Sistemas Digitais
Representação Digital de Informação

João Paulo Baptista de Carvalho

joao.carvalho@inesc-id.pt

TÉCNICO LISBOA

inesc id lisboa

Representação de números em Base b

- Base 10 (decimal):
  - $435_{10} = 4 \times 100 + 3 \times 10 + 5 \times 1$

    $= 4 \times 10^2 + 3 \times 10^1 + 5 \times 10^0$
  - O número resulta da soma de sucessivas potências de 10, cada uma pesada pelo valor do algarismo correspondente

Representação de números em Base b (II)

- O mesmo pode ser feito para qualquer base. Por exemplo:
- Base 7:
  - $1161_7 = 1 \times 7^3 + 1 \times 7^2 + 6 \times 7 + 1$

    $= 1 \times 343 + 1 \times 49 + 6 \times 7 + 1$

    $= 435_{10}$
  - 1161_7 é uma forma alternativa de representar 435_{10}

Representação de números em Base b (III)

- Pode representar-se qualquer número inteiro *N* em qualquer base *b*:

$$N = p_{n-1} \times b^{n-1} + p_{n-2} \times b^{n-2} + \ldots + p_1 \times b^1 + p_0 \times b^0$$

$$N = \sum_{j=0}^{n-1} p_j \times b^j$$

  - Para a representação de números numa base *b*, não podem ser utilizados algarismos *p* de valor igual ou superior a *b*

Conversão entre números de base *b* para base 10

- Já se mostrou como se faz:
  - $1161_7 = 1 \times 7^3 + 1 \times 7^2 + 6 \times 7 + 1$
    $= 1 \times 343 + 1 \times 49 + 6 \times 7 + 1$
    $= 435_{10}$
  - $210_5 = 2 \times 5^2 + 1 \times 5 + 0 = 55_{10}$

- E como fazer a conversão inversa?

Conversão entre números de base 10 para base *b*

- Método das divisões sucessivas

$$N = p_{n-1} \times b^{n-1} + p_{n-2} \times b^{n-2} + \ldots + p_1 \times b^1 + p_0 \times b^0$$

$$\frac{N}{b} = \left(p_{n-1} \times b^{n-2} + p_{n-2} \times b^{n-3} + \ldots + p_1 \times b^0 \right) + \frac{p_0 \times b^0}{b}$$

$$= \left(p_{n-1} \times b^{n-2} + p_{n-2} \times b^{n-3} + \ldots + p_1 \times b^0 \right) + \frac{p_0}{b}$$

  - O resto da divisão de *N* por *b*, é p_0
  - O resto da divisão do resultado anterior por *b*, é p_1
  - ...e assim sucessivamente até se obter p_{n-1}

Conversão entre números de base 10 para base *b* (II)

- Método das divisões sucessivas (exemplo):
  - $273_{10} = ?_5$
    - $p_0 = 3$
    - $p_1 = 4$
    - $p_2 = 0$
    - $p_3 = 2$
  - $273_{10} = 2043_5$

