Struttura dei calcolatori

A cura di:
Luca Breveglieri[*] Giacomo Buonanno[#]
Roberto Negrini[*] Giuseppe Pozzi[*] Donatella Sciuto[*]

[*] DEI, PoliMI, Milano
[#] LIUC, Castellanza (VA)

breveglieri,negrini,pozzi,sciuto@elet.polimi.it
buonanno@liuc.it
- versione del 24 marzo 2003 -

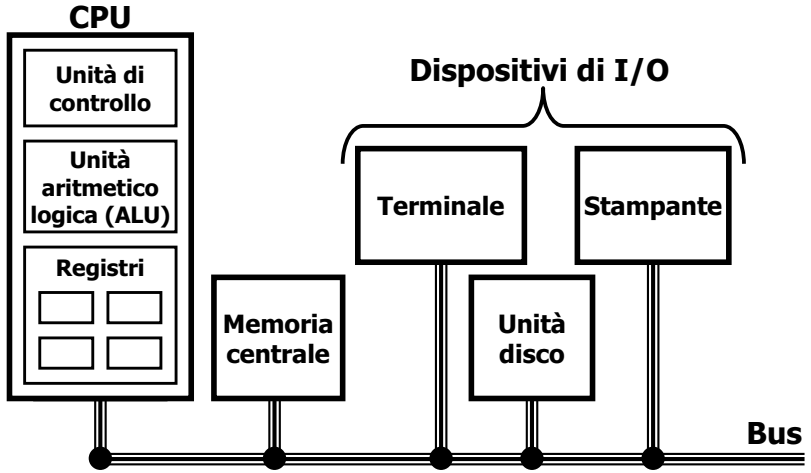
Unità Centrale di Elaborazione (CPU)

- **Bibliografia**:

  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - **paragrafo 2.1**

- **Sommario**:
  1. **Organizzazione** della CPU.
  2. **Esecuzione** delle istruzioni.
  3. **Differenze** fra **RISC** e **CISC**.
  4. **Principi di progettazione** per i moderni calcolatori.
  5. **Parallelismo** a livello delle **istruzioni**.
  6. **Parallelismo** a livello di **processore**.

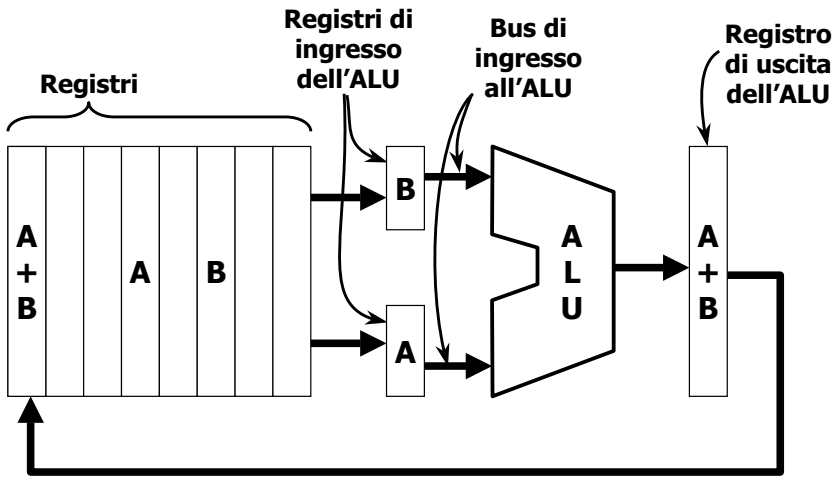
Organizzazione tipica di un calcolatore "bus oriented"



Elementi di una CPU

- **Unità di controllo**
  - legge le istruzioni dalla memoria e ne determina il tipo.
- **Unità aritmetico–logica**
  - esegue le operazioni necessarie per eseguire le istruzioni.
- **Registri**
  - **memoria ad alta velocità** usata per risultati temporanei e informazioni di controllo;
  - il **valore massimo** memorizzabile in un registro è determinato dalle **dimensioni** del registro;
  - esistono registri di uso generico e registri specifici:
    - **Program Counter** (**PC**) – qual è l'istruzione successiva;
    - **Instruction Register** (**IR**) – istruzione in corso d'esecuzione;
    - **...**

Struttura del "data path"



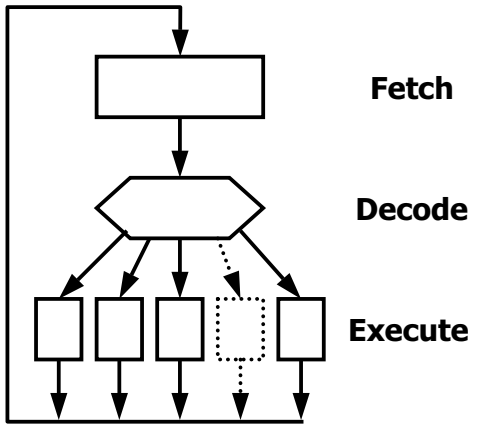
Categorie di istruzioni

- **Registro–Memoria**
  - gestiscono il **trasferimento dati** tra i registri della CPU e la memoria centrale;
  - l'unità di informazione trasferita tra memoria e registri prende il nome convenzionale di **parola** (**word**);
- **Registro–Registro**
  - utilizzano il contenuto dei registri per svolgere operazioni (tramite l'ALU) e memorizzano il risultato in un registro (**ciclo del data path**);
  - il **ciclo del data path** è al centro del funzionamento di quasi tutte le CPU, **quanto più piccolo è il tempo di ciclo tanto più veloce è il calcolatore**.

Esecuzione delle istruzioni

- Ciclo **Fetch–Decode–Execute** (**leggi–decodifica–esegui**)
  1. Prendi l'**istruzione corrente** dalla memoria e mettila nel **registro istruzioni** (**IR**).
  2. **Incrementa** il **program counter** (**PC**) in modo che contenga l'indirizzo dell'istruzione successiva.
  3. Determina il tipo dell'istruzione corrente (**decodifica**).
  4. Se l'istruzione usa una parola in memoria, determina dove si trova.
  5. Carica la parola, se necessario, in un registro della CPU.
  6. **Esegui** l'istruzione.
  7. Torna al punto 1 e inizia a eseguire l'istruzione successiva.

Ciclo Fetch–Decode–Execute



Interprete Java che simula la CPU

```
public class Interp {
  static int PC;            // program counter
  static int AC;            // accumulatore
  static int instr;         // un registro istruzione corrente
  static int instr type;    // tipo di istruzione (opcode)
  static int data loc;      // indirizzo dei dati, –1 se non presente
  static int data;          // operando corrente
  static boolean run bit = true;        // un bit di stop

  public static void interpret(int memory[], int starting_address) {
    PC = starting–address;
    while (run bit) {
      instr= memory[PC];       // leggi la prossima istruzione e mettila in instr
      PC = PC + l;             // incrementa il program counter
      instr type = get instr type(instr);        // determina il tipo di istruzione
      data_loc = find_data(instr, instr type); // trova i dati (–1 se non presenti)
      if (data–loc >= 0) // se data loc è –1, non c'è operando
        data = memory[data loc];      // leggi i dati
      execute(instr type, data);        //esegui l'istruzione
  } }
```


CPU vs Interpreti

- Un programma può essere eseguito sia da una CPU hardware che da un interprete software.
- Definito un linguaggio L si può realizzare
  - un processore hardware PL capace di eseguire i programmi scritti in L;
  - un interprete IL che interpreti i programmi in L e ne permetta l'esecuzione su una macchina hardware PIL che possa essere più semplice e meno costosa di PL.

L'evoluzione delle istruzioni

- Primi calcolatori ➠ **istruzioni semplici**.
- Per creare calcolatori più potenti sono state introdotte **istruzioni più complesse**
  - esecuzione **più rapida** dei programmi, ma **aumenta il tempo di ciclo** del data path;
  - esecuzione sovrapposta o in paralleo di singole operazioni utilizzando **hardware addizionale**;
  - inserite anche nei calcolatori a basso costo per
    - contrastare l'aumento nei costi di sviluppo sw;
    - soddisfare i requisiti di compatibilità.

Il concetto di "architettura"

- Nasce negli anni '50 quando IBM sviluppa una famiglia di macchine capaci di eseguire le **stesse istruzioni**.
- Implementazioni diverse si distinguono per il **costo** e per il livello di **prestazioni** (implementazione hardware solo sui modelli più costosi).
- I calcolatori più semplici hanno anche altri vantaggi:
  - correggere **on field** l'implementazione di alcune **istruzioni;**
  - correggere **on field** errori di progettazione **hardware**;
  - aggiungere **nuove istruzioni** dopo la consegna;
  - **progettazione strutturata** che permetteva sviluppo, test e documentazione efficiente di istruzioni complesse.

Sviluppo delle macchine microprogrammate

- Motivato dalla richiesta di calcolatori a **basso costo**:
  - realizzare calcolatori complessi mantenendo **semplice** la parte **hardware** (in linea con la tecnologia degli anni '70);
  - disponibilità di memorie read-only veloci (**control store**) dove memorizzare il microprogramma (**l'interprete**);
  - i modelli più costosi erano realizzati in hardware (e.g. Cray).
- Esempio tipico: **DEC–VAX**
  - **centinaia** di istruzioni (tra cui anche alcune di valore marginale, molto difficili da eseguire direttamente);
  - più di **200 modi di specificare gli operandi**.
- Evoluzione: colmare il **gap** tra **instruction set** e **linguaggio di programmazione**.

Le architetture RISC

- Sviluppo della **tecnologia** degli anni '80:
  - maggior **complessità** dei **circuiti integrati** (VLSI);
  - memoria centrale di **velocità** pari alla control store.
- Possibilità di realizzare CPU **cablate**
  - obiettivo: ottimizzare le **prestazioni complessive**;
  - fase 1: istruzioni semplici, eseguibili in **poco tempo**;
  - fase 2: istruzioni che possano essere **messe in esecuzione velocemente**;
  - risultato: CPU con un numero ristretto di istruzioni chiamate **RISC** (**Reduced Instruction Set Computer**) in contrasto con le **CISC** (**Complex ISC**) che dominavano il mercato;
  - esempi: SPARC, MIPS, …

RISC vs CISC

- RISC ha **prestazioni migliori**
  - istruzioni semplici, eseguite in un solo ciclo del data path;
  - 4/5 istruzioni RISC equivalgono a una istruzione CISC;
  - assenza di interpretazione dà un vantaggio di circa 10 volte.
- CISC ha garantito la **compatibilità** col passato
  - i clienti non hanno perso l'investimento SW;
  - modularità e flessibilità dell'evoluzione.
- CISC si è evoluto verso un **approccio misto**
  - processore combinato: una parte (core) RISC e una CISC;
  - approccio RISC per le istruzioni più semplici (e più comuni) eseguite in un ciclo del data path;
  - approccio CISC per le istruzioni più complesse.

Alcuni principi di progettazione

- Eseguire direttamente in **hardware** le istruzioni (soprattutto quelle **più comuni**).
- Ottimizzare la velocità di "lancio" delle istruzioni:
  - cercare di **far partire tante istruzioni** insieme;
  - sviluppare il **parallelismo a livello hardware**;
  - gestire l'esecuzione "**out of order**" delle istruzioni.
- **Semplificare la** fase di **decodifica**.
- Progettare architetture **load/store** in cui l'accesso alla memoria avviene solo tramite queste 2 istruzioni.
- Aumentare il **numero di registri** disponibili.

Parallelismo

- La **frequenza di clock**
  - influenza direttamente il tempo di ciclo del data path e quindi le prestazioni di un calcolatore;
  - è limitata dalla tecnologia disponibile.
- Il **parallelismo** permette di migliorare le prestazioni senza modificare la frequenza di clock. Esistono due forme di parallelismo:
  - **parallelismo a livello delle istruzioni** (architetture **pipeline** o architetture **superscalari**);
  - **parallelismo a livello di processori** (**Array computer**, **multiprocessori** o **multicomputer**).

Architettura pipeline

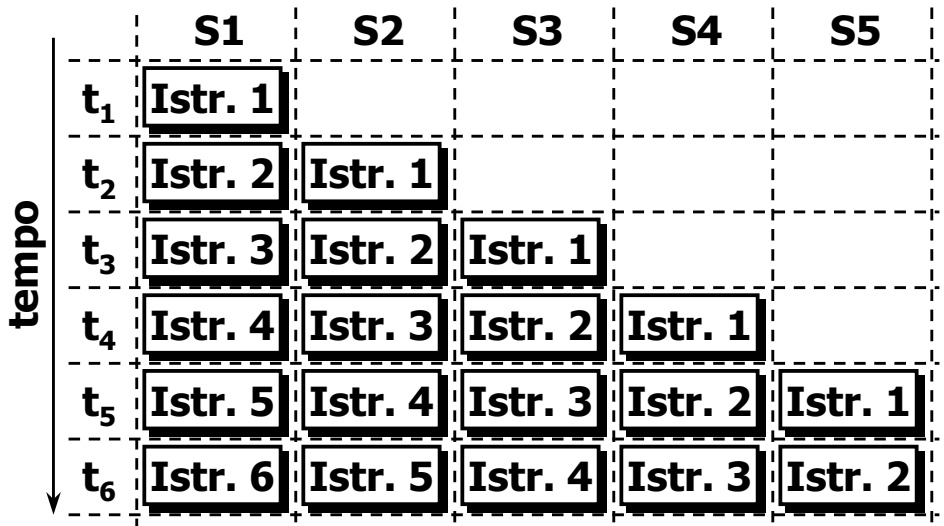
- Organizzazione della CPU come una "**catena di montaggio**"
  - la CPU viene suddivisa in "**stadi**", ognuno dedicato all'esecuzione di un compito specifico;
  - l'esecuzione di un'istruzione richiede il **passaggio attraverso** (tutti o quasi tutti) **gli stadi della pipeline**;
  - in un determinato istante, **ogni stadio esegue la parte di sua competenza di una istruzione**;
  - in un determinato istante, esistono **diverse istruzioni contemporaneamente in esecuzione**, una per ogni stadio.

Esempio di pipeline /1

Pipeline in **cinque stadi:**

S1. **lettura istruzioni** dalla memoria e loro caricamento in un apposito buffer;
S2. **decodifica** dell'istruzione per determinarne il tipo e gli operandi richiesti;
S3. individuare e **recuperare gli operandi** dai registri o dalla memoria;
S4. **esecuzione** dell'istruzione, tipicamente facendo passare gli operandi per il data path;
S5. **invio dei risultati** al registro appropriato.

Esempio di pipeline /2



Esempio di pipeline /3

	S1	S2	S3	S4	S5
t_i	i	i - 1	i - 2	i - 3	i - 4
t_{i+1}	i + 1	i	i - 1	i - 2	i - 3
t_{i+2}	i + 2	i + 1	i	i - 1	i - 2
t_{i+3}	i + 3	i + 2	i + 1	i	i - 1
t_{i+4}	i + 4	i + 3	i + 2	i + 1	i

All'istante t_{i+2} ci sono 5 istruzioni in esecuzione

Esempio di pipeline /4

	S1	S2	S3	S4	S5
t_i	i	i - 1	i - 2	i - 3	i - 4
t_{i+1}	i + 1	i	i - 1	i - 2	i - 3
t_{i+2}	i + 2	i + 1	i	i - 1	i - 2
t_{i+3}	i + 3	i + 2	i + 1	i	i - 1
t_{i+4}	i + 4	i + 3	i + 2	i + 1	i

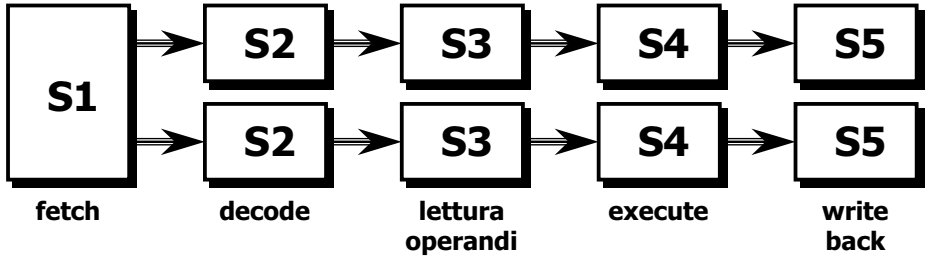
Lo stadio S3 esegue la parte di sua competenza di istruzioni successive l'una all'altra.

Prestazioni di una pipeline

- Il tempo di esecuzione (**latenza**) della singola istruzione non diminuisce, anzi **aumenta**
  - il tempo di attraversamento (latenza) della pipeline corrisponde al numero degli stadi (**N**) moltiplicato per il tempo di ciclo (**T**);
  - il tempo di ciclo è limitato dallo stadio più lento!
- **Aumenta** il numero di istruzioni completate nell'unità di tempo (**throughput**)
  - si completa **un'istruzione a ogni ciclo di clock**;
  - l'**incremento** di throughput è quasi **proporzionale al numero degli stadi!**

Architetture superscalari

- Vista la disponibilità di un maggior numero di transistor si inseriscono **più pipeline** nella stessa CPU
  - aumenta il parallelismo perché è possibile eseguire contemporaneamente diversi **flussi di istruzioni**;
  - è necessario garantire che non ci siano **conflitti** tra le istruzioni che vengono eseguite insieme; di solito il controllo è affidato al compilatore.



Architetture superscalari

In alternativa si possono **replicare le unità funzionali**

- rappresentano lo stadio più lento della pipeline (in genere richiedono diversi cicli di clock);
- è più semplice evitare i **conflitti** tra le diverse istruzioni.

Array computer

- Pensati per la soluzione di problemi che richiedono l'esecuzione delle **stesse operazioni su tanti dati**
  - **algoritmi** facilmente **parallelizzabili**, cioè possono essere suddivisi in modo efficiente tra diversi esecutori;
  - vettori o **matrici come operandi**.
- Due tipi di architetture
  - **array processor:** composti da tante unità funzionali uguali che eseguono le stesse istruzioni su dati diversi;
  - **vector processor:** le operazioni vengono eseguite da una sola unità funzionale, dotata di una pipeline profonda con tempo di ciclo molto breve. I dati stanno in **registri vettoriali** (un insieme di registri tradizionali che si possono leggere dalla memoria in una sola istruzione).

La memoria principale (RAM)

- **Bibliografia**:

  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - **paragrafo 2.2**

- **Sommario**:
  - Concetto di **bit**.
  - **Indirizzi** di memoria.
  - **Ordinamento dei byte** (little endian vs. big endian).
  - Codici di **correzione degli errori**.
  - Memoria **cache**.
  - Packaging e **tipi di memoria**.

Il concetto di bit

- Un bit può contenere uno 0 o un 1 e rappresenta **l'unità di base** della memoria.
  - Per memorizzare informazione bisogna avere un dispositivo che possa assumere **almeno due** stati distinti;
  - i dispositivi che hanno **solo due** stati sono più **affidabili** e **veloci** di quelli con tre o più stati;
- A volte si parla di dispositivi che usano l'**aritmetica decimale** intendendo con ciò la codifica binaria delle cifre decimali (**BCD - Binary Coded Decimal**)
  - con **quattro bit** (16 combinazioni) si codificano le cifre **da 0 a 9**, che poi vengono utilizzate per comporre un numero;
    - $1944_{dieci} = 0001\ 1001\ 0100\ 0100_{BCD} = 11110011000_{due}$;
    - 16 bit in formato BCD codificano 10000 numeri (0-10'000);
    - 16 bit in formato binario codificano 65536 numeri (2^{16}).

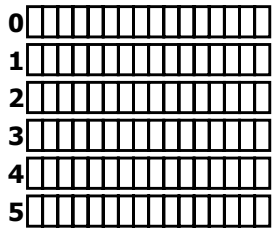
Indirizzi di memoria

- I bit nelle memorie sono raggruppati in **celle**:
  - tutte le celle sono formate dallo **stesso numero di bit**;
  - una cella composta da **k bit**, è in grado di contenere una qualunque tra **2^k combinazioni** diverse di bit.
- Ogni cella ha un **indirizzo**:
  - serve come accesso all'informazione;
  - in una memoria con **N celle** gli indirizzi vanno da **0** a **N–1**.
- **La cella è l'unità indirizzabile più piccola.** In quasi tutti i calcolatori è di **8 bit** (un **byte**).
- I byte vengono raggruppati in **parole** (che oggi sono di **32/64 bit**), su cui la CPU esegue le operazioni.

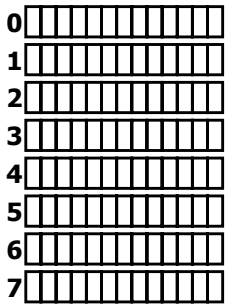
Organizzazione della memoria

- Anche gli indirizzi della memoria sono rappresentati come numeri binari:
  - un indirizzo di **M** bit consente di indirizzare **2^M** celle;
  - per 6 o 8 celle bastano 3 bit, per 12 celle ne servono 4;
  - il **numero di bit nell'indirizzo** determina il **numero massimo di celle indirizzabili** nella memoria ed è indipendente dal numero di bit per cella (una memoria con 2^{12} celle richiede sempre 12 bit di indirizzo, quale che sia la dimensione di una cella).
- Una memoria può essere organizzata in diversi modi:
  - con 96 bit possiamo avere 6 celle di 16 bit (6*16=96), o 8 celle di 12 bit (8*12=96) o 12 celle di 8 bit (12*8=96).

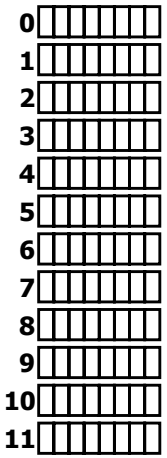
Organizzazione della memoria



6 parole da 16 bit



8 parole da 12 bit



12 parole da 8 bit

Ordinamento dei byte

- Una **parola** contiene **più byte** (e.g. 32 bit ➠ 4 byte)
- I byte di una parola possono essere enumerati da sinistra a destra o viceversa:
  - **big endian**, con i byte numerati da sinistra a destra (usato nelle CPU SPARC, Motorola e dei mainframe IBM);
  - **little endian**, con i byte numerati da destra a sinistra, (usato nelle CPU Intel).
- Il valore 6_{dieci} (memorizzato nella parola di indirizzo 0) è rappresentato da 29 bit a 0 seguiti dai bit 110
  - big endian: i bit 110 sono nel byte 3,
  - little endian: i bit 110 sono nel byte 0.

Big vs little endian /1

0	1	2	3
00000000	00000000	00000000	00000110

Big endian

0	0	1	2	3
4	4	5	6	7
8	8	9	10	11
12	12	13	14	15

0	3	2	1	0
4	7	6	5	4
8	11	10	9	8
12	15	14	13	12

Little endian

3	2	1	0
00000000	00000000	00000000	00000110

Big vs little endian /2

JIM SMITH, 21 anni, ufficio 260 (1*256+4)

Big endian

0	J	I	M	
4	S	M	I	T
8	H	0	0	0
12	0	0	0	21
16	0	0	1	4

Little endian

0		M	I	J
4	T	I	M	S
8	0	0	0	H
12	0	0	0	21
16	0	0	1	4

Big ➠ little

	M	I	J
T	I	M	S
0	0	0	H
21	0	0	0
4	1	0	0

Big ➠ little + inversione

J	I	M	
S	M	I	T
H	0	0	0
0	0	0	21
0	0	1	4

In entrambi i casi sorge un problema!!

Memoria vs. CPU

- Le **CPU** sono sempre state **più veloci** delle **memorie**
  - l'aumento di integrazione ha consentito di realizzare **CPU** pipeline e super scalari, molto efficienti e **veloci**;
  - nelle memorie è aumentata la **capacità** più che la **velocità**.
- L'**accesso** alla memoria passa **attraverso** il **bus**
  - la **frequenza** di funzionamento **del bus** è molto **più bassa** di quella della CPU;
  - il **bus** può essere **impegnato** ad effettuare trasferimenti controllati **da dispositivi** di I/O **"autonomi"** (e.g. DMA).
- **È difficile riordinare le istruzioni** in modo da poter sfruttare i tempi di attesa della memoria.
- È possibile fare **memorie molto veloci** se stanno nel chip della CPU, ma **sono piccole** e **costose**.

La memoria centrale

- Tecnologia elettronica (**veloce** ma **volatile**)
- Gerarchia di memoria:
  ai **livelli più alti** corrispondono le **tecnologie più veloci** ma anche **più costose**
  - cache interna (Static RAM – SRAM)
  - cache esterna (SRAM)
  - memoria RAM
    (Dynamic RAM – DRAM e sue varianti)
  - area di swap su memoria di massa

Memoria cache: SRAM

- Interna (cache di livello 1)
  - stessa frequenza della CPU
- Esterna (cache di livello 2)
  - frequenza inferiore a quella della CPU
- Esterna (cache di livello 3)
  - generalmente saldata sulla motherboard
  - sempre meno usata
  - TA dell'ordine dei nsec

Memoria centrale – DRAM

- Fast Page Mode DRAM (**FPM DRAM**)
  - TA=70-60ns
  - Per la lettura si attiva la riga, la colonna, si validano i dati, si trasferiscono i dati, poi si disattiva la colonna
  - I miglioramenti di velocità nascono dal progresso della tecnologia di integrazione.
- Extended Data Out DRAM (**EDO DRAM**)
  - TA = 70-50ns
  - Non richiede la disattivazione della colonna e del buffer di uscita.
- Synchronous DRAM (**SDRAM**)
  - Sfrutta la sequenzialità delle richieste: una volta trovato il primo dato gli altri vengono recuperati velocemente.
  - Fornisce dati fino a 10ns (100MHz)

La memoria secondaria (di massa)

- **Bibliografia**:

  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - **paragrafo 2.3**

- **Sommario**:
  - Gerarchie di memoria.
  - Dischi magnetici
  - Dischi ottici

Caratteristiche dei diversi livelli

	Capacità	Velocità (TA)	€/MByte
registri	**~1KB**	**~1ns**	**NA**
cache	**256 ÷ 1024 KB**	**~1ns**	**150-N/A**
RAM	**128 ÷ 2048 MB**	**~4ns**	**0.33**
HD	**40 ÷ 100 GB**	**~10ms**	**0.005**
nastri/CD	**~GB per unità**	**~100ms**	**0.005**

Una gerarchia di memoria

E' tipicamente costituita da:

- **registri** contenuti nella CPU (qualche KB)
- **cache** (da circa 32KB a circa 1024KB)
- **memoria principale** (da circa 64MB a qualche GB)
- **dischi fissi** (da qualche GB a qualche TB)
- **nastri magnetici e dischi ottici** (da qualche GB a qualche TB per ogni supporto)

Man mano che ci si sposta verso il basso nella gerarchia aumenta il valore dei parametri fondamentali:

- **aumenta il tempo di accesso**;
- **aumenta la capacità** di memorizzazione;
- ma **diminuisce il costo per bit**.

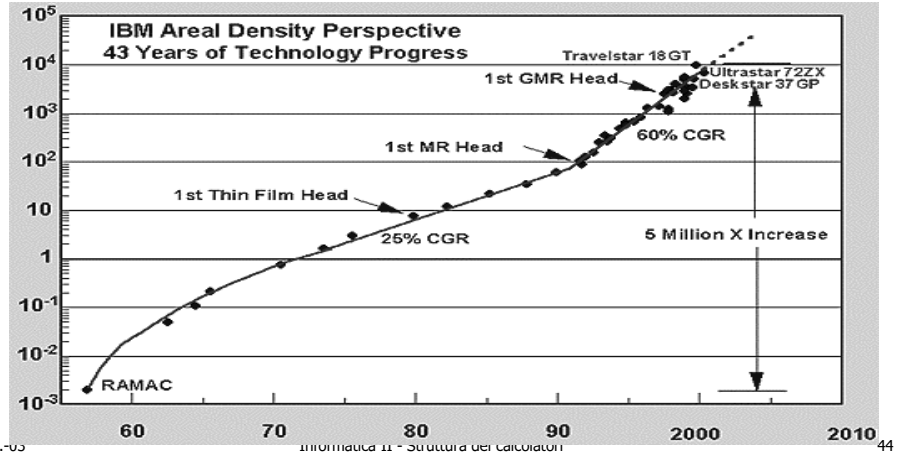
Prestazioni dei dischi

- **Tempo di acceso** (ms o 10^{-3}s)
  - **Seek time**
    - la testina deve arrivare alla traccia giusta;
    - dipende dalla meccanica (5-15 ms, 1 per tracce adiacenti).
  - **Latency**
    - il disco deve ruotare fino a portare il dato nella posizione giusta;
    - dipende dalla velocità di rotazione (5400-10800 RPM ➠ 2.7-5.4ms).
- **Transfer Rate** (MBps)
  - **Velocità di trasferimento del disco**
    - dipende dalla densità di registrazione e dalla velocità di rotazione;
    - un settore di 512 byte richiede fra 25 e 100 µsec (5-20 MB/sec).
  - **Velocità di trasferimento del sistema di controllo**
    - SCSI vs. EIDE

Floppy disk

- Funzioni:
  - **distribuzione** software su grande scala (avvento PC);
  - archiviazione dati.
- Struttura analoga a quella di un disco magnetico,
  - il disco si **ferma** quando non è operativo;
  - **l'avvio della rotazione** comporta un **ritardo** di **½ sec**.
- Caratteristiche tipiche di un floppy da 3.5"
  - Capacità di **1.44 MB**
  - Tracce x settori: **80 x 18**
  - RPM = **300**
  - velocità di trasferimento di **500Kbps**

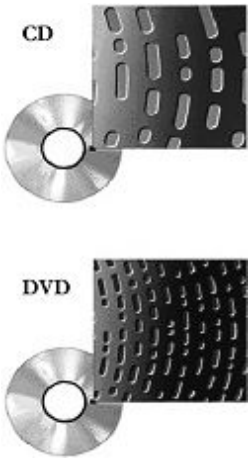
Andamento densità HD [by IBM]



Compact Disk - CD

- Proposto nel 1980 [da Philips e Sony] per sostituire i dischi in vinile per la musica.
- Standard internazionale IS-10149 [**libro rosso**].
  - diametro di **12 cm**, spessore di 1.2 mm con un foro di 15 mm in mezzo;
  - produzione:
    - laser ad alta potenza che brucia fori di 0,8 μm in un **disco master** (le depressioni si chiamano **pit** e le aree fra pit si chiamano **land**);
    - dal master si ricava uno **stampo**;
    - nello stampo viene iniettata una resina liquida di **policarbonato** che forma un CD con la stessa sequenza di fori del master,
    - sul policarbonato viene depositato uno strato molto sottile di **alluminio riflettente**,
    - copertura con uno strato **protettivo** e infine con **un'etichetta**.

Digital Versatile Disk (DVD)



- Evoluzione tecnologica ➡ maggior densità dei dati:
  - pit più piccoli (0.4 vs. 0.8 μm);
  - spirale più serrata (0.74 vs. 1.6 μm);
  - laser rosso (0.65 vs. 0.78 μm).
- Caratteristiche dei DVD
  - capacità di 4.7 GB
    - 133 minuti di video fullscreen MPEG-2 ad alta risoluzione (720 x 480) con colonna sonora in 8 lingue e sottotitoli in altre 32;
  - 1x indica 1.4 MB/sec (vs. 150 KB/sec).

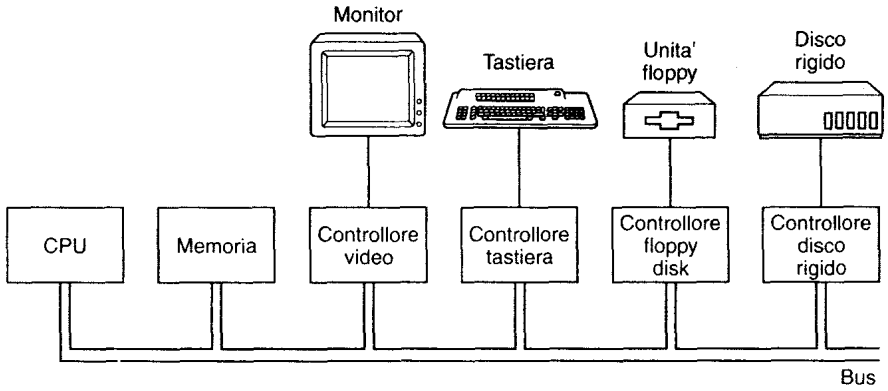
Dispositivi di Ingresso/Uscita (I/O)

- **Bibliografia**:

  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - **paragrafo 2.4**

- **Sommario**:
  - Bus
  - Terminali
  - Mouse
  - Stampanti
  - Modem

Struttura logica di un PC



Struttura fisica di un PC

- Nella scatola (case) sono contenuti:
  - una **scheda madre** che contiene una CPU, alcuni connettori nei quali inserire moduli DIMM e vari chip di supporto;
  - uno o due **bus**, uno ad alta velocità (per schede moderne) e uno a bassa velocità (per schede più vecchie);
  - prese in cui si possono inserire i connettori delle **schede di I/O** che agiscono da **controllori** dei dispositivi di I/O, cioè ne **gestiscono l'accesso al bus**:
    - un controllore che legge o scrive dati verso e da una memoria senza interventi da parte della CPU effettua un accesso diretto alla memoria (**Direct Memory Access – DMA**)
    - completato il trasferimento, il controllore **effettua un interrupt**, la CPU sospende il programma in corso e inizia una procedura speciale, (**interrupt handler**); quando l'interrupt handler termina, la CPU continua con il programma.

Terminali

- Composti di due parti: **tastiera** e **schermo**.
  - Nel mondo dei mainframe, sono integrati in un dispositivo singolo e collegati al calcolatore principale per mezzo di una linea seriale
  - Nel settore dei personal computer, sono dispositivi separati.
- **Tastiere**
  - molti tipi diversi, meccaniche o elettromagnetiche;
  - quando si preme un tasto viene generato un interrupt e viene avviato il gestore degli interrupt della tastiera, che legge un registro hardware all'interno del controllore della tastiera per avere il numero del tasto (da 1 a 102) premuto;
  - quando il tasto viene rilasciato si verifica un secondo interrupt.

