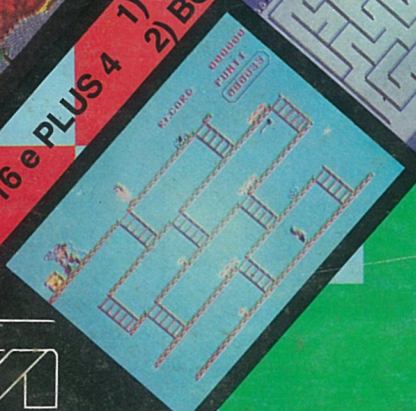
GO GAMES

mensile d'informatica e video-games - n. 4 - dicembre 1985 - L. 8000

CBM 64 1) STREGONA 3) LANDER 5) FLIGHTS OF MAN
2) MAXI GOLF 4) FENICE 6) THE LANDS 7) MAGICO BILL

C16 e PLUS 4 1) JET MAN 3) IL GORILLA
2) BOMBE 4) LABIRINTO



Editions Fermont s.r.l. - Milano - DISTRIBUZIONE: Messaggerie Periodici - Via G. Carcano, 32 - 20141 Milano
Pubblicazione registrata presso il Tribunale di Milano col N° 411 del 31.8.85

GO GAMES

Mensile di informatica
e video giochi

Anno I
N. 4 - Dicembre '85

EDITORE:
Editions Fermont s.r.l.
20121 Milano

REDAZIONE:
Via Cialdini, 11
20161 Milano
Tel. 02/6453775/6

FOTOLITO:
Fotolito RVM s.n.c.
Via Puricelli, 4
20147 Milano

COMPOSIZIONE:
Nuovo Gruppo Grafico s.r.l.
Frazione Venina, 7
20090 Assago (Mi)

STAMPA:
A.G.E.L. s.r.l.
Viale dei Kennedy, 92
20027 Rescaldina

DISTRIBUZIONE:
MePe
Via G. Carcano, 32
20141 Milano

DIRETTORE RESPONSABILE:
Amilcare Medici

Numeri arretrati: Ogni numero arretrato £. 8.000 più £. 3.000 di spese postali - Versamento da effettuare sul c/c postale n. 37332202 intestato a EDITIONS FERMONT, Via Cialdini, 11 20161 Milano

ATTENZIONE

In questo numero GO GAMES introduce una "novità". Infatti, per poter caricare i giochi, occorre digitare una "parola chiave", senza la quale i giochi non vengono caricati e Voi non potete.. divertirVi.
Caricare corettamente la "parola chiave" è semplice: seguite attentamente le nostre istruzioni e non sbaglierete.
Trovate la "parola chiave" nelle pagine seguenti, nella descrizione di ogni singolo gioco.

CBM 64

C'è un sistema di caricamento che permette di scegliere il gioco che volete caricare e di posizionare il nastro con l'avanzamento veloce.
Per quanto riguarda l'inserimento della parola chiave, è necessario innanzitutto caricare la testata della cassetta e, quindi, scegliere dal sommario il gioco desiderato.
A questo punto, il computer Vi "chiederà" la "parola chiave". Digitatela correttamente e, quindi, proseguite eseguendo le varie istruzioni che, di volta in volta, appariranno sul video del Vostro computer.
AVVERTIMENTO: se lo schermo si riempirà di righe colorate molto fitte, significa che il caricamento procede regolarmente. Se le righe colorate non appaiono, spegnete il computere e ripetete tutte le operazioni dall'inizio.

C16 / PLUS 4

1) **Posizionare il nastro all'inizio del gioco desiderato**
2) **Digitare LOAD e premere RETURN**
3) **Attendere il caricamento**
4) **Digitare RUN e premere RETURN**
5) **Digitare la "parola chiave" del gioco prescelto e premere RETURN**
6) **Attendere il caricamento del gioco**

l'uomo e il computer 2

Esordiamo con una citazione piuttosto dotta:

Annus erat, decimum cum luna receperat orbem:
hic numerus magno tunc in honore fuit,
seu quia tot digiti, per quos numerare solemus,
seu quia bis quino femina mense parit, ...

È Ovidio che parla (in **Fasti**, III) e ci spiega che dieci è un grande numero perchè è quello delle dita delle nostre mani, il numero che è servito ad imparare a contare e su cui gli uomini hanno basato il loro sistema di numerazione.
Se invece i classici non vi interessano ho subito pronta un'altra citazione:

Un biglietto rosso spuntava tra il tergicristallo e il parabrezza; con cura lo lacerai in due, quattro, otto pezzi.

Questa frase è stata tratta da **Lolita** di Vladimir Nabokov.
Ma che centra qui, e soprattutto che cos'ha a che vedere con l'altra frase di Ovidio?
Si parla in entrambe di numeri ma dietro ai due stili completamente differenti si celano altre implicazioni che mi azzardo a definire matematiche...
Abbiamo dieci dita, e quindi contiamo come ci suggerisce Ovidio... se invece ne avessimo solo due, se contassimo a **mani** invece che a dita, conteremmo come Nabokov conta i pezzi strappati del foglietto rosso. Non vi sarebbero più **decine** ma **coppie**.
La posizione delle cifre nel numero non dipenderebbe da potenze di dieci ma da potenze di due... tante quante sarebbero le cifre a nostra disposizione.
Il sistema non sarebbe più **decimale** ma **binario**.
Proviamo ad elencare i numeri successivi usando soltanto i simboli 0 e 1 a nostra disposizione: 1, 2... no, non c'è; allora quanto fa 1+1? 10, ovviamente, non è la nostra usuale **decina** bensì una **coppia**.
Proseguiamo: 10+1 = 11 (tutto OK), 11+1 = 100 (nel sistema decimale è una decina di decine, nel sistema binario è una coppia di coppie) in effetti corrisponde al decimale 4.
Sapreste ora continuare con la numerazione e dirmi, per esempio, a quale numero binario corrisponde il decimale 8?
Semplice, ricordate che 8 è una coppia al cubo quindi è uguale alla nostra decina al cubo cioè 1000:

decimale	**binario**
$10 = 10^1$	$10 = 2^1$
$100 = 10^2$	$100 = 2^2$
$1000 = 10^3$	$1000 = 2^3$

Ma se non volete impazzire con problemi di conversione tra i due diversi sistemi di numerazione eccovi pronto e confezionato un piccolo e semplice programma che trasforma istantaneamente un numero decimale in binario:

```
READY.
10 REM   CONVERSIONE DECIMALE/BINARIO
20 :
30 REM    DI UN QUALSIASI NUMERO DA
40 REM    ZERO A MILLE MILIARDI ...
50 :
60 REM      PER VIC 20 O CBM 64
70 :
80 :
100 PRINTCHR$(147)
110 PRINTCHR$(144)CHR$(17)
120 PRINT"NUMERO DECIMALE (0-10↑12)"
```

```
130 PRINT:INPUT N
140 IFN<0ORN>10↑12THENGOTO100
150 X$=""
160 UX=N
170 U1=INT(UX/2):J=UX-2*U1
180 J$=RIGHT$(STR$(J),1)
190 X$=J$+X$
200 K$=RIGHT$((B$+X$),40)
210 UX=U1
220 IFU1<>0THEN170
230 PRINTCHR$(17)CHR$(28) NUMERO BINARIO
="
240 PRINT:PRINTK$
250 PRINTCHR$(17)CHR$(31)
260 INPUT"UN'ALTRA CONVERSIONE";A$
270 IFA$="N"THENEND
280 IFA$="S"THEN110
290 IFA$<>"S"THEN260
READY.
```

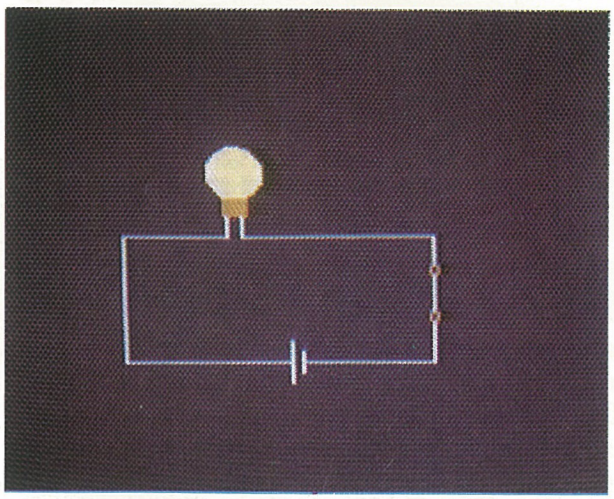
Dopo averlo accuratamente ricopiato e fatto partire (con RUN) sullo schermo apparirà un invito a scrivere un qualsiasi numero (fino a mille miliardi cioè fino a 1.000.000.000.000) e premendo RETURN... voilà, ecco il numero scritto nel sistema di numerazione binaria.
Tutto chiaro? Beh, prendiamoci comunque una pausa con un gioco che sembra di divinazione ma che invece ha molto a che fare con quello di cui stiamo parlando.

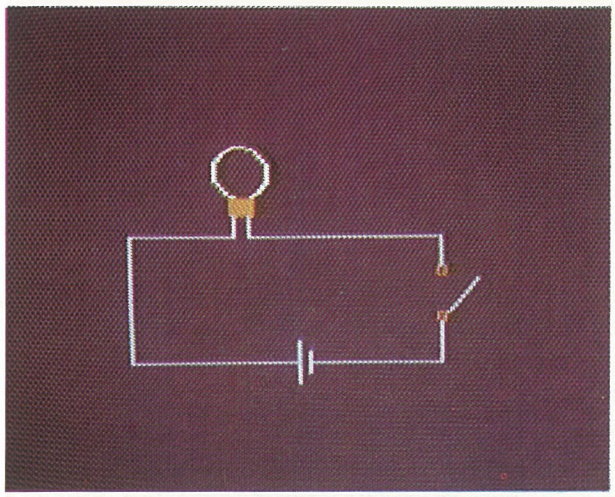
Osservate bene la tabella qui a lato:
dopo averla guardata bene e considerato che apparentemente in essa non vi è nulla di misterioso né tantomeno di magico proponiamo ad un amico questo **trucco**:
« Pensa un numero qualsiasi da uno a cento e non dirmelo, mostrami invece in quali tra queste sette colonne compare quel numero ».
A questo punto siete in grado di indovinare il numero pensato dal vostro amico.
Facciamo un esempio.
In questo momento ho pensato un numero e vi dico che si trova nella prima, nella seconda e nella quarta colonna. Come fate a indovinare il mio numero?
Facilissimo, sommate il primo termine delle colonne che vi ho detto: 1+2+8 = 11 ... il numero pensato era proprio 11.
Complimenti, siete stati bravissimi, ora provate con qualche altra persona. Dite loro di fare bene attenzione e di indicarvi **tutte** le colonne dove appare il numero.
Fine della pausa.

