UNIVERSITÀ
DI PISA

PROGRAMMAZIONE e ALGORITMICA

introduzione

LUN 09 - 11
MAR 09 - 11
MER 09 - 11

LUN **09:00 - 10:30**
MER **09:00 - 10:30**

corrado priami - corrado.priami@unipi.it

oggi vedremo…

1. linguaggi di programmazione
2. compilatori e interpreti
3. architettura dei computer
4. paradigmi di programmazione

cosa imparerete

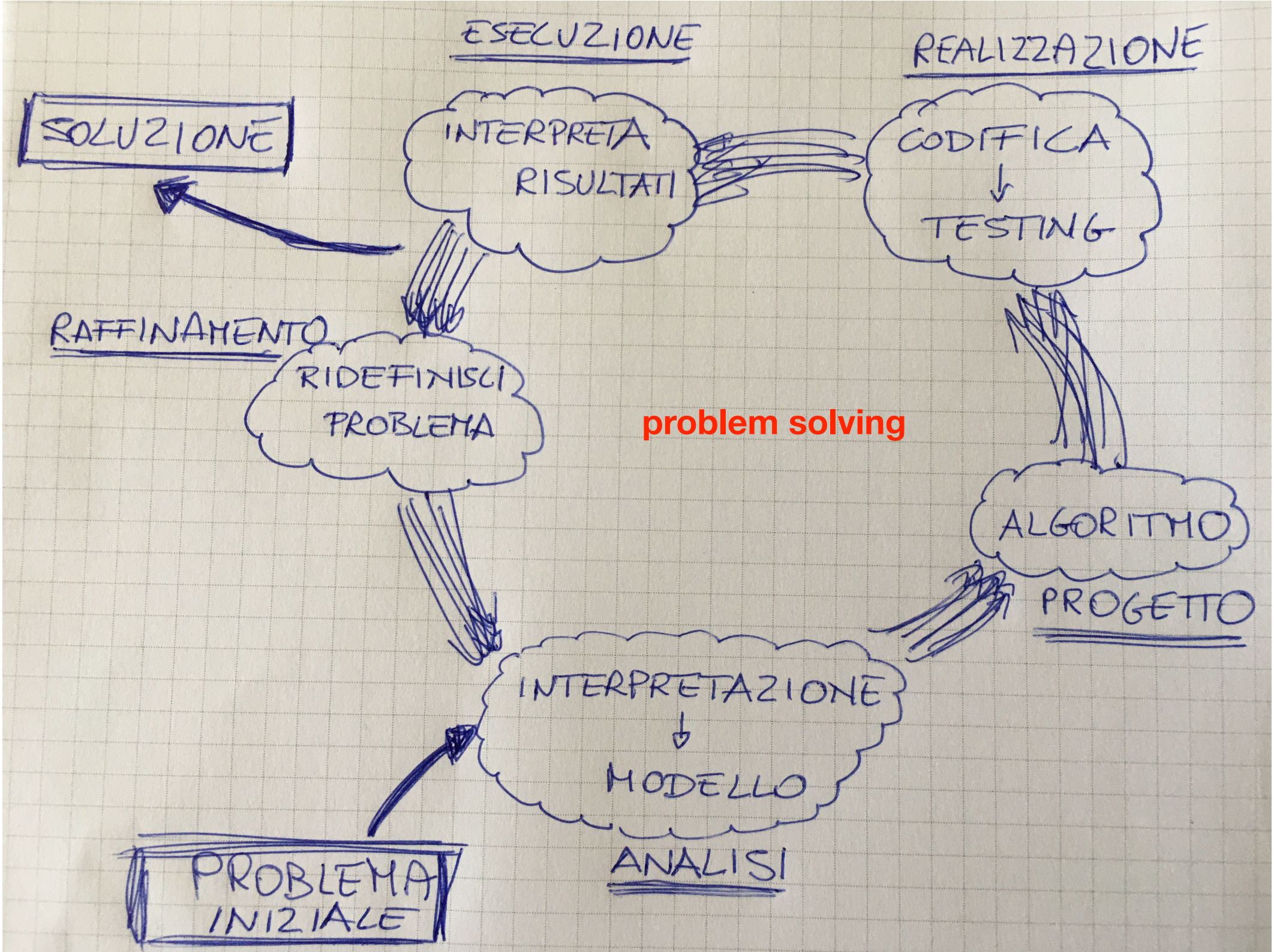
analizzare problemi complessi e progettare algoritmi per risolverli

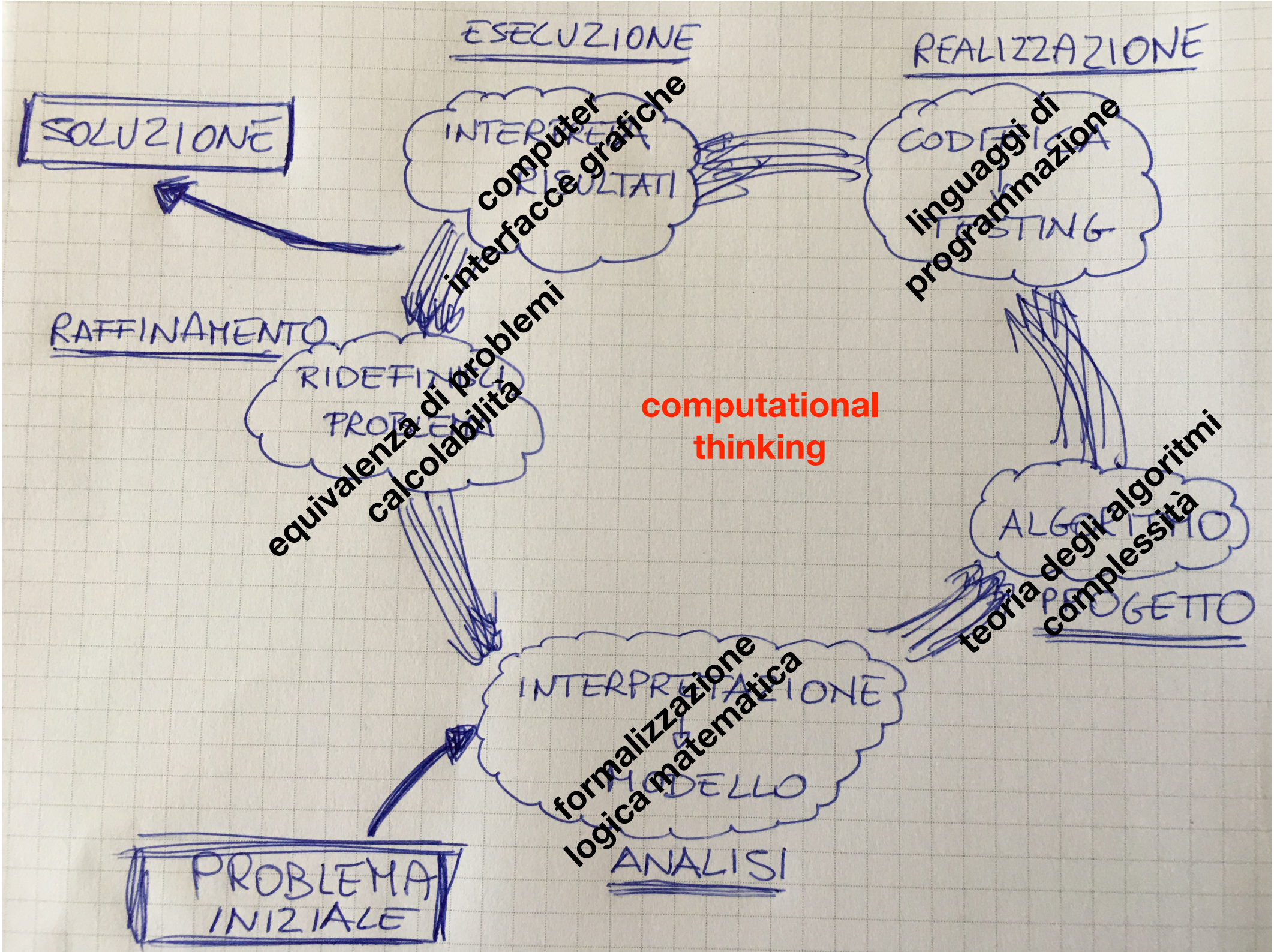
confrontare algoritmi in base al loro costo di esecuzione in tempo e spazio

