UNIVERSIDADE EDUARDO MONDLANE

FACULDADE DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA

Electrónica Digital

Engº. Albino B Cuinhane

AULA TEÓRICA 2
SUMÁRIO

Capítulo 2
Representação de Informação em Sistemas Digitais

2.1 Sistemas de Numeração

2.1.1 Introdução

•**D**esde tempos remotos o homem sentiu a necessidade de utilizar os sistemas numéricos para saber a quantidade de bens que tinha.

•Existem vários sistemas de numeração mas pela sua simplicidade e praticabilidade destacam-se:
- Sistema de numeração decimal,
- Sistema de numeração octal,
- Sistema de numeração hexadecimal e
- Sistema de numeração binário

•O último (e os dois anteriores) é o mais usado nos Sistemas Digitais dada a simplicidade de ser "percebido" por estes.

2.1.1 Introdução

O sistema de numeração decimal é o mais usado no dia-a-dia e é o mais facilmente compreensível para os humanos. É o sistema mais importante no tempo actual e para sempre.

Baseia-se num conjunto de 10 algarismos (também chamados coeficientes) com os quais podemos formar qualquer número, representando qualquer quantidade.

São os seguintes os algarismos do sistema de numeração decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9

2.1.1 Introdução

Teoricamente qualquer quantidade pode ser representada no sistema numérico de base r (= 2, 8, 10, 16, ...) pelo polinómio:

$$N_r = d_n.r^n + d_{n-1}.r^{n-1} + \ldots + d_1.r^1 + d_0 + d_{-1}.r^{-1} + d_{-2}.r^{-2} + \ldots + d_{-m}.r^{-m} \qquad (2.1)$$

Em que:

N_r é um número (quantidade) no sistema de numeração da base r
d_n é um dos coeficientes do sistema de numeração em causa e
n representa o peso desse coeficiente no número

Exemplos:

a) $872_{10} = 8*10^2 + 7*10^1 + 2*10^0$

b) $872{,}27_{10} = 8*10^2 + 7*10^1 + 2*10^0 + 2*10^{-1} + 7*10^{-2}$

c) $872{,}27_{16} = 8*16^2 + 7*16^1 + 2*16^0 + 2*16^{-1} + 7*16^{-2}$

2.1.1 Introdução

O sistema de numeração na base *r* tem *r* coeficientes sendo *0* o menor de todos e *r*-1 o maior. A quantidade *r* é representada pelo retorno ao inicio do ciclo de contagem buscando o segundo dígito para coadjuvar o primeiro, e assim sucessivamente (Fig. 2.1)

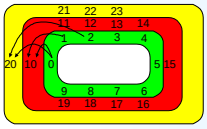


Fig. 2.1. Ciclo combinatório para geração de números

A Fig.2.1 foi gerada com base na fórmula (2.1). O ciclo nuclear tem peso 0, o seguinte tem peso 1 e assim sucessivamente.
Pela sua importância nos dias de hoje, iremos apresentar neste trabalho o sistema de numeração binária e o sistema de numeração hexadecimal.

2.1.2 Sistema de Numeração Binária

O sistema de numeração binária comporta apenas dois dígitos, coeficientes ou bits, nomeadamente 0 e 1. A base do sistema de numeração binária é 2. De (2.1) vemos que podemos representar no sistema de numeração binária qualquer quantidade pela assim:

$$N_2 = d_n.2^n + d_{n-1}.2^{n-1} + \ldots + d_1.2^1 + d_0 + d_{-1}.2^{-1} + d_{-2}.2^{-2} + \ldots + d_{-m}.2^{-m} \qquad (2.2)$$

Sendo:

$d = 0$ ou $d = 1$

Exemplos:

a) $101_2 = 1*2^2 + 0*2^1 + 1*2^0$

b) $101,11_2 = 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} + 1*2^{-2}$

2.1.2 Sistema de Numeração Binária

Posto isto obteremos a seguinte equivalência entre o sistema de numeração decimal e sistema de numeração binária:

