situazione e di prendere decisioni in proposito.

Tutte queste attività vengono svolte sia per il pianeta madre che per quelli che man mano vengono colonizzati.

Bene, un piccolo passo avanti è stato fatto. Certo occorre ancora scendere nei dettagli veri e propri che poi ci permetteranno di realizzare il programma... vero e proprio. Sarà proprio questo il compito che ci aspetta nella prossima puntata e forse si potrà anche cominciare a parlare di programmazione (finalmente!).

Nel frattempo vi ricordo che se avete qualche consiglio o qualche aggiunta da propormi io sono sempre pronto ad ascoltarvi.

Scuola di videogame

Moltiplicazione degli sprite (seconda parte)

Dove eravamo rimasti?... ah! sì, «quindi alla prossima IRQ il microprocessore eseguirà il codice contenuto a partire dalla locazione $7030»

Dunque, come è possibile fare la stessa cosa in Basic? Basterebbe (notare il condizionale) «pokare» nella locazione $DC0D il valore #$7F e nelle locazioni $0314 e $0315 rispettivamente i valori #$30 e #$70. Purtroppo però non è possibile inserire queste due ultime poke senza provocare il blocco del computer, in quanto abbiamo trascurato la funzione dell'istruzione in LM "SEI" che come detto disabilita le IRQ e (come non detto) in Basic non esiste. Disabilitare le IRQ è necessario in quanto altrimenti anche in LM ci sarebbe il blocco del computer (non è detto, ma meglio premunirsi). Perché il C-64 si blocca? Supponiamo che, girando per Roma con la vostra automobile, non riuscite a trovare la strada giusta (non è poi così difficile). Un gentile signore si fa avanti e comincia a descrivervi la strada che dovete seguire. Vostro padre che è alla guida, però, non ha capito che state parlando con quel tizio e continua ad andare avanti. Non siete riusciti a capire bene l'informazione del signore e inevitabilmente vi perderete. La stessa cosa avviene nel Commodore. La macchina che continua a camminare sono le IRQ e voi che state ascoltando l'informazione siete il vettore di IRQ. Il microprocessore tenta di comunicarvi «la strada» ma non fa in tempo e l'interruzione salta a un vettore sbagliato! (spero di non aver complicato la comprensione di questo semplice concetto). Dicevamo che in Basic non esiste questa istruzione, ma non è un problema in quanto un programma del genere non viene mai scritto in Basic. Dunque SEI, LDA, STA, LDA, LDX, STA, STX... NOP! Cosa vuol dire NOP? Questa istruzione che occupa sempre solo un byte serve per riempire una zona di memoria quando quest'ultima è occupata da altre istruzioni inutili. Direti voi: «a cosa serve?». Serve quando quelle istruzioni sono di intralcio al programma e quindi devono essere ignorate. Sostituendole con delle NOP il risultato che si ottiene è che il microprocessore, incontrando le NOP, non fa niente e continua con la prossima istruzione. L'utilità di questo sistema si riscontra quando (come nel mio caso) un programma viene modificato e quindi ci si ritrova con codici superflui, come accade ad esempio se la modifica comporta la scrittura di un'istruzione che occupa due byte invece di tre. In questo caso il terzo byte sarebbe di intralcio al programma e magari comporterebbe confusione di istruzioni, quindi va sostituito con una NOP. Esempio pratico: supponiamo che (tenete d'occhio il listato del numero di aprile) a partire dalla locazione $7010 abbiamo scritto (invece della NOP) l'istruzione LDA $7840. Ci accorgiamo solo dopo aver già scritto il resto del programma che l'istruzione è errata. Apparentemente il rimedio è semplice; basterebbe sostituire a LDA $7840 l'istruzione giusta (nel nostro caso la LDA #$00). Quest'ultima istruzione però occupa 2 byte invece che tre come nel caso della LDA $7840. Se scriviamo «sopra» ad essa, l'ultimo byte resterà al suo posto e verrà interpretato come una normale istruzione (cosa che noi non vogliamo). Infatti il terzo byte (equivale al $78), corrisponde all'istruzione SEI che in tal caso non darebbe fastidio (perché due SEI di seguito non sono dannosi), ma poteva anche andarci peggio. Se ad esempio avevamo un $8D, equivalente al codice operativo dell'istruzione STA, i successivi due byte venivano interpretati come facenti parte della menzionata istruzione quindi veniva fuori l'istruzione STA $128D e il disastro era assicurato perché, oltre alla presenza dell'istruzione sbagliata, ci sarebbe stata anche la modifica delle successive istruzioni (meglio non parlarne). La soluzione migliore in questo caso è depositare nella locazione $7010 la NOP e scrivere a partire dalla locazione $7011 l'istruzione LDA #$00. Chiaramente era anche possibile traslare tutto il programma con un'istruzione da impartire al monitor-assemblatore, ma in questo caso è superfluo, soprattutto tenendo conto che non sempre è possibile eseguire questa operazione senza modificare alcuni punti del programma. Per il momento lasciamo stare e proseguiamo. A proposito, a partire dalla locazione $7025 fino alla 702F sono state inserite una serie di istruzioni NOP ma questo non vuol dire che avevo sbagliato tutto il programma. Le NOP sono utili anche nel caso in cui non sappiamo a priori se in seguito sarà necessario inserire altre istruzioni, quindi solitamente vengono utilizzate per «lasciare occupato il posto». Da tutto questo discorso traspare che a differenza di quanto avviene in Basic, l'inserzione di nuove linee di programma non è poi così semplice.

