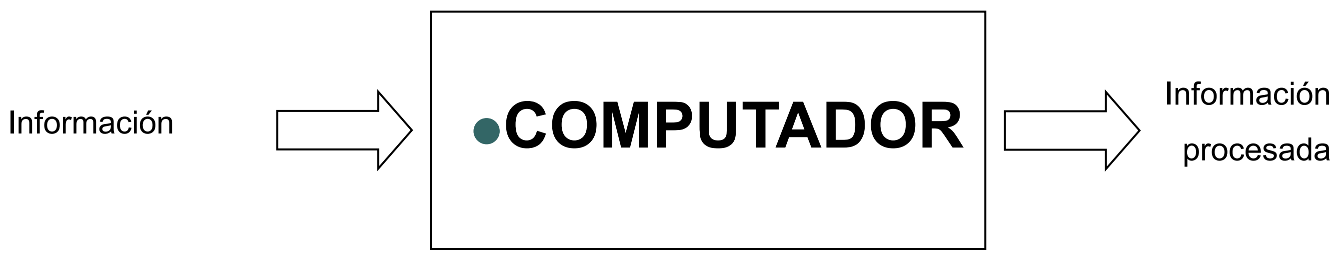
Representación de la Información en los Sistemas Digitales

© Luis Entrena, Celia López,
Mario García, Enrique San Millán

Universidad Carlos III de Madrid

Introducción a los computadores

- Computador: Máquina que procesa información



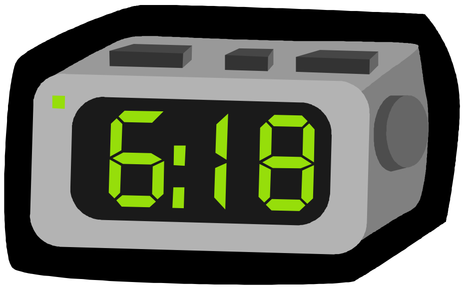
Sistemas analógicos y digitales

- Sistemas analógicos: aquellos cuyas variables toman valores continuos en el tiempo
  - Las magnitudes físicas son en su mayoría analógicas
- Sistemas digitales: aquellos cuyas variables toman valores discretos en el tiempo
  - Se utilizan valores discretos llamados dígitos
  - Precisión limitada
  - Las cantidades digitales son más fáciles de manejar
  - Las magnitudes analógicas se pueden convertir a magnitudes digitales mediante muestreo

Sistemas analógicos y digitales

- Sistema Analógico

- Sistema digital



Sistemas binarios

- Sistemas binarios: sistemas digitales que sólo utilizan dos posibles valores
  - Los digitos binarios se denominan bits (Binary digiT)
  - Se representan mediante los símbolos 0 y 1, ó L y H
  - Los sistemas binarios son casi los únicos utilizados. Por extensión, se utiliza el término digital como sinónimo de binario
- ¿Por qué binario?
  - Más fiable: mayor inmunidad frente al ruido
  - Más sencillo de construir: sólo hay que distinguir entre dos valores

Índice

- Sistemas de Numeración
- Conversiones entre sistemas de numeración
- Códigos Binarios:
    - Códigos BCD
    - Códigos progresivos y cíclicos
    - Códigos alfanuméricos
    - Códigos detectores y códigos correctores de errores
    - Representación de números enteros y reales

Sistemas de Numeración

- Permiten representar los números mediante dígitos
- El sistema que utilizamos habitualmente es el sistema decimal:
  - $N = a_n\ 10^n + a_{n-1}\ 10^{n-1} + \ldots + a_1\ 10 + a_0$
  - Ejemplo: $272_{10} = 2*10^2 + 7*10 + 2$
- Se puede hacer lo mismo pero utilizando bases diferentes a 10:
  - $N = a_n\ b^n + \underbrace{a_{n-1}}_{\text{Dígito}}\ \underbrace{b}_{\text{Base}}{}^{\overbrace{n-1}^{\text{Peso}}} + \ldots + a_1\ b + a_0$