1	2	4	8	16	32	64
3	3	5	9	17	33	65
5	6	6	10	18	34	66
7	7	7	11	19	35	67
9	10	12	12	20	36	68
11	11	13	13	21	37	69
13	14	14	14	22	38	70
15	15	15	15	23	39	71
17	18	20	24	24	40	72
19	19	21	25	25	41	73
21	22	22	26	26	42	74
23	23	23	27	27	43	75
25	26	28	28	28	44	76
27	27	29	29	29	45	77
29	30	30	30	30	46	78
31	31	31	31	31	47	79
33	34	36	40	48	48	80
35	35	37	41	49	49	81
37	38	38	42	50	50	82
39	39	39	43	51	51	83
41	42	44	44	52	52	84
43	43	45	45	53	53	85
45	46	46	46	54	54	86
47	47	47	47	55	55	87
49	50	52	56	56	56	88
51	51	53	57	57	57	89
53	54	54	58	58	58	90
55	55	55	59	59	59	91
57	58	60	60	60	60	92
59	59	61	61	61	61	93
61	62	62	62	62	62	94
63	63	63	63	63	63	95
65	66	68	72	80	96	96
67	67	69	73	81	97	97
69	70	70	74	82	98	98
71	71	71	75	83	99	99
73	74	76	76	84	100	100
75	75	77	77	85		
77	78	78	78	86		
79	79	79	79	87		
81	82	84	88	88		
83	83	85	89	89		
85	86	86	90	90		
87	87	87	91	91		
89	90	92	92	92		
91	91	93	93	93		
93	94	94	94	94		
95	95	95	95	95		
97	98	100				
99	99					

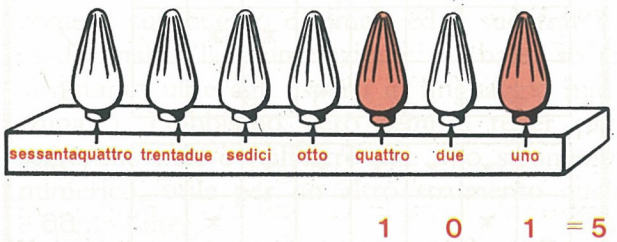
E il sistema binario che centra con questo gioco? Osservate bene la tabella. I primi numeri delle varie colonne sono le prime potenze di due. Poi i numeri si susseguono in questo modo: nella prima colonna un numero sì uno no (ovvero tutti i numeri dispari);

nella seconda colonna due numeri sì e due no; nella terza quattro numeri sì e quattro no e così via. Praticamente nella prima colonna vi sono tutti i numeri che nel sistema binario hanno per ultima cifra l'unità; nella seconda colonna quelli che hanno l'unità al penultimo posto; nella terza al terz'ultimo eccetera.
Anche se il vostro amico non se ne è reso conto, elencandovi le colonne dove si trova il numero da lui pensato vi ha fornito il numero binario corrispondente al decimale che aveva scelto.
Sento sorgere spontanea una domanda dal profondo dei vostri animi: **ma tutto ciò a cosa serve?** Cioè: abbiamo dieci dita, ci hanno insegnato la tavola pitagorica e a contare in base dieci e quindi perchè parlare di **notazione** (si dice così scientificamente) binaria?
Ma perchè questo è il modo in cui opera con i numeri (e non soltanto con questi) il nostro beneamato computer!!
Come è noto lui non possiede dieci dita (hanno provato anche a costruirne ma si chiamano ROBOT e ne parleremo il mese prossimo). Le uniche sue dita (se mi è concessa l'espressione metaforica) sono degli interruttori che possono essere accesi o spenti.
Acceso/spento, si/no, vero/falso, chiaro/scuro, vuoto/pieno... chi più ne ha più ne metta... si tratta sempre di una scelta binaria, che si può sempre e comunque ridurre ad uno **0** oppure ad un **1**.
Ecco finalmente rivelato (mi scusino i sapientoni) come funziona un computer:





La differenza tra queste due immagini (cioè che l'interruttore sia aperto o chiuso e che di conseguenza la lampada sia accesa o spenta) è la sola informazione che ci è data dal circuito. È la cosiddetta **unità di informazione** che ha preso il nome di **bit** (binary digit).
Moltiplichiamo i circuiti e potremo così rappresentare qualsiasi numero:



Questo è il principio, ma se sul nostro schermo televisivo ci sono migliaia di punti quanti circuiti dovrà contenere il computer? Grazie ai microprocessori, cioè alla miniaturizzazione dei circuiti su delle piastrine che viste al microscopio sembrano delle piante topografiche di una città, si è risolto il problema. Ma c'è un altro fatto: il calcolatore riconosce non un solo, misero, singolo, isolato **bit** cioè la lettera di un alfabeto binario, bensì quella che per lui è una parola, il cosiddetto **byte** che è un raggruppamento di otto bit.

Osserviamo la prima griglia nella figura 6. È costruita con otto righe di otto quadrati ciascuna. Le caselle sono vuote, ogni bit è uguale a 0.
Nella seconda griglia il primo quadrato a sini-

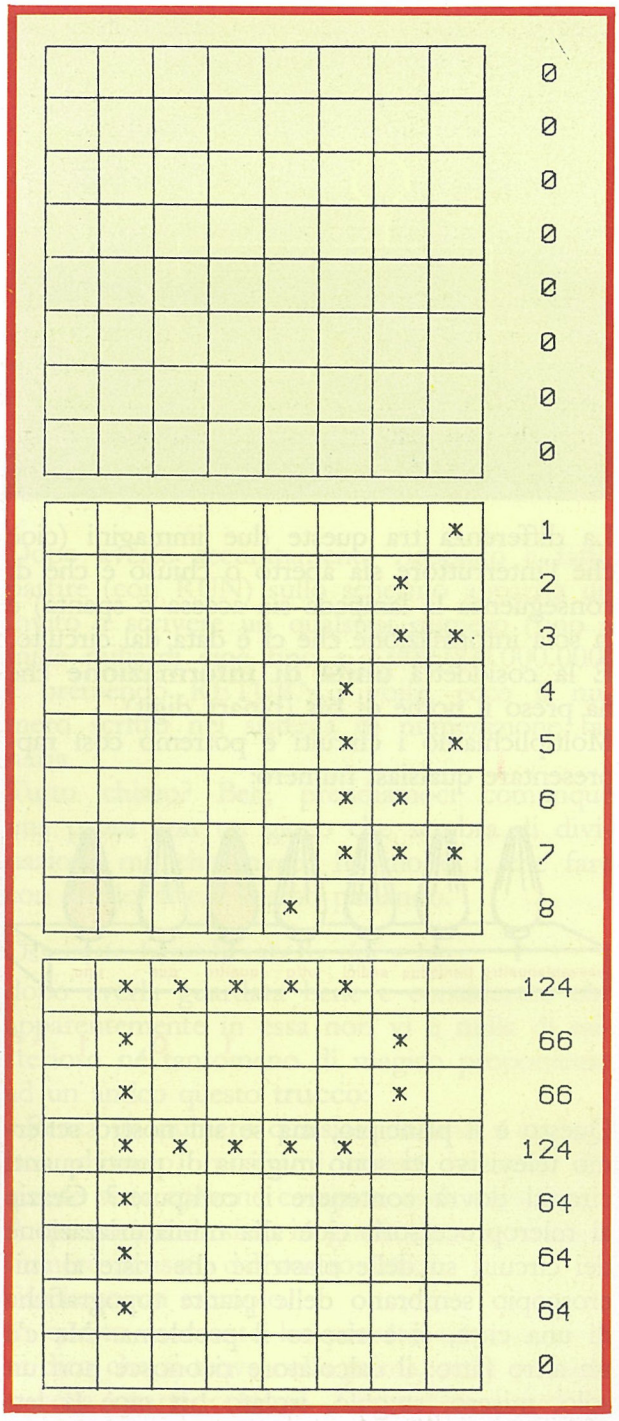
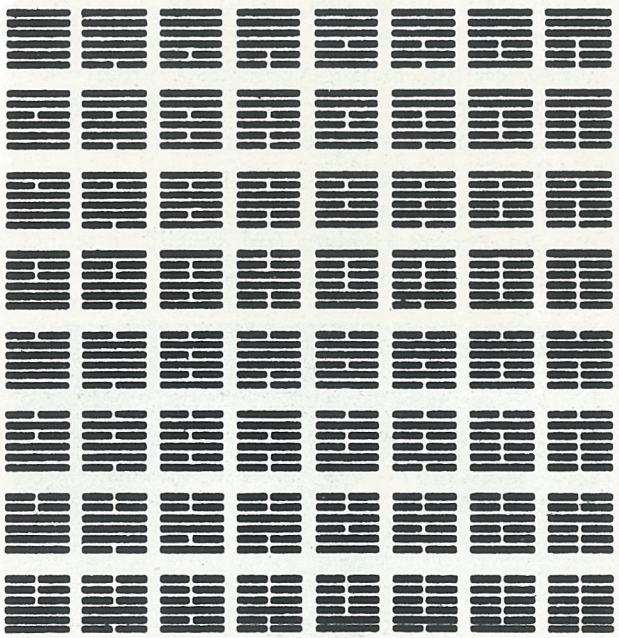


Fig. 6

stra è pieno (lo rappresentiamo con un asterisco) e quindi il byte ha valore (decimale) uguale a uno; la seconda riga è un byte con valore 2 e così via.
Nella terza griglia i bit pieni formano la lettera P ed a fianco potete leggere i valori degli otto byte che costruiscono il carattere.

