L'Assembler del TI 99

Per prima cosa è opportuno chiarire quale sia la differenza tra linguaggio Assembler e linguaggio macchina poiché a tal proposito molte persone hanno le idee confuse.

L'Assembler è un linguaggio simbolico a basso livello; rispetto ad un linguaggio ad alto livello come il Basic questo implica una maggiore difficoltà di scrittura dei programmi a causa della loro struttura analitica spinta, ma rende molto più veloce la loro esecuzione e permette di accedere a tutte le risorse potenziali del sistema. L'Assembler normalmente è un linguaggio compilato, e pertanto richiede l'uso di un Editor per creare il file testo contenente le istruzioni composte da codici mnemonici e da operandi esprimibili anche per mezzo di nomi simbolici; è indispensabile inoltre l'uso del sistema di memorizzazione a dischi dal quale vengono caricate le varie utility del compilatore e tramite il quale vengono svolte le fasi intermedie di passaggio dal programma sorgente in Assembler a quello oggetto in binario (fase di assemblaggio o compilazione).

Il linguaggio macchina invece non è altro che la programmazione immediata della macchina tramite codici esadecimali (più pratici da impiegare di quelli binari), i quali permettono rispetto all'Assembler di saltare in pratica la fase della compilazione pur andando incontro a notevoli difficoltà di ordine pratico, quali quella di dover tener conto manualmente degli indirizzi ai quali si fa riferimento all'interno del programma e quella di non poter impiegare codici mnemonici per le istruzioni o nomi simbolici per i dati.

Non tutti i personal in commercio hanno disponibile un compilatore Assembler, quasi tutti hanno però implementate le istruzioni PEEK e POKE che permettono l'uso di routine in linguaggio macchina richiamabili da programmi Basic.

Il TI 99, come al solito, tiene fede alla sua fama di essere un caso anomalo e pertanto se da una parte offre due tipi di Assembler (l'altro è quello tradotto riga per riga fornito con la Mini Memory) dall'altra non permette l'uso delle suddette istruzioni né tramite il TI-Basic né tramite l'Extended Basic. È vero che nell'Extended esistono i comandi CALL INIT (controlla se è collegata l'espansione da 32K e in caso affermativo vi carica le routine di supporto), CALL LOAD (carica il programma nella Ram da 32K), CALL LINK (passa l'esecuzione ed eventuali parametri al sottoprogramma in Assembly) e CALL PEEK (ritorna il valore della locazione di memoria) ma il loro impiego richiede l'espansione di memoria e permette di richiamare ed eseguire programmi preparati con il compilatore Assembler, ma non la loro scrittura.

Ovviamente non è possibile in questa sede tenere un corso completo sul linguaggio Assembler del microprocessore TMS 9900, vogliamo però dedicare qualche pagina alla descrizione sommaria della struttura interna del TI 99 e chiarire a grandi linee in che modo sia possibile ottenere delle funzioni (specie grafiche) non accessibili tramite il linguaggio residente.

La mappa di memoria

Come già accennato in precedenza il microprocessore del TI 99/4A è il 16 bit TMS 9900. La memoria massima che esso può indirizzare direttamente è pari a 64K. 32K sono costituiti dalla scheda di espansione Ram in due sezioni: una da 8K (Low Memory) e una da 24K (High Memory), 8K dalle Rom presenti sui moduli di comando SSS eventualmente inseriti nello scivolo della consolle, 8K dalle due Rom di sistema contenenti parte del sistema operativo e dell'interprete Basic, 8K dalle Rom contenute nelle schede di controllo periferiche del rack di espansione (dischi e stampanti) ed infine gli ultimi 8K dai registri di comunicazione con le VDP Ram, le Grom, il microprocessore grafico TMS 9918A, quello musicale TMS 9919 e con il sintetizzatore vocale. Ora dal momento che la consolle di base non è ovviamente dotata né di espansione Ram da 32K né di interfacce per dispositivi periferici e che normalmente non viene inserito nessun modulo SSS per la programmazione in TI-Basic, ecco che 48K della memoria indirizzabile direttamente (64K) sono inizialmente vacanti; viene spontaneo domandarsi dove diavolo verrà memorizzato il programma con i relativi dati.

