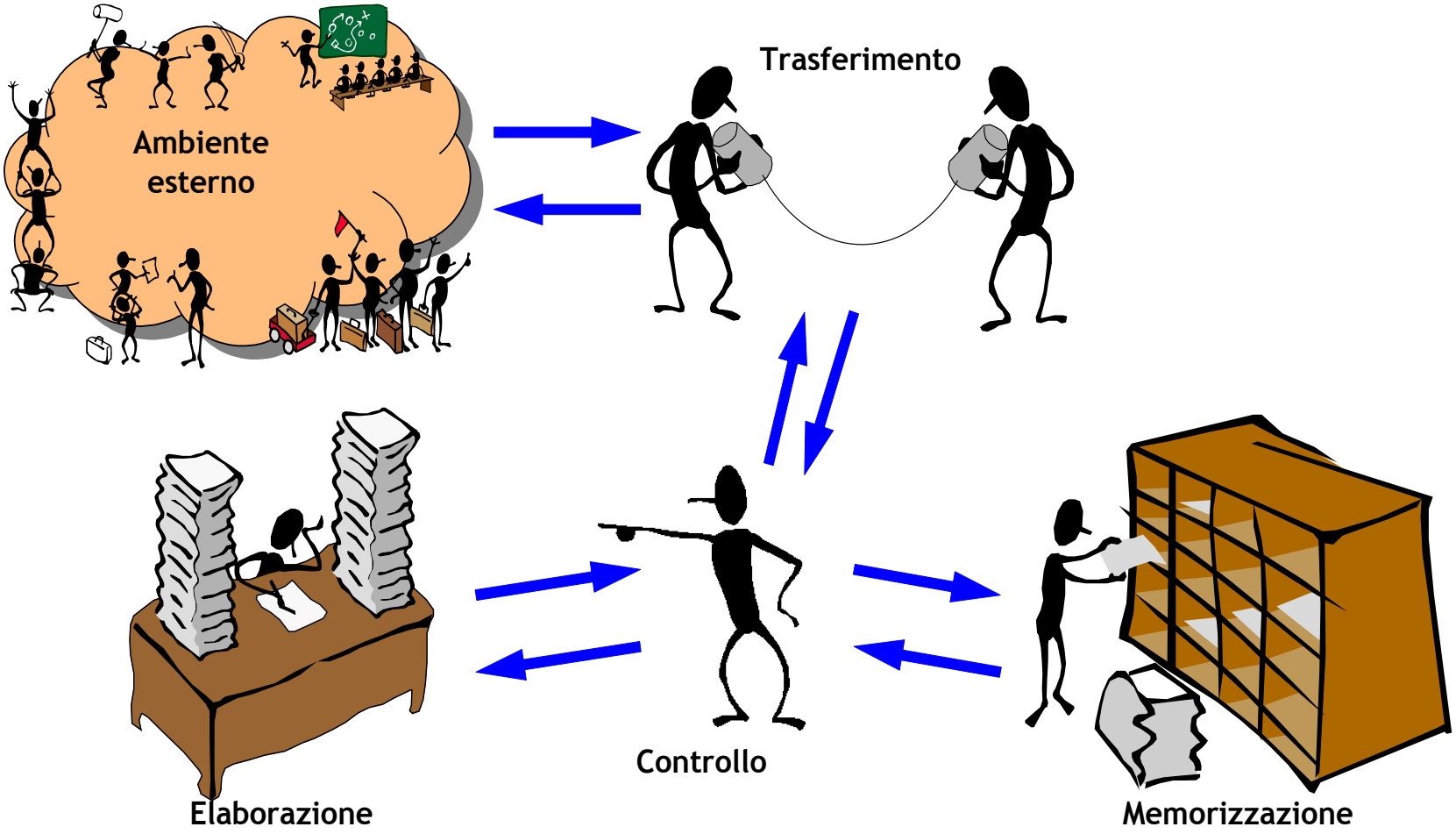
Il sistema di elaborazione

Cosa serve per eseguire istruzioni
in modo automatico

Obiettivi di apprendimento

- Modello concettuale, funzionale e architetturale di un calcolatore
- Codifica delle istruzioni e dei dati per l'esecuzione automatica
- Architettura dei calcolatori e struttura dei sistemi di collegamento delle unità
- Struttura e principi di funzionamento della CPU e della memoria centrale