$$\frac{273}{5} = 54 + \frac{3}{5}$$

$$\frac{54}{5} = 10 + \frac{4}{5}$$

$$\frac{10}{5} = 2 + \frac{0}{5}$$

$$\frac{2}{5} = 0 + \frac{2}{5}$$

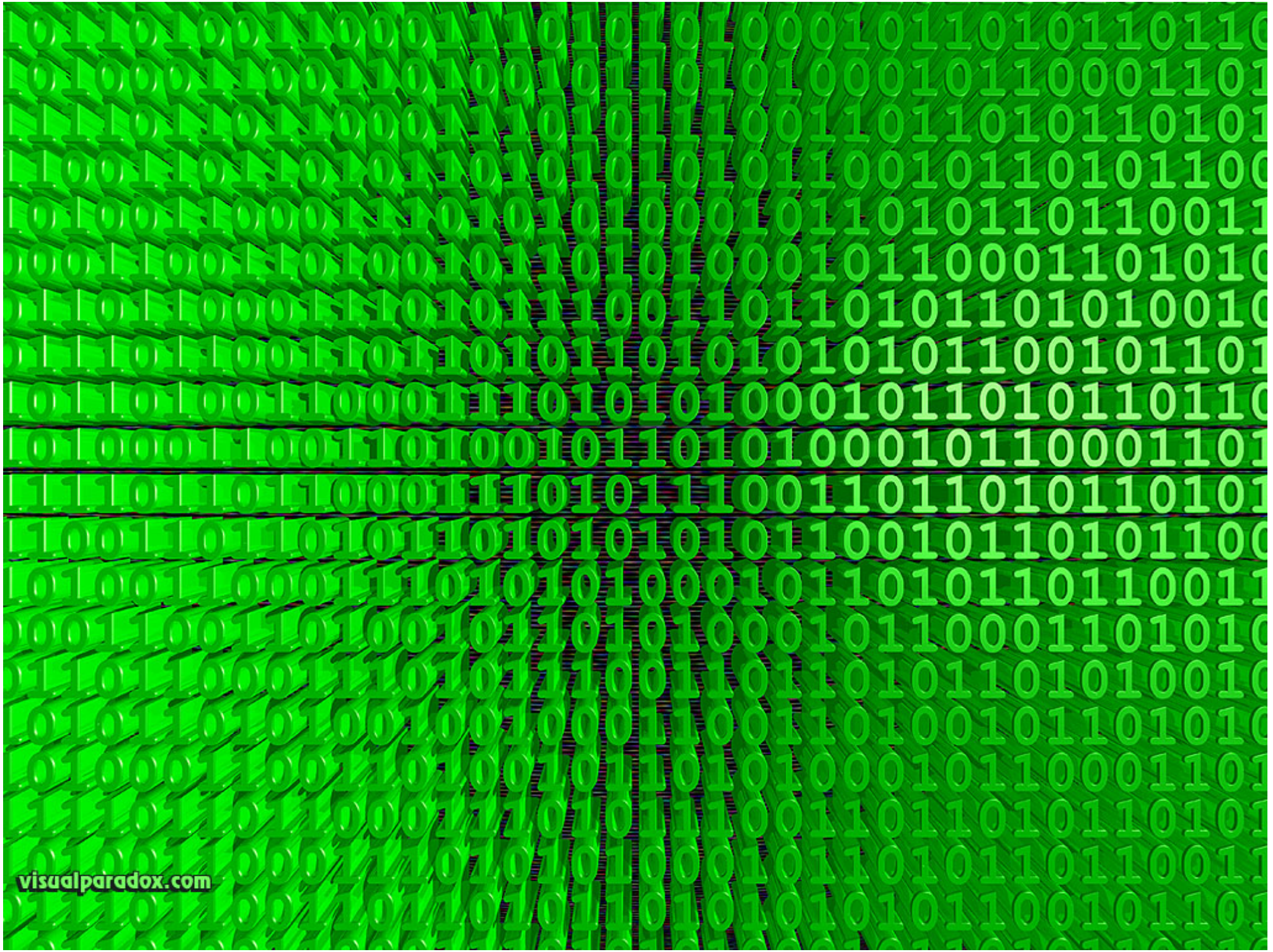
Conversão entre números de base 10 para base 2

- Método das divisões sucessivas (exemplo):
  - $20_{10} = ?_2$
    - $p_0 = 0$
    - $p_1 = 0$
    - $p_2 = 1$
    - $p_3 = 0$
    - $p_4 = 1$
  - $20_{10} = 10100_2$

```
20 |2
0⓪ 10 |2
   ⓪  5 |2
   ①  2 |2
   ⓪  1 |2
   ①  0
```