Una possibile risposta sarebbe FIFT (acronimo per Fatti I Fatti Tuoi) ma dal momento che ciò non risolverebbe il problema da un punto di vista strettamente tecnico, sarà opportuno prendere nota del fatto che il programma e le variabili Basic trovano posto sulle VDP Ram, ossia sui famosi 16K costituiti da Ram Video Display Processor indirizzate indirettamente, metodo usato anche per accedere alle Grom da 6K (Graphic Read Only Memory) contenute sia nella consolle (Grom 0-1-2) che nei moduli SSS (Grom 3-4-5-6-7); tali Grom sono programmate in GPL (Graphic Programming Language) che nel caso delle Grom di sistema provvede a gestire la parte del sistema operativo e dell'interprete Basic non contenuta nelle Rom da 8K della memoria CPU. Facendo un rapido calcolo abbiamo 16K di VDP Ram, 48K di Grom (6K × 8) e 64K di memoria CPU, ossia in totale il TI 99 può gestire direttamente o indirettamente 128K di memoria.

A questo punto vi sveliamo il motivo per cui non è possibile programmare il Texas in linguaggio Assembly nella sua versione base: le routine in linguaggio macchina non possono essere memorizzate sulle VDP Ram a causa del loro indirizzamento indiretto e d'altra parte non esiste Ram della memoria CPU disponibile per tale funzione. Unica soluzione al problema è la Mini Memory che impiega gli 8K di Rom SSS (>6000->7FFF) in modo diverso dagli altri moduli, prevedendo una Ram al posto della seconda Rom (>7000-

MAPPA MEMORIA CPU E GROM

CPU

0000 – 1FFF	ROM DI SISTEMA
2000 – 3FFF	8K ESPANS. RAM
4000 – 5FFF	ROM PERIFERICHE
6000 – 7FFF	ROM MODULI SSS
8000 – 9FFF	ZONA COMUNICAZ.
A000 – FFFF	24K ESPANS. RAM

GROM

0000 – 17FF	GROM 0
2000 – 37FF	GROM 1
4000 – 57FF	GROM 2
6000 – 77FF	GROM 3 (SSS)
8000 – 97FF	GROM 4 »
A000 – B7FF	GROM 5 »
C000 – D7FF	GROM 6 »
E000 – F7FF	GROM 7 »

Questa figura indica in quale modo sia divisa la memoria CPU del TI 99 (ossia quella indirizzabile direttamente dal TMS 9900) e la memoria Grom, costituita da otto blocchi da 6K di cui tre presenti nella consolle ed altri cinque eventualmente nei moduli SSS.

>7FFF). Su tale Ram "veloce" e con l'ausilio della Rom e della Grom interna è possibile memorizzare un programma Assembly prodotto dall'Assembler line-by-line fornito su nastro che oltretutto non richiede necessariamente l'impiego del dispositivo di memorizzazione a dischi. Come già detto su MC n.17 evitate però di acquistare la Mini Memory se prima non avete avuto l'opportunità di reperire il solo manuale dell'Editor/Assembler senza il quale la sua utilizzazione è praticamente impossibile.

Da tener presente infine che ogni linguaggio utilizza la Mappa di Memoria in modo differente. Nelle illustrazioni è riportata una doppia configurazione per le VDP Ram; la prima si riferisce all'impiego del Basic, la seconda a quello dell'Assembler in Bit-Map Mode (vedremo tra poco il significato di tale termine).

Il TI 99 e la grafica

In redazione abbiamo ricevuto molte richieste di chiarimenti sulle capacità grafiche del Texas e se ed eventualmente in che modo sia possibile indirizzare il singolo Pixel senza dover ridefinire l'intera matrice del singolo carattere (costituito da 64 punti disposti su 8 righe per 8 colonne) tramite l'istruzione CALL CHAR e l'assegnazione di una stringa esadecimale di 16 caratteri ad un codice ASCII.

La risposta è complessa perché in via teorica è possibile studiare un programma Basic che sia in grado di accendere un singolo punto dello schermo tramite delle coordinate di riferimento, ma la lentezza della sua esecuzione sarebbe enorme e la zona dello schermo che si vuole rendere grafica dovrebbe essere ridotta a (159-32)+1 = 128 caratteri i quali disposti rettangolarmente formerebbero un quadro di 11 × 11 caratteri pari a 88 × 88 Pixel con la rimanenza di 7 codici ASCII. In realtà siamo riusciti, dopo non poche difficoltà, ad ottenere quanto sopra descritto; per il listing e le spiegazioni relative dobbiamo però rimandarvi al nostro prossimo appuntamento, per avere il tempo di ottimizzare il programma.

