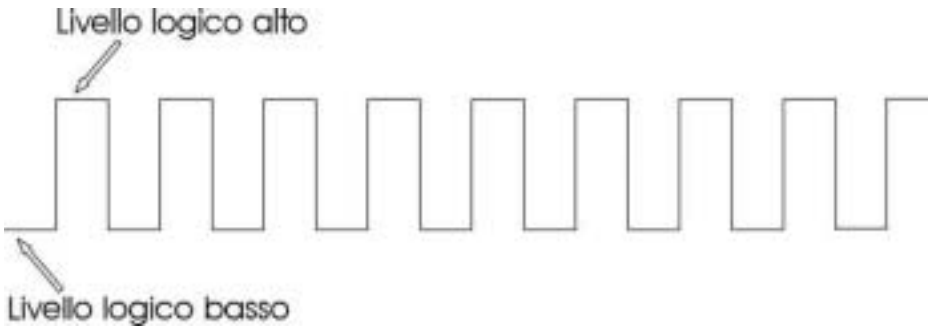
RAPPRESENTAZIONE ED ELABORAZIONE DEI DATI

Indice

Introduzione

La rappresentazione dei dati può avvenire sostanzialmente in due modi:
Analogico: modalità basata sull'impiego di dispositivi che realizzano una grandezza fisica che può variare in modo continuo (per esempio una tensione elettrica); esiste un'*analogia* diretta tra i valori della grandezza adottata e i dati da rappresentare. Tuttavia il rumore impedisce una precisione infinita nella valutazione di una variabile fisica. In figura è mostrato un esempio di segnale analogico.

Digitale: modalità con la quale ogni dato viene codificato impiegando entità distinte individualmente, discrete e organizzate in modo opportuno (per es. cifre numeriche); *digit* = cifra.
Un segnale digitale può assumere due stati logici: *alto* o *basso*. Di conseguenza l'immunità al rumore é più alta perché l'informazione è legata al livello alto o basso di un segnale e non alla sua forma esatta. In definitiva, nel caso di rappresentazione dei dati in maniera digitale, occorre distinguere semplicemente se il segnale è al livello logico alto o basso e non il suo valore preciso.



Nel prossimo paragrafo accenneremo alle tecniche utilizzate per rappresentare digitalmente i dati. In questo contesto vedremo in che modo le leggi fisiche fondamentali e le potenti tecnologie consentono la rappresentazione e l'elaborazione dei dati in modo digitale binario.

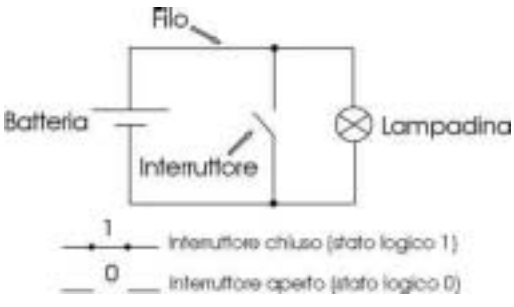
Dispositivi elettronici: transistor e gate booleani

Alla base dell'elettronica vi sono le proprietà dell'elettrone. Un elettrone é una particella elementare, cioè non ulteriormente scomponibile, dotato di carica elettrica (negativa).
Nei metalli gli elettroni sono cariche negative libere di muoversi in ogni punto del metallo stesso. Per creare una *corrente elettrica* occorre fornire, in maniera continua, un'energia capace di ordinare gli elettroni all'interno del metallo. Tale energia può essere fornita ad esempio da una batteria (caratterizzata da un polo positivo e da uno negativo) che permette di creare un flusso ordinato di elettroni. Tale flusso sta alla base del funzionamento dei sistemi elettronici binari.
Un *sistema elettronico binario* è caratterizzato dalla presenza di due stati (alto o basso), concettualmente dalla corrente che ***passa*** o che ***non passa***.

Ogni componente hardware di un computer è costituito essenzialmente da circuiti elettronici digitali, denominati **Chip**, realizzati mediante differenti tecnologie su materiale semiconduttore.
Queste tecnologie dipendono dal tipo di componenti elettronici utilizzati per la realizzazione dei chip; tali componenti sono denominati **Transistor**. A seconda del processo di costruzione, i transistor si possono classificare in: **transistor a giunzione** o **BJT** e **transistor ad effetto di campo** o **MOS (M**etallo **O**ssido **S**emiconduttore).
La tecnologia che sfrutta i transistor BJT è denominata **logica TTL** (Transistor Transistor Logic). I circuiti digitali realizzati mediante questo tipo di logica hanno la caratteristica di essere molto veloci, ma di per contro hanno un consumo di potenza elevato. Inoltre la logica associata ad essi è molto complicata, per cui si tratta di circuiti complessi che occupano molto spazio.
La tecnologia che utilizza i transistor MOS viene chiamata **logica CMOS**. Le peculiarità dei chip costruiti con tale logica sono il basso consumo e il minor spazio occupato rispetto ai circuiti TTL. Lo svantaggio di tali circuiti è che sono più lenti rispetto ai precedenti.
In base allo scopo per il quale si progetta un calcolatore si decide quale delle due logiche utilizzare. Per esempio, se si hanno problemi di spazio e non è richiesta una velocità elevata, conviene utilizzare la logica CMOS. Se invece non interessa il consumo elevato e si desidera avere un'alta velocità, anche a scapito dello spazio occupato, conviene utilizzare la logica TTL.
Il transistor è un dispositivo elettronico realizzato mediante materiale semiconduttore che, solitamente, è il **silicio**. A causa della sua particolare struttura microscopica interna si presta molto bene alla realizzazione di dispositivi elettronici a semiconduttore. Questo materiale fornisce la possibilità di costruire elementi di dimensioni ridottissime e non conduce facilmente gli elettroni come i metalli ma presenta una certa inerzia alla conduzione, occorre cioè fornire una certa energia agli elettroni perché si possano liberare.
Il transistor può lavorare in diverse condizioni: in base a queste viene utilizzato nelle diverse applicazioni. Nel caso dei circuiti elettronici digitali viene impiegato come *interruttore On-Off*, ossia transistor *acceso* o transistor *spento*.
Vi sono infinite possibilità di costruzione di circuiti elettronici, ma solo alcune sono fondamentali. I circuiti di base sono quelli che computano i connettivi logici fondamentali NOT, AND, OR, in quanto permettono di calcolare tutte le funzioni binarie di variabili binarie. Tali circuiti di base sono denominati **porte logiche**.
Il connettivo NOT è rappresentato dalla seguente tavola di verità

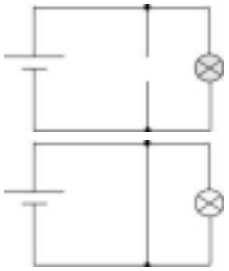
x	NOT(x)
0	1
1	0

Il circuito che svolge la funzione NOT esegue la negazione dell'ingresso. Ad esempio, se l'ingresso è allo stato 1 l'uscita sarà a 0 e viceversa. È comunque possibile simulare il funzionamento di una porta NOT mediante una semplice batteria, una lampadina, un interruttore e dei fili metallici opportunamente collegati fra loro. L'interruttore può essere considerato come l'ingresso x e la lampadina come l'uscita NOT(x).