Base 10	Base 2	De facto...
0	0	$0*2^0 = (0)_{10} = 0_{10}$
1	1	$1*2^0 = (1)_{10} = 1_{10}$
2	10	$1*2^1 + 0*2^0 = (2+0)_{10} = 2_{10}$
3	11	$1*2^1 + 1*2^0 = (2+1)_{10} = 3_{10}$
4	100	$1*2^2 + 0*2^1 + 0*2^0 = (4+0+0)_{10} = 4_{10}$
5	101	$1*2^2 + 0*2^1 + 1*2^0 = (4+0+1)_{10} = 5_{10}$
6	110	$1*2^2 + 1*2^1 + 0*2^0 = (4+2+0)_{10} = 6_{10}$
7	111	$1*2^2 + 1*2^1 + 1*2^0 = (4+2+1)_{10} = 7_{10}$
8	1000	$1*2^3 + 0*2^2 + 0*2^1 + 0*2^0 = (8+0+0+0)_{10} = 8_{10}$
9	1001	$1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = (8+0+0+1)_{10} = 9_{10}$

Tab. 2.1. Conversão dos 10 algarismos decimais em binário

2.1.3 Sistema de Numeração Hexadecimal

O sistema de numeração hexadecimal é um sistema que possui a base 16 e dezasseis algarismos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Surge pela necessidade de abreviar ainda mais a representação de números, principalmente na área de programação. Nota-se que os dez primeiros algarismos coincidem com os do sistema decimal. O algarismo A representa a quantidade DEZ, o B a quantidade ONZE assim por diante até ao F que representa a quantidade QUINZE.

Para representar a quantidade dezasseis utilizaremos o conceito básico de formação de números mostrado na figura 2.1.

Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15

Tab. 5.2

2.1.3 Sistema de Numeração Hexadecimal

Para representar a quantidade dezasseis utilizaremos o conceito básico de formação de números mostrado na figura 2.1.

Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15

Veja que 11_{16} representa a quantidade $1x16^1 + 1x16^0 = 17_{10}$!

Exemplos:

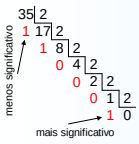
a) $121_{16} = 1*16^2 + 2*16^1 + 1*16^0$

b) $BD1,A1_{16} = B*16^2 + D*16^1 + 1*16^0 + A*16^{-1} + 1*16^{-2}$

2.1.4 Conversão Entre Sistemas

Conversão do decimal para binário

A forma prática de efectuar esta operação é, como exemplo, $35_{10} = X_2$



A operação consiste na divisão inteira e sucessiva do número decimal por 2 até obter o quociente 0. Depois colectamos os restos começando pelo último. Este será o dígito mais significativo no número binário.

logo, $35_{10} = 100011_2 = 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =$
$= (32 + 0 + 0 + 0 + 2 + 1)_{10} =$
$= 35_{10}$

2.1.4 Conversão Entre Sistemas

Conversão do decimal para binário

Vamos agora converter um número fraccionário:

$0{,}29_{10} = X_2$

Desta vez a operação consiste na multiplicação sucessiva do número decimal por 2. Cada novo multiplicando é a parte fraccionária do produto anterior

Depois colectamos as partes inteiras dos produtos começando pelo primeiro. Este será o dígito mais significativo no número binário fraccionário.

0,29	0,58	0,16	0,32	0,64	0,28
x 2	x 2	x 2	x 2	x 2	x 2
0,58	**1**,16	**0**,32	**0**,64	**1**,28	**0**,56

Ou seja: $0{,}29_{10} = 0{,}010010_2$

2.1.4 Conversão Entre Sistemas

Conversão do decimal para binário

A quantidade de produtos que vamos obter depende da precisão desejada

Para o nosso exemplo as centenas de milésimo nos é suficiente.

Logo:

$$0.29_{10} \cong 0.01001 = 0*2^{-1} + 1*2^{-2} + 0*2^{-3} + 0*2^{-4} + 1*2^{-5} = 0.25 + 0.03125 = 0.28125$$

Repare-se que aceitamos um erro de 3,01%!
É conveniente a escolha da precisão pois por vezes deparamos-nos com dízimos infinitos

2.1.4 Conversão Entre Sistemas

Conversão do binário e hexadecimal para decimal

A conversão de úmeros do sistema de numeração binária para o sistema de numeração decimal faz-se por aplicação directa da fórmula (2.1), ou seja

$$N_2 = d_k d_{k-1} d_{k-2} \ldots d_1 d_0 , d_{-1} d_{-2} \ldots d_{-p}$$
$$= (d_k.2^k + d_{k-1}.2^{k-1} + \ldots + d_1.2^1 + d_0 + d_{-1}.2^{-1} + d_{-2}.2^{-2} + \ldots + d_{-p}.2^{-p})_{10}$$

