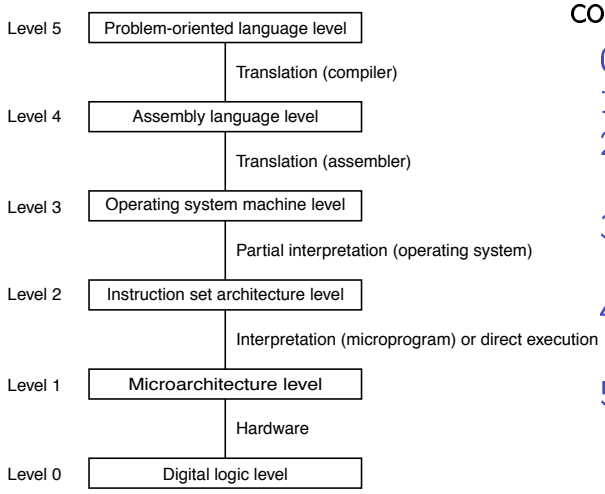
Un Computer a Sei Livelli



- Gli attuali computer sono costituiti da sei livelli:
  0. Livello logico digitale;
  1. Livello di microarchitettura;
  2. Livello di architettura dell'insieme di istruzioni (ISA)
  3. Livello macchina del sistema operativo;
  4. Livello del linguaggio assemblativo;
  5. Livello del linguaggio orientato al tipo di problema.

Livello Logico Digitale

- Il Livello 0 rappresenta l'hardware della macchina:
  - i suoi circuiti eseguono i programmi scritti nel linguaggio macchina del livello 1.
  - in questo corso non ci interessiamo del livello dei dispositivi analogici.
- Gli oggetti in questione sono le **porte**.
- Ogni porta:
  - è costituita da pochi transistor;
  - è dotata di uno o più input digitali (segnali 0 o 1)
  - calcola in output una semplice funzione dei valori in input:
    - $X = Y\ AND\ Z, X = Y\ OR\ Z$
- Un piccolo numero di porte possono essere combinate per formare una memoria a 1 bit.
- Combinando le memorie in gruppi di 16, 32 o 64 unità si possono comporre i cosiddetti **registri**.

Livello di Microarchitettura

- Nel livello 1, vi è
  - una **memoria locale**, formata da un gruppo di registri;
  - un circuito, chiamato **ALU** (*Arithmetic Logic Unit*), capace di effettuare semplici operazioni aritmetiche.
- I registri sono connessi alla ALU tramite un **percorso dati.**
- Le operazioni base del percorso dati sono:
  - selezionare uno o due registri;
  - permettere alla ALU di operare su di loro (i.e. sommandoli);
  - memorizzare il risultato in uno dei registri.
- Il percorso dati può essere controllato da:
  - un programma che l'utente può modificare, detto **microprogramma**;
  - circuiti (hardware) che non sono modificabili dall'utente (ma anche in questo caso parleremo di microprogramma).
- Attualmente, esso è controllato da circuiti.

Livello ISA e Livello del Sistema Operativo

- **ISA** significa *Instruction Set Architecture Level*.
- Il Livello ISA è il primo livello pubblico.
  - Il produttore di un computer fornisce le istruzioni che il computer è in grado di eseguire.
  - Queste istruzioni formano il livello ISA.
- Dentro il computer le istruzioni del Livello ISA sono eseguite dal microprogramma.
- Il Livello del Sistema Operativo è un'estensione del Livello ISA, ottenuta aggiungendo alcuni nuovi servizi.
- Si tratta di un livello ibrido:
  - Moltissime istruzioni di questo livello fanno parte anche del livello ISA.
  - Inoltre, il livello del Sistema Operativo contiene nuove istruzioni + servizi (e.g. la capacità di eseguire 2 programmi allo stesso tempo, la gestione della memoria)

Dal Livello 3 In Avanti

- Tra i livelli 3 e 4 vi sono differenze fondamentali.
  I livelli inferiori:
  - non sono utilizzati dal programmatore medio;
  - sono progettati per eseguire interpreti e traduttori per i livelli più alti;
  - sono sempre interpretati;
  - i loro programmi sono scritti dai **programmatori di sistema**;
  - i loro linguaggi sono numerici.

  I livelli superiori:
  - sono pensati per i programmatori che devono risolvere **problemi applicativi**;
  - sono generalmente supportati mediante traduzione;
  - i loro linguaggi contengono sequenze di caratteri alfanumerici per essere comprensibili dall'uomo.

Livello del Linguaggio Assemblativo e Livello del Linguaggio Orientato al tipo di Problema

- Il Livello 4 fornisce ai programmatori un modo per scrivere programmi per i livelli 1, 2 e 3 in una forma meno difficoltosa.
- Il programma che traduce i programmi in linguaggio assemblativo in programmi del livello ISA è detto **assemblatore**.
- Il Livello 5 consiste di linguaggi, chiamati **linguaggi ad alto livello**, definiti per essere utilizzati dai programmatori di applicazioni:
  - C++, Java, Prolog, ...
- Il programma che traduce i programmi in linguaggio ad alto livello in programmi del linguaggio assemblativo è detto **compilatore**.

Hardware e Software

- I programmi scritti nel linguaggio macchina (e.g. livello 1) possono essere eseguiti direttamente dai circuiti del computer (livello 0).
- Questi circuiti, insieme alla memoria e ai dispositivi input/oputput, formano l'**hardware** del computer.
- Quindi, l'hardware consiste di oggetti tangibili (circuiti integrati, schede, cavi etc).
- Il **software** invece consiste di idee astratte:
  - **algoritmi**: istruzioni dettagliate su come realizzare un determinato compito;
  - **programmi**: rappresentazioni degli algoritmi per i computer.

Evoluzione delle Macchine Multilivello

Nei primi anni

- Il confine tra hardware e software era chiaro.
- La maggior parte dei computer era di tipo "fai da te":
  - i programmatori erano costretti a imparare come far funzionare la macchina.
- I primi computer digitali degli anni '40 avevano solo due livelli:
  - il livello ISA, in cui erano realizzati tutti i programmi;
  - il livello logico digitale, che li eseguiva.
- Per questo motivo, i circuiti del livello logico digitale erano complessi, difficili da capire e da realizzare, e inaffidabili.
- Quindi fu introdotto un terzo livello, il **Livello della Microarchitettura.**
- Questo livello era realizzato da un programma (il **microprogramma**), che doveva eseguire programmi:
  - al livello ISA
  - mediante **interpretazione**.

Invenzione della Microprogrammazione

- Qual era il vantaggio?

⇒ Si riduceva il numero di circuiti elettronici nel computer.

- Perché?

⇒ L'hardware doveva eseguire microprogrammi costituiti da un insieme limitato di istruzioni,

⇒ invece i programmi a livello ISA usano un insieme di istruzioni più grande.

  ⇒ La complessità dell'hardware in questo modo diminuisce. (Aumenta la complessità del software).

- Per questo motivo, negli anni '70 tutti i principali computer implementavano il livello di microarchitettura.

Migrazione delle Funzionalità verso il Microcodice

- A partire dagli anni '70, i progettisti capirono che si posssono aggiungere nuove istruzioni al livello ISA, semplicemente estendendo il microprogramma.
- In altre parole, si può aggiungere nuovo "hardware" (nuove istruzioni macchina) per mezzo della programmazione.
- Questa scoperta portò a una "esplosione" dell'insieme delle istruzioni macchina.
- Le nuove istruzioni potevano essere ottenute combinando istruzioni esistenti comunque.
- Non era in realtà necessario aggiungere nuove istruzioni.
- Però in molti casi erano utili, perché permettevano di rendere alcune attività particolari di uso frequente più veloci e.g.:
  - Istruzioni per la moltiplicazione e divisione tra interi.
  - Istruzioni aritmetiche di vario tipo.
  - Istruzioni per chiamare procedure.

Eliminazione della Microprogrammazione

- Per avere sempre più istruzioni, servono microprogrammi sempre più grandi.
- Quali sono le conseguenze?

⇒ Microprogrammi di grandi dimensioni sono **meno efficienti** (più lenti) di microprogrammi più piccoli.

- Una soluzione a questo problema è:
  - eliminare il microprogramma,
  - ridurre l'insieme di istruzioni,
  - eseguire le istruzioni complesse direttamente (mediante un controllo hardware del percorso dati).
- Questa è la soluzione adottata oggi dai moderni computer.
- La microprogrammazione è scomparsa?

⇒ È difficile rispondere a questa domanda...

  - ...poiché la differenza tra harware e software è arbitraria!

Evoluzione delle Macchine Multilivello

Nel corso del tempo

- La distinzione tra hardware e software è diventata meno importante:
  - Ogni operazione eseguita via software può anche essere costruita in hardware;
  - Ogni istruzione eseguita dall'hardware può essere simulata via software.
  - A volte è difficile scegliere se implementare una funzione in hardware o in software!
- Le operazioni ripetitive necessarie a far funzionare un calcolatore sono state automatizzate e ciò ha portato alla creazione del **sistema operativo**.

Cosa è il Sistema Operativo?

- Il Livello del Sistema Operativo è un'estensione del Livello ISA, ottenuta aggiungendo nuovi servizi.
- Questi servizi vengono eseguiti (interpretati) da un programma del livello ISA, detto **sistema operativo**.
- Alcuni esempi di servizi sono:
  - Organizzazione e gestione della memoria.
  - Protezione dei dati e gestione della privacy.
  - Capacità di eseguire programmi in modo concorrente.
  - Capacità di gestire più utenti in modo concorrente.
- Perché è stato inventato il sistema operativo?

Invenzione del Sistema Operativo

...Quando non c'era il sistema operativo...

- I computer funzionavano con delle schede perforate (*punch card*), che contenevano
  - i programmi
  - i dati di ingresso
  - i dati prodotti dal computer
- Per "caricare" un programma quindi bisognava inserire nel computer le punch card che contenevano il programma.
- Il computer produceva il risultato perforando delle punch card.

Invenzione del Sistema Operativo

Punch card con una istruzione Fortran: $Z(1) = Y + W(1)$

- Per eseguire un programma su un computer bisognava seguire una procedura difficile:
  1. Caricare un programma, chiamato "compilatore;"
  2. Caricare il programma che si vuole eseguire;
     - In caso di errore, caricare di nuovo il compilatore;
     - In caso di successo, il risultato era una serie di punch card, che contenevano il programma tradotto in linguaggio macchina;
  3. Inserire le punch card perforate.

Invenzione del Sistema Operativo

- Quali sono gli svantaggi di questa procedura?
  - Il computer rimane inattivo in molte occasioni (e.g., nell'attesa di complilare il programma).
  - Molte operazioni ripetitive richiedono l'intervento umano.
- All'inizio degli anni '60, si cercò di ridurre il tempo sprecato.
- Come?
  - Usando un programma, chiamato **sistema operativo**, che era tenuto sempre nel computer.
  - Il computer veniva attivato/fermato mediante schede di controllo.
  - Le schede contenevano dei "job:" istruzioni per la lettura/caricamento di altri programmi.
  - Queste istruzioni possono essere pensate come istruzioni per una macchina virtuale.
  - Negli anni successivi, i sistemi operativi divennero sempre più sofisticati.
  - Più tardi vennero inventati i sistemi a divisione di tempo.

Pietre Miliari nell'Architettura dei Computer

Per comprendere lo stato attuale dei calcolatori, parliamo brevemente degli sviluppi storici.

Calcolatori Meccanici e Elettromeccanici (1642-1945)

- I primi calcolatori erano interamente meccanici o elettromeccanici.
- *Pascal* e *Leibniz* costrirono dispositivi per effettuare solo le **operazioni aritmetiche** (alla fine del '600).
- All'inizio dell'800, *Babbage* inventò:
  - **Difference Engine**
  - **Analytical Engine**

Babbage

- **Difference Engine** era un dispositivo per eseguire semplici operazioni (somma, sottrazione).
  - implementava **un solo algoritmo**: il metodo matematico delle differenze finite per il calcolo delle funzioni polinomiali, e.g., $f(x) = 2x^2 - 3x + 2$.

Modello della Difference Engine, ricostruito dal progetto di Babbage (Science Museum, Londra)

Babbage

- La Difference Engine contiene N colonne e può eseguire **solo una operazione**: la **somma** tra due valori **in due colonne vicine.**
- Ciascuna colonna può contenere **un valore decimale.**
- Il metodo usa la somma in modo iterativo.
- La funzione è calcolata senza dover usare la moltiplicazione.

Colonne della Difference Engine

Babbage

- **Analytical Engine** era composta di 4 componenti:

  - una memoria ("magazzino"), composta di 1000 parole di 50 cifre, utilizzata per memorizzare dati e risultati;
  - un'unità computazionale ("mulino"), che poteva prendere gli operandi dal magazzino per eseguire addizioni, sottrazioni, moltiplicazioni e divisioni e poi memorizzare il risultato nel magazzino;
  - un dispositivo di input (schede perforate)
  - un dispositivo di output (output stampato e perforato)
- Qual è la grande innovazione di questa macchina?

