FONDAMENTI DI INFORMATICA

SECONDA LEZIONE (1/2)

Prof. Alfredo Accattatis

(accattatis@ing.uniroma2.it)

Tutor:

Prof. Vittorio Emanuele Calafiore

(vcalafio@gmail.com)

Prima cosa:

GRAZIE!

Delphi : 150 iscritti

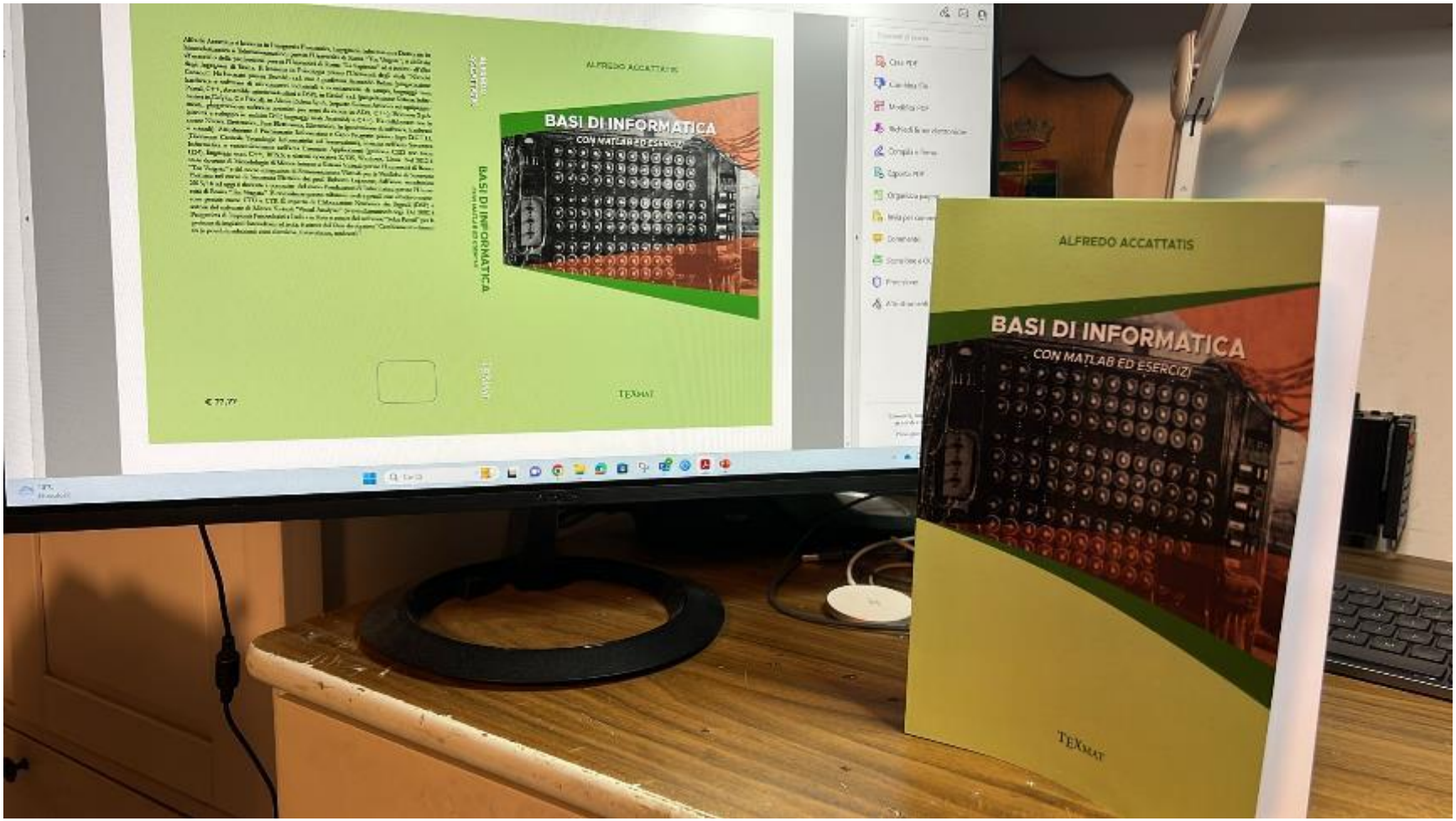
Teams : 161

Domenica 9 marzo ore 17.06

Conferma

- TUTOR per il corso
- Prof. Vittorio Emanuele Calafiore

Libro di testo



Definizione di Informatica (Computer Science)

- Le definizioni sono varie…ma simili….prendiamone una autorevole

Treccani:

- *Scienza che studia l'elaborazione delle informazioni e le sue applicazioni; più precisamente l'Informatica si occupa della **rappresentazione**, **dell'organizzazione** e del trattamento **automatico** della **informazione***

"Computer science is no more about computers than astronomy is about telescopes." (L'informatica non riguarda i computer più di quanto l'astronomia riguardi i telescopi)

E. Dijkstra

Punto della situazione

- (1) Informazione
- (2) Rappresentazione (dell'informazione)
- (3) Organizzazione (dell'informazione)
- (4) Trattamento automatico(dell'informazione)
  - Algoritmi automatizzati

Algoritmo: sequenza di passi elementari

per risolvere un problema

Modalità tipicamente umana per eseguire un «compito»

Esempio tipico: ricetta di cucina!

Procedure "automatiche"

220,70,220,100,70,110,210,80

rappresentazione

+

(1)Verifica: ***se*** (altezza(A)<altezza(porta)) ***e*** (larghezza(A)<larghezza(porta))

Se (1) è VERO **ok**; *altrimenti* verifica la faccia B (passo successivo);

(2)Verifica: ***se*** (altezza(B)<altezza(porta)) ***e*** (larghezza(B)<larghezza(porta))

Se (2) è VERO **ok**; *altrimenti* verifica faccia C (passo successivo);

(3)Verifica: ***se*** (altezza(C)<altezza(porta)) ***e*** (larghezza(C)<larghezza(porta))

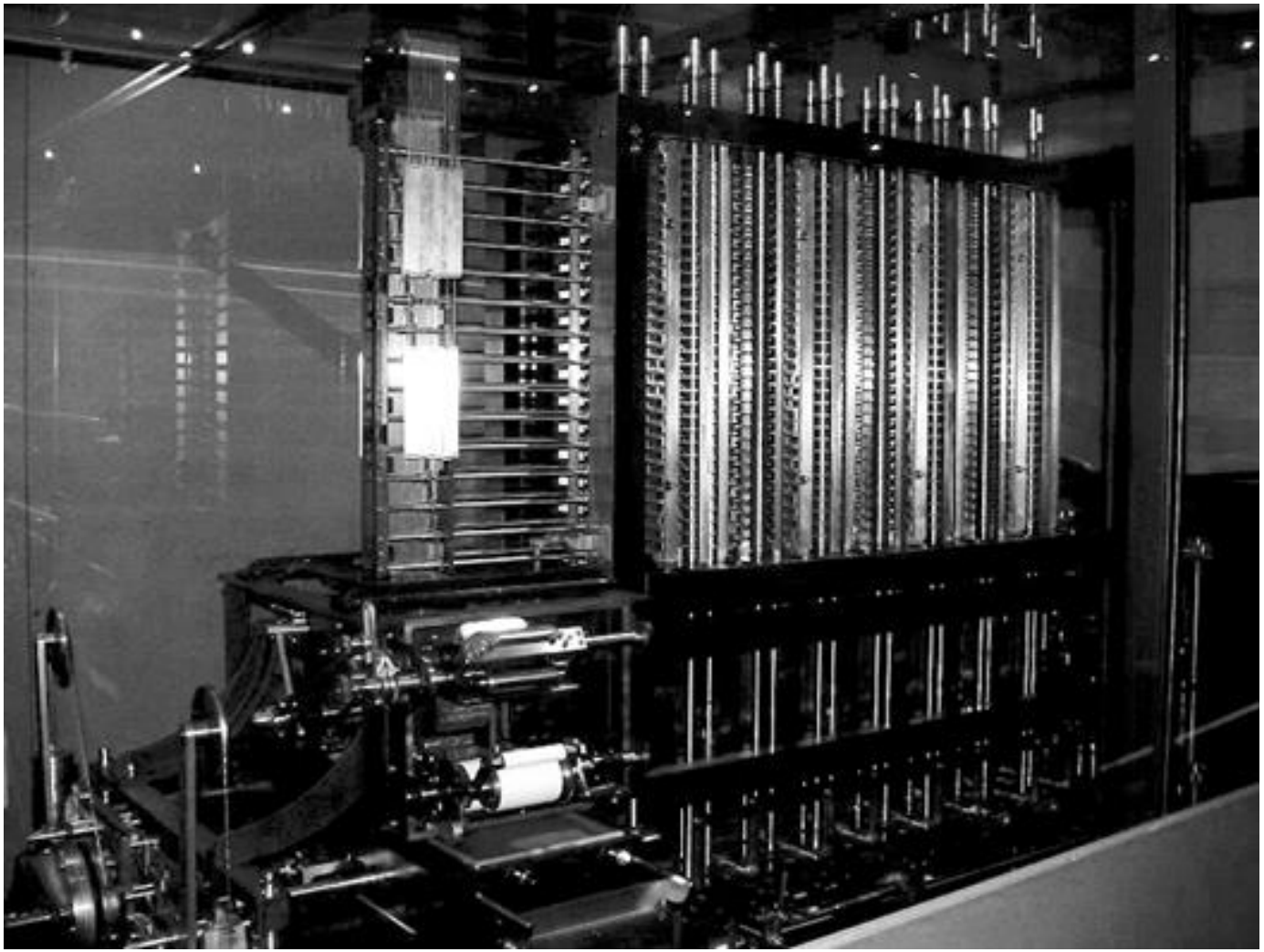
Se (3) è VERO **ok**; *altrimenti* "*la scatola non passa*";

Procedura (algoritmo)

- "La scienza della rappresentazione dell'informazione ed elaborazione automatica". Automatico?

