

10. Il rapporto tra le prestazioni del 360 model 75 e del 360 model 30 era di 50 volte, mentre il ciclo di clock era solo 5 volte più veloce. Come si spiega questa differenza?
11. Nella Figura 1.5 e nella Figura 1.6 sono mostrati due elementari progetti di un sistema. Si descriva come potrebbero avvenire gli input e output in ciascuno di loro. Quale dei due può potenzialmente fornire migliori prestazioni generali del sistema?
12. Si supponga che ogni giorno ciascuno dei 300 milioni di cittadini americani consumi due confezioni di un certo articolo a cui è associata un'etichetta RFID. Quante etichette RFID devono essere prodotte annualmente per soddisfare la domanda? Qual è il costo totale delle etichette, se ognuna di loro costa un penny? Considerando l'ammontare del Prodotto Interno Lordo questa quantità di denaro potrebbe oppure no essere un ostacolo all'utilizzo di etichette RFID per ogni confezione messa in vendita?
13. Si citino tre elettrodomestici che potrebbero funzionare mediante una CPU integrata.
14. A un certo momento della storia un transistor su un microprocessore arrivò ad avere un diametro di 1 dm. Secondo la legge di Moore, che dimensioni avrebbe assunto nell'anno successivo?
15. È stato mostrato che la legge di Moore non si applica solo alla densità dei semiconduttori, ma predice anche l'aumento dello spazio occupato e la diminuzione del tempo d'esecuzione di una simulazione. Si mostri che per una simulazione di meccanica dei fluidi che impiega 4 ore per essere eseguita su una macchina di oggi sarà necessaria 1 ora di esecuzione su una macchina costruita tra 3 anni e solo 15 minuti su una macchina costruita tra 6 anni. Si mostri poi che una simulazione molto onerosa con un tempo di esecuzione stimato di 5 anni terminerebbe prima se aspettassimo 3 anni prima di iniziare l'esecuzione.
16. Nel 1959, un IBM 7090 poteva eseguire circa 500.000 istruzioni al secondo, aveva una memoria di 32.768 parole di 36 bit e costava 3 milioni di dollari. Si confronti questa macchina con una macchina recente e si determini quanto migliore è il computer attuale moltiplicando il rapporto tra le dimensioni della memoria con quello tra le velocità e dividendo questa quantità per il rapporto tra i prezzi. Vediamo ora ciò che lo stesso guadagno avrebbe portato nell'aviazione nello stesso periodo di tempo. Il Boeing 707 è stato consegnato alle compagnie aeree, in notevoli quantità, nel 1959. La sua velocità era di 950 Km/h, poteva inizialmente trasportare 180 passeggeri e costava 4 milioni di dollari. Quali sarebbero ora la velocità, la capacità e il costo di un aeromobile se avesse ottenuto gli stessi guadagni che ha ottenuto un computer? Si indichino le assunzioni fatte su velocità, dimensione della memoria e prezzo.
17. L'evoluzione del settore dei computer è spesso ciclica. In origine, gli insiemi delle istruzioni erano cablati, poi si è passati ai microprogrammata, quindi, con l'avvento delle macchine RISC si è ripreso a cablarli. In origine, il calcolo era centralizzato su mainframe di grandi dimensioni. Si citino due sviluppi allo scopo di mostrare che anche in questo caso si è avuto un comportamento ciclico.
18. La diatriba legale su chi ha inventato il computer fu risolta nell'aprile del 1973 dal giudice Earl Larson; egli si occupò della causa legale di violazione di brevetto intentata dalla Sperry Rand Corporation, società che aveva comprato i brevetti dell'elaboratore ENIAC. La posizione della Sperry Rand era che chiunque avesse prodotto un computer avrebbe dovuto pagarle i diritti, in quanto proprietaria dei brevetti chiave. Il caso arrivò in giudizio nel giugno 1971 e furono depositati più di 30.000 documenti. La trascrizione degli atti superò le 20.000 pagine. Utilizzando le vaste informazioni disponibili su Internet si studi il caso più approfonditamente e si scriva una relazione sui suoi aspetti tecnici. Che cosa sosteneva il brevetto di Eckert e Mauchley e perché il giudice ritenne che il loro sistema era basato sul precedente lavoro di Atanasoff?
19. Si scelgano le tre persone che si ritiene abbiano avuto la maggiore influenza nella creazione dell'hardware moderno; si scriva quindi una breve relazione che descriva il loro contributo e motivi la scelta effettuata.
20. Si scelgano le tre persone che si ritiene abbiano avuto la maggiore influenza nella creazione del software di sistema moderno; si scriva quindi una breve relazione che descriva il loro contributo e motivi la scelta effettuata.
21. Si scelgano le tre persone che si ritiene abbiano avuto la maggiore influenza nella creazione dei più trafficati siti web moderni; si scriva quindi una breve relazione che descriva il loro contributo e motivi la scelta effettuata.



## Organizzazione dei sistemi di calcolo

Un calcolatore digitale è un sistema in cui processori, memorie e dispositivi periferici sono connessi tra loro. Questo capitolo fornisce un'introduzione a questi tre componenti e alle loro interconnessioni, servendo da base per l'analisi dettagliata dei singoli livelli che verrà data nei cinque capitoli successivi. Dato che i processori, le memorie e le periferiche sono concetti chiave che ritorneranno durante la trattazione di ciascun livello, iniziamo lo studio dell'architettura degli elaboratori esaminandoli uno alla volta.

### 2.1 Processori

La Figura 2.1 mostra l'organizzazione di un semplice calcolatore *bus-oriented*. La CPU (*Central Processing Unit*, "unità centrale di calcolo") è il "cervello" del computer e la sua funzione è quella di eseguire i programmi contenuti nella memoria principale prelevando le loro istruzioni, esaminandole ed eseguendole una dopo l'altra. I componenti sono connessi fra loro mediante un bus, cioè un insieme di cavi paralleli sui quali vengono trasmessi indirizzi, dati e segnali di controllo. I bus possono essere esterni alla CPU, per connetterla alla memoria e ai dispositivi di I/O, oppure interni, come vedremo tra poco.

La CPU è composta da parti distinte, tra cui l'unità di controllo e l'unità aritmetico-logica. La prima si occupa di prelevare le istruzioni dalla memoria principale e di determinarne il tipo, mentre la seconda esegue le operazioni, come l'addizione e l'AND, necessarie per portare a termine l'esecuzione delle istruzioni.

La CPU contiene anche una piccola memoria ad alta velocità, utilizzata per memorizzare i risultati temporanei e alcune informazioni di controllo. Questa memoria è costituita da un certo numero di registri, ciascuno dei quali ha una funzione e una dimensione predefinite. Ognuno di loro può contenere un numero, il cui valore può variare fino a un massimo che dipende dalla dimensione del registro, che di solito è uguale per tutti i registri. Dato che sono interni alla CPU possono essere letti e scritti a velocità elevate.

Il registro più importante è il **contatore d'istruzioni**, o **Program Counter (PC)**, che punta alla successiva istruzione che dovrà essere prelevata per l'esecuzione (il termine "contatore d'istruzioni", pur essendo universalmente accettato, è in qualche modo ambiguo, dato che tale registro non effettua alcun *conteggio*). Un altro registro particolarmente importante è il **registro istruzione corrente**, o *Instruction Register (IR)*, contenente l'istruzione che si trova in fase di esecuzione. La maggior parte dei calcolatori possiede molti altri registri, alcuni di uso generale, altri dedicati a un uso specifico, altri ancora utilizzati dal sistema operativo per controllare il computer.

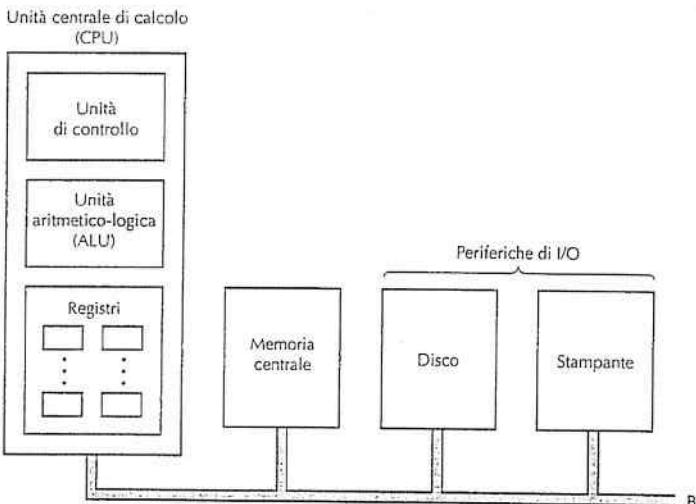


Figura 2.1 Organizzazione di un semplice calcolatore con una CPU e due periferiche di I/O.

### 2.1.1 Organizzazione della CPU

La Figura 2.2 mostra com'è organizzata internamente una parte, chiamata **percorso dati** (*data path*), di una tipica CPU di von Neumann; essa è composta dai registri (generalmente da 1 a 32), dalla ALU (*Arithmetic Logic Unit*, "unità aritmetico-logica") e da alcuni bus che connettono fra loro le diverse parti. I registri alimentano due registri di input della ALU (indicati nella figura con le lettere A e B) che mantengono i dati d'ingresso della ALU mentre questa è occupata nell'esecuzione di alcune computazioni. Il percorso dati riveste una grande importanza in tutte le macchine e nel corso del libro lo tratteremo in modo approfondito.

La ALU esegue alcune semplici operazioni sui suoi input, come addizioni e sottrazioni, e genera un risultato che viene memorizzato in un suo apposito registro di output. Questo valore può essere successivamente immagazzinato in uno dei registri della CPU che, volendo, può essere copiato in memoria in un secondo momento. Non tutti i tipi di

progettazione prevedono la presenza dei registri A, B e di quello di output. Nell'esempio è illustrata l'addizione, ma l'ALU può eseguire anche altre operazioni.

La maggior parte delle istruzioni può essere divisa in due categorie principali: le istruzioni registro-memoria e quelle registro-registro. Le istruzioni registro-memoria permettono di prelevare parole di memoria per portarle all'interno dei registri, dove sono utilizzabili, per esempio, come input della ALU per effettuare istruzioni successive. (Le "parole" sono le unità di dati che vengono spostate tra la memoria e i registri; una parola potrebbe essere un intero. Più avanti nel capitolo analizzeremo l'organizzazione della memoria.) Altre istruzioni registro-memoria permettono invece di copiare i valori dei registri nella memoria.

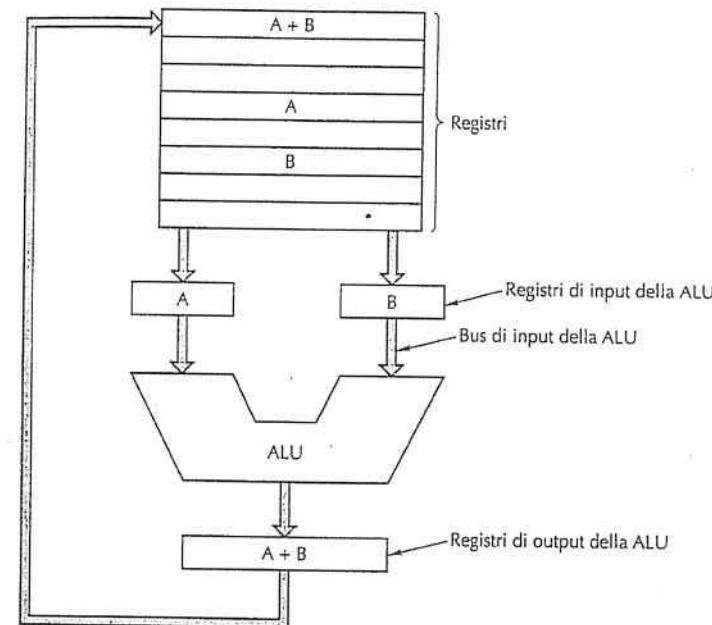


Figura 2.2 Percorso dati di una tipica macchina di von Neumann.

L'altra classe d'istruzioni è quella registro-registro. Una tipica istruzione di questo tipo preleva due operandi dai registri, li porta all'interno dei registri di input della ALU, esegue su di loro una qualche operazione (per esempio l'addizione o l'AND) 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** e rappresenta il cuore della maggior parte delle CPU. Con buona approssimazione si può dire che definisce che cosa sia in grado di fare una macchina. I moderni computer hanno

più ALU che operano in parallelo, specializzate per funzioni diverse. Più veloce è il ciclo del percorso dati, maggiore risulta la velocità del calcolatore.

### 2.1.2 Esecuzione dell'istruzione

La CPU esegue ogni istruzione compiendo una serie di piccoli passi che, in linea generale, possono essere descritti nel seguente modo:

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. tornare al punto 1 per iniziare l'esecuzione dell'istruzione successiva.

Spesso ci si riferisce a questa sequenza di passi con il termine di **ciclo esecutivo delle istruzioni**, o **ciclo di prelievo-decodifica-esecuzione**. Essa ha un'importanza centrale nel funzionamento di qualsiasi calcolatore.

La precedente descrizione del funzionamento della CPU assomiglia osservandola attentamente a un programma scritto in italiano. La Figura 2.3 mostra come sia possibile formalizzare questo programma in termini di un metodo (cioè una procedura Java, che prende il nome di *interprete*). La macchina che viene interpretata possiede due registri visibili all'utente: il contatore di programma (PC), per tener traccia dell'indirizzo dell'istruzione successiva da prelevare, e l'accumulatore (AC), per contenere il risultato delle operazioni aritmetiche. Essa possiede anche alcuni registri interni per mantenere l'istruzione corrente (*instr*), il tipo d'istruzione corrente (*instr\_type*), l'indirizzo dell'operando dell'istruzione (*data\_loc*) e l'operando stesso (*data*). Si assume che le istruzioni contengano un unico indirizzo di memoria e che nella locazione di memoria corrispondente sia memorizzato l'operando, che può essere, per esempio, un dato da sommare nell'accumulatore.

Il fatto che sia possibile scrivere un programma che simula il comportamento della CPU mostra che esso non deve per forza essere eseguito da una scatola piena di componenti elettronici, cioè dalla CPU fisica. Un programma può infatti essere portato a termine anche da un altro programma, che preleva, esamina ed esegue le proprie istruzioni; come abbiamo già detto nel corso del Capitolo 1, quest'ultimo programma viene chiamato **interprete**.

Questa equivalenza tra i processori hardware e gli interpreti ha alcune importanti implicazioni nell'organizzazione e nel progetto dei sistemi di calcolatori. Un gruppo di progettisti, dopo aver specificato il linguaggio macchina, *L*, per un nuovo calcolatore, può scegliere se realizzare un processore hardware capace di eseguire direttamente i programmi in *L* oppure scrivere un programma che interpreti tali programmi. Se si sceglie di scrivere un interprete occorre anche prevedere i componenti hardware che permettano di eseguirlo. È inoltre possibile avere delle costruzioni ibride, nelle quali alcune esecuzioni vengono svolte dall'hardware, mentre altre dall'interpretazione software.

```

public class Interp
{
    static int PC;                                // il program counter contiene l'indirizzo dell'istruzione successiva
    static int AC;                                // l'accumulatore, un registro per i calcoli aritmetici
    static int instr;                             // un registro che contiene l'istruzione corrente
    static int instr-type;                         // il tipo d'istruzione (opcode)
    static int data-loc;                           // l'indirizzo del dato, o -1 se non c'è
    static int data;                               // contiene l'operando corrente
    static boolean run-bit = true;                 // un bit che può essere impostato a 0 per arrestare la macchina

    public static void interpret(int memory[], int starting-address) {
        // Questa procedura interpreta programmi per una semplice macchina le cui istruzioni hanno
        // un solo operando in memoria. La macchina ha un registro AC (accumulatore), utilizzato per
        // i calcoli aritmetici. L'istruzione ADD, per esempio, somma un intero memorizzato in memoria al registro AC.
        // L'interprete continua finché il bit di esecuzione non viene impostato a 0 dall'istruzione HALT.
        // Lo stato di un processo eseguito su questa macchina è costituito dalla memoria, dal
        // contatore di programma, dal bit di esecuzione e dal registro AC. I parametri di input consistono
        // nell'immagine della memoria e nell'indirizzo di partenza.

        PC = starting-address;
        while(run-bit) {
            instr = memory[PC];                      // preleva l'istruzione successiva e la memorizza in instr
            PC = PC + 1;                            // incrementa il program counter
            instr-type = get-instr-type(instr);       // determina il tipo d'istruzione
            data-loc = find-data(instr, instr-type);  // trova la posizione del dato (-1 se non c'è)
            if(data-loc >= 0) {
                data = memory[data-loc];           // se "data-loc" vale -1, allora non c'è nessun operando
                execute(instr-type, data);          // preleva il dato
            }                                         // esegue l'istruzione
        }
    }

    private static int get-instr-type(int addr) { ... }
    private static int find-data(int instr, int type) { ... }
    private static void execute(int type, int data) { ... }
}

```

Figura 2.3 Interpret Java per un semplice calcolatore.

Dato che un interprete scomponete le istruzioni della propria macchina obiettivo in piccoli passi, il calcolatore sul quale viene eseguito può essere molto più semplice e meno costoso rispetto a quanto lo sarebbe un processore hardware appositamente progettato per la macchina obiettivo. Tale convenienza risulta ancor più significativa se la macchina obiettivo possiede molte istruzioni e se queste sono particolarmente complesse e caratterizzate da molte opzioni. Il vantaggio deriva essenzialmente dal fatto che si sostituisce parte dell'hardware con il software (l'interprete), molto meno costoso da riprodurre.

I primi calcolatori avevano un piccolo insieme di semplici istruzioni, ma la richiesta di calcolatori con prestazioni più elevate portò, fra le altre cose, all'introduzione d'istruzioni più potenti. In una fase iniziale si notò che spesso l'uso d'istruzioni più complesse aumentava la velocità di esecuzione dei programmi, anche se le singole istruzioni richiedevano più tempo per essere eseguite. Alcuni esempi di queste istruzioni più complesse comprendono quelle in virgola mobile e quelle per l'accesso diretto agli elementi di un

array. In alcuni casi si osservò semplicemente che due istruzioni apparivano spesso consecutive e che quindi una singola istruzione avrebbe potuto svolgere il lavoro di entrambe.

Le istruzioni più complesse erano preferibili in quanto le singole esecuzioni delle operazioni elementari potevano essere sovrapposte oppure eseguite in parallelo utilizzando parti diverse dell'hardware. Dato che i calcolatori ad alte prestazioni giustificavano pienamente il costo di questo hardware aggiuntivo, essi arrivarono ad avere un numero molto maggiore d'istruzioni rispetto alle macchine di costo inferiore. Tuttavia, a causa della crescita del costo dello sviluppo del software e per via dei requisiti di compatibilità, sorse la necessità d'introdurre istruzioni complesse anche nelle macchine più economiche, per le quali il costo era più importante della velocità.

Alla fine degli anni '50 IBM (la società di computer più potente dell'epoca) comprese che mantenere una singola famiglia di macchine, tutte capaci di eseguire le stesse istruzioni, risultava molto vantaggioso non solo per IBM stessa, ma anche per i suoi clienti. IBM introdusse il termine architettura per descrivere questo livello di compatibilità. Una nuova famiglia di calcolatori avrebbe dovuto avere un'unica architettura, ma varie e distinte implementazioni che, pur distinguendosi nel prezzo e nelle prestazioni, sarebbero state in grado di eseguire lo stesso programma. Com'è possibile tuttavia costruire una macchina economica capace di eseguire tutte le complicate istruzioni di macchine costose e ad alte prestazioni?

La risposta risiede nell'interpretazione. Questa tecnica, suggerita inizialmente da Maurice Wilkes (1951), ha permesso di progettare calcolatori semplici ed economici, in grado tuttavia di eseguire un gran numero d'istruzioni. Il risultato dell'interpretazione fu la nascita dell'architettura IBM System/360, una famiglia di calcolatori fra loro compatibili e che variavano di quasi due ordini di grandezza sia nel prezzo sia nelle prestazioni. Soltanto nei modelli più costosi si ricorse a un'implementazione diretta dell'hardware (cioè senza interpretazione).

I calcolatori semplici che utilizzano istruzioni interpretate presentano anche altri vantaggi, tra i quali:

1. la capacità di correggere agevolmente le istruzioni implementate in modo errato, o addirittura di compensare le mancanze progettuali dell'hardware di base;
2. la possibilità di aggiungere nuove istruzioni a un costo minimo, anche dopo aver distribuito la macchina;
3. un progetto strutturato che permetta di sviluppare, testare e documentare efficientemente le istruzioni complesse.

Negli anni '70, quando il mercato dei computer esplose in modo impressionante e la potenza di calcolo cominciò a crescere rapidamente, la richiesta di calcolatori a basso costo favorì la progettazione di macchine che usavano interpreti. La possibilità di adattare l'hardware e l'interprete per un determinato insieme d'istruzioni apparve come una modalità di progettazione dei processori altamente conveniente dal punto di vista dei costi. Dato che la tecnologia dei semiconduttori avanzava rapidamente, i vantaggi sui costi acquisirono un peso maggiore rispetto alla possibilità di avere prestazioni più elevate; le architetture basate sugli interpreti divennero così il modo convenzionale di

progettare calcolatori. Quasi tutti i computer costruiti negli anni '70, dai minicomputer fino ai mainframe, si basavano infatti sull'interpretazione.

Nei tardi anni '70 divenne molto diffuso l'uso di processori semplici che eseguivano interpreti, fatta eccezione per i modelli più costosi e dalle prestazioni più elevate, come il Cray-1 e la serie Control Data Cyber. L'uso di un interprete eliminò i problemi di costo che avevano limitato l'uso d'istruzioni complesse e i progettisti iniziarono a esplorare istruzioni ancor più complicate, soprattutto nel modo in cui venivano specificati gli operandi da utilizzare.

Questa tendenza raggiunse il suo apice con il calcolatore VAX della Digital Equipment Corporation, che aveva varie centinaia d'istruzioni e più di 200 modi diversi di specificare gli operandi da utilizzare in ciascuna istruzione. Sfortunatamente l'architettura VAX fu concepita fin dall'inizio per essere implementata attraverso un interprete e con poca attenzione alla realizzazione di un modello ad alte prestazioni. Questo approccio portò all'inclusione di un grandissimo numero d'istruzioni la cui importanza era marginale e che era difficile poter eseguire direttamente. Questo errore risultò fatale alla macchina VAX e, in ultima analisi, alla DEC stessa (nel 1998 Compaq acquistò DEC e nel 2001 Hewlett-Packard acquistò Compaq).

Se i primi microprocessori a 8-bit erano macchine con un insieme d'istruzioni molto semplice, alla fine degli anni '70 ogni microprocessore era passato invece a una progettazione basata sull'interpretazione. Durante questo periodo una delle maggiori sfide che si presentavano ai progettisti dei microprocessori era quella di gestire la crescente complessità resa possibile dai circuiti integrati. Uno dei maggiori vantaggi dell'approccio basato sull'interpretazione era la possibilità di progettare un processore semplice, la cui complessità era in gran parte confinata nell'interprete contenuto in memoria; in questo modo era possibile portare la complessità progettuale dall'hardware al software.

Il successo del Motorola 68000, che disponeva di un ampio insieme d'istruzioni interpretate, e il contemporaneo fallimento dello Zilog Z8000 (che aveva anch'esso un grande insieme d'istruzioni, ma senza interprete) mostrò quali vantaggi offriva un interprete nel mettere velocemente sul mercato un nuovo microprocessore. Questo successo fu ancora più sorprendente se si considera il vantaggio che all'epoca aveva la società Zilog (il processore Z80, predecessore dello Z8000, era stato infatti molto più diffuso del 6800, predecessore del 68000). Ovviamente vi furono anche altre concuse, non ultima la lunga esperienza di Motorola come produttore di processori e quella di Exxon (la società che controllava la Zilog) come società petrolifera, e non certo come produttrice di processori.

Un altro fattore che all'epoca favorì l'interpretazione fu l'esistenza di veloci memorie di sola lettura, chiamate **memorie di controllo** (*control store*), usate per memorizzare l'interprete. Supponiamo che una normale istruzione interpretata richiedesse 10 istruzioni dell'interprete, chiamate **microistruzioni**, ciascuna da 100 ns, oltre a due riferimenti alla memoria centrale da 500 ns l'uno. L'esecuzione totale richiedeva quindi 2000 ns, un valore solo due volte peggiore del miglior risultato raggiungibile tramite l'esecuzione diretta; senza l'utilizzo della memoria di controllo, le istruzioni avrebbero invece impiegato 6000 ns. È evidente che un fattore di penalizzazione di sei volte è molto più difficile da accettare rispetto a un fattore due.

### 2.1.3 RISC contro CISC

Durante i tardi anni '70 gli interpreti permisero di sperimentare istruzioni molto complesse; il tentativo inseguito dai progettisti era quello di ridurre il "gap semantico" che divideva ciò che erano in grado di fare le macchine da quello che invece richiedevano i linguaggi di programmazione ad alto livello. Quasi mai si pensava di progettare macchine più semplici, esattamente come oggi non sono molti i ricercatori che si impegnano nel progettare fogli di calcolo, reti, server web e così via. meno potenti (e forse ciò è un peccato).

Ciononostante un gruppo guidato da John Cocke di IBM andò contro questa tendenza e provò a implementare alcune delle idee di Seymour Cray all'interno di un minicomputer ad alte prestazioni. Il risultato di questo lavoro fu un minicomputer sperimentale, chiamato 801<sup>1</sup>. Anche se IBM non mise mai in commercio questa macchina e i risultati furono pubblicati soltanto alcuni anni dopo (Radin, 1982), trapelarono alcune voci riguardanti il progetto e altri ricercatori iniziarono a studiare architetture simili. Nel 1980, a Berkley, un gruppo guidato da David Patterson e Carlo Séquin cominciò la progettazione di chip VLSI per CPU che non utilizzavano l'interpretazione (Patterson, 1985; Patterson e Séquin, 1982). Per indicare questo progetto essi coniarono l'acronimo RISC e chiamarono la loro CPU RISC I, che fu seguita a breve dal modello RISC II. Poco dopo, nel 1981, a Stanford, sull'altra sponda della baia di San Francisco, John Hennessy progettò e costruì un chip, chiamato MIPS, leggermente differente sotto alcuni aspetti (Hennessy, 1984). Entrambi i chip evolsero in due importanti prodotti commerciali, rispettivamente SPARC e MIPS.

Questi nuovi processori erano significativamente diversi rispetto a quelli in commercio all'epoca. Dato che non dovevano essere retrocompatibili con alcun prodotto esistente, i loro progettisti furono liberi di scegliere nuovi insiemi d'istruzioni che massimizzassero le prestazioni generali del sistema. Mentre l'enfasi iniziale fu posta su istruzioni semplici la cui esecuzione potesse essere completata velocemente, presto si capì che la chiave per ottenere buone prestazioni consisteva nel progettare istruzioni che potessero essere emesse (iniziate) velocemente. Il tempo impiegato effettivamente per l'esecuzione di un'istruzione aveva un'importanza minore rispetto a quante se ne potevano iniziare in un secondo.

Quando vennero progettati questi semplici processori, la caratteristica che colpì maggiormente l'attenzione di tutti fu che il numero d'istruzioni disponibili era relativamente basso, generalmente attorno a 50. Questo valore era nettamente inferiore alle 200-300 dei computer convenzionali come il VAX e i grandi mainframe IBM. L'acronimo RISC significa infatti *computer con un insieme ridotto d'istruzioni (Reduced Instruction Set Computer)*, scelto in contrapposizione a CISC, che significa *computer con un insieme d'istruzioni complesso (Complex Instruction Set Computer)*: un sottile riferimento al VAX, che aveva dominato i dipartimenti universitari di quegli anni. L'uso del nome si è radicato ed è utilizzato ancora oggi, sebbene attualmente siano in pochi a ritenere che la dimensione dell'insieme d'istruzioni sia un problema importante.

<sup>1</sup> Pare che questo nome derivi dal numero dell'edificio del centro IBM in cui si svolse il progetto (N.d.R.).

In sintesi si può affermare che si scatenò una guerra di religione che vedeva i sostenitori di RISC attaccare l'ordine prestabilito (VAX, Intel e i grandi mainframe IBM). Essi sostenevano che il miglior modo per progettare un calcolatore fosse quello di avere poche istruzioni semplici che potevano essere eseguite in un ciclo del percorso dati mostrato nella Figura 2.2, per esempio leggendo due registri, combinandoli in qualche modo (come sommandoli o calcolandone il prodotto logico) e memorizzando infine il risultato all'interno di un registro. La loro tesi era che una macchina RISC prevaleva nei confronti di una CISC perché, anche se impiegava quattro o cinque istruzioni per fare ciò che una macchina CISC realizzava con una sola, le sue istruzioni erano 10 volte più veloci (in quanto non interpretate). Vale la pena sottolineare che a quel tempo la velocità delle memorie centrali aveva raggiunto quella delle memorie di controllo di sola lettura, rendendo ancora più marcata la penalizzazione dovuta all'interpretazione; ciò contribuì a favorire fortemente le macchine RISC.

Si potrebbe pensare che, grazie ai vantaggi offerti dalla tecnologia RISC, queste macchine (come la Sun UltraSPARC) abbiano fatto scomparire dal mercato le macchine CISC (come gli Intel Pentium), e invece ciò non è avvenuto: perché?

Primo fra tutti va considerato il problema della retrocompatibilità e con esso i miliardi di dollari investiti dalle società per produrre il software per le macchine Intel. Secondo, Intel è riuscita sorprendentemente a impiegare le stesse idee anche all'interno dell'architettura CISC. A partire dal processore 486, le CPU di Intel contengono un nucleo di tipo RISC che esegue le istruzioni più semplici (e spesso più comuni) in un unico ciclo del percorso dati, mentre interpreta le istruzioni più complesse secondo la classica modalità CISC. Il risultato finale è che le istruzioni più comuni sono veloci, mentre quelle meno comuni sono più lente.

Anche se questo approccio non è altrettanto veloce quanto una pura architettura RISC, esso fornisce prestazioni generali competitive, permettendo allo stesso tempo di eseguire senza alcuna modifica il software progettato precedentemente.

### 2.1.4 Principi di progettazione dei calcolatori moderni

Ora che sono passati più di due decenni dall'introduzione delle prime macchine RISC, alcuni principi di progettazione sono stati accettati come un buon modo di progettare un calcolatore, tenendo in considerazione lo stato attuale della tecnologia dell'hardware. Se dovesse verificarsi un importante cambiamento tecnologico (per esempio un nuovo processo di produzione tale da rendere il ciclo della memoria 10 volte più veloce di quello della CPU), molte delle scommesse finora fatte risulterebbero perdenti. I progettisti di calcolatori devono quindi tenere sempre d'occhio i cambiamenti tecnologici, in quanto possono modificare l'equilibrio fra i vari componenti.

Detto questo, esiste un insieme di principi di progettazione, talvolta chiamati *principi di progettazione RISC*, che i progettisti delle CPU cercano di seguire il più possibile. Anche se vincoli esterni, come i requisiti di retrocompatibilità con alcune architetture esistenti, richiedono di tanto in tanto alcuni compromessi, questi principi sono comunque degli obiettivi che i progettisti si sforzano di raggiungere. Di seguito ne analizzeremo i principali.

### Tutte le istruzioni sono eseguite direttamente dall'hardware

Tutte le comuni istruzioni sono eseguite direttamente dall'hardware, e non sono interpretate mediante microistruzioni. L'eliminazione di un livello di interpretazione garantisce velocità più alte per la maggior parte delle istruzioni. Per i calcolatori che implementano insiemi d'istruzioni CISC, quelle più complesse devono essere divise in varie parti che possono essere eseguite come una sequenza di microistruzioni. Questo passo aggiuntivo rallenta la macchina, ma può tuttavia essere accettabile per le istruzioni utilizzate con minor frequenza.

### Massimizzare la frequenza di emissione delle istruzioni

I calcolatori moderni ricorrono a molti trucchi per massimizzare le loro prestazioni, il principale dei quali consiste nel cercare di iniziare a eseguire più istruzioni al secondo. Dopo tutto, se si riescono a emettere 500 milioni d'istruzioni al secondo, si ottiene un processore da 500 MIPS, indipendentemente da quanto tempo impieghino quelle istruzioni per essere completate<sup>2</sup>. Questo principio suggerisce che il parallelismo può giocare un ruolo importante nelle prestazioni; infatti è possibile emettere un gran numero di diverse istruzioni in un breve intervallo di tempo solo se si riescono a eseguire più istruzioni allo stesso tempo.

Anche se le istruzioni vengono sempre incontrate nell'ordine in cui appaiono nel programma, non sempre sono emesse rispettando questa sequenza (dato che alcune risorse necessarie potrebbero essere occupate) e non è neanche necessario che terminino in tale ordine. Ovviamente, se l'istruzione 1 imposta il valore di un registro e l'istruzione 2 usa lo stesso registro, occorre prestare grande attenzione per evitare che l'istruzione 2 lo legga prima che contenga il giusto valore. Per far ciò in modo corretto è necessario tener conto di molti aspetti; ciononostante questo approccio garantisce un guadagno nelle prestazioni, dato che è possibile 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. Tutto ciò che può aiutare questo processo si rivela utile: per esempio rendere le istruzioni regolari, di lunghezza fissa e con pochi campi. Meno formati d'istruzioni ci sono, meglio è.