comprendere i costrutti dei linguaggi di programmazione
e il loro funzionamento

utilizzare i linguaggi di programmazione per codificare algoritmi

in una frase: (LE BASI DEL)L'INFORMATICA


ESECUZIONE
REALIZZAZIONE
SOLUZIONE
INTERPRETA RISULTATI
CODIFICA
TESTING
RAFFINAMENTO
RIDEFINISCI PROBLEMA
problem solving
ALGORITMO
PROGETTO
INTERPRETAZIONE
MODELLO
ANALISI
PROBLEMA INIZIALE


ESECUZIONE
REALIZZAZIONE
SOLUZIONE
computer
interfacce grafiche
linguaggi di
programmazione
RAFFINAMENTO
equivalenza di problemi
calcolabilità
computational
thinking
teoria degli algoritmi
complessità
formalizzazione
logica matematica
ANALISI
PROBLEMA
INIZIALE

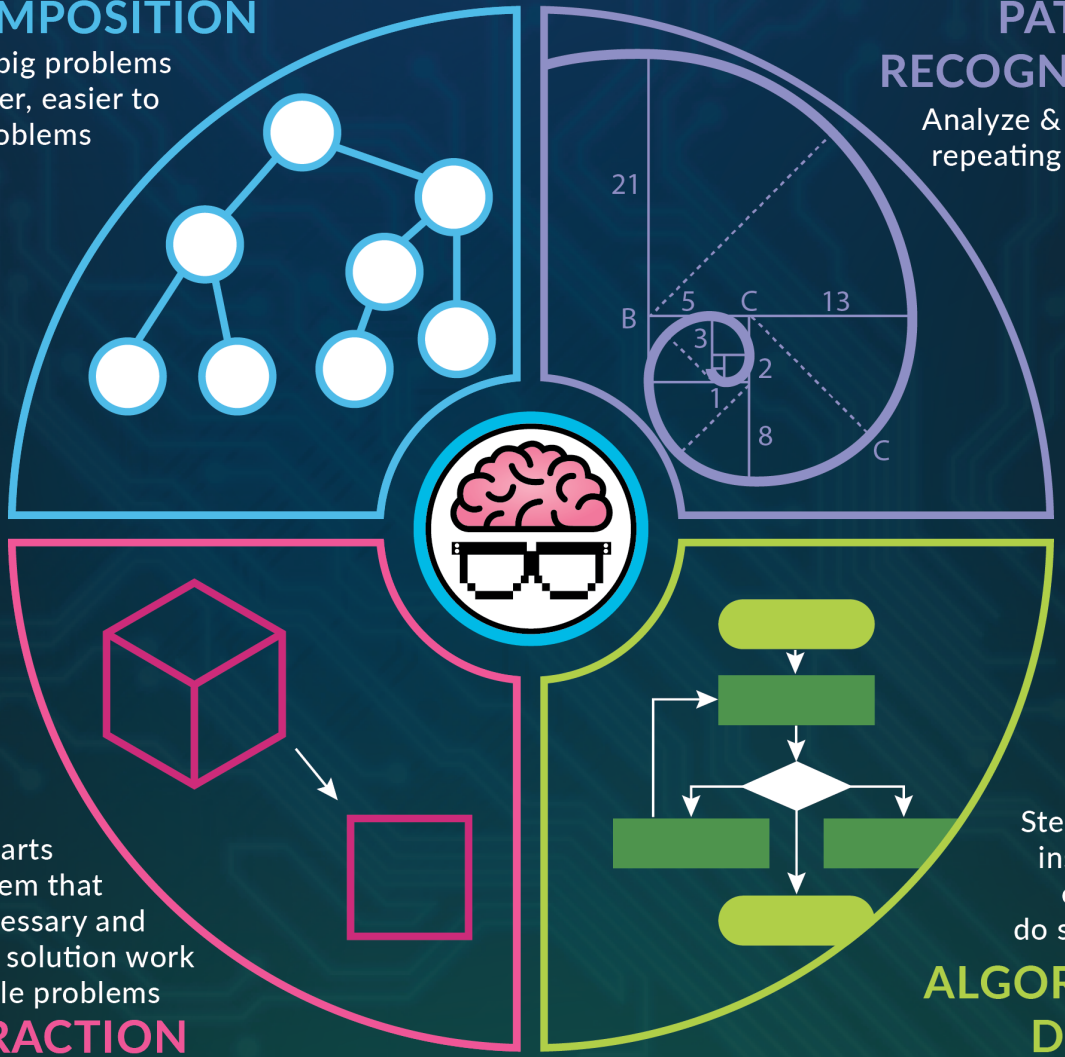
programmare è

Jeannette Marie Wing is Avanessians Director of the Data Sciences Institute at Columbia University, where she is also a professor of computer science.

