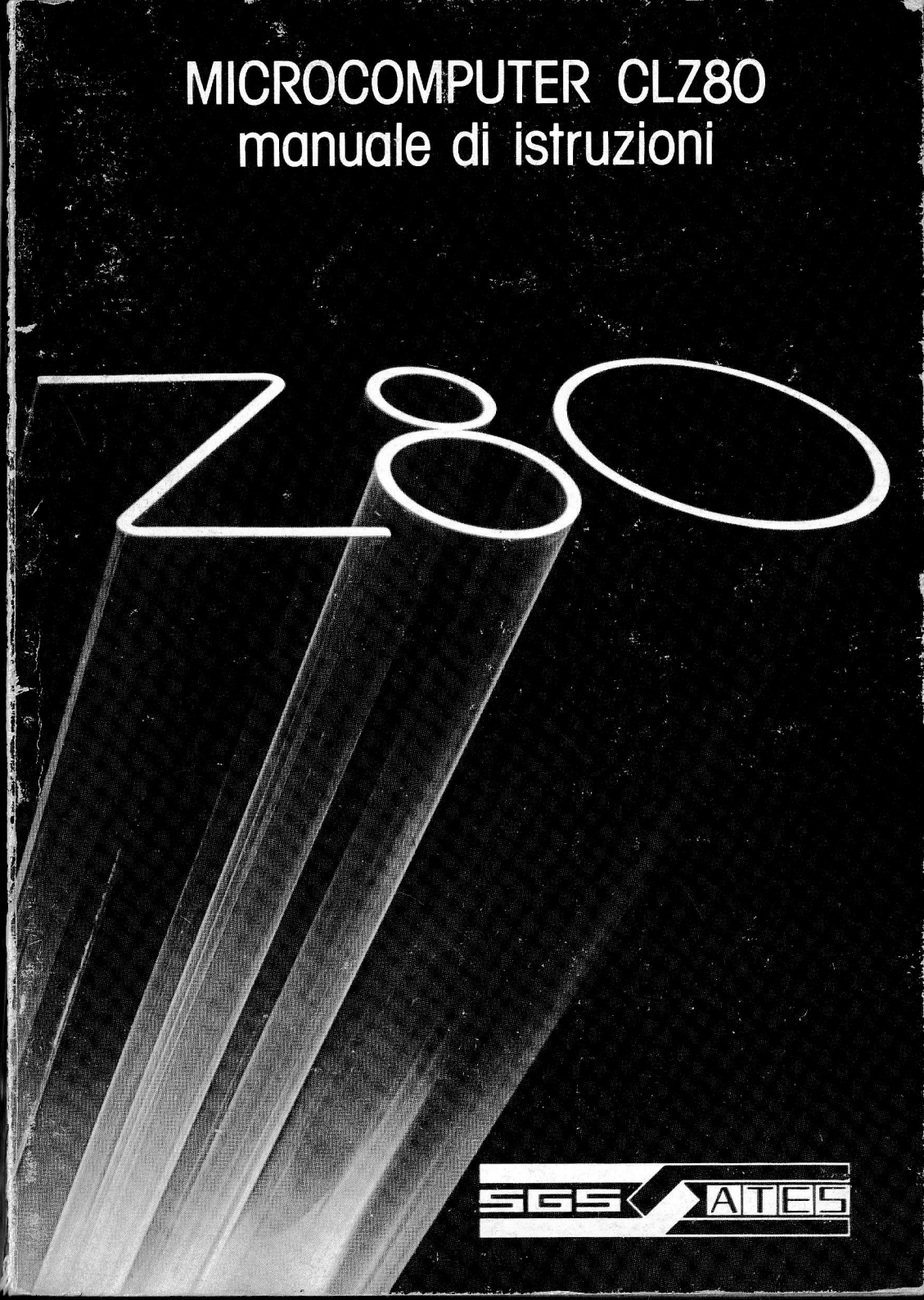
MICROCOMPUTER CLZ80
manuale di istruzioni



SGS ATES

MICROCOMPUTER CLZ80
manuale di istruzioni

Sommario:

Cap. 1 Hardware del CLZ80

Cap. 3 Assembler

Cap. 4 Monitor - Debug - Loader (M0-Z)

Cap. 5 Editor

Cap. 6 Binary Dump

Cap. 7 Input/Output Ports

N. B. Il capitolo 2 di questo libro, relativo al set di istruzioni del microprocessore Z80™(sul quale é basato il microcalcolatore CLZ80), é stato stampato in un volume separato, dal titolo "Z80 CPU INSTRUCTION SET".

Cap. 1-1 - HARDWARE DEL CLZ80

HARDWARE DEL CLZ80

INDICE

			PAG.
1.1	Introduzione		1.1-1÷3
1.2	Schema a blocchi		1.2-1÷3
1.3	CPU		1.3-1÷2
	1.3.1	Architettura CPU	1.3-2÷2
	1.3.2	Registri CPU	1.3-2÷11
1.4	Bus-Espansione		1.4-1
	1.4.1	Segnali Bus-Espansione	1.4-1÷13
	1.4.2	Caratteristiche elettriche Bus-Espansione	1.4-13÷23
1.5	Memoria non volatile - Rom - Prom - Eprom		1.5-1÷4
1.6	Memoria R.A.M.		1.6-1÷4
1.7	Interfaccia Unità Seriale		1.7-1÷4
1.8	Interfaccia Unità Magnetica		1.8-1÷4
1.9	Selettore Periferiche I/O		1.9-1÷4
1.10	Porte I/O		1.10-1÷2
1.11	Caratteristiche funzionali Bus-Espansione		1.11-1÷1
	1.11.1	Temporizzazione	1.11-1÷4
	1.11.2	Ricerca codice operativo	1.11-5÷9
	1.11.3	Scrittura e lettura dati in memoria	1.11-10÷13
	1.11.4	Scrittura e lettura dati nella periferica	1.11-14÷17
	1.11.5	Richiesta e riconoscimento Bus-Espansione	1.11-18÷20
	1.11.6	Richiesta e riconoscimento interruzione	1.11-21÷24
	1.11.7	Richiesta e riconoscimento interruzione non mascherabile	1.11-25÷26
	1.11.8	Istruzione Halt	1.11-27÷28
1.12	Composizione Modulo CLZ80		1.12-1÷3

1.1 INTRODUZIONE

Il CLZ80 è un microcalcolatore contenuto su di un unico modulo completo di unità centrale (CPU), di memoria (ROM e RAM), di interfaccia verso unità periferiche e di bus di espansione per l'eventuale aggiunta di altre memorie e interfaccie di periferiche.

Il microcalcolatore CLZ80 è disponibile in due versioni : CLZ80-4 e CLZ80-16 che differiscono per la capacità della memoria ad accesso casuale disponibile; il primo dispone di 4096 "bytes" ed il secondo di 16384 "bytes".

L'uso del microcalcolatore CLZ80 permette la gestione di processi industriali,il controllo di automatismi , il processamento di dati contabili e l'elaborazione di dati aritmetici e logici presenti in qualsiasi controllore elettronico.

Le prestazioni fondamentali del CLZ80 sono le seguenti:

. disponibilità di un processore a basso prezzo con la possibilità di integrarlo in qualsiasi sistema di calcolo piccolo o medio;

. Indirizzamento diretto (multimodo) di 64K bytes (K=1024) a 8 bit;

. elaborazione efficiente di caratteri a 8 bit senza la necessità di rotazioni, scambi o mascherature;

- operazioni asincrone (grazie allo stato di WAIT) che permettono a tutti i componenti del sistema di operare alla massima velocità possibile;
- catasta di memoria e registri indice che permettono di manipolare facilmente dati tabellari, sottoprogrammi ed interruzioni ;
- accesso diretto alla memoria per rendere accessibili ad alta velocità i dati a periferiche veloci;
- sedici registri a 8 bit di uso generale più due registri a 16 bit per indirizzare tabelle di dati, più 4 registri dedicati (PC, SP, I, R) di cui due a 16 bit;
- struttura di espansione a bus che permette di connettere le periferiche secondo una catena di priorità nei confronti delle risposte a richieste di interruzione;
- tre modi di risposta all'interruzione;
- un gruppo numeroso (158) e potente di istruzioni operanti su dati di 1,4,8 e 16 bit;
- partenza automatica dei programmi all'accensione, tramite due opzioni entrambi disponibili (partenza alla locazione Ø, oppure al programma di monitor M0-Z);

- dimensioni compatte e rispondenti alle norme europee relative ai moduli di circuito stampato;
- utilizzo di circuiti integrati LSI che concentrano notevoli prestazioni in dimensioni ridotte e in minime richieste di potenza di alimentazione (≃ 5 Watt).

1.2 SCHEMA A BLOCCHI

La fig. 1.2.1 illustra lo schema a blocchi relativo al micro-calcolatore **CLZ80** .

Sono evidenziati i nove blocchi principali che costituiscono il sistema e le linee di collegamento al loro interno.

Sono evidenziati i tre BUS che portano le principali informazioni relative alle interazioni fra il CPU ed i relativi blocchi.

Si notano:

- BUS-DATI che sopporta su 8 linee bidirezionali tutto il traffico di dati tra il CPU e la memoria o le periferiche.
- BUS-INDIRIZZI che controlla su 16 linee unidirezionali (con possibilità di aprirsi nel terzo-stato) l'indirizzamento della memoria comprendente il rinfresco di quelle dinamiche e l'indirizzamento delle periferiche.
- BUS-CONTROLLI che distribuisce i segnali dinamici di temporizzazione relativi alle fasi di scrittura e lettura nei confronti di memorie e periferiche; distribuisce inoltre i segnali necessari al controllo esterno dei BUS-DATI e BUS-INDIRIZZI ed i segnali relativi alle richieste di interruzione.

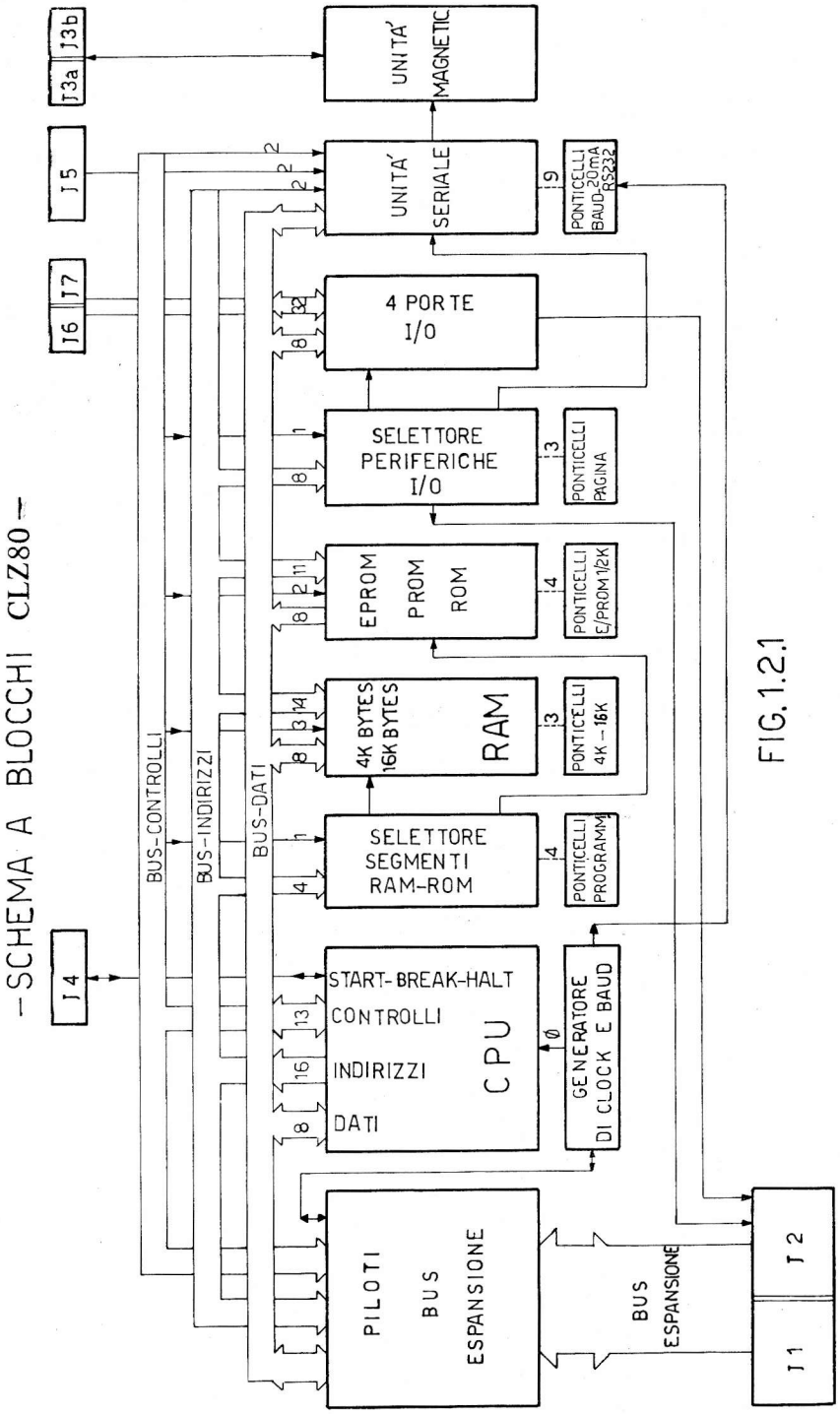


FIG. 1.2.1

. BUS-ESPANSIONE che permette di espandere le prestazioni del modulo **CLZ80** con l'aggiunta di moduli di memoria o di controllo di periferiche; per controllo di periferiche si intendono anche tutte quelle interfacce che interagiscono con un qualsiasi processo esterno.
Di questo BUS verranno evidenziate tutte le caratteristiche elettriche e funzionali in modo da permettere a qualsiasi utente di progettare e costruire il particolare tipo di interfaccia che è richiesto dal suo controllo.
Su questo BUS si connettono anche la serie di moduli facenti parte della famiglia del **CLZ80** e attuanti le funzioni più richieste (espansione di memoria, interfaccie di ingresso-uscita, controlli per FLOPPY-DISK, PROM-Programmer,etc.).

I connettori J1 e J2 sono accessibili al piede del modulo del **CLZ80**, mentre i connettori J3, J4, J5, J6, J7 sono disponibili sulla testa del modulo stesso.

1.3 CPU

Il microprocessore (CPU) utilizzato nel modulo **CLZ80** è il tipo Z80 prodotto dalla SGS-ATES con la tecnologia MOS-SILICON-GATE con impiantazione ionica e canale N.
Le caratteristiche e prestazioni salienti sono le seguenti:

. Disponibilità di 158 istruzioni, includenti tutte le 78 del microprocessore 8080A verso cui conserva una piena compatibilità in software. Alcune delle istruzioni aggiunte (rispetto all'insieme delle 78 del CPU 8080A) permettono di operare su notazioni binarie a 4, 8 e 16 bit ed inoltre di usufruire di modi nuovi di indirizzamento come quello indicizzato, quello relativo e quello operante su di un solo bit del dato (con questo ultimo modo è possibile indirizzare ed operare su qualsiasi bit della memoria o registro interno del CPU).
. 22 registri interni per un totale di 207 bits.
. 3 tipi di risposte alle richieste di interruzione sulla linea normale (INT), più una risposta non mascherabile su di una linea dedicata (NMI).
. Interfacciamento diretto e semplificato verso memorie statiche o dinamiche con tempo di accesso inferiore a 450 nsec.

. Ciclo di clock di 400 nsec; ciclo di istruzione minimo di 1.6 μsec.

. Controllo completo e trasparente del rinfresco delle memorie dinamiche (da 4K o 16K) senza perdita di tempo per attuarlo.

1.3.1 Architettura CPU

Una illustrazione dell'architettura interna del CPU Z80 è raffigurata dalla fig. 1.3.1.1 che ne pone in risalto i maggiori elementi; la figura farà da riferimento alla descrizione che segue .

1.3.2 Registri CPU (fig. 1.3.2.1)

Il CPU Z80 contiene 207 bit di memoria suddivisi fra 17 registri che sono interamente accessibili al programmatore.

La fig. 1.3.2.1 illustra come questa memoria è configurata in 17 registri a 8 bit, quattro registri a 16 bit ed un registro a 7 bit. Tutti i registri del CPU Z80 sono realizzati con memorie completamente statiche. I registri a 8 bit comprendono due gruppi di 6 registri che possono essere utilizzati individualmente o in coppia per formare registri a 16 bit; comprendono inoltre due accu

ARCHITETTURA CPU Z80

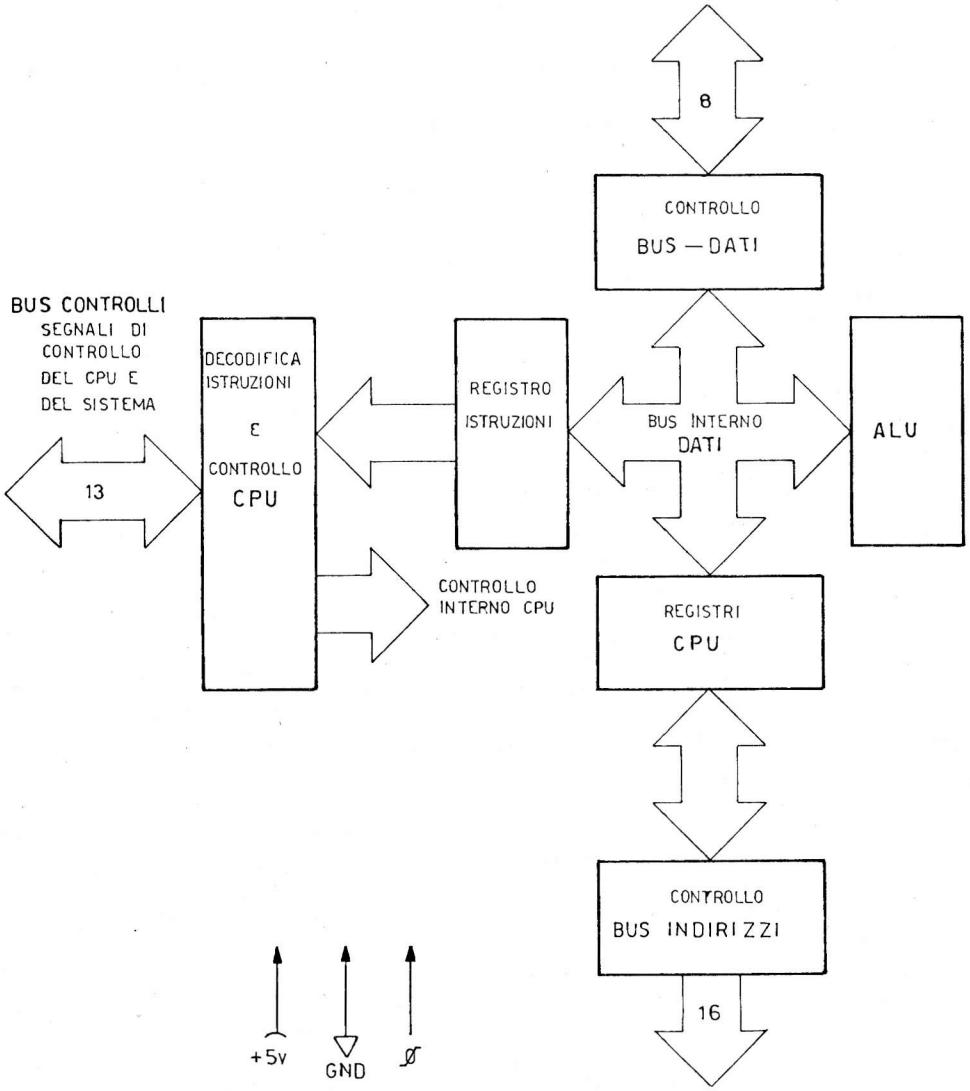


FIG. 1.3.1.1

REGISTRI CPU

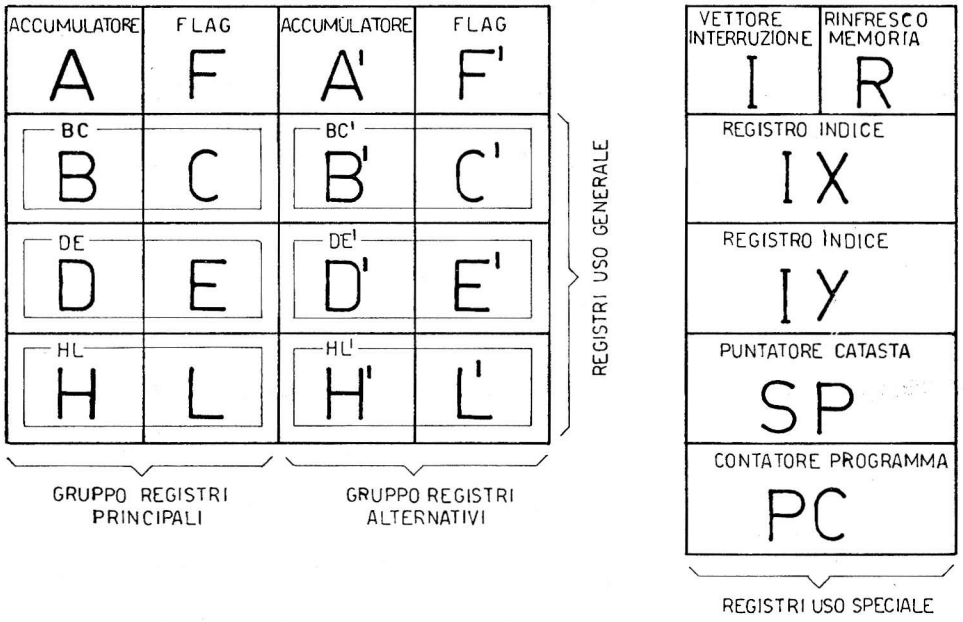


FIG. 1.3.2.1

mulatori e due registri di "flag".