Torniamo al listato. Il resto del programma, contenuto dalla locazione $7011 alla locazione $7020, fa in modo che la prima IRQ si verifichi in corrispondenza della posizione «zero» del pennello elettronico e abilità le interruzioni del RASTER. In seguito riabilita le IRQ e ritorna (in questo caso) al Basic. Le locazioni $D012 e $D011 contengono la posizione del pennello elettronico, ma se usate in scrittura, permettono di impostare una particolare posizione del pennello elettronico che, quando verificata, genererà l'interruzione (che discorso contorto). In dettaglio, la locazione

$D012 contiene gli otto bit meno significativi e la $D011 contiene (tra le altre cose) il bit più significativo (in quanto 256 divisioni non sono sufficienti a ricoprire l'intera area del RASTER; 512 sono più che sufficienti). Per impostare la posizione «zero» occorre azzerare la locazione $D012 e impostare a zero il bit più significativo della locazione $D011. La prima operazione è semplice e la sappiamo fare tutti, mentre la seconda richiede l'introduzione di una nuova istruzione: la AND. Azzerare solo un bit infatti significa lasciare inalterati gli altri e il sistema migliore per fare ciò è quello di depositare il contenuto del registro nell'accumulatore, sottoporlo ad un'operazione di and logico e ridepositarlo nella sua locazione. Il valore #$7F in binario si scrive 01111111, cioè il bit più significativo è posto a zero. Supponiamo di avere un qualunque valore nel registro $D011, ad esempio 10 (cioè, in binario, 00001010). Un'operazione di and logico con il valore 01111111 produce come risultato il valore (binario)... 00001010. Praticamente il byte è restato inalterato, ma se invece di avere 00001010 avevamo 11111111, il byte diventava 01111111, cioè il bit più significativo veniva azzerato (come volevasi dimostrare). A titolo di esempio, se volevamo azzerare il bit meno significativo, bastava eseguire un and logico tra il valore 11111110 e quello contenuto nel registro $D011 (semplice no?). Non pretendo di farvi un corso di elettronica digitale, ma spero che la mia spiegazione sia stata ugualmente sufficiente. Torniamo all'istruzione AND. Essa è proprio quella che esegue l'operazione di and logico tra accumulatore e valore fornito. Basta un LDA $D011, AND #$7F, STA $D011 e l'azzeramento è effettuato. Le successive due istruzioni depositano il valore #$00 nella locazione D01A e cioè abilitano le interruzioni RASTER. Il CLI sappiamo già che serve a riabilitare le IRQ e infine l'RTS serve a tornare indietro (è simile al RETURN del Basic).

Dalla locazione $7030 in poi troviamo la famigerata routine IRQ (ora viene il bello). La funzione di queste istruzioni non dovrebbe essere tanto oscura, almeno per quello che riguarda il loro... funzionamento. È interessante invece osservare il risultato che esse producono. «Carica l'accumulatore con #$FF e mettilo in $D019». Questa operazione è la meno interessante, ma va ugualmente eseguita ad ogni IRQ. Seguono quattro istruzioni adibite al cambiamento della routine IRQ (di nuovo!). Sono molto simili a quelle usate la prima volta. È curioso osservare che questa volta le IRQ non vengono disabilitate, ma questo perché siamo già in IRQ (eh, eh, eh) e quindi non c'è pericolo. Le istruzioni seguenti sono ancora del tipo... «già viste». Servono a cambiare la posizione del IRQ raster che si verificherà ora alla posizione #$60. Da $704E in poi troviamo due istruzioni che azzerano o meglio pongono il valore 0 nella locazione $D020, il che (finalmente) produce qualcosa di visibile, cioè cambia il colore del bordo (e lo fa diventare nero). Locazione $7081, nuova istruzione: JSR. Velocemente: è l'equivalente dell'istruzione Basic GOSUB. In questo caso c'è un "GOSUB $70D0". Saltiamo anche noi a questa locazione e continuiamo la nostra spiegazione da lì, cioè continueremo la nostra spiegazione da lì visto che per questo mese direi che può bastare, no?..

