



Calcolatori, Internet e il Web



Refresher on Computer Fundamentals and Networking

- History of computers
- Architecture of a computer
- Data representation within a computer
- Computer networks and the Internet
- The World Wide web



Early visions



Charles Babbage
1791-1871

Professor of
Mathematics,
Cambridge University,
1827-1839



Babbage's engines



- *Difference Engine* 1823

- *Analytic Engine* 1833

Blaise Pascal
Pascalina - 1642

- The forerunner of modern **programmable** digital computer

Application

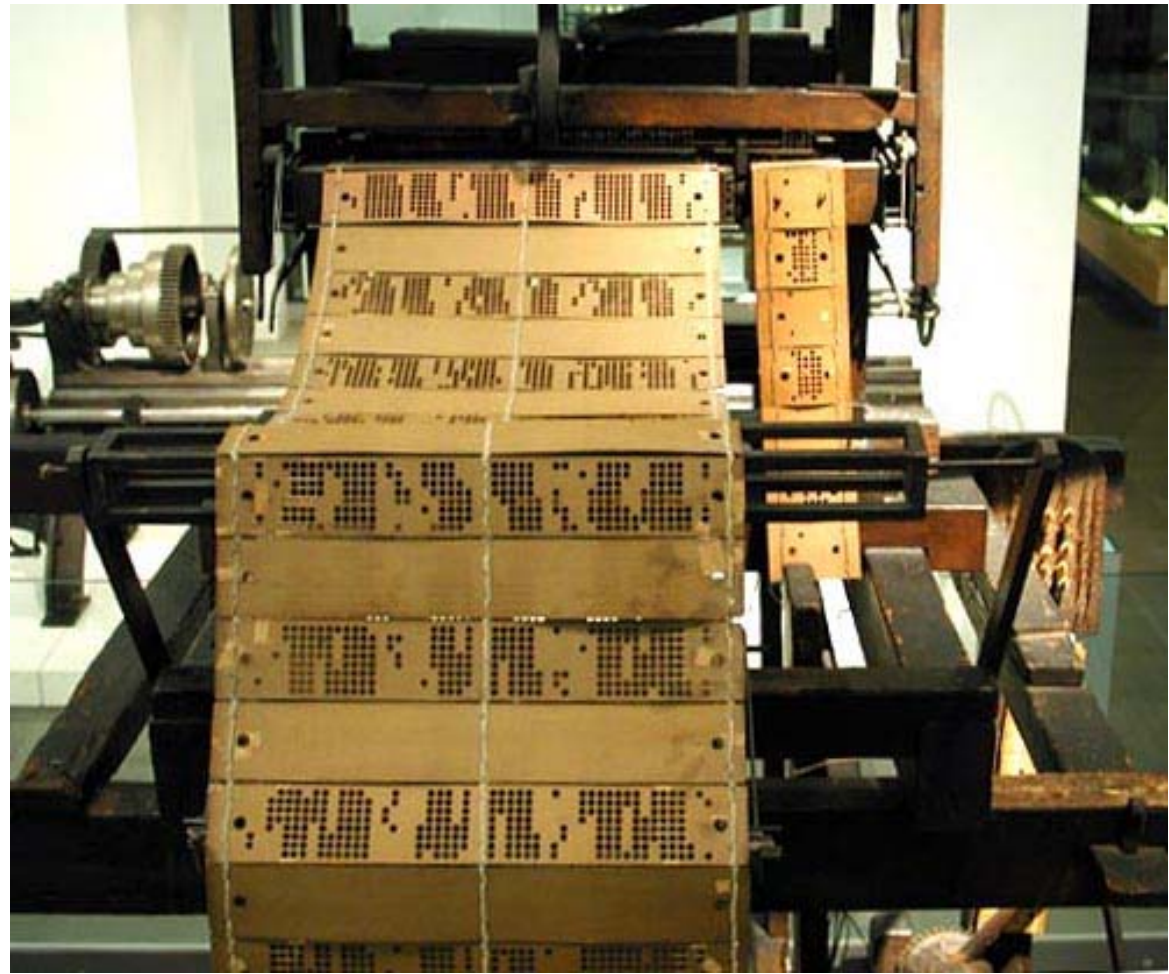
- Mathematical Tables – Astronomy
- Nautical Tables – Navy

Technology

- mechanical gears, Jacquard's loom (1801), simple calculators



Use of punched paper tape



The organ grinder





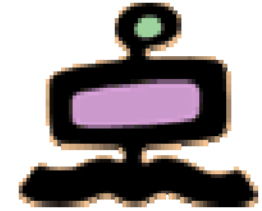
Early experiment



- 1936 – Alan Turing (England) publishes “On computable numbers” (Turing machine)
- 1936-1939 – Konrad Zuse (Germany) builds Z1, Z2 and Z3. Z3 is considered to be the first example of a “modern” computer
- 1937-1939 - John Vincent Atanasoff (USA), with Clifford Berry builds the “Atanasoff-Berry Computer”, known as ABC. In 1973 Atanasoff was credited to be the inventor of the “first electronic digital computer”



Harvard Mark I



- Built in 1944 in IBM Endicott laboratories
 - Howard Aiken – Professor of Physics at Harvard
 - Essentially mechanical but had some electro-magnetically controlled relays and gears
 - Weighed *5 tons* and had *750,000* components
 - A synchronizing clock that beat every *0.015* seconds (66KHz)

Performance:

0.3 seconds for addition

6 seconds for multiplication

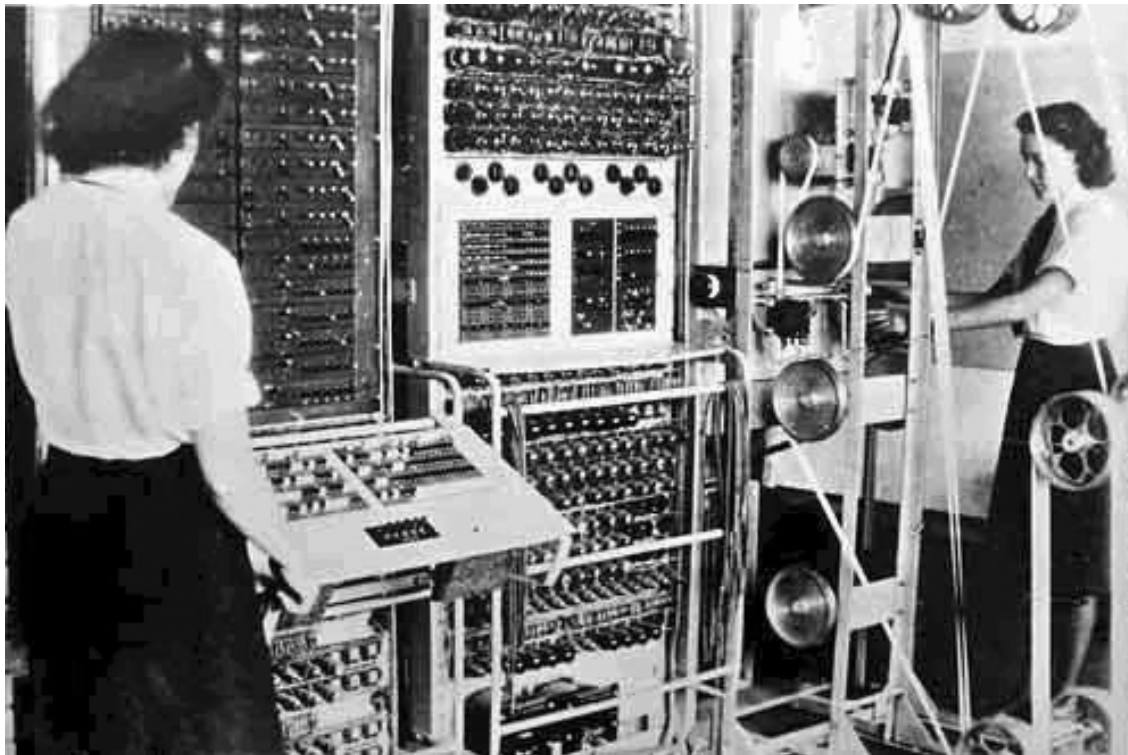
1 minute for a sine calculation

WW-2 Effort

Broke down once a week!



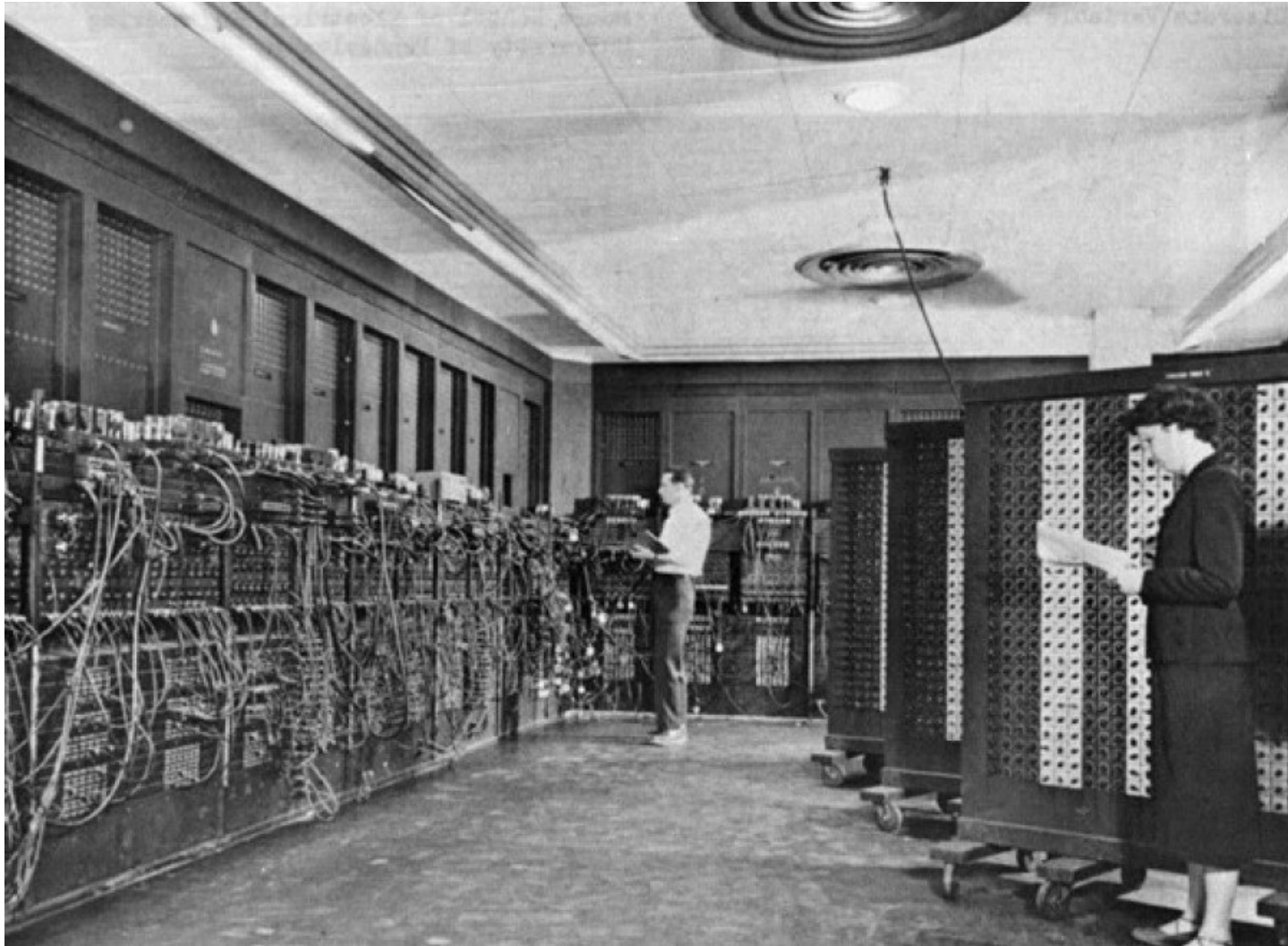
Colossus and Bombe (UK, 1942-1945)



