Commodore 64 e la programmazione in linguaggio macchina

(Il costrutto read-poke-data per la gestione di controlli, suono, grafica, sprite, file koala...)

Dott. Bartolomeo Davide Bertinetto

INDICE:

Prima pubblicazione: 6-12-2015

Ultima revisione: 17-11-2019

Bertinetto torna ai PDF gratuiti

Con queste righe vorrei spiegare perché sono ritornato al mondo dei PDF gratuiti.
AmazXX è un servizio stupefacente ma che non rispecchiava però più le mie intenzioni divulgative...
Tanti anni fa ho pubblicato il mio primo libro PDF e poi sono andato avanti con la scrittura di altri lavori. Tutti hanno avuto un buon successo nella loro forma gratuita. Successivamente ero passato alla grande 'A' e così è diventato il mio 'mondo' di scrittura per tantissimo tempo. Nessun problema quindi e il formato KindXX è stato un bel successo per me. Nonostante ciò il 'tarlo' del PDF gratuito nella mia mente è costantemente cresciuto fino ad arrivare ad oggi giorno, in cui realizzo questa idea. Voglio pertanto pubblicare per pura soddisfazione, senza troppi perfezionamenti 'maniacali' nella sintassi dei lavori, che fa perdere all'autore tempo in modo sproporzionato! Voglio dedicarmi unicamente all'idea e renderla accettabile nella forma ma senza esagerare troppo nei perfezionismi. Quando un lavoro è completo, viene pubblicato sul mio sito e quindi posso concentrarmi su altre idee senza dover riprendere continuamente vecchie opere per eliminare sviste insignificanti, solo perché si tratta di materiale a pagamento.
Purtroppo o per fortuna sono sommerso da nuovi progetti che affollano la mia mente, continuamente... Devo 'adattarmi' ancora una volta per riuscire a fare tutto da solo. Non ho voglia di affidarmi ad altre persone e così la pubblicazione gratuita PDF mi permetterà di caricare online lavori, esprimendo le mie idee anche se magari non perfetti dal punto di vista sintattico o di impaginazione.
Non pensate però a lavori di serie 'B', anzi confido che il risultato non sarà poi così diverso per il lettore rispetto a quando usavo Ebook Reader. Al contrario per me lo sarà molto in termini di mole di lavoro, dato che non avrò lo 'stress' dei 'sensi di colpa' se mi sarò scordato di rivedere una parte. Se qualcosa sarà da rianalizzare lo farò ma quando ne avrò il tempo, senza fretta.
Mi piace condividere i miei pensieri sui vari temi su cui sfociano le mie passioni. Certo qualche vecchia opera resterà su AmazXX e ne sono contento dato che si tratta di lavori realizzati con altri appassionati, con cui ho felicemente collaborato, con entusiasmo e soddisfazione!
Con questa premessa non intendo giustificarmi ma la scrivo per far conoscere il senso della mia decisione al lettore, per evitare che le mie creazioni sia considerate 'da poco'.
Per ultimo voglio aggiungere che il mio intento, fin da quando ho iniziato la passione di scrittore/divulgatore, è sempre stato quello di raggiungere la 'massima diffusione' e confido che in questo modo ci riuscirò all'ennesima potenza!
Concludo che così facendo sarà debellata completamente la pirateria dei mie lavori digitali, dato che ho lottato molto contro tale piaga del web ma ahimè senza successo!
Proprio perché i miei scritti sono in costante rilettura vi imploro di visitare il mio sito web www.bertinettobartolomeodavide.it di tanto in tanto, e controllare di essere in possesso dell'ultima versione del 'X' lavoro e non fidarvi troppo dei files provenienti da siti web che non hanno la mia approvazione.

Premessa

Quando una passione nasce da bambino e continua negli anni è molto difficile che svanisca nell'età adulta... E' per me senz'altro il caso del Commodore 64. Un legame affettivo che mi riporta indietro di moltissimi anni facendomi rivivere momenti di mistero legati alla mia infanzia nei confronti dei computer ed in particolare del prodigioso C64 che ai tempi mi sembrava possedesse capacità illimitate!

La tecnologia è progredita in modo esponenziale tra gli anni '80 del secolo scorso e il 2015 in cui scrivo. Tutte le performance informatiche delle macchine di allora, che sembravano straordinarie ora sembrano ridicole sciocchezze. Posso scrivere che non è così dato che in proporzione all'hardware di trenta e più anni orsono i programmatori di allora creavano miracoli veri e propri grazie ad una impressionante conoscenza del computer usato. Mentre i programmatori di oggi non conoscono per nulla quello che succede nel microprocessore che hanno a disposizione per via dei linguaggi altamente slegati dalla macchia in voga attualmente. Ora se la macchina può fare allora si creano dei bei programmi e se l'hardware non ci arriva si aspetta qualche mese che il settore progredisca ulteriormente a livello hardware per far funzionare tutto al meglio. Hai tempi del C64 o dopo con l'Amiga, l'hardware della macchina era costante ma l'impegno dei programmatori riusciva a scavalcare i limiti della macchina per trovare soluzioni altamente ottimizzate per il calcolatore a disposizione. Naturalmente si utilizzava per lo più un linguaggio su misura al computer usato ed era il caso dell'Assembler. Se però ci fissiamo sulla piattaforma del grandioso Commodore 64 tutto si spostava verso il linguaggio macchina(LM) fatto di codici numerici. Bene, questo ebook vuole introdurre l'utente appassionato di programmazione retrò verso l'uso questo speciale strumento per creare il proprio software e comprendere appieno il funzionamento interno di un computer, inserendo solo codici numerici! Se si pensa inoltre che per farlo non servono software aggiuntivi ma soltanto quello che offre il computer all'accensione, direi che tutto diventa fantastico perché attraverso pochi comandi Basic è possibile lavorare in puro linguaggio macchina attraverso il costrutto Read...Data.... Vedremo a breve come riuscirci.

Non si tratta di un revival per smanettoni nostalgici ma di una soluzione ideale rivolta a coloro che voglio diventare o approfondire il loro essere programmatori di computer, arrivando a capire come erano le cose all'inizio dell'era informatica di massa. Anche se per i canoni di oggi può essere un settore superato quello del linguaggio macchina, sarà bello apprendere che si potranno acquisire nuovi schemi di ragionamento in ambito di sviluppo software.

La cpu del CBM 64 è il 6510(parte della famiglia del 6502 usato su svariate vecchie piattaforme). Un processore molto limitato nelle sue istruzioni di base ma che grazie all'ingegno diventa completo perché il programmatore dovrà costruirsi da solo il materiale necessario per usarlo. Basti pensare che il 6510 non dispone di comandi diretti per svolgere operazioni di moltiplicazione e divisione. In più un'ulteriore carenza, come molti altri processori ad 8 bit, è la gestione di valori fino a 256 unità rendendo l'operazione di calcolo più banale un vero 'spacca cervello', che regalerà incredibili soddisfazioni al programmatore intraprendente! Non a caso molte superstar della programmazione su processori più evoluti come la famiglia 68000 e 8086 si sono fatti le ossa con il 6502 per poi continuare a realizzare prodigi software anche nelle era dell'informatica moderna...

Nel manuale che state leggendo potrete comprendere la programmazione in LM in ogni aspetto principale del C64, dalle routine testuali e matematiche, alla gestione della memoria, all'uso delle periferiche di gioco, l'uso della tastiera e manipolazione dei caratteri, la gestione del suono, l'utilizzo degli sprite, per arrivare fino alla gestione della grafica con il caricamento di immagini in formato Koala. In sostanza potrete gestire in ogni aspetto del computer più veduto della storia della portentosa Commodore di quegli anni... Sicuramente una macchina che meriterebbe di essere prodotta ancora oggi per potrebbe essere usata come 'scuola' informatica di base per ogni aspirante che desideri diventare un vero programmatore vecchio stampo, produzione che godrebbe oggi giorno di una grandissima nicchia di mercato vista l'enorme longevità del piccolo 8 bit di casa Commodore.

Chiudo questa premessa aggiungendo che ho iniziato a scrivere questo ebook a tempo perso ogni estate, a partire dal 2006 nel mese di agosto, durante le ore buche delle vacanze estive e in questi giorni l'ho

concludo nel medesimo mese dell'anno 2015. Spero che le pagine qui scritte servano a tenere vivo il Commodre 64 ed a appassionare molti nuovi 'addetti' attraverso i bei emulatori presenti su internet...

Un ringraziamento particolare al mio amico Paolo che fu il primo in quegli anni ottanta, quando eravamo ancora bambini, a permettermi di vedere ed usare il C64 facendo nascere in me la passione per l'informatica.

Impostazione dell'apprendimento

Seguiranno per titoletti i vari esempi di programmazione, che saranno mostrati per via diretta come mia abitudine già in altri lavori. Ciò permetterà all'utente di utilizzare immediatamente il contenuto di ogni sezione. E' comunque fondamentale dopo aver esaminato i lavori, che l'aspirante programmatore modifichi a piacimento la routine esposta in modo da esplorarne ogni aspetto possibile.

Scaricare i sorgenti dei programmi di questo ebook, file d64: Link di download presente al fondo del presente ebook...

Non dovete pensare, abbandonando precocemente l'idea di mettere in pratica i contenuti di questo manuale, dovendo digitare tutto a mano copiando ogni listato. Infatti verso la fine del libro troverete un link di download e relativa chiave di apertura, con l'immagine di un floppy zipppato in formato D64 contenente ogni sorgente qui spiegato.

Aggiungo ancora che il manuale è stato recentemente revisionato per essere agevolmente consultato su tutti i formati di ebook reader. Ho potuto effettuare test sui piccoli cellulari Android con App Kindle installata, sui più grandi tablet da dieci pollici ed ovviamente sul bellissimo Kindle Reader modello base. Come già menzionato, viste le dimensioni variabili dei numerosi prodotti hardware disponibili l'impaginazione potrà subire variazioni, così come la dislocazione delle immagini/dimensione potrà risultare modificata tra i vari dispositivi. E' importante sapere che sarà saggio usare l'indice analitico per raggiungere velocemente l'argomento desiderato contenente la nozione utile. Inoltre le varie immagini potranno essere ingrandite a tutto schermo per una più attenta visione. Sul bellissimo Kindle Reader modello base sarà sufficiente usare il pad per spostare il puntatore sulla fotografia desiderata e confermare per ingrandirla, quindi per ritornare al manuale basterà premere il tasto di ritorno. Ugualmente per raggiungere l'argomento preferito nell'indice bisognerà far ricorso al puntatore... Il discorso è differente per i dispositivi touchscreen, dove con un tocco potrà essere scelta la foto da ingrandire o il tema dell'indice da consultare... Per lo spostamento semplice tra le pagine sarà sufficiente con i dispositivi a tasti premere il bottone giusto per scorrere avanti o indietro, mentre su quelli a tocco basterà scorrere sullo schermo le pagine in avanti od indietro attraverso la punta di un dito. Buona lettura e spero che il mio lavoro, frutto di molto impegno, possa esservi utile.

Mi scuso per i numerosi errori di battitura nelle note dei listati, che sono solo dei promemoria per mantenere traccia del codice e sono serviti a me durante lo sviluppo ma che non incidono in nessun modo sulla corretta esecuzione dei programmi.

Alcuni ringraziamenti bibliografici.

E' doveroso per me menzionare i manuali che hanno aiutato il mio apprendimento del Linguaggio Macchina sul Commodore 64. In particolare il superbo libro di Armando Caiazzo 'Il vero linguaggio macchina del Commodore 64', del quale riporto qui moltissimi listati rielaborati o nel codice o nella descrizione o su entrambi i fronti. Inoltre il libro rivolto al Basic di Stan Krute 'Grafica e suoni con il Commodore 64' dove ho tradotto/snellito alcuni listati significativi in linguaggio macchina per riproporli

nel presente ebook. Ancora la rivista Commodore Computer Club ed in particolare il numero 64 da cui ho tradotto in LM il listato dedicato alla visualizzazione delle immagini in formato Koala. Senza questi testi il linguaggio macchina sarebbe rimasto un argomento oscuro, limitato alla ricopiatura passiva dei listati presentati da terzi sulla ancor intrigante macchina C64 oggi nel 2015!

Argomenti correlati sul C64 dell'autore

Potrete trovare altro materiale collegato alla mia passione retrò delle fantastica piattaforma C64 sul mio sito web:

http://www.bertinettobartolomeodavide.it/programmazione/commodore64/

Programma 01 - stampare un carattere su schermo.

Le linee che saranno descritte di seguito, da 10 a 60 servono al computer per scrivere e leggere in memoria il codice macchina. Quindi si useranno per ogni programma scritto in linguaggio macchina.

10 k=49152

In questa linea, numero 10, viene caricato nella variabile 'k' il primo indirizzo utile di memoria dove si possono iniziare a scrivere i programmi macchina.

Ogni programma scritto in linguaggio macchina, se viene collocato a partire da questo indirizzo(49152) non andrà mai a sovrapporsi con altre routine già presenti in memoria.

20 for n=k to k+10

Il comando 'for' serve per iniziare la lettura(read) e l'inserimento(poke) delle 11 caselle(data) di memoria partendo dall'inidirizzo 49152(compreso, per questo sono 11), contenuto in 'k'.

Ovviamente se i comandi fossero più numerosi non riporteremo più il numero '+10'(per 11 comandi) ma un valore maggiore.

30 read a

Questo comando legge il contenuto della memoria in cui è stato dichiarato un comando(con i data). Inserendolo nella variabile 'a'. Ciclo per ciclo.

40 poke n,a

La linea 40 prende la lettura della memoria, avvenuta con il comando 'read' ed inserita in 'a'. Sarà poi applicata con 'poke' all'indirizzo 'n' con il relativo comando contento in 'a'.

50 next n

'Next' circoscrive il ciclo ai comandi precedentemente riportati.

60 sys k

Ora, terminato il ciclo, con il comando 'sys', diamo il via alla lettura ed esecuzione della routine memorizzata partendo dall'indirizzo 'k'(49152).

70 data 169,1

I numeri contenuti dopo 'data' indicano: accumulatore(169), valore(1). Utilizzare il comando 169, che rappresenta l'accumulatore, è un passo obbligato per caricare un dato in memoria. In quanto non è possibile farlo direttamente. Il secondo valore rappresenta la lettera 'A' maiuscola del codice ASCII. Se volessimo visualizzare un'altro carattere dovremmo inserire un valore diverso nell'accumulatore.

80 data 141,0,4

Il valore 141 immette il contenuto dell'accumulatore in una determinata posizione della memoria del computer. In questo caso nella memoria video testuale. La sezione indicata è la riga 0 di pagina 4. La memoria video del Commodore 64 in modalità testo è composta da 1000 caratteri(40x25). Quindi da altrettante caselle di memoria: da riga 0 pagina 4(angolo in alto a sinistra) a riga 231 di pagina 7(angolo in basso a destra).

La memoria del Commodore 64 è 'grande'(per quei tempi) 64 Kb, che corrispondono a 65536 bytes (caratteri). Che corrispondono a 256 pagine da 256 righe.

Nel caso della memoria video testuale si parte dall'indirizzo 1024 (256x4) fino a 2023 (256x7+231). Corrispondenti esattamente ai mille caratteri che compongono lo schermo.

90 data 169,7

Ora inseriremo un nuovo valore nell'accumulatore, che indica al computer il colore del carattere(7=giallo).

100 data 141,0,216

Sempre grazie al comando 141 assegneremo il colore giallo al carattere spostando il valore 7, presente nell'accumulatore all'indirizzo 55296 (riga 0 pagina 216 ovvero 256*216+0). Se volessimo assegnare un colore diverso è sufficiente cambiare il valore dell'accumulatore con un numero da 0 a 15. Visto che il commodore 64 possiede una palette di 16 colori.

110 data 96

L'ultima riga indica al computer di restituire il controllo al sistema operativo (Basic).

Il commodore 64 non dispone solamente dell'accumulatore per caricare i dati in memoria. E esistono due indici, chiamati "x' e 'y', che possono fare la stessa cosa. Così chiameremo l'accumulatore 'a' e gli indici 'x'

e ' y', per risparmiare tempo.

I rispettivi comandi sono:

169,n ---> carica in 'a' un numero.

162,n ---> carica in 'x' un numero.

160,n ---> carica in 'y' un numero.

Quindi i rispettivi ordini di stampa sono:

141,r,p ---> stampa il contenuto di 'a' ad una determinata riga(r) e pagina(p) della memoria.

142,r,p ---> stampa il contenuto di 'x' ad una determinata riga e pagina della memoria.

140,r,p ---> stampa il contenuto di 'y' ad una determinata riga e pagina della memoria.

