# Riassunto Architettura dei calcolatori

Valerio Tolli

April 2025

# Indice

# 1 Introduzione

## 1.1 Approccio Strutturale

Un programma è una sequenza di istruzioni che descrive come portare a termine un dato compito.
Per poter eseguire un programma, bisogna poter comunicare ai componenti elettronici del computer delle istruzioni. Il linguaggio macchina è l'insieme delle operazioni dette primitive, che solitamente sono:

- sommare due numeri;
- controllare se un numero vale zero;
- copiare una porzione di dati da una parte all'altra della memoria.

Progettare un calcolatore implica dover scegliere le operazioni elementari, ma nel corso del tempo si è passati a un approccio più strutturale nel concepire l'architettura dei computer. Per far ciò bisogna riuscir a tradurre gli input voluti dagli utenti in istruzioni comprensibili dal computer (obiettivo del corso).

### 1.1.1 Linguaggi, livelli e macchine virtuali

Una tecnica per risolvere il problema posto è la tecnica della **traduzione**: consideriamo un programma scritto in un linguaggio L1, e un linguaggio L0 comprensibile dal computer; per eseguire il programma scritto in L1 possiamo sostituire ogni istruzione di L1 con una equivalente in L0. In questo modo il programma sarà composto solo da istruzioni L0 e potrà essere eseguito sul computer.
Un'altra tecnica consiste nello scrivere un programma (detto interprete) in L0 che accetta come dati d'ingresso programmi in L1; il programma esaminerà tutti i dati in input traducendoli in linguaggio L0. Questa tecnica non prevede un nuovo programma L0, ed è detta **interpretazione**.
La traduzione converte in una fase iniziale il programma scritto in L1 in L0 e lo carica in memoria, così facendo il programma scritto in L1 non è più necessario. L'interpretazione esamina e decodifica ogni riga del programma scritto in L1 eseguendola direttamente. Entrambi i metodi sono utilizzati.
Un altro modo di vederla è immaginare l'esistenza di una macchina virtuale, diciamo M1, il cui linguaggio macchina sia L1. In questo modo l'utente può eseguire programmi in L1 direttamente sul computer. L'idea è man mano definire linguaggi più pratici rispetto al precedente fino a che non se ne ottenga uno adeguato. Ciascun linguaggio usa quello precedente come base, in questo modo il computer può essere visto come una serie di livelli. **METTERE IMMAGINE**

### 1.1.2 Attuali macchine multilivello

La maggior parte dei computer moderni consiste in due o più livelli. **METTERE IMMAGINE**

- **Livello 0, livello dei dispositivi:** i progettisti hanno a che fare con i singoli transistor; i componenti elettronici che possono essere modellati come dispositivi digitali sono i **gate**. Ciascun gate è costituito da una manciata di transistor ed è dotato di uno o più input digitali e calcola in output una semplice funzione dei valori d'ingresso (AND, OR, ...). Combinando più gate è possibile creare delle memorie di 1 bit, 0 o 1; combinando più memorie ad 1 bit è possibile creare dei **registri** ad esempio a 16, 32 o 64 bit. Ciascun registro può contenere un numero il cui valore può variare fino a un certo limite.
- **Livello 1, livello di micro-architettura:** è presente una memoria locale formata da un gruppo di registri (generalmente a 8 o 32 bit) e un circuito chiamato ALU (**Arithmetic Logic Unit**) capace di effettuare semplici operazioni aritmetiche. I registri formano un percorso dati che permettono lo spostamento dei dati in modo tale da avere uno o due registri che vengono usati dalla ALU per effettuare operazioni che vengono infine memorizzate su un altro registro. In alcune macchine le operazioni del percorso dati vengono controllate da un programma, **microprogramma**, mentre su altre viene controllato direttamente dal HW. Il microprogramma è un interprete per le istruzioni del livello 2, che usa il percorso dati per prelevare, esaminare ed eseguire un'istruzione alla volta.
- **Livello 2, livello ISA (Instruction Set Architecture Level):** i produttori quando rilasciano manuali sulle istruzioni della macchina, presentano le istruzioni eseguite in modo interpretato dal microprogramma o dai circuiti.
- **Livello 3, livello macchina del sistema operativo:** presenta oltre alle istruzioni del livello 2, nuove istruzioni, una diversa organizzazione della memoria e la capacità di eseguire programmi in

modo concorrente. I nuovi servizi aggiunti nel livello 3 sono realizzati/eseguiti da un interprete eseguito al livello 2 chiamato **sistema operativo**. Le operazioni in condivisione col livello 2 sono eseguite direttamente dal microprogramma.

- **Livello 4, livello del linguaggio assemblativo:** i primi tre livelli sono progettati per eseguire interpreti e traduttori scritti da dei programmatori di sistema (persone), specializzati nella progettazione e nell'implementazione di nuove macchine virtuali. I livelli 4 o superiori sono pensati per programmatori che devono risolvere problemi applicativi. Mentre i primi 3 livelli sono interpretati, solitamente i livelli 4 e 5 usano traduttori. I programmi in **linguaggio assemblativo** sono inizialmente tradotti nei linguaggi dei livelli 1, 2 e 3 e successivamente interpretati dalla macchina virtuale o reale. Il programma che esegue la traduzione è chiamato **assemblatore**.
- **Livello 5, linguaggi ad alto livello:** sono i linguaggi ad alto livello definiti per essere utilizzati dai programmatori di applicazione. Alcuni linguaggi sono tradotti al livello 3 o 4 da un programma detto **compilatore**; altri sono interpretati.

L'insieme dei tipi di dati, delle operazioni e delle funzionalità di ciascun livello sono chiamate **architettura**.

### 1.1.3 Evoluzione delle macchine multilivello

Hardware(circuiti integrati, cavi, memorie, trasformatori, ...) e software(algoritmi o istruzioni) sono logicamente equivalenti:

- Qualsiasi operazione eseguita dal software può essere svolta direttamente dall'hardware;
- Ogni istruzione eseguita dall'hardware può essere simulata dal software.

### 1.1.4 Invenzione della microprogrammazione

I primi computer digitali, risalendo agli anni 40, avevano solamente due livelli: livello **ISA** in cui erano realizzati tutti i programmi, e il **livello logico digitale**, che li eseguiva.

### 1.1.5 Invenzione del sistema operativo

Nel 1960 fu progettato un computer con un software, sempre attivo, in grado di gestire l'hw, detto sistema operativo.

### 1.1.6 Migrazione delle funzionalità verso il microcodice

A partire dal 1970 i progettisti cominciarono ad arricchire microcodice con nuove istruzioni più efficienti(esempio INC è più veloce di ADD).

### 1.1.7 Eliminazione della microprogrammazione

Verso gli anni '70 i microprogrammi diventarono sempre più grandi e lenti, si pensò allora di eliminare questa tecnica utilizzando solo microcodice eseguito direttamente dall'hw. I moderni processori si affidano alla microprogrammazione.

## 1.2 Pietre miliari nell'architettura dei computer

Un breve riassunto dei vari elaboratori costruiti negli anni.

### 1.2.1 Generazione 0 - Computer meccanici 1942-1945

Le principali macchine sono:

- Macchina di Pascal, in grado di eseguire somme e sottrazioni;
- Macchina di Leibniz in grado di eseguire anche moltiplicazioni e divisioni;
- Macchina differenziale (Different engine) di Babbage in grado di eseguire somme e sottrazioni che incideva i risultati mediante una punta d'acciaio su una lastra di rame.

- Macchina analitica di Babbage composta da quattro componenti: una memoria, un unità computazionale, un dispositivo di input e di output mediante schede perforanti.
- Macchina di Atanasoff basata sull'aritmetica binaria e usava dei condensatori per la memoria che dovevano essere aggiornati periodicamente per non far perdere la carica secondo un processo chiamato jogging the memory (stesso principio delle DRAM).

Con Aiken che realizzà il Mark I che usava nastri perforati per input/output e il successivo mark II finì l'era dei relé e iniziò l'era dell'elettronica.

### 1.2.2 Generazione 1 - Valvole termoioniche 1945-1955

- COLOSSUM sviluppata da Alan Turing dutante la seconda guerra mondiale, usata per decifrare i messaggi del dispositivo ENIGMA.
- ENIAC(Electronic Numerical Integrator And Computer) macchina che aveva registri decimali in grado di mantenere fino a un massimo numero di cifre decimali. Presentata in una scuola estiva dai creatori che ispirò molte altre macchine: ILLIAC, MANIAC, EDSAC, EDVAC, ...
- IAS(Institute for Advanced Study), creata da Neumann basandosi sull'architettura della EDVAC e delle altre macchine del tempo, costituita da 5 componenti fondamentali: memoria, ALU, unità di controllo (CU) e dispositivi di input/output. Il cervello della macchina era costituito dalla CU e dall'ALU, negli attuali computer appartengono ad un unico chip chiamato CPU(Central Processing Unit). L'ALU esegue operazioni aritmentiche e logiche attraverso un registro a 40 bit chiamato accumulatore. La CU dirige le operazioni. In genere queste macchine avevano in comune questa architettura detta macchina di Von Neumann: la CPU è composta da CU, LU e Accumulatore, memoria interna della CPU utilizzata per ocntenere gli operandi delle istruzioni eseguite dalla ALU; INPUT/OUTPUT costrituiscono l'interfacciamento del calcolatore verso l'esterno; Memoria che conserva sia il programma che i dati su cui deve lavorare il programma; bus di cominicazione è il canale che permentte la comunicazione tra tutte le precedenti unità.

Successivamente IBM inizia a ritagliarsi uno spazio nel commercio di macchine e schede perforate.

### 1.2.3 Generazione 2 - Transistor 1955-1965

Il transistoe è stato inventato da Bardeen nel 1948.

- TX-0(Transistorez eXperimental computer 0): ideato come dispositivo di test, evoluto nel TX-2. Non videro commercio.
- PDP-1: molto simile al TX-0, poteva eseguire 200.000 istruzioni al secondo.
- PDP-8: aveva un unico bus, chiamato omnibus, ovvero un insieme di cavi paralleli utilizzati per connettere i diversi componenti di un computer.
- IBM 7094: possedeva una memoria centrale di 25.536 parole da 36 bit.
- 1401 era una macchina in grado di leggere e scrivere nastri magnetici, leggere e perforare schede e stampare output.
- CDC 6600: la CPU era altamente parallela, dotata di unità che potevano lavorare parallelamente. Al suo interno aveva un certo numero di piccoli computer che lo aiutavano a gestire tutti i dettagli dei programmi e dell'input/output mentre la CPU macinava numeri.

### 1.2.4 Generazione 3 - Circuiti integrati 1955-1980

I circuiti integrati consistono in decine di transistor su un singolo chip. In questo modo si potevano avere computer più piccoli e allo stesso tempo più potenti. Nota di merito alla serie IBM 360 che introdusse la multiprogrammazione, grazie alla quale si potevano avere più programmi in memoria allo stesso tempo. Inoltre fu la prima serie di macchine in grado di emulare altri computer.

### 1.2.5 Generazione 4 - Integrazione a grandissima scala 1980-??

Con la tecnologia VLSI(Very Large Scale Integration si riescono a stampare milioni di transistor su un singolo chip (IBM PC, Apple Lisa, Intel 8080/8088/80386). Numerose aziende progettarono cloni di PC, mentre Apple introdusse il primo computer dotato di GUI.

### 1.2.6 Generazione 5 - Computer a basso consumo e computer invisibili

Computer invisibili che sono integrati in elettrodomesstici, orologi, ecc..

## 1.3 Tipologie di computer

Una veloce paronamica delle varie tipologie di computer:

- Computer usa e getta: chip come RFID(Radio Frequency Identificator) usati in oggetti, solitamente a basso costo, che vengono usati poche volte (esempio: cartoline melodiche).
- Microcontrollori: computer che sono integrati in apparecchiature che non sono vendute come elaboratori (Frigoriferi, televisioni, giocattoli, armi).
- Dispositivi mobili e da gioco: Computer con speciali capacità grafiche e sonore, non espandibili e non programmabili dall'utente (PS5, XBOX, ...)
- Personal Computer: un ampia gamma di macchine destinate all'informatica individuale (laptop, desktop, ...)
- Server: spesso sono versioni potenziate dei PC come server di rete destinati a usi aziendali e non.Esistono configurazioni dotate di mono o multiprocessore, dotate di centinaia di GB di memoria ad alta velocità.
- Raggruppamento di Workstation: normali PC connessi tra di loro mediante reti la cui velocità è nell'ordine dei GB al secondo. Eseguono software speciale che permette loro di lavorare in modo congiunto su uno stesso problema, spesso di tipo scientifico o ingegneristico.
- Mainframe: Sono computer con molte CPU ed alte capacità elaborative, affidabilità, sicurezza e costo. Hanno maggiore potenza di calcolo rispetto ai server e superiori capacità di I/O. Utilizzati da enti governativi e grandi aziende. Gli attuali MF ereditano le caratteristiche del primo MF ovvero IBM System/360.

| Chip | Data | MHz | N. di transistor | Memoria | Descrizione |
|---|---|---|---|---|---|
| 4004 | 4/1971 | 0,108 | 2300 | 640 | Primo microprocessore su un solo chip |
| 8008 | 4/1972 | 0,108 | 3500 | 16 KB | Primo microprocessore a 8 bit |
| 8080 | 4/1974 | 2 | 6000 | 64 KB | Prima CPU di uso generale su un solo chip |
| 8086 | 6/1978 | 5-10 | 29.000 | 1 MB | Prima CPU a 16 bit su un solo chip |
| 8088 | 6/1979 | 5-8 | 29.000 | 1 MB | Usato nel PC IBM |
| 80286 | 2/1982 | 8-12 | 134.000 | 16 MB | Introduzione della modalità protetta |
| 80386 | 10/1985 | 16-33 | 275.000 | 4 GB | Prima CPU a 32 bit |
| 80486 | 4/1989 | 25-100 | 1,2 M | 4 GB | Memoria cache da 8 KB integrata |
| Pentium | 3/1993 | 60-233 | 3,1 M | 4 GB | Due pipeline; istruzioni MMX nei modelli successivi |
| Pentium Pro | 3/1995 | 150-200 | 5,5 M | 4 GB | Cache integrata a due livelli |
| Pentium II | 5/1997 | 233-450 | 7,5 M | 4 GB | Pentium Pro con istruzioni MMX |
| Pentium III | 2/1999 | 650-1400 | 9,5 M | 4 GB | Istruzioni SSE per la grafica 3D |
| Pentium 4 | 11/2000 | 1300-3800 | 42 M | 4 GB | Hyperthreading; ulteriori istruzioni SSE |

Figura 1: Evoluzione delle CPU intel

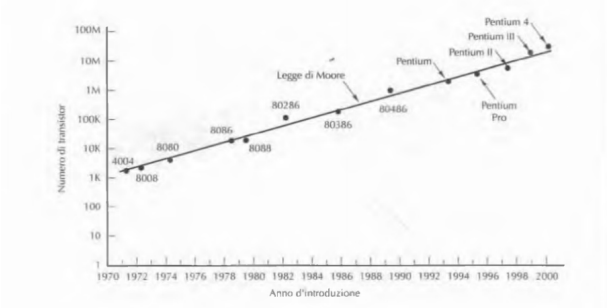



Figura 2: Legge di Moore per i chip della CPU intel

## 1.4 Esempi di famiglie di computer

Ci sono tre principali tipologie di architettura: ARM(spesso usata sui dispositivi mobile), AVR(diffusa nei microcontrollori economici) e x86(usata su pc e server).

### 1.4.1 Introduzione all'architettura x86

Nel 1968 fu fondata la Intel Corporation e da allora iniziarono a creare microprocessori su un solo chip. Il primo di essi fu il 4004, si noti la tabella che riassume le varie evoluzioni dei chip intel 1 Con il processore 80486 venne introdotta la memoria cache di 8KB usata per conservare le parole di memoria utilizzate con più frequenza.
Successivamente con il Pentium vennero introdotte l'insieme di istruzioni speciali MMX (MultiMedia eXtension) molto utili per velocizzare i calcoli per l'elaborazione di audio e video. Successivamente nei Pentium pro venne introdotta una cache a due livelli, una usata per le istruzioni più utilizzate e una per i dati. Un ulteriore aggiunta nei Pentium III furono le istruzioni SSE (Streaming SIMD Extrensions) per migliorare le prestazioni con la grafica 3D. Nel 2006 intel cambiò il marchio Pentium con i Core e introdusse il chip a due core: Core 2 Duo. Un'altra linea di chip introdotta da intel erano i Celeron, un marchio più economico e i Xeon usati per i server vista la cache più grande e un miglior supporto multiprocessore.
Notare come la legge di Moore associata al numero di bit di memoria, risulti valida anche per i chip della CPU 2. Usare transistor più piccoli permette di raggiungere frequenze più alte, ma allo stesso tempo richiede una tensione più elevata.

### 1.4.2 Introduzione all'architettura ARM

La società Acorn Computer sviluppò il primo computer con una CPU ARM. Successivamente dei processori ARM vennero implementati anche da Nintendo nel game boy advance e più recentemente nei smartphone e tablet. Nvidia creò un SoC Tegra 2 che ha due CPU ARM Cortex A9 implementata in alcune sue schede video.

### 1.4.3 Introduzione all'architettura AVR

Si tratta di un'architettura utilizzata nei sistemi integrati di fascia bassa. Nel 1996 nacque la CPU RISC a 8-bit chiamata AVR acquistata successivamente da Atmel. Ci sono tre classi di microcontrollori:

| Esp. | Valore esplicito | Prefisso | Esp. | Valore esplicito | Prefisso |
|---|---|---|---|---|---|
| $10^{-3}$ | 0,001 | milli | $10^{3}$ | 1.000 | Kilo |
| $10^{-6}$ | 0,000001 | micro | $10^{6}$ | 1.000.000 | Mega |
| $10^{-9}$ | 0,000000001 | nano | $10^{9}$ | 1.000.000.000 | Giga |
| $10^{-12}$ | 0,000000000001 | pico | $10^{12}$ | 1.000.000.000.000 | Tera |
| $10^{-15}$ | 0,000000000000001 | femto | $10^{15}$ | 1.000.000.000.000.000 | Peta |
| $10^{-18}$ | 0,000000000000000001 | atto | $10^{18}$ | 1.000.000.000.000.000.000 | Exa |
| $10^{-21}$ | 0,000000000000000000001 | zepto | $10^{21}$ | 1.000.000.000.000.000.000.000 | Zetta |
| $10^{-24}$ | 0,000000000000000000000001 | yocto | $10^{24}$ | 1.000.000.000.000.000.000.000.000 | Yotta |

Figura 3: Unità metriche

- TinyAVR: molto piccolo, i suoi pin devono assolvere un doppio compito; possono essere riprogrammati in fase di esecuzione per una qualsiasi delle funzioni digitali o analogiche supportate dal microcontrollore.
- MegaAVR: si trova nel sistema open-source Arduino, è dotato anche di supporto per l'I/O seriale, di orologi interni e di uscite analogiche programmabili.
- AVR XMEGA: incorpora un acceleratore per operazioni di crittografia e il supporto integrato per interfacce USB.

## 1.5 Unità metriche

Quando si fa riferimento ai dati si utilizza la base 2 poiché lo spazio di indirizzamento è sempre una potenza di due. Quindi 1 kB di memorica contiene $2^16$=1024 Byte. Guardare lo schema per tutti i prefissi 3.

# 2 Organizzazione dei sistemi di calcolo

Un calcolatore digitale è un sistema in cui processori, memorie e dispositivi periferici sono connessi tra loro.

## 2.1 Processi

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, eseminandole ed eseguendole una dopo l'altra. I componenti sono connessi fra di loro mediante un bus. La CPU contiene un unità di controllo e un unità aritmetico-logica. Oltre a queste contiene una piccola memoria costituita da un certo numero di registri. Il registro più importante è il contatore d'istruzioni o Program Counter (PC), che punta alla successiva istruzione che dovrà essere eseguita. Un altro registro è il registro istruzione corrente, o Instruction Register (IR), contenente l'istruzione che si trova in fase di esecuzione.

### 2.1.1 Organizzazione della CPU

Il data path di una CPU di Von Neumann è tipicamente composta da dei registra da 1 a 32 bit; una ALU (Arithmectic Logic Unit) e da alcuni bus che connettono fra loro le diverse parti. I registri alimentano due registri di input della ALU mentre questa è occupata nell'esecuzione di alcune computazioni (addizioni/sottrazioni su input, il risultato viene memorizzato nei registri o copiato in memoria in un secondo momemnto).
La maggior parte delle istruzioni può essere divisa in:

- Registro-memoria (necessita di una fase di caricamento delle parole della memoria nei registri): permettono di prelevare parole di memoria per portarle all'interno dei registri (o viceversa), dove sono utilizzabili, per esempio, come input della ALU per effettuare istruzioni successive.
- Registro-registro (gli operandi sono già pronti nei registri): preleva due operandi dai registri, li porta all'interno dei registri di input della ALU, esegue su di loro una qualche operazione e ne memorizza il risultato in uno dei registri.

Il processo che consiste nel portare i due operandi attraverso la ALU e nel memorizzare il risultato è chiamato ciclo del percorso dati.

### 2.1.2 Esecuzione dell'istruzione

I passi per l'esecuzione di ogni istruzione della CPU generalmente può essere riassunta con il seguente **ciclo esecutivo delle istruzioni** (o **ciclo di prelievo-decodifica-esecuzione**):

1. Prelevare la successiva istruzione dalla memoria per portarla nell'IR;
2. Modificare il PC per farlo puntare all'istruzione seguente;
3. Determinare il tipo dell'istruzione appena prelevata;
4. Se l'istruzione usa una parola in memoria, determinare dove si trova;
5. Se necessario, prelevare la parola per portarla in un registro della CPU;
6. Eseguire l'istruzione;
7. Torna al punto 1 per iniziare l'esecuzione dell'istruzione successiva.

Questo ciclo può anche essere tradotto ed eseguito da programmi che prelevano, esaminano ed eseguono le proprie istruzioni, detti **interpreti**.
Nel passato gli interpreti erano favoriti per via delle **memorie di controllo** che erano usate per memorizzare l'interprete. In questo modo le microistruzioni, ovvero le istruzioni che l'interprete usava per eseguire un'istruzione del programma da interpretare, impiegavano pochi millisecondi in più rispetto a quanto impiega il programma principale. Senza le memorie di controllo si impiegava circa 6 volte il tempo impiegato con la memori di controllo.

### 2.1.3 RISC contro CISC

Negli anni '80 si progettarono alcune CPU che non utilizzavano l'interpretazione. Ci sono varie strategie di progettazione:

- RISC (Reduced Instruction Set Computer): si base su poche istruzioni semplici, la cui esecuzione possa essere completata velocemente (un solo ciclo di clock idealmente). I vantaggi sono: maggiore velocità per ciclo di clock, Consumo energetico più basso. Gli svantaggi sono: codice macchina più lungo (più istruzioni per fare la stessa cosa), Richiede compilatori più intelligenti.
- CISC (Complex Instruction Set Computer): si basa su molte istruzioni complesse: anche centinaia, con operazioni ad alto livello (es. "moltiplica e somma"). Una singola istruzione può compiere operazioni multiple (es. accesso memoria + calcolo). Microcodice interno che traduce le istruzioni complesse in micro-operazioni più semplici. I vantaggi sono programmi più compatti e una buona compatibilità con linguaggi ad alto livello. Gli svantaggi sono: Hardware più complesso, esecuzione più lenta per via della decodifica complessa.
- Ibrido: a partire dal x486, le CPU intel contengono un sottoinsieme di istruzioni RISC (quelle più comuni) che possono essere eseguite in un singolo ciclo nel datapath, mentre le altre complesse sono interpretate secondo la classica modalità CISC.

