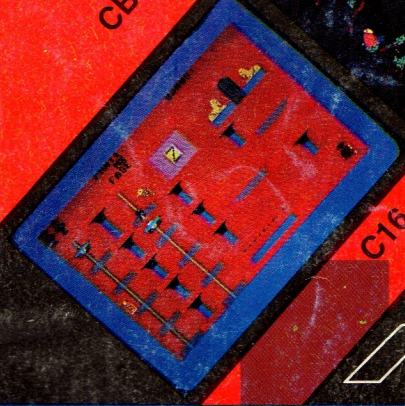
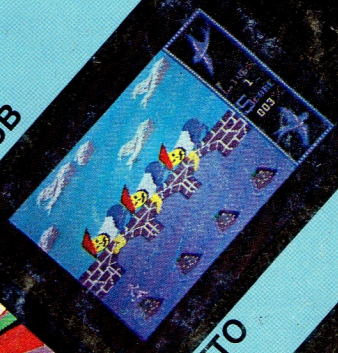
# GO GAMES

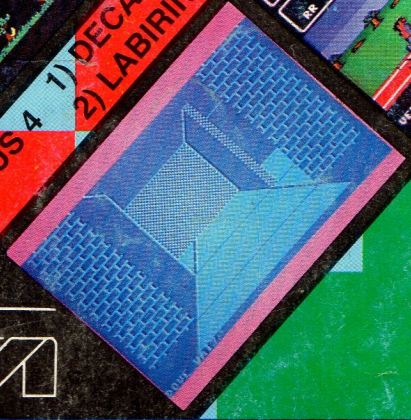
mensile d'informatica e video-games - n. 5 - gennaio 1986 - L. 8000

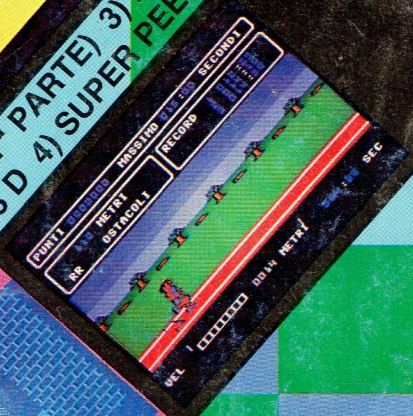
CBM 64 1) SAMURAI 3) MONKY 5) ALIENO 7) GNOMO
2) CAVALIERE BIANCO 4) GALAXIA 6) MISTER SUB





C16 e PLUS 4 1) DECATHLON (1ª PARTE) 3) MANGIATUTTO
2) LABIRINTO 3 D 4) SUPER PEEK-MAN





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 II
N. 5 - Gennaio '86

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

Fotografie di Stefano Monti

**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

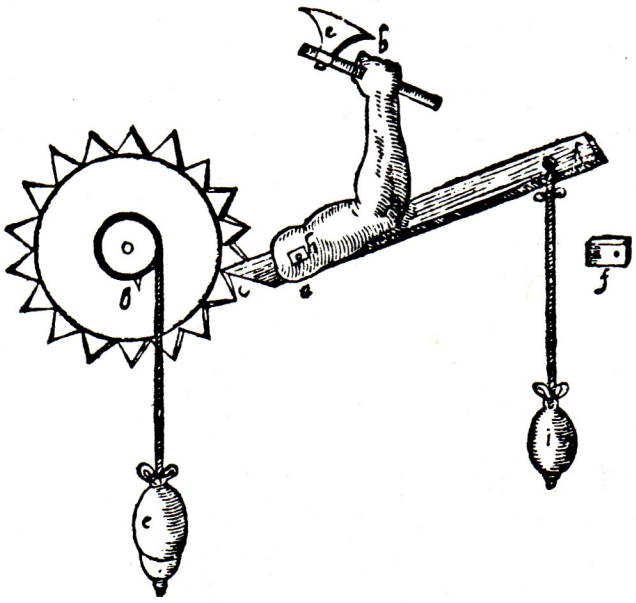
1) **Posizionare il nastro all'inizio del gioco desiderato**
2) **Digitare LOAD e premere RETURN**
3) **Attendere il caricamento**
4) **Digitare RUN e premere RETURN**
5) **Digitare la "parola chiave" del gioco prescelto e premere RETURN**
6) **Attendere il caricamento del gioco**

# l'uomo e il computer 3

La tradizione narra che Michelangelo, appena portata a termine l'imponente statua del Mosé, la guardasse corrucciato ed esclamasse in un impeto di insoddisfazione: « Perchè non parli? »
Le mani dell'*homo faber* non avrebbero potuto scolpire meglio ma il desiderio dell'*homo sapiens* va oltre, pretende che la sua creatura dia un segno di vita; non contento della perfezione della forma, Michelangelo vuole forse che il suo capolavoro non sia soltanto un'opera d'arte ma anche un automa.
È sempre stata un'aspirazione dell'uomo il volersi avvicinare al suo Creatore e costruire qualcosa che, anche soltanto nell'apparenza, sia dotato di un "soffio vitale". È infatti con questa intenzione che sono nati tutti i robots ma anche le macchine in genere. Noi, oggi, abbiamo un altro e ben diverso concetto di macchina. La macchina è un oggetto destinato a compiere una funzione; oppure, per dirlo in termini più "cibernetici": « la macchina è un apparecchio per convertire messaggi d'ingresso in messaggi d'uscita ».([1]) Ci sarebbe da discutere se anche il cervello umano, dato che esprime degli *outputs* se stimolato da particolari *inputs*, sia da considerarsi una macchina. Comunque il *cervello elettronico* lo è e lo sono anche i vari automi che più o meno antropomorfi (cioé simili come forma all'uomo) si vanno sempre più accrescendo intorno a noi.
All'inizio invece, ai primordi della tecnica, questi automi erano considerati soltanto come un trastullo, uno svago manuale e mentale per filosofi (...allora anche gli scenziati erano compresi in quella categoria). Per esempio Archita di Taranto, un seguace di Pitagora, costruì una colomba meccanica in grado di volteggiare leggiadramente nell'aria utilizzando le rudimentali conoscenze dell'epoca per una nuova visione "applicativa" dell'angusta speculazione teorica della geometria. Plutarco accusò Archita « di guastare e corrompere il "bene della geometria", facendola discendere dal regno delle cose incorporee e intellettuali in quello delle cose sensibili, per impiegarla indegnamente nei corpi che hanno bisogno "dell'opera vile e tediosa della mano". »([2])
Ma lasciamo perdere le sottili e sofisticate distinzioni sulla definizione di "scienza". Sta di fatto che questi oggetti, pur anticipando concetti oggi alla base dei meccanismi più elementari, erano allora considerati, anche dai loro stessi inventori, come dei giocattoli. Non si riesce a spiegare altrimenti perché Erone, nel primo secolo d.C., progettò e costruì l'eolipila (una piccola "scatola magica" con una sfera rotante azionata dal vapor acqueo) e non pensò di applicare lo stesso principio per brevettare, con largo anticipo, la macchina a vapore e la turbina.
Forse perché nonostante le varie invenzioni che caratterizzano quell'epoca, che ebbe in Erone un rilevante protagonista, non si pensò di inventare (...e brevettare) il brevetto.
Ma siamo seri e torniamo ai giochini contenuti in un libro dello stesso Erone e che descriveva come realizzare dei teatrini di figure semoventi che davano luogo ad intere rappresentazioni in più atti con tanto di intervalli e cambiamenti di scena. Questo *Automata* è senza dubbio il capostipite degli svariati libri di videogiochi che oggi vanno per la maggiore. Invece dei listati da ricopiare sulla tastiera, venivano

