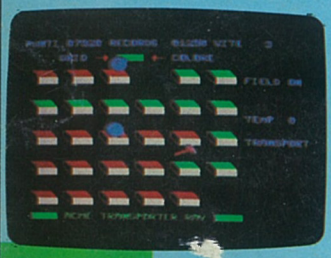
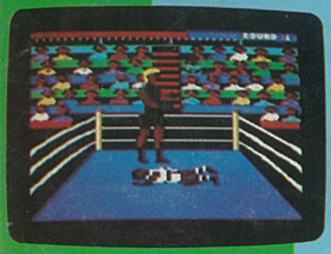
GO GAMES

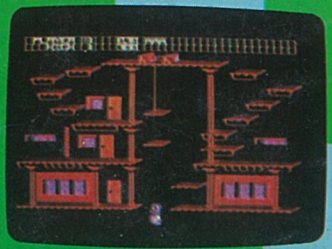
mensile d'informatica e video-games - 13 ottobre '86 - L. 8000

7 VIDEO-GAMES per CBM 64 e 128

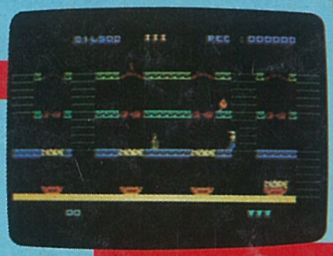


1 - RAID
2 - MR. TOM
3 - LANDING
4 - MEGABOX
5 - WARRIORS
6 - MOLLA
7 - HAMMER

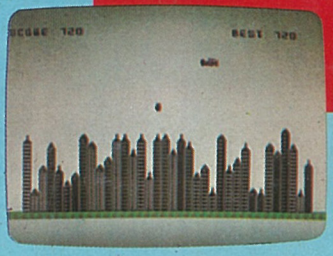


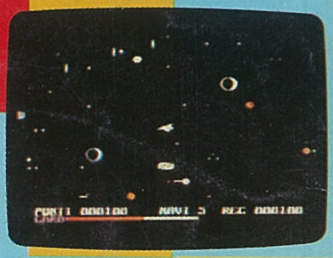


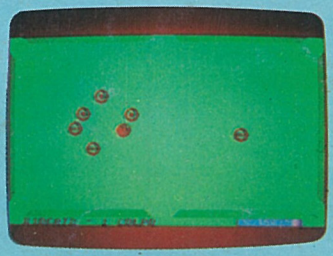
7 VIDEO-GAMES per C 16 e PLUS 4

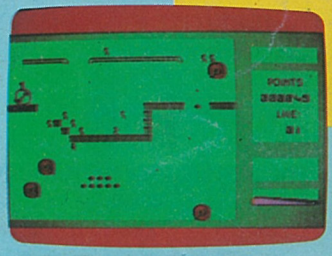


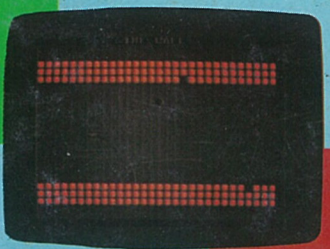
1 - BIPLANO
2 - BILIARDO
3 - MILLEPIEDI
4 - SISTEMA SOLARE
5 - IL CUOCO
6 - THE WALL
7 - VIETNAM











Editions Fermont s.r.l. - Milano - DISTRIBUZIONE: Messaggerie Periodici - V.le Famagosta, 75 - 20142 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. 13 - Ottobre '86

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

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

FOTOLITO:
Claudio Lavezzi
Via Terruggia, 3
20162 Milano

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

DISTRIBUZIONE:
MePe
V.le Famagosta, 75
20142 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:
I bollettini dei versamenti devono essere scritti in stampatello o a macchina.
Quelli poco chiari verranno cestinati.

ATTENZIONE

CBM 64

Per il CBM 64 ti proponiamo un nuovo sistema di caricamento che ti permette di scegliere il gioco che vuoi caricare e di posizionare il nastro con l'avanzamento veloce (F.FWD) subito prima del gioco da te prescelto, quindi di procedere al caricamento normale. Con questo sistema eviti di dover passare tutto il nastro per cercare il programma che ti interessa.
Le operazioni da fare sono:

1) **Digita Load e premi Return.**
2) **Attendi che sul video compaia la presentazione.**
3) **Premi Stop sul registratore.**
4) **Dopo qualche secondo apparirà una schermata con l'elenco dei giochi preceduto da un numero e la scritta «Programma N°» col cursore che lampeggia.**
5) **Inserisci il N° corrispondente al programma desiderato e premi «Return».**
6) **Comparirà la scritta «premi F.FWD» quindi il registratore si fermerà subito prima del programma da te scelto. A questo punto premi «STOP» e successivamente premi «PLAY».**

AVVERTIMENTO: se lo schermo si riempirà di righe colorate significa che il caricamento procede regolarmente. Se non escono le righe torna indietro all'inizio del gioco e premi nuovamente Play.

C16 / PLUS 4

Ecco le istruzioni per il caricamento dei programmi: Avvolgere completamente la cassetta dalla parte che si desidera caricare. Quindi digitare LOAD & RETURN e far iniziare il caricamento. Quando ricompare il cursore digitare RUN & RETURN ed attendere. La prima volta che si caricano i programmi conviene azzerare il contatore del registratore alla fine dell'avvolgimento e scrivere il numero dell'inizio del gioco in modo che in un tempo successivo si conosce l'esatto inizio del gioco.

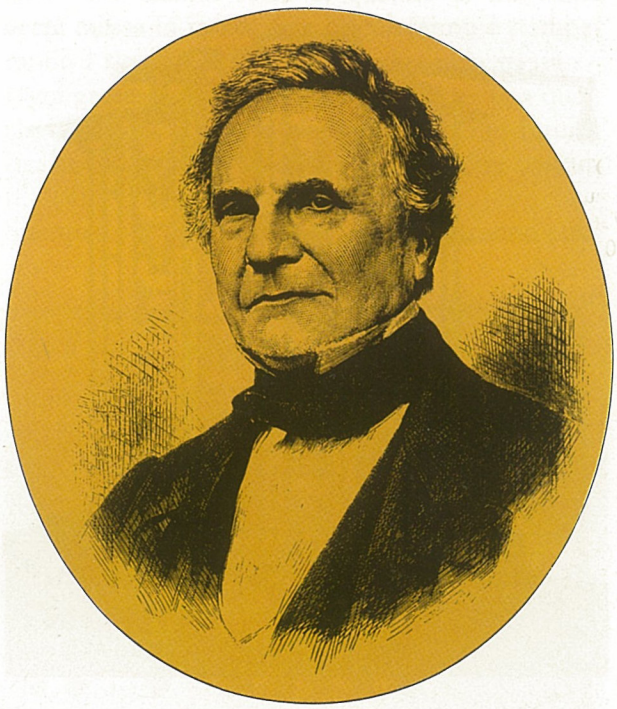
l'uomo e il computer 11

Nella figura qui a lato vediamo il protagonista di questa puntata: il matematico inglese Charles Babbage. Ne avevamo già parlato il mese scorso presentandone il carattere un po' irascibile e scontroso ma anche l'indubbia genialità applicata ai più vari campi della scienza e della tecnica. Ne abbiamo anche sottolineato la pignoleria e la meticolosità che gli facevano scrivere: "Gli errori derivati dalla scarsità di dati sono molto più numerosi e più duraturi di quelli che provengono da un cattivo ragionamento a partire da dati esatti." (1)

I dati a disposizione del matematico sono quelli che si trovano sulle tavole numeriche che venivano compilate con un lungo e faticoso lavoro per essere poi pubblicate ed usate.

Babbage (ma non solo lui) riscontra che parecchie tavole contengono degli errori: errori di stampa, piccoli errori dovuti all'approssimazione, ma anche errori "lasciati ad arte da alcuni autori gelosi, che contavano di smascherare così più facilmente i frequenti plagi". (2)

Babbage si mette perciò a confrontare tavole di vari autori e scopre che quelle redatte da Taylor contengono 19 errori... "Il 'Nautical Almanac' del 1832 pubblica la lista dei diciannove errori. Però ci si avvede che v'è un errore nell'errata corrige, cosicché nel 1833 si deve pubblicare un 'erratum of the errata'; in questa correzione si inserisce un nuovo errore e, poiché nel frattempo l'almanacco per il 1835 è già stato stampato, si deve attendere fino al 1836 per stampare l''erratum of the erratum of the errata' ed avere così l'edizione corretta delle tavole..." (3)



Charles Babbage

Per evitare tutte queste catene di errori, già molti anni prima (nel 1812) Babbage aveva pensato di far eseguire i lunghi e laboriosi calcoli da una macchina. Ecco come lui stesso ci descrive la nascita di quella idea: "Per quanto ricordo, la primissima idea della possibilità di calcolare tavole numeriche per mezzo di una macchina si affacciò alla mia mente così: una sera, nella sala della Analytical Society di Cambridge, mentre mi stavo appisolando con una tavola di logaritmi aperta sul tavolo davanti a me, un altro socio, entrando

nella stanza ed accorgendosi della sonnolenza cui stavo per cedere, esclamò: 'Babbage, che stai sognando?'. Io risposi, additando i logaritmi: 'Penso che tutte queste tavole potrebbero essere calcolate a macchina.'" (4)
È nato così da un sogno ad occhi chiusi quello che sarebbe poi diventato per Babbage un sogno ad occhi aperti prima di trasformarsi in un incubo che lo seguirà fino alla fine dei suoi giorni.
Negli anni 1820-22 elabora una prima versione della "macchina alle differenze", che verrà poi sostituita da una seconda versione nel 1823. Nella fig. 2 vediamo la ricostruzione del primo modello della macchina.