### 2.1.4 Principi di progettazione dei calcolatori moderni

Esiste un insieme di principi di progettazione, talvolta chiamati principi di progettazione RISC, che i progettisti cercano di seguire il più possibile (salvo vincoli esterni):

- **Tutte le istruzioni sono eseguite direttamente dall'hardware:** Le istruzioni non devono essere interpretate. Per le architettura CISC, quelle istruzioni più complesse (e più rare) possono essere suddivise in parti ed eseguite, successivamente, come sequenze di microistruzioni.
- **Massimizzare la frequenza di emissione delle istruzioni:** Il parallelismo gioca un ruolo essenziale nelle prestazioni di un computer. Infatti è possibile emettere un gran numero di lente istruzioni in un breve intervallo di tempo solo se si riescono a eseguire più istruzioni allo stesso tempo.
- **Le istruzioni devono essere facili da decodificare:** Un limite critico sulla frequenza di emissione delle istruzioni è dato dal processo di decodifica, che deve essere effettuato per ogni singola istruzione allo scopo di determinare le risorse necessarie. Bisogna rendere le istruzioni regolari, di lunghezza fissa e con pochi campi. Meno formati di istruzioni ci sono, meglio è.
- **Solo le istruzioni Load e Store fanno riferimento alla memoria:** L'operazione di spostamento degli operandi dalla memoria ai registri può essere compiuta separatamente mediante apposite istruzioni. Dato che l'accesso alla memoria può richiedere un tempo considerevole, queste operazioni possono essere efficientemente sovrapposte all'esecuzione di altre istruzioni. Tale osservazione porta alla conclusione che soltanto le istruzioni LOAD e STORE dovrebbero far riferimento alla memoria, mentre tutte le altre dovrebbero operare esclusivamente sui registri.
- **Molti registri disponibili:** Dato che l'accesso alla memoria è relativamente lento occorre prevedere molti registri (almeno 32) di modo che, una volta prelevata la parola, possa essere mantenuta nel registro fintanto sia necessario. È particolarmente inefficiente trovarsi senza registri liberi, in quanto obbliga a scaricare in memoria tutti i valori dei registri per poi ricaricarli.

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 prestazioni. Il parallelismo si può ottenere in due diversi modi:

- **Parallelismo a livello di istruzione:** il parallelismo è sfruttato all'interno delle istruzioni per ottenere un maggior numero di istruzioni al secondo;
- **Parallelismo a livello di processore:** più CPU collaborano per risolvere lo stesso problema.

### 2.1.5 Parallelismo a livello d'istruzione

- **Pipeline:** Il collo di bottiglia della velocità di esecuzione delle istruzioni è rappresentato dal prelievo delle istruzioni dalla memoria. In passato, per alleviare questo problema, le istruzioni venivano prelevate in anticipo e inserite in dei registri, chiamati buffer di prefetch, dove venivano memorizzate le istruzioni le quali potevano essere prelevate senza dover attendere una lettura della memoria. In pratica la tecnica di prefetching divide l'esecuzione dell'istruzione in due parti: il prelievo dell'istruzione e la sua esecuzione effettiva.
  Il pipeline divide l'esecuzione di un'istruzione in un numero maggiore di parti che possono essere eseguite in parallelo; ciascuna di queste parti è gestita da componenti hardware dedicati. Un seguito un esempio di pipeline a cinque stadi4

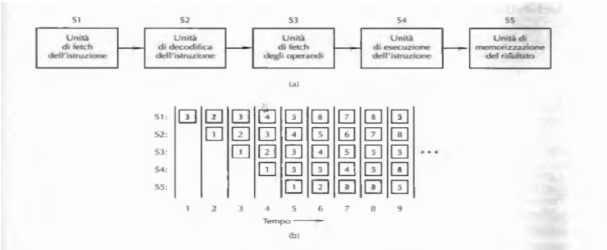



Figura 4: Pipeline a 5 stadi

L'uso della pipeline permette di bilanciare la latenza (il tempo che un'istruzione impiega per essere elaborata) e la larghezza di banda del processore (i MIPS, Milion Instrutcion per second, della CPU). Più nello specifico: con un ciclo di clock di T ns e una pipeline a n stadi, la latenza è di nT ns, poiché ogni istruzione attraversa n stadi, ognuno dei quali richiede T ns. Dato che ogni ciclo di clock viene eseguita un istruzione e che ci sono $10^9/T$ cicli di clock, vengono eseguite $10^9/T$ istruzioni al secondo (se T=2 500 milioni di istruzioni al secondo.

- **Processori con più pipeline:** Nel caso di processori con più pipeline, è presente una singola unità di fetch che preleva due istruzioni alla volta e le inserisce nelle pipeline, ognuna della quali dotata di ALU. Per poter eseguire le istruzioni in parallelo non devono esserci conflitti che devono essere gestiti o eliminati durante l'esecuzione per mezzo di componenti hardware. Intel sviluppo il processore 486 che aveva due pipeline; la prima, pipeline u, poteva eseguire una qualsiasi istruzione Pentium. La seconda, pipeline v, poteva invece eseguire solamente semplici istruzioni su interi. Erano presenti regole che limitavano l'esecuzione di istruzioni in parallelo nel caso di istruzioni non semplici o incompatibili.

- **Architettura superscalare:** Invece di avere due pipeline duplicate, per risparmiare sui componenti da duplicare, intel per prima adottò una singola pipeline con più unità funzionali in corrispondenza di alcuni stadi. Per renderlo ottimale è necessario che la velocità di emissione dello stadio precedente a quello duplicato sia superiore, in questo modo lo stadio successivo puà eseguire in parallelo più istruzioni.

### 2.1.6 Parallelismo a livello di processore

Il parallelismo a livello di processore può aumentare le prestazioni di un fattore 50, 100 o più, cosa che il parallelismo a livello d'istruzione riesce a malapena ad arrivare a un fattore di 5 o 10. Esistono tre approcci:

- Computer con parallelismo sui dati: ci sono fondamentalmente due approcci:
  1. Procesori SIMD (Single Instruction Multiple Datastream): consiste in un elevato numero di processori identici che eseguono la stessa sequenza d'istruzioni su insiemi diversi di dati. Le moderne GPU si affidano in larga misura all'elaboraziione SIMD per offrire grande potenza di calcolo con pochi transistor (poiché la maggior parte degli algoritmi sono regolari con operazioni ripetute su pixel, vertici, ecc).
  2. Processori Vettoriali: eseguono in modo molto efficiente una stessa sequenza di operazioni su coppie di dati, anche se, a differenza dei processori SIMD, tutte le operazioni di addizione sono eseguite da un unico sommatore strutturato in pipeline.

Entrambi lavorano su array di dati, ma mentre il SIMD sfrutta sommatori quanti sono gli elementi dei vettori, il processore vettoriale usa un registro vettoriale che consiste in un insieme di registri convenzionali caricati in memoria in una singola istruzione. Questo fa si che quando si fa una somma tra vettori viene fuori un vettore che può essere memorizzato in un registro vettoriale o utilizzato direttamente come operando.

- Multiprocessori: è un sistema composto da più CPU complete con memoria in comune. Poiché ciascuna CPU può leggere o scrivere qualsiasi zona della memoria comune, le CPU devono sincronizzarsi via software. In questo caso le CPU hanno la necessità di interagire in modo così profondo che il sistema è detto fortemente accoppiato (tightly coupled). L'approccio più semplice è avere un singolo bus con più cpu, tutte connesse in un'unica memoria. Per ridurre la contesa della memoria, ogni processore è dotato di una memoria locale che non è accessibile agli altri. Questa è usata per mantenere il codice del programma e quei dati che non devono essere condivisi. Questa memoria non utilizza il bus principale.

- Multiprocessori con molte CPU sono difficili da realizzare, per via del problema delle connessioni di ciascuna CPU verso la memoria comune. I progettisti hanno superato il problema abbandonando il concetto di memoria comune e realizzando un elevato numero di CPU interconnesse, ciascuna con la propria memoria privata. Le CPU in un multicomputer sono accoppiate in modo lasco (loosely coupled) e comunicano attraverso scambi di messaggi. In architetture grandi la completa interconnessione non è fattibile così sono utilizzate topologie differenti come la griglia, l'albero o l'anello.

## 2.2 Memoria Principale

La memoria è quella parte del calcolatore in grado di memorizzare i programmi e i dati.

### 2.2.1 Bit

L'unità di base della memoria è la cifra binaria, i bit. Il bit può assumere valori 0 e 1. I bit sono efficienti poiché i valori digitali possono essere memorizzati usando la tensione (acceso / spento). Un computer ragiona unicamente interpretando gruppi di bit, cioè comandi rappresentati da sequenze di "zero" e "uno". 8 bit rappresentano 1 byte. Il BCD (Binary Code Decimal) è un codice utilizzato da alcuni sistemi di calcolo, come i grandi mainframe IBM, per rappresentare i numeri decimali.

- Scopo: Il BCD è un codice che permette a certi computer di utilizzare un'aritmetica sia decimale che binaria.

- Codifica: Il BCD utilizza 4 bit per memorizzare una singola cifra decimale.

- Capacità: Poiché 4 bit possono fornire 24=16 combinazioni, nel sistema BCD vengono utilizzate le 10 combinazioni per rappresentare le cifre decimali da 0 a 9. Le restanti sei combinazioni non vengono utilizzate.

Il BCD è poco efficiente rispetto al binario poiché 16 bit in BCD possono memorizzare solo 10.000 combinazioni (0 a 9999), mentre in binario puro possono memorizzare 65.536 combinazioni diverse (216)

### 2.2.2 Indirizzi di memoria

Le memorie sono costituite da un certo numero di celle (o locazioni) ciascuna delle quali può memorizzare informazioni. Ciascuna cella ha un numero, chiamato indirizzo, attraverso il quale il programma può riferirsi ad essa. Se una memoria ha n celle i suoi indirizzi andranno da 0 a n-1. Tutte le celle contengono lo stesso numero di bit; se una cella è costituita da k bit, essa può contenere una qualsiasi delle $2^k$ combinazioni di bit. I calcolatori che usano un sistema binario esprimono gli indirizzi di memoria in notazione binaria; se un indirizzo ha m bit, il massimo numero di celle indirizzabili è $2^m$. Lo standard impone che la dimensione di una cella sia di 8 bit = 1 byte. I byte sono raggruppati in parole (word) la cui lunghezza varia a seconda dell'architettura del computer (ad esempio, una parola da 32 bit contiene 4 byte). La lunghezza della parola è significativa poiché la maggior parte delle istruzioni opera su intere parole, ad esempio, sommando due parole intere

### 2.2.3 Ordinamento dei byte

L'ordinamento dei byte si riferisce al modo in cui un intero multi-byte viene memorizzato (l'ordine in cui sono posizionati i byte di ordine superiore e inferiore all'interno della parola). Esistono due principali schemi di endianness: big-endian (da destra a sinistra) e little-endian (da sinistra e destra). Questa differenza può creare problemi di compatibilità (il "problema della referenza in avanti") quando i dati devono essere trasferiti tra macchine con endianness diverse. Per superare questo problema, è necessario che il software inverta l'ordine dei byte durante il trasferimento.

### 2.2.4 Codice correttori

Occasionalmente le memorie dei calcolatori possono commettere degli errori per via di picchi di tensione sulle linee di alimentazione o per altre cause. Per proteggersi da tali errori, alcune memorie utilizzano dei codici di rilevazione e/o di correzione degli errori. Quando si impiegano questi codici vengono aggiunti dei bit extra a ogni parola di memoria secondo una modalità particolare. Quando una parola viene letta dalla memoria, si controllano questi bit aggiuntivi per vedere se si è verificato un errore.
Supponiamo che una parola di memoria consista di m bit di dati ai quali aggiungiamo r bit di controllo; sia quindi n = m + r la lunghezza totale. Un'unità di n bit contenente m bit di dati e r bit di controllo è spesso chiamata **parola di codice (codeword)** a n bit.
Date due parole di codice, diciamo 10001001 e 10110001, è possibile determinare il numero di bit corrispondenti rispetto ai quali differiscono. In questo caso la differenza è di 3 bit. Per determinare il numero di bit diversi, bisogna semplicemente calcolare l'**OR ESCLUSIVO (XOR)** tra i bit delle due parole e contare quanti bit valgono 1 nel risultato; questo valore è chiamato **distanza di Hamming**. Il suo significato principale è che, se fra due parole di codice vi è una distanza di Hamming pari a d allora saranno necessari d errori singoli per trasformare una parola nell'altra.
Gli r bit di controllo devono essere in grado di indicare se la parola è corretta o se c'è un errore in uno degli m bit della parola. Tra le $2^n$ combinazioni della codeword solo $2^m$ saranno valide (le combinazioni della parola di memoria a m bit). Quindi se dovesse uscire una delle $2^n - 2^m$ combinazioni non valide il calcolatore saprà che è avvenuto un errore. L'algoritmo che calcola i bit di controllo, ci fornisce la lista completa delle parole di codice lecite, da cui è possibile calcolare la distanza di hamming minima, la distanza di hamming dell'intero codice.
Tutto dipende dalla distanza di hamming: per **rilevare** d errori singoli è necessaria una parola con distanza d+1, in questo modo non esiste alcun modo in cui d errori singoli possano cambiare una parola valida in un'altra parola di codice valida; per **correggere** d errori singoli è necessario un codice con distanza 2d+1; questa distanza assicura che le parole codice valide siano sufficientemente distanti tra loro, in modo che anche con d cambiamenti di bit, il codice originale rimanga il più vicino alla parola ricevuta e possa essere univocamente determinato.
Ad esempio, aggiungendo un singolo **bit di parità** ad una parola in modo tale che il numero di 1 presenti sia sempre pari, il calcolatore può facilmente capire se c'è un errore verificando se il numero di 1 è pari o dispari. Un codice di questo tipo ha distanza di Hamming 2, dato che ogni errore singolo genera una parola di codice la cui parità è errata. Per passare da una parola di codice valida ad un'altra servono 2 errori singoli, il programma non riesce a correggere l'errore.
Ipotizziamo di voler progettare un codice con m bit di dati e r di controllo capace di correggere tutti gli errori singoli, sia n=m+r. Ogni parola codice valida di lunghezza n (ovvero $2^m$ parole totali) ha n parole illegali che si trovano a una distanza di Hamming pari a 1 da essa (risultanti dall'inversione di un singolo bit). Se il codice ha $2^m$ parole codice valide, e ciascuna richiede n+1 combinazioni di bit uniche (la parola stessa più le n possibili versioni errate a singolo bit), la condizione necessaria e sufficiente è che:

$$(n+1)2^m \leq 2^n \iff (m+r+1)2^m \leq 2^{m+r} \iff m+r+1 \leq 2^r$$

Questa è una relazione tra il numero minimo di bit di controllo e la lunghezza della parola di memoria.5

| 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 5: Numero di bit di controllo per un codice che può correggere errori singoli

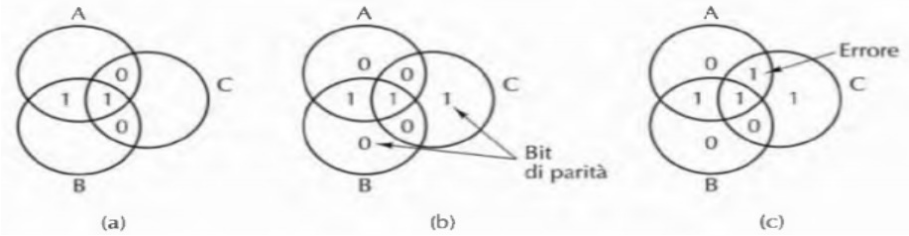



Figura 6: (a) codifica di 1100 (b) aggiunta parità pari (c)Errore in AC

Ora illustriamo **l'algoritmo di Hamming**:
può essere usato per costruire codici a correzione di errore per parole di memoria dalla dimensione arbitraria. Data una parola a m bit, vengono aggiunti r bit di parità (di controllo) nelle posizioni a potenza di 2. I bit sono numerati a partire da 1, con 1 bit nella posizione più significativa. Ciascun bit di parità controlla le posizioni che scritte in binario, hanno un "1" nella posizione corrispondente al bit di controllo. Esempio:

- $P_1(pos.2^0 = 1)$: Controlla tutte le posizioni che hanno l'ultimo bit a 1 (1, 3, 5, 7...).
- $P_2(pos.2^1 = 2)$: Controlla tutte le posizioni che hanno il secondo bit a 1 (2, 3, 6, 7...).
- $P^3(pos.2^2 = 4)$: Controlla tutte le posizioni che hanno il terzo bit a 1 (4, 5, 6, 7...).

In generale il bit di posto b è controllato dai bit $b_1, b_2, \ldots, b_n$ tali che $b_1 + b_2 + \cdots + b_n = b$. Ogni bit di dati è sorvegliato da almeno due bit di controllo diversi. È questo "incrocio" che permette la correzione.

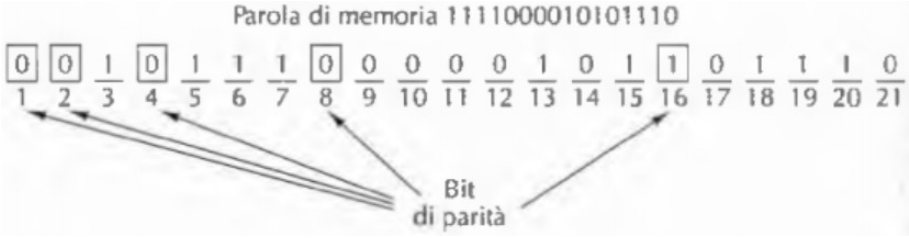



Figura 7: Costruzione del codice di hamming per la parola di memoria 1111000010101110 aggiungendo 5 bit di controllo ai 16 bit di dati

Un esempio7 su questa parola 1111000010101110 di 16 bit alla quale sono stati aggiunti 5 bit di parità diventando 001011100000101101110 a 21 bit. Ora supponiamo che il bit 5 subisca un errore e venga invertito: 001011100000101101110. Controllando i 5 bit di parità si otterrebbe:

- bit di parità 1 non corretto (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contengono cinque 1)
- bit di parità 2 corretto (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contengono sei 1)

- bit di parità 4 non corretto (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contengono cinque 1)
- bit di parità 8 corretto (8, 9, 10, 11, 12, 13, 14, 15 contengono due 1)
- bit di parità 16 corretto (16, 17, 18, 19, 20, 21 contengono quattro 1).

Il numero totale di 1 nelle posizioni 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 e 21 dovrebbe essere un numero pari dato che si sta usando la parità pari. Il bit non corretto deve essere uno dei bit controllati dal bit di parità 1, cioè uno fra i bit 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 e 21.

Il bit di parità 4 è errato, il che significa che uno fra i bit 4, 5, 6, 7, 12, 13, 14, 15, 20 e 21 non è corretto. L'errore deve essere uno dei bit presenti in entrambe le liste, cioè il bit 5, 7, 13, 15 oppure 21. Tuttavia il bit 2 è corretto e quindi i bit 7 e 15 sono eliminati. Analogamente il bit 8 è corretto, eliminando così il bit 13. Infine, dato che anche il bit 16 è corretto, va eliminato pure il 21. Il solo bit rimasto è il bit 5, che è quindi quello in cui si è verificato l'errore; dato che è stato letto come 1, esso deve assumere il valore 0. Questa successione di considerazioni consente di correggere gli errori. Un semplice metodo per trovare i bit errati consiste nel calcolare inizialmente tutti i bit di parità. Se sono tutti corretti allora non si è verificato alcun errore (oppure più di uno). Successivamente si sommano tutti i bit di parità errati, contando 1 per il bit 1, 2 per il bit 2, 4 per il bit 4, e così via, e la somma risultante corrisponde alla posizione del bit errato. Se per esempio i bit di parità 1 e 4 sono errati, ma 2, 8 e 16 sono corretti, significa che il bit 5 (1 + 4) è stato invertito.

### 2.2.5 Memoria cache

La memoria cache è un tipo di memoria piccola e molto veloce il cui scopo principale è immagazzinare le parole di memoria (dati e istruzioni) che sono state utilizzate più di recente. La **cache** è stata introdotta per risolvere il problema della disparità di velocità tra la CPU, che è diventata sempre più veloce, e la memoria principale (RAM), che è rimasta relativamente più lenta.
L'idea di base è semplice: le parole di memoria usate più di frequente sono mantenute all'interno della cache. Quando la CPU necessita di una parola, la cerca nella cache e, solo nel caso in cui essa non sia presente (**cache miss**), la richiede alla memoria centrale. È possibile ridurre drasticamente il tempo medio di accesso se una frazione significativa delle parole è presente nella cache. Il successo o il fallimento dipendono quindi da quali parole sono presenti nella cache.
L'efficacia della memoria cache si basa sul **principio di località** (locality principle), che afferma che i riferimenti di memoria di un programma non sono casuali ma tendono ad aggregarsi:

- **Località Temporale:** se una posizione di memoria è stata utilizzata di recente, è probabile che venga riutilizzata a breve.
- **Località Spaziale:** Se una posizione di memoria è stata utilizzata, è probabile che anche le posizioni di memoria adiacenti vengano utilizzate presto.

Quando si verifica un cache miss, l'intera linea di cache (cache line), ovvero un blocco di parole consecutive, viene trasferita dalla memoria principale alla cache, sfruttando il principio di località spaziale.
Per disporre sistemi di memoria più sofisticati, è possibile avere anche tre o più livelli di cache:

- (L1): generalmente è separata in cache istruzioni (L1-I) e cache dati (L1-D), ed è spesso inclusa direttamente nel chip della CPU.
- (L2): non si trova nel chip del processore, ma che può essere inclusa all'interno del suo involucro; questa cache, di solito è **unificata** (contiene sia dati che istruzioni) e la sua dimensione può variare dai 512 Kb a 1 Mb.
- (L3): Nei sistemi più recenti, può essere presente una cache più grande, a volte condivisa tra più core.

### 2.2.6 Tipi di memorie

La memoria principale (**RAM**) è generalmente realizzata con DRAM (Dynamic RAM). Le **DRAM** utilizzano un transistor e un condensatore per bit e necessitano di essere rinfrescate (**refreshed**) periodicamente, poiché la carica si disperde. Le DRAM offrono un'alta densità di bit per chip, ma sono relativamente lente (decina di nanosecondi). Per costruire la memoria cache, che è estremamente veloce, si utilizza la **SRAM** (Static RAM). Le SRAM non richiedono il refresh e mantengono i dati finché sono alimentate; sono molto

più veloci (un nanosecondo o meno) rispetto alle DRAM. Esistono anche le memorie **ROM** (Read-Only Memory), che sono non volatili, ovvero mantengono i dati anche senza alimentazione. Esempi di ROM includono le PROM (programmabili una sola volta), le EPROM (cancellabili con luce ultravioletta), le EEPROM (cancellabili elettricamente) e la memoria Flash,. La **memoria Flash** è cancellabile a blocchi e riscrivibile ed è molto usata in dispositivi come le fotocamere digitali e per sostituire i dischi (SSD). Storicamente, i chip di memoria sono stati raggruppati su schede, come i **SIMM** (Single Inline Memory Module) o, più comunemente oggi, i **DIMM** (Dual Inline Memory Module), come le DDR3 DIMM.

## 2.3 Memoria Secondaria

### 2.3.1 Gerarchie di memoria

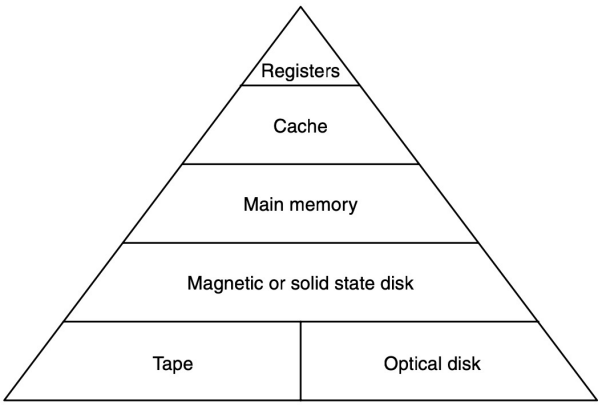



Figura 8: Gerarchia di memoria a 5 livelli

Nella gerarchia di memoria, le memorie secondarie si trovano al di sotto della memoria principale e della cache. Scendendo nella gerarchia:

- Il tempo di accesso (access time) diventa maggiore.
- La capacità di archiviazione (storage capacity) aumenta.
- Il costo per bit (misurato in dollari/gigabyte) diminuisce.

Nella parte superiore della gerarchia si trovano i **registri** della CPU, ai quali si può accedere alla stessa velocità della CPU. Successivamente vi è la memoria **cache**, la cui dimensione può variare da 32 KB fino ad alcuni megabyte. La **memoria centrale** è il passo successivo e la sua dimensione è compresa tra 16 MB per i sistemi più economici fino a decine di gigabyte per quelli professionali. Successivamente troviamo i **dischi magnetici o dischi a stato solido**, la vera forza lavoro per quanto riguarda la memorizzazione permanente. Infine ci sono i **nastri magnetici** e i **dischi ottici** utilizzati per l'archiviazione.

### 2.3.2 Dischi magnetici

I dischi magnetici (**hard disk**) sono il tipo di memoria secondaria più comune e sono composti da uno o più piatti di alluminio ricoperti di materiale magnetico.