Base 2 – Números binários

- Em computadores e outros sistemas digitais, a representação de números está limitada a 2 valores diferentes (normalmente representados pela diferença de tensão eléctrica entre 2 pontos de um circuito electrónico)
  - 0, Low, L, 0V
  - 1, High, H, +5V
- Assim, num sistema digital um número inteiro é representado por uma sequência de algarismos binários, ou *bits* (*Binary digIT*)
  - 110101
  - $110101_2 = 1x2^5+1x2^4+0x2^3+1x2^2+0x2^1+1x2^0 = 32+16+4+1 = 53_{10}$
  - 01011101011010111010111010111010111010111101010


visualparadox.com

Base 2 – Números binários (II)

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

- Vamos aprender a contar? ;-)

Números fraccionários em base 2

$$N = p_{-1} \times b^{-1} + p_{-2} \times b^{-2} + ... + p_{-n} \times b^{-n}$$

- Exemplo:
  - $0.1011010_2 = 1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-6}$
    $= 0.5 + 0.125 + 0.0625 + 0.015625$
    $= 0.703125_{10}$
- E como fazer a conversão inversa?
  - Multiplicações sucessivas por 2 e aproveitar a parte inteira do resultado
    - $0.62710 = 0.p_{-1}p_{-2}p_{-3}p_{-n}$
    - $0.62710 \times 2 = 1.254$; $p_{-1} = 1$
    - $0.254 \times 2 = 0.508$; $p_{-2} = 0$
    - ...
  - $0.62710_{10} = 0.101000001_2$

Na realidade:
$0.1011010_2 = 0.70_{10}$
Dado que não se deve acrescentar precisão ao número original, $10^p \leq 2^7$, $p \leq [\log_{10} 2^7]$, $p=2$

Números em bases potências de 2

- É difícil para um humano trabalhar em binário dadas as sequências bastante longas de 0 e 1
  - Ex: $153{,}845_{10} = 10011001{,}110110001_2$
- As bases que são potências de 2 permitem uma representação condensada da base 2 com as quais a conversão é muito rápida
- As bases 8 (octal) e principalmente a base 16 (hexadecimal) são as mais comuns

Decimal	Octal /Hexadecimal	Decimal	Hexadecimal
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

Conversões para Hexadecimal

- $4A6F_{16} = 4 \times 16^3 + 10 \times 16^2 + 6 \times 16 + 15$
  $= 4 \times 4096 + 10 \times 256 + 6 \times 16 + 15$
  $= 19055_{10}$
- $4A6F_{16} = ?_2$ $(16 = 2^4)$
  - $0100\ 1010\ 0110\ 1111_2$
    4 A 6 F
- $347_8 = ?_2$ $(8 = 2^3)$
  - $011\ 100\ 111_2$
    3 4 7

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

Bits, bytes, ks,...

- ‘0’, ‘1’ são bits
- A um conjunto de 8 bits chama-se byte
- A um conjunto de 4 bits chama-se nibble (cada algarismo BCD é um nibble)
- À unidade mínima processada por um sistema digital chama-se palavra (word) – por exemplo, o processador Intel 8080 tem palavras de 8 bits, o processador Motorola 68000 processa palavras de 16 bits, o Pentium palavras de 32 bits, o Core i7 palavras de 64 bits, etc.
- Os sistemas digitais necessitam de grandes quantidades de bits para representar informação, sendo por isso comum utilizar múltiplos como kbyte=2^{10}=1024bytes, Mbyte=2^{20}bytes (Mega), Gbyte=2^{30}bytes (Giga), Tbyte=2^{40} (Tera), etc...