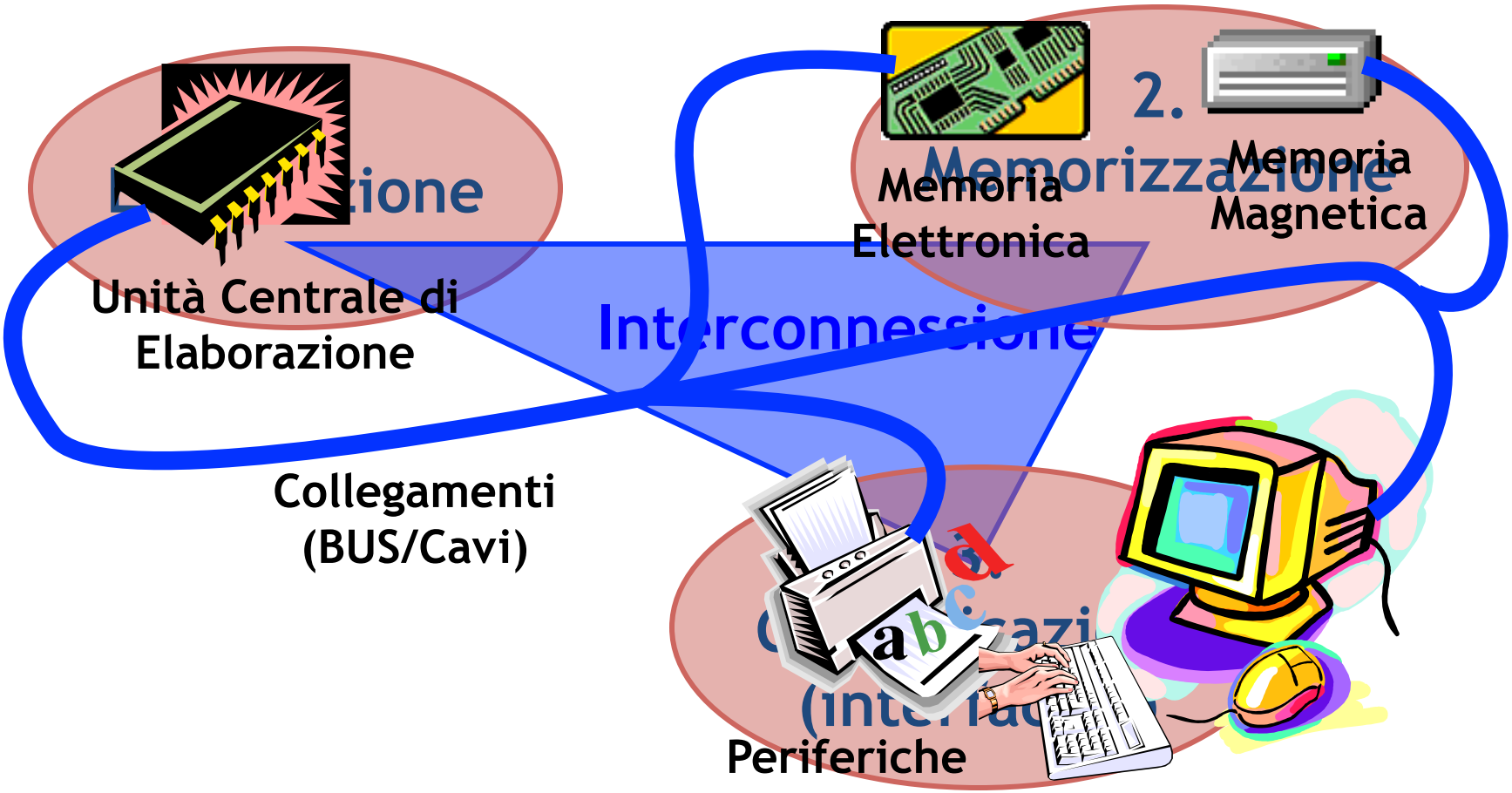
Funzionalità di un calcolatore



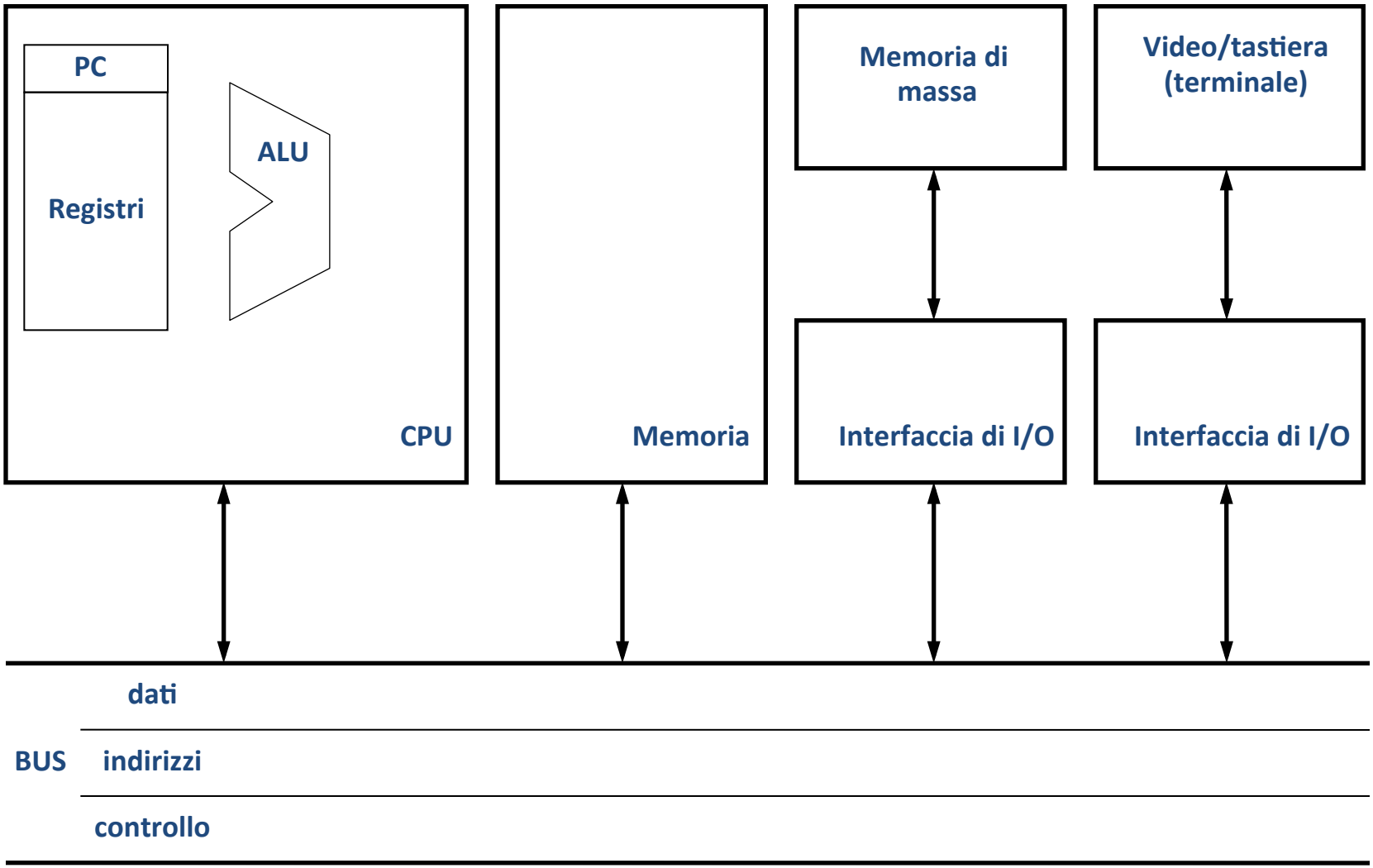
Caratteristiche dell'architettura

- ***Flessibilità nel calcolo,***
  - architettura non specializzata per un solo tipo di utilizzo ma adatta a svolgere diversi compiti;
- ***modularità della struttura***
  - a ogni componente viene demandato lo svolgimento di una funzione specifica del sistema complessivo;
- ***scalabilità dei componenti***
  - ognuno dei quali può essere sostituito con uno funzionalmente equivalente ma in grado di fornire prestazioni migliori;
- ***standardizzazione dei componenti***
  - per facilitarne la sostituzione in caso di malfunzionamenti;
- ***abbattimento dei costi,***
  - grazie alla produzione su vasta scala dei componenti;
- ***semplicità di installazione*** **e di** ***esercizio del sistema*****;**
- ***disponibilità di applicazioni*** **a basso prezzo di vendita.**

