VIC da zero +64

di Tommaso Pantuso

Un maxi carattere sullo schermo

Oggi affrontiamo il problema della lettura del generatore di caratteri del Vic e del C 64. Per il C 64 avremo prima bisogno di esaminare certi aspetti hardware del Sistema mediante i quali sarà poi semplice comprendere il perché di alcune operazioni da compiere per la lettura della mappa caratteri.

Il generatore di caratteri del Vic 20

Cominciamo ad esaminare la mappa dei caratteri del Vic 20. Come abbiamo visto, essa è posta a partire dalla locazione 32768. In essa (come in quella del C 64) il primo carattere, composto da un gruppo di otto byte, quello di codice "0", è la "@". In una matrice 8×8, come visto la volta scorsa, esso può essere rappresentato come nella figura 1 e agli otto byte che lo compongono, come sappiamo, possiamo associare i numeri:

28, 34, 74, 86, 76, 32, 30, 0.

Se infatti andiamo a leggere le locazioni che vanno da 32768 a 32775 con:

```
10 FORI=0TO7
20 PRINT PEEK(32768+1)
30 NEXT
```

troveremo tali numeri.

Nota. Se avessimo letto le locazioni corrispondenti al generatore del C 64 (ammesso di poterlo fare in maniera immediata) avremmo trovato (figura 2) i valori:

60, 102, 110, 110, 96, 98, 60, 0.

Infatti, nei due computer, non tutti i caratteri hanno la stessa forma.

Il metodo per individuare l'esatta posizione d'inizio di un carattere, noto il suo codice di schermo S, è il seguente:

1) si prende S e si moltiplica per 8;

2) si somma il valore trovato all'indirizzo d'inizio del generatore di caratteri.

Nel caso precedente, il codice di @ è 0 per cui, $0 \times 8 + 32768 = 32768$ che è appunto la posizione cercata. Per una A, di codice 1, avremo invece $1 \times 8 + 32768 =$ 32776 che è appunto la locazione da cui comincia a formarsi tale carattere.

Facciamo ora una verifica che ci sembra interessante. Abbiamo detto che ciascun numero contenuto in ciascuna locazione del generatore di caratteri, se trasformato in binario, fornisce un insieme di "0" e di "1" (otto in tutto) che, analizzato dal sistema renderà sullo schermo un pixel acceso in corrispondenza di ogni "1" ed un pixel spento in corrispondenza di ciascuno "0". Se ora noi leggiamo ad esempio il contenuto dei precedenti otto byte, lo trasformiamo in un numero binario e poniamo i pattern ottenuti l'uno sotto l'altro, dovremmo ritrovare, sullo schermo ed ingrandita, una griglia nella quale, osservando tutti gli "1", dovremmo poter intravedere la forma del carattere. Verifichiamo se questo fatto è vero.

Per la @, decodificando in binario i valori trovati, abbiamo i numeri: 00011100, 00100010, 01001010, 01010110, 01001100, 00100000, 00011110, 00000000. Ponendo tali "stringhe" l'una sotto l'altra abbiamo:

```
 00011100
 00100010
 01001010
 01010110
 01001100
 00100000
 00011110
 00000000
```

e se per maggior chiarezza, asportiamo gli "0" troveremo la seguente forma:

```
  111
 1   1
1  1 1
1 1 11
1  11
 1
  1111
```

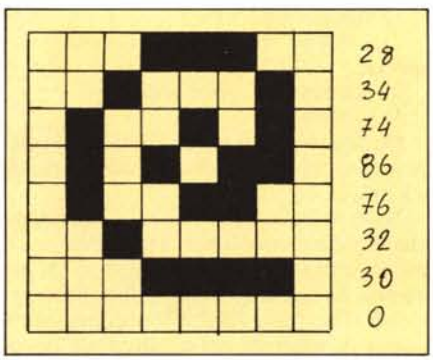

Figura 1 - La @ così come viene visualizzata dal Vic 20.