Di macchina, meccanismo o dispositivo che, regolato opportunamente, è capace di compiere determinate operazioni o lavorazioni, per lo più ripetute in serie, senza il diretto intervento dell'uomo

Nascita dell'elaboratore?



- La macchina analitica di Babbage (1791 – 1871)
- Ideata nel 1853
- Realizzata nel 1991 (Museo della scienza di Londra)

Il primo programma: Si basava su un algoritmo per generare numeri di Bernoulli

Ada Lovelace = Augusta Ada Byron

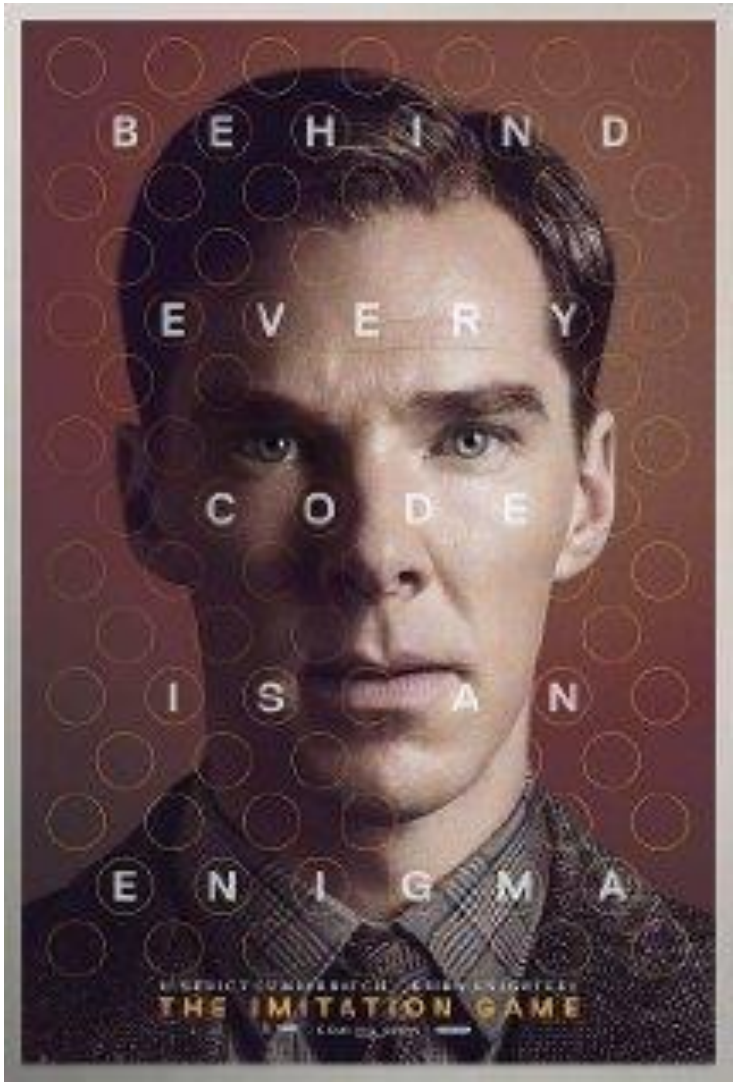
(1815 – 1852)

Matematica Inglese

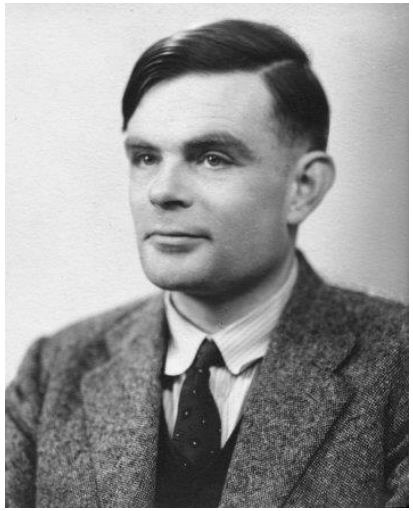
Linguaggio di programmazione moderno

con nome dato in suo onore:

ADA (applicazioni militari)



Durante la seconda querra mondiale il matematico Alan Turing dette un contributo fondamentale alla "decifratura" del codice Enigma usato dai tedeschi. Turing è considerato uno dei padri fondatori della moderna informatica (macchina di Turing, algoritmo, test di Turing etc.)



http://en.wikipedia.org/wiki/Alan_Turing

L'informatica, dunque, ha sostanzialmente "due anime":

Tecnologica:

comprende lo studio dei calcolatori e dei sistemi che li utilizzano;

Metodologica:

studia i metodi per la soluzione di problemi e la gestione delle informazioni (es. algoritmi, basi dati, dbms, strutture dati etc.)

In particolare il **Calcolatore** o **Elaboratore** o **Computer** è uno:

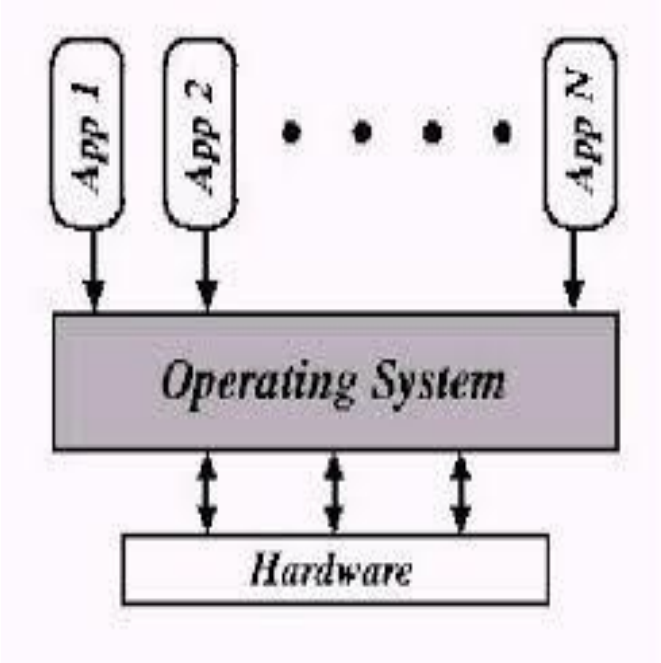
"*Strumento programmabile per rappresentare, memorizzare ed elaborare informazioni*"

- Il computer, inteso «brutalmente» come macchina per effettuare meccanicamente (ed automaticamente) operazioni di calcolo (tali che ad una variabile di *input* dell'utente si produce un corrispondente *output* come effetto di un processo dei dati immessi, e determinato da una regola matematica o logica) è molto antico, probabilmente dal I secolo a.C. (fonte: wikipedia).
- Il computer in senso moderno, inteso come macchina programmabile, nasce con la macchina analitica di Charles Babbage nel 1833.
- Per chi vuole approfondire può iniziare dando una prima occhiata in rete.
- Esempio: https://it.wikipedia.org/wiki/Storia_del_computer

Hardware: insieme dei circuiti fisici comprendenti la memoria, la logica di calcolo (microprocessori, unità grafiche, co-processori etc.) le periferiche di ingresso uscita (tastiere, monitor, stampanti, mouse, penne ottiche etc.);

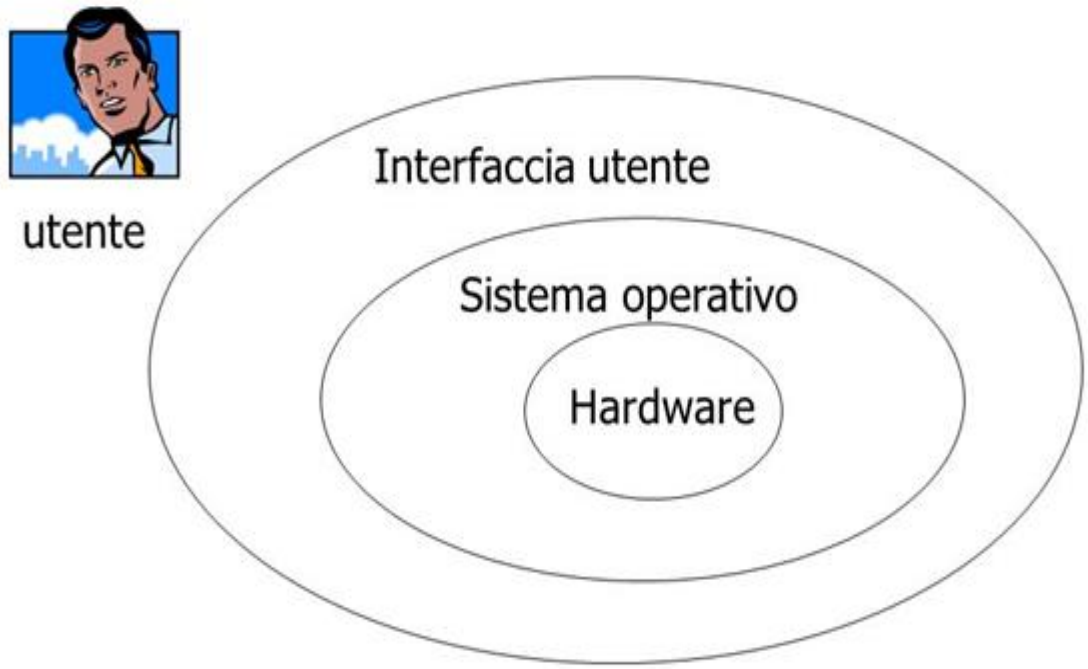
Software di base o Sistema Operativo

Software applicativo: è l'insieme dei "programmi" o "app" che mettono in grado l'hardware di svolgere dei compiti "utili"