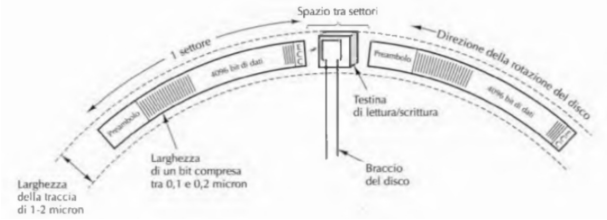



Figura 9: Una porzione di una traccia di un disco meccanico

I dati sono registrati su anelli concentrici chiamati **tracce**. Ogni traccia è divisa in **settori** di lunghezza fissa, che tipicamente contengono 512 byte di dati, preceduti da un preambolo per la sincronizzazione

della testina. Dopo i dati segue un codice per la correzione di errore (ECC, Hamming o Reed-Solomon).
La geometria di un disco include bracci mobili con **testine** di lettura/scrittura che fluttuano su un cuscino d'aria sulla superficie del piatto. Una corrente che passa attraverso la testina magnetizza la superficie che si trova al di sotto, orientando le particelle magnetiche in direzione opposta a seconda del verso della corrente. Quando la testina passa sopra un'area magnetizzata, viene indotta nella testina una corrente positiva o negativa rendendo così possibile la lettura dei bit memorizzati.
L'insieme delle tracce alla stessa distanza dal centro su tutti i piatti è chiamato **cilindro**.
Tra due settori consecutivi vi è un piccola area chiamata **spazio tra settori**.
La maggior parte dei dischi consiste di più piatti impilati verticalmente in cui ciascuna superficie ha il proprio braccio e la propria testina. I dischi dei PC attuali hanno dai 6 ai 12 piatti, mettendo quindi a disposizione dalle 12 alle 24 superfici registrabili.
Le prestazioni dei dischi si misurano con il **tempo di accesso**, che combina il **tempo di ricerca** (seek time) per muovere la testina al cilindro corretto e la latenza rotazionale per aspettare che il settore desiderato passi sotto la testina.
A ogni disco è associato un processore dedicato chiamato controllore del disco, il quale deve accertare i comandi dal software.

### 2.3.3 Dischi IDE

I dischi **IDE** (Integrated Drive Electronics) sono l'evoluzione dei dischi per personal computer, dove l'elettronica del controller è stata integrata con il drive.

- I primi dischi IDE utilizzavano convenzioni **BIOS** (Basic Input Output System) con limiti sul numero di testine, cilindri e settori, portando a una capacità massima iniziale di 504 MB.
- Questi limiti sono stati superati attraverso l'introduzione dell'**LBA** (Logical Block Addressing), che numera i settori in modo sequenziale partendo da zero, eliminando la necessità di conoscere la geometria fisica del disco.

### 2.3.4 Dischi SCSI

I dischi **SCSI** (Small Computer System Interface) sono un'alternativa ai dischi IDE, che offrono un'interfaccia e tassi di trasferimento dati molto più veloci.

- I sistemi SCSI utilizzano un bus e un protocollo più complessi per comunicare. Le versioni più recenti, come Ultra320 SCSI, hanno aumentato notevolmente la velocità.
- Similmente a IDE, i dischi SCSI non sono concettualmente diversi (anch'essi sono organizzati in cilindri, tracce e settori), ma l'interfaccia e la gestione dei trasferimenti sono più sofisticate.

### 2.3.5 Raid

RAID (Redundant Array of Independent Disks, ora reinterpretato come **Redundant Array of Inexpensive Disks** o Independent Disks) è una tecnica che utilizza più dischi per migliorare le prestazioni e/o la tolleranza ai guasti.
I dati sono distribuiti (stripping) su più dischi per consentire operazioni parallele e aumentare la velocità. Al software appaiono come un singolo disco. Esistono diversi livelli di RAID (RAID 0, 1, 2, 3, 4, 5, 6), ciascuno con diverse strategie per la distribuzione dei dati e l'uso di ridondanza (parità o mirroring) per la tolleranza ai guasti.

- **Raid 0 (Striping):** I dati vengono suddivisi in stripes consecutive e distribuiti in modo round-robin su più dischi. I settori da 0 a k - 1 che compongono la strip 0, i settori da k a 2k - 1 la strip 1 e così via; il RAID livello 0 lavora meglio quando le richieste sono di grandi dimensioni. Lo svantaggio è che non ha nessuna ridondanza; se un disco fallisce, tutti i dati sono persi. Non è considerato un vero RAID.
- **Raid 1 (Mirroring):** Duplicazione completa (mirroring) dei dati su un set di dischi. Ogni strip è scritta due volte. Eccellente tolleranza ai guasti e buone prestazioni in lettura (fino al doppio), ma è costoso (il 50% dello spazio è sprecato per la ridondanza).

- **Raid 2:** Opera a livello di parola o byte (non a livello di settore/strip). Aggiunge bit di controllo di codice Hamming per la correzione degli errori e richiede che i dischi siano sincronizzati. Richiede un elevato numero di dischi e un controller complesso; offre un basso numero di richieste I/O al secondo. Lo schema ha senso soltanto se si utilizza un numero significativo di unità (anche con 32 dischi di dati e 6 dischi di parità l'overhead è del 19
- **Raid 3:** è una versione semplificata del RAID livello 2. Il bit di parità viene calcolato per ogni parola di dati e poi scritto su un apposito disco. Dato che le parole di dati sono distribuite su più unità, anche in questo caso, come per il RAID livello 2, i dischi devono essere sincronizzati. Permette la correzione di errori singoli (se il disco guasto è noto) e offre un throughput elevato, ma offre un basso numero di richieste I/O al secondo.
- **Raid 4:** Utilizza stripes (come RAID 0 e 1), ma con un disco di parità dedicato dove viene scritta la parità XOR di tutti gli stripes. Se un disco si guasta è possibile ricalcolare i byte persi grazie al disco di parità. Questo schema protegge dalla perdita di un disco, ma ha prestazioni scarse quando si aggiornano piccole quantità di dati (ogni modifica richiede quattro operazioni (due letture e due scritture) per aggiornare sia i dati che la parità). Il disco di parità può inoltre diventare un collo di bottiglia, a causa del grande carico di lavoro che pesa su di esso.
- **Raid 5:** Simile a RAID 4, ma distribuisce le informazioni di parità su tutti i dischi, eliminando il collo di bottiglia del disco di parità dedicato. Tuttavia, quando si verifica un guasto a un disco, la ricostruzione del suo contenuto è un processo complesso.

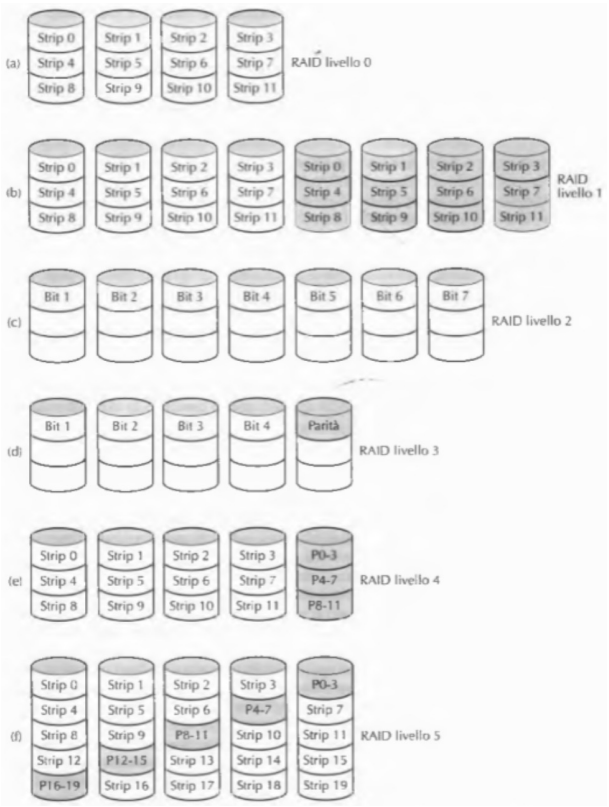



Figura 10: Raid dal livello 0 al livello 5; i dischi in grigio sono dischi di backup e per la parità.

### 2.3.6 Dischi a stato solido

Sono dischi basati su memoria flash non volatile e si sono diffusi come alternativa ad alta velocità ai tradizionali dischi magnetici.

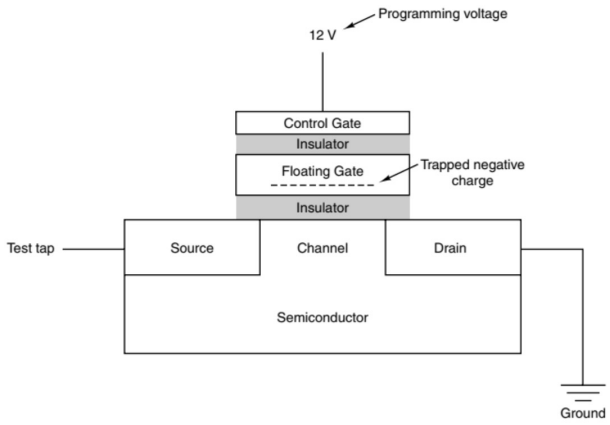



**Figure 2-24.** A flash memory cell.

Figura 11: Disco con memoria flash

I dischi flash sono fatti di celle di memoria flash a stato solido. Queste celle sono costituite da un singolo transistor flash speciale. Per programmare il bit flash, si applica alla porta di controllo una tensione elevata che accelera il processo di iniezione a caldo nella porta flottante. Gli elettroni vengono intrappolati nella porta flottante, portando così una carica negativa interna al transistor flash. La carica negativa aumenta la tensione necessaria ad accendere il transistor flash e, testando se il canale si accende con una tensione alta o bassa, è possibile determinare se porta floattante è carica oppure no, con conseguente valore di 0 o 1 della cella flash.

- Vantaggio: Prestazioni superiori (2-3 volte più veloci) e tempo di ricerca nullo. Sono ideali per dispositivi mobili in quanto non hanno parti in movimento.
- Svantaggio: Costo per gigabyte significativamente più alto e durabilità limitata a circa 100.000 scritture per cella. Per mitigare questo, si usa l'algoritmo di wear leveling.

### 2.3.7 CD-ROM

Il funzionamento dei CD-ROM si basa sulla distinzione fisica tra **pit** (fossette) e **land** (aree piatte).

- **Meccanismo di lettura:** Un laser a infrarossi a bassa potenza viene proiettato sul disco. I pit sono progettati per avere un'altezza pari a un quarto della lunghezza d'onda della luce laser. Questo fa sì che la luce riflessa da un pit sia sfasata di mezza lunghezza d'onda rispetto a quella riflessa dal land circostante, causando un'interferenza distruttiva. Il fotorivelatore traduce questa assenza di luce come un pit.
- **Codifica dei bit:** Contrariamente a quanto si potrebbe pensare, non è il pit in sé a rappresentare lo 0 o l'1. Il sistema rileva come 1 la transizione pit-land o land-pit, mentre l'assenza di transizione viene interpretata come 0.
- **Velocità Lineare Costante (CLV):** Per garantire un flusso di dati uniforme, i CD-ROM ruotano a velocità variabile. La rotazione è più veloce vicino al centro (530 RPM) e rallenta man mano che la testina si sposta verso l'esterno (fino a 200 RPM), mantenendo una velocità lineare costante di 120 cm/sec.

### 2.3.8 CD-Riscrivibili e CD-Registrabili

Poiché i CD-R (CD-Registrabili) e CD-RW (CD-ReWritable, CD -Riscrivibili) non possono avere fossette fisiche stampate in fabbrica, utilizzano diverse strategie ottiche per simulare i pit:

- **CD-R** (Dye): Utilizzano uno strato di tinta (cianina o ftalocianina) tra il policarbonato e lo strato riflettente. In fase di scrittura, un laser ad alta potenza (8-16 mW) scalda la tinta creando una macchia scura. In fase di lettura, il fotorivelatore vede la differenza tra le macchie scure e le aree trasparenti, interpretandole come pit e land.

- **CD-RW** (Phase Change): Lo strato di registrazione è una lega metallica con due stati stabili: cristallino (alta riflettività) e amorfo (bassa riflettività).
  - Un laser ad alta potenza fonde la lega rendendola amorfa (pit).
  - Un laser a media potenza la fonde permettendole di ricristallizzare (land).
  - Il laser a bassa potenza serve solo per la lettura senza alterare lo stato fisico.

I CD-RW vergini sono molto più costosi dei CD-R vergini e per questo non li hanno sostituiti completamente. Inoltre il fatto che, una volta scritto, un CD-R non possa essere cancellato accidentalmente, rappresenta un gran vantaggio per il backup dei dischi.

### 2.3.9 DVD e Blu-ray

L'aumento massiccio della capacità di archiviazione (dai 650 MB del CD ai 25 GB del Blu-ray) è dovuto principalmente alla riduzione delle dimensioni fisiche e alla precisione del laser:

- **DVD (Digital Versatile Disc):** Utilizza un laser rosso con una lunghezza d'onda più corta (0,65 micron contro 0,78 dei CD), che permette di leggere pit più piccoli (0,4 micron) e tracce più strette (0,74 micron).
- **Tecnologia Dual-Layer:** Alcuni DVD includono due strati di registrazione su un lato: uno riflettente e uno semiriflettente. Il laser può mettere a fuoco l'uno o l'altro strato semplicemente cambiando la sua lunghezza focale. Il livello più basso richiede che i pit e i land siano leggermente più grandi per essere leggibili in modo affidabile, e per questo motivo la sua capacità è leggermente inferiore rispetto a quella dello strato superiore.
  Ci sono quattro formati di DVD:
  - singolo lato, singolo strato (4,7 GB);
  - singolo lato, doppio strato (8,5 GB);
  - doppio lato, singolo strato (9,4 GB);
  - doppio lato, doppio strato (17 GB).
- **Blu-ray:** Utilizza un laser blu-violetto con una lunghezza d'onda ancora inferiore, consentendo una messa a fuoco estrema e supportando pit e land drasticamente più piccoli, portando la capacità a 25 GB per strato.

## 2.4 Input/Output

Un calcolatore è composto da tre componenti principali: la CPU, le memorie e i dispositivi di I/O.

### 2.4.1 Bus

Il sistema di I/O è strutturato attorno ai bus, cammini elettrici comuni che collegano CPU, memoria e dispositivi.

- **Struttura Logica:** Ogni dispositivo è composto da un **controller** (l'elettronica che gestisce il dispositivo) e dal dispositivo fisico stesso. Il controller comunica con la CPU tramite registri interni per ricevere comandi o fornire stato. In particolare il suo scopo è governare il proprio dispositivo di I/O e gestire il suo accesso al bus.
- **Meccanismi di Trasferimento:** il controller legge o scrive dati direttamente nella memoria principale senza l'intervento costante della CPU grazie al **DMA** (Direct Memory Access).
  Al termine di un'operazione, il controller genera un segnale di **interrupt** che forza la CPU a sospendere il programma corrente per eseguire una procedura specifica chiamata **interrupt handler** che controlla se ci sono errori, compie le azioni eventualmente necessarie e informa il sistema operativo che l'I/O è terminato.
- Arbitraggio: Poiché più componenti possono richiedere il bus simultaneamente, un chip chiamato **arbitro del bus** decide a chi concederne l'uso, dando solitamente la precedenza ai dispositivi di I/O rispetto alla CPU per evitare perdite di dati (processo noto come **cycle stealing** che rallenta le prestazioni) del calcolatore

- **Evoluzione:** Dai bus paralleli come ISA (Industry Standard Architecture, lento) e PCI (Peripheral Component Interconnect, più veloce, 133 MB/sec), si è passati al PCI Express (PCIe),,. Il PCIe non è tecnicamente un bus ma una rete point-to-point a commutazione di pacchetti, che usa connessioni seriali veloci chiamate lane (fino a 16 GB/sec per schede grafiche x16).

### 2.4.2 Terminali

I terminali sono le principali interfacce utente:

- **Tastiere:** Quando un tasto viene premuto o rilasciato, viene generato un interrupt. Il software (gestore degli interrupt) legge il numero del tasto e gestisce in modo logico le sequenze combinate come SHIFT o CTRL.
- **Touch Screen:** Esistono tre tecnologie principali: infrarossi (rilevano l'interruzione di fasci di luce), resistivi (pressione tra due strati flessibili) e capacitivi. Gli schermi multitouch moderni usano la capacità reciproca per rilevare più dita contemporaneamente, evitando il problema del ghosting (false coordinate) tipico delle altre tecnologie.
- **Display LCD:** Basati sui cristalli liquidi (molecole organiche viscose che si muovono come un liquido, ma che hanno anche una struttura spaziale simile a quella di un cristallo) che cambiano orientamento con campi elettrici, controllando il passaggio della luce polarizzata. Esistono a matrice passiva o attiva (TFT); quest'ultima usa transistor in ogni pixel per immagini più nitide.
- **Ram della scheda video:** Una memoria dedicata memorizza l'immagine come bitmap. Su uno schermo di 1600 x 1200 pixel (picture element, "elemento d'immagine") la RAM della scheda video deve contenere 1600 x 1200 valori; se è RGB ogni pixel deve memorizzare un valore RGB (3 byte), quindi uno schermo 1600 x 1200 pixel e 3 byte per pixel richiede circa 5.5 MB di memoria Ram per memorizzare l'immagine. Per video a 25 fps circa 155MB/s. Per risparmiare spazio, si può usare il colore indicizzato, dove ogni pixel punta a una tavolozza (palette) di 256 colori a 24 bit.

### 2.4.3 Mouse

Il mouse permette di puntare elementi sullo schermo. Esistono tre tipi di mouse: meccanici, ottici e opto-meccanici.

- **Meccanici:** composto da due rotelle di gomma sporgenti con degli assi disposti perpendicolarmente, in tal modo quando il mouse veniva spostato parallelamente a uno degli assi principali, ruotava soltanto una rotella.
- **Ottici:** che ha sul fondo un LED (Light Emitting Diode, "diodo luminescente") e un fotorilevatore. Viene utilizzato su uno speciale tappetino sul quale c'è una griglia di linee molto vicine fra loro; quando il mouse si muove il fotorilevatore riconosce l'incrocio tra due linee rilevando un cambiamento nella quantità di luce generata dal LED che viene riflessa.
- **opto-meccanico:** anch'esso dotato di una pallina che fa ruotare due cilindretti perpendicolari tra loro. Questi sono collegati a codificatori che hanno una serie di fori attraverso i quali può passare la luce; quando il mouse si sposta, i cilindretti ruotano e la luce colpisce il rilevatore ogni volta che un foro si trova allineato con il LED e il suo fotorilevatore. Il numero di impulsi che vengono rilevati è proporzionale allo spostamento effettuato.

In genere un mouse spedisce al calcolatore una sequenza di 3 byte, chiamata in alcuni casi mickey. Il primo byte contiene la distanza della x effettuata dall'ultimo movimento, mentre il secondo byte fornisce la stessa informazione per la y e il terzo byte contiene lo stato dei pulsanti del mouse.

### 2.4.4 Controller di Gioco

- **Wiimote:** Utilizza un accelerometro a 3 assi (masse che variano la capacità elettrica con l'accelerazione) per tracciare il movimento nello spazio. Per il puntamento fine, usa una telecamera interna che inquadra una barra sensore con LED posti sopra la TV.
- **Kinect:** Utilizza la "luce strutturata". Un laser emette una griglia di punti infrarossi e una telecamera ne cattura la distorsione per mappare la profondità della stanza e riconoscere i corpi umani,.

### 2.4.5 Stampanti

- **Stampanti laser:** All'inizio di ciascun ciclo di pagina il tamburo è caricato fino a circa 1000 volt e rivestito con un materiale fotosensibile. Successivamente la luce generata dal laser passa lungo tutta lunghezza del tamburo e per ottenere la deviazione orizzontale si utilizza uno specchio ottagonale rotante. Il fascio di luce viene modulato per produrre regioni luminose e regioni scure: le parti del tamburo colpite dal raggio perdono la loro carica elettrica.
  Il tamburo, dopo aver disegnato una linea di punti, ruota di una frazione di grado per permettere il disegno della linea successiva. A questo punto la prima linea di punti raggiunge il toner, un contenitore di polvere nera elettrostaticamente sensibile. II toner è attirato dai punti che hanno ancora una carica elettrica, formando così un'immagine visiva sulla linea del tamburo. Un istante dopo, ruotando, il tamburo ricoperto di toner viene premuto contro il foglio di carta, trasferendo su questo la polvere nera. Il foglio passa quindi attraverso dei rulli riscaldati per fissare l'immagine, fondendo in modo permanente il toner sulla carta. Nel seguito della rotazione il cilindro viene scaricato e ripulito di ogni residuo del toner, per poter essere nuovamente caricato e ricoperto di materiale fotosensibile, pronto per la stampa della pagina successiva.
  Una stampante laser può stampare in bianco e nero ma non può riprodurne le sfumature. Per riprodurre la scala dei grigi la tecnica più comune è quella dei mezzitoni. A seconda dei valori del grigio diverse celle vengono riempite di nero in maniera tale che l'occhio le percepisca come fosse del grigio.

- **Stampante a colori:** Le immagini a colori vengono viste in due modi distinti:

  – le immagini per luce trasmessa sono create mediante la sovrapposizione di tre colori primari additivi, il rosso, il verde e il blu.

  – le immagini per luce riflessa assorbono alcune lunghezze d'onda di luce e riflettono le restanti. Queste immagini sono create dalla sovrapposizione di tre colori primari sottrattivi, il ciano (assorbito dal rosso), il giallo (assorbito dal blu) e il magenta (assorbito dal verde).

  Le cinque tecnologie di stampa attualmente in uso si basano sul sistema CMYK (Cyan, Magenta, Yellow, blacK).

- **Stampanti speciali:**

  – **stampanti a colori a getto d'inchiostro:** funzionano come quelle monocromatiche ma con quattro cartucce (C, M, Y e K); a un basso prezzo forniscono risultati di buona qualità ma le cartucce non sono molto economiche.

  – **stampanti a inchiostro solido:** contengono quattro speciali inchiostri a cera solidificati in blocchi che vengono poi sciolti all'interno di serbatoi riscaldati.

  – **stampanti a getto di cera:** hanno un ampio nastro rivestito di quattro inchiostri a cera e suddiviso in settori lunghi quanto la larghezza della pagina. Al passaggio della carta migliaia di elementi riscaldanti sciolgono la cera, che si fonde con la carta.

  – **stampanti a sublimazione:** (passaggio dallo stato solido a quello gassoso senza passare per il liquido, esempi: ghiaccio secco e zolfo) un elemento mobile contenente i coloranti CMYK passa sopra una testina di stampa in cui vi sono migliaia di elementi riscaldanti programmabili; i coloranti vengono vaporizzati e assorbiti da una carta speciale collocata vicino alla testina. Maggiore è la temperatura, maggiore è l'intensità del colore.

  – **stampante termica:** contiene una piccola testina di stampa su cui vi sono un certo numero di piccoli aghi. Quando una corrente passa attraverso un ago, vengono disegnati su della carta speciale termosensibile dei punti in corrispondenza degli aghi caldi.

### 2.4.6 Apparecchiature per Telecomunicazioni

Il sistema di telecomunicazione permette la trasmissione di dati tra computer, spesso utilizzando l'infrastruttura telefonica o televisiva esistente.

- **Modem (Modulatore/Demodulatore):** Poiché le linee telefoniche analogiche sono progettate per la voce umana e non per segnali digitali grezzi (che usano 0V per lo '0' e 3-5V per '1'), i segnali subirebbero una distorsione eccessiva. Il modem risolve il problema utilizzando un'onda sinusoidale chiamata portante (tra 1000 e 2000 Hz) e modificandone le caratteristiche attraverso tre tecniche di modulazione:

- **Modulazione di Ampiezza (AM):** Vengono utilizzati due diversi livelli di tensione per rappresentare 0 e 1.
- **Modulazione di Frequenza (FM/FSK):** La tensione resta costante, ma la frequenza della portante cambia per distinguere i bit.
- **Modulazione di Fase (PM):** La fase della portante viene invertita di 180 gradi quando i dati cambiano. Sistemi complessi usano la codifica di fase dibit, che sposta la fase di angoli specifici (45, 135, 225, 315 gradi) per trasmettere 2 bit per intervallo.

È fondamentale distinguere tra Baud rate (numero di cambiamenti di segnale al secondo) e Bit rate (numero di bit trasmessi al secondo); il bit rate è solitamente un multiplo del baud rate. Le linee possono essere **full-duplex** (trasmissione simultanea in entrambe le direzioni), **half-duplex** (una direzione alla volta) o **simplex** (una sola direzione fissa).

