

Architettura degli Elaboratori I - 2018/19 - B

Università di Torino - Corso di Laurea in Informatica

Prof. Marco Aldinucci (Teoria ed Esercitazioni)

Dr. Maurizio Lucenteforte (Laboratorio)

Dr. Claudio Schifanella (Laboratorio)

Organizzazione del corso

- Il corso B è per studenti il cui cognome inizia con L-Z
 - **Nessuna eccezione.** No passaggi di corso
- Il programma svolto sarà sostanzialmente lo stesso per il corso A e B
 - Stesse modalità per il corso A e B: esame scritto
- Docenti diversi
 - Ognuno interpreta lo stesso programma in modo leggermente diverso
 - In ogni caso i compiti degli studenti del corso B verranno corretti dal docente del corso B
- Sarete tenuti, alla fine, a dare una vostra **valutazione del corso**
 - Senza di questa non è consentita l'iscrizione all'esame

Organizzazione del corso

- Siete tenuti a calcolare il vostro turno di laboratorio come segue:
 - se il numero di login (le cifre che seguono 'st') è dispari, il turno di appartenenza è il T1 (Martedì, ore 14-17. Prof. Lucenteforte)
 - se il numero di login (le cifre che seguono 'st') è pari, il turno di appartenenza è il T2 (Lunedì, ore 9-12, Prof. Schifanella)
- La prima lezione si terrà dal 11 Marzo

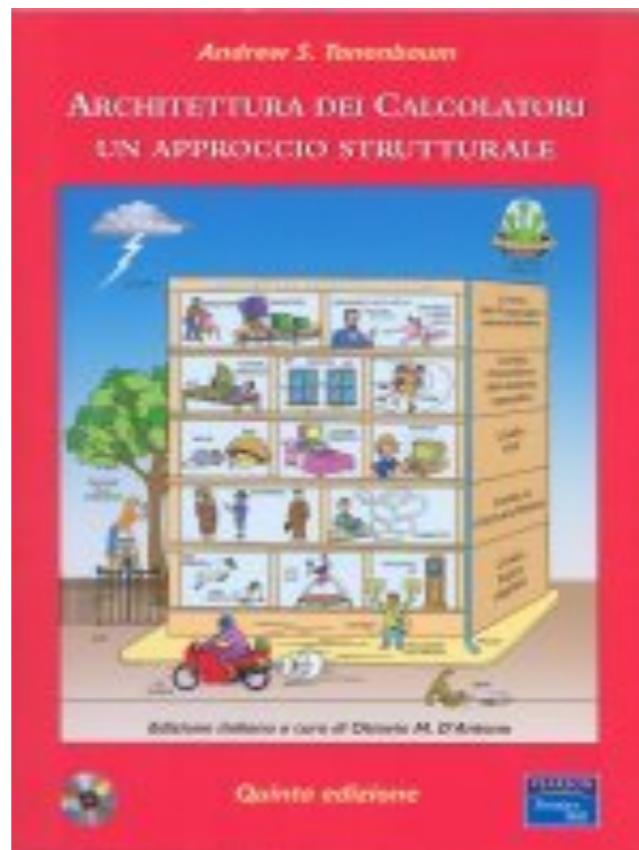
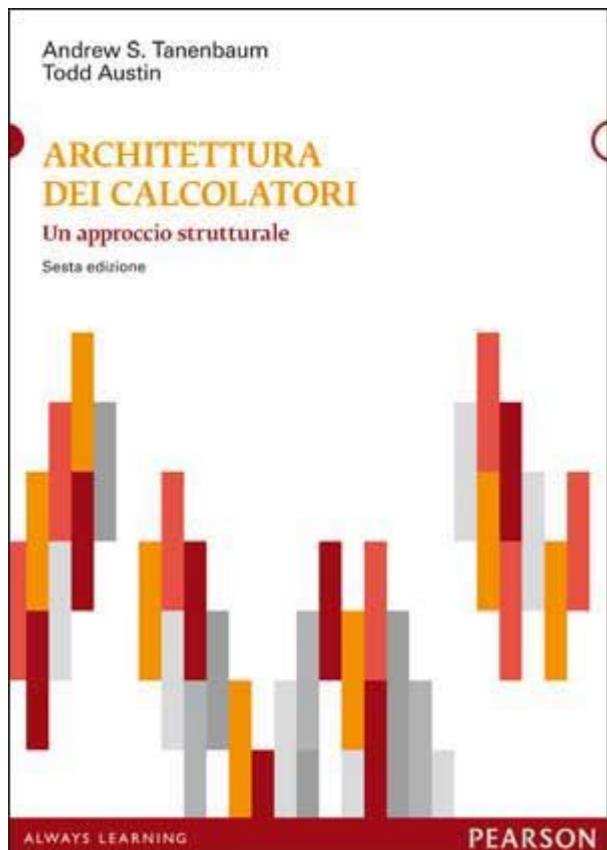
Esame

- L'esame consiste di una prova scritta:
 - per sostenere le prove scritte è necessario iscriversi on-line
 - annullare la propria iscrizione (o avvertire via mail se scaduti i termini) in caso di assenza alla prova d'esame
 - il voto va accettato entro 5 giorni dalla pubblicazione dei risultati
- Per superare l'esame è necessario:
 - **studiare sul libro di testo e sul materiale didattico suggerito**
 - i lucidi che saranno messi a disposizione non lo sostituiscono – inoltre parti significative saranno alla lavagna
 - seguire le esercitazioni, prendere appunti
 - seguire le lezioni, liberarsi dalla schiavitù dello smartphone per 2 ore

Esame

- Tipicamente consiste di 10 domande:
- 8 di teoria con domande aperte, scelta multipla o esercizi (24 punti)
- 2 di laboratorio (8 punti) in cui si deve dimostrare:
 - Conoscenza degli strumenti software utilizzati durante le ore di laboratorio
 - Assembly (IJVM)
 - Micro-architettura e micro-programmazione (MAL)

Materiale didattico



Testi di consultazione



Andrew S. Tanenbaum, Todd Austin
Architettura dei calcolatori: Un approccio strutturale
5a o 6a Edizione, Pearson.

M. Vanneschi.
Architettura degli Elaboratori. Pisa University Press, 2013.

D. A. Patterson,
J. L. Hennessy.
Struttura e progetto dei calcolatori, terza edizione.
Zanichelli, 2014

Supporto on-line (moodle)

- Tutto il materiale usato a lezione viene fornito dopo un breve (e deliberato) lasso di tempo
 - Comunicazioni e news dell'ultimo minuto, forum di discussione
 - Pubblicazione del materiale didattico (es. lucidi, programmi, link)
 - **Si studia sul libro non sulle slide**
- In ogni caso, durante il corso saranno possibili correzioni, aggiunte, integrazioni
 - Verificate regolarmente il materiale messo a disposizione

<http://informatica.i-learn.unito.it/>

Conoscenze necessarie

- Programmazione I
- Algebra di Boole (Matematica discreta e Logica)

A.A. 2018/19

- 1 credito teoria = 8 ore
- 1 credito laboratorio = 10 ore
- Corso
 - Teoria (48 ore)
 - Esercitazioni (12 ore)
 - Laboratorio (30 ore)

Syllabus (per blocchi)

- Introduzione (2 ore)
- Sistema binario e codifica (4 ore)
- Assembly (8 ore)
- Reti combinatorie e sequenziali (8 ore)
- Memorie (6 ore)
- Microarchitettura (16 ore)
- Toolchain (4 ore)

Orari

- Esercitazione
 - in 6 date da fissare (12 ore) nel giorno di Giovedì ore 14-16
- Ricevimento
 - Mercoledì 14-15
 - Giovedì 14-15

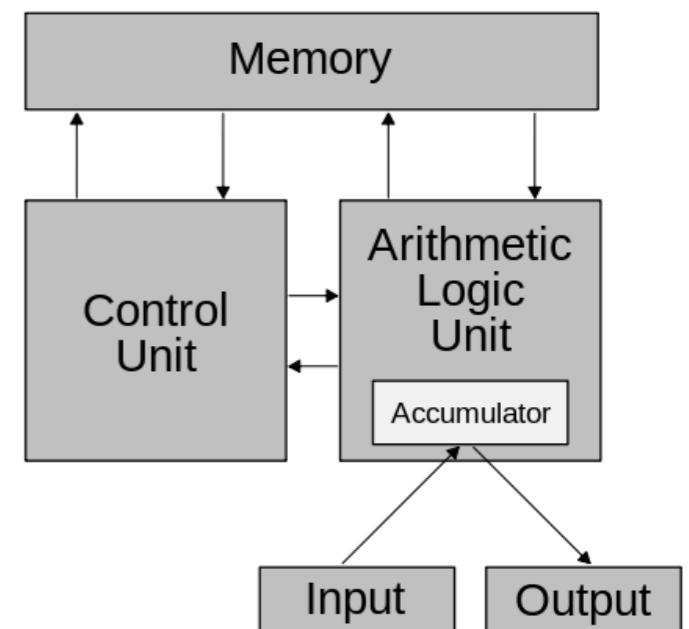
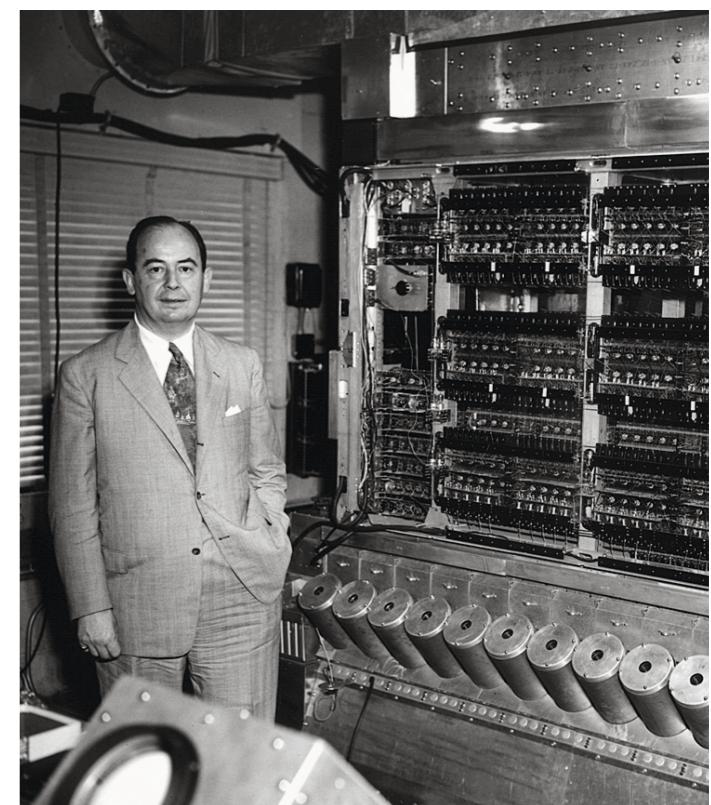
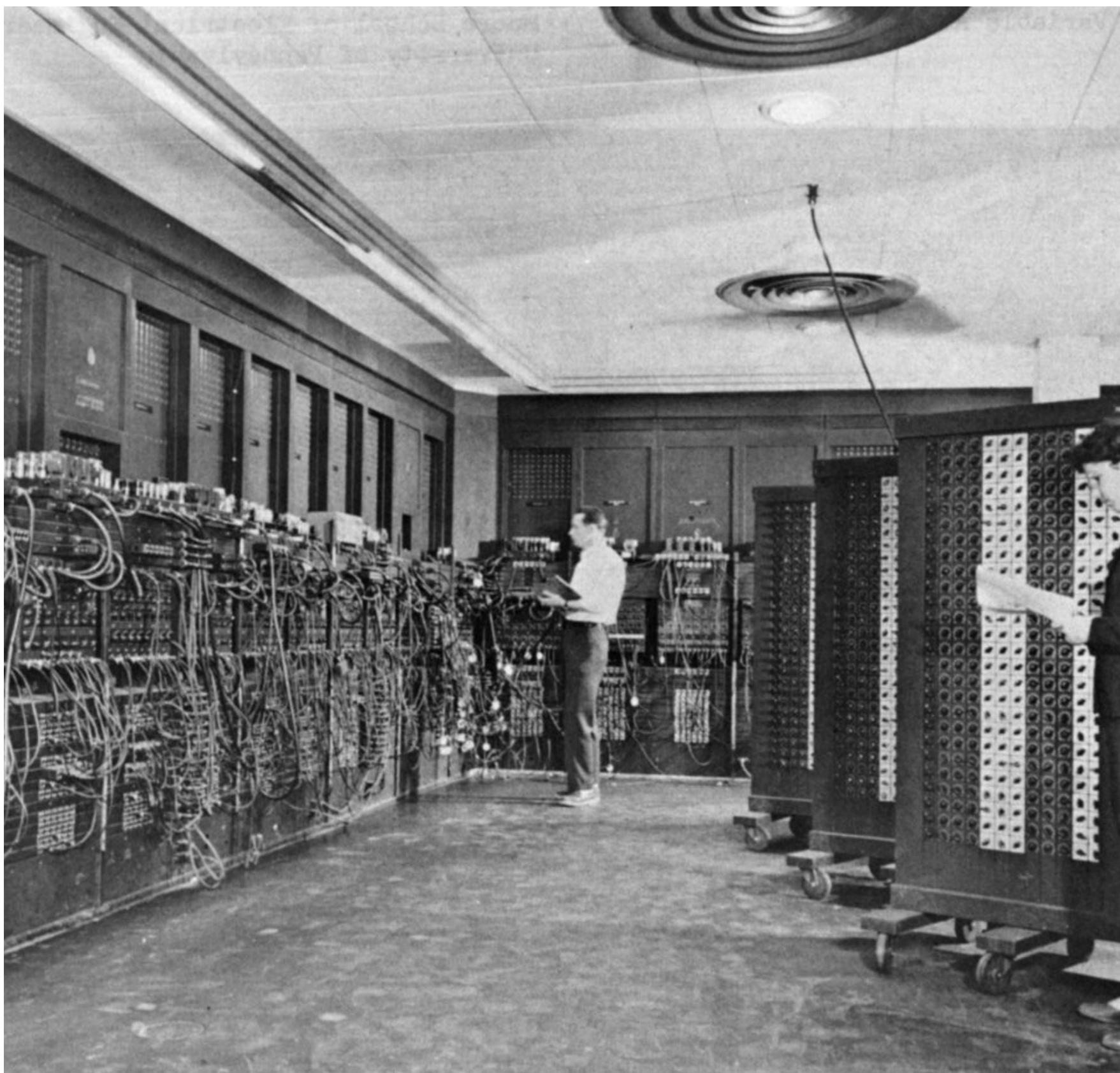
“Pillole di storia dei calcolatori”