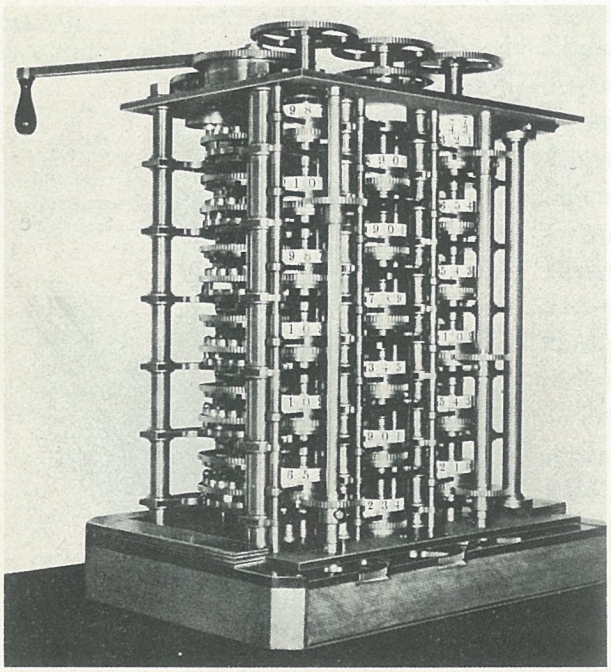


Fig. 2

Come funziona questa macchina? "La macchina differenziale non è fatta per rispondere a domande particolari. Suo scopo è di calcolare e stampare una serie di risultati derivanti dall'applicazione di talune leggi assegnate." (4) Non è quindi una semplice 'calcolatrice' come quelle di Pascal o Leibniz; servendosi della sola operazione di addizione riesce a calcolare i valori di funzioni abbastanza complesse giocando sulle differenze prime, seconde, ecc. ad ognuna delle quali è abbinata una colonna di ruote numerate.
È forse più chiaro un esempio. Prendiamo la funzione $Y = X^2 - X$ e facciamo una tabella:

valore di X	valore di Y	differenza Prima (*)	differenza seconda
1	0	0	0
2	2	2	2
3	6	4	2
4	12	6	2
5	20	8	2
...	...	...	...

(*) differenze tra il valore di Y e il suo precedente

Le colonne di ruote che costituiscono la macchina (vedi ancora la fig. 2) mostrano queste differenze che quando diventano costanti (nel nostro esempio la colonna delle differenze seconde) permettono alla macchina di calcolare il valore di Y sommando, per colonne successive di ruote, le differenze seconde alle prime e poi le prime alla colonna dei valori e trovare così il dato richiesto. Se volessimo avere il valore di Y con X=6 basterebbe perciò aggiungere 2 and 8 (diff. prima = 10) e poi 10 a 20 (Y=30).
Ovviamente la macchina non è velocissima tuttavia il Governo inglese diede il suo benestare per il finanziamento avendone riconosciuto gli altri e ben più importanti meriti.
Solo che Babbage è un perfezionista e come tale è sempre insoddisfatto: si susseguono progetti di migliorie, aumentano le colonne e il numero delle cifre che la macchina può trattare e così si giunge a quel settembre 1834 in cui Babbage è folgorato da una nuova e più entusiasmante idea...
Lasciamolo un attimo e seguiamo brevemente gli sviluppi di questa prima macchina.
Nel 1853 lo svedese Georg Scheutz costruì un altro esemplare della macchina alle differenze perfettamente funzionante (vedi fig. 3). Babbage non ne fu per niente invidioso, anzi appoggiò il collega, si rammaricò perché la Royal Society non ne avesse riconosciuto il merito e qualche anno dopo la addusse come esempio 'vivente' della possibilità e della necessità di continuare i finanziamenti per quella ricerca.

Anche se Babbage ormai era già un passo più avanti. Già nel 1834 gli era infatti balenata l'idea di una macchina più complessa che non solo riuscisse a 'calcolare' ma che, seguendo un programma articolato e predisposto dall'uomo, fosse anche in grado di memorizzare i risultati parziali ottenuti per riutilizzarli all'occorrenza.

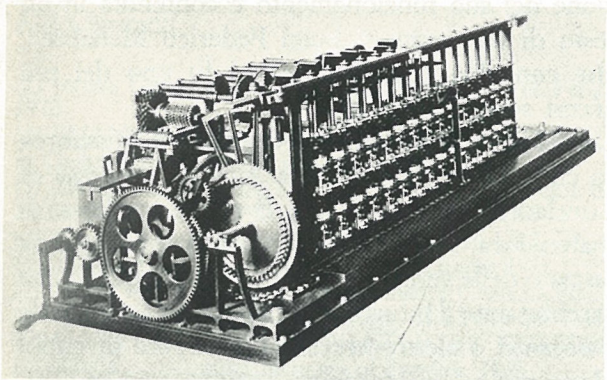


Fig. 3

Ecco quindi il progetto del primo elaboratore: nascono nuovi concetti come 'store' (magazzino dei dati ovvero memoria) 'mill' ('mulino' ovvero la parte della macchina interessata all'esecuzione delle operazioni) ed anche 'programma' che avrebbe dovuto essere inserito nella macchina mediante delle schede perforate. Ne vediamo alcune nella fig. 4. Sono del tutto simili, nella forma e nell'uso a quelle usate anni prima da Jacquard per il suo telaio automatico.

"La macchina analitica è quindi una macchina di natura quanto mai generale. Qualunque sia la formula da sviluppare bisogna comunicare alla macchina la legge per il suo sviluppo mediante due gruppi di schede. Quando queste siano state collocate, la macchina è impostata per quella formula particolare. Il valore numerico delle sue costanti deve allora essere posto sulle colonne di ruote che stanno sotto e, quando la macchina verrà messa in moto, esse calcoleranno e stamperanno i risultati numerici della formula stessa.

Ogni gruppo di schede predisposto per una qualsiasi formula potrà, in qualsiasi momento futuro, ricalcolare la medesima formula con altre costanti." (4)

