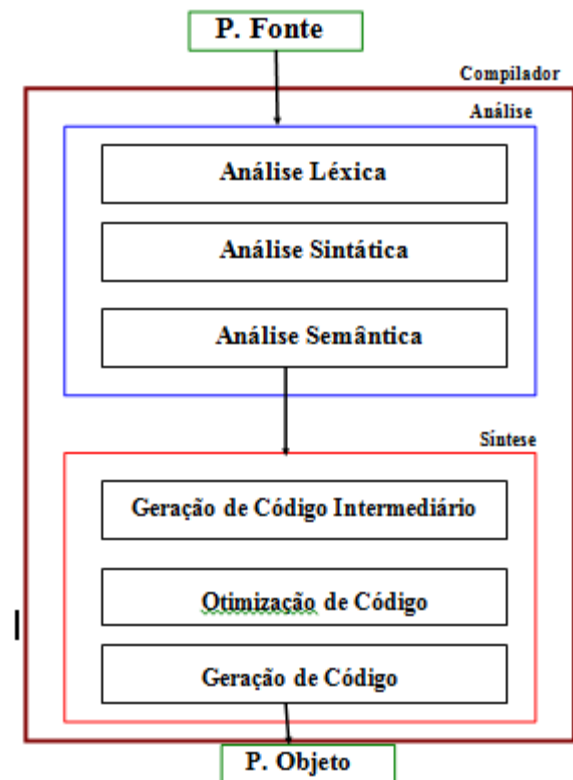
Tokens	Lexemas
FOR	for
IF	if
WHILE	while
NÚMERO	1089, 142857, 0.2, 3.14159
IDENTIFICADOR	i, j, contador, nomeAluno
OP_SOMA	+
OP_MAIOR_IGUAL	>=
ABRE_PAR	(

Compilador

■ Análise Sintática

- Obtém uma sequência de *tokens* fornecidos pelo analisador léxico e **verifica se a mesma pode ser gerada pela gramática.**
- Verificar se a sintaxe da linguagem na qual o **programa foi escrito está sendo respeitada.**
- **Detectar/Diagnosticar** erros sintáticos.

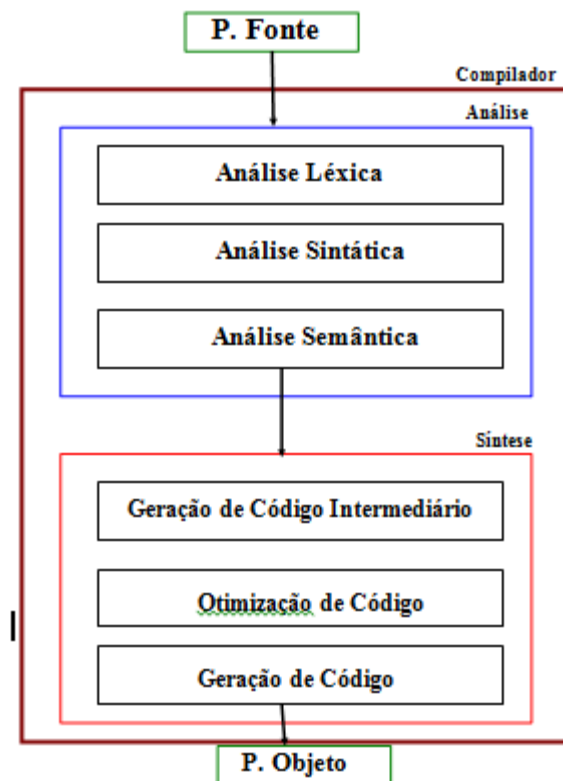
Estrutura geral de um Compilador



Compilador

- **Análise Semântica**
- SEMÂNTICA \cong COERÊNCIA \cong SIGNIFICADO \cong SENTIDO LÓGICO
- Verificar se as construções utilizadas no Prog. Fonte estão semanticamente corretas
- Embora corretos sintaticamente, mas detectar e diagnosticar erros semânticos, A+B Funciona em muitas linguagens