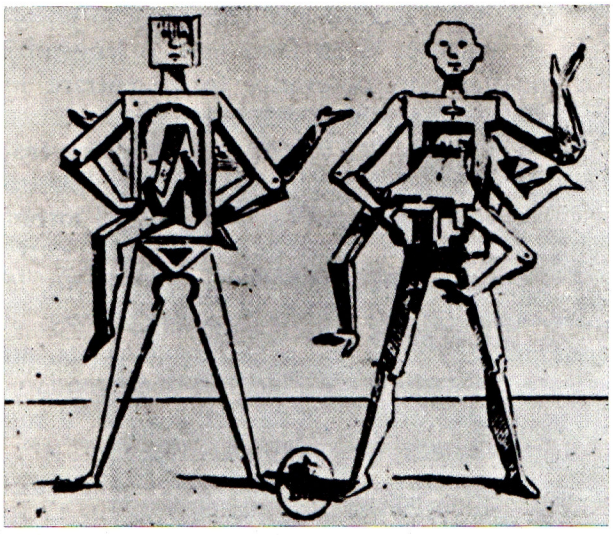

*La colomba di Archita di Taranto in una stampa del '600*

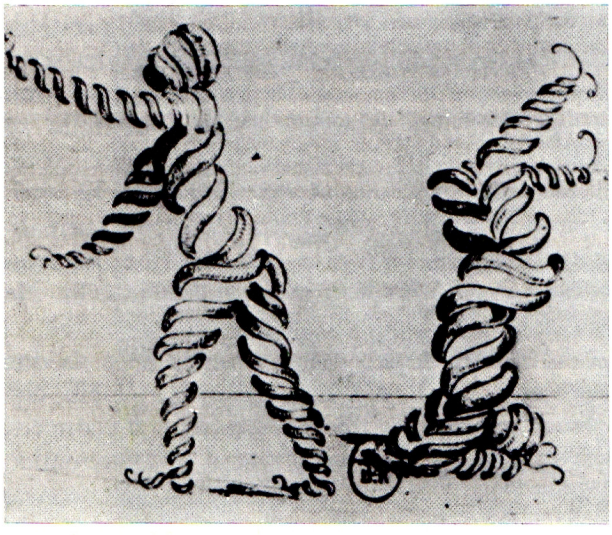
descritte 76 scenografie corredate da tutte le necessarie istruzioni per costruire i semplici meccanismi che, mediante funicelle e contrappesi, facevano muovere i personaggi secondo un *programma* stabilito a priori.



*Uno dei meccanismi descritti da Erone nel libro* **Automata**. *L'immagine è ricavata da una edizione del 1589*

Un altro programma era alla base di una curiosa macchina descritta da Filone di Bisanzio nel suo trattato *Pneumatica*:
in una specie di lavandino l'acqua sgorga dal becco di un cigno (...fin qui niente di speciale...) ma al fianco di questo erogatore c'è un piccolo sportello da cui esce una mano (naturalmente finta...) che porge la pietra pomice (non c'era ancora il sapone). Dopo un po' di tempo l'acqua non scende più e la mano si ritira nello sportello.
Non c'è nulla da sorridere, pensate che questo marchingegno fu inventato più di duemila anni fa...
Pur nei suoi limiti questo era un congegno "utile". Ma dei robots che servono a qualcosa ne parleremo il mese prossimo. Continuiamo invece con gli automi, anche inutili, che hanno l'intenzione di assomigliare a delle forme viventi.
Come ogni altra cosa nel Medio Evo l'automa si colora di magia e stregoneria. Virgilio, un vescovo di Napoli, inventa una mosca di ferro che per otto anni sorveglia la porta della città per impedire l'accesso alle vere mosche portatrici di malattie. È un fatto vero o leggenda? Le mosche non potevano, forse, entrare in città da qualche altra parte?
Ancora ai limiti dell'invenzione fantastica troviamo l'androide costruito dal monaco Alberto Magno: un vero e proprio servitore ed intrattenitore di ospiti. Sempre la leggenda narra che Tommaso d'Aquino, allievo di Alberto, lo distrusse « e non solo per la sua natura di creatura partorita dal demonio, ma per un motivo più pratico e banale, che la leggenda ci ha consegnato con tutta serietà: perché "il suo cicaleccio lo disturbava durante il lavoro". »([3])
Nascono poi animali vari (celebre il gallo meccanico installato nel 1357 sulla cattedrale di Strasburgo che apriva il becco e muoveva coda e ali starnazzando e cantando alle ore prestabilite) e umanoidi con diverse funzioni ma sempre banali. Come il gallo, molti di essi sono infatti destinati a battere con un martello su una campana per segnare le ore; altri sono pure e semplici fantasie, illuminazioni che anticipano le sfrenate immagini dei pittori surrealisti come gli automi disegnati alla fine del sedicesimo secolo da Giovan Battista Bracelli. Ne vediamo alcuni qui a destra.
All'inizio del seicento due personaggi e da due differenti punti di vista, ribaltano quelli che erano sempre stati i concetti e i preconcetti sugli automi. Da una parte il matematico Blaise Pascal sfocia finalmente in una tecnica finalizzata ad uno scopo pratico: dopo lunghi esperimenti e tentativi costruisce la prima calcolatrice meccanica.
Dall'altra parte il filosofo René Descartes (Cartesio, per gli italiani) disserta, ed allora sembrava che vaneggiasse, sull'uomo inteso come macchina ("biologica", aggiungiamo noi): « Non si dà alcuna differenza fra le macchine costruite dagli artigiani e i diversi corpi che la sola natura compone se non questa: che gli effetti delle macchine dipendono solo dall'azione di tubi o molle o altri strumenti che, dovendo avere qualche proporzione con le mani di coloro che li costruiscono, sono sempre così grandi da far apparire visibili le loro figure e i loro movimenti, mentre invece i tubi o le molle che producono gli effetti naturali sono generalmente troppo piccoli per poter essere percepiti dai nostri sensi. »





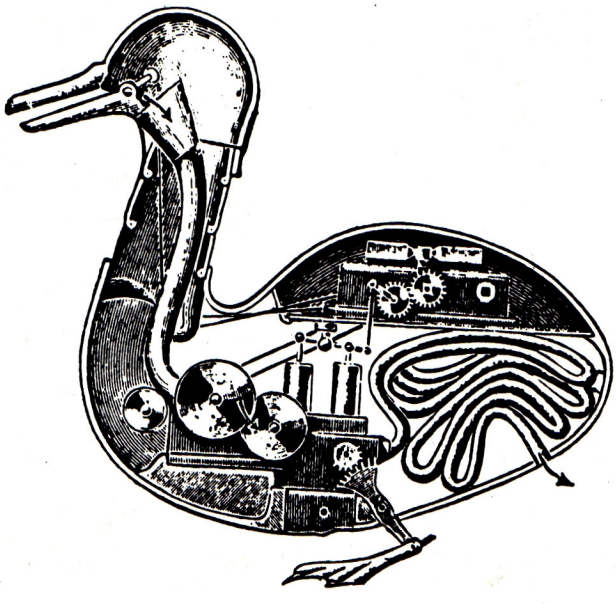
Nasce così il modo moderno di vedere l'automa: non è più un vano e mistico tentativo di imitare la natura ma è un oggetto a sé stante e finalizzato a scopi pratici (secondo Pascal) o uno studio, seppur approssimativo, delle funzioni del corpo umano.

Non ci si stupisce allora se Jacques Vaucanson, forse il più abile artefice di automi meccanici, realizzi verso la metà del settecento un'anatra che unisce alla spettacolarità dei movimenti una più "naturale" affinità con gli esseri viventi: infatti questo uccello meccanico beccava il mangime che gli veniva gettato e dopo averlo "digerito" ne abbandonava i maleodoranti residui!

In una visione totalmente cartesiana ecco come lo stesso autore parla della sua creatura:

« Non credo che gli anatomisti abbiano alcunché da ridire sulle sue ali ... ho imitato, osso per osso, tutte quelle parti che si chiamano apofisi ... l'omero, con il suo movimento di rotazione in tutti i sensi ... e quel che viene chiamato il cubito dell'ala ... tutta la meccanica del *canard*, che sarebbe troppo lungo elencare, sarà vista allo scoperto, perchè il mio scopo non è tanto quello di mostrare, ma di dimostrare ».

