Architettura dell'Informazione

1. Cenni su Architetture degli Elaboratori e Programmazione

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
http://www.di.unipi.it/∼milazzo
milazzo@di.unipi.it

Master in ICT e Turismo
A.A. 2015/2016

Sommario

Com'è fatto un computer (1)

In questo corso impareremo a conoscere il World Wide Web (WWW) e alcune nuove tecnologie ad esso collegate

Fino a pochi anni fa il WWW era uno strumento prevalentemente di navigazione:

- si passava da un sito all'altro allo scopo di leggerne i contenuti

Da qualche anno a questa parte (con la diffusione delle tecnologie di comunicazione a "banda larga") il WWW è diventato sempre più un mondo popolato da applicazioni

- i siti web sono diventati veri e propri programmi con i quali gli utenti interagiscono

Per capire come funzionano le nuove applicazioni web è quindi necessario sapere minimamente come funzionano i programmi e, di conseguenza, come funzionano i computer (su cui i programmi vengono eseguiti)...

Com'è fatto un computer (2)

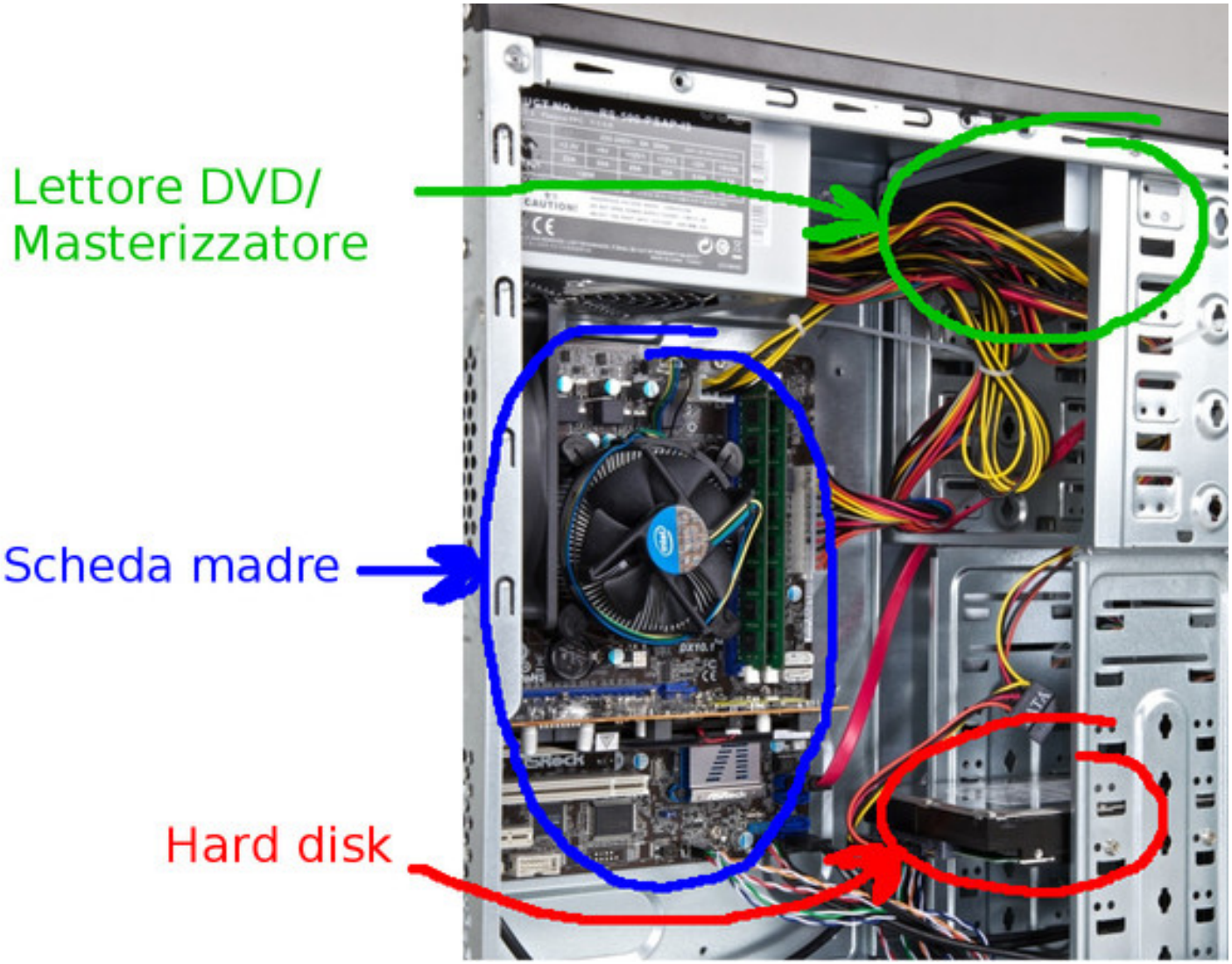
Il miglior modo per capire com'è fatta una cosa è... smontarla!

