UNIT 1

INFORMATION REPRESENTATION (II)

REAL NUMBERS

When we write a real number in a paper we use a decimal comma to distinguish between integer part and fractional part.

74,34

In a computer, the space to represent this kind of numbers is divided in two areas: one for the integer part and one for the fractional part.

There are two ways to denote the size of this areas (fields), and therefore, the comma position:

- Fixed point
- Floating point

FIXED POINT

In this notation, we assign a fixed size to the integer part and the fractional part of the number, in other words, a fixed place to the comma.The advantage is that the process to perform basic operations is the same that integers numbers.

However, this method does not take advantage of the capacity of representation format used. For instance, a computer with 8 bits to represent numbers, could use 5 bits for integer part and 3 for fractional part.

- Maximum number = 01111,111
- Minimum (positive) number = 00000,001.

FLOATING POINT

The range of numbers that can be represented in the fixed point format is insufficient for many applications, particularly for scientific applications which often use very large and very small numbers.

To represent a wide range of numbers using relatively few digits, is well known in the decimal system, the scientific representation or exponential notation.

$0{,}00000025 = 2{,}5 * 10^{-7}$

$N = M * B^E \;||\; N = (M;B;E)$

M: mantissa, B: base, E: exponent

FLOATING POINT

- In decimal (B=10):

$$259{,}75_{(10)} = 0{,}25975 * 10^{3} \text{ or } (0{,}25975;10;3)$$

- In binary (B=2):

$$259{,}75_{(10)} \rightarrow 100000011{,}11_{(2)} \rightarrow 0{,}10000001111 * 2^{9}{}_{(2)} \rightarrow$$

$$0{,}10000001111 * 2^{1001}{}_{(2)} \rightarrow \rightarrow (0{,}10000001111;2;1001)$$

The representable numbers range for a given value of B, is fixed by the number of bits of the exponent E, while the accuracy is determined by the number of bits of M.

NORMALIZATION

The same real value can be represented in infinite ways by exponential notation.

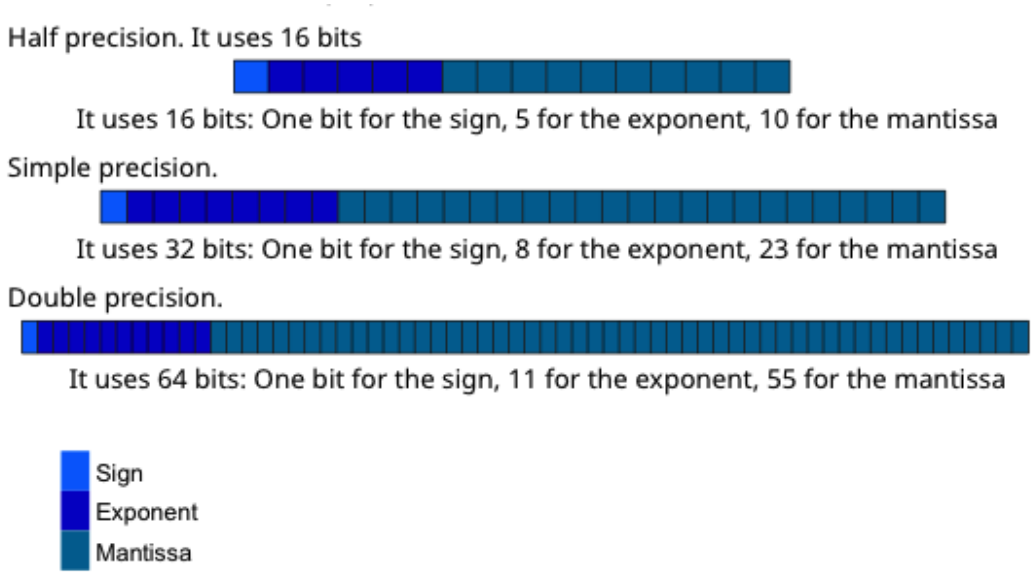
$$2{,}5 = 0{,}25*10^{1} = 0{,}025*10^{2} = 250*10^{-2} \ldots$$

To avoid confusion, we should choose one of these formats as the standard for floating point representation of a real number. The form chosen is called Normalized form and it is one that maintains the highest accuracy in the representation of numbers.

This is achieved when the binary point is located immediately to the left of the first significant digit, so that no space is wasted representing no significant digits.

IEEE754

This format can represent special cases such as infinite values and undefined results, NaN or Not a Number results. It proposes 3 formats:



BOOLEAN ALGEBRA

NOT (¬):

- NOT 0 = 1
- NOT 1 = 0

AND (Y, ^, *):

- 0 AND 0 = 0
- 1 AND 0 = 0
- 0 AND 1 = 0
- 1 AND 1 = 1

OR (O, v):

- 0 AND 0 = 0
- 1 AND 0 = 1
- 0 AND 1 = 1
- 1 AND 1 = 1

XOR:

- 0 XOR 0 = 0
- 1 XOR 0 = 1
- 0 XOR 1 = 1
- 1 XOR 1 = 0

OCTAL

The octal is a numeral system with a system base equal to 8 (symbols 0,1 ,2 ,3 ,4 ,5 ,6 ,7).
Its base is a exact power of binary system base $2^3 = 8$ or, in other words, with three binary digits (with three bits) we can represent all the octal digits.