Vaucanson è famoso per altri suoi "personaggi viventi" ma l'anitra, il *canard* più volte distrutto e poi ricostruito, rimane ancor oggi il suo capolavoro. Non importa se fu poi dimostrato che la presunta digestione del cibo era soltanto un trucco; ogni rifacimento meccanico o genetico di tutto ciò che è "natura" è comunque un trucco, un artificio... Ciò che conta è l'abilità dell'esecutore e ne fa fede questo stralcio da un giornale dell'epoca: « Dopo una leggera pressione su una parte dello zoccolo di supporto, il *canard* cominciò, con l'aria più naturale del mondo, a guardarsi attorno, gettando sugli spettatori uno sguardo pieno di intelligenza. Riempita una scodella, l'animale affamato vi tuffò con visibile soddisfazione il becco... in un momento, la scodella era vuota a metà... ma la cosa più sbalorditiva fu quando l'animale fece chiaramente comprendere che il suo stomaco era un po' imbarazzato per quel pasto abbon-



*Schema dell'anatra di Vaucanson ricostruita da J.B. Rechsteiner nel secolo scorso*

dante e consumato così in fretta... e dopo poco fummo convinti, nel modo più probante, che si era sbarazzato di questa noia intestina... e l'odore nella sala divenne addirittura insopportabile: allora corremmo tutti intorno all'artista dicendogli tutta la soddisfazione che ci aveva procurato la sua dimostrazione. »

Meno scatologici e di aspetto raffinato e gentile sono invece gli automi costruiti da una famiglia di orologiai (svizzeri, naturalmente) nei lunghi inverni nevosi di fine settecento. Sono tre fratellini: due piccoli maschietti e una ragazza. I loro artefici, Pierre Jaquet-Droz e il figlio Henri-Louis, li vollero ancor più umanizzare dando loro dei nomi: Charles, il primo, è uno scrivano: intinge la penna nel calamaio e con bella calligrafia è in grado di scrivere alcune frasi muovendo con la mano sinistra un telaietto su cui è fissato il foglio di carta. Il secondo automa è Henri. Come il primo è un bimbetto di circa tre anni che riesce a disegnare alcuni soggetti e poi, con un delicato soffio, elimina dalla carta la polvere di piombo in eccesso (allora le matite non usavano la grafite). La terza figura, sempre perfetta nelle forme esteriori e nel meccanismo interiore, è una fanciulla le cui fragili manine muovono le dita su un organo producendo leggiadri brani musicali. Un sistema di mantici le fa muovere delicatamente il petto come se stesse realmente respirando... e alla fine del concerto il suo capo si inchina con gentilezza ringraziando il pubblico.

*Charles Jaquet-Droz, ovvero l'automa che scrive*

# È

# IN EDICOLA

supplemento a Peek n. 4
dicembre '85 - L. 10.000

SPECIALE

VIDEO-GAMES

12 STUPENDI GIOCHI
PER COMMODORE 64

## NON PERDETELO!

# i nostri magnifici supergiochi

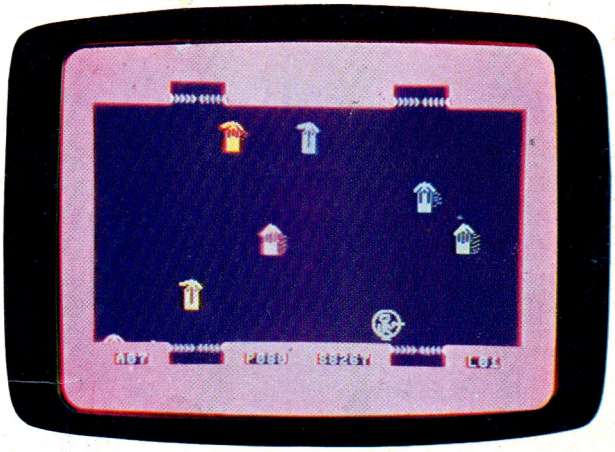
## SAMURAI

**CBM 64 - Joystick in porta 2**
**Parola chiave: ORIENTE**


Samurai è uno stupendo gioco che vi permetterà di immergervi nel misterioso e affascinante mondo orientale.
Vi trovate in un viale di una città misteriosa e dovete percorrerla in lungo ed in largo, cercando i vari tesori che nasconde. Molti passaggi obbligati sono difesi da agguerriti guardiani che eliminerete servendovi delle scimitarre (le troverete lungo il vostro cammino).
Per lanciare le scimitarre premere il pulsante di sparo..... buona fortuna.
Per iniziare premere la barra spaziatrice.

## CAVALIERE BIANCO

**CBM 64 - Joystick in porta 2**
**Parola chiave: NAVETTA**



Un cavaliere galattico a bordo della propria navetta deve percorrere le 120 stanze di una

grande piramide alla ricerca di un misterioso tesoro.
Per passare da una stanza all'altra dovete distruggere con il laser i vari ostacoli e toccare con la navetta più volte i diamanti quando questi saranno di colore verde.
Buon divertimento.

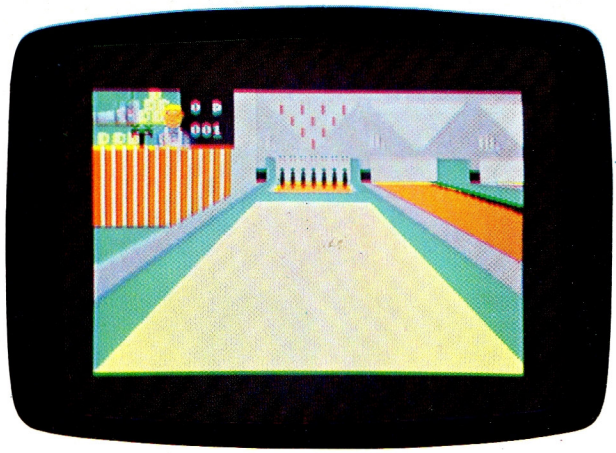
## MONKY

**CBM 64 - Joystick in porta 1**
**Parola chiave: VOLO**


Con la vostra navicella dovrete cercare di atterrare in una foresta per potervi rifocillare, ma alcuni mostri cercheranno di impedirvelo.
Quindi, dovrete abbatterli per riuscire nel vostro intento.
Molte altre sorprese vi attendono nel gioco: fate molta attenzione.
Buona fortuna.
Per iniziare premere Fire.

## GALAXIA

**CBM 64 - Joystick in porta 2**
**Parola chiave: BIRILLO**

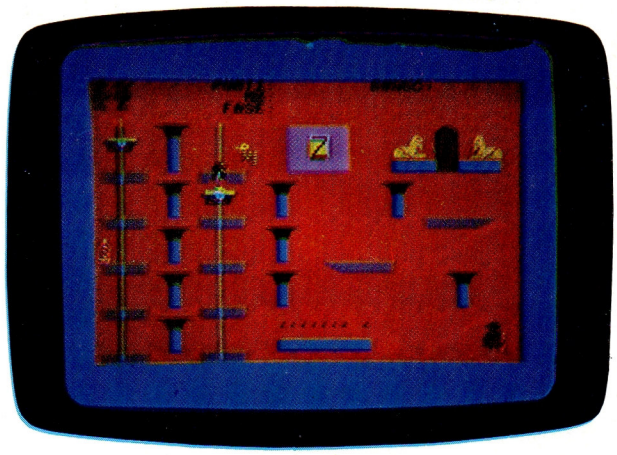


Nella favolosa sala "Galaxia" si svolgono i campionati mondiali di boowling. Se vuoi partecipare puoi farlo, la pista n. 1 è a tua disposizione. Per selezionare il livello di difficoltà muovere il joystick verso destra e premere il pulsante sul numero desiderato. Dopo il lancio della palla si può correggere il tiro muovendo il joystick verso destra o verso sinistra.
Buon divertimento.

## ALIENO

**CBM 64 - Joystick in porta 2**
**Parola chiave: LUNARE**

Siamo nel 2050 e un poderoso attacco alieno minaccia la città dove abitate. Voi siete stati incaricati di difenderla con la vostra potentissima astronave.
Le fasi dell'attacco sono numerose, quindi dovrete superare moltissimi livelli di difficoltà per ottenere lo scopo.
Buon divertimento.
Per iniziare premere Fire.



## MR. SUB

**CBM 64 - Joystick in porta 2**
**Parola chiave: ORANGO**

Misurate la vostra abilità alla "guida" del misterioso personaggio. Il gioco si divide in diversi capitoli ognuno di questi presentano una prova diversa di sopravvivenza per il nostro eroe.
Certamente non sarà facile.
Buona fortuna!!

