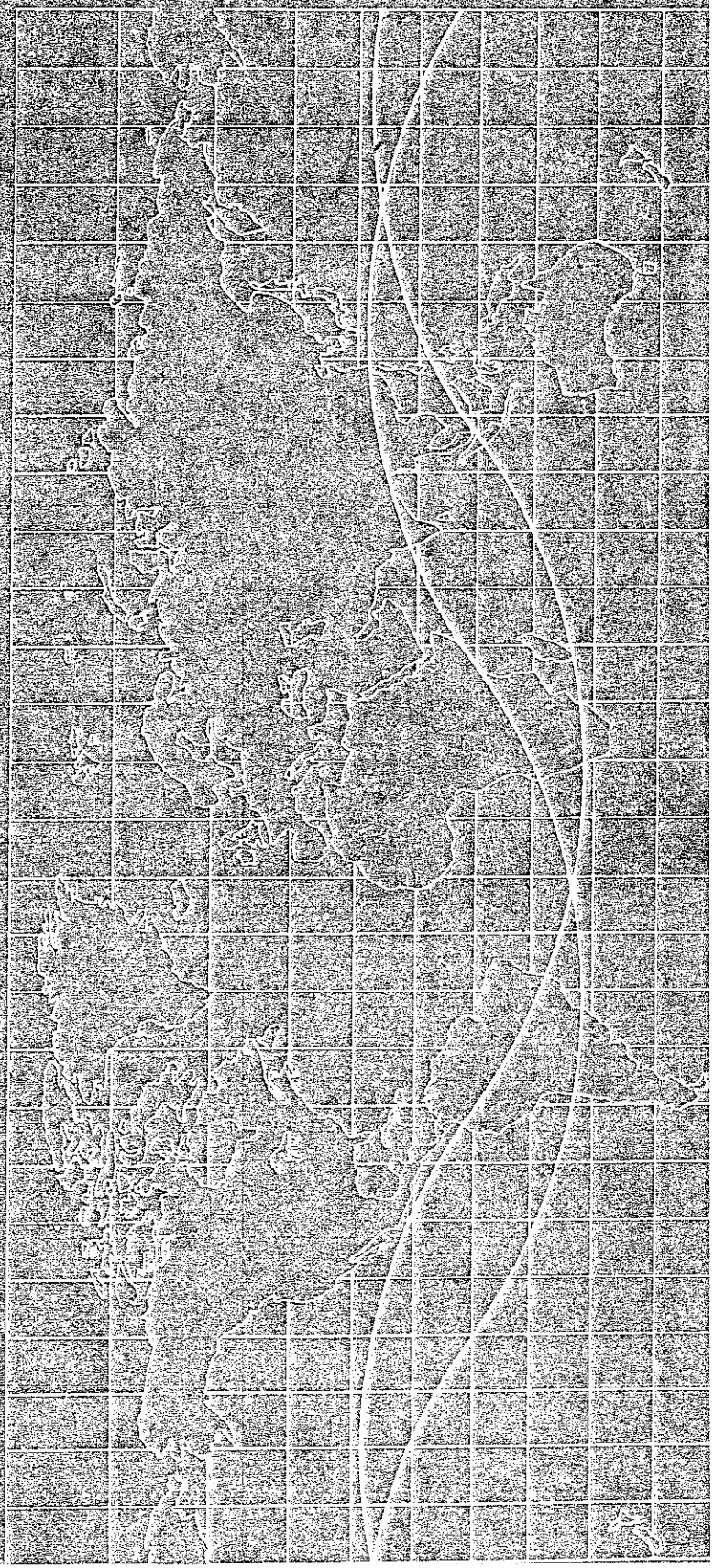
SHARP

PERSONAL COMPUTER

MZ-80A

MANUALE DELL'UTENTE



Personal Computer

MZ-80A

Manuale dell'Utente

080314-250182

Notizia

Quest'apparecchio é stato prodotto in confornitá alle direttive CEE 76/889/CEE.

Il presente manuale è per l'SA-5510 BASIC Interpreter usato con il Personal Computer SHARP MZ-80A. Il Personal Computer per uso generale MZ-80A è sostenuto da un sistema di software contenuto in software pack (nastri magnetici o diskette).

Tutto il sistema di software è soggetto o revisione senza previa notizia, per cui è particolarmente importante fare attenzione ai numeri di versione del contenuto. Il presente manuale è stato preparato con cura e poi controllato affinché risulti essere chiaro e preciso. Nel caso, però, che si riscontrino degli errori oppure che ci siano delle ambiguità, si prega di prendere contatto con il locale rappresentante della Sharp per tutti i chiarimenti del caso.

Prefazione

Il presente manuale descrive il personal computer MZ-80A della Sharp. Si legga completamente questo manuale al fine di divenire famigliare con le operazioni da eseguire, linguaggio BASIC SA-5510 e con le precauzioni da osservare ancora prima di prendere contatto e di usare l'MZ-80A. Questo manuale è uno della serie di manuali che descrivono l'MZ-80A ed il relativo software.

Il capitolo 1 descrive le caratteristiche principali dell'MZ-80A, le caratteristiche linguistiche del sistema di software standard del BASIC Interpreter SA-5510 e le procedure generali di funzionamento; si legga, per prima cosa, questo capitolo. Il capitolo 2 descrive i comandi e le sotto routine del sistema standard di software MONITOR SA-1510. Il capitolo 3 descrive l'hardware. Le informazioni contenute in questo capitolo risulteranno essere della massima utilitá nel caso si vogli ampliare il sistema.

Tutto il software viene fornito sotto forma di file. Insieme all'MZ-80A viene fornita una cassetta che contiene l'interpreter BASIC SA-5510.

Precauzioni

L'MZ-80A è uno dei migliori personal computer del mondo; lo sua progettaz ione comprende tutta la conoscenza tecnica che la Sharp ha potuto accumulare durante i numerosissimi anni di esperienza nel campo dell'elettronica. Tutte le unità sono sempre scrupolosamente controllate prima della spedizione in modo che ognuna d'esse possa, funzionare normalmente una volta tolta dall'imballaggio. Si controlli, in ogni caso, per vederese si sono avuti danni durante il trasporto. Se si riscontra qualche danno, anche il più piccolo, si contatti immediatamente il proprio fornitore.

Al fine di mantenere l'unita nel sua stato migliore durante il funzionamento, si osservino le seguenti norme.

- Non si metta l'MZ-80A in luoghi dove la temperatura è estremamente alta oppure estremamente bassa ovvero dove si ha un'ampia escursione termica. S'eviti l'esposizione diretta alla luce solare, alle vibrazioni ed alla polvere.
- Si maneggi sempre con estrema precauzione il cavo d'alimentazione al fine di prevenire la possibilktà di danneggiamento. Quando lo si estrae dalla presa, per prima cosa si spenga l'unità, e poi si tiri la spina, mai, pen nessuna ragione, il cordone.
- Se l'interruttore viene posto sulla posizione di spento e poi immediatamente dopo rimesso sulla posizione di acceso, può accadere che l'inizializzazione non venga eseguita correttamente. Dopo aver spento l'apparecchio attendere sempre alcuni secondi prima di riaccendere l'apparecchio.

Altri punti sono delineati alla fine del presente manuale.

Contenuto

Il Vostro MZ-80A e la Programmazione BASIC

Capitolo 1

1.1 Profilo dell'MZ-80A

È necessario conoscere la configurazione del computer per poi praticamente elaborare i programmi che debbono essere fatti girare su di esso. È più si conosce sulla console, sulla memoria, sulle periferiche e sulla lingua di processo dei programmi e più efficentemente è possibile elaborare i programmi in quanto è possibile creare dei programmi che sfruttano completamente le possibilità della macchina. È possibile arricchire la propria conoscenza soltanto accumulando esperienza nel concepimento e nel facendo girare programmi nel computer.

Questa prima sezione presenta un profilo del microcomputer SHARP MZ-80A per permettere all'utente di afferrare nelle sue grandi linee la configurazione dell'hardware e delle procedure BASIC. Nella sezione seguente saranno descritti i primi passi nella programmazione del computer.

Profilo

L'MZ-80A è un micro computer integrato che è stato immesso sul mercato nell'autunno del 1981. Si tratta di un piccolo computer multi uso completamente nuovo progettato avendo in mente un ampio sviluppo sia dell'hardware che del software. Le sue maggiori caratteristiche sono l'altra velocità e la facilità operativa. Al momento della sua presentazione, l'MZ-80A, è stato favorevolmente criticato ed è stato detto che è un computer che aprirà una nuova dimensione nella programmazione dei computer.

Nella figura 1.1 si può vedere una configurazione semplificata dell'hardware dell'MZ-80A. Consiste di una unitù di immagazzinamento (che immagazzina programmi e dati), di una unità centrale del calcolatore (che esegue ed elabora le operazioni sulla base dei dati come da programma che si trova nell'unità di immagazzimamento ed esegue il trasferimento) e di diverse unità di input ed output. L'unità d'immagazzimamento viene a sua volta suddivisa in memoria principale, programma monitor e sezione RAM. La sezione della memoria principale dell'MZ-80A possiede 32 K bytes di RAM (memoria di scrittura e lettura). La sezione della memoria principale può essere espansa fino a 48 K bytes aggiungendo 16 K bytes di RAM. Le unità di input sono la tastiera e la cassetta. Le unità di output sono il CRT (unità video), la cassetta e l'uscita audio.

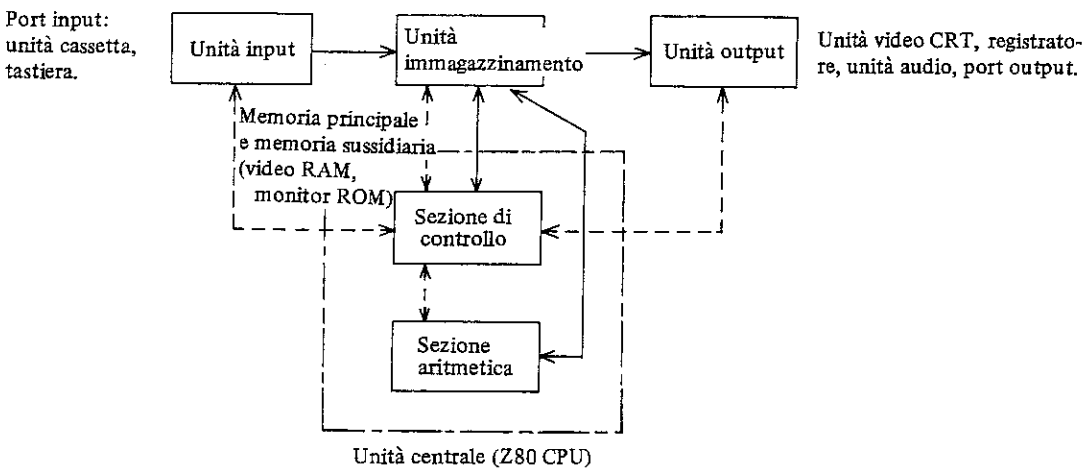


Figura 1.1 MZ-80A configurazione.

L'unità centrale, costituita dalle sezioni di controllo e aritmetica, agisce attivamente e dinamicamente; e il cervello del computer e controlla tutte le operazioni. Il suo funzionamento, però, è costituito dalla ripetizione di questa semplice sequenza operativa:

1. Un dato contenente un'istruzione è pronto per l'immagazzinamento.
2. L'istruzione è eseguita.

In altre parole, logicamente parlando, essa è una collezione di dati nell'unità di immagazzinamento che da delle istruzioni che fanno sì che il computer esegua il suo lavoro.

All'ingerno del computer i dati ed i segnali di controllo sono rappresentati logicamente dalle cifre 0 ed 1. Il numero di cifre di una figura binaria (cioè la sequenza di 0 e di 1) viene contata in termini di bit. Per esempio, un numero binario di 8 bit:

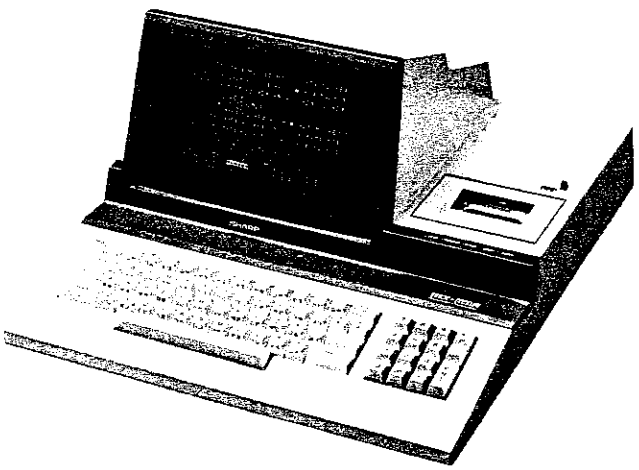
0 0 1 1 0 1 0 1

è un dato che ha la lunghezza (equivalente, nella rappresentazione decimale, a 53). Siccome i bit sono troppo piccoli per essere convenienti al fine dell'indicazione della lunghezza di un dato, c'è una unità chiamata "byte" che viene usata per indicare un dato costituito da 8 bit. Un byte può rappresentare fino ad un massimo di 2^8 (=256) numeri differenti.

L'MZ-80A utilizza un Z80, un cosidetto microprocessore da 8 bit (che elabora un byte di dato per volta), come unità centrale. Per cui i programmi che danno le istruzioni ed i dati che debbono essere elaborati sono tutti immagazzinati e trasferiti in unità di byte. Le locazioni di byte nell'unità di immagazzinamento sono designate da un pointer da 2 byte nell'unità centrale. Con questo pointer da 2 byte, lo Z80 può indirizzare fino a 2^{16} (=65536) locazioni. Siccome 2^{10} (=1024) rappresenta 1 K byte, significa che lo Z80 possiede spazio sufficiente per 64 K byte di address. Come detto in precedenza, l'MZ-80A ha una capacità di immagazzinamento, nell'unità centrale, che arriva a 48 K byte, ovvero 3/4 dello spazio d'indirizzo della memoria Z80 RAM (Random Access Memory). La RAM è un tipo di memoria che piò essere libecamente letta e scritta, dove, invece, la ROM (Read Only Memory) può essere solo letta.

La maggior parte dei computer ad uso specialistico dedicati al controllo automatico e molti personal computer posseggono memorie dove 1/3, in alcuni casi 1/2 o più della memoria è costituito da ROM per l'immagazzinamento dei programmi di controllo (come per esempio i programmi di BASIC interpreter). L'utilizzazione della RAM nella configurazione di memoria dell'MZ-80A è basata sulla premessa che la memoria principale deve essere a libera disposizione per una varietà di usi. L'MZ-80A immagazzina tutti i programmi di sistema in file esterni dai quali possono essere caricati nella memoria principale per mezzo del programma monitor.

L'SA-5510 BASIC interpreter, uno dei sistemi di programma dell'MZ-80A, funziona per tradurre i programmi BASIC in codice di macchina per l'esecuzione.



Il personal computer MZ-80A

1.2 Uso dell'MZ-80A

Questo paragrafo descrive le unità constituenti dell'MZ-80A e relative funzioni.

■ Fronte dell'MZ-80A

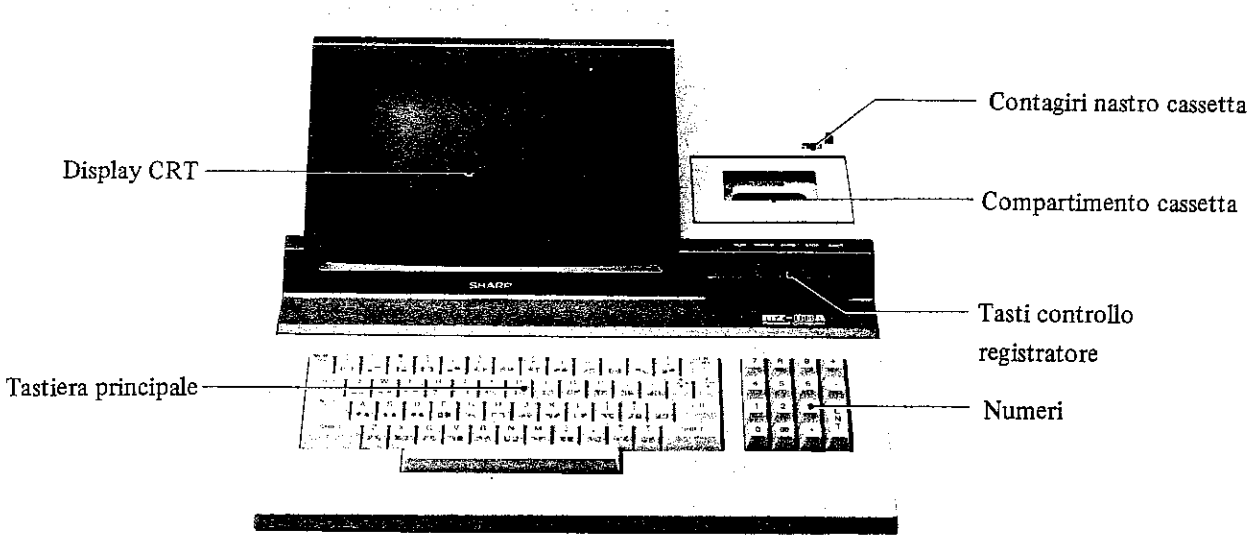


Figura 1.2

■ Retro dell'MZ-80A

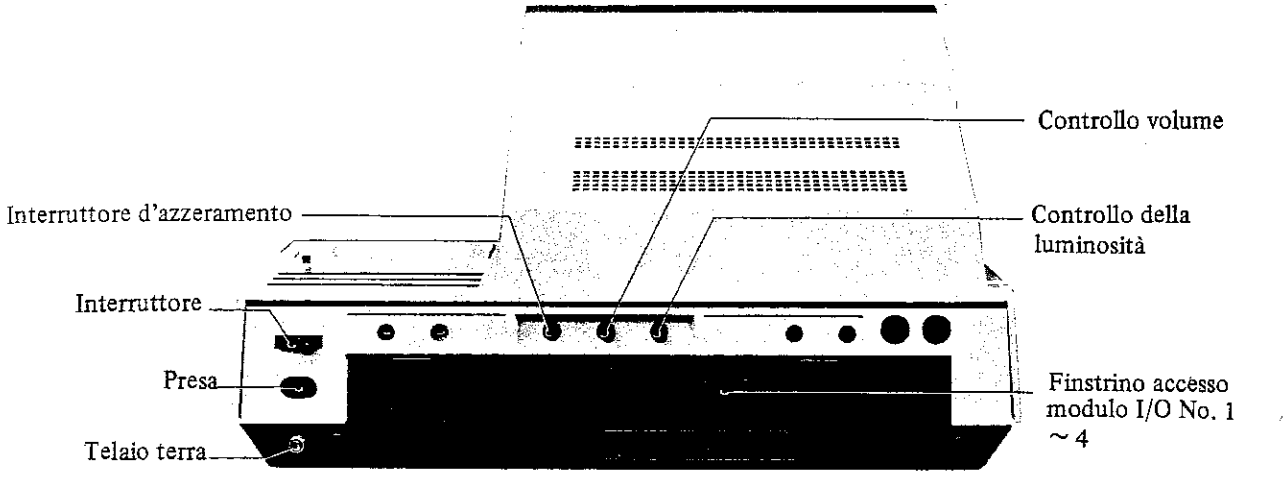


Figura 1.3

1.2.1 Attivazione del sistema di software

Il personal computer MZ-80A possiede come supporto un sistema di software che si trova in file di software.

Il BASIC SA-5510 si trova su file di cassette e deve essere sottoposto al loading (caricamento) di programma iniziale ogni qualvolta deve essere utilizzato. Il loading è estremamente facile.

Per prima cosa si accenda l'apparecchio chiudendo l'interruttore che si trova nella parte poteriore dell'MZ-80A. Il programma monitor incomincia e sul CRT appare il messaggio seguente:

```
* *  MONITOR SA-1510  * *
*
```

il cursore lampeggia

Mettere la cassetta con il file BASIC nella piastra e premere il tasto [L], dopodiché si prema il tasto [CR]. (L: load)

Il caricatore del programma di Monitor incomincia a girare e appare il messaggio " ⬇ PLAY". Premere il pulsante [PLAY] della piastra.

Il loader di programma carica il BASIC interpreter (foto a sinistra della figura 1.4), ed una volta completato il caricamento l'MZ-80A mostra il messaggio illustrato nella foto di destra della figura e l'interpreter BASIC inizia ad operare.

Il messaggio "Ready" indica che il controllo del sistema si trova a livello di comendo BASIC ed il sistema è pronto per accettare qualsiasi comando.

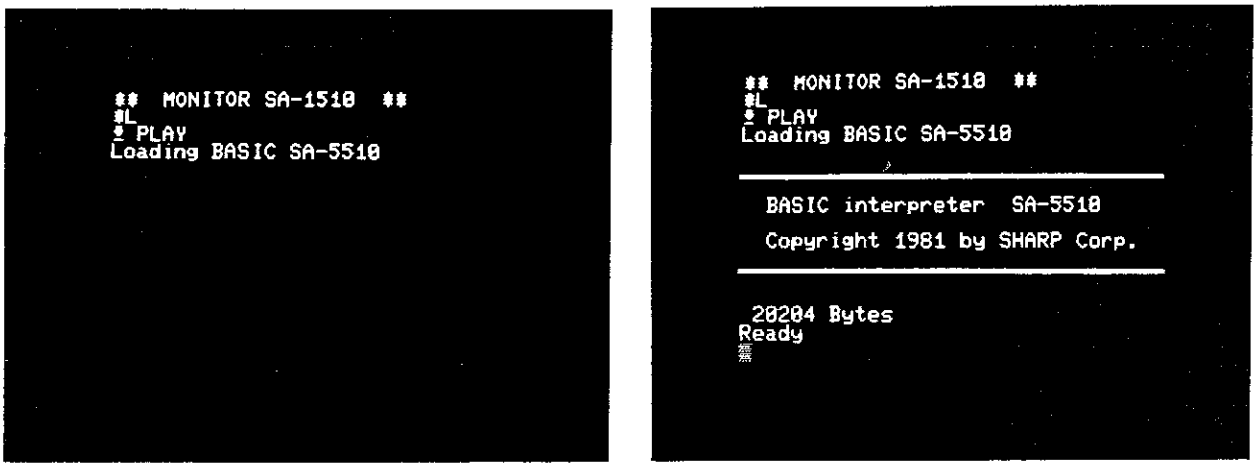


Figura 1.4

Si prega di vedere il capitolo 2 per l'attivazione del sistema di software da file di diskette e comandi monitor.

1.2.2 Tastiera

La tastiera dell'MZ-80A è disposta come mostrato nella figura 1.5 ed è suddivisa in due zone, la tastiera principale alfanumerica ed il tastierino numerico.

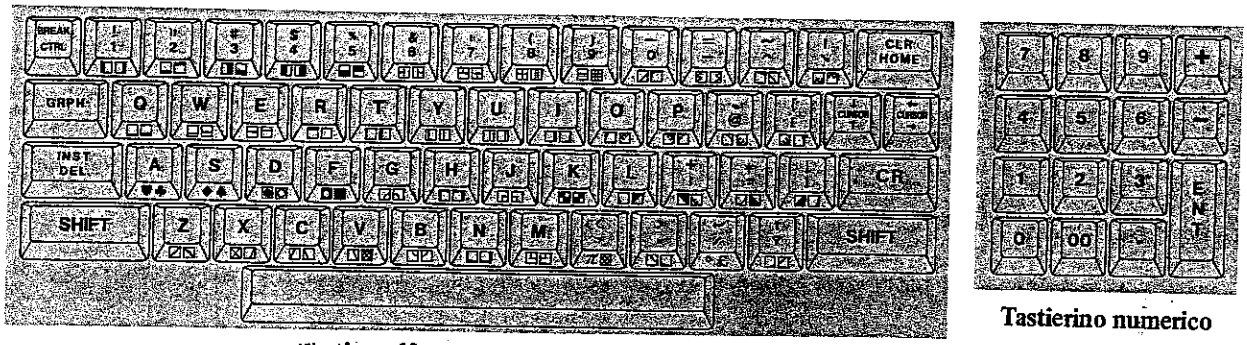


Figura 1.5 La tastiera dell'MZ-80A.

La tastiera principale alfanumerica (tipo macchina da scrivere) conforme alle norme ASCII standard, comprende i tasti di carattere e i tasti di controllo (come il tasto di ritorno del carrello, il tasto di controllo ed i tasti di controllo del cursore).

Il tastierino numerico a preposto alla registrazione dei dati numerici è simile a quello dei normali calcolatori.

La tastiera principale funzionea in due modi:

[1] Modo normale
[2] modo grafico

I tasti che si trovano nella tastiera principale alfanumerica producono differenti caratteri a seconda del modo di operazionne, come mosrato nella figura 1.6.

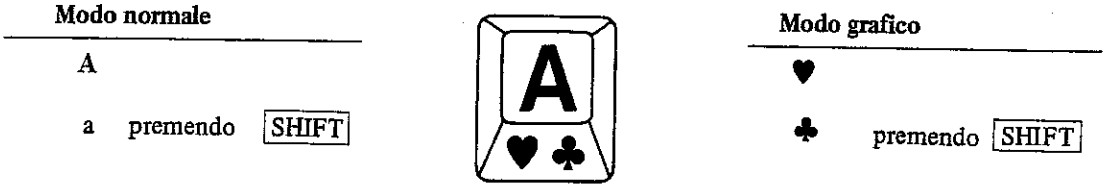


Figura 1.6 Differenti caratteri del tasto A

Si noti che le lettere normalmente prodotte sono maiuscole. Per registrare le lettere minuscole, premere il tasto SHIFT e poi premere i tasti alfabetici – proprio l'opposto di quanto avviene nelle comuni macchine da scrivere. Il motivo di ciò è che le lettere maiuscole sono, generalemente, più facili da leggere sullo schermo, per cui la maggioranza delle persone preferisce scrivere i propri programmi in lettere maiuscole.

La figura 1.7 mostra i tasti di controllo (i tasti in neretto).

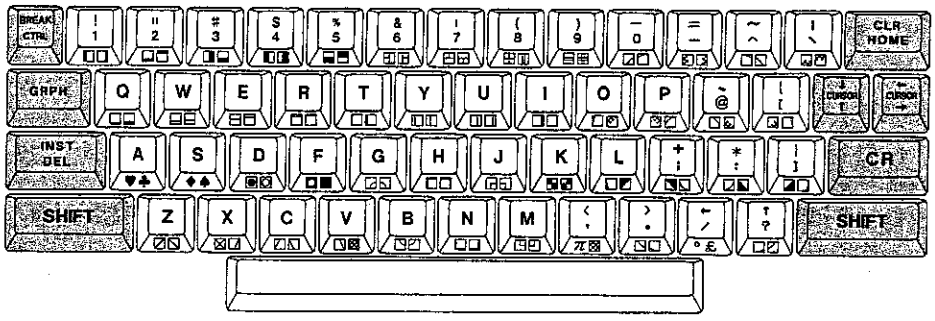


Figura 1.7 Tasti di controllo.

La funzioni dei tasti di controllo sono spiegate di sequito:

SHIFT : simile al tasto delle maiuscole che si trova nelle comuni macchine da scriv questo tasto si possono usare i caratteri che si trovano di sopra.

CR : tasto per il ritorno del carrello. Il tasto [ENT] ha la stessa funzione del ta

GRPH : se si spreme questo tasto nel modo normale, viene immesso il modo grafi cambia da " ▩ " a " ▪ " e viceversa.

INST DEL : DEL cancalla il carattere che si trova alla sinistra della posizione del curs caratteri della stringa di uno spazio a sinistra. INST inserisce uno spa spostando verso destra tutti i caratteri della stringa.

CLR HOME : HOME fa ritornare il cursore nella posizione in alto a sinistra dello scherm Nel modo grafico, HOME produce il carattere inverso " ▪ " e CLR p " ▩ ".

CURSOR **CURSOR** : Tasti di controllo del cursore. Ogni tasto sposta il cursore nella direzior nella posizione normale che in quella di maiuscolo).
Nel modo grafico ogni tasto produce la freccia come carattere inverso; ▪

BREAK CTRL : quando questo tasto viene premuto con il tasto [SHIFT] premuto, vie break (interruzione) e l'interruzione dei programmi BASIC.

La figura 1.8 mostra il tasto CTRL ed alcuni altri tasti (i tasti in neretto).

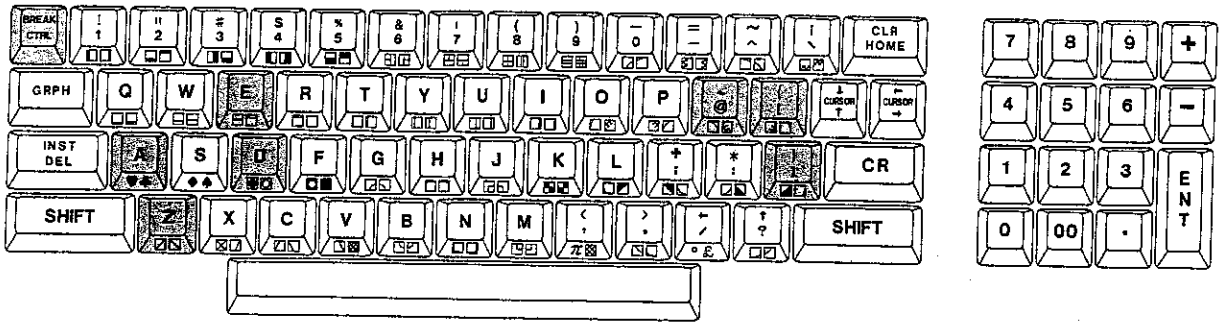


Figura 1.8 Il tasto CTRL ed alcuni altri tasti.

La funzioni di questi tasti quando si preme il tasto CTRL sono spiegate qui di seguito:

CTRL + A : questo blocca il tasto SHIFT per cui non c'è più la necessità di mantenerlo premuto. Premendo nuovamente questo tasto oppure premendo il tasto CR si libera il blocco.

CTRL + E : questo svolge il listing del display CRT.

CTRL + D : questo avvolge il listing del display CRT.

CTRL + Z : questo genera il carattere "→". Questo carattere viene usato come delimitatore (PASCAL, FDOS, ecc).

CTRL + @ : questo pone il modo di rappresentazione dei caratteri nel modo inverso. Premendo nuovamente questi tasti si ha di nuovo il modo di carattere normale.

CTRL + [: questo mette a punto la configurazione V-RAM nel modo MZ-80K.

CTRL +] : questo mette a punto la configurazione V-RAM nel modo MZ-80A.

1.3 Operazioni BASIC per la programmazione

Iniziamo a questo punto lo studio della programmazione. In questa sede ci si prefigge lo scopo di permettere al principiante di acquistare famigliarità con gli elementi dei programmi BASIC. Nella prima sezione si costruiranno programmi molto corti per illustrare i concetti fondamentali e per imparare le operazioni del BASIC che sono richieste lungo il corso di un programma BASIC. Si apprenderà:

1. Come si costruisce un programma
2. Come si fa girare un programma
3. Come si corregge un programma
4. Come si conserva un programma (su nastro magnetico)
5. Come si fa girare un programma immagazzinato su un file esterno.

1 La costruzione di un programma

Affinché il computer faccia il suo lavoro è necessario impartirgli una serie di istruzioni secondo le quali eseguirà un certo iter lavorativo. La dterminazione della sequenza delle istruzioni, impiantandole come programma BASIC, la registrazionne dei programmi nell'MZ-80A dalla tastiera e la correzione del programma sono operazioni fondamentali per lo sviluppo del programma. Il problema che viene dato di seguito è un semplice esempio di lavoro che viene eseguito con un computer.

Esempio 0: Leggere due dati numerici dalla tastiera, computare la loro somma e mostrare il risultato.

La sequenza delle istruzione è, come indicata dal problema, "leggere due dati numerici dalla tastiera", "computare la loro somma" e "mostrare il risultato". Queste istruzioni sono scritte in BASIC nel modo seguente:

```
10  INPUT A
20  INPUT B          } Leggere due dati numerici dalla tastiera
30  LET C = A + B  ..... Computare la loro somma
40  PRINT C ..... Mostrare il risultato
50  END ..... Fine
```

Nelle prime due righe alle variabili A e B sono assegnati due valori numerici per mezzo dello statement INPUT, che ha la funzione di ricevere i dati provenienti dalla tastiera. Nella riga seguente la somma di A e B viene assegnata alla variabile C. Il contenuto di C viene mostrato sullo schermo per mezzo dello statement PRINT che si trova nella riga seguente, e che ha la funzione di mostrare i dati sul CRT. In seguito il programma finisce. Si sono spiegati tutti questi passi anche se essi possono sembrare ovvi. È proprio da questo punto che deve iniziare uno studio serio.

Del problema su citato, due punti sono da tenersi ben in mente:

- Un programma BASIC viene scritto usando parole come INPUT, LET, PRINT, END, ecc. Le righe contenenti queste parole sono chiamate statement INPUT, statement PRINT, e così via.
- Ogni riga comincia con un numero come 10.

In altre parole si può dire che un programma BASIC è costituito di uno statement che comincia con una serie di parole (dette parole riservate) oppure la loro abbreviazione e numeri (detti numeri di riga) che precedono lo statement. Anche se il programma che precede ha solo cinque righe, esso è un programma completo. Infatti una singola riga può costituire un programma se essa contiene un numero di riga ed uno statement. I programmi più grandi posseggono gli stessi elementi di un programma di una singola riga.

Il passo seguente è quello di registrare il programma nel computer dalla tastiera. Ciò non è eccessivamente difficile da fare: basta registrarlo come si farebbe con una macchina da scrivere. È necessario, però, tenere presente i punti seguenti:

- Tutti i nomi variabili e le parole come INPUT e PRINT debbono essere registrati in lettere maiuscole. La tastiera alfanumerica dell'MZ-80A stampa le lettere maiuscole nel modo normale e le lettere minuscole sono stampate premendo il tasto [SHIFT] (proprio come al contrario di una normale macchina da scrivere).
- Ogni riga deve terminare con la pressione sul tasto [CR] (oppure [ENT] nel caso del tastierino numerico). Una riga di dati non viene registrata nella memoria come riga di programma fino a che nono si preme il tasto [CR].

Ecco i tasti premuti per la prima riga.

1 0 I N P U T A [CR]

Il cursore sullo schermo si sposta all'inizio della riga seguente quando si preme il tasto [CR]. Registrare la seconda e terza riga in successione. L'intero programma viene ad essere registrato nella memoria quando lo statement END alla riga numero 50 viene registrato, seguito dalla pressione sul tasto [CR].

Si prema ora:

L I S T [CR]

Sullo schermo appare la lista del programma registrato. LIST è un comando che mostra la lista delle righe di programma che sono state immagazzianate nella memoria. Esso viene detto combando per distinguere la sua funzione dallo statement (come per esempio INPUT) che viene usato all'interno del programma.

2 Esecuzione di un programma

Per far girare un programma, immettere il comando RUN nel computer. Premere:

R U N [CR]

Il segno "?" appare nella riga seguente ed il cursore prende a lampeggiare. Ciò significa che l'esecuzione del programma è iniziata e che il primo statement INPUT è in esecuzione. Premere ora, per esempio, il numero 19 come valore della variabile A. Anche la registrazione dei dati durante l'esecuzione dello statement INPUT deve terminare con la pressione sul tasto [CR].

1 9 [CR]

È conveniente usare il tasto [ENT] al posto del tasto [CR], quando si registra un valore numerico dal tastierino. Il secondo statement INPUT viene, di seguito, eseguito ed un "?" appare nuovamente sullo schermo. Registrare "81" come valore della variabile B.

8 1 [CR]

Il computer, al ricevere il valore della variabile B, esegue il computo e l'operazione di assegnazione come scritto nella riga numero 30, poi mostra il risultato

1 0 0

sullo schermo come risultato dello statement PRINT della riga numero 40. Ciò viene ottenuto come risultato dell'addizionne di 19+81. Il computer finisce l'esecuzione del programma quando incontra lo statement END alla riga numero 50, e mostra

Ready

sullo schermo ed il cursore riprende a lampeggiare. Il messagio "Ready" indica che il computer è in un modo, detto modo di comando, nel quale non viene eseguito programma alcuno ed i comamdi sono in posizione di attesa. Nel modo di comando è possibile registrare i comandi LIST oppure RUN oppure modificare il programma.

3 Correzione del programma

La procedura per correggere oppure modificare un programma è fondamentalmente la stessa procedura usata per la creazione del programma stesso.

Per esempion, per correggere il programma prima riportato in mode che il risultato di A – B sia assegnato alla variabile C ed il contenuto di C sia mostrato sullo schermo, è necessario registrare:

3 0 L E T C = A – B [CR]

Quando una riga con lo stesso numero di una riga già esistente viene registrata, essa viene a sostituirsi alla riga precedente.

Un metodo più conveniente detto editing di schermo, può essere usato quando si deve cambiare solo una porzione della riga, come in questo esempio dove solo il segno del più deve essere sostituito dal segno del meno. Con l'editing dello schermo tutto ciò che si deve fare è di muovere il cursore sullo schermo e di metterlo nella posizione dove si desidera fare il cambiamento utilizzando i tasti di controllo del cursore e poi scrivere sopra il carattere(i). Per terminare l'editing, premere il tasto [CR] (il tasto [CR] può essere premuto con il cursore in una posizione qualsiasi). Per inserire oppure cancellare dei caratteri, si usi il tasto INSERT/DEL. Dopo averlo modificato si faccia girare il programma.

4 Registrazione di un programma

I programmi che vengono registrati nella memoria principale del computer vengono cancellati quando si spegne l'apparecchio. È necessario inparare, dunque, come registrare i programmi in file esterni al fine di eseguirli oppure di completarli successivamente, oppure scambiarli con altri amici che posseggono l'MZ-80A.

L'unità a cassette dell'MZ-80A è uno istrumento di input/output che non viene usato solo per l'iniziazione dell'Interpreter BASIC, ma anche per la registrazione e la lettura di programmi e dati. Per registrare un programma nella cassetta, si segua la procedura qui riportata:

- Mettere una cassetta nell'unità. Quando si registra all'inizio del nastro, riavvolgerlo premendo il tasto REW prima di procedere al passo successivo.
- Registrare il comando SAVE insieme al nome del programma.

In comando SAVE fa si che il programma nel computer sia registrato sulla cassetta.

Registrare, ora, il programma in argomento (cambiato in sottrazione per mezzo dell'editing di schermo) sul nastro della cassetta. Supponendo che il nome del programma sia "Subtraction", dopo aver posto la cassetta nell'MZ-80A, si esegua:

S A V E "S u b t r a c t i o n" [CR]

Poi,

↓RECORD. PLAY

appare sullo schermo. Premere i tasti RECORD e PLAY contemporaneamente e

Writing "Subtraction"

appare sullo schermo, per indicare che l'operazione di registrazione è in corso.

Quando il programma è stato registrato appare la scritta "Ready" sullo schermo.

5 Lettura di un programma dalla cassetta

Il comando LOAD viene usato per la lettura di programmi scritti su nastro. Per leggere il programma "Subtraction" premere:

L O A D "S u b t r a c t i o n" [CR]

La figura 1.9 mostra, dopo l'inizio dell'Interpreter BASIC, come è stato letto il programma Subtraction nel computer dalla cassetta per mezo del comando LOAD.

Mostra anche il messaggio "Found" e "Loading" che sono mostrati nel corso della lettura del programma per indicare che il file richiesto è stato trovato ed è in corso di lettura.

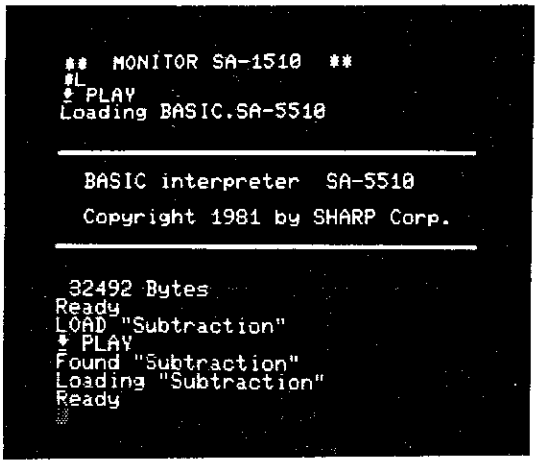


Figura 1.9 Caricamento di un programma

Che cos'è il modo diretto?

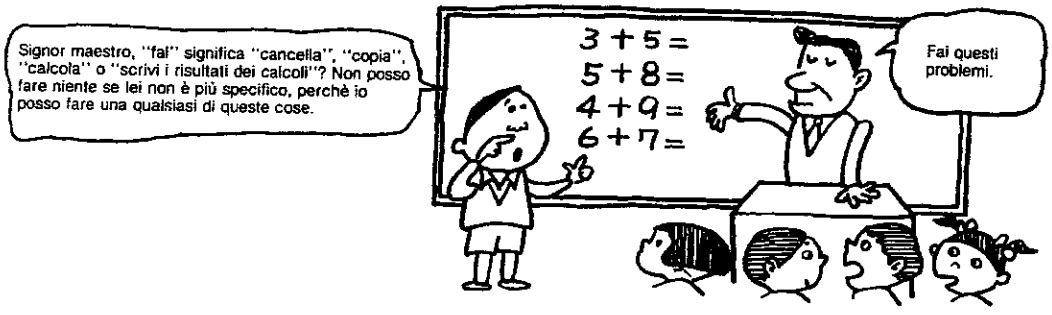
Se ve ne è bisogno, l'elaboratore può anche venire usato come una calcolatrice. Questo tipo di funzionamento viene chiamato "Modo Diretto".

Come fareste su di una calcolatrice, provate ad immettere 5 + 8 =.

Per immettere il [+], premere il tasto corrispondente tenendo abbassato il tasto [SHIFT].

In effetti però, l'elaboratore visualizza i caratteri sullo schermo come sono stati immessi e nessun calcolo viene eseguito anche se viene premuto il tasto [CR]. Questa è la vera differenza tra il vostro elaboratore e la calcolatrice.

L'elaboratore richiede che gli venga fornita una istruzione su che cosa fare di 5 + 8 =.



PRINT

Perchè si possa usare l'elaboratore come una calcolatrice, si deve poter visualizzare sullo schermo il calcolo di 5 + 8. Per ottenere questo risultato, si può usare il comando PRINT come istruzione. Per trasferire le istruzioni, usiamo questo comando nel modo seguente:

[P][R][I][N][T][5][+][8][CR]

Mano a mano che i tasti vengono premuti, i caratteri corrispondenti vengono visualizzati sullo schermo. E l'elaboratore cosa fa?

Ready ..... Significa "Comincia a lavorare".
PRINT 5 + 8 ..... Visualizza il risultato di 5 + 8 e quando si preme il tasto [CR] si comunica la fine del comando.
13 ..... Questo è il risultato del comando eseguito.
Ready ..... Che cosa devo fare adesso?
▩ ..... Cursore

Le quattro operazioni aritmetiche

Se desiderate proseguire con le moltiplicazioni e le divisioni, fate attenzione che l'elaboratore usa dei simboli leggermente diversi da quelli della matematica comune.

Simbolo di moltiplicazione	*
Simbolo di divisione	/

Calcoli con l'uso delle Parentesi

L'elaboratore è in grado di gestire calcoli più complessi di quanto non sia capace una semplice calcolatrice. Questi sono i calcoli con l'uso delle parentesi.

Nel caso delle normali operazioni matematiche, vengono usati simboli diversi per raggruppare, come ad esempio:

$3 \times 6\ [6 + 3\ \{\ 9 - 2\ (4 - 2) + 1\ \}\ ]$

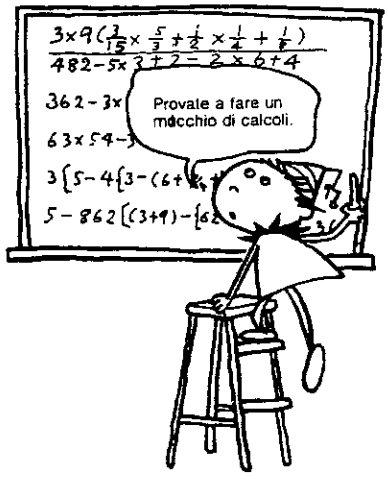
Nel caso dell'elaboratore viene invece sempre usato il simbolo ( ).

$3 * 6 * (6 + 3 * (9 - 2 * (4 - 2) + 1))$

Anche se usa solo quei simboli, l'elaboratore non dimentica mai la regola che i calcoli entro i segni di raggruppamento più interni devono venire eseguiti per primi e non commette mai degli errori.

Esercizio

```
PRINT (6 + 4) / (6 – 4)
  5
PRINT 3 * (5 + 9 * (9 – 2) – 6 / (4 – 2) ) + 5
  200
PRINT (3 + 4) * (5 + 6)
  77
PRINT (10 + 20) / 6 * (2 + 3)
  25
PRINT (10 + 20) / (6 * (2 + 3) )
  1
```



Stringa? Espressione?

```
PRINT 3 + 5
```

Se, dopo aver scritto la riga precedente, si preme CR, si ottiene 8, non è vero? Adesso provate a racchiudere l'espressione tra virgolette:

```
PRINT "3 + 5" e CR
3 + 5
```

O bella! Il risultato è differente. Proviamo ancora una volta.

```
PRINT "BUON GIORNO, CARO AMICO" CR
BUON GIORNO, CARO AMICO
```

Come appare chiaro da quanto precede, i caratteri o i simboli racchiusi tra virgolette vengono visualizzati sullo schermo esattamente come sono.

Il blocco di caratteri e/o simboli racchiusi tra virgolette viene chiamato una stringa.

PRINT "3 + 5"

Questa, che è tra virgolette, è una stringa.

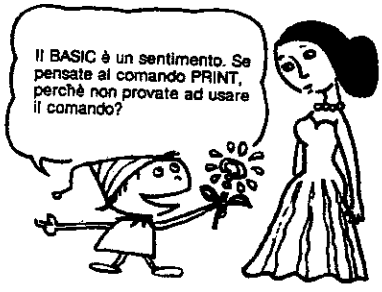
PRINT 3 + 5

Questa è un'espressione e non una stringa.

Dovete saperne di più sulle stringhe. La possibilità di usare tranquillamente le stringhe raddoppierà il divertimento dell'uso dell'elaboratore.

Questo è il comando con cui molto spesso avrete a che fare. Se pensate che sia noioso immettere ogni volta PRINT, premete al suo posto il tasto?

L'elaboratore converte automaticamente il? in un PRINT.




```
? 3 * 5
  15
? (3 + 4) * 10
  70
```

1° e 2° approccio alla PRINT

Al comando PRINT, si possono far seguire una grande quantità di voci, come stringhe ed espressioni. In tal caso, le singole voci devono venire separate da punti e virgole e da virgole.

```
PRINT "3 + 5 = " ; 3 + 5 CR
3 + 5 =  8
```

Punto e virgola.

L'espressione tra virgolette è una stringa.

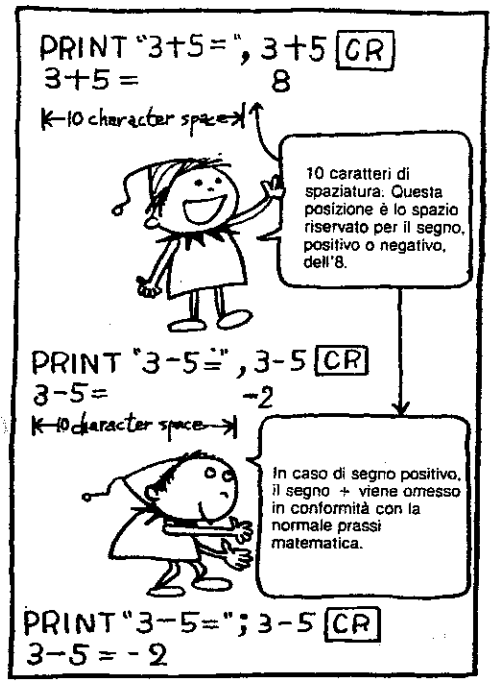
L'effettivo calcolo viene eseguito in conformità all'espressione che segue il punto e virgola. Provatelo.

Che cosa succede se invece del punto e virgola (;) si usa una virgola(,)?

```
PRINT "3 + 5 =" , 3 + 5 CR
3 + 5 =       8
```

Virgola.

Ma guarda! Il risultato di 3 + 5 viene visualizzato molto lontano dall'espressione. Questo avviene perché tra l'uso della virgola e del punto e virgola vi è la seguente differenza.



; ..... La visualizzazione viene eseguita vicino all'espressione

, ... La visualizzazione viene eseguita a dieci spazi di distanza dall'espressione.

Quando la separazione viene fatta mediante la virgola, invece che eseguire una spaziatura di 6 caratteri dalla posizione finale della stringa, viene eseguita una spaziatura di 10 posizioni dall'inizio della stringa. Questo è un fatto che dovete sempre tenere presente.

```
PRINT "12345", 3 + 5 CR
12345      8
```

Spaziatura di 10 caratteri.

```
PRINT "123456789", 3 + 5 CR
123456789  8
```

Spaziatura di 10 caratteri.

Se la stringa è più lunga di 10 caratteri, il numero 8 viene automaticamente spostato di altri 10 caratteri.

```
PRINT "123456789012", 3 + 5 CR
123456789012               8
```

Spaziatura di 20 caratteri.

Secondo approccio

Primo approccio.

Facciamo funzionare l'elaboratore.

Ecco un programma con istruzioni che occupano parecchie righe.

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
```

Questo programma istruisce l'elaboratore a fare A = 3, B = 5 e a visualizzare A, B e C sullo schermo, in cui C è il risultato della relazione che segue:

C = A + B

Il numero all'inizio di ogni istruzione è chiamato numero della riga. L'elaboratore assicura che le righe vengono eseguite in sequenza da quella con mumero minore a quella con numero maggiore. Di conseguenza, è possible inserire un'altra riga nel programma in un momento successivo, come ad esempio:

35 D = B – C

L'elaboratore esegue un programma in sequenza di numero di riga e quindi la numerazione delle righe avviene con un intervallo di 10, come indicato nell'esempio precedente, in modo che sia possibile aggiungere nuove righe quando ciò sia necessario. I numeri di sequenza possono essere liberamente scelti tra 1 e 65535.

Numero di riga Istruzione CR

La visualizzazione di un carattere sullo schermo non significa che esso è stato immesso. Dopo la visualizzazione di ogni riga, deve venire premuto il tasto CR per affidare la riga alla memoria dell'elaboratore.

Se per esempio viene immesso,

35 CR

l'istruzione contenuta nella riga 35 viene cancellata.

Vediamo ora di eseguire il programma.

Premere i tasti come segue RUN CR.

```
RUN
 3                    5                    8
```

←Spaziatura di 10 caratteri→ ←Spaziatura di 10 caratteri→

Una posizione per il segno positivo o negativo. Per i valori negativi viene inserito il segno meno (–).

List serve per capire in fretta

Mentre è in corso la conversazione con l'elaboratore, basandosi sui ripetuti tentativi, il primo programma che è stato immesso potrebbe scomparire dallo schermo. Anche in questo caso, non preoccupatevi. L'elaboratore non dimentica mai un programma che è stato immesso. Se volete rivedere il programma che avete immesso precedentemente, immettete:

LIST CR

Sullo schermo compariranno tutti i programmi che sono stati immessi. Se il programma si estende su decine o centinaia di righe che non possono venire visualizzate tutte in una sola volta, si può visualizzare una parte del programma.

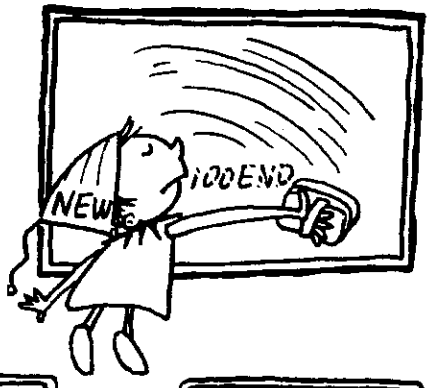
LIST – 30 CR	Visualizza un programma fino alla riga 30.
LIST 30 – CR	Visualizza un programma a partire dalla riga 30.
LIST 30 – 50 CR	Visualizza le righe da 30 a 50 di un programma.
LIST 30 CR	Visualizza la riga 30 di un programma.

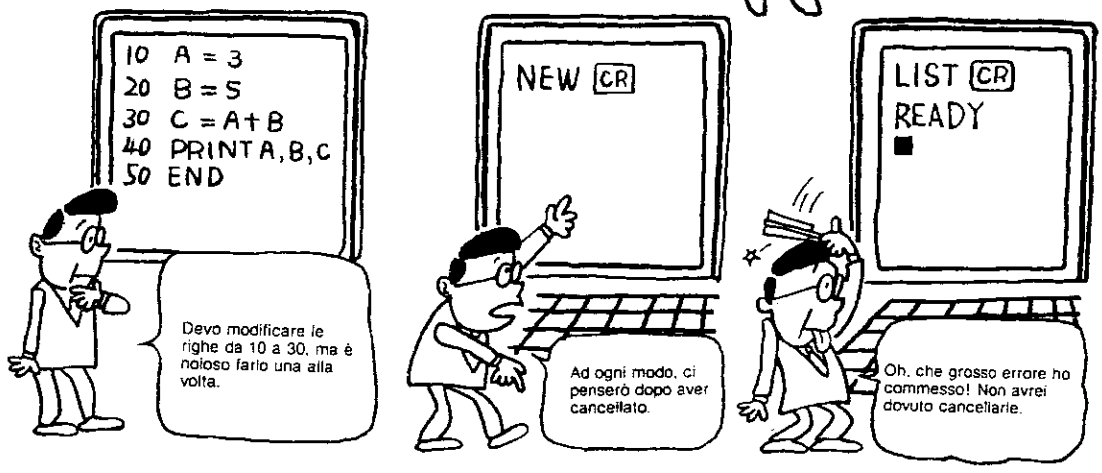
I risultati di NEW.....

Per memorizzare un nuovo programma, cancellare il programma precedente mediante il comando NEW. In caso contrario, i due programmi potrebbero sovrapporsi a causare confusione.

NEW CR

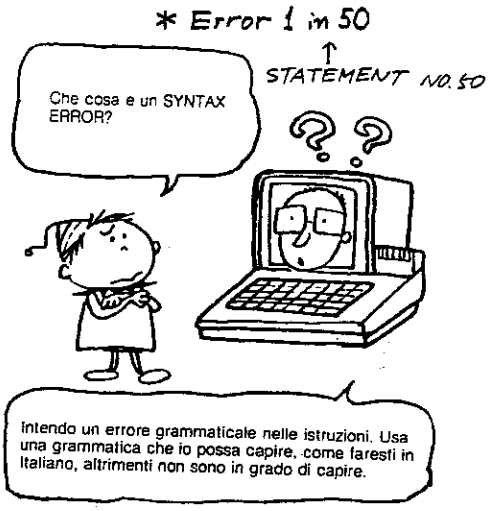
Questo comando cancella completamente il programma precedente. Per accertarvi che questa operazione è stata effettuata, immettere il comando LIST per controllare che il programma sia stato cancellato.





Gli errori confondono l'elaboratore

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
```



Questo è lo stesso programma usato in precedenza. Ha funzionato bene? Se in una qualsiasi riga vi è un errore, l'elaboratore ve lo dice. Per esempio.

```
50 EMD
```

Se vi sbagliate e battete una M invece che una N, l'elaboratore esegue il programma fino alla riga 40, ma non sa di che cosa si tratti quando incontra EMD. L'elaboratore vi segnala un errore di sintassi nel modo seguente:

```
* Error 1 in 50
```

Si deve allora immettere la riga giusta, 50 END. In caso abbia due righe con identico numero di riga, l'elaboratore tiene per buona la riga che è stata immessa per ultima. In questo modo il vostro programma è completo?

Se è così, provate a fare un errore nella riga 20, per esempio.

```
20  5 = B
```

Fatto questo, l'istruzione alla riga 20 deve venire corretta. Siete sicuri? ..... Usate il comando LIST per controllare.

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
205 = B
```

Oh bella, è successo qualche cosa di strano. La riga 20 non è stata cambiata. È poi una strana riga con numero di riga con numero di riga 205 viene listato. Questo è provocato dal fatto che l'elaboratore ha trascurato lo spazio in bianco tra 20 e 5 e ha considerato questi due numeri come un unico numero di riga. Lo spazio in bianco è del tutto privo di significato per l'elaboratore e viene trascurato.

Nota: Vedere pagina 120 per informazioni più dettagliate sugli errori.

Colleggi l'iztrusione ?

Potreste desiderare di eseguire una o più delle seguenti operazioni sul programma che è stato memorizzato.

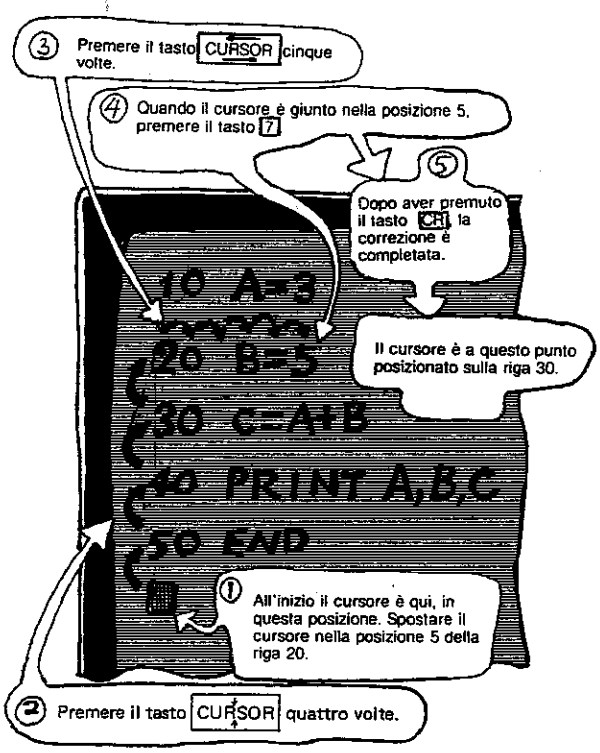
Correggere errori
Modificarlo per scrivere istruzioni migliori.
Modificarlo per separare alcune istruzioni.
Modificare parte di una istruzione all'elaboratore per generare una nuova istruzione.
.....

Vediamo come si possono eseguire modifiche, inserimenti e cancellazioni di righe e istruzioni.

Spostamento del cursore

Per poter rivedere i caratteri in una riga, il cursore deve venire spostato nelle rispettive posizioni corrispondenti. Vediamo di correggere l'istruzione B = 5 alla riga 20 sostituendo il 5 con un 7. Per come procedere agli spostamenti, fare riferimento alla figura a destra.

Con ciò il programma visualizzato sullo schermo TV è stato modificato. Di fatti, però, ciò non ha ancora modificato il programma registrato nella memoria dell'elaboratore. Per modificare la registrazione, deve essere schiacciato il tasto [CR]. Cosa hai fatto? Hai battuto il 6 al posto del 7? Per modificare il carattere alla sinistra del cursore ci sono due metodi, a scelta.



Metodo 1 Usando il tasto **INST·DEL**

Se si preme il tasto [INST·DEL], il cursore si sposta verso sinistra di un carattere, eliminando il carattere che si trova immediatamente sulla sinistra. Premere il tasto [7]. Naturalmente, alla fine si deve premere il tasto [CR].

Metodo 2 Spostamento a sinistra usando il tasto **CURSOR**

Premere il tasto [CURSOR] tenendo abbassato il tasto [SHIFT]. Il cursore si sposta a sinistra di un numero di posizioni pari al numero di volte che il tasto viene premuto.

■ Correggi l'istruzione!

Inserimento di caratteri

Cambiare C in D

Per modificare la riga 30 del programma a pagina 18 in modo che appaia così:

30 D = 100 + A + B ⟸ Non premere il tasto CR.

spostare il cursore sul carattere A. A questo punto, premere i tasti come indicato qui sotto.

Tenendo premuto il tasto SHIFT, premere quattro volte il tasto INST·DEL.

Si deve ottenere lo spazio esatto per aggiungere 4 caratteri e cioè 100 +. Immettere in questo spazio 100 +. Per cambiare C in D non serve altro. Visto che la riga è stata modificata, perché non modificare anche il suo numero da 30 a 35 e poi premere il tasto CR. Modificare la riga 40 come viene indicato qui di seguito.

```
40 PRINT A, B, C, D
```

Adesso premere RUN CR

```
RUN
3          5          8          108
```

Cancellazioni di carattere

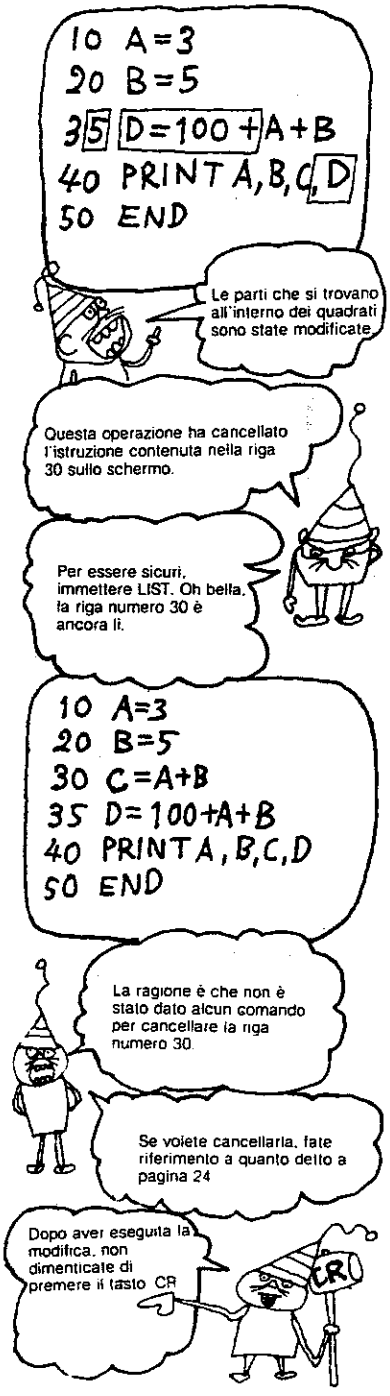
```
35 D = 100 + A + B
```

Modifichiamo questa riga in modo che risulti:

```
35 D = A + B + C
```

Spostare il cursore sul carattere A e premere il tasto INST·DEL 4 volte. Questa azione sposta il cursore fino a quando la porzione A + B arriva immediatamente alla destra del segno =.

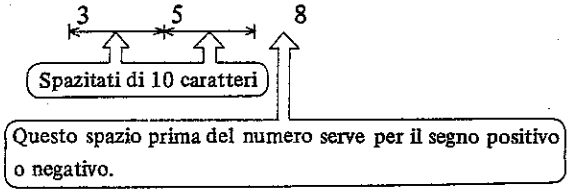
```
RUN
3          5          8          16
```

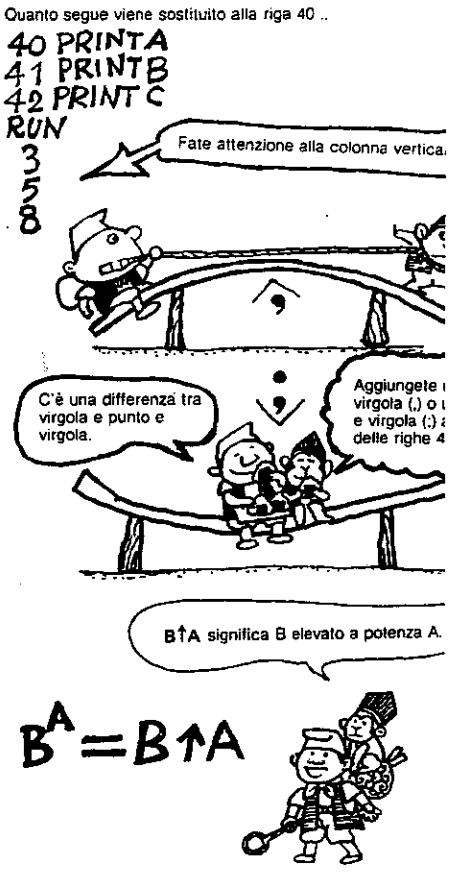


Ulteriore studio di"," e ","

Riprendiamo in esame l'esempio precedente.

```
10  A = 3
20  B = 5
30  C = A + B
40  PRINT A, B, C
50  END
```





Tutto questo ve lo ricordate, non è vero? In altre parole, quando tra A, B e C viene usata una virgola, i numeri vengono visualizzati con una spaziatura di 10 posizioni. Adesso scrivete un programma inserendo delle nuove istruzioni e fatelo funzionare. Le nuove istruzioni sono le seguenti:

```
32  D = B ↑ A
34  E = B * A
36  F = B / A
45  PRINT D, E, F
```

```
RUN
 3          5          8
 125        15         1.6666667
```

Fate funzionare il programma ancora una volta, sostituendo il punto e virgola (;) alla virgola (,). Per mc programma, immettete LIST e usate il cursore nel modo migliore che vi riesce.

```
RUN      Spazio per il segno positivo/negativo.
 3  5  8
 125    15  1.6666667
```

Il punto e virgola (;) ha la funzione di combinare assieme caratteri e simboli sullo schermo. Per accerta aggiungere un punto e virgola alla fine della riga 40 e poi eseguite di nuovo il programma.

```
40  PTINT A; B; C;
RUN
 3  5   8   125  15  1.6666667
```

I ":" ed il suo uso

Uso dei due punti

```
10  A = 3
20  B = 5
30  C = A + B
32  D = B ↑ A
34  E = B * A
36  F = B / A
40  PRINT A; B; C
45  PRINT D, E, F
50  END
```

Questo programma consiste di istruzioni brevi. Se si vuole, un programma di questa lunghezza può venire scritto in un'unica riga.

```
100 A = 3 : B = 5 : C = A + B : D = B ↑ A : E = B * A : F = B / A : PRINT
A; B; C : PRINT D, E, F : END
RUN 100
```

I due punti (;) sono un simbolo che deve venire usato quando nella stessa riga sono presenti due o più istruzioni. Questo tipo di riga viene chiamata "riga multipla". Una istruzione che occupa due righe può avere un unico numero di riga. Ogni riga consiste di 40 caratteri e quindi si possono usare 76 caratteri, compreso il numero di riga.

Quanto spazio mi è rimasto?..... SIZE

È naturale che desideriate sapere quanto spazio di memoria è ancora disponibile mano mano che i programmi vengono memorizzati uno dopo l'altro nell'elaboratore.

Per saperlo basta fare così:

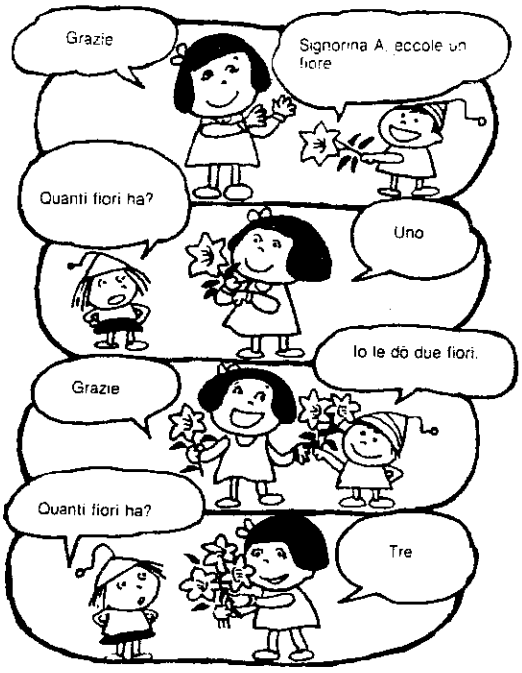
```
PRINT SIZE
```

In risposta a questo comando, l'elaboratore vi dice quanta capacità di memoria in byte avete ancora disponibile.

"A=B" equivale a "B=A"?

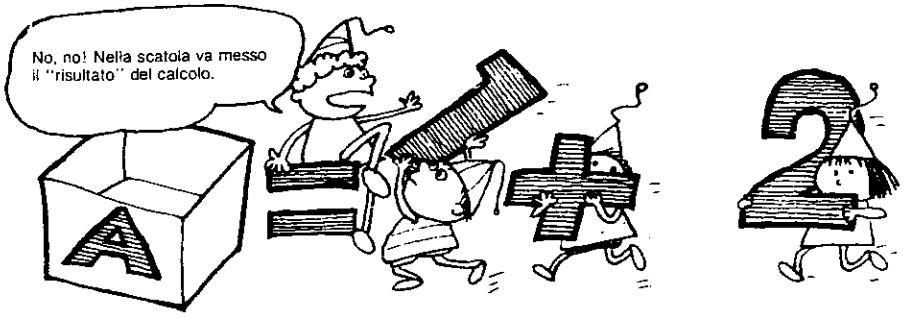
Facciamo adesso un po' di attenzione al segno = che abbiamo così spesso usato fino ad ora. Provate ad eseguire il programma seguente.

```
10  A = 1
20  PRINT A,
30  A = A + 2
40  PRINT A
50  END
RUN
 1            3
```



Sullo schermo compaiono 1 e 3. La riga 30 dice A = A + 2. Se questa fosse un'espressione, si potrebbe sottrarre A da ambedue i termini, ottenendo così 0 = 2, che è una contraddizione. Non si tratta quindi di un'espressione.

Il segno = significa che il risultato dell'espressione sulla destra viene sostituito nel simbolo A predisposto sulla sinistra.



Nella riga 10, il valore 1 viene sostituito nel simbolo A e nella parte di destra della riga 30 il valore di A e 2 vengono sommati e sostituiti al simbolo A usando il simbolo =.

A questo punto il valore 1, che era stato messo in A, non esiste più.

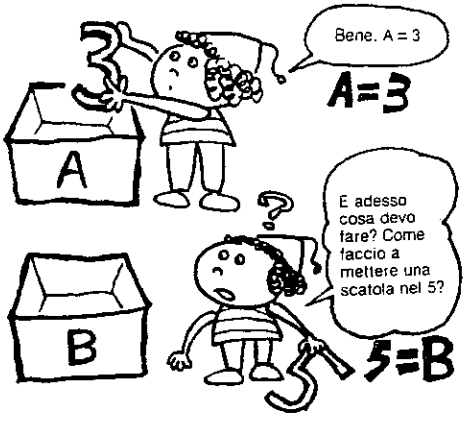
I due programmi che seguono ottengono risultati differenti, il che dimostra che "A = B" non è equivalente a "B = A".

```
10  A = 5
20  B = 7
30  PRINT A, B
40  A = B
50  PRINT A, B
60  END
RUN
 5            7
 7            7
```


```
10  A = 5
20  B = 7
30  PRINT A, B
40  B = A
50  PRINT A, B
60  END
RUN
 5            7
 5            5
```

L'elaboratore ama molto le variabili

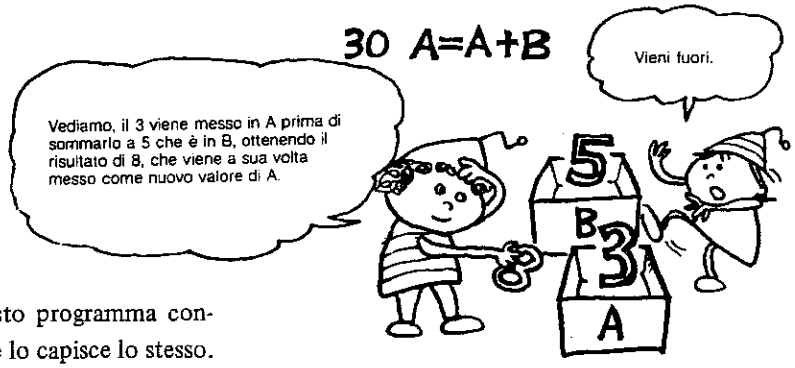
Le variabili che vengono usate nelle istruzioni per l'elaboratore hanno un uso diverso da quello delle variabili matematiche. Le variabili usate nelle istruzioni sono i nomi che vengono assegnati alle scatole in cui vengono messi dei valori.



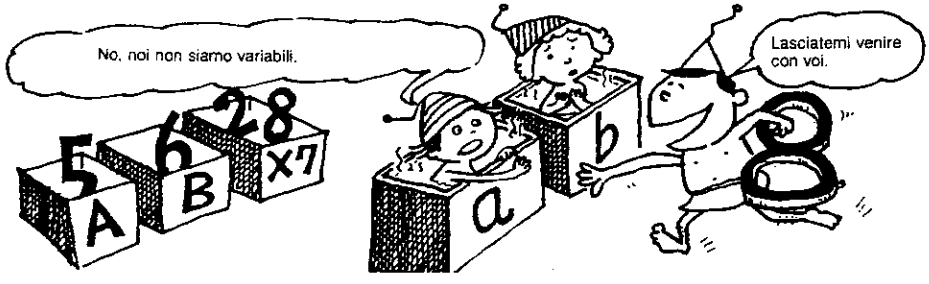

```
B = 5
```

Significa che il valore 5 dovrebbe venire messo nella scatola B. Di conseguenza, l'uso descritto nella pagina "Gli errori fanno confondere l'elaboratore" risulta in un'istruzione difficile per l'elaboratore, anche se non può essere confuso con un numero di riga. Questo perché dice di mettere la scatola B in 5.

```
10  A = 3
20  B = 5
30  A = A + B
40  PRINT A
50  END
RUN
 8
```



Dal punto di vista matematico, questo programma contiene una contraddizione, ma l'elaboratore lo capisce lo stesso.



Caratteri che si possono usare per le variabili

1. Un nome di variabile dovrebbe consistere di uno o due caratteri. Un nome di variabile con più di due caratteri può venire memorizzato, ma i caratteri dopo il secondo vengono trascurati durante l'elaborazione. Per esempio, ABC e ABD possono venire visualizzati, ma durante l'elaborazione essi vengono considerati uguali ad AB.
2. I seguenti caratteri possono venire usati nei nomi di variabili:
   (1) I caratteri alfabetici da A a Z.
   Esempio: A, M, Z
   (2) Le 260 combinazioni di lettere alfabetiche con i numeri da 0 a 9.
   Esempio: A0, K5, Z9
   (3) La combinazione di due caratteri alfabetici.
   Esempio: AA, BK, XZ
   Non si devono però usare alcune combinazioni, come IF, ON, TO che rappresentano dei comandi BASIC.

Un po' di conti sulla Terra

Il principe di una stella lontana sta facendo delle accurate indagini sulla Terra. "La Terra è un pianeta blu, laggiù nel Sistema Solare. Pur non essendo esattamente una sfera, si può dire che la Terra ha un diametro di circa 13.000 chilometri. Dai calcoli fatti sulla sua orbita, si può dedurre che la sua massa è circa 6×10^{18} mila tonnellate."

Il principe si reca al suo elaboratore per scrivere il programma che segue per il calcolo del volume VE, della superficie SE e della densità media ME della Terra.

```
10  DE = 13000 ..... Mette il diametro della Terra nella v
20  WE = 6E + 18 ..... Mette la massa della Terra nella vari
30  SE = 4 * π * (DE/2) ↑ 2 ..... Mette la superfice della Terra nella
40  VE = 4 * π * (DE/2) ↑ 3/3 ..... Mette il volume della Terra nella va
50  ZE = WE/VE * (1E – 2) ..... Mette la densità media della Terra n
60  PRINT "DIAMETRO DELLA TERRA": PRINT DE; "CHILOMETRI": P
70  PRINT "SUPERFICIE TERRESTRE": PRINT SE; "CHILOMETRI QUAI
80  PRINT "VOLUME TERRESTRE": PRINT VE; "CHILOMETRI CUBI": I
90  PRINT "MASSA TERRESTRE": PRINT WE; "MIGLIAIA DI TONNELL
100  PRINT "DENSITA' MEDIA TERRESTRE": PRINT ZE; "CHILOGRAM
110  END
```

Il principe della stella lontana ha notato che la dimensione della Terra è leggermer zione alle unità di misura usate nel calcolo. Bisogna anche stare molto attenti alla seq sione aritmetica contiene *, + oppure ↑. La priorità delle operazioni è indicata qui sott

1 ↑ (Calcoli esponenziali)
2 – (Segno del meno)
3 *, / (Moltiplicatione, divisione)
4 +, – (Addizione, sottrazione)

Le espressioni riportate qui sotto sono di formazione complessa. Avete notato

```
2 + 3 ↑ 2 = 11
(2 + 3) ↑ 2 = 25

12/3 * 2 = 8
12/(3 * 2) = 2

2 * 2 ↑ 3 = 16
(2 * 2) ↑ 3 = 64.000001

12/3 ↑ 2 = 1.3333333
(12/3) ↑ 2 = 16
```

Archimede e il soldato misterioso

La somma degli angoli interni di un triangolo è pari a 180 gradi. Archimede, con un lampo di genio, si siede sulla strada e disegna un triangolo. Ed ecco avvicinarsi un soldato misterioso con la sua lancia puntata su Archimede.

Soldato: Come è? L'angolo A è pari a 30 gradi e l'angolo B è un angolo retto?
È facile determinare l'angolo C: è di 60 gradi. Se si conosce la lunghezza del lato AC, si possono facilmente determinare le lunghezze dei lati AB e BC e anche l'area del triangolo.

Archimede: Non dire sciocchezze.

Soldato: Tutto ciò che si deve fare è scrivere un programma BASIC. Vediamo: ah si, viene bene con AC = 12.

