Capitolo 1

Macchine astratte

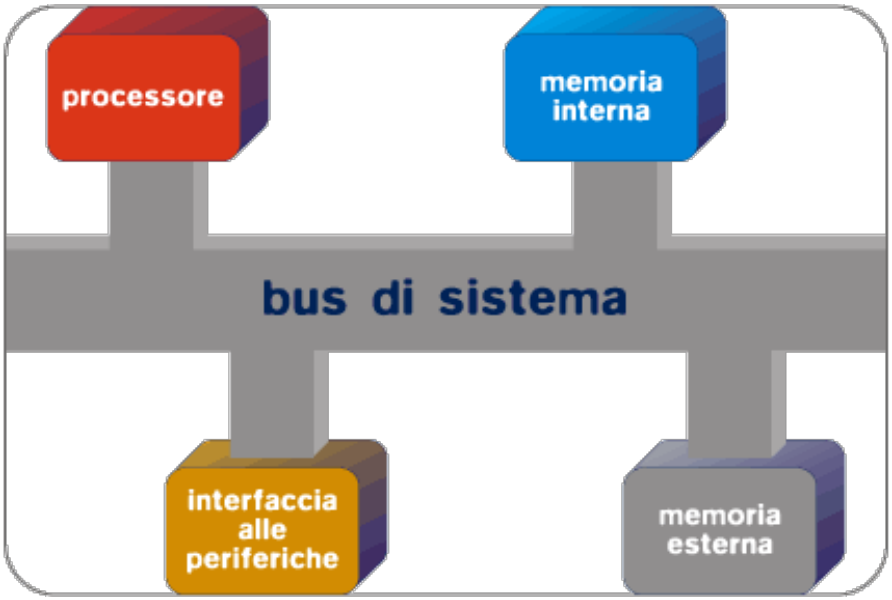
Interpreti e Compilatori

Linguaggi, macchine astratte, implementazioni

- **Argomenti**:
  - Macchine astratte
  - Implementare un linguaggio: Interpreti e Compilatori
  - Gerarchie di macchine astratte

la macchina di von Neumann

- Le tecnologie sono basate su una struttura estremamente semplice



...ma anche estremamente flessibile e suscettibile di molte varianti

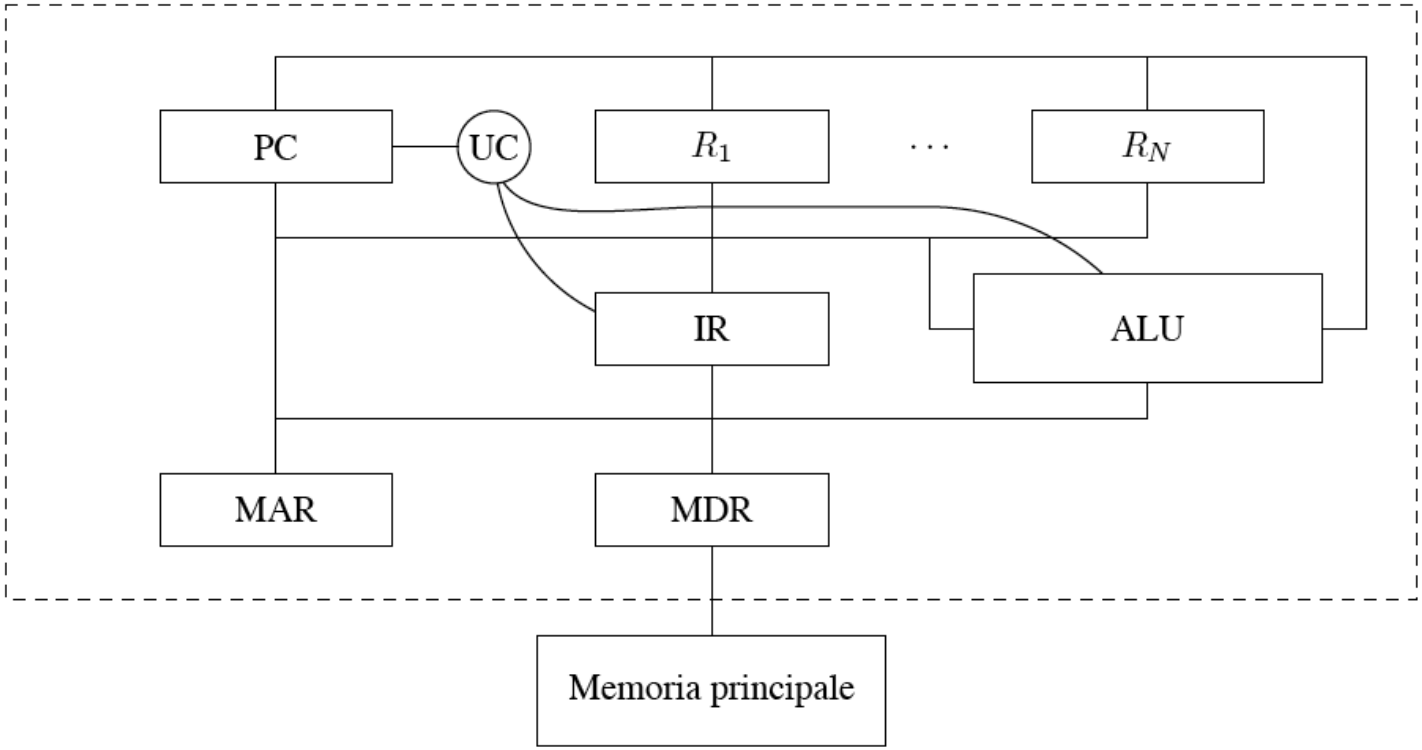
…e il suo “interprete”