Colossus and Bombe were used in London during the second World War to decipher secret German messages, coded with Enigma. They were built with substantial contributions from Alan Turing

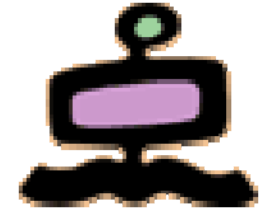


ENIAC - Electronic Numerical Integrator And Computer





ENIAC



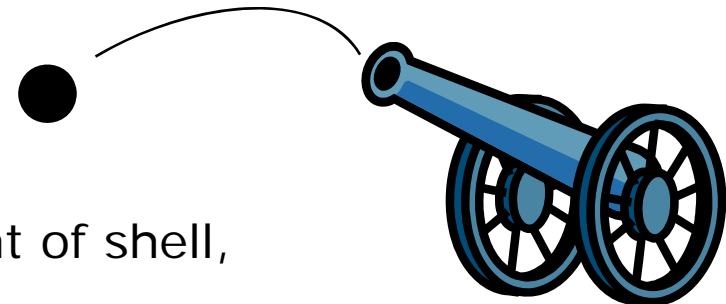
- Inspired by Atanasoff and Berry, Eckert and Mauchly designed and built ENIAC (1943-45) at the University of Pennsylvania
- The first, completely electronic, operational, general-purpose analytical calculator!
 - 30 tons, 72 square meters, 200KW
- Performance
 - Read in 120 cards per minute
 - Addition took 200 μ s, Division 6 ms
 - 1000 times faster than Mark I

WW-2 Effort

Ballistic calculations:

20 hours by human
30 seconds by ENIAC

angle = f (location, tail wind, cross wind,
air density, temperature, weight of shell,
propellant charge, ...)





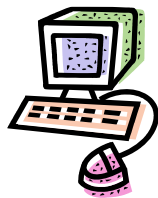
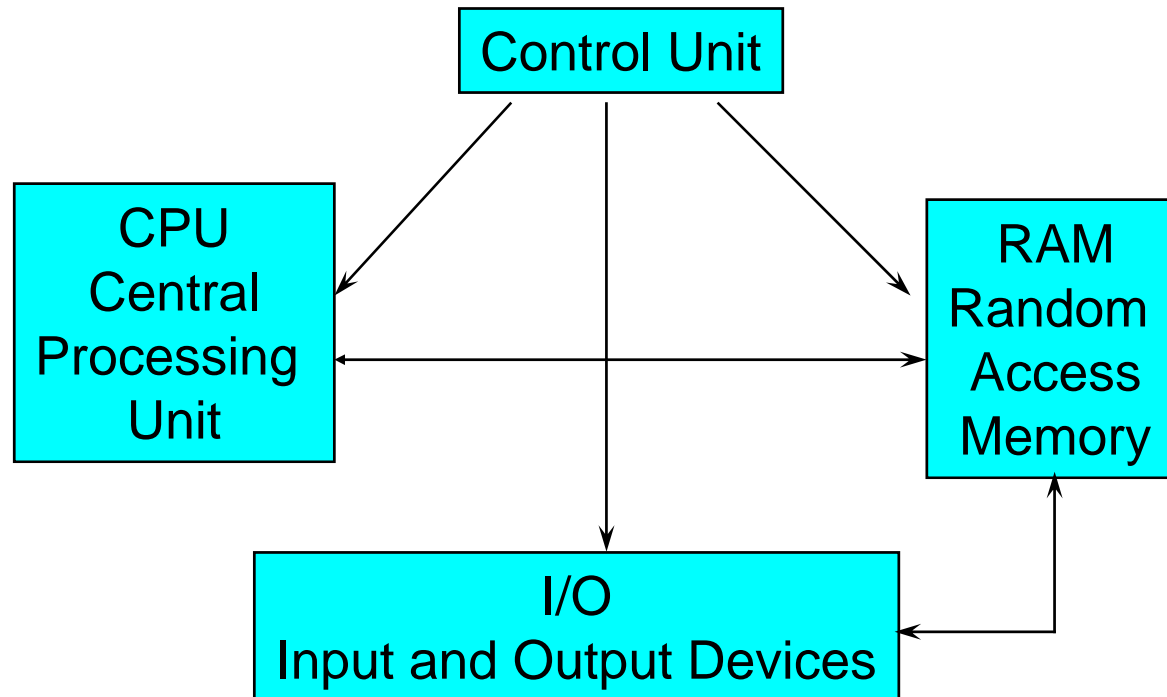
EDVAC - Electronic Discrete Variable Automatic Computer



- ENIAC's programming system was external
 - Sequences of instructions were executed independently of the results of the calculation
 - Human intervention required to take instructions “out of order”
- Eckert, Mauchly, John von Neumann and others designed EDVAC (1944) to solve this problem
 - Solution was the *stored program computer*
 - ⇒ “*program can be manipulated as data*”
- *First Draft of a report on EDVAC* was published in 1945, but had only Von Neumann's signature
- In 1973 the court of Minneapolis attributed the honor of *inventing the computer* to John Atanasoff



Basic components of a computer





Von Neuman architecture



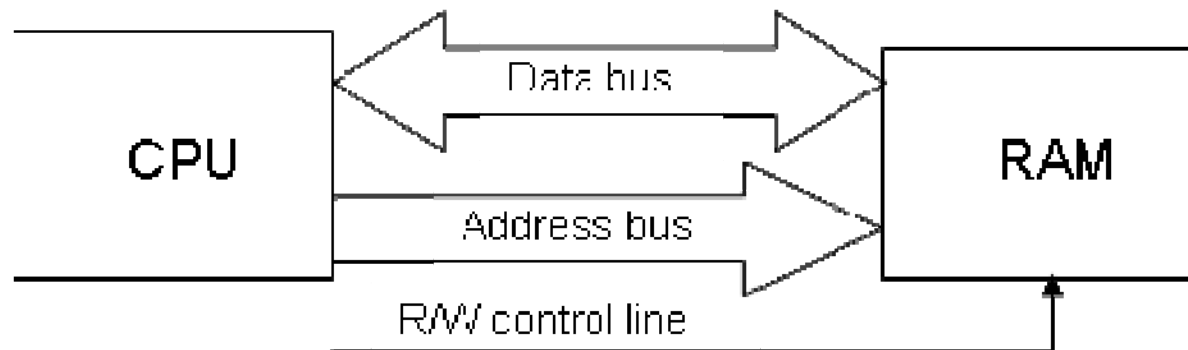
- The RAM contains both the program (machine instructions) and the data
- The basic model is “sequential execution”
 - each instruction is extracted from memory (in sequence) and executed
- Basic execution cycle
 - Fetch instruction (from memory) at location indicated by the Location Counter (LC)
 - Increment Location Counter (to point to the next instruction)
 - Bring instruction to CPU
 - Execute instruction
 - Fetch operand from memory (if needed)
 - Execute operation
 - Store result
 - in “registers” (temporary memory)
 - in memory (RAM)



Random Access Memory

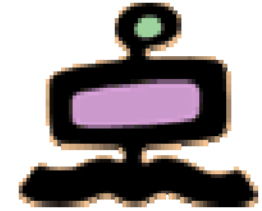


- The RAM is a linear array of “cells”, usually called “words”
- The words are numbered from 0 to N, and this number is the “address” of the word
- In order to read/write a word from/into a memory cell, the CPU has to provide its address on the “address bus”
- A “control line” tells the memory whether it is a read or write operation
- In a read operation the memory will provide the content of that word on the “data bus”
- In a write operation the memory will store in that word the data on the “data bus”





Data within a computer



- The Control Unit, the RAM, the CPU and all the physical components in a computer act on electrical signals and on devices that (basically) can be in only one of two possible states
- From a physical point of view the two states correspond to two voltage levels; from a conceptual point of view the two states are conventionally indicated as “zero” and “one” (0 and 1)
- The consequence is that all the data within a computer (or in order to be processed by a computer) has to be represented with 0s and 1s, i.e. it has to be represented in “binary notation”



Positional notation base 10



Positional notation in base 10

Ten different symbols are needed for the digits (0,1,2,3,4,5,6,7,8,9)

The “weight” of each digit is a power of 10 (the base) and depends on its position in the number

$$10^0=1$$

$$10^1=10$$

$$10^2=100$$

$$10^3=1000$$

$$10^4=10000$$

3

4

7

$$3 \times 10^2 + 4 \times 10^1 + 7 \times 10^0 = 347$$



Positional notation base 8



Positional notation in base 8

Eight different symbols are needed for the digits (0,1,2,3,4,5,6,7)

The “weight” of each digit is a power of 8 (the base) and depends on its position in the number

$$8^0=1$$

$$8^1=8$$

$$8^2=64$$

$$8^3=512$$

$$8^4=4096$$

3

4

7

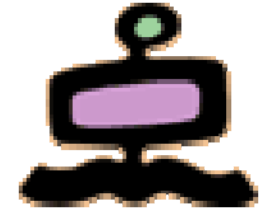
$$3 \times 8^2 + 4 \times 8^1 + 7 \times 8^0$$

$$3 \times 64 + 4 \times 8 + 7 \times 1$$

$$192 + 32 + 7 = 231$$



Positional notation base 2



Positional notation in base 2

Two different symbols are needed for the digits (0,1)

The “weight” of each digit is a power of 2 (the base) and depends on its position in the number

$$2^0=1$$

$$2^1=2$$

$$2^2=4$$

$$2^3=8$$

$$2^4=16$$

$$2^5=32$$

$$2^6=64$$

$$2^7=128$$

$$2^8=256$$

1 **0** **1** **1**

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$$

$$8 + 0 + 2 + 1 = 11$$



Powers of 2



$2^0=1$	$2^9=512$		
$2^1=2$	$2^{10}=1024$	1K	
$2^2=4$	$2^{11}=2048$	2K	
$2^3=8$	$2^{12}=4096$	4K	
$2^4=16$	$2^{13}=8192$	8K	
$2^5=32$	$2^{14}=16384$	16K	
$2^6=64$	$2^{15}=32768$	32K	
$2^7=128$	$2^{16}=65356$	64K	
$2^8=256$		
	$2^{20}=1.048.576$	1024K	1M
	$2^{30}=1.073.741.824$	1024M	1G
	$2^{32}=4.271.406.736$	4096M	4G