- **ADSL (Asymmetric Digital Subscriber Line):** L'ADSL ottimizza il doppino telefonico (local loop) rimuovendo i filtri che limitavano la banda a 3000 Hz per la voce e sfruttando uno spettro fino a 1,1 MHz.
  - **Canalizzazione:** Lo spettro è diviso in 256 canali indipendenti da 4312,5 Hz ciascuno. Il canale 0 è per la telefonia tradizionale (POTS), i canali 1-5 sono vuoti per evitare interferenze, e i restanti 250 sono usati per dati e controllo.
  - **Asimmetria:** Poiché la maggior parte degli utenti scarica più dati di quanti ne carichi, viene assegnato l'80-90% della larghezza di banda al canale di download.
  - **Hardware:** Il segnale viene gestito da un NID (Network Interface Device) presso l'utente, uno splitter che separa voce e dati, e un DSLAM (Digital Subscriber Line Access Multiplexer) presso la centrale telefonica.
- **Internet via Cavo:** A differenza dell'ADSL, che è una linea privata, il cavo è un mezzo condiviso tra centinaia di utenti collegati a un headend.
  - **Meccanismo di accesso:** Utilizza minislot temporali per la trasmissione; per sincronizzarsi, il modem esegue il ranging, misurando il tempo di propagazione del segnale verso l'headend.
  - **Gestione conflitti:** Se più modem tentano di inviare richieste nello stesso minislot, si verifica una collisione; il modem attende allora un tempo casuale raddoppiato a ogni fallimento successivo.
  - **Sicurezza:** Essendo un mezzo condiviso, tutto il traffico viene crittografato in entrambe le direzioni per impedire intercettazioni tra vicini.

### 2.4.7 Codici di Caratteri

Ogni calcolatore ha un insieme di caratteri che, come minimo indispensabile, comprende le 26 lettere maiuscole, le 26 lettere minuscole, le cifre da 0 a 9 e un insieme di simboli speciali, come spazio, punto, virgola, segno meno e ritorno a capo.
Per poter utilizzare questi caratteri nel calcolatore occorre assegnare loro un numero: per esempio a = 1, b = 2, ..., z = 26, + = 27, - = 28. La corrispondenza tra caratteri e numeri naturali costituisce un codice di caratteri. È necessario che due calcolatori che comunicano tra loro utilizzino lo stesso codice, altrimenti non saranno in grado di capirsi. Per questa ragione sono stati definiti alcuni standard.

- **ASCII (American Standard Code for Information Interchange):** È un codice a 7 bit che permette 128 combinazioni totali.
  I codici compresi tra 0 e 1F (in esadecimale) sono **caratteri di controllo** e non vengono stampati. Molti dei caratteri di controllo ASCII sono pensati per la trasmissione di dati. Un messaggio potrebbe per esempio consistere di un carattere SOH (Start of Header, "inizio intestazione"), un'intestazione, un carattere STX (Start of Text, "inizio testo"), il testo stesso, un ETX (End of Text, "fine testo") e infine un carattere EOT (End of Transmission, "fine trasmissione"). I **caratteri ASCII stampabili** comprendono lettere maiuscole e minuscole, cifre, simboli di punteggiatura e alcuni simboli matematici.

| Hex | Name | Meaning | Hex | Name | Meaning |
|---|---|---|---|---|---|
| 0 | NUL | Null | 10 | DLE | Data Link Escape |
| 1 | SOH | Start Of Heading | 11 | DC1 | Device Control 1 |
| 2 | STX | Start Of TeXt | 12 | DC2 | Device Control 2 |
| 3 | ETX | End Of TeXt | 13 | DC3 | Device Control 3 |
| 4 | EOT | End Of Transmission | 14 | DC4 | Device Control 4 |
| 5 | ENQ | Enquiry | 15 | NAK | Negative AcKnowledgement |
| 6 | ACK | ACKnowledgement | 16 | SYN | SYNchronous idle |
| 7 | BEL | BELl | 17 | ETB | End of Transmission Block |
| 8 | BS | BackSpace | 18 | CAN | CANcel |
| 9 | HT | Horizontal Tab | 19 | EM | End of Medium |
| A | LF | Line Feed | 1A | SUB | SUBstitute |
| B | VT | Vertical Tab | 1B | ESC | ESCape |
| C | FF | Form Feed | 1C | FS | File Separator |
| D | CR | Carriage Return | 1D | GS | Group Separator |
| E | SO | Shift Out | 1E | RS | Record Separator |
| F | SI | Shift In | 1F | US | Unit Separator |

| Hex | Char | Hex | Char | Hex | Char | Hex | Char | Hex | Char | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 20 | (Space) | 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 12: ASCII character set

- **UNICODE:** Progettato per superare i limiti dell’ASCII nelle lingue non inglesi, assegna a ogni simbolo un valore univoco a 16 bit chiamato **code point**.
  - **Organizzazione:** Lo spazio è diviso in blocchi dedicati (es. Latino: 336 posizioni, Greco: 144, Cirillico: 256). Include oltre 20.000 ideogrammi Han (cinesi/giapponesi) e 11.000 sillabe Hangul coreane.
  - **Espansione:** Poiché 65.536 posizioni non bastavano, nel 1996 sono stati aggiunti altri 16 piani, portando il totale a oltre un milione di caratteri.
- **UTF-8 (UCS Transformation Format):** Alla fine anche UNICODE ha esaurito i code point e inoltre utilizza 16 bit per carattere per rappresentare testo ASCII puro, il che è uno spreco. Per rispondere a questi problemi, è stato introdotto il sistema UTF-8 UCS Transformation Format. I codici UTF-8 sono di lunghezza variabile, da 1 a 4 byte, e possono codificare circa due miliardi di caratteri. Uno dei vantaggi di UTF-8 è che i codici da 0 a 127 corrispondono ai caratteri ASCII, che possono essere espressi in 1 byte. Sono utilizzati in tutto sei formati differenti.

| Bits | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 |
|---|---|---|---|---|---|---|
| 7 | 0ddddddd | | | | | |
| 11 | 110ddddd | 10dddddd | | | | |
| 16 | 1110dddd | 10dddddd | 10dddddd | | | |
| 21 | 11110ddd | 10dddddd | 10dddddd | 10dddddd | | |
| 26 | 111110dd | 10dddddd | 10dddddd | 10dddddd | 10dddddd | |
| 31 | 1111110x | 10dddddd | 10dddddd | 10dddddd | 10dddddd | 10dddddd |

Figura 13: UTF-8 ecoding scheme

I bit contrassegnati con “d” sono bit di dati. Un altro vantaggio di UTF-8 è il fatto che il primo byte di ogni carattere UTF-8 determina univocamente il numero di byte nel carattere. Inoltre, i byte successivi al primo in un carattere UTF-8 iniziano sempre con 10, cosa mai vera per il byte iniziale, rendendo il codice auto sincronizzante. Cioè, in caso di errore di comunicazione o di memoria, è sempre possibile andare avanti e trovare l’inizio del carattere successivo.

# 3 Livello Logico Digitale

## 3.1 Porte Logiche e algebra di Boole

I circuiti digitali possono essere costruiti combinando tra loro un piccolo numero di componenti elementari.

### 3.1.1 Porte logiche

I circuiti digitali operano esclusivamente con due valori logici, solitamente rappresentati da intervalli di tensione specifici (ad esempio, 0-0,5 volt per lo 0 binario e 1-1,5 volt per l'1 binario). Il componente elettronico di base che calcola funzioni su questi segnali è la **porta logica**, costruita tramite transistor che agiscono come interruttori binari velocissimi.

- **Composizione HW:**Tutta la moderna logica digitale si fonda sul fatto che un transistor può essere costruito in modo da funzionare come un velocissimo interruttore binario. I transistor hanno tre connessioni: il **collettore**, la **base** e l'**emettitore**. La **resistenza** è necessaria per limitare la quantità di corrente nel transistor ed evitare che esso si fonda.

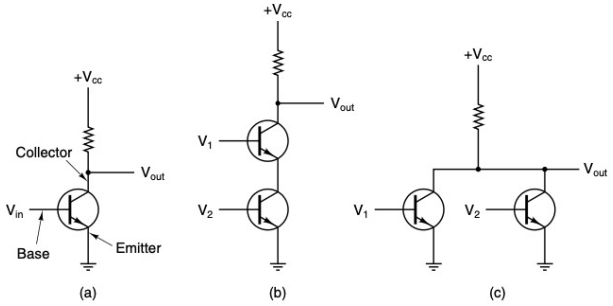



Figura 14: (a) transistor inverter (b) NAND gate (c) NOR gate

- **Funzionamento:** Quando la tensione in ingresso scende sotto un valore critico, chiamato $V_in$, il transistor viene disabilitato e si comporta come una resistenza infinita. La conseguenza è che l'output del circuito, $V_out$ assume un valore vicino a $V_cc$ (solitamente +1,5 volt): una tensione regolata esternamente che, per questo tipo di transistor, vale generalmente +5 volt. Quando, al contrario, $V_in$, supera il valore critico, il transistor si attiva e si comporta come un conduttore ideale, facendo scaricare $V_out$ a terra (per convenzione, 0 volt).

- **Collegamento in serie:** se V1 e V2 sono alte, allora entrambi i transistor saranno in conduzione e $V_out$ sarà portato al valore basso, mentre, se almeno uno degli ingressi è basso, allora i transistor corrispondenti saranno disattivati e l'uscita sarà alta. In altre parole $V_out$ sarà basso se e soltanto se sia V1 sia V2 sono alte.

- **Collegamento in parallelo:** se uno dei due ingressi è alto, allora il transistor corrispondente sarà attivato e l'uscita sarà scaricata a terra, mentre, se entrambi gli ingressi sono bassi, l'uscita rimarrà alta.

Questi tre circuiti formano le tre porte logiche più semplici, chiamate rispettivamente NOT, NAND e NOR.

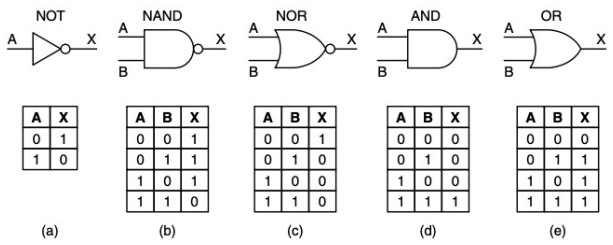



Figura 15: I simboli e il funzionamento delle cinque porte logiche base

- **Invertitore (NOT)**: Un singolo transistor configurato come descritto sopra inverte il segnale (ingresso basso produce uscita alta e viceversa).
- **Porta NAND**: Si ottiene mettendo due transistor in serie; l'uscita al collettore sarà bassa (0) solo se entrambi ricevono una tensione alta alle rispettive basi.
- **Porta NOR**: Si ottiene mettendo due transistor in parallelo; l'uscita sarà bassa se almeno uno dei transistor riceve tensione alla base, poiché essa viene scaricata a terra.
- **Porta AND:** Se si fa passare il valore in uscita dalla porta NAND in una NOT si ottiene un nuovo circuito, dove l'uscita vale 1 se e solo se entrambi gli ingressi valgono 1; esso realizza la porta logica chiamata AND.
- **Porta OR:** Analogamente è possibile collegare la porta logica NOR a un invertitore, in modo da ottenere un circuito la cui uscita valga 1 se almeno uno dei due ingressi vale 1, mentre valga 0 se entrambi gli ingressi valgono 0; esso realizza la porta logica chiamata OR.

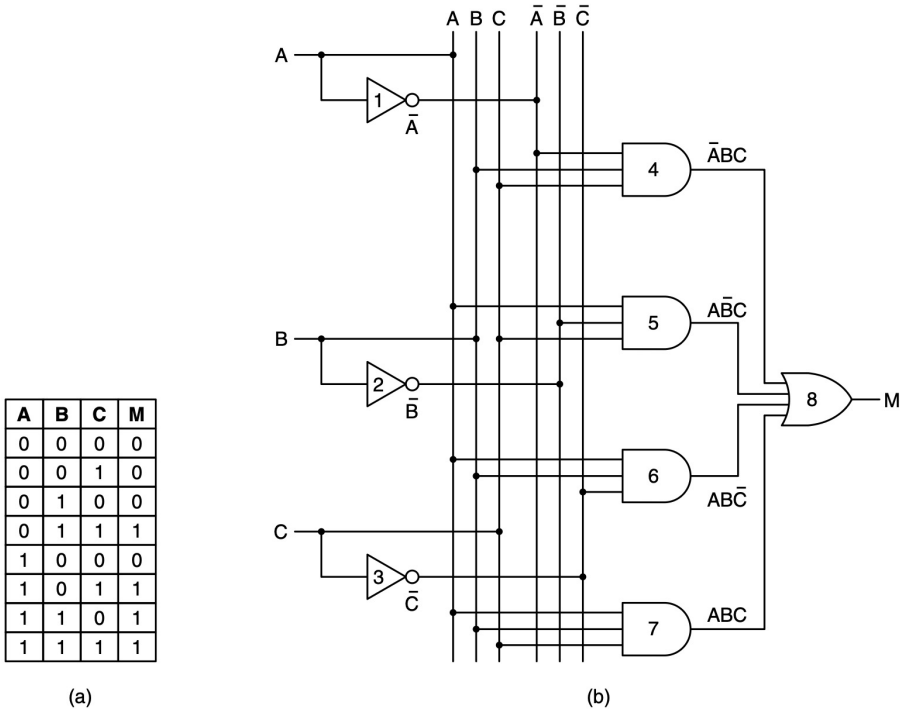
Quindi, è chiaro che le porte NAND e NOR necessitano di due transistor ciascuna, mentre le porte AND e OR ne richiedono tre; per questa ragione molti calcolatori sono basati sulle porte logiche NAND e NOR piuttosto che sulle più familiari porte AND e OR. Le porte logiche possono avere anche più di due ingressi, anche se in pratica difficilmente ce ne sono più di otto.
Esistono due principali tecnologie di costruzione delle porte logiche:

- **bipolare:** i tipi più importanti sono la TTL (Transistor-Transistor Logic), per anni utilizzata, e la ECL (Emitter-Coupled Logic), utilizzata quando è richiesto un funzionamento a velocità molto elevate;
- **MOS** (Metal Oxide Semiconductor, "semiconduttore metallo ossido"): è attualmente impiegata su larga scala nei circuiti dei calcolatori. Sono più lente delle TTL e delle ECL, ma richiedono molta meno potenza e hanno una dimensione decisamente inferiore, permettendo così di metterne insieme un numero elevato in uno spazio limitato.

### 3.1.2 Algebra di Boole

L'**algebra di Boole** (chiamata anche algebra di commutazione) è il formalismo matematico utilizzato per descrivere i circuiti digitali, in cui variabili e funzioni possono assumere solo i valori 0 e 1. Questo sistema è fondamentale per progettare l'hardware, poiché permette di trattare i segnali elettrici come entità logiche astratte.

- Una **funzione booleana** accetta una o più variabili in ingresso e produce un risultato che dipende unicamente dai valori di tali variabili.
- Poiché ogni variabile ha solo 2 stati possibili, una funzione con n variabili possiede esattamente $2^n$ combinazioni di input. Di conseguenza, la funzione può essere descritta in modo esaustivo da una **tabella di verità** con $2^n$ righe.
- Ad esempio, per due variabili esistono solo 16 possibili funzioni booleane, corrispondenti alle diverse stringhe di 4 bit ottenibili nella colonna dei risultati della tabella di verità. Al contrario, l'algebra ordinaria ha un numero infinito di funzioni di due variabili.




| A | B | C | M |
|---|---|---|---|
| 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 |

Figura 16: Tabella di verità e circuito associato

Per scrivere le funzioni booleane in modo compatto, si usano convenzioni simboliche:

- **NOT:** Una barra sopra la variabile ($\overline{A}$) indica l'inversione del valore.
- **AND:** Viene indicato da un punto o dalla semplice giustapposizione delle variabili (A·B o AB). Un termine come ABC assume valore 1 solo se tutti gli ingressi sono 1.
- **OR:** Viene indicato dal segno più (+)

Qualsiasi funzione booleana può essere specificata indicando quali combinazioni di variabili producono un output pari a 1. Questo porta alla formulazione della funzione come una somma di termini di prodotto (ognuno lungo n variabili). Ad esempio, la funzione di maggioranza per tre variabili può essere scritta come: $M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$. Questa formulazione è cruciale perché permette di passare direttamente dall'equazione matematica alla progettazione di un circuito fisico utilizzando porte logiche standard.

### 3.1.3 Implementazione delle funzioni booleane

Metodo generale per implementare un circuito per qualsiasi funzione booleana:

1. Scrivere la tabella di verità della funzione;
2. Fornire invertitori per generare il complemento di ogni ingresso;
3. Disegnare una porta AND per ogni termine con un 1 nella colonna del risultato;
4. Collegare le porte AND agli ingressi appropriati;
5. Convogliare l'uscita di tutte le porte AND in una porta OR.

Spesso è più vantaggioso implementare un circuito utilizzando un solo tipo di porta; di seguito una figura 17 che mostra come implementare una porta AND, OR, NOT usando solo NAND o NOR.

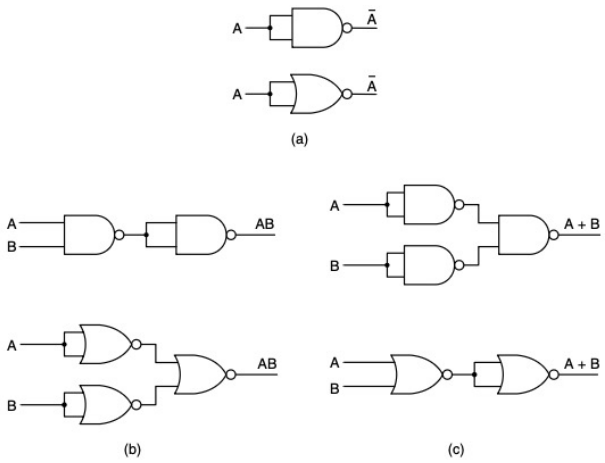



Figura 17: Implementazione delle porte logiche NOT (a), AND (b) e OR (c) usando soltanto porte NAND o solo porte NOR

Sebbene questa procedura non porti a circuiti ottimali, nel senso del numero minimo di porte, essa mostra che una soluzione è sempre fattibile. Sia le porte NAND che le porte NOR sono definite **complete**, perché qualsiasi funzione booleana può essere calcolata utilizzando esclusivamente una di esse.

### 3.1.4 Equivalenza di circuiti

I progettisti cercano di ridurre il numero di porte per tre ragioni principali:

- **Ridurre l'area del chip:** Meno porte occupano meno spazio;
- **Minimizzare il consumo energetico:** Ogni porta consuma elettricità;
- **Aumentare la velocità:** I segnali attraversano il circuito più rapidamente se incontrano meno ostacoli.

Due circuiti si dicono equivalenti se, per ogni possibile combinazione di ingressi, producono lo stesso identico output. Ad esempio, l'espressione $AB + AC$ può essere fattorizzata in $A(B + C)$ applicando la legge distributiva. Sebbene le funzioni siano logicamente identiche, il secondo circuito è preferibile perché richiede meno porte per essere realizzato.
Di solito un progettista di circuiti parte da una formula e in seguito, applicando le leggi dell'algebra di Boole, cerca di semplificarla:

| 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 | $\overline{AB} = \bar{A} + \bar{B}$ | $\overline{A + B} = \bar{A}\bar{B}$ |

Figura 18: Identità dell'algebra booleana

Grazie alle leggi di De Morgan, è possibile rappresentare le stesse porte in modi diversi graficamente; una porta NAND è equivalente ad una porta OR con gli ingressi invertiti. Queste inversioni sono chiamate **bolle di inversione**, che possono essere spostate lungo una linea per facilitare la conversione dei ciruiti in sole porte NAND o NOR.

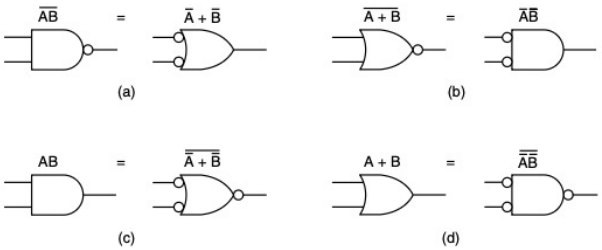



Figura 19: Simboli equivalenti per alcune porte logiche: (a) NAND, (b) NOR, (c) AND, (d) OR

## 3.2 Circuiti logici digitali elementari

Il seguente paragrafo descrive i circuiti logici digitali di base, spiegando come le porte logiche vengano raggruppate in moduli complessi che fungono da blocchi costruttivi per l'intero computer.

### 3.2.1 Circuiti integrati (IC)

Le porte logiche non sono vedute singolarmente, ma fabbricante su pezzi rettangolari di silicio chiamati **circuiti integrati (IC) o chip**. Si tratta di quadrati 5x5 mm circa. Ciascun pin è collegato a un input o a un output di una porta logica oppure all'alimentazione o alla terra.

- **Packaging:** I piccoli die (pezzi di silicio) sono montati in contenitori di plastica o ceramica con pin metallici per il collegamento esterno. I formati comuni includono i DIP (Dual Inline Packages) a due file di pin, i PGA (Pin Grid Array) con pubn sul fondo, e gli LGA (Land Grid Array) che usano piazzole piatte premute contro i connettori del socket.
- **Installazione:** Per garantire l'orientamento corretto, i DIP hanno una tacca, i PGA hanno un pin mancante (chiave), e gli LGA hanno intagli laterali che impediscono l'inserimento errato nel socket.
- **Limiti fisici:** Le porte reali presentano un ritardo di porta (**gate delay**), ovvero il tempo necessario per la propagazione del segnale e la commutazione (da picosecondi a pochi nanosecondi). Sebbene si possano integrare oltre un miliardo di transistor, il limite principale è il **numero di pin**; per questo si progettano circuiti con un alto rapporto porte/pin.

### 3.2.2 Circuiti combinatori

I circuiti combinatori sono quei circuiti logici con ingressi e uscite multiple in cui i valori di uscita sono determinati univocamente dai valori correnti degli ingressi. A differenza dei circuiti sequenziali, questi non contengono elementi di memoria e non dipendono da stati precedenti. Di seguito le principali tipologie di circuiti combinatori:

- **Multiplexer (Selettori):** un multiplexer è un circuito che possiede $2^n$ ingressi dati, un'unica porta di uscita e n ingressi di controllo.
  - **Funzionamento:** Gli ingressi di controllo codificano un numero binario che seleziona quale delle linee di ingresso deve essere "instradata" verso l'uscita. Ad esempio, in un multiplexer a otto ingressi, tre linee di controllo decidono quale porta AND abilitare per lasciar passare il segnale.
  - **Applicazioni pratiche:** è possibile realizzare qualsiasi funzione booleana a n variabili collegando gli ingressi del multiplexer a $V_{cc}$ (1 logico) o a terra (0 logico) in base ai valori della tabella di verità. Inoltre può essere usato per convertire dati da un formato parallelo a uno seriale, come avviene nelle tastiere dove la pressione di un tasto genera un codice che deve essere trasmesso bit dopo bit su un collegamento seriale.
  - **Demultiplexer:** è l'esatto opposto: instrada un unico segnale di ingresso verso una delle $2^n$ uscite in base al valore delle linee di controllo.

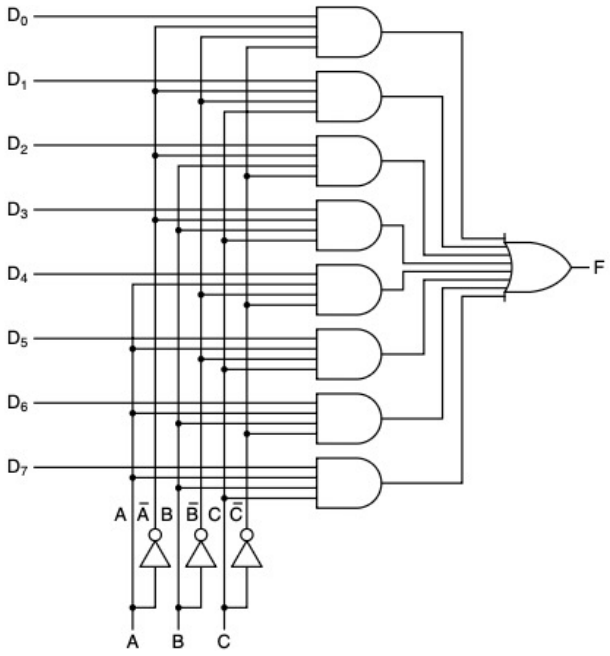



Figura 20: Un multiplexer a otto input

- **Decoder (Decodificatore):** Un decodificatore è un circuito che riceve in ingresso un numero di n bit e attiva (imposta a 1) esattamente una delle sue $2^n$ linee di controllo.
  - **Struttura:** ogni porta AND del decoder è abilitata da una combinazione unica di ingressi (diretti o invertiti).
  - **Esempio di utilizzo:** è fondamentale nella selezione dei chip di memoria. Se un sistema ha otto chip di memoria, i 3 bit di ordine superiore dell'indirizzo possono essere inviati a un decoder 3-8 per abilitare solo il chip corrispondente a quella specifica porzione di indirizzamento.