Stampati i risultati in tavole, questi potranno esse-

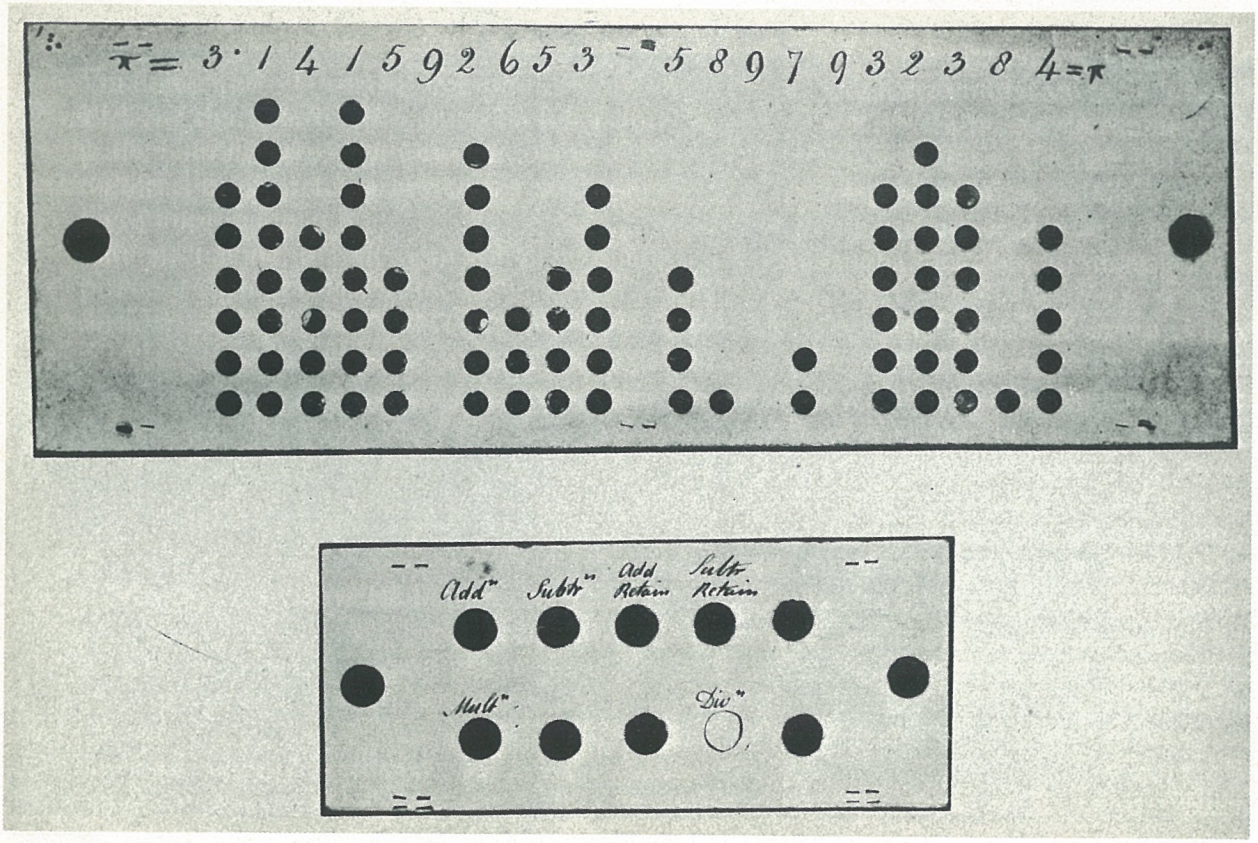


Fig. 4

re riinseriti nella macchina in qualsiasi momento "ma i calcoli che si possono fare con l'aiuto della macchina si fanno con tanta rapidità che sembra più probabile che essa taglierà corto, calcolando direttamente le formule adatte anziché ricorrere alle tavole da essa stessa calcolate." (4)
È un'ottima idea, forse troppo precorritrice per quell'epoca ancora immatura giacché il Governo inglese nel 1842 blocca i finanziamenti poiché i disegni e i progetti di Babbage non sono seguiti da un 'prodotto'.
Lo schema generale della 'macchina analitica' ed un disegno di un particolare sono rappresentati nelle fig. 5 e 6.
Come si può immaginare i maggiori problemi di Babbage stavano nella difficoltà tecnica di realizzare minuscoli e precisissimi meccanismi con le tecnologie dell'epoca. Senza il supporto del Governo, Babbage si deve autofinanziare giungendo a pagare un disegnatore una ghinea al giorno (a quell'epoca, il prezzo di venti chili di carne).
"I progetti di questa macchina sono contenuti in più di 90 disegni ognuno largo tre piedi e alto due. Le notazioni meccaniche con le quali sono rappresentati i vari movimenti occupano all'incirca 400 grandi fogli. Gli schizzi, le descrizioni e le note originali sono contenuti in quattro piccoli volumi..." (5)
Sebbene sia studiata nei minimi dettagli, la macchina analitica rimane sulla carta.
Oltre a queste note di Babbage, la miglior descrizione del suo funzionamento è contenuta in un testo di un torinese, Luigi Federico Menabrea, che con altri scienziati italiani fu uno dei più accesi sostenitori dell'idea.
La traduzione inglese dell'articolo di Menabrea fu curata e anotata da Ada Augusta contessa di Lovelace, figlia del Poeta Byron e studiosa di matematica. Lady Lovelace, nelle sue note, ci ha lasciato delle vere e proprie profezie su quello che sarebbe stato il futuro degli elaboratori (quando da meccanici o elettromeccanici sarebbero diventati elettronici): "La Macchina Analitica non ha alcuna pretesa di *creare* qualcosa. Essa può eseguire tutto ciò che noi *sappiamo domandarle* di eseguire. Essa può seguire un'analisi ma è incapace di *prevedere* una relazione analitica o un fatto reale." (6)

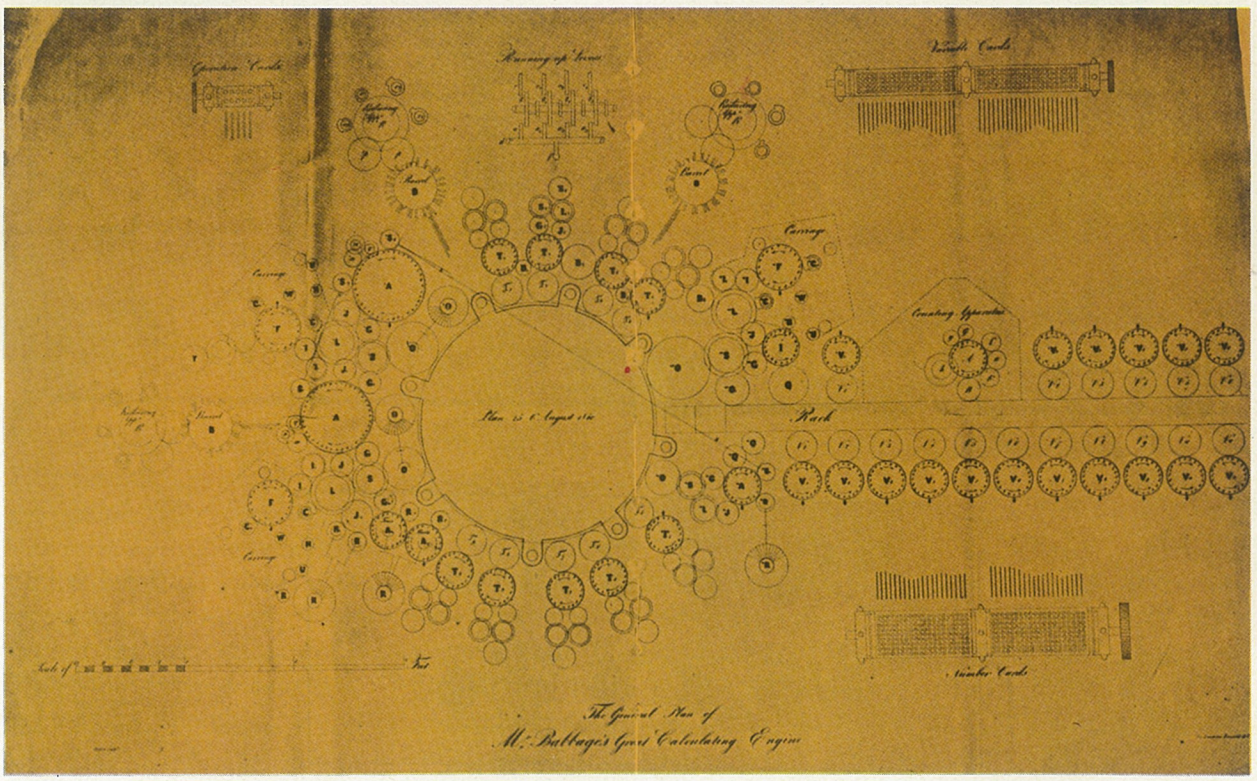


Fig. 5

Sembra una frase scritta oggi, una frase che descrive perfettamente il ruolo e i limiti degli elaboratori, ma queste parole risalgono al 1843...

Byte byte a tutti

Aldo Spinelli

(1) Jeremy Bernstein, LES ORDINATEURS, Dunod, Parigi 1970.
(2) CHARLES BABBAGE E LA SUA CALCOLATRICE, in Civiltà delle Macchine, sett./ott. 1955.
(3) BABBAGE, LA MACCHINA ANALITICA, a cura di Mario G. Losano, Etas Kompass libri, Milano 1973.
(4) Charles Babbage, PASSAGES FROM THE LIFE OF A PHILOSOPHER, Dawson of Pall Mall, Londra 1968.
(5) da una lettera di Babbage al matematico Giovanni Plana
(6) Ada Augusta contessa di Lovelace, SKETCH OF THE ANALYTICAL ENGINE INVENTED BY CHARLES BABBAGE..., Scientific Memoirs of London, 1843.

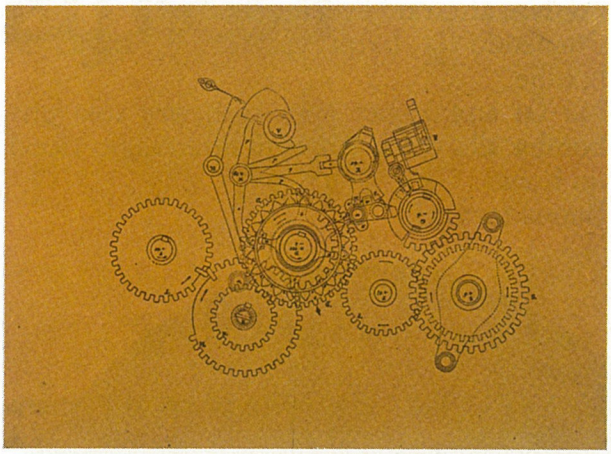
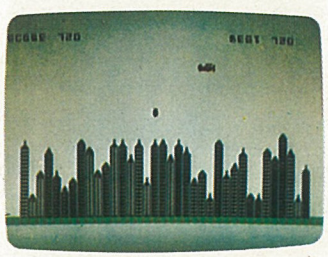


Fig. 6

i nostri magnifici supergiochi

BIPLANO

C 16 e PLUS 4 - Tastiera



TASTI:
Spazio - sgancio bombe

Il tuo nome è Ranki, il Barone Rosso del XX secolo.
Nell'ultima battaglia aerea sopra la città ne-

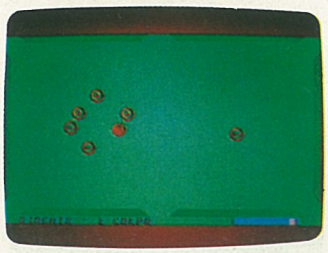
mica, sei rimasto senza carburante e la tua unica speranza è di radere al suolo la città per fare atterrare l'aereoplano.
Premendo la barra spaziatrice, sganci le bombe.
Ma fai attenzione: puoi sganciare solo una bomba alla volta. Ad ogni atterraggio ricevi un bonus.
Buona fortuna!