* Registri Speciali

. Contatore del programma, PC

Il contatore del programma fornisce l'indirizzo a 16 bit dell'istruzione corrente da ricercarsi dalla memoria. Il valore del PC è automaticamente incrementato dopo che il suo contenuto è stato trasferito alle linee del BUS-INDIRIZZO. Quando nel programma viene incontrata una istruzione di salto il nuovo valore fornito al PC va a sostituirsi a quello precedente durante l'esecuzione dell'istruzione di salto annullando il valore incrementato.

. Puntatore della catasta, SP

Il puntatore della catasta fornisce l'indirizzo a 16 bit relativo alla sommità corrente della catasta che risiede in qualsiasi parte della memoria RAM del sistema. La memoria esterna riservata alla catasta si comporta in senso figurato come una catasta di oggetti in cui l'ultimo deposto rimane il primo ad essere disponibile per un prelievo. Dalla o nella catasta possono essere prelevati o deposti dati relativi a registri specifici del CPU mediante l'attuazione delle istruzioni POP e PUSH.

Il meccanismo proprio della catasta permette l'attivazione di interruzioni a livello multiplo, l'annidamento senza limiti dei richiami dei sottoprogrammi e la notevole semplificazione della manipolazione di certi tipi di dati tabellari .

. Registri indice IX, IY

I due registri IX e IY, completamente indipendenti fra di loro, contengono un indirizzo a 16 bit utilizzato nel modo di indirizzamento definito indicizzato.

Grazie a questo modo un registro indice viene usato come base per puntare ad una certa zona della memoria dove normalmente alloggiano dati di tabelle o stringhe.

Le istruzioni che operano con il modo di indirizzamento indicizzato contengono un byte addizionale che definisce la deviazione rispetto al puntamento definito dai registri IX e IY.

La deviazione è espressa come numero in complemento a 2 permettendo così di ottenere numeri positivi e negativi; questo ha il significato, per il programmatore, di poter accedere a dati che stanno prima o dopo dell'indirizzo base espresso dai registri indice IX e IY.

. Registro di interruzione, I

In risposta ad una interruzione il CPU Z80 può operare un richiamo indiretto a qualsiasi locazione di memoria.

Il registro I fornisce a questo scopo la parte più alta dell'in

dirizzo indiretto (gli 8 bit più significativi), mentre la periferica responsabile dell'interruzione fornisce la parte più bassa (gli 8 bit meno significativi).

Questa prestazione permette che i programmi serventi le interruzioni siano dinamicamente allocati in qualsiasi zona della memoria senza per questo sacrificare il tempo di accesso nei confronti degli stessi programmi di servizio.

Registro di rinfresco, R

Per permettere l'uso di memorie dinamiche, che richiedono una operazione di rinfresco, con la stessa semplicità di quelle statiche il CPU Z80 contiene un contatore di rinfresco definito anche come registro di rinfresco, R. Questo contatore a 8 bit viene automaticamente incrementato dopo ogni fase di ricerca dell'istruzione ed il suo contenuto viene distribuito sulla parte più bassa del BUS-INDIRIZZI assieme ad un segnale di presenza del rinfresco. Questa operazione di rinfresco esterna avviene contemporaneamente all'operazione interna del CPU relativa alla decodifica dell'istruzione appena letta dalla memoria; ciò significa che il rinfresco delle memorie dinamiche non

priva di nessun ciclo l'attività del CPU.

Inoltre questa operazione è completamente trasparente nei confronti del programma che non se ne deve minimamente occupare.

Solo per usi di controllo e manutenzione il programmatore può caricare il registro R che normalmente viene completamente ignorato.

* Registri accumulatori e di Flag

Per accumulatore si intende un registro sul quale possono venire eseguite operazioni aritmetiche, logiche e di scorrimento; per registro di flag si intende un registro che contiene indicazioni importanti relative alle operazioni aritmetiche o logiche in corso (esempio segno, polarità, annullamento, parità, etc.). Il CPU Z80 contiene due accumulatori a 8 bit indipendenti e due registri di flag a 8 bit ad essi associati. Il programmatore può scegliere la coppia di accumulatore - registro di flag con cui lavorare correntemente e scambiarla con l'altra coppia in qualsiasi momento con l'esecuzione di una semplice istruzione; questo permette in definitiva di lavorare con entrambi le coppie di accumulatore - flag disponibili.

* Registri di uso generale

Il CPU-Z80 rende disponibile due gruppi di registri di uso generale; ogni gruppo contiene sei registri a 8 bit che possono essere indifferentemente usati singolarmente come registri a 8 bit oppure in coppia come registri a 16 bit.

Il primo gruppo è composto dai registri B, C, D, E, H e L che in coppia diventano BC, DE, HL; il secondo gruppo è composto dai registri B', C', D', E', H', e L' che in coppia diventano BC', DE', HL'.

In qualsiasi momento è data facoltà al programmatore di scegliersi il gruppo desiderato attraverso una semplice istruzione di scambio che si riferisce a tutto il gruppo.

Un tipico esempio di impiego dei due gruppi è dato nei sistemi in cui viene richiesta una veloce risposta ad una interruzione; in questo caso un gruppo di registri e una coppia di accumulatore - flag viene completamente riservata per la manipolazione dei programmi di interruzione; l'altro gruppo con l'altra coppia di accumulatore - flag rimane sempre a disposizione del programma principale. Al passaggio del programma principale al programma di servizio dell'interruzione si attua un semplice scambio di registri con un notevole risparmio di tempo nei confronti del modo consueto di salvare e richiamare il contenuto

dei registri nella e dalla memoria.

In generale questi registri sono usati dal programmatore in un grande campo di applicazioni per minimizzare il tempo di esecuzione del programma e la sua occupazione di memoria; essi inoltre semplificano la programmazione di piccolissimi sistemi aventi solamente delle memorie ROM e privi (quasi o completamente) di memorie RAM.

Unità aritmetica e logica, ALU

L'unità aritmetica - logica esegue tutte le elaborazioni di genere aritmetico e logico che è in grado di fornire il CPU.
L'ALU comunica, all'interno del CPU, con i registri ed il controllo del BUS-DATI attraverso il BUS-INTERNO-DATI.
Le prestazioni aritmetico-logiche dell'ALU sono le seguenti:

- Somma
- Sottrazione
- AND logico
- OR logico
- EX-OR logico
- Paragone
- Rotazione o scorrimento (a destra e a sinistra), del tipo logico e del tipo aritmetico
- Incremento
- Decremento

- Attivazione del singolo bit
- Azzeramento del singolo bit
- Controllo del bit

Dalla potenza dell'ALU dipendono direttamente le capacità elaborative dell'intero CPU ed indirettamente del microcalcolatore CLZ80.

Registro d'Istruzione e controllo del CPU

Un registro invisibile al programmatore ma di importanza vitale per l'attività del CPU è il registro d'istruzione e controllo; esso immagazzina il codice operativo dell'istruzione letta della memoria e attraverso la sua decodifica controlla il traffico interno ed esterno al CPU dei dati in modo di attuare la fedele esecuzione dell'istruzione stessa; genera i controlli relativi alla lettura o scrittura dei registri interni, i controlli relativi all'abilitazione dell'unità aritmetico-logica e le temporizzazioni opportune dei segnali di interfaccia con l'esterno (memorie ed unità periferiche).

1.4 BUS - ESPANSIONE

Il BUS-ESPANSIONE disponibile sui connettori J1- 2 permette al sistema di espandersi con l'aggiunta di altri moduli di memoria o di controllo di periferiche; su questo BUS-ESPANSIONE l'utente può connettere le proprie interfacce e controlli seguendo e rispettando le regole funzionali ed elettriche che verranno date nel seguito del capitolo.

1.4.1 Segnali Bus-Espansione

Di seguito vengono elencati tutti i segnali relativi al BUS-ESPANSIONE con una loro breve descrizione funzionale oltre alle caratteristiche elettriche; questo elenco deve rappresentare un riferimento continuo alle descrizioni che seguiranno.

BADØ - BAD15 INDIRIZZI

Uscite con terzo stato, attive alte, BADØ corrisponde al bit meno significativo.

Questo gruppo di segnali rappresenta le informazioni di indirizzo relative a:

- memoria RAM: in questo caso tutte le linee sono utilizzate con una capacità di indirizzamento di 64K byte (K = 1024);

- periferiche; in questo caso sono utilizzate solo le prime otto linee (BADØ ÷ BAD7) con una capacità di indirizzamento di 256 periferiche.

  Quando le linee di indirizzo sono utilizzate per indirizzare una periferica, le ultime otto (BAD8÷BAD15) portano l'informazione dell'accumulatore; questa informazione può essere utilizzata o per estendere oltre 256 la possibilità di indirizzamento delle periferiche oppure per elaborare esternamente i dati dell'accumulatore.

- indirizzo di rinfresco ; in questo caso le prime sette linee, BADØ ÷ BAD6 portano l'informazione relativa al contatore di rinfresco, registro R, per pilotare le memorie dinamiche.

BDØ ÷ BD 7 DATI

Ingressi - uscite con terzo stato, attive alte, BDØ corrisponde al bit meno significativo .

Queste otto linee rappresentano il bus bidirezionale per i dati della memoria e delle periferiche.

Tutto il traffico dei dati relativo all'attività del sistema è concentrato su queste linee.

$\overline{BM1}$ CICLO 1 DI MACCHINA

Uscita con terzo stato, attiva bassa.

Il segnale BM1 indica che il ciclo corrente del CPU corrisponde alla ricerca dalla memoria del codice operativo dell'istruzione; seguiranno altri cicli relativi all'esecuzione dell'istruzione durante i quali questo segnale non sarà attivato.

$\overline{BMREQ}$ RICHIESTA DI MEMORIA

Uscita con terzo stato, attiva bassa.

L'attivazione di questo segnale indica che è in corso una richiesta d'intervento della memoria per leggervi o scrivervi un dato; durante l'attivazione di questo segnale il bus degli indirizzi porta l'indirizzo della locazione di memoria oggetto delle operazioni di scrittura o lettura.

$\overline{BIORQ}$ RICHIESTA DI INGRESSO-USCITA

Uscita con terzo stato, attiva bassa.

L'attivazione di questo segnale indica che è in corso una richiesta d'intervento della periferica per una azione di scrittura o lettura di un dato; durante l'attivazione

di questo segnale il bus degli indirizzi (BAD$\emptyset$ ÷ BAD7) porta l'indirizzo relativo alla periferica interessata.

Il segnale $\overline{BIORQ}$ esegue un'altra importante funzione relativa al riconoscimento di una interruzione da parte del CPU; questa funzione è attivata dalla contemporanea presenza dei due segnali $\overline{BM1}$ e $\overline{BIORQ}$ che non avviene MAI nelle altre condizioni operative; questa coincidenza comunica alla periferica interrompente che è stata riconosciuta la sua azione di interruzione e che essa può di conseguenza forzare sul BUS-DATI il proprio vettore di interruzione.

$\overline{BRD}$ LETTURA DATI

Uscita con terzo stato, attiva bassa.

Questo segnale indica che il CPU vuole leggere un dato o dalla memoria o dalla periferica; quest'ultime dovrebbero fornire il dato forzando il BUS-DATI in coincidenza con l'attivazione del segnale $\overline{BRD}$.

$\overline{BWR}$ SCRITTURA DATI

Uscita con terzo stato, attiva bassa. Questo segnale indica che il CPU vuole scrivere un dato o nella memo-

ria o nella periferica; durante l'attivazione di questo segnale il BUS-DATI porta il dato relativo all'operazione di scrittura e fornito dal CPU.

BRFSH RINFRESCO

Uscita con terzo stato, attiva bassa.

Il segnale BRFSH indica, durante la sua attuazione, che il BUS-INDIRIZZI porta sulle sette linee meno segnificative (BAD$\emptyset$ + BAD6) il contenuto del contatore di rinfresco R; durante la sua attuazione ed in concomitanza del corrente segnale BMREQ deve essere effettuato un accesso di rinfresco per tutte le memorie dinamiche esistenti nel sistema.

Le altre linee di indirizzo (BAD8 + BAD15) non interessate al rinfresco durante l'attuazione di BRFSH portano il contenuto del registro I.

BHALT HALT

Uscita attiva bassa.

Il segnale BHALT indica che il CPU ha eseguito una istruzione di HALT e si è posto in una situazione di stallo temporaneo con l'esecuzione continua di istruzioni NOP (nessuna operazione); durante questa situazione il

BUS-INDIRIZZI durante la fase BM1, punta sempre alla locazione seguente a quella relativa all'istruzione di HALT.

L'esecuzione ripetuta dell'istruzione NOP permette il normale rinfresco delle memorie dinamiche.

Solo tre azioni possono scuotere il CPU da questa situazione; esse sono:

- richiesta di interruzione non mascherabile (segnale NMI)
- richiesta di interruzione con la maschera abilitata
- richiesta di azzeramento (segnale RESET)

Dopo aver attuato le prime due il CPU, al ritorno dai programmi di servizio delle interruzioni, prosegue con l'esecuzione dell'istruzione seguente a quella di HALT.

BWAIT ATTESA

Ingresso, attivo basso.

Il segnale BWAIT fornito dalla memoria o dalla periferica informa il CPU che le stesse non sono ancora pronte per un trasferimento di dato; come conseguenza il CPU congela la propria attività sino a che rimane attivato il segnale di BWAIT.

Questo segnale permette che memorie o periferiche siano sincronizzate nell'attuazione dei trasferimenti dati con il CPU.

$\overline{\text{BINT}}$ INTERRUZIONE

Ingresso, attivo basso.

Il segnale di richiesta di interruzione è generato dalle periferiche che hanno terminato la loro azione funzionale e richiedono l'intervento del CPU.

La richiesta sarà soddisfatta dal CPU solo quando saranno contemporaneamente valide le seguenti condizioni:

- esecuzione completa dell'istruzione durante la quale é stata inviata la richiesta di interruzione;
- attivazione del FLIP-FLOP interno del CPU abilitante il processo di riconoscimento di una richiesta di interruzione; l'attivazione di questo FLIP-FLOP è sotto controllo del software;
- disattivazione (segnale non vero) del segnale $\overline{\text{BUSRQ}}$.

Il CPU comunica all'esterno l'accettazione di una interruzione tramite la contemporanea affermazione dei segnali $\overline{\text{M1}}$ e $\overline{\text{IORQ}}$ durante l'inizio dell'istruzione che segue il riconoscimento dell'interruzione.

Il CPU può rispondere ad una richiesta di interruzione in tre modi differenti che saranno illustrati nelle sezioni specifiche riguardanti il software.

$\overline{\text{BNMI}}$ INTERRUZIONE NON MASCHERABILE

Ingresso, attivo basso.

La linea di richiesta di interruzione non mascherabile ha una funzionalità simile a quella della richiesta normale di interruzione ($\overline{\text{BINT}}$) con le seguenti differenze:

- l'interruzione è sempre riconosciuta alla fine di una istruzione con priorità più alta rispetto ad una eventuale e coincidente richiesta normale di interruzione;
- l'interruzione è riconosciuta indipendentemente dall'attivazione del FLIP-FLOP di abilitazione dell'interruzione.

In pratica il segnale $\overline{\text{BNMI}}$ forza il CPU, al termine dell'istruzione corrente, a saltare alla locazione 0066H salvando contemporaneamente il contenuto del PC nella catasta esterna; questo permette di ritornare al programma interrotto dopo avere eseguito il programma di interruzione.

BRESET - AZZERAMENTO

Ingresso, attivo basso.

Il segnale BRESET forza l'inizializzazione del CPU che comprende:

- azzeramento del PC
- disattivazione del FLIP-FLOP abilitante l'interruzione
- Azzeramento del registro I
- Azzeramento del registro R

Durante il tempo in cui rimane attivato il segnale BRESET i controlli (uscite) vanno nello stato inattivo, cioè alto.

BBUSRQ - RICHIESTA BUS

Ingresso, attivo basso.

Il segnale BBUSRQ permette la richiesta del BUS-ESPANSIONE da parte di una periferica che ne vuole diventare il controllore.

Questo segnale permetterà ai piloti del modulo CLZ80 del BUS-ESPANSIONE di andare nel terzo stato alla fine del corrente ciclo di macchina lasciando le linee corrispondenti completamente disponibili ad un altro controllore connesso alle stesse.

BBUSAK - RICONOSCIMENTO BUS

Uscita, attiva bassa.

Questa uscita indica, quando attiva, che la richiesta del BUS-ESPANSIONE da parte della periferica (attuata tramite l'attivazione del segnale BBUSRQ) è stata soddisfatta e che di conseguenza le linee del BUS-ESPANSIONE sono completamente disponibili per l'uso appropriato della periferica.