Positional notation base 16



Positional notation in base 16

Sixteen different symbols are needed for the digits (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

The “weight” of each digit is a power of 16 (the base) and depends on its position in the number

$$16^0=1$$

$$16^1=16$$

$$16^2=256$$

$$16^3=4096$$

$$16^4=65536$$

3

B

F

$$3 \times 16^2 + B \times 16^1 + F \times 16^0$$

$$3 \times 256 + 11 \times 16 + 15 \times 1$$

$$768 + 176 + 15 = 959$$



Binary and hexadecimal numbers



$$2^0=1$$

$$2^1=2$$

$$2^2=4$$

$$2^3=8$$

$$2^4=16$$

$$2^5=32$$

$$2^6=64$$

$$2^7=128$$

$$2^8=256$$

$$0000=0$$

$$0001=1$$

$$0010=2$$

$$0011=3$$

$$0100=4$$

$$0101=5$$

$$0110=6$$

$$0111=7$$

$$1000=8$$

$$1001=9$$

$$1010=10 \text{ A}$$

$$1011=11 \text{ B}$$

$$1010=12 \text{ C}$$

$$1011=13 \text{ D}$$

$$1110=14 \text{ E}$$

$$1111=15 \text{ F}$$

$$10000=16 \text{ 10}$$

decimal and hexadecimal

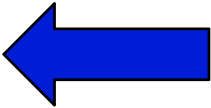
decimal

hexadecimal



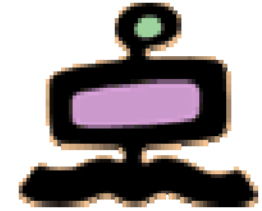
Representation of information



- Numbers 
- Text (characters and ideograms)
- Images
- Video
- Audio



Positional notation base 2



Positional notation in base 2

Two different symbols are needed for the digits (0,1)

The “weight” of each digit is a power of 2 (the base) and depends on its position in the number

$$2^0=1$$

$$2^1=2$$

$$2^2=4$$

$$2^3=8$$

$$2^4=16$$

$$2^5=32$$

$$2^6=64$$

$$2^7=128$$

$$2^8=256$$

1 **0** **1** **1**

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$$

$$8 + 0 + 2 + 1 = 11$$



Representation of information



- Numbers
- Text (characters and ideograms) ←
- Documents
- Images
- Video
- Audio



Representation of characters



- The “natural” way to represent (alphanumeric) characters (and symbols) within a computer is to associate a character with a number, defining a “coding table”
- How many bits are needed to represent the Latin alphabet ?



The ASCII characters



! " # \$ % & ' () * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [\] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~

The 95
printable
ASCII
characters,
numbered
from 32 to
126 (decimal)

33 control
characters



ASCII table (7 bits)



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(72	48	H	104	68	h
9	09	Horizontal tab	41	29)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□



Representation standards



- ASCII 7 bits (late fifties)
 - American Standard Code for Information Interchange
 - 7 bits for 128 characters (Latin alphabet, numbers, punctuation, control characters)
- EBCDIC (early sixties)
 - Extended Binary Code Decimal Interchange Code
 - 8 bits; defined by IBM in early sixties, to take advantage of the byte architecture, still used and supported on many computers
- ASCII 8 bits (ISO 8859-xx) extends original ASCII to 8 bits to include accented letters and non Latin alphabets (e.g. Greek, Russian)
- UNICODE or ISO-10646 (1993)
 - Merged efforts of the Unicode Consortium and ISO
 - UNiversal CODE still evolving
 - It incorporates all(?) the pre-existing representation standards



ISO-8859-xx (ASCII 8-bits)



- Developed by ISO (International Organization for Standardization)
- There are 16 different tables coding characters with 8 bit
- Each table includes ASCII (7 bits) in the lower part and other characters in the upper part for a total of 191 characters and 32 control codes
- It is also known as ISO-Latin-xx (includes all the characters of the “Latin alphabet”)



ISO-8859-xx code pages



- 8859-1 Latin-1 Western European languages
- 8859-2 Latin-2 Central European languages
- 8859-3 Latin-3 South European languages
- 8859-4 Latin-4 North European languages
- 8859-5 Latin/Cyrillic Slavic languages
- 8859-6 Latin/Arabic Arabic language
- 8859-7 Latin/Greek modern Greek alphabet
- 8859-8 Latin/Hebrew modern Hebrew alphabet
- 8859-9 Latin-5 Turkish language (similar to 8859-1)
- 8859-10 Latin-6 Nordic languages (rearrangement of Latin-4)
- 8859-11 Latin/Thai Thai language
- 8859-12 Latin/Devanagari Devanagari language (abandoned in 1997)
- 8859-13 Latin-7 Baltic Rim languages
- 8859-14 Latin-8 Celtic languages
- 8859-15 Latin-9 Revision of 8859-1
- 8859-16 Latin-10 South-Eastern European languages



Representation standards



- ASCII (late fifties)
 - American Standard Code for Information Interchange
 - 7 bits for 128 characters (Latin alphabet, numbers, punctuation, control characters)
- EBCDIC (early sixties)
 - Extended Binary Code Decimal Interchange Code
 - 8 bits; defined by IBM in early sixties, still used and supported on many computers
- ISO 8859-1 extends ASCII to 8 bits (accented letters, non Latin characters)
- UNICODE or ISO-10646 (1993)
 - Merged efforts of the Unicode Consortium and ISO
 - UNiversal CODE still evolving
 - It incorporates all(?) the pre-existing representation standards



UNICODE



- In Unicode, the word “character” refers to the notion of the abstract form of a “letter”, in a very broad sense
 - a letter of an alphabet
 - a mark on a page
 - a symbol (in a language)
- A “glyph” is a particular rendition of a character (or composite character). The same Unicode character can be rendered by many glyphs
 - Character “a” in 12-point Helvetica, or
 - Character “a” in 16-point Times
- In Unicode each “character” has a name and a numeric value (called “code point”), indicated by U+hex value.
For example, the letter “G” has:
 - Unicode name: “LATIN CAPITAL LETTER G”
 - Unicode value: U+0047 (see ASCII codes)



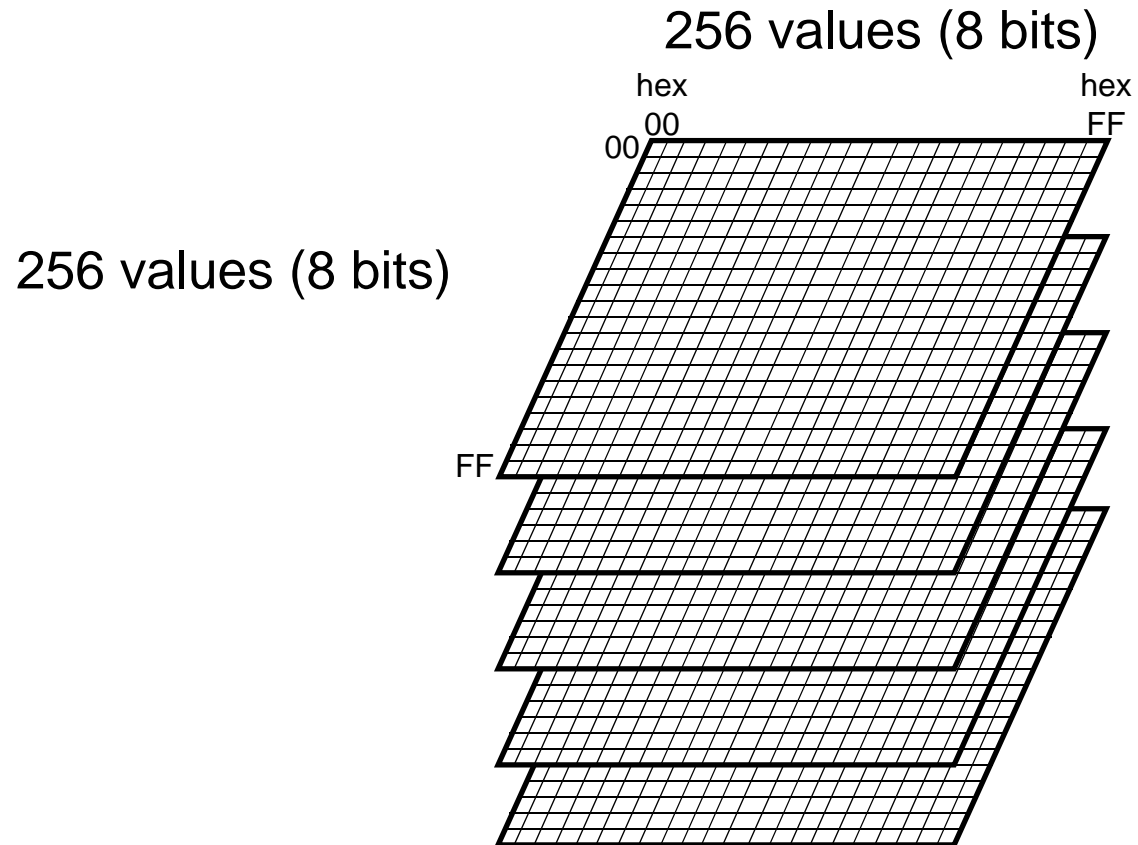
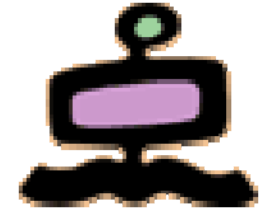
Unicode representation



- The Unicode standard has specified (and assigned values to) about 96.000 characters
- Representing Unicode characters (code points)
 - 32 bits in ISO-10646
 - 21 bits in the Unicode Consortium
- In the 21 bit address space, there are 32 “planes” of 64K characters each (256X256)
- Only 6 planes defined as of today, of which only 4 are actually “filled”
- Plane 0, the Basic Multilingual Plane, contains most of the characters used (as of today) by most of the languages used in the Web



The planes of Unicode





Unicode planes



Plane 0	Basic Multilingual Plane	U+0000 to U+FFFF	modern languages and special characters. Includes a large number of Chinese, Japanese and Korean (CJK) characters.
Plane 1	Supplementary Multilingual Plane	U+10000 to U+1FFFF	historic scripts and musical and mathematical symbols
Plane 2	Supplementary Ideographic Plane	U+20000 to U+2FFFF	rare Chinese characters
Plane 14	Supplementary Special-purpose Plane	U+E0000 to U+EFFFF	non-recommended language tag and variation selection characters
Plane 15	Supplementary Private Use Area-A	U+F0000 to U+FFFFF	private use (no character is specified)
Plane 16	Supplementary Private Use Area-B	U+100000 to U+10FFFF	private use (no character is specified)