La deduzione logica se vogliamo stampare la solita lettera 'A' su schermo ma utilizzando gli indici anziché che l'accumulatore. Come abbiamo già visto nell'esempio precedente, dovremo inserire i seguenti numeri nei 'data':

INDICE X

162,1

142,0,4

162,7

142,0,216

96

INDICE Y

160,1

140,0,4

160,7

140,0,216

96

In questo primo passaggio abbiamo visto come stampare su schermo un semplice 'A' di colore bianco utilizzando solo dei codici numerici in linguaggio macchina, grazie alle poche righe del caricatore basic necessario. Senza alcuno strumento aggiuntivo a parte il materiale che ci fornisce il computer alla sua accensione!

PROGRAMMA 02 - Caricare un dato da un indirizzo.

Da questo secondo programma in avanti non riporterò più le prime linee Basic necessarie per caricare il listato in linguaggio macchina nella memoria, dato che saranno sempre le stesse del primo esempio salvo per il numero di locazioni. Inoltre saranno riportati solamente i codici numerici senza il comando 'data', per una questione di comodità di sintesi.

Spiegherò come caricare un dato, che magari non conosciamo, da un determinato indirizzo di memoria, visto che esistono dei comandi speciali per farlo:

173,r,p ---> legge un valore contenuto in una determinata riga e pagina della memoria e lo inserisce in 'a'.

174,r,p ---> legge un valore contenuto in una determinata riga e pagina della memoria e lo inserisce in 'x'.

172,r,p ---> legge un valore contenuto in una determinata riga e pagina della memoria e lo inserisce in 'y'.

Analizziamo i seguenti valori che compongono il programma, da inserire nei 'data':

169,1 ---> Carica il valore 1, cioè la lettera 'A' in ASCII in accumulatore.

141,5,193 inserisce il comando precedente nella locazione di memoria, riga 5 e pagina 193.

174,5,193 legge dalla memoria il valore contenuto nella riga 5 a pagina 193. Il contenuto è posizionato in 'x'. Inserito con il comando precedente.

142,0,4 preleva il valore ora contenuto in 'x' e lo stampa nella riga 0 di pagina 4 della memoria video del testo.

96 Il controllo torna al Basic.

Con questo esempio abbiamo visualizzato una 'A' nell'angolo alto sinistro dello schermo che era già contenuta in memoria. Sono stati usati sia l'accumulatore che l'indice X, spostandone il contenuto da uno all'altro, prima di essere visualizzato.

PROGRAMMA 03 - Posizionare un carattere sullo schermo

Cercheremo di capire come posizionare un carattere su schermo con un particolare ciclo.

Vediamo come:

160,10 ---> carichiamo il valore 10 in 'y', che indicherà lo spostamento del carattere(Indicato più avanti nel programma) per tutte quelle posizioni della riga di memoria.

169,1 ---> carichiamo il codice video in 'a' della lettera 'A'. Cioè '1'.

153,0,4 ---> indica di stampare il contenuto dell'accumulatore (quindi 1) tante volte quante sono indicate nell'indice Y (cioé 10). Perciò il codice 153 indica un ciclo sull' indice Y. I valori '0' e '4' indicano rispettivamente la riga e la pagina da cui iniziare il conteggio.

96 come al solito questo comando chiude il programma.

Se volessimo creare un ciclo sull' indice X dovremo sostituire i valori 160 con 162 e 153 con 157. Il resto non cambia. Vediamo la versione differente sull'indice X:

162,10

169,1

157,0,4

96

PROGRAMMA 04 - Stampare 10 caratteri con un ciclo

Approfondiamo la tecnica di esecuzione dei cicli, senza però analizzarla nella sua completezza.

In questo programma variamo leggermente il contenuto del programma precedente, inserendo nuovi comandi.

160,10 ---> Indica il numero di volte di ripetizione del ciclo, inserito nell'indice Y. Quindi quante 'A' saranno visualizzate su schermo.

169,1 ---> Caricamento nell'accumulatore del valore '1' che indica la lettera 'A'.

153,255,3 ---> stampa la 'A' partendo dall'indirizzo 1023(256*3+255) più il valore di 'y'.

136 ---> Diminuisce di un'unità il valore di 'y'

208,250 ---> il comando 208 indica al computer che si deve effettuare un salto all'indietro di un certo quantitativo di istruzioni, in questo caso '5' (255-250). Se fosse indicato il valore 255 non verrebbe effettuato alcun salto. Il salto all'indietro sarà effettuato fin tanto che l'indice Y o X non conterrà il valore '0'. Quando 'y' o 'x' arriverà a '0' il programma continuerà in avanti. In questo caso specifico torneremo all'istruzione '153', contando a ritroso tutti i numeri presenti nei data partendo dall'istruzione di salto(208). Così faremo: 208(5), 136(4), 3(3), 255(2), 155(1).

96 ---> fine

Se volessimo fare la stessa cosa con l'indice X, allora dovremo digitare: 162 al posto di 160, 157 al posto di 153 e 202 invece di 136. Otterremo la seguente codifica:

160,10

169,1

157,255,3

202

208,250

96

PROGRAMMA 05 - leggere dei caratteri diversi dalla memoria

Questo interessante listato ci permetterà di scrivere in memoria la parola 'ciao' e di leggerla visualizzandola su schermo.

160,4 ---> Carica il numero 4, dato che intendiamo leggere 4 caratteri dalla memoria. Se volessimo leggere dalla memoria un numero maggiore di caratteri dovremmo inserire un valore tanto più grande quanti sono i caratteri da leggere.

185,11,192 ---> Questo comando inizia a leggere la memoria da una determinata riga e pagina, più il valore indicato nell'indice Y. I '4' caratteri che intendiamo leggere saranno situati proprio nelle caselle di memoria: 12,13,14,15. Esattamente il numero di caselle indicato dall'indice Y. Attenzione però al posizionamento del programma esecutivo in memoria che va dalla riga 0 e pagina 192 alla riga 11 sempre in pagina 192. Quindi ogni eventuale carattere da leggere in memoria deve essere posizionato in un indirizzo diverso dal programma principale, per evitare conflitti. Tutto quello che viene letto dal comando '185' viene caricato nell'accumulatore. In questo caso subito dopo il valore 96, al termine del programma...

153,255,3 ---> Stampa il contenuto dell'accumulatore, letto dal precedente comando '185'. Il valore in questo caso sarà stampato nella memoria video subito dopo la riga 255, pagina 3, più 4 caselle visto che l'indice Y ha ancora valore '4'.

136 ---> diminuisce di una unità il valore d 'y'.

208,247 ---> Se 'y' non è ancora arrivato a '0' esegue un salto di 8 istruzioni (255-8=247), tornando al comando di lettura '185'. Questa volta '185' non leggerà più dalla casella '11+4' ma da '11+3', visto che 'y' è stato decrementato da '4' a '3'. Appena 'y' diventerà '0' il comando '208' non sarà più eseguito ed il programma proseguirà.

96 ---> torna al Basic

3,9,1,15 ---> sono i codici video (ASCII) per visualizzare i caratteri 'C,I,A,O'.

Se vogliamo caricare nell'accumulatore dei dati presenti in memoria ma riferendoci all'indice X, dovremo utilizzare il comando '189' anziché '185'.

PROGRAMMA 06 - uso della memoria in modo indiretto

Prima di passare al programma in questione e bene approfondire l'utilizzo della pagina zero.

La pagina zero è utilizzata dal computer stesso per il funzionamento del Basic e del Kernel. Il sistema operativo del Commodore 64 ha la priorità di utilizzo sulla prima pagina di memoria semplicemente perché è la più veloce.

Per la sua velocità questa pagina è molto ambita dai programmatori in linguaggio macchina. Possiamo scegliere di usare la pagina zero con alcuni compromessi:

· Utilizzare obbligatoriamente dei comandi speciali, che sono 39.

· Se scegliamo di utilizzare contemporaneamente sia il sistema operativo che il linguaggio macchina, avremo accesso a solo 5 righi sicuri(2, 251, 252, 253, 254 e 255)

· Utilizzando solamente il linguaggio macchina senza il Basic potremmo accedere dalla riga 7 alla 143. Mentre senza il kernel dalla 144 alla 250.

Dovrete valutare cosa è meglio per i vostri scopi. Se vi serve molta velocità di calcolo, come in un video gioco allora ne avrete assoluta necessità. Visto che nella pagina zero la velocità dei cicli raddoppia, dato che le istruzioni sono formate da 1 valore e mai da 2!

Vediamo i comandi già visti fino qui per le altre pagine della memoria ma funzionanti nella pagina zero(la prima colonna indica la pagina zero):

· 165,n = 173,n,n

· 166,n = 174,n,n

· 164,n = 172,n,n

· 133,n = 141,n,n

· 134,n = 142,n,n

· 132,n = 140,n,n

· 181,n = 189,n,n

· 149,n = 157,n,n

· 180,n = 188,n,n

· 182,n = 190,n,n

· 148,n = solo in pagina zero, stampa in memoria 'y' + riga + 'x'

· 150,n = solo in pagina zero, stampa in memoria 'x' + riga + 'y'

Avrete notato che i comandi di pagina zero possiedono solo un'istruzione. Bene, questo è il motivo per cui questa pagina è molto più veloce rispetto al resto della memoria, i comandi sono più brevi, dato che il computer da per scontato l'indirizzo di pagina. Che è sempre zero.

Ora possiamo esaminare un nuovo programma. Abbiamo la possibilità, utilizzando la pagina zero, di stampare qualcosa (una 'A') in qualunque parte della memoria del computer, senza però indicarne direttamente le coordinate.

169,0 ---> inseriamo nell'accumulatore il valore '0', che sarà poi la riga '0' di memoria in cui stamperemo.

133,251 ---> stampa lo '0' contenuto in 'a' nella riga '251' di pagina zero.

169,4 ---> carica nell'accumulatore il numero '4', che indicherà al computer la pagina 4 in cui stamperemo.

133,252 ---> stampa il '4' già memorizzato in 'a' nella riga '252' di pagina zero.

169,1 ---> carica il valore della lettera 'A', che stamperemo indirettamente alla riga '0' di pagina '4'

162,11 ---> carica il numero '11' in 'x'. Questo valore sarà sommato a quello del comando successivo per andare a leggere le coordinate di memoria in cui si trova il carattere 'A' e stamparlo.

129,240 ---> Il comando '129' indica di andare all'indirizzo '240' + '11' di 'x', che sono uguali a '251'. Indirizzo in cui si trova la riga di memoria in cui stampare il carattere. Il comando '129' a questo punto andrà a leggere la casella di memoria successiva, la '252', in cui è contenuta la pagina di memoria in cui stampare il carattere. Finalmente ottenuto l'indirizzo di memoria, il computer provvederà a stampare il carattere 'A'.

96 ---> Restituisce il controllo al Basic

Questo programma utilizza tutti comandi dedicati alla pagina zero.

Se volessimo indicizzarlo per 'Y' le cose cambiano leggermente:

Il sostituiamo valore '162' con '160' utilizzando 'y'. Useremo l'istruzione '145' anziché la '129'. L'istruzione '145' è leggermente diversa da quella per 'x'. Ora indicheremo direttamente la riga in cui andare a leggere(in questo caso 251) e l'indice 'y' servirà per spostare il contenuto a '251+10'. Il comando andrà poi a cercare la pagina successiva a '251', cioè '252'. Il risultato sarà una 'A' spostata di '10' spazi su schermo. Quindi '145' stampa direttamente il valore presente nell'accumulatore in una data locazione di memoria.

PROGRAMMA 07 - stampa indiretta

Il nuovo listato ci permette di caricare un valore nell'accumulatore, partendo da delle coordinate di memoria fornite indirettamente, senza però stamparlo direttamente ad un certo indirizzo. Infatti la solita 'A' verrà stampata con un'altro comando.

169,16 ---> carica la coordinata di riga nell'accumulatore, dove si trova la 'A'

133,251 ---> memorizza l'indirizzo di riga alla locazione 251 di pagina zero

169,192 ---> carica la coordinata di pagina nell'accumulatore, dove si trova la lettera 'A'

133,252 ---> memorizza l'indirizzo di pagina alla locazione 252 di pagina zero

162,11 ---> carica nell'indice X il valore '11', che servirà per indicizzare l'istruzione successiva.

161,240 ---> va alla riga '240' più 'x', quindi '240+11=251'. '251' è la locazione dove trovare la riga in cui è collocato il carattere da immagazzinare nell'accumulatore. Subito dopo legge la casella successiva '252', trovando la pagina... Il risultato va nell'accumulatore(A).

141,0,4 ---> visualizza il contenuto di 'a' su schermo.

96 ---> fine

1 ---> carattere 'A' da leggere alla locazione '16,192' ed inserito nell'accumulatore.

Vediamo ora come utilizzare l'indice Y, questa volta le cose sono diverse:

169,0 ---> carica '0' in 'a'

133,251 ---> colloca '0' a '251' di pag. Zero

169,192 ---> carica '192' in 'a'

133,252 ---> colloca '192' a '252' di pag. Zero

160,16 ---> carica '16' in 'y'

177,251 ---> va a '251' e legge la riga di partenza(0). Alla locazione successiva(252) legge la pagina(192). Ora aggiunge '16' locazioni di 'y', quindi '0+16', dove troverà il dato da inserire in 'a'

141,0,4 ---> stampa il contenuto di 'a' su schermo

96 ---> fine

1 ---> la 'A' alla riga '16' di pagina '192'

PROGRAMMA 08 - ciclo for...next ed if...then in LM

Prima di entrare nel vivo dell'argomento ciclo è bene analizzare il cosiddetto 'registro di stato'.

Il registro di stato è composto da 8 bit(otto locazioni), estremamente collegato a moltissime operazioni in ogni elaboratore, in particolare alle istruzioni di confronto e salto.

I valori di questo registro sono binari, quindi espressi con '1' e '0'. Vediamo:

7 6 5 4 3 2 1 0 ---> Bit

N V - B D I Z C ---> Riferimento

Legenda:

C ---> Carry (riporto e prestito)

Z ---> Zero (risultato uguale a '0')

I ---> Interrupt (interruzione Hardware)

D ---> Decimal (modo decimale)

B ---> BRK (Interruzione sortware)

- ---> Bit 5 non usato

V ---> Overflow (superamento capacità)

N ---> Negativ (risultato negativo o superiore a 127)

Molte 'decisioni' del computer vengono prese in base allo stato di questo registro.

Finalmente possiamo passare alla programmazione pratica:

Creiamo un listato LM che in Basic corrisponderebbe a questo:

· For n=1 to 10: print "a": next

Il risultato di questi comandi visualizza su schermo la successione di lettere A.

169,1 ---> carica in 'a' un '1', che è il codice video dalla lettera 'A' da visualizzare.

160,0 ---> Carica '0' in 'y' cioè il valore di partenza. L'indice Y ha la funzione di contatore.

153,0,4 ---> stampa il risultato a partire dall'indirizzo '0,4'(Su schermo in questo caso) più il valore di 'y', contenuto nell'accumulatore'.

200 ---> aumenta di un'unità il valore di 'y'.

192,10 ---> Confronta il valore di 'y' con '10'. Se 'y' è uguale a '10' allora 'z' nel registro di stato diventa '1'.

208,248 ---> istruzione di salto è condizionata dal valore di 'z' nel registro di stato. Se 'z' è uguale a '0' allora esegue il salto indietro di sette istruzioni (255-248), andando a ripetere partendo dal comando '153'. Quando 'z' sarà uguale a '1' allora il ciclo finirà e si passerà all'istruzione successiva.

96 ---> fine programma.

Questo listato è il primo passo per creare programmi che facciano qualcosa.

Aggiungiamo qualche parola per utilizzare le istruzioni condizionali if...then:

Confronto con 'a', alcuni comandi:

201,n ---> confronta l'accumulatore con l'operando

197,n ---> confronta l'accumulatore con il contenuto di una riga a pagina zero

213,n,n ---> confronta l'accumulatore con il contenuto ad un indirizzo specificato dagli operandi.

Confronto con 'x', alcuni comandi:

224,n ---> Confronta l'indice X con l'operando

228,n ---> confronta l'indice X con il contenuto di una riga a pagina zero

236,n,n ---> confronta l'indice X con il contenuto ad un indirizzo specificato dagli operandi.

Confronto con 'y', alcuni comandi:

192,n ---> confronta l'indice Y con l'operando

196,n ---> confronta l'indice Y con il contenuto di una riga a pagina zero

204,n,n ---> confronta l'indice y con il contenuto ad un indirizzo specificato dagli operandi.

Il registro di stato(bit N,Z,C) viene modificato in questo modo a seconda del tipo di confronto:

A,X,Y - numero ci confronto - bit N - bit Z - bit C

maggiore minore 0 0 0

uguale uguale 0 1 1

minore maggiore 1 0 0

Eseguono il salto, a seconda che un certo bit del registro di stato assuma la posizione '0' o '1':

16,n ---> esegue il salto solo se N=0

48,n ---> esegue il salto solo se N=1

208,n ---> esegue il salto solo se Z=0

240,n ---> esegue il salto solo se Z=1

144,n ---> esegue il salto solo se C=0

176,n ---> esegue il salto solo se C=1

80,n ---> esegue il salto solo se V=0

112 ---> esegue il salto solo se V=1

Nella programmazione LM del Commodore 64 inserire la condizione if...then riprende gran parte del codice necessario per creare un ciclo for...next. Perciò datevi da fare e provate a realizzare qualche programmino!