Foto da tutorial PC professionale

http://www.pcprofessionale.it/2012/03/28/costruire-un-pc-la-guida-passo-passo-completa/

Com'è fatto un computer (3)

Identifichiamo i componenti più "appariscenti"

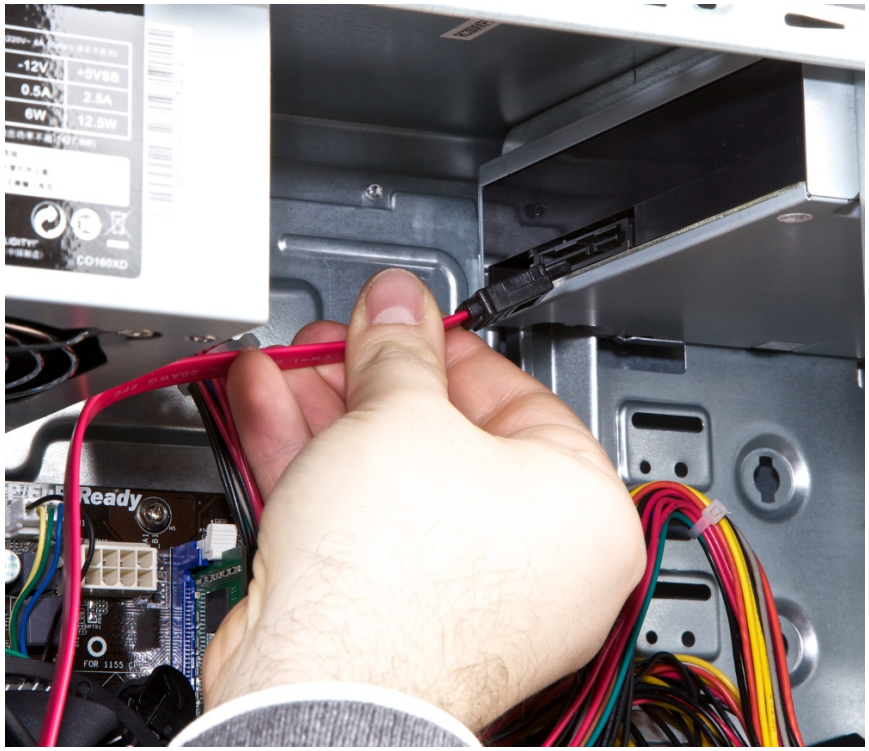


Com'è fatto un computer (4)

Innazitutto dobbiamo scollegare i cavi di alimentazione...

Com'è fatto un computer (5)

... e i cavi-dati



Com'è fatto un computer (6)

Ora possiamo rimuovere il DVD e l'hard disk

Com'è fatto un computer (7)

A questo punto possiamo dedicarci alla scheda madre e ai componenti ad essa collegati. Per prima la scheda video (a cui era collegato il monitor)

Com'è fatto un computer (8)

Ora la memoria RAM

Com'è fatto un computer (9)

Sotto il grosso dissipatore (con ventola) posizionato al centro della scheda madre...

Com'è fatto un computer (10)

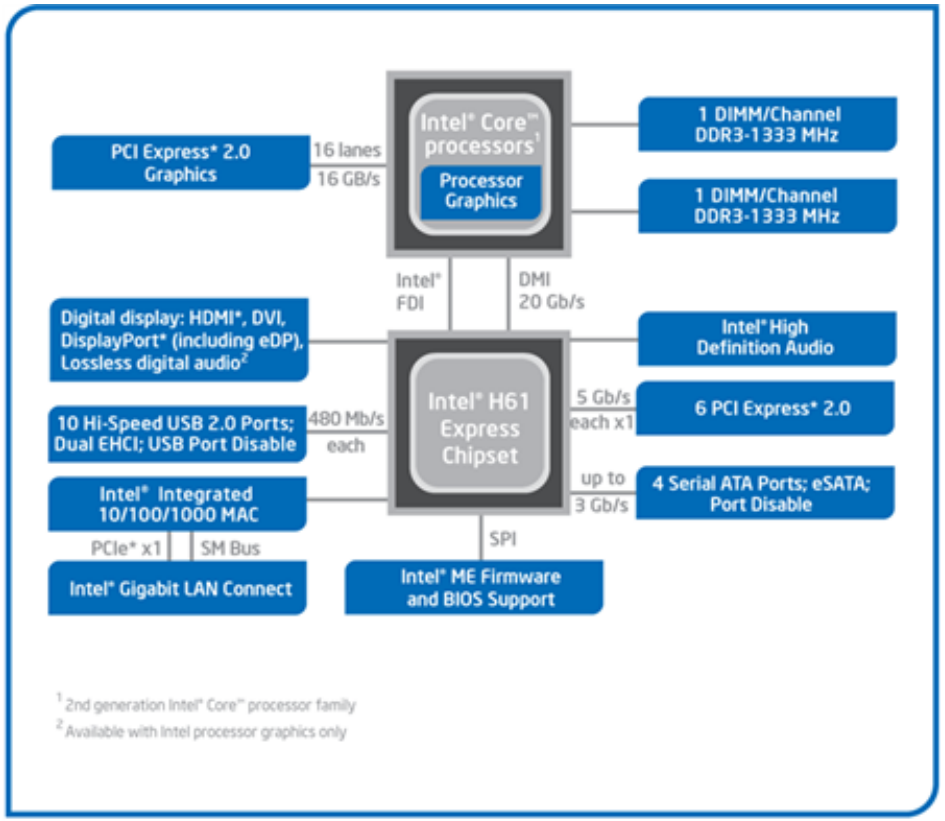
...troviamo il processore (o CPU)

Com'è fatto un computer (11)

E infine non ci rimane che rimuovere la scheda madre

Com'è fatto un computer (12)