- La macchina esegue un ciclo ripetitivo

programmi e dati sono indistinguibili
e risiedono nella memoria interna

La macchina esiste per eseguire il proprio linguaggio

il processore tipico

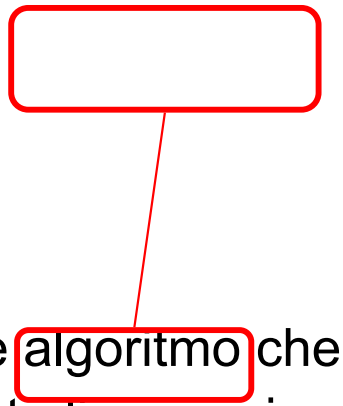


può essere complicato a piacere:

- architetture parallele
- con *piping*
- con *prefetch*

Morale

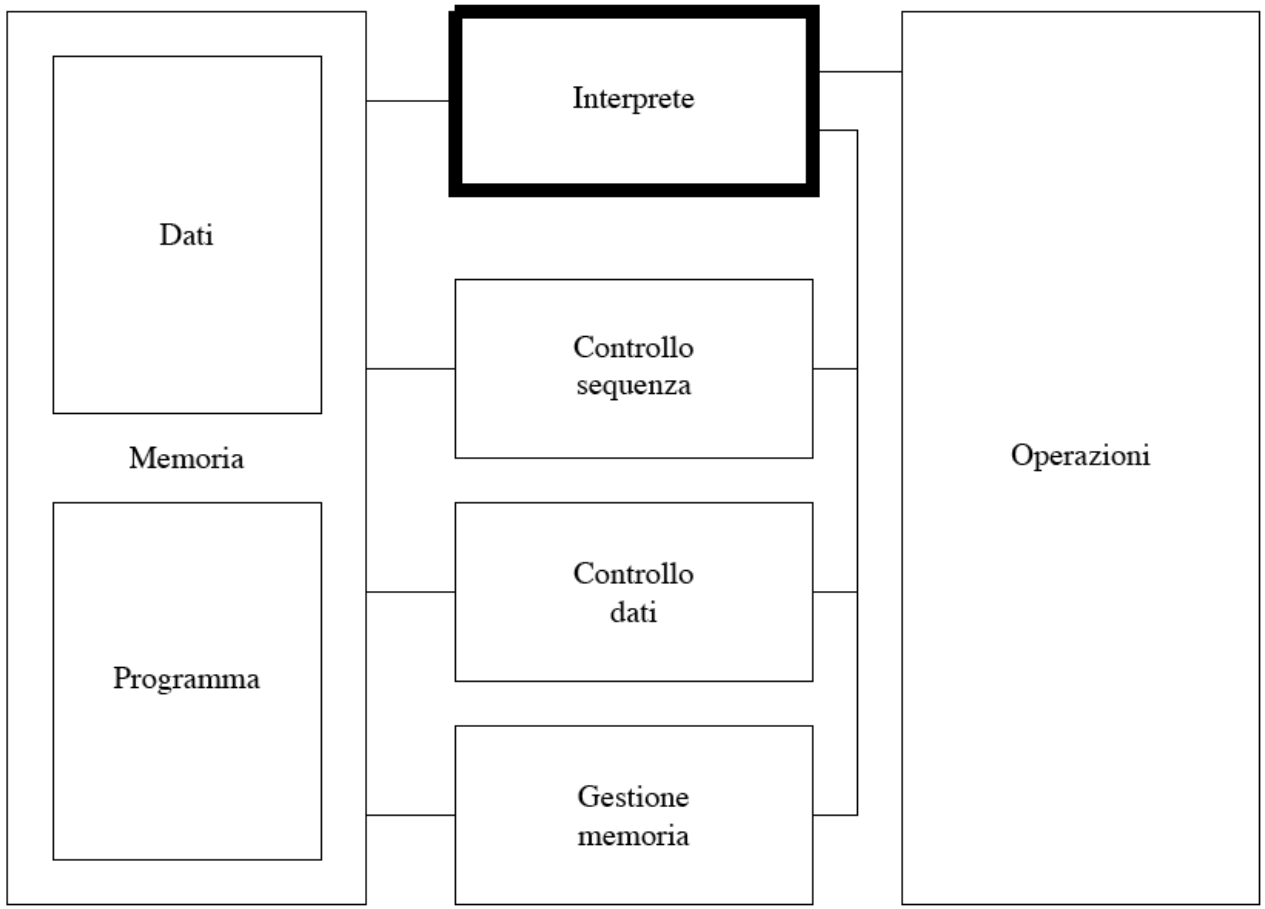
- Una macchina fisica esiste per eseguire il suo linguaggio
- Linguaggio e macchina esistono in simbiosi. Ma:
  - una macchina corrisponde ad un linguaggio (il suo)
  - un linguaggio può essere eseguito da più macchine
- Cuore di una macchina fisica:
  - il ciclo fondamentale *fetch-decode-execute*: l'interprete