Unicode charts



Language characters	Kannada
Basic Latin	Khmer Symbols
Latin-1 Supplement	Khmer
Latin Extended-A	Lao
Latin Extended-B	Limbu
Latin Extended Additional	Linear B Ideograms
	Linear B Syllabary
Language specific characters	Malayalam
Alphabetic Presentation Forms	Mongolian
Arabic Presentation Forms-A	Myanmar
Arabic Presentation Forms-B	Ogham
Arabic	Old Italic
Armenian	Oriya
Bengali	Osmanya
Buhid	Runic
Cherokee	Shavian
Cypriot Syllabary	Sinhala
Cyrillic Supplement	Syriac
Cyrillic	Tagalog
Deseret	Tagbanwa
Devanagari	Tai Le
Ethiopic	Tamil
Georgian	Telugu
Gothic	Thaana
Greek and Coptic	Thai
Greek Extended	Tibetan
Gujarati	Ugaritic
Gurmukhi	Unified Canadian Aboriginal Syllabics
Hanunoo	Yi Radicals
Hebrew	Yi Syllables

Language specific characters (Chinese, Japanese, Korean)	Numbers
Bopomofo Extended	Aegean Numbers
Bopomofo	Number Forms
CJK Compatibility Forms	
CJK Compatibility Ideographs Supplement	Other symbols
CJK Compatibility Ideographs	Braille Patterns
CJK Compatibility	Byzantine Musical Symbols
CJK Radicals Supplement	Combining Diacritical Marks for Symbols
	Control Pictures
CJK Symbols and Punctuation	Currency Symbols
CJK Unified Ideographs Extension A	Enclosed Alphanumerics
CJK Unified Ideographs Extension B	Letterlike Symbols
CJK Unified Ideographs	Miscellaneous Technical
Enclosed CJK Letters and Months	Musical Symbols
Hangul Compatibility Jamo	Optical Character Recognition
Hangul Jamo	Tai Xuan Jing Symbols
Hangul Syllables	Yijing Hexagram Symbols
Hiragana	
Ideographic Description Characters	
Kanbun	Character modifiers and punctuation
Kangxi Radicals	Combining Diacritical Marks
Katakana Phonetic Extensions	IPA Extensions
Katakana	Phonetic Extensions
	Spacing Modifier Letters
Graphic symbols	Combining Half Marks
Arrows	General Punctuation
Block Elements	Superscripts and Subscripts
Box Drawing	
Geometric Shapes	Miscellaneous
Misc. Symbols and Arrows	Halfwidth and Fullwidth Forms
Supplemental Arrows-A	High Private Use Surrogates
Supplemental Arrows-B	High Surrogates
	Low Surrogates
Pictorial symbols	Private Use Area
Dingbats	Small Form Variants
Miscellaneous Symbols	Specials
	Supplementary Private Use Area-A
Mathematical symbols	Supplementary Private Use Area-B
Math. Alphanumeric Symbols	Tags
Math. Operators	Variation Selectors Supplement
Miscellaneous Math. Symbols-A	Variation Selectors
Miscellaneous Math. Symbols-B	
Supplemental Math. Operators	



Unicode encoding



- UTF-32 (fixed length, four bytes)
 - UTF stands for “UCS Transformation Format” (UCS stands for “Unicode Character Set”)
 - UTF-32BE and UTF-32LE have a “byte order mark” to indicate “endianness”
- UTF-16 (variable length, two bytes or four bytes)
 - All characters in the BMP represented by two bytes
 - The 21 bits of the characters outside of the BMP are divided in two parts of 11 and 10 bits; to each part is added an offset to bring it in the “surrogate zone” of the BMP (low surrogate at D800 and high surrogate at DC800)
 - in other words, they are represented as two characters in the BMP
 - UTF-16BE and UTF-16LE to indicate “endianness”
- UTF-8 (variable length, one, two, three or four bytes)
 - Characters in the 7-bit ASCII represented by one byte
 - Variable length encoding (2, 3 or 4 bytes) for all other characters



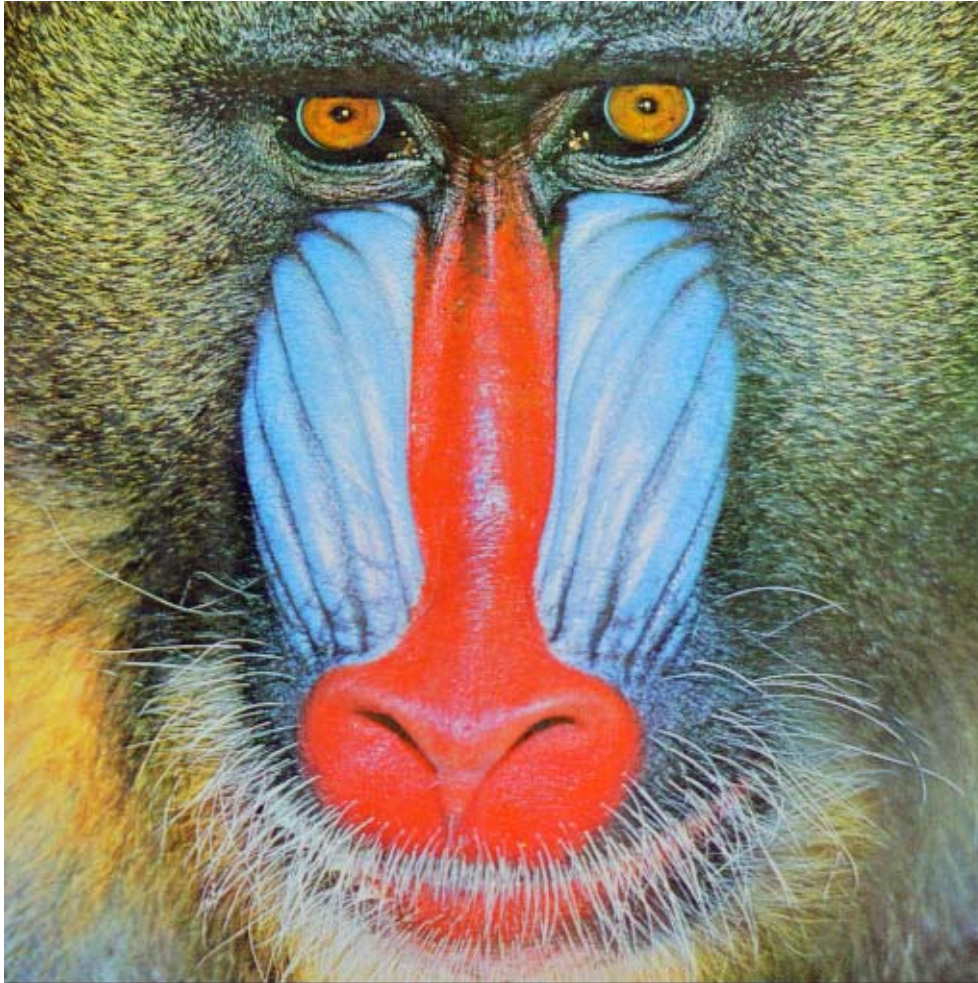
Representation of information



- Numbers
- Text (characters and ideograms)
- Images ←
- Video
- Audio



Welcome



Welcome to image
representation and
compression





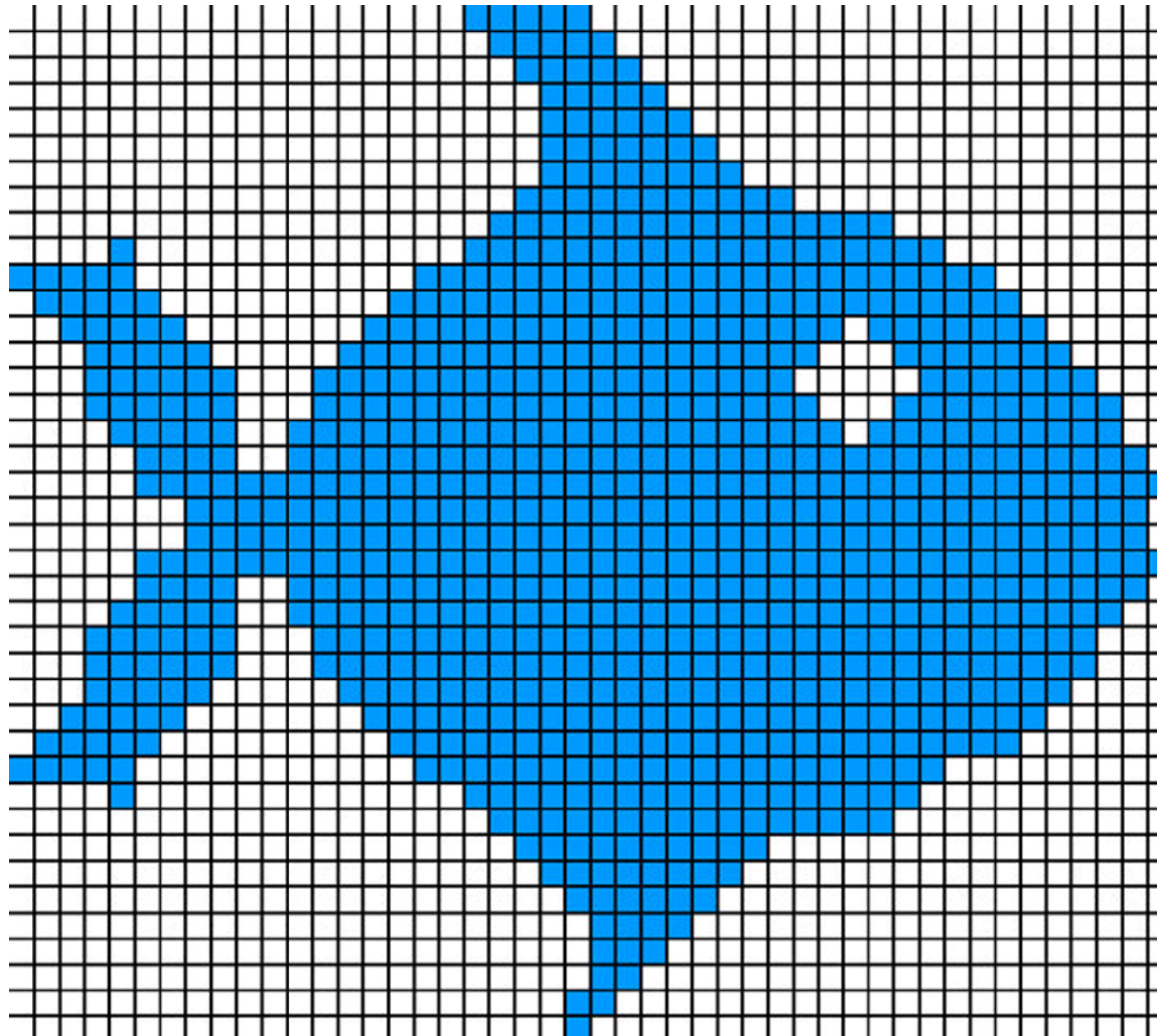
Representation of images



- Vector formats (geometric description)
 - Postscript
 - PDF
 - SVG (Scalable Vector Graphics)
 - SWF (ShockWave Flash)
 - from FutureWave Software to Macromedia to Adobe
 - vector-based images, plus audio, video and interactivity
 - can be played by Adobe Flash Player (browser plug-in or stand-alone)
- Raster formats (array of “picture elements”, called “pixels”)



Picture elements (pixels)

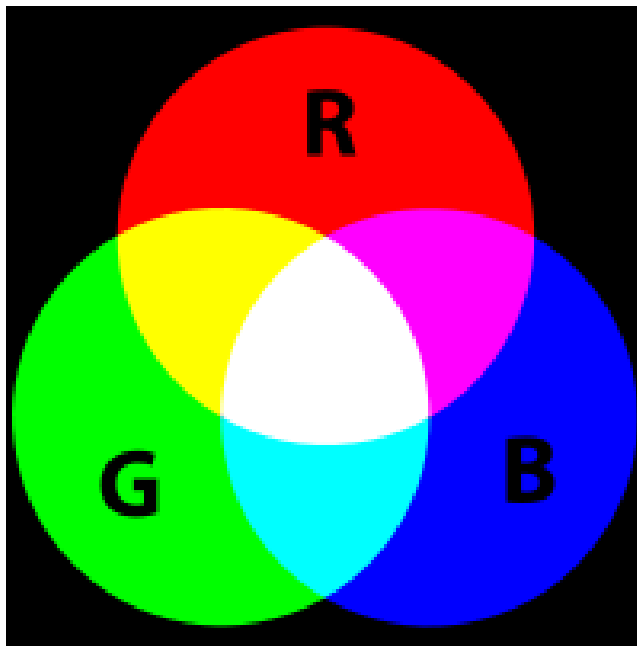


A pixel must be small enough so that its color can be considered uniform for the whole pixel.