Exemplos:

a) $110{,}01_2 = (1.2^2 + 1.2^1 + 0.2^0 + 0.2^{-1} + 1.2^{-2})_{10} =$
$= 1x4 + 1x2 + 0x1 + 0x0.5 + 1x0{,}25 =$
$= 4+2+0+0+0{,}25 =$
$= 6{,}25_{10}$

b) $1B_{16} = 1x16^1 + Bx16^0 =$
$= (1x16 + 11x1)_{10} =$
$= (16 + 11)_{10} =$
$= 27_{10}$

2.1.4 Conversão Entre Sistemas

Conversão do hexadecimal para binário

Para percebermos o modo de conversão do sistema de numeração binária para o sistema de numeração hexadecimal vamos voltar à Tabela 2.2 acrescentando-lhe uma linha para o equivalente em binário

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

Tab. 2.3. equivalência entre o sistema de numeração decimal, hexadecimal e binário

Verificamos nesta tabela que para representarmos todos os coeficientes hexadecimal precisamos, em última instância, de 4 dígitos binários. Convencionemos que mesmo nos casos em que para representar o número hexadecimal sejam suficientes menos de 4 dígitos, como é de 0 a 7, tenhamos que usar 4 dígitos binários, para uniformizar.

2.1.4 Conversão Entre Sistemas

Conversão do hexadecimal para binário

Nesse caso a tabela 2.3 será ligeiramente modificada como mostra a Tabela 2.4 o que esgota tudo o que dizer da conversão do sistema de numeração hexadecimal para o binário. Portanto, a cada dígito hexadecimal faz-se corresponder um grupo de 4 dígitos binários.

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

Tab. 2.4. Conversão do sistema de numeração hexadecimal para binário

EXEMPLOS:

a) Converter para binário o número hexadecimal 23C

Ora, conforme a tabela 2.4, 2 = 0010, 3 = 0011 e C = 1100

Logo: $23C_{16}$ = 0010 0011 1100 = $001000111100 = 1000111100_2$

b) Converter para binário o número hexadecimal 23C,05

Do mesmo modo que na alínea anterior, $23C{,}05_{16} = 1000111100{,}00000101_2$

2.1.4 Conversão Entre Sistemas

Conversão do binário para hexadecimal

Para converter um número inteiro binário para hexadecimal agrupar os bits 4 a 4, a começar da direita para a esquerda. Se o último grupo não se completa, acresce Zeros à esquerda

Para converter um número fraccionário binário para hexadecimal agrupar os bits 4 a 4, a começar da esquerda para a direita. Se o último grupo não se completa, acresce Zeros à direita

Exemplos

a) $1110101_2 = 0111\_0101_2 = 0111\_0101 = 75_{16}$

b) $1110101{,}1100011_2 = 0111\_0101{,}1100\_0110_2 =$
$= 0111\_0101{,}1100\_0110 =$
$= 75{,}C6_{16}$

2.1.4 Conversão Entre Sistemas

Conversão do decimal para hexadecimal

A conversão do sistema de numeração decimal para o sistema de numeração hexadecimal processa-se de duas maneiras:

a) Conversão Directa e
b) Conversão com ponte em binário

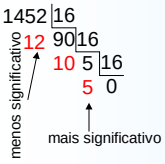
a) **CONVERSÃO DIRECTA**

a.1) Para converter a parte inteira do número decimal efectuamos divisões sucessivas por 16, até obter o quociente 0, com retenção dos restos. Depois colectamos estes restos tomando o último como sendo o dígito mais significativo do número hexadecimal

2.1.4 Conversão Entre Sistemas

Conversão do decimal para hexadecimal

Tomemos como o exemplo o número decimal 1452 a convertê-lo para hexadecimal:



Portanto:

$1452_{10}=(5)(10)(12)=5AC_{16}$

2.1.4 Conversão Entre Sistemas

Conversão do decimal para hexadecimal

a.2) Para converter a parte fraccionária do número decimal efectuamos multiplicações sucessivas por 16 com retenção das partes inteiras de cada produto. Depois colectamos estas partes inteiras começando pelo primeiro como sendo o dígito mais significativo.