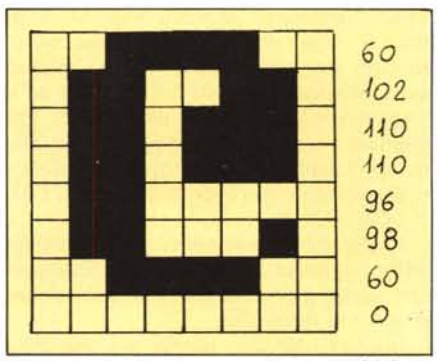

Figura 2 - La @ nel generatore di caratteri del C 64.

Listato 1

```
1 REM ---------------------------------
2 REM CON QUESTO PROGRAMMA VIENE LETTO
3 REM IL CONTENUTO DELLA MAPPA DEI CA-
4 REM RATTERI E VISUALIZZATA LE FORMA
5 REM CORRISPONDENTE INSIEME AI VALORI
6 REM DELLE LOCAZIONI LETTE E DEL LORO
7 REM CONTENUTO  (VIC 20)
8 REM---------------------------------
9 PRINT"■■■"
10 PRINT"□"
20 INPUT"CARATTERE";Z$
25 PRINT"■ --------------------"
26 PRINT" FORMA   LOC. CONT."
27 PRINT" -------------------- "
30 Z=PEEK(7713):PRINT
34 R=34816
40 FORI=0TO7
70 D=R+Z*8+I
110 DC=PEEK((R)+Z*8+I)
370 R$="        "
410 N$ = "":VV=DC
420 V1=INT(VV/2):T=VV-2*V1:T$=RIGHT$(STR$(T),1)
425 IFT=0THENT$=" "
426 IFT=1THENT$="."
427 N$=T$+N$
430 G$=RIGHT$((R$+N$),8):G2$=RIGHT$(G$,8):VV=V1
440 IF V1<>0THEN420
460 PRINTG2$;D;DC
480 NEXT
510 GETA$:IFA$=""THEN510
520 GOTO10
```

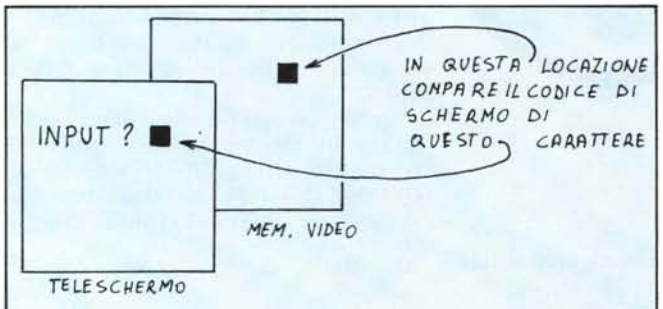

Figura 3 - Possiamo individuare il codice di schermo di un carattere visto sul teleschermo individuandone l'immagine nella memoria video.

