Sistemas Digitais I

LESI - 2º ano

Lesson 2 - Number Systems

Prof. João Miguel Fernandes
(miguel@di.uminho.pt)

Dept. Informática

2. Number Systems

- Positional Number Systems (1) -

- We use daily a positional number system.
- A number is represented by a string of decimal digits, where each digit position has an associated weight.
  - **5365 = 5*1000 + 3*100 + 6*10 + 5*1**
  - **162.39 = 1*100 + 6*10 + 2*1 + 3*0.1 + 9*0.01**
- A number D of the form $d_1 d_0 . d_{-1} d_{-2} d_{-3}$ has the value:
  $D = d_1 * 10^1 + d_0 * 10^0 + d_{-1} * 10^{-1} + d_{-2} * 10^{-2} + d_{-3} * 10^{-3}$
- 10 is called the base or the radix.
- Generally, the base can be any integer r >= 2 and a digit position i has weight r^i.

2. Number Systems

- Positional Number Systems (2) -

- The book “2+2=11” has a mathematically wrong title if we use the decimal base.
- In which base is the title correct?

- Natália Bebiano da Providência, 2+2=11, série “O Prazer da Matemática”, Gradiva, Lisboa, 2001. ISBN 972-622-809-1.

2. Number Systems

- Binary Numbers -

- Digital circuits have signals that are normally in one of two conditions (0 or 1, LOW or HIGH, charged or discharged).
- These signals represent binary digits (bits), that can have 2 possible values (0 or 1).
- The binary base (r=2) is used to represent numbers in digital systems.
- Examples of binary numbers and their decimal equivalents:
  - $\mathbf{11010_2 = 1*16 + 1*8 + 0*4 + 1*2 + 0*1 = 26_{10}}$
  - $\mathbf{100111_2 = 1*32 + 0*16 + 0*8 + 1*4 + 1*2 + 1*1 = 39_{10}}$
  - $\mathbf{10.011_2 = 1*2 + 0*1 + 0*0.5 + 1*0.25 + 1*0.125 = 2.375_{10}}$
- MSB: most significant bit; LSB: least significant bit.

2. Number Systems

- Octal and Hexadecimal Numbers (1) -

- The <u>octal</u> number system uses base 8 (r=8). It requires 8 digits, so it uses digits 0-7.
- The <u>hexadecimal</u> number system uses base 16 (r=16). It requires 16 digits, so it uses digits 0-9 and letters A-F.
- These number systems are useful for representing multibit numbers, because their bases are powers of 2.
- Octal digits can be represented by 3 bits, while hexadecimal digits can be represented by 4 bits.
- The octal number system was popular in the 70s, because certain computers had their front-panel lights arranged in groups of 3.
- Today, octal numbers are not used much, because of the preponderance of 8-bit bytes machines.

2. Number Systems

- Octal and Hexadecimal Numbers (2) -

- It is difficult to extract individual byte values in multibyte quantities represented in the octal system.
- What are the octal values of the 4 bytes in the 32-bit number with the octal representation 12345670123_8?
- $01\ 010\ 011\ 100\ 101\ 110\ 111\ 000\ 001\ 010\ 011_2$
  The 4 bytes in octal are: 123_8 227_8 160_8 123_8
- In the hexadecimal system, 2 digits represent a 8-bit byte, and 2n digits represent an n-byte word.
- Each pair of digits represent a byte.
- A 4-bit hexadecimal digit is sometimes called a <u>nibble</u>.

2. Number Systems

- Octal and Hexadecimal Numbers (3) -

Binary	*Decimal*	*Octal*	*3-Bit String*	*Hexadecimal*	*4-Bit String*
0	0	0	000	0	0000
1	1	1	001	1	0001
10	2	2	010	2	0010
11	3	3	011	3	0011
100	4	4	100	4	0100
101	5	5	101	5	0101
110	6	6	110	6	0110
111	7	7	111	7	0111
1000	8	10	—	8	1000
1001	9	11	—	9	1001
1010	10	12	—	A	1010
1011	11	13	—	B	1011
1100	12	14	—	C	1100
1101	13	15	—	D	1101
1110	14	16	—	E	1110
1111	15	17	—	F	1111