### Solo le istruzioni Load e Store fanno riferimento alla memoria

Uno dei modi più semplici per spezzare le operazioni in passi separati è quello di richiedere che, per la maggior parte delle istruzioni, gli operandi vengano prelevati dai registri e siano memorizzati al loro interno. L'operazione di spostamento degli operandi dalla memoria ai registri può essere invece compiuta separatamente mediante apposite istru-

zioni. Dato che l'accesso alla memoria può richiedere un tempo considerevole, il cui ritardo non è prevedibile, queste operazioni, a patto che non facciano niente altro se non muovere operandi tra registri e memoria, 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 ciò obbliga a scaricare in memoria tutti i valori dei registri per poi ricaricarli. Il miglior modo per evitare il più possibile questa operazione consiste nel disporre di un numero sufficiente di registri.

## 2.1.5 Parallelismo a livello d'istruzione

I progettisti di calcolatori si sforzano costantemente di migliorare le prestazioni delle loro macchine. Aumentare la velocità di clock per rendere i processori più veloci è una possibilità, ma per qualsiasi nuova architettura esiste un limite, dipendente dal momento storico, su che cosa sia possibile ottenere mediante la semplice forza bruta. Per questo motivo molti progettisti di computer vedono nel parallelismo (cioè nel compiere più azioni allo stesso tempo) un modo per ottenere prestazioni più elevate con una data velocità di clock.

Il parallelismo può essere presente in due forme principali: a livello d'istruzione e a livello di processore. Nella prima forma il parallelismo è sfruttato all'interno delle singole istruzioni per far sì che la macchina possa elaborarne un maggior numero al secondo, mentre nella seconda sono presenti più CPU che lavorano congiuntamente su uno stesso problema. Entrambi gli approcci hanno i propri pregi; in questo paragrafo analizzeremo il primo tipo di parallelismo, mentre nel prossimo studieremo quello a livello di processore.

### Pipelining

Da anni ormai è assodato che uno dei maggiori colli di bottiglia nella velocità di esecuzione delle istruzioni è rappresentato dal prelievo delle istruzioni dalla memoria. Fin dal modello IBM Stretch (1959), per alleviare questo problema, i calcolatori sono stati dotati della capacità di poter prelevare in anticipo le istruzioni dalla memoria, in modo da averle già a disposizione nel momento in cui dovessero rendersi necessarie. Le istruzioni venivano memorizzate in un insieme di registri chiamati buffer di prefetch, dai quali potevano essere prese nel momento in cui venivano richieste, senza dover attendere che si completasse 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 concetto di *pipeline* spinge questa strategia molto più avanti; invece di dividere l'esecuzione di un'istruzione solamente in due fasi, la si divide in un numero maggiore di parti (spesso una dozzina o più) che

<sup>2</sup> Tra l'altro, MIPS – acronimo di *Millions of Instructions Per Second* – sta per "milioni d'istruzioni al secondo", e il processore MIPS fu così chiamato in questo modo come gioco di parole in relazione all'acronimo. Ufficialmente il nome del processore sta per *Microprocessor without Interlocked Pipeline Stages* (N.d.R.).

### Tutte le istruzioni sono eseguite direttamente dall'hardware

Tutte le comuni istruzioni sono eseguite direttamente dall'hardware, e non sono interpretate mediante microistruzioni. L'eliminazione di un livello di interpretazione garantisce velocità più alte per la maggior parte delle istruzioni. Per i calcolatori che implementano insiemi d'istruzioni CISC, quelle più complesse devono essere divise in varie parti che possono essere eseguite come una sequenza di microistruzioni. Questo passo aggiuntivo rallenta la macchina, ma può tuttavia essere accettabile per le istruzioni utilizzate con minor frequenza.

### Massimizzare la frequenza di emissione delle istruzioni

I calcolatori moderni ricorrono a molti trucchi per massimizzare le loro prestazioni, il principale dei quali consiste nel cercare di iniziare a eseguire più istruzioni al secondo. Dopo tutto, se si riescono a emettere 500 milioni d'istruzioni al secondo, si ottiene un processore da 500 MIPS, indipendentemente da quanto tempo impieghino quelle istruzioni per essere completate<sup>2</sup>. Questo principio suggerisce che il parallelismo può giocare un ruolo importante nelle prestazioni; 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.

Anche se le istruzioni vengono sempre incontrate nell'ordine in cui appaiono nel programma, non sempre sono emesse rispettando questa sequenza (dato che alcune risorse necessarie potrebbero essere occupate) e non è neanche necessario che terminino in tale ordine. Ovviamente, se l'istruzione 1 imposta il valore di un registro e l'istruzione 2 usa lo stesso registro, occorre prestare grande attenzione per evitare che l'istruzione 2 lo legga prima che contenga il giusto valore. Per far ciò in modo corretto è necessario tener conto di molti aspetti; ciononostante questo approccio garantisce un guadagno nelle prestazioni, dato che è possibile 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. Tutto ciò che può aiutare questo processo si rivela utile: per esempio rendere le istruzioni regolari, di lunghezza fissa e con pochi campi. Meno formati d'istruzioni ci sono, meglio è.

### Solo le istruzioni Load e Store fanno riferimento alla memoria

Uno dei modi più semplici per spezzare le operazioni in passi separati è quello di richiedere che, per la maggior parte delle istruzioni, gli operandi vengano prelevati dai registri e siano memorizzati al loro interno. L'operazione di spostamento degli operandi dalla memoria ai registri può essere invece compiuta separatamente mediante apposite istru-

<sup>2</sup> Tra l'altro, MIPS – acronimo di *Millions of Instructions Per Second* – sta per "milioni d'istruzioni al secondo", e il processore MIPS fu così chiamato in questo modo come gioco di parole in relazione all'acronimo. Ufficialmente il nome del processore sta per *Microprocessor without Interlocked Pipeline Stages* (N.d.R.).

zioni. Dato che l'accesso alla memoria può richiedere un tempo considerevole, il cui ritardo non è prevedibile, queste operazioni, a patto che non facciano niente altro se non muovere operandi tra registri e memoria, 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 ciò obbliga a scaricare in memoria tutti i valori dei registri per poi ricaricarli. Il miglior modo per evitare il più possibile questa operazione consiste nel disporre di un numero sufficiente di registri.

### 2.1.5 Parallelismo a livello d'istruzione

I progettisti di calcolatori si sforzano costantemente di migliorare le prestazioni delle loro macchine. Aumentare la velocità di clock per rendere i processori più veloci è una possibilità, ma per qualsiasi nuova architettura esiste un limite, dipendente dal momento storico, su che cosa sia possibile ottenere mediante la semplice forza bruta. Per questo motivo molti progettisti di computer vedono nel parallelismo (cioè nel compiere più azioni allo stesso tempo) un modo per ottenere prestazioni più elevate con una data velocità di clock.

Il parallelismo può essere presente in due forme principali: a livello d'istruzione e a livello di processore. Nella prima forma il parallelismo è sfruttato all'interno delle singole istruzioni per far sì che la macchina possa elaborarne un maggior numero al secondo, mentre nella seconda sono presenti più CPU che lavorano congiuntamente su uno stesso problema. Entrambi gli approcci hanno i propri pregi; in questo paragrafo analizzeremo il primo tipo di parallelismo, mentre nel prossimo studieremo quello a livello di processore.

### Pipelining

Da anni ormai è assodato che uno dei maggiori colli di bottiglia nella velocità di esecuzione delle istruzioni è rappresentato dal prelievo delle istruzioni dalla memoria. Fin dal modello IBM Stretch (1959), per alleviare questo problema, i calcolatori sono stati dotati della capacità di poter prelevare in anticipo le istruzioni dalla memoria, in modo da averle già a disposizione nel momento in cui dovessero rendersi necessarie. Le istruzioni venivano memorizzate in un insieme di registri chiamati *buffer di prefetch*, dai quali potevano essere prese nel momento in cui venivano richieste, senza dover attendere che si completasse 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 concetto di pipeline spinge questa strategia molto più avanti; invece di dividere l'esecuzione di un'istruzione solamente in due fasi, la si divide in un numero maggiore di parti (spesso una dozzina o più) che

possono essere eseguite in parallelo; ciascuna di queste parti è gestita da componenti hardware dedicati.

La Figura 2.4(b) illustra il modello di pipeline. Durante il primo ciclo di clock lo stadio S1 sta lavorando sull'istruzione 1, prelevandola dalla memoria. Durante il ciclo di clock 2 lo stadio S2 decodifica l'istruzione 1, mentre lo stadio S1 preleva l'istruzione 2. Durante il ciclo 3 lo stadio S3 preleva gli operandi per l'istruzione 1, lo stadio S2 decodifica l'istruzione 2 e lo stadio S1 preleva la terza istruzione. Durante il quarto ciclo lo stadio S4 esegue l'istruzione 1, S3 preleva gli operandi per l'istruzione 2, S2 decodifica l'istruzione 3 e S1 preleva l'istruzione 4. Infine, durante l'ultimo ciclo S5 scrive il risultato dell'istruzione 1, mentre gli altri componenti lavorano sulle istruzioni successive.

Per rendere più chiaro il concetto di *pipelining* consideriamo un'analogia. Immaginiamo una fabbrica di dolciumi in cui i reparti di cottura e di confezionamento sono separati.

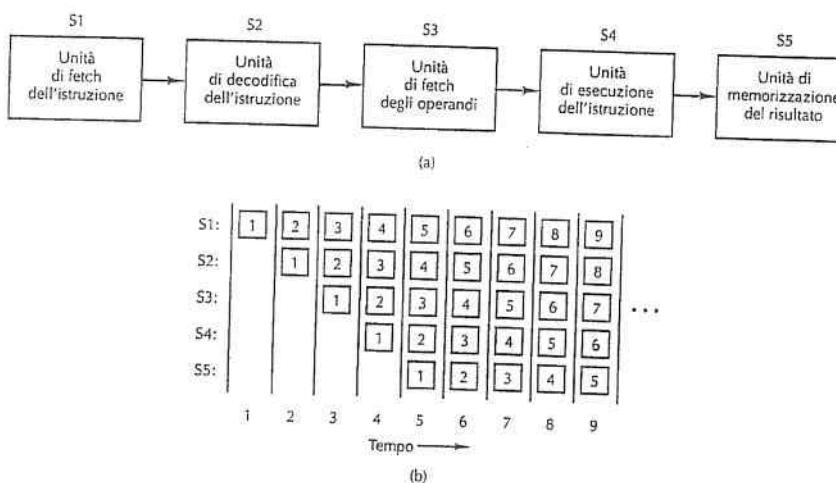


Figura 2.4 (a) Pipeline a cinque stadi. (b) Lo stato degli stadi in funzione del tempo. Sono mostrati nove cicli di clock.

Supponiamo che nel reparto spedizioni ci siano cinque operai (le unità di elaborazione) allineati lungo un nastro trasportatore. Ogni 10 secondi (il ciclo di clock), il primo operaio mette sul nastro una scatola vuota. La scatola viene passata al secondo operaio che inserisce al suo interno una torta. Poco dopo la scatola arriva alla postazione successiva dove viene chiusa e sigillata. Poi la scatola giunge al quarto operaio che vi attacca un'etichetta. Infine l'ultimo operaio rimuove la scatola dal nastro e la ripone in un grande contenitore per la spedizione a un supermercato. In sostanza questo è il modo in cui funziona la pipeline di un calcolatore: ogni istruzione (torta) attraversa vari passi di elaborazione prima di uscire, una volta completata, all'estremità opposta.

Tornando alla Figura 2.4 supponiamo che il ciclo di clock della macchina sia di 2 ns e che quindi un'istruzione impieghi 10 ns per percorrere i cinque stadi della pipeline. A prima vista, si potrebbe dire che la macchina abbia una velocità di 100 MIPS, ma in realtà le sue prestazioni sono molto migliori. Dato che in ogni ciclo di clock (2 ns) viene completata una nuova istruzione, la velocità reale di elaborazione è di 500 MIPS e non di soli 100.

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 della CPU). 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 a ogni ciclo di clock viene completata un'istruzione e visto che vi sono  $10^9/T$  cicli di clock al secondo, il numero d'istruzioni eseguite al secondo è  $10^9/T$ . Se per esempio  $T = 2$  ns, vengono eseguite 500 milioni d'istruzioni al secondo. Dividendo la velocità di esecuzione delle istruzioni per un milione si ottiene il numero di MIPS, cioè  $(10^9/T)/10^6 = 1000/T$  MIPS. In teoria potremmo misurare la velocità di esecuzione delle istruzioni in BIPS<sup>3</sup>, al posto di MIPS, ma dato che nessuno lo fa non lo faremo neanche noi.

### Architetture superscalari

Se è bene avere una pipeline, averne due è sicuramente meglio. La Figura 2.5 mostra un ipotetico progetto di una CPU con due pipeline, entrambe basate sullo schema della Figura 2.4. In questa situazione una singola unità di *fetch* preleva due istruzioni alla volta e le inserisce nelle pipeline, ognuna delle quali è dotata di una ALU. Affinché le due istruzioni possano essere eseguite in parallelo, non devono però esserci conflitti nell'uso delle risorse (cioè i registri) e nessuna delle due istruzioni deve dipendere dal risultato dell'altra. Come nel caso della singola pipeline, o è il compilatore a occuparsi di gestire correttamente questa situazione (l'hardware non effettua quindi alcun controllo e se le istruzioni sono incompatibili restituisce un risultato errato) oppure i conflitti sono rilevati ed eliminati durante l'esecuzione per mezzo di componenti hardware ad hoc.

Anche se le pipeline, singole o doppie, sono principalmente usate sulle macchine RISC, a partire dal 486 (il 386 e i suoi predecessori ne erano privi), Intel ha cominciato a introdurre le pipeline anche nelle proprie CPU. Il 486 aveva una sola pipeline, mentre il primo Pentium ne aveva due, entrambe a cinque stadi e simili a quelle mostrate nella Figura 2.5, sebbene la divisione esatta del lavoro tra gli stadi 2 e 3 (chiamati *decode-1* e *decode-2*) fosse leggermente diversa rispetto al nostro esempio. La pipeline principale, chiamata pipeline u, poteva eseguire una qualsiasi istruzione Pentium. La seconda pipeline, chiamata pipeline v, poteva invece eseguire solamente semplici istruzioni su interi (oltre a una semplice istruzione in virgola mobile, FXCH).

Alcune regole prestabilite determinavano se due istruzioni potevano essere eseguite in parallelo. Se tali istruzioni non erano sufficientemente semplici oppure erano incompatibili, soltanto la prima delle due veniva eseguita (nella pipeline u), mentre la seconda veniva trattenuta per essere successivamente accoppiata all'istruzione seguente. Le istruzioni erano sempre eseguite in ordine; per questo motivo compilatori specifici per

<sup>3</sup> Qui la B sta per *Billions*, cioè miliardi (N.d.T.).

il Pentium, che erano in grado di produrre coppie d'istruzioni compatibili, potevano generare programmi la cui esecuzione era più veloce rispetto a quella dei vecchi compilatori. Nel caso di programmi che lavoravano su interi, varie misurazioni mostrarono che su un Pentium il codice appositamente ottimizzato per la sua architettura veniva eseguito esattamente due volte più velocemente che su un 486 funzionante alla stessa velocità di clock (Pountain, 1993). Questo guadagno era da attribuire interamente alla seconda pipeline.

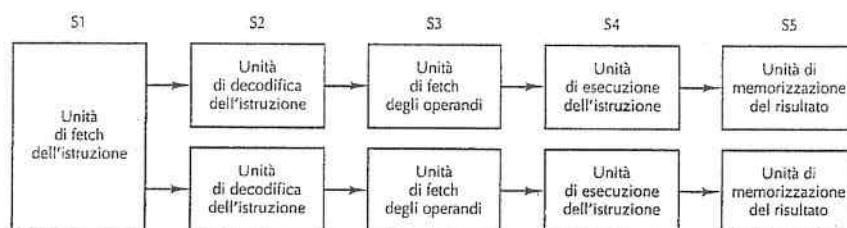


Figura 2.5 Doppia pipeline a cinque stadi con l'unità di *fetch* dell'istruzione in comune.

Anche se si potrebbe immaginare un'architettura a quattro pipeline, ciò obbligherebbe a duplicare troppi componenti hardware (gli informatici non credono nella magia del numero tre); per questo motivo nelle CPU di gamma alta si usa un approccio di tipo diverso. L'idea base consiste nell'avere una singola pipeline, ma di associarle più unità funzionali, com'è mostrato nella Figura 2.6. L'architettura Intel Core, per esempio, ha una struttura simile a quella della figura e sarà esaminata nel corso del Capitolo 4. Nel 1987 fu coniato il termine **architettura superscalare** (Agerwala e Cocke, 1987) per indicare questo approccio; le sue radici risalgono però al calcolatore CDC 6600, ideato più di 40 anni prima. Il 6600 prelevava un'istruzione ogni 100 ns e la passava a una delle 10 unità funzionali che lavoravano in parallelo mentre la CPU era occupata a lanciare l'istruzione successiva.

Nel corso del tempo la definizione di "superscalare" si è in qualche modo evoluta; ora è utilizzata per descrivere processori che lanciano più istruzioni (spesso quattro o sei) durante un ciclo di clock. Ovviamente una CPU superscalare, per poter gestire tutte queste istruzioni, deve avere più unità funzionali. Dato che i processori superscalari hanno generalmente una sola pipeline, essi assomigliano allo schema della Figura 2.6.

Se si utilizza questa definizione, allora da un punto di vista tecnico il 6600 non era superscalare, in quanto lanciava una sola istruzione ogni ciclo. Tuttavia il risultato finale era praticamente lo stesso: le istruzioni venivano lanciate a una frequenza molto più elevata rispetto a quella a cui potevano essere eseguite. Vi è una differenza concettuale molto piccola tra una CPU con ciclo di clock di 100 ns che lancia un'istruzione per ciclo a un gruppo di unità funzionali e una CPU con ciclo di clock di 400 ns che lancia allo stesso gruppo di unità funzionali quattro istruzioni per ciclo. In entrambi i casi l'idea chiave è che il tasso di lancio è molto più elevato del tasso di esecuzione, con il carico di lavoro che viene distribuito su un gruppo di unità funzionali.

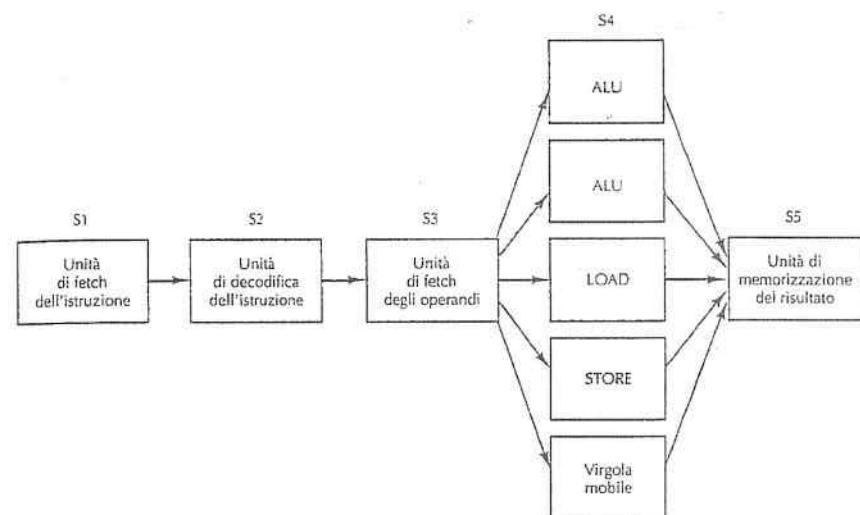


Figura 2.6 Processore superscalare con cinque unità funzionali.

Nell'idea di un processore superscalare è implicito il fatto che lo stadio S3 può lanciare istruzioni a una velocità significativamente superiore rispetto a quella a cui lo stadio S4 riesce a eseguirle. Se lo stadio S3 lanciasse un'istruzione ogni 10 ns e tutte le unità funzionali potessero compiere il loro lavoro in 10 ns, allora non più di un'istruzione alla volta sarebbe in esecuzione, facendo così crollare l'idea base. In realtà la maggior parte delle unità funzionali dello stadio 4 richiede un tempo di esecuzione significativamente maggiore di un singolo ciclo di clock, e ciò è in particolare vero per quelle che accedono alla memoria o che compiono calcoli aritmetici in virgola mobile. Come si può vedere nella figura è possibile avere più unità ALU nello stadio S4.

## 2.1.6 Parallelismo a livello di processore

La richiesta di calcolatori sempre più veloci sembra inarrestabile. Gli astronomi vogliono simulare che cosa successe durante il primo microsecondo successivo al big bang, gli economisti vogliono modellare l'economia su scala mondiale e gli adolescenti vogliono giocare su Internet con i loro amici virtuali con giochi 3D multimediali e interattivi. Dato che le CPU continuano a diventare più veloci, prima o poi ci si scontrerà con i problemi legati alla velocità della luce, il cui ritardo di propagazione è di 20 cm/ns sia nei cavi di rame sia nelle fibre ottiche, indipendentemente dall'abilità degli ingegneri di Intel. Inoltre, chip più veloci producono anche più calore, e la sua dissipazione costituisce un problema. È proprio la difficoltà di dissipare il calore prodotto a costituire il principale motivo per cui la velocità di clock delle CPU negli ultimi dieci anni ha subito una stagnazione.

Il parallelismo a livello d'istruzione aiuta in parte, ma difficilmente l'uso della pipeline e le operazioni superscalari possono aumentare le prestazioni di un fattore cinque o dieci. Per ottenere guadagni di 50, 100, e più, l'unica soluzione è quella di progettare calcolatori con più CPU; per questo motivo analizzeremo ora come sono organizzati alcuni di questi sistemi.

### Computer con parallelismo sui dati

Un gran numero di problemi in ambiti computazionali come la fisica, l'ingegneria e la computer graphic vengono trattati con l'utilizzo di cicli e array, o comunque hanno una struttura altamente regolare. Spesso gli stessi calcoli vengono eseguiti ripetutamente su diversi insiemi di dati. La regolarità e la struttura di questi programmi li rende particolarmente adatti a una esecuzione parallela che ne incrementa le prestazioni. Sono stati utilizzati principalmente due metodi per eseguire questi programmi altamente regolari in modo rapido ed efficiente: i processori SIMD e i processori vettoriali. Anche se questi due sistemi sono per molti versi piuttosto simili, il primo è in genere visto come un calcolatore parallelo mentre il secondo è stranamente considerato come un'estensione di un singolo processore.

I computer con parallelismo sui dati, grazie alla loro efficienza, hanno trovato molte applicazioni di successo. Essi sono in grado di offrire una grande potenza computazionale con l'utilizzo di un minor numero di transistor rispetto ad altri approcci. Gordon Moore (l'inventore della omonima legge) ha notoriamente ricordato che il costo del silicio è di circa 1 miliardo di dollari per ogni acro (equivalente a 4047 metri quadrati). Maggiore è la potenza calcolo che può essere ottenuta da un acro di silicio, più soldi una società di computer può guadagnare vendendolo. I processori con parallelismo sui dati sono uno dei mezzi più efficaci per ottenere alte prestazioni dal silicio. Poiché tutti i processori sono in esecuzione sulla stessa istruzione, il sistema ha bisogno di un solo "cervello" per controllare il computer. Il processore necessita dunque soltanto di uno stadio di prelievo, uno di decodifica e di una logica di controllo. Si tratta di un risparmio in termini di silicio che mette questi computer in condizioni di grande vantaggio sugli altri processori, a patto che il software eseguito sia altamente regolare e con molto parallelismo.

Un processore SIMD (*Single Instruction-stream Multiple Datastream*, "istruzioni singole, dati multipli") consiste di un elevato numero di processori identici che eseguono la stessa sequenza d'istruzioni su insiemi diversi di dati. Il primo sistema SIMD fu il calcolatore ILLIAC IV della University of Illinois (Bouknight et al., 1972). Il progetto originale prevedeva di costruire una macchina costituita da quattro quadranti, ciascuno formato da una griglia quadrata di 8 coppie di processori e memorie per lato. Una singola unità di controllo per ogni quadrante trasmetteva in broadcast le istruzioni, che venivano eseguite a passi sincronizzati da tutti i processori, ciascuno dei quali utilizzava i dati letti dalla propria memoria. Anche se venne costruito un solo quadrante per limitarne i costi, le prestazioni di ILLIAC IV raggiunsero i 50 megaflop (milioni di operazioni in virgola mobile al secondo). Si dice che, se la macchina fosse stata costruita interamente e se le sue prestazioni avessero raggiunto l'obiettivo originario (1 gigaflop), essa avrebbe avuto una potenza computazionale pari a due volte quella del mondo intero.

Le moderne unità di elaborazione grafica (GPU) si affidano in larga misura all'elaborazione SIMD per offrire grande potenza di calcolo con pochi transistor. L'elaborazione grafica si presta a processori SIMD perché la maggior parte degli algoritmi sono altamente regolari, con operazioni ripetute su pixel, vertici, texture e contorni. La Figura 2.7 mostra il processore SIMD della GPU Nvidia Fermi. Una GPU Fermi contiene fino a 16 multiprocessori streaming (SM) di tipo SIMD, e ogni SM contiene 32 processori SIMD. A ogni ciclo, lo scheduler sceglie due thread da eseguire sul processore SIMD.

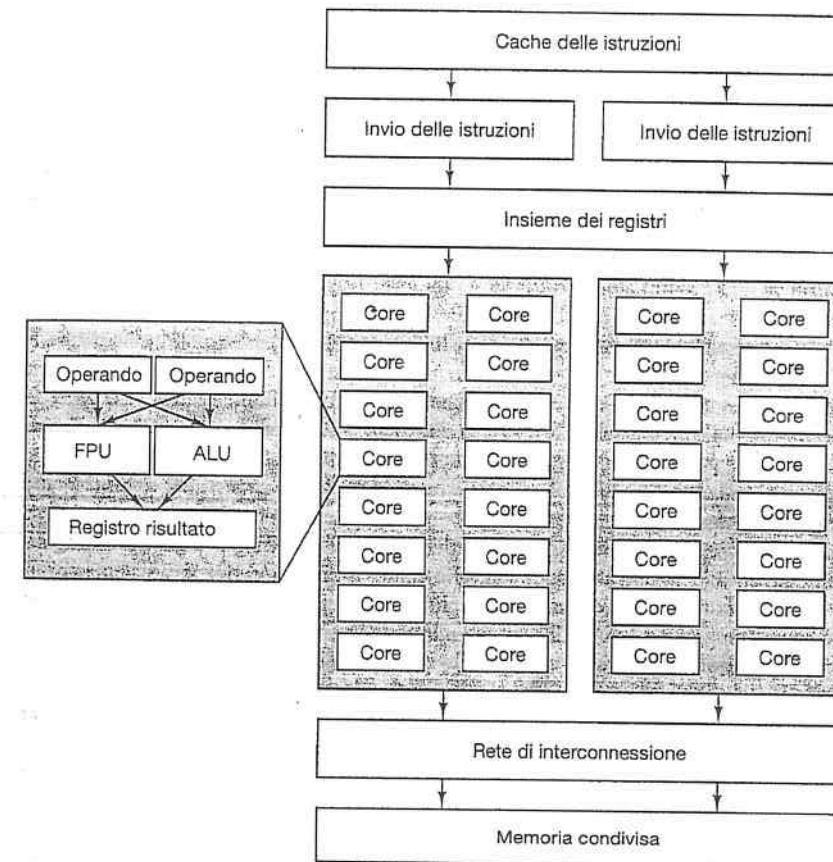


Figura 2.7 Il processore SIMD della GPU Fermi.

L'istruzione successiva di ogni thread verrà poi eseguita da 16 processori SIMD (o meno, se non c'è abbastanza parallelismo dei dati). Se ogni thread è in grado di esegui-

re 16 operazioni per ciclo, una GPU Fermi con 32 SM a pieno carico eseguirà ben 512 operazioni per ciclo. Si tratta di un risultato imponente, considerando che una CPU quad-core di uso generale di simili dimensioni dovrebbe lottare per raggiungere anche solo un trentaduesimo di tale capacità di elaborazione.

Agli occhi di un programmatore, i processori vettoriali risultano molto simili a un processore SIMD. Anche questi processori 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, altamente strutturato a pipeline. L'azienda fondata da Seymour Cray, la Cray Research (che oggi fa parte di SGI), ha prodotto un gran numero di processori vettoriali, a partire dal modello Cray-1 nel 1974, fino ai modelli attuali.

Sia i processori SIMD sia i processori vettoriali lavorano su array di dati. Entrambi eseguono singole istruzioni che, per esempio, sommano a coppie gli elementi di due vettori ma, mentre un processore SIMD lo fa usando tanti sommatori quanti sono gli elementi dei vettori, in un processore vettoriale si utilizza invece un **registro vettoriale**, che consiste di un insieme di registri convenzionali caricabili dalla memoria in una singola istruzione (che, in realtà, li carica in modo sequenziale). Un'istruzione di somma tra vettori viene quindi eseguita su coppie di elementi prelevati da due registri vettoriali per alimentare un sommatore strutturato a pipeline. Il risultato è un altro vettore che può essere memorizzato in un registro vettoriale oppure utilizzato direttamente come operando per una nuova operazione vettoriale.

Le istruzioni SSE (*Streaming SIMD Extension*) dell'architettura Intel Core utilizzano questo modello di esecuzione per velocizzare i programmi altamente regolari, come le applicazioni multimediali o il software scientifico; sotto questo aspetto il calcolatore ILLIAC IV va considerato come uno degli antenati dell'architettura Intel Core.

### Multiprocessori

In un processore parallelo sui dati le unità di elaborazione non sono delle CPU indipendenti, dato che c'è un'unica unità di controllo condivisa fra tutte. Il primo sistema parallelo che analizziamo e che è composto da più CPU complete è il multiprocessore, cioè un sistema composto da più CPU con una memoria in comune (così come un gruppo di persone condivide la stessa lavagna). Dato che ogni CPU può leggere e scrivere una qualsiasi parte della memoria, esse devono coordinarsi (via software) per evitare di ostacolarsi a vicenda. Quando due o più CPU hanno la possibilità di interagire in modo così profondo si dice che sono *tightly coupled* (cioè legate strettamente).

Sono possibili vari schemi d'implementazione, il più semplice dei quali consiste nell'avere un singolo bus con più CPU, tutte connesse a un'unica memoria. La Figura 2.8(a) mostra un diagramma di un simile multiprocessore.

Non serve una gran fantasia per capire che si verificano dei conflitti se un gran numero di processori veloci tenta costantemente di accedere alla memoria attraverso lo stesso bus. Per ridurre queste contese e migliorare le prestazioni i progettisti di multiprocessori hanno ideato vari schemi. La Figura 2.8 mostra un'architettura in cui ogni processore possiede una propria memoria locale, non accessibile agli altri. Questa memoria è utilizzabile per contenere il codice del programma e quei dati che non dovranno essere condivisi. L'accesso a questa memoria privata non utilizza il bus principale, riducendo quindi in modo considerevole il traffico sul bus. Oltre a questo schema ne esistono anche degli altri (per esempio, quello detto *caching* si veda più avanti).

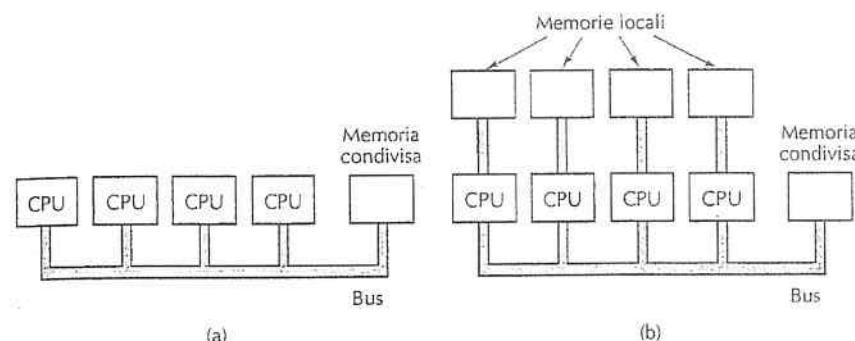


Figura 2.8 (a) Multiprocessore a bus singolo. (b) Multicomputer con memorie locali.

Rispetto ad altri tipi di calcolatori paralleli i multiprocessori hanno il vantaggio che è facile lavorare con un modello di programmazione basato su una memoria condivisa. Si immagini per esempio un programma che ricerca le cellule tumorali in un'immagine di un certo tessuto presa al microscopio. La fotografia digitalizzata potrebbe essere mantenuta in memoria comune e a ciascun processore potrebbe essere assegnata una particolare regione dell'immagine al cui interno effettuare la ricerca. Dato che ciascun processore ha accesso all'intera memoria, non vi è alcun problema se lo studio di una cellula, che inizia nella regione a esso assegnata, scavalca il limite della regione successiva.

### Multicomputer

Se da un lato è relativamente semplice costruire multiprocessori composti da un modesto numero di processori (non più di 256), è invece decisamente più complicato realizzarne di più grandi. La difficoltà risiede nel connettere tutti i processori alla memoria. Per aggirare questi problemi molti progettisti hanno semplicemente abbandonato l'idea di avere una memoria condivisa e hanno costruito sistemi composti da un gran numero di calcolatori interconnessi, ciascuno dotato di una memoria privata. Questi sistemi sono detti **multicomputer**. In questi sistemi le CPU sono dette con legame lasso (*loosely coupled*), in contrapposizione con quelle che compongono i multiprocessori.

Le CPU dei multicomputer comunicano fra loro inviandosi messaggi, simili alle e-mail, ma molto più veloci. Nel caso di grandi sistemi, dato che non è efficiente connettere mutualmente tutti i calcolatori, si utilizzano topologie diverse come griglie 2D e 3D, alberi e anelli. Ne consegue che i messaggi tra due calcolatori per spostarsi dalla sorgente alla destinazione spesso devono passare attraverso uno o più macchine intermedie oppure attraverso dei commutatori. Ciononostante è possibile ottenere che lo

scambio di messaggi richieda un tempo nell'ordine di pochi microsecondi. Sono stati costruiti dei multicomputer dotati di oltre 250.000 CPU, come l'IBM Blue Gene/P.