Pillole di storia

- **Pascal** (1623 – 1662): Calcolatrice meccanica per eseguire somme e sottrazioni
- **Leibniz** (1646 – 1716): Macchina meccanica in grado di eseguire anche moltiplicazioni e divisioni
- **Babbage** (1792 – 1871): *Difference engine*: calcolare tavole di numeri, eseguiva un solo algoritmo. *Analytical engine*: con un magazzino per memorizzare dati
- **Zuse** (1910 – 1995): negli anni '30 costruì macchine calcolatrici con relé elettromagnetici, distrutte da un bombardamento nel '44. Il calcolatore "Z1", completato da Zuse nel 1938, è considerato il primo computer moderno. A Konrad Zuse si deve anche l'invenzione del primo linguaggio di programmazione della storia, ideato per fornire le istruzioni allo "Z1": il Plankalkül.

1° generazione – Valvole (1945 – 1955)

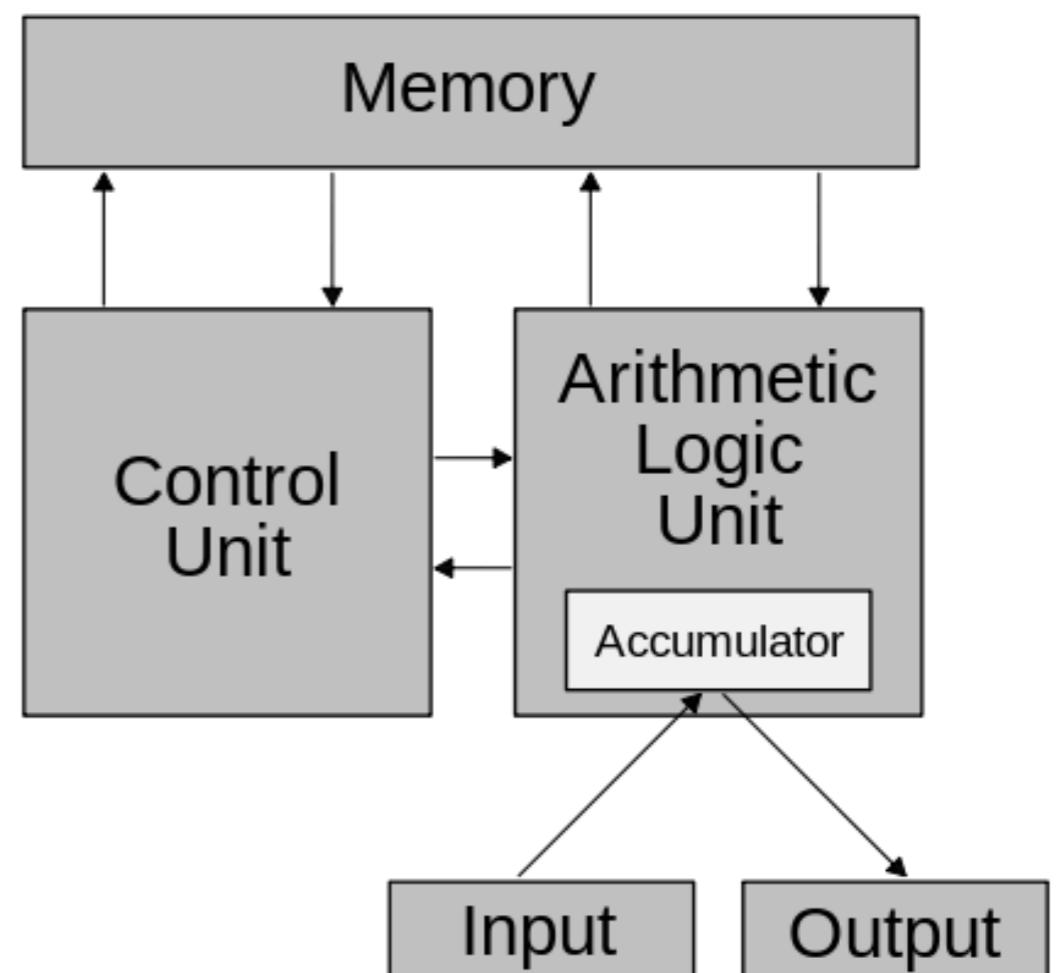
- Colossus per decifrare ENIGMA (Alan Turing)
- ENIAC (Mauchley – Eckert) 18.000 valvole termoioniche, 1.500 relé, 30 tonnellate, 140 Kw consumo di energia,
- IAS (von Neumann): aritmetica binaria, dati e istruzioni in memoria



ENIAC, IAS con Von Neumann e la sua architettura

L'architettura Von Neumann

- “A programma memorizzato”
- I programmi sono dati, dati e programmi sono memorizzati nella stessa memoria



E le generazioni successive

- 2° generazione – Transistor (1955 – 1965)
- 3° generazione – Circuiti Integrati (1965 – 1980)
- 4° generazione – Integrazione su vasta scala (1980 – ...)
- ...
- Poi ... iscrivetevi alla laurea magistrale: Architetture II

Ciclo Estrazione-Esecuzione

1. Prendi l'istruzione seguente dalla memoria e mettila nel registro delle istruzioni
2. Cambia il program counter per indicare l'istruzione seguente
3. Determina il tipo dell'istruzione appena letta
4. Se l'istruzione usa un dato in memoria, determina dove si trovaMetti il dato, se necessario, in un registro della CPU
5. Esegui l'istruzione
6. Torna al punto 1 e inizia a eseguire l'istruzione successiva

Ciclo Estrazione-Esecuzione

```
static int PC, AC;
static int instr, instr_type;
static int data_loc, data;

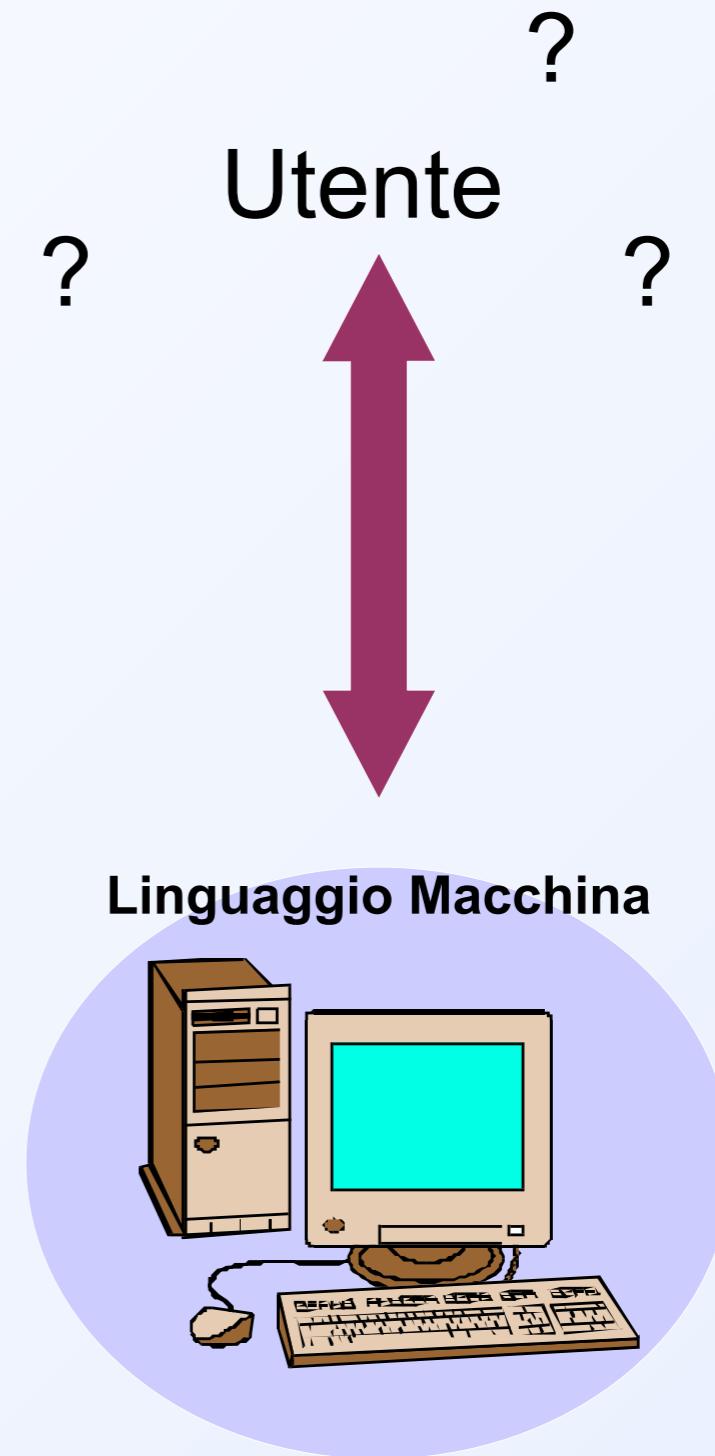
public static void interpret(int memory[],
    int starting_address) {
    PC = starting_address;
    while (true) {
        instr = memory[PC];
        PC = PC + 1;
        instr_type = get_instr_type(instr);
        data_loc = find_data(instr, instr_type);
        if (data_loc >= 0)
            data = memory[data_loc];
        execute(instr_type, data);
    }
}
```

Come faccio a programmare...

