Algoritmos e Estrutura de Dados - I

Unidade 3 - Implementação de Algoritmos

Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com

Bibliografia

- Forbellone, André L. V.; Eberspächer, Henri Frederico, **Lógica de Programação**, 2ª Edição. Editora Pearson Education, São Paulo, 2001

- Berg, Alexandre; Figueiró, Joice Pavek, **Lógica de Programação**, 3ª Edição, Editora Ulbra,Canoas, 2000

- Souza M. A. F.; Gomes M.M.; Soares M. V.; Concilio R., **Algoritmos e Lógica de Programação** – 2ª edição – CENGAGE, 2014

Implementação de Algoritmos

Algoritmo é um conjunto finito de regras, bem definidas, para a solução de um problema em um tempo finito.

Programa é um algoritmo codificado (escrito) em uma linguagem de programação (C/C++).

Em primeiro lugar...

O que é uma Linguagem de Programação?

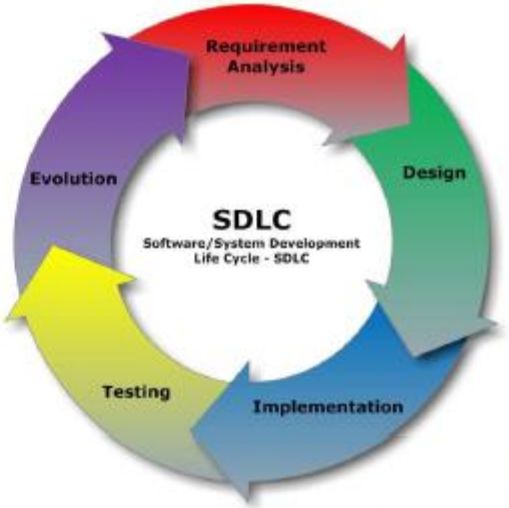
Linguagem de Programação

- Na programação de computadores, uma linguagem de programação serve como **meio de comunicação** entre o indivíduo que deseja resolver um determinado problema e o computador.

- A linguagem de programação deve fazer a ligação entre o **pensamento humano** (muitas vezes de natureza não estruturada) e a **precisão requerida para o processamento** pelo computador.

Linguagem de Programação

- Uma linguagem de programação auxilia o programador no processo de desenvolvimento de software:
  - Projeto;
  - Implementação;
  - Teste;
  - Verificação;
  - Manutenção do software;



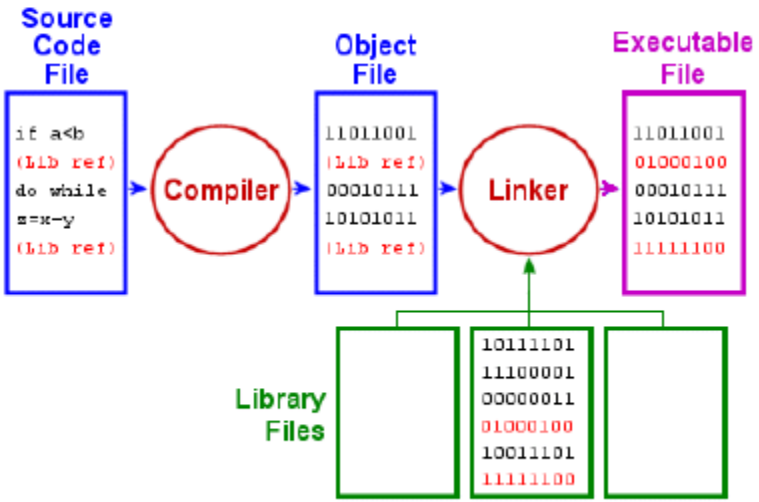
Linguagem de Programação

- Uma Linguagem de Programação é uma linguagem usada por uma pessoa para **expressar** um processo através do qual um **computador** possa **resolver** um **problema**;

- Os paradigmas de linguagens de programação correspondem a diferentes **modelos** pelos quais os processos possam ser expressados.
  - Exemplos: **Imperativo**, **Orientado a Objetos**, **Funcional**, **Lógico**.

Linguagem de Programação

- Para que se tornem operacionais, os programas escritos em **linguagens de alto nível** devem ser traduzidos para **linguagem de máquina**.



Linguagem de Programação

- A conversão de um código em **linguagem alto nível para linguagem de máquina** é realizada através de sistemas especializados:

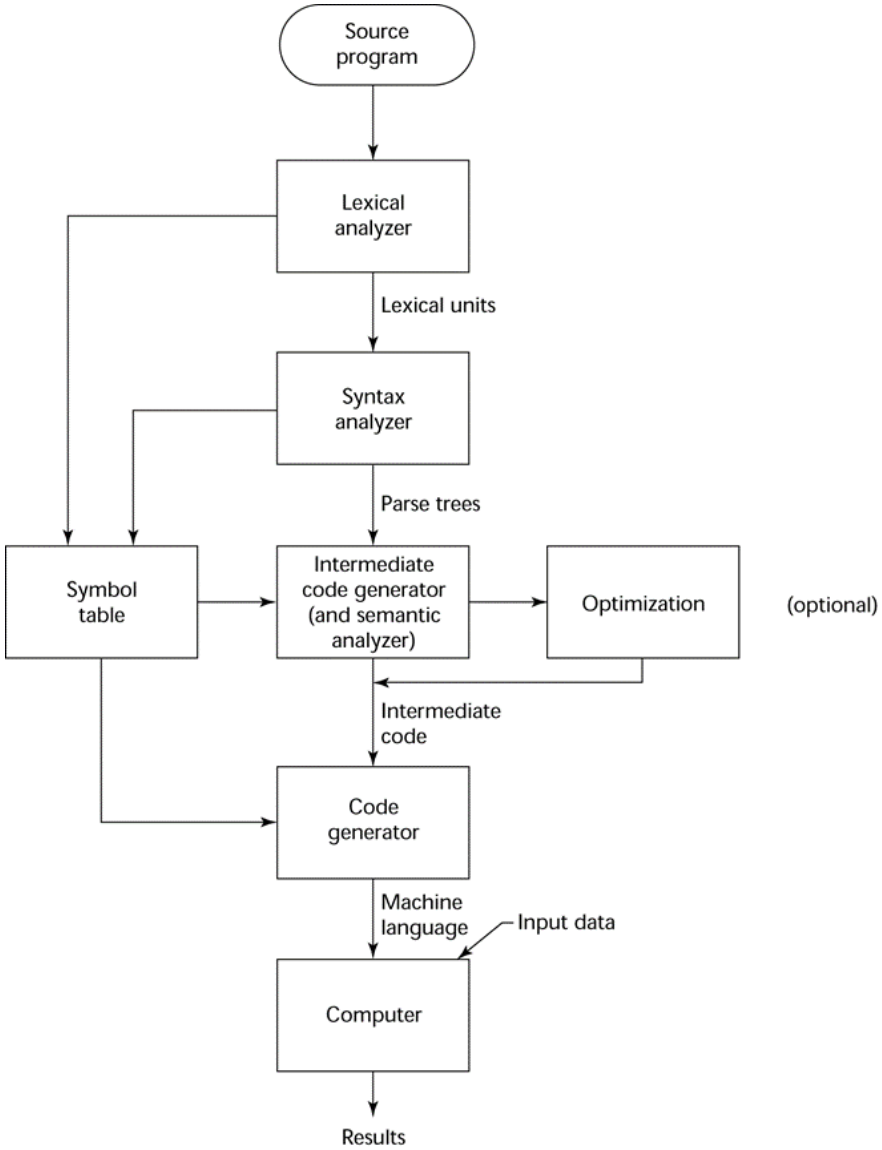
Compiladores ou **Interpretadores**

- Esses sistemas recebem como entrada uma representação textual da solução de um problema (expresso em uma **linguagem fonte**) e produzem uma representação do mesmo algoritmo expresso em uma **linguagem de máquina**.

