

Sistema di Elaboratore dell'Informazione (Computer)

- Un **Sistema di Elaborazione dell'Informazione** (o **Elaboratore Elettronico** o **Computer**) è una macchina *digitale, elettronica, automatica e programmabile* capace di effettuare trasformazioni o elaborazioni sui dati.
- **Digitale**: l'informazione è rappresentata in forma numerica discreta.
- **Elettronica**: la logica di memorizzazione e manipolazione dell'informazione sono implementate con tecnologie di tipo elettronico.
- **Automatica**: è in grado di eseguire una successione di operazioni in modo autonomo (senza intervento di un operatore umano).
- **Programmabile**: un operatore umano (o elettronico ma con decisione iniziale di un umano) può modificare “facilmente” la successione di operazioni da eseguire.

Le operazioni che l'elaboratore deve eseguire sono “comandate” all'elaboratore stesso mediante delle istruzioni espresse in una forma (un linguaggio) comprensibile all'elaboratore. Le istruzioni sono delle piccole sequenze di bit.

Le istruzioni (**il programma**) che l'elaboratore esegue costituiscono il **software**.

Software vs. hardware

Il programma (le istruzioni) che l'elaboratore deve eseguire potrebbero anche essere codificate nell'hardware, ma si perderebbe la possibilità di modificare il programma.

Il vantaggio di poter modificare il programma consiste nella flessibilità d'uso del computer. La flessibilità del computer ne fa uno strumento general-purpose, cioè uno strumento di uso generico, che viene specificato e specializzato dal software.

Lo svantaggio del computer rispetto ad un dispositivo esclusivamente hardware risiede nella maggior lentezza di esecuzione.

Requisiti di un Elaboratore

Dove mantenere tutto il programma da eseguire?	memoria
Dove mantenere tutti i dati su cui operare?	memoria
Dove eseguire le singole istruzioni per operare sui singoli dati?	CPU(ALU,reg.)
Come trasferire le istruzioni ed i dati per l'esecuzione?	BUS
Dove mantenere i dati quando l'elaboratore è spento?	Dischi
Come far interagire l'utente con l'elaboratore?	Input/Output

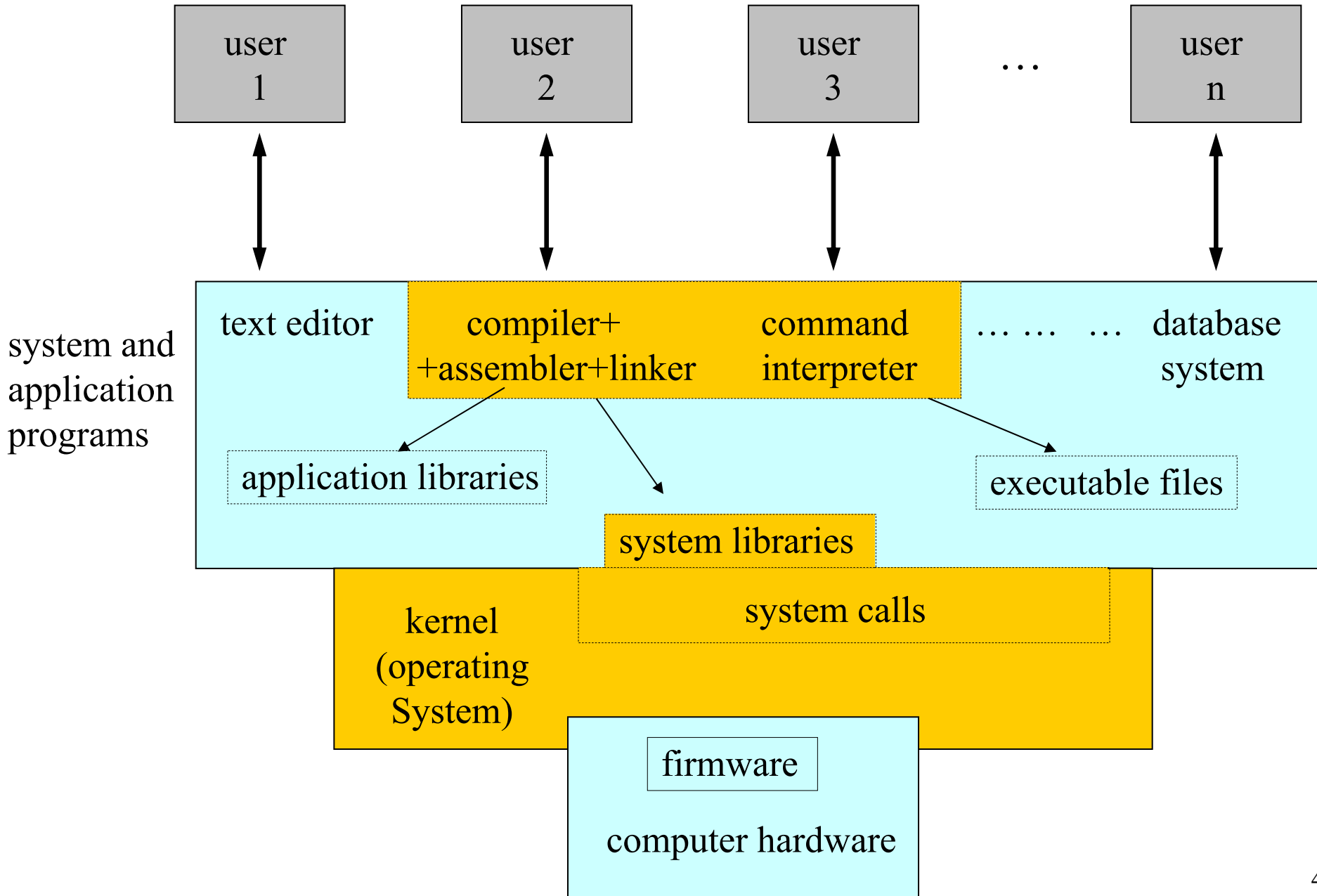
Un elaboratore *programmabile* è caratterizzato dalla presenza di CPU e memoria separate. La CPU preleva le istruzioni dalla memoria e le esegue !!!!

Organizzazione di un Computer

Un Computer in generale è organizzato con una struttura “gerarchica”, ossia a livelli:

1. **Hardware:** composto dai componenti *fisici* del sistema.
2. **Firmware:** insieme di microprogrammi che svolgono operazioni più o meno complesse nei singoli componenti. Può essere modificato solo con tecniche speciali.
 - es: BIOS, risiede nella ROM, esegue nella CPU al boot.
 - es: Firmware delle schede di rete wireless, coordina la trasmissione dei bit sulla portante radio
3. **Software:** i programmi eseguiti dal sistema, che a loro volta si suddividono in:
 - 3.1 Software di base: più conosciuto con il termine di *Sistema Operativo* , è dedicato alla gestione delle risorse hardware, alla gestione di più utenti, alla gestione di più processi (i programmi quando sono in esecuzione).
 - 3.2 Software applicativo: insieme dei programmi dedicati a specifiche esigenze applicative da parte dell'*utente*.
4. **Utente:** capace di qualunque atto dannoso, più o meno involontario (**utonto** ?)

