Representation of information

UNIT 1: USE OF MICROCOMPUTING SYSTEMS. SECOND PART

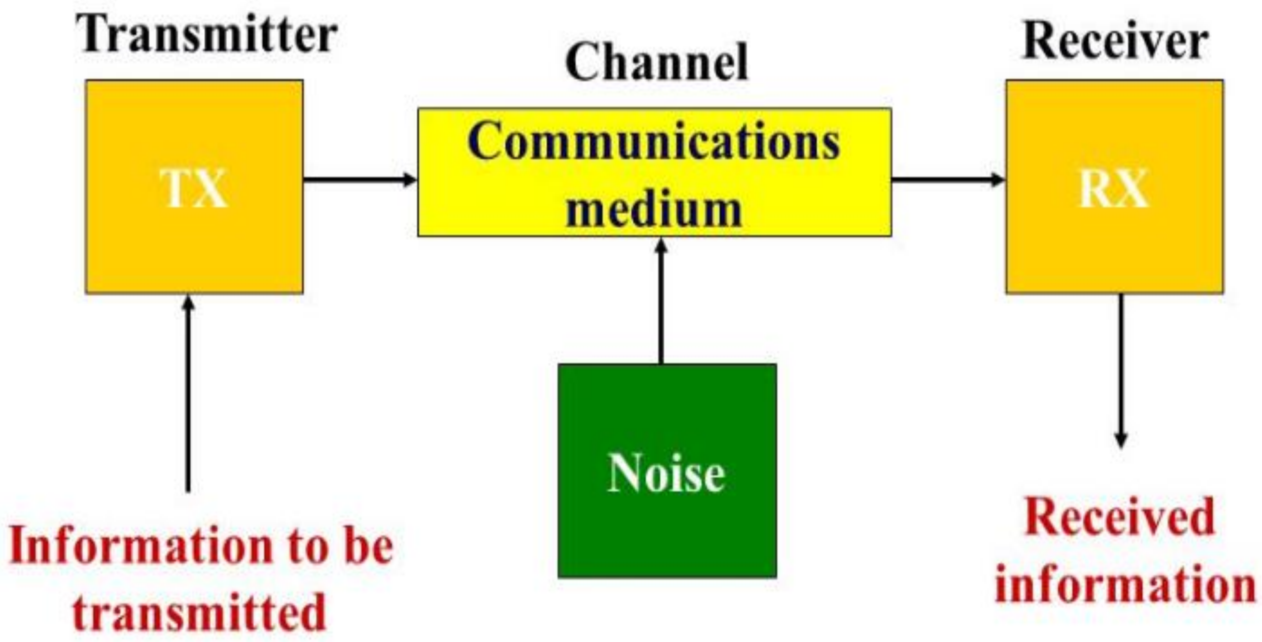
1. Introduction
2. Encoding information
3. Numeral systems
    3.1. Binary system
    3.2. Conversion between numeral systems
    3.3. Octal and hexadecimal numbers
4. Alphanumeric characters
    4.1. ASCII
    4.2. EBDIC
    4.3. Unicode vs UTF-8
5. Units of information

1. Introduction

- Computers basically get input data, transform this data and produce results.
- The purpose of computers is to automate work.
- Computer components understand neither letters nor numbers, but electric current and other physical magnitudes.
- We need codification systems, which convert letters and numbers to something understandable for computers.
- The OS, together with the hardware components, converts the information into electricity and vice versa.

1. Introduction

- Communication system:
  - Transmitter
  - Receiver
  - Channel
- Unidirectional or bidirectional



2. Encoding information

- Information broadcasting
  - Alphanumeric characters (a…z, A..Z, 0….9)
  - Sounds
  - Videos
  - Graphs and pictures

Each type of information works different and has its own way of representing data.

The existence of information requires a common code between transmitter and receiver.