In particolare le linee disponibili sono:

- le linee degli INDIRIZZI : BAD$\emptyset$ ÷ 15
- le linee dei DATI : BD$\emptyset$ ÷ 7
- le linee dei CONTROLLI: BM1; BMREQ; BIORQ; BRD; BWR; BRFSH

Rimangono attive le linee di servizio quali: BHALT, B$\emptyset$, BFCU, IOQ3, IOE$\emptyset$ ÷ 3, IOQ$\emptyset$ ÷ 3, IEI, IEO, DEI, DEO e quelle del tipo a collettore-aperto quali: BWAIT, BINT, BNMI, BBUSRQ, BRESET.

Rimane naturalmente sempre sotto controllo del CPU la linea BBUSAK.

BØ CLOCK DI MACCHINA

Uscita, attiva alta.

Questo segnale rappresenta il CLOCK di macchina del modulo CLZ80 ed è responsabile direttamente o indirettamente della sequenzialità di tutte le operazioni e delle loro temporizzazioni.

BFCU CLOCK DI CONVERSIONE

Uscita, attiva alta.

Questa segnale di servizio permette di sincronizzare eventuali convertitori di tensione (DC - DC) presenti sul BUS-ESPANSIONE del CLZ80. La frequenza di questo segnale è 1/8 quella del segnale BØ.

$\overline{IOQ3}$ DECODIFICA PERIFERICA

$\overline{IOEØ÷3}$

$\overline{IOUØ÷3}$ Uscite, attive basse.

Questi segnali rappresentano la decodifica di alcune linee di indirizzi di periferiche in modo da facilitare l'utente del modulo CLZ80 nella progettazione di interfaccia e controlli particolari.

Nella sezione riguardante l'interfacciamento verrà esemplificato l'uso di questi segnali.

IEI,IEO - ABILITAZIONE INTERRUZIONE

IEI: ingresso, attivo alto

IEO: uscita, attiva alta

Questi due segnali rappresentano l'ingresso e l'uscita della catena della priorità relativa all'interruzione.

L'uso di questi segnali, in congiunzione del modo 2 di interruzione, permette la creazione di una catena di priorità nei confronti delle richieste di interruzione fatte da più periferiche alloggiate sul BUS-ESPANSIONE del modulo CLZ80.

Si può dire che il modulo della periferica più prossimo fisicamente al modulo del CLZ80 avrà una priorità più alta rispetto ad un altro modulo di periferica più lontano; questo permette di coordinare e gestire con ordine le varie richieste di interruzione che possono essere generate, anche contemporaneamente, da più periferiche.

DEI,DEO - ABILITAZIONE BUS

DEI: ingresso, attivo alto

DEO: ingresso, attivo alto.

Questi due segnali rappresentano l'ingresso e l'uscita della catena della priorità relativa alla richiesta del

BUS-ESPANSIONE.

Funzionalmente questi due segnali si comportano in modo simile a quelli relativi all'abilitazione dell'interruzione IEI e IEO; la loro funzione quindi è quella di permettere la gestione ordinata delle richieste di BUS-ESPANSIONE che vengono contemporaneamente da più periferiche.

1.4.2 Caratteristiche Elettriche Bus Espansione

Vengono date in questo paragrafo tutte le informazioni relative alle caratteristiche ed ai parametri elettrici del BUS-ESPANSIONE . La figura 1.4.2.1 a), b) illustra la parte del modulo CLZ80 relativa al pilotaggio e controllo del BUS-ESPANSIONE con indicati i dispositivi integrati utilizzati.

Si possono dividere in cinque le categorie che raggruppano i tipi di connessioni elettriche al BUS-ESPANSIONE individuati in figura 1.4.2.1 a) e b) da numeri racchiusi in circoletti.

La categoria (1) raggruppa le linee del BUS-ESPANSIONE che tipicamente sono forzate da piloti a tre stati che possono assorbire verso massa una corrente maggiore

di 24mA con una tensione inferiore a 0,5V e possono fornire dall'alimentazione (+ 5V) una corrente di 2,6mA con una tensione maggiore di 2,4V ; tipico pilota di queste linee è il dispositivo 74LS367.

Il forzamento di questi piloti nel terzo dato è opera dall'affermazione del segnale $\overline{\text{BUSAK}}$ (nel modulo CLZ80).

E' necessario porre attenzione affinchè le linee (1) non vengano forzate da più piloti contemporaneamente.

La categoria (2) raggruppa le linee del BUS-ESPANSIONE (tipicamente le linee dei dati) che sono bidirezionali; i piloti di queste linee sono dello stesso tipo elettrico delle linee (1), mentre i ricevitori connessi a queste linee sono normali ingressi di dispositivi della serie 74LS aventi una corrente fornita di 0,36 mA per segnali bassi (inferiori a 0,4 V) ed una corrente assorbita di 20 μA per segnali alti (maggiori di 2,7 V).

SCHEMA DELLA CONNESSIONE DEL CLZ80 AL BUS-ESPANSIONE

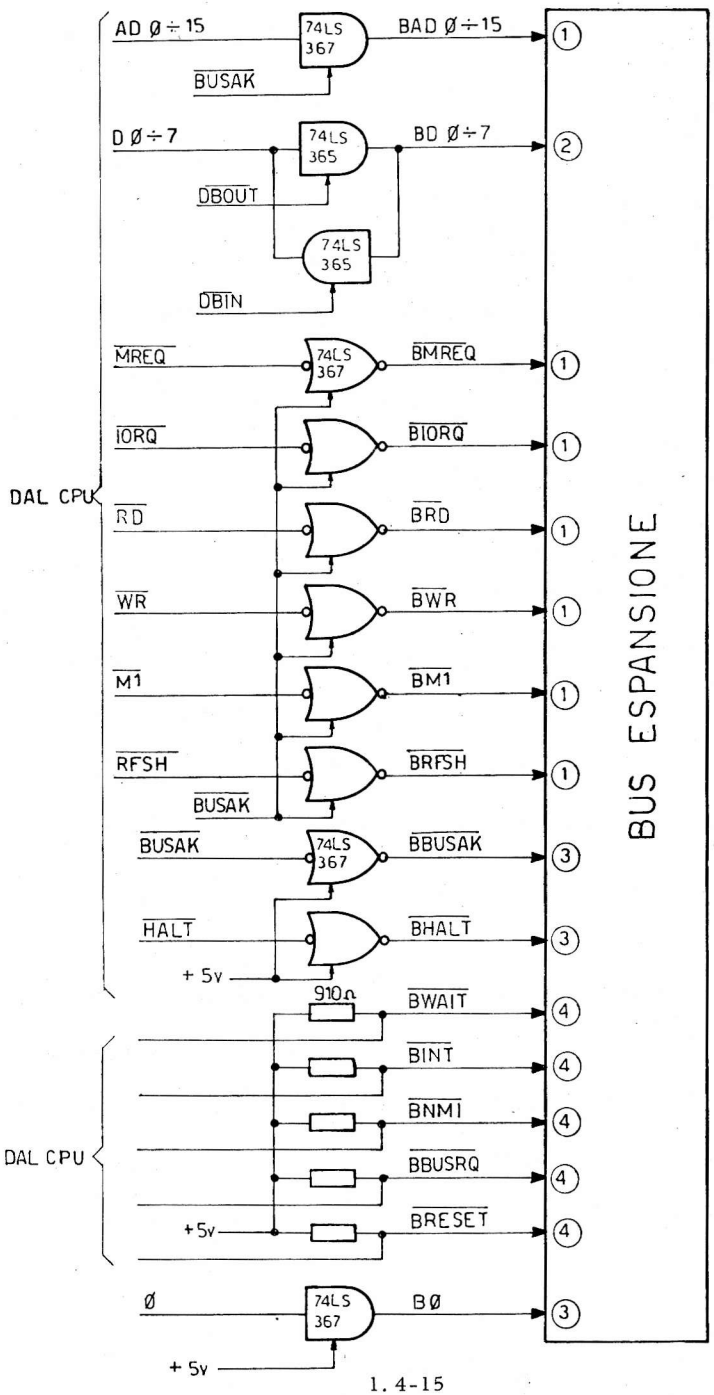


FIG. 1.4.2.1 a

Possono essere vantaggiosamente utilizzati come ricevitori anche dispositivi aventi una più alta impedenza di ingresso (tipicamente dispositivi MOS o ricevitori di linea) e una eventuale isteresi di ingresso.
Nel caso specifico del modulo CLZ80 sono utilizzati come ricevitori e piloti gli stessi dispositivi, 74LS365, che vengono opportunamente controllati dai segnali interni al modulo, $\overline{DBIN}$ e $\overline{DBOUT}$, in modo da evitare che entrami siano attivi.

La categoria (3) raggruppa le linee che vengono pilotate unicamente dal modulo CLZ80 e non da altri moduli presenti sul BUS-ESPANSIONE che rimangono solo utilizzatori di queste linee. I piloti usati sono uguali a quelli della categoria (1) con la differenza che non andranno mai nello stato di alta impedenza.

La categoria (4) raggruppa le linee che possono essere forzate al livello basso da qualsiasi modulo connesso al BUS-ESPANSIONE; sono conosciute con il termine di linee da pilotarsi con collettore - aperto poichè i forzatori che vi possono accedere sono proprio di questo ge-

nere; un tipico dispositivo da utilizzare a questo scopo è il 74LS38; in ogni caso qualsiasi dispositivo a collettore aperto che possa assorbire più di 8mA con tensione inferiore a 0,5 V è adatto a questo uso.

Ognuna di queste linee rappresenta una funzione logica di OR-CABLATO i cui ingressi sono sparsi su qualsiasi modulo connesso al BUS-ESTENSIONE.

Le resistenze di terminazione sono raggruppate sul modulo CLZ80 come illustrato in fig. 1.4.2.1. **a)**.

La categoria (5) raggruppa sia le linee di servizio sia le linee della gestione della priorità di interruzione e di accesso al BUS-ESTENSIONE.

Queste linee portano segnali generati dal modulo CLZ80 e utilizzabili dagli altri moduli; i livelli usati sono corrispondenti a quelli della serie di circuiti integrati 74LS. Ogni linea può quindi pilotare sino a dieci ricevitori della stessa serie. Si raccomanda in ogni caso di non oltrepassare mai questo carico usando eventualmente ricevitori ad alta impedenza d'ingresso.

SCHEMA DELLA CONNESSIONE DEL CLZ80 AL BUS ESPANSIONE

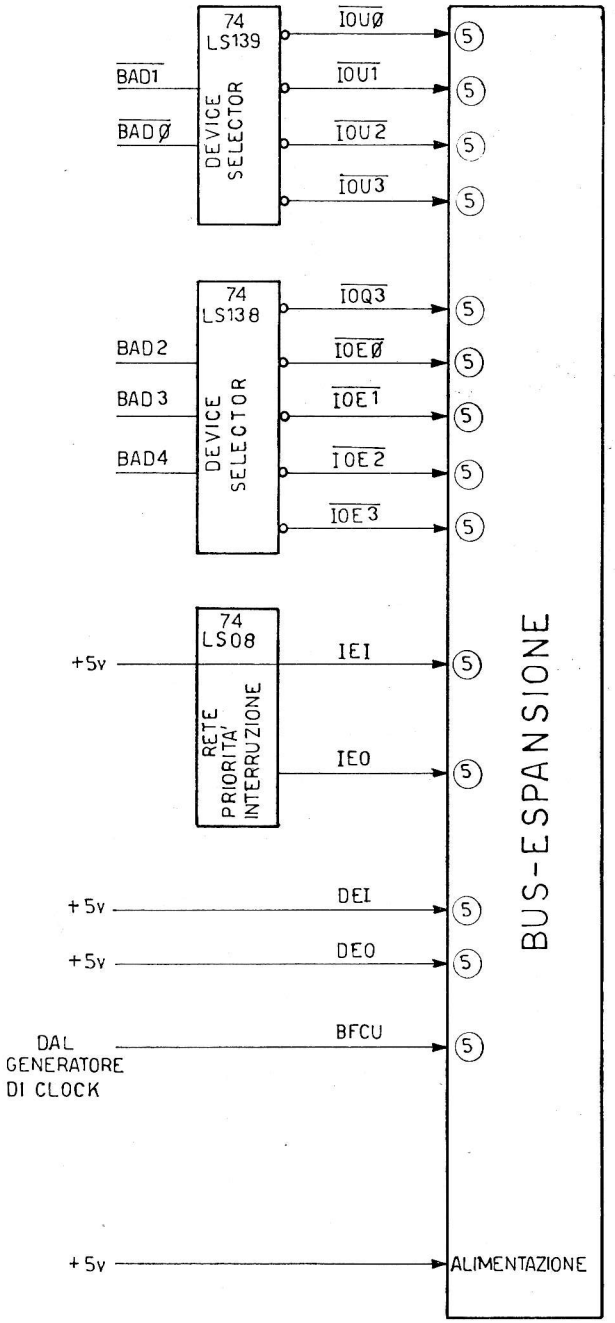


FIG.1.4.2.1b

CONNETTORI J1-J2 DEL BUS-ESPANSIONE

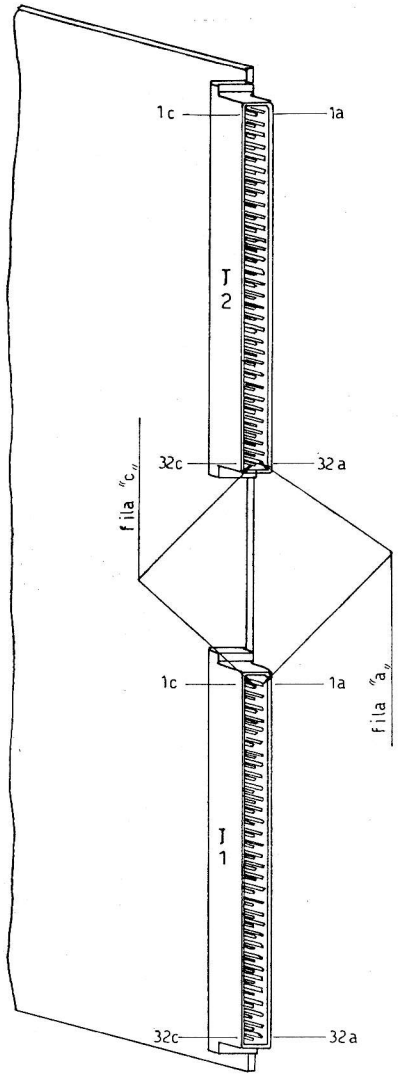


FIG.1.4.2.2.

CONNETTORI J4-J5-J6-J7 SULLA TESTA DEL MODULO CLZ80

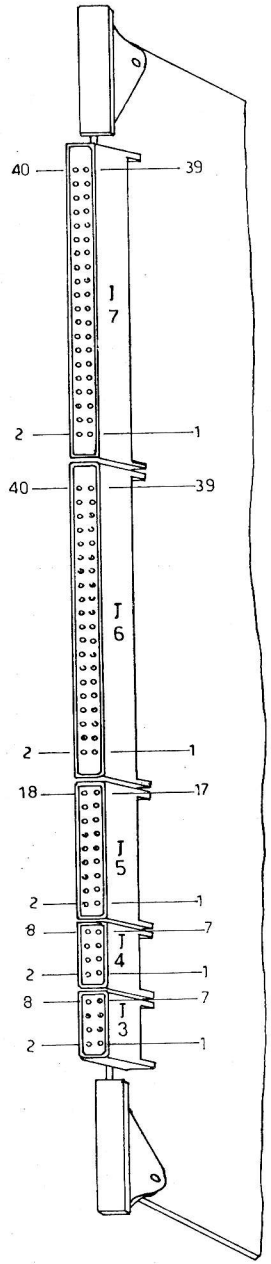


FIG. 1.4.2.3.

La tabella 1.4.2.1 sintetizza tutte le informazioni relative alle linee del BUS-ESPANSIONE indicando:

a) nome della linea

b) funzione del segnale sulla linea

c) categoria elettrica di appartenza

d) piedino occupato sui connettori J1 o J2.

La figura 1.4.2.2 illustra la posizione fisica dei connettori J1 e J2 del BUS-ESPANSIONE con evidenziati i pin di connessione.

TABELLA 1.4.2.1

Linea	Funzione	Cat.	Pin
+5V	Alimentazione 5V ± 2%	N.A.	J1-1ac/J2-1ac
BMREQ	Richiesta di memoria	1	J2-22c
BIORQ	Richiesta di periferica	1	J2-21c
BRD	Lettura da memoria o periferica	1	J2-13c
BWR	Scrittura in memoria o periferica	1	J2-14c
BM1	Ciclo di macchina 1	1	J1-15c
BRFSH	Ciclo di rinfresco	1	J1-11c
BBUSAK	Riconoscimento di rich. di Bus	3	J1-20c
BBUSRQ	Richiesta del BUS-ESPANSIONE	4	J2-25c
BHALT	Ciclo di HALT	3	J1-18c
BWAIT	Attesa per memorie o periferica	4	J2-26c
BINT	Richiesta interruzione	4	J2-24c
BNMI	Richieste non mascherabili di int.	4	J2-23c
BRESET	Inizializzazione CPU	4	J1-28c
BØ	Fase del sistema	3	J1-17c
BFCU	Fase covertitore DC-DC	5	J1-8c
IOUØ	Decodifica di indirizzo relativa alle linee: BADØ, BAD1	5	J1-24c
IOU1		5	J1-23c
IOU2		5	J1-22c
IOU3		5	J1-21c
IOQ3	Decodifica parziale (3,4,5,6,7) degli indirizzi relativa alle linee: BAD2, BAD3, BAD4	5	J2-5c
IOEØ		5	J2-6c
IOE1		5	J2-8c
IOE2		5	J2-9c
IOE3		5	J2-11c

TABELLA 1.4.2.1 (seguito)

Linea	Funzione	Cat.	Pin
+ 12V		N.A.	J1-16ac/J2-16ac
- 12V			J2-3ac
- 5 V			J2-4ac
GND	Ritorno relativo alle alimentazioni	N.A.	J1-32ac/J2-32ac
BDØ	Bit meno significativo	2	J1-27c
BD1		2	J1-26c
BD2		2	J1-25c
BD3		2	J1-29c
BD4	Linee dei dati	2	J1-30c
BD5		2	J1-31c
BD6		2	J2-12c
BD7	Bit più significativo	2	J2-10c
BADØ	Bit meno significativo	1	J1-3c
BAD1		1	J1-7c
BAD2		1	J1-6c
BAD3		1	J1-2c
BAD4		1	J1-4c
BAD5		1	J1-5c
BAD6		1	J2-30c
BAD7		1	J2-29c
BAD8	Linee degli indirizzi	1	J2-31c
BAD9		1	J2-28c
BAD1Ø		1	J1-12c
BAD11		1	J2-27c
BAD12		1	J2-19c
BAD13		1	J2-7c
BAD14		1	J2-18c
BAD15	Bit più significativo	1	J2-17c

1.5 MEMORIA NON VOLATILE ROM-PROM-EPROM

La parte del modulo CLZ80 relativa alla memoria non volatile consente all'utente di disporre di quattro opzioni per quanto riguarda l'utilizzo di memorie del tipo ROM-PROM-EPROM.

La tabella 1.5.I illustra le quattro combinazioni possibili utilizzando quattro dispositivi diversi di cui due sono EPROM, uno è una ROM e uno è una PROM.

Qualunque sia il dispositivo scelto, esso viene alloggiato sui quattro zoccoli a 24 piedini presenti sullo stampato del CLZ80 per questo scopo.

Per ogni tipo di dispositivo è necessario predisporre un opportuno collegamento a mezzo di ponticelli che personalizzano l'uso di quel particolare dispositivo nei confronti del resto del modulo CLZ80.

