# Organização de Computadores

## Fundamentos da Programação de Computadores

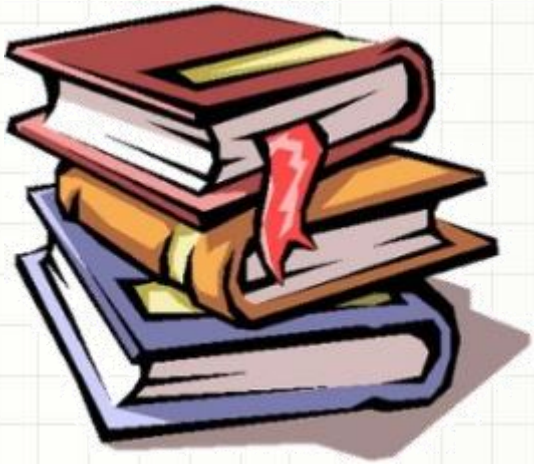
Prof. Dr. Daniel Caetano

2014 - 1

# Objetivos

- Compreender do que é composto um programa
- Compreender e conhecer as diferentes formas de programar o computador
- Conhecer as unidades usadas na informática e seus múltiplos

# Material de Estudo



| Material | Acesso ao Material |
| --- | --- |
| Notas de Aula | http://www.caetano.eng.br/<br>(Organização de Computadores - Aula 2) |
| Apresentação | http://www.caetano.eng.br/<br>(Organização de Computadores - Aula 2) |
| Material Didático | ... |

# A Resolução de Problemas e os Algoritmos

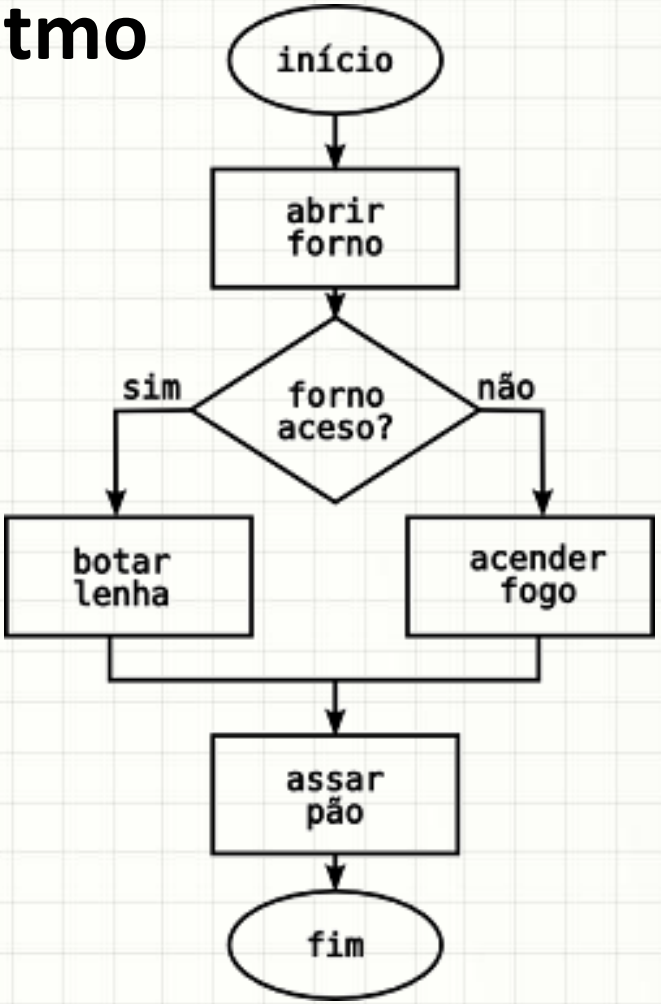
# O que são Algoritmos

- Toda tarefa complexa pode ser subdividida
  - Tarefas menores e mais simples
- Exemplo: fabricar vinho para venda
  - Plantar a uva
  - Colher a uva
  - Amassar a uva
  - Deixar fermentar
  - Engarrafar
  - Distribuir para a venda

# O que são Algoritmos

- O procedimento para realizar uma tarefa complexa chama-se **algoritmo**

- Um algoritmo envolve:
  - **Tarefas/Processos**
  - **Decisões**



# O que são Algoritmos

- Há algoritmos em nosso dia a dia:
  - Listas de Compras
  - Receitas
  - Caminhos do Google Maps...