Terminali grafici

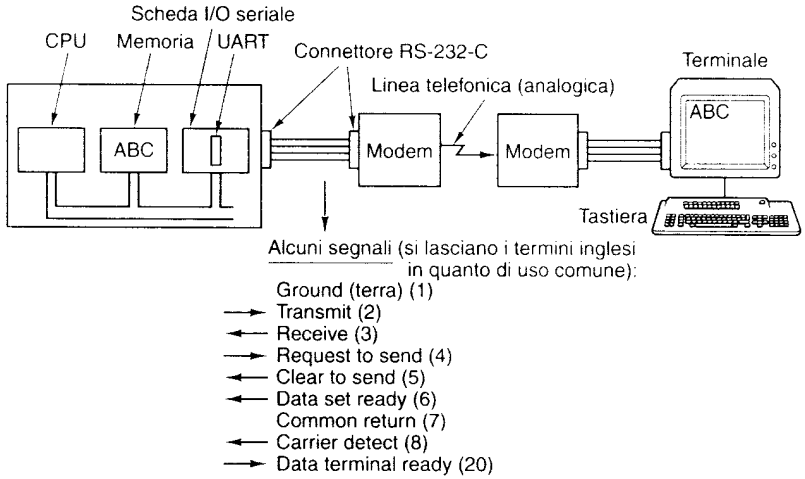
- Visualizzazione "**bit map**": lo schermo è una matrice di **pixel indipendenti**
  - per indicare il **colore** di ogni pixel si usano fino a **32 bit** (8 bit per ogni colore fondamentale + 8 bit per la trasparenza);
  - per rappresentare un carattere si usa un rettangolo di pixel e si configurano i bit necessari per visualizzare il carattere (così si possono realizzare diversi **font**);
  - comodi per i **sistemi operativi** a **finestre**;
  - richiedono una **memoria video** di grandi dimensioni
    - VGA: 640 x 480 x 4 byte = 1.2 Mbyte
    - SVGA: 800 x 600 x 4 byte = 1.9 Mbyte
    - XGA: 1024 x 768 x 4 byte = 3.2 Mbyte
    - UXGA: 1600 x 1200 x 4 byte = 7.5 Mbyte

  riducibili grazie all'utilizzo di una "**palette**" (scelta di $\mathbf{2^8}$=**256** colori tra i $\mathbf{2^{32}}$ possibili).

Terminale RS-232-C

- RS-232-C: standard per collegare qualsiasi (o quasi) terminale a qualsiasi (o quasi) calcolatore.
  - connettore standard di **25 pin** (non sempre utilizzati tutti);
  - Possibilità di sfruttare la linea telefonica via **modem** (**modulatore-demodulatore**);
  - calcolatore e terminale sono dotati di un chip chiamato **UART (Universal Asynchronous Receiver Transmitter)** e di logica per accedere al bus;
  - l'UART agisce da **convertitore parallelo-seriale**
    - riceve un carattere intero (1 byte = 8 bit);
    - fa uscire i bit uno per volta ad una velocità prefissata;
    - aggiunge un bit di inizio (start bit) e un bit di fine (stop bit).

Struttura terminale RS-232



Porte Standard

- **Interfaccia Seriale**
  - Trasporta un bit per volta.
  - Velocità massima di 115 kbps
  - Utilizzata per periferiche lente, come mouse e modem esterni
- **Interfaccia parallela**
  - Trasporta 8 bit alla volta.
  - Velocità di 150 KB/sec (2MB/s in modalità EPP)
  - Usata per stampanti, scanner e unità di backup (nastri, Zip).
- Direzione della comunicazione
  - **Simplex**: la linea trasmette solo in una direzione;
  - **Half-duplex**: la linea trasmette in entrambe le direzioni ma non contemporaneamente (una direzione per volta);
  - **Full-duplex**: la linea trasmette contemporaneamente in entrambe le direzioni.

Modem

- Connessione di calcolatori attraverso la rete telefonica (**analogica**).
- Velocità crescenti dal 1980 in poi
  - V.22bis, V.32 & V.32bis furono i primi standard per velocità di 2.4, 9.6 e 14.4Kbit/s.
  - V.34 (1994) supporta 28.8Kbit/s e corrisponde al minimo livello attualmente accettato
  - V.34+ (1996) arriva a 33.6Kbit/s
  - V.90 arriva a 56Kbit/s downstream e a 33.6Kbit/s upstream.
    - downstream indica dal digitale all'analogico
    - upstream indica dall'analogico al digitale

Integrated Services Digital Network - ISDN

- Linea analogica sostituita da **linea digitale**
  - in realtà non viene sostituita la linea, ma solo le **attrezzature alle due estremità**.
  - **Uso domestico**: **due canali** digitali indipendenti, ognuno da 64'000 bit/sec, e un canale di segnalazione da 16'000 bit/sec (per un totale di **144'000 bps**)
  - **Uso commerciale**: 30 canali per uso commerciale.
- Caratteristiche
  - tempo di **setup** della connessione praticamente nullo (1 s);
  - non serve più un modem analogico (**connessione digitale-digitale**);
  - è molto più **affidabile** (meno errori) di una linea analogica.