La fig. 1.5.1 identifica topologicamente la posizione dei ponticelli interessati a questa operazione.

Si noti in particolare che i dispositivi da 2 K x 8 bit (2716 e 8316 E) vengono ad occupare nella progressione di sistemazione sul modulo CLZ80 una direzione diversa dai dispositivi da 1 K x 8 bit (2708 e 6381).

TABELLA 1.5.I

E-P-ROM Scelta	Ponticelli	Partizioni relative			
		Q49	Q50	Q51	Q52
2708	1 - 3 - 6	Ø÷1K	1K÷2K	2K÷3K	3K÷4K
6381	2 - 4 - 7	Ø÷1K	1K÷2K	2K÷3K	3K÷4K
2716	1 - 5 - 8	Ø ÷ 2K	4K÷6K	2K÷4K	6K÷8K
2316-E	1 - 5 - 8	Ø ÷ 2K	4K÷6K	2K÷4K	6K÷8K

TABELLA 1.5.II

E-P-ROM Confini partizioni 4K		Ponticelli			
		9 - 26	9 - 25	9 - 24	9 - 23
PONTICELLI	30 - 15	Ø ÷ 4K	4K÷8K	8K÷12K	12K÷16K
	30 - 16	16K÷20K	20K÷24K	24K÷28K	28K÷32K
	30 - 17	32K÷36K	36K÷40K	40K÷44K	44K÷48K
	30 - 18	48K÷52K	52K÷56K	56K÷60K	60K÷64K

TABELLA 1.5.III

E-P-ROM Confini partizioni 8K		Ponticelli			
		10 - 15	10 - 16	10 - 17	10 - 18
PONTICELLI	27 - 30	0 ÷ 8K	16K÷24K	32K÷40K	48K÷56K
	28 - 29	8K÷16K	24K÷32K	40K÷48K	56K÷64K

PONTICELLI RELATIVI ALLA MEMORIA NON VOLATILE

TABELLE 1.5.I–1.5.II–1.5.III

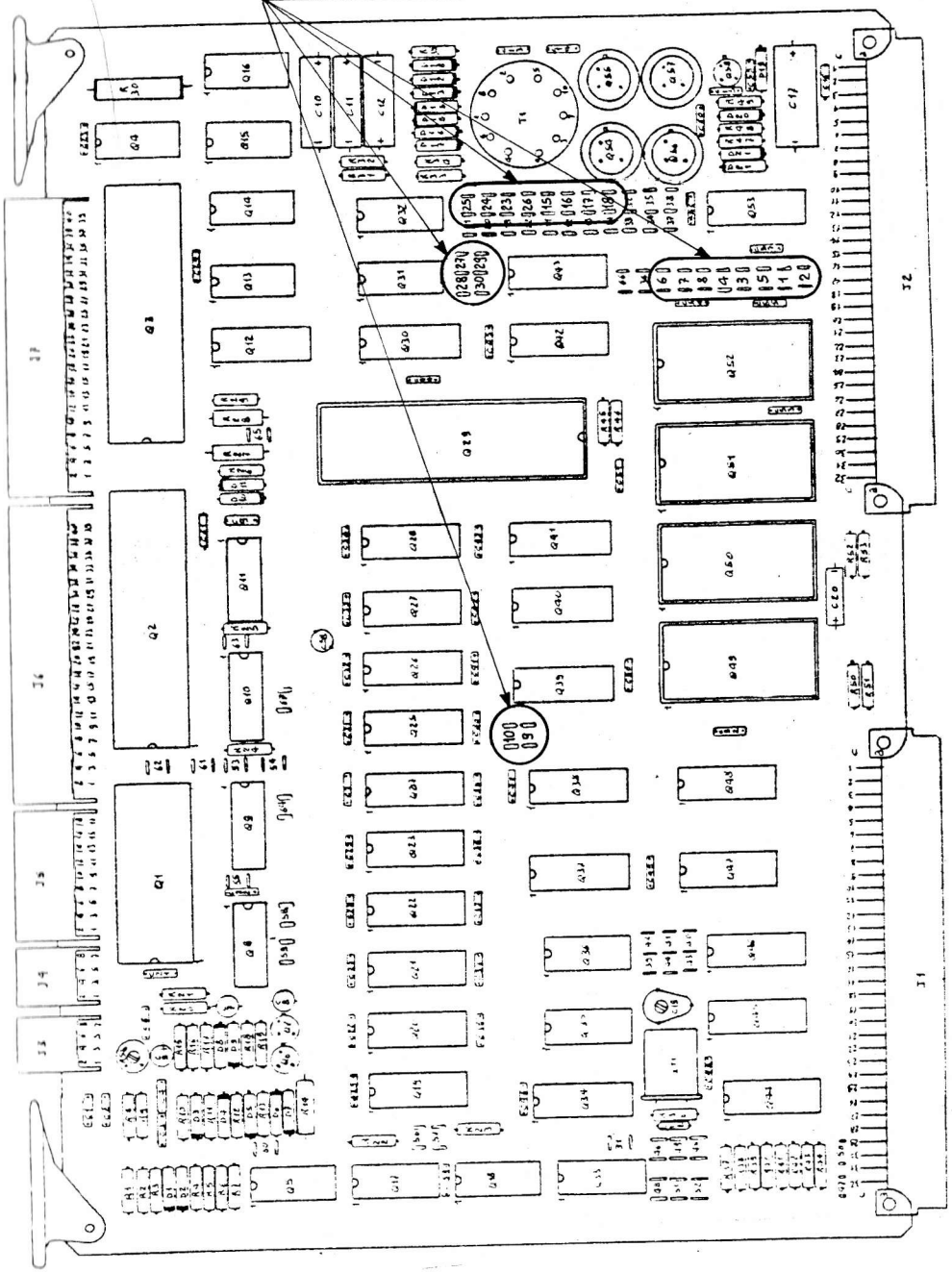


FIG. 1.5.1

Usando o i dispositivi da 1 K (2708 - 6381) o quelli da 2 K (2716 - 8316 E) si può disporre di partizioni di memorie o di 4 K byte o di 8 K byte rispettivamente; poichè il CPU può indirizzare sino a 64 K byte è necessario decidere in quale partizione alloggiare la memoria non volatile disponibile.

Per ottenere questo il modulo CLZ80 conserva all'utente di scegliere una partizione qualsiasi attraverso l'inserimento di opportuni ponticelli; le tabelle 1.5.II e 1.5.III danno le regole necessarie a questo scopo.

1.6 MEMORIA RAM

La parte del modulo CLZ80 relativa alla memoria di scrittura - lettura consente all'utente di disporre di due opzioni per quanto riguarda la capacità totale.

La prima opzione, utilizzante il dispositivo dinamico da 4 K bit 4027, rende disponibile 4 K byte di memoria RAM; la seconda utilizzante il dispositivo dinamico da 16 K bit 4016, rende disponibile invece 16 K byte.

La tabella 1.6.I illustra le due combinazioni con riguardo alla opportuna scelta di ponticelli che le caratterizzano.

Le tabelle 1.6.II e 1.6.III illustrano le disposizioni di ponticelli necessarie ad alloggiare opportunamente, nel campo di 64 K byte di memoria indirizzabile dal CPU, le partizioni di RAM da 4 K byte o da 16 K byte.

La figura 1.6.1 identifica topologicamente la posizione dei ponticelli interessati a queste operazioni.

Il rinfresco delle memorie dinamiche è attuato in modo automatico e completamente trasparente al programmatore; si deve solo porre attenzione al fatto che quando il CPU rilascia il

controllo del BUS-ESTENSIONE il rinfresco automatico cessa e l'attuazione, se necessaria, deve essere opera della periferica che ha guadagnato il BUS-ESTENSIONE.

TABELLA 1.6.I

SCELTA RAM		PONTICELLI	DIMENSIONE
DISPOS.	4027	39 - 40 -41	4 K x 8
	4116	42 - 43 -44	16 K x 8

TABELLA 1.6.II

RAM Confini 4K Partizioni		PONTICELLI			
		22	21	20	19
PONTICELLI	11	Ø ÷ 4K	4K ÷ 8K	8K ÷12K	12K÷16K
	12	16K ÷20K	20 K÷24K	24K ÷28K	28K÷32K
	13	32K ÷36K	36 K÷40K	40K ÷44K	44K÷48K
	14	48 K ÷52K	52 K÷56K	56K ÷60K	60K÷64K

TABELLA 1.6.III

RAM 16 K	PONTICELLI			
	31-11	31-12	31-13	31-14
Confini Partizioni	Ø -16K	16K ÷32K	32K ÷48K	48K÷64K

PONTICELLI RELATIVI ALLA MEMORIA RAM

TABELLE 1.6.I–1.6.II–1.6.III

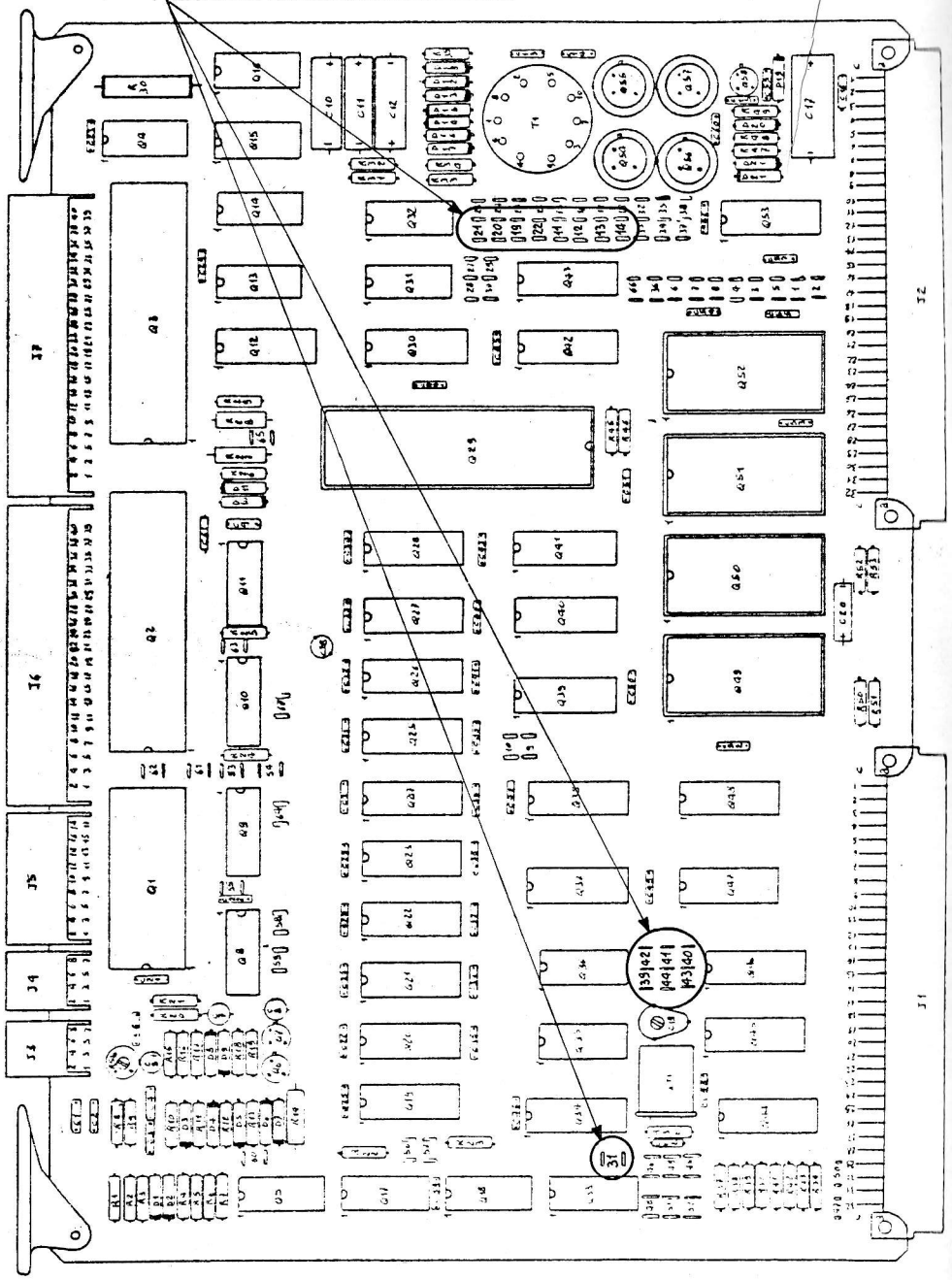


FIG. 1.6.1

1.7 UNITA' SERIALE

Il modulo CLZ80 comprende una interfaccia completa relativa ad unità seriali.

Questa interfaccia permette il collegamento di unità seriali aventi tre diversi livelli elettrici di collegamento.

a) livelli concordi allo standard RS232-C

b) livelli concordi allo standard telegrafico di 20 mA (TTY)

c) livelli compatibili con la famiglia di circuiti integrati TTL

La scelta di uno di questi livelli viene effettuata dall'utente tramite l'opportuno posizionamento di ponticelli in accordo con la tabella 1.7.I.

L'interfaccia permette anche il pilotaggio dell'avanzamento di un eventuale piccolo lettore di banda perforata esistente sul terminale seriale.

La velocità di ricetrasmissione è scelta tramite ponticelli in accordo con la tabella 1.7.II.

Il controllo da parte del software dell'interfaccia seriale può essere effettuato sotto controllo della circuiteria di interruzione oppure no.

La scelta è effettuata dall'utente in funzione della collocazione dei ponticelli opportuni secondo le regole della tabella 1.7.III

TABELLA 1.7.I

INTERFACCIA SERIALE	TIPO DI LINEA DI TRASMISSIONE		
	TTY	RS232	TTL
Ponticelli	54 58 65	54 59 60	53
Ingresso Seriale	RXTTY	RXRS	RXTTL
Ritorno ---	RRXTTY	GND	GND
Uscita Seriale	TXTTY	TXRS	TXTTL
Ritorno ---	RRXTTY	GND	GND

Lettore Unità Seriale	normale	56
	invertito	57

TABELLA 1.7.II

VELOCITA' DI TRASMISSIONE UNITA' SER. BAUD	PONTICELLI			
50	52	50	--	46
75	52	50	--	--
134,5	52	--	48	46
200	52	--	48	--
600	52	--	--	46
2400	52	--	--	--
9600	--	50	48	46
4800	--	50	48	--
1800	--	50	--	46
1200	--	50	--	--
2400	--	--	48	46
300	--	--	48	--
150	--	--	--	46
110	--	--	--	--

TABELLA 1.7.III

INTERRUZIONE DALL' UNITA' SERIALE	PONTICELLI
Trasmiss.	63 - 67
Ricezione	64 - 67
Trasmissione o Ricezione	63-64-67

PONTICELLI DELL'UNITA' SERIALE

TABELLE 1.7.I-1.7.II-1.7.III

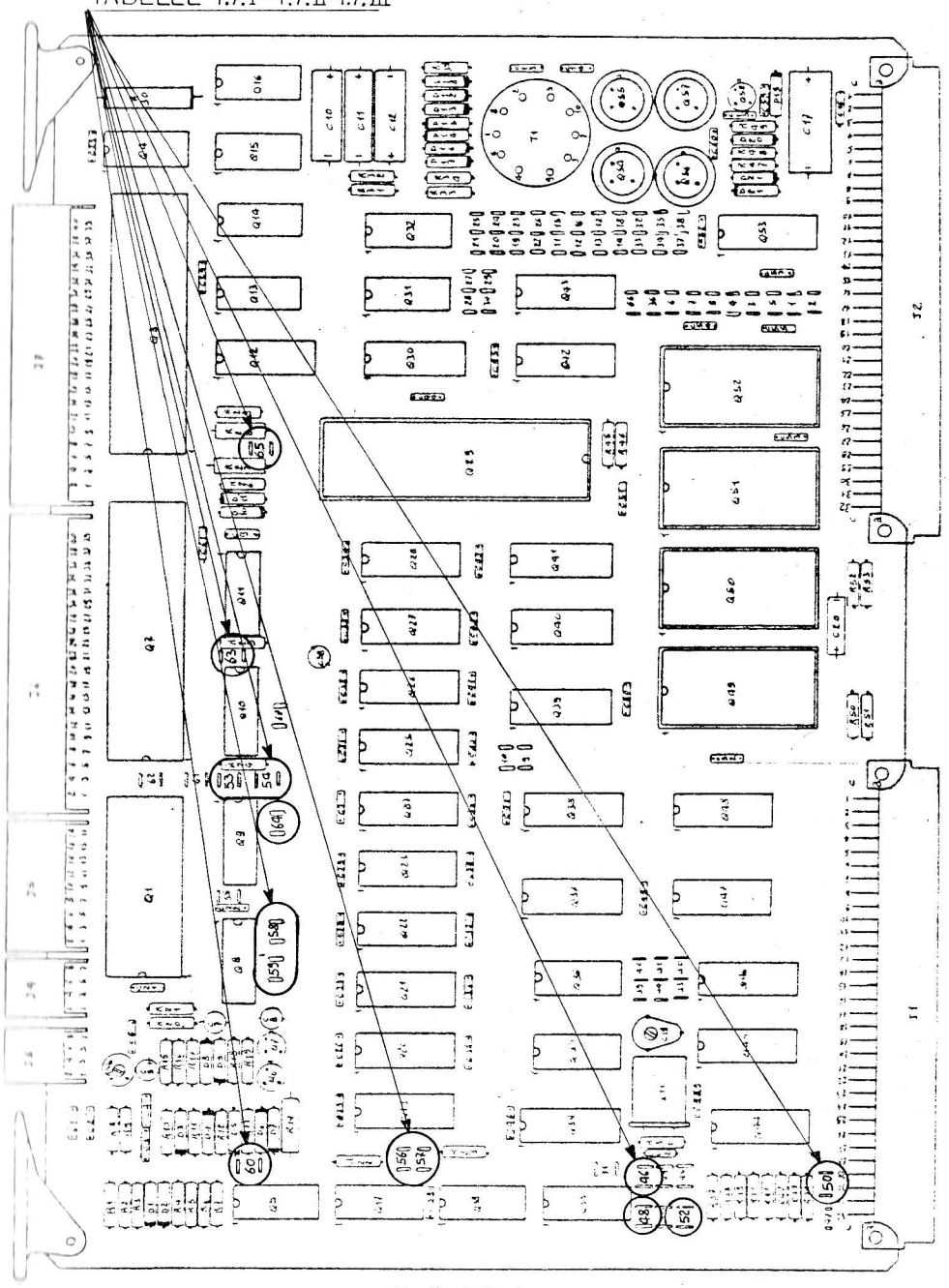


FIG. 1.7.1

L'allocazione topologica dei ponticelli relativi alle tabelle 1.7.1, II e III è illustrata dalla figura 1.7.1.

1.8 INTERFACCIA UNITA' MAGNETICA

Il modulo CLZ80 permette il collegamento verso due registratori magnetici del tipo fonico a cassetta; questo tipo di unità magnetica serve a caricare programmi o liste di caratteri in memoria, ad assemblare programmi in linguaggio sorgente ed a creare nuovi programmi con vaste possibilità di "editing".

L'interfaccia che fa capo al connettore J3 prevede l'allacciamento di due registratori con sensibilità di ingresso di circa 10 ÷ 50 mV e segnale di uscita di circa 300 ÷ 400 mV.

La sensibilità di ingresso può essere variata tramite il potenziometro R54 sul modulo CLZ80 o a mezzo di partitori esterni; il segnale di uscita dal registratore può essere ricondotto ai valori sopra specificati, se richiesto, tramite attenuazione esterna al modulo (partitore resistivo).

