Scuola di videogame
Moltiplicazione degli sprite

Eccoci di ritorno dopo il salto del numero di marzo. Questo mese vorrei cominciare a parlare di LM e quindi distruggere quell'orribile maldicenza che lo classifica come il più complicato e, al tempo stesso, stupido dei linguaggi. In questa nostra spiegazione cercheremo, per quanto possibile, di fare analogie con il Basic, che, a quanto pare, è il linguaggio che conoscete un po' tutti (a mio avviso, solo perché fornito in dotazione con la macchina, con tanto di «istruzioni per l'uso»)

Nella sua accezione più classica, il linguaggio macchina è il vero e unico linguaggio che il computer capisce in modo diretto, cioè senza l'aiuto di un interprete. Vi ricordo che il sistema operativo non è l'interprete del linguaggio macchina, ma solo una serie di subroutine che ne facilitano l'uso. Tutte quelle che vi dirò non saranno parole buttate al vento, ma serviranno per la comprensione del listato proposto che, come si intuisce dal titolo, serve ad aumentare gli sprite disponibili sul C64.

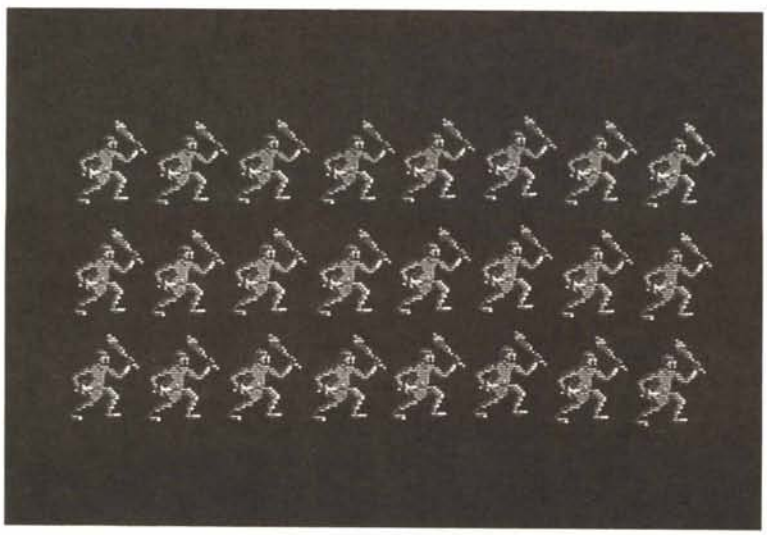
Piccola premessa: seguendo la filosofia di questa rubrica, tenterò di non annoiare nessuno e quindi di non dilungarmi troppo sulle cose semplici e, nello stesso tempo, di non parlare «troppo difficile».

Dunque, come molti di voi già sanno, il C64 è un computer dotato di:

64 KByte di RAM
20 KByte di ROM
4 KByte di I/O.

Il microprocessore è il circuito centrale che praticamente comanda tutto il resto. Per accedere agli altri circuiti (generatore del suono, generatore della grafica etc.) deve in qualche modo essere collegato con questi. Detto collegamento avviene tramite i 4K di I/O (Input/Output). Per comprendere quanto detto occorre prima dare un'occhiata al funzionamento del microprocessore stesso. Questi è in grado di «indirizzare» 64 KByte di locazioni di memoria, cioè può accedere (in scrittura o in lettura) a un numero limitato di byte. Consideriamo quello che succede all'accensione del Commodore 64; ci ritroviamo in uno stato di pieno disordine. Il microprocessore ha il compito di riordinare tutto. Partendo da una locazione di memoria forzata, comincia la sua opera, «prendendone» il contenuto. Questo vuol dire che un byte della ROM, che contiene il sistema operativo (S.O.), viene memorizzato all'interno del microprocessore. Cosa se ne fa di questo byte? Semplice: lo traduce in una istruzione da eseguire. A seconda del tipo di istruzione il microprocessore può prendere anche i successivi byte (uno o due) e considerarli parte integrante della stessa. Questo perché un'istruzione non è sempre definita da un solo byte. In ogni caso i successivi (sempre uno o due) byte non sono necessari per la decodifica dell'istruzione, ma servono, ad esempio per fornire dei dati da utilizzare unitamente con essa.