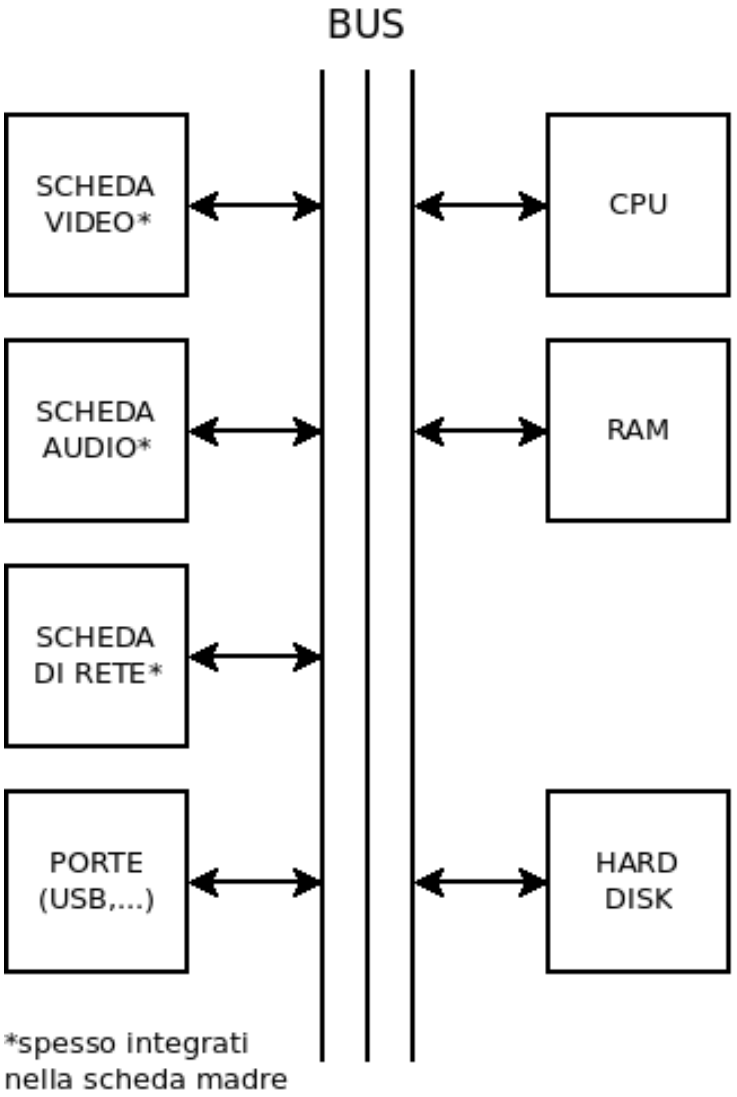
Una rappresentazione schematica (ufficiale – Intel). I componenti sono collegati tramite BUS di comunicazione gestiti dal processore e da un apposito chipset.



Intel® H61 Express Chipset Platform Block Diagram

Com'è fatto un computer (13)

Una rappresentazione semplificata (unico bus di comunicazione in cui il chipset non è rappresentato)



La CPU e la memoria RAM

I componenti chiave per l'esecuzione di un programma sono la CPU e la memoria RAM.

- La CPU è un chip capace di eseguire operazioni molto semplici (operazioni aritmetiche, operazioni logiche, scrittura/lettura di piccole quantità di dati nella memoria RAM, ecc...) in tempi rapidissimi, nell'ordine dei miliardi di operazioni al secondo (in teoria...).
- La memoria RAM è una memoria volatile (si cancella quando non è alimentata) che consente di memorizzare grandi quantità di dati rappresentati in formato binario (ossia come sequenze di valori 0 e 1).

CPU e memoria RAM interagiscono continuamente:

- La CPU esegue le proprie operazioni e usa la RAM come un "taccuino" in cui annotarsi i risultati intermedi e finali

La memoria RAM non va confusa con la memoria di massa (e.g. disco fisso, schede SD) che memorizza i file a disposizione del computer in maniera persistente.

La CPU

Le operazioni che può svolgere la CPU sono codificate tramite un linguaggio in codice binario detto linguaggio macchina

```
Program Fragment:        Y = Y + X
Machine Language Code
(Binary Code)
Opcode          Address
1100 0000       0010 0000 0000 0000
1011 0000       0001 0000 0000 0000
1001 0000       0010 0000 0000 0000
```

Il linguaggio macchina è di solito rappresentato tramite una notazione simbolica detta linguaggio assembly

```
mov bx,ax        ;copy the content of ax into bx
add ax,bx        ;add value in bx to value in ax
sub bx,1         ;subtract 1 from the bx value
add [bx],ax      ;add value in ax to memory _word_ pointed to by bx
add [bx],al      ;add value in al to memory _byte_ pointed to by bx
```

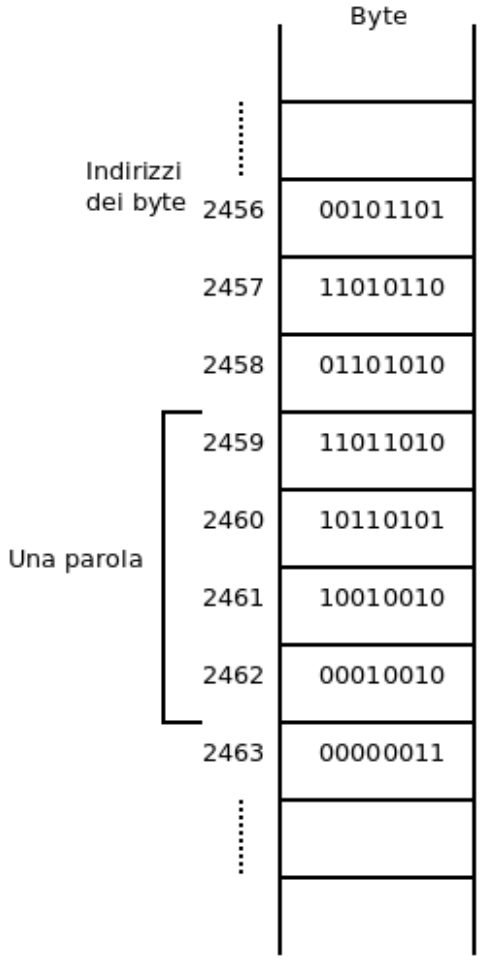
La memoria RAM

Un modulo di memoria RAM può memorizzare decine di miliardi di bit 0/1.