- ... un insieme di transistor?
- Programma è sequenza di istruzioni che descrive come portare a termine un certo compito.
- Una CPU capisce Java?
- Se una CPU non può capire Java, come fa a funzionare?
- Perché usiamo Java e non il linguaggio della CPU?

Livelli come macchine virtuali

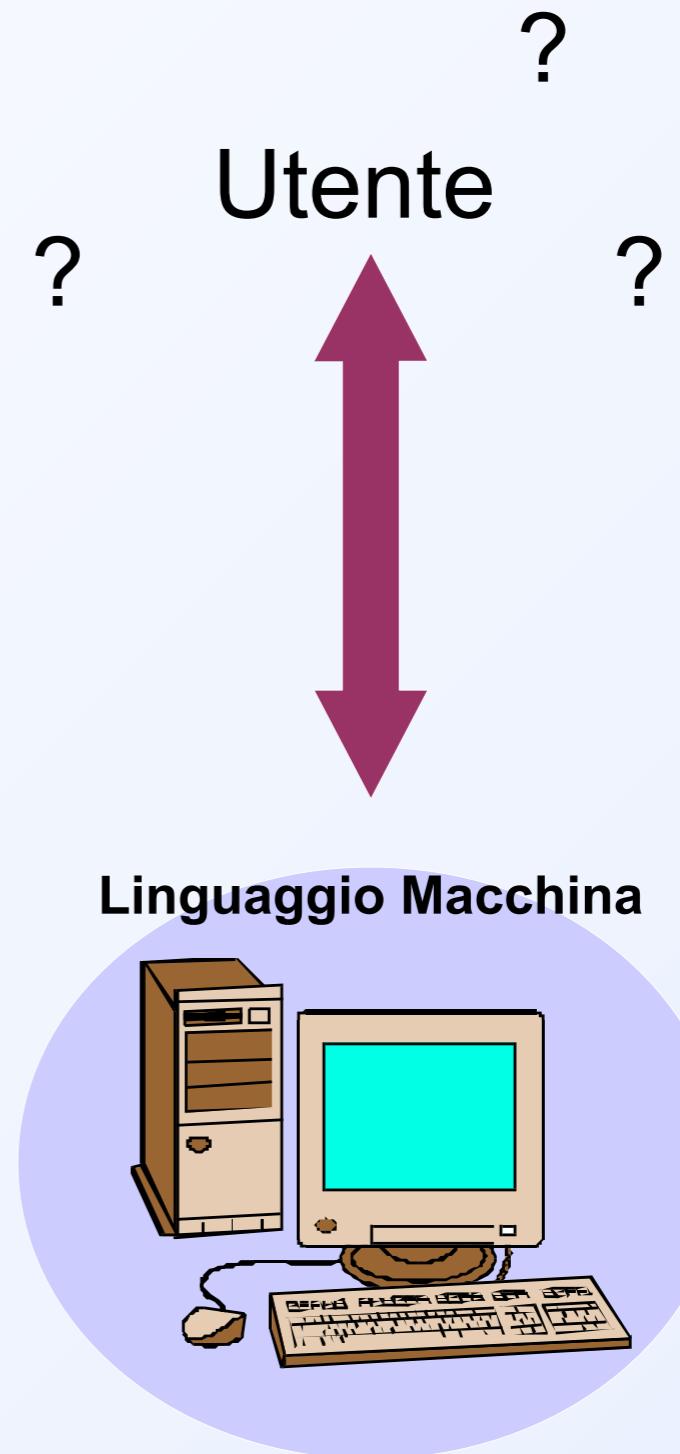
- Un computer è una macchina programmabile, tuttavia esso non è direttamente utilizzabile da parte degli utenti poiché richiederebbe la conoscenza sull'organizzazione fisica della specifica macchina e del suo linguaggio macchina
- Ogni macchina avrebbe le sue differenti caratteristiche
- Il linguaggio macchina è estremamente complicato e non di facile gestione



Livelli come macchine virtuali I

Il linguaggio macchina è estremamente **complicato** e non di facile **gestione**

Cosa vuol dire?

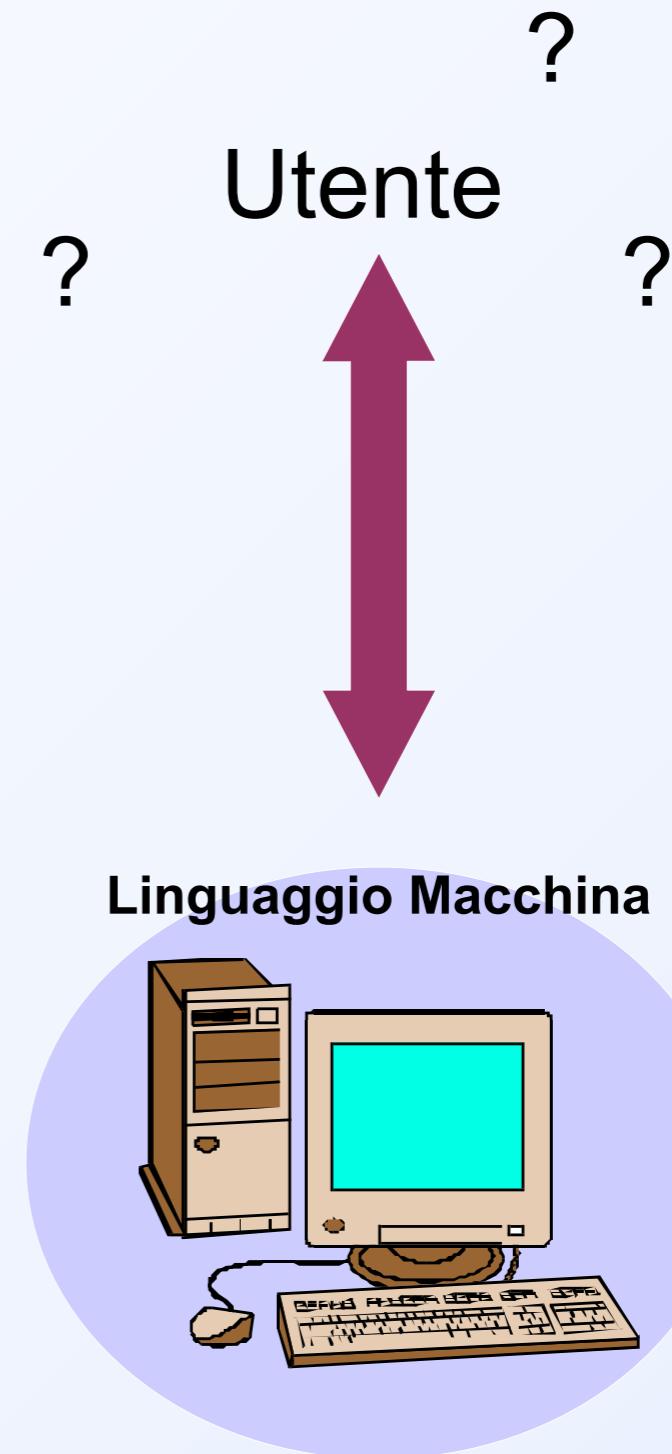


Livelli come macchine virtuali II

Un computer esegue un programma su certi dati.

Il programma è in memoria,
la memoria contiene bit il cui valore rappresenta numeri in base 2,

...



Livelli come macchine virtuali III

Un computer esegue un programma su certi dati.

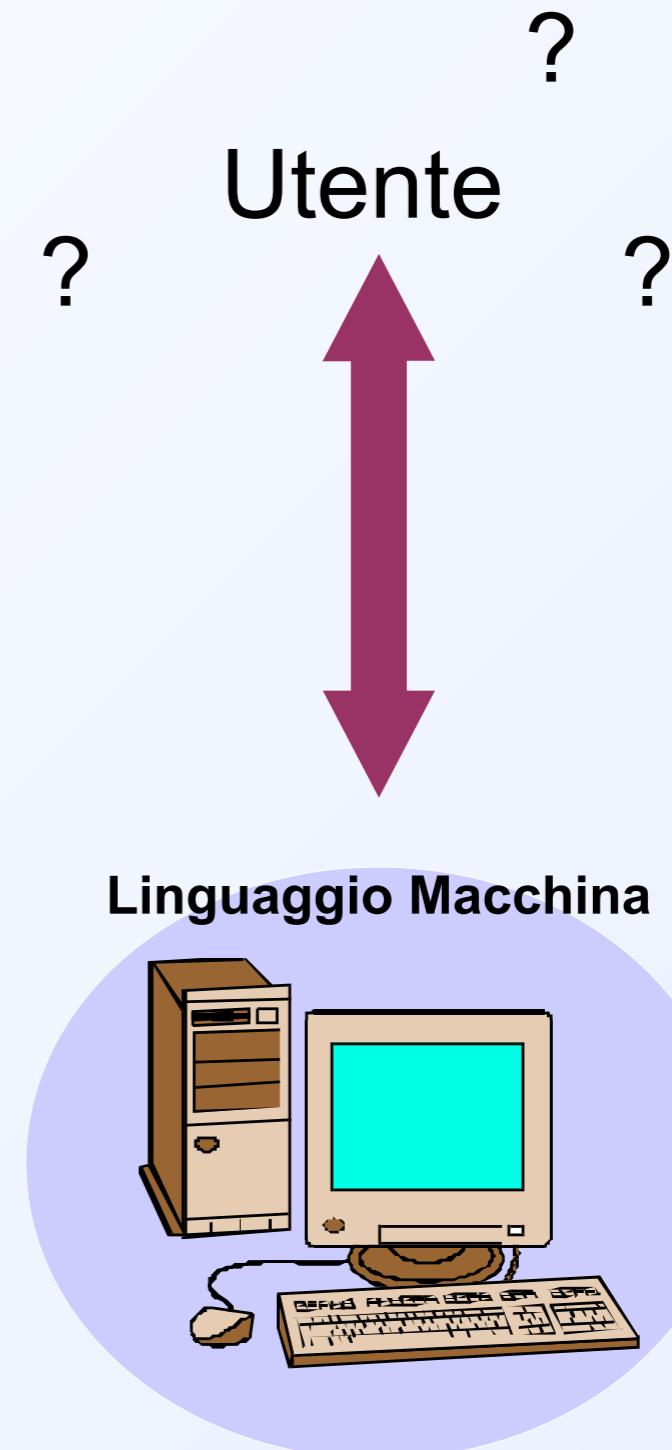
Il programma è in memoria,
la memoria contiene bit il cui valore
rappresenta numeri in base 2,

Un programma è fatto di bit il cui valore
rappresenta numeri in base 2

Come comprenderlo?