Riporto un paio di righe Basic che dovrete aggiungere alla fine dei dei listati LM per sapere la condizione in cui si trova il registro di stato:

1000 print" N V – B D I Z C": print: a = peek(783)

1010 for n=7 to 0 step -1: b=int(a/2^n): print b;: a = a-b*2^n: next

PS: La riga 1000 e 1010 sono in Basic per maggior chiarezza nella lettura del programma da parte dell'utente. Non sono necessarie a programma LM concluso, quindi potranno essere cancellate.

Vediamo ora con che comandi possiamo modificare volontariamente i bit del registro di stato a nostro piacimento:

24 ---> mette a 0 il bit C

56 ---> mette a 1 il bit C

88 ---> mette a 0 il bit I

120 ---> mette a 0 il bit I

216 ---> mette a 0 il bit D

248 ---> mette a 1 il bit D

184 ---> mette a 0 il bit V

I più attenti avranno notato che non sono disponibili i comandi necessari per modificare tutti i bit di stato. Questo è vero, però grazie a questi pochi bit modificati potremo destreggiarci nella maggior parte delle operazioni. Se poi avessimo necessità di modificare gli altri bit riusciremo a farlo indirettamente ricreando delle situazioni particolari con altri comandi...

Nota:

Con il comando '248' possiamo mettere il bit 'D' a '1'. Questo ci permetterà di far ragionare il computer in decimale ma **codificato binario**. Significa che il computer trasformerà ogni cifra(da 0 a 9) in binario e non il numero completo(da 0 a 255).

Avremo:

Un nibble (4 bit):

bit ---> 3 2 1 0 es: 3 2 1 0

Valore ---> 8 4 2 1 es: 8 4 2 1

stato ---> 0 0 0 0 es: 0 1 1 0 ---> risultato: 6

Spero di essere stato abbastanza chiaro!

PROGRAMMA 09 – istruzioni di salto gosub...return e goto

I comandi 'gosub' e 'return' servono per raggiungere un determinato indirizzo e continuare a leggere di lì in poi e ritornare al comando successivo al 'gosub' appena si incontra 'return'. E' così nel Basic come lo è in linguaggio macchina.

Vediamo in pratica:

Stabiliamo innanzi tutto l'indirizzo in cui saltare per raggiungere i dati che ci interessano. Quindi scriviamo in questo modo:

10 k=52992: for n=k to k+5

20 read a: poke n,a: next

Indica che intendiamo collocare i dati a partire da '52992' ovvero riga '0' di pagina '207'. Lì inseriremo '6' comandi.

30 data 169,2

Carica il codice video '2' del carattere 'B' nell'accumulatore.

40 data 141,0,4

Stampa il contenuto dell'accumulatore (la lettera 'B') nell'area di memoria video, cioè riga '0' di pagina '4'.

50 data 96

In questo caso il comando '96' non ha più la funzione di restituire il controllo al Basic ma serve per ritornare al comando successivo al 'gosub' che ci ha fatto saltare in questa parte della memoria.

60 k=49152: for n=k to k+8

70 read a: poke n,a: next: sys 49152

Nella seconda parte del programma provvederemo ad inserire i '9' dati dalla solita locazione '49152' (riga 0 di pagina 192). Qui però è contenuto il comando 'sys 49152' che lancerà il programma LM.

80 data 169,1

Carica il codice video '1' del carattere 'A' nell'accumulatore.

90 data 141,1,4

Stampa il contenuto dell'accumulatore (la lettera 'A') nell'area di memoria video, cioè riga '1' di pagina '4'.

100 data 32,0,207

Salteremo, come già accennato, alla riga '0' di pagina '207'. Leggendo così il segmento di programma che abbiamo inserito con le linee da '10' a '50'. Quando incontreremo il comando equivalente a 'return' (96), il computer tornerà a leggere dalla linea Basic '110'. Incontrando nuovamente '96' che questa volta ci farà tornare al Basic.

110 data 96

Questa volta torniamo al Basic

Notate che il programma principale viene lanciato dall'indirizzo '49152', saltando poi all'indirizzo '52992'.

La parte principale del programma è stata inserita per ultima in quanto se l'avessimo inserita prima, il via dato dal comando 'sys' non avrebbe più permesso il caricamento in memoria dei dati successivi...

Vedremo come risultato visualizzate una 'B' ed un 'A' nell'angolo alto a sinistra dello schermo.

Modifichiamo ora, il listato appena spiegato per utilizzare l'istruzione 'goto' del Basic ma in LM.

Vediamo il listato per intero ma leggermente modificato:

10 k=52992: for n=k to k+5

20 read a: poke n,a: next

30 data 169,2

40 data 141,0,4

50 data 96

60 k=49152: for n=k to k+7

70 read a: poke n,a: next: sys 49152

80 data 169,1

90 data 141,1,4

100 data 76,0,207

PROGRAMMA 10 – Operazioni aritmetiche

Alcuni comandi si somma e sottrazione di un'unità:

232 – incrementa di una unità il valore di X

202 – decrementa di una unità il valore di X

200 – incrementa di una unità il valore di Y

136 – decrementa di una unità il valore di Y

230,n - incrementa di una unità il valore di pag. zero alla riga indicata dall'operando.

PS: attenzione a porre sempre a '0' il bit del registro di stato 'C'. Dato che il Commodore 64 ragiona ad 8 bit, il massimo numero che può esprimere in una sola volta è 256 (2^8), superata questa soglia viene indicato un riporto e 'C' diventa '1'. Dopo questo si prosegue il conteggio al byte successivo... Il comando che pone 'C' a '0' è '24', mentre quello che lo pone a '1' è il '56'.

Analizziamo un piccolo listato per il comando 232:

24 ---> pone a '0' il valore sul riporto del bit 'C' del registro di stato.

162,0 ---> carica '0' nell'indice X

232 ---> incrementa di un'unità il l'indice X

134,251 ---> stampa alla riga '251' il contenuto di 'x'

96 ---> chiude il programma.

Per visualizzare quello che abbiamo fatto con questo listato è sufficiente digitare in Basic:

print peek(251) ---> va a leggere quello che è presente all'interno dell'indirizzo 251 di pagina zero.

Alcuni comandi di addizione:

105,n – aggiunge un numero indicato dall'operando nell'accumulatore.

101,n – somma un valore indicato dall'accumulatore a pagina zero alla riga indicata dall'operando.

117,n – somma al valore dell'accumulatore, il contenuto di pagina zero, che viene raggiunta partendo dal valore indicato dall'operando più l'indice X.

109.n.n – somma il valore contenuto in una locazione indicata dagli operanti al valore contenuto nell'accumulatore.

PS: nella somma normalmente si parla di riporto, quindi 'C' deve essere '0' prima dell'operazione.

Analizziamo un piccolo listato per il comando 105:

24 ---> pone a '0' il valore sul riporto del bit 'C' del registro di stato.

169,0 ---> carica '0' nell'accumulatore

105,10 ---> incrementa di '10' il contenuto dell'accumulatore

133,251 ---> stampa alla riga '251' il contenuto di 'a'

96 ---> chiude il programma.

Per visualizzare quello che abbiamo fatto con questo listato è sufficiente digitare in Basic:

print peek(251) ---> va a leggere quello che è presente all'interno dell'indirizzo 251 di pagina zero.

Alcuni comandi di sottrazione:

233,n – sottrae un numero indicato dall'operando nell'accumulatore.

229,n – sottrae un valore indicato dall'accumulatore a pagina zero alla riga indicata dall'operando.

245,n – sottrae al valore dell'accumulatore, il contenuto di pagina zero, che viene raggiunta partendo dal valore indicato dall'operando più l'indice X.

237.n.n – sottrae il valore contenuto in una locazione indicata dagli operanti al valore contenuto nell'accumulatore.

PS: nella sottrazione normalmente si parla di prestito, quindi 'C' deve essere '1' prima dell'operazione.

Analizziamo un piccolo listato per il comando 233:

56 ---> pone a '1' il valore sul prestito del bit 'C' del registro di stato.

169,10 ---> carica '10' nell'accumulatore

233,5 ---> sottrae '5' al contenuto dell'accumulatore

133,251 ---> stampa alla riga '251' il contenuto di 'a'

96 ---> chiude il programma.

Per visualizzare quello che abbiamo fatto con questo listato è sufficiente digitare in Basic:

print peek(251) ---> va a leggere quello che è presente all'interno dell'indirizzo 251 di pagina zero.

Il microprocessore 6510 non dispone di istruzioni per effettuare operazioni di moltiplicazione, divisione, ecc... ma solo di somma e sottrazione. Si possono creare delle routine, che partendo da queste due operazioni, possono elaborare qualsiasi altra operazione matematica.

PROGRAMMA 11 - comandi logici: AND, OR e OR esclusivo.

Passaggio da decimale a binario:

I numeri decimali, com'è noto, sono composti da 10 cifre(0,1,2,3,4,5,6,7,8,9). Mentre quelli binari da 2(1,0). Con entrambe le numerazioni è possibile scrivere qualunque numero... I computer 'ragionano' in binario e quando viene indicato un numero decimale sono eseguite delle conversioni, più o meno immediate.

Per trasformare un numero decimale in binario e vice versa si deve pensare a delle potenze di 2. Quindi:

dovendo convertire il numero 8 in binario penseremo in questo modo ---> 2^3 + 0^2 + 0^1 + 0^0 ---> 1 0 0 0

dovendo convertire il numero 10 in binario penseremo in questo modo ---> 2^3 + 0^2 + 2^1 + 0^0 ---> 1 0 1 0

dovendo convertire il numero 1 in binario penseremo in questo modo ---> 0^3 + 0^2 + 0^1 + 2^0 ---> 0 0 0 1

dovendo convertire il numero 15 in binario penseremo in questo modo ---> 2^3 + 2^2 + 2^1 + 2^0 ---> 1 1 1 1

Riporto un piccola tabella di conversione riferita al byte (8bit):

byte

bit: 7 – 6 – 5 – 4 – 3 – 2 – 1 – 0

dec: 128 – 64 – 32 – 16 – 8 - 4 - 2 – 1

Sommando i valori decimali indicati potremmo ricavare qualsiasi numero da 0 a 255. Associando poi la notazione binaria per la conversione.

Esempio:

byte
bit: 7 – 6 – 5 – 4 – 3 – 2 – 1 – 0

dec: 128 – 64 – 32 – 16 – 8 - 4 - 2 – 1

bin: 1 - 0 - 0 - 1 - 0 – 0 – 0 – 1 ---> 128(bit 7) + 0(bit 6) + 0(bit 5) + 16(bit 4) + 0(bit 3) + 0(bit 2) + 0(bit 1) + 1(bit 0) = 145 in decimale

Questa parte sulla conversione in binario può apparire noiosa ma se volete programmare in LM, la sua comprensione rappresenta un passo obbligato! Come stiamo per vedere con gli operatori logici...

Gli operatori logici sono utilizzati molto spesso con una funzione di 'filtro'. Cioè viene forzato un byte per variarne il contenuto binario a piacimento in esso presente(sostituire gli '0' in'1' a piacimento).

AND LOGICO:

Mette a confronto gli otto bit contenuti in una locazione di memoria con l'accumulatore

Esempio 1:

Contenuto locazione 251 = decimale 48 –> binario 0 0 1 1 0 0 0 0

Contenuto accumulatore = decimale 80 -> binario 0 1 0 1 0 0 0 0

Risultato: Avremo AND logico = decimale 16 -> binario 0 0 0 1 0 0 0 0

Esempio 2:

Contenuto locazione 251 = decimale 15 –> binario 0 0 0 0 1 1 1 1

Contenuto accumulatore = decimale 247 -> binario 1 1 1 1 0 1 1 1

Risultato: Avremo AND logico = decimale 7 -> binario 0 0 0 0 0 1 1 1

Da questi due esempi possiamo capire che il comando AND confronta due byte in memoria e pone il risultato a '1' quando i bit rapportati sono entrambi '1' mentre se sono '0' e '0' o '1' e '0' il risultato sarà '0'.

Esaminiamo il listato relativo all'esempio 2:

169,15 ---> carica '15' in 'a'

133,251 ---> il contenuto di 'a' viene posto alla locazione '251'

169,247 ---> '247' in 'a' è il così detto 'filtro' su cui si eseguirà il confronto.

37,251 ---> '37' è il comando corrispondente all'AND che confronterà il 'filtro(247)' con il contenuto della locazione '251(15)'. Il risultato viene posto in 'a', sostituendone il precedente contenuto.

133,251 ---> il risultato '7' è posto in locazione '251'

96 ---> fine

Per verificare il risultato del programma è sufficiente digitare una linea Basic che visualizza il contenuto della locazione '251':

PRINT PEEK(251)

Che visualizzerà il risultato decimale '7' ---> binario '00000111'.

Vediamo alcuni comandi per l'AND:

45,n,n ---> Esegue l'AND sulla locazione indicata dagli operandi.

41,n ---> Esegue l'AND con il contenuto dello stesso accumulatore.

37,n ---> Esegue l'AND con il contenuto di una locazione di pagina zero.

OR LOGICO:

Mette a confronto gli otto bit contenuti in una locazione di memoria con l'accumulatore

Esempio 1:

Contenuto locazione 251 = decimale 6 –> binario 0 0 0 0 0 1 1 0

Contenuto accumulatore = decimale 3 -> binario 0 0 0 0 0 0 1 1

Risultato: Avremo OR logico = decimale 7 -> binario 0 0 0 0 0 1 1 1

Esempio 2:

Contenuto locazione 251 = decimale 7 –> binario 0 0 0 0 0 1 1 1

Contenuto accumulatore = decimale 8 -> binario 0 0 0 0 1 0 0 0

Risultato: Avremo OR logico = decimale 15 -> binario 0 0 0 0 1 1 1 1

Da questi due esempi possiamo capire che il comando OR LOGICO confronta due byte in memoria e pone il risultato a '1' quando i bit rapportati sono entrambi '1' o '1' e '0' mentre se sono '0' e '0' il risultato sarà '0'.

Esaminiamo il listato relativo all'esempio 2:

169,7 ---> carica '7' in 'a'

133,251 ---> il contenuto di 'a' viene posto alla locazione '251'

169,8 ---> '8' in 'a' è il così detto 'filtro' su cui si eseguirà il confronto.

5,251 ---> '5' è il comando corrispondente all'OR che confronterà il 'filtro(8)' con il contenuto della locazione '251(7)'. Il risultato viene posto in 'a', sostituendone il precedente contenuto.

133,251 ---> il risultato '15' è posto in locazione '251'

96 ---> fine

Per verificare il risultato del programma è sufficiente digitare una linea Basic che visualizza il contenuto della locazione '251':

PRINT PEEK(251)

Che visualizzerà il risultato decimale '15' ---> binario '00001111'.

Vediamo alcuni comandi per l'OR:

13,n,n ---> Esegue l'OR con la locazione indicata dagli operandi.

9,n ---> Esegue l'OR con il contenuto dello stesso accumulatore.

5,n ---> Esegue l'OR con il contenuto di una locazione di pagina zero.

OR ESCLUSIVO:

Mette a confronto gli otto bit contenuti in una locazione di memoria con l'accumulatore

Esempio 1:

Contenuto locazione 251 = decimale 12 –> binario 0 0 0 0 1 1 0 0

Contenuto accumulatore = decimale 10 -> binario 0 0 0 0 1 0 1 0

Risultato: Avremo OR esclusivo = decimale 6 -> binario 0 0 0 0 0 1 1 0

Esempio 2:

Contenuto locazione 53269 = decimale 240 –> binario 1 1 1 1 0 0 0 0

Contenuto accumulatore = decimale 255 -> binario 1 1 1 1 1 1 1 1

Risultato: Avremo OR esclusivo = decimale 15 -> binario 0 0 0 0 1 1 1 1

Da questi due esempi possiamo capire che il comando OR ESCLUSIVO confronta due byte in memoria e pone il risultato a '0' quando i bit rapportati sono entrambi '1', mentre se sono '0' e '0' o '1' e '0' il risultato sarà '0'.

Esaminiamo il listato relativo all'esempio 2:

169,240 ---> carica '240' in 'a'

141,21,208 ---> il contenuto di 'a' viene posto alla locazione '53269(riga 21 di pagina 208)'

169,255 ---> '255' in 'a' è il così detto 'filtro' su cui si eseguirà il confronto.

77,21,208 ---> '77' è il comando corrispondente all'OR ESCLUSIVO che confronterà il 'filtro(255)' con il contenuto della locazione '53269(240)'. Il risultato viene posto in 'a', sostituendone il precedente contenuto.

141,21,208 ---> il risultato '15' è posto in locazione '53269', sostituendone il precedente contenuto.

96 ---> fine

Per verificare il risultato del programma è sufficiente digitare una linea Basic che visualizza il contenuto della locazione '53269':

PRINT PEEK(53269)

Che visualizzerà il risultato decimale '15' ---> binario '00001111'.