BILIARDO

C 16 e PLUS 4 - Joystick in porta 2 - Tastiera

TASTI:
↑ - Su
↓ - Giù
→ - Destra
← - Sinistra

Esc - Menù principale
1 - 1 giocatore in ogni ordine
2 - 1 giocatore palle numerate
3 - 1 giocatore palle e buche numerate
4 - 2 giocatori palle e buche numerate
5 - 2 giocatori all'americana
6 - 2 giocatori Su o Giù
7 - migliori punteggi

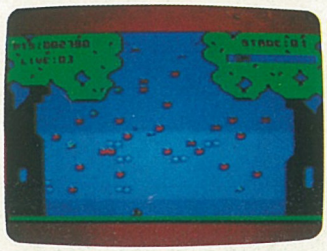


Destreggiatevi con tiri ad effetto mentre sfidate il computer o un vostro amico in una partita di biliardo.
Potrete giocare ai vari giochi sopra elencati mentre scommettete milioni di dollari.
Buon divertimento e... vinca il migliore!!

MILLEPIEDI

C 16 e PLUS 4 - Joystick in porta 1 - Tastiera

TASTI:
S - Su
R - Giù
Return - Fuoco
D - Sinistra
6 - Destra



Versione per il vostro commodore 16 e PLUS 4 del famosissimo gioco da bar Centipede. Con il vostro cannoncino dovete disintegrare l'insetto che si avvicina a voi in maniera impressionante.
Fate attenzione, perché ogni volta che lo colpirete si dividerà ulteriormente, moltiplicandosi.

SISTEMA SOLARE

C 16 e PLUS 4 - Joystick in porta 2 - Tastiera

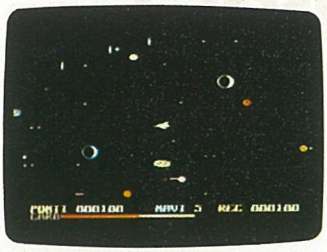
TASTI:
P - Pausa
Spazio - Fuoco
Tasto commodore - Sinistra
Shift - Destra
J - alto
/ - basso

IN TUTTE LE EDICOLE D'ITALIA

Estasy Rosa

I NUOVI ROMANZI D'AMORE
CHE RACCONTANO *TUTTO*

Estasy Rosa

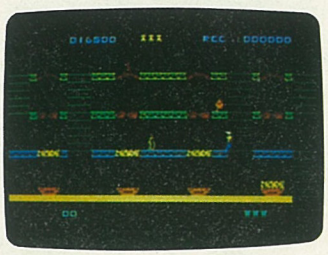


Distruggi i radar nemici se non vuoi essere intercettato e abbattuto.
Vola con la tua astronave interplanetare nello spazio facendo attenzione alle meteoriti che ti verranno contro.
Quando finisci il carburante, rifornisciti all'astronave madre e riparti per una nuova missione.
Buon divertimento!

IL CUOCO

C 16 e PLUS 4 - Tastiera

TASTI:
F1 - Partenza gioco
J - Sinistra
. - Destra
A - Alto
Z - Basso

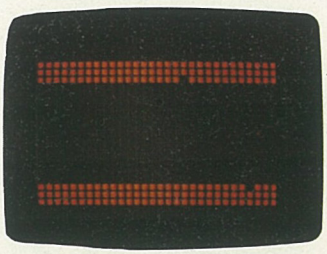


Il tuo nome è Camillo il cuoco ed il tuo compito è quello di preparare alcuni panini. Ad ostacolare il tuo lavoro vi sono alcuni würstel e alcune cipolle che, se ti prenderanno, ti uccideranno.
L'unica tua arma di difesa è il pepe. Lanciando il pepe rimarrai immune dagli attacchi nemici per circa un minuto.
Prepara i panini più in fretta che puoi se non vuoi che il tuo fast-food fallisca.

THE WALL

C 16 e PLUS 4 - Tastiera

TASTI:
Spazio - Per partire
£ - Sinistra
= - Destra

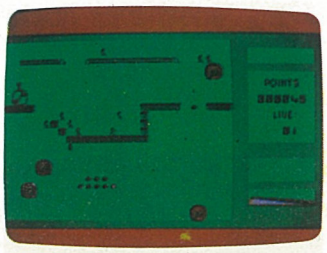


Lo scopo di questo gioco è quello di abbattere la muraglia mattone per mattone per poter evadere.
Buona fotuna!!!

VIETNAM

C 16 e PLUS 4 - Joystick in porta 1

A Joe Rambo è stata affidata una impossibile missione suicida e solo lui è in grado di portarla a termine tornando vivo dal territorio nemico alla base di partenza.



Sul vostro cammino troverete insidie di ogni tipo: dai guardiani della rivoluzione ai soldati sovietici, per non parlare poi dei campi minati e delle torrette mitragliatrici.
Good luck!!!

RAID

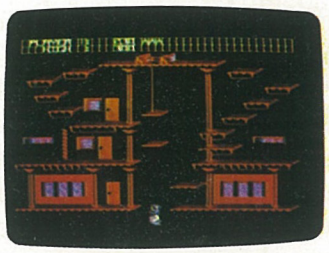
CBM 64 e 128 - Joystick in porta 2

Alla guida della vostra astronave dovete cimentarvi in un movimentato ed entusiasmante RAID.
Moltissimi saranno i vostri nemici da combattere e altrettanti gli obbiettivi da bombardare per aumentare il vostro punteggio. Vi potete servire di cinque tipi di astronave che potrete selezionare con il movimento del joystick prima di iniziare l'attacco. L'autonomia della vostra astronave è condizionata dal numero di serbatoi contrassegnati con la lettera "F" che riuscirete a bombardare.

Alla fine di ogni gioco, quando comparirà la richiesta di scrivere il vostro nome per il punteggio, se digiterete la parola "demo" otterrete una suggestiva rappresentazione.
Per resettare il gioco in qualsiasi momento, basterà premere il tasto restore.

MR. TOM

CBM 64 e 128 - Joystick in porta 1



Misurate la vostra abilità guidando Mr. Tom nell'ispezione della sua industria e grandi magazzini annessi.
Per passare da un reparto all'altro, si dovrà raccogliere una chiave che si troverà in zone prestabilite, dirigendosi poi verso le piattaforme contrassegnate con "exit".
Troverete, naturalmente, moltissime difficoltà: per difendervi potrete sparare ai vostri nemici.
Il gioco consente la partecipazione di uno o due giocatori.

LANDING

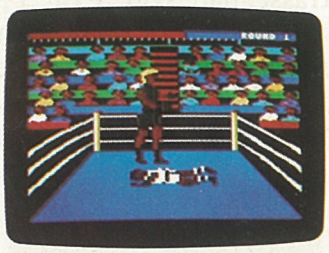
CBM 64 e 128 - Joystick in porta 1

Un suggestivo paesaggio vi si presenterà avvicinandovi con la vostra astronave ad uno dei tanti pianeti sconosciuti della galassia che state attraversando.

Servendovi del modulo d'atterraggio, potrete esplorare i vari pianeti. Dovrete guidare il modulo dosando il getto dei razzi direzionali ed introdurvi nelle cavità sotterranee cercando di non toccare contro le pareti; pena: la distruzione del modulo. Quando avrete effettuato l'atterraggio morbido sull'apposita piattaforma, dovrete rifare il percorso inverso per tornare all'astronave e proseguire il viaggio verso altri pianeti.

MEGABOX

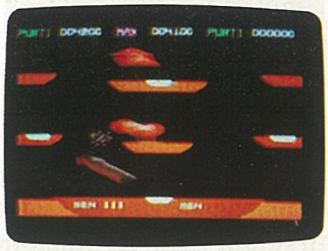
CBM 64 e 128 - Joystick in porta 1



Questo bellissimo gioco rappresenta un incontro di box, al quale potrete partecipare giocando contro un campione guidato dal computer oppure contro un secondo partecipante.
La selezione delle opzioni si effettua spingendo la leva del joystick in avanti o indietro e premendo il pulsante per confermare la scelta.
Buon combattimento.

