Sistemas de Numeração e Códigos

123 = Cento e Vinte e Três! Porquê?

Qual o algoritmo de cálculo de um número?

$123 = 1*10^2 + 2*10^1 + 3*10^0$

$123{,}95 = 1*10^2 + 2*10^1 + 3*10^0 + 9*10^{-1} + 5*10^{-2}$

- Símbolos válidos:0,1,2,3,4,5,6,7,8,9 total 10, logo **Base 10**.

Sistema Binário

- Sistema binário, Símbolos válidos:0,1 - total 2, logo **Base 2**

$$10011_2 = 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =$$
$$= 1*16 + 0*8 + 0*4 + 1*2 + 1*1 = 19_{10}$$

$$101{,}001_2 = 1*2^2 + 0*2^1 + 1*2^0 + 0*2^{-1} + 0*2^{-2} + 1*2^{-3} =$$
$$= 1*4 + 0*2 + 1*1 + 0*0{,}5 + 0*0{,}25 + 1*0{,}125 =$$
$$= 5{,}125_{10}$$

- Bit da esquerda: MSB (*Most Significant Bit*)
- Bit da direita: LSB (*Least Significant Bit*)

  Ex: 2039 — 2 é o dígito mais significativo (peso 2000)

  9 é o dígito menos significativo (peso 1)

Octal e Hexadecimal

- Outros Sistemas de numeração usados são:
  Hexadecimal (Base 16) e Octal (Base 8)

- Estas Bases são potências de 2 que é a base do sistema binário

  Octal = 8 = 2^3 Hexadecimal = 16 = 2^4

Um dígito de um n.º octal representa 3 dígitos (bits) de um n.º binário.

Um dígito de um n.º hexadecimal representa 4 bits de um n.º binário.

Sistema Octal

- **Oito símbolos (0…7)**

  Utilizado na representação de números binários pois permite reduzir o tamanho do número.
- **Símbolos**: 0, 1, 2, 3, 4, 5, 6, 7
- **Peso**: 1, 8, 64, 512, 4096, …
- **Base**: 8 – representada por **o**
- **Exemplos:**

$17_o = 1*8+7 = 15_{10} = 001\ 111_2$

$55_o = 5*8+5 = 45_{10} = 101\ 101_2$

$124_o = 1*64+2*8+4 = \ldots\ldots = 001\ 010\ 100_2$

Sistema Hexadecimal

- **Dezasseis símbolos (0…F)**

  Utilizado na representação de números binários pois permite reduzir o tamanho do número, os microprocessadores usam múltiplos de 8 bits (1 byte).

- **Símbolos**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A(10), B(11), C(12), D(13), E(14), F(15)

- **Peso**: 1, 16, 256, 4096, 65536, …

- **Base**: 16 – representada por x ou h

- **Exemplos:**

  $AF_h = 10*16+15 = 175_{10} = 1010\ 1111_2 = 257_o$

  $1DEC_h = 1*16^3+13*16^2+14*16^1+12 = 0001\ 1101\ 1110\ 1100_2 = 16754_o$

  $ABBA_h = 1010\ 1011\ 1011\ 1010_2 = 125672_o$

Tabela Conversão

Table 2–1
Binary, decimal, octal, and hexadecimal numbers.

Binary	Decimal	Octal	3-Bit String	Hexadecimal	4-Bit String
0	0	0	000	0	0000
1	1	1	001	1	0001
10	2	2	010	2	0010
11	3	3	011	3	0011
100	4	4	100	4	0100
101	5	5	101	5	0101
110	6	6	110	6	0110
111	7	7	111	7	0111
1000	8	10	—	8	1000
1001	9	11	—	9	1001
1010	10	12	—	A	1010
1011	11	13	—	B	1011
1100	12	14	—	C	1100
1101	13	15	—	D	1101
1110	14	16	—	E	1110
1111	15	17	—	F	1111

Conversões

Table 2–2 (continued)

Conversion	Method	Example
Decimal to Binary	Division	$108_{10} = 1101100_2$ $108_{10} \div 2 = 54$ remainder 0 (LSB) $\div 2 = 27$ remainder 0 $\div 2 = 13$ remainder 1 $\div 2 = 6$ remainder 1 $\div 2 = 3$ remainder 0 $\div 2 = 1$ remainder 1 $\div 2 = 0$ remainder 1 (MSB)
Octal	Division	$108_{10} = 154_8$ $108_{10} \div 8 = 13$ remainder 4 (least significant digit) $\div 8 = 1$ remainder 5 $\div 8 = 0$ remainder 1 (most significant digit)
Hexadecimal	Division	$108_{10} = 6C_{16}$ $108_{10} \div 16 = 6$ remainder 12 (least significant digit) $\div 16 = 0$ remainder 6 (most significant digit)