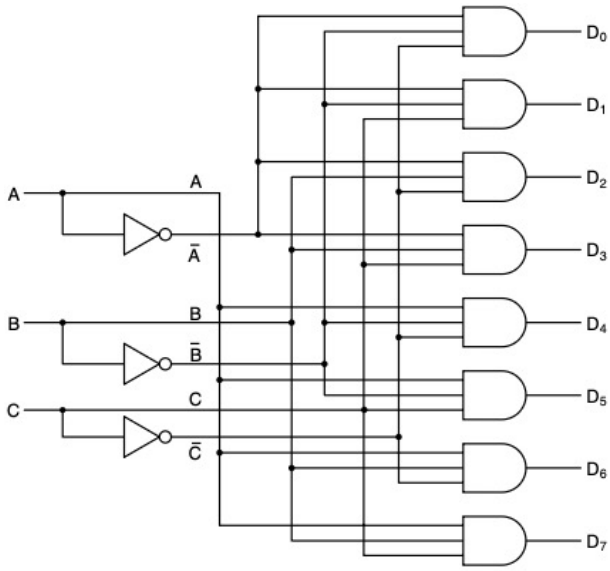



Figura 21: Un decodificatore da 3 a 8

- **Comparatori:** il comparatore è un circuito utilizzato per confrontare due parole (sequenze di bit).
  - **Meccanismo logico:** si basa sull'utilizzo di porte XOR (OR esclusivo). Poiché una porta XOR emette 0 se i due ingressi sono uguali e 1 se sono diversi, il circuito confronta i bit delle due parole a coppie.

– **Risultato finale:** i segnali in uscita dalle porte XOR vengono inviati a una porta NOR finale. Se tutti i bit corrispondono (tutti gli XOR danno 0), la porta NOR emette 1, indicano che le due parole sono uguali; altrimenti emette 0.

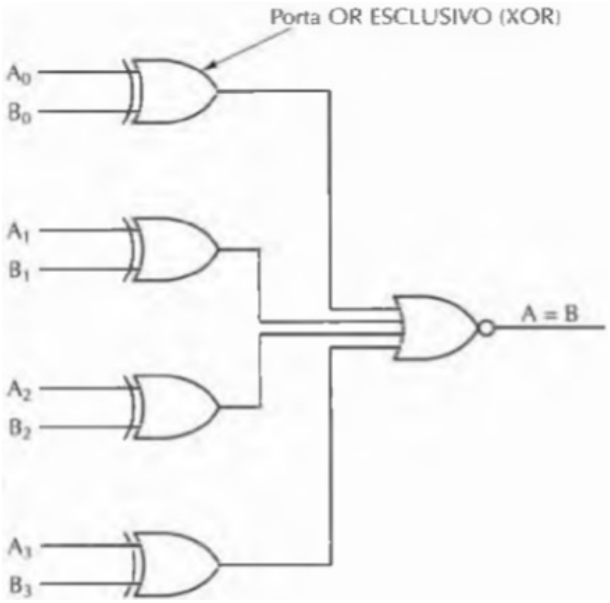



Figura 22: Semplice comparatore a 4 bit

- **Array Logici Programmabili:** un chip molto generale che permette di calcolare somme di prodotti; un esempio è il seguente 23 che vede 12 ingressi che vengono invertiti per un totale di 24 segnali di input. Il cuore del circuito è costituito da una schiera di 50 porte AND che possono avere come input un qualsiasi sottoinsieme dei 24 segnali di input.

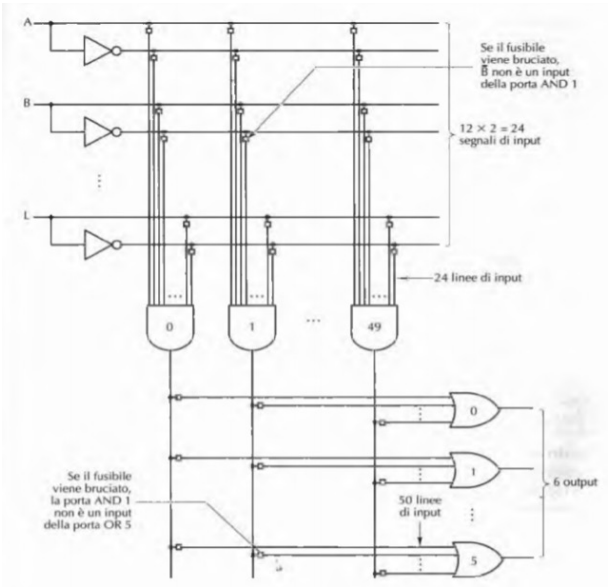



Figura 23: Array logico programmabile a 12 input e 6 output. I quadratini rappresentano i fusibili che possono essere bruciati per determinare la funzione da calcolare. I fusibili sono disposti in due matrici: quella superiore per le porte AND, quella inferiore per le porte OR

### 3.2.3 Circuiti per l'aritmetica

I circuiti combinatori specificamente progettati per eseguire operazioni matematiche e manipolazioni di bit all'interno del processore.

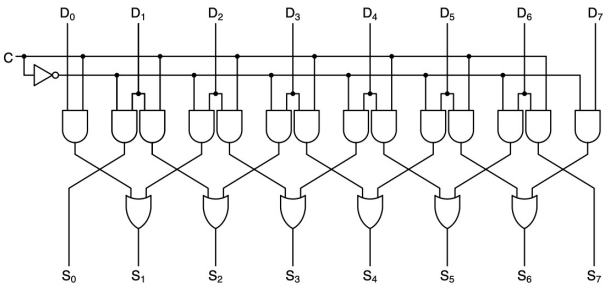



Figura 24: Uno shifter a sinistra di 1 bit

- **Shifters (Registri a scorrimento):** il circuito shifter permette di spostare i bit di una parola verso destra o verso sinistra.
  - **Struttura:** In un modello a 8 bit, il circuito accetta otto linee di input $(D_0, \ldots, D_7)$ e produce otto linee di output $(S_0, \ldots, S_7)$.
  - **Funzionamento:** Una linea di controllo (C) determina la direzione: se C=0 avviene uno spostamento a sinistra, se C=1 avviene uno spostamento a destra. Internamente, il circuito utilizza coppie di porte AND per abilitare il passaggio del bit verso la porta OR della posizione successiva o precedente.

- **Adder (sommatore):**La capacità di sommare interi è fondamentale per ogni CPU. Ci sono più livelli di complessità:
  - **Half Adder (Semi-sommatore):** è il circuito più semplice; somma due bit e produce un output: la Somma e il Riporto (Carry). È limitato perché non può gestire un riporto proveniente da una posizione precedente, quindi è utile solo per i bit meno significativi.

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

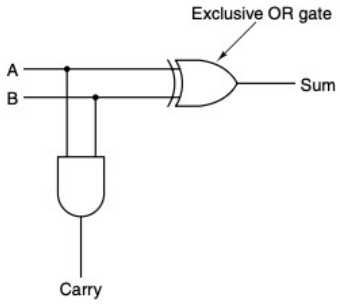



Figura 25: (a)Tabella delle verità della somma di 1 bit (b) Circuito di un semi-sommatore

  - **Full Adder (Sommatore):** risolve il imite del semi-sommatore accettando tre ingressi: i due bit da sommare e il riporto in ingresso (Carry in). È costruito unendo due semi-sommatori e una porta OR.

| 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 |

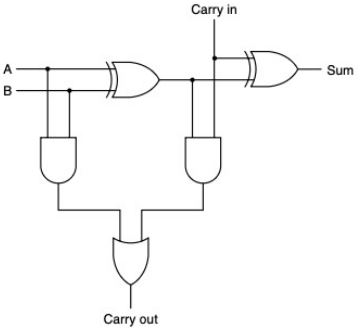



Figura 26: (a) tabella di verità di un sommatore (b) Circuito di un sommatore

- **Ripple Carry Adder (Sommatore a propagazione di riporto):** si ottiene collegando in cascata più sommatori (Full Adder). Il riporto si propaga da destra verso sinistra. Il difetto principale è il ritardo: l'operazione non è completa finché il riporto non ha attraversato l'intero circuito, il che rallenta il sistema.
- **Carry Select Adder (Sommatore a selezione del riporto):** è un'ottimizzazione per aumentare la velocità. Un sommatore a 32 bit viene diviso in due metà da 16 bit; la metà superiore viene duplicata in hardware. Una versione calcola il risultato assumendo che il riporto in ingresso sia 0, l'altra assumendo che sia 1. Quando la metà inferiore finisce il calcolo, la CPU seleziona istantaneamente il risultato corretto tra i due già pronti, dimezzando i tempi di attesa.

- **Unità aritmetico logica (ALU):** L'ALU è l'unico circuito integrato capace di eseguire diverse funzioni logiche (AND, OR) e aritmetiche (somma) su due parole.
  - **Struttura di un ALU a 1 bit:** Una ALU per parole a n bit è tipicamente costruita collegando in parallelo n circuiti identici chiamati bit slices. Ogni singola cella a 1 bit contiene: un'untità logica (per calcolare funzioni AND, OR e NOT); un'unità aritmetica (composta da due full-adder che gestiscono gli addendi e i segnali di riporto); un decoder a 2 bit (situato nell'algolo inferiore sinistro del circuito, riceve i segnali di controllo (F0, F1) per selezionare quale delle quattro operazioni deve essere inviata all'uscita finale)
  - **Segnali di controllo e funzioni:** una ALU standard è pilotata da 6 linee di controllo principali che permettono di manipolare gli input e l'operazione finale. **F0, F1** determinano la funzione selezionata dal decoder ($AND, OR, \overline{B}, A + B$).
    **ENA, ENB (Enable A/B)** permettono di abilitare individualmente gli ingressi A e B. Se negati, forzano l'ingresso relativo a zero.
    **INVA (Invert A)** inverte l'ingresso sinitro (A)
    **INC (Increment)** forza un riporto nel bit meno significativo (Carry in), permettendo operazioni come A+1 o A+B+1.

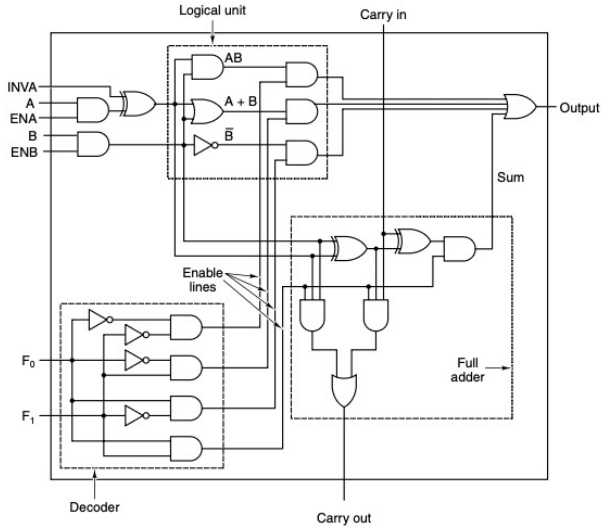



Figura 27: ALU a 1 bit

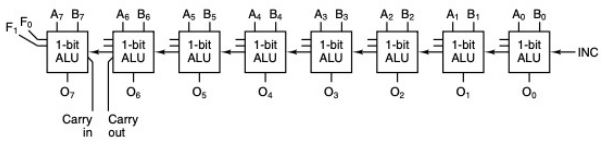



Figura 28: 8 ALU a 1 bit che formano un ALU a 1 bit. Per semplificare non sono mostrati segnali di abilitazione e segnali di inversione

### 3.2.4 Clock

I circuiti digitali devono gestire la sincronizzazione e l'ordine degli eventi temporali, un aspetto critico per il funzionamento di qualsiasi computer.

- **Definizione clock:** in questo conteso, un clock è un circuito che emette una serie di impulsi di larghezza definita e a intervalli temporali costanti.
- **Ciclo di clock:** l'intervallo temporale compreso tra le estremità di due impulsi consecutivi è detto ciclo di clock.
- **Frequenza e Precisione:** le frequenze tipiche variano oggi tra 100 MHz e 4 GHz (corrispondenti a cicli tra 10 nanosecondi e 250 picosecondi). Per garantire un'elevata precisione, la frequenza è solitamente controllata da un oscillatore a cristallo.

In un calcolatore possono verificarsi più eventi durante uno stesso ciclo di clock. Se però è necessario che si verifichino in uno specifico ordine occorre dividere il ciclo di clock in sottocicli. Una tecnica usata è la seguente:

- **Segnali sfasati (Phase-shifting):** è possibile derivare un segnale secondario dalla linea principale inserendo un circuito con un ritardo noto. Questo genera un secondo segnale (C2) sfasato rispetto al primario (C1).
- **Riferimenti temporali multipli:** utilizzando due segnali (C1 e C2), si ottengono quattro riferimenti temporali distinti in un unico ciclo: il fronte di salita di C1, il fronte di discesa di C1, il fronte di salita di C2 e il fronte di discesa di C2. Associando diversi eventi ai quattro fronti è possibile stabilire per loro una desiderata sequenza. Se, all'interno di ogni ciclo di clock, sono necessari più di quattro riferimenti temporali occorre collegare al clock principale altre linee secondarie e utilizzare circuiti con ritardi diversi.
- **Intervalli logici:** Se il circuito richiede intervalli di tempo invece di istanti discreti, si possono usare le combinazioni logiche degli stati alti dei clock (ad esempio, l'intervallo in cui C1 è alto AND C2 è alto) per distinguere fino a quattro sotto intervalli diversi.

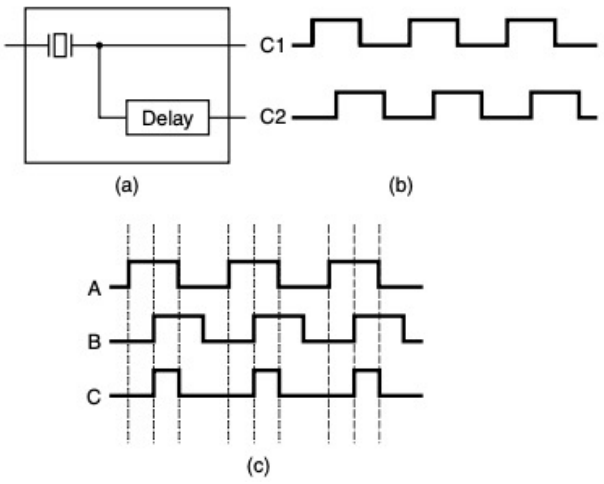



Figura 29: (a) Clock (b) Diagramma di temporizzazione del clock. (c) Generazione di un clock asimmetrico

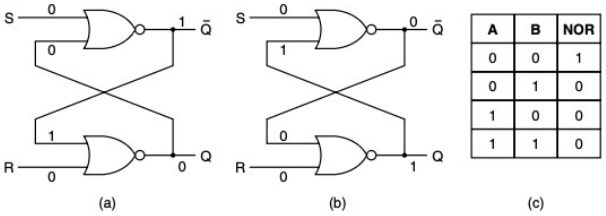
## 3.3 Memoria

Una componente essenziale di ogni calcolatore è la memoria; se non ci fosse non potrebbe esistere nessun calcolatore, almeno nella forma in cui lo conosciamo. La memoria è utilizzata per conservare sia le istruzioni da eseguire sia i dati.

### 3.3.1 Latch

Per creare una memoria da 1 bit, è necessario un circuito che possa ricordare i valori di ingresso passati. Questo si ottiene tramite un latch, la cui forma più semplice è l'SR latch.

- **Struttura SR Latch (Set-Reset):** è costituito da due porte NOR collegate in modo incrociato, dove l'uscita di una porta è l'ingresso dell'altra.
- **Ingressi e Uscite:** Possiede due ingressi, S (Set) per impostare lo stato e R (Reset) per azzerarlo, e due uscite complementari $Q$ e $\overline{Q}$
- **Stati stabili:** Quando sia S che R sono a 0, il latch ha due stati stabili (0 e 1) e mantiene indefinitamente il valore precedentemente memorizzato
- **Funzionamento:** portando momentaneamente S a 1, il latch passa allo stato $Q = 1$, indipendentemente dallo stato precedente. Portando momentaneamente R a 1, il latch viene forzato allo stato $Q = 0$.
- **Indeterminatezza:** se sia S che R vengono impostati a 1 contemporaneamente, il circuito diventa non deterministico nel momento in cui entrambi tornano a 0, poiché lo stato finale dipenderà da quale dei due segnali scende per ultimo.




| A | B | NOR |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Figura 30: (a) Latch di tipo NOR nello stato 0 (b) Latch di tipo NOR nello stato 1 (c) Tabella di verità del NOR

Per controllare con precisione il momento in cui il latch cambia stato, si aggiunge un terzo ingresso chiamato clock o (enable/strobe).

- **Funzionamento:** Quando il segnale di clock è 0 (strobe), le uscite delle porte AND interne sono 0 e il latch mantiene il suo stato attuale indipendente da S e R.
- **Abilitazione:** il latch diventa sensibile agli ingressi S e R solo quando il segnale di clock è 1 (enable).
- **Non determinismo:** il circuito diventa non deterministico finché sia R che S non tornino ad assumere il valore 0. L'unico stato consistente per $S = R = 1$ è quando $Q = \overline{Q} = 0$; non appena entrambi gli input tornano al valore 0 il latch deve tuttavia passare istantaneamente in uno dei suoi due stati stabili. Se uno dei due input torna a 0 prima dell'altro, prevale quello che rimane al valore 1 più a lungo, dato che quando uno solo degli input vale 1 esso determina lo stato del latch. Se invece entrambi gli input ritornano a 0 nello stesso istante il latch passa in uno dei due stati stabili in modo del tutto casuale.

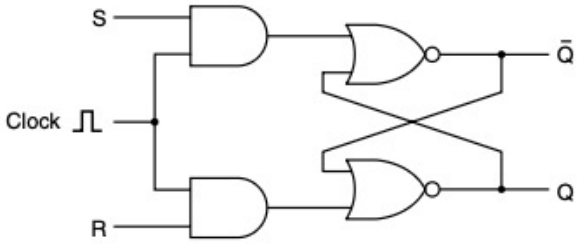



Figura 31: Latch SR temporizzato

Per risolvere l'instabilità causata dalla condizione $S = R = 1$ è con il Latch D temporizzato.

- **Struttura:** Utilizza un unico ingresso dati, D. Poiché l'ingresso alla seconda porta è l'inverso del primo, non è fisicamente possibile che entrambi siano contemporaneamente 1.
- **Capacità:** Quando il clock è a 1, il valore corrente di D viene campionato e memorizzato nel latch. Il valore memorizzato è sempre disponibile in uscita su Q. Quindi questo circuito è una **vera memoria a 1 bit**.
- **Efficienza:** Sebbene il circuito base richieda 11 transistor, i design moderni più sofisticati possono memorizzare 1 bit con soli sei transistor.

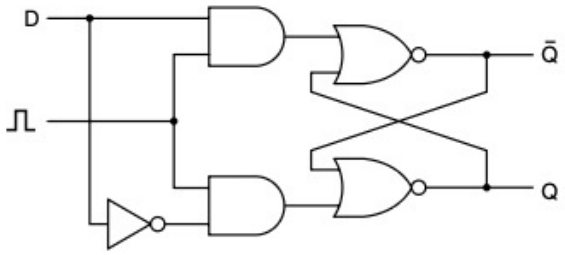



Figura 32: Latch D Temporizzato

**Analogia:** Un latch è come una porta con una serratura a scatto. L'ingresso Set è come spingere la porta per chiuderla: una volta chiusa, resta tale anche se smetti di spingere. L'ingresso Reset è come tirarla per aprirla. Il Clock agisce come una chiave: puoi spingere o tirare quanto vuoi, ma la porta cambierà posizione solo se la chiave è inserita e girata (clock a 1).

### 3.3.2 Flip-Flops

Un flip-flop è un dispositivo di memoria che campiona il valore presente su una linea in un istante particolare e lo memorizza. La differenza cruciale rispetto a un latch è la seguente:

- **Latch:** è level-triggered, ovvero sensibile al livello del segnale (lo stato cambia finché il clock è 1).
- **Flip-flop:** è edge-triggered, ovvero sensibile alla transizione (lo stato cambia solo durante il passaggio del segnale da uno stato all'altro).

Poiché il flip-flop reagisce solo ai cambiamenti, la durata dell'impulso di clock diventa irrilevante, purché le transizioni siano rapide. Esistono due varianti principali:

- **Fronte di salita:** La transizione avviene quando il segnale passa da 0 a 1;
- **Fronte di discesa:** La transizione avviene quando il segnale passa da 1 a 0.

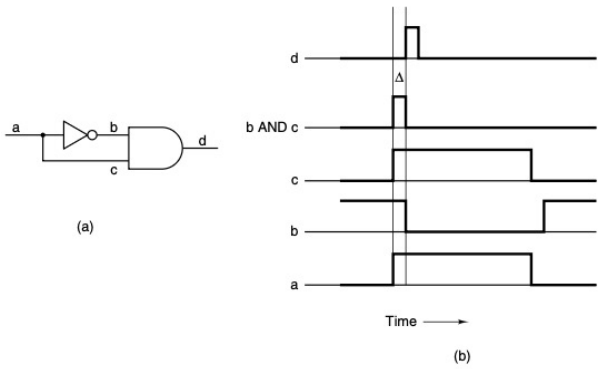



Figura 33: (a) Generatore d'impulsi. (b) Diagrammi temporali.

Un metodo comune per implementare un flip-flop consiste nel **generare un impulso** brevissimo in corrispondenza del fronte di salita del clock e inviarlo a un latch D. Questo impulso viene creato sfruttando il ritardo di propagazione ($\Delta$) dei circuiti:

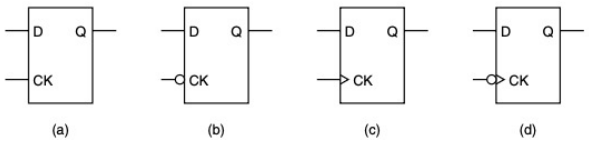



Figura 35: Latch D (a,b) e Flip-Flop (c,d)

- Il segnale del clock viene inviato a una porta AND attraverso due strade: una diretta e una che passa per un Inverter.
- L'inverter introduce un ritardo minimo (tipicamente pochi nanosecondi o picosecondi).
- Per un brevissimo istante dopo che il clock è salito, entrambi gli ingressi della porta AND risultano alti (il segnale originale è già a 1, mentre quello invertito non è ancora diventato 0), generando un impulso di durata pari a $\Delta$.

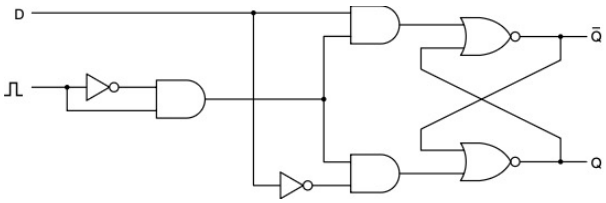



Figura 34: Flip-Flop D

Negli schemi logici, i flip-flop si distunguono dai latch per un simbolo a forma di triangolo posto sull'ingresso del clock (CK). Molti modelli includono anche ingressi supplementari per il controllo diretto:

- **Set o Preset:** Forza lo stato del flip-flop a Q=1.
- **Reset o Clear:** Forza lo stato a Q=0.

**Analogia:** Immagina di voler catturare l'immagine di un corridore. Un latch è come una finestra aperta: finché la finestra resta aperta (livello alto), puoi vedere il corridore cambiare posizione. Un flip-flop è come una fotocamera con flash: non importa quanto a lungo tieni il dito sul pulsante, l'immagine viene catturata esattamente nell'istante in cui scatta il flash (il fronte del clock), "congelando" la realtà in quel preciso momento.