WARRIORS

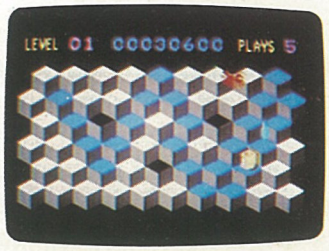
CBM 64 e 128 - Joystick in porta 2



Siete un valoroso guerriero che dovrà combattere contro numerosi ed acerrimi nemici. Molte saranno le difficoltà che, una volta superate, vi porteranno nelle fasi successive del gioco.
Per guidare il vostro guerriero, basterà muovere la leva del joystick nel senso desiderato; per sollevarsi basta premere ripetutamente il pulsante.
Il gioco vi consente di partecipare da soli contro il computer oppure in due giocatori.
Comandi:
F1 - Seleziona il numero dei giocatori
F3 - Seleziona il livello del gioco
F5 - Joystick/Tastiera primo giocatore
F7 - Joystick/Tastiera secondo giocatore

MOLLA

CBM 64 e 128 - Joystick in porta 2



Guidate la molla salterina sui cubi facendo cambiare il colore delle varie piattaforme e, naturalmente, evitando i numerosi ostacoli che cercheranno di rendervi difficile il compito.
Quando tutte le basi avranno cambiato colore, si passerà alla fase successiva che naturalmente riserverà maggiori difficoltà. Portandovi nelle basi lampeggianti, otterrete dei bonus necessari alla vostra autonomia di gioco.

HAMMER

CBM 64 e 128 - Joystick in porta 1

Dimostrate la vostra abilità usando un pesante martello per picchiare sulle tavole che formano la griglia del gioco.

Spostandovi da una tavola all'altra, queste cambieranno colore. Per completare il gioco e poter passare alla fase successiva, dovrete far cambiare di colore tutte le tavole, naturalmente evitando i vostri nemici.
Buona fortuna!

NON PERDETE IL PROSSIMO NUMERO DI GO GAMES

BUON LAVORO!!

impariamo a programmare in assembler 10

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

Inizio questo mese riproponendovi il diagramma a blocchi della scorsa volta per una questione di comodità di lettura, quindi vi sottopongo il relativo programma completo scritto in assembler.
Come ho avuto occasione di dire nelle prime puntate, il programma di questo numero è la traduzione del diagramma di flusso. Ciascun blocco del diagramma di flusso è stato tradotto in una o più istruzioni. Si è presupposto che MPR e MPD avessero già un valore.
Guardando il diagramma di flusso vediamo che il primo blocco è un blocco di "inizializzazione" dove è necessario porre a "Ø" un certo numero di registri o locazioni di memoria poiché verranno utilizzati dal programma. I registri utilizzati dal programma della moltiplicazione sono visibili in fig. 1.
Sulla sinistra dell'illustrazione appare la porzione che riguarda il microprocessore 6510 mentre sulla destra la sezione attinente alla memoria. Si è stabilito che gli indirizzi di memoria aumentino dall'alto al basso dell'illustrazione.
Il registro X, sul lato sinistro (uno dei due registri indice del 6510) verrà utilizzato come "contatore" cosa necessaria in quanto si sta eseguendo una moltiplicazione ad 8 bit e di conseguenza si devono verificare gli 8 bit del moltiplicatore.
Per nostra sfortuna il 6510 non è dotato di un'istruzione che ci consenta di provare detti bit in sequenza, si possono testare conve-

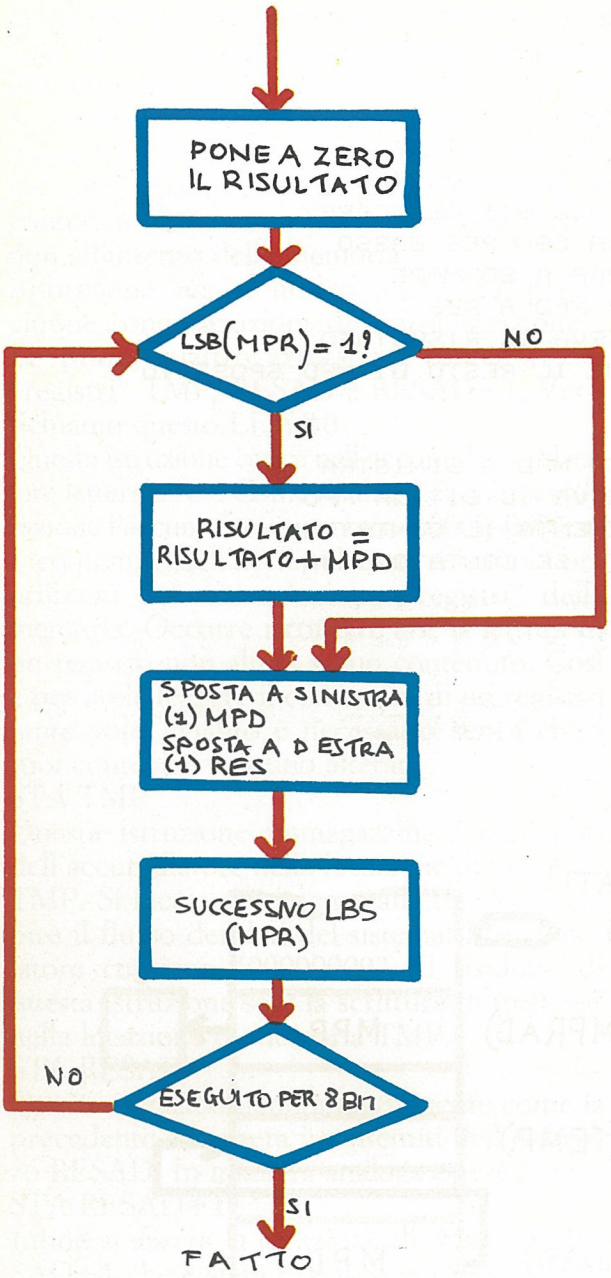


diagramma a blocchi

nientemente solo i flag del registro di stato. Questa limitazione, peraltro comune alla maggior parte dei microprocessori, ci costringerà, per verificare successivamente tutti i bit del moltiplicatore, trasferire il valore del moltiplicatore nell'accumulatore, quindi i contenuti dell'accumulatore saranno fatti scorrere a destra.

Le istruzioni di scorrimento muovono ogni bit del registro di una posizione a destra o a sinistra. Il risultato di un'operazione di scorrimento la possiamo vedere in fig. 2. Le varianti possibili sono molte in dipendenza del bit che entra nel registro, a questo penseremo un pochino più avanti.

Torniamo alle successive verifiche di ciascuno degli 8 bit del moltiplicatore. Poiché è frequente il verificarsi del bit carry, il moltiplicatore sarà spostato di una posizione 8 volte. Ogni volta il suo bit più a destra cadrà nel bit carry e sarà verificato. Il problema che si verrà subito dopo a creare è che il prodotto parziale che sarà accumulato durante le addizioni successive richiederà 16 bit, quindi diremo che la moltiplicazione di due numeri ad 8 bit può produrre un risultato a 16 bit.
La cosa è abbastanza chiara in quanto $2\uparrow 8 = 2\uparrow 16$, per questo risultato è necessario riservare 16 bit.
Un'altra limitazione del 6510 è il numero limitato di registri interni di conseguenza il prodotto parziale non potremo memorizzarlo all'interno del 6510, il moltiplicatore, il moltiplicando e i prodotti parziali li porremo in memoria.
Questa operazione genera inevitabilmente un'esecuzione lenta o perlomento più lenta di quella ottenibile memorizzando tutto all'interno del microprocessore.
In fig. 1 nella parte alta possiamo vedere la parola di memoria allocata per il moltiplicatore. Si assumerà, per esempio, che esso contenga "3" in binario. L'indirizzo di questa locazione di memoria è MPRAD. Sotto a questo si trova un "temporaneo" il cui indirizzo è TMP. Il ruolo di questa locazione lo chiarirò in seguito. Si sposterà il moltiplicando a sinistra nella locazione principale aggiungendolo al prodotto parziale. Il moltiplicando è successivo e si assumerà contenente il valore di "5" in binario. Il suo indirizzo è MPDAD.
Per finire, in fondo alla memoria, si trovano due parole allocate per il prodotto parziale ovvero il risultato. Il loro indirizzo è RESAD.
Queste locazioni di memoria saranno i registri di lavoro e la parola "registro" può esse-

```
        LDA   #0        AZZERA L'ACCUMULATORE
        STA   TMP       AZZERA QUESTO INDIRIZZO
        STA   RESAD     AZZERA
        STA   RESAD+1   AZZERA
        LDX   #8        X E' IL CONTATORE
MULT    LSR   MPRAD     SPOSTA MPR A DESTRA
        BCC   NO ADD    TEST DEL BIT DEL CARRY
        LDA   RESAD     CARICA CON RES BASSO
        CLC             PREPARA A SOMMARE
        ADC   MPDAD     SOMMA MPD A RES
        STA   RESAD     CONSERVA IL RISULTATO
        LDA   RESAD+1   SOMMMA IL RESTO DI MPD SPOSTATO
        ADC   TMP
        STA   RESAD+1
NOADD   ASL   MPDAD     SPOSTA MPD A SINISTRA
        ROL   TMP       CONSERVA IL BIT DA MPD
        DEX             DECREMENTA IL CONTATORE
        BNE   MULT      RIPETI SE CONTATORE #0
```

