Calcolatori Elettronici A
a.a. 2008/2009

INTRODUZIONE AL CORSO

Massimiliano Giacomin

Materiale di studio

Sito Internet del Corso: **http://zeus.ing.unibs.it/calca/**

- Lucidi del corso
- Link, informazioni varie, eventuale software da scaricare
- Regole, news, risultati degli esami, ecc.
- Ogni altra cosa dovesse risultare utile!

Libro

- Patterson & Hennessy:

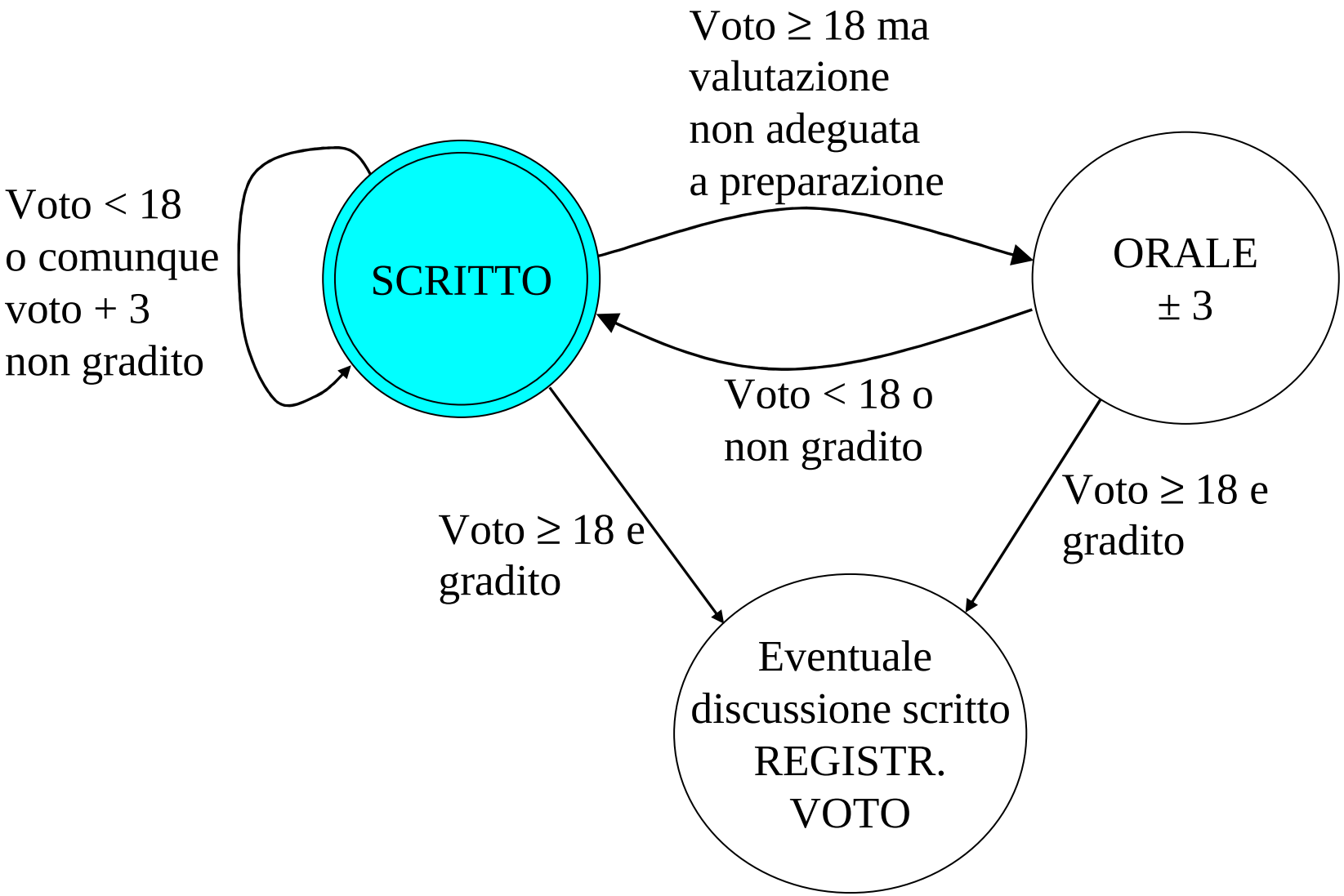
  Computer Organization and Design [Third Edition]

  [Morgan Kaufmann - Elsevier]

  anche in italiano: *"Struttura e progetto dei calcolatori"*
  – seconda edizione Zanichelli condotta sulla terza edizione americana

Lo stesso che sarà usato in calcolatori B

REGOLE PER L'ESAME



TIPOLOGIA DI ESAME SCRITTO

- Lungo e quanto più possibile ampio nei contenuti
- Teoria + esercizi
- Diversi quesiti (domande a risposta aperta, a crocette, esercizi, ecc.); ciascuno ha una valutazione compresa tra 0 e max punteggio indicato, somma compresa tra 30 e 32 (a seconda dei casi)
  NB: non ci sono valutazioni negative degli esercizi (voto min=0)
- Tempo di svolgimento sovrabbondante (di solito 2 ore e mezza o 3 ore + ev. recupero)
- Per partecipare è necessario iscriversi (se mancano posti vengono esclusi tutti coloro che non si sono iscritti)!

TIPOLOGIA DI ORALE

- Breve e “di aggiustamento”
- Può essere svolto in qualunque data (anche fuori sessione) previo accordo con il docente
- Può includere una discussione dello scritto, ma non si limita a questa
- Come riportato nel lucido precedente, l’incremento è un numero relativo (quindi può anche essere negativo)
- Quindi: fatelo se vi sentite preparati!

VALIDITA’ DEI VOTI

- Un voto sufficiente che non perda di validità a seguito di orale o scritto successivo (vedi poi) può essere registrato in qualunque data, anche dopo anni. Si può registrare durante un qualsiasi scritto, durante la visione/registrazione di un compito di qualsiasi appello o, in caso di necessità, anche in altra data (purché in un periodo di sessione di esami, consultare il calendario accademico) previo avviso via mail
- Un voto conseguito allo scritto è sufficiente se maggiore o uguale a 18
  Esempio: 17 non è un voto sufficiente
- Si può sostenere l’orale solo con un voto sufficiente ancora valido conseguito allo scritto
- Se a seguito di un orale il voto risulta inferiore a 18, esso perde di validità ed è quindi necessario rifare lo scritto
- Il voto che risulta sufficiente (maggiore o uguale a 18) dopo un orale non può più essere modificato. Naturalmente, si può sostenere un nuovo scritto, con il rischio però di perdere il voto (vedi il punto successivo)
- Chi consegna uno scritto (senza ritirarsi) perde l’eventuale voto positivo conseguito precedentemente (con scritto o con orale)
- Chi non consegna lo scritto (si ritira) mantiene l’eventuale voto sufficiente valido

Contattarmi

Ufficio n. 27 del DEA
E-mail: giacomin@ing.unibs.it

Orario di ricevimento:

GIOVEDI' DALLE 11.30 alle 13.00

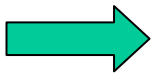
[in caso di problemi: altre giornate su appuntamento]

- Garantito in periodo di lezione [eventi speciali - p.es. sedute di laurea - a parte] e di norma anche dopo, ma si consiglia appuntamento via mail per sicurezza
- Per particolari esigenze (es. NO per chiedere quando escono i risultati!): si consiglia di usare l'e-mail
- Eventuali **variazioni di orario/giorno** indicate nel **sito internet** del corso.

INTRODUZIONE AL PROGRAMMA DEL CORSO

Il concetto di calcolatore

- Macchina che risolve problemi che ammettono un algoritmo risolvente
  - variabili di ingresso: assumendo valori nel loro dominio (dati del problema) generano tutte le possibili istanze del problema
  - variabili di uscita: assumono valori, detti risultati del problema
  - algoritmo: metodo generale che specifica, mediante una sequenza di istruzioni, come produrre per ogni dato in ingresso il risultato che costituisce la soluzione dell'istanza del problema
- Il calcolatore è un esecutore universale di algoritmi
- Per essere comprensibile da un calcolatore reale, un algoritmo deve essere espresso mediante un linguaggio di programmazione: l'insieme delle istruzioni espresse viene chiamato programma



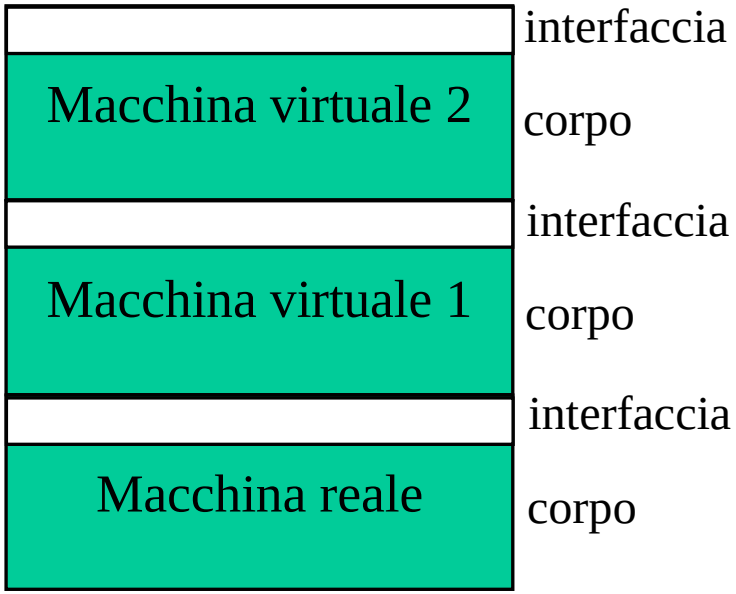
- Il calcolatore reale è in grado di eseguire qualunque programma
- Per definizione, il programma non è "cablato" nel calcolatore (concetto di programma memorizzato)

Oggetto del corso: il calcolatore elettronico

- Un calcolatore digitale può essere realizzato usando diverse tecnologie: noi ci occupiamo di calcolatori elettronici (calcolatori realizzati mediante circuiti elettronici)
- I segnali elettrici vengono utilizzati per rappresentare cifre binarie (0-1, on-off)
- Altre tecnologie:
  - magnetiche (stato di polarizzazione)
  - ottiche
- Ma questo è solo un aspetto (livello) del calcolatore…

Livelli di astrazione e macchine virtuali

- Nella progettazione di un sistema, per dominarne la complessità lo si scompone in diversi livelli gerarchici:
  - ogni livello inferiore nasconde l'implementazione al livello superiore (information hiding – incapsulamento)
  - l'unica cosa che il livello superiore deve conoscere del livello inferiore è la sua interfaccia