2. Number Systems

- Conversions (1) -

- It is easy to convert a binary number to octal or hexadecimal, and vice versa.
- Binary - Octal
  - **$110100101000_2 = 110\ 100\ 101\ 000_2 = 6450_8$**
  - **$11000110111010_2 = 011\ 000\ 110\ 111\ 010_2 = 30672_8$**
- Binary - Hexadecimal
  - **$110100101000_2 = 1101\ 0010\ 1000_2 = D28_{16}$**
  - **$11000110111010_2 = 0011\ 0001\ 1011\ 1010_2 = 31BA_{16}$**
- Octal - Binary
  - **$1324_8 = 001\ 011\ 010\ 100_2 = 1011010100_2$**
- Hexadecimal - Binary
  - **$19F_{16} = 0001\ 1001\ 1111_2 = 110011111_2$**

2. Number Systems

- Conversions (2) -

- In general, conversions between two bases cannot be done by simple substitutions. Arithmetic operations are required.
- Examples of conversions to the decimal base:
  - **$10001010_2 = 1*2^7 + 0*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 = 138_{10}$**
  - **$4063_8 = 4*8^3 + 0*8^2 + 6*8^1 + 3*8^0 = 2099_{10}$**
  - **$311.74_8 = 3*8^2 + 1*8^1 + 1*8^0 + 7*8^{-1} + 4*8^{-2} = 201{,}9375_{10}$**
  - **$19F_{16} = 1*16^2 + 9*16^1 + 15*16^0 = 415_{10}$**
  - **$134.02_5 = 1*5^2 + 3*5^1 + 4*5^0 + 0*5^{-1} + 2*5^{-2} = 44{,}08_{10}$**

2. Number Systems

- Conversions (3) -

- Example of Decimal to Binary Conversions ($138_{10} = 10001010_2$)
  - **138÷2 = 69 remainder 0**
  - **69÷2 = 34 remainder 1**
  - **34÷2 = 17 remainder 0**
  - **17÷2 = 8 remainder 1**
  - **8÷2 = 4 remainder 0**
  - **4÷2 = 2 remainder 0**
  - **2÷2 = 1 remainder 0**
  - **1÷2 = 0 remainder 1**

2. Number Systems

- Conversions (4) -

- Example of Decimal to Octal Conversions ($2099_{10} = 4063_8$)
  - **2099÷8 = 262 remainder 3**
  - **262÷8 = 32 remainder 6**
  - **32÷8 = 4 remainder 0**
  - **4÷8 = 0 remainder 4**
- Example of Decimal to Hexadecimal Conversions ($415_{10} = 19F_{16}$)
  - **415÷16 = 25 remainder 15 (F)**
  - **25÷16 = 1 remainder 9**
  - **1÷16 = 0 remainder 1**

2. Number Systems

- Addition of Binary Numbers -

- Addition and Subtraction of Non-Decimal Numbers use the same technique that we use for decimal numbers.
- The only difference is that the table are distinct.
- Table for addition of two binary digits.
- Similar tables can be built for other bases.
- Example of a binary addition:

c_{in}	x	y	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

				1	1	1	1			
X	190		1	0	1	1	1	1	1	0
Y	+ 141	+	1	0	0	0	1	1	0	1
$X+Y$	331	1	0	1	0	0	1	0	1	1

2. Number Systems

- Representation of Negative Numbers -

- There are many ways to represent negative numbers with bits.
  - **<u>Signed-Magnitude Representation</u>**
  - **Complement Number Systems**
  - **Radix-Complement Representation**
    - <u>Two's-Complement Representation</u>
  - **Diminished Radix-Complement Representation**
    - <u>One's-Complement Representation</u>
  - **Excess Representations**

2. Number Systems

- Signed-Magnitude Representation -

- A number consists of a magnitude and a symbol indicating whether the magnitude is positive or negative.
- In binary systems, we use an extra bit (usually the MSB) to indicate the sign (0=plus, 1=minus).
- Some 8-bit signed-magnitude integers:

  $\mathbf{01010101_2 = +85_{10}}$ $\mathbf{01111111_2 = +127_{10}}$ $\mathbf{00000000_2 = +0_{10}}$

  $\mathbf{11010101_2 = -85_{10}}$ $\mathbf{11111111_2 = -127_{10}}$ $\mathbf{10000000_2 = -0_{10}}$
- For n bits, number $\in$ $\{-2^{n-1}+1...2^{n-1}-1\}$; n=8, number $\in$ $\{-127...+127\}$.
- There are two representations of zero: “+0” e “-0”.