## GNOMO

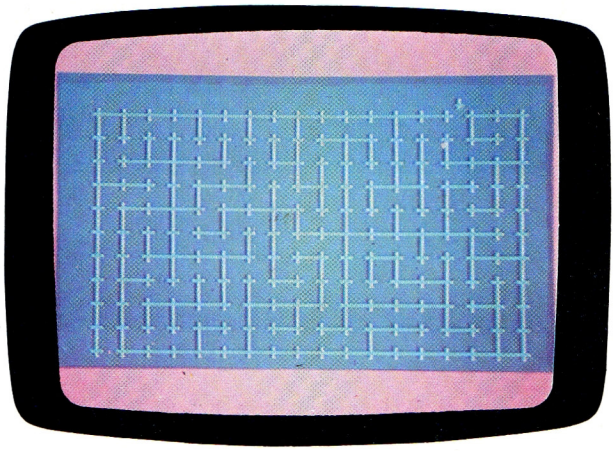
**CBM 64 - Joystick in porta 2**
**Parola chiave: FANTASMA**

Guidate l'omino alla ricerca delle lettere che formeranno delle parole magiche. Alcuni personaggi sono in agguato per impedirglielo, quindi sarà vostro compito evitarli per poter continuare la missione. Attenzione alle mosse che fate, perché i vostri "nemici" sono furbi e... veloci!!
Buon divertimento.

## DECATHLON (1a PARTE)

**C16 e PLUS 4 - Joystick in porta 2**
**Parola chiave: OSTACOLI**

Questo gioco è solamente per i veri atleti. Nella prima parte delle Olimpiadi dovete competere in quattro specialità: 100 metri piani, 400 metri piani, 110 metri ostacoli, 200 metri ostacoli.
Per correre devi muovere il joystick velocemente e per saltare devi premere il tasto di fuoco. Durante la gara puoi controllare la velocità (sulla sinistra), i metri percorsi (al centro) e il tempo istantaneo (a destra): questo per facilitarti il compito di qualifica-

zione. Il tempo massimo è visualizzato in alto a destra. Cerca di migliorare il record del mondo per ottenere un alto punteggio.
Nel prossimo numero di Go Games troverete la seconda parte di queste favolose Olimpiadi.

## LABIRINTO 3 D

**C16 e PLUS 4 - Tastiera**
**Parola chiave: PIANTINA**

**Tasti:**
**R - destra**
**L - sinistra**
**F - avanti**
**H - aiuto**

Non potevi capitare in un labirinto più difficile di questo! Un labirinto che sfrutta le capacità del computer per farsi vedere tridimensionalmente. Cerca di trovare l'uscita nel minor tempo possibile. Il computer cercherà di darti una mano quando premerai il tasto di aiuto: vedrai il labirinto con la tua esatta posizione per alcuni secondi.
Puoi scegliere la grandezza del labirinto all'inizio del gioco tra un minimo di 2 per 1 e un massimo di 17 per 10. In questo labirinto il computer ti aiuterà moltissimo ma non per questo il tuo compito sarà facile.

## MANGIATUTTO

**C16 e PLUS 4 - Joystick o Tastiera**
**Parola chiave: INSETTI**

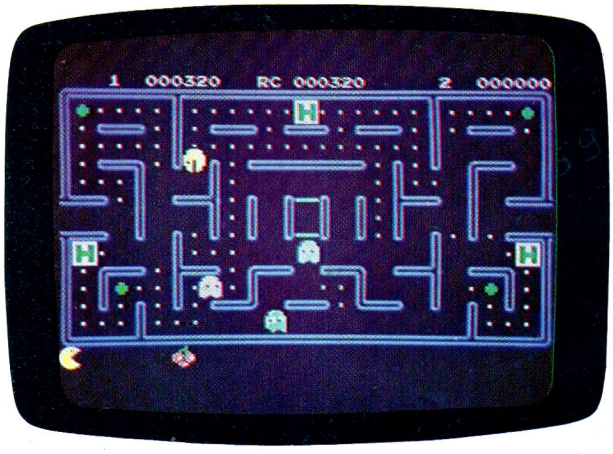
**Tasti:**
**Z - sinistra**
**X - destra**
**K - alto**
**M - basso**
**SHIFT - fuoco**

Con la tua astronave sei sbarcato su un pianeta sconosciuto popolato da strani insetti. Con le armi a tua disposizine devi distruggere i nemici per passare ad uno schermo più difficile ed impegnativo.
Stai attento: questo games non è come quelli da bar perché ha una difficoltà in più. C'è, infatti, una rampa di lancio (sulla sinistra) che spara dei pericolosissimi missili che possono colpirti. Quindi oltre a fare attenzione ai vari bruchi spaziali, dovrai stare attento anche ai missili. Per selezionare la tastiera o il joystick devi premere rispettivamente i tasti K o J. Per memorizzare i record con Z e X scegli le lettere e con SHIFT le memorizzi.

## SUPER PEEK-MAN

**C16 e PLUS 4 - Joystick o Tastiera**
**Parola chiave: PILLOLE**

**Tasti:**
**1 - seleziona 1 giocatore**
**2 - seleziona 2 giocatori**
**A - alto**
**Z - basso**
**3 - sinistra**
**4 - destra**




In questa versione di PEEK-MAN abbiamo cercato di riprodurre fedelmente il gioco da bar sia nella grafica che nel suono.
Ed ecco che, dopo mesi di fatica, siamo riusciti a realizzare SUPER PEEK-MAN!!! Un gioco di fantasmini e pillole energetiche estremamente entusiasmante. Abbiamo anche cercato di renderlo diverso dal solito inserendo, oltre alle pillole di energia, delle porte di energia.
Quando entri in una di queste porte come per magia ti ritrovi in un'altra parte dello schermo. Il resto del gioco è perfettamente uguale a quello da bar ma per i più smemorati lo ricordiamo: dovete cercare di mangiare tutti i puntini sparsi nel percorso.
Quando mangiate le pillole di energia (i punti grossi) i fantasmini cambiano colore e voi potrete mangiarli. Attenzione al tempo perché l'effetto delle pillole finische dopo alcuni secondi. Un consiglio? Cercate di non gareggiare in velocità con i fantasmi! Loro sono più veloci.