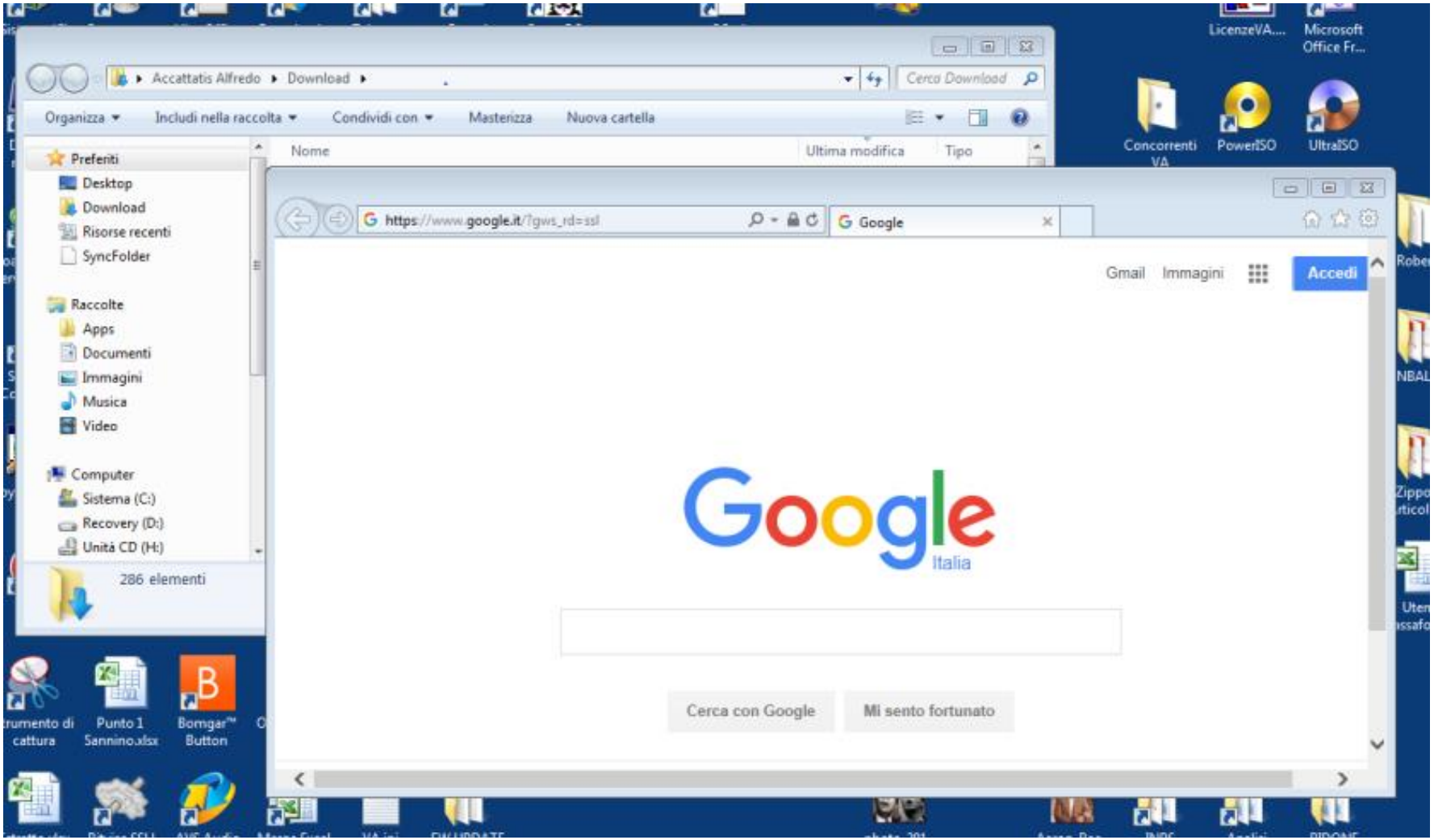


Interfaccia utente:

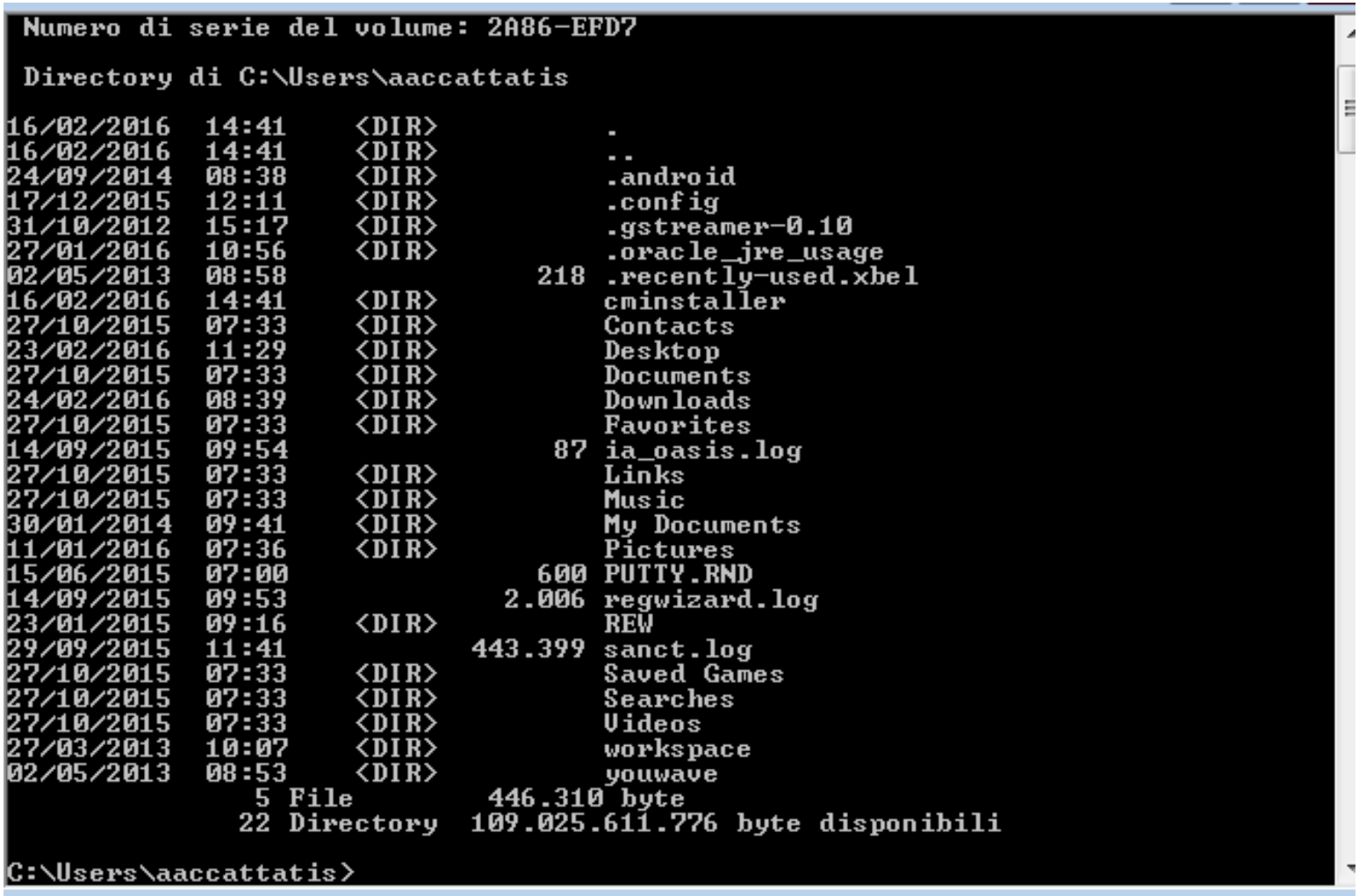
modalità di interazione con la macchina



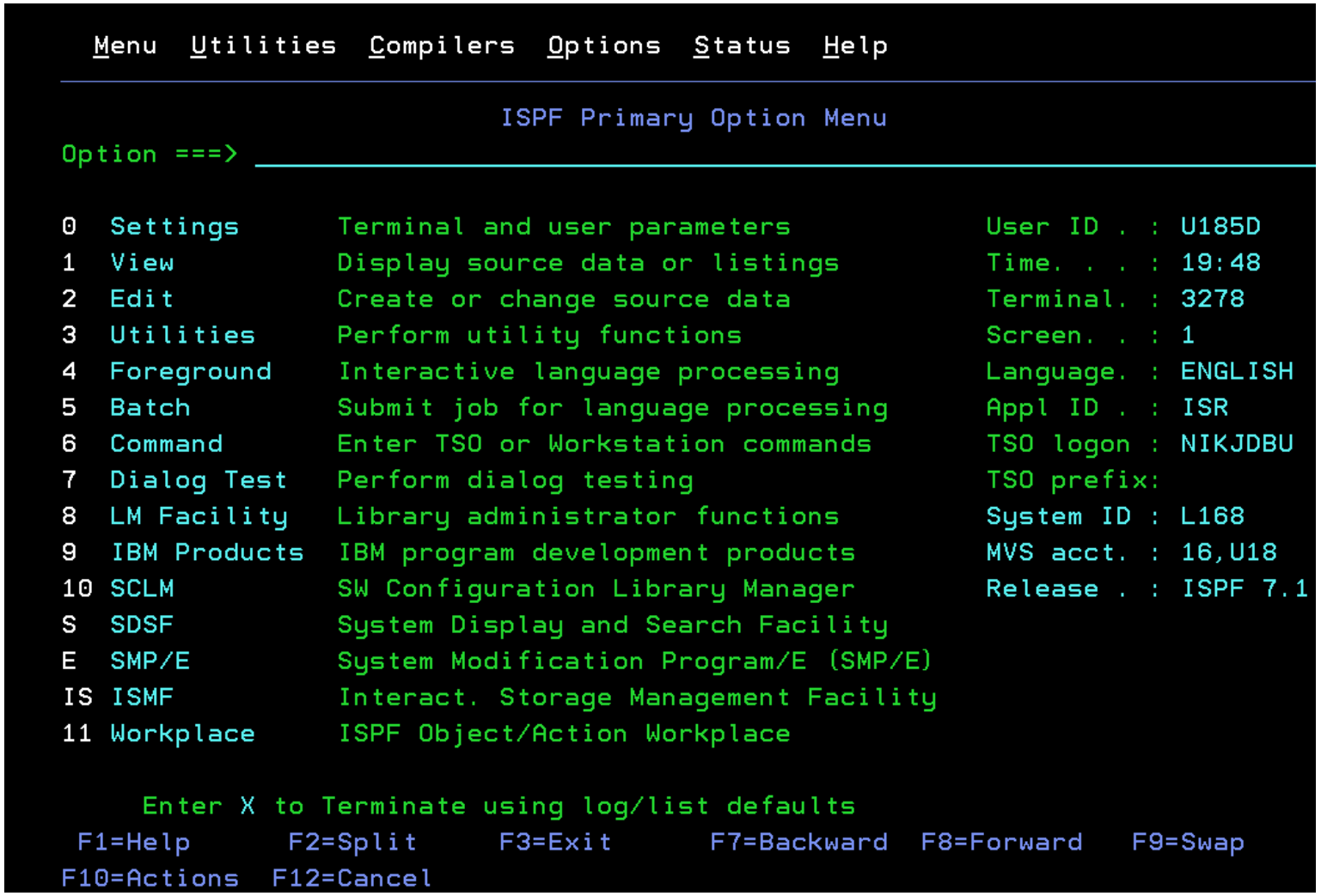
Windows (interfaccia a finestre, GUI)



DOS (interfaccia a caratteri, riga di comando)



Z/OS (interfaccia a caratteri, riga di comando): MAINFRAME



-Linux

- Mac/OS

- IOS

- Android

- Z/OS

- OSE

- VxWorks

Fare molte cose in parallelo

Multitasking

Multi Threading

Interfacce potenti

Uso in dispositivi miniaturizzati

DSP,microcontrollori

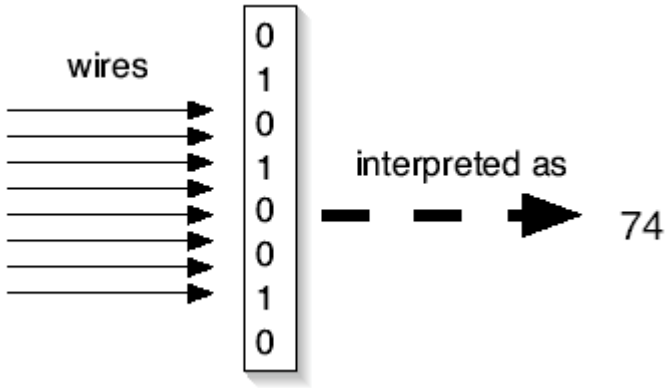
(Arduino, Raspberry)

Punto della situazione

- (1) Informazione
- (2) Rappresentazione (dell'informazione)
- (3) Organizzazione (dell'informazione)
- (4) Trattamento automatico(dell'informazione)
  - Algoritmi automatizzati
  - SOFTWARE

Concetto chiave: codifica (rappresentare l'informazione)

- Possiamo *interpretare* gli 0 e 1 contenuti nella in memoria del computer come vogliamo.
  - Possiamo considerarli numeri.
  - Possiamo *codificare altre* informazioni in questi numeri



- Persino la nozione che il computer "comprende" solo numeri è una interpretazione
  - Noi codifichiamo i dati sui "fili" (conduttori) come 0 e 1 (livelli di tensione) a gruppi di otto definisce un *byte*
  - …che possiamo, di converso, interpretare come numeri decimali…

A proposito…perché interpretiamo questa sequenza di bit come il numero 74 ?

La codifica può essere "stratificata" in un qualsiasi numero di livelli

- Codifica ASCII dei caratteri
  - "A" codificato come 65
  - "B" codificato come 66
  - …
  - Se esiste un byte che contiene il numero 65, e decidiamo che è un carattere…voilà…diventa una "A"!

- Possiamo costruire una "serie" di numeri per definire un testo
  - "77, 97, 114, 107" sta per "Mark"
  - "60, 97, 32, 104, 114, 101, 102, 61" sta per:
    " < a h r e f = "

    (**<a href=** è un frammento di linguaggio HTML)

Multimedia e unimedia

- Ma lo stesso byte che contiene il valore 65 potrebbe essere interpretato come…
  - Suono (una piccolissima parte…p. es. 1/44100 di secondo)
  - La quantità di colore rosso di un singolo punto di una immagine
  - La quantità di colore rosso di un singolo punto di una immagine che è un singolo "fotogramma" di un film
- Usiamo software e hardware per gestire tutti questi "strati"
  - Come decidere il "significato" di un numero e come *organizzare* i numeri per rappresentare i dati desiderati?
  - Tramite le *strutture dati*
- I dati da organizzare in "strutture" possono essere numerosi
  - Per rappresentare tutti i punti di una schermata tipica possono volerci milioni di byte: esempio : 1280x800 = 102400 punti…normalmente si usano 24 bit (3 byte) per punto ossia 3*1024000 = 3072000 byte
  - Ogni secondo di musica in un CD occupa 44100x2 byte

Punto della situazione

- (1) Informazione
- (2) Rappresentazione (dell'informazione)
- (3) Organizzazione (dell'informazione)
- (4) Trattamento automatico (dell'informazione)
  - Algoritmi automatizzati