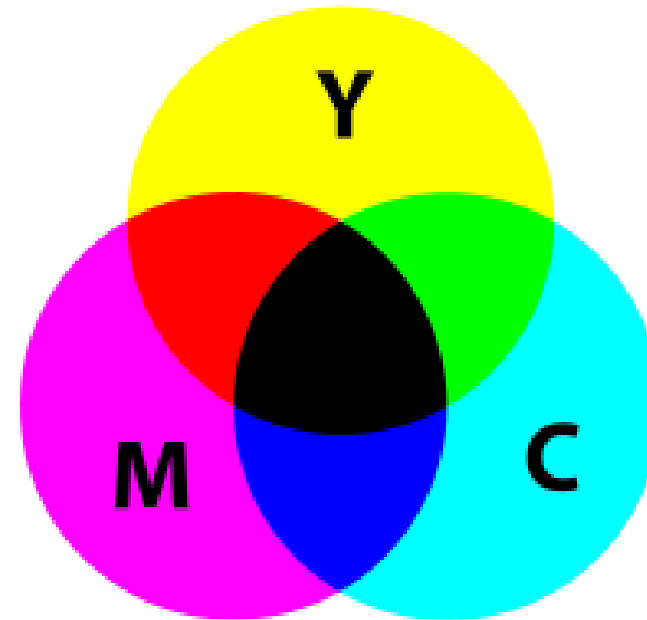
Inside the computer, a pixel is represented with a number representing its color.



RGB and CMY color components



Additive color mixing



Subtractive color mixing



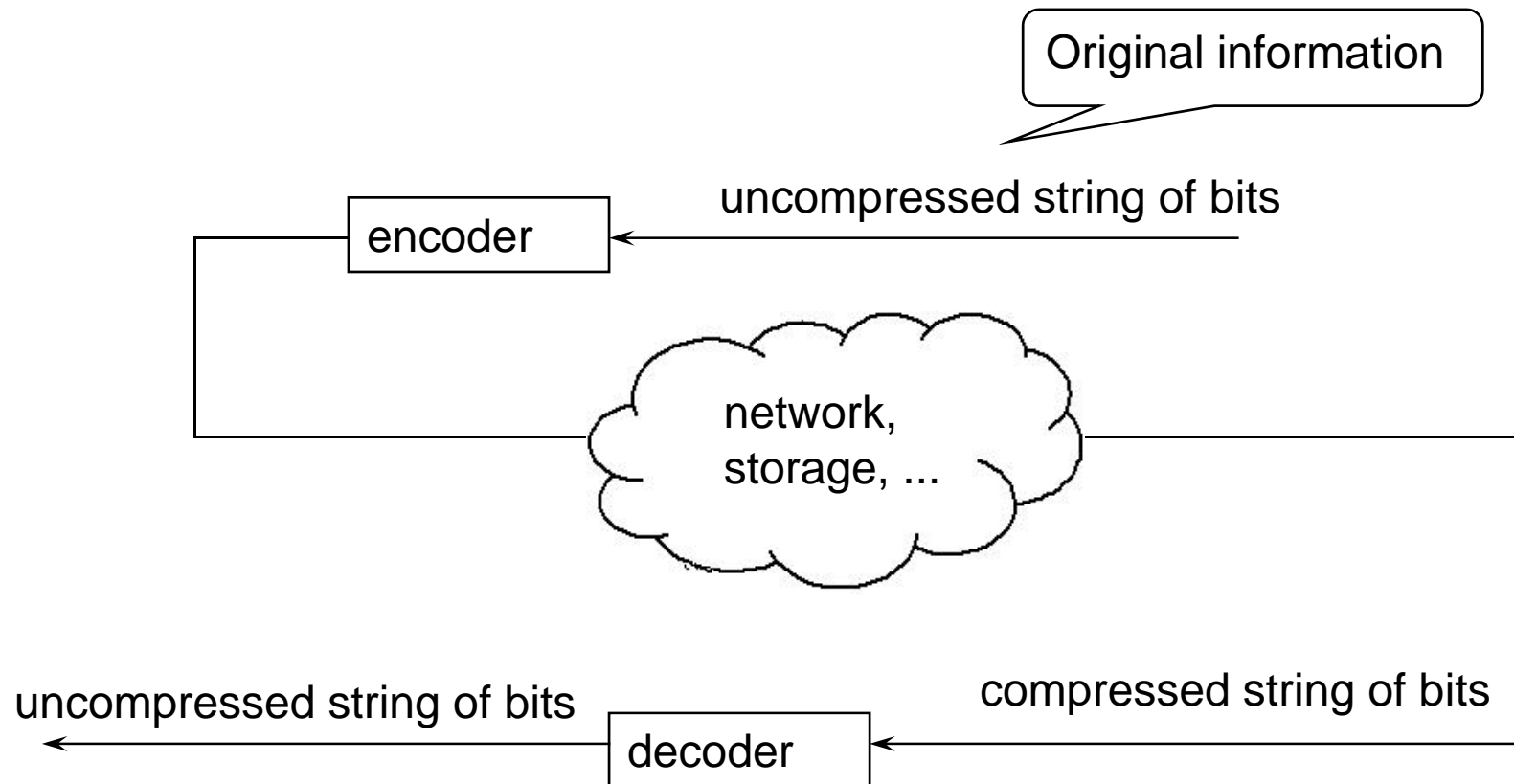
Raster format



- In raster format an image (picture) is represented by a matrix of “pixels”
- Colors are represented by three numbers, one for each “color component”
- The quality of a picture is determined by:
 - The number of rows and columns in the matrix
 - Very often it is expressed as “dots per inch” (dpi)
 - 200-4800 dpi (most common ranges)
 - The number of bits representing one pixel (called depth)
 - 1 bit for black and white
 - 8-16 bits for gray scale (most common ranges)
 - 24-48 bits for color images (most common ranges)
- Big file sizes for (uncompressed color) pictures
 - For example, one colour page scanned at 600 dpi is about 100 MB
 - Compression is needed



Compression process





Data compression



- The idea of data compression is that when the data is not needed for processing (e.g. when in transit over a network or when stored on secondary storage), then it can be represented in a more compact form (with less bits), provided that the “original” information can be brought back when needed.
- Compression can be “lossless” or “lossy”
- In **lossless compression** (usually used for text or numbers) the uncompressed information is “exactly” equal (bit for bit) to the original information
- In **lossy compression** (usually used for images, video and sound) the uncompressed information has less information than the original, but (usually) the human eye and ear do not perceive the difference
- Compression is not usually noticed (which means that it is well done) but it is used in a number of applications, such as transmission of fax, downloading of web pages, transmission of data over a network, storage of data onto secondary storage, zip files, tar files, pictures, videos, music



Lossless compression techniques



- There are two main classes of lossless text compression methods
 - Symbol-wise encoding
 - Dictionary encoding
- Symbolwise encoding
 - The basic idea is that the most frequent symbols can be coded with less bits (short codewords) than the less frequent symbols (long codewords)
 - Symbol coders work by taking one symbol at the time from the input string, and coding it with a codeword whose length depends on the frequency (probability) of the symbol in the given alphabet
 - The most common symbol encoders are:
 - Huffman coding
 - Arithmetic coding
- Dictionary coding
 - The basic idea is to replace a sequence of symbols in the input string with an “index” in a dictionary (list) of “phrases”
 - Lempel-Ziv 77
 - Lempel-Ziv 78
 - Lempel-Ziv-Welch



“Symbolwise” Morse code (about 1840)



1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.

A	• ■■	U	• • ■■
B	■■■ • • •	V	• • • ■■
C	■■■ • ■■ •	W	• ■■ ■■
D	■■■ • •	X	■■■ • • ■■
E	•	Y	■■■ • ■■ ■■
F	• • ■■ •	Z	■■■ ■■ • •
G	■■■ ■■ •		
H	• • • •		
I	• •		
J	• ■■ ■■ ■■		
K	■■■ • ■■	1	• ■■ ■■ ■■ ■■
L	• ■■ • •	2	• • ■■ ■■ ■■
M	■■■ ■■	3	• • • ■■ ■■
N	■■■ •	4	• • • • ■■
O	■■■ ■■ ■■	5	• • • • •
P	• ■■ ■■ •	6	■■■ • • • •
Q	■■■ ■■ • ■■	7	■■■ ■■ • • •
R	• ■■ •	8	■■■ ■■ ■■ • •
S	• • •	9	■■■ ■■ ■■ ■■ •
T	■■■	0	■■■ ■■ ■■ ■■ ■■



Frequency distribution of the English letters



A	0.0856	0.1304	E	.
B	0.0139	0.1045	T	-
C	0.0279	0.0856	A	.-
D	0.0378	0.0797	O	---
E	0.1304	0.0707	N	-.
F	0.0289	0.0677	R	.-.
G	0.0199	0.0627	I	..
H	0.0528	0.0607	S	...
I	0.0627	0.0528	H
J	0.0013	0.0378	D	-..
K	0.0042	0.0339	L	.-..
L	0.0339	0.0289	F	..-.
M	0.0249	0.0279	C	-.-.
N	0.0707	0.0249	M	--
O	0.0797	0.0249	U	..-
P	0.0199	0.0199	G	--.
Q	0.0012	0.0199	Y	-.-.
R	0.0677	0.0199	P	.-.
S	0.0607	0.0149	W	.-.
T	0.1045	0.0139	B	-....
U	0.0249	0.0092	V	...-
V	0.0092	0.0042	K	.-.
W	0.0149	0.0017	X	-....
X	0.0017	0.0013	J
Y	0.0199	0.0012	Q	---.
Z	0.0008	0.0008	Z	---.



Lempel Ziv 77 coding (1/4)



Repeat occurrences of character sequences?
— replace them with a pointer back to the first occurrence

to • be • or • not • to • be , • that • is • ...