Códigos

- Até agora viu-se como é possível representar números inteiros e fraccionários utilizando simplesmente 0's e 1's (bits)
- No entanto, para além dos números, existem muitos mais tipos de informação que necessitam ser tratados num sistema digital
- O caso mais óbvio é a representação de texto...
- Para representação dos mais diversos tipos de informação, convém ter a noção de "código"

Codificação

- A solução para a representação de um determinado tipo de informação é a sua codificação utilizando os bits que forem necessários
- Exemplo: Pretende-se representar o andar em que se encontra um elevador num prédio de 6 pisos.
  - São necessários pelo menos 3 bits

Comprimento do código

Andar
2ª Cave
1ª Cave
R/C
1º Andar
2º Andar
3º Andar

Palavra do código

Codificação

- Para que um código seja válido apenas é necessário que não existam configurações repetidas
- De resto, a escolha das codificações é livre e pode depender dos mais diversos critérios

Andar	Codificação1	Codificação 2	Codificação 3
2ª Cave	000	110	0011
1ª Cave	001	111	0101
R/C	010	000	1001
1º Andar	011	001	0110
2º Andar	100	010	1010
3º Andar	101	011	1100

Códigos Numéricos

- O código numérico mais simples consiste na representação de um número inteiro por uma palavra de código que é a sua representação em binário. Se cada palavra tem um comprimento constante, o código denomina-se Código Binário Natural (CBN)

Número	CBN de 4 bits	Número	CBN de 4 bits
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Códigos Numéricos - BCD

- O código BCD – Binary Coded Decimal, permite representar cada dígito de um número decimal pelo seu equivalente binário:

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

- Exemplo:

$27_{(10)} = \underbrace{0010}_{2}\ \underbrace{0111}_{7}{}_{(BCD)}$

De notar que não existe correspondência entre BCD e base 2:

$27_{(10)} = 11011_{(2)}$

$0010\ 0111_{(BCD)} = 27_{(10)}$

$0010\ 0111_{(2)} = 39_{(10)}$

Códigos Reflectidos

- Código Binário Reflectido ou código de Gray: números sucessivos são codificados por palavras que diferem apenas um bit entre si

Número	CBR de 3 bits
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

Códigos Alfanuméricos

- Representação de texto.Ex: Código ASCII (*American Standard Code for Information Exchange*) – 7 bits

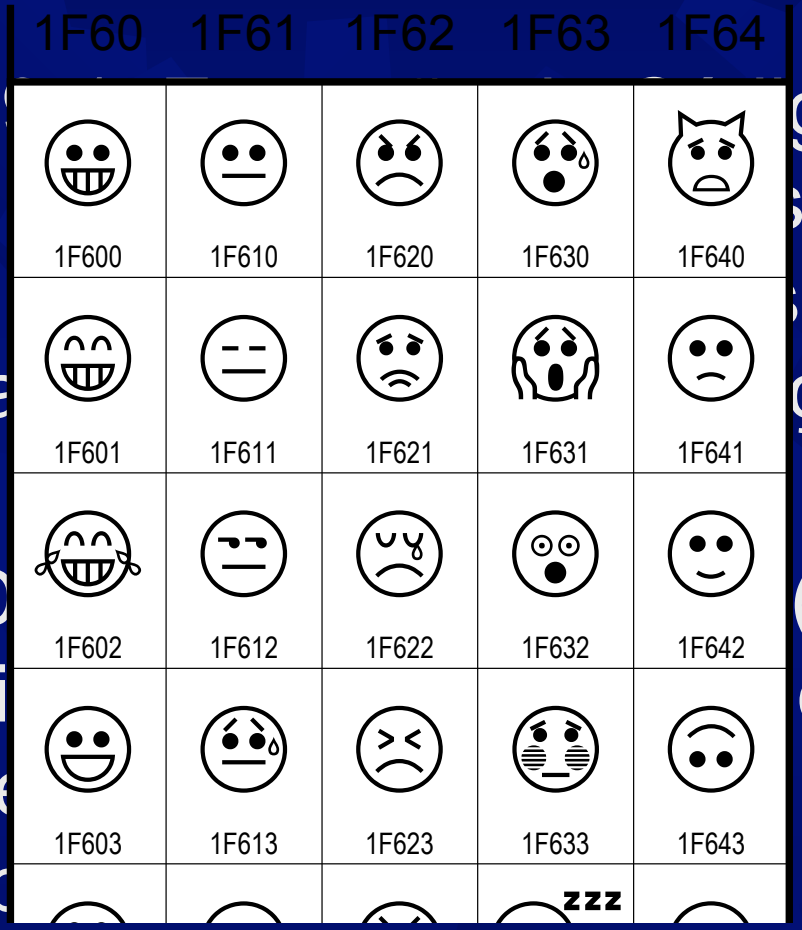
	$b_6b_5b_4$							
$b_3b_2b_1b_0$	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
0000	NUL	DLE	SP	0	@	P	`	p
0001	STH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	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	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Extensões ao Código ASCII