2. Encoding information

- All the information in a computer is stored with two symbols: 1 and 0.
- These two numbers represent two electrical states, which make possible to build reliable internal computer circuits.
- It is necessary a clear correspondence between human symbols (characters, numbers, signs, etc.) and binary symbols (1 and 0),
- This process is called encoding, while decoding is the opposite one.

3. Numeral systems

- A numeral system (also known as system of numeration or number system) is a mathematical notation for representing numbers of a given set, using digits or other symbols, according to some rules.
- The common system uses 10 digits, but we have different bases depending on how many digits are used in a system of numeration to represent numbers
  - **Decimal system (base 10):** 0,1,2,3,4,5,6,7,8,9.
  - **Binary system (base 2):** 0,1
  - **Octal system (base 8):** 0,1,2,3,4,5,6,7
  - **Hexadecimal system (base 16):** 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

3. Numeral systems

- The most commonly used systems today are positional
- Value of numbers is determined by **weight** and **position** of each digit.
- The value of each digit position is the value of its digit, multiplied by a power of the base.
- In general, if b is the base, one writes a number in the numeral system of base b by expressing it in the form $a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \ldots + a_0 b^0$ and writing the enumerated digits $a_n a_{n-1} a_{n-2} \ldots a_0$ in descending order.

3. Numeral systems

- Any numeral system can be summarized by the following relationship:

  $N = \sum b_i q^i$

  where: N is a real positive number
  b is the digit
  q is the base value
  and integer (i) can be positive, negative or zero

- So, in positional base-10 notation: 458 (base 10) = $4*10^2 + 5*10^1 + 8*10^0$
- And, in positional base-2 notation: 1110000 (base 2) = $1*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 0*2^0$

3.1. Binary system

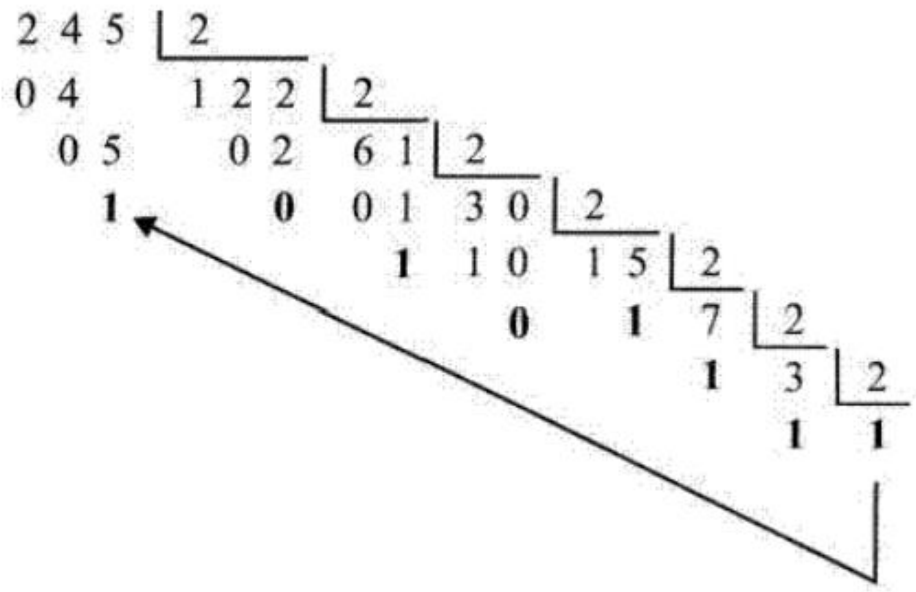
- It is composed of two symbols: 0,1
- Base 2
- Digital systems are based on binary numbers.