Analytical engine (Babbage)

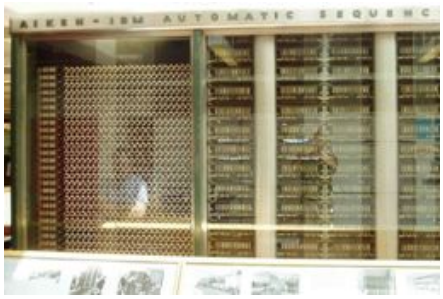
- Al contrario della difference engine, la analytical engine era **di uso generale.** Aveva:
  - dispositivi di input-output per leggere le istruzioni dalle schede perforate,
  - istruzioni per trasferire dati dal magazzino al mulino e viceversa,
  - istruzioni per analizzare un numero e differenziare l'esecuzione in base al segno.
- Quindi, perforando un diverso programma dalle schede di input era possibile far eseguire alla macchina diverse computazioni.

Calcolatori Meccanici e Elettromeccanici (1642-1945)

- Le macchine seguenti usavano i relé:
  - **Z1** da *Zuse*, fu la prima macchina calcolatrice basata su relé elettromagnetici (al fine degli anni'30).
  - **ABC** da *Atanasoff*, era basata sull'aritmetica binaria (1939).
  - **Complex Number Calculator** da *Stibitz*, era in grado di calcolare nel dominio dei numeri complessi (1940).
  - **Mark I** da *Aiken*, che migliorò analytical engine con relé elettromagetici (1944).

Zuse

Atanasoff-Berry Computer

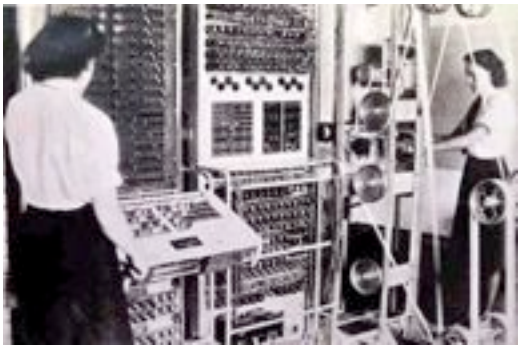


Mark-I, lato sinistro

Mark-I, unità di Input/Output

Valvole (1945-1955)

- Nel 1945, cominciò l'era elettronica con l'uso delle valvole. Alcuni esempi..
- **COLOSSUS** (1943):

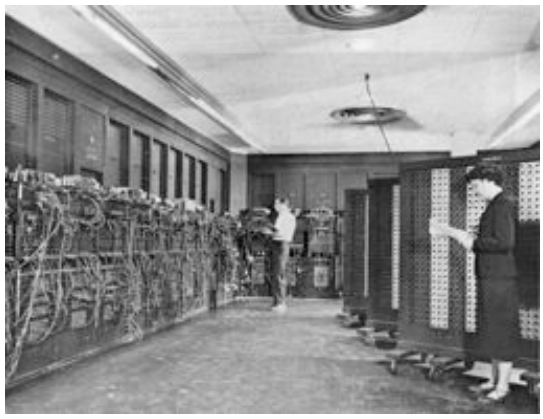


Colossus al Bletchley Park Museum a Londra http://www.bletchleypark.org.uk/

  - Aveva scopi militari.
  - Era usato dai servizi segreti inglesi per decifrare i messaggi in codice dei tedeschi durante la seconda guerra mondiale.
  - Restò segreto per decenni.
  - *Alan Turing* partecipò al suo progetto.
  - Fu il primo dispositivo di calcolo elettronico digitale programmabile.

Valvole (1945-1955)

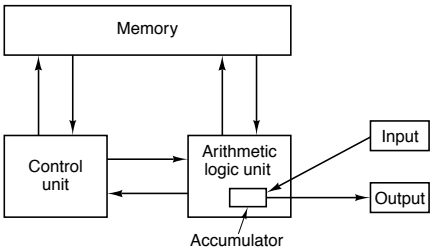
- **ENIAC** (University of Pennsylvania, 1946):
  - Era dotata di 20 registri da 10 cifre **decimali**.
  - Si programmava regolando 6000 **interruttori** multi-posizione.
  - In questo senso il programma è "cablato" nella macchina, cioè per programmare la macchina si modifica l'hardware.
  - *John Von Neumann* lavorò al suo progetto.



IAS (Von Neumann)

- Dopo l'esperienza con l'ENIAC, negli anni '50 Von Neumann andò a Princeton per costruire un nuovo computer IAS.
- Von Neumann partì da alcune osservazioni importanti:
  - La programmazione di un computer mediante un gran numero di interruttori e cavi era lenta e poco flessibile.
  - Anche i programmi, così come i dati, possono essere rappresentati in forma numerica all'interno della memoria del computer.
  - L'aritmetica binaria era sufficiente per sostituire la complicata aritmetica decimale dei computer esistenti.
- Con queste idee, Von Neumann progettò un'architettura di computer molto famosa: la **macchina di Von Neumann**.

IAS (Von Neumann)



- La macchina di Von Neumann era composta di cinque componenti principali:
  - la memoria;
  - l'unità aritmetico-logica;
  - l'unità di controllo;
  - il dispositivo di input/output;
- La memoria poteva conservare dati e istruzioni, eliminando il bisogno di programmare con gli interruttori.
- L'unità aritmetico-logica+l'unità di controllo:
  - formavano insieme il cervello del computer, eliminando il bisogno dell'aritmetica decimale.
  - Oggi sono combinate in un solo chip (**CPU**) negli calcolatori moderni.

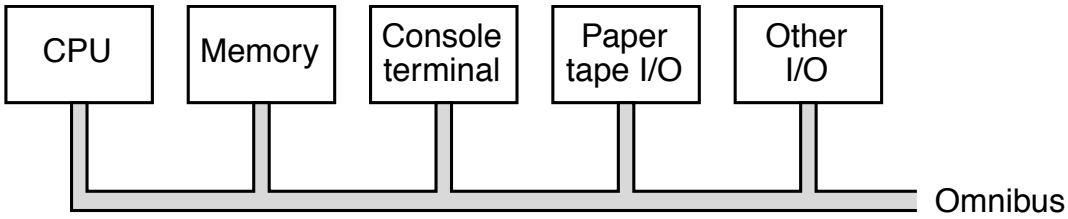
Fasi Successive

- Nelle fasi successive, il transistor rivoluzionò il computer.
- **Transistor** (1955-1965)
  - Permettono di rappresentare l'informazione in meno spazio, consumando meno energia e processando l'informazione più velocemente, rispetto alle valvole.
  - Portano alle memorie nell'ordine delle migliaia di caratteri, migliaia di istruzioni in un secondo.
  - Esempi: **PDP-1**, **PDP-8**, **IBM 7094**, **Cray**, ...

DEC PDP-8

IBM 7094

PDP-8



- Notiamo i blocchi di Memoria e I/O della macchina di Von Neumann, e una CPU, che contiene ALU e unità di controllo.
- La principale innovazione di PDP-8 fu avere **un unico bus**, chiamato "omnibus".
- Il bus è un insieme di cavi paralleli, utilizzati per connettere i diversi componenti di un computer.
- Questa architettura fu un grande passo avanti rispetto a IAS, che era centralizzata rispetto alla memoria.
- Fu poi adottata in quasi tutti i computer di piccole dimensioni.

Fasi Successive

- I computer che abbiamo visto finora sono appartenuti a diverse "generazioni," ciascuna caratterizzata da una diversa tecnologia:
  - della generazione "zero" (computer meccanici) abbiamo visto l'analytical engine;
  - della "prima generazione" (valvole) abbiamo visto l'ISA;
  - della "seconda generazione" (transistor) abbiamo visto il PDP-8.
- La "terza generazione" è quella dei **Circuiti Integrati** (1965-1980)
  - Permetteno di inserire decine di transistor su un unico chip.
  - Di conseguenza, le macchine sono più piccole, veloci ed economiche.
  - Esempi: **IBM System/360**, **PDP-11**, ...

System/360

- Faceva parte di una **famiglia** di macchine:
  - le macchine erano tutte dotate dello **stesso linguaggio assemblativo**,
  - però avevano dimensione e potenza diverse.
  - Fu possibile per la prima volta cambiare computer, senza dover cambiare il software!
- Supportava la **multiprogrammazione**. In questo modo:
  - è possibile avere **più programmi in memoria allo stesso tempo**;
  - quando si è in attesa di completare un'operazione di input/output si possono eseguire dei calcoli.
  - Quindi si ottiene un maggior utilizzo della CPU.

System/360 e PDP-11

...esistevano ancora le punch card in questa era ...

Macchina IBM System 360. Sulla destra due lettori di punch card multi-funzione

Punch-tape per la Macchina PDP-11 della Digital Equipment Corp.

Integrazione a Larghissima Scala (VLSI) (1980-?)

- Dal 1980 in poi si sviluppa la "quarta generazione", basata sull'integrazione a grandissima scala (*Very Large Scale Integration*)
- Permette di inserire milioni di transistor su un unico chip.
- La tecnologia VLSI consentì di vendere i computer a prezzi molto più bassi rispetto a prima.
  - Il computer diventò un oggetto accessibile all'individuo medio.
  - Iniziò l'era del Personal Computer.
- Alcune novità di questa generazione sono:
  - Le interfacce grafiche, o GUI (*Graphical User Interface*), introdotte nel 1984 con il Macintosh.
  - I computer portatili da Compaq.

Tipologie di Computer

Dopo la storia dei calcolatori, guardiamo al presente e al futuro.

Legge di Moore

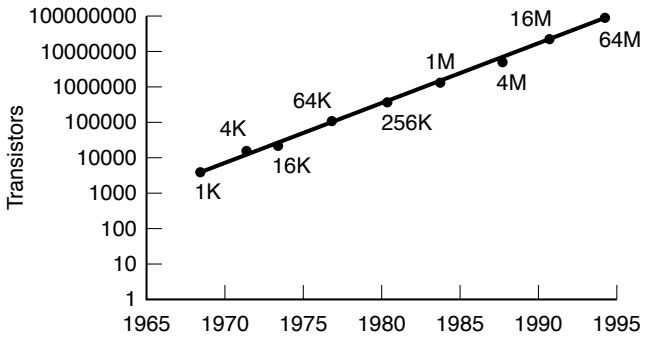
- Un'osservazione empirica da Gordon Moore (cofondatore e ex presidente di Intel).
- Dato che:
  - ogni 3 anni è introdotta una nuova generazione di chip;
  - il nuovo chip ha una quantità di memoria 4 volte maggiore della precedente;

→ il numero di transistor in un chip aumenta a una velocità costante;

→ questa crescita continuerà allo stesso modo per decenni.

- Praticamente, la legge prevede un aumento annuale del 60% del numero di transistor che può essere inserito su un chip.

Legge di Moore

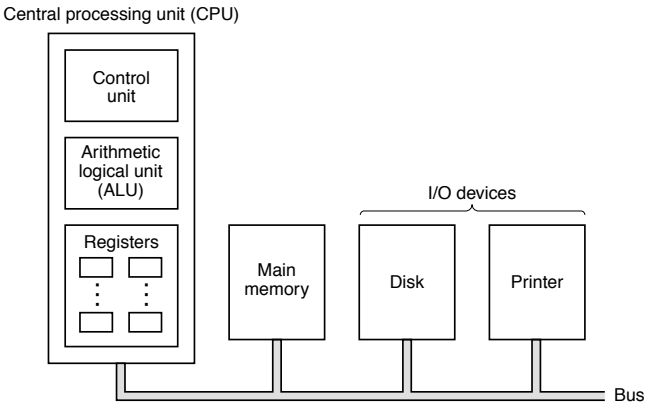


- La figura rappresenta un grafico semilogaritmico.
  - Una linea retta rappresenta una crescita esponenziale.
  - È una crescita molto rapida!
    - Nel 1974 si avevano 16mila transistors su chip
    - Dal 1974 al 1977: 3 anni $\Rightarrow 4\times$ 16mila
    - Nel 1977 si avevano 64mila transistors su chip
    - etc.

Legge di Moore

- Moore ha formulato la sua "legge" nel 1965.
- Questa legge è rimasta valida nella realtà per 40 anni.
- Ci si aspetta che continui a valere per almeno altri 10 anni.
- Esistono però dei problemi fisici all'aumento del numero di transistor su chip. E.g.,
  - Dissipazione di energia:
    - Molti transistor producono molto calore che deve essere dissipato.
    - Una superficie piccola fa fatica a dissipare calore.
    - Troppo calore può danneggiare il chip.
      (ecco perché sopra una piccola CPU c'è una grande ventola)
  - Dispersione di corrente:
    - Molti transistor su un chip devono essere molto vicini.
    - La corrente può attraversare lo spazio tra i transistor.
    - Questo aumenta il consumo di potenza e può danneggiare il chip.
- Quindi, per continuare con la legge di Moore, la tecnologia deve risolvere questi problemi.