- ISO-8859-1: Extensão do Código ASCII a 8 bits que permite representar os caracteres próprios existentes nas línguas da Europa Ocidental, como por exemplo “ç”, “ñ”, “å”, “ø”, “€”, etc.
- UNICODE: Extensão a 16 bits (actualmente até 32 bits) que pretende codificar todos os caracteres de todas as línguas mantendo a compatibilidade com o ASCII
  - Até emoticons :D…

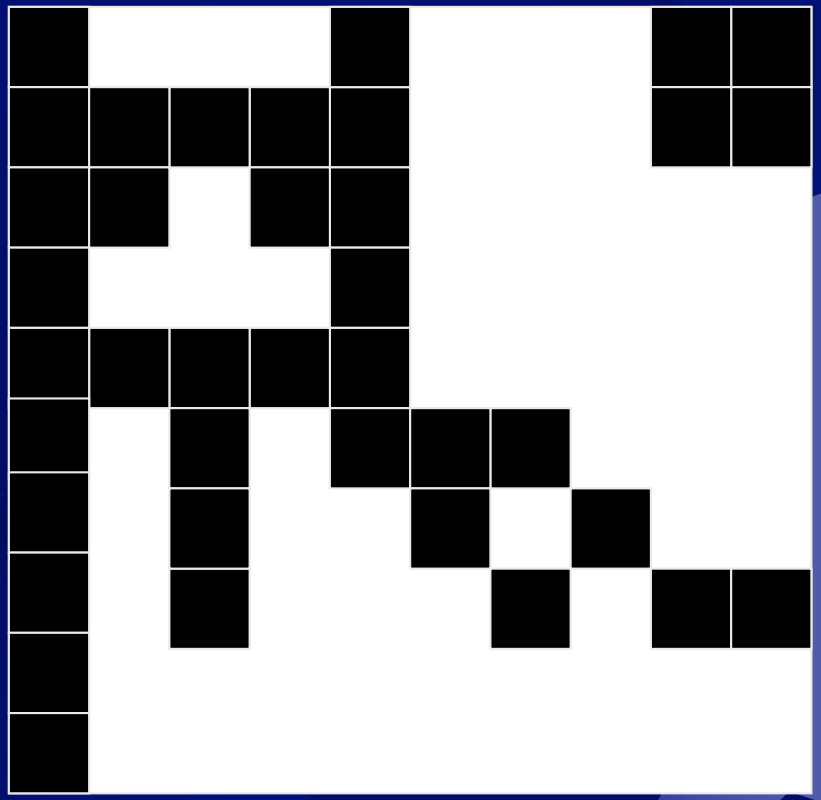
Extensões ao Código ASCII

- ISO-885 go ASCII a 8 bits que s caracteres próprios s da Europa Ocidenta ç”, “ñ”, “å”, “ø”, “€”, etc.
- UNICOD (actualmente até 32 bi car todos os caracter mantendo a compatib
  - Até emoticons :D…



Representação de imagem

- Imagens a preto e branco
  - Exemplo, uma imagem de tamanho 10x10 pixeis. Cada pixel preto é representado com ‘1’, cada pixel branco com ‘0’.
  - 100 bits são (mais que) suficientes para guardar a informação correspondente à imagem.