Visto che è facile programmare i multiprocessori, mentre i multicomputer sono facili da costruire, molte ricerche sono indirizzate alla realizzazione di sistemi ibridi che uniscono le qualità di entrambi. Tali calcolatori cercano di dare l'illusione che esista una memoria condivisa, senza però averne realmente una in quanto troppo costosa. Nel Capitolo 8 analizzeremo in dettaglio multiprocessori e multicomputer.

## 2.2 Memoria principale

La memoria è quella parte del calcolatore in cui sono depositati programmi e dati. Alcuni informatici (specialmente quelli britannici) utilizzano il termine inglese *store* o *storage* ("immagazzinare", "immagazzinamento") al posto di memoria, anche se il termine *storage* viene utilizzato sempre più frequentemente per riferirsi alla registrazione dei dati nei dischi. Se non ci fosse una memoria da cui il processore potesse leggere e scrivere informazioni, non esisterebbero i calcolatori digitali a programma memorizzato.

### 2.2.1 Bit

L'unità base della memoria è la cifra binaria, chiamata bit. Un bit può avere valore 0 oppure 1 ed è l'unità più semplice possibile. (Difficilmente un dispositivo in grado di memorizzare soltanto 0 potrebbe essere alla base di un sistema di memorizzazione; è infatti necessario che ci siano almeno due valori distinti.)

Quando si dice che i calcolatori utilizzano l'aritmetica binaria perché è "efficiente", s'intende (anche se spesso non ci si rende conto) che l'informazione digitale può essere memorizzata utilizzando dei valori di una certa quantità fisica continua, come la tensione o la corrente. Se occorre distinguere più valori, allora ci deve essere una minor separazione tra valori adiacenti e la memoria risulta di conseguenza meno affidabile. Dato che la numerazione binaria (Appendice A) richiede due soli valori distinti, risulta il metodo più affidabile per codificare l'informazione digitale.

Alcuni calcolatori, tra i quali i mainframe IBM, sono pubblicizzati affermando che sono dotati di aritmetica decimale oltre di quella binaria. Il trucco utilizzato consiste nell'usare 4 bit per memorizzare una cifra decimale mediante un codice chiamato BCD (*Binary Coded Decimal*, "decimale codificato in binario"). Quattro bit forniscono 16 combinazioni, di cui 10 sono utilizzate per le cifre da 0 a 9, mentre 6 sono inutilizzate. Di seguito mostriamo la codifica decimale e binaria del numero 1944, utilizzando 16 bit in entrambi i casi:

decimale: 0001 1001 0100 0100      binaria: 000001110011000

Nel formato decimale sedici bit possono memorizzare i numeri da 0 a 9999, permettendo solo 10.000 combinazioni, mentre una stringa binaria di 16 bit può avere 65.536 combinazioni distinte. Per questa ragione si dice che il sistema binario è più efficiente.

Consideriamo tuttavia che cosa accadrebbe se un giovane e brillante ingegnere elettronico inventasse un dispositivo altamente affidabile in grado di memorizzare direttamente le cifre da 0 a 9 dividendo in 10 intervalli la regione compresa tra 0 e 10 Volt. Quattro di questi dispositivi potrebbero memorizzare un qualsiasi numero decimale tra 0 e 9999, fornendo così 10.000 combinazioni. Essi potrebbero essere utilizzati anche per memorizzare numeri binari usando solamente le cifre 0 e 1, ma in questo caso, con quattro dispositivi, si otterebbero soltanto 16 combinazioni. Con un simile dispositivo il sistema decimale risulterebbe ovviamente più efficiente.

mentre le cifre da 0 a 9 dividendo in 10 intervalli la regione compresa tra 0 e 10 Volt. Quattro di questi dispositivi potrebbero memorizzare un qualsiasi numero decimale tra 0 e 9999, fornendo così 10.000 combinazioni. Essi potrebbero essere utilizzati anche per memorizzare numeri binari usando solamente le cifre 0 e 1, ma in questo caso, con quattro dispositivi, si otterebbero soltanto 16 combinazioni. Con un simile dispositivo il sistema decimale risulterebbe ovviamente più efficiente.

### 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 a essa. Se una memoria ha  $n$  celle, i suoi indirizzi varieranno da 0 a  $n - 1$ . Tutte le celle di una memoria contengono lo stesso numero di bit; se una cella è costituita da  $k$  bit, essa può contenere una qualsiasi delle  $2^k$  diverse combinazioni di bit. La Figura 2.9 mostra tre diverse organizzazioni di una memoria a 96 bit; si noti che (per definizione) le celle adiacenti hanno indirizzi consecutivi.

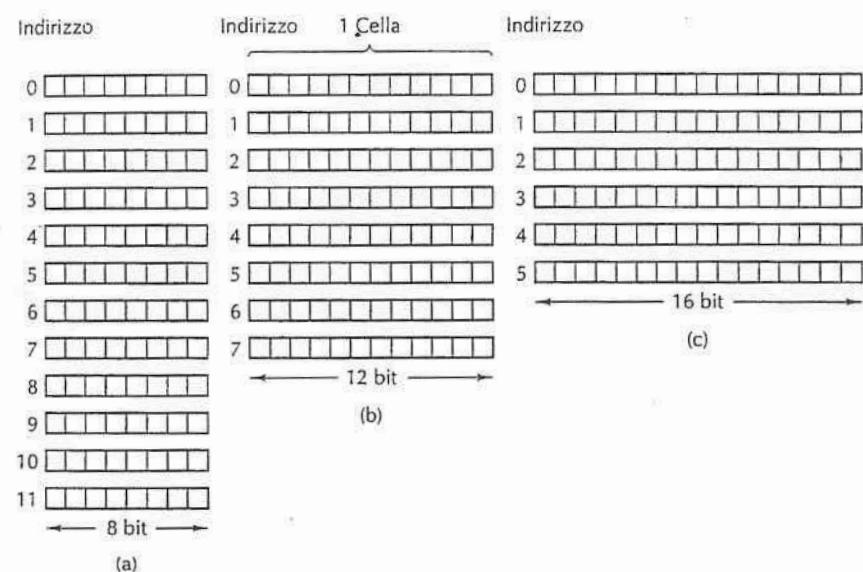


Figura 2.9 Tre modi di organizzare una memoria a 96 bit.

I calcolatori che usano il sistema numerico binario (comprese la notazione ottale ed esadecimale) esprimono gli indirizzi di memoria in notazione binaria. Se un indirizzo ha  $m$  bit, il massimo numero di celle indirizzabili è  $2^m$ . Per esempio un indirizzo usato

che modifica per esempio la parola 0000000000 nella parola 0000000111 non può essere corretto.

Immaginiamo di voler progettare un codice con  $n$  bit di dati e  $r$  bit di controllo, capace di correggere tutti gli errori singoli. Ciascuna delle  $2^m$  parole di memoria legali ha  $n$  parole di codice illegali a distanza 1 da essa. Queste sono formate invertendo sistematicamente ciascuno degli  $n$  bit nella parola di codice generata dalla parola di memoria considerata. Ciascuna delle  $2^m$  parole di memoria legali richiede quindi  $n + 1$  stringhe di bit a essa dedicate ( $n$  per i possibili errori e 1 per la combinazione corretta). Dato che il numero totale di combinazioni di bit è  $2^n$  deve valere  $(n + 1)2^m \leq 2^n$ ; questa disequazione può essere riscritta come  $(n + r + 1) \leq 2^r$ , dato che  $n = m + r$ . Conoscendo  $m$ , essa fornisce un limite inferiore al numero di bit di controllo richiesti per correggere errori singoli. La Figura 2.13 mostra il numero di bit di controllo richiesti per parole di memoria di varie lunghezze.

È possibile ottenere questo limite teorico utilizzando un metodo ideato da Richard Hamming. Prima di analizzare l'algoritmo di Hamming, guardiamo una semplice rappresentazione grafica che illustra chiaramente l'idea di un codice a correzione di errore per parole a 4 bit. Il diagramma di Venn della Figura 2.14(a) contiene tre cerchi,  $A$ ,  $B$  e  $C$ , che insieme formano sette regioni. Come esempio codifichiamo la parola di memoria a 4 bit 1100 nelle regioni  $AB$ ,  $ABC$ ,  $AC$  e  $BC$ , inserendo un bit in ogni regione (in ordine alfabetico). Questa codifica è mostrata nella Figura 2.14(a).

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

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

Aggiungiamo poi un bit di parità a ciascuna delle tre regioni vuote per ottenere al loro interno una parità pari, come illustrato nella Figura 2.14(b). Per costruzione la somma dei bit in ciascuno dei tre cerchi,  $A$ ,  $B$  e  $C$ , è ora un numero pari. Nel cerchio  $A$  abbiamo i quattro numeri 0, 0, 1 e 1, la cui somma fa 2, un numero pari. Nel cerchio  $B$  i numeri sono 1, 1, 0 e 0, che sommati fanno ancora 2, un numero pari. Infine nel cerchio  $C$  abbiamo nuovamente la stessa situazione. In questo esempio tutti i cerchi hanno la stessa somma, ma, in altre situazioni, è possibile ottenere somme diverse, come 0 e 4. Questa figura corrisponde a una parola di codice con 4 bit di dati e 3 bit di parità.

Supponiamo ora che il bit nella regione  $AC$  diventi errato, passando dal valore 0 al valore 1, come mostrato nella Figura 2.14(c). A questo punto il calcolatore può vedere

che i cerchi  $A$  e  $C$  hanno la parità errata (dispari). L'unico cambiamento di un singolo bit che li può correggere corrisponde a riportare  $AC$  al valore 0; questa modifica corregge correttamente l'errore. Seguendo questa strategia il calcolatore può riparare automaticamente gli errori di memoria che coinvolgono un solo bit.

Vediamo ora come l'algoritmo di Hamming può essere usato per costruire codici a correzione di errore per parole di memoria dalla dimensione arbitraria. In un codice di Hamming gli  $r$  bit di parità sono aggiunti a una parola a  $m$  bit, formando una nuova parola di lunghezza  $m + r$  bit. I bit sono numerati a partire da 1, non 0, con il bit 1 nella posizione più a sinistra (più significativa). Tutti i bit la cui posizione è una potenza di 2 sono bit di parità; quelli restanti sono usati invece per i dati. Per esempio, con una parola a 16 bit, vengono aggiunti 5 bit di parità nelle posizioni 1, 2, 4, 8 e 16, mentre le altre contengono bit di dati. In totale la parola di memoria diventa di 21 bit (16 di dati, 5 di parità). In questo esempio useremo (arbitrariamente) la parità pari.

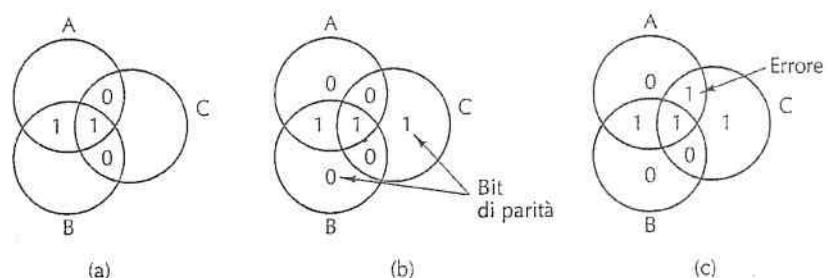


Figura 2.14 (a) Codifica di 1100. (b) Aggiunta della parità pari. (c) Errore in AC.

Ciascun bit di parità controlla alcune posizioni dei bit specifiche ed è impostato in modo che sia pari il numero totale di bit che hanno valore 1 nelle posizioni controllate. Ogni bit di parità controlla le seguenti posizioni:

- il bit 1 controlla i bit 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21
- il bit 2 controlla i bit 2, 3, 6, 7, 10, 11, 14, 15, 18, 19
- il bit 4 controlla i bit 4, 5, 6, 7, 12, 13, 14, 15, 20, 21
- il bit 8 controlla i bit 8, 9, 10, 11, 12, 13, 14, 15
- il bit 16 controlla i bit 16, 17, 18, 19, 20, 21

In generale il bit di posto  $b$  è controllato dai bit  $b_1, b_2, \dots, b_j$  tali che  $b_1 + b_2 + \dots + b_j = b$ . Per esempio il bit 5 è controllato dai bit 1 e 4 dato che  $1 + 4 = 5$ . Il bit 6 è controllato dai bit 2 e 4 in quanto  $2 + 4 = 6$ , e così via.

La Figura 2.15 mostra la correzione di un codice di Hamming per la parola di memoria a 16 bit 1111000010101110. La parola di codice a 21 bit corrispondente è 001011100000101101110. Per vedere come funziona la correzione di errore consideriamo che cosa accadrebbe se il bit 5 fosse invertito da un sovraccarico di tensione sulla linea di alimentazione. La nuova parola di codice diventerebbe 001001100000101101110

(invece di 00101110000010110110). Controllando i 5 bit di parità si otterebbero i seguenti risultati:

- 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).

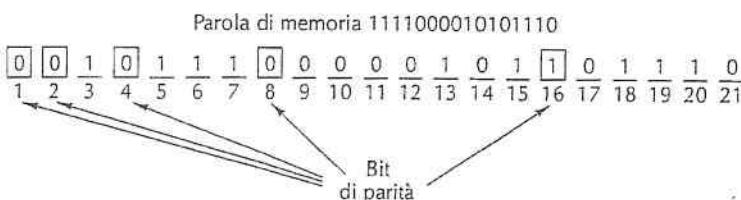


Figura 2.15 Costruzione del codice di Hamming per la parola di memoria 111100001010110 aggiungendo 5 bit di controllo ai 16 bit di dati.

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 errato 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

Storicamente le CPU sono sempre state più veloci delle memorie; i miglioramenti di quest'ultime hanno contribuito a incrementare anche le prestazioni delle CPU, preservan-

do di fatto lo squilibrio. Infatti, dato che diventa possibile collocare sempre più circuiti su un chip, i progettisti delle CPU stanno usando queste nuove possibilità per sviluppare le architetture a pipeline e superscalari, migliorando ulteriormente le velocità delle CPU. Al contrario i progettisti delle memorie hanno generalmente usato questa nuova tecnologia per aumentare la capacità dei loro chip, ma non la loro velocità, facendo peggiorare con il tempo lo squilibrio tra i due componenti. Il significato pratico di questa differenza di prestazioni è che, quando la CPU lancia una richiesta alla memoria, essa non otterrà la parola desiderata se non dopo molti cicli di CPU. Più lenta è la memoria, più cicli dovrà attendere la CPU.

Come abbiamo precisato precedentemente esistono due modi per trattare questo problema. Il metodo più semplice consiste nel far iniziare le istruzioni di lettura dalla memoria non appena vengono incontrate, permettendo al contempo di continuare l'esecuzione e bloccando la CPU quando un'istruzione tenta di usare una parola di memoria non ancora arrivata. Più lenta è la memoria, più frequentemente si verifica questo problema e maggiore è la penalizzazione che ne consegue. Per esempio, se un'istruzione su cinque accede alla memoria e il ritardo della memoria è di cinque cicli, il tempo di esecuzione sarà il doppio di quanto sarebbe stato con memoria istantanea. Se però il tempo di accesso alla memoria è di 50 cicli, il tempo di esecuzione crescerà di un fattore 11 (5 cicli per l'esecuzione delle istruzioni, più 50 cicli per l'attesa per la memoria).

L'altra soluzione consiste nel richiedere ai compilatori di non generare codice che utilizzi parole ancor prima che queste siano arrivate, il che consente di avere macchine che non si bloccano. Il problema di questo approccio è che è molto più facile a dirsi che a farsi. Spesso dopo una LOAD non vi è nient'altro da fare e quindi il compilatore è costretto a inserire delle istruzioni fasulle, dette NOP (cioè nessuna operazione), che non eseguono alcuna azione, ma occupano uno slot di tempo. In sostanza questo approccio è uno stall software invece che hardware; il peggioramento delle prestazioni non cambia.

Attualmente il problema non sta nella tecnologia, ma piuttosto in considerazioni economiche. Gli ingegneri sanno come costruire memorie veloci quanto le CPU, ma, per poter andare alla stessa velocità, esse devono essere collocate sul chip della CPU (dato che utilizzare il bus per la memoria è troppo lento). Inserire una grande memoria sul chip della CPU la rende più grande, il che significa più costosa; inoltre, anche se il costo non fosse un problema, esistono comunque dei limiti alla dimensione di un chip di CPU. La scelta si riduce quindi tra avere una piccola quantità di memoria veloce o una grande quantità di memoria più lenta, anche se, idealmente, vorremmo avere una grande quantità di memoria veloce a un prezzo basso.

Nonostante queste limitazioni, ci sono delle tecniche interessanti che permettono di combinare una piccola quantità di memoria veloce con una grande quantità di memoria lenta al fine di ottenere a un prezzo modesto sia la velocità della memoria veloce (quasi) sia la capacità della memoria più grande. La piccola e veloce memoria è chiamata *cache* (dal francese *cacher*, che significa "nascondere")<sup>4</sup>. Ora descriveremo brevemente come

<sup>4</sup> In inglese la parola *cache* indica un deposito, una scorta di beni da utilizzare al momento opportuno (N.d.T.).

si usano queste memorie e come funzionano, mentre una descrizione più dettagliata sarà data nel Capitolo 4.

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, 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. Da anni si sa che i programmi non accedono alle loro memorie in modo completamente casuale. Se a un certo istante la memoria fa un riferimento all'indirizzo A è molto probabile che il successivo riferimento alla memoria si troverà nelle vicinanze di A. Un semplice esempio è il programma stesso, in quanto, fatta eccezione per i salti e le chiamate a procedura, le istruzioni sono prelevate da locazioni contigue di memoria. Inoltre la maggior parte del tempo di esecuzione di un programma è spesa nei cicli, nei quali si esegue ripetutamente un numero limitato d'istruzioni. Analogamente, è molto probabile che un programma per la manipolazione di matrici effettuerà svariati riferimenti alla stessa matrice prima di spostarsi su altri dati.

L'osservazione secondo la quale i riferimenti alla memoria fatti in un breve intervallo temporale tendono a utilizzare solo una piccola frazione della memoria totale è chiamata principio di località ed è alla base di tutti i sistemi di cache. L'idea generale prevede che quando una parola viene referenziata, la parola stessa e alcune parole vicine sono portate dalla grande e lenta memoria all'interno della cache, in modo che sia possibile accedervi velocemente in un secondo momento. La Figura 2.16 mostra una tipica organizzazione di CPU, cache e memoria centrale. Se durante un piccolo intervallo una parola è letta o scritta  $k$  volte, il calcolatore dovrà effettuare un solo riferimento alla memoria lenta e  $k - 1$  riferimenti alla memoria veloce. Maggiore è  $k$ , migliori sono le prestazioni complessive.

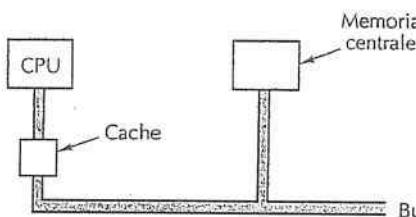


Figura 2.16 Da un punto di vista logico la cache si trova tra la CPU e la memoria centrale. Fisicamente può essere collocata in varie posizioni.

Possiamo formalizzare questo calcolo introducendo, il tempo di accesso alla cache, il tempo di accesso alla memoria centrale, la frequenza di successi (*hit ratio*), che corrisponde alla frazione di riferimenti che può essere soddisfatta dalla cache. Indicheremo

tali grandezze rispettivamente con le lettere  $c$ ,  $m$ , e  $h$ . Nel semplice esempio del paragrafo precedente si ottiene  $h = (k - 1)/k$ . Alcuni autori definiscono anche la frequenza di fallimento (*miss ratio*), che vale  $1 - h$ .

È quindi possibile calcolare il tempo medio di accesso utilizzando la seguente relazione:

$$\text{Tempo medio di accesso} = c + (1 - h)m$$

Se  $h$  si avvicina a 1, tutti i riferimenti possono essere soddisfatti dalla cache e il tempo medio di accesso si approssima a  $c$ . Al contrario, quando  $h$  tende a 0 è necessario effettuare ogni volta un riferimento alla memoria e quindi il tempo di accesso medio si avvicina a  $c + m$ , ovvero alla somma del tempo  $c$  necessario per controllare inizialmente (senza successo) la cache e del tempo  $m$  per fare successivamente riferimento alla memoria. Su alcuni sistemi il riferimento alla memoria può essere fatto partire parallelamente alla ricerca nella cache, di modo che, se si verifica un fallimento della cache, il ciclo di memoria sia già iniziato. Tuttavia questa strategia richiede che la memoria possa essere bloccata in ogni momento quando si verifica un successo della cache, rendendo l'implementazione più complicata.

Le memorie centrali e le cache sono divise in blocchi di grandezza fissa per trarre vantaggio dal principio di località. Solitamente quando si parla di questi blocchi all'interno della cache, ci si riferisce a essi con il termine linea di cache. Quando si verifica un fallimento della cache, è l'intera linea a essere caricata dalla memoria centrale all'interno della cache, e non soltanto la parola richiesta. Per esempio con una linea di 64 byte, un riferimento all'indirizzo di memoria 260 porterà la linea compresa tra i byte 256 e 319 all'interno della linea di cache. Con un po' di fortuna alcune delle altre parole presenti nella linea di cache diventeranno necessarie nel giro di poche istruzioni. È più efficiente seguire questa strategia piuttosto che prelevare singole parole, dato che richiede meno tempo prelevare  $k$  parole tutte in una volta che prelevarle una a una in  $k$  momenti diversi. Inoltre il fatto di avere elementi di cache più grandi di una singola parola significa che ce n'è un numero minore, il che richiede un overhead inferiore. Infine, molti computer sono in grado di trasferire 64 o 128 bit in parallelo in un singolo ciclo di bus, anche se si tratta di macchine a 32 bit.

La progettazione delle cache sta acquisendo un'importanza sempre maggiore nelle CPU ad alte prestazioni. Un problema riguarda la grandezza della cache; più grande è, migliori sono le sue prestazioni, ma anche il suo costo è maggiore. Un secondo problema è la dimensione della linea di cache. Una cache da 16 KB può essere infatti divisa in 1024 linee di 16 byte, 2048 linee di 8 byte, o in altri modi. Un terzo problema è come deve essere organizzata la cache, cioè come tener traccia di quali parole di memoria sono memorizzate al suo interno in un dato momento.

Un quarto problema consiste nella scelta se mantenere istruzioni e dati in una sola cache oppure in due. Il fatto di avere una cache unificata (istruzioni e dati usano la stessa cache) implica una progettazione più semplice e bilancia automaticamente i prelievi delle istruzioni e dei dati. Al giorno d'oggi tuttavia si tende a favorire la cache specializzata, in cui le istruzioni sono memorizzate in una cache e i dati in un'altra. Questa organizzazione è chiamata architettura Harvard, in riferimento al vecchio

calcolatore Mark III di Howard Aiken, che aveva memorie separate per le istruzioni e per i dati. Ciò che spinge i progettisti in questa direzione è l'utilizzo molto diffuso delle CPU con pipeline in cui l'unità di prelievo dell'istruzione richiede l'accesso alle istruzioni nello stesso momento in cui l'unità di prelievo dell'operando richiede l'accesso ai dati. Le cache specializzate permettono che questi due accessi avvengano in parallelo, mentre con una unificata ciò non è possibile. Inoltre dato che le istruzioni non sono modificate durante l'esecuzione, non vi è mai bisogno di riscrivere in memoria il contenuto della cache delle istruzioni.

Infine un quinto problema è rappresentato dal numero delle cache. Al giorno d'oggi è comune avere chip contenenti una cache primaria, avere una cache secondaria fuori dal chip, ma assemblata sullo stesso circuito della CPU, e avere anche una terza cache un po' più lontana.

## 2.2.6 Assemblaggio e tipi di memoria

A partire dalle prime memorie basate su semiconduttori e fino ai primi anni '90, la memoria era prodotta, venduta e installata in un unico chip. La densità dei chip passò da 1 Kbit a oltre 1 Mbit, ma ogni chip veniva venduto come unità separata. Spesso i primi PC erano dotati di prese nelle quali era possibile inserire chip di memoria aggiuntivi, se e quando l'utente ne avesse avuto bisogno.

A partire dall'inizio degli anni '90 viene utilizzata una diversa organizzazione. Vari chip, in genere di 8 o 16 elementi, sono montati su una piccola scheda a circuiti stampati venduta singolarmente. Essa è chiamata **SIMM** (*Single Inline Memory Module*, "modulo di memoria con connettore singolo") oppure **DIMM** (*Dual Inline Memory Module*, "modulo di memoria con doppia linea di contatti"), a seconda che abbia i connettori allineati su uno o su due lati della scheda. Le SIMM, poco utilizzate ai giorni nostri, hanno un connettore laterale con 72 contatti e trasferiscono 32 bit per ciclo di clock; le DIMM invece hanno generalmente connettori con 120 contatti su ogni lato, per un totale di 240 contatti, e trasferiscono 64 bit per ciclo di clock. Le DIMM più diffuse oggi sono le DDR3, la terza versione delle memorie a doppia velocità (*Double Data Rate*). La Figura 2.17 mostra lo schema tipico di una DIMM.

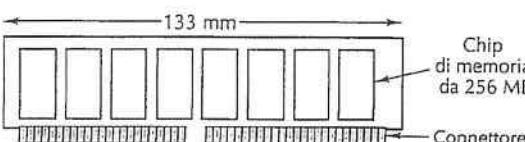


Figura 2.17 Vista di un modulo DIMM da 4GB con otto chip da 256 MB su ogni lato.  
L'altro lato è identico.

Una tipica configurazione di SIMM o DIMM può avere otto chip di dati da 256 MB ciascuno, per un totale di 2 GB sull'intero modulo. Molti calcolatori hanno spazio per

quattro moduli, permettendo una capacità totale di 8 GB in caso di utilizzo di moduli da 2 GB, o maggiore se si utilizzano memorie più grandi.

Nei calcolatori portatili si usano DIMM più piccole, chiamate **SO-DIMM** (*Small Outline DIMM*, "DIMM dal profilo compatto"). Alle SIMM e alle DIMM è possibile aggiungere un bit di parità o la correzione d'errore, ma, dato che la frequenza media degli errori di un modulo è di uno ogni 10 anni, spesso nella maggior parte dei calcolatori ordinari le funzionalità di rilevazione e correzione degli errori non sono presenti.

## 2.3 Memoria secondaria

La memoria centrale non è mai troppo grande. Si vorrebbero sempre memorizzare più informazioni rispetto alle reali capacità; la ragione principale è che, grazie ai miglioramenti della tecnologia, si comincia a voler memorizzare cose che precedentemente ricadevano interamente nel regno della fantascienza. Per esempio, visto che il governo degli USA obbliga gli enti governativi a pubblicare i propri bilanci, si potrebbe immaginare che la Biblioteca del Congresso decida di digitalizzare e mettere in commercio tutto il proprio contenuto ("Tutta la conoscenza umana a soli \$ 299,95"). Memorizzare all'incirca 50 milioni di libri, ciascuno con 1 MB di testo e 1 MB di immagini compresse, richiede  $10^{14}$  byte, vale a dire 100 Terabyte. La memorizzazione di tutti i 50.000 film finora prodotti richiederebbe più o meno lo stesso spazio. Questa quantità d'informazione, almeno per alcuni decenni, sarà troppo grande per poter essere contenuta in memoria centrale.

### 2.3.1 Gerarchie di memoria

La soluzione che viene tradizionalmente adottata per memorizzare una gran mole di dati consiste nell'organizzare gerarchicamente la memoria, com'è mostrato nella Figura 2.18. Nella parte alta della gerarchia si trovano i registri della CPU, ai quali si può accedere alla stessa velocità della CPU. Più sotto 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 1 (per i sistemi più economici e centinaia di gigabyte (per quelli professionali). Troviamo poi i dischi magnetici, la vera forza lavoro della memorizzazione permanente. Infine ci sono i nastri magnetici e i dischi ottici utilizzati per l'archiviazione.

Muovendosi verso il basso della gerarchia aumentano tre parametri chiave. Innanzitutto, il tempo di accesso diventa via via più grande. Ai registri della CPU è possibile accedere in un nanosecondo, o meno. Le memorie cache richiedono invece un tempo che è un piccolo multiplo di quello dei registri della CPU, mentre alle memorie centrali è generalmente possibile accedere in una decina di nanosecondi. A questo punto si incontra un grande salto, dato che il tempo di accesso ai dischi è maggiore di almeno 10 volte nel caso dei dischi a stato solido e di centinaia di volte nel caso dei dischi magnetici. L'accesso ai nastri e ai dischi ottici può essere misurato in secondi se i supporti devono essere presi e inseriti in un lettore.

Secondariamente, la capacità di memorizzazione aumenta man mano si scende verso il basso: i registri della CPU vanno bene per immagazzinare circa 128 byte, le cache per decine di megabyte, le memorie centrali per pochi gigabyte, i dischi a stato solido per

giungere un bit di parità o la correzione d'errore, ma, dato che la frequenza media degli errori di un modulo è di uno ogni 10 anni, spesso nella maggior parte dei calcolatori ordinari le funzionalità di rilevazione e correzione degli errori non sono presenti.

## 2.3 Memoria secondaria

La memoria centrale non è mai troppo grande. Si vorrebbero sempre memorizzare più informazioni rispetto alle reali capacità; la ragione principale è che, grazie ai miglioramenti della tecnologia, si comincia a voler memorizzare cose che precedentemente ricadevano interamente nel regno della fantascienza. Per esempio, visto che il governo degli USA obbliga gli enti governativi a pubblicare i propri bilanci, si potrebbe immaginare che la Biblioteca del Congresso decida di digitalizzare e mettere in commercio tutto il proprio contenuto ("Tutta la conoscenza umana a soli \$ 99,95"). Memorizzare all'incirca 50 milioni di libri, ciascuno con 1 MB di testo e 1 MB di immagini compresse, richiede  $10^{14}$  byte, vale a dire 100 Terabyte. La memorizzazione di tutti i 50.000 film finora prodotti richiederebbe più o meno lo stesso spazio. Questa quantità d'informazione, almeno per alcuni decenni, sarà troppo grande per poter essere contenuta in memoria centrale.

### 2.3.1 Gerarchie di memoria

La soluzione che viene tradizionalmente adottata per memorizzare una gran mole di dati consiste nell'organizzare gerarchicamente la memoria, com'è mostrato nella Figura 2.18. Nella parte alta 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, la vera forza lavoro per quanto riguarda la memorizzazione permanente. Infine ci sono i nastri magnetici e i dischi ottici utilizzati per l'archiviazione.

Muovendosi verso il basso della gerarchia aumentano tre parametri chiave. Innanzitutto, il tempo di accesso diventa via via più grande. Ai registri della CPU è possibile accedere in pochi nanosecondi. Le memorie cache richiedono invece un tempo che è un piccolo multiplo di quello dei registri della CPU, mentre alle memorie centrali è generalmente possibile accedere in poche decine di nanosecondi. A questo punto si incontra un grande salto, dato che il tempo di accesso ai dischi è di almeno 10 ms e l'accesso ai nastri e ai dischi ottici può essere misurato in secondi se i supporti devono essere presi e inseriti in un lettore.

Secondariamente, la capacità di memorizzazione aumenta man mano si scende verso il basso. I registri della CPU vanno bene per immagazzinare circa 128 byte, mentre le cache pochi megabyte; le memorie centrali possono memorizzare da decine a migliaia di megabyte e i dischi magnetici possono contenere quantità che variano da alcuni gigabyte fino a decine di gigabyte. I nastri e i dischi magnetici sono generalmente scollegati dal sistema e quindi la loro capacità è limitata soltanto dalle disponibilità economiche dell'utente.

In terzo luogo, scendendo lungo la gerarchia diminuiscono anche i costi unitari. Sebbene i prezzi attuali cambino rapidamente, i costi della memoria centrale sono dell'ordine

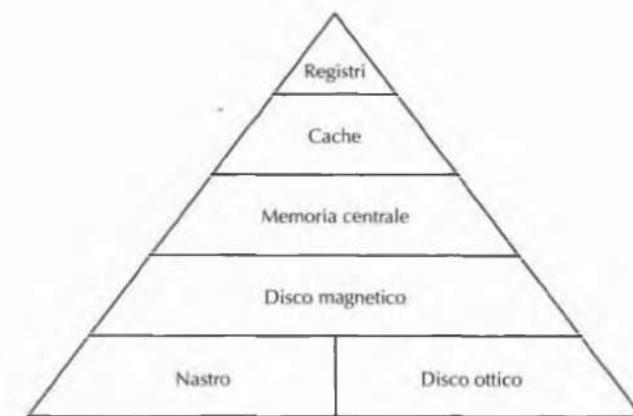


Figura 2.18 Gerarchia di memoria a cinque livelli.

di euro/megabyte, quelli dei dischi magnetici sono circa 100 volte inferiori e ancora più bassi quelli dei nastri magnetici.

Abbiamo già analizzato registri, cache e memoria centrale; nei paragrafi successivi studieremo i dischi magnetici e i dischi ottici. Non considereremo i nastri magnetici dato che sono usati raramente se non per il backup e, in ogni caso, non vi è molto da dire su di loro.

### 2.3.2 Dischi magnetici

Un disco magnetico consiste di uno o più piatti di alluminio rivestiti di materiale magnetico. Originariamente il diametro dei piatti poteva raggiungere i 50 cm, mentre al giorno d'oggi variano generalmente tra i 3 e i 12 cm; i dischi per i calcolatori portatili continuano a diventare sempre più piccoli e già oggi le loro dimensioni sono inferiori a 3 cm. La testina del disco, contenente un solenoide, sfiora la superficie rimanendo sospesa su un cuscinetto d'aria (mentre la testina dei floppy disk tocca realmente la superficie). 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. In questo modo, mentre il piatto ruota sotto la testina, è possibile scrivere e leggere un flusso di bit. La Figura 2.19 mostra la struttura di una traccia del disco.