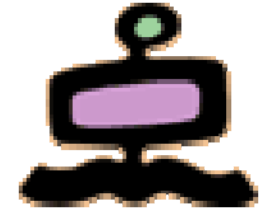
Pointer is <position,length>

to • be • or • not • <1,5> , • that • is • ...

In both are <256, pointer fits in 2 bytes



Lempel Ziv 77 coding (2/4)



The more
regular the
input, the
better the
compression

aaaaaaaaaaaaaaaaarrrrrrrr
rrrrrgggggggggghhhh

byte 1	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a			byte 16	
	a	r	r	r	r	r	r	r	r	r	r	r	r	r	g	g			
	g	g	g	g	g	g	g	g	g	h	h	h	h	h					

a	116	r	18	12	g	31	10	h	42	4										



Testbed for text compression methods

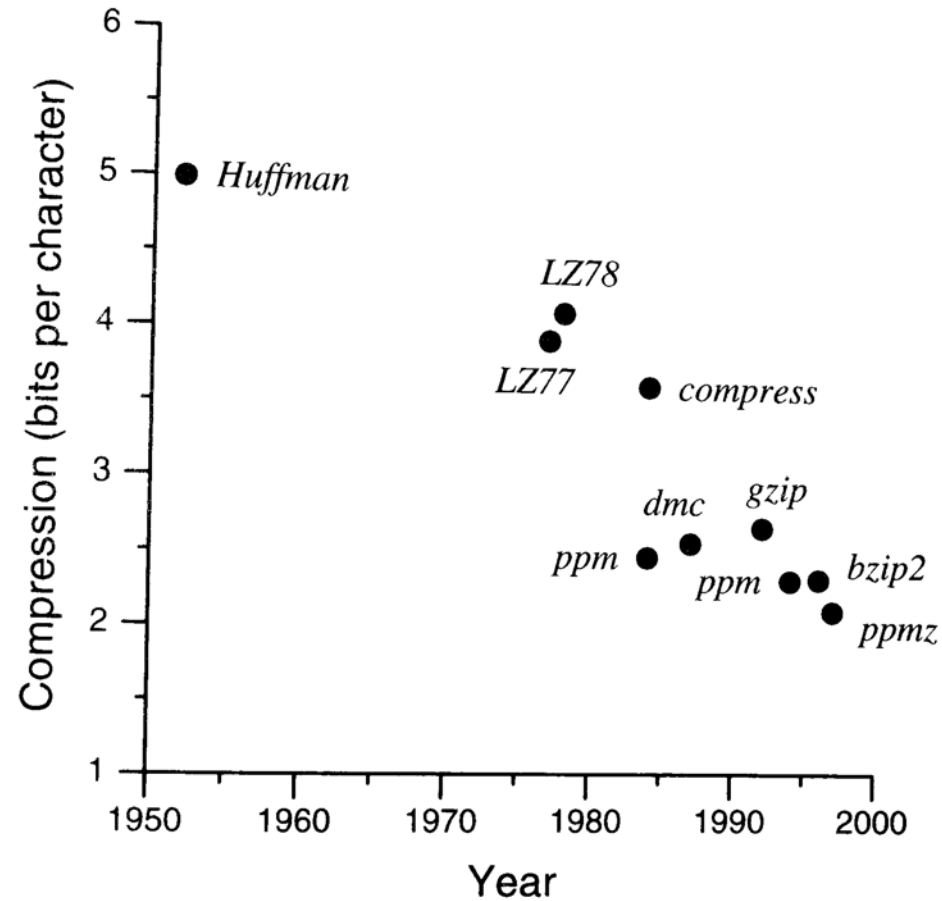


Table 2.7 The Canterbury corpus of text used to evaluate compression methods.

File	Bytes	Content
<i>text</i>	152,089	The text of Lewis Carroll's <i>Alice's Adventures in Wonderland</i>
<i>fax</i>	513,216	A fax bitmap image (CCITT test document 5)
<i>Csrc</i>	11,150	C source code
<i>Excl</i>	1,029,744	Excel spreadsheet
<i>SPRC</i>	38,240	Executable object code for Sun SPARC architecture
<i>tech</i>	426,754	Technical writing (workshop proceedings)
<i>poem</i>	481,861	<i>Paradise Lost</i> by John Milton
<i>HTML</i>	24,603	Hypertext Markup Language source
<i>lisp</i>	3,721	A Lisp program
<i>man</i>	4,227	A Unix manual page in the <i>roff</i> format
<i>play</i>	125,179	Shakespeare's play, <i>As You Like It</i>



Comparison of compression methods





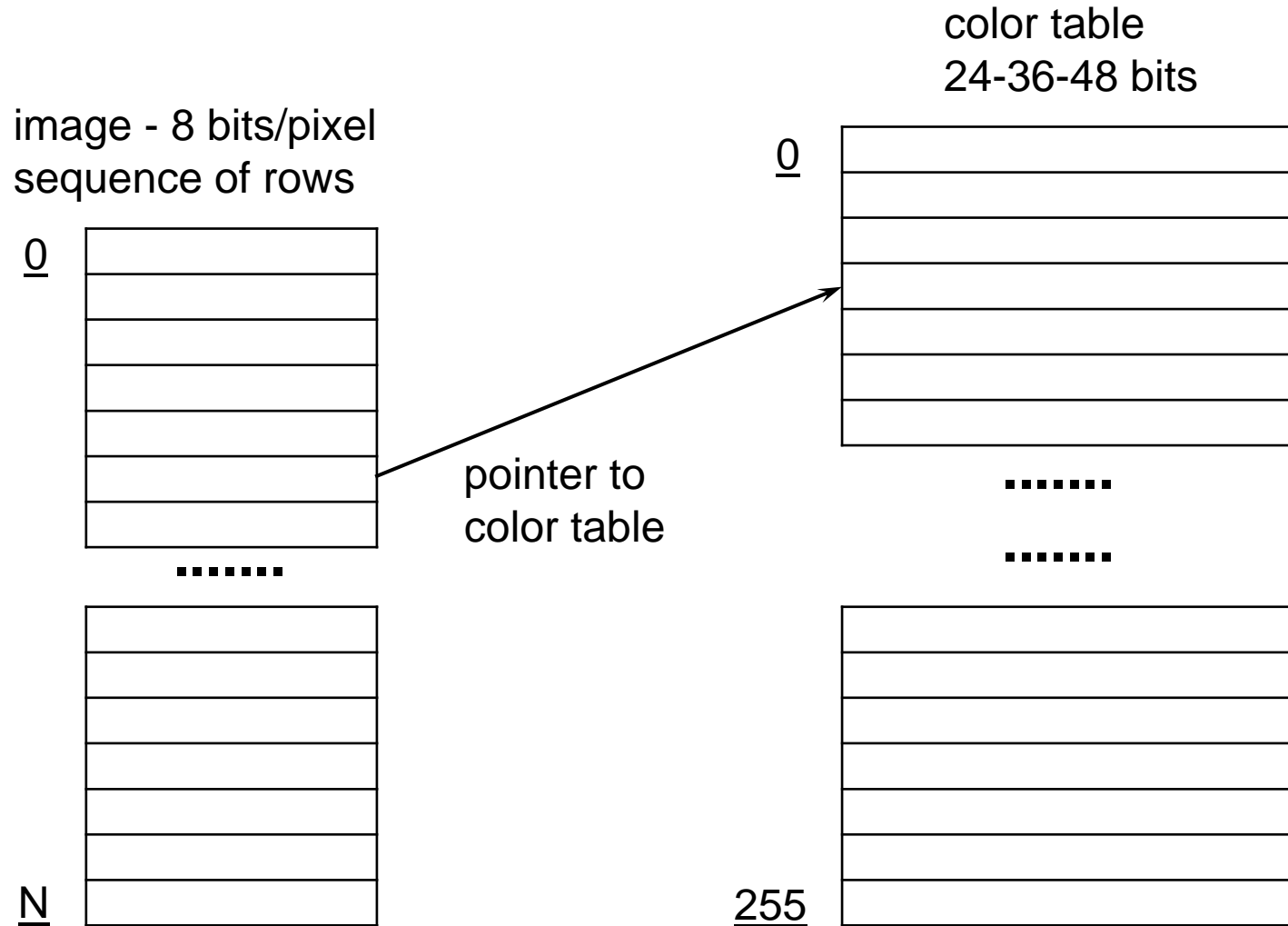
Common raster image file formats



- Lossless compression
 - G3, G4, JBIG, ZIP
 - GIF, PNG
- Lossy compression
 - JPEG
- Not compressed
 - BMP, RAW (sensor output), DNG (Digital Negative), etc.
- Image containers
 - TIFF

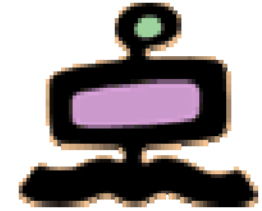


Pixel representation in GIF

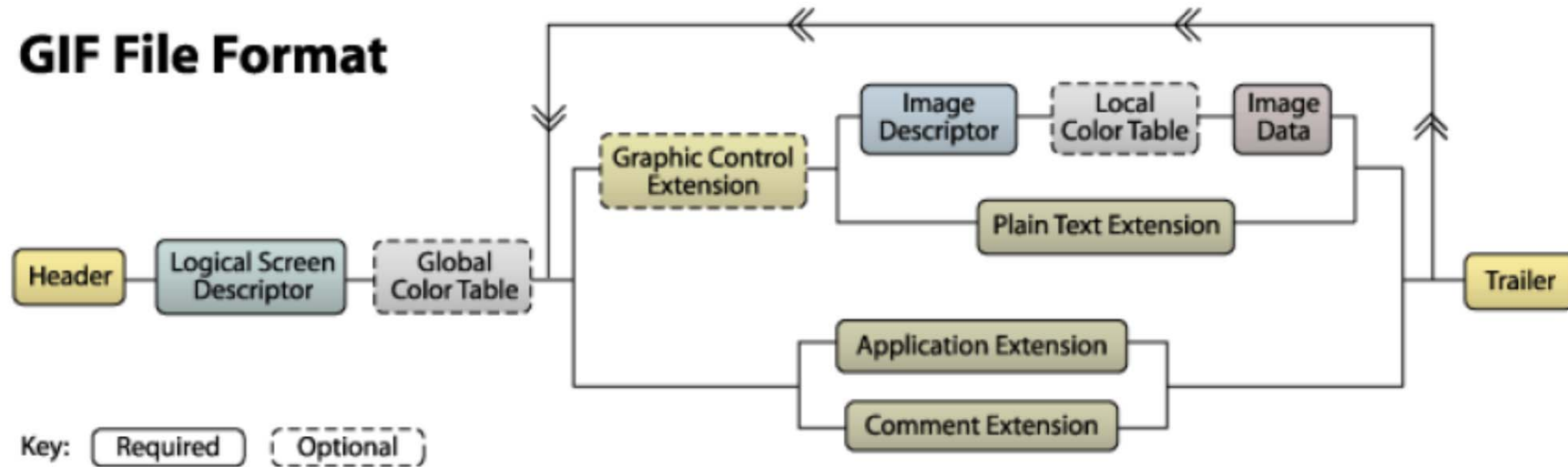




GIF format



GIF File Format





Common raster image file formats



- Lossless compression
 - G3, G4, JBIG, ZIP
 - GIF, PNG
- Lossy compression
 - JPEG
- Not compressed
 - BMP, RAW (sensor output), DNG (Digital Negative), etc.
- Image containers
 - TIFF