ESEMPIO: vedremo i VETTORI, MATRICI,

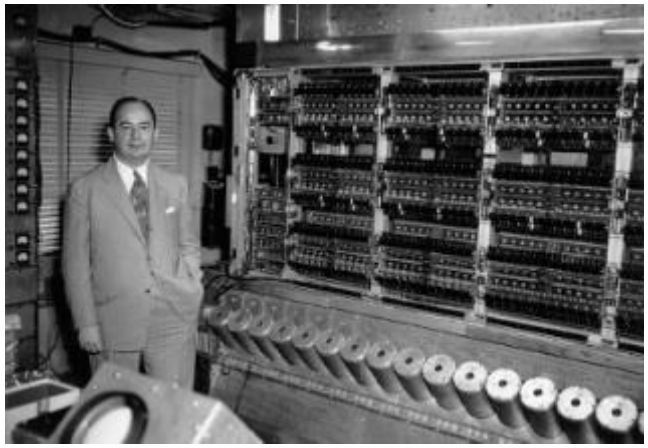
File, Structure, Cell Array

Segnali elettrici a due livelli (es. 5V = 1, 0V=0) in circuiti fisici

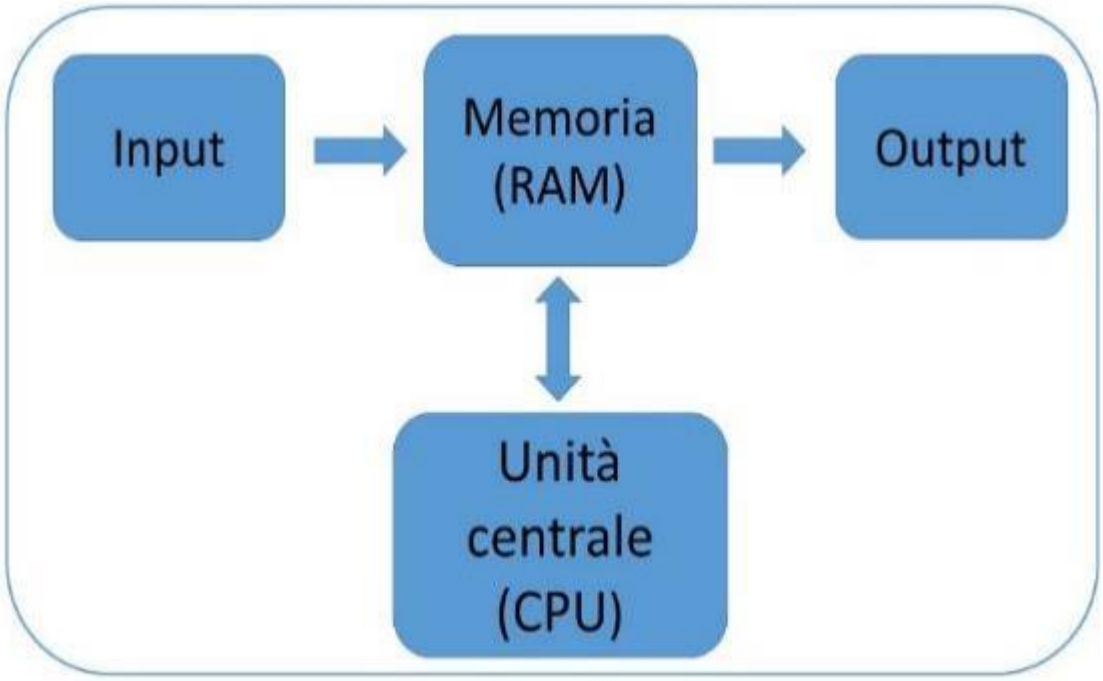
Software

- Consente di automatizzare gli algoritmi
- E una sequenza di istruzioni (comandi) che agiscono sui dati (informazioni)
- Gli algoritmi sono espressi da una sequenza di passi
- I passi sono costituiti da una o più istruzioni
- Le istruzioni fanno parte di un set predefinito a livello hardware dalla macchina (vedremo a breve la struttura generale di una macchina hardware)
- Analogia: spartito musicale (software) musicista (hardware)

Architettura di Von Neumann

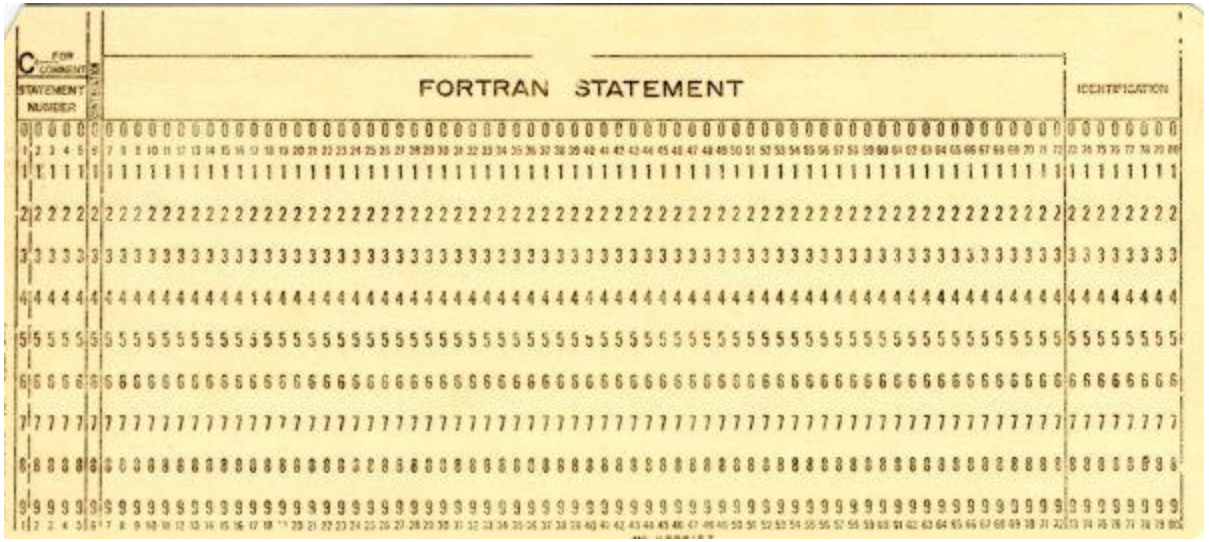


Schema base

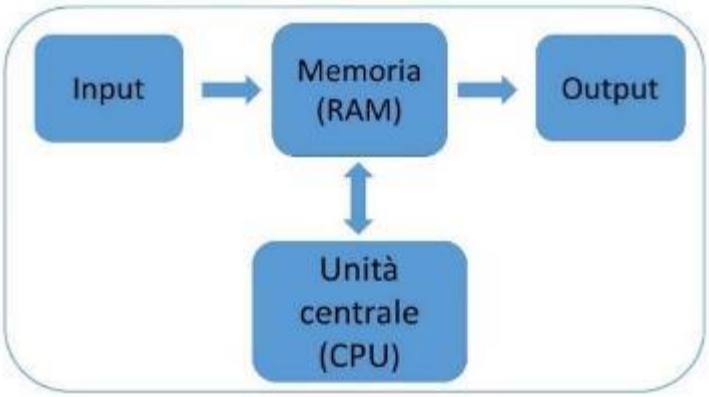


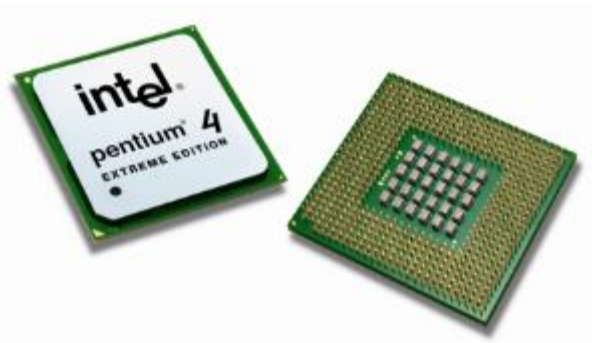
Il sistema di Input dei dati

- Unità di ingresso dati (attualmente tastiere, mouse, etc): un esempio delle origini



Von Neumann



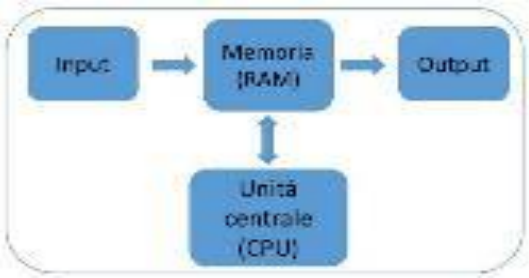


- Memoria Centrale = RAM = Random Access Memory: vengono immagazzinate istruzioni e dati. *Perde* i dati in mancanza di alimentazione.
- Input/Output : ingresso e uscita dati.
- CPU = Central Processing Unit : effettua fisicamente elaborazioni algebriche e logiche
- Frecce: rappresentano il flusso dei dati, che viaggiano sui cd. BUS, ossia dei dispositivi (tipicamente uno o più conduttori) che trasportano i segnali elettrici tra i vari organi