Se l'interruttore è aperto, lo stato logico corrispondente è 0, ossia la corrente elettrica non passa. Se, viceversa, l'interruttore è chiuso allora lo stato logico corrispondente è 1, ossia la corrente elettrica passa.
Se la lampadina è accesa, vuol dire che è attraversata da corrente ed il corrispondente livello logico è pari a 1. Se invece la lampadina è spenta, non è attraversata da corrente e il rispettivo livello logico è 0.

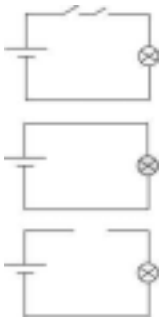
Descriviamo ora il funzionamento di tale circuito elettrico.



Se l'interruttore è aperto (x=0), la corrente passa attraverso i fili non interrotti e la lampadina si accende (NOT(x) = 1).
Se l'interruttore è chiuso (x=1), la corrente percorre la "strada più breve" e "più comoda" (ossia dove la resistenza elettrica è più bassa), di conseguenza la corrente stessa non arriva alla lampadina che rimane spenta (NOT(x)=0).
In definitiva, tale semplice circuito elettrico simula il comportamento di una porta NOT.
Il connettivo AND rappresenta la congiunzione ed è una funzione di due variabili definita dalla seguente tavola di verità

x	y	AND(x, y)
0	0	0
0	1	0
1	0	0
1	1	1

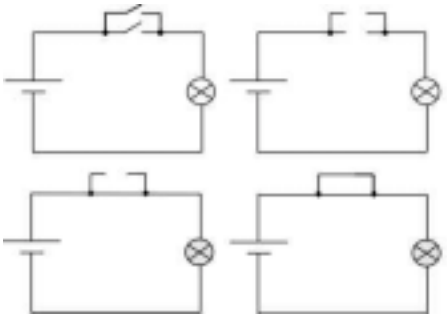
(ha come risultato il valore 1 solo se entrambi x ed y hanno valore 1, altrimenti assume valore 0)



In questo caso abbiamo bisogno di due interruttori per simulare il comportamento della porta logica AND che svolge la funzione omonima.
I due interruttori sono posti uno dietro l'altro. Affinché la lampadina si accenda (livello logico 1), entrambi gli interruttori dovranno essere chiusi (x=1, y=1). Infatti in questa situazione il circuito è chiuso e la corrente passa.
In tutti gli altri casi (un interruttore chiuso e l'altro aperto e viceversa, o entrambi aperti) i fili di collegamento tra la batteria e la lampadina risulteranno interrotti e quindi non ci sarà passaggio di corrente elettrica. Di conseguenza la lampadina rimarrà spenta (stato logico 0).
Il connettivo OR rappresenta la disgiunzione ed è una funzione di due variabili definita dalla seguente tavola di verità

x	y	OR(x, y)
0	0	0
0	1	1
1	0	1
1	1	1

(ha come risultato il valore 1 solo se almeno uno fra x ed y ha valore 1, altrimenti assume valore 0)



Un discorso analogo vale anche per tale circuito logico fondamentale. In questo caso i due interruttori sono collegati tra loro ("in parallelo"), in modo tale che la lampadina rimanga spenta solo se entrambi sono aperti. Di conseguenza basta che uno dei due tasti sia chiuso, affinché in uscita si abbia uno stato logico pari a 1 (lampadina accesa).
Infatti se entrambi gli interruttori sono aperti (x=0 e y=0), la corrente non passa poiché il circuito è aperto e la lampadina rimane spenta (OR(0,0)=0).
Se ad esempio un interruttore è aperto e l'altro è chiuso (x=0 e y=1), la corrente passa poiché il circuito è chiuso (OR(0,1)=1). Si ottiene lo stesso risultato (lampadina accesa) se entrambi i tasti sono chiusi (x=1, e y=1).

Le porte logiche fondamentali possono essere realizzate collegando opportunamente dei transistor che lavorano nel funzionamento On-Off. In tal caso questi ultimi agiscono come delle specie di interruttori elettrici controllati elettronicamente.

Il transistor è un dispositivo a tre terminali, uno di ingresso (input) e due di uscita (output), al quale sono connessi opportunamente tre fili.
Nella pratica un transistor regola l'eventuale passaggio di elettroni attraverso i fili di output, definendo così un segnale elettronico binario (1 se passano elettroni, 0 se non passano).
Nel dettaglio le lettere binarie (0 e 1) sono rappresentate nei fili dai due stati fisici seguenti: per convenzione il filo trasporta e/o memorizza la lettera 0 se il suo potenziale elettrico è GND = 0 volt, la lettera 1 se il suo potenziale elettrico è V_{DD} = 3 volt (cinque anni fa era V_{DD} = 5 volt e fra qualche anno sarà V_{DD} = 2,5 volt; la riduzione di tale intensità viene ricercata sostanzialmente per questioni di velocità e consumo di energia). V_{DD} rappresenta la tensione di alimentazione, ossia l'energia necessaria al transistor per poter funzionare, mentre GND (ground) è il punto a più basso potenziale.
Il segnale elettronico di un transistor può essere usato per comandare un altro transistor e ottenere un nuovo segnale elettronico che possiamo utilizzare per comandare un altro transistor e così via. L'insieme di questi interruttori elettronici che si comandano a vicenda viene detto circuito elettronico o gate booleano.
L'hardware di una macchina è costituito per lo più da circuiti digitali binari, ovvero da circuiti elettrici ottenuti collegando opportunamente alcuni transistor ed in cui gli stati fisici possibili sono soltanto due (quelli già descritti). In questo contesto i transistors permettono di manipolare tali informazioni "binarie", mentre i fili elettrici le memorizzano oppure le trasportano da un transistor all'altro.

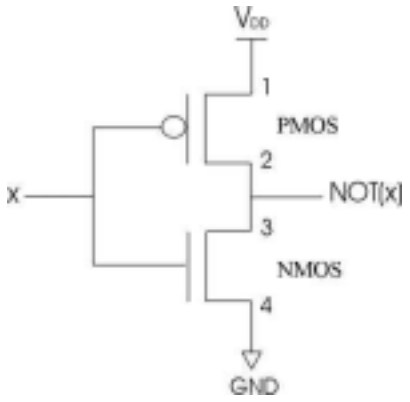
Questo è il punto di contatto tra hardware e software:

l'hardware è costituito da fili elettrici, il software dallo stato fisico dei fili elettrici.