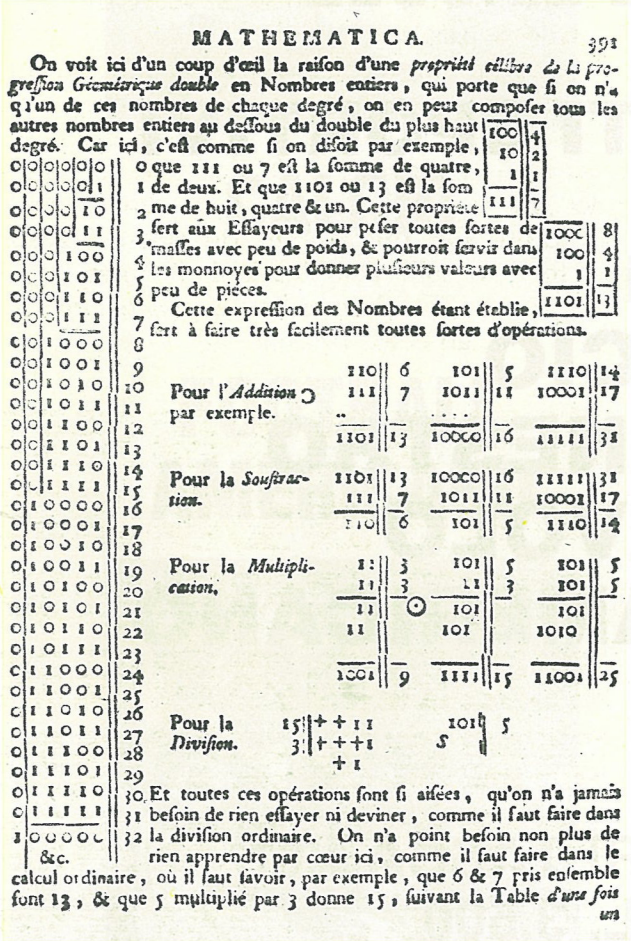
Ma in questa rubrica non vogliamo approfondire l'aspetto tecnico o operativo dei computers, facciamo quindi una rapida virata e cerchiamo di rispondere a questa domanda: quando è nato il **bit**, o meglio, quando è nato il sistema di numerazione binaria? Con l'avvento dei calcolatori più o meno meccanici o elettrici o elettronici?
No, molto molto prima, tanto è vero che esistono ancora tribù primitive in Australia e in Africa che usano questo sistema di numerazione. Questi selvaggi non hanno ancora raggiunto lo stadio del contare sulle dita, hanno vocaboli indipendenti per indicare **uno** e **due** e vocaboli composti fino a sei; oltre il sei usano la parola « mucchio »...
Ma anche altre civiltà antiche, molto più progredite, si sono servite dei simboli binari. Basti pensare allo **yin** e **yang** dei cinesi. Lo **yang** è rappresentato da una linea continua, lo **yin** da una linea spezzata al centro. Tutto il **Libro dei mutamenti** (in cinese **I King**) è fondato sulle combinazioni di questi segni, in gruppi di 6 (i cosiddetti *esagrammi*) che venivano usati come oracoli:



i 64 esagrammi del libro dei mutamenti

Le stesse componenti magico/mistiche dello zero e dell'uno furono poi riprese nel '600 da Gottfried Leibniz che « ...vide nella sua arit-

metica binaria l'immagine della creazione. Egli immaginò che l'unità rapprentasse Dio e lo zero il nulla: che l'Ente Supremo avesse tratto tutti gli esseri dal nulla proprio come l'unità e lo zero esprimono tutti i numeri nel suo sistema di numerazione ».
È famosa la sua affermazione: « Omnibus ex nihil ducendis sufficit unum » (uno basta per trarre tutto dal nulla).
Ma Leibniz fu anche e soprattutto uno scienziato e mise in evidenza come fosse semplice eseguire operazioni aritmetiche con i numeri binari.



MATHEMATICA. 391

On voit ici d'un coup d'œil la raiſon d'une *propriété célèbre de la progreſſion Géométrique double* en Nombres entiers, qui porte que ſi on n'a qu'un de ces nombres de chaque degré, on en peut compoſer tous les autres nombres entiers au deſſous du double du plus haut degré. Car ici, c'eſt comme ſi on diſoit par exemple, que 111 ou 7 eſt la ſomme de quatre, de deux. Et que 1101 ou 13 eſt la ſomme de huit, quatre & un. Cette propriété ſert aux Eſſayeurs pour peſer toutes ſortes de maſſes avec peu de poids, & pourroit ſervir dans les monnoyes pour donner pluſieurs valeurs avec peu de pièces.

Cette expreſſion des Nombres étant établie, ſert à faire très facilement toutes ſortes d'opérations.

Pour l'*Addition* par exemple.

Pour la *Souſtraction.*

Pour la *Multiplication,*

Pour la *Diviſion.*

Et toutes ces opérations ſont ſi aiſées, qu'on n'a jamais beſoin de rien eſſayer ni deviner, comme il faut faire dans la diviſion ordinaire. On n'a point beſoin non plus de rien apprendre par cœur ici, comme il faut faire dans le calcul ordinaire, où il faut ſavoir, par exemple, que 6 & 7 pris enſemble ſont 13, & que 5 multiplié par 3 donne 15, ſuivant la Table *d'une fois un* &c.

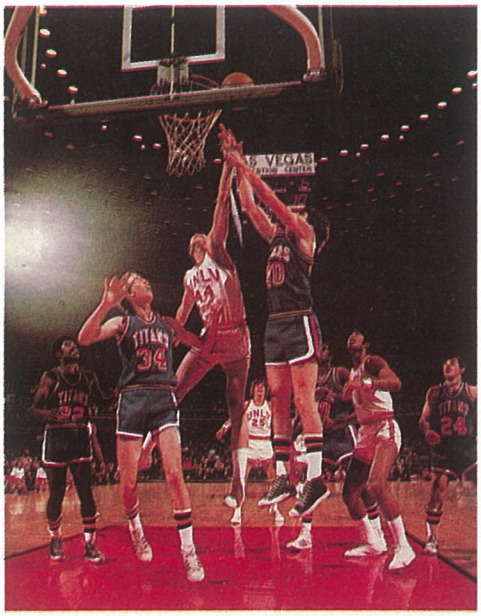
Bisogna tuttavia aspettare la metà dell'ottocento perchè la numerazione binaria assuma maggior importanza, con due ricercatori che operavano in campi allora considerati totalmente disgiunti:
Samuel Morse inventa il telegrafo che usa un alfabeto binario (il famoso punto e linea delle prime emittenti via etere);
Georges Boole è invece un matematico che allarga i confini della logica aristotelica proponendo e formalizzando un sistema che opera algebricamente sulle preposizioni servendosi di tavole in cui compaiano soltanto i numeri 0 e 1.
L'avvento dei primi calcolatori non fece altro che fondere questi due diversi usi della notazione binaria. Da una parte la comodità di produzione e trasmissione di una serie di segnali base on/off (si/no). Dall'altra il supporto logico matematico che permettesse con facilità qualsiasi tipo di operazione logica o proposizionale.

Ed eccoci ai giorni nostri. Il numero binario che era partito dalla semplicità di culture rozze e che poi divenne quasi un monumento al culto del monoteismo è ora giunto al suo massimo apice di consumo... il suo ciclo si concluderà quando si insegnerà ad usarlo nelle scuole elementari. Sarà solo un utensile, come il suo cugino decimale ed il suo fratello esadecimale (la numerazione in base sedici altrettanto utile agli esperti in linguaggio macchina...). Dobbiamo però sempre tener presente che non è null'altro che uno strumento numerico, utile per un altro strumento quale è il computer.
Nonostante si parli di base decimale o binaria, la vera base sta nell'uso che ne facciamo. E per sottolineare questa mia opinione prendo a prestito una frase di Giovanni Caramuel, vescovo di Vigevano negli ultimi anni del '600 e precursore di Leibniz nello studio dei numeri binari:
« Se io, infatti, avessi nella mia biblioteca quattro pendole, e se tutte suonassero contemporaneamente l'una, non dirò che hanno suonato le quattro, ma che hanno suonato quattro volte l'una. Questa differenza non è inserita sulle cose, non è indipendente dalle operazioni della mente: dipende – anzi – dalla mente di colui che conta ».

Bye bye a tutti, anzi byte byte e arrivederci al prossimo mese.