Devo dare rappresentazione **simbolica** dei
codici numerici delle istruzioni macchina:

ASSEMBLER



Linguaggio macchina/Assembler

Ling. macchina	Assembler	Significato
0001 0101 0110 1100	LOAD R5 108	M[108] -> R5
0001 0110 0110 1101	LOAD R6 109	M[109] -> R6
0101 0000 0101 0110	ADD R0 R5 R6	R5 + R6 -> R0
0011 0000 0101 1110	STORE R0 110	R0 -> M[110]
1100 0000 0000 0000	Halt	Halt

Livelli come macchine virtuali IV

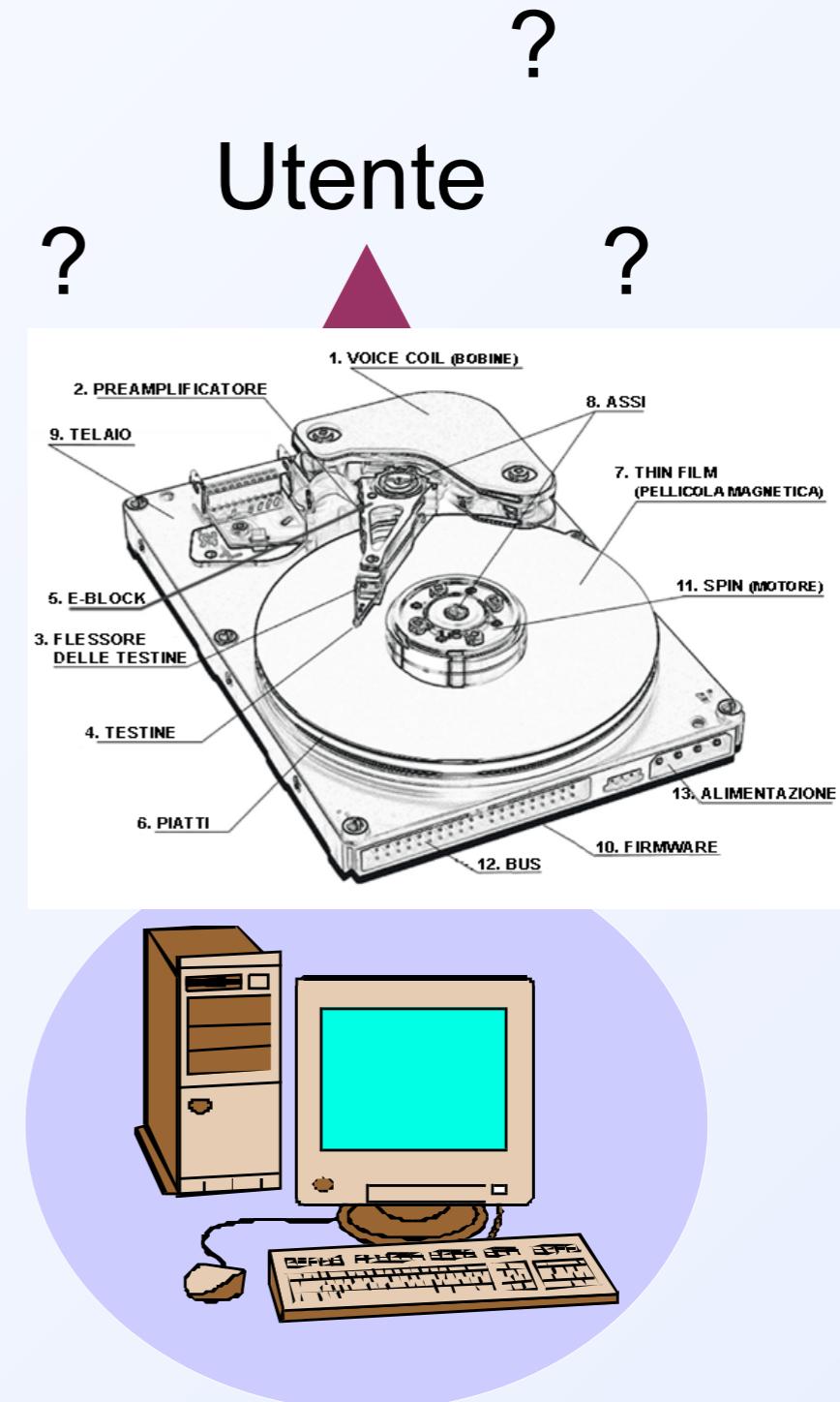
Un programma non fa solo calcoli:
interagisce anche con i **dispositivi di input/output**.

Problemi:

- I dispositivi sono complicati,
- I dispositivi sono diversi anche su macchine con stessa CPU.
- Le operazioni sono le stesse per tutti i programmi (leggere/scrivere file)
- Stesse operazioni su dispositivi di tipo diverso: e.g. file su HD, floppy, USB key, Flash memory, etc. etc.



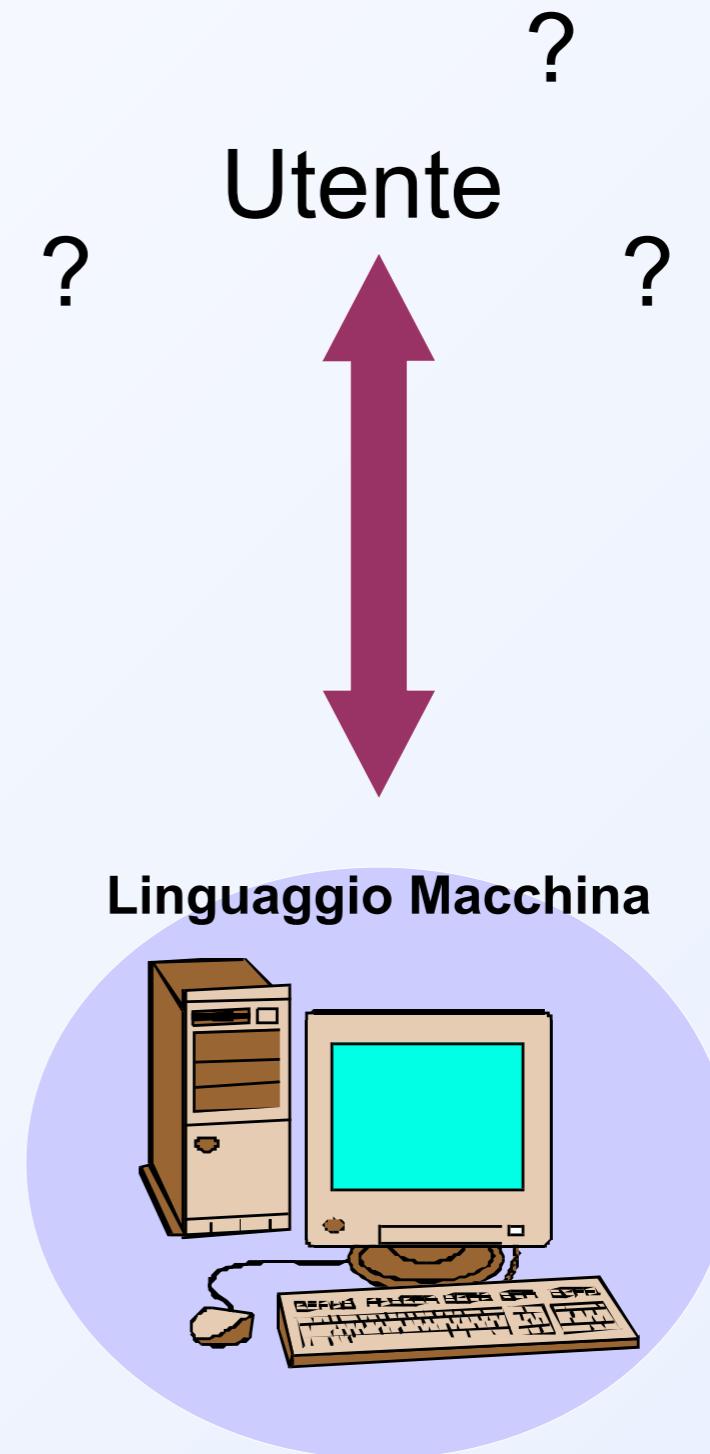
Sistema operativo



Livelli come macchine virtuali V

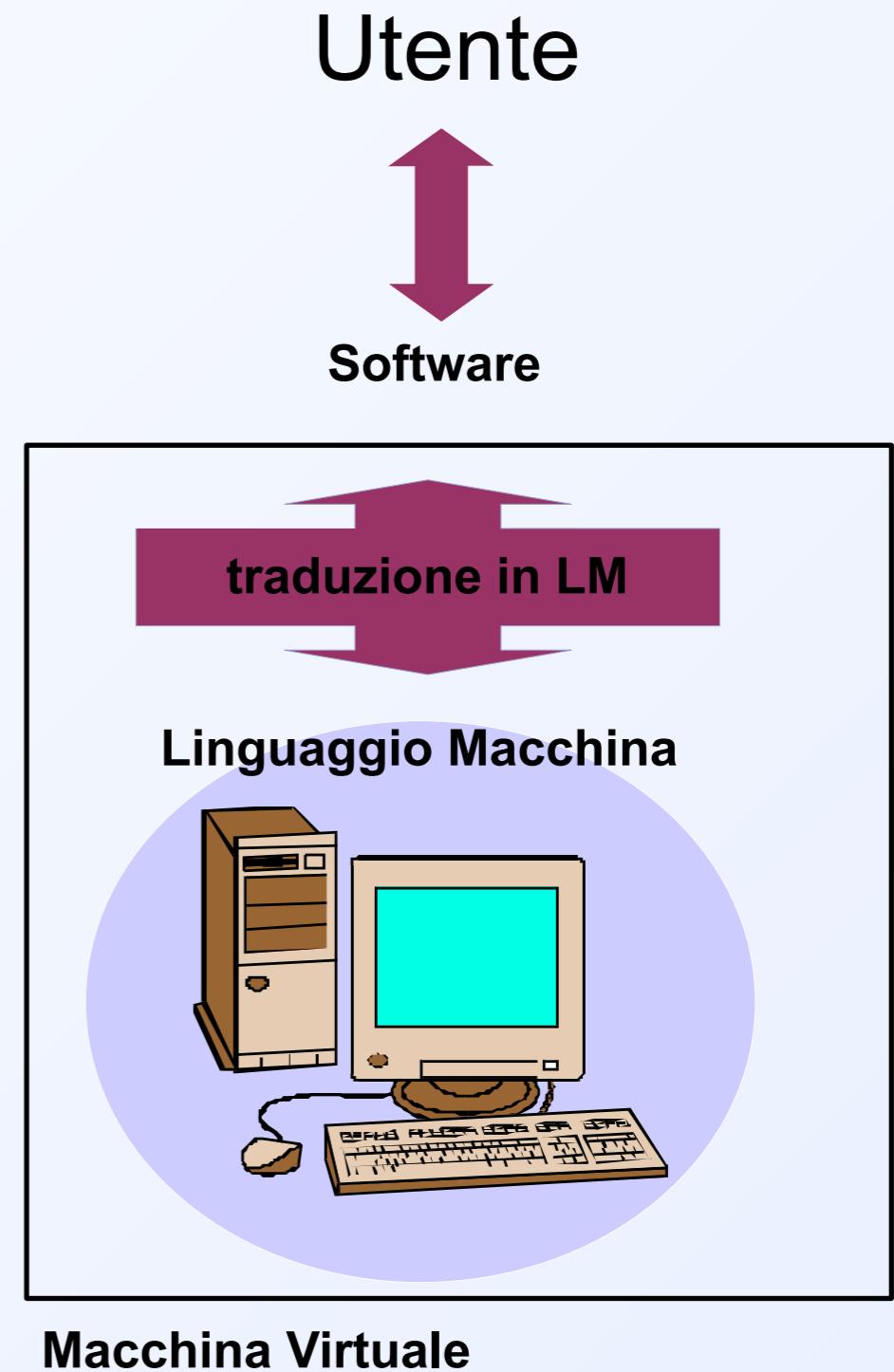
ASSEMBLER è troppo primitivo per programmare. Un linguaggio come Java ha:

- 1) Tipi di dati complessi
- 2) Controllo dei tipi
- 3) Programmazione strutturata (while, for vs go to)
- 4) Polimorfismo
- 5) ...