Esaminiamo ora il funzionamento On-Off di un transistor MOS.
Esistono due tipi di transistor MOS:

Transistor MOS a canale N (NMOS)	filo di input 0 → fili di output sconnessi filo di input 1 → fili di output connessi	
Transistor MOS a canale P (PMOS)	filo di input 0 → fili di output connessi filo di input 1 → fili di output sconnessi	

Di seguito illustriamo brevemente un esempio di gate booleano, detto "inverter", della tecnologia CMOS (Complementary Metal Oxyde Semiconductor) utilizzato per computare il connettivo logico NOT. Tale circuito utilizza un NMOS e un PMOS, ossia una coppia di transistor complementari.



Gli input dei due transistor sono connessi da un filo che trasporta il valore di input x (che può essere 0 o 1). L'output 1 del transistor PMOS (quello sopra in figura) è a V_{DD} volts (valore binario 1),

l'output 4 del transistor NMOS (quello sotto in figura) è a GND volts (valore binario 0). Gli altri due fili di output, ovvero l'output 2 del transistor PMOS e l'output 3 del transistor NMOS, sono connessi tra loro in un filo che trasporta il filo di output NOT(x).
Se l'input x è 0 il transistor PMOS ha i suoi fili di output connessi mentre il transistor NMOS ha i suoi fili di output sconnessi, quindi l'output dell'inverter (NOT(x)) è connesso con l'output 1 del transistor PMOS e quindi è al suo stesso potenziale elettrico V_{DD}, ovvero trasporta il valore 1.
Se invece l'input x è 1, è il transistor NMOS ad avere i suoi fili di output connessi, mentre il transistor PMOS ha i suoi fili di output sconnessi, quindi l'output dell'inverter è connesso con l'output 4 del transistor NMOS e quindi è al suo stesso potenziale elettrico GND ovvero trasporta il valore binario 0.

La tecnologia VLSI

Tutti i più moderni computer sono realizzati con una tecnologia che prende il nome di VLSI (Very Large Scale Integration) che permette di miniaturizzare i circuiti elettrici.
In VLSI tutti i circuiti elettrici costituenti la logica di un computer sono suddivisi in vari pezzettini rettangolari di silicio detti chip. Essi vengono ricavati mediante una particolare tecnica che consiste nel tagliare delle fettine da una barra di silicio purissimo. Tali chip sono connessi tra loro da fili elettrici e situati su un circuito stampato o scheda (circuit board o scheda madre). Ogni chip può contenere anche centinaia di milioni di transistor (ad esempio, l'Intel Pentium IV Processor può contenerne fino a 55 milioni; per approfondimenti consultare la pagina web http:\\www.intel.com/pressroom/kits/quickreffam.htm) di dimensioni 0,13 μm (1 micron = 1 milionesimo di metro) e per una grandezza complessiva del chip fino a 1,5 cm^2. Tale parametro (0,13 μm) tende a diminuire con lo sviluppo della tecnologia VLSI e sono già allo studio processori prodotti con un processo inferiore: l'ultima frontiera è sullo studio di processi a 0,11 μm.
Essendo molto piccoli, i transistors coniugano bene alcune esigenze: da un lato rispondono velocemente ai "comandi" impartiti, così sono capaci di interrompere il flusso di elettroni nel giro di qualche miliardesimo di secondo, dall'altro consumano pochissima energia per funzionare.

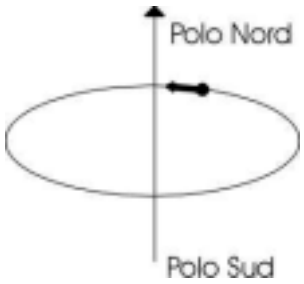
Memorie elettroniche

La memorizzazione di informazioni binarie si può ottenere con dispositivi elettronici che assomigliano a dei piccolissimi secchielli, dette celle, contenenti elettroni. La fase in cui la cella di memoria viene caricata viene detta scrittura della memoria. La carica accumulata nella cella viene utilizzata nella fase di lettura della memoria.
La velocità con cui é possibile caricare una cella di memoria di questo tipo é di poche decine di miliardesimi di secondo e anche se può sembrare una velocità enorme é uno dei limiti principali dei computer moderni. Infatti la velocità di interruzione di un transistor é di pochi miliardesimi di secondo e quindi la memoria impiega un tempo decine di volte maggiore.
Sono state comunque create memorie elettroniche più complesse e più veloci. Esse sono costruite con due transistor che si comandano reciprocamente. Se il primo conduce corrente comanda al secondo di condurre, quindi questo a sua volta conduce e manda cariche positive al primo. I due transistor si bloccano reciprocamente in questo stato indefinitamente, memorizzando l'ultimo segnale binario che é stato loro trasmesso. Insieme i due transistor formano una cella di memoria molto veloce che viene chiamata *flip-flop*.
Naturalmente il costo di questo tipo di memoria é molto maggiore del primo tipo a 'secchiello' in quanto composta da molti più elementi e viene utilizzata per le memorie interne all'unità di calcolo dell'elaboratore.
Un terzo tipo di memorie elettroniche è la memoria ROM (Read Only Memory). La ROM consiste di celle di memoria che si possono solo leggere e sono normalmente costituite da un filo di piccolissime dimensioni che può essere bruciato oppure no. L'informazione binaria viene ad essere

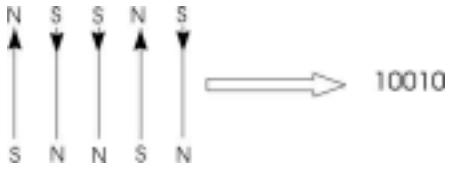
il fatto che passi o meno corrente nella cella: se il filo non è stato bruciato passa corrente e quindi l'informazione binaria contenuta sarà 1, se il filo è stato bruciato non passa corrente e l'informazione è 0.

Dispositivi magnetici: i dischi

Il magnetismo è la proprietà di alcune sostanze di orientare frammenti di ferro. Diciamo brevemente cosa si intende per campo elettrico e per campo magnetico. Un campo elettrico è un sistema di forze esercitato da una o più cariche elettriche capace di influenzare altre cariche nelle vicinanze, così anche un campo magnetico è un sistema di forze capace di esercitare la sua influenza in una certa regione di spazio.