### 3.3.3 Registri

I flip-flop possono essere combinati in gruppi per creare registri, progettare per contenere tipi di dati con una lunghezza superiore a 1 bit, come parole da 8, 32, 64 bit. Mentre un singolo flip-flop memorizza solo uno 0 o un 1, un registro permette di immagazzinare un intero valore numerico o una stringa di bit.
Il funzionamento coordinato del registro è garantito dal segnale di clock:

- **Caricamento simultaneo:** Per far si che tutti i bit del registro vengano caricati nello stesso istante, tutte le linee di clock dei singoli flip-flop che lo compongono sono collegate a un unico segnale di ingresso comune.
- **Triggering:** Nel modello descritto, il caricamento avviene durante la transizione positiva (fronte di segnale) del segnale del clock.

Oltre al clock, i registri dispongono spesso di un segnale di **Clear** (CLR):

- Tutte le linee di reset dei flip-flop sono collegate insieme (ganded)
- Quando il segnale CLR viene portato a 0, tutti i flip-flop vengono forzati simultaneamente allo stato 0, permettendo di azzerare istantaneamente l'intero contenuto del registro.

Un aspetto critico nella progettazione dei registri riguarda la potenza del segnale:

- Poiché un singolo segnale di input potrebbe non avere corrente sufficiente per pilotare contemporaneamente otto o più flip-flop, viene inserito un inverter sulla linea del clock in ingresso.
- Questo inverter funge da **amplificatore**, garantendo che ogni componente del registro riceva un segnale pulito e forte per eseguire l'operazione richiesta.

I registri sono **strutture modulari**: un registro a 8 bit può essere utilizzato come blocco base per costruire registri più grandi. Ad esempio, è possibile creare un registro a 32 bit combinando due moduli da 16 bit e collegando tra loro i rispettivi segnali di clock e di reset.

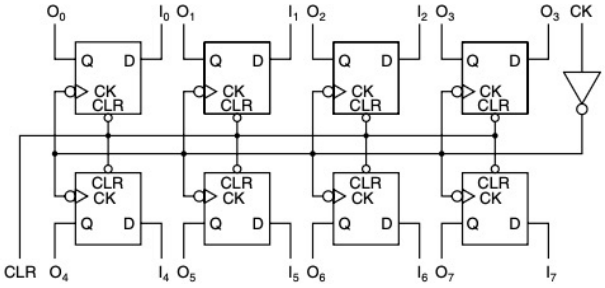



Figura 36: Registro a 8 bit costruito da singoli flip-flop

**Analogia:** Un registro è come una rastrelliera per biciclette con un unico lucchetto centrale. Ogni posto nella rastrelliera (il flip-flop) può contenere o meno una bici (un bit). Invece di dover chiudere ogni singolo lucchetto uno alla volta, il segnale di clock agisce come la leva centrale che blocca o sblocca tutti i posti contemporaneamente, assicurando che l'intera fila venga aggiornata nello stesso istante.

### 3.3.4 Organizzazione della memoria

Per costruire memorie di grandi dimensioni, non è sufficiente raggruppare bit in semplici registri, ma è necessaria una struttura organizzata in cui le singole parole (word) siano indirizzabili individualmente. Questa organizzazione deve seguire uno schema regolare per permettere al sistema di essere facilmente esteso a capacità superiori e per ridurre il numero di collegamenti fisici necessari. Uno dei vantaggi principali di questo approccio è l'**ottimizzazione del numero di pin** sul chip: mentre un registro da 8 bit richiede circa 20 segnali, una memoria da 12 bit organizzata in parole ne richiede solo 13, poiché i bit possono condividere le linee di uscita tramite una logica di selezione.

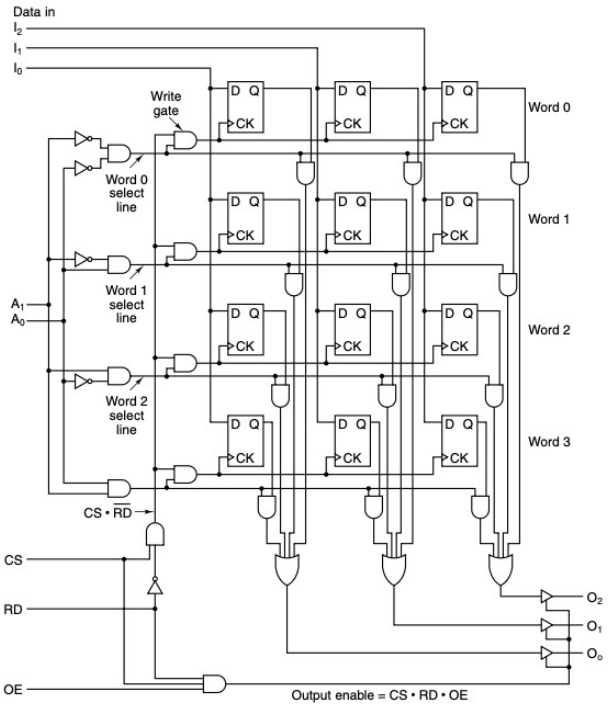



Figura 37: Diagramma logico di una memoria 4 x 3. Ogni riga è una delle quattro parole a 3 bit. Lettura e scrittura riguardano sempre parole complete.

La figura 37 illustra una memoria con quattro parole a 3 bit e anche se presenta una capacità totale (di 12 bit) decisamente maggiore rispetto a quella del flip-flop ottale visto precedentemente, richiede un numero inferiore di pin. Una caratteristica ancora più rilevante è che questa organizzazione è facilmente estendibile a memorie di dimensione maggiore.

- **Interfacce di segnale:** La memoria dispone di tre linee di input dati $(I_0, I_1, I_2)$, due linee di indirizzo $(A_0, A_1)$ e tre segnali di controllo: **CS (Chip Select)** per abilitare il chip, $RD(Read)$ per distinguere tra lettura e scrittura e **OE (Output Enable)** per gestire l'ingresso dei dati sul bus.
- **Processo di indirizzamento:** All'interno del chip, un decodificatore converte i bit di indirizzo per attivare un'unica linea di selezione della parola alla volta. Solo la riga di flip-flop corrispondente all'indirizzo binario fornito risulterà operativa per l'azione richiesta.
- **Operazione di Scrittura:** Quando CS è alto (valore 1) e PD è basso (valore 0), la linea di selezione abilita i gate che trasmettono il segnale di clock ai flip-flop della parola scelta, caricando i valori presenti sulle linee di input. Le altre parole non vengono modificate.
- **Operazione di Lettura:** Quando CS e RD sono alti (valore 1), la linea di selezione abilita i gate AND collegati alle uscite Q dei flip-flop della parola selezionata. Questi dati fluiscono attraverso porte OR comuni che convergono verso le linee di uscita del chip.
- **Buffer tri-state:** Per evitare conflitti elettrici quando le linee di ingresso e uscita condividono lo stesso bus, si utilizzano buffer non invertenti a tre stati (tri-state). Questi componenti agiscono come interruttori che possono emettere 0, 1 o uno stato di alta impedenza (circuito aperto). Il chip risulta elettricamente scollegato dal resto del sistema a meno che i segnali CS, RD e OE non siano tutti simultaneamente attivi.

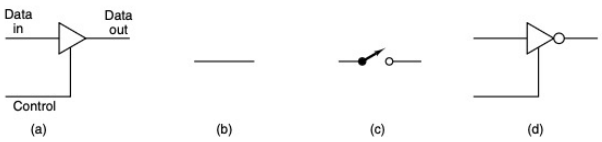



Figura 38: (a) Buffer non invertente (b) Risultato di (a) quando il controllo è alto (c) Risultato di (a) quando il controllo è basso (c) Buffer invertente

L'architettura descritta si dimostra estremamente **modulare e scalabile**: per aumentare il numero di bit per parola è sufficiente aggiungere colonne di flip-flop, mentre per aumentare il numero di parole si aggiungono nuove righe e si espande la logica del decodificatore aggiungendo linee di indirizzo. Un principio cardine di questa progettazione è che, per massimizzare l'efficienza della logica di indirizzamento, il numero di parole contenute in un modulo di memoria è sempre espresso come una potenza di 2.

### 3.3.5 Chip di memoria

Poiché la tecnologia dei circuiti integrati è ideale per creare schemi bidimensionali ripetitivi, i chip di memoria rappresentano un'applicazione perfetta per questa tecnologia. Con il miglioramento dei processi produttivi, il numero di bit che possono essere inseriti in un singolo chip aumenta costantemente, raddoppiando circa ogni 18 mesi seguendo la **Legge di Moore**. Tuttavia, i chip più grandi non rendono sempre obsoleti quelli piccoli a causa di diversi compromessi tra velocità, consumo energetico e prezzo.
Ad **esempio**, un chip di memoria da 4 Mbit ha due possibili organizzazioni:

- **Organizzazione 512Kx8:** in questa configurazione, il chip richiede 19 linee di indirizzo per selezionare uno dei $2^{19}$ byte e 8 linee di dati per caricare o leggere il byte selezionato.
- **Organizzazione 4096Kx1:** qui il chip è visto internamente come una matrice 2048x2048 di celle da 1 bit. Per ridurre il numero di pin necessari, viene utilizzato l'indirizzamento sequenziale: prima si seleziona una riga tramite l'impulso RAS (Row Address Strobe) e successivamente una colonna tramite CAS (Column Address Strobe).

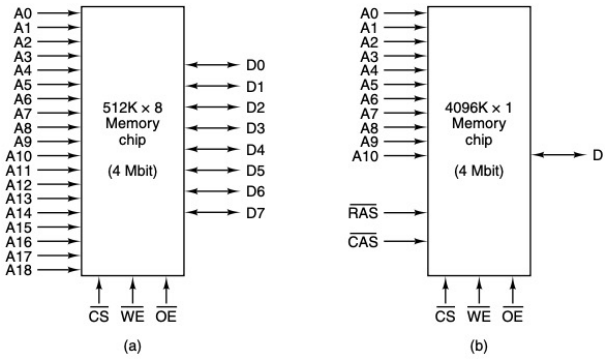



Figura 39: Due modi di organizzare un chip di memoria a 4 Mbit.

Indipendentemente dall'organizzazione, il chip utilizza segnali specifici:

- **CS (Chip Select):** Viene assertito per abilitare il chip specifico tra i molti presenti nel sistema.
- **WE (Write Enable):** Indica se i dati devono essere scritti (1) o letti (0).
- **OE (Output Enable):** Quando asserito, permette ai segnali di uscire dal chip; se negato, il chip viene scollegato elettricamente dal circuito (alta impedenza).

Un segnale è detto **asserito** (piuttosto che dire che assume valore alto o basso) per indicare che è impostato in modo da generare una qualche azione. Alcuni pin sono asseriti con valore alto, mentre altri con valore basso. I pin asseriti con valore basso sono identificati da una linea sopra il loro nome. L'opposto di asserito è negato. Se un pin è attivo basso (ovvero si attiva con tensione bassa), il suo nome è indicato con una barra sopra (es.$\overline{CS}$).
I chip di memoria di grandi dimensioni sono spesso costruiti come matrici di n x n indirizzate da numeri di riga e colonna. Questo tipo di architettura riduce il numero di pin necessari, ma rende allo stesso tempo più lento il chip, in quanto sono necessari due cicli di indirizzamento, uno per la riga e uno per la colonna. Per riguadagnare parte della velocità persa a causa dell'architettura, in alcuni chip è possibile specificare un indirizzo di riga seguito da una sequenza di indirizzi di colonna in modo da poter accedere a bit consecutivi all'interno di una stessa riga.
La scelta del design di un chip dipende da due fattori indipendenti: la **larghezza dell'output** (quanti bit vengono consegnati contemporaneamente, es. 1, 4, 8 o 16) e **la modalità di presentazione degli indirizzi** (tutti insieme o suddivisi in riga e colonna). Mentre i chip larghi 1 bit erano comuni in passato, l'aumento della dimensione delle parole di memoria (32 o 64 bit) ha reso più convenienti le famiglie di chip con larghezze di 4, 8 o 16 bit per evitare di dover collegare troppi chip in parallelo.

### 3.3.6 RAM e ROM

Le memorie che possono essere sia lette che scritte sono comunemente chiamate RAM (Random Access Memory), sebbene il termine sia tecnicamente improprio poiché quasi tutti i chip di memoria consentono l'accesso casuale. La distinzione principale avviene tra memorie volatili, che perdono i dati senza alimentazione, e memorie non volatili, che conservano le informazioni permanentemente.
Le memorie RAM si dividono in due categorie tecnologiche fondamentali:

- **SRAM (Static RAM):** Sono costruite internamente con circuiti simili ai flip-flop D e mantengono i dati finché c'è alimentazione. Sono estremamente veloci (tempi di accesso nell'ordine dei nanosecondi) e per questo vengono utilizzate principalmente per le memorie cache.
- **DRAM (Dynamic RAM):** Non usano flip-flop ma una matrice di celle composte da un solo transistor e un minuscolo condensatore. Poiché la carica elettrica tende a disperdersi, devono essere rinfrescate (refreshed) ogni pochi millisecondi. Nonostante siano più lente delle SRAM, la loro altissima densità le rende ideali per la memoria principale.
- **SDRAM (Syncrhonous RAM):** Il più datato è il DRAM FPM (Fast Page Mode), ancora utilizzato nei calcolatori più vecchi. La DRAM FPM è stata sostituita dalla DRAM EDO (Extended Data Output) in cui un riferimento alla memoria può avere inizio ancor prima che sia completato il precedente. Sono pilotate dal clock di sistema eliminando i segnali di controllo, mentre le SDRAM DDR (Double Data Rate) raddoppiano la velocità trasferendo dati sia sul fronte di salita che su quello di discesa del clock.

In molte applicazioni il programma e alcuni dati devono rimanere memorizzati anche a macchina spenta.

- **ROM (Read-Only Memory):** hanno dati inseriti in fabbrica e sono immutabili. Ciò avviene esponendo alla luce un materiale fotosensibile attraverso una maschera contenente il pattern di bit desiderato e incidendo la superficie esposta.
- **PROM (Programmable ROM):** possono essere programmate una sola volta sul campo bruciando dei minuscoli fusibili interni: ciò viene fatto selezionando righe e colonne e applicando un'alta tensione a un particolare pin del chip.
- **EPROM (Erasable PROM):** i cui campi non solo possono essere programmati, ma anche cancellati tramite esposizione a luce ultravioletta (in questo modo tutti i bit assumono il valore 1). Generalmente hanno la stessa organizzazione delle RAM statiche.
- **EEPROM (Electrically EPROM):** possono essere cancellate e riscritte elettricamente senza essere rimosse dal circuito (non possono essere più grandi di 1/64 delle comuni EPROM), pur essendo molto più lente e costose delle comuni DRAM.
- **Memoria Flash:** è una variante delle EEPROM che permette la cancellazione e riscrittura a blocchi. Grazie alla sua velocità e non volatilità, sta progressivamente sostituendo i dischi meccanici sotto forma di SSD.

| Tipo | Categoria | Cancellazione | Byte modificabili | Volatile | Tipico utilizzo |
|---|---|---|---|---|---|
| SRAM | Read/write | Elettrica | Sì | Sì | Cache di secondo livello |
| DRAM | Read/write | Elettrica | Sì | Sì | Memoria centrale (vecchia) |
| SDRAM | Read/write | Elettrica | Sì | Sì | Memoria centrale (recente) |
| ROM | Read-only | Impossibile | No | No | Elettrodomestici (prodotti in grandi volumi) |
| PROM | Read-only | Impossibile | No | No | Dispositivi (prodotti in piccoli volumi) |
| EPROM | Read-mostly | Raggi UV | No | No | Prototipazione di dispositivi |
| EEPROM | Read-mostly | Elettrica | Sì | No | Prototipazione di dispositivi |
| Flash | Read/write | Elettrica | No | No | "Pellicola" per macchine fotografiche digitali |

Figura 40: Confronto tra vari tipi di memoria

## 3.4 Chip della CPU e bus

Analizziamo come la CPU si interfaccia con la memoria e i dispositivi periferici, un processo strettamente legato al design dei bus utilizzati per la comunicazione.

### 3.4.1 Chip della CPU

Tutte le moderne CPU sono contenute in un **unico chip**, il che rende la loro interazione con il resto del sistema estremamente definita e circoscritta. Ogni chip comunica con il mondo esterno esclusivamente attraverso un insieme di pin (contatti), che possono emettere segnali, riceverli o svolgere entrambe le funzioni.
I pin di un chip CPU possono essere classificati in tre tipologie fondamentali:

- **Indirizzi (Address):** Questi pin emettono l'indirizzo della locazione di memoria che la CPU desidera leggere o scrivere. Un chip con m pin di indirizzo può indirizzare fino a $2^m$ locazioni di memoria.
- **Dati (Data):** Questi pin sono usati per scambiare i dati effettivi tra la CPU e la memoria o i dispositivi di input/output (I/O). Il numero di pin di dati (n) determina quanti bit possono essere letti o scritti in una singola operazione; un chip con 32 pin di dati è significativamente più veloce di uno con soli 8 pin, sebbene sia più costoso da produrre.
- **Controllo (Control):** Questi pin regolano il flusso e la temporizzazione dei dati e gestiscono diverse funzioni ausiliarie.

Oltre a questi, tutti i chip dispongono di pin standard per l'alimentazione (solitamente tra +1,2 e +1,5 volt), la messa a terra (ground) e il segnale di clock (un'onda quadra a una frequenza definita). I pin di controllo, pur variando molto tra i diversi produttori, possono essere raggruppati in sei categorie principali:

- **Controllo del bus:** Principalmente segnali in uscita che indicano alla memoria se la CPU desidera eseguire una lettura, una scrittura o altre operazioni.
- **Interrupt (Interruzioni):** Segnali in entrata dai dispositivi di I/O che informano la CPU del completamento di un'operazione o della presenza di errori.
- **Arbitraggio del bus:** Necessari per regolare il traffico sul bus, impedendo a due dispositivi (inclusa la CPU stessa) di tentare di usarlo contemporaneamente.
- **Segnalazione del coprocessore:** Pin dedicati a facilitare la comunicazione tra la CPU e chip specializzati, come quelli per il calcolo in virgola mobile o la grafica.
- **Stato (Status):** Forniscono o accettano informazioni sullo stato del sistema.
- **Vari:** Pin utili per il debug, il reset del computer o per garantire la compatibilità con chip di I/O più vecchi.

Il processo tipico di esecuzione vede la CPU porre l'indirizzo sui pin di indirizzo, attivare le linee di controllo per informare la memoria della richiesta di lettura, e infine accettare il dato che la memoria deposita sui pin di dati.

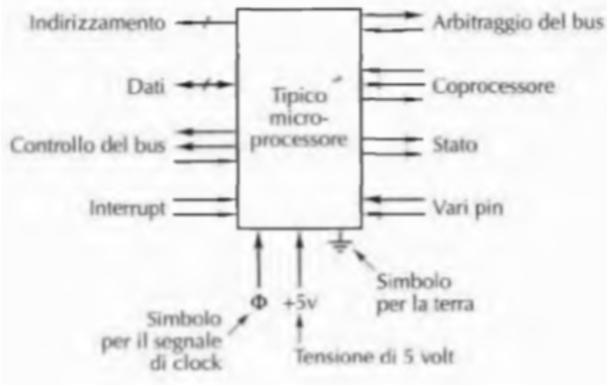



Figura 41: Una generica CPU. Le frecce indicano i segnali di input e di output. I trattini diagonali indicano pin multipli; per una specifica CPU un valore ne indica la quantità

### 3.4.2 Bus del calcolatore

Un bus è un collegamento elettrico che unisce diversi dispositivi, ma possono essere anche interni alla CPU. Sono classificati in base alla loro funzione e ciascun tipo di bus soddisfa certi requisiti e gode di proprietà specifiche.
Mentre i primi personal computer utilizzavano un unico **bus di sistema** (system bus) composto da 50-100 fili paralleli incisi sulla scheda madre, i sistemi moderni sono più complessi. Oggi si preferisce utilizzare un bus di memoria dedicato per il traffico ad alta velocità tra CPU e RAM, e uno o più bus separati per le periferiche di I/O. Questa separazione evita che i dispositivi più lenti intasino la comunicazione critica tra processore e memoria.

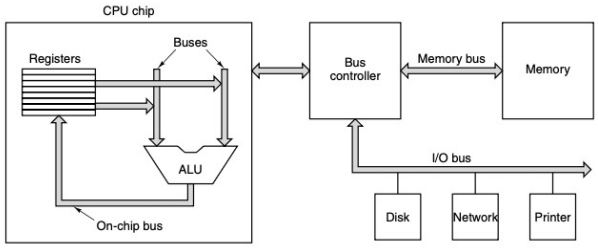



Figura 42: Un sistema informatico con più bus

Perché dispositivi di produttori diversi possano funzionare insieme, il bus deve seguire regole rigorose chiamate **protocollo del bus**. Queste regole includono:

- **Specifiche meccaniche:** La forma dei connettori e delle schede per garantire l'inserimento fisico.
- **Specifiche elettriche:** I livelli di tensione e i tempi necessari per i segnali. Inoltre, il bus dispone di diverse tipologie di linee: linee per gli indirizzi, linee per i dati e linee di controllo. Spesso non esiste una corrispondenza uno-a-uno tra i pin della CPU e le linee del bus; ad esempio, un decoder potrebbe essere necessario per convertire i segnali codificati di un chip in comandi specifici per il bus.

Esempi più famosi sono: il bus ISA, il bus EISA, il bus PCI, il bus SCSI, l'Universal Serial Bus.

| Master | Slave | Example |
|---|---|---|
| CPU | Memory | Fetching instructions and data |
| CPU | I/O device | Initiating data transfer |
| CPU | Coprocessor | CPU handing instruction off to coprocessor |
| I/O device | Memory | DMA (Direct Memory Access) |
| Coprocessor | CPU | Coprocessor fetching operands from CPU |

Figura 43: Esempio di bus master e slave

Un concetto fondamentale nella logica del bus è la distinzione tra dispositivi Master (attivi) e Slave (passivi).

- **Master:** è il dispositivo che prende l'iniziativa e avvia il trasferimento (solitamente la CPU o un controller del disco durante le operazioni DMA)
- **Slave:** è il dispositivo che attende le richieste e risponde ad esse. Le fonti sottolineano una regola ferrea: la memoria non può mai essere un master, ma agisce sempre e solo come slave.

Se i master trasmettono segnali deboli e sono collegati a molte periferiche, vengono connessi al bus mediante un chip chiamato **driver del bus**, che funge da amplificatore digitale; in modo analogo gli slave sono connessi al bus attraverso un ricevitore. Per le periferiche coprenti entrambi i ruoli si utilizza un chip chiamato **trasmettitore-ricevitore del bus**.
La progettazione di un bus deve bilanciare diversi fattori critici: la **larghezza** (numero di linee di indirizzo e dati), il **clocking** (sincrono o asincrono), l'**arbitraggio** (chi decide chi usa il bus quando più master lo richiedono) e le **operazioni** supportate (lettura, scrittura, trasferimenti a blocchi).

### 3.4.3 Ampiezza del bus

Nella progettazione dei bus maggiore è il numero di linee d'indirizzo di un bus, maggiore sarà la quantità di memoria che la CPU potrà indirizzare direttamente. . Il numero di linee di indirizzo determina la quantità massima di memoria che la CPU può indirizzare direttamente. Se un bus possiede n linee di indirizzo, la CPU può gestire fino a $2^n$ locazioni di memoria distinte.

- **Compromesso costi-benefici:** L'aumento del numero di linee richiede più cavi fisici, occupa più spazio sulla scheda madre e necessita di connettori più grandi, aumentando il costo complessivo del sistema.

Con il tempo i produttori hanno cercato di migliorare il bus aumentando la larghezza di banda dei dati su un bus. Questo fu possibile con due tecniche: diminuendo il periodo di clock del bus (più trasferimenti al secondo) e aumentando la larghezza dei dati del bus.
Aumentare la velocità del bus (frequenza) è più difficile che aumentarne la larghezza a causa di due fattori principali:

- **Disallineamento del bus:** È un fenomeno per cui i segnali su linee diverse viaggiano a velocità leggermente differenti. Maggiore è la velocità del bus, più grave diventa questo problema, portando a possibili errori di sincronizzazione.
- **Retrocompatibilità:** Cambiare drasticamente la velocità del bus renderebbe inutilizzabili le vecchie schede progettate per standard più lenti, danneggiando sia i produttori che i consumatori. Per questo motivo, spesso si preferisce aggiungere più linee di dati piuttosto che aumentare la frequenza.

Per contenere i costi e lo spazio fisico senza rinunciare a un numero elevato di linee, alcuni progettisti optano per un bus multiplexato.