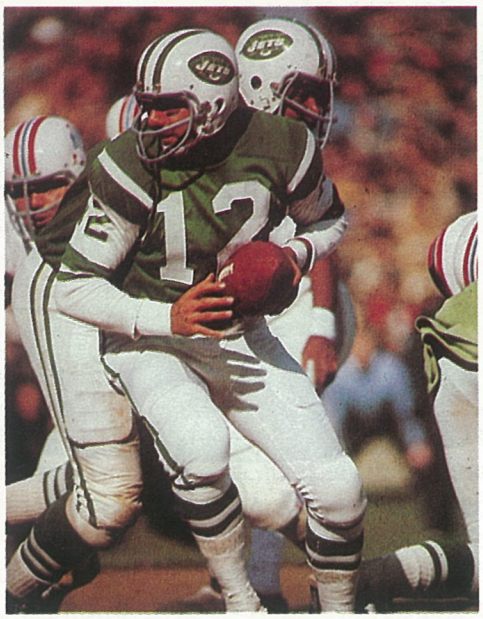
Aldo Spinelli

CALCIO

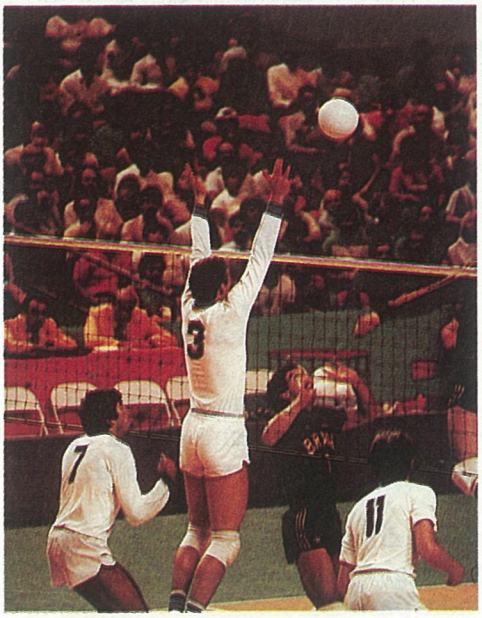


PALLACANESTRO

FOOTBALL AMERICANO



PALLAVOLO



i nostri magnifici supergiochi

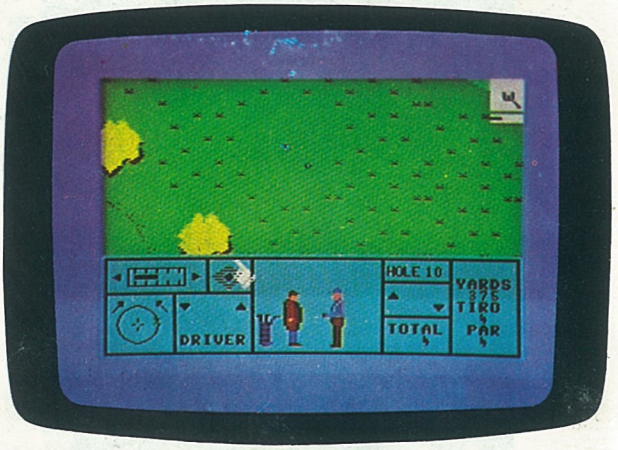
STREGONA

CBM 64 - Joystick in porta-2
Parola chiave: SCOPA

Stupendo gioco con molti schermi e livelli di difficoltà che aumentano con il cambio degli schermi (dopo il superamento della fase precedente).
Dovete guidare la vostra strega in un lunghissimo viaggio costellato di difficoltà.
Per non perdere energia necessaria al viaggio dovrete evitare i fulmini spregionati da tante nubi minacciose ed abbattere le streghe nemiche.
Buon divertimento.

MAXI GOLF

CBM 64 - Joystick in porta-2
Parola chiave: BUCA



Se siete appassionati di golf, questo è il gioco che fa per voi. Potrete usufruire di un campo

da golf con le 18 buche regolamentari ed una rappresentazione del gioco complessivo, molto realistica.
Il gioco vi consente di optare fra diversi tipi di partecipazione: per effettuare le vostre scelte guidate con il joystick la mano bianca che compare sullo schermo.
Prima di effettuare ogni tiro dovrete scegliere il tipo di ferro adeguato, direzione del tiro e potenza.
Dopo ogni tiro, la mano bianca si posiziona in modo tale che premendo il pulsante del joystick vi consentirà di avere una visione globale del campo, rilasciando il pulsante tornerete alla visione settoriale della vostra posizione.

LANDER

CBM 64 - Joystick in porta-2
Parola chiave: NAVICELLA

Lander è un gioco d'abilità, dovrete dimostrare il vostro coraggio e destrezza alla guida di una navicella con la quale dovrete recuperare molti soldati prigionieri del nemico. Questo nemico possiede una città sotterranea molto difesa, alla quale potrete accedere distruggendo il cratere di un vulcano.
Oltre al recupero dei soldati, dovete distruggere la centrale nucleare che si trova nel cuore dei sotterranei.
Durante il viaggio ricordatevi di atterrare nei vari punti predisposti per il rifornimento carburante altrimenti non riuscirete a finire la missione.

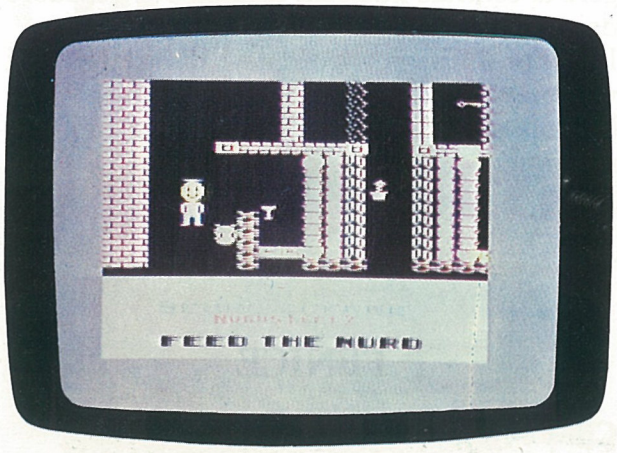
FENICE

CBM 64 - Joystick in porta-2
Parola chiave: DIFESA

Siete un valoroso cavaliere alla difesa del vostro castello attaccato dalla fenice.
Avete a disposizione una poderosa arma per abbattere i nemici.
Nel gioco finale, un triplice arco si formerà fra le due montagne antistanti il vostro castello.

FLIGHTS OF MAN

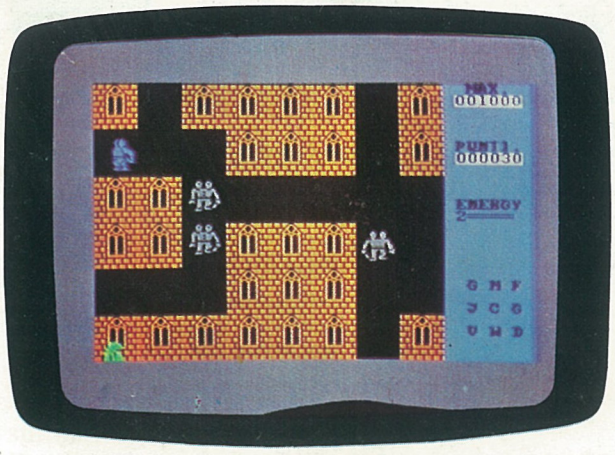
CBM 64 - Joystick in porta-2
Parola chiave: ELICA



Con il vostro OMINO volante dovrete raccogliere una serie di oggetti disseminati nelle varie stanze.
Naturalmente, gli ostacoli che troverete lungo il vostro cammino saranno tanti: starà a voi evitarli nel migliore dei modi.
Per far sollevare da terra l'omino volante basta premere il pulsante, e seguire le indicazioni dello schermo per scegliere le opzioni.

THE LANDS

CBM 64 - Joystick in porta-1
Parola chiave: LABIRINTO



Dimostrate il vostro senso d'orientamento percorrendo uno sterminato labirinto alla ricerca di tesori nascosti.
Il gioco prende 160 schermi diversi, quindi non sarà troppo facile finire il gioco, soprattutto se non eviterete i vari mostri che popolano il labirinto.
Buon divertimento, ed attenzione alle mille soprese... nascoste.

MAGICO BILL

CBM 64 - Joystick in porta-2
Parola chiave: FUNGO

Bill, servendosi di un simpatico gnomo, ama raccogliere funghi. Però molte difficoltà lo insidiano, come piante carnivore, uccelli predatori.
Come se tutto questo non bastasse, un vulcano con la sua colata lavica sta seminando distruzione.
Per controllare le condizioni del vulcano potete in qualsiasi momento del gioco premere la barra spaziatrice per cambiare schermo, poi ritornare al gioco premere nuovamente la barra.
Buon divertimento.

JET MAN

C 16 e PLUS 4 - Joystick o tastiera
Parola chiave: VENERE

Tasti:
SPAZIO - sposta in basso le opzioni
SHIFT - sposta in alto le opzioni
RETURN - cambia le opzioni

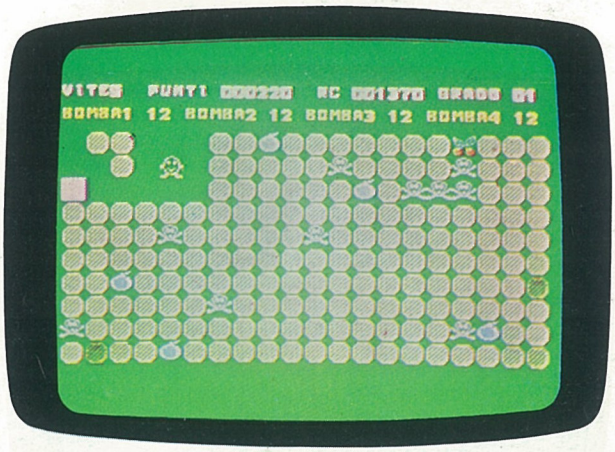
PER GIOCARE
Z - sinistra
X - destra
RETURN - fuoco
SHIFT - alto
ESC - incomincia una nuova partita

Le tue piantagioni di Xenonite sul pianeta Venere sono minacciate da uno storno di insetti provenienti da un asteroide di passaggio. Gli insetti, molto affamati, cercheranno di mangiare tutte le piante prima della fioritura. Tu devi difenderle sterminando gli insetti. Quando la barra che indica la fioritura sarà a zero, passerai in un nuovo quadro dove le difficoltà saranno maggiori.
Quando la barra dell'energia segnerà zero, rifugiati dietro un fiore per rifornire i tuoi razzi con del polline astrale.
Se ti scontri con un insetto perdi dell'energia.
Per memorizzare i record con Z e X sposti il cursore e con SPAZIO e SHIFT cambi le lettere, con RETURN memorizzi il record.