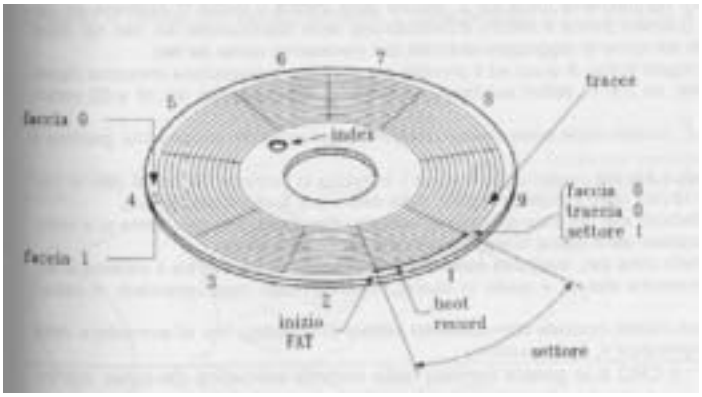


Vi è una stretta connessione tra le proprietà elettriche e le proprietà magnetiche: la variazione di un campo elettrico produce un campo magnetico, la variazione di un campo magnetico produce un campo elettrico. In termini qualitativi queste sono proprio le due leggi fondamentali dell'elettromagnetismo.
Un campo magnetico è caratterizzato da una direzione. Negli atomi la rotazione degli elettroni attorno al nucleo genera dei campi magnetici, ciascuno con una certa direzione nello spazio, ovvero con un polo sud ed un polo nord. La direzione dei campi dipende dal verso di percorrenza degli elettroni.
In natura è possibile costruire piccoli magneti, ovvero dispositivi magnetizzati, usando dei materiali detti ferromagnetici.
Nei dispositivi magnetici digitali vi sono dei piccoli magneti che vengono fatti orientare in una direzione od in quella opposta (cioè facendo ruotare le particelle in una delle due direzioni) ottenendo così due diversi orientamenti interpretabili come segnale binario, in altri termini a ciascuno si può associare un bit di informazione. Una magnetizzazione verso l'alto (come in figura) darà un segnale binario 1, una magnetizzazione verso il basso darà una segnale binario 0.



Dato che non é possibile costruire interruttori magnetici, cioè sistemi magnetici che producano la variazione di un segnale magnetico, i segnali binari magnetici non vengono utilizzati per l'elaborazione dell'informazione digitale: è necessario trasformare un segnale digitale magnetico in uno elettronico prima di poterlo elaborare.
I segnali binari magnetici vengono utilizzati solo per la memorizzazione, infatti è possibile ottenere la magnetizzazione di zone molto piccole e conservare la direzione dei campi magnetici (nei materiali ferromagnetici) per molti anni.
Nei moderni calcolatori l'informazione digitale magnetica viene depositata su dischi in continua rotazione. Su questi dischi la testina di lettura e di scrittura é libera di muoversi a distanze diverse dal centro del disco percorrendo cerchi più o meno ampi detti *tracce*. Su ogni traccia di un disco vengono memorizzati moltissimi bit, é quindi necessario suddividere una traccia in tante parti chiamate *settori*.

Secondo la convenzione normalmente utilizzata, la faccia superiore di un disco (o di un elemento in una pila di dischi dell'hard), viene considerata come la *faccia 0*, mentre la inferiore è la *faccia 1*. In un hard disk la faccia 0 del primo disco della pila e la 1 dell'ultimo, non vengono generalmente utilizzate per la collocazione dei dati.



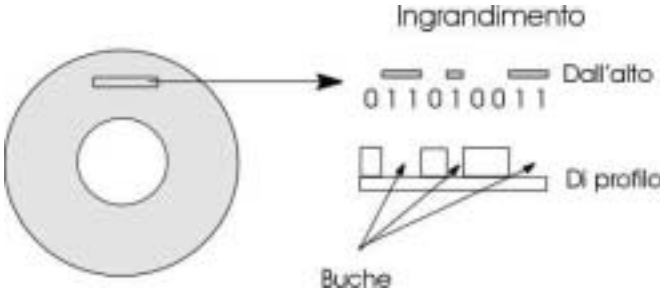
La **formattazione** è un'operazione che rende utilizzabile un disco, in quanto vi inserisce gli elementi per individuare le tracce, i settori ed i dati.
Durante la formattazione vengono definite 4 zone:

1. *Zona di avviamento* (boot record): occupa un settore, precisamente il settore 1 della traccia 0; reca le informazioni per la individuazione del sistema operativo dell'elaboratore che gestisce la memoria.
2. *FAT* (File Allocation Table): contiene gli elementi della formattazione (numero tracce e settori) e l'indicazione della distribuzione dei dati nel disco (settore d'inizio), individuati dal nome di raggruppamento dei dati medesimi (nome del file).
3. *Directory*: rappresenta l'indice dei dati contenuti sul disco e li individua in funzione del nome dato al file, della lunghezza del file, ….
4. *Spazio dati*: i settori non dedicati alle tre zone precedenti, vengono riservati ai dati.

Il settore é un blocco di bit che viene letto e scritto contemporaneamente. Infatti su di un disco non conviene leggere il singolo bit ma il gruppo di bit compreso in un settore. Questa tecnica condiziona, tra l'altro, l'intero funzionamento del calcolatore.

Ottica: laser e dischi ottici

Una delle forme più semplici di rappresentazione binaria é quella ottica.
La luce é un'onda elettromagnetica senza peso ma può essere considerata comunque composta da particelle (quanti di luce o fotoni).
Naturalmente la luce non può essere facilmente conservata, cioè conservata in un luogo, ma può essere usata per cambiare lo stato di un altro dispositivo binario di tipo elettronico. Sappiamo infatti che la luce può essere assorbita da un elettrone. Un elettrone la cui energia aumenta per effetto del fotone assorbito può rendersi libero del legame con i protoni del nucleo ed essere quindi libero di muoversi e produrre un flusso di corrente.