Tomemos o número $0,1452_{10}$ a convertê-lo para hexadecimal:

Precisão	-1	-2	-3	-4	-5
Multiplicando	0,1452	0,3232	0,1712	0,7392	0,8272
Produto	2,3232	5,1712	2,7392	11,8272	13,2352
Parte inteira	2	5	2	11	13
	Mais significativo	→	→	→	Menos significativo

Ou seja:

$0,1452_{10}=0,(2)(5)(2)(11)(13)=0,252BD_{16}$

2.1.4 Conversão Entre Sistemas

Conversão do decimal para hexadecimal

b) **CONVERSÃO COM PONTE EM BINÁRIO**

A conversão com ponte em binário quer dizer que inicialmente se converte do sistema de numeração decimal para o binário e depois aplicam-se as regras de conversão deste sistema para o hexadecimal.

Exemplos:

a) $35_{10} = 100011_2 = 0010\_0011 = 23_{16}$

b) $0,29_{10} = 0,010010_2 = 0, 0100\_1000 = 0,48_{16}$

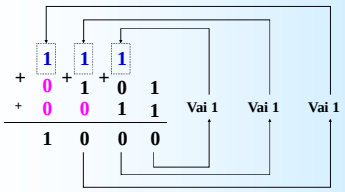
2.1.5 Adição em Binário

Para efectuar a adição dos números binários usam-se as mesmas regras convencionais usadas no sistema decimal. Porém temos que ter atenção para o facto de que no sistema binário existem apenas dois dígitos

A tabuada da adição em binário resume-se em:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 e vai 1
1 + 1 + 1 = 1 e vai 1

Vamos consolidar ideias somando os dois números binários 101 e 11.



2.1.5. Subtracção em Binário

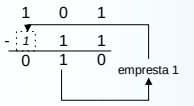
Para efectuarmos a subtracção dos números binários também usamos as regras convencionais do sistema decimal.
A tabuada da subtracção em binário resume-se em:

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 1 e empresta-se 1

Repare no empréstimo de 1 que se faz ao aditivo (ou diminuendo). Este é o mesmo 1 que se empresta no sistema decimal para coadjuvar o aditivo quando é menor que o subtractivo (ou diminuidor). Lembramos que em decimal (Conjunto dos Números Naturais!) 2-3 é igual a 9 com 1 emprestado a 2 para se tornar 12.

2.1.5. Subtracção em Binário

Vamos agora subtrair o binário 101 ao binário 11.



Este exemplo mostra a mecânica mais elementar para a operação de subtracção. Tivemos apenas um empréstimo de 1 que encontrou vazia a coluna correspondente no diminuendo.

Acompanhemos a seguir um caso mais complexo de subtracção de 111 em 1000 onde ocorrem muitos empréstimos de 1.

De cada vez que se empresta 1 para coadjuvar o dígito do diminuendo este 1 é somado ao digito imediatamente mais significativo do subtractivo. Só que esta soma origina um "vai 1" que deve ficar retido na mente para ser somado ao bit seguinte mais significativo do subtractivo. Esta soma por sua vez gerou um outro "vai 1". A cadeia continua até terminar a operação

2.1.5. Subtracção em Binário

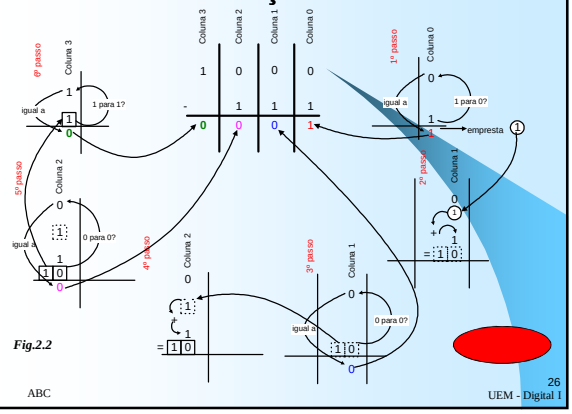


Fig.2.2

2.2. Códigos Binários Comuns

2.2.1 Códigos Binários

Um código é o uso sistemático e padronizado dum conjunto de símbolos para representar uma informação. Um exemplo típico e familiar para muitos é o código de estrada. A Fig. 2.3 mostra 3 símbolos que qualquer condutor reconhece logo a informação que pretendem fornecer. O primeiro símbolo indica que o condutor está aproximando-se a uma passagem de nível com guarda; o segundo indica que não se deve virar para a direcção indicada pela seta preta; o terceiro indica a obrigatoriedade de parar.