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

OCTAL

Binary → Octal

The process lies in creating groups of threes bits, starting on the right hand, and replace them for the related octal value.

$$1101011_{(2)} = 1\ \ 101\ \ 011 = 153_{(8)}$$

Octal → Binary

The process is reversed to the previous: it is converted to binary each of the numbers of octal number.

$$7402_{(8)} = 111\ \ 100\ \ 000\ \ 010 = 111100000010_{(2)}$$

HEXADECIMAL

Its system base is 16. As the number of symbols used in the system is greater than 10, 6 characters must be used, in this case from A to F. Thus, the ordered set of symbols is: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

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

HEXADECIMAL

Binary → Hexadecimal

The process is similar to the binary-octal process, except in this case groups are in fours.

$$1101011_{(2)} = 110\ 1011 = 6B_{(16)}$$

Hexadecimal→ Binary

The process is reversed to the previous: it is converted to binary each of the numbers of octal number.

$$7F0A_{(16)} = 0111\ \ 1111\ \ 0000\ \ 1010 = 0111111100001010_{(2)}$$

OTHER CONVERSION

Hexadecimal ⟷ Octal

We convert into binary and group the bits in fours or threes, whichever is the numeral system destination

$$6B_{(16)} = 110\ 1011 = 1101011_{(2)} = 1\ \ 101\ 011 = 153_{(8)}$$

Hexadecimal or Octal ⟷ Decimal

Can be performed following the methods to convert between binary and decimal, but multiplying by power of 8 or 16 or dividing by these numbers and to get the remainders.

However, it is usually more practical to perform directly conversion into binary and then, convert into the system requested.

ALPHANUMERIC REPRESENTATION

A data is numeric if it is possible to perform mathematical operations. In contrast, a data is alphanumeric If you can NOT perform mathematical operations on it.

- How old are you? 45 (Numeric)
- What is your name? "Roberto" (Alphanumeric)

It is usual to think that numeric data are numbers and alphanumeric data are only letters. But this is not correct.

- What is your address? "Avenida de las Palmeras 34"
- What is your mobile number? "662738422"

ASCII

There have been several tables that have always been characterized by the number of bits used to represent each character. One of the best examples in the ASCII table.

The number of bits is 7, which left room for 128 characters (2^7=128)

The problem of this table is its limited space. As you can see, it has room for all the Latin spellings used in Anglo-Saxon languages, but we can not find spellings like the ñ, ç or accented vowels. Therefore, the extended ASCII table of 8 bits (256 characters) was created.

Dec	Oct	Hex	C	Dec	Oct	Hex	C	Dec	Oct	Hex	C	Dec	Oct	Hex	C
0	0	0	^@	32	40	20		64	100	40	@	96	140	60	`
1	1	1	^A	33	41	21	!	65	101	41	A	97	141	61	a
2	2	2	^B	34	42	22	"	66	102	42	B	98	142	62	b
3	3	3	^C	35	43	23	#	67	103	43	C	99	143	63	c
4	4	4	^D	36	44	24	$	68	104	44	D	100	144	64	d
5	5	5	^E	37	45	25	%	69	105	45	E	101	145	65	e
6	6	6	^F	38	46	26	&	70	106	46	F	102	146	66	f
7	7	7	^G	39	47	27	'	71	107	47	G	103	147	67	g
8	10	8	^H	40	50	28	(	72	110	48	H	104	150	68	h
9	11	9	^I	41	51	29	)	73	111	49	I	105	151	69	i
10	12	a	^J	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	b	^K	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	c	^L	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	d	^M	45	55	2d	-	77	115	4d	M	109	155	6d	m
14	16	e	^N	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	f	^O	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	^P	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	^Q	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	^R	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	^S	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	^T	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	^U	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	^V	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	^W	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	^X	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	^Y	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1a	^Z	58	72	3a	:	90	132	5a	Z	122	172	7a	z
27	33	1b	^[	59	73	3b	;	91	133	5b	[	123	173	7b	{
28	34	1c	^\	60	74	3c	<	92	134	5c	\	124	174	7c	|
29	35	1d	^]	61	75	3d	=	93	135	5d	]	125	175	7d	}
30	36	1e	^^	62	76	3e	>	94	136	5e	^	126	176	7e	~
31	37	1f	^_	63	77	3f	?	95	137	5f	_	127	177	7f	

UNICODE

Unicode is a standard that can use more than one byte for each character, allowing a much wider range of representation. The most important elements of Unicode are:

- Code point: A “code point” represents a single character. They are written as U+nnnnnn. Unicode supports more than a million, and just under 100 thousand are currently allocated.
- Glyph: The glyph indicates how a code should be painted on the screen. Different glyph can exist for the same character (for example, different fonts types)
- Codification: There are different ways to represent a character:
  - UTF8: It has a variable length, it uses one to six bytes for each character.
  - UTF16: It’s similar to UTF8, but a minimum of 2 bytes per character are used.
  - UTF32: It has a fixed length. 4 bytes are used for each character.