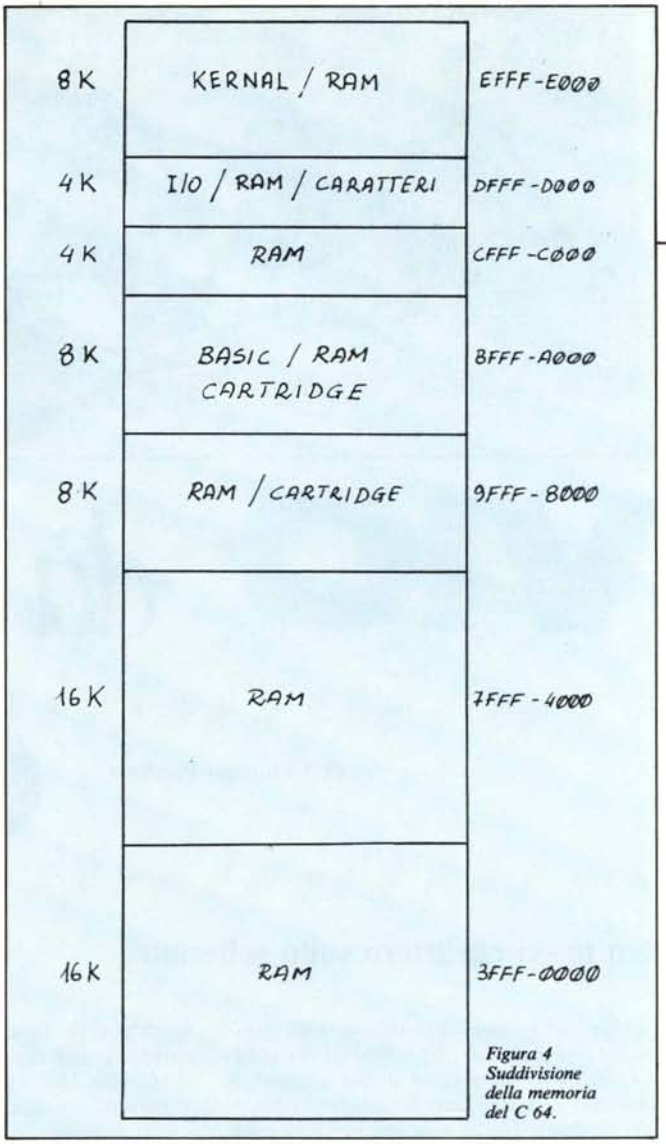

Figura 4 Suddivisione della memoria del C 64.

che è proprio quella della @ così come viene rappresentata dalla macchina.

Un po' di pratica

Nel listato 1, riportiamo un breve programma che, riferendosi al secondo insieme di caratteri, effettua automaticamente questo processo restituendo, dopo aver trasformato il contenuto di ogni byte interessato in binario, al posto degli "0" degli spazi bianchi e al posto degli "1" un punto. Inoltre, vengono forniti anche i numeri delle locazioni occupati dal carattere ed il loro contenuto in notazione decimale. Per ottenere i caratteri in campo inverso, alla richiesta della macchina, inserire il carattere desiderato dopo aver premuto <CTRL + RVS ON >.

Anche se molto semplice e di immediata interpretazione, è forse il caso di commentare alcuni punti del programmino dimostrativo in questione.

Intanto, qualche perplessità potrebbe sorgere dall'osservazione della linea 9. Con essa viene semplicemente abilitato il set di caratteri alternativo (N in campo inverso) e disabilitata la funzione della pressione contemporanea dei tasti Shift/ Commodore (H in campo inverso). La parte più difficile — si fa per dire — è ottenere una scrittura simile a quella della linea in questione: il problema si risolve molto semplicemente. La sequenza di operazioni da compiere è la seguente:

1) digitare PRINT"";

2) portarsi con il cursore sulle prime virgolette;

3) premere due volte Shift/Inst;

4) scrivere NH.

Naturalmente lo stesso effetto può essere sortito con i codici di controllo Chr$(14) e Chr$(8)

Nella linea 20 viene richiesto il carattere che vogliamo visualizzare ingrandito, ma il procedimento di input impiegato è un po' diverso da quelli consueti. Infatti dalla variabile d'ingresso Z$ dobbiamo ricavare il codice di schermo del carattere da riprodurre ingrandito per poi andarlo ad intercettare nel generatore. Si potrebbe impiegare un procedimento di calcolo partendo dal codice Ascii del carattere in questione e calcolare da esso quello di schermo basandoci sulle formule da noi fornite non molto tempo fa. Con un semplice artificio possono essere però evitati tutti i calcoli. Vediamo come.