Continuiamo il nostro discorso sulle potenziali capacità grafiche del TI 99, sfruttabili però, lo ripetiamo ancora una volta, solo tramite programmi Assembler (Mini Memory o compilatore).

La gestione dello schermo può essere fatta in 4 modi:

1) Modo grafico

Dal momento che è il tipo di funzionamento standard (impiegato per il TI-Basic), non ci soffermeremo sulla sua descrizione generica.

MAPPA MEMORIA VDP RAM

VDP RAM in Basic

0000 – 02FF	TAVOLA VIDEO
0300 – 031F	TAVOLA COLORI-SPR.
0320 – 03BD	BUFFER BASIC
03BE – 03FF	AREA LAVORO
0400 – 05FF	TAVOLA CARATTERI
0600 – 3FFF	PROGRAMMA E DATI

VDP RAM in Bit Map

0000 – 17FF	TAVOLA FORME
1800 – 1AFF	TAVOLA VIDEO
1B00 – 1FFF	TAVOLA SPRITE
2000 – 37FF	TAVOLA COLORI
3800 – 3FFF	BUFFER I-O

Le VDP Ram del Texas sono le 4116 da 16K per 1 Bit di memoria dinamica. La loro utilizzazione varia a seconda del linguaggio di programmazione usato ed al modo grafico selezionato nel caso dell'impiego dell'Assembler. La tabella riporta gli indirizzi del Basic e dell'Assembler in Bit Map Mode.

2) Modo multicolore

Se quando avete acquistato la vostra consolle il rivenditore vi ha inserito il modulo Diagnostic per provarla, forse vi sarà capitato di vedere sull'intero schermo una serie di rettangoli o "mattoncini" colorati nelle 16 tinte disponibili; questo è il modo multicolore, nel quale il video viene suddiviso in 64 colonne per 48 linee. Ogni rettangolo così definito è costituito da un quarto della matrice originale del carattere (4 × 4 pixel), ma in tal caso ne può essere specificato solo il colore interno.

Ci rendiamo conto di aver chiamato rettangolino una figura formata da 4 × 4 punti che per definizione sarebbe più corretto definire quadratino, ma dal momento che purtroppo l'ampiezza orizzontale del pixel è maggiore di quella verticale, l'effetto ottico prodotto rende più efficace l'uso del primo termine.

3) Modo testo

È possibile impiegare i caratteri standard ASCII più altri definibili dall'utente; i colori disponibili sono limitati a due: uno per lo sfondo e uno per i caratteri. Lo schermo è formato da 40 colonne per 24 linee. Attualmente questo modo grafico viene impiegato solo dagli Editor dei linguaggi compilati.

4) Modo bit-map

Eccoci giunti alla tanto desiderata possibilità di indirizzare il singolo punto ottenendo dei grafici con una velocità di esecuzione accettabile. La definizione è pari a 256 × 192 pixel per un totale di 49.152 punti per ciascuno dei quali è possibile specificare uno qualsiasi dei 16 colori disponibili con la sola limitazione di non indicarne più di due diversi nell'ambito di un gruppo di otto pixel adiacenti sulla stessa riga.

Da notare infine che è possibile impiegare, tranne che nel modo testo, fino a 32 Sprite. Per chi non lo sapesse lo Sprite è un carattere definito dall'utilizzatore che può indicarne oltre che la forma anche la direzione e la velocità del suo movimento sullo schermo e che una volta creato non ha più bisogno di essere controllato dal programma che lo ha generato.

La gestione dello schermo

A questo punto sicuramente starete morendo dalla curiosità di sapere come sia possibile selezionare uno dei quattro modi grafici previsti e di come sia gestita nella realtà la memoria dedicata al video.