- I bit sono organizzati in gruppi di 8, detti byte
- I byte sono organizzati in gruppi di 4 (32 bit) oppure 8 (64 bit) detti word (o parola)

I singoli byte della memoria RAM sono associati ad un valore numerico (indirizzo di memoria) che li distingue.

Una parola solitamente è la quantità massima di memoria che una CPU può utilizzare in una singola operazione. Operazioni che lavorano su dati più grandi vengono svolte in più passi dalla CPU.



E ora?

Prima di poter programmare un computer dobbiamo:

- trovare un modo per rappresentare i nostri dati in memoria
  - Rappresentazione binaria
- trovare dei linguaggi più semplici da usare rispetto all'assembly
  - Linguaggi di programmazione e algoritmi

Vediamo un po' meglio questi due argomenti...

Rappresentazione binaria dell'informazione (1)

Per informazione intendiamo tutto quello che viene manipolato da un calcolatore:

- Numeri (naturali, interi, reali, ...)
- Caratteri
- Immagini
- Suoni
- Programmi

Rappresentazione binaria dell'informazione (2)

La più piccola unità di informazione memorizzabile o elaborabile da un calcolatore, il bit, corrisponde allo stato di un dispositivo fisico (ad esempio spento/acceso) che viene interpretato come 0 o 1.

In un calcolatore tutte le informazioni sono rappresentate in forma binaria, come sequenze di 0 e 1.

Per motivi tecnologici: distinguere tra due valori di una grandezza fisica è più semplice che non ad esempio tra dieci valori

- Ad esempio, verificare se su un connettore c'è una tensione elettrica di 5V o meno è più facile che verificare se sullo stesso connettore c'e' una tensione di 0V, 1V, 2V, 3V, 4V o 5V.

Rappresentazione binaria dell'informazione (3)

La rappresentazione binaria dei numeri (naturali) è analoga alla rappresentazione decimale (che usiamo abitualmente)

- L'unica differenza è che non si usano le cifre $0, \ldots, 9$ ma solo 0 e 1.

Esempio. Contiamo fino a dieci in binario:

Decimale	0	1	2	3	4	5	6	7	8	9	10
Binario	0	1	10	11	100	101	110	111	1000	1001	1010

Rappresentazione binaria dell'informazione (4)

Le operazioni aritmetiche su numeri binari sono analoghe a quelle su numeri decimali

- Ma ricordando che $1 + 1 = 10$

$$\begin{array}{rcl} 11100 & + & \\ 1011 & = & \\ \hline 100111 & & \end{array} \qquad \begin{array}{rcl} 10011001 & - & \\ 00110011 & = & \\ \hline 01100110 & & \end{array} \qquad \begin{array}{rcl} 10011001 & \times & \\ 1011 & = & \\ \hline 10011001 & + & \\ 10011001 & + & \\ 00000000 & + & \\ 10011001 & = & \\ \hline 11010010011 & & \end{array}$$

$$\begin{array}{r|l} 111100 & 100 \\ \cline{2-2} 100 & 1111 \\ \cline{1-1} =111 & \\ 100 & \\ \cline{1-1} =110 & \\ 100 & \\ \cline{1-1} =100 & \\ 100 & \\ \cline{1-1} === & \end{array}$$

Rappresentazione binaria dell'informazione (5)

I singoli caratteri di un testo possono essere rappresentati come numeri (che a loro volta possono essere tradotti in binario).

La seguente tabella riporta la codifica ASCII (o meglio, US-ASCII) è nata negli anni 60 per rappresentare (con 7 bit) i simboli della tastiera americana

0	*NUL*	16	*DLE*	32	SPC	48	0	64	@	80	P	96	`	112	p
1	*SOH*	17	*DC1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*STX*	18	*DC2*	34	"	50	2	66	B	82	R	98	b	114	r
3	*ETX*	19	*DC3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*EOT*	20	*DC4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*ENQ*	21	*NAK*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ACK*	22	*SYN*	38	&	54	6	70	F	86	V	102	f	118	v
7	*BEL*	23	*ETB*	39	'	55	7	71	G	87	W	103	g	119	w
8	*BS*	24	*CAN*	40	(	56	8	72	H	88	X	104	h	120	x
9	*HT*	25	*EM*	41	)	57	9	73	I	89	Y	105	i	121	y
10	*LF*	26	*SUB*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*VT*	27	*ESC*	43	+	59	;	75	K	91	[	107	k	123	{
12	*FF*	28	*FS*	44	,	60	<	76	L	92	\	108	l	124	|
13	*CR*	29	*GS*	45	-	61	=	77	M	93	]	109	m	125	}
14	*SO*	30	*RS*	46	.	62	>	78	N	94	^	110	n	126	~
15	*SI*	31	*US*	47	/	63	?	79	O	95	_	111	o	127	DEL

Rappresentazione binaria dell'informazione (6)

La codifica ASCII è stata superata negli anni da codifiche più ricche.

