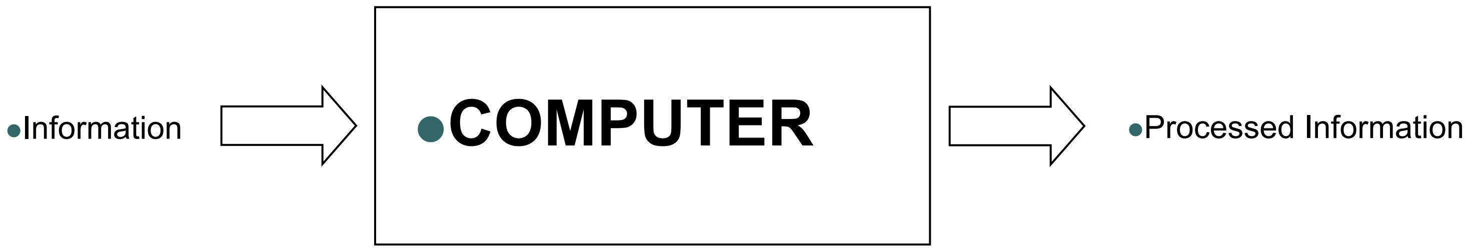
Representation of Information in Digital Systems

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

Universidad Carlos III de Madrid

Introduction to computers

- Computer: Machine that processes information



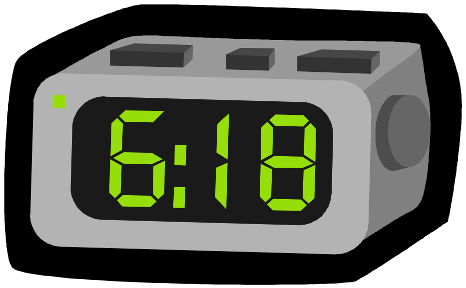
Analog and Digital Systems

- Analog Systems: Systems where variables have continuous values
  - Physical magnitudes are usually analog
- Digital systems: Systems where variables have discrete values
  - Discrete values are called digits
  - Limited precision
  - Digital magnitudes are easier to handle
  - Analog magnitudes can be converted to digital using sampling

Analog and Digital Systems

- Analog System

- Digital System



Binary Systems

- Binary Systems: Digital systems that use only two possible values
  - Binary digits are named bits (BInary Digit)
  - They are represented with symbols 0 and 1, or L and H
  - Binary Systems are almost the only digital systems used. By extension, the term digital is used as a synonym of binary
- ¿Why binary?
  - More reliability: more inmunity to noise
  - Easier to build: only two values to distinguish

Outline

- Number Systems
- Number Systems Conversions
- Binary Codes:
  - BCD Codes
  - Progressive and cyclic codes
  - Alphanumeric codes
  - Error detection and error correction codes
  - Real and integer numbers representation

Number Systems

- Numbers are represented using digits
- The system we commonly use is decimal:
  - $N = a_n\,10^n + a_{n-1}\,10^{n-1} + \ldots + a_1\,10 + a_0$
  - Example: $272_{10} = 2{*}10^2 + 7{*}10 + 2$
- The same representation can be used with different bases:

  - $N = a_n\,b^n + \underbrace{a_{n-1}}_{\text{Digit}}\,\underbrace{b}_{\text{Base}}{}^{\overbrace{n-1}^{\text{Weight}}} + \ldots + a_1\,b + a_0$

Number Systems

- In a system using base b, possible digits are:
  - 0, 1, … , b-1
- Using n digits, b^n different possible numbers can be represented, from 0 to b^n-1
- This representation can be used for not natural numbers as well:
  - Example: $727,23_{10} = 7*10^2 + 2*10 + 7 + 2*10^{-1} + 2* 10^{-2}$
- The numeral systems used in digital systems are: binary (b=2), octal (b=8) and hexadecimal (b=16)

Binary System

- In this system the base is 2.
  - Possible digits are 0 and 1. A digit in binary system is named “bit”.
  - 2^n different numbers can be represented using n bits.
- The bit with highest weight is called **MSB** (“Most Significant Bit”), and the lowest weighted bit is called **LSB** (“Least Significant Bit”)

Usually the most significant bit is written to the left, and the least significant bit is written at the right