Vediamo alcuni comandi per l'OR ESCLUSIVO:

77,n,n ---> Esegue l'OR ESCLUSIVO con la locazione indicata dagli operandi.

73,n ---> Esegue l'OR ESCLUSIVO con il contenuto dello stesso accumulatore.

69,n ---> Esegue l'OR ESCLUSIVO con il contenuto di una locazione di pagina zero.

PROGRAMMA 12 – comandi di scivolamento Dx e Sx, rotazione Dx e Sx e trasferimento.

Vediamo ora un gruppo di istruzioni legate alla modifica dei bit, utilizzate prevalentemente per effettuare operazioni matematiche. Facciamo riferimento come al solito al registro di stato ed in particolare al bit 0 'C' legato al prestito e al riporto.

Scivolamento a sinistra.

'C' <--- bit 7 <--- bit 6 <--- bit 5 <--- bit 4 <--- bit 3 <--- bit 2 <--- bit 1 <--- bit 0 <--- '0'

Questo piccolo schema sta ad indicare che il contenuto del byte (8 bit) è spostato di una casella a sinistra: al posto del 'bit 0' è assegnato il valore '0' ed il 'bit 7' è spostato in 'C' del registro di stato(diventando '1' o '0').

Alcuni comandi:

14,n,n ---> Gli operandi indicano la locazione da manipolare.

10 ---> Lo scivolamento avviene all'interno dell' accumulatore .

6 ---> La locazione di pagina zero è indicata dall'operando.

Scivolamento a destra.

'0' ---> bit 7 ---> bit 6 ---> bit 5 ---> bit 4 ---> bit 3 ---> bit 2 ---> bit 1 ---> bit 0 ---> '0'

Questo piccolo schema sta ad indicare che il contenuto del byte (8 bit) è spostato di una casella a destra: al posto del 'bit 7' è assegnato il valore '0' ed il 'bit 0' è spostato in 'C' del registro di stato(diventando '1' o '0').

Alcuni comandi:

78,n,n ---> Gli operandi indicano la locazione da manipolare.

74 ---> Lo scivolamento avviene all'interno dell' accumulatore .

70 ---> La locazione di pagina zero è indicata dall'operando.

Rotazione a sinistra.

<---'C' <--- bit 7 <--- bit 6 <--- bit 5 <--- bit 4 <--- bit 3 <--- bit 2 <--- bit 1 <--- bit 0 <--- 'C' <---

Questo piccolo schema sta ad indicare che il contenuto del byte (8 bit) è ruotato di una casella a sinistra: al posto del 'bit 0' è assegnato il valore di 'C' ed il 'bit 7' è spostato in 'C' del registro di stato(diventando '1' o '0').

Alcuni comandi:

46,n,n ---> Gli operandi indicano la locazione da manipolare.

42 ---> Lo rotazione avviene all'interno dell' accumulatore .

38 ---> La locazione di pagina zero è indicata dall'operando.

Rotazione a destra.

--->'C' ---> bit 7 ---> bit 6 ---> bit 5 ---> bit 4 ---> bit 3 ---> bit 2 ---> bit 1 ---> bit 0 ---> 'C' --->

Questo piccolo schema sta ad indicare che il contenuto del byte (8 bit) è ruotato di una casella a sinistra: al posto del 'bit 7' è assegnato il valore di 'C' ed il 'bit 0' è spostato in 'C' del registro di stato(diventando '1' o '0').

Alcuni comandi:

110,n,n ---> Gli operandi indicano la locazione da manipolare.

106 ---> Lo rotazione avviene all'interno dell' accumulatore .

102 ---> La locazione di pagina zero è indicata dall'operando.

Trasferimento da accumulatore a indice X e Y e viceversa.

170 ---> Trasferisce da 'a' ad 'x'.

168 ---> Trasferisce da 'a' ad 'y'.

138 ---> Trasferisce da 'x' ad 'a'.

152 ---> Trasferisce da 'y' ad 'a'.

PROGRAMMA 13 – la pila o stack che dir si voglia

Il termine 'pila' in inglese significa 'stack' e viene usato per indicare la pagina 1 della memoria locazione (256 – 511). Questa zona della memoria è utilizzata per le 'note' del computer... possiamo però utilizzarla anche noi per i nostri scopi.

Le informazioni in questa porzione di memoria sono accantonate come una 'pila di piatti', uno sopra l'altro, quindi avremo una sequenza di questo tipo: sono depositati dal fondo (511, 510, 509, ecc...) e sono prelevati a partire dall'ultimo depositato (508,509,510,511). Abbastanza semplice vero!? Il computer per 'ricordarsi' dove è arrivato con la numerazione utilizza uno specifico puntatore...

Un esempio col quale il C64 ricorre allo stack è con il comando '32...96' cioè 'gosub...return', la pagina uno è utilizzata per tutte le interruzioni (che vedremo in seguito) e molto altro...

Vediamo brevemente questi comandi:

72 ---> trasferisce il contenuto da 'a' alla pila.

104 ---> trasferisce il contenuto dalla pila ad 'a'

8 ---> trasferisce il contenuto del registro di stato nella pila

40 ---> trasferisce il contenuto della pila nel registro di stato

186 ---> trasferisce il contenuto del puntatore in 'x'

154 ---> trasferisce il contenuto di 'x' al puntatore

Grazie a questi comandi abbiamo la possibilità di variare a nostro piacimento sia il puntatore che il contenuto dello stack, come fa già il computer silenziosamente in automatico quando esegue un programma.

PROGRAMMA 14 – ultimi comandi utili

Analizziamo due comandi che possono sembrare insignificanti ma che possono avere una loro efficacia: '0' e '234'.

0 ---> Sta per break, cioè sospende il programma nel punto in cui si trova il comando. Identica funzione di RUN-STOP e RESTORE...

234 ---> E' uno pseudo comando che viene utilizzato per tenere libere delle locazioni di memoria. Non ha altre funzioni.

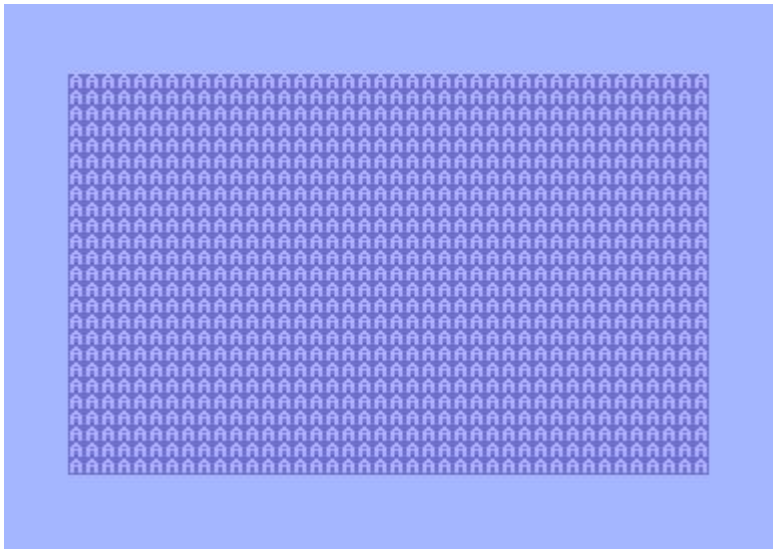
Eccovi alcune chicche che vi saranno certamente utili:

print peek(780) ---> legge il contenuto dell'accumulatore

print peek(781) ---> legge il contenuto dell'indice X

print peek(782) ---> legge il contenuto dell'indice Y

PROGRAMMA 15 – Le interruzioni (interrupt)



In questa sezione analizziamo un argomento di estrema importanza ovvero le interruzioni del Commodore 64. Un interruzione nel ciclo(dura un sessantesimo di secondo) del processore avviene quando si avvia un certo programma che è stato richiamato dal computer o da noi(come stiamo per vedere). Un esempio può essere quello dei tasti RUN/STOP+RESTORE che fermato il programma attivando un'interruzione durante l'esecuzione della maggior parte dei programmi, questo perché si richiama una specifica routine di blocco memorizzata dalla fabbrica nella memoria del computer. Le interruzioni però possiamo crearcele noi a nostro piacere manipolando in modo opportuno alcuni indirizzi. Alla riga di memoria 20 di pagina 3(locazione 788) il primo valore (indirizzo di riga a cui saltare) che normalmente è 49 ma noi possiamo cambiarlo a piacimento per indirizzarlo verso una nostra routine. Mentre alla riga di memoria 21 di pagina 3(locazione 789) troviamo il secondo valore (indirizzo di pagina a cui saltare) che normalmente è 234, che anche in questo caso possiamo variare a seconda delle nostre esigenze.

Esistono altre due locazioni (655-656) che ci posso indirizzare ad un'altra interruzione creata da noi che possiamo modificare. Riga 143 e 144 di pag 2, che come sopra contengono le coordinate di memoria a cui saltare e normalmente sono rispettivamente riga 72 e pagina 235.

Vediamo un esempio pratico:

Grazie alle interruzioni eseguiamo un programma LM che alla pressione del tasto 'F7' cambia il colore al bordo dello schermo e contemporaneamente esegue un programma Basic senza che questo si interrompa!

10 k=679: for n=k to k+15

20 reada: poke n,a: next

Queste due linee leggono i 16 comandi in LM partendo dalla locazione 679.

30 data 166,197 ---> Carica il valore contenuto nella locazione 197 di pagina zero in X che corrisponde al codice del tasto premuto

40 data 224,3 ---> Confronta il contenuto di X con il valore 3 (tasto F7), se è stato premuto il bit Z diventa '1'

50 data 208,7 ---> Se il bit Z è '0' il tasto non è stato premuto e perciò esegue un salto in avanti di 8 posti.

60 data 174,32,208 ---> Se il tasto F7 è premuto allora il bit Z è uguale a 1 quindi salta riga 32 di pagina 208 (locazione 53280) è preleva il valore in essa contenuto spostandolo in X (colore cornice attuale)

70 data 202 ---> decrementa il valore appena caricato in X facendo cosi variare i colori della cornice dello schermo.

80 data 142,32,208 ---> Inserisce il contenuto di X nuovamente della locazione 53280 (riga 32, pagina 208). Visualizzando materialmente il nuovo colore del bordo.

90 data 76,72,235 ---> Questo comando corrisponde a goto del Basic e in questo caso determina la fine della nostra interruzione ritornando all'indirizzo originale che sarebbe stato contenuto nelle locazioni '655' (qui la riga 72) e '656' (qui la pagina 235) se noi non le avessimo modificate a nostro piacimento(vedremo più avanti nel programma in che modo si modificano...)

100 poke 56334,0 ---> Interrompiamo momentaneamente l'orologio interno del computer, altrimenti la CPU ogni sessantesimo di secondo utilizzerebbe tutte le interruzioni e non potremmo modificarle!

110 poke 655,167 ---> Altera il contenuto di riga nella locazione 655 (riga 143, pagina 2) inserendo 167 anziché 72(riga al quale il computer avrebbe normalmente saltato)

120 poke 656,2 ---> Altera il contenuto di pagina nella locazione 656 (riga 144, pagina 2) inserendo 2 anziché 235(pagina al quale il computer salterebbe normalmente)

Nelle linee 110 e 120 abbiamo indicato la locazione 679 da cui parte l'interruzione che abbiamo scritto e al quale il computer porgerà la sua attenzione automaticamente ogni sessantesimo di secondo per un numero infinito di cicli.

130 poke 56334,1 ---> Dopo avere variato il valori dove il computer troverà la nostra interruzione possiamo riattivare l'orologio interno del C64 in modo che riprendano i cicli...

Aggiungiamo le ultime due linee Basic dimostrative se sono eseguite contemporaneamente al programma LM di interruzione.

140 print” “: for n=1024 to 2023: poke n,1: next

150 print” “: for n=1024 to 2023: poke n,1: next

160 goto 140

Le linee Basic eseguono un ciclo goto infinito in cui sono visualizzate 1000 'A' e poi altrettante 'B'. Proviamo a premere F7 e vediamo quello che succede...

Vi faccio presente una piccola chicca Basic:

10 print peek(197): goto 10 ---> Genera un ciclo infinito in cui sarà controllato e visualizzato il contenuto dell'indirizzo 197 di pagina zero, mostrando il valore numerico di ogni tasto premuto!

PROGRAMMA 16 – pulizia schermo, posizione cursore, stampa carattere e joystick

Questo è il primo vero programma di questo ebook sulla programmazione in linguaggio macchina del Commodore 64. In questa lezione impareremo molti concetti utili con un solo programma. Vediamo le sequenze di codici data che qui partono dalla locazione 49152:

32,68,229 ---> Richiama una routine Basic per la cancellazione dello schermo

169,10 ---> Carica in accumulatore il numero '10' che indica la riga centrale dello schermo.

133,214 ---> Stampa il contenuto dell'accumulatore alla locazione 214 di pagina zero

169,20 ---> Carica in accumulatore il numero '20' che indica la colonna centrale dello schermo.

133,211 ---> Stampa il contenuto dell'accumulatore alla locazione 211 di pagina zero

32,16,229 ---> Colloca il cursore nella posizione indicata nelle locazioni 214 e 211

169,81 ---> Carica in accumulatore il valore '81' che indica il simbolo della pallina che sarà stampato nella posizione indicata

145,209 ---> istruzione di stampa indicizzata 'Y' del carattere voluto. All'indirizzo 209 e successivo 210 si trovano la riga e la pagina dove si partirà per la stampa. Servirà per il movimento.

173,1,220 ---> mette in accumulatore l'indirizzo '56321'(riga 1, pagina 220) corrispondente alla porta 1 del joystick.

201,255 ---> confronta il contenuto dell'accumulatore con il numero '255' corrispondente alla posizione joystick fermo. Se fermo il bit Z del registro di stato è '1' , se mosso Z=0.

240,249 ---> Se Z=1 torna indietro di sei posti (255-249=6). Se '0' prosegue all'istruzione successiva.

201,251 ---> Confronta '56321' con il valore '251' che indica se il joystick è spinto a sinistra. Se fermo Z=0 mentre se spinto a sx è Z=1.

208,12 ---> Se il bit Z è '0' salta avanti di 13 posti (da 0 a 12 e il conteggio parte dal codice successivo a '208,12') all'istruzione successiva. Mentre se Z è '1' esegue la routine di movimento.

192,0 ---> confronta il contenuto dell'indice Y, responsabile dei movimenti orizzontali, se è uguale a '0' significa che il carattere è arrivato a contatto con il bordo dello schermo mettendo Z=0.

240,5 ---> Se Z=1 prosegue all'istruzione successiva, visto che non c'è contatto con il bordo. Se Z=0 salta avanti di '6' posti (da 0 a 5 partendo dal codice successivo a 240,5) perché c'é contatto.

32,41,192 ---> richiama la sub routine di cancellazione carattere è ritardo alla riga 41 di pagina 192.

198,211 ---> Diminuisce il contenuto della locazione '211', determinando il movimento di un carattere a sinistra.

76,11,192 ---> Salta alla riga 11 di pagina 192 che è la locazione dove inizia il processo di stampa della pallina.

160,50 ---> Carica in 'y' 50.

162,50 ---> Carica in 'x' 50.

202 ---> diminuisce di '1' il valore di 'x'

208,253 ---> Se 'x' non è '0' allora torna indietro fino a che il ciclo non lo raggiunge. Rallentando il computer.

136 ---> diminuisce di '1' il valore di 'y'

208,248 ---> Se 'y' non è '0' allora torna indietro fino a che il ciclo non lo raggiunge. Rallentando ulteriormente il computer. Questi due esempi racchiudono un ciclo annidato 'for...next'

164,211 ---> Carica il contenuto dell'indice Y dalla locazione 211 contenete la posizione del carattere.

169,32 ---> Cancella la pallina con uno spazio (caricando il codice video 32).

145,209 ---> Stampa nella posizione dettata da 'y' il carattere di spazio.

96 ---> Funzione RETURN che chiude la sub routine andando al comando successivo al codice '32,41,192' (gosub).

Per eseguire dei movimenti nelle quattro diagonali è sufficiente combinare alcuni dei comandi che vi ho appena illustrato.

Il listato esaminato permette di spostare una pallina per mezzo della leva joystick 1 nella direzione sinistra. Per spostare la pallina in tutte le otto direzioni è sufficiente ampliare il programma seguendo la logica esposta confrontando tutti i valori emessi dalla periferica di gioco.

Per cambiare direzione è sufficiente sostituire i valori compresi partendo dal comando 201,251 arrivando fino al comando 76,11,192, vediamo:

Movimento alto.

201,254 ---> Confronta '56321' con il valore '254' che indica se il joystick è spinto in alto. Se fermo Z=0 mentre se spinto in alto è Z=1.

208,12 ---> Se il bit Z è '0' salta avanti di 13 posti (da 0 a 12 e il conteggio parte dal codice successivo a '208,12') all'istruzione successiva. Mentre se Z è '1' esegue la routine di movimento.