Fig.2.3

A compreensão dos símbolos parte de um "acordo" entre os utentes sobre a interpretação da informação dada.

Os próprios sistemas de numeração vistos são códigos. Com efeito o sistema de numeração decimal, na sua forma original, estabelece que cada símbolo contém uma quantidade de ângulos internos exactamente igual à quantidade numérica que representa (Vide tabela 2.5)

2.2.1 Códigos Binários

Quantidade representada	Símbolo	Quantidade de ângulos internos
	○	○
•	1	1
••	2	2
•••	3	3
••••	4	4
•••••	5	5
••••••	6	6
•••••••	7	7
••••••••	8	8
•••••••••	9	9

Tab. 2.5 significado do números decimais

Códigos binários são os que empregam combinações dos dois dígitos binários 0 e 1 para codificar várias regras. Dado *n* bits pode-se obter 2^n combinações diferentes. A cada combinação se faz corresponder uma determinada quantidade.

Tomando parte dessas 2^n combinações com características específicas definem-se códigos diferentes. O sistema de numeração binária designa-se por código binário natural em virtude de utilizar todas as 2^n combinações na sequência natural.

2.2.1 Códigos Binários

Código Gray

Um dos códigos binário mais conhecido é o Código Gray. A sua principal característica é que varia apenas um bit na passagem dum número para o outro. A tabela 2.6 mostra a formação do código Gray de 2, 3 e 4 bits.

Decimal	Binário natural	Código Gray		
		2 bits	3 bits	4 bits
0	0	00	000	0000
1	1	01	001	0001
2	10	11	011	0011
3	11	10	010	0010
4	100		110	0110
5	101		111	0111
6	110		101	0101
7	111		100	0100
8	1000			1100
9	1001			1101
10	1010			1111
11	1011			1110
12	1100			1010
13	1101			1011
14	1110			1001
15	1111			1000

Tab. 2.6

2.2.1 Códigos Binários

Código Johonson

O segundo código cíclico mais conhecido é o Código Johnson apresentado na Tab. 2.7. A capacidade de codificação código Johnson de n bits é de 2^n quantidades diferentes. Quer isto dizer que para representar os dez dígitos decimais pelo código Johnson, são necessários 5 bits. Não é um código muito usado por consumir muitos recursos para representar uma quantidade bem menor que 2^n

Decimal	Código Johnson
0	00000
1	00001
2	00011
3	00111
4	01111
5	11111
6	11110
7	11100
8	11000
9	10000

Tab. 2.7

2.2.1 Códigos Binários

Código 2-entre-5

O terceiro código é o 2-entre-5. Tem 5 bits dos quais 2 são iguais a 1, conforme mostra a Tab. 2.8 ao lado. Codifica os 10 dígitos decimais

Decimal	Código 2 entre 5
0	00011
1	00101
2	01001
3	10001
4	00110
5	01010
6	10010
7	01100
8	10100
9	11000

Tab. 2.8

2.2.1 Códigos Binários

Código Excesso-3

O quarto código é o Excesso-3(Tab. 2.9). Codifica os dígitos decimais com a particularidade de adicioná-lo a 3 antes de o converter em binário

Decimal	Binário natural	Código Excesso 3
0	0	11
1	1	100
2	10	101
3	11	110
4	100	111
5	101	1000
6	110	1001
7	111	1010
8	1000	1011
9	1001	1100

Tab. 2.9

2.2.2 Códigos Decimais Codificados em Binário

Os códigos decimais codificados em binário, em inglês BCD, isto é, Binary Coded Decimal, são muito usados nas aplicações digitais. Codificam os dígitos decimais para binário, fazendo uso de 4 bits com determinados pesos no número. Por vezes se estendem para codificarem igualmente os dígitos hexadecimais.
Os códigos BCD usam a notação BCD*dcba* em que BCD indica que o código é BCD e *dcba* é uma sequência de números que indicam que o código comporta 4 bits com pesos d, c, b e a

2.2.2 Códigos Decimais Codificados em Binário

Código BCD8421

O código BCD8421 é certamente o código BCD mais usado. Os termos 8421 indicam que o bit *d* tem peso $8(=2^3)$, o bit *c* tem peso $4(=2^2)$, o bit *b* tem peso $2(=2^1)$ e, finalmente, o bit *a* tem peso $1(=2^0)$.