0	**0 volts**	**OFF**	**NO**
1	**5 volts**	**ON**	**YES**

- Ex: 1110000 (base 2) = $1*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 0*2^1 + 0*2^0$
- To convert a binary number to decimal, just add the final value of each power of two. According to the example: 1110000 (base 2) = 112 (base 10)

3.1. Binary system

- **To convert a decimal number to binary**:
  - ➢ Divide the input decimal number by 2.
  - ➢ Repeat this process till quotient becomes zero.
  - ➢ Equivalent binary number will be the remainders in above process in reverse order.
- Example: N = 245



- Result = 11110101

3.1. Binary system

- Having n bits, we can represent 2^n values
- Basic powers of 2:
  - $2^1 = 2$… Having 1 bit we represent 2 numbers
  - $2^2 = 4$… Having 2 bits we represent 4 numbers
  - $2^3 = 8$… Having 3 bits we represent 8 numbers
  - $2^4 = 16$… Having 4 bits we represent 16 numbers
- And so on…

3.1. Binary system

- Exercises:

1. Convert from decimal to binary
   - 517
   - 425
   - 315
2. Convert from binary to decimal
   - 1011
   - 10100110
   - 11011

3.2. Conversion between numeral systems

- We have studied how to transform a binary number into decimal and vice versa.
- To convert a binary number to decimal, the equivalent value is equal to the sum of binary digits (d_n) times their power of 2 (2^n). To convert from any numeral system to decimal, we can use the same method, replacing base-2 with the corresponding base.
- To convert a decimal number to binary, we divided and multiplied by 2. To convert from a decimal number to any other numeral system, we can divide and multiply in the same way, but using the corresponding base.
- So…

3.2. Conversion between numeral systems

- We can set a general rule to convert a base-a number to a base-b number:
  - Convert the original number to a decimal number (base 10).
  - Convert the decimal number obtained to the new base number.

  - $N_A \Rightarrow N_{10}$
  - $N_{10} \Rightarrow N_B$

3.2. Conversion between numeral systems

- Exercises:

1. Convert 5270_{10} to base 8
2. Convert 543_{10} to base 8.
3. Convert 3456_8 to binary.
4. Convert 243_{10} to base 5
5. Convert 531_6 to base 10

3.3. Octal and hexadecimal numbers

- Binary system becomes complicated because of the amount of digits used to represent numbers.
- Computers sometimes use base 8 and 16.
- This way, it is easier to convert from and to 2, 8 and 16 bases.
- Octal uses 8 symbols (0..7)
- Hexadecimal uses 16 symbols (0..9, A..F)
- **Let’s take a look on the conversion table in the next slide.**

3.3. Octal and hexadecimal numbers

Dec	Hex	Oct	Bin
0	0	000	0000
1	1	001	0001
2	2	002	0010
3	3	003	0011
4	4	004	0100
5	5	005	0101
6	6	006	0110
7	7	007	0111
8	8	010	1000
9	9	011	1001
10	A	012	1010
11	B	013	1011
12	C	014	1100
13	D	015	1101
14	E	016	1110
15	F	017	1111

3.3. Octal and hexadecimal numbers

- There is a shortcut method for conversions between octal and binary and vice versa.

1. Octal to binary

   - Convert each octal digit to a 3 digit binary number (the octal digits may be treated as decimal for this conversion), according to the previous table.
   - Combine all the resulting binary groups (of 3 digits each) into a single binary number.
     - ✓ $351)_8 = \underline{\mathbf{0}11}\ \underline{101}\ \underline{001})_2$ (we can pad zeros on the left)

3.3. Octal and hexadecimal numbers

2. Binary to octal

- Divide the binary digits into groups of three (starting from the right).
- Convert each group of three binary digits to one octal digit.
  - ✓ $\underline{010}\ \underline{101}\ \underline{100})_2 = 254)_8$ (if there are less than 3 numbers on the left, we pad with zeros)

3.3. Octal and hexadecimal numbers

- To convert from hexadecimal to binary and vice versa, we use the same method as before, but grouping in four.
- Each four numbers are equivalent to an hexadecimal digit.
  - ✓ $15F)_{16}$ = **000**1 0101 1111$)_2$ (we can pad zeros on the left)
  - ✓ 00**11** **1101**$)_2$ = $3D)_{16}$ (if there are less than 4 numbers on the left, we pad with zeros)

3.3. Octal and hexadecimal numbers

- Exercises:

1. Convert $5270)_{10}$ to octal.
2. Convert $3456)_{8}$ to binary.
3. Convert $5270)_{10}$ to hexadecimal
4. Convert $3456)_{16}$ to binary.
5. Convert $454)_{8}$ to hexadecimal.
6. Convert $1C)_{16}$ to octal.