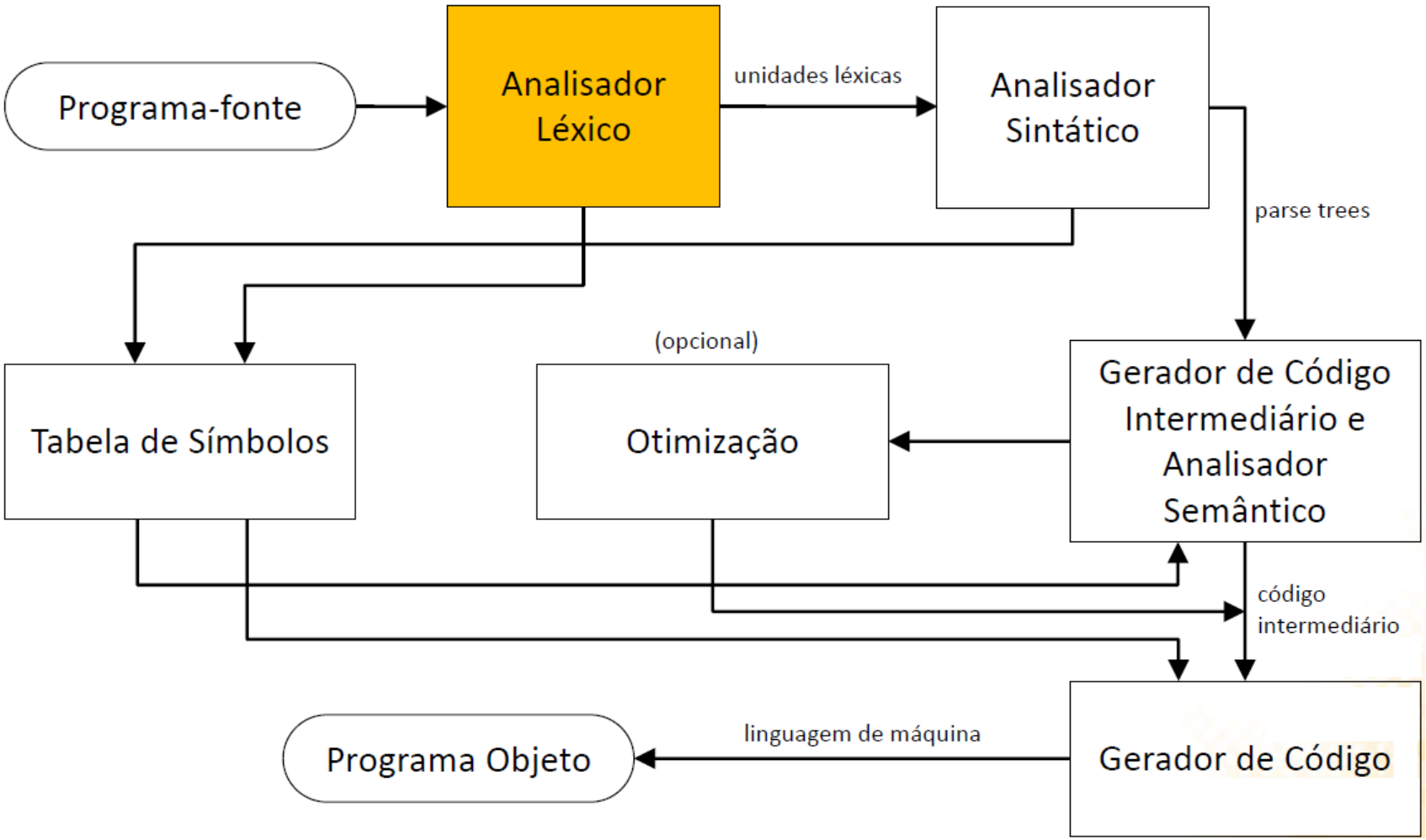
TIOBE – Março 2020

Mar 2020	Mar 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.78%	+2.90%
2	2		C	16.33%	+3.03%
3	3		Python	10.11%	+1.85%
4	4		C++	6.79%	-1.34%
5	6	^	C#	5.32%	+2.05%
6	5	v	Visual Basic .NET	5.26%	-1.17%
7	7		JavaScript	2.05%	-0.38%
8	8		PHP	2.02%	-0.40%
9	9		SQL	1.83%	-0.09%
10	18	^^	Go	1.28%	+0.26%

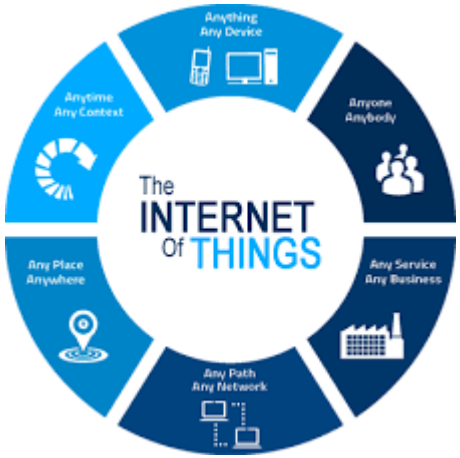
Processo de Compilação



Processo de Compilação

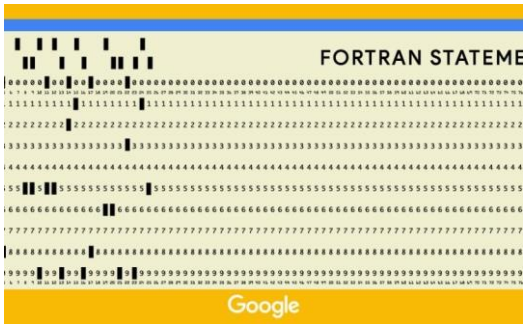


Domínios da Programação



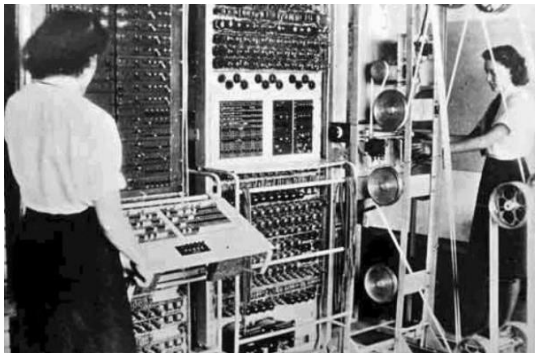
Domínios da Programação
Aplicações Científicas

- Os primeiros computadores que surgiram na década de 40 foram projetados e utilizados para **aplicações científicas**.



- Nesta categoria se enquadram todos os problemas que necessitam um grande volume de processamento, com operações geralmente feitas em ponto flutuante, e com poucas exigências de entrada e saída.
  - Uma da preocupações primarias neste tipo de aplicação é a **eficiência**.

- As aplicações científicas incentivaram a criação de algumas linguagens de alto nível, como por exemplo o **Fortran**.

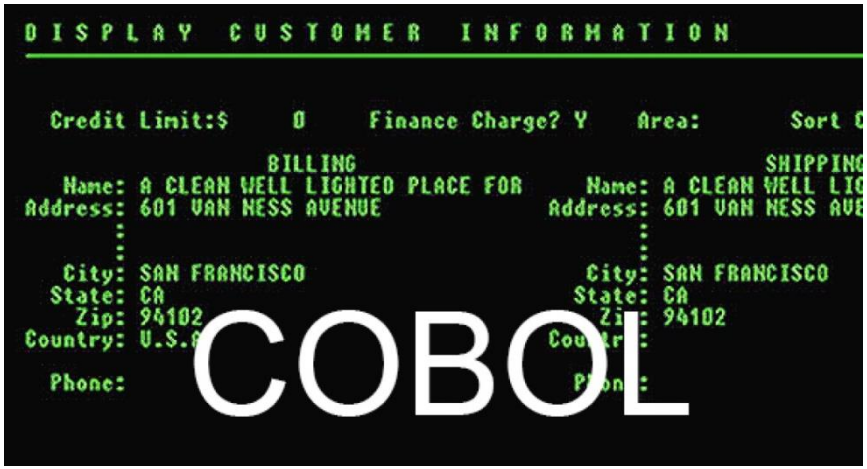


Domínios da Programação
Aplicações Comerciais

- O desenvolvimento de aplicações comerciais teve início na década de 50.