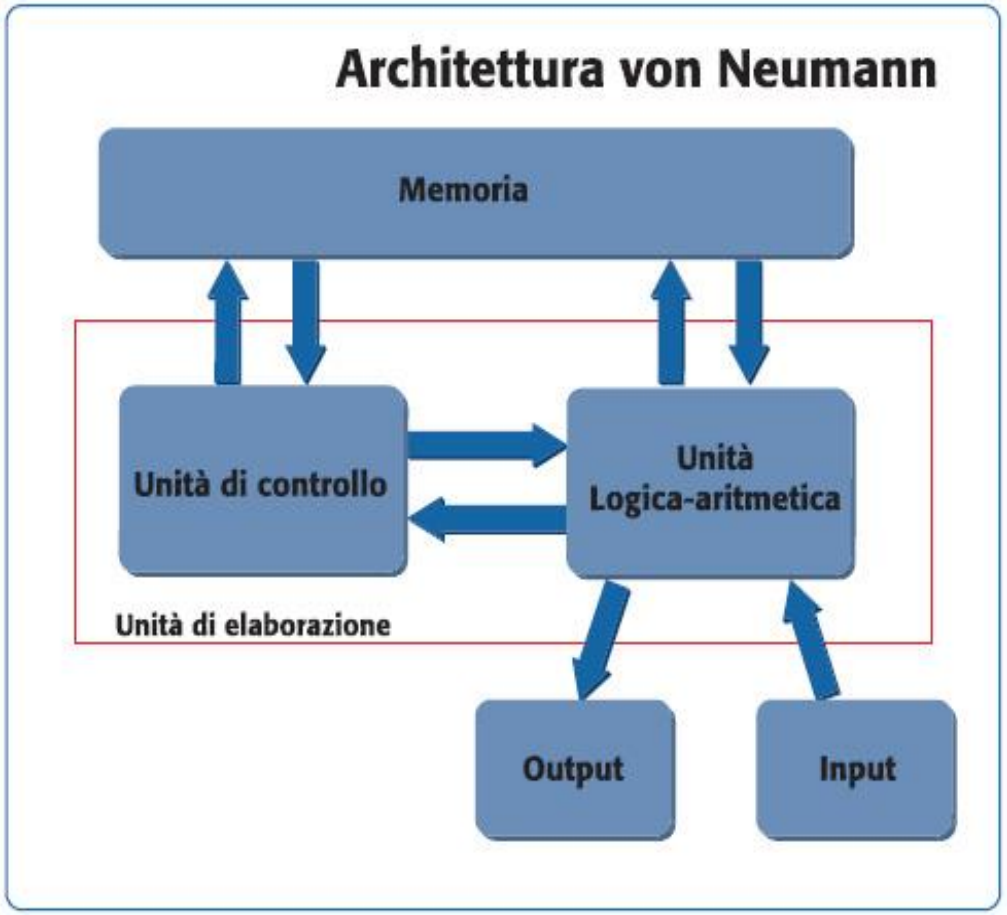
Letture suggerite

http://en.wikipedia.org/wiki/John_von_Neumann

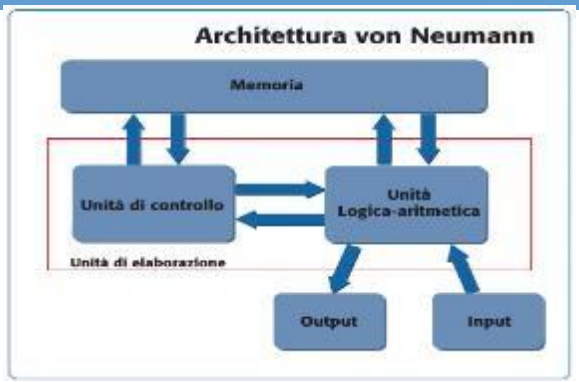
Andiamo in maggiore dettaglio



- Come detto, esegue calcoli e confronti: *Central Processing Unit (CPU) – Unità di elaborazione*.
- La CPU "comunica" con la memoria:
  - Pensate la memoria come una sequenza di "mailbox" ognuna formata da un byte (o multipli), e con un suo *indirizzo*
- Input è :Tastiera, Mouse, Microfono, Touchscreen, Sensori etc….
- Output è: Monitor, Display, Altoparlanti, Attuatori …
- L'*hard disk* (memoria di massa: è *input, output e memoria*) fornisce una grande capacità di memoria rispetto alla RAM, ma è enormemente più lento…anche se SSD



Von Neumann



- Unità di controllo: genera il segnale di clock, e predispone il flusso di segnali per eseguire l'istruzione nella unità logica-aritmetica
- L'istruzione è eseguita nella unità logica-aritmetica
- I dati sono prelevati e depositati in memoria
- I dati sono prelevati e depositati nelle unità di input e output
- Possibilità di collegamento diretto tra memoria e organi di Input Output

Punto della situazione (…repetita juvant)

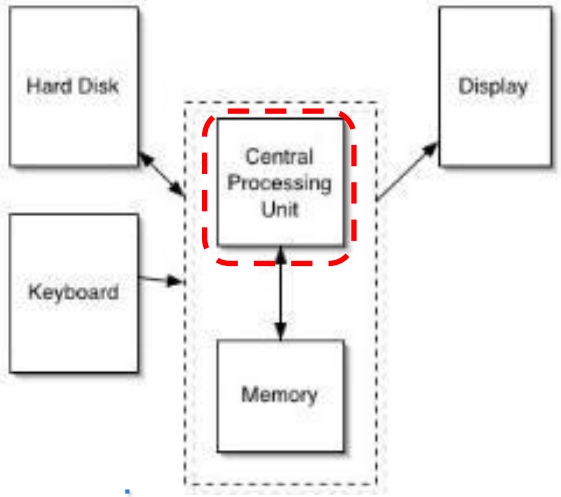
- Algoritmi: lavorano su astrazioni della realtà (rappresentazioni)
  - … ed ora dovremmo sapere di cosa stiamo parlando
- Computer: eseguono gli algoritmi tramite l'esecuzione del software, ossia li "automatizzano"
  - …senza sapere cosa fanno

Che cosa «capiscono» i computer

- Importante: i computer **non** sono affatto multimediali!
  - ..sono *unimedia*
    - (parole di Nicholas Negroponte, fondatore del "MIT Media Lab")
  - *Ogni cosa è costituita* da 0 e 1 …solo NUMERI

- I computer sono *esageratamente* stupidi
  - Gli unici dati che capiscono sono 0 e 1
  - Con queste cifre (0 e 1) possono fare solo cose molto semplici
    - Spostare un valore
    - Sommare, moltiplicare, sottrarre, dividere
    - Effettuare confronti, e se uno è minore di un altro andare ad un passo invece che un'altro
  - Ma…una enormità di cose elementari può fare cose incredibili!
  - Reti neurali

Central Processing Unit (CPU)



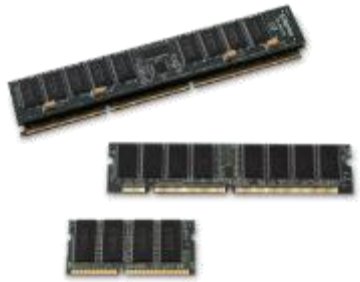
- Esegue le operazioni elencate in un algoritmo
  - L'algoritmo è «contenuto» in memoria (memorizzato)
  - Macchine "a programma memorizzato"

- Cicli di esecuzione
  1. Fetch (Prelievo) di una operazione (istruzione) dalla memoria
     1. Ogni operazione è codificata "in hardware" oppure a "microcodice"
  2. Decode (Decodifica) istruzione
  3. Execute (Esecuzione) istruzione

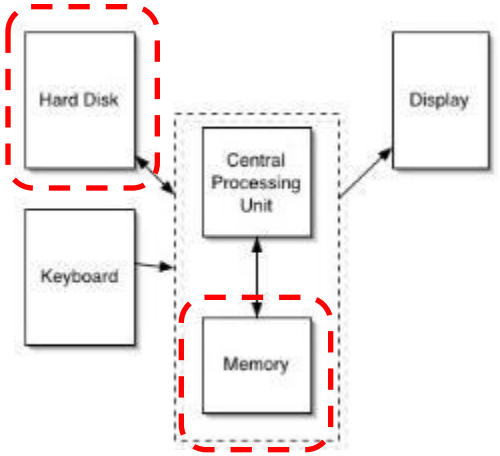
- Ogni CPU è caratterizzata da un suo insieme di istruzioni (*linguaggio macchina*) :

```
0100 0000 0000 1000
0100 0000 0000 1001
0000 0000 0000 1000
...
```

Memoria



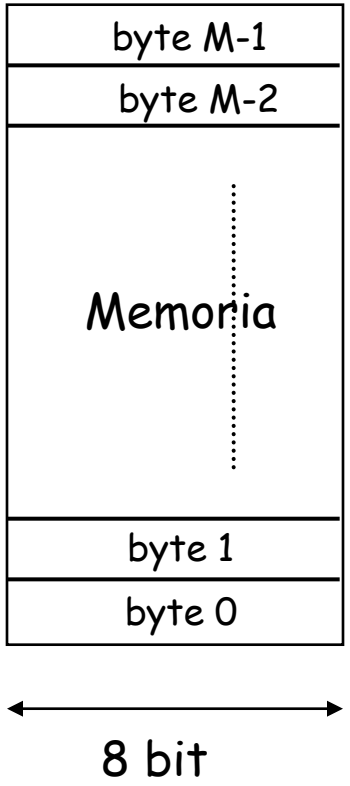
- Memoria principale: *central memory*
  - Memorizza DATI e ISTRUZIONI dei programmi in esecuzione
    - …in formato binario (base 2)
  - volatile
  - "random access" (RAM)
    - Tempo di accesso costante
    - SRAM, DRAM, etc.
  - veloce (~10..100 nsec), costosa
  - Capacità limitata (fino a qualche Gigabyte)
  - Struttura gerarchica
    - Cache di primo e secondo livello, …

