

Cos'è la programmazione

Informatica@DSS 2019/2020 — Il canale

Massimo Lauria <massimo.lauria@uniroma1.it>
<https://massimolauria.net/courses/informatica2019/>

Informatica??

Introduzione all'informatica

In inglese “computer science”

- È una disciplina principalmente matematica
- Nasce come derivazione della logica
- Ha una forte componente tecnica

Introduzione all'informatica

In inglese “computer science”

- È una disciplina principalmente matematica
- Nasce come derivazione della logica
- Ha una forte componente tecnica

Non solo programmazione dei computer:

“L'informatica non riguarda i computer più di quanto l'astronomia riguardi i telescopi.”

– E. Dijkstra

Informatica sul dizionario

informatica [in-for-mà-ti-ca] s.f. Scienza e tecnica che si occupa del trattamento automatico dell'informazione, per mezzo di elaboratori elettronici in grado di raccogliere i dati nella propria memoria, e di riordinarli secondo il programma assegnato.

Informatica sul dizionario

informatica [in-for-mà-ti-ca] s.f. Scienza e tecnica che si occupa del trattamento automatico dell'informazione, per mezzo di elaboratori elettronici in grado di raccogliere i dati nella propria memoria, e di riordinarli secondo il programma assegnato.

- (Sì) ha una componente scientifica ed una tecnica
- (NO) non solo elaboratori elettronici

Pionieri

Charles Babbage e Ada Lovelace (183x)

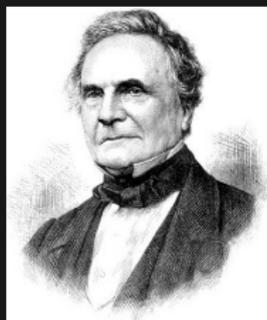


Figure: Charles Babbage
(1791–1871)



Figure: Ada Lovelace
(1815–1852)

- ▶ macchina differenziale (1822)
- ▶ macchina analitica (1837) mai realizzata

Alan Turing (1936)



Figure: Alan Turing (1912–1954)

David Hilbert chiede nel 1928: esiste una **procedura meccanica** in grado di stabilire se un'affermazione matematica è un teorema o meno?

Per risolvere il problema si deve spiegare cosa sia una procedura meccanica: la “spiegazione” di Turing è la **macchina di Turing**. Noi le chiamiamo “computer”.

La macchina universale

L'idea **fondamentale** della macchina Turing (embrionale in Babbage)

- ▶ macchina polivalente e **universale**
- ▶ una macchina programmabile
- ▶ il programma è un input come gli altri

Tesi di Church-Turing: *tutto ciò che è calcolabile lo è tramite una macchina di Turing più un programma.*

Il risultato è che i computer sono utilizzati per scopi ben oltre quelli per cui sono stati progettati.

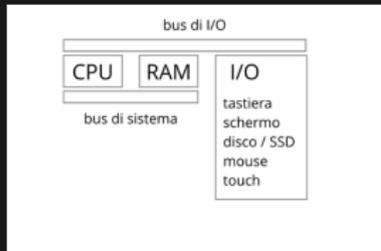
Modello di Von Neumann (1945)



Figure: John von Neumann (1903–1957)

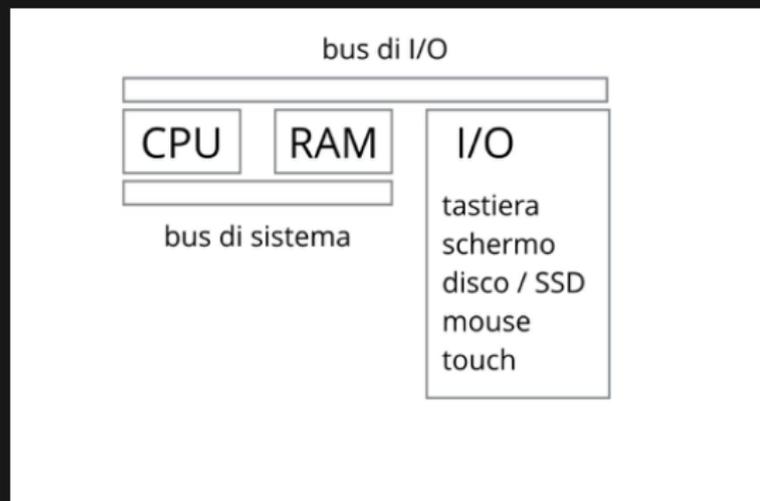
Modello su cui sono basati gli odierni calcolatori, più o meno.

- ▶ CPU: il cervello
- ▶ RAM: la memoria
- ▶ I/O: dispositivi di input and output
- ▶ bus: canali di comunicazione



Memoria e CPU

La memoria è il “foglio bianco” sul quale tenete traccia di calcoli e appuntate dati.