Soma e Subtracção Binária

- Tabela de Verdade da Adição e Subtracção binária

Table 2–3
Binary addition and subtraction table.

c_{in} or b_{in}	x	y	c_{out}	s	b_{out}	d
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	1	0	1
0	1	1	1	0	0	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

- Exemplos de Soma

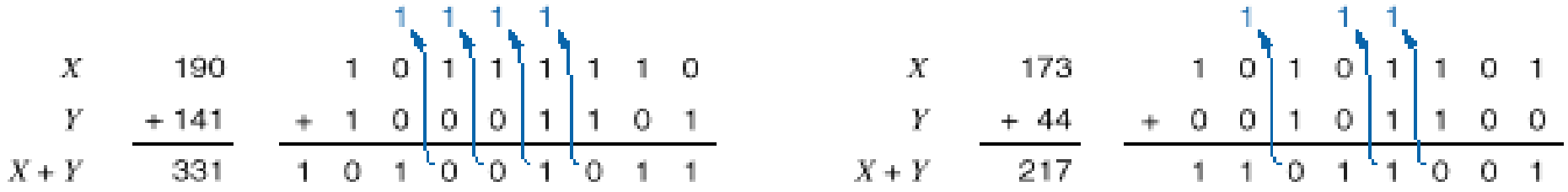


Figure 2–1 Examples of decimal and corresponding binary additions.

Soma e Subtracção Binária

- Exemplos de Subtracção

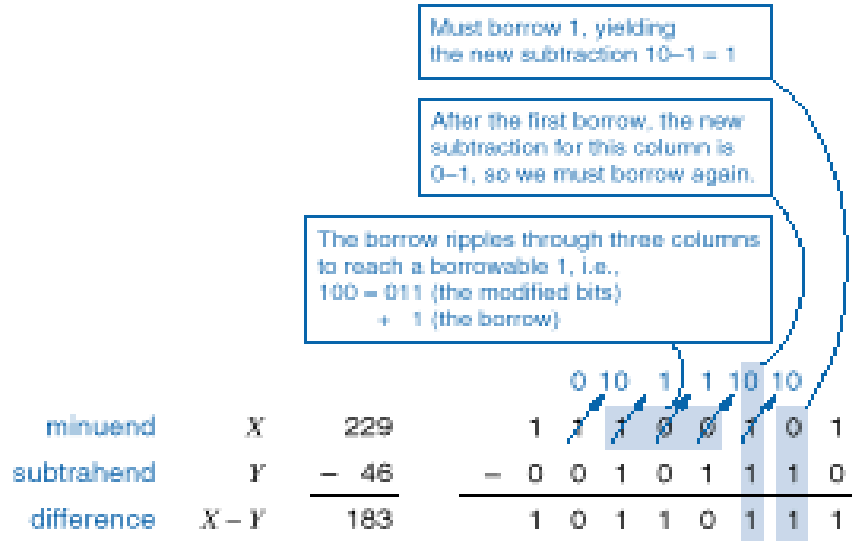
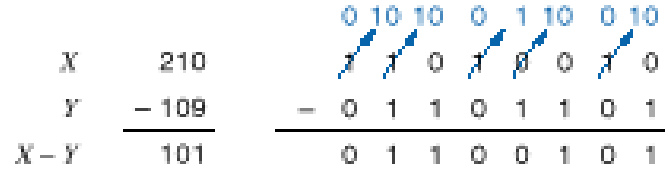


Figure 2–2
Examples of decimal and corresponding binary subtractions.



- Tabelas de Adição e Subtracção podem ser criadas para qualquer Base (Octal, Hexadecimal, 5 ...)

Representação Números Negativos

- A realização de operações matemáticas exige números negativos.

- Diferentes modos de representação de números negativos:
  - Sinal + Amplitude,

    $01010101_2 = +85_{10}$ $11010101_2 = -85_{10}$

    Primeiro bit para sinal os outros amplitude.
  - Complemento para 1

    $01010101_2 = +85_{10}$ $10101010_2 = -85_{10}$

    Complementação bit a bit.
  - Complemento para 2

    $01010101_2 = +85_{10}$ $10101011_2 = -85_{10}$

    Complementação bit a bit e ainda é adicionado um.

    Regra simples para conversão de binário para complemento para dois.

    Representação preferida para realização de operações aritméticas.