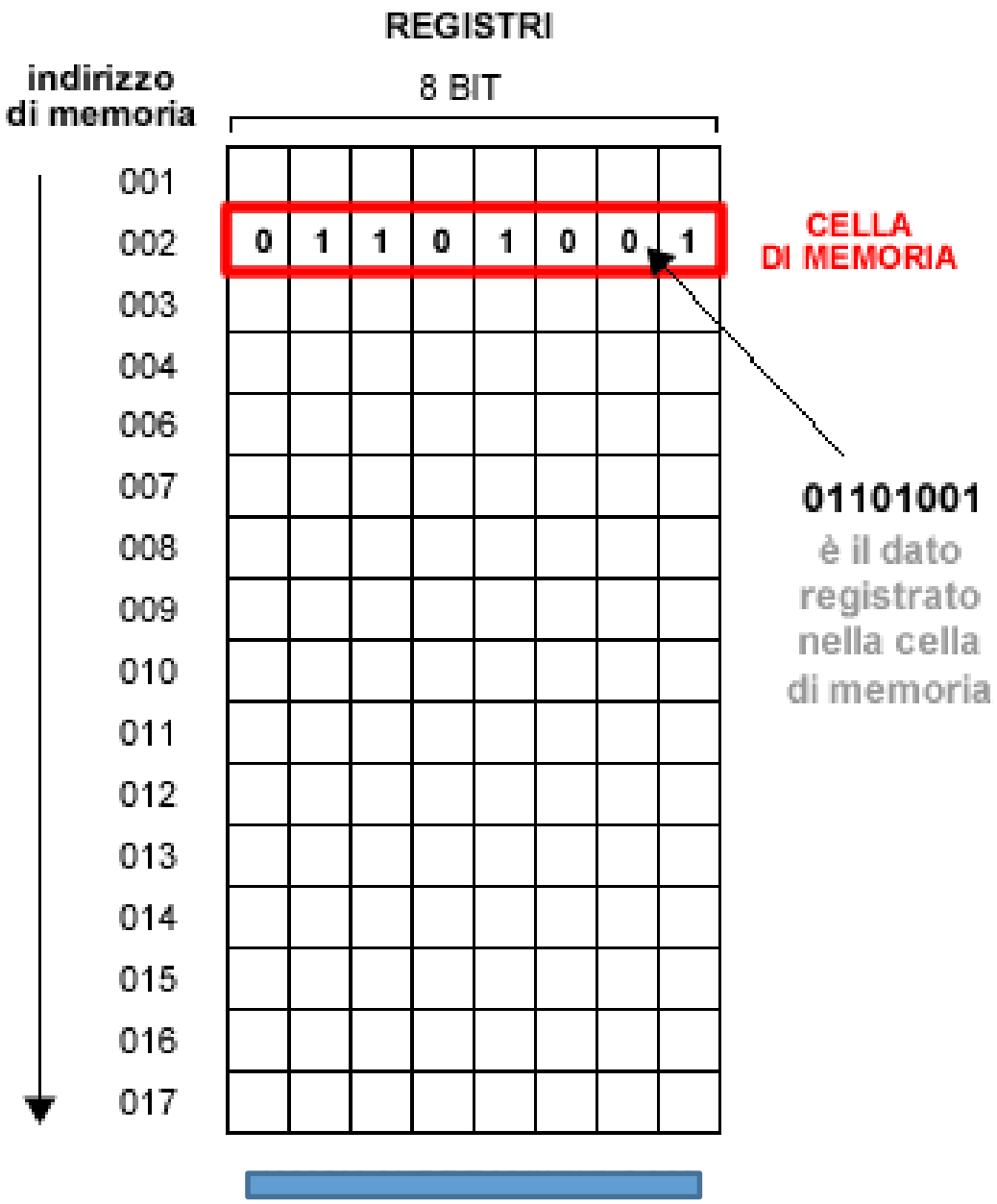


- Memoria secondaria (di massa) : hard disk, CD, etc..
  - Non volatile
  - Grande capacità (centinaia di Gigabytes)
  - Lenta (~ms e più), economica

Memoria

- Costituita da *celle* (*locazioni*), ognuna a sua volta costituita da un certo numero di BIT
  - Ogni elemento binario (**un bit**) memorizza (RAPPRESENTA) solo due valori : 0 o 1
  - standard: una cella = **1 byte (8 bit)**
  - **word** sono gruppi di byte: 2, 4, 8 bytes = 16, 32, 64 bit
- Ogni cella è associata ad un *indirizzo* nel range [0,1,…,M-1]
  - M: dimensioni della memoria *(e.g. 4 GB, 2 MB, 512 kB)*
  - La memoria principale può vedersi come un "vettore" di byte
- CPU legge/scrive il contenuto delle celle specificando l'indirizzo
  - read: preleva il contenuto di una cella
  - write: modifica il contenuto di una cella
  - Indirizzo a m bit ⇒ "*address space*" di dimensioni 2^m
    - Non obbligatorio: $M = 2^m$




REGISTRI
indirizzo di memoria
8 BIT
001
002
003
004
006
007
008
009
010
011
012
013
014
015
016
017
0 1 1 0 1 0 0 1
CELLA DI MEMORIA
01101001
è il dato registrato nella cella di memoria

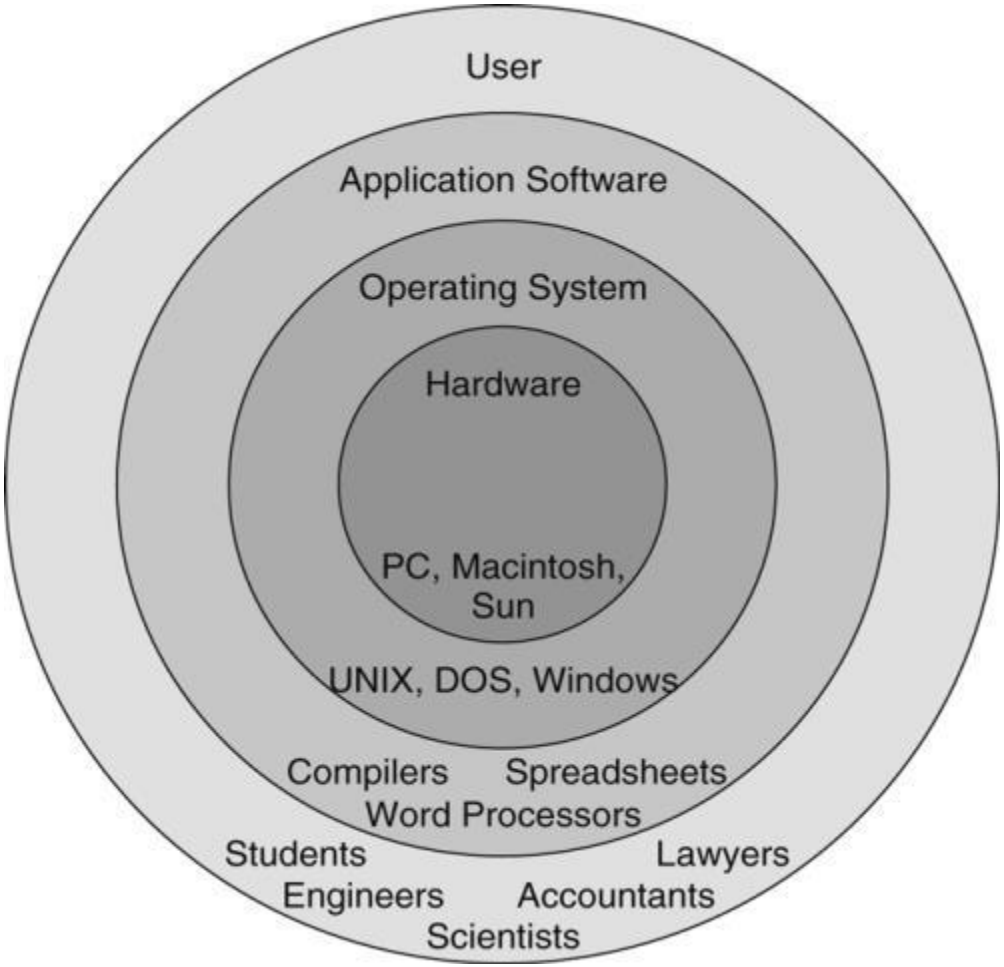
Facciamo chiarezza sulle unità di misura della memoria:

- 1 kB ossia 1 chilobyte = 2^{10} byte = 1 024 byte
- k sempre in minuscolo
- kb è il CHILOBIT (sequenza dati BIT a BIT)
- Nel moderno SI 1 chilobyte = 10^3 byte = 1000 byte.
- Allora il KiB multiplo di 2 si chiama kbibyte (termine poco noto e poco diffuso)
- https://web.archive.org/web/20100529022053/http://www.iec.ch/zone/si/si_bytes.htm

Multipli del byte

Prefissi SI			Prefissi binari		
Nome	**Simbolo**	**Multiplo**	**Nome**	**Simbolo**	**Multiplo**
chilobyte	kB	10^3	kibibyte	KiB	2^{10}
megabyte	MB	10^6	mebibyte	MiB	2^{20}
gigabyte	GB	10^9	gibibyte	GiB	2^{30}
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}

Architettura di un sistema informatico



L'architettura di un sistema informatico descrive come interagiscono hardware e software.

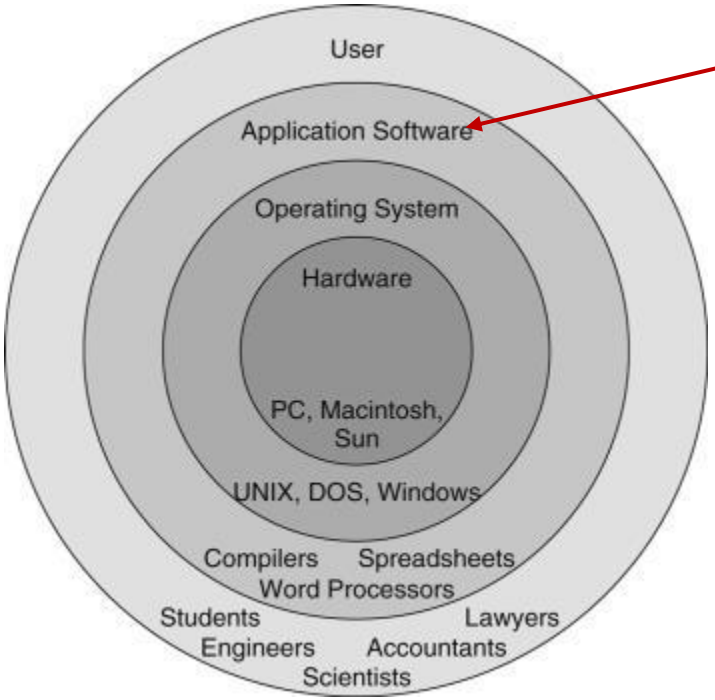
Ciascuno strato (livello di astrazione) *aggiunge* nuove funzionalità al sistema.