Sistemas de Numeración

- En un sistema con base b los dígitos posibles son:
  - 0, 1, … , b-1
- Con n dígitos se pueden representar b^n números posibles, desde el 0 hasta el b^n-1
- Esta representación sirve también para números que no sean naturales:
  - Ejemplo: $727{,}23_{10} = 7*10^2 + 2*10 + 7 + 2*10^{-1} + 2*10^{-2}$
- Los sistemas que se utilizan en los sistemas digitales son: binario (b=2), octal (b=8) y hexadecimal(b=16)

Sistema Binario

- En este sistema la base es 2. Permiten representar perfectamente la información en los sistemas digitales.
  - Los dígitos posibles son 0 y 1. Un dígito en sistema binario se denomina “bit”.
  - Con n bits se pueden representar 2^n números
- El bit de mayor peso se denomina **bit más significativo** o **MSB** (“Most Significant Bit”), y el bit de menor peso se denomina **bit menos significativo** o **LSB** (“Least Significant Bit”)

Habitualmente el MSB se escribe a la izquierda y el LSB a la derecha

MSB LSB

  - Ejemplo: $1001010_2 = 1{*}2^6 + 1{*}2^3 + 1{*}2^1 = 74_{10}$

Sistema Octal

- En este sistema la base es 8.
  - Los dígitos son 0,1,2,3,4,5,6,7
  - Con n dígitos se pueden representar 8^n números
- Está muy relacionado con el sistema binario (8 es una potencia de 2, en concreto $2^3=8$)
  - Esto permite convertir fácilmente de octal a binario y de binario a octal
- Ejemplo:

$$137_8 = 1*8^2 + 3*8^1 + 7*8^0 = 95_{10}$$

Sistema Hexadecimal

- En este sistema la base es 16.
  - Los dígitos son 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
  - Está relacionado con el sistema binario (2^4=16)
  - Un dígito hexadecimal permite representar lo mismo que 4 bits (ya que 2^4=16). Un dígito hexadecimal se denomina también "**nibble**".
  - Dos dígitos hexadecimales equivalen por tanto a 8 bits. El conjunto de 8 bits o dos dígitos hexadecimales, se denomina "**byte**".
- Notaciones: $23AF_{16} = 23AF_{hex}$ = 23AFh = 0x23AF = 0x23 0xAF.
- Ejemplo: 23AFh = $2*16^3 + 3*16^2 + 10*16 + 15 = 9135_{10}$

Conversiones entre sistemas de numeración

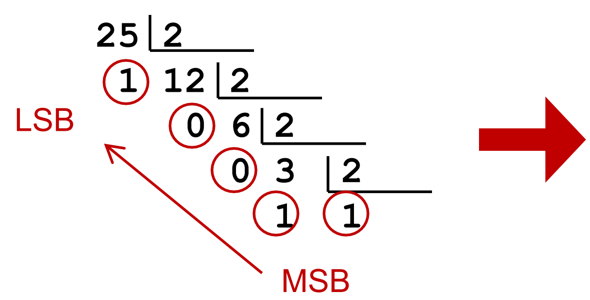
- Pasar de cualquier sistema a sistema decimal:
  - $N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0$
  - Ejemplos:
    - $1001010_2 = 1*2^6 + 1*2^3 + 1*2^1 = 74_{10}$
    - $137_8 = 1*8^2 + 3*8^1 + 7*8^0 = 95_{10}$
    - $23AFh = 2*16^3 + 3*16^2 + 10*16 + 15 = 9135_{10}$
- Para pasar de decimal a otro sistema:
  - Método de descomposición en pesos
  - Método de divisiones sucesivas por la base

Método de descomposición en pesos

- Consiste en descomponer el número en potencias de la base.
  - Se busca la potencia de la base (menor) más cercana al número.
  - Se van buscando potencias sucesivamente para que la suma de todas ellas sea el número decimal que se quiere convertir.
  - Finalmente los pesos de las potencias utilizadas se utilizan para representar el número en la base buscada.
- Éste método es útil sólo para sistemas donde las potencias de la base son conocidas. Por ejemplo para sistema binario: 1, 2, 8, 16, 32, 64, 128, 256, …
- Ejemplo:
  - $25_{10} = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 = 11001_2$

Método de divisiones sucesivas por la base

- Consiste en dividir el número decimal a convertir sucesivamente por la base y los cocientes obtenidos en las divisiones anteriores
  - El último cociente obtenido es el MSB del resultado
  - Los restos obtenidos son el resto de dígitos, siendo el primero de los restos obtenidos el LSB
- Ejemplo:



$25_{10} = 11001_2$

- Este método es más general que el anterior. Sirve para convertir de decimal a cualquier otra base.