Megaposta

Ci divertiamo eh? Sono arrivate ben otto lettere con la soluzione del giochino proposto dal famigerato Gianni Z. Sei di queste l'hanno risolto in maniera esatta, mentre le restanti due... hanno sbagliato! Mi sembra doveroso quindi elencare i nominativi dei lettori che sono riusciti nell'impresa. Permettetemi di cominciare con un gruppo di bambini che, con l'aiuto del loro maestro, hanno risolto il giochino a scuola. Sono gli alunni della terza classe delle scuole elementari «L. Ariosto» di Reggio Emilia:
Federica Melegari, Elisa Ferroni, Giuseppe Aiello, Salvatore Rivello, Ernesto Lettieri, Francesco Maseroli, Alessandro Spaggiari, Stefano Trotta.

Cosa dire a questi cari bambini... BRAVISSIMI! Un bravo lo merita comunque anche il loro maestro, Raimondo Motti.

Proseguiamo l'elenco dei bravi.
Roberto Croci da Legnano (MI)
Paolo Costabel da Genova
Damiano Verzulli da Chieti Scalo (CH)
Riccardo Giannetti da Torrenieri (SI)
Marco Zuccarini Chieti Scalo (CH)

Ed ora che ne dite dell'elenco dei... meno bravi? Si tratta di Gianni Sarti, che dopo aver insultato tutta la redazione si pavoneggia con una soluzione sbagliata (ah, ah, ah). L'altro è Emanuele Aliberti, che per lo meno apre il discorso con un «credo di aver trovato la soluzione al problema...». La soluzione è apparsa sul numero precedente e a quest'ora dovrebbero averla già letta...

Passiamo alle lettere.

Ma come hai fatto?

«... Anche il programmino più sciocco si traduce nel blocco totale del computer o nel migliore dei casi in un ritorno al Basic (ovviamente senza risultati). Vorrei chiedere perciò a Marco Pesce, che mi sembra molto tosto sull'argomento, come ha fatto a diventare così bravo, cioè quali libri ha letto e ritiene necessari e sufficienti per imparare le cose giuste e fare un po' di esperienza. Io dispongo di un Commodore 64 e di un Sinclar QL».

Roberto Croci, Legnano (MI)

Caro Roberto, com'è facile intuire che la mia esperienza (modestia a parte) non deriva solo da un profondo studio teorico. Essa è il frutto di molta pratica e di molti tentativi andati a male. Nel campo informatico è necessario fare così e tu dovresti saperlo. Non è il caso quindi di allarmarsi se il nostro programmino LM non gira al primo colpo. A proposito, è opportuno munirsi di tasto RESET; il blocco di un programma in linguaggio macchina non perdona. Cosa consigliarti quindi se non di seguire le mie «lezioni» e i miei suggerimenti?

Voglio uno scrolling!

«Sono consapevole dell'insufficienza di spazio da dedicare alla scuola di Videogame, ma non mi spiacerebbe un microscopico listatino in LM sullo scrolling in 2D...».

Roberto Ricci, Torino

Accontenteremo Roberto? Ma certo e accontenteremo anche qualcun altro un po' più esigente...

Offresi programmatore

«Idee per un videogame onestamente non ne ho, anche perché penso che ormai tirar fuori qualcosa di nuovo sia molto, ma molto arduo... Posso comunque offrire, se necessario, la mia esperienza di programmatore sia Basic che LM (v. per esempio AlfaDisk e NL-Printgraf pubblicati su MC, più molti altri articoli su CCC e Personal Computer)...».

Roberto Morassi, Pistoia

Preparati Roberto...

Per finire ringrazio inoltre:
Marco Paolini di Cernusco sul Naviglio (MI), Giuliano Peritore di Latina, Andrea Beltrame di Novi Ligure (AL), Nicola Marangon di Cappella di Scorze' (VE), Luca Sassone di Caronno P.lla (VA), per aver inviato il loro contributo.

Tanti saluti a tutti.

Le pubblicazioni Technimedia

AUDIOREVIEW
La più qualificata rivista italiana di elettroacustica ed alta fedeltà

MCMICROCOMPUTER
La più diffusa e più autorevole rivista italiana di informatica

OROLOGILE MISURE DEL TEMPO
La prima rivista per chi conosce il valore del proprio tempo

Technimedia
Via Carlo Perrier, 9 - 00157 Roma - Tel. 06/4513931