Ebbene esistono 8 registri a 8 bit (VDP Write Only Register) i quali si occupano, a seconda della loro impostazione, di predisporre la divisione della memoria VDP Ram in modo opportuno alla gestione del tipo di grafica selezionata e del linguaggio impiegato. A titolo indicativo diremo che per predisporre il TI 99 al Bit-Map Mode occorre porre il bit n. 6 del Registro 0 a 1 ed i bit n. 3 e 4 del Registro 1 a 0. Per l'indirizzamento dei punti dello schermo e la memorizzazione delle forme e dei colori da visualizzare, il microprocessore grafico TMS 9918A impiega tre zone di memoria VDP Ram chiamate Tavole, ossia la Tavola della Mappa Video, la Tavola delle Forme e quella dei Colori. Queste tavole hanno una struttura diversa in relazione al tipo di modo grafico operativo. Faremo riferimento al Bit-Map dal momento che è sicuramente quello più interessante.

Tavola della Mappa Video

La tavola è divisa in tre sezioni contenenti ciascuna 256 informazioni lunghe un byte. Il primo byte è posto alla locazione >1800 delle VDP Ram. Ogni byte contiene il nome del carattere che deve essere visualizzato nella corrispondente posizione dello schermo e che è costituito da un numero esadecimale da >00 a >FF pari

appunto a 256 nomi per ogni sezione. La descrizione della forma e dei colori si troverà nelle tavole rispettive in corrispondenza del relativo nome. In pratica il TMS 9918A andrà a vedere quale sia il nome del carattere che occupa la prima posizione in alto a sinistra dello schermo, ne cercherà la forma (matrice 8×8) e i colori nelle tavole relative e quindi lo mostrerà sul video con le caratteristiche indicate; il ciclo si ripeterà fino alla completa visualizzazione dello schermo.

Tavola delle Forme

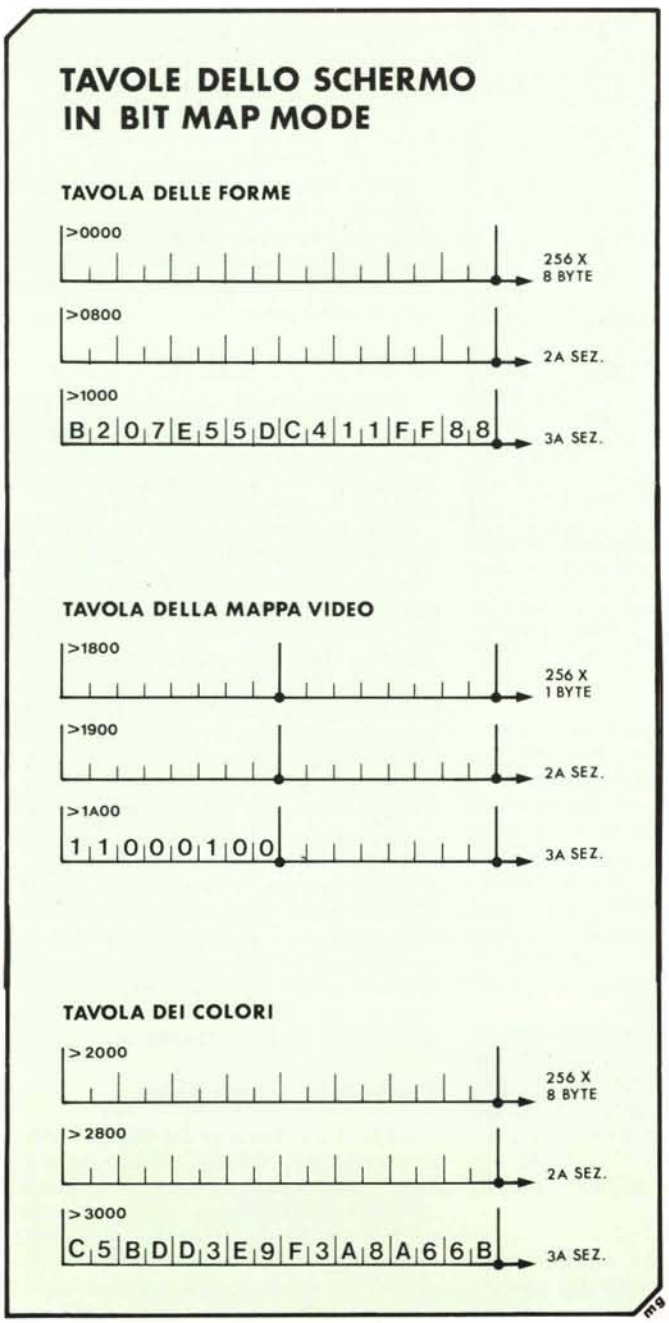
Anche la Tavola delle Forme è divisa in tre sezioni contenenti 256 informazioni ciascuna. Ogni informazione è però in tal caso costituita da 8 byte ed è proprio grazie a questa maggiore occupazione della memoria video ($256 \times 8 \times 3 = 6.144$ byte pari a 6K che è possibile indirizzare il singolo punto sullo schermo. La Tavola delle Forme può essere collocata a >0000 oppure a >2000, a seconda di dove si voglia avere quella dei colori che occupa la stessa quantità di memoria. Per ottenere tale variante occorre agire sul VDP Write Only Register 4. La definizione dei singoli caratteri avviene tramite la solita stringa di 16 codici esadecimali.