COMPUTATIONAL THINKING

DECOMPOSITION
Breaking big problems into smaller, easier to mange problems

PATTERN RECOGNITION
Analyze & look for a repeating sequence



Remove parts of a problem that are unnecessary and make one solution work for multiple problems
ABSTRACTION

Step-by-Step instructions on how to do something
ALGORITHM DESIGN

un algoritmo è

descrizione formale di una sequenza finita di azioni elementari e non ambigue che prendono dei dati in input, vengono eseguite e producono dei dati in output per risolvere una classe di problemi

I dati in input determinano una istanza della classe di problemi

1. Mettere sul fuoco una pentola di acqua
2. Aggiungere 1 cucchiaino di sale
3. Attendere che l'acqua bolla
4. Gettare 80gr di fusilli nella pentola
5. Attendere 11 minuti
6. Scolare i fusilli

1. Riscrivere in forma normale l'equazione intera di grado 2
2. Calcolare il delta b^2 - 4ac
3. Se il delta è > 0
    1. x1 = (-b + sqrt(delta))/2a & x2 = (-b - sqrt(delta))/2a
4. se il delta è = 0
    1. x1 = x2 = -b/2a
5. altrimenti
    1. nessuna soluzione reale

programmare è

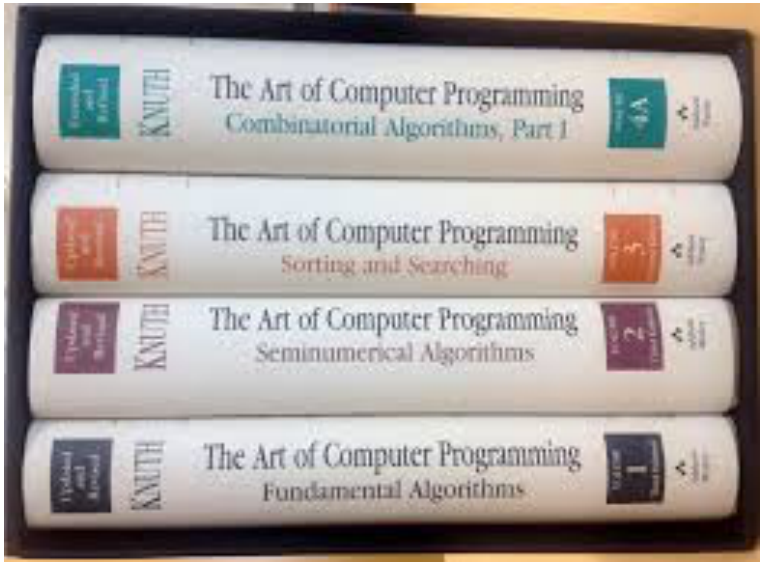
scrivere un **documento** che risolve un **problema reale** e che può essere **compreso e eseguito** da un computer

trovando la soluzione più **efficiente** possibile

un’attività creativa e complessa che sviluppa il **problem solving**

un **programma** è la scrittura di un algoritmo in un linguaggio di programmazione

Donald Ervin Knuth (1938 -) is an American computer scientist, mathematician, and professor emeritus at Stanford University. He is the author of the multi-volume work The Art of Computer Programming. Turing Award.



UNIVERSITÀ
DI PISA

linguaggi di programmazione

ce ne sono moltissimi

impareremo i fondamenti di ciascuno…

…senza usarne nessuno in particolare

struttura di un linguaggio

lessico: come si scrivono le parole

pippo **maangia** la **mla**

grammatica: come si compongono le frasi

pippo **mangiano gli** mela

sintassi

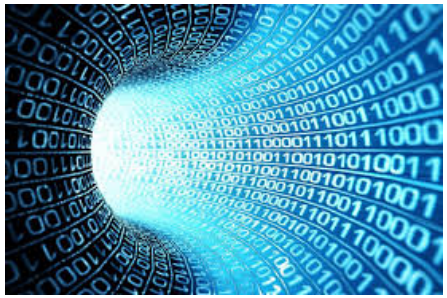
semantica: significato delle frasi

la mela mangia pippo

linguaggi di programmazione: classificazione

```
    pushl   %ebp                # \
    movl    %esp, %ebp          #  ) reserve space for local variables
    subl    $16, %esp           # /
    call    getint              # read
    movl    %eax, -8(%ebp)      # store i
    call    getint              # read
    movl    %eax, -12(%ebp)     # store j
A:  movl    -8(%ebp), %edi      # load i
    movl    -12(%ebp), %ebx     # load j
    cmpl    %ebx, %edi          # compare
    je      D                   # jump if i == j
    movl    -8(%ebp), %edi      # load i
    movl    -12(%ebp), %ebx     # load j
    cmpl    %ebx, %edi          # compare
    jle     B                   # jump if i < j
    movl    -8(%ebp), %edi      # load i
    movl    -12(%ebp), %ebx     # load j
    subl    %ebx, %edi          # i = i - j
    movl    %edi, -8(%ebp)      # store i
    jmp     C
B:  movl    -12(%ebp), %edi     # load j
    movl    -8(%ebp), %ebx      # load i
    subl    %ebx, %edi          # j = j - i
    movl    %edi, -12(%ebp)     # store j
C:  jmp     A
D:  movl    -8(%ebp), %ebx      # load i
    push    %ebx                # push i (pass to putint)
    call    putint              # write
    addl    $4, %esp            # pop i
    leave                       # deallocate space for local variables
    mov     $0, %eax            # exit status for program
    ret                         # return to operating system
```

Linguaggio assembler



Linguaggio binario

```
var x = 4;
var y = 2;
if (x % 2 == 0) {print("la variabile x e' pari e vale \(x)")}
else {print("la variabile x e' dispari e vale \(x)")}
```

Linguaggio alto livello

traduttori

```
//prg pippo
#include <stdio.h>

int main( void)
{..
```

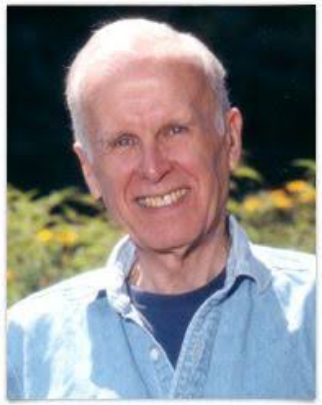
→ **traduttore** →

```
0100100100010010 10
1001010101010011 11
1000111010101010 00
1100100100100100 01
0010010010001010 10
```

ma **semanticamente equivalente**

Grace Murray Hopper (1906 – 1992) è stata una matematica, informatica e militare statunitense.