224,0 ---> confronta il contenuto dell'indice X, responsabile dei movimenti verticali, se è uguale a '0' significa che il carattere è arrivato a contatto con il bordo dello schermo mettendo Z=0.

240,5 ---> Se Z=1 prosegue all'istruzione successiva, visto che non c'è contatto con il bordo. Se Z=0 salta

avanti di '6' posti (da 0 a 5 partendo dal codi successivo a 240,5) perché c'é contatto.

32,41,192 ---> richiama la sub routine di cancellazione carattere è ritardo alla riga 41 di pagina 192.

198,214 ---> Diminuisce il contenuto della locazione '214', determinando il movimento di un carattere in alto.

76,11,192 ---> Salta alla riga 11 di pagina 192 che è la locazione dove inizia il processo di stampa della pallina.

Movimento a destra.

201,247 ---> Confronta '56321' con il valore '247' che indica se il joystick è spinto a destra. Se fermo Z=0 mentre se spinto a destra è Z=1.

208,12 ---> Se il bit Z è '0' salta avanti di 13 posti (da 0 a 12 e il conteggio parte dal codice successivo a '208,12') all'istruzione successiva. Mentre se Z è '1' esegue la routine di movimento.

192,39 ---> confronta il contenuto dell'indice Y, responsabile dei movimenti orizzontali, se è uguale a '39' significa che il carattere è arrivato a contatto con il bordo dello schermo mettendo Z=0.

240,5 ---> Se Z=1 prosegue all'istruzione successiva, visto che non c'è contatto con il bordo. Se Z=0 salta avanti di '6' posti (da 0 a 5 partendo dal codi successivo a 240,5) perché c'é contatto.

32,41,192 ---> richiama la sub routine di cancellazione carattere è ritardo alla riga 41 di pagina 192.

230,211 ---> Aumenta il contenuto della locazione '211', determinando il movimento di un carattere a destra.

76,11,192 ---> Salta alla riga 11 di pagina 192 che è la locazione dove inizia il processo di stampa della pallina.

Movimento basso.

201,253 ---> Confronta '56321' con il valore '253' che indica se il joystick è spinto in basso. Se fermo Z=0 mentre se spinto in basso è Z=1.

208,12 ---> Se il bit Z è '0' salta avanti di 13 posti (da 0 a 12 e il conteggio parte dal codice successivo a '208,12') all'istruzione successiva. Mentre se Z è '1' esegue la routine di movimento.

224,24 ---> confronta il contenuto dell'indice X, responsabile dei movimenti verticali, se è uguale a '24' significa che il carattere è arrivato a contatto con il bordo dello schermo mettendo Z=0.

240,5 ---> Se Z=1 prosegue all'istruzione successiva, visto che non c'è contatto con il bordo. Se Z=0 salta avanti di '6' posti (da 0 a 5 partendo dal codi successivo a 240,5) perché c'é contatto.

32,41,192 ---> richiama la sub routine di cancellazione carattere è ritardo alla riga 41 di pagina 192.

230,214 ---> Aumenta il contenuto della locazione '214', determinando il movimento di un carattere in basso

76,11,192 ---> Salta alla riga 11 di pagina 192 che è la locazione dove inizia il processo di stampa della pallina.

Aggiungo che la locazione 56321 è identica alla locazione 145 di pagina zero. La locazione 145 però è senz'altro più veloce perché è gestita senza l'indicazione della pagina, quindi con un operando in meno.

Per quello che concerne la gestione della porta joystick numero 2 l'indirizzo è il 56320.

Per conoscere tutti i valori della porta 1 è sufficiente digitare il seguente programma Basic:

10 print peek (56321): goto 10

Se si vogliono conoscere i valori della porta 2 è sufficiente sostituire il valore 56321 con 56320 oppure 145 per la porta 1 ma da pagina zero.

PROGRAMMA 17 – movimenti con la tastiera

Con qualche piccola modifica al programma illustrato nella lezione 16 possiamo sostituire i movimenti con il joystick con quelli della tastiera. Alla pressione della lettera 'A' la pallina scorre a sinistra. Esaminiamo il codice macchina che segue:

32,68,229 ---> Richiama una routine Basic per la cancellazione dello schermo

169,10 ---> Carica in accumulatore il numero '10' che indica la riga centrale dello schermo.

133,214 ---> Stampa il contenuto dell'accumulatore alla locazione 214 di pagina zero

169,20 ---> Carica in accumulatore il numero '20' che indica la colonna centrale dello schermo.

133,211 ---> Stampa il contenuto dell'accumulatore alla locazione 211 di pagina zero

32,16,229 ---> Colloca il cursore nella posizione indicata nelle locazioni 214 e 211

169,81 ---> Carica in accumulatore il valore '81' che indica il simbolo della pallina che sarà stampato nella posizione indicata

145,209 ---> istruzione di stampa indicizzata 'Y' del carattere voluto. All'indirizzo 209 e successivo 210 si trovano la riga e la pagina dove si partirà per la stampa. Servirà per il movimento.

165,197 ---> Carica il contenuto della locazione 197 di pagina zero in accumulatore. Cioè il codice del tasto premuto.

201,1 ---> Confronta il valore presente nell'accumulatore con quello specificato dall'operando, '1' in questo caso equivalente al tasto 'return'. Quindi Z=1 se sono uguali.

240,250 ---> Salta indietro di cinque posizioni se Z=1. Quindi esegue il controllo fino a che viene premuto il tasto 'return'. Alla pressione di un altro tasto passa all'istruzione successiva.

165,197 ---> Carica il contenuto della locazione 197 di pagina zero in accumulatore. Cioè il codice del tasto premuto.

201,64 ---> Confronta il valore presente nell'accumulatore con quello specificato dall'operando, '64' in questo caso equivalente a 'nessun tasto premuto'. Quindi Z=1 se sono uguali.

240,250 ---> Salta indietro di cinque posizioni se Z=1. Quindi esegue il controllo fino a che non è premuto alcun tasto. Alla pressione di un tasto passa all'istruzione successiva(anche 'return').

165,197 ---> Carica il contenuto della locazione 197 di pagina zero in accumulatore. Cioè il codice del

tasto premuto.

201,10 ---> Confronta accumulatore con il valore '10' che indica la pressione della lettera 'A' . Se fermo Z=0 mentre se premuta la 'A' è Z=1.

208,12 ---> Se il bit Z è '0' salta avanti di 13 posti (da 0 a 12 e il conteggio parte dal codice successivo a '208,12') all'istruzione successiva. Mentre se Z è '1' esegue la routine di movimento.

192,0 ---> confronta il contenuto dell'indice Y, responsabile dei movimenti orizzontali, se è uguale a '0' significa che il carattere è arrivato a contatto con il bordo dello schermo mettendo Z=0.

240,5 ---> Se Z=1 prosegue all'istruzione successiva, visto che non c'è contatto con il bordo. Se Z=0 salta avanti di '6' posti (da 0 a 5 partendo dal codi successivo a 240,5) perché c'é contatto.

32,48,192 ---> richiama la sub routine di cancellazione carattere è ritardo alla riga 48 di pagina 192 (per la tastiera servono più comandi rispetto al joystick).

198,211 ---> Diminuisce il contenuto della locazione '211', determinando il movimento di un carattere a sinistra.

76,11,192 ---> Salta alla riga 11 di pagina 192 che è la locazione dove inizia il processo di stampa della pallina.

160,50 ---> Carica in 'y' 50.

162,50 ---> Carica in 'x' 50.

202 ---> diminuisce di '1' il valore di 'x'

208,253 ---> Se 'x' non è '0' allora torna indietro fino a che il ciclo non lo raggiunge. Rallentando il computer.

136 ---> diminuisce di '1' il valore di 'y'

208,248 ---> Se 'y' non è '0' allora torna indietro fino a che il ciclo non lo raggiunge. Rallentando ulteriormente il computer. Questi due esempi racchiudono un ciclo annidato 'for...next'

164,211 ---> Carica il contenuto dell'indice Y dalla locazione 211 contenete la posizione del carattere.

169,32 ---> Cancella la pallina con uno spazio (caricando il codice video 32).

145,209 ---> Stampa nella posizione dettata da 'y' il carattere di spazio.

96 ---> Funzione RETURN che chiude sa sub routine andando al comando successivo al codice '32,41,192' (gosub).

Come avete notato non sono presenti molte variazioni rispetto al programma legato alla gestione del joystick con questo per la tastiera. Le poche variazioni presenti consistono nel controllo del tasto 'return'(codice 1), visto che il C64 è molto veloce e nel momento in cui si lancia il programma con il comando 'run' la pressione di 'return' avviene svariate volte senza che ce ne accorgiamo, facendo interrompere immediatamente il programma se non ci fosse un controllo che ne esclude la rilevazione! Una seconda routine di controllo è quella per la pressione di 'nessun tasto' visto che il C64 quando non è premuto alcun tasto rileva comunque un valore(64) che anche in questo caso provocherebbe la fine del programma...

Per rilevare la tastiera invece del joystick è stata sostituita la locazione '56321' con la '197' utilizzata in

questo caso o la '203' di pagina zero.

In questo programma si verifica lo spostamento a sinistra di una pallina sullo schermo. Se volessimo spostarla in una direzione diversa sarà sufficiente variare il codice dei tasti da premere e applicare le regole esposte nella lezione 16.

Per verificare il codici di pressione tasti scrivete questo breve programma Basic:

10 print peek(197): goto 10

PROGRAMMA 18 - Visualizziamo il primo sprite in LM

- Poke in Linguaggio Macchina

Basic: poke 53281,1 --> cambia il colore dello schermo

LM:

169, 1 --> in accumulatore valore 1
141, 33,208 --> stampa l'accumulatore all'indirizzo 53281

- Visualizziamo il primo sprite in LM

Carico i 31 comandi macchina dal BASIC:
10 k=49152
20 for n=k to k+30
30 read a
40 poke n,a
50 next n
60 sys k

I valori poke convertiti in LM:

169, 11 --> inserisce 11 in accumulatore ulguale a indirizzo 704(64*11), prima locazione utile dello sprite
141, 248,7 --> Carica l'accumulatore nella zona del primo sprite (indirizzo 2040)

LOCAZIONI DI ATTIVAZIONE DI MEMORIA DEGLI OTTO SPRITE

SPRITE 1 --> 2040 --> RIGA 248 PAGINA 7
SPRITE 2 --> 2041 --> RIGA 249 PAGINA 7
SPRITE 3 --> 2042 --> RIGA 250 PAGINA 7
SPRITE 4 --> 2043 --> RIGA 251 PAGINA 7
SPRITE 5 --> 2044 --> RIGA 252 PAGINA 7
SPRITE 6 --> 2045 --> RIGA 253 PAGINA 7
SPRITE 7 --> 2046 --> RIGA 254 PAGINA 7
SPRITE 8 --> 2047 --> RIGA 255 PAGINA 7

DOVE INSERIRE 63 CODICI DI DISEGNO SPRITE

VALORE DA INTRODURRE NELLA LOCAZIONE 2040 TRATTO DI MEMORIA CORRISPONDENTE
- RIGA 248 DI PAGINA 7 - IN POI... - 64 MOLTIPLICATO VALORE -
11 704-766
13 832-894
14 896-958
15 960-1022
32 2048-2110
33 2112-2174
63 4032-4094
128 8192-8254
129 8256-8318
255 16320-16382

169, 1 --> Inserisce 1 nella accumulatore che servirà per attivare il primo sprite
141, 21,208 --> trasferisce l'accumulatore a questa locazione (53269) per attivare lo sprite

ATTIVARE UNO O PIU' SPRITE NELLA INTERAGENDO CON LA LOCAZIONE 53269
- RIGA 21 DI PAGINA 208 -

TABELLA BINARIA CON VALORE DA INSERIRE IN ACCUMULATORE

128 - 64 - 32 - 16 - 8 - 4 - 2 - 1
X 128--> SPRITE 8 ATTIVO
X 64--> SPRITE 7 ATTIVO
X 32--> SPRITE 6 ATTIVO
X 16--> SPRITE 5 ATTIVO
X 8--> SPRITE 4 ATTIVO
X 4--> SPRITE 3 ATTIVO
X 2--> SPRITE 2 ATTIVO
X 1--> SPRITE 1 ATTIVO
0--> TUTTI GLI SPRITE NON ATTIVI

Se volessimo attivare più sprite sarà necessario inserire la somma dei loro valori.
Es: sprite 1(valore 1) e Sprite 7(valore 64) --> 1+64=65 --> valore 65 attiverà gli sprite 1 e 7

169, 160 --> Assegno 100 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite
141, 0,208 --> Assegna 100 dall'accumulatore alla locazione(53248) di posizione sprite 1 su asse X

Locazioni per il posizionamento degli sprite sull'asse x dello schermo. Valore in pixel nell'accumulatore da 0 a 255

SPRITE N. LOCAZIONE

1 53248 - RIGA 0 PAGINA 208
2 53250 - RIGA 2 PAGINA 208
3 53252 - RIGA 4 PAGINA 208
4 53254 - RIGA 6 PAGINA 208
5 53256 - RIGA 8 PAGINA 208
6 53258 - RIGA 10 PAGINA 208
7 53260 - RIGA 12 PAGINA 208
8 53262 - RIGA 14 PAGINA 208

Molti noteranno che con l'immissione massima di 255 non si copre tutta la superficie orizzontale dello schermo.
A tale proposito esiste la locazione 53264 che permette di attivare i movimenti degli sprite oltre il limite 255.
Vedremo poco dopo come si usa...

169, 160 --> Assegno 120 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite
141, 1,208 --> Assegna 120 dall'accumulatore alla locazione(53249) di posizione sprite 1 su asse y

Locazioni per il posizionamento degli sprite sull'asse Y dello schermo. Valore in pixel nell'accumulatore da 0 a 255

SPRITE N. LOCAZIONE
1 53249 - RIGA 1 PAGINA 208
2 53251 - RIGA 3 PAGINA 208
3 53253 - RIGA 5 PAGINA 208
4 53255 - RIGA 7 PAGINA 208
5 53257 - RIGA 9 PAGINA 208
6 53259 - RIGA 11 PAGINA 208
7 53261 - RIGA 13 PAGINA 208
8 53263 - RIGA 15 PAGINA 208

169, 0 --> Inserendo 0 nell'accumulatore dice a computer che l'aerea di schermo x e nei primi 256 pixel
141, 16, 208 --> La locazione 53264 determina l'aerea di schermo in cui e collocato lo sprite e quale degli 8 disponibili

ATTIVARE LO SPOSTAMENTO OLTRE IL 255ESIMO PIXEL PER UNO O PIU' SPRITE INTERAGENDO CON LA LOCAZIONE 53264
- RIGA 16 DI PAGINA 208 -

TABELLA BINARIA CON VALORE DA INSERIRE IN ACCUMULATORE

128 - 64 - 32 - 16 - 8 - 4 - 2 - 1
X 128--> SPRITE 8
X 64--> SPRITE 7
X 32--> SPRITE 6
X 16--> SPRITE 5
X 8--> SPRITE 4
X 4--> SPRITE 3
X 2--> SPRITE 2
X 1--> SPRITE 1
0--> SECONDA PARTE SCHERMO NON ATTIVA PER TUTTI GLI SPRITE

Se volessimo eseguire il comando su più sprite sarà necessario inserire la somma dei loro valori.
Es: sprite 1(valore 1) e Sprite 7(valore 64) --> 1+64=65 --> valore 65 attiverà gli sprite 1 e 7 NELLA

SECONDA PARTE DELLO SCHERMO

169, 1 --> ASSEGNA IL COLORE ALLO SPRITE NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 39,208 --> LOCAZIONE 53287 CHE INDICA IL COLORE DELLO SPRITE 1

LOCAZIONI COLORE DEGLI 8 SPRITE:
SPRITE 1 --> 53287
SPRITE 2 --> 53288
SPRITE 3 --> 53289
SPRITE 4 --> 53290
SPRITE 5 --> 53291
SPRITE 6 --> 53292
SPRITE 7 --> 53293
SPRITE 8 --> 53294

96 --> restituisce il controllo al BASIC

Carico i 63 valori dei DATA del sprite 1 dalla locazione 704:

```
1000 k=704
1010 for n=k to k+62
1020 read a
1030 poke n,a
1040 next n
```

Digito i valori 63 di disegno distribuiti su 21 linee(Ricordo che cambiando i valori di seguito sotto cambia il disegno dello sprite):

```
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
```

Tabella riassuntiva d'esempio su come si disegna uno sprite monocromatico(24X21pixel):

pixel --> 8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1 8 7 6 5

4 3 2 1
128 - 64 - 32 - 16 - 8 - 4 - 2 - 1 | 128 - 64 - 32 - 16 - 8 - 4 - 2 - 1 | 128 - 64 - 32 - 16 - 8 - 4 - 2 - 1

255,129,64 --> 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 | 1 - 0 - 0 - 0 - 0 - 0 - 0 - 1 | 0 - 1 - 0 - 0 - 0 - 0 - 0 - 0

...