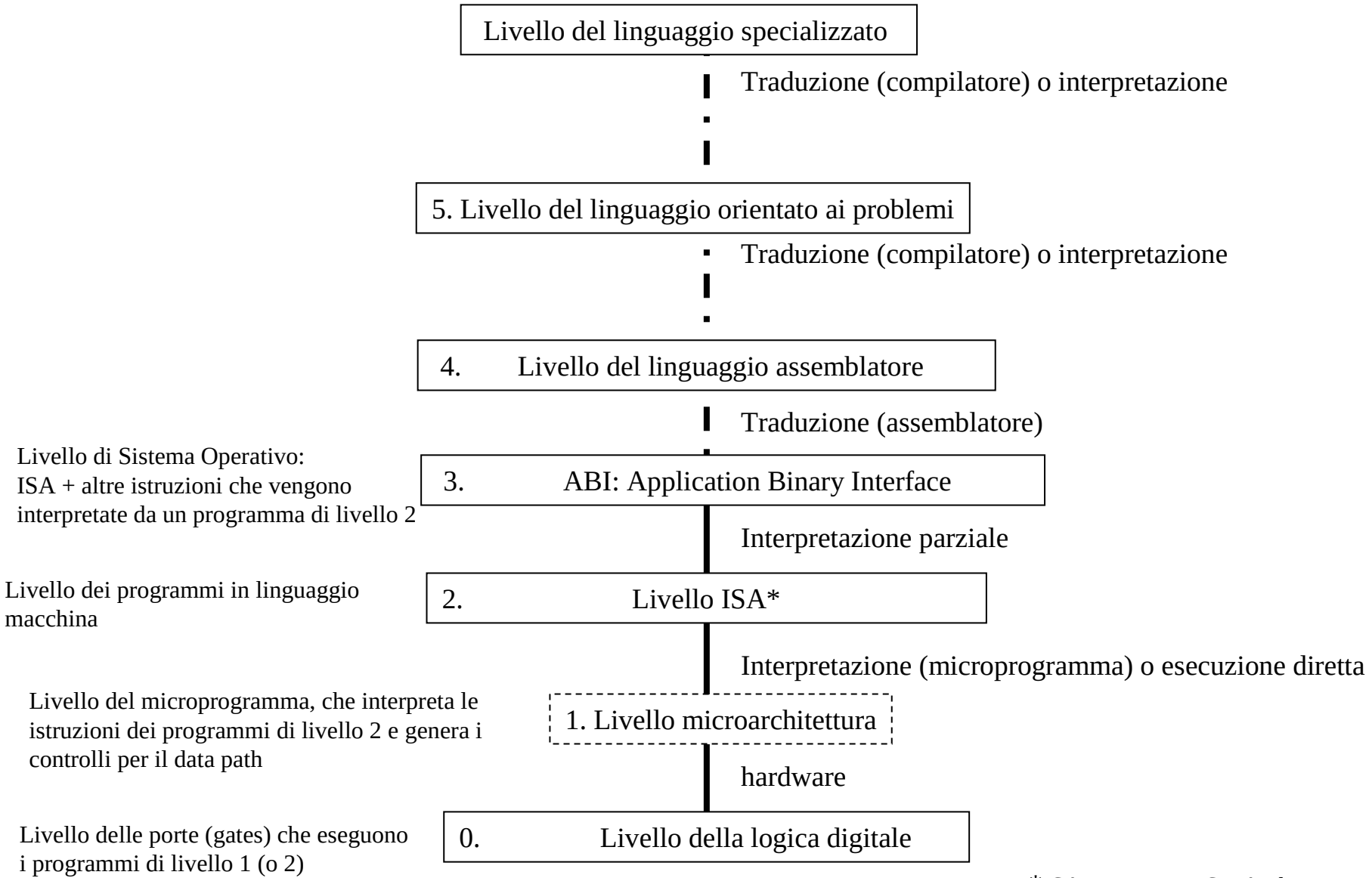


Interfaccia:
attraverso di essa gli utenti della macchina virtuale interagiscono con la macchina stessa, utilizzando un insieme di comandi

Corpo:
non visibile all'esterno, esegue i comandi ricevuti utilizzando a sua volta i servizi dell'eventuale macchina sottostante

Architetture

descrivono il calcolatore a diversi livelli di astrazione



*ISA: Instruction Set Architecture

• Il calcolatore è progettato come una sequenza di macchine a diversi livelli, ognuna costruita sulle macchine definite ai livelli sottostanti.

• Ogni livello rappresenta una distinta astrazione, caratterizzata da differenti oggetti e operazioni.

• I tipi di dati, le operazioni e le caratteristiche di ogni livello sono chiamate **architettura.**

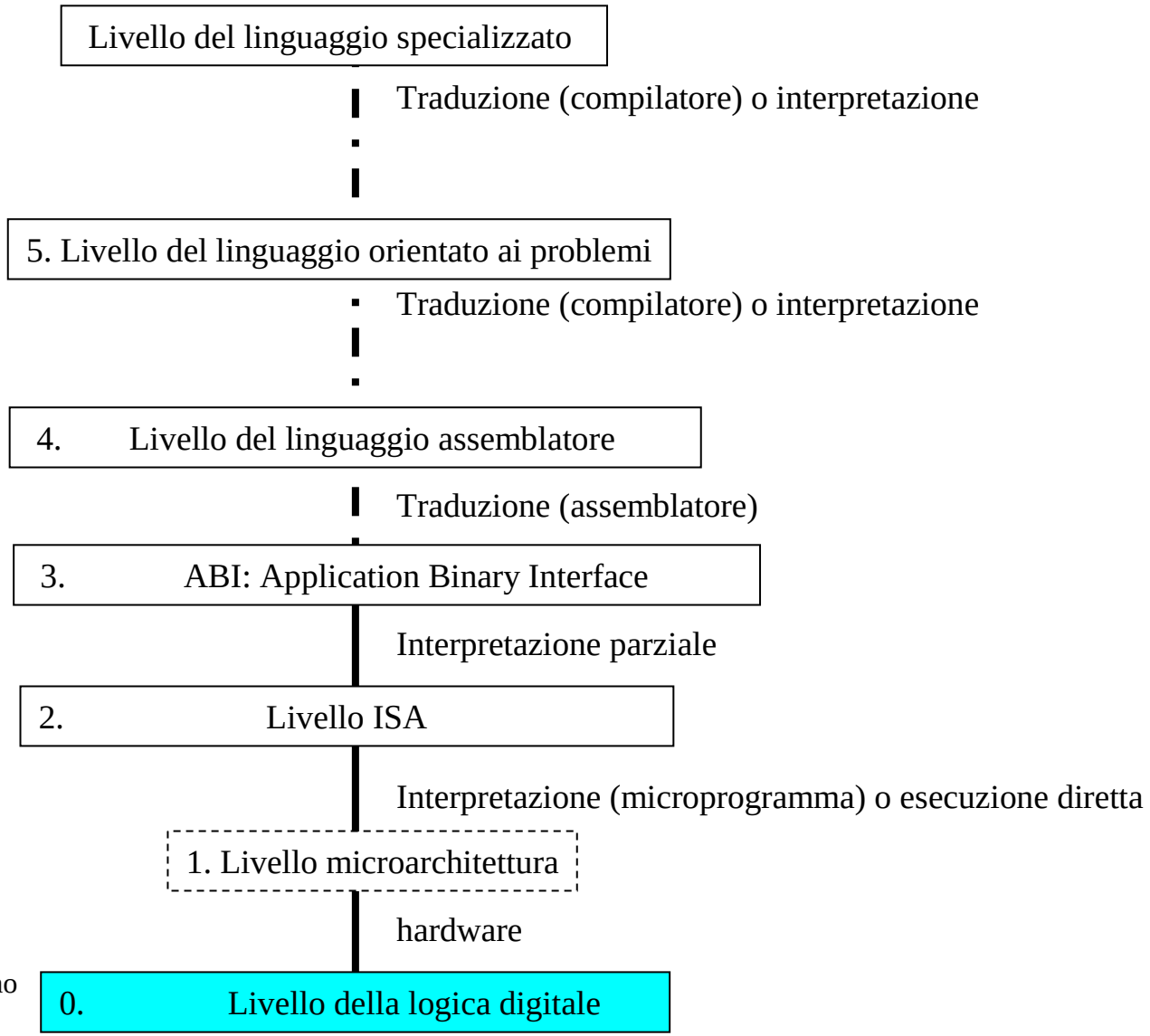
- Dal livello 5 in su, i linguaggi in cui esprimere i programmi sono chiamati linguaggi di alto livello: forniscono dati e operazioni per descrivere soluzioni di problemi in termini comprensibili per persone esperte in un certo campo.

- I programmi del livello 4 sono in forma simbolica;
  vengono generalmente tradotti (o a volte interpretati) da altri programmi.