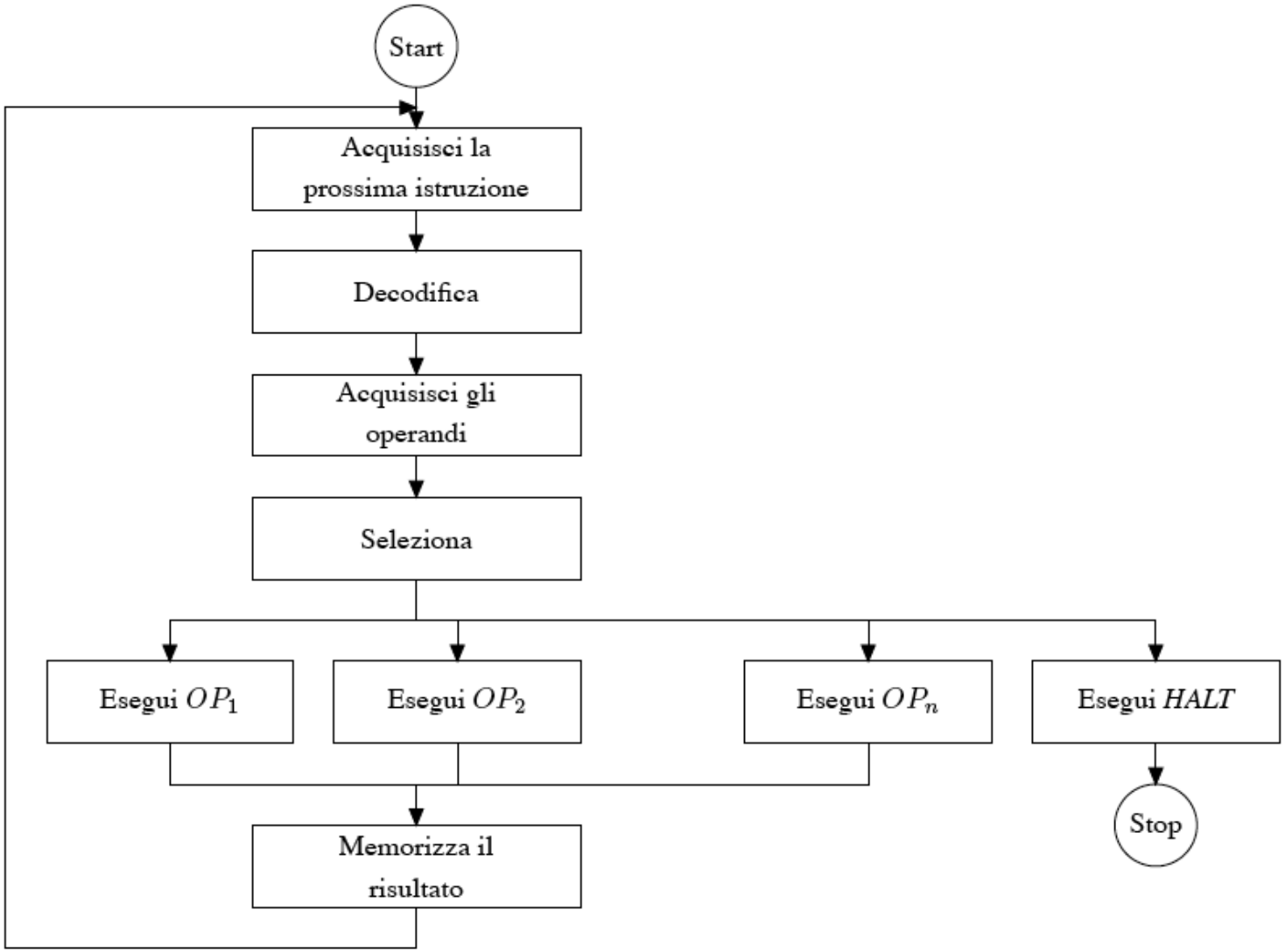
Una macchina fisica è la realizzazione "a fili" di un particolare algoritmo che, sfruttando alcune strutture dati, è capace di "eseguire" un certo linguaggio

Macchina astratta

- Insieme di strutture dati ed algoritmi in grado di memorizzare ed eseguire programmi. Il componente essenziale è l'interprete



Interprete


Start
Acquisisci la prossima istruzione
Decodifica
Acquisisci gli operandi
Seleziona
Esegui OP_1
Esegui OP_2
Esegui OP_n
Esegui *HALT*
Stop
Memorizza il risultato

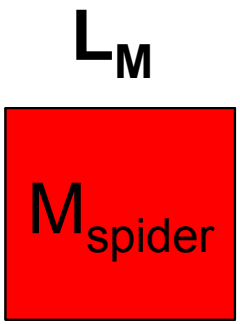
Linguaggio Macchina

- **M** Macchina Astratta
- $\mathbf{L_M}$ Linguaggio Macchina di M: linguaggio che è "compreso" dall'interprete di **M**:
- Ai componenti di **M** corrispondono opportuni componenti di $\mathbf{L_M}$
- Esistono diverse rappresentazioni dei programmi scritti in $\mathbf{L_M}$ (interna ed esterna)

la macchina hw

- Una processore convenzionale è una (forma molto concreta di...) macchina astratta