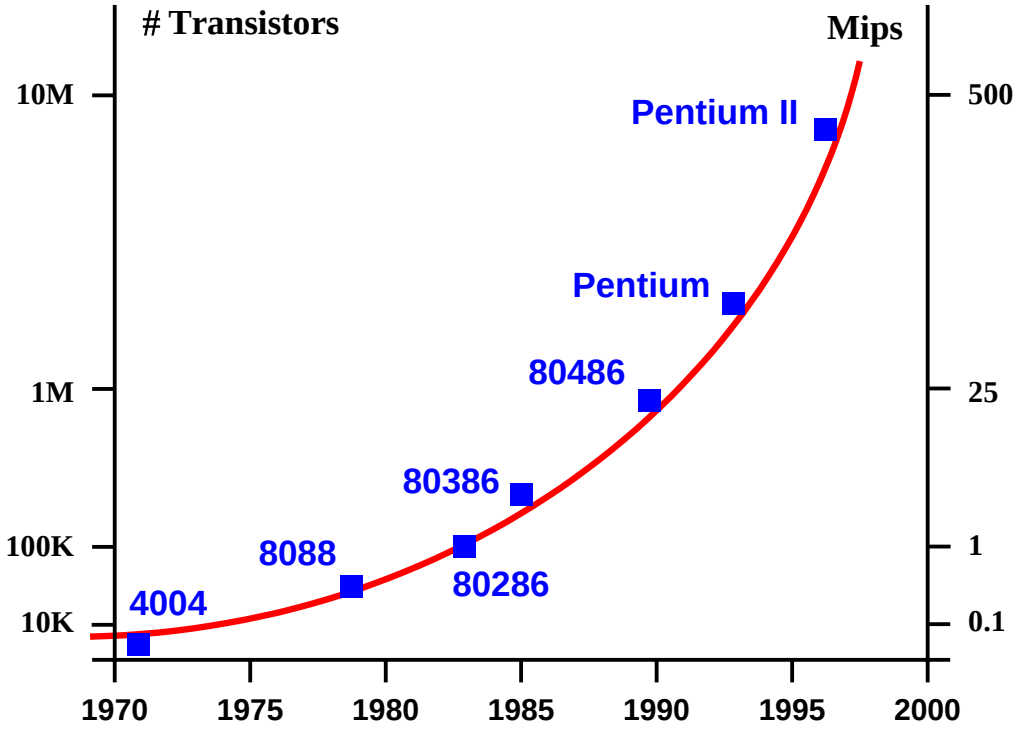


I dispositivi ottici per eccellenza sono oggi dei dischi sui quali l'elemento binario é di fatto una piccola buca (denominata *pit*) scavata da un laser nella stagnola racchiusa nella plastica del disco
Se esiste una buca lo stato dell'elemento binario é 1 se la buca non é stata fatta lo stato é 0. Naturalmente nel CD le buche sono disposte lungo un percorso a spirale che viene letto da un fascio luminoso molto sottile prodotto da un laser a bassa potenza. Durante la riproduzione il fascio luminoso percorre la spirale se incontra una buca non viene riflesso, se invece la buca non c'é il fascio viene riflesso. In questo modo la presenza o

meno della riflessione rappresenta un sistema binario il cui stato può essere rilevato da un sensore di luce che trasforma il segnale luminoso binario in un sistema elettrico binario.
Anche nel caso di segnali ottici si ha dunque la trasformazione in segnali elettronici prima di poter essere elaborati. Sono quindi difficilmente costruibili calcolatori ottici.

Legge di Moore

Nel 1965 Gordon Moore elaborò una tesi analizzando il trend evolutivo dei chip. Oggi, fatte le debite correzioni, questa legge afferma che:
"ogni chip ha una capacità circa doppia rispetto al suo predecessore e ogni 18-24 mesi nasce una nuova generazione di chip"
Questa legge, come si vede dal grafico riportato, è applicabile con discreta precisione ai processori e può quindi essere considerata un buon strumento di pianificazione per le industrie microelettroniche.
Vi sono due corollari significativi della legge di Moore:
Corollario di Machrone: "il computer che vuoi (o che ti dicono che devi assolutamente comperare) costa sempre la stessa cifra nonostante gli anni passino".
Corollario di Rock: "l'ammontare degli investimenti che vanno impiegati per costruire semiconduttori raddoppia ogni quattro anni".
Il primo corollario sta forse un po' crollando perchè ormai con un PC "economico", da mille euro, si possono fare benissimo tutte le applicazioni a livello Office e si può navigare in rete senza particolari problemi. Il secondo corollario può contribuire a spiegare con quale misura aumentano oggi gli investimenti delle industrie di processori, che hanno bisogno di capitali sempre maggiori per portare avanti i loro progetti, e quanto dunque è necessario in termini di investimenti per rispettare la crescita ipotizzata (e fin qui rispettata) da Moore.



Nell'articolo che segue, pubblicato dal quotidiano "La Repubblica" nel dicembre 2000, si riportano gli effetti, le conseguenze e la sua revisione odierna, nonché gli umori che la legge di Moore porta con sé (e che forse il solo enunciato non evidenzia appieno).

Intel ha annunciato il transistor più piccolo del mondo che conferma la previsione sul raddoppio della potenza dei pc

Chip 7 volte più veloci: resiste la legge di Moore

di RICCARDO STAGLIANO' (La Repubblica) – 12 dicembre 2000

ROMA - Se il vecchio Gordon Moore si fosse sbagliato, adesso il giornale elettronico che state leggendo non esisterebbe e non saremmo qui a raccontarvi le "magnifiche sorti e progressive" dell'era digitale. E invece la previsione del cofondatore dell'Intel sul "raddoppio, ogni anno, della capacità di calcolo dei microprocessori" ha sostanzialmente tenuto - consentendo ai pc di diventare sempre più potenti, piccoli ed economici - e continuerà a farlo anche nei prossimi 5-10 anni.

Questo, almeno, hanno assicurato ieri, a San Francisco, gli ingegneri dell'Intel in occasione della presentazione del **transistor più piccolo del mondo che misura 30 nanometri** (ovvero 30 miliardesimi di metro) **e costituirà la base per nuovi chip che potranno contenerne ciascuno sino a 400 milioni e "gireranno" alla sconcertante velocità di 10 GigaHertz**. Per farsi un'idea della scala dell'evoluzione basti pensare alle attuali specifiche del più veloce nato di casa Intel, il Pentium 4, che contiene 42 milioni di transistor e ha una frequenza di 1,5 GigaHertz.

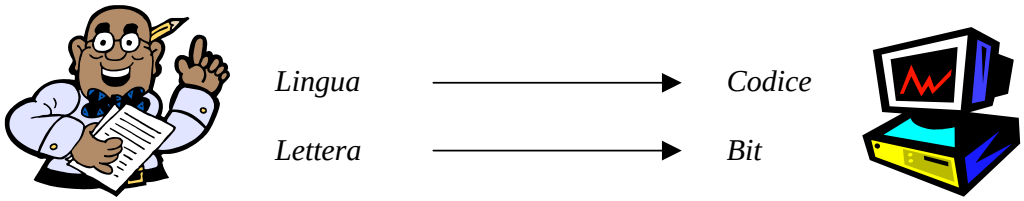
"Cosa si riuscirà a fare con computer dai motori così potenti è ancora difficile prevedere", spiega l'ingegner Mario Guarnone, business development e manager di Intel Italia, "ma le potenzialità sono enormi: ad esempio si potranno tradurre conversazioni da una lingua all'altra in tempo reale o setacciare, a velocità oggi inimmaginabili, banche dati molto complesse rintracciando al volo l'informazione che ci serve".

Ma l'aumento di velocità non va inteso in maniera lineare, come quello delle automobili. "Non significa necessariamente - prosegue Guarnone - che i pc, passando da 1,5 a 10 Ghz, eseguiranno le medesime operazioni con una rapidità 7 volte superiore, perché una buona parte della potenza supplementare viene assorbita da funzioni che prima non era nemmeno possibile compiere", come la visualizzazione di video a risoluzioni altissime, la resa della grafica e altri calcoli particolarmente pesanti.

Ma l'affermazione di ieri, in California, è importante soprattutto per la conferma della validità della "legge di Moore", ignota al grande pubblico ma essenziale per lo sviluppo della società internettiana. A più riprese, negli anni, la sua tenuta era stata messa in dubbio. Nel 1965, quando Moore l'aveva enunciata, aveva parlato di raddoppio ogni 12 mesi. Con il passare del tempo l'ottimismo aveva dovuto essere ritoccato al ribasso e l'arco di tempo era passato a 18 mesi. Oggi, nella pagina che il sito dell'Intel le dedica, si parla di "un trend avveratosi in maniera rimarcabilmente accurata" ma spostando l'intervallo ogni 18-24 mesi. **Si diceva che i transistor non avrebbero potuto continuare a rimpicciolirsi all'infinito, che si sarebbe raggiunta una barriera fisica che avrebbe smentito la legge. Per altri due lustri, almeno, non sarà così** e i pc del Natale 2001 faranno sembrare - come al solito - quelli delle imminenti festività delle ansimanti caffettiere.

