Curso: Engenharia de Computação

Sistemas Digitais

Prof. Clayton J A Silva, MSc
clayton.silva@professores.ibmec.edu.br

ibmec

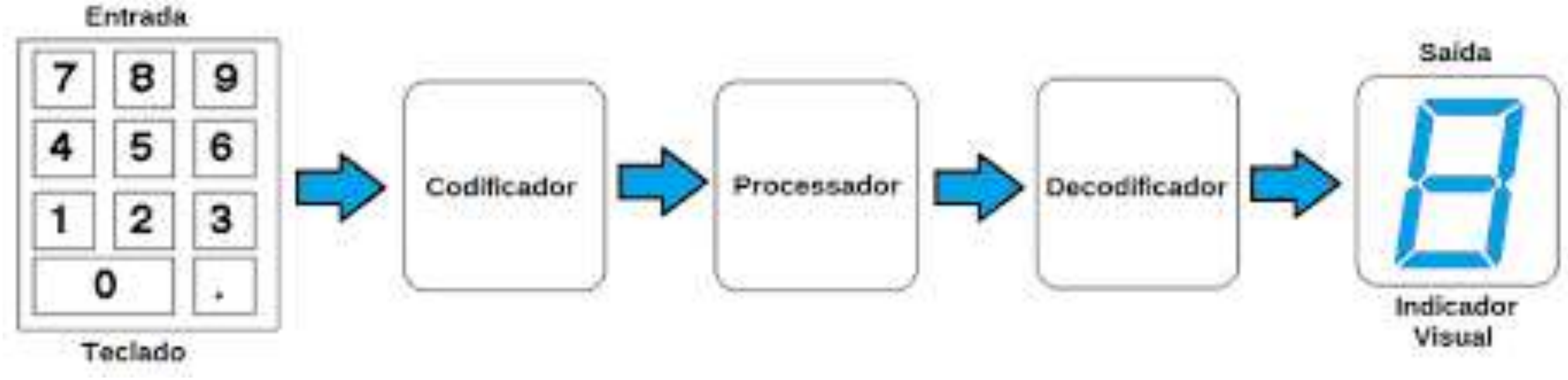
Codificação binária

- Um **conjunto finito** de elementos de **qualquer natureza** pode ser relacionado a um conjunto de **palavras binárias distintas**, cada uma das quais representa um elemento do conjunto

- O conjunto das palavras binárias que se relaciona com o conjunto discreto é chamado de **código**

- Todo código binário deve possuir ***m*** **bits,** de forma que suas 2^m combinações possíveis sejam suficientes para representar todos os elementos discretos do conjunto finito com o qual ser relacione

- Os sistemas digitais podem operar com elementos de vários tipos processando as palavras do código correspondentes



Codificação binária

Codificação binária

- De um modo geral os sistemas digitais utilizam códigos que propiciam representar

   - **caracteres alfanuméricos:** caracteres de **alfabeto** e dígitos **numéricos**

   - **símbolos especiais**

Representação numérica

Codificação binária

- O número codificado pelo seu binário equivalente utiliza uma **codificação em binário puro**

- Considerando a representação em binário puro, os sistemas digitais tratam o número de interesse, geralmente expresso em decimal, pela correspondente **conversão em binário**

- O limite da capacidade de representação numérica está limitado aos *m* bits utilizados pelo código

base 10
x
base 2

Base 2	Base 10
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
10000	16
10001	17