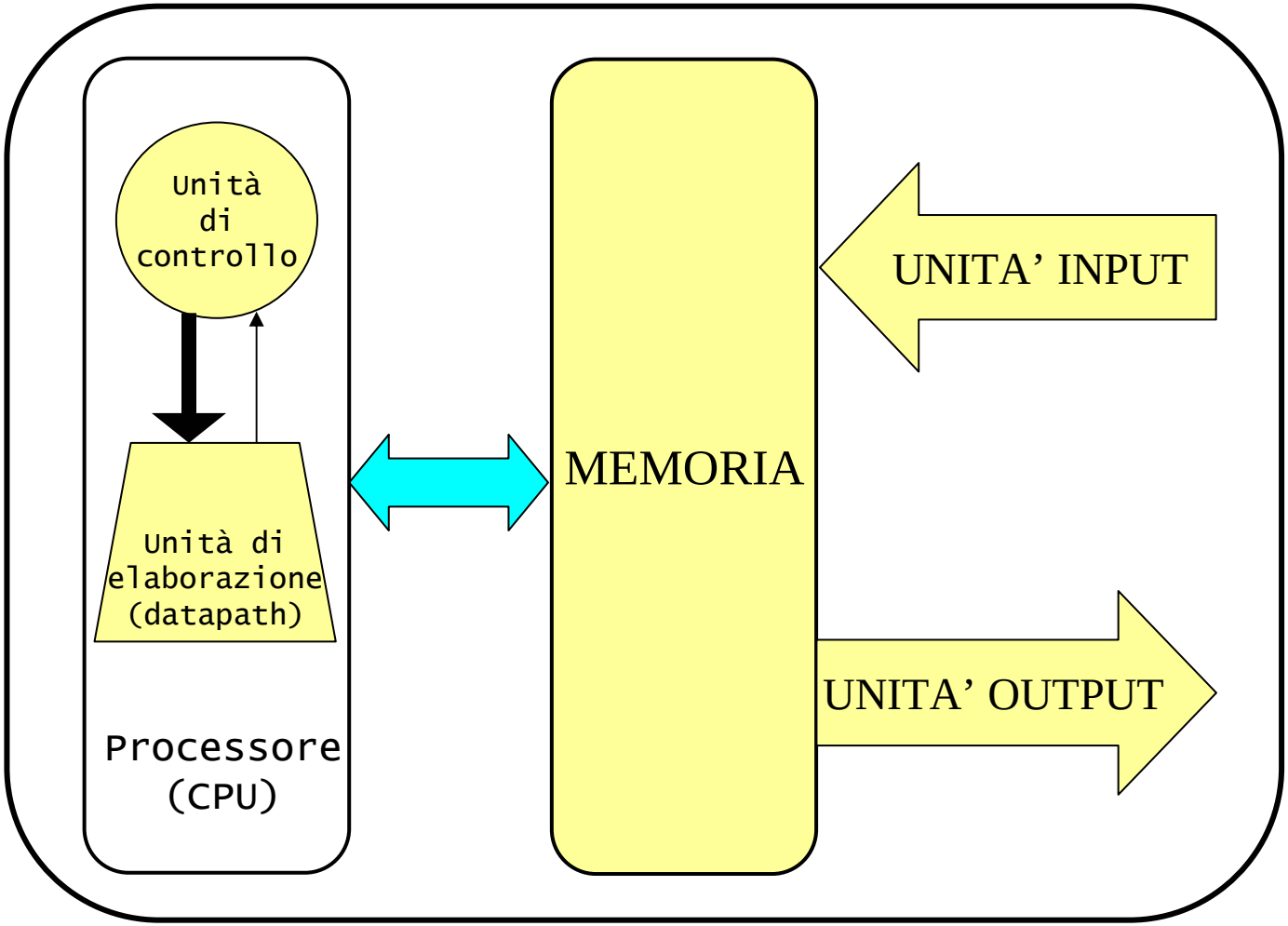
- I programmi ai livelli 1, 2 e 3 sono sempre interpretati;
  la forma interpretata dalla macchina è sempre numerica.

Architetture

descrivono il calcolatore a diversi livelli di astrazione



In realtà, anche il livello 0 si potrebbe decomporre in sottolivelli: noi ci occupiamo di hardware, ma ad un livello più astratto rispetto a quello della “logica digitale”



• I/O

• Memoria: contiene sia il programma sia i dati, trattati “allo stesso modo”
(concetto di programma memorizzato: cambiando il programma
la macchina appare diversa, ma la struttura fisica non cambia)

NB: a questo livello le istruzioni sono rappresentate come sequenze di bit
(linguaggio macchina)

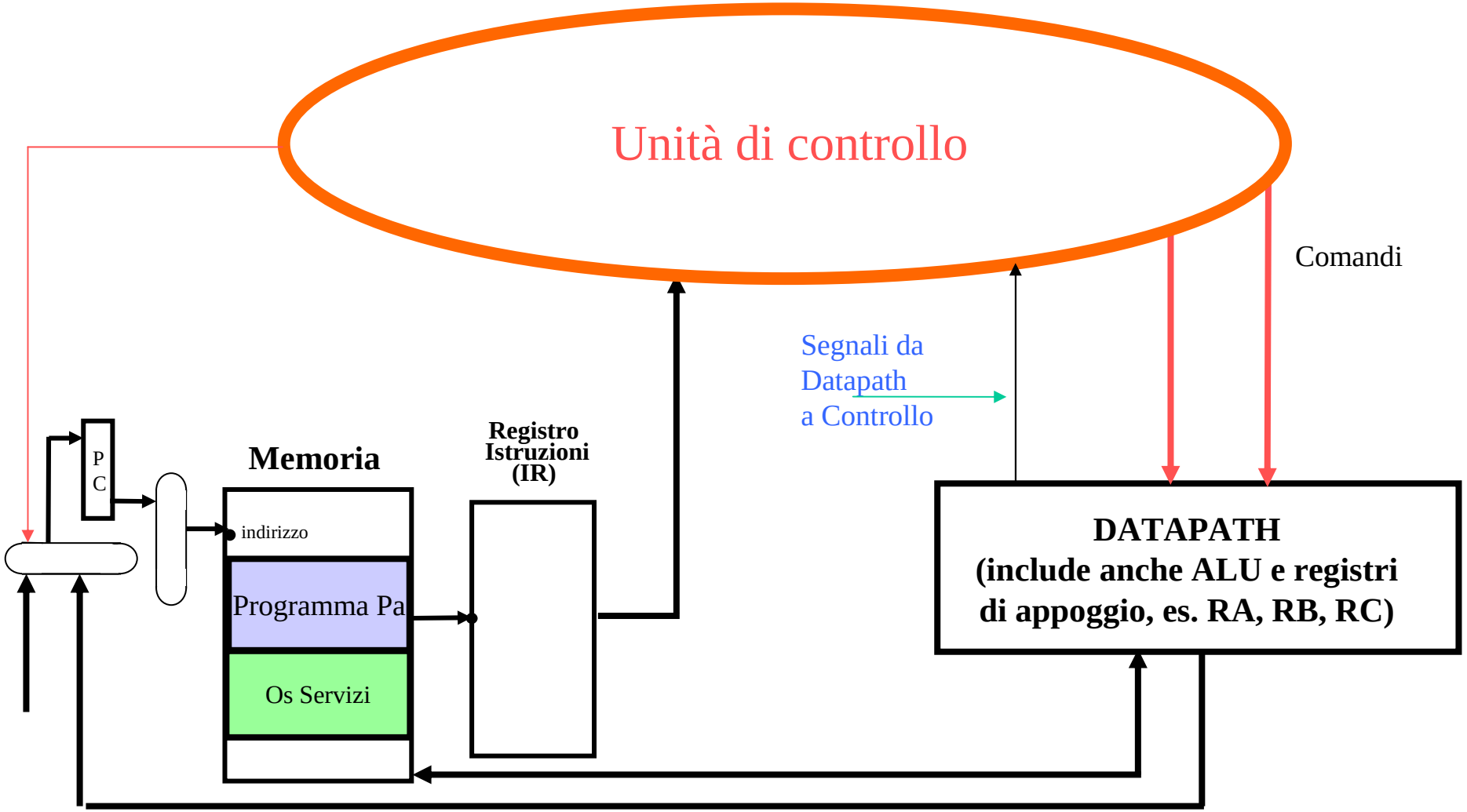
• **Processore** (o Unità centrale, **CPU**): esegue le istruzioni in memoria iterando il ciclo macchina (FETCH – DECODE – EXECUTE).

Include due componenti principali:

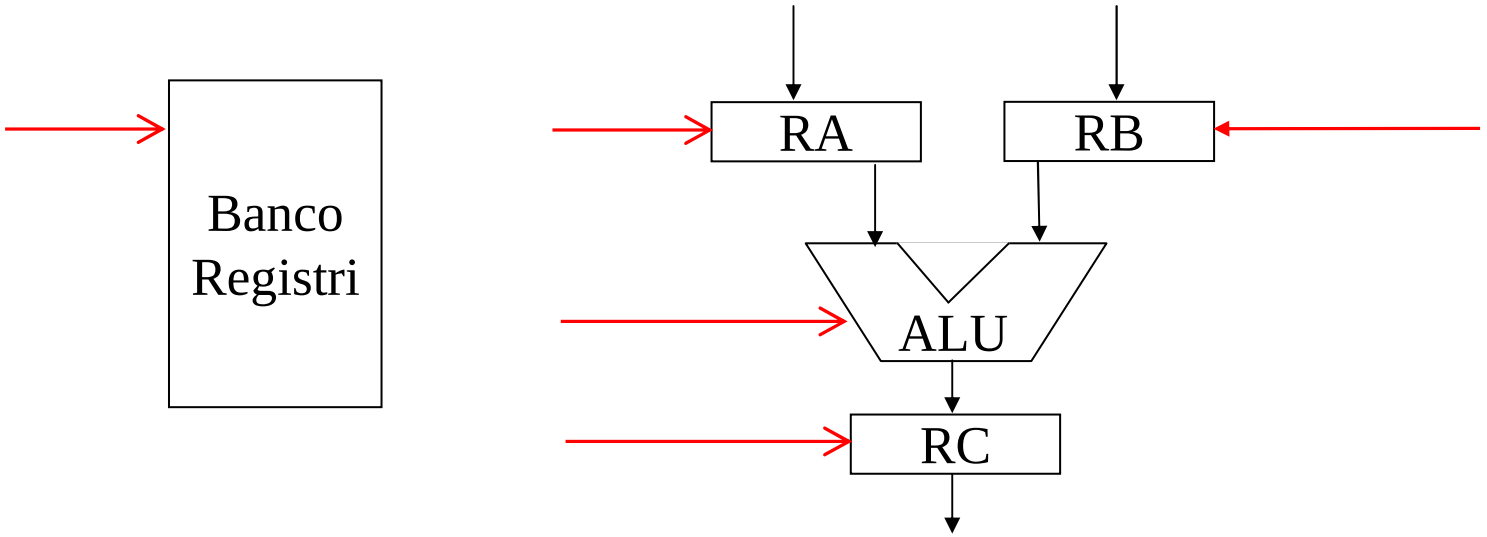
- **datapath**: insieme di elementi logici di computazione e di registri
che permettono di effettuare le operazioni aritmetiche e logiche
- **unità di controllo**: contiene la logica per inviare i segnali al datapath,
alla memoria e ai dispositivi I/O in modo da eseguire
le operazioni prescritte dall’istruzione del programma

Vediamo più in dettaglio lo schema processore-memoria

Durante l'esecuzione di un programma applicativo Pa, i circuiti interpretano le istruzioni del programma in linguaggio macchina costituito dal < Pa (tradotto) ∘ i servizi OS>