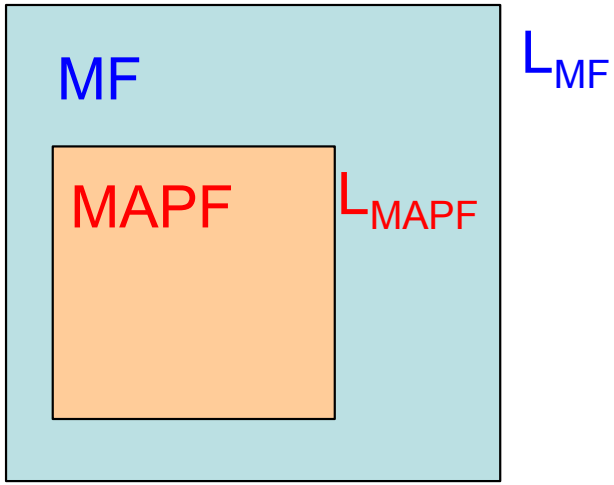
- Il suo linguaggio è il linguaggio macchina



cambia l'architettura,
cambia l'interprete,
ma non il linguaggio:
$L_M = L_{Mspider}$

Un'architettura più sofisticata può interpretare lo stesso linguaggio macchina

Scatole cinesi



- Alcune macchine hw (eg non RISC) sono *microprogrammate*:
  - ciclo fetch-execute di MF non è realizzato in hardware;
  - ogni istruzione di L_{MF} è realizzata mediante istruzioni di più basso livello (microistruzioni)
  - interpretata da un microinterprete

Un programma in linguaggio macchina (in L_{MF}):
- viene interpretato da un interprete (scritto in L_{MAPF})
- a sua volta eseguito dal (micro)interprete (hw) di MAPF

Realizzare una macchina astratta

Possiamo realizzare una MA:

1) in HARDWARE
   - Usata solo per macchine di basso livello o macchine dedicate
   - massima velocità, flessibilità nulla.

2) mediante *emulazione* o simulazione FIRMWARE
   - strutture dati e algoritmi MA realizzati da microprogrammi
   - alta velocità, flessibilità maggiore che HW puro.

3) mediante simulazione SOFTWARE
   - strutture dati e algoritmi MA realizzati da programmi
   - minore velocità, massima flessibilità, macchina ospite qualsiasi

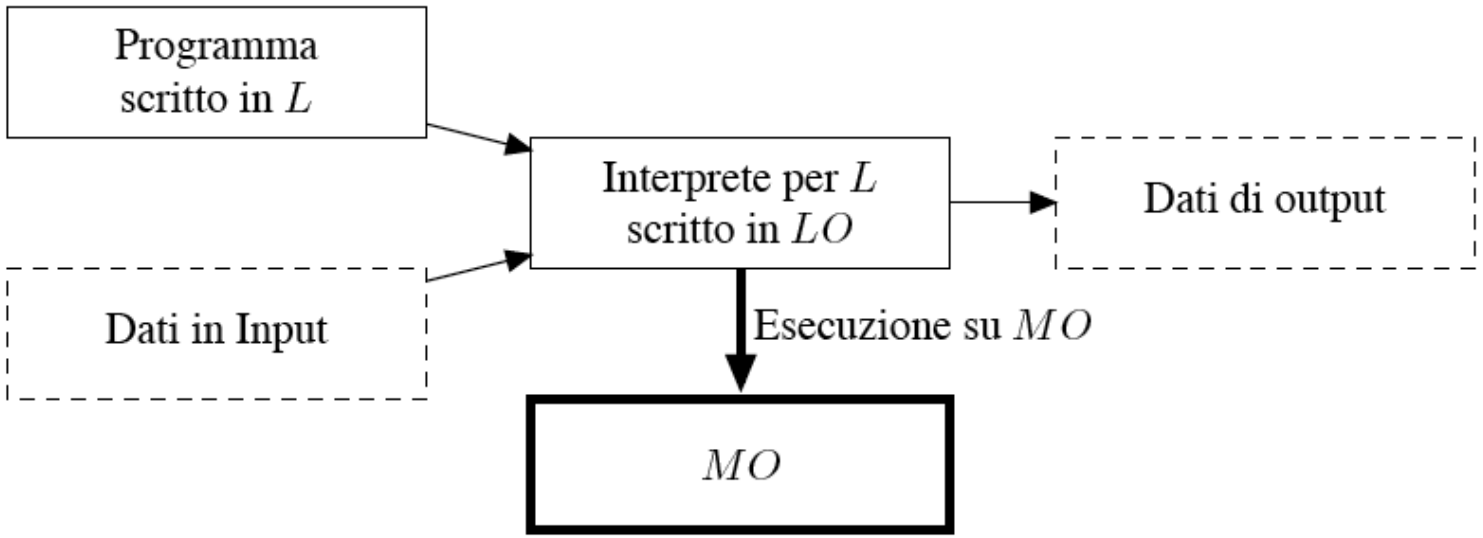
- Nella realtà, la MA viene realizzata su di una MO fisica mediante una combinazione delle tre tecniche
  1) N*on necessariamente ogni livello maschera completamente i livelli sottostanti*

Implementare un linguaggio

- Scartiamo la soluzione hw
- Assimiliamo software e firmware
- Sono dati:
  - un linguaggio L da implementare
    - cioè di cui realizzare la macchina astratta M_L
  - una macchina astratta Mo_{L0} (macchina *ospite*)
    - col suo linguaggio $L0$
- Si vuole
  - implementare L su Mo_{L0}
- Due modi radicalmente diversi…