La sequenza circolare di bit scritti mentre il disco compie una rotazione completa è chiamata **traccia**. Le tracce sono divise in un certo numero di **settori** di lunghezza fissa, contenenti in genere 512 byte di dati e preceduti da un **preambolo** che permette alla testina di sincronizzarsi prima della lettura o della scrittura. Dopo i dati segue un codice per la correzione di errore (ECC, *Error Correction Code*), un codice di Hamming oppure, più comunemente, un codice capace di correggere errori multipli, chiamato codice **Reed-Solomon**. Tra due settori consecutivi vi è un piccola area chiamata **spazio tra settori**. Alcuni produttori

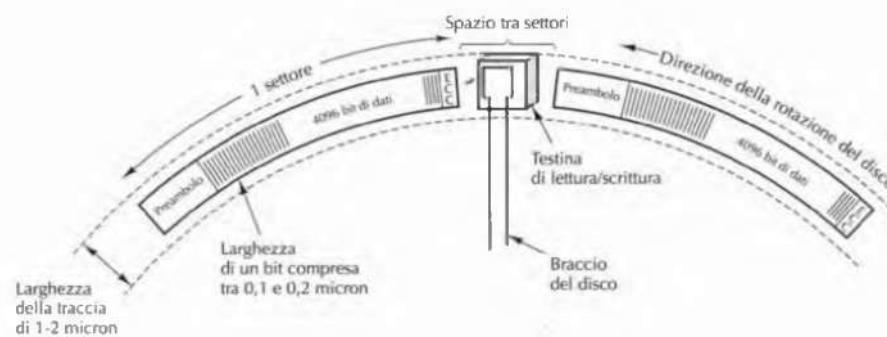


Figura 2.19 Porzione di una traccia del disco. Sono mostrati due settori.

indicano la capacità dei loro dischi nello stato non formattato (come se ciascuna traccia contiene solo dati), ma una misura più onesta dovrebbe essere la capacità formattata, che non conta i preamboli, gli ECC e gli spazi tra i dati. Di solito la capacità formattata è inferiore del 15% circa rispetto a quella lorda.

Tutti i dischi hanno bracci mobili in grado di spostarsi radialmente per posizionarsi su diverse distanze rispetto al centro di rotazione del piatto. Dato che è possibile scrivere una diversa traccia in corrispondenza di ogni diversa distanza radiale, le tracce sono una serie di centri concentrici attorno all'asse di rotazione del disco. La larghezza di una traccia dipende da quanto è larga la testina e da quanto accuratamente può essere posizionata. Con la tecnologia attuale i dischi hanno tra 5000 e 10.000 tracce per centimetro, con una larghezza per traccia che varia tra 1 e 2  $\mu$  ( $1 \mu = 1/1000$  mm). Occorre sottolineare che una traccia non è una scanalatura fisica nella superficie, ma semplicemente una corona circolare di materiale magnetizzato, con piccole aree di sicurezza che la separano da quella più interna e da quella più esterna.

La densità lineare dei bit lungo le tracce è diversa dalla densità radiale, ed è in gran parte determinata dalla purezza della superficie e dalla qualità dell'aria. I dischi attuali raggiungono densità comprese tra i 50.000 e i 100.000 bit/cm; la dimensione di un bit è quindi circa 50 volte maggiore sia nella direzione radiale sia lungo la circonferenza. Per raggiungere densità ancor più elevate i produttori di dischi stanno sviluppando tecnologie nelle quali la dimensione "lunga" dei bit non è disposta lungo la circonferenza dei dischi, ma verticalmente all'interno del materiale ferromagnetico. Questa tecnica è chiamata **registrazione perpendicolare** e verrà commercializzata a breve.

Al fine di ottenere un'alta qualità della superficie e dell'aria molti dischi sono sigillati durante la fabbricazione per impedire che vi si depositi la polvere. Questi dischi sono chiamati **dischi Winchester** e il primo modello (prodotto da IBM) aveva 30 MB di memoria fissa e sigillata e 30 MB di memoria removibile. Molto probabilmente il nome "Winchester" ha preso piede poiché questi dischi 30-30 fanno tornare alla mente i fucili Winchester 30-30 che ebbero un ruolo importante nell'avanzamento della frontiera americana.

La maggior parte dei dischi consiste di più piatti impilati verticalmente (come rappresentato nella Figura 2.20) in cui ciascuna superficie ha il proprio braccio e la propria

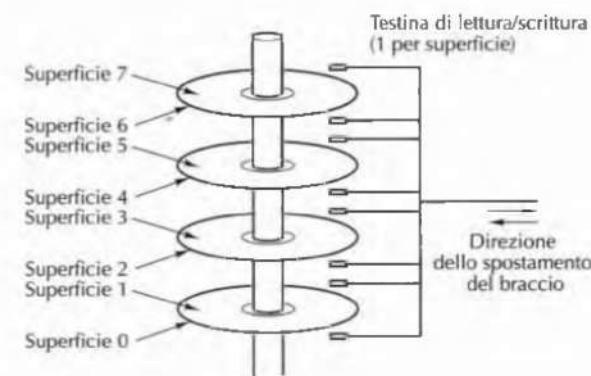


Figura 2.20 Disco con quattro piatti.

testina. L'insieme di tracce alla stessa distanza dal centro è chiamato **cilindro**. 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 dipendono da un insieme di fattori. Per leggere o scrivere un settore il braccio deve inizialmente effettuare quella che viene chiamata **seek** (ricerca), cioè lo spostamento radiale sulla posizione corretta. Il tempo medio di ricerca è compreso tra i 5 e i 10 ms, mentre gli spostamenti tra due tracce consecutive sono attualmente al di sotto del millisecondo. Una volta che la testina si è posizionata radialmente c'è un ritardo, chiamato **latenza rotazionale**, nell'attesa che il settore desiderato ruoti sotto la testina. Dato che la maggior parte dei dischi ruota a 5400, 7200 o 10.800 giri al minuto, il ritardo medio (corrispondente a metà rotazione) è compreso tra i 3 e i 6 ms. Il tempo di trasferimento dipende invece dalla densità lineare e dalla velocità rotazionale. Considerando che le velocità di trasferimento sono comunemente comprese tra i 20 e i 40 MB/s, un settore di 512 byte richiede tra i 13 e i 16  $\mu$ s; da ciò ne consegue che il tempo di ricerca su disco e la latenza rotazionale dominano la velocità di trasferimento. Leggere casualmente settori sparsi sul disco è dunque un modo chiaramente inefficiente di operare.

Vale la pena sottolineare che, a causa di preamboli, ECC, spazi tra i settori, tempi di ricerca su disco e latenze rotazionali, esiste una gran differenza tra **burst rate** e **sustained rate** di un disco. Il primo è la velocità a cui la testina può leggere dati dopo essersi posizionata sul primo bit. Il calcolatore deve essere in grado di gestire i dati letti a questa velocità (*burst* in inglese significa "raffica"); dall'altra parte il disco può mantenere questa velocità soltanto per un settore. Per alcune applicazioni, come quelle multimediali, è più importante il valore del **sustained rate**, cioè la velocità di lettura media nell'arco di un periodo di alcuni secondi, nel quale sono compresi anche i necessari tempi di ricerca e le latenze rotazionali.

Una semplice riflessione e l'uso della formuletta dei tempi del liceo,  $c = 2\pi r$ , svelano che le tracce più esterne hanno fra loro una distanza lineare maggiore rispetto a quanta non ne abbiano quelle più interne. Dato che, indipendentemente da dove si trova la testina, tutti i dischi magnetici ruotano a velocità angolare costante, questa osservazione mette in evidenza

un problema. Nei dischi più vecchi i produttori erano abituati a massimizzare la densità lineare nelle tracce più interne e ad abbassare la densità lineare di bit su quelle più esterne. Se un disco aveva, per esempio, 18 settori per traccia, allora ognuno di loro occupava un arco di 20 gradi, indipendentemente dal cilindro nel quale si trovasse.

Al giorno d'oggi si utilizza una strategia diversa. I cilindri sono divisi in zone (in genere da 10 a 30) e il numero di settori per traccia aumenta in ciascuna zona man mano che ci si sposta dalla traccia più interna verso l'esterno. Tutti i settori sono della stessa dimensione, ma la loro particolare disposizione rende più difficile tener traccia di dove si trovano le informazioni. Tuttavia questa tecnica permette di aumentare la capacità del disco e ciò è considerato un aspetto di maggiore importanza. La Figura 2.21 mostra un disco con cinque zone.

A ogni disco è associato un processore dedicato, chiamato **controllore del disco**. Fra i compiti del controllore c'è quello di accettare i comandi dal software, come READ, WRITE e FORMAT (scrittura dei preamboli), di controllare il movimento del braccio, di rilevare e correggere gli errori e di convertire gli 8 bit dei byte letti dalla memoria in uno stream seriale di bit e viceversa. Alcuni controlleri gestiscono anche la bufferizzazione di più settori, memorizzando in una cache i settori letti per un loro eventuale riutilizzo, oltre alla gestione dei settori contenenti errori. Quest'ultima funzione è necessaria in quanto esistono settori con regioni magnetizzate in modo permanente; quando il controllore scopre uno di questi settori lo sostituisce con uno dei settori liberi appositamente riservati a questo scopo all'interno di ciascun cilindro o di ciascuna zona.

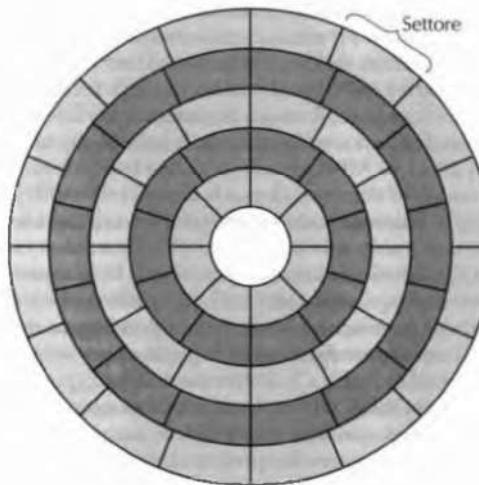


Figura 2.21 Disco con cinque zone. Ciascuna zona ha più tracce.

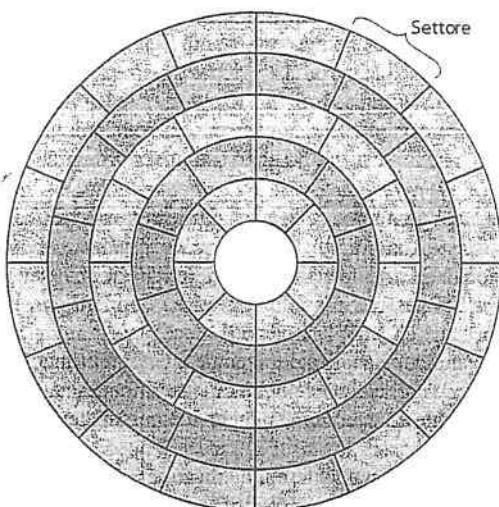


Figura 2.21 Disco con cinque zone. Ciascuna zona ha più tracce.

### 2.3.3 Dischi IDE

I dischi dei moderni personal computer sono un'evoluzione di quello che era presente nel PC XT della IBM; un disco da 10 MB della Seagate gestito da un controllore prodotto dalla Xebec collocato su una scheda aggiuntiva. Il disco della Seagate aveva 4 testine, 306 cilindri e 17 settori per traccia e il controllore era in grado di gestire due unità. Il sistema operativo leggeva dal disco e scriveva su di esso inserendo alcuni parametri nei registri della CPU e chiamando successivamente il **BIOS** (*Basic Input Output System*, "sistema elementare di input/output"), situato in una memoria di sola lettura integrata nel PC. Il BIOS lanciava le istruzioni macchina necessarie per caricare i registri del controllore, che iniziava il trasferimento dei dati.

La tecnologia evolse rapidamente e a metà degli anni '80 si passò alle prime unità **IDE** (*Integrated Drive Electronics*, "memoria di massa con elettronica integrata") in cui il controllo era strettamente integrato con l'unità, mentre prima si trovava su una scheda separata. Tuttavia, per ragioni di retrocompatibilità, le chiamate al BIOS non furono modificate. Secondo queste convenzioni l'indirizzamento dei settori avveniva specificando i numeri della loro testina, del cilindro e del settore, con la numerazione delle testine e dei cilindri che partiva da 0, mentre quella dei settori partiva da 1. Probabilmente questa scelta fu dovuta a un errore da parte del programmatore del BIOS originario, che scrisse il suo capolavoro usando l'assemblatore dell'8088. Utilizzando 4 bit per la testina, 6 bit per il settore e 10 bit per il cilindro, un'unità poteva avere al massimo 16 testine, 63 settori e 1024 cilindri, per un totale di 1.032.192 settori. Una simile unità poteva avere una capacità massima di 504 MB, che all'epoca poteva sembrare praticamente infinita, ma che al giorno d'oggi non lo è certamente. (Si criticerebbe oggi una macchina che non sia in grado di gestire dei dischi più grandi di un 1000 TB?)

Sfortunatamente, quando in seguito apparvero unità più capaci, la loro struttura (per esempio 4 testine, 32 settori, 2000 cilindri, cioè 256.000 settori) non poteva essere sfruttata; a causa delle convenzioni sulle chiamate al BIOS, rimaste per lungo tempo immutate, il sistema operativo non poteva infatti indirizzare questi dischi. Per aggirare il problema i controllori dei dischi cominciarono a imbrogliare, richiedendo che la geometria fosse all'interno dei limiti del BIOS, ma rimappando in realtà la geometria virtuale su quella reale. Pur funzionando, questo approccio rovinava però il lavoro del sistema operativo, che prestava attenzione nel collocare i dati in modo da minimizzare i tempi di ricerca.

Le unità IDE evolsero successivamente nelle unità **EIDE** (*Extended IDE*) che supportavano anche un secondo schema d'indirizzamento chiamato **LBA** (*Logical Block Addressing*, "indirizzamento logico dei blocchi"), che semplicemente numerava i settori a partire da 0 e fino a un massimo di  $2^{28} - 1$ . Questo schema, anche se richiedeva che il controllore convertisse gli indirizzi LBA negli indirizzi della testina, del settore e del cilindro, permetteva di superare il limite dei 504 MB. Sfortunatamente questa modalità d'indirizzamento creò un nuovo collo di bottiglia corrispondente a  $2^{28} \times 2^9$  byte (128 GB). Quando nel 1994 fu adottato lo standard EIDE, nessuno, nemmeno con la più fervida immaginazione, poteva immaginare dischi di quelle dimensioni. I comitati degli standard, allo stesso modo dei politici, hanno la tendenza a rinviare nel tempo i problemi, in modo che sia il comitato successivo a doverli risolvere.

Le unità e i controllori EIDE presentavano anche altri miglioramenti. I controllori EIDE, per esempio, potevano avere due canali, ciascuno dei quali poteva avere un disco primario e uno secondario. Questa organizzazione ammetteva un massimo di quattro unità per controllore. Vennero supportati anche i lettori di CD-ROM e DVD e la velocità di trasferimento dati passò da 4 MB/s a 16,67 MB/s.

Mentre la tecnologia dei dischi continuava a migliorare, evolgeva anche lo standard EIDE; per qualche strana ragione il suo successore fu però chiamato **ATA-3** (*AT Attachment*, "tecnologia avanzata per la connessione"), con un riferimento al PC/AT IBM (dove AT si riferiva alla allora Tecnologia Avanzata di una CPU a 16 bit che andava alla velocità di 8 MHz). Nella versione successiva lo standard fu chiamato **ATAPI-4** (*ATA Packet Interface*, "interfaccia per pacchetti ATA") e la velocità fu aumentata fino a 33 MB/s. Nello standard ATAPI-5 la velocità raggiunse invece i 66 MB/s.

A questo punto il limite dei 128 GB imposto dagli indirizzi LBA a 28 bit cominciò a profilarsi sempre più minaccioso, e quindi in ATAPI-6 la dimensione di LBA fu portata a 48 bit. Il nuovo standard finirà nei guai quando i dischi raggiungeranno la dimensione di  $2^{48} \times 2^9$  byte. Con un aumento annuale del 50% della capacità, il limite dovuto ai 48 bit dovrebbe sopravvivere fino al 2035 circa (per scoprire la soluzione a questo problema si consulti l'undicesima edizione di questo libro!). I più astuti stanno già scommettendo su un aumento della dimensione di LBA a 64 bit. Lo standard ATAPI-6 ha anche aumentato la velocità di trasferimento dati a 100 MB/s e ha affrontato per la prima volta il problema del rumore dei dischi.

Lo standard ATAPI-7 rappresenta una rottura radicale con il passato. Questo standard, invece di aumentare la dimensione del connettore del disco (per migliorare la velocità di trasferimento), usa quella che viene chiamata interfaccia serial ATA per

trasferire 1 bit alla volta su un connettore a 7 pin a una velocità che parte da 150 MB/s, ma che nel tempo dovrebbe raggiungere 1,5 GB/s. La sostituzione di un cavo piatto a 80 fili con un cavo rotondo di pochi millimetri di spessore migliora la circolazione dell'aria all'interno del calcolatore. Inoltre l'interfaccia serial ATA utilizza 0,5 volt per i segnali (rispetto ai 5 V delle unità ATAPI-6), riducendo il consumo di energia. Il problema del consumo energetico dei dischi sta acquistando una grande importanza, sia nei sistemi di fascia alta, sia in quelli di fascia bassa, dove i portatili hanno un'alimentazione limitata (Gurumurthi et al., 2003).

### 2.3.4 Dischi SCSI

Per quanto riguarda l'organizzazione di cilindri, tracce e settori, i dischi SCSI (si pronuncia *scsi*) non sono differenti da quelli IDE, ma hanno una diversa interfaccia e una velocità di trasferimento dati molto più elevata. Le origini dei dischi SCSI risalgono a Howard Shugart, inventore dei floppy disk, utilizzati nei primi modelli di personal computer negli anni '80. La sua società introdusse nel 1979 il disco SASI (*Shugart Associates System Interface*, "sistema d'interfaccia Shugart Associates"). Nel 1986, dopo alcune modifiche e molte polemiche, venne standardizzato dalla ANSI e il suo nome cambiò in SCSI (*Small Computer System Interface*, "sistema d'interfaccia per piccoli sistemi"). A partire da quel momento sono state standardizzate versioni sempre più veloci con il nome di Fast SCSI (10 MHz), Ultra SCSI (20 MHz), Ultra2 SCSI (40 MHz), Ultra3 SCSI (80 MHz), Ultra4 SCSI (160 MHz) e Ultra5 SCSI (320 MHz). Ciascuna di queste interfacce aveva anche una versione *wide* ("larga") a 16 bit, l'unica utilizzata ai giorni nostri; la Figura 2.22 mostra le combinazioni principali.

Nome	Bit di dati	MHz del bus	MB/s
SCSI-1	8	5	5
Fast SCSI	8	10	10
Wide Fast SCSI	16	10	20
Ultra SCSI	8	20	20
Wide Ultra SCSI	16	20	40
Ultra2 SCSI	8	40	40
Wide Ultra2 SCSI	16	40	80
Ultra3 SCSI	8	80	80
Wide Ultra3 SCSI	16	80	160
Ultra4 SCSI	8	160	160
Wide Ultra4 SCSI	16	160	320
Wide Ultra5 SCSI	16	320	640

Figura 2.22 Parametri dei modelli SCSI.

Per via della velocità di trasferimento più elevata, i dischi SCSI sono lo standard in workstation e server di fascia alta, in particolare nelle macchine con configurazione RAID (si veda più avanti).

SCSI non è soltanto un'interfaccia per hard disk, ma è anche un bus al quale possono essere collegati un controllore e sette dispositivi. Questi possono comprendere uno o più hard disk SCSI, CD-ROM, masterizzatori, scanner, unità a nastro e altre periferiche SCSI. Ciascun dispositivo SCSI possiede un ID, compreso tra 0 e 7 (15 nel caso di *wide* SCSI), e due connettori, uno per l'input e uno per l'output. I cavi connettono in serie l'output di un dispositivo con l'input del successivo, come un filo di lucine natalizie. L'ultimo dispositivo della fila deve avere una terminazione per evitare che le riflessioni nell'estremità del bus SCSI interferiscano con altri dati lungo il bus. In genere il controllore si trova su una scheda aggiuntiva all'inizio della catena, anche se ciò non è esplicitamente richiesto dallo standard.

Il più comune cavo per lo standard SCSI a 8 bit ha 50 contatti, 25 dei quali sono cavi di terra accoppiati uno a uno con altri 25 in modo da fornire un'eccellente immunità al rumore, necessaria per poter operare ad alte velocità. Dei 25 cavi 8 sono per i dati, 1 per la parità, 9 per il controllo, mentre i rimanenti sono per l'alimentazione oppure riservati per usi futuri. I dispositivi a 16 bit (e quello a 32 bit) richiedono un secondo cavo per trasportare i segnali aggiuntivi. I cavi possono avere una lunghezza di vari metri, permettendo l'utilizzo di hard disk, scanner e altre periferiche esterne.

I controllori SCSI possono funzionare come iniziatori o come destinatari di comunicazione. Di solito il controllore funziona come iniziatore e lancia comandi ai dischi e ad altre periferiche; questi comandi sono blocchi della dimensione massima di 16 byte che indicano al destinatario che cosa deve fare. I comandi e le risposte avvengono in fase, utilizzando vari segnali di controllo per delimitare la fase e arbitrare l'accesso al bus quando più dispositivi tentano di usarlo nello stesso momento. Questo arbitraggio è importante in quanto SCSI consente a tutti i dispositivi di funzionare contemporaneamente, aumentando potenzialmente le prestazioni in un ambiente in cui vi possono essere più processi attivi allo stesso tempo (per esempio UNIX o Windows XP); IDE ed EIDE ammettono invece un solo dispositivo attivo a ogni istante.

### 2.3.5 RAID

Nel corso dell'ultimo decennio le prestazioni delle CPU sono aumentate in modo esponenziale, raddoppiando all'incirca ogni 18 mesi, mentre la stessa cosa non si è verificata per i dischi. Negli anni '70 il tempo medio di ricerca nei dischi dei minicomputer era compreso tra 50 e 100 ms, mentre ora è di 10 ms. In molti settori dell'industria (come quella automobilistica o aeronautica) un miglioramento delle prestazioni di un fattore 5 o 10 nell'arco di due decenni sarebbe una notizia di rilievo, ma nel settore dei computer è un dato imbarazzante. Per questo motivo la differenza tra le prestazioni della CPU e quella dei dischi è aumentata con il tempo in modo considerevole.

Come già visto, spesso si utilizza l'elaborazione parallela per velocizzare le prestazioni delle CPU. Quindi anche l'I/O parallelo è sembrato ad alcuni una buona idea. In un articolo del 1988 (Patterson et al., 1988) fu suggerita un'organizzazione a sei dischi per migliorare le prestazioni, l'affidabilità oppure entrambe allo stesso tempo. Questa

idea fu adottata velocemente dal mercato e portò a una nuova classe di dispositivi di I/O chiamata RAID. Patterson e i suoi colleghi definirono RAID come Redundant Array of Inexpensive Disks ("insieme ridondante di dischi economici"), ma il mercato ridefinì la "I" di Inexpensive ("economico") con il termine Independent ("indipendente"): forse per poter utilizzare dischi costosi? Dato che c'è sempre bisogno di un nemico (come in CISC contro RISC, anch'esso dovuto a Patterson), il ruolo dei cattivi fu assunto da SLED (Single Large Expensive Disk, "singolo disco capiente e costoso").

L'idea su cui si basa RAID è quella di installare vicino al calcolatore, di solito un server di grandi dimensioni, un contenitore pieno di dischi, di sostituire la scheda contenente il controllore del disco con un controllore RAID, di copiare i dati sul RAID e di continuare quindi le normali operazioni. In altre parole un RAID dovrebbe apparire agli occhi del sistema operativo come uno SLED, seppur con prestazioni e affidabilità più elevate. Dato che i dischi SCSI offrono buone prestazioni, prezzo basso e possibilità di avere fino a 7 unità su un singolo controllore (15 per l'interfaccia wide SCSI) è naturale che la maggior parte dei sistemi RAID consista di un controllore RAID SCSI, più un insieme di dischi SCSI che appaiono al sistema operativo come un'unica e grande unità. In questo modo non sono necessarie modifiche al software per poter usare RAID, il che rappresenta un grande vantaggio per gli amministratori di sistema.

Tutti i RAID, oltre ad apparire al software come un singolo disco, hanno la proprietà di distribuire i dati sulle diverse unità per permettere la gestione parallela. Il gruppo di Patterson ha definito diversi schemi RAID, conosciuti con nomi che vanno da RAID livello 0 a RAID livello 5; esistono anche altri livelli di minor importanza che però non tratteremo. Il termine "livello" è in un certo modo fuorviante, in quanto non esiste alcuna gerarchia; si tratta semplicemente di sei diverse possibili organizzazioni, ognuna con un diverso mix di caratteristiche di affidabilità e prestazioni.

Il RAID livello 0 è mostrato nella Figura 2.23(a). Secondo questa organizzazione il disco virtuale simulato dal RAID è visto come se fosse suddiviso in *strip* ("strisce") di  $k$  settori ciascuna, con 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; se  $k = 1$  ogni *strip* è un settore, se  $k = 2$  una *strip* è composta da due settori e così via. L'organizzazione RAID livello 0 scrive sulle *strip* consecutive in modo circolare (modalità *round robin*), come mostra la Figura 2.23(a) nel caso di un RAID con quattro unità. Questo modo di distribuire dati su più unità è chiamato *striping*. Se per esempio il software lancia un comando per leggere un blocco di dati costituito da quattro *strip* consecutive a partire dall'inizio di una *strip*, il controllore RAID spezzerà questo comando in quattro comandi separati, uno per ciascuno dei quattro dischi, e li farà eseguire in parallelo. In questo modo si ottiene un I/O parallelo senza che il software ne sia a conoscenza.

Il RAID livello 0 lavora meglio quando le richieste sono di grandi dimensioni. Se la dimensione di una richiesta supera il numero di unità moltiplicato per la dimensione di una *strip*, allora alcune unità riceveranno più richieste, che verranno soddisfatte una dopo l'altra. Spezia al controllore separare la richiesta e assegnare alle unità appropriate i comandi corretti nella giusta sequenza, oltre ad assemblare poi, correttamente, i risultati in memoria. Le prestazioni sono eccellenti e l'implementazione è semplice.

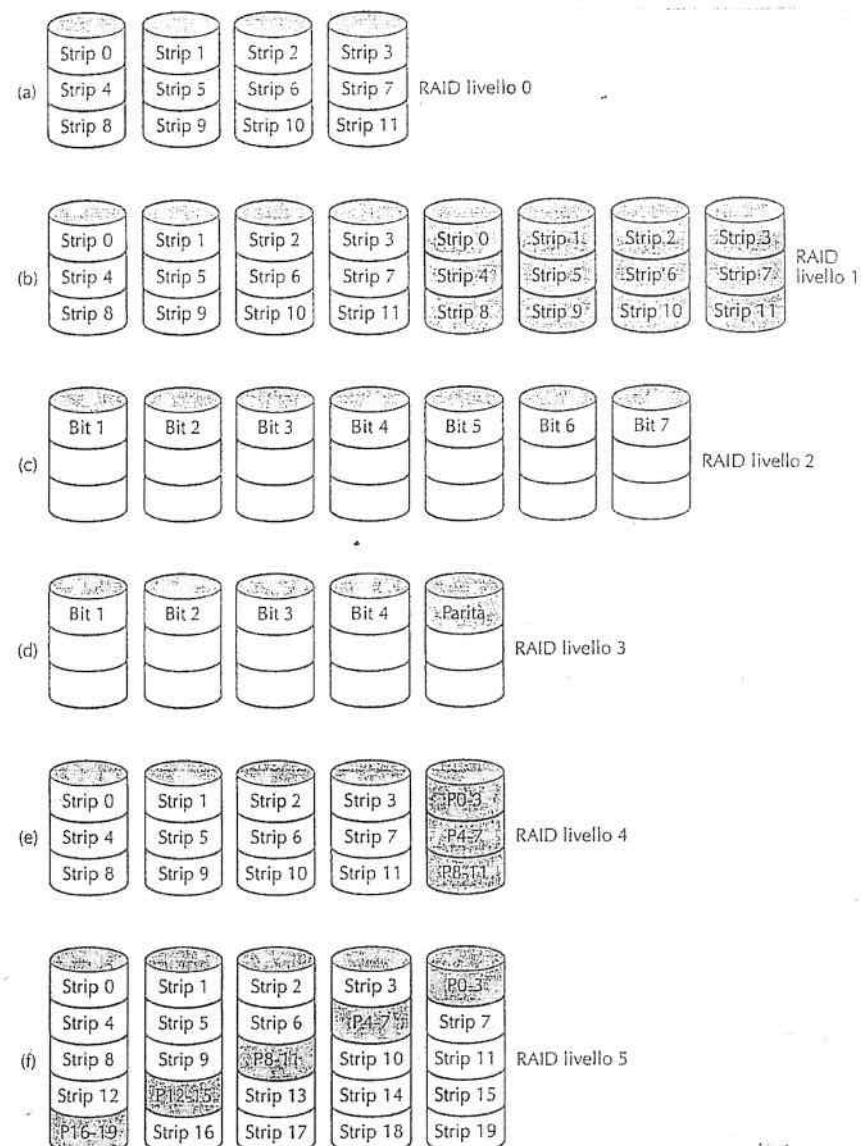


Figura 2.23 RAID dal livello 0 al livello 5. I dischi di backup e di parità sono in grigio.

Il RAID livello 0 lavora peggio con i sistemi operativi che tendono a richiedere i dati un settore alla volta. In una simile situazione il risultato sarà comunque corretto, ma non vi sarà parallelismo da sfruttare e quindi non si otterra alcun guadagno nelle prestazioni. Un altro svantaggio di questa organizzazione è che l'affidabilità è potenzialmente peggiore di quella che si ha con uno SLED. Se un RAID consiste di quattro dischi, ciascuno con un intervallo medio tra guasti di 20.000 ore, all'incirca ogni 5000 ore un disco si guasterà e i dati saranno interamente persi; uno SLED con un intervallo medio tra guasti di 20.000 ore sarebbe quattro volte più affidabile. Inoltre questo tipo di progettazione non è in realtà un vero e proprio RAID, in quanto non c'è alcuna ridondanza.

Il RAID livello 1, mostrato nella Figura 2.23(b), è invece un vero RAID in quanto duplica tutti i dischi, risultando così composto, in questo esempio, da quattro dischi primari e da quattro di backup, ma è anche possibile un qualsiasi altro numero pari di dischi. Nel caso di una scrittura ogni *strip* viene scritta due volte, mentre nel caso di una lettura è possibile usare entrambe le copie, distribuendo il carico di lavoro su più unità. Ne consegue che le prestazioni in scrittura non sono migliori di quelle che si ottengono con una singola unità, mentre quelle in lettura possono essere fino a due volte migliori. La tolleranza ai guasti è eccellente: se un disco si rompe, si può semplicemente utilizzare la copia. La riparazione consiste nell'installare un nuovo disco e copiarvi l'intero disco di backup.

Diversamente dai livelli 0 e 1 che lavorano con *strip* di settori, il RAID livello 2 funziona sulla base di una parola o, in alcuni casi, anche sulla base di un byte. Immaginiamo di dividere ciascun byte del singolo disco virtuale in una coppia di *nibble* (mezzo byte) di 4 bit, e di aggiungere a ciascuno di loro un codice di Hamming per formare una parola di 7 bit, in cui i bit 1, 2 e 4 sono bit di parità. Immaginiamo inoltre che i sette dischi della Figura 2.23(c) siano sincronizzati per quanto riguarda le posizioni dei bracci e delle rotazioni. In questo modo sarebbe possibile scrivere una parola del codice di Hamming a 7 bit sui sette dischi, usando un bit per ogni disco.

Il calcolatore Thinking Machines CM-2 utilizzava questo schema, prendendo parole di dati a 32 bit, aggiungendovi 6 bit di parità per formare una parola di Hamming a 38 bit, oltre a un bit extra per la parità dell'intera parola, e distribuendo ciascuna parola su 39 dischi. Il throughput totale era estremamente elevato, dato che nel tempo necessario per accedere a un settore era possibile scrivere una quantità di dati pari a 32 settori. Inoltre la perdita di un disco non causava problemi, in quanto comportava la perdita di 1 solo bit per ciascuna parola a 39 bit che veniva letta, perdita che il codice di Hamming riesce a gestire autonomamente.

Questa organizzazione presenta però alcuni difetti: la rotazione dei dischi deve essere sincronizzata e 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%). Inoltre si richiede molto lavoro al controllore, dato che deve calcolare la checksum del codice di Hamming a ogni tempo di bit.

Il RAID livello 3, mostrato nella Figura 2.23(d), 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. A prima vista

potrebbe sembrare che un solo bit di parità possa permettere la sola rilevazione degli errori, ma non la loro correzione. Nel caso di errori casuali, questa osservazione è vera; tuttavia, nel caso di una rottura di un disco, lo schema permette la correzione completa degli errori singoli, in quanto la posizione del bit errato è conosciuta. Se un disco si guasta il controllore assume semplicemente che tutti i suoi bit valgano 0; se una parola ha un valore errato di parità, allora il bit del disco guastatosi doveva per forza valere 1 e poteva quindi essere corretto. Anche se i RAID livello 2 e livello 3 permettono un tasso di trasferimento dati elevato, il numero di richieste di I/O al secondo è lo stesso che per un singolo disco.