Anni ’50: progetta e realizza il primo compilatore sperimentale



John Warner Backus (1924 – 2007) was an American computer scientist. He directed the team that invented and implemented FORTRAN, the first widely used high-level programming language, and was the inventor of the Backus–Naur form (BNF), a widely used notation to define formal language syntax. He was awarded the ACM Turing Award in 1977

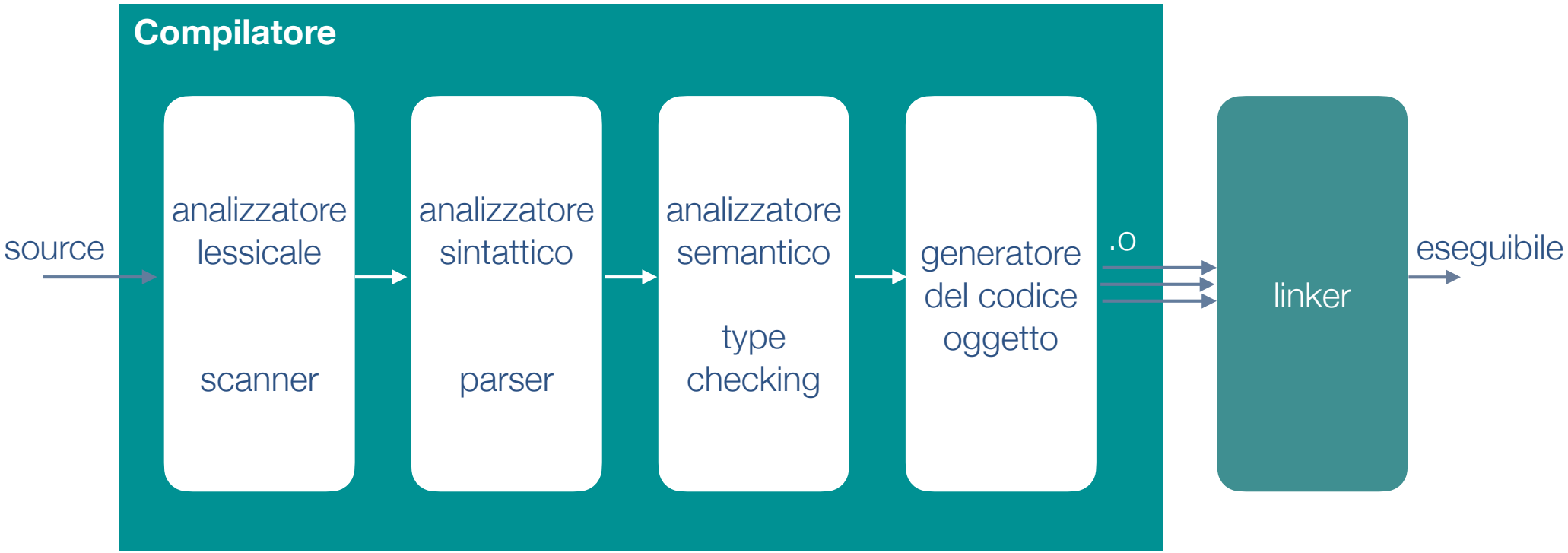
1957: progetta e realizza il primo compilatore completo

compilatori e interpreti

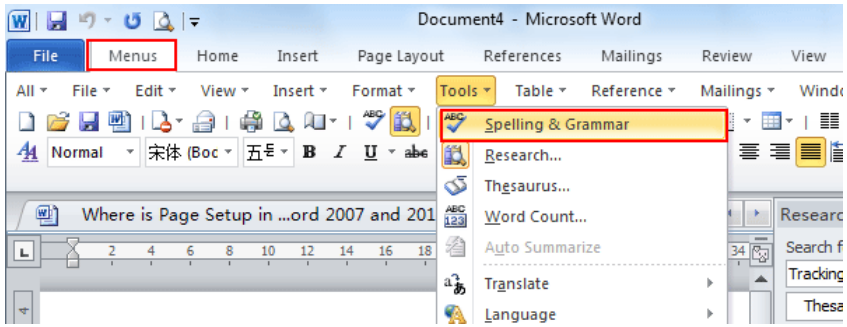
Compilatore: traduce tutto il programma e genera un eseguibile