Decimal	BCD8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Na verdade o código BCD8421 segue a sequência do binário natural, excepto o facto de que é obrigatório reservar 4 bit e logo, a maior quantidade codificável ser

$15_{10} = F_{16} = 1111_2(=1x8+1x4+1x2+1x1)$.

Tab. 2.10

2.2.2 Códigos Decimais Codificados em Binário

Código BCD2421

Decimal	BCD2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

O código BCD2421, também conhecido por Código Aiken, é outro código muito usado nas aplicações digitais. Os digitos têm os pesos: *d* tem peso $2(=2^1)$, o bit *c* tem peso $4(=2^2)$, o bit *b* tem peso $2(=2^1)$ e, finalmente, o bit *a* tem peso $1(=2^0)$.

Tab. 2.11

2.2.3 Códigos Alfanuméricos

Os códigos vistos até aqui permitem representar números. Entretanto na maior parte dos sistemas digitais, tanto de controle como de processamento de dados, existe a necessidade de representar letras e outros caracteres especiais, como se pode ver na Tab. 2.12. Aqui surge a necessidade de criar os chamados códigos Alfanuméricos, que representam mais do que números. De entre os diversos códigos Alfanuméricos foi definido como código internacional o ASCII (American Standard Code for Information Interchage).

Para poder representar as 26 letras e os 10 dígitos decimais são necessárias 36 combinações diferentes. Sabendo que com ***n*** bits se obtém 2^n combinações diferentes, é fácil deduzir que para conseguir as 36 combinações são necessários 6 bits com os quais até se consegue 64 combinações.

2.2.3 Códigos Alfanuméricos

$C_3C_2C_1C_0$	$C_6C_5C_4$ 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	`	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	S0	RS	.	>	N	^	n	~
1111	S1	US	/	?	O	_	o	DEL

Tab. 2.12

2.2.4 Códigos Detectores de Erros

Existem diversos tipos de códigos detectores de erros de entre os quais se tem destacado os *códigos de paridade* e os de *peso constante*.
Peso num número binário é a quantidade de bits 1 que ele possui.

CÓDIGOS DE PARIDADE
Os códigos de paridade acrescentam um bit igual a 1 chamado bit de paridade. Este bit é acrescido por forma que em todas as palavras do código exista uma quantidade par de 1’s – paridade par, ou ímpar de 1’s – paridade ímpar. Ao mesmo tempo este bit vai indicar que na palavra em causa se espera encontrar uma quantidade ímpar de 1’s (na paridade par) ou uma quantidade par de 1’s(na paridade ímpar). Assumindo que este bit não seja corrompido e ocorra um erro.

2.2.4 Códigos Detectores de Erros

O código de paridade será eficiente apenas para detectar a ocorrência de 1 erro. A ocorrência duma quantidade ímpar de erros será detectada como 1 erro. Se ocorrer uma quantidade par de erros não será detectado

Decimal	Código Excesso 3	Paridade par	Paridade ímpar
0	0011	0	1
1	0100	1	0
2	0101	0	1
3	0110	0	1
4	0111	1	0
5	1000	1	0
6	1001	0	1
7	1010	0	1
8	1011	1	0
9	1100	0	1

Tab. 2.13. Códigos de paridade

Estes códigos não permitem correcção dos erros. Quando se detecta erro pede-se a retransmissão

2.2.4 Códigos Detectores de Erros

CÓDIGOS DE PESO CONSTANTE
O código 2-entre-5 é o mais usado nesta categoria. Consiste de 5 bits para representar os 10 algarismos decimais. Desses 5 bits apenas 2 são iguais a 1(Tab. 2.14). Portanto o peso de qualquer palavras do código é 2.

Decimal	Código 2 entre 5
0	00011
1	00101
2	01001
3	10001
4	00110
5	01010
6	10010
7	01100
8	10100
9	11000

A distância mínima neste código é 2 podendo apenas detectar 1 erro.