```
10  A = 30 : B = 90 : CA = 12
20  AB = CA * COS (A * π/180)
30  BC = CA * SIN (A * π/180)
40  S = AB * BC/2
50  C = 180 – A – B
60  PRINT "AB =" ; AB, "BC = " ; BC, "CA = " ; CA
70  PRINT "AREA S = " ; S
80  PRINT "A = " ; A, "B = " ; B, "C = " ; C
90  END
```

Usando l'arcotangente ATN, determiniamo l'angolo C, note le lunghezze dei lati AB e BC. Per far ciò immettete:

```
50 C = ATN (AB/BC) * 180/π
```

Il risultato è in gradi. Si ottiene lo stesso risultato, non è vero?

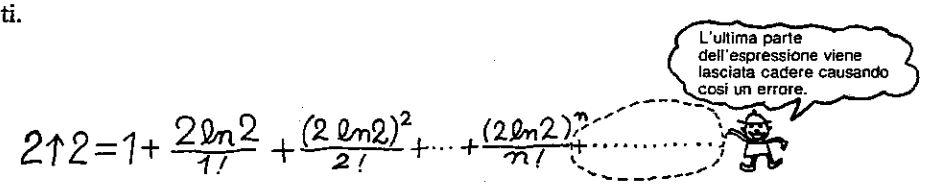
I membri della famiglia delle funzioni

Vi presentiamo qui alcune altre funzioni, come per esempio SIN (X). Queste funzioni vengono usate con delle parentesi, entro le quali possono venire messe delle costanti, delle variabili o delle espressioni.

Funzione	Simbolo BASIC	Valore calcolato	Esempio
Intero	**INT (X)**	Massimo intero inferiore a X	INT (3.14) = 3 INT (0.55) = 0 INT (–7.9) = –8
Valore assoluto	**ABS (X)**	Valore assoluto di X	ABS (2.9) = 2.9 ABS (–5.5) = 5.5
Segno	**SGN (X)**	1 se X è maggiore di 0 0 se X è uguale a 0. –1 se X è minore di 0.	SGN (500) = 1 SGN (0) = 0 SGN (–3.3) = –1
Funzione esponenziale	**EXP (X)**	e^X (e = 2,7182818)	EXP (1) = 2.7182818 EXP (0) = 1
Logaritmo in base 10	**LOG (X)**	$\log_{10} X$ X deve essere maggiore di 0	LOG (3) = 0.47712126
Logaritmo naturale	**LN (X)**	loge X X deve essere maggiore di 0	LN (3) = 1.0986123
Radice quadrata	**SQR (X)**	$\sqrt{X}$ X deve essere maggiore di 0	SQR (9) = 3 SQR (0) = 0

PRINT 2 * 2 è identico a PRINT 2 ↑ 2?

Be' quasi; 2 ↑ 2 dà come risultato 4.0000001. Come espressione aritmetica i due modi di scrivere sono identici, ma i calcoli della potenza vengono eseguiti su un numero limitato di cifre e quindi danno origine a errori inaspettati.



Questo tipo di calcolo può far gridare l'elaboratore. L'elaboratore può generare certi tipi di errore. Tali errori sono però di importanza trascurabile.

■ Definisci nuove funzioni... DEF FN

Abbiamo già descritte diverse funzioni e adesso parliamo di DEF FN, che permette di definire nuove funzioni che ragruppino quelle di cui abbiamo già parlato. Qui di seguito ne diamo alcuni esempi:

```
DEF  FNA (X) = 2 * X ↑ 2 + 3 * X + 1 . . . . . 2X² + 3X + 1 è definito come FNA (X)
DEF  FNB (X) = SIN (X) ↑ 2 + COS (X) ↑ 2 . . . . sin²X + cos² X è definito FNB (X), ed avrà sempre il
                                                  valore 1.
DEF  FNE (V) = 1/2 * M * V ↑ 2 . . . . . 1/2MV² è definito come FNE (V).
```

DEF sta per "definisci". Le nuove funzioni ricevono un nome composto da FN ed un suffisso. La X o la V tra parentesi viene chiamata l'argomento. Per esempio, la terza funzione (che rappresenta la quantità di moto) viene usata così.

```
10  DEF FNE (V) = 1/2 * M * V ↑ 2
20  M = 5.5 : V = 3.5
30  PRINT FNE (V), FNE (V * 2), FNE (V * 3)
40  END
```

La quantità di moto alla velocità iniziale V e quella ad una velocità doppia e tripla vengono visualizzate. Il comando DEF FN è estremamente utile specialmente quando la stessa funzione viene ripetutamente usata in un lungo programma.

Caduta da un'altitudine di 10.000 metri!

Come pensate che la velocità e l'altitudine varino al secondo per una caduta da 10.000 metri di altezza?

La funzione FNV (T) nel programma che segue calcola la velocità di caduta dopo un intervallo di tempo T e FNH (T) calcola l'altitudine allo stesso tempo.

La riga 20 assegna i valori della costante di gravità G, del fattore di resistenza atmosferica K e dell'altezza H da cui inizia la caduta.

```
10  ? " ■ " : T = 0
20  G = 9.8 : K = 0.15 : H = 10000
30  DEF FNV (T) = G/K * (1 – EXP (– K * T))
40  DEF FNH (T) = H – FNV (T) * T
50  ? " ■ "
60  PRINT "TEMPO      "; T : MUSIC "+" A0" . . . .Istruzione con suono, che verrà spiegata a pagina 76.
70  PRINT "VELOCITA"; FNV (T)
80  PRINT "ALTITUDINE"; FNH (T)
90  T = T + 1 : GOTO 50 . . . . . Questa è un'istruzione di salto che ordina al programma di
                                  saltare alla riga 50.
```

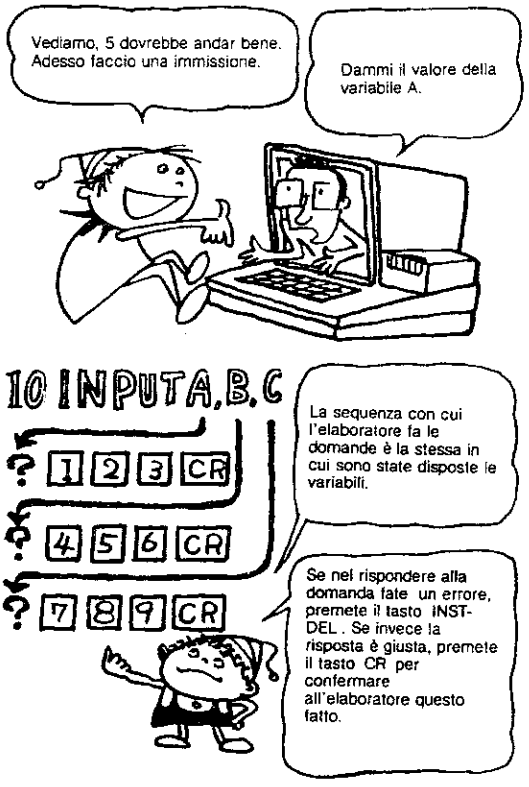
Qui è INPUT, rispondere, prego!

Fino ad ora per informare l'elaboratore del valori delle variabili abbiamo seguito il metodo di determinare per prima cosa il valore delle variabili stesse:

```
10  A = 3
20  B = 5
.....
```

Vi sono svariati altri metodi con cui si può informare l'elaboratore del valore assunto dalle variabili. Uno di tali metodi usa il comando INPUT.

```
10  INPUT A, B, C
20  D = A + B + C
30  PRINT A, B, C, D
40  END
RUN
? ■
```



Questo costituisce una visualizzazione nuova, non è vero? ? ■ vi sta domandando il valore della prima variabile, A, che segue il comando INPUT. In risposta a questa richiesta, immettere il valore della variabile e premere il tasto [CR] per informare l'elaboratore che tutto va bene.

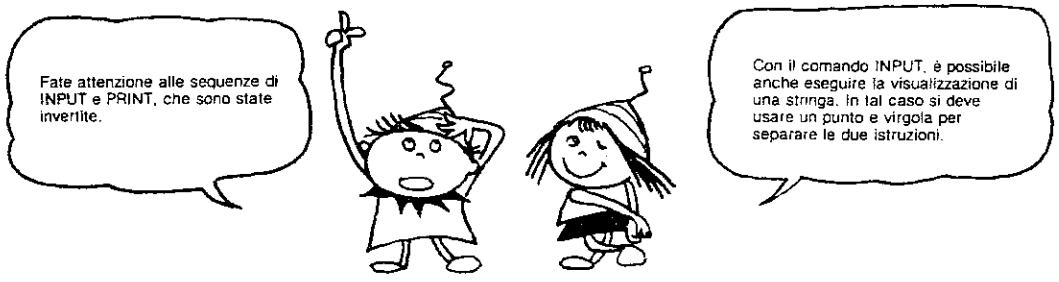
Guardate! Ecco di nuovo la stessa visualizzazione. Questa è la richiesta del valore della variabile B. Se vi sono 3 variabili, l'elaboratore pone la stessa domanda tre volte. Se per errore rispondete usando un tasto diverso da quelli dallo 0 al 9, e premete il tasto [CR], sullo schermo compare:

```
* Error 4 in 10 (Dati non dello stesso tipo)
```

L'elaboratore poi ricomincia a fare le domande relative ai valori.

```
10  INPUT A, B, C, D
20  INPUT E, F, G, H
30  PRINT H, G, F, E
40  PRINT D, C, B, A
50  END
```

```
10  INPUT "A = ?" ; A
20  INPUT "B = ?" ; B
30  INPUT "C = ?" ; C
40  S = A + B + C
50  M = S/3
60  PRINT "TOTALE"; S, "MEDIA"; M
70  END
```



■ "Sì" o "No" a una proposta di nozze?

In una bella Domenica soleggiata, un gentiluomo ed una signorina siedono faccia a faccia in un piccolo caffè. Lui ha 43 anni e lei ne ha 22.

Gentiluomo: – Ti amo dal primo momento in cui ti ho vista. Mi vuoi sposare?
Signorina: – Certo, se mi ami tanto. Non mi importa della differenza di età. Ma non adesso. Dovrai aspettare fino a quando la mia età sarà la metà della tua.

Supponiamo che l'età di lui sia A e quella di lei sia B e che X sia il numero di anni che lei gli ha chiesto di aspettare. Dopo X anni, lui avrà A + X anni e lei B + X. Dato che a questo punto l'età di lei sarà la metà di quella di lui, si richiede che sia soddisfatta la condizione A + X = 2 (B + X). Risolvendo l'equazione rispetto ad X, si ottiene:

X = A – 2B.

```
10  PRINT "QUANTI ANNI HA LUI?"
20  INPUT A
30  PRINT "QUANTI ANNI HA LEI?"
40  INPUT B
50  X = A – 2 * B
60  PRINT "ATTENDERE"; X; "ANNI!"
70  END
RUN
QUANTI ANNI HA LUI?
?  43 [CR]
QUANTI ANNI HA LEI?
?  22 [CR]
ATTENDERE – 1 ANNO!
```

Non è possibile aspettare – 1 anni. In altre parole, avrebbero potuto sposarsi un anno fa. Se domandate improvvisamente qualche cosa all'elaboratore, può darsi che lui sia confuso a proposito di quale variabile voi state parlando. In questo programma, nelle righe 10 e 30 viene inserita una stringa che indica il contenuto della richiesta, nella riga 60 viene anche usata una stringa per specificare la risposta.

Il metodo per usare l' INPUT adoperato in questo programma può venire semplificato. Modificare le righe 10 e 30 come indicato qui di seguito e cancellare le righe 20 e 40 dal programma.

```
10 INPUT "QUANTI ANNI HA LUI?"; A
30 INPUT "QUANTI ANNI HA LEI?"; B
```

Le righe che seguono dalla 50 in poi non vengono modificate.

```
RUN
QUANTI ANNI HA LUI? 43 [CR]
QUANTI ANNI HA LEI? 22 [CR]
ATTENDERE – 1 ANNO!
```

DATA e READ vanno mano nella mano

Un altro metodo per informare l'elaboratore a proposito del valore delle variabili.

```
10  READ A, B, C, D
20  X = A + B + C + D
30  PRINT X
40  DATA 3, 5, 7, 9
RUN
 24
```

Questo programma ottiene del valori che sono poi usati nell'esecuzione dei calcoli.

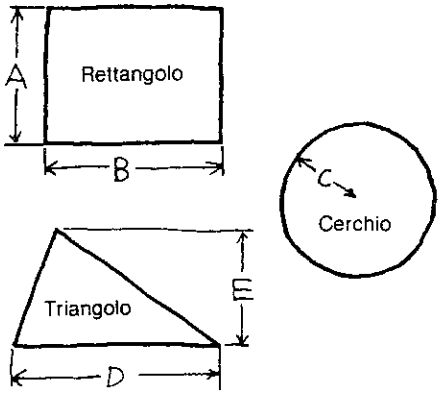
Con questo metodo vengono usati due tipi diversi di comando, READ e DATA.

READ A, B, C, D	Vengono predisposte alcune variabili
DATA 10, 11, 12, 13	Il numero dei valori è identico al numero delle variabili che seguono READ.

Come nel caso del comando INPUT, l'ordine delle variabili e dei loro corrispondenti valori deve essere lo stesso.

Usando i comandi READ e DATA è incredibilmente facile scrivere dei programmi che calcolano il valore dell'area di un rettangolo, di un cerchio e di un triangolo.

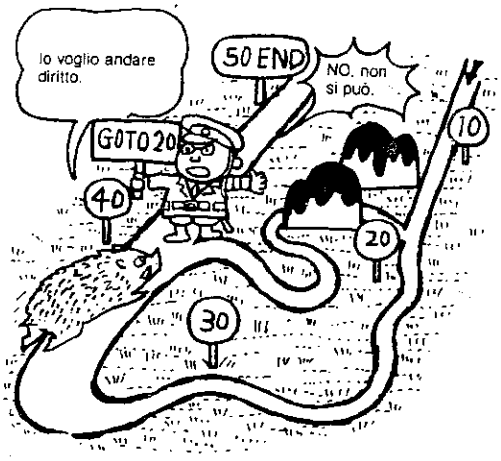
```
10  READ A, B
20  S1 = A * B
30  PRINT "RETTANGOLO=" ; S1
40  READ C
50  S2 = π * C ↑ 2
60  PRINT "CERCHIO = " ; S2
70  READ D, E
80  S3 = D * E / 2
90  PRINT "TRIANGOLO = " ; S3
100  DATA 2, 4, 6, 8, 10
110  END
```



Questo programma potrebbe venire migliorato. Provate voi in varie maniere.

Non opporti a un GOTO

L'elaboratore esegue tutti i programmi nella normale sequenza dal numero di riga più basso a quello maggiore. In alcuni casi però l'esecuzione del programma richiede che la sequenza venga modificata. In tali casi, è molto utile l'istruzione GOTO.




```
10  N = 1
20  PRINT N
30  N = N + 1
40  GOTO 20
50  END
RUN
 1
 2
 3
```

Non ci si ferma più? ..... Premere il tasto [SHIFT] e quello [BREAK].

In un tempo lontano, il grande cavaliere Ser Lancillotto del Lago compì una grande impresa per conto del Re Artù di Camelot.

Re Artù fu così riconoscente verso Ser Lancillotto che gli disse: "Ti darò qualisiasi premio tu mi richieda". Ser Lancillotto rispose: "Grazie mio Signore. Tutto ciò che ti chiedo è una ghinea oggi, due ghinee domani, 4 ghinee nel terzo giorno, 8 nel quarto e così via fino al 30simo giorno".

Re Artù fu così sorpreso da una richiesta così modesta che sidichiarò subito d'accordo.

Eseguiamo il programma riportato qui di seguito per trovare quanto deve pagare Re Artù.

```
10  D = 1 : F = 1 : S = 1
20  PRINT "GIORNI", "TOTALE GIORN.", "TOTALE"
30  PRINT D; TAB (10); F; TAB (24); S
40  D = D + 1 ..... Qui si aggiunge uno al numero dei giorni
50  F = 2 * F ..... Qui si moltiplica il valore corrente per due
60  S = S + F ..... Qui si calcola il totale sommando il valore odierno ai precedenti.
70  IF D = 31 THEN 90
80  GOTO 30
90  END
RUN
GIORNI          TOTALE GIORN.          TOTALE
   1            1                       1
   2            2                       3
   .            .                       .
   .            .                       .
   .            .                       .
   10           512                     1023
   .            .                       .
   .            .                       .
   .            .                       .
   20           524288                  1048575
   .            .                       .
   .            .                       .
   .            .                       .
   30           .53687091E + 09         .10737418E + 10
```

Il decimo giorno ha ricevuto 1023 ghinee, il ventesimo 1048575 e nel trentesimo giorno 1000000000 ghinee.

■ Il mondo di: IF... THEN

IF ~ THEN

10 IF △△△ THEN □□□
20 ■■■

Se △△△ condizioni sono soddisfatte, allora □□□ lavori possono venire eseguiti. Se no, salta □□□ e vai a ■■■ nella prossima riga. Questa è l'istruzione IF ~ THEN. Se □□□ è un numero, si salta alla riga che porta quel numero.

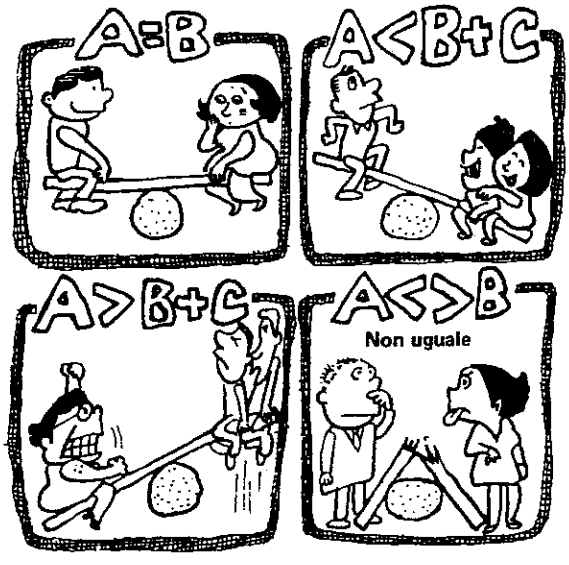
```
10 READ A
20 IF A >= 0 THEN PRINT "A =" ; A
30 GOTO 10
40 DATA -10, 20, 5, -9, 8, -6, 5
50 END
RUN
A = 20
A = 5
A = 8
A = 5
```

La forma generale dell'istruzione IF . . . . . THEN è la seguente:

IF condizioni THEN istruzione o numero di riga.

Le espressioni cui si fa qui riferimento sono del tipo "maggiore di" o "minore di" che fanno inoltre uso del segno di uguale o di non uguale.

Condizione	Come va usata
=	A = B
<	A < B + C
>	A > B + C
<=	A + B <= C
o =<	
>=	A >= B
o =>	
<>	A <> B
o ><	



"IF... THEN" e loro associati

Se . . . le condizioni sono soddisfatte (Si), viene eseguita l'istruzione che segue THEN. Se le condizioni non sono soddisfatte (No), l'esecuzione prosegue con la riga successiva. Ecco una presentazione dei vari tipi di IF . . . THEN.

IF . . . . . THEN GOTO oppure IF . . . . . GOTO
IF . . . . . THEN PRINT oppure IF . . . . . THEN?
IF . . . . . THEN A = 5 ✳ 7 istruzione sostituibile
IF . . . . . THEN INPUT
IF . . . . . THEN READ
IF . . . . . THEN GOSUB
IF . . . . . THEN RETURN
IF . . . . . THEN STOP
IF . . . . . THEN END


```
10 PRINT " ■ "
20 PRINT "BATTI UN NUMERO FRA 1 e 9."
25 PRINT "BATTI 0 PER TERMINARE."
30 L = 0 : M = 0 : N = 0
40 INPUT A
50 IF A = 0 THEN 90
60 IF A <= 3 THEN L = L + 1 : GOTO 40
70 IF A <= 6 THEN M = M + 1 : GOTO 40
80 N = N + 1 : GOTO 40
90 PRINT "HAI BATTUTO NUMERI"
100 PRINT "DA 1 A 3," ; L ; "VOLTE"
110 PRINT "DA 4 A 6," ; M ; "VOLTE"
115 PRINT "DA 7 A 9," ; N ; "VOLTE"
120 END
```

Alla riga 10 viene usato un nuovo simbolo: ■. È possibile visualizzare il simbolo ■ tra virgolette premendo il tasto [CLR HOME] tenendo abbassato il tasto [SHIFT] nel modo grafico. Questo comando cancella tutti i caratteri dallo schermo e sposta il cursore nell'angolo in alto a sinistra dello schermo.

Auando invece viene premuto il tasto [CLR HOME] travirgolette, compare il simbolo ■ nel modo grafico. La sola funzione di questo simbolo è di trasferire il cursore nell'angolo in alto a sinistra.

Se queste descrizioni non sono chiare, eseguite una prova con PRINT " ■ " o PRINT " ■ ".

Lascia decidere all'"IF"

IF può scegliere i numeri pari

Studiamo ora un programa che tra molti numeri scelga solo quelli pari, usando l'istruzione IF . . . THEN. IF ha delle eccezionali capacità nello scegliere i numeri.

```
10  READ X : IF X = –9999 THEN STOP
20  IF X/2 <> INT (X/2) GOTO 10
30  PRINT X ; : GOTO 10
40  DATA 2, 13, 56, 55, 4, 78, 31
50  DATA 6, 22, 15, 19, 80, 11, –9999
RUN
 2 56 4 78 6 22 80
```

INT (X/2) alla riga 20 è l'istruzione che permette di selezionare i numeri pari. Di conseguenza, se X è pari, X/2 <> INT (X/2) non può essere e il programma passa ad eseguire la riga 30. Se invece la condizione è possibile, il numero è considerato dispari e viene letto il valore successivo.

Per provare il vostro programma, facciamo qualche esercizio. Come si può decidere se un numero è un multiplo di 3 o di 4? Lo sapete, non è vero? La risposta è questa.

Modifiche per trovare i multipli di 3 .....

```
20 IF X/3 <> INT (X/3) GOTO 10
```

Modifiche per trovare i multipli di 4 .....

```
20 IF X/4 <> INT (X/4) GOTO 10
```

IF può scegliere il Minimo o il Massimo

```
10  S = 999 : L = –999
20  READ X : IF X = –9999 THEN 80
30  IF X > L THEN L = X
40  IF X > S THEN S = X
50  GOTO 20
60  DATA 2, –5, 91, 256, –43
70  DATA 87, 321, –76, –9999
80  PRINT "VALORE MASSIMO ="; L
90  PRINT "VALORE MINIMO ="; S
100 END
RUN
VALORE MASSIMO = 321
VALORE MINIMO = –76
```

La riga 10 è molto importante. Nella variabile S si deve mettere il numero più grande possibile, perché venga sostituito al valore minimo, e il numero più piccolo possibile deve venire messo nella variabile L, perché venga sostituito al valore massimo. Quali sono i risultati dell'esecuzione del programma? Le variabili L ed S vengono visualizzate ed hanno come contenuto il vero massimo e minimo. Questo è un ottimo esempio dell'uso di IF . . . . . THEN.

Una parola d'ordine per i numeri

Il massimo comun divisore è una specie di parola d'ordine per i numeri. Per esempio, supponiamo che i due numeri siano 10 e 20. I numeri per cui 10 e 20 sono divisibili sono 1, 2, 5, e 10. Tra questi numeri, il valore massimo, e cioè 10, costituisce il massimo comun divisore tra 10 e 20.

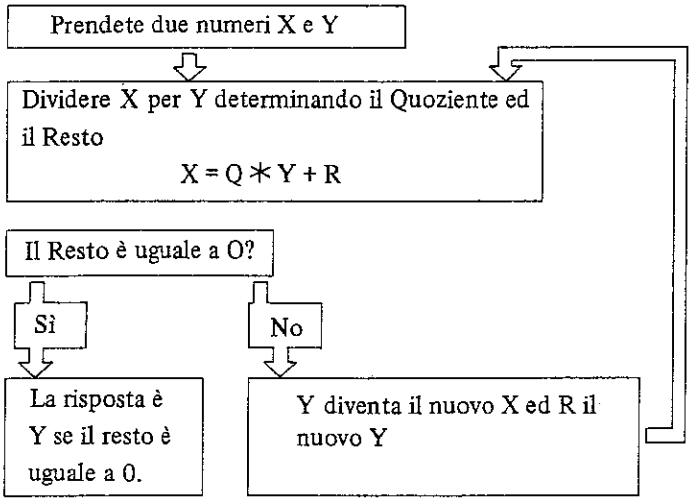
Ora impostiamo un programma

```
10  PRINT "X", "Y", "PAROLA D'ORDINE"
20  READ X, Y
30  PRINT X, Y
40  Q = INT (X/Y)
50  R = X – Q * Y
60  X = Y : Y = R
70  IF R > 0 THEN 40
80  PRINT X : GOTO 20
90  DATA 63, 99, 1221, 121, 64, 658
100   DATA 12345678, 987654321
110   END
RUN
```

La virgola (,) che segue la Y è molto comoda per ottenere una visualizzazione continua sullo schermo.

Vi facciamo vedere qual'è il trucco di questo programma!

Molto tempo fa, un matematico greco di nome Euclide ha sviluppato questo metodo per la soluzione del problema.



Fare il maggior numero di prove possibili usando IF.

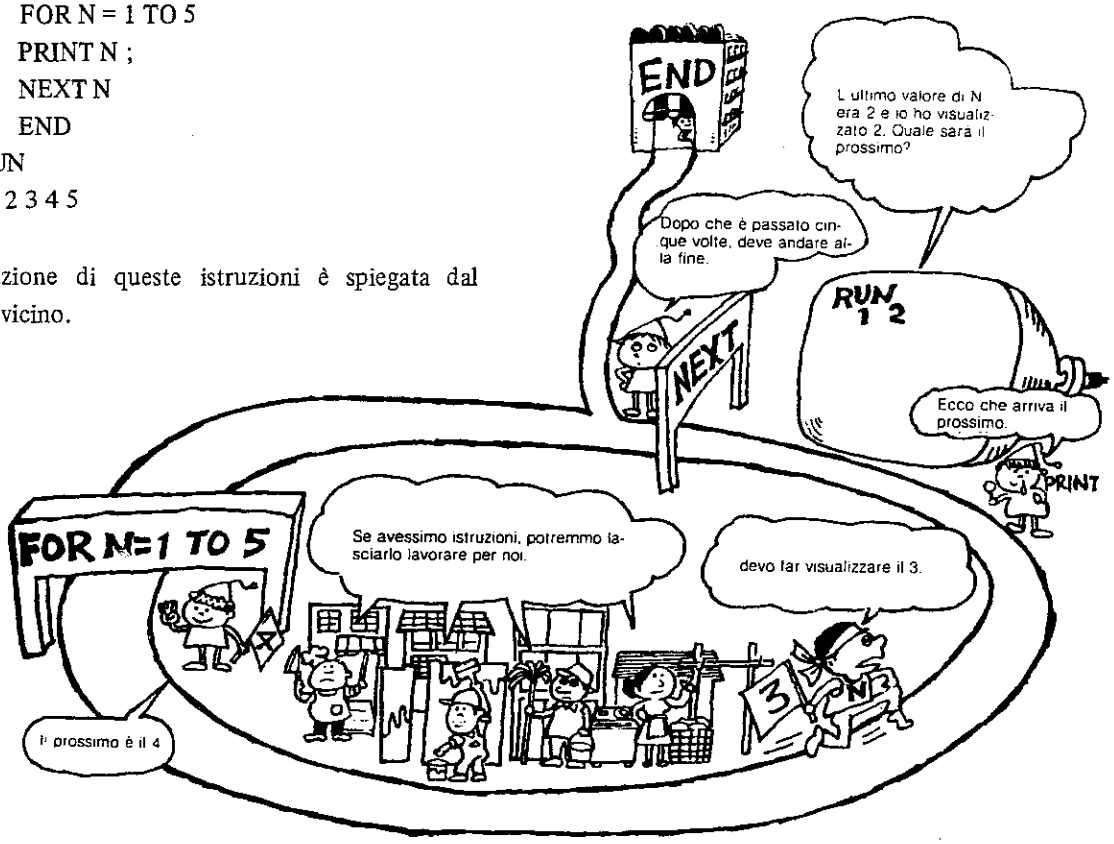
```
10  IF  SGN (X) = –1     THEN? "MENO"
20  IF  SGN (X) = 0      THEN? "ZERO"
30  IF  SGN (X) = 1      THEN? "PIU"
```

"FOR...NEXT" esperto in ripetizioni

L'istruzione FOR . . . . . NEXT viene usata per far ripetere una serie di istruzioni in un programma. Prima di tutto, diamo un'occhiata ad un programma semplice.

```
10  FOR N = 1 TO 5
20  PRINT N ;
30  NEXT N
40  END
RUN
 1 2 3 4 5
```

L'esecuzione di queste istruzioni è spiegata dal disegno qui vicino.



Le variazioni di N possono non solo essere degli incrementi di 1, ma anche, per esempio, degli aumenti di 0,5 o delle diminuzioni di 2. Ogni volta la variazione viene assegnata mediante la parola STEP.

Per far aumentare N con incremento di 0,5:

```
10 FOR N = 1 TO 5 STEP 0.5
```

Per farlo diminuire con decremento di 2:

```
10 FOR N = 5 TO 1 STEP –2
```

La forma di FOR . . . . . NEXT è la seguente

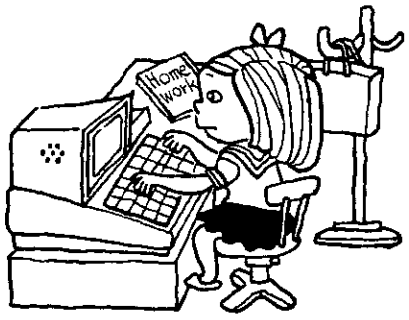
FOR variabile = Valore iniziale TO Valore finale STEP Variazione
Istruzioni inpetute
NEXT Variabile

Il valore iniziale, il valore finale e la variazione possono essere una constante, una variabile o un'espressione.

Loop in un loop

Alice sta facendo i suoi compiti. Sta preparando una tavola delle moltiplicazioni usando l'elaboratore ed un programma che contiene una doppia istruzione FOR . . . . . NEXT.

```
10  FOR X = 1 TO 9
20  FOR Y = 1 TO 9
30  PRINT X * Y ;        Loop Y
40  NEXT Y
50  PRINT                Loop X
60  NEXT X
```



Il loop per la variabile Y è compreso nel loop FOR . . . . . NEXT relativo alla variabile X. Entrambe le variabili X e Y variano da i a 9; ad X viene assegnato come valore iniziale 1 e poi viene eseguito il loop per Y. In altre parole, la variabile X rimane al valore 1 mentre la variabile Y assume i valori 1, 2, 3 . . . . . 9, e per ogni incremento viene visualizzato il risultato del prodotto di X per Y, come specificato dalla riga 30. Quando la variabile Y raggiunge il valore 9, la riga 50 fa eseguire una spaziatura di una riga e la riga 60 incrementa di 1 il valore della variabile X.

Il loop FOR . . . . . NEXT può venire usato molte volte, uno all'interno dell'altro, fino a un massimo di 15. Si deve però tener presente che i loop non si devono incrociare e che non è permesso saltare all'interno di un loop per mezzo di un istruzione GOTO.

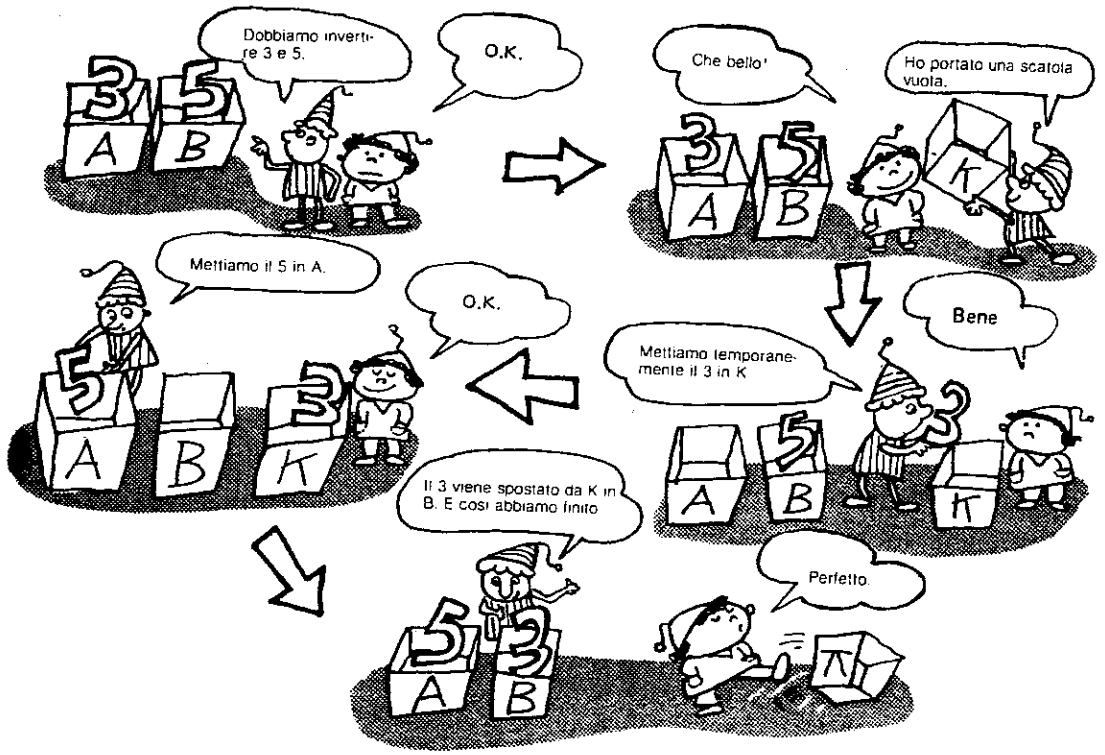
In questo modo, il loop C à completamente compreso nel loop B che a sua volta è completamente compreso nel loop A. Come si può vedere a destra, un'unica istruzione NEXT può venire usata per tutti i loop.

Allineamento in ordine numerico

L'elaboratore può disporre in ordine numerico crescente 4 numeri scelti a caso ed immessi. Il programma che segue svolge appunto tale funzione. Usate il comando INPUT.

```
10  PRINT " ■ "
20  PRINT "DIMMI 4 NUMERI": PRINT
30  INPUT A, B, C, D
40  IF A <= B THEN K = A : A = B : B = K
50  IF B <= C THEN K = B : B = C : C = K
60  IF C <= D THEN K = C : C = D : D = K
70  IF A < B GOTO 40
80  IF B < C GOTO 40
90  IF A < C GOTO 40
100  PRINT A, B, C, D
110 PRINT : PRINT "ANCORA UNA VOLTA, PREGO" : PRINT
120  GOTO 30
```

Fate attenzione alla riga 40. Grazie all'uso di un'altra variabile, K, l'instruzione THEN, il lavoro viene svolto invertendo i valori di A e B. Se i valori iniziali di A e B sono rispettivamente 3 e 5:



Con il lavoro descritto sopra si ottiene A = 5 e B = 3. Una elaborazione identica viene fatta nella riga 50 e 60. Le righe da 70 a 90 sono scritte per permettere la ripetizione dello stesso lavoro.

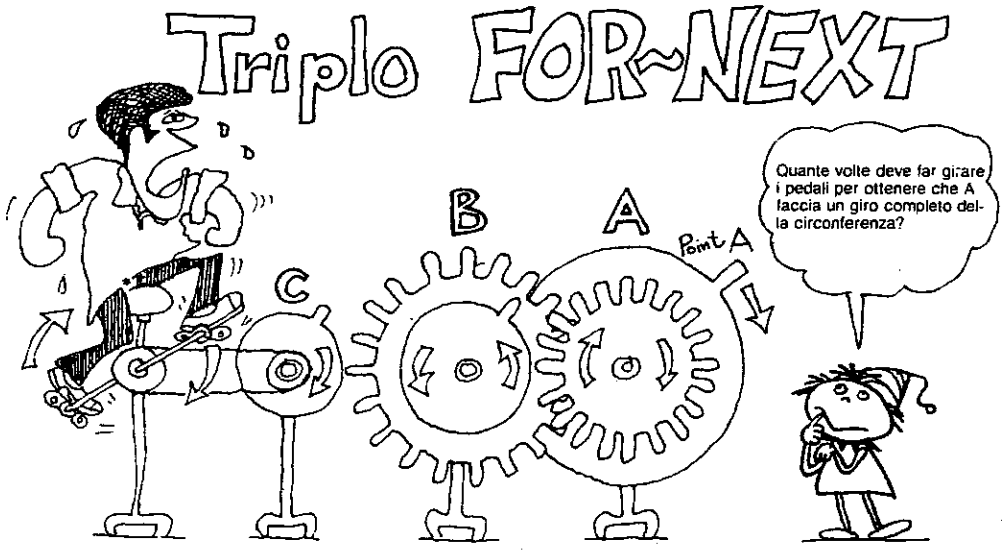
Quanti triangoli rettangoli ci stanno?

Scriviamo ora un programma che sceglie tutti i numeri tra 1 e 20 che soddisfano il Teorema di Pitagora $A^2 = B^2 + C^2$.

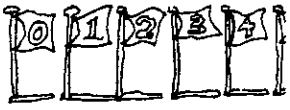
```
10 PRINT " ■ "
20 PRINT "      /\ "
30 PRINT "     /  \ "
40 PRINT "   B/    \A"
50 PRINT "   /      \ "
60 PRINT "  /________\ "
70 PRINT "             "
80 PRINT "      C"
90 PRINT : PRINT"INTERI POSITIVI CHE SODDISFANO IL TEOREMA DI PITAGORA"
110 PRINT : PRINT " ➡ A", " ➡ B", " ➡ C"
120 FOR A = 1 TO 20
130 FOR B = 1 TO 20
140 FOR C = 1 TO 20
150 IF A * A - B * B - C * C = 0 THEN PRINT A, B, C
160 NEXT C, B, A
180 END
```

Sapete già qual'è il significalo della riga 10. Cercate di fare bene il disegno in modo che grazie alle righe da 20 a 80 risulti sullo schermo un bel triangolo. Nelle righe tra 120 e 160 vi è un triplo loop FOR . . . . . NEXT. L'equazione contenuta nella riga 150 viene ripetuta 8000 volte (20 x 20 x 20) con A = 1 e C che varia da 1 a 20, poi con A = 1, B = 2 e C che varia da 1 a 20, e così via.

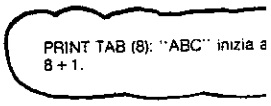
Questa operazione richiede per il suo svolgimento un tempo considerevole.



■ TAB( )è versatile



È possibile specificare dove iniziare a scrivere sullo schermo i caratteri o i simboli di una stringa. Per farlo si usa l'istruzione TAB ( ).



Se si specifica l'istruzione PRINT TAB (8); "ABC", la stringa ABC viene visualizzata a partire dalla posizione indicata dal numero tra parentesi a cominciare dall'estremità sinistra dello schermo, e cioè in questo caso dalla posizione 9.

I numeri che possono comparire tra le parentesi vanno da 0 a 78 e si può anche specificare una variabile, purché questa sia stata definita come numerica.

Facciamo un esempio con un semplice programma che usa anche l'istruzione FOR

```
10 FOR X = 1 TO 20
20 PRINT TAB (X) ; " * "
30 NEXT X
```

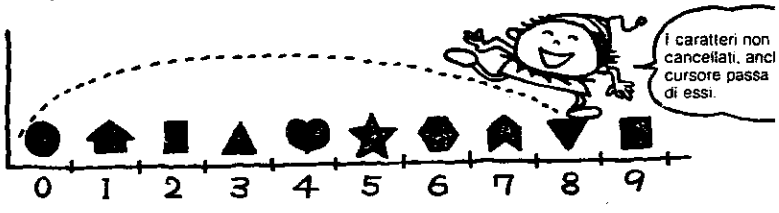
```
10 FOR Y = 1 TO 20
20 PRINT TAB (20 –
30 NEXT Y
```

E adesso proviamo a scrivere un programma un po' più difficile.

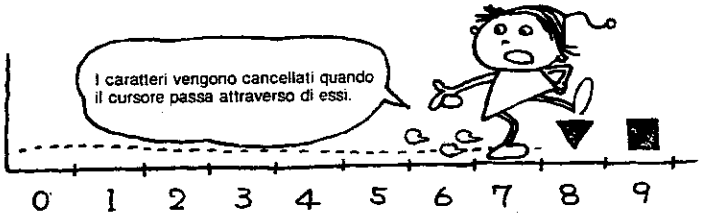
```
10 PRINT " ■ " : PRINT SPACE$ (8) ;
20 FOR X = 1 TO 22 : PRINT " * " ; : NEXT X : PRINT
30 FOR Y = 1 TO 20
40 PRINT TAB (8) ; " * " ; TAB (29 – Y) ; " ■ " ; TAB (29) ; " * " :
50 FOR Z = 1 TO 22 : PRINT " * " ; : NEXT Z
```

Qui, nelle righe 10 e 40, viene usata una nuova istruzione. Quando invece di T. ottengono esattamente gli stessi risultati. Tra SPACE$ e TAB vi è però la differenza illu

TAB (8) sposta sullo schermo il cursore di 8 caratteri a partire da sinistra.



SPACE$ (8) visualizza 8 spazi in bianco

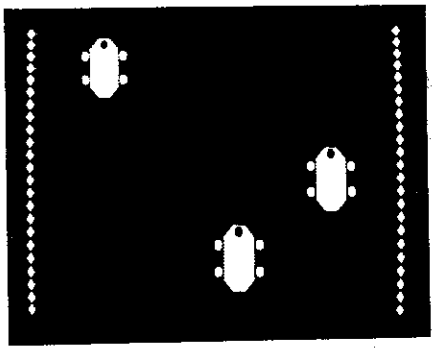


Un Gran Premio usando RESTORE

La sfida di una Gara Automobilistica.....

Che ne direste di un semplice programma che simula una gara automobilistica?

```
10  X = 33 * RND (1)
20  FOR A = 1 TO 5
30  READ M$
40  PRINT TAB (0) ; " ♦ " ; TAB (X) ; M$ ;
50  PRINT TAB (37) ; " ♦ "
60  NEXT A
70  Y = 10 * RND (1)
80  FOR A = 1 TO Y
90  PRINT TAB (0) ; " ♦ " ;
100 PRINT TAB (37) ; " ♦ " : NEXT
110 RESTORE : GOTO 10
120 DATA "  ","  "
130 DATA "  ","  "
140 DATA "  "
```



TAB (X) alla riga 40 determina dove visualizzare le automobili sulla strada a partire dal margine sinistro. Quale distanza deve esserci tra le automobili? Per rispondere a questo quesito, alla riga 70 vengono generati dei numeri a caso tra 1 e 9 e nelle righe da 80 a 100 le automobili vengono controllate in modo che non si scontrino. A proposito, RESTORE alla riga 110 non è un comando conosciuto, vero?

RESTORE fa ritornare all'inizio dei dati

L'instruzione DATA viene letta dall'istruzione READ, non importa dove la prima sia o quanto i dati siano sparpagliati.

O K.

```
10  DATA 27
20  READ A, B, C
30  DATA 10
40  ......
50  DATA 9, 13
60  READ D
70  END
```

NO

```
10  READ A, B
20  READ C
30  DATA 27, 10, 9
40  READ D
50  END
```

Perchè NO? Perchè la variabile D non ha alcun valore di DATA che possa venire letto.
E allora che cosa ne dite di questo?

```
10  READ A, B
20  READ C
30  DATA 27, 10, 9
35  RESTORE
40  READ D
50  END
```

L'istruzione RESTORE permette di ritornare a leggere il primo dato contenuto nella prima istruzione DATA del programma.

Stringhe chiacchierone

L'elaboratore dovrebbe generare un linguaggio comprensibile per gli esseri umani e dovrebbe parlare con noi . . . . . Per rendere questo desiderio una realtà, sono assolutamente necessarie le variabili di stringa.

```
10  A$ = "MIKE" : B$ = "PAUL"
20  C$ = "TONY" : D$ = "PETE"
30  E$ = "DENIS" : F$ = "MARTIN"
40  G$ = "PHILIP"
50  I$ = "JACK" : J$ = "HARRY"
60  K$ = "BILL" : L$ = "DAVID"
```

I normali simboli di variabile con un suffisso della lettera $ (il segno del dollaro) vengono chiamati variabili di stringa. Si possono elaborare in un modo molto simile a quello usato per le normali variabili. Vediamo alcuni esempi per riuscire a capire le loro caratteristiche.


```
70  PRINT B$
80  PRINT A$
RUN
PAUL
MIKE
```

L'uso di ";" collega le variabili di stringa.

```
100  PRINT B$ ; C$ ; A$ ; E$ ; L$ ; D$ ; K$
RUN  100
PAULTONYMIKEDENISDAVIDPETEBILL
```

Che cosa succede se si usa "," invece di ";"?

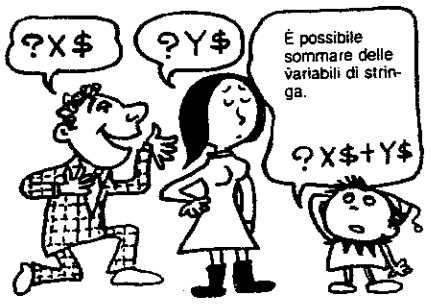
```
120  PRINT B$, A$, I$
RUN  120
PAUL          MIKE          JACK
```

10 caratteri di spaziatura

Se si vogliono combinare delle variabili di stringa in modo da generare una nuova stringa, sommare le variabili di stringa usando il simbolo "+".

```
140  X$ = B$ + C$ + D$ + J$ + G$
150  Y$ = A$ + C$ + B$ + F$ + I$ + G$
160  PRINT X$
170  PRINT Y$
```

In questo modo si può generare una nuova variabile di stringa.



Un altro tipo di INPUT

Cerchiamo di scrivere un programma che combina variabili di stringa e istruzioni INPUT per ottenere una poesia.

```
10  INPUT A$, B$, C$
20  PRINT A$; " "; B$; " "; C$
30  GOTO 10
RUN
? UNA RANA SALTA
? IN UNO STAGNO
? CON UNO SPRUZZO D'ACQUA.
UNA RANA SALTA IN UNO STAGNO CON UNO SPR UZZO D'ACQUA.
```

Gli spazi tra le variabili di stringa sono caratteri in comune.

Mediante l'uso dell'istruzione INPUT, si può immettere una stringa senza bisogno delle virgolette " ".

Il programma riportato qui sotto è un altro esempio.

```
10  PRINT "BATTI QUELLO CHE PREFERISCI"
20  INPUT AA$
30  PRINT "HAI APPENA BATTUTO"; AA$
40  GOTO 10
```

Le variabili di stringa, combinate con istruzioni di READ e DATA, possono venire generate all'interno del programma.

```
10  READ X1$, X2$
20  PRINT X1$; "PIACE LA" ; X2$
30  DATA TI, TORTA?
RUN
TI PIACE LA TORTA?
```

Prendete nota del fatto che le virgolette non sono necessarie quando si usa l'istruzione READ.

LEFT$, MID$, RIGHT$

LEFT$ ( ), MID$ ( ) e RIGHT$ ( ) sono istruzioni che permettono di generare delle nuove stringhe estraendo parte di stinghe già esistenti.

```
10  A$ = "ACQUARIO PESCI ARIETE LEONE"
20  B$ = LEFT$ (A$, 15)
30  PRINT B$
RUN
ACQUARIO PESCI
```

I caratteri fino al 15° a partire dell'estremità sinistra.

LEFT$ (A$, 15) sceglie i primi 15 caratteri a partire da sinistra della stringa A$ e li usa per generare una nuova stringa. Per la nuova variabile di stringa è stato scelto il nome di variabile di stringa B$.

Per scegliere i caratteri a partire dall'estremità di destra, si usa RIGHT$ ( ).

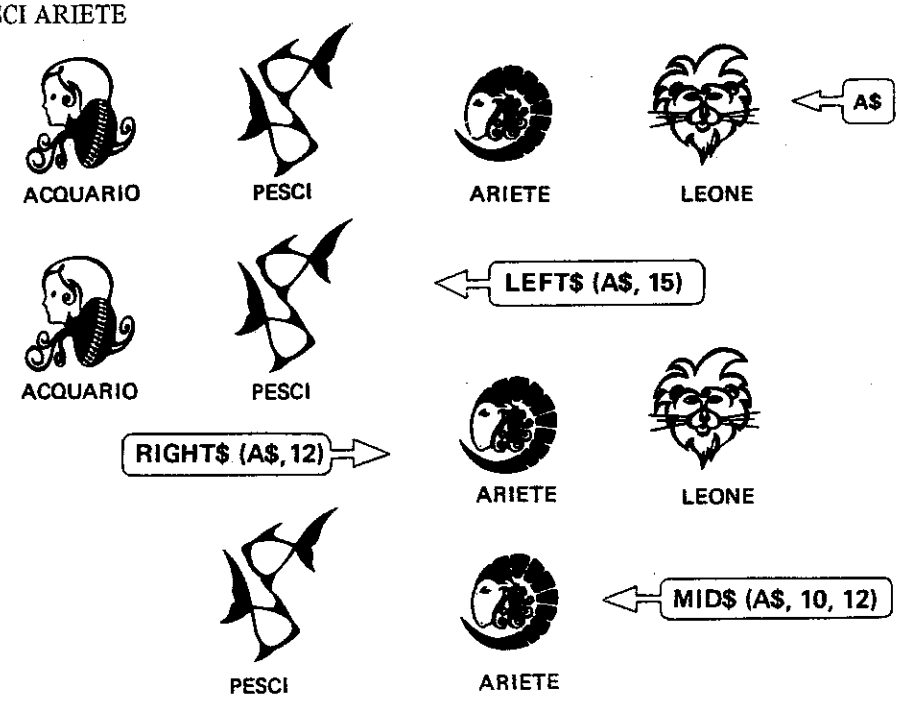
```
40  C$ = RIGHT $ (A$, 12)
50  PRINT C$
RUN
ARIETE LEONE
```

Scegliere gli ultimi 12 caratteri di A$

Per scegliere invece dei caratteri dalla parte di mezzo della stringa, si usa MID$ ( ).

```
60  D$ = MID$ (A$, 10, 12)
70  PRINT D$
RUN
PESCI ARIETE
```

Scegliere 12 caratteri partendo dalla posizione 10 di A$



LEN, unità di misura per stringhe

LEN ( ) viene usato per determinare quanti caratteri sono contenuti in una stringa. Ecco un semplice esempio di questa istruzione.


```
10  A$ = "ABCDEFG"
20  PRINT LEN (A$)
RUN
 7
```

Viene visualizzato il numero dei caratteri contenuti nella stringa A$, e cioé 7.

Ed ecco un programma che usa LEN ( ) per disegnare un quadrato.

```
10  PRINT " ■ " : PRINT "BATTERE ALMENO 3 CARATERI ＊ "
20  INPUT A$
30  FOR J = 1 TO LEN (A$) – 2
40  PRINT TAB (2) ; " ＊ " ; SPACE$ (LEN (A$) – 2) ; " ＊ "
50  NEXT J
60  PRINT TAB (2) ; A$ : GOTO 20
```

Variate il numero di ＊ che immettete con INPUT. L'elaboratore esegue il disegno dei quadrati utilizzando LEN ( ). È inoltre possible disegnare con simboli diversi da ＊ . Se si usa LEFT$ ( ), si devono modificare le righe 20 e 40 del precedente programma.

```
20  INPUT A$ : AA$ = LEFT $ (A$, 1)
40  PRINT TAB (2) ; AA$ ; SPACE $ (LEN (A$) – 2) ; AA$
```

L'utilizzo di LEN rende possibile eseguire una parata di stringhe.

```
10  S$ = "SHARP BASIC"
20  FOR M = 1 TO LEN (S$)
30  PRINT LEFT$ (S$, M)
40  NEXT M
RUN
S
SH
SHA
SHAR
SHARP
SHARP
SHARP B
SHARP BA
SHARP BAS
SHARP BASI
SHARP BASIC
```

```
10  S$ = "SHARP BASIC"
20  FOR M = 1 TO LEN (S$)
30  PRINT RIGHT$ (S$, M)
40  NEXT M
RUN
C
IC
SIC
ASIC
BASIC
 BASIC
P BASIC
RP BASIC
ARP BASIC
HARP BASIC
SHARP BASIC
```

■ ASC e CHR$ sono parenti

ASC

```
10  PRINT ASC ("A") ;
20  PRINT ASC ("ABC") ;
30  T$ = "Z" : PRINT ASC (T$)
40  END
RUN
  65  65  90
Ready
```

Quando tra le parentesi che seguono ASC vi sono delle stringhe, l'istruzione PRINT ha come risultato la visualizzazione di numeri. Questi numeri sono in effetti i relativi codici ASCII. Tutti i caratteri usati nell'elaboratore sono basati sul codice ASCII. ASC ( ) sceglie il codice ASCII corrispondente al primo carattere della stringa indicata tra parentesi. Ecco perchè si ottiene lo stesso risultato anche se le stringhe contenute nelle parentesi sono diverse nelle righe 10 e 20. Il codice ASCII vale per caratteri fino a 255.

CHR$

Se i caratteri possono venire convertiti in codice ASCII, è naturale che sia possible eseguire la conversione inversa. Esatto. L'istruzione CHR$ svolge questo compito.

```
PRINT CHR$ (65), CHR$ (ASC ("K"))
A            K
Ready
```

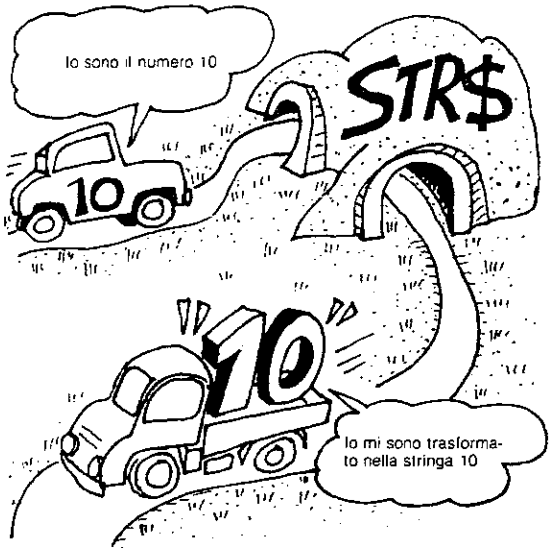
Usando dei numeri, viene scritto un messaggio cifrato. Lasciamo che CHR$ lo decifri.

```
10  FOR J = 1 TO 24 : READ A
20  B$ = CHR$ (A)
30  PRINT B$ ; : NEXT : END
40  DATA 73, 32, 83, 84, 85, 68
50  DATA 89, 32, 66, 65, 83, 73
60  DATA 67, 32, 79, 70, 32, 77
70  DATA 90, 45, 56, 48, 65, 46
RUN
I  STUDY  BASIC  OF  MZ – 80A.
Ready
```

STR$ e VAL convertono i numeri

STR$

```
10  A = 12 : B = 3 : C = A + B
20  C$ = STR$ (A) + STR$ (B)
30  PRINT C, C$
40  END
RUN
 15          123
Ready
```



Il valore della variabile A viene convertito in una stringa di caratteri da STR$ (A) ed elaborato come una stringa. La ragione per cui C$ contiene 123 dovrebbe esservi chiara. Nel programma che esegue, usate STR$ per allineare il "." dei dati.

Il risultato del programma a sinistra è il seguente:

```
10  FOR X = 1 TO 5
20  READ A
30  L = 5 – LEN (STR$ (INT (A) ))
40  PRINT TAB (L) ; A
50  NEXT : END
60  DATA 1 . 2 3 4 5 6, 1 2 .  3 4 5 6
70  DATA 1 2 3 . 4 5 6, 1 2 3 4 . 5 6
80  DATA 1 2 3 4 5 . 6
```

```
    1. 2 3 4 5 6
   1 2.  3 4 5 6
 1 2 3.  4 5 6
1 2 3 4.  5 6
1 2 3 4 5.  6
Ready
```

VAL

L'istruzione VAL ha la funzione opposta a quella dell'istruzione STR$. In altre parole converte una stringa di caratteri in numeri.


```
10  A$ = "1 2 3 4 5 6"
20  B = VAL (A$)
30  C = 6 5 4 3 2 1 + B
40  PRINT A$
50  PRINT B
60  PRINT C
80  END
RUN
1 2 3 4 5 6  ..... Non un numero, ma una stringa.
 1 2 3 4 5 6 ..... Numero e quindi viene lasciato uno spazio prima della cifra più significati
 7 7 7 7 7 7       del numero per + o – (segno positivo o negativo). In caso di un numero ne
Ready              tivo, nella posizione viene messo un segno meno.
```

Con PRINT: 123, 456, 789...

Questo programma legge un numero fornito mediante l'istruzione INPUT e aggiunge una virgola (,) ogni 3 cifre a partire dalla destra. Se il numero dato è 0, il programma termina.

```
10  PRINT "BATTI UN NUMERO INTERO";
20  INPUT X$
30  IF X$ = "0" THEN END
40  PRINT "£" ;
50  FOR Y = 1 TO LEN (X$)
60  PRINT MID$ (X$, Y, 1) ;
70  Z = LEN (X$) – Y
80  IF Z/3 <> INT (Z/3) GOTO 110
90  IF Z = 0 GOTO 110
100  PRINT " , " ;
110  NEXT Y
120  PRINT : PRINT : GOTO 10
RUN
BATTI UN NUMERO INTERO ? 1 2 3 4 5 6 7 8 9
£ 123,456,789

BATTI UN NUMERO INTERO ? 1 2 3 4
£ 1,234
```

La riga di istruzioni 80 controlla se Z (posizioni di caratteri contate dalla destra) è un multiplo di 3. Se è così, la riga 100 mette una virgola. Per esempio, supponendo che il numero sia un numero di 9 cifre, si ottiene il risultato seguente.

	Conteggio dei caratteri del numero intero								
	← LEN (X$) →								
BATTI UN NUMERO INTERO	□	□	□	■	□	□	□	□	□
Y VARIABILE	1	2	3	4	5	6	7	8	9
Z VARIABILE	8	7	6	5	4	3	2	1	0
				← Z = LEN (X$) – Y →					

Scegliete un numero le cui cifre siano comprese fra 1 e 4 ed un altro numero che consiste delle stesse cifre, ma in ordine inverso e sommate questi due numeri. Troverete che il risultato della somma può venire letto indifferentemente da destra o da sinistra.

```
10  PRINT " ◙ BATTI UN NUMERO COMPOSTO"
15  PRINT "DA 1 A 4 (MAX 8 CIFRE)"
20  Z$ ="": INPUT X$
30  FOR K = LEN (X$) TO 1 STEP – 1
40  Y$ = MID$ (X$, K, 1)
50  Z$ = Z$ + Y$ : NEXT K : X = VAL (X$) + VAL (Z$)
60  PRINT X : PRINT : GOTO 20
```

Qual'è la differenza tra interessi semplici e composti?

In una banca vengono depositate 10.000 sterline e alla fine dell'anno ne vengono ritirate 10.600. Il tasso di interesse in questo caso, risulta essere 600/1000 = 0,06 ovvero il 6%. Se le cose stanno così, qual'è l'interesse se il depositoviene lasciato per 2 anni? Vi sono due diversi metodi per il calcolo degli interessi. Uno è il calcolo dell'interesse semplice basato sul fatto che l'interesse dovuto per il primo anno limita a raddoppiare, diventando così 1200 sterline. L'altro è il metodo dell'interesse composto, basato sull'idea che il deposito all'inizio del secondo anno è di 10600 sterline con un interesse di 636 sterline (pari ancora al 6%) che, sommate alle prime 600 danno un totale per i due anni di 1236 sterline. Il calcolo degli interessi con il metodo composto è leggermente migliore per quanto riguarda il tasso di interesse. Per una somma di denaro abbastanza grande e depositata per un lungo numero di anni, la differenza nel tasso di interesse derivante dall'applicazione dei due metodi di interesse è notevole. Qui sotto riportiamo le formule per il calcolo degli interessi con i due metodi.

Interesse calcolato con il metodo semplice (n anni a tasso R)

$$B = X \text{ (capitale)} + n \cdot X \cdot R$$

Interesse calcolato con il metodo composto (n anni a tasso R)

$$C = X \cdot (1 + R)^n$$

Basandosi su queste formule, scrivere il programma che segue, che calcola gli interessi sia nel caso semplice che in quello composto.

```
10  PRINT "CAPITALE"
20  INPUT X
30  PRINT "TASSO D'INTERESSE"
40  INPUT R
50  PRINT "NUMERO DI ANNI"
60  INPUT Y : PRINT : PRINT
70  PRINT "CAPITALE =" ; X
80  PRINT "TASSO D'INTERESSE =" ; R ; "%"
90  PRINT "ANNI" ; TAB (6); "SEMPLICE" ;
100  PRINT TAB (17) ; "COMPOSTO" ;
110  PRINT TAB (30) ; "DIFFERENZA"
120  FOR A = 1 TO Y
130  B = X + A * X * (R/100)
140  C = INT (10 * X * (1 + R/100) ↑ A)/10
150  D = C – B
160  PRINT A ; TAB (6) ; B ;
170  PRINT TAB (15) ; C ; TAB (30) ; D
180  NEXT A
190  PRINT : PRINT : GOTO 10
```

Qui sotto riportiamo un esempio di esecuzione del programma:

```
CAPITALE = 10000
TASSO D'INTERESSE = 6%
ANNI SEMPLICE COMPOSTO     DIFFERENZA
 1      10600    10600       0
 2      11200    11236       36
 3      11800    11910.1     110.1
 4      12400    12624.7     224.7
 5      13000    13382.2     382.2
 6      13600    14185.1     585.1
 7      14200    15036.3     836.29999
```

Reddito annuo col deposito di 5 anni

Nell'esempio precedente abbiamo dato uno sguardo alla differenza tra interessi semplici e interessi composti nel caso di deposito di una certa somma. In realtà, di solito siamo più familiari con i depositi a mese, come ad esempio i depositi vincolati. Se ogni mese viene depositato un ammontare fisso, X, l'interesse aumenta il primo anno di X (1 + R), il secondo anno di X $(1 + R)^2$ e così via. Inoltre, se il primo anno viene depositata la cifra X, l'anno successivo si dovrà depositare la somma X (1 + R).

Interesse dopo un anno (X capitale, R tasso di interesse)

$$M_1 = X (1 + R)$$

Interesse dopo due anni

$$M_2 = X (1 + R)^2 + X (1 + R)$$

Interesse dopo tre anni

$$M_3 = X (1 + R)^3 + X (1 + R)^2 + X (1 + R)$$

Basandosi su queste formule, l'interesse dopo n anni viene calcolato in questo modo:

$$M_n = X (1 + R)^n + X (1 + R)^{n-1} \ldots\ldots + X (1 + R)$$

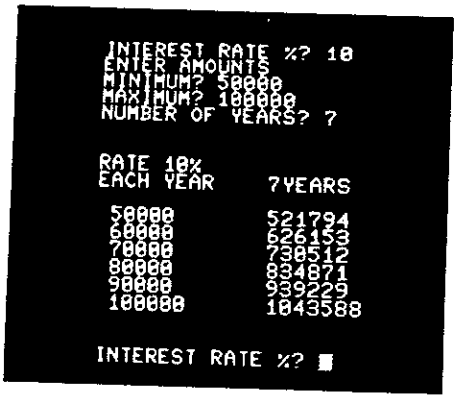
Questa espressione viene semplificata nel modo seguente:

$$M_n = X ((1 + R)^{n+1} - (1 + R))/R$$

Ecco il programma che dice qual'è l'interesse maturato per un anno qualsiasi depositando ogni anno la stessa somma. Anche se l'ammontare depositato ogni anno è lo stesso, il programma è strutturato per accettare anche importi minimi e massimi.

```
10  PRINT "TASSO D'INTERESSE %" ;
20  INPUT R
30  PRINT "BATTI L'AMMONTARE"
40  PRINT "MINIMO" ; : INPUT L
50  PRINT "MASSIMO" ; : INPUT H
60  PRINT "NUMERO DI ANNI" ;
70  INPUT Y : PRINT : PRINT
80  PRINT "TASSO" ; R ; "%"
90  PRINT "OGNI ANNO" ; TAB (12); Y ; "ANNI"
100  R = R/100 : PRINT
110  FOR A = L TO H STEP 10000
120  B = INT (A * ((1 + R) ↑ (Y + 1) - (1 + R))/R)
130  PRINT A ; TAB (12) ; B
140  NEXT A
150  PRINT : PRINT : GOTO 10
```

The Result of Program Execution

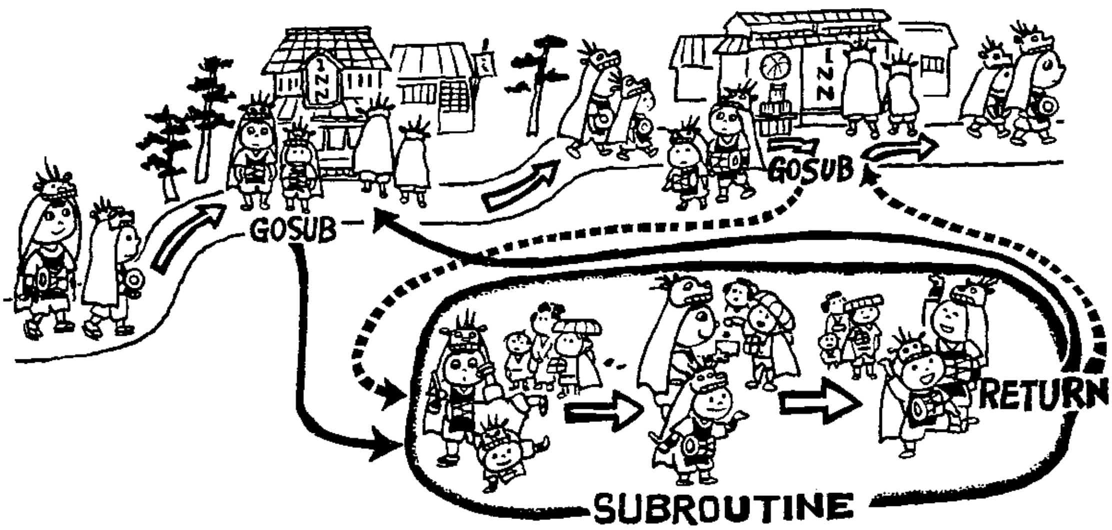


■ SUBROUTINE, asso dei programmi

In quasi tutti i programmi certi lavori vengono eseguiti ripetutamente. Tali lavori possono venire raggruppati in subroutine a cui si può saltare in qualsiasi momento da qualsiasi punto in cui ciò sia richiesto dal programma principale. Questi sottoprogrammi vengono appunto chiamati subroutine e con essi si deve usare l'istruzione GOSUB. Qui sotto riportiamo un esempio di un programma che usa l'istruzione GOSUB.

```
10  PRINT " ◙ "
20  PRINT TAB (10) ; " ** TOTALE VENDITA ** " : PRINT
30  PRINT "          5         10         15         20         25         30         35"
40  PRINT " □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
50  PRINT " □     FARINA"     : GOSUB 200
60  PRINT " □     ZUCCHERO"   : GOSUB 200
70  PRINT " □     VINO"       : GOSUB 200
80  PRINT " □     SALSA"      : GOSUB 200
110 PRINT " □ "
120 PRINT : END
200 PRINT " □ " ; : READ A
210 FOR N = 1 TO A : PRINT " ▩ " ; : NEXT N
220 PRINT A
230 RETURN
240 DATA 20, 15, 21, 24
```

Nel programma che precede, la subroutine è rappresentata dalle righe da 200 a 230. Mediante l'istruzione RETURN, alla fine della subroutine, il controllo dell'esecuzione viene restituito al programma principale.



Arresto, fermata continuazione

Non sempre l'elaboratore lavora come noi desideriamo, anche se sta eseguendo un programma che abbiamo scritto noi. Per questo è necessario inserire una istruzione STOP, in modo da poter controllare il contenuto delle variabili al momento in cui l'elaboratore si ferma. Per esempio, nel seguente programma è stata inserita una istruzione STOP.

```
10 READ A, B
20 X = A * B
30 STOP
40 Y = A/B
50 END
60 PRINT X, Y
70 DATA 15, 5
80 END
RUN
Stop in 30
```

A questo punto, la visualizzazione delle variabili viene eseguita in modo diretto, come segue:

PRINT A, B, X [CR]

Questo vi permette di controllare il programma. Per far ripartire il programma, dare il seguente comando all'elaboratore:

CONT [CR]

Il comando CONT viene usato quando:

- L'esecuzione del programma viene arrestata premendo i tasti [SHIFT] + [BREAK].
- L'esecuzione del programma è stata arrestata mediante le istruzioni STOP e END.
- Si smette di immettere dati in risposta ad una istruzione INPUT premendo i tasti [SHIFT] + [BREAK].

Il comando CONT non può venire usato:

- Prima che il programma sia stato fatto partire con il comando RUN.
- Quando, dopo che l'esecuzione è stata fermata, il programma viene corretto.
- Se durante l'esecuzione si verifica un errore Il programma ritorna alla condizione di "Ready".
- Quando per fermare il funzionamento della cassetta a nastro si usa il tasto [BREAK].
- Quando il comando MUSIC che produce la musica viene fatto fermare.

L'elaboratore riprende l'esecuzione dal punto in cui si era fermato. Quando raggiunge il comando END alla riga 50, l'elaboratore si arresta dopo aver visualizzato "Ready". Allora, in modo diretto, immettete:

X = 3 : Y = 5 [CR]

L'elaboratore, quando gli darete il comando CONT, riprende l'esecuzione del programma e visualizza 3 e 5 per le variabili X e Y.

Il programma che segue continua a visualizzare un triangolo fatto * senza interruzione.

```
10 A = 0 : B = 38 : C = 1
20 FOR X = A TO B STEP C
30 FOR Y = 0 TO X
40 PRINT " * " ;
50 NEXT Y : PRINT : NEXT X
60 K = A : A = B : B = K
70 C = -C : GOTO 20
```

L'esecuzione di questo programma viene interrotta premendo il tasto [BREAK] mentre è abassato il tasto [SHIFT]. Inserire quanto segue nel programma e poi dare il comando CONT.

100 END

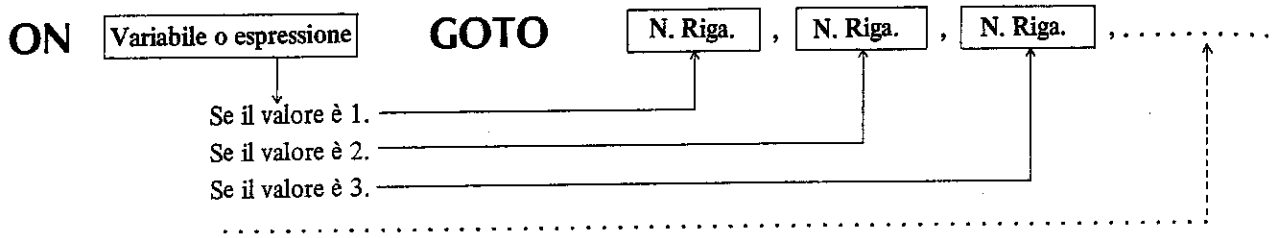
Ciò provoca la visualizzazione del seguente messaggio:

* Error 17 ..... Il comando CONT non può essere eseguito

Il comando CONT non può venire usato quando un programma viene corretto usando un numero di riga dopo che si è arrestato a casusa di una istruzione STOP, END o della pressione sul tasto [BREAK] . Questo deve essere sempre ricordato.

Salti in massa usando "ON... GOTO"

Ormai avete imparato molto sull'istruzione GOTO. Vi diamo adesso la descrizione di ON . . . . . GOTO una estensione dell'istruzione GOTO.



Per esempio, se il valore della variabile o dell'espressione che segue il ON è 3, viene effettuato un salto al numero di riga terzo dopo il GOTO. In altre parole, è possibile stabilire i salti di un programma in funzione dei valori delle variabili.

```
10  INPUT "NUMERO (1 – 3)?" ; A
20  ON A GOTO 50, 60, 70
50  PRINT "X X X" : GOTO 10
60  PRINT "Y Y Y" : GOTO 10
70  PRINT "Z Z Z" : GOTO 10
NUMERO (1 – 3) ? 1
X X X
NUMERO (1 – 3) ? 2
Y Y Y
NUMERO (1 – 3) ?
```

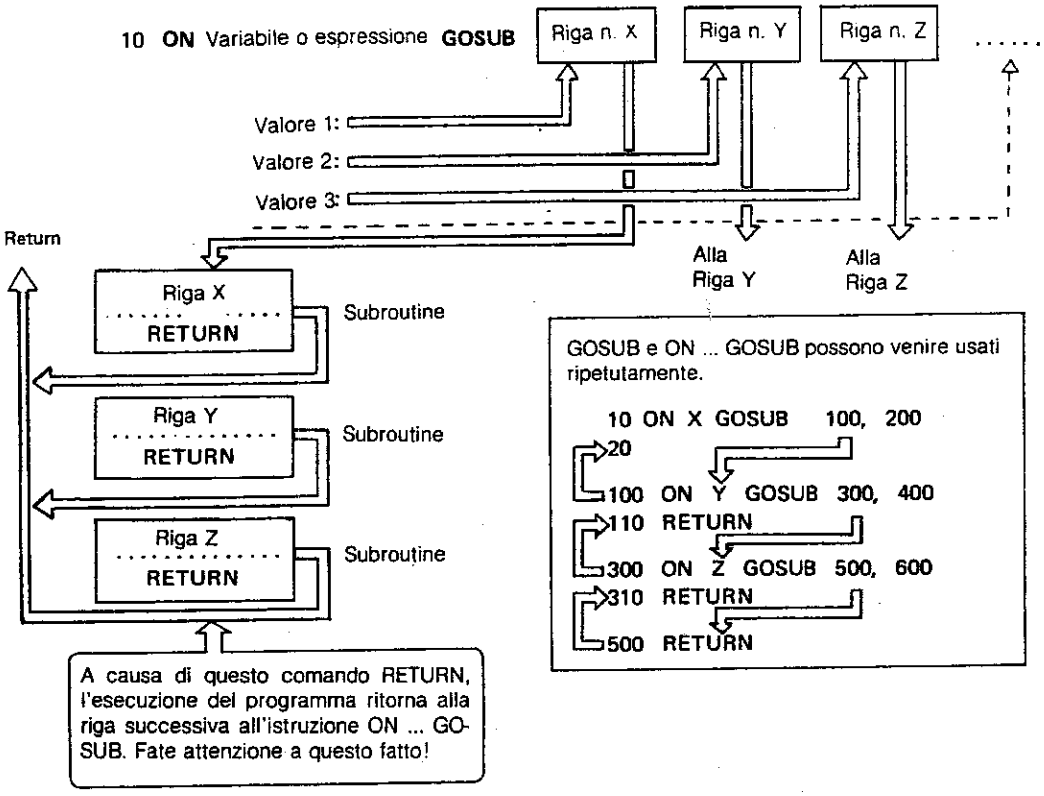
Se si risponde 1,2 per esempio, viene elaborato il numero intero 1, tralasciando tutta la parte decimale.

Facciamo il gioco di "trovare la matta" usando l'istruzione ON . . . . . GOTO. Prendiamo cinque carte, una delle quali è la matta. Naturalmente nessuno sa quale delle cinque carte è la matta. Cercate di indovinare dove sia la matta e battere l'elaboratore. Quando il programma vi domanda "Che cosa vuoi fare?", rispondete 1 se volete passare, 2 se non volete passare e 3 se non volete giocare. Vi è permesso passare tre volte.