Representação de imagem

- Imagens a preto e branco
  - Exemplo, uma imagem de tamanho 10x10 pixeis. Cada pixel preto é representado com ‘1’, cada pixel branco com ‘0’.
  - 100 bits são (mais que) suficientes para guardar a informação correspondente à imagem.
- Imagens com 256 tons de cinzento
  - Para cada pixel serão necessários 8 bits (2^8=256) para representar o tom de cinzento
  - O número de bits necessário é dado pela dimensão da imagem multiplicada por 8…

Representação de imagem

- Imagens a preto e branco
  - Exemplo, uma imagem de tamanho 10x10 pixeis. Cada pixel preto é representado com ‘1’, cada pixel branco com ‘0’.
  - 100 bits são (mais que) suficientes para guardar a informação correspondente à imagem.
- Imagens com 256 tons de cinzento
  - Para cada pixel serão necessários 8 bits (2^8=256) para representar o tom de cinzento
- Imagens coloridas
  - O número de bits necessário para representar cada pixel depende de quão “detalhada” é a cor (e normalmente cada pixel contem informação para 3 cores: vermelho, verde e azul - RGB)
  - Um ecran com resolução 1366x768 (como este), quando utiliza 24 bits por pixel, pode representar 2^{24} = 16777216 cores diferentes (50% mais do que as que um olho humano pode distinguir)…
    - e são necessários 25178112 bits para representar a imagem apresentada no ecrã

Representação de imagem

- Imagens a preto e branco
  - Exemplo, uma imagem de tamanho 10x10 pixeis. Cada pixel preto é representado com ‘1’, cada pixel branco com ‘0’.
  - 100 bits são (mais que) suficientes para guardar a informação correspondente à imagem.
- Imagens com 256 tons de cinzento
  - Para cada pixel serão necessários 8 bits (2^8=256) para representar o tom de cinzento
- Imagens coloridas
  - O número de bits necessário para r… …pende de quão “detalhada” é a cor (e norm… …em informação para 3 cores: vermelho, ve…
  - Um ecran com resolução 1366x768 (c… …iza 24 bits por pixel, pode representar 2^{24} = 1… …ntes (50% mais do que as que um olho hur… .
    - e são necessários 25178112 bits p… …gem apresentada no ecrã