Anche i RAID livello 4 e livello 5 lavorano in base alle *strip*, e non a singole parole con parità; inoltre non richiedono la sincronizzazione tra i dischi. Il RAID livello 4 [vedi Figura 2.23(e)] è come il RAID livello 0, con una parità *strip-per-strip* scritta su un disco aggiuntivo. Per esempio, se ogni *strip* è lunga  $k$  byte, si calcola l'OR ESCLUSIVO fra tutte, ottenendo una *strip* di parità lunga  $k$  byte. 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. Infatti, se si modifica un settore è necessario leggere tutti i dischi, in quanto occorre ricalcolare e riscrivere la parità. Una soluzione alternativa consiste nel leggere i precedenti dati e valori di parità e, a partire da questi, calcolare quelli nuovi. Anche ricorrendo a questa ottimizzazione una piccola modifica dei dati richiede comunque due letture e due scritture; un'organizzazione dei dischi di questo tipo è chiaramente inefficiente.

Il disco di parità può inoltre diventare un collo di bottiglia, a causa del grande carico di lavoro che pesa su di esso. Il RAID livello 5 elimina questo collo di bottiglia distribuendo uniformemente i bit di parità su tutti i dischi, in modalità *round robin*, com'è mostrato nella Figura 2.23(f). Tuttavia, quando si verifica un guasto a un disco, la ricostruzione del suo contenuto è un processo complesso.

### 2.3.6 Dischi a stato solido

I dischi basati su memoria flash non volatile, spesso chiamati dischi a stato solido (SSD), stanno diffondendosi sempre di più come un'alternativa ad alta velocità alle tradizionali tecnologie a disco magnetico. L'invenzione della SSD è una classica storia del tipo "Quando ti danno i limoni, fai una limonata". Anche se l'elettronica moderna può sembrare totalmente affidabile, in realtà i transistor si consumano lentamente con l'utilizzo: ogni volta che entrano in funzione si consumano un po' di più e si avvicina il momento della fine del loro ciclo di vita. Un guasto di cui può soffrire un transistor è dovuto alla cosiddetta "iniezione a caldo" (*hot carrier injection*), un meccanismo di avaria per cui una carica di elettroni viene rinchiusa all'interno del transistor, lasciandolo in uno stato permanente di blocco, acceso o spento. Nonostante questo meccanismo sia stato generalmente considerato come una condanna a morte per un (presunto) innocente transistor, Fujio Masuoka, mentre lavorava per Toshiba, ha scoperto un modo per sfruttarlo per creare una nuova memoria non volatile e, nei primi anni 1980, ha inventato la prima memoria flash.

I CD-R permettono ai privati e alle società di copiare facilmente i CD-ROM (e i CD audio), spesso in violazione dei diritti d'autore. Sono stati ideati vari schemi per ostacolare questa pirateria e far sì che sia difficile leggere un CD-ROM se non utilizzando il software fornito dall'autore dello stesso. Uno di questi sistemi prevede di registrare le lunghezze di tutti i file del CD-ROM come se fossero di vari gigabyte, ostacolando ogni tentativo da parte dei normali software di copiatura di trasferirli su hard disk. Le vere lunghezze sono memorizzate all'interno del software del produttore oppure sono nascoste (possibilmente criptate) in un'area inaspettata all'interno del CD-ROM. Un altro schema utilizza intenzionalmente ECC errati in specifici settori, con l'aspettativa che il programma di copiatura li "correggerà". Il software dell'applicazione controllerà anch'esso gli ECC e si rifiuterà di funzionare se essi sono corretti. Altre possibilità prevedono l'utilizzo di intervalli non standard tra le tracce e lo sfruttamento di altri "difetti" fisici.

### 2.3.9 CD-riscrivibili

Anche se abitualmente vengono usati supporti che permettono un'unica scrittura, come la carta o la pellicola fotografica, c'è una grande richiesta di CD-ROM riscrivibili. Una tecnologia oggi disponibile è quella dei CD-RW (*CD-ReWritable*, CD-Riscrivibili), che usano supporti della stessa dimensione dei CD-R. I CD-RW però, invece di usare la cianina o la ftalocianina per lo strato sul quale registrare, impiegano una lega di argento, indio, antimonio e tellurio. Questa lega ha due stati stabili, quello cristallino e quello amorfo, con diverse proprietà riflettenti.

I lettori CD-RW utilizzano laser che funzionano a tre potenze distinte. Alla potenza più elevata il laser scioglie la lega portandola dallo stato cristallino altamente riflettente a quello amorfo, dotato di una riflettività minore e che rappresenta un *pit*. Alla potenza media la lega si scioglie e si ricompone nello stato cristallino naturale per tornare a rappresentare un *land*. Alla potenza più bassa è possibile rilevare lo stato del materiale (per operazioni di lettura), senza però indurre alcuna trasformazione.

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, e non un difetto.

### 2.3.10 DVD

Il formato base dei CD/CD-ROM è stato definito attorno al 1980. Dalla metà degli anni '90 la tecnologia è notevolmente migliorata, al punto che oggi è possibile realizzare dischi ottici di maggiore capacità in modo economicamente conveniente. Nello stesso periodo, Hollywood stava cercando un modo per sostituire le videocassette analogiche con dischi a tecnologia ottica che offrissero una qualità più elevata, che fossero più economici da produrre, che durassero più a lungo, che occupassero meno spazio sugli scaffali dei negozi e che non dovessero essere riavvolti. Sembrava che per i dischi ottici la ruota del progresso stesse per girare ancora una volta.

Questa combinazione tra tecnologia e domanda proveniente da tre mercati estremamente ricchi e potenti ha portato al DVD, che originalmente era l'acronimo di *Digital Video Disk* (disco video digitale), mentre ora significa ufficialmente *Digital Versatile Disk* (disco digitale versatile). I DVD sono progettati in modo simile ai CD: si inietta policarbonato in uno stampo e sulla superficie si possono distinguere *pit* e *land* che vengono illuminati da un diodo laser e letti da un fotorilevatore. Le novità riguardano l'uso di:

1. *pit* più piccoli (0,4 μ contro gli 0,8 dei CD);
2. una spirale più stretta (0,74 μ di distanza tra le tracce rispetto agli 1,6 dei CD);
3. un laser rosso (a 0,65 μ invece che a 0,78 come nei CD).

Insieme, questi miglioramenti permettono una capacità sette volte maggiore, che arriva fino a 4,7 GB. Un DVD 1x ha una velocità di 1,4 MB/s (contro i 150 KB/s dei CD). Sfortunatamente il passaggio ai laser rossi, come quelli usati nei supermercati, obbliga i lettori DVD ad avere un secondo laser per poter leggere i CD e i CD-ROM esistenti, cosa che aggiunge un po' di complessità e aumenta i costi.

La capacità di 4,7 GB è sufficiente? Forse. Utilizzando la compressione MPEG-2 (standardizzata in IS 13346) un disco DVD da 4,7 GB può memorizzare 133 minuti di video a schermo intero in alta risoluzione (720 × 480), più di otto diverse colonne sonore e sottotitoli in 32 lingue diverse. Circa il 92% dei film che Hollywood ha finora prodotto ha una durata inferiore ai 133 minuti. Tuttavia, dato che alcune applicazioni, come i giochi multimediali o le encyclopedie, potrebbero richiedere quantità di dati più elevate e dato che Hollywood vorrebbe inserire più film sullo stesso disco, sono stati definiti quattro formati:

1. singolo lato, singolo strato (4,7 GB);
2. singolo lato, doppio strato (8,5 GB);
3. doppio lato, singolo strato (9,4 GB);
4. doppio lato, doppio strato (17 GB).

Perché così tanti formati? In una parola sola: politica. Philips e Sony volevano dischi a singolo lato e doppio strato per le versioni ad alta capacità, mentre Toshiba e Time Warner volevano dischi a doppio lato e doppio strato. Philips e Sony ritenevano che la gente non fosse disposta a girare i dischi per cambiar lato, mentre Time Warner non credeva che potesse funzionare l'utilizzo di due strati su un solo lato. Il compromesso fu quello di adottare tutte le soluzioni e di lasciare al mercato il compito di determinare quale sarebbe sopravvissuta. Alla fine il mercato emise il suo verdetto: Philips e Sony avevano ragione. Mai scommettere contro la tecnologia.

La tecnologia a doppio strato ha uno strato riflettente nella parte bassa, coperto da uno semiriflettente. A seconda del punto in cui il laser è messo a fuoco, esso rimbalza su un livello oppure sull'altro. 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.

I dischi a doppio lato sono creati prendendo due dischi a singolo lato da 0,6 mm di spessore e incollandoli l'uno contro l'altro, cioè "schiena contro schiena". Per far sì che tutte le versioni abbiano lo stesso spessore, un disco a singolo lato è composto da un disco spesso 0,6 mm incollato con un sottostrato vuoto (o forse, in futuro, con uno contenente 133 minuti di pubblicità, nella speranza che la gente sia curiosa e giri il disco per vedere che cosa c'è sull'altro lato). La Figura 2.28 illustra la struttura del disco a doppio strato e doppio lato.

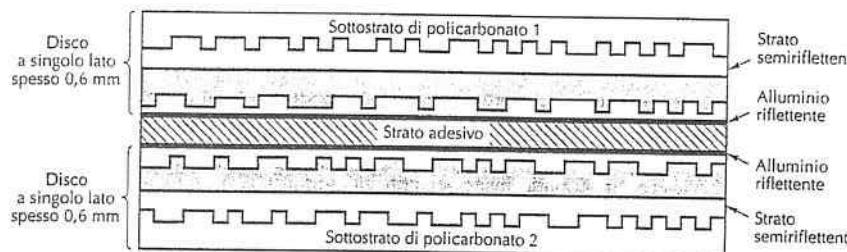


Figura 2.28 Disco DVD a doppio lato e doppio strato.

Il DVD è stato ideato da un consorzio di 10 aziende produttrici, sette delle quali giapponesi, in stretta collaborazione con i principali studi di produzione di Hollywood (alcuni dei quali sono di proprietà delle aziende giapponesi facenti parte del consorzio). Le società di computer e delle telecomunicazioni non furono invitate a questo "picnic" e l'attenzione finì per concentrarsi sull'utilizzo del DVD per il noleggio di film. Tra le funzionalità standard sono presenti, per esempio, l'omissione in tempo reale delle scene scabrose (per permettere ai genitori di rendere adatto ai bambini un film vietato ai minorenni), l'audio a sei canali e il supporto per il *Pan-and-Scan*. Quest'ultima funzionalità permette di decidere dinamicamente come tagliare le bande laterali dei film (il cui rapporto larghezza-altezza è di 3:2) per essere visualizzate sugli attuali schermi televisivi (che hanno un rapporto di forma di 4:3).

Un altro aspetto che molto probabilmente non sarebbe stato tenuto in conto dalle società di computer è la scelta intenzionale di rendere incompatibili i dischi previsti per gli Stati Uniti rispetto a quelli previsti per l'Europa, e anche per gli altri continenti. Hollywood ha preteso questa "caratteristica" poiché i nuovi film escono sempre prima negli Stati Uniti e vengono distribuiti in Europa solo quando negli Stati Uniti sono ormai in vendita i video.

L'idea era quella di impedire che i negozi di video europei potessero comprare i video negli Stati Uniti prima dell'uscita in Europa, riducendo quindi gli introiti delle sale cinematografiche. Se Hollywood fosse stata il motore del mercato dei calcolatori avremmo avuto floppy disk da 3,5 pollici negli Stati Uniti e floppy disk da 9 cm in Europa.

### 2.3.11 Blu-Ray

Nel mercato dei calcolatori nulla rimane fermo, tanto meno la tecnologia per la memorizzazione dei dati. Il DVD è stato appena introdotto e già il suo successore minaccia di renderlo obsoleto; si tratta del **Blu-Ray**, chiamato in questo modo poiché utilizza un laser blu invece di quello rosso dei DVD. I laser blu hanno una lunghezza d'onda più piccola di quelli rossi, il che permette una messa a fuoco più accurata e l'uso di *pit* e *land* più piccoli. I dischi Blu-Ray a singolo lato contengono circa 25 GB di dati; quelli a doppio lato 50 GB. La velocità di trasferimento dati è di circa 4,5 MB/s, buona per essere un disco ottico, ma ancora molto lontana da quella dei dischi magnetici (per esempio ATAPI-6 a 100 MB/s e Ultra5 SCSI a 640 MB/s). Ci si aspetta che alla fine Blu-Ray sostituirà i CD-ROM e i DVD, ma questo passaggio potrebbe richiedere degli anni.

## 2.4 Input/Output

Com'è stato detto all'inizio di questo capitolo, un calcolatore è composto da tre componenti principali: la CPU, le memorie e i dispositivi di I/O. Finora abbiamo analizzato la CPU e le memorie; adesso passiamo allo studio dei dispositivi di I/O e alla loro connessione con il resto del sistema.

### 2.4.1 Bus

Da un punto di vista fisico la struttura della maggior parte dei calcolatori e delle workstation è simile a quella mostrata nella Figura 2.29. L'organizzazione più comune consiste in una scatola metallica contenente una grande scheda di circuiti stampati nella parte bassa, chiamata **scheda madre** (scheda genitore, sembra più *politically correct*). La scheda madre contiene il chip della CPU, alcuni slot nei quali è possibile inserire i moduli DIMM e vari altri chip di supporto. Contiene inoltre un bus stampato lungo la sua lunghezza e alcune prese nelle quali è possibile inserire i connettori delle schede di I/O (il bus PCI). I PC più vecchi hanno anche un secondo bus (il bus ISA) per connettere le schede di I/O costruite molto tempo prima; di solito nei calcolatori più moderni questo bus non è presente e il suo utilizzo sta rapidamente scomparendo.

La struttura logica di un semplice personal computer di fascia bassa è mostrata nella Figura 2.30. Questo sistema ha un unico bus utilizzato per connettere la CPU, la memoria e i dispositivi di I/O, mentre la maggior parte dei sistemi ha invece due o più bus. Ciascun dispositivo di I/O è composto da due parti: una contenente la maggior parte dell'elettronica, chiamata il **controllore**, e un'altra contenente il dispositivo stesso, per esempio un lettore di dischi. Di solito il controllore è integrato direttamente sulla scheda madre, qualche volta è collocato su una scheda inserita in uno slot. Anche se lo schermo (il monitor) non è opzionale, il controllore video è talvolta situato su una scheda aggiuntiva per permettere all'utente di scegliere tra schede con o senza acceleratori hardware, memoria aggiuntiva e altre caratteristiche avanzate. Il controllore è connesso al suo dispositivo mediante un cavo che si collega al connettore nella parte posteriore della scatola metallica.

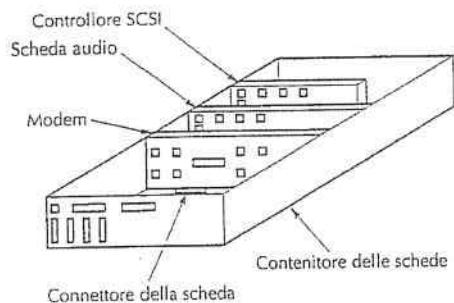


Figura 2.29 Struttura fisica di un personal computer.

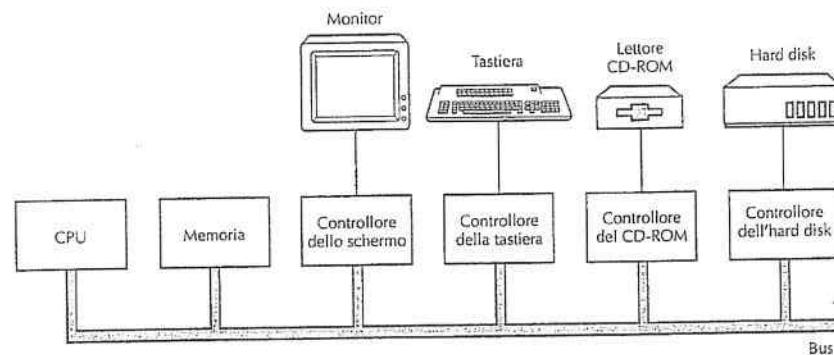


Figura 2.30 Struttura logica di un semplice personal computer.

Il compito di un controllore è di governare il proprio dispositivo di I/O e gestire il suo accesso al bus. Per esempio, quando un programma deve ottenere dei dati dal disco, manda un comando al suo controllore, che spedisce delle istruzioni al lettore, tra cui quella per la ricerca su disco. Il lettore, dopo aver individuato la traccia e il settore corretti, restituisce i dati al controllore sotto forma di un flusso seriale di bit. È compito del controllore spezzare il flusso di bit in unità, generalmente costituite da una o più parole, e scriverle successivamente in memoria. Quando un controllore legge e scrive dati in memoria senza l'intervento della CPU si dice che effettua un **Direct Memory Access** ("accesso diretto alla memoria"), meglio conosciuto con il suo acronimo DMA. Di solito, una volta che il trasferimento è completato, il controllore produce un **interrupt** che obbliga la CPU a sospendere immediatamente l'esecuzione del programma corrente e a far iniziare una speciale procedura, chiamata **gestore dell'interrupt (interrupt handler)** che controlla se ci sono errori, compie le azioni eventualmente necessarie e informa il sistema operativo che l'I/O è terminato. Dopo che il gestore dell'interrupt ha terminato

il proprio lavoro, la CPU riprende l'esecuzione del programma precedentemente sospeso all'arrivo dell'interrupt.

Il bus non è utilizzato solamente dai controllori di I/O, ma anche dalla CPU quando deve prelevare istruzioni o dati. Che cosa succede se la CPU e un controllore di I/O cercano di usare il bus allo stesso tempo? Un chip chiamato **arbitro del bus** stabilisce i turni. Generalmente le periferiche di I/O hanno la precedenza sulla CPU, dato che i dischi e altri dispositivi mobili non possono essere fermati e obbligarli ad aspettare vorrebbe dire perdere dati. Quando nessuna operazione di I/O è in esecuzione, la CPU può utilizzare tutti i cicli del bus per accedere alla memoria. Tuttavia quando un dispositivo di I/O è già in funzione, se richiede l'accesso al bus, questo gli viene subito concesso. Questo processo si chiama **furto di cicli** e rallenta le prestazioni del calcolatore.

Questo schema funzionava bene per i primi personal computer, dato che tutti i componenti erano più o meno bilanciati. Tuttavia, con l'aumento della velocità di CPU, memorie e dispositivi di I/O, sorse un problema: il bus non era più in grado di gestire il carico affidatogli. In un sistema chiuso, come le workstation adibite a calcoli ingegneristici, la soluzione fu quella di progettare un nuovo bus più veloce per il modello successivo. Questo approccio funzionava correttamente, dato che nessuno pensava di riutilizzare i vecchi dispositivi di I/O con il nuovo sistema.

Nel mondo dei PC però gli utenti aggiornavano spesso le loro CPU, ma allo stesso tempo volevano continuare a utilizzare stampanti, scanner e modem anche con il nuovo sistema. Era nato inoltre un enorme mercato attorno alla fornitura di periferiche di ogni tipo per il bus del PC IBM e non si voleva certo buttare via tutti gli investimenti per ricominciare da capo. IBM interpretò questo fatto nel modo peggiore quando lanciò il successore del PC IBM, la linea PS/2. Il PS/2 era dotato di un nuovo bus più veloce, ma molti produttori di cloni continuarono a utilizzare il bus dei vecchi PC, ora chiamato **bus ISA (Industry Standard Architecture, "standard industriale di architettura")**. Molti produttori di dischi e di altre periferiche continuarono a costruire controllori per quel bus e così IBM si trovò nella paradossale situazione di essere l'unico produttore di PC non più compatibili con IBM. Alla fine fu obbligata a tornare a supportare il bus ISA. Oggi il bus ISA è relegato a vecchi sistemi e musei dei computer ed è stato rimpiazzato da nuovi e più veloci standard. Aprendo una parentesi, occorre notare che ISA se utilizzato nel contesto dei livelli di una macchina significa **Instruction Set Architecture** ("architettura dell'insieme d'istruzioni"), mentre **Industry Standard Architecture** quando si parla di bus.

### I bus PCI e PCIe

Anche se il mercato esercitava pressioni affinché nulla venisse modificato, i vecchi bus divennero realmente troppo lenti. Questa situazione portò altre società a sviluppare macchine dotate di più bus, tra le quali c'era ancora il vecchio bus ISA o il suo successore retrocompatibile EISA (*Extended ISA, "ISA esteso"*). Alla fine il vincitore fu il bus PCI (*Peripheral Component Interconnect, "interconnessione di componenti periferici"*), progettato da Intel. Al fine di incoraggiare l'intero mercato (compresi i propri concorrenti) ad adottarlo, Intel decise però di rendere di dominio pubblico tutti i brevetti.

Il bus PCI può essere utilizzato in varie configurazioni, di cui la più comune è illustrata nella Figura 2.31. In questo schema la CPU comunica con il controllore della

memoria lungo una connessione dedicata ad alta velocità. Il controllore comunica con la memoria e con il bus PCI in modo diretto, per far sì che il traffico tra CPU e memoria non occupi il bus PCI. Altre periferiche sono connesse direttamente al bus PCI. Una macchina così progettata contiene di solito due o tre slot PCI liberi, per permettere all'utente di inserire le proprie schede PCI di I/O per nuove periferiche.

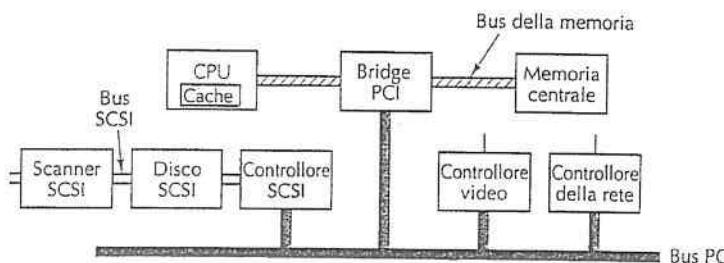


Figura 2.31 Tipico PC odierno con bus PCI. Il controllore SCSI è un dispositivo PCI.

Nel mondo dei computer non importa quanto veloce sia una cosa: un sacco di persone penseranno che è troppo lenta. Questo destino è toccato anche ai bus PCI, sostituiti con i bus PCIe. La maggioranza dei computer moderni supporta entrambi gli standard, così che gli utenti possano connettere dispositivi nuovi e veloci al bus PCIe e dispositivi più vecchi e più lenti al bus PCI.

Mentre quest'ultimo è stato solo un aggiornamento del vecchio ISA con velocità più alte e la possibilità di trasferire un maggior numero di bit in parallelo, il PCIe rappresenta un cambiamento radicale rispetto al PCI. In effetti non è nemmeno un bus, ma una rete punto a punto che utilizza linee seriali di bit e commutazione di pacchetto, più simile a Internet che a un bus tradizionale. L'architettura PCIe è rappresentata nella Figura 2.32.

Sono diverse le cose che balzano all'occhio riguardo al PCIe. Prima di tutto la connessione seriale tra le periferiche, ovvero un'ampiezza di 1 bit piuttosto che di 8, 16, 32 o 64. Anche se si può pensare che una connessione a 64 bit offra maggior larghezza di banda rispetto a una connessione a 1 bit, nella pratica le differenze nel tempo di propagazione di 64 bit, dette *skew*, portano a dover utilizzare velocità relativamente basse.

Nella comunicazione seriale possono invece essere usate velocità molto alte e questo compensa di gran lunga la mancanza di parallelismo. I bus PCI lavorano a una frequenza massima di 66 MHz; con il trasferimento di 64 bit per ciclo la massima velocità di trasferimento dati risulta di 528 MB/sec. Con un clock rate di 8Gbps, anche se la comunicazione è seriale, la velocità di trasferimento di PCIe è di 1GB/sec. Inoltre, la comunicazione tra i dispositivi e la root complex, o lo switch, non è limitata a una sola coppia di fili. Un dispositivo può disporre di 32 coppie di fili, chiamate corsie (*lanes*). Siccome le corsie non sono sincrone, lo skew non gioca alcun ruolo. La maggior parte delle schede madri ha uno slot a 16 corsie per la scheda grafica, che con l'uso di PCIe 3.0 offrono

alle schede video una larghezza di banda di 16 GB/sec, circa 30 volte la velocità che le schede video PCI potevano raggiungere. Questa velocità è necessaria per le esigenze sempre crescenti di alcune applicazioni, come il 3D.

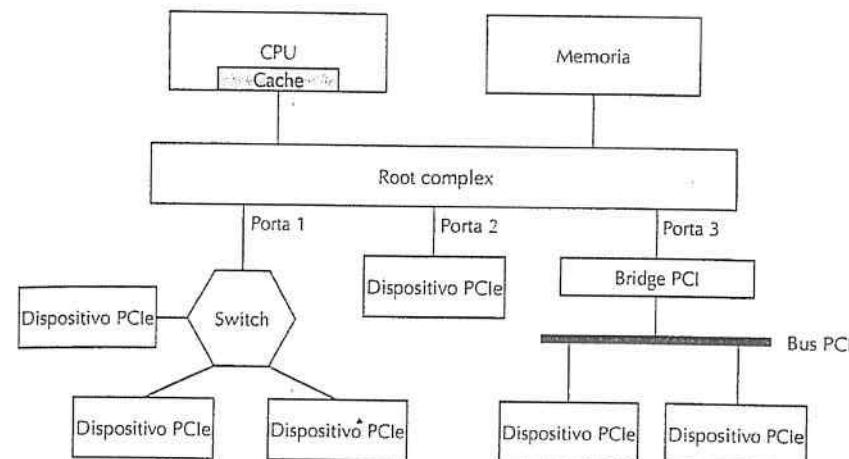


Figura 2.32 Esempio di architettura di un sistema PCIe con tre porte PCIe.

In secondo luogo, tutte le comunicazioni sono punto a punto. Quando la CPU deve comunicare con un dispositivo, invia un pacchetto a quel dispositivo dal quale riceve, generalmente, una risposta. Il pacchetto passa attraverso la root complex, che sta sulla scheda madre, e quindi va al dispositivo, talvolta passando per lo switch (o, se il dispositivo è PCI, per il bridge PCI). Questa evoluzione da un sistema in cui tutti i dispositivi restavano in ascolto sullo stesso bus a un sistema che utilizza una comunicazione punto a punto richiama lo sviluppo di Ethernet (una rete locale molto popolare), che agli inizi trasmetteva in broadcast e ora utilizza degli switch per permettere una comunicazione punto a punto.

## 2.4.2 Terminali

Diverse tipologie di dispositivi di I/O sono oggi disponibili. Alcune delle più comuni sono trattate in questo paragrafo. I terminali sono composti da due parti principali: la tastiera e il monitor. Nel mondo dei mainframe spesso questi componenti sono integrati in un unico dispositivo, collegato al calcolatore principale mediante una linea seriale o un cavo telefonico. Ancor oggi questi dispositivi sono ampiamente usati per le prenotazioni delle compagnie aeree, nelle banche e, più in generale, nei sistemi basati su mainframe. Nel mondo dei personal computer la tastiera e il monitor sono invece dei dispositivi indipendenti; la tecnologia è in ogni caso la stessa.

### Tastiere

Esistono vari tipi di tastiere. Il PC IBM originario era dotato di una tastiera che aveva, sotto ogni tasto, un pulsante a scatto che dava un feedback tattile e produceva un clic quando il tasto veniva premuto sufficientemente. Oggi i tasti delle tastiere più economiche hanno semplicemente dei contatti meccanici, mentre quelle di qualità più elevata hanno un foglio di materiale elastometrico (una specie di gomma) tra i tasti e il circuito stampato sottostante. In questi modelli vi è, sotto ogni tasto, una piccola calotta che si piega quando il tasto viene premuto a sufficienza; una minuscola regione di materiale conduttivo presente al suo interno permette di chiudere il circuito. In alcune tastiere, la pressione di un tasto fa scorrere un piccolo corpo metallico all'interno di una bobina conduttrice inducendo in essa una corrente rilevabile. Vengono utilizzati anche vari altri metodi, di tipo meccanico o elettromagnetico.

La pressione di un tasto di un PC genera un interrupt che stimola una parte del sistema operativo, chiamata *gestore dell'interrupt della tastiera*. Questa routine, leggendo un registro hardware della tastiera, ricava il numero (da 1 a 102) associato al tasto premuto. Anche quando si rilascia un tasto viene generato un interrupt. In questo modo se l'utente preme il tasto SHIFT, poi preme e rilascia il tasto M e infine rilascia il tasto SHIFT, il sistema operativo può capire che l'utente voleva digitare una "M" maiuscola invece che una "m" minuscola. La gestione di sequenze multi-tasto che coinvolgono i tasti SHIFT, CTRL e ALT è fatta interamente via software (compresa la famosa e detestata sequenza CTRL-ALT-CANC utilizzata per riavviare i PC).

### Touch screen

Nonostante le tastiere non corrano alcun rischio di fare la fine delle macchine da scrivere, c'è un nuovo arrivato nel mondo dei dispositivi di input: il touch screen. Anche se questi dispositivi sono entrati nel mercato di massa con l'avvento degli iPhone di Apple nel 2007, la loro storia inizia molto prima. Il primo touch screen è stato sviluppato al Royal Radar Establishment a Malvern, Regno Unito, nel 1965. Anche la tanto proclamata possibilità di pinching dell'iPhone risale in realtà a un lavoro fatto nell'Università di Toronto nel 1982. Da allora, sono state sviluppate e commercializzate diverse tecnologie. I dispositivi tattili possono essere classificati in due gruppi: gli opachi e i trasparenti. Un tipico esempio di dispositivo tattile opaco è il touchpad di un computer portatile. Un tipico dispositivo trasparente è lo schermo di uno smartphone o di un tablet. Considereremo qui solo quest'ultimo tipo. Tali dispositivi sono noti come touch screen. I touch screen più diffusi sono di tipo resistivo, capacitivo o a infrarossi.

Gli schermi a infrarossi hanno dei trasmettitori di raggi infrarossi, come diodi a emissione di luce infrarossa o laser, sul lato sinistro per esempio e sul lato superiore della smussatura attorno allo schermo e dei ricevitori sul lato destro e sul lato inferiore. Quando un dito, una penna o un qualunque oggetto opaco blocca uno o più fasci di luce il ricevitore corrispondente rileva la caduta del segnale e l'hardware del dispositivo comunica al sistema operativo quali fasci sono stati interrotti, permettendogli così di calcolare le coordinate del dito o della penna. Anche se questi dispositivi hanno una lunga tradizione e sono ancora usati in alcune applicazioni, non vengono utilizzati per i dispositivi mobili.

Un'altra vecchia tecnologia è quella dei **touch screen resistivi**. Questi sono formati da due strati, di cui il superiore è flessibile e contiene un gran numero di fili orizzontali, mentre quello inferiore contiene i fili verticali. Quando un dito o un altro oggetto fa pressione su un punto dello schermo, uno o più fili dello strato superiore viene a contatto con i quelli del livello inferiore, oppure vi si avvicina. L'elettronica del dispositivo fa sì che si possa localizzare l'area che ha subito la depressione. La produzione di questi schermi è molto economica, e sono quindi molto utilizzati in ambiti applicativi particolarmente sensibili ai costi.

Le due tecnologie fin qui presentate funzionano bene quando lo schermo è premuto da un solo dito, ma hanno problemi in caso di utilizzo con due dita. Descriveremo il problema per gli schermi a infrarossi, ma lo stesso problema si ha per gli schermi resistivi. Si immagini che due dita siano posizionate alle coordinate (3,3) e (8,8). In questa condizione risultano interrotti i fasci di luce orizzontali  $x=3$  e  $x=8$  e i fasci di luce verticali  $y=3$  e  $y=8$ . Si consideri ora la situazione, differente dalla precedente, in cui le dita sono posizionate nei punti (3,8) e (8,3), ovvero ai vertici opposti del rettangolo (3,3), (8,3), (8,8), (3,8). In questo caso risultano interrotti esattamente gli stessi fasci e il software non può in alcun modo riconoscere quale dei due casi descritti si sia verificato. Questo problema è noto come *ghosting*.

Per poter riconoscere la pressione contemporanea di più dita, una proprietà richiesta per l'utilizzo di gesti come espansione e pizzicamento (pinching), si ricorre a un'altra tecnologia. Quella utilizzata in gran parte degli smartphone e dei tablet (ma non nelle macchine fotografiche e in altri dispositivi) è il touch screen capacitivo a proiezione. Ci sono diverse tipologie di questi schermi, ma il tipo più diffuso è quello a mutua capacità. Tutti i touch screen in grado di riconoscere due o più punti di contatto contemporanei sono detti schermi multitouch. Vediamo brevemente come funzionano.

Ricordiamo che un condensatore è un dispositivo in grado di immagazzinare carica elettrica. Un semplice condensatore è formato da due conduttori separati da un isolante. Nei touch screen moderni, una griglia di "fili" sottili disposti verticalmente è separata da una griglia orizzontale per mezzo di un sottile strato isolante. Quando un dito tocca lo schermo modifica la capacità in tutte le intersezioni toccate (anche distanti) e questo cambiamento può essere misurato. Per verificare che i moderni touch screen sono diversi dai più datati schermi resistivi o a infrarossi, basta provare a toccarli con una penna, una matita, una graffetta o un dito ricoperto da un guanto per scoprire che non succede nulla. Il corpo umano è in grado di immagazzinare cariche elettriche, come chiunque abbia sfregato un tappeto in una giornata fredda e secca e quindi toccato una maniglia di metallo può dolorosamente testimoniare. Gli oggetti in plastica, legno e metallo non godono delle stesse proprietà.