Interprete: traduce ed esegue un'istruzione alla volta

struttura del compilatore



non solo linguaggi di programmazione



le fasi di sviluppo di un programma

scrivi il programma

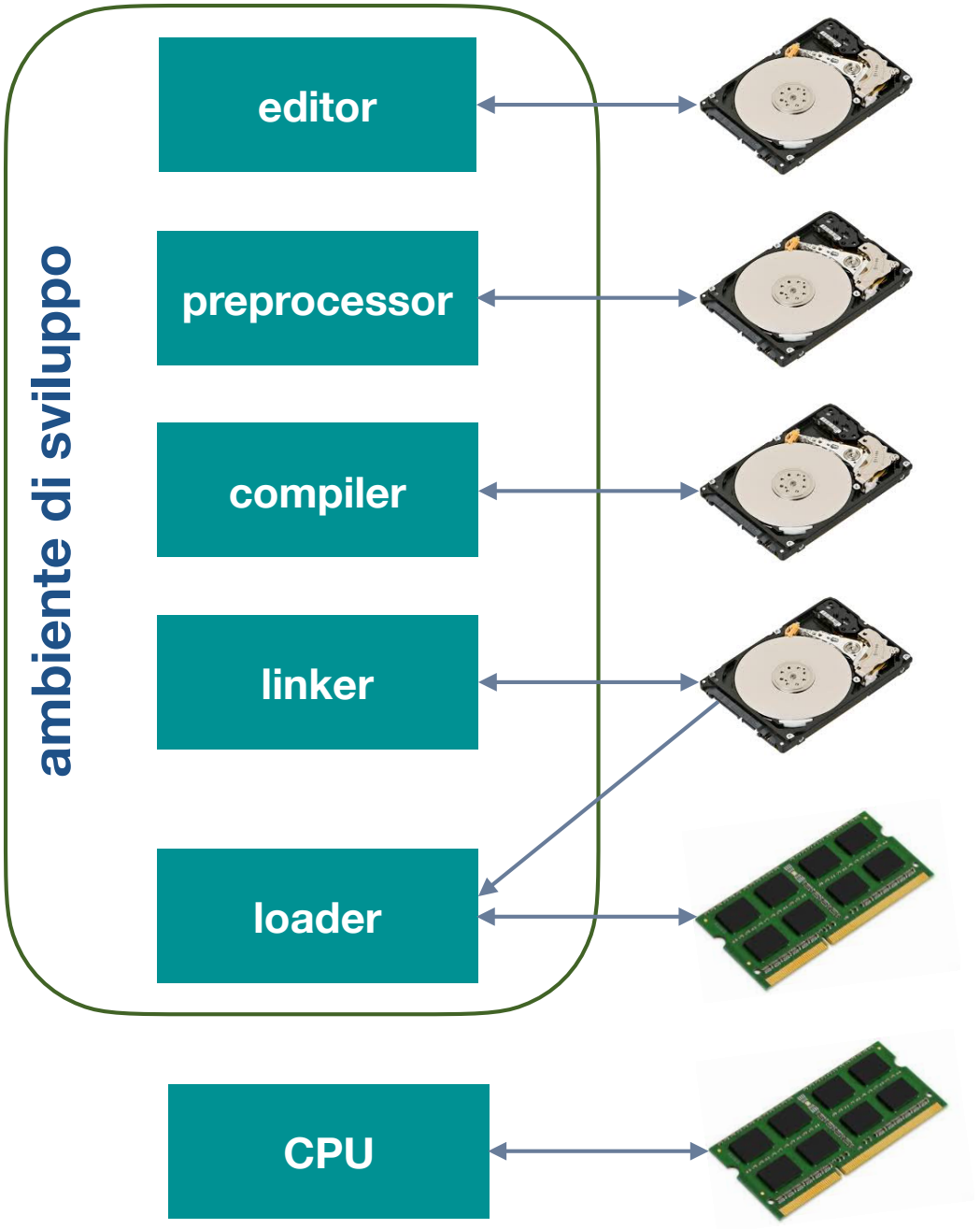
prepara il programma per il compilatore

genera 1 o più file oggetto

collega i file oggetto e genera l'eseguibile

carica l'eseguibile in memoria principale

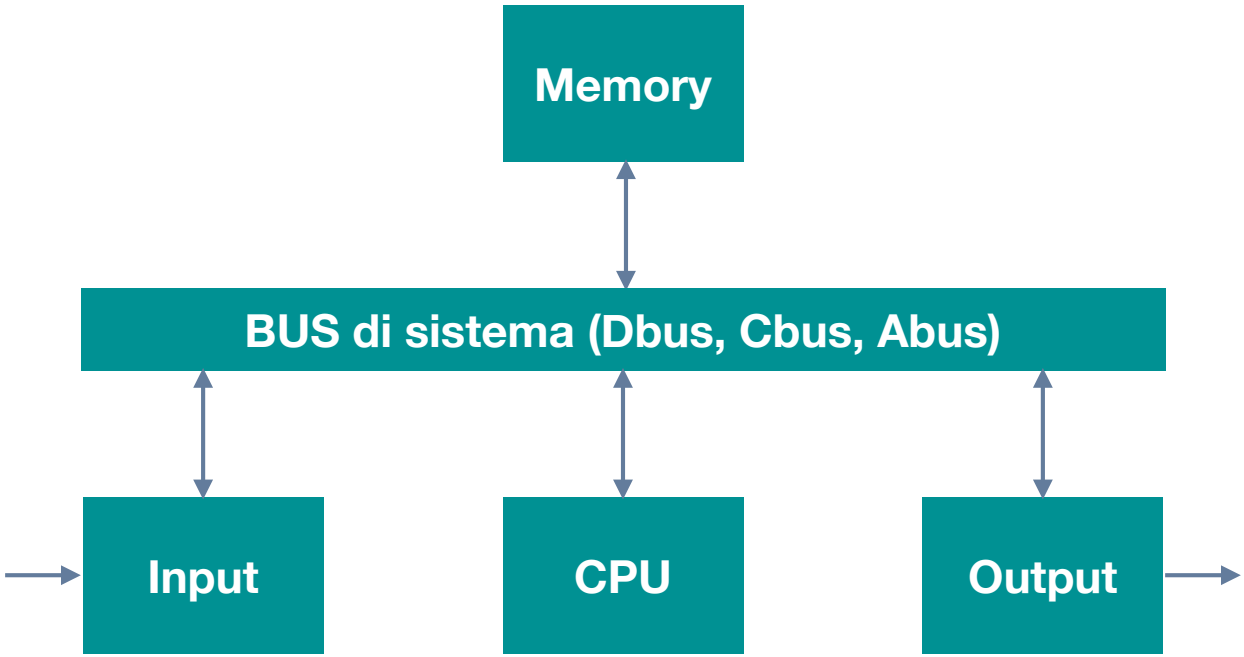
esegue il programma