Dopo la richiesta di Input, noi premeremo il tasto corrispondente al carattere che vogliamo visualizzare ingrandito. Il carattere in dimensioni normali, dopo la risposta all'input viene posto sul teleschermo in una certa posizione e quindi esso avrà un codice immagine nella memoria di schermo (figura 3). Conoscendo allora l'esatta locazione del carattere sul video, sapremo anche in quale locazione della memoria di schermo è contenuto il codice corrispondente e quindi potremo prelevarlo ed usarlo per individuare il punto da cui il carattere comincia a formarsi nel generatore. Facendo qualche osservazione, ci accorgiamo che troveremo il codice del carattere in questione nel byte 7713 (linea 30) della

memoria video. Servendoci allora del contenuto di tale byte e servendoci del calcolo della linea 110 cominceremo a leggere il contenuto degli otto byte individuati nel generatore di caratteri convertendolo in stringhe binarie (in cui gli "0" sono stati sostituiti da spazi e gli "1" da punti) che visualizzeremo sullo schermo l'una sotto l'altra. L'effetto risultante sarà un carattere simile nella forma a quello di partenza, ma dalle dimensioni maggiori.

La memoria ambigua del C 64

Se leggere il contenuto del generatore di caratteri del Vic 20 è una cosa abbastanza semplice, per il C 64 il procedimento è leggermente più complicato, questo per la particolare struttura interna della macchina. Per capire quindi bene il significato delle operazioni che compiremo in seguito è allora necessario dare uno sguardo ad alcuni aspetti fondamentali di questa struttura.

Come saprete, il C 64 è gestito dal microprocessore 6510 che utilizza un bus di indirizzi a 16 bit: ciò significa che esso è capace di indirizzare 64K di memoria. Sappiamo però che il C 64 possiede, oltre a 64K di Ram, anche 20K di Rom occupati dal Basic, dal Sistema Operativo e dal generatore di caratteri. Se allora calcoliamo l'ammontare della memoria presente nel computer ci accorgiamo che supera i 64K effettivamene indirizzabili dal microprocessore e viene logico chiedersi come ciò sia possibile (figura 4).

Il "trucco" sta nel 6510 stesso il quale contiene una "porta d'ingresso/uscita" con la quale, al momento opportuno, viene selezionata la Ram o la Rom del sistema o le zone di I/O. In altre parole (figura 5) possiamo paragonare la memoria del C 64 a due pagine sovrapposte di un libro a cui non possiamo accedere contemporaneamente, ma dobbiamo leggerne necessariamente una sola per volta. La porta d'I/O cui accennavamo è composta da 5 bit ed è situata nella locazione 1 della memoria del computer. La sua configurazione iniziale è 101111: ogni bit alto rappresenta un'uscita ed ogni bit basso un ingresso.

Naturalmente, visto che tale "porta" a 5 bit occupa una locazione di memoria ad 8 bit è evidente che lo stato dei tre bit che avanzano, nel nostro caso quelli più alti (6, 7, 8,), non è rilevante. Per essere più chiari sulla funzione di questa porta, aggiungiamo che, a seconda che uno dei suoi bit venga posto alto oppure basso, le influenze sul sistema consistono nell'abilitazione di un determinato dispositivo periferico (registratore a cassette) e nella discriminazione tra la Ram e la Rom sovrapposte. Descriviamo rapidamente la funzione di ciascun bit.

Il bit 0 posto ad 1, quindi in stato di output, controlla la Rom del Basic situata tra gli indirizzi esadecimali A000 a BFFF e la Ram sottostante. Il bit 1 alto, è invece dedicato alla Rom del Kernal ed alla Ram entrambe situate da E000 a FFFF. Anche il bit 2 è un'uscita e quindi si trova nello stato logico 1. Esso si occupa della selezione della Rom e dell'I/O (Vic, Sid, Cia, ecc.) situati da D000 a DFFF. Il bit 3, sempre un'uscita, è la linea di scrittura del registratore mentre il bit 4, questa volta posto come ingresso, rivela la pressione dei tasti (sempre del registratore). Infine, il bit 5, controlla lo stop e lo start del motore del Datasette. Per il momento questa sommaria descrizione può bastare; daremo comunque in seguito ulteriori delucidazioni sulla porta di cui abbiamo appena parlato e su come si possa influire sulla configurazione delle sue linee mediante l'opportuno posizionamento di un registro di controllo posto "alle sue spalle" e chiamato "registro di direzione dati".

