Sistemas numéricos: conceitos, simbologia e representação de base numérica

Eduardo Furlan Miranda

2024-09-28

Baseado em: Tangon, LG; Santos, RC. Arquitetura e organização de computadores. EDE, 2016. ISBN 978-85-8482-382-6.

Sistemas de numeração

- Sistema Decimal: este é o mais utilizado, baseado em 10 dígitos (0-9). É usado para a maioria das contagens e cálculos diários

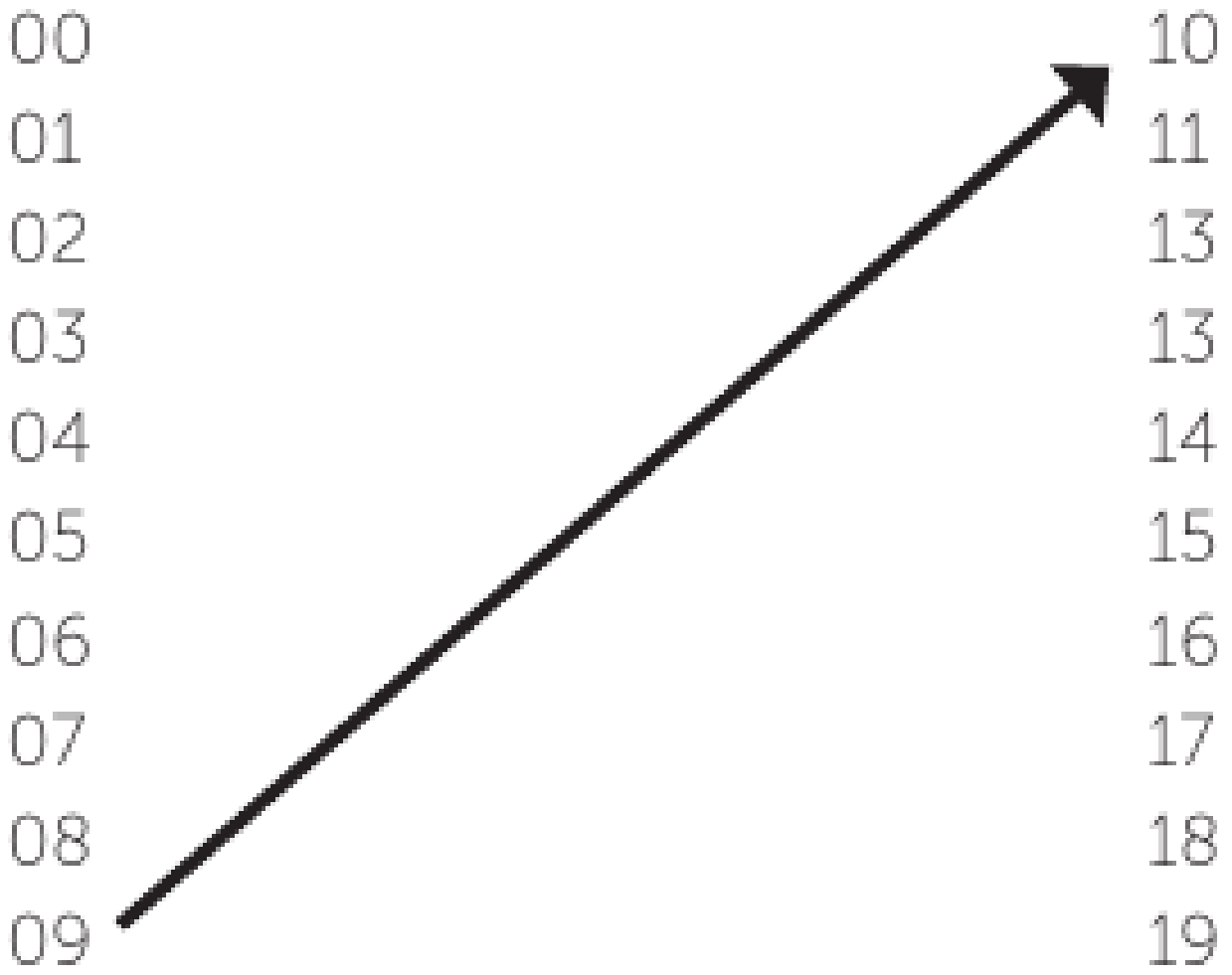
- Sistema Sexagesimal: baseado no número 60, é usado para medir tempo (60 segundos em um minuto, 60 minutos em uma hora) e ângulos (360 graus em um círculo, que é 6 vezes 60)

- Sistema Duodecimal: baseado no número 12, é frequentemente usado em contextos como a contagem de meses no ano (12 meses) e na venda de itens em dúzias

- Sistema Binário: utilizado principalmente em computação, é baseado em 2 dígitos (0 e 1). Todos os dados digitais são processados usando este sistema1

- Sistema Romano: ainda usado em contextos específicos, como a numeração de séculos e capítulos de livros. Utiliza letras como I, V, X, L, C, D e M para representar números

Sistema de numeração decimal (ou base 10)



Número 387

Figura 3.1 | Representação matemática de um número decimal

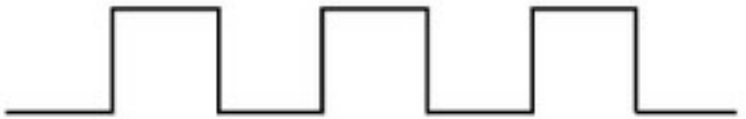
10^2	10^1	10^0
3	8	7

- $(3 \times 10^2) + (8 \times 10^1) + (7 \times 10^0) =$

- $300 + 80 + 7 = 387_{10}$ (387 na base 10)

Numeração binária

- Computadores: 0 e 1 representando níveis de tensão
  - O = geralmente próximo a 0 V
  - 1 = próximo a 3,3V, 5V, etc.