I "fili" in un touch screen non sono i tradizionali fili di rame che si trovano nei normali dispositivi elettrici, che bloccherebbero la luce dalla schermata, ma sono sottili (tipicamente 50 micron) strisce di ossido di indio e stagno trasparenti, legate ai lati opposti di una piastra di vetro sottile, insieme alla quale formano i condensatori. In alcuni modelli più recenti, la lastra di vetro isolante viene sostituita da un sottile strato di diossido di silicio (sabbia!), con i tre strati atomizzati (spruzzati, atomo per atomo) su un substrato. In entrambi i casi, i condensatori sono protetti da polvere e graffi da una

lastra di vetro sovrapposta che forma la superficie dello schermo da toccare. Più sottile è la lastra di vetro superiore e più sensibile è lo schermo, ma la fragilità del dispositivo aumenta.

Durante il funzionamento, le tensioni vengono applicate alternativamente ai “fili” orizzontali e verticali mentre i valori di tensione, che sono influenzati dalla capacità di ciascuna intersezione, vengono letti lungo l'altra direzione. Questa operazione è ripetuta diverse volte al secondo, con le coordinate toccate che alimentano il driver del dispositivo con un flusso di coppie (x, y). Ulteriori operazioni, per esempio determinare un puntamento, un pizzicamento, un'espansione o una strisciata in atto, sono gestite dal sistema operativo. Se si utilizzano tutte e dieci le dita e si invita un amico ad aggiungere altri tocchi il sistema operativo potrebbe essere incapace di gestire l'operazione, anche se l'hardware del multitouch sarà all'altezza del compito.

### Schermi piatti

I primi schermi dei computer utilizzavano tubi catodici (CRT) esattamente come le vecchie televisioni. Questi schermi erano troppo ingombranti e pesanti per poter essere utilizzati nei computer portatili e così si rese necessario sviluppare una nuova tecnologia: quella degli schermi piatti, in grado di offrire il fattore di forma adatto ai notebook e di consumare meno potenza. Al giorno d'oggi, i benefici in termini di consumo elettrico e dimensioni offerti dagli schermi piatti hanno soppiantato la tecnologia CRT.

La tecnologia più comune è quella dello schermo a cristalli liquidi, LCD (*Liquid Crystal Display*). La sua descrizione verrà fortemente semplificata, dato che si tratta di una tecnologia in continuo mutamento e particolarmente complessa, di cui esistono molte varianti.

I cristalli liquidi sono molecole organiche viscose che si muovono come un liquido, ma che hanno anche una struttura spaziale simile a quella di un cristallo. Scoperti nel 1888 da un botanico austriaco (Friedrich Reinitzer), negli anni '60 sono stati applicati per la prima volta agli schermi (per esempio in quelli di calcolatrici e orologi). Quando tutte le molecole sono allineate nella stessa direzione le proprietà ottiche del cristallo dipendono dalla direzione e dalla polarizzazione della luce incidente. È possibile modificare l'allineamento delle molecole, e quindi le proprietà ottiche, tramite l'applicazione di un campo elettrico. In particolare è possibile controllare elettricamente l'intensità della luce uscente facendola passare attraverso un cristallo liquido. Questa proprietà può essere sfruttata per costruire schermi piatti.

Uno schermo LCD consiste in due lastre di vetro parallele tra le quali è sigillato un cristallo liquido. Alle lastre sono attaccati degli elettrodi trasparenti. Una luce (naturale o artificiale) proveniente da dietro la lastra posteriore illumina lo schermo. Gli elettrodi trasparenti collegati a ciascuna lastra sono utilizzati per creare campi elettrici all'interno del cristallo liquido e, variando la tensione sulle diverse parti dello schermo, si può controllare l'immagine da rappresentare. Per la visualizzazione sono anche necessari dei filtri di polarizzazione incollati allo schermo frontale e a quello posteriore. Lo schema generale è mostrato nella Figura 2.33(a).

Anche se vengono usati diversi tipi di schermi LCD, ne considereremo come esempio solo uno: lo schermo TN (*Twisted Nematic*). In questo schermo la lastra posteriore con-

tiene piccoli solchi orizzontali, mentre in quella frontale i solchi sono disposti verticalmente, come mostra la Figura 2.33(b). In assenza di campo elettrico le molecole del LCD tendono ad allinearsi lungo i solchi e, dato che gli allineamenti sulle due lastre differiscono di 90 gradi, le molecole (e quindi la struttura del cristallo) effettuano una torsione nello spazio intermedio.

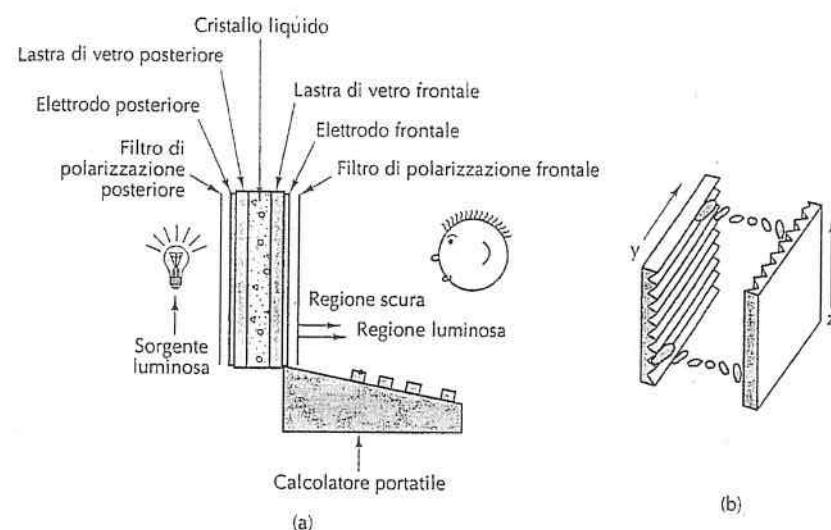


Figura 2.33 (a) Struttura di uno schermo LCD. (b) I solchi sulla lastra frontale e su quella posteriore sono perpendicolari tra loro.

Dietro allo schermo è collocato un filtro di polarizzazione orizzontale, che permette esclusivamente il passaggio di luce polarizzata in quella direzione, mentre nella parte frontale c'è un filtro di polarizzazione verticale. Se tra le lastre non ci fosse del liquido, la luce polarizzata orizzontalmente lasciata passare dal filtro polarizzatore posteriore sarebbe bloccata da quello anteriore, rendendo lo schermo uniformemente nero.

La torsione della struttura del cristallo delle molecole guida la luce nel suo passaggio e ruota la sua polarizzazione rendendola verticale. Quindi, in assenza di un campo elettrico, lo schermo LCD risulta uniformemente luminoso. Applicando tensione a particolari regioni della lastra si distrugge la struttura cristallina e la sua torsione, bloccando localmente il passaggio della luce.

È possibile utilizzare due schemi per l'applicazione della tensione. Negli schermi a matrice passiva (i più economici) entrambi gli elettrodi contengono fili paralleli. Per esempio, in uno schermo 1920 x 1080 (la dimensione del Full HD) l'elettrodo posteriore potrebbe avere 1920 fili verticali, e quello frontale 1080 orizzontali. Applicando una tensione a uno dei cavi verticali e un impulso a uno dei cavi orizzontali si modifica il

voltaggio di un pixel in una particolare posizione, facendolo diventare scuro per un breve intervallo di tempo. Il pixel (da *picture element*, ovvero elemento del disegno), è il punto colorato a partire dal quale si realizzano tutte le immagini digitali. Applicando lo stesso impulso al pixel successivo e poi a quello ancora seguente si può disegnare sullo schermo una linea scura. Solitamente lo schermo viene interamente ridisegnato 60 volte in un secondo, in modo da ingannare l'occhio e fargli credere che stia guardando un'immagine costante.

L'altro schema utilizzato estensivamente fa uso di schermi a matrice attiva: è più costoso, ma produce un'immagine di qualità superiore. In questo caso, invece di avere solamente due insiemi di fili paralleli, su uno degli elettrodi è presente, in corrispondenza di ciascun pixel, un piccolo elemento di commutazione. Accendendo o spegnendo questi elementi si può creare sullo schermo uno schema di tensioni, che corrisponde a un arbitrario pattern di bit. Gli elementi di commutazione sono chiamati *thin film transistor* ("transistor a pellicola sottile") e gli schermi piatti che li utilizzano sono spesso chiamati *schermi TFT*. Al giorno d'oggi la maggior parte dei calcolatori portatili e dei monitor a schermo piatto per computer desktop utilizza la tecnologia TFT.

Finora abbiamo descritto il funzionamento di uno schermo monocromatico. Anche se i dettagli sono molto più complicati, è sufficiente dire che gli schermi a colori si basano sugli stessi principi. Si usano filtri ottici per separare, in corrispondenza di ogni pixel, la luce bianca nelle componenti rossa, verde e blu, in modo da poterle visualizzare indipendentemente.

Nuove tecnologie di schermi sono all'orizzonte. Una delle più promettenti è lo schermo OLED (*Organic Led Emitting Diode*), che consiste in uno strato di molecole organiche caricate elettricamente schiacciate tra due elettrodi. Le variazioni di tensione eccitano le molecole e le portano in stati energetici più alti. Quando le molecole tornano allo stato normale emettono luce. Fornire maggiori dettagli su questa tecnologia va oltre gli scopi di questo libro (e dei suoi autori).

### RAM della scheda video

La maggior parte dei monitor vengono ridisegnati dalle 60 alle 100 volte al secondo, accedendo a una memoria speciale, chiamata RAM della scheda video (*Video RAM*), che si trova sulla scheda del controllore dello schermo. Questa memoria ha una o più bitmap che rappresentano l'immagine dello schermo; per esempio su uno schermo di 1920×1080 pixel la RAM della scheda video deve contenere 1920×1080 valori, uno per ogni pixel. In realtà, per permettere un veloce passaggio da un'immagine dello schermo a un'altra, questa memoria potrebbe contenere più bitmap di questo tipo.

Su un'ordinario schermo ogni pixel dovrebbe essere rappresentato da un valore RGB (da *Red*, *Green* e *Blue*) composto da 3 byte, uno per l'intensità di ciascun componente (rosso, verde e blu) del colore del pixel (gli schermi di fascia alta utilizzano 10 o più bit per colore). Le leggi dell'ottica ci dicono che qualsiasi colore può essere costruito combinando linearmente luci rosse, verdi e blu.

Una RAM della scheda video con 1920×1080 pixel e 3 byte per pixel richiede oltre 6,2 MB per memorizzare l'immagine e una quantità non indifferente di tempo di CPU per operare su di essa. Per questa ragione alcuni calcolatori scendono al compromesso

di utilizzare un unico numero a 8 bit per indicare il colore desiderato. Questo numero è utilizzato come indice di una tabella hardware, chiamata tavolozza (*color palette*), contenente 256 elementi, ciascuno dei quali memorizza un valore RGB a 24 bit. Questo tipo di strategia, chiamata colore indicizzato, riduce di 2/3 i requisiti della RAM della scheda video, ma permette di visualizzare su schermo solo 256 colori per volta. Spesso, quando sullo schermo sono mostrate contemporaneamente più finestre, solo i colori di una di queste vengono *renderizzati* correttamente; infatti, anche se di solito ogni finestra ha la sua propria mappatura, esiste una sola tavolozza hardware. Vengono anche utilizzate tavolozze da 2<sup>16</sup> colori, ma in questo caso il guadagno si riduce a 1/3.

Gli schermi basati su bitmap richiedono una grande quantità di larghezza di banda.

Per visualizzare dati multimediali a schermo intero su uno schermo 1920×1080 occorre trasferire a ogni fotogramma 6,2 MB di dati. Nel caso di video in movimento (*full-motion*) sono necessari almeno 25 fotogrammi al secondo, che corrispondono a una velocità totale di trasferimento dati di 155 MB/s. Questo carico supera le capacità di un bus PCI (132 MB/sec), ma il bus PCIe è in grado di gestirlo tranquillamente.

### 2.4.3 Mouse

Più il tempo passa, più i calcolatori sono utilizzati da persone con sempre minor conoscenza del loro funzionamento. I calcolatori della generazione ENIAC erano usati da coloro che li avevano costruiti, mentre negli anni '50 i computer venivano utilizzati soltanto da programmatore altamente specializzati. Oggigiorno invece i calcolatori sono largamente impiegati da persone che non sanno (o talvolta non vogliono sapere) troppi dettagli sul funzionamento dei calcolatori o sulla loro programmazione.

Agli inizi, la maggior parte dei calcolatori era dotata di un'interfaccia a linea di comando, attraverso la quale gli utenti impartivano i comandi. Dato che chi non è uno specialista dei calcolatori reputa che le interfacce a linea di comando siano spesso poco amichevoli, se non addirittura ostili, molti produttori di calcolatori, in seguito, hanno sviluppato interfacce punta-e-clicca, come quelle del Macintosh e di Windows. Per poter utilizzare questo tipo d'interfaccia è necessario avere uno strumento per puntare sullo schermo, e il metodo più comune consiste nell'usare un mouse.

Il mouse è un piccolo oggetto di plastica situato a fianco della tastiera. Quando viene mosso sul tavolo, un piccolo puntatore si sposta, in corrispondenza, sullo schermo, permettendo agli utenti di puntare agli elementi desiderati. Il mouse, sulla parte superiore, ha uno, due o tre pulsanti che consentono all'utente di selezionare elementi e voci dei menu. Gli utenti più inesperti preferiscono i mouse con un unico tasto (è infatti difficile premere quello sbagliato se ce n'è soltanto uno), mentre quelli più abili preferiscono la potenza di un mouse a più tasti che permette azioni più sofisticate.

Sono stati prodotti tre tipi di mouse: meccanici, ottici e opto-meccanici. Il primo mouse aveva sul fondo due rotelle di gomma che sporgevano all'esterno; gli assi delle due rotelle erano disposti perpendicolarmente, di modo che, quando il mouse veniva spostato parallelamente a uno degli assi principali, ruotava soltanto una rotella. Ciascuna rotella guidava una resistenza variabile o un potenziometro, in modo che misurando il loro cambiamento era possibile calcolare la rotazione e, di conseguenza, la distanza percorsa dal mouse lungo una particolare direzione. Negli ultimi anni questo tipo di

mouse è stato quasi completamente sostituito da un modello, mostrato nella Figura 2.34, che al posto delle due rotelle fa uso di una pallina leggermente sporgente sul fondo.

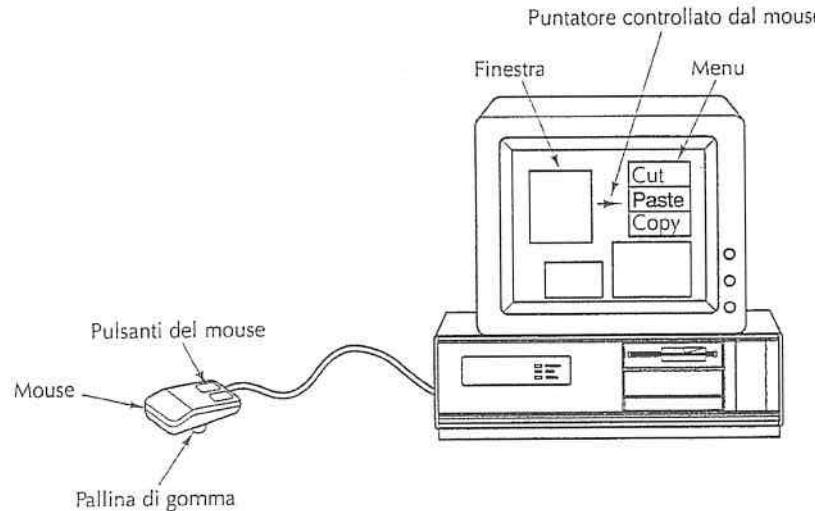


Figura 2.34 Mouse utilizzato per puntare a voci di un menu.

Il secondo tipo di mouse è quello ottico, che non ha né rotelle né pallina. Al loro posto ha, sul fondo, un LED (*Light Emitting Diode*, “diodo luminescente”) e un fotorilevatore. I primi mouse ottici richiedevano un mouse pad in cui era raffigurata una griglia di linee molto vicine tra loro per determinare quante linee venivano attraversate e quindi l’entità dello spostamento del mouse. I mouse ottici moderni contengono un LED che illumina le imperfezioni sottostanti e una sottile fotocamera che registra piccole immagini (di solito di 18x18 pixel) circa 1000 volte al secondo. Le immagini consecutive vengono quindi analizzate per determinare di quanto il mouse si è eventualmente spostato. Alcuni mouse utilizzano un laser al posto del LED e sono più precisi, ma anche più costosi.

Il terzo tipo di mouse è quello opto-mecanico, anch’esso dotato di una pallina che fa ruotare due cilindretti perpendicolari tra loro. I cilindretti 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.

Anche se sono possibili diverse soluzioni, in genere un mouse spedisce al calcolatore una sequenza di 3 byte ogni qualvolta si muove di una certa distanza minima (per esempio 0,01 pollici), chiamata in alcuni casi *mickey*. Di solito questi caratteri vengono spediti lungo una linea seriale, un bit alla volta. Il primo byte contiene un intero con segno che

indica quante unità sono state percorse dal mouse nella direzione x rispetto all’ultima rilevazione, mentre il secondo byte fornisce la stessa informazione per quanto riguarda il movimento lungo y; il terzo byte contiene invece lo stato corrente dei pulsanti del mouse. A volte si utilizzano 2 byte per i valori di spostamento di ogni coordinata.

Il software del calcolatore riceve queste informazioni sui movimenti relativi del mouse, le converte in coordinate assolute e, in corrispondenza della posizione calcolata, visualizza una piccola freccia. Per selezionare un elemento sullo schermo l’utente deve soltanto posizionare la freccia sulla regione desiderata e premere un pulsante: il calcolatore può determinare quale elemento è stato selezionato dato che sa in ogni momento in che posizione dello schermo si trova la freccia.

#### 2.4.4 Controller per videogiochi

Grazie alle molteplici esigenze degli utilizzatori di videogiochi il mercato delle console di gioco ha sviluppato dispositivi di input specializzati. In questo paragrafo consideriamo due recenti sviluppi tra i controller per videogiochi: in Nintendo Wiimote e il Microsoft Kinect.

##### Il controller Wiimote

Apparso nel 2006 insieme alla console Nintendo Wii, il controller Wiimote aggiunge ai tradizionali pulsanti del gamepad la capacità di rilevare i movimenti. Tutte le interazioni con il Wiimote sono inviate in tempo reale alla console di gioco tramite una sistema bluetooth interno. I sensori di movimento presenti in Wiimote gli permettono di rilevare movimenti in tre dimensioni. Inoltre, se rivolto verso il televisore, il controller è in grado di offrire una precisa funzione di puntamento.

La Figura 2.35 mostra come il Wiimote implementa la funzione di rilevazione dei movimenti. Il monitoraggio dei movimenti del Wiimote lungo tre direzioni è realizzato da un accelerometro a tre assi. Questo dispositivo contiene tre piccole masse, ciascuna in grado di muoversi lungo uno dei tre assi x, y e z (rispetto alla posizione del chip dell’accelerometro). Il movimento di ogni massa è proporzionale all’accelerazione lungo il proprio asse e modifica la capacità della massa rispetto a una parete fissa di metallo. Misurando le tre variazioni di capacità diventa possibile rilevare l’accelerazione lungo le tre direzioni. Grazie a questa tecnologia e alla matematica classica, la console Wii può seguire i movimenti del Wiimote nello spazio. Mentre si tenta di colpire con il Wiimote una pallina da tennis virtuale, il movimento del controller viene monitorato; se si ruota il polso all’ultimo momento per cercare di dare un effetto alla pallina, l’accelerometro del Wiimote sarà in grado di rilevare questo ulteriore movimento.

Gli accelerometri danno buoni risultati nella rilevazione dei movimenti tridimensionali del Wiimote, ma non possono offrire la precisione necessaria per controllare un puntatore sullo schermo televisivo. Gli accelerometri soffrono infatti di un’inevitabile margine di errore nella rilevazione dei movimenti. Con il passare del tempo, il calcolo della posizione esatta del Wiimote (basato sull’integrazione delle sue accelerazioni) diventa sempre meno accurato.

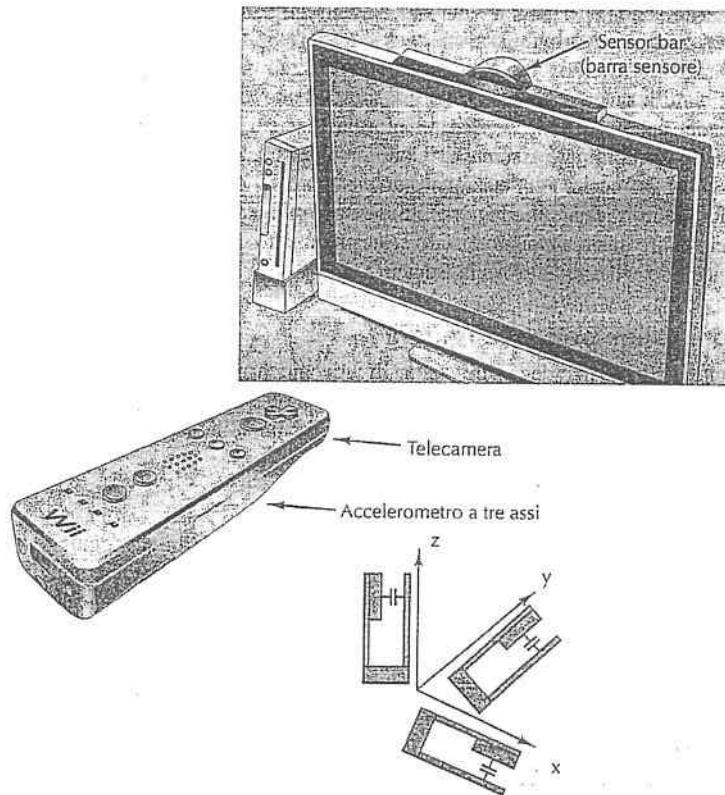


Figura 2.35 I sensori di movimento del controller Wiimote.

Per offrire una rilevazione di movimento più fine il Wiimote utilizza tecnologie intelligenti di computer vision. Posta sopra il televisore vi è una *sensor bar* (“barra sensore”) che contiene dei LED a distanza fissata. Il Wiimote contiene una videocamera che, se puntata verso il televisore, è in grado di dedurre la distanza e l’orientamento del controller rispetto allo schermo. Visto che i LED della sensor bar sono a distanza fissa, la loro distanza vista dal Wiimote è proporzionale alla distanza del controller dalla sensor bar. La posizione della sensor bar nel campo visivo del Wiimote indica la direzione da cui il controller sta puntando il televisore. Monitorando continuamente questo orientamento è possibile ottenere una capacità di puntamento preciso senza soffrire degli errori degli accelerometri.

#### Il controller Kinect

Microsoft Kinect porta le capacità di computer vision dei controller per videogiochi a un nuovo livello. Questo dispositivo utilizza solamente tecniche di computer vision per

determinare le interazioni dell’utente con la console di gioco. Il suo funzionamento si basa sulla rilevazione della posizione dell’utente nella stanza, sul suo orientamento e sui movimenti del suo corpo. Il controllo dei giochi avviene mediante movimenti predeterminati effettuati dalle mani, dalle gambe e da qualsiasi parte del corpo che il programmatore del videogioco vuole far muovere per il controllo del gioco.

La capacità di rilevazione di Kinect si basa su una fotocamera di profondità e una video camera. La camera di profondità misura la distanza dell’oggetto nel campo visivo di Kinect emettendo un fascio di raggi laser infrarossi e catturando poi i loro riflessi grazie a una camera a infrarossi. Usando una tecnica di computer vision nota come “structured light”. Kinect può determinare la distanza degli oggetti che vede basandosi sul disturbo che il fascio infrarosso subisce dalla superficie illuminata.

L’informazione sulla profondità viene combinata con le informazioni strutturali restituite dalla videocamera per produrre una mappa di profondità della struttura. Questa mappa può quindi essere processata da un algoritmo di computer vision per localizzare le persone nella stanza (riconoscendo persino il loro volto) e l’orientamento e i movimenti del loro corpo. Dopo questa fase le informazioni sulle persone presenti nella stanza vengono inviate alla console di gioco che utilizza i dati raccolti per il controllo del videogioco.

#### 2.4.5 Stampanti

Dopo aver preparato un documento o scaricato una pagina da Internet, gli utenti vogliono spesso stampare il lavoro fatto. Per questa ragione molti computer dispongono di una stampante. In questo paragrafo descriveremo alcune delle più diffuse tipologie di stampanti.

##### Stampanti laser

Dopo l’invenzione della stampa, da parte di Johann Gutenberg nel quindicesimo secolo, lo sviluppo più significativo nella riproduzione dei testi è probabilmente rappresentato dalla stampante laser. Questo dispositivo combina in un’unica periferica alta qualità dell’immagine, eccellente flessibilità, grande velocità e costo limitato. Le stampanti laser usano una tecnologia molto simile a quella delle fotocopiatrici; infatti molte società producono periferiche che permettono sia di effettuare fotocopie sia di stampare (e talvolta di spedire e ricevere fax).

La Figura 2.36 illustra la tecnologia di base. Il cuore della stampante è un tamburo rotante molto preciso (o, in alcuni sistemi di fascia alta, un nastro). 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 si riflette su 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. Il toner è attratto 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.

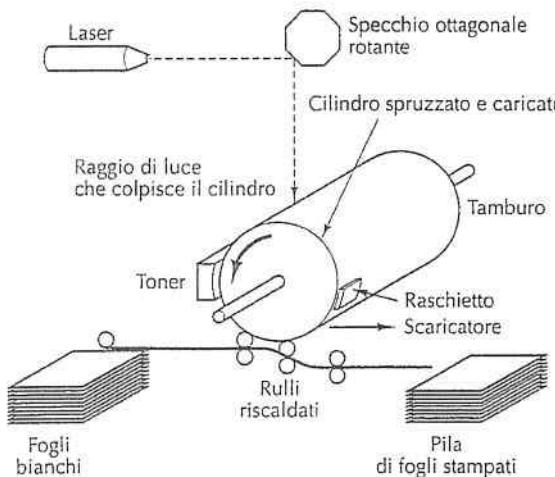


Figura 2.36 Funzionamento di una stampante laser.

È necessario precisare che questo processo è possibile grazie a una combinazione estremamente complessa di fisica, chimica, meccanica e ottica. Nonostante questa complessità alcuni produttori vendono dei meccanismi completi, chiamati *motori di stampa*, che comprendono tutte queste fasi. Per realizzare una stampante completa i produttori di stampanti laser combinano i motori di stampa con la propria elettronica e il proprio software. L'elettronica consiste in una veloce CPU integrata, alcuni megabyte di memoria per memorizzare la mappa dei bit di un'intera pagina e numerosi set di caratteri, alcuni dei quali predefiniti.

La maggior parte delle stampanti accetta comandi che descrivono come devono essere stampate le pagine. Questi comandi sono espressi in linguaggi come PCL di HP, PostScript di Adobe o PDF, dei veri e propri linguaggi di programmazione completi, anche se specializzati.

Le stampanti laser da 600 dpi e più possono stampare in modo accettabile fotografie in bianco e nero, ma la tecnologia è più complessa di quanto potrebbe sembrare. Si consideri una fotografia scannerizzata a 600 dpi che deve essere stampata con una stampante a 600 dpi. L'immagine scannerizzata contiene  $600 \times 600$  pixel/pollice, ciascuno dei

quali consiste in un valore di grigio compreso fra 0 (bianco) e 255 (nero). Anche la stampante può stampare a 600 dpi, ma ciascun pixel stampato è o nero (*toner* presente) o bianco (*toner* assente); i grigi non possono essere stampati.

Per riprodurre la scala dei grigi la tecnica più comune è quella dei *mezzitoni*, la stessa utilizzata per i poster commerciali. L'immagine è divisa in celle, solitamente di  $6 \times 6$  pixel: ogni cella può quindi contenere tra 0 e 36 pixel neri. Le celle con più pixel sono percepite dall'occhio come più scure rispetto alle altre. I valori di grigio compresi tra 0 e 255 sono rappresentati dividendo questo intervallo in 37 zone. I valori da 0 a 6 appartengono alla zona 0, i valori da 7 a 13 alla zona 1 e così via (la zona 37 è leggermente più piccola delle altre in quanto 256 non è divisibile esattamente per 37). Ogni qualvolta si incontra un grigio appartenente alla zona 0, si lascia la sua cella dei mezzitoni bianca, come mostra la Figura 2.37(a). Un valore appartenente alla zona 1 è stampato con un pixel nero; uno della zona 2 con due pixel neri, come mostra la Figura 2.37(b). Le Figure 2.37(c)-(f) mostrano altre zone con valori diversi. Se si prende una fotografia scannerizzata a 600 dpi e la si rappresenta mediante questa tecnica, la sua risoluzione effettiva si riduce ovviamente a 100 celle/pollice; questa risoluzione viene chiamata *frequenza del retino dei mezzitoni*, ed è misurata convenzionalmente in *lpi* (*lines per inch*, "linee per pollice").

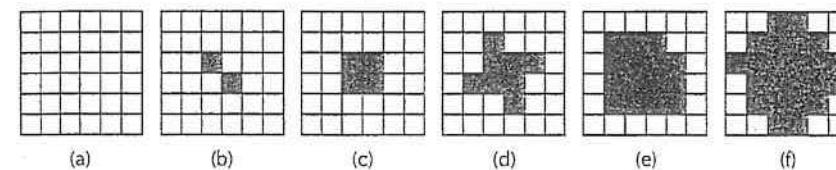


Figura 2.37 Mezzitoni per alcuni intervalli di livelli di grigio. (a) 0-6. (b) 14-20. (c) 28-34. (d) 56-62. (e) 105-111. (f) 161-167.

### La stampa a colori

Sebbene la maggior parte delle stampanti laser sia monocromatica, le stampanti laser a colori stanno diventando sempre più comuni. Per questa ragione riteniamo sia utile dare qualche spiegazione di questa tecnica (valida anche per stampanti a getto d'inchiostro e per altre stampanti). Come potete immaginare, non è una tecnica banale. Le immagini a colori vengono viste in due modi distinti: per luce trasmessa o per luce riflessa. Le immagini per luce trasmessa, come quelle prodotte sui monitor, sono create mediante la sovrapposizione dei tre colori primari additivi: il rosso, il verde e il blu. Al contrario, le immagini per luce riflessa, come le fotografie a colori e le immagini di riviste su carta lucida, assorbono alcune lunghezze d'onda di luce e riflettono le restanti. Queste immagini sono create dalla sovrapposizione di tre colori primari sottrattivi, il ciano (completamente assorbito dal rosso), il giallo (completamente assorbito dal blu) e il magenta (completamente assorbito dal verde). In teoria ogni colore può essere riprodotto mischiando i colori ciano, giallo e magenta, ma in pratica è difficile produrre inchiostri

sufficientemente puri da ottenere un vero nero. Per questo motivo quasi tutti i sistemi di stampa a colori utilizzano quattro colori: il ciano, il giallo, il magenta e il nero, e sono chiamati stampanti CMYK. La K è a volte attribuita al nero (black), ma in realtà fa riferimento alla lastra chiave (*key plate*) alla quale le lastre dei colori sono allineate in una macchina da stampa convenzionale in quadricromia. Per rappresentare i colori i monitor usano, al contrario, luce trasmessa e il sistema RGB.

L'insieme completo di colori rappresentabili da uno schermo o da una stampante è chiamato **gamut**, cioè **gamma dei colori**. Nessun dispositivo ha una gamma di colori che corrisponde a quelli del mondo reale, dato che ogni colore assume tipicamente 256 intensità, che producono in totale solo 16.777.216 colori discreti. A causa di alcune imperfezioni dovute alla tecnologia, il numero totale di colori si riduce ulteriormente e quelli rimanenti non sono neanche distribuiti in modo uniforme nello spettro dei colori. Inoltre la percezione dei colori non dipende solo dalle leggi della fisica, ma è determinata in buona parte anche dal funzionamento dei coni e dei bastoncelli dell'occhio umano. Di conseguenza si vede che non è affatto banale convertire un'immagine a colori che appare correttamente sullo schermo in un'identica immagine stampata. I problemi principali che si incontrano in questo processo sono i seguenti:

1. i monitor a colori usano luce trasmessa, mentre le stampanti a colori usano luce riflessa;
2. i CRT generano 256 intensità per colore, mentre le stampanti usano i mezzitoni;
3. i monitor hanno un fondo nero, mentre la carta ha un colore chiaro;
4. le gamme dei colori RGB e CMYK sono diverse.

Per ottenere immagini stampate che corrispondano alla realtà (o anche alle immagini a schermo) è necessario calibrare i dispositivi, utilizzare software sofisticati per la costruzione e l'utilizzo dei profili ICC (*International Color Consortium*) e affidarsi a una considerevole esperienza da parte dell'utente.

#### **Stampanti a getto d'inchiostro**

Per le stampe casalinghe a basso costo molti preferiscono le stampanti a getto d'inchiostro (inkjet). La testina mobile di stampa, che contiene le cartucce di inchiostro, viene trascinata orizzontalmente da un nastro lungo la carta mentre l'inchiostro viene spruzzato da ugelli molto piccoli. Le goccioline di inchiostro hanno un volume di circa 1 picolitro, quindi 100 milioni di queste gocce formano una sola goccia d'acqua. Le stampanti a getto d'inchiostro sono di due tipi: piezoelettriche (utilizzate da Epson) e termiche (utilizzate da Canon, HP e Lexmark). Le prime hanno un particolare tipo di cristallo a fianco della camera contenente l'inchiostro. Quando gli viene applicata una tensione il cristallo si deforma leggermente, forzando l'uscita di una gocciolina di inchiostro dall'ugello. Il software può controllare la dimensione delle gocce regolando la tensione: maggiore è la tensione applicata, più grande è la goccia.