UNIVERSITÀ
DI PISA

architettura del computer

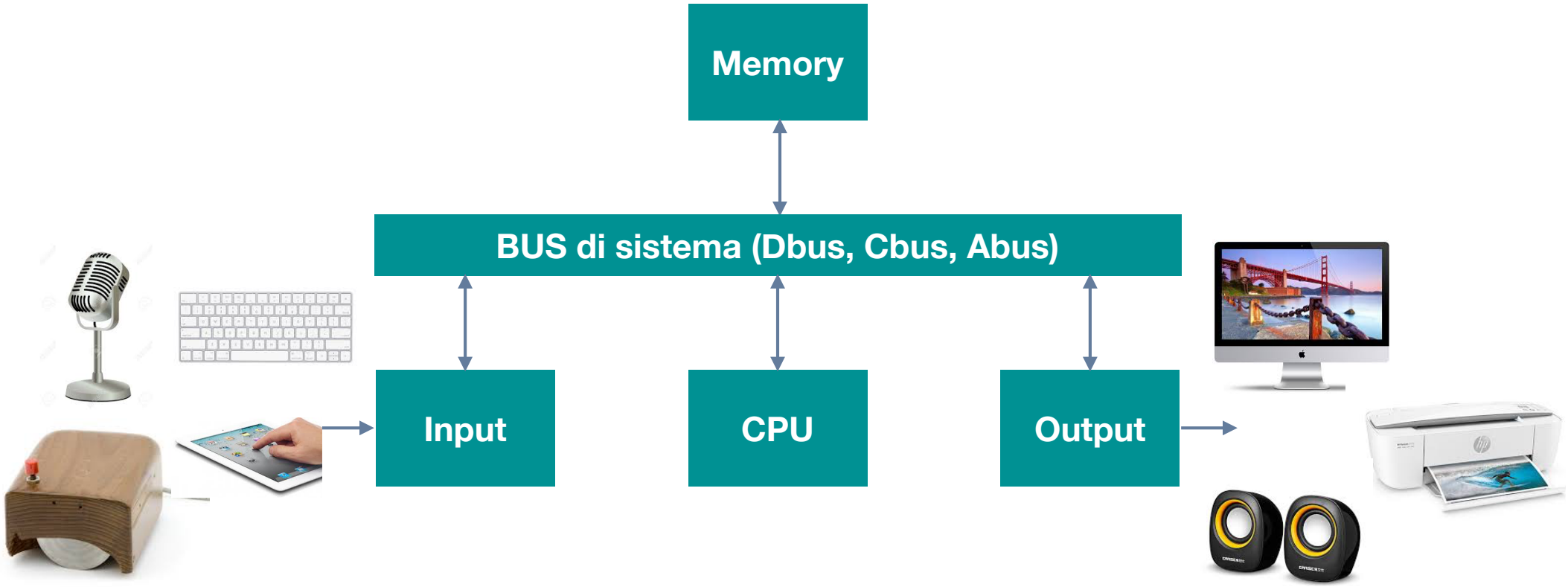
architettura elementare



John von Neumann (1903 – 1957) was a Hungarian-American mathematician, physicist, computer scientist, and polymath.

Dati e programmi sono la stessa cosa

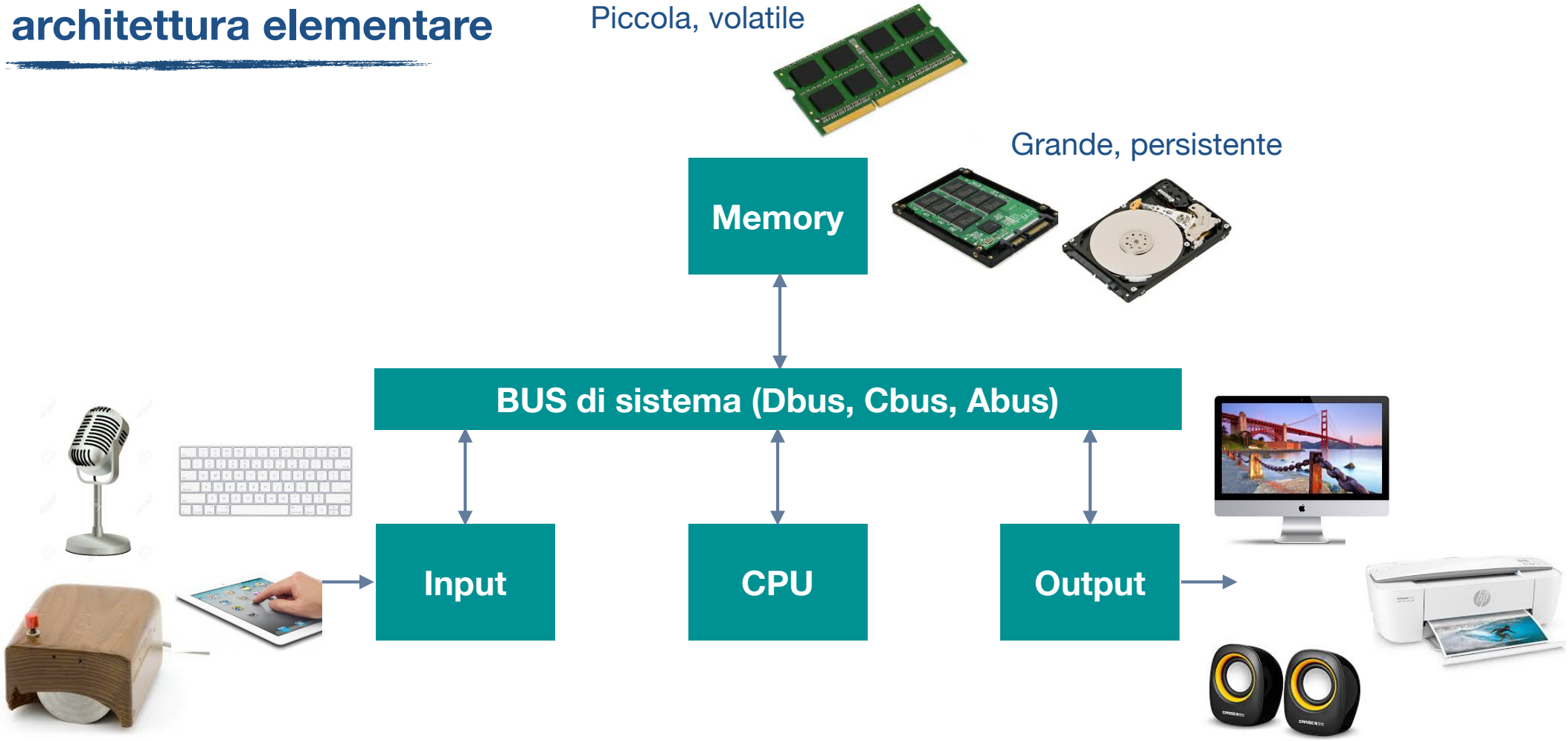
architettura elementare



John von Neumann (1903 – 1957) was a Hungarian-American mathematician, physicist, computer scientist, and polymath.