BOMBE

C 16 e PLUS 4 - Joystick o tastiera
Parola chiave: ATOMIC

Tasti:
Z - sinistra
X - destra
/ - basso
; - alto

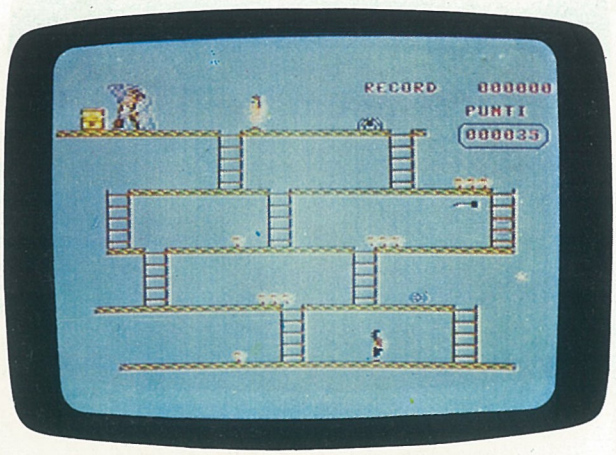


Bombe è un gioco di abilità e destrezza: il tuo compito è neutralizzare quattro bombe disposte a caso sullo schermo. All'inizio del gioco sei in alto a sinistra; con i tasti o con il joystick spostato sopra le bombe per disinnescarle. Il "secondo passo" da fare è cercare l'entrata che ti permetterà di passare ad un secondo quadro. L'entrata la riconoscerai dalle righe colorate: fai attenzione, però, perché devi trovare quella giusta.
Per spostarti; questa è una regola molto importante, cerca di essere sempre a contatto (in una delle quattro direzioni),
con un pallino; se resterai isolato... non ti resterà che aspettare lo scoppio delle bombe.
Per iniziare il gioco, premi Z se usi la tastiera, e quindi Spazio.

IL GORILLA

C 16 e PLUS 4 - Joystick o tastiera
Parola chiave: BARILI

Tasti:
Q - sale le scale
Z - scende le scale
I - sinistra
P - destra
SPAZIO - salto



Questo è il classico gioco del gorilla che lancia contro di te pesantissimi barili. Il tuo compito è evitarli e raggiungere la cima del grattacielo per passare ad un quadro più difficile.

Per ottenere dei punti in più puoi prendere il martello e schiacciare i barili, attento però, perché il martello dopo un periodo di tempo scompare.

Cerca di arrivare in cima molto alla svelta, prima che il ragno posto alla destra del gorilla arrivi prima di te (rischi la morte immediata). Nel secondo schermo il gorilla non ti tirerà più barili ma dei velenosissimi ragni. E nel terzo schermo?
Scropritelo voi!

LABIRINTO

C 16 e PLUS 4 - Tastiera
Parola chiave: MURI

Tasti:
J - sinistra
L - destra
I - alto
K - basso

Hai sbagliato ad usare la tua macchina del tempo e questa ti ha trasportato nell'antica Grecia. Precisamente nel castello dove si perse Dedalo. Il computer ti genererà un labirinto (potrai sceglierne le dimensioni) e tu dovrai uscirne nel minor tempo possibile.
A seconda della grandezza che scegli hai a disposizione un determinato periodo di tempo. Se finisci il tempo muori.

BUON DIVERTIMENTO

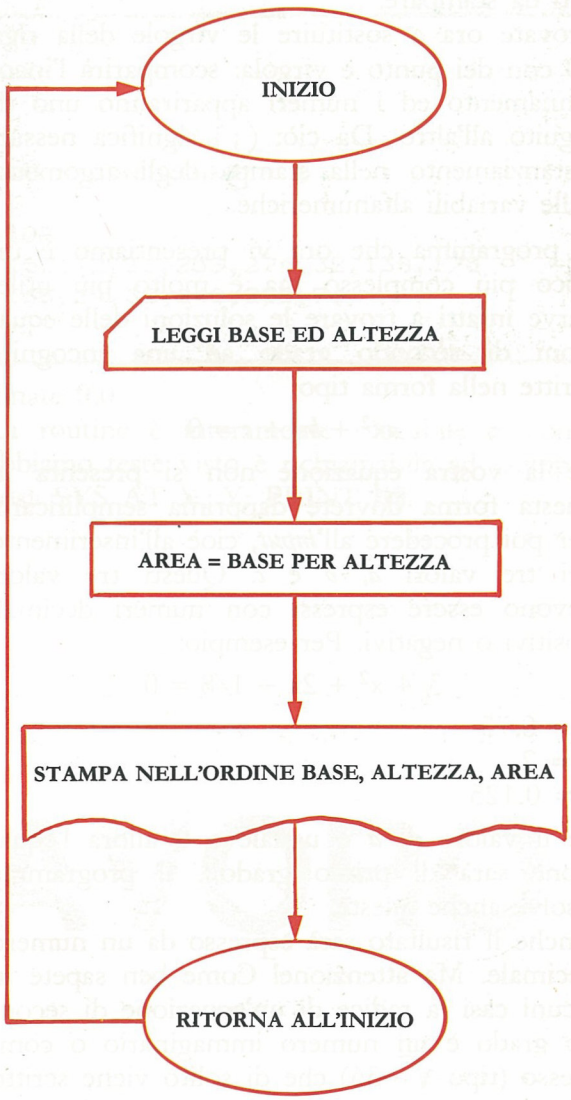
impariamo a usare il computer 2

I PRIMI RUDIMENTI DEL LINGUAGGIO BASIC

La puntata precedente abbiamo visto come sono strutturate le STRINGHE: ovvero quelle piccole sequenze di caratteri alfanumerici tramite le quali riusciamo ad esprimere all'interno di un programma, l'argomento di alcuni comandi essenziali per il lavoro con l'elaboratore.
In genere è indispensabile definire con rigorosa precisione come, dove e quando il computer dovrà eseguire certuni comandi da noi impartiti.
Prendiamo ad esempio il caso di PRINT ovvero STAMPA: ciò significa che il computer dovrà riprodurre in una zona dello schermo o del foglio di carta il contenuto di una stringa alfanumerica prestabilita.
Ci troviamo quindi nella necessità di stabilire uno o più metodi di identificazione della posizione di stampa, ricorrendo alle risorse del linguaggio BASIC che il nostro computer è in grado di comprendere.
Per inciso vale la pena di ricordare che:
5 print «a» realizza la stampa della lettera a;
mentre 5 print A realizza la stampa del contenuto (o valore) della variabile di nome A.

La prima soluzione è di affidarci a ciò che il computer sa eseguire automaticamente: l'incolonnamento ad esempio.
Un semplice programma che esegue una piccola serie di moltiplicazioni e ne scrive il risultato è adatto allo scopo illustrativo:

PROGRAMMA AREA DEL QUADRATO



LISTATO DEL PROGRAMMA AREA SCRITTO IN BASIC

```
10 REM PROGRAMMA AREA
20 READ B, A
30 LET S = B*A
40 PRINT B, A, S
50 GOTO20
60 DATA 15,7,34,78,1.45,5.67,100,362
100 END
```

Eseguite il programma e potrete vedere che i dati elaborati, con i rispettivi risultati appariranno incolonnati in bell'ordine.
Nulla di miracoloso, solamente avete utilizzato correttamente l'istruzione (,) che segnala al computer di incolonnare sequenzialmente i dati da stampare.
Provate ora a sostituire le virgole della riga 40 con dei punto e virgola: scomparirà l'incolonnamento ed i numeri appariranno uno di seguito all'altro. Da ciò: (;) significa nessun distanziamento nella stampa degli argomenti delle variabili alfanumeriche.

Il programma che ora vi presentiamo è un poco più complesso ma è molto più utile. Serve infatti a trovare le soluzioni delle equazioni di secondo grado ad una incognita scritte nella forma tipo:

$$ax^2 + bx + c = 0$$

Se la vostra equazione non si presenta in questa forma dovrete dapprima semplificarla per poi procedere all'*input*, cioè all'inserimento dei tre valori *a*, *b* e *c*. Questi tre valori devono essere espressi con numeri decimali positivi o negativi. Per esempio:

$$3/4\ x^2 + 2x - 1/8 = 0$$

$a = 0.75$
$b = 2$
$c = 0.125$

Se il valore di *a* è uguale a 0 allora l'equazione sarà di primo grado... il programma risolve anche queste.
Anche il risultato sarà espresso da un numero decimale. Ma attenzione! Come ben sapete in alcuni casi la radice di un'equazione di secondo grado è un numero immaginario o complesso (tipo $\sqrt{-36}$) che di solito viene scritto seguito da una i. Con questo programma i numeri immaginari (o la parte immaginaria di un numero complesso) viene visualizzata con il numero scritto in NERO. Invece ogni radice reale sarà scritta in BIANCO.
Ecco il listato:

```
10 REM    RISOLUZIONE DELLE EQUAZIONI
11 :
12 REM    DI (PRIMO E) SECONDO GRADO
13 :
14 :
20 REM       PER VIC 20 O CBM 64
21 :
22 :
30 POKE53280,11:POKE53281,11
31 :
32 REM    PER IL VIC 20 SOSTITUIRE LA
33 REM    RIGA 30 CON:
34 REM    POKE36879,110
35 :
40 PRINTCHR$(147)CHR$(158)CHR$(17)
50 PRINT"EQUAZIONE TIPO:"
60 PRINTCHR$(17)SPC(2)"2"
70 PRINTCHR$(14)"AX + BX + C = 0"
80 PRINTCHR$(17)CHR$(5)
90 INPUT"A= ";A
91 INPUT"B= ";B
92 INPUT"C= ";C
100 IFA=0ANDB=0ANDC=0THEN1000
110 IFA=0ANDB<>0ANDC=0THEN1100
120 IFA=0ANDB<>0ANDC<>0THEN1200
130 IFA<>0ANDB=0ANDC=0THEN1300
140 IFA<>0ANDB=0ANDC<>0THEN1400
150 IFA<>0ANDB<>0ANDC=0THEN1500
160 IFA<>0ANDB<>0ANDC<>0THEN1600
1000 PRINTCHR$(17)"X= "A:GOSUB2000
1100 PRINTCHR$(17)"X= "1/B:GOSUB2000
1200 PRINTCHR$(17)"X= "(-C/B):GOSUB2000
1300 PRINTCHR$(17)"X1=X2= "B:GOSUB2000
1400 IFC>0ANDA>0THENC=-C:PRINTCHR$(144)
1405 IFC<0ANDA<0THENC=-C:PRINTCHR$(144)
1410 J=SQR(-C/A):K=(-(SQR(-C/A)))
1420 PRINTCHR$(17)"X1= "J
1430 PRINTCHR$(17)"X2= "K:PRINTCHR$(5):G
OSUB2000
1500 PRINTCHR$(17)"X1= "0
1510 PRINTCHR$(17)"X2= "(-B/A):GOSUB2000
1600 IF(B↑2-4*A*C)<0THENGOSUB1700
1605 IF(B↑2-4*A*C)<0THENGOSUB1700
1610 D=SQR(B↑2-4*A*C)
1620 PRINTCHR$(17)"X1= "((-B+D)/(2*A))
1630 PRINTCHR$(17)"X2= "((-B-D)/(2*A)):G
OSUB2000
1700 D=SQR(ABS(B↑2-4*A*C))
1710 PRINTCHR$(17)"X1= "(-B/(2*A))"+"CHR
$(144)(D/(2*A))CHR$(5)
1720 PRINTCHR$(17)"X2= "(-B/(2*A))"-"CHR
$(144)(D/(2*A)):GOSUB2000
2000 PRINTCHR$(17)CHR$(17)CHR$(158)"UN'A
LTRA EQUAZIONE (S/N) ?"
2010 GETA$:IFA$=""THEN2010
2020 IFA$<>"S"THENEND
2030 GOTO40

READY.
```