Representação de grandezas numéricas

- Notações **posicional e polinomial**

+significativo

$$1457= 1x10^3+4x10^2+5x10^1+7x10^0$$

- Dígitos do conjunto *D={0,1,2,3,...,9}*, **base decimal**

Código BCD

- BCD – ***Binary Coded Decimal***

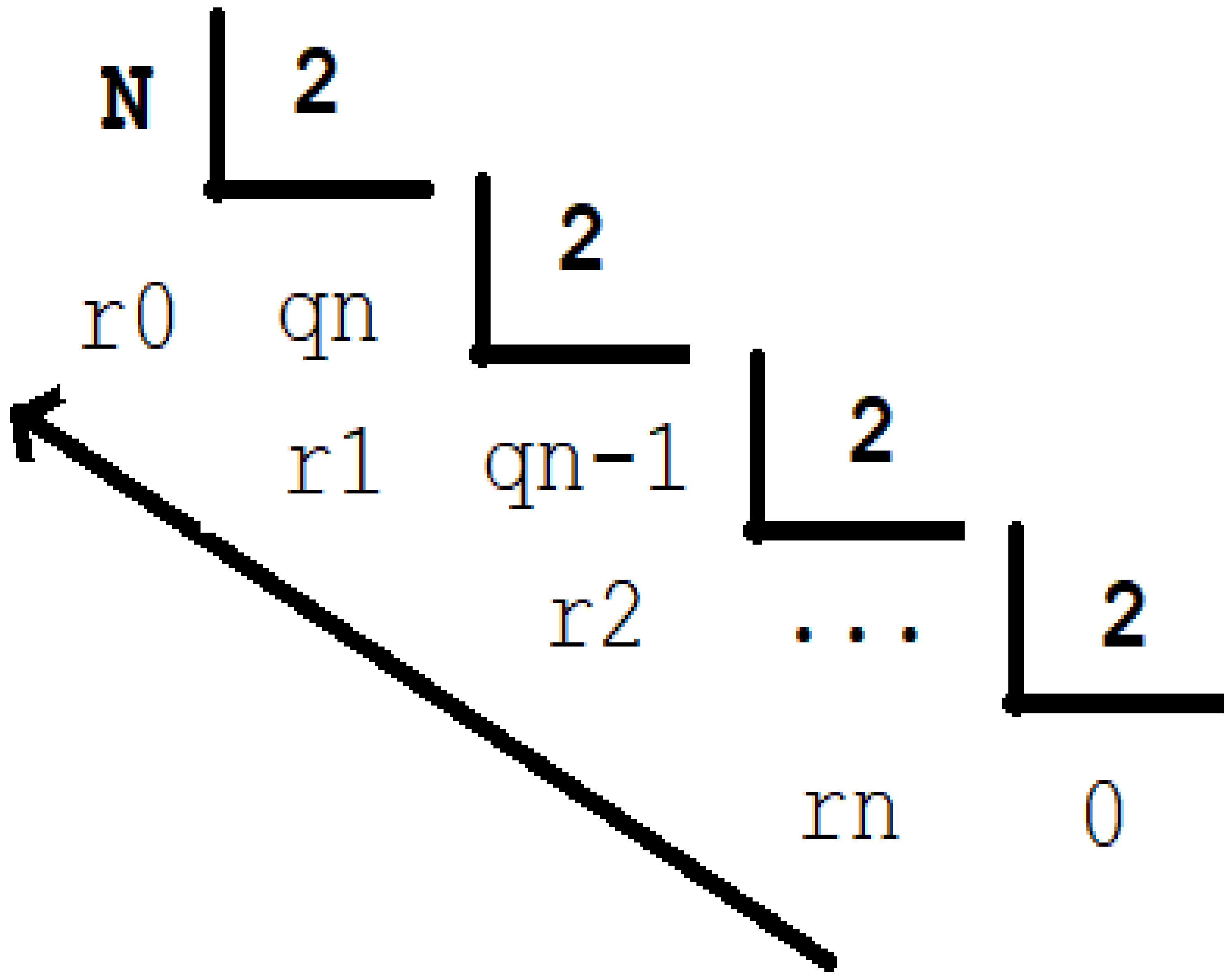
- Cada dígito decimal será representado pelo seu equivalente binário, ou seja, para um número decimal são necessários 4 bits para **cada** dígito decimal

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

Conversão de números: bases decimal e binária

Conversão de bases

- Base 10 para base 2: **divisões sucessivas**



$$\boxed{N = (r_n\ r_{n-1}\ r_{n-2}\ \ldots r_0)_2}$$

Conversão de bases

- Base 2 para base 10: **notação polinomial**

$$b_n \times 2^n + b_{n-1} \times 2^{n-1} \ldots + b_0 \times 2^0$$

Representação de números negativos

- Representação em bit sinal (sinal e magnitude)
- Representação em complemento de 1
- Representação em complemento de 2

Representação em ponto flutuante

- Notação científica: $N = f \times 10^{e}$

, onde f – fração ou mantissa; e – expoente

- Pela representação em ponto flutuante – equivalente computacional, quando se **convenciona o número de dígitos** para representar mantissa e expoente:

  - a faixa de representação é determinada pelo número de dígitos do expoente e

  - a precisão é determinada pelo número de dígitos da mantissa.