# Algoritmos x Programas

- No computador, os algoritmos recebem o nome de **programas**

# A Linguagem de Máquina

# Linguagem de Máquina

- Como indicar instruções para o processador?

# Linguagem de Máquina

- Como indicar instruções para o processador?

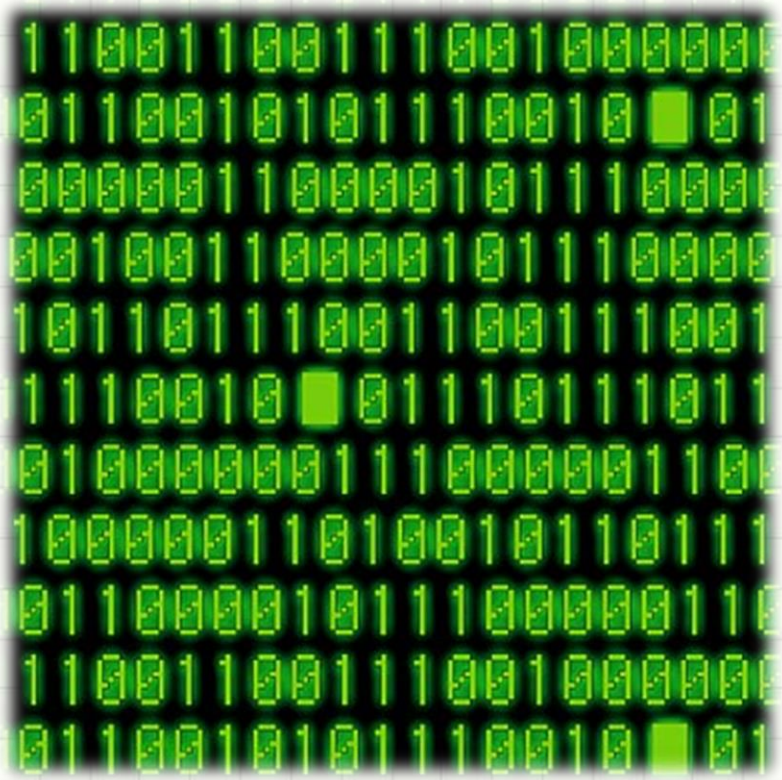
**FIOS**

# Linguagem de Máquina

- Como indicar instruções para o processador?

# Linguagem de Máquina

- Como seria um programa em linguagem de máquina?



# LINGUAGEM ASSEMBLY

# Linguagem Assembly

- Ninguém percebeu que LM era horrível?
  - Não demorou!
- Qual a solução?

| Op1 | Mnemonico |
|---|---|
| **00000000** | NOP |
| **00000001** | LD BC,nnmm |
| **00000010** | LD (BC),A |
| **00000011** | INC BC |
| **00000100** | INC B |
| **00000101** | DEC B |
| **00000110** | LD B,mm |
| **00000111** | RLCA |

# Linguagem Assembly

- Linguagem Assembly: Instruções Mnemônicas

- E quem converte?
  - No início, manualmente
  - Muito rapidamente: **assemblers**

- Qual o papel dos “assemblers”?