Proseguendo così per 21 righe

...

Programma 19 - Visualizziamo 2 sprite identici in LM

Carico i 51 comandi macchina dal BASIC:

```
10 k=49152
20 for n=k to k+50
30 read a
40 poke n,a
50 next n
60 sys k
```

169, 11 --> inserisce 11 in accumulatore uguale a indirizzo 704(64*11), prima locazione utile del primo sprite
141, 248,7 --> Carica l'accumulatore nella zona del primo sprite (indirizzo 2040)

169, 3 --> Inserisce 3 nella accumulatore che servirà per attivare i 2 sprite (bit 1 + bit 2)
141, 21,208 --> trasferisce l'accumulatore a questa locazione (53269) per attivare gli sprite

169, 20 --> Assegno 20 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite
141, 0,208 --> Assegna 20 dall'accumulatore alla locazione(53248) di posizione sprite 1 su asse X

169, 229 --> Assegno 229 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite
141, 1,208 --> Assegna 229 dall'accumulatore alla locazione(53249) di posizione sprite 1 su asse y

169, 2 --> Inserendo 2 nell'accumulatore dice a computer che l'area di schermo x è oltre i primi 256 pixel per il secondo sprite mentre per il primo no
141, 16, 208 --> La locazione 53264 determina l'area di schermo in cui è collocato lo sprite e quale degli 8 disponibili

169, 3 --> ASSEGNA IL COLORE ALLO SPRITE 1 NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 39,208 --> LOCAZIONE 53287 CHE INDICA IL COLORE DELLO SPRITE 1

- indicazioni per il secondo sprite

169, 11 --> inserisce 11 in accumulatore uguale a indirizzo 704(64*11), prima locazione utile dello sprite
141, 249,7 --> Carica l'accumulatore nella zona del secondo sprite (indirizzo 2041)
(qui viene attivato caricato in memoria un secondo sprite con disegno identico al primo)

169, 50 --> Assegno 50 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite 2
141, 2,208 --> Assegna 20 dall'accumulatore alla locazione(53250) di posizione sprite 2 su asse X
(ricordo che al secondo sprite è assegnata la posizione dello schermo oltre 256(qui 255+50))

169, 129 --> Assegno 129 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite 2
141, 3,208 --> Assegna 129 dall'accumulatore alla locazione(53251) di posizione sprite 2 su asse y

169, 7 --> ASSEGNA IL COLORE ALLO SPRITE 2 NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 40,208 --> LOCAZIONE 53288 CHE INDICA IL COLORE DELLO SPRITE 2

96 --> restituisce il controllo al BASIC

Carico i 63 valori dei DATA del sprite 1 dalla locazione 704:
(il disegno composto da questi valori è utilizzato da entrambi gli sprite in questo esempio)

```
1000 k=704
1010 for n=k to k+62
1020 read a
1030 poke n,a
1040 next n
```

Digito i valori 63 di disegno distribuiti su 21 linee(Ricordo che cambiando i valori di seguito sotto cambia il disegno dello sprite):

```
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
```

255,255,255
255,255,255
255,255,255
255,255,255

Programma 20 - Visualizziamo 2 sprite differenti in LM

Carico i 51 comandi macchina dal BASIC:

```
10 k=49152
20 for n=k to k+50
30 read a
40 poke n,a
50 next n
60 sys k
```

169, 11 --> inserisce 11 in accumulatore uguale a indirizzo 704(64*11), prima locazione utile dello sprite
141, 248,7 --> Carica l'accumulatore nella zona del primo sprite (indirizzo 2040)

169, 3 --> Inserisce 3 nella accumulatore che servirà per attivare i 2 sprite (bit 1 + bit 2)
141, 21,208 --> trasferisce l'accumulatore a questa locazione (53269) per attivare lo sprite

169, 30 --> Assegno 30 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite
141, 0,208 --> Assegna 30 dall'accumulatore alla locazione(53248) di posizione sprite 1 su asse X

169, 229 --> Assegno 229 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite
141, 1,208 --> Assegna 229 dall'accumulatore alla locazione(53249) di posizione sprite 1 su asse y

169, 2 --> Inserendo 2 nell'accumulatore dice a computer che l'aerea di schermo x è oltre i primi 256 pixel per il secondo sprite mentre per il primo no
141, 16, 208 --> La locazione 53264 determina l'aerea di schermo in cui è collocato lo sprite e quale degli 8 disponibili

169, 3 --> ASSEGNA IL COLORE ALLO SPRITE 1 NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 39,208 --> LOCAZIONE 53287 CHE INDICA IL COLORE DELLO SPRITE 1

- indicazioni per il secondo sprite

169, 13 --> inserisce 13 in accumulatore uguale a indirizzo 832(64*13), prima locazione utile dello sprite 2
141, 249,7 --> Carica l'accumulatore nella zona del secondo sprite (indirizzo 2041)
(qui viene attivato caricato in memoria un secondo sprite con disegno differente dal primo residente in un'altra zona di memoria -832-)

169, 50 --> Assegno 50 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite 2
141, 2,208 --> Assegna 20 dall'accumulatore alla locazione(53250) di posizione sprite 2 su asse X

169, 129 --> Assegno 129 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite 2
141, 3,208 --> Assegna 129 dall'accumulatore alla locazione(53251) di posizione sprite 2 su asse y

169, 7 --> ASSEGNA IL COLORE ALLO SPRITE 2 NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 40,208 --> LOCAZIONE 53288 CHE INDICA IL COLORE DELLO SPRITE 2

96 --> restituisce il controllo al BASIC

Carico i 63 valori dei DATA del sprite 1 dalla locazione 704:

```
1000 k=704
1010 for n=k to k+62
1020 read a
1030 poke n,a
1040 next n
```

Digito i valori 63 di disegno distribuiti su 21 linee X 24 colonne che creano lo sprite C64 monocromatico:

3,3,2

12,4,6

48,12,10

64,8,18

64,16,46

128,48,66

128,32,130

128,64,130

128,192,130

128,128,130

128,112,130

128,140,130

128,130,127

128,130,2

128,129,2

128,129,2

64,65,2

64,65,2

48,33,2

12,17,2

3,14,2

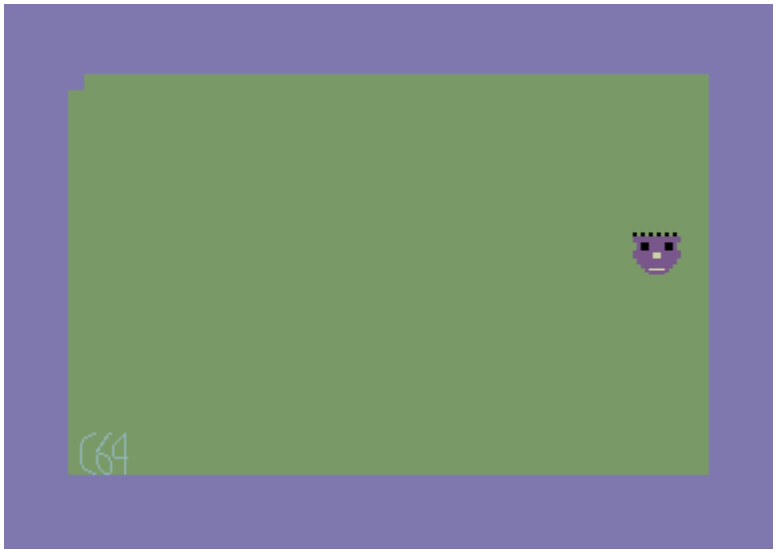
Valori nell'aerea di memoria 13(64*13=832) del secondo sprite diverso dal primo

```
2000 k=832
2010 for n=k to k+62
2020 read a
2030 poke n,a
2040 next n
```

Digito i valori 63 di disegno distribuiti su 21 linee:

255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,0,255
255,0,255
255,0,255
255,0,255
255,0,255
255,0,255
255,0,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255
255,255,255

Programma 21 - Visualizziamo 2 sprite differenti di cui uno in multicolore e li dilatiamo in LM



Carico i 51 comandi macchina dal BASIC:
10 k=49152
20 for n=k to k+75
30 read a
40 poke n,a
50 next n
60 sys k

169, 11 --> inserisce 11 in accumulatore uguale a indirizzo 704(64*11), prima locazione utile dello sprite
141, 248,7 --> Carica l'accumulatore nella zona del primo sprite (indirizzo 2040)

169, 3 --> Inserisce 3 nella accumulatore che servirà per attivare i 2 sprite (bit 1 + bit 2)
141, 21,208 --> trasferisce l'accumulatore a questa locazione (53269) per attivare lo sprite

169, 30 --> Assegno 30 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite
141, 0,208 --> Assegna 30 dall'accumulatore alla locazione(53248) di posizione sprite 1 su asse X

169, 229 --> Assegno 229 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite
141, 1,208 --> Assegna 229 dall'accumulatore alla locazione(53249) di posizione sprite 1 su asse y

169, 2 --> Inserendo 2 nell'accumulatore dice a computer che l'aerea di schermo x è oltre i primi 256 pixel per il secondo sprite mentre per il primo no
141, 16, 208 --> La locazione 53264 determina l'aerea di schermo in cui è collocato lo sprite e quale degli 8 disponibili

169, 3 --> ASSEGNA IL COLORE ALLO SPRITE 1 NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 39,208 --> LOCAZIONE 53287 CHE INDICA IL COLORE DELLO SPRITE 1

- indicazioni per il secondo sprite

169, 13 --> inserisce 13 in accumulatore uguale a indirizzo 832(64*13), prima locazione utile dello sprite 2
141, 249,7 --> Carica l'accumulatore nella zona del secondo sprite (indirizzo 2041)
(qui viene attivato caricato in memoria un secondo sprite con disegno differente dal primo residente in un'altra zona di memoria-832-)

169, 50 --> Assegno 50 all'accumulatore, che sarà la posizione in pixel sull'asse x dello sprite 2
141, 2,208 --> Assegna 20 dall'accumulatore alla locazione(53250) di posizione sprite 2 su asse X

169, 129 --> Assegno 129 all'accumulatore, che sarà la posizione in pixel sull'asse y dello sprite 2
141, 3,208 --> Assegna 129 dall'accumulatore alla locazione(53251) di posizione sprite 2 su asse y

169, 7 --> ASSEGNA IL COLORE ALLO SPRITE 2 NELL'ACCUMULATORE, VALORI DA 0 A 15
141, 40,208 --> LOCAZIONE 53288 CHE INDICA IL COLORE DELLO SPRITE 2

- indicazioni per il multicolore del primo sprite

Gli sprite del commodore 64 come già detto sono 8 e possono essere monocromatici oppure multicolore. Il multicolore si attiva con un compromesso sulla risoluzione della figura in orizzontale.
Ovvero non visualizzeremo più una immagine di 24X21 pixel ma di 12X21 pixel. Questo per la particolare
gestione dei pixel colore, dove 2 bit formeranno un pixel con l'informazione colore. Mentre nel monocromatico ogni bit rappresenta un pixel senza necessità di informazioni colore.
Avremo così quattro tipi di bit(coppie di 2 bit) colore: 00, 01, 11, 10.

Dopo questa piccola premessa è necessario attivare il multicopore per lo sprite desiderato, che può essere uno o più...

Con la locazione '53276 è possibile selezionare gli sprite multicolore. In linguaggio macchina tale indirizzo corrisponde a '28,208'

169,2 --> e attivo il multicolore sullo sprite 2 lasciando gli altri monocromatici

141,28,208 --> stampa alla locazione 28,208 il comando 1 dell'accumulatore per indicare il multicolore dello sprite 1

Se volessimo attivare il multicolore per lo sprite 1 e 2 dovremo caricare in accumulatore il valore 3 ovvero 1+2. Mentre se volessimo rendere multicolore lo sprite 1 e 8 allora dovremo caricare 129... Secondo questo schema:

Sprite 1 - Sprite 2 - Sprite 3 - Sprite 4 - Sprite 5 - Sprite 6 - Sprite 7 - Sprite 8

1 2 4 8 16 32 64 128

Quindi per scegliere 2 o più sprite sarà sufficiente sommare i valori corrispondenti ed indicarli in accumulatore, sapendo che '0' disattiva il multicolore per tutti gli sprite e '255' lo attiva per tutti!

Il Commodore 64 può assegnare la modalità multucolore a 4 colori in totale ad ognuno dei suoi 8 sprite. Ci sono però alcuni accorgimenti di cui ricordarsi:

- Dalla locazione 53287(39,208) alla 53294(46,208) gestisce il primo colore che può essere differente in ogni sprite

- Le locazioni 53285(37,208) e 53286(38,208) gestiscono il secondo e terzo colore di tutti gli 8 sprite e questo significa che saranno identici!

- L'ultimo è il colore di sfondo che è dato dalla locazione 53281(33,208)

Avremo sempre per il colore 1 del primo sprite:

169,7 --> Assegna il colore 7 ad accumulatore

141,40,208 --> attiva il colore 7 dettato da accumulatore sul primo sprite (locazione 53287)

Per i colori 2 e 3 comuni in tutti gli sprite:

169, 4 --> seleziona il colore 4 in accumulatore

141,37,208 --> Attiva il secondo colore di valore 4 di accumulatore negli sprite (locazione 53285)

169, 0 --> seleziona il colore 0 in accumulatore

141,38,208 --> Attiva il terzo colore di valore 0 di accumulatore negli sprite (locazione 53286)

Assegna il colore di sfondo:

169, 1 --> seleziona il colore 1 in accumulatore

141,33,208 --> Attiva colore del fondale di valore 1 di accumulatore (locazione 53281)

96 --> rende il controllo al basic

Scheda d'esempio per la creazione dello sprite multicolor e la dislocazione dei 4 colori(12x21 piexl):

pixel --> 8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1 8 7 6 5 4 3 2 1
128 - 64 - 32 - 16 - 8 - 4 - 2 - 1 | 128 - 64 - 32 - 16 - 8 - 4 - 2 - 1 | 128 - 64 - 32 - 16 - 8 - 4 - 2 - 1

255,162,124 --> |1 1 - 1 1 - 1 1 - 1 1 | 1 0 - 1 0 - 0 0 - 1 0 | 0 1 - 0 1 - 1 1 - 0 0

L'abbinamento dei bit da seguire per la distribuzione dei colori nell'immagine sarà:

01 --> colore locazione 53285

11 --> colore locazione 53286

10 --> colore da locazione 53287 a 53294 per gli 8 sprite

00 --> colore sfondo locazione 53281

.

Inseriamo i valori di disegno dello sprite 2 multicolor per creare una faccina:

204,204,204
204,204,204
85,85,85
85,85,85
85,85,85
31,85,244
31,85,244
31,85,244
31,85,244
85,85,85
85,105,85
85,105,85
85,105,85

21,85,84
21,85,84
21,85,84
5,85,80
5,85,80
1,170,64
1,85,64
0,85,0

Aggiungo ancora qualche parola sulla dilatazione degli sprite. Soluzione 'economica' e veloce che permette con due sole istruzioni di ridimensionare le figure degli sprite su entrambi glia assi sul vostro piccolo C64:

locazione 53271 raddoppia l'asse Y

locazione 53277 raddoppia l'asse X

Inserendo al precedente listato i comandi:

169,3 --> scelgo di raddoppiare sull'asse Y gli sprite 1 e 2

141,23,208 --> attiva la locazione di memoria per raddoppiare gli sprite 1 e 2

169,3 --> scelgo di raddoppiare sull'asse X gli sprite 1 e 2

141,29,208 --> attiva la locazione di memoria per raddoppiare gli sprite 1 e 2

Programma 22 - Ridisegnare un carattere e visualizzarlo su schermo

Tratto di memoria in cui è contenuta la lettera 'A' --> locazione 12296. Qui verranno caricati gli 8 bit(7+0) che andranno a riscrivere la lettera

che verrà poi visualizzata nella pagina grafica monocromatica... Ma attenzione la vedrete Rovesciata!!!

10 for byte= 12296 to 12296+7

20 read info

30 poke byte, info

40 next byte

Il comando 'data' che contiene i valori validi per ridisegnare il carattere alfabetico

100 data 238,204,204,252,204,216,112,0

Sembrerà, dando il 'run', che non sia successo nulla ma all'attivazione della pagina grafica con 'poke 53272,29' vedremo il risultato alla pressione del 'tasto A'

Il funzionamento di disegno è identico a quello usato per gli sprite, con la differenza che è lungo un solo byte(8bit)

Tabella riassuntiva su come si disegna uno carattere monocromatico(8X8 pixel):

pixel --> 8 7 6 5 4 3 2 1

128 - 64 - 32 - 16 - 8 - 4 - 2 - 1

255 --> 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1

...

Proseguendo così per 8 righe

...

Inserimento a partire dalla locazione 49152 per i 19 comandi di gestione del carattere(18+0 istruzioni) in codice macchina

200 k=49152

210 for n=k to k+18

220 read a

230 poke n,a

240 next n

250 sys k

Prosegue l'inserimento dei codici macchina nei comandi 'data'...