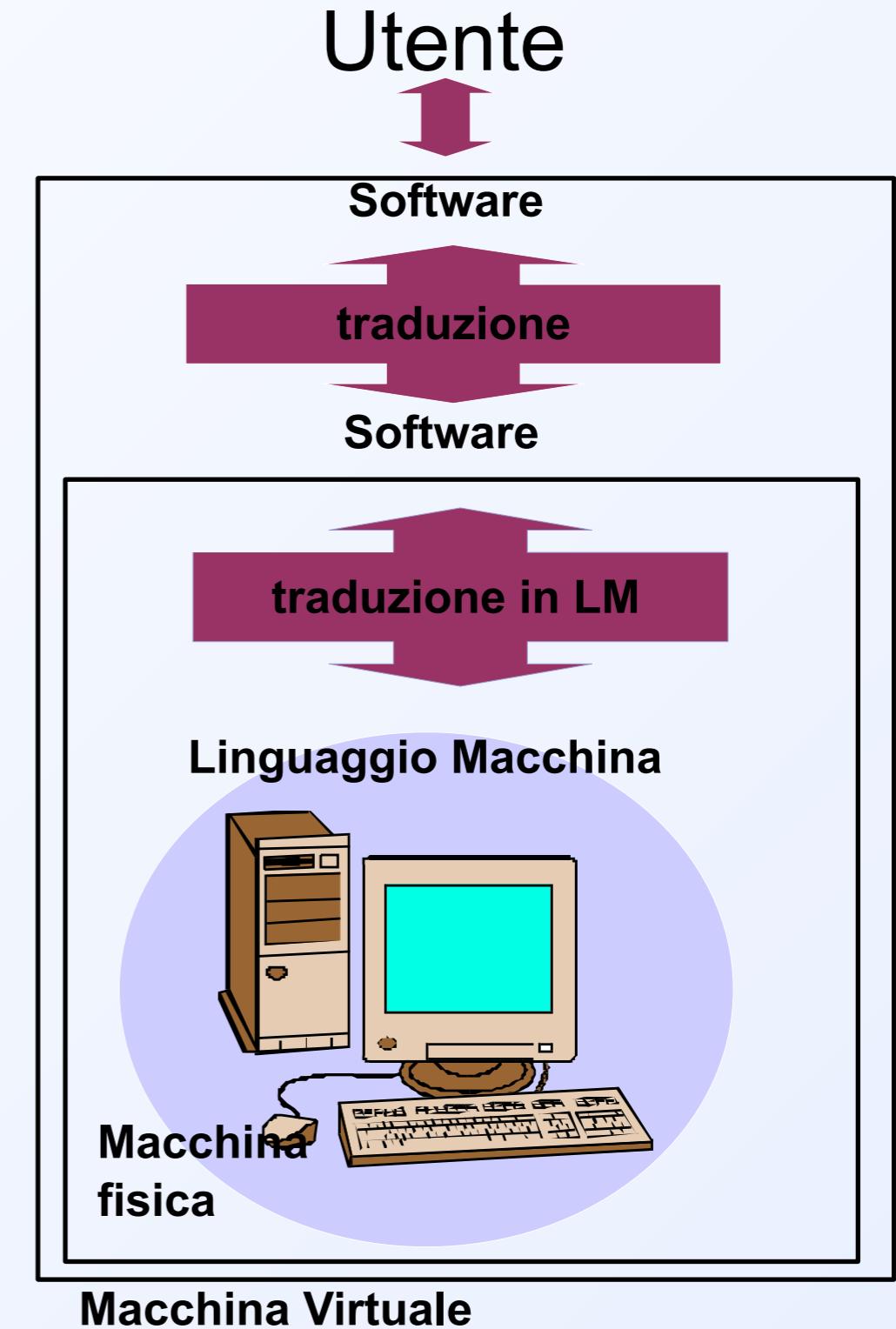
Livelli come astrazioni

- In altre parole desideriamo **astrarci** dai dettagli fisici della macchina in oggetto e dal suo specifico linguaggio macchina
- L'idea è quella di realizzare al di sopra della macchina reale una **macchina virtuale astratta** che abbia le funzionalità desiderate e che sia facile da utilizzare per l'utente
- L'utente interagisce con la macchina virtuale, ogni comando viene poi **tradotto** nei corrispondenti comandi sulla macchina fisica
- La macchina virtuale è realizzata mediante software (programmi)



Livelli come astrazioni

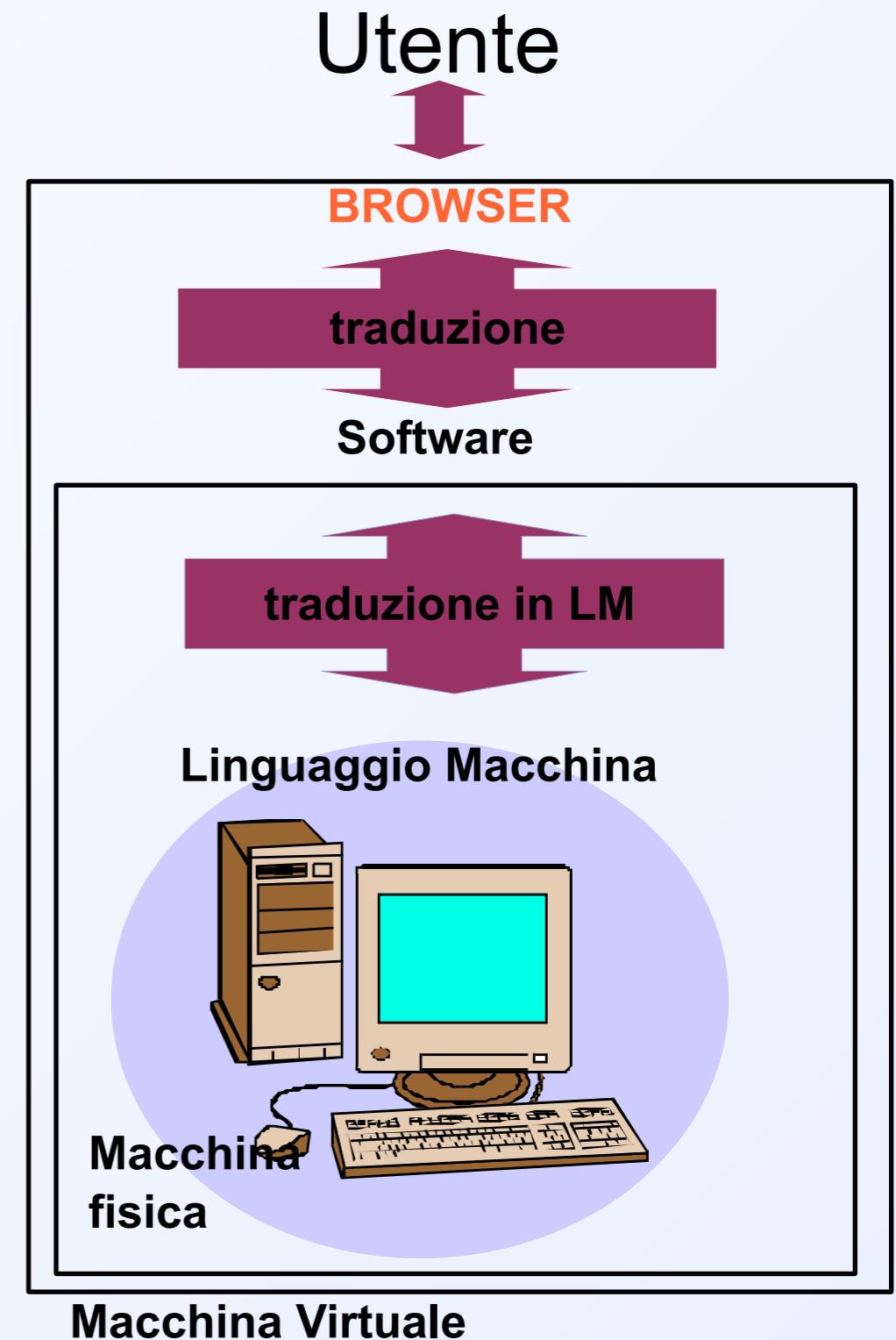
- **La macchina virtuale viene realizzata in genere mediante il software di base:**
 - **Sistema Operativo: file system, memoria, cpu, risorse ausiliarie, comunicazione**
 - **Linguaggi e ambienti di programmazione ad alto livello: interpreti e compilatori**
- **Non vi sono limiti al numero e al tipo di macchine virtuali che possono essere realizzate**
- **In genere nelle macchine moderne sono strutturate su più livelli (struttura a cipolla)**