Il linguaggio binario

Per realizzare il trattamento delle informazioni all'interno di un elaboratore e per permettere la comunicazione tra uomo ed elaboratore senza ambiguità occorre definire un linguaggio proprio per l'elaboratore ed un sistema di conversione dal linguaggio umano al linguaggio digitale binario dell'elaboratore.
Un **codice** è, in generale, una legge di rappresentazione che assegna, univocamente, ad ogni simbolo del linguaggio originario un simbolo o una sequenza di simboli del nuovo linguaggio. La trasformazione del simbolo originario a quello nuovo è chiamato *codifica*, la trasformazione inversa è detta *decodifica*.
Per motivi tecnologici, come visto nei precedenti paragrafi, i dati manipolati e memorizzati in un computer sono gestiti da dispositivi che possono presentarsi in due stati distinti. Ai due stati sono convenzionalmente associati i valori 0, 1 per cui l'alfabeto del calcolatore è costituito dai simboli {0, 1} ed il singolo elemento di informazione viene chiamato **Bit**, acronimo di Binary digiT (ovvero cifra binaria). Infine ogni *parola* dell'elaboratore sarà determinata da una sequenza su tale alfabeto (sequenze binarie).
Dunque, dal punto di vista logico:



La corrispondenza tra la nostra lingua ed il codice macchina binario avviene assegnando ai nostri caratteri, numeri, segni di punteggiatura, ecc. una sequenza ordinata di bit, ovvero il Byte, costituita da una sequenza ordinata di 8 bits (2^3 bits). Dunque sebbene il bit rappresenta l'unità di misura di base, è il byte ad esprimere più compiutamente una misura sulla quantità di dati.
Le altre unità di misura sono:
1 Kilobyte (1 Kb) = 1024 bytes = 2^{10} bytes = 2^{13} bits
1 Megabyte (1 Mb) = 1024 Kilobytes = 2^{10} Kilobytes = 2^{20} bytes = 2^{23} bits
1 Gigabyte (1 Gb) = 1024 Megabytes = 2^{10} Megabytes = 2^{30} bytes = 2^{33} bits
1 Terabyte (1 Tb) = 1024 Gigabytes = 2^{10} Gigabytes = 2^{40} bytes = 2^{43} bits
Nel seguito vedremo alcuni esempi di codici utilizzati nella pratica quotidiana degli elaboratori.

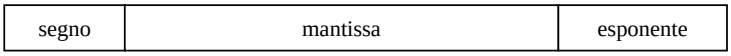
Rappresentazione dei numeri interi.
Per il trattamento dei dati numerici i calcolatori utilizzano di solito un numero fisso di bit (o di byte) per la rappresentazione dei numeri (*lunghezza di parola* fissa). Di norma, il segno di un numero è codificato dal primo bit del numero stesso: "0" rappresenta il segno positivo, "1" rappresenta il segno negativo. I numeri positivi sono rappresentati utilizzando la loro codifica nel sistema di numerazione binaria (al più aggiungendo in testa al numero binario alcuni zeri non significativi), mentre per la rappresentazione dei numeri negativi si utilizza il criterio del "complemento a 2": si prende la conversione binaria del numero (preso al positivo) e la si trasforma sostituendo "0" ad "1" e viceversa e facendo la somma con "1" alla fine.
Supponiamo di utilizzare parole di 2 byte (16 bit) per rappresentare i numeri interi con segno.
I 16 bit a disposizione vengono così utilizzati: 1 bit per il segno, 15 bit per la grandezza del numero. In tal modo possiamo rappresentare tutti e soltanto i numeri interi minori di $2^{15}-1 = 32767$ e maggiori di $-2^{15} = -32768$. Facciamo qualche esempio.

Esempio 1. Calcolare la rappresentazione interna del numero intero “621”.
Trasformando nel sistema di numerazione binaria si ha 621 = 1001101101; per arrivare a 15 bit occorre aggiungere cinque zeri in testa al numero e quindi aggiungere il bit del segno, in questo caso “0” perché il numero è positivo. In conclusione 621 si rappresenta all’interno dell’elaboratore con “0000001001101101”.

Esempio 2. Calcolare la rappresentazione interna del numero intero “–6569”.
Si trasforma nel sistema di numerazione binaria: 6569 = 1100110101001, quindi
- si aggiungono gli zeri necessari per raggiungere 15 bit … → 001100110101001;
si effettua il “complemento a 2”:
- si scambiano gli zeri con gli uni e viceversa … → 110011001010110
- si addiziona “1” … → 110011001010111;
infine si aggiunge il bit del segno (“1” poiché è negativo) … → 1110011001010111.
(Da notare che l’aggiunta del bit del segno può avvenire anche quando si aggiungono gli zeri per raggiungere la lunghezza della parola: in tal caso, però, si aggiunge uno zero - in quanto il numero, a quello stadio, è positivo - e con la complementazione viene trasformato in 1)

Rappresentazione dei numeri reali.
Anche se l’elaborazione dei dati presenta, di solito, operazioni su numeri reali, la rappresentazione di tutti questi numeri in maniera esatta costituisce un limite irrangiugibile per un calcolatore. I calcolatori sono in grado di rielaborare solo sequenze di cifre di lunghezza finita, e quindi i numeri reali vengono trasformati in numeri razionali per mezzo di un valore approssimato (che consiste nel determinare il grado di precisione, ovvero il numero di cifre significative: ad esempio il numero reale 12,3678416654375555… approssimato fino alla sesta cifra significativa fornisce il numero razionale 12,3678).
Esistono due modalità per la rappresentazione di valori numerici (numeri razionali) su un processore digitale: in virgola mobile (floating point) o in virgola fissa (fixed point).

La rappresentazione in **virgola mobile** avviene secondo la notazione scientifica: $z = m \times b^n$ dove
z = numero in virgola mobile, m = mantissa, b = base del sistema di numerazione, n = esponente. Esiste, inoltre, una rappresentazione in virgola mobile detta “normalizzata” che si ottiene quando $1/b \le (m)_{10} < 1$, in tal caso l’esponente n è detto anche *caratteristica*.
Ogni calcolatore riserva una sequenza fissa di celle di memorie per mantissa ed esponente e poiché i calcoli si svolgono tutti nella stessa base, questa non viene memorizzata.



Supponendo di avere a disposizione parole di 4 byte, la rappresentazione interna di un numero reale utilizza i 32 bit in questo modo: un bit per il segno, 7 bit per l’esponente (la caratteristica), 24 bit per la mantissa. In tal caso i numeri rappresentabili possono avere 24 cifre binarie significative e un ordine di grandezza compreso tra $2^{-2^{7-1}+1} = 2^{-63}$ e $2^{2^{7-1}-1} = 2^{63}$.