Supponiamo quindi che la nostra prima istruzione è stata interpretata e che gli eventuali dati siano stati prelevati dalla ROM. Il microprocessore, a questo punto, esegue l'istruzione. Questa si identifica normalmente in un'operazione che modifica lo stato interno del microprocessore o che modifica una locazione di memoria. Visto che siamo alla partenza e che quindi dobbiamo riordinare lo stato di tutti i circuiti del computer, questa prima ed altre saranno adibite a tale scopo. Quindi il microprocessore continuerà con il prelievo delle istruzioni (e con la relativa esecuzione) fin quando tutto sarà rimesso in ordine. Coma fa il microprocessore a comandare i circuiti che lo circondano? Anche qui la risposta è molto semplice. I 4 KByte di I/O, che avevamo descritto prima, non sono altro che delle locazioni



I 24 sprite.

di memoria «RAM» interne a detti circuiti (quindi non 4K di RAM normale). Scrivendo dentro questa particolare RAM il microprocessore automaticamente impone un comando a questi benedetti circuiti che reagiscono di conseguenza. I quattro kappa di RAM sono posizionati a partire dalla locazione 53248 (D000 in esadecimale). A questo punto è d'obbligo porsi la seguente domanda: «ma se il microprocessore può indirizzare solo 64 KByte, come fanno ad esserci 64K di RAM, 20K di ROM e 4K di I/O?». Mi auguro che detta domanda ve l'eravate già posta da molto prima. La risposta comunque arriva solo adesso. Con opportuni artifici, il C64 è in grado di scegliere se indirizzare o no una parte di RAM o di ROM quindi l'input/output. Deve cioè fare in modo che il totale della «roba» da indirizzare sia di 64 KByte. Tutto qui. Normalmente il nostro Commodore predispone:

44 KByte di RAM
16 KByte di ROM
4 KByte di I/O.

Le ROM comprendono, come noto, il sistema operativo e il Basic, mentre i 4K «scartati» contengono la forma dei 512 caratteri grafici disponibili sul 64. Dato che non è il microprocessore quello che vuole saperne il contenuto (si tratta invece del VIC-II, Video-Chip-Interface), sono eliminati e ci permettono perciò di usare altri 4K di RAM. Intendiamoci, in qualunque momento la mappa di memoria accessibile può essere alterata (volontariamente) quindi non c'è assolutamente da preoccuparsi per la restante RAM non utilizzata. In tal caso, però, il prezzo da pagare è la momentanea impossibilità di accedere alla ROM e all'input/output fino a configurazione originale ripristinata. Confusi? Vi concedo 5 minuti d'aria... poi riprendremo di corsa...

...Come vola il tempo eh?

Bene, chiarite queste noiose, ma non trascurabili, cosette, possiamo andare avanti con la lezione di linguaggio macchina, descrivendo le caratteristiche del li-

```
  PC   IRQ   SR AC XR YR SP
.E37B EA31 A4 F1 30 11 DC
.7000 78        SEI
.7001 A9 7F     LDA #$7F
.7003 8D 0D DC  STA $DC0D
.7006 A9 30     LDA #$30
.7008 A2 70     LDX #$70
.700A 8D 14 03  STA $0314
.700D 8E 15 03  STX $0315
.7010 EA        NOP
.7011 A9 00     LDA #$00
.7013 8D 12 D0  STA $D012
.7016 AD 11 D0  LDA $D011
.7019 29 7F     AND #$7F
.701B 8D 11 D0  STA $D011
.701E A9 01     LDA #$01
.7020 8D 1A D0  STA $D01A
.7023 58        CLI
.7024 60        RTS
.7025 EA        NOP
.7026 EA        NOP
.7027 EA        NOP
.7028 EA        NOP
.7029 EA        NOP
.702A EA        NOP
.702B EA        NOP
.702C EA        NOP
.702D EA        NOP
.702E EA        NOP
.702F EA        NOP
.7030 A9 FF     LDA #$FF
.7032 8D 19 D0  STA $D019
.7035 A9 60     LDA #$60
.7037 A0 70     LDY #$70
.7039 8D 14 03  STA $0314
.703C 8C 15 03  STY $0315
.703F A9 60     LDA #$60
.7041 8D 12 D0  STA $D012
.7044 AD 11 D0  LDA $D011
.7047 29 7F     AND #$7F
.7049 8D 11 D0  STA $D011
.704C A9 00     LDA #$00
.704E 8D 20 D0  STA $D020
.7051 20 C0 70  JSR $70C0
.7054 4C BC FE  JMP $FEBC
.7057 EA        NOP
.7058 EA        NOP
.7059 EA        NOP
.705A EA        NOP
.705B EA        NOP
.705C EA        NOP
```