2. Number Systems

- Two's-Complement Representation -

- The radix-complement is called 2's-complement, for binary numbers. Most computers use it to represent negative numbers.
- The MSB of a number serves as the sign bit.
- The weight of the MSB is -2^{n-1}. The other bits have weight $+2^i$.
- For n bits, number $\in$ $\{-2^{n-1} \ldots 2^{n-1}-1\}$; n=8, number $\in$ $\{-128 \ldots +127\}$.
- Only one representation of zero $\Rightarrow$ an extra negative number.
- Some 8-bit integers and their two's complements:
  - $\mathbf{+17_{10} = 00010001_2 \quad \Rightarrow \quad 11101110_2 + 1 = 11101111_2 = -17_{10}}$
  - $\mathbf{0_{10} = 00000000_2 \quad \Rightarrow \quad 11111111_2 + 1 = \underline{1}\ 00000000_2 = 0_{10}}$
  - $\mathbf{-128_{10} = 10000000_2 \quad \Rightarrow \quad 01111111_2 + 1 = 10000000_2 = -128_{10}}$

2. Number Systems

- One's-Complement Representation -

- The diminished radix-complement is called 1's-complement, for binary numbers.
- The MSB of a number serves as the sign bit.
- The weight of the MSB is $-2^{n-1}+1$. The other bits have weight $+2^i$.
- For n bits, number $\in$ {$-2^{n-1}+1 ... 2^{n-1}-1$}; n=8, number $\in$ {-127...+127}.
- Two representations of zero (00000000 and 11111111).
- Some 8-bit integers and their one's complements :
  - **$+17_{10} = 00010001_2 \quad \Rightarrow \quad 11101110_2 = -17_{10}$**
  - **$+0_{10} = 00000000_2 \quad \Rightarrow \quad 11111111_2 = -0_{10}$**
  - **$-127_{10} = 10000000_2 \quad \Rightarrow \quad 01111111_2 = +127_{10}$**

2. Number Systems

- Why Two's-Complement? -

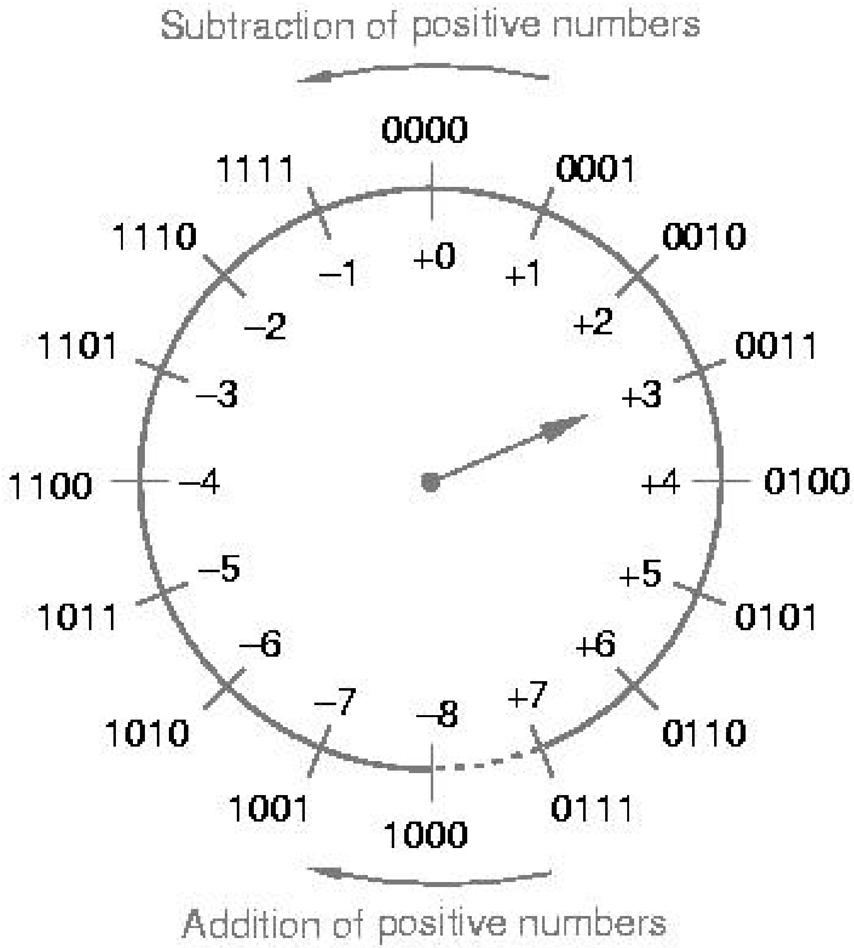
- Hard to build a digital circuit that adds signed-magnitude numbers.
- In 1's-complement, there are two zero representations.
- A 1's-complement adder is more complex that a 2's complement adder.