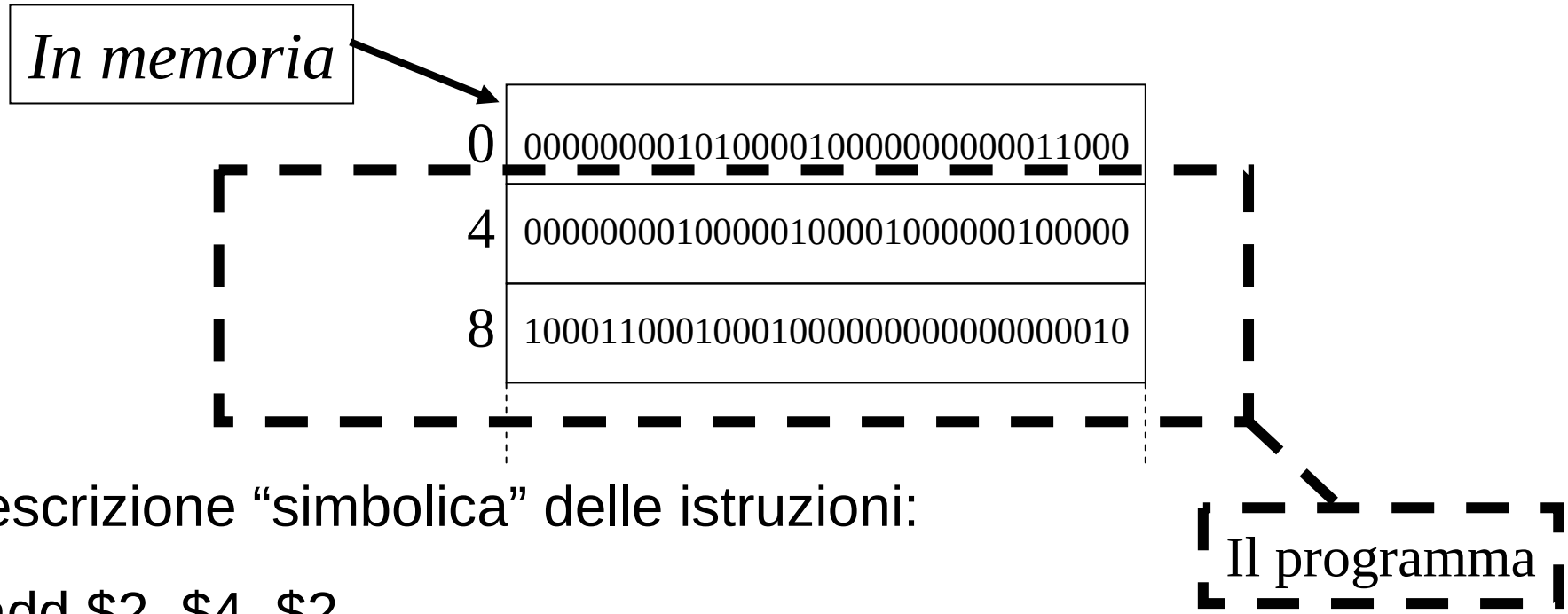


NB: nel datapath potrebbe esserci un’organizzazione del tipo



NB: Controlli unità di controllo indicati in rosso

Un Esempio di due istruzioni

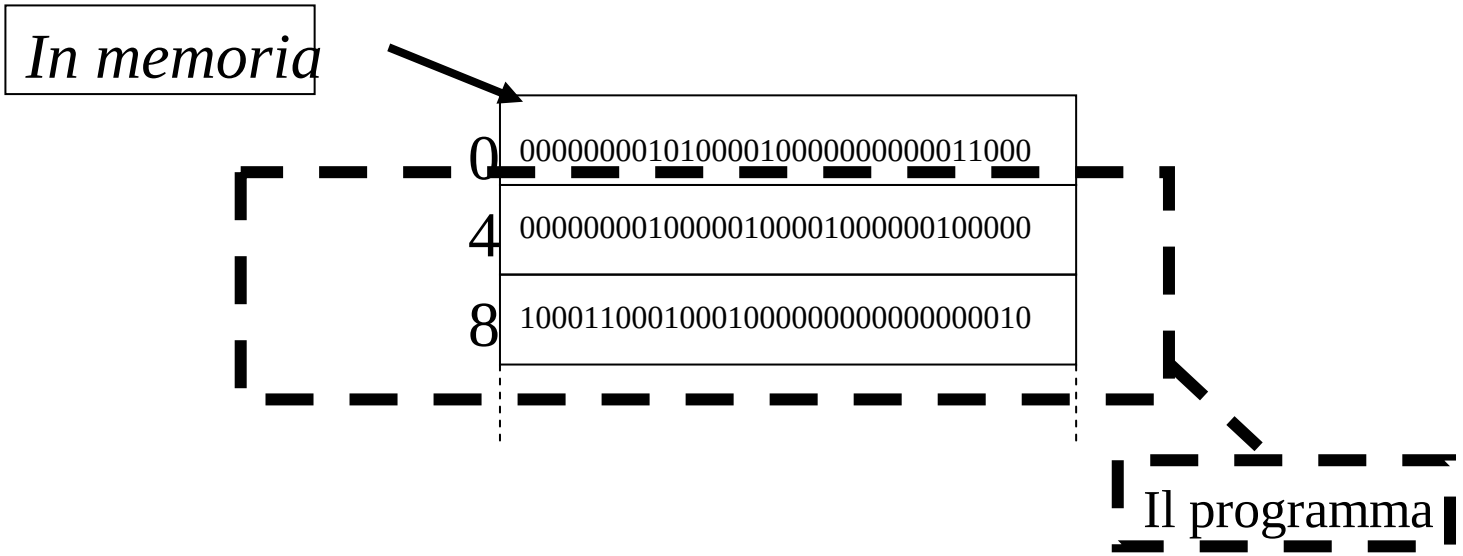


Descrizione “simbolica” delle istruzioni:

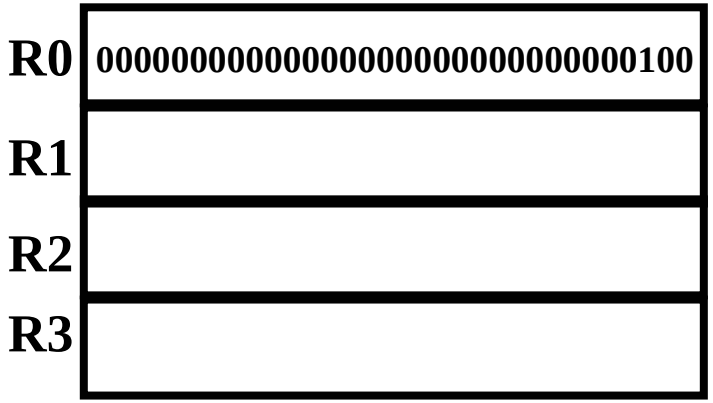
- add $2, $4, $2
- lw $4, 2($2)

Somma il contenuto del registro R2 al contenuto di R4 e metti il risultato in R2.
Poi carica nel registro R_4 il valore della parola presente all’indirizzo di memoria ottenuto sommando 2 al contenuto in R_2

Un Esempio di due istruzioni

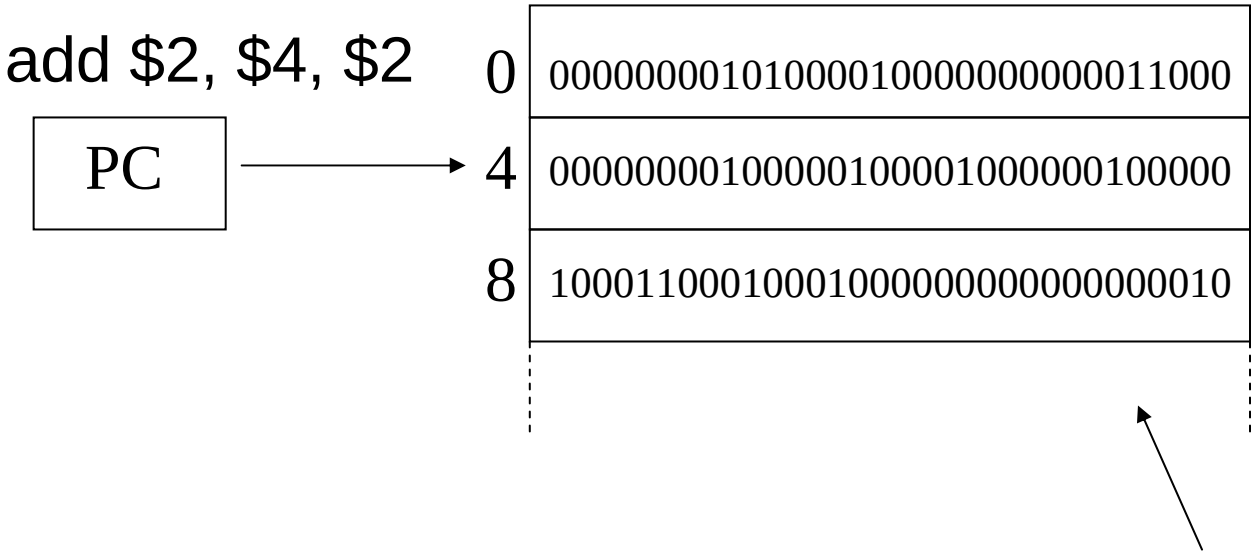


Ma anche i registri contengono dati



Esecuzione di una istruzione aritmetica

Somma il contenuto del registro R_4 al contenuto del registro R_2 e memorizza il risultato in R_2



Programma in memoria
in linguaggio macchina

Passi per eseguire l'addizione

- Passo 1: Carica istruzione in IR e aggiorna PC

  IR ← 00000000100000100001000000100000

  (si indica di solito con IR ← (PC))

  R_A ← [PC] R_B ← 4

- Passo 2: DEC decodifica istruzione in IR

  000000 00100 00010 00010 00000 100000

  add $4 $2 $2

  ALU calcola [PC] +4 e lo memorizza in R_C

Passi per eseguire l'addizione

- Passo 3: Caricamento valori dei registri

  $R_A \leftarrow R_2$

  $R_B \leftarrow R_4$

  Si aggiorna il PC $\leftarrow R_C$

- Passo 4: Somma (operazione con ALU)

  $R_C \leftarrow R_A + R_B$

- Passo 5: Memorizza risultato in R2

  $R_2 \leftarrow R_C$

  TOTALE = 5 passi (eseguiti in 5 cicli di clock)

Segnale di clock

- Un segnale di clock caratterizzato da una frequenza costante sincronizza i vari eventi all'interno dell'hardware
- Determina in questo modo intervalli di tempo discreti che sono denominati cicli di clock
- Si fa riferimento alla durata di un periodo di clock oppure alla frequenza di clock