32,32,229 --> Pulisce lo schermo

169,1 --> Visualizza la lettera 'A', caricando '1' in accumulatore

141,244,5 --> Inserisce la lettera selezionata nell'accumulatore in centro allo schermo (1024+500 caratteri = 256*5+244)

169,29 --> Carica in accumulatore 29 per aprire la pagina grafica

141,24,208 --> Apre la pagina grafica(identico a poke 53272,29)

96 --> Restituisce il controllo al basic

PS: Per tornare alla modalità testuale digitare 'poke 53272,21'

Programma 23 - Ridisegnare un carattere e animarlo

Tratto di memoria in cui è sono contenute dalla lettera 'A' alla 'D'--> locazione da 12296 a 12327 (8X4=32-1 per la prima locazione)

Qui verranno caricati gli 8 bit(7+0) dei 4 caratteri che andranno a riscrivere le lettere A,B,C,D

```
10 for byte= 12296 to 12296+31

20 read info

30 poke byte, info

40 next byte
```

I comandi 'data' che contiengono i valori validi per ridisegnare i primi 4 caratteri alfabetici

```
100 data 238,204,204,252,204,216,112,0

101 data 127,127,73,104,63,31,1,0

102 data 0,30,54,102,126,102,102,238

103 data 0,128,248,252,22,146,254,254
```

Inserimento a partire dalla locazione 49152 per i 51 comandi di gestione del carattere(50+0 istruzioni)

```
200 k=49152

210 for n=k to k+50
```

220 read a

230 poke n,a

240 next n

250 sys k

INSERIAMO I CODICI MACCHINA NEI COMANDI 'DATA'...

32,32,229 --> Pulisce lo schermo

169,29 --> Carico 29 in accumulatore

141,24,208 --> Apre la pagina grafica(identico a poke 53272,29)

169,1 --> Visualizza la lettera 'A', caricando '1' in accumulatore

141,244,5 --> Inserisce la lettera selezionata nell'accumulatore in centro allo schermo (1024+500 caratteri = 256*5+244)

Temporizziamo l'animazione rallentandola con una funzione simile a 'for' nel basic

Inserisco nel registro 'y' il coefficiente di rallentamento 255

160,255 --> Il conteggio parte da 255

136 --> Decremento di una unità il registro 'y' andando a ritroso

208,253 --> Controllo che il registro 'y' sia uguale a '0' se si, prosegue il programma, altrimenti salta indietro di due codici fino alla funzione 136

169,2 --> Visualizza la lettera 'B', caricando '2' in accumulatore

141,244,5 --> Inserisce la lettera selezionata nell'accumulatore in centro allo schermo (1024+500 caratteri = 256*5+244)

Temporizziamo l'animazione rallentandola con una funzione simile a 'for' nel basic

Inserisco nel registro 'y' il coefficiente di rallentamento 255

160,255 --> Inizia il conteggio a partire da 255

136 --> Decremento di una unità il registro 'y'

208,253 --> Controllo che il registro 'y' sia uguale a '0' se si, prosegue il programma, altrimenti salta indietro di due codici fino alla funzione 136

169,3 --> Visualizza la lettera 'c', caricando '3' in accumulatore

141,244,5 --> Inserisce la lettera selezionata nell'accumulatore in centro allo schermo (1024+500 caratteri = 256*5+244)

Temporizziamo l'animazione rallentandola con una funzione simile a 'for' nel basic

Inserisco nel registro 'y' il coefficiente di rallentamento 255

160,255 -> Parte da 255

136 --> Decremento di una unità il registro 'y'

208,253 --> Controllo che il registro 'y' sia uguale a '0' se si, prosegue il programma, altrimenti salta indietro di due codici fino alla funzione 136

169,4 --> Visualizza la lettera 'D', caricando '4' in accumulatore

141,244,5 --> Inserisce la lettera selezionata nell'accumulatore in centro allo schermo (1024+500 caratteri = 256*5+244)

Temporizziamo l'animazione rallentandola con una funzione simile a 'for' nel basic

Inserisco nel registro 'y' il coefficiente di rallentamento 255

160,255 --> Parto con il conteggio da 255

136 --> Decremento di una unità il registro 'y'

208,253 --> Controllo che il registro 'y' sia uguale a '0' se si, prosegue il programma, altrimenti salta indietro di due codici fino alla funzione 136

76,10,192 --> Ripeto il ciclo d'animazione con un comando simile a 'goto' nel basic saltando alla locazione 49152(256*192) più 10 locazioni per iniziare esattamente dal punto in cui inizia il ciclo

PS1: per interrompere l'animazione premere 'run/stop + restore'

PS2: Per tornare alla modalità testuale digitare 'poke 53272,21'

Programma 24 - Routine kernal 65490: print chr$(ascii)

Funzione che stampa su schermo una sequenza di caratteri tipo 'print' del basic, richiamando la

routine kernal all'indirizzo 65490(210,255).

Tale funzione legge il valore presente nell'accumulatore interpretandolo come 'standard ascii' e non come visto prima in codice schermo

Vediamo un esempio minimo:

169,65 --> Carica il valore ascii della lettera 'A' in accumulatore

32,210,255 --> Salta attraverso il comando simile a 'gosub' all'indirizzo della routine simile a 'print'

96 --> restituisce il controllo al basic

Il programma appena descritto visualizza una sola 'A' dove si trova il cursore.

Vediamo un esempio dove vengono stampati più caratteri su schermo:

160,0 --> Carica nel registro Y il valore di inizio dei caratteri da visualizzare

185,14,192 --> Salta all'indirizzo successivo al programma, dove sono inseriti i caratteri da visualizzare

Quindi quindi li visualizza partendo da valore contenuto in Y

32,210,255 --> Salta attraverso il comando simile a 'gosub' all'indirizzo della routine simile a 'print'

200 --> aumenta di 1 il valore di Y per passare al carattere successivo

192,4 --> confronta y se uguale a 4(quantità dei caratteri da stampare su schermo)

208,245 --> se Y è diverso da 4 torna indietro di 10(255-10=245) posizioni e stampa un nuovo carattere

96 --> torna al basic

67,73,65,79 --> valori ascii corrispondenti ai caratteri 'C, I, A, O'

Programma 25 - FUNZIONE KERNAL 65220 -PLOT- PER POSIZIONARE IL CURSORE SULLO SCHERMO

ESEMPIO -PLOT- UN CARATTERE IN CENTRO ALLO SCHERMO:

160,20 --> Inserisce 20 nel registro Y per posizionare il cursore in quella colonna

162,11 --> Inserisce 11 nel registro X per posizionare il cursore in quella linea

24 --> Azzera il bit riposto del registro di stato

32,240,255 --> richiama la funzione 'PLOT' all'indirizzo 65520(240,255)

169,65 --> Carica il codice ascii per la lettera 'A'

32,210,255 --> Richiama la funzione 'PRINT'(descritta sopra) all'indirizzo 65490(210,255)

96 --> restituisce il controllo al BASIC

ESEMPIO -PLOT- VISUALIZZA PIU' CARATTERI IN CENTRO ALLO SCHERMO:

160,18 --> Inserisce 18 nel registro Y per posizionare il cursore in quella colonna

162,11 --> Inserisce 11 nel registro X per posizionare il cursore in quella linea

24 --> Azzera il bit riposto del registro di stato

32,240,255 --> richiama la funzione 'PLOT' all'indirizzo 65520(240,255)

160,0 --> Carica nel registro Y il valore di inizio dei caratteri da visualizzare

185,22,192 --> Salta all'indirizzo successivo al programma, dove sono inseriti i caratteri da visualizzare
Quindi quindi li visualizza partendo dal valore contenuto in Y

32,210,255 --> Salta attraverso il comando simile a 'gosub' all'indirizzo della routine simile a 'print'

200 --> aumenta di 1 il valore di Y per passare al carattere successivo

192,4 --> confronta y se uguale a 4(quantità dei caratteri da stampare su schermo)

208,245 --> se Y è diverso da 4 torna indietro di 10(255-10=245) posizioni e stampa un nuovo carattere

96 --> torna al basic

67,73,65,79 --> valori ascii corrispondenti ai caratteri 'C, I, A, O'

Programma 26 - FUNZIONE -GET- PER L'INPUT ROUTINE KERNAL 65508 e CARATTERI/NUMERI CASUALI

ESEMPIO -GET- attendo l'input di un carattere e lo visualizza:

32,68,229 --> ROUTINE DI CANCELLAZIONE DELLO SCHERMO

32,228,255 --> ROUTINE -GET- CHE CARICA IN ACCUMULATORE IL VALORE ASCII DEL TASTO PREMUTO

32,210,255 --> STAMPA SU SCHERMO IL CONTENUTO DELL'ACCUMULATORE

76,3,192 --> TORNA IL RIGO 3 CREANDO UN CICLO GOTO INFINITO

96 --> RENDE IL CONTROLLO AL BASIC(SUPERFLUA VISTO IL CICLO INFINITO)

CARATTERI CASUALI:

-USO DEL REGISTRO RASTER 53266 (18,208)

Il registro raster ridisegna tutte le linee dello schermo 24 volte ogni secondo, rappresenta

quindi un ottima fonte di caratteri pseudo casuali per la sua grande velocità

173,18,208 --> carica in accumulatore il valore sempre differente del registro raster

141,244,5 --> Stampa il carattere casuale di accumulatore in centro allo schermo

96 --> torna al basic

-USO DEI TRE REGISTRI DELL'OROLOGIO INTERNO DEL C64: 160(lento) - 161(medio) - 162(veloce)

Il c64 dispone di tre registri scanditi dall'orologio interno composti non dai secondi ma da valori grandi 256 unità. Al raggiungimento di 256 in 162, avanzerà di un'unità il 161. Quando il 161 raggiungerà anch'esso 256, il 160 avanzerà di un'unità.

173,162,0 --> carica in accumulatore il valore sempre differente registro 162 di pagina 0 dell'orologio

96 --> torna al basic

Programma 27 - ROUTINE KERNAL DI SCROLL 59626 (234,232) più altri comandi utili...

Sposta lo schermo di una linea verso l'alto

169,1 --> carica la 'A' in accumulatore

141,203,7 --> Stampa una il contenuto dell'accumulatore su schermo

141,218,7 --> Stampa una il contenuto dell'accumulatore su schermo in una seconda posizione

32,234,232 --> Chiama la routine di scroll

76,0,192 --> salta all'inizio del programma 'goto'

96 --> rende il controllo al basic

ALTRE ROUTINE UTILI

58726(102,229) --> Porta il cursore all'inizio dello schermo

58692(68,229) --> cancella lo schermo

59903(255,233) --> cancella una linea dello schermo specificata nell'indice X

SALVARE I PROGRAMMI IN LINGUAGGIO MACCHINA, CON E SENZA BASIC

Il primo modo di salvare il contenuto della memoria dove è situato il programma LM è quello di utilizzare il comando SAVE"PROGRAMMA",8,1(o ,1,1 se su cassetta).

1) si digita il listato in basic con i relativi comandi DATA in LM

2) si cancella il programma basic digitando NEW

3) quindi con SAVE"NOME",8,1

4) si spegne e riaccende il c64

5) con LOAD"NOME",8,1 si carica il programma LM

6) se il codice scritto partiva da 49152, allora si digita sys 49152 per eseguire

Questa tecnica presenta alcun incovenienti legati alla gesgtioni dei puntatori per gli indirizi di memoria Per tanto sarà bene usare il programma basic che segue per il migliore salvataggio del codice LM:

```
1000 input"salvo su nastro o su disco(n/d)";x$
1020 if x$= "n" then poke 781,1
1030 if x$= "d" then poke 781,8
1040 sys65466: input"nome del programma";a$
1050 k=53200:for n=1 to len(a$): poke k+n-1,asc(mid$(a$,n,1)): next
```

```
1060 poke 780,len(a$): poke 781,208: poke 782,207: sys 65469

1070 input"indirizzo iniziale: rigo";b%

1080 input"indirizzo iniziale: pagina";c%

1090 input"indirizzo finale: rigo";d%

1100 input"indirizzo finale: pagina";e%

1110 poke 251,b%: poke 252,c%: poke 780,251: poke 781,d%: poke 782,e%: sys65496
```

Il listato illustrato permette di salvare su cassetta o nastro il codice LM, inoltre di indicare

riga e pagina iniziali e finali. Per sicurezza è meglio indicare una riga in più finale per la chiusura del salvataggio.

Per caricare si userà LOAD"NOME",8,1 è per lanciare SYS all'indirizzo di partenza

L'ultima strategia utile per salvare sia il programma basic che quello in LM in un unico file è:

1) digitare il listato basic per il caricamento dei codici LM con i DATA. O caricarlo da un file

2) eseguite il programma per caricare i valori LM in memoria

3) quindi cancellate il programma basic con NEW

4) digitate in basic '1 SYS 49152' ed otterrete in basic la linea di avvio per il RUN

5) digitare in modo diretto PRINT PEEK(45); PEEK(46) per visualizzare i puntatori di termine basic per riga e pagina

6) modificare i puntatori digitando in modo diretto l'indirizzo di termine del programma LM:

es indirizzo: 49152+14

useremo POKE 45,15 --> rigo 14 + 1 per sicurezza

useremo POKE 46,192 --> pagina 192

7) Dopo aver fissato i puntatori per il salvataggio, scrivere SAVE"NOME",8,1

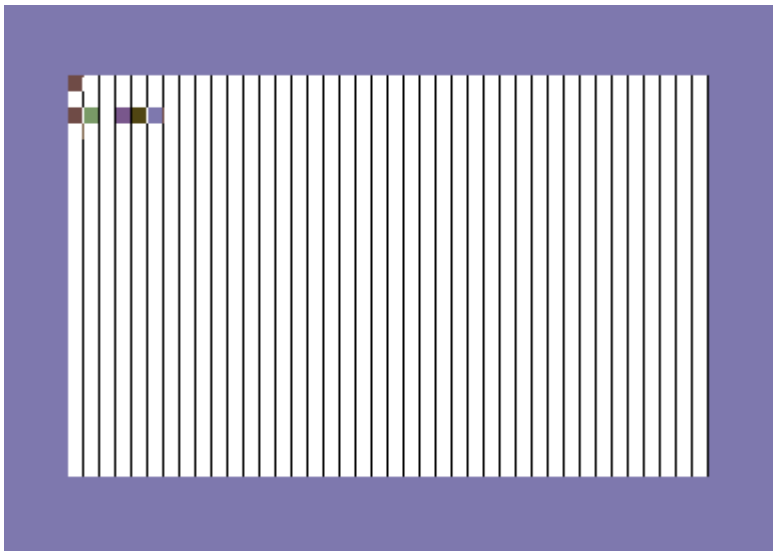
8) spegnere e riaccendere il computer, quindi caricare il programma con LOAD"nome",8,1

9) digitate LIST e comparirà la linea basic per l'avvio con il comando RUN. digitarlo e premere INVIO

Soluzione valida se il software scritto viene salvato nella prima posizione dei file sul dischetto. Così

attraverso il comando diretto LOAD"*",8,1 partirà automaticamente il programma.