listato 1

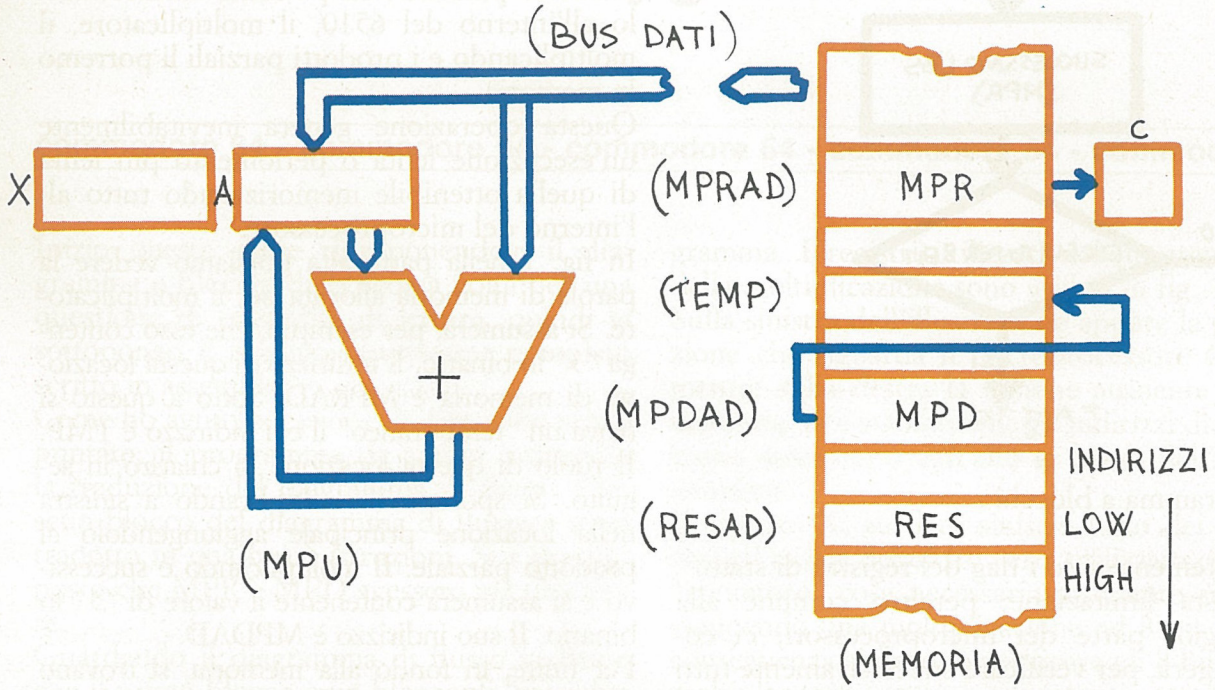


Fig. 1

re utilizzata come sinonimo di "locazione" in questo particolare contesto.
Con la freccia che appare in alto a destra dell'illustrazione che fa entrare MPR nel bit C ho voluto rappresentare simbolicamente come il moltiplicatore sia fatto scorrere nel bit carry, naturalmente questo bit carry è contenuto fisicamente all'interno del 6510 e non all'interno della memoria.
Ritorniamo ora al nostro listato. Le prime cinque sono istruzioni di "inizializzazione". Le prime quattro azzerano i contenuti dei "registri" TMP, RESAD e RESAD+1. Verifichiamo questo LDA#0
Questa istruzione carica nell'accumulatore il valore letterale "0". Come risultato di questa operazione l'accumulatore conterrà "00000000".
I contenuti dell'accumulatore verranno ora utilizzati per azzerare i tre "registri" della memoria. Occorre ricordare che la lettura di un registro non altera il suo contenuto. Così è possibile leggere il contenuto di un registro tante volte quanto è necessario senza che i suoi contenuti vengano alterati.
STA TMP
Questa istruzione immagazzina i contenuti dell'accumulatore nella locazione di memoria TMP. Si faccia riferimento alla fig. 1 per capire il flusso dei dati del sistema. L'accumulatore contiene "00000000". Il risultato di questa istruzione sarà la scrittura di tutti zeri nella locazione di memoria TMP.
STA RESAD
Questa istruzione opera esattamente come la precedente ed azzera i contenuti dell'indirizzo RESAD. In maniera analoga opera:
STA RESAD+1
Infine si azzera la locazione di memoria RESAD+1 che è stata riservata per memorizzare la parte alta del risultato. (La parte alta sono i bit 8-15; la parte bassa sono i bit 0-7).
Per arrestare lo scorrimento dei bit del moltiplicatore all'istante corretto, è necessario contare il numero di scorrimenti che sono stati eseguiti. Sono necessari 8 scorrimenti. Il registro X sarà utilizzato come contatore ed inizializzato al valore di "8". Ogni volta che viene eseguito uno scorrimento, i contenuti di questo contatore saranno decrementati di 1. Quando il valore diventa 0 la moltiplicazione è terminata.
LDX#8
Questa istruzione inizializza il registro x ad "8", pone, cioè, il numero "8" nel registro X.
Con riferimento al diagramma a blocchi si deve verificare il bit meno significativo del moltiplicatore. È stato indicato precedentemente che questa prova non può essere eseguita in una singola istruzione. Occorre utilizzare due istruzioni. Prima il moltiplicatore sarà fatto scorrere a destra, poi il bit che esce sarà verificato. Questo è il bit carry.
LSR MPRAD
Questa istruzione è uno spostamento logico a destra dei contenuti della locazione di memoria MPRAD.
Ora dobbiamo verificare il valore del bit carry con:
BCC NOADD
Questa istruzione significa "Se il Carry è a Zero" vai all'indirizzo NOADD.
Questa è la prima istruzione di diramazione che incontriamo. Tutti i programmi fino ad ora visti erano sequenziali. Per capire l'importanza dei test logici, come quello del bit carry, si deve essere in grado di eseguire istruzioni dovunque nel programma dopo il test. Le istruzioni di diramazione eseguono appunto tale funzione. Si esaminerà il valore del bit carry. Se il carry era "0", cioè azzerato, allora l'esecuzione del programma proseguirà all'indirizzo NOADD. Questo significa che la successiva istruzione eseguita dopo BCC sarà l'istruzione all'indirizzo NOADD, se il test è soddisfatto.
Altrimenti, se il test non è soddisfatto, non si verificherà alcuna diramazione e sarà eseguita l'istruzione sequenziale successiva BCC NOADD.
NOADD necessita di una spiegazione ulteriore: dire NOADD equivale a dire un numero effettivo di memoria. In poche parole NOADD è una "label simbolica".
Essa rappresenta fisicamente un indirizzo effettivo all'interno della memoria. Il programmatore, per evidente convenienza, utilizza nomi simbolici al posto di indirizzi fisici, il programma assemblatore, che accetta questa tecnica, al momento della compilazione, sostituirà le label simboliche con gli effettivi indirizzi di memoria.