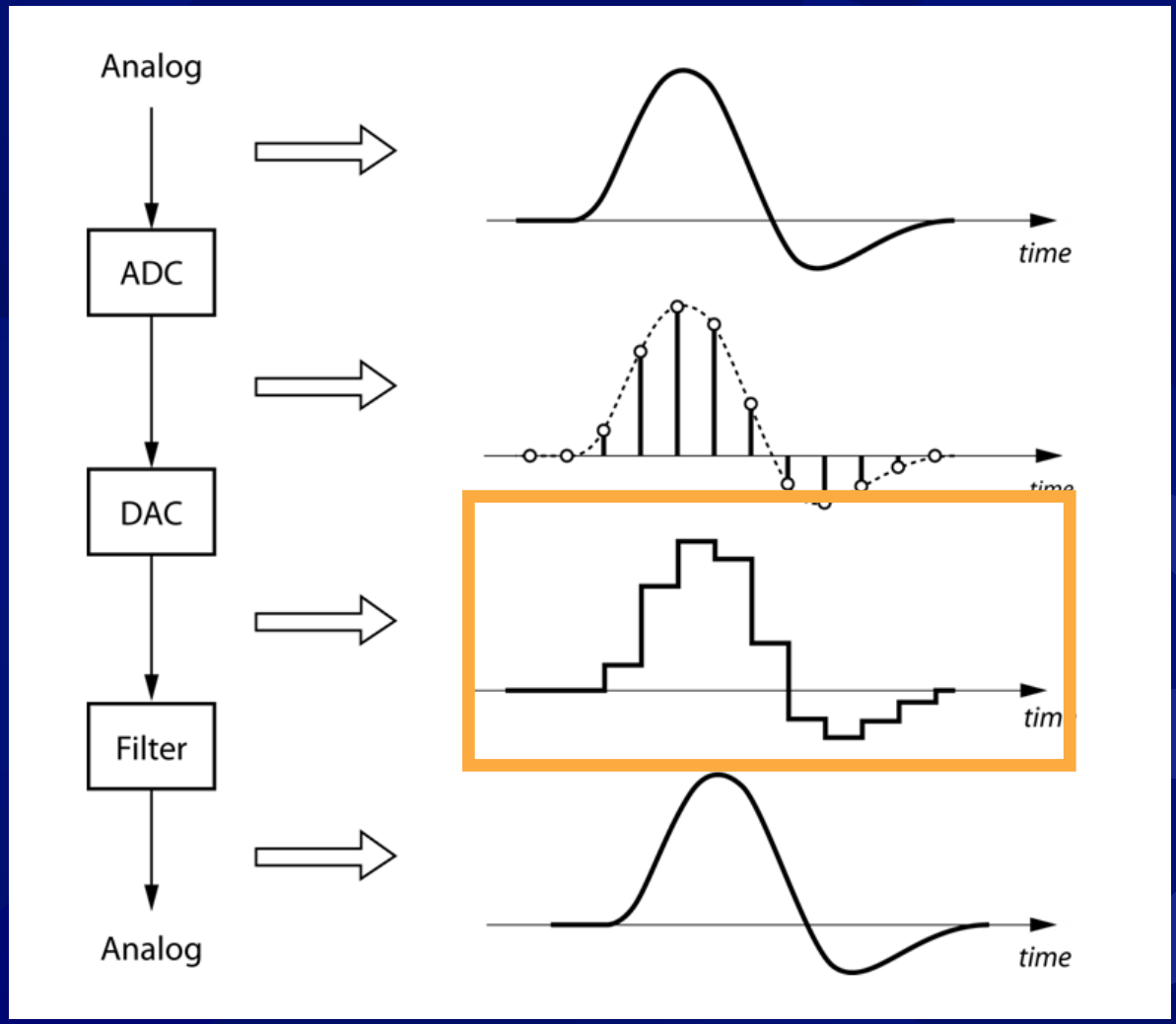
Representação de som

- O som é uma onda
- Para representar som num computador é necessário codificar a onda correspondente, num processo chamado conversão analógico-digital (ADC)