- A primeira linguagem bem sucedida para o desenvolvimento de aplicações comerciais foi o **COBOL** (em 1960).

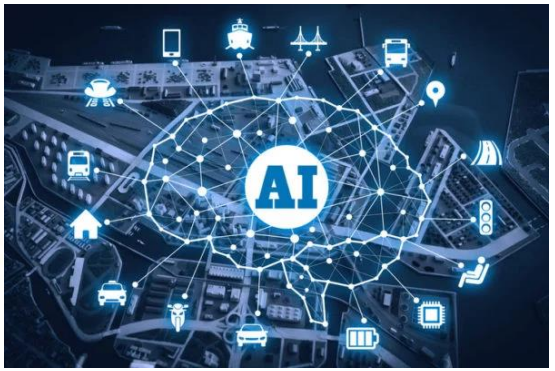
- As linguagens de programação comerciais se caracterizam pela facilidade de elaborar relatórios e armazenar números decimais e dados de caracteres.



Domínios da Programação
Inteligência Artificial

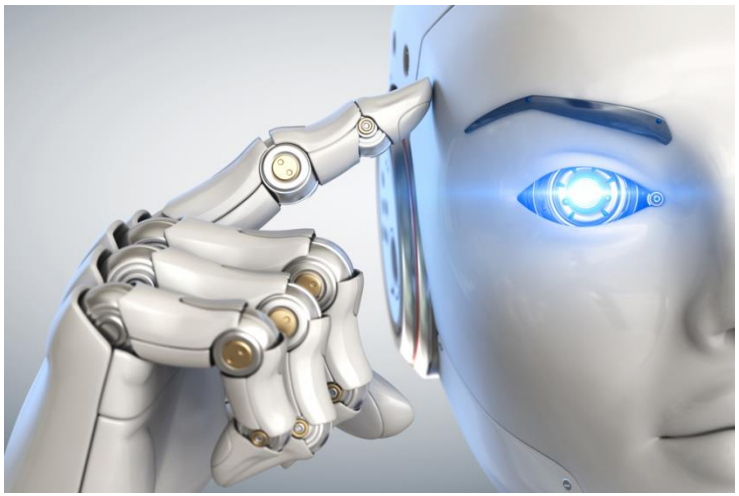
- O desenvolvimento de aplicações para inteligência artificial teve inicio no final da década de 50.

- Essas aplicações caracterizam-se pelo uso de computações simbólicas em vez de numéricas (são manipulados nomes e não números);



- A primeira linguagem desenvolvida para IA foi a funcional **LISP** (1959).

- No início dos anos 70 surge a programação lógica: **Prolog**.



Domínios da Programação
Software Básico

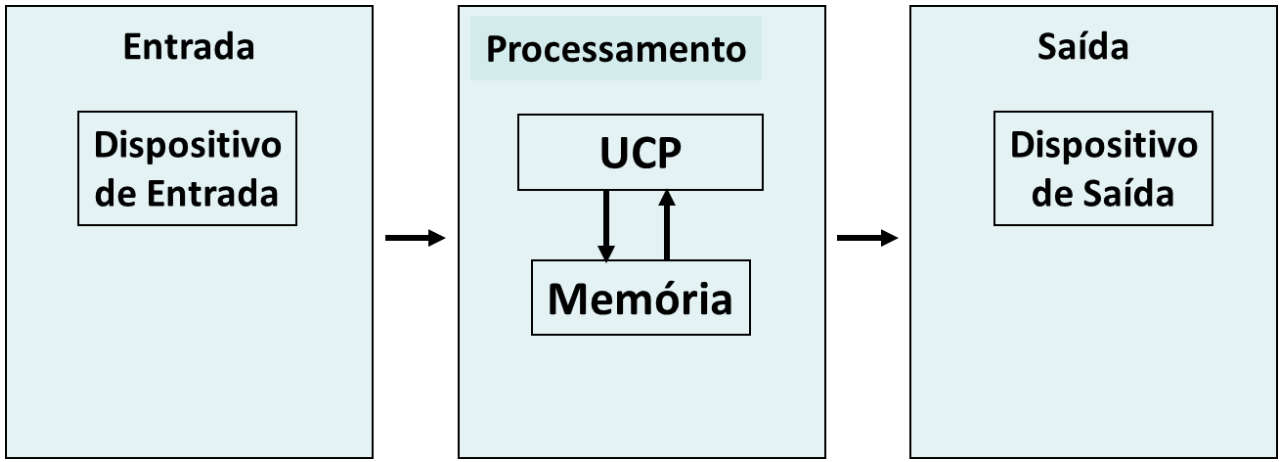
- O software básico (sistema operacional) deve possuir eficiência na execução por propiciar suporte a execução de outros aplicativos.

- As linguagens de programação para este tipo de sistema devem oferecer execução rápida e ter recursos de baixo nível que permitam ao software fazer interface com os dispositivos externos.

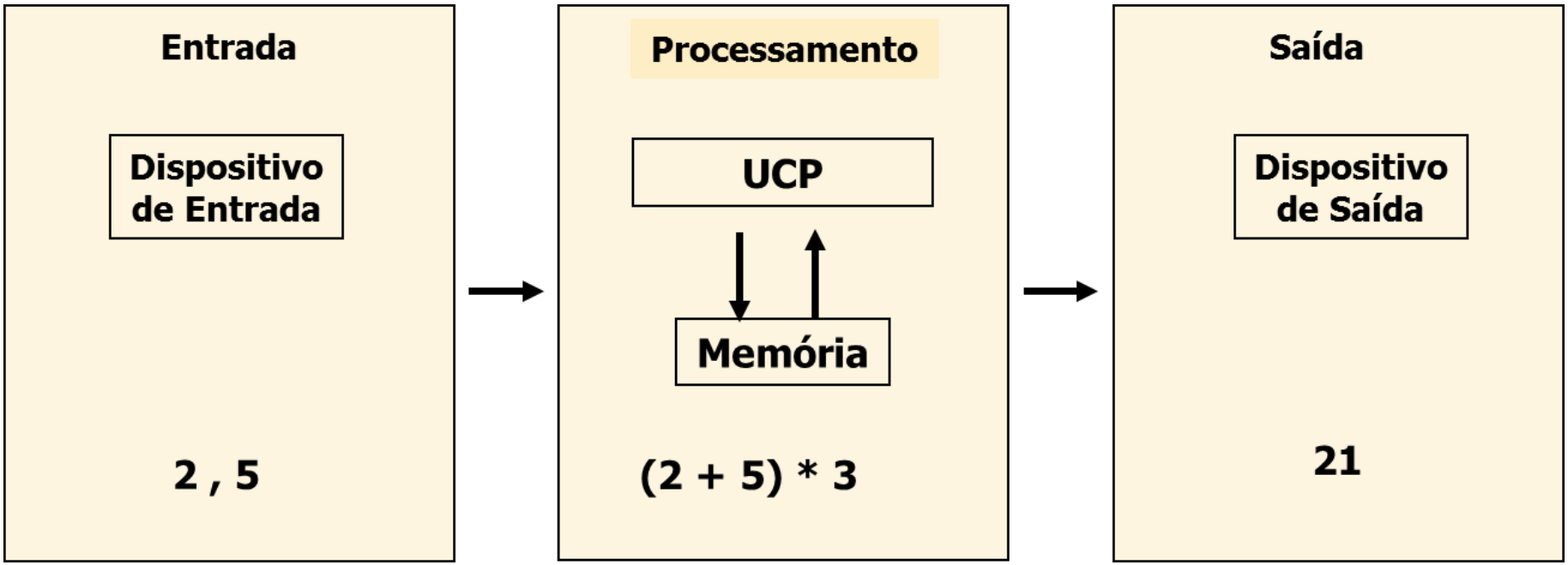
- O sistema operacional UNIX foi desenvolvido quase inteiramente em C (tornando-o fácil de portar para diferentes máquinas).