Conversión de números reales

- La conversión de binario a decimal se hace igual que para números enteros (utilizando pesos negativos para la parte decimal):

$$101{,}011_2 = 1{*}2^2 + 0{*}2^1 + 1{*}2^1 + 0{*}2^{-1} + 1{*}\ 2^{-2} + 1{*}\ 2^{-3} =$$
$$= 4 + 1 + 0{,}25 + 0{,}125 = 5{,}375_{10}$$

- La conversión de decimal a binario, se hace en dos partes:
  - Se convierte primero la parte entera por el método de divisiones sucesivas por la base (o por descomposición de pesos)
  - Luego se convierte la parte decimal por un método análogo, multiplicaciones sucesivas por la base.

Método de multiplicaciones sucesivas por la base (parte decimal)

- Consiste en multiplicar la parte decimal del número por la base sucesivamente.
  - Se multiplica la parte decimal del número por 2. La parte entera del resultado es el primer dígito (MSB de la parte decimal) de la conversión
  - Se vuelve a tomar la parte decimal, y se multiplica por 2 otra vez, y nuevamente la parte entera es el siguiente dígito.
  - Se itera tantas veces como se quiera, según la precisión que se quiera obtener en la conversión.
- Ejemplos:

$0{,}3125_{10} = 0{,}0101_2$

0,3125 x 2 = 0,625 => 0
0,625 x 2 = 1,25 => 1
0,25 x 2 = 0,5 => 0
0,5 x 2 = 1 => 1

$0{,}1_{10} = 0{,}0\ 0011\ 0011\ \ldots_2$

0,1 x 2 = 0,2 => 0
0,2 x 2 = 0,4 => 0
0,4 x 2 = 0,8 => 0
0,8 x 2 = 1,6 => 1
0,6 x 2 = 1,2 => 1
0,2 x 2 = 0,4 => 0 <- se repiten las cuatro cifras, periódicamente
0,4 x 2 = 0,8 => 0
0,8 x 2 = 1,6 => 1
...

Otros métodos de conversión

- Los sistemas octal y hexadecimal están relacionados con el binario, ya que sus bases son potencias exactas de 2 (la base binaria). Esto permite convertir entre estos sistemas de forma muy sencilla:
  - **OCTAL a BINARIO**: Convertir cada dígito en 3 bits
    - Ejemplo: $735_8 = 111\ 011\ 101_2$
  - **BINARIO a OCTAL**: Agrupar en grupos de 3 bits y convertirlos de forma independiente a octal.
    - Ejemplo: $1\ 011\ 100\ 011_2 = 1343_8$
  - **HEXADECIMAL a BINARIO**: Convertir cada dígito en 4 bits
    - Ejemplo: $3B2h = 0011\ 1011\ 0010_2$
  - **BINARIO a HEXADECIMAL**: Agrupar en grupos de 4 bits y convertirlos de forma independiente a octal
    - Ejemplo: $10\ 1110\ 0011_2 = 2E3h$

Códigos Binarios

- Los códigos binarios son códigos que utilizan únicamente 0s y 1s para representar la información
- La información que se puede representar con códigos binarios puede ser de múltiples tipos:
  - Números naturales
  - Números enteros
  - Números reales
  - Caracteres alfabéticos y otros símbolos
- Una misma información (por ejemplo un número natural) se puede representar utilizando diferentes códigos.
  - Es importante especificar siempre qué código que se está utilizando cuando se representa una información en un código binario.