Processori

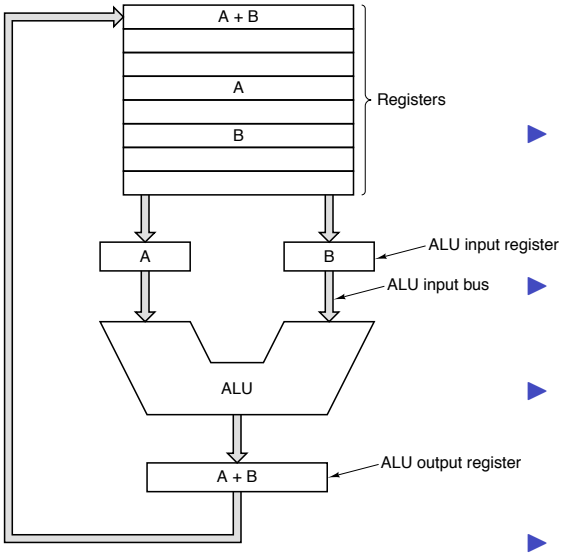


- L'organizzazione di un calcolatore.
- I componenti sono connessi mediante un **bus**.
- La **memoria principale** contiene i programmi da eseguire.
- La **CPU** ha il compito di eseguire i programmi.
  - Cominciamo con l'organizzazione della CPU...

Organizzazione della CPU

- La **CPU** (*Central Processing Unit*) è il cervello del calcolatore.
- La sua funzione è:
  - prelevare i programmi dalla memoria principale;
  - eseguire le relative istruzioni, una alla volta.
- La CPU è composta da tre parti distinte:
  - L'**Unità di Controllo** (**UC**), preleva le istruzioni dalla memoria, e determina il loro tipo.
  - L'**Unità Aritmetico Logica**, (**ALU**), esegue le operazioni necessarie per portare a termine le istruzioni.
  - I **registri**, memorie ad alta velocità, tengono traccia di risultati e informazioni di controllo.
- Due registri svolgono ruoli particolarmente importanti:
  - Il registro **PC** (**program counter**), che contiene l'indirizzo della prossima istruzione da eseguire.
  - Il registro **IR** (**instruction register**), che contiene l'istruzione che si sta eseguendo.

Percorso Dati di una Tipica Macchina di von Neumann



- Il **data path** è il percorso che i dati seguono all'interno di una CPU di von Neumann.
- È composto dai **registri**, dalla **ALU**, e da alcuni **bus** (interni alla CPU).
- Tra i registri notiamo quelli di ingresso alla ALU (**A** e **B**).
- L'ALU esegue semplici operazioni (e.g. la somma tra A e B).
- Il risultato viene memorizzato in un registro di output.

Categorie di Istruzioni

- La maggior parte delle istruzioni è divisa in due categorie:
  - Le istruzioni registro-memoria.
  - Le istruzioni registro-registro.
- Le **istruzioni registro-memoria** permettono:
  - di prelevare parole dalla memoria per portarle all'interno dei registri;
  - oppure di copiare i valori dei registri nella memoria.
- Invece, una tipica istruzione **registro-registro**:
  - preleva due operandi dai registri;
  - li porta all'interno dei registri di input della ALU;
  - esegue su di loro qualche operazione (e.g., l'addizione);
  - e infine ne memorizza il risultato in uno dei registri.

Esecuzione dell'Istruzione (il Ciclo PDE)

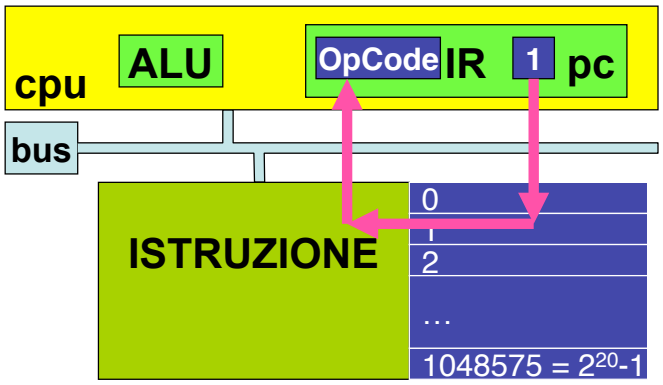
- La CPU esegue ogni istruzione ripetendo una serie di passi:
  1. Preleva l'istruzione cui punta il PC e la copia nell'IR.
  2. Modifica il PC per farlo puntare all'istruzione seguente.
  3. Determina il tipo dell'istruzione che si trova nell'IR.
  4. Se si tratta di un'istruzione che usa una parola di memoria, determina dove si trova tale parola.
  5. Se necessario, preleva la parola per portarla in un registro della CPU.
  6. Esegue l'istruzione.
  7. Torna al passo 1.
- Questo ciclo è detto **prelievo-decodifica-esecuzione** (PDE) oppure **fetch-decode-execute**.
- Ma **chi** esegue questo ciclo? La risposta la conosciamo già: l'unità di controllo!
  - Vediamo come è realizzata l'unità di controllo...

2

IL CICLO fetch / decode / execute

FETCH

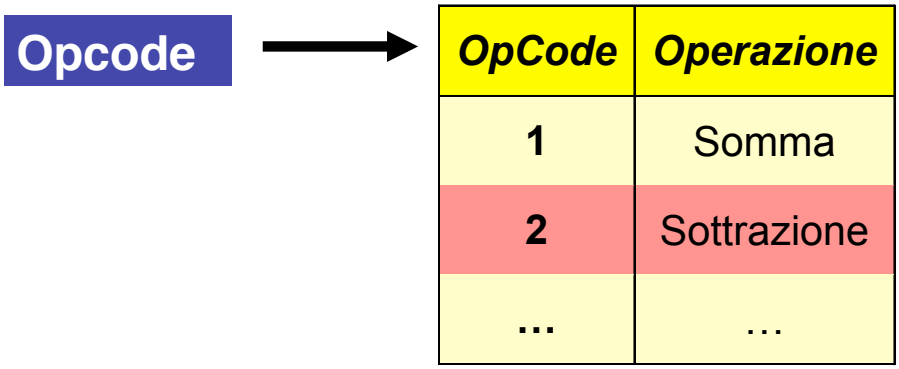
- si accede alla **prossima istruzione** (cella il cui indirizzo è contenuto nel registro **PC**) ...
- … e **la si porta dalla memoria centrale**, memorizzandola nel ***Registro Istruzioni*** **(IR)**



IL CICLO fetch / decode / execute

DECODE

- **si decodifica il tipo dell'istruzione** in base al suo ***OpCode*** (codice operativo)

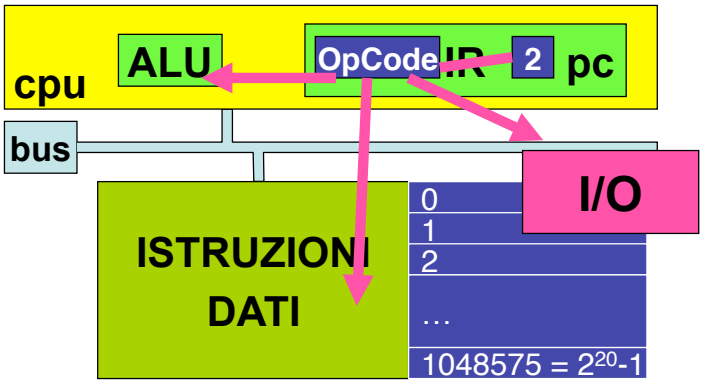


OpCode	*Operazione*
1	Somma
2	Sottrazione
…	…

IL CICLO fetch / decode / execute

EXECUTE

- **si individuano i dati** usati dall'istruzione
- **si trasferiscono tali dati** nei registri opportuni
- **si esegue l'istruzione.**



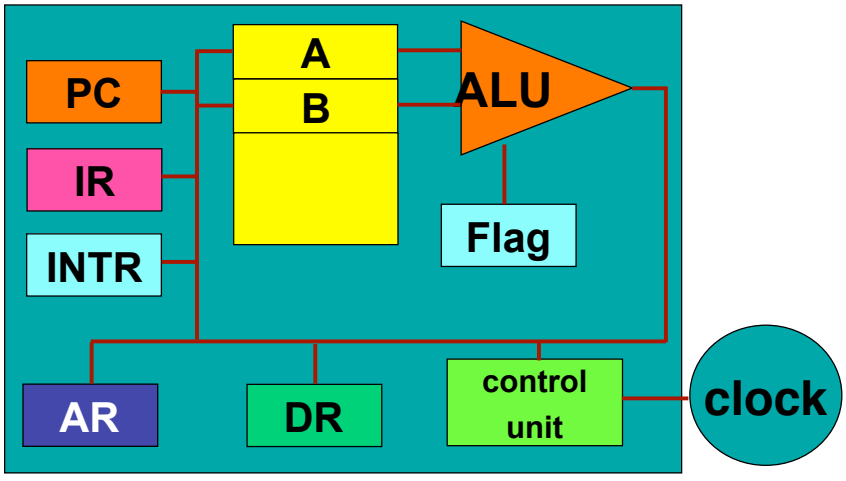
IL CICLO fetch / decode / execute

ATTENZIONE

Istruzioni particolari possono ***alterare il prelievo delle istruzioni da celle consecutive***:

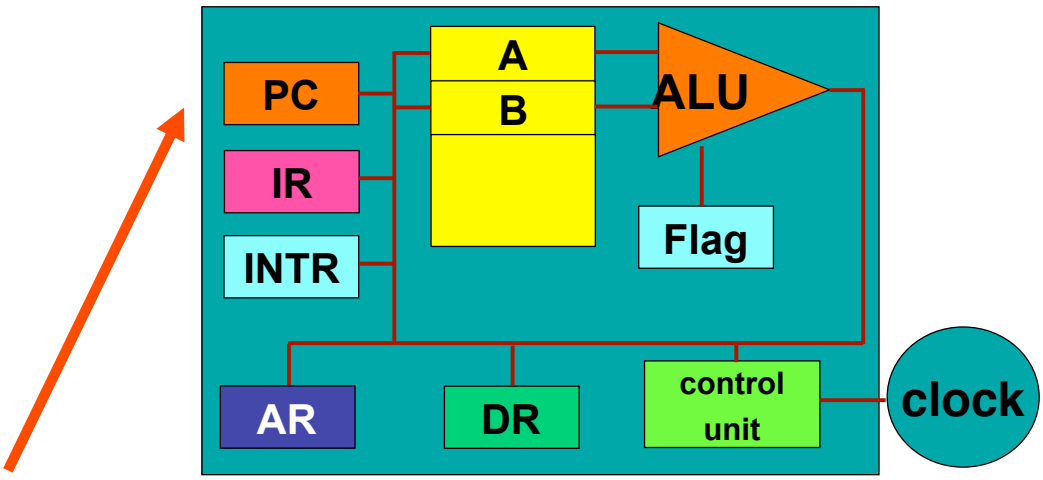
- istruzioni di **salto**
- istruzioni di **chiamata a sotto-programmi**
- istruzioni di **interruzione**

REGISTRI



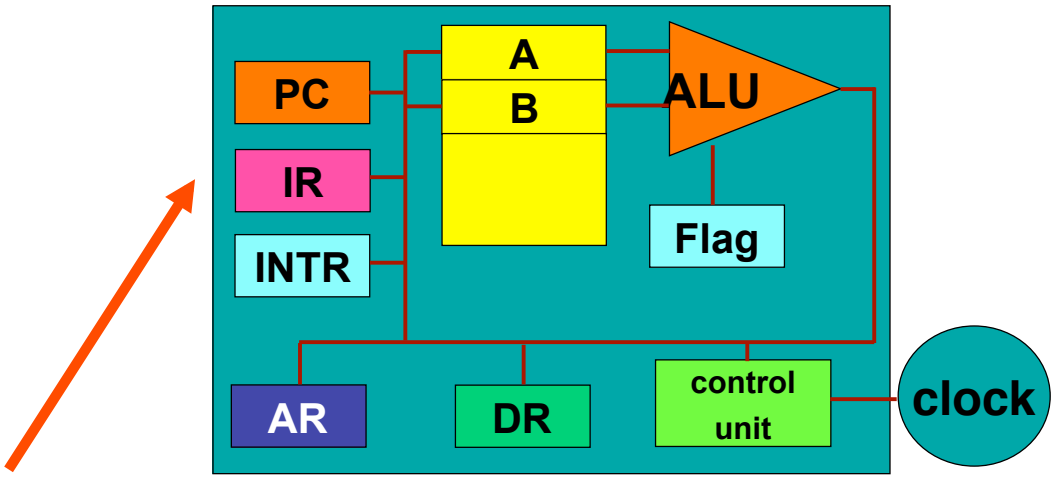
I registri sono ***locazioni* di memoria *interne a CPU***, e come tali *molto veloci*.