Dati e programmi sono la stessa cosa

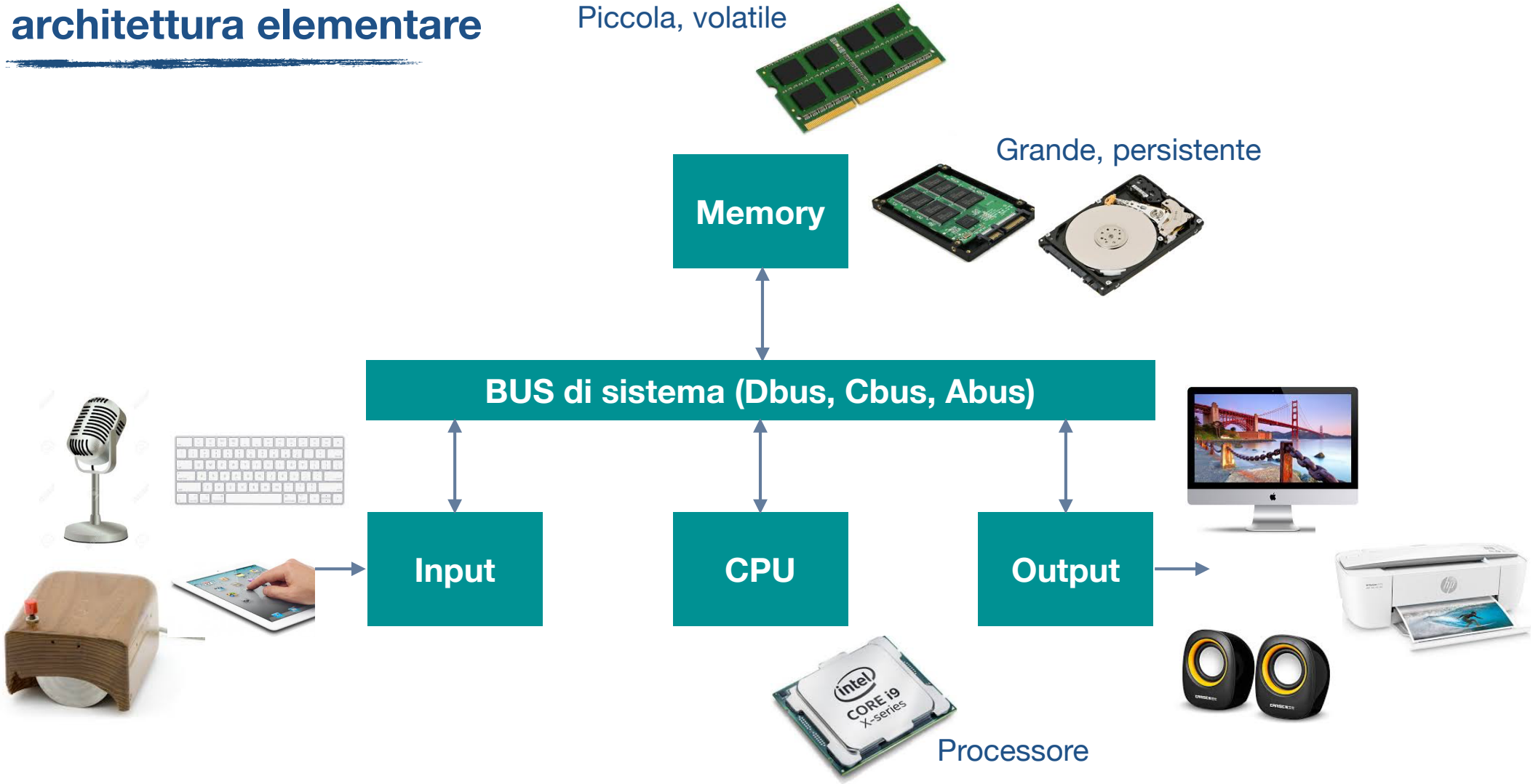
architettura elementare



John von Neumann (1903 – 1957) was a Hungarian-American mathematician, physicist, computer scientist, and polymath.