00	100
01	101
10	110
11	111

- 00110111_2
- $(0 * 2^7) + (0 * 2^6) + (1 * 2^5) + (1 * 2^4) + (0 * 2^3) + (1 * 2^2) + (1 * 2^1) + (1 * 2^0) = 55$
- 0 ou 1 também é chamado de bit (Binary Digit)
- 8 bits = 1 byte
- Internamente o computador usa apenas binário

- Octal e hexadecimal são usados pois são mais curtos e relativamente fáceis de converter de/para binário

- $55_{10} = 110111_2 = 67_8 = 37_{16}$

- $167_8 = (1 * 8^2) + (6 * 8^1) + (7 * 8^0) = 64 + 48 + 7 = 119_{10}$

- Octal:

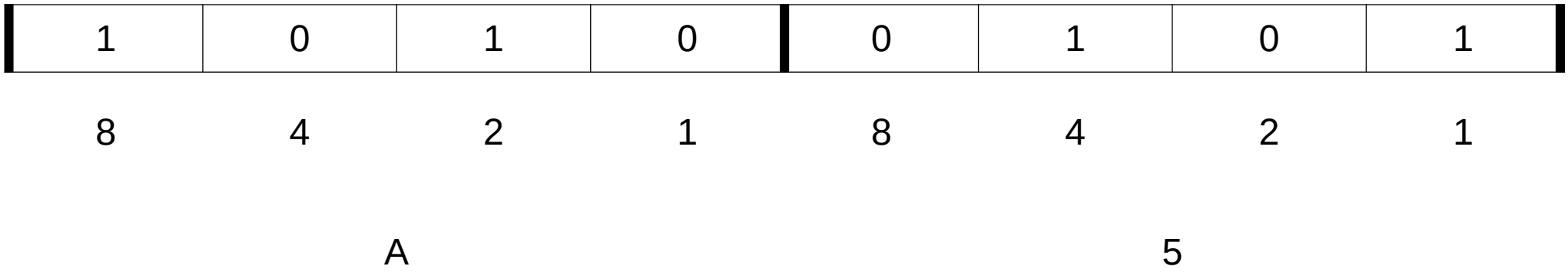
0	10	20
1	11	21
2	12	22
3	13	23
4	14	24
5	15	25
6	16	26
7	17	27

Hexadeximal

DECIMAL	HEXADECIMAL
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

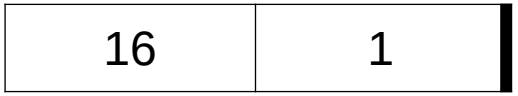
- $2F4_{16} = 2^2\ F^1\ 4^0 =$
- $2^2\ 15^1\ 4^0 =$
- $(2 * 16^2) + (15 * 16^1) + (4 * 16^0) =$
- $512 + 240 + 4 = 756_{10}$

DECIMAL	BINÁRIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F


1 0 1 0 0 1 0 1
8 4 2 1 8 4 2 1
A
5

128	64	32	16	8	4	2	1
1	0	1	0	0	1	0	1

128
32
4
1
+
165

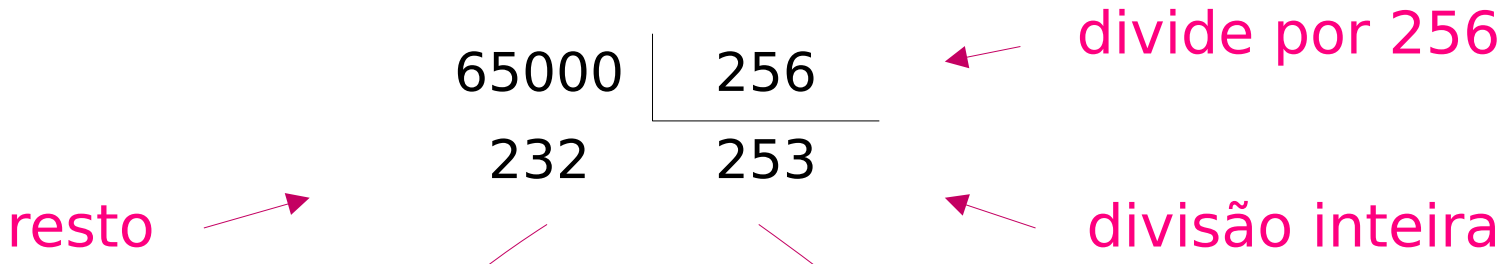


A 5

$$\begin{array}{r} 16\text{x}10 \\ 5 \\ +\quad\quad \\ \hline 165 \end{array}$$

Armazenar o número 65000_{10} na memória

Como em um byte “cabe” um número entre 0 e 255, podemos pensar em usar a base 256, cada byte sendo um "dígito"



Endereço	Conteúdo (Decimal)	Conteúdo (Hex)
0000	232_{10}	$E8_{16}$
0001	253_{10}	FD_{16}

Ordem dos bytes: *Little-Endian* na arquitetura x86