REGISTRI



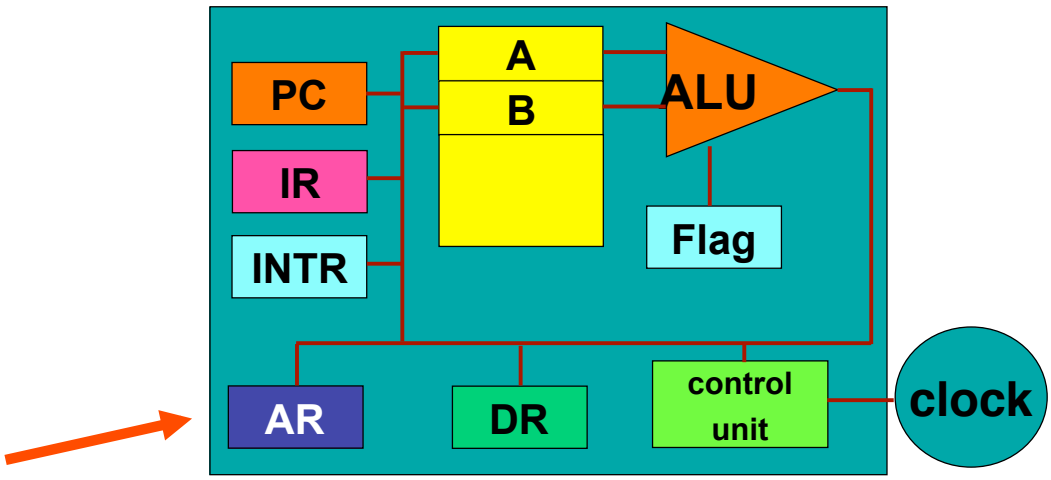
Program Counter (PC)
Indica l'indirizzo della cella di memoria che contiene la prossima istruzione da eseguire

REGISTRI



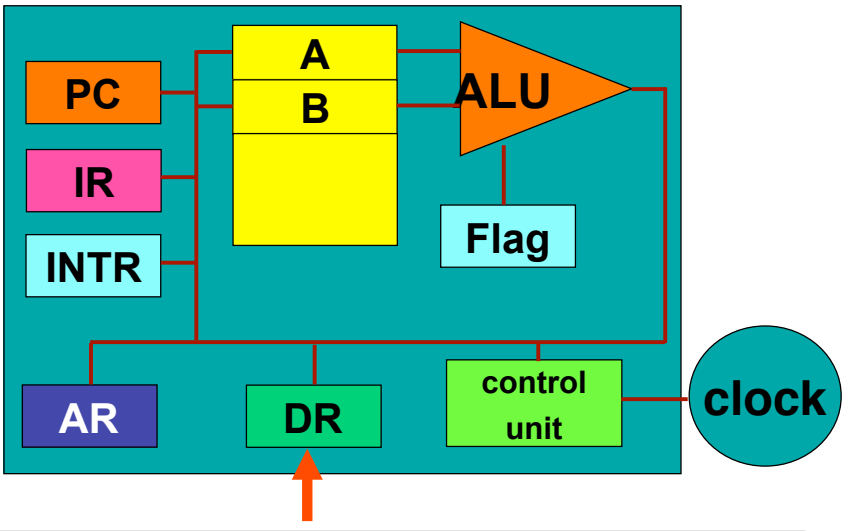
Instruction Register (IR)
Contiene l'istruzione da eseguire.

REGISTRI



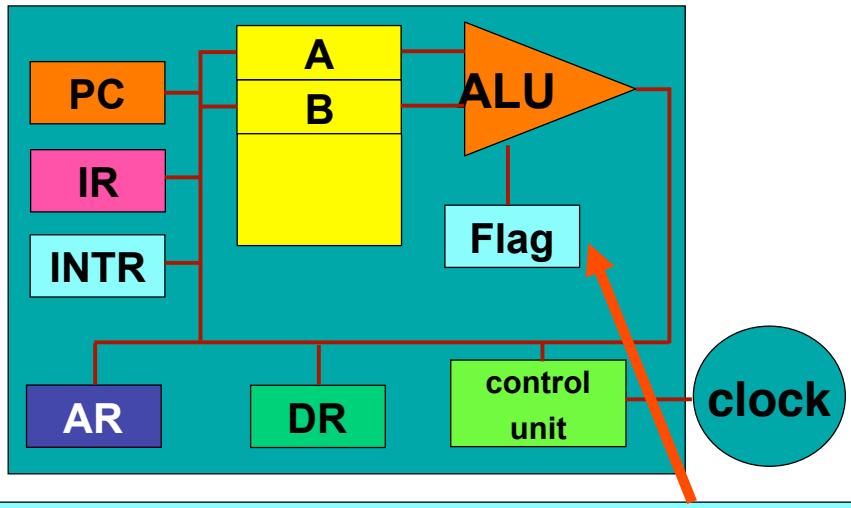
Registro Indirizzi (Address Register, AR)
Contiene l'indirizzo della cella di memoria da selezionare per il trasferimento di un dato con la CPU

REGISTRI



Registro Dati (Data Register, DR)
Contiene il dato attualmente oggetto di elaborazione

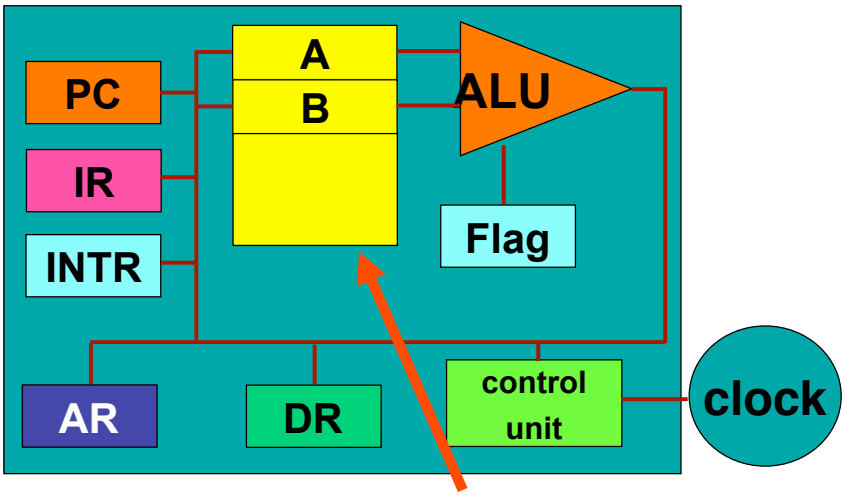
REGISTRI



Registro dei Flag (Flag)
Ogni flag indica la presenza/assenza di una proprietà nell'ultimo risultato generato dalla ALU. Altri bit riassumono lo stato del processore.

2

REGISTRI



Registri di uso generale (A,B,C,...)
Sono usati per contenere dati (in particolare, operandi/risultati di operazioni aritmetico/logiche)

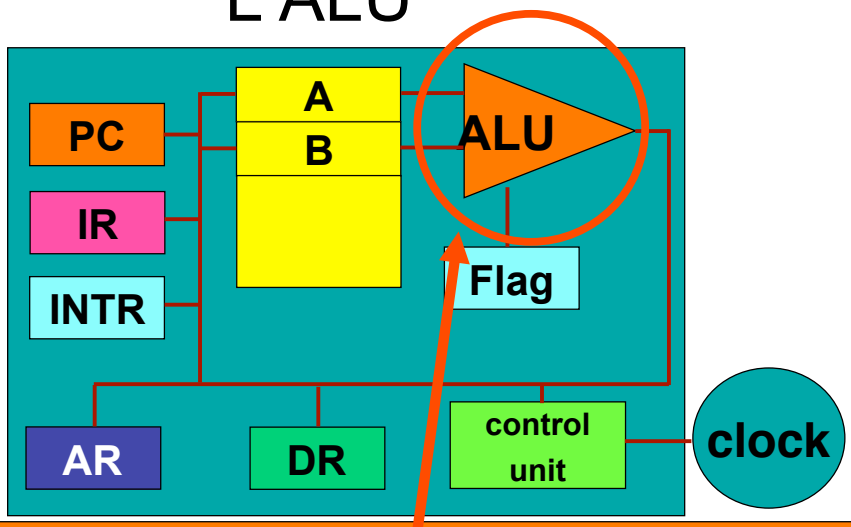
MULTITASKING

Poiché i registri compendiano *tutto lo stato dell'elaborazione* di un certo processo,

- **salvando in memoria il contenuto di tutti i registri** è possibile ***accantonare un processo* per passare a svolgerne un altro**
- **ripristinando dalla memoria il contenuto di tutti i registri** precedentemente salvati è possibile ***ripristinare lo stato di un processo accantonato*, riprendendone l'esecuzione *come se nulla fosse accaduto (“context switch”).***

Questa possibilità è ciò che consente a un sistema operativo di eseguire *più compiti* “allo stesso tempo”

L'ALU



Esegue operazioni aritmetiche, logiche e confronti sui dati della memoria centrale o dei registri.

L'ALU (segue)

ESEMPIO SEMPLICE:

ALU in grado di eseguire **somma**, **sottrazione**, **prodotto**, **divisione** con due operandi contenuti nei registri A e B.

1. I due operandi vengono caricati nei registri A e B;
2. La ALU viene attivata da un comando inviato dalla CPU che specifica il tipo di operazione;
3. Nel registro A viene caricato il risultato dell'operazione eseguita dalla ALU;
4. Il registro FLAG riporta sui suoi bit indicazioni sul risultato dell'operazione (riporto, segno, etc.).

Alterazione di due bit nel registro **Flag**: ***carry*** (riporto) e ***sign*** (segno)

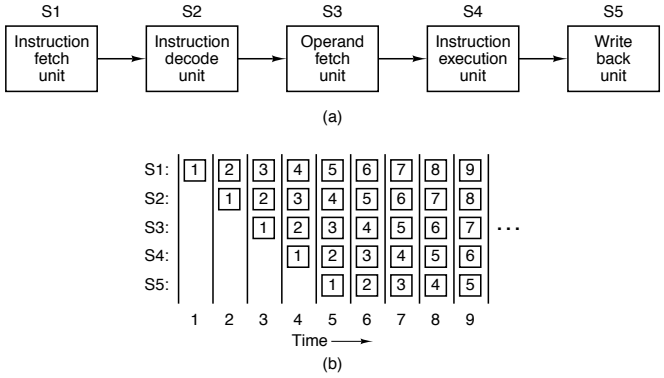
Parallelismo

- Ora vediamo come si possono aumentare le prestazioni di un computer.
- Aumentare la velocità di clock non è una soluzione:
  - qualsiasi architettura ha un limite.
- Invece, con il parallelismo è possibile:
  - eseguire più azioni allo stesso tempo;
  - ottenere prestazioni maggiori alla stessa velocità di clock.
- Il parallelismo può essere presente in due forme:
  - parallelismo a livello d'istruzione;
  - parallelismo a livello di processore.

Parallelismo a Livello d'Istruzione

- In questa forma di parallelismo, cicli di prelievo-decodifica-esecuzione (PDE) di istruzioni distinte sono sovrapposti tra loro.
- Una forma primitiva di questo parallelismo è il **buffer di prefetch**. L'idea è:
  - mentre si svolgono le ultime fasi del ciclo PDE di una istruzione, si carica l'istruzione successiva nel registro chiamato **buffer di prefetch**;
  - quindi, la prossima istruzione è disponibile quando serve, senza dover aspettare una lettura della memoria.
- Generalizzando l'idea a tutte le fasi del ciclo PDE, si ottiene la cosiddetta **pipeline**:
  - tutte le fasi del ciclo PDE sono sovrapposte.

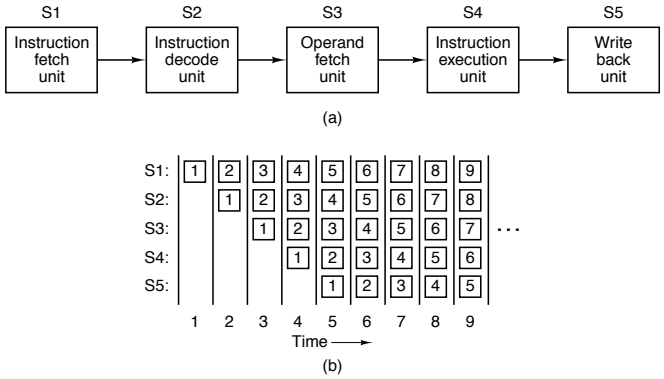
Pipelining (esempio di pipelining a cinque stadi)



(a) Una pipeline con 5 unità, chiamate **stadi**.