Il listato del «Moltiplicatore di sprite».

```
.705D EA        NOP
.705E EA        NOP
.705F EA        NOP
.7060 A9 FF     LDA #$FF
.7062 8D 19 D0  STA $D019
.7065 A9 90     LDA #$90
.7067 A0 70     LDY #$70
.7069 8D 14 03  STA $0314
.706C 8C 15 03  STY $0315
.706F A9 90     LDA #$90
.7071 8D 12 D0  STA $D012
.7074 AD 11 D0  LDA $D011
.7077 29 7F     AND #$7F
.7079 8D 11 D0  STA $D011
.707C A9 01     LDA #$01
.707E 8D 20 D0  STA $D020
.7081 20 D0 70  JSR $70D0
.7084 4C BC FE  JMP $FEBC
.7087 EA        NOP
.7088 EA        NOP
.7089 EA        NOP
.708A EA        NOP
.708B EA        NOP
.708C EA        NOP
.708D EA        NOP
.708E EA        NOP
.708F EA        NOP
.7090 A9 FF     LDA #$FF
.7092 8D 19 D0  STA $D019
.7095 A9 30     LDA #$30
.7097 A0 70     LDY #$70
.7099 8D 14 03  STA $0314
.709C 8C 15 03  STY $0315
.709F A9 C0     LDA #$C0
.70A1 8D 12 D0  STA $D012
.70A4 AD 11 D0  LDA $D011
.70A7 29 7F     AND #$7F
.70A9 8D 11 D0  STA $D011
.70AC A9 02     LDA #$02
.70AE 8D 20 D0  STA $D020
.70B1 20 E0 70  JSR $70E0
.70B4 4C 31 EA  JMP $EA31
.70B7 EA        NOP
.70B8 EA        NOP
.70B9 EA        NOP
.70BA EA        NOP
.70BB EA        NOP
.70BC EA        NOP
.70BD EA        NOP
.70BE EA        NOP
.70BF EA        NOP
.70C0 A2 10     LDX #$10
.70C2 BD 00 71  LDA $7100,X
```

```
.70C5 9D 00 D0  STA $D000,X
.70C8 CA        DEX
.70C9 10 F7     BPL $70C2
.70CB 60        RTS
.70CC EA        NOP
.70CD EA        NOP
.70CE EA        NOP
.70CF EA        NOP
.70D0 A2 10     LDX #$10
.70D2 BD 20 71  LDA $7120,X
.70D5 9D 00 D0  STA $D000,X
.70D8 CA        DEX
.70D9 10 F7     BPL $70D2
.70DB 60        RTS
.70DC EA        NOP
.70DD EA        NOP
.70DE EA        NOP
.70DF EA        NOP
.70E0 A2 10     LDX #$10
.70E2 BD 40 71  LDA $7140,X
.70E5 9D 00 D0  STA $D000,X
.70E8 CA        DEX
.70E9 10 F7     BPL $70E2
.70EB 60        RTS
.70EC EA        NOP
.70ED EA        NOP
.70EE EA        NOP
.70EF EA        NOP
.70F0 EA        NOP
.70F1 EA        NOP
.70F2 EA        NOP
.70F3 EA        NOP
.70F4 EA        NOP
.70F5 EA        NOP
.70F6 EA        NOP
.70F7 EA        NOP
.70F8 EA        NOP
.70F9 EA        NOP
.70FA EA        NOP
.70FB EA        NOP
.70FC EA        NOP
.70FD EA        NOP
.70FE EA        NOP
.70FF EA        NOP
```

stato proposto. Esso è in grado di visualizzare contemporaneamente ben 24 sprite, ma, apportandovi alcune modifiche, questo numero può ancora incrementare.