```
10  R = INT (5 * RND (1)) + 1
20  N = N + 1 : IF N = 6 THEN 120
30  INPUT "COSA VUOI FARE?" ; X
40  ON X GOTO 60, 90, 50
50  PRINT "MANO NON GIOCATA !!!" : GOTO 120
60  NP = NP + 1
70  IF NP > = 4 THEN NP = NP – 1 : N = N – 1 : PRINT "NON PUOI PASSARE"
80  GOTO 20
90  NR = NR + 1
100 IF R = NR + NP THEN PRINT "SEI SFORTUNATO! HAI SCELTO LA MATTA" : GOTO 120
110 PRINT "SEI FORTUNATO! NON HAI TROVATO LA MATTA" : GOTO 30
120 END
```

■ ON... GOSUB permette di usare gruppi di SUBROUTINE

L'istruzione ON . . . . . GOBUS ha un uso molto simile a quello dell'istruzione ON . . . . . GOTO.



Per controllare i vostri progressi, vediamo adesso un programma che produce un orario. La cosa più importante da notare in questo programma è che alla riga 180 vengono chiamate delle subroutine, malgrado nella riga 90 si faccia un salto alle righe da 170 a 190 delle subroutine.

Ciò dimostra che le istruzioni GOSUB e ON . . . . . GOSUB possono venire usate in modo multiplo, il che è molto comodo.

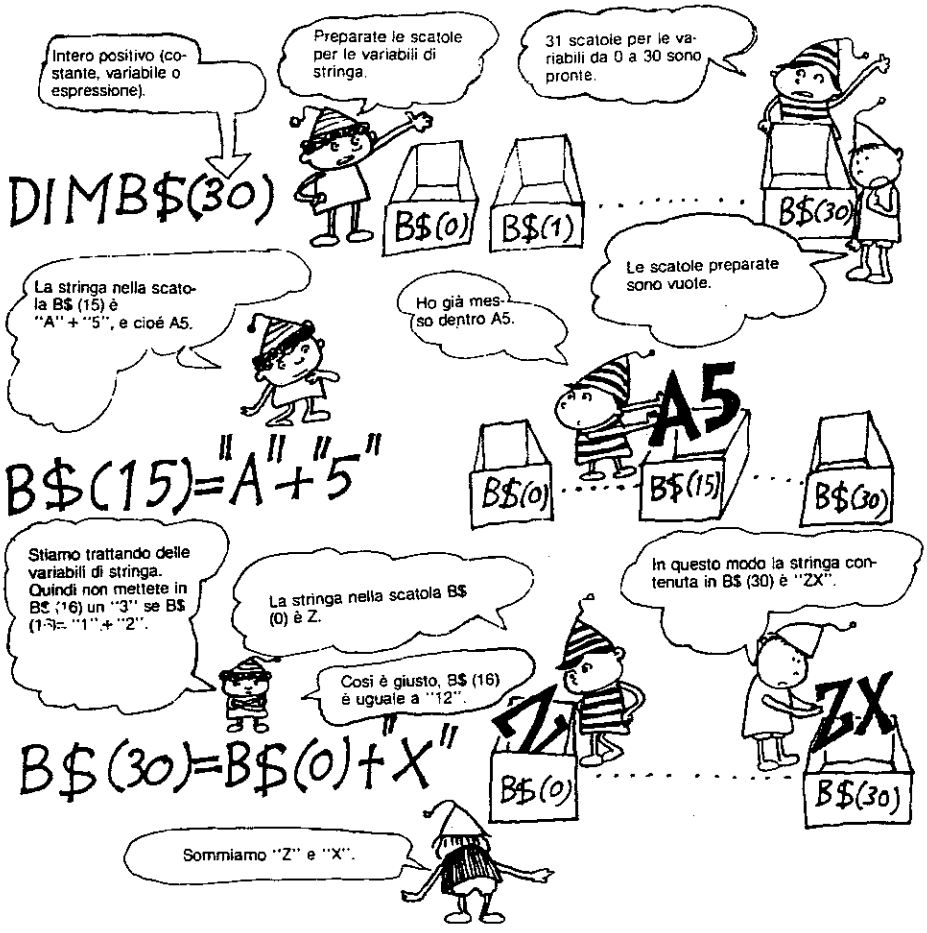
```
10  A$ = "FRANCESE " : B$ = "MATEMATICA" : C$ = "INGLESE"
20  D$ = "SCIENZA " : E$ = "MUSICA " : F$ = "ATLETICA"
30  G$ = "SOCIOLOGIA " : H$ = "ARTE " : I$ = "TECNOLOGIA"
40  J$ = "RELIGIONE " : K$ = "ECONOMIA"
50  PRINT "QUALE GIORNO DELLA SETTIMANA ? "
55  PRINT "(1-LUN, 2-MAR, 3-MER, 4-GIO, 5-VEN, 0-TUTTI)"
60  INPUT X$ : X = ASC (X$) – 47
70  FOR Y = 0 TO 3 : PRINT TAB (3 + 8 * Y) ; Y + 1;
80  NEXT Y : PRINT
90  ON X GOSUB 170, 110, 120, 130, 140, 150
100  PRINT : GOTO 50
110  PRINT "LUN:" ; A$ ; C$ ; D$ ; B$ : RETURN
120  PRINT "MAR:" ; H$ ; H$ ; E$ ; B$ : RETURN
130  PRINT "MER:" ; A$ ; C$ ; J$ ; K$ : RETURN
140  PRINT "GIO:" ; D$ ; A$ ; E$ ; F$ : RETURN
150  PRINT "VEN:" ; A$ ; D$ ; I$ ; G$ : RETURN
170  FOR Y = TO 5
180  ON Y GOSUB 110, 120, 130, 140, 150
190  PRINT : NEXT Y
200  RETURN
```

Matrice primaria, forza di 100 uomini

Prendiamo in considerazione l'assegnazione del valore a 100 voci. Usando i nomi di variabili A1, A2, ecc. si può fare così:

```
10  A1 = 5
20  A2 = 30
30  A3 = 12
. . . . . .
```

Un momento. Se dobbiamo farlo per 100 righe, questo è davvero un lavoraccio! Per eseguire questo tipo di lavoro, si può usare una matrice primaria, che è un nuovo tipo di variabile, molto utile per semplificare la scrittura dei programmi. Vediamo che cosa è una matrice primaria.



A questo punto avrete capito che cosa è una matrice primaria, non è vero?

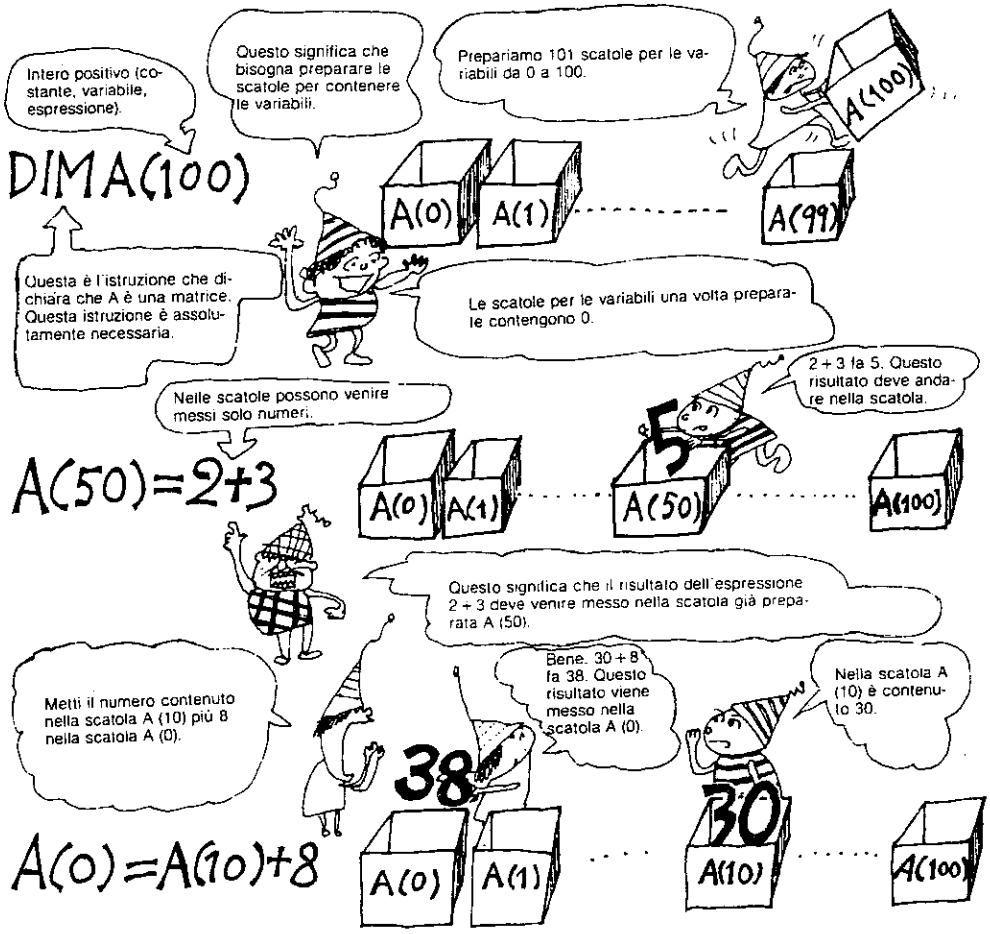
Usando una matrice primaria, il programma può venire scritto nel modo seguente:

```
10  DIM A (100)
20  FOR J = 1 TO 100
30  READ A (J)
40  NEXT J
50  DATA 5, 30, 12, . . . . . .
```

Come potete vedere il programma è abbastanza corto. Come si vede dall'esempio, le variabili sotto forma di matrice possono venire usate con un indice tra parentesi, come avviene per A (J), in cui la variabile J è l'indice. Questa è una delle caratteristiche principali delle matrici primarie.

Matrici, sempre a disposizione delle variabili di stringa

Dato che le matrici possono venire usate per le variabli numeriche, ci dovrebbe essere un tipo di matrice anche per le variabili di stringa. Eccovi una descrizione di come si presentano le matrici primarie per le variabili di stringa.



Scriviamo un semplice programma. Dateci un'occhiata. Tenere le variabili di stringa sotto forma di matrici elimina la fatica di scrivere molto ogni volta che esse vengono usate. Il programma stesso è pulito e semplice.

```
10  DIM A$ (2), B$ (2), C$ (2)
20  FOR J = 1 TO 2 : READ A$ (J), B$ (J)
30  C$ (J) = A$ (J) + " " + B$ (J)
40  PRINT A$ (J), B$ (J), C$ (J)
50  NEXT J
60  END
70  DATA GIOVANE, RAGAZZA, BIANCA, ROSA

RUN
GIOVANE     RAGAZZA     GIOVANE
BIANCA      ROSA        BIANCA
Ready
```

Matrici, abilissime a generare i File (?)

Alcuni insegnanti dicono che fare esami non è difficile, ma che invece la cosa veramente difficile è mettere in ordine i risultati degli esami. Se le cose stanno così, alcuni studenti sostengono che si dovrebbe smettere di fare esami. Un ottimo metodo è a disposizione degli insegnanti che devono sottoporre gli studenti agli esami.

L'uso delle matrici li aiuta a risolvere i loro problemi. Qui sotto sono riportati alcuni numeri di riconoscimento di studenti e i loro rispettivi voti in matematica.

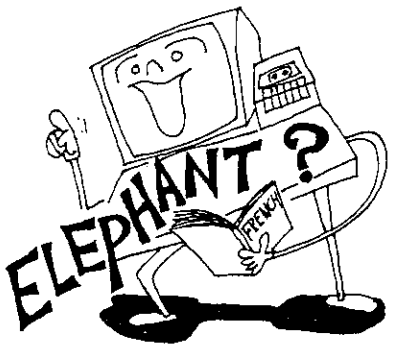
Numero dello studente	20	15	12	40	23	16	31	45	26	11
Voto	75	51	28	56	100	81	60	43	66	48

Scriviamo un programma che organizza gli studenti in ordine di merito scolastico.

```
10  DIM A (10), B (10)
20  FOR J = 1 TO 10
30  READ A (J), B (J) : NEXT
40  FOR K = 1 TO 9 : M = 0
50  FOR J = K TO 10
60  IF B (J) < = M THEN 80
70  M = B (J) : L = J
80  NEXT J
90  B (L) = B (K) : B (K) = M
100  A1 = A (L) : A (L) = A (K) : A (K) = A1
120  NEXT K
130  PRINT " ■ "
140  PRINT "ORDINE DI MERITO (MATEMATICA)"
150  PRINT
160  PRINT "STUDENTE NO." ;
170  PRINT TAB (14) ; "VOTO"
180  FOR J = 1 TO 10
190  PRINT A (J) ; TAB (14) ; B (J) : NEXT J
200  END
210  DATA 20, 75, 15, 51, 12, 28, 40, 56, 23, 100
220  DATA 16, 82, 31, 60, 45, 43, 26, 66, 11, 48
RUN
ORDINE DI MERITO (MATEMATICA)

STUDENTE NO.        VOTO
    23               100
    16               81
    20               75
    26               66
    31               60
    40               56
    15               51
    11               48
    45               43
    12               28
Ready
```

La sfida dello studio del francese



Eravamo abituati a studiare il francese imparando le parole dal libri di studio. Mediante l'uso dell'elaboratore abbiamo a disposizione dei libri di studio molto più intelligenti e più semplici. Le parole francesi ed il loro significato di parole francesi prese dal file e l'altro chiedendo di tradurre in francese delle parole italiane. Nel programma, la matrice primaria di stringhe viene usata come file che contiene le parole francesi ed il loro significato. Eseguendo il programma che segue, potrete mettere alla prova la vostra conoscienza del francese, rispondendo alle diverse domande che l'elaboratore vi pone.


```
10  DIM A$ (10), B$ (10), C$ (10)
20  FOR J = 1 TO 10
30  READ A$ (J), B$ (J)
40  C$ (J) = A$ (J) + B$ (J)
50  NEXT J
60  K = INT (10 * RND (1) ) + 1
70  PRINT " ◙ QUALE 'E` IL SIGNIFICATO DELLA PAROLA?"
80  PRINT A$ (K),
90  INPUT X$
100  AX$ = A$ (K) + X$
110  IF C$ (K) = AX$ THEN PRINT "O.K. !! " : FOR M = 1 TO 3000 : NEXT : GOTO 150
120  PRINT "WRONG" : FOR M = 1 TO 1000 : NEXT M
130  PRINT " ⬆ " ; SPACE$ (10) : PRINT " ⬆ ⬆ " ; TAB (12) ; SPACE$ (25)
140  PRINT " ■ " : GOTO 80
150  K = INT (10 * RND (1) ) + 1
160  PRINT " ◙ TRADUCETE IN FRANCESE"
170  PRINT B$ (K),
180  INPUT Y$
190  YB$ = Y$ + B$ (K)
200  IF C$ (K) = YB$ THEN PRINT "O.K.!!" : FOR M = 1 TO 3000 : NEXT M : GOTO 60
210  PRINT "SBAGLIATO" : FOR M = 1 TO 1000 : NEXT M
220  PRINT " ⬆ " ; SPACE$ (10) : PRINT " ⬆ ⬆ " ; TAB (12); SPACE$ (25)
230  PRINT " ■ " : GOTO 170
240  END
250  DATA CHAT, GATTO, PORTE, PORTA, MAISON, CASA CHIEN
260  DATA CANE, CANARD, ANITRA, POISSON, PESCE, MAIN, MANO
270  DATA FENETRE, FINESTRA, FILLETTE, RAGAZZA, FEMME
280  DATA MOGLIE
RUN
QUAL'E` IL SIGNIFICATO DELLA PAROLA?
POISSON
```

In questo caso, alla domanda sul significato di "POISSON" si risponde immettendo il suo significato in italiano. Se avete risposto in modo giusto, sullo schermo compare OK!! Se date una qualsiasi risposta errata, viene visualizzato un messaggio di errore. Vi è poi il modo inverso, in cui dovrete rispondere "POISSON" quando vi viene richiesta la traduzione.

Matrici secondarie, sono più pptenti

Diamo un'occhiata a questa tabella, qui sotto a destra, che rappresenta un miglioramento rispetto alla tabella dei risultati degli esami (qui sotto a sinistra) per le materie di matematica, Inglese e Francese di tre studenti.

Nome / Materia	John	Peter	Paul
Matematica	92	75	72
Inglese	70	94	78
Francese	65	60	95

	Nome	John	Peter	Paul
Materia	M \ N	1	2	3
Matematica	1	A (1, 1)	A (1, 2)	A (1, 3)
Inglese	2	A (2, 1)	A (2, 2)	A (2, 3)
Francese	3	A (3, 1)	A (3, 2)	A (3, 3)

M = 1 . . . Matematica
M = 2 . . . Inglese
M = 3 . . . Francese

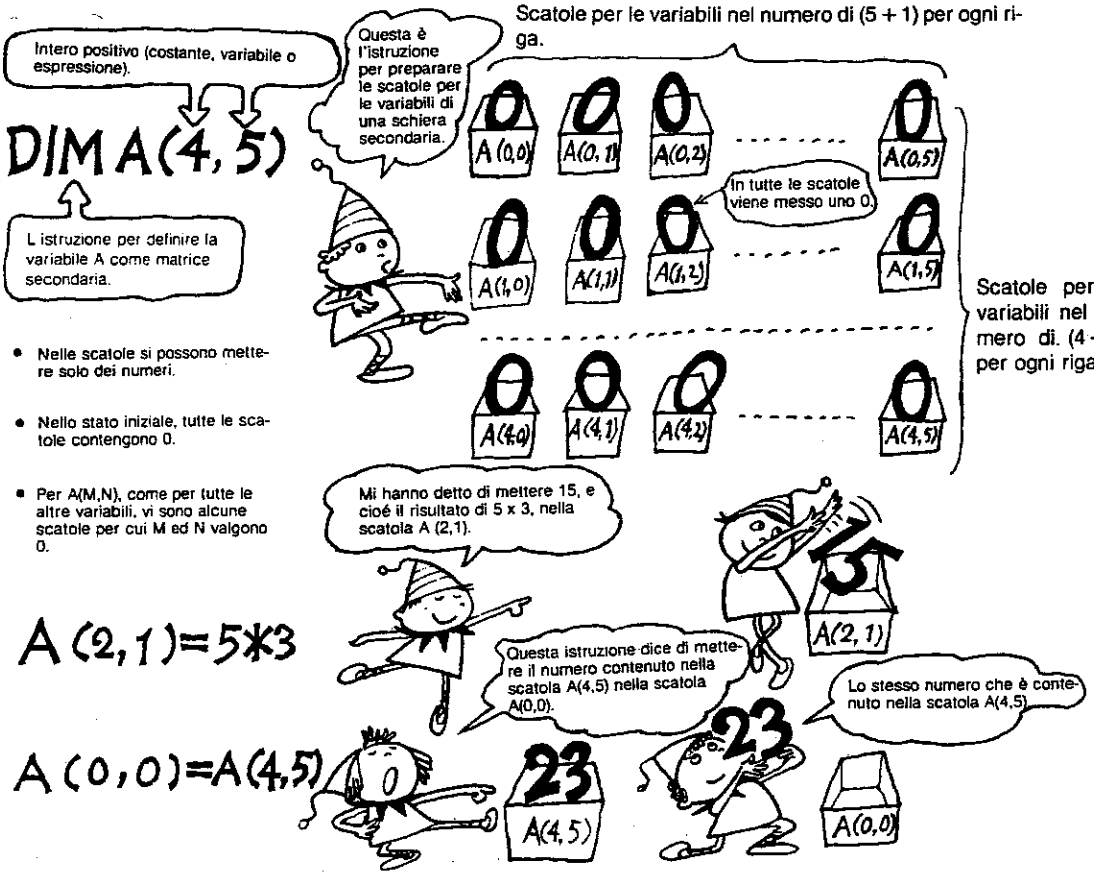
N = 1 . . . John
N = 2 . . . Peter
N = 3 . . . Paul

Nella tabella di destra, la materia, lo studente e il voto vengono rispettivamente espressi come M (1-3), N (1-3) e A (M, N). Questo può essere molto comodo, come si vede dall'esempio che segue:

A (2, 3) . . . M = 2 significa Inglese, N = 3 significa Paul . . . voto in Inglese di Paul

Semplicissimo! Basta scrivere A (2, 3) per dare una descrizione chiara del voto in Inglese di Paul. M e N in A (M, N) rappresentano due voci separate. Scrivere A (M, N) usando due voci significa usare una matrice secondaria. La matrice primaria descritta prima ha solo una voce.

Adesso fate attenzione a come questa matrice secondaria può venire usata per scrivere un programma per l'elaboratore.



Che ne dici di una tavola pitagorica?

Se i vostri ragazzi devono studiare le moltiplicazioni, l'elaboratore può aiutarli nello studio. Si può ge tavola pitagorica che può venire usata ogni volta ve ne sia bisogno. Nel programma viene usata una matrice del tipo A (M, N). In altre parole, ad A (M, N) viene assegnato il valore di M x N.

```
10  DIM A (9, 9)
20  PRINT " ■ "
30  FOR J = 1 TO 79
40  SET J, 6 : NEXT J
50  FOR J = 3 TO 27
60  SET 5, J : NEXT J
70  PRINT " ■ MOLTIPLICAZIONE"
80  PRINT
90  FOR J = 1 TO 9
100  PRINT TAB (4 * (J – 1 ) + 4) ; J ;
110  NEXT J : PRINT : PRINT
120  FOR M = 1 TO 9
130  PRINT M ;
140  FOR N = 1 TO 9
150  A (M, N) = M * N
160  PRINT TAB (4 * (N – 1) + 4) ;
170  PRINT A (M, N) ;
180  NEXT N
190  PRINT
200  NEXT M
210  END
RUN
MOLTIPLICAZIONE
```

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

```
Ready
```

In questo programma la matrice secondaria usa molti loop multipli del tipo FOR . . . . . NEXT. Vi si quindi di cercare di capire bene il significato degli indici nelle matrici secondarie. Come appare chiaro dalle m qui abbiamo una matrice 9 x 9. Le matrici secondarie hanno un ruolo importantissimo nell'elaborazione di n elementi secondari.

Random (RND) è il numero uscito a caso

Lo sapevate che i lanciatori professionisti di baseball usano una tavola di numeri usciti a caso (Random, RND)? La sequenza dei numeri usciti a caso nella tavola dei numeri casuali viene chiamata una "progressione", ed in essa ogni numero ha le stesse probabilità di uscire e il modo con cui questi numeri escono non ha alcuna regola (e viene chiamato "a caso", ovvero Random). Tali numeri vengono chiamati "numeri a caso" (numeri RND).

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, ...

In questa progressione c'è la stessa probabilità per cui ogni numero esca. Il modo però secondo cui sono usciti non è casuale. Perciò questi non sono numeri a caso.

0, 1, 1, 1, 8, 7, 3, 9, 1, 6, 1, 2, 1, 1, 5, ...

Questa progressione invece è abbastanza casuale nella sua generazione. Il ritmo però con cui esce il numero 1 è molto alto. Anche questi dunque non sono numeri che possono uscire a caso.

Perché non provare a creare una sequenza di numeri a caso? Probabilmente vi sarete resi conto che non si tratta di un lavoro facile. Non preoccupatevene: il vostro elaboratore svolge il lavoro di generare numeri a caso. Propriò così, basta usare il comando RND ( ). Tutto ciò che vi resta da fare è mettere tra le parentesi il vostro numero intero positivo preferito. Andrà bene qualsiasi numero intero positivo.

```
10  FOR R = 1 TO 3
20  PRINT RND (1) ;
30  NEXT R
40  END
RUN
 0.38079483  0.75828109  0.44046507
Ready
```

I risultati non sono sempre gli stessi, in quanto questi sono numeri a caso.

Come risulta evidente dal risultato, i valori generati dal comando RND sono soggetti alla condizione:

$0 < \text{RND}(1) < 1$ ⟸ Attenzione al fatto che 0 e 1 non sono compresi.

Per trovare la media di 1000 numeri a caso, eseguire quanto segue:

```
10  FOR J = 1 TO 1000
20  R = RND (1) : S = S + R
30  NEXT J : RG = S/1000
40  PRINT RG
50  END
RUN
 0.4980582
Ready
```

Questa media è un valore molto vicino a 0,5. Questo fatto è la dimostrazione che si tratta di numeri a caso.

Lancio dei dadi con la funzione RND

Provate a lanciare un dado. Naturalmente si presenterà una delle faccie con un valore da 1 a 6. Il divertimento nel giocare a dadi sta nel fatto che si può presentare una qualsiasi faccia. In altre parole, vengono generati numeri a caso da 1 a 6.

L'elaboratore è dotato della funzione RND che permette di generare numeri a caso. Il problema è se, usando questa funzione si può simulare un dado. A dire la verità è possibile. Come si fare?

Prima di tutto moltiplichiamo la funzione RND per 6.

$$0 < \text{RND}(1) < 1 \xrightarrow{*6} 0 < 6 * \text{RND}(1) < 6$$

Probabilmente vi ricordate il comando INT. Esso viene usato per arrotondare il valore di 6 x RND (1).

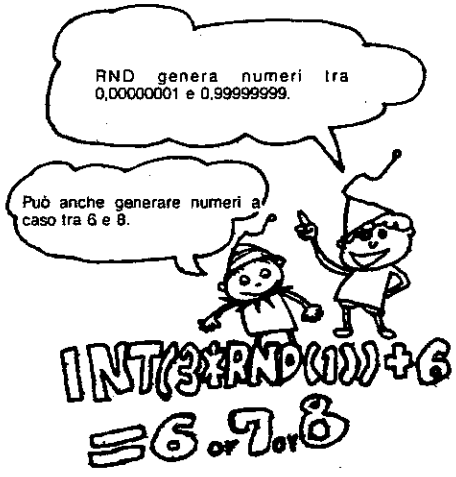
INT (6 * RND (1)) = 0, 1, 2, 3, 4, 5

In questo modo, abbiamo generato a caso degli interi tra 0 e 5. Il dado richiede però i numeri da 1 a 6 e quindi il nostro risultato non basta. Come vi sarete resi conto, basta aggiungere 1 ai risultati precedenti.

INT (6 * RND (1)) + 1 = 1, 2, 3, 4, 5, 6

A questo punto, il dado è pronto. Usando questo dado, controlliamo la frequenza con cui ogni faccia vien fuori.

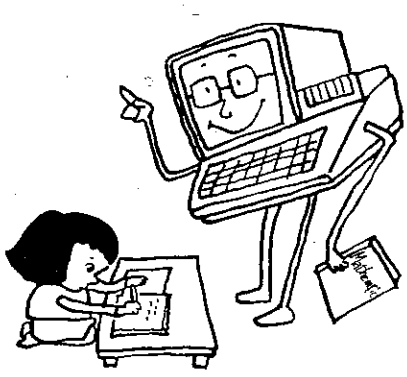
```
10  PRINT "QUANTE VOLTE VUOI LANCIARE IL DADO?"
20  INPUT N
30  FOR J = 1 TO N
40  R = INT (6 * RND (1) ) + 1
50  IF R = 1 THEN N1 = N1 + 1
60  IF R = 2 THEN N2 = N2 + 1
70  IF R = 3 THEN N3 = N3 + 1
80  IF R = 4 THEN N4 = N4 + 1
90  IF R = 5 THEN N5 = N5 + 1
100  IF R = 6 THEN N6 = N6 + 1
110  NEXT J
120  P1 = N1/N : P2 = N2/N : P3 = N3/N
130  P4 = N4/N : P5 = N5/N : P6 = N6/N
140  PRINT P1, P2, P3, P4, P5, P6
150  END
RUN
QUANTE VOLTE VUOI LANCIARE IL DADO?
? 5000
  0.1702        0.1654        0.1628        0.1626
  0.169         0.17
Ready
```



Cosa ne pensate del risultato? Ogni faccia ha praticamente la stessa probabilita di uscire: Dal punto di vista matematico, la situazione teorica è che ogni faccia esca una volta, se il dado viene tirato sei volte. Di conseguenza, i numeri ottenuti dovrebbero essere esattamente nel rapporto 1 a 6.

Un Professore di matematica privato

Vediamo di procurare un insegnante di matematica al vostro fratellino o la vostra sorellina. Tanto per cominciare, esercitiamoci a sommare e moltiplicare i numeri tra 1 e 9. Per esercitarci sulle somme.



```
10  A = INT (RND (1) * 9 + 1)
20  B = INT (RND (1) * 9 + 1)
30  PRINT "   " ; A : PRINT "+" ; B
40  PRINT "  □□□□□□  " : INPUT C
50  IF C = A + B THEN 80
60  PRINT "PENSA BENE UNA VOLTA DI PIU' "
70  GOTO 30
80  PRINT "GIUSTO! BEN FATTO" : GOTO 10
```

Usando il generatore di numeri a caso nelle righe di programma 10 e 20, sostituiamo due numeri tra 1 e 9 alle variabili A e B. La somma dei due numeri viene ottenuta visualizzando "?" Nella riga 50 del programma viene formulato il giudizio sulla risposta, se esatta o no. Per fare degli esercizi sulle moltiplicazioni, basta sostituire il segno "+" con il segno " * " nelle righe di programma 30 e 50.

Se si desidera usare i numeri da 1 a 99, le righe 10 e 20 devono venire modificate come segue:

```
10  A = INT (RND (1) * 99 + 1)
20  B = INT (RND (1) * 99 + 1)
```

Sembra tutto giusto. In realtà però i numeri ad una cifra non sono incolonnati con quelli a due cifre. Bisogna quandi fare qualcosa per incolonnare i numeri. In altre parole, con l'uso della possibilità di manipolare le stringhe, controlliamo il numero di cifre per scrivere bene l'equazione. Per incolonnare le cifre, modificare il programma come segue.

```
25  A$ = STR$ (A) : B$ = STR$ (B)
30  PRINT TAB (5 – LEN (A$) ) ; A
35  PRINT "+" ; TAB (5 – LEN (B$) ) ; B
40  PRINT "  □□□□□□  " : INPUT C
```

Alla riga 25, l'elaboratore converte due cifre in una stringa e visualizza i rispettivi numeri con le cifre incolonnate nelle righe 30 e 40. Notare l'uso di TAB ( ) e LEN ( ).

Sospensione per il tè.

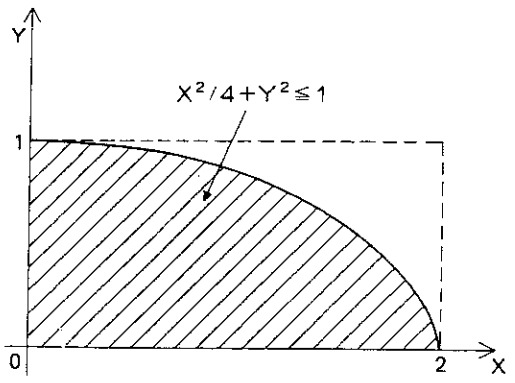
È arrivato il momento di riposarsi. Immettere il programma riportato a destra e guardare gli schemi casuali che vengono generati.

```
100  ? "  ■  " ;
110  ? "  ■  " : W = 1
120  FOR X = 1 TO 7 : FOR Y = 1 TO 5 : FOR Z = 1 TO 5
130  ? TAB ((X – 1)  *  5) ; A$ ; : NEXT Z : ? : NEXT Y
140  A$ = CHR$ (INT (RND (1)  * 223 + 33))
150  ? "  ■■■■■  " ; : NEXT X : ? "  ■■■■  " : W = W + 1
160  IF W > 4 THEN 110
170  GOTO 120
```

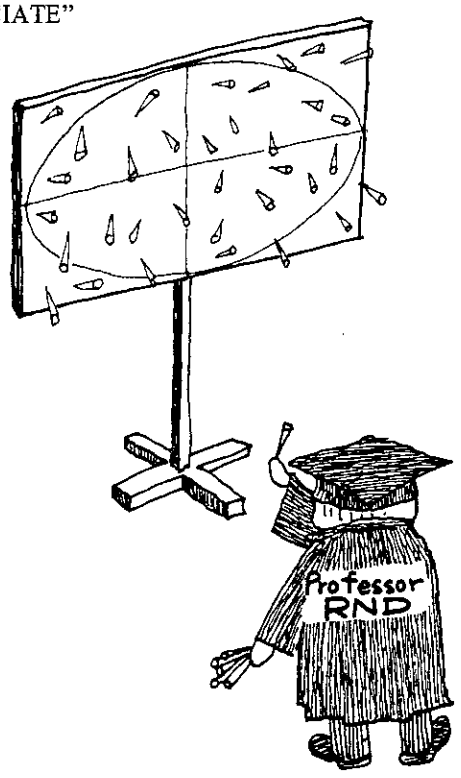
Calcoliamo le aree con la Probabilità

L'uso dei numeri a caso rende possibile il calcolo degli integrali. Viene usato un calcolo di probabilità chiamato il metodo di integrazione numerica di Montecarlo. Proviamo a determinare l'area di una ellisse (integrale definito).

Per semplificare il calcolo, prendiamo in cosiderazione solo 1/4 dell'ellisse. L'area tratteggiata nella figura a destra, rappresenta l'interno di una ellisse definita dalla curva $X^2/4 + Y^2 = 1$. Scagliamo a caso delle frecce sul rettangolo definito dalla linea tratteggiata. Se lanciamo un numero di frecce sufficientemente grande, il rapporto tra il numero di volte in cui si colpisce l'area ellittica ed il numero totale dei lanci è molto vicino al rapporto tra l'area ellittica e quella rettangolare. La funzione RND può sostituirsi all'azione della freccia. Quello che segue è il programma per il calcolo dell'area ellittica.




```
10  PRINT "QUANTE FRECCE DEVONO ESSERE LANCIATE"
20  INPUT N
30  PRINT "CALCOLO IN CORSO"
40  FOR J = 1 TO N
50  X = 2 * RND (1) : Y = RND (1)
60  IF X * X/4 + Y * Y <= 1 THEN ND = ND + 1
70  NEXT J
80  S = 4 * (2 * ND/N)
90  PRINT "AREA DELL'ELLISSE S =" ; S
100 END
```




```
RUN
QUANTE FRECCE DEVONO ESSERE LANCIATE
?  100
CALCOLO IN CORSO
AREA DELL'ELLISSE S = 6.4
Ready

RUN
QUANTE FRECCE DEVONO ESSERE LANCIATE?
?  1000
CALCOLO IN CORSO
AREA DELL'ELLISSE S = 6.336
Ready
```

L'effettiva area dell'ellisse è 6,28. Con 100 lanci di frecia, il risultato è 6,4, molto vicino all'area effettiva. Con il lancio di 1000 frecce, il risultato è ancora più vicino all'area effettiva. Non trovate che è interessante determinare un'area mediante i numeri a caso? E, fatto ancora piú strano, quanto piú i numeri sono a caso, cioè escono senza regole, tanto piú accurata è la misura dell'area.

Al casinò, soldi con la Slot Machine

Qui a Las Vegas, l'eccitante città dove si può sognare di divertirsi e di fare fortuna in fretta, Michael, un appassionato giocatore, sta giocando con una Slot machine. Dopo aver introdotto una moneta da 10 cents, abbassa la leva. Nelle tre finestrelle compaiono Picche, Quadri, Fiori, Cuori o Dollari e monete piovono fuori a seconda delle combinazioni degli schemi. Se non esce nessuna combinazione vincente, si perde la moneta da 10 cents. Vediamo di generare un programma che esegua le stesse operazioni.


```
10  PRINT "  "  : D = 0 : H$ = "  "
20  DIM N$ (5), A (3)
30  FOR X = 1 TO 10 : H$ = H$ + "  " : NEXT X
40  FOR X = 1 TO 5 : READ N$ (X) : NEXT X
50  PRINT H$ ; "  $ = " ; SPACE$ (5) ; "" ; D
60  PRINT H$ ; "  " : PRINT SPACE$ (38) ; "  "
70  INPUT "FORZA PER ABBASSARE LA LEVA" ; NN
80  FOR Y = 0 TO NN : RR = RND (1) : NEXT Y
90  FOR X = 1 TO 3 : A (X) = INT (5 * RND (1) ) + 1 : NEXT X
100 PRINT "  " ;
110 FOR X = 1 TO 3 : PRINT N$ (A (X) ), : NEXT X : PRINT
120 IF A (1) <> A (2) THEN 170
130 IF A (2) <> A (3) THEN 160
140 IF A (1) = 5 THEN PRINT H$ ; "EUREKA!! $1.00" ; SPACE$ (15) : D = D + 1.00 : GOTO 50
150 PRINT H$ ; "TUTTO UGUALE! $0.25" ; SPACE$ (10) : D = D + 0.25 : GOTO 50
160 PRINT H$ ; "HAI VINTO 10$" ; SPACE$ (15) : D = D + 0.1 : GOTO 50
170 IF A (1) = 1 THEN PRINT H$ ; "HAI VINTO S.05" ; SPACE$ (15) : D = D + 0.05 : GOTO 50
180 PRINT H$ ; "ALTRI 10 CENTS PER FAVORE" : D = D – 0.1 : GOTO 50
190 DATA  ♠♠♠,♦♦♦,♥♥♥,♣♣♣, $ $ $
```

Avete vinto? Come Michael, anche voi siete degli appassionati giocatori e quindi perché non provate a generare la forma di una Slot machine e ad introdurla nel programma precedente?

Esercizi con l'uso della funzione RND

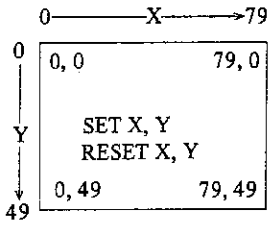
Facciamo degli esercizi per il calcolo dell'area di un triangolo usando la funzione RND. In questo caso la funzione RND è il vostro professore. La funzione, usando i numeri a caso, vi fornisce i valori del lati di un triangolo e voi dovete calcolare l'area del triangolo. La funzione RND è il vostro miglior compagno di studi.


```
10  DIM A (3), L$ (4)
20  FOR J = 1 TO 4
30  READ L$ (J) : NEXT J
40  FOR J = 1 TO 3
50  A (J) = INT (20 * RND (1)) + 1
60  NEXT J
70  IF A (1) >= A (2) + A (3) GOTO 40
80  IF A (2) >= A (1) + A (3) GOTO 40
90  IF A (3) >= A (1) + A (2) GOTO 40
100  W = (A (1) + A (2) + (3))/2
110  T = W : FOR J = 1 TO 3
120  T = T * (W – A (J)) : NEXT J
130  SS = SQR (T) : S = INT (SS)
140  IF SS – S >= 0.5 THEN S = S + 1
150  PRINT " ■■■■ "
160  PRINT "CALCOLA L'AREA DEL SEGUENTE TRIANGOLO"
170  PRINT "MA ARROTONDA IL RISULTATO ALL'UNITA` PIU` VICINA"
180  PRINT
190  PRINT TAB (8) ; "A"
200  PRINT TAB (8) ; " ▨▧ "                    ; TAB (15) ; L$ (1) ; A (1)
210  PRINT TAB (7) ; " ▨     ▧ "               ; TAB (15) ; L$ (2) ; A (2)
220  PRINT TAB (6) ; " ▨         ▧ "           ; TAB (15) ; L$ (3) ; A (3)
230  PRINT TAB (5) ; " ▨             ▧ "
240  PRINT TAB (3) ; "B ▨                 ▧ C"
250  PRINT TAB (4) ; " □□□□□□□□□□ "
260  PRINT " ■■■ "
270  PRINT TAB (3) ; L$ (4) ;
280  INPUT Y
290  IF Y = S THEN PRINT SPACE$ (7) ; "O.K.!!" : FOR J = 1 TO 3000 : NEXT J : GOTO 40
300  IF Y < S THEN PRINT SPACE$ (5) ; "TROPPO PICCOLA" : GOTO 320
310  PRINT SPACE$ (4), "TROPPO GRANDE"
320  PRINT " ■■ " ;
330  PRINT TAB (26) ; SPACE$ (13) : PRINT " ■ " ;
340  GOTO 270
350  DATA LUNGHEZZA LATO AB = , LUNGHEZZA LATO BC =
360  DATA LUNGHEZZA LATO CA = , L'AREA S TRIANGOLO ABC
```

SET o RESET?

Usando il modo diretto, immettere la seguente riga:

SET 79, 49 [CR]

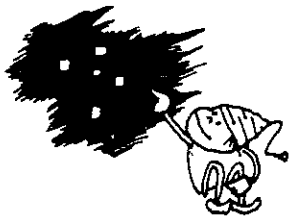


Schermo televisivo

Cosa succede? Si è accesa la luce sull'angolo inferiore destro dello schermo. Allora

RESET 79, 49, [CR]

Si è spenta la luce, non è vero? Come avete potuto vedere, le istruzioni SET e RESET servono per accendere e spegnere un punto luminoso, come è mostrato nella figura a destra.



Lampeggiamento nell'angolo inferiore destro.

```
10 PRINT " ■ "
20 X = 79 : Y = 49
30 SET X, Y ← acceso
40 RESET X, Y ← spento
50 GOTO 30
```

Per scrivere un rettangolo delle dimensioni dello schermo.

(ES. 2)

```
10 PRINT " ■ "
20 FOR X = 0 TO 79
30 SET X, 0
40 SET X, 49
50 NEXT X
60 FOR Y = 0 TO 49
70 SET 0, Y
80 SET 79, Y
90 NEXT Y
99 GOTO 99
```

Vediamo qualcosa di nuovo!

```
10 PRINT " ■ "
20 FOR Z = 1 TO 99
30 SET Z, Z
40 NEXT Z
50 GOTO 50
```

Proprio così, in effetti, quando si supera l'80 nella direzione delle X o il 50 nella direzione delle Y.

```
X ← X – 80
Y ← Y – 50
```

Il calcolo viene fatto automaticamente.

Schermo tutto bianco

(ES. 1)

```
10 PRINT " ■ "
20 FOR X = 0 TO 79
30 FOR Y = 0 TO 49
40 SET X, Y
50 NEXT Y, X
60 GOTO 60
```

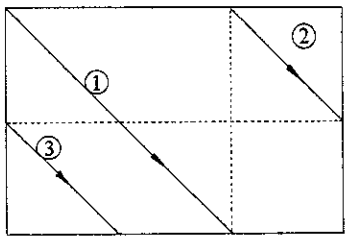
Per scrivere una riga inclinata

(ES. 3)

```
10 PRINT " ■ "
20 PRINT "LUNGHEZZA DELLA DIAGONALE" ;
30 INPUT Y
40 X = SQR (Y * Y/2)
50 FOR A = 1 TO X
60 SET A, 49 – A
70 NEXT A
80 GOTO 80
```

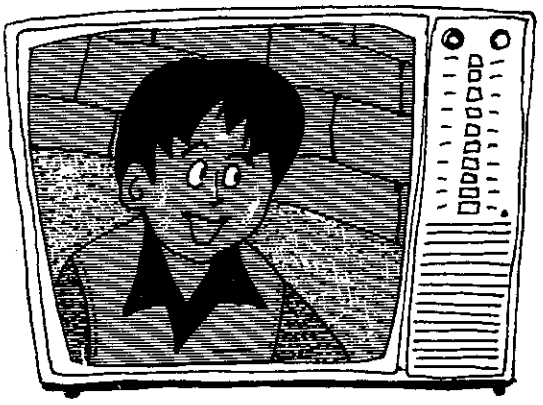
È applicato il Teorema di Pitagora.

Schermo dopo l'esecuzione del programma.



Introduzione ai principi della TV

```
10 PRINT " ■ " : Y = 1
20 FOR X = 0 TO 79
30 SET X, Y
40 RESET X, Y – 1
50 NEXT X
60 Y = Y + 1
70 GOTO 20
```



Avete visto la riga bianca che avanza orizzontalmente e salta di una riga alla volta verticalmente? Questa è stata realizzata per aiutarvi a capire i principi del funzionamento della TV.

Un normale apparecchio televisivo utilizza circa 500 righe di questo tipo per creare un'immagine e vengono trasmesse circa 30 immagini al secondo.

Palla che rimbalza.....

```
10 PRINT " ■ "
20 FOR X = 0 TO 79
30 SET X, 0 : SET X, 49
40 NEXT X
50 FOR Y = 0 TO 49
60 SET 0, Y : SET 79, Y
70 NEXT Y
80 X = 79 * RND (1) : Y = 49 * RND (1)
90 A = 1 : B = 1
100 SET X, Y
110 IF X < 2 GOSUB 200
120 IF X > 78 GOSUB 200
130 IF Y < 2 GOSUB 250
140 IF Y > 48 GOSUB 250
150 RESET X, Y
160 X = X + A : Y = Y + B : GOTO 100
200 A = –A : MUSIC " + A0" : RETURN
250 B = –B : MUSIC "A0" : RETURN
```

Cosa ve ne pare? Avete capito il flusso del programma? Questo programma ha tre punti importanti. Il primo è nella riga 80, in cui, utilizzando numeri a caso, viene stabilito il punto di partenza di una palla. Il secondo sono le righe da 110 a 140, in cui viene fatto il controllo se la palla è rimbalzata sui quattro muri. Il terzo sono le righe da 200 a 250, in cui la direzione della palla viene modificata quando rimbalza sul muro.

Sasso tirato nello stagno

```
10 X = 40 : Y = 25
20 DEF FNY (Z) = SQR (R * R – Z * Z)
30 PRINT " ■ " : SET X, Y
40 R = R + 5
50 FOR Z = 0 TO R
60 T = FNY (Z)
70 SET X + Z, Y + T
80 SET X + Z, Y – T
90 SET X – Z, Y + T
100 SET X – Z, Y – T
110 NEXT Z
120 IF R <> 25 THEN 40
130 GOTO 130
```

Scenetta agreste (coniglio e volpe)

Prendiamo adesso in considerazione un modello che riguarda l'ecologia animale realizzato sul video usando l'istruzione SET. Il modello considerato è quello della lotta tra gli animali per la sopravvivenza. Questo modello viene ecologicamente rappresentato mediante un'equazione differenziale.

Può venire comunque usato come esempio di calcolo numerico eseguilo dall'elaboratore.

CONIGLI E VOLPI

Un coniglio ed una volpe vengono usati come esempio della relazione "Il debole diviene la vittima del forte" nel regno animale. In questo esempio si presume anche che la volpe viva solo di conigli, mangiatori di erba. Secondo il modello ecologico, il numero di conigli e di volpi, chiamati rispettivamente X e Y, varia in accordo con la seguente equazione differenziale:

$$\frac{dx}{dt} = AX - BXY \quad \text{. . . . . Variazione nel numero di conigli}$$

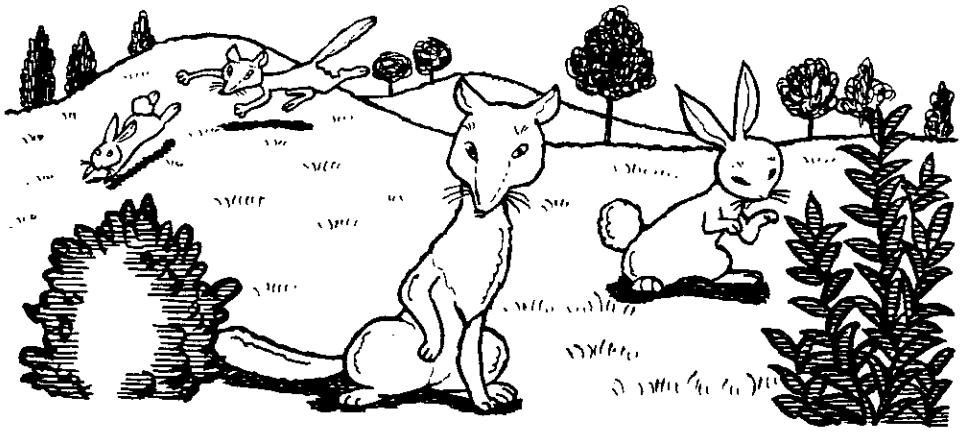
$$\frac{dy}{dt} = CXY - DY \quad \text{. . . . . Variazione nel numero delle volpi}$$

A, B, C e D sono delle costanti. La voce XY rappresenta la relazione "Il debole diviene la vittima del forte". Prendendo in considerazione la variazione nel numero di conigli, la seconda voce sarebbe eguale zero se le volpi non esistessero. I conigli continuerebbero ad aumentare in progressione geometrica. Di fatto però le volpi esistono ed aumentano di numero in modo da fermare l'aumento dei conigli (2^a voce). Prendendo in considerazione la variazione nel numero delle volpi, la prima voce sarebbe zero se non esistessero i conigli, e le volpi morirebbero una dopo l'altra. Dato che i conigli esistono, il numero delle volpi aumenta conformemente al valore della prima voce CXY. Questo è il modo in cui conigli e volpi sono legati gli uni alle altre.

Per prendere in considerazione questa equazione differenziale numericamente usando l'elaboratore, vengono usati dei valori approssimati basati sul metodo delle tangenti. Tale metodo permette di approssimare la derivazione rispetto al tempo mediante il rapporto tra l'incremento delle variabili e l'incremento del tempo, in modo che sostitutendo DT a dt, DX a dx e DY a dy, l'equazione differenziale diventa un'equazione algebrica:

$$DX = (AX - BXY)\,DT$$
$$DY = (CXY - DY)\,DT$$

Nella pagina seguente è riportato un esempio di programma basato su questo metodo.



■ Il segreto di un grafico di forma ovale

```
10  REM CONIGLIO e VOLPE
20  READ A, B, C, D, DT, TT, K
30  DATA 4, 2, 1, 3, 0.01, 2, 30
40  FOR H = 1 TO 20 : H$ = H$ + " ⬇ " : NEXT H
50  PRINT " ▣⬇⬇⬇⬇⬇⬇⬇➡◥◧◥ " ; SPACE$ (3) ; " ◥ " : PRINT " ➡➡➡◢▀▀◥ "
60  PRINT TAB (4) ; "FOX" ; " ▣ " ; H$ ; " ⬆⬆⬆ " ; TAB (22) ; " ◧◥ "
70  PRINT TAB (22) ; " ◧◧▄◙ " : PRINT TAB (22) ; " ■◩▄◥ "
80  PRINT TAB (21) ; "RABBIT"
100 PRINT " ▣⬇ " ; H$ ; SPACE$ (38) : PRINT SPACE$ (38) ; " ▣ " ; H$
110 INPUT "NUMERO DI CONIGLI R0 =" ; X
120 INPUT "NUMERO DI VOLPI F0 =" ; Y
200 FOR J = 1 TO K
210 FOR T = 0 TO TT/K STEP DT
220 X = X + (A * X – B * X * Y) * DT
230 Y = Y + (C * X * Y – D * Y) * DT
240 NEXT T
250 SET 4 * (X + 6), 4 * (8 – Y)
260 NEXT J
270 INPUT "VUOI CERCARNE UN ALTRO? (S/N)" ; Y$
280 PRINT " ⬆ " ; SPACE$ (38) ; " ⬆ "
290 IF Y$ = "S" THEN 100
300 END
```

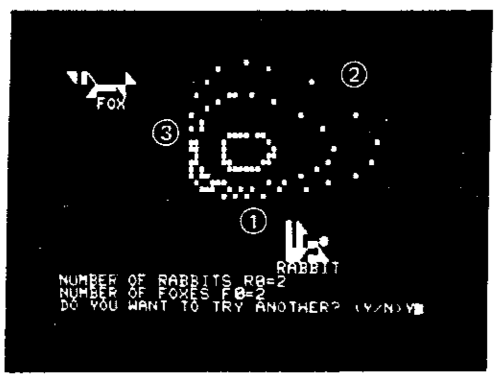
Parliamo ora del funzionamento del programma. L'elaboratore richiede il numero iniziale del conigli e delle volpi. Per esempio, immettiamo R0 = 5 e F0 = 5. Cominciando con i valori iniziali, sia i conigli che le volpi cambiano di numero nel tempo, come si può vedere sullo schermo.

Viene così disegnata una bella figura ovale. Consideriamo il contenuto della figura.

Parte 1 Poiché le volpi all'inizio sono poche, i conigli si moltiplicano rapidamente, mentre le volpi aumentano gradualmente.

Parte 2 Quando le volpi aumentano di numero, i conigli hanno raggiunto la punta massima nel loro numero e cominciano a diminuire.

Parte 3 I conigli continuano a diminuire. Così pure le volpi. Entrambi diminuiscono rapidamente.



In questo modo, il numero dei rispettivi animali ritorna al valore iniziale. Predisponendo diversi valori iniziali per i numeri di conigli e di volpi si può ottenere una grande varietà di grafici ovali diversi nelle dimensioni.

GET è utile per l'entrata da tastiera

Introduciamo ora un nuovo tipo di istruzione per effettuare immissione di dati dalla tastiera. Si chiama:

GET

Questa istruzione è particolarmente adatta per essere usata durante l'esecuzione di elaborazioni rapide usate, ad esempio, durante un gioco. Usando l'istruzione INPUT, l'elaboratore usa il carattere "?" per segnalare una richiesta, e una volta immessi i dati, si deve premere il tasto CR per spostarsi al passo successivo. Il comando GET invece procede nel modo seguente:

- Quando viene premuto un tasto mentre un programma è sotto il controllo di un comando GET, una posizione di dati viene sostituita nella variabile e si passa poi al passo successivo.
- Quando non viene premuto nessun tasto, nella variabile viene sostituito 0 o spazio in bianco a seconda se si tratta di variabile numerica o di stringa e poi si passa al passo successivo.

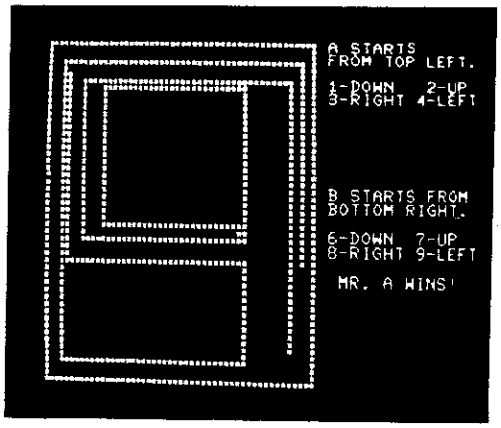
Il comando GET viene spesso usato in modo che il programma attenda prima di continuare che venga immesso un carattere numerico o alfabetico.

Prima di giocare con un gioco che contiene il comando GET, facciamo un semplice esercizio. Usando il comando GOTO, eseguire ripetutamente il comando GET. Cercare di immettere i numeri da 1 a 9 come valore della variabile Z.

```
10 GET Z
20 PRINT Z; : GOTO 10
```

Quando il tasto viene premuto il numero del tasto viene visualizzato.

Generiamo adesso un programma per un gioco di posizione, usando i comandi GET e SET. (Il programma è riportato nella pagina successiva). Questo gioco è per due persone. All'interno della cornice quadrata dello schermo, due giocatori cominciano rispettivamente dalle posizioni superiore sinistra e inferiore destra e tentano di bloccare i movimenti dell'avversario cambiando direzione mediante l'uso dei tasti, come indicato qui sotto. Il giocatore che riesce a bloccare i movimenti dell'avversario ha vinto la partita.



Signor A (Comincia dall'angolo in alto a sinistra)

Tasti	1	2	3	4
	↓	↑	→	←

Signor B (Comincia dall'angolo in basso a destra)

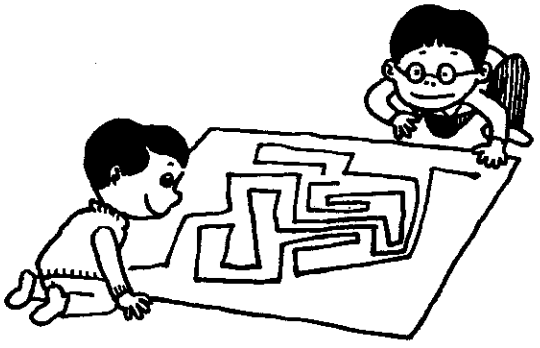
Tasti	6	7	8	9
	↓	↑	→	←

Quando l'avversario sta premendo un tasto, nessun altro tasto può essere fatto funzionare. Si deve quindi premere i tasti in alternanza.

Occhio a un gioco di posizione

Questo programma usa un certo numero di istruzioni e di comandi che voi non avete ancora imparato. Date un'occhiata a come vengono scritti. In effetti, questo programma richiede un'area di programma di quassi 13K bytes.

```
10  CLR
15  DIM Z (49, 49)
20  PRINT " ■ " ; TAB (26) ; "(A), PARTE"
25  PRINT TAB (26) ; "DA SINISTRA" : PRINT
30  PRINT " ■ " : PRINT TAB (26) ; "1 – GIU` 2 – SU"
35  PRINT TAB (26) ; "3 – DESTRA"
40  PRINT TAB (26) ; "4 – SINISTRA" : PRINT
45  PRINT : PRINT : PRINT TAB (26) ; "(B) PARTE"
47  PRINT TAB (26) ; "DA DESTRA"
50  PRINT : PRINT TAB (26) ; "6 – GIU` 7 – SU"
55  PRINT TAB (26); "8 – DESTRA"
57  PRINT TAB (26) ; "9 – SINISTRA"
60  FOR A = 0 TO 49
70  SET A, 0 : Z (A, 0) = –1
80  SET A, 49 : Z (A, 49) = –1
90  NEXT A
100  FOR A = 1 TO 49
110  SET 0, A : Z (0, A) = –1
120  SET 49, A : Z (49, A) = –1
130  NEXT A
140  X1 = 4 : Y1 = 4 : D1 = 1
150  X2 = 45 : Y2 = 45 : D2 = 7
160  GOSUB 200 : IF M = 1 GOTO 10
170  GOSUB 300 : IF M = 1 GOTO 10
180  GOTO 160
200  GET X
210  IF X = 0 THEN 260
220  IF X = 1 THEN Y1 = Y1 + 1 : GOTO 270
230  IF X = 2 THEN Y1 = Y1 – 1 : GOTO 270
240  IF X = 3 THEN X1 = Y1 + 1 : GOTO 270
250  IF X = 4 THEN X1 = X1 – 1 : GOTO 270
260  X = D1 : GOTO 220
270  D1 = X : IF Z (X1, Y1) = –1 THEN 400
280  MUSIC "+A0" : SET X1, Y1 : Z (X1, Y1) = –1
290  RETURN
300  GET X
310  IF X = 0 THEN 360
320  IF X = 6 THEN Y2 = Y2 + 1 : GOTO 370
330  IF X = 7 THEN Y2 = Y2 – 1 : GOTO 370
340  IF X = 8 THEN X2 = X2 + 1 : GOTO 370
350  IF X = 9 THEN X2 = X2 – 1 : GOTO 370
360  X = D2 : GOTO 320
370  D2 = X : IF Z (X2, Y2) = –1 THEN 450
380  MUSIC "A0" : SET X2, Y2 : Z (X2, Y2) = –1
390  RETURN
400  PRINT TAB (26) ; "■■■■ VINCE IL SIG. B"
410  MUSIC "C3DEG + CEG + CCDEG + CEG + CC" : M = 1 : RETURN
450  PRINT TAB (26) ; "■■■■ VINCE IL SIG. A"
460  MUSIC "– G3 – A – BCDE #FGAB + C + D + #D + E + F + #F + G" : M = 1
470  RETURN
```



TI$ è un orologio digitale

Un orologio digitale che dipende dal programma. Tale funzione può venire adempiuta da TI$.

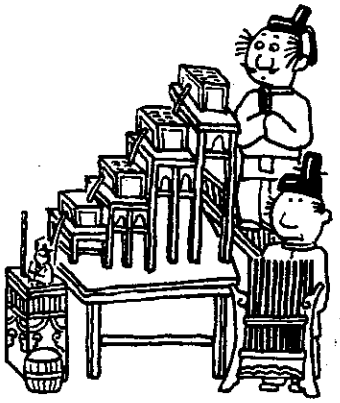
Immettere nell'elaboratore quanto segue, e premere il tasto CR.

```
PRINT TI$
```

Verrà visualizzato quanto segue, ad esempio:

```
001234
```

Che cosa significa questo numero di sei cifre che rappresenta il contenuto della variabile di stringa TI$? Esatto, rappresenta un tempo, come si vede qui sotto:



Antico orologio ad acqua Giapponese.

00 12 34 ⇨ 00 ore, 12 minuti e 34 secondi

Ore Minuti Secondi

Questo tempo non è necessariamente sincronizzato con il vostro orologio. Questo accade perché TI$ viene automaticamente predisposto a 00 ore, 00 minuti e 00 secondi quando l'elaboratore viene acceso. Visualizzate un'altra volta TI$ e potrete vedere la differenza rispetto alla prima visualizzazione. È giusto: il numero di sei cifre contenuto in TI$ cambia di momento in momento esattamente allo stesso modo in cui cambiano le cifre di un orologio digitale. Il valore di TI$ fino a questo momento rappresenta l'intervallo di tempo dal momento in cui è stato acceso l'elaboratore.

```
10  TI$ = "102634"
20  PRINT TI$
30  TI$ = "256471"
40  PRINT TI$
50  END
RUN
102634
* Error 3 in 30
Ready
```

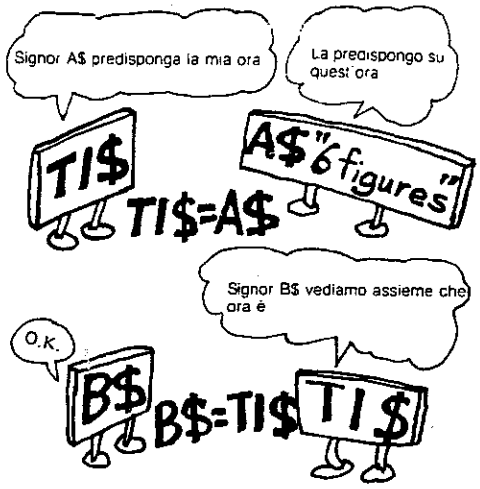
Con le istruzioni contenute nelle righe 10 e 30, TI$ può venire predisposto su di un particolare valore del tempo. Il numero compreso tra le virgolette deve però contenere non più di 6 cifre. La riga 30 indica 25 ore, 64 minuti e 71 secondi, che dovrebbe normalmente venire indicato come 2 ore, 5 minuti 3 11 secondi.

Che ora è adesso a Tokyo?

Il programma che segue mostra la relazione tra TI$ ed una normale stringa, come ad esempio A$.

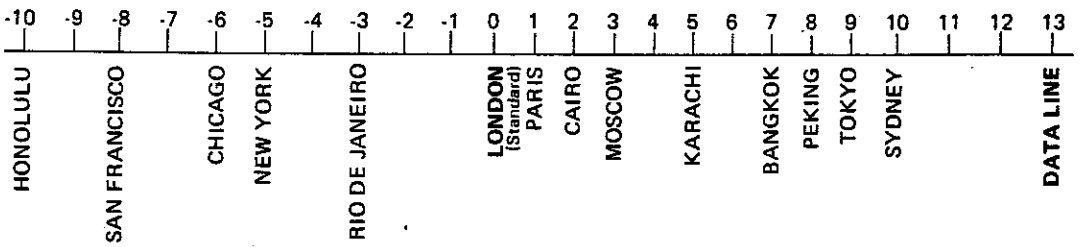
```
10  A$ = "123456"
20  TI$ = A$
30  PRINT TI$
40  FOR T = 1 TO 5000 : NEXT T
50  B$ = TI$
60  PRINT B$
70  END
RUN
123456
123501
Ready
```

TI$ registra un intervallo di 5 secondi durante l'esecuzione del programma.



Ecco un semplice orologio che vi dà l'ora in tutto il mondo. Naturalmente è un orologio digitale. Vi può dire che ora è nel paese che preferite. Eseguiamo adesso il programma che segue, in cui si usa TI$ come orologio. Tra Londra e Tokyo vi è una differenza di orario di 9 ore, se non sbagliamo. Tocca a voi generare un programma che dia l'ora di tutti gli altri paesi del mondo.

```
10  PRINT " ■ "
20  DIM C$ (10), D (10), E (10), T$ (10)
30  FOR J = 1 TO 10 : READ C$ (J), D (J) : NEXT J
40  PRINT "CHE ORE E` A LONDRA?"
50  INPUT TI$ : PRINT " ■ "
60  PRINT " ■ " : T$ (1) = TI$
70  FOR J = 1 TO 10
80  E (J) = VAL (LEFT$ (T$ (1), 2)) + D (J)
85  IF E (J) = >24 THEN E(J) = E(J)-24
90  IF E (J) < 0 THEN E (J) = 24 + E (J)
100  T$ (J) = STR$ (E (J)) + RIGHT$ (T$ (1), 4)
110  IF LEN (T$ (J)) = 5 THEN T$ (J) = "0" + T$ (J)
120  PRINT C$ (J) ; TAB (20) ; LEFT$ (T$ (J), 2) ;
130  PRINT " H " ; MID$ (T$ (J), 3, 2) ; " M " ;
140  PRINT RIGHT$ (T$ (J), 2) ; " S " : NEXT J
150  GOTO 60
160  DATA LONDON, 0, MOSCOW, 3, RIO DE JANEIRO, −3
170  DATA SYDNEY, 10, HONOLULU, −10, TOKYO, 9, CAIRO, 2
180  DATA NEW YORK, −5, SAN FRANCISCO, −8, PARIS, 1
```



DIFFERENZE ORARIE

Godersi la musica

Questa pagina riguarda le istruzioni TEMPO e MUSIC, fanno sì che l'elaboratore suoni della musica. L'instruzione TEMPO assegna il tempo, come è logico.

Le note musicali vengono convertite in stringhe in accordo con il tono (compresi i semitoni e le ottave) e con la durata dei toni e vengono poi suonate dall'istruzione MUSIC.

Prima di dare una descrizione dettagliata delle istruzioni TEMPO e MUSIC, qui sotto ne diamo una descrizione della costituzione:

Per l'assegnazione del tempo	:	si usa TEMPO per un tempo dal lento al veloce, usando numeri o variabili che hanno un valore da 1 a 7.
Per suonare una melodia	:	si usa MUSIC, con delle variabili di stringa per assegnare le note.
Assegnazioni di note	:	Ottave #(Diesis) Scala Durata

TEMPO X(X=da 1 a 7) : Assegnazione del TEMPO

TEMPO X è un'istruzione che assegna un tempo mediante X. Se non viene assegnato alcun tempo, l'istruzione MUSIC viene automaticamente eseguita con una assegnazione di TEMPO 4.

30	TEMPO 1	Il più lento (Lento e Adagio)
30	TEMPO 4	Moderato: 4 volte più veloce di TEMPO 1
30	TEMPO 7	Il più veloce (Molto Allegro e Presto): 7 volte più veloce di TEMPO 1.

Muto le stringhe in musica

MUSIC "Stringa", M$(Variabile di stringa)

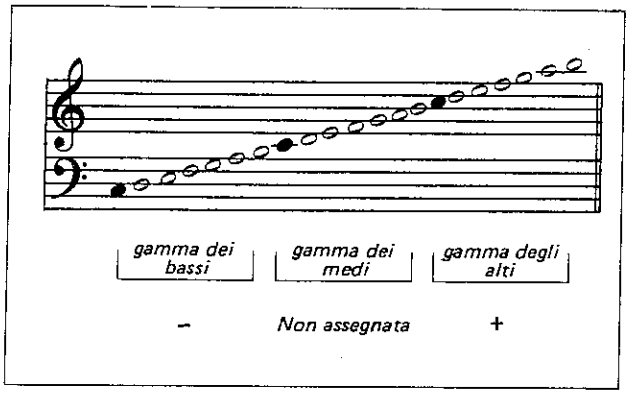
Questa è un'istruzione mediante la quale viene effettivamente prodotta su di un altoparlante una melodia o un effetto sonoro assegnato per mezzo di una stringa o di una variabile di stringa. Il suo tempo è in conformità a quello assegnato mediante l'istruzione TEMPO.

Quanto segue indica come la melodia o l'effetto sonoro siano covertiti in una stringa. Le note musicali vengono assegnate in accordo con la tonalita del tono (ottava o scala) e con la durata (quarto di nota o ottavo di nota). Viene data inoltre una descrizione dell'assegnazione dell'ottava, della scala e della durata in quest'ordine.

Assegnazione dell'ottava –, +

La gamma di suoni che è possibile ottenere dall'elaboratore copre tre ottave, come indicato a destra. Il punto nero indica un DO ("do" in do Maggiore), e tre note di do sono separate da un'assegnazione dell'ottava, come segue:

DO nella gamma dei bassi . . . . . –C
DO nella gamma dei medi C
DO nella gamma degli alti . . . . . +C

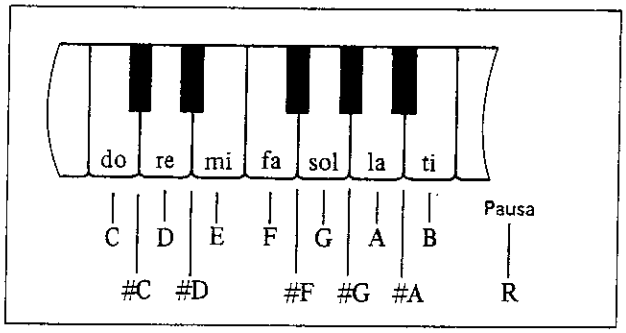


Assegnazione della scala: CDEFGAB #F R

Per l'assegnazione della scala vengono usati i simboli CDEFGAB e il simbolo # (diesis).

Nel diagramma riportato a destra, viene indicata la relazione fra la scala e CDEFGAB. Il simbolo del diesis # è usato per l'assegnazione dei semitoni.

La pausa (nessun suono) viene assegnata mediante R.



Assegnazione della durata

L'assegnazione della durata viene fatta per una tonalità il cui tono è già stato assegnato. Le note da 1/32 ai 4/4 vengono assegnate mediante i numeri da 0 a 9 (la stessa assegnazione di durata e validità è valida anche per R).

Pausa di 1/32	Pausa di 1/16	Pausa di 1/16 con punto	Pausa di 1/8	Pausa di 1/8 con punto	Pausa di 1/4	Pausa di 1/4 con punto	Pausa di 2/4	Pausa di 2/4 con punto	Pausa di 4/4
nota di 1/32	nota di 1/16	nota di 1/16 con punto	nota di 1/8	nota di 1/8 con punto	nota di 1/4	nota di 1/4 con punto	nota di 2/4	nota di 2/4 con punto	nota di 4/4
0	1	2	3	4	5	6	7	8	9

Quando vengono ripetute delle note che hanno la stessa durata, non c'è bisogno di ripetere l'assegnazione di durata dalla seconda nota in poi.

Se all'inizio non viene assegnata alcuna durata, l'esecuzione del programma viene fatta come se fossero state assegnate note di 1/4 (durata 5).

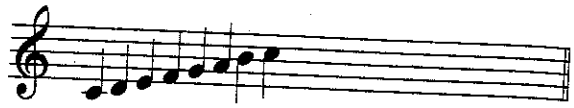
Preludio, Allegro, ma non troppo

Facciamo un po' di esercizi sulla conversione di note musicali in stringhe. Le note che seguono vengono convertite facendo riferimento alla tavola contenuta nella pagina precedente.

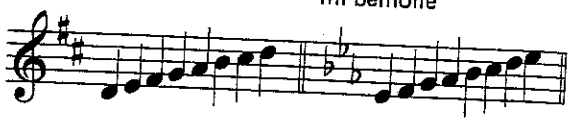
♪ Prima di tutto, le scale in do Maggiore, re Maggiore e mi bemolle (1/4 di nota).

do Maggiore "CDEFGAB + C"
re Maggiore "DE #FGAB + #C + D"
mi bemolle "#DFG#G#A + C + D + #D"

do Maggiore



re Maggiore mi bemolle



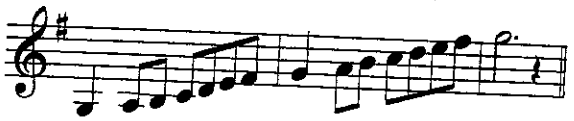
♪ Mettiamo le due scale di ottava in sol Maggiore in G$, usando note da 1/4 e 1/8 di durata.

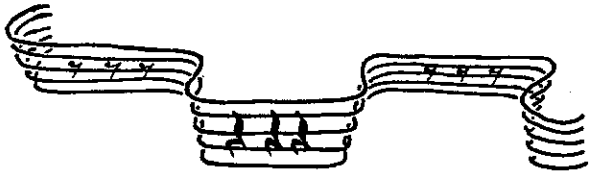
```
G$ = " – G – A 3 – B C D E #F G 5
     A 3 B + C + D + E + #F + G 8
     R 5"
```

pausa di 1/8

nota sol di 2/4 con punto nella gamma degli alti

sol Maggiore

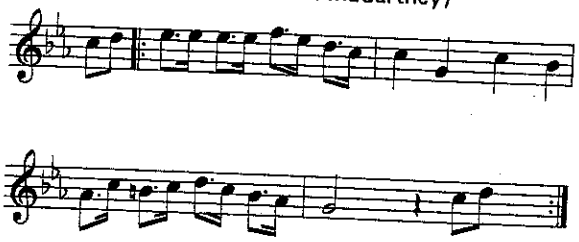




♪ Mettiamo l'inizio della canzone GIRL dei Beatles in GR$.

```
GR$ = " + C 3 + D
      + #D 4 + #D 1 + #D 4 + #D 1
      + F 4 + #D 1 + D 4 + C 1
      + C 5 G + C #A
      #G 4 + C 1 B 4 + C 1
      + D 4 + C 1 B 4 #G 1
      G 7 R 5 "
```

GIRL (John Lennon and Paul MaCartney)



L'inizio di "GIRL" è ripetuto più volte.

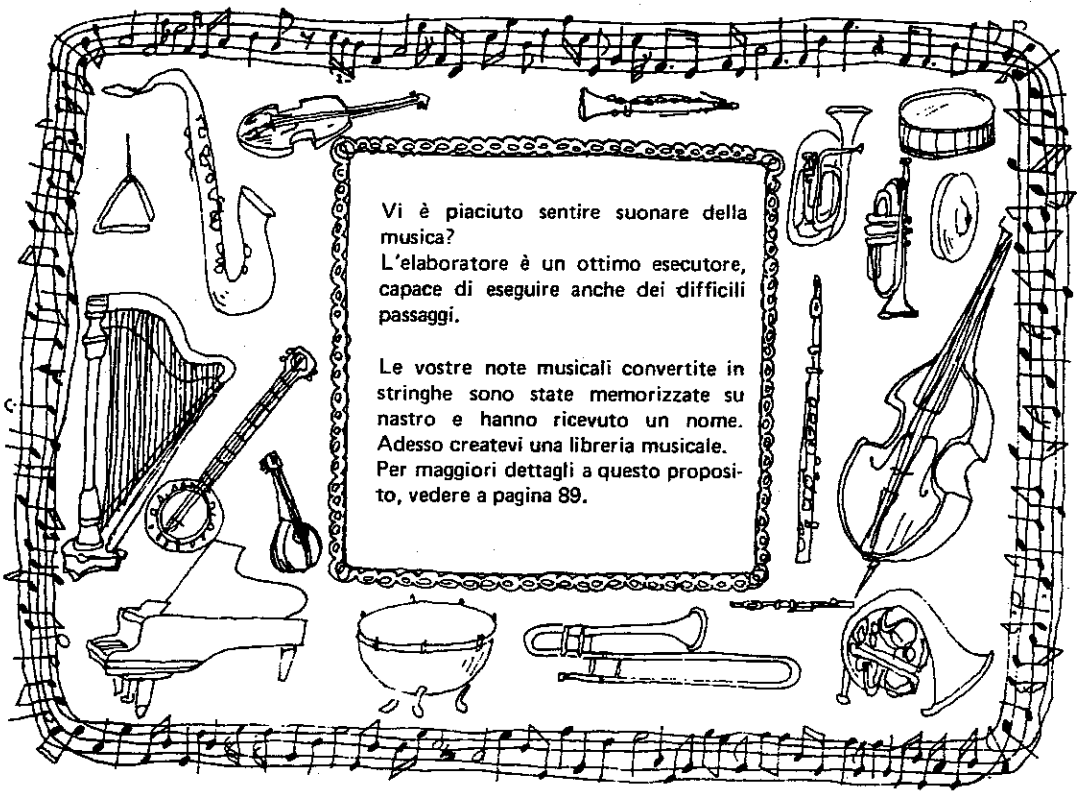
Tutto questo sembra essere molto difficile, ma farete presto ad abituarvici.

In caso di una canzone lunga, la stringa diventa troppo lunga per poter essere contenuta in una istruzione di due righe sullo schermo. In tal caso, la canzone viene codificata mediante più di una variabile di stringa e l'istruzione MUSIC assume come variabile di stringa la loro somma.

E ora fatti una discoteca

♪ Vediamo di mettere assieme una semplice scatola musicale. Questa scatola contiene due brevi melodie, una piacevole e l'altra triste. Le stringhe sono dimensionate nel modo giusto per essere usate nella scatola. Immettete una J od una N.

```
10  REM MUSICASSETTA (DA UNO STUDIO DI F. KROEPSCH)
20  J1$ = "C 1 + C + E + C + G + E + C + E C A + D A + F + D A + D"
30  J2$ = "G B + B + A + G + #F + F + D B + C + E + C B G F D "
40  N3$ = "C – B C – G E C G E + C 5 R "
50  N1$ = "D – F – A D F A – G – #A D G A #A – A #C E G #A A G F + F + D A F "
60  N2$ = "D F A + D + F + A E G #A + E + G + #A + A + F + D + E + #C A + D #G A E F #C "
70  N3$ = "D – D – F – A D E F #C D F A + #C + D 5 R 5 "
100 INPUT "MAGGIORE O MINORE? (G, N)" ; M$
110 IF M$ = " N " THEN 230
200 MUSIC J1$ ; J2$ ; J1$ ; J2$ ; J3$
220 GOTO 100
230 TEMPO 4
240 MUSIC N1$ ; N2$ ; N1$ ; N2$ ; N3$
250 GOTO 100
```



Vi è piaciuto sentire suonare della musica?
L'elaboratore è un ottimo esecutore, capace di eseguire anche dei difficili passaggi.

Le vostre note musicali convertite in stringhe sono state memorizzate su nastro e hanno ricevuto un nome. Adesso createvi una libreria musicale.
Per maggiori dettagli a questo proposito, vedere a pagina 89.

Domani voglio svegliarmi alle 7

Alice ama dormire. Ogni volta che decide di alzarsi presto al mattino per passeggiare nei boschi, si sveglia e si accorge che è già ora di andare a scuola. Alice ha parlato di questo problema con il suo Principe e lui le ha consigliato un programma, dicendole: "Alice, perché non metti la sveglia, in modo da poterti alzare alle 7. Suonerà il tuo pezzo favorito di Chopin e lo verrò nel boschi ad attenderti".

```
10  REM CHOPIN'S MAZURKA
20  MM$ = "A3" : M1$ = "A 5 + #C 3 + D + E + #F + G + #F 0 + G + #F 4 + E 3 + D + #C B "
30  M2$ = "A 3 + D 2 R 0 + D 1 + E 2 + D + #C 3 B + #C 7 + #C 3 "
40  M3$ = "A 3 + #C 2 R 0 + #C 1 + D 2 + #C B 3 A + D 7 + D 3 "
100 PRINT " ■ " : INPUT "CHE ORA E`?" ; TI$
110 INPUT "A CHE ORA DESIDERI SVEGLIARTI?" ; TM$
120 PRINT " ■ "
130 CURSOR 17, 6 : PRINT TI$
140 IF TI$ <> TM$ THEN 130
150 TM$ = "9999"
160 TEMPO 3
170 MUSIC MM$, M1$, M2$, M1$, M3$, M1$, M2$, M1$, M3$
180 GOTO 120
```

Immettere l'ora effettiva nella prima istruzione di INPUT. Esatto, l'ora effettiva viene messa in TI$. Come vi ricorderete, si deve usare un numero di 6 cifre, per esempio 200000 per le otto di sera. Premere il tasto CR contemporaneamente all'immissione del segnale di tempo. Poi immettere l'ora a cui si vuole essere svegliati. Per esempio, 070000 sarebbe il valore giusto per Alice.

Potete anche crearvi il vostro orologio personale per indicare che ora è in tutto il mondo. Tutto dipende dalla vostra abilità nello scrivere programmi.

Due esercizi

Determinare il valore e il grafico di SIN per un angolo il cui valore varia da 0 a 180 gradi con incrementi di 10 grad:

```
10  PRINT "  ■■  " ; TAB (5) ; "SIN"
20  PRINT
30  FOR K = 0 TO 180 STEP 10
40  X = K * π/180 · · · · · Il valore in gradi cambia
50  S = SIN (X)              per ottenere il radiante.
60  A = INT (10 * S)
70  PRINT K ; TAB (4) ;
80  PRINT S ; TAB (18 + A) ;
90  FOR J = 0 TO A
100  PRINT " * " ;
110  NEXT J
120  PRINT
130  NEXT K
140  END
```

Far eseguire il programma riportato qui sopra. Il grafico risultante è abbastanza grossolano. Per quanto riguarda il disegno di grafici, vi sono molti esempi più avanti in modo che potrete imparare molto di più al loro proposito.

Scriviamo adesso un programma che genera i numeri primi. Un numero primo è un numero che non può essere diviso per nessun numero inferiore, eccetto che per 1. Dato che il primo numero è 2, i multipli di due, e cioè i numeri pari, non sono numeri primi. I numeri pari vengono esclusi fin dall'inizio, in modo di poter usare le variabili indexate in modo efficiente.