In primis, la codifica Extended-ASCII (8 bit)

Attualmente, una codifica che si sta affermando (specialmente in ambito web) è UTF-8

- usa da 1 a 4 byte (variabile) per rappresentare un carattere

Le stringhe sono sequenze di caratteri che possono contenere un testo

- possono essere rappresentate nella memoria di un computer come una sequenza di caratteri terminata dal carattere NUL (= numero 0)

Rappresentazione binaria dell'informazione (7)

Una immagine può essere rappresentata nella memoria di un computer codificando il colore dei singoli pixel che la compongono tramite valori numerici

Ad esempio, nel formato `bmp` a 24 bit ogni pixel dell'immagine viene rappresentato da 3 byte

- ogni byte (256 valori) rappresenta il livello di un colore fondamentale RGB
- 111111110000000000000000 = (255,0,0) = rosso
- 000000001111111100000000 = (0,255,0) = verde
- 000000000000000011111111 = (0,0,255) = blu
- 111111111111111111111111 = (255,255,255) = bianco
- 111111111001100110011001 = (255,204,204) = rosa

Rappresentazione binaria dell'informazione (8)

Altri formati più sofisticati (`gif`, `png`, `jpeg`) utilizzano metodi di compressione per ridurre la dimensione della rappresentazione

- I formati `gif` e `png` limitano il numero di colori utilizzabili nell'immagine (palette) e in più applicano un algoritmo di compressione all'immagine (simile a zip)
- Il formato `jpeg` non limita il numero di colori (preferibile per immagini fotografiche) ma applica una leggera sfocatura per ridurre la dimensione del file dell'immagine.

I formati `gif` e `jpeg` sono pienamente supportati nell'ambito del Web. Da un po' di anni lo stesso vale anche per `png`...

Rappresentazione binaria dell'informazione (9)

Approcci simili sono adottati per rappresentare suoni e filmati:

- mp3 formato compresso per l'audio digitale
- Ogg formato compresso per l'audio e il video digitale
- mp4 formato compresso per il video digitale
- WebM formato compresso per il video digitale

Sommario

Problemi computazionali

L'Informatica è una scienza che studia principalmente metodi e strumenti per la risoluzione di problemi computazionali (problem solving)

Un problema computazionale è un problema che richiede

- di calcolare un risultato (output)
- a partire da determinati valori noti (input)

Esempi di problemi computazionali:

- Calcolo del massimo comune divisore di due numeri
- Preparazione di un risotto ai funghi
- Tessitura di tappeto con un telaio meccanico
- Ordinamento di una sequenza di numeri
- Ricerca di una parola in un testo
- Risoluzione di un Sudoku

Algoritmi (1)

I problemi computazionali possono essere risolti tramite algoritmi.

Un algoritmo è una sequenza finita di passi di elaborazione che, dato un input, consentono di ottenere l'output ad esso corrispondente.

Algoritmi (2)

Un algoritmo si può esprimere in molti modi diversi:

- In linguaggio naturale:
  - "Per calcolare il massimo comune divisore di X e Y maggiori di 0 bisogna ripetutamente sottrarre il più piccolo dei due dal più grande. Quando i due numeri saranno diventati uguali tra loro, il loro valore corrisponderà al risultato."
- Tramite formule matematiche:
  - $mcd(X, Y) = \begin{cases} mcd(X - Y, Y) & \text{se } X > Y \\ mcd(X, Y - X) & \text{se } Y > X \\ X & \text{altrimenti} \end{cases}$
- In pseudo-codice (testo schematico):

```
finchè X!=Y ripeti {
  se Y>X scambia(X,Y)
  X = X-Y
}
stampa X
```

Algoritmi (3)

Un buon algoritmo deve soddisfare alcune proprietà:

- Non ambiguità: I singoli passi devono essere "elementari": facilmente eseguibili (atomici) e non ambigui
- Determinismo: Eseguito più volte sullo stesso input, l'algoritmo deve eseguire sempre la stessa sequenza di passi (e conseguentemente deve dare lo stesso risultato)
- Terminazione: L'esecuzione dell'algoritmo deve prima o poi terminare e fornire un risultato

Esempi di Algoritmi (1)

Supponiamo di voler comprare un'auto il più possibile economica:

Ci vengono proposte due alternative:

	
10000 euro	15000 euro
8 km/l	20 km/l

Quale scegliamo?

Esempi di Algoritmi (2)

Ci serve qualche informazione in più:

- Vogliamo percorrere circa 50000 km
- Il prezzo della benzina è circa 2 euro al litro

Ora possiamo usare il seguente algoritmo (in pseudo-codice):

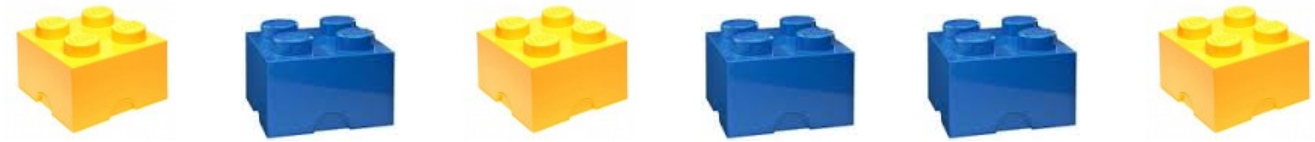
```
per ogni auto calcola {
   costo_gas = (km_percorsi/km_al_litro) x prezzo_benzina
   costo_tot = prezzo_acquisto + costo_gas
}
se costo_tot(auto1) < costo_tot(auto2)
  compra auto1
altrimenti
  compra auto2
```

Questo algoritmo è non ambiguo (i passi sono semplici), termina (il ciclo "per ogni" prima o poi finisce) ed è deterministico

Qual è il risultato?