Per visualizzare 24 sprite (3 X 8) occorre usare la tecnica dell'interruzione (IRQ). Nel nostro caso si tratta di interruzioni video, cioè che avvengono quando il pennello elettronico raggiunge una determinata posizione. Detta posizione è imponibile da parte del programmatore e, soprattutto, non deve essere per forza una sola. In altre parole, stiamo mettendo in pratica quanto visto nel numero di febbraio a proposito della «riutilizzazione degli sprite» durante la stessa «pennellata» video. Il listato, al contrario di quanto potrebbe sembrare a prima vista, è tutt'altro che complicato. Richiede comunque la conoscenza di alcune istruzioni che ci apprestiamo a descrivere. A proposito, non ditemi che non avete a portata di mano un Monitor-Assemblatore. È assolutamente indispensabile per programmare in linguaggio macchina «come si deve». Per chi non lo sa, un Monitor-Assemblatore è il programma che facilita il compito di trascrittura delle istruzioni nel computer e che oltretutto permette di effettuare alcune operazioni utili come il riempire con un determinato valore una grande area memoria molto velocemente o copiare un blocco di RAM/ROM a partire da una locazione differente da quella attuale etc. Quindi se siete seriamente intenzionati a diventare dei programmatori, occorre procurarselo.

Tornando a noi, vediamo innanzi tutto di capire come si effettua la lettura di un disassemblato come quello del nostro listato. Partendo dall'alto a sinistra vediamo, in successione, le diciture:

```
PC IRQ SR AC XR YR SP.
```

Sotto ognuna di queste sigle troviamo i valori:

```
E37B EA31 A4 F1 30 11 DC
```

I nomi elencati prima sono nomi attribuiti a dei registri interni (RAM interna) al microprocessore, tranne l'IRQ, che è invece il «vettore» di salto che indica la routine da eseguire in caso di IRQ (quella normalmente provocata dal timer ogni sessantesimo di secondo). Diamo ora una breve spiegazione alla funzione di questi registri. Il PC (o Program Counter) è un registro a 16 bit (2 byte) che contiene un valore corrispondente all'indirizzo dell'istruzione da prelevare dalla memoria. L'SR (o Status Register) contiene dei valori che servono a indicare alcune condizioni generatesi in seguito all'esecuzione delle istruzioni precedenti (non preoccupatevi, ne riparleremo). L'AC (Accumulatore) è il registro nel quale vengono depositati tutti i dati (a 8 bit) che devono subire un'operazione e quindi una modifica. L'XR (Registro X) viene normalmente utilizzato in operazioni che richiedono un contatore (ma non è un contatore) o un puntatore; è a 8 bit e non supporta tutte le operazioni possibili, come fa l'Accumulatore. L'YR (Registro Y), come è facile intuire, è praticamente uguale a l'XR (cioè può fare quasi le stesse cose).

Infine l'SP (Stack Pointer) che è un puntatore a una particolare zona di RAM, adibita al momentaneo deposito di dati. Che fatica! Dunque, i valori (sono in esadecimale) di cui parlavamo, corrispondono al contenuto di detti registri e sono relativi all'ultima volta che il microprocessore ha eseguito un programma (ovviamente prima dell'avviamento del Monitor-Assemblatore, che è anch'esso un programma e quindi li modificherebbe in continuazione).

Queste due righe vengono visualizzate non appena il M.-A. viene fatto «partire» e possono essere visualizzate, normalmente, premendo il tasto R seguito da un return. Ci siamo dilungati un po' troppo vero? Proseguiamo. La terza riga comincia a rappresentare il programma vero e proprio, o meglio il disassemblato di questo. Il primo numero (sempre in esadecimale), cioè il 7000, è la prima locazione di memoria occupata da questi. Il numero che segue (78) rappresenta l'equivalente in esadecimale di detta locazione ed è un'istruzione.
La scritta «SEI» è il codice mnemonico dell'istruzione stessa, ovvero un nome che serve per ricordare più facilmente quell'istruzione (vi immaginate «che macello» se dovessimo ricordare a memoria il numero corrispondente a ciascuna istruzione?). Per scrivere questa istruzione in memoria io ho informato il mio M.-A. che volevo scrivere un programma a partire dalla locazione $7000 (il $ sta per esadecimale), con la seguente istruzione:

```
A7000
```

Di seguito ho scritto, sulla stessa linea, il codice «SEI» e quindi ho premuto il tasto return. Automaticamente il M.-A. ha assemblato il mio codice scrivendo al posto della mia riga, quella seguente:

```
A7000 78 SEI
```

e ne ha aggiunta un'altra come questa qui sotto:

```
7001.
```

Dopo il 7001 io ho scritto il codice LDA #$7F e così via. Non vi credo degli handicappati; penso solo che così è più semplice capire la lezione!