```
10  DIM P (255)
20  FOR J = 0 TO 255
30  P (J) = J * 2 + 3 : NEXT J
40  FOR K = 0 TO SQR (255)
50  IF P (K) = 0 THEN 90
60  KK = K + P (K)
70  FOR L = KK TO 255 STEP P (K)
80  P (L) = 0 : NEXT L
90  NEXT K
100  PRINT 2 ;
110  FOR M = 0 TO 255
120  IF P (M) = 0 THEN 140
130  PRINT P (M) ;
140  NEXT M
150  END
```

Sostituire 256 numeri dispari da 3 a 513 nella variabile P con indice.

Trovare i numeri primi di valore più piccolo e sostituire degli zeri ai loro multipli contenuti in P.

Viene visualizzato per primo il solo numero pari, "2", e poi i valori di P ( ) che non sono uguali a zero, e cioè i numeri primi.

Questo programma è un po' complesso, vero? Notate che i multipli dei numeri primi vengono esclusi fin dall'inizio.

Consigli su come far le liste

Quando si fà una lista di nomi, di solito questi vengono disposti in ordine alfabetico. L'uso di un eventuale programma può facilitare le liste di ogni tipo.

In questa pagina vi insegnamo come selezionare stringhe per usarle in agende degli indirizzi, in elenchi telefonici o per fare i conti di casa.

```
10  PRINT "■QUANTE PERSONE VENGONO SELEZIONATE?"
20  INPUT X
30  DIM N$ (X)
40  PRINT "INTRODUCI UN NOME ALLA VOLTA"
50  PRINT "BATTI 0 PER TERMINARE"
60  FOR A = 1 TO X : A$ = STR$ (A)
70  PRINT "NAME PLEASE " ; " ( " ; A$ ; " ) "
80  INPUT N$ (A)                          <- Viene immesso un nome
90  IF N$ (A) = "0" THEN 110
100  NEXT A
110  A = A – 1
120  FOR B = 1 TO A – 1
130  FOR C = 1 TO A – B
140  D = LEN (N$ (C)) : E = LEN (N$ (C + 1)) : F = 1 : IF D < E THEN E = D
142  X = ASC (MID$ (N$ (C), F, 1))
143  Y = ASC (MID$ (N$ (C + 1), F, 1)) : IF X > Y THEN 150
144  IF X < Y THEN 180
145  IF (E = F) * (D = E) THEN 180
146  IF (E = F) * (D > E) THEN 150
148  F = F + 1 : GOTO 142
150  K$ = N$ (C)
160  N$ (C) = N$ (C + 1)                   } L'ordine viene cambiato.
170  N$ (C + 1) = K$
180  NEXT C, B
190  PRINT
200  FOR B = 1 TO A
210  PRINT N$ (B)                          <- Il risultato viene visualizzato
220  NEXT B
230  PRINT : END
```

Lista originale (immessa)

MARIO ROSSI
LUIGI LUCCINI
GIANCARLA SCAGLIONI
MARCO GRASSI
PAOLO BRUSCHI
GIO DELGROSSI

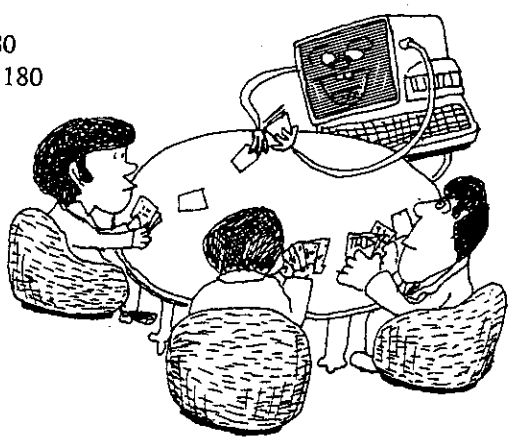
Lista selezionata

GIANCARLA SCAGLIONI
GIO DELGROSSI
LUIGI LUCCINI
MARCO GRASSI
MARIO ROSSI
PAOLO BRUSCHI

Da le carte come un giocatore di Poker

L'elaboratore distribuisce le carte per voi. Le mescola nel modo giusto, usando numeri a caso ed evitando che possa avvenire qualsiasi trucco.

```
 10  DIM X (4, 13)
 20  C = 0
 30  PRINT : FOR A = 1 TO 5
 40  GOSUB 90 : PRINT : NEXT A : PRINT
 50  PRINT "VUOI CAMBIARE CARTE?"
 60  INPUT "NO (1), DAMMENE UN'ALTRA (2)" ; A
 70  ON A GOTO 400, 30
 80  GOTO 50
 90  C = C + 1 : IF C = 51 THEN 500
100  M = INT (4 * RND (1)) + 1
110  N = INT (13 * RND (1)) + 1
120  IF X (M, N) = –1 THEN 100
130  X (M, N) = –1
140  IF N = 1 THEN PRINT "ACE : " ; : GOTO 180
150  IF N = 10 THEN PRINT N ; TAB (5) ; " : " ; : GOTO 180
160  IF N < 10 THEN PRINT N ; TAB (5) ; " : " ; : GOTO 180
170  ON N – 10 GOTO 200, 210, 220
180  ON M GOTO 300, 310, 320, 330
200  PRINT "JACK" ; TAB  (7) ; ":" ; : GOTO 180
210  PRINT "REGINA" ; TAB (7) ; ":" ; : GOTO 180
220  PRINT "RE" ; TAB (7) ; ":" ; : GOTO 180
300  A$ = " ♠ " : GOTO 340
310  A$ = " ♥ " : GOTO 340
320  A$ = " ◆ " : GOTO 340
330  A$ = " ♣ " : GOTO 340
340  FOR B = 1 TO N
350  PRINT A$ ;
360  NEXT B
370  RETURN
400  PRINT
410  PRINT "ALLORA IO MISCHIO LE CARTE."
420  FOR M = 1 TO 4 : FOR N = 1 TO 13
430  X (M, N) = 0
440  NEXT N, M : GOTO 20
500  PRINT
510  PRINT "RIMANGONO DUE CARTE . . . . . VUOI CONTINUARE?"
520  INPUT "SI (1), NO (2)?" ; B
530  ON B GOTO 400, 550
540  GOTO 510
550  END
```



Punti notevoli del programma:

Righe 100 e 110	Scoprire una nuova carta.
Riga 120	Se la carta scoperta era già stata scoperta, si deve girare un'altra carta.
Riga 130	Marcare le carte distribuite con "–1".
Righe da 420 a 440	Tutte le carte vengono raccolte e i loro indicatori vengono portati a 0.

Memorizzare un programma (SAVE)

E' opportuno prendere nota del valore del contatore quando si usa il comando SAVE.

Immettere SAVE e premere CR

Immettere SAVE "NAME" e premere CR

Il nome non deve avere più di 16 caratteri.

Visualizzazione sullo schermo

↓ RECORD · PLAY

L'approntamento della memorizzazione è tutta a posto!!

Funzionamento del registratore a nastro a cassetta

Premi i tasti RECORD e PLAY

Ha inizio la registrazione!!

Visualizzazione sullo schermo

Writing "NAME"

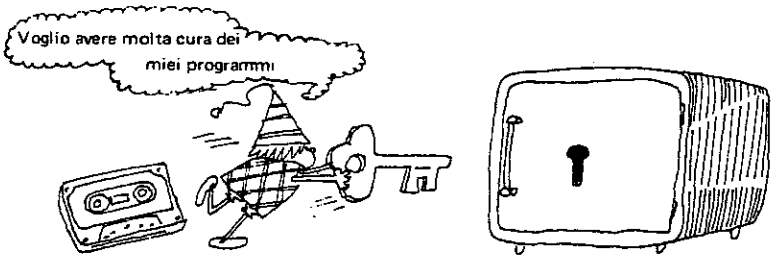
La registrazione è in corso

Se al programma non viene passato un nome di file sullo schermo non viene visualizzato alcun nome.

Visualizzazione sullo schermo

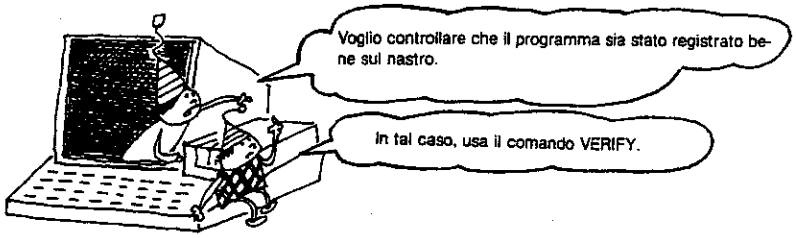
Ready

La registrazione è terminata



■ Uso di VERIFY e di LOAD

Verify



(1) Riavvolgere il nastro.

(2) **VERIFY "NAME"** CR
Non è necessario specificare "Nome" se al programma non è stato fornito un nome del file.

(3) ⬇ **PLAY**... Lo schermo visualizza.

(4) Il pulsante PLAY è stato premuto. Il registratore a nastro a cassetta è in funzione.

(5) Found "****" ..... Questo compare sullo schermo se viene trovato un altro programma. Il suo nome di file, se dato, viene visualizzato come *****
***** (nome del file)

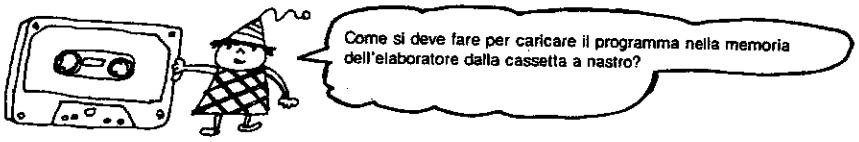
(6) Found "NAME" ..... Visualizzazione che dice che il programma da verificare è stato trovato.

(7) Verifying "NAME"
È in esecuzione il controllo che la registrazione sia stata correttamente eseguita.

(8) OK
Ready
Significa che la registrazione è stata verificata.

(8) CHECK SUM ERROR
* Error 70
Ready
Significa che si è verificato un errore nella registrazione. Quando questo messaggio viene visualizzato, ripetere l'operazione SAVE.

Load



(1) **LOAD "NAME"** CR
(2) ⬇ **PLAY**
(3) **Premi il tasto PLAY**
(4) Found "****"
(5) Found "NAME"

Ciò che viene visualizzato e le operazioni da compiere sono le stesse che per i punti da (3) a (6) del comando VERIFY, eccetto che per i punti (4) e (5) che sono visualizzazioni tipiche del comando LOAD.

(6) Loading "NAME" ..... È in corso il trasferimento all'elaboratore (caricamento)

(7) Ready ..... Il caricamento è termianto.

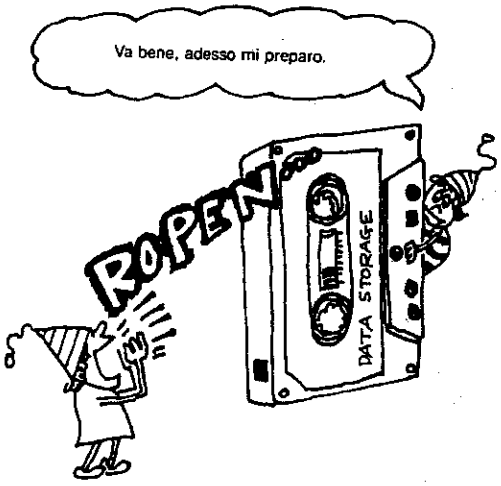
Anche i dati, nella cassetta a nastro

Se si deve memorizzare un programma, è necessario avere a disposizione dello spazio per memorizzare i dati . . . . . Per far questo, bisogna imparare altri 5 comandi. Poi si può usare un nastro a cassetta per memorizzare i dati.

WOPEN/T . . . . . Prepara alla scrittura dei dati. Serve anche per dare il nome ad un gruppo di dati.
PRINT/T . . . . . Identica nel modo di usarla all'istruzione PRINT, registra i dati su un nastro a cassetta.
ROPEN/T . . . . . Questa istruzione prepara alla lettura di quei dati. Serve anche a trovare il gruppo di dati di cui è stato dato il nome.
INPUT/T . . . . . Identica nel modo di usarla all'istruzione INPUT, legge dati dal nastro a cassetta.
CLOSE/T . . . . . Se è stata eseguita una WOPEN o una ROPEN, questa istruzione deve venire eseguita prima che si possa eseguire, rispettivamente, una ROPEN od una WOPEN.

Nel programma che segue, i numeri da 1 a 99 vengono scritti su di un nastro a nastro a cassetta. La parola DATA nella riga 10 rappresenta il nome che viene assegnato ad un gruppo di dati che devono venire scritti. Per il nome di un gruppo di dati si possono usare un massimo di 16 caratteri. Naturalmente, non è obbligatorio dare un nome al gruppo di dati, se non lo si desidera.

```
10 WOPEN/T "DATA"
20 FOR X = 1 TO 99
30 PRINT/T X
40 NEXT X
50 CLOSE/T
60 END
```



Adesso è arrivato il momento di leggere i dati che sono appena stati scritti. Prima di tutto riavvolgere il nastro a cassetta, poi eseguire il programma che segue:

```
ROPEN
10 WOPEN/T "DATA"
20 FOR X = 1 TO 99
30 INPUT/T A
40 PRINT A
50 NEXT X
60 CLOSE/T
70 END
```

Provate ad eseguire di nuovo il programma, dopo aver sostituito 100a 99 nella riga 20. Questa volta si verificherà un errore, in quanto il centesimo dato non è stato scritto. Non è possibile memorizzare il numero dei dati che sono stati scritti. Per segnalare la fine dei dati scritti, si può memorizzare un numero noto qualsiasi, per esempio – 99999999, che non ha nulla a che fare con i dati.

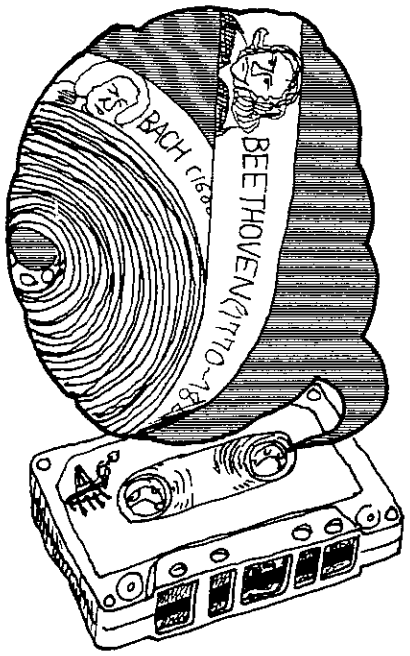
```
10 WOPEN/T "DATA"
20 FOR X = 1 TO 99
30 PRINT/T X
40 NEXT X
50 PRINT/T –99999999
60 CLOSE/T
70 END
```

```
10 ROPEN/T "DATA"
20 FOR X = 1 TO 200
30 INPUT/T A
40 IF A = –99999999 THEN 70
50 PRINT A
60 NEXT X
70 CLOSE/T
80 END
```

Come registri una storia della musica

Le istruzioni per la memorizzazione e la lettura dei dati possono anche venire usate per le stringhe. I nomi dei 5 compositori vengono scritti sulla cassetta e poi riletti.

```
10  DIM N$ (5)
20  N$ (1) = "BACH"
30  N$ (2) = "MOZART"
40  N$ (3) = "BEETHOVEN"
50  N$ (4) = "CHOPIN"
60  N$ (5) = "BRAHMS"
70  WOPEN/T "GRANDI MUSICISTI"
80  FOR J = 1 TO 5
90  PRINT/T N$ (J)
100  NEXT J
110  CLOSE/T
120  END
```



Il che è come scrivere dei dati numerici. Quindi la rilettura viene eseguita nel modo seguente.

```
200  DIM M$ (5)
210  ROPEN/T "GRANDI MUSICISTI"
220  FOR K = 1 TO 5
230  INPUT/T M$ (K)
240  PRINT M$ (K)
250  NEXT K
260  CLOSE/T
```

Con questo programma, scrittura e rilettura sono complete.

Probabilmente vi sarete accorti che il nome della variabile di stringa N$ usata per la scrittura è diverso da quello M$ usato per la lettura. Dato che sul nastro viene scritto il valore del dato stesso, esso non ha nulla a che fare con il nome della variabile sostituita. Ciò rende possibile cambiare il nome della variabile nel programma, purché i dati numerici vengano letti in una variabile numerica e quelli di stringa in una variabile di stringa.

A questo punto, usando ciò che avete imparato fino ad ora, scriviamo un file di dati composto di dati misti, numerici e di stringa. Ad esempio, supponiamo di voler scrivere anche le date in cui i compositori dell'esempio precedente sono morti; nel programma precedente si dovranno modificare le seguenti righe:

```
15  DIM D (5)
65  D (1) = 1750 : D (2) = 1791 : D (3) = 1827
67  D (4) = 1849 : D (5) = 1897
90  PRINT/T N$ (J), D (J)
```

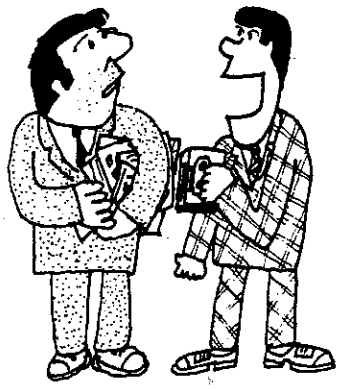
Dal programma appare chiaro che il file creato contiene dati numerici e di stringa mescolati in sequenza alternata. Nello stesso modo, la lettura del file deve rispettare la stessa sequenza alternata e quindi le righe 200, 230 e 240 devono venire modificate in questo modo:

```
200  DIM M$ (5), T (5)
230  INPUT/T M$ (K), T (K)
240  PRINT M$ (K), T (K)
```

Se queste righe rimanessero come erano prima, i dati numerici verrebbero trasferiti nella variabile di stringa M$ ( ) causando un errore.

Lista dei lavori scolastici preparata da un insegnante intelligente

Quello che segue è un programma per memorizzare i risultati di tre materie, Inglese, Francese e scienze, per una certa classe.



```
10  INPUT "QUANTI STUDENTI VI SONO NELLA CLASSE?" ; N
20  DIM N$ (N), K (N), E (N)
30  DIM R (N)
40  A$ = "(VOTI)"
50  FOR X = 1 TO N
60  PRINT : PRINT X
70  INPUT "IL NOME PREGO?" ; N$ (X)
80  PRINT "FRANCESE" ; A$; : INPUT K (X)
90  PRINT "INGLESE" ; A$ ; : INPUT E (X)
100  PRINT "SCIENZE" ; A$ ; : INPUT R (X)
120  NEXT X
130  WOPEN/T "RESULT"        <- Per scrivere un certo gruppo di dati chiamato "RESULT".
140  PRINT/T N               <- Scrittura del numero di studenti della classe.
150  FOR X = 1 TO N
160  PRINT/T N$ (X), K (X), E (X) ; R (X)   <- Scrittura dei voti degli studenti.
170  NEXT X
180  CLOSE/T                 <- Completamento della scrittura.
```

Rileggiamo adesso i dati contenenti i voti e facciamo la media dei singoli studenti e la media per ogni materia.

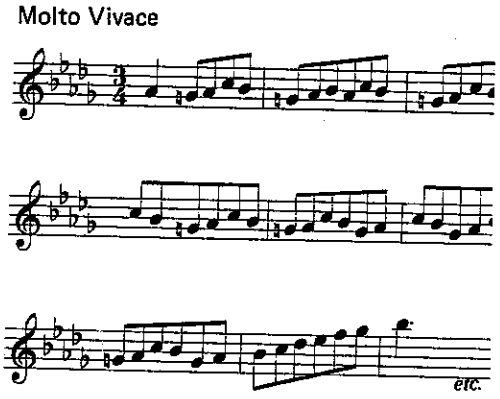
```
10  ROPEN/T "RESULT"        <- Per trovare il gruppo di dati chiamato " RESULT ".
20  INPUT/T N               <- Lettura del numero di studenti per ogni classe.
30  DIM N$ (N), K (N), E (N)
40  DIM R (N)
50  FOR X = 1 TO N
60  INPUT/T N$ (X), K (X)   <- Lettura del nome e dei voti per Francese.
70  INPUT/T E (X), R (X)    <- Lettura dei voti per Inglese e scienze.
80  NEXT X
90  CLOSE/T                 <- Lettura terminata.
100  PRINT TAB (8); "FRANCESE" ;
110  PRINT TAB (15) ; "INGLESE" ;
120  PRINT TAB (23) ; "SCIENZE" ;
130  PRINT TAB (33) ; "MEDIA"
140  FOR X = 1 TO N
150  PRINT N$ (X) ; TAB (11) ; K (X) ;
160  PRINT TAB (20) ; E (X) ;
170  PRINT TAB (27) ; R (X) ;
190  PRINT TAB (35) ; INT (10/3 * (K (X) + E (X) + R (X)))/10
200  K (0) = K (0) + K (X) : E (0) = E (0) + E (X)
210  R (0) = R (0) + R (X)
220  NEXT X : PRINT "MEDIA" ;
230  PRINT TAB (11) ; INT (10 * (K (0)/N))/10 ;
240  PRINT TAB (20) ; INT (10 * (E (0)/N))/10 ;
250  PRINT TAB (27) ; INT (10* (R (0)/N))/10 ;
260  END
```

Una libreria musicale tenuta su nastro

Questo file di dati è indispensabile per creare una Libreria musicale come descritta nel pragrafo "Istruzione MUSIC".

I dati per le varie musiche sono dati di stringa che consistono di vari gruppi di simboli. Se i gruppi di dati ricevono un nome per ogni pezzo musicale, ognuno di questi può venire richiamato da nastro quando viene indicato il suo nome.

Per esempio, si può prelevare un pezzo musicale dalla libreria musicale per usarlo nella scatola musicale del vostro orologio, con qualche modifica. I pezzi contenuti nella libreria musicale possono anche venire usati per programmi che eseguono giochi e grafici, fornendo un gran numero di applicazioni.



F. Chopin " Puppy Waltz "

Per scrivere su un file di dati lo studio musicale di F. Kroepsch, usato a pagina 79, bisogna eseguire le modifiche:

```
300 WOPEN/T "STUDIO"
310 PRINT/T J1$, J2$, J3$, N1$, N2$, N3$
320 CLOSE/T
```

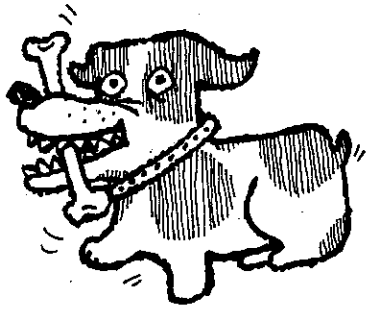
Bisogna fare attenzione al fatto che il numero dei caratteri per dei dati che vengono scritti non deve super

```
305 MA$ = J1$ + J2$ + J3$ : MB$ = N1$ + N2$ + N3$
310 PRINT/T MA$, MB$
```

Il contenuto delle variabili di stringa MA$ e MB$ supera i 255 caratteri e quindi la scrittura dei dati risulta inc

La lunghezza dei dati della stringa può differire da pezzo a pezzo ed è quindi necessario scrivere dei dati ch no la fine di ogni pezzo in modo che non si verifichi un errore di dati durante la lettura. Ad esempio, il segi pezzo " ■■ " fa sì che ogni pezzo musicale consista di una stringa di dati di non più di 100 caratteri nell'es del programma che segue:

```
500 ROPEN/T "PUPPY WALTZ"
510 FOR A = 1 TO 100
520 INPUT/T M$ (A)
530 IF M$ (A) = " ■■ " THEN 550
540 NEXT A
550 CLOSE/T
```



Questo programma può leggere tutto il "Puppy Walts".

Il Data Bank, specialità del computer

La splendida memoria dell'elaboratore rende facile compilare delle liste con un gran numero di voci. Tutta la sequenza di dati, come ad esempio nome, età, data di nascita, indirizzo attuale ed indirizzo permanente possono venire portati in memoria. Questo complesso di informazioni viene chiamato "data bank" (Banca di dati).

Il data bank rende facile leggere e selezionare le voci a seconda del tipo di programma. La lettura viene fatta per tutti i dati relativi a qualsiasi voce richiesta. Per esempio, si può definire un gruppo sanguigno ed usarlo per leggere tutte le persone che hanno lo stesso gruppo sanguigno.

In questo caso si fa una lettura per gruppo sanguigno. Inoltre, tenendo sotto controllo una voce particolare, i nomi possono venire disposti in ordine alfabetico oppure le età possono venire disposte dal più giovane al più vecchio. Questo tipo di procedimento si chiama "selezione".

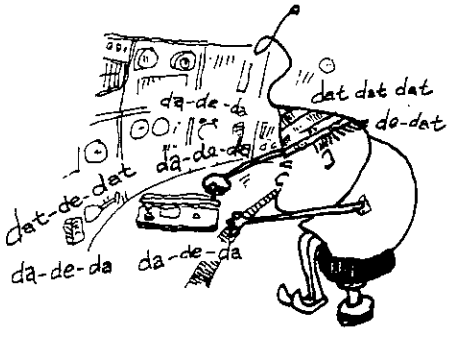
Data Bank anche i numeri telefonici

Avendo compreso quanto precede, viene fatto un sommario di un programma in cui dei dati di stringa vengono messi in memoria con una istruzione DATA. È possibile fare delle modifiche a questo programma per fare in modo che siano disponibili anche l'indirizzo ed il codice di avviamento postale.

```
10  N = 12
20  DIM M$ (N) ..... Cognome
30  DIM N$ (N) ..... Nome
40  DIM A$ (N)..... Prefisso telefonico di casa
50  DIM B$ (N)..... Numero di telefono di casa
60  DIM C$ (N)..... Prefisso telefonico dell'ufficio
70  DIM D$ (N)..... Numero di telefono dell'ufficio
80  DIM F (N)
90  FOR K = 1 TO N
100  READ M$ (K), N$ (K)
110  READ A$ (K), B$ (K) ..... Lettura dei dati
120  READ C$ (K), D$ (K)
130  NEXT K
140  PRINT : PRINT : X = 0
150  PRINT "QUAL'E`IL COGNOME"  ;
160  INPUT X$ ..... Immettere il nome che deve venire letto
170  FOR K = 1 TO N
180  IF M$ (K) = X$ THEN X = X + 1 : F (X) = K . . Lettura usando il cognome
190  NEXT K
200  IF X < > 0 THEN 240
210  PRINT "NON ABBIAMO TROVATO PERSONE DI RILIEVO"
220  PRINT "PER FAVORE RIENTRA"
230  GOTO 140
240  PRINT : PRINT
250  FOR K = 1 TO X
260  L = F (K) ..... Visualizzazione di persone che hanno lo stesso cognome
270  PRINT "NOME" ; TAB (21) ; ":" ; N$ (L)
280  PRINT "NUMERO CIVICO" ; TAB (21) ; ":" ;
290  PRINT "(" ; A$ (L) ; ") " ; B$ (L)
300  PRINT "NUMERO CODICE POSTALE :"
310  PRINT " (" ; C$ (L) ; ") " ; D$ (L) : PRINT
320  NEXT K
330  GOTO 140
340  DATA JONES, JOHN, 01, 364, 9617, 01, 969, 3678
350  DATA DAVIS, PETER, 021, 396, 2137, 01, 323, 6146
360  DATA SMITH, PAUL, 0449, 73246, 0449, 71277
370  DATA JONES, DAVID, 061, 631, 1235, 061, 312, 1975
380  DATA RICHARDS, ROBIN, 0273, 61976, 0903, 47216
390  DATA SMITH, HARRY, 01, 638, 2174, 29, 147636
400  DATA LAKE, COLIN, 4967, 13642, 4967, 32132
410  DATA WATSON, JOHN, 01, 961, 2431, 0427, 21369
420  DATA CARTER, DAVID, 6317, 21974, 01, 316, 2638
430  DATA HOMLES, FRANK, 2238, 76194, 2238, 78352
440  DATA JONES, FRED, 9743, 61665, 01, 424, 6913
450  DATA WILSON, JAMES, 01, 692, 5687, 0374, 68421
460  END
```

Segnali con punti e linee

```
10  DIM A1 (100), M$ (127)
20  M$ (65) = " +A2R2+A5"
30  M$ (66) = " +A5R2+A2R2+A2R2+A2"
40  M$ (67) = " +A5R2+A2R2+A5R2+A2"
50  M$ (68) = " +A5R2+A2R2+A2"
60  M$ (69) = " +A2"
70  M$ (70) = " +A2R2+A2R2+A5R2+A2"
80  M$ (71) = " +A5R2+A5R2+A2"
90  M$ (72) = " +A2R2+A2R2+A2R2+A2"
100  M$ (73) = " +A2R2+A2"
110  M$ (74) = " +A2R2+A5R2+A5R2+A5"
120  M$ (75) = " +A5R2+A2R2+A5"
130  M$ (76) = " +A2R2+A5R2+A2R2+A2"
140  M$ (77) = " +A5R2+A5"
150  M$ (78) = " +A5R2+A2"
160  M$ (79) = " +A5R2+A5R2+A5"
170  M$ (80) = " +A2R2+A5R2+A5R2+A2"
180  M$ (81) = " +A5R2+A5R2+A2R2+A5"
190  M$ (82) = " +A2R2+A5R2+A2"
200  M$ (83) = " +A2R2+A2R2+A2"
210  M$ (84) = " +A5"
220  M$ (85) = " +A2R2+A2R2+A5"
230  M$ (86) = " +A2R2+A2R2+A2R2+A5"
240  M$ (87) = " +A2R2+A5R2+A5"
250  M$ (88) = " +A5R2+A2R2+A2R2+A5"
260  M$ (89) = " +A5R2+A2R2+A5R2+A5"
270  M$ (90) = " +A5R2+A5R2+A2R2+A2"
280  REM NO.
290  M4 (48) = " +A5R2+A5R2+A5R2+A5R2+A5"
300  M$ (49) = " +A2R2+A5R2+A5R2+A5R2+A5"
310  M$ (50) = " +A2R2+A2R2+A5R2+A5R2+A5"
320  M$ (51) = " +A2R2+A2R2+A2R2+A5R2+A5"
330  M$ (52) = " +A2R2+A2R2+A2R2+A2R2+A5"
340  M$ (53) = " +A2R2+A2R2+A2R2+A2R2+A2"
350  M$ (54) = " +A5R2+A2R2+A2R2+A2R2+A2"
360  M$ (55) = " +A5R2+A5R2+A2R2+A2R2+A2"
370  M$ (56) = " +A5R2+A5R2+A5R2+A2R2+A2"
380  M$ (57) = " +A5R2+A5R2+A5R2+A5R2+A2"
390  REM "SPACE"
400  M$ (32) = "R5"
1000  INPUT "TRASMETTI UN MESSAGGIO" ; A$
1010  FOR J = 1 TO LEN (A$)
1020  A1 (J) = ASC (MID$ (A$, J, 1))
1030  NEXT J
1040  FOR J = 1 TO LEN (A$)
1050  MUSIC M$ (A1 (J)), "R5"
1060  NEXT J
1070  GOTO 1000
```

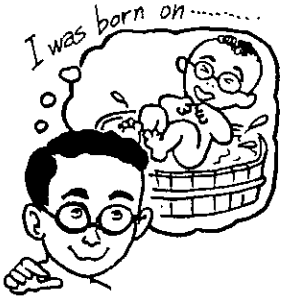


Immetter le lettere da A a Z ed i numeri da 0 a 9. Se per esempio immettete "TI AMO", il programma genera il codice Morse corrispondente. Potete fare una dichiarazione di amore alla vostra ragazza usando il codice Morse!

Tempo senza fine...

Al termine di questa guida al Linguaggio BASIC, viene presentato un programma per un 'Calendario Perpetuo'. Non c'è bisogno di alcuna spiegazione dettagliata. Il nostro "tempo" continua, eterno.

```
 5  DIM M$ (12), W$ (7)
 10  FOR K = 1 TO 12 : READ M$ (K) : NEXT K
 20  FOR K = 1 TO 7 : READ W$ (K) : NEXT K
 30  INPUT "L'ANNO, PREGO!" ; Y : INPUT "IL MESE, PREGO!" ; MT
 40  H = MT : GOSUB 400 : K2 = YB + 1
 50  H = MT + 1 : GOSUB 400 : K1 = YB + 1
 60  N = K1 – K2 : IF N > = 0 THEN L = 28 + N : GOTO 70
 65  L = 35 + N
 70  IF MT = 12 THEN L = 31
 75  PRINT "  ■  " : GOSUB 190
 80  PRINT TAB (8) ; Y ; "    " ; M$ (MT) : PRINT : T = 4
 90  FOR N = 1 TO 7 : PRINT TAB (T) ; W$ (N) ; : T = T + 4 : NEXT N : PRINT
 100  T = 0 : IF K2 = 0 THEN 120
 110  FOR N = 1 TO K2 : PRINT TAB (T) ; : T = T + 4 : NEXT N : T = T – 4
 120  FOR N = 1 TO L : N$ = STR$ (N) : J = LEN (N$)
 130  PRINT TAB (T + 5 – J) ; N$ ; : T = T + 4
 140  IF T = 28 THEN T = 0 : PRINT
 150  NEXT N
 160  IF T < > 0 THEN PRINT
 170  GOSUB 190
 180  PRINT "  ■  " : GOTO 30
 190  FOR Z = 1 TO 31 : PRINT " * " ; : NEXT Z : PRINT : RETURN
 200  DATA GEN, FEB, MAR, APR, MAG, GIU
 210  DATA LUG, AGO, SET, OTT, NOV, DIC
 220  DATA DOM, LUN, MAR, MER, GIO, VEN, SAB
 230  END
 400  X = Y
 410  N = H – 3 : J = 12 : GOSUB 600 : MM = Z
 420  IF MM > 9 THEN X = X – 1
 430  N = X : J = 400 : GOSUB 600 : X = Z
 440  X4 = INT (X/4) : X1 = INT (X/100)
 450  KY = X + X4 – X1
 460  N = MM : J = 5 : GOSUB 600 : MZ = Z
 470  M5 = INT (MM/5) : M2 = INT (MZ/2)
 480  N = MZ : J = 2 : GOSUB 600 : P = Z
 490  KM = 13 * M5 + 5 * M2 + 3 * P
 500  N = KY + KM + 3 : J = 7 : GOSUB 600 : YB = Z
 510  RETURN
 600  REM Z = N, J
 610  K = INT (N/J)
 620  Z = N – K * J
 630  IF Z < 0 THEN Z = Z + J
 640  RETURN
```



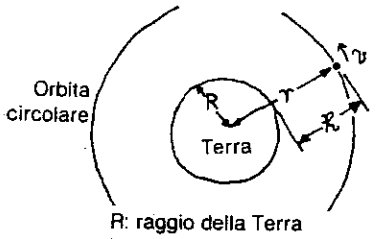

```
**********************************
          1980  JAN

 SUN MON TUE WED THU FRI SAT
                 1   2   3   4   5
   6   7   8   9  10  11  12
  13  14  15  16  17  18  19
  20  21  22  23  24  25  26
  27  28  29  30  31
**********************************

YEAR PLEASE ■
```

Dizionario spaziale in miniatura

Se siete interessati allo spazio, all'astronomia ed ai satelliti creati dall'uomo, potrebbe farvi piacere eseguire con l'elaboratore alcuni calcoli e fare dei grafici. Qui sotto riportiamo le equazioni ed i valori costanti che sono necessari per fare tali calcoli. Il movimento degli oggetti nello spazio, a differenza di quanto avviene sulla Terra, dovrebbe coincidere con i calcoli matematici senza alcuna complessità dovuta all'effetto della resistenza atmosferica. Per ottenere dei valori più accurati si devono però prendere in considerazione gli effetti dei pianeti, le perturbazioni dovute alla non perfetta rotondità della Terra ed alla pressione dei gas nello spazio, per quanto rarefatti essi siano. Ad esempio, nello spazio, ad un'altitudine di 800 Km, vi è una pressione atmosferica pari a 10^{-9} mmHg. Inoltre un satellite artificiale posizionato ad un'altitudine di circa 36.000 Km viene deviato dalla sua orbita di circa 1 grado all'anno a causa dell'influenza degli altri corpi celesti.



R: raggio della Terra
h: distanza dalla superficie della Terra

Velocità di moto di un satellite in orbita ellittica: $v = \sqrt{\frac{G \cdot M}{r}}$ (m/s)

Periodo: $T = 2\pi\sqrt{\frac{r^3}{G \cdot M}}$ (s)

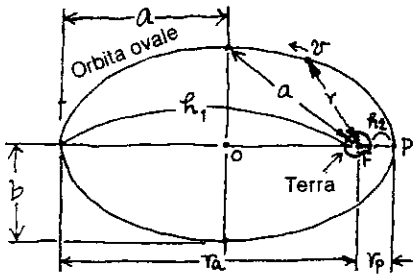
$G = 6.67 \times 10^{-11}$ ($N \cdot m^2/kg^2$)
Costante gravitazionale universale

M: Massa della Terra (kg)

r: Distanza del satellite dal centro della Terra (m)

Velocità di moto di un satellite in orbita circolare: $v = \sqrt{G \cdot M\left(\frac{2}{r} - \frac{1}{a}\right)}$ (m/s)

Periodo: $T = 2\pi\sqrt{\frac{a^3}{G \cdot M}}$ (s)



Raggio maggiore: $a = \frac{h_1 + h_2 + 2R}{2}$ (m)

Raggio Minore: $b = \sqrt{a^2 - (a \cdot e)^2}$ (m)

Eccentricità: $e = OF/OP = \frac{\sqrt{a^2 - b^2}}{a}$

$r_p = a(1 - e)$ (m)

$r_a = a(1 + e)$ (m)

	Massa (Sole uguale a 1)	Raggio Equatoriale	Eccentricità	Distanza media dal Sole (a))
Sole	1.000	696 000 km	—	—
Mercurio	0.166×10^{-6}	2 440	0.20563	0.57910×10^8 km
Venere	2.448×10^{-6}	6 056	0.00678	1.08210 "
Terra	30.034×10^{-7}	6 378	0.01672	1.49600 "
Marte	3.227×10^{-7}	3 390	0.09338	2.27944 "
Giove	95.479×10^{-5}	71 400	0.04829	7.7834 "
Saturno	2.856×10^{-4}	60 400	0.05604	14.2700 "
Urano	4.373×10^{-5}	23 700	0.04613	28.7103 "
Nettuno	5.178×10^{-5}	25 110	0.01004	44.971 "
Plutone	0.552×10^{-6}	3 400	0.24842	59.136 "
Luna	3.694×10^{-8}	1 738	0.0549*	384 400 km*

Massa del Sole = 1.991×10^{30} kg * Valore rispetto alla Terra Fonte: Tavola Cronologica della Scienza

Soluzione di equazioni simultanee

Introduzione allo studio metodologico della programmazione [1]

La soluzione di sistemi di equazioni lineari simultanee è un problema di elaborazione dati basic associato con tutti i problemi di scienze ed ingegneria. In questa sezione, si prova a costruire una sotto routine basic che viene usata per risolvere sistemi di equazioni lineari simultanee in incognite n.

Anche se ci sono diversi algoritmi per la soluzione delle equazioni lineari simultanee, qui si adotta il metodo dell' eliminazione, in qualche modo famigliare al lettore fin dai tempo della scuola.

■ Approccio al problema

Considerare la soluzione del sistema di equazioni lineari simultanee mostato (1) sotto.

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots\cdots+a_{1n}x_n=b_1\ , \\ a_{21}x_1+a_{22}x_2+\cdots\cdots+a_{2n}x_n=b_2\ , \\ \cdots\cdots \\ a_{n1}x_1+a_{n2}x_2+\cdots\cdots+a_{nn}x_n=b_n \end{cases} \tag{1}$$

Moltiplicando entrambi i lati della seconda equazione in (1) per a_{11}/a_{21} e sottraendo il risultato dalla prima equazione, si ottiene:

$$\left(a_{12}-a_{22}\frac{a_{11}}{a_{21}}\right)x_2+\left(a_{13}-a_{23}\frac{a_{11}}{a_{21}}\right)x_3+\cdots\cdots+\left(a_{1n}-a_{2n}\frac{a_{11}}{a_{21}}\right)x_n=b_1-b_2\frac{a_{11}}{a_{21}} \tag{2}$$

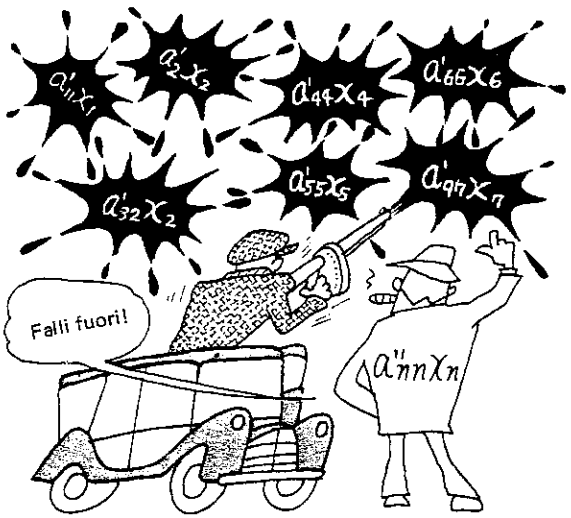
Ciò significa che si è ottenuta una equazione dove x_1 è eliminato. Eseguendo questo processo fino all'ennesima equazione, si ottiene una serie di equazioni lineari simultanee dove non compare x_1, cioè viene eliminata l'incognita. Riscrivendo il coefficente di tutte le equazioni come a'_{22}, a'_{2n}, e così via, si ottiene:

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots\cdots+a_{1n}x_n=b_1\ , \\ \quad a'_{22}x_2+\cdots\cdots+a'_{2n}x_n=b'_2\ , \\ \quad \cdots\cdots \\ \quad a'_{n2}x_2+\cdots\cdots a'_{nn}x_n=b'_n \end{cases} \tag{3}$$

Il processo di creazione di una serie di equazioni (3) dalla serie di equazioni (1) è detto eliminazione usando la prima linea e colonna come perno.

Eseguendo il processo eliminatorio sulla serie di equazioni in (3) usando la seconda linea e colonna come perno, si trova un sistema di equazioni lineari simultanee che dalla terza all'ennesima equazione non contengono il termine x_2. Ripetendo il processo di eliminazione imperniando $n-1$ volte (4), si ottiene:

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots\cdots+a_{1n}x_n=a_{1n+1}\ , \\ \quad a'_{22}x_2+\cdots\cdots\quad a'_{2n}x_n=a'_{2n+1}\ , \\ \quad \cdots\cdots \\ \quad\quad\quad\quad a''_{nn}x_n=a''_{nn+1} \end{cases} \tag{4}$$



Il valore di x_n viene ottenuto dall'ennesima equazione (4), poi il valore di x_{n-1} viene ottenuto dall'equazione $(n-1)$ e così via.

Anche se il metodo sembra semplice, se esso viene risolto manualmente, ci vuole un mucchio di tempo e moltissima carta se vengono coinvolte 5 o 6 incognite. Si perde interesse al problema se vengono coinvolte 10 oppure 20 incognite.

È l'uso migliore per il computer il ripetere operazioni semplici molte volte. Il computer può eliminare gli ostacoli in giri e dare la risposta immediatamente.

■ **Programmazione**

Facciamo l'algoritmo per la soluzione del problema del sistema delle equazioni lineari simultanee.

1	Assegnare il numero delle variabili incognite ad N. Preparare un insieme monodimensionale X (N) per immagazzinare il valore delle incognite. Preparare un insieme bidimensionale A (N, N +1) ed assegnargli i coefficenti delle equazioni (1).
2	Chiamare una sotto routine per la soluzione delle equazioni lineari simultanee.
3	Stampare il valore delle incognite (cioè il contenuto di X (1) fino a X (N) sullo schermo del CRT.

Sottoroutine (per risolvere i sistemi di equazioni lineari simultanee).

4	Eseguire il processo di eliminazioni N−1 volte per cambiare il contenuto dell'insieme A nei coefficenti delle equazioni (4).
5	Trovare i valori delle incognite ed assegnarli a X (N) fino a X (1).

In questo modo è possibile identificare chiaramente la sotto routine (per la soluzione delle equazioni lineari simultanee) come modulo basic che prende il valore della variabile N come il numero delle incognite ed il contenuto dell'insieme bidimensionale A (N, N+1) come il coefficente delle equazioni in (1), trova il valore delle incognite e lo assegna agli insiemi da X (N) fino a X (1).

Per prima cosa si consideri il passo 4 che è il più importante passo di eliminazione. Come appare dalle equazioni in (2), il processo di eliminazione k^{mo} (k = 1 fino a N−1) viene espletato per assegnamento di ripetizione

$$a_{ij} \longleftarrow a_{kj} - a_{ij}\frac{a_{kk}}{a_{ik}} \qquad (\leftarrow \text{denota l'assegnamento}) \qquad (5)$$

per il coefficente a_{ij}. Questo può essere fatto eseguendo un loop di due livelli

```
FOR i=k+1 TO N
    FOR j=k+1 TO N+1
        a(i,j) ⟵ a(k,j) − a(i,j)·a(k,k)/a(i,k)
    NEXT j
NEXT i
```

Loop per una fila

Loop per le file da k a N.

Il passo numero 4 può essere programmato in questo modo utilizzando le variabili K, I e J come segue:

```
FOR K=1 TO N−1
  FOR I=K+1 TO N
    FOR J=K+1 TO N+1
      A(I,J)=A(K,J)−A(I,J)*A(K,K)/A(I,K)
    NEXT J
  NEXT I
NEXT K
```

Questi statement possono essere combinati in uno come NEXT J, I, K.

Passare ora al passo 5, dove i valori delle incognite sono trovati nella sequenza. Per trovare il valore dell'incognita x_i, tutto ciò che si richiede è di assegnare x_{i+1} fino a x_n per la messa a punto delle equazioni in (5). Conseguentemente si ottiene:

```
FOR j=i+1 TO N
    a(i,N+1) ⟵ a(i,N+1) − a(i,j) x(j)
NEXT j
x(i) ⟵ a(i,N+1)/a(i,i)
```

Sebbene il codice di programma

```
FOR I=N TO 1 STEP-1
  FOR J=I+1 TO N
    A(I , N+1)=A(I , N+1)-A(I , J)*X(J)
  NEXT J
  X(I)=A(I , N+1)/A(I , I)
NEXT I
```

sembri soddisfacente, nel programma qui eleborato, quando I = N, J possiede il valore di N + 1 ed il computer esegue lo statement entro il loop controllato da J. Così com'è si incappa in una eccedenza di capacità dimensionale in X (J) oppure si ottiene una risposta sbagliata (anche quando X (N + 1) è definito) se il suo contenuto è diverso da zero. È necessario evitare simili errori mettendo un passo per trovare il valore dell'incognita x_n fuori del loop; ovvero cambiando il loop controllato da I nel modo seguente:

```
X(N)=A(N , N+1)/A(N , N)
FOR I=N-1 TO 1 STEP -1
.....
NEXT I
```

È possiamo completare la sotto routine per la soluzione dei sistemi di equazioni lineari simultanee aggiungendo uno statement RETURN alla fine del codice di programma sopra riportato.

Il problema essenziale del passo [1] è di come assegnare le incognite ed i coefficenti delle equazioni lineari simultanee in questione alla variabile N ed all'insieme bidimensionale A (N, N + 1). Sono applicabili molti metodi, per esempio si possono usare gli statement READ e DATA; però è necessario, a questo punto, decidere di registrarli uno alla volta dalla tastiera. Nel passo 3, si è deciso di mostrare il valore delle incognite sullo schermo del CRT nel format:

```
X1 = .....
X2 = .....
.....
```

Qesti passi possono essere codificati senza difficoltà. Finalmente si ottiene un programma completo come segue:

```
5  REM ..... Dati di lettura
10  INPUT "Number of unknown numbers =" ;N
20  DIM A(N, N+1), X(N)
30  FOR S1=1 TO N: FOR S2=1 TO N+1
40  INPUT A(S1, S2)
50  NEXT S2, S1
100  GOSUB 1000
195  REM ..... Stampa Xi
200  FOR I=1 TO N
210  PRINT "X": STR$(I); "=" ;X(I)
220  NEXT I
230  END
995  REM ..... Eliminazione
1000  FOR K=1 TO N-1
1010  FOR I=K+1 TO N
1020  FOR J=K+1 TO N+1
1030  A(I, J)=A(K, J)-A(I, J) * A(K, K)/A(I, K)
1040  NEXT J, I, K
1095  REM ..... Trovare Xi
2000  X(N)=A(N, N+1)/A(N, N)
2010  FOR I=N-1 TO 1 STEP -1
2020  FOR J=I+1 TO N
2030  A(I, N+1)=A(I, N+1)-A(I, J) * X(J)
2040  NEXT J
2050  X(I)=A(I, N+1)/A(I, I)
2060  NEXT I
2070  RETURN
```

Lanciare il programma e risolvere il seguente sistema di equazioni lineari simultanee con tre incognite.

$$\begin{cases} 3x_1+2x_2-5x_3=13 \\ 5x_1+7x_2+2x_3=-35 \\ x_1-3x_2-x_3=28 \end{cases} \quad (6)$$

Quando il programma gira ed i valori di coefficente ed i valori che appaiono nella parte destra delle equazioni sono registrati come mostrato nella figura, si ha la soluzione seguente:

$x_1=4$
$x_2=-7$
$x_3=-3$

```
RUN
Number of unknown numbers = 3
? 3
? 2
? -5
? 13
? 5
? 7
? 2
? -35
? 1
? -3
? -1
? 28
X1= 4
X2=-7
X3=-3
Ready
```

■ **Esercizi**

1. Risolvere i seguenti sistemi di equazioni lineari simultanee:

$$\begin{cases} x_1+2x_2+3x_3-x_4=5 \\ x_1-3x_2+5x_3+2x_4=3 \\ -x_1-4x_2+x_3+7x_4=-7 \\ -x_1+x_2+11x_3-3x_4=-22 \end{cases} \qquad \begin{cases} x_1+2x_2+4x_3+8x_4+16x_5=32 \\ x_1+3x_2+9x_3+27x_4+81x_5=243 \\ x_1+4x_2+16x_3+64x_4+256x_5=1024 \\ x_1+5x_2+25x_3+125x_4+625x_5=3125 \\ x_1+6x_2+36x_3+216x_4+1296x_5=7776 \end{cases}$$

Il programma precedentemente costruito non specifica il numero delle incognite. In pratica, però, il numero delle incognite viene ristretto allo spazio disponibile per la definizione degli insiemi necessari (oppure l'ampiezza della zona di testo BASIC). Se lo si usa su problemi reali, però, si scopre che il programma può risolvere sistemi di equazioni lineari simultanee con più di 80 incognite quando esso viene girato sull'MZ-80A. Si provi, per esercizio, a risolvere diversi sistemi di equazioni lineari simultanee con un grande numero di incognite.

2. Dopo aver fatto girare il programma diverse volte, è possibile avere un messaggio d'errore
   ∗ Error 2 in 1030

che indica un errore di eccedenza della capacità nell'esecuzione della riga numero 1030. Si incappa in questa condizione se si prova a risolvere un sistema di equazioni lineari simultanee come segue:

$$\begin{cases} x_1+2x_2+3x_3=-26 \\ 3x_1+5x_2+2x_3=-39 \\ 2x_1+4x_2+x_3=-27 \end{cases} \quad (7)$$

La condizione dell'eccedenza della capacità si ha perché x_1 ed x_2 sono eliminati contemporaneamente dalla terza equazione durante l'operazione della prima eliminazione ed il denominatore della divisione alla riga numero 1030 viene messo a punto su 0 durante la seconda operazione di eliminazione.

Questo tipo di errore può essere evitato se si cambia la seconda e la terza linea prima della registrazione dei dati. Siccome la posizione di perno si muove diagonalmente con il procedere del programma, si possono avere condizioni non desiderabili se un elemento diagonale capita essere zero. Si sviluppino le contromisure per questo problema.

Trovare 1000 numeri primi
Introduzione allo studio metodologico della programmazione [2]

È essenziale avere sempre presente nella mente l'obbiettivo e la relativa procedura quando si formula un programma. Molti programmatori, però, non sono consci di questo fatto e creano programmi imperscrutabili e complicati di uno stile tutto personale. Spesso simili programmi sono di difficile comprensione anche per coloro che li creano.

Tali problemi sorgono perché il programmatore non chiarifica la relazione che intercorre tra la struttura del problema e l'algoritmo utilizzato per la sua soluzione quando il programma è scritto. Proprio questo problema ha portato allo studio attivo della metodologia di programmazione. E.W. Dijkstra, una autorità in questo campo, ha scrittomolte opere sull'argomento. In uno di tali libri, egli presenta la sua propria idea sulla programmazione (detta programmazione strutturata), utilizzando il problema di trovare i numeri primi come esempio. In questa sezione si esaminerà brevemente tale condetto usando lo stesso problema.

Problema:

Scrivere 1000 numeri primi: 2, 3, 5, 7, 11 . . . in ordine di grandezza.

- **Approccio al problema**

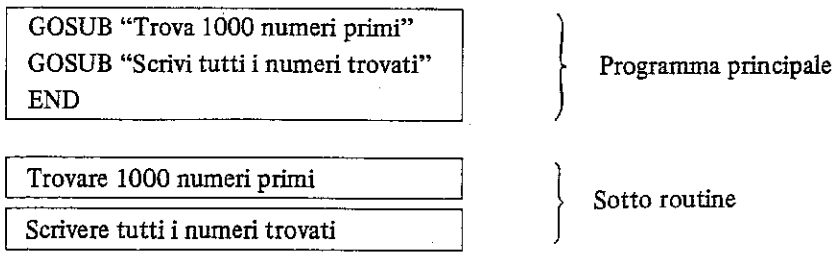
Per la risoluzione di questo problema possono essere usati molti differenti programmi. Il fatto più importante, però, è che il programma costruito dia di utilizzazione pratica.

Un estremo, come approccio, potrebbe essere il seguente. "Trovare e scrivere i numeri primi incominciando dal più piccolo? 2 è il più piccolo numero primo, per cui, scriviamo 2 per primo. Dopo viene 3, poi 5, di seguito 7, tutto ciò che si deve fare è di scriverli per mezzo dello statement PRINT . . . . ".

Questo, però, non è un metodo efficiente per l'uso dell'interpreter BASIC e non può essere classificato come programma.

Secondo ciò che espone Dijkstra, sono necessarie molte decisioni prima di poter completare un programma. È necessario prendere delle decisioni solo quando esse sono praticamente necessarie, piuttosto che crearne senza utilità alcuna. In altre parole si può dire che la programmazione deve essere condotta per gradi.

La prima decisione da prendere nel problema è se i 1000 numeri primi debbomo prima essere trovati e poi mostrati tutti insieme oppure se essi debbono essere mostrati mano a mano che vengono trovati. Per decisione come in precedenza, si mettono i programmi in prospettiva come segue.



Si noti che il problema è stato diviso nettamente in un programma principale ed in due sotto routine. Questo è uno dei principi basilari della programmazione strutturata suggerita da Dijkstra.

Ora si passi al punto successivo. Siccome si debbono trovare 1000 numeri primi prima di scriverli, essi debbono essere registrati in qualche modo in memoria. Il passo successivo, dunque, è definire quale metodo di immagazionazione deve essere usato per i numeri primi.

Non sarebbe adatto dichiarare un insieme numerico di 1000 elementi semplicemente perché 1000 numeri primi debbono essere memorizzati. È possibile definire un insieme di stringa e mettere il "T" (vero) in luoghi nella stringa designati dai subscript che capitano come numeri primi e "F" (falso) negli altri luoghi. Un altro possibile metodo è quello di registrare i numeri primi in un file di dati su cassetta. È facile identificare i numeri primi se essi sono immagazzianati in un insieme di stringa ed identificati dai caratteri "T" ed "F", ed è possibile registrare ed avere tali numeri primi per lungo tempo se sono registrati su cassetta. Quale metodo deve essere usato?

Per un algoritmo nel quale numeri non elaborati sono divisi dai numeri primi già trovati per identificarli, il primo metodo è il più appropriato; ovvero preparare un insieme numerico di 1000 elementi. Per cui si dichiara un insieme numerico di 1000 elementi all'inizio del programma principale.

```
DIM "Insieme numerico di 1000 elementi"
```

In seguito deve essere determinata la struttura dell'insieme. Si può usare PRIM come nome dell'insieme (anche se il nome dell'insieme viene identificato solo dai primi due caratteri) ed usare un insieme bidimensionale numerico. Ciò perché, in quanto il valore massimo per subscript per l'insieme monodimensionale è 255, non è possibile identificare tutti gli elementi dell'insieme con un insieme monodimensionale. Siccome tutti gli elementi dell'insieme debbono essere identificati nella gamma di subscript di 225, si usa una struttura d'insieme nella quale i 1000 elementi siano raggruppati in dieci sottoinsiemi di 1000 elementi ciascuno. Lo statement DIM per l'insieme richiesto è il seguente:

```
DIM PRIM (9,99)
```

Con questa decisione, è possibile prendere in considerazione il format per mostrare i 1000 numeri primi.

Ci sono molti modi per eseguire l'output dei risultati come per esempio la scelta di farli apparire sullo schermo oppure stamparli su carta con la stampatride. Per "format" un numero primo può essere scritto in riga oppure in forma tabulare; e così via dicendo. In questa sede si userà il format più semplice, ovvero scriverli sequenzialmente sullo schermo del CRT senza dar loro un format particolare. La sotto routine seguente ("scrivi tutti i numeri trovati") sarà sufficente per questo scopo:

```
FOR M=0 TO 9 : FOR N=0 TO 99
    PRINT PRIM (M , N) ;
NEXT N , M
```

A questo punto si è terminato il programma principale e la sotto routine "Scrivi tutti i numeri trovati". Rimane ancora da elaborare la sotto routine "Trova 1000 numeri primi". Siccomme si utilizzerà l'insieme prima descritto, è naturale trovare i numeri primi sequenzialmente, a cominciare dal più piccolo e poi metterli in un insieme in ordine creseente di subscript.

Supponendo che si ha una sotto routine "Trova il numero primo successivo" che, dato il parametro I, esamina I+ 1, I+ 2, ... e così via, mettere il primo numero primo trovato in I e far ritornare il controllo al programma di chiamata, è possibile creare la sotto routine "Trovare 1000 numeri primi" nel modo seguente:

```
I←1
FOR M=0 TO 9 : FOR N=0 TO 99
    GOSUB  "Trova il numero primo seguente"
    PRIM (M , N)←I
NEXT N , M
RETURN
```

Ci si avvicina ora al nucleo del programma per trovare i numeri primi. La sotto routine "Trova il numero primo seguente" deve trovare ed assegnare a I il numero primo più piccolo che sia maggiore di I. Un semplicae algoritmo viene subito in mente ed è quello di dividere I per 2, 3, 4, 5, 6, ... I-1, ed identificare I come numero primo quando I è indivisibile per ognuno di essi. Con questo algoritmo è necessario eseguire 99 divisioni usando i divisori da 2 a 100 per riconoscere 101 come numero primo. Si capisce subito che questo algoritmo spreca una grande quantità di tempo. È ovvio pensare che i numeri che non sono divisibili per due non lo sono neanche per i suoi multipli (cioè:4, 6, 8, ...) ed i numeri che non sono divisibili per 3 non lo sono neanche per i suoi multipli, e così via discorrendo. Siccome il proprio obiettivo è quello di trovare 1000 numeri primi sequenzialmente a partire dal più piccolo, per determinare se un numero è primo o meno è sufficente determinare se esso è divisibile per i numeri primi fino al momento trovati. Per esempio per determinare se 101 è un numero primo o no, basta dividerlo per un totale di 25 numeri primi, cioè: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89 e 97.

Anche questo algoritmo spreca tempo. Per esempio, non c'è bisogno di dividere 101 per 11. Quando un numero p può essere diviso per un numero diverso da sé stesso, può essere espresso come $p=f\cdot g$. Presupponendo che f non è maggiore di g ($f \leqq g$), si trova $f^2 \leqq f\cdot g = p$. Per la qual ragione bisogna esaminare solo gli interi che non sono maggiori di $\sqrt{p}$. Per 101, basta dividerlo per i primi quattro numeri (2, 3, 5 e 7), se non è divisibile per questi numeri si può considerare 101 come numero primo. Ciò influenza grandemente l'efficenza del programma. Di fatti per determinare se 10100 (che è 100 volte maggiore di 101) è un numero primo o meno, basta dividerlo per i primi 25 numeri primi da 2 a 97. Altrimenti esso dovrebbe essere diviso, come si vedrà, per più di 1000 numeri primi.

Consideranto tutto ciò che si finóra detto si può formulare ora l'algoritmo per la sotto routine "Trovare il numero primo seguente" nel modo seguente:

```
l1   I ← I + 1 : X ← 0 : Y ← 0
l2   Mentre (PRIM (X, Y))² < I
┌──Dividere I per PRIM (X, Y)
│  ┌─ Se divisibile GOTO l1
│  │    Altrimenti
│  │  ┌─ Determinare i subscript X ed Y degli elementi dell'insieme contenenti il numero primo successivo.
└──└──└─ GOTO l2
Return
```

Questo algoritmo viene codificato nel modo seguente:

```
1500   REM — — Trova il numero primo seguente · — —
1510   I=I+1 : X=0 : Y=0
1520   IF PRIM(X , Y)*PRIM(X , Y)>I THEN RETURN
1530   L=I/ PRIM(X , Y)
1540   IF L- INT(L)=0 THEN 1510
1550   Y=Y+1
1560   IF Y<100 THEN 1520
1570   X=X+1 : Y=0 : GOTO 1520
```

Lo statement della riga 1520 determina se il quadrato del numero primo per il quale il parametro I deve essere diviso eccede I.

La ragione per la quale l'operatore di potenza non viene usato in questo statement è che una espressione che contiene un operatore di potenza non è appropriata come clausola di condizione per lo statement IF causa la valutazione di espressioni contenente l'operatore di potenza che sono internamente condotte per approssimazione†.

Per finire il programma non si dimentichi di assegnare il primo numero primo (cioè 2) a PRIM (0, 0). Ciò perché la sotto routine "Trova il numero primo seguente" presume che ci sia un numero primo precedente.

† Se bisogna usare un operatore di potenza alla riga 1520, la riga deve essere codificata nel modo seguente:
1520 IF INT (PRIM (X, Y) ↑ 2 + 0.00000001) > I THEN RETURN

```
10  REM ..........................................1000 numeri primi
20  DIM PRIM (9,99)
30  PRIM (0,0) = 10
40  GOSUB 1000
50  GOSUB 3000
60  END
1000  REM ..........................................Trovare 1000 numeri primi
1010  I = 1
1020  FOR M = 0 TO 9 : FOR N = 0 TO 99
1030  GOSUB 2000
1040  PRIM (M, N) = I
1050  NEXT N, M
1060  RETURN
2000  REM ..........................................Trovare il numero primo seguente
2010  I = I + 1 : X = 0 : Y = 0
2020  IF PRIM (X, Y) * PRIM (X, Y) > I THEN RETURN
2030  L = I/PRIM (X, Y)
2040  IF L - INT (L) = 0 THEN 2000
2050  Y = Y + 1
2060  IF Y < 100 THEN 2020
2070  X = X + 1 : Y = 0 : GOTO 2020
3000  REM ..........................................Scrivere 1000 numeri primi
3010  FOR M = 0 TO 9 : FOR N = 0 TO 99
3020  PRINT/P PRIM (M, N),
3030  NEXT N, M
3040  RETURN
```

Stampare il listing dei 1000 numeri primi sulla stampatrice lineare MZ-80P5.

```
2       3       5       7       11      13      17      19
23      29      31      37      41      43      47      53
59      61      67      71      73      79      83      89
97      101     103     107     109     113     127     131
137     139     149     151     157     163     167     173
179     181     191     193     197     199     211     223
227     229     233     239     241     251     257     263
269     271     277     281     283     293     307     311
313     317     331     337     347     349     353     359
367     373     379     383     389     397     401     409
419     421     431     433     439     443     449     457
461     463     467     479     487     491     499     503
509     521     523     541     547     557     563     569
571     577     587     593     599     601     607     613
617     619     631     641     643     647     653     659
661     673     677     683     691     701     709     719
727     733     739     743     751     757     761     769
773     787     797     809     811     821     823     827
829     839     853     857     859     863     877     881
883     887     907     911     919     929     937     941
947     953     967     971     977     983     991     997
1009    1013    1019    1021    1031    1033    1039    1049
1051    1061    1063    1069    1087    1091    1093    1097
1103    1109    1117    1123    1129    1151    1153    1163
1171    1181    1187    1193    1201    1213    1217    1223

                        .....

                      Omissis

                        .....