Per il VIC 20 la riga 30 va sostituita con la poke della riga 34. Il programma prende in considerazione i vari tipi di equazioni e, per ciascuna, rimanda a delle piccole subroutines per la soluzione. Questo perchè possiate facilmente intervenire con le modifiche che vi interessano (ad esempio la stampa del discriminante).
Buon lavoro e prima di chiudere questa parentesi «matematica» vi anticipiamo che il prossimo mese riparleremo di equazioni... di grado superiore al secondo.

Ora riprendiamo il nostro discorso.

Il linguaggio BASIC fornisce al programmatore differenti possibilità di scelta sul modo di stampare numeri e parole: oltre alla virgola ed al punto e virgola troviamo PRINT TAB(X) ove il valore di X indica la colonna sulla quale si disporranno verticalmente i dati; PRINT USING molto utile per saldare tra di loro alcune stringhe alfanumeriche ed infine PRINT AT per la definizione di un punto dello schermo nel quale iniziare la stampa.
L'utilizzo del cursore di controllo entro l'istruzione di stampa, nel caso del Commodore 64, permette un uso flessibile dello schermo e della stampante ma inevitabilmente ciò va a complicare troppo la gestione della istruzione di stampa; da ciò si rende necessaria l'impostazione di un altro comando che chiameremo appunto PRINT AT.
Non essendo presente tale comando all'interno del linguaggio BASIC del Commodore 64, verrà scritta una routine che ne fornisce la facoltà d'uso.

```
10 rem routine print at
20 ADDR = 10000
30 CONTATORE = 0
40 FOR T = ADDR TO ADR +28
50 : READ D: POKE T,D: CONTATORE = CONTATORE + D
60 NEXT T
70 IF CONTATORE <> 3857 THEN STOP
80 DATA 32,253,174,32,247,183,152,72,32,253,274,32,138,173
90 DATA 32,247,183,166,20,104,168,24,32,240,255,96
100 END
```

Questa breve routine è richiamabile da un comando SYS seguito dalle coordinate X (con valori compresi tra 0 e 39) e dalle coordinate Y (con valori compresi tra 0 e 24); il punto più alto a sinistra dello schermo ha coordinate 0,0.
La routine è interamente rilocabile e, come abbiamo testè visto è richiamabile ad esempio così: SYS AT X, Y: PRINT B$.

* * *

BUON LAVORO

impariamo a programmare in assembler 1

commodore 64 - commodore 64 - commodore 64 - commodore 64 - commodore

Con questo numero inizia una serie di articoli dedicati alla comprensione e all'uso pratico del linguaggio macchina (abbreviato L/M) del COMMODORE 64.

Le caratteristiche di tutti gli articoli saranno: l'assoluta mancanza di riferimenti al BASIC (salvo per il comando SYS) e una chiara descrizione di quanto avviene all'interno del vostro home-computer quando si fa girare un programma.

L'obbiettivo da raggiungere è mettere qualsiasi possessore di CBM 64 in condizione di progettare, realizzare, correggere e far funzionare un proprio programma in L/M.

Nello stilare gli articoli si è tenuto conto dei seguenti presupposti.

1) Non è richiesta la conoscenza del BASIC
2) Non è necessario conoscere l'elettronica digitale
3) Non è indispensabile la pur minima esperienza di programmazione.

Prima di iniziare a parlare di L/M e affini, vorrei fare un ultimo doveroso accenno al Basic.

Il BASIC è un linguaggio "evoluto", cioè la sua struttura tende ad avvicinarsi al modo di comunicare umano e trascura nella maniera più assoluta la "lingua" parlata dalla macchina su cui gira.

La sua diffusione è stata enormemente facilitata dal fenomeno home e personal computer che adottano come "lingua madre" il BASIC.

La sua caratteristica principale è sicuramente la relativa facilità di programmazione e di debugging (collaudo), mentre il limite più evidente è la lentezza di esecuzione che talvolta rende il suo uso inaccettabile e quindi impossibile. Il linguaggio macchina, rispetto al Basic, è la "lingua" esclusiva di un ben determinato computer, anzi, di un ben determinato microprocessore quindi pensato e realizzato in funzione della macchina non tenendo conto del modo di comunicare umano.

L'aspetto più importante del L/M è senza dubbio l'enorme possibilità di controllo accurato di ogni funzione del computer e naturalmente la velocità di esecuzione dei programmi.

A svantaggio del L/M c'è senza dubbio il fatto di essere costituito solo da numeri e quindi incomprensibile a prima vista ma per ovviare parzialmente questo fatto, adotteremo un altro linguaggio, l'ASSEMBLER, che sostituisce ai numeri un codice fatto di parole abbreviate detto "codice mnemonico". Ultima considerazione sulla differenza tra BASIC e ASSEMBLER; la stragrande maggioranza dei comandi BASIC è, all'interno della macchina, in realtà una serie di comandi in L/M che viene eseguita dall'interprete come una routine; in ASSEMBLER ogni istruzione o comando corrisponde a un comando in L/M pertanto si presenta la necessità di costruire ogni cosa passo dopo passo.

Gli esempi e i listati contenuti in questa serie di articoli sono stati realizzati con il MACRO ASSEMBLER della COMMODORE ma qualsiasi altro ASSEMBLER per il CBM 64 andrà egregiamente.

IL MICROPROCESSORE E IL CODICE BINARIO

Il microprocessore è il risultato della naturale evoluzione dei circuiti chiamati logici ed è il componente fondamentale di ogni micro o home computer.
Si tratta di un circuito integrato contenente un insieme estremamente complesso di circuiti logici, formato da migliaia di transistor, che riunisce in sé quattro caratteristiche fondamentali: ridotte dimensioni, basso costo, elevate prestazioni e grande versatilità.
Come tutti i circuiti logici, anche il microprocessore o CPU (Central Processing Unity) è in grado di capire e quindi di gestire soltanto "stati elettrici", vale a dire presenza (1) o assenza (Ø) di corrente o di tensione.
Da questa caratteristica è nata l'esigenza di rappresentare i numeri (anche le lettere dell'alfabeto o qualsiasi altra cosa sono riducibili a numeri) in un codice che avesse come "base" 2, cioè che usasse soltanto zeri e uno nelle loro possibili combinazioni, vale a dire il CODICE BINARIO.
Il codice normalmente usato per la rappresentazione numerica è notoriamente il sistema DECIMALE, cioè a "base" 10 che vuol significare che con 10 simboli (0,1,2,3,4,5,6,7,8,9) è possibile, sfruttando tutte le possibili combinazioni, rappresentare un numero qualsiasi ed effettuare operazioni. Ora supponiamo di chiedere al nostro COMMODORE 64 il risultato di una somma di numeri decimali, la sua risposta sarà un numero decimale. Se il sistema di lavoro del microprocessore è binario, come mai comunica con noi in decimale? La risposta è palese, noi chiedendo il risultato della somma non abbiamo utilizzato direttamente il microprocessore ma abbiamo sfruttato le caratteristiche di un "programma" che ha tradotto prima da decimale in binario i nostri numeri da sommare, quindi ha dato tutte le direttive necessarie perché avvenisse la somma e alla fine ha trasformato da binario a decimale il risultato stampandolo sul nostro video.

A questo punto è evidente la differenza tra utilizzatore e programmatore; l'utilizzatore deve usare la sua macchina nel modo più semplice possibile e il programmatore deve scrivere programmi che tengano conto delle esigenze di chi li utilizzerà quindi, dal punto di vista del computer le rappresentazioni possibili saranno almeno 2: una interna, usata dal programmatore per farsi capire dalla macchina, e una esterna, usata dalla macchina per farsi capire dall'utilizzatore.