Decimal	*Two's Complement*	*Ones' Complement*	*Signed Magnitude*	*Excess 2^{m-1}*
−8	1000	—	—	0000
−7	1001	1000	1111	0001
−6	1010	1001	1110	0010
−5	1011	1010	1101	0011
−4	1100	1011	1100	0100
−3	1101	1100	1011	0101
−2	1110	1101	1010	0110
−1	1111	1110	1001	0111
0	0000	1111 or 0000	1000 or 0000	1000
1	0001	0001	0001	1001
2	0010	0010	0010	1010
3	0011	0011	0011	1011
4	0100	0100	0100	1100
5	0101	0101	0101	1101
6	0110	0110	0110	1110
7	0111	0111	0111	1111

2. Number Systems

- Two's-Complement Addition and Subtraction (1) -

- We can add +n, by counting up (clockwise) n times.
- We can subtract +n, by counting down (counterclockwise) n times.
- Valid results if the discontinuity between -8 and +7 is <u>not</u> crossed.
- We can also subtract +n, by counting up (clockwise) 16-n times.



2. Number Systems

- Two's-Complement Addition and Subtraction (2) -

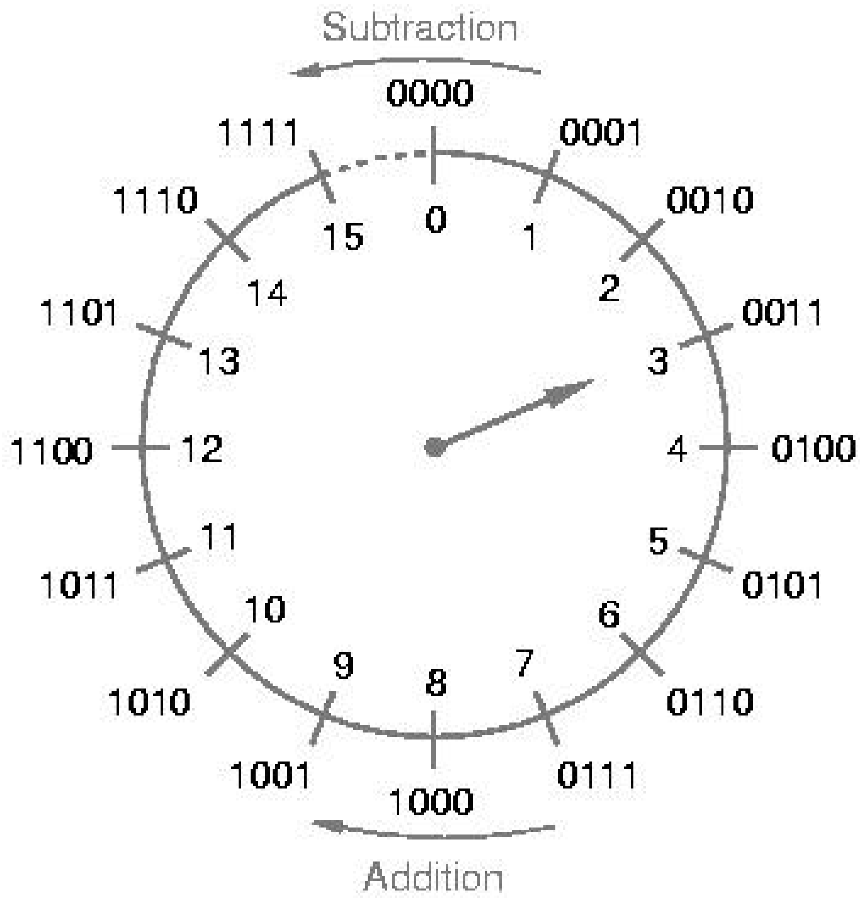
- Overflow occurs when an addition produces a result that exceeds the range of the number system.
- Addition of 2 numbers with different signs never produces overflow.
- An addition overflows if the signs of the addends are the same and the sign of the sum is different form the addends' sign.
- Examples of overflowed additions:

-3	1101		+5	0101	
+ -6	1010		+ +6	0110	
-9	1 0111	= +7	+11	1011	= -5
-8	1000		+7	0111	
+ -8	1000		+ +7	0111	
-16	1 0000	= 0	+ 14	1110	= -2

2. Number Systems

- Two’s-Complement Addition and Subtraction (3) -

- The same adder circuit can be used to handle both 2’s-complement and unsigned numbers.
- However the results must be interpreted differently.
- Valid results if the discontinuity between 15 and 0 is not crossed.



2. Number Systems

- Binary Codes for Decimal Numbers -

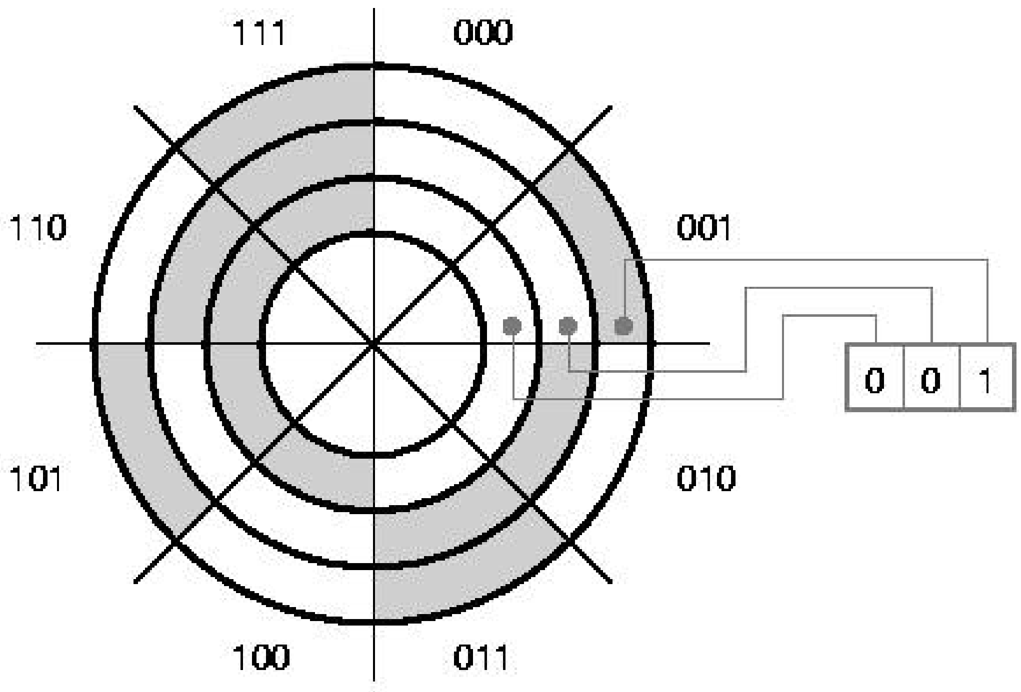
- People prefer to deal with decimal numbers.
- A decimal number is represented by a string of bits.
- A code is a set of bit strings in which different strings represent different numbers (entities).
- A particular combination of bits is a code word.

Decimal digit	*BCD (8421)*	*2421*	*Excess-3*	*Biquinary*	*1-out-of-10*
0	0000	0000	0011	0100001	1000000000
1	0001	0001	0100	0100010	0100000000
2	0010	0010	0101	0100100	0010000000
3	0011	0011	0110	0101000	0001000000
4	0100	0100	0111	0110000	0000100000
5	0101	1011	1000	1000001	0000010000
6	0110	1100	1001	1000010	0000001000
7	0111	1101	1010	1000100	0000000100
8	1000	1110	1011	1001000	0000000010
9	1001	1111	1100	1010000	0000000001
		Unused code words			
	1010	0101	0000	0000000	0000000000
	1011	0110	0001	0000001	0000000011
	1100	0111	0010	0000010	0000000101
	1101	1000	1101	0000011	0000000110
	1110	1001	1110	0000101	0000000111
	1111	1010	1111	. . .	. . .