- **Lo stadio S1 preleva l'istruzione** dalla memoria e la mette in un buffer, dove rimane finché non serve.
- **S2 decodifica l'istruzione**, determinando il tipo dell'istruzione, e gli operandi di cui ha bisogno.
- **S3 localizza e preleva gli operandi** dai registri o dalla memoria.
- **S4 esegue** l'istruzione, facendo avanzare gli operandi nel percorso dati.
- **S5 scrive** sul registro il risultato finale.

Pipelining (esempio di pipelining a cinque stadi)



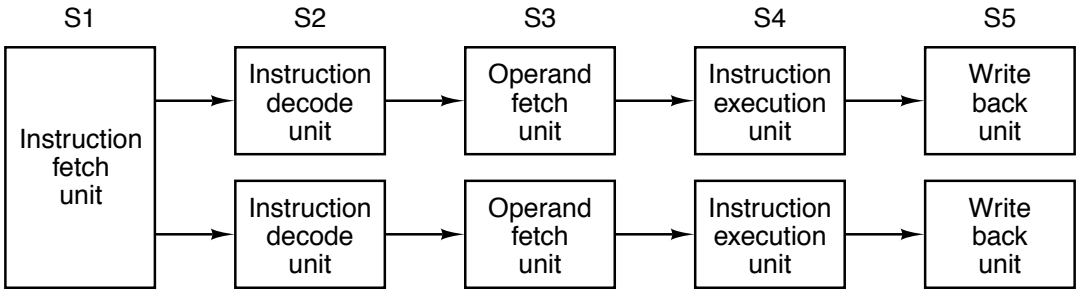
(b) il modello di pipeline.

- **Durante il ciclo di clock 1**: lo stadio S1 preleva l'ist. 1 dalla memoria;
- **Il ciclo di clock 2**: S2 decodifica l'ist. 1, S1 preleva l'ist 2;
- **Il ciclo di clock 3**: S3 preleva gli operandi per l'ist. 1, S2 decodifica l'ist. 2, S1 preleva l'ist 3;
- **Il ciclo di clock 3**: S4 esegue l'ist 1, S3 preleva gli operandi dell'ist. 2, S2 decodifica l'ist. 3, S1 preleva l'ist 4;
- **Il ciclo di clock 4**: S5 scrive il risultato dell'ist 1, S4 esegue l'ist 2, S3 preleva gli operandi per l'ist. 3, S2 decodifica l'ist. 4, S1 preleva l'ist 5.

Doppia Pipeline

- Una generalizzazione del concetto di pipeline è la **pipeline doppia**.
  - Un'unica unità di fetch carica due istruzioni alla volta.
  - Le fasi successive sono sovrapposte.
  - Utilizzata nel processore **Pentium**.

Doppia Pipeline a Cinque Stadi



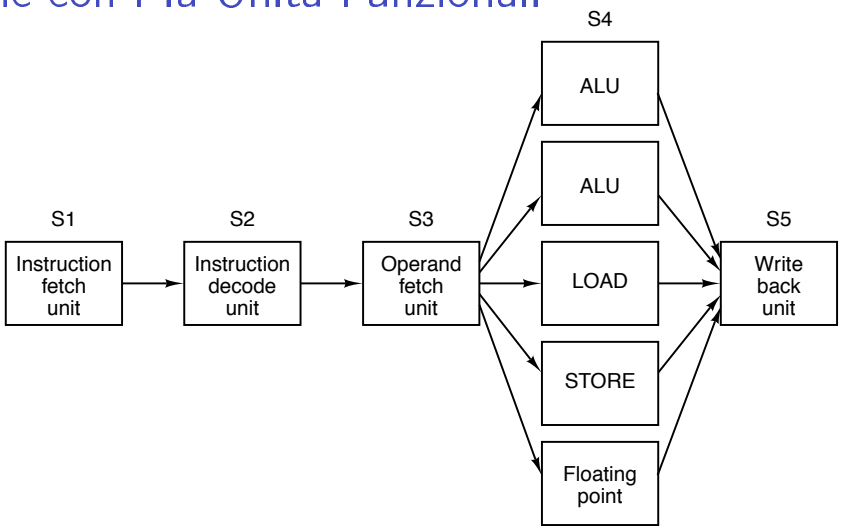
- La figura mostra il progetto di una CPU con due pipeline.
- Una singola unità di *fetch* (S1) preleva **due istruzioni alla volta** e le inserisce nelle due pipeline.
- Ciascuna pipeline è dotata di una ALU (S4).
- Per consentire l'esecuzione in parallelo:
  - non devono utilizzare lo stesso registro;
  - nessuna delle due istruzioni deve dipendere dal risultato dell'altra;

  altrimenti avremmo **conflitti**.

Architettura Superscalare

- Un'ennesima generalizzazione a più di due pipeline richiederebbe la duplicazione di troppi componenti hardware.
- Invece si può avere un'unica pipeline con più unità funzionali specializzate nello stadio di esecuzione, S4.
- In questo caso si parla di **architettura superscalare**.
  - Utilizzata nel processore **Pentium II**.

Singola Pipeline con Più Unità Funzionali



- La figura mostra il progetto di una CPU con una pipeline e più unità funzionali di esecuzione.
  - In particolare, S4 ha due ALU.
- Per sfruttare il parallelismo, lo stadio S3 deve essere molto più veloce dello stado S4.

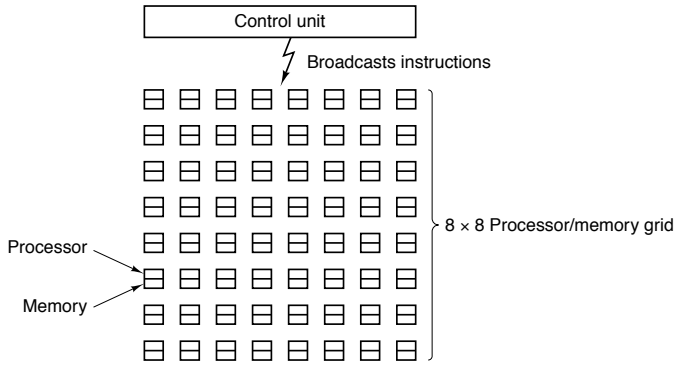
Parallelismo a Livello di Processore

- Con le architetture con parallelismo a livello di istruzione si riescono ad ottenere aumenti di prestazioni di un fattore 10.
- Una forma di parallelismo più forte, per avere un guadagno maggiore (50, 100, o più) consiste nell'avere **molte CPU** che lavorano in modo coordinato.
- Un primo esempio sono gli **array computer** dove:
  - una singola unità di controllo esegue un broadcast di **una sequenza di istruzioni**;
  - molti **processori identici** eseguono tale sequenza di istruzioni su **insiemi diversi di dati**.
  - In questo caso, si parla di **SIMD** (Single Instruction-stream, Multiple Data-stream).

→ Perché si vogliono avere molte CPU che eseguono le stesse istruzioni?

- Esistono molti problemi di calcolo scientifico, o dell'ingegneria, che hanno una struttura regolare.
- In questi problemi, spesso gli stessi calcoli possono essere effettuati nello stesso tempo sui dati diversi.

Array Computer

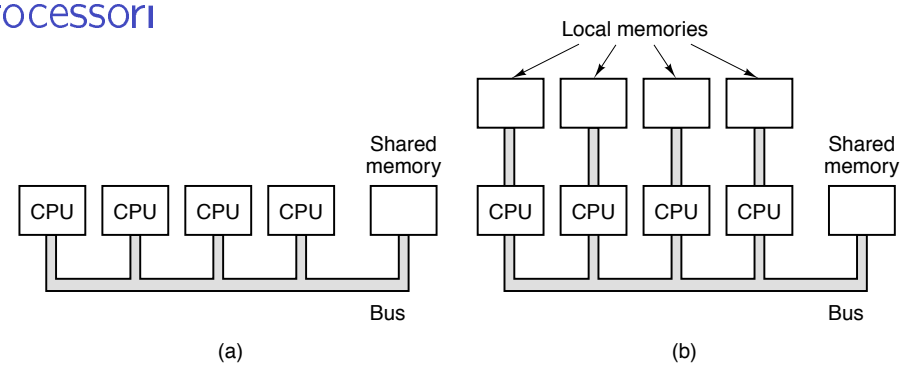


- La figura rappresenta un array di processori ILLIAC IV.
  - L'unità di controllo trasmette in broadcast la prima istruzione a tutti i processori.
  - I processori eseguono l'istruzione allo stesso tempo su dati diversi.
  - Poì, si passa alla prossima istruzione che viene trattata allo stesso modo.
  - Si continua così fino alla fine delle istruzioni.
  - Si noti che ciascun processore utilizza i dati letti dalla propria memoria (caricata all'inizio).

Multiprocessori

- I processori di un array computer sono identici ma non sono indipendenti, perché esiste un'unica unità di controllo condivisa.
- Esistono però altre architetture parallele.
- Una possibilità con CPU indipendenti e coordinate è di avere:
  - molti processori che **condividono una memoria**,
  - ma che possono eseguire istruzioni diverse.
- Questa architettura viene chiamata **multiprocessore**, e il suo modello di calcolo è chiamato **MIMD** (Multiple Instruction-stream, Multiple Data-stream).
- La condivisione della memoria è l'aspetto più critico dei multiprocessori.
  - Le CPU devono co-ordinarsi.
  - L'accesso alla memoria avviene con un bus, che rappresenta un collo di bottiglia (*bottleneck*).
- Per questo motivo sono stati studiati vari schemi...

Multiprocessori

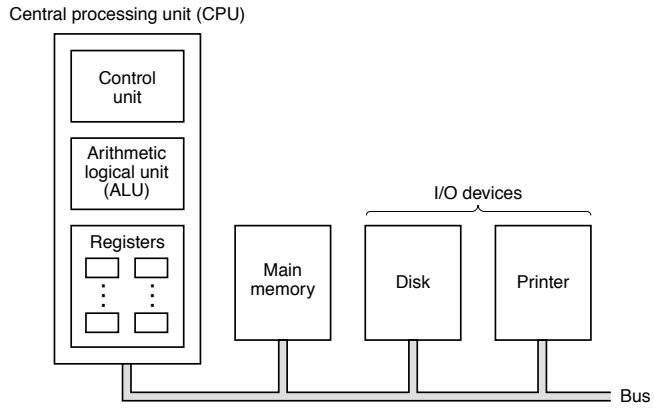


- Le figure rappresentano vari schemi di multiprocessori:
  - La prima mostra uno schema con un singolo bus e più CPU, tutte connesse a un'unica memoria.
  - La seconda mostra un'architettura simile alla (a), in cui però ogni processore ha una memoria locale, non accessibile agli altri.
- In entrambi gli schemi, le CPU sono independenti con le loro proprie unità di controllo.
- Si noti che la seconda soluzione riduce l'utilizzo della memoria condivisa e il traffico sul bus.

Multicomputer

- Multiprocessori con numeri molto grandi di CPU sono difficili da realizzare.
  - Il problema è collegare tutte le CPU alla memoria.
- Per questo motivo, per avere moltissimi computer in una sola architettura, bisogna evitare di avere la memoria condivisa.
- I **multi-computer** sono la forma più estrema di parallelismo.
- Si tratta di molti processori che:
  - **non condividono una memoria** ;
  - comunicano invece **per scambio di messaggi**.
- In questo caso, moltissime (e.g. 10.000) CPU possono cooperare.
- Esiste però un problema: la velocità delle connessioni tra i computer.
  - Non si possono avere collegamenti diretti tra 10.000 computer!
  - Spesso un messaggio deve passare attraverso più computer.
- Comunque, lo scambio di un singolo messaggio richiede pochi microsecondi (10^{-6}s).

Memoria principale



- Memoria **principale** (e.g. la **RAM**) è **volatile**: il suo contenuto è perso quando si spegne il calcolatore.
- Memoria **secondaria** (e.g. hard disk) **non è volatile**: il suo contenuto non viene perso quando si spegne il calcolatore.

Bit

- L'informazione digitale è memorizzata nella memoria utilizzando dei valori di una quantità fisica. E.g.:
  - la corrente
  - la tensione
- L'unità base di memorizzazione è la **cifra binaria**, detta **bit**, che assume valore 0 o 1.
- Il sistema binario è migliore di altri sistemi perché non richiede di distinguere tra i valori adiacenti:
  - è sufficiente distinguere tra "valore basso" e "valore alto".
  - E.g., -2V è "basso", +2V è "alto",

  → tutti i valori positivi (+1V, +3V, etc) sono riconosciuti come valori "alti".

  → tutti i valori negativi (-1V, -3V, etc) sono riconosciuti come valori "bassi".
- Il sistema binario è il modo più efficiente di rappresentare l'informazione.

Insiemi di bit

- L'unità minima di memoria si chiama **cella** o **locazione**, e contiene un insieme di bit.
- Quanti valori distinti si possono rappresentare in una cella?
  - Con un bit si possono rappresentare due valori: 0 e 1.
  - Con due bit si possono rappresentare 4 valori:

    00
    01
    10
    11
- In generale, con n bit si possono rappresentare 2^n valori distinti.
- Ciascun valore si chiama "configurazione" di bit.