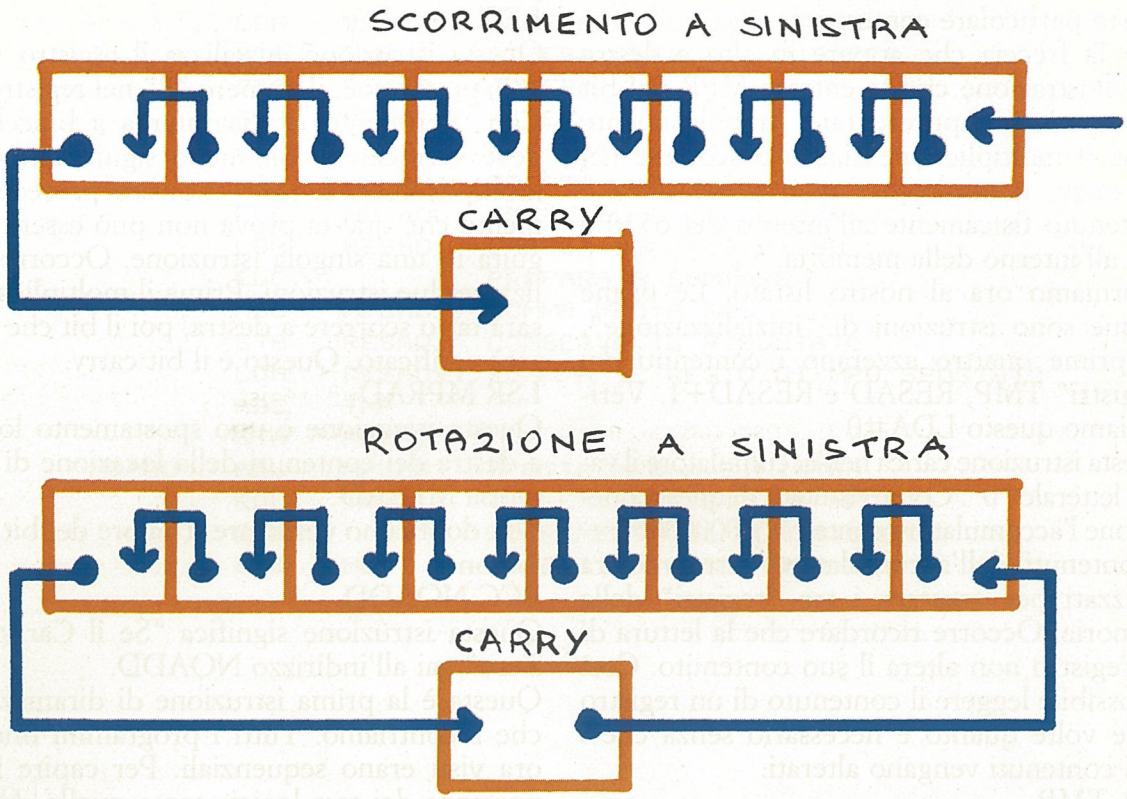


Fig. 2

Questa possibilità unisce alla evidente maggior facilità di programmazione anche una migliore leggibilità del programma sorgente e consente al programmatore di inserire istruzioni addizionali tra il punto di diramazione e la label in questione senza la necessità di dover riscrivere ogni cosa.
Torniamo al nostro programma, se il test sul Carry non è soddisfatto viene eseguita l'istruzione sequenzialmente successiva nel programma.
Esaminiamo ora entrambe le alternative.
ALTERNATIVA 1: il Carry è a "1"
In questo caso il test non è soddisfatto e viene eseguita:
LDA RESAD
ALTERNATIVA 2: il carry è a "0"
In questo caso il test è soddisfatto e viene eseguita l'istruzione con la label NOADD.
Riferendoci al nostro diagramma di flusso, si specifica che se il bit carry è a 1, il moltiplicando deve essere sommato al prodotto parziale (nel nostro caso i registri RES). Inoltre va eseguito uno scorrimento. Il prodotto parziale deve essere mosso di una posizione a destra ovvero il moltiplicando deve essere mosso di una posizione a sinistra. Si adotterà qui la convenzione normalmente impiegata nell'esecuzione manuale della moltiplicazione e si muoverà il moltiplicando di una posizione a sinistra.
Il moltiplicando è contenuto nei registri TMP ed MPDAD. (Per semplicità saranno normalmente chiamate "registri" le locazioni di memoria). I 16 bit del prodotto parziale sono contenuti agli indirizzi di memoria RESAD e RESAD+1.

Per spiegare questo si assuma che il moltiplicando sia "5". I vari registri li vediamo in fig. 1.

Si devono semplicemente sommare due numeri a 16 bit. Questo problema lo abbiamo già affrontato e risolto (vedi l'addizione a 16 bit). Si sommeranno prima i byte di basso ordine e poi quelli di ordine elevato in que-

sto modo:
LDA RESAD
Carichiamo l'accumulatore con la parte bassa di RES
CLC
Prima di qualsiasi operazione il nostro 6510 richiede che il bit carry sia azzerato. In questo caso è particolarmente importante perché si sa che il bit carry era stato posto ad 1.
ADC MPDAD
Il moltiplicando è sommato all'accumulatore, che contiene la parte bassa di RES.
STA RESAD
Il risultato dell'addizione è conservato all'appropriata locazione di memoria (RES) BASSO. Viene poi eseguita la seconda metà dell'addizione. Durante l'esecuzione del controllo manuale di questo programma non si dimentichi che l'addizione porrà il bit carry. Il carry sarà posto a "0" o ad "1" in dipendenza del risultato dell'addizione. Qualsiasi riporto che deve essere generato sarà portato automaticamente nella parte di ordine elevato del risultato.
Si completa ora l'addizione:

```
LDA  RESAD+1
ADC  TMP
STA  RESAD+1
```

Queste istruzioni completano l'addizione a 16 bit. Si è ora sommato il moltiplicando a RES. Occorre ora spostarlo di una posizione a sinistra prima dell'addizione successiva. Si può anche considerare lo spostamento del moltiplicando di una posizione a sinistra "prima" dell'addizione, eccetto che per la prima volta. Questa è una delle molte scelte di programmazione sempre aperte al programmatore.
Si faccia scorrere il moltiplicando a sinistra:
NOADD ASL MPDAD
Questa istruzione è uno "Spostamento Aritmetico a Sinistra".
Essa sposterà di una posizione a sinistra i contenuti della locazione di memoria MPDAD che contiene la parte bassa del moltiplicando. Questo non basta. Non ci si può permettere di perdere il bit che cade dalla parte estrema sinistra del moltiplicando. Questo bit cadrà nel bit carry. Esso qui non può essere immagazzinato permanentemente perché poi può essere distrutto da qualsiasi operazione aritmetica. Questo dovrebbe essere conservato in un registro "permanente". Esso dovrebbe essere fatto scorrere nella locazione di memoria TMP. Questo è infatti realizzato dall'istruzione successiva:
ROL TMP
Questa istruzione realizza una "Rotazione a sinistra" dei contenuti di TMP.
A questo punto si può fare un'interessante osservazione. Sono stati appena utilizzati due diversi tipi di istruzioni di scorrimento per fare scorrere un registro di una posizione a sinistra. La prima è ASL e la seconda è ROL; vediamo la differenza.
L'istruzione ASL fa scorrere i contenuti del registro mentre ROL è un'istruzione di rotazione. Essa sposta i contenuti del registro di una posizione a sinistra ed il bit che cade dall'estremità sinistra va nel bit carry, come al solito. La differenza sta nel fatto che i "contenuti precedenti del bit carry sono forzati nella posizione più a destra". Questa in matematica è chiamata rotazione circolare (rotazione a 9 bit). Questo è esattamente quello che si vuole come risultato della ROL, il bit spinto fuori da TMP sulla sinistra e preservato nel bit carry C arriverà nella posizione più a destra del registro TMP così come si voleva.
Si è così terminato con la parte aritmetica di questo programma. Si dovrà verificare se l'operazione è stata eseguita 8 volte, cioè se la moltiplicazione è terminata. Normalmente nella stragrande maggioranza dei microprocessori questo richiede due istruzioni:
DEX
che decrementa i contenuti del registro X. Se prima conteneva "8" ora contiene "7".
BNE MULT
Questa è un'altra istruzione di verifica e diramazione. Essa specifica "salta alla locazione MULT se il risultato non è uguale a 0". Finché il registro contatore decrementa ad un intero non zero, c'è un salto automatico indiero alla label MULT. Questo è chiamato ciclo di moltiplicazione. Con riferimento al nostro diagramma di flusso questo corrisponde alla freccia che esce dall'ultimo blocco. Questo ciclo sarà eseguito 8 volte.

Nella maggior parte dei casi il programma appena visto costituirà una subroutine e l'istruzione finale di una subroutine sarà RST. Il meccanismo delle subroutine lo vedremo in una prossima puntata, per ora sorbitevi quanto detto in questo mese e poi si vedrà. Arrivederci a tutti e non dimenticate di non mancare il mese prossimo, ciao.

impariamo a usare il computer 11

Iniziamo quest'oggi a parlare di un nuovo ed affascinante argomento sul quale ci soffermeremo lungamente: il linguaggio macchina. Non vi siete mai chiesti ad esempio come vengono costruiti i nostri fantastici videogiochi, da dove viene tanta velocità ed accuratezza nei dettagli in campo grafico e sonoro soprattutto? Per esprimervi a livello introduttivo questo nuovo concetto, considerate come suo sottotitolo qualcosa di simile a "oltre le frontiere del BASIC". Questo non ci dice molto dal punto di vista tecnico, ma per il momento ci pare una caratterizzazione sufficientemente indicativa.

Vediamo di definire a grandi linee che cosa si intende per "linguaggio macchina". Generalmente, possiamo considerarlo come la forma di espressione attraverso la quale è possibile comunicare direttamente con il calcolatore, senza bisogno di passare attraverso dispositivi di traduzione. Per farvi un parallelo con la realtà quotidiana, immaginatevi un dialogo fra due persone straniere. Nel caso entrambi abbiano la facoltà di parlare e comprendere la medesima lingua, potranno scambiarsi in un tempo determinato una quantità superiore d'informazione che non passando forzatamente attraverso una procedura intermedia di traduzione, anche se simultanea. Da un'altro punto di vista, una particolare quantità costante d'informazione viene trasmessa e ricevuta in un tempo minore, quindi ad una velocità superiore. Questo spiega la principale peculiarità del linguaggio macchina rispetto agli altri linguaggi: la rapidità esecutiva.