Programação

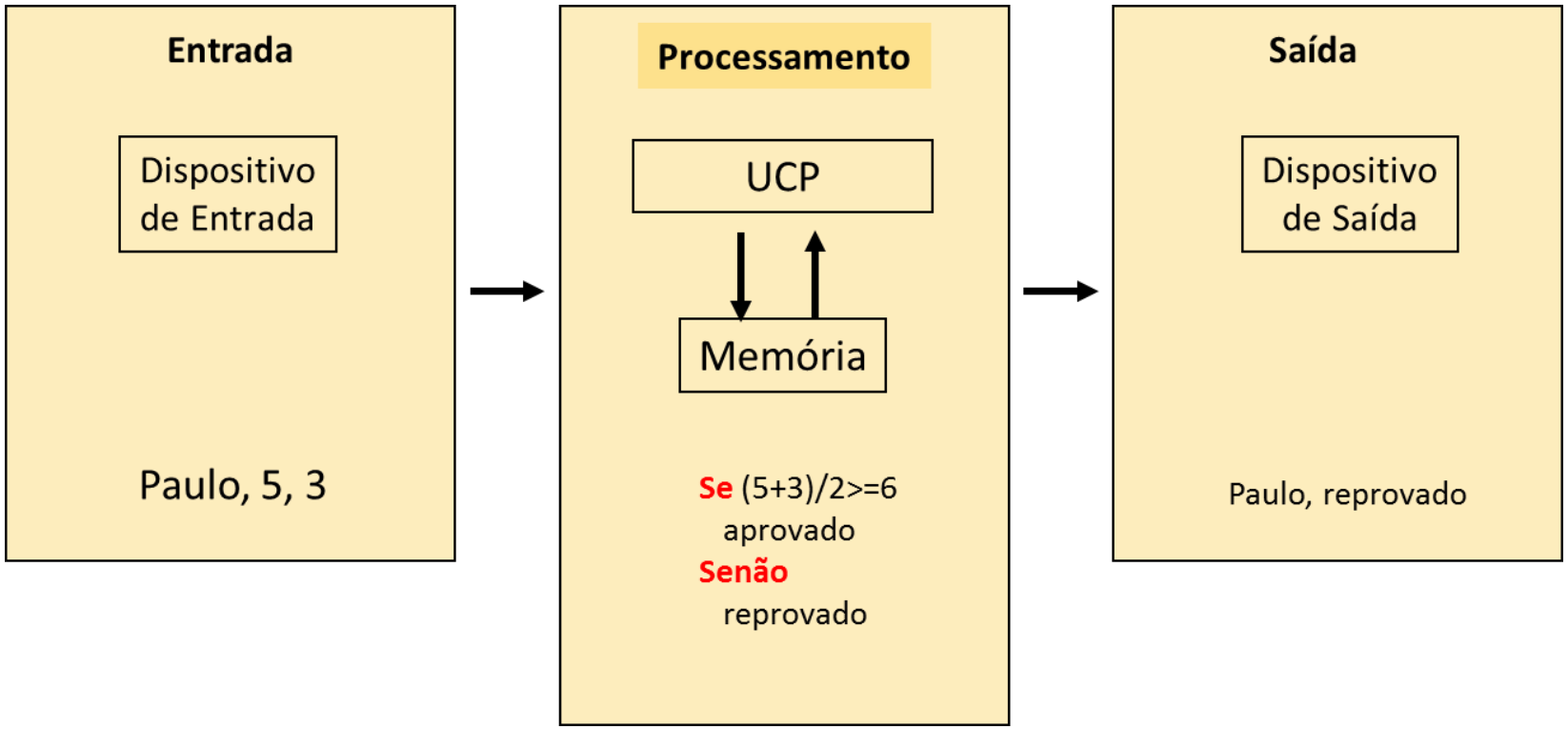
- ✓ A tarefa de programação é composta por 3 passos:



Programação



Programação



Linguagens de Programação

1 - Totalmente codificadas em **binário** (**0´s** e **1´s**)

2 - Usa instruções simbólicas para representar os **0´s** e **1´s**

3 - Voltadas para facilitar o raciocínio humano

Linguagem de Máquina	**Linguagem Assembly (*Mnemônica*)**	**Linguagem de Alto Nível**
0010 0001 1110	LOAD R1, val1	val2 = val1+val2
0010 0010 1111	LOAD R2, val2	
0001 0001 0010	ADD R1, R2	
0011 0001 1111	STORE R1, val2	

1

2

3

Noções de Lógica

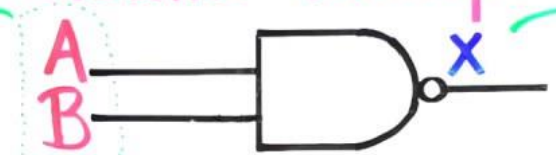
- **Proposição:** é um enunciado verbal, ao qual deve ser atribuído, sem **ambiguidade**, um valor **lógico** verdadeiro (**V**) ou falso (**F**);

  - Exemplos de **proposições**:
    - Antonio Carlos é Professor (**V**)
    - 1 + 6 = 9 (**F**)
    - 1 < 7 (**V**)

Algebra Booleana (Compuertas lógicas)

NAND

var. entrada



var. Salida

Modo de operación:

$X = \overline{AB}$

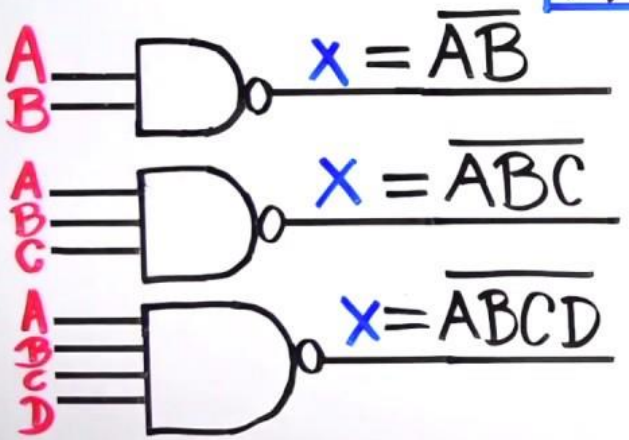
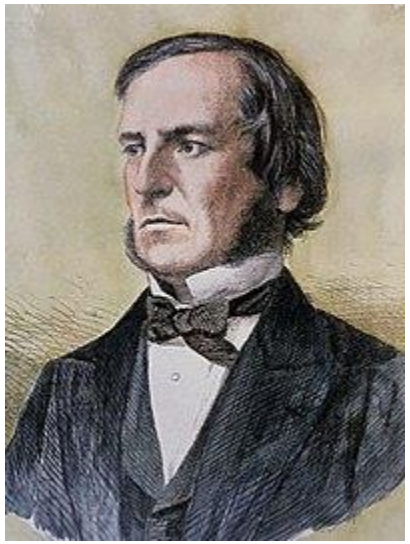


Tabla de verdad: Alto(1) Bajo(0)

Entradas A	Entradas B	AB	Salida $\overline{AB} = X$
0	0	0	$\overline{0} = 1$
0	1	0	$\overline{0} = 1$
1	0	0	$\overline{0} = 1$
1	1	1	$\overline{1} = 0$

Álgebra de Boole

- ✓ Para entendermos como bits são armazenados e manipulados no computador, é conveniente imaginar que o **bit 0** representa **FALSE** e o bit **1** representa **TRUE**;
- ✓ Operações que manipulam **TRUE** e **FALSE** são chamadas operações booleanas, em homenagem ao matemático **George Boole** (**1815-1864**).