Organizzazione di un Computer



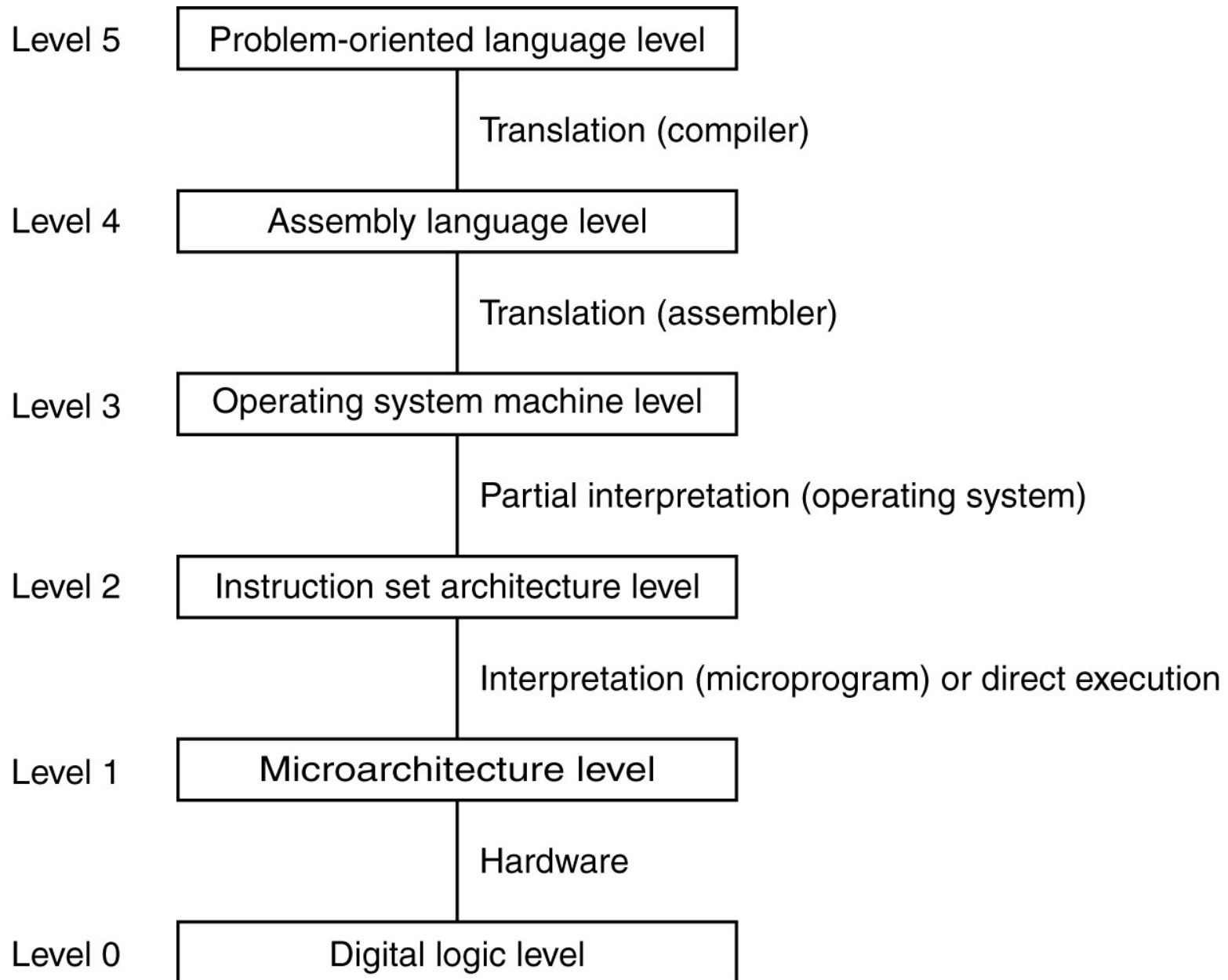
Organizzazione del Corso di Architettura degli Elaboratori

Il corso analizza i diversi livelli architetturali degli elaboratori.

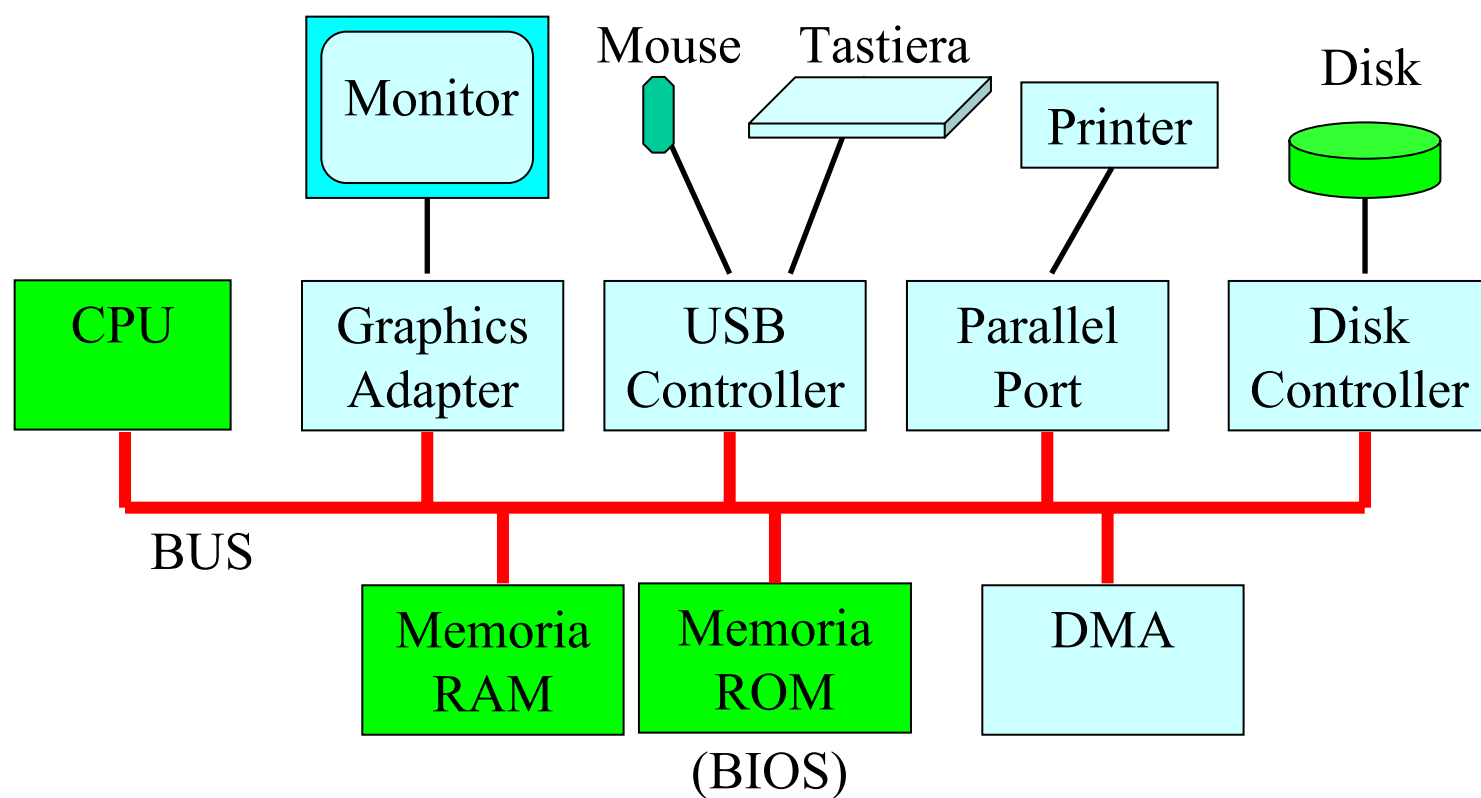
Si vuole fornire allo studente gli strumenti necessari ad approfondire mediante prove ed esercitazioni pratiche gli argomenti introdotti, perciò si inizia dall'alto, studiando le Applicazioni e come queste vengono costruite ed eseguite.

- Livello applicazioni: Programmi Eseguibili, Processi, Organizzazione della memoria dei processi, Librerie, Librerie Dinamiche, compilatore, linker, loader, **Linguaggio ANSI C.**
- Livello assembly: Linguaggio Assembly, Assemblatore, mappaggio del linguaggio C sull'Assembly.
- Livello Sistema Operativo: Indirizzamento della memoria, System Calls, Interrupts (maggiori approfondimenti nel corso di sistemi operativi).
- Livello Livello ISA (Instruction Set Architecture): le istruzioni utilizzabili messe a disposizione dalla macchina nuda, e dal BIOS.
- Livello Microprogrammazione: come il firmware realizza le singole istruzioni.
- Livello Logica Digitale: come è fatto l'hardware.

Organizzazione di un Computer – Linguaggi e Strumenti



Hardware di un Computer



CPU esegue istruzioni, effettua calcoli, controlla input/output, coordina spostam. dati.

BUS trasferisce i dati tra la memoria e gli altri dispositivi.

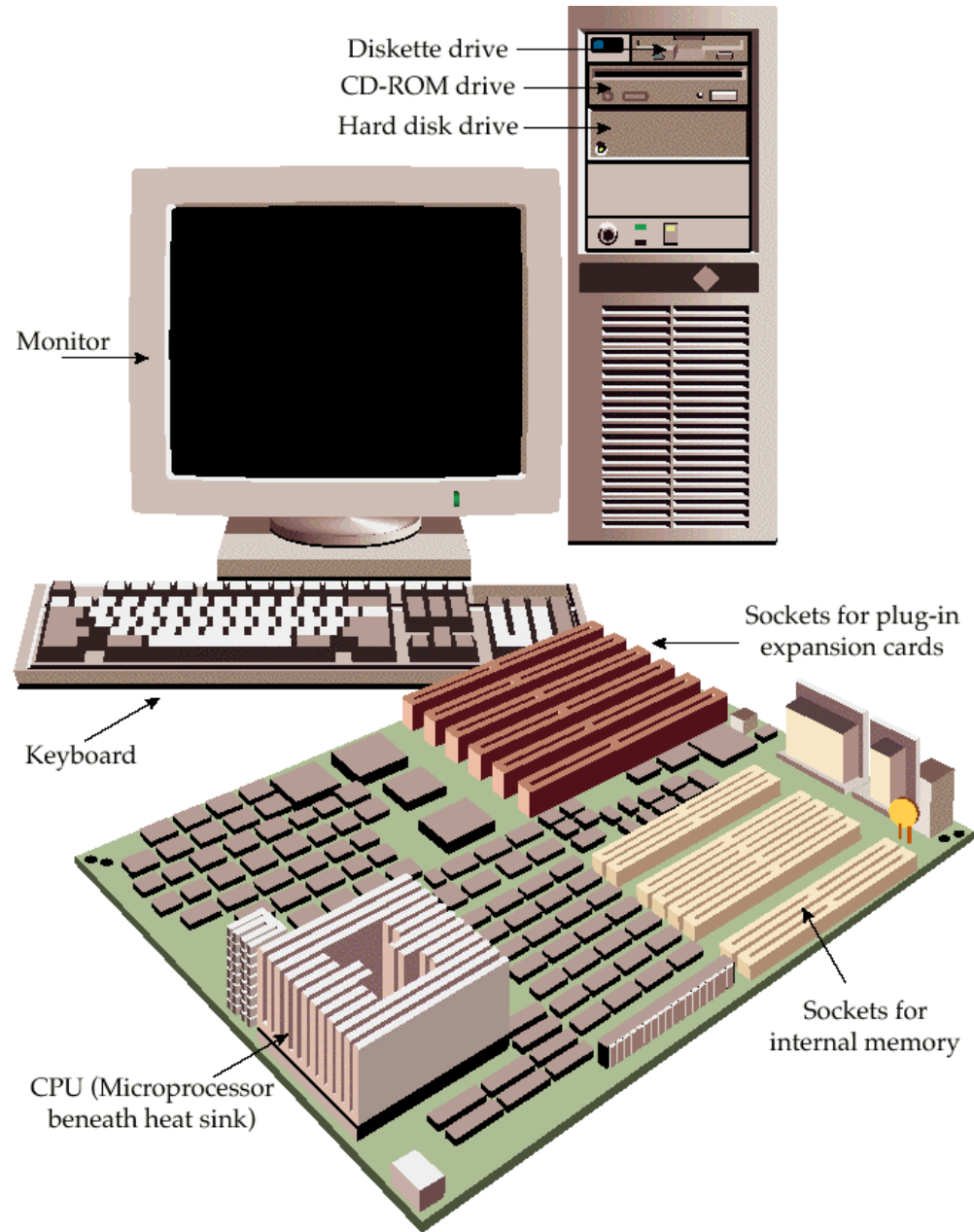
Memoria RAM (volatile) mantiene programmi (seq. di istruzioni) quando devono essere eseguiti, e i dati che i programmi usano, ma solo finché il computer è acceso.

Memoria ROM (permanente) contiene il BIOS, cioè le istruzioni per fornire i servizi base, usate dalla CPU anche nel momento dell'accensione del computer (il BOOT).

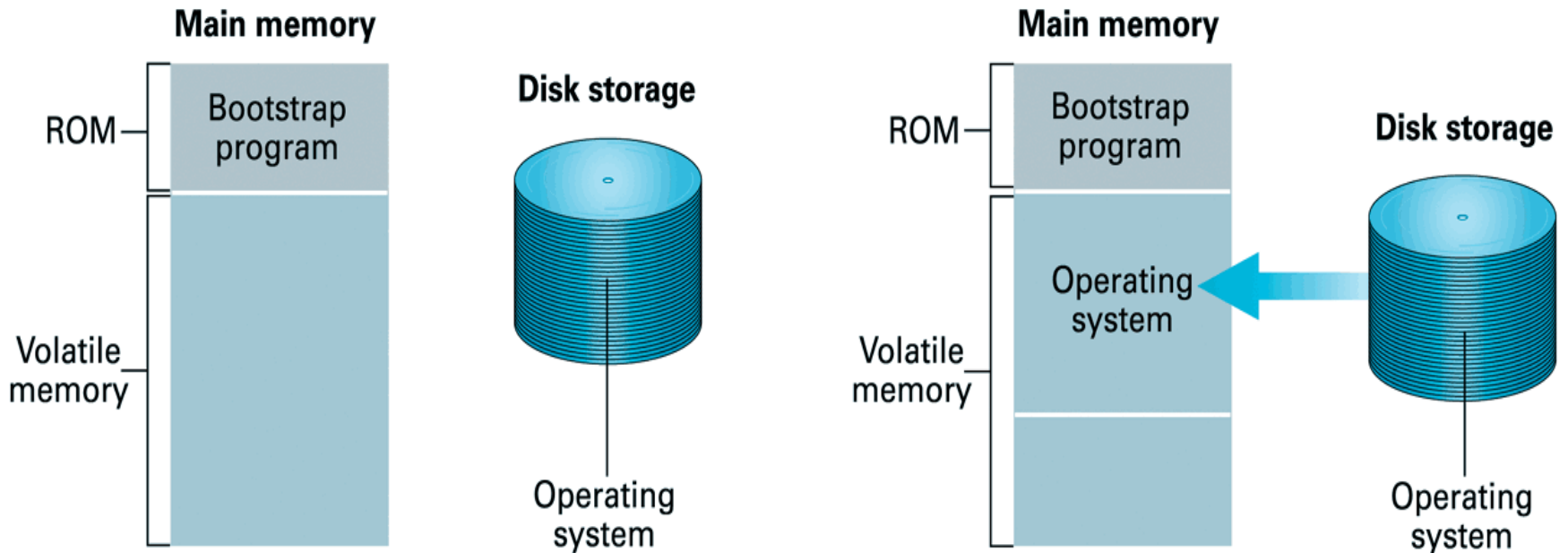
Dischi mantengono enormi quantità di programmi e dati, anche dopo lo spegnimento del computer.

Sono inoltre presenti alcune periferiche di **I/O** (Input/Output).

Un tipico Computer



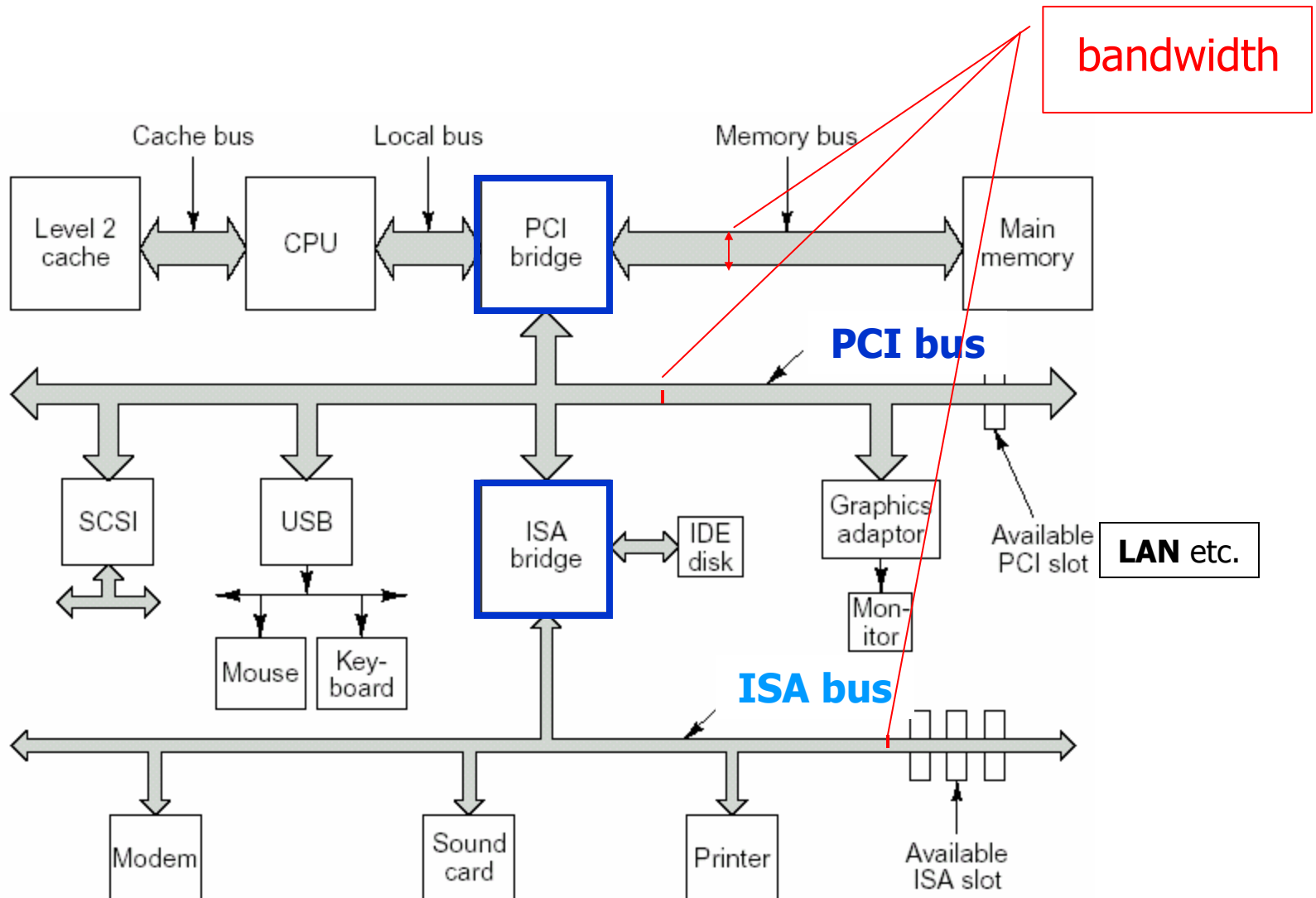
L'avvio - Bootstrap



Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.

Step 2: Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it.

Diversi tipi di BUS



Codifica Binaria dell'Informazione

Il computer utilizza un **alfabeto binario**:

usiamo dispositivi elettronici digitali in grado di assumere due solo stati:
acceso/spento, ON/OFF, 1/0, VERO/FALSO.

Informazione Minima è il **bit** (da **B**inary **digIT**): quantità d'informazione che si ottiene selezionando una configurazione da un insieme che ne contiene due.

Con un bit rappresento due valori 0, 1.

Con due bit rappresento 4 valori 00, 01, 10, 11.

Con tre bit rappresento 8 valori 000, 001, 010, 011, 100, 101, 110, 111.

In generale con N bit rappresento 2^N valori.

La memoria di un computer è organizzata a blocchi di 8 bit

Byte blocco di 8 bit $\Rightarrow 2^8 = 256$ stati

Notazione Posizionale

Consideriamo i numeri interi assoluti (numeri naturali più lo zero).

Notazione posizionale in base b (generica):

Alfabeto: $c_i \in \{0 \dots b-1\}$ comprende b simboli (cifre)

Un numero è rappresentato come una sequenza di simboli dell'alfabeto.

La sequenza $(c_k c_{k-1} \dots c_1 c_0)_b$

rappresenta, secondo la base b , il numero che vale:

$$N = c_k * b^k + c_{k-1} * b^{k-1} + \dots + c_0 * b^0 = \sum_{(i=0 \dots k)} c_i * b^i$$

Notare che le posizioni (gli indici) iniziano da destra con il valore 0.

es. in base 10: $(937)_{10} = 9*10^2 + 3*10^1 + 7*10^0 = 900 + 30 + 7$

Definizione: Poiché i simboli più a destra nella sequenza sono i coefficienti delle potenze con esponente minore, si dice che **quello più a destra è il simbolo meno significativo**, e **quello più a sinistra è il simbolo più significativo**.

c_0 simbolo meno significativo (Least Significant)

c_k simbolo più significativo (Most Significant)

Notazione Posizionale – base binaria

Basi più frequentemente utilizzate:

base 2 : $c_i \in \{0, 1\}$

base 8: $c_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$

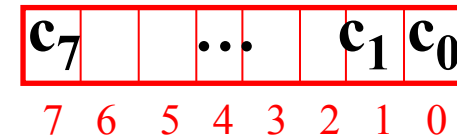
base 10: $c_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

base 16: $c_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Codifica Binaria

Numeri binari: base $b = 2$

Alfabeto binario: cifre $c_i \in \{0, 1\}$



$(c_k \dots c_1 c_0)_2$ rappresenta: $N = \sum_{(i=0 \dots k)} c_i * 2^i$

c_k (MSb, Most Significant bit) c_0 (LSb, Least Significant bit)

Conversione Binario->Decimale

$$N = (1101)_2$$

$$N = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

NB: se non indico la base es: $(..)_{10}$ intendo riferirmi alla base 10.

Notazione Posizionale – basi importanti

Vediamo un esempio, nel caso di base decimale, che usa i simboli $\{0, 1, 2, \dots, 8, 9\}$.

Il valore del numero 201 (qui espresso in base 10), è calcolato come:

$$2*10^2 + 0*10^1 + 1*10^0 = 2*100 + 0*10 + 1*1 = 200 + 0 + 1 = 201$$

Nel caso di base esadecimale, si usano i 16 simboli $\{0, 1, 2, 3, \dots, 8, 9, A, B, C, D, E, F\}$, con A che vale 10, B vale 11, C vale 12, D vale 13, E vale 14 e F vale 15.

Il valore del numero $(C9)_{16}$, espresso in base 16, è calcolato come:

$$C*16^1 + 9*16^0 = 12*16 + 9*1 = 192 + 9 = 201$$

- **E' importante sottolineare che con 2 cifre esadecimali (cioè con una coppia di simboli che possono assumere ciascuno 16 diversi valori) io posso rappresentare 256 valori, esattamente quanti sono i valori esprimibili con un singolo Byte.**
- **Per tale motivo, spesso si utilizza la base esadecimale per indicare il valore di un Byte, poiché servono esattamente due cifre esadecimali per rappresentare tutti e soli i possibili valori di un byte, dal valore $0=(00)_{16}$ al valore $255=(FF)_{16}$**

Notazione Posizionale – casi importanti – base esadecimale

Codifica Esadecimale

Numeri esadecimali: base $b = 16$

Alfabeto: cifre $c_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

A vale 10, B vale 11, C vale 12, D vale 13, E vale 14, F vale 15

$(c_k \dots c_1 c_0)_{16}$ rappresenta: $N = \sum_{(i=0 \dots k)} c_i * 16^i$

Conversione Esadecimale->Decimale

$$N = (F2)_{16}$$

$$N = F \times 16^1 + 2 \times 16^0 = 15 * 16 + 2 * 1 = 242_{10}$$

NB: se non indico la base es: $(..)_2$ intendo riferirmi alla base 10.

Anticipazione – Notazione Posizionale nel Linguaggio C

Il linguaggio ANSI C, nella sua smisurata saggezza, mette a disposizione una notazione semplice, senza tanti pedici o parentesi, per rappresentare dei numeri interi.

In particolare, nella notazione posizionale, il linguaggio C indica la base mediante la quale è rappresentato un numero, premettendo al numero un prefisso.

Ai numeri espressi in base 2 viene messo il prefisso `0b`

Ai numeri espressi in base 8 viene messo il prefisso `0`

Ai numeri espressi in base 16 viene messo il prefisso `0x`

Ai numeri espressi in base 10 non viene messo alcun prefisso.

In base 2 **0b101** = 5

In base 2 **0b10000** = 16

In base 8 **020** = 16

In base 8 **077** = 63

In base 16 **0xFF** = 255

In base 16 **0x10** = 16

In base 16 **0x32** = 50

In base 10 44

Rappresentazione di numeri interi positivi in base esadecimale

Rappresentazione di Interi Positivi (≥ 0)

Se usiamo **un Byte di memoria**, possiamo memorizzare $2^8=256$ valori da 0 a 255 e li possiamo rappresentare usando 2 cifre esadecimali.

$$0x00=0$$

$$0x20=32$$

$$0x33=51$$

$$0x80=128$$

$$0xC0=192$$

$$0xFF=255$$

Se usiamo **due Byte di memoria**, possiamo memorizzare $2^{16}=65536$ valori da 0 a 65535 e li possiamo rappresentare usando 4 cifre esadecimali.

$$0x0000=0$$

$$0x0100=256$$

$$0x0400=4*256=1024$$

$$0x8000=128*256=32768$$

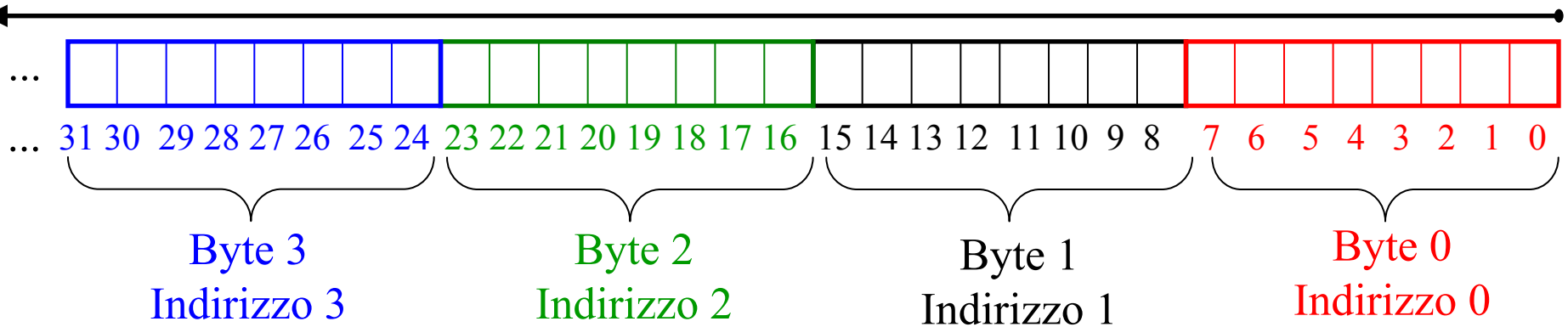
$$0xFFFF=65535$$

Se usiamo **quattro Byte di memoria**, possiamo memorizzare $2^{32}=4\,294\,967\,296$ valori da 0 a 4miliardi e un po', e li possiamo rappresentare usando 8 cifre esadecimali.

$$0xFFFFFFFF=4294967295$$

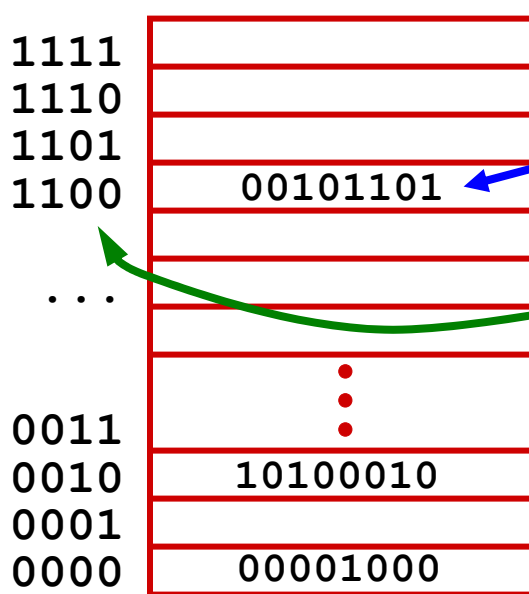
Organizzazione della Memoria Principale

Indirizzi



Indirizzi

Celle di memoria=
Blocchi di un Byte



La memoria è organizzata come una sequenza di celle ciascuna costituita da un singolo Byte (8 bit).

Una cella di memoria è caratterizzata da:

- Un **valore**

Il contenuto della cella stessa, visto come numero binario.

- Un **indirizzo**

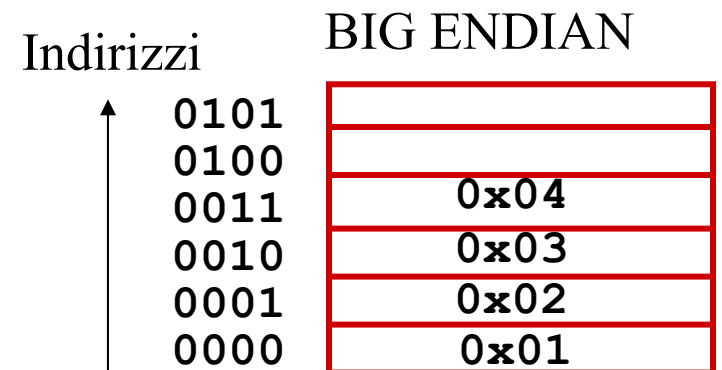
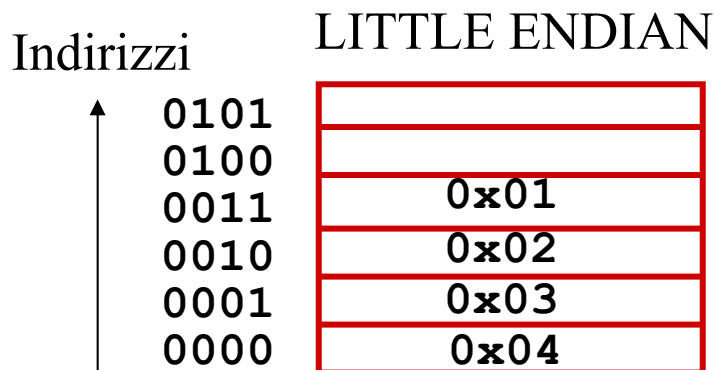
L'indirizzo di inizio della cella.

Byte 2, valore 162, indirizzo 0010

Byte 0, valore 8, indirizzo 0000

Convenzione: Endianess – Ordine dei Byte degli Interi

- Un numero intero formato da 2, 4 o 8 Byte può essere collocato in quei byte in diversi modi. Essenzialmente i processori si suddividono in due classi a seconda di questa caratteristica, detta Endianess. Si parla di:
- **Little Endian:** i Byte più significativi del numero stanno nei Byte di indirizzo maggiore. (LSB, Least Significant Byte first).
- **Big Endian:** i Byte più significativi del numero stanno nei Byte di indirizzo Minore. (MSB, Most Significant Byte first).
- Ad es. consideriamo il numero formato da 4 Byte e rappresentato in base esadecimale da 0x01020304.
- Il Byte più significativo è quello che contiene il valore 0x01.
- Il Byte meno significativo è quello che contiene il valore 0x04.
- Ecco come i Byte sono collocati nelle due classi.



Un esempio di Organizzazione della CPU

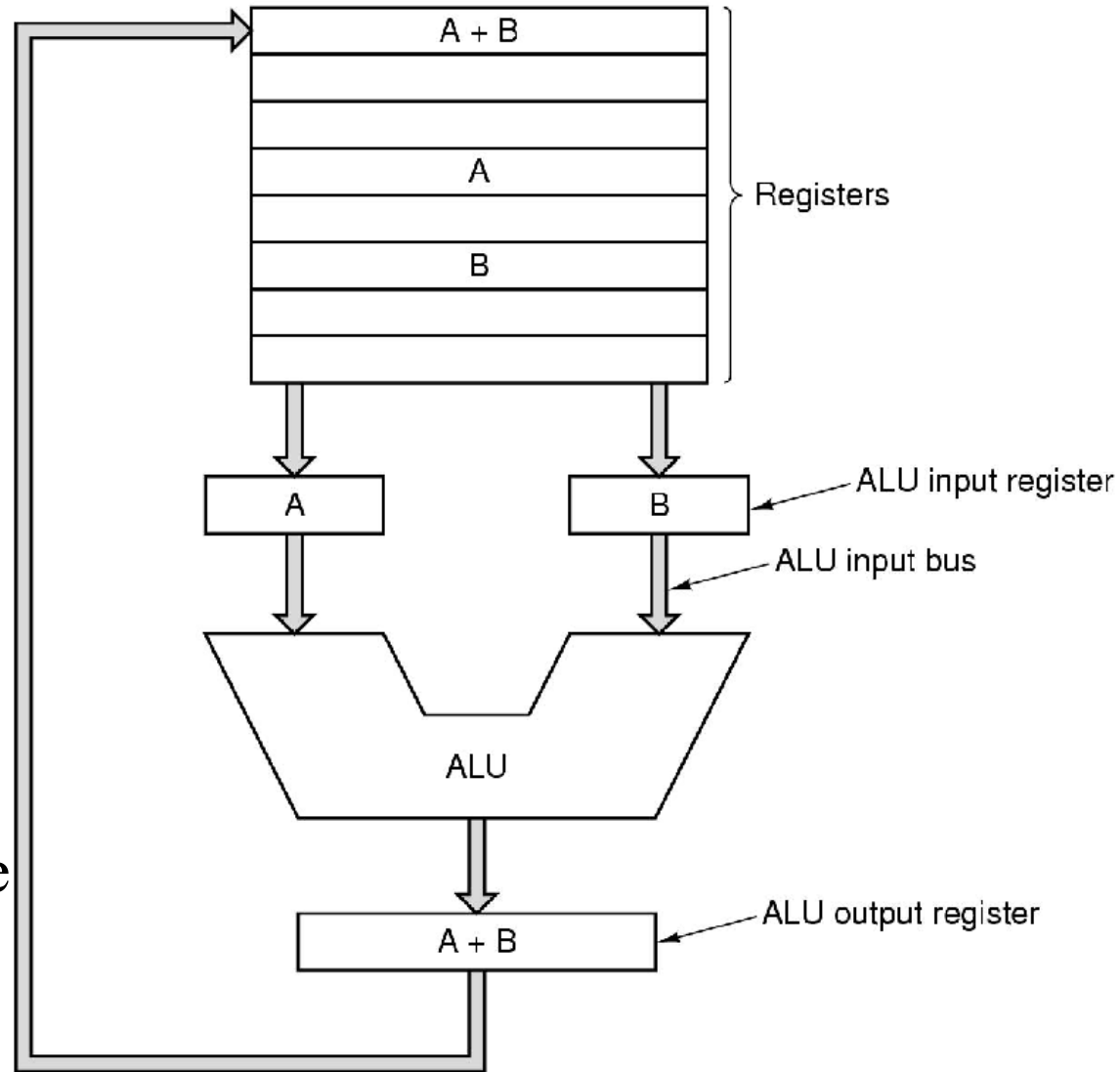
Data Path: organizzazione interna di una CPU (registri, ALU, bus interno)

Registro: memorie veloci per dati temporanei

Registri A e B, input dell'ALU, e A+B output dell'ALU

Istruzioni registro-registro e registro-memoria

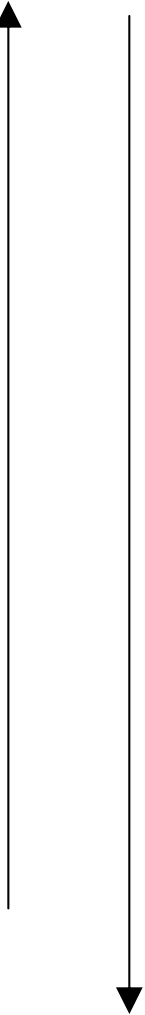
Ciclo del data path



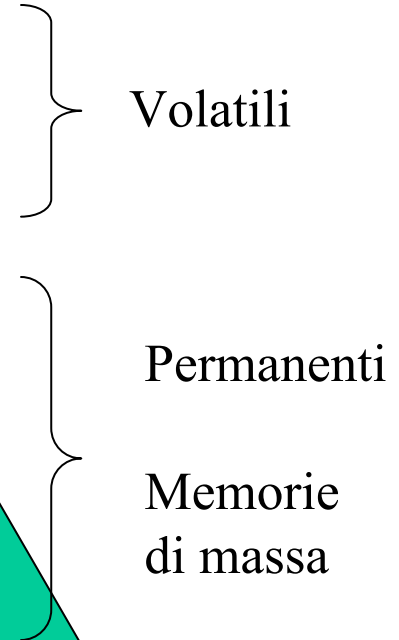
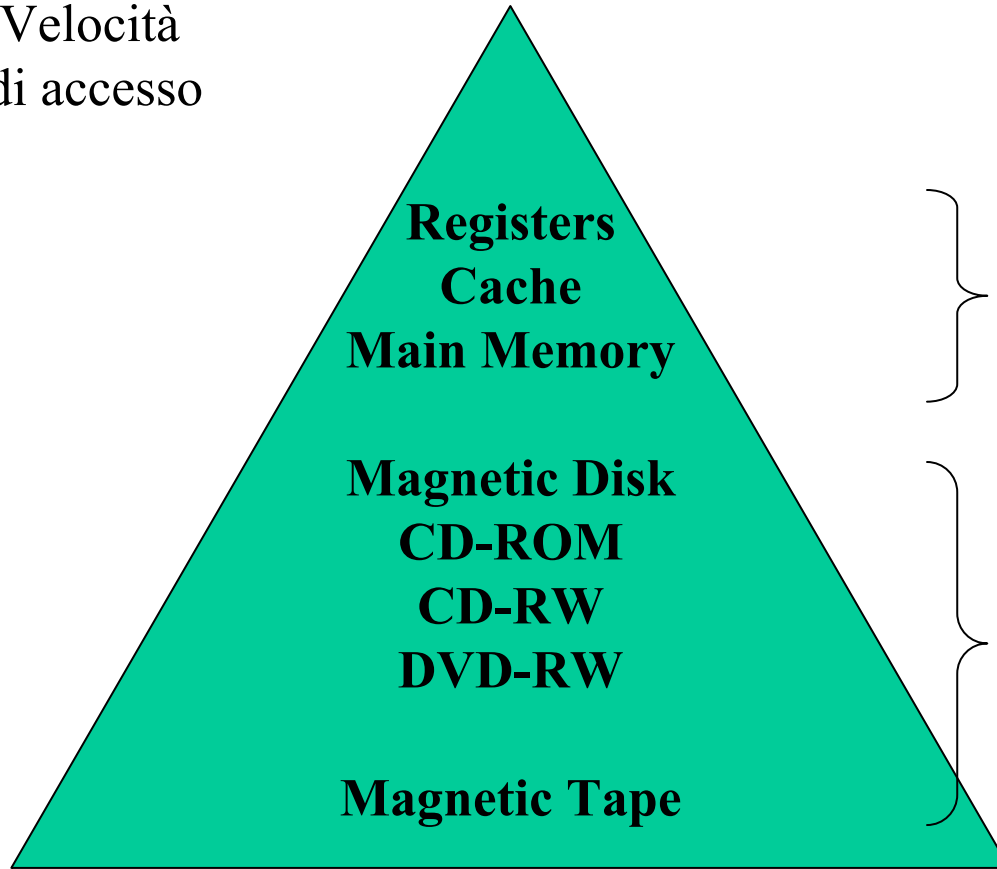
Gerarchie di Memorie

Costo

Velocità
di accesso

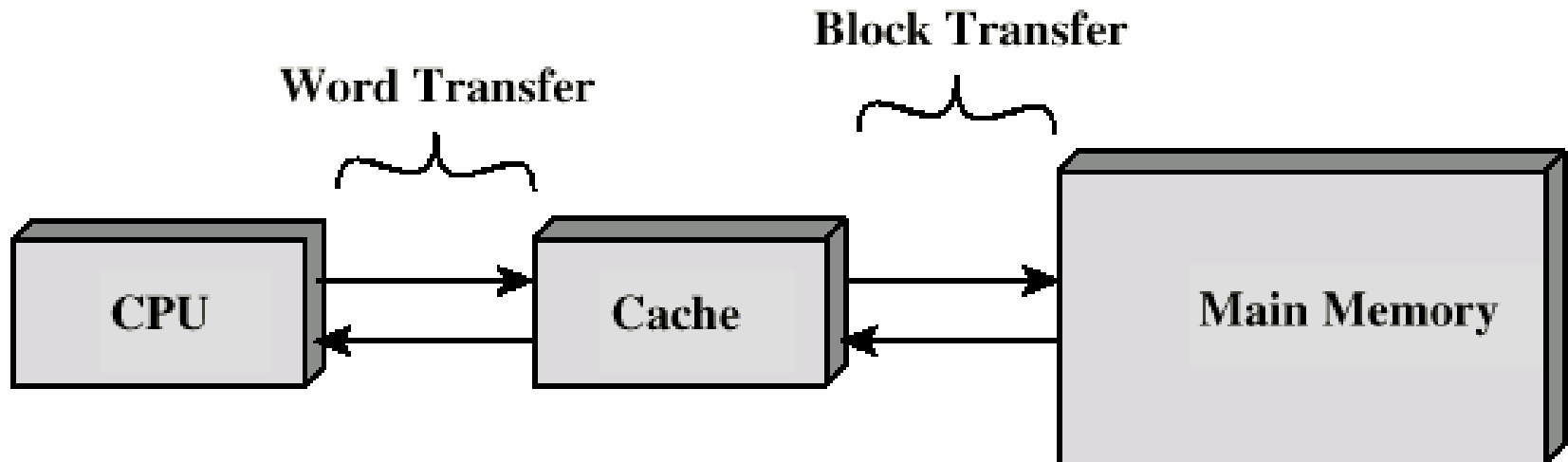


Dimensione
Dati
Memorizzabili

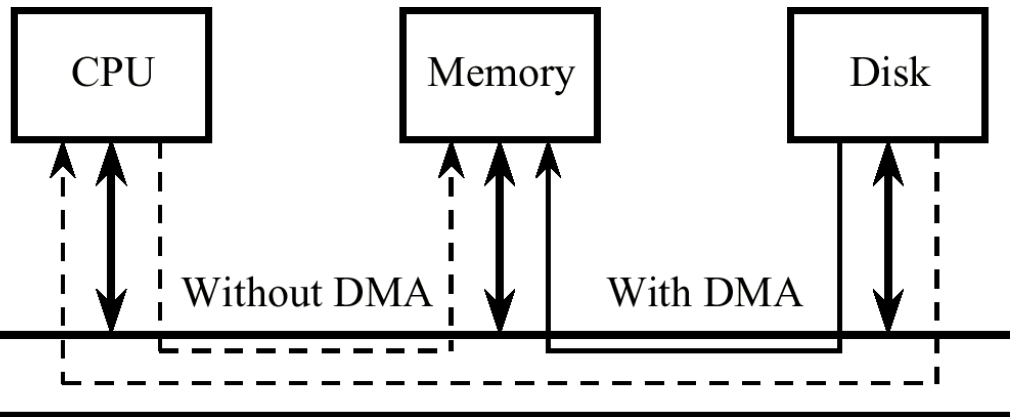


Il concetto di Cache della memoria

- La memoria veloce è poca, perché costa troppo.
- Viene inserita una piccola quantità di memoria veloce tra la CPU e la memoria principale.
- Può essere collocata anche sul disk controller, o anche tra i diversi componenti della CPU.



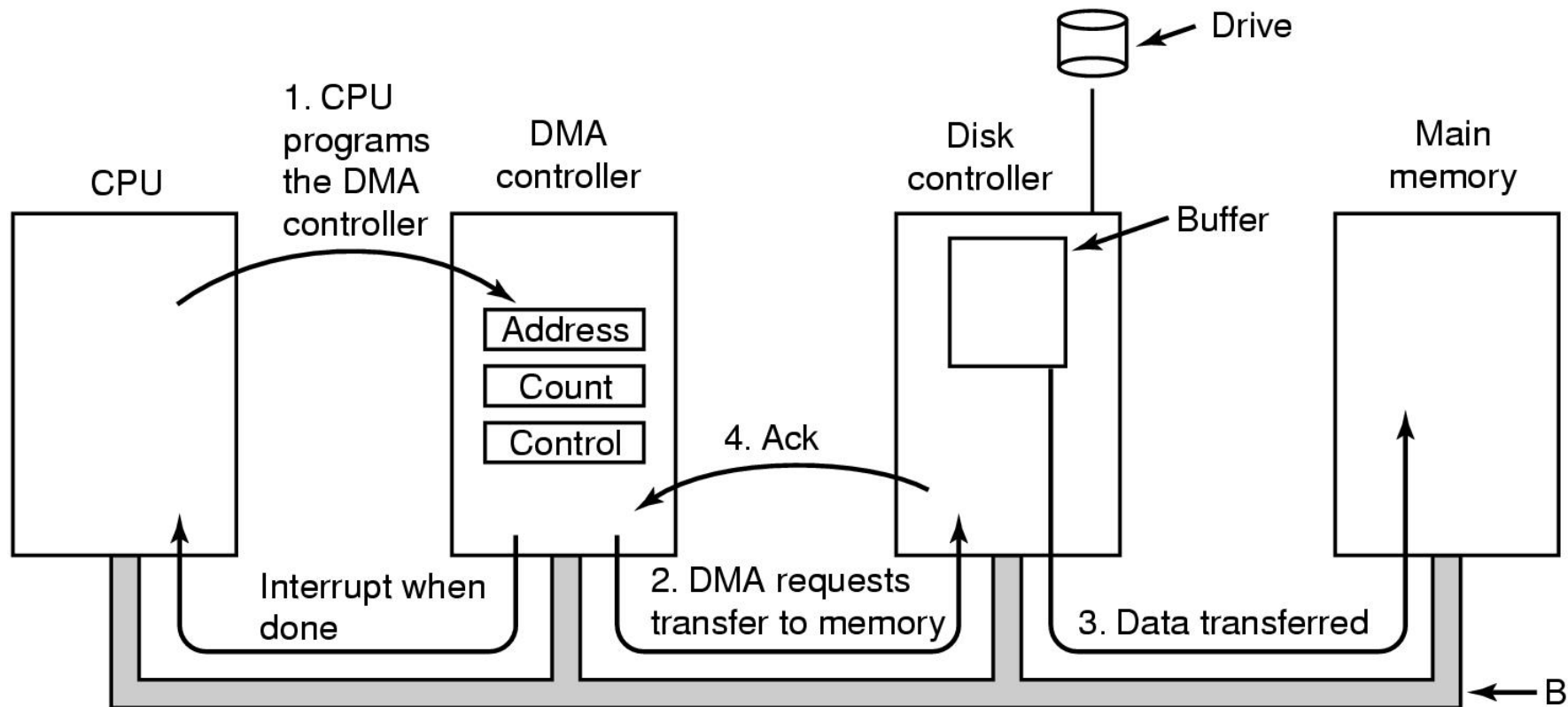
Accesso Diretto alla memoria – DMA - Direct Memory Access