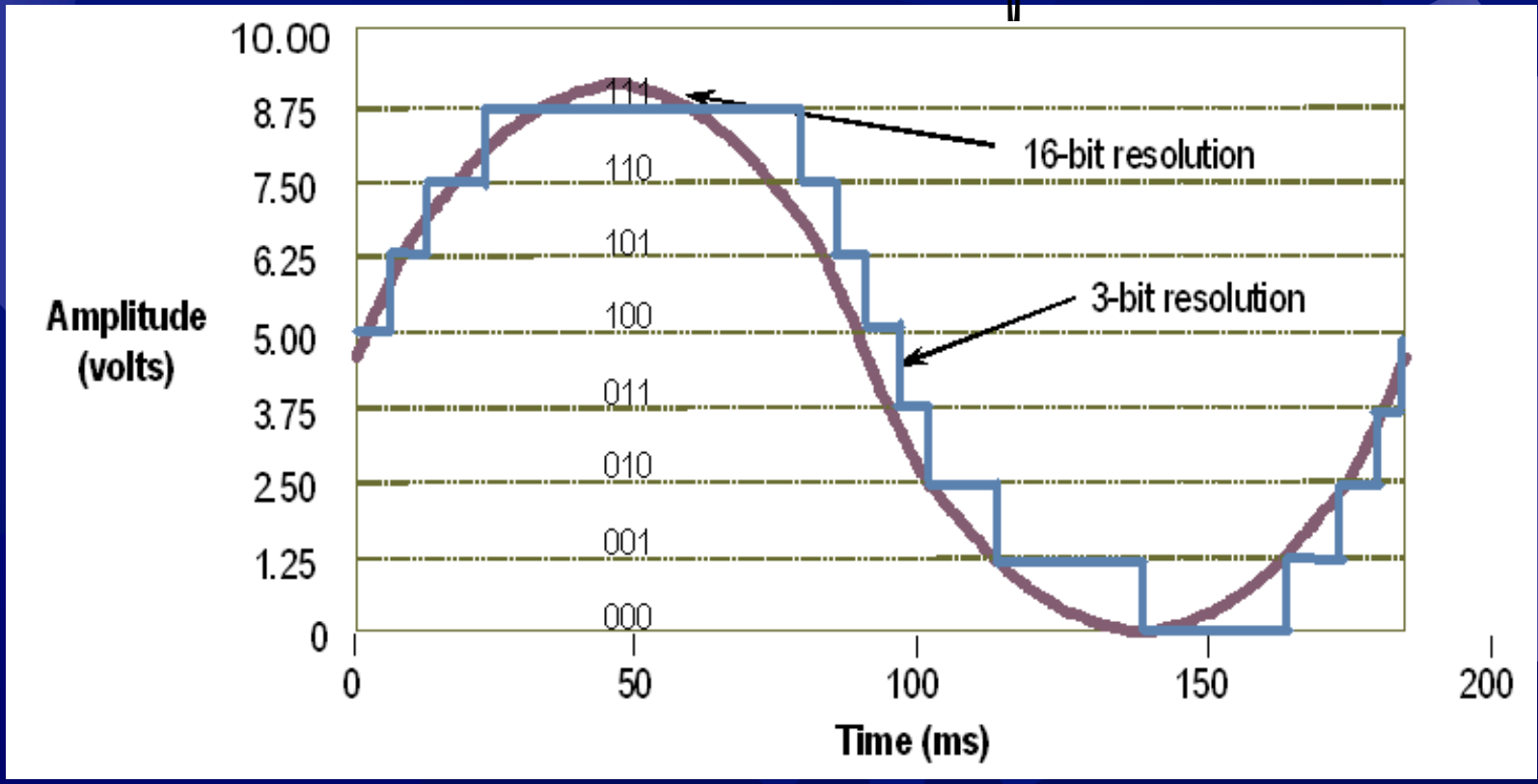
Representação de som

- O som é uma onda
- Para representar som num computador é necessário codificar a onda correspondente, num processo chamado conversão analógico-digital (ADC)
  - A amplitude máxima da onda (em volts) é representada por um determinado número de bits (resolução)
  - Em cada instante mede-se qual a amplitude da onda sonora, calcula-se e armazena-se o respectivo valor digital
- Ao processo inverso chama-se conversão digital-analógica (DAC)

Representação de som



Representação de som



Representação de som

- O som é uma onda
- Para representar som num computador é necessário codificar a onda correspondente, num processo chamado conversão analógico-digital (ADC)
  - A amplitude máxima da onda (em volts) é representada por um determinado número de bits (resolução)
  - Em cada instante mede-se qual a amplitude da onda sonora, calcula-se e armazena-se o respectivo valor digital
- Ao processo inverso chama-se conversão digital-analógica (DAC)
- Exemplo: CD estéreo
  - Resolução 16 bits; Frequência de amostragem 44.1Khz (x2)
    - Bitrate 1411Kbps (K bits por segundo)
    - Por cada segundo de música: 176.4KByte (1 hora = 635MB)

Bibliografia

- Arroz,G., Monteiro,J.C., Oliveira,A., "Arquitectura de Computadores – Dos Sistemas Digitais aos Microprocessadores", Capítulo 1, 2ª Edição, IST Press, 2009
- Sêrro,C. – "Sistemas Digitais: Fundamentos Algébricos", IST Press, 2003