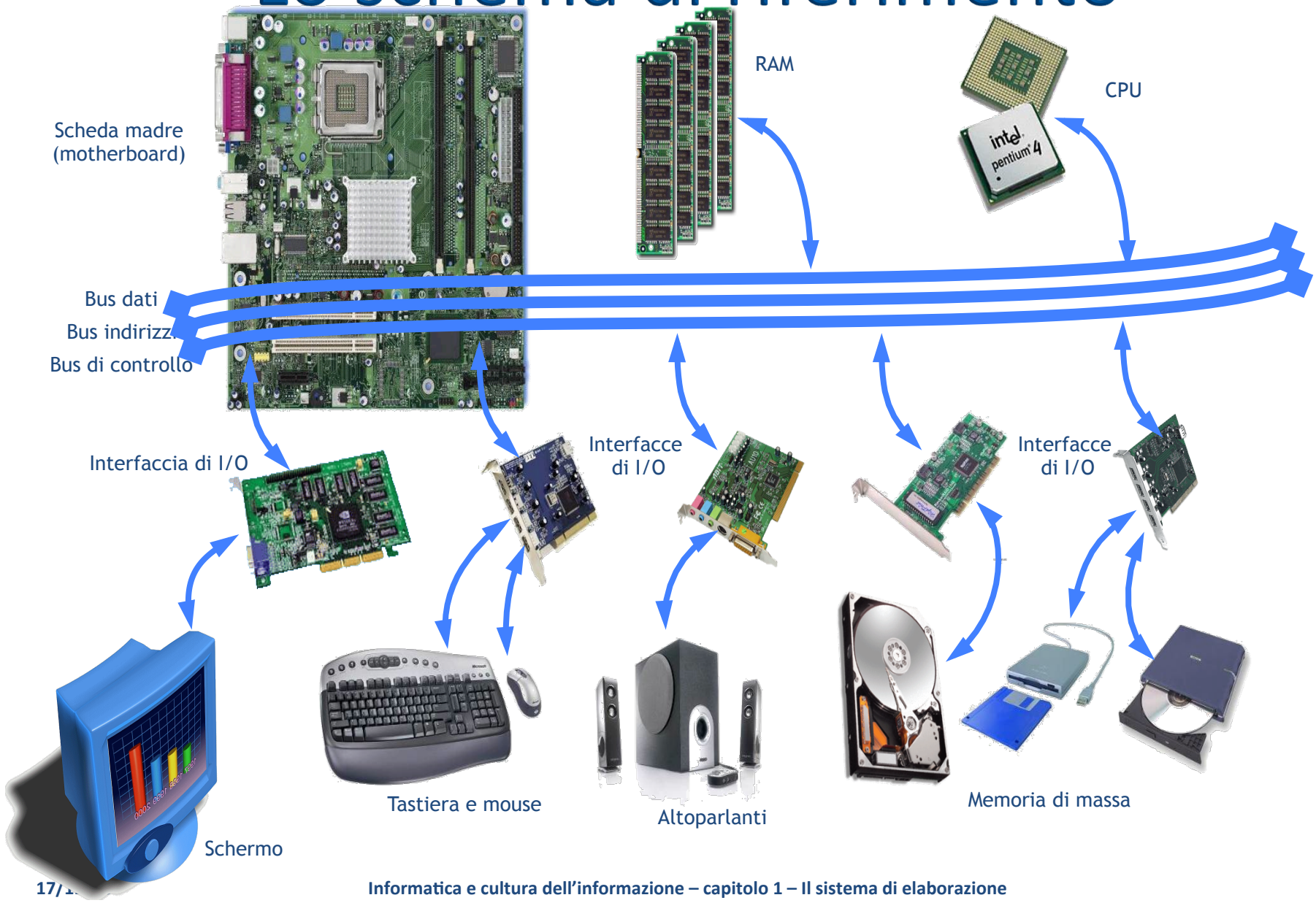
Il calcolatore: modello architetturale



Lo schema di riferimento



Lo schema di riferimento



Breve storia dei calcolatori (1)

- Blaise Pascal (1623-1662)
  dispositivo meccanico (ingranaggi azionati da una manovella) per l'esecuzione di somme e sottrazioni

- Charles Babbage (1792-1871)
  macchina programmabile: "analytical engine" con istruzioni di controllo e output su schede perforate

- Konrad Zuse (Germania, anni '30 e '40)
  - Realizza macchine calcolatrici automatiche basate su relè elettromagnetici

- COLOSSUS, ENIAC (Inghilterra 1943, USA 1946 anni '30 e '40)
  - Negli anni '40 si sviluppa una nuova tecnologia: le valvole termoioniche rendono obsoleti i relè elettromagnetici

Breve storia dei calcolatori (2)

- **La svolta alla fine degli anni 40: il transistor**
- **Inventato ai Bell Labs nel 1948 da John Bardeen, Walter Brattain e William Shockley:**
  - nel giro di 10 anni rivoluziona la ricerca sui calcolatori;
  - alla fine degli anni ’50 i calcolatori a valvole sono già obsoleti
  - nel 1961 DEC realizza il PDP–1, il primo **minicalcolatore**

- **Sviluppo della tecnologia d’integrazione:**
  - decine (SSI), centinaia (MSI) e migliaia (LSI) di transistor sono integrati sullo stesso pezzo di silicio (chip);
  - possibilità di realizzare calcolatori più piccoli, più veloci e meno costosi dei loro predecessori.

Very Large Scale Integration (VLSI)

- **10^5–10^9 transistor integrati per chip.**
- **Passaggio dai minicalcolatori, alle workstation, ai Personal Computer (PC):**
  - usati per applicazioni **fortemente interattive** (elaborazione testi, fogli elettronici, …);
  - in origine proposti come **kit da assemblare**, senza software;
  - due architetture principali:
    - RISC (Motorola PowerPC, ARM)
      - Reduced Intruction Set Computing, ottiene alte performance riducendo il numero di operazioni offerte dal chip
      - **Oggi in iPhone e Android**
    - CISC (Intel, AMD, Cyrix)
      - Complex Instruction Set Computing, ottiene alte performance realizzando sul chip un numero alto di operazioni basilari
      - **Oggi su PC e laptop sia Windows sia Macintosh**
    - Col tempo la differenza si è fatta lieve e non più particolarmente significativa.