Istruzione di lettura da memoria

- Il PC è stato aggiornato è punta all'istruzione successiva alla add

PC 00000000000000000000000000001000

Passi per eseguire una load

- Passo 1: Carica istruzione in IR e aggiorna PC

  $IR \leftarrow (PC)$

  $R_A \leftarrow [PC]$ $\qquad\qquad$ $R_B \leftarrow 4$

- Passo 2: Decodifica istruzione in IR

  $\underbrace{100011}_{\text{lw}}\ \underbrace{00010}_{\$2}\ \underbrace{00100}_{\$4}\ \underbrace{0000000000000010}_{\text{offset= 2}}$

  ALU calcola [PC] +4 e lo memorizza in R_C

- Passo 3: Copia 2 e R_2 nei registri usati dalla ALU

  $R_A \leftarrow 2$

  $R_B \leftarrow R_2$

Si aggiorna il PC $\leftarrow R_C$

- Passo 4: Somma contenuto registri

  $R_C \leftarrow R_A + R_B$
- Passo 5: Copia contenuto di R_C in MAR
  (registro indirizzi per accedere alla memoria)

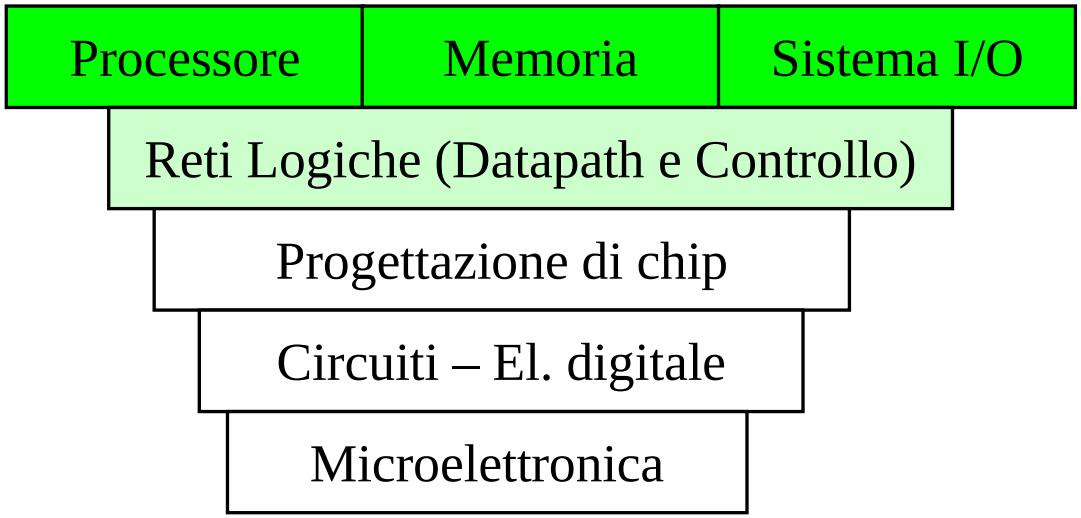
  $MAR \leftarrow R_C$
- Passo 6: Accedi alla memoria e carica dato

  $MDR \leftarrow (MAR)$
- Passo 7: Copia contenuto di MDR in R_4

  $R_4 \leftarrow MDR$

TOTALE = 7 passi (eseguiti in 7 cicli di clock – attenzione all'accesso in memoria!)

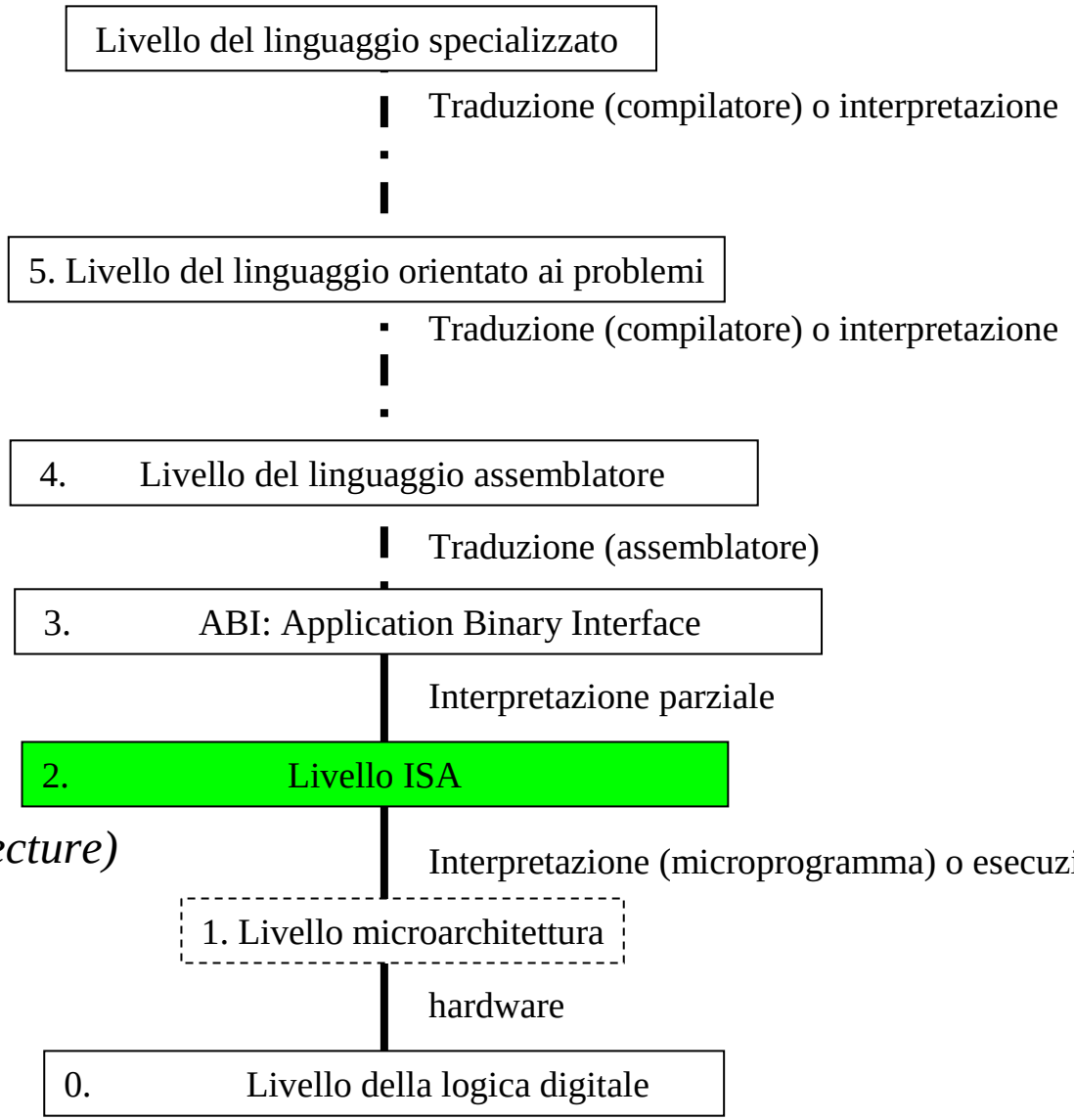
Di cosa ci occuperemo a questo livello?



RISULTATO: l'interfaccia con il livello superiore viene detta ISA (instruction set architecture del processore)

Architetture

descrivono il calcolatore a diversi livelli di astrazione



Livello dei programmi in linguaggio macchina: ISA (Instruction Set Architecture)

ISA (Instruction set architecture)

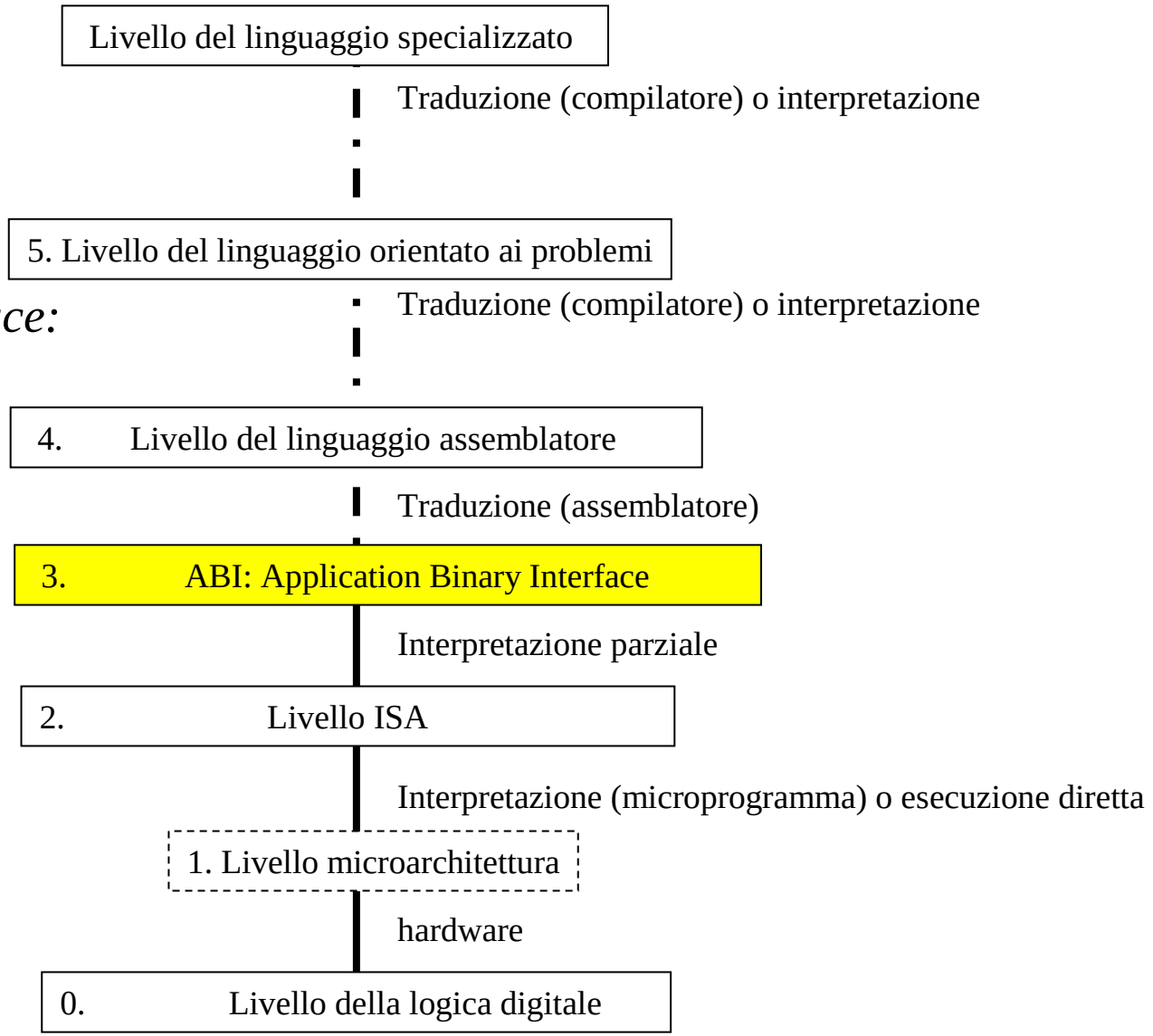
- Costituisce l'interfaccia tra l'hardware e il livello più basso del software
- Specifica tutto ciò che i programmatori devono sapere per produrre programmi in linguaggio macchina corretti:
  - istruzioni in linguaggio macchina (formato, semantica)
  - dispositivi I/O
  - …
- Una stessa architettura può avere implementazioni diverse!
- Alcune architetture attuali:
  - IA-32 (Intel, AMD)
  - MIPS
  - PowerPC
  - ARM
  - …