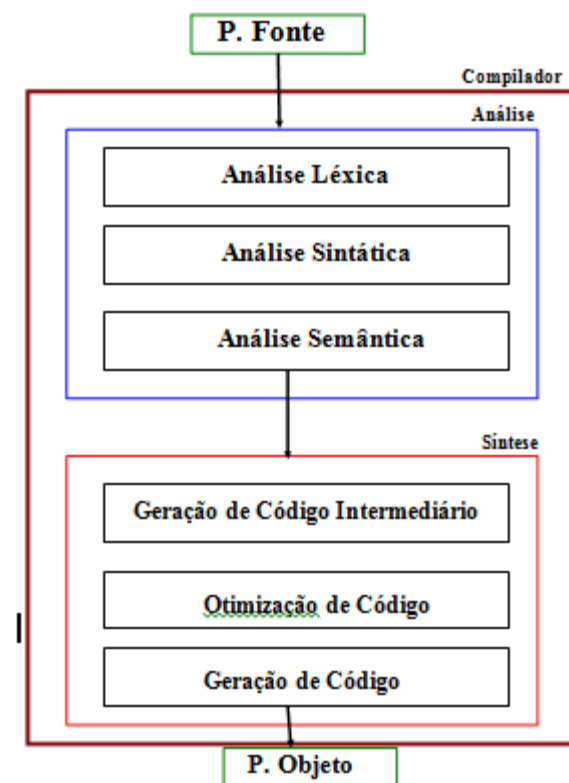
Estrutura geral de um Compilador



Compilador

- **Geração de Código Intermediário**
- Usa a estruturas produzidas pelo analisador sintático e verificadas pelo analisador semântico para criar uma sequência instruções simples ditas **Código intermediário**.
- Consiste na geração de um conjunto de instruções (equivalentes ao programa fonte de entrada) para uma **máquina hipotética** (virtual) – Código Objeto.

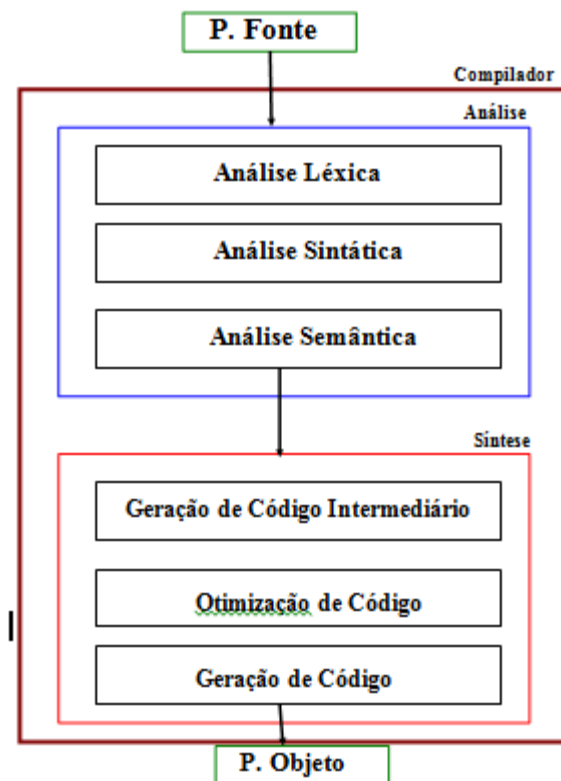
Estrutura geral de um Compilador



Compilador

- **Otimização de Código**
- É um **módulo opcional** (presente na maioria dos compiladores) que objetiva **melhorar o código intermediário** com objetivo reduzir o código para ocupar **menor memória** e **mais rápido a execução**.
- A **saída** do otimizador de códigos é um **novo código intermediário**.
- Melhorar o código, de forma que a execução seja mais eficiente quanto ao **tempo** e/ou **espaço ocupado**

Estrutura geral de um Compilador



Compilador

- **Geração de Código**
- Converter o programa fonte (diretamente ou a partir de sua representação na forma de código intermediário) para uma sequência de instruções (assembler ou **máquina**) de uma máquina real.
- **Produz o código objeto final** tomando decisões com relação a **alocação de espaço** para os dados do programa, selecionando a forma de acessá-los, definindo os registradores da UCP serão utilizados.