Per scendere maggiormente nei dettagli, non possiamo non criticare il suo nome. È infatti improprio considerarlo un linguaggio alla stessa stregua del BASIC, tanto per fare un esempio. In effetti, le forme di comunicazione che, per riprendere la precedente caratterizzazione, necessitano di una traduzione intermedia si esprimono in maniera letterale e sono dunque molto più vicine all'uomo. Il linguaggio macchina, per contro, è totalmente numerico, e pertanto direttamente comprensibile dal calcolatore. Sarebbe più esatto allora parlare di "codice macchina", riservando il termine "linguaggio" per identificare forme di espressione letterale, appositamente concepite allo scopo di rendere più agevole all'uomo la manipolazione dell'informazione (e contemporaneamente più difficoltosa per il calcolatore in quanto esso deve dipendere da un intermediario esterno come ad esempio l'interprete BASIC)

A livello pratico, programmare in codice macchina consiste idealmente nel memorizzare il valore giusto al posto giusto. La teoria è affascinante, ma ragioniamo in termini applicativi. Se in linguaggio macchina si facilita la comprensione al computer, guadagnando così in rapidità esecutiva, non altrettanto si può dire nei riguardi del programmatore umano. Considerate la difficoltà oggettiva di concepire un programma in numeri, od anche più semplicemente un concetto, un'idea.

E che dire allora della sua manipolazione? Se a programmare in codice macchina il nostro computer fosse un altro computer, il problema non sussisterebbe; ad ogni modo, in origine tutto è comunque partito dall'uomo. È dunque necessario alterare la forma del codice macchina, preservandone contemporaneamente le caratteristiche peculiari. In pratica, abbiamo bisogno di qualcosa di assolutamente corrispondente al codice numerico, ma facilmente gestibile da parte dell'uomo, e pertanto esprimibile alfabeticamente. Questo qualcosa è naturalmente già stato inventato, e si chiama "linguaggio assembler" (il termine linguaggio è in questo caso perfettamente appropriato).

Il linguaggio assembler altro non è che l'immagine umana del codice macchina. Ogni istruzione assembler equivale infatti ad una ed una sola istruzione in codice macchina; si dice che le due istruzioni in questione siano in corrispondenza biunivoca. I due insiemi sono invece isomorfi. L'unica distinzione riguarda solo ed unicamente l'immagine, la forma. Il medesimo concetto viene trattato numericamente dal calcolatore ed alfabeticamente dall'operatore umano.

Il codice macchina (in senso concettuale, non formale) rappresenta l'unico possibile anello di congiunzione fra uomo e computer. I due interlocutori, sebbene di natura completamente diversa, dispongono di un punto di contatto attraverso il quale possono comunicare liberamente, senza avere bisogno d'intermediari. L'unica limitazione risiede purtroppo nella forma: non è infatti possibile unificarla, ma il problema è molto relativo.

Da questo momento, in quanto uomo che parla ad altri uomini, mi limiterò a condiserare unicamente la forma letterale del codice macchina: il linguaggio assembler.

Il codice macchina è identico per tutti i calcolatori, ma variabile in funzione del loro cuore, il microprocessore. I computers appartenenti alla medesima famiglia, basati sul medesimo microprocessore, dispongono dell'identico codice macchina. Noi ci limiteremo a considerare soltanto quello relativo al microprocessore 6502, adottato da moltissimi micro-computers come APPLE II, II+ e IIe, ATARI 400 e 800, nonché i nostri cari COMMODORE VIC 20 e 64, e tutti i vari PET. Il modello che prenderemo come riferimento sarà il COMMODORE 64, del quale abbiamo ogni volta a disposizione meravigliose dimostrazioni delle sue capacità sulla cassetta allegata alla rivista. In poche parole, il nostro scopo consiste principalmente nel farvi giocare e divertire, cercando contemporaneamente di stimolarvi a conoscere quanto state utilizzando in quel momento.

In questo nostro primo approccio con il nuovo argomento eviteremo di addentrarci subito in particolari tecnici per non rischiare di confondervi inutilmente le idee. È molto importante comprendere a fondo la nuova filosofia di programmazione prima ancora di saperla gestire in termini di applicazione pratica. Cerchiamo ora di definire in maniera progressivamente più dettagliata la personalità del codice macchina, prendendo il BASIC come termine di paragone.

Uno dei suoi principali vantaggi consiste nel permettere al programmatore l'effettuazione di determinate funzioni per le quali il BASIC non è indicato. La sua più appariscente caratteristica è tuttavia rappresentata dalla notevole rapidità esecutiva. Nel COMMODORE 64 è infatti possibile eseguire approssimativamente centomila istruzioni al secondo. I comandi BASIC sono svariate centinaia di volte più lenti.

Ciò è dovuto al fatto che il BASIC è scritto a sua volta in codice macchina, e pertanto ogni suo comando può essere considerato un vero e proprio programma composto anche da alcune centinaia d'istruzioni. Questa caratteristica si riflette in tutti i linguaggi strutturati.

Le istruzioni del codice macchina sono estremamente limitate per quanto riguarda la funzione assolta. Esse svolgono appunto compiti elementari, per cui è necessario combinarne molteplici al fine di ottenere un risultato tangibile. Le loro funzioni si riferiscono direttamente all'effettiva architettura interna del computer. Esse provvedono fra le altre cose a segnalare al calcolatore quali numeri ricordare e quali dimenticare, determinare se un tasto è stato premuto, leggere e scrivere in-

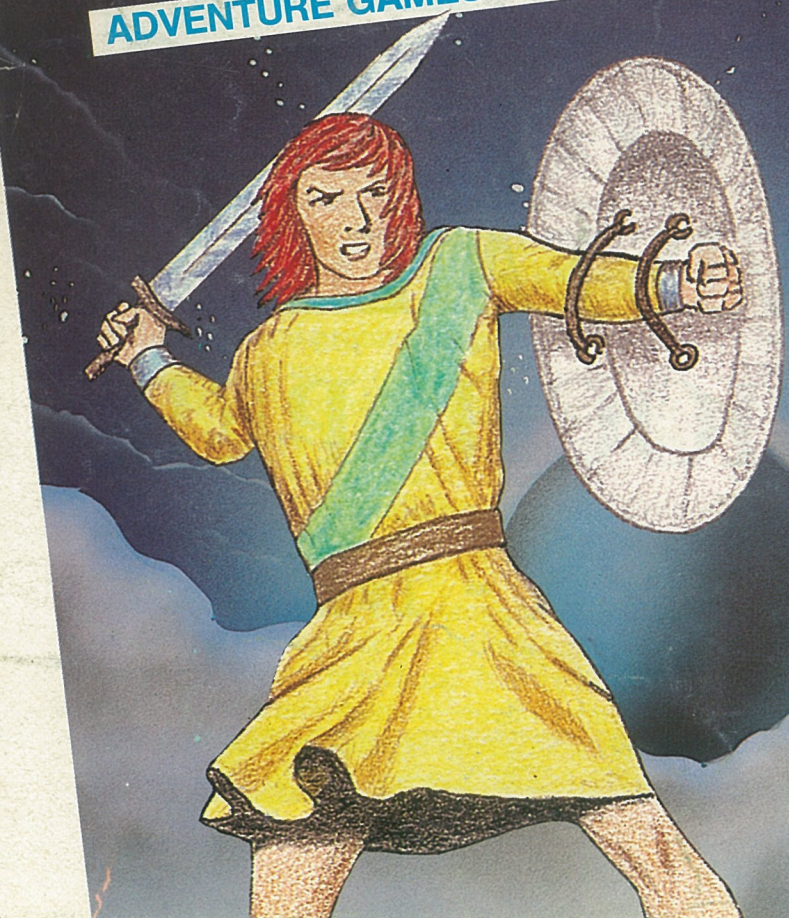
formazione su disco o nastro, stampare caratteri sullo schermo, ed innumerevoli altri compiti.
I programmi in codice macchina possono essere inoltre considerati sotto forma di subroutines, esattamente come in BASIC, ovvero un programma nel programma richiamabile in qualunque momento ed al termine del quale il controllo viene restituito al comando immediatamente successivo alla chiamata.
Per richiamare una subroutine scritta in codice macchina a partire da un programma BASIC utilizzeremo il comando SYS (indirizzo). Analogamente a GOSUB, è necessario segnalare al computer il punto di partenza della subroutine; così come GOSUB 5000 richiama la subroutine BASIC avente inizio alla linea di programma 5000, SYS 5000 richiamerà la subroutine in codice macchina localizzata a partire dall'indirizzo decimale di memoria 5000. Ovviamente, vi invitiamo a non confondere un indirizzo di memoria con un numero di linea BASIC.
La prossima volta continueremo il nostro discorso andando a studiare più da vicino l'impiego del codice macchina relativamente alla struttura interna della memoria del nostro COMMODORE 64.

ADVENTURE EPIC 3000 GAMES

IN ITALIANO

N° 4
SETTEMBRE 1986
L. 8.000

ADVENTURE GAMES PER C.64/128 E SPECTRUM 48K



Commodore 64 e 128

JACK BYTESON
Duello con Dracula
(AVVENTUROSO)

DUST HANTER
Level Nine
(POLIZIESCO)

KALABUR (play role)
(FANTASY)

Spectrum 48K

HERO
Nel campo nemico
(GUERRA)

FANTASY
La vallata magica
(FIABESCO)

VIRUS DELTA
Dov'è Mark Williams?
(POLIZIESCO)