L'eventuale controllo automatico del guadagno (C.A.G. o C.A.U.) deve essere assolutamente escluso o con l'opportuno commutatore sul registratore o mediante la modifica del circuito di registrazione.

L'interfaccia prevede anche il controllo del motore di avanzamento dei registratori.

Questo permette l'organizzazione a "file" delle informazioni incise sul nastro magnetico.

Il punto di collegamento sul registratore viene normalmente definito come "REMOTE CONTROL" e se è lasciato sconnesso permette il normale funzionamento del registratore; quando viene connesso a massa blocca la rotazione del motore di trascinamento del nastro magnetico.

La velocità di registrazione corrisponde a 600 BAUD (60 caratteri al secondo) ed il modo di impacchettamento delle informazioni seriali e sotto controllo dell'unità seriale (vedasi cap. 1.7). Le informazioni relative alla programmazione dell'unità seriale sono date nei capitoli avanti che parlano della programmazione.

La tabella 1.8.1 evidenzia le connessioni che permettono il controllo dei motori di avanzamento dei registratori e la figura 1.8.1 la posizione topologica degli stessi.

Si raccomanda l'uso di cassette speciali per usi "DIGITALI" se non si vuole incorrere in errori di registrazione.

PONTICELLI DELL'INTERFACCIA MAGNETICA

TABELLA 1.8.I

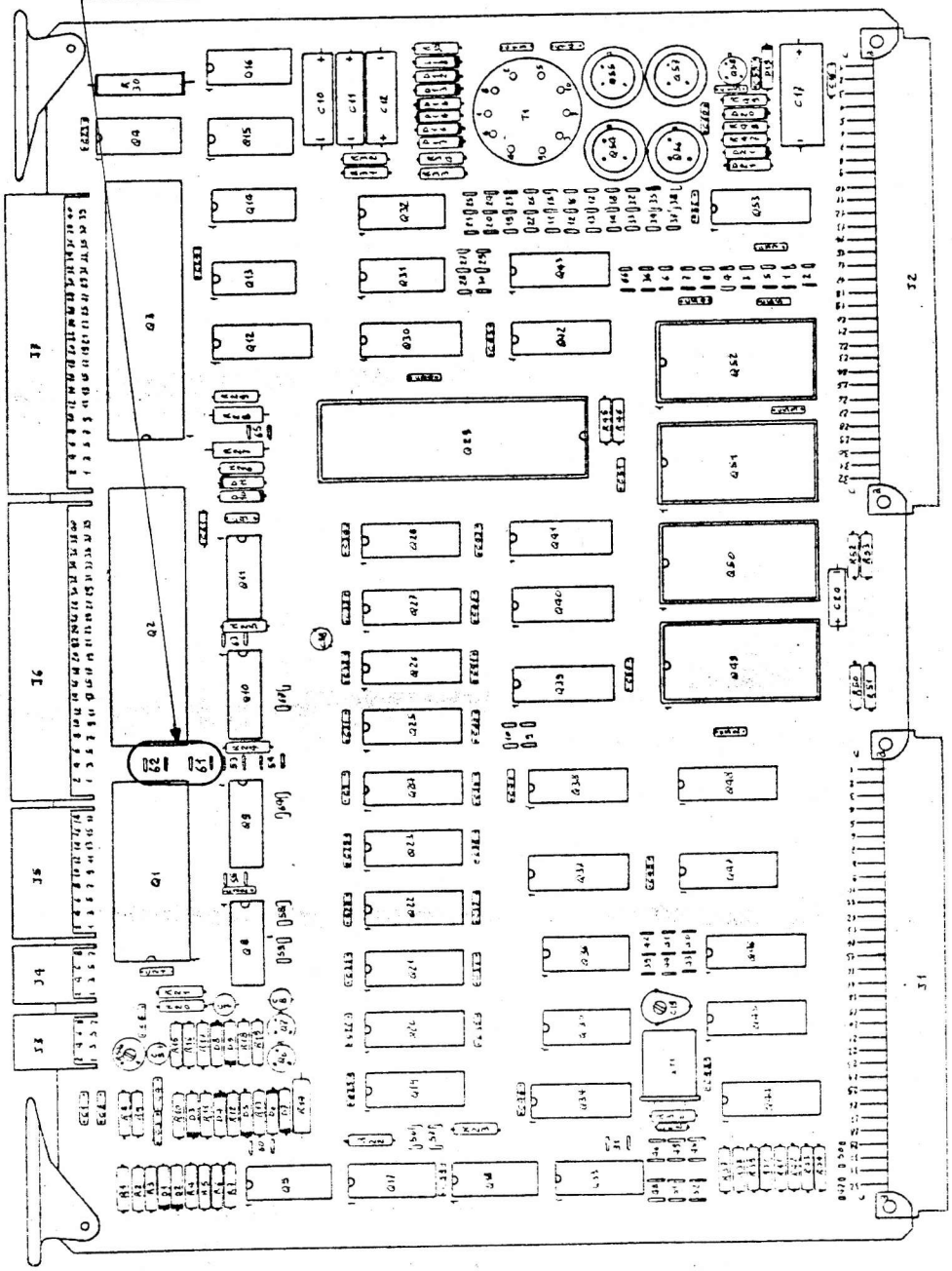


FIG. 1.8.1

TABELLA 1.8.I

CONTROLLO AVANZAMENTO CASSETTE	PONTICELLI
CASSETTA 1	61
CASSETTA 2	62

FIG. 1.8.1 Ponticelli dell'INTERFACCIA MAGNETICA

1.9 SELETTORE PERIFERICHE I/O

Il modulo CLZ80 comprende un circuito transcodificatore di 32 posizioni di indirizzo di periferiche (delle 256 possibili); le prime 12 posizioni sono utilizzate all'interno per indirizzare l'UNITA' SERIALE (cap. 1.7) e le PORTE I/O (cap. 1.10; le altre 20 sono disponibili per l'utente che ne voglia far uso. Queste 20 posizioni già transcodificate su 9 linee sono presenti sul BUS-ESPANSIONE e permettono all'utente di decodificare qualsiasi delle 20 posizioni tramite l'uso di un semplice "AND" a 2 ingressi; ciò semplifica e riduce notevolmente l'uso di circuiti esterni necessari alla transcodifica degli indirizzi di I/O.

La tabella 1.9.I illustra la combinazione dei segnali (a due a due) che danno origine ai 32 indirizzi di periferiche; il numero della periferica è espresso in esadecimale per semplificare il lavoro del programmatore.

La base del blocco delle 32 posizioni di indirizzo può essere cambiata in modo da poter allocare il blocco in una qualsiasi delle 8 partizioni esistenti nel campo di indirizzamento delle periferiche; la tabella 1.9.II evidenzia questa possibilità ed i ponticelli interessati all'operazione; questi sono poi evidenziati topologicamente in fig. 1.9.I.

Tabella 1.9.I

INDIRIZZI I/O	$\overline{IOU0}$	$\overline{IOU1}$	$\overline{IOU2}$	$\overline{IOU3}$
$\overline{IOQ0}$	0 USART	1 USART	2 N O N USATA	3 FLAG I/O
$\overline{IOQ1}$	4 DATI PORTA "A"	5 DATI PORTA "B"	6 CONT. PORTA "A"	7 CONT. PORTA "B"
$\overline{IOQ2}$	8 DATI PORTA"C"	9 DATI PORTA "D"	A CONTR. PORTA "C"	B CONT. PORTA "D"
$\overline{IOQ3}$	C	D	E	F
$\overline{IOE0}$	10	11	12	13
$\overline{IOE1}$	14	15	16	17
$\overline{IOE2}$	18	19	1A	1B
$\overline{IOE3}$	1C	1D	1E	1F

LINEE DISPONIBILI

Decodifiche per uso interno non dispon. per l'esterno.'

PONTICELLI DEL SELETTORE PERIFERICHE I/O

TABELLA 1.9.1

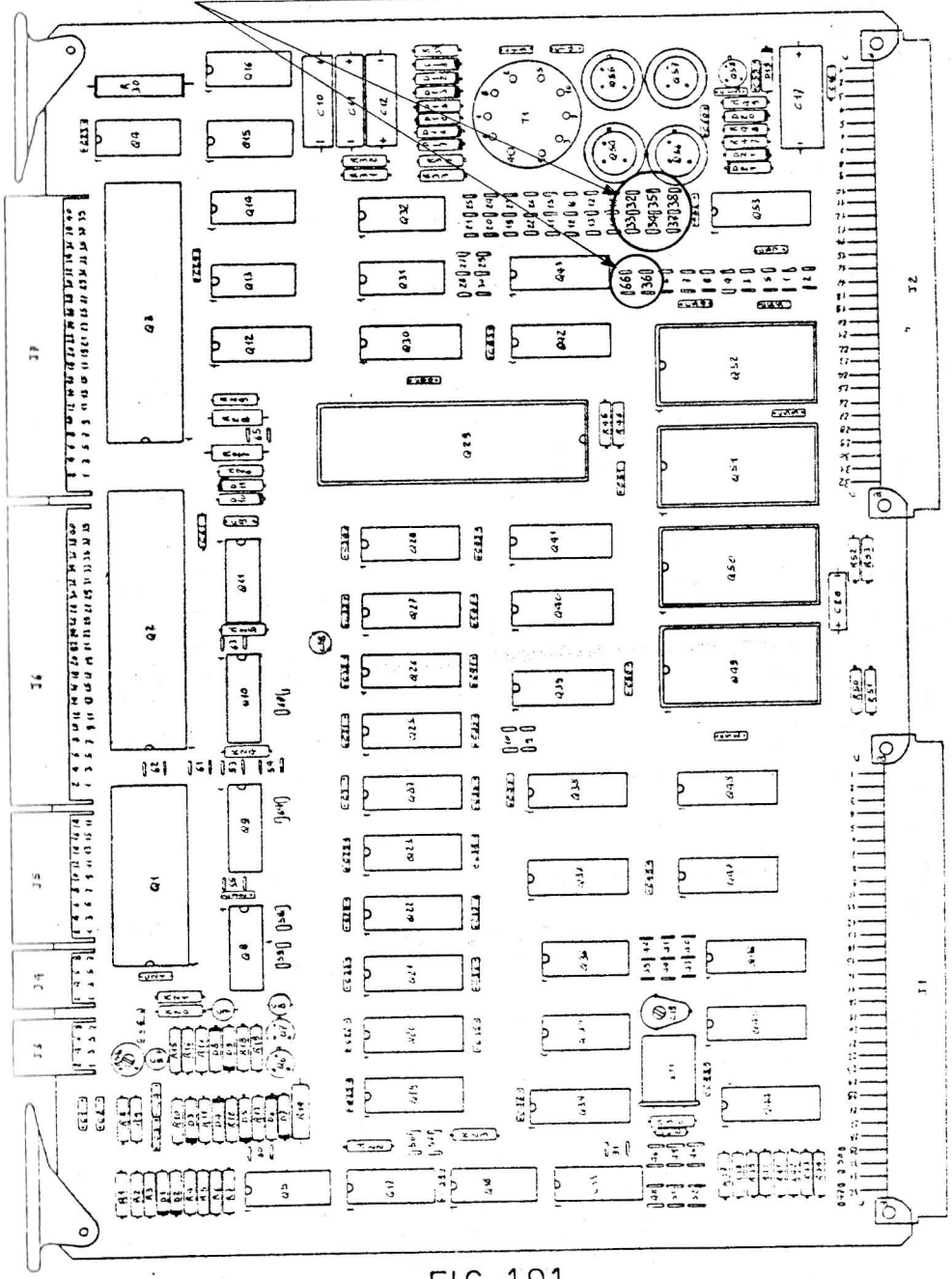


FIG. 1.9.1

TABELLA 1.9.II

CODICE INIZIALE DECODIFICA PERIFERICHE		PONTICELLI			
		36-37	34-38	36-66	35-38
PONTICELLI	32	0	20	40	60
	33	80	100	120	160

1.10 PORTE I/O

Il modulo CLZ80 dà all'utente la possibilità di interfacciarsi direttamente al processo esterno da controllare senza la necessità di costruirsi un'interfaccia dedicata; il modulo CLZ80 fornisce 4 porte di 8 bit ciascuna con la possibilità di attuarle come porte di ingresso o porte di uscita in una qualsiasi combinazione desiderata.

Ogni porta oltre a fornire 8 linee per trasportare i dati fornisce due segnali di sincronizzazione (o handshaking) FLPA (B,C,D) e $\overline{\text{STPA}}$ (B,C,D); il primo, FLPA, indica la disponibilità della porta a trasmettere o ricevere un segnale; il secondo, $\overline{\text{STPA}}$, rappresenta l'indicazione da parte dell'utente esterno della validità del dato da trasmettere o da ricevere.

In un capitolo più avanti verrà trattato in dettaglio l'argomento con riguardo alla programmazione e alla temporizzazione delle porte.

Si vuole qui invece indicare che la coppia di porte A e B fa capo al connettore J6, mentre la coppia B e C fa capo al connettore J7.

Quando la porta funge da porta di uscita, ogni sua linea può pilotare un carico TTL normale; quando funge da porta di ingresso assorbe dal pilota esterno una corrente massima di 10 μA.

Si noti che il numero alto di "pins" di massa sui connettori J6 e J7 non è casuale ma intenzionale e dovrebbe permettere la connessione a massa dello schermo delle linee dei segnali di sincronizzazione FLPA/B/C/D, $\overline{\text{STPA}}$/B/C/D; con questa precauzione si evitano fastidiose interazioni con disturbi provenienti dall'esterno; buona precauzione sarebbe quella di usare del cavetto schermato anche per le linee dei dati.

1.11 CARATTERISTICHE FUNZIONALI BUS-ESPANSIONE

I paragrafi di questo capitolo illustrano in dettaglio il funzionamento del BUS ESPANSIONE in modo che l'utente possa interfacciarsi direttamente con la propria unità; vengono date tutte le informazioni che permettono di progettarsi i controlli per memorie, per periferiche controllate dal CPU e per periferiche totalmente autonome che trasferiscono i dati direttamente in memorie (periferiche DMA).

1.11.1 Temporizzazione del CLZ80

Il modulo CLZ80 esegue, attraverso il proprio CPU-Z80, delle istruzioni mediante il sequenziamento preciso di un gruppo limitato di operazioni; queste operazioni sono:

- LETTURA O SCRITTURA IN MEMORIA
- LETTURA O SCRITTURA IN PERIFERICHE
- RICONOSCIMENTO DI INTERRUZIONI

Tutte le istruzioni, quindi, possono essere ricondotte all'esecuzione sequenziale di una miscela di queste operazioni.

Ognuna di queste operazioni di base può richiedere, in termini di tempo, da tre a sei periodi di clock (segnale BØ) per la sua completa attuazione; in caso di utilizzo di memorie o periferiche lente è possibile che il numero di periodi di clock aumentino; ciò avviene non perchè vengano eseguite operazioni più complesse, ma semplicemente perchè il CPU congela le proprie azioni in aspettativa dei dati dall'esterno.

L'operazione di congelamento è effettuata tramite il segnale $\overline{\text{BWAIT}}$.

I periodi di clock base saranno chiamati nel resto della trattazione con il nome di cicli T (T_1, T_2, ecc.), mentre le operazioni di base con il nome di cicli M (M_1, M_2, ecc.).

La figura 1.11.1.1 illustra il sequenziamento tipico di una ipotetica istruzione.

Si noti che questa istruzione si compone di tre cicli di macchina (M_1, M_2, M_3).

Il primo ciclo di macchina di qualsiasi istruzione è SEMPRE un ciclo di ricerca del codice operativo che identifica univocamente l'istruzione da eseguirsi.

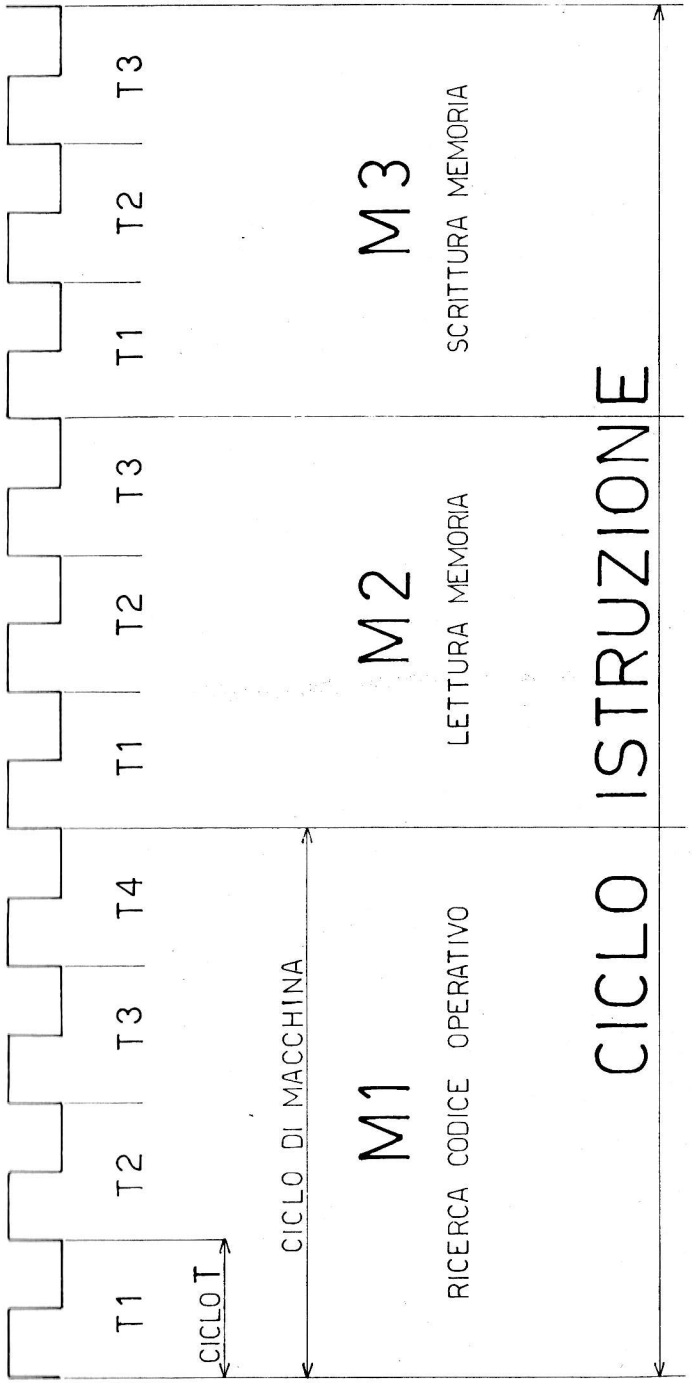


FIG.1.11.1.1

ESEMPIO DI SEQUENZIAMENTO DI UNA ISTRUZIONE

Questo ciclo, chiamato M_1, si può comporre di quattro, cinque o sei cicli T (senza considerare gli allungamenti del ciclo provocati dal segnale di attesa $\overline{\text{BWAIT}}$)

I seguenti cicli M_2, M_3, etc. provvedono al movimento dati tra il CPU e la memoria o le perifieriche; questi cicli possono essere composti da tre o cinque cicli T (anche questi cicli possono essere allungati dal segnale di attesa $\overline{\text{BWAIT}}$).