Esempi di Algoritmi (3)

Vogliamo scrivere un algoritmo che possa essere usato da un braccio meccanico per ordinare i mattoncini di Lego mettendo i pezzi blu a sinistra e pezzi gialli a destra.



Esempi di Algoritmi (4)

Primo Tentativo (in pseudo-codice):

```
solleva tutti i mattoncini
posali tutti in ordine giusto
```

- Questo algoritmo è ambiguo in quanto i singoli passi non sono atomici (elementari)
  - il braccio meccanico può sollevare un mattoncino per volta

Secondo Tentativo (in pseudo-codice):

```
finche i mattoncini non sono in ordine giusto {
  scambia un mattoncino blu e uno giallo
}
```

- Questo algoritmo non è deterministico: ad ogni passo ho più scelte di mattoncini da scambiare
- Inoltre potrebbe non terminare

Esempi di Algoritmi (5)

Terzo Tentativo (in pseudo-codice):

```
finche i mattoncini non sono in ordine giusto {
  individua la coppia di mattoncini giallo-blu piu a sinistra
  scambia i due mattoncini
}
```

Questo algoritmo:

- non è ambiguo: in quanto i singoli passi sono semplici
- è deterministico: la scelta della coppia di mattoncini da scambiare è precisamente ben determinata (la più a sinistra)
- ed è facile convincersi che termini, perchè ad ogni iterazione un mattoncino blu si sposta a sinistra e uno giallo verso destra

Questo è un buon algoritmo!

Esempi di Algoritmi (6)

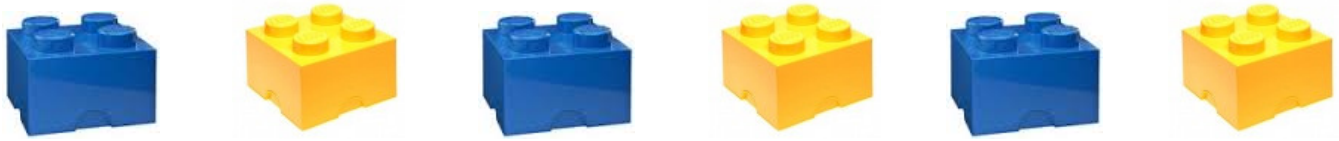
Prova di esecuzione:

```
finche i mattoncini non sono in ordine giusto {
  individua la coppia di mattoncini giallo-blu piu a sinistra
  scambia i due mattoncini
}
```

Input:

Passo 1:

Passo 2:



Passo 3: come continua?

Dagli algoritmi ai programmi

Ora: come dare in pasto un algoritmo al computer?

Il computer parla il linguaggio macchina... è piuttosto difficile esprimere un algoritmo in linguaggio macchina!

Ecco: i linguaggi di programmazione!

- più dettagliati dello pseudo-codice, ma molto più facilmente utilizzabili del linguaggio macchina

```
//programma che implementa l'algoritmo di ordinamento dei mattoncini
import java.util.Scanner;
public class OrdinaMattoncini {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    String[] a = new String[10];
    for (int i=0; i<10; i++) {
      a[i] = input.nextLine();
    }
    while (!Checker.OK(a)) {
      int i=0;
      while (a[i].equals("blu")) i++;
      a[i-1] = "blu"; a[i] = "giallo";
    }
  }
}
```

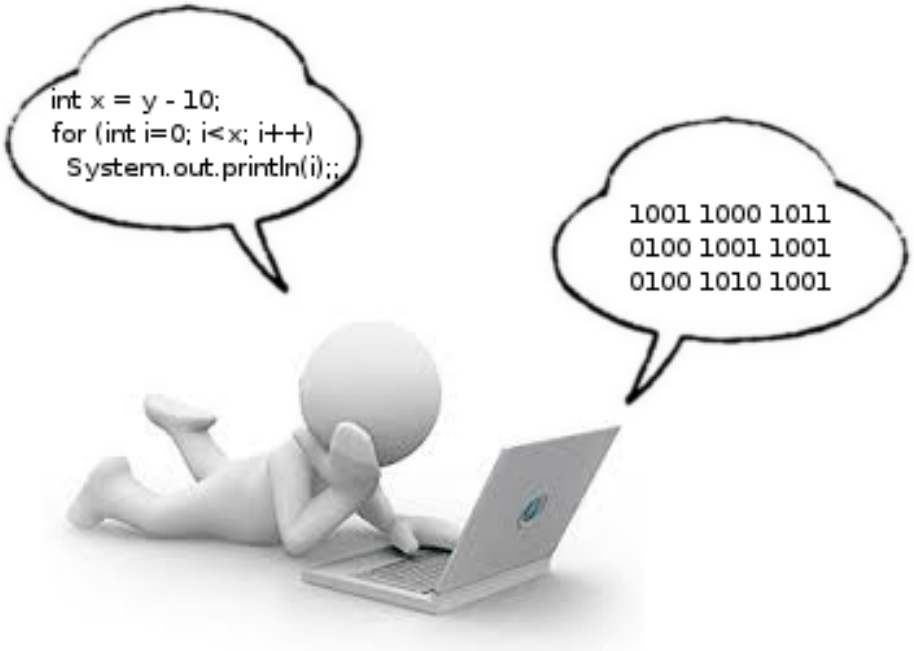
I linguaggi di programmazione

Un linguaggio di programmazione è un "linguaggio formale" che consente di scrivere programmi che realizzano algoritmi

- Un linguaggio formale (a differenza del linguaggio naturale) è un linguaggio con regole sintattiche e semantiche ben precise che rendono i costrutti del linguaggio stesso privi di ambiguità
- I programmi potranno poi essere tradotti in linguaggio macchina per essere eseguiti dal computer
- Essendo il linguaggio specificato su regole ben precise, la traduzione può essere fatta da un altro programma!