Macchina di von Neumann

- **John von Neumann partecipa al progetto ENIAC**
- **Due intuizioni fondamentali:**
  - memorizzare i **programmi** in **forma digitale** nella stessa memoria dei **dati** per rendere più semplice la programmazione (rispetto all'utilizzo di cavi e interruttori)
  - utilizzare **l'aritmetica binaria** invece di quella decimale (due valvole per bit invece di dieci per cifra)
- **Il suo progetto è ancora oggi alla base di quasi tutti i calcolatori digitali!**

La codifica (binaria) dell'informazione

Codifica dati e istruzioni

- **Algoritmo**
  - **descrizione** della **soluzione di problema** scritta in modo da poter essere eseguita da un **esecutore** (eventualmente diverso dall'autore dell'algoritmo)
  - sequenza di **istruzioni** che operano su **dati**.
- **Programma**
  - **algoritmo** scritto in modo da poter essere eseguito da un **calcolatore** (esecutore automatico)

- **Per scrivere un programma è necessario rappresentare istruzioni e dati in un formato tale che l'esecutore automatico sia capace di memorizzare e manipolare.**

Codifica binaria

- Come si memorizza l'**informazione** in un calcolatore?

- Per gli esseri umani una rappresentazione naturale dei numeri è in base dieci, come le dita delle mani, mentre per il testo si può utilizzare un alfabeto di caratteri o un insieme di ideogrammi.

- **Il calcolatore usa una codifica binaria, basata su due simboli 0 e 1**

- Da un punto di vista tecnologico è molto semplice rappresentare un valore binario, basta infatti disporre di un meccanismo che abbia due posizioni distinte. A esempio un interruttore, che può essere aperto o chiuso.
- Una singola cifra binaria prende il nome di **bit**, che è l'acronimo inglese di **binary digit**, ossia cifra binaria. Per rappresentare numeri più grandi di 1 si utilizzano gruppi di 8 bit, detti **byte**.

Sistema posizionale

- Il sistema di numerazione decimale che utilizziamo correntemente è un sistema cosiddetto ***posizionale***, ovvero dove conta la posizione di ogni cifra rispetto alle altre.
  - Ad esempio, in base dieci esistono dieci valori che possono essere rappresentati con una sola cifra, per i valori più grandi di 9 si accostano tra loro più cifre.
- A seconda della posizione delle cifre, queste assumono un significato diverso. Si parla infatti di colonne delle unità, delle decine, delle centinaia, e così via.

Codifica binaria

- Il numero 317 in base dieci può essere interpretato come 3 *centinaia*, 1 *decina* e 7 *unità* ossia:

$$3*10^2 + 1*10^1 + 7*10^0 = 3*100 + 1*10 + 7*1$$

- Allo stesso modo si interpreta un numero in base due.
- Ad esempio:

$$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1$$

$$1*2^5 + 0*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 =$$

$$32 + 0 + 0 + 4 + 0 + 1 = 37$$

Codifica binaria

$101 = 1*2^2 + 0*2^1 + 1*2^0 = 4 + 1 = 5$

$1111 = 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 8 + 4 + 2 + 1 = 15$

$10001101 = 1*2^7 + 1*2^3 + 1*2^2 + 1*2^0 = 141$

$10001100 = 140$

Da decimale a binario

$12 = 8 + 4 = 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0 = 1100$

$33 = 32 + 1 = 1*2^5 + 1*2^0 = 100001$

$67 = 64 + 2 + 1 = 1*2^6 + 1*2^1 + 1*2^0 = 1000011$

Esercizi

100_{10} = 1100100

129_{10} = 10000001

250_{10} = 11111010

1000_{10} = 1111101000

1000_2 = 8

1010_2 = 10

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1024$

Codifica binaria

- Come i numeri decimali anche i numeri binari possono essere sommati, moltiplicati, etc:

```
  101 +
  100 =
-------
 1001
```

```
  111 +
  111 =
-------
 1110
```

```
  111 *
   10 =
-------
  000
 111
-------
 1110
```

Codifica binaria

- Alfabeto binario: usiamo dispositivi con solo due stati
- Problema: assegnare un codice univoco a tutti gli oggetti compresi in un insieme predefinito (e.g. studenti)

- Quanti oggetti posso codificare con k bit?
  - 1 bit ⇒ 2 stati (0, 1) ⇒ 2 oggetti (e.g. Vero/Falso)
  - 2 bit ⇒ 4 stati (00, 01, 10, 11) ⇒ 4 oggetti
  - 3 bit ⇒ 8 stati (000, 001, …, 111) ⇒ 8 oggetti
  - …
  - k bit ⇒ 2^k stati ⇒ 2^k oggetti

Codifica binaria

- Alfabeto binario: usiamo dispositivi con solo due stati
- Problema: assegnare un codice univoco a tutti gli oggetti compresi in un insieme predefinito (e.g. studenti)

- Quanti bit mi servono per codificare N oggetti?