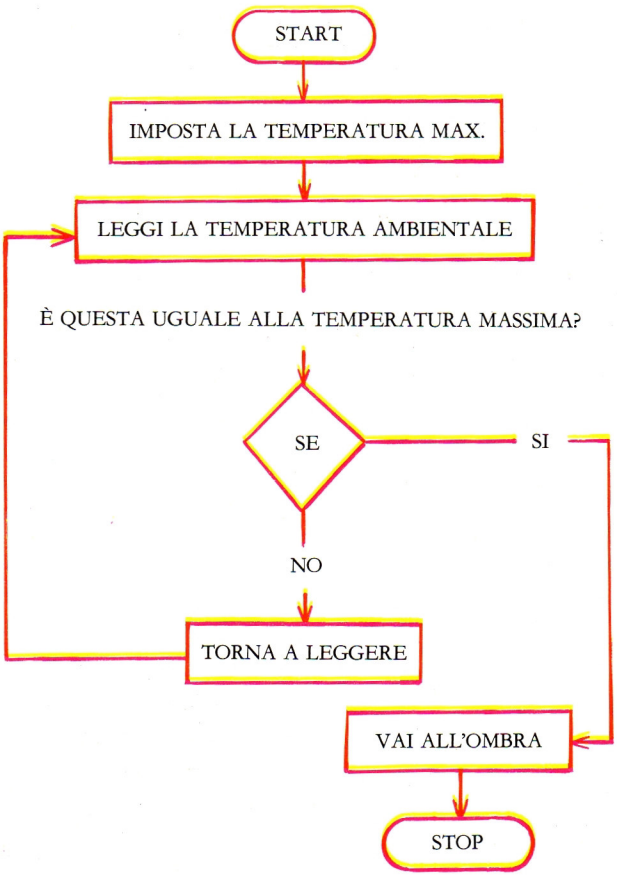
**BUON DIVERTIMENTO!**

**E NON DIMENTICATE IL PROSSIMO NUMERO DI GO GAMES**

# impariamo a usare il computer 3

## LA LOGICA DEL COMPUTER

Uno degli elementi che caratterizzano maggiormente l'attività di colui che per le prime volte si accosta all'uso del computer è la necessità di prestare molta attenzione alla variazione di alcune abitudini mentali acquisite prima del suo approccio.
Se vogliamo ben vedere, un esempio di questo modo nuovo di pensare come impostare e risolvere i problemi, ci è dato proprio dal tentativo di risolvere le equazioni di grado secondo e superiori. In questo caso siamo abituati ad attivare alcune procedure standard imparate sui banchi di scuola le quali ci assicurano il corretto risultato che andiamo cercando.
Purtroppo il computer invece non possiede (chiaramente se non lo istruiamo) questo principio di automatismo.
Ma anche la implementazione (realizzazione operativa) di un programma che risolva le equazioni non è per nulla un fatto semplice e scontato: occorre conoscere bene quali sono le risorse della macchina che possono essere utilizzate, descrivere una loro applicazione formale corretta ed infine controllare che il metodo matematico prescelto possa portare ai risultati richiesti senza ambiguità di sorta.
Ciò evidenzia la necessità di impostare bene « a tavolino » il flusso delle operazioni logiche che il computer dovrà svolgere: ma non corriamo troppo... fermiamoci un attimo e sviluppiamo alcune considerazioni sul linguaggio BASIC.
Una delle caratteristiche peculiari che lo contraddistinguono è la possibilità che esso fornisce di impostare una sequenza di decisioni logiche e di farle eseguire dal computer secondo un ordine prestabilito che si sviluppa in relazione ai risultati emersi dai controlli stessi.
Tipico esempio in merito sono i salti condizionati.
Chi di noi non ha mai pensato: « vado ad acquistare il giornale in quell'edicola; se non lo trovo lì andrò altrove » oppure: « vado a prendere il sole fintanto che non sarà troppo caldo, allora tornerò all'ombra. »
Bene, noi sappiamo che il BASIC ci dà la possibilità oltreché di effettuare dei salti incondizionati all'interno del programma (l'istruzione GO TO), anche di impostare delle vere e proprie operazioni di controllo logico attraverso i salti condizionati.
Il susseguirsi di una serie di questi (come se di continuo misurassi la temperatura al sole per decidere se ritornare all'ombra) genera un LOOP o anello che controlla di continuo il valore di una data variabile fintanto che esso non corrisponde ad un numero prestabilito: in quel caso il LOOP si interrompe ed il programma prosegue nella sua esecuzione (come se il valore della temperatura al sole avesse raggiunto il limite della tollerabilità tanto da decidere il ritorno all'ombra).
Proviamo dunque a descrivere questo programma antiscottatura!




In realtà il BASIC permette di utilizzare sei OPERATORI RELAZIONALI che possono servire per controllare se le caratteristiche di stringhe alfanumeriche e di variabili corrispondano o meno alle condizioni imposte.
Questi operatori sono:

| | |
|---|---|
| UGUALE A | = |
| NON UGUALE A | <> |
| MINORE DI | < |
| MINORE O UGUALE A | <= |
| MAGGIORE DI | > |
| MAGGIORE O UGUALE A | >= |

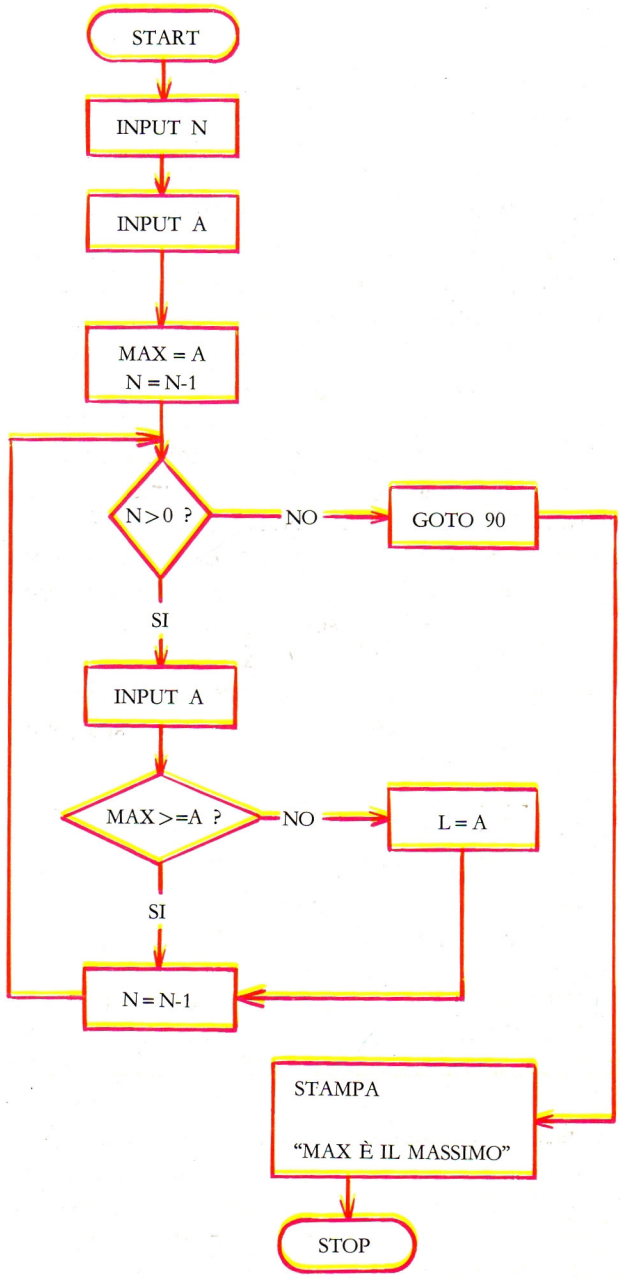
Abbiamo detto che gli OPERATORI RELAZIONALI possono essere applicati a variabili numeriche ed alfanumeriche, vediamo ora alcuni esempi:

A=23 X>=3.56 B> (X2+X3) *8
Z6<>Z5 A<=X*G A$="ROSSI"
X$<>Z$ B$>A$

Nel linguaggio BASIC il salto condizionato si attua attraverso listruzione IF-THEN (SE-ALLORA):

10 IF N<=0 THEN 90

dove 10 è il numero di identificazione della linea di programma in attività e 90 quella cui il programma salta se è confermata la condizione N<=0.



Il listato in BASIC:

```
10 input n, a
20 let l=a
30 let n=n-1
40 if n<=0 then 90
50 input a
60 if l>=a then 80
70 let l=a
80 let n=n-1
85 goto 40
90 print l
100 end
```

Qui sopra possiamo ora vedere in termini più operativi l'uso di questa istruzione, attraverso un programma che determina il numero più alto di un gruppo non ordinato.
Il diagramma a flusso ci evidenzia che è proprio una doppia applicazione degli operatori relazionali il cuore del programma: alle linee 40 e 60 troviamo appunto espresse due condizioni che definiscono l'esaurimento dei numeri (n) del gruppo ed il maggiore tra di essi.

Possiamo ora far ritorno all'obbiettivo che ci eravamo preposti all'inizio: la risoluzione delle equazioni.
Prendiamo ad esempio la equazione

$$X^5 + 3X^2 - 10 = 0$$

Questa equazione richiede, per trovare il valore di X, l'applicazione di una procedura iterativa, che definisca il valore di X per successive raffinazioni.
Possiamo riscrivere la stessa equazione in una differente forma:

$$X = \sqrt[5]{10 - 3X^2}$$

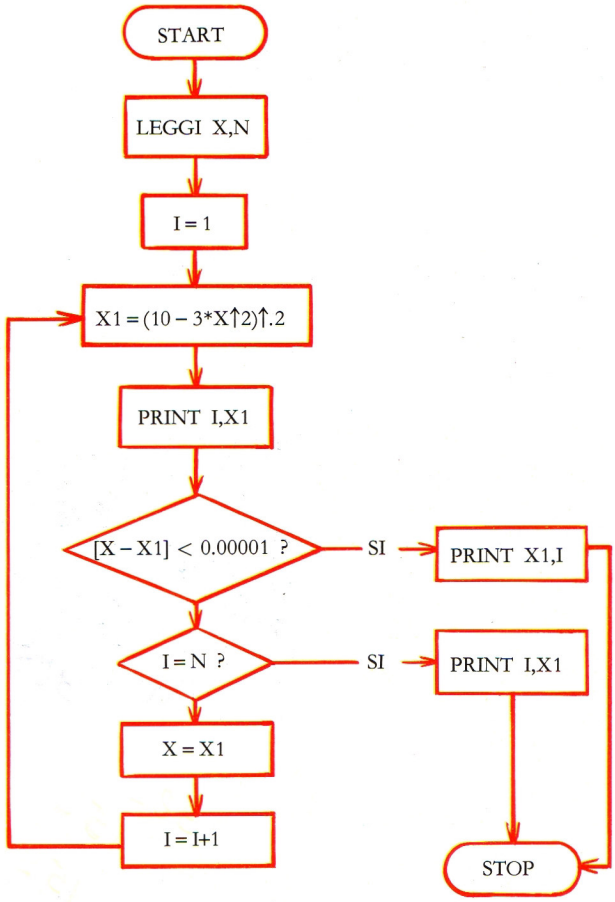
Applicando il metodo della convergenza possiamo far eseguire i calcoli sino a quando la differenza tra due successivi valori di X è adeguatamente bassa, oppure quando un alto numero di iterazioni è stato effettuato.
In pratica avremo una serie di operazioni di questo tipo:

$$X = \sqrt[5]{10 - 3(1.0)^2} = 1.47577$$

$$X = \sqrt[5]{10 - 3(1.47577)^2} = 1.28255$$

$$X = \sqrt[5]{10 - 3(1.28255)^2} = 1.38344$$

e via di seguito sino a che i valori di X tenderanno a convergere in quello che potrà essere considerato un risultato valido.
Passiamo ora a definire il diagramma di flusso del programma per la risoluzione di questa equazione:




espresso in BASIC:

```
10 INPUT X,N
20 PRINT
30 LET I=1
40 LET X1=(10-3*X↑2)↑.2
50 PRINT "I=";I,"X1=";X1
60 IF ABS(X-X1)<.00001 THEN 110
70 IF I=N GOTO 160
```

```
80 LET X=X1
90 LET I=I+1
100 GOTO 40
110 PRINT
120 PRINT " IL RISULTATO E' X=";X1
130 PRINT
140 PRINT " NUMERO DI ITERAZIONI
RICHIESTE=";I
150 GOTO 190
160 PRINT
165 PRINT " NON VI E' CONVERGENZA
DOPO ";I;" ITERAZIONI "
170 PRINT
180 PRINT " L'ULTIMO VALORE DI X
E'";X1
190 END
```

Per scrivere questa routine in Basic abbiamo definito i seguenti simboli:

X = valore di X sostituito nella parte destra dell'equazione.

X1 = a ogni iterazione, nuovo valore calcolato di X.

I = contatore dell'iterazione che viene incrementato di 1 ad ogni successiva iterazione.

N = numero massimo di iterazioni permesse oltre le quali l'elaboratore blocca il calcolo.

L'elaboratore continuerà il calcolo fino a che non si avvererà una delle seguenti possibilità:

1) la differenza del valore di due successivi X sia minore di 0.00001.
2) il numero di iterazioni contato in I raggiunga il massimo da noi ammesso.

Per far eseguire il programma all'elaboratore è sufficiente immettere nell'ordine (come si vede anche dal diagramma di flusso) il valore di X e il valore di N. Dopodiché l'elaboratore eseguirà i calcoli secondo la procedura definita sino a raggiungere l'ipotesi 1 o 2 sopra citate.

Per oggi basta, col prossimo numero riprenderemo il nostro discorso sulla programmazione.

# impariamo a programmare in assembler 2

**commodore 64 - commodore 64 - commodore 64 - commodore 64 - commodore**

## Linguaggio macchina del C64/2

*di Marco Maffei*

Prima di riprendere la mia trattazione sul linguaggio macchina, vi rubo poche righe per rinfrescarvi i concetti chiave esposti nel passato articolo, in cui, dopo aver parlato di algoritmi, di diagramma di flusso e di rappresentazione dell'informazione, abbiamo chiarito che i calcolatori trattano l'informazione sottoforma di numeri o di caratteri e stabilito che possiamo considerare l'informazione sia come un attributo esterno sia come un attributo interno alla macchina.

Finora abbiamo iniziato a trattare unicamente della sua rappresentazione interna che abbiamo suddiviso in tre entità fondamentali: il programma; i dati sui quali opera e l'eventuale testo alfanumerico.

Dopo esserci soffermati sul programma, abbiamo introdotto la rappresentazione dei dati numerici, accennando al sistema di numerazione binario che – come è arcinoto – può rappresentare con le sole due cifre – Ø e 1 – qualunque numero. In tale sistema, poichè la base è 2, ogni unità di un dato ordine vale 2 unità dell'ordine immediatamente inferiore, quindi 2 unità – cioè una coppia – formano un'unità del secondo ordine, 2 unità del secondo ordine – cioè due coppie – formano un'unità del terzo ordine e così via.

Date queste premesse, ora possiamo introdurre le regole che permettono le operazioni fondamentali con numeri espressi nel sistema binario, per arrivare a descrivere in fine **nel prossimo numero** le rappresentazioni esterne dell'informazione.

Chiariamo subito che le operazioni fondamentali dell'aritmetica sono indipendenti dalla base che si sceglie per il sistema di numerazione. Quindi, anche per l'ADDIZIONE, si procede operando con la regola già nota per il sistema decimale, facendo, però, attenzione che occorre riportare una unità nell'ordine superiore ogni volta che se ne hanno due nell'inferiore. Possiamo sintetizzare con il seguente schemino:

Ø + Ø = Ø
Ø + 1 = 1
1 + Ø = 1
1 + 1 = (1) Ø dove(1) indica un riporto (carry) di 1

Ora passiamo ad esaminare alcuni esempi di addizione confrontando il sistema decimale a quello binario:

| SISTEMA DECIMALE | SISTEMA DECIMALE |
|---|---|
| 2 + | 1Ø |
| 1 = | Ø1 |
| 3 | 11 |
| | non c'è alcun riporto |

| SISTEMA DECIMALE | SISTEMA BINARIO |
|---|---|
| 3 + | Ø11 |
| 1 = | ØØ1 |
| 4 | 1ØØ |

Partendo dalla colonna di destra abbiamo la somma delle unità semplici 1 + 1 che equivale ad un'unità del secondo ordine; si scrive perciò Ø nella colonna delle unità semplici e si riporta 1 (carry) in quella delle unità del secondo ordine incui troviamo: 1 + Ø = 1 + 1 di carry per cui scriviamo Ø nella colonna del secondo ordine con carry 1 che verrà sommato alla colonna di terzo ordine in cui troveremo Ø + Ø = Ø/+ 1 di carry = 1.

Ora consideriamo l'addizione binaria proprio in rapporto al nostro computer. Abbiamo già chiarito che il microprocessore 6510 è un 8 bit, tratta, cioè, 8 bit alla volta (1 byte) per cui in esso si possono rappresentare DIRETTAMENTE – utilizzando appunto 8 bit – i numeri interi razionali da ØØØØØØØØ a 11111111 che corrispondono ai numeri indicati da Ø a 255 nel sistema decimale.

Dobbiamo rilevare che i limiti che si verificano con l'utilizzo di 8 bit sono evidenti; infatti, i numeri rappresentabili in tal modo DIRETTAMENTE con il sistema binario sono INTERI RAZIONALI ASSOLUTI (cioè privi di segno) e necessariamente confinati entro l'ottavo ordine, quindi minori o uguali a 255.

Possiamo comuque ovviare ad entrambi gli inconvenienti sopra esposti.

Incominciamo a considerare come possiamo rappresentare (utilizzando i nostri 8 bit) con il sistema binario i numeri INTERI RAZIONALI RELATIVI (cioè preceduti dal segno + o dal segno –). Per indicare il segno del numero, ricorriamo al bit più a sinistra (l'ottavo) che perderà così la sua funzione numerica: per tradizione con Ø (all'ottavo posto) indicheremo un numero positivo (segno +); mentre con 1 (all'ottavo posto) indicheremo un numero negativo (segno –). Questo tipo di rappresentazione si chiama NOTAZIONE BINARIA CON SEGNO.

Ora proviamo a leggere il numero indicato con il sistema binario "11111111": se rappresenta un numero razionale ASSOLUTO (senza segno) corrisponde al numero indicato nel sistema decimale con 255; se, invece, rappresenta un numero razionale RELATIVO (preceduto da segno) corrisponde al numero indicato nel sistema decimale con –127, mentre "Ø1111111" corrisponde a +127.