Compilazione e interpretazione (1)

Come tradurre un programma dal linguaggio di programmazione in linguaggio macchina?



Compilazione e interpretazione (2)

Pensiamo al mondo reale: come si può comunicare con una persona che parla solo cinese?

Due soluzioni:

- scriviamo una lettera in italiano e la portiamo all'ambasciata cinese per farla tradurre
- troviamo un interprete che faccia una traduzione simultanea

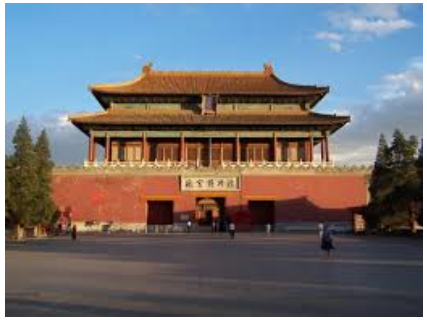


Image courtesy of stockimages / FreeDigitalPhotos.net

Compilazione e interpretazione (3)

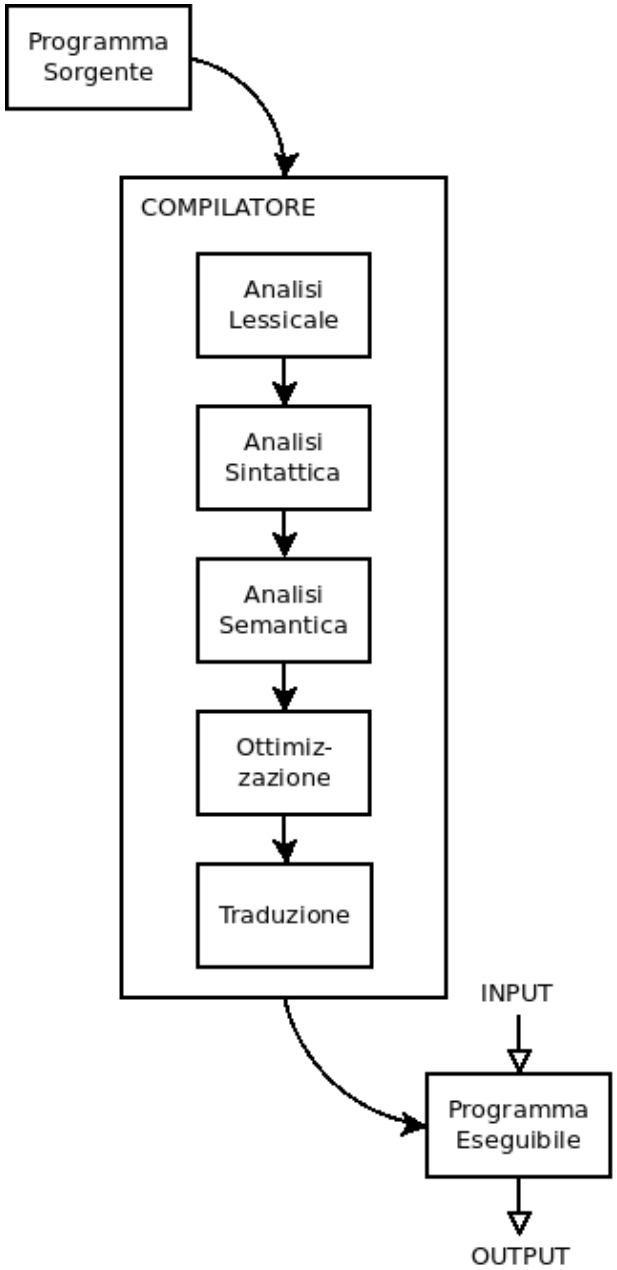
Analogamente, per tradurre un programma (sorgente) in linguaggio macchina si può usare:

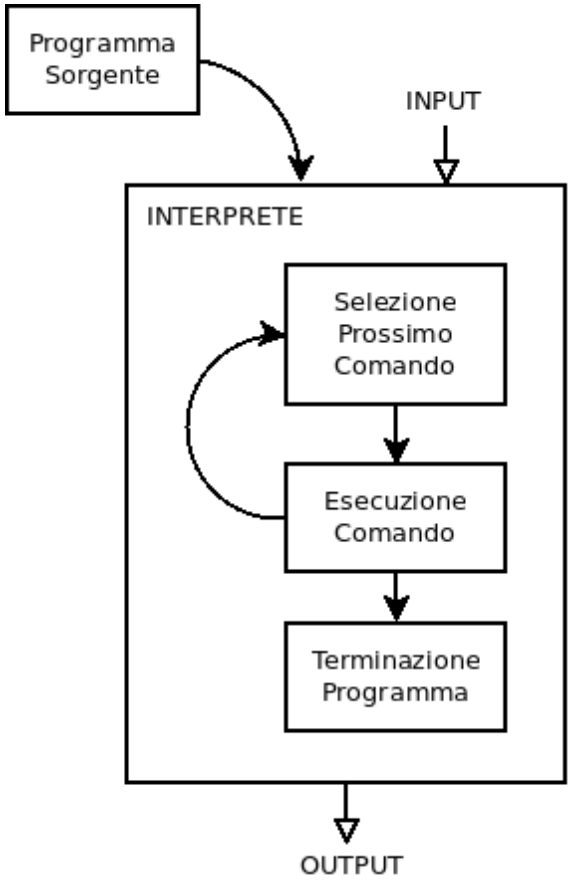
- Un compilatore: ossia un programma che prende in input il programma sorgente e produce in output il corrispondente programma in linguaggio macchina, eseguibile successivamente dal computer
- Un interprete: ossia un programma che prende in input il programma sorgente, traduce un comando per volta e lo esegue man mano.