  – $N \leq 2^k \Rightarrow k \geq \log_2 N \Rightarrow k = \lceil \log_2 N \rceil$ (intero superiore)

- Attenzione:
  ipotesi implicita che i codici abbiano tutti la stessa lunghezza

Esempio di codifica binaria

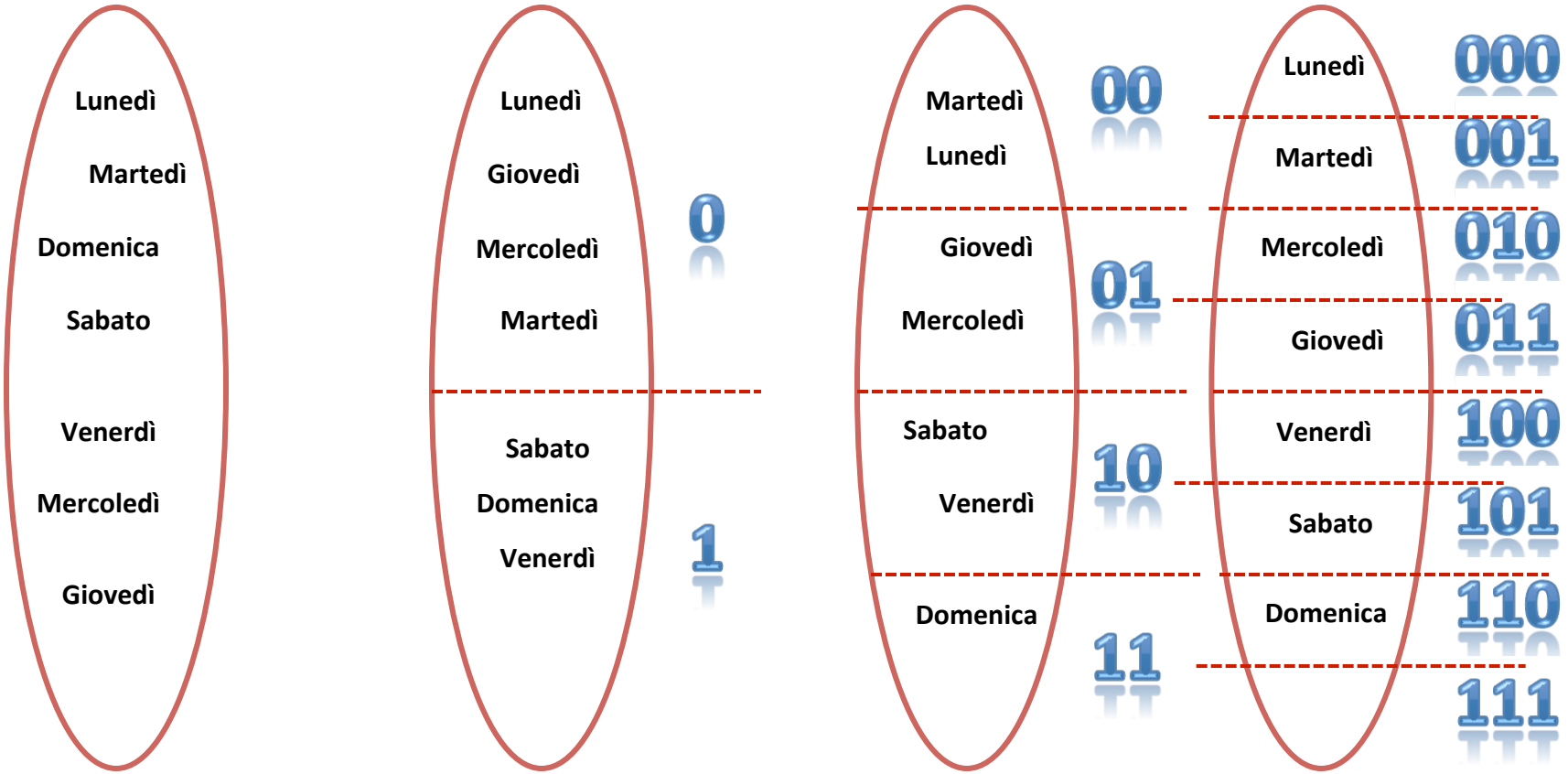
- Problema:
  assegnare un codice binario univoco a tutti i giorni della settimana

- Giorni della settimana: $N = 7 \Rightarrow k \geq \log_2 7 \Rightarrow k = 3$

- Con 3 bit possiamo ottenere 8 diverse configurazioni:
  - Ne servono 7, quali utilizziamo?
  - Quale configurazione associamo a quale giorno?