6143    6151    6163    6173    6197    6199    6203    6211
6217    6221    6229    6247    6257    6263    6269    6271
6277    6287    6299    6301    6311    6317    6323    6329
6337    6343    6353    6359    6361    6367    6373    6379
6389    6397    6421    6427    6449    6451    6469    6473
6481    6491    6521    6529    6547    6551    6553    6563
6569    6571    6577    6581    6599    6607    6619    6637
6653    6659    6661    6673    6679    6689    6691    6701
6703    6709    6719    6733    6737    6761    6763    6779
6781    6791    6793    6803    6823    6827    6829    6833
6841    6857    6863    6869    6871    6883    6899    6907
6911    6917    6947    6949    6959    6961    6967    6971
6977    6983    6991    6997    7001    7013    7019    7027
7039    7043    7057    7069    7079    7103    7109    7121
7127    7129    7151    7159    7177    7187    7193    7207
7211    7213    7219    7229    7237    7243    7247    7253
7283    7297    7307    7309    7321    7331    7333    7349
7351    7369    7393    7411    7417    7433    7451    7457
7459    7477    7481    7487    7489    7499    7507    7517
7523    7529    7537    7541    7547    7549    7559    7561
7573    7577    7583    7589    7591    7603    7607    7621
7639    7643    7649    7669    7673    7681    7687    7691
7699    7703    7717    7723    7727    7741    7753    7757
7759    7789    7793    7817    7823    7829    7841    7853
7867    7873    7877    7879    7883    7901    7907    7919
```

1.4 Parole riservate

Una frase BASIC non è composta solamente da porole riservate (comandi, statement, funzioni incorporate, oppure operatore) ma anche da altri elementi come le costanti, le variabili, gli insiemi e le espressioni.

Tutte le parole riservate – chiamate anche parole chiave – hanno uno speciale significato definito dalle regole del BASIC e che non può essere cambiato. Queste parole comprendono i comandi, gli statement, le funzioni incorporate e i segni speciali, definiti come un label variabile dal programmatore.

A	ABS		INT		RETURN
	ASC	L	LEFT$		RIGHT$
	ATN		LEN		RND
	AUTO		LET		ROPEN/T
C	CHARACTER$		LIMIT		RUN
	CHR$		LIST	S	SAVE
	CLOSE/T		LIST/P		SET
	CLR		LN		SGN
	CONT		LOAD		SIN
	COPY/P		LOG		SIZE
	COS	M	MID$		SPACE$
	CSRH		MON		SQR
	CSRV		MUSIC		STEP
	CURSOR	N	NEW		STOP
D	DATA		NEXT		STR$
	DEF FN	O	ON		STRING$
	DIM		OUT	T	TAB
E	END	P	PAGE/P		TAN
	EXP		PEEK		TEMPO
F	FOR		POKE		THEN
G	GET		PRINT		TI$
	GOSUB		PRINT/P		TO
	GOTO		PRINT/T	U	USR
I	IF	R	READ	V	VAL
	INP		REM		VERIFY
	INPUT		RESET	W	WOPEN/T
	INPUT/T		RESTORE		

Figura 1.1

1.5 Sommario delle istruzioni BASIC SA-5510

1.5.1 Comandi

LOAD	LOAD	Legge un programma BASIC memorizzato su cassetta. Dato che in questo caso non si è dato un nome di file, viene letto il primo programma che viene incontrato.
	LOAD "NOME"	Legge un programma BASIC con il nome file "NOME" (il nome del programma può contenere fino ad un massimo di 16 caratteri validi).
	LIMIT $A000: LOAD "B"	Per il load (caricamento) di un machine language da file che deve essere collegato con il testo BASIC, la zona BASIC della memoria deve essere divisa dalla zona del machine language per mezzo dello statement LIMIT. Nota: Quando il comando LOAD viene eseguito per un file di testo BASIC, programmi precedenti vengono cancellati dalla zona di testo.
SAVE	SAVE	Scrive sulla cassetta il programma attualmente memorizzato nella memoria dell'elaboratore.
	SAVE "NOME"	Simile al precedente, scrive il programma dandogli il nome di file "NOME".
VERIFY	VERIFY	Confronta il programma attualmente memorizzato nell'elaboratore ed il programma BASIC scritto nella cassetta. In questo caso, viene confrontato il primo programma che viene trovato. Analogamente al precedente, confronta il programma memorizzato nell'elaboratore e il programma il cui nome di file è "NOME".
	VERIFY "NOME"	
AUTO	AUTO	Automaticamente genera ed assegna numeri di riga 10, 20, 30 .... durante la creazione di programmi.
	AUTO 200, 20	Automaticamente genera numeri di riga ad intervalli di 20 ad iniziare dalla riga 200. 200, 220, 240, 260, ... Il comando AUTO perde il suo valore premendo il tasto BREAK.
RUN	RUN	Esegue un programma. In questo caso, dato che non è specificato alcun numero di riga, l'esecuzione ha inizio dal più basso numero di riga del programma.
	RUN 100	L'esecuzione ha inizio dalla riga 100. Se viene messo un numero di riga inesistente, si verifica un errore (anche RUN azzera tutte le variabili).
LIST	LIST	Visualizza sullo schermo tutto il programma memorizzato nell'elaboratore.
	LIST 70	Visualizza la riga 70 del programma.
	LIST 70 –	Visualizza tutto il programma a partire dalla riga 70.
	LIST 70 – 100	Visualizza le righe di programma tra 70 e 100.
	LIST – 100	Visualizza il programma fino alla riga 100.

LIST/P	LIST/P LIST/P 70 LIST/P 70 – 100 LIST/P – 100	Esegue le stesse funzioni di LIST, ma sulla stampatrice. (Questo dà origine ad un errore se la configurazione dell'elaboratore non comprende una stampatrice).
NEW	NEW 10 NEW	Cancella il programma attualmente presente in memoria e predispone tutte le variabili a 0 o spazio in bianco. Anche questo programma può venire usato durante l'esecuzione di un programma.
CONT	CONT	Qunado l'esecuzione di un programma è stata arrestata (mediante il tasto BREAK o le istruzioni STOP ed END durante l'esecuzione del programma dal punto in cui si era arrestato. (Questo programma non può venire usato se il programma viene corretto usando i numeri di riga. Può invece venire usato il comando in modo diretto).
MON	MON 10 MON	Arresta l'uso delle istruzioni BASIC e restituisce il controllo al monitor. Questo comando può venire usato durante l'esecuzione di un programma, ma bisogna fare molta attenzione!

1.5.2 Istruzioni di assegnamento

LET	⟨LET⟩ A = X + 3	Sostituisce X + 3 nella variabile numerica A. LET può essere omesso.

1.5.3 Istruzioni di input/output

PRINT	10 PRINT A	Visualizza sullo schermo il valore della variabile A.
	20 PRINT A$	Visualizza sullo schermo i caratteri che formano la variabile di stringa A$.
	30 PRINT A; A$, B; B$	Se poi si usa un punto e virgola come separatore, si può ottenere una visualizzazione continua senza spaziature. Se viene usata una virgola come separatore, le visualizzazioni su campi successivi, ognuno dotato di 10 posizioni.
	40 PRINT "COST ="; A	Il contenuto di una stringa contenuta tra virgolette (" ") viene visualizzato esattamente come è stato scritto.
	50 PRINT	In caso di una riga contenente solo una istruzione PRINT, si ha solo la spaziatura di una riga.
INPUT	10 INPUT A	Legge dalla tastiera un numero come valore della variabile A.
	20 INPUT A$	Legge dalla tastiera una stringa come valore della variabile di stringa A$.
	30 INPUT "VALUE?"; A	Prima di leggere l'immissione dalla tastiera, questa istruzione visualizza la stringa VALUE?.
	40 INPUT A, A$, B, B$	Per separare la stringa dalla variabile viene usato il punto e virgola ";". Non viene però stampato di nuovo? per il valore numerico. Variabili numeriche e variabili di stringa possono venire mescolate purchè siano separate da una virgola ","; una volta ricevuta una forma di variabile questa deve però rimanere la stessa.
GET	10 GET A	Legge dalla tastiera un numero di una cifra per la variabile A. Se in questo caso non viene premuto nessun tasto, viene assunto il valore 0.
	20 GET A$	Legge una stringa di un carattere da tastiera per la variabile A$. Se non viene premuto alcun tasto, A$ diventa uno spazio in bianco.

READ~DATA		Sostituisce il contenuto della variabile nominata nell'istruzione READ con quello di una variabile o di una stringa nominata nell'istruzione DATA. La prima istruzione di READ legge il primo valore di DATA, la seconda legge il secondo valore e così via. La variabile dell'istruzione READ ed il corrispondente valore dell'istruzione DATA devono essere dello stesso tipo, con numeri per le variabili numeriche e stringhe per le variabili di stringa. Nell'istruzione DATA non può essere inoltre contenuta alcuna equazione. Le istruzioni DATA possono venire messe in qualsiasi posizione all'interno del programma, ma, per comodità, vengono di norma messe all'inizio o alla fine del programma.
	10 READ A, B, C, D 20 DATA 25, –0.5, 0, 500	A causa delle istruzioni READ e DATA riportate a sinistra, i valori 25, –0,5, 0 e 500 vengono sostituiti alle variabili A, B, C e D.
	10 READ H$, H, S$, S, D$, D 20 DATA HEART, 3 30 DATA CLUB, 9 40 DATA SPADE, 6	Nell'esempio riportato a sinistra, il primo valore nell'istruzione DATA, e cioè la stringa "HEART" viene sostituito nella prima variabile dell'istruzione READ, e cioè la variabile di stringa H$. Il valore 3 viene sostituito nella seconda variabile, H, e quindi i due valori vengono letti uno dopo l'altro.
	10 READ X1$, X2$, Y$ 20 DATA "A, B, C, D" 30 DATA "E, F, G", ":16"	Quando nella stringa sono contenuti i simboli "," e ":", essi debbono venire indicati tra virgolette. Nell'esempio riportato qui a sinistra, le stringhe indicate tra virgolette dell'istruzione DATA vengono sostituite nelle variabili di stringa X1$, X2$ e Y$ rispettivamente.
RESTORE		Nelle istruzioni READ . . . DATA, i valori letti dall'istruzione DATA si spostano progressivamente mano a mano che la READ procede. Con l'uso dell'istruzione RESTORE si può però ritornare alla lettura del primo valore di DATA.
	10 READ A, B, C 20 RESTORE 30 READ D, E 40 DATA 3, 6, 9, 12, 15	Nell'esempio a sinistra, i valori 3, 6 e 9 vengono rispettivamente sostituiti nelle variabili A, B e C dall'istruzione READ della riga 10. Dato che l'istruzione successiva è una RESTORE, i valori che l'istruzione READ alla riga 30 sostituisce nelle variabili D ed E sono rispettivamente 3 e 6 e non 12 e 15.

1.5.4 Istruzioni per eseguire cicli

FOR ~ NEXT STEP	10 FOR A = 1 TO 10 20 PRINT A 30 NEXT A	La riga 10 fissa che la variabile A deve venire cambiata da 1 a 10 e che il primo valore di A è 1. La riga 20 fissa che il valore di A deve venire visualizzato sullo schermo, e quindi viene visualizzato 1. Quando questo loop viene ripetuto, arrivati alla riga 30 il valore di A diventa 2. Di conseguenza viene visualizzato 2 come valore di A. In questo modo il loop viene ripetuto fino a quando A raggiunge il valore di 10. (Quando il loop viene terminato, il valore di A è uguale a 11).
	10 FOR B = 2 TO 8 STEP 3 20 PRINT B↑2 30 NEXT	La riga 10 stabilisce che il valore di B deve venire aumentato da 2 a 8 con incrementi di 3. Se a STEP si assegna un valore negativo, la variabile viene fatta diminuire. Alla riga 20 viene visualizzato B al quadrato. La riga 30 non contiene l'istruzione NEXT B. In questo caso però, dato che non viene assegnata nessuna variabile a NEXT, tale istruzione viene automaticamente collegata con la FOR più prossima. Nell'esmpio riportato qui a sinistra, viene eseguito il loop FOR . . . NEXT relativo alla variabile B. Si suggerisce di dare sempre il nome della variabile cui si riferisce l'istruzione NEXT.

	10 FOR A = 1 TO 3 20 FOR B = 10 TO 30 30 PRINT A, B 40 NEXT B 50 NEXT A	Questo è un esempio di un doppio loop FOR . . . NEXT relativi alle variabili A e B. Fare attenzione al fatto che il loop B è posizionato all'interno del loop A. Loop doppi, tripli, ecc. sono possibili, ma un loop interno deve essere completamente compreso all'interno del loop esterno. Una tale configurazione di molti loop viene chiamata concatenamento. Nell'esempio qui a sinistra, all'inizio la variabile A assume il valore 1 e la variabile B il valore 10. Con A che rimane ad 1, B viene cambiato in 11, 12 e così via fino a raggiungere il valore 30 durante l'esecuzione del loop B. Alla riga 50, il valore di A diventa 2 e ricomincia l'esecuzione del loop B.

1.5.5 Istruzioni di salto

GOTO	100 GOTO 200	Fa saltare il programma dalla riga 100 alla riga 200.
GOSUB	100 GOSUB 700	Fa saltare il programma all'esecuzione della subroutine alla riga 700.
RETURN	800 RETURN	Termina l'esecuzione di una subroutine e restitusce il controllo alla riga specificata nell'istruzione di GOSUB nel programma principale.
ON ~ GOTO	10 ON A GOTO 70, 80, 90	Il programma salta alla riga 70 se il valore della variabile A è 1, alla riga 80 se è 2 ed alla riga 90 se il valore è 3. Esegue l'istruzione successiva se A è uguale a 0 o se è maggiore o uguale a 4. ON presuppone INT e quindi se A è uguale a 2,7 viene considerato uguale a 2 e il programma salta alla riga 80.
ON ~ GOSUB	100 ON A GOSUB 700, 800	Cede il controllo alla riga 700 se A ha il valore 1, alla riga 800 se ha il valore 2. Esegue l'istruzione successiva se A=0 o maggiore o uguale a 3. ON presuppone INT.
IF ~ THEN	10 IF A>20 THEN 100	Salta alla riga 100 se A è maggiore di 20. Esegue l'istruzione successiva se A è minore di 20.
	20 IF A <> 10 THEN 200	Salta alla riga 200 se la variabile A non è uguale a 10. Esegue l'istruzione successiva se A è uguale a 10.
	30 IF B<3 THEN B = B + 3	Sostituisce il valore di B + 3 nella variabile B se la variabile B è minore di 3. Esegue l'istruzione successiva se B è maggiore di 3.
	40 IF B>7 THEN PRINT B	Visualizza il valore di B sullo schermo se tale valore à maggiore di 7. Esegue l'istruzione successiva se B è minore di 7.
IF ~ GOTO	100 IF A>=B GOTO 10	Salta alla riga 10 se il valore della variabile A è maggiore di quello della variabile B. Esegue l'istruzione successiva se A è minore di B.
IF ~ GOSUB	100 IF A=B*2 GOSUB 700	Cede il controllo alla riga 700 se il valore della variabile A è il doppio di quello della variabile B. In caso contrario esegue l'istruzione successiva. (Se una frase condizionale è seguita da più di una istruzione, l'istruzione ON viene eseguita quando la condizione non è soddisfatta, mentre l'istruzione IF cede il controllo alla riga successiva quando la condizione non è soddisfatta e il resto delle istruzioni multiple vengono ignorate).

1.5.6 Istruzioni di definizione

DIM		Quando si usa una variabile indicizzata, il valore massimo dell'indice deve venire dichiarato mediante l'istruzione DIM (abbreviazione di dimensione). L'indice può assumere i valori tra 0 e 255 (massimo). L'indice, purché compreso nella gamma sopra indicata, può anche venire dichiarato mediante una variabile od una equazione.
	10 DIM A (20)	Vengono preparate 21 matrici da A (0) ad A (20) per la variabile A indicizzata.
	20 DIM B (99, 99)	La variabile B con doppio indice può avere fino a 10000 matrici da B (0, 0) a B (99, 99).
	30 DIM A (20), B (99, 99)	Le precedenti righe di istruzioni 10 e 20 possono venire combinate per fare le dischiarative nella forma della riga 30.
	40 DIM C1$ (10)	Vengono preparate 11 matrici per la variabile di stringa con indice C1$ ( ), da C1$ (0) a C1$ (10).
	50 DIM AA (X), AB (X∗2)	Per le variabili con indice AA ( ) e AB ( ) vengono preparate rispettivamente le matrici da AA (0) ad AA (X) e da AB (0) ad AB (X∗2).
DEF FN	100 DEF FNA(X) = X↑2 – X 110 DEF FNB(X) = LOG (X) 120 DEF FNC(Y) = LN (Y)	DEF FN definisce delle funzioni. La riga 100 definisce l'espression X^2-X come FNA (X), la riga 110 definisce $\log_{10}X$ come FNB (X), e la riga 120 definisce $\log_e Y$ come FNC (Y). Le funzioni definite non possono avere più di una variabile. I nomi delle funzioni devono avere un formato del tipo "variabile ( )", come si può vedere dagli esempi dati.

1.5.7 Istruzioni di commento e controllo

REM	100 REM GAME 200 REM GAME : A = 30	Questa istruzione identifica un commento e viene trascurata durante l'esecuzione del programma. Anche nell'istruzione REM, i due punti sono usati per indicare delle righe con più istruzioni; in questo caso il valore 30 viene sostituito nella variabile A.
STOP	200 STOP	Arresta l'esecuzione del programma ed attende la successiva istruzione. Per far riprendere l'esecuzione del programma dal punto in cui era stata sospesa, usare l'istruzione CONT.
END	999 END	Questa istruzione pone termine all'esecuzione di un programma e se viene data l'istruzione CONT si procede all'esecuzione del programma successivo.
CLR	CLR 10.CLR	Azzera tutte le variabili numeriche e mette a spazi in bianco tutte le variabili di stringa. Questo programma può venire usato utilizzato durante l'esecuzione di un programma.
CURSOR	50 CURSOR 25, 15 60 PRINT "ABC"	Il comando CURSOR sposta il cursore per tutto lo schermo. Il primo operando rappresenta la locazione orizzontale della destinazione e deve essere tra 0 e 39. Il secondo operando rappresenta la locazione verticale della destinazione e deve essere tra 0 e 24. L'esempio di sinistra mostra "ABC" ad iniziare dalla locazione (25, 15) (la 26ma posizione da sinistra e la 16ma posizione dall'alto).
CSRH		Sistema di variabile indicante la coordinata X (locazione orizzontale) del cursore.

CSRV		Sistema di variabile indicante la coordinata Y (locazione verticale) del cursore.
SIZE	PRINT SIZE	Visualizza le dimensioni in byte della memoria non utilizzata attualmente disponibile nell'elaboratore.
TI$	TI$ = "102030" 10 TI$ = "102030"	Predispone l'orologio interno su 10 ore, 20 minuti e 30 secondi antimeridiane. Si deve usare un numero di 6 cifre tra virgolette.
	20 PRINT TI$	Visualizza l'ora attuale segnata dall'orologio interno.

1.5.8 Istruzioni di controllo musicale

MUSIC **TEMPO**		Queste istruzioni richiedono che venga eseguita della musica. Il tempo viene assegnato dall'istruzione TEMPO e poi una stringa (equivalente ad un gruppo di note assegnate per scala e durata) specificata tra virgolette " " nell'istruzione MUSIC viene convertita in suoni ed eseguita attraverso un altoparlante.
	MUSIC "CDEFG"	Do, re, mi, fa e sol nel mezzo della scala dei suoni con note di quarta con TEMPO 4.
	300 TEMPO 7	TEMPO 7 (il più veloce) viene assegnato nella riga 300.
	310 MUSIC "DE # FGA"	La riga 310 provoca l'esecuzione di re, mi ' fa diesis, sol e la nel mezzo della scala con durata di nota di quarta. Se non viene specificata l'istruzione TEMPO, la musica viene eseguita con TEMPO 4.
	300 M1$ = "C3EG + C" 310 M2$ = "B3GD – G" 320 M3$ = "C8R5" 330 MUSIC M1$, M2$, M3$	Nell'esempio riportato a sinistra, viene sostituita una melodia alle tre variabili di stringa e poi viene eseguita l'istruzione MUSIC. Viene esequita la melodia riportata qui a destra. Dato che non è assegnato alcun TEMPO, viene eseguita con TEMPO 4.

1.5.9 Istruzioni di controllo grafico

SET **RESET**	SET X, Y RESET X, Y	Illumina sullo schermo le coordinate assegnate dalle variabili X ed Y dell'istruzione SET. L'illuminazione viene rimossa sui punti indicati dalle coordinate dell'istruzione RESET. L'ascissa, partendo dal lato sinistro dello schermo, può assumere i valori da 0 a 79, mentre l'ordinata, partendo dal lato superiore dello schermo, può assumere i valori da 0 a 49. Ciò significa che l'angolo in alto a sinistra ha le coordinate 0, 0 mentre l'angolo in basso a destra ha le coordinate 79, 49. (È bene ricordarsi che, in funzionamento normale, X varia da 0 a 39 ed Y da 0 a 24).
	SET 0, 0	Illumina l'angolo in alto a sinistra.
	10 SET 79, 49 20 RESET 79, 49 30 GOTO 10	A causa delle istruzioni nelle righe da 10 a 30, la luce nell'angolo in basso a destra lampeggia ripetutamente. I valori di X ed Y possono venire assegnati per mezzo di una variabile, di un'equazione o di una costante.

1.5.10 Istruzioni di transferimento dati

WOPEN/T	10 WOPEN/T 20 PRINT/T X, X$	Apre un file per uso esclusivo di dati, per scrivere una variabile numerica od una variabile di stringa sul nastro. (Per la memorizzazione dei programmi si deve usare la SAVE). Simile al precedente, questo comando predispone per la scrittura di dati su nastro in un file dotato di nome "COST". (Il nome di un file può avere fino a 16 caratteri).
	30 WOPEN/T "COST" 40 PRINT/T X, X$	

PRINT/T	10 PRINT/T A 20 PRINT/T A; A$, B; B$ 30 PRINT/T "COST ="; A	La funzione WRITE ha lo stesso effetto della PRINT, ma su nastro. Se però non vi è un file aperto mediante l'istruzione WOPEN/T, si verifica un errore.
ROPEN/T	10 ROPEN/T 20 INPUT/T Y, Y$	Apre un file esclusivo di dati, per leggere una variabile numerica od una variabile di stringa precedentement memorizzata su nastro (Per la lettura di programmi si deve usare la LOAD).
	30 ROPEN/T "COST" 40 INPUT/T Y, Y$	Simile al precedente, questo comando predispone per la lettura di dati variabili da nastro da un file dotato di nome "COST".
INPUT/T	10 ROPEN/T 20 INPUT/T A 30 INPUT/T A$ 40 INPUT/T A, A$, B, B$	Funzione di lettura da nastro, identica come funzionamento alla INPUT. Se però non vi è un file di dati aperto mediante ROPEN/T, si verifica un errore.
CLOSE/T	50 CLOSE/T	Questo comando segnala la fine dell'uso del nastro per dati da file. Quando vengono usate le istruzioni WOPEN/T e ROPEN/T, il file deve essere chiuso.

1.5.11 Istruzioni di controllo di routines in linguaggio machina

LIMIT	LIMIT A 10 LIMIT A	Specifica mediante la variabile A i limiti di memoria che possono venire usati dal programma BASIC. Quando si usa il comando USR, è bene assicurarsi di un certo spazio in memoria con l'uso dell'istruzione LIMIT e memorizzare qui il programma in linguaggio di macchina. (Quando la macchina viene accesa, il limite della memoria viene automaticamente predisposto per la massima memoria disponibile).
	LIMIT MAX 20 LIMIT MAX	Predispone il limite di memoria sul massimo. Quando il limite di memoria viene predisposto usando le istruzioni della precedente riga 10, con questa istruzione si può riportare il limite della memoria al suo massimo.
	200 LIMIT $BFFF 210 LOAD "S-R1"	Esegue il load del programma di linguaggio macchina (programma oggetto) "S-R1" nella zona collegamento linguaggio macchina dal nastro di cassetta quando l'indirizzo di caricamento è $C000 oppure maggiore.
PEEK	10 A = PEEK (24110)	Converte i dati contenuti all'indirizzo 24110 in un numero decimale e lo sostituisce alla variabile A.
	20 B = PEEK (C) (Assegnando un indirizzo nel programma BASIC, viene letto sempre 32)	Converte i dati contenuti all'indirizzo assegnato dalla variabile C (considerata come numero decimale) in un numero decimale e lo sostituisce alla variabile B. (Le variabili A e B sono entrambe predisposte ad un valore compreso fra 0 e 255).
POKE	10 POKE A, D	I numeri (compresi tra 0 e 255) indicati dalla variabile D vengono memorizzati all'indirizzo assegnato dalla variabile A. Pena la distruzione del programma, l'istruzione POKE deve venire usata con molta attenzione.
	20 POKE 24100, 32	Nell'esempio, viene scritto 32 nella posizione di memoria 24100.
USR	10 USR (A)	Trasferisce il controllo alla posizione di memoria indicata dalla variabile A. Svolge lo stesso compito della chiamata delle subroutine in linguaggio assoluto. Richiede la conoscenza del linguaggio macchina ed è bene aver compreso esattamente il suo funzionamento prima di usarla.

	20 USR (24150)	Questa istruzione trasferisce il controllo al contenuto dell'indirizz 24150.
	570 USR ($C000)	Chiama il programma a cominciare dall'indirizzo $C000.
	770 USR (AD, DA$)	Quando un dato di stringa viene dato insieme al dato d'indirizzc questa funzione URS mette il primo indirizzo della memoria cor tenente il dato di stringa DA$ nel registro DE del CPU e la lun ghezza di DA$ nel registro BC prima dell'esecuzione dell'istruzion di CALL.

1.5.12 Istruzioni di controllo stampa

PRINT/P		Esegue pressocché la stessa operazione dello statement PRINT su printer opzionale (MZ-80P4, P5 oppure P6).
	10 PRINT/P A, A$	Stampa il valore numerico di e la variabile A di stringa di carattere A$ sul printer lineare.
	20 PRINT/T CHR$ (5)	Esegue il rifornimento della carta (CHR$(5) è un codice di controllo).
	30 PRINT/P CHR$ (18)	Mette in opera il modo di stampa per i caratteri ampliati (CHR$ (18) è un altro codice di controllo).
COPY/P	40 COPY/P 1	Fa copiare i caratteri mostrati al printer.
PAGE/P	100 PAGE/P 20	Specifica 20 linee che debbono esserecontenute in una pagina del printer lineare.

1.5.13 Istruzioni di input/output delle interface

INT		Legge i dati del porto I/O specificato.
	10 INP @12, A 20 PRINT A	Lo statement alla riga numero 10 legge i dati al porto I/O 12.
OUT		Emette i dati al porto I/O specificato.
	30 B = ASC ("A") 40 OUT @13, B	Lo statement alla riga numero 40 esegue l'output del codice ASCII per il carattere "A" al porto I/O 13.

1.5.14 Funzione arithmetiche

ABS	100 A = ABS (X)	Mette nella variabile A il valore assoluto della variabile X. Il termine contenuto tra le parentesi può essere una costante od una equazione. Esempio: ABS (–3) = 3 ABS (12) = 12
SGN	100 A = SGN (X)	Mette nella variabile A –1 se il valore della variabile X è negativo, 0 se X = 0 e 1 se X è positivo. Il termine tra parentesi può essere una costante od una equazione. Esempio: SGN (0.4) = 1 SGN (0) = 0 SGN (–400) = –1
INT	100 A = INT (X)	Determina il massimo valore intero che non supera il valore della variabile X e lo mette nella variabile A. Il termine tra parentesi può essere una costante od una equazione. Esempio: INT (3.87) = 3 INT (0.6) = 0 INT (–3.87) = –4

SIN	100 A = SIN (X)	Determina e mette nella variabile A il valore di sin X (con X in radianti). Il termine tra parentesi può essere una costante od una equazione. Dato che il rapporto tra radianti e gradi è il seguente: 1 grado = $\frac{\pi}{180}$ radianti
	110 A = SIN (30∗π/180)	Il calcolo del seno di 30 gradi andrebbe effettuato come indicato nella riga 110.
COS	200 A = COS (X)	Determina e mette nella variabile A il valore di cos X (con X in radianti). Il termine tra parentesi può essere una costante o una equazione.
	210 A = COS (200∗π/180)	Il calcolo di valori in gradi può venire effettuato in maniera identica a quella indicata per la funzione SIN. La riga 210 calcola e mette in A il valore di cos 200 gradi.
TAN	300 A = TAN (X)	Determina e mette nella variabile A il valore di tan X (con X in radianti). Il termine tra parentesi può essere una costante od una equazione.
	310 A = TAN (Y∗π/180)	Il calcolo dei valori in gradi può venire effettuato in maniera identica a quella indicata per la funzione SIN. La riga 310 è un'istruzione per calcolare e mettere nella variabile A il valore della tangente di Y gradi.
ATN	400 A = ATN (X)	Determina e mette nella variabile A il valore di $\tan^{-1} X$ (in radianti).
	410 A = 180/π∗ATN (X)	Il termine tra parentesi può essere una costante od una equazione. Il risultato del calcolo è il valore compreso tra $\pi/2$ e $-\pi/2$. L'istruzione 410 converte il valore di $\tan^{-1} X$ in gradi e lo mette nella variabile A.
SQR	100 A = SQR (X)	Determina e mette nella variabile A il valore della radice quadrata di X. Il termine tra parentesi può essere una costante od una equazione, ma il valore deve essere positivo o uguale a 0.
EXP	100 A = EXP (X)	Determina e mette nella variabile A il valore di e^x (e elevato a potenza x). Il termine tra parentesi può essere una costante od una equazione.
LOG	100 A = LOG (X)	Determina e mette nella variabile A il valore del logaritmo in base 10 di X $\log_{10} X$. Il termine tra parentesi può essere una costante od una equazione, ma deve avere un valore positivo.
LN	100 A = LN (X)	Determina e mette nella variabile A il valore del logaritmo naturale di X $\log_e X$. Il termine tra parentesi può essere una costante od una equazione, ma il valore deve essere positivo.
	110 A = LOG (X)/LOG (Y) 120 A = LN (X)/LN (Y)	Per determinare il logaritmo in base Y di X, $\log_Y X$, si possono usare le istruzioni contenute alle righe 110 o 120.
RND		Questa funzione genera numeri a caso con valori compresi tra 0,00000001 e 0,99999999. Vi sono due modi di usarla: un primo che usa tra parentesi lo 0 o un numero negativo, l'altro con un numero positivo tra parentesi.
	100 A = RND (0) 110 B = RND (−3)	Quando tra le parentesi viene messo un numero negativo o lo 0, come nelle righe 100 e 110, la generazione dei numeri casuali viene inizializzata per generare un valore specifico ogni volta e in A e B viene messo lo stesso valore.

	200 A = RND (1) 210 B = RND (10)	Quando nelle parentesi viene indicato un numero positivo, come nelle righe 200 e 210, viene generato un numero a caso con valore compreso tra 0,00000001 e 0,99999999, ogni volta che viene usata la funzione RND. In tal caso però, tale fatto non ha nulla a che fare con il numero positivo che è scritto tra le parentesi.

1.5.15 Funzioni di caratteri

LEFT$	300 A$ = LEFT$ (X$, N)	Sostituisce i primi N caratteri della variabile di stringa X$ nella variabile A$. N può essere una costante, una variabile od un'equazione.
RIGHT$	600 B$ = RIGHT$ (X$, N)	Sostituisce gli ultimi N caratteri della variabile X$ nella variabile di stringa B$. N può essere una costante, una variabile od un'equazione.
MID$	900 C$ = MID$ (X$, M, N)	Sostituisce N caratteri a partire dal carattere Msimo della variabile X$ nella variabile di stringa C$. M ed N possono essere delle costanti, delle variabili o delle equazioni.
LEN	100 LX = LEN (X$)	Mette nella variabile LX la lunghezza (numero di caratteri) della variabile di stringa X$.
	110 LS = LEN (X$ + Y$)	Mette nella variabile LS la somma delle lughezze delle variabili X$ ed Y$.
ASC	200 A = ASC (X$)	Mette nella variabile A il valore del codice ASCII (decimale) corrispondente al primo carattere della variabile di stringa X$.
CHR$	220 X1$ = CHR$ (A)	All'inverso dell'istruzione ASC, questa mette nella variabile X1$ il codice ASCII corrispondente al contenuto della variabile A. A può essere una costante, una variabile od un'equazione.
VAL	400 K = VAL (N$)	Mette come valore della variabile K une stringa numerica equivalente alla variabile di stringa N$.
STR$	440 N$ = STR$ (K)	All'inverso dell'istruzione VAL, questa istruzione mette nella variabile di stringa N$ il valore della variabile K come se fosse una stringa.
SAPCE$	40 D$ = SPACE$ (N)	Sostituisce N spazi nella variabile di stringa D$.
STRING$	50 E$ = STRING$ (" * ", 10)	Sostituisce le diece ripetizioni di " * " nella variabile di stringa E$.

1.5.16 Funzione di tabulazione

TAB	10 PRINT TAB (X) ; A	Sposta sullo schermo il cursore di X caratteri da sinistra e visualizza il valore della variabile A.
	20 PRINT TAB (5) ; A$	Sposta sullo schermo il cursore di 5 caratteri verso sinistra e visualizza la serie di caratteri della variabile di stringa A$.

1.5.17 Operatori arithmetici

I numeri bianchi su sfondo nero sulla sinistra indicano l'ordine di priorità delle operazioni. Le operazioni tra parentesi hanno ulteriore priorità.

=	10 A = X + 3 (Sostituzione)	Sostituisce nella variabile A il valore di 3 + il valore della variabile X.
	20 B = π	Sostituisce π (3,1415927) come valore della variabile B.
❶ ↑	10 A = X↑Y (potenza)	Calcola X^Y e mette il risultato nella variabile A. (Nei caso che X sia un valore negativo ed Y non sia un intero si verifica un errore).
❷ –	10 A = –B (Segno meno)	Fare attenzione che 0 – B è una sottrazione, ma il segno "–" di –B è un segno negativo.
❸ *	10 A = X*Y (Moltiplicazione)	Mette nella variabile A il risultato del prodotto di X per Y.
❸ /	10 A = X/Y (Divisione)	Mette nella variabile A il quoziente tra i valori di X ed Y.
❹ +	10 A = X + Y (Somma)	Mette nella variabile A il risultato della somma di X ed Y.
❹ –	10 A = X – Y (Sottrazione)	Mette nella variabile A il risultato della sottrazione tra X ed Y.

1.5.18 Operatori logici

=	10 IF A = X THEN . . .	Esegue le istruzioni dopo THEN se il valore della variabile A e quello di X sono uguali.
	20 IF A$ = "XYZ" THEN . . . (Uguale)	Esegue le istruzioni dopo THEN se il contenuto della variabile di stringa A$ è uguale a quello della stringa "XYZ".
<> ><	10 IF A <> X THEN . . . (Non uguale)	Esegue le istruzioni dopo THEN se il valore della variabile A e quello di X non sono uguali.
>= o =>	10 IF A >= X THEN . . . (Maggiore o uguale)	Esegue le istruzioni dopo THEN se il valore della variabile A è maggiore o uguale a quello di X.
<= o =<	10 IF A <= X THEN . . . (Minore o uguale)	Esegue le istruzioni dopo THEN se il valore della variabile A è minore o uguale a quello di X.
*	10 IF (A>X) * (B>Y) THEN . . . (Moltiplicazione Logica)	Esegue le istruzioni dopo THEN se il valore della variabile A è maggiore di quello di X e quello della variabile B è maggiore del valore di Y.
+	10 IF (A>X) + (B>Y) THEN . . . (Somma Logica)	Esegue le istruzioni dopo THEN se il valore della variabile A è maggiore di quello di X o quello della variabile B è maggiore del valore di Y.

1.5.19 Altri simboli

?	200? "A + B"; A + B 210 PRINT "A + B"; A + B	Questa forma può venire usata in sostituzione dell'istruzione PRINT. Le righe 200 e 210 sono quindi identiche.
:	220 A = X : B = X↑2: PRINT A, B	Il simbolo rappresenta una pausa nella riga ed è usato per righe che contengono più di una istruzione. La riga 220 comprende 3 istruzioni.
;	230 PRINT "AB" ; "CD" ; "E"	Esegue in continuità l'istruzione PRINT. In seguito alla riga 230, sullo schermo vengono visualizzati "ABCDE" senza spazi intermedi.
	240 INPUT "X = " ; X$	Visualizza sullo schermo "X = " e attende che vengano immessi i valori della variabile di stringa X$.
,	250 PRINT "AB", "CD", "E"	Esegue l'istruzione PRINT con tabulazione. La riga 250 fa sì che sullo schermo venga visualizzato dapprima AB, poi CD in una posizione 10 caratteri sulla destra di A e finalmente E in una posizione 10 caratteri sulla destra di C.
	260 DIM A (20), B (30)	Esempio in cui la virgola viene usata come separatore delle variabili.
" "	270 A$ = "SHARP BASIC" 280 PRINT " * " ; A$; " * "	Indica che una stringa è contenuta tra virgolette " ".
$	290 A1$ = LEFT$ (A$, 5) 300 A2$ = "MZ-80A"	Indica una variabile di stringa.
	350 LIMIT $A000	Indica che il dato numerico che segue il segno del dollaro è rappresentato nella notazione esadecimale.
π	400 S = SIN (X * π/180)	La costante 3,1415927 è indicata da π.

1.5.20 Lista dei numeri d'errore del BASIC SA-5510

Errore No.	Significato
1	Errore sintattico
2	Eccedenza nel risultato d'operazione
3	Dato non valido
4	Dati non dello stesso tipo
5	La lunghezza della stringa eccede 255 caratteri
6	Insufficiente capacità di memoria
7	Il formato dell'insieme definito è più grande di quello definito precedentemente
8	La lunghezza di una linea del testo BASIC è eccessiva
9	
10	Il numero dei livelli di loop GOSUB eccede il numero di 16.
11	Il numero dei livelli di loop FOR-NEXT eccede il numero di 16.
12	Il numero dei livelli di funzione eccede 6.
13	NEXT è stato usato senza il corrispondente FOR.
14	RETURN è stato usato senza il corrispondente GOSUB.
15	È stata usata una funzione indefinita.
16	È stato usato un numero di linea non utilizzato.
17	Il comando CONT non può essere eseguito.
18	Uno statement di scrittura è stato emesso alla zona di controllo BASIC.
19	I comandi del modo diretto e gli statement sono mischiati.
20	
21	
22	
23	
24	Uno statement READ è stato usato senza il corrispondente statement DATA.
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	

Errore No.	Significato
36	
37	
38	
39	
40	
41	
42	
43	Lo statement OPEN (ROPEN oppure WOPEN) viene emesso per un file che è già aperto.
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	Non c'è file
64	
65	Il printer non è pronto.
66	Si ha un errore nell'hardware del printer.
67	Non c'è carta.
68	
69	
70	Controllo errore di somma.

1.6 Come si ottengono le copie di nastri BASIC

Il nastro BASIC non viene scambiato con uno nuovo dopo l'acquisto. Si raccomanda di fare delle copie del nastro originale BASIC usando il procedimento qui descritto e di usarlo al posto del nastro originale. Riporre il nastro in un luogo sucuro.

Attivare l'interpreter BASIC per mezzo del nastro originale BASIC. Eseguire i comandi seguenti:

USR ($11FD)

appare il messaggio " ↓ RECORD. PLAY" sullo schermo. Mettere una nuova cassetta nella piastra e premere il pulsanti RECORD e PLAY.

A completamento della scrittura, si ottiene una copia del nastro BASIC. È possibile ottenere un numero qualsiasi di copie seguendo questa procedura, però non è possibile ottenere copie da un nastro BASIC copiato.

Monitor Program dell'MZ-80A

Capitolo 2

2.1 Funzione del monitor program e sotto routine

Un monitor program generalmente controlla programmi di sistema quali l'interpreter BASIC. L'MZ-80A usa un monitor program chiamato MONITOR SA-1510. Esso comprende varie sotto routine funzionali che controllano la tastiera, il display, il circuito sonoro, la piastra di registrazione magnetica a cassette, ecc. Queste sotto routine sono chiamate dall'interpreter BASIC quando questi esegue statement INPUT, comandi SAVE, statement MUSIC oppure altri comandi o statement. Le sotto routine di monitor possono anche essere richiamate dall'utente a volontà.

Il MONITOR SA-1510 occupa, 4 K byte della memoria (MONITOR ROM) ed è registrato negli indirizzi di memoria da $0000 a $0FFF.

2.1.1 Uso dei comandi di monitor

Sotto viene mostrato il messaggio che appare quando si accende l'MZ-80A.

```
* * MONITOR SA-1510 * *
*
```

Il cursore lampeggia per informare l'operatore che il controllo del sistema è al livello di comando Monitor. I comandi Monitor sono i seguenti:

L [CR] . . . . . Esegue il load (caricamento) dal file di cassetta nella memoria.

J xxxx [CR] . . . . . Trasferisce il controllo di sistema agli indirizzi specificati, cioè, carica gli indirizzi specificati nel program counter del CPU.
xxxx : 4 cifre di numero esadecimale.
Gli indirizi d'inizio per il BASIC SA-5510 sono i seguenti:
Indirizzo d'inizio a calco = $1250
Indirizzo d'inizio a freddo = $1200

F [CR] . . . . . Trasferisce il controllo di sistema alla routine di controllo del drive del floppy disk che si trova sulla interfaccia del drive del floppy disk.

B [CR] . . . . . Mette a punto med azzera alternativamente il suono di registrazione di tasto.

2.1.2 La sotto routine monitor

Le sotto routine MONITOR SA-1510 sono elencate nella figura 2.1. I nomi delle sotto routine indicate sono gli stessi delle label mostrate nel Monito Program Assembly Listing. Ogni nome è una rappresentazione mnemonica della funzione di sotto routine.

Tavola 2.1 Lista della sotto routine Monitor.

Nome della sottoroutine (indirizzo esadecimale)	Funzione	Regist riserv
CALL LETNL ($0006)	Per cambiare la riga e per la messa a punto del cursore all'inizio della riga seguente.	Tutti i reg eccetto AI
CALL NL ($0009)	Cambia la riga e mette a punto il cursore all'inizio se non è già in posizione.	Tutti i reg eccetto AI
CALL PRNTS ($000C)	Mostra uno spazio solo nella posizione del cursore sullo schermo.	Tutti i regi eccetto AI
CALL PRNT ($0012)	Mostra il carattere corrispondente al codice ASCII memorizzato nel registro A nell'attuale posizione del cursore.	Tutti i regi eccetto AI
CALL MSG ($0015)	Tiene i dati nel registro A come codice ASCII e lo mostra sullo schermo, ad iniziare dalla posizione nella quale si trova il cursore. Però un ritorno di carrello viene eseguito per 0DH e le varie operazioni di controllo del cursore sono eseguite per 11H-16H quando sono inclusi.	Tutti i regi
CALL MSGX ($0018)	Praticamente lo stesso di MSG, con la sola sifferenza che i codici di controllo del cursore sono per i caratteri inversi.	Tutti i regi
CALL BELL ($003E)	Emette un suono momentaneo (circa 880 Hz).	Tutti i regi eccetto AF
CALL MELDY ($0030)	Eseque la musica a seconda dei dati musicali. L'indirizzo d'inizio dei dati musicali deve essere specificato in anticipo nel registro DE. In quanto nel BASIC, l'intervallo musicale e la durate delle note dei dati musicali sono espressi in questo ordine per mezzo del codice ASCII. Il segno della fine deve essere o 0DH oppure C8H " ■ ". La musica finisce se il flag C è 0 quando si effettua il ritorno; se il flag C è 1 indica che sono stati premuti i tasti [SHIFT] + [BREAK].	Tutti i regi eccetto AF
CALL XTEMP ($0041)	Mette a punto il tempo musicale. I dati del tempo (da 01 a 07) viene impostato e chiamato dal registro A. 01 : il più lento 04 : medio 07 : il più svelto È necessario fare attenzione a che il dato di tempo sia registrato nel registro A in codice binario e non nel codice ASCII corrispondenteai numeri dall'1 al 7 (31H fino a 37H).	Tutti i regi
CALL MSTA ($0044)	Suona una nuota continuamente a seconda del fattore divisione specificato. Il fattore di divizione *nn'* consiste di due byte di dati; *n'* viene registrato nell'indirizzo 11A1H e *n* viene registrato nell'indirizzo 11A2H. La relazione tra il fattore di divisione e la frequenza prodotta è 2MHz/*nn'*.	Solo BC e I

<table>
<tr><th>Nome della sottoroutine (indirizzo esadecimale)</th><th>Funzione</th><th>Registri riservati</th></tr>
<tr><td>CALL MSTP
($0047)</td><td>Interrompe il tono che sta suonando.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL TIMST
($0033)</td><td>Mette a punto l'orologio incorporato (l'orologio viene attivato da questa chiamata). Le condizioni di chiamata sono:
Registro A ← 0 (AM), Registro A ← 1 (PM)
Registro DE ← l'ora in secondi (2 byte).</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL TIMRD
($003B)</td><td>Legge l'orologio incorporato. I dati orari sono ordinati nel modo seguente:
Registro A ← 0 (AM), Registro A ← 1 (PM)
Registro DE ← l'ora in secondi (2 byte).</td><td>Tutti i registri eccetto AF e DE</td></tr>
<tr><td>CALL BRKEY
($001E)</td><td>Controlla se [SHIFT] + [BREAK] erano stati premuti. Il flag Z viene messo a punto se erano stati premuti altrimenti il flag Z viene azzerato.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL GETL
($0003)</td><td>Esegue l'input di una riga registrata dalla tastiera. L'indirizzo d'inizio dove i dati debbono essere registrati deve essere specificato in precedenza nel registro DE. [CR] funziona come segno di fine. 80 è il numero massimo di caratteri che possono essere registrati (compreso il segno di fine 0DH).
L'input dei tasti viene mostrato sullo schermo e viene accettato anche il controllo del cursore. Il codice BREAK (1BH) seguito dal ritorno del carrello (0DH) viene posto all'inizio dell'indirizzo specificato nel registro DE quando [SHIFT] + [BREAK] sono premuti.</td><td>Tutti i registri</td></tr>
<tr><td>CALL GETKY
($001B)</td><td>Prende un carattere solo dentro il registro A dalla tastiera nel codice ASCII. Il ritorno viene fatto dopo che 00 viene posto nel registro A se non vengono premuti altri tasti quando viene eseguita una sotto routine. Però l'input dei tasti non viene mostrato sullo schermo.
I codici che sono presi nel registro A quando questi tasti speciali sono premuti sono mostrati sotto.
<table>
<tr><th>Tasti speciali</th><th>Codici presi nel registro A</th></tr>
<tr><td>[DEL]</td><td>60 H</td></tr>
<tr><td>[INST]</td><td>61 H</td></tr>
<tr><td>[GRPH] Modo grafico</td><td>62 H</td></tr>
<tr><td>[GRPH] Modo normale</td><td>63 H</td></tr>
<tr><td>[BREAK]</td><td>64 H</td></tr>
<tr><td>[CR] or [ENT]</td><td>66 H</td></tr>
<tr><td>[CTRL] + [A] ~ [Z]</td><td>01 H ~ 1 AH</td></tr>
<tr><td>[CTRL] + [[]</td><td>1 BH</td></tr>
<tr><td>[CTRL] + [\]</td><td>1 CH</td></tr>
<tr><td>[CTRL] + []]</td><td>1 DH</td></tr>
<tr><td>[CTRL] + [^]</td><td>1 EH</td></tr>
<tr><td>[CTRL] + [—]</td><td>1 FH</td></tr>
</table>
</td><td>Tutti i registri eccetto AF</td></tr>
</table>

<table>
<tr><th>Nome della sottoroutine (indirizzo esadecimale)</th><th>Funzione</th><th>Registri reservati</th></tr>
<tr><td>CALL PRTHL
($03BA)</td><td>Mostra il contenuto del registro HL sullo schermo come numero esadecimale di quattro ciffre.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL PRTHX
($03C3)</td><td>Mostra sullo schermo il contenuto del registro A come numero esadecimale di due cifre.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL ASC
($03DA)</td><td>Converte il contenuro dei quattro bit inferiori del registro A dalla notazione esadecimale alla rappresentazione in codice ASCII e ritorna dopo la messa a punto dei dati convertiti nel registro A.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL HEX
($03F9)</td><td>Converte gli 8 bit del registro A dal codice ASCII alla notazionne esadecimale e ritorna dopo la messa a punto dei dati convertiti nei 4 bit inferiori del registro A.
Quando il flag C = 0 fino al ritorno registro A ← esadecimale.
Quando il flag C = 1 fino al ritorno registro A non sicuro.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL HLHEX
($0410)</td><td>Possiede una stringa consecutiva di 4 caratteri in codice ASCII come stringa di dati esadecimale e ritorna dopo la messa a punto dei dati nel registro HL. Le condizioni di chiamata e di ritorno sono come di seguito.
DE ← indirizzo d'inizio della stringa ASCII (stringa “3” “1” “A” “5”)
↑—DE
CALL HLHEX
Flag C = 0 HL ← numero esadecimale (cioè HL = 31A5H)
Flag C = 1 HL non è assicurato.</td><td>Tutti i registri eccetto AF e HL</td></tr>
<tr><td>CALL 2HEX
($041F)</td><td>Possiede 2 stringhe consecutive ASCII come stringhe esadecimali e ritorna dopo la messa a punto dei dati nel registrao A. Le condizioni di chiamata e ritorno sono le seguenti.
DE ← indirizzo d'inizio della stringa ASCII (cioè “3” “A”)
↑—DE
CALL 2HEX
Flag C = 0 registro A ← numero esadecimale (cioè Registro A=3AH)
Flag C = 1 il registro A non è assicurato.</td><td>Tutti i registri eccetto AF e DE</td></tr>
<tr><td>CALL ??KEY
($09B3)</td><td>Attende la registrazione di tasti mentre il cursore lampeggia. Quando viene effettuata la registrazione di un tasto viene convertita nel codice di schermo e messa a punto nel registro A, poi viene effettuato un ritorno.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL ?ADCN
($0BB9)</td><td>Converte un valore ASCII nel codice di schermo. Le condizioni di chiamata e di ritorno sono le seguenti.
Registro A ← Valore ASCII
CALL ?ADCN
Registro A ← Codice di schermo.</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL ?DACN
($0BCE)</td><td>Converte un codice di schermo in un valore ASCII. Le condizioni di chiamata e di ritorno sono le seguenti.
Registro A ← Codice di shermo
CALL ?DACN
Registro A ← Valore ASCII</td><td>Tutti i registri eccetto AF</td></tr>
<tr><td>CALL ?DPCT
($0DDC)</td><td>Controlla il display sullo schermo. La relazione tra il registro A ed il tempo di chiamata e controllo è il seguente.
<table>
<tr><th>Registro A</th><th>Stessa funzione</th><th>Registro A</th><th>Stessa funzione</th></tr>
<tr><td>C0H
C1H
C2H
C3H
C4H
C5H
C6H
C7H</td><td>Scrolling
[↓]
[↑]
[→]
[←]
[HOME]
[CLR]
[DEL]</td><td>C8H
C9H
CAH
CCH
CDH
CEH
CFH</td><td>[INST]
[GRPH] (graphic → normal)
[GRPH] (normal → graphic)
[CTRL] + [@] (normal ↔ reverse)
[CR] or [ENT]
[CTRL] + [D] (roll up)
[CTRL] + [E] (roll down)</td></tr>
</table></td><td>Tutti i registri</td></tr>
</table>

Nome della sottoroutine (indirizzo esadecimale)	Funzione	Registri reservati
CALL ?BLNK ($0DA6)	Controlla lo spazio verticale dello schermo. Attende fino a che non inizia l'intervallo di spaziatura verticale e ritorno dove ha posto la spaziatura.	Tutti i registri

CALL ?PONT ($0FB1)	Mette a punto la posizione corrente del cursore sullo schermo nel registro HL. Le condizioni per il ritorno sono le seguenti.
	CALL ?PONT
	HL ← posizione del cursore sullo schermo (indirizzo V-RAM)
	(Nota) Le coordinate X-Y del cursore sono coordinate DSPXY (1171H). La posizione corrente del cursore viene caricata nel modo seguente.
	LD HL, (DSPXY) ; H ← Y corrdinate sullo schermo
	L ← X coordinate sullo schermo
	La posizione del cursore viene messa a punto come segue.
	LD (DSPXY), HL
		Tutti i registri eccetto AF e HL

CALL WRINF ($0021)	Scrive il contenuto corrente di certe parti del registro (descritto oltre) nel nastro, ad iniziare dalla posizione dove si trova il nastro. Condizioni di ritorno Flag C = 0 non si è avuto errore Flag C = 1 Il tasto [BREAK] è stato premuto.	Tutti i registri eccetto AF
CALL WRDAT ($0024)	Scrive il contenuto corrento della zona di memoria nel nastro. Condizioni di ritorno Flag C = 0 non si è avuto errore Flag C = 1 Il tasto [BREAK] è stato premuto.	Tutti i registri eccetto AF
CALL RDINF ($0027)	Legge il primo inizializatore CMT trovato a cominciare dalla posizione nella quale si trova il nastro in certe parti del registro inizializatore. Condizioni di ritorno Flag C = 0 — Non si è avuto errore. Flag C = 1, registro A = 1 — Si è avuto errore di controllo di somma Flag C = 1, registro A = 2 — È stato premuto il tasto [BREAK].	Tutti i registri eccetto AF
CALL RDDAT ($002A)	Legge il blocco di dati nel CMT (registratore) a seconda del contenuto corrente di certe parti del registro inizializatore. Condizioni di ritorno Flag C = 0 — Non si è avuto errore Flag C = 1, registro A = 1 — Si è avuto errore di controllo di somma Flag C = 1, registro A = 2 — È stato premuto il tasto [BREAK].	Tutti i registri eccetto AF
CALL VERFY ($002D)	Compara il primo file CMT trovato dopo la posizione attuale del nastro con il contenuto della zona di memoria indicata dall'inizializzatore. Condizioni di ritorno Flag C = 0 — Non si è avuto errore Flag C = 1, registro A = 1 — Non si è ottenuta compatibilità Flag C = 1, registro A = 2 — Il tasto [BREAK] è stato premuto.	Tutti i registri eccetto AF

(Nota) Il contenuto del registro inizializzatore agli indirizzi specificati sono come di seguito. Il registro inizia all'indirizzo $10F0 e consiste di 116 byte.

Indirizzo	Contenuto
IBUFE ($10F0)	Questo byte indica uno dei seguenti modi di file. 01. File oggetto (programma di linguaggio macchina) 02. File di testo BASIC 03. File di dati BASIC 04. File di sorgente (file ASCII) 05. File ricollocabile (file binario ricollocabile) A0. File di text interpreter PASCAL A1. File dati interpreter PASCAL
IBU1 ($10F1~$1101)	Questi 17 byte indicano il nome del file. Però siccome ODH è usato come segno di fine in realtà il nome di file è limitato a 16 byte. Esempio: S A M P L E 0D
IBU18 ($1102~$1103)	Questi due byte indicano la misura di byte del blocco di dati che segue.
IBU20 ($1104~$1105)	Questi due byte indicano l'indirizzo di dati del blocco di dati che segue. L'indirizzo di caricamento del blocco di dati che segue viene indicato da "CALL RDDAT". L'indirizzo iniziale della zona di memoria che deve essere posta in output come blocco di dati viene indicata da "CALL WRDAT".
IBU22 ($1106~$1107)	Questi due byte indicano l'indirizzo di esecuzione del blocco di dati che segue.
IBU24 ($1108~$1163)	Questi byte sono usati per informazioni supplementari, come i commenti.

Esempio

Indirizzo	Contenuto	
10F0	01	; indica un file oggetto (programma di linguaggio macchina)
10F1	'S'	; Il nome del file è "SAMPLE"
10F2	'A'	
10F3	'M'	
10F4	'P'	
10F5	'L'	
10F6	'E'	
10F7	0D	
10F8 ~ 1101	Variabile	
1102	00	; la misura del file è di 2000H byte.
1103	20	
1104	00	; l'indirizzo di dati del file è 1200H.
1105	12	
1106	50	; l'indirizzo di esecuzione del file è 1250H.
1107	12	

2.2 Lista d'assemblaggio del MONITOR SA-1510

La lista d'assemblaggio del MONITOR SA-1510 è mostrata sotto.
Questa lista d'assemblaggio è stata ottenuta con il macro assembler Z80. Il significato di ogni colonna è il seguente.

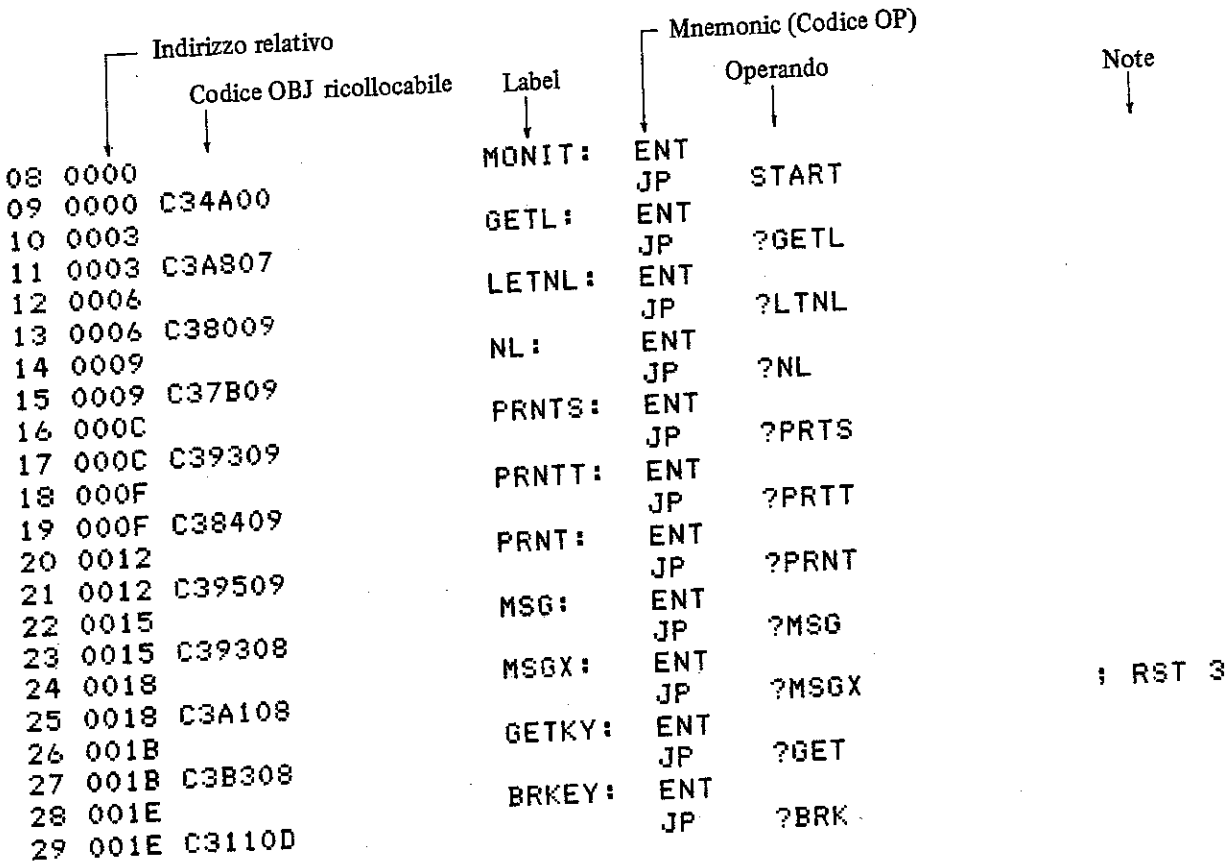


Figura 2.1

Siccome il primo indirizzo del MONITOR SA-1510 è $0000, gli indirizzi relativi ed i codici OBJ ricollocabili p
sono essere considerati come indirizzi assoluti e codici OBJ senza interpretazione.

Questa lista d'assemblaggio è solo per riferimento. La Sharp Corporation non è obbligata a rispondere a domar
riguardanti il contenuto di questo programma.

Questa lista d'assemblaggio è stata stampata con il printer opzionale MZ-80P5.

```
**  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 01          09/04/81

0000                    ;
0000                    ;
0000                    ;       MONITOR PROGRAM
0000                    ;           (MZ-80A)
0000                    ;
0000                    ;                SA-1510
0000                    ;             REV. '81.8.26
0000                    ;
0000 C34A00             MONIT:  ENT
0003                            JP      START
0003 C3A807             GETL:   ENT
0006                            JP      ?GETL
0006 C38009             LETNL:  ENT
0009                            JP      ?LTNL
0009 C37B09             NL:     ENT
000C                            JP      ?NL
000C C39309             PRNTS:  ENT
000F                            JP      ?PRTS
000F C38409             PRNTT:  ENT
0012                            JP      ?PRTT
0012 C39509             PRNT:   ENT
0015                            JP      ?PRNT
0015 C39308             MSG:    ENT
0018                            JP      ?MSG
0018 C3A108             MSGX:   ENT
001B                            JP      ?MSGX           ; RST 3
001B C3B308             GETKY:  ENT
001E                            JP      ?GET
001E C3110D             BRKEY:  ENT
0021                            JP      ?BRK
0021 C33604             WRINF:  ENT
0024                            JP      ?WRI
0024 C37004             WRDAT:  ENT
0027                            JP      ?WRD
0027 C3CF04             RDINF:  ENT
002A                            JP      ?RDI
002A C3EF04             RDDAT:  ENT
002D                            JP      ?RDD
002D C37505             VERFY:  ENT
0030                            JP      ?VRFY
0030 C38801             MELDY:  ENT
0033                            JP      ?MLDY           ; RST 6
0033 C3FA02             TIMST:  ENT
0036 00                         JP      ?TMST
0037 00                         NOP
0038 C33810                     NOP
003B                            JP      1038H           ; INTERRUPT ROUTINE
003B C34403             TIMRD:  ENT
003E                            JP      ?TMRD
003E C3E502             BELL:   ENT
0041                            JP      ?BEL
0041 C3EC02             XTEMP:  ENT
0044                            JP      ?TEMP
0044 C3AB02             MSTA:   ENT
0047                            JP      MLDST
0047 C3BE02             MSTP:   ENT
004A                            JP      MLDSP
004A                    ;
004A                    ;;;;;
004A                    ;
004A 31F010             START:  LD      SP,SP
```

```
**  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 02          09/04/81

004D ED56                       IM      1               ; IM 1 SET
004F CD4D07                     CALL    ?MODE
0052 06FF                       LD      B,FFH           ; BUFFER CLEAR
0054 21F110                     LD      HL,NAME
0057 CDD80F                     CALL    ?CLER
005A 3E16                       LD      A,16H           ; LASTER CLR.
005C CD1200                     CALL    PRNT
005F 3ECF                       LD      A,CFH
0061 2100D8                     LD      HL,D800H
0064 1803                       JR      +5
0066 C33510                     JP      1035H           ; JUMP NMI
0069 CDE309                     CALL    #CLR8
006C 217903                     LD      HL,TIMIN        ; INTERUPT
006F 3EC3                       LD      A,C3H
0071 323810                     LD      (1038H),A
0074 223910                     LD      (1039H),HL
0077 3E04                       LD      A,04
0079 329E11                     LD      (TEMPW),A
007C CDBE02                     CALL    MLDSP
007F CD0900                     CALL    NL
0082 110001                     LD      DE,MSG?3
0085 DF                         RST     3
0086 CDE502                     CALL    ?BEL
0089 3EFF               SS:     LD      A,FFH
008B 329D11                     LD      (SWRK),A        ; KEY IN SILENT
008E 2100E8                     LD      HL,E800H        ; USR ROM ?
0091 3655                       LD      (HL),55H
0093 1835                       JR      FD2
0095 CD0900             ST1:    CALL    NL
0098 3E2A                       LD      A,2AH           ; '*' PRINT
009A CD1200                     CALL    PRNT
009D 11A311                     LD      DE,BUFER
00A0 CD0300                     CALL    GETL
00A3 1A                 ST2:    LD      A,(DE)
00A4 13                         INC     DE
00A5 FE0D                       CP      0DH
00A7 28EC                       JR      Z,ST1
00A9 FE4A                       CP      'J'             ; JUMP
00AB 280E                       JR      Z,GOTO
00AD FE4C                       CP      'L'             ; LOAD PROGRAM
00AF 2828                       JR      Z,LOAD
00B1 FE46                       CP      'F'             ; FLOPPY ACCESS
00B3 2812                       JR      Z,FD
00B5 FE42                       CP      'B'             ; KEY IN BELL
00B7 2808                       JR      Z,SG
00B9 18E8                       JR      ST2
00BB                    ;
00BB                    ;       JUMP COMMAND
00BB                    ;
00BB CD1004             GOTO:   CALL    HLHEX
00BE 38D5                       JR      C,ST1
00C0 E9                         JP      (HL)
00C1                    ;
00C1                    ;  KEY SOUND ON OFF
00C1                    ;
00C1 3A9D11             SG:     LD      A,(SWRK)
00C4 2F                         CPL
00C5 18C4                       JR      SS+2
00C7                    ;
00C7                    ;  FLOPPY
```

```
   **  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 03          09/04/81

01 00C7                    ;
02 00C7 2100F0             FD:     LD    HL,F000H
03 00CA 7E                 FD2:    LD    A,(HL)
04 00CB B7                         OR    A
05 00CC 20C7                       JR    NZ,ST1
06 00CE E9                 FD1:    JP    (HL)
07 00CF                    ;
08 00CF                    ;
09 00CF                    ;   ERROR (LOADING)
10 00CF                    ;
11 00CF FE02               ?ER:    CP    02H
12 00D1 28C2                       JR    Z,ST1
13 00D3 111801                     LD    DE,MSGE1
14 00D6 DF                         RST   3                 ; CALL MSGX
15 00D7 18BC                       JR    ST1
16 00D9                    ;
17 00D9                    ;
18 00D9                    ;  LOAD COMMAND
19 00D9                    ;
20 00D9 CDCF04             LOAD:   CALL  ?RDI
21 00DC 38F1                       JR    C,?ER
22 00DE CD0900             LOA0:   CALL  NL
23 00E1 11F700                     LD    DE,MSG?2
24 00E4 DF                         RST   3                 ; CALL MSGX
25 00E5 11F110                     LD    DE,NAME
26 00E8 DF                         RST   3                 ; CALL MSGX
27 00E9 CDEF04                     CALL  ?RDD
28 00EC 38E1                       JR    C,?ER
29 00EE 2A0611                     LD    HL,(EXADR)
30 00F1 7C                         LD    A,H
31 00F2 FE12                       CP    12H
32 00F4 389F                       JR    C,ST1
33 00F6 E9                         JP    (HL)
34 00F7 4C                 MSG?2:  DEFM  'L'               ; LOADING
35 00F8 B7A1                       DEFW  A1B7H
36 00FA 9CA6                       DEFW  A69CH
37 00FC B097                       DEFW  97B0H
38 00FE 200D                       DEFW  0D20H
39 0100 2A2A2020           MSG?3:  DEFM  '**  MONITOR SA-1510  **'
40 0104 4D4F4E49
41 0108 544F5220
42 010C 53412D31
43 0110 35313020
44 0114 202A2A
45 0117 0D                         DEFB  0DH
46 0118                    ;
47 0118                    ;
48 0118 43                 MSGE1:  DEFM  'C'               ; CHECK SUM ERROR
49 0119 9892                       DEFW  9298H
50 011B 9FA9                       DEFW  A99FH
51 011D 20A4                       DEFW  A420H
52 011F A5B3                       DEFW  B3A5H
53 0121 2092                       DEFW  9220H
54 0123 9D9D                       DEFW  9D9DH
55 0125 B79D                       DEFW  9DB7H
56 0127 0D                         DEFB  0DH
57 0128                    ;
58 0128                    ;  CR PAGE MODE1
59 0128                    ;
60 0128 CD2B0A             .CR:    CALL  .MANG
```

```
   **  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 04          09/04/81

01 012B 0F                         RRCA
02 012C D2860E                     JP    NC,CURS2
03 012F 2E00                       LD    L,0
04 0131 24                         INC   H
05 0132 FE18                       CP    +24
06 0134 2804                       JR    Z,.CR1
07 0136 24                         INC   H
08 0137 C3660E                     JP    CURS1
09 013A 227111             .CR1:   LD    (DSPXY),HL
10 013D                    ;
11 013D                    ;   SCROL PAGE MODE1
12 013D                    ;
13 013D                    .SCROL: ENT
14 013D 01C003                     LD    BC,03C0H
15 0140 1100D0                     LD    DE,SCRN
16 0143 2128D0                     LD    HL,SCRN+40
17 0146 EDB0                       LDIR
18 0148 EB                         EX    DE,HL
19 0149 0628                       LD    B,+40
20 014B CDD80F                     CALL  ?CLER
21 014E 011A00                     LD    BC,26
22 0151 117311                     LD    DE,MANG
23 0154 217411                     LD    HL,MANG+1
24 0157 EDB0                       LDIR
25 0159 3600                       LD    (HL),0
26 015B 3A7311                     LD    A,(MANG)
27 015E B7                         OR    A
28 015F CAE50E                     JP    Z,?RSTR
29 0162 217211                     LD    HL,DSPXY+1
30 0165 35                         DEC   (HL)
31 0166 18D5                       JR    .SCROL
32 0168                    ;
33 0168                    ;
34 0168                    ;
35 0168                    ;   CTBL PAGE MODE1
36 0168                    ;
37 0168 3D01               .CTBL:  DEFW  .SCROL
38 016A 5D0E                       DEFW  CURSD
39 016C 6E0E                       DEFW  CURSU
40 016E 7B0E                       DEFW  CURSR
41 0170 950E                       DEFW  CURSL
42 0172 0904                       DEFW  HOM0
43 0174 B30E                       DEFW  CLRS
44 0176 F20E                       DEFW  DEL
45 0178 2D0F                       DEFW  INST
46 017A E10E                       DEFW  ALPHA
47 017C EE0E                       DEFW  KANA
48 017E E50E                       DEFW  ?RSTR
49 0180 170A                       DEFW  REV
50 0182 2801                       DEFW  .CR
51 0184 E50E                       DEFW  ?RSTR
52 0186 E50E                       DEFW  ?RSTR
53 0188                    ;
54 0188                    ;
55 0188                    ;
56 0188                    ;   MELODY
57 0188                    ;
58 0188                    ;    DE=DATA LOW ADR.
59 0188                    ;   EXIT  CF=1 BREAK
60 0188                    ;         CF=0 OK
```

```
0188                    ;
0188                    ?MLDY:  ENT
0188 C5                         PUSH    BC
0189 D5                         PUSH    DE
018A E5                         PUSH    HL
018B 3E02                       LD      A,02H
018D 32A011                     LD      (OCTV),A
0190 0601                       LD      B,01
0192 1A                 MLD1:   LD      A,(DE)
0193 FE0D                       CP      0DH             ; CR
0195 283B                       JR      Z,MLD4
0197 FEC8                       CP      0C8H            ; END MARK
0199 2837                       JR      Z,MLD4
019B FECF                       CP      0CFH            ; UNDER OCTAVE
019D 2827                       JR      Z,MLD2
019F FE2D                       CP      2DH             ; '-'
01A1 2823                       JR      Z,MLD2
01A3 FE2B                       CP      2BH             ; '+'
01A5 2827                       JR      Z,MLD3
01A7 FED7                       CP      0D7H            ; UPPER OCTAVE
01A9 2823                       JR      Z,MLD3
01AB FE23                       CP      23H             ; "#"
01AD 212902                     LD      HL,MTBL
01B0 2004                       JR      NZ,+6
01B2 214102                     LD      HL,M#TBL
01B5 13                         INC     DE
01B6 CDDD01                     CALL    ONPU
01B9 38D7                       JR      C,MLD1
01BB CDC802                     CALL    RYTHM
01BE 3815                       JR      C,MLD5
01C0 CDAB02                     CALL    MLDST
01C3 41                         LD      B,C
01C4 18CC                       JR      MLD1
01C6 3E03               MLD2:   LD      A,+3
01C8 32A011                     LD      (OCTV),A
01CB 13                         INC     DE
01CC 18C4                       JR      MLD1
01CE 3E01               MLD3:   LD      A,1
01D0 18F6                       JR      MLD2+2
01D2 CDC802             MLD4:   CALL    RYTHM
01D5 F5                 MLD5:   PUSH    AF
01D6 CDBE02                     CALL    MLDSP
01D9 F1                         POP     AF
01DA C39F06                     JP      RET3
01DD                    ;
01DD                    ;   ONPU TO RATIO CONV
01DD                    ;
01DD                    ;    EXIT (RATIO)=RATIO VALUE
01DD                    ;         C=ONTYO**TEMPO
01DD                    ;
01DD C5                 ONPU:   PUSH    BC
01DE 0608                       LD      B,8
01E0 1A                 ONP1:   LD      A,(DE)
01E1 BE                         CP      (HL)
01E2 2809                       JR      Z,ONP2
01E4 23                         INC     HL
01E5 23                         INC     HL
01E6 23                         INC     HL
01E7 10F8                       DJNZ    -6
01E9 37                         SCF
```



```
01EA 13                         INC     DE
01EB C1                         POP     BC
01EC C9                         RET
01ED 23                 ONP2:   INC     HL
01EE D5                         PUSH    DE
01EF 5E                         LD      E,(HL)
01F0 23                         INC     HL
01F1 56                         LD      D,(HL)
01F2 EB                         EX      DE,HL
01F3 7C                         LD      A,H
01F4 B7                         OR      A
01F5 2809                       JR      Z,+11
01F7 3AA011                     LD      A,(OCTV)
01FA 3D                         DEC     A
01FB 2803                       JR      Z,+5
01FD 29                         ADD     HL,HL
01FE 18FA                       JR      -4
0200 22A111                     LD      (RATIO),HL
0203 21A011                     LD      HL,OCTV
0206 3602                       LD      (HL),2
0208 2B                         DEC     HL
0209 D1                         POP     DE
020A 13                         INC     DE
020B 1A                         LD      A,(DE)
020C 47                         LD      B,A
020D E6F0                       AND     0F0H            ; ONTYO ?
020F FE30                       CP      30H
0211 2803                       JR      Z,+5
0213 7E                         LD      A,(HL)          ; HL=ONTYO
0214 1805                       JR      +7
0216 13                         INC     DE
0217 78                         LD      A,B
0218 E60F                       AND     0FH
021A 77                         LD      (HL),A          ; HL=ONTYO
021B 215902                     LD      HL,OPTBL
021E 85                         ADD     A,L
021F 6F                         LD      L,A
0220 4E                         LD      C,(HL)
0221 3A9E11                     LD      A,(TEMPW)
0224 47                         LD      B,A
0225 AF                         XOR     A
0226 C3AB09                     JP      ONP3
0229                    ;
0229 43                 MTBL:   DEFB    43H             ; C
022A 7707                       DEFW    0777H
022C 44                         DEFB    44H             ; D
022D A706                       DEFW    06A7H
022F 45                         DEFB    45H             ; E
0230 ED05                       DEFW    05EDH
0232 46                         DEFB    46H             ; F
0233 9805                       DEFW    0598H
0235 47                         DEFB    47H             ; G
0236 FC04                       DEFW    04FCH
0238 41                         DEFB    41H             ; A
0239 7104                       DEFW    0471H
023B 42                         DEFB    42H             ; B
023C F503                       DEFW    03F5H
023E 52                         DEFB    52H             ; R
023F 0000                       DEFW    0
0241 43                 M#TBL:  DEFB    43H             ; #C
```

```
0242 0C07                       DEFW  070CH
0244 44                         DEFB  44H               ; #D
0245 4706                       DEFW  0647H
0247 45                         DEFB  45H               ; #E
0248 9805                       DEFW  0598H
024A 46                         DEFB  46H               ; #F
024B 4805                       DEFW  0548H
024D 47                         DEFB  47H               ; #G
024E B404                       DEFW  04B4H
0250 41                         DEFB  41H               ; #A
0251 3104                       DEFW  0431H
0253 42                         DEFB  42H               ; #B
0254 BB03                       DEFW  03BBH
0256 52                         DEFB  52H               ; #R
0257                    ;
0257 0000                       DEFW  0
0259 01                 OPTBL:  DEFB  1
025A 02                         DEFB  2
025B 03                         DEFB  3
025C 04                         DEFB  4
025D 06                         DEFB  6
025E 08                         DEFB  8
025F 0C                         DEFB  0CH
0260 10                         DEFB  10H
0261 18                         DEFB  18H
0262 20                         DEFB  20H
0263                    ;
0263                    ;
0263                    ;   FLASHING DATA SAVE
0263                    ;
0263 219211             ?SAVE:  LD    HL,FLSDT
0266 36EF                       LD    (HL),EFH
0268 3A7011                     LD    A,(KANAF)
026B B7                         OR    A
026C 2802                       JR    Z,KSL0
026E 36FF                       LD    (HL),FFH
0270 7E                 KSL0:   LD    A,(HL)
0271 F5                         PUSH  AF
0272 CDB10F                     CALL  ?PONT
0275 7E                         LD    A,(HL)
0276 328E11                     LD    (FLASH),A
0279 F1                         POP   AF
027A 77                         LD    (HL),A
027B AF                         XOR   A
027C 2100E0                     LD    HL,KEYPA
027F 77                         LD    (HL),A
0280 2F                         CPL
0281 77                         LD    (HL),A
0282 C9                         RET
0283                    ;
0283                    ;
0283                    ;    COLOUMN MANAGEMENT POINTER
0283                    ;         MGP.D=DECREMENT POINTER
0283                    ;         MGP.I=INCREMENT POINTER
0283                    ;
0283 F5                 MGP.I:  PUSH  AF
0284 E5                         PUSH  HL
0285 217C11                     LD    HL,MGPNT
0288 7E                         LD    A,(HL)
0289 3C                         INC   A
```



```
028A FE33                       CP    51                ; COLUMN=51 THEN 0
028C 2001                       JR    NZ,MGP0
028E AF                         XOR   A
028F E5                 MGP0:   PUSH  HL
0290 6F                         LD    L,A
0291 3A9111                     LD    A,(SPAGE)
0294 B7                         OR    A
0295 7D                         LD    A,L
0296 E1                         POP   HL
0297 2001                       JR    NZ,+3
0299 77                         LD    (HL),A
029A E1                         POP   HL
029B F1                         POP   AF
029C C9                         RET
029D                    ;
029D                    ;
029D                    ;
029D F5                 MGP.D:  PUSH  AF
029E E5                         PUSH  HL
029F 217C11                     LD    HL,MGPNT
02A2 7E                         LD    A,(HL)
02A3 3D                         DEC   A
02A4 F28F02                     JP    P,MGP0
02A7 3E32                       LD    A,50
02A9 18E4                       JR    MGP0
02AB                    ;
02AB                            ORG   02ABH             ; MLDST
02AB                    ;
02AB                    ;   MELODY START & STOP
02AB                    ;
02AB 2AA111             MLDST:  LD    HL,(RATIO)
02AE 7C                         LD    A,H
02AF B7                         OR    A
02B0 280C                       JR    Z,MLDSP
02B2 D5                         PUSH  DE
02B3 EB                         EX    DE,HL
02B4 2104E0                     LD    HL,CONTO
02B7 73                         LD    (HL),E
02B8 72                         LD    (HL),D
02B9 3E01                       LD    A,1
02BB D1                         POP   DE
02BC 1806                       JR    MLDS1
02BE                    ;
02BE 3E34               MLDSP:  LD    A,34H             ; MODE SET
02C0 3207E0                     LD    (CONTF),A
02C3 AF                         XOR   A
02C4 3208E0             MLDS1:  LD    (SUNDG),A
02C7 C9                         RET
02C8                    ;
02C8                    ;   RHYTHM
02C8                    ;
02C8                    ;     B=COUNT
02C8                    ;   EXIT CF=1 BREAK
02C8                    ;         CF=0 OK
02C8                    ;
02C8 2100E0             RYTHM:  LD    HL,KEYPA
02CB 36F0                       LD    (HL),F0H
02CD 23                         INC   HL
02CE 7E                         LD    A,(HL)
02CF E681                       AND   81H
```

```
02D1 2002                JR    NZ,+4
02D3 37                  SCF
02D4 C9                  RET
02D5 3A08E0              LD    A,(TEMP)
02D8 0F                  RRCA
02D9 38FA                JR    C,-4
02DB 3A08E0              LD    A,(TEMP)
02DE 0F                  RRCA
02DF 30FA                JR    NC,-4
02E1 10F2                DJNZ  -12
02E3 AF                  XOR   A
02E4 C9                  RET
02E5                ;
02E5                ;   BELL
02E5                ;
02E5                ?BEL:  ENT
02E5 D5                  PUSH  DE
02E6 11B10D              LD    DE,?BELD
02E9 F7                  RST   6              ; CALL MELDY
02EA D1                  POP   DE
02EB C9                  RET
02EC                ;
02EC                ;
02EC                ;   TEMPO
02EC                ;
02EC                ;     ACC=VALUE
02EC                ?TEMP: ENT
02EC F5                  PUSH  AF
02ED C5                  PUSH  BC
02EE E60F                AND   0FH
02F0 47                  LD    B,A
02F1 3E08                LD    A,8
02F3 90                  SUB   B
02F4 329E11              LD    (TEMPW),A
02F7 C1                  POP   BC
02F8 F1                  POP   AF
02F9 C9                  RET
02FA                ;
02FA                ;   TIME SET
02FA                ;
02FA                ;     ACC=0 : AM
02FA                ;        =1 : PM
02FA                ;     DE=SEC, BINARY
02FA                ;
02FA                ?TMST: ENT
02FA F3                  DI
02FB C5                  PUSH  BC
02FC D5                  PUSH  DE
02FD E5                  PUSH  HL
02FE 329B11              LD    (AMPM),A
0301 3EF0                LD    A,0F0H
0303 329C11              LD    (TIMFG),A
0306 21C0A8              LD    HL,0A8C0H      ; 12H
0309 AF                  XOR   A
030A ED52                SBC   HL,DE
030C E5                  PUSH  HL
030D 23                  INC   HL
030E EB                  EX    DE,HL
030F 2107E0              LD    HL,CONTF
0312 3674                LD    (HL),74H
```



```
0314 36B0                LD    (HL),0B0H
0316 2B                  DEC   HL             ; CONT2
0317 73                  LD    (HL),E
0318 72                  LD    (HL),D
0319 2B                  DEC   HL             ; CONT1
031A 360A                LD    (HL),0AH
031C 3600                LD    (HL),0
031E 23                  INC   HL
031F 23                  INC   HL             ; CONTF
0320 3680                LD    (HL),80H
0322 2B                  DEC   HL             ; CONT2
0323 4E                  LD    C,(HL)
0324 7E                  LD    A,(HL)
0325 BA                  CP    D
0326 20FB                JR    NZ,?TMS1
0328 79                  LD    A,C
0329 BB                  CP    E
032A 20F7                JR    NZ,?TMS1
032C 2B                  DEC   HL
032D 00                  NOP
032E 00                  NOP
032F 00                  NOP
0330 360C                LD    (HL),0CH
0332 367B                LD    (HL),7BH
0334 23                  INC   HL
0335 D1                  POP   DE
0336 4E                  LD    C,(HL)
0337 7E                  LD    A,(HL)
0338 BA                  CP    D
0339 20FB                JR    NZ,?TMS2
033B 79                  LD    A,C
033C BB                  CP    E
033D 20F7                JR    NZ,?TMS2
033F E1                  POP   HL
0340 D1                  POP   DE
0341 C1                  POP   BC
0342 FB                  EI
0343 C9                  RET
0344                ;
0344                ;;
0344                ;   TIME READ
0344                ;
0344                ;     EXIT  ACC=0 :AM
0344                ;              =1 :PM
0344                ;           DE=SEC. BINARY
0344                ;
0344                ?TMRD: ENT
0344 E5                  PUSH  HL
0345 2107E0              LD    HL,CONTF
0348 3680                LD    (HL),80H
034A 2B                  DEC   HL             ; CONT2
034B F3                  DI
034C 5E                  LD    E,(HL)
034D 56                  LD    D,(HL)
034E FB                  EI
034F 7B                  LD    A,E
0350 B2                  OR    D
0351 280E                JR    Z,?TMR1
0353 AF                  XOR   A
0354 21C0A8              LD    HL,A8C0H
```

```
   **  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 11          09/04/81

01 0357 ED52                    SBC   HL,DE
02 0359 3810                    JR    C,?TMR2
03 035B EB                      EX    DE,HL
04 035C 3A9B11                  LD    A,(AMPM)
05 035F E1                      POP   HL
06 0360 C9                      RET
07 0361 11C0A8          ?TMR1:  LD    DE,A8C0H
08 0364 3A9B11                  LD    A,(AMPM)
09 0367 EE01                    XOR   1
10 0369 E1                      POP   HL
11 036A C9                      RET
12 036B F3              ?TMR2:  DI
13 036C 2106E0                  LD    HL,CONT2
14 036F 7E                      LD    A,(HL)
15 0370 2F                      CPL
16 0371 5F                      LD    E,A
17 0372 7E                      LD    A,(HL)
18 0373 2F                      CPL
19 0374 57                      LD    D,A
20 0375 FB                      EI
21 0376 13                      INC   DE
22 0377 18EB                    JR    ?TMR1+3
23 0379                 ;
24 0379 -               ;
25 0379                 ;   TIME INTERRUPT
26 0379                 ;
27 0379                 TIMIN:  ENT
28 0379 F5                      PUSH  AF
29 037A C5                      PUSH  BC
30 037B D5                      PUSH  DE
31 037C E5                      PUSH  HL
32 037D 219B11                  LD    HL,AMPM
33 0380 7E                      LD    A,(HL)
34 0381 EE01                    XOR   1
35 0383 77                      LD    (HL),A
36 0384 2107E0                  LD    HL,CONTF
37 0387 3680                    LD    (HL),80H
38 0389 2B                      DEC   HL                ; CONT2
39 038A E5                      PUSH  HL
40 038B 5E                      LD    E,(HL)
41 038C 56                      LD    D,(HL)
42 038D 21C0A8                  LD    HL,A8C0H
43 0390 19                      ADD   HL,DE
44 0391 2B                      DEC   HL
45 0392 2B                      DEC   HL
46 0393 EB                      EX    DE,HL
47 0394 E1                      POP   HL
48 0395 73                      LD    (HL),E
49 0396 72                      LD    (HL),D
50 0397 E1                      POP   HL
51 0398 D1                      POP   DE
52 0399 C1                      POP   BC
53 039A F1                      POP   AF
54 039B FB                      EI
55 039C C9                      RET
56 039D                 ;
57 039D                 ;   DSP03 PAGE MODE1
58 039D                 ;
59 039D EB              .DSP03: EX    DE,HL
60 039E 3601                    LD    (HL),+1
```

```
   **  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 12          09/04/81

01 03A0 23                      INC   HL
02 03A1 3600                    LD    (HL),0
03 03A3 C37B0E                  JP    CURSR
04 03A6                 ;
05 03A6                 ;
06 03A6                 ;
07 03A6                 ;  MANAGMENT PAGE MODE1
08 03A6                 ;
09 03A6                 ;
10 03A6 3A7211          .MANG2: LD    A,(DSPXY+1)
11 03A9 85                      ADD   A,L
12 03AA 6F                      LD    L,A
13 03AB 7E                      LD    A,(HL)
14 03AC 23                      INC   HL
15 03AD CB16                    RL    (HL)
16 03AF B6                      OR    (HL)
17 03B0 CB1E                    RR    (HL)
18 03B2 0F                      RRCA
19 03B3 EB                      EX    DE,HL
20 03B4 2A7111                  LD    HL,(DSPXY)
21 03B7 C9                      RET
22 03B8                 ;
23 03B8                 ;
24 03BA                         ORG   03BAH             ;PRNTHL  9
25 03BA                 PRTHL:  ENT
26 03BA 7C                      LD    A,H
27 03BB CDC303                  CALL  PRTHX
28 03BE 7D                      LD    A,L
29 03BF 1802                    JR    PRTHX
30 03C1                 ;
31 03C3                         ORG   03C3H             ;PRTHX   23
32 03C3                 ;
33 03C3                 ;   (ACC) PRINT
34 03C3                 ;
35 03C3                 PRTHX:  ENT
36 03C3 F5                      PUSH  AF
37 03C4 0F                      RRCA
38 03C5 0F                      RRCA
39 03C6 0F                      RRCA
40 03C7 0F                      RRCA
41 03C8 CDDA03                  CALL  ASC
42 03CB CD1200                  CALL  PRNT
43 03CE F1                      POP   AF
44 03CF CDDA03                  CALL  ASC
45 03D2 C31200                  JP    PRNT
46 03D5                 ;
47 03D5                 ;
48 03D5                 ;    GETL RETURN
49 03D5                 ;
50 03D5 D1              GETLL:  POP   DE
51 03D6 E1                      POP   HL
52 03D7 C1                      POP   BC
53 03D8 F1                      POP   AF
54 03D9 C9                      RET
55 03DA                 ;
56 03DA                 ;
57 03DA                 ;
58 03DA                         ORG   03DAH             ;ASC     31
59 03DA                 ;
60 03DA                 ;  HEXA TO ASCII
```

```
                        ;
                ASC:    ENT
03DA E60F               AND     0FH
03DC FE0A               CP      0AH
03DE 3802               JR      C,NOADD
03E0 C607               ADD     A,7
03E2 C630       NOADD:  ADD     A,30H
03E4 C9                 RET
                        ;
                        ;  ASCII TO HEXA
                        ;
03E5 FE30       HEXJ:   CP      30H
03E7 D8                 RET     C
03E8 FE3A               CP      3AH
03EA 3806               JR      C,HEX1
03EC D607               SUB     7
03EE FE40               CP      40H
03F0 3003               JR      NC,HEX2
03F2 E60F       HEX1:   AND     0FH
03F4 C9                 RET
03F5 37         HEX2:   SCF
03F6 C9                 RET
03F7                    ;
03F7                    ;
03F9                    ORG     03F9H           ;HEX    23
03F9            HEX:    ENT
03F9 18EA               JR      HEXJ
                        ;
03FB                    ;;
03FB                    ;   @DPCT PUCH
03FB                    ;
03FB 2A7111     HOME:   LD      HL,(DSPXY)
03FE 3A7C11             LD      A,(MGPNT)
0401 94                 SUB     H
0402 3002               JR      NC,HOM1
0404 C632               ADD     A,50
0406 327C11     HOM1:   LD      (MGPNT),A
0409 210000     HOMO:   LD      HL,0
040C C3690E             JP      CURS3
040F                    ;
040F                    ;
0410                    ORG     0410H           ;HLHEX  15
0410                    ;
0410                    ;
0410                    ;  4 ASCII TO (HL)
0410                    ;
0410                    ;  DE=DATA LOW ADR.
0410                    ;  EXIT  CF=0 : OK
0410                    ;         =1 : OUT
0410                    ;
0410            HLHEX:  ENT
0410 D5                 PUSH    DE
0411 CD1F04             CALL    2HEX
0414 3807               JR      C,+9
0416 67                 LD      H,A
0417 CD1F04             CALL    2HEX
041A 3801               JR      C,+3
041C 6F                 LD      L,A
041D D1         HL1:    POP     DE
041E C9                 RET
```