Esempi:

- Il linguaggio C è un linguaggio compilato
- Il linguaggio JavaScript è un linguaggio interpretato

Compilazione e interpretazione (4)





Compilazione e interpretazione (5)

Vantaggi ⇑ e svantaggi ⇓ della compilazione:

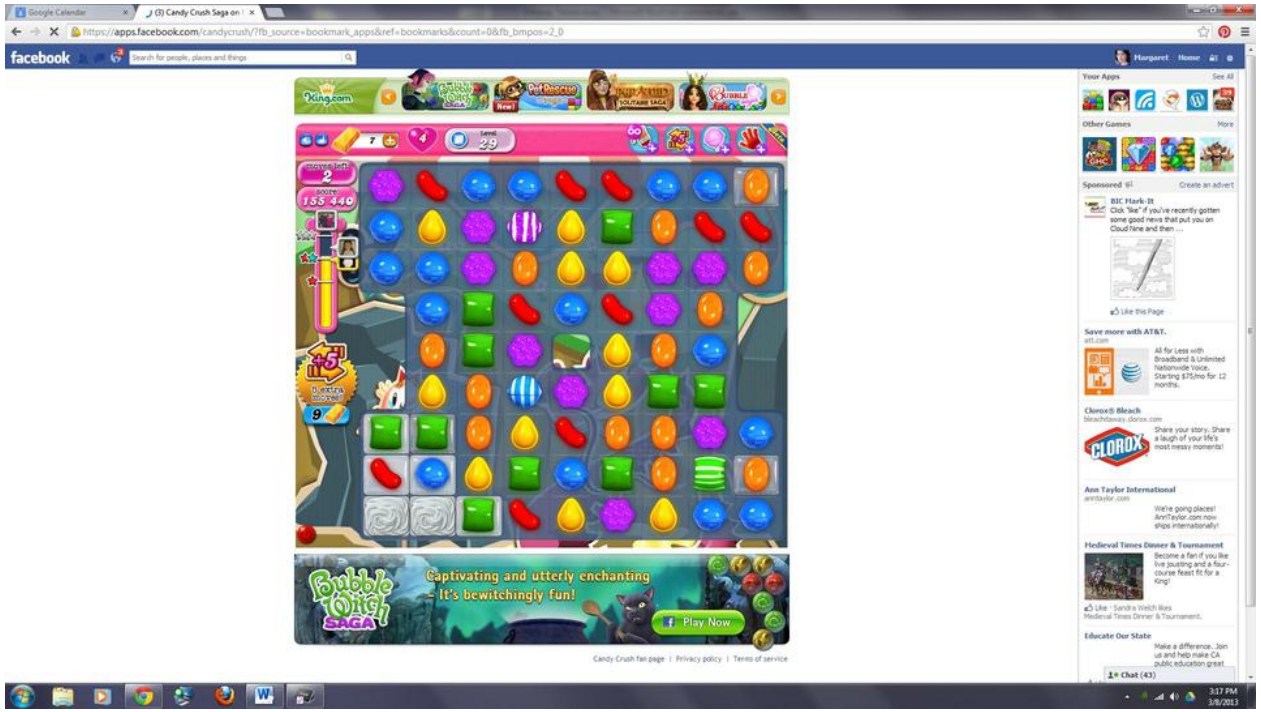
- ⇑ Il programma eseguibile che si ottiene è veloce (direttamente utilizzabile dalla CPU)
- ⇓ Un compilatore produce un eseguibile che funziona su una sola piattaforma hardware (Intel, ARM, ...) e sistema operativo (Windows, Linux, MacOS, ...)

Vantaggi ⇑ e svantaggi ⇓ della interpretazione:

- ⇓ L'esecuzione del programma è rallentata dall'interprete, che deve tradurre ogni comando
- ⇑ Portabilità: lo stesso identico programma può essere eseguito su architetture (Intel, ARM,...) e sistemi operativi (Windows, Linux, MacOS, ...) diversi
- ⇑ Sicurezza: l'interprete può tenere il programma sotto controllo ed evitare che acceda a dati privati o interferisca con gli altri programmi

Applicazioni Web (1)

I siti web moderni sono ormai applicazioni web



Ossia, contengono programmi veri e propri con i quali l'utente interagisce tramite il browser

Applicazioni Web (2)

I linguaggi di programmazione usati nell'ambito del World Wide Web utilizzano solitamente l'interpretazione

- Il browser (o chi per lui – e.g. Flash player) funge da interprete del programma contenuto nelle pagine web

Le proprietà di portabilità e sicurezza dei linguaggi interpretati consentono di:

- eseguire il codice su qualunque computer dotato di un browser, indipendentemente dalla piattaforma hardware e dal sistema operativo
- offrire qualche garanzia che il codice eseguito (sulla cui provenienza l'utente non ha molto controllo) non possa accedere a informazioni private o interferire con gli altri programmi eseguiti nel proprio computer