4. Alphanumeric characters

- Computers not only process numbers, but also letters, special symbols or complex types of data such as sound and pictures.
- In computer science, **alphanumeric is a combination of letters and numbers**.
- Each character has a numeric representation, whose binary conversion allows to store data in a “human” format.
- We have different <u>character sets and encodings</u> to represent information.
- ASCII, EBCDIC, UNICODE or UTF-8 are four of the most important ones.

4.1. ASCII

- ASCII is the abbreviated from American Standard Code for Information Interchange
- The original ASCII table is encoded on 7 bits therefore it has 128 characters.
- Nowadays most computers use an **extended ASCII table** (from ISO 8859-1), which is encoded on 8 bits
- Current ASCII can represent 256 characters (including Á, Ä, Œ, é, è and other characters useful for European languages as well as mathematical glyphs and other symbols).

4.1. ASCII (7 bits)

ASCII Code Chart

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

4.1. ASCII (8 bits)

128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	φ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

4.2. EBDIC

- **Extended Binary Coded Decimal Interchange Code (EBCDIC)** is an 8-bit character encoding used mainly on IBM mainframe and IBM midrange computer operating systems.
- When moving information between EBCDIC machines and ASCII machines it is quite often necessary to convert the information.
- It is not a straightforward process, as EBCDIC put lowercase letters before uppercase letters and letters before numbers, exactly the opposite of ASCII.
- EBCDIC is used nowadays in modern mainframes only to provide backwards compatibility.

4.2. EBDIC

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
0000 0	NUL 0	DLE 16	DS 32	48	SP 64	& 80	- 96	112	128	144	160	176	{ 192	} 208	\ 224	0 240
0001 1	SOH 1	DCI 17	SOS 33	49	65	/ 81	97	113	a 129	j 145	161	177	A 193	J 209	225	1 241
0010 2	STX 2	DC2 18	FS 34	SYN 50	66	82	98	114	b 130	k 146	s 162	178	B 194	K 210	S 226	2 242
0011 3	ETX 3	TM 19	35	51	67	83	99	115	c 131	l 147	t 163	179	C 195	L 211	T 227	3 243
0100 4	PF 4	RES 20	BYP 36	PN 52	68	84	100	116	d 132	m 148	u 164	180	D 196	M 212	U 228	4 244
0101 5	HT 5	NL 21	LF 37	RS 53	69	85	101	117	e 133	n 149	v 165	181	E 197	N 213	V 229	5 245
0110 6	LC 6	BS 22	ETB 38	UC 54	70	86	102	118	f 134	o 150	w 166	182	F 198	O 214	W 230	6 246
0111 7	DEL 7	IL 23	ESC 39	EOT 55	71	87	103	119	g 135	p 151	x 167	183	G 199	P 215	X 231	7 247
1000 8	8	CAN 24	40	56	72	88	104	120	h 136	q 152	y 168	184	H 200	Q 216	Y 232	8 248
1001 9	RLF 9	EM 25	41	57	73	89	105	\ 121	i 137	r 153	z 169	185	I 201	R 217	Z 233	9 249
1010 A	SMM 10	CC 26	SM 42	58	cent 74	! 90	| 106	: 122	138	154	170	186	202	218	234	250
1011 B	VT 11	CU1 27	CU2 43	CU3 59	. 75	$ 91	, 107	# 123	139	155	171	187	203	219	235	251
1100 C	FF 12	IFS 28	44	DC4 60	< 76	* 92	% 108	@ 124	140	156	172	188	204	220	236	252
1101 D	CR 13	IGS 29	ENQ 45	NAK 61	(77	) 93	_ 109	’ 125	141	157	173	189	205	221	237	253
1110 E	SO 14	IRS 30	ACK 46	62	+ 78	; 94	> 110	= 126	142	158	174	190	206	222	238	254
1111 F	SI 15	IUS 31	BEL 47	SUB 63	| 79	¬ 95	? 111	” 127	143	159	175	191	207	223	239	255