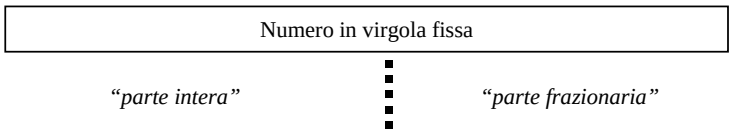
Esempio 1. Calcolare la rappresentazione in virgola mobile normalizzata del numero “102,5679” (con 16 bit di cui 1 per il segno, 11 per la mantissa e 4 per l’esponente)
Si trasforma il numero nel sistema binario: 102,5679 = 1100110.10010001…
- si prendono le 11 cifre più significative (da sinistra a destra) … → 1100110.1001;
- si normalizza … → $.11001101001 \times 2^7$
- si trasforma l’esponente in binario (utilizzando 4 bit) $7 \rightarrow 111 \rightarrow 0111$
Quindi: segno + mantissa + esponente = “0” + “11001101001” + “0111” = 0110011010010111 .

Osservazione. Qualora si renda necessaria l'aggiunta di zeri per il completamento dei bit occupati dalla mantissa, questi vanno aggiunti alla destra delle cifre significative del numero.

Esempio 2. Calcolare la rappresentazione in virgola mobile normalizzata del numero "-53,281" (con 16 bit di cui 1 per il segno, 11 per la mantissa e 4 per l'esponente)
Si trasforma il numero in modulo nel sistema binario: 53,281 = 110101.01000111…
- si prendono le 11 cifre più significative (da sinistra a destra) … → 110101.01000;
- si normalizza … → $.11010101000 \times 2^6$
- si effettua il complemento a due … → 00101011000;
- si trasforma l'esponente in binario (utilizzando 4 bit) 6 → 101 → 0101

Quindi: segno + mantissa + esponente = "1" + "00101011000" + "0101" = 1001010110000101 .

La rappresentazione in **virgola fissa**, invece, avviene con sequenze di cifre (in un sistema posizionale) senza una esplicita virgola di suddivisione: la sua posizione viene ricavata dalla dichiarazione per convenzione. Infatti oltre ad assegnare la lunghezza complessiva della parola, si stabilisce anche la posizione a cui corrisponde la virgola decimale.
Ad esempio, se la lunghezza di parola è 8, con 5 cifre destinate alla parte intera e 3 cifre destinate alla parte frazionaria, allora il numero binario (positivo) 1011.01 viene rappresentato con la sequenza 01011010 (senza virgole, aggiungendo uno zero all'inizio della parte intera a completamento dei 5 bit per il segno ed uno zero alla fine della parte frazionaria a completamento dei 3 bit).



Analogamente, per rappresentare il numero decimale –13,5 nella stessa convenzione (ed utilizzando il complemento a due in quanto è un numero negativo) si procede nel seguente modo:
- conversione $(13,5)_{10} = (1101.1)_2$
- aggiunta degli zeri mancanti (1 a sinistra, 2 a destra) … → 01101100
- complemento a due … → 10010011 + 1 → 10010100

Esempio 1. Calcolare la rappresentazione in virgola fissa del numero "102,5679" (con 16 bit di cui 7 per la parte frazionaria)
Da osservare subito che per la parte intera sono disponibili 16 – 7 = 9 bit, dunque 8 bit per la parte intera ed 1 per il segno.
Si trasforma il numero nel sistema binario: 102,5679 = 1100110.10010001…
- si prendono le cifre che ci interessano … → 1100110.1001000;
- si completa con gli zeri mancanti (1 a sinistra) … → 011001101001000;
- si aggiunge il bit del segno a sinistra … → 0011001101001000 .

Esempio 2. Calcolare la rappresentazione in virgola fissa del numero "-53,281" (con 16 bit di cui 7 per la parte frazionaria)
Si trasforma il numero in modulo nel sistema binario: 53,281 = 110101.01000111…
- si prendono le cifre che ci interessano … → 110101.0100011;
- si completa con gli zeri mancanti (2 a sinistra) … → 001101010100011;
- si effettua il complemento a due … → 110010101011101;
- si aggiunge il bit del segno … → 1110010101011101;

Quindi: segno + mantissa + esponente = "1" + "00101011000" + "0101" = 1001010110000101 .

La rappresentazione più utilizzata è quella in virgola fissa, che generalmente consente di eseguire le operazioni in tempi più rapidi (molti processori, tra l'altro, supportano solamente operazioni in virgola fissa a livello dell'hardware, eventuali operazioni in virgola mobile devono essere realizzate a livello del software). Tuttavia ciò comporta anche qualche svantaggio:
– il range della rappresentazione è molto più piccolo rispetto al corrispondente in virgola mobile;
– richiede una implementazione più accurata.

Osservazione. Le operazioni, per via dello spazio limitato e predefinito su ogni calcolatore destinato alla rappresentazione numerica, possono dar luogo ad errori di overflow. Ciò accade quando si oltrepassa il range disponibile per la rappresentazione del risultato, ovvero quando l'operazione produce un riporto non rappresentabile da nessun bit disponibile.

Rappresentazione dei caratteri alfanumerici.
Nell'elaborazione di dati compaiono anche caratteri non numerici, ovvero lettere, caratteri speciali, simboli di comando. A tutti questi caratteri vengono associate comunque sequenze di bit appartenenti ad un insieme finito detto *set*.
L'ampiezza di un set di caratteri (ovvero il numero di simboli diversi) dipende dalla lunghezza massima delle sequenze utilizzate nella rappresentazione. Ad esempio, avendo a disposizione 6 bit per carattere si possono rappresentare $2^6 = 64$ caratteri diversi, con 7 bit si rappresentano $2^7 = 128$ caratteri diversi.
L'ASCII (American Standard Code for Information Interchange) è un alfabeto molto diffuso ed è codificato di norma su 7 bit, l'alfabeto ha quindi una disponibilità di $2^7 = 128$ caratteri di cui 32 riservati ai simboli di comando, con un ultimo bit, l'ottavo, dedicato al controllo di parità (il numero degli 1 deve essere dispari). Dunque nella codifica ASCII: 1 carattere $\rightarrow$ 1 byte. La tabella che segue evidenzia la codifica ASCII; ad esempio:
il numero "6" viene codificato con 011, 0110 ed il bit di parità 1, dunque "6" $\rightarrow$ "01101101";
la lettera "k" viene codificata con 110, 1011 ed il bit di parità 0, dunque "k" $\rightarrow$ "11010110";
il simbolo speciale "$" viene codificato con 010, 0100 ed il bit 1, dunque "$" $\rightarrow$ "01001001";
il simbolo di comando "DEL" si codifica con 111, 1111 ed il bit 0, dunque "DEL" $\rightarrow$ "11111110".