I paragrafi che seguono illustreranno l'esatta temporizzazione di tutti i possibili cicli di macchina con o senza lo stato di attesa (il quale è aggiunto nel caso di memorie o periferiche lente).

Diamo di seguito l'elenco di questi cicli di macchina.

- RICERCA DEL CODICE OPERATIVO DELL'ISTRUZIONE
- SCRITTURA E LETTURA DATI IN MEMORIA
- SCRITTURA E LETTURA DATI NELLA PERIFERICA
- RICHIESTA E RICONOSCIMENTO BUS-ESPANSIONE
- RICHIESTA E RICONOSCIMENTO INTERRUZIONE
- RICHIESTA E RICONOSCIMENTO INTERRUZIONE NON MASCHERABILE.
- ISTRUZIONE HALT.

1.11.2 Ricerca Codice Operativo

La figura 1.11.2.1 illustra la temporizzazione durante il ciclo M_1.

Il contenuto del PC è forzato sulle linee di indirizzo (BAD0÷15) immediatamente all'inizio del ciclo in modo da minimizzare il tempo di accesso richiesto alla memoria.

Con un ritardo di mezzo ciclo di clock viene attivato il segnale $\overline{\text{BMREQ}}$ in modo da dare tempo sufficiente alle linee di indirizzo di stabilizzarsi. Ciò permette anche di usare questo segnale come CHIP-ENABLE delle memorie dinamiche.

Contemporaneamente al segnale $\overline{\text{BMREQ}}$ diventa attivo anche il segnale $\overline{\text{BRD}}$; questo segnale dovrebbe essere usato per abilitare sulle linee-dati (BD0÷7) il dato letto o in procinto di essere letto dalla memoria.

Il CPU campiona i dati forniti dalla memoria con il fronte positivo del clock relativo al ciclo T_3; lo stesso fronte è utilizzato dal CPU per disattivare i segnali $\overline{\text{BMREQ}}$ e $\overline{\text{BRD}}$; ciò significa anche che i dati sono campionati dal CPU prima che il segnale $\overline{\text{BRD}}$ diventi inattivo.

Gli stati T_3 e T_4 di un ciclo di ricerca M_1 sono adibiti all'azione di rinfresco delle memorie dinamiche.

Mentre all'esterno avviene questa operazione di rinfresco, nel suo interno il CPU provvede alle decodifica del codice operativo appena letto e, nel caso di istruzioni che non coinvolgono la memoria o le periferiche all'esecuzione dell'istruzione stessa.

E' importante notare che durante T_3 e T_4 di M_1 non viene attuato nessun traffico di dati con l'esterno del CPU per cui questi due cicli possono essere completamente dedicati al rinfresco senza peraltro pessimizzare il tempo di esecuzione dell'istruzione.

L'azione di rinfresco viene in dettaglio attuata forzando sulle sette linee di indirizzo relative ai bit meno significativi (BAD0 ÷6) il contenuto del registro I ed attivando il segnale $\overline{BRFSH}$ che segnala a tutte le memorie dinamiche (indipendentemente dalla loro allocazione particolare nel campo possibile di 64K bytes) l'attuazione del rinfresco.

Si noti che durante l'attivazione del segnale $\overline{BRFSH}$ non viene attuato il segnale $\overline{BRD}$ in modo da evitare che i dati provenienti da più memorie dinamiche possano essere forzati contemporaneamente sul bus-dati; d'altra parte anche durante il ciclo di rinfresco viene attivato il segnale $\overline{BMREQ}$ poichè solo durante la sua attivazione è garantito stabile il

valore presente sulle linee degli indirizzi.

Si noti in figura 1.11.2.1 che durante il fronte negativo del clock nel ciclo T_2 il segnale $\overline{BWAIT}$ è disattivato; questo nasce dal presupposto che la memoria interessata dalla lettura abbia un tempo di accesso inferiore o uguale a quello richiesto dal CPU.

Se ciò non fosse possibile è necessario congelare l'attività del CPU attivando il segnale $\overline{BWAIT}$ durante il fronte negativo del clock in T_2.

Come illustrato in figura 1.11.2.2 è possibile inserire un numero qualsiasi di stati di attesa mantenendo opprtunamente attivata durante il fronte negativo del clock la linea $\overline{BWAIT}$.

Alla disattivazione di questa linea il CPU proseguirà ad eseguire i seguenti cicli T_3 e T_4.

TEMPORIZZAZIONE RELATIVA AL CICLO DELLA RICERCA DEL CODICE OPERATIVO

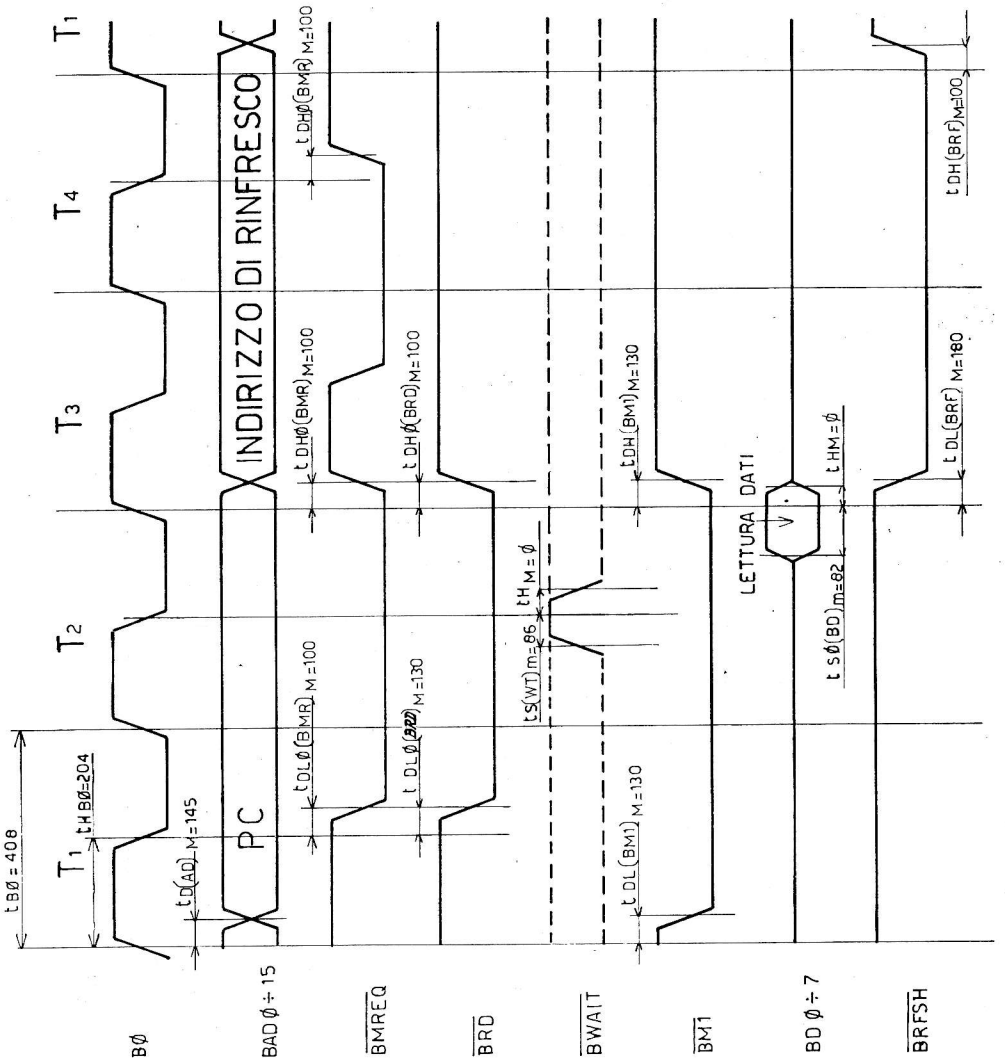


FIG.1.11.2.1

CICLO RICERCA CODICE OPERATIVO CON STATI DI ATTESA Tw

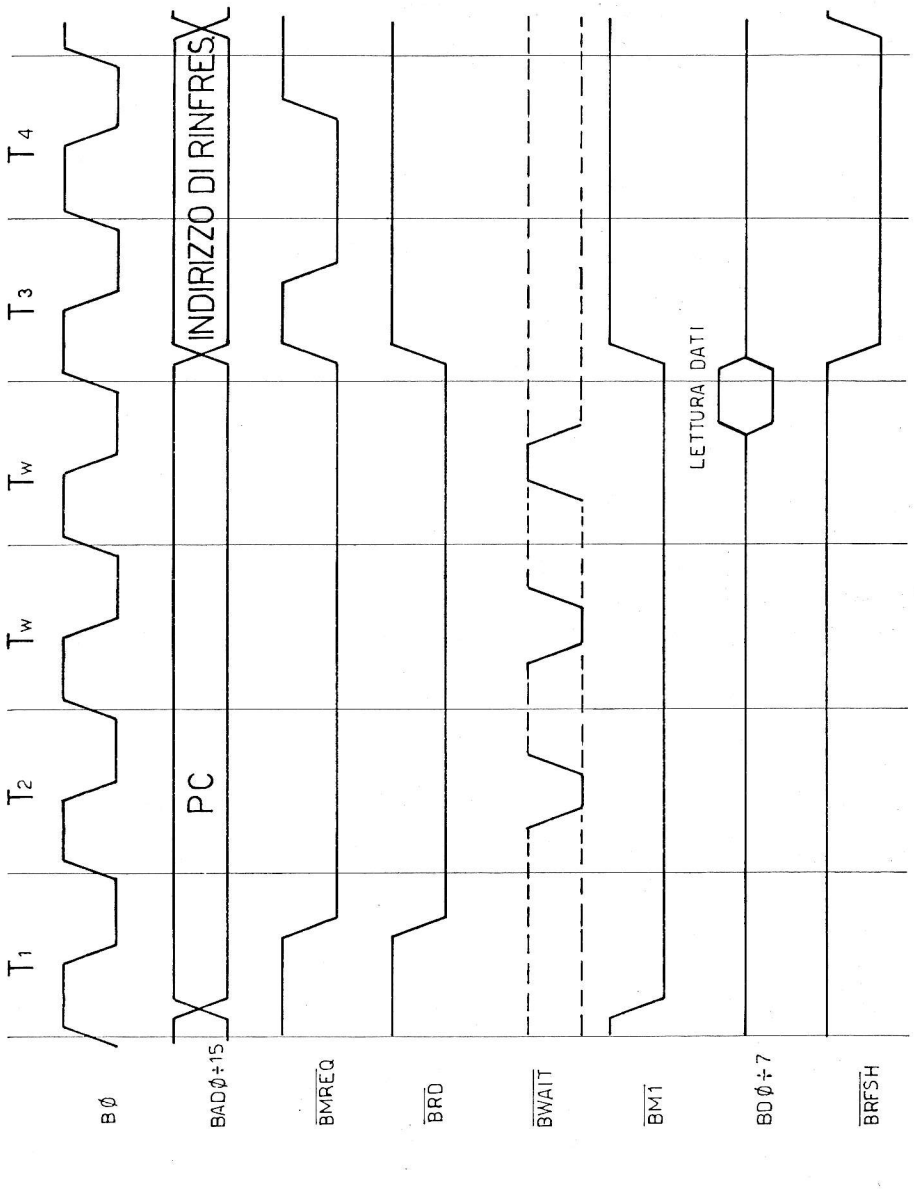


FIG.1.11.2.2

1.11.3 Scrittura e lettura dati in memoria

La figura 1.11.3.1 illustra la temporizzazione relativa ad un ciclo di lettura e ad un ciclo di scrittura in memoria (il ciclo di lettura qui considerato non è da confondere con quello considerato nel ciclo di ricerca del codice operativo. M_1).

Questi due cicli sono generalmente composti da tre periodi di clock nel caso che non vengano richiesti cicli di attesa TW da parte della memoria.

Il segnale $\overline{\text{BMREQ}}$ ed il segnale $\overline{\text{BRD}}$ sono usati nello stesso modo che nel ciclo di ricerca del codice operativo con la differenza che il tempo di accesso richiesto in questo caso è maggiore.

Per il ciclo di lettura dalla memoria il segnale $\overline{\text{BMREQ}}$ diventa attivo quando le linee degli indirizzi BAD0÷15 sono stabili; Questo permette di usarlo come segnale di "chip-enable" per le memorie dinamiche.

Nel ciclo di scrittura in memoria si noti che la linea $\overline{\text{BWR}}$ diventa attiva allo stesso istante in cui diventano attivi i segnali da scrivere in memoria. Questo significa che nel caso di alcune memorie dinamiche è necessario ritardare il segnale $\overline{\text{BWR}}$ nei confronti dei dati prima di applicarlo al piedino

"R/W" del dispositivo integrato; ciò è normalmente non necessario per quasi tutte le memorie statiche.

Il segnale $\overline{\text{BWR}}$, d'altra parte, ridiventa inattivo mezzo ciclo T (in particolare T_3) prima che cambino le linee degli indirizzi e dei dati. Questo permette di soddisfare le esigenze e le richieste di quasi tutte le memorie dinamiche.

La figura 1.11.3.2 illustra l'effetto di allungamento del ciclo provocato dal segnale $\overline{\text{BWAIT}}$. Il ciclo illustrato in questa figura è funzionalmente identico a quelli di figura 1.11.3.1 con la sola differenza del congelamento delle condizioni trovate durante T_2.

Si ponga attenzione altresì che in questa figura per comodità di illustrazione sono disegnati contemporaneamente il ciclo di lettura e quello di scrittura: è evidentemente da intendersi che questo non avviene MAI in realtà.

CICLI DI LETTURA E SCRITTURA IN MEMORIA

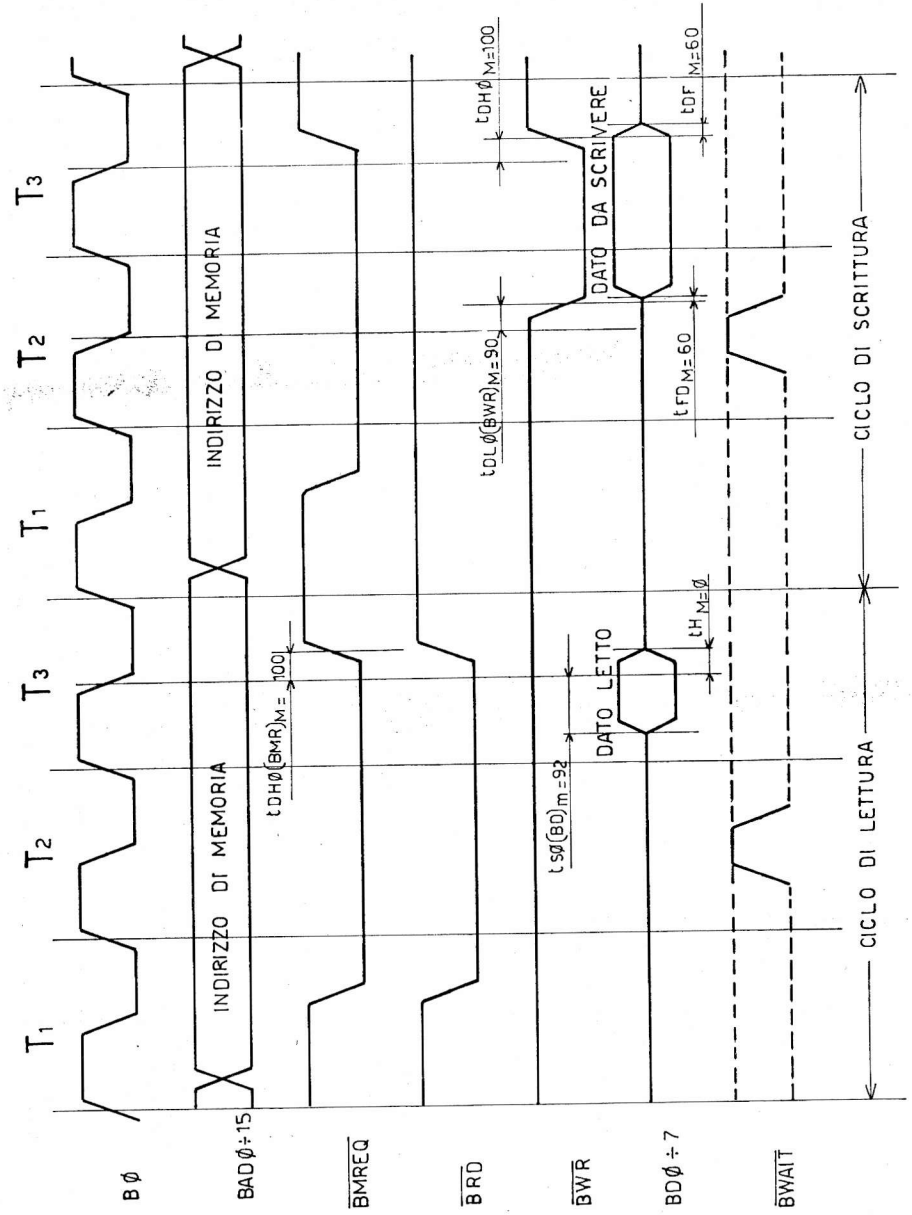


FIG. 1.11.3.1

CICLO DI SCRITTURA O LETTURA IN MEMORIA CON STATI DI ATTESA

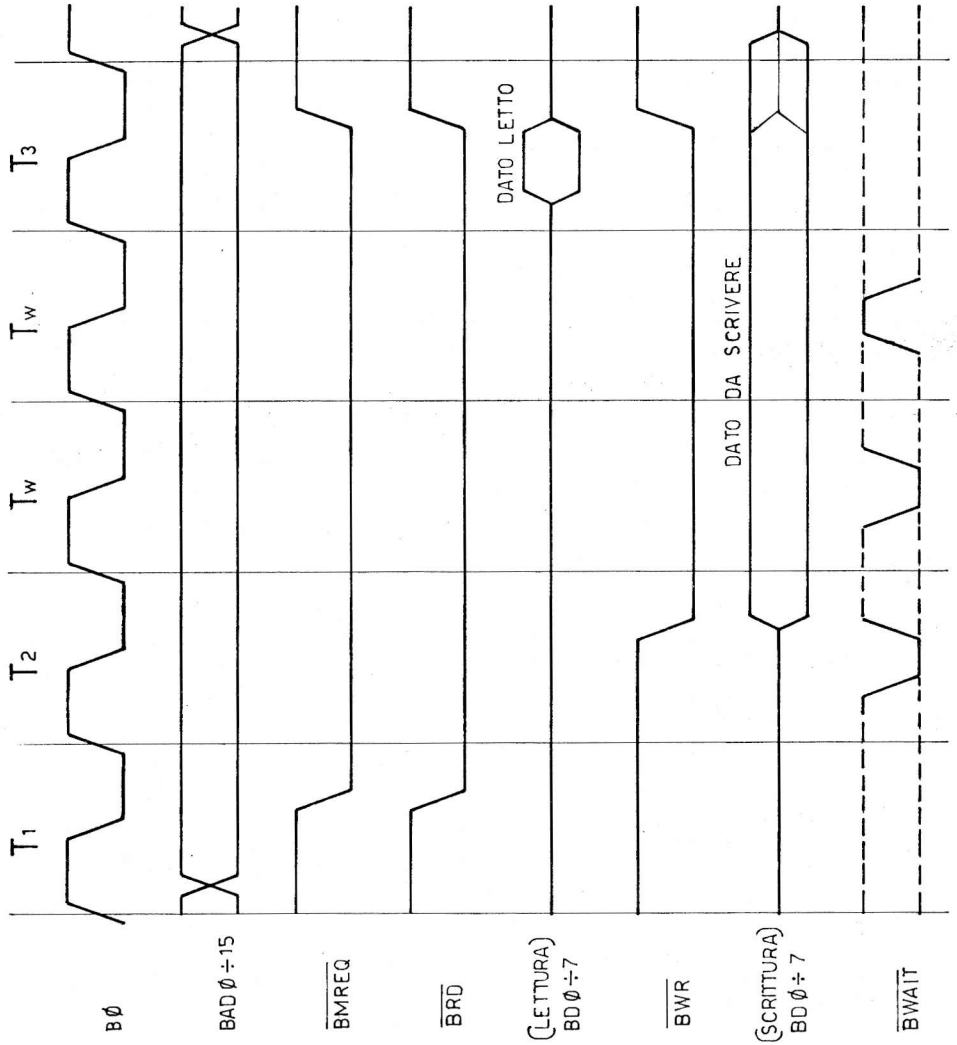


FIG. 1.11.3.2

1.11.4 Scrittura e lettura dati nella periferica

La figura 1.11.4.1 illustra la temporizzazione relativa ad un ciclo di lettura e ad un ciclo di scrittura di una periferica.