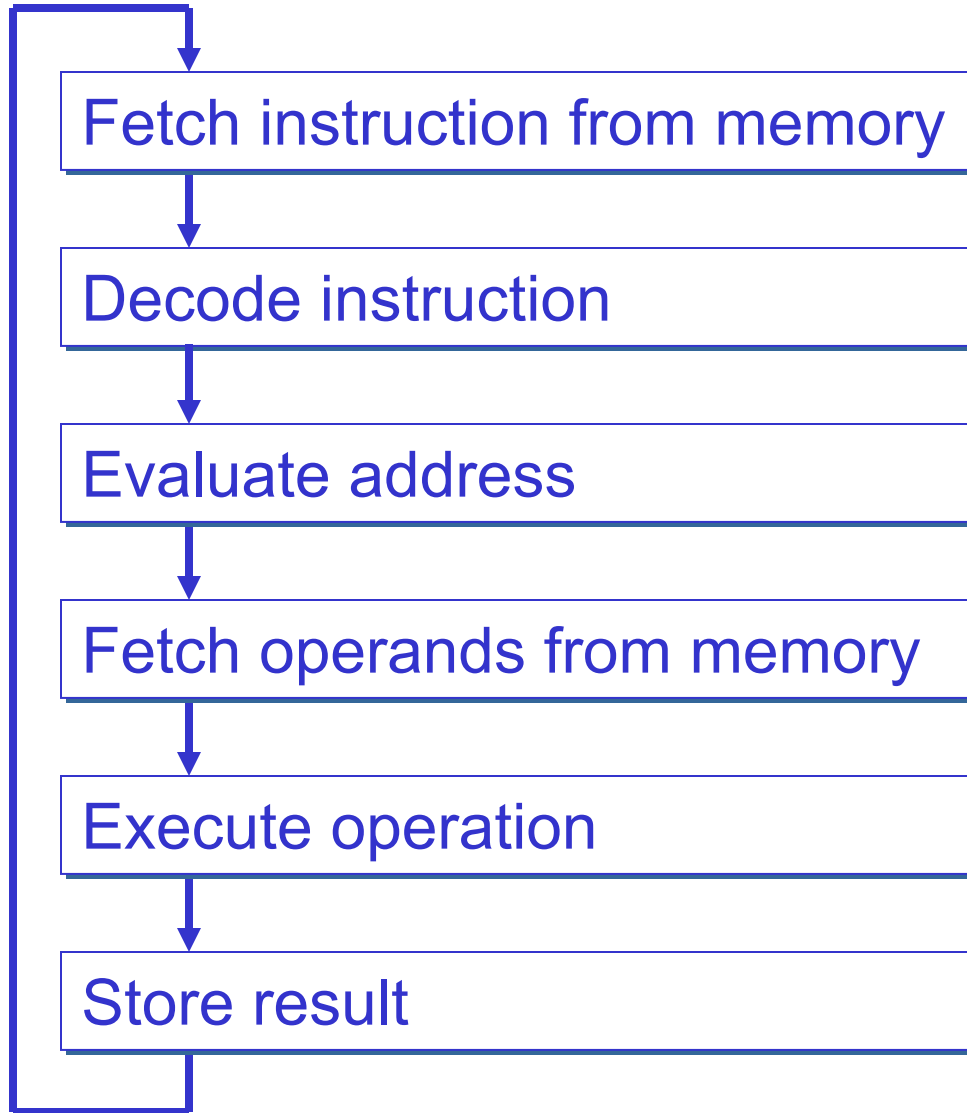
Il trasferimento dei dati, dal disco alla memoria, effettuato dal DMA non coinvolge la CPU tranne che:

- Quando la CPU ordina al DMA lo spostamento del blocco dati.
- Quando il DMA avvisa la CPU dell'avvenuto trasferimento.

Bus dell'avvenuto trasferimento.



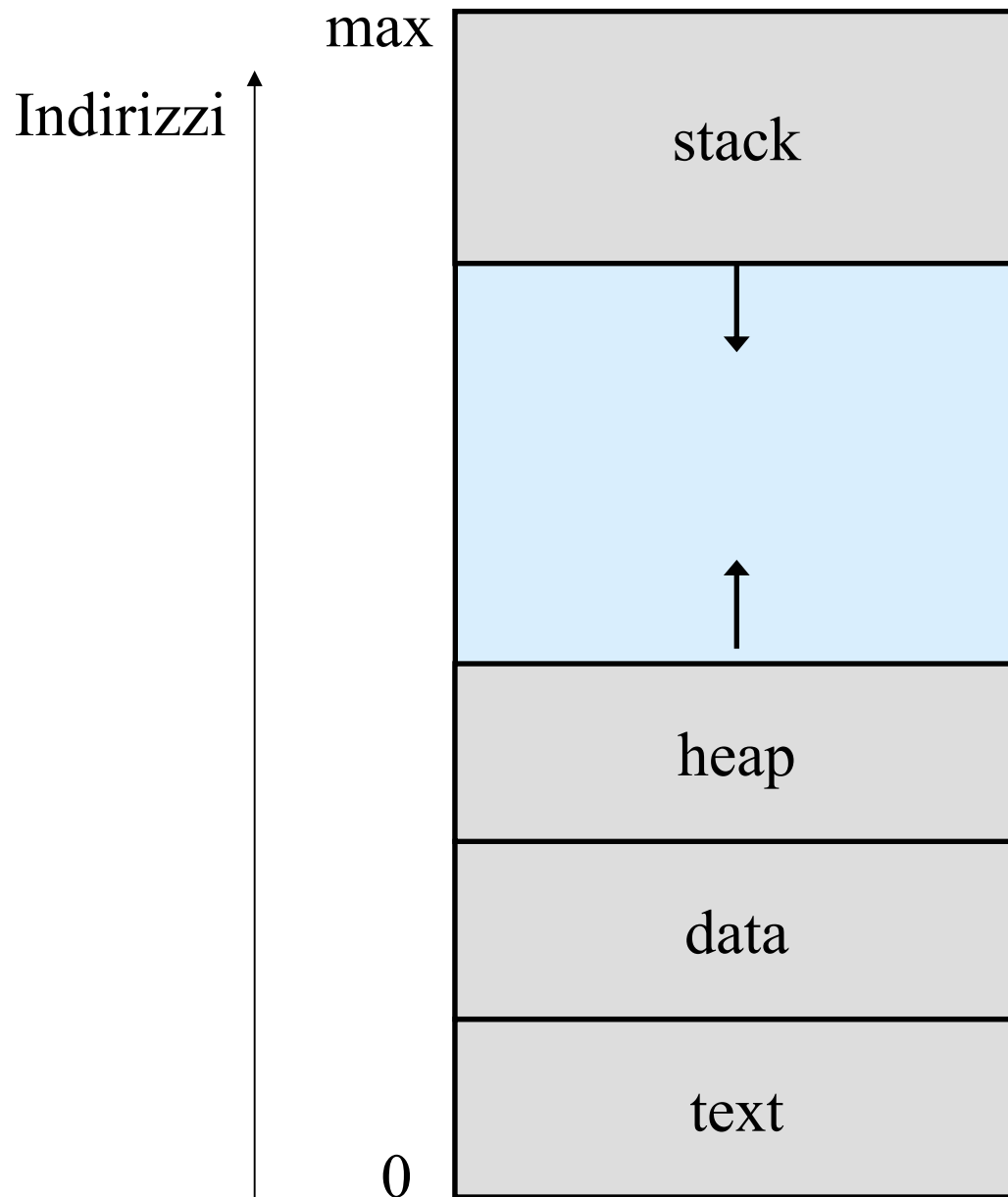
Instruction Processing - Ciclo di Neumann



Relazione tra “Ciclo di Neumann” e “Ciclo di data path”

- come visto nella slide precedente, il processo di un’istruzione macchina (detto ciclo di Neumann) è composto da diverse fasi,
- alcune di queste fasi richiedono di accedere alla memoria (in lettura (fetch dell’istruzione, fetch degli operandi) o in scrittura (store del risultato)
- altre fasi possono richiedere di volgere dei calcoli aritmetici o valutazioni logiche
- Ciascuna di queste fasi viene eseguita utilizzando uno o più cicli di data path
- Quindi c’è qualcosa che si occupa di far corrispondere
 - ad una istruzione macchina
 - la sequenza di cicli di data path necessari alla completa esecuzione dell’istruzione.
- Nelle CPU meno recenti, questo qualcosa era realizzato esclusivamente mediante circuiteria elettronica (hardware).
- Attualmente, invece, **esiste del software collocato all’interno della CPU stessa (firmware di microprogrammazione) che si occupa di interpretare le istruzioni macchina e di far eseguire la corrispondente sequenza di cicli di data path**

Organizzazione della Memoria usata dai Processi



Il concetto di Thread nei Processi