Architetture

descrivono il calcolatore a diversi livelli di astrazione

Sistema operativo gestisce:
- *I/O*
- *memoria (centrale e di massa)*
- *CPU (processi)*

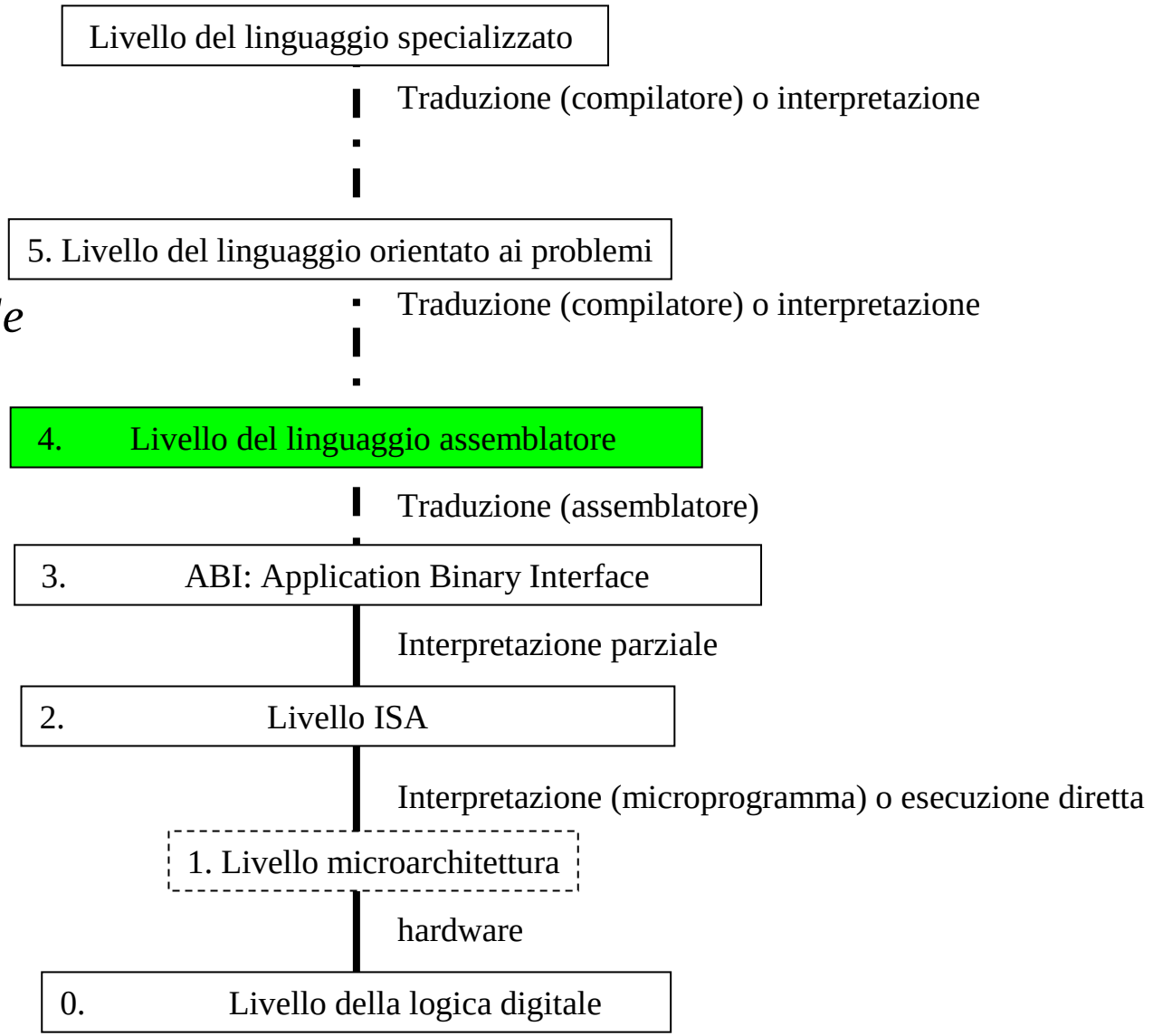
➡ ABI: combinazione istruzioni ISA + API



Architetture

descrivono il calcolatore a diversi livelli di astrazione

Forma "simbolica" delle istruzioni macchina, ma con
- *pseudoistruzioni*
- *etichette*
- *direttive*
- *commenti*



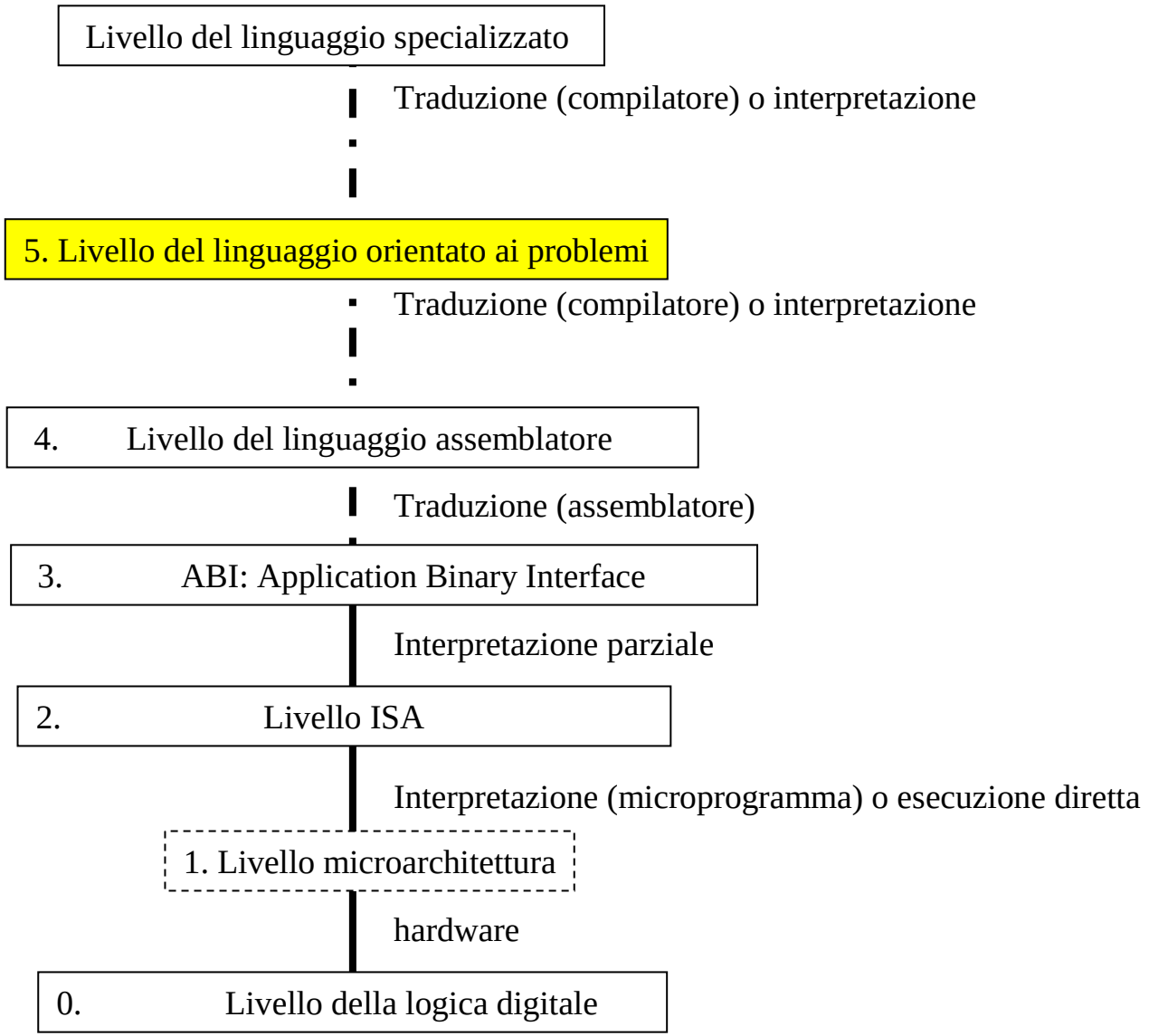
Linguaggio macchina vs. linguaggio assembler

- Il linguaggio macchina è un linguaggio binario, praticamente impossibile da gestire per il programmatore
- **Linguaggio assembly**: consente una rappresentazione simbolica delle istruzioni macchina
- La traduzione a linguaggio machina viene eseguita da un programma detto assemblatore
- Il linguaggio assembly è caratterizzato da:
  - codici mnemonici associati ai codici operativi e ai registri
  - etichette per rappresentare indirizzi di memoria
  - commenti
  - direttive
  - pseudoistruzioni