Linguaggi di Programmazione

Linguaggi naturali

- usati tra persone
- ambigui
- ridondanti
- contestuali
- evoluti spontaneamente

Linguaggi di programmazione

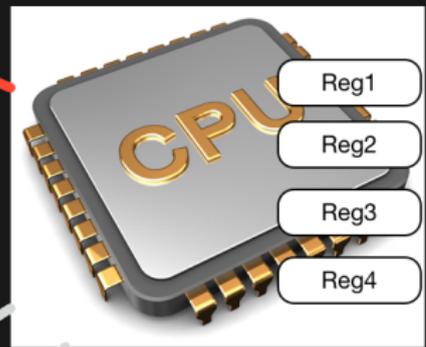
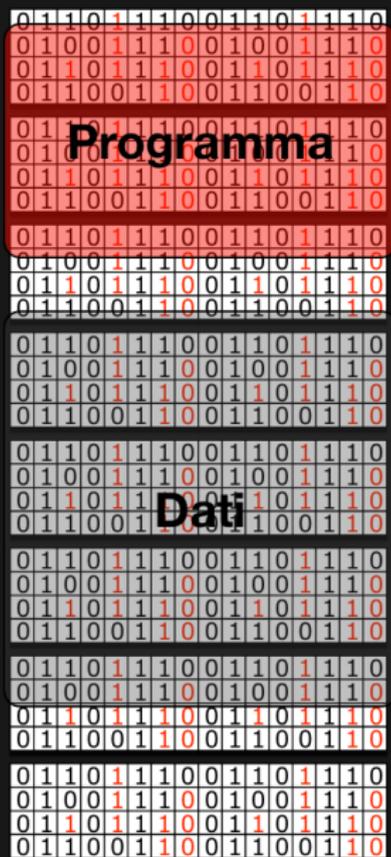
- parlare alla macchina
- non ambigui
- letterali
- criptici e compatti
- disegnati a tavolino

Esempio

Stampa la sequenza di numeri ottenuta elevando 4, 8, 3, 1, 5, -3, 6 al quadrato.

```
def square(x):  
    return x*x  
  
for elemento in [4,8,3,1,5,-3,6]:  
    print(square(elemento))
```

Esecuzione di un programma



1. RAM <-> Registri
4 o 8 byte
2. Registri <-> Registri
3. Operazioni matematiche
o logiche tra registri
4. saltare nel programma

Linguaggi di alto e basso livello

Alto livello

- più espressivi
- più facili da usare
- più leggibili
- meno efficienti
- orientati all'uomo

```
def square(x):  
    return x*x  
  
for elemento in [4,8,3,1,5,-3,6]:  
    print(square(elemento))
```

Basso livello

- astrazioni meno potenti
- più difficili
- più efficienti
- orientati alla macchina

```
entrypoint:  
    movq    %rdi, -8(%rbp)  
    movq    -8(%rbp), %rdi  
    cmpq    $0, 80(%rdi)  
    sete   %al  
label2:  
    xorb    $-1, %al  
    andb    $1, %al  
    movzbl  %al, %ecx  
    movslq  %ecx, %rdi
```


Esecuzione interattiva (e.g. Python)

```
"""Components for command line interface
Cython has many command line entry points to its functionality, and
some of these expose the same functionality over and over. This module
contains useful common components.
Copyright (C) 2012, 2013, 2014, 2015, 2016, Massimo Lauria <lauria@in.ibm.com>
https://github.com/MassimoLauria/cython.git
"""
from __future__ import print_function

import sys
import argparse
import networkx
import random

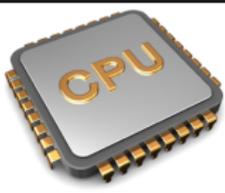
from itertools import combinations, product

from .graphs import supported_formats as graph_formats
from .graphs import read_from_url_from
from .graphs import bipartite, random, left_regular, bipartite, random, regular, big
if:
from .graphs import bipartite_sets
from .graphs import dag, complete, binary_tree, dag_pyramid
from .graphs import omnic, clustering_coefficient

try: # NetworkX <= 1.10
    complete_bipartite_graph = networkx.bipartite.complete_bipartite_graph
    bipartite_random_graph = networkx.bipartite.random_graph
    bipartite_gnm_random_graph = networkx.bipartite.gnm_random_graph
except AttributeError: # NetworkX > 1.10
    from networkx import complete_bipartite_graph
    from networkx import bipartite_random_graph
    from networkx import bipartite_gnm_random_graph

_all_ = [ "register_cython_subcommand", "is_cython_subcommand",
         "birectedcycle_iterator", "SimpleCythonMember", "BipartiteGraph
```

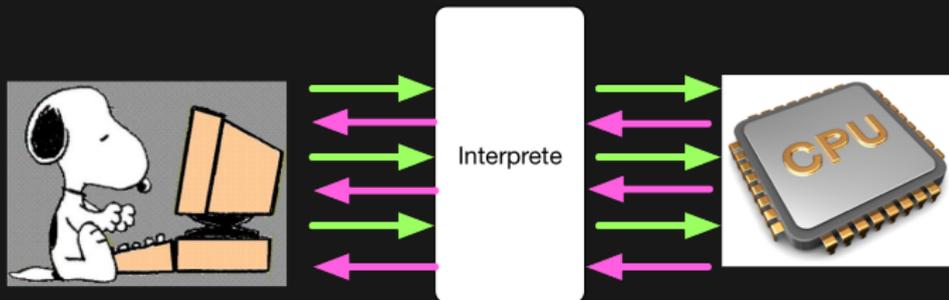
Interprete



Il programma viene letto da un **interprete** che esegue passo passo quello che è scritto nel programma.

- ▶ meno sicuri
- ▶ meno efficienti
- ▶ più flessibili
- ▶ ling. di alto livello

Esecuzione interattiva (e.g. Python)



Il programma viene letto da un **interprete** che esegue passo passo quello che è scritto nel programma.

- ▶ meno sicuri
- ▶ meno efficienti
- ▶ più flessibili
- ▶ ling. di alto livello