Livelli come astrazioni

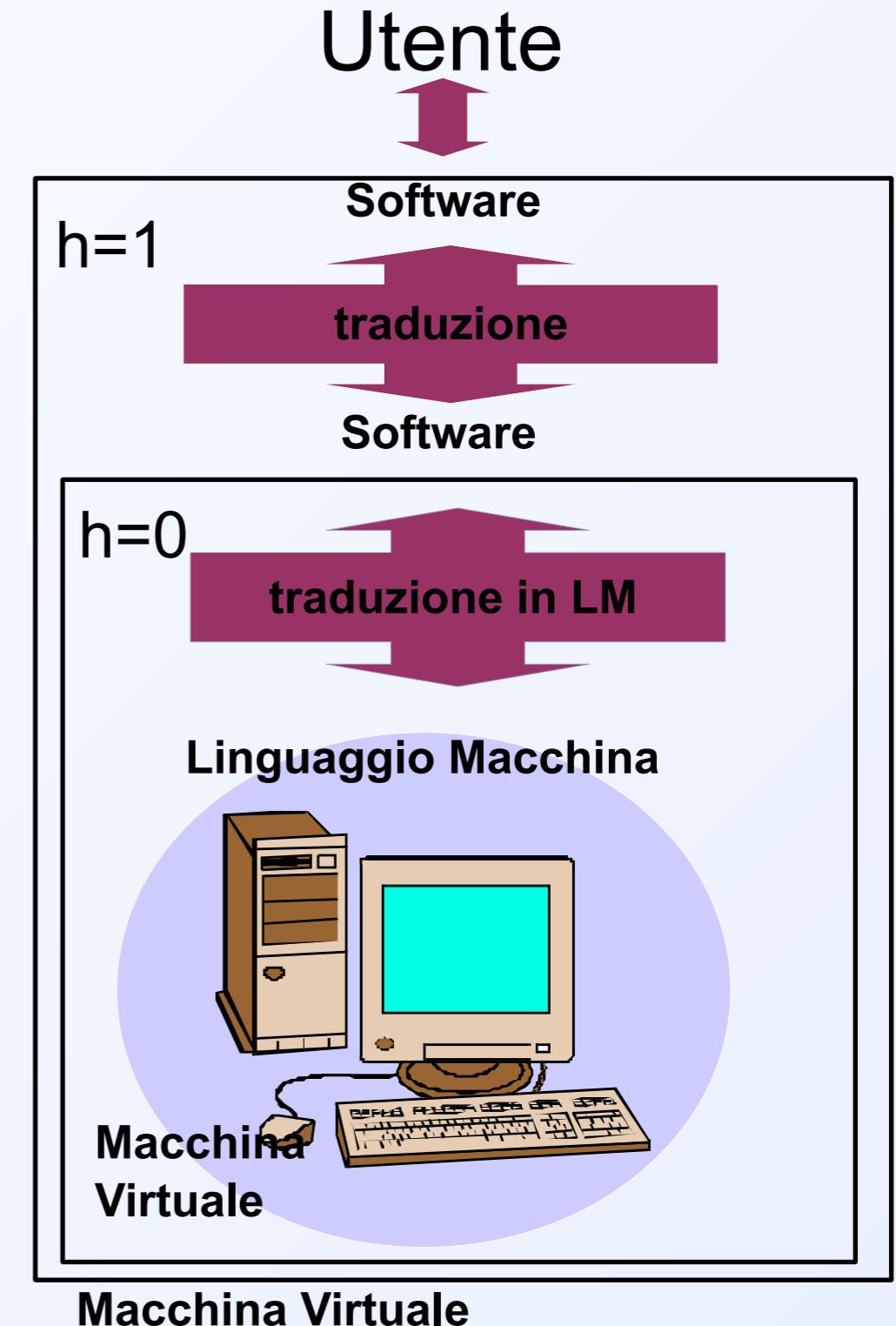
Un browser è un programma

- Un browser interpreta il linguaggio HTML e visualizza le pagine sullo schermo
- Per la visualizzazione delle pagine un browser si appoggia al software di base
- Il software di base si appoggia alla macchina fisica per realizzare effettivamente il “rendering” della pagina su video



Livelli come astrazioni

- ✗ *In termini un po' più formali, detto:*
 - ✓ *I_h l'insieme delle istruzioni che costituiscono il linguaggio LM della macchina virtuale del livello h*
 - ✓ *M_h l'insieme delle istruzioni utilizzabili al livello h, ma mascherate nei confronti dei livelli superiori*
 - ✓ *C_h l'insieme dei comandi implementati a livello h utilizzando il linguaggio macchina I_h*
- ✗ *Il linguaggio macchina I_{h+1} della macchina virtuale di livello h+1 puo' essere definito nel modo seguente:*
 - ✓ $I_{h+1} = I_h + C_h - M_h$

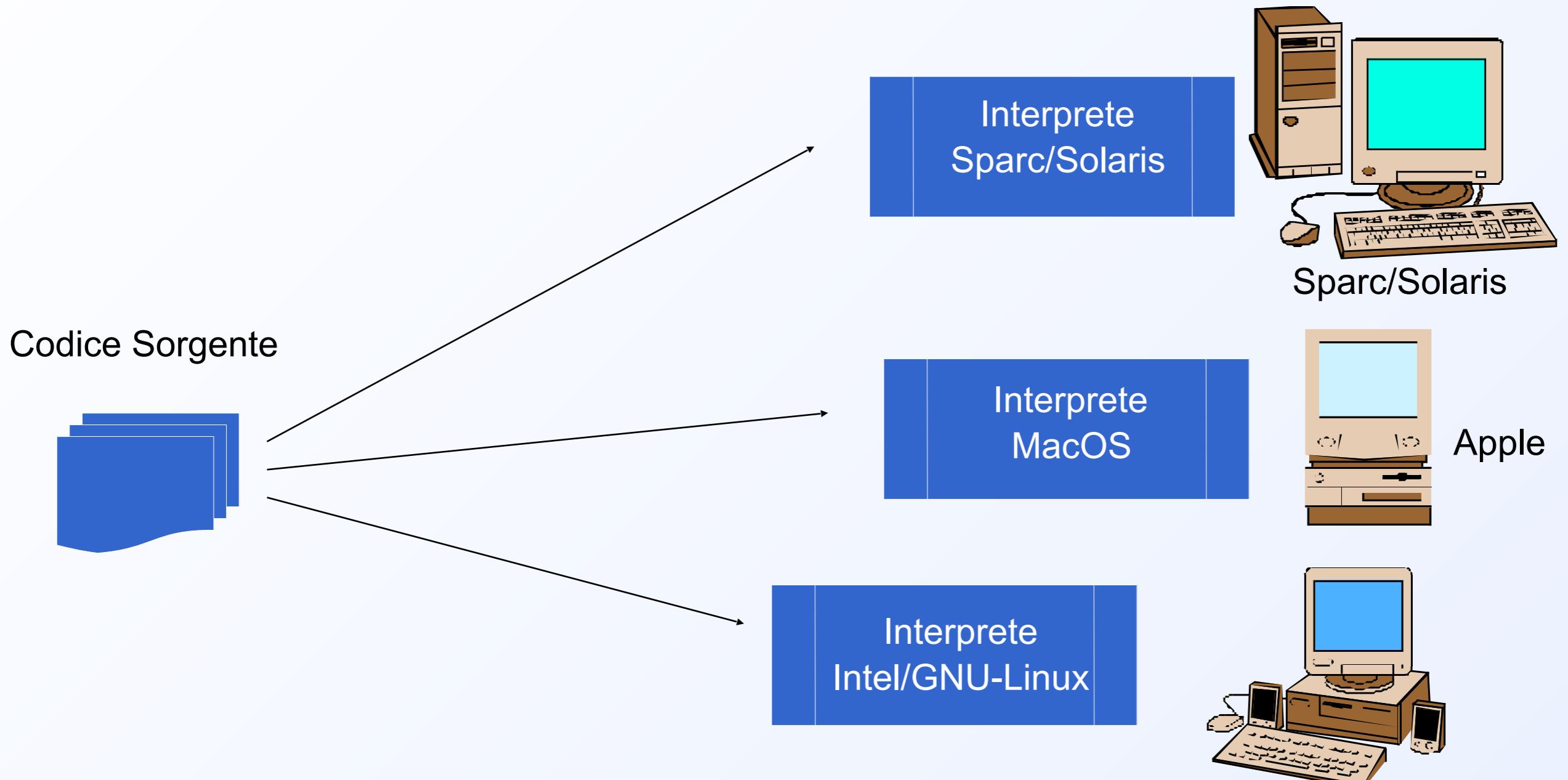


Come si passa da un livello all'altro?

Due metodi:

- **Interpretazione**
- **Compilazione**

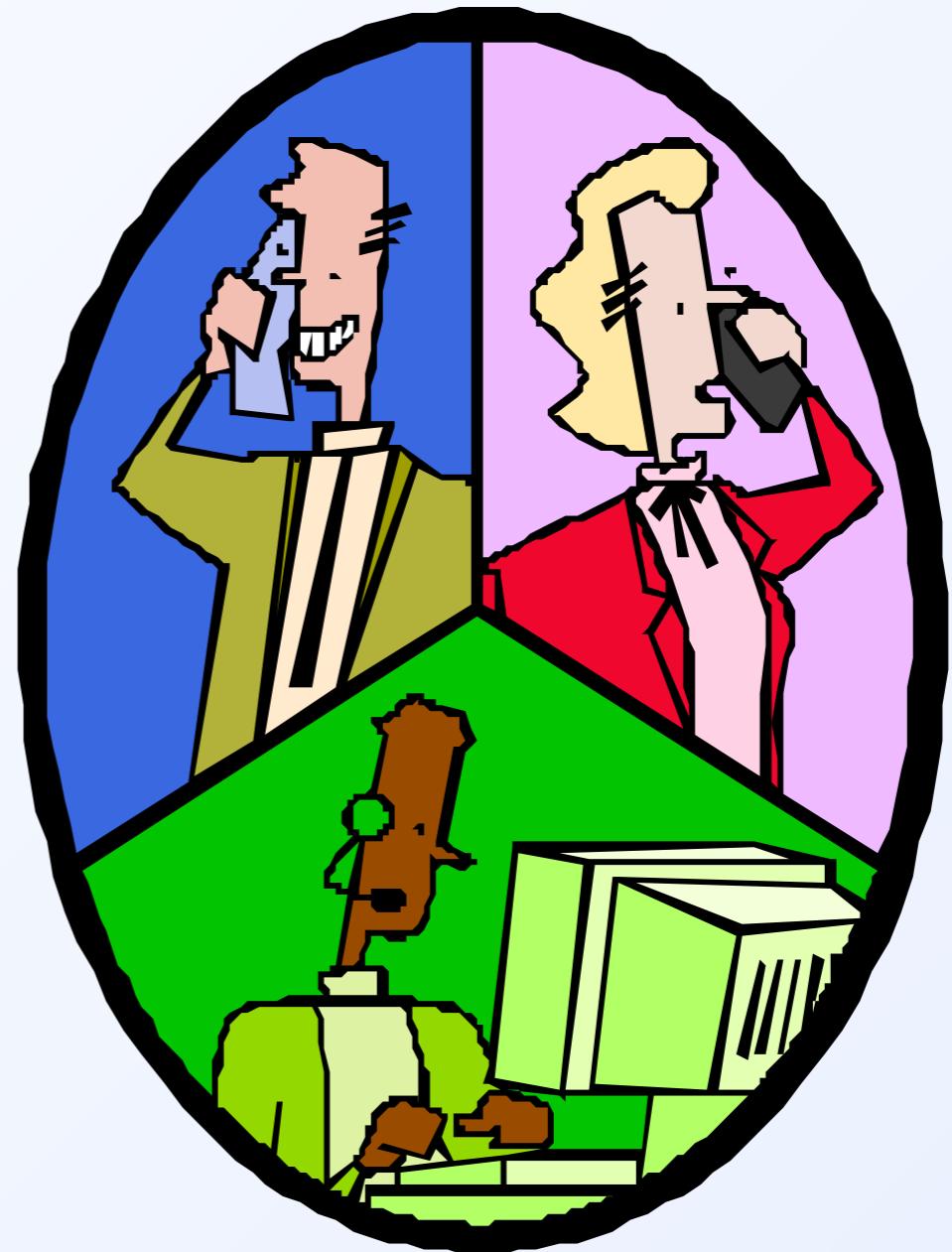
Interpretazione



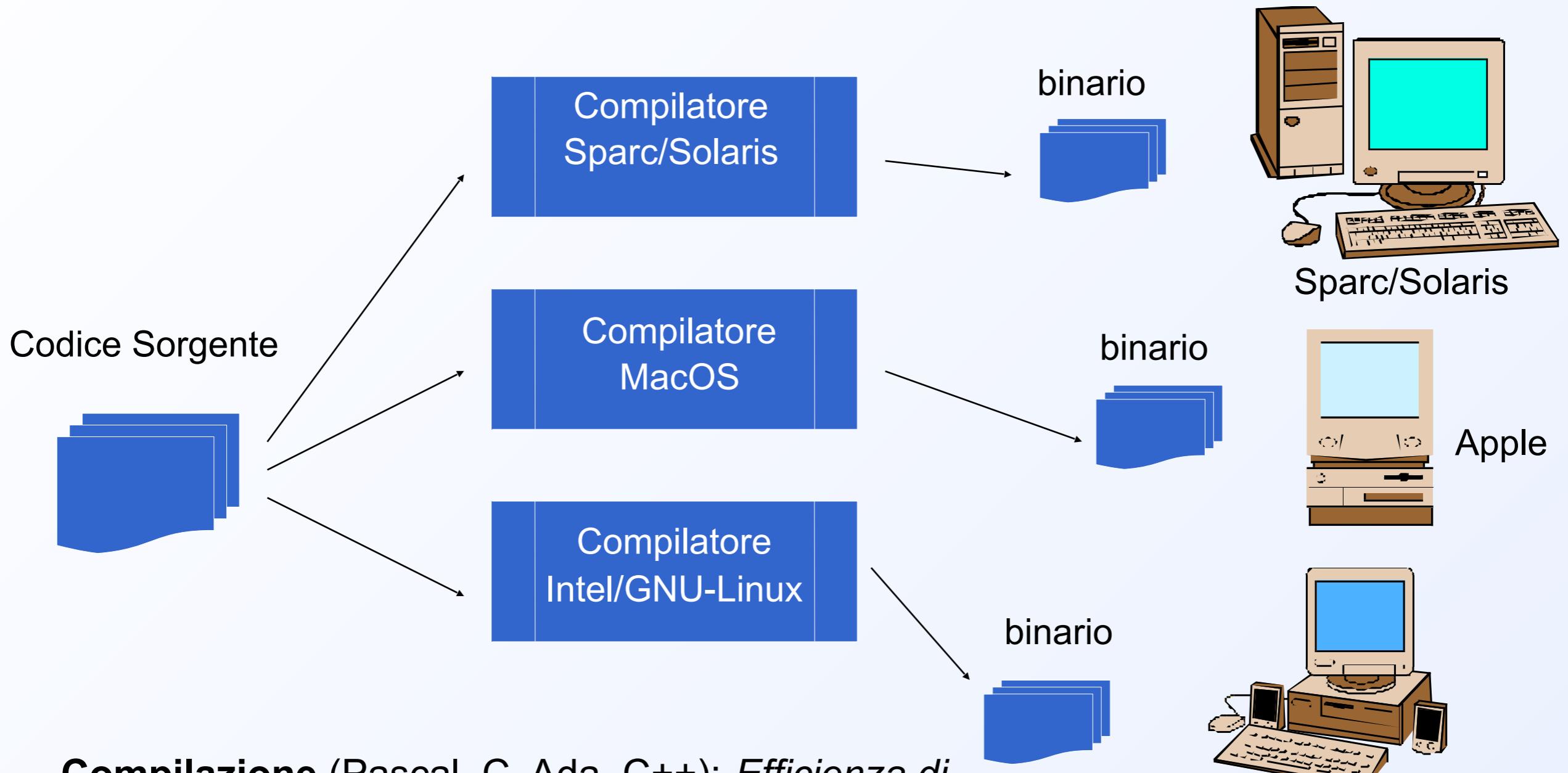
Interprete (JavaScript, Scheme, Prolog, php, asp):
Interazione. Più facile modificare un programma durante l'esecuzione

Interpretazione

- Affinché le due persone di lingua diversa possano dialogare tra di loro (nel caso nessuna delle due conosca la lingua dell'altro) è necessario che qualcuno interpreti (traduca sul momento) quanto dice una persona nella propria lingua nella lingua di chi ascolta
- Si interpreta quando è necessario una stretta interazione, quando si desidera dialogare e non solo trasmettere un messaggio



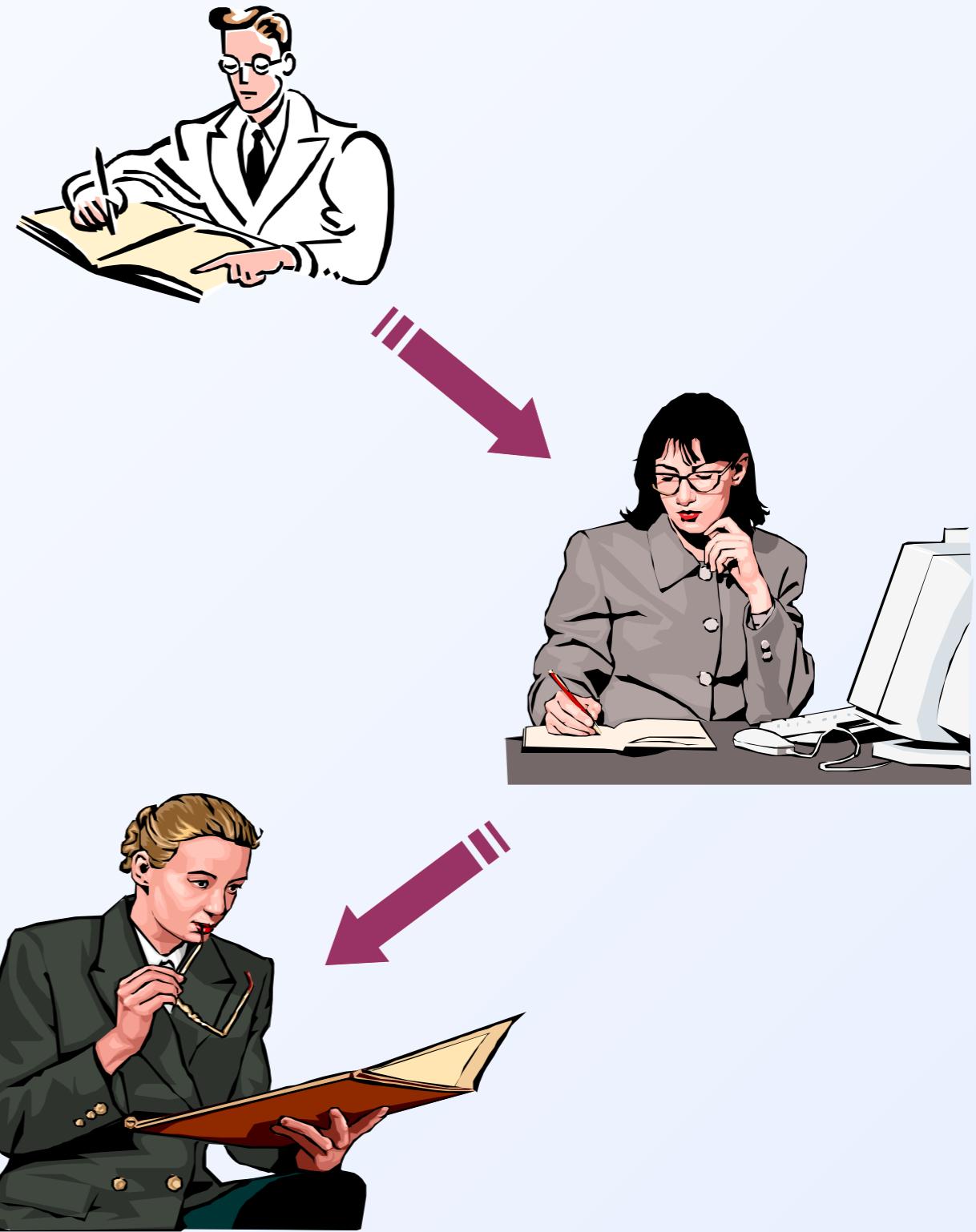
Compilazione



Compilazione (Pascal, C, Ada, C++): *Efficienza di esecuzione*. Il codice generato dal compilatore può essere ottimizzato, perché la compilazione è fatta una sola volta

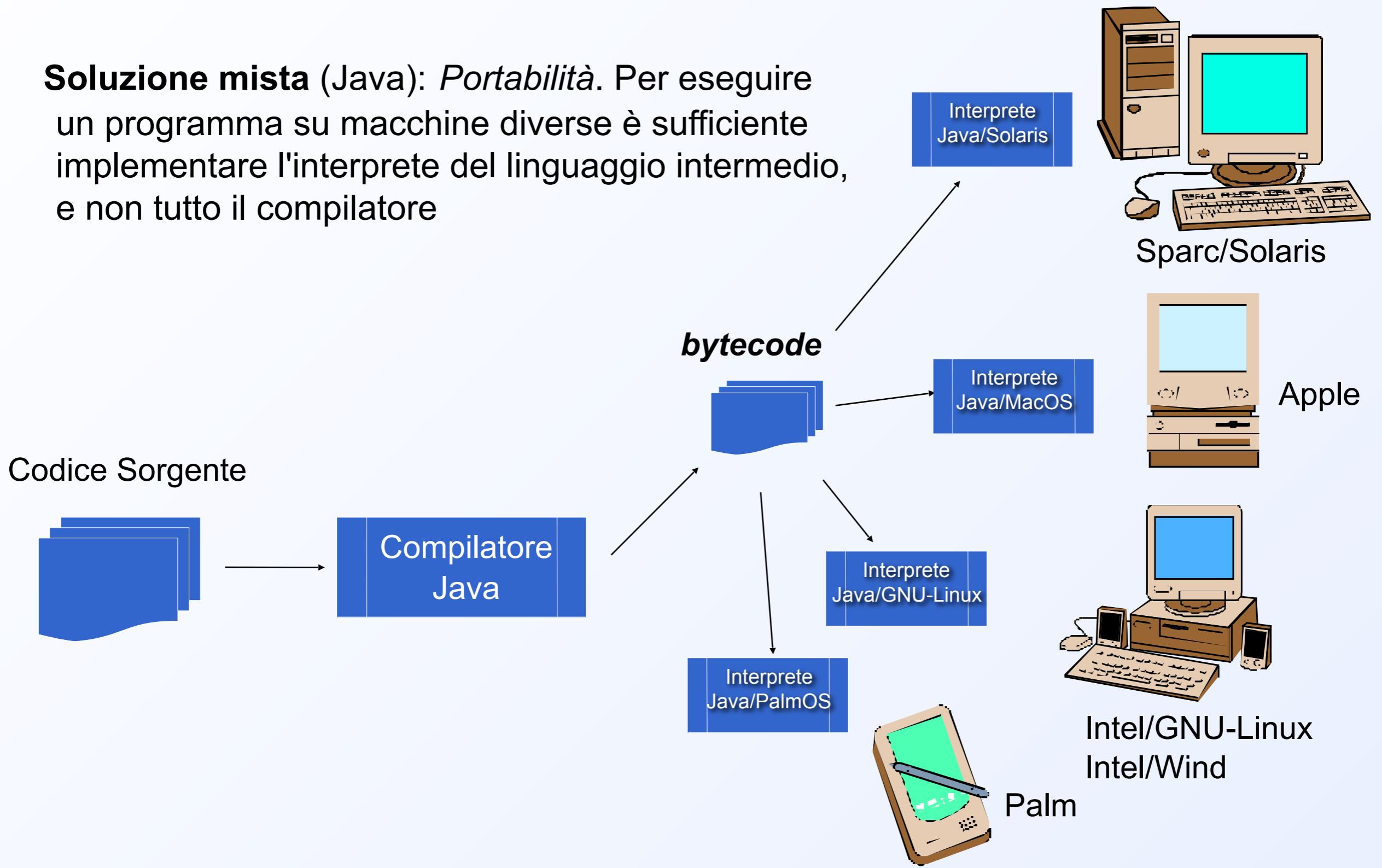
Interpretazione o compilazione

- ***La traduzione è adatta per comunicare un messaggio, come ad esempio una lettera***
- ***Anche per effettuare una traduzione è necessario l'intervento di qualcuno che sia in grado di comprendere le frasi di un linguaggio e riportarle in un altro ma questo può operare in tempi separati rispetto la scrittura del messaggio e la sua lettura***
- ***La lettura è più rapida e semplice, il traduttore ha senz'altro avuto tempo per meglio adattare il testo***
- ***ma si penalizza l'interattività***