MSB LSB

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

Octal Number System

- In this system the base is 8
  - Digits are 0,1,2,3,4,5,6,7
  - 8^n different numbers can be represented with n digits
- It is related to the binary system (8 is a power of 2, $2^3=8$)
  - This relationship allows to convert easily from octal to binary and from binary to octal.
- Example:

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

Hexadecimal Number System

- In this system de base is 16.
  - Digits are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.
  - It is related to binary system as well ($2^4=16$)
  - A hexadecimal digit allows to represent the same as 4 bits (because $2^4=16$). An hexadecimal digit can be named as "**nibble**".
  - Two hexadecimal digits are equivalent to 8 bits. A set of 8 bits, or equivalently 2 hexadecimal digits, are called "**byte**".
- Notations: $23AF_{16} = 23AF_{hex}$ = 23AFh = 0x23AF = 0x23 0xAF.
- Example: $23AFh = 2*16^3 + 3*16^2 + 10*16 + 15 = 9135_{10}$

Number Systems Conversions

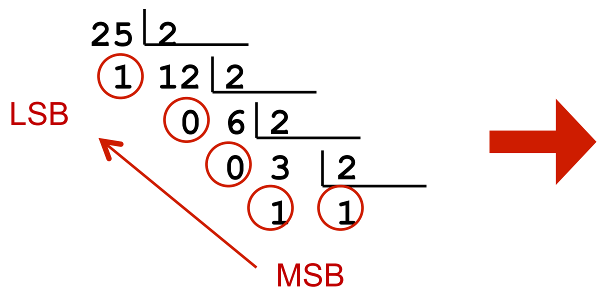
- Conversion from any system to decimal:
  - $N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0$
  - Examples:
    - $1001010_2 = 1*2^6 + 1*2^3 + 1*2^1 = 74_{10}$
    - $137_8 = 1*8^2 + 3*8^1 + 7*8^0 = 95_{10}$
    - $23AFh = 2*16^3 + 3*16^2 + 10*16 + 15 = 9135_{10}$
- Conversion from decimal to any other system:
  - Weight decomposition
  - Repeated division

Weight Decomposition

- The number is decomposed in powers of the base.
  - The nearest power of the base (lower) is searched.
  - Iteratively, powers of the base are been searched so that the sum of all of them is the decimal number to convert
  - Finally, the weights are used to represent the number in the desired base.
- This method is only useful for systems with well known powers. For example, for binary system: 1, 2, 8, 16, 32, 64, 128, 256, …
- Example:
  - $25_{10} = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 = 11001_2$

Repeated Division

- The number and the quotients in previous divisions are divided repeatedly by the destination base
  - The last quotient obtained is the MSB
  - The remainders are the other bits, the first one corresponding to the LSB.
- Example:



$25_{10} = 11001_2$

- This method is more general than the previous one. It can be used for any base conversion

Real numbers conversion

- Conversion from binary to decimal can be obtained using the same method as for integer numbers (just using negative weights for the decimal part) :

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

- Conversion from decimal to binary is obtained in two steps:
  - Convert first the integer part, using repeated division or weight decomposition.
  - Then convert the decimal part, using an analogous method: repeated multiplication by the base.

Repeated multiplication method (decimal part)

- The decimal part of the number is multiplied repeatedly by the base:
  - The decimal part is multiplied by 2. Then the integer part of the result is the first bit (MSB of the decimal part) of the conversion.
  - The obtained decimal part is multiplied by 2, and again, the integer part is the next digit of the conversion.
  - Iterate this procedure several times, depending on the desired precision for the conversion.
- Examples:

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

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 <- the last four digits will repeat periodically
0,4 x 2 = 0,8 => 0
0,8 x 2 = 1,6 => 1
...

Other conversion methods

- Octal and Hexadecimal number systems are related with binary because their bases are exact powers of the binary base. This makes very easy the conversion between these systems and binary.
  - **OCTAL to BINARY**: Convert each digit into binary (3 bits each digit)
    - Example: $735_8 = 111\ 011\ 101_2$
  - **BINARY to OCTAL**: Gruop
    - Example: $1\ 011\ 100\ 011_2 = 1343_8$
  - **HEXADECIMAL to BINARY**: Convert each digit into binary (4 bits each digit)
    - Example: $3B2h = 0011\ 1011\ 0010_2$
  - **BINARY to HEXADECIMAL**: Agrupar en grupos de 4 bits y convertirlos de forma independiente a octal
    - Example: $10\ 1110\ 0011_2 = 2E3h$