4.3. Unicode vs UTF-8

- **UTF-8 is an encoding - Unicode is a character set**
- A character set is a list of characters with unique numbers. For example, in the Unicode character set, the number for A is 41.
- Unicode is a standard that defines a superset of all existing characters required to represent practically all known languages.
- UTF-8 encodes each of the 1,112,064 valid code points in Unicode using one to four 8-bit bytes.
- The first 128 characters of Unicode, which correspond one-to-one with ASCII, are encoded using a single octet with the same binary value as ASCII.
- There are other encodings, such UTF-8, UTF-16, UTF-32, UTF-EBCDIC or GB 18030

5. Units of information

- A **bit** is a binary digit, the smallest increment of data on a computer. As we mentioned at the beginning, a bit can hold only one of two values: 0 or 1, corresponding to the electrical values of off or on, respectively.
- Several conventional names are used for collections or groups of bits:
  - Nibble = 4 bits
  - Byte = 8 bits
  - Kilobyte (KB) = 1000 bytes = 10^3 bytes
  - Megabyte (MB) = 1000 KB = 10^6 bytes
  - Gigabyte (GB) = 1000 MB = 10^9 bytes
  - Terabyte (TB) = 1000 GB = 10^{12} bytes
  - Petabyte (PB) = 1000 TB = 10^{15} bytes
  - Exabyte (EB) = 1000 PB = 10^{18} bytes
- <u>BE CAREFUL:</u> We do not use the "1024" equivalences anymore because the prefixes kilo and mega mean 1000 and 1000000 respectively in the in the International System of Units.

5. Units of information

- Computers count by base 2, so we commonly use “binary” units of information.
  - Nibble = 4 bits
  - Byte = 8 bits
  - Kibibyte (KiB) = 1024 bytes
  - Mebibyte (MiB) = 1024 KiB = 1,048,576 bytes
  - Gibibyte (GiB) = 1024 MiB = 1,048,576 bytes
  - Tebibyte (TiB) = 1024 GiB = 1,099,511,627,776 bytes
  - Pebibyte (PiB) = 1024 TiB = 1,125,899,906,842,624 bytes
  - Exbibyte (EiB) = 1024 PiB = 1,152,921,504,606,846,976 bytes

5. Units of information

- Computers count by base 2, so we use "binary" conversions.
  - 1 KiB = 1024 bytes = 2^{10} bytes = 2^{13} bits
  - 1 MiB = 1024 KiB = $2^{10} * 2^{10} = 2^{20}$ bytes = 2^{23} bits
  - 1 GiB = 1024 MiB = $2^{10} * 2^{10} * 2^{10} = 2^{30}$ bytes = 2^{33} bits
  - 1 TiB = 1024 GiB = $2^{10} * 2^{10} * 2^{10} * 2^{10} = 2^{40}$ bytes = 2^{43} bits
  - …..
- <u>Notice that:</u> **B represents bytes** and **b means bit**. For example:
  - 30 MB = 30 megabytes (used in storage devices)
  - 30 MiB = 30 mebibytes (used in storage devices)
  - 30 Mbs = 30 megabits per second (used in networking or devices bandwidth)
  - 30 Mibs = 30 mebibits per second (used in networking or devices bandwidth)

END