Noções de Lógica

- ✓ **Operações Lógicas:** Usadas para formar novas proposições a partir de proposições existentes;

  - ❖ Considerando ***p*** e ***q*** duas proposições genéricas, pode-se aplicar as seguintes operações lógicas básicas sobre elas:

Operação	Símbolo	Significado
Negação	~	Não
Conjunção	^	E
Disjunção	v	OU

  - ❖ Definindo-se a **prioridade**:
    - Recomenda-se o uso de **parênteses**
    - Exemplo: ((p v q) ^ (~q))

Noções de Lógica

- ✓ **NOT** (**~**) troca o valor lógico. Se for **FALSO**, passa a ser **VERDADE** e vice-versa;
- ✓ **AND** (**^**) só é **V** quando ambas proposições também forem **V**. Basta uma proposição ser **F** para o resultado também ser **F**;
- ✓ **OR** (**V**) só é **F** quando ambas proposições também forem **F**. Basta uma proposição ser **V** para o resultado também ser **V**;

p	q	~p	p ^ q	p v q
V	V	F	V	V
V	F	F	F	V
F	V	V	F	V
F	F	V	F	F

Exercício

- ✓ Considerando ***p = V*** e ***q = F***, resolva as seguintes expressões lógicas:

~p
~q
p ^ q
p v q
(~p) ^ q
(~p) v q
p ^ (~q)
p v (~q)
(~p) ^ (~q)
(~p) v (~q)

Respostas

- ✓ Considerando **p = V** e **q = F**, resolva as seguintes expressões lógicas:

~p = **F**

~q = **V**

p ^ q = **F**

p v q = **V**

(~p) ^ q = **F**

(~p) v q = **F**

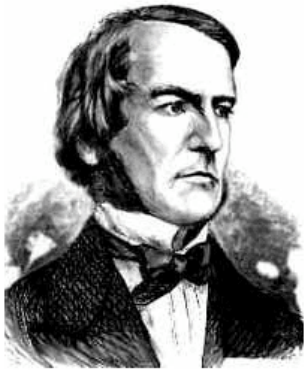
p ^ (~q) = **V**

p v (~q) = **V**

(~p) ^ (~q) = **F**

(~p) v (~q) = **V**

George Boole (1815–1864)



A	B	A • B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Bits e Memória

- Toda a informação armazenada em computadores é codificada em padrões de **0s** e **1s**;
- Estes dígitos são chamados **BITS** (abreviação de **binary digits**);
- Padrões de bits podem representar **números**, **textos**, **imagens**, **sons**, etc.

> Para entendermos como bits são armazenados e manipulados no computador, é conveniente imaginar que o **bit 0** representa **FALSE** e o bit **1** representa **TRUE**;

Operações Booleanas

AND

	0		0		1		1
AND	0	AND	1	AND	0	AND	1
	0		0		0		1

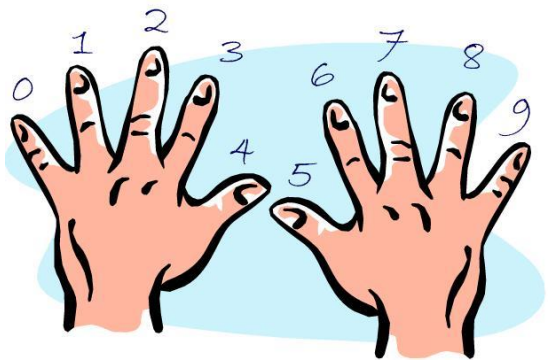
OR

	0		0		1		1
OR	0	OR	1	OR	0	OR	1
	0		1		1		1

O Sistema Binário

Sistema Decimal - Origem

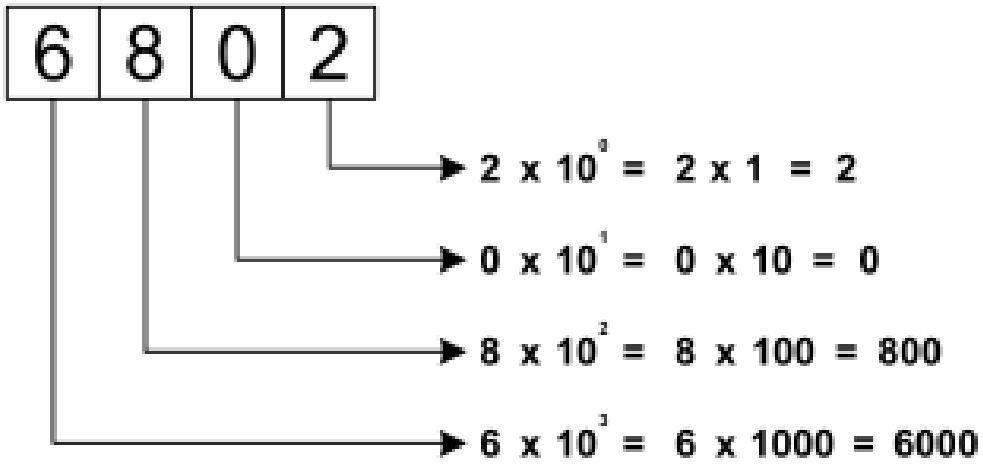
- Quando um animal ia para o pasto, uma pedra era colocada em uma sacola.
- Quando o animal voltava do pasto, a pedra era retirada da sacola.
- Se sobrasse alguma pedra significaria que algum animal não voltou.



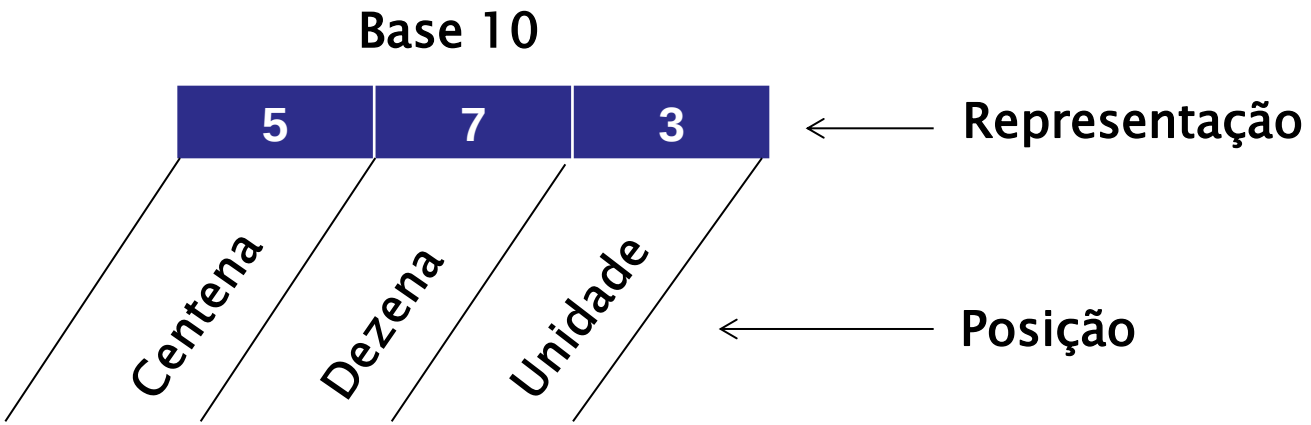
Sistema Decimal - Montagem

- O sistema decimal possui dez símbolos:

0 1 2 3 4 5 6 7 8 9



Sistema Decimal



Sistema Binário

Utiliza apenas dois símbolos:

0 - Desligado – OFF - Falso - F

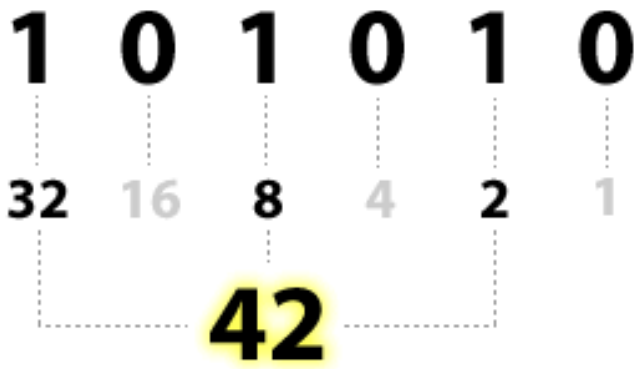
1 - Ligado – ON - Verdadeiro - V

Sistema Binário - Exemplos

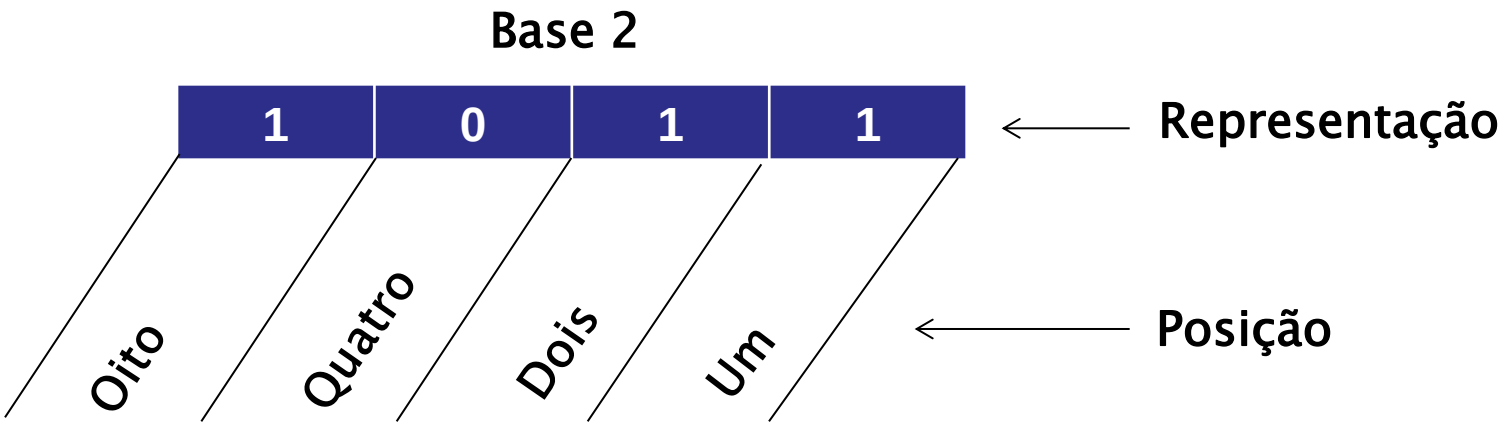
• 11001010	**8 bits**
• 110	**3 bits**
• 1	**1 bit**
• 11110	**5 bits**

Sistema Binário

Baseado em potência de 2

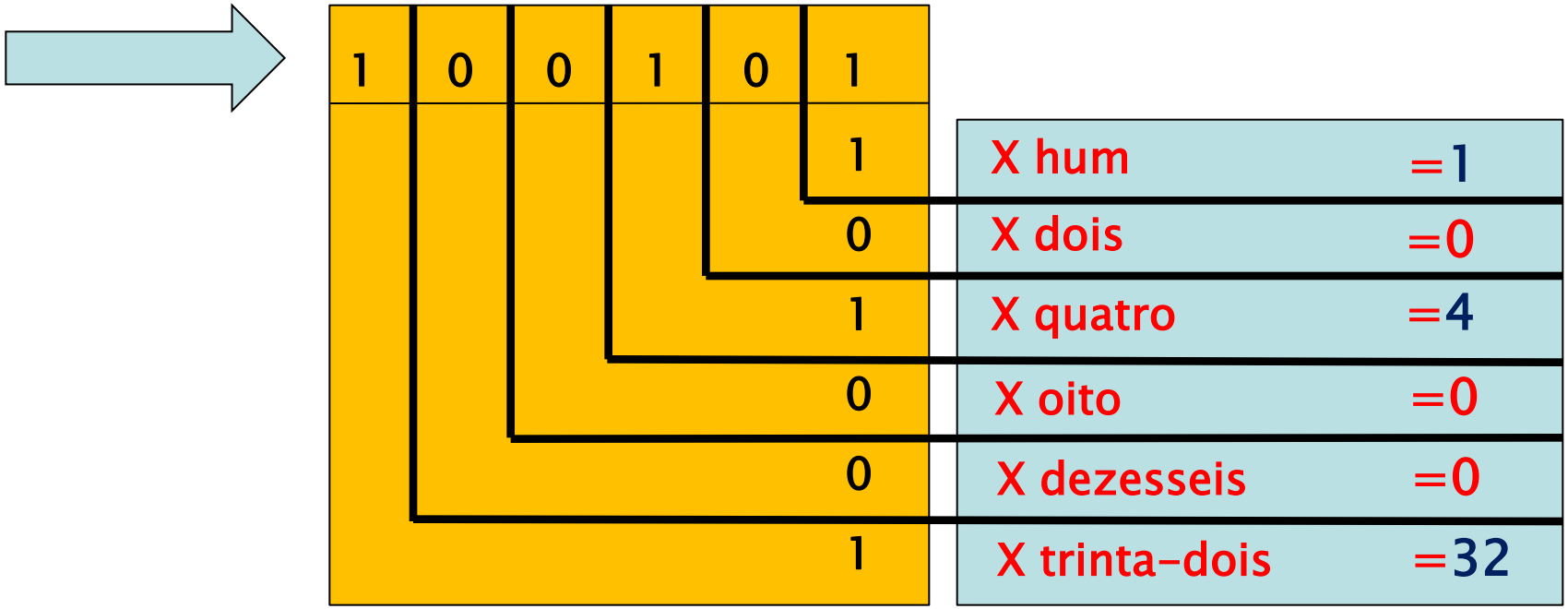


Sistema Binário



Decodificando o número binário

Padrão de Bits

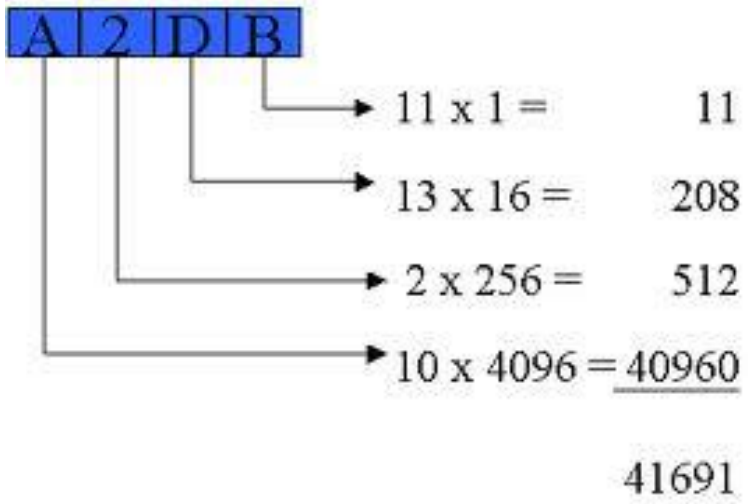


Total: 37

O Sistema Hexadecimal

Notação Hexadecimal

- Ao considerarmos as atividades internas de um computador, devemos lidar com padrões de bits.
- Um longo string de bits é frequentemente chamado de **stream**.
- Infelizmente, **streams** são difíceis de serem compreendidos por humanos.
- Para simplificar a representação de tais padrões de bits, usualmente, empregamos uma notação chamada **hexadecimal**.