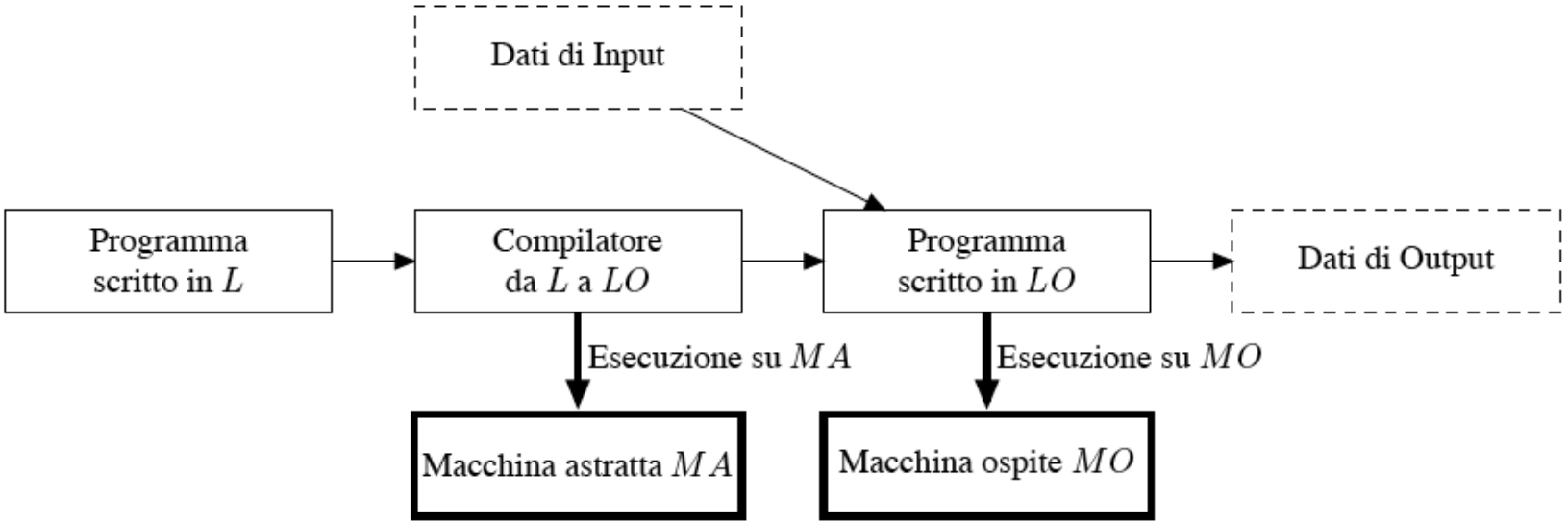
Implementazione interpretativa pura

- M è realizzata scrivendo un *interprete* per L su Mo :



Implementazione compilativa pura

- I programmi in L sono *tradotti* in programmi *equivalenti* in LO
- Traduzione effettuata da un (altro) programma detto compilatore



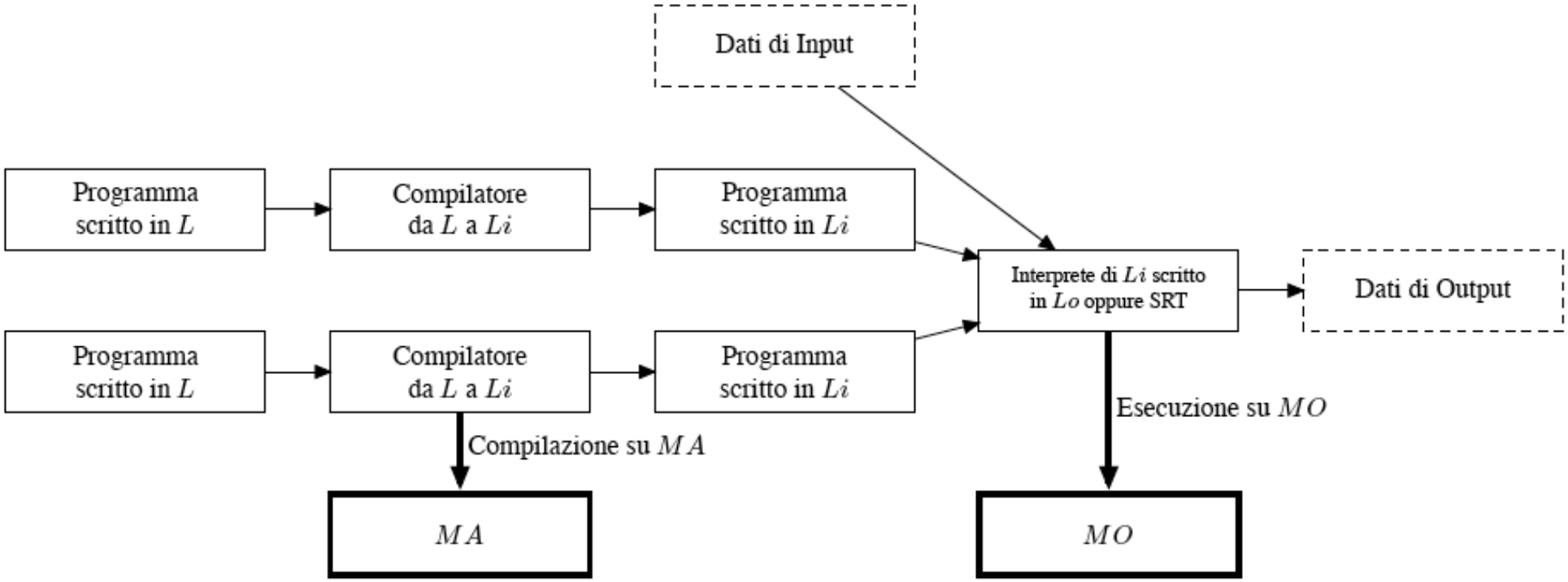
Compilazione o interpretazione ?