Architetture

descrivono il calcolatore a diversi livelli di astrazione

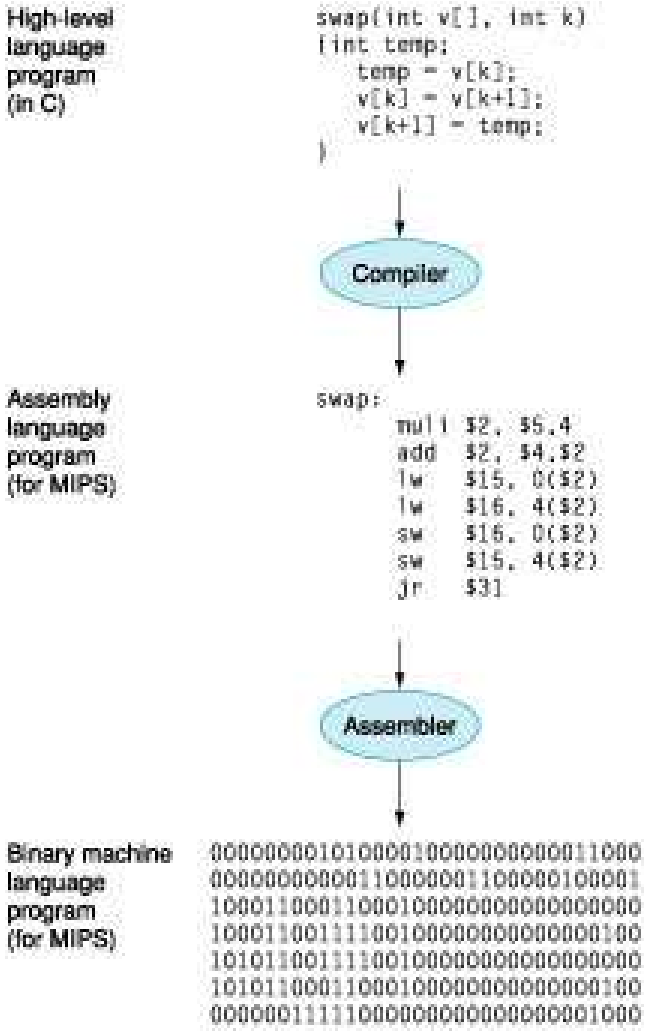
cfr. Fondamenti di Informatica B e C



Il linguaggio assembly ha svariati difetti, connessi con il basso livello di astrazione

⇨ Linguaggi di alto livello:

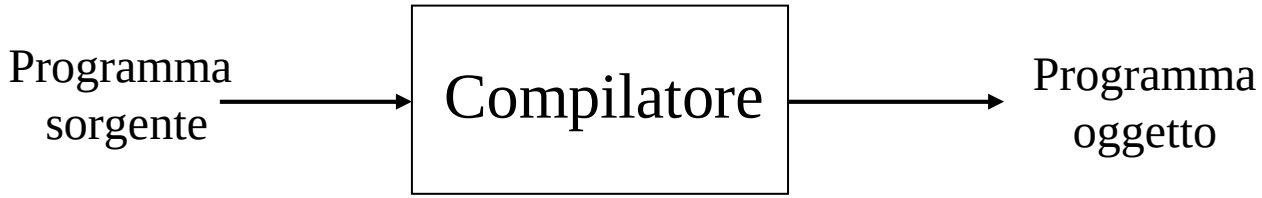
- aumento produttività
- indipendenza dalla macchina (portabilità)



Compilazione vs Interpretazione

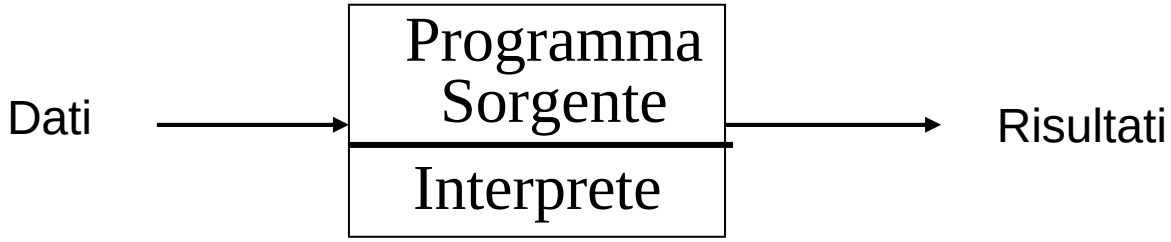
a) compilazione: per ottenere risultati 2 fasi

Fase 1: al tempo di compilazione



Fase 2: al tempo di esecuzione

b) interpretazione



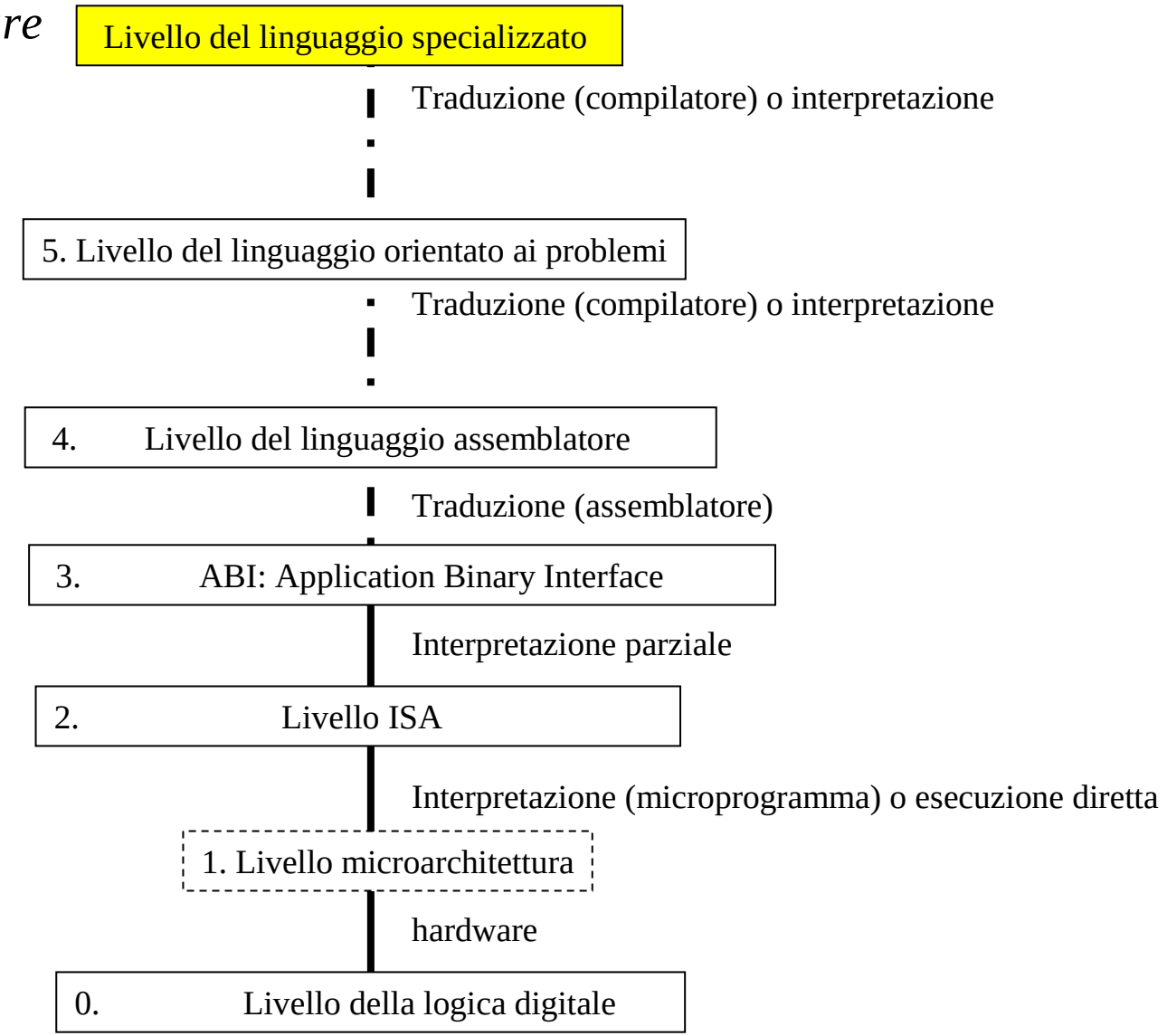
Il calcolatore è un interprete hardware!

Architetture

descrivono il calcolatore a diversi livelli di astrazione

La macchina come appare all'utente. Esempi:

- *word processor*
- *browser*
- *simulatore*



CLASSI DI CALCOLATORI

• I concetti visti sono validi per qualunque calcolatore, però classi di calcolatori diverse hanno diversi requisiti di progetto

1. Desktop computer
   - buona performance per il singolo utente
   - costo moderato
   - software realizzato da terze parti

CLASSI DI CALCOLATORI

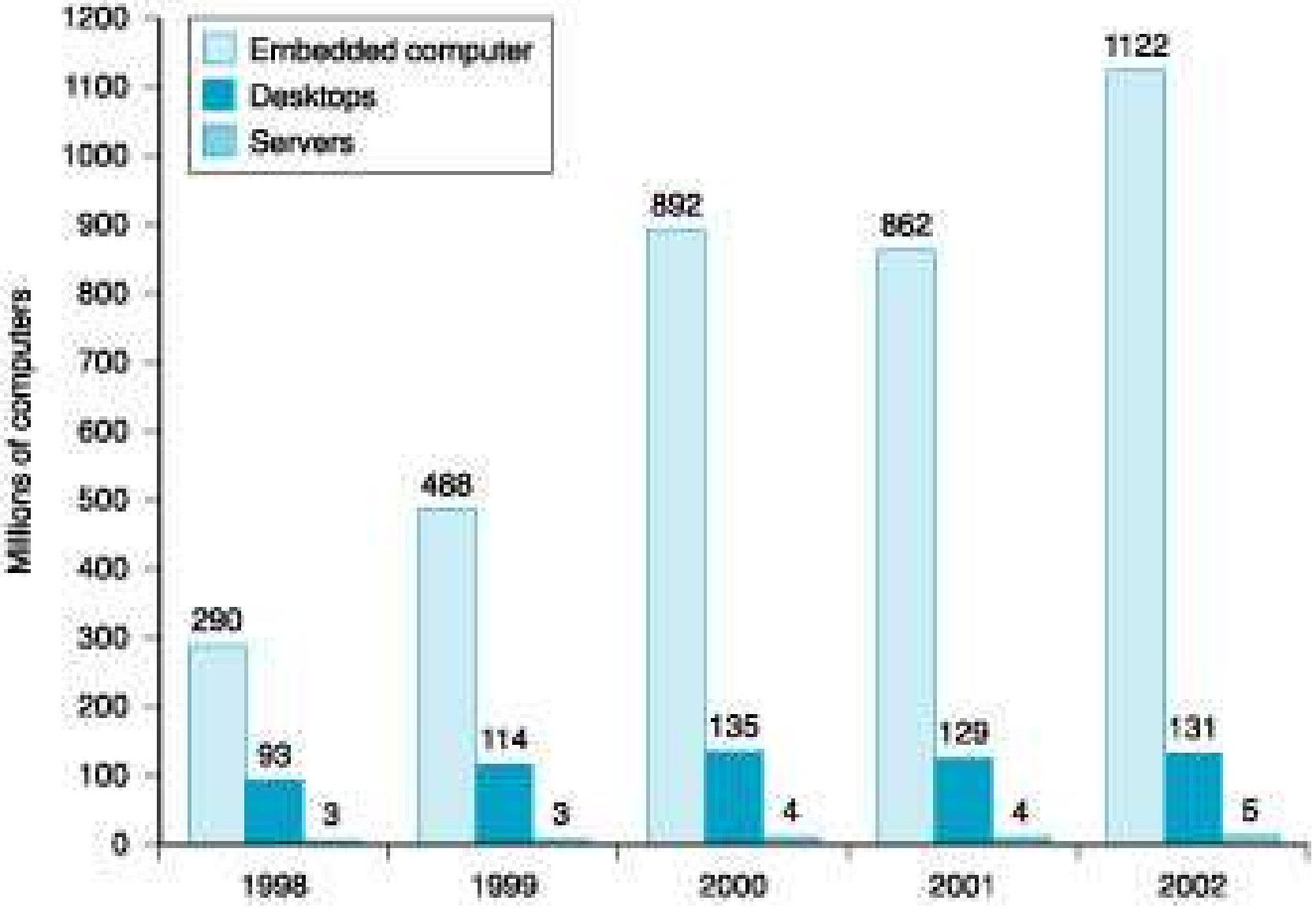
• I concetti visti sono validi per qualunque calcolatore, però classi di calcolatori diverse hanno diversi requisiti di progetto