# O que faz o Assembler?

Programador

LD A,B
ADD A,C
XOR C
INC B

Assembly

Assembler

001010101010
101010101010
110111011011
111110010101

Código Binário


Computador

# Por que não usar Assembly?

- Muito burocrático!
- Exige compreensão do hardare
- Instruções muito “simples”
- Difícil de ler e de escrever!

```
LD      A,16
LD      B,32
ADD     A,B
SRA     A
```

$$A = (16 + 32)/2$$

# Linguagens de Programação de Alto Nível

# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- FORTRAN

```
PROGRAM TRIVIAL
    INTEGER I
    I=2
    IF(I .GE. 2) CALL PRINTIT
    STOP
END
SUBROUTINE PRINTIT
    PRINT *,'Hola Mundo'
    RETURN
END
```

# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.
PROCEDURE DIVISION.
    DISPLAY 'Hello, world'.
    STOP RUN.
```

# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- LISP

```
;;; Hello World in Common Lisp

(defun helloworld ()
  (print "Hello World!")
)
```

# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- C/C++

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- BASIC

```
10 PRINT “HELLO WORLD”
```

# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- Pascal

```
PROGRAM hello;
USES CRT;
BEGIN
Writeln('Hello World!');
END.
```

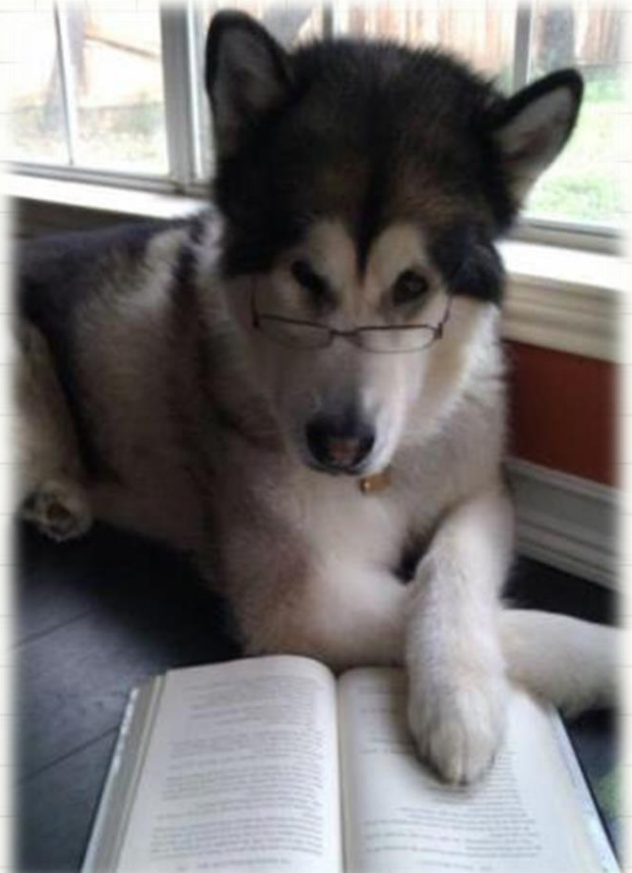
# Linguagens de Alto Nível

- Linguagens mais próximas da humana
- Java

```
public class Hello {

    public static void main(String[] args) {

    System.out.println("Hello");

    }

}
```

# Linguagens de Alto Nível

- Cada uma tem vantagens e desvantagens
- Todas: **incompreensíveis** para o computador



HUM!?

# Processo de Compilação

Programador