Si noti che la più grande differenza rispetto ai rispettivi cicli di memoria è l'inserimento automatico di uno stato di attesa non causato da un segnale di $\overline{\text{BWAIT}}$; chiameremo questo speciale stato di attesa con il simbolo TW*.

Lo stato di attesa supplementare TW* è stato inserito a causa del breve tempo esistente tra l'attivazione del segnale $\overline{\text{BIORQ}}$ e l'istante (fronte negativo del clock in T_2) in cui il CPU campiona la richiesta di ciclo di attesa (fatta tramite il segnale $\overline{\text{BWAIT}}$ (tipicamente questo tempo è circa di mezzo periodo T).

Senza questo stato supplementare TW* sarebbe stato molto difficile per la periferica decodificare il proprio indirizzo e attivare in tempo un normale stato di attesa se richiesto; in questa situazione invece il segnale $\overline{\text{BWAIT}}$ è campionato durante lo stato TW*. Si può dire inoltre che sarebbe difficile progettare circuiti che possano sfruttare bene ed efficientemente la loro velocità senza questa alterazione al ciclo.

Il resto del ciclo richiama le stesse regole viste per i cicli di memoria.

Il segnale $\overline{\text{BRD}}$ è usato per abilitare il dato della periferica sulle linee BDO÷7, mentre il segnale $\overline{\text{BWR}}$ può essere usato nella periferica per accedere al dato presente sulle linee BDO ÷ 7 e immagazzinarlo nel proprio interno.

La fig. 1.11.4.2 illustra come eventuali stati di attesa normali TW possano essere aggiunti tramite il controllo della linea $\overline{\text{BWAIT}}$.

Una funzione molto importante da sottolineare è quella delle linee di indirizzo BAD8÷15 che non sono utilizzate per indirizzare la periferica (che è indirizzata dalle linee BAD0÷7). Queste linee portano il contenuto del registro "A" nelle istruzioni semplici di scrittura o lettura nelle periferiche (istruzioni IN A,(n); OUT(n),A), mentre portano il contenuto del registro "B" nelle istruzioni di ingresso-uscita con indirizzamento indiretto attraverso il registro C.

CICLO DI LETTURA O SCRITTURA DELLA PERIFERICA

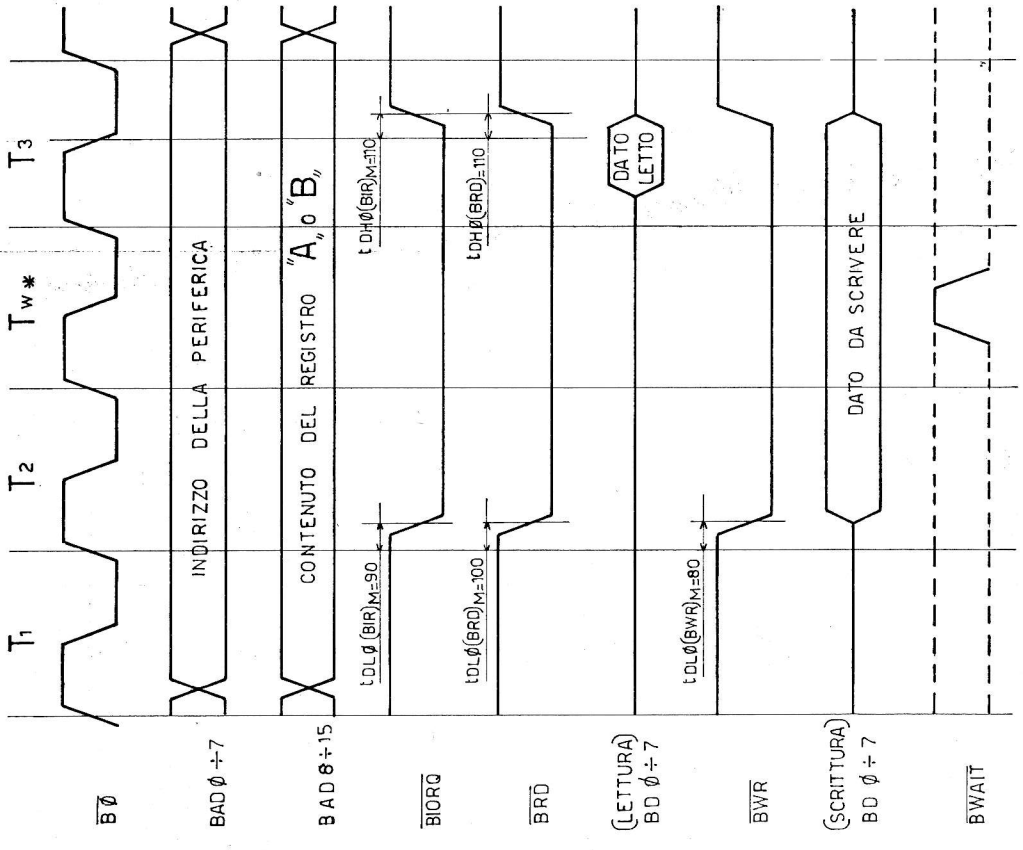


FIG.1.11.4.1

CICLO DI SCRITTURA O LETTURA DELLA PERIFERICA CON STATI DI ATTESA TW

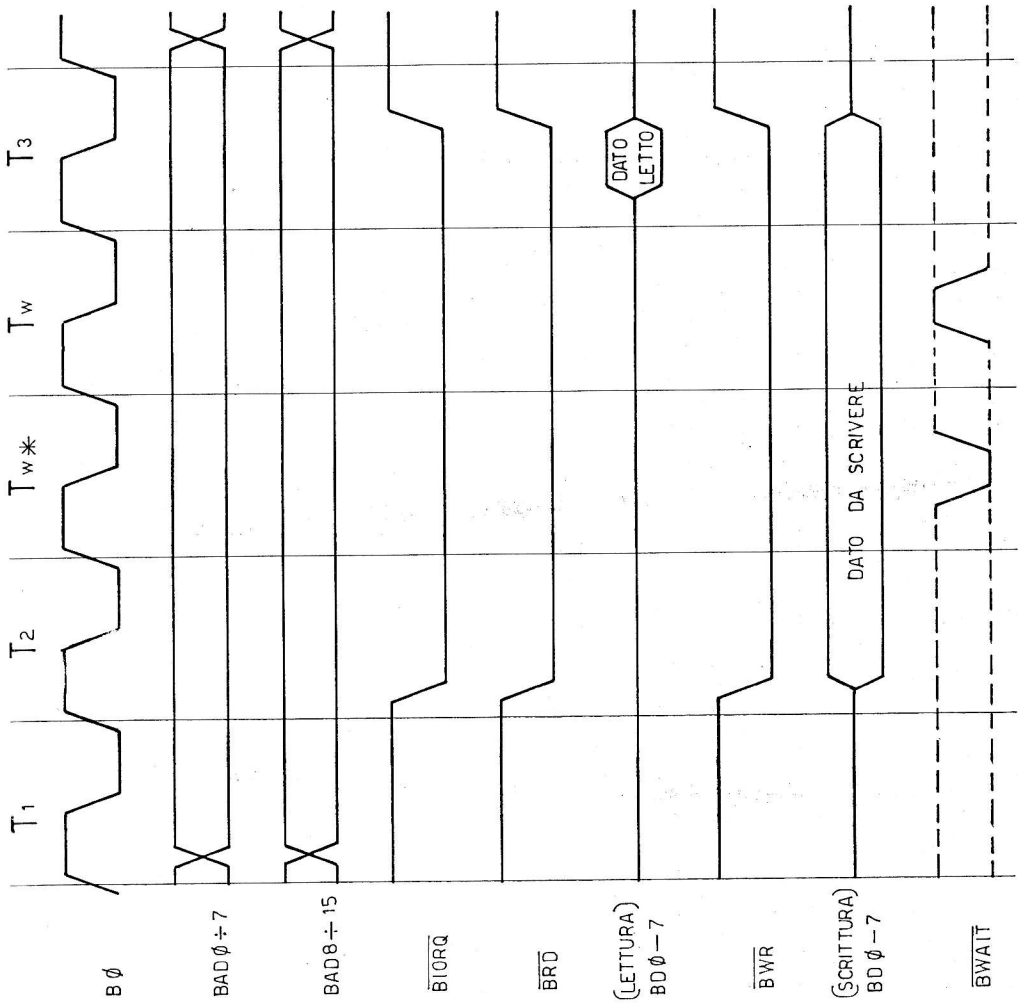


FIG. 1.11.4.2

1.11.5 Richiesta e riconoscimento Bus-espansione

Quando una periferica vuole accedere direttamente ai dati della memoria del sistema si deve sostituire al CPU del modulo CLZ80 e pilotare autonomamente il BUS-ESPANSIONE.

La temporizzazione relativa alla richiesta e al rilascio del BUS, rispettivamente da parte della periferica e del CPU, è illustrata in figura 1.11.5.1.

La periferica comunica al CPU la sua intenzione di prendere il BUS-ESPANSIONE attivando in qualsiasi momento la linea $\overline{BBUSRQ}$; il CPU campiona, alla fine di qualsiasi ciclo M (M_1, M_2, ecc.), con il fronte positivo dell'ultimo periodo di clock, la linea $\overline{BBUSRQ}$.

Se il segnale $\overline{BBUSRQ}$ è trovato attivato il CPU porrà a disposizione della periferica il BUS-ESPANSIONE in concomitanza del fronte positivo del seguente ciclo di clock affermando il segnale $\overline{BBUSAK}$.
Porre a disposizione significa forzare nel terzo dato tutti i piloti del BUS-ESPANSIONE indicati con il gruppo elettrico ① (Si veda capitolo 4, paragrafo 2).

Da questo istante la periferica è responsabile dell'uso delle linee del BUS-ESPANSIONE per poter accedere alla memoria o ad altre periferiche.

RICHIESTA E RICONOSCIMENTO BUS-ESPANSIONE

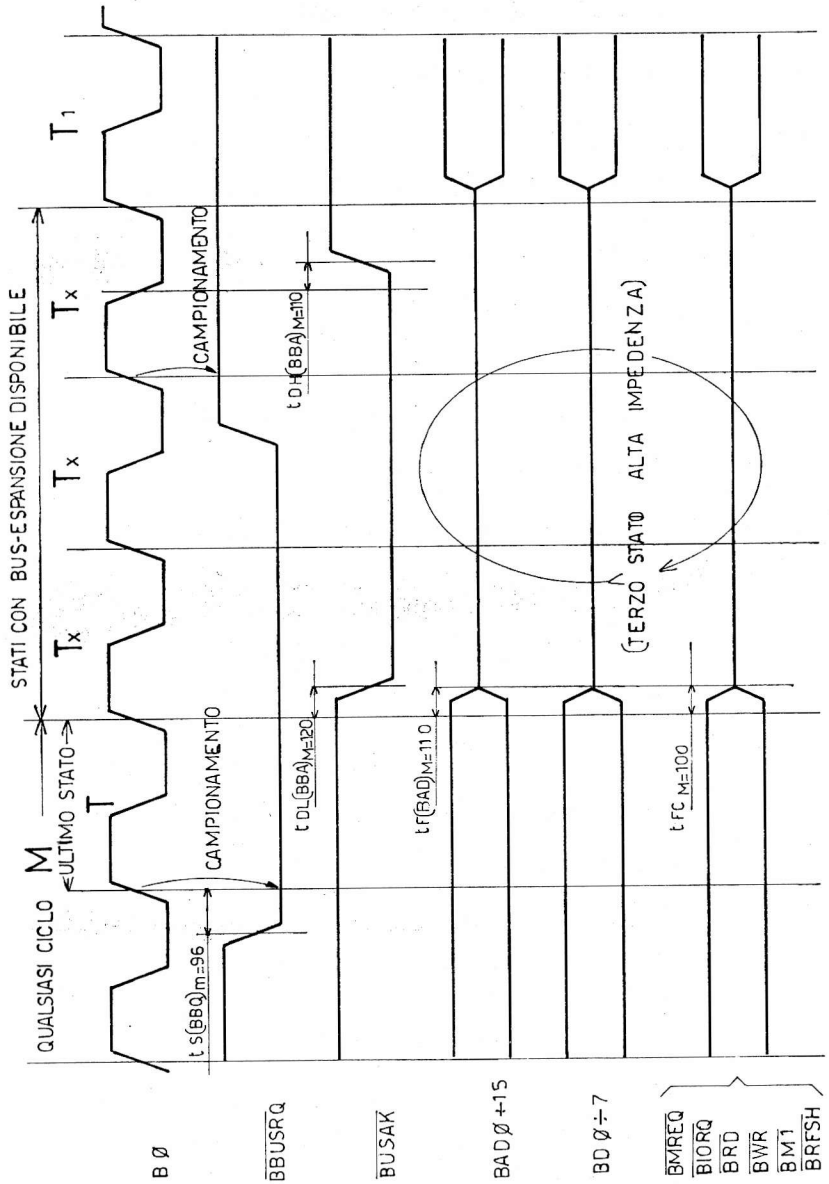


FIG.1.11.5.1

Il tempo massimo necessario al CPU del modulo CLZ80 per rispondere ad una richiesta di BUS-ESPANSIONE corrisponde al ciclo macchina più lungo; il controllore esterno del BUS-ESPANSIONE può d'altra parte mantenere la gestione dello stesso per un numero indeterminato di cicli di clock.

Si deve però notare che se la periferica controllante il BUS-ESPANSIONE richiede molti cicli di clock, essa deve gestire anche il rinfresco delle memorie dinamiche, che,non più rinfrescate dal CPU, perderebbero le loro informazioni.

Si noti, d'altra parte, che questa azione è richiesta solo quando vengono effettuati trasferimenti di grossi blocchi di dati dalla memoria.

1.11.6 RICHIESTA E RICONOSCIMENTO INTERRUZIONE

Nella figura 1.11.6.1 è illustrata la temporizzazione associata ad un ciclo di richieste di interruzione. Il segnale di interruzione è campionato dal CPU con il fronte positivo dell'ultimo ciclo di clock alla fine dell'esecuzione di ogni istruzione.

La richiesta di interruzione non sarà accettata dal CPU se non è stata preceduta dall'istruzione abilitante il flip-flop di sensibilizzazione all'interruzione, oppure quando è abilitato contemporaneamente il segnale $\overline{\text{BBUSRQ}}$; il CPU quindi è sotto controllo del software per ciò che riguarda la sensibilizzazione alla richiesta di interruzione; una volta eseguita l'istruzione di abilitazione del flip-flop di interruzione qualsiasi richiesta sia fatta in un tempo seguente, sarà soddisfatta.

Quando la richiesta è accettata il CPU esegue uno speciale ciclo M1; durante questo ciclo speciale diventa attivo il segnale $\overline{\text{BIORQ}}$ (al posto del segnale normale $\overline{\text{BMREQ}}$) per indicare alla periferica interrompente che può forzare sulle linee BD 0 ÷ 7 il vettore di interruzione