LA PROGRAMMAZIONE

Scrivere un programma significa innanzi tutto escogitare un metodo per risolvere un problema dato.
Questa soluzione è espressa come una procedura di fasi successive chiamata "ALGORITMO", cioè una specificazione fase-per-fase della soluzione da dare al problema in un numero di passi definito.
L'algoritmo può essere espresso in qualsiasi linguaggio.
Un esempio di algoritmo molto comunemente usato è il seguente:

1) Premere il tasto "EJECT"
2) Inserire nell'apposito portacassette la cassetta con il nastro rivolto verso di voi
3) Chiudere il portacassette
4) Premere il tasto "PLAY"

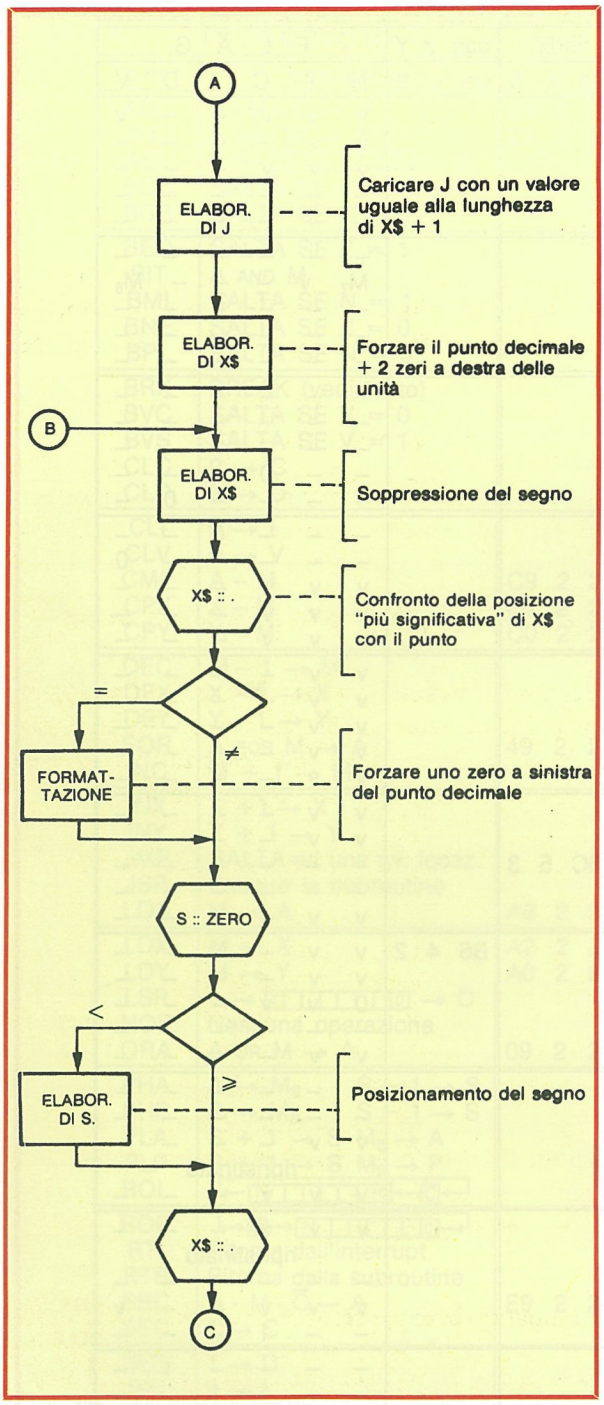
a questo punto se l'algoritmo è giusto per il tipo Questo è un algoritmo a 4 fasi per mettere in trasmissione un nastro magnetico. Se noi ora dovessimo comunicare il nostro algoritmo al computer succederebbe che come minimo non verrebbe capito, da qui l'esigenza di un linguaggio di programmazione.

La PROGRAMMAZIONE è la conversione di un algoritmo in una sequenza di istruzioni in linguaggio di programmazione, cioè la "CODIFICA" dell'algoritmo + il progetto globale dei programmi e le "strutture dati" che realizzano l'algoritmo stesso.
Per programmare in maniera effettiva si richiede oltre che la comprensione delle tecniche di realizzazione possibili per gli algoritmi convenzionali, anche una capacità di impiego di tutte le risorse hardware (diciamo meccaniche) del calcolatore come la memoria, i registri interni ed i dispositivi periferici ed infine la padronanza dell'impiego costruttivo di opportune strutture di dati.

ISTRUZIONI		immed.			ass.			pag. z.			accum.			impl.			(ind. X)		
c.m.	operazione	co	c	b	co	c	b	co	c	b	co	c	b	co	c	b	co	c	b
ADC	A+M+C → A	69	2	2	6D	4	3	65	3	2							61	6	2
AND	A AND M → A	29	2	2	2D	4	3	25	3	2							21	6	2
ASL	C ← [7][6][5][4][3][2][1][0] ← 0				0E	6	3	06	5	2	0A	2	1						
BCC	SALTA SE C = 0																		
BCS	SALTA SE C = 1																		
BEQ	SALTA SE Z = 1																		
BIT	A AND M				2C	4	3	24	3	2									
BMI	SALTA SE N = 1																		
BNE	SALTA SE Z = 0																		
BPL	SALTA SE N = 0																		
BRK	BREAK (vedi testo)													00	7	1			
BVC	SALTA SE V = 0																		
BVS	SALTA SE V = 1																		
CLC	0 → C													18	2	1			
CLD	0 → D													D8	2	1			
CLI	0 → I													58	2	1			
CLV	0 → V													B8	2	1			
CMP	A – M	C9	2	2	CD	4	3	C5	3	2							C1	6	2
CPX	X – M	E0	2	2	EC	4	3	E4	3	2									
CPY	Y – M	C0	2	2	CC	4	3	C4	3	2									
DEC	M – 1 → M				CE	6	3	C6	5	2									
DEX	X – 1 → X													CA	2	1			
DEY	Y – 1 → Y													88	2	1			
EOR	A EOR M → A	49	2	2	4D	4	3	45	3	2							41	6	2
INC	M + 1 → M				EE	6	3	E6	5	2									
INX	X +1 → X													E8	2	1			
INY	Y + 1 → Y													C8	2	1			
JMP	SALTA ad una nv. locaz.				4C	3	3												
JSR	Esegue la subroutine				20	6	3												
LDA	M → A	A9	2	2	AD	4	3	A5	3	2							A1	6	2
LDX	M → X	A2	2	2	AE	4	3	A6	3	2									
LDY	M → Y	A0	2	2	AC	4	3	A4	3	2									
LSR	0 → [7][6][5][4][3][2][1][0] → C				4E	6	3	46	5	2	4A	2	1						
NOP	Nessuna operazione													EA	2	1			
ORA	A OR M → A	09	2	2	0D	4	3	05	3	2							01	6	2
PHA	A → M_S S – 1 → S													48	3	1			
PHP	P → M_S S – 1 → S													08	3	1			
PLA	S + 1 → S M_S → A													68	4	1			
PLP	S + 1 → S M_S → P													28	4	1			
ROL	←[7][6][5][4][3][2][1][0]←[C]←				2E	6	3	26	5	2	2A	2	1						
ROR	→[C]→[7][6][5][4][3][2][1][0]→				6E	6	3	66	5	2	6A	2	1						
RTI	Ritorna dall'interrupt													40	6	1			
RTS	Ritorna dalla subroutine													60	6	1			
SBC	A - M - $\overline{C}$ → A	E9	2	2	ED	4	3	E5	3	2							E1	6	2
SEC	1 → C													38	2	1			
SED	1 → D													F8	2	1			
SEI	1 → I													78	2	1			
STA	A → M				8D	4	3	85	3	2							81	6	2
STX	X → M				8E	4	3	86	3	2									
STY	Y → M				8C	4	3	84	3	2									
TAX	A → X													AA	2	1			
TAY	A → Y													A8	2	1			
TSX	S → X													BA	2	1			
TXA	X → A													8A	2	1			
TXS	X → S													9A	2	1			
TYA	Y → A													98	2	1			

Set istruzioni

(ind.), Y	pag. z, X	ass, X	ass, Y	rel.	indir.	pag. z, Y	F L A G					
co c b	co c b	co c b	co c b	co c b	co c b	co c b	N	Z	C	I	D	V
71 5 2	75 4 2	7D 4 3	79 4 3				v	v	v	–	–	v
31 5 2	35 4 2	3D 4 3	39 4 3				v	v	–	–	–	–
	16 6 2	1E 7 3					v	v	v	–	–	–
				90 2 2			–	–	–	–	–	–
				B0 2 2			–	–	–	–	–	–
				F0 2 2			–	–	–	–	–	–
							M_7	v	–	–	–	M_6
				30 2 2			–	–	–	–	–	–
				D0 2 2			–	–	–	–	–	–
				10 2 2			–	–	–	–	–	–
							–	–	–	–	–	–
				50 2 2			–	–	–	–	–	–
				70 2 2			–	–	–	–	–	–
							–	–	0	–	–	–
							–	–	–	–	0	–
							–	–	–	0	–	–
							–	–	–	–	–	0
D1 5 2	D5 4 2	DD 4 3	D9 4 3				v	v	v	–	–	–
							v	v	v	–	–	–
							v	v	v	–	–	–
	D6 6 2	DE 7 3					v	v	–	–	–	–
							v	v	–	–	–	–
							v	v	–	–	–	–
51 5 2	55 4 2	5D 4 3	59 4 3				v	v	–	–	–	–
	F6 6 2	FE 7 3					v	v	–	–	–	–
							v	v	–	–	–	–
							v	v	–	–	–	–
					6C 5 3		–	–	–	–	–	–
							–	–	–	–	–	–
B1 5 2	B5 4 2	BD 4 3	B9 4 3				v	v	–	–	–	–
			BE 4 3			B6 4 2	v	v	–	–	–	–
	B4 4 2	BC 4 3					v	v	–	–	–	–
	56 6 2	5E 7 3					0	v	v	–	–	–
							–	–	–	–	–	–
11 5 2	15 4 2	1D 4 3	19 4 3				v	v	–	–	–	–
							–	–	–	–	–	–
							–	–	–	–	–	–
							v	v	–	–	–	–
								ripristinato				
	36 6 2	3E 7 3					v	v	v	–	–	–
	76 6 2	7E 7 3					v	v	v	–	–	–
								ripristinato				
							–	–	–	–	–	–
F1 5 2	F5 4 2	FD 4 3	F9 4 3				v	v	v	–	–	v
							–	–	1	–	–	–
							–	–	–	–	1	–
							–	–	–	1	–	–
91 6 2	95 4 2	9D 5 3	99 5 3				–	–	–	–	–	–
						96 4 2	–	–	–	–	–	–
	94 4 2						–	–	–	–	–	–
							v	v	–	–	–	–
							v	v	–	–	–	–
							v	v	–	–	–	–
							v	v	–	–	–	–
							–	–	–	–	–	–
							v	v	–	–	–	–



Esempio di diagramma di flusso

Vorrei sottolineare anche un'altro aspetto della programmazione che spesso viene trascurato ma che a parer mio è molto importante ed è la documentazione interna ed esterna.
Per documentazione interna intendo l'inserimento nel programma di note atte a renderlo comprensibile a chiunque. Per documentazione esterna intendo il corredare il programma di diagrammi di flusso e di manuali.