```
#include <io...
int main(void)
  {
  cout << “Oi”;
```

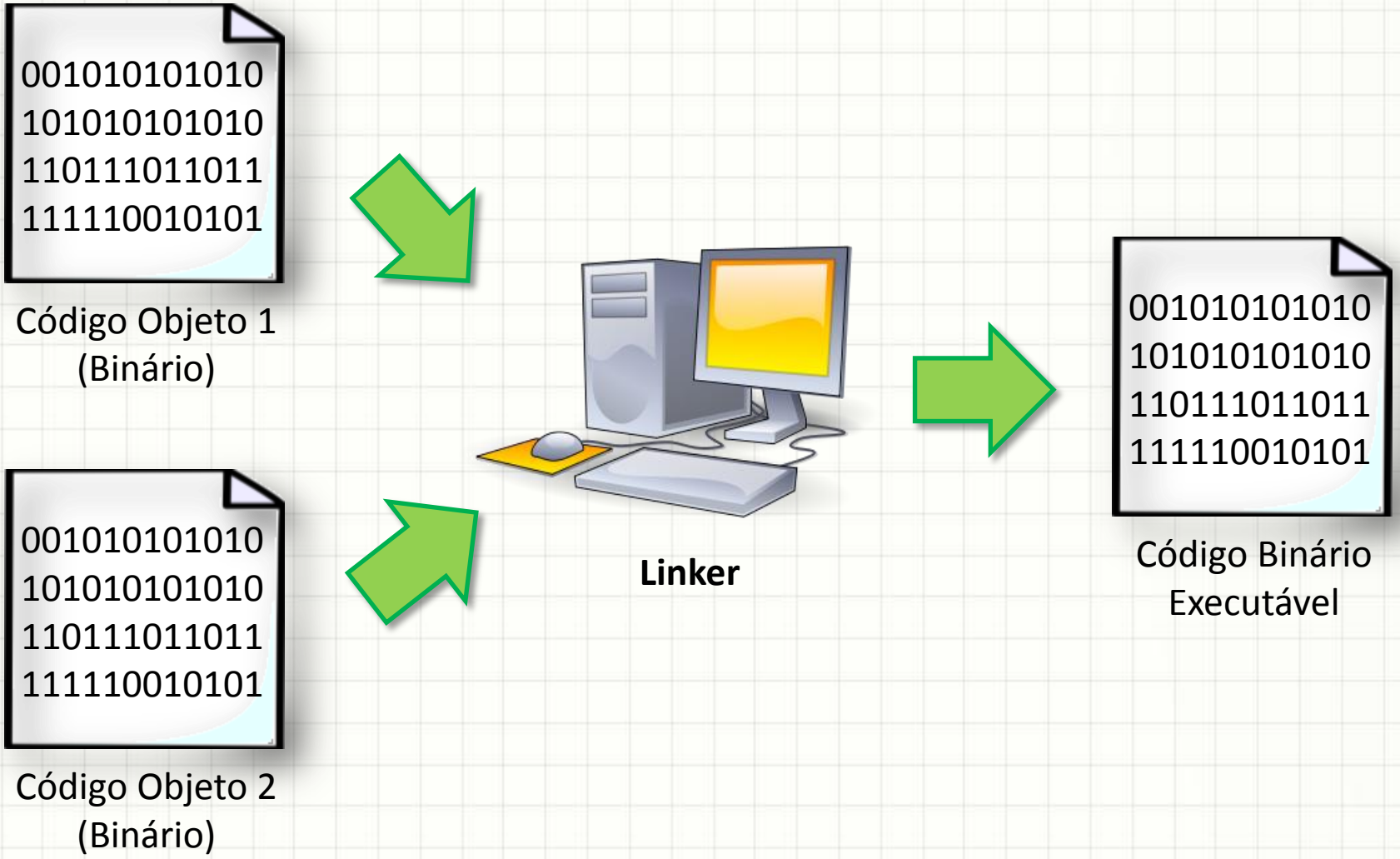
Código Fonte

**Compilador**