IMAGES

The usual way of representing images is to project them onto a grid of a given size and save the “color” of each of the squares on that grid. Each of these squares is called a pixel. The size of that grid is the resolution of the image.

For each one of the pixels, it is necessary to store its color. Two fundamental concepts come into play there:

- Color Space: Indicates which colors can be represented. The most common color spaces used in computing are sRGB and AdobeRGB.
- Color model: define how the colors of the color space will be represented, using tuples of numbers. The most common models are RGB, CYMK, HSV and HSL.

There are also vector formats to represent geometric images. These formats do not store pixels but a description of what needs to be drawn. Vector formats can be scaled without problems, since they contain a description of the image and not the pixels that make it up.

VIDEOS

The videos are represented by sequences of images that are slightly modified. Each image is called a frame. A video can be rendered with different number of frames per second, typically from 23.56 to 60.

Videos are usually compressed using different algorithms called Codecs, such as mpeg-2, h.s64 xvid or divx. Since the images between frames normally change little, video codecs can have very high efficiency.

The videos are stored in files with different formats. The format of the video fileis called a container (for example: avi, divx or matroska) They are commonly identified by their extension. A container does not have to contain only video, they habitually contain videos and sounds. Likewise, it may contain information encoded with different codecs.

SOUNDS

A sound is a continuous wave of vibrations. Sampling is used to represent sound: the measurement of the intensity of the wave is taken at regular intervals. From the seintensities, the original signal can be reproduced again.

The audible spectrum of the human ear ranges from 20hz to 20,000hz, depending onthe age of the individual. Therefore, and according to the Nyquist-Shannon theorem, asampling of twice the frequency of the signal is needed, in this case 40Khz. A sampling of 44.1 is usually used Khz in digital formats.

The number of sample bits needed to store the sound depends on the signal-to-noiseratio of the sample. Usually, 16 bits per sample is enough.

SOUNDS

When it comes to digitally representing sounds, three types of formats are used:

- **PCM or RAW formats:** contain the sample as it was received. They occupy alarge size.
- **Compressed formats:** can be lossy (mp3, ogg) or lossless (flac) Lossy formatstry to remove information not perceptible to the human ear, and can be veryefficient. Lossless formats allow you to reproduce the signal as it was sampled.
- **Descriptive formats:** it would be the equivalent of vector formats in images: itcontains the instructions for a synthesizer to interpret a series of notes: it issimilar to a score that contains the notes, the instruments and the times.

UNIT SYSTEM

As discussed above, the bit is the smallest unit of information. Today we do not work at bit level, but in groups of bits (in the previous section we have seen that a character is encoded using 7 or 8 bits).

Because inside the computer everything is in binary code, an easy way to handle groups is to use some value that is a power of 2, being the most basic $2^3 = 8$. A group of 8 bits is named byte.

Today is not usual to use the byte group, but some multiple of it: Kilobyte (kB),Megabyte (MB), Gigabyte (GB), Terabyte (TB)... . In the International System these multiples are powers of 10 (they are based on the decimal system), but in computing powers of 2 are used. However, the trend is to use the International System, although it should be noted that the values are similar but not the same.

UNIT SYSTEM

We use two different kinds of words to differentiate between the system units. When we talk about kilobyte we refer to decimal system and when we talk about kibibytes to the binary system.

Nombre SI	SI	Binario	Nombre Binario
Kilobyte (KB)	10^3 bytes = 1000 bytes	10^{10} bytes = 1024 bytes	Kibibyte (KiB)
Megabyte (MB)	10^6 bytes = 1000 MB	10^{20} bytes = 1024 bytes	Mebibyte (MiB)
Gigabyte (GB)	10^9 bytes = 1000 GB	10^{30} bytes = 1024 bytes	Gibibyte (GiB)
Terabyte (TB)	10^{12} bytes = 1000 TB	10^{40} bytes = 1024 bytes	Tebibyte (TiB)
Petabyte (PB)	10^{15} bytes = 1000 PB	10^{50} bytes = 1024 bytes	Pebibyte (PiB)
Exabyte (EB)	10^{18} bytes = 1000 EB	10^{60} bytes = 1024 bytes	Exbibyte (EiB)
Zetabyte (ZB)	10^{21} bytes = 1000 ZB	10^{70} bytes = 1024 bytes	Zebibyte (ZiB)