1. Desktop computer
   - buona performance per il singolo utente
   - costo moderato
   - software realizzato da terze parti
2. Server (es. web server, database server oppure supercomputer)
   - grande carico di lavoro
     (singola applicazione complessa vs. molti lavori “leggeri”)
   - software realizzato da terzi ma tipicamente customizzato
   - attenzione a espandibilità, affidabilità
   - il costo dipende dalla tipologia (a partire da desktop fino a supercomputer dotato di centinaia o migliaia di processori)

CLASSI DI CALCOLATORI

• I concetti visti sono validi per qualunque calcolatore, però classi di calcolatori diverse hanno diversi requisiti di progetto

1. Desktop computer
   - buona performance per il singolo utente
   - costo moderato
   - software realizzato da terze parti
2. Server (es. web server, database server oppure supercomputer)
   - grande carico di lavoro
     (singola applicazione complessa vs. molti lavori “leggeri”)
   - software realizzato da terzi ma tipicamente customizzato
   - attenzione a espandibilità, affidabilità
   - il costo dipende dalla tipologia (a partire da desktop fino a supercomputer dotato di centinaia o migliaia di processori)
3. Embedded computer (cellulari, elettrodomistici, automobili, ecc.)
   - progettati per specifiche applicazioni/funzioni integrate nell’hardware
   - performance “minima” per tali funzioni
   - attenzione al costo e a requisiti di consumo
   - attenzione all’affidabilità (spesso ottenuta con la semplicità!)
   - molti utilizzano *processor cores* (facile integrare hw per specifiche applicazioni + flessibilità nella scelta della linea di fabbricazione)


Embedded computer
Desktops
Servers
Millions of computers
1200
1100
1000
900
800
700
600
500
400
300
200
100
0
290
93
3
468
114
3
892
135
4
862
129
4
1122
131
5
1998
1999
2000
2001
2002

PROGRAMMA DEL CORSO

Prestazioni CPU:

Tempo di CPU - Metriche - CPI - Legge di Amdahl

Assembler MIPS:

Modalità di indirizzamento - Istruzioni - Formati - Compilazione (cenni)

Reti combinatorie:

Circuiti, tabelle, funzioni - Porte logiche - Algebra booleana - Rappresentazioni canoniche e sintesi mediante PLA e ROM - Esempi: codificatori, decodificatori, multiplexer, sommatore

Macchine sequenziali:

Bistabili - Temporizzazione - Automi a stati finiti di Moore e di Mealy – Sintesi di macchine sequenziali

Registri e memoria:

Registri e connessioni - Il register file - Tipi di memorie - SRAM – DRAM

Aritmetica dei calcolatori:

Notazione posizionale - Aritmetica binaria - Rappresentazione dei numeri negativi - ALU

PROGRAMMA DEL CORSO

Tecniche di specifica e realizzazione del controllo del processore:

Implementazione a singolo ciclo (unità di controllo e datapath)
Implementazione multi-ciclo (unità di controllo e datapath)
Microprogrammazione – Eccezioni – Prestazioni

Gerarchia di memoria:

Principio di località - Cache (a corrispondenza diretta, set-associative e a più livelli) – Frequenza di hit/miss - Tempo di hit - Penalità di miss –
Accesso in lettura e scrittura (write-through e write-back) –
Gestione dei miss in lettura e scrittura –
Prestazioni (tempo medio di accesso alla memoria) - Memoria virtuale (cenni)

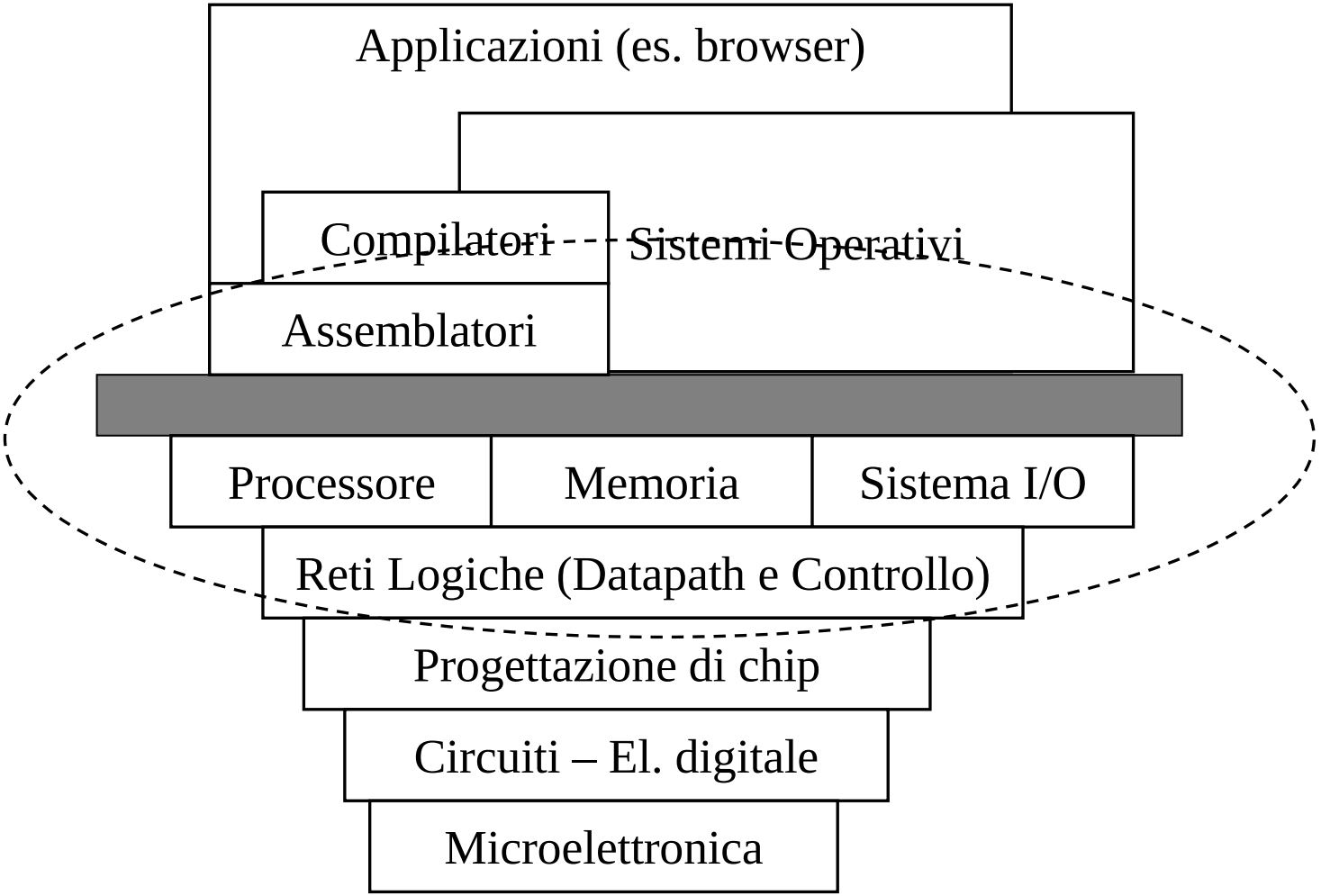
I/O:

Tipi di dispositivi - Interfacciamento dei dispositivi di I/O con la memoria ed il processore - Prestazioni

Nuovi Argomenti trattati nel corso di Calcolatori B

Calcolatori A	**Calcolatori B**
Assembler MIPS	Assembler Intel (cenni)
CPU: Multi-Ciclo	Pipeline
Memoria Cache	Memoria virtuale
I/O	Bus sincroni e asincroni

FOCUS DEI CORSI DI CALCOLATORI (A+B)



PERCHE' SERVE CONSIDERARE PIU' LIVELLI?

- Motivi "culturali": conoscere ciò che si usa
  - come un programma C o Java viene tradotto in assembler e linguaggio macchina?
  - in che modo il linguaggio macchina viene eseguito dall'hardware?
    Quali sono le possibili organizzazioni dei calcolatori?

  NB: cfr. apprendimento (e sviluppo) di sistemi operativi
  - perché le performance di un calcolatore sono migliori rispetto ad un altro?
- I livelli non sono mai completamente indipendenti, in particolare se si tiene in considerazione la performance

  <u>Esempio 1</u>

  I confini tra hardware e software sono sfumati:

  Hardware and and software are logically equivalent (AndrewTanenbaum)
  Hardware is just petrified software (Karen Panetta Lenz)

  ma "dovendo scegliere":

  E' vero che il software non potrebbe esercitare i poteri della sua leggerezza se non mediante la pesantezza dell'hardware; ma è il software che comanda, che agisce sul mondo esterno e sulle macchine, le quali esistono solo in funzione del software, si evolvono in modo d'elaborare programmi sempre più complessi. (Italo Calvino)

Esempio 2 (dal punto di vista del programmatore)

Le prestazioni di un programma sono determinate da:
- il programma stesso
- prestazioni dell'hardware nell'eseguire le istruzioni dell'ISA
- il compilatore, che può produrre software "ottimizzato" per l'hw sottostante

Esempio 3 (dal punto di vista del progettista hardware)

Necessità di conoscere i livelli più alti (esempi):
- le prestazioni sono definite con riferimento al sw (carico di lavoro-benchmark)
- le ottimizzazioni dell'hw si fanno sulla base dei carichi di lavoro previsti