Código Binario Natural

- Es un código binario en el que se representa un número natural mediante su representación en sistema binario
  - Es el código binario más simple
  - Aprovecha que la representación en sistema binario de un número natural utiliza únicamente 0s y 1s
- Notación: Utilizaremos el indicador “BIN” para indicar que un código binario es el código binario natural:
  - $1001_{BIN} = 1001_2$

Códigos BCD (“Binary-Coded Decimal”)

- Permiten representar números naturales de una forma alternativa al binario natural.
- Se asigna un código de 4 bits a cada dígito decimal. Un número decimal se codifica en BCD dígito a dígito.
- El código BCD más habitual es el BCD natural (existen otros códigos BCD).
- Ejemplo:
  - $78_{10} = 0111\ 1000_{BCD}$
- La codificación BCD de un número no tiene por qué coincidir con el código binario natural:
  - $78_{10} = 1001110_{BIN}$
- INCONVENIENTE: No todas las combinaciones corresponden a un código BCD. Por ejemplo,1110_{BCD} no existe.
- VENTAJA: Facilidad de conversión decimal-binario.

dígito decimal	Código BCD
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Códigos progresivos y cíclicos

- Dos codificaciones binarias se dice que son **adyacentes** si sólo hay bit diferente entre ambas.
  - **0000** y **0001** son adyacentes, ya que sólo difieren en el último bit
  - **0001** y **0010** no son adyacentes, ya que los dos últimos bits son diferentes
- Un código se dice que es **progresivo** si todas las codificaciones consecutivas son adyacentes.
  - El código binario natural no es progresivo, ya que 0001 y 0010 no son adyacentes.
- Un código se dice que es **cíclico** si además la primera y la última codificación son adyacentes.
- Los códigos progresivos y cíclicos más utilizados son:
  - Código Gray
  - Código Johnson

Código Gray

- El Código Gray es un código progresivo y cíclico
- Ejemplo de Código Gray de 3 bits:

Decimal	Código Gray
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

Todas las codificaciones consecutivas son adyacentes

Código Gray

- Construcción del código Gray de n bits:
  - Primero se copian los códigos de n-1 bits y se añaden otros n-1 copiando los anteriores en orden inverso
  - Luego se añade un cero a la izquierda en los de arriba y un uno en los de abajo
- Código de 1 bit:

  0
  1

- Código de 2 bits:

  0
  1
  ———
  1
  0

  →

  0 0
  0 1
  ———
  1 1
  1 0

Código Gray

- Código de 3 bits:

0	0
0	1
1	1
1	0
1	0
1	1
0	1
0	0

→

0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

- Iterando se pueden construir los códigos Gray de n bits

Conversión entre los códigos Gray y Binario Natural

Se puede convertir directamente de Gray a Binario y de Binario a Gray, sin necesidad de construir toda la tabla:

BINARIO A GRAY:

$(A_0 A_1 A_2 \ldots A_n)_{BIN} \rightarrow (B_0 B_1 B_2 \ldots B_n)_{GRAY}$

- $B_0 = A_0$
- $B_1 = A_0 + A_1$
- $B_2 = A_1 + A_2$
- …
- $B_n = A_{n-1} + A_{n-2}$

Ejemplo:
$1011_{BIN} \rightarrow 1110_{GRAY}$

GRAY A BINARIO:

$(A_0 A_1 A_2 \ldots A_n)_{GRAY} \rightarrow (B_0 B_1 B_2 \ldots B_n)_{BIN}$

- $B_0 = A_0$
- $B_1 = A_1 + B_0$
- $B_2 = A_2 + B_1$
- …
- $B_n = A_n + B_{n-1}$

Ejemplo:
$1011_{GRAY} \rightarrow 1101_{BIN}$

BIN				GRAY			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Código Johnson

- Es otro código progresivo y cíclico
- En cada codificación aparecen agrupados los ceros a la izquierda y los unos a la derecha, o viceversa.
- Ejemplo código Johnson de 3 bits:

Decimal	Johnson		
0	0	0	0
1	0	0	1
2	0	1	1
3	1	1	1
4	1	1	0
5	1	0	0

Códigos alfanuméricos

- Representan símbolos, que pueden ser:
  - Dígitos
  - Letras mayúsculas y minúsculas
  - Signos de puntuación
  - Caracteres de control (espacio, salto de línea, retorno de carro, etc.)
  - Otros símbolos gráficos (operadores matemáticos, etc.)
- Un código alfanumérico mínimo que contenga los 10 dígitos, las 26 letras del alfabeto inglés, mayúsculas y minúsculas (52), necesita al menos 6 bits.
- Los códigos más utilizados en la actualidad son:
  - Código ASCII (7 bits)
  - Códigos ASCII extendidos (8 bits)
  - Códigos unicode (8-32 bits)