Código Hexadecimal

Bit Pattern	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Conversão Decimal - Hexa

$$(45)_{10} = (2D)_{16}$$

Basta dividir por 16 e tomar os restos:

2 ÷ 16 = 0 Resto = 2

45 ÷ 16 = 2 Resto = 13

2 D

Conversão Hexa – Decimal

- Para converter um número hexadecimal em decimal, utilizamos o mesmo algoritmo utilizada na conversão binário para decimal, sendo que a **base 2 é trocada por 16**;

Exemplo, para converter **B2A** em decimal:

- ✓ A => $10*(16)^0$ = 10
- ✓ 2 => $2*(16)^1$ = 32
- ✓ B => $11*(16)^2$ = 2816

2858

Tabela de conversão Hexa Binário

Hexa	Binário	Hexa	Binário	Hexa	Binário
0	0000	6	0110	C	1100
1	0001	7	0111	D	1101
2	0010	8	1000	E	1110
3	0011	9	1001	F	1111
4	0100	A	1010		
5	0101	B	1011		

Conversão Hexa-Binário

$(E0A2)_{16}=(\ 1110\ 0000\ 1010\ 0010\ )_2$

Utilizar a tabela de conversão para cada algarismo hexadecimal:

E = 1110
0 = 0000
A = 1010
2 = 0010

E	0	A	2
1110	0000	1010	0010

Conversão Binário-Hexa

$$(111\ 1110\ 0011)_2 = (\ 7E3)_{16}$$

- Separar o número binário em grupos de 4 bits da direita para a esquerda:

  111 1110 0011

- Completar 4 bits no 1º grupo:

  0111 1110 0011

- Consultar a tabela:

  0111=7 1110=E 0011=3

Exercício

- Empregue notação hexadecimal para representar os seguintes padrões de bits:

a) 0110101011110010

b) 111010001111000000111100

c) 10001000000100010000111110001

Exercício

- Que padrões de bits são representados pelos seguintes padrões hexadecimais?

a) 5DF90

b) 820F

c) ABCF

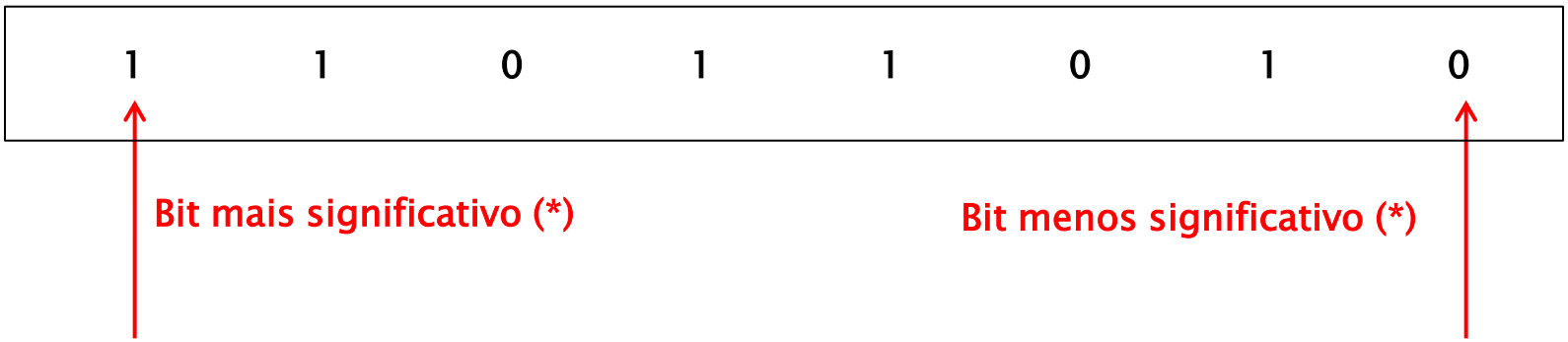
d) 0100

Memória Principal

- Computadores manuseiam informações por meio de uma grande coleção de circuitos capazes de armazenar um simples bit.
- Este repositório de bits é conhecido por **memória principal**.

Organização da Memória

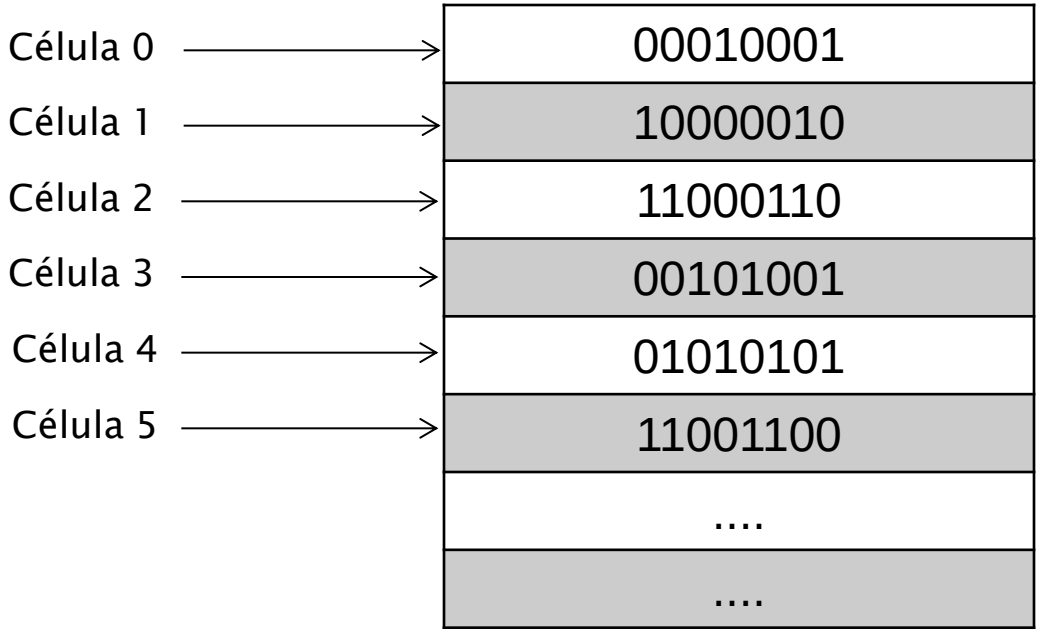
- A memória principal é organizada em unidades gerenciáveis chamadas células.
- Tipicamente o tamanho de uma célula é **8 bits.**
- Um string de **8 bits** é denominado **byte**.
- O bit mais a esquerda é chamado **bit de alta ordem** (high-order end)
- O bit mais a direita é chamado **bit de baixa ordem** (low-order end)