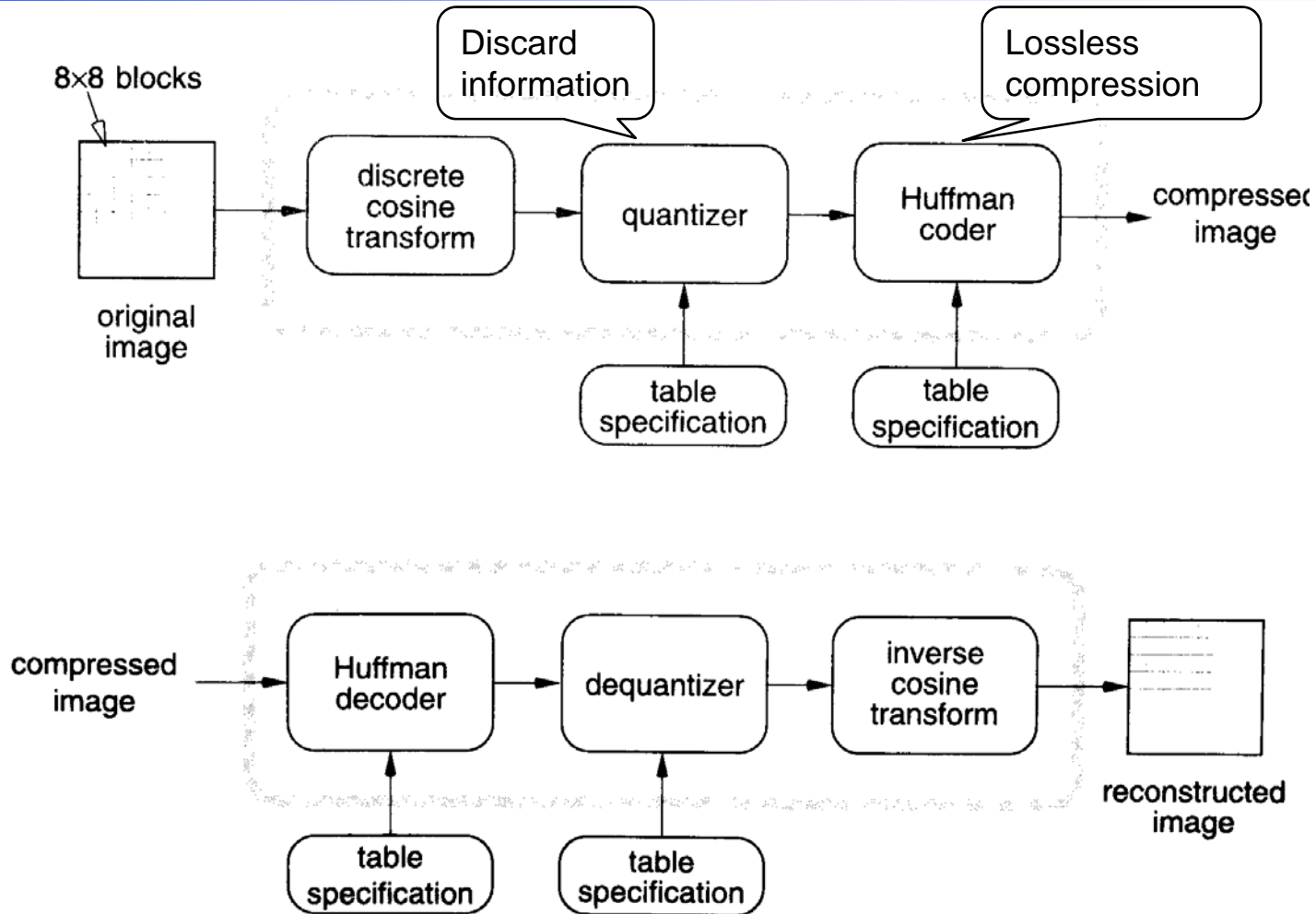
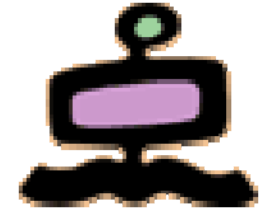


JPEG

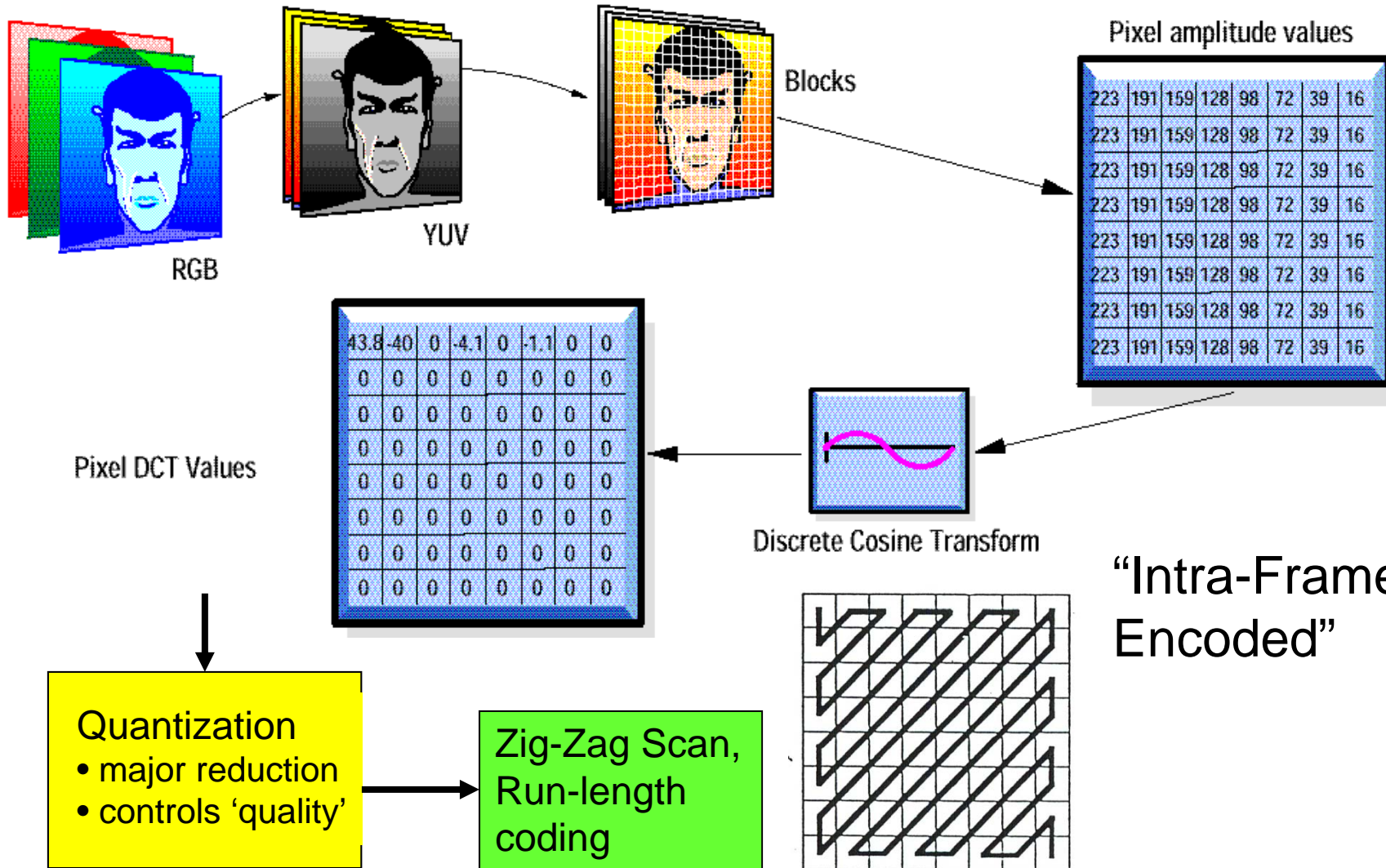


- For grayscale and color images, lossless compression still results in “too many bits”
- Lossy compression methods take advantage from the fact that the human eye is less sensitive to small greyscale or color variation in an image
- JPEG - Joint Photographic Experts Group and Joint Binary Image Group, part of CCITT and ISO
- JPEG can compress down to about one bit per pixel (starting with 8-48 bits per pixel) still having excellent image quality
 - Not very good for fax-like images
 - Not very good for sharp edges and sharp changes in color
- The encoding and decoding process is done on an 8x8 block of pixels (separately for each color component)

JPEG encoding and decoding



Spatial Redundancy Reduction (DCT)





Common raster image file formats



- Lossless compression
 - G3, G4, JBIG, ZIP
 - GIF, PNG
- Lossy compression
 - JPEG
- Not compressed
 - BMP, RAW (sensor output), DNG (Digital Negative), etc.
- Image containers
 - TIFF



TIFF



- Tagged Image File Format – file format that includes extensive facilities for **descriptive metadata**
 - note that TIFF tags are not the same thing as XML tags
- Owned by Adobe, but public domain (no licensing)
- Large number of options
 - Problems of backward compatibility
 - Problems of interoperability
(Thousands of Incompatible File Formats 😊)
- Can include (and describe) four types of images
 - bilevel (black and white), greyscale, palette-color, full-color
- Support of different color spaces
- Support of different compression methods
- Much used in digital libraries and archiving



Representation of information



- Numbers
- Text (characters and ideograms)
- Images
- Video ←
- Audio



Representing video



- Sequence of *frames* (still images) displayed with a given frequency
 - NTSC 30 f/s, PAL 25 f/s, HDTV 60 f/s
- Resolution of each frame depend on quality and video standard
 - 720x480 NTSC, 768x576 PAL, 1920x1080 HDTV
- Uncompressed video requires “lots of bits”
 - e.g. $1920 \times 1080 \times 24 \times 30 = \sim 1,5 \text{ GB/sec}$
- It is possible to obtain very high compression rates
 - Spatial redundancy (within each frame, JPEG-like)
 - Temporal redundancy (across frames)