Binary Codes

- Binary codes are codes that use only 0s and 1s to represent information
- Information that can be represented with binary codes can be of several types:
  - Natural Numbers
  - Integer Numbers
  - Real Numbers
  - Alphanumeric characters and other symbols
- The same information (a natural number for example) can be represented using different codes
  - It is important to especify which encoding is been used when some information is represented with a binary code

Natural Binary Code

- It is a binary code where a natural number is represented using its binary number representation
  - It is the simplest binary code
  - This can be done because the binary number system for natural numbers needs only 0s and 1s (no extra symbols for decimal point or sign)
- Notation: The “BIN” subindex is used to especify that a binary code corresponds to the natural binary code.
  - $1001_{BIN} = 1001_2$

BCD Codes ("Binary-Coded Decimal")

- They are an alternative to the natural binary code for representation of natural numbers
- A 4-bit encoding is assigned to each decimal digit. A decimal number is encoded in BCD code digit to digit.
- The most common BCD code is natural BCD (there are other BCD codes).
- Example:
  - $78_{10} = 0111\ 1000_{BCD}$
- The BCD encoding of a number may be different to the natural binary encoding
  - $78_{10} = 1001110_{BIN}$
- CONS: No all encodings correspond to a binary BCD encoding. For example,1110_{BCD} does not exist.
- PRO: It is easy to convert natural numbers to BCD.

Decimal digit	BCD code
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

Progressive and Cyclic Codes

- Two binary encodings are **adjacent** if there is only 1 different bit between them.
  - **0000** y **0001** are adjacent, as they differ only in the last bit
  - **0001** y **0010** are not, because the last two bits are different
- A code is progressive is all consecutive encodings are **adjacent**.
  - Natural binary code is not progressive, as 0001 y 0010 are not adjacent.
- A code is **cyclic** if the first and the last encodings are adjacent.
- The most used progressive and cyclic codes are:
  - Gray code
  - Johnson code

Gray Code

- Gray codes are progressive and cyclic
- Example: 3-bit Gray Code

Decimal	Gray Code
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

All consecutive encodings are adjacent

Gray Code

- Construction of n-bit Gray codes:
  - First the n-1 bit code is copied. Then it is copied again in inverse order
  - Then a 0 is added in the first part of the table, and a 1 in the second part
- 1-bit code:

0
1

- 2-bits code:

0		0	0
1	→	0	1
1		1	1
0		1	0

Código Gray

- 3-bits code:

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

- n-bit Gray codes can be obtained by iteration

Binary-Gray and Gray-Binary conversions

It is possible to convert directly from Gray to Binary and from Binary to Gray, there is no need to build the whole table

BINARY TO GRAY:

$(A_0A_1A_2 \ldots A_n)_{BIN} \rightarrow (B_0B_1B_2 \ldots B_n)_{GRAY}$

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

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

GRAY TO BINARY:

$(A_0A_1A_2 \ldots A_n)_{GRAY} \rightarrow (B_0B_1B_2 \ldots B_n)_{BIN}$

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

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

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

Johnson Codes

- It is another progressive and cyclic code
- In each encoding, zeros are grouped to the left and ones to the right, or vice versa.
- Example: 3 bits Johnson code

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

Alphanumeric Codes

- They can represent different symbols:
  - Number Digits
  - Uppercase and lowercase letters
  - Punctuation marks
  - Control characters (espace, carriage return, line feed, etc.)
  - Other graphical symbols (mathematical operators, etc.)
- An alphanumeric code to represent at least 10 digits and 52 alphabet letters (26 lowercase and 26 uppercase) needs at least 6 bits.
- The most used alphanumeric codes are:
  - ASCII code (7 bits)
  - Extended ASCII codes (8 bits)
  - Unicode (8-32 bits)

ASCII codes and extended ASCII codes

- ASCII code (“American Standard Code for Information Interchange”) was publish for the first time in1963.
- It is a standard 7-bit code (128 encodings) which contains:
  - Digits
  - Uppercase and lower case letters (international English alphabet)
  - Punctuation marks
  - Basic control characters
- Extended ASCII codes are used to complement with additional characters:
  - Not standard, they change from a regional zone to another
  - The first 128 encodings are the same as in ASCII code for compatibility