Possiamo concludere che con 8 bit si possono rappresentare numeri razionali relativi da –128 a + 127. (**Se utilizzassimo 16 bit – vedi sopra – saremmo in grado di rappresentare numeri da –32K a +32K ecc.**).

Ora possiamo introdurre l'ADDIZIONE nel sistema binario di numeri RELATIVI DISCORDI e verificare se la nostra rappresentazione di numeri positivi e negativi del sistema binario (NOTAZIONE BINARIA CON SEGNO) sia valida anche nel momento in cui non ci limiteremo alla lettura, ma eseguiremo dei calcoli ben precisi.

Come già per l'addizione dei numeri razionali assoluti, esaminiamo ora un esempio di addizione di numeri relativi discordi, confrontando il sistema decimale a quello binario.

| DECIMALE | BINARIO | |
|---|---|---|
| +15 | ØØØØ1111 | |
| – 5 = | 1ØØØØ1Ø1 | |
| +1Ø | 1ØØ1Ø1ØØ | leggi in sistema decimale –2Ø |

Questo risultato non è esatto; ne consegue che l'addizione di numeri relativi espressi nel sistema binario con questa rappresentazione non opera in modo corretto. Così, considerando che un calcolatore non deve essere in grado di rappresentare un'informazione (nel nostro caso numeri positivi e negativi), ma deve anche eseguire delle operazioni aritmetiche, dobbiamo trovare un'altra soluzione alla rappresentazione dei numeri relativi nel sistema binario. Introduciamo, quindi, la rappresentazione COMPLEMENTO A DUE.

Allo scopo di meglio comprendere il complemento a due, incominciamo a considerare una fase intermedia: IL COMPLEMENTO A UNO.

Nella rappresentazione in complemento a uno tutti i numeri positivi (con segno +) sono rappresentati come nella NOTAZIONE BINARIA CON SEGNO (per esempio + $3_{(10)}$ = $ØØØØØØ11_{(2)}$). I loro opposti – cioè i numeri

che hanno lo stesso valore assoluto e segno negativo – (per esempio –3) si otterranno complementando ogni bit della rappresentazione originaria: ogni Ø diverrà 1 e ogni 1 sarà trasformato in Ø (rifacendoci al nostro esempio –3 sarà rappresentato con 11111ØØ).
Esaminiamo ora un altro esempio di addizione di numeri relativi discordi espressi nel sistema binario in rappresentazione COMPLEMENTO A UNO per verificare la validità operazionale di tale convenzione.

DECIMALE
– 4
+ 6 =
2

$+ 4_{(10)} = ØØØØØ1ØØ_{(2)}$
$– 4_{(10)}$ = COMPLETAMENTO A 1
11111Ø11

BINARIO
11111Ø11
ØØØØØ11Ø
(1) ØØØØØØØ1 leggi 1 con riporto (1)

Anche in questo caso la rappresentazione del numero relativo a segno negativo potrebbe essere valida, ma le operazioni aritmetiche non funzionano: il problema non è ancora risolto. Così ci troviamo a considerare un'evoluzione della rappresentazione in complemento a uno, finalmente la RAPPRESENTAZIONE IN COMPLEMENTO A DUE.
Nella rappresentazione in complemento a due avremo i numeri positivi normalmente rappresentati in NOTAZIONE BINARIA CON SEGNO e i numeri negativi ottenuti sommando 1 (ØØØØØØØ1) al loro complemento a uno.
Proviamo ora a sperimentare la validità operazionale di quest'ultima convenzione di rappresentazione eseguendo nel sistema di numerazione binario la seguente addizione di numeri relativi discordi che nel sistema decimale è indicata con la scrittura:

$+3 \quad -5 \quad = \quad -2$

$+3_{(10)} = ØØØØØØ_{(2)}$
$+5_{(10)} = ØØØØØ1Ø1$ il suo complemento a 1 è 11111Ø1Ø

ora eseguiamo:

11111Ø1Ø
ØØØØØØØ1
11111Ø11 abbiamo così ottenuto il COMPLEMENTO A DUE

Ora possiamo sommare:

ØØØØØØ
ØØØØØØ11
11111Ø11 =
1111111Ø chiaramente un numero negativo

Per leggere il risultato dovremo compiere l'operazione inversa a quella che ci aveva permesso di calcolare il complemento a due del 5 negativo di partenza; quindi ricavare il complemento a uno di 1111111Ø = ØØØØØØØ1 a cui sommare 1:

ØØØØØØØ1
ØØØØØØØ1 =
ØØØØØØ1Ø leggi $+2_{(10)}$

Concludiamo che il risultato 1111111Ø rappresenta nel corrispondente sistema decimale –2, cioè è esatto.
Siamo finalmente pervenuti – pur senza fornire la dimostrazione matematica completa – a una convenzione di rappresentazione di numeri razionali interi di segno negativo espressi nel sistema binario: IL COMPLEMENTO A DUE, i risultati in tal modo ottenuti eseguendo operazioni aritmetiche sono infatti esatti, sia nel valore assoluto sia nel segno. Il riporto deve essere trascurato.
Ovviamente con la rappresentazione COMPLEMENTO A DUE siamo anche in grado di eseguire qualsiasi tipo di **sottrazione** di numeri interi razionali assoluti, considerando, appunto il SOTTRAENDO come numero relativo di segno negativo.
Proponiamoci, ad esempio, di volere eseguire nel sistema binario la sottrazione che nel sistema decimale è indicata con la scrittura:

$7 - 4 = 3$

Si ha: $7_{(10)} = ØØØØØ111_{(2)}$ e $4_{(10)} = ØØØØØ1ØØ_{(2)}$
completamento a uno: 11111Ø11
completamento a due: 11111ØØ
quindi:

ØØØØØ111
11111ØØ =
(1) ØØØØØØ11

Trascuro (come si è detto) il riporto (1) e leggo direttamente in quanto il risultato è un

numero positivo: ØØØØØØ11$_{(2)}$ = 3$_{(10)}$; è esatto.
D'ora in poi tutti i numeri interi razionali negativi (fra cui il sottraendo in una sottrazione fra numeri razionali assoluti) verranno – in modo implicito – rappresentati interamente con il sistema binario nella notazione in complemento a due.
Per comodità la Fig. 1 rappresenta una tabella dei numeri in complemento a due.
Consideriamo ora il seguente esempio di addizione nel sistema binario in cui il carry di riporto sia essenziale per ottenere un risultato esatto e completo. Come per gli esempi precedenti – per facilitarvi la lettura – eseguiamo la nostra operazione confrontando il sistema decimale a quello binario.

| DECIMALE | BINARIO |
|---|---|
| 128 + | 1ØØØØØØØ |
| 129 = | 1ØØØØØØ1 |
| 257 | (1) ØØØØØØØ1 |

Come si può vedere il risultato sarà corretto unicamente se terremo conto dell'(1) di riporto che necessariamente dovrà occupare un nono bit – cioè il bit 8 come spiegato nel numero precedente.
In questo caso il bit 8 è chiamato carry – "C" –e viene assunto come nono bit del risultato, in modo da poter riconoscere che il risultato dell'esempio precedente è 1ØØØØØØØ1$_{(2)}$ = 257$_{(10)}$.
Ma noi sappiamo che all'interno del nostro microprocessore, i registri utilizzati per conservare l'informazione sono larghi solo otto bit, quindi nella memorizzazione del risultato normalmente sono conservati soltanto i bit da Ø a 7. Per cui un riporto (carry – C –) richiede un procedimento particolare: deve, cioè, essere rivelato da specifiche istruzioni per poi essere elaborato.
Consideriamo ora la somma:

| DECIMALE | BINARIO |
|---|---|
| 64 + | Ø1ØØØØØØ |
| 65 = | Ø1ØØØØØ1 |
| 127 | 1ØØØØØØ1 |

in cui è stato generato un riporto interno dal bit 6 al bit 7. Questo è ciò che viene definito un OVERFLOW (che si chiamerà formalmente V).
Il risultato ottenuto è accidentalmente negativo. In ogni caso simile a quello sopra prospettato è indispensabile indicare la condizione di OVERFLOW in modo da poter intervenire.
Proviamo ora a vedere cosa succede in un caso come il seguente:

| DECIMALE | BINARIO | |
|---|---|---|
| – 1 + | 11111111 + | |
| – 1 = | 11111111 = | |
| – 2 | (1) 1ØØØØØØ1 | leggi in decimale –2 |