Códigos ASCII y ASCII extendidos

- El código ASCII (“American Standard Code for Information Interchange”) fue publicado por primera vez en 1963.
- Es un código de 7 bits (128 códigos) estándar que contiene:
  - Dígitos
  - Letras mayúsculas y minúsculas del alfabeto inglés internacional
  - Signos de puntuación
  - Caracteres básicos de control
- Los códigos ASCII extendidos se utilizan para añadir caracteres adicionales:
  - No son estándar, difieren de una región a otra
  - Los 128 primeros códigos coinciden con el ASCII estándar por compatibilidad

Código ASCII Estándar

	0	1	2	3	4	5	6	7
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1 XON	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3 XOFF	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	del

Códigos ASCII Extendidos

EJEMPLO:

ACII extendido LATIN-1

(ISO 8859-1)

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B-	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C-	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D-	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E-	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F-	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Códigos ASCII Extendidos

EJEMPLO:

ASCII extendido Cirílico

ISO 8859-5

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	008A	008B	008C	008D	008E	008F
9-	0090	0091	0092	0093	0094	0095	0096	0097	0098	0099	009A	009B	009C	009D	009E	009F
A-	00A0	Ё 0401	Ђ 0402	Ѓ 0403	Є 0404	Ѕ 0405	І 0406	Ї 0407	Ј 0408	Љ 0409	Њ 040A	Ћ 040B	Ќ 040C	- 00AD	Ў 040E	Џ 040F
B-	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
C-	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
D-	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
E-	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F
F-	№ 2116	ё 0451	ђ 0452	ѓ 0453	є 0454	ѕ 0455	і 0456	ї 0457	ј 0458	љ 0459	њ 045A	ћ 045B	ќ 045C	§ 00A7	ў 045E	џ 045F

Códigos Unicode

- Los códigos Unicode (“Universal Code”) fueron creados en 1991 para tener códigos alfanuméricos estándar, comunes en todas las regiones
  - Se utiliza el mismo código unicode para idiomas Chino, Árabe, etc.
- Como máximo necesitan 32 bits
  - Los primeros 7 bits permiten la compatibilidad con ASCII
  - Con 1 byte se puede representar el código US-ASCII
  - Con 2 bytes: caracteres latinos y alfabetos árabes, griego, cirílico, armenio, hebreo, sirio y thaana.
  - Con 3 bytes: resto de caracteres utilizados en todos los lenguajes
  - Con 4 bytes: caracteres gráficos y poco comunes
- Diferentes versiones de representación. Las más comunes:
  - **UTF-8**: Códigos de 1 byte, pero son de longitud variable (se pueden utilizar 4 grupos de 1 byte para representar un símbolo)
  - **UCS-2**: Códigos de 2 bytes de longitud fija
  - **UTF-16**: Códigos de 2 bytes, de longitud variable (se pueden utilizar 2 grupos de 2 bytes para representar un símbolo)
  - **UTF-32**: Códigos de 4 bytes

Códigos Unicode

EJEMPLO:

- Parte del Unicode correspondiente al alfabeto cirílico

    Se necesita el segundo byte para la representación

- Las codificaciones completas se pueden encontrar en:

    http://www.unicode.org/charts

0400 Cyrillic 04FF

	040	041	042	043	044	045	046	047	048	049	04A	04B	04C	04D	04E	04F
0	Ѐ 0400	А 0410	Р 0420	а 0430	р 0440	ѐ 0450	Ѡ 0460	Ѱ 0470	Ҁ 0480	Ґ 0490	Ҡ 04A0	Ұ 04B0	Ӏ 04C0	Ӑ 04D0	Ӡ 04E0	Ӱ 04F0
1	Ё 0401	Б 0411	С 0421	б 0431	с 0441	ё 0451	ѡ 0461	ѱ 0471	ҁ 0481	ґ 0491	ҡ 04A1	ұ 04B1	Ӂ 04C1	ӑ 04D1	ӡ 04E1	ӱ 04F1
2	Ђ 0402	В 0412	Т 0422	в 0432	т 0442	ђ 0452	Ѣ 0462	Ѳ 0472	҂ 0482	Ғ 0492	Ң 04A2	Ҳ 04B2	ӂ 04C2	Ӓ 04D2	Ӣ 04E2	Ӳ 04F2
3	Ѓ 0403	Г 0413	У 0423	г 0433	у 0443	ѓ 0453	ѣ 0463	ѳ 0473	◌҃ 0483	ғ 0493	ң 04A3	ҳ 04B3	Ӄ 04C3	ӓ 04D3	ӣ 04E3	ӳ 04F3
4	Є 0404	Д 0414	Ф 0424	д 0434	ф 0444	є 0454	Ѥ 0464	Ѵ 0474	◌҄ 0484	Ҕ 0494	Ҥ 04A4	Ҵ 04B4	ӄ 04C4	Ӕ 04D4	Ӥ 04E4	Ӵ 04F4
5	Ѕ 0405	Е 0415	Х 0425	е 0435	х 0445	ѕ 0455	ѥ 0465	ѵ 0475	◌҅ 0485	ҕ 0495	ҥ 04A5	ҵ 04B5	Ӆ 04C5	ӕ 04D5	ӥ 04E5	ӵ 04F5
6	І 0406	Ж 0416	Ц 0426	ж 0436	ц 0446	і 0456	Ѧ 0466	Ѷ 0476	◌҆ 0486	Җ 0496	Ҧ 04A6	Ҷ 04B6	ӆ 04C6	Ӗ 04D6	Ӧ 04E6	Ӷ 04F6
7	Ї 0407	З 0417	Ч 0427	з 0437	ч 0447	ї 0457	ѧ 0467	ѷ 0477	◌҇ 0487	җ 0497	ҧ 04A7	ҷ 04B7	Ӈ 04C7	ӗ 04D7	ӧ 04E7	ӷ 04F7
8	Ј 0408	И 0418	Ш 0428	и 0438	ш 0448	ј 0458	Ѩ 0468	Ѹ 0478	◌҈ 0488	Ҙ 0498	Ҩ 04A8	Ҹ 04B8	ӈ 04C8	Ә 04D8	Ө 04E8	Ӹ 04F8
9	Љ 0409	Й 0419	Щ 0429	й 0439	щ 0449	љ 0459	ѩ 0469	ѹ 0479	◌҉ 0489	ҙ 0499	ҩ 04A9	ҹ 04B9	Ӊ 04C9	ә 04D9	ө 04E9	ӹ 04F9
A	Њ 040A	К 041A	Ъ 042A	к 043A	ъ 044A	њ 045A	Ѫ 046A	Ѻ 047A	Ҋ 048A	Қ 049A	Ҫ 04AA	Һ 04BA	ӊ 04CA	Ӛ 04DA	Ӫ 04EA	Ӻ 04FA
B	Ћ 040B	Л 041B	Ы 042B	л 043B	ы 044B	ћ 045B	ѫ 046B	ѻ 047B	ҋ 048B	қ 049B	ҫ 04AB	һ 04BB	Ӌ 04CB	ӛ 04DB	ӫ 04EB	ӻ 04FB
C	Ќ 040C	М 041C	Ь 042C	м 043C	ь 044C	ќ 045C	Ѭ 046C	Ѽ 047C	Ҍ 048C	Ҝ 049C	Ҭ 04AC	Ҽ 04BC	ӌ 04CC	Ӝ 04DC	Ӭ 04EC	Ӽ 04FC
D	Ѝ 040D	Н 041D	Э 042D	н 043D	э 044D	ѝ 045D	ѭ 046D	ѽ 047D	ҍ 048D	ҝ 049D	ҭ 04AD	ҽ 04BD	Ӎ 04CD	ӝ 04DD	ӭ 04ED	ӽ 04FD
E	Ў 040E	О 041E	Ю 042E	о 043E	ю 044E	ў 045E	Ѯ 046E	Ѿ 047E	Ҏ 048E	Ҟ 049E	Ү 04AE	Ҿ 04BE	ӎ 04CE	Ӟ 04DE	Ӯ 04EE	Ӿ 04FE
F	Џ 040F	П 041F	Я 042F	п 043F	я 044F	џ 045F	ѯ 046F	ѿ 047F	ҏ 048F	ҟ 049F	ү 04AF	ҿ 04BF	ӏ 04CF	ӟ 04DF	ӯ 04EF	ӿ 04FF