Representação em ponto flutuante

- A versão de ponto flutuante nos sistemas digitais requer a representação da mantissa e do expoente no sistema binário.

Aritmética binária

Aritmética binária

- **Adição** de dois dígitos binários: b_1+b_0

		b0		
		0	1	
b1	0	0	1	
	1	1	10	*carry ou vai um*

Aritmética binária

- Adição binária de dois números de *m* bits

1. Realizar a operação bit a bit do menos ao mais significativo (da direita para a esquerda)
2. Aplica-se a tabela anterior
3. Se houver bit 1 de *carry* transporta-se para a soma dos bits seguintes mais significativos (à esquerda)
4. Repete-se o processo até alcançar o bit mais significativo.

Aritmética binária

- **Subtração** de dois dígitos binários: b_1-b_0

		b0		
		0	1	
b1	0	0	11	*carry ou vai menos um*
	1	1	0	

Aritmética binária

- Subtração binária de dois números de *m* bits – Minuendo>Subtraendo

1. Operação bit a bit, do menos ao mais significativo
2. Aplica-se a tabela anterior
3. Se houver bit -1 de *carry transporta-se* para a subtração dos dígitos seguintes (à esquerda – mais significativos),
4. Repete-se o processo até alcançar o bit mais significativo.
5. Se o minuendo for menor do que o subtraendo inverter a operação e representar o número negativo

Observações

- As máquinas possuem **palavras** com tamanho definido de *m* bits
- Se a operação resultante ultrapassar a capacidade do sistema representar o número obtido...
- caracteriza-se ***overflow = 'estouro'***

Códigos de caracteres

Código ASCII

- Cada caractere ASCII possui 7 bits
- As palavras de 0x0 a 0x1F representam **caracteres de controle** e **não são impressos**

Código ASCII

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Código UNICODE

- Cada caractere ou símbolo UNICODE utiliza 16 bits, intitulado **ponto de código**
- Gerenciado por um consórcio de empresas, os pontos de códigos são alocados evolutivamente
- O espaço é dividido em múltiplos de 16 pontos de código, que podem ser alocados a diferentes alfabetos no mundo
- Ver https://pt.wikipedia.org/wiki/Unicode

Detecção e correção de erros

Erros

- As palavras dos códigos **armazenadas** em memória podem ser alteradas por problemas elétricos na **leitura** ou **escrita** dos dados, assim como podem ser alteradas na **transmissão** através de **canais** de comunicações => erro

- Alguns sistemas digitais utilizam códigos para **detecção** de ou a **correção** de erros

Códigos com correção ou detecção de erros

Palavra de código

n bits = m + r bits

m bits	r bits
dados	redundância ou verificação

- Considerando o formato, ***2^n combinações*** *possíveis*, no entanto somente ***2^m são válidas***

- Os *r* bits de redundância ou verificação devem ser capazes de permitir a identificação ou correção de erros com a aplicação de um algoritmo do sistema