Che Significato Hanno le Configurazioni di Bit?

- È possibile assegnare a ciascuna configurazione un significato arbitrario.
- E.g., si possono assegnare ai valori 00/01/10/11 il significato di dati (numeri interi o caratteri), o di istruzioni per la CPU!

00	→	0
01	→	1
10	→	2
11	→	3

00	→	0
01	→	1
10	→	−2
11	→	−1

00	→	A
01	→	B
10	→	C
11	→	D

00	→	ADD
01	→	INC
10	→	LOAD
11	→	STORE

Byte, Parole e Indirizzi

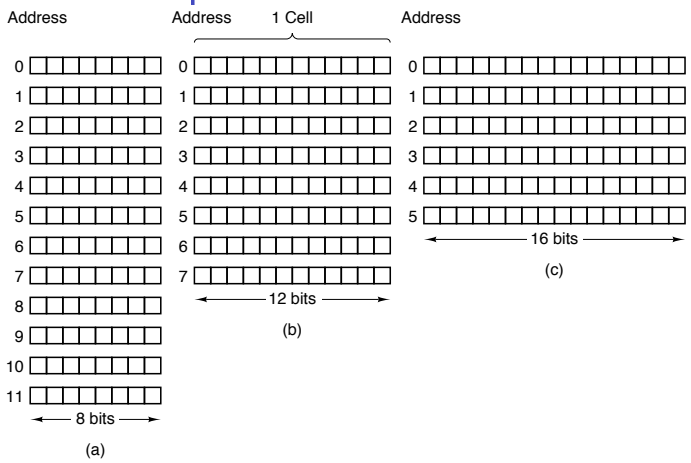
→ Come sono organizzati i bit nelle celle di memoria?

- Sono raggruppati:
  - in **byte**, cioè gruppi di 8 bit;
  - oppure in **parole** (**word**), cioè gruppi di byte.
- Si noti che calcolatori diversi usano parole di dimensione diversa. E.g.:
  - Nei computer a 32 bit, la parola è composta di 4 byte.
  - Nei computer a 64 bit, la parola è composta di 8 byte.

→ Come sono organizzate le celle dentro la memoria?

- La posizione di una cella in memoria è indicata da un numero, chiamato **indirizzo**.
- Celle adiacenti hanno indirizzi consecutivi.
- All'interno del computer, anche gli indirizzi sono rappresentati con il sistema binario.
  - Vediamo un esempio...

Un Esempio: Memoria a 96 Bit



- La figura rappresenta 3 diversi modi di organizzare la memoria, con
  - (a) 12 celle di 8 bit (1 byte);
  - (b) 8 celle di 12 bit;
  - (c) 6 celle di 16 bit (2 byte).

- Le celle adiacenti hanno indirizzi consecutivi.
  - E.g., nella figura (a) gli indirizzi vanno da 0 a 11.
- Nel computer, gli indirizzi di memoria sono espressi in notazione binaria.
  - Se un indirizzo ha m bit, il massimo numero di celle indirizzabili è 2^m.
  - E.g., nel caso (a) servono almeno 4 bit per esprimere tutti i numeri da 0 a 11.
- Un gruppo di celle adiacenti (e.g. ogni 4) forma una parola.

LA MEMORIA CENTRALE

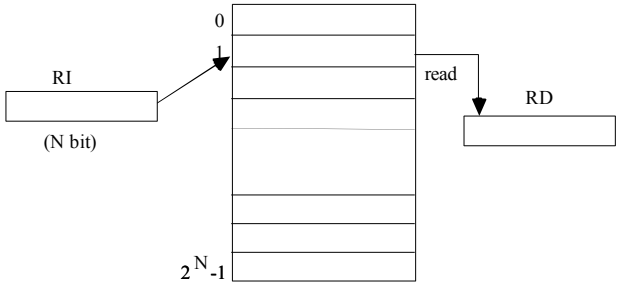
INDIRIZZAMENTO

- E' l'attività con cui l'elaboratore seleziona una particolare cella di memoria
- Per farlo, l'elaboratore pone l'indirizzo della cella desiderata nel Registro Indirizzi (AR).
  - **se AR è lungo *N* bit, si possono indirizzare 2^N celle di memoria** (numerate da 0 a 2^N-1)
  - esempio: N=10 ⇒ 1024 celle.
- **Oggi**, AR è lungo tipicamente **32 / 40 bit**
  → SPAZIO INDIRIZZABILE di 4 GB / 1 TB

LA MEMORIA CENTRALE (2)

OPERAZIONI

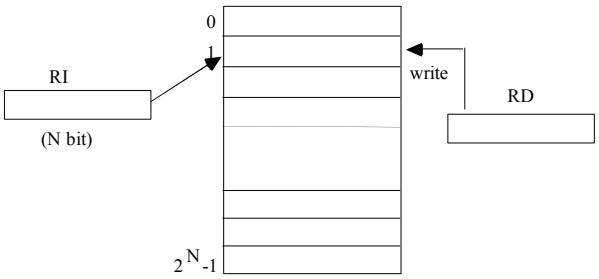
- **Lettura *(Read)***: il contenuto della cella di memoria indirizzata dal Registro Indirizzi è copiato nel Registro Dati.



LA MEMORIA CENTRALE (3)

OPERAZIONI

- **Scrittura *(Write)***: il contenuto del Registro Dati è copiato nella cella di memoria indirizzata dal Registro Indirizzi.



DISPOSITIVI DI MEMORIA

DISPOSITIVI FISICI

- **RAM**: Random Access Memory (ad accesso diretto): su di essa si possono svolgere operazioni sia di lettura che di scrittura
- **ROM**: Read Only Memory (a sola lettura): non volatile e non scrivibile dall'utente; in essa sono contenuti i dati e programmi per inizializzare il sistema
- **PROM**: Programmable ROM. Si può scrivere soltanto una volta, mediante particolari apparecchi (detti programmatori di PROM).

DISPOSITIVI DI MEMORIA

DISPOSITIVI FISICI

- **EPROM**: Erasable-Programmable ROM (si cancella sottoponendola a raggi ultravioletti).
- **EEPROM**: Electrically-Erasable-PROM (si cancella elettricamente).

Il *Firmware* è costituito da software memorizzato su ROM, EPROM, etc. (codice microprogrammato).

Memoria Cache

- Le moderne CPU sono **molto più veloci** delle memorie.
- Quando la CPU richiede un dato dalla memoria, è possibile che per un certo tempo la CPU sia costretta ad aspettare l'arrivo del dato.
- La CPU quindi **perde tempo** a causa della memoria!
- Una soluzione a questo problema è ordinare le istruzioni dei programmi nel modo migliore...

Memoria Cache

Soluzione 1. Si può operare nel modo seguente:

(1) cominciare le operazioni di lettura (READ) appena si incontrano nel programma,

(2) mentre si aspetta il dato, eseguire in parallelo le istruzioni seguenti.
Se però si incontra una istruzione che deve usare il dato, la CPU deve restrae "idle" (inattiva).
Se la memoria è lenta, aumenta la probabilità di avere la CPU idle.

Soluzione 2. Alternativamente, il compilatore può generare programmi che non hanno questo problema.
E.g., ogni richiesta di lettura può essere **seguita** da alcune istruzioni che non richiedono altre letture.
Purtroppo, questa soluzione non è sempre applicabile.

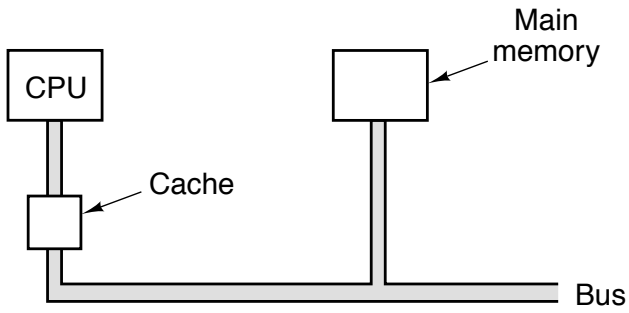
Memoria Cache

- Quindi, queste soluzioni non risolvono il problema in generale.
- Oltre a questi problemi **tecnologici**, vi sono anche considerazioni di carattere **economico**:
  - La tecnologia consente di realizzare memorie veloci quanto la CPU,
  - però, il bus CPU-memoria è comunque molto più lento,
  - quindi, per andare al massimo della velocità, queste memorie devono essere integrate nella CPU,
  - ma integrare la memoria nella CPU significa avere CPU molto grandi...
  - ...e CPU grandi sono molto costose.

Memoria Cache

- Una soluzione a questo problema è utilizzare una memoria non troppo grande ma molto veloce, chiamata **cache**.
- Le parole di memoria **usate più spesso** sono tenute all'interno della cache.
- La cache è collegata alla CPU con un bus molto veloce.
- Quando la CPU ha bisogno di una parola:
  1. la cerca nella cache e,
  2. solo se la parola non è in cache, la richiede alla memoria centrale.
     In tal caso, il contenuto della memoria centrale è trasferito nella cache.
- Per avere un aumento di prestazioni, i dati trasferiti nella cache dalla memoria centrale devono essere usati molte volte.

Memoria Cache



- Questa figura mostra una tipica organizzazione di CPU, cache e memoria centrale.
- Se durante un piccolo intervallo di tempo una parola è letta o scritta k volte, il calcolatore dovrà leggere/scrivere 1 sola volta dalla memoria (lenta), e $k - 1$ volte dalla memoria cache (veloce).
- Quindi, per avere buone prestazioni, k deve essere "grande".

Hit Ratio

- Le prestazioni globali del sistema dipendono da **quale frazione** delle richieste di accesso alla memoria può essere soddisfatta direttamente dalla cache.
- Supponiamo che:
  - c sia il tempo di accesso alla memoria cache,
  - m sia il tempo di accesso alla memoria centrale,
  - h sia **hit ratio**, cioè la frazione di riferimenti che può essere soddisfatta dalla cache,
  - $1 - h$ è chiamato **miss ratio**, cioè la frazione di riferimenti che non può essere soddisfatta dalla cache.
- Dato che:
  - ogni accesso a un dato (hit o miss) è prima fatto sulla cache, e quindi richiede un tempo c;
  - se il dato non è disponibile in cache (miss), deve essere preso dalla memoria centrale, e scritto nella cache;
  - questa operazione, fatta con frequenza $1 - h$, richiede un tempo m;

  il tempo medio di accesso (t) al dato è $c + (1 - h)m$.

Il Principio di Località

- Per avere buone prestazioni, è necessario avere un alto hit ratio:
  - Quando h è vicino a 1, t è vicino a c, mentre quando h è vicino a 0, t è vicino a $c + m$.
  - Nel secondo caso, la cache *peggiora* le prestazioni!
- Per fortuna ci aiuta il **principio di località**.
- Il principio di località è una caratteristica di molti programmi:
  - Riferimenti alla memoria vicini nel tempo sono spesso anche vicini nello spazio.
  - Inoltre, il riferimento a una data cella di memoria è spesso ripetuto nel giro di poco tempo.
- Per questo motivo, le cache sono organizzate in **linee**:
  - Quando si verifica un fallimento (miss), un'intera linea, che contiene molti indirizzi vicini, è portata in cache dalla memoria centrale.

Organizzazioni della cache

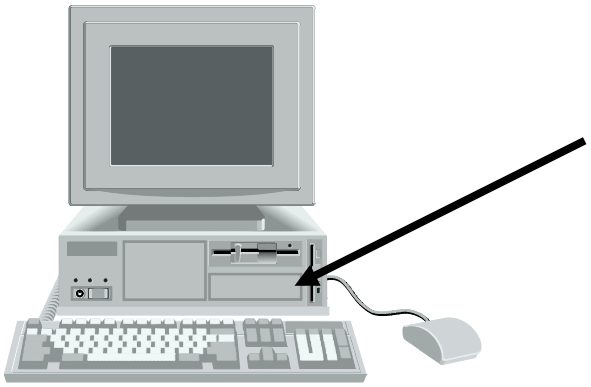
- Le cache, se progettate bene, possono aumentare le prestazioni.
- Il progetto delle cache quindi è un aspetto molto importante del progetto delle CPU.
- E.g., possiamo avere:
  - una **cache unificata**, con dati e istruzioni;
  - due **cache specializzate**, una per le istruzioni, una per i dati;
  - più cache, organizzate a **livelli** (la più veloce nella CPU, le altre fuori).