Dati e programmi sono la stessa cosa

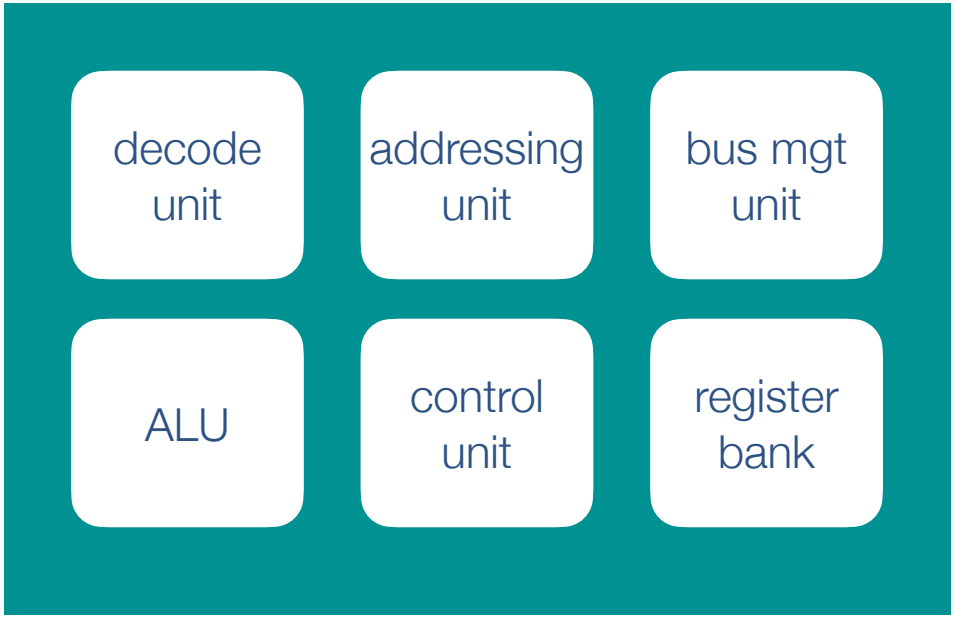
architettura elementare



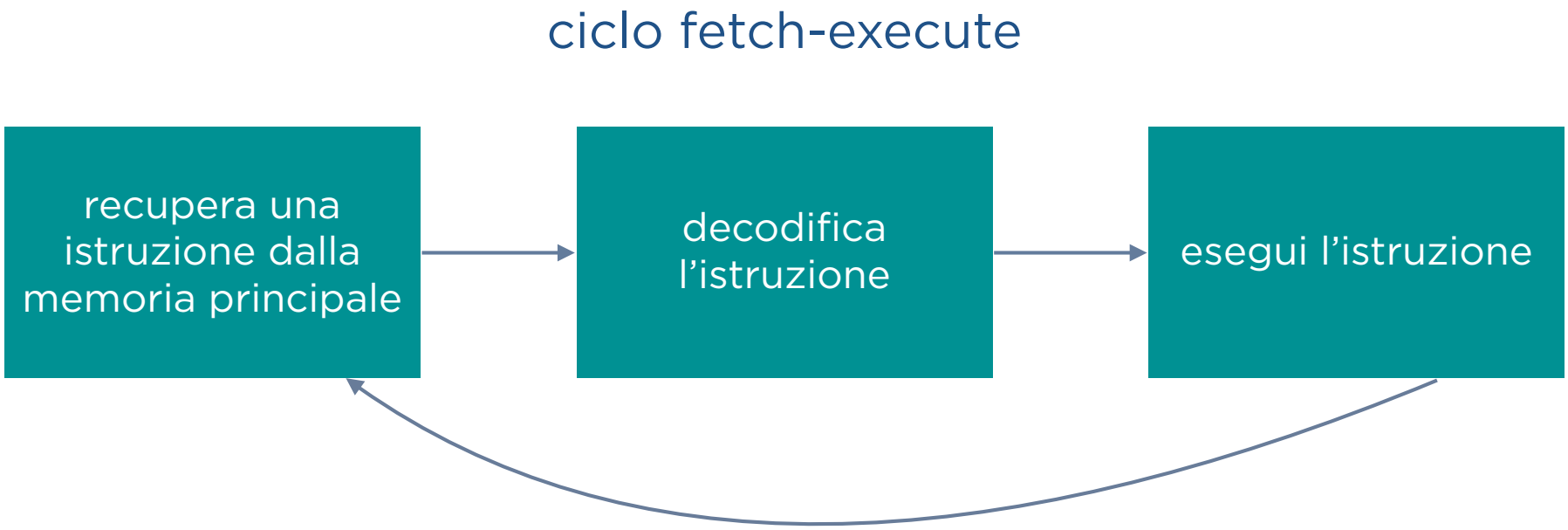
John von Neumann (1903 – 1957) was a Hungarian-American mathematician, physicist, computer scientist, and polymath.

Dati e programmi sono la stessa cosa

dettaglio della CPU



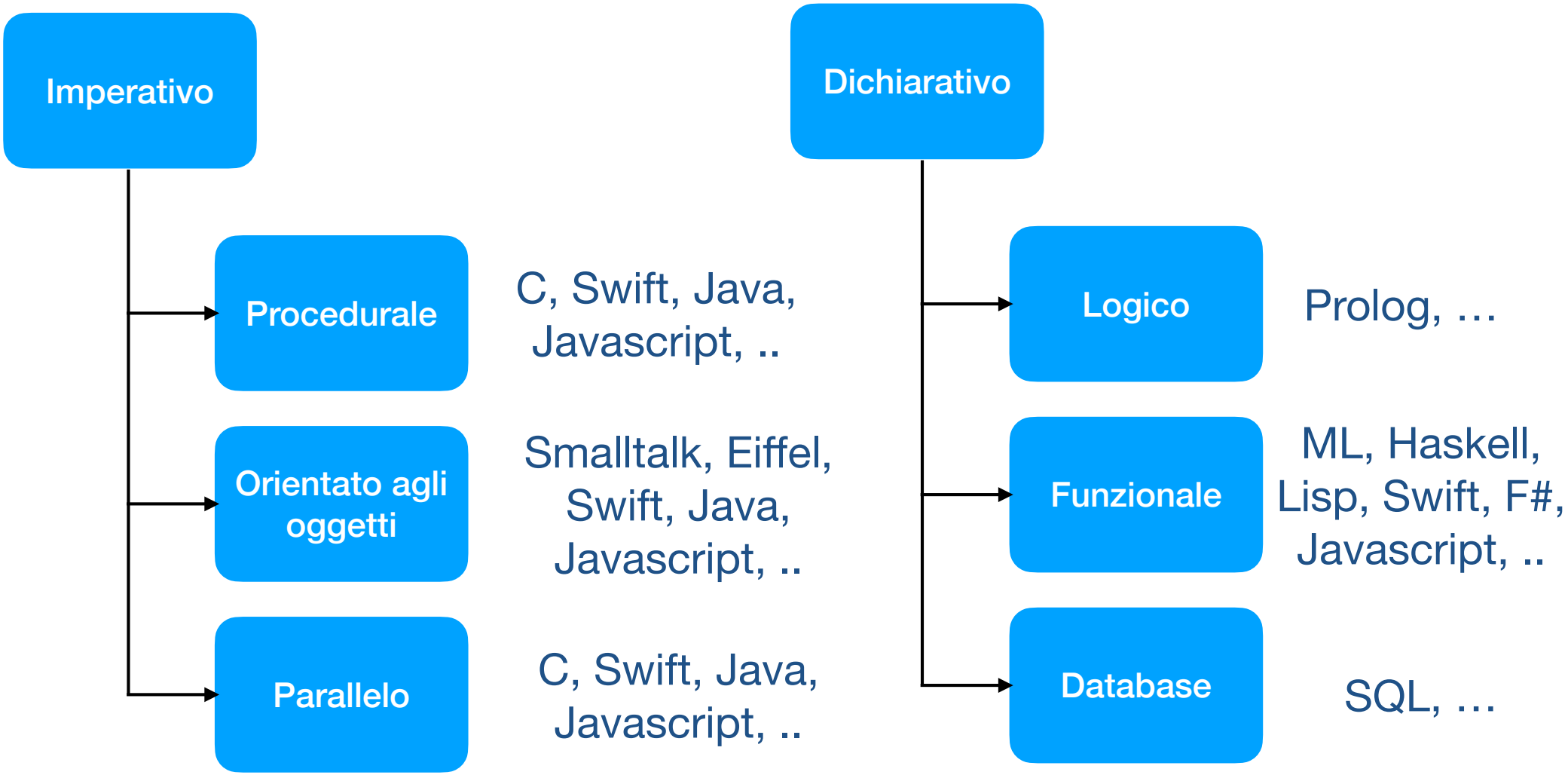
esecuzione di programmi



UNIVERSITÀ
DI PISA

paradigmi di programmazione

paradigmi



quasi tutti i linguaggi moderni sono multiparadigma

noi vedremo

Procedurale

È una astrazione dell'architettura di Von Neumann e si programma definendo come cambia la memoria dopo ogni azione elementare

Funzionale

Si definiscono funzioni matematiche sul dominio dei dati in ingresso e si fornisce il corrispondente valore del condominio. Un programma è una composizione matematica di funzioni. Viene meno il concetto di memoria.

Orientato agli oggetti

Si definiscono degli oggetti in grado di contenere dati e operare su questi. Gli oggetti del programma si coordinano scambiandosi messaggi

oggi abbiamo visto…

1. linguaggi di programmazione
2. compilatori e interpreti
3. architettura dei computer
4. paradigmi di programmazione

UNIVERSITÀ
DI PISA

buona giornata!!!

corrado priami - corrado.priami@unipi.it