Soluzione mista: Java

Soluzione mista (Java): Portabilità. Per eseguire un programma su macchine diverse è sufficiente implementare l'interprete del linguaggio intermedio, e non tutto il compilatore



JVM

Istruzioni del bytecode interpretato dalla JVM:

- IADD
- ISUB
- ILOAD x (nome variabile locale, spiazzamento)
- ISTORE y
- BIPUSH a (valore 8 bit)
- GOTO L (etichetta istruzione, spiazzamento)
- IFCMPEQ L

Da Java a JVM

y = x + 1;

compilatore

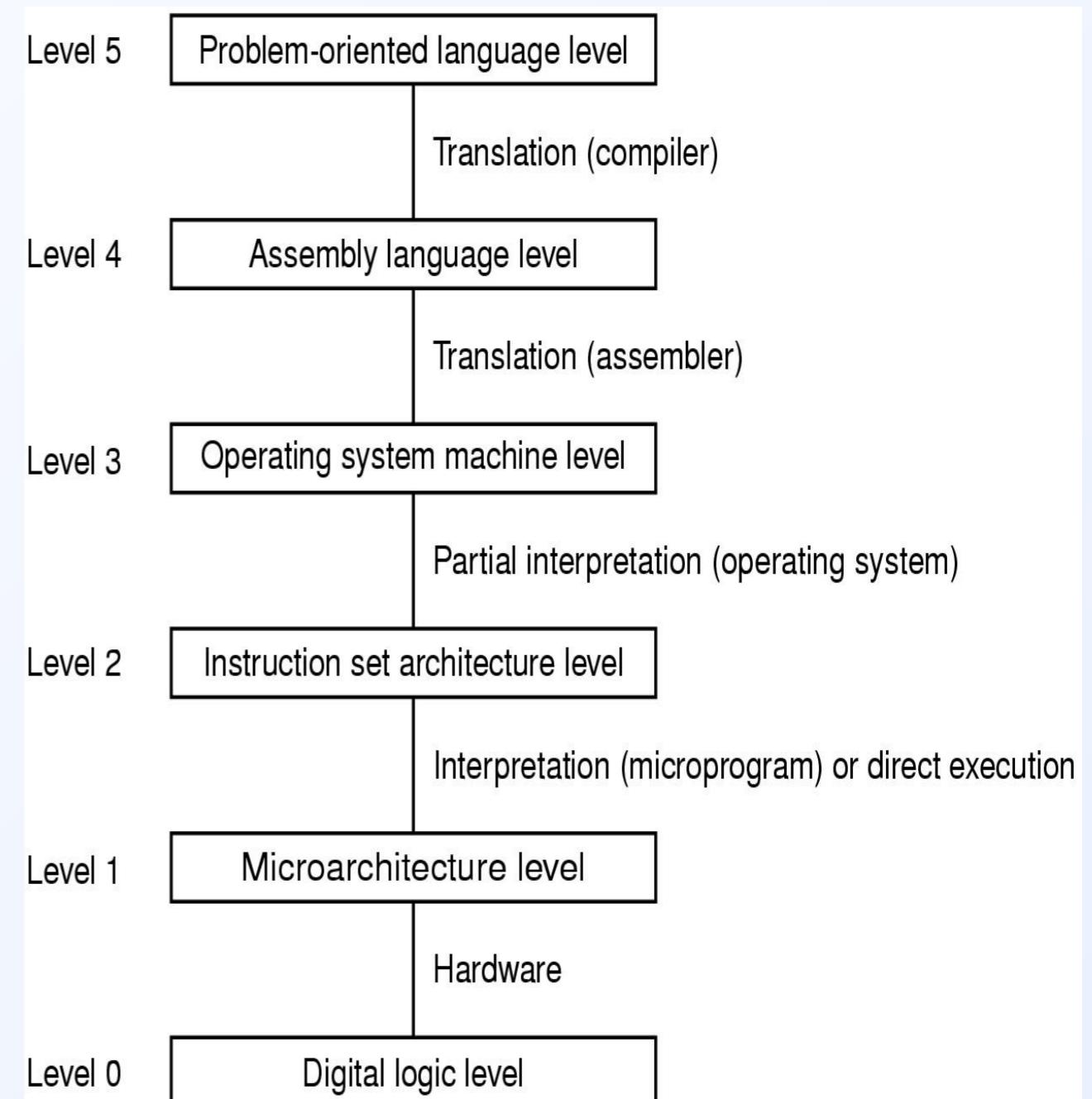
```
ILOAD x  
BIPUSH 1  
IADD  
ISTORE y
```

Ognuna delle istruzioni del livello IJVM può in realtà essere ulteriormente analizzata (dall'interprete):

ILOAD x: preleva valore dalla memoria alla posizione x e mettilo sul top dello stack, alza lo stack;

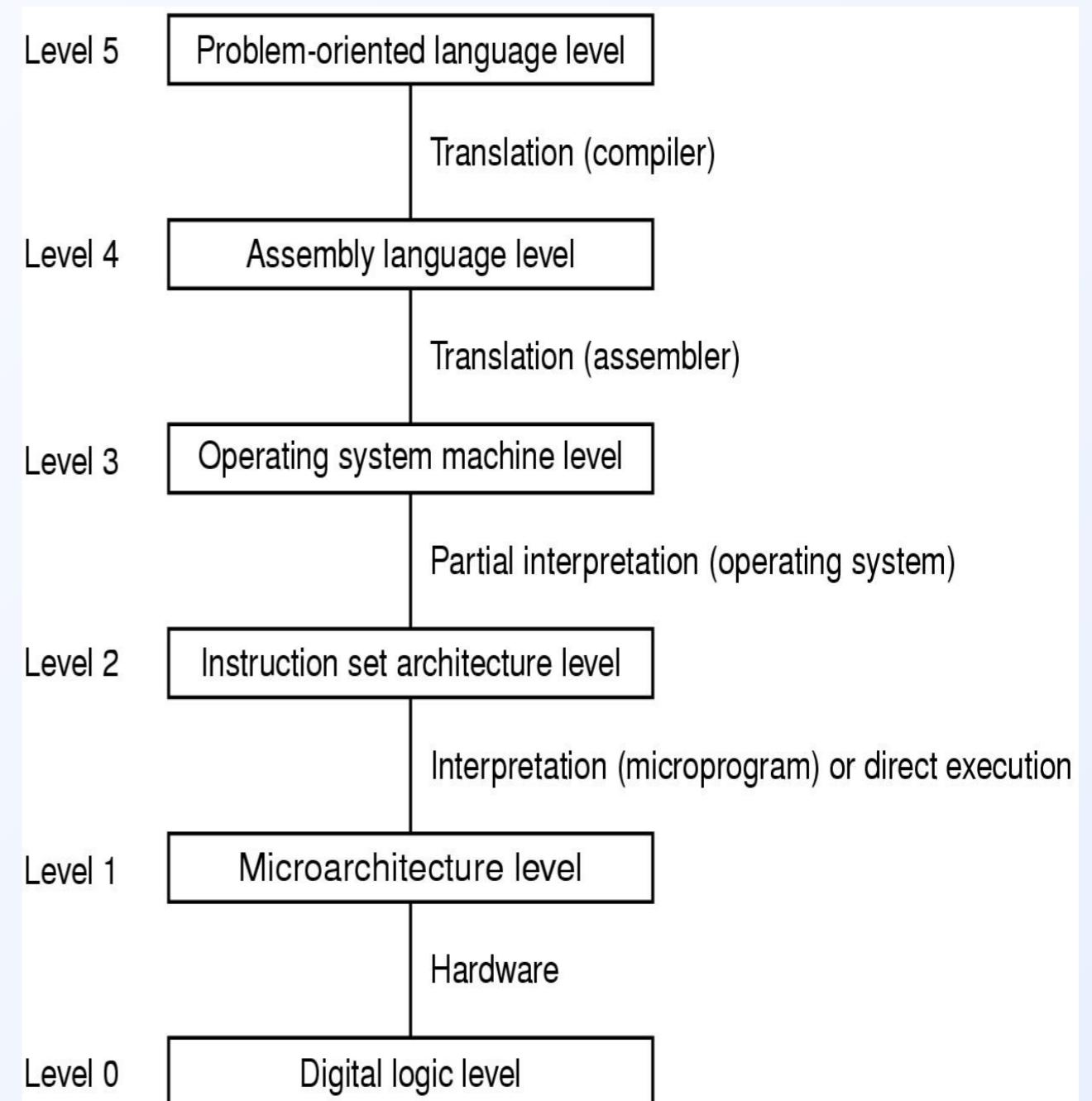
Organizzazione a livelli

- La maggior parte dei moderni computer consiste di 2 o più livelli, generalmente 6.
- Il **livello 0** rappresenta l'hardware della macchina i cui circuiti eseguono i programmi scritti nel linguaggio macchina del livello 1.
- Al livello **logico-digitale** ci sono le **porte**, costituite da alcuni transistor, dotate di 1 o più input digitali (segnali corrispondenti ai valori 0 e 1) che calcolano semplici funzioni dei valori in ingresso. Le porte si possono combinare per formare una **memoria da 1 bit**. Queste memorie si possono a loro volta combinare per formare i **registri**, memorie in grado di memorizzare numeri di valore variabile.



Organizzazione a livelli

- Al livello 1, quello della **microarchitettura**, vi è un gruppo di registri e un circuito chiamato **ALU**, capace di effettuare semplici operazioni aritmetiche. I registri sono connessi alla ALU per formare il **percorso dati** lungo il quale i dati si spostano. In alcune macchine le operazioni del percorso dati sono controllate da un programma: il **micropogramma**, mentre in altre è controllato direttamente dall'**hardware**. In passato era quasi sempre rappresentato da un interprete software, oggi invece è spesso controllato in modo diretto dall'hardware.



Organizzazione a livelli

- Il **livello 2** è il livello **ISA (Instruction Set Architecture)**: è il livello del linguaggio macchina. I manuali che descrivono le istruzioni macchina presentano le istruzioni di questo livello, eseguite in modo interpretato dal microprogramma o dai circuiti elettronici.
- La maggior parte delle istruzioni del **livello 3** fa parte anche del livello ISA; vi sono inoltre nuove istruzioni, diversa organizzazione della memoria, capacità di eseguire i programmi in modo concorrente, ...