- Attenzione:
  ipotesi che i codici abbiano tutti la stessa lunghezza

I giorni della settimana in binario (1)



1 bit
2 "gruppi"

2 bit
4 "gruppi"

3 bit
8 "gruppi"

Codifica binaria della scrittura

- Quanti sono gli oggetti compresi nell'insieme?
  - 26 lettere maiuscole + 26 minuscole ⇒ 52
  - 10 cifre
  - Circa 30 segni d'interpunzione
  - Circa 30 caratteri di controllo (EOF, CR, LF, …)

  circa 120 oggetti complessivi ⇒ $k = \lceil \log_2 120 \rceil = 7$
- Codice ASCII: utilizza 7 bit e quindi può rappresentare al massimo 2^7=128 caratteri
  - Con 8 bit (= byte) rappresento 256 caratteri (ASCII esteso)
  - Con codici più estesi (e.g. UNICODE) rappresentiamo anche i caratteri delle lingue orientali

ASCII su 7 bit

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{	|	}	~	canc

c è codificato in ASCII: 1100011

C è codificato in ASCII: 1000011

Esercizio

- Decodificare la seguente sequenza in ASCII (7 bit)

  1001000110010111011001101100111011111

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
010	sp	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
110	`	a	b	c	d	e	f	g	h	I	j	k	l	m	n	o
111	p	q	r	s	t	u	v	w	x	Y	z	{	|	}	~	canc

Evoluzioni nella codifica dei caratteri

- Esistono molte variazioni dell'alfabeto latino (lettere accentate in italiano, Umlaut in tedesco, ecc.) che non essendo presenti in inglese non sono state incluse in ASCII
- Nel 1991 è stato approvato uno standard, chiamato ISO-Latin 1 (ISO/IEC 8859) che contiene in 8 bit (256 combinazioni) tutti i caratteri nazionali basati su lettere dell'alfabeto latino. ISO Latin 1 è un'estensione di ASCII.
- Parallelamente alla codifica dei caratteri latini, ci sono state analoghe codifiche per caratteri del cirillico, del greco, dell'ebraico, dell'arabo, dell'hindu, ecc. Sono indipendenti da ASCII o ISO-Latin 1 e incompatibili con esso.
- Discorso diverso per i caratteri CJKV (giapponesi, cinesi, coreani e vietnamiti), di origine cinese e per cui non bastano 8 bit.
- Il mondo occidentale ha adottato uno schema di codifica a lunghezza variabile chiamato UTF-8, identica a ASCII per i primi 128 caratteri e di lunghezza più ampia per i successivi.

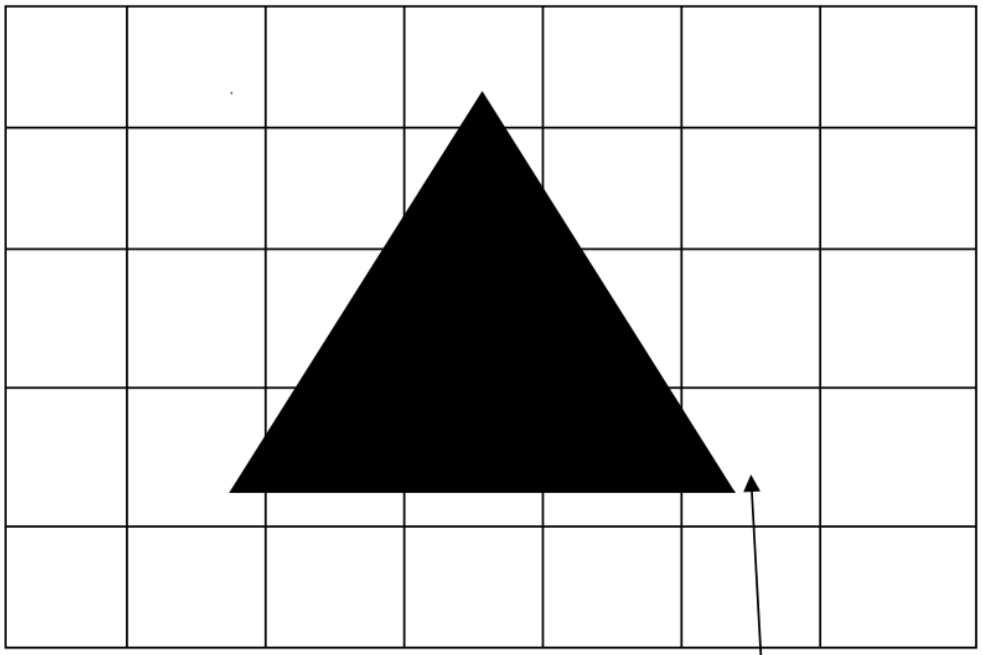
Come codificare immagini?

- Esistono due tecniche principali:
  - **Formato Raster/Bitmap**: scomposizione dell'immagine in una griglia di elementi (punti, detti pixel) che sono l'unità minima di memorizzazione
  - **Formato Vettoriale**: composizione di strutture elementari quali linee, circonferenze, poligoni, etc.