Programma 28 - MODALITA' GRAFICA BITMAP MONOCROMATICA



```
Attivare una pagina grafica di 320x200 pixel a due colori e disegnare dei simboli

Il presente programma disegna in modalità grafica molte linee verticali di un pixel
come esempio.

In questo caso il programma verrà caricato a partire dall'indirizzo 21248



Attiva la modalità bitmap



169,29          --> carica 29 in accumulatore

157,24,208      --> aggiunge il contenuto dell'accumulatore per attivare la modalità
bitmap all'indirizzo 53272(24,208)

169,59          --> carica 59 in accumulatore

157,17,208      --> aggiunge l'accumulatore all'indirizzo 53265(17,208) per scegliere
se usare la prima o la seconda parte dei 16k dedicati alla modalità grafica



Parte di assegnazione del colore(da 1024 a 2043)



169,0           --> carico 0 in accumulatore (assegna il rigo 0 d'inizio
indirettamente)

133,251         --> stampa accumulatore all'indirizzo 251 di pagina 0
```

169,4 --> carica 4 in accumulatore (pagina indiretta 1024=256*4 di inizio)

133,252 --> stampa accumulatore all'indirizzo 252 di pagina 0

162,4 --> carica 4 in X (conta 1024=256*4 caratteri all'indietro)

160,1 --> carica 1 in Y, per assegnare il colore (4 bit per il primo colore e 4 bit per il secondo colore)

152 --> copia in accumulatore il contenuto di Y

145,251 --> stampa accumulatore in un indirizzo contenuto indirettamente nella locazione 251 di pagina 0

200 --> aggiunge 1 al valore contenuto in Y

208,251 --> se Z del registro di stato è 0 salta indietro di 4 locazioni(255-251=4)

202 --> diminuisce di 1 il valore contenuto in X

240,4 --> salta avanti di 5 locazioni se Z del registro di stato è 1 (0,1,2,3,4)

230,252 --> incrementa di 1 il valore contenuto nella locazione 252 di pagina 0

208,244 --> se Z del registro di stato è 0 salta indietro di 11 locazioni(255-244=11)

Disegna 8192 bit dello schermo bitmap

169,0 --> carico 0 in accumulatore (assegna il rigo 0 d'inizio indirettamente)

133,251 --> stampa accumulatore all'indirizzo 251 di pagina 0

169,32 --> carica 32 in accumulatore (pagina indiretta 8192=256*32 di inizio)

133,252 --> stampa accumulatore all'indirizzo 252 di pagina 0

162,32 --> carica 32 in X (conta 8192=256*32 bit all'indietro)

160,1 --> carica 1 in Y, così che disegna una matrice di 8 bit

152 --> copia in accumulatore il contenuto di Y

145,251 --> stampa accumulatore in un indirizzo contenuto indirettamente nella locazione 251 di pagina 0

200 --> aggiunge 1 al valore contenuto in Y

208,251 --> se Z del registro di stato è 0 salta indietro di 4 locazioni(255-251=4)

202 --> diminuisce di 1 il valore contenuto in X

```
240,4           --> salta avanti di 5 locazioni se Z del registro di stato è 1
(0,1,2,3,4)

230,252         --> incrementa di 1 il valore contenuto nella locazione 252 di pagina
0

208,244         --> se Z del registro di stato è 0 salta indietro di 11
locazioni(255-244=11)

96              --> restituisce il controllo al basic
```

Programma 29 - LISTATO NOTA MUSICALE

C64 -> RANGE FREQUENZA -> DA 0.6 A 3995 Hz

SID = Sound Interface Device

Frequenza Voce 1= 54272 + 54273

Frequenza Voce 2= 54279 + 54280

Frequenza Voce 3= 54286 + 54287

Valore da assegnare ai due registri per comporre la frequenza in pratica:

Es: voce 1

hz 440 -> 3995 / 65536 * 440 = 7218

quindi: 440 = 7218

7218 è da distribuire fra i due registri di 8 bit(256 unità)

7218 / 256 = 28,1953125

registro 54273 -> 28 (parte intera)

-- 0,1953125 * 256 = 50 --

registro 54272 -> 50 (parte dopo la virgola)

Forma d'onda:

-Voce 1 -> 54276:

-Triangolare valore 17 attiva e 16 rilascia

-Dente di sega valore 33 attiva e 32 rilascia

-Impulso valore 65 attiva e 64 rilascia

-Rumore bianco valore 129 attiva e 128 rilascia

-Voce 2 -> 54283:

-Triangolare valore 17 attiva e 16 rilascia

-Dente di sega valore 33 attiva e 32 rilascia

-Impulso valore 65 attiva e 64 rilascia

-Rumore bianco valore 129 attiva e 128 rilascia

-Voce 3 -> 54290:

-Triangolare valore 17 attiva e 16 rilascia

-Dente di sega valore 33 attiva e 32 rilascia

-Impulso valore 65 attiva e 64 rilascia

-Rumore bianco valore 129 attiva e 128 rilascia

Larghezza dell'impulso:

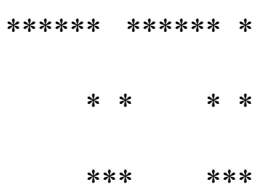
-Voce 1 -> indirizzi 54275 e 53276, con valore suddiviso in due parti da 256 ciascuno

-Voce 2 -> indirizzi 54282 e 53283, con valore suddiviso in due parti da 256 ciascuno

-Voce 3 -> indirizzi 54286 e 53287, con valore suddiviso in due parti da 256 ciascuno

La base del valore da inserire è la percentuale d'ampiezza:

quindi se vorremo un impulso per il 75% alto e 25% basso dovremo svolgere i seguenti calcoli



40.95 * valore percentuale = risultato da arrotondare

risultato arrotondato / 256 = parte intera nel all'indirizzo 54276

parte dopo la virgola * 256 = inserire il risultato all'indirizzo 54275

es:

75*40.95 = 3071.25 (arrotondamento a 3071)

3071 / 256 = 11 -> all'indirizzo 54276

0.99609375 * 256 = 255 -> all'indirizzo 54275

VOLUME ADSR(ATTACCO, DECADIMENTO, SOSTEGNO, RILASCIO)

-Voce 1 -> indirizzo 54277 per attacco(valore da 0 a 15) e decadimento(valore da 16 a 255)

-Voce 1 -> indirizzo 54278 per sostegno(valore da 0 a 15) e rilascio(valore da 16 a 255)

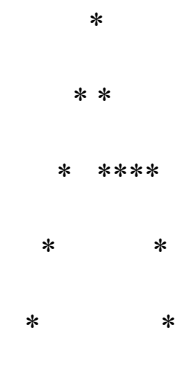
-Voce 2 -> indirizzo 54284 per attacco(valore da 0 a 15) e decadimento(valore da 16 a 255)

-Voce 2 -> indirizzo 54285 per sostegno(valore da 0 a 15) e rilascio(valore da 16 a 255)

-Voce 3 -> indirizzo 54291 per attacco(valore da 0 a 15) e decadimento(valore da 16 a 255)

-Voce 3 -> indirizzo 54292 per sostegno(valore da 0 a 15) e rilascio(valore da 16 a 255)

Forma d'onda:



atta, dec, sos, ril

In pratica somma del primo nibble(da 0 a 15) con secondo nibble(16 a 255)

ES:

suono 100% -> 50% Attacco(8), 25% decadimento(60), 25% sostegno(4), 50% rilascio(120)

avremo per voce 1:

54278 con 8+60 = 68

54279 con 4+120 = 124

VOLUME COMPLESSIVO DELLE TRE VOCI

Indirizzo 54296 con valore da 0 a 15

Listato in Linguaggio macchina per la produzione di una nota:

169,9 -> ATTACCO

141,5,212

169,0 -> RILASCIO

141,6,212

169,15 -> VOLUME MASSIMO

141,24,212

169,33 -> ATTIVA IMPULSO AL BYTE PIU' SIGNIFICATIVO

141,4,212

169,25 -> SETTAGGIO FREQUENZA VOCE 1 PER BYTE PIU' SIGNIFICATIVO(NOTA) - DENTE DI SEGA

141,1,212

160,255 -> PAUSA PER DURATA SUONO CON FOR ANNIDATO

162,0

232

224,255

208,251

136

208,246

169,32 -> RILASCIA IMPULSO PER NOTA A DENTE DI SEGA

141,4,212

160,255 -> PAUSA CON FOR ANNIDATO PER SEPARARE DA UNA NUOVA EVENTUALE NOTA

162,0

232

224,255

208,251

136

208,246

96 -> RESTITUISCE IL CONTROLLO AL BASIC

Programma 30 - Visualizzare su schermo un'immagine bitmap koala

Su C64 sono state create immagini bellissime con i sedici colori complessivi sulla sua risoluzione di 160x200 pixel. Immagini studiate fin nei minimi dettagli per sfruttare al meglio quello che il computer aveva a disposizione. I grafici che si sono susseguiti in tutta la storia di questa macchia hanno saputo creare delle vere opere d'arte!

In questa parte impareremo a gestire le immagini grafiche create con koala painter o simili nei programmi scritti in linguaggio macchina...

Segue elenco dei codici macchia da inserire attraverso il costrutto 'read-poke-data', a partire dalla locazione 49152:

169,0 - RIGA DI LETTURA

133,251 - IMMAGAZZINA IN PAGINA 0 LA RIGA DI LETTURA

169,96 - PAGINA DI LETTURA

133,252 - IMMAGAZZINA IN PAGINA 0 LA PAGINA DI LETTURA

169,0 - RIGA DI SCRITTURA

133,253 - IMMAGAZZINA IN PAGINA 0 LA RIGA DI SCRITTURA

169,32 - PAGINA DI SCRITTURA

133,254 - IMMAGAZZINA IN PAGINA 0 LA PAGINA DI SCRITTURA

160,255 - VALORE DA RAGGIUNGERE PER IL CONTEGGIO DELLE RIGHE DI

MEMORIA

162,11 - CARICA IN x LE POSIZIONI MANCANTI PER TROVARE L'INDIRIZZO 0,96

161,240 - CARICA IN ACCUMULATORE IL CONTENUTO DI 0,96

141,0,200 - SALVA ACCUMULATORE ALLA LOCAZIONE 51200

173,0,200 - CARICA IN ACCUMULATORE IN CONTENUTO DELLA LOCAZIONE 51200

162,13 - CARICA IN x LE POSIZIONI MANCANTI PER TROVARE L'INDIRIZZO 0,32

129,240 - CARICA ACCUMULATORE IN 0,32 (8192)

230,251 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 251(SORGENTE)

230,253 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 253(DESTINAZIONE)

196,251 - CONFRONTA IL CONTENUTO DI Y, UGUALE AL CONTENUTO DELLA RIGA 252 DI PAGINA ZERO ALLORA FERMA IL CICLO

208,234 - SALTA AL ISTRUZIONE 162,11 PER SVOLGERE TUTTO CON UNA RIGA DI MEMORIA IN PIU'

160,159 - VALORE FINALE DEL CONTEGGIO SULLE PAGINE CON L'INDICE Y

230,252 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 252(SORGENTE)

230,254 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 254(DESTINAZIONE)

196,252 - CONTROLLA SE LA LOCAZIONE 252 DI PAGINA ZERO HA ASSUNTO IL VALORE DELL'INDICE Y

208,222 - SE NON E' STATA RAGGIUNTA L'ULTIMA PAGINA(96+63=63ESIMA) INDICATA NEL INDICE Y RIPETE IL CICLO FINO ALL'ISTRUZIONE 160,255

– ALTRO CICLO –

169,64 - RIGA DI LETTURA

133,251 - IMMAGAZZINA IN PAGINA 0 LA RIGA DI LETTURA

169,127 - PAGINA DI LETTURA

133,252 - IMMAGAZZINA IN PAGINA 0 LA PAGINA DI LETTURA

169, 0 - RIGA DI SCRITTURA

133,253 - IMMAGAZZINA IN PAGINA 0 LA RIGA DI SCRITTURA

169,4 - PAGINA DI SCRITTURA

133,254 - IMMAGAZZINA IN PAGINA 0 LA PAGINA DI SCRITTURA

160,255 - VALORE DA RAGGIUNGERE PER IL CONTEGGIO DELLE RIGHE DI MEMORIA

162,11 - CARICA IN x LE POSIZIONI MANCANTI PER TROVARE L'INDIRIZZO

161,240 - CARICA IN ACCUMULATORE IL CONTENUTO

141,0,200 - SALVA ACCUMULATORE ALLA LOCAZIONE 51200

173,0,200 - CARICA IN ACCUMULATORE IN CONTENUTO DELLA LOCAZIONE 51200

162,13 - CARICA IN x LE POSIZIONI MANCANTI PER TROVARE L'INDIRIZZO

129,240 - CARICA ACCUMULATORE

230,251 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 251(SORGENTE)

230,253 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 253(DESTINAZIONE)

196,251 - CONFRONTA IL CONTENUTO DI Y, UGUALE AL CONTENUTO DELLA RIGA 252 DI PAGINA ZERO ALLORA FERMA IL CICLO

208,234 - SALTA AL ISTRUZIONE 162,11 PER SVOLGERE TUTTO CON UNA RIGA DI MEMORIA IN PIU'

160,130 - VALORE FINALE DEL CONTEGGIO SULLE PAGINE CON L'INDICE Y

230,252 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 252(SORGENTE)

230,254 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 254(DESTINAZIONE)

196,252 - CONTROLLA SE LA LOCAZIONE 252 DI PAGINA ZERO HA ASSUNTO IL VALORE DELL'INDICE Y

208,222 - SE NON E' STATA RAGGIUNTA L'ULTIMA PAGINA(96+63=63ESIMA) INDICATA NEL INDICE Y RIPETE IL CICLO FINO ALL'ISTRUZIONE 160,255

– ALTRO CICLO –

169,40 - RIGA DI LETTURA

133,251 - IMMAGAZZINA IN PAGINA 0 LA RIGA DI LETTURA

169,131 - PAGINA DI LETTURA

133,252 - IMMAGAZZINA IN PAGINA 0 LA PAGINA DI LETTURA

169, 0 - RIGA DI SCRITTURA

133,253 - IMMAGAZZINA IN PAGINA 0 LA RIGA DI SCRITTURA

169,216 - PAGINA DI SCRITTURA

133,254 - IMMAGAZZINA IN PAGINA 0 LA PAGINA DI SCRITTURA

160,255 - VALORE DA RAGGIUNGERE PER IL CONTEGGIO DELLE RIGHE DI MEMORIA

162,11 - CARICA IN x LE POSIZIONI MANCANTI PER TROVARE L'INDIRIZZO

161,240 - CARICA IN ACCUMULATORE IL CONTENUTO

141,0,200 - SALVA ACCUMULATORE ALLA LOCAZIONE 51200

173,0,200 - CARICA IN ACCUMULATORE IN CONTENUTO DELLA LOCAZIONE 51200

162,13 - CARICA IN x LE POSIZIONI MANCANTI PER TROVARE L'INDIRIZZO

129,240 - CARICA ACCUMULATORE

230,251 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 251(SORGENTE)

230,253 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 253(DESTINAZIONE)

196,251 - CONFRONTA IL CONTENUTO DI Y, UGUALE AL CONTENUTO DELLA RIGA 252 DI PAGINA ZERO ALLORA FERMA IL CICLO

208,234 - SALTA AL ISTRUZIONE 162,11 PER SVOLGERE TUTTO CON UNA RIGA DI MEMORIA IN PIU'

160,134 - VALORE FINALE DEL CONTEGGIO SULLE PAGINE CON L'INDICE Y

230,252 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 252(SORGENTE)

230,254 - INCREMENTA DI UNA UNITA' IL CONTENUTO DI PAGINA ZERO LOCAZIONE 254(DESTINAZIONE)

196,252 - CONTROLLA SE LA LOCAZIONE 252 DI PAGINA ZERO HA ASSUNTO IL VALORE DELL'INDICE Y

208,222 - SE NON E' STATA RAGGIUNTA L'ULTIMA PAGINA(96+63=63ESIMA) INDICATA NEL INDICE Y RIPETE IL CICLO FINO ALL'ISTRUZIONE 160,255

- ISTRUZIONI DI VISUALIZZAZIONE –

169,29

141,24,208 – EQUIVALE A POKE 53272,29

169,59

141,17,208 – EQUIVALE A POKE 53265,59

169,16

141,22,208 – EQUIVALE A POKE 53270,16

185,16,135

153,33,208 – EQUIVALE A POKE 53281, PEEK (34576)

96 - RENDE IL CONTROLLO AL BASIC

La soluzione di visualizzazione di una immagine grafica koala qui riportata, è una delle più semplici. Non è però esente da problemi relativi alla memoria 'sporca' che può alterare la visualizzazione. Così come per la pulizia dei caratteri precedentemente digitati. Tale esempio però spero possa essere utile ad apprendere i rudimenti, certamente perfezionabili, della programmazione grafica con il C64.

Ho sviluppato un tutorial apposito, rivolto alla conversione di immagini grafiche nei formati moderni(jpg, bmp...) verso la vecchia estensione Koala(.koa) visualizzabile su su Commodore 64!

Tutte le immagini qui riportate nel manuale sono esposte per come le vedreste sulle schermo di un vero Commodore 64. Perfino l'immagine di copertina è stata convertita per essere vista come su un vero C64...

La guida potete trovarla sulla mia pagina web:

http://www.bertinettobartolomeodavide.it/programmazione/commodore64

Conclusione

Dopo questa lunga carrellata di listati termina il mio percorso verso l'apprendimento di questo primordiale linguaggio di programmazione. Tanti anni ad esaminare molti dettagli, sempre in ritagli di tempo durante i miei periodi di vacanze, che mi hanno regalato grandi soddisfazioni.

Ancor più bella è però ora la la possibilità di diffondere il materiale chiave ad altri appassionati per realizzare molti lavori su questa bellissima piattaforma! Perché il C64 vive ancora!

Può sembrare tutto un po' a scoppio ritardato rispetto all'era boom del piccolo computer Commodore ed al tempo in cui erano in voga questo genere di lavori. Tutto però può servire ed è anche per questo che vi ho presentato manuale che avete sotto gli occhi.

Nella speranza che ogni parte del presente ebook sia stata concretamente realizzabile dall'utente per ottenere un reale insegnamento, ringrazio i lettori per aver consultato questo mio lavoro!

Il file D64 contente ogni listato sviluppato sopra è disponibile a questo indirizzo web in formato zip coperto da password:

http://www.bertinettobartolomeodavide.it/programmazione/commodore64/floppy lm.zip

La chiave d'apertura è(attenti, senza spazi e con la prima lettera maiuscola!!!): **Commodore64**

Grazie per aver preso visione di questo mio sforzo iniziato da una grande passione fin dall'infanzia, molti anni fa...

Bertinetto Bartolomeo Davide