```
001010101010
101010101010
110111011011
111110010101
```

Código Objeto
(Binário)

# Processo de Compilação



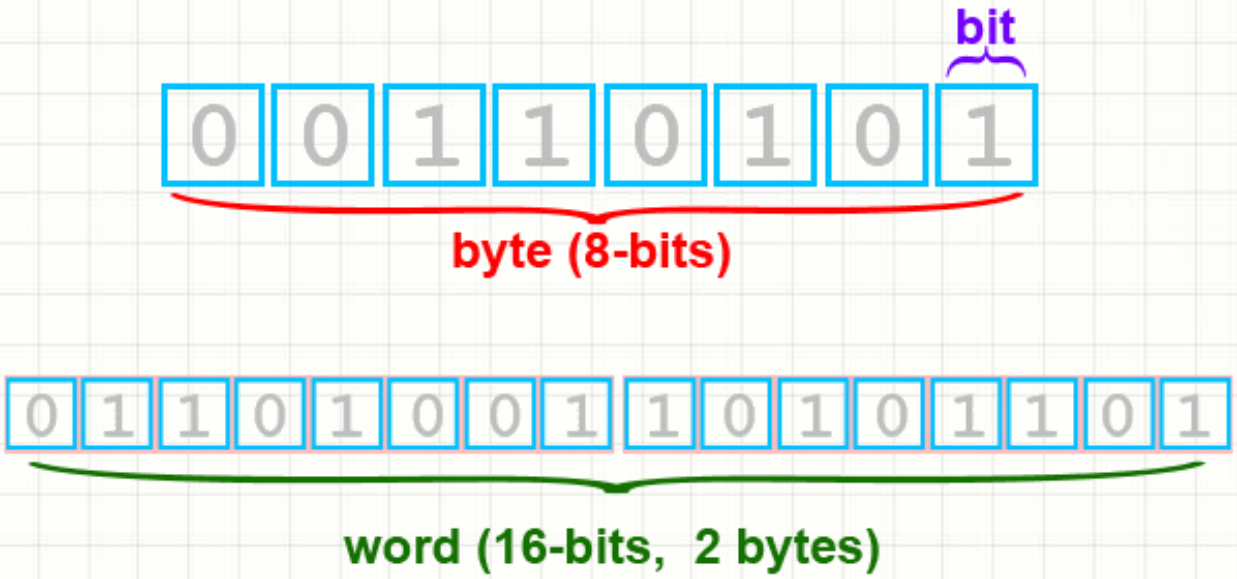
# Tamanho dos Dados e suas Unidades

# Tamanho dos Dados e Unidades

- Como cada **bit** pode ser apenas 0 ou 1...

...o nome dessa representação é "**bi**nária".

- Um único bit armazena pouca informação
- Usualmente, os bits aparecem agrupados



# Múltiplos Usados em Computação

- Os múltiplos são:
  - K: kilo
  - M: mega
  - G: giga
  - T: tera
  - P: peta

| Representação | Valor em Bytes |
|---|---|
| **1KB** | **$2^{10}$ bytes (1.024 bytes)** |
| **1MB** | **$2^{20}$ bytes (1.024 Kbytes)** |
| **1GB** | **$2^{30}$ bytes (1.024 Mbytes)** |
| **1TB** | **$2^{40}$ bytes (1.024 Gbytes)** |
| **1PB** | **$2^{50}$ bytes (1.024 Tbytes)** |

# EXERCÍCIO

# Exercício

- Realize os seguintes cálculos
  - Quanto resulta 2KB x 2KB?
  - Quanto resulta $2^{10}$ x 32?
  - Quanto resulta 32 x 64?
  - Quantos KB tem 1GB?
  - Quantos bytes tem 1KB?

# Exercício

- Realize os seguintes cálculos
  - Quanto resulta 2KB x 2KB? **4MB**
  - Quanto resulta $2^{10}$ x 32? **$2^{15}$**
  - Quanto resulta 32 x 64? **$2^{11}$**
  - Quantos KB tem 1GB? **1.048.576**
  - Quantos bytes tem 1KB? **1.024**

# PERGUNTAS?

# Conclusões

# Resumo

- Algoritmos são passos para uma solução
- Programa é um algoritmo para o computador
- Linguagem de Máquina e Assembly: complexas
- Linguagens de Alto Nível: mais práticas
  - Compiladores e Linkers
- Unidades básicas e seus múltiplos

---

- Sistemas de Numeração
  - Bases numéricas
  - Conversões decimal/binário/hexadecimal