Bitmap in bianco e nero

- L'immagine è divisa in una griglia di pixel (picture element)
- Ogni pixel è codificato con:
  - 0 se è bianco
  - 1 se è nero
- Si stabilisce un ordinamento dei bit usati per la codifica
  - Ad esempio, da sinistra a destra e dall'alto in basso

Bitmap in bianco e nero



Pixel = 1

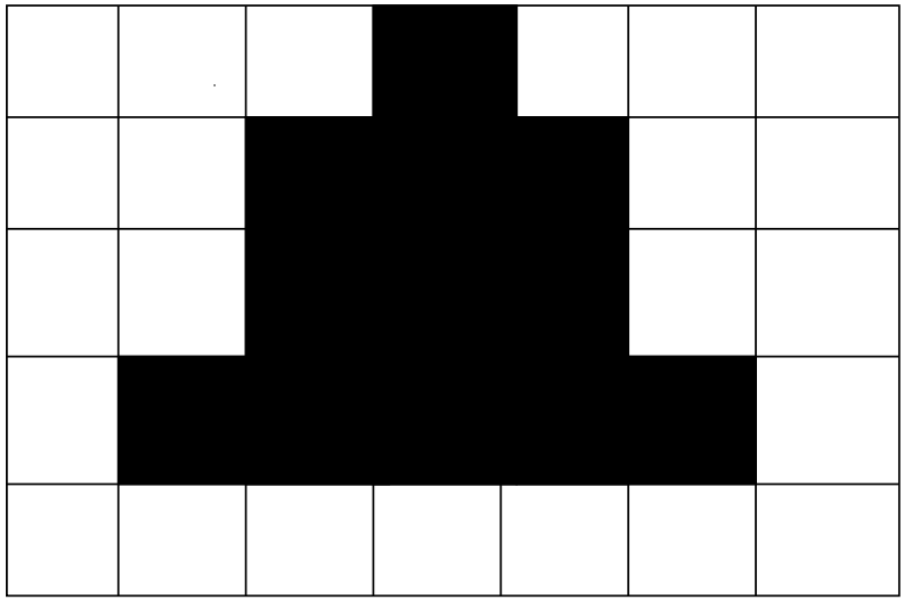
0 0 0 1 0 0 0
0 0 1 1 1 0 0
0 0 1 1 1 0 0
0 1 1 1 1 1 0
0 0 0 0 0 0 0

codifica

Bitmap in bianco e nero

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

Codifica

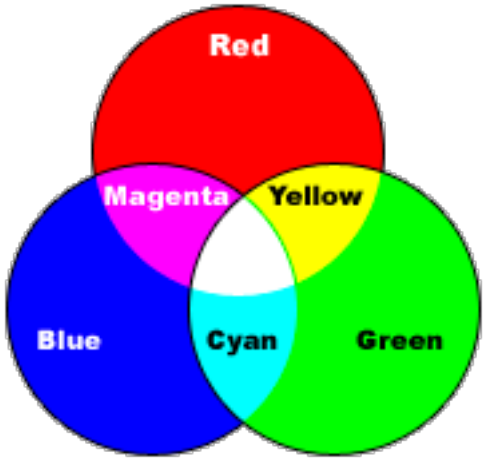


Immagine

Figura approssimata! Più pixel, maggior qualità

E a colori?

- Si definiscono insiemi di **sfumature di colore** rappresentate tramite **sequenze di bit**.
- Nella codifica RGB si usano 3 colori (Red, Green, Blue)
  - Quanti colori possibili usando 2 bit per colore?
  - Quanti usando 1 byte?



Immagini vettoriali

- Si utilizza quando le immagini da memorizzare hanno caratteristiche geometriche ben definite
- Il disegno viene scomposto in elementi base (linea, arco, etc.)
- Si memorizzano le **istruzioni per disegnare l'immagine**
- Esempi: PDF, SVG
- Vantaggi?
  - Spazio ridotto
  - Scalabilità

Codifica informazione

- Per poter essere elaborata l'informazione va codificata quindi in sequenze di bit
- Con lo stessi principi visti finora riusciamo a codificare quindi
  - Numeri
  - Caratteri
  - Testi
  - Immagini
  - Suoni
  - Video
- Lo spazio occupato dipende quindi dalla codifica usata (ed eventualmente dalla compressione dei dati)

bit, Byte, KiloByte, MegaByte, …

bit = solo due stati, "0" oppure "1".
Byte = 8 bit, quindi $2^8 = 256$ stati
KiloByte [KB] $= 2^{10}$ Byte = 1024 Byte
~ 10^3 Byte
MegaByte [MB] $= 2^{20}$ Byte = 1'048'576 Byte ~ 10^6 Byte
GigaByte [GB] $= 2^{30}$ Byte ~ 10^9 Byte
TeraByte [TB] $= 2^{40}$ Byte ~ 10^{12} Byte
PetaByte [PB] $= 2^{50}$ Byte ~ 10^{15} Byte
ExaByte [EB] $= 2^{60}$ Byte ~ 10^{18} Byte

Riferimenti

- Libro di testo, Capitolo 1
- Codifica binaria in Capitolo 5, in particolare 5.3