- Implementazione interpretativa pura:
  - scarsa efficienza della macchina $\mathbf{M}_L$
  - buona flessibilità e portabilità
  - facilità di interazione a run-time (es. debugging)

- Implementazione compilativa pura:
  - difficile, data la lontananza fra L e L_0
  - buona efficienza (decodifica a carico del compilatore e ottimizzazioni)
  - scarsa flessibilità
  - perdita di informazione sulla struttura del programma sorgente

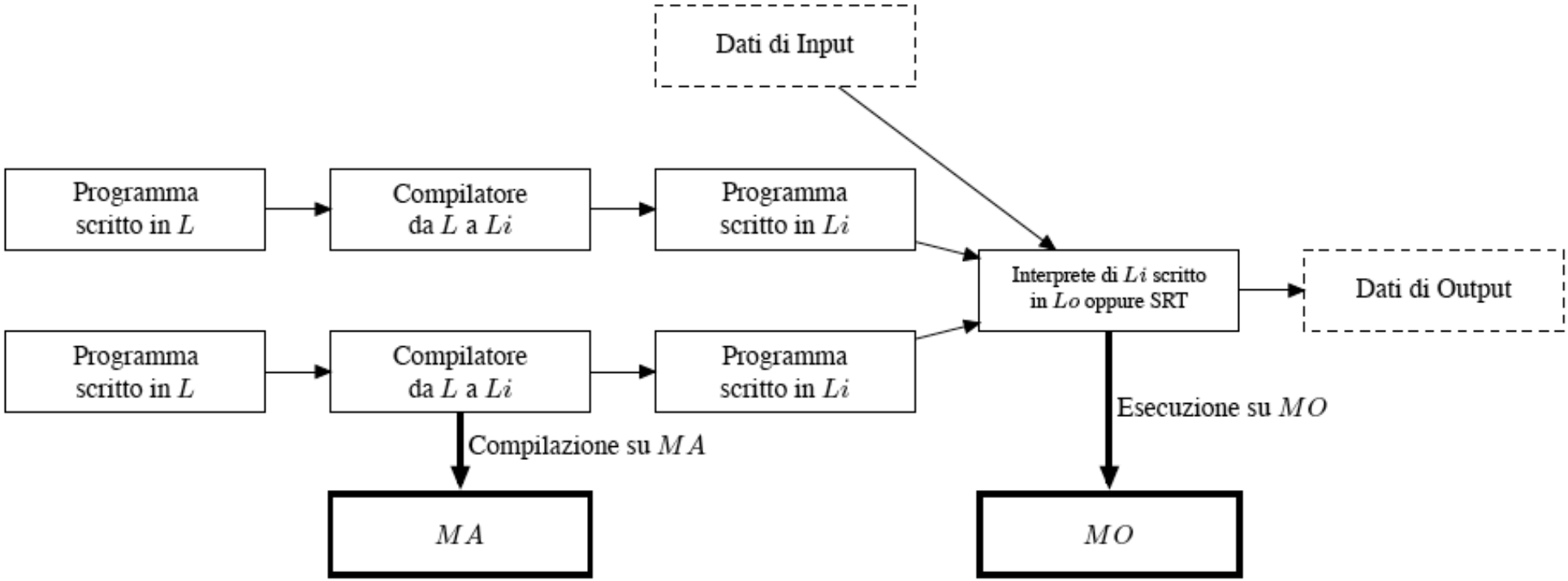
Nel caso reale

- **Entrambe le componenti coesistono**

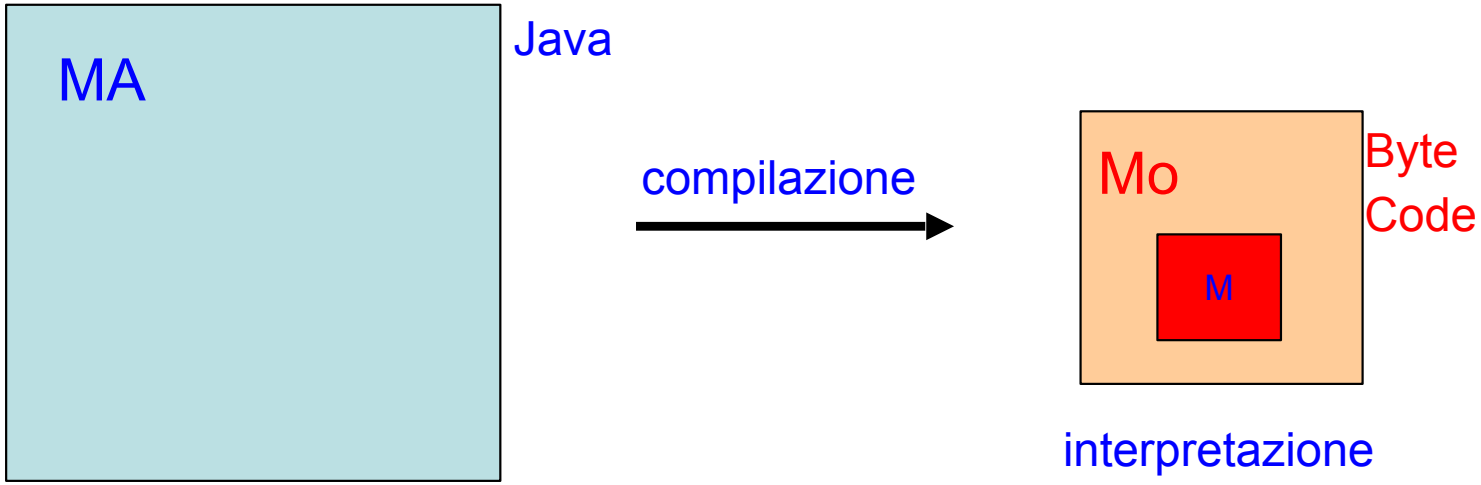


Implementazione di tipo interpretativo

- L'interprete della macchina intermedia M_{Li} è sostanzialmente diverso dall'interprete della macchina ospite M_L