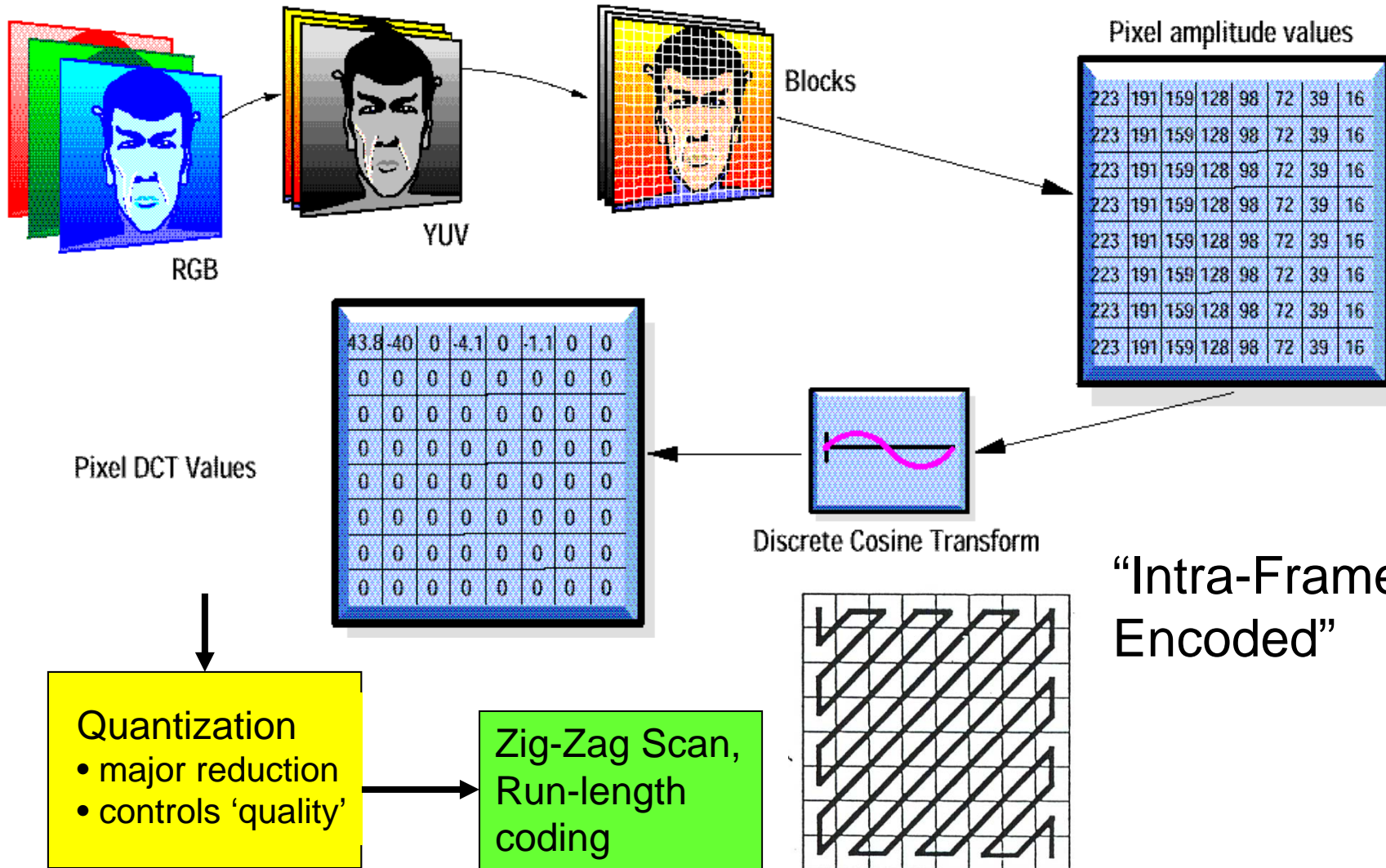


MPEG



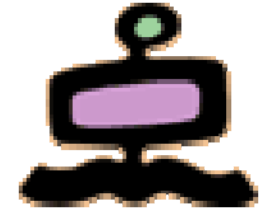
- MPEG - Motion Picture Experts Group established in 1988 as a committee of ISO to develop an open standard for digital TV format (CD-ROM)
- Business motivations
 - Two types of application for videos:
 - Asymmetric (encoded once, decoded many times)
 - Broadcasting, CD's
 - Video games, Video on Demand
 - Symmetric (encoded once, decoded once)
 - Video phone, video mail ...
- Design point for MPEG-1
 - Video at about 1.5 Mbits/sec
 - Audio at about 64-192 kbits/channel

Spatial Redundancy Reduction (DCT)



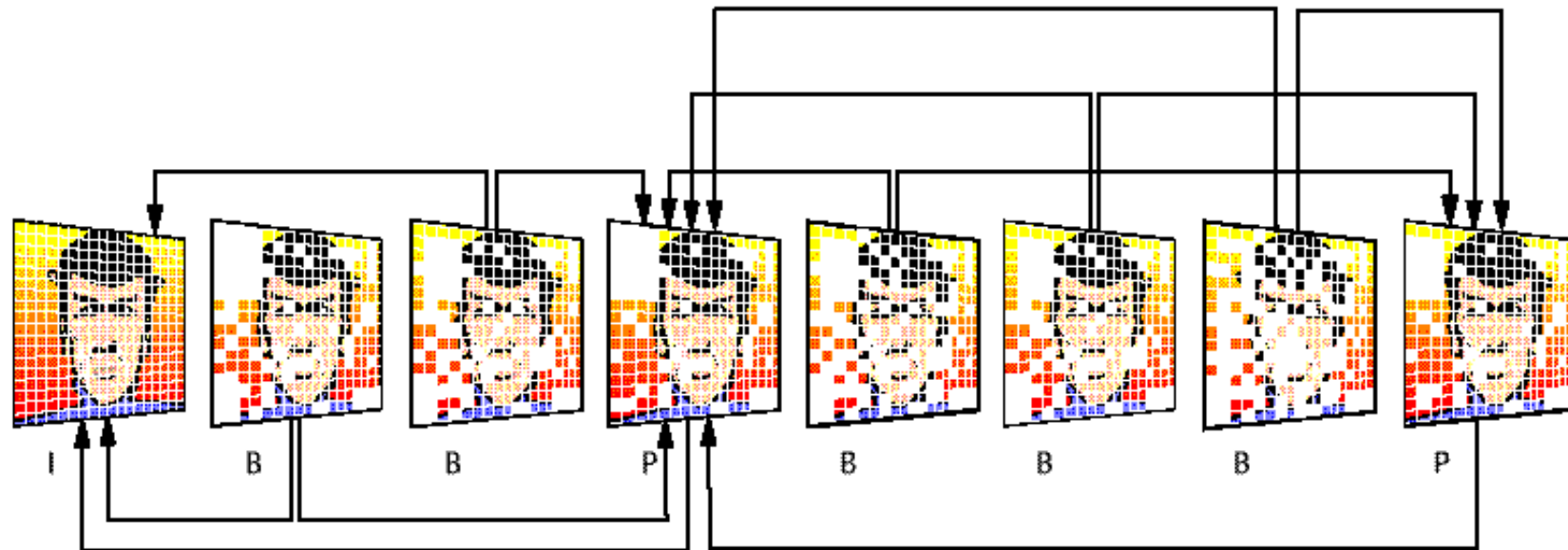


Types of frames in compression



- MPEG uses three types of frames for video coding (compressing)
 - I frames: intra-frame coding
 - Coded without reference to other frames
 - Moderate compression (DCT, JPEG-like)
 - Access points for random access
 - P frames: predictive-coded frames
 - Coded with reference to **previous** I or P frames
 - B frames: bi-directionally predictive coded
 - Coded with reference to **previous and future** I and P frames
 - Highest compression rates

Temporal Redundancy Reduction



- *I* frames are independently encoded
- *P* frames are based on previous *I* and *P* frames
 - Can send motion vector plus changes
- *B* frames are based on previous and following *I* and *P* frames



Typical Compression Performance



Type Size Compression

I	18 KB	7:1
P	6 KB	20:1
B	2.5 KB	50:1
Avg	4.8 KB	27:1



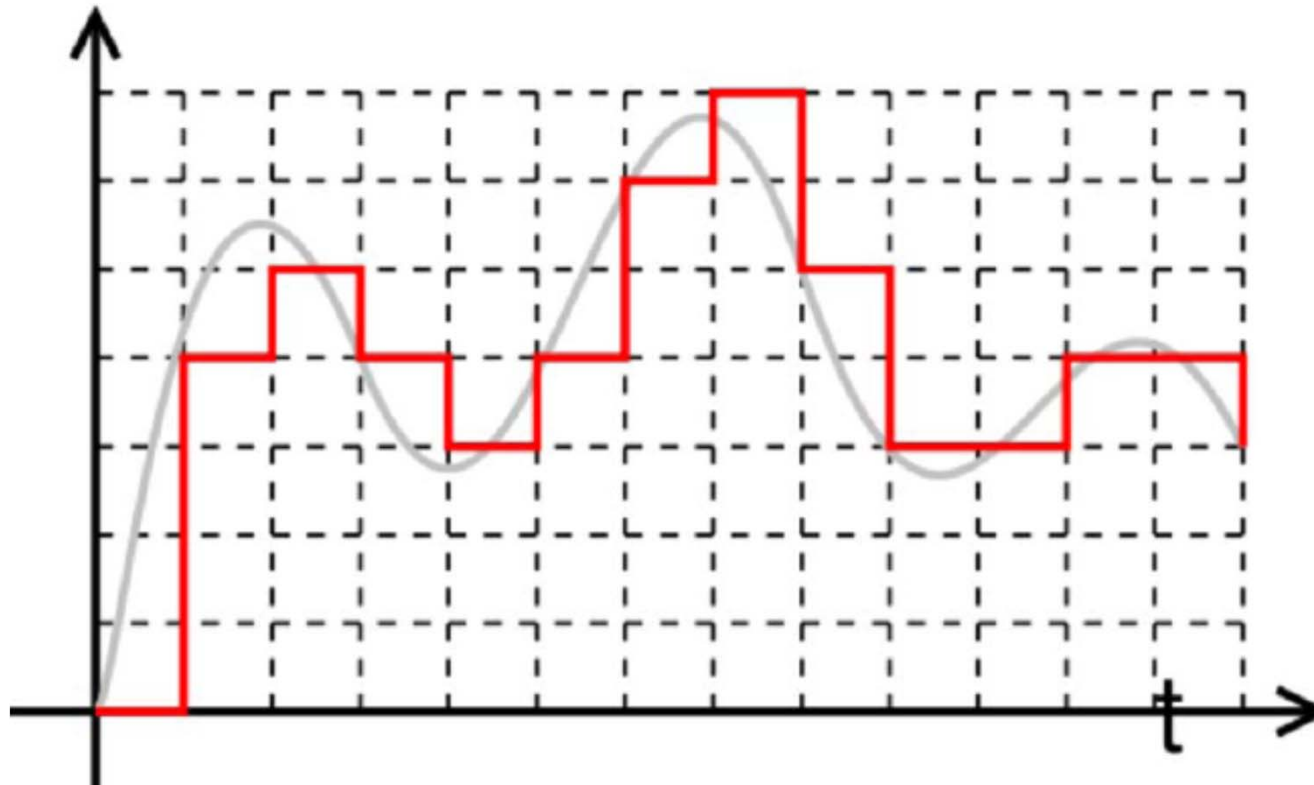
Representation of information



- Numbers
- Text (characters and ideograms)
- Images
- Video
- Audio ←



Digitization of audio (analog) signals



- sampling rate should be at least the double of the highest frequency in the signal (Shannn theorem)
- at least 8 bit per sample

Representing audio



- MPEG-1 defines three different schemes (called *layers*) for compressing audio
- All layers support sampling rates of 32, 44.1 and 48 kHz
- MP3 is MPEG-1 Layer 3