Organizzazioni della cache

Altri aspetti da considerare:

- la dimensione della cache:
  - più è grande, migliore è hit ratio, maggiore il costo.
- la dimensione delle linee di cache:
  - meno linee più lunghe possono essere meglio di molte linee più corte. Perché?
    - la cache è più semplice;
    - linee lunghe sfruttano meglio il principio di località.
  - però, nel caso di un miss:
    - bisogna sostituire l'intera linea, che ha un costo;
    - molte parole sono cancellate dalla cache, e questo può causare altri miss.
- l'organizzazione interna della cache, cioè:
  - la logica per scegliere quale linea cancellare in caso di miss;
  - la consistenza con la memoria centrale.

MEMORIA DI MASSA



- Dischi
- CD, DVD, …
- Nastri
- ...

- memorizza **grandi quantità** di informazioni
- **persistente** (le informazioni sopravvivono all'esecuzione dei programmi: non si perdono spegnendo la macchina)
- accesso molto meno rapido rispetto alla memoria centrale (**milli**secondi contro **nano**secondi / differenza 10^6)

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI FONDAMENTALI:

- **ad *accesso sequenziale*** (ad esempio, **NASTRI**): per recuperare un dato è necessario accedere prima a tutti quelli che lo precedono sul dispositivo
- **ad *accesso diretto*** (ad esempio, **DISCHI e penne USB**): si può recuperare direttamente un qualunque dato memorizzato

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI F

- **ad *accesso s***
  recuperare un
  che lo precedo
- **ad *accesso d***
  si può recuper
  memorizzato

Unità di misura: tempo di accesso

- msec (10^{-3} secondi)
- μsec (10^{-6} secondi)
- nsec (10^{-9} secondi)
- …

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI FONDAMEN

- **ad *accesso sequenzial***
  recuperare un dato è nec
  che lo precedono sul dis
- **ad *accesso diretto*** (ad
  si può recuperare diretta
  memorizzato

Unità di misura: capacità

- KByte (1.024 Byte)
- MByte (1.048.576 Byte)
- GByte (1.073.741.824 Byte)
- TByte (1.099.511.627.776 Byte)

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI FONDAME

- **ad *accesso sequenzia***
  recuperare un dato è ne
  che lo precedono sul dis
- **ad *accesso diretto*** (ad
  si può recuperare direttamente un qualunque dato memorizzato

Unità di misura: bandwidth

- Kbps (10^3 bit al secondo)
- Mbps (10^6 bit al secondo)
- …

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI FONDAMEN

- **ad *accesso sequenziale***
  recuperare un dato è nece
  che lo precedono sul dispo
- **ad *accesso diretto*** (ad es
  si può recuperare direttam
  memorizzato

Misure di costo

- $/KByte
- $/MByte
- $/GByte
- …

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

DISPOSITIVI DI MEMORIA DI MASSA

DUE CLASSI FONDAMENTA

- **ad *accesso sequenziale*** (a
  recuperare un dato è neces
  che lo precedono sul dispos
- **ad *accesso diretto*** (ad ese
  si può recuperare direttamente un qualunque dato memorizzato

Tecnologie

- Magnetica (ML hours)
- Ottica
- Flash memory (circ. integrati)

Caratteristiche:

- ***Tempo di accesso***
- ***Capacità***
- ***Tempo di trasferimento (bandwidth)***
- ***Costo***
- ***Tecnologia e affidabilità***

DISPOSITIVI MAGNETICI

- L'area del dispositivo è suddivisa in **micro-zone**
- Ogni micro-zona memorizza una **informazione elementare** sotto forma **di *stato di magnetizzazione*:**

  area magnetizzata / area non magnetizzata
- Ai due possibili stati di magnetizzazione vengono **associate le due cifre binarie 0 e 1**

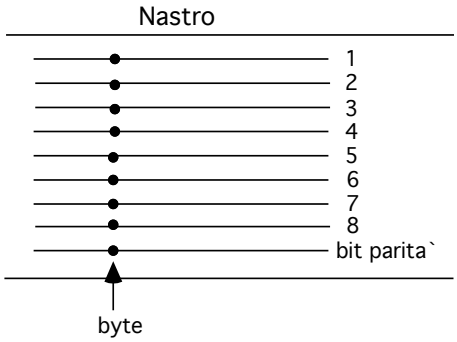
  bit (Binary digIT)
- Quindi, **ogni micro-zona memorizza 1 bit**
- Per memorizzare informazioni più complesse si considerano *collezioni di bit*:

  BYTE (collezione di **8 bit**) e suoi multipli

NASTRI MAGNETICI

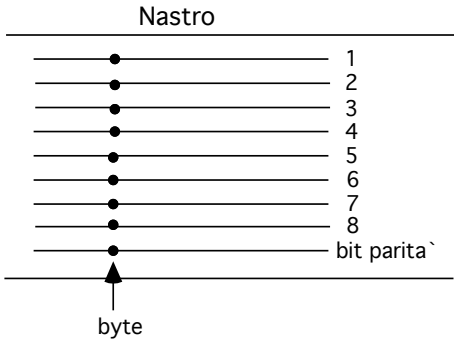
Nastri di materiale magnetizzabile arrotolati su supporti circolari, o in cassette.

Sul nastro sono tracciate delle **piste orizzontali parallele** (di solito 9, di cui 8 corrispondono ad un byte e la nona è il bit di parità).



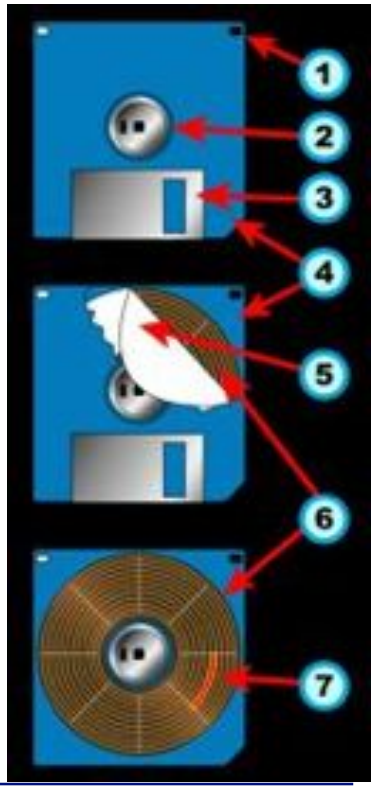
NASTRI MAGNETICI

I dati sul nastro sono organizzati in zone contigue dette ***record***, separate da zone prive di informazione (*inter-record gap*).



- Tutte le **elaborazioni** sono **sequenziali:** le operazioni su uno specifico record sono ***lente***
- Oggi servono solo per mantenere copie di riserva (***backup***) dei dati

DISCHETTI (FLOPPY)



Fine anni ’60-oggi (poco)

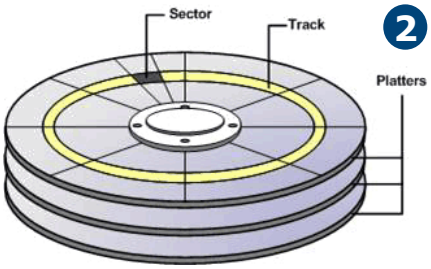
Sono dischi magnetici di ***piccola capacità***, portatili, usati per trasferire informazioni tra computer diversi.

Sono costituiti da un **unico disco** con due superfici.

Sopravvivono solo quelli da 3.5” di diametro (1.4 MB)

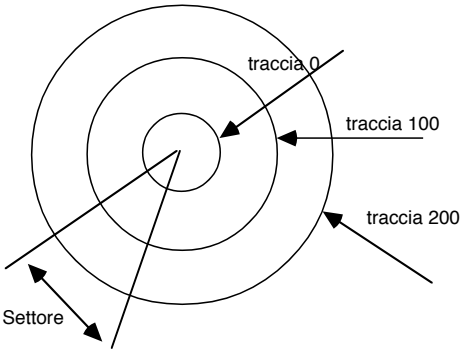
Dati scritti su *tracce* (7), create in fase di ***formattazione***.

DISCHI MAGNETICI



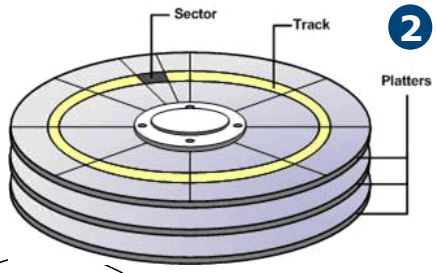
Un disco consiste in un certo numero di ***piatti*** con **due superfici** che ruotano attorno ad un perno centrale.

Ogni superficie dispone di una propria **testina di lettura / scrittura.**

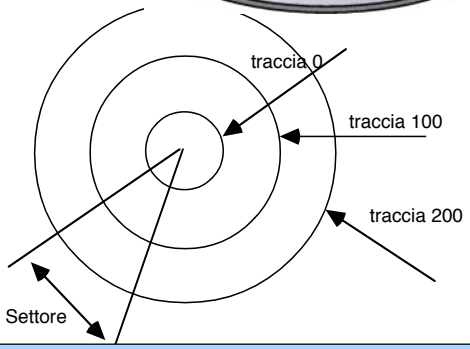


Le superfici sono organizzate in ***cerchi concentrici*** (**tracce**) e in ***spicchi*** di ugual grandezza (**settori**).
Le tracce equidistanti dal centro formano un **cilindro.**

DISCHI MAGNETICI



I dati sono scritti in posizioni successive *lungo le tracce*: ogni bit corrisponde a uno stato di *magnetizzazione* del materiale magnetico della superficie del disco.



Ogni ***blocco*** del disco è identificato con la terna
⟨***superficie***, ***traccia***, ***settore***⟩
Per effettuare il trasferimento dei dati in memoria centrale occorre disporre di un'area di memoria (*buffer*) di dimensioni pari al blocco.

DISCHI MAGNETICI: PERFORMANCE

Grandezze da tenere in considerazione:
seek time (~10 ms)
vel. rotazione (~10000 RPM)
transfer rate (~ 10 MBps)

Tempo medio necessario a trasferire un blocco di dati da HD a RAM: (~)
seek time + ½ rotation time + dimensione dati / transfer rate
e.g. blocco da 1MB:
10ms + 3ms + 1MB / 10 MBps
= 13 ms (latency) + 100 ms (transfer)

DISCHI MAGNETICI: PERFORMANCE

NOTA: il vero collo di bottiglia può diventare il sistema di trasferimento dall'HD verso l'esterno!!
→PCI-X Bus 133MHz 64bit: 1 GBps
→Firewire 400/800 Mbps
→USB 2.0 480 Mbps

Grandezze da tenere in considerazione:
seek time (~10 ms)
vel. rotazione (~10000 RPM)
transfer rate (~ 10 MBps)

Tempo medio necessario a trasferire un blocco di dati da HD a RAM: (~)
seek time + ½ rotation time + dimensione dati / transfer rate
g. blocco da 1MB:
MBps
ms (transfer)

Redundant Arrays of Inexpensive Disks (RAID)

- I **RAID** sono **gruppi** di dischi IDE, SCSI, etc, organizzati allo scopo di **distribuire** e/o **replicare** l'informazione.
- I RAID sono visti nell'insieme come **un unico disco** virtuale.
- I diversi modi di organizzare i dischi RAID sono chiamati **livelli RAID**, e vanno da 0 a 5.
  - Il **RAID di livello 0** è studiato per aumentare la velocità di trasferimento.
    - I dati sono distribuiti tra vari dischi.
    - Diversi dischi possono leggere/scrivere dati diversi allo stesso tempo.
    - Però, se si rompe un disco vengono persi dati.

Redundant Arrays of Inexpensive Disks (RAID)

- I **RAID** sono **gruppi** di dischi IDE, SCSI, etc, organizzati allo scopo di **distribuire** e/o **replicare** l'informazione.
- I RAID sono visti nell'insieme come **un unico disco** virtuale.
- I diversi modi di organizzare i dischi RAID vengono chiamati **livelli RAID**, e vanno da 0 a 5.
  - Il **RAID di livello 1** è studiato per duplicare tutti i dati.
    - Esiste una copia di ciascun disco.
    - La scrittura avviene sulle due copie.
    - La lettura avviene da una copia sola.
    - Diversi dischi possono leggere/scrivere dati diversi allo stesso tempo.
    - Anche se si rompe un disco, non si perdono dati.

Redundant Arrays of Inexpensive Disks (RAID)

- I **RAID** sono **gruppi** di dischi IDE, SCSI, etc, organizzati allo scopo di **distribuire** e/o **replicare** l'informazione.
- I RAID sono visti nell'insieme come **un unico disco** virtuale.
- I diversi modi di organizzare i dischi RAID vengono chiamati **livelli RAID**, e vanno da 0 a 5.
  - I **RAID di livello 2, 3, 4 e 5** distribuiscono i dati in modo da ottenere codici a rilevazione e correzione di errore.
    - RAID 2 usa il codice di Hamming per la correzione d'errore.
    - La lettura avviene in parallelo.
    - Anche se si rompe un disco, non si perdono dati (si può ricostruire l'informazione usando il codice di Hamming).