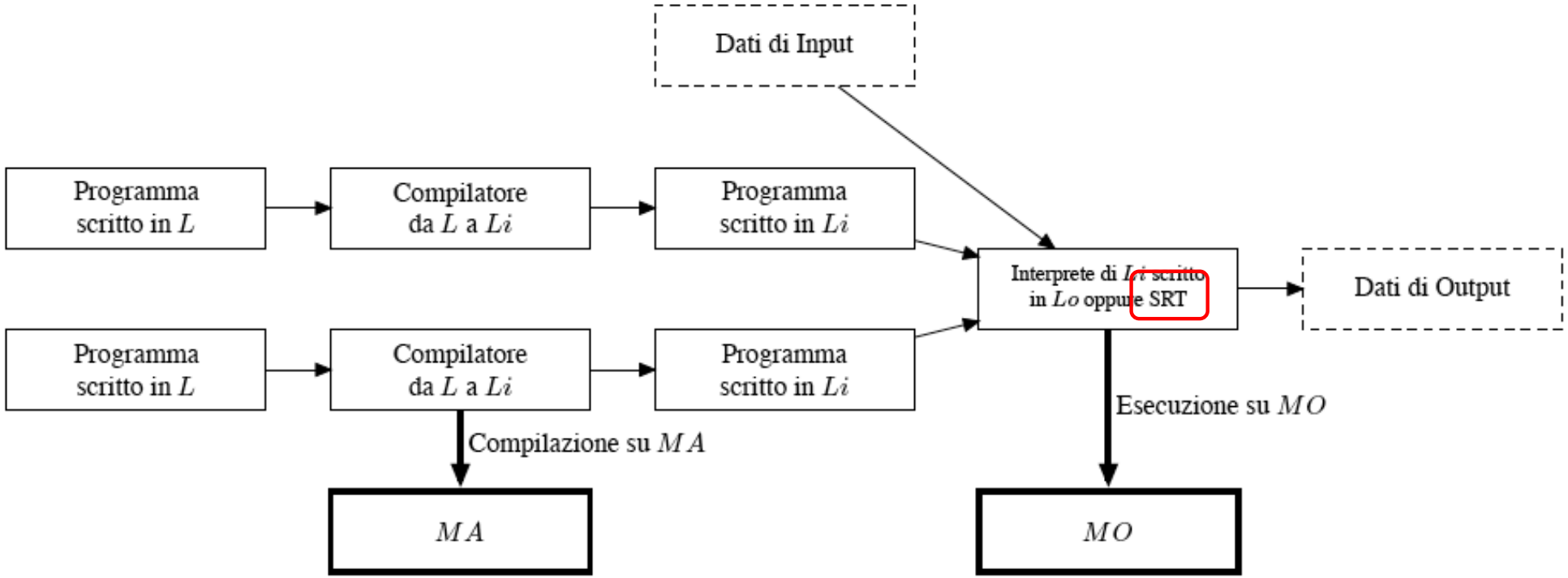
Esempio: Java



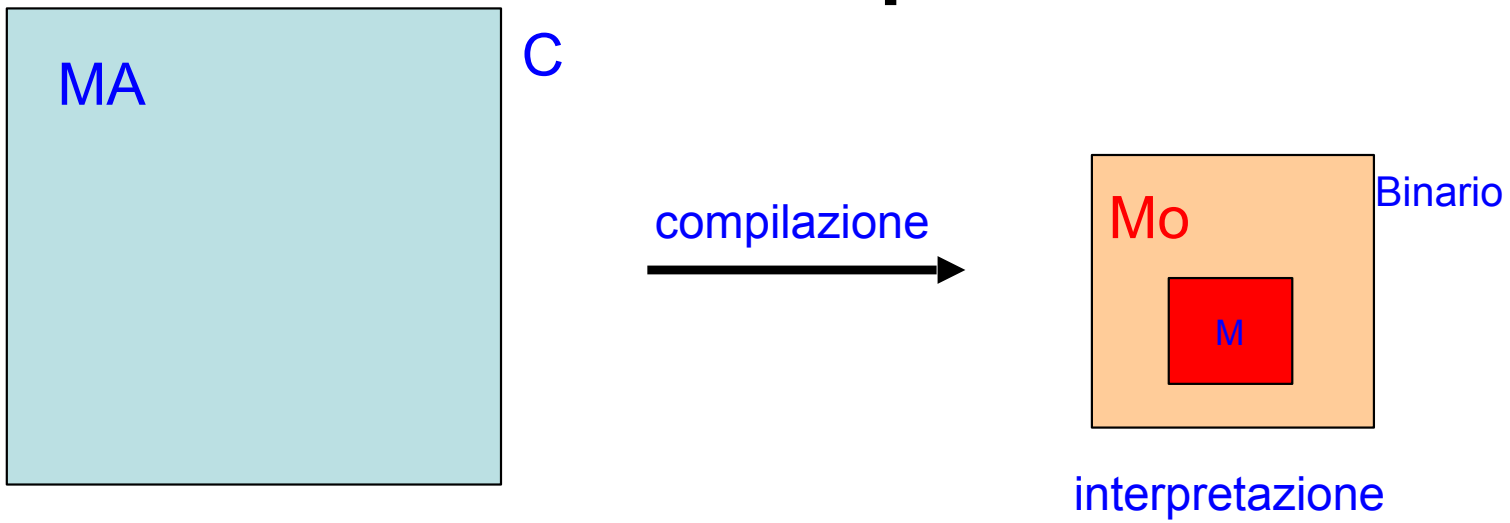
L_{MA} = Java
L_{Mo} = Bytecode
Mo = JVM
M = opportuna macchina che esegue l'interprete della JVM