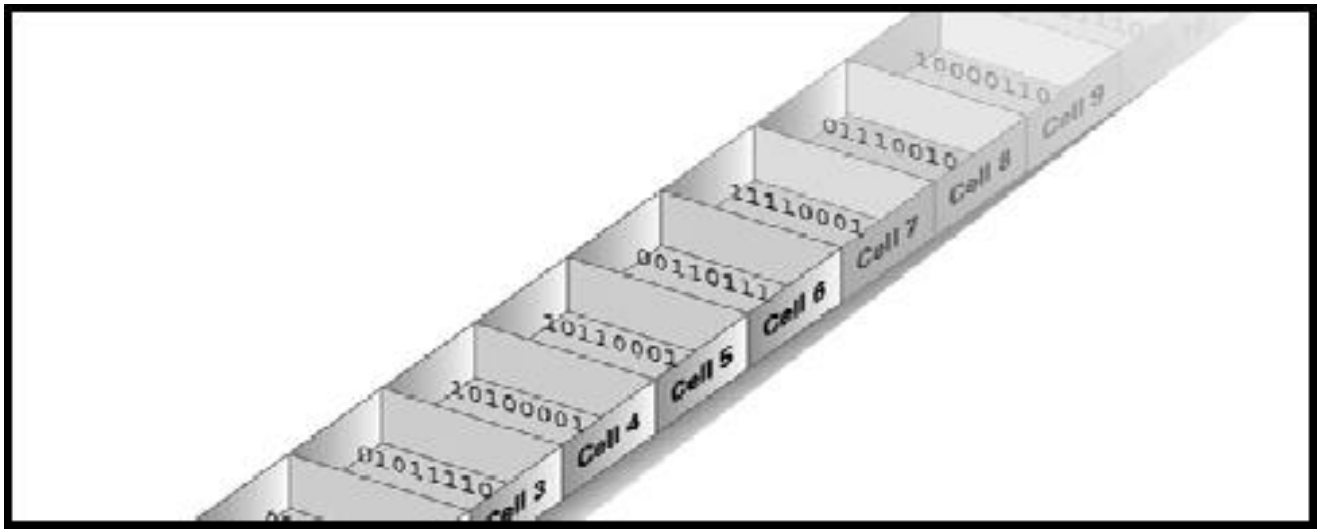
Como se acessa uma célula da memória ?

Endereçamento

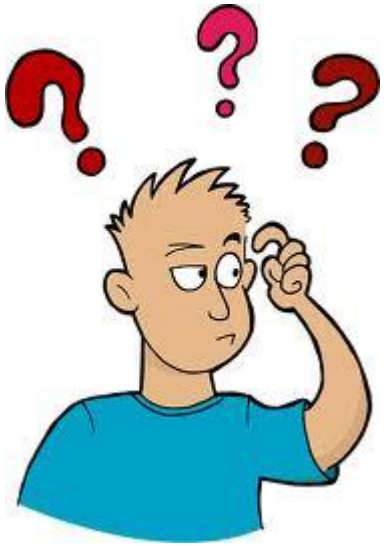
- Células individuais são acessadas por meio de um nome único chamado **endereço**.
- **Endereços** são sempre valores numéricos, iniciando com **zero**.



Endereçamento

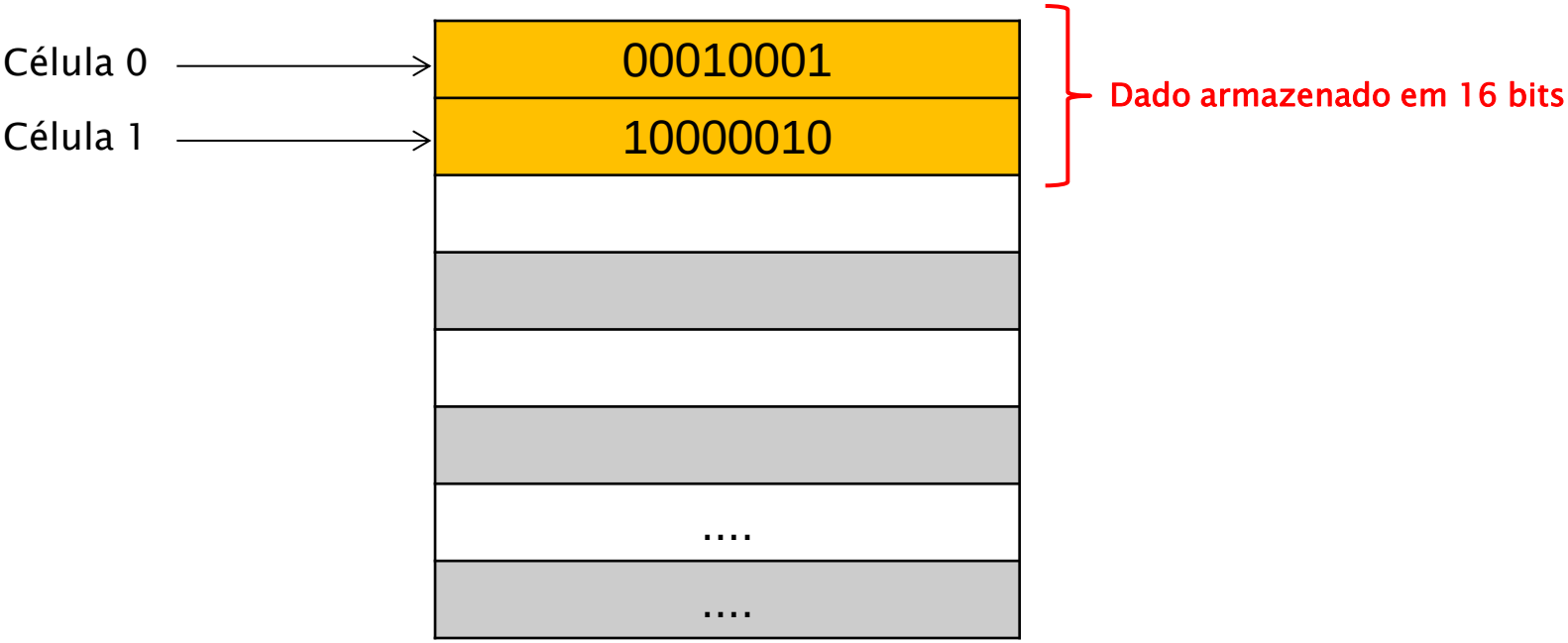


Como se armazena um dado de 16 bits ?



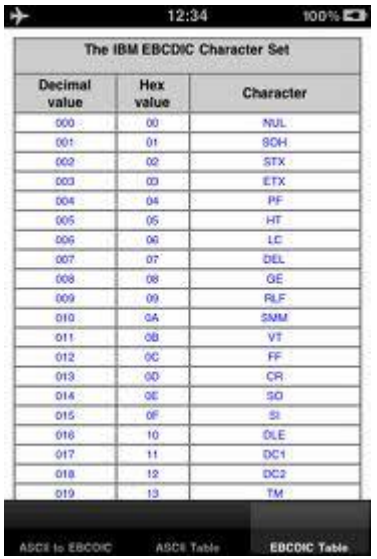
Armazenamento de dado com 16 bits

- O armazenamento de um dado de **16 bits** é feito por meio do emprego de duas células consecutivas.

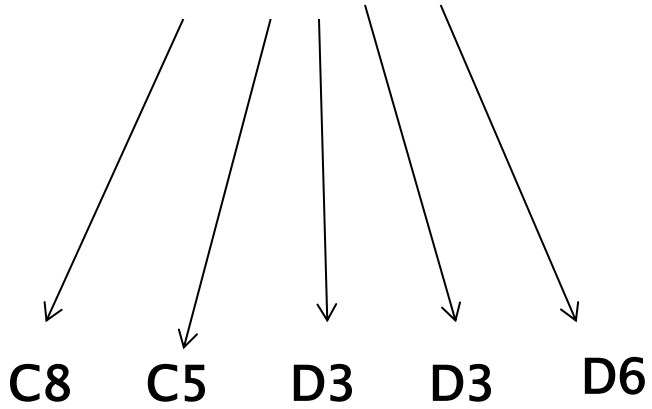


Representando texto por meio de um padrão de bits

- Parão **ASCII**
- Padrão **EBCDIC**
- Padrão **UNICODE**



Mensagem "HELLO" em EBCDIC



Representando valores numéricos

- Se usarmos código ASCII para representar números, o maior número armazenado em 16 bits seria 99.
- No entanto, em binário podemos armazenar em 16 bits qualquer número inteiro entre 0 e 65535.
- Por isso, a notação binária é usada de forma extensiva para representar valores numéricos.
- Números negativos são comumente representados em **COMPLEMENTO de 2**.
- Números com partes fracionárias usam outra técnica chamada notação **PONTO FLUTUANTE**.