Soffermiamoci ora per un attimo solo sulla funzione svolta dal bit 2 della porta in questione. Per quanto ci interessa, diciamo che con esso è possibile selezionare, nell'area accessibile dal microprocessore, la Rom dei caratteri o la sezione sovrastante dei dispositivi d'I/O del sistema. Se tale bit si trova in condizione 1 (condizione di default), il controllo viene ceduto ai dispositivi d'I/O e quindi il generatore di caratteri non è naturalmente accessibile dall'utente. Se invece abbiamo la necessità di leggere il contenuto del generatore, ad esempio per travasarlo in Ram, dovremo effettuare la commutazione di tale bit da 1 a 0. E questo è il primo passo da compiere per la lettura.

L'ulteriore necessità è quella che non si verifichino interruzioni (interrupt) nel momento in cui si sta accedendo al generatore di caratteri e questo per una ragione molto semplice: la Rom dei caratteri e dispositivi d'I/O sono sovrapposti. Ora, dato che le interruzioni sul sistema sono gestite appunto da tali dispositivi, se durante la lettura del generatore si verificano degli interrupt, il sistema subirebbe un blocco. Per tale ragioni le interruzioni vanno necessariamente escluse.

Le interruzioni sono gestite da uno dei Cia presenti nella macchina mediante una sua porta d'ingresso/uscita quindi è su quest'ultima che dovremo agire per l'accesso all'ambito generatore di caratteri. Sarà appunto questo l'argomento del prossimo articolo nel quale analizzeremo più in dettaglio alcuni degli aspetti descritti questa volta.

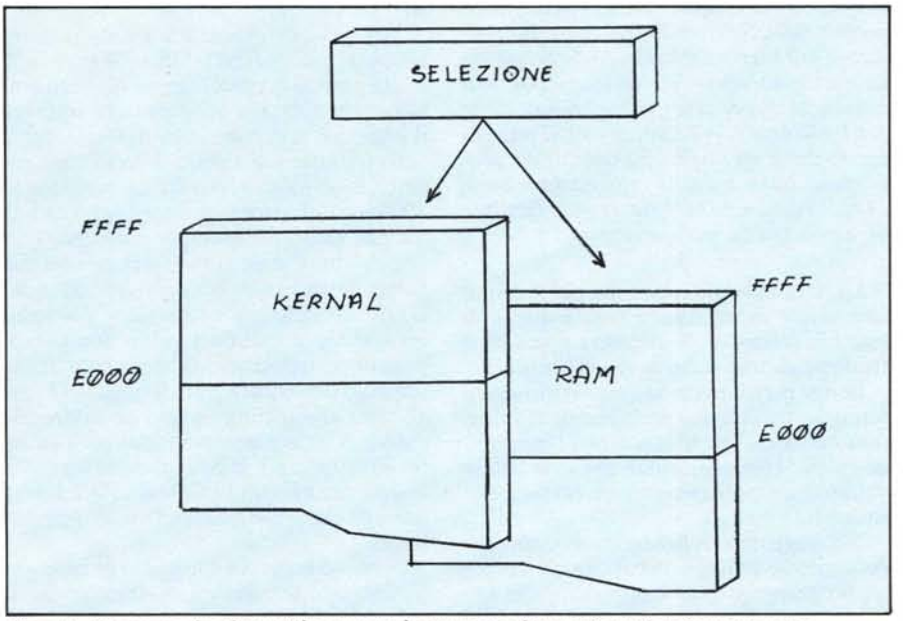

Figura 5 - L'accesso a due elementi che occupano la stessa zona di memoria non è mai contemporaneo.