Representação Números Negativos

Tabela de representação de números negativos:

Table 2–6
Decimal and 4-bit numbers.

Decimal	Two's Complement	Ones' Complement	Signed Magnitude	Excess 2^{m-1}
−8	1000	—	—	0000
−7	1001	1000	1111	0001
−6	1010	1001	1110	0010
−5	1011	1010	1101	0011
−4	1100	1011	1100	0100
−3	1101	1100	1011	0101
−2	1110	1101	1010	0110
−1	1111	1110	1001	0111
0	0000	1111 or 0000	1000 or 0000	1000
1	0001	0001	0001	1001
2	0010	0010	0010	1010
3	0011	0011	0011	1011
4	0100	0100	0100	1100
5	0101	0101	0101	1101
6	0110	0110	0110	1110
7	0111	0111	0111	1111

Complemento para Dois

- Cada número é obtido do anterior somando “1” e ignorando o transporte (carry) que surja na 5ª posição.
- Como a adição é apenas uma extensão da contagem, os números em complemento para 2 podem ser somados através da adição binária normal, ignorando qualquer carry a partir do MSB.
- O resultado será sempre correcto desde que o intervalo do sistema de numeração seja respeitado.

Overflow

- Quando a operação de adição produz um resultado que excede um dos limites do sistema numérico, diz-se que ocorre o *overflow*.
- A adição de dois números com sinais diferentes nunca produz o *overflow*. A soma de dois números com sinais idênticos pode provocar o *overflow*.
- Há uma regra simples para detectar o *overflow* na adição:

  - Quando os números a somar têm o mesmo sinal e o resultado tem sinal diferente;

  ou

  - Se os bits Cin (*Carry In*) e Cout (*Carry Out*) da posição do sinal forem diferentes.

Sistemas de Numeração e Códigos

- Códigos Decimais

Table 2–9
Decimal codes.

Decimal digit	BCD (8421)	2421	Excess-3	Biquinary	1-out-of-10
0	0000	0000	0011	0100001	1000000000
1	0001	0001	0100	0100010	0100000000
2	0010	0010	0101	0100100	0010000000
3	0011	0011	0110	0101000	0001000000
4	0100	0100	0111	0110000	0000100000
5	0101	1011	1000	1000001	0000010000
6	0110	1100	1001	1000010	0000001000
7	0111	1101	1010	1000100	0000000100
8	1000	1110	1011	1001000	0000000010
9	1001	1111	1100	1010000	0000000001
Unused code words					
	1010	0101	0000	0000000	0000000000
	1011	0110	0001	0000001	0000000011
	1100	0111	0010	0000010	0000000101
	1101	1000	1101	0000011	0000000110
	1110	1001	1110	0000101	0000000111
	1111	1010	1111	...	...

- BCD – *Binary Coded Decimal*

BCD – *Binary Coded Decimal*

- Codifica os dígitos 0 a 9 através de representações binárias sem sinal de 4-bit: 0000_b a 1001_b. As representações 1010_b até 1111_b não são utilizadas.

- As conversões entre BCD e representações digitais são simplesmente uma substituição directa de 4-bit por cada dígito decimal.

- Alguns programas colocam dois dígitos BCD num byte (*Packed BCD*). O objectivo é minimizar a memória ocupada. Um byte representa assim números entre 0 e 99.

Adição de nºs em BCD

- Idêntica à operação de adição de números de 4-bit sem sinal. No entanto, se o resultado exceder 1001 tem de ser somado 6 (0110) ao resultado de modo a obter o valor correcto.

- O BCD é um código onde cada dígito decimal pode ser obtido a partir da sua palavra de código, atribuindo a cada bit da palavra um peso específico, os pesos são: 8,4,2 e 1. Por esta razão o código BCD é por vezes referido como código 8421.

Sistemas de Numeração e Códigos

- Tabela ASCII

Table 2–11
American Standard Code for Information Interchange (ASCII), Standard No. X3.4–1968 of the American National Standards Institute.

		$b_6b_5b_4$ (column)							
$b_3b_2b_1b_0$	Row (hex)	000 0	001 1	010 2	011 3	100 4	101 5	110 6	111 7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O	_	o	DEL

Sistemas de Numeração e Códigos

- ASCII

Table 2–11
(continued)

Control codes			
NUL	Null	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronize
BEL	Bell	ETB	End transmitted block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete or rubout

Sistemas de Numeração e Códigos

- Código de Gray

Table 2–10
A comparison of 3-bit binary code and Gray code.

Decimal number	Binary code	Gray code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Figure 2–6
A mechanical encoding disk using a 3-bit Gray code.