Anche in questo caso è stato generato un carry interno dal bit 6 al bit 7 che, però, a sua volta ha provocato un riporto dal bit 7 al bit 8 (il carry – C – formale). Ma le regole del sistema binario rappresentato in complemento a due specificano che questo riporto deve essere ignorato, perciò, – limitandoci alla lettura fino al bit 7 – il risultato ottenuto nell'addizione di esempio è corretto. Ne consegue una considerazione: il riporto dal bit 6 al bit 7 che determina un ulteriore riporto al bit 8 non cambia il segno. Quindi, quando si opera su numeri negativi e l'overflow non è semplicemente un riporto dal bit 6 al bit 7 NON ci troviamo realmente di fronte a una condizione di OVERFLOW.
Un ultimo esempio per chiarire la condizione di OVERFLOW:

| DECIMALE | BINARIO | |
|---|---|---|
| – 64 + | 11ØØØØØØ + | |
| – 65 = | 1Ø111111 = | |
| –129 | (1) Ø1111111 | = 127$_{(10)}$ |

Questa volta il riporto non si è verificato dal bit 6 al bit 7, ma l'addizione dei termini all'ottavo ordine ha cambiato il segno – cioè il bit 7 – e ha determinato un riporto esterno. In questo caso è indispensabile indicare la condizione di OVERFLOW.
Riassumendo, l'OVERFLOW si verificherà in queste quattro situazioni:

1) somma di numeri positivi che genera un risultato di grandezza superiore a quella consentitaci;
2) somma di numeri negativi che genera un risultato di valore assoluto superiore a quello consentitoci;

## TABELLA DEI COMPLEMENTI A DUE

| SEGNO + | COMPLEMENTO A DUE | SEGNO − | COMPLEMENTO A DUE |
|---|---|---|---|
| | | − 128 | 10000000 |
| + 127 | 01111111 | − 127 | 10000001 |
| + 126 | 01111110 | − 126 | 10000010 |
| ..... | | ..... | |
| + 51 | 00110011 | − 51 | 11001101 |
| + 50 | 00110010 | − 50 | 11001110 |
| ..... | | ..... | |
| + 41 | 00101001 | − 41 | 11010101 |
| + 40 | 00101000 | − 40 | 11010110 |
| ..... | | ..... | |
| + 31 | 00011111 | − 31 | 11100001 |
| + 30 | 00011110 | − 30 | 11100010 |
| ..... | | ..... | |
| + 21 | 00010101 | − 21 | 11101011 |
| + 20 | 00010100 | − 20 | 11101100 |
| ..... | | ..... | |
| + 11 | 00001011 | − 11 | 11110101 |
| + 10 | 00001010 | − 10 | 11110110 |
| + 9 | 00001001 | − 9 | 11110111 |
| + 8 | 00001000 | − 8 | 11111000 |
| + 7 | 00000111 | − 7 | 11111001 |
| + 6 | 00000110 | − 6 | 11111010 |
| + 5 | 00000101 | − 5 | 11111011 |
| + 4 | 00000100 | − 4 | 11111100 |
| + 3 | 00000011 | − 3 | 11111101 |
| + 2 | 00000010 | − 2 | 11111110 |
| + 1 | 00000001 | − 1 | 11111111 |
| + 0 | 00000000 | | |

*Fig. 1*

3) sottrazione di un numero positivo di grandezza elevata da un numero negativo di valore assoluto elevato;
4) sottrazione di un numero negativo di valore assoluto elevato da un numero positivo di grandezza elevata.

All'interno del nostro microprocessore i bit CARRY e OVERFLOW vengono denominati « FLAG » e sono utilizzati mediante una procedura che prenderemo in considerazione in un prossimo articolo. Questi due indicatori – **CARRY e OVERFLOW** – sono posizionati in un registro speciale – detto « REGISTRO DI STATO » o « DEI FLAG » – che contiene altri indicatori le cui funzioni saranno anch'esse chiarite in un prossimo articolo.

Tecnicamente l'indicatore di overflow, cioè il « flag » di overflow, sarà posto ad 1 quando:

A) si verifica un riporto dal bit 6 al bit 7 senza riporto esterno;
B) non si verifica un riporto dal bit 6 al bit 7 ed avviene un riporto esterno;
C) compiendo un OR ESCLUSIVO dal bit 7 entrante ed uscente (il bit del segno).

Risolta la rappresentazione in sistema binario dei numeri interi razionali relativi, resta da risolvere il problema della grandezza. Se vogliamo rappresentare, cioè, numeri interi di grandezza superiore all'ordine consentitoci, bisognerà necessariamente usare più byte.

Il numero di byte deve **rimanere fisso per tutta la durata di uno stesso programma** per una ragione di esecuzione efficente di operazioni aritmetiche; quindi, una volta scelto il numero di byte, la grandezza massima del numero rappresentabile sarà, ovviamente, fissa.
Abbiamo già chiarito che utilizzando 8 bit (che è il numero di bit per volta su cui il microprocessore opera) siamo limitati all'utilizzo di numeri interi razionali relativi che vanno da −128 a +127 (chiaramente insufficiente in molte applicazioni) e che per aumentare tale arco di rappresentazione è necessario utilizzare un formato a due, tre, oppure n byte. Proviamo a considerare come esempio un formato in doppia precisione a 16 bit:

| bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Ø | = | DECIMALE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | = | Ø |
| | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | 1 | = | 1 |
| | Ø | Ø | Ø | Ø | Ø | Ø | Ø | Ø | 1 | Ø | Ø | Ø | Ø | Ø | Ø | Ø | = | 128 |
| | Ø | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | 32.767 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | − 1 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | Ø | = | − 2 |

Purtroppo, però, anche questo metodo presenta evidenti svantaggi. Per addizionare due numeri sarà necessario operare sommando 8 bit alla volta con conseguente rallentamento dell'elaborazione.
Questa rappresentazione, inoltre, impiega 16 bit per qualsiasi numero anche se esso potrebbe essere rappresentato con 8 bit.
Consideriamo ora un punto importante: il numero – qualunque esso sia – scelto per la rappresentazione in complemento a due, resterà invariato anche se qualsiasi risultato o calcolo intermedio dovesse generare un numero che richieda più bit di quelli fissati all'inizio. La conseguenza logica è l'inevitabile perdita di alcuni bit – nel nostro caso il programma conserva i bit di sinistra (quelli più significativi) e perde quelli di basso ordine – generando il troncamento del risultato.
La rappresentazione con più byte è chiamata anche « RAPPRESENTAZIONE IN FORMATO FISSO » e può causare – come abbiamo visto – una perdita di precisione che generalmente è tollerabile per calcoli comuni e per le operazioni matematiche. Nel caso di calcoli contabili, purtroppo non è ammessa alcuna perdita di precisione, quindi sarà necessario – anzi indispensabile – utilizzare un altro tipo di rappresentazione.
La soluzione più comune è la rappresentazione BCD, cioè *decimale codificato binario* (per dirla in italiano).

Per questo numero direi di fermarci qui per lasciarvi il tempo necessario a sorbirvi tutta questa sfilza di Ø e di 1 nella speranza di essere stato chiaro.
Tengo a precisare che mi rendo perfettamente conto che quanto è stato finora trattato non risulta proprio divertente, anzi – diciamolo pure – risulta decisamente noioso, ma è fondamentale per un uso « spigliato » del linguaggio macchina del vostro COMMODORE 64, pertanto vi saluto anticipandovi gli argomenti della prossima puntata.
Il mese venturo parleremo di: codice BCD, rappresentazione a virgola mobile e rappresentazione di dati alfanumerici, concludendo così la rappresentazione interna. Quindi esauriremo l'argomento rappresentazione con la rappresentazione esterna. Se lo spazio concessomi me lo consentirà, inizieremo a trattare dell'organizzazione hardware del nostro microprocessore 6510. Al mese prossimo...

**BUON LAVORO**

# STREPITOSO È IN EDICOLA

# TUTTOCBM64

TUTTOCBM64
TUTTOCBM64

## 8 video games