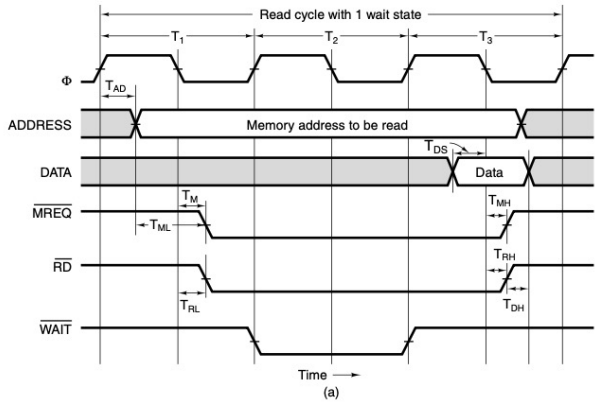
- In questo design, le stesse linee vengono condivise per gli indirizzi e per i dati: all'inizio del ciclo del bus le linee portano l'indirizzo, e successivamente vengono utilizzate per il dato.
- Sebbene questa scelta riduca il numero di pin necessari (e quindi il costo), rende il sistema più lento poiché i due tipi di informazioni non possono viaggiare contemporaneamente.

### 3.4.4 Temporizzazione del bus

I bus possono essere separati in due categorie in base alla loro temporizzazione. Un **bus sincrono** è governato da un oscillatore a cristallo che genera un'onda quadra a frequenza costante (solitamente tra 5 e 133 MHz). Il segnale del clock scandisce le varie transizioni dei segnali e il passaggio da un ciclo di bus al ciclo successivo. Tutte le operazioni sul bus richiedono un numero intero di questi cicli, detti **cicli di bus**. Quando la CPU vuole leggere dalla memoria:

- la CPU, nel primo ciclo di clock, fornisce l'indirizzo della parola sulle linee di indirizzo;
- dopo che le linee d'indirizzo si sono stabilizzate sui nuovi valori, vengono asserite $\overline{MREQ}$ (indica che si accede alla memoria) e $\overline{RD}$ (assertita per le letture e negato per le scritture)
- se la memoria impiega tempo supplementare a rispondere asserisce $\overline{\text{WAIT}}$ per segnalare alla CPU di non aspettarla. Questa azione inserisce alcuni stati di attesa (cicli di bus addizionali) finché la memoria non completi l'operazione;
- una volta che la memoria è pronta nega $\overline{\text{WAIT}}$ e mette i dati sulle linee apposite;
- la CPU legge le linee dati, memorizzando il valore in un registro;
- dovendo leggere i dati la CPU nega $\overline{MREQ}$ e $\overline{RD}$.

In ogni caso, master e slave devono rispettare dei vincoli temporali (es. rispetto ai fronti di salita e di discesa del clock). Ad esempio le specifiche di temporizzazione richiedono che i dati siano disponibili sulle linee prima che la CPU legga, per dare la possibilità alla linea di stabilizzarsi. Inoltre con le specifiche di temporizzazione è garantito che l'indirizzo venga impostato prima che sia asserito.




| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $T_{AD}$ | Address output delay | | 4 | nsec |
| $T_{ML}$ | Address stable prior to $\overline{MREQ}$ | 2 | | nsec |
| $T_M$ | $\overline{MREQ}$ delay from falling edge of Φ in $T_1$ | | 3 | nsec |
| $T_{RL}$ | RD delay from falling edge of Φ in $T_1$ | | 3 | nsec |
| $T_{DS}$ | Data setup time prior to falling edge of Φ | 2 | | nsec |
| $T_{MH}$ | $\overline{MREQ}$ delay from falling edge of Φ in $T_3$ | | 3 | nsec |
| $T_{RH}$ | $\overline{RD}$ delay from falling edge of Φ in $T_3$ | | 3 | nsec |
| $T_{DH}$ | Data hold time from negation of $\overline{RD}$ | 0 | | nsec |

(b)

Figura 44: (a) Temporizzazione di una lettura di un bus sincrono. (b) Specifiche di alcuni tempi critici.

A differenza dei primi, i **bus asincroni** non utilizzano un clock centrale. La velocità di ogni transazione è determinata esclusivamente dalla velocità di risposta dei dispositivi coinvolti. Infatti, uno dei problemi del bus sincrono è proprio che qualsiasi operazione sul bus si svolge in tempi multipli del clock del bus. La comunicazione avviene tramite una sequenza di causa-effetto basata su quattro eventi principali:

- Il master asserisce $\overline{MSYN}$ (Master Synchronization) dopo aver impostato l'indirizzo.
- Lo slave completa il lavoro e risponde asserendo $\overline{SSYN}$ (Slave Synchronization)
- Il master preleva i dati e nega (disattiva) $\overline{MSYN}$
- Lo slave, vedendo $\overline{MSYN}$ disattivato, nega a sua volta $\overline{SSYN}$, riportando il sistema allo stato originale.

Ogni evento è causato da un evento precedente e non da un impulso del clock. Se una particolare coppia master-slave è lenta essa non influisce in alcun modo su una successiva coppia master-slave la cui velocità potrebbe essere molto più elevata.

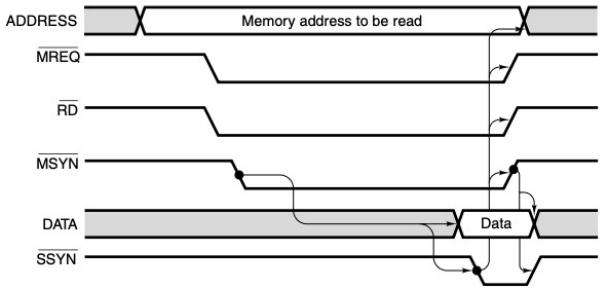



Figura 45: Operazioni di un bus asincrono

Nonostante la flessibilità e l'adattabilità tecnologica dei bus asincroni, la maggior parte dei bus reali è sincrona. La ragione principale è ingegneristica: i bus sincroni sono molto più facili da progettare, costruire e testare, poiché il comportamento del sistema è rigidamente prevedibile sulla base di intervalli di tempo fissi.

### 3.4.5 Arbitraggio del bus

Nel caso in cui due dispositivi vogliano accedere al bus, si può incorrere in un meccanismo di arbitraggio, che può essere centralizzato o decentralizzato.

1. **Arbitraggio Centralizzato:** In questo schema, un singolo componente hardware, chiamato arbitro del bus, decide quale dispositivo può procedere. Il bus contiene un'unica linea di richiesta OR-cablata che in qualsiasi momento può essere asserita da uno o più dispositivi. L'arbitro non è in grado di capire quanti hanno fatto la richiesta, ma solo se ce n'è qualcuna o nessuna, quindi la concede asserendo la linea per la concessione del bus.

    - **Daisy Chaining (collegamento a festone):** È la forma più semplice. Tutti i dispositivi condividono una singola linea di richiesta bus (wired-OR). Quando l'arbitro riceve una richiesta, invia un **segnale di concessione (bus grant)**. Questa linea attraversa tutti i dispositivi in serie. Il primo dispositivo della catena che ha fatto richiesta intercetta il segnale e prende il controllo del bus; se non ha bisogno del bus, passa il segnale al vicino.
    - **Priorità Implicita:** Nel daisy chaining, la priorità è determinata dalla distanza fisica dall'arbitro: i dispositivi più vicini vincono sempre sui più lontani.
    - **Livelli di Priorità Multipli:** Per evitare che la priorità dipenda solo dalla posizione, molti bus utilizzano più livelli di priorità (4, 8, 16). Ogni livello ha le sue linee di richiesta e concessione. L'arbitro concede il bus al livello più alto; all'interno dello stesso livello, si usa il daisy chaining.
    - **Ottimizzazione dei cicli:** Alcuni sistemi usano una terza linea per segnalare che il bus è stato occupato, permettendo all'arbitro di negoziare la successiva concessione mentre il trasferimento attuale è ancora in corso.

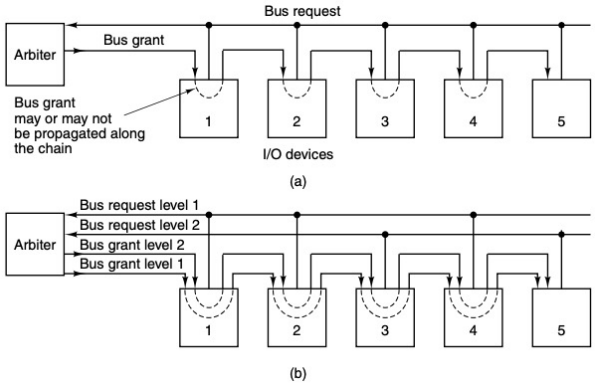



Figura 46: (a) Arbitro del bus centralizzato e a un livello che utilizza una daisy chaining (b)Stesso arbitro, ma a due livelli

2. **Arbitraggio Decentralizzato:** Questo approccio elimina la necessità di un arbitro centrale, riducendo i costi e il rischio di un "single point of failure" (arbitro centrale smette di funzionare, nessun dispositivo (inclusa la CPU) può più ottenere l'accesso al bus per comunicare, rendendo l'intero computer inutilizzabile).

    - **Monitoraggio delle linee:** Un metodo prevede che ogni dispositivo abbia una propria linea di richiesta prioritaria e monitori tutte le altre. Alla fine di ogni ciclo, ogni dispositivo sa se è il richiedente a priorità più alta e se può procedere.
    - **Schema a tre linee:** Un altro metodo usa solo tre linee: una di richiesta, una di "occupato" (BUSY) e una di arbitraggio collegata in daisy chain. Un dispositivo può diventare master solo se il bus è libero e il segnale di arbitraggio in ingresso è attivo. Se decide di prendere il bus, disattiva il segnale in uscita verso i vicini a valle, garantendo che solo un dispositivo diventi master.

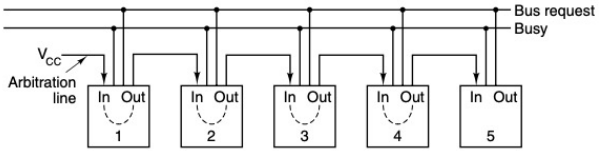



Figura 47: Arbitro del bus decentralizzato

Con un semplice ragionamento si deduce che fra i dispositivi che richiedono il bus lo ottiene quello che si trova più a sinistra.

### 3.4.6 Operazioni del bus

I bus non si limitino a semplici letture o scritture di singole parole, ma supportino operazioni più complesse per ottimizzare le prestazioni e gestire la comunicazione tra i componenti del sistema.
Mentre i cicli di bus standard gestiscono una parola alla volta, i trasferimenti a blocchi consentono di spostare intere sequenze di dati in modo molto più efficiente.

- **Utilità:** Questa modalità è fondamentale per il caching, dove è necessario prelevare un'intera linea di cache (ad esempio 8 parole consecutive) in una sola volta.
- **Meccanismo:** Durante il primo ciclo di clock ($T_1$), il bus master comunica allo slave il numero di parole da trasferire. Lo slave risponde restituendo una parola ad ogni ciclo successivo finché il contatore non si sia esaurito.
- **Efficienza:** Questo approccio, spesso chiamato modalità burst, riduce il numero di cicli necessari; ad esempio, una lettura di 4 parole può richiedere solo 6 cicli invece dei 12 o più necessari per quattro letture singole separate.

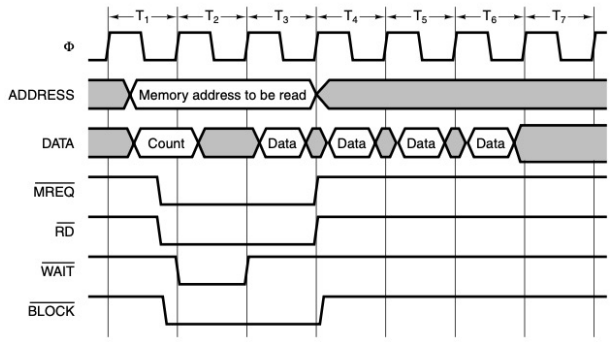



Figura 48: Trasferimento di un blocco

Esistono anche altri tipi di cicli di bus.

- Ad esempio il ciclo di bus **leggi-modifica-scrivi** che consente a una qualsiasi CPU di leggere una parola dalla memoria, analizzarla e riscriverla in memoria, tutto senza rilasciare il bus. In un sistema con più CPU, questo tipo di ciclo evita che altre CPU che intendono utilizzare il bus riescano a impadronirsene interferendo con l'operazione della prima CPU.
- Un altro importante ciclo di bus serve a gestire gli **interrupt**. Quando la CPU ordina a un dispositivo di I/O di effettuare qualche operazione si aspetta di solito un interrupt alla fine del lavoro; la segnalazione di questi interrupt richiede l'utilizzo del bus. Dato che più dispositivi potrebbero voler generare un interrupt simultaneamente, si verificano gli stessi tipi di problemi di arbitraggio visti nel caso dei cicli di bus ordinari.

La soluzione più usuale prevede di assegnare priorità ai dispositivi e di utilizzare un arbitro centralizzato per dare priorità al dispositivo per cui la temporizzazione è più critica. Esistono chip **controllori di interrupt** di tipo standard che sono utilizzati in modo molto diffuso, tra cui il chip Intel 8259A.

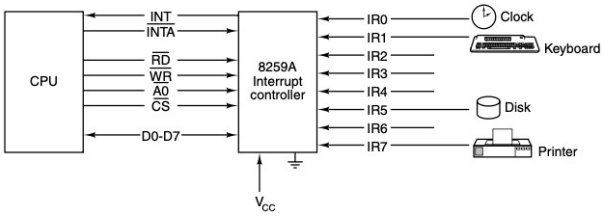



Figura 49: Utilizzo del controllore degli interrupt 8259A

Il chip dispone di **otto ingressi di richiesta interruzione (IR0-IR7)** ai quali possono essere collegati direttamente altrettanti controller di I/O, come la tastiera, il timer di sistema, i dischi o la stampante. Quando una o più periferiche richiedono attenzione, l'8259A valuta le priorità e attiva la linea **INT (Interrupt)** collegata direttamente alla CPU.
L'interazione tra l'8259A e la CPU segue una sequenza precisa:

1. **Segnalazione:** L'8259A riceve una richiesta su una linea IRx e asserisce il pin INT della CPU.
2. **Riconoscimento:** Se la CPU è pronta a gestire l'interruzione, invia un impulso di risposta sul pin INTA (Interrupt Acknowledge).
3. **Identificazione (Vettore):** Ricevuto l'impulso INTA, l'8259A deposita sul bus dati un numero chiamato **vettore di interruzione**.
4. **Esecuzione:** La CPU utilizza questo numero come indice in una tabella (tabella dei vettori di interruzione) per trovare l'indirizzo della procedura di servizio (ISR) da eseguire per quella specifica periferica.

Il chip è altamente flessibile grazie a diversi registri interni che la CPU può leggere o scrivere utilizzando cicli di bus standard (tramite i pin $\overline{RD}, \overline{WR}, \overline{CS}$ e $\overline{A0}$). Attraverso questi registri, il software può:

- **Mascherare le interruzioni:** Ignorare selettivamente alcuni segnali IRx
- **Impostare le modalità:** Definire come gestire le priorità (ad esempio, priorità fissa o rotativa).
- **Reset:** Inviare un codice speciale al termine della gestione di un'interruzione per segnalare al controller che è pronto per la successiva.

## 3.5 Esempi di CPU

### 3.5.1 Pentium 4

Il Pentium 4 ha segnato il passaggio dalla vecchia microarchitettura P6 (usata nei Pentium Pro, II e III) alla nuova NetBurst.

- **Pipeline Profonda:** Progettata per raggiungere frequenze di clock molto elevate, NetBurst utilizza una pipeline molto lunga che permette di suddividere il lavoro in piccoli passi atomici.
- **ALU a Doppia Velocità:** Il chip contiene due ALU (Unità Aritmetico-Logiche) che funzionano al doppio della frequenza nominale del clock. Ad esempio, in un processore a 3 GHz, le ALU operano a 6 GHz, permettendo di eseguire fino a 12 miliardi di operazioni su interi al secondo.
- **Hyperthreading:** Introdotto nella versione a 3,06 GHz, permette alla CPU di gestire due thread di controllo in parallelo. L'hardware duplica alcune risorse (come il Program Counter e la mappa dei registri) per dare l'illusione al sistema operativo di avere due CPU fisiche, migliorando le prestazioni del 25% con un aumento della superficie del chip di solo il 5%.

Il sistema di caching del Pentium 4 è stato ottimizzato per alimentare costantemente la sua pipeline profonda:

- **L1 Trace Cache:** A differenza delle cache tradizionali che memorizzano byte grezzi, la cache L1 (8 KB) del Pentium 4 è una "cache delle tracce" che memorizza fino a 12.000 micro-operazioni già decodificate. Questo evita di dover decodificare ripetutamente la stessa istruzione ISA complessa.

- **L2 e L3 Cache:** La cache di secondo livello (da 256 KB a 1 MB) è unificata e di tipo write-back. Versioni speciali come la Extreme Edition includono una cache L3 da 2 MB.
- **Bus Dati e Indirizzi:** Sebbene sia una macchina a 32 bit per il software, l'hardware comunica con la memoria con unità di 64 bit. Il bus indirizzi a 36 bit permette di indirizzare fino a 64 GB di RAM.

Le transazioni sul bus di memoria sono altamente strutturate e organizzate a pipeline, divise in 6 fasi indipendenti:

1. **Arbitraggio:** Determina quale master prende il controllo.
2. **Richiesta:** Emissione dell'indirizzo e dei comandi.
3. **Segnalazione Errore:** Notifica di eventuali errori di parità.
4. **Snoop:** Fondamentale nei sistemi multiprocessore per garantire la coerenza della cache.
5. **Risposta:** Conferma della disponibilità dei dati.
6. **Dati:** Trasferimento effettivo.

Il Pentium 4 è stato il chip che ha evidenziato i limiti della dissipazione del calore:

- **Transistor:** Le versioni sono passate da 42 milioni (0,18 micron) a 55 milioni di transistor (0,09 micron).
- **Calore:** A 3,6 GHz, il chip consuma circa 115 watt, riscaldandosi quanto una lampadina da 100 watt. Questa emissione termica ha portato alla cancellazione della versione a 4 GHz nel 2004, poiché gli ingegneri non riuscivano a dissipare il calore in modo efficiente per i desktop.
- **Thermal Throttling:** Se i sensori interni rilevano una temperatura di 130°C, la CPU riduce automaticamente le prestazioni per raffreddarsi e prevenire danni fisici.

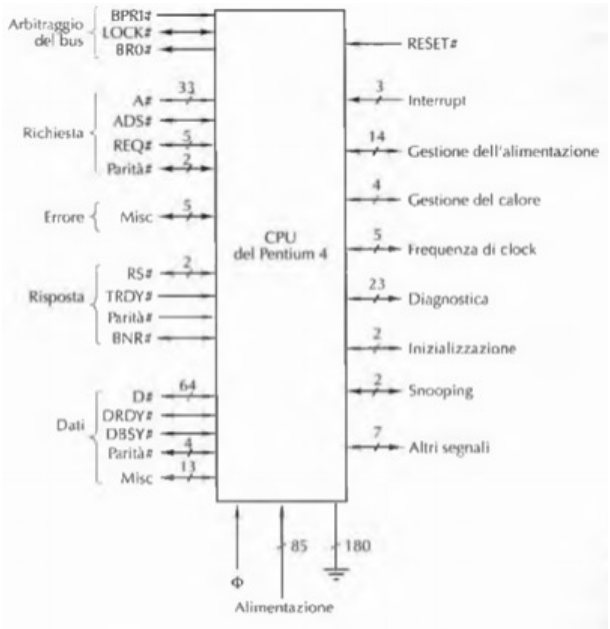



Figura 50: Disposizione logica dei contatti del Pentium 4. I nomi interamente in maiuscolo sono i nomi ufficiali di Intel per i singoli segnali; gli altri sono nomi di gruppi di segnali o descrizioni dei segnali

### 3.5.2 Intel Core i7

Ecco un approfondimento dettagliato basato sui vari aspetti del chip:

- **Complessità e Produzione:** Il Core i7 (versione Sandy Bridge) è composto da circa 1,16 miliardi di transistor realizzati con un processo a 32 nanometri. Può raggiungere frequenze di clock fino a 3,5 GHz.

- **Parallelismo:** È una macchina superscalare a 4 livelli, capace di emettere ed eseguire fino a quattro istruzioni per ciclo di clock.
- **Multi-core e Hyperthreading:** Il chip contiene più core indipendenti (solitamente da 2 a 6 per i modelli desktop). Supporta l'hyperthreading (o multithreading simultaneo), che permette di gestire due thread hardware per ogni core, aiutando a nascondere le latenze (come i cache miss) alternando velocemente l'esecuzione.

Il sistema di cache è strutturato su tre livelli per massimizzare il throughput dei dati:

- **L1 (Livello 1)**: Ogni core ha la propria cache divisa: 32 KB per le istruzioni e 32 KB per i dati.
- **L2 (Livello 2):** Ogni core ha una cache unificata (istruzioni + dati) da 256 KB.
- **L3 (Livello 3):** È una cache unificata condivisa tra tutti i core, con dimensioni che variano da 4 a 15 MB (fino a 20 MB in alcuni modelli).

**Coerenza (Snooping):** Per garantire che tutti i core vedano dati aggiornati, ogni CPU esegue lo snooping ("ficcanaso") sul bus di memoria: se un core tenta di leggere un dato che un altro core ha modificato nella propria cache privata, quest'ultimo interviene fornendo il dato corretto prima della RAM. Il Core i7 integra internamente molte funzioni che un tempo risiedevano sulla scheda madre:

- **Memoria DDR3:** Supporta due canali DDR3 indipendenti a 666 MHz (1333 milioni di transazioni/sec). Poiché l'interfaccia è a 64 bit, i due canali insieme forniscono una larghezza di banda fino a 20 GB/s.
- **PCI Express (PCIe):** Dispone di un'interfaccia a 16 linee (x16) integrata per collegare direttamente le periferiche (come schede video) con una banda di 16 GB/s.
- **QPI (Quick Path Interconnect):** Utilizzato nelle versioni high-end per connettere più processori fisici tra loro, gestendo richieste di coerenza della cache e interruzioni interprocessore.
- DMI (Direct Media Interface): Collega la CPU al chipset (modelli P67 e ICH10) a una velocità di circa 2.5 GB/s.

Gestione Termica ed Energetica:

- **Consumo:** Il processore consuma tra 17 e 150 watt.
- **Thermal Throttling:** Se i sensori interni rilevano una temperatura di 130°C, la CPU attiva il "throttling", ovvero esegue le istruzioni solo ogni N cicli di clock per raffreddarsi rapidamente ed evitare danni fisici.
- **Stati di Sospensione:** Sono previsti cinque stati, dall'attività piena al "deep sleep", in cui le cache vengono spente e i valori dei registri preservati per risparmiare energia.

Per evitare che la CPU rimanga senza dati, il bus DDR3 opera in modo pipelined, gestendo fino a quattro transazioni simultanee. Ogni richiesta segue tre fasi:

- ACTIVATE: Apre una riga della memoria DRAM per prepararla a successivi accessi;
- READ/WRITE: Accesso effettivo ai dati (anche in modalità burst, accessi multipli).
- PRECHARGE: Chiude la riga per prepararla alla successiva operazione.

Il Core i7 è definito come una CPU multicore, il che significa che un singolo stampo di silicio (die) contiene fisicamente più processori indipendenti (core), solitamente da 2 a 6.

- **Vantaggio:** Sfruttando thread e lock, i programmatori possono ottenere incrementi di velocità significativi facendo lavorare i diversi core in parallelo su compiti distinti.
- **Costi e Limiti:** Inserire due CPU complete su un unico chip raddoppia quasi l'area del silicio se ognuna ha le proprie cache, raddoppiando di fatto i costi di produzione. Inoltre, non tutte le applicazioni desktop hanno abbastanza parallelismo intrinseco da giustificare l'uso di due CPU complete.

Oltre a essere multicore, ogni singola CPU all'interno del Core i7 è dotata di hyperthreading

- **Funzionamento:** Questa tecnologia permette a più thread hardware di essere attivi contemporaneamente sullo stesso core. Per il sistema operativo, un chip hyperthreaded appare come un processore duale (o multiplo) che condivide la stessa cache e memoria principale.
- **Gestione delle Latenze:** L'obiettivo principale è tollerare latenze molto brevi, come i cache miss, effettuando uno switch immediato tra i thread hardware. Al contrario, il multithreading basato sul software richiede centinaia di cicli per gestire lo scambio di contesto.
- **Efficienza:** Intel ha rilevato che un aumento di appena il 5% dell'area del chip per supportare l'hyperthreading può portare a un incremento delle prestazioni del 25% in molte applicazioni.