Linguaggi di programmazione

- Il linguaggio di programmazione permette di istruire il computer alla soluzione di problemi (algoritmo). Crea nuove funzionalità:

  App, pacchetto software, …

  Il linguaggio di programmazione è a sua volta un'App



Tipologie di linguaggi di programmazione:

- Prima generazione (I): linguaggi macchina
- Seconda generazione (II): linguaggio assembly
- Terza generazione (III): C,C++, fortran, Basic
- Quarta generazione (IV): Java, Matlab

Linguaggi di programmazione e livelli di astrazione

- Si usano linguaggi diversi per differenti livelli di astrazione
- Linguaggio macchina (I)

```
0100 0000 0000 1100
0101 0001 0000 1001
0110 0110 0000 1000
```

Sequenza di istruzioni binarie eseguibili direttamente dalla CPU

- Linguaggio assembly (II)

```
0100 0000 0000 1100 -> LOAD X
0101 0001 0000 1001 -> ADD 9,Y
0110 0110 0000 1000 -> STORE 8,Z
```

Istruzioni in corrispondenza diretta (1-a-1) con le istruzioni binarie ma espresse in formato simbolico

- Linguaggio di alto livello (III, IV)

```
def fun():
  a = 0;
  print  a+5
```

Machine independent: non è legato al tipo di hardware e sistema operativo

Linguaggi di programmazione e livelli di astrazione

Istruzioni in corrispondenza diretta (1-a-1) con le istruzioni binarie ma espresse in formato simbolico

Linguaggio assembly (II)

```
0100 0000 0000 1100 -> LOAD X
0101 0001 0000 1001 -> ADD 9,Y
0110 0110 0000 1000 -> STORE 8,Z
```

Per capirci meglio:

0100 0000 0000 significa LOAD (carica dati) da organi di input (es.: tastiera)

1100 significa REGISTRO X

Quindi: trasferisci il valore del tasto premuto nella variabile X

0101 0001 0000 significa ADD (aggiungi) il valore nel registro interno Y

1001 è il numero che vogliamo aggiungere a Y, ossia 9

Quindi: aggiungi il valore 9 alla variabile Y

Linguaggi di programmazione e livelli di astrazione

Machine independent: non è legato al tipo di hardware e sistema operativo

- Linguaggio di alto livello (III, IV)

```
def fun():
  a = 0;
  print  a+5

Istruzione print a+5 => MOLTE istruzioni assembly
Per esempio:
0100 0001 0000 1100 0101 0001
0111 0101 0000 1001 1110 1111
1110 0110 0010 1000 etc

a + 5 sarà tradotta con ADD (già vista) dunque:
0101 0001 0000 0101 -> ADD 5,Y
```

Linguaggio di programmazione

- Formalizzato da
  - Alfabeto
  - Grammatica (vocabolario)
  - Sintassi (sequenze)
  - Semantica (significato)

  Vocabolario
  - Parole chiave (Keywords)
  - Variabili (Variables)

C++

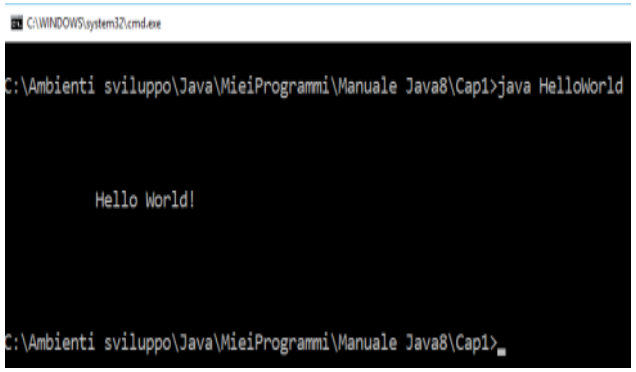
```
#include <iostream.h>

main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

Java

```
class HelloWorld {
  static public void main( String args[] ) {
    System.out.println( "Hello World!" );
  }
}
```

- Una frase in linguaggio naturale (soggetto-verbo-complemento) può essere grammaticalmente e sintatticamente corretta ma non avere significato:

  L'erba scrive il gatto



Linguaggio di programmazione

La frase:

Il gatto mangia il topo

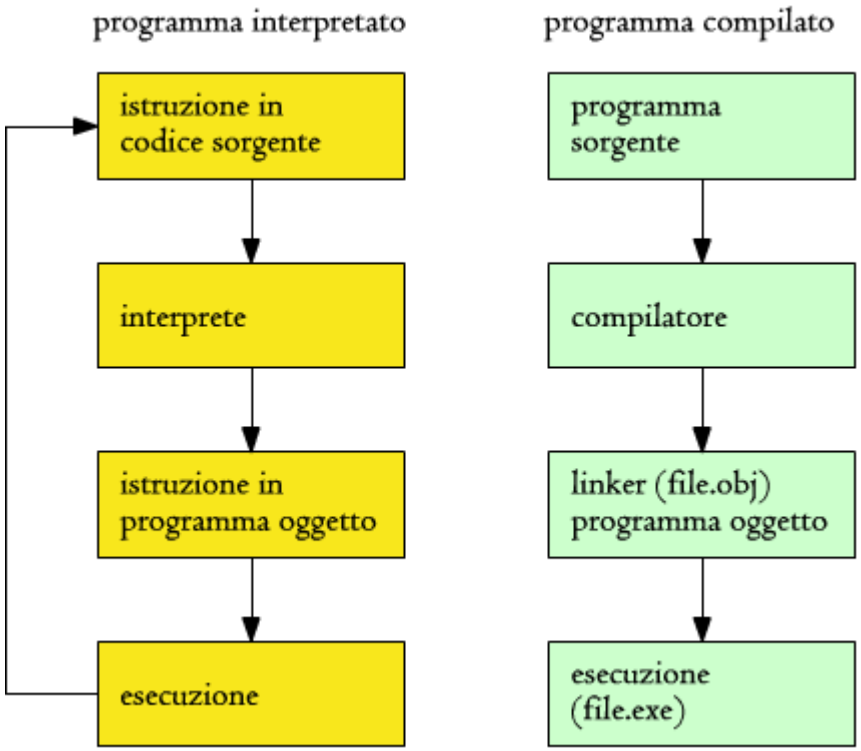
E' invece corretta grammaticalmente, sintatticamente ed anche semanticamente.

Come si scrive un programma?

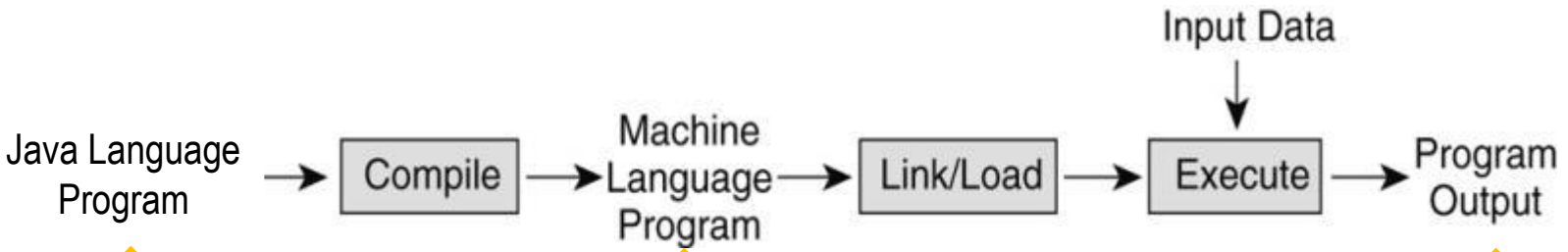
- Si codifica l'algoritmo con un linguaggio
- Esempio: Matlab
- Esempio: Linguaggio macchina
- Il programma scritto si chiama codice sorgente

Due tipi di modalità:

- Interprete (una riga alla volta)
- Compilatore (tutto il programma: genera .exe)



Sequenza di realizzazione di un programma



Progettazione e stesura del codice sorgente

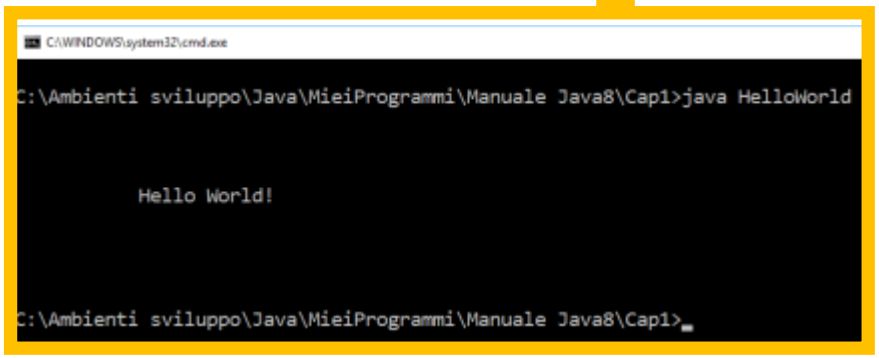
Java

```
class HelloWorld {
  static public void main( String args[] ) {
    System.out.println( "Hello World!" );
  }
}
```

0100 0000
0000 1000

0100 0000
0000 1001

0000 0000
0000 1000



MATLAB

- Principalmente INTERPRETATO
- Necessità di avere installato Matlab
- Compilazione difficoltosa, ma possibile

Usiamo Matlab: primi rudimenti della scrittura dei programmi

Si parlerà di…

- Variabili e assegnazioni di valori
- Caratteri e codifica
- Tipi di dati numerici
  - Interi
  - Reali: Floating point (numeri in virgola mobile)
  - Cenni al tipo “carattere” e “vettore”
- Espressioni numeriche

Usare Matlab/Octave

- Installazione
- Lancio Matlab oppure Octave