```
041F                    ;
041F                    ORG     041FH           ;2HEX   482
041F                    ;
041F                    ;
041F                    ;  2 ASCII TO (ACC)
041F                    ;
041F                    ;  DE=DATA  LOW ADR.
041F                    ;
041F                    ;  EXIT  CF=0 : OK
041F                    ;           =1 : OUT
041F                    ;
041F            2HEX:   ENT
041F C5                 PUSH    BC
0420 1A                 LD      A,(DE)
0421 13                 INC     DE
0422 CDF903             CALL    HEX
0425 380D               JR      C,+15
0427 0F                 RRCA
0428 0F                 RRCA
0429 0F                 RRCA
042A 0F                 RRCA
042B 4F                 LD      C,A
042C 1A                 LD      A,(DE)
042D 13                 INC     DE
042E CDF903             CALL    HEX
0431 3801               JR      C,+3
0433 B1                 OR      C
0434 C1         2HE1:   POP     BC
0435 C9                 RET
0436                    ;
0436                    ;;
0436                    ;  WRITE INFORMATION
0436                    ;
0436            ?WRI:   ENT
0436 F3                 DI
0437 D5                 PUSH    DE
0438 C5                 PUSH    BC
0439 E5                 PUSH    HL
043A 16D7               LD      D,D7H           ; 'W'
043C 1ECC               LD      E,CCH           ; 'L'
043E 21F010             LD      HL,IBUFE
0441 018000             LD      BC,80H
0444 CD1A07     WRI1:   CALL    CKSUM
0447 CDA306             CALL    MOTOR
044A 3818               JR      C,WRI3
044C 7B                 LD      A,E
044D FECC               CP      CCH             ; 'L'
044F 200D               JR      NZ,WRI2
0451 CD0900             CALL    NL
0454 D5                 PUSH    DE
0455 116704             LD      DE,MSG#7
0458 DF                 RST     3               ; CALL MSGX
0459 11F110             LD      DE,NAME
045C DF                 RST     3               ; CALL MSGX
045D D1                 POP     DE
045E CD7A07     WRI2:   CALL    GAP
0461 CD8504             CALL    WTAPE
0464 C35205     WRI3:   JP      RET2
0467                    ;
0467 57         MSG#7:  DEFM    'W'             ; WRITING
```

```
   **  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 15          09/04/81

01 0468 9DA6                    DEFW  A69DH
02 046A 96A6                    DEFW  A696H
03 046C B097                    DEFW  97B0H
04 046E 200D                    DEFW  0D20H
05 0470                 ;;
06 0470                 ;   WRITE DATA
07 0470                 ;
08 0470                 ;    EXIT CF=0 : OK
09 0470                 ;           =1 : BREAK
10 0470                 ;
11 0470                 ?WRD:   ENT
12 0470 F3                      DI
13 0471 D5                      PUSH  DE
14 0472 C5                      PUSH  BC
15 0473 E5                      PUSH  HL
16 0474 16D7                    LD    D,0D7H            ;'W'
17 0476 1E53                    LD    E,53H             ; 'S'
18 0478 ED4B0211                LD    BC,(SIZE)
19 047C 2A0411                  LD    HL,(DTADR)
20 047F 78                      LD    A,B
21 0480 B1                      OR    C
22 0481 2848                    JR    Z,RET1
23 0483 18BF                    JR    WRI1
24 0485                 ;
25 0485                 ;;
26 0485                 ;   TAPE WRITE
27 0485                 ;
28 0485                 ;     BC=BYTE SIZE
29 0485                 ;     HL=DATA LOW ADR.
30 0485                 ;
31 0485                 ;    EXIT  CF=0 : OK
32 0485                 ;            =1 : BREAK
33 0485                 ;
34 0485 D5              WTAPE:  PUSH  DE
35 0486 C5                      PUSH  BC
36 0487 E5                      PUSH  HL
37 0488 1602                    LD    D,2
38 048A 3EF0                    LD    A,0F0H
39 048C 3200E0                  LD    (KEYPA),A
40 048F 7E              WTAP1:  LD    A,(HL)
41 0490 CD6707                  CALL  WBYTE
42 0493 3A01E0                  LD    A,(KEYPB)
43 0496 E681                    AND   81H               ; SHIFT & BREAK
44 0498 C29E04                  JP    NZ,WTAP2
45 049B 37                      SCF
46 049C 182D                    JR    WTAP3
47 049E 23              WTAP2:  INC   HL
48 049F 0B                      DEC   BC
49 04A0 78                      LD    A,B
50 04A1 B1                      OR    C
51 04A2 C28F04                  JP    NZ,WTAP1
52 04A5 2A9711                  LD    HL,(SUMDT)
53 04A8 7C                      LD    A,H
54 04A9 CD6707                  CALL  WBYTE
55 04AC 7D                      LD    A,L
56 04AD CD6707                  CALL  WBYTE
57 04B0 CD570D                  CALL  LONG
58 04B3 15                      DEC   D
59 04B4 C2BB04                  JP    NZ,+7
60 04B7 B7                      OR    A

   **  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 16          09/04/81

01 04B8 C3CB04                  JP    WTAP3
02 04BB 0600                    LD    B,0
03 04BD CD3E0D                  CALL  SHORT
04 04C0 05                      DEC   B
05 04C1 C2BD04                  JP    NZ,-4
06 04C4 E1                      POP   HL
07 04C5 C1                      POP   BC
08 04C6 C5                      PUSH  BC
09 04C7 E5                      PUSH  HL
10 04C8 C38F04                  JP    WTAP1
11 04CB                 WTAP3:
12 04CB E1              RET1:   POP   HL
13 04CC C1                      POP   BC
14 04CD D1                      POP   DE
15 04CE C9                      RET
16 04CF                 ;
17 04CF                 ;;
18 04CF                 ;   READ INFORMATION
19 04CF                 ;
20 04CF                 ;    EXIT ACC=0 : OK CF=0
21 04CF                 ;           =1 : ER CF=1
22 04CF                 ;           =2 : BREAK CF=1
23 04CF                 ;
24 04CF                 ?RDI:   ENT
25 04CF F3                      DI
26 04D0 D5                      PUSH  DE
27 04D1 C5                      PUSH  BC
28 04D2 E5                      PUSH  HL
29 04D3 16D2                    LD    D,0D2H            ; 'R'
30 04D5 1ECC                    LD    E,0CCH            ; 'L'
31 04D7 018000                  LD    BC,80H
32 04DA 21F010                  LD    HL,IBUFE
33 04DD CDA306          RD1:    CALL  MOTOR
34 04E0 DA7005                  JP    C,RTP6
35 04E3 CD5806                  CALL  TMARK
36 04E6 DA7005                  JP    C,RTP6
37 04E9 CD0505                  CALL  RTAPE
38 04EC C35205                  JP    RTP4
39 04EF                 ;;
40 04EF                 ;   READ DATA
41 04EF                 ;
42 04EF                 ;    EXIT SAME UP
43 04EF                 ;
44 04EF                 ?RDD:   ENT
45 04EF F3                      DI
46 04F0 D5                      PUSH  DE
47 04F1 C5                      PUSH  BC
48 04F2 E5                      PUSH  HL
49 04F3 16D2                    LD    D,0D2H            ; 'R'
50 04F5 1E53                    LD    E,53H             ; 'S'
51 04F7 ED4B0211                LD    BC,(SIZE)
52 04FB 2A0411                  LD    HL,(DTADR)
53 04FE 78                      LD    A,B
54 04FF B1                      OR    C
55 0500 CA5205                  JP    Z,RTP4
56 0503 18D8                    JR    RD1
57 0505                 ;
58 0505                 ;;
59 0505                 ;   READ TAPE
60 0505                 ;
```

```
0505                    ;     BC=SIZE
0505                    ;     DE=LOAD ADR.
0505                    ;
0505                    ;    EXIT  ACC=0 : OK CF=0
0505                    ;             =1 : ER   =1
0505                    ;             =2 : BREAK=1
0505                    ;
0505 D5                 RTAPE:  PUSH  DE
0506 C5                         PUSH  BC
0507 E5                         PUSH  HL
0508 2602                       LD    H,2
050A 0101E0             RTP1:   LD    BC,KEYPB
050D 1102E0                     LD    DE,CSTR
0510 CD0106             RTP2:   CALL  EDGE
0513 DA7005                     JP    C,RTP6
0516 CDA209                     CALL  DLY3                ; CALL DLY2*3
0519 1A                         LD    A,(DE)
051A E620                       AND   20H
051C CA1005                     JP    Z,RTP2
051F 54                         LD    D,H
0520 210000                     LD    HL,0
0523 229711                     LD    (SUMDT),HL
0526 E1                         POP   HL
0527 C1                         POP   BC
0528 C5                         PUSH  BC
0529 E5                         PUSH  HL
052A CD2406             RTP3:   CALL  RBYTE
052D DA7005                     JP    C,RTP6
0530 77                         LD    (HL),A
0531 23                         INC   HL
0532 0B                         DEC   BC
0533 78                         LD    A,B
0534 B1                         OR    C
0535 C22A05                     JP    NZ,RTP3
0538 2A9711                     LD    HL,(SUMDT)
053B CD2406                     CALL  RBYTE
053E DA7005                     JP    C,RTP6
0541 5F                         LD    E,A
0542 CD2406                     CALL  RBYTE
0545 DA7005                     JP    C,RTP6
0548 BD                         CP    L
0549 C26305                     JP    NZ,RTP5
054C 7B                         LD    A,E
054D BC                         CP    H
054E C26305                     JP    NZ,RTP5
0551 AF                 RTP8:   XOR   A
0552                    RTP4:
0552 E1                 RET2:   POP   HL
0553 C1                         POP   BC
0554 D1                         POP   DE
0555 CD0007                     CALL  MSTOP
0558 F5                         PUSH  AF
0559 3A9C11                     LD    A,(TIMFG)
055C FEF0                       CP    F0H
055E 2001                       JR    NZ,+3
0560 FB                         EI
0561 F1                         POP   AF
0562 C9                         RET
0563                    ;
0563 15                 RTP5:   DEC   D
```



```
0564 2806                       JR    Z,RTP7
0566 62                         LD    H,D
0567 CDE20F                     CALL  GAPCK
056A 189E                       JR    RTP1
056C 3E01               RTP7:   LD    A,1
056E 1802                       JR    RTP9
0570 3E02               RTP6:   LD    A,2
0572 37                 RTP9:   SCF
0573 18DD                       JR    RTP4
0575                    ;
0575                    ;;
0575                    ;   VERIFY
0575                    ;
0575                    ;    EXIT ACC =0 : OK CF=0
0575                    ;             =1 : ER CF=1
0575                    ;             =2 : BREAK CF=1
0575                    ;
0575 F3                 ?VRFY:  ENT
0576 D5                         DI
0577 C5                         PUSH  DE
0578 E5                         PUSH  BC
0579 ED4B0211                   PUSH  HL
057D 2A0411                     LD    BC,(SIZE)
0580 16D2                       LD    HL,(DTADR)
0582 1E53                       LD    D,D2H               ; 'R'
0584 78                         LD    E,53H               ; 'S'
0585 B1                         LD    A,B
0586 28CA                       OR    C
0588 CD1A07                     JR    Z,RTP4
058B CDA306                     CALL  CKSUM
058E 38E0                       CALL  MOTOR
0590 CD5806                     JR    C,RTP6
0593 DA7005                     CALL  TMARK
0596 CD9B05                     JP    C,RTP6
0599 18B7                       CALL  TVRFY
059B                            JR    RTP4
059B                    ;
059B                    ;;
059B                    ;   DATA VERIFY
059B                    ;
059B                    ;     BC=SIZE
059B                    ;     HL=DATA LOW ADR
059B                    ;     CSMDT=CHECK SUM
059B                    ;    EXIT ACC=0 : OK CF=0
059B                    ;             =1 : ER   =1
059B                    ;             =2 : BREAK=1
059B                    ;
059B D5                 TVRFY:  PUSH  DE
059C C5                         PUSH  BC
059D E5                         PUSH  HL
059E 2602                       LD    H,2
05A0 0101E0             TVF1:   LD    BC,KEYPB
05A3 1102E0                     LD    DE,CSTR
05A6 CD0106             TVF2:   CALL  EDGE
05A9 DA7005                     JP    C,RTP6
05AC CDA209                     CALL  DLY3                ; CALL DLY2*3
05AF 1A                         LD    A,(DE)
05B0 E620                       AND   20H
05B2 CAA605                     JP    Z,TVF2
```

```
   **  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 19          09/04/81

01 05B5 54                      LD    D,H
02 05B6 E1                      POP   HL
03 05B7 C1                      POP   BC
04 05B8 C5                      PUSH  BC
05 05B9 E5                      PUSH  HL
06 05BA CD2406          TVF3:   CALL  RBYTE
07 05BD DA7005                  JP    C,RTP6
08 05C0 BE                      CP    (HL)
09 05C1 C26C05                  JP    NZ,RTP7
10 05C4 23                      INC   HL
11 05C5 0B                      DEC   BC
12 05C6 78                      LD    A,B
13 05C7 B1                      OR    C
14 05C8 C2BA05                  JP    NZ,TVF3
15 05CB 2A9911                  LD    HL,(CSMDT)
16 05CE CD2406                  CALL  RBYTE
17 05D1 BC                      CP    H
18 05D2 2098                    JR    NZ,RTP7
19 05D4 CD2406                  CALL  RBYTE
20 05D7 BD                      CP    L
21 05D8 2092                    JR    NZ,RTP7
22 05DA 15                      DEC   D
23 05DB CA5105                  JP    Z,RTP8
24 05DE 62                      LD    H,D
25 05DF 18BF                    JR    TVF1
26 05E1                 ;
27 05E1                 ;     PRINT '00'
28 05E1                 ;
29 05E1 11FC09          GETLD:  LD    DE,00MSG
30 05E4 DF                      RST   3
31 05E5 C30708                  JP    AUTO2
32 05E8                 ;
33 05E8                 ;
34 05E8                 ;
35 05E8                 ;     ROLL UP
36 05E8                 ;
37 05E8 217A11          ROLUP:  LD    HL,PBIAS
38 05EB 3A7F11                  LD    A,(ROLEND)
39 05EE BE                      CP    (HL)
40 05EF CAE50E                  JP    Z,?RSTR
41 05F2 C3A90F                  JP    ROLU1
42 05F5                 ;
43 05F5                 ;
44 05F5                 ;   FLASHING DATA LOAD
45 05F5                 ;
46 05F5 F5              ?LOAD:  PUSH  AF
47 05F6 3A8E11                  LD    A,(FLASH)
48 05F9 CDB10F                  CALL  ?PONT
49 05FC 77                      LD    (HL),A
50 05FD F1                      POP   AF
51 05FE C9                      RET
52 05FF                 ;
53 05FF                 ;
54 05FF                 ;
55 0601                         ORG   0601H             ;EDGE   35
56 0601                 ;
57 0601                 ;;
58 0601                 ;   EDGE
59 0601                 ;
60 0601                 ;     BC=KEYPB
```

```
   **  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 20          09/04/81

01 0601                 ;     DE=CSTR
02 0601                 ;   EXIT CF=0 : EDGE
03 0601                 ;          =1 : BREAK
04 0601                 ;
05 0601 3EF0            EDGE:   LD    A,F0H
06 0603 3200E0                  LD    (KEYPA),A
07 0606 00                      NOP
08 0607 0A              EDG1:   LD    A,(BC)
09 0608 E681                    AND   81H               ; SHIFT & BREAK
10 060A C20F06                  JP    NZ,+5
11 060D 37                      SCF
12 060E C9                      RET
13 060F 1A                      LD    A,(DE)
14 0610 E620                    AND   20H
15 0612 C20706                  JP    NZ,EDG1
16 0615 0A              EDG2:   LD    A,(BC)
17 0616 E681                    AND   81H
18 0618 C21D06                  JP    NZ,+5
19 061B 37                      SCF
20 061C C9                      RET
21 061D 1A                      LD    A,(DE)
22 061E E620                    AND   20H
23 0620 CA1506                  JP    Z,EDG2
24 0623 C9                      RET
25 0624                 ;
26 0624                 ;
27 0624                         ORG   0624H             ;RBYTE  220
28 0624                 ;
29 0624                 ;;
30 0624                 ;   1 BYTE READ
31 0624                 ;
32 0624                 ;     EXIT  SUMDT=STORE
33 0624                 ;      CF=1 : BREAK
34 0624                 ;      CF=0 : DATA=ACC
35 0624                 ;
36 0624 C5              RBYTE:  PUSH  BC
37 0625 D5                      PUSH  DE
38 0626 E5                      PUSH  HL
39 0627 210008                  LD    HL,0800H
40 062A 0101E0                  LD    BC,KEYPB
41 062D 1102E0                  LD    DE,CSTR
42 0630 CD0106          RBY1:   CALL  EDGE
43 0633 DA5406                  JP    C,RBY3
44 0636 CDA209                  CALL  DLY3              ; CALL DLY2*3
45 0639 1A                      LD    A,(DE)
46 063A E620                    AND   20H
47 063C CA4906                  JP    Z,RBY2
48 063F E5                      PUSH  HL
49 0640 2A9711                  LD    HL,(SUMDT)
50 0643 23                      INC   HL
51 0644 229711                  LD    (SUMDT),HL
52 0647 E1                      POP   HL
53 0648 37                      SCF
54 0649 7D              RBY2:   LD    A,L
55 064A 17                      RLA
56 064B 6F                      LD    L,A
57 064C 25                      DEC   H
58 064D C23006                  JP    NZ,RBY1
59 0650 CD0106                  CALL  EDGE
60 0653 7D                      LD    A,L
```

```
0654 E1          RBY3:   POP   HL
0655 D1                  POP   DE
0656 C1                  POP   BC
0657 C9                  RET
0658             ;
0658             ;;
0658             ;   TAPE MARK DETECT
0658             ;
0658             ;     E=@L@ :INFORMATION
0658             ;       =@S@ :DATA
0658             ;    EXIT CF=0 :OK
0658             ;           =1 :BREAK
0658             ;
0658 CDE20F      TMARK:  CALL  GAPCK
065B C5                  PUSH  BC
065C D5                  PUSH  DE
065D E5                  PUSH  HL
065E 212828              LD    HL,2828H
0661 7B                  LD    A,E
0662 FECC                CP    0CCH              ;'L'
0664 CA6A06              JP    Z,+6
0667 211414              LD    HL,1414H
066A 229511              LD    (TMCNT),HL
066D 0101E0              LD    BC,KEYPB
0670 1102E0              LD    DE,CSTR
0673 2A9511      TM1:    LD    HL,(TMCNT)
0676 CD0106      TM2:    CALL  EDGE
0679 DA9F06              JP    C,TM4
067C CDA209              CALL  DLY3              ; CALL DLY2*3
067F 1A                  LD    A,(DE)
0680 E620                AND   20H
0682 CA7306              JP    Z,TM1
0685 25                  DEC   H
0686 C27606              JP    NZ,TM2
0689 CD0106      TM3:    CALL  EDGE
068C DA9F06              JP    C,TM4
068F CDA209              CALL  DLY3              ; CALL DLY2*3
0692 1A                  LD    A,(DE)
0693 E620                AND   20H
0695 C27306              JP    NZ,TM1
0698 2D                  DEC   L
0699 C28906              JP    NZ,TM3
069C CD0106              CALL  EDGE
069F             RET3:
069F E1          TM4:    POP   HL
06A0 D1                  POP   DE
06A1 C1                  POP   BC
06A2 C9                  RET
06A3             ;
06A3             ;;
06A3             ;   MOTOR ON
06A3             ;
06A3             ;     D=@W@ :WRITE
06A3             ;       =@R@ : READ
06A3             ;  EXIT CF=0 :OK
06A3             ;           =1 :BREAK
06A3 C5          MOTOR:  PUSH  BC
06A4 D5                  PUSH  DE
06A5 E5                  PUSH  HL
06A6 060A                LD    B,10
```



```
06A8 3A02E0      MOT1:   LD    A,(CSTR)
06AB E610                AND   10H
06AD 280A                JR    Z,MOT4
06AF 06A6        MOT2:   LD    B,A6H             ; 2 SEC DLAY
06B1 CDA70D              CALL  DLY12
06B4 10FB                DJNZ  -3
06B6 AF                  XOR   A
06B7 18E6        MOT7:   JR    RET3
06B9 3E06        MOT4:   LD    A,06H
06BB 2103E0              LD    HL,CSTPT
06BE 77                  LD    (HL),A
06BF 3C                  INC   A
06C0 77                  LD    (HL),A
06C1 10E5                DJNZ  MOT1
06C3 CD0900              CALL  NL
06C6 7A                  LD    A,D
06C7 FED7                CP    0D7H              ; 'W'
06C9 2805                JR    Z,MOT8
06CB 119E0D              LD    DE,MSG#1
06CE 1807                JR    MOT9
06D0 11F406      MOT8:   LD    DE,MSG#3
06D3 DF                  RST   3                 ; CALL MSGX
06D4 11A00D              LD    DE,MSG#2
06D7 DF          MOT9:   RST   3                 ; CALL MSGX
06D8 3A02E0      MOT5:   LD    A,(CSTR)
06DB E610                AND   10H
06DD 20D0                JR    NZ,MOT2
06DF CD110D              CALL  ?BRK
06E2 20F4                JR    NZ,MOT5
06E4 37                  SCF
06E5 18D0                JR    MOT7
06E7             ;
06E7             ;
06E7             ;
06E7             ;  ?KEY  GRAPH KEY INPUT
06E7             ;   :
06E7 06C9        #GRP:   LD    B,0C9H
06E9 3A7011              LD    A,(KANAF)
06EC B7                  OR    A
06ED 2001                JR    NZ,+3
06EF 04                  INC   B
06F0 78                  LD    A,B
06F1 C3D608              JP    ?KY1
06F4             ;
06F4             ;
06F4 7F20        MSG#3:  DEFW  207FH             ; PRESS RECORD
06F6 5245434F            DEFM  'RECORD.'
06FA 52442E
06FD 0D                  DEFB  0DH
06FE             ;
06FE             ;
0700                     ORG   0700H             ;MSTOP   89
0700             ;
0700             ;;
0700             ;  MOTOR STOP
0700             ;
0700 F5          MSTOP:  PUSH  AF
0701 C5                  PUSH  BC
0702 D5                  PUSH  DE
0703 060A                LD    B,10
```

```
** Z80 ASSEMBLER SB-7201 (MZ-80A.MONITOR) PAGE 23          09/04/81

0705 3A02E0          MST1:   LD    A,(CSTR)
0708 E610                    AND   10H
070A 280B                    JR    Z,MST3
070C 3E06            MST2:   LD    A,06H
070E 3203E0                  LD    (CSTPT),A
0711 3C                      INC   A
0712 3203E0                  LD    (CSTPT),A
0715 10EE                    DJNZ  MST1
0717 C3E60E          MST3:   JP    ?RSTR1
071A                 ;
071A                 ;
071A                 ;
071A                 ;   CHECK SUM
071A                 ;
071A                 ;     BC=SIZE
071A                 ;     HL=DATA ADR.
071A                 ;    EXIT SUMDT=STORE
071A                 ;         CSMDT=STORE
071A                 ;
071A C5              CKSUM:  PUSH  BC
071B D5                      PUSH  DE
071C E5                      PUSH  HL
071D 110000                  LD    DE,0
0720 78              CKS1:   LD    A,B
0721 B1                      OR    C
0722 200B                    JR    NZ,CKS2
0724 EB                      EX    DE,HL
0725 229711                  LD    (SUMDT),HL
0728 229911                  LD    (CSMDT),HL
072B E1                      POP   HL
072C D1                      POP   DE
072D C1                      POP   BC
072E C9                      RET
072F 7E              CKS2:   LD    A,(HL)
0730 C5                      PUSH  BC
0731 0608                    LD    B,+8
0733 07              CKS3:   RLCA
0734 3001                    JR    NC,+3
0736 13                      INC   DE
0737 10FA                    DJNZ  CKS3
0739 C1                      POP   BC
073A 23                      INC   HL
073B 0B                      DEC   BC
073C 18E2                    JR    CKS1
073E                 ;
073E                 ;    SWEP PART2
073E                 ;
073E 07              SWEP4:  RLCA
073F 07                      RLCA
0740 07                      RLCA
0741 4F                      LD    C,A
0742 7B                      LD    A,E
0743 25                      DEC   H
0744 0F                      RRCA
0745 30FC                    JR    NC,-2
0747 7C                      LD    A,H
0748 81                      ADD   A,C
0749 4F                      LD    C,A
074A C3740A                  JP    SWEP01
```

```
** Z80 ASSEMBLER SB-7201 (MZ-80A.MONITOR) PAGE 24          09/04/81

074D                 ;
074D                 ;   MODE SET OF KEYPORT
074D                 ;
074D                 ?MODE:  ENT
074D 2103E0                  LD    HL,KEYPF
0750 368A                    LD    (HL),8AH        ; 10001010
0752 3607                    LD    (HL),07H        ; PC3=1
0754 3605                    LD    (HL),05H        ; PC2=1
0756 3601            VGOFF:  LD    (HL),01H        ; PC0 SET
0758 C9                      RET
0759                 ;
0759                 ;
0759                 ;
0759                 ;
0759                         ORG   0759H           ;DLY1  7
0759                 ;;
0759                 ;   107 MICRO SEC DELY
0759                 ;
0759 3E0E            DLY1:   LD    A,14
075B 3D                      DEC   A
075C C25B07                  JP    NZ,-1
075F C9                      RET
0760                 ;
0760                         ORG   0760H           ;DLY2  240
0760                 ;
0760 3E0D            DLY2:   LD    A,13
0762 3D                      DEC   A
0763 C26207                  JP    NZ,-1
0766 C9                      RET
0767                 ;
0767                 ;
0767                 ;;
0767                 ;  1 BYTE WRITE
0767                 ;
0767 C5              WBYTE:  PUSH  BC
0768 0608                    LD    B,+8
076A CD570D                  CALL  LONG
076D 07              WBY1:   RLCA
076E DC570D                  CALL  C,LONG
0771 D43E0D                  CALL  NC,SHORT
0774 05                      DEC   B
0775 C26D07                  JP    NZ,WBY1
0778 C1                      POP   BC
0779 C9                      RET
077A                 ;
077A                 ;;
077A                 ;   GAP + TAPEMARK
077A                 ;
077A                 ;    E=@L@ LONG GAP
077A                 ;     =@S@ SHORT GAP
077A                 ;
077A C5              GAP:    PUSH  BC
077B D5                      PUSH  DE
077C 7B                      LD    A,E
077D 01F055                  LD    BC,55F0H
0780 112828                  LD    DE,2828H
0783 FECC                    CP    CCH             ;'L'
0785 CA8E07                  JP    Z,GAP1
0788 01F82A                  LD    BC,2AF8H
078B 111414                  LD    DE,1414H
```

```
01 078E CD3E0D          GAP1:   CALL    SHORT
02 0791 0B                      DEC     BC
03 0792 78                      LD      A,B
04 0793 B1                      OR      C
05 0794 20F8                    JR      NZ,-6
06 0796 CD570D          GAP2:   CALL    LONG
07 0799 15                      DEC     D
08 079A 20FA                    JR      NZ,-4
09 079C CD3E0D          GAP3:   CALL    SHORT
10 079F 1D                      DEC     E
11 07A0 20FA                    JR      NZ,-4
12 07A2 CD570D                  CALL    LONG
13 07A5 D1                      POP     DE
14 07A6 C1                      POP     BC
15 07A7 C9                      RET
16 07A8                 ;
17 07A8  P              KEYPA:  EQU     E000H
18 07A8  P              KEYPB:  EQU     E001H
19 07A8  P              KEYPC:  EQU     E002H
20 07A8  P              KEYPF:  EQU     E003H
21 07A8  P              CSTR:   EQU     E002H
22 07A8  P              CSTPT:  EQU     E003H
23 07A8  P              CONT0:  EQU     E004H
24 07A8  P              CONT1:  EQU     E005H
25 07A8  P              CONT2:  EQU     E006H
26 07A8  P              CONTF:  EQU     E007H
27 07A8  P              SUNDG:  EQU     E008H
28 07A8  P              TEMP:   EQU     E008H
29 07A8                         SKP     H
```



```
01 07A8                 ;
02 07A8                 ;
03 07A8                 ;   GET 1 LINE STATEMENT  *
04 07A8                 ;
05 07A8                 ;           DE = DATA STORE LOW ADR.
06 07A8                 ;                (END =CR )
07 07A8                 ;              ORG 07E6H
08 07A8                 ;
09 07A8                 ?GETL:  ENT
10 07A8 F5                      PUSH    AF
11 07A9 C5                      PUSH    BC
12 07AA E5                      PUSH    HL
13 07AB D5                      PUSH    DE
14 07AC CD6302          GETL0:  CALL    ?SAVE
15 07AF CDCA08                  CALL    ?KEY
16 07B2 FECB                    CP      CBH
17 07B4 28F9                    JR      Z,-5
18 07B6 CDCA08          GETL1:  CALL    ?KEY
19 07B9 CDFF09                  CALL    ?FLAS
20 07BC 28F8                    JR      Z,-6
21 07BE F5                      PUSH    AF
22 07BF AF                      XOR     A
23 07C0 329311                  LD      (STRGF),A
24 07C3 F1                      POP     AF
25 07C4 47              AUTO3:  LD      B,A
26 07C5 CDF505                  CALL    ?LOAD
27 07C8 3A9D11                  LD      A,(SWRK)        ; BELL WORK
28 07CB B7                      OR      A
29 07CC CCE502                  CALL    Z,?BEL
30 07CF 78                      LD      A,B
31 07D0 FEE7                    CP      E7H             ; 00
32 07D2 CAE105                  JP      Z,GETLD
33 07D5 FEE6                    CP      E6H             ; ↑[ PAGE MODE K
34 07D7 2868                    JR      Z,CHGPK
35 07D9 FEEE                    CP      EEH             ; ↑J PAGE MODE A
36 07DB 2861                    JR      Z,CHGPA
37 07DD FEE5                    CP      E5H             ; ↑Z CODE=5AH
38 07DF 2876                    JR      Z,DMT
39 07E1 FEE0                    CP      E0H
40 07E3 CA8B08                  JP      Z,LOCK
41 07E6 30CE                    JR      NC,GETL1
42 07E8 E6F0                    AND     F0H
43 07EA FEC0                    CP      C0H
44 07EC 2015                    JR      NZ,GETL2
45 07EE 78                      LD      A,B
46 07EF FECD                    CP      CDH             ; CR
47 07F1 2868                    JR      Z,GETL3
48 07F3 FECB                    CP      CBH             ; BREAK
49 07F5 CA4E08                  JP      Z,GETLC
50 07F8 FEC7                    CP      C7H             ; CRT EDITION
51 07FA 303D                    JR      NC,GETL5
52 07FC 3A7011                  LD      A,(KANAF)
53 07FF B7                      OR      A
54 0800 78                      LD      A,B
55 0801 2836                    JR      Z,GETL5
56 0803 78              GETL2:  LD      A,B
57 0804 CDB50D                  CALL    ?DSP
58 0807 CD6302          AUTO2:  CALL    ?SAVE
59 080A 3A9311                  LD      A,(STRGF)
60 080D B7                      OR      A
```

```
0846 3EC6            LD    A,C6H         ; CLRS CODE
0848 CDDC0D          CALL  ?DPCT
084B C3AC07  CHGP1:  JP    GETL0
084E                 ;
084E                 ;     BREAK IN
084E                 ;
084E E1      GETLC:  POP   HL
084F E5              PUSH  HL
0850 361B            LD    (HL),1BH
0852 23              INC   HL
0853 360D            LD    (HL),0DH
0855 1829            JR    GETLR
0857         ;
0857         ;
0857         ;
0857 065A    DMT:    LD    B,5AH         ; +
0859 18A8            JR    GETL2
085B         ;
085B                 ORG   085BH         ;GETL3    122
085B         ;
085B CD2B0A  GETL3:  CALL  .MANG         ; CR
085E 0628            LD    B,40          ; 1LINE
0860 3024            JR    NC,GETLA
0862 25              DEC   H             ; BEFORE LINE
0863 0650    GETLB:  LD    B,80          ; 2 LINE
0865 2E00    GETL6:  LD    L,0
0867 CDB40F          CALL  ?PNT1         ; STORE TOP ADR.
086A D1              POP   DE
086B D5              PUSH  DE
086C 7E              LD    A,(HL)
086D CDCE0B          CALL  ?DACN
0870 12              LD    (DE),A
```

```
0893                 ;
0893                 ;  MESSAGE PRINT  ;
0893                 ;
0893                 ;     DE = PRINT DATA LOW ADR.
0893                 ;       (END = CR)
0893                 ;
0893                 ?MSG:   ENT
0893 F5                      PUSH  AF
0894 C5                      PUSH  BC
0895 D5                      PUSH  DE
0896 1A              MSG1:   LD    A,(DE)
0897 FE0D                    CP    0DH          ; CR
0899 280C                    JR    Z,MSGX2
089B CD9509                  CALL  ?PRNT
089E 13                      INC   DE
089F 18F5                    JR    MSG1
08A1                 ;
08A1                 ?MSGX:  ENT                ; ALL PRINT MSG
08A1 F5                      PUSH  AF
08A2 C5                      PUSH  BC
08A3 D5                      PUSH  DE
08A4 1A              MSGX1:  LD    A,(DE)
08A5 FE0D                    CP    0DH
08A7 CAE60E          MSGX2:  JP    Z,?RSTR1
08AA CDB90B                  CALL  ?ADCN
08AD CD6C09                  CALL  PRNT3
08B0 13                      INC   DE
08B1 18F1                    JR    MSGX1
08B3                 ;
08B3                 ;
08B3                 ;   GET 1KEY
08B3                 ;  EXIT  ACC=ASCII
```

```
01 08B3                 ;   IF NO KEY   ACC=00
02 08B3                 ;
03 08B3                 ?GET:   ENT
04 08B3 C5                      PUSH  BC
05 08B4 E5                      PUSH  HL
06 08B5 0609                    LD    B,9
07 08B7 216511                  LD    HL,SWPW+1
08 08BA CDDB0F                  CALL  ?CLRFF
09 08BD E1                      POP   HL
10 08BE C1                      POP   BC
11 08BF CDCA08                  CALL  ?KEY
12 08C2 D6F0                    SUB   F0H
13 08C4 C8                      RET   Z
14 08C5 C6F0                    ADD   A,F0H
15 08C7 C3CE0B                  JP    ?DACN
16 08CA                 ;
17 08CA                 ;
18 08CA                         ORG   08CAH             ;?KEY   124
19 08CA                 ;
20 08CA                 ;   1KEY INPUT
21 08CA                 ;  IN       B = KEY MODE(SHIFT,CTRL,GRAPH)
22 08CA                 ;             C=KEY DATA (COLUMN & ROW)
23 08CA                 ;  EXIT    ACC=DISPLAY CODE
24 08CA                 ;             IF NO KEY   ACC=F0H
25 08CA C5              ?KEY:   PUSH  BC
26 08CB D5                      PUSH  DE
27 08CC E5                      PUSH  HL
28 08CD CD500A          ?KY8:   CALL  ?SWEP
29 08D0 78                      LD    A,B
30 08D1 07                      RLCA
31 08D2 381E                    JR    C,?KY2
32 08D4 3EF0                    LD    A,F0H
33 08D6 5F              ?KY1:   LD    E,A
34 08D7 CDF109                  CALL  AUTCK
35 08DA 3A6E11                  LD    A,(KDATH)
36 08DD B7                      OR    A
37 08DE 280A                    JR    Z,?KY11
38 08E0 CDA70D                  CALL  DLY12             ; 12 M SEC DELAY
39 08E3 CD500A                  CALL  ?SWEP
40 08E6 78                      LD    A,B
41 08E7 07                      RLCA
42 08E8 3808                    JR    C,?KY2
43 08EA 7B              ?KY11:  LD    A,E
44 08EB FEF0                    CP    F0H
45 08ED 2050                    JR    NZ,?KY9
46 08EF C39F06          ?KY10:  JP    RET3
47 08F2 07              ?KY2:   RLCA
48 08F3 07                      RLCA
49 08F4 07                      RLCA
50 08F5 DAE706                  JP    C,#GRP
51 08F8 07                      RLCA
52 08F9 DAC50B                  JP    C,#BRK
53 08FC 2600                    LD    H,0
54 08FE 69                      LD    L,C               ; HL<-BC
55 08FF 79                      LD    A,C               ; TENKEY CHECK
56 0900 FE38                    CP    38H
57 0902 3026                    JR    NC,?KY6           ; TEN KEY THEN JUMP
58 0904 3A7011                  LD    A,(KANAF)
59 0907 B7                      OR    A
60 0908 78                      LD    A,B
```



```
01 0909 07                      RLCA
02 090A 2023                    JR    NZ,?KY4
03 090C 47                      LD    B,A
04 090D 3A8F11                  LD    A,(SFTLK)
05 0910 B7                      OR    A
06 0911 78                      LD    A,B
07 0912 2803                    JR    Z,+5
08 0914 17                      RLA
09 0915 3F                      CCF
10 0916 1F                      RRA
11 0917 17                      RLA
12 0918 17                      RLA
13 0919 3007                    JR    NC,?KY3
14 091B 11DA0C                  LD    DE,KTBLC          ; CONTROL ON
15 091E 19              ?KY5:   ADD   HL,DE
16 091F 7E                      LD    A,(HL)            ; DISPLAY CODE
17 0920 18B4                    JR    ?KY1
18 0922 1F              ?KY3:   RRA                     ; CTRL OFF
19 0923 3005                    JR    NC,?KY6
20 0925 11320C                  LD    DE,KTBLS          ; SHIFT ON
21 0928 18F4                    JR    ?KY5
22 092A 11EA0B          ?KY6:   LD    DE,KTBL           ; NORMAL KEY
23 092D 18EF                    JR    ?KY5
24 092F 07              ?KY4:   RLCA
25 0930 3808                    JR    C,?KY7
26 0932 07                      RLCA
27 0933 38E6                    JR    C,?KY5-3          ; JUMP CTRL
28 0935 116A0C                  LD    DE,KTBLG          ; GRAPH MODE
29 0938 18E4                    JR    ?KY5
30 093A 11A20C          ?KY7:   LD    DE,KTBLGS         ;GRAPH MODE & SHIFT ON
31 093D 18DF                    JR    ?KY5
32 093F                 ;
33 093F CDF109          ?KY9:   CALL  AUTCK
34 0942 3C                      INC   A                 ; Z FLG CLR
35 0943 7B                      LD    A,E
36 0944 18A9                    JR    ?KY10
37 0946                 ;
38 0946                 ;
39 0946                 ;
40 0946                         ORG   0946H             ;?PRT   109
41 0946                 ;
42 0946                 ;
43 0946                 ;       PRINT ROUTINE
44 0946                 ;           1 CHA.
45 0946                 ;  C=ASCII DATA
46 0946                 ;
47 0946 79              ?PRT:   LD    A,C
48 0947 CDB90B                  CALL  ?ADCN
49 094A 4F                      LD    C,A
50 094B E6F0                    AND   F0H
51 094D FEF0                    CP    F0H               ; ILLEGAL DATA
52 094F C8                      RET   Z
53 0950 FEC0                    CP    C0H
54 0952 79                      LD    A,C
55 0953 2017                    JR    NZ,PRNT3
56 0955 CDDC0D          PRNT5:  CALL  ?DPCT
57 0958 FEC3                    CP    C3H
58 095A 2813                    JR    Z,PRNT4
59 095C FEC5                    CP    C5H               ; HOME
60 095E 2807                    JR    Z,PRNT2
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 31          09/04/81

0960 FECD                       CP    CDH             ; CR
0962 2803                       JR    Z,PRNT2
0964 FEC6                       CP    C6H             ; CLR
0966 C0                         RET   NZ
0967 AF                 PRNT2:  XOR   A
0968 329411                     LD    (DPRNT),A
096B C9                         RET
096C CDB50D             PRNT3:  CALL  ?DSP
096F 3A9411             PRNT4:  LD    A,(DPRNT)       ; TAB POINT +1
0972 3C                         INC   A
0973 FE50                       CP    +80
0975 3802                       JR    C,+4
0977 D650                       SUB   +80
0979 18ED                       JR    PRNT2+1
097B                    ;
097B                    ?NL:    ENT
097B 3A9411                     LD    A,(DPRNT)
097E B7                         OR    A
097F C8                         RET   Z
0980                    ;
0980                    ;
0980                    ;  NEW LINE
0980                    ;
0980                    ?LTNL:  ENT
0980 3ECD                       LD    A,CDH           ; CR
0982 18D1                       JR    PRNT5
0984                    ;
0984                    ;
0984                    ;  PRINT TAB  ;
0984                    ;
0984                    ?PRTT:  ENT
0984 CD0C00                     CALL  PRNTS
0987 3A9411                     LD    A,(DPRNT)
098A B7                         OR    A
098B C8                         RET   Z
098C D60A                       SUB   +10
098E 38F4                       JR    C,-10
0990 20FA                       JR    NZ,-4
0992 C9                         RET
0993                    ;
0993                    ;   PRINT SPACE
0993                    ;
0993                    ?PRTS:  ENT
0993 3E20                       LD    A,20H
0995                    ;
0995                    ;  PRINT ROUTINE  ;
0995                    ;
0995                    ?PRNT:  ENT
0995 FE0D                       CP    0DH
0997 28E7                       JR    Z,?LTNL
0999 C5                         PUSH  BC
099A 4F                         LD    C,A
099B 47                         LD    B,A
099C CD4609                     CALL  ?PRT
099F 78                         LD    A,B
09A0 C1                         POP   BC
09A1 C9                         RET
09A2                    ;
09A2                    ;
09A2                    ;   DLY3  324MICRO SEC DELAY
```

```
**  Z80 ASSEMBLER SB-7201  <MZ-80A.MONITOR>  PAGE 32          09/04/81

09A2                    ;
09A2                    ;       DLY2*3
09A2                    ;
09A2 ED44               DLY3:   NEG
09A4 ED44                       NEG
09A6 3E2A                       LD    A,42
09A8 C36207                     JP    DLY2+2
09AB                    ;
09AB                    ;   ONPU 2
09AB                    ;
09AB 81                 ONP3:   ADD   A,C
09AC 10FD                       DJNZ  -1
09AE C1                         POP   BC
09AF 4F                         LD    C,A
09B0 AF                         XOR   A
09B1 C9                         RET
09B2                    ;
09B2                    ;
09B2                    ;
09B3                            ORG   09B3H             ;??KEY   76
09B3                    ;
09B3                    ;  KEY BOAD SEARCH  ;
09B3                    ;    & DISPLAY CODE CONV.  ;
09B3                    ;
09B3                    ;   EXIT A = DISPLAY CODE
09B3                    ;   WITH CURSOR DISPLAY
09B3                    ;
09B3 C5                 ??KEY:  PUSH  BC
09B4 D5                         PUSH  DE
09B5 E5                         PUSH  HL
09B6 CD6302                     CALL  ?SAVE
09B9 CDCA08             KSL2:   CALL  ?KEY              ; KEY INPUT
09BC CDFF09                     CALL  ?FLAS
09BF 28F8                       JR    Z,KSL2
09C1 CDF505                     CALL  ?LOAD
09C4 C39F06                     JP    RET3
09C7                    ;
09C7                    ;
09C7                    ;     PAGE TOP CALCULATION;
09C7                    ;      (PAGETP)=(PBIAS)*8
09C7                    ;
09C7 D5                 PAGE:   PUSH  DE
09C8 E5                         PUSH  HL
09C9 217A11                     LD    HL,PBIAS
09CC AF                         XOR   A
09CD ED6F                       RLD
09CF 57                         LD    D,A
09D0 5E                         LD    E,(HL)
09D1 ED67                       RRD
09D3 AF                         XOR   A
09D4 CB1A                       RR    D
09D6 CB1B                       RR    E
09D8 2100D0                     LD    HL,SCRN
09DB 19                         ADD   HL,DE
09DC 227D11                     LD    (PAGETP),HL
09DF E1                         POP   HL
09E0 D1                         POP   DE
09E1 C9                         RET
09E2                    ;
09E2                    ;
```

```
09E2                    ;    CLEAR 2
09E2                    ;
09E2 AF                 #CLR08: XOR   A
09E3 010008             #CLR8:  LD    BC,0800H
09E6 D5                 CLEAR:  PUSH  DE
09E7 57                         LD    D,A
09E8 72                 CLEAR1: LD    (HL),D
09E9 23                         INC   HL
09EA 0B                         DEC   BC
09EB 78                         LD    A,B
09EC B1                         OR    C
09ED 20F9                       JR    NZ,CLEAR1
09EF D1                         POP   DE
09F0 C9                         RET
09F1                    ;
09F1                    ;
09F1                    ;    AUTO REPEAT CHECK
09F1                    ;
09F1 216E11             AUTCK:  LD    HL,KDATW
09F4 7E                         LD    A,(HL)
09F5 23                         INC   HL
09F6 56                         LD    D,(HL)
09F7 77                         LD    (HL),A
09F8 92                         SUB   D
09F9 D0                         RET   NC
09FA 34                         INC   (HL)
09FB C9                         RET
09FC                    ;
09FC                    ;     00 MESSAGE
09FC                    ;
09FC 3030               00MSG:  DEFW  3030H
09FE 0D                         DEFB  0DH
09FF                    ;
09FF                    ;
09FF                    ;
09FF                            ORG   09FFH             ;?FLAS   81
09FF                    ;
09FF                    ;
09FF                    ;
09FF                    ;  FLASHING  ;
09FF                    ;
09FF F5                 ?FLAS:  PUSH  AF
0A00 E5                         PUSH  HL
0A01 3A02E0                     LD    A,(KEYPC)
0A04 07                         RLCA
0A05 07                         RLCA
0A06 380A                       JR    C,FLAS1
0A08 3A9211                     LD    A,(FLSDT)
0A0B CDB10F             FLAS2:  CALL  ?PONT
0A0E 77                         LD    (HL),A
0A0F E1                 FLAS3:  POP   HL
0A10 F1                         POP   AF
0A11 C9                         RET
0A12 3A8E11             FLAS1:  LD    A,(FLASH)
0A15 18F4                       JR    FLAS2
0A17                    ;
0A17                    ;    REVERSE CRT
0A17                    ;
0A17 219011             REV:    LD    HL,REVFLG
0A1A 7E                         LD    A,(HL)
```



```
0A1B B7                         OR    A
0A1C 2F                         CPL
0A1D 77                         LD    (HL),A
0A1E 2805                       JR    Z,REV1
0A20 3A14E0                     LD    A,(E014H)         ; RESET
0A23 1803                       JR    +5
0A25 3A15E0             REV1:   LD    A,(E015H)         ; SET
0A28 C3E50E                     JP    ?RSTR
0A2B                    ;
0A2B                    ;
0A2B                    ;    CRT MANAGEMENT
0A2B                    ;      EXIT  HL:DSPXY
0A2B                    ;            DE:MANG ADR. (ON DSPXY)
0A2B                    ;            B :MANG BIT POSITION
0A2B                    ;            A :MANG DATA
0A2B                    ;            CY:MANG=1
0A2B                    ;
0A2B 217311             .MANG:  LD    HL,MANG
0A2E 3A9111                     LD    A,(SPAGE)
0A31 B7                         OR    A
0A32 C2A603                     JP    NZ,.MANG2         ; PAGE MODE1
0A35 3A7C11                     LD    A,(MGPNT)
0A38 D608                       SUB   08H
0A3A 23                         INC   HL
0A3B 30FB                       JR    NC,-3
0A3D C608                       ADD   A,08H
0A3F 4E                         LD    C,(HL)
0A40 2B                         DEC   HL
0A41 47                         LD    B,A
0A42 04                         INC   B
0A43 C5                         PUSH  BC
0A44 7E                         LD    A,(HL)
0A45 CB19                       RR    C
0A47 1F                         RRA
0A48 10FB                       DJNZ  -3
0A4A C1                         POP   BC
0A4B EB                         EX    DE,HL
0A4C 2A7111             .MANG1: LD    HL,(DSPXY)
0A4F C9                         RET
0A50                    ;
0A50                    ;
0A50                    ;;
0A50                            ORG   0A50H             ; ?SWEP #106#111
0A50                    ;
0A50                    ;
0A50                    ;   KEY BOAD SWEEP
0A50                    ;
0A50                    ;   EXIT B,D7=0  NO DATA
0A50                    ;           =1   DATA
0A50                    ;         D6=0   SHIFT OFF
0A50                    ;           -1   SHIFT ON
0A50                    ;        C   = ROW & COLOUMN
0A50                    ;
0A50 D5                 ?SWEP:  PUSH  DE
0A51 E5                         PUSH  HL
0A52 AF                         XOR   A
0A53 326E11                     LD    (KDATW),A         ; RESET KEY COUNTER
0A56 06FA                       LD    B,FAH
0A58 57                         LD    D,A
```

```
**  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 35          09/04/81

01 0A59 CD110D              CALL   ?BRK
02 0A5C 2004                JR     NZ,SWEP6
03 0A5E 1688                LD     D,88H            ; BREAK ON
04 0A60 1828                JR     SWEP9
05 0A62 216411      SWEP6:  LD     HL,SWPW
06 0A65 E5                  PUSH   HL
07 0A66 3026                JR     NC,SWEP11
08 0A68 57                  LD     D,A
09 0A69 E660                AND    60H              ; SHIFT & CTRL =NO DATA
10 0A6B 2021                JR     NZ,SWEP11        ; GRAPH CHECK
11 0A6D 7A                  LD     A,D
12 0A6E AE                  XOR    (HL)
13 0A6F CB67                BIT    4,A
14 0A71 72                  LD     (HL),D
15 0A72 2802                JR     Z,SWEP0
16 0A74 CBFA        SWEP01: SET    7,D
17 0A76 05          SWEP0:  DEC    B                ; SWEP COLOUMN WORK
18 0A77 E1                  POP    HL
19 0A78 23                  INC    HL
20 0A79 78                  LD     A,B
21 0A7A 3200E0              LD     (KEYPA),A
22 0A7D FEF0                CP     0F0H
23 0A7F 2011                JR     NZ,SWEP3
24 0A81 7E                  LD     A,(HL)           ; KDATW
25 0A82 FE03                CP     03H
26 0A84 3804                JR     C,SWEP9
27 0A86 3600                LD     (HL),0           ; RESET DATA IN
28 0A88 CBBA                RES    7,D
29 0A8A 42          SWEP9:  LD     B,D
30 0A8B E1                  POP    HL
31 0A8C D1                  POP    DE
32 0A8D C9                  RET
33 0A8E                     ;
34 0A8E 3600        SWEP11: LD     (HL),0
35 0A90 18E4                JR     SWEP0
36 0A92 3A01E0      SWEP3:  LD     A,(KEYPB)
37 0A95 5F                  LD     E,A
38 0A96 2F                  CPL
39 0A97 A6                  AND    (HL)
40 0A98 73                  LD     (HL),E
41 0A99 E5                  PUSH   HL
42 0A9A 216E11              LD     HL,KDATW
43 0A9D C5                  PUSH   BC
44 0A9E 0608                LD     B,8
45 0AA0 CB03        SWEP8:  RLC    E
46 0AA2 3801                JR     C,SWEP7
47 0AA4 34                  INC    (HL)
48 0AA5 10F9        SWEP7:  DJNZ   SWEP8
49 0AA7 C1                  POP    BC
50 0AA8 B7                  OR     A
51 0AA9 28CB                JR     Z,SWEP0
52 0AAB 5F                  LD     E,A
53 0AAC 2608        SWEP2:  LD     H,8
54 0AAE 78                  LD     A,B
55 0AAF 3D                  DEC    A                ; TBL ADJUST
56 0AB0 E60F                AND    0FH
57 0AB2 C33E07              JP     SWEP4
58 0AB5                     ;
59 0AB5                     ;  ASCII TO DISPLAY CODE TABL  ;
60 0AB5                     ;
```

```
**  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 36          09/04/81

01 0AB5                     ATBL:
02 0AB5                     ;  00 - 0F  ;
03 0AB5 CC                          DEFB   0CCH      ; ↑@
04 0AB6 E0                          DEFB   0E0H      ; ↑A SHIFT LOCK
05 0AB7 F2                          DEFB   0F2H      ; ↑B
06 0AB8 F3                          DEFB   0F3H      ; ↑C
07 0AB9 CE                          DEFB   0CEH      ; ↑D ROLL UP
08 0ABA CF                          DEFB   0CFH      ; ↑E ROLL DOWN
09 0ABB F6                          DEFB   0F6H      ; ↑F
10 0ABC F7                          DEFB   0F7H      ; ↑G
11 0ABD F8                          DEFB   0F8H      ; ↑H
12 0ABE F9                          DEFB   0F9H      ; ↑I
13 0ABF FA                          DEFB   0FAH      ; ↑J
14 0AC0 FB                          DEFB   0FBH      ; ↑K
15 0AC1 FC                          DEFB   0FCH      ; ↑L
16 0AC2 FD                          DEFB   0FDH      ; ↑M
17 0AC3 FE                          DEFB   0FEH      ; ↑N
18 0AC4 FF                          DEFB   0FFH      ; ↑O
19 0AC5                     ;  10 - 1F
20 0AC5 E1                          DEFB   0E1H      ; ↑P
21 0AC6 C1                          DEFB   0C1H      ; ↑Q CUR. DOWN
22 0AC7 C2                          DEFB   0C2H      ; ↑R CUR. UP
23 0AC8 C3                          DEFB   0C3H      ; ↑S CUR. RIGHT
24 0AC9 C4                          DEFB   0C4H      ; ↑T CUR. LEFT
25 0ACA C5                          DEFB   0C5H      ; ↑U HOME
26 0ACB C6                          DEFB   0C6H      ; ↑V CLEAR
27 0ACC E2                          DEFB   0E2H      ; ↑W
28 0ACD E3                          DEFB   0E3H      ; ↑X
29 0ACE E4                          DEFB   0E4H      ; ↑Y
30 0ACF E5                          DEFB   0E5H      ; ↑Z SEP.
31 0AD0 E6                          DEFB   0E6H      ; ↑[
32 0AD1 EB                          DEFB   0EBH      ; ↑\
33 0AD2 EE                          DEFB   0EEH      ; ↑]
34 0AD3 EF                          DEFB   0EFH      ; ↑^
35 0AD4 F4                          DEFB   0F4H      ; ↑-
36 0AD5                     ;  20 - 2F  ;
37 0AD5 00                          DEFB   00H       ; SPACE
38 0AD6 61                          DEFB   61H       ; !
39 0AD7 62                          DEFB   62H       ; "
40 0AD8 63                          DEFB   63H       ; #
41 0AD9 64                          DEFB   64H       ; $
42 0ADA 65                          DEFB   65H       ; %
43 0ADB 66                          DEFB   66H       ; &
44 0ADC 67                          DEFB   67H       ; '
45 0ADD 68                          DEFB   68H       ; (
46 0ADE 69                          DEFB   69H       ; )
47 0ADF 6B                          DEFB   6BH       ; *
48 0AE0 6A                          DEFB   6AH       ; +
49 0AE1 2F                          DEFB   2FH       ; ,
50 0AE2 2A                          DEFB   2AH       ; -
51 0AE3 2E                          DEFB   2EH       ; .
52 0AE4 2D                          DEFB   2DH       ; /
53 0AE5                     ;  30 -3F  ;
54 0AE5 20                          DEFB   20H       ; 0
55 0AE6 21                          DEFB   21H       ; 1
56 0AE7 22                          DEFB   22H       ; 2
57 0AE8 23                          DEFB   23H       ; 3
58 0AE9 24                          DEFB   24H       ; 4
59 0AEA 25                          DEFB   25H       ; 5
60 0AEB 26                          DEFB   26H       ; 6
```

```
0AEC 27                         DEFB  27H             ; 7
0AED 28                         DEFB  28H             ; 8
0AEE 29                         DEFB  29H             ; 9
0AEF 4F                         DEFB  4FH             ; :
0AF0 2C                         DEFB  2CH             ; ;
0AF1 51                         DEFB  51H             ; <
0AF2 2B                         DEFB  2BH             ; =
0AF3 57                         DEFB  57H             ; >
0AF4 49                         DEFB  49H             ; ?
0AF5                    ;  40 - 4F  ;
0AF5 55                         DEFB  55H             ; @
0AF6 01                         DEFB  01H             ; A
0AF7 02                         DEFB  02H             ; B
0AF8 03                         DEFB  03H             ; C
0AF9 04                         DEFB  04H             ; D
0AFA 05                         DEFB  05H             ; E
0AFB 06                         DEFB  06H             ; F
0AFC 07                         DEFB  07H             ; G
0AFD 08                         DEFB  08H             ; H
0AFE 09                         DEFB  09H             ; I
0AFF 0A                         DEFB  0AH             ; J
0B00 0B                         DEFB  0BH             ; K
0B01 0C                         DEFB  0CH             ; L
0B02 0D                         DEFB  0DH             ; M
0B03 0E                         DEFB  0EH             ; U
0B04 0F                         DEFB  0FH             ; O
0B05                    ;  50 - 5F  ;
0B05 10                         DEFB  10H             ; P
0B06 11                         DEFB  11H             ; Q
0B07 12                         DEFB  12H             ; R
0B08 13                         DEFB  13H             ; S
0B09 14                         DEFB  14H             ; T
0B0A 15                         DEFB  15H             ; U
0B0B 16                         DEFB  16H             ; V
0B0C 17                         DEFB  17H             ; W
0B0D 18                         DEFB  18H             ; X
0B0E 19                         DEFB  19H             ; Y
0B0F 1A                         DEFB  1AH             ; Z
0B10 52                         DEFB  52H             ; [
0B11 59                         DEFB  59H             ; \
0B12 54                         DEFB  54H             ; ]
0B13 50                         DEFB  50H             ; ^
0B14 45                         DEFB  45H             ; _
0B15                    ;  60 - 6F  ;
0B15 C7                         DEFB  C7H             ; UFO
0B16 C8                         DEFB  C8H
0B17 C9                         DEFB  C9H
0B18 CA                         DEFB  CAH
0B19 CB                         DEFB  CBH
0B1A CC                         DEFB  CCH
0B1B CD                         DEFB  CDH
0B1C CE                         DEFB  CEH
0B1D CF                         DEFB  CFH
0B1E DF                         DEFB  DFH
0B1F E7                         DEFB  E7H
0B20 E8                         DEFB  E8H
0B21 E9                         DEFB  E9H
0B22 EA                         DEFB  EAH
0B23 EC                         DEFB  ECH
0B24 ED                         DEFB  EDH
```



```
0B25                    ;  70 - 7F  ;
0B25 D0                         DEFB  D0H
0B26 D1                         DEFB  D1H
0B27 D2                         DEFB  D2H
0B28 D3                         DEFB  D3H
0B29 D4                         DEFB  D4H
0B2A D5                         DEFB  D5H
0B2B D6                         DEFB  D6H
0B2C D7                         DEFB  D7H
0B2D D8                         DEFB  D8H
0B2E D9                         DEFB  D9H
0B2F DA                         DEFB  DAH
0B30 DB                         DEFB  DBH
0B31 DC                         DEFB  DCH
0B32 DD                         DEFB  DDH
0B33 DE                         DEFB  DEH
0B34 C0                         DEFB  C0H
0B35                    ;  80 - 8F  ;
0B35 40                         DEFB  40H             ; }
0B36 BD                         DEFB  BDH             ;
0B37 9D                         DEFB  9DH
0B38 B1                         DEFB  B1H
0B39 B5                         DEFB  B5H
0B3A B9                         DEFB  B9H
0B3B B4                         DEFB  B4H             ; !!O
0B3C 9E                         DEFB  9EH             ; !!A
0B3D B2                         DEFB  B2H             ; !!I
0B3E B6                         DEFB  B6H             ; !!U
0B3F BA                         DEFB  BAH             ; !!E
0B40 BE                         DEFB  BEH             ; ^
0B41 9F                         DEFB  9FH             ; !!YA
0B42 B3                         DEFB  B3H             ; !!YU
0B43 B7                         DEFB  B7H             ; !!YO
0B44 BB                         DEFB  BBH             ; !!TSU
0B45                    ;  90 - 9F  ;
0B45 BF                         DEFB  BFH             ; —
0B46 A3                         DEFB  A3H             ; !A
0B47 85                         DEFB  85H             ; !I
0B48 A4                         DEFB  A4H             ; `
0B49 A5                         DEFB  A5H             ; ~
0B4A A6                         DEFB  A6H             ; !0
0B4B 94                         DEFB  94H             ; !KA
0B4C 87                         DEFB  87H             ; !KI
0B4D 88                         DEFB  88H             ; !KU
0B4E 9C                         DEFB  9CH             ; !KE
0B4F 82                         DEFB  82H             ; !KO
0B50 98                         DEFB  98H             ; !SA
0B51 84                         DEFB  84H             ; !SHI
0B52 92                         DEFB  92H             ; !SU
0B53 90                         DEFB  90H             ; !SE
0B54 83                         DEFB  83H             ; !SO
0B55                    ;  A0 - AF  ;
0B55 91                         DEFB  91H             ; !TA
0B56 81                         DEFB  81H             ; !CHI
0B57 9A                         DEFB  9AH             ; !TSU
0B58 97                         DEFB  97H             ; !TE
0B59 93                         DEFB  93H             ; !TO
0B5A 95                         DEFB  95H             ; !NA
0B5B 89                         DEFB  89H             ; !NI
0B5C A1                         DEFB  A1H             ; !NU
```

```
0B5D AF                         DEFB  AFH               ; !NE
0B5E 8B                         DEFB  8BH               ; !NO
0B5F 86                         DEFB  86H               ; !HA
0B60 96                         DEFB  96H               ; !HI
0B61 A2                         DEFB  A2H               ; !FU
0B62 AB                         DEFB  ABH               ; !HE
0B63 AA                         DEFB  AAH               ; !HO
0B64 8A                         DEFB  8AH               ; !MA
0B65                    ;  B0 - BF  ;
0B65 8E                         DEFB  8EH               ; !MI
0B66 B0                         DEFB  B0H               ; !MU
0B67 AD                         DEFB  ADH               ; !ME
0B68 8D                         DEFB  8DH               ; !MO
0B69 A7                         DEFB  A7H               ; !YA
0B6A A8                         DEFB  A8H               ; !YU
0B6B A9                         DEFB  A9H               ; !YO
0B6C 8F                         DEFB  8FH               ; !RA
0B6D 8C                         DEFB  8CH               ; !RI
0B6E AE                         DEFB  AEH               ; !RU
0B6F AC                         DEFB  ACH               ; !RE
0B70 9B                         DEFB  9BH               ; !RO
0B71 A0                         DEFB  A0H               ; !WA
0B72 99                         DEFB  99H               ; !N
0B73 BC                         DEFB  BCH               ; (
0B74 B8                         DEFB  B8H
0B75                    ;  C0 - CF  ;
0B75 80                         DEFB  80H               ; )
0B76 3B                         DEFB  3BH
0B77 3A                         DEFB  3AH
0B78 70                         DEFB  70H
0B79 3C                         DEFB  3CH
0B7A 71                         DEFB  71H
0B7B 5A                         DEFB  5AH
0B7C 3D                         DEFB  3DH
0B7D 43                         DEFB  43H
0B7E 56                         DEFB  56H
0B7F 3F                         DEFB  3FH
0B80 1E                         DEFB  1EH
0B81 4A                         DEFB  4AH
0B82 1C                         DEFB  1CH
0B83 5D                         DEFB  5DH
0B84 3E                         DEFB  3EH
0B85                    ;  D0 - DF  ;
0B85 5C                         DEFB  5CH
0B86 1F                         DEFB  1FH
0B87 5F                         DEFB  5FH
0B88 5E                         DEFB  5EH
0B89 37                         DEFB  37H
0B8A 7B                         DEFB  7BH
0B8B 7F                         DEFB  7FH
0B8C 36                         DEFB  36H
0B8D 7A                         DEFB  7AH
0B8E 7E                         DEFB  7EH
0B8F 33                         DEFB  33H
0B90 4B                         DEFB  4BH
0B91 4C                         DEFB  4CH
0B92 1D                         DEFB  1DH
0B93 6C                         DEFB  6CH
0B94 5B                         DEFB  5BH
0B95                    ;  E0 - EF  ;
```



```
0B95 78                         DEFB  78H
0B96 41                         DEFB  41H
0B97 35                         DEFB  35H
0B98 34                         DEFB  34H
0B99 74                         DEFB  74H
0B9A 30                         DEFB  30H
0B9B 38                         DEFB  38H
0B9C 75                         DEFB  75H
0B9D 39                         DEFB  39H
0B9E 4D                         DEFB  4DH
0B9F 6F                         DEFB  6FH
0BA0 6E                         DEFB  6EH
0BA1 32                         DEFB  32H
0BA2 77                         DEFB  77H
0BA3 76                         DEFB  76H
0BA4 72                         DEFB  72H
0BA5                    ;  F0 - FF  ;
0BA5 73                         DEFB  73H
0BA6 47                         DEFB  47H
0BA7 7C                         DEFB  7CH
0BA8 53                         DEFB  53H
0BA9 31                         DEFB  31H
0BAA 4E                         DEFB  4EH
0BAB 6D                         DEFB  6DH
0BAC 48                         DEFB  48H
0BAD 46                         DEFB  46H
0BAE 7D                         DEFB  7DH
0BAF 44                         DEFB  44H
0BB0 1B                         DEFB  1BH
0BB1 58                         DEFB  58H
0BB2 79                         DEFB  79H
0BB3 42                         DEFB  42H
0BB4 60                         DEFB  60H
0BB5                    ;
0BB9                            ORG   0BB9H             ;?ADCN   21
0BB9                    ;
0BB9                    ;
0BB9                    ;       ASCII TO DISPLAY CODE CONVERTE
0BB9                    ;
0BB9                    ;           IN   ACC:ASCII
0BB9                    ;           EXIT ACC:DISPLAY CODE
0BB9                    ;
0BB9 C5                 ?ADCN:  PUSH  BC
0BBA E5                         PUSH  HL
0BBB 21B50A                     LD    HL,ATBL
0BBE 4F                         LD    C,A
0BBF 0600                       LD    B,0
0BC1 09                         ADD   HL,BC
0BC2 7E                         LD    A,(HL)
0BC3 181B                       JR    DACN3
0BC5                    ;
0BC5                    ;
0BC5                    ;    ?KEY BREAK KEY INPUT
0BC5                    ;
0BC5 3ECB               #BRK:   LD    A,CBH             ; BREAK CODE
0BC7 B7                         OR    A
0BC8 C3EF08                     JP    ?KY10
0BCB                    ;
0BCB                    ;
0BCB                    ;
```

```
                                ORG   0BCEH            ;?DACN   472
                        ;
                        ;  DISPLAY CODE TO ASCII CONV.  ;
                        ;
                        ;     ACC = DISPLAY CODE
                        ;   EXIT ACC = ASCII
                        ;
0BCE C5                 ?DACN:  PUSH  BC
0BCF E5                         PUSH  HL
0BD0 D5                         PUSH  DE
0BD1 21B50A                     LD    HL,ATBL
0BD4 54                         LD    D,H
0BD5 5D                         LD    E,L
0BD6 010001                     LD    BC,0100H
0BD9 EDB1                       CPIR
0BDB 2806                       JR    Z,DACN1
0BDD 3EF0                       LD    A,F0H
0BDF D1                 DACN2:  POP   DE
0BE0 E1                 DACN3:  POP   HL
0BE1 C1                         POP   BC
0BE2 C9                         RET
0BE3                    ;
0BE3 B7                 DACN1:  OR    A
0BE4 2B                         DEC   HL
0BE5 ED52                       SBC   HL,DE
0BE7 7D                         LD    A,L
0BE8 18F5                       JR    DACN2
0BEA                    ;
0BEA                    ;
0BEA                    ;  KEY MATRIX TO DISPLAY CODE TABL
0BEA                    ;
0BEA                    KTBL:
0BEA                    ;S0   00 - 07  ;
0BEA 22                         DEFB  22H               ; 2
0BEB 21                         DEFB  21H               ; 1
0BEC 17                         DEFB  17H               ; W
0BED 11                         DEFB  11H               ; Q
0BEE 01                         DEFB  01H               ; A
0BEF C7                         DEFB  C7H               ; DEL
0BF0 00                         DEFB  00H               ; NULL
0BF1 1A                         DEFB  1AH               ; Z
0BF2                    ;S1   08 - 0F  ;
0BF2 24                         DEFB  24H               ; 4
0BF3 23                         DEFB  23H               ; 3
0BF4 12                         DEFB  12H               ; R
0BF5 05                         DEFB  05H               ; E
0BF6 04                         DEFB  04H               ; D
0BF7 13                         DEFB  13H               ; S
0BF8 18                         DEFB  18H               ; X
0BF9 03                         DEFB  03H               ; C
0BFA                    ;S2   0 - 17  ;
0BFA 26                         DEFB  26H               ; 6
0BFB 25                         DEFB  25H               ; 5
0BFC 19                         DEFB  19H               ; Y
0BFD 14                         DEFB  14H               ; T
0BFE 07                         DEFB  07H               ; G
0BFF 06                         DEFB  06H               ; F
0C00 16                         DEFB  16H               ; V
0C01 02                         DEFB  02H               ; B
```



```
0C02                    ;S3   18 - 1F  ;
0C02 28                         DEFB  28H               ; 8
0C03 27                         DEFB  27H               ; 7
0C04 09                         DEFB  09H               ; I
0C05 15                         DEFB  15H               ; U
0C06 0A                         DEFB  0AH               ; J
0C07 08                         DEFB  08H               ; H
0C08 0E                         DEFB  0EH               ; N
0C09 00                         DEFB  00H               ; SPACE
0C0A                    ;S4   20 - 27  ;
0C0A 20                         DEFB  20H               ; 0
0C0B 29                         DEFB  29H               ; 9
0C0C 10                         DEFB  10H               ; P
0C0D 0F                         DEFB  0FH               ; O
0C0E 0C                         DEFB  0CH               ; L
0C0F 0B                         DEFB  0BH               ; K
0C10 2F                         DEFB  2FH               ; ,
0C11 0D                         DEFB  0DH               ; M
0C12                    ;S5   28 - 2F  ;
0C12 BE                         DEFB  BEH               ; ^
0C13 2A                         DEFB  2AH               ; -
0C14 52                         DEFB  52H               ; [
0C15 55                         DEFB  55H               ; @
0C16 4F                         DEFB  4FH               ; :
0C17 2C                         DEFB  2CH               ; ;
0C18 2D                         DEFB  2DH               ; /
0C19 2E                         DEFB  2EH               ; .
0C1A                    ;S6   30 - 37  ;
0C1A C5                         DEFB  C5H               ; HOME
0C1B 59                         DEFB  59H               ; \
0C1C C3                         DEFB  C3H               ; CURSOL →RIGHT
0C1D C2                         DEFB  C2H               ; CURSOL ↑UP
0C1E CD                         DEFB  CDH               ; CR
0C1F 54                         DEFB  54H               ; ]
0C20 00                         DEFB  00H               ;NULL
0C21 49                         DEFB  49H               ; ?
0C22                    ;S7   38 - 3F  ;
0C22 28                         DEFB  28H               ; 8
0C23 27                         DEFB  27H               ; 7
0C24 25                         DEFB  25H               ; 5
0C25 24                         DEFB  24H               ; 4
0C26 22                         DEFB  22H               ; 2
0C27 21                         DEFB  21H               ; 1
0C28 E7                         DEFB  E7H               ; 00  ***
0C29 20                         DEFB  20H               ; 0
0C2A                    ;S8   40 - 47  ;
0C2A 6A                         DEFB  6AH               ; +
0C2B 29                         DEFB  29H               ; 9
0C2C 2A                         DEFB  2AH               ; -
0C2D 26                         DEFB  26H               ; 6
0C2E 00                         DEFB  00H               ; NULL
0C2F 23                         DEFB  23H               ; 3
0C30 00                         DEFB  00H               ;NULL
0C31 2E                         DEFB  2EH               ;.
0C32                    ;    KTBL SHIFT ON
0C32                    KTBLS:
0C32                    ;S0
0C32 62                         DEFB  62H               ; "
0C33 61                         DEFB  61H               ; !
0C34 97                         DEFB  97H               ; w
```

```
0C35 91          DEFB  91H          ; q
0C36 81          DEFB  81H          ; a
0C37 C8          DEFB  C8H          ; INSERT
0C38 00          DEFB  00H          ; NULL
0C39 9A          DEFB  9AH          ; z
0C3A             ;S1
0C3A 64          DEFB  64H          ; $
0C3B 63          DEFB  63H          ; #
0C3C 92          DEFB  92H          ; r
0C3D 85          DEFB  85H          ; e
0C3E 84          DEFB  84H          ; d
0C3F 93          DEFB  93H          ; s
0C40 98          DEFB  98H          ; x
0C41 83          DEFB  83H          ; c
0C42             ;S2
0C42 66          DEFB  66H          ; &
0C43 65          DEFB  65H          ; %
0C44 99          DEFB  99H          ; y
0C45 94          DEFB  94H          ; t
0C46 87          DEFB  87H          ; g
0C47 86          DEFB  86H          ; f
0C48 96          DEFB  96H          ; v
0C49 82          DEFB  82H          ; b
0C4A             ;S3
0C4A 68          DEFB  68H          ; (
0C4B 67          DEFB  67H          ; '
0C4C 89          DEFB  89H          ; i
0C4D 95          DEFB  95H          ; u
0C4E 8A          DEFB  8AH          ; j
0C4F 88          DEFB  88H          ; h
0C50 8E          DEFB  8EH          ; n
0C51 00          DEFB  00H          ; SPACE
0C52             ;S4
0C52 BF          DEFB  BFH          ; _
0C53 69          DEFB  69H          ; )
0C54 90          DEFB  90H          ; p
0C55 8F          DEFB  8FH          ; o
0C56 8C          DEFB  8CH          ; l
0C57 8B          DEFB  8BH          ; k
0C58 51          DEFB  51H          ; <
0C59 8D          DEFB  8DH          ; m
0C5A             ;S5
0C5A A5          DEFB  A5H          ; ~
0C5B 2B          DEFB  2BH          ; =
0C5C BC          DEFB  BCH          ; {
0C5D A4          DEFB  A4H          ; `
0C5E 6B          DEFB  6BH          ; *
0C5F 6A          DEFB  6AH          ; +
0C60 45          DEFB  45H          ; ¥
0C61 57          DEFB  57H          ; >
0C62             ;S6
0C62 C6          DEFB  C6H          ; CLR
0C63 80          DEFB  80H          ; |
0C64 C4          DEFB  C4H          ; CURSOL ¥LEFT
0C65 C1          DEFB  C1H          ; CURSOL ¥DOWN
0C66 CD          DEFB  CDH          ; CR
0C67 40          DEFB  40H          ; }
0C68 00          DEFB  00H          ; NULL
0C69 50          DEFB  50H          ; ↑
0C6A             KTBLG1:
```



```
0C6A             ;S0    GRAPHIC MODE
0C6A 3E          DEFB  3EH          ; #2
0C6B 37          DEFB  37H          ; #1
0C6C 38          DEFB  38H          ; #W
0C6D 3C          DEFB  3CH          ; #Q
0C6E 53          DEFB  53H          ; #A
0C6F C7          DEFB  C7H          ; DELETE
0C70 00          DEFB  00H          ; NULL
0C71 76          DEFB  76H          ; #Z
0C72             ;S1
0C72 7B          DEFB  7BH          ; #4
0C73 7F          DEFB  7FH          ; #3
0C74 30          DEFB  30H          ; #R
0C75 34          DEFB  34H          ; #E
0C76 47          DEFB  47H          ; #D
0C77 44          DEFB  44H          ; #S
0C78 6D          DEFB  6DH          ; #X
0C79 DE          DEFB  DEH          ; #C
0C7A             ;S2
0C7A 5E          DEFB  5EH          ; #6
0C7B 3A          DEFB  3AH          ; #5
0C7C 75          DEFB  75H          ; #Y
0C7D 71          DEFB  71H          ; #T
0C7E 4B          DEFB  4BH          ; #G
0C7F 4A          DEFB  4AH          ; #F
0C80 DA          DEFB  DAH          ; #V
0C81 6F          DEFB  6FH          ; #B
0C82             ;S3
0C82 BD          DEFB  BDH          ; #8
0C83 1F          DEFB  1FH          ; #7
0C84 7D          DEFB  7DH          ; #I
0C85 79          DEFB  79H          ; #U
0C86 5C          DEFB  5CH          ; #J
0C87 72          DEFB  72H          ; #H
0C88 32          DEFB  32H          ; #N
0C89 00          DEFB  00H          ; SPACE
0C8A             ;S4
0C8A 9C          DEFB  9CH          ; #0
0C8B A1          DEFB  A1H          ; #9
0C8C D6          DEFB  D6H          ; #P
0C8D B0          DEFB  B0H          ; #O
0C8E B4          DEFB  B4H          ; #L
0C8F 5B          DEFB  5BH          ; #K
0C90 60          DEFB  60H          ; ¥PAI,#,
0C91 1C          DEFB  1CH          ; #M
0C92             ;S5
0C92 9E          DEFB  9EH          ; #^
0C93 B2          DEFB  D2H          ; #-
0C94 D8          DEFB  D8H          ; #[
0C95 B2          DEFB  B2H          ; #@
0C96 B6          DEFB  B6H          ; #:
0C97 42          DEFB  42H          ; #;
0C98 DB          DEFB  DBH          ; #/
0C99 B8          DEFB  B8H          ; #.
0C9A             ;S6
0C9A C5          DEFB  C5H          ; HOME
0C9B D4          DEFB  D4H          ; #\
0C9C C3          DEFB  C3H          ; CURSOL ¥RIGHT
0C9D C2          DEFB  C2H          ; CURSOL ¥UP
0C9E CD          DEFB  CDH          ; CR
```