Poiché i due thread condividono l'hardware dello stesso core, il sistema NetBurst (e successivamente Sandy Bridge) adotta diverse strategie:

- **Duplicazione:** Alcune risorse sono replicate per ogni thread, come il Program Counter, la tabella di mappatura dei registri e il controllore delle interruzioni.
- **Ripartizione (Partitioning):** Le code interne alla pipeline sono divise rigidamente: metà degli slot sono riservati al thread 1 e metà al thread 2, garantendo che un thread non "soffochi" l'altro.
- **Condivisione Dinamica (Soglia):** Risorse come lo scheduler sono condivise dinamicamente ma con una soglia massima di utilizzo per evitare che un thread monopolizzi tutto il core.

### 3.5.3 UltraSPARC III

L'UltraSPARC III rappresenta l'implementazione di Sun Microsystems dell'architettura RISC a 64 bit (conforme allo standard SPARC Version 9). A differenza dei processori desktop come il Pentium 4, questa CPU è stata concepita specificamente per alimentare server multiprocessore di grandi dimensioni con memoria condivisa. Ecco un approfondimento dettagliato sulle sue caratteristiche hardware e architetturali:

1. **Architettura e Parallelismo:**
   - **Design RISC Puro:** L'UltraSPARC III non necessita di complessi traduttori da istruzioni CISC a micro-operazioni; le sue istruzioni native sono già semplici e regolari.
   - **Pipeline Profonda:** Dispone di 6 pipeline interne specializzate: due per gli interi (a 14 stadi), due per la virgola mobile, una per le operazioni di load/store e una per la gestione dei salti.
   - **Emissione Multipla:** È in grado di lanciare quattro istruzioni per ciclo di clock, garantendo prestazioni elevate nonostante una frequenza di clock nominale inferiore rispetto ai concorrenti Intel.
   - **Istruzioni VIS 2.0:** Include il Visual Instruction Set per accelerare grafica 3D, compressione dati e decodifica video (MPEG) in tempo reale.
2. **Caratteristiche Fisiche e Produzione:**
   - **Evoluzione Tecnologica:** I primi modelli operavano a 600 MHz (tecnologia a 0,18 micron e alluminio) con 29 milioni di transistor. Le versioni successive hanno raggiunto 1,2 GHz utilizzando collegamenti in rame a 0,13 micron.
   - **Packaging:** Si presenta come un Land Grid Array (LGA) con 1368 contatti (pin) disposti in una matrice 37x37.
3. **Gerarchia della Memoria e Caching:**
   - **Cache di Primo Livello (L1):** Integrata nel chip, divisa in 32 KB per le istruzioni e 64 KB per i dati.
   - **Cache di Secondo Livello (L2):** A differenza del Pentium 4, la cache L2 dell'UltraSPARC III è esterna al chip (SRAM commerciali). Questa scelta offre flessibilità (dimensioni da 1 a 8 MB), ma richiede un bus molto largo (256 bit) per trasferire un intero blocco di cache da 32 byte in un unico ciclo.
   - **Cache Specializzate:** Integra una cache di prefetch (2 KB) per anticipare i dati necessari e una cache di scrittura (2 KB) per ottimizzare i trasferimenti verso la L2.
4. **Gestione del Bus e del Multiprocessing:**

- **Addressing:** Il bus indirizzi a 43 bit permette di gestire teoricamente fino a 8 Terabyte (TB) di memoria centrale.
- **Banda Passante:** Il bus dati è largo 128 bit e opera a 150 MHz, fornendo una larghezza di banda di 2,4 GB/s.
- **Interfaccia UPA:** Utilizza l'architettura Ultra Port Architecture (UPA), che può essere implementata come bus o commutatore, per connettere più CPU UltraSPARC alle memorie.
- **Chip UDB II:** Include l'UltraSPARC Data Buffer II, un componente che funge da memoria tampone tra la CPU e il sistema di memoria, permettendo ai due elementi di funzionare in modo asincrono e migliorando la gestione dei codici a correzione d'errore (ECC).

### 3.5.4 Chip 8051

L'Intel 8051 è un microcontrollore a 8 bit introdotto nel 1980, diventato uno standard "di fatto" e un pilastro nel settore dei sistemi integrati (embedded) grazie alla sua longevità e versatilità.

**Contesto Storico e Successo di Mercato**

La storia dell'8051 inizia nel 1976, quando Intel rilasciò l'8748 per soddisfare i costruttori che desideravano integrare CPU, memoria e I/O in un unico chip, riducendo i costi rispetto al precedente Intel 8080 che richiedeva componenti separati. Nonostante sia un'architettura di vecchia data, l'8051 è ancora oggi più venduto dei moderni processori Pentium, con volumi che superano gli 8 miliardi di unità all'anno. Il segreto del suo successo risiede nel prezzo estremamente basso (pochi centesimi di euro), nella vasta disponibilità di software, librerie e compilatori, e nella presenza di numerosi produttori che lo realizzano sotto licenza, garantendo velocità da 12 a 100 MHz.

**Architettura Hardware e Segnali**

Il chip originale contiene circa 60.000 transistor e dispone di caratteristiche ottimizzate per il controllo di dispositivi fisici:

- **Memoria interna:** Integra 4 KB di ROM (o 8 KB nell'8052) per il programma e 128 byte di RAM (o 256 byte nell'8052) per i dati.
- **I/O Versatile:** La caratteristica distintiva è la presenza di 32 linee di I/O organizzate in quattro porte da 8 bit, che permettono di collegare direttamente interruttori, LED o sensori.
- **Gestione della Memoria Esterna:** Se la memoria interna è insufficiente, il chip può indirizzare fino a 64 KB di memoria esterna per il codice e altrettanti per i dati tramite 16 linee di indirizzo e 8 linee di dati.
- **Multiplexing:** Per ridurre il numero di pin, le 8 linee di indirizzo meno significative condividono gli stessi contatti fisici con le linee dei dati; il segnale ALE (Address Latch Enable) indica alla memoria quando l'indirizzo sul bus è valido.

**Caratteristiche Uniche del Livello ISA**

L'architettura dell'insieme d'istruzioni (ISA) dell'8051 presenta soluzioni ingegnose per massimizzare l'efficienza in spazi ridotti:

- **Indirizzabilità a livello di bit:** Esiste un'area speciale di 16 byte (indirizzi 32-47) in cui ogni singolo bit può essere manipolato individualmente tramite istruzioni booleane, evitando le costose operazioni di mascheramento richieste da altri processori.
- **Finestre di Registri:** Dispone di quattro insiemi di otto registri (R0-R7); durante un'interruzione, il sistema può semplicemente cambiare l'insieme attivo invece di salvare i registri, garantendo una risposta estremamente rapida ai segnali in tempo reale.
- **Architettura Harvard:** Mantiene spazi di indirizzamento separati per programma e dati, utilizzando il segnale PSEN (Program Store Enable) per leggere le istruzioni.

**Ciclo di Esecuzione e Temporizzazione**

L'8051 è un processore sincrono dove la maggior parte delle istruzioni richiede un solo ciclo di clock, diviso internamente in sei stati: prelievo (fetch) dalla ROM, decodifica, preparazione degli operandi, caricamento dei latch TMP1 e TMP2, esecuzione della ALU e scrittura del risultato finale.

## 3.6 Esempi di BUS

### 3.6.1 Il Bus ISA (Industry Standard Architecture)

Il bus ISA rappresenta il retaggio del mondo PC originale.

- **Origini:** Introdotto con l'IBM PC (basato su Intel 8088), era un bus semplice a 8 bit che operava a 4,77 MHz.
- **Evoluzione:** Con l'avvento del PC/AT (Intel 80286), fu esteso a 16 bit aggiungendo un secondo connettore sulla scheda madre per mantenere la retrocompatibilità con le vecchie schede.
- **Limiti:** Operava a un massimo di 8,33 MHz con una larghezza di banda di soli 16,7 MB/s, diventando presto un collo di bottiglia per le applicazioni grafiche.
- **Stato attuale:** Sebbene sia stato esteso a 32 bit (EISA), è oggi considerato obsoleto e relegato ai musei del computer.

### 3.6.2 Il Bus PCI (Peripheral Component Interconnect)

Progettato da Intel nel 1990, il PCI è nato per gestire la crescente richiesta di banda dei video a pieno schermo e in movimento.

- **Caratteristiche tecniche:** Inizialmente trasferiva 32 bit per ciclo a 33 MHz (133 MB/s); le versioni successive sono arrivate a 66 MHz e 64 bit, raggiungendo i 528 MB/s.
- **Funzionamento:** È un bus sincrono che utilizza il multiplexing (indirizzi e dati condividono gli stessi pin in cicli diversi) per ridurre il numero di contatti fisici.
- **Arbitraggio:** Utilizza un arbitro centrale che gestisce le richieste (REQ#) e le concessioni (GNT#) per ogni dispositivo.
- Adozione: Intel ha reso i brevetti di pubblico dominio, rendendolo lo standard di fatto per quasi tutti i computer dalla generazione Pentium in poi.

### 3.6.3 PCI Express (PCIe)

Nonostante il nome, il PCI Express non è un bus nel senso tradizionale, ma una rete point-to-point a commutazione di pacchetto.

- **Architettura a "Lane":** Al posto di un bus parallelo largo, utilizza collegamenti seriali ad alta velocità chiamati lane (corsie). Una singola lane PCIe 1.0 supporta 2,5 Gbps.
- **Scalabilità:** È possibile combinare più lane (x1, x8, x16, x32); uno slot x16 in versione 3.0 può raggiungere una banda di 16 GB/s.
- **Protocollo a strati:** Utilizza una pila di protocolli (Fisico, Trasmissione, Transazione, Software) simile a quella delle reti locali come Ethernet.
- **Vantaggi:** Le connessioni seriali eliminano i problemi di disallineamento dei segnali (skew) tipici dei bus paralleli e permettono connettori molto più piccoli.

### 3.6.4 USB (Universal Serial Bus)

L'USB è stato creato da un consorzio di aziende per collegare periferiche a bassa velocità senza dover aprire il case del PC o riavviare il sistema.

- **Topologia:** Il sistema è organizzato ad albero, con una radice (root hub) interna al PC che può gestire fino a 127 dispositivi.
- **Trasferimento dati:** L'hub principale invia un frame ogni 1,00 ms per mantenere sincronizzati tutti i dispositivi.
- **Versioni e velocità:** Si è evoluto da 1,5 Mbps (v1.0) e 12 Mbps (v1.1) fino ai 480 Mbps (v2.0) e ai 5 Gbps dello standard 3.0.
- **Versatilità:** Supporta diversi tipi di traffico: isocrono (per audio/video in tempo reale), bulk (grandi trasferimenti non urgenti), interrupt (input lenti come tastiere) e controllo.

## 3.7 Interfacce

Le interfacce di I/O rappresentano l'ultimo tassello fondamentale per permettere al computer di comunicare con il mondo esterno attraverso le cosiddette porte di I/O. Tra le interfacce più comuni si trovano i chip UART, USART, i controllori di dischi e le interfacce parallele o PIO (Parallel Input/Output).

### 3.7.1 Le Interfacce I/O: Il caso del chip PIO

Un esempio tipico di interfaccia programmabile è l'Intel 8255A, un chip dotato di 24 linee di I/O che possono interfacciarsi con quasi ogni dispositivo logico digitale, come tastiere, interruttori, luci o stampanti.

- **Struttura interna:** Il chip è organizzato in tre porte indipendenti da 8 bit (A, B e C).
- **Programmabilità:** Attraverso un registro di controllo interno, la CPU può configurare ogni porta per funzionare come ingresso (bit 0) o come uscita (bit 1).
- **Connettività:** Il chip dispone di otto linee collegate direttamente al bus dati, una linea di Chip Select (CS), linee di lettura (RD) e scrittura (WR), una linea di reset e due linee di indirizzo (A0-A1) usate per selezionare uno dei quattro registri interni.

### 3.7.2 La Decodifica dell'Indirizzo

La decodifica dell'indirizzo è il meccanismo che determina come e quando il segnale di Chip Select (CS) viene asserito per attivare un determinato chip di memoria o di I/O. Esistono due filosofie principali per gestire queste interfacce:

1. **I/O Standard:** Il dispositivo viene selezionato tramite una linea di bus esplicita che indica un riferimento a un dispositivo di I/O e non alla memoria.
2. **I/O Mappato in Memoria:** Al dispositivo vengono assegnati alcuni byte dello spazio di indirizzamento della memoria (ad esempio, 4 byte per i registri di un chip PIO).

Tecniche di Decodifica:

- **Decodifica Completa:** Utilizza porte logiche (come porte OR o NAND a più ingressi) collegate alle linee d'indirizzo per garantire che il chip risponda a un unico indirizzo specifico. Ad esempio, per selezionare un chip solo quando i bit A11-A15 sono tutti zero, si può usare una porta NOR a cinque ingressi.
- **Decodifica Parziale:** Per semplificare il circuito, si possono usare solo i bit più significativi (es. solo A15) per pilotare il Chip Select. Sebbene sia economica (richiede meno chip logici), questa tecnica causa il fenomeno per cui lo stesso dispositivo risponde a molteplici indirizzi diversi (alias), limitando la futura espandibilità del sistema.
- **Uso di Decodificatori:** Un'altra tecnica prevede l'impiego di chip decodificatori (come i decoder 3-a-8), che accettano n bit in ingresso e attivano una sola delle $2^n$ linee di uscita, permettendo di selezionare agevolmente uno tra diversi chip di RAM o I/O.

# 4 Livello di microarchitettura

Al di sopra del livello logico digitale si trova il livello di microarchitettura incaricato di implementare il livello ISA (Instruction Set Architecture, "architettura dell'insieme di istruzioni"). Il modo in cui viene progettato il livello di microarchitettura non dipende solamente dall'ISA che si intende implementare, ma anche dagli obiettivi di costi e prestazioni del calcolatore. Molti ISA moderni sono costituiti da istruzioni semplici (principalmente RISC) che generalmente è possibile eseguire in un unico ciclo di clock. Nel caso di ISA più complessi (CISC) l'esecuzione di una singola istruzione può invece richiedere più cicli.

## 4.1 Esempio di microarchitettura

Non esistono principi generali validi per ogni progetto: ogni ISA (Instruction Set Architecture) rappresenta un caso a sé. Verrà analizzato un sottoinsieme della Java Virtual Machine che gestisce solo numeri interi, denominato IJVM.
Iniziamo descrivendo la microarchitettura sopra la quale implementeremo IJVM:

- **Il Microprogramma come Interprete:** conterrà un microprogramma il cui compito sarà quello di prelevare, decodificare ed eseguire le istruzioni IJVM. A differenza dell'interprete JVM standard (scritto in C per portabilità), questo microprogramma è progettato per pilotare in modo efficiente i singoli gate hardware (porte logiche).
- **Modello di Progettazione Logica:** Il design della microarchitettura può essere approcciato come un problema di programmazione, in cui ogni istruzione del livello ISA è vista come una funzione chiamata da un programma principale.
- **Il Ciclo di Controllo Principale:** Il motore del sistema è un ciclo infinito, spesso chiamato ciclo fetch-decode-execute. Questo ciclo determina quale funzione invocare, la esegue e ricomincia da capo.
- **Stato del Calcolatore:** Il microprogramma opera su un insieme di variabili che costituiscono lo "stato" della macchina. Un esempio fondamentale di variabile di stato è il Program Counter (PC), che indica la locazione di memoria contenente la successiva istruzione ISA da eseguire.
- **Struttura delle Istruzioni ISA:** Le istruzioni IJVM interpretate dalla microarchitettura sono composte da un opcode (codice operativo), che identifica l'azione (come una ADD o un salto), e talvolta da un operando, che specifica i dati su cui agire (ad esempio, quale variabile locale utilizzare).
- **Controllo per Ciclo di Clock:** Ogni singola microistruzione all'interno del microprogramma ha il compito di assumere il controllo del percorso dati per la durata di un unico ciclo di clock.

### 4.1.1 Percorso Dati

Il percorso dati contiene una serie di registra a 32 bit, identificati da nomi simbolici (come PC, SP, MDR, MAR, H). È fondamentale notare che questi registri sono accessibili solo a livello di microarchitettura dal microprogramma.

- **Bus B:** La maggior parte dei registri può immettere il proprio contenuto sul bus B, che alimenta l'ingresso destro dell'ALU.
- **Registro H (Holding):** Questo registro alimenta l'ingresso sinistro (A) dell'ALU. H può essere caricato solo dal bus C e serve a mantenere il primo operando mentre il secondo viene prelevato dal bus B.
- **Bus C:** Riceve il risultato dall'ALU (dopo il passaggio nello shifter) e lo distribuisce ai registri; il valore sul bus C può essere scritto in uno o più registri contemporaneamente al termine del ciclo.

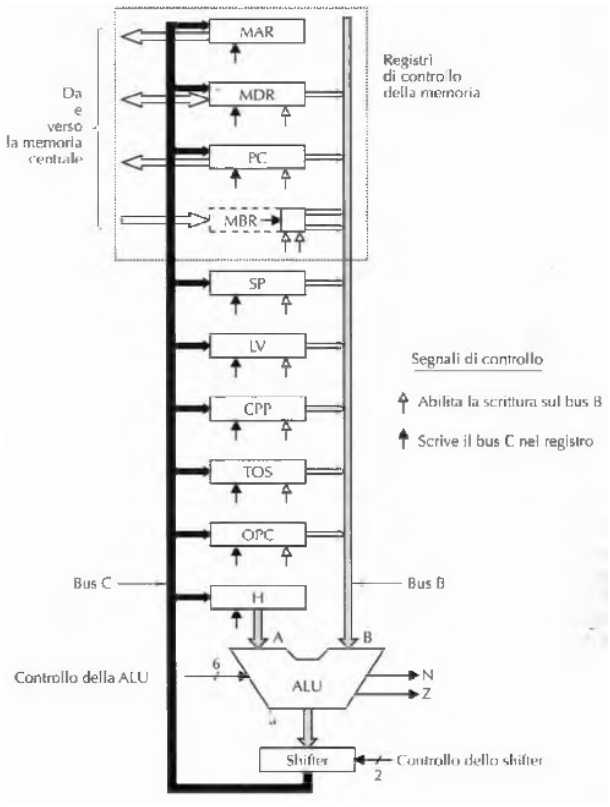



Figura 51: Percorso dati della microarchitettura utilizzata per l'esempio di questo capitolo

L'ALU esegue operazioni aritmetiche e logiche regolate da 6 linee di controllo:

- **F0, F1:** Determinano il tipo di operazione.
- **ENA, ENB:** Abilitano individualmente gli ingressi A e B.
- **INVA:** Inverte l'ingresso sinistro.
- **INC:** Forza un riporto nel bit meno significativo, permettendo operazioni come B + 1.

All'uscita dell'ALU si trova lo **shifter**, che può traslare il valore prima di immetterlo sul bus C. Supporta due operazioni principali: SLL8 (Shift Left Logical), che sposta a sinistra di 1 byte (8 bit) inserendo zeri, e SRA1 (Shift Right Arithmetic), che sposta a destra di 1 bit preservando il bit di segno.

| $F_0$ | $F_1$ | ENA | ENB | INVA | INC | Function |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 0 | A |
| 0 | 1 | 0 | 1 | 0 | 0 | B |
| 0 | 1 | 1 | 0 | 1 | 0 | $\bar{A}$ |
| 1 | 0 | 1 | 1 | 0 | 0 | $\bar{B}$ |
| 1 | 1 | 1 | 1 | 0 | 0 | A + B |
| 1 | 1 | 1 | 1 | 0 | 1 | A + B + 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | A + 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | B + 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | B – A |
| 1 | 1 | 0 | 1 | 1 | 0 | B – 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | –A |
| 0 | 0 | 1 | 1 | 0 | 0 | A AND B |
| 0 | 1 | 1 | 1 | 0 | 0 | A OR B |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | –1 |

Figura 52: Segnali di Controllo in ingrasso alla ALU e corrispondenti funzioni.

Un aspetto cruciale è la capacità di leggere e scrivere lo stesso registro in un unico ciclo (es. $SP = SP+1$). Questo è possibile perché le operazioni avvengono in momenti diversi del ciclo di clock:

1. $\Delta w$**:** All'inizio del ciclo (fronte di discesa), si impostano i segnali di controllo per guidare il percorso dati.
2. $\Delta x$**:** Il registro selezionato carica il suo valore sul bus B.
3. $\Delta y$**:** L'ALU e lo shifter operano sui dati stabili.
4. $\Delta z$**:** Il risultato si propaga lungo il bus C verso i registri in cui possono essere caricati in corrispondenza del fronte di salita dell'impulso successivo.
5. **Fine Ciclo:** Sul fronte di salita del clock, i risultati vengono salvati nei registri di destinazione. Grazie ai ritardi di propagazione finiti, il nuovo valore non raggiunge l'ALU in tempo per alterare il calcolo dello stesso ciclo.

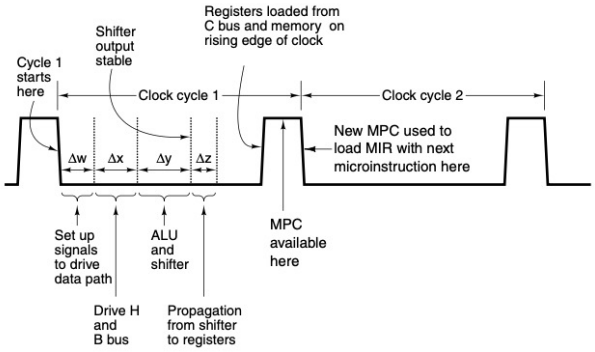



Figura 53: Temporizzazione di un ciclo di clock del percorso dati

La microarchitettura comunica con la memoria attraverso due porte distinte:

- **Porta a 32 bit (Parole):** Controllata dai registri MAR (Memory Address Register) e MDR (Memory Data Register). MAR contiene indirizzi espressi in parole (0, 1, 2...), ma viene mappato sul bus indirizzi fisico in modo che l'indirizzo 1 corrisponda al byte 4, il 2 al byte 8, e così via.

- **Porta a 8 bit (Byte):** Controllata dal registro PC, che legge un singolo byte nel registro MBR. Questa porta è di sola lettura e viene usata per prelevare il flusso di istruzioni ISA.
- **MBR (Memory Buffer Register):** Può essere immesso sul bus B in due modalità: MBRU (unsigned), dove il byte è esteso con 24 zeri, o MBR (signed), dove viene effettuata l'estensione del segno.

Le operazioni di memoria seguono una temporizzazione precisa per garantire la stabilità dei dati:

- **Latenza di un ciclo:** Si assume che la memoria impieghi un solo ciclo per operare (ipotizzando un tasso di successo della cache L1 del 100
- **Disponibilità dei dati:** Se un'operazione di lettura viene avviata alla fine del ciclo k, i dati saranno pronti nel registro (MBR o MDR) alla fine del ciclo k+1. Ciò significa che non possono essere utilizzati nel ciclo immediatamente successivo, ma solo nel ciclo k+2 o più tardi.
- **Contemporaneità:** Entrambe le porte (32 bit e 8 bit) possono operare simultaneamente. Tuttavia, tentare di leggere e scrivere lo stesso byte nello stesso istante produce risultati indefiniti.

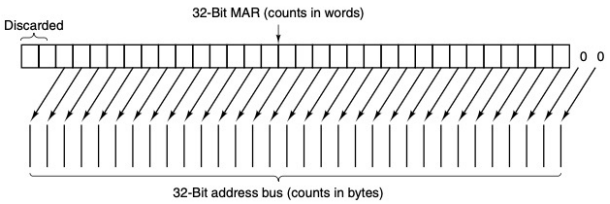



Figura 54: Corrispondenza tra i bit di MAR e i bit di indirizzo

### 4.1.2 Microistruzioni

Per gestire le operazioni del percorso dati, sono necessari 29 segnali di controllo. Questi segnali sono suddivisi in cinque gruppi funzionali che operano durante un unico ciclo di clock:

- **9 segnali per il Bus C:** Controllano la scrittura dei dati dal bus C nei registri (come PC, SP, MDR)
- **9 segnali per il Bus B:** Controllano quale registro deve immettere il proprio contenuto sul bus B per alimentare l'ingresso destro della ALU.
- **8 segnali per ALU e Shifter:** Determinano la specifica operazione aritmetico-logica e l'eventuale scorrimento (SLL8 o SRA1) del risultato.
- **2 segnali per MAR/MDR:** Indicano operazioni di lettura o scrittura della memoria tramite questi registri.
- **2 segnali per MAR/MDR:** Indica il prelievo (fetch) di un byte dalla memoria verso il registro MBR.