Tavola dei Colori

È divisa come le precedenti in tre sezioni comprendenti 256 informazioni ciascuna. Ogni informazione occupa 8 byte per un totale di 6K. A questo punto possiamo quindi affermare che per definire una porzione dello schermo pari ad una matrice di 8×8 punti occorrono: 1 byte per il nome del carattere di riferimento, 8 byte per la sua forma e 8 byte per i suoi colori, ossia 17 byte che moltiplicati per i 768 caratteri che costituiscono il video fanno la bellezza di 13.056 byte. Questo è un altro motivo per cui non si può fare grafica spinta con i 16K (che in realtà non sono precisamente tali) della consolle base, dal momento che il programma relativo e le variabili dovranno pur trovare posto da qualche parte. 6K di memoria dedicati solo alla Tavola dei Colori sono veramente tanti, ma bisogna tener presente che il TMS 9918A è uno dei migliori microprocessori grafici a colori disponibili sul mercato ed è inevitabile che per gestire le sue capacità richieda più memoria degli altri.

Vediamo come avviene la codificazione delle tinte. Ad ogni carattere corrispondono 8 byte suddivisi in 16 semi-byte. I primi 4 bit del primo byte indicano il colore da assegnare ai punti da accendere nell'ambito degli 8 disponibili per la prima riga, i secondi 4 bit il colore dei punti spenti e così via per le altre 7 righe e gli altri 7 byte. Viene naturale pensare che un Pixel spento non abbia nessun colore o che perlomeno sia contrassegnato da quello nero, in tal caso forse sarà più opportuno dire che i punti definiti come ON avranno il colore indicato nei primi 4 bit a sinistra e gli altri quello indicato nei 4 bit rimanenti. I codici delle tinte sono 16 (gli stessi del TI-Basic) e sono individuati tramite i caratteri esadecimali da 0 a F, esprimibili appunto in 4 bit. Come già accennato prima è quindi possibile definire il colore di uno qualsiasi dei Pixel dello schermo, purché non si indichino più di due tinte nell'ambito di un intervallo orizzontale di 8 punti. In verticale il problema non esiste, volendo si può disegnare una linea composta da 16 Pixel ognuno di colore diverso!

La Tavola dei Colori può essere allocata agli indirizzi >0000 o >2000 delle VDP Ram, a seconda di quanto specificato nel VDP Write Only Register 3. MC



Nella gestione del video in Bit Map Mode tramite l'Assembler, le VDP Ram vengono quasi totalmente utilizzate per la memorizzazione delle tre tavole necessarie per la procedura di indirizzamento del singolo Pixel e l'indicazione del suo colore. La tavola della Mappa Video è composta da 768 informazioni (una per ogni possibile posizione dei caratteri sullo schermo) che fanno da puntatori a quelle memorizzate nelle tavole delle Forme e dei Colori. Praticamente il microprocessore grafico TMS 9918A legge nella tavola della Mappa Video il nome del carattere da visualizzare in un preciso punto dello schermo, ne cerca il pattern, ossia la forma, ed i colori nelle tavole relative e quindi lo mostra sullo schermo. Il tutto porta via poco meno di 13K di memoria Ram. L'Assembler, dal momento che per funzionare richiede l'espansione da 32K, può permettersi di sacrificare quasi tutte le VDP Ram per la gestione del video, il TI Basic ovviamente no.