Le stampanti termiche (chiamate anche stampanti BubbleJet) contengono una minuscola resistenza all'interno di ciascun ugello. Quando viene applicata una tensione alla resistenza, questa si scalda molto velocemente, aumentando istantaneamente la temperatura dell'inchiostro fino al punto di ebollizione; a questo punto l'inchiostro evapora e

forma una bolla di gas. La bolla di gas occupa un volume maggiore rispetto all'inchiostro che l'ha formata e preme sull'ugello. L'unico posto da cui l'inchiostro può fuoriuscire è la parte anteriore dell'ugello, verso la carta. L'ugello viene quindi raffreddato e il conseguente vuoto aspira un'altra goccia di inchiostro dal serbatoio. La velocità di stampa è limitata dalla velocità del ciclo di ebollizione/raffreddamento. Le goccioline sono tutte delle stesse dimensioni, ma inferiori a quelle utilizzate dalle stampanti piezoelettriche.

Le stampanti a getto d'inchiostro economiche hanno in genere risoluzioni di almeno 1200 dpi (punti per pollice), quelle di fascia alta di 4800 dpi. Queste stampanti sono economiche e di buona qualità, ma sono anche lente e utilizzano cartucce d'inchiostro costose. Quando le versioni migliori delle stampanti a getto d'inchiostro vengono utilizzate per stampare una foto professionale ad alta risoluzione su una carta fotografica con speciale rivestimento, i risultati non si distinguono dalle stampe convenzionali, anche fino a dimensioni di 8 × 10 pollici.

Per ottenere risultati migliori occorre utilizzare carte e inchiostri speciali. Esistono due tipi d'inchiostro: a base di **coloranti** e a **pigmenti**. I primi consistono in coloranti diluiti in un liquido, producono colori luminosi e fluiscono facilmente, ma il loro principale svantaggio è quello di sbiadirsi quando sono esposti a radiazioni ultraviolette, come quelle della luce solare. I secondi contengono invece delle particelle solide di pigmento sospese in un liquido che evapora dalla carta depositando il colore. Questi inchiostri non si sbiadiscono con il tempo, ma non sono luminosi quanto quelli basati su coloranti e le particelle di pigmento hanno la tendenza a bloccare gli ugelli, rendendone necessaria una periodica pulizia. Per stampare fotografie è necessario usare un tipo di carta lucida o patinata appositamente progettata per trattenere le gocce d'inchiostro e impedire che si spandano eccessivamente.

#### **Stampanti speciali**

Mentre le stampanti laser e a getto d'inchiostro dominano il mercato delle stampanti domestiche e da ufficio, in altre situazioni, dove sono necessari particolari requisiti in termini di qualità del colore, costi e altro, sono utilizzati altri tipi di stampanti. Una variante di quelle a getto d'inchiostro è costituita dalle stampanti a inchiostro solido. Questo tipo di stampante utilizza quattro speciali inchiostri a cera solidificati in blocchi che vengono successivamente sciolti all'interno di serbatoi riscaldati. Per queste stampanti il tempo di attesa all'accensione può raggiungere i 10 minuti, per consentire lo scioglimento dei blocchi d'inchiostro. L'inchiostro caldo viene spruzzato sulla carta, dove si solidifica e si fonde con essa grazie all'utilizzo di due rulli rigidi. In un certo senso vengono combinate le idee di spruzzare l'inchiostro, come nelle stampanti a getto d'inchiostro, e di fondere l'inchiostro sulla carta con rulli di gomma dura, come nelle stampanti laser.

Un altro tipo di stampante a colori è la **stampante a getto di cera**. Essa ha un ampio nastro rivestito di quattro inchiostri a cera e suddiviso in settori lunghi quanto la larghezza della pagina. Quando la carta passa sotto il nastro migliaia di elementi riscaldanti sciogliono la cera, che si fonde con la carta sotto forma di pixel utilizzando il sistema CMYK. Una volta le stampanti a cera erano la principale tecnologia di stampa a colori,

ma ora stanno per essere sostituite da altri tipi i cui materiali di consumo sono più economici.

Abbiamo poi la stampante a sublimazione. Anche se il termine ha un che di freudiano, fisicamente indica il passaggio dallo stato solido a quello gassoso, senza passare da quello liquido. Un noto materiale che sublima è il ghiaccio secco (diossido di carbonio ghiacciato), un altro è lo zolfo. In una stampante a sublimazione 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. Ciascun elemento riscaldante può produrre 256 livelli di temperatura; maggiore è la temperatura, maggiore è la quantità di colorante che si deposita e quindi l'intensità del colore. Diversamente da tutte le altre stampanti a colori non è necessario ricorrere alla tecnica dei mezzitoni, dato che è possibile creare una gamma quasi continua di colori. Di solito le stampati fotografiche di piccole dimensioni usano il processo di sublimazione per creare immagini fotografiche altamente realistiche su carta speciale (e costosa).

In fine, veniamo alla stampante termica, che contiene una piccola testina di stampa su cui vi sono un certo numero di piccoli aghi. Quando una corrente elettrica passa attraverso un ago, questo si scalda molto e molto in fretta. Nel momento in cui una carta speciale termosensibile passa sotto la testina di stampa, vengono disegnati sulla carta dei punti in corrispondenza degli aghi caldi. In effetti, una stampante termica è simile a una vecchia stampante ad aghi in cui gli aghi premevano contro un nastro da macchina da scrivere per disegnare punti sulla carta dietro il nastro. Le stampanti termiche sono ampiamente utilizzate per stampare gli scontrini nei negozi, nei bancomat, nelle stazioni di servizio automatiche e così via.

#### 2.4.5 Apparecchiature per le telecomunicazioni

Al giorno d'oggi la maggior parte dei calcolatori è connessa a una rete di calcolatori, che spesso è Internet. In questo paragrafo descriveremo come funzionano le apparecchiature che permettono l'accesso a queste reti.

##### Modem

Dato che l'uso dei calcolatori si è diffuso fortemente negli ultimi anni, è sempre più comune che due computer debbano comunicare fra loro. Molti, per esempio, hanno un PC a casa e lo utilizzano per comunicare con il loro calcolatore al lavoro, con un Internet provider oppure con un sistema bancario on-line. In molti casi è la linea telefonica a permettere la comunicazione fisica tra due sistemi.

Una linea telefonica non è adatta a trasmettere i segnali di un calcolatore, che di solito rappresentano il bit 0 con 0 volt e il bit 1 con 3 o 5 volt, come mostra la Figura 2.38(a). Quando sono trasmessi su una linea telefonica progettata per la voce, i segnali a due livelli subiscono una notevole distorsione, provocando di conseguenza errori di trasmissione. Tuttavia un segnale dalla forma sinusoidale e con una frequenza che varia tra 1000 e 2000 Hz può essere trasmesso con una distorsione relativamente piccola; l'uso di questo segnale, chiamato portante, è alla base della maggior parte dei sistemi di telecomunicazione.

Dato che le pulsazioni di un'onda sinusoidale sono conosciute a priori, una sinusode perfetta non trasmette alcuna informazione. Variandone però l'ampiezza, la frequenza o la fase è possibile trasmettere una sequenza di 1 o di 0, come mostra la Figura 2.38. Questo processo è conosciuto con il termine di modulazione e i dispositivi in grado di attuarlo si chiamano modem, dalla prime lettere delle parole inglesi MOdulator DEModulator. Nella modulazione d'ampiezza [vedi Figura 2.38(b)] si utilizzano due diverse tensioni per rappresentare rispettivamente i valori 0 e 1. Se si ascoltasse la trasmissione di dati digitali a una velocità molto bassa si sentirebbe un forte rumore per i valori 1 e nessun rumore per i valori 0.

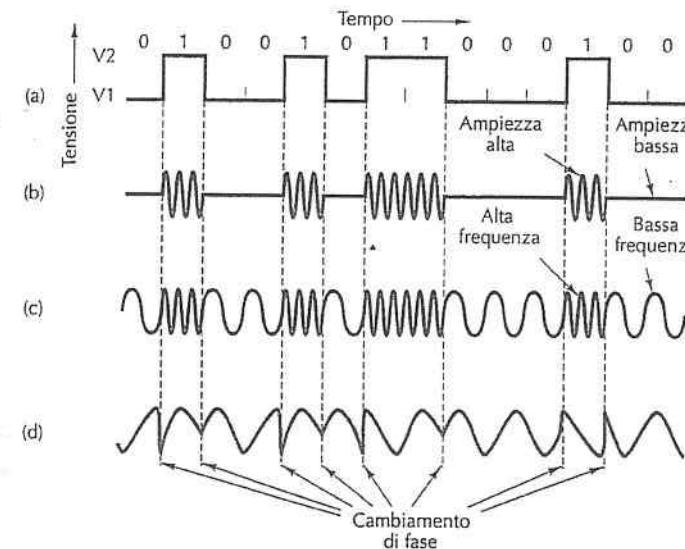


Figura 2.38 Trasmissione bit a bit della stringa 01001011000100 su una linea telefonica. (a) Segnale a due livelli. (b) Modulazione d'ampiezza. (c) Modulazione di frequenza. (d) Modulazione di fase.

Nella modulazione di frequenza [vedi Figura 2.38(c)] la tensione rimane a un livello costante, ma la frequenza della portante cambia in corrispondenza dei valori 1 e 0. Se si ascoltassero i dati digitali modulati rispetto alla frequenza si sentirebbero due toni distinti, in corrispondenza dei valori 0 e 1. La modulazione di frequenza è spesso indicata con il termine modulazione (numerica) di frequenza (*frequency shift keying*).

Nella modulazione di fase [vedi Figura 2.38(d)] l'ampiezza e la frequenza non cambiano, ma la fase della portante è invertita di 180 gradi quando i dati passano da 0 a 1 o viceversa. In sistemi a modulazione di fase più sofisticati la fase della portante cambia improvvisamente di 45, 135, 225 o 315 gradi all'inizio di ogni intervallo di tempo, in

modo da poter rappresentare 2 bit per ogni intervallo; questa codifica viene chiamata **modulazione a coppia di bit**. Uno sfasamento della fase di 45 gradi potrebbe rappresentare per esempio 00, uno di 135 gradi potrebbe rappresentare 01, e così via. Esistono anche altri schemi che permettono di trasmettere 3 o più bit per intervallo. Il numero di intervalli di tempo (cioè il numero di potenziali cambiamenti del segnale al secondo) è chiamato **baud**. Se si utilizzano 2 o più bit per intervallo, il **bit rate** (il valore della velocità in bit al secondo) sarà maggiore del **baud rate** (il valore della velocità in baud al secondo); è un errore molto comune confondere queste due unità di misura. Ripetiamo per l'ennesima volta: il baud rate esprime il numero di variazioni del segnale per secondo, il bit rate esprime il numero di bit trasmessi per secondo. Il bit rate è generalmente un multiplo del baud rate, anche se teoricamente potrebbe essere più basso.

Se i dati da trasmettere consistono in una serie di caratteri a 8 bit, potrebbe essere comodo disporre di una connessione in grado di trasmettere simultaneamente 8 bit. Per ottenere ciò servirebbero 8 coppie di fili, ma, dato che le linee telefoniche forniscono un solo canale, i bit devono essere spediti in modo seriale, uno dopo l'altro (o in gruppi di due se si usa la modulazione a coppia di bit). Con il termine **modem** si indica il dispositivo che riceve da un calcolatore i caratteri sotto forma di segnali a due livelli, un bit alla volta, e che trasmette i bit in gruppi di uno o due utilizzando una modulazione di ampiezza, di frequenza o di fase. Di solito si fa precedere ogni carattere a 8 bit da un bit d'inizio e lo si fa seguire da un bit di fine, in modo da marcarne le estremità; in questo modo ogni carattere richiede 10 bit.

Un modem in fase di trasmissione spedisce i singoli bit di un carattere a intervalli di tempo regolari. Una velocità di 9600 baud significa per esempio che il segnale cambia ogni 104 µs. Si utilizza poi un modem in ricezione per convertire una portante modulata in stringhe binarie. Dato che i bit giungono al ricevente a intervalli di tempo uniformi, il modem determina l'inizio del carattere e poi sincronizza il suo orologio per sapere quando deve campionare la linea per leggere i bit in entrata.

I modem moderni lavorano a 56 Kbps e, solitamente, a un baud rate molto più basso, utilizzando una combinazione di tecniche diverse, modulando ampiezza, frequenza e fase. Tutti questi modem sono **full-duplex**, il che significa che possono trasmettere allo stesso tempo in entrambe le direzioni (utilizzando frequenze diverse). Al contrario i modem e le linee di trasmissione che in un dato istante possono trasmettere in una sola direzione sono chiamate **half-duplex**. Le linee che possono trasmettere in una sola direzione sono invece chiamate **simplex**.

### Digital subscriber line

Quando le società telefoniche riuscirono a raggiungere la velocità di 56 Kbps, si complimentarono tra loro con grande entusiasmo. Nel frattempo però l'industria della TV via cavo offriva già velocità superiori a 10 Mbps su cavi condivisi e la TV satellitare stava raggiungendo sistemi da 50 Mbps. Dato che l'accesso a Internet aveva assunto un'importanza sempre maggiore nei loro affari, le cosiddette **telcos** (da *telephone companies*, compagnie telefoniche) cominciarono a rendersi conto che avevano bisogno di un prodotto più competitivo rispetto alle linee *dial-up*. La loro risposta fu l'offerta di un nuovo servizio di accesso digitale a Internet; i servizi che offrono una larghezza di banda

maggiori rispetto al servizio telefonico standard sono spesso chiamati a **banda larga** (*broadband*), anche se in realtà questo termine è più che altro utilizzato a fini promozionali. Da un punto di vista squisitamente tecnico **broadband** indica la presenza di più canali di trasmissione mentre **baseband** indica che esiste un solo canale. Quindi, in teoria, una rete Ethernet a 10 Gbit, più veloce di qualunque servizio "broadband" offerto dalle compagnie telefoniche, non è per niente broadband, visto che utilizza un solo canale.

All'inizio vi erano molte offerte che si sovrapponevano e che rientravano, al variare di *x*, sotto il nome generale di **xDSL** (*Digital Subscriber Line*). In seguito descriveremo il servizio **ADSL** (*Asymmetric DSL*, "DSL asimmetrica"), destinato con ogni probabilità a diventare il più diffuso. Dato che lo sviluppo della ADSL è ancora in corso e non sono stati definiti tutti gli standard, alcuni dei dettagli seguenti potrebbero cambiare in futuro; ciononostante l'immagine che ne verrà data dovrebbe rimanere sostanzialmente valida. Per maggiori informazioni riguardo la ADSL si veda (Summers, 1999 e Vetter et al., 2000).

La lentezza dei modem deriva dal fatto che i telefoni sono stati inventati per trasportare la voce umana e l'intero sistema è stato attentamente ottimizzato a questo scopo e i dati hanno sempre interpretato il ruolo dei "fratellastri". Il cavo che collega un abbonato alla compagnia telefonica, chiamato **ciclo locale**, di solito viene limitato a 3000 Hz. Questo vincolo, che limita anche la velocità di trasferimento dati, viene ottenuto mediante un filtro che si trova nell'ufficio della compagnia telefonica. La larghezza di banda reale del ciclo locale dipende dalla sua lunghezza e può raggiungere 1,1 MHz per distanze di pochi chilometri.

La Figura 2.39 mostra l'approccio più comune nell'offerta del servizio ADSL. Quello che in pratica viene fatto consiste nel suddividere il ciclo locale in 256 canali indipendenti di 4312,5 Hz ciascuno. Il canale 0 è usato per il servizio telefonico tradizionale, **POTS** (*Plain Old Telephone Service*). I canali 1-5 non sono utilizzati e servono a mantenere separate la voce e i dati in modo che non interferiscano tra loro. Dei 250 canali rimanenti, uno è usato per il controllo del traffico in uscita, un altro per il controllo del traffico in entrata, mentre gli altri sono a disposizione per la trasmissione dei dati. Una ADSL corrisponde quindi a 250 modem.

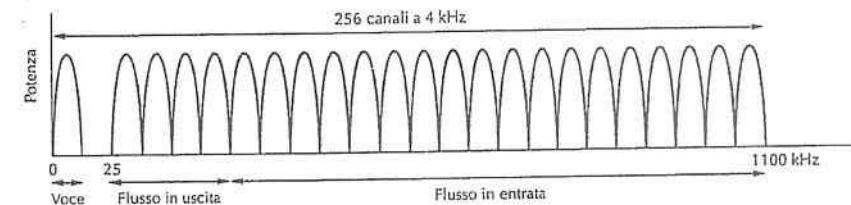


Figura 2.39 Funzionamento di una ADSL.

In teoria ciascuno dei canali rimanenti può essere utilizzato per un flusso dati full-duplex, ma in pratica le armoniche, le interferenze e altri effetti mantengono i sistemi reali molto al di sotto di questo dato teorico. Spetta al fornitore di servizi Internet determinare quanti canali sono usati in uscita e quanti in entrata. Dato che la maggior parte degli utenti scarica più dati di quanti non ne trasmetta, la maggior parte dei fornitori di servizi alloca l'80%-90% della larghezza di banda al canale in entrata (anche se una suddivisione 50-50 tra i due canali sarebbe tecnicamente possibile). A questa scelta si deve la lettera "A" nell'acronimo ADSL. Una suddivisione piuttosto comune prevede di utilizzare 32 canali in uscita e gli altri in entrata.

All'interno di ogni canale si controlla costantemente la qualità della linea e, se necessario, si modifica la velocità di trasferimento dati: per questo motivo canali differenti possono avere velocità diverse. I dati vengono spediti con una combinazione di modulazione di ampiezza e di fase, che permette di trasmettere fino a 15 bit per baud. Per esempio, con 224 canali in entrata e 15 bit/baud a 4000 baud, si ottiene una larghezza di banda in entrata di 13,44 Mbps. In pratica il rapporto tra segnale e rumore non è mai abbastanza buono per ottenere questa velocità, ma su cicli ad alta qualità e su distanze brevi è possibile raggiungere velocità di 8 Mbps.

La Figura 2.40 mostra una tipica organizzazione di una ADSL. In questo schema l'utente o un tecnico della compagnia telefonica devono installare un dispositivo d'interfaccia, NID (*Network Interface Device*) presso l'edificio del cliente. Questa piccola scatola di plastica segna la fine della proprietà della compagnia telefonica e l'inizio di quella dell'utente. Vicino al NID (o talvolta combinato con quest'ultimo) è presente un divisore (*splitter*), cioè un filtro analogico che divide la banda 0-4000 Hz (per il servizio telefonico tradizionale) dai dati. Il segnale POTS è instradato verso il telefono o il fax esistenti, mentre il segnale dei dati è instradato verso un modem ADSL. Il modem ADSL è un processore di segnale digitale impostato appositamente per funzionare come 250 modem in parallelo a diverse frequenze. Dato che la maggior parte dei modem ADSL attuali è esterna, il calcolatore deve essere connesso attraverso un collegamento ad alta velocità. Di solito ciò si ottiene inserendo una scheda Ethernet (la più diffusa tra le reti locali) nel calcolatore e creando una piccola Ethernet a due nodi che comprende soltanto il calcolatore e il modem ADSL. A volte si utilizza una porta USB al posto della Ethernet, ma in futuro saranno disponibili schede specifiche per modem ADSL.

All'altra estremità del cavo, presso la compagnia telefonica, è installato un altro divisore. Qui la parte del segnale relativa alla voce è separata tramite un filtro e spedita verso un normale commutatore per la voce. I segnali a frequenza maggiore di 26 kHz sono instradati verso un nuovo tipo di dispositivo, chiamato DSLAM (*Digital Subscriber Line Access Multiplexer*), che contiene lo stesso tipo di processore di segnale digitale all'interno del modem ADSL. Dopo che il segnale digitale è stato riconvertito in un flusso di bit, si creano i pacchetti di dati da spedire all'ISP.

#### Internet via cavo

Oggi molte società di TV via cavo offrono l'accesso a Internet attraverso i propri cavi. Dato che la tecnologia differisce significativamente da quella della ADSL, vale la pena analizzarla brevemente. In ogni grande città c'è una sede principale dell'operatore via

cavo, mentre, sparse sul territorio, ci sono un gran numero di scatole piene di componenti elettronici, chiamate stazioni di testa. Le stazioni di testa sono connesse alla sede principale da fibre ottiche o da cavi con un'alta larghezza di banda.

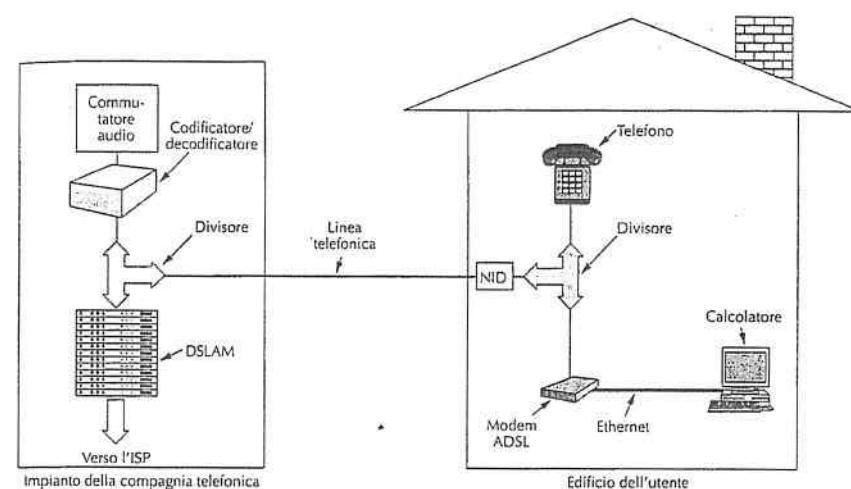


Figura 2.40 Tipica configurazione dei dispositivi per la ADSL.

Da ciascuna stazione di testa partono dei cavi che raggiungono centinaia di case e uffici e i clienti si collegano al cavo che passa vicino alla propria abitazione. Centinaia di utenti condividono quindi uno stesso cavo che parte dalla stazione di testa e la cui larghezza di banda, in genere, è di circa 750 MHz. Questo sistema è radicalmente differente dalla ADSL dove gli utenti hanno un cavo privato (cioè non condiviso) che li connette alla sede della compagnia telefonica. Tuttavia, all'atto pratico, non vi è una gran differenza tra avere il proprio canale a 1,1 MHz verso la sede della compagnia telefonica e dover condividere lo spettro di 200 MHz di un cavo connesso a una stazione di testa con altri 400 utenti, metà dei quali non lo usa nello stesso istante. Ciò significa però che un utente di Internet via cavo avrà un servizio migliore alle 4 di mattina, piuttosto che alle 4 di pomeriggio, mentre le prestazioni del servizio ADSL sono costanti per tutto il giorno. Chi voglia avere un servizio Internet via cavo dalle ottime prestazioni dovrebbe considerare l'ipotesi di trasferirsi in un quartiere residenziale (gli edifici sono lontani gli uni dagli altri e quindi ci sono pochi clienti per cavo) oppure in un quartiere povero (nessuno può permettersi un servizio Internet).

Il fatto che il cavo sia un mezzo condiviso pone il grande problema di determinare chi può spedire dati, quando e a quale frequenza. Per vedere come funziona tutto ciò, occorre prima descrivere brevemente come funziona la TV via cavo. Nel Nord America i canali televisivi via cavo occupano di solito l'intervallo tra 50 e 550 MHz (tranne per

le frequenze delle radio FM comprese tra 88 e 108 MHz). Questi canali hanno una larghezza di 6 MHz, comprese le bande di sicurezza necessarie per prevenire la sovrapposizione tra canali contigui. In Europa lo schema di allocazione è simile, anche se la banda parte da 65 MHz e ogni canale occupa tra i 6 e gli 8 MHz, per via della maggiore risoluzione richiesta dai sistemi PAL e SECAM. La parte inferiore della banda non è utilizzata per la trasmissione televisiva.

Per aggiungere il servizio di Internet, le società via cavo devono risolvere due problemi:

1. come aggiungere l'accesso a Internet senza interferire con i programmi TV;
2. come ottenere un traffico bidirezionale, dato che gli amplificatori sono per loro natura unidirezionali.

Le soluzioni adottate sono le seguenti. I cavi moderni offrono una banda minima di 550 MHz, ma arrivano spesso a 750 MHz e oltre. I canali in uscita (cioè dall'utente alla stazione di testa) sono collocati nella banda compresa tra 5 e 42 MHz (leggermente più alta in Europa), mentre il traffico in entrata (cioè dalla stazione di testa all'utente) utilizza la parte alta della banda totale (Figura 2.41).

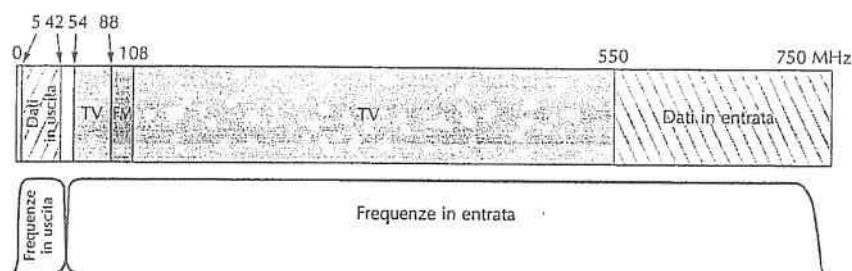


Figura 2.41 Allocazione delle frequenze in un classico sistema di TV via cavo utilizzato per l'accesso a Internet.

Occorre notare che, dato che i segnali televisivi sono tutti in entrata, è possibile utilizzare amplificatori in uscita solo nella regione tra 5 e 42 MHz e amplificatori in entrata che lavorano dai 54 MHz in su, com'è mostrato nella figura. Di conseguenza si ottiene un'asimmetria tra le bande in entrata e quelle in uscita, dato che è disponibile una regione di spettro sopra la frequenza dedicata alla televisione maggiore che non al di sotto.

D'altra parte il traffico è principalmente in entrata e quindi ciò non impensierisce né rattrista gli operatori via cavo. Inoltre, come abbiamo già visto, anche le compagnie telefoniche forniscono spesso un servizio di DSL asimmetrico, senza che ve ne sia alcuna esigenza tecnica.

L'accesso a Internet richiede un apposito modem, che abbia due interfacce: una verso il calcolatore e l'altra verso il cavo della rete. L'interfaccia tra il calcolatore e il modem via cavo è semplice e di solito si tratta di una scheda Ethernet, come nel caso della

ADSL. In futuro l'intero modem potrebbe essere una piccola scheda inserita all'interno del calcolatore, esattamente come per i vecchi modem telefonici.

L'altra interfaccia è invece più complicata. Una gran parte dello standard via cavo è legata all'ingegneria delle trasmissioni radio, un argomento che esula dallo scopo di questo libro. Il solo aspetto che vale la pena sottolineare è che i modem via cavo, come quelli ADSL, sono sempre attivi: stabiliscono una connessione non appena la macchina viene accesa e la mantengono finché non viene spenta, dato che le tariffe degli operatori via cavo non dipendono dal tempo di connessione.

Per comprendere meglio il loro funzionamento, vediamo che cosa succede quando un modem via cavo viene collegato e acceso. Il modem effettua una scansione dei canali in entrata per cercare uno speciale pacchetto spedito periodicamente dalla stazione di testa e che fornisce alcuni parametri di sistema. Non appena trova tale pacchetto il modem comunica la propria presenza utilizzando uno dei canali in uscita. A questo punto la stazione di testa risponde assegnando al modem alcuni canali in entrata e in uscita; la stazione di testa può modificare in un secondo momento tale assegnamento per bilanciare il carico di traffico quando lo reputa necessario.

Il modem determina quindi la propria distanza dalla stazione di testa spedendo uno speciale pacchetto e calcolando quanto tempo impieghi per ricevere una risposta; questo processo è chiamato *ranging*. È importante che il modem conosca questa distanza affinché possa regolare appropriatamente l'utilizzo di canali in uscita e per impostarne correttamente la temporizzazione. I canali sono divisi in intervalli temporali chiamati *minislot*; ogni pacchetto di dati in uscita deve entrare in uno o più *minislot* consecutivi. La stazione di testa comunica periodicamente l'inizio di una nuova serie di *minislot*, ma, per via dei tempi di propagazione sul cavo, questo segnale non è sentito simultaneamente da tutti i modem. Conoscendo la distanza che lo separa dalla stazione di testa ogni modem può calcolare quanto tempo prima era stato in realtà spedito il primo *minislot*. La lunghezza dei *minislot* dipende dalla rete, ma una tipica lunghezza del campo dati è di 8 byte.

Durante l'inizializzazione la stazione di testa assegna a ciascun modem un *minislot* da usare per richiedere una parte della larghezza di banda in uscita. In genere uno stesso *minislot* viene assegnato a più modem, generando problemi di contesa. Quando un calcolatore vuole spedire un pacchetto, lo trasferisce al modem, che richiede il numero di *minislot* necessario; se la stazione di testa accetta la richiesta, spedisce una conferma sul canale in entrata indicando al modem quali *minislot* sono stati riservati per il suo pacchetto. Il pacchetto viene quindi spedito a partire dal *minislot* che gli è stato allocato ed è inoltre possibile utilizzare un campo dell'intestazione per richiedere pacchetti aggiuntivi.

Se invece si verifica una contesa per la richiesta dei *minislot*, non verrà spedita alcuna conferma e il modem aspetterà un tempo casuale prima di riprovare. Dopo ciascun fallimento, il tempo casuale di attesa viene raddoppiato di modo che il carico si distribuisca meglio nel tempo quando il traffico è molto intenso.

I canali in entrata sono gestiti in maniera differente da quelli in uscita. Per prima cosa, esiste un solo mittente (la stazione di testa) e quindi non ci può essere contesa, e non c'è bisogno dei *minislot*, che in realtà non sono altro che una moltiplicazione statisti-

ca a divisione di tempo. In secondo luogo, poiché il traffico in entrata è solitamente molto più grande di quello in uscita, si utilizza una dimensione di 204 byte per tutti i pacchetti. Una parte di questi byte contengono un codice a correzione di errore Reed-Solomon e altre informazioni di servizio, lasciando solo 184 byte ai dati dell'utente. Questi valori sono stati scelti per essere compatibili con la televisione terrestre basata su MPEG-2, in modo che la formattazione dei canali della TV e di quelli del traffico in entrata sia la stessa. La Figura 2.42 mostra l'organizzazione logica delle connessioni.

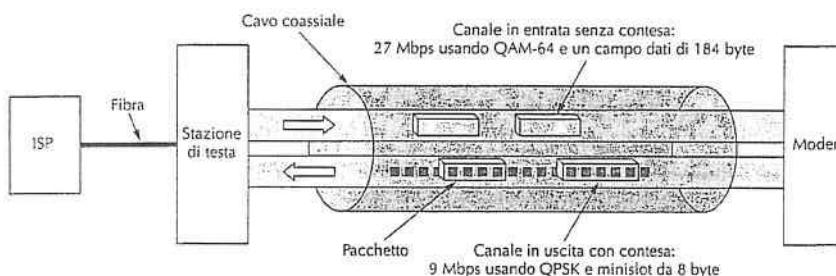


Figura 2.42 Tipici dettagli dei canali in entrata e in uscita nel Nord America.

Tornando all'inizializzazione del modem, dopo aver terminato l'operazione di ranging e ottenuto i propri canali in uscita, quelli in entrata e l'assegnamento dei minislot, il modem è libero di iniziare a spedire pacchetti. Questi giungono alla stazione di testa, che li trasmette via cavo su un canale dedicato alla sede principale della compagnia e quindi al fornitore di servizi (che potrebbe essere la compagnia stessa). Il primo pacchetto serve a richiedere all'ISP un indirizzo di rete (tecnicamente un indirizzo IP), assegnato dinamicamente. Il modem inoltre richiede, e ottiene, l'ora del giorno con grande precisione.

Il passo successivo riguarda la sicurezza. Dato che i cavi sono condivisi, chi lo volesse potrebbe leggere tutto il traffico che passa da lui. Per questo motivo il traffico viene interamente criptato in entrambe le direzioni, per evitare che qualcuno ficchi il naso (letteralmente) nei dati dei propri vicini. Una parte della procedura d'inizializzazione serve a stabilire le chiavi di cifratura. A prima vista si potrebbe pensare che è impossibile che due estranei, la stazione di testa e il modem, possano stabilire una chiave segreta alla luce del sole e davanti a migliaia di persone che li osservano. In realtà ciò è possibile, ma la tecnica utilizzata (l'algoritmo Diffie-Hellman) va oltre lo scopo di questo libro. Per un'analisi al riguardo si veda (Kaufman et al. 2002).

Infine il modem deve farsi identificare e fornire il proprio numero identificativo univoco sul canale sicuro. A questo punto l'inizializzazione è completa e l'utente può registrarsi presso l'ISP e cominciare a utilizzare il servizio.

Ci sarebbero molte altre cose da dire a proposito dei modем via cavo; alcuni significativi riferimenti bibliografici sono (Adams e Dulchinos, 2001; Donaldson e Jones, 2001 e Dutta-Roy, 2001).