Continuiamo. Se arrivati a un certo punto ci vogliamo fermare, basta premere return a vuoto, ovvero senza scrivere nessuna istruzione. Possiamo così «listare» il nostro codice in Assembly, imponendo la seguente riga e premendo return:

```
D7000
```

La lettera D sta per «Dissassembla», così come la lettera A sta per «Assembla».

Finalmente possiamo spiegare cosa significano tutte quelle istruzioni. L'istruzione «SEI» fa in modo, ovviamente quando viene eseguita, che tutte le IRQ vengano disabilitate (niente più IRQ). Per riabilitarle occorre l'istruzione «CLI». Dimenticavo che per far partire un programma occorre impostare:

```
G7000 (ma quando è finito, non ora!)
```

oppure, se siamo in ambiente Basic il classico SYS seguito dall'indirizzo iniziale (cioè SYS 28672).

L'istruzione «LDA» ha diversi formati, ma nel nostro caso viene usata per «caricare» nell'accumulatore il valore $7F (il # vuol dire che si tratta di un dato e non di una locazione di memoria). Questo è il primo esempio di istruzione che occupa 2 byte. Tornando al programma, se viene eseguito solo fino a questo punto, esso disabilita le IRQ e mette nell'accumulatore il valore $7F, la successiva istruzione serve a «scaricare» l'accumulatore e anche essa ha diversi formati, ma nel nostro caso deposita detto valore nella locazione di memoria $DC0D (è uno dei registri dell'I/O). Cosa abbiamo fatto finora? Più semplice che mai. Abbiamo messo nella locazione $DC0D il valore $7F. Questo ci servirà per il regolare svolgimento delle IRQ.

Mi rifiuto di spiegarvi la funzione delle successive istruzioni contenute nelle locazioni antecedenti la $7010. Servono comunque a depositare i valore $30 e $70 rispettivamente nelle locazioni $0314 e $0315. Queste due ultime locazioni contengono il byte basso e il byte alto del vettore di IRQ e noi così facendo lo abbiamo modificato. Quindi alla prossima IRQ il microprocessore eseguirà il codice contenuto a partire dalla locazione $7030. Lo sapevo che sarebbe andata a finire così... adesso si arrabbieranno tutti... ma... non c'è più spazio. Vi prego, siate comprensivi continueremo la prossima puntata. Nel frattempo studiatevi (se ci riuscite) il listato e cercate di capirne il funzionamento... Vi ringrazio e vi lascio alla Megaposta.

Megaposta

Non arrabbiatevi!!!

...Ho seguito con interesse la rubrica Megagame 64 fino dalla prima puntata e devo dire che avete davvero avuto una buona idea. Non fate caso a qualche invidioso che dice di lasciar perdere perché è impossibile creare un gioco per posta e non fate caso ugualmente a qualche altro presuntuoso che, dopo aver fatto il saputello con uno sfoggio di cultura non richiesto, suggerisce egoisticamente di «passare attraverso MC-Link». Mi ha sorpreso il caro Gianni Z. che si pensa di essere Einstein proponendo il quiz delle tre età dei figli. Probabilmente la risposta è una freddura, perché con quei tre dati è matematicamente impossibile dare una risposta. Ma perché date importanza a gente come Gianni Z. facendo occupare alla sua suppongo chilometrica lettera ben 66 righe? Certo l'idea di un multigioco nel tempo è originale, ma è solo questa la cosa originale...

Alessandro Merolli, Roma

È chiaro che ognuno è libero di esprimere la propria opinione... Alessandro ha detto la sua. Tuttavia «hai toppato» prendendotela con Gianni Z. e con il suo giochino che, come puoi constatare dalla «lettera» del prossimo lettore, ha una soluzione rigorosamente matematica. Questo comunque non vuol dire che chi propone indovinelli ha sempre ragione...

Soluzione...

Caro Marco, ti scrivo per rispondere al quesito proposto da Gianni Z. nella rubrica Megaposta. Mi pare sia molto semplice e non ci ho messo molto per risolverlo. Ma veniamo al dunque:

— le età sono tre ed il loro prodotto fa 36, ci viene detto: poiché si assume che le età siano intere, bisogna scomporre in fattori il prodotto: 1-2-2-3-3; raggruppando i fattori in modo da ottenere tre numeri, si ottengono le seguenti otto combinazioni (escludendo le duplicazioni):

1-1-36
1-2-18
1-3-12
1-4-9
1-6-6
2-2-9
2-3-6
3-3-4

— con la seconda risposta ci viene detto che la somma delle tre età è pari al numero civico della casa di fronte, ma l'amico risponde che l'indizio non è ancora sufficiente; le varie combinazioni indicate danno come somma rispettivamente:

38-21-16-14-13-13-11-10.