DISPOSITIVI OTTICI

1984, CD-ROM (Compact-Disk Read-Only Memory)
- Capacità: > 600 MB
- Costo: < $1
- Velocità di trasferimento:
  - originariamente 150 KB / s (“1X”)
  - oggi 24, 32, 52 volte tanto…

1984, WORM (Write Once Read Many)
- Sono dischi ottici scrivibili (una sola volta)
- Parenti stretti dei CD audio (CD-DA, 1982)
- Accesso diretto ai settori (capacità 2.048 KB)

2

DISPOSITIVI OTTICI

1986, CD - I (Compact-Disk Interactive)
- Per memorizzare immagini, filmati, grafica, suono, testi e dati (*multimedialità*).

1997, DVD (Digital Video Disk)
- Evoluzione del CD-ROM
- Capacità di 5,7 / 20 / 50 … GB
- Velocità di trasferimento molto elevata

DVD inizialmente ideato per film e opere pesantemente multimediali.

DISPOSITIVI OTTICI

Ormai il CD/DVD sono (assieme alla rete) tra i principali mezzi per lo scambio di grandi quantità di informazioni

- installazione di nuovi programmi di utilità
- archiviazione di immagini, suoni, opere multimediali
- copie di riserva (backup)
- distribuzione di materiale pubblicitario o “di prova”

Affidabilità: fino a 10-15 anni.

USB STICK

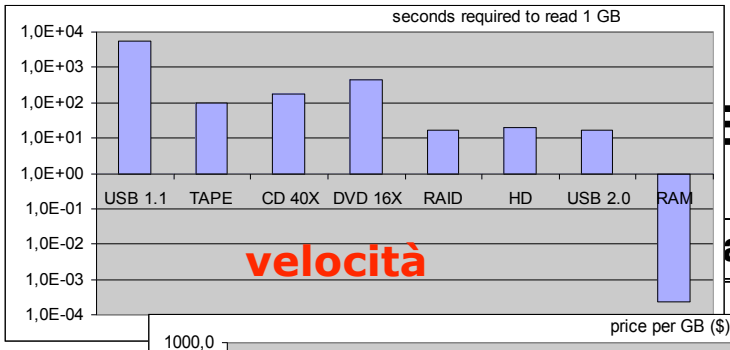
Flash memory stick ("penne USB"):

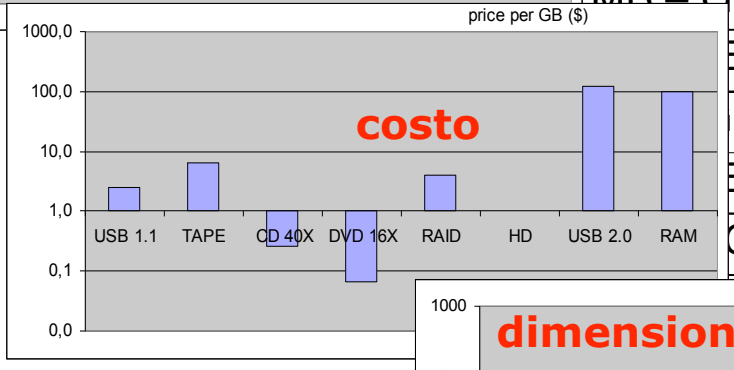
- memorie persistenti che possono essere riscritte/ cancellate piu` volte
- capacità: ordine dei GB
- Nota: USB 1.1 vs. 2.0: 40 volte più veloce

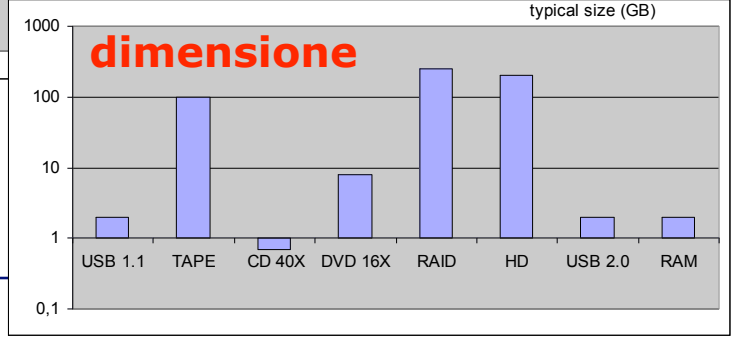
CAPACITÀ DELLE MEMORIE

Tipo di memoria	Capacità
Memoria centrale	10^2 MB – GB
Dischi magnetici	10 GB - TB
Penne USB	10^2 MB - GB
Nastri (bobina)	10 - 10^2 MB
Nastri (cassetta)	10^2 MB - 10 GB
Dischi ottici	650 MB - 50 GB

2







E MEMORIE

apacità

MB – GB

B

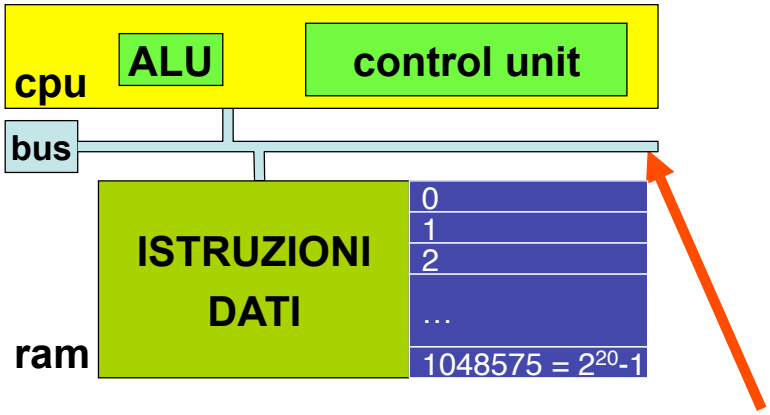
B

B

GB

Fondamenti di Informatica T-AB

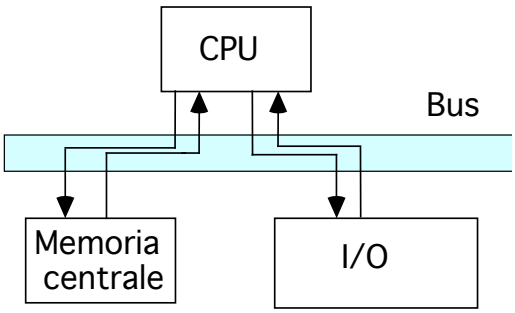
BUS DI SISTEMA



Il Bus di Sistema interconnette la CPU, le memorie e le interfacce verso dispositivi periferici (I/O, memoria di massa, etc.)

2

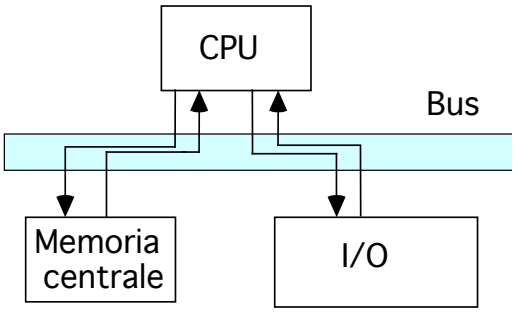
BUS DI SISTEMA



Il Bus collega ***due unità funzionali alla volta***:
- **una trasmette**…
- … e **l'altra riceve**.

Il trasferimento dei dati avviene o *sotto il controllo della CPU*, o mediante *accesso diretto alla memoria (DMA).*

BUS DI SISTEMA



Il Bus è in realtà **un insieme di linee diverse:**
- **bus dati (*data bus*)**
- **bus indirizzi (*address bus*)**
- **bus comandi (*control bus*)**

2

BUS DI SISTEMA

BUS DATI
- **bidirezionale**
- serve per trasmettere dati ***dalla memoria o viceversa***.

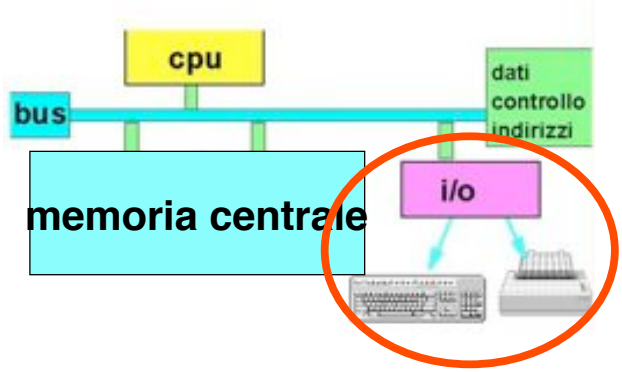
BUS INDIRIZZI
- **unidirezionale**
- serve per trasmettere ***il contenuto del registro indirizzi alla memoria***
  (si seleziona una specifica cella su cui viene eseguita o un'operazione di lettura o una operazione di scrittura)

BUS DI SISTEMA

BUS COMANDI

- **bidirezionale**
- tipicamente usato per ***inviare comandi verso la memoria*** (es: lettura o scrittura) o ***verso una periferica*** (es. stampa verso la stampante → interfaccia)
- può essere usato per ***inviare comandi verso il processore*** nel caso di DMA (o interfacce di I/O)

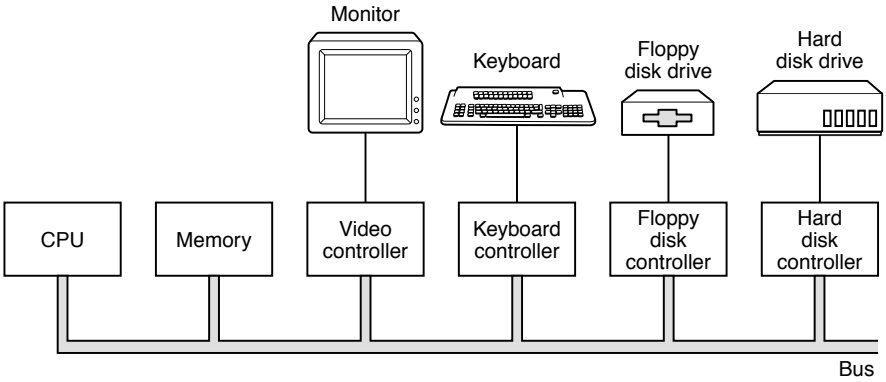
INTERFACCE DI I/O



Le interfacce sono molto diverse tra loro, e dipendono dal tipo di unità periferica da connettere.

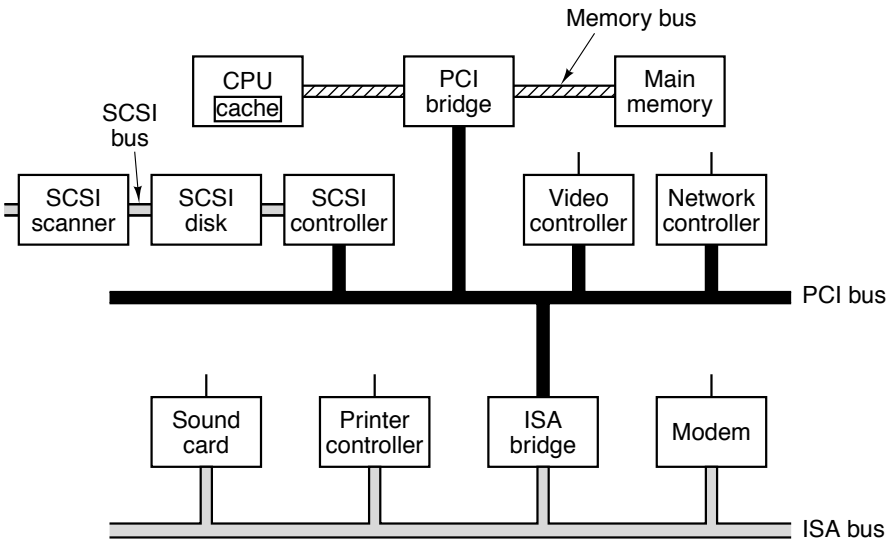
Una **interfaccia** è un dispositivo che consente all'elaboratore di **comunicare con una periferica** (tastiere, mouse, dischi, terminali, stampanti, ...).

Struttura Logica di un Semplice Personal Computer



- Logicamente, il **bus** collega la CPU alla memoria e ai controllori dei dispositivi di Input-Output (**periferiche**).
- Alcuni esempi di periferiche sono: il monitor, la tastiera, il mouse, modem, stampanti, schede di rete.
- Notate che alcuni dispositivi sono più lenti di altri!
  - Se tutti i dispositivi sono su un solo bus, quelli lenti possono bloccare quelli veloci.
  - Quindi, è opportuno avere più bus integrati in un'unica scheda madre.
  - E.g., un bus per i dispositivi lenti, uno per i dispositivi veloci.

Tipico PC Odierno



- **PCI**, **ISA** e **Memory bus** sono tre tipi di bus.
  - Il PCI, più recente, è più veloce del bus ISA.
  - Il memory bus è il più veloce dei tre.
  - I bus sono collegati tra loro con dei *bridge* (PCI/ISA bridge).