Distância de *Hamming (h)*

- Número de posições de bits em que duas palavras de um código são diferentes

- P. ex., em um código de 8 bits, a distância entre as palavras

  1011 0111 e

  1111 0111

  é ***h=1***

Distância de *Hamming (h)*

- Seja o código

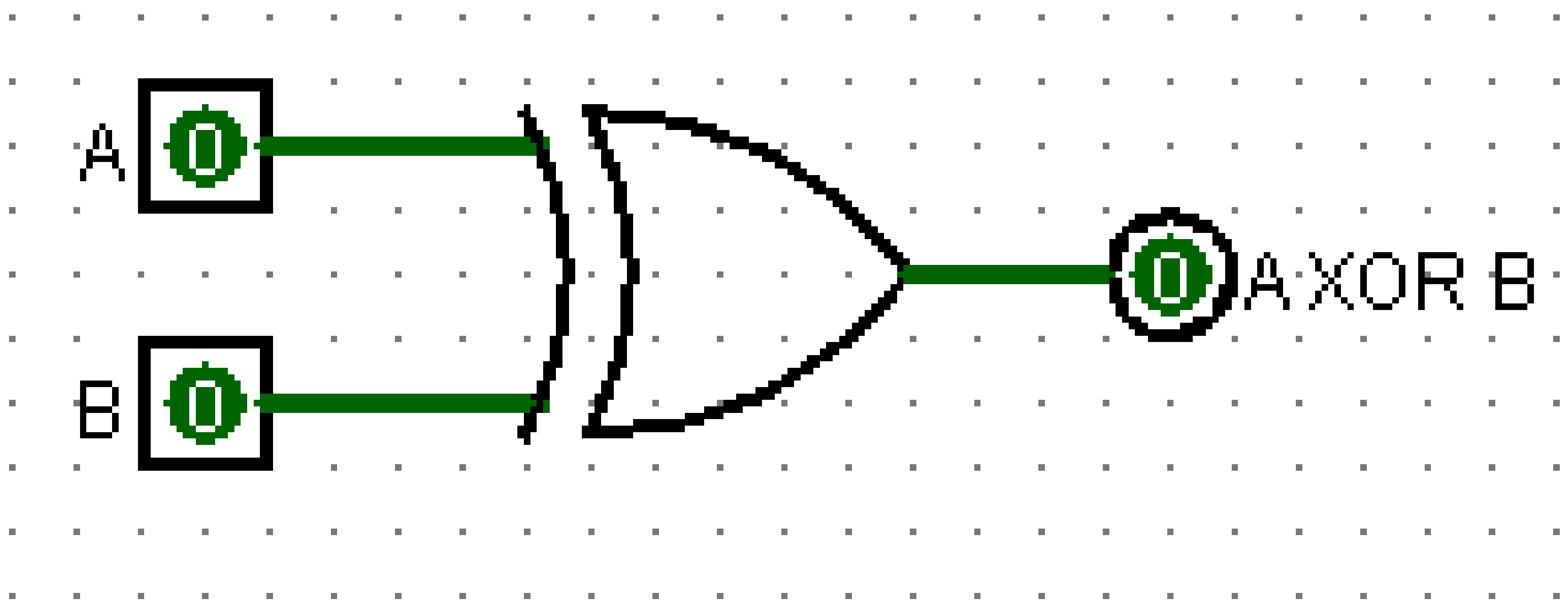
Palavra	Elemento
000	A
110	B
101	C
011	D

- A distância de *Hamming* é $h=2$

Distância de *Hamming (h)*

- Calcula-se logicamente pela aplicação da operação lógica OU EXCLUSIVO (*XOR*, $\oplus$, $\underline{\vee}$)

A	B	A *XOR* B
0	0	0
0	1	1
1	0	1
1	1	0



Códigos com correção ou detecção de erros

- As propriedades de correção e detecção de erros de um código dependem de
  - para *detectar* ***d*** **erros** é necessário um código com uma distância de Hamming de ***h=d+1***
  - para *corrigir* ***d*** **erros** é necessário um código com uma distância de Hamming de ***h=2d+1***

Código com bit de paridade

- Código para detecção de 1 erro
- Inserir 1 ou 0 para assegurar um número par (ou ímpar) de 1s
- *h=2*, permite **detectar** 1 erro

Dados de 3 bits	1 bit de paridade	Elemento
000	0	A
001	1	B
010	1	C
011	0	D
100	1	E
101	0	F
110	0	G
111	0	H

Código para correção de 1 erro

- Pode-se demonstrar que r deve ser tal que $(m + r + 1) \leq 2^r$

Tamanho da palavra	bits de verificação
8 bits	4 bits
16 bits	5 bits
32 bits	6 bits
64 bits	7 bits
128 bits	8 bits

- Algoritmo de *Hamming*

IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec