



Architettura dei sistemi di elaborazione

 Materials

Introduzione

Il calcolatore digitale è una macchina che svolte dei compiti eseguendo delle istruzioni che gli vengono fornite dall'utente; l'insieme di queste istruzioni prende il nome di **programma**.

I circuiti elettronici dei computer sono in grado di comprendere ed eseguire soltanto alcune istruzioni base (sommare due numeri, riconoscere se un numero è uguale o diverso da 0, spostare dei dati da una parte all'altra della memoria) che prendono il nome di **linguaggio macchina**, in cui tutti i programmi devono essere convertiti per essere utilizzati.

Da qui la necessità di creare un nuovo insieme di istruzioni che sia più comodo da utilizzare rispetto al linguaggio macchia.

Approccio strutturale

Questa necessità di dover creare più insiemi di istruzioni prende il nome di **approccio strutturale**. Globalmente queste nuove istruzioni definiscono un **linguaggio**, che chiameremo **L1**, mentre indicheremo con **L0** il **linguaggio macchina**.

Tenendo sempre in mente che il computer è in grado di riconoscere ed eseguire soltanto istruzioni in **L0**, il problema che affrontiamo con l'approccio strutturale è quello di convertire istruzioni **L1** in istruzioni **L0**. Questa conversione può essere affrontata in due modi distinti:

- **Traduzione:** Le istruzioni del programma **L1** vengono convertite in istruzioni analoghe **L0**. Il programma risultante avrà solo istruzioni **L0**; di conseguenza il

computer salverà in memoria il nuovo programma che verrà usato al posto di quello **L1**.

- **Interpretazione:** Viene creato un programma **L0** che accetta in ingresso istruzioni **L1**. In questo caso le istruzioni **L1** non sono altro che dati che vengono interpretati e sostituiti con l'equivalente in **L0**. Il computer avrà quindi il controllo soltanto del programma **interprete**.

Per rendere la traduzione e l'interpretazione realmente efficaci, è necessario che il linguaggio **L1** non sia troppo diverso dal linguaggio **L0**. Questo, però, fa sì che il linguaggio **L1**, seppur superiore e più facile da usare del linguaggio macchina, è ancora lontano dall'essere un linguaggio ideale.

La soluzione più ovvia è quella di creare un linguaggio **L2** ancora più rivolto agli utenti, che verrà poi tradotto in o interpretato in **L1**. Questa ricerca di un linguaggio sempre più indirizzato agli utenti può continuare in modo indefinito, finché non si arrivi ad un linguaggio realmente ideale. Ciò ci porta a pensare al computer come ad una struttura a **livelli, o strati**.

Macchine multilivello

Gli attuali computer sono formati da due o più livelli. Il livello 0 rappresenta il vero e proprio hardware della macchina, i cui circuiti eseguono programmi scritti nel linguaggio macchina del livello 1.

Uno degli elementi più importanti del livello 0, detto anche **livello logico digitale**, sono le **porte** (che sono costruite con componenti analogici come i **transistor**). Ogni **porta** è in grado di ricevere uno o più **input digitali** (ovvero segnali **0** e **1**) e calcola in output una semplice funzione dei valori in ingresso, come una funzione **AND** o **OR**.

È possibile combinare un piccolo numero di porte per creare una **memoria da 1bit**.

Mettendo insieme più memorie, in gruppi da **16, 32 o 64**, si vengono a creare quelli che vengono definiti **registri**. I **registri** sono delle memorie molto piccole in grado di memorizzare un valore fino ad un massimo numero di cifre digitali.

Subito dopo troviamo il **livello di micro-architettura, livello 1**. Qui troviamo una memoria locale, formata da un gruppo di registri (di solito da 8 o 32), e un circuito chiamato **ALU** (Arithmetic Logic Unit, Unità Aritmetico Logica) capace di effettuare delle semplici operazioni aritmetiche. I registri sono collegati alla **ALU** formando un **percorso dati** che permette di selezionare uno o più registri, permettere alla **ALU** di effettuare

delle operazioni su di loro (ad esempio sommandoli), e registrare il risultato su uno dei registri.

- In alcune macchine questo passaggio avviene tramite un **microprogramma**, mentre in altre avviene direttamente a livello hardware. Nelle macchine in cui questo passaggio avviene a livello software, il **microprogramma** non è altro che un interprete del linguaggio macchina del livello 2 che sfrutta il percorso dati per analizzare le istruzioni ed eseguirle una alla volta.

Il livello 2 viene comunemente chiamato il **livello ISA** (Architettura dell'insieme delle istruzioni). In questo livello troviamo il **sistema operativo**, che non è altro che un'interprete delle istruzioni del livello 3.

Il livello 3, che prende il nome di **livello macchina del sistema operativo**, è solitamente un livello ibrido. Condivide alcune istruzioni con il secondo livello, **Livello ISA**, mentre aggiunge nuove istruzioni che vengono interpretate dal **sistema operativo**. Le funzioni in comune tra **livello ISA** e **livello 3** vengono eseguite direttamente dal microprogramma (o dai circuiti elettronici), mentre le nuove istruzioni vengono interpretate dal sistema operativo.

I livelli che abbiamo visto fino ad ora sono pensati per eseguire interpreti e traduttori di supporto per i livelli più alti (questi traduttori e interpreti vengono scritti da professionisti chiamati **programmatori di sistema**). Mentre a partire dal livello 4 iniziamo a trovare linguaggi utilizzati per risolvere problemi applicativi. Un altro cambiamento è che, mentre i livelli sottostanti erano sempre interpretati, a partire dal livello 4 i linguaggi che vengono utilizzati sono convertiti mediante traduzione.

Il livello 4, **livello del linguaggio assemblativo**, consente ai programmatori un modo per scrivere programmi per i livelli 1, 2 e 3 in una forma più “facile” rispetto alla scrittura nel linguaggio dei livelli precedenti. Questo **linguaggio assemblativo** viene prima tradotto (il traduttore prende il nome di **assemblatore**) nel linguaggio dei livelli sottostanti e poi interpretato dai relativi microprogrammi o dalla componente hardware.

Il livello 5 è formato da tutti quei **linguaggi ad alto livello** che vengono usati per programmi applicativi (C, C++, Java, Python..). Questi linguaggi vengono generalmente tradotti al livello 3 o 4 da un traduttore chiamato **compilatore**.

Riassumendo, il concetto chiave da ricordare è che il computer è una macchina costruita a livelli. Ogni livello viene costruito a partire da quello che lo precede ed ogni livello è caratterizzato dalla presenza di oggetti o operazioni differenti.

- Per **Hardware** intendiamo gli oggetti tangibili del computer: circuiti, cavi, trasformatori, memorie..). Per **Software** intendiamo gli **algoritmi**, ovvero le istruzioni dettagliate su come eseguire un determinato compito, e la loro rappresentazione per i computer, ovvero i **programmi**.

Nei primissimi computer il confine tra questi due mondi era chiaro, ma nel corso del tempo il confine si è sfocato per via dell'aggiunta, della rimozione e dell'unione di livelli, tanto che oggi è difficile separare questi due mondi.

Hardware e software sono logicamente equivalenti.

Generazione zero - Computer meccanici (1642 - 1945)

- Computer meccanici in grado di eseguire operazioni aritmetiche di base.
- **1944 - Mark I** (Howard Aiken): Calcolatore composto da 72 parole e 23 numeri decimali.

Prima generazione - Valvole (1945-1955)

- **1945 - Enigma**: Macchina codificatrice/decodificatrice di messaggi.
- **Colossus**: Primo elaboratore digitale
- **1946 - Eniac**: Electronic Numerical Integrator And Computer. Composto da 18000 valvole termoioniche, 1500 relé e 20 registri. Ogni registro era in grado di memorizzare un numero decimale a 10 cifre.
- **Macchina di von Neumann (Macchina IAS)**: 4 Componenti principali:

- La memoria da 4096 locazioni, ogni locazione conteneva 40 bit. Ogni parola poteva occupare due istruzioni da 20 bit oppure un numero da 40 bit. Le istruzioni erano composte da 8 bit, che definivano il tipo di istruzione, e dai restanti bit che specificavano una delle 4096 parole di memoria.
- Unità aritmetico-logica e l'unità di controllo erano il cervello del computer. Nei computer moderni sono fusi in un unico chip, la **CPU**. Nell'unità aritmetico-logica troviamo uno speciale registro da 40 bit chiamato **accumulatore**. La tipica istruzione sommava una parola di memoria all'accumulatore oppure ne copiava il contenuto in memoria.
- Unità di controllo
- Dispositivi di input ed output

Seconda generazione - Transistor (1955-1965)

- **TX-0**: primo computer a transistor
- **PDP-1**: 4096 parole di memoria da 18 bit e poteva eseguire circa 200.000 istruzioni al secondo. Era dotato di un display visuale in grado di disegnare punti in qualsiasi parte dello schermo da 512x512 px.
- **PDP-8**: Per la prima volta un unico bus (omnibus). Un **bus** è un insieme di cavi paralleli utilizzato per connettere i diversi componenti del computer.
- **CDC 6600**: CPU dotata di varie unità funzionali che potevano lavorare contemporaneamente (operazioni matematiche). Si potevano eseguire circa 10 istruzioni contemporaneamente.

Terza generazione - Circuiti integrati (1965-1980)

- L'invenzione dei circuiti integrati consente di realizzare su un unico chip decine di transistor.
- Computer più piccoli, veloci ed economici.
- **IBM System/360**:
 - Famiglia di computer dotati dello stesso linguaggio assemblativo, con dimensioni e potenza delle macchine crescente.

- Il software scritto per un modello “base” funzionava anche sul modello più grande, ma non valeva il contrario: spostando un programma per un computer più grande su uno più piccolo, poteva non bastare la memoria a disposizione.
- **Multiprogrammazione**: avere più programmi in memoria nello stesso tempo, mentre si è in attesa di completare un’operazione si possono eseguire calcoli.
- 2^{24} e 2^{32} byte.

Quarta generazione - Integrazione a grandissima scala (1980-?)

- Tecnologia **VLSI (Very Large Scale Integration)**:
 - Si passa da decina di migliaia, a centinaia di migliaia, a milioni di transistor stampati in un unico chip.
 - Computer più piccoli e più veloci
- Inizio dell’era dei **Personal Computer**:
 - Era compito dell’acquirente montare il computer
 - Il software non era incluso, qualsiasi programma si voleva eseguire andava scritto
 - **Intel 8080**
- **CP/M**: Primo vero sistema operativo con un file system e uno **shell**, un interprete di comandi in grado di eseguire una serie di istruzioni
- **Apple I e Apple II**: Primi personal computer più diffusi
- **PC IBM**:
 - Integra il nuovo processore **Intel 8088**
 - Sistema operativo **Microsoft MS/DOS**
- **Macintosh (1984)**:
 - Primo personal computer dotato di **GUI (Graphical User Interface)**
- **Osborne-1**: Primo vero personal computer portatile (peso di 11kg)
- Prime versioni di **Windows**

- **Intel 80386 e 80486:**
 - Processori a 32 bit
 - Le versioni successive diventeranno rispettivamente **Intel Pentium** ed **Intel Core**
 - Architettura x86
- **Progettazione RISC:**
 - Architetture molto più semplici e veloci
 - In grado di eseguire più istruzioni contemporaneamente
- **Circuito FPGA:**
 - Circuito integrato con porte logiche che potevano essere “programmate” per creare sistemi informatici specializzati per applicazioni uniche al servizio di un ristretto numero di utenti (prototipazione, applicazioni di progettazione, istruzione)
- **1992 Prima macchina a 64 bit di tipo RISC**
- **Fine anni 90:**
 - Stallo nella ottimizzazione dei chip: non si riusciva a rendere i programmi più veloci e raffreddare i processori era troppo costoso
 - **Nascono le prime macchine dual core:**
 - **IBM dual-core Power4:** prima macchina con due processori su un unico chip

Quinta generazione - Computer a basso consumo e computer invisibili (1989-?)

- **PDA (Personal Digital Assistant):**
 - **GridPad ed Apple Newton:**
 - Piccolo schermo dove gli utenti potevano scrivere con una speciale penna.
 - L’evoluzione di questi dispositivi porta agli smartphone
 - **Primi anni 90 - IBM Simon:**

- Primo telefono cellulare touchscreen dotato di PDA
- **Computer Invisibili:**
 - Computer integrati in elettrodomestici, orologi, carte di credito..
 - In futuro i computer saranno integrati in qualsiasi oggetto della vita quotidiana e per questo saranno invisibili.

Famiglie di computer

Architettura x86

Con il nome **Architettura x86** si intende l'**architettura di set di istruzioni (ISA)** introdotta da **Intel** con il processore **8086**, processore a 16 bit. I processori successivi erano tutti retrocompatibili con il set di istruzioni originale utilizzato nel processore **8086**, e poiché tutti i processori terminavano con il numero 86, l'**ISA** prende il nome di **x86**. Con l'introduzione del processore **80386**, il set di istruzioni **x86** viene esteso ad un sistema a 32 bit. Da qui nasceranno tutti i processori **Pentium** (Pentium I, II, III..), anche loro con architettura a 32 bit.

A questo set di istruzioni vengono introdotte, nel tempo, nuove istruzioni speciali tra cui **MMX** e **SSE**. Le istruzioni **MMX** (MultiMedia eXtension) servivano per migliorare i calcoli necessari per l'elaborazione audio e video, rendendo superflua l'esistenza di coprocessori dedicati alle attività multimediali. Le istruzioni **SSE** vanno ad ampliare l'**MMX** migliorando tutto ciò che è inherente alla grafica 3D.

All'architettura x86 seguirà l'architettura **x64**, ovvero il set di istruzioni che viene esteso alle macchine a 64 bit.

Architettura ARM

L'**architettura ARM** indica una famiglia di microprocessori a 32-64 bit che ha avuto grande successo nel mercato dei dispositivi mobili, a basso consumo e integrati. I processori con architettura ARM godono di alta velocità e bassissimi consumi, impiegati principalmente in dispositivi come i **PDA**.

Architettura AVR

L'architettura AVR viene utilizzata in sistemi integrati di fascia bassa (microcontrollori). Oggi l'interesse maggiore per questo tipo di architettura è dovuto alla sua presenza nella piattaforma Arduino.

L'architettura AVR viene realizzata in tre classi.

La classe più bassa, **tinyAVR**, viene progettata per le applicazioni con maggiori vincoli in termini di spazio, potenza e costi. Essa comprende una CPU a 8 bit, il supporto di base per I/O digitale e il supporto per un ingresso analogico (per esempio, la lettura della temperatura di un termostato).

Il **megaAVR** (presente in Arduino), è dotato anche del supporto I/O seriale, di orologi interni e di uscite analogiche programmabili.

Il **AVR XMEGA** incorpora anche un acceleratore per le operazioni di crittografia e il supporto integrato per interfacce USB.

Ogni classe ci processori AVR include generalmente 3 tipi di memoria: **flash**, **EEPROM** e **RAM**.

La memoria flash è programmabile con l'ausilio di un'interfaccia esterna ed è qui dove il codice del programma ed i dati vengono memorizzati. Questo tipo di memoria non è volatile, ovvero i dati in essa contenuti vengono mantenuti anche quando il sistema viene spento. Ance la memoria EEPROM non è volatile, ma a differenza della flash, può essere modificata da I programma durante l'esecuzione. In questa memoria esiste un sistema integrato che mantiene le informazioni di configurazione dell'utente (ad esempio orario visualizzato in 12 o 24 ore).

Infine, la **RAM** è la memoria in cui sono mantenute le variabili del programma in esecuzione. Questa memoria è di tipo volatile, quindi i dati contenuti in essa vengono persi quando si toglie alimentazione al sistema.

Unità metriche

Quando parliamo delle dimensioni di **memorie**, dischi, file, database.. le loro dimensioni sono sempre **potenze di 2**. 1 KB contiene 1024 e non 1000 byte ($2^{10} = 1024$), analogamente 1 MB contiene 2^{20} byte, 1 GB 2^{30} .

Quando parliamo invece di velocità di trasferimento dati, parliamo di **Kbps (10^3)**, **Mbps**, **Gbps**.. e in questo caso parliamo di potenze di 10. 1 Kbps trasmette 1000 bit al secondo, una Lan a 10 Mbps trasmette 10.000.000 bit/s.

Prefisso	Valore	Prefisso	Valore
KB	2^{10}	Kbps	10^3

MB	2^{20}	Mbps	10^6
GB	2^{30}	Gbps	10^9
TB	2^{40}	Tbps	10^{12}

Processori

La **CPU (Central Processing Unit)** è il “cervello” del computer e la sua funzione è quella di eseguire i programmi contenuti nella memoria principale prelevando le loro istruzioni ed eseguendole una dopo l’altra. I componenti sono collegati tra loro mediante un **bus** (insieme di cavi paralleli); i **bus** possono essere esterni alla **CPU**, per connetterla alla memoria e ad altri dispositivi, oppure interni.

Tra le parti della quale si compone la **CPU** troviamo l'**unità di controllo** e l'**unità aritmetico logica**. La prima si occupa di prelevare le istruzioni dalla memoria e di determinarne il tipo, mentre la seconda esegue le operazioni necessarie (ad esempio l’addizione e l’AND) per eseguire le istruzioni.

La **CPU** contiene anche una piccola memoria ad alta velocità costituita da **registri**. Questa memoria serve per memorizzare i risultati temporanei e alcune informazioni di controllo. Solitamente i registri che compongono questa memoria hanno una funzione e una dimensione predefinite; ognuno di loro può contenere un numero, il cui valore può variare fino ad un massimo che dipende dalla dimensione del registro (dimensione solitamente uguale per tutti i registri).

I **registri** più importanti sono:

- **Program Counter**: “punta” alla successiva istruzione da prelevare per l’esecuzione
- **Instruction Counter** o **Registro istruzione corrente**: contiene l’istruzione attualmente in esecuzione.

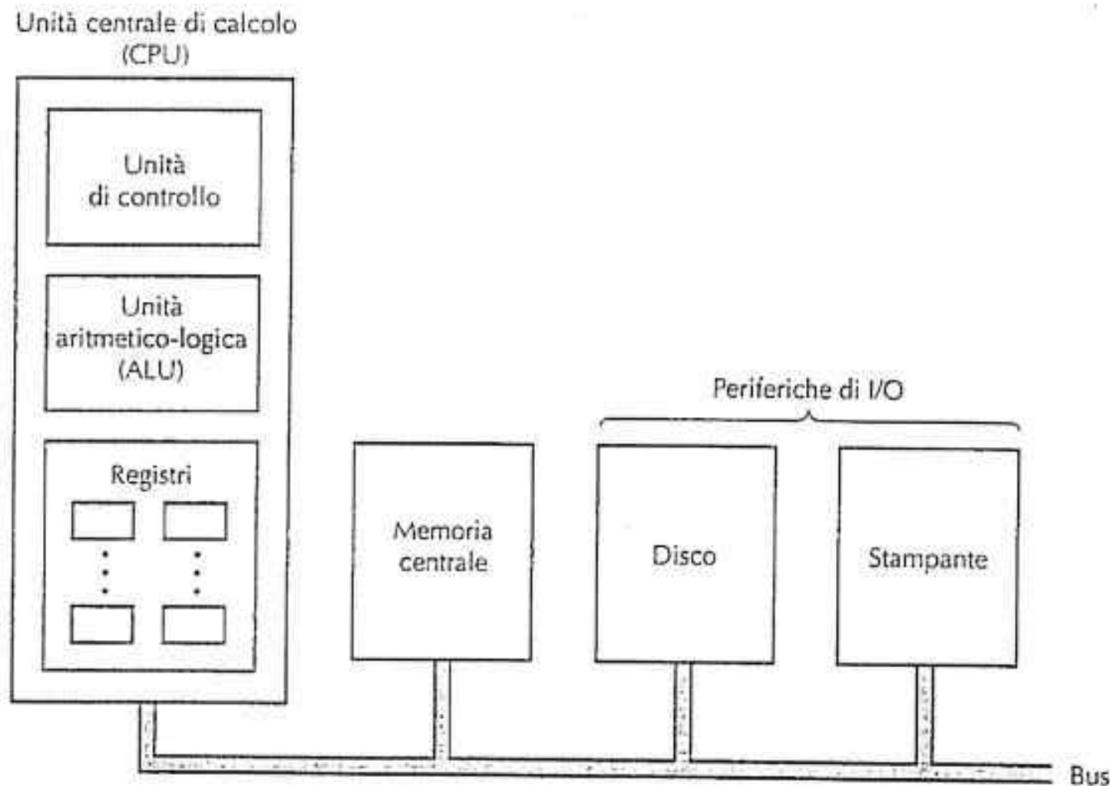


Figura 2.1 Organizzazione di un semplice calcolatore con una CPU e due periferiche di I/O.

Organizzazione della CPU

Il **percorso dati (datapath)** di una tipica CPU di **von Neumann** è composto dai **registri** e dalla **ALU** (unità logico aritmetica) collegati tra loro tramite bus. I registri della CPU sono collegati ai due registri input della ALU (indicati in figura con le lettere A e B) che, mentre la ALU è occupata nell'esecuzione di alcune computazioni, mantengono i dati in ingresso finché non posso essere scaricati nell'ALU.

La ALU esegue semplici operazioni sui suoi input (come addizioni e sottrazioni) ed il risultato generato viene memorizzato nell'apposito registro di output. Volendo, questo risultato può essere riportato in uno dei registri della CPU per essere, successivamente, salvato in memoria.

La maggior parte delle **istruzioni del percorso dati** si dividono in due principali categorie:

- Istruzioni **registro-memoria**: non fanno altro che mettere in comunicazione i registri della CPU con la memoria. Possiamo prelevare “parole di memoria” per

portarle nei registri, dove sono poi utilizzabili nell'alu, oppure possiamo copiare valori di registri nella memoria.

- Istruzioni **registro-registro**: mettono in comunicazione i vari registri. Una tipica istruzione è prelevare due operandi dai registri della CPU per essere portati nei registri di input dell'ALU.

■ Il processo che consiste nel portare i due operandi attraverso la ALU prende il nome di **ciclo del percorso dati** e rappresenta il cuore della maggior parte delle CPU. I computer moderni dispongono di più ALU che operano in parallelo, specializzate su funzioni diverse.

Più veloce è il ciclo del percorso dati, maggiore sarà la velocità del calcolatore.

Il modo in cui le **CPU** eseguono le istruzioni prende il nome di **ciclo esecutivo delle istruzioni (fetch-decode-execute)**:

- Viene prelevata la successiva istruzione dalla memoria e viene portata nell'IR
- Viene modificato il PC per farlo puntare alla prossima istruzione
- Viene determinato il tipo di istruzione
- Se l'istruzione usa una parola di memoria, viene localizzata
- Se necessario, viene prelevata la parola e portata in un registro della CPU
- Esegue l'istruzione

■ Questo procedimento può essere eseguito anche a livello software con un programma **interprete**.

■ Per **architettura** si intende una famiglia di macchine con un set base di istruzioni comune.

RISC e CISC

RISC: computer con un insieme ridotto di istruzioni (Reduced Instruction Set Computer)

CISC: computer con un insieme di istruzioni complesso (Complex Instruction Set Computer)

I processori **RISC** offrono un numero relativamente basso di istruzioni (es. 50), al contrario dei processori **CISC** (es. 200-300). Inoltre, le istruzioni dei processori **RISC** sono semplici e vengono eseguite direttamente in un ciclo di **percorso dati**; anche se i computer **RISC** impiegano 4-5 istruzioni per fare ciò che un computer **CISC** esegue con una sola istruzione, comunque risultano più veloci in quanto non interpretate.

Principi di progettazione dei calcolatori moderni

Esiste un insieme di principi di progettazione, chiamati **principi di progettazione RISC** che vengono sempre seguiti:

- **Tutte le istruzioni vengono eseguite direttamente dall'hardware:** saltando il livello di interpretazione la velocità di esecuzione delle istruzioni aumenta.
- **Massimizzare la frequenza di emissione delle istruzioni:** e questo è possibile solo se sono anche in grado di eseguire più istruzioni contemporaneamente.
- **Le istruzioni devono essere facili da decodificare:** emettendo milioni di istruzioni in un secondo, non risolvo niente se la macchina impiega molto a decodificare ogni singola istruzione.
- **Solo le istruzioni LOAD e STORE fanno riferimento alla memoria:** dato che l'accesso alla memoria può richiedere un tempo considerevole e non prevedibile, le operazioni di memoria devono essere eseguite sovrapposte ad altre istruzioni (che operano sui registri).
- **Disporre di molti registri:** dato che l'accesso alla memoria è lento, occorre avere molti registri (almeno 32) in modo che la parola di memoria possa essere mantenuta in un registro tutto il tempo necessario. Bisogna ridurre al minimo il rischio di restare senza registri liberi.

Parallelismo: più istruzioni nell'unità di tempo

Poiché l'incremento del clock del processore ha raggiunto un limite fisico, i progettisti di CPU guardano al **parallelismo (più istruzioni nello stesso tempo)** per incrementare le performance.

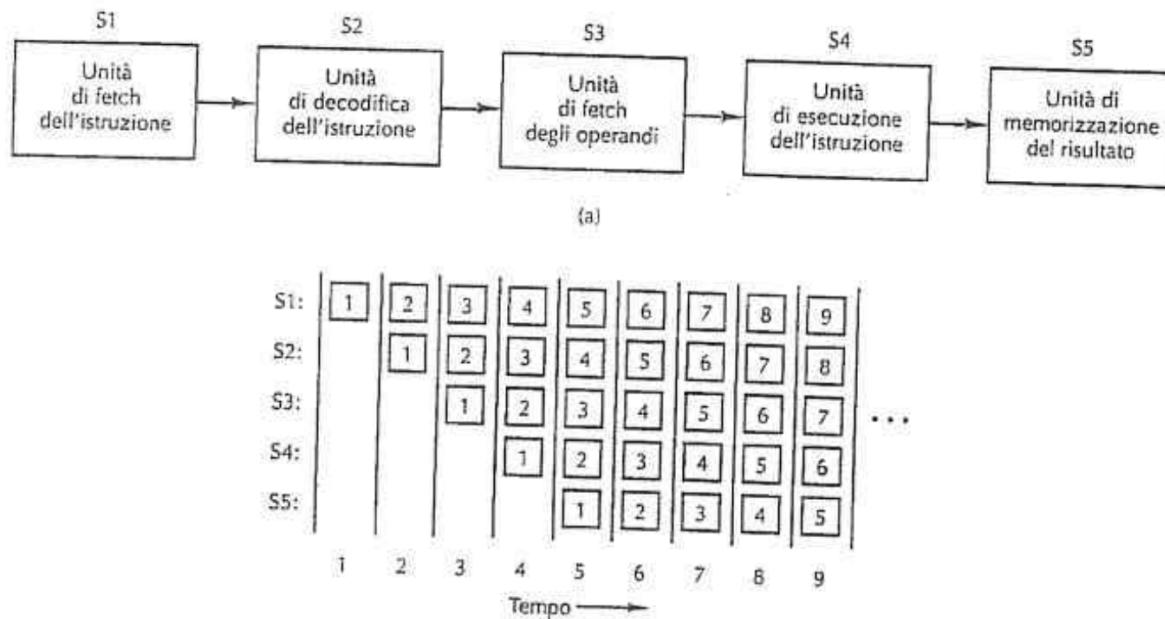
Il parallelismo può essere presente in due forme:

- A **livello d'istruzione** in cui il parallelismo viene sfruttato all'interno delle singole istruzioni in modo da elaborarne il più possibile al secondo.
- A **livello di processore** in cui ci sono più CPU che lavorano insieme su uno stesso problema.

Pipelining e architetture superscalari (parallelismo a livello d'istruzione)

Uno dei maggiori problemi nella velocità di esecuzione delle istruzioni è rappresentato dal prelievo delle istruzioni dalla memoria. Per aggirare questo problema si fa uso del concetto di **pipelining**; le istruzioni vengono divise in un determinato numero di parti che possono essere eseguite in parallelo (ogni parte viene gestita da una componente hardware dedicata).

Presumiamo di avere una **pipeline** a cinque stadi:



Durante il primo ciclo di clock lo stadio **S1** preleva l'istruzione **1** dalla memoria. Durante il secondo ciclo di clock, lo stadio **S2** decodifica l'istruzione **1** mentre lo stadio **S1** preleva l'istruzione **2** dalla memoria. Nel ciclo di clock successivo, lo stadio **S3** preleva gli operandi per l'istruzione **1**, mentre lo stadio **S2** decodifica l'istruzione **2** e lo stadio **S1** preleva l'istruzione **3**. Successivamente lo stadio **S4** esegue l'istruzione ed infine, al

quinto ciclo di clock, lo stadio **S5** scrive il risultato dell'istruzione.

L'uso della **pipeline** ci consente di bilanciare la **latenza** (il tempo necessario per elaborare un'istruzione) e la **larghezza di banda del processore** (I MIPS della CPU).

- Tuttavia non tutte le istruzioni possono essere eseguite contemporaneamente, perché può succedere che un'istruzione sia dipendente dal risultato dell'altra. Quindi o è il compilatore a gestire questa situazione (l'hardware non effettua nessun controllo e se le istruzioni sono incompatibili restituisce un risultato errato) oppure i conflitti vengono rilevati ed eliminati durante l'esecuzione da componenti hardware ad hoc.

- Anche se le pipeline sono principalmente usate su macchine RISC, Intel inizia ad usare delle pipeline nelle sue CPU a partire dal x486. Il 486 aveva una sola pipeline, mentre il primo Pentium ne aveva due. La pipeline principale, chiamata **pipeline u**, poteva eseguire una qualsiasi istruzione Pentium; la pipeline secondaria **pipeline v**, eseguiva solamente semplici istruzioni su interi.

Con il termine **architettura superscalare** si intende una CPU con singola pipeline ma con più unità funzionali. Le architetture superscalari moderne sono in grado di lanciare più istruzioni durante un solo ciclo di clock (solitamente 4 o 6).

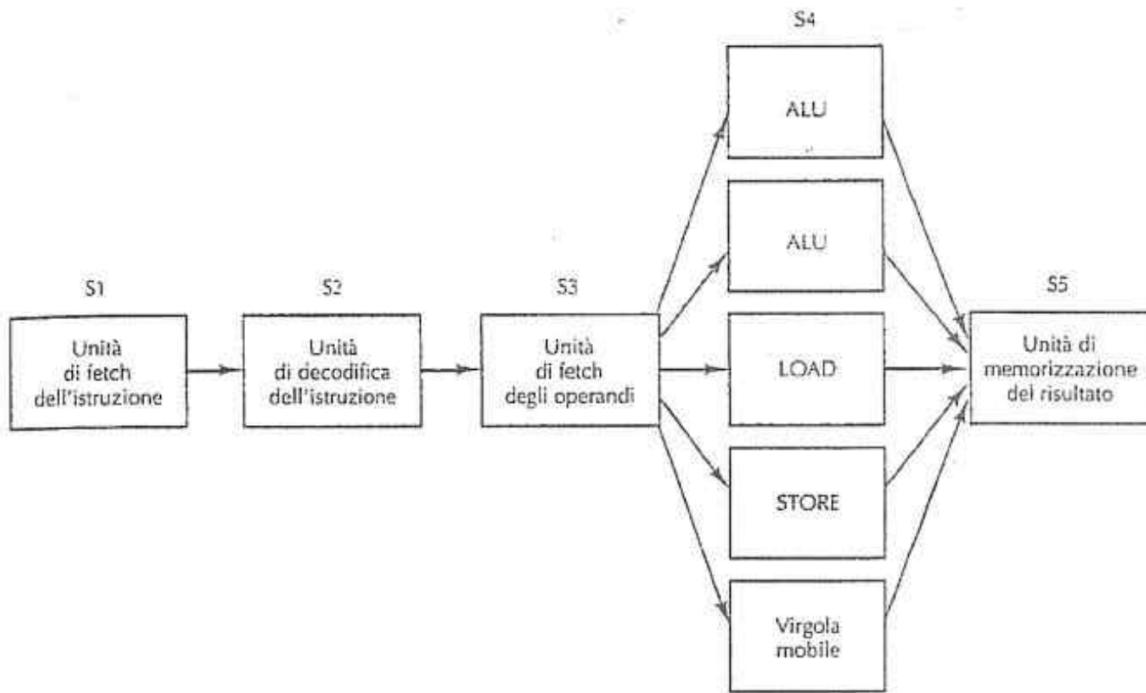


Figura 2.6 Processore superscalare con cinque unità funzionali.

Parallelismo a livello di processore

Per aumentare ancora di più le performance delle macchine occorre progettare sistemi dotati di più CPU. I tre approcci principali a questo tipo di progettazione sono:

- **Computer con parallelismo sui dati**
- **Multiprocessori**
- **Multicomputer**

Computer con parallelismo sui dati

Questo tipo di progettazione risulta molto utile quando abbiamo a che fare con problemi dalla struttura regolare, in cui gli stessi calcoli vengono ripetuti continuamente su insiemi diversi di dati.

Per eseguire questi programmi altamente regolari vengono usati, solitamente, i **processori SIMD**. Questo tipo di processore consiste di un elevato numero di processori identici che eseguono la stessa sequenza di istruzioni su insiemi diversi di dati.

Il modello SIMD è composto da un'unica unità di controllo che esegue una istruzione

alla volta controllando più **ALU** che operano in maniera sincrona. Ad ogni passo, tutti gli elementi eseguono la stessa istruzione scalare, ma ciascuno su un dato differente.

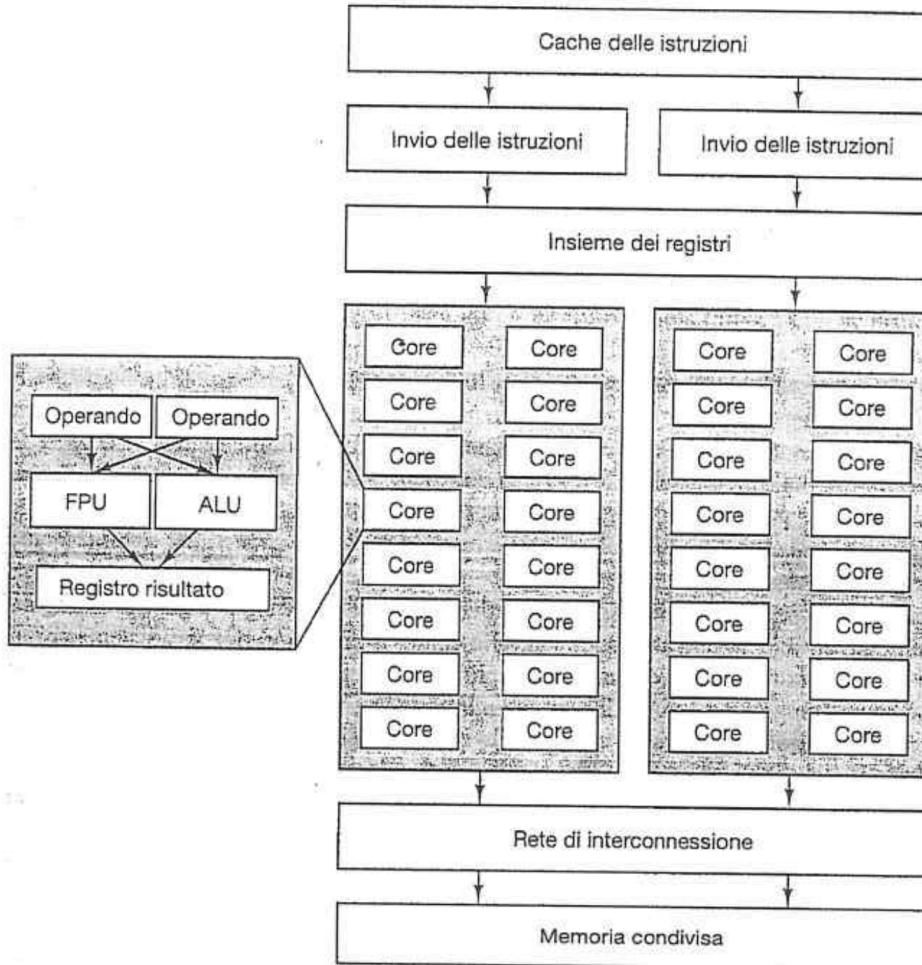


Figura 2.7 Il processore SIMD della GPU Fermi.

- Le moderne **GPU** fanno uso dell'elaborazione SIMD dato che la maggior parte degli algoritmi sono altamente regolari, con operazioni ripetute su pixel, vertici, texture..

- **Thread o processo?** Un **processo** è un programma in **esecuzione**. Un **Thread** è l'insieme di **istruzioni** che caratterizzano il flusso di **esecuzione** di un **processo**.

Multiprocessori

A differenza dei processori SIMD, che hanno un'unica unità di controllo, i **multiprocessori** sono sistemi formati da più CPU distinte, collegate tra loro con un singolo bus, e tutte connesse con una memoria in comune.

Dato che ogni CPU può leggere e scrivere una qualsiasi parte della memoria, esse devono coordinarsi via software per evitare di ostacolarsi. Per evitare problemi, si usa un'architettura in cui ogni processore possiede una memoria locale utilizzabile per contenere il codice del programma e quei dati che non devono essere condivisi.

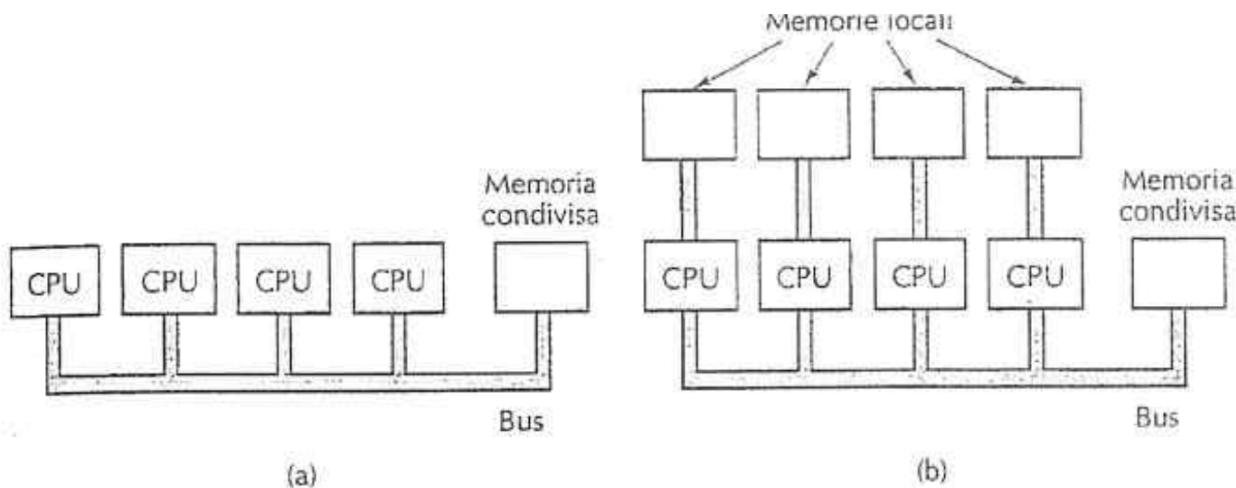


Figura 2.8 (a) Multiprocessore a bus singolo. (b) Multicomputer con memorie locali.

Multicomputer

Nei **multicomputer** si abbandona l'idea di avere una memoria comune a favore di un sistema composto da un gran numero di calcolatori interconnessi, ognuno con una memoria privata, che comunicano fra loro inviandosi dei messaggi.

Memoria principale

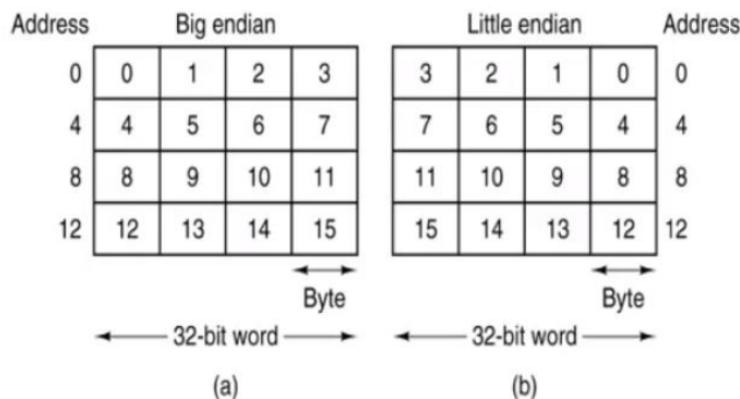
La **memoria** è quella parte del calcolatore in cui sono depositati programmi e dati. La sua unità base è il **bit**, o unità binaria, che può valere 0 oppure 1. Le informazioni vengono memorizzate in un certo numero di **celle**, o **locazioni**; ogni cella viene indicata con un numero, chiamato **indirizzo**, attraverso il quale può essere richiamata dal programma.

- Se una memoria ha n celle, i suoi indirizzi varieranno da 0 a $n - 1$.

Tutte le celle di memoria contengono lo stesso numero di bit, che negli ultimi anni è stato standardizzato ad **8 bit**. Questa dimensione viene denominata **byte** (quindi **1 byte = 8 bit**), ed i **byte** vengono raggruppati in **parole**.

Quindi un calcolatore con parole a 32 bit, avrà 4 byte per parola; un calcolatore con parole a 64 bit, avrà 8 byte per parola.

All'interno di una parola, i **byte** possono essere scritti da destra verso sinistra (*little endian*) o da sinistra verso destra (*big endian*).



Codici di correzione di errore

Al fine di correggere e rilevare eventuali errori commessi dal computer si fa uso del **codice di Hamming**, che va ad aggiungere dei **bit di controllo** in mezzo al **codeword** in base ad un algoritmo prestabilito.

- Per **codeword** (o **parola di codice**) intendiamo una parola, indicata con n , che contiene m **bit** di dati + r **bit ridondanti**. ($n = m + r$)

Per di **distanza di Hamming** intendiamo il numero di bit per la quale due parole differiscono.

$$D(1000,1010) = 1$$

$$D(0110,0000) = 2$$

$$D(0100,1010) = 3$$

Ed è importante ricordare che la **distanza minima di Hamming** corrisponde sempre a $K + 1$ (K inteso come lunghezza del bit)

Immaginiamo di avere un codice formato da 7 bit, 0000101, di quanti bit di controllo avrò bisogno? Per scoprirllo, ci basta verificare questa disegualanza: bit di messaggio + 1 $\leq 2^x - x$. In cui ovviamente x = bit di controllo. Nel nostro caso avremo bisogno di 4 bit di controllo, dato che $2^4 = 16$ allora: $7 + 1 \leq 16 - 4$ è verificata.

Di conseguenza il nostro messaggio finale sarà composto dal messaggio originale da 8 bit + 4 bit di controllo (12 bit)

Dimensione parola	Bit di controllo	Dimensione totale	Percentuale di overhead
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2

Figura 2.13 Numero di bit di controllo per un codice che può correggere errori singoli.

Detto ciò, i bit di controllo vanno inseriti nelle posizioni della potenza del 2; dobbiamo mettere i bit di controllo nelle posizioni di $2^0, 2^1, 2^2, 2^3$.

Quindi, dato $2^0 = 1$, noi andiamo ad inserire il primo bit di controllo in posizione 1; $2^1 = 2$, quindi secondo bit di controllo in posizione 2; $2^2 = 4$, terzo bit di controllo in posizione 4; $2^3 = 8$, quarto bit di controllo in posizione 8.

Ipotizziamo di voler trasmettere il messaggio 00101101, Il nostro messaggio di partenza diventerà così: (* = bit di controllo)

bit 1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12
*	*	0	*	0	1	0	*	1	1	0	1
pos 1	2	3	4	5	6	7	8	9	10	11	12

Per “trovare” il valore da inserire nei primo bit di controllo, dobbiamo prima capire quali bit controlla ogni bit di controllo, e per questo dobbiamo rifarci sempre alle potenze del 2. Abbiamo detto che il primo bit è $2^0 = 1$ e questo vuol dire che il primo bit controlla un bit sì ed un bit no: ovvero controlla i bit 1,3,5,7,9,11.

A seguire, il secondo bit di controllo è $2^1 = 2$ e controlla due bit sì e due bit no (da ricordare che il conteggio dei bit parte sempre dal numero del bit stesso, in questo caso partiamo quindi dal bit di controllo in seconda posizione): 2,3,6,7,10,11.

- 2^0 un bit sì e uno no: 1,3,5,7,9,11
- 2^1 due bit sì e due no: 2,3,6,7,10,11
- 2^2 quattro bit sì e quattro no: 4,5,6,7,12
- 2^3 otto bit sì e otto no: 8,9,10,11,12

Ora che abbiamo capito quali bit controlla ogni bit di controllo, possiamo procedere ad inserire il valore corretto. Per fare ciò utilizziamo il **bit di parità**, ovvero andiamo a vedere quanti 1 ci sono in ogni stringa di bit di controllo: se il numero di 1 presenti è pari, inseriamo uno 0, se il numero di 1 è dispari inseriamo un 1 (inseriamo un 1 perché così facendo facciamo tornare la conta dei numeri 1 pari, es: abbiamo 5 volte il valore 1, quindi aggiungiamo il bit di parità 1 per arrivare a 6.)

bit 1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12
*	*	0	*	0	1	0	*	1	1	0	1
pos 1	2	3	4	5	6	7	8	9	10	11	12

Rivedendo il codice, procediamo con il calcolo:

- Il primo bit di controllo è formato da: 00010. Il valore 1 compare solo una volta, quindi aggiungiamo il bit di parità 1.

bit 1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12
1	*	0	*	0	1	0	*	1	1	0	1
pos 1	2	3	4	5	6	7	8	9	10	11	12

- Il secondo bit di controllo è formato da: 01010. Il valore 1 appare due volte, quindi il bit b2 sarà 0.
- Il terzo bit di controllo è formato da: 0101. Il valore 1 appare due volte, quindi b4 sarà 0.
- il quarto bit di controllo è formato da: 1101. Il valore 1 appare tre volte, quindi b8 sarà 1.

bit 1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12
1	0	0	0	0	1	0	1	1	1	0	1
pos 1	2	3	4	5	6	7	8	9	10	11	12

Memorie cache

(Fast)
CPU > Memorie

Progettisti di
memorie
(Più circuiti su un chip)

Architetture Pipeline e
superscalari

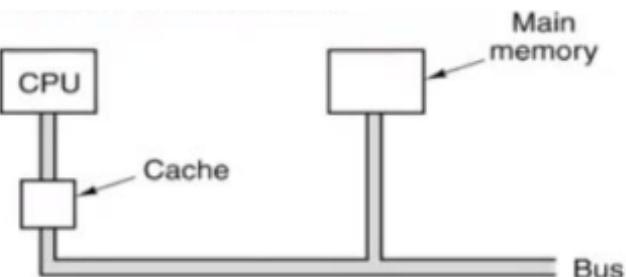
→ più' velocità

Squilibrio tra memoria e CPU a livello di performance

La ricezione della CPU
richiede molti cicli macchina

L'inserimento di memorie veloci nelle CPU
comporterebbe un aumento di dimensioni
e di prezzo

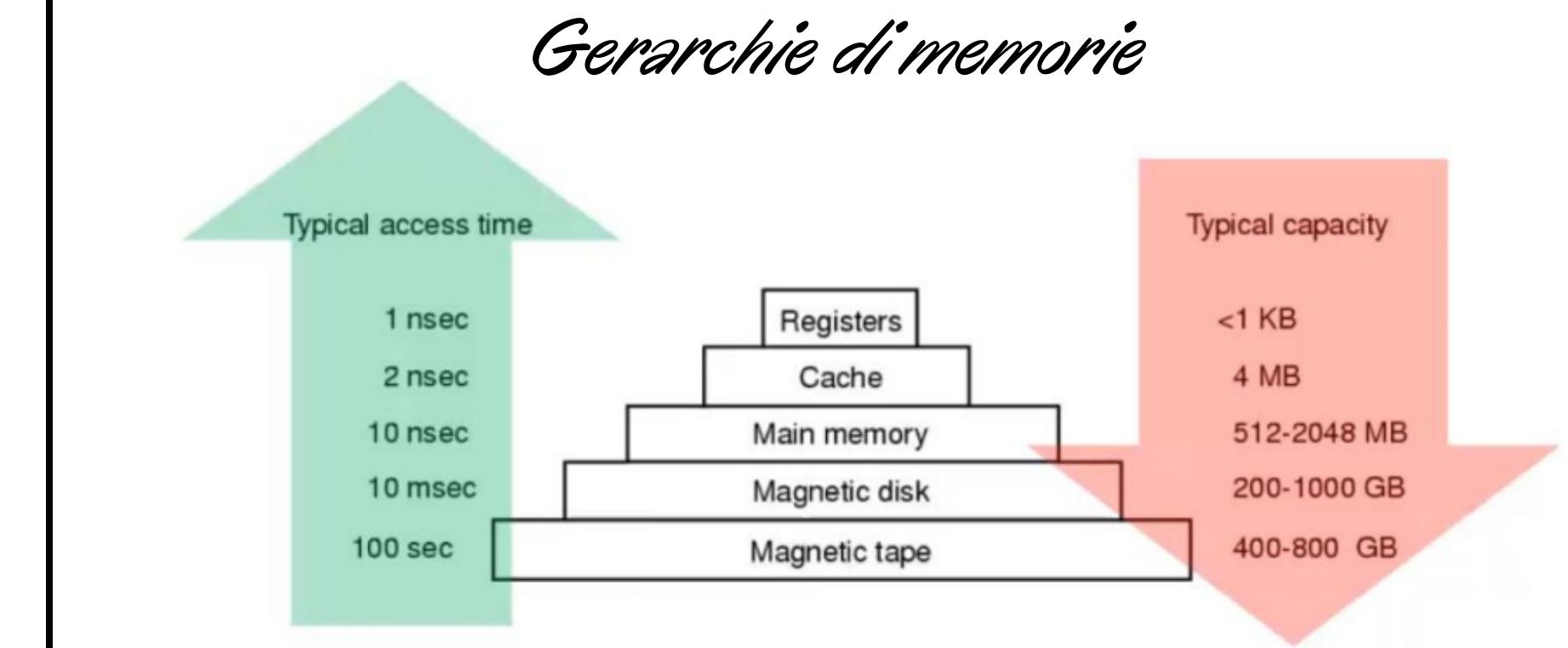
Soluzione



Memorie cache

Una memoria piccola e veloce posizionata tra CPU e Memoria
Memorizza piccole porzioni di dati
Registra e conserva le parole maggiormente utilizzate

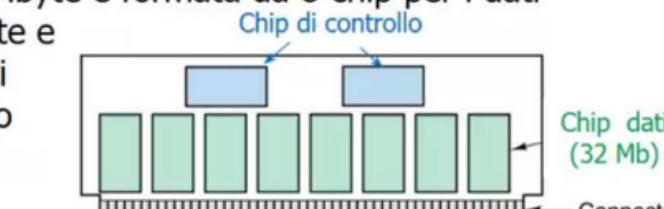
Gerarchie di memorie



Assemblaggio | tipologie

Gruppo di integrati
da 8 o 16

- SIMM (Single Inline Memory Module) : UNA riga di connettori su un solo lato della scheda
32 bits x ciclo clock



Una SIMM da 256 Mbyte è formata da 8 chip per i dati
ciascuno di 32 Mbyte e
due circuiti integrati
destinati al controllo
della scheda.

- DIMM (Dual Inline Memory Module) DUE righe di connettori su entrambi i lati della scheda.
64 bits x ciclo clock

Dischi magnetici

Composizione:

uno o piú piatti di alluminio rivestiti con una superficie magnetizzabile che ruota rispetto al proprio centro



Testina con
movimento
longitudinale



SOLENOIDE

Oriente (0/1) le
particelle di
materiale ferroso a
seconda della
polaritá (+/-)

N.B.

Nei nuovi dischi:
maggior settori = Maggior zone
esterne

Sequenza
circolare
completa dei bit



TRACCIA

- Ogni traccia é divisa in settori (len 512byte)
- distanza tra settori --> Intersector gap
- Tracce interne = maggiore densitá MA medesimo contenuto



Costituita da:

- Un preambolo --> Sincronizzazione delle testie pre read/write
- ECC , Hamming code o Reed-Solomon

Floppy disk

Primi dischi rimovibili - Quantitá minima di informazioni memorizzabili

Hard disk

Disp. di memorizzazione: collegamento Interno/esterno, esistono quelli magnetici e solid state. Costituito da una pila di dischi, l'insieme di tracce costituisce il cilindro.

Le performance dipendono da: Tempo medio di seek (tra 5 e 10 ms) - latenza rotazionale (5400, 7200, 10k RPM (tra 3 e 6 ms)) - tempo di trasferimento(dipeso dalla densitá lineare e la velocitá di rotazione).

IDE

(Integrated Drive
Electronics)



EIDE

(Extended
IDE Drives)

Gestione disco con capacità fino a 504 MB con una velocitá di 4 MB/s
il SO read&write i dati sul disco lavorando sui registri della CPU
invocando successivamente il BIOS

ATA-3 - ATAPI-7

ATA-3 (Advanced Technology for Attachment-3) analogia PC/ATIBM -
velocitá 33 MB/s.

ATAPI-4 (ATA Packet Interface-4)

ATAPI-5 : 28bit e velocitá di 66 MB/s

ATAPI-6 : 48bit e velocitá di 100 MB/s

ATAPI-7 : Serial ATA / connettore a 7-pin (NO 80pin paralleli) velocitá 150 MB/s

RAID

Livello 0

Non é un vero raid --> No ridondanza

n dischi = n backup.

Duplica i dischi - strisce scritte 2 volte - lettura anche delle copie = carico su piú dischi. - scrittura SLED lettura migliorata - replica del disco in caso di guasto
-Ripristino del disco rotto.

Livello 1

Utilizzo di Parole binarie o byte.

7 dischi 1 bit ciascuno = 4 bit + 3 bit (hamming)
Ripristino disco rotto

Livello 2

livello 3 = Livello 2 semplificato

1bit di paritá scritto su 1 disco di paritá /parola |
Dischi sync - poca affidabilitá su errori casuali ma
efficiente ripristino del disco. Vel.di trasferimento = SLED

Livello 3

Utilizzo delle strisce

Livello 4

Simile al 0 - EXOR bit-a-bit = striscia
di paritá

Livello 5

Strisce di paritá --> round-robin
mode = no collo di bottiglia DP.

Ripristino complesso

SSD

Memoria flash non volatile

capacitá di memorizzazione < usura dei transitor (nel tempo)

Prestazioni eccellenti x3 HD magnetico - Costoso - No elem.meccanici

CD-ROM



settore



- preambolo (12 byte + 4 byte)
- 2KB modo 1 - 2336 byte modo 2
- ECC 288 byte modo 1

CD-R

Strato di pigmento
(trascrizione pit)

La Scanalatura guida il laser
--> Se colpisce il pigmento =
crazione regione oscura non
ripristinabile.

CR-RW

No pigmento ma Lega
Laser 3 diverse intensitá:

- 1) Alta potenza (pit)
- 2) Media Potenza (land)
- 3) Bassa potenza (pit/land)

DVD

(Digital video Disk)

Simil CD / Composto di policarbonato / Lettura Pit&Land.
Illuminati da un diodo laser e letti da un fotorilevatore.
Caratteristiche: 1* Pit più piccoli 2*Spirale stretta 3*Laser color rosso 4* 4.7 GB e vel. 1.4 MB/s

Pan&Scan (3:2) --> (4:3)

Lettura tramite Laser BLU --> lunghezza d'onda più piccola
--> Messa a fuoco più accurata --> Pit&land

singolo lato - 25GB
doppio lato - 50 GB

I/O Scheda MADRE →

Scatola metallica contenente una grande scheda e circuiti,
Contiene: Chip della CPU, slot DIMM, chip di supporto ed il
Bus PCI
(PCI > ISA)

I/O → • Controllore
• Dispositivo stesso } Collegati tra loro

Controllore →

Gestione dell'I/O e accesso al bus
Spedisce istruzioni al lettore
Legge e scrive senza l'intervento della CPU(Direct Memory
ACCESS «DMA»)

Una volta effettuato un trasferimento il controllore produce un "interrupt" che permette alla CPU di sospendere qualsivoglia esecuzione (GESTIRE INTERRUPT).

Post interrupt --> CPU riprende il lavoro.

Arbitro del Bus = Chip che gestisce i turni d'uso del Bus

Furto di cicli = concessione d'uso del bus mentre è utilizzato da altri

ISA

→
Extended ISA
(Successore e retrocompatibile)

EISA

Efficiente rispetto al suo predecessore ma ancora non ottimale per gli standard dei calcolatori

PCI

→
(Peripheral component
interconnect)

Prodotto dalla Intel.
66MHz x 64bit

→ Controllore →
• Memoria
• Bus PCI

↓
Trafico non occupa solo il bus

PCIe

→
Relativamente più veloce di PCI.
1bit d'ampiezza rispetto ai 7bit del PCI

Velocità bassa latenza con un trasferimento di 1GB/s

PCIe 3.0

→
Schede madri con slot a 16 corsie per la Scheda grafica usano la tecnologia 3.0 che permette loro di raggiungere una larghezza di banda pari a 16 GB/sec.

Tecnologia comunemente usata per il 3D

Terminali

Tastiera, Touch screen, Schermi piatti, RAM, Mouse, Controller videogame, kinect, stampanti, modem, LAN, macchina fotografica.

Codifica caratteri: ASCII

(American Standard Code For Information Interchange)



- 7 bit
 - 128 caratteri distinti
- 1 byte = 1 carattere
I codici da 128 a 255 non rientrano nell'ASCII

Esa	Nome	Significato	Esa	Nome	Significato
0	NUL	Nullo	10	DLE	Uscita trasmissione (Data Link Escape)
1	SOH	Inizio intestazione (Start Of Heading)	11	DC1	Controllo periferica 1
2	STX	Inizio testo (Start Of Text)	12	DC2	Controllo periferica 2
3	ETX	Fine testo (End Of Text)	13	DC3	Controllo periferica 3
4	EOT	Fine trasmissione (End Of Transmission)	14	DC4	Controllo periferica 4
5	ENQ	Interrogazione (Enquiry)	15	NAK	Riconoscimento negativo (Negative Acknowledgement)
6	ACK	Riconoscimento (ACKnowledgement)	16	SYN	Annula (SYNchronous Idle)
7	BEL	Campanello (BELL)	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANCEL
9	HT	Tabulazione orizzontale (Horizontal Tab)	19	EM	Fine supporto (End of Medium)
A	LF	Riga nuova (Line Feed)	1A	SUB	Sostituisci (SUBstitute)
B	VT	Tabulazione verticale (Vertical Tab)	1B	ESC	Esc (ESCAPE)
C	FF	Avanzamento carta/nuova pagina (Form Feed)	1C	FS	Separatore di file (File Separator)
D	CR	Ritorno a capo (Carriage Return)	1D	GS	Separatore di gruppi (Group Separator)
E	SO	Disinserzione (Shift Out)	1E	RS	Separatore di record (Record Separator)
F	SI	Inserzione (Shift In)	1F	US	Separatore di unità (Unit Separator)

Esa	Car	Esa	Car	Esa	Car	Esa	Car	Esa	Car	Esa	Car
20	Spazio	30	0	40	@	50	P	60	'	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	\$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(38	8	48	H	58	X	68	h	78	x
29)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	:	4B	K	5B	[6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	
2D	-	3D	=	4D	M	5D]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	-	6E	n	7E	-
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Figura 2.44. Caratteri ASCII.

UNICODE



Risoluzione dei problemi con UNICODE



-
-
-

Ordine degli ideogrammi
Nuovi ideogrammi
Stessa codifica ASCII (code point limitati)

UTF-8



Standard WWW

Universal Character Set Transformation Format
Lunghezza variabile 1 a 4 byte --> 2 miliardi caratteri
I caratteri da 0 a 128 riprendono la codifica ASCII

Se sono presenti piú di un byte, i successivi al primo inizieranno sempre per 10xxxxxx

Se il primo byte inizia per:

0xxxxxxx --> 1 byte

110xxxxx --> 2 byte

1110xxxx --> 3 byte

11110xxx --> 4 byte

Nuovo sistema di codifica
dallo 0 al 255 sfrutta gli stessi caratteri di ASCII
Code point --> 16 bit unico x carattere

Livello logico digitale → Hardware del calcolatore
 Composto da porte logiche che lavorano con Boole.
 Compone Funzioni complesse.

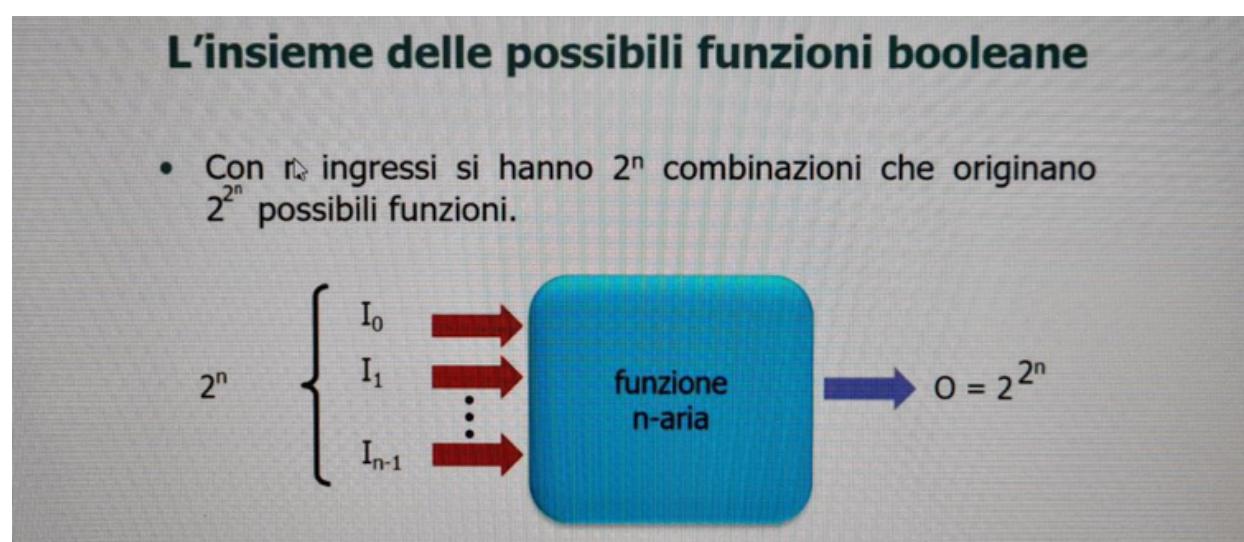
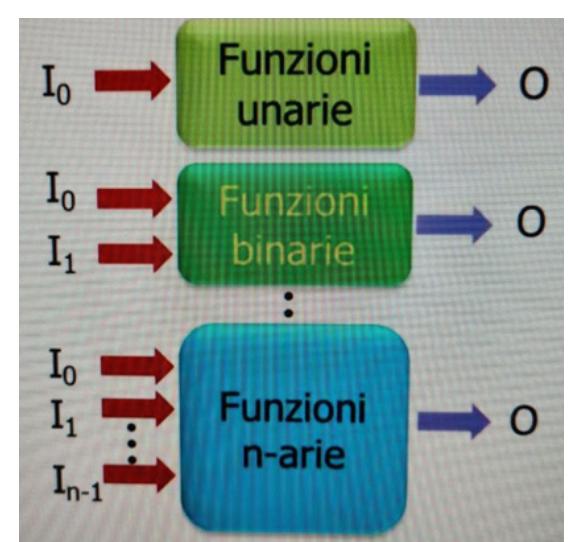
Algebra di BOOLE → Algebra basta su un insieme {Dominio} di due valori:
 True & False



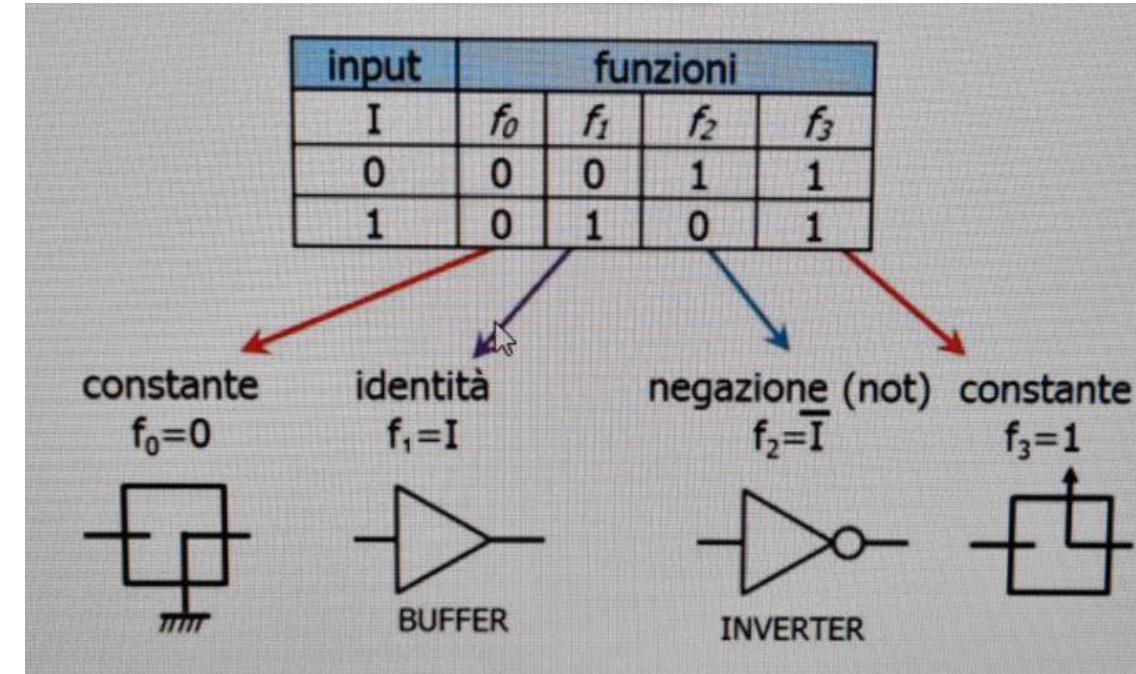
True=1

False=0

Funzioni Booleane:
 Unarie: $B \rightarrow B$
 Binarie: $B \times B \rightarrow B$ → Ingressi/uscite variano in un insieme finito di valori.
 N-arie: $B \times \dots \times B \rightarrow B$



Le funzioni ad un solo operando sono $2^2=4$



NOT →

Tavola di verità

A	X
0	1
1	0

$X = \bar{A}$

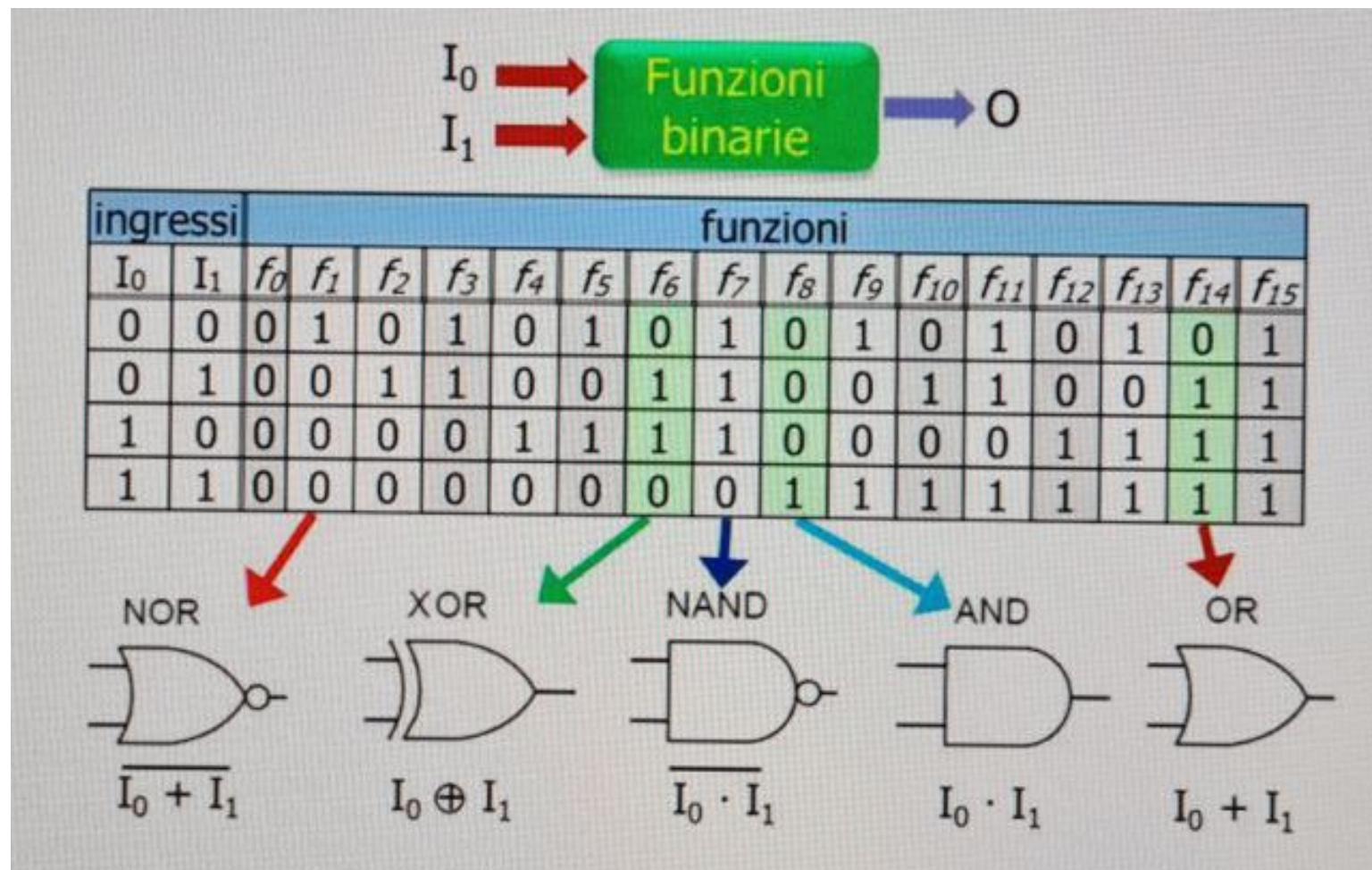
La negazione è una funzione che inverte il valore della variabile in ingresso

Proprietá dell'algebra Booleana

→

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\bar{AB} = \bar{A} + \bar{B}$	$\bar{A + B} = \bar{A}\bar{B}$

Possibili funzioni binarie

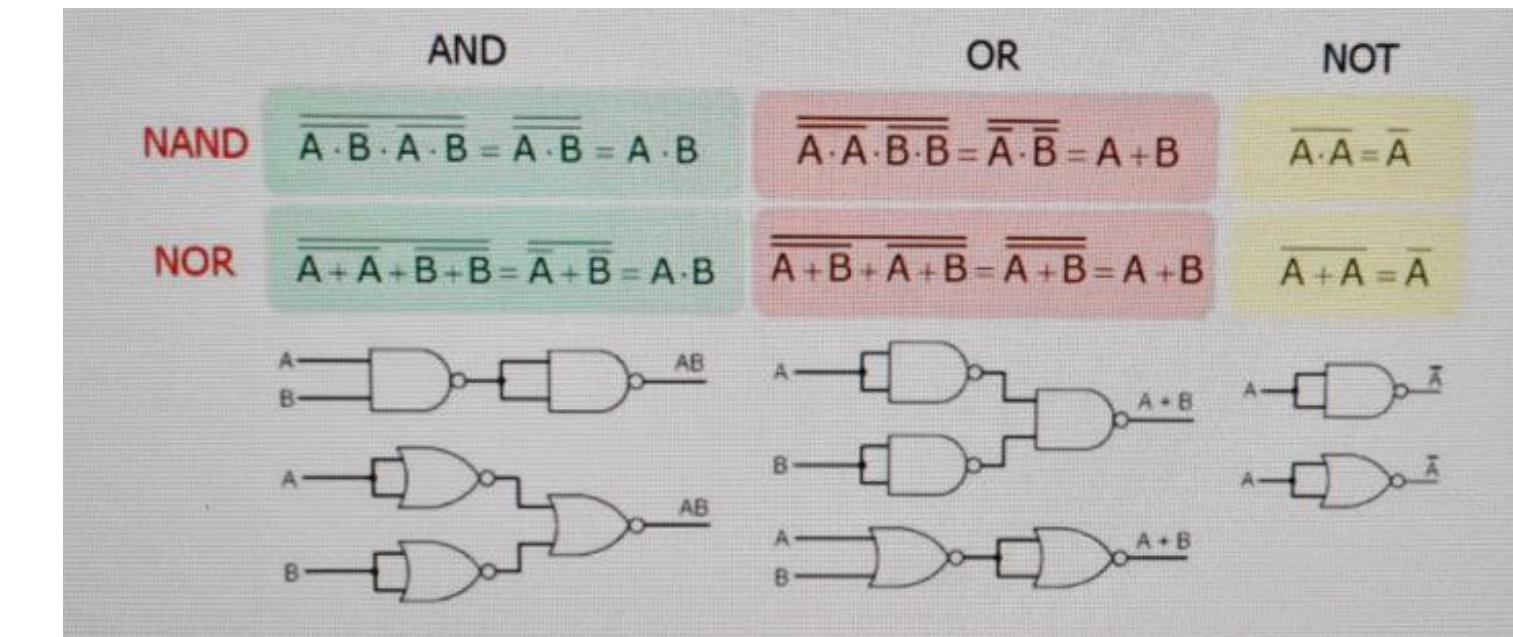


- Per ogni espressione logica
 - Circuito digitale
 - Colonna della tavola di verità
- Per ogni colonna della tavola di verità
 - Espressione rappresentante
 - Circuito digitale
- Per ogni circuito digitale
 - Espressione rappresentante
 - Colonna della tavola di verità

Funzione logica booleana → Realizzabile con AND, OR e NOT

Realizzabile con un solo operatore detto UNIVERSALE → Realizzabili con una sola porta NAND o NOR

Porte logiche → Realizzate utilizzando solo porte NAND o NOR



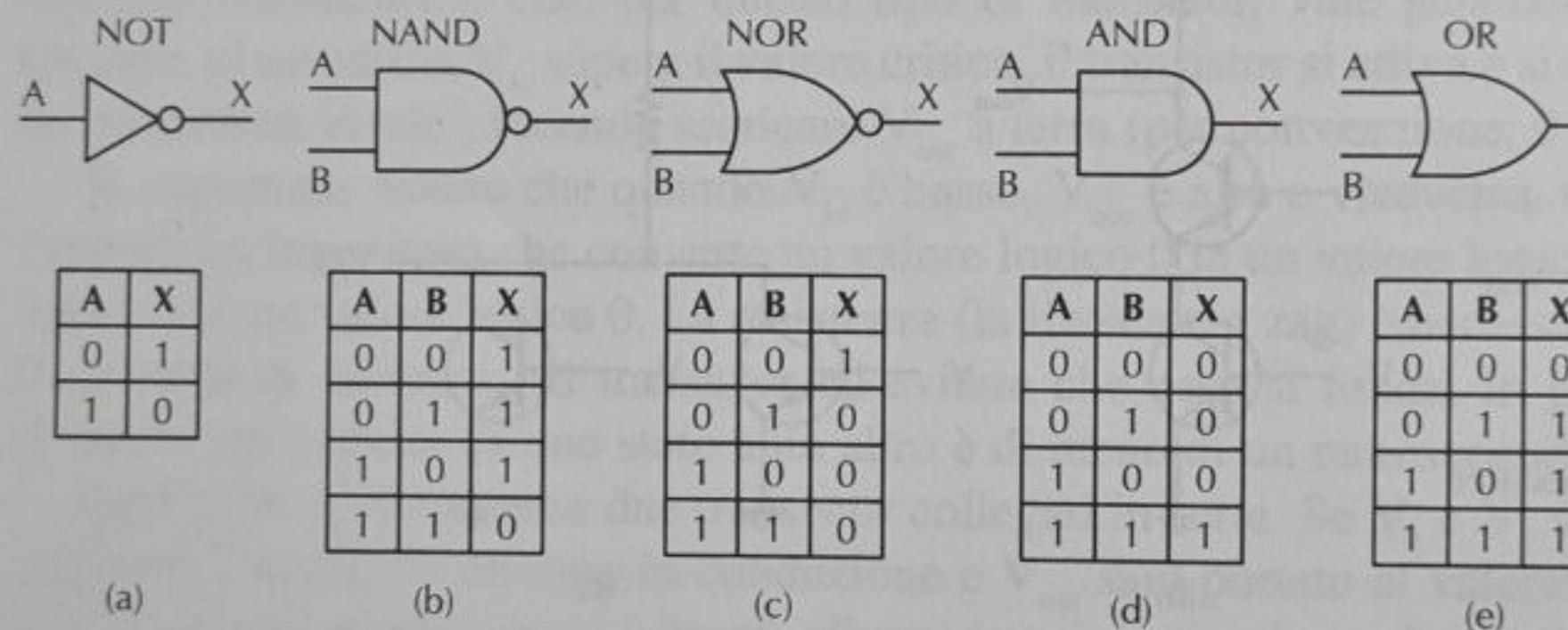
Vendute individualmente dentro circuiti integrati (chip)

SSI (small scale integrated) : Meno di 10 porte.
 MSI (medium scale integrated): Tra le 10 e le 100 porte
 LSI (large scale integrated): Tra le 100 e le 100.000 porte
 VLSI (Very large scale integrated): Over 100.000 porte.

Circuiti combinatori → Circuiti digitali nei quali l'uscita dipende esclusivamente dagli ingressi

Circuiti sequenziali → Circuito digitale dove l'uscita dipende dagli ingressi e dallo stato del circuito

COMBINAZIONI NOT-NAND-NOR-AND-OR



Decoder →

Utile per selezionare il chip corrispondente ad un indirizzo.

Circuito che riceve un valore n-bit come ingresso e come uscita l'unica linea corrispondente al suo valore numerico

Tavola di verità								
A	B	C	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅
0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0
0	1	1	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	1	0
1	1	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	1

MULTIPLEXER →

Circuito con 2^n ingressi (un'uscita e n ingressi di controllo «selettorie»)

Utilizzato per realizzare qualsiasi funzione logica

Funzione F con 3 ingressi

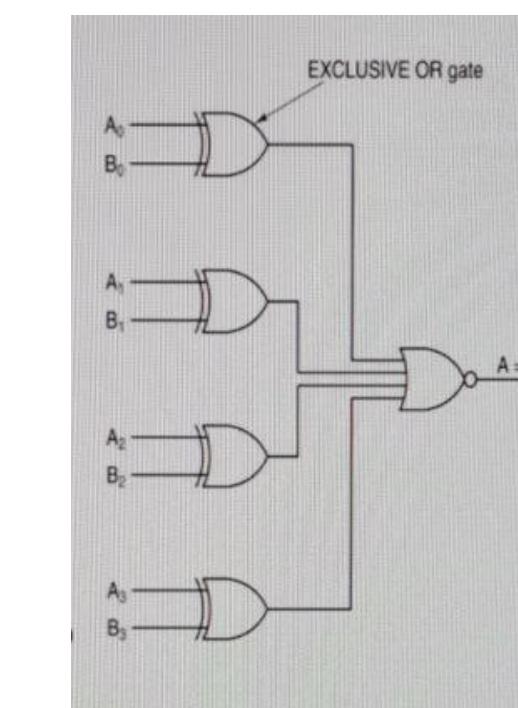
{A, B, C}

Tavola di verità			
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Comparatore →

Circuito che compara due parole A e B di 4bit
=1 se i bit della prima sono uguali alla seconda

EXCLUSIVE OR gate (XOR) per bit-a-bit.
risultati parziali inviati ad un NOR



PLA [Programmable Logic Arrays] →

Chip generico per la costruzione di una funzione arbitraria



Utilizza gruppi di AND e OR

HALF Adder →

Circuito in grado di eseguire la somma tra bit



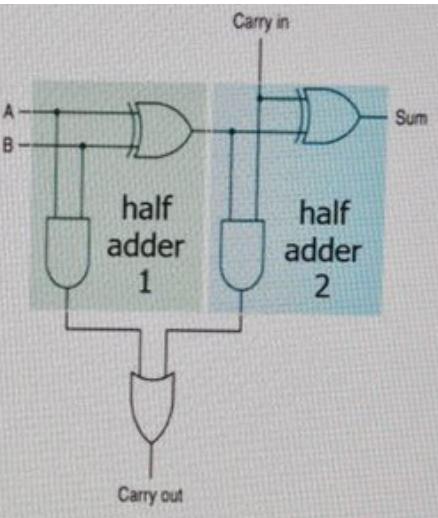
Due variabili: A e B
il circuito genera somma e riporto

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

FULL Adder →

In Ingresso 3 bit [A, B e il riporto in ingresso]
In uscita somma e riporto (carryOut)

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Unitá aritmetica logica (ALU) →

Circuito in grado di calcolare tutte e 4 le funzioni [A and B - A Or B - B/ - A + B]



A+B --> somma aritmetica (no Booleana)

Valore binario : 00 - 10 - 01 - 11

Bit Slices → (Suddivisione di un bit) Costruisce ALU di larghezza arbitraria



comando INC --> Incrementa +1 la somma o il valore (A+1) (A+B+1)

CLOCK → Circuito che emette impulsi standard a intervalli costanti



Memoria → Componente utilizzato per la memorizzazione di dati e istruzioni
Composto da: circuiti sequenziali in cui l'uscita (O) dipende dagli ingressi (I) e dallo stato (S)

LATCH →

Realizzato con due NOR (o NAND) con uscite retroazionate

LATCH SR → 2 Ingressi : S (Set) imposta il valore ad 1, R (reset) imposta il valore a 0



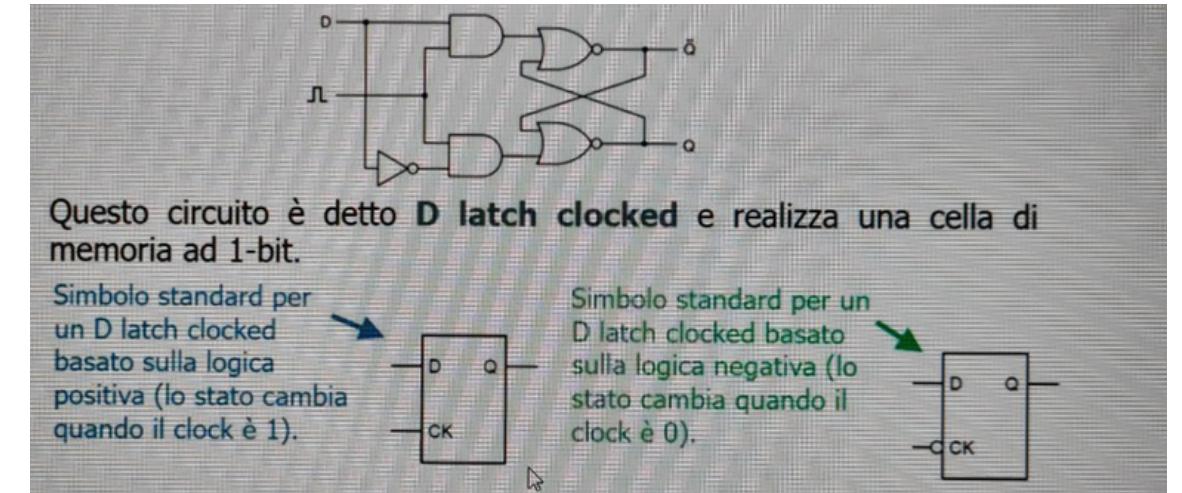
S ed R agiscono solo quando il clock è attivo.

LATCH SR CLOCKED → Realizzato aggiungendo 2 AND che mascherano S e R originali.
Stato instabile = S=R=1 --> Stato stabile = S=R=0

D-Latch clocked → Per risolvere l'instabilità di SR ($S=R=1$) utilizziamo un solo Ingresso D

↓ ↓

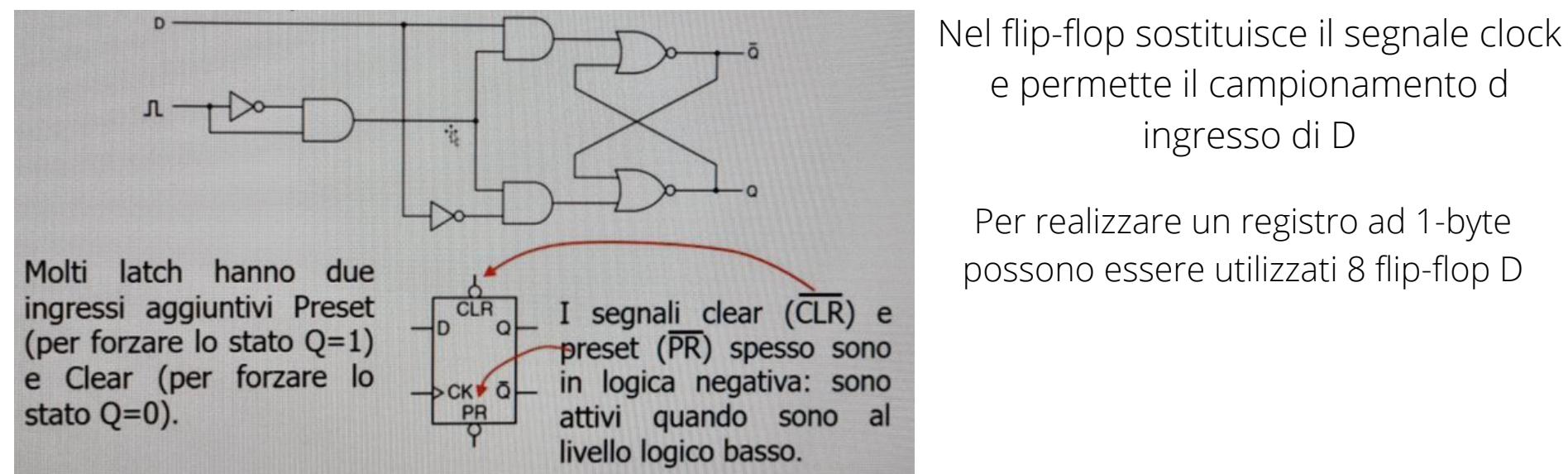
D=1 + clock alto = Q=1 D=0 + clock alto = Q=0



Flip-Flop → Segnale clock alto = level triggered.
Nel flip-flop lo stato (S) cambia durante la transizione del clock da 0 a 1 (fronte in salita) oppure da 1 a 0 (fronte in discesa)

Edge triggered → Scatto sul fronte (in salita/in discesa)

Generatore di impulsi → Realizzabile con una porta AND e un inverter (ritarda il segnale)



Organizz. Memoria → Grandi memorie richiedono un indirizzo x localizzare la cella sul quale eseguire il read/write
Ogni cella = n' bit

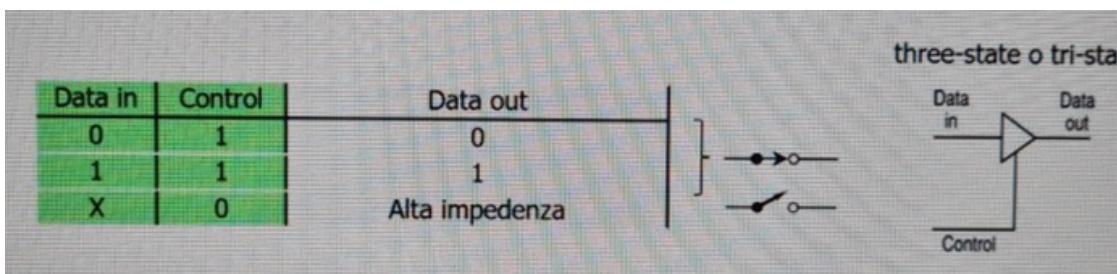
Segnali di controllo

- CS (Chip Select) Usato per selezionare il circuito.
- RD (ReaD) : Specifica l'operazione: RD=1 lettura, RD=0 scrittura
- OE (Output Enable) Abilita l'uscita.

Connessione tra uscite →

- Open collector
- Buffer Tri-State

Nel Buffer Tri-state l'uscita assume un terzo stato di alta impedenza (circuito aperto) e la porta logica fa da interruttore che abilita o meno il segnale d'ingresso all'uscita



Scollegare le uscite nell'operazione di scrittura

Chip di memoria →

Schemi riusabili espandibili.
Segnali divisi in aree logiche positive o logiche negative



CS (Chip Select) : Seleziona il Chip da attivare

WE (Write Enable) Seleziona l'operazione

OE (Output Enable): Fonde insieme i segnali d'uscita.

- A è asserito quando A=1
- \bar{B} è asserito quando B=0

- A è negato se A=0
- \bar{B} è negato se B=1

RAM →

[Random Access Memories] Memorie con lo stesso tempo di accesso indipendentemente dalla posizione della cella

- RAM statica (SRAM) : Circuito simile Flip-flop D attive finché vi è alimentazione. Molto veloci , utili per cache 2'Livello.
- RAM dinamica (DRAM) : No Flip-Flop ma Array con transistor ed un condensatore (Carico=1 Scarico=0). DRAM > SRAM ma più complessi

Chip di Memoria [Non volatile]



- ROM [Read-Only Memories] : Sola lettura - Dati inseriti in fase di costruzione.
- PROM [Programmable ROM] : ROM programmata una sola volta bruciando i fusibili.
- EPROM [Erasable PROM] : Una PROM riprogrammabile attraverso raggi ultravioletti.
- EEPROM [Electrically Erasable PROM] No Luce ultravioletta ma impulsi elettrici per la riprogrammazione - Byte-a-Byte.
- Memoria flash : Speciale EEPROM cancellata a blocchi e non a Byte.

Chip della CPU →

Pin/segnali



- Indirizzo
- Dati
- Controllo }

BUS

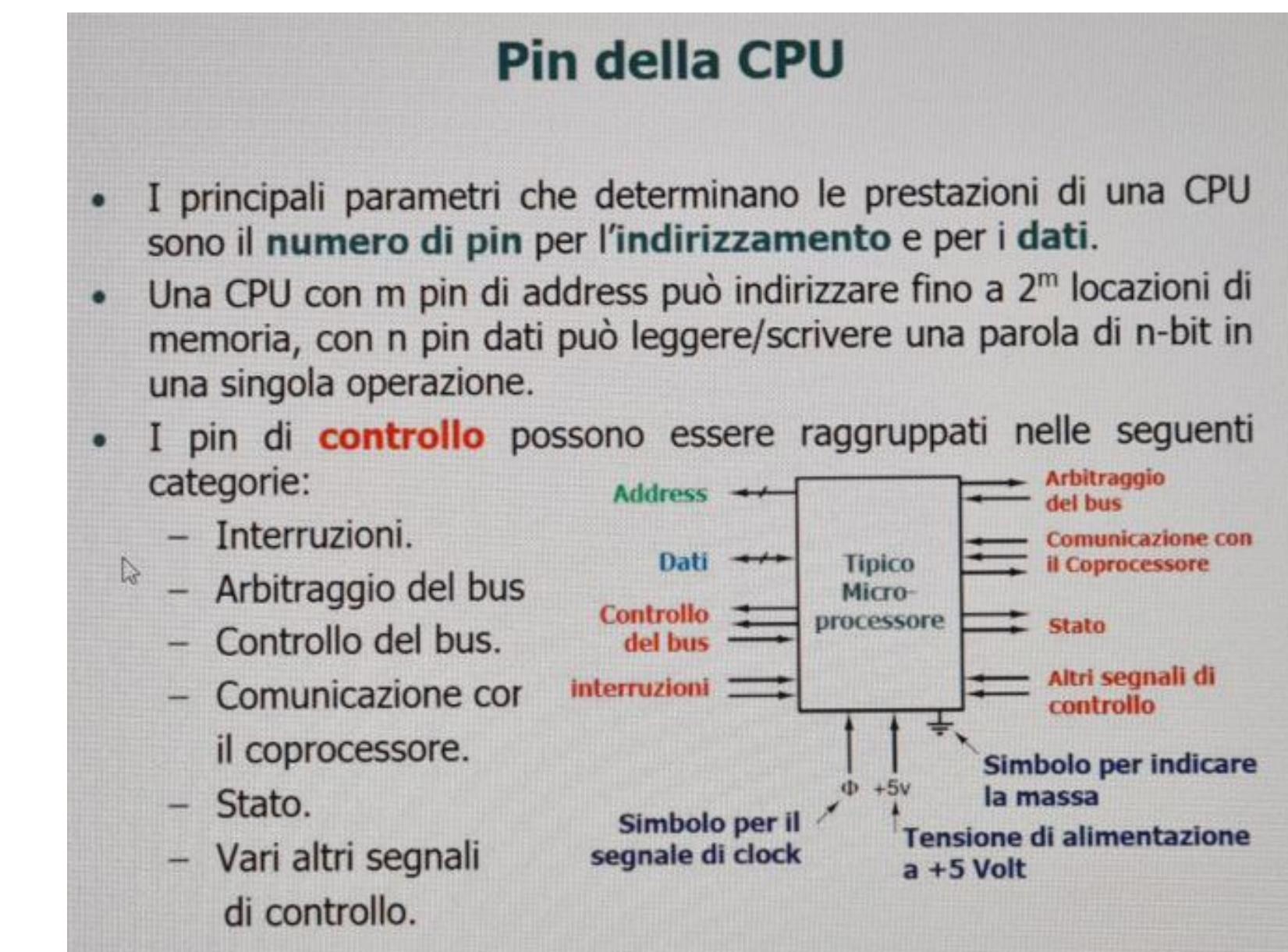
→ Connessione agli altri componenti

Pone l'indirizzo di memoria dove sta l'istruzione

Informa la memoria della prossima lettura

risposta della memoria ponendo la parola sul bus

CPU riceve il segnale di controllo, legge e predisponde l'operazione,



Bus del Calcolatore →

- Interni alla CPU : Connessione registri-ALU
- Esterne alla CPU : Connessione alla memoria o periferiche I/O



{ Esterne --> Regole imposte dal Protocollo del Bus }

Collegamento elettrico comune che unisce vari dispositivi

Vecchi PC un SOLO bus di sistema - Nuovi PC diversi bus per il collegamento CPU-Memoria e CPU-device I/O

- Dispositivi ATTIVI (o master): Possono iniziare un trasferimento
- Dispositivi PASSIVI (o slave): Restano in attesa di una richiesta di un master.
- Bus Driver: Chip che amplifica il segnale master
- Bus Receiver: Chip che permette il collegamento degli slave al Bus

AMPIEZZA DEL BUS →

Se un Bus ha n linee di address, allora la CPU ha 2^n locazione di memoria

Temporizzazione del bus →

- Bus Sincrono: Utilizzano un clock --> Ciclo del bus (Operazione = tot periodi di clock)
- Bus Asincrono: Qualsiasi durata per il ciclo di bus = Maggior efficienza --> Difficoltà nella costruzione

Arbitraggio del bus →

Previene le situazioni di conflitto in cui più dispositivi tentino di diventare master.

CENTRALIZZATO: Riceve una richiesta, concede la richiesta asserendo una linea di concessione del bus. Vince il più vicino all'arbitro. [Priorità cablaggio della connessione]

DECENTRALIZZATO: Ogni dispositivo ha una propria linea di richiesta. Una richiesta per volta - Al termine dell'utilizzo la linea viene disattivata.



Schema:

- La linea di richiesta del bus (Wired-OR)
 - Linea THE BUSY asserita dal disp Master corrente
 - Linea di arbitraggio propagata rta i dispositivi in cascata
- Il bus deve:
- Busy negata l'Ingresso IN asserito
 - Nega OUT, asserisce Busy e diventa il master
 - Sblocca OUT asserendola e libera Busy

Comando INT [INTerrupt] → Comando di interruzione alla CPU

Comando INTA [Acknowledge signal] → Comando per la presa gestione della richiesta di interruzione da parte della CPU

Comando ISR [Interrupt Service Routine]



Numero utilizzato per accedere al vettore di interruzione e trovare l'indirizzo

Esempi di CPU

PENTIUM 4 →

Microarchitettura
interna (NetBurst)

- Ha una pipeline più profonda dei suoi predecessori.
- 2 Unità ALU che operano al doppia della frequenza di clock.
- Supporta l' Hyperthreading.
- Due registri che simulano due CPU fisiche.
- 2 o 3 livelli di cache.

8 KB di SRAM sulla cache di primo livello L1
1 MB per il secondo livello di cache L2
Extreme Edition 2 MB per il terzo livello di cache L3
Consistenza tra cache attraverso lo snooping

- Possiede due Bus sincroni primari
- Il memory bus è utilizzato per accedere alla (S)DRAM.
- Il bus PCI utilizzato per il colloquio tra disp. I/O
- Il Pentium4 consuma 63/82 W.

Diretto discendente della CPU 8088.

Contiene 55 milioni di transistor ed opera ad una freq. di clock fino a 3.2GHz.

Scambia dati con memorie a 64-bit ma con macchine a 32-bit.

Funzionamento del Pentium 4

→ Spegnimento calcolatore = CPU stato inattivo / sonno profondo.

PIN e Segnali del Pentium4

Possiede 478 Pin:

- 198 x i segnali
- 85 alimentazione
- 180 massa
- 15 free

Segnali di ARBITRAGGIO DEL BUS:

- BR0# richiesta bus
- BPRI# richiesta bus alta priorità
- LOCK# Bus occupato

Segnali di risposta al bus:

- RS# contenente il codice di stato.
- TRDY# lo slave è pronto per accettare dati.
- BNR# Lo slave può inserire stati di attesa (Wait state)

↓

- i valori delle cache e dei registri sono preservati nonostante il clock e le unità interne sono spente.
- Solo un segnale hardware può risvegliarlo

Segnali dell'Address bus e di controllo:

- A# 36 bit di indirizzamento
- ADS# indica che il bus indirizzi contiene un rif.valido
- REQ# Specifica l'operazione

Segnali del bus dati:

- D# 64 bit del bus dati
- BNR# Bus occupato
- DRDY# dati pronti sul bus

Intel core i7 →

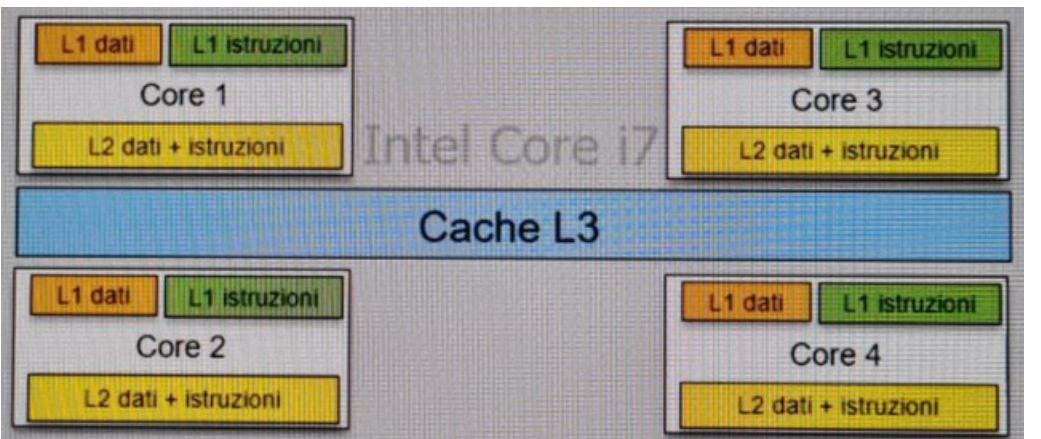
Macchina 64bit diretto discendente della CPU 8088

Ogni core è hyperthreaded

Livelli di cache:



Dispone di 3 livelli di cache - ogni core effettua uno snooping sul bus



- Due distinte cache L1 da 32KB per dati e istruzioni, per core.
- Una cache L2 da 256KB integrata di dati e istruzioni, per core
- Una cache L3 da 4 a 15MB, condivisa tra i core.

UltraSPARC III →

CPU RISC a 64bit prodotta dalla Sun Microsystem.

Compatibile con architettura 32bit (SPARC v8).

Chip prodotto dalla Texas Instruments.

Larghezza di linea 130nm e clock 1,3 GHz.

50W di potenza con probl. di dissipazione termica.

• 2 cache L1: 32KB istruzioni e 64KB dati

• 2 cache L2: 2 KB prefetch e 2 KB colleziona scritture

• controller della cache e la logica di ricerca dentro il chip anziché la memoria.

• Utilizza un address bus di 43bit = gestione 8 TB di memoria princ.

• bus dati era di 128bits = 16 bytes per volta

• velocità del bus 150 MHz = banda di 2.4 GB/s.

• Architettura UPA (Ultra Port Architecture).

• Memoria principale divisa in line di cache da 64 byte.

• L1 contiene 256 line di istruzioni e 25 line di dati

• Ciò che non entra in L1 passa a L2

Microcontrollore 8051 →

Chip Intel 8051 microcontrollori più diffusi per il basso costo.

Circuito integrato da 40pin con 16-bit di address e 8-bit bus dati.

Ha 32 line di I/O, organizz. in 4 gruppi da 8 bit ciascuno.

- PSEN (Program Store Enable): Indica che la CPU vuole leggere il program. di memoria.
- EA (External Access) : Alto--> Usa sia la M.Interna che la M.Esterna; Basso--> Usa soltanto la memoria esterna.
- Possiede due clock esterni, due contatori a 16-bit, due livelli di priorità di interruzione, asseriti al livello basso.

Le linee di I/O sono:

- TXD : per l'uscita seriale.
- RXD: Per l'ingresso seriale.
- 4 porte bi-direzionali ciascuna con 8-bit paralleli
- RST: Reset del chip

Esempi di BUS → PCI , PCI Express (PCIe) e USB

I primi BUS →

Il bus del primo PC IBM era basato su architettura 8088 con 62 segnali:

Segnali di controllo

20 bit address bus

8 bit bus dati

segnali come: INT, DMA ...

Bus PCI lavora con

freq.clock fino a 66

MHz, gestiva

trasferimenti a

64bits con banda di

528 MBs

Il Bus AGP introdotto anni 90 dedicato alle schede grafiche eseguibile a 2,1 GBs

Il bus PCIe esegue fino a 16GBs. basato su corei7 le interface sono integrate nel chip:

- 2 canali di memoria a 1,333 GHz, banda di 10GBs
- Un canale PCIe a 16 corsie con banda di 16GBs verso I/O.

Arbitraggio del bus PCI → Arbitraggio centralizzato inserito nel chip del bridge

Connettori dell'arbitro:

- REQ# Richiesta del bus
- GNT# Conferma della richiesta

BUS PCIe → Architettura basata su connessioni seriali punto-punto.
CPU, memoria e cache sono connesse al chip di bridge.
Presenza di una Switch con connessione dedicata punto-punto.

Connessione composta da coppia di canali unidirezionali, ciascuno composto da 2 canali

Connessione Master-slave

Ogni pack contiene informazioni di controllo (intestazione), ed elimina i segnali di controllo (payload). Ogni pack contiene un codice di correzione.

Architettura del bus PCIe → Mini rete a commutazione di pacchetto;
connessione tra switch e dispositivi non può eccedere i 50cm. System espandibile con doppio switch.

Connettori seriali

USB → (Universal Serial Bus) Standardizzato nel 1998 x il collegamento con dispositivi lenti.

Versione 1.0 banda 1,5 Mbps
Versione 2.0 banda 480 Mbps
Versione 3.1 banda 10 Gbps

- Composto da un Hub Principale (o Root Hub).
- Il cavo di collegamento si compone di 4 fili:
 - 1) Due per i dati (D+ e D-)
 - 2) Uno di alimentazione
 - 3) Uno per la massa
- USB 3.0 possiede invece 10 fili

Hub root rileva il nuovo dispositivo collegato e genera un'interruzione del sys operativo che:

- Interroga il dispositivo
- se la largh banda è sufficiente gli assegna un numero unico
- Confermato l'utilizzo del disp.

Soddisfatte le richieste del Hub, viene effettuato un collegamento punto-punto.

Per il mantenimento del sincronismo spediscono dei Frame:

- Frame controllo
- Frame isocroni
- Frame bulk
- Frame interrupt

Ogni frame contiene uno o più pack:\

- Token Pack
- Dati Pack
- Handshake Pack
- Special Pack

Interfacce I/O

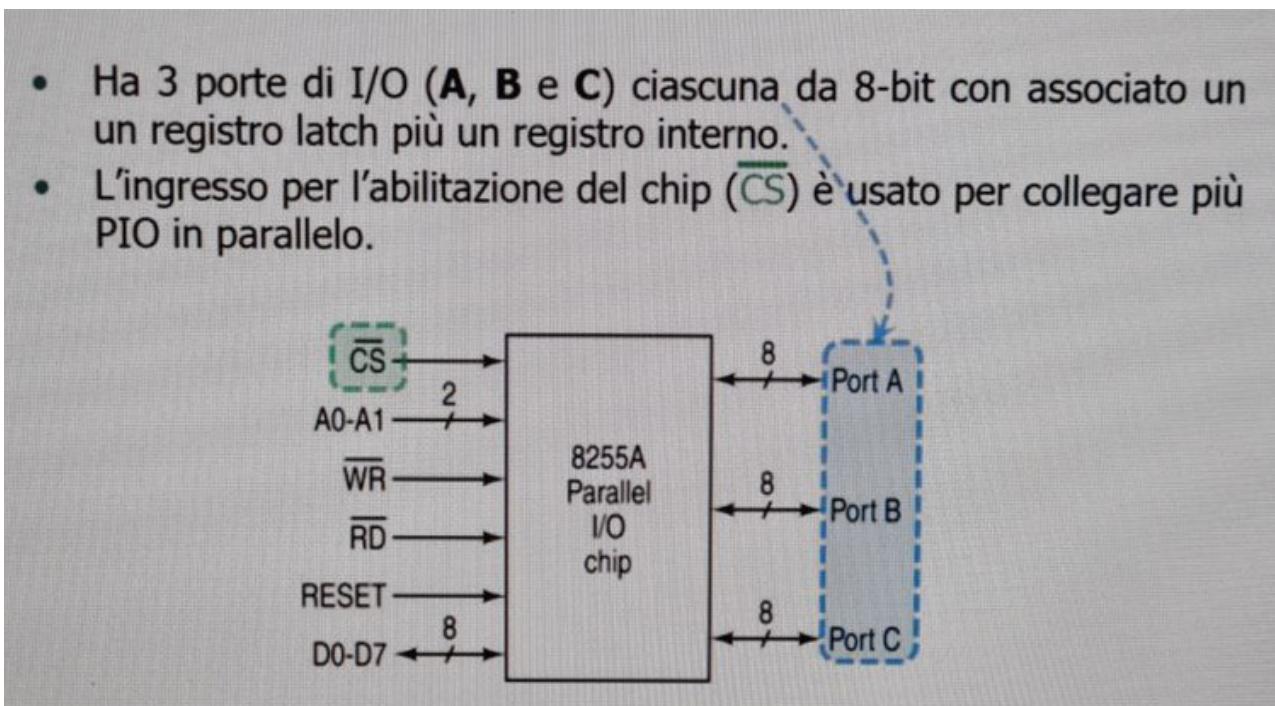
Schede che permettono ai dispositivi di collegarsi sul bus e scambiare dati nel calcolatore

Indirizzamento dell' I/O

- Port-mapped I/O o I/O isolato
Necessaria una linea del control bus che distingua dove effettuare l'operazione istruzioni specifiche

Chip dei controllori:

- UART (Universal Asynchronous Receiver Trasmitter): Legge un byte e trasmette un bit alla volta (o viceversa).
- USART (Universal Synchronous Asynchronous Receiver Trasmitter): UART + trasmissione sincrone.
- PIO (Parallel I/O): Chip utile per la comunicazione parallela.



Intel 8255A (Es. PIO)