```
0C9F 4E                 DEFB  4EH             ; #J
0CA0 00                 DEFB  00H             ; NULL
0CA1 BA                 DEFB  BAH             ; #?
0CA2                KTBLGS:
0CA2                ;S0     GRAPHIC MODE & SHIFT ON
0CA2 36                 DEFB  36H             ; ?2
0CA3 3F                 DEFB  3FH             ; ?1
0CA4 78                 DEFB  78H             ; ?W
0CA5 7C                 DEFB  7CH             ; ?@
0CA6 46                 DEFB  46H             ; ?A
0CA7 C8                 DEFB  C8H             ; INST
0CA8 00                 DEFB  00H             ; NULL
0CA9 77                 DEFB  77H             ; ?Z
0CAA                ;S1
0CAA 3B                 DEFB  3BH             ; ?4
0CAB 7E                 DEFB  7EH             ; ?3
0CAC 70                 DEFB  70H             ; ?R
0CAD 74                 DEFB  74H             ; ?E
0CAE 48                 DEFB  48H             ; ?D
0CAF 41                 DEFB  41H             ; ?S
0CB0 DD                 DEFB  DDH             ; ?X
0CB1 D9                 DEFB  D9H             ; ?C
0CB2                ;S2
0CB2 1E                 DEFB  1EH             ; ?6
0CB3 7A                 DEFB  7AH             ; ?5
0CB4 35                 DEFB  35H             ; ?Y
0CB5 31                 DEFB  31H             ; ?T
0CB6 4C                 DEFB  4CH             ; ?G
0CB7 43                 DEFB  43H             ; ?F
0CB8 A6                 DEFB  A6H             ; ?V
0CB9 6E                 DEFB  6EH             ; ?B
0CBA                ;S3
0CBA A2                 DEFB  A2H             ; ?8
0CBB 5F                 DEFB  5FH             ; ?7
0CBC 3D                 DEFB  3DH             ; ?1
0CBD 39                 DEFB  39H             ; ?U
0CBE 5D                 DEFB  5DH             ; ?J
0CBF 73                 DEFB  73H             ; ?H
0CC0 33                 DEFB  33H             ; ?N
0CC1 00                 DEFB  00H             ; SPACE
0CC2                ;S4
0CC2 9D                 DEFB  9DH             ; ?0
0CC3 A3                 DEFB  A3H             ; ?9
0CC4 B1                 DEFB  B1H             ; ?P
0CC5 D5                 DEFB  D5H             ; ?O
0CC6 56                 DEFB  56H             ; ?L
0CC7 6C                 DEFB  6CH             ; ?K
0CC8 D0                 DEFB  D0H             ; ?,
0CC9 1D                 DEFB  1DH             ; ?M
0CCA                ;S5B
0CCA 9F                 DEFB  9FH             ; ?^
0CCB D1                 DEFB  D1H             ; ?-
0CCC B3                 DEFB  B3H             ; ?[
0CCD D7                 DEFB  D7H             ; ?@
0CCE 4D                 DEFB  4DH             ; ?;
0CCF B5                 DEFB  B5H             ; ?:
0CD0 1B                 DEFB  1BH             ; ?/
0CD1 B9                 DEFB  B9H             ; ?.
0CD2                ;S6
0CD2 C6                 DEFB  C6H             ; CLR
```



```
0CD3 D3                 DEFB  D3H             ; ?\
0CD4 C4                 DEFB  C4H             ; CURSOL →LEFT
0CD5 C1                 DEFB  C1H             ; CURSOL ↓DOWN
0CD6 CD                 DEFB  CDH             ; CR
0CD7 B7                 DEFB  B7H             ; ?]
0CD8 00                 DEFB  00H             ; NULL
0CD9 BB                 DEFB  BBH             ; ??
0CDA                ;
0CDA                KTBLC:
0CDA                ;S0     CTRL ON
0CDA F0                 DEFB  F0H             ; CODE 80H=NOT KEY
0CDB F0                 DEFB  F0H
0CDC E2                 DEFB  E2H             ; ↑W
0CDD C1                 DEFB  C1H             ; ↑Q
0CDE E0                 DEFB  E0H             ; ↑A SHIFT LOCK
0CDF F0                 DEFB  F0H
0CE0 00                 DEFB  00H
0CE1 E5                 DEFB  E5H             ; ↑Z
0CE2                ;S1
0CE2 F0                 DEFB  F0H
0CE3 F0                 DEFB  F0H
0CE4 C2                 DEFB  C2H             ; ↑R
0CE5 CF                 DEFB  CFH             ; ↑E ROLL DOWN
0CE6 CE                 DEFB  CEH             ; ↑D ROLL UP
0CE7 C3                 DEFB  C3H             ; ↑S
0CE8 E3                 DEFB  E3H             ; ↑X
0CE9 F3                 DEFB  F3H             ; ↑C
0CEA                ;S2
0CEA F0                 DEFB  F0H
0CEB F0                 DEFB  F0H
0CEC E4                 DEFB  E4H             ; ↑Y
0CED C4                 DEFB  C4H             ; ↑T
0CEE F7                 DEFB  F7H             ; ↑G
0CEF F6                 DEFB  F6H             ; ↑F
0CF0 C6                 DEFB  C6H             ; ↑V CLR
0CF1 F2                 DEFB  F2H             ; ↑B
0CF2                ;S3
0CF2 F0                 DEFB  F0H
0CF3 F0                 DEFB  F0H
0CF4 F9                 DEFB  F9H             ; ↑I
0CF5 C5                 DEFB  C5H             ; ↑U HOME
0CF6 FA                 DEFB  FAH             ; ↑J
0CF7 F8                 DEFB  F8H             ; ↑H
0CF8 FE                 DEFB  FEH             ; ↑N
0CF9 F0                 DEFB  F0H
0CFA                ;S4
0CFA F0                 DEFB  F0H
0CFB F0                 DEFB  F0H
0CFC E1                 DEFB  E1H             ; ↑P
0CFD FF                 DEFB  FFH             ; ↑O
0CFE FC                 DEFB  FCH             ; ↑L
0CFF FB                 DEFB  FBH             ; ↑K
0D00 F0                 DEFB  F0H
0D01 FD                 DEFB  FDH             ; ↑M
0D02                ;S5
0D02 EF                 DEFB  EFH             ; ↑^
0D03 F4                 DEFB  F4H             ; ↑-
0D04 E6                 DEFB  E6H             ; ↑[
0D05 CC                 DEFB  CCH             ; ↑@ REVERSE
0D06 F0                 DEFB  F0H
```

```
                        DEFB  F0H
                        DEFB  F0H
                        DEFB  F0H
                ;S6
                        DEFB  F0H
                        DEFB  EBH               ; ↑\
                        DEFB  F0H
                        DEFB  F0H
                        DEFB  F0H
                        DEFB  EEH               ; ↑]
                        DEFB  F0H
                ;
                ;
                ;
                ;  BREAK KEY CHECK  ;
                ;      AND SHIFT,GRPH,CTRL KEY CHECK
                ;
                ;  EXIT   BREAK ON : ZERO = 1
                ;          BREAK OFF : ZERO = 0
                ;         BREAK OFF: ZERO = 0
                ;         NOT KEY   :  CY=0
                ;         KEY IN    :  CY=1
                ;          A D6=1   :  SHIFT ON
                ;             =0    :         OFF
                ;            D5=1   :  CTRL  ON
                ;              =0   :         OFF
                ;            D4=1   :  GRPH  ON
                ;              =0   :         OFF
                ;
                ;
0D11 3EF0       ?BRK:   ENT
0D13 3200E0             LD    A,F0              ; LINE 0 SWEEP
0D16 00                 LD    (KEYPA),A
0D17 3A01E0             NOP
0D1A B7                 LD    A,(KEYPB)
0D1B 17                 OR    A
0D1C 3019               RLA
0D1E 1F                 JR    NC,?BRK1          ;CTRL?
0D1F 1F                 RRA
0D20 3005               RRA
0D22 1F                 JR    NC,?BRK2          ; SHIFT?
0D23 3006               RRA
0D25 3F                 JR    NC,?BRK3          ; GRPH ?
0D26 C9                 CCF                     ; NOT KEY
0D27 3E40               RET
0D29 37         ?BRK2:  LD    A,40H             ; A D6=1 SHIFT ON
0D2A C9                 SCF
0D2B 3A6E11             RET
0D2E 3E01       ?BRK3:  LD    A,(KDATW)
0D30 326E11             LD    A,1
0D33 3E10               LD    (KDATW),A
0D35 37                 LD    A,10H             ; A:D4=1 GRPH
0D36 C9                 SCF
0D37 E602               RET
0D39 C8         ?BRK1:  AND   02H               ; SHIFT ?
0D3A 3E20               RET   Z
0D3C 37                 LD    A,20H             ; A:D5=1  CTRL
0D3D C9                 SCF
0D3E                    RET
                ;;
                ;   1 BIT WRITE
```



```
0D3E                    ;
0D3E F5         SHORT:  PUSH  AF
0D3F 3E03               LD    A,03H
0D41 3203E0             LD    (CSTPT),A
0D44 CD5907             CALL  DLY1
0D47 CD5907             CALL  DLY1
0D4A 3E02               LD    A,02H
0D4C 3203E0             LD    (CSTPT),A
0D4F CD5907             CALL  DLY1
0D52 CD5907             CALL  DLY1
0D55 F1                 POP   AF
0D56 C9                 RET
0D57                    ;
0D57 F5         LONG:   PUSH  AF
0D58 3E03               LD    A,03H
0D5A 3203E0             LD    (CSTPT),A
0D5D CD5907             CALL  DLY1
0D60 CD5907             CALL  DLY1
0D63 CD5907             CALL  DLY1
0D66 CD5907             CALL  DLY1
0D69 3E02               LD    A,02H
0D6B 3203E0             LD    (CSTPT),A
0D6E CD5907             CALL  DLY1
0D71 CD5907             CALL  DLY1
0D74 CD5907             CALL  DLY1
0D77 CD6007             CALL  DLY2
0D7A F1                 POP   AF
0D7B C9                 RET
0D7C                    ;
0D7C                    ;;
0D7C                    ;    @DSP PUCH
0D7C                    ;
0D7C FE08       ?DSPA:  CP    08H
0D7E 2810               JR    Z,DSP03
0D80 CB0E               RRC   (HL)
0D82 10FC               DJNZ  -2
0D84 CBC6               SET   0,(HL)
0D86 CB8E               RES   1,(HL)
0D88 47                 LD    B,A
0D89 CB06               RLC   (HL)
0D8B 10FC               DJNZ  -2
0D8D C37B0E     DSP04:  JP    CURSR
0D90                    ;
0D90 23         DSP03:  INC   HL
0D91 CBC6               SET   0,(HL)
0D93 CB8E               RES   1,(HL)
0D95 18F6               JR    DSP04
0D97                    ;
0D97 CBFE       DSP02:  SET   7,(HL)
0D99 23                 INC   HL
0D9A CB86               RES   0,(HL)
0D9C 18EF               JR    DSP04
0D9E                    ;
0D9E                    ;   PRESS PLAY MESSAGE
0D9E                    ;
0D9E 7F20       MSG#1:  DEFW  207FH
0DA0 504C4159   MSG#2:  DEFM  'PLAY'
0DA4 0D                 DEFB  0DH
0DA5                    ;
0DA5                    ;
```

```
0DA5
0DA6
0DA6
0DA6
0DA6
0DA6
0DA6 C9
0DA7
0DA7 C5
0DA8 0623
0DAA CDA209
0DAD 10FB
0DAF C1
0DB0 C9
0DB1
0DB1
0DB1 D7
0DB2 4130
0DB4 0D
0DB5
0DB5
0DB5
0DB5
0DB5
0DB5
0DB5
0DB5
0DB5
0DB5 F5
0DB6 C5
0DB7 D5
0DB8 E5
0DB9 47
0DBA CDB10F
0DBD 70
0DBE 2A7111
0DC1 7D
0DC2 FE27
0DC4 20C7
0DC6 CD2B0A
0DC9 38C2
0DCB 3A9111
0DCE B7
0DCF C29D03
0DD2 EB
0DD3 78
0DD4 FE07
0DD6 28BF
0DD8 18A2
0DDA
0DDA
0DDA
0DDC
0DDC
0DDC
```

```
        ORG   0DA6H             ;?BLNK   15
;
;
;  V-BLANK CHECK  ;
;
?BLNK:  RET
;
DLY12:  PUSH  BC
        LD    B,35              ; 12 M SEC DELAY
        CALL  DLY3
        DJNZ  -3
        POP   BC
        RET
;
;   BELL DATA
?BELD:  DEFB  D7H
        DEFM  'AO'
        DEFB  0DH
;
        ORG   0DB5H             ;?DSP    39
;
;
;
;  DISPLAY ON POINTER  ;
;
;    ACC = DISPLAY CODE
;  EXCEPT F0
?DSP:   PUSH  AF
        PUSH  BC
        PUSH  DE
        PUSH  HL
        LD    B,A
DSP01:  CALL  ?PONT
        LD    (HL),B
        LD    HL,(DSPXY)
        LD    A,L
        CP    +39
        JR    NZ,DSP04
        CALL  .MANG
        JR    C,DSP04
        LD    A,(SPAGE)
        OR    A
        JP    NZ,.DSP03
        EX    DE,HL
        LD    A,B
        CP    07H
        JR    Z,DSP02
        JR    ?DSPA
;
;
;
        ORG   0DDCH             ;?DPCT
;
;
```



```
0DDD C5
0DDE D5
0DDF E5
0DE0 47
0DE1 E6F0
0DE3 FEC0
0DE5 C2E50E
0DE8 A8
0DE9 07
0DEA 4F
0DEB 0600
0DED 21FF0D
0DF0 3A9111
0DF3 B7
0DF4 2803
0DF6 216801
0DF9 09
0DFA 5E
0DFB 23
0DFC 56
0DFD EB
0DFE E9
0DFF
0DFF 1F0E
0E01 5D0E
0E03 6E0E
0E05 7B0E
0E07 950E
0E09 FB03
0E0B B30E
0E0D F20E
0E0F 2D0F
0E11 E10E
0E13 EE0E
0E15 E50E
0E17 170A
0E19 730F
0E1B E805
0E1D 590F
0E1F
0E1F
0E1F
0E1F 217A11
0E22 0E05
0E24 3A7F11
0E27 81
0E28 327F11
0E2B 3A7B11
0E2E 81
0E2F 327B11
0E32 79
0E33 86
0E34 77
0E35 CDC709
0E38 2A7D11
```

```
        PUSH  BC
        PUSH  DE
        PUSH  HL
        LD    B,A
        AND   F0H
        CP    C0H
        JP    NZ,?RSTR
        XOR   B
        RLCA
        LD    C,A
        LD    B,+0
        LD    HL,CTBL
        LD    A,(SPAGE)
        OR    A
        JR    Z,+5
        LD    HL,.CTBL          ; PAGE MODE1
        ADD   HL,BC
        LD    E,(HL)
        INC   HL
        LD    D,(HL)
        EX    DE,HL
        JP    (HL)
;
CTBL:   DEFW  SCROL
        DEFW  CURSD
        DEFW  CURSU
        DEFW  CURSR
        DEFW  CURSL
        DEFW  HOME
        DEFW  CLRS
        DEFW  DEL
        DEFW  INST
        DEFW  ALPHA
        DEFW  KANA
        DEFW  ?RSTR
        DEFW  REV
        DEFW  CR
        DEFW  ROLUP
        DEFW  ROLD
;
;
;
SCROL:  LD    HL,PBIAS
        LD    C,5
        LD    A,(ROLEND)
        ADD   A,C
        LD    (ROLEND),A
        LD    A,(ROLTOP)
        ADD   A,C
        LD    (ROLTOP),A
SCROL1: LD    A,C
        ADD   A,(HL)
        LD    (HL),A
        CALL  PAGE
        LD    HL,(PAGETP)
        LD    DE,1000
```

```
0E44 77                     LD    (HL),A
0E45 23                     INC   HL
0E46 10FA                   DJNZ  SCROL2
0E48 3A7A11                 LD    A,(PBIAS)       ; SCROL
0E4B 6F                     LD    L,A
0E4C 26E2                   LD    H,E2H
0E4E 7E                     LD    A,(HL)
0E4F 217911                 LD    HL,MANGE        ; COLOUMN MANAG. END
0E52 B7                     OR    A
0E53 0607                   LD    B,7
0E55 CB1E                   RR    (HL)
0E57 2B                     DEC   HL
0E58 10FB                   DJNZ  -3
0E5A C3E50E                 JP    ?RSTR
0E5D                ;
0E5D                ;
0E5D 2A7111         CURSD:  LD    HL,(DSPXY)
0E60 7C                     LD    A,H
0E61 FE18                   CP    +24
0E63 282E                   JR    Z,CURS4
0E65 24                     INC   H
0E66 CD8302         CURS1:  CALL  MGP.I
0E69 227111         CURS3:  LD    (DSPXY),HL
0E6C 1877                   JR    ?RSTR
0E6E                ;
0E6E 2A7111         CURSU:  LD    HL,(DSPXY)
0E71 7C                     LD    A,H
0E72 B7                     OR    A
0E73 2835                   JR    Z,CURS5
0E75 25                     DEC   H
0E76 CD9D02         CURSU1: CALL  MGP.D
0E79 18EE                   JR    CURS3
0E7B                ;
0E7B 2A7111         CURSR:  LD    HL,(DSPXY)
0E7E 7D                     LD    A,L
0E7F FE27                   CP    +39
0E81 3003                   JR    NC,CURS2
0E83 2C                     INC   L
0E84 18E3                   JR    CURS3
0E86 2E00           CURS2:  LD    L,+0
0E88 24                     INC   H
0E89 7C                     LD    A,H
0E8A FE19                   CP    +25
0E8C 38D8                   JR    C,CURS1
0E8E 2618                   LD    H,+24
0E90 227111                 LD    (DSPXY),HL
0E93 1842           CURS4:  JR    CURS6
0E95                ;
0E95 2A7111         CURSL:  LD    HL,(DSPXY)
0E98 7D                     LD    A,L
0E99 B7                     OR    A
0E9A 2803                   JR    Z,+5
0E9C 2D                     DEC   L
0E9D 18CA                   JR    CURS3
0E9F 2E27                   LD    L,+39
0EA1 25                     DEC   H
0EA2 F2760E                 JP    P,CURSU1
0EA5 2600                   LD    H,0
0EA7 227111                 LD    (DSPXY),HL
0EAA 3A9111         CURS5:  LD    A,(SPAGE)
```



```
0EAD B7                     OR    A
0EAE 2035                   JR    NZ,?RSTR
0EB0 C3590F                 JP    ROLD
0EB3                ;
0EB3 217311         CLRS:   LD    HL,MANG
0EB6 061B                   LD    B,27
0EB8 CDD80F                 CALL  ?CLER
0EBB 2100D0                 LD    HL,D000H        ; SCRN TOP
0EBE E5                     PUSH  HL
0EBF CDE209                 CALL  #CLR08
0EC2 E1                     POP   HL
0EC3 3A9111                 LD    A,(SPAGE)
0EC6 B7                     OR    A
0EC7 2008                   JR    NZ,CLRS1
0EC9 227D11                 LD    (PAGETP),HL
0ECC 3E7D                   LD    A,7DH
0ECE 327F11                 LD    (ROLEND),A
0ED1 3A00E2         CLRS1:  LD    A,(E200H)
0ED4 C30904         HOM00:  JP    HOM0
0ED7                ;
0ED7                ;   CURS PART2 (CURS6)
0ED7                ;
0ED7 3A9111         CURS6:  LD    A,(SPAGE)
0EDA B7                     OR    A
0EDB C23D01                 JP    NZ,.SCROL
0EDE C39F0F                 JP    ROLU
0EE1                ;
0EE1                ;
0EE1                ;
0EE1                        ORG   0EE1H           ;ALPHA   13
0EE1                ;
0EE1 AF             ALPHA:  XOR   A
0EE2 327011         ALPH1:  LD    (KANAF),A
0EE5                ;
0EE5                ;
0EE5                ;  RESTORE  ;
0EE5                ;
0EE5                ?RSTR:  ENT
0EE5 E1                     POP   HL
0EE6 D1             ?RSTR1: POP   DE
0EE7 C1                     POP   BC
0EE8 F1                     POP   AF
0EE9 C9                     RET
0EEA                ;
0EEA                ;  MONITOR WORK AREA  ;
0EEA                ;
0EEA  P             SCRN:   EQU   D000H
0EEA  P             KANST:  EQU   E003H           ; KANA STATUS PORT
0EEA                ;
0EEA                ;
0EEA                ;
0EEE                        ORG   0EEEH           ;KANA   195
0EEE                ;
0EEE 3E01           KANA:   LD    A,+1
0EF0 18F0                   JR    ALPH1
0EF2                ;
0EF2                ;
0EF2 2A7111         DEL:    LD    HL,(DSPXY)
0EF5 7C                     LD    A,H             ; HOME ?
0EF6 B5                     OR    L
```

```
0EF7 28EC                JR    Z,?RSTR
0EF9 7D                  LD    A,L
0EFA B7                  OR    A
0EFB 200D                JR    NZ,DEL1          ; LEFT SIDE ?
0EFD CD2B0A              CALL  .MANG
0F00 3808                JR    C,DEL1
0F02 CDB10F              CALL  ?PONT
0F05 2B                  DEC   HL
0F06 3600                LD    (HL),+0
0F08 188B                JR    CURSL
0F0A CD2B0A        DEL1: CALL  .MANG
0F0D 0F                  RRCA
0F0E 3E28                LD    A,40
0F10 3001                JR    NC,+3
0F12 07                  RLCA                   ; ACC=80
0F13 95                  SUB   L
0F14 47                  LD    B,A
0F15 CDB10F              CALL  ?PONT
0F18 E5                  PUSH  HL
0F19 D1                  POP   DE
0F1A 1B                  DEC   DE
0F1B CBE2                SET   4,D
0F1D CB9C          DEL2: RES   3,H
0F1F CB9A                RES   3,D
0F21 7E                  LD    A,(HL)
0F22 12                  LD    (DE),A
0F23 23                  INC   HL
0F24 13                  INC   DE
0F25 10F6                DJNZ  DEL2
0F27 2B                  DEC   HL
0F28 3600                LD    (HL),+0
0F2A C3950E              JP    CURSL
0F2D               ;
0F2D CD2B0A        INST: CALL  .MANG
0F30 0F                  RRCA
0F31 2E27                LD    L,+39
0F33 7D                  LD    A,L
0F34 3001                JR    NC,+3
0F36 24                  INC   H
0F37 CDB40F              CALL  ?PNT1
0F3A E5                  PUSH  HL
0F3B 2A7111              LD    HL,(DSPXY)
0F3E 3002                JR    NC,+4
0F40 3E4F                LD    A,+79
0F42 95                  SUB   L
0F43 47                  LD    B,A
0F44 D1                  POP   DE
0F45 1A                  LD    A,(DE)
0F46 B7                  OR    A
0F47 209C                JR    NZ,?RSTR
0F49 CDB10F              CALL  ?PONT
0F4C 7E                  LD    A,(HL)
0F4D 3600                LD    (HL),0
0F4F 23            INST1: INC   HL
0F50 CB9C                RES   3,H
0F52 5E                  LD    E,(HL)
0F53 77                  LD    (HL),A
0F54 7B                  LD    A,E
0F55 10F8                DJNZ  INST1
0F57 188C                JR    ?RSTR
```



```
0F59               ;
0F59 217A11        ROLD: LD    HL,PBIAS
0F5C 3A7B11              LD    A,(ROLTOP)
0F5F BE                  CP    (HL)
0F60 2883                JR    Z,?RSTR
0F62 CD9D02              CALL  MGP.D
0F65 7E                  LD    A,(HL)
0F66 D605                SUB   5
0F68 77            ROL2: LD    (HL),A
0F69 6F                  LD    L,A
0F6A 26E2                LD    H,E2H
0F6C 7E                  LD    A,(HL)
0F6D CDC709              CALL  PAGE
0F70 C3E50E              JP    ?RSTR
0F73
0F73               ;
0F73 CD2B0A        CR:   CALL  .MANG
0F76 0F                  RRCA
0F77 D2860E              JP    NC,CURS2
0F7A 2E00                LD    L,0
0F7C 24                  INC   H
0F7D 7C                  LD    A,H
0F7E FE18                CP    +24
0F80 2817                JR    Z,CR3
0F82 3007                JR    NC,CR2
0F84 CD8302              CALL  MGP.I
0F87 24                  INC   H
0F88 C3660E              JP    CURS1
0F8B 25            CR2:  DEC   H
0F8C 227111              LD    (DSPXY),HL
0F8F 219F0F              LD    HL,ROLU
0F92 E5                  PUSH  HL
0F93 F5                  PUSH  AF
0F94 C5                  PUSH  BC
0F95 D5                  PUSH  DE
0F96 CD9F0F              CALL  ROLU
0F99 227111        CR3:  LD    (DSPXY),HL
0F9C CD8302              CALL  MGP.I
0F9F               ;
0F9F 217A11        ROLU: LD    HL,PBIAS
0FA2 3A7F11              LD    A,(ROLEND)
0FA5 BE                  CP    (HL)
0FA6 CA1F0E              JP    Z,SCROL
0FA9 CD8302        ROLU1: CALL  MGP.I
0FAC 7E                  LD    A,(HL)
0FAD C605                ADD   A,5
0FAF 18B7                JR    ROL2
0FB1               ;
0FB1                     ORG   0FB1H            ;?PONT    3
0FB1               ;
0FB1               ;
0FB1               ;  COMPUTE POINT ADR .  ;
0FB1               ;
0FB1               ;    HL = SCREEN CORDINATE
0FB1               ;    EXIT
0FB1               ;    HL = POINT ADR. ON SCREEN
0FB1               ;
0FB1 2A7111        ?PONT: LD    HL,(DSPXY)
0FB4               ;
0FB4                     ORG   0FB4H            ;?PNT1
```

```
**  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 55          09/04/81

0FB4                    ;
0FB4 F5                 ?PNT1:  PUSH  AF
0FB5 C5                         PUSH  BC
0FB6 D5                         PUSH  DE
0FB7 E5                         PUSH  HL
0FB8 C1                         POP   BC
0FB9 112800                     LD    DE,0028H          ; 40
0FBC 21D8CF                     LD    HL,SCRN-40
0FBF 3A9111                     LD    A,(SPAGE)
0FC2 B7                         OR    A
0FC3 2005                       JR    NZ,?PNT2
0FC5 2A7D11                     LD    HL,(PAGETP)
0FC8 ED52                       SBC   HL,DE
0FCA 19                 ?PNT2:  ADD   HL,DE
0FCB 05                         DEC   B
0FCC F2CA0F                     JP    P,-2
0FCF 0600                       LD    B,+0
0FD1 09                         ADD   HL,BC
0FD2 CB9C                       RES   3,H
0FD4 D1                         POP   DE
0FD5 C1                         POP   BC
0FD6 F1                         POP   AF
0FD7 C9                         RET
0FD8                    ;
0FD8                    ;
0FD8                            ORG   0FD8H             ;?CLER    40
0FD8                    ;
0FD8                    ;
0FD8                    ;  CLER  ;
0FD8                    ;  B=SIZE
0FD8                    ;  HL=LOW ADR.
0FD8                    ;
0FD8                    ?CLER:  ENT
0FD8 AF                         XOR   A
0FD9 1802                       JR    +4
0FDB                    ?CLRFF: ENT
0FDB 3EFF                       LD    A,FFH
0FDD                    ?DINT:  ENT
0FDD 77                         LD    (HL),A
0FDE 23                         INC   HL
0FDF 10FC                       DJNZ  -2
0FE1 C9                         RET
0FE2                    ;
0FE2                    ;
0FE2                    ;    GAP CHECK
0FE2                    ;
0FE2 C5                 GAPCK:  PUSH  BC
0FE3 D5                         PUSH  DE
0FE4 E5                         PUSH  HL
0FE5 0101E0                     LD    BC,KEYPB
0FE8 1102E0                     LD    DE,CSTR
0FEB 2664               GAPCK1: LD    H,100
0FED CD0106             GAPCK2: CALL  EDGE
0FF0 380B                       JR    C,GAPCK3
0FF2 CDA209                     CALL  DLY3              ; CALL DLY2*3
0FF5 1A                         LD    A,(DE)
0FF6 E620                       AND   20H
0FF8 20F1                       JR    NZ,GAPCK1
0FFA 25                         DEC   H
0FFB 20F0                       JR    NZ,GAPCK2
```

```
**  Z80 ASSEMBLER SB-7201  (MZ-80A.MONITOR)  PAGE 56          09/04/81

0FFD C39F06             GAPCK3: JP    RET3
1000                            SKP   H
```

```
1000                    ;
1000                    ;
1000                    ;   MONITOR WORK AREA  ;
1000                    ;     (MZ-80A)         ;
1000                    ;
1000                    ;
1000                    ;
10F0                            ORG   10F0H
10F0                    SP:     ENT
10F0                    IBUFE:  ENT                     ; TAPE BUFFER(128B)
10F0                    ATRB:   ENT                     ; ATRIBUTE
                                DEFS  +1
1101                    NAME:   ENT                     ; FILE NAME
                                DEFS  +17
1102                    SIZE:   ENT                     ; BYTE SIZE
                                DEFS  +2
1104                    DTADR:  ENT                     ; DATA ADR
                                DEFS  +2
1106                    EXADR:  ENT                     ; EXECUTION ADR
                                DEFS  +2
1108                    COMNT:  ENT                     ; COMMENT
                                DEFS  +92
1164                    SWPW:   ENT                     ; SWEEP WORK
                                DEFS  +10
116E                    KDATW:  ENT                     ; KEY WORK
                                DEFS  +2
1170                    KANAF:  ENT                     ; KANA FLAG
                                DEFS  +1
1171                    DSPXY:  ENT                     ; DISPLAY CO-ORDINATES
                                DEFS  +2
1173                    MANG:   ENT                     ; COLOUMN MANAGEMENT
                                DEFS  +6
1179                    MANGE:  ENT
                                DEFS  +1                ; COLOUMN MANAG. END
117A                    PBIAS:  ENT
                                DEFS  +1                ; PAGE BIAS
117B                    ROLTOP: ENT
                                DEFS  +1                ; ROLL TOP BIAS
117C                    MGPNT:  ENT
                                DEFS  +1                ; COLOUMN MANAG. POINTER
117D                    PAGETP: ENT
                                DEFS  +2                ; PAGE TOP
117F                    ROLEND: ENT
                                DEFS  +1                ; ROLL END
1180                            DEFS  +14               ; BIAS
118E                    FLASH:  ENT                     ; FLASHING DATA
                                DEFS  +1
118F                    SFTLK:  ENT                     ; SHIFT LOCK
                                DEFS  +1
1190                    REVFLG: ENT                     ; REVERSE FLAG
                                DEFS  +1
1191                    SPAGE:  ENT                     ; PAGE CHENGE
                                DEFS  +1
1192                    FLSDT:  ENT                     ; CURSOL DATA
                                DEFS  +1
1193                    STRGF:  ENT                     ; STRING FLAG
                                DEFS  +1
1194                    DPRNT:  ENT                     ; TAB COUNTER
                                DEFS  +1
1195                    TMCNT:  ENT                     ; TAPE MARK COUNTER
```



```
1195                            DEFS  +2
1197                    SUMDT:  ENT                     ; CHECK SUM DATA
1197                            DEFS  +2
1199                    CSMDT:  ENT                     ; FOR COMPARE SUM DATA
1199                            DEFS  +2
119B                    AMPM:   ENT                     ; AMPM DATA
119B                            DEFS  +1
119C                    TIMFG:  ENT                     ; TIME FLAG
119C                            DEFS  +1
119D                    SWRK:   ENT                     ; KEY SOUND FLAG
119D                            DEFS  +1
119E                    TEMPW:  ENT                     ; TEMPO WORK
119E                            DEFS  +1
119F                    ONTYO:  ENT                     ; ONTYO WORK
119F                            DEFS  +1
11A0                    OCTV:   ENT                     ; OCTAVE WORK
11A0                            DEFS  +1
11A1                    RATIO:  ENT                     ; ONPÚ RATIO
11A1                            DEFS  +2
11A3                    BUFER:  ENT                     ; GET LINE BUFFER
11A3                            DEFS  +81
11F4                            END
```

#BRK	0BC5	#CLR08	09E2	#CLR8	09E3	#GRP	06E7	.CR	0128
.CR1	013A	.CTBL	0168	.DSP03	039D	.MANG	0A2B	.MANG1	0A4C
.MANG2	03A6	.SCROL	013D	00MSG	09FC	2HE1	0434	2HEX	041F
??KEY	09B3	?ADCN	0BB9	?BEL	02E5	?BELD	0DB1	?BLNK	0DA6
?BRK	0D11	?BRK1	0D37	?BRK2	0D27	?BRK3	0D2B	?CLER	0FD8
?CLRFF	0FDB	?DACN	0BCE	?DINT	0FDD	?DPCT	0DDC	?DSP	0DB5
?DSPA	0D7C	?ER	00CF	?FLAS	09FF	?GET	08B3	?GETL	07A8
?KEY	08CA	?KY1	08D6	?KY10	08EF	?KY11	08EA	?KY2	08F2
?KY3	0922	?KY4	092E	?KY5	091E	?KY6	092A	?KY7	093A
?KY8	08CD	?KY9	093F	?LOAD	05F5	?LTNL	0980	?MLDY	0188
?MODE	074D	?MSG	0893	?MSGX	08A1	?NL	097B	?PNT1	0FB4
?PNT2	0FCA	?PONT	0FB1	?PRNT	0995	?PRT	0946	?PRTS	0993
?PRTT	0984	?RDD	04EF	?RDI	04CF	?RSTR	0EE5	?RSTR1	0EE6
?SAVE	0263	?SWEP	0A50	?TEMP	02EC	?TMR1	0361	?TMR2	036B
?TMRD	0344	?TMS1	0323	?TMS2	0336	?TMST	02FA	?VRFY	0575
?WRD	0470	?WRI	0436	ALPH1	0EE2	ALPHA	0EE1	AMPM	119B
ASC	03DA	ATBL	0AB5	ATRB	10F0	AUTCK	09F1	AUTO2	0807
AUTO3	07C4	AUTO5	0824	AUTOL	0810	BELL	003E	BRKEY	001E
BUFER	11A3	CHGP1	084B	CHGPA	083E	CHGPK	0841	CKS1	0720
CKS2	072F	CKS3	0733	CKSUM	071A	CLEAR	09E6	CLEAR1	09E8
CLRS	0EB3	CLRS1	0ED1	COMNT	1108	CONT0	E004	CONT1	E005
CONT2	E006	CONTF	E007	CR	0F73	CR2	0F8B	CR3	0F99
CSMDT	1199	CSTPT	E003	CSTR	E002	CTBL	0DFF	CURS1	0E66
CURS2	0E86	CURS3	0E69	CURS4	0E93	CURS5	0EAA	CURS6	0ED7
CURSD	0E5D	CURSL	0E95	CURSR	0E7B	CURSU	0E6E	CURSU1	0E76
DACN1	0BE3	DACN2	0BDF	DACN3	0BE0	DEL	0EF2	DEL1	0F0A
DEL2	0F1D	DLY1	0759	DLY12	0DA7	DLY2	0760	DLY3	09A2
DMT	0857	DPRNT	1194	DSP01	0DBA	DSP02	0D97	DSP03	0D90
DSP04	0D8D	DSPXY	1171	DTADR	1104	EDG1	0607	EDG2	0615
EDGE	0601	EXADR	1106	FD	00C7	FD1	00CE	FD2	00CA
FLAS1	0A12	FLAS2	0A0B	FLAS3	0A0F	FLASH	118E	FLSDT	1192
GAP	077A	GAP1	078E	GAP2	0796	GAP3	079C	GAPCK	0FE2
GAPCK1	0FEB	GAPCK2	0FED	GAPCK3	0FFD	GETKY	001B	GETL	0003
GETL0	07AC	GETL1	07B6	GETL11	081A	GETL2	0803	GETL3	085B
GETL5	0839	GETL6	0865	GETLA	0886	GETLB	0863	GETLC	084E
GETLD	05E1	GETLL	03D5	GETLR	0880	GOTO	00BB	HEX	03F9
HEX1	03F2	HEX2	03F5	HEXJ	03E5	HL1	041D	HLHEX	0410
HOM0	0409	HOM00	0ED4	HOM1	0406	HOME	03FB	IBUFE	10F0
INST	0F2D	INST1	0F4F	KANA	0EEE	KANAF	1170	KANST	E003
KDATW	116E	KEYPA	E000	KEYPB	E001	KEYPC	E002	KEYPF	E003
KSL0	0270	KSL2	09B9	KTBL	0BEA	KTBLC	0CDA	KTBL6	0C6A
KTBLGS	0CA2	KTBLS	0C32	LETNL	0006	LOA0	00DE	LOAD	00D9
LOCK	088B	LONG	0DB7	M#TBL	0241	MANG	1173	MANGE	1179
MELDY	0030	MGP.D	029D	MGP.1	0283	MGP0	028F	MGPNT	117C
MLD1	0192	MLD2	01C6	MLD3	01CE	MLD4	01D2	MLD5	01D5
MLDS1	02C4	MLDSP	02BE	MLDST	02AB	MONIT	0000	MOT1	06A8
MOT2	06AF	MOT4	06B9	MOT5	06D8	MOT7	06B7	MOT8	06D0
MOT9	06D7	MOTOR	06A3	MSG	0015	MSG#1	0D9E	MSG#2	0DA0
MSG#3	06F4	MSG#7	0467	MSG1	0896	MSG?2	00F7	MSG?3	0100
MSGE1	0118	MSGX	0018	MSGX1	08A4	MSGX2	08A7	MST1	0705
MST2	070C	MST3	0717	MSTA	0044	MSTOP	0700	MSTP	0047
MTBL	0229	NAME	10F1	NL	0009	NOADD	03E2	OCTV	11A0
ONP1	01E0	ONP2	01ED	ONP3	09AB	ONPU	01DD	ONTY0	119F
OPTBL	0259	PAGE	09C7	PAGETP	117D	PBIAS	117A	PRNT	0012
PRNT2	0967	PRNT3	096C	PRNT4	096F	PRNT5	0955	PRNT8	000C
PRNTT	000F	PRTHL	038A	PRTHX	03C3	RATIO	11A1	RBY1	0630
RBY2	0649	RBY3	0654	RBYTE	0624	RD1	04DD	RDDAT	002A
RDINF	0027	RET1	04CB	RET2	0552	RET3	069F	REV	0A17
REV1	0A25	REVFLG	1190	ROL2	0F68	ROLD	0F59	ROLEND	117F
ROLTOP	117B	ROLU	0F9F	ROLU1	0FA9	ROLUP	05E8	RTAPE	0505

RTP1	050A	RTP2	0510	RTP3	052A	RTP4	0552	RTP5	0563
RTP6	0570	RTP7	056C	RTP8	0551	RTP9	0572	RYTHM	02CB
SCRN	D000	SCROL	0E1F	SCROL1	0E32	SCROL2	0E42	SFTLK	118F
SG	00C1	SHORT	0D3E	SIZE	1102	SP	10F0	SPAGE	1191
SS	0089	ST1	0095	ST2	00A3	START	004A	STRGF	1193
SUMDT	1197	SUNDG	E008	SWEP0	0A76	SWEP01	0A74	SWEP11	0A8E
SWEP2	0AAC	SWEP3	0A92	SWEP4	073E	SWEP6	0A62	SWEP7	0AA5
SWEP8	0AA0	SWEP9	0A8A	SWPW	1164	SWRK	119D	TEMP	E008
TEMPW	119E	TIMFG	119C	TIMIN	0379	TIMRD	003B	TIMST	0033
TM1	0673	TM2	0676	TM3	0689	TM4	069F	TMARK	0658
TMCNT	1195	TVF1	05A0	TVF2	05A6	TVF3	05BA	TVRFY	059B
VERFY	002D	VGOFF	0756	WBY1	076D	WBYTE	0767	WRDAT	0024
WRI1	0444	WRI2	045E	WRI3	0464	WRINF	0021	WTAP1	048F
WTAP2	049E	WTAP3	04CB	WTAPE	0485	XTEMP	0041		

Capitolo 3

Configurazione dell'Hardware dell'MZ-80A

3.1 Diagramma del sistema MZ-80A

La figura 3.1 mostra la configurazione standard del sistema di personal computer MZ-80A.

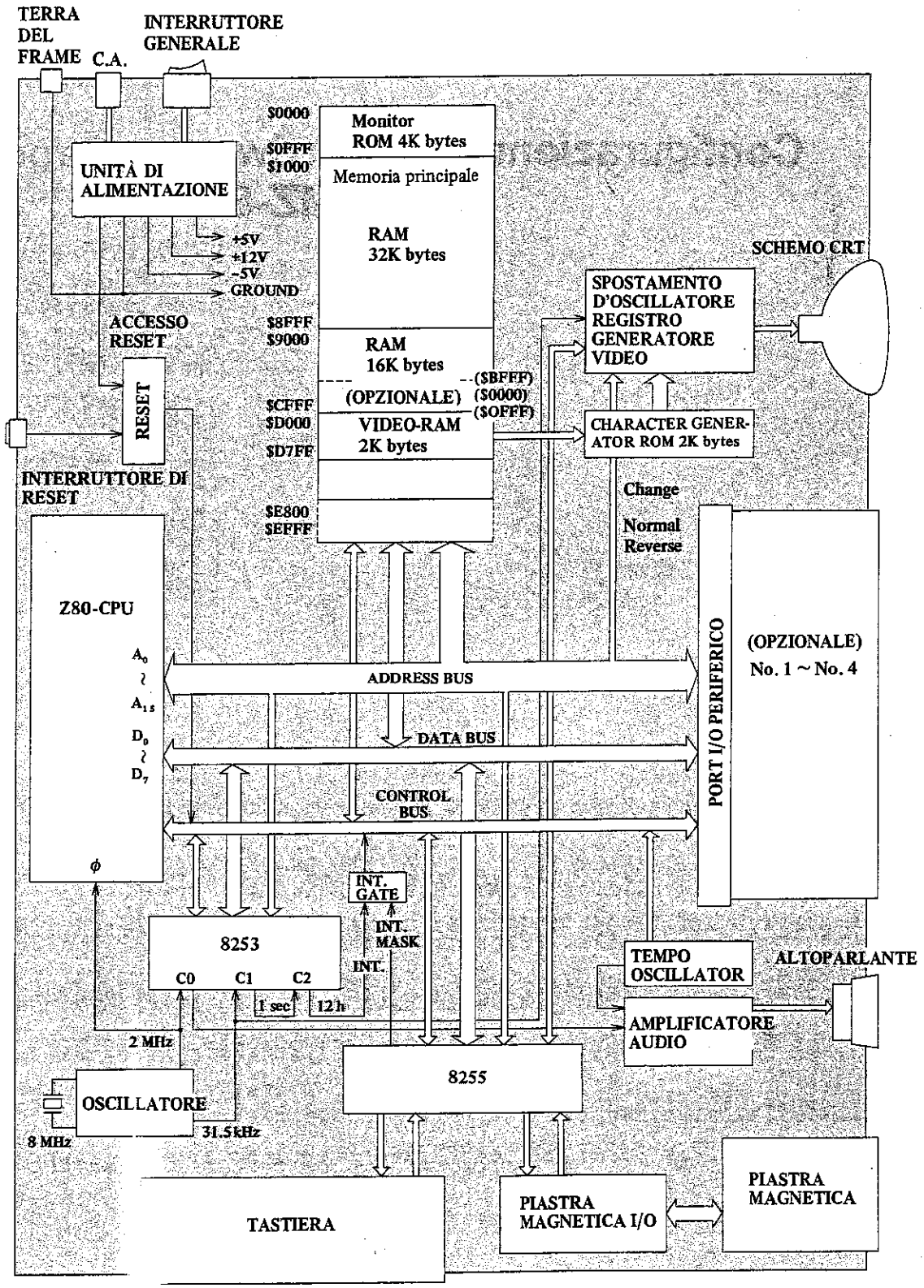


Figura 3.1 Diagramma del sistema MZ-80A.

Come mostrato nella figura lo Z80 (Sharp LH0080) viene usato come CPU, ed è operato con un clock di 2 MHz. Il CPU viene azzerato qiando l'apparecchio viene spento oppure quando si preme l'interruttore di "reset". La configurazione di memoria corrisponde ai bus d'indirizzo $0000-$FFFF come segue.

$000-$0FFF viene usato per il programma monitor (ROM); l'ampio spazio di 48 K byte da $1000-$CFFF viene usato come memoria principale (la memoria da $9000-$CFFF è opzionale); gli indirizzi da $D000 in su sono usati per il RAM video, controllo floppy e memoria I/O.

La tastiera e la piastra magnetica sono controllagili per mezzo di periferiche programmabili interfacciate da 8255. In oltre un'onda radio rettangolare generata dal port output del counter 1 del timer programmabile 8253 viene messo in input nel generatore sonoro, che viene transformato in suono dal trasdutore acustico. I due counter diversi da questo IC servono come clock interno dell'MZ-80A.

La tavola 3.1 mostra la configurazione della memoria dell'MZ-80A in mappa I/O.

Tavola 3.1 Assegnamento della memoria in mappa I/O

Indirizzo	Lettura memoria	Scrittura memoria	Dispositivo
$E000		D_7 : Azzera il cursore del timer $D_3 \sim D_0$: Strobo di tasto	8255
$E001	$D_7 \sim D_0$: Key data		
$E002	D_7 : $\overline{\text{V-Blank}}$ D_6 : Stato del cursore timer D_5 : Lettura dati (cassetta) D_4 : READ/WRITE (cassetta)	D_3 : Motore ON/OFF (cassetta) D_2 : Mascheratura dell'interrupt timer D_1 : Scrittura dati (cassetta) D_0 : $\overline{\text{V-Gate}}$	
$E003		Controllo modo	
$E004		Messa a punto counter # 0	8253
$E005	Lettura del counter # 1	Messa a punto counter # 1	
$E006	Lettura del counter # 2	Messa a punto counter # 2	
$E007		Controllo modo	
$E008	D_7 : Stato del tempo timer D_0 : H-Blank	D_0 : ON/OFF del suono	
$E00C	Memory swap		
$E010	Resets memory swap		
$E014	Nomale (CRT display)		
$E015	Inverso (CRT display)		
$E200 ~ $E2FF	Avvolgimento/svolgimento		

3.1.1 Configurazione della memoria

La mappa della memoria per l'MZ-80A è mostrata nella figura 3.2. La parte ombreggiata della figura indica la zona d'utente ed i 32 K byte della memoria principale RAM che sono nel package standard. I rimanenti 16 K byte della memoria principale RAM sono opzionali e sono installabili nella presa RAM fornita nella tavola del CPU.

I 4 K byte della zona di memoria principale che sono indicate dalla parte scura possono essere usati per un ampliamento di indirizzo usato dal MONITOR ROM. La parte di sinistra della figura mostra la mappa della memoria in condizioni normali, mentre la parte di destra mostra la memoria quando è stata ampliata la MONITOR ROM. Come mostrato dalla tavola 3.1, gli ampliamenti di memoria sono eseguiti sotto il controllo della memoria di mappa I/O eseguendo le istruzioni di lettura di memoria come segue.

Per mettere la memoria nella situazione illustrata a destra . . . . . LA D, (E00CH)

Per mettere la memoria nella situazione illustrata a sinistra . . . . . LD A, (E010H)

La configurazione della memoria mostrata a destra è utile quando programmi di sistema usati iniziano dall'indirizzo $0000 e quando i programmi di sistema utilizzati fanno un uso reale del processo di interrupt.

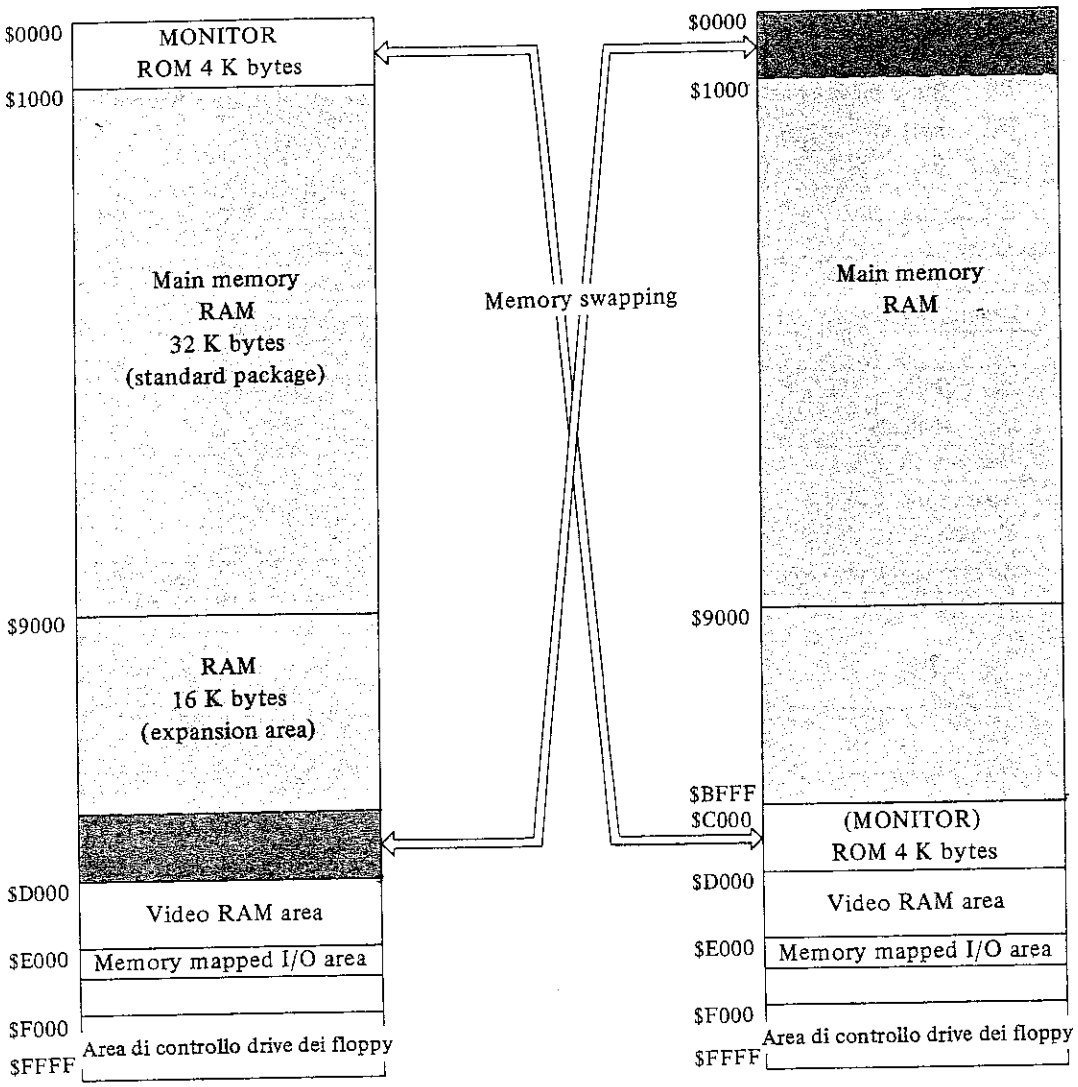


Figura 3.2 Mappe di memoria per lo stato normale e per lo stato espanso.

Quando la memoria è stata espansa 4 K ROM occupano lo spazio di indirizzo $C000-$CFFF; però il programma monitor non è effettivo a queste condizioni. Se necessario è possibile rimuovere il ROM monitor dall'alloggiamento del CPU e sostituirlo con un altro ROM d'utente programmato per permettere l'operazione nello spazio $C000-$CFFF. In tal caso usare la ROM 2732 che è simile alla ROM monitor. In oltre se è necessario alterare parte del programma per l'uso, può essere modificato per trasferimento di blocco da $C000-$CFFF a $0000-$OFFF ed effettuare il cambiamento in RAM.

La zona di 2 K byte $D000-$D7FF è asegnata al RAM video.

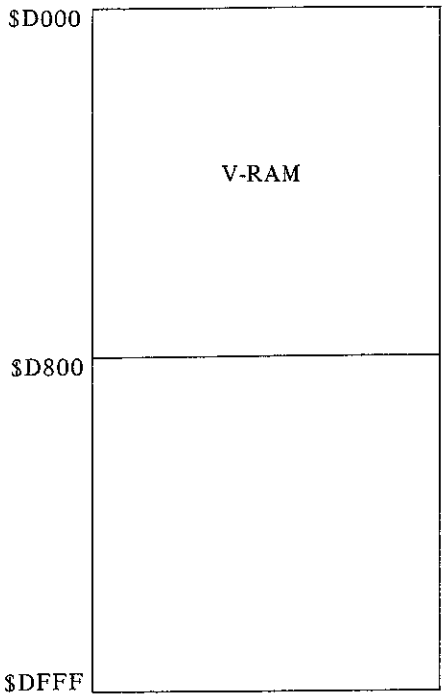
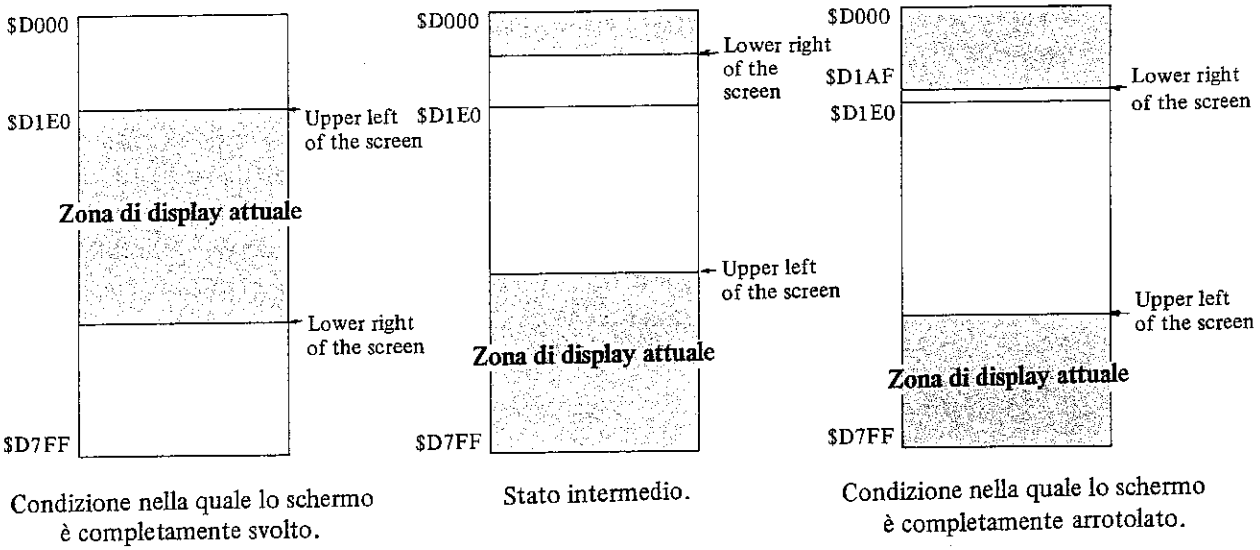


Figura 3.3 Zona RAM video dell'MZ-80A.

Sullo schermo CRT dell'unità principale MZ-80A possono essere mostrati fino a 1000 caratteri (40 caratteri x 25 linee), nella memoria RAM video c'è il doppio di questa zona. Ciò rende possible svolgere e riavvolgere lo schermo a piacimento.

Fino all'azzeramento del sistema, i dati sono scritti in RAM video a cominciare dall'indirizzo $D000 e quando più di 50 linee di dati sono scritte-cioè quando i dati sono stati scritti in una linea scroll, la zona $D050-$D7FF diventa zona RAM video, seguita dalla zona $D000-$D020. Per cui la RAM video è usata in una configurazione ancorata. La figura 3.4 mostra la relazione tra la RAM video ed il display per i 2 K byte della zona RAM video quando il suo primo K byte, il K byte medio oppure il K byte finale viene mostrato sullo schermo CRT. Però in questo esempio la zona RAM video capace di mostrare i dati sul CRT sono i 2000 byte che iniziano da $D1E0.



Condizione nella quale lo schermo è completamente svolto.

Stato intermedio.

Condizione nella quale lo schermo è completamente arrotolato.

Figura 3.4 Relazione tra RAM video e la zona attualemente mostrata.

La memoria I/O indirizzo $E200-$E2FF è usata per arrotolare il display su e giù. Quando le istruzioni di lettura della memoria vengono eseguite sull'indirizzo $E200 (come LD A, (E200H)), il CRT viene completamente svolto. Quando un'istruzione viene eseguita sull'indirizzo . . . il CRT viene completamente avvolto. I Byte bassi di questi indirizzi 00H-FFH possono essere liberamente usati per avvolgere oppure svolgere lo schermo in unità di 8 caratteri.

3.1.2 Sistema scansione tasti

La relazione tra i segnali strobo ed i dati bit durante la scansione di tastiera è mostrata nella figura 3.6.

I segnali strobo sono inviati a quattro terminali (PA_3, PA_2, PA_1, PA_0) dell'8255, vanno nel BCD al decimal decoder/driver 74145, poi inviati a dieci segnali input di tastiera. I tasti sono discriminati da segnali strobo e emettono dati. Per esempio quando lo strobo è "2H" ed il dato di tasto è "FBH", indica che è stato premuto il tasto [S].

Terminali input di tastiera strobo in numero 10

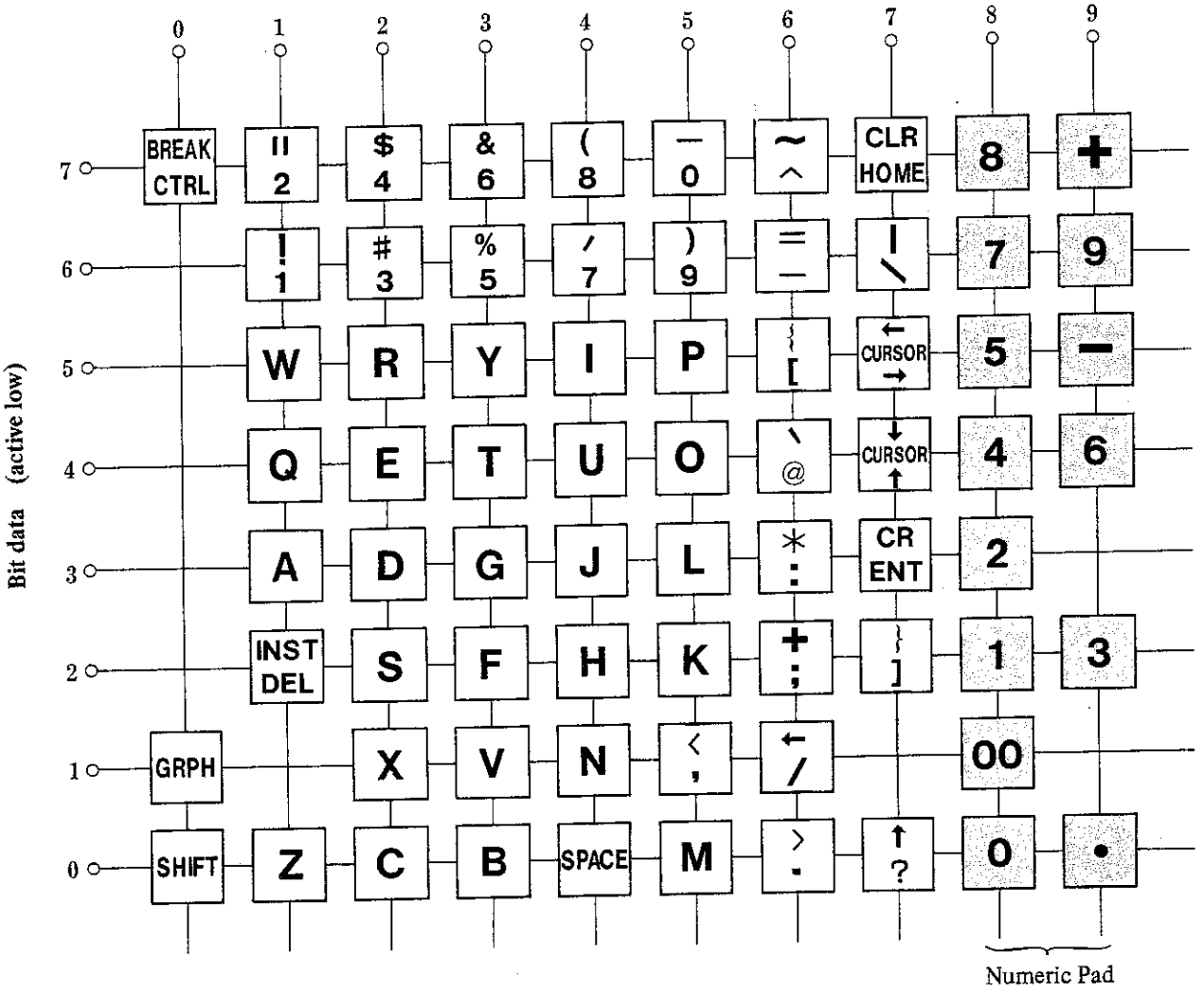


Figura 3.6 Segnale strobo di scansione tasti e dati bit.

3.2 Diagrammi circuitali dell'MZ-80A

Questa sezione comprende tutti i diagrammi ciruitali dell'MZ-80A per riferimento. Questi diagrammi sono sistemati nel modo seguente:

(1) Plancia CPU, blocco 1: sistema segnale CPU
(2) Plancia CPU, blocco 2
(3) Plancia CPU, blocco 3: sistema segnale RAM
(4) Plancia CPU, blocco 4: sistemi segnali 8255 e 8253
(5) Plancia CPU, blocco 5

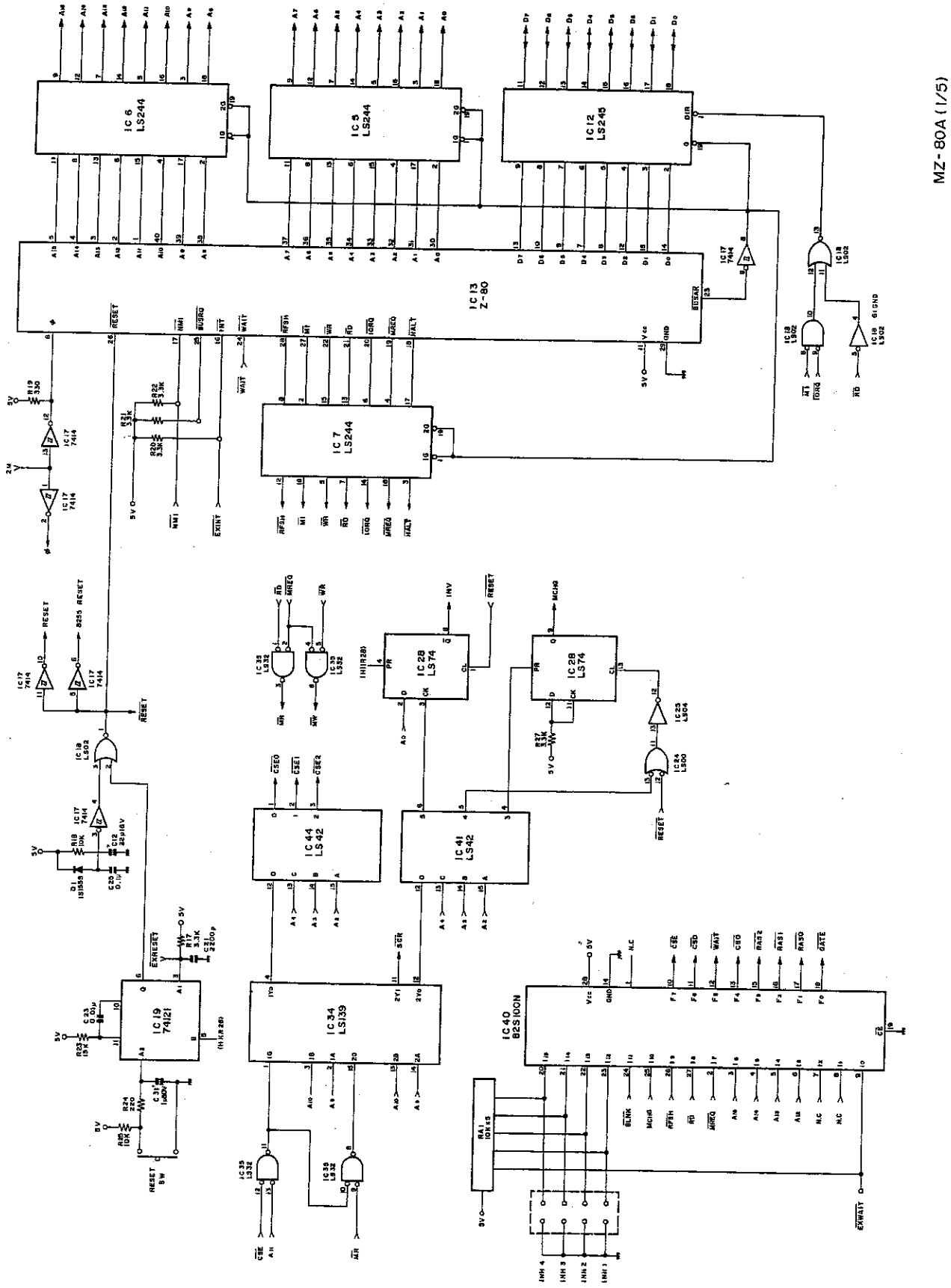


Figura 3.7 Plancia CPU, blocco 1: sistema segnale CPU.

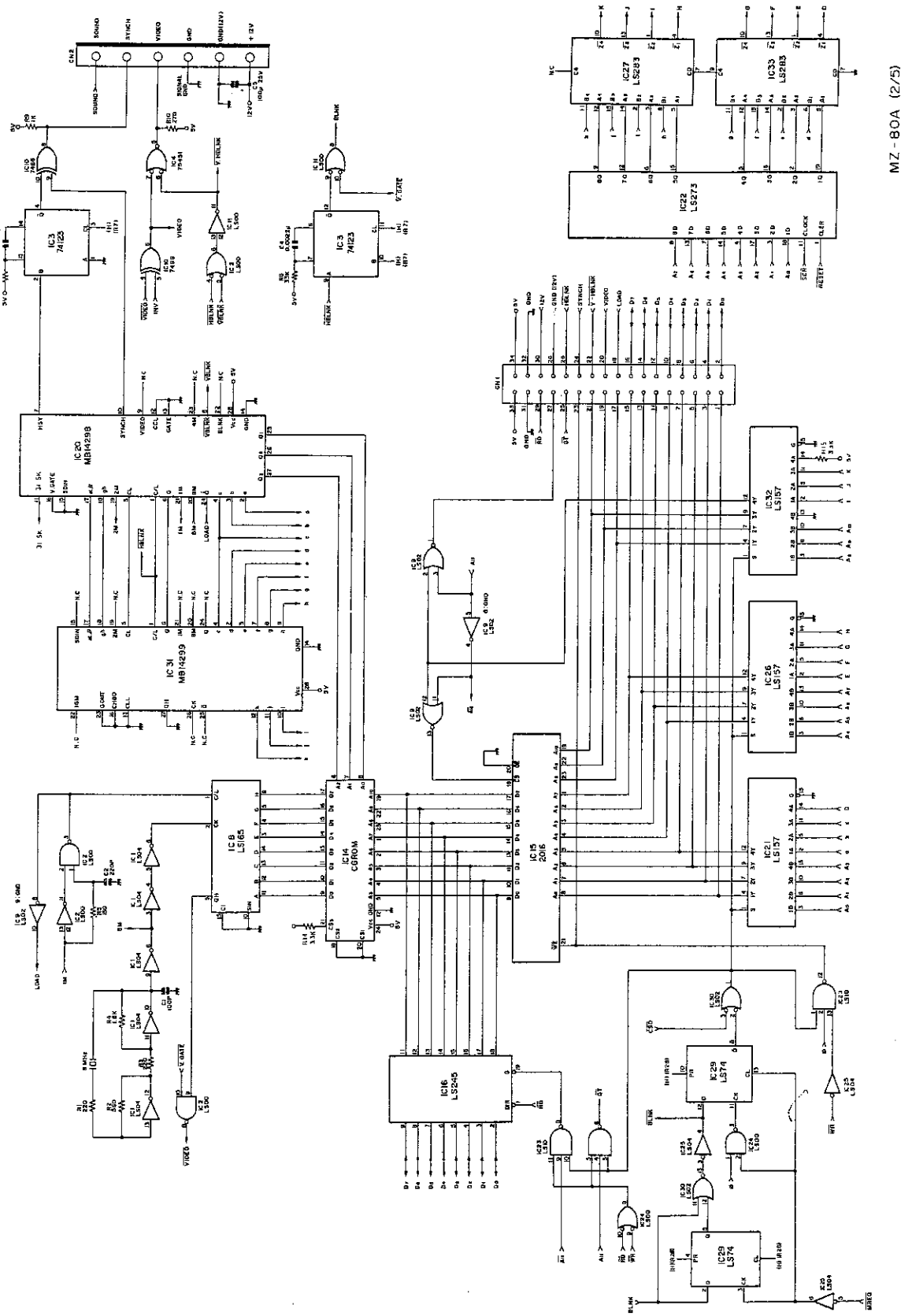


Figura 3.8 Plancia CPU, blocco 2.

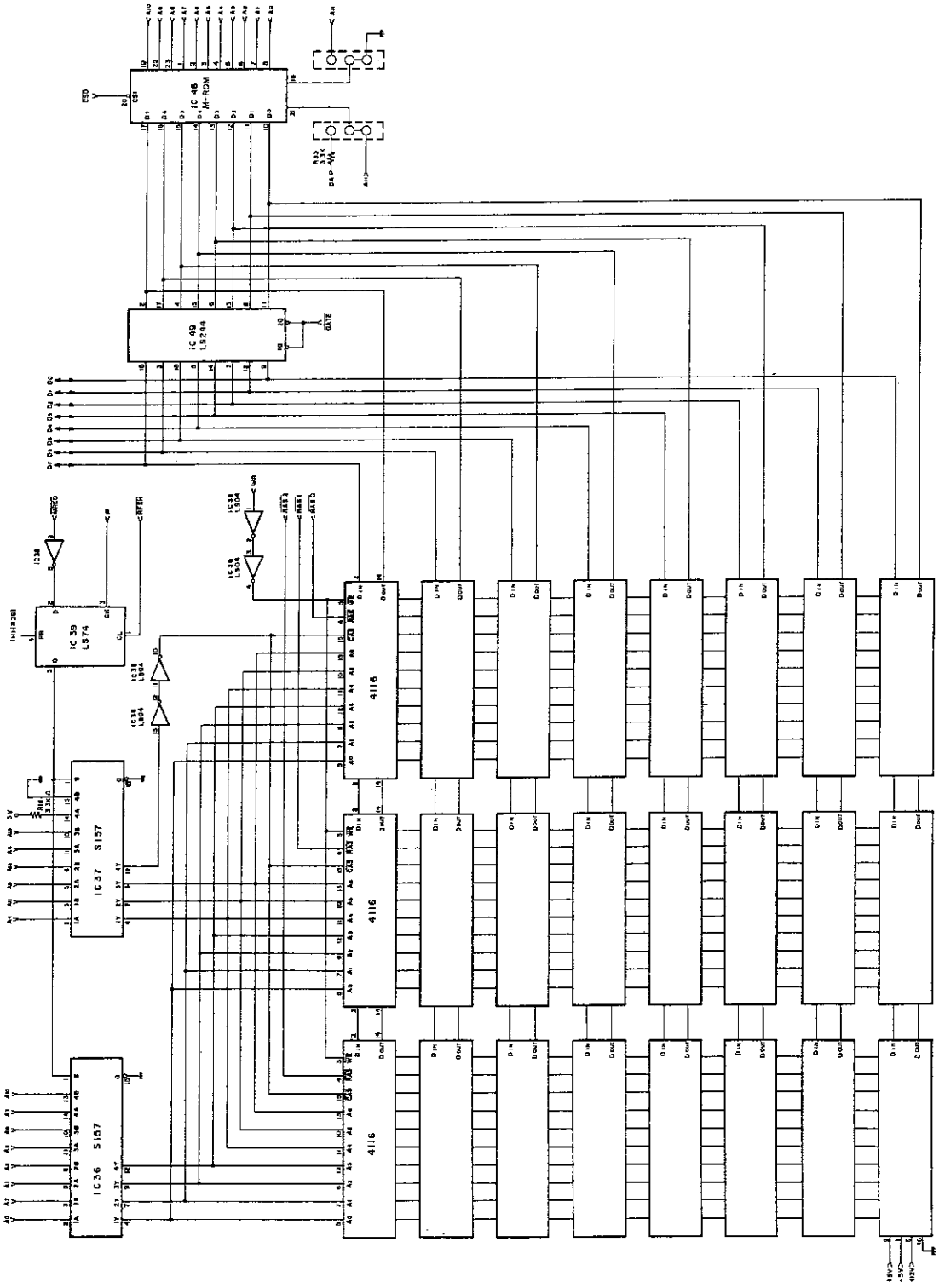


Figura 3.9 Plancia CPU, blocco 3: sistema segnale RAM.

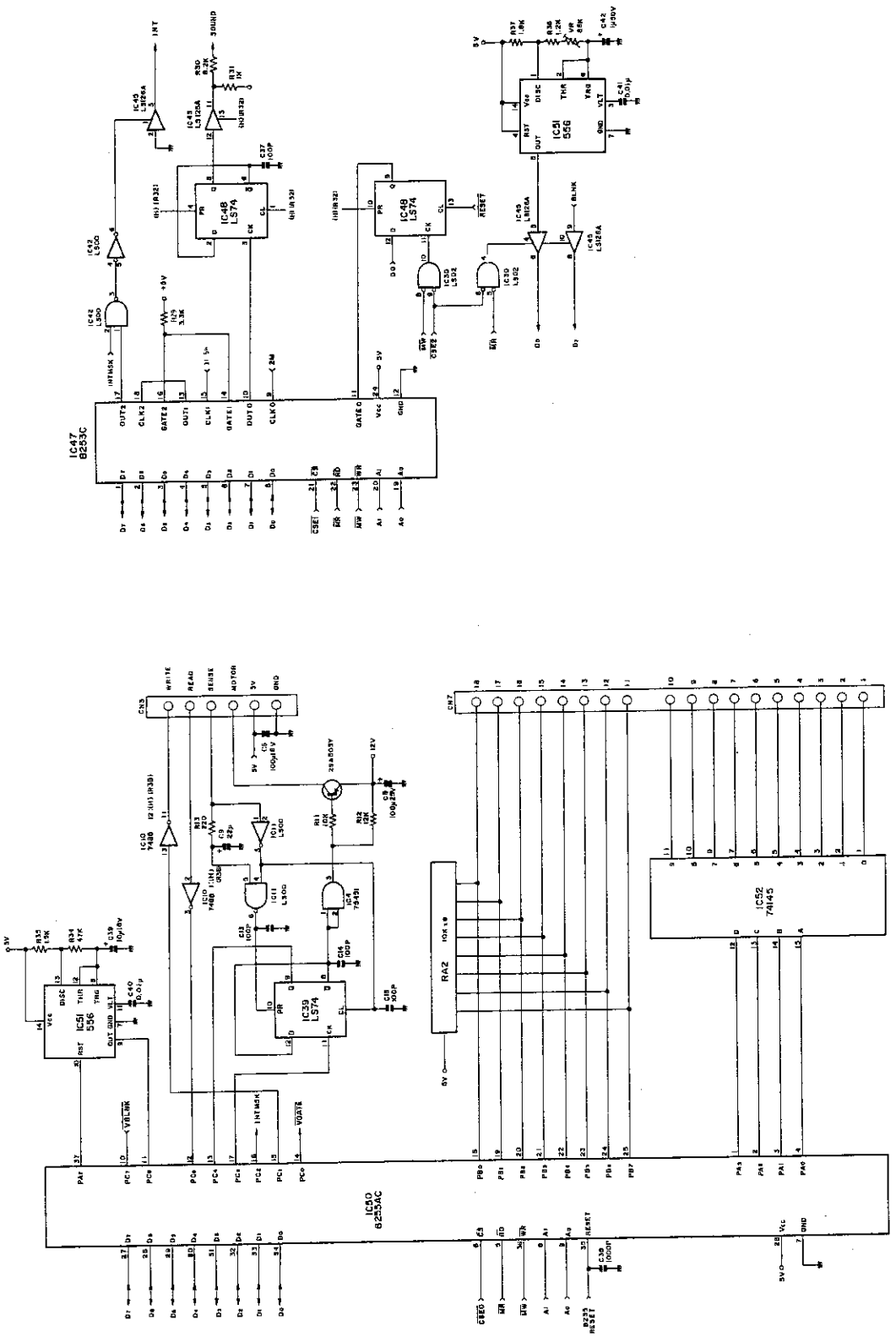
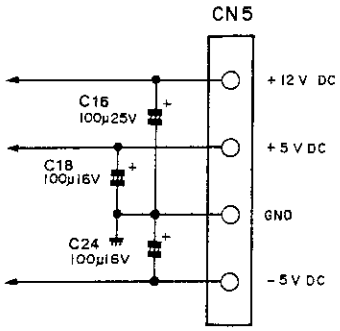


Figura 3.10 Plancia CPU, blocco 4: sistemi segnali 8255 e 8253.



CN 4

	A	B
1	D1	D0
2	D3	D2
3	GND	GND
4	D5	D4
5	D7	D6
6	GND	A0
7	RESET	A1
8	GND	A2
9	$\overline{\text{HALT}}$	A3
10	$\overline{\text{M1}}$	A4
11	GND	A5
12	$\overline{\text{WR}}$	A6
13	$\overline{\text{RD}}$	A7
14	GND	A8
15	$\overline{\text{IOREQ}}$	A9
16	$\overline{\text{MREQ}}$	A10
17	GND	A11
18	$\overline{\text{EXINT}}$	A12
19	GND	A13
20	$\overline{\text{NMI}}$	A14
21	$\overline{\text{EX WAIT}}$	A15
22	$\overline{\text{EX RESET}}$	Φ

A : PARTS SIDE

MZ-80A(5/5)

Figura 3.11 Plancia CPU, blocco 5.

3.3 Equipagiamento d'ampliamento

Varie periferiche sono disponibili per l'ampliamento del sistema del personale computer MZ-80A. La figura 3.12 mostra la tipica configurazione di un sistema ampliamento. Con il drive del floppy disk, si possono immagazzinare ed avere l'accesso a numerosi dati e file di programma ad alta velocità. Con il printer è possibile ottenere la copia su carta delle liste. Ciò migliora l'efficenza del processing, avendo come risultato l'ampliamento delle applicazioni.

L'MZ-80A Dual Floppy Disk Drive impiega un mini-floppy diskette (286 K byte/diskette) a doppia densità con un diametro di 5,25 pollici, le cui due facce sono usate per l'incisione. Permette l'utilizzazione dell'interpreter DISK BASIC, che è particolarmente adatto per molte utilizzazioni commerciali pratiche dell'interpreter DISK BASIC a doppia precisione, che esegue operazioni BCD a 16 cifre. Il sistema espanso mostra delle caratteristiche che sono proprie di computer molto più grandi con l'ausilio di una varietà di floppy disk per il funzionamento del sistema di software.

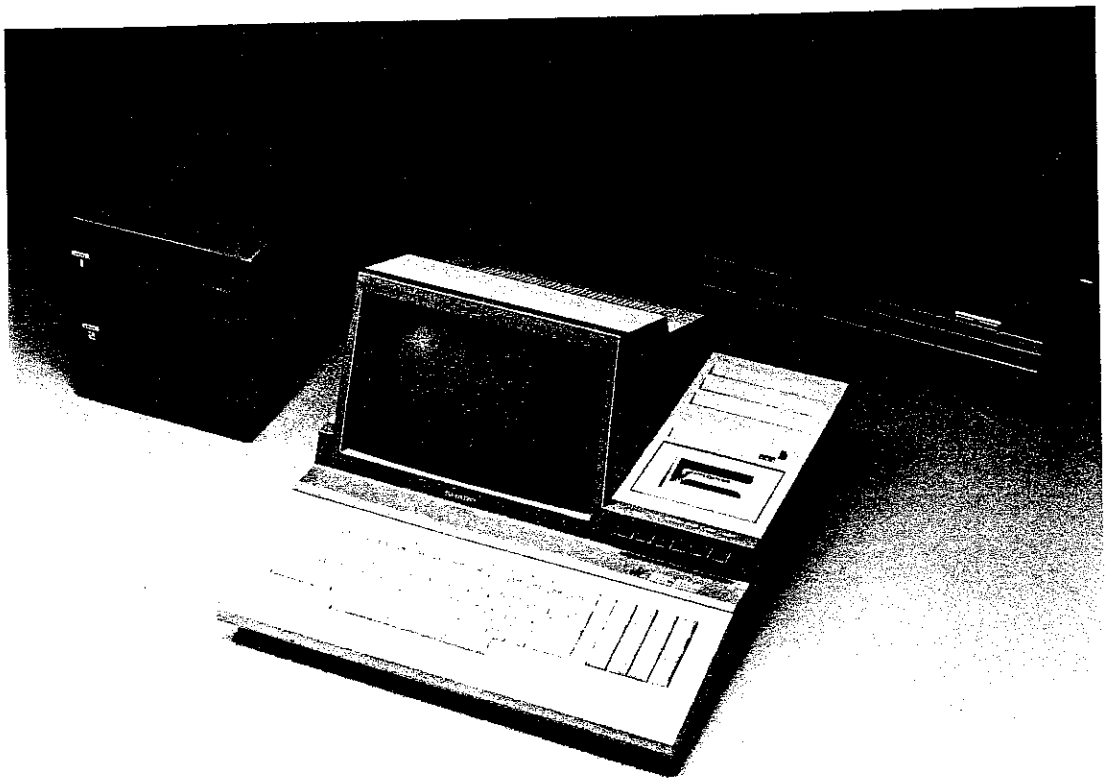


Figura 3.12 Tipico sistema espanso.

La figura 3.13 mostra le periferiche che possono essere collegate all'MZ-80A. Gli apparecchi che sono racchiusi, in figura, nella linea spessa continua sono collegati con il card interfaccia del porto d'ampliamento I/O oppure collegati al mobile principale con gli appositi adattatori.

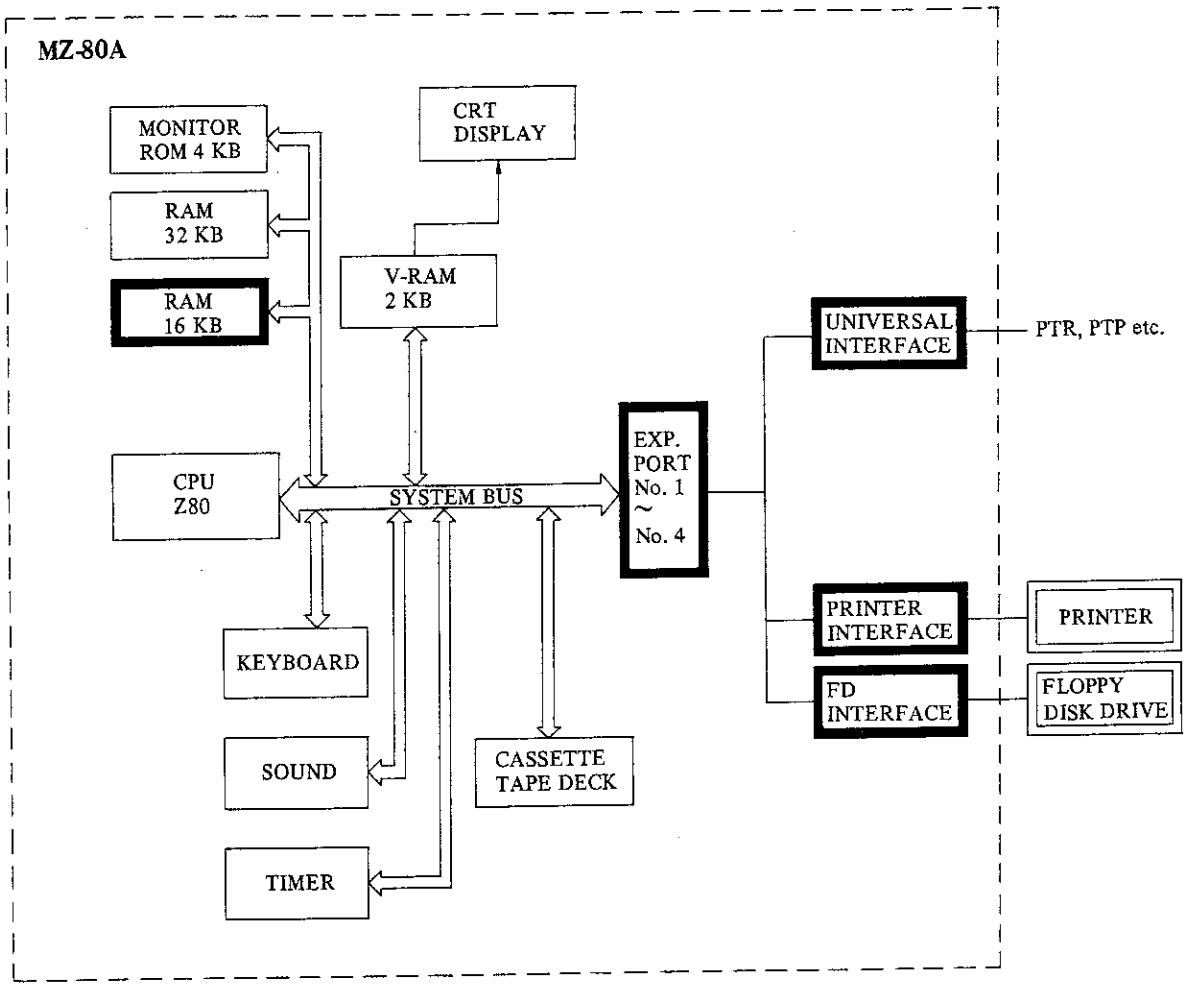


Figura 3.13 Periferiche del personal computer MZ-80A.

ATTENZIONE E AVVERTENZA

Attenzione: Nel mobiletto principale ci sono voltaggi molto elevati. Si facciano fare le istallazioni delle parti opzionali dal personale specializzato del rivenditore.

Avvertenza: Se si accende l'apparecchio con il mobiletto dell'appareccio sollevato, si possono dannefgiare delle parti elettriche.

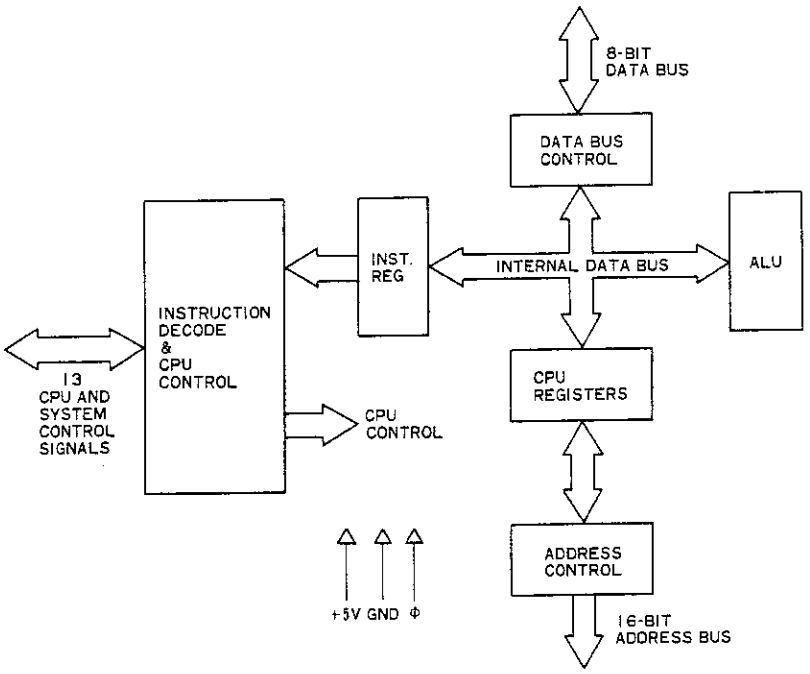
Oggetti metallici che rimangono nel mobiletto possono causare seri danni.

Ci si assicuri di non aver lasciato cadere oggetti metallici come fermagli ecc all'interno del mobiletto.

3.4 Technical data of Z-80 CPU

1.0 ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in Figure 1.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



Z-80 CPU BLOCK DIAGRAM
FIGURE 1.0-1

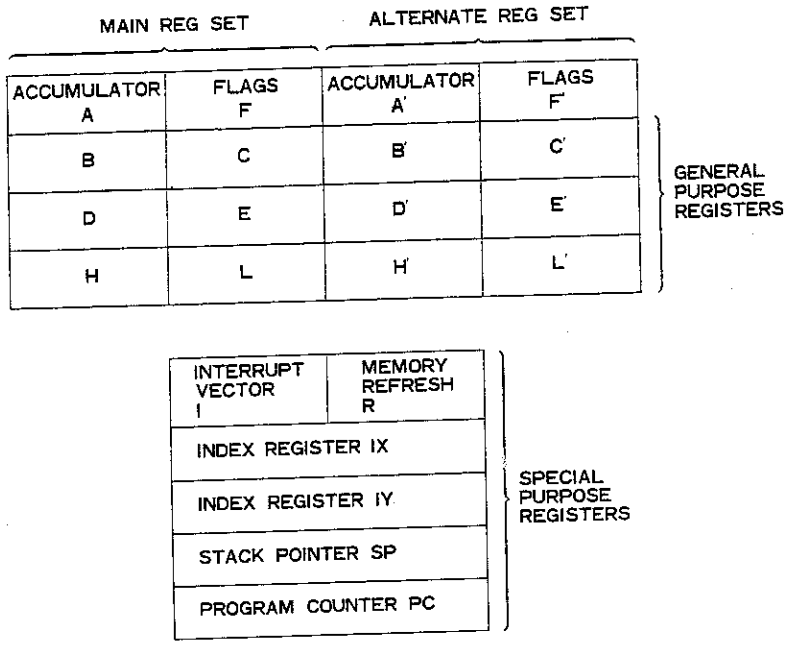
1.1 CPU REGISTERS

The Z-80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 1.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.

2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.



Z-80 CPU REGISTER CONFIGURATION
FIGURE 1.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z-80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8-bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address but along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC', DE' and HL'. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange commands need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

1.2 ARITHMETIC AND LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

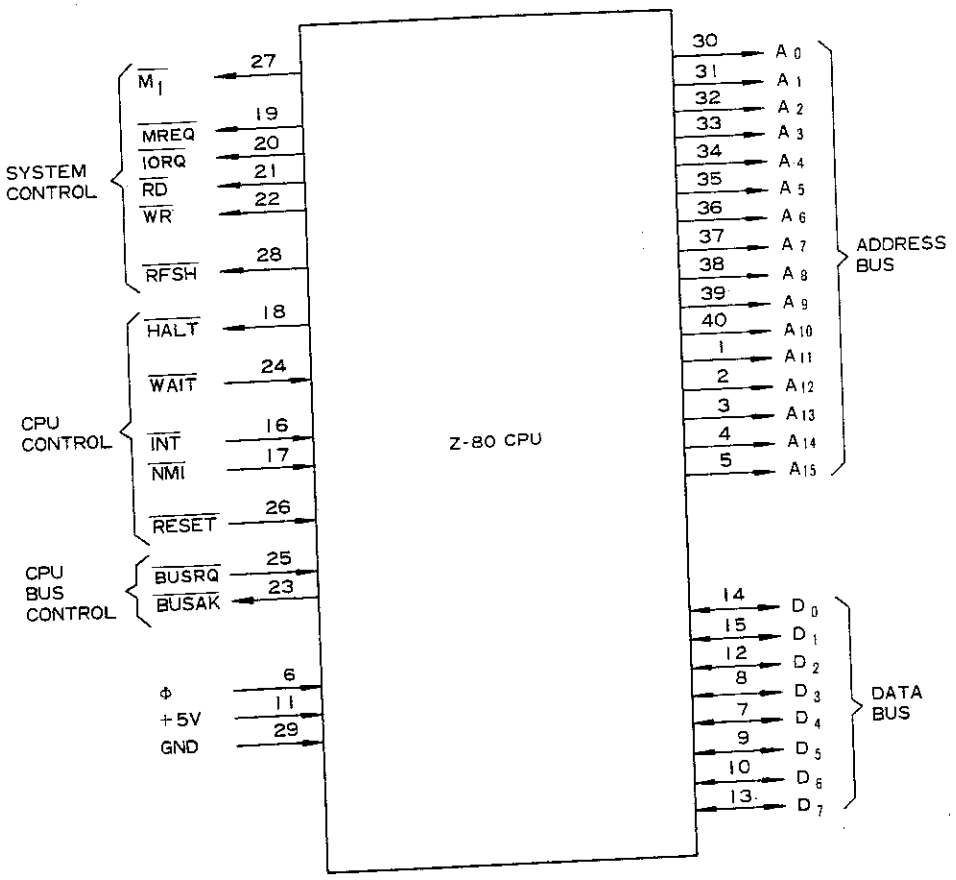
Add
Subtract
Logical AND
Logical OR
Logical Exclusive OR
Compare
Left or right shifts or rotates (arithmetic and logical)
Increment
Decrement
Set bit
Reset bit
Test bit

1.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the ALU and provide all required external control signals.

2.0 PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in Figure 2.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 2.0-1

A_0-A_{15} (Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanger and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7 (Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M_1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M_1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$ (Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{IORQ}$ (Input/Output Request)	Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An IORQ signal is also generated with an $\overline{M_1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.
$\overline{RD}$ (Memory Read)	Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
$\overline{WR}$ (Memory Write)	Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.
$\overline{RFSH}$ (Refresh)	Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.
$\overline{HALT}$ (Halt state)	Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.
$\overline{WAIT}$ (Wait)	Input, active low. $\overline{WAIT}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.
$\overline{INT}$ (Interrupt Request)	Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{BUSRQ}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{IORQ}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes.
$\overline{NMI}$ (Non Maskable Interrupt)	Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80 CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{BUSRQ}$ will override a $\overline{NMI}$.

$\overline{\text{RESET}}$

Input, active low, $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{\text{BUSRQ}}$
(Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$
(Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements. (2 MHz)

3.0 INSTRUCTION SET

The Z-80 CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

3.1 INTRODUCTION TO INSTRUCTION TYPES

The load instructions move data internally between CPU registers or between CPU registers and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general purpose registers such as move the data to Register B from Register C. This group also includes load immediate to any CPU register or to any to Register B from Register C. This group also includes load immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z-80. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z-80 block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found or the end of the block is reached, the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The rotate and shift group allows any register or any memory location to be rotated right or left with or without carry either arithmetic or logical. Also, a digit in the accumulator can be rotated right or left with two digits in any memory location.

The bit manipulation instructions allow any bit in the accumulator, any general purpose register or any external memory location to be set, reset or tested with a single instruction. For example, the most significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general purpose programming.

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of call is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

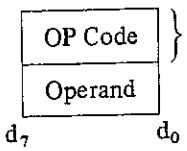
The input/output group of instructions in the Z-80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z-80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z-80 CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z-80 CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

3.2 ADDRESSING MODES

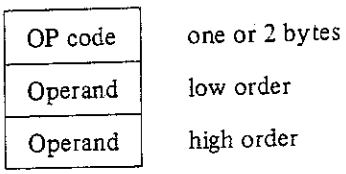
Most of the Z-80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z-80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



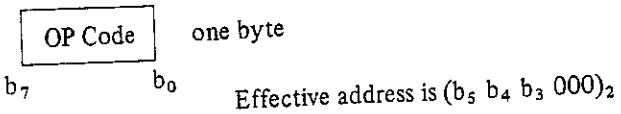
Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the OP codes are the operand.

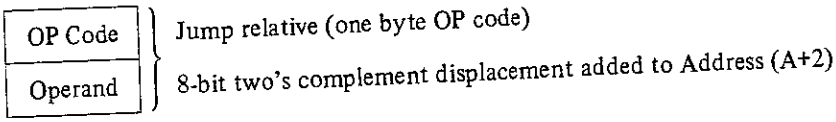


Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z-80 has a special single byte CALL instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

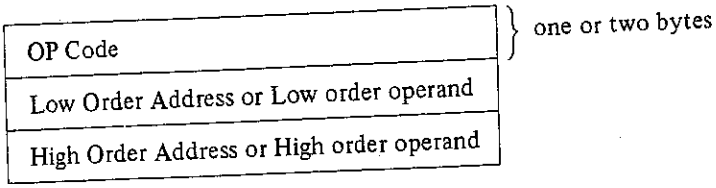


Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.



The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signal displacement can range between +127 and −128 from A+2. This allows for a total displacement of +129 to −126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

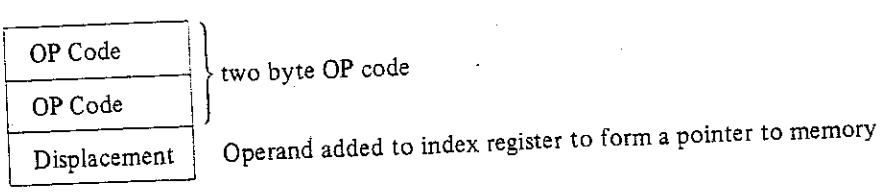
Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.



Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.



An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z-80 are referred to as IX and IY. To indicate indexed addressing the notation:

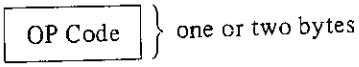
(IX + d) or (IY + d)

is used. Here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z-80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.



An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z-80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z-80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

3.3 INSTRUCTION OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z-80 instructions set. The instructions are logically arranged into groups as shown on tables 3.3-1 through 3.3-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction.

All instruction OP codes are listed in binary notation. Single byte OP codes require two hex characters while double byte OP codes require four hex characters. The conversion from binary to hex is repeated here for convenience.

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r,r'	r←r'	•	•	•	•	•	•	01 r r'	1	1	4	r,r' Reg.
LD r,n	r←n	•	•	•	•	•	•	00 r 110 ← n →	2	2	7	000 B 001 C 010 D 011 E 100 H 101 L 111 A
LD r,(HL)	r←(HL)	•	•	•	•	•	•	01 r 110	1	2	7	
LD r,(IX+d)	r←(IX+d)	•	•	•	•	•	•	11 011 101 01 r 110 ← d →	3	5	19	
LD r,(IY+d)	r←(IY+d)	•	•	•	•	•	•	11 111 101 01 r 110 ← d →	3	5	19	
LD (HL),r	(HL)←r	•	•	•	•	•	•	01 110 r	1	2	7	
LD (IX+d),r	(IX+d)←r	•	•	•	•	•	•	11 011 101 01 110 r ← d →	3	5	19	
LD (IY+d),r	(IY+d)←r	•	•	•	•	•	•	11 111 101 01 110 r ← d →	3	5	19	
LD (HL),n	(HL)←n	•	•	•	•	•	•	00 110 110 ← n →	2	3	10	
LD (IX+d),n	(IX+d)←n	•	•	•	•	•	•	11 011 101 00 110 110 ← d → ← n →	4	5	19	
LD (IY+d),n	(IY+d)←n	•	•	•	•	•	•	11 111 101 00 110 110 ← d → ← n →	4	5	19	
LD A,(BC)	A←(BC)	•	•	•	•	•	•	00 001 010	1	2	7	
LD A,(DE)	A←(DE)	•	•	•	•	•	•	00 011 010	1	2	7	
LD A,(nn)	A←(nn)	•	•	•	•	•	•	00 111 010 ← n → ← n →	3	4	13	
LD (BC),A	(BC)←A	•	•	•	•	•	•	00 000 010	1	2	7	
LD (DE),A	(DE)←A	•	•	•	•	•	•	00 010 010	1	2	7	
LD (nn),A	(nn)←A	•	•	•	•	•	•	00 110 010 ← n → ← n →	3	4	13	
LD A,I	A←I	•	‡	IFF2	‡	0	0	11 101 101 01 010 111	2	2	9	
LD A,R	A←R	•	‡	IFF2	‡	0	0	11 101 101 01 011 111	2	2	9	
LD I,A	I←A	•	•	•	•	•	•	11 101 101 01 000 111	2	2	9	
LD R,A	R←A	•	•	•	•	•	•	11 101 101 01 001 111	2	2	9	

Notes: r, r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

8-BIT LOAD GROUP
TABLE 3.3-1

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments	
		C	Z	P/V	S	N	H	76 543 210				dd	Pair
LD dd,nn	dd←nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	00	BC
LD IX,nn	IX←nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	4	4	14	01	DE
LD IY,nn	IY←nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	4	4	14	10	HL
LD HL,(nn)	H←(nn+1) L←(nn)	•	•	•	•	•	•	00 101 010 ← n → ← n →	3	5	16	11	SP
LD dd,(nn)	dd_H←(nn+1) dd_L←(nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	4	6	20		
LD IX,(nn)	IX_H←(nn+1) IX_L←(nn)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	4	6	20		
LD IY,(nn)	IY_H←(nn+1) IY_L←(nn)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	4	6	20		
LD (nn),HL	(nn+1)←H (nn)←L	•	•	•	•	•	•	00 100 010 ← n → ← n →	3	5	16		
LD (nn),dd	(nn+1)←dd_H (nn)←dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	4	6	20		
LD (nn),IX	(nn+1)←IX_H (nn)←IX_L	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	4	6	20		
LD (nn),IY	(nn+1)←IY_H (nn)←IY_L	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	4	6	20		
LD SP,HL	SP←HL	•	•	•	•	•	•	11 111 001	1	1	6		
LD SP,IX	SP←IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10		
LD SP,IY	SP←IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10		

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments	
		C	Z	P/V	S	N	H	76 543 210				qq	Pair
PUSH qq	$(SP-2) \leftarrow qq_L$ $(SP-1) \leftarrow qq_H$	•	•	•	•	•	•	11 qq0 101	1	3	11	00	BC
PUSH IX	$(SP-2) \leftarrow IX_L$ $(SP-1) \leftarrow IX_H$	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	01	DE
PUSH IY	$(SP-2) \leftarrow IY_L$ $(SP-1) \leftarrow IY_H$	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	10	HL
POP qq	$qq_H \leftarrow (SP+1)$ $qq_L \leftarrow (SP)$	•	•	•	•	•	•	11 qq0 001	1	3	10	11	AF
POP IX	$IX_H \leftarrow (SP+1)$ $IX_L \leftarrow (SP)$	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14		
POP IY	$IY_H \leftarrow (SP+1)$ $IY_L \leftarrow (SP)$	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14		

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
E.g. $BC_L = C$, $AF_H = A$

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ flag is affected according to the result of the operation.

16-BIT LOAD GROUP
TABLE 3.3-2

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
EX DE,HL	DE↔HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF,AF'	AF↔AF'	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix}\leftrightarrow\begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP),HL	H↔(SP+1) L↔(SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP),IX	IX_H↔(SP+1) IX_L↔(SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP),IY	IY_H↔(SP+1) IY_L↔(SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
				①								
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	‡	•	0	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Repeat until BC=0	•	•	0	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	If BC≠0 If BC=0
				①								
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	•	•	‡	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE)←(HL) DE←DE-1 HL←HL-1 BC←BC-1 Repeat until BC=0	•	•	0	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC≠0 If BC=0
				②	①							
CPI	A−(HL) HL←HL+1 BC←BC−1	•	‡	‡	‡	1	‡	11 101 101 10 100 001	2	4	16	

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
			②	①								
CPIR	A−(HL) HL←HL+1 BC←BC−1 Repeat until A=(HL) or BC=0	•	‡	‡	‡	1	‡	11 101 101 10 110 001	2 2	5 4	21 16	If BC≠0 and A≠(HL) If BC=0 or A=(HL)
			②	①								
CPD	A−(HL) HL←HL−1 BC←BC−1	•	‡	‡	‡	1	‡	11 101 101 10 101 001	2	4	16	
			②	①								
CPDR	A−(HL) HL←HL−1 BC←BC−1 Repeat until A=(HL) or BC=0	•	‡	‡	‡	1	‡	11 101 101 10 111 001	2 2	5 4	21 16	If BC≠0 and A≠(HL) If BC=0 or A=(HL)

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP
TABLE 3.3-3

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A,r	A←A+r	‡	‡	V	‡	0	‡	10 000 r	1	1	4	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
ADD A,n	A←A+n	‡	‡	V	‡	0	‡	11 000 110 ← n →	2	2	7	
ADD A,(HL)	A←A+(HL)	‡	‡	V	‡	0	‡	10 000 110	1	2	7	
ADD A,(IX+d)	A←A+(IX+d)	‡	‡	V	‡	0	‡	11 011 101 10 000 110 ← d →	3	5	19	
ADD A,(IY+d)	A←A+(IY+d)	‡	‡	V	‡	0	‡	11 111 101 10 000 110 ← d →	3	5	19	
ADC A,s	A←A+s+CY	‡	‡	V	‡	0	‡	001				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction The indicated bits replace the 000 in the ADD set above.
SUB s	A←A−s	‡	‡	V	‡	1	‡	010				
SBC A,s	A←A−s−CY	‡	‡	V	‡	1	‡	011				
AND s	A←A∧s	0	‡	P	‡	0	1	100				
OR s	A←A∨s	0	‡	P	‡	0	0	110				
XOR s	A←A∀s	0	‡	P	‡	0	0	101				
CP s	A−s	‡	‡	V	‡	1	‡	111				
INC r	r←r+1	•	‡	V	‡	0	‡	00 r 100	1	1	4	
INC (HL)	(HL)←(HL)+1	•	‡	V	‡	0	‡	00 110 100	1	3	11	
INC (IX+d)	(IX+d)←(IX+d)+1	•	‡	V	‡	0	‡	11 011 101 00 110 100 ← d →	3	6	23	
INC (IY+d)	(IY+d)←(IY+d)+1	•	‡	V	‡	0	‡	11 111 101 00 110 100 ← d →	3	6	23	
DEC m	m←m−1	•	‡	V	‡	1	‡	101				m is any of r, (HL), (IX+d), (IY+d) as shown for INC. Same format and states as INC. Replace 100 with 101 in OP-code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow, P = 1 means parity of the result is even, P = 0 means panty of the result is odd.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

8-BIT ARITHMETIC AND LOGICAL GROUP
TABLE 3.3-4

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments
		C	Z	P/V	S	N	H	76 543 210				
DAA	Converts acc content into packed BCD following add or subtract with packed BCD operands	‡	‡	P	‡	•	‡	00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \bar{A}$	•	•	•	•	1	1	00 101 111	1	1	4	Complement accumulator (one's complement) N
NEG	$A \leftarrow \bar{A}+1$	‡	‡	V	‡	1	‡	11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	‡	•	•	•	0	X	00 111 111	1	1	4	Complement carry flag
SCF	CY←1	1	•	•	•	0	0	00 110 111	1	1	4	Set carry flag
NOP	No operation PC←PC+1	•	•	•	•	•	•	00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	1	1	4	
DI	IFF←0	•	•	•	•	•	•	11 110 011	1	1	4	
EI	IFF←1	•	•	•	•	•	•	11 111 011	1	1	4	
IM 0	Sst interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	2	2	8	
IM 2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

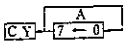
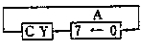
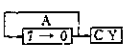
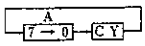
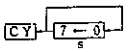
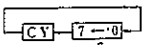
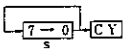
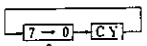
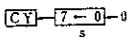
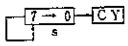
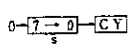
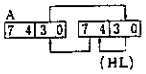
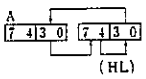
GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS
TABLE 3.3-5

Mnemonic	Symbolic Operation	Flags						OP-Code	No. of Bytes	No. of M Cycles	No. of T States	Comments	
		C	Z	P/V	S	N	H	76 543 210				ss	Reg.
ADD HL,ss	HL←HL+ss	‡	•	•	•	0	X	00 ss1 001	1	3	11	00	BC
ADC HL,ss	HL←HL+ss+CY	‡	‡	V	‡	0	X	11 101 101 01 ss1 010	2	4	15	01	DE
SBC HL,ss	HL←HL−ss−CY	‡	‡	V	‡	1	X	11 101 101 01 ss0 010	2	4	15	10	HL
ADD IX,pp	IX←IX+pp	‡	•	•	•	0	X	11 011 101 00 pp1 001	2	4	15	11	SP
												pp	Reg.
												00	BC
												01	DE
												10	IX
												11	SP
ADD IY,rr	IY←IY+rr	‡	•	•	•	0	X	11 111 101 00 rr1 001	2	4	15	rr	Reg.
												00	BC
												01	DE
												10	IY
												11	SP
INC ss	ss←ss+1	•	•	•	•	•	•	00 ss0 011	1	1	6		
INC IX	IX←IX+1	•	•	•	•	•	•	11 011 101 00 100 011	2	2	10		
INC IY	IY←IY+I	•	•	•	•	•	•	11 111 101 00 100 011	2	2	10		
DEC ss	ss←ss−1	•	•	•	•	•	•	00 ss1 011	1	1	6		
DEC IX	IX←IX−1	•	•	•	•	•	•	11 011 101 00 101 011	2	2	10		
DEC IY	IY←IY−1	•	•	•	•	•	•	11 111 101 00 101 011	2	2	10		

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

16-BIT ARITHMETIC GROUP
TABLE 3.3-6

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLC A		‡	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RL A		‡	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRC A		‡	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RR A		‡	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		‡	‡	P	‡	0	0	11 001 011 00 [000] r	2	2	8	Rotate left circular register r
RLC (HL)		‡	‡	P	‡	0	0	11 001 011 00 [000] 110	2	4	15	
RLC (IX+d)		‡	‡	P	‡	0	0	11 011 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RLC (IY+d)		‡	‡	P	‡	0	0	11 111 101 11 001 011 ← d → 00 [000] 110	4	6	23	
RL s		‡	‡	P	‡	0	0	[010]				Instruction format and states are as shown for RLC, m. To form new OP-code replace [000] of RLC, m with shown code.
RRC s		‡	‡	P	‡	0	0	[001]				
RR s		‡	‡	P	‡	0	0	[011]				
SLA s		‡	‡	P	‡	0	0	[100]				
SRA s		‡	‡	P	‡	0	0	[101]				
SRL s		‡	‡	P	‡	0	0	[111]				
RLD		•	‡	P	‡	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected.
RRD		•	‡	P	‡	0	0	11 101 101 01 100 111	2	5	18	

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

ROTATE AND SHIFT GROUP
TABLE 3.3-7

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b,r	$Z \leftarrow \overline{r_b}$	•	‡	X	X	0	1	11 001 011 01 b r	2	2	8	
BIT b,(HL)	$Z \leftarrow \overline{(HL)_b}$	•	‡	X	X	0	1	11 001 011 01 b 110	2	3	12	
BIT b,(IX+d)	$Z \leftarrow \overline{(IX+d)_b}$	•	‡	X	X	0	1	11 011 101 11 001 011 ← d → 01 b 110	4	5	20	
BIT b,(IY+d)	$Z \leftarrow \overline{(IY+d)_b}$	•	‡	X	X	0	1	11 111 101 11 001 011 ← d → 01 b 110	4	5	20	
SET b,r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b r	2	2	8	
SET b,(HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b 110	2	4	15	
SET b,(IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 ← d → [11] b 110	4	6	23	
SET b,(IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 ← d → [11] b 110	4	6	23	
RES b,s	$s_b \leftarrow 0$ s ≡ r,(HL), (IX+d), (IY+d)							[10]				To form new OP-code replace [11] of SET b, m with [10]. Flags and time states for SET instruction.

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	Bit Tested
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

BIT SET, RESET AND TEST GROUP
TABLE 3.3-8

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC←nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	3	3	10	
JP cc,nn	If condition cc is true PC←nn, otherwise continue	•	•	•	•	•	•	11 cc 010 ← n → ← n →	3	3	10	cc: 000 NZ non zero; 001 Z zero; 010 NC non carry; 011 C carry; 100 PO parity odd; 101 PE parity even; 110 P sign positive; 111 M sign negative
JR e	PC←PC+e	•	•	•	•	•	•	00 011 000 ← e-2 →	2	3	12	
JR C,e	If C=0 continue	•	•	•	•	•	•	00 111 000 ← e-2 →	2	2	7	If condition not met
	If C=1 PC←PC+e								2	3	12	If condition is met
JR NC,e	If C=1 continue	•	•	•	•	•	•	00 110 000 ← e-2 →	2	2	7	If condition not met
	If C=0 PC←PC+e								2	3	12	If condition is met
JR Z,e	If Z=0 continue	•	•	•	•	•	•	00 101 000 ← e-2 →	2	2	7	If condition not met
	If Z=1 PC←PC+e								2	3	12	If condition is met
JR NZ,e	If Z=1 continue	•	•	•	•	•	•	00 100 000 ← e-2 →	2	2	7	If condition not met
	If Z=0 PC←PC+e								2	3	12	If condition is met
JP (HL)	PC←HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC←IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC←IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ,e	B←B−1 If B=0 continue	•	•	•	•	•	•	00 010 000 ← e-2 →	2	2	8	If B=0
	If B=0 PC←PC+e								2	3	13	If B=0

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <−126, 129>

e−2 in the op-code provides an effective address of pc +e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

JUMP GROUP
TABLE 3.3-9

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC\leftarrow nn$	•	•	•	•	•	•	11 001 101 ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11 cc 100 ← n → ← n →	3 3	3 5	10 17	If cc is false If cc is true
RET	$PC_L\leftarrow(SP)$ $PC_H\leftarrow(SP+1)$	•	•	•	•	•	•	11 001 001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11 cc 000	1 1	1 3	5 11	If cc is false If cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11 101 101 01 001 101	2	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 101 101 01 000 101	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H\leftarrow 0$ $PC_L\leftarrow P$	•	•	•	•	•	•	11 t 111	1	3	11	

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
‡ = flag is affected according to the result of the operation.

CALL AND RETURN GROUP
TABLE 3.3-10

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A,(n)	A←(n)	•	•	•	•	•	•	11 011 011 ← n →	2	3	11	n to A_0~A_7 Acc to A_8~A_{15}
IN r,(C)	r←(C) If r=110 only the flags will be affected	•	‡ ①	P	‡	0	0	11 101 101 01 r 000	2	3	12	C to A_0~A_7 B to A_8~A_{15}
INI	(HL)←(C) B←B−1 HL←HL+1	•	‡	X	X	1	X	11 101 101 10 100 010	2	4	16	C to A_0~A_7 B to A_8~A_{15}
INIR	(HL)←(C) B←B−1 HL←HL+1 Repeat until B=0	•	1 ①	X	X	1	X	11 101 101 10 110 010	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}
IND	(HL)←(C) B←B−1 HL←HL−1	•	‡	X	X	1	X	11 101 101 10 101 010	2	4	16	C to A_0~A_7 B to A_8~A_{15}
INDR	(HL)←(C) B←B−1 HL←HL−1 Repeat until B=0	•	1	X	X	1	X	11 101 101 10 111 010	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}
OUT (n),A	(n)←A	•	•	•	•	•	•	11 010 011	2	3	11	n to A_0~A_7 Acc to A_8~A_{15}
OUT (C),r	(C)←r	•	• ①	•	•	•	•	11 101 101 01 r 001	2	3	12	C to A_0~A_7 B to A_8~A_{15}
OUTI	(C)←(HL) B←B−1 HL←HL+1	•	‡	X	X	1	X	11 101 101 10 100 011	2	4	16	C to A_0~A_7 B to A_8~A_{15}
OTIR	(C)←(HL) B←B−1 HL←HL+1 Repeat until B=0	•	1 ①	X	X	1	X	11 101 101 10 110 011	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}
OUTD	(C)←(HL) B←B−1 HL←HL−1	•	‡	X	X	1	X	11 101 101 10 101 011	2	4	16	C to A_0~A_7 B to A_8~A_{15}
OTDR	(C)←(HL) B←B−1 HL←HL−1 Repeat until B=0	•	1	X	X	1	X	11 101 101 10 111 011	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0~A_7 B to A_8~A_{15}

Notes: ① If the result of B-1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

INPUT AND OUTPUT GROUP
TABLE 3.3-11

3.4 FLAGS

Each of the two Z-80 CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) **Carry Flag (C)** – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) **Zero Flag (Z)** – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) **Sign Flag (S)** – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) **Parity/Overflow Flag (P/V)** – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z-80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (−128) number than can be represented in two's complement notation. For example consider adding:

```
+120 =    0111 1000
+105 =    0110 1001
  C = 0   1110 0001  = −31 (wrong) Overflow has occurred
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
 −5 =    1111 1011
−16 =    1111 0000
  C = 1  1110 1011  = −21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) **Half carry (H)** – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) **Add/Subtract Flag (N)** – Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 3.4-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, a '0' means that it is reset, a '1' means that it is set and the symbol '↕' indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 3.4-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B = 0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or I register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A, s	‡	‡	V	‡	0	‡	8-bit add or add with carry
SUB s; SBC A, s; CP s; NEG	‡	‡	V	‡	1	‡	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	‡	P	‡	0	1	Logical operations
OR s; XOR s	0	‡	P	‡	0	0	And set's different flags
INC s	•	‡	V	‡	0	‡	8-bit increment
DEC m	•	‡	V	‡	1	‡	8-bit decrement
ADD DD, ss	‡	•	•	•	0	X	16-bit add
ADC HL, ss	‡	‡	V	‡	0	X	16-bit add with carry
SBC HL, ss	‡	‡	V	‡	1	X	16-bit subtract with carry
RLA; RLCA; RRA; RRCA;	‡	•	•	•	0	0	Rotate accumulator
RL m; RLC m; RR m; RRC m SLA m; SRA m; SRL m	‡	‡	P	‡	0	0	Rotate and shift location s
RLD; RRD	•	‡	P	‡	0	0	Rotate digit left and right
DAA	‡	‡	P	‡	•	‡	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	‡	•	•	•	0	X	Complement carry
IN r, (C)	•	‡	P	‡	0	0	Input register indirect
INI; IND; OUTI; OUTD	•	‡	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI; LDD	•	X	‡	X	0	0	Block transfer instructions
LDIR; LDDR	•	X	0	X	0	0	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI; CPIR; CPD; CPDR	•	‡	‡	X	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	‡	IFF	‡	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	‡	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	‡	‡	V	‡	1	‡	Negative accumulator

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z = 1 if the result of the operation is zero.
S	Sign flag. S = 1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
‡	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

SUMMARY OF FLAG OPERATION
TABLE 3.4-1

4.0 INTERRUPT RESPONSE

The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

INTERRUPT ENABLE – DISABLE

The Z-80 CPU has two interrupt inputs, a software maskable interrupt and a non maskable interrupt. The non maskable interrupt (NMI) can *not* be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt (INT) can be selectively enable or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z-80 CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF_1 and IFF_2.

IFF_1	IFF_2
Actually disables interrupts from being accepted.	Temporary storage location for IFF_1.

The state of IFF_1 is used to actually inhibit interrupts while IFF_2 is used as a temporary storage location for IFF_1. The purpose of storing the IFF_1 will be subsequently explained.

A reset to the CPU will force both IFF_1 and IFF_2 to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF_1 and IFF_2 to the enable state. When an interrupt is accepted by the CPU, both IFF_1 and IFF_2 are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF_1 and IFF_2 are always equal.

The purpose of IFF_2 is to save the status of IFF_1 when a non maskable interrupt occurs. When a non maskable interrupt is accepted, IFF_1 is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF_1 has been saved so that the complete state of the CPU just prior to the non maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF_2 is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF_1 is thru the execution of a Return From Non Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF_2 are now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non maskable interrupt will be restored automatically.

Figure 4.0-1 is a summary of the effect of different instructions on the two enable flip flops.

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	$IFF_2 \rightarrow$ Parity flag
LD A, R	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$

"•" indicates no change

FIGURE 4.0-1
INTERRUPT ENABLE/DISABLE FLIP FLOPS

CPU RESPONSE

Non Maskable

A nonmaskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control.

After the application of RESET the CPU will automatically enter interrupt Mode 0.

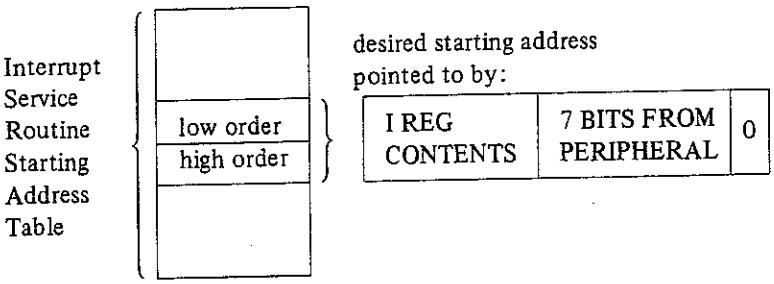
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

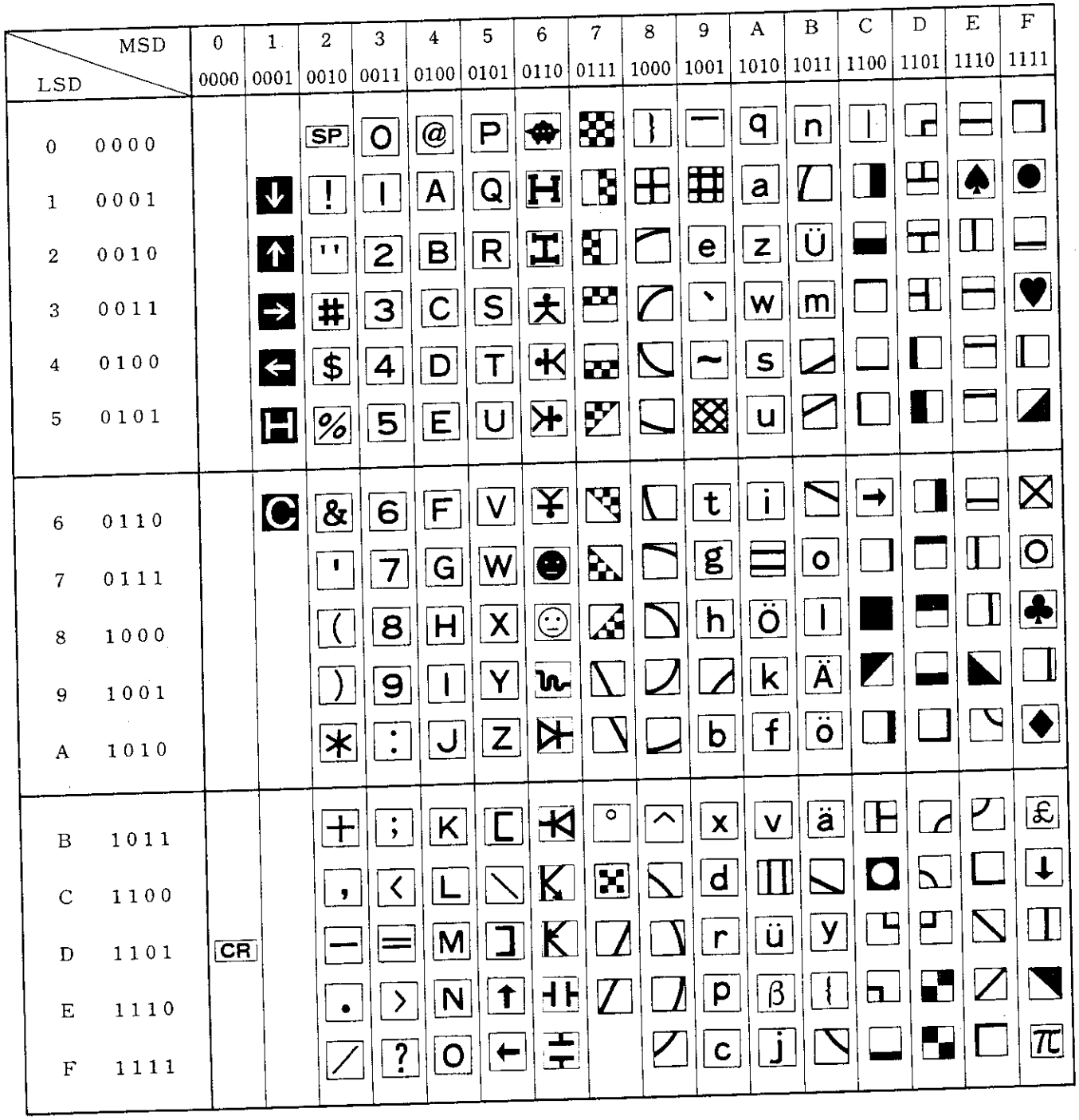
Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting devices supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Appendice

A.1 Tavola dei Codici ASCII

Qui di seguito viene riportata la tavola dei condici ASCII.



A.2 Tavola dei codici di visualizzazione

Qui di seguito viene riportata la tavola dei codici di visualizzazione dell'MZ-80A:

LSD \ MSD	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0 0000	SP	P	O				π			p						
1 0001	A	Q	!				!		a	q						
2 0010	B	R	2				"		b	r						
3 0011	C	S	3				#		c	s						
4 0100	D	T	4				$		d	t						
5 0101	E	U	5				%		e	u	~					
6 0110	F	V	6				&		f	v						
7 0111	G	W	7				'		g	w						
8 1000	H	X	8				(		h	x						
9 1001	I	Y	9				)		i	y						
A 1010	J	Z	—				+		j	z	β					
B 1011	K	£	=				*		k	ä	ü			°		
C 1100	L		;						l	ö	{					
D 1101	M		/						m	Ü			¥			
E 1110	N		.						n	Ä	^					
F 1111	O		,						o	Ö	—					

A. 3 Codici mnemonici e corrispondenti codici oggetto

(I codici mnemonici sono disposti in ordine alfabetico)

Codice OP	Mnemonic
8E	ADC A,(HL)
DD8E*05*	ADC A,(IX+d)
FD8E*05*	ADC A,(IY+d)
8F	ADC A,A
88	ADC A,B
89	ADC A,C
8A	ADC A,D
8B	ADC A,E
8C	ADC A,H
8D	ADC A,L
CE*20*	ADC A,n
ED4A	ADC HL,BC
ED5A	ADC HL,DE
ED6A	ADC HL,HL
ED7A	ADC HL,SP
86	ADD A,(HL)
DD86*05*	ADD A,(IX+d)
FD86*05*	ADD A,(IY+d)
87	ADD A,A
80	ADD A,B
81	ADD A,C
82	ADD A,D
83	ADD A,E
84	ADD A,H
85	ADD A,L
C6*20*	ADD A,n
09	ADD HL,BC
19	ADD HL,DE
29	ADD HL,HL
39	ADD HL,SP
DD09	ADD IX,BC
DD19	ADD IX,DE
DD29	ADD IX,IX
DD39	ADD IX,SP
FD09	ADD IY,BC
FD19	ADD IY,DE
FD29	ADD IY,IY
FD39	ADD IY,SP

Nota

nn, n, d ed e negli operandi di ogni codice mnemonico rappresentano il dato costante. I valori d'esempio per ogni codice mnemonico più oltre riportati sono usati per queste costanti in questa tavola.

nn = 584H
n = 20H
d = 5
e = 30H

I codici di dato rappresentati dai valori d'esempio sono mostrati in caratteri italici e sottolineati.

Codice OP	Mnemonic
A6	AND (HL)
DDA6*05*	AND (IX+d)
FDA6*05*	AND (IY+d)
A7	AND A
A0	AND B
A1	AND C
A2	AND D
A3	AND E
A4	AND H
A5	AND L
E6*20*	AND n
CB46	BIT 0, (HL)
DDCB*05*46	BIT 0, (IX+d)
FDCB*05*46	BIT 0, (IY+d)
CB47	BIT 0, A
CB40	BIT 0, B
CB41	BIT 0, C
CB42	BIT 0, D
CB43	BIT 0, E
CB44	BIT 0, H
CB45	BIT 0, L
CB4E	BIT 1, (HL)
DDCB*05*4E	BIT 1, (IX+d)
FDCB*05*4E	BIT 1, (IY+d)
CB4F	BIT 1, A
CB48	BIT 1, B
CB49	BIT 1, C
CB4A	BIT 1, D
CB4B	BIT 1, E
CB4C	BIT 1, H
CB4D	BIT 1, L
CB56	BIT 2, (HL)
DDCB*05*56	BIT 2, (IX+d)
FDCB*05*56	BIT 2, (IY+d)
CB57	BIT 2, A
CB50	BIT 2, B
CB51	BIT 2, C
CB52	BIT 2, D
CB53	BIT 2, E

Codice OP	Mnemonic
CB54	BIT 2, H
CB55	BIT 2, L
CB5E	BIT 3, (HL)
DDCB*05*5E	BIT 3, (IX+d)
FDCB*05*5E	BIT 3, (IY+d)
CB5F	BIT 3, A
CB58	BIT 3, B
CB59	BIT 3, C
CB5A	BIT 3, D
CB5B	BIT 3, E
CB5C	BIT 3, H
CB5D	BIT 3, L
CB66	BIT 4, (HL)
DDCB*05*66	BIT 4, (IX+d)
FDCB*05*66	BIT 4, (IY+d)
CB67	BIT 4, A
CB60	BIT 4, B
CB61	BIT 4, C
CB62	BIT 4, D
CB63	BIT 4, E
CB64	BIT 4, H
CB65	BIT 4, L
CB6E	BIT 5, (HL)
DDCB*05*6E	BIT 5, (IX+d)
FDCB*05*6E	BIT 5, (IY+d)
CB6F	BIT 5, A
CB68	BIT 5, B
CB69	BIT 5, C
CB6A	BIT 5, D
CB6B	BIT 5, E
CB6C	BIT 5, H
CB6D	BIT 5, L
CB76	BIT 6, (HL)
DDCB*05*76	BIT 6, (IX+d)
FDCB*05*76	BIT 6, (IY+d)
CB77	BIT 6, A
CB70	BIT 6, B
CB71	BIT 6, C
CB72	BIT 6, D
CB73	BIT 6, E

Codice OP	Mnemonic
CB74	BIT 6,H
CB75	BIT 6,L
CB7E	BIT 7,(HL)
DDCB*05*7E	BIT 7,(IX+d)
FDCB*05*7E	BIT 7,(IY+d)
CB7F	BIT 7,A
CB78	BIT 7,B
CB79	BIT 7,C
CB7A	BIT 7,D
CB7B	BIT 7,E
CB7C	BIT 7,H
CB7D	BIT 7,L
DC*8405*	CALL C,nn
FC*8405*	CALL M,nn
D4*8405*	CALL NC,nn
CD*8405*	CALL nn
C4*8405*	CALL NZ,nn
F4*8405*	CALL P,nn
EC*8405*	CALL PE,nn
E4*8405*	CALL PO,nn
CC*8405*	CALL Z,nn
3F	CCF
BE	CP (HL)
DDBE*05*	CP (IX+d)
FDBE*05*	CP (IY+d)
BF	CP A
B8	CP B
B9	CP C
BA	CP D
BB	CP E
BC	CP H
BD	CP L
FE*20*	CP n
EDA9	CPD
EDB9	CPDR
EDA1	CPI

Codice OP	Mnemonic
EDB1	CPIR
2F	CPL
27	DAA
35	DEC (HL)
DD35*05*	DEC (IX+d)
FD35*05*	DEC (IY+d)
3D	DEC A
05	DEC B
0B	DEC BC
0D	DEC C
15	DEC D
1B	DEC DE
1D	DEC E
25	DEC H
2B	DEC HL
DD2B	DEC IX
FD2B	DEC IY
2D	DEC L
3B	DEC SP
F3	DI
10*2E*	DJNZ e
FB	EI
E3	EX (SP),HL
DDE3	EX (SP),IX
FDE3	EX (SP),IY
08	EX AF,AF'
EB	EX DE,HL
D9	EXX
76	HALT
ED46	IM 0
ED56	IM 1

Codice OP	Mnemonic
ED5E	IM 2
ED78	IN A,(C)
DB*20*	IN A,(n)
ED40	IN B,(C)
ED48	IN C,(C)
ED50	IN D,(C)
ED58	IN E,(C)
ED60	IN H,(C)
ED68	IN L,(C)
34	INC (HL)
DD34*05*	INC (IX+d)
FD34*05*	INC (IY+d)
3C	INC A
04	INC B
03	INC BC
0C	INC C
14	INC D
13	INC DE
1C	INC E
24	INC H
23	INC HL
DD23	INC IX
FD23	INC IY
2C	INC L
33	INC SP
EDAA	IND
EDBA	INDR
EDA2	INI
EDB2	INIR
E9	JP (HL)
DDE9	JP (IX)
FDE9	JP (IY)
DA*8405*	JP C,nn
FA*8405*	JP M,nn
D2*8405*	JP NC,nn
C3*8405*	JP nn

Codice OP	Mnemonic
C2*8405*	JP NZ,nn
F2*8405*	JP P,nn
EA*8405*	JP PE,nn
E2*8405*	JP PO,nn
CA*8405*	JP Z,nn
38*2E*	JR C,e
18*2E*	JR e
30*2E*	JR NC,e
20*2E*	JR NZ,e
28*2E*	JR Z,e
02	LD (BC),A
12	LD (DE),A
77	LD (HL),A
70	LD (HL),B
71	LD (HL),C
72	LD (HL),D
73	LD (HL),E
74	LD (HL),H
75	LD (HL),L
36*20*	LD (HL),n
DD77*05*	LD (IX+d),A
DD70*05*	LD (IX+d),B
DD71*05*	LD (IX+d),C
DD72*05*	LD (IX+d),D
DD73*05*	LD (IX+d),E
DD74*05*	LD (IX+d),H
DD75*05*	LD (IX+d),L
DD36*0520*	LD (IX+d),n
FD77*05*	LD (IY+d),A
FD70*05*	LD (IY+d),B
FD71*05*	LD (IY+d),C
FD72*05*	LD (IY+d),D
FD73*05*	LD (IY+d),E
FD74*05*	LD (IY+d),H
FD75*05*	LD (IY+d),L
FD36*0520*	LD (IY+d),n
32*8405*	LD (nn),A
ED43*8405*	LD (nn),BC

Codice OP	Mnemonic
ED53*8405*	LD (nn),DE
22*8405*	LD (nn),HL
DD22*8405*	LD (nn),IX
FD22*8405*	LD (nn),IY
ED73*8405*	LD (nn),SP
0A	LD A,(BC)
1A	LD A,(DE)
7E	LD A,(HL)
DD7E*05*	LD A,(IX+d)
FD7E*05*	LD A,(IY+d)
3A *8405*	LD A,(nn)
7F	LD A,A
78	LD A,B
79	LD A,C
7A	LD A,D
7B	LD A,E
7C	LD A,H
ED57	LD A,I
7D	LD A,L
3E*20*	LD A,n
46	LD B,(HL)
DD46*05*	LD B,(IX+d)
FD46*05*	LD B,(IY+d)
47	LD B,A
40	LD B,B
41	LD B,C
42	LD B,D
43	LD B,E
44	LD B,H
45	LD B,L
06*20*	LD B,n
ED4B*8405*	LD BC,(nn)
01*8405*	LD BC,nn
4E	LD C,(HL)
DD4E*05*	LD C,(IX+d)
FD4E*05*	LD C,(IY+d)
4F	LD C,A
48	LD C,B
49	LD C,C
4A	LD C,D

Codice OP	Mnemonic
4B	LD C,E
4C	LD C,H
4D	LD C,L
0E*20*	LD C,n
56	LD D,(HL)
DD56*05*	LD D,(IX+d)
FD56*05*	LD D,(IY+d)
57	LD D,A
50	LD D,B
51	LD D,C
52	LD D,D
53	LD D,E
54	LD D,H
55	LD D,L
16*20*	LD D,n
ED5B*8405*	LD DE,(nn)
11*8405*	LD DE,nn
5E	LD E,(HL)
DD5E*05*	LD E,(IX+d)
FD5E*05*	LD E,(IY+d)
5F	LD E,A
58	LD E,B
59	LD E,C
5A	LD E,D
5B	LD E,E
5C	LD E,H
5D	LD E,L
1E*20*	LD E,n
66	LD H,(HL)
DD66*05*	LD H,(IX+d)
FD66*05*	LD H,(IY+d)
67	LD H,A
60	LD H,B
61	LD H,C
62	LD H,D
63	LD H,E
64	LD H,H
65	LD H,L
26*20*	LD H,n
2A*8405*	LD H,(nn)

Codice OP	Mnemonic
21*8405*	LD HL,nn
ED47	LD I,A
DD2A*8405*	LD IX,(nn)
DD21*8405*	LD IX,nn
FD2A*8405*	LD IY,(nn)
FD21*8405*	LD IY,nn
6E	LD L,(HL)
DD6E*05*	LD L,(IX+d)
FD6E*05*	LD L,(IY+d)
6F	LD L,A
68	LD L,B
69	LD L,C
6A	LD L,D
6B	LD L,E
6C	LD L,H
6D	LD L,L
2E*20*	LD L,n
ED7B*8405*	LD SP,(nn)
F9	LD SP,HL
DDF9	LD SP,IX
FDF9	LD SP,IY
31*8405*	LD SP,nn
EDA8	LDD
EDB8	LDDR
EDA0	LDI
EDB0	LDIR
ED44	NEG
00	NOP
B6	OR (HL)
DDB6*05*	OR (IX+d)
FDB6*05*	OR (IY+d)
B7	OR A
B0	OR B
B1	OR C
B2	OR D
B3	OR E
B4	OR H
B5	OR L
F6*20*	OR n
EDBB	OTDR
EDB3	OTIR
ED79	OUT (C),A
ED41	OUT (C),B
ED49	OUT (C),C
ED51	OUT (C),D
ED59	OUT (C),E
ED61	OUT (C),H
ED69	OUT (C),L
D3*20*	OUT (n),A
EDAB	OUTD
EDA3	OUTI
F1	POP AF
C1	POP BC
D1	POP DE
E1	POP HL
DDE1	POP IX
FDE1	POP IY
F5	PUSH AF
C5	PUSH BC
D5	PUSH DE
E5	PUSH HL
DDE5	PUSH IX
FDE5	PUSH IY
CB86	RES 0,(HL)
DDCB*05*86	RES 0,(IX+d)
FDCB*05*86	RES 0,(IY+d)
CB87	RES 0,A
CB80	RES 0,B
CB81	RES 0,C
CB82	RES 0,D
CB83	RES 0,E
CB84	RES 0,H

Codice OP	Mnemonic
CB85	RES 0,L
CB8E	RES 1,(HL)
DDCB*05*8E	RES 1,(IX+d)
FDCB*05*8E	RES 1,(IY+d)
CB8F	RES 1,A
CB88	RES 1,B
CB89	RES 1,C
CB8A	RES 1,D
CB8B	RES 1,E
CB8C	RES 1,H
CB8D	RES 1,L
CB96	RES 2,(HL)
DDCB*05*96	RES 2,(IX+d)
FDCB*05*96	RES 2,(IY+d)
CB97	RES 2,A
CB90	RES 2,B
CB91	RES 2,C
CB92	RES 2,D
CB93	RES 2,E
CB94	RES 2,H
CB95	RES 2,L
CB9E	RES 3,(HL)
DDCB*05*9E	RES 3,(IX+d)
FDCB*05*9E	RES 3,(IY+d)
CB9F	RES 3,A
CB98	RES 3,B
CB99	RES 3,C
CB9A	RES 3,D
CB9B	RES 3,E
CB9C	RES 3,H
CB9D	RES 3,L
CBA6	RES 4,(HL)
DDCB*05*A6	RES 4,(IX+d)
FDCB*05*A6	RES 4,(IY+d)
CBA7	RES 4,A
CBA0	RES 4,B
CBA1	RES 4,C
CBA2	RES 4,D
CBA3	RES 4,E
CBA4	RES 4,H
CBA5	RES 4,L
CBAE	RES 5,(HL)
DDCB*05*AE	RES 5,(IX+d)
FDCB*05*AE	RES 5,(IY+d)
CBAF	RES 5,A
CBA8	RES 5,B
CBA9	RES 5,C
CBAA	RES 5,D
CBAB	RES 5,E
CBAC	RES 5,H
CBAD	RES 5,L
CBB6	RES 6,(HL)
DDCB*05*B6	RES 6,(IX+d)
FDCB*05*B6	RES 6,(IY+d)
CBB7	RES 6,A
CBB0	RES 6,B
CBB1	RES 6,C
CBB2	RES 6,D
CBB3	RES 6,E
CBB4	RES 6,H
CBB5	RES 6,L
CBBE	RES 7,(HL)
DDCB*05*BE	RES 7,(IX+d)
FDCB*05*BE	RES 7,(IY+d)
CBBF	RES 7,A
CBB8	RES 7,B
CBB9	RES 7,C
CBBA	RES 7,D
CBBB	RES 7,E
CBBC	RES 7,H
CBBD	RES 7,L
C9	RET
D8	RET C
F8	RET M
D0	RET NC
C0	RET NZ
F0	RET P
E8	RET PE
E0	RET PO

Codice OP	Mnemonic
C8	RET Z
ED4D	RETI
ED45	RETN
CB16	RL (HL)
DDCB*05*16	RL (IX+d)
FDCB*05*16	RL (IY+d)
CB17	RL A
CB10	RL B
CB11	RL C
CB12	RL D
CB13	RL E
CB14	RL H
CB15	RL L
17	RLA
CB06	RLC (HL)
DDCB*05*06	RLC (IX+d)
FDCB*05*06	RLC (IY+d)
CB07	RLC A
CB00	RLC B
CB01	RLC C
CB02	RLC D
CB03	RLC E
CB04	RLC H
CB05	RLC L
07	RLCA
ED6F	RLD
CB1E	RR (HL)
DDCB*05*1E	RR (IX+d)
FDCB*05*1E	RR (IY+d)
CB1F	RR A
CB18	RR B
CB19	RR C
CB1A	RR D
CB1B	RR E
CB1C	RR H
CB1D	RR L
1F	RRA

Codice OP	Mnemonic
CB0E	RRC (HL)
DDCB*05*0E	RRC (IX+d)
FDCB*05*0E	RRC (IY+d)
CB0F	RRC A
CB08	RRC B
CB09	RRC C
CB0A	RRC D
CB0B	RRC E
CB0C	RRC H
CB0D	RRC L
0F	RRCA
ED67	RRD
C7	RST 00H
CF	RST 08H
D7	RST 10H
DF	RST 18H
E7	RST 20H
EF	RST 28H
F7	RST 30H
FF	RST 38H
9E	SBC A,(HL)
DD9E*05*	SBC A,(IX+d)
FD9E*05*	SBC A,(IY+d)
9F	SBC A,A
98	SBC A,B
99	SBC A,C
9A	SBC A,D
9B	SBC A,E
9C	SBC A,H
9D	SBC A,L
DE*20*	SBC A,n
ED42	SBC HL,BC
ED52	SBC HL,DE
ED62	SBC HL,HL
ED72	SBC HL,SP
37	SCF

Codice OP	Mnemonic
CBC6	SET 0,(HL)
DDCB*05*C6	SET 0,(IX+d)
FDCB*05*C6	SET 0,(IY+d)
CBC7	SET 0,A
CBC0	SET 0,B
CBC1	SET 0,C
CBC2	SET 0,D
CBC3	SET 0,E
CBC4	SET 0,H
CBC5	SET 0,L
CBCE	SET 1,(HL)
DDCB*05*CE	SET 1,(IX+d)
FDCB*05*CE	SET 1,(IY+d)
CBCF	SET 1,A
CBC8	SET 1,B
CBC9	SET 1,C
CBCA	SET 1,D
CBCB	SET 1,E
CBCC	SET 1,H
CBCD	SET 1,L
CBD6	SET 2,(HL)
DDCB*05*D6	SET 2,(IX+d)
FDCB*05*D6	SET 2,(IY+d)
CBD7	SET 2,A
CBD0	SET 2,B
CBD1	SET 2,C
CBD2	SET 2,D
CBD3	SET 2,E
CBD4	SET 2,H
CBD5	SET 2,L
CBD8	SET 3,B
CBDE	SET 3,(HL)
DDCB*05*DE	SET 3,(IX+d)
FDCB*05*DE	SET 3,(IY+d)
CBDF	SET 3,A
CBD9	SET 3,C
CBDA	SET 3,D
CBDB	SET 3,E
CBDC	SET 3,H
CBDD	SET 3,L

Codice OP	Mnemonic
CBE6	SET 4,(HL)
DDCB*05*E6	SET 4,(IX+d)
FDCB*05*E6	SET 4,(IY+d)
CBE7	SET 4,A
CBE0	SET 4,B
CBE1	SET 4,C
CBE2	SET 4,D
CBE3	SET 4,E
CBE4	SET 4,H
CBE5	SET 4,L
CBEE	SET 5,(HL)
DDCB*05*EE	SET 5,(IX+d)
FDCB*05*EE	SET 5,(IY+d)
CBEF	SET 5,A
CBE8	SET 5,B
CBE9	SET 5,C
CBEA	SET 5,D
CBEB	SET 5,E
CBEC	SET 5,H
CBED	SET 5,L
CBF6	SET 6,(HL)
DDCB*05*F6	SET 6,(IX+d)
FDCB*05*F6	SET 6,(IY+d)
CBF7	SET 6,A
CBF0	SET 6,B
CBF1	SET 6,C
CBF2	SET 6,D
CBF3	SET 6,E
CBF4	SET 6,H
CBF5	SET 6,L
CBFE	SET 7,(HL)
DDCB*05*FE	SET 7,(IX+d)
FDCB*05*FE	SET 7,(IY+d)
CBFF	SET 7,A
CBF8	SET 7,B
CBF9	SET 7,C
CBFA	SET 7,D
CBFB	SET 7,E
CBFC	SET 7,H
CBFD	SET 7,L

Codice OP	Mnemonic
CB26	SLA (HL)
DDCB*05*26	SLA (IX+d)
FDCB*05*26	SLA (IY+d)
CB27	SLA A
CB20	SLA B
CB21	SLA C
CB22	SLA D
CB23	SLA E
CB24	SLA H
CB25	SLA L
CB2E	SRA (HL)
DDCB*05*2E	SRA (IX+d)
FDCB*05*2E	SRA (IY+d)
CB2F	SRA A
CB28	SRA B
CB29	SRA C
CB2A	SRA D
CB2B	SRA E
CB2C	SRA H
CB2D	SRA L
CB3E	SRL (HL)
DDCB*05*3E	SRL (IX+d)
FDCB*05*3E	SRL (IY+d)
CB3F	SRL A
CB38	SRL B
CB39	SRL C
CB3A	SRL D
CB3B	SRL E
CB3C	SRL H
CB3D	SRL L
96	SUB (HL)
DD96*05*	SUB (IX+d)
FD96*05*	SUB (IY+d)
97	SUB A
90	SUB B
91	SUB C
92	SUB D

Codice OP	Mnemonic
93	SUB E
94	SUB H
95	SUB L
D6*20*	SUB n
AE	XOR (HL)
DDAE*05*	XOR (IX+d)
FDAE*05*	XOR (IY+d)
AF	XOR A
A8	XOR B
A9	XOR C
AA	XOR D
AB	XOR E
AC	XOR H
AD	XOR L
EE*20*	XOR n

A.4 Caratteristiche Tecniche

1. CARATTERISTCHE TECHNICHE GENERALI DELL'MZ-80A

CPU	SHARP LH0080 (Z80-CPU)
Orologio	2 MHz
Memoria	ROM 4K bytes (Monitor program) ROM 2K bytes (character generator) RAM 32K bytes (dynamic RAM) Può essere espansa fino a 48K byte (opzionale).
Display	9" CRT (display verde) Display dei caratteri Matrice a punti 8 x 8 Caratteri: 1000 (40 caratteri x 25 linee) Display grafico 80 x 50 punti
Cassetta	Cassetta audio a nastro magnetico standard Velocità transfer data: 1200 bit/sec Sistema transfer data: SHARP PWM

Lineamenti tasti	73 tasti Tastiera standard ASCII Numeri
Funzione di editing	Controllo del cursore: sù, giù, destra e sinistra, home e clear. Cancellazione ed inserimento.
Funzione orologio	Incorporata
Alimentazione	Alimentazione a voltaggio locale.
Temperatura	Temperatura di funzionamento: 0°C ~ 35°C Temperatura di immagazzinamento: –15°~60°C
Umidità	Inferiore all'80%
Peso	Circa 10 kg
Dimensioni	Larghezza 440mm Profondità 480mm Altezza 260mm

2. CARATTERISTICHE DELLA SEZIONE CPU BOARD

CPU	SHARP LH0080 (Z80-CPU)	1 pc.
ROM	MONITOR ROM (4K bytes) Character generator ROM (2K bytes)	1 pc. 1 pc.
RAM	Standard; 16K bits dynamic RAM (SHARP LH4116) Video RAM (2K bytes)	16 pcs. 1 pc.

Contatore programmabile	8253	1 pc.
Interfaccia periferica programmabile	8255	1 pc.

3. CARATTERISTICHE DELLA SEZIONE DI ALIMENTAZIONE

INPUT	Si usi una sorgente di alimentazione con lo stesso voltaggio indicato sulla piastrina.

OUTPUT	5V, –5V, 12V (stabilizzato) 12V (non stabilizzato)

4. CARATTERISTICHE DELLA SEZIONE DISPLAY

Formato	9"
Frequenza verticale e orizzontale	60Hz (verticale) 15,75KHz (orizzontale)
Alimentazione	CC 12V, 1,1A ±10%
Tubo a raggi catodici	E2728B3: 9" 90° di deflessione tipo antiesplodente Riscaldamento: 12V, 75mA
IC	2 pcs.
Transistori	7 pcs.
Diodi	13 pcs.
Uscita sonora	400mW max. (440Hz) Altoparlante da 8cm, tipo dinamico rotondo (32 Ω)
Controlli	Volume, V-Hold, contrasto, H-Hold, brillantezza, fuoco.
Temperatura di funzionamento	da –10°C a 50°C
Output video	40Vp-p standard (35Vp-p limit)
Risoluzione	Orizzontale *Il disegno a sinistra deve essere chiaro al centro dell'immagine.

Distorzione non lineare	Orizzontale: ±8% (±14% max.) Verticale: ±8% (±12% max.)
Distorsione geometrica	Distorsione a puntaspilli: 1% (2% max.) Distorsione di botte: 1% (2% max.) Distorsione trapezoidale: 1% (2% max.) Distorsione di parallelogramma: 1° (2.5° max.)
Alto voltaggio	Zero beam: 11.0kV (10.0kV, min., 12.0kV, max.)
Alimentazione	CC 12V
Gamma d'utenza	12V ±10%
Scansione	Orizzontale: 10% (15% max.) Verticale: 10% (15% max.)
Campo orizzontale	±300Hz (±100Hz limit)
Campo verticale	–12Hz (–6Hz limit)
Caratteristiche delle frequenze audio	440Hz (0dB) –10dB ±4dB at 100Hz –12dB ±4dB at 10kHz

5. CARATTERISTICHE DELLA SEZIONE PIASTRA MAGNETICA

Systema	Registrazione PWM
Alimentazione	5V ±0.25V
Semiconduttori	4 transistori 1 IC 4 diodi
Nastro	da C30 a C60
Banda	2 bande monoaurali

Polarizzazione	Sistema CC
Cancellazione	Sistema CC
Sensibilità playback	667 μsec. to 333 μsec. (standard)
Temperatura di funzionamento	–10°C to +40°C
Temperatura di immagazzinamento	–25°C to +65°C

A.5 Cura del sistema

- **Cavo dell'alimentazione**
  Non si mettano oggetti pesanti, come scrivanie o sedie, sul cavo dell'alimentazione altrimenti si rovina la copertura procurando in tal modo seri danni. Quando si eatrae al spina dalla presa di corrente non si tiri il cordone.

- **Voltaggio**
  Il voltaggio preciso è mostrato sulla piastrina. Voltaggi troppo alti o troppo bassi possono causare inconvenienti oppure operazioni sbagliate.

- **Ventilazione**
  Sul mobiletto ci sono dei fori di ventilazione. Non si metta l'unità su tappeti o su panni in wuanto si ostruiscono i fori di ventilazione che si trovano sul fondo. Si metta il set in un luogo ben ventilato.

- **Umidità e polvere**
  Si metta l'unità in un luogo che non sia né umido né polveroso.

- **Temperatura**
  Non si ponga l'unità sotto la diretta azione dei raggi solari e non la si metta, vicino a fonti di riscaldamento altrimenti la temperatura interna dell'unità diviene troppo alta.

- **Acqua ed altre sostanze estranee**
  Far funzionare l'unità quando è bagnata ovvero quando contiene oggetti estranei come spilli, fermagli di natura metallica, ecc, può essere pericoloso. Se acqua od altri liquidi entrano nell'unità si tolga immediatamente la spina dalla presa e si contatti immediatamente il proprio rivenditore.

- **Urti**
  Se l'unità viene sottoposta ad urti od a colpi le parti elettroniche estremamente sensibili in essa contenute possono essere irreparabilmente danneggiate.

- **Inconvenienti**
  Se si riscontrano degli inconvenienti si fermi immediatamente l'unità e si prenda contatto con il rivenditore.

- **Lunghi periodi di disuso**
  Quando non si utilizza l'unità per lunghi periodi si stacchi la corrente.

- **Collegamento con le periferiche**
  Quando si collegano periferiche si utilizzino dolo parti ed apparecchi designati dalla Sharp Corporation. L'uso di parti ed apparecchi non consono (ovvero modificati) può causare incovenienti anche di seria natura.

- **Macchie**
  Togliere le eventuali macchie dall'unità usando un panno morbido leggermente imbevuto d'acqua. Non si usino, per motivo alcuno, liquidi volatili, come benzina, triellina, ecc, per pulire l'unità.

- **Noie**
  Quando si usa l'unità in luoghi dove si riscontrano livelli di rumore elettrico piuttosto alti nella linea AC, si usi un filtro di linea per eliminare il fastidio. Si tenga sempre il cavo di segnale lontano dal cavo d'alimentazione o da altre apparecchiature elettriche.

- **Uso e non uso**
  Non si utilizzi e non si riponga l'unità con il coperchio superiore aperto altrimenti si possono avere danni.

- **Interferenza d'onde radio**
  L'uso di apparecchi radio o TV vicino all'MZ-80A può causare interferenze con la ricezione. Apparecchi che provocano forti campi manetici interferiscono con il funzionamento dell'MZ-80A.
  Tali apparecchi debbono essere tenuti almeno a due o tre metri dall'unità.

- **Interruttore d'alimentazione**
  Dopo aver spento l'unità si attenda almeno dieci secondi prima di riaccenderla. Ciò assicura il corretto funzionamento dei microprocessori. Non si inserisca, per motivo lacuno, la spina nella presa con l'interruttore di alimentazione messo sulla posizione ON.

- **Manutenzione della piastra magnetica a cassette**
  Le testine di registrazione e ci riproduzione del registratore possono riprodurre i dati non correttamente. Si puliscano le testine una volta al mese con un detergente di commercio.

- **Scolorimento dello schermo CRT**
  Se alcuni punti dello schermo rimangono accesi per lunghissimi periodi di tempo essi possono scolorirsi (se risulta necessario per certuni punti rimanere accesi per lunghi periodi di tempo si riduca la luminosità tramite il comando dell'unità di controllo del display).