2. Number Systems

- Gray Code (1) -

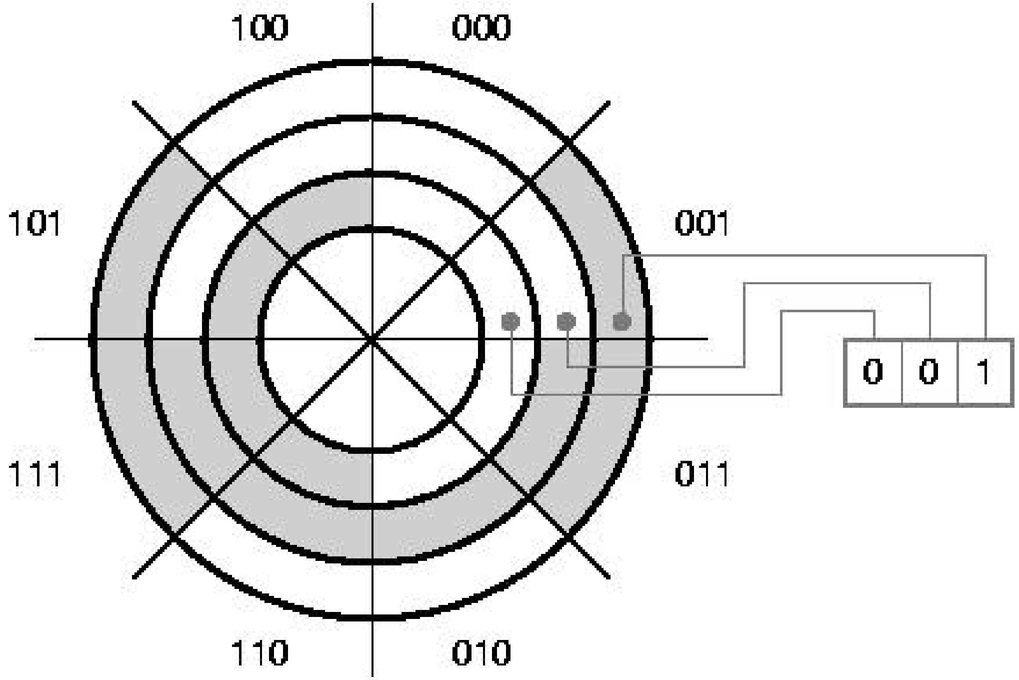
- Input sensor indicates a mechanical position.
- Problems may arise at certain boundaries.
- Boundary between 001 and 010 regions (2 bits change).
- A solution is to devise a digital code in which only one bit changes between successive codes.



2. Number Systems

- Gray Code (2) -

- Gray code solves that problem!
- Only one bit changes at each border.
- Gray codes are also used in Karnaugh maps, since adjacent cells must differ in just one input variable.



2. Number Systems

- Character Codes (1) -

- A string of bits need not represent a number.
- In fact most of the information processed by computers is nonnumeric.
- The most common type of nonnumeric data is text: strings of characters from some character set.
- Each character is represented in the computer by a bit string (code) according to an established convention.
- The most commonly used character code is ASCII (American Standard Code for Information Interchange).
- ASCII represents each character with a 7-bit string, yielding a total of 128 characters.

2. Number Systems

- Character Codes (2) -

		$b_6b_5b_4$ *(column)*							
$b_3b_2b_1b_0$	*Row (hex)*	*000 0*	*001 1*	*010 2*	*011 3*	*100 4*	*101 5*	*110 6*	*111 7*
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	A	LF	SUB	*	:	J	Z	j	z
1011	B	VT	ESC	+	;	K	[	k	{
1100	C	FF	FS	,	<	L	\	l	|
1101	D	CR	GS	-	=	M	]	m	}
1110	E	SO	RS	.	>	N	^	n	~
1111	F	SI	US	/	?	O	_	o	DEL

ASCII (Standard no. X3.4-1968 of the ANSI).