A proposito di DIAGRAMMI DI FLUSSO, normalmente è consigliabile tra l'algoritmo e la sua codifica porre una fase intermedia che è appunto la creazione di un diagramma di flusso cioè la rappresentazione simbolica dell'algoritmo. Si sappia che una ben piccola percentuale di programmatori riesce a scrivere un programma funzionale senza diagramma di flusso e che purtroppo una grande percentuale di programmatori si reputa facente parte di quella piccola schiera.
Si sappia inoltre che oltre l'ottanta per cento dei programmi non girano la prima volta che vengono provati e che una buona documentazione diventa uno strumento per il debugging (collaudo) importantissimo che farà risparmiare molto tempo.
Tutti i computer sono nati per manipolare dell'informazione sia essa sotto forma di numeri o di caratteri.

RAPPRESENTAZIONE INTERNA DELL'INFORMAZIONE

Nel calcolatore tutte le informazioni sono immaginate come gruppi di bit. Un BIT significa un DIGIT BINARIO cioè Ø oppure 1. A causa delle limitazioni dell'elettronica convenzionale risulta pratico rappresentare le informazioni impiegando la logica a due strati, cioè Ø e 1, cioè la logica binaria e virtualmente ne consegue che tutta l'elaborazione dell'informazione attuale è eseguita in formato binario. Nella grande maggioranza dei casi nei microprocessori, e anche nel nostro 6510, questi bit sono strutturati in gruppi di 8, vale a dire di BYTE.

Per ricapitolare velocemente ricordiamo, un DIGIT BINARIO si chiama BIT, un gruppo di 8 BIT si chiama BYTE, un gruppo di 4 bit è chiamato NIBBLE.

Esamineremo ora come l'informazione è rappresentata internamente nel suo formato binario.
Perché tutto funzioni a dovere, è necessario che all'interno del calcolatore siano rappresentate due entità: la prima è il programma con tutta la sua sequenza di istruzioni, la seconda sono i dati sui quali operare che a loro volta possono essere di natura numerica o alfanumerica cioè faciente parta di un testo.

RAPPRESENTAZIONE DEL PROGRAMMA

Le istruzioni di un programma sono internamente rappresentate come BYTE singoli o multipli.
Un'istruzione breve è costituita da un solo BYTE, una più lunga da due o più BYTE.
Essendo il 6510 un microprocessore a 8 byte, preleva in modo sequensiale i byte della memoria, ne deriva che un'istruzione breve è sempre potenzialmente di esecuzione più veloce di un'istruzione a più byte. Questa è la caratteristica più importante di un set di istruzioni che nel caso del 6510 è stato fatto un particolare sforzo per fornirlo di più istruzioni a singolo byte possibile per migliorarne l'efficienza di esecuzione. Il fatto che il nostro microprocessore sia a 8 bit ne risultano importanti limitazioni ma questo è uno degli esempi migliori tra efficienza di velocità e flessibilità di programmazione.
La rappresentazione binaria delle istruzioni è dettata dal costruttore ed il 6510 viene equipaggiato con un set di istruzioni fisso.

RAPPRESENTAZIONE DI DATI NUMERICI

Per rappresentare numeri, la cosa non è sufficientemente immediata ed in alcuni casi deve essere distinta. Dobbiamo tenere sempre presente che si deve essere in grado di rappresentare numeri interi, numeri con segno, cioè positivi e negativi, ed infine numeri decimali. La rappresentazione di numeri interi è eseguita impiegando una rappresentazione BINARIA DIRETTA. Per rappresentazione binaria diretta si intende semplicemente la rappresentazione del valore decimale di un numero nel sistema binario.
Nel sistema binario il bit più a destra rappresenta 2 elevato alla potenza 0. Quello successivo a sinistra rappresenta 2 alla potenza 1, il successivo rappresenta 2 alla potenza 2 ed il bit più a sinistra rappresenta 2 alla potenza 7 = 128:

$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_\emptyset$

rappresenta

$b_7\ 2^7 + B_6\ 2^6 + b_5\ 2^5 + b_4\ 2^4 + b_3\ 2^3 + b_2\ 2^2 + b_1\ 2^1 + b_\emptyset\ 2^\emptyset$

Le potenze di 2 sono:

$^{27} = 128,\ 2^6 = 64,\ 2^5 = 32,\ 2^4 = 16,\ 2^3 = 8,\ 2^2 = 4,\ 2^1 = 2,\ 2^\emptyset = 1$

La rappresentazione binaria dei numeri è omologa a quella decimale, nella quale "123" rappresenta

$$\begin{array}{rrrr} 1 & \times & 100 & = 100 \\ + 2 & \times & 10 & = 20 \\ + 3 & \times & 1 & = 3 \\ \hline & & & = 123 \end{array}$$

Va notato che $100 = 10^2$, $10 = 10^1$, $1 = 10^\emptyset$

In questa notazione precisionale ogni cifra rappresenta una potenza di 10. Nel sistema binario ogni digit o Bit rappresenta una potenza di 2, invece di una potenza di 10 del sistema decimale.

Esempio: ØØØØ1ØØ1 in binario rappresenta:

$$\begin{array}{rrrr} 1 & \times & 1 & = 1\ (2^\emptyset) \\ \emptyset & \times & 2 & = \emptyset\ (2^1) \\ \emptyset & \times & 4 & = \emptyset\ (2^2) \\ 1 & \times & 8 & = 8\ (2^3) \\ \emptyset & \times & 16 & = \emptyset\ (2^4) \\ \emptyset & \times & 32 & = \emptyset\ (2^5) \\ \emptyset & \times & 64 & = \emptyset\ (2^6) \\ \emptyset & \times & 128 & = \emptyset\ (2^7) \\ \hline \multicolumn{4}{c}{\text{in decimale} = 9} \end{array}$$

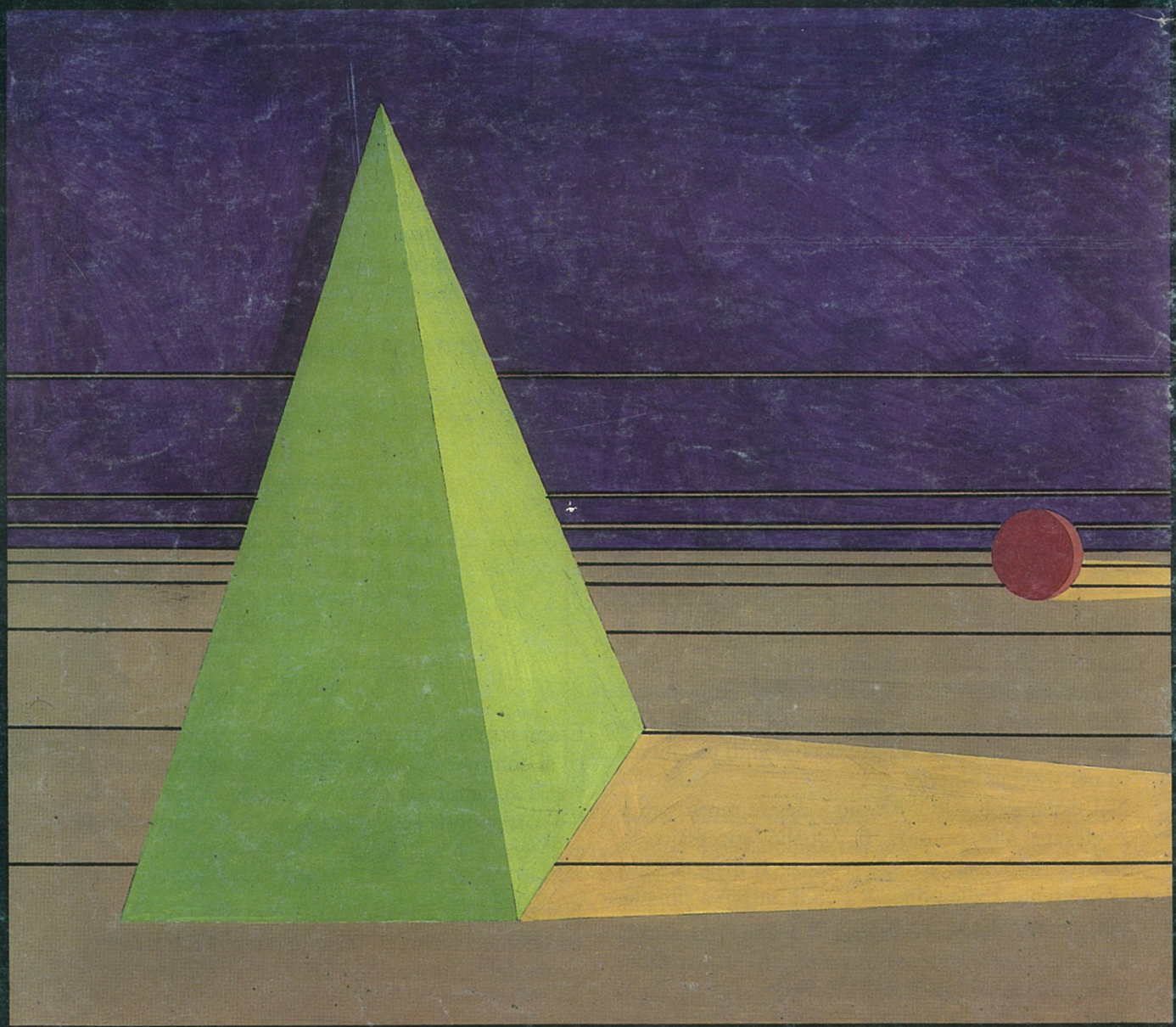
Esaminando la rappresentazione binaria dei numeri, si comprenderà perché i bit sono numerati da y a 7, andando da destra a sinistra. Il bit Ø e "$b_\emptyset$" e corrisponde a 2. Il bit 1 è un "b_1" e corrisponde a 2^1 ecc.

Nel prossimo articolo parleremo delle operazioni con numeri binari ed esamineremo la rappresentazione dell'informazione nei suoi altri aspetti.

STREPITOSO È IN EDICOLA

DREAM

ADVENTURES GAMES AND NEWS



COMMODORE 64