Standard ASCII Code

	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

Extended ASCII Codes

EXAMPLE:

LATIN-1 extended ACII

(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

Extended ASCII Codes

Example:

Cyrillic extended ASCII

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

Unicode

- Unicode codes ("Universal Codes") were created in 1991 to introduce an standard alphanumeric code for all regions
  - The same code for languages like Chinese, Arabic, etc.
- Maximum 32 bits
  - First 7 bits allow compatibility with ASCII
  - Using 1 byte the US-ASCII can be represented
  - Using 2 bytes: latin, arabic, greek, cyrillic, armenian, hebrew, syriac and thaana alphabets
  - Using 3 bytes: rest of characters in remaining languages
  - Using 4 bytes: graphic characters and uncommon symbols
- Different versions of the representation. The most common are:
  - **UTF-8**: 1-byte codes, variable length (4 groups of 1 byte can be used to represent 1 symbol)
  - **UCS-2**: 1-byte codes, fixed lenght
  - **UTF-16**: 2-byte codes, variable length (2 groups of 2 bytes can be used to represent 1 symbol)
  - **UTF-32**: 4-byte codes

Unicode

Example:

- Unicode fragment, corresponding to Cyrillic alphabet

  A second byte is needed for the representation

- Full tables can be found at:

  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

Error detector and Error corrector codes

- Errors may appear in digital systems
  - Physical errors in the circuits
  - Electromagnetic interferences (EMI)
  - Power supply errors
  - Etc.
- Error detector codes:
  - They can detect an error in an encoding
- Error corrector codes:
  - They can detect an error an even correct it
- Error detector and error corrector codes don't use all 2^n posible n-bit encodings of the n-bit code

Error detector codes

- Parity codes:
  - An additional bit is added (parity) which allow to detect simple errors in the encoding
  - The considered parity is the sum of the encoding n-bits
    - **NOTE**: parity does not mean if a number is even or odd (a binary number is even if the last bit is 0 and odd if it is 1). In our case parity is related to the addition of all the bits in the encoding.
  - Two possible conventions:
    - Add 0 when parity is even and 1 if it is odd. In this case the parity code is named even parity code (as the addition of n bits + parity bit is always even)
    - Add 1 when parity is even and 0 if it is odd. In this case the parity code is named odd parity code (as the addition of n bits + parity bit is always odd)

Error detector codes

- Parity example:

  Error detector code (odd-parity code) obtained from a 2-bit natural binary code:

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

- Application example:

  If we use this code in a communication between two digital systems, the receiver may detect if there is an error in the transmitted encoding (checking the parity bit).

  Example: 001 is transmitted but the receiver recieves 000 (there is an error in the last bit)

  Parity of 001: odd

  Parity of 000: even

  Different:
  Error detected

Error detector codes

- There are more error detector codes:
  - Number of ones:
    - The sum of the ones in the encoding is added (not only the parity, but de full addition)
  - Number of transitions:
    - The number of transitions from 0 to 1 and 1 to 0 is added to the encoding
  - CRC codes (Cyclic Redundancy Checking):
    - They try to add the least possible number of bits to detect the maximum possible number or errors
    - Some CRC codes may also correct some errors
- The most used codes are parity (for simplicity) and CRC (for effectiveness)

Error corrector codes

- These codes allow not only to detect errors, they can correct them as well.
- The minimum distance (minimum number of different bits between two encodings) must be greater than 2 so that a code can correct errors.
  - The encoding can be corrected by looking for the closest encoding belonging to the code.
- Hamming showed a general method to obtain codes with minimum distance equal to 3, which are known as **Hammig codes**.
- These codes are important, many of the currently codes used in communications are obtained from them (for example Reed-Solomon codes)

Integer and real number codes

- There are more codes to represent integer and real numbers:
  - Integer numbers: Sign and magnitude, 1s-complement, 2s-complement
  - Real numbers: Fixed point and floating point
- We will see these codes in detail in unit 4 (Arithmetic combinational circuits)

References

- Digital Systems Fundamentals. Thomas L. Floyd. Pearson Prentice Hall
- Introduction to Digital Logic Design. John P. Hayes. Addison-Wesley
- Digital Design. John F. Wakerly. Pearson Prentice Hall