Bisogna escludere le somme che danno luogo ad una sola soluzione, perché in questo caso l'amico non avrebbe bisogno (pur vedendo il numero civico) di una terza domanda: la somma è quindi 13! Si tratta ora di scegliere tra 1-6-6 e 2-2-9;

— con l'ultima risposta ci viene detto che il più piccolo dei figli ha gli occhi azzurri: nel caso 2-2-9 il più piccolo NON ESISTE dato che ci sono due gemelli di due anni; la soluzione cercata è quindi:

1-6-6 !!!

Vinicio Coletti, (inviata tramite MC-Link)

Così, grazie a Vinicio, abbiamo arrestato la caduta dei capelli di tutti quelli che si stavano scervellando per trovare la soluzione.

Strane proposte

...Proporrei di fare il gioco in Basic semplice (mi spiego: senza «Simon's Basic» o linguaggio macchina), per fare in modo che veramente tutti possano provare l'emozione di farsi un vero gioco di quelli tosti. Quando spieghi però (il «però» è onnipresente), dovresti essere un po' più «ordinato», cioè, ogni tanto non riesco a seguire il discorso, ma forse succede perché sono un po' «paraplegico». Non so se questa lettera verrà pubblicata, ma se succede, potrò festeggiare perché sarebbe la prima volta che il mio nome (non spaventarti per la sua lunghezza) compare su una rivista.

Christian Kokkinomagoulos, Bologna

Aiuto... chiamate i pompieri!

A parte il nome che mi sembra abbastanza comune (!), ma ti rendi conto di quello che hai scritto? Cercare di realizzare un Megagame in Basic è come provare a scavare una galleria sotto al Monte Bianco con paletta e secchiello da spiaggia! Prova a seguire le mie lezioni e vedrai che non è poi così difficile «sto' linguaggio macchina».

Colonna sonora

Salve Marco, mi chiamo Gianfranco Gramiglia, ho 21 anni e abito a Napoli...

... A cosa ti posso servire?

Interessi principali della mia umile esistenza:

PRIMO-la mia ragazza
SECONDO-pensare
TERZO-suonare e digitare (pari merito)
ecc. ecc.

Il primo non può interessarti (anzi, non deve).

Il secondo può interessarti relativamente visto che potrei mettermi a pensare a un nuovo videogame.

Il terzo: digitare (programmare) non sono un mostro (forse con l'aiuto di MC e dei tuoi articoli chissà...); decisamente rimane suonare. Chiedi una colonna sonora e sarà fatta; veloce, avvincente, dolce, romantica, lenta, maniacale, hard, disco, come vuoi tu e io la farò (ehm, tenterò, ma tenterò fino allo spasmo visto che ho i mezzi).

P.S. Il «nostro» videogame hai detto deve sfondare, ma dove? Italia, Europa, U.S.A., per quale età dev'essere? Dovresti imporre dei limiti.

Gianfranco Gramiglia, Napoli

Ragazzi abbiamo trovato il nostro musicista! Caro Gianfranco, finora sei stato l'unico che ha pensato alla musica... mandami una musicassetta con un tuo elaborato... vediamo quello che si può fare! Comunque non capisco perché vuoi imporre limiti...

Referendum

...Purtroppo il mio apporto alla rubrica potrà essere, credo, molto piccolo per via della scelta del computer su cui scrivere il gioco, il 64. Sono un felice possessore di Amiga 1000, proveniente tra l'altro da uno Spectrum 48k, quindi conosco ben poco il mio fratellino minore. Capisco che aprire il campo un po' a tutti i computer avrebbe comportato una mole di lavoro troppo grossa, ma almeno un Referendum si poteva fare. Dopo tutto il 64 è il più diffuso tra gli utenti di videogiochi, non tra i produttori...

Emilio Orione, Nizza Monferrato (AT)

Sei proprio convinto di quello che dici?

C'è anche lui

...Noi improvvisiamo i 2 poliziotti della serie CHIPS...

Gianni Manenti, Noto (SR)

Pubblico la tua «lettera» (cioè il tuo nome) per dovere di cronaca, ma i CHIPS lasciamoli stare eh... (non ti sarai mica offeso?).

Anche per questo mese abbiamo finito, ma non disperatevi... Ciao! MC