ASCII "ridotto"	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	‘	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	TSX	DC2	„	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	x
1000	BS	CAN	(	8	H	X	h	y
1001	HT	EM	)	9	I	Y	i	w
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

(Significato dei simboli di comando: DEL = cancellare; EOF = fine della trasmissione;
LF = spaziatura verticale, CR = ritorno carrello; SP = spazio intermedio; ecc.)

Da osservare che nell'alfabeto ASCII la rappresentazione dei numeri avviene codificando le singole cifre; ad esempio: "285" → "2", "8", "5" → "01100100", "01110000", "01101011". Anche l'eventuale segno è rappresentato separatamente: "-7" → "-" , "7" → "01011011", "01101110".
Questo alfabeto permette di inserire tutti i tipi di carattere maiuscoli e minuscoli ma non i caratteri particolari dei vari paesi. Per risolvere problemi di questo tipo esistono soluzioni con possibilità di codifica più estese, ad esempio quella a 8 bit (ASCII esteso) che contiene 256 configurazioni binarie, quindi con possibilità di codificare un numero doppio di caratteri, per cui si possono rappresentare anche metafonie, ulteriori caratteri speciali, ecc.
Il supporto fisico principale con cui questi dati vengono memorizzati é il disco magnetico. Il supporto fisico con cui questi dati vengono memorizzati durante l'utilizzo da parte dell'elaboratore é la memoria elettronica. Il supporto fisico con cui queste informazioni vengono elaborate é l'insieme di interruttori elettronici con cui é costruito l'elaboratore.
Nei moderni sistemi operativi vengono usate sequenze a 16 bit in modo da avere 2^{16} = 65536 possibili simboli da rappresentare (si possono rappresentare anche i particolari caratteri delle varie lingue, come ad esempio gli ideogrammi giapponesi). Questo nuovo standard viene detto UNICODE.

Esercizi risolti

1. Considerare il numero reale r = -13,69. Determinare:
a) La rappresentazione in virgola fissa di r su 12 bit, di cui 6 per la parte frazionaria (utilizzando l'aritmetica con completamento a due);
b) La rappresentazione in virgola mobile normalizzata di r su 12 bit, di cui 7 per la mantissa.
Soluzione. In base due, *r* è espresso dalle cifre $r = (-1101.101100001...)_2$, dunque:
a) r = -1101.101100001... → -1101.101100 (solo 6 cifre "frazionarie") → -01101101100 (5 cifre "intere") → -10010010100 (complemento a due) → 110010010100 (bit del segno);
b) osserviamo preliminarmente che la rappresentazione richiesta è formata da 1 bit per il segno, 7 bit per la mantissa e 4 bit per l'esponente; dunque: r = -1101.101100001... → -1101.101 (7 cifre per la mantissa) → -0.1101101×2^4 (in forma normalizzata) → 10010011 (complemento a due + bit del segno) → 100100110100 (aggiungendo l'esponente 4→100→0100).

2. Considerare il numero reale r = 21,27. Determinare:
a) La rappresentazione in virgola fissa di r su 16 bit, di cui 7 per la parte frazionaria;
b) La rappresentazione in virgola mobile normalizzata di r su 16 bit, di cui 10 per la mantissa.
Soluzione. In base due, *r* è espresso dalle cifre $r = (10101.010001010001...)_2$, dunque:
a) la rappresentazione richiesta è composta da 1 bit per il segno, 8 bit per la parte intera, 7 bit per la parte frazionaria; dunque: r = 10101.010001010001... → 00010101.0100010 (completando con 3 zeri la parte intera e tagliando la parte frazionaria a "solo" 7 cifre) → 0000101010100010 (bit del segno);
b) la rappresentazione richiesta è formata da 1 bit per il segno, 10 bit per la mantissa e 5 bit per l'esponente; dunque: r = 10101.010001010001... → 10101.01000 (10 cifre per la mantissa) → 0.1010101000×2^5 (in forma normalizzata) → 00101011000 (complemento a due + bit del segno) → 0010101100000101 (aggiungendo l'esponente 5→101→00101).

3. Ho a disposizione le seguenti aree di memoria: 8 Gb su un Hard disk, 2 Cd-Rom vuoti ciascuno da 650 Mb, 1 Mb e 120 Kb su un floppy disk. Quanta memoria ho a disposizione complessivamente (in termini di Kbytes)?
Soluzione: conteggiamo singolarmente la memoria libera di ciascun supporto, quindi sommiamo.
Hard disk → 8 Gb = 8 × 1024 Mb = 8192 Mb = 8192 × 1024 Kb = 8388608 Kb;

CD-Rom $\rightarrow$ 2 $\times$ 650 Mb = 1300 Mb = 1300 $\times$ 1024 Kb = 1331200 Kb;
Floppy disk $\rightarrow$ 1 Mb e 120 Kb = 1024 Kb + 120 Kb = 1144 Kb;
infine sommando: 8388608 Kb + 1331200 Kb + 1144 Kb = 9720952 Kb.

4. Quanti caratteri ASCII posso immagazzinare in 1 Megabyte?
Soluzione: 1 Mega = 2^{20} bytes = 1.048.576 bytes. Quindi posso immagazzinare 1.048.576 caratteri ASCII.

5. Se un file è composto da 1100 caratteri alfanumerici, di quanto memoria ho bisogno per "salvarlo"?
Soluzione: 1100 byte.

6. Il file "home.txt" è costituito dalla frase (le virgolette non fanno parte della frase stessa) ***"Ad ogni rinuncia corrisponde una contropartita considerevole."*** *Ripetuta 24 volte e con esattamente tre spazi fra le ripetizioni della frase. Quanto spazio occupa questo file in formato ASCII?*
Soluzione. La frase è formata da 61 caratteri, da ripetere 24 volte, quindi 61x24=1464. I tre spazi tra una frase e l'altra vengono ripetuti 23 volte (24-1), quindi 3x23=69 caratteri blank (caratteri di spaziatura). In tutto si avrà un numero di caratteri pari a 1464+69=1533. Poiché ad un carattere corrisponde un byte, lo spazio occupato dal file "home.txt" è pari a 1533 byte. Inoltre:

1 Kbyte=1024 byte

quindi sottraiamo tale valore a 1533 byte

1533 – 1024 = 509 byte

per cui il file occuperà 1 Kbyte e 509 byte.

7. Il file "pippo.txt" è costituito dalla frase (le virgolette non fanno parte della frase stessa) ***"She's just a cosmic girl, from another galaxy, my heart's at zero gravity."*** *Ripetuta 18 volte e con uno spazio fra le ripetizioni della frase. Quanto spazio occupa questo file in formato ASCII?*
Soluzione. La frase è formata da 74 caratteri, da ripetere 18 volte, quindi 74x18=1332. Lo spazio tra una frase e l'altra viene ripetuto 17 volte (18-1). Dunque in formato ASCII la frase occupa

1332 byte + 17 byte = 1349 byte = 1 Kbyte e 325 byte.