Implementazione di tipo compilativo

- Interprete della macchina intermedia M_{Li} = interprete della macchina ospite M_{l0} + opportuni meccanismi (p.e. I/O, gestione



Esempio: C



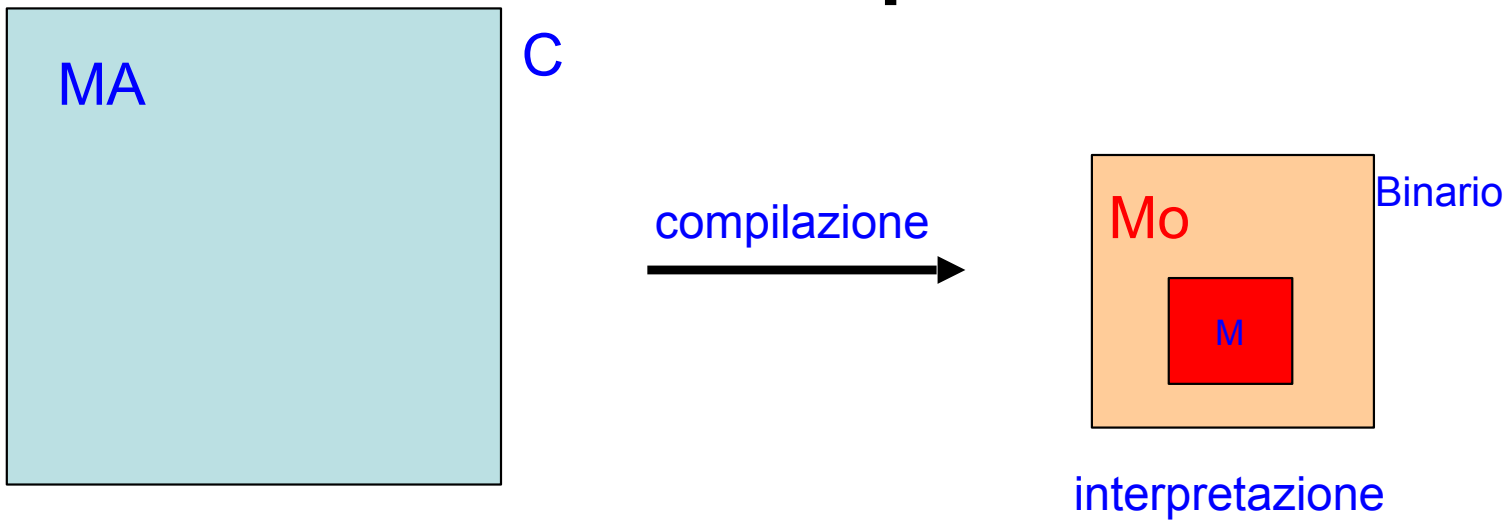
L_{MA} = C

L_{mo} = codice generato da `cc`

Mo = ?

M sembra inutile: coincide con Mo — NO

Esempio: C



L_{MA} = C

L_{mo} = linguaggio generato da cc +
supporto per gestione memoria, I/O ecc.

Mo = M + interprete per le chiamate del supporto a run-time

M = macchina ospite

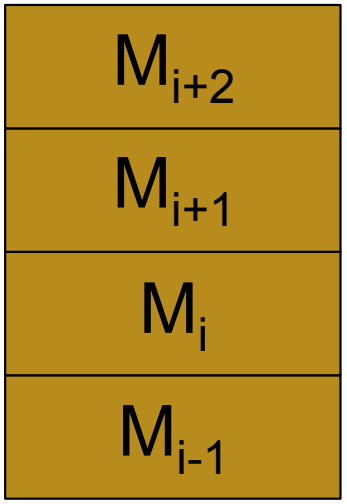
supporto a run-time

Linguaggi reali

- Linguaggi tipicamente implementati in modo compilativo
  - **C, C++, FORTRAN, Pascal, ADA**
- Linguaggi tipicamente implementati in modo interpretativo
  - **LISP, ML, Perl, Postscript, Pascal, Prolog, Smalltalk, Java**

gerarchia di macchine astratte

un’architettura informatica (*hw* o *sw*) si struttura in una serie di macchine astratte gerarchiche



M_i:
- usa i servizi forniti da M_{i-1} (il linguaggio L_{Mi-i})
- per fornire servizi a M_{i+1}(interpretare L_{Mi+i})
- nasconde (entro certi limiti) la macchina M_{i-1}

Stando al livello i può non essere noto (e in genere non serve sapere...) quale sia il livello 0 (hw)

Una gerarchia tipica: web application