Códigos detectores y correctores de errores

- En los sistemas digitales pueden aparecer errores
  - Errores físicos de los circuitos
  - Interferencias electromagnéticas (EMI)
  - Fallos de alimentación eléctrica
  - Etc.
- Códigos detectores de error:
  - Pueden permitir detectar un error en la codificación
- Códigos correctores de error:
  - Permiten detectar un error y además corregirlo
- Los códigos detectores de error y los códigos correctores de error no utilizan las 2^n posibles codificaciones con n bits

Códigos detectores de error

- Códigos de paridad:
  - Añaden un bit adicional (paridad del número) que permite detectar errores simples en la codificación (error en 1 bit)
  - La paridad que se considera es la de la **suma** de los n bits de la codificación
    - **NOTA**: la paridad no tiene nada que ver con si la codificación binaria es par o impar (un número binario es par si acaba en 0 e impar si acaba en 1).
  - Dos posibles convenios:
    - Añadir un 0 cuando la paridad sea par y 1 cuando sea impar: Se denomina código de paridad par (ya que considerando la suma de los n bits + el bit de paridad la paridad siempre es par)
    - Añadir un 1 cuando la paridad sea par y 0 cuando sea impar: Se denomina código de paridad impar (ya que la suma de los n bits + bit paridad es siempre impar)

Códigos detectores de error

- Ejemplo de paridad:

Código detector de errores (código de paridad impar) a partir del código binario natural de dos bits:

0	0	1
0	1	0
1	0	0
1	1	1

- Ejemplo de utilidad del código detector:

Si utilizamos este código en una comunicación entre dos sistemas binarios, el sistema receptor podría detectar si hay un error comprobando la paridad.

Ejemplo: Se transmite la codificación 001, y el receptor recibe la codificación 000 (error en el último bit).

Paridad de 001: impar
Paridad de 000: par

No coinciden:
Error detectado

Códigos detectores de error

- Existen más códigos detectores de error:
  - Número de unos:
    - Se añade a la codificación la suma de unos de la codificación (no sólo la paridad de la suma, sino la suma completa)
  - Número de transiciones:
    - Se añade a la codificación el número de transiciones de 0 a 1 y de 1 a 0 en la codificación
  - Códigos CRC (Cyclic Redundancy Checking):
    - Buscan añadir el menor número posible de bits que permitan detectar el mayor número posible de fallos
    - Estos códigos también permiten corregir algunos errores
- Los códigos más utilizados son los de **paridad** (por su sencillez) y **CRC** (por su eficacia)

Códigos correctores de error

- Los códigos correctores permiten no sólo detectar sino también pueden corregir un error.
- Para que un código permita corregir errores, la distancia mínima (número mínimo de bits diferentes entre dos codificaciones) debe ser mayor de 2.
  - Se puede corregir la codificación buscando la codificación más cercana perteneciente al código
- Hamming describió un método general para construir códigos con distancia mínima de 3, conocidos como **códigos de Hamming**
- Estos códigos son importantes, a partir de ellos se obtienen muchos de los utilizados en sistemas de comunicaciones (por ejemplo los códigos de bloque Reed-Solomon)

Codificación de números enteros y reales

- Además de los códigos binarios vistos hasta ahora, hay otros códigos importantes que se utilizan para representar números enteros y números reales:
  - Números enteros: Códigos de signo y magnitud, Complemento a Uno, Complemento a Dos
  - Números reales: Códigos de Punto fijo y Coma Flotante
- Estos códigos se estudiarán en detalle en el Tema 4: Aritmética Binaria

Referencias

- Fundamentos de Sistemas Digitales. Thomas L. Floyd. Pearson Prentice Hall
- Introducción al Diseño Lógico Digital. John P. Hayes. Addison-Wesley
- Diseño Digital. John F. Wakerly. Pearson Prentice Hall