#### 2.4.6 Macchine fotografiche digitali

Le macchine fotografiche digitali sono ormai diventate una periferica del calcolatore, dato che i computer vengono sempre più frequentemente usati anche per la fotografia digitale. Spieghiamo brevemente come funziona. Tutte le macchine fotografiche hanno una lente che forma nella parte posteriore, internamente all'apparecchio, un'immagine del soggetto inquadrato. In una macchina fotografica di tipo classico la parte posteriore è coperta da una pellicola su cui si forma un'immagine latente nel momento in cui viene colpita dalla luce; durante lo sviluppo delle foto si può rendere visibile questa immagine utilizzando dei composti chimici. Una macchina fotografica digitale funziona nello stesso modo tranne per il fatto che al posto della pellicola c'è una griglia rettangolare di CCD (Charge-Coupled Device, "dispositivo ad accoppiamento di carica"), sensibili alla luce (alcune macchine fotografiche digitali usano i CMOS, ma noi ci concentreremo sui più comuni CCD).

Quando un CCD è colpito dalla luce si carica elettricamente in modo proporzionale alla quantità di luce ricevuta. Questa carica può essere letta da un convertitore analogico-digitale e trasformata in un valore compreso tra 0 e 255 (per gli apparecchi economici) oppure tra 0 e 4095 (per le macchine fotografiche reflex digitali con un'unica lente). Il funzionamento è schematizzato nella Figura 2.43.

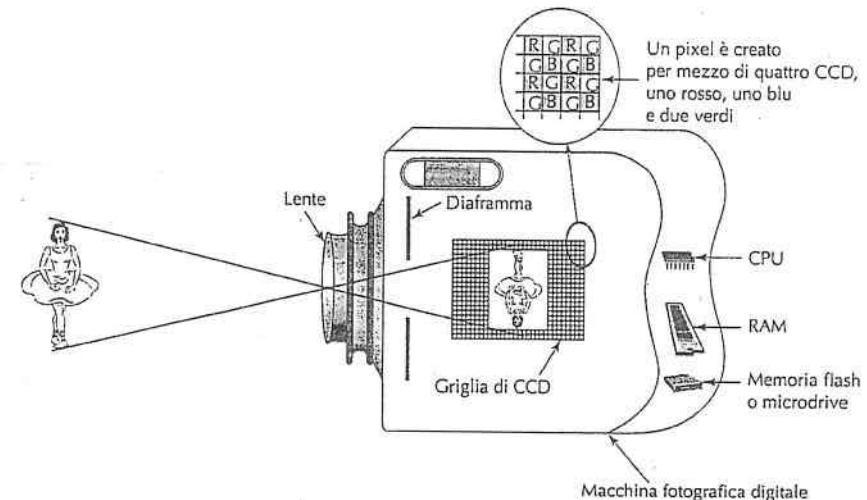


Figura 2.43 Macchina fotografica digitale.

Ciascun CCD produce un unico valore, indipendentemente dal colore della luce che lo colpisce. Per formare un'immagine a colori i CCD sono organizzati in gruppi di quattro elementi. Un filtro di Bayer è posizionato sopra il CCD per permettere che la luce rossa

colpisca solo uno dei quattro CCD in ciascun gruppo, la luce blu ne colpisca soltanto unaltro e la luce verde i due restanti. Si utilizzano due elementi per il verde, in quanto è molto più pratico utilizzare quattro CCD per ogni pixel al posto di tre e perché l'occhio umano è più sensibile alla luce verde che alle luci rosse e blu. Quando un produttore di macchine fotografiche digitali afferma che un modello ha, per esempio, 6 milioni di pixel, sta mentendo. La macchina ha 6 milioni di CCD, che corrispondono a 1,5 milioni di pixel. L'immagine sarà letta come una griglia di  $2828 \times 2121$  pixel (sulle macchine a basso costo) oppure di  $3000 \times 2000$  pixel (sulle SLR digitali), mentre i pixel aggiuntivi sono prodotti per interpolazione dal software interno alla macchina fotografica.

Quando viene premuto il pulsante per scattare una foto il software compie tre azioni: impone la messa a fuoco, determina l'esposizione e calcola il bilanciamento del bianco. La messa a fuoco automatica funziona analizzando le alte frequenze dell'immagine e muovendo la lente finché non vengono massimizzate al fine di ottenere il maggiore dettaglio possibile. L'esposizione è determinata misurando la luce che raggiunge i CCD e regolando di conseguenza il diaframma della lente e il tempo di esposizione, di modo che l'intensità luminosa corrisponda al punto medio dell'intervallo gestito dai CCD. Il bilanciamento del bianco consiste nel misurare lo spettro della luce incidente e nell'effettuare, successivamente, le necessarie correzioni cromatiche.

Successivamente l'immagine viene letta dai CCD e memorizzata come una griglia di pixel nella memoria interna della macchina digitale. Le macchine digitali SLR di livello professionale, utilizzate dai fotografi professionisti, sono in grado di scattare otto fotogrammi ad alta risoluzione al secondo per 5 secondi; esse richiedono circa 1 GB di RAM interna per salvare le immagini prima di poterle elaborare e memorizzare in modo permanente. Le macchine fotografiche meno costose hanno una quantità di memoria RAM leggermente minore.

Nella fase successiva alla cattura dell'immagine, il software della macchina effettua la correzione per il bilanciamento del bianco, in modo da compensare le componenti rossastre o bluastre della luce (dovute per esempio a un soggetto in ombra o all'uso del flash). In seguito applica un algoritmo per la riduzione del rumore e un altro per compensare i CCD difettosi. Tenta poi di rendere l'immagine più definita e meno sfocata (a meno che questa funzione non sia stata disabilitata) cercando i contorni e aumentando l'intensità del gradiente intorno a loro.

Infine l'immagine può essere compressa per ridurre la quantità di spazio richiesta. Un formato comunemente utilizzato è JPEG (*Joint Photographic Experts Group*), che prevede l'applicazione della trasformata di Fourier bidimensionale e il taglio di alcune componenti ad alta frequenza. Questa trasformazione riduce il numero di bit necessari a rappresentare l'immagine, sacrificando però i dettagli più fini.

Quando sono terminate tutte le elaborazioni interne alla macchina fotografica, l'immagine viene memorizzata, generalmente su una memoria flash o un hard disk rimovibile di piccole dimensioni chiamato *microdrive*. La fase di post-elaborazione e la scrittura possono richiedere alcuni secondi per ciascuna immagine.

Quando l'utente torna a casa può collegare la macchina fotografica digitale al calcolatore utilizzando solitamente un cavo USB. Le immagini vengono così trasferite dalla macchina fotografica al disco del calcolatore. Utilizzando programmi appositi, come

Adobe Photoshop, l'utente può ritagliare l'immagine, regolarne luminosità, contrasto, bilanciamento cromatico, nitidezza e fuoco, rimuovere parti dell'immagine e applicare numerosi filtri. Quando è soddisfatto del risultato l'utente può stampare i file delle immagini con una stampante a colori, spedirle via Internet a un sito di condivisione di immagini o a un negozio di stampa oppure può riversarle su un CD-ROM o un DVD per archiviarle ed eventualmente stamparle in un secondo momento. In una macchina fotografica SLR la quantità di potenza computazionale, di RAM, di spazio su hard disk e di software necessario è notevole. Il calcolatore non solo deve compiere tutte le azioni descritte, ma anche comunicare in tempo reale con la CPU della lente e con la CPU del flash, ridisegnare l'immagine sullo schermo LCD e gestire tutti i pulsanti, rotelle, luci, display e tasti della macchina. Si tratta di un sistema integrato estremamente potente, che spesso, in termini di prestazioni, è paragonabile ai calcolatori desktop di qualche anno fa.

#### 2.4.7 Codifica dei 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 fra loro utilizzino lo stesso codice, altrimenti non saranno in grado di capirsi. Per questa ragione sono stati definiti alcuni standard, di cui esamineremo i due più importanti.

#### ASCII

Un codice ampiamente utilizzato si chiama ASCII (*American Standard Code for Information Interchange*, "standard americano per lo scambio d'informazioni"). I caratteri ASCII sono definiti da 7 bit, permettendo così un totale di 128 caratteri distinti. Ciononostante, visto che i computer sono orientati ai byte, ogni carattere ASCII viene normalmente memorizzato in un byte distinto. La Figura 2.44 mostra il codice ASCII. I codici compresi tra 0 e 1F (in esadecimale) sono caratteri di controllo e non vengono stampati. I codici da 128 a 255 non fanno parte della codifica ASCII, ma i PC IBM li hanno utilizzati per caratteri speciali come le *emoticon* e la maggior parte dei computer attuali li utilizza ancora.

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"). In realtà i messaggi spediti sulle linee telefoniche e sulle reti sono formattati in maniera leggermente differente e i caratteri di controllo ASCII pensati per la trasmissione non sono praticamente più utilizzati.

I caratteri ASCII stampabili comprendono lettere maiuscole e minuscole, cifre, simboli di punteggiatura e alcuni simboli matematici.

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

Esa	Car	Esa	Car	Esa	Car	Esa	Car	Esa	Car	Esa	Car
20	Spazio	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	¤	71	q
22	”	32	2	42	B	52	R	62	¤	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	\$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	‘	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	:	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	-	3E	>	4E	N	5E	-	6E	n	7E	-
2F	/	3F	?	4F	O	5F	-	6F	o	7F	DEL

Figura 2.44 Caratteri ASCII.

## UNICODE

Il mercato dei calcolatori è nato e cresciuto principalmente negli Stati Uniti, il che ha portato a un insieme di caratteri, quello ASCII, adatto alla lingua inglese, ma meno per altre lingue. I francesi hanno bisogno degli accenti (per esempio, *système*), i tedeschi dei segni diacritici (per esempio, *für*), e così via. Alcune lingue europee hanno delle lettere che non sono comprese nel codice ASCII, come la tedesca ß e la danese ø. Altre hanno alfabeti completamente differenti (per esempio, il russo o l'arabo) e alcune lingue addirittura non hanno alfabeto (per esempio, il cinese). Dato che i computer si sono diffusi nei quattro angoli del mondo e i rivenditori di software desiderano vendere i propri prodotti anche nei paesi in cui la maggior parte della gente non parla inglese, è sorta la necessità di un diverso insieme di caratteri.

Il primo tentativo di espandere il codice ASCII fu il codice IS 646, che aggiungeva ulteriori 128 caratteri, trasformandolo così in un codice a 8 bit chiamato Latin-1. I caratteri aggiuntivi erano per lo più lettere latine accentate o con segni diacritici. Il tentativo successivo fu il codice IS 8859 che introduceva il concetto di *code page*, un insieme di 256 caratteri specifico di una particolare lingua o di un gruppo di lingue. Il codice IS 8859-1 corrisponde a Latin-1, il codice IS 8859-2 gestisce alcune lingue slave (per esempio, il ceco e il polacco), mentre il codice IS 8859-3 contiene i caratteri richiesti dal turco, dal maltese, dall'esperanto, dal galiziano e da altre lingue. Il problema principale del *code page* è che il software deve tener traccia di quale pagina è attiva, rendendo tra l'altro impossibile mischiare lingue diverse in uno stesso documento; inoltre questo schema non copre né il giapponese né il cinese.

Un gruppo di produttori di calcolatori ha deciso di risolvere il problema formando un consorzio per creare un nuovo sistema, chiamato UNICODE, che successivamente è diventato uno Standard Internazionale (IS 10646).

Oggi UNICODE è supportato da alcuni linguaggi di programmazione (per esempio, Java), da alcuni sistemi operativi (per esempio, Windows XP) e da molte applicazioni. Dato che il mercato dei calcolatori tende a diventare sempre più globale è molto probabile che questo standard venga accettato sempre di più.

L'idea alla base di UNICODE consiste nell'assegnare a ogni carattere un valore a 16 bit, chiamato *code point*; non esistono quindi caratteri o sequenze speciali composte da più byte. Il fatto che ogni simbolo sia composto da 16 bit rende più semplice la scrittura di programmi.

Utilizzando simboli a 16 bit, UNICODE ha 65.536 *code point*. Dato che le lingue di tutto il mondo usano complessivamente circa 200.000 simboli, i *code point* sono una risorsa scarsa che deve essere allocata con molta attenzione. Circa metà dei *code point* sono già stati assegnati e il consorzio UNICODE valuta in continuazione nuove proposte per i rimanenti. Per far sì che il codice UNICODE fosse accettato più velocemente, il consorzio ha deciso intelligentemente di utilizzare Latin-1 per i *code point* compresi tra 0 e 255, rendendo più facile la conversione tra ASCII e UNICODE. Inoltre, per evitare uno spreco eccessivo di *code point*, ogni segno diacritico ha il suo proprio *code point* e spetta al software combinarlo con la lettera vicina per formare un nuovo carattere. Anche se ciò dà più lavoro al software si risparmiano preziosi *code point*.

Lo spazio dei *code point* è diviso in blocchi, multipli di 16. All'interno di UNICODE a ciascun alfabeto principale è assegnata una sequenza di zone consecutive; alcuni

esempi (e il numero di *code point* allocati) sono il latino (336), il greco (144), il cirillico (256), l'armeno (96), l'ebraico (112), il devanagari (128), il gurmukhi (128), l'oriya (128), il telugu (128) e il kannada (128). Si noti che a queste lingue sono stati assegnati più *code point* delle loro lettere. Questa scelta deriva in parte dal fatto che molte lingue hanno più forme per ciascuna lettera; ciascuna lettera inglese ha, per esempio, due forme: quella minuscola e quella MAIUSCOLA. Alcune lingue hanno tre o quattro forme, talvolta dipendenti dalla posizione della lettera all'interno di una parola, se si trova cioè all'inizio, nel mezzo o alla fine.

I *code point*, oltre alle lettere di questi alfabeti, sono stati assegnati anche ai segni diacritici (112), ai simboli di punteggiatura (112), ai caratteri soprascritti e sottoscritti (48), ai simboli matematici (256), alle forme geometriche (96) e ai simboli ornamentali (192).

Dopo questi *code point* ci sono i simboli richiesti dal cinese, dal giapponese e dal coreano. Prima ci sono 1024 simboli fonetici (cioè il katakana e il bopomofo) e poi gli ideogrammi Han (20.992), usati sia dal cinese sia dal giapponese, e infine le sillabe coreane Hangul (11.156).

6400 *code point* sono inoltre stati allocati per uso locale, di modo che gli utenti possono definire caratteri speciali per usi particolari.

Anche se UNICODE risolve molti problemi legati all'internazionalizzazione, esso non risolve (e non cerca di risolvere) tutti i problemi globali. Per esempio, mentre le lettere dell'alfabeto latino sono nel loro ordine corretto, gli ideogrammi Han non sono ordinati come nel dizionario. Da ciò ne consegue che un programma inglese può ordinare alfabeticamente le parole *cat* e *dog* controllando i valori UNICODE delle loro iniziali, mentre lo stesso non può però essere fatto con un programma giapponese, che necessita di tabelle esterne per capire quale simbolo preceda l'altro.

Un altro problema è che con il tempo continuano ad apparire nuove parole. Cinquant'anni fa nessuno parlava di *app*, *chatroom*, *cyberspazio*, *emoticon*, *gigabyte*, *laser*, *modem*, *smiley* o *videocassette*. Se in inglese l'aggiunta di nuove parole non richiede nuovi *code point*, in giapponese sì. Oltre a nuove parole tecniche, c'è la richiesta di aggiungere almeno 20.000 nomi personali (soprattutto cinesi) e di luoghi. I non vedenti ritengono che il linguaggio Braille dovrebbe far parte di UNICODE, così come altri gruppi con particolari interessi (di tutti i tipi) vorrebbero che ciò che percepiscono fosse codificato, di loro diritto, in *code point*. Il consorzio UNICODE analizza e valuta tutte le nuove proposte.

UNICODE utilizza lo stesso *code point* per i caratteri che, in giapponese e cinese, hanno un aspetto simile, ma un significato differente o una scrittura leggermente diversa (come se gli elaboratori di testo inglesi scrivessero “blue” come “blew” solo perché suonano allo stesso modo). Alcune persone interpretano questo fatto come un'ottimizzazione dovuta alla scarsità di *code point*; altri come un esempio di imperialismo culturale anglosassone (assegnare ai caratteri valori a 16 bit non era forse una scelta altamente politica?). Per rendere la cosa ancora peggiore si pensi che un dizionario giapponese completo contiene 50.000 kanji (nomi esclusi); avendo a disposizione soltanto 20.992 *code point* per gli ideogrammi Han, è necessario compiere delle scelte. Non tutti i giapponesi ritengono che un consorzio di produttori di calcolatori, anche se alcuni sono giapponesi, sia il foro ideale per prendere queste decisioni.

Immaginate che neanche 65.536 *code point* sono bastati per soddisfare tutti! E così nel 1996 furono aggiunti 16 plane da 16 bit, portando il numero totale dei caratteri a 1.114.112.

### UTF-8

Anche se migliore della codifica ASCII, Unicode alla fine ha esaurito i *code point*. Inoltre Unicode usa 16 bit per carattere per rappresentare testo ASCII puro, il che è uno spreco. Come conseguenza, per rispondere a questi problemi è stato sviluppato un altro schema di codifica: il sistema chiamato **UTF-8 UCS Transformation Format**, dove UCS è l'acronimo di *Universal Character Set* (insieme universale di caratteri), in sostanza Unicode. I codici UTF-8 sono di lunghezza variabile, da 1 a 4 byte, e possono codificare circa due miliardi di caratteri. UTF-8 è l'insieme di caratteri dominante nel World Wide Web.

Una delle vantaggiose proprietà di UTF-8 è che codici da 0 a 127 corrispondono ai caratteri ASCII, che possono così essere espressi in 1 byte (rispetto ai 2 di Unicode). Per gli altri caratteri, il bit più significativo del primo byte è impostato a 1, a indicare che seguono uno o più byte aggiuntivi. Sono utilizzati in tutto sei formati differenti, come illustrato nella Figura 2.45. I bit contrassegnati con “d” sono bit di dati.

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

Figura 2.45 Lo schema di codifica UTF-8.

UTF-8 ha diversi vantaggi rispetto a Unicode e ad altri sistemi di codifica. Primo, se un programma o un documento utilizza solo i caratteri che fanno parte della codifica ASCII, ogni carattere può essere rappresentato con 8 bit. In secondo luogo, il primo byte di ogni carattere UTF-8 determina univocamente il numero di byte nel carattere. Terzo, 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. In particolare, in caso di errore di comunicazione o di memoria, è sempre possibile andare avanti e trovare l'inizio del carattere successivo (ammesso che non sia stato danneggiato).

Normalmente UTF-8 è usato per codificare soltanto i 17 plane di Unicode, anche se lo schema ammette molto di più dei 1.114.112 *code point*. Tuttavia, se gli antropologi scoprissero nuove tribù in Nuova Guinea o altrove le cui lingue non sono attualmente note (o se in futuro entreremo in contatto con gli extraterrestri), UTF-8 sarebbe pronto per l'inserimento dei loro alfabeti o ideogrammi.

## 2.5 Riepilogo

I sistemi di calcolo sono costituiti da tre tipi di componenti: processori, memorie e dispositivi di I/O. Il compito di un processore è quello di prelevare istruzioni, decodificarle ed eseguirle. Il ciclo preleva-decodifica-esegui può sempre essere descritto come un algoritmo e, difatti, è spesso eseguito da un interprete software che funziona a un livello inferiore. Per guadagnare velocità molti calcolatori hanno una o più pipeline oppure un'architettura superscalare con più unità funzionali che lavorano in parallelo. La pipeline permette di spezzare le istruzioni in più parti e di eseguire parti di istruzioni diverse in contemporanea. L'uso di unità funzionali multiple è un altro modo per ottenere un parallelismo senza influenzare l'insieme delle istruzioni o l'architettura visibili dal programmatore o dal compilatore.

I sistemi con più processori sono sempre più diffusi. I calcolatori paralleli comprendono gli array di processori, nei quali la stessa operazione è calcolata su più insiemi di dati allo stesso tempo, i multiprocessori, nei quali più CPU condividono una stessa memoria, e i multicompiler, nei quali più calcolatori, ciascuno dotato di una propria memoria, comunicano scambiandosi messaggi.

Le memorie possono essere classificate come primarie o secondarie. La memoria primaria è utilizzata per contenere il programma che viene eseguito in un dato momento. Il suo tempo di accesso è breve, di poche decine di nanosecondi al massimo, e indipendente dall'indirizzo al quale si vuole accedere. Le cache riducono ulteriormente questo tempo di accesso e sono utili perché essendo la velocità del processore maggiore di quella della memoria, l'attesa per l'accesso alla memoria rallenta notevolmente l'esecuzione dei processi. Alcune memorie sono inoltre equipaggiate con codici a correzione d'errore per migliorarne l'affidabilità.

Le memorie secondarie, al contrario, hanno tempi di accesso molto più lunghi (miliisecondi e più) e dipendenti dalla posizione del dato che deve essere letto o scritto. Le più comuni memorie secondarie sono i nastri, le memorie flash, i dischi magnetici e i dischi ottici. Esistono diversi tipi di dischi magnetici, tra i quali gli IDE, gli SCSI e i RAID. I dischi ottici comprendono i CD-ROM, i CD-R, i DVD e i Blu-Ray.

I dispositivi di I/O sono utilizzati per trasferire informazioni dal e nel calcolatore, e sono collegati al processore e alla memoria mediante uno o più bus. Alcuni esempi comprendono i terminali, i mouse, le stampanti e i modem. La maggior parte dei dispositivi di I/O utilizza il codice di caratteri ASCII, ma viene utilizzato anche UNICODE e UTF-8 sta rapidamente guadagnando sempre più consenso man mano che il mondo dei computer diventa sempre più web-centrico.

### PROBLEMI

- Consideriamo il funzionamento di una macchina con il percorso dati mostrato nella Figura 2.2. Supponiamo che il caricamento dei registri di input richieda 5 ns, l'esecuzione della ALU 10 ns e la memorizzazione del risultato nel registro 5 ns. Qual è il numero massimo di MIPS che può raggiungere questa macchina senza far ricorso alla pipeline?
- Qual è lo scopo del passo 2 nella lista del Paragrafo 2.1.2? Che cosa succederebbe se questo passo fosse omesso?

- Sul calcolatore 1 tutte le istruzioni richiedono 10 ns per essere eseguite. Sul calcolatore 2 esse richiedono invece 5 ns. Si può dire con certezza che il calcolatore 2 è più veloce? Si argomenti la risposta.
- Immaginiamo di progettare un calcolatore a singolo chip per un sistema integrato. Il chip conterrà anche tutta la memoria e funzionerà alla stessa velocità della CPU senza alcuna penalizzazione per l'accesso ai dati. Si esaminino i principi illustrati nel Paragrafo 2.1.4 e si dica se essi sono così importanti (ipotizzando che si desideri ottenere alte prestazioni).
- Un certo calcolo è altamente sequenziale, in quanto ogni passo dipende dai precedenti. Per questa elaborazione sarebbe più appropriato un array di processori o un processore a pipeline? Si spieghi la scelta.
- Per competere con le macchine tipografiche appena inventate, un monastero medioevale decise di riprodurre libri scritti a mano in un gran numero di copie facendo lavorare insieme, in un'enorme stanza, numerosi amanuensi. L'abate pronunciava ad alta voce la prima parola del libro da riprodurre e tutti gli amanuensi la trascrivavano. L'abate pronunciava quindi la seconda parola e gli amanuensi la trascrivavano. Questo processo veniva ripetuto finché l'intero libro non era letto ad alta voce e copiato. Quale fra i sistemi con parallelismo a livello di processore descritti nel Paragrafo 2.1.6 assomiglia più da vicino a questo sistema?
- Scendendo nei cinque livelli della gerarchia di memorie trattati nel testo, i tempi di accesso aumentano. Si faccia una ragionevole stima del rapporto del tempo di accesso dei dischi ottici rispetto a quello dei registri della memoria. Si assuma che i dischi siano sempre caricati nel lettore.
- Alla tipica domanda: "Credi al topolino dei dentini caduti?" i sociologi possono fornire tre risposte, sì, no e non so. In base a ciò la Società di Calcolatori Sociomagnetici ha deciso di costruire un calcolatore per elaborare i dati dei sondaggi. Questo computer ha una memoria trinaria, cioè ciascun byte (tryte?) consiste di 8 trit, ognuno dei quali può contenere i valori 0, 1 oppure 2. Quanti trit sono necessari per memorizzare un numero a 6 bit? Si dia un'espressione per il numero di trit necessari per memorizzare  $n$  bit.
- Si calcoli la velocità di trasferimento dati dell'occhio umano utilizzando le seguenti informazioni. Il campo visivo è composto da circa 106 elementi (pixel). Ciascun pixel può essere ridotto alla sovrapposizione di tre colori primari, ciascuno dei quali ha 64 intensità. La risoluzione temporale è di 100 ms.
- Si calcoli la velocità di trasferimento dati dell'orecchio umano usando le seguenti informazioni. Le persone possono sentire frequenze fino a 22 KHz. Per catturare tutta l'informazione contenuta in un segnale sonoro a 22 KHz è necessario campionare il suono al doppio della frequenza, cioè a 44 KHz. Con ogni probabilità un campione a 16 bit è sufficiente per catturare la maggior parte delle informazioni uditive (cioè l'orecchio non può distinguere più di 65.536 livelli d'intensità).
- In tutti gli esseri viventi l'informazione genetica è codificata nelle molecole di DNA. Una molecola di DNA è una sequenza lineare dei quattro nucleotidi base: A, C, G e T. Il genoma umano contiene approssimativamente  $3 \times 10^9$  nucleotidi sotto forma di circa 30.000 geni. Qual è la capacità totale (in bit) dell'informazione del genoma umano? Qual è la massima capacità d'informazione (in bit) del gene medio?
- Un certo calcolatore può essere equipaggiato con 1.073.741.824 di byte di memoria. Perché un produttore dovrebbe scegliere un numero così particolare, invece di un numero più facile da ricordare come 1.000.000.000?
- Si definisca un codice di Hamming a 7 bit con parità pari per le cifre comprese tra 0 e 9.
- Si definisca un codice per le cifre comprese tra 0 e 9 la cui distanza di Hamming sia 2.
- In un codice di Hamming alcuni bit vengono "sprecati", nel senso che sono utilizzati per effettuare il controllo e non per memorizzare l'informazione. Qual è il percentuale di bit sprecati per messaggi la cui lunghezza totale (dati + bit di controllo) è  $2^n - 1$ ? Si valuti numericamente questa espressione per valori di  $n$  compresi tra 3 e 10.
- Un carattere ASCII esteso è rappresentato su 8 bit. La codifica di Hamming associata a ogni carattere può dunque essere rappresentata da una stringa di 3 cifre esadecimali. Si codifichi il seguente testo composto da 5 caratteri ASCII estesi con l'utilizzo di un codice di Hamming a parità pari: Earth. Si rappresenti la risposta come stringa di cifre esadecimali.

17. La seguente stringa di cifre esadecimale codifica alcuni caratteri ASCII estesi in un codice di Hamming a parità pari: 0D3 DD3 0F2 5C1 1C5 CE3. Si decodifichino queste stringhe e si scrivano i caratteri che erano codificati.
18. Il disco mostrato nella Figura 2.19 ha 1024 settori per traccia e una velocità di rotazione di 7200 RPM. Qual è la velocità di trasferimento dati del disco per una singola traccia?
19. Un calcolatore ha un bus con un tempo di ciclo di 5 ns, durante il quale può leggere o scrivere dalla memoria una parola a 32 bit. Il calcolatore ha un disco Ultra4-SCSI che utilizza un bus e funziona a 160 MB/s. Solitamente la CPU preleva ed esegue un'istruzione a 32 bit a ogni intervallo di 1 ns. In che misura il disco rallenta la CPU?
20. Si immagini di dover scrivere la parte di un sistema operativo dedicata alla gestione del disco. Dal il punto di vista logico il disco sarà rappresentato come una sequenza di blocchi, a partire da 0 nella parte interna fino a un certo valore massimo nella parte esterna. Quando i file vengono creati occorre allocare settori liberi, cosa che si può fare a partire dall'interno oppure dall'esterno. Ha importanza la scelta della strategia? Si motivi la risposta.
21. Quanto tempo è necessario per leggere un disco con 10.000 cilindri, ciascuno dei quali contiene quattro tracce di 2048 settori? Prima devono essere letti tutti i settori della traccia 0 a partire dal settore 0, poi tutti i settori della traccia 1 a partire dal settore 0, e così via. Il tempo di rotazione è di 10 ms e una ricerca richiede 1 ms tra cilindri adiacenti e 20 ms nel caso peggiore. Lo spostamento da una traccia all'altra dello stesso cilindro può essere fatto istantaneamente.
22. Il RAID livello 3 è in grado di correggere errori singoli utilizzando solo un disco di parità. Qual è la particolarità del RAID livello 2? Dopo tutto anch'esso è in grado di correggere solo gli errori singoli, ma necessita di un numero maggiore di dischi.
23. Qual è l'esatta capacità (in byte) di un CD-ROM Modo 2 per il supporto, ormai standardizzato, da 80 minuti? Qual è la capacità disponibile per i dati degli utenti utilizzando il Modo 1?
24. Per masterizzare un CD-R il laser deve pulsare ad alta frequenza. Qual è la lunghezza di un impulso, in nanosecondi, quando la frequenza è 10x e si utilizza il Modo 1?
25. Per poter inserire 133 minuti di video su un DVD a singolo lato e singolo strato è necessario utilizzare un'elevata compressione. Si calcoli il fattore di compressione richiesto, assumendo che sia disponibile uno spazio di 3,5 GB per la traccia video, che la risoluzione dell'immagine sia di  $720 \times 480$  pixel con colori a 24 bit (RGB, 8 bit per colore) e che le immagini siano visualizzate a 30 frame/s.
26. I Blu-Ray hanno una velocità di 4,5 MB/s e una capacità di 25 GB. Quanto tempo è necessario per leggere l'intero disco?
27. La velocità di trasferimento dati tra la CPU e la sua memoria associata è di vari ordini di grandezza maggiore rispetto alla velocità delle componenti meccaniche di I/O. In che modo questo sbilanciamento può causare inefficienze? Com'è possibile alleviarle?
28. Un produttore afferma che il proprio terminale può visualizzare una bitmap con  $2^{24}$  colori diversi. Se l'hardware ha 1 solo byte per ciascun pixel, com'è possibile ottenere questo valore?
29. Siete parte di un team internazionale segreto di scienziati al quale è stato appena affidato il compito di studiare un essere chiamato Herb, un extra-terrestre proveniente dal pianeta 10 recentemente arrivato sulla terra. Herb vi ha fornito le seguenti informazioni sul funzionamento dei suoi occhi. Il suo campo visivo consiste in  $10^8$  pixel. Ogni pixel è essenzialmente una sovrapposizione di 5 "colori" (per esempio, infrarosso, rosso, verde, blu e ultravioletto) ognuno dei quali ha 32 intensità. Il tempo di risoluzione del campo visivo di Herb è di 10 msec. Calcolare la velocità, espressa in GB/s, degli occhi di Herb.
30. Un terminale ha uno schermo con una risoluzione di  $1920 \times 1080$ , che viene ridisegnato 75 volte al secondo. Quanto è lungo l'impulso che corrisponde a un pixel?

31. Una stampante monocromatica può stampare in ogni pagina 50 linee da 80 caratteri. Un carattere occupa in media un riquadro di  $4 \text{ mm}^2$ , di cui circa il 25% è ricoperto da toner, mentre il resto ne è privo. Uno strato di toner ha uno spessore di  $25 \mu$  e una cartuccia di toner per la stampante misura  $25 \times 8 \times 2$  cm. Per quante pagine può essere utilizzata una cartuccia?
32. Quando un testo ASCII con parità dispari viene trasmesso in modo asincrono alla velocità di 5600 caratteri/s su un modem da 56.000 bps, qual è la percentuale dei bit ricevuti che contiene effettivamente dati (e non è overhead)?
33. La società di Modem Hi-Fi ha appena progettato un nuovo modem a modulazione di frequenza che usa 64 frequenze invece di solo 2. Ciascun secondo è diviso in  $n$  intervalli temporali uguali, ognuno dei quali contiene uno dei 64 possibili toni. Quanti bit per secondo può trasmettere questo modem, utilizzando la trasmissione asincrona?
34. Un utente Internet si è abbonato a un servizio ADSL a 2 Mbps. La sua vicina si è abbonata a un servizio di Internet via cavo che utilizza una banda condivisa di 12 MHz. Lo schema di modulazione utilizzato è QAM-64 e ci sono  $n$  case per cavo, ciascuna con un calcolatore. Solo una frazione  $f$  di questi calcolatori è in linea nello stesso momento. In quali condizioni l'utente via cavo ottiene un servizio migliore dell'utente ADSL?
35. Una macchina fotografica digitale ha una risoluzione di  $3000 \times 2000$  pixel e usa 3 byte/pixel per rappresentare i colori RGB. Il produttore della macchina fotografica vorrebbe poter scrivere su una memoria flash, in soli 2 s, un'immagine JPEG con un fattore di compressione 5x. Quale velocità di trasferimento dati è richiesta?
36. Una macchina fotografica digitale professionale ha un sensore con 24 milioni di pixel, ciascuno con 6 byte/pixel. Quante immagini possono essere memorizzate su una memoria flash da 8 GB se il fattore di compressione è 5x? Si assuma che 1 GB valga  $2^{30}$  byte.
37. Si stimi quanti caratteri, spazi compresi, contiene in genere un libro di informatica. Quantii bit sono necessari per codificare un libro usando il codice ASCII con parità? Quantii CD-ROM servono per immagazzinare una libreria di 10.000 libri? Quantii DVD a doppio lato e doppio strato sono necessari per memorizzare la stessa libreria?
38. Si scriva una procedura *hamming* (*ascii*, *codificata*) che converta i 7 bit meno significativi di *ascii* in una parola di codice che sia un intero a 11 bit e che venga memorizzato in *codificata*.
39. Si scriva una funzione *distanza* (*codice*, *n*, *k*) che prenda come input un array *codice* di *n* caratteri di *k* bit ciascuno e restituisca come output la distanza dell'insieme di caratteri.