Tab. 2.14. Código 2-entre-5

2.3. Números Com Sinal

- **As representações vistas até agora envolvem apenas números inteiros positivos. No entanto na vida real temos números com sinal negativo e outros fraccionados.**
- **A representação dos números com sinal faz-se de diversas maneiras:**

1. Uso dum bit mais significativos para representar o sinal.

2. Uso do complemento 2 do número.

3. Uso do complemento 1 do número

Vamos tratar em detalhe cada um deles

2.3.1. Sinalização pelo bit mais significativo

Em termos teóricos o método mais simples de representar números com sinal é o de sinalização pelo bMS.

Consiste em definir um comprimento fixo para o modulo do número e acrescentar um bMS para o sinal que vale 0 para número positivo e 1 para número negativo

Comparado com os outros métodos, este peca, na prática, por requerer circuitos aritméticos e algoritmos auxiliares para computação dos números com sinal, o que se reflecte no custo.

Número Decimal	Número Binário
+7	0 111
+6	0 110
+5	0 101
+4	0 100
+3	0 011
+2	0 010
+1	0 001
0	0 000
-1	1 001
-2	1 010
-3	1 011
-4	1 100
-5	1 101
-6	1 110
-7	1 111

Tab. 2.15. Representação dos números com sinal no sistema de uso do BMS

2.3.2. Sinalização Pelo Complemento 2 ou 1

Neste método os números positivos são representados como no anterior enquanto que os negativos são representados pelo complemento 2 ou 1 do número positivo (Tab. 2.16)

O complemento 2 dum número binário N de *n* bits é dado por

$$[N]_2 = 2^n - N \qquad (2.4)$$

EXEMPLO.
Determinar o complemento 2 do número 01100101.

Sol: $[01100101]_2 = 2^8 - 01100101 =$
$= 100000000 - 01100101 =$
$= 10011011_2$

2.3.2. Sinalização Pelo Complemento 2 ou 1

O complemento 1 dum número binário N de *n* bits é dado inversão de todos os bit ou pela fórmula:

$$[N]_1 = 2^n - N - 1 \qquad (2.5)$$

EXEMPLO.
Determinar o complemento 1 do número 01100101.

Sol: $[01100101]_2 = 2^8 - 01100101 - 1 =$
$= 100000000 - 01100101 - 1 =$
$= 10011010_2$

2.3.2. Sinalização Pelo Complemento 2 ou 1

Número Decimal Com sinal	Número Binário	
	Em complemento 1	Em complemento 2
+7	0 111	0 111
+6	0 110	0 110
+5	0 101	0 101
+4	0 100	0 100
+3	0 011	0 011
+2	0 010	0 010
+1	0 001	0 001
0	0 000	0 000
-1	1 110	1 111
-2	1 101	1 110
-3	1 100	1 101
-4	1 011	1 100
-5	1 010	1 011
-6	1 001	1 010
-7	1 000	1 001

Tab. 2.16. Representação dos números com sinal no sistema de complemento

2.3.3. Operações aritmética com Complemento

EXEMPLO 1. REALIZAR 5-10

1º . 5 = 101

2º . 10 = 1010

3º . +10 = 01010

4º . Como +10 ocupa mais bits então o 5 deve seguir o mesmo padrão: +5 = 00101

5º . -10 =compl(01010) = 10110

6º . 5-10 = 5+(-10) = 00101+10110

Ou seja:
0 0 1 0 1
+ 1 0 1 1 0
1 1 0 1 1

7º . Os nossos numeros são de 4 bits. o 5º bit revela o sinal. O nosso resultado revela que o numero é negativo. Então qual é esse numero?

8º . Calculamos o seu complemento:
Compl(11011)=
1 0 0 0 0
- 1 1 0 1 1
0 0 0 1 0 1

9º . O 0 em azul está fora do nosso sistema

2.3.3. Operações aritmética com Complemento

EXEMPLO 2. REALIZAR 10-5

1º . 5 = 101

2º . 10 = 1010

3º . +10 = 01010

4º . Como +10 ocupa mais bits então o 5 deve seguir o mesmo padrão: +5 = 00101

5º . -5 =compl(00101) = 11011

6º . 10-5 = 10+(-5) = 01010+11011

Ou seja:
0 1 0 1 0
+ 1 1 0 1 1
1 0 0 1 0 1

7º . O 1 em azul está fora do nosso sistema, então é descartável

8º . Os nossos numeros são de 4 bits. O 5º bit revela o sinal. O nosso resultado revela que o numero é positivo. Então não precisa de ser complementado!