RICHIESTA E RICONOSCIMENTO INTERRUZIONE

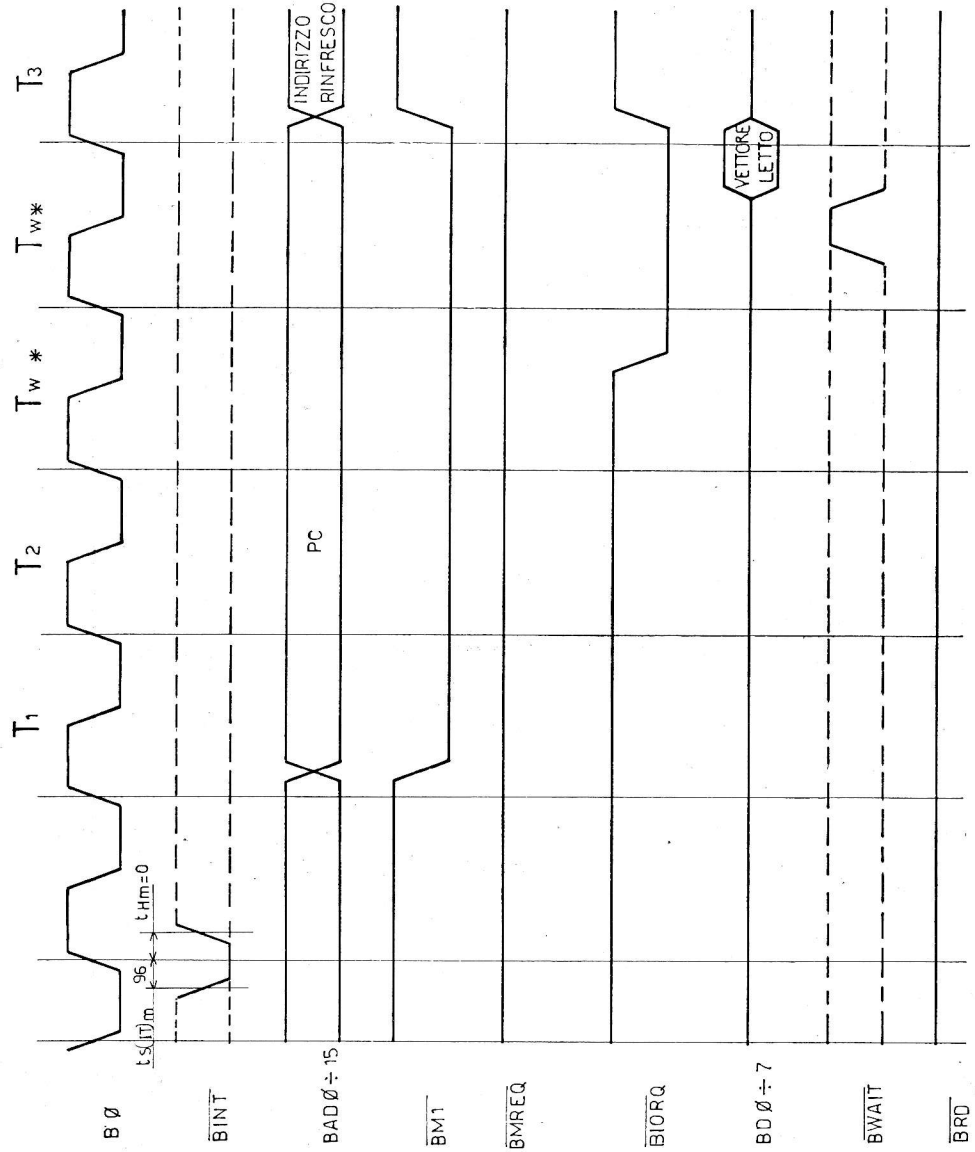


FIG.1.11.6.1

RICHIESTA E RICONOSCIMENTO INTERRUZIONE CON STATI DI ATTESA

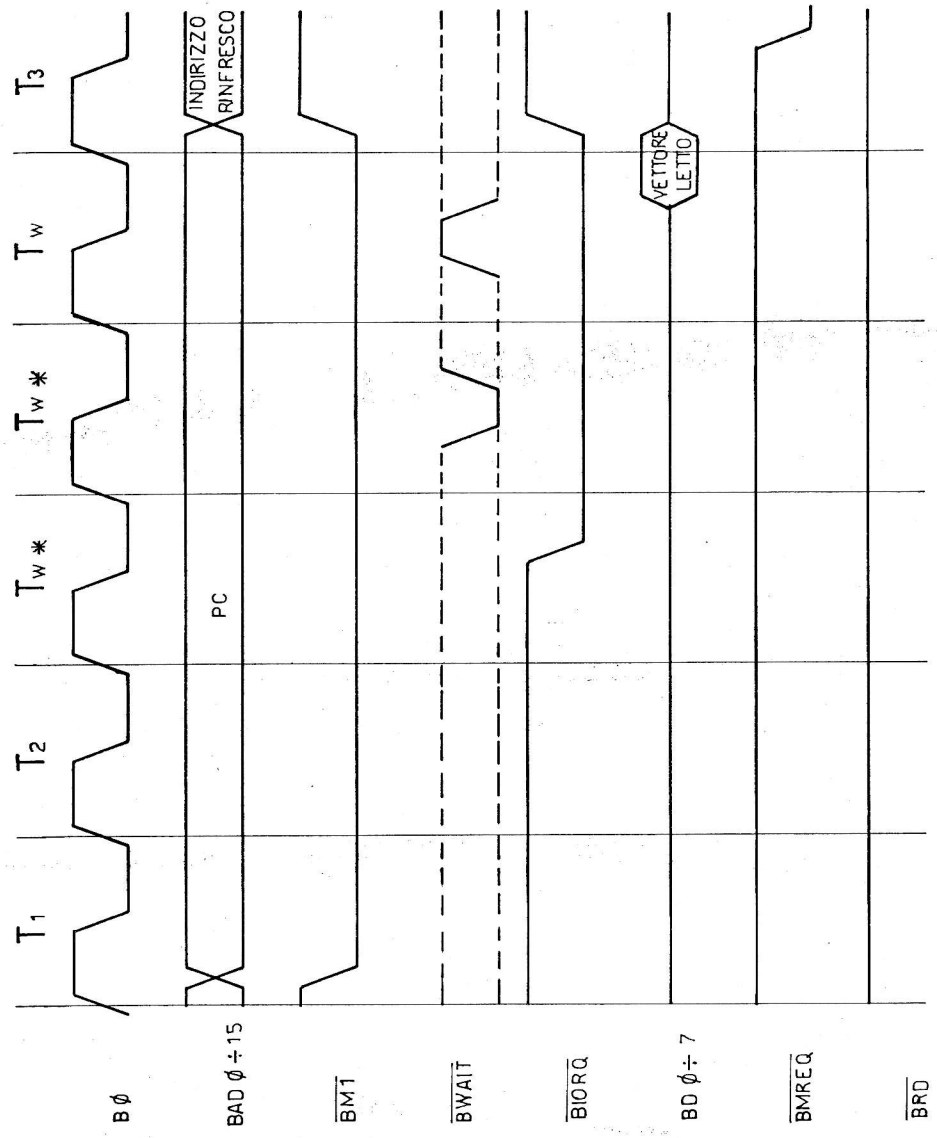


FIG.1.11.6.2

Si noti che sono inseriti nel ciclo di riconoscimento due cicli di attesa in modo automatico; questo permette che la circuiteria ad anello della priorità abbia tempo di assestarsi senza richiedere dispositivi esterni molto veloci; i due stati di attesa permettono anche che ci sia tempo sufficiente a che il segnale dell'anello della priorità si stabilizzi ed identifichi quale periferica debba inserire il vettore di risposta.

Si faccia riferimento alle sezioni del software per la spiegazione in dettaglio di come il CPU utilizzi al suo interno il vettore di interruzione.

La fig. 1.11.6.2 illustra un ciclo di richiesta e riconoscimento dell'interruzione con un ciclo di attesa TW inserito.

1.11.7 RICHIESTA E RICONOSCIMENTO INTERRUZIONE NON MASCHERABILE

La figura 1.11.7.1 illustra la temporizzazione di un ciclo relativo ad una richiesta e riconoscimento di una interruzione non mascherabile.

Il segnale $\overline{BNMI}$ non viene campionato alla fine di un ciclo d'istruzione; ma è accettato a qualsiasi momento con la sola restrizione che la sua durata minima non deve essere inferiore a 80 nsec.

Se l'attivazione è avvenuta prima del fronte positivo dell'ultimo clock di un ciclo d'istruzione, viene attivata la risposta all'interruzione; questa risposta non è sottoposta al controllo del software per cui il CPU risponde sempre e comunque ad una richiesta di interruzione non mascherabile (solo una richiesta del BUS-ESPANSIONE mediante il segnale $\overline{BBUSRQ}$ ha più alta priorità).

L'interruzione non mascherabile ha perciò priorità più alta sull'interruzione normale e genera automaticamente una esecuzione dell'istruzione RESTART alla locazione 0066H.

Questo tipo di interruzione è normalmente dedicato alle funzioni richiedenti una risposta rapidissima.

$\overline{BNMI}$ ha un massimo non oltrepassabile.

RICHIESTA E RICONOSCIMENTO INTERRUZIONE NON MASCHERABILE

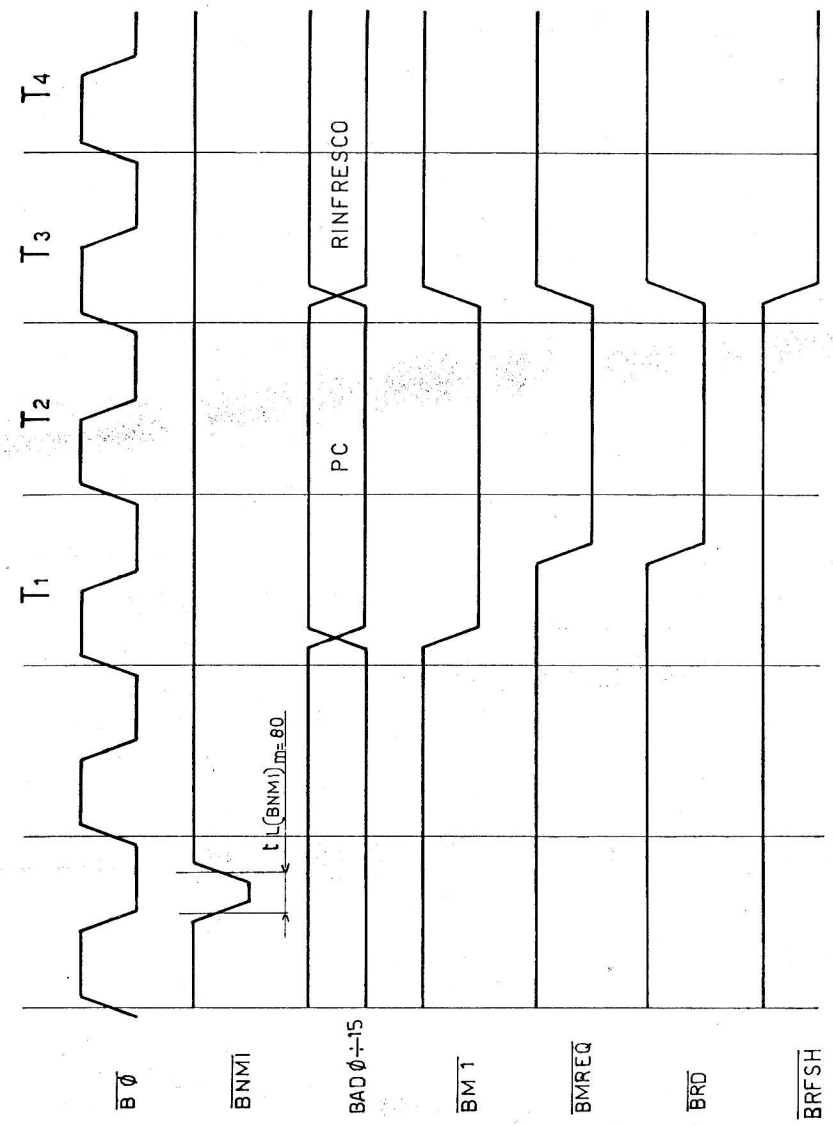


FIG. 1.11.7.1

ISTRUZIONE HALT

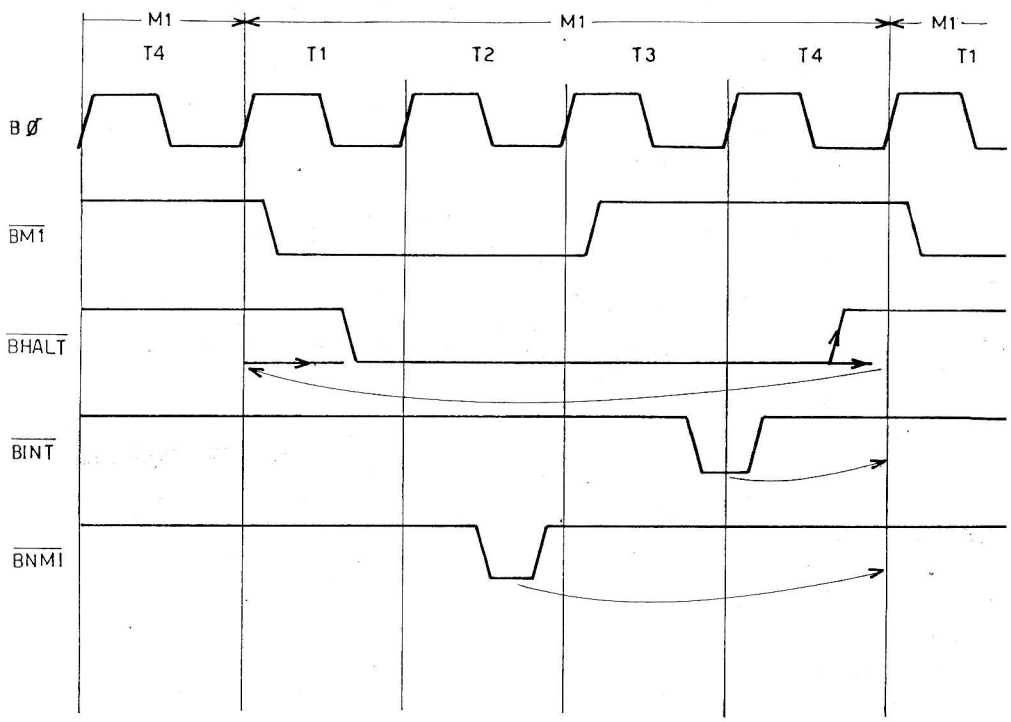


FIG. 1.11.8.1

1.11.8 ISTRUZIONE HALT

La figura 1.11.8.1 illustra la temporizzazione relativa all'esecuzione dell'istruzione HALT; questa istruzione si comporta nei confronti del Software come una sospensione dell'attività di elaborazione del CPU, mentre nei confronti dell'hardware riserva una funzionalità dinamica come sotto descritto.
Difatti ogni qualvolta venga eseguita una istruzione di HALT il CPU inizia l'esecuzione simulata di una serie di istruzioni NOP (nessuna operazione); questo permette che vengano rinfrescate le memorie dinamiche e che il CPU rimanga in allarme nei confronti di richieste di interruzione e di BUS-ESPANSIONE.

La linea $\overline{\text{BINT}}$ è sempre campionata con il fronte positivo del clock durante T4; se viene trovata attuata (con abilitato il Flip-Flop di sensibilizzazione all'interruzione), si attua un riconoscimento di richiesta di interruzione come illustrato nei capitoli 1.11.6.
La stessa cosa vale per la linea $\overline{\text{BNMI}}$ con le debite osservazioni relative al caso particolare (vedasi cap. 1.11.7).
Se entrambi i segnali sono ricevuti contemporaneamente, quello relativo alla linea $\overline{\text{BNMI}}$ sarà servito per primo.

Al ritorno dal sottoprogramma di servizio verrà eseguita l'istruzione seguente all'istruzione di HALT; perciò il riconoscimento di una interruzione ha due significati: il primo di attuare un programma opportuno di servizio; il secondo di procedere nell'esecuzione del programma che si era fermato all'istruzione di HALT.

La linea $\overline{\text{BHALT}}$ diventa attiva durante l'esecuzione iterativa del NOP evidenziando all'esterno questa condizione o stato del processore.

1.12 COMPOSIZIONE MODULO CLZ80

Il modulo CLZ80 viene fornito in 6 versioni diverse in funzione del tipo di memoria RAM e EPROM montata sul modulo stesso.

La tabella 1.12.I evidenzia la sigla particolare del modulo in funzione dei due parametri accennati.

Per esempio il modello CLZ80-4 può considerarsi il più povero ed il modello CLZ80 - 16/8 il più completo.

Il modulo CLZ80 viene fornito con una serie di ponticelli già prestampati che danno la composizione - standard con le prestazioni evidenziate dalla tabella 1.12.II .

Nel caso di diverse prestazioni l'utente deve incidere (per aprirli) i ponticelli prestampati ed utilizzare opportunamente (secondo le indicazioni dei paragrafi precedenti) dei ponticelli di filo rigido montati su colonnine inserite negli alloggiamenti specifici.

Queste colonnine possono venir fornite, su richiesta, dalla SGS-ATES .

TABELLA 1.12.I

SIGLA COMPLETA CLZ8 ../..		Capacità RAM K BYTES	
		4	16
CAPACITA'	Ø	CLZ80-4	CLZ80-16
EPROM	1K (M0-Z)	CLZ80-4/2	CLZ80-16/2
KBYTES	8K (O.S)	CLZ80-4/8	CLZ80-16/8

TABELLA 1.12.II

FUNZIONE	PONTICELLI PRESTAMPATI
EPROM = 2708	1 - 3 - 6
PARTIZIONE EPROM 60 ÷ 64 K	9 - 18 23 - 30
RAM = M4116	42 - 43 - 44
PARTIZIONE RAM 0 ÷ 16 K	11 - 31
CODICE SELETTORE PERIFERICHE Ø ÷ 31	32 - 36 - 37
UNITA' SERIALE LINEA 20 mA LOOP	54 - 58 - 65
UNITA' SERIALE BAUD = 110	52 - 46
READER TTY INSERITO	56
CASSETTE INSERITE	55 - 61 - 62

ASSEMBLER

INDICE

			PAG.
3.1	INTRODUZIONE		3.1-1÷2
3.2	CARATTERISTICHE DELL'ASSEMBLER		3.2-1
	3.2.1	Passo 1	3.2-1÷3
	3.2.2	Passo 2	3.2-4÷5
	3.2.3	Passo 3	3.2-5÷6
	3.2.4	Passo CTRL/C	3.2-6
	3.2.5	Passo X	3.2-6
3.3	CARATTERI LEGALI		3.3-1
	3.3.1	Caratteri Letterali	3.3-1
	3.3.2	Caratteri Numerici	3.3-1
	3.3.3	Caratteri Speciali	3.3-2÷3
3.4	FORMATO DEGLI STATEMENTS		3.4-1÷4
	3.4.1	Label	3.4-4÷6
	3.4.2	Operatore	3.4-6÷7
	3.4.3	Operando	3.4-7÷8
	3.4.4	Commento	3.4-8÷9
	3.4.5	Formati Legali	3.4-9
3.5	ISTRUZIONI		3.5-1
	3.5.1	IX+OFF, IY+OFF	3.5-1÷4
	3.5.2	NN	3.5-4÷6
	3.5.3	N	3.5-6÷8
	3.5.4	DIS1, DIS2	3.5-8

			Pag.
3.6	PSEUDO - ISTRUZIONI		3.6-1
	3.6.1	ORG XX	3.6-1÷3
	3.6.2	END	3.6-3÷4
	3.6.3	EQU	3.6-4÷6
	3.6.4	DEFB X	3.6-6÷7
	3.6.5	DEFW XY	3.6-8÷9
	3.6.6	DEFS XX	3.6-9÷10
	3.6.7	DEFM Stringa	3.6-10÷12
3.7	FORMATO DELLA LISTA		3.7-1÷2
	3.7.1	Campo errore	3.7-2
	3.7.2	Campo contatore di riga	3.7-3
	3.7.3	Campo indirizzo	3.7-3
	3.7.4	Campo contenuto	3.7-4
	3.7.5	Campo source	3.7-5
3.8	FORMATO DEL BINARIO		3.8-1÷2
3.9	USO DELL'ASSEMBLER		3.9-1÷3
3.10	ERRORI		3.10-1÷5
APPENDICE A		Istruzioni Legali	3.11-1÷13
APPENDICE B		Pagina di Lista	3.11-14

3.1 INTRODUZIONE

Lo scopo di questo capitolo è quello di mettere in grado gli utenti del CLZ80 di usare il programma ASSEMBLER nel miglior modo possibile, sfruttando tutti i vantaggi che questo programma mette a disposizione. Uno dei presupposti richiesti al lettore è quello di conoscere già il set delle istruzioni del CPU Z80 che è descritto nel capitolo 2 del manuale. Pertanto prima di proseguire la lettura di questo capitolo è consigliabile aver letto il capitolo precedente, in quanto eventuali esempi di istruzioni faranno riferimento a quel capitolo.

Per facilitare la comprensione di questo capitolo è utile chiarire il significato di alcuni termini che verranno usati in seguito:

STATEMENT: Con questo termine si indica abitualmente una serie di caratteri che formano una istruzione.

LABEL: Con questo termine si indica abitualmente il nome che identifica un determinato statement.

SOURCE: Con questo termine si indica abitualmente l'insieme degli statements che formano un programma. Questi statements sono scritti in linguaggio simbolico su nastro di carta o su nastro magnetico.

I numeri che vengono usati in questo capitolo possono essere di due tipi:

1. Decimali : in questo caso vengono indicati con il numero seguito da un punto: 12.
2. Esadecimali: in questo caso vengono indicati con il numero seguito dalla lettera H: 12H .

3.2 CARATTERISTICHE DELL'ASSEMBLER

L'Assembler è un programma a tre passi. Le funzioni e lo scopo di ciascun passo si possono vedere nei paragrafi che seguono.

3.2.1 Passo 1

Questo passo viene eseguito leggendo il nastro contenente il source. Durante questa lettura viene fatta una indagine accurata di tutti gli statements e vengono segnalati, con un formato che vedremo in seguito, gli eventuali errori trovati. Il ritrovamento di eventuali errori non blocca la compilazione, ma questa viene continuata fino alla fine del nastro per permettere all'utente di avere un quadro completo degli eventuali errori. Durante questo passo viene creata in memoria (nella parte di memoria RAM) una tabella contenente tutte le labels incontrate e l'indirizzo relativo ad ognuna. Per il calcolo dell'indirizzo a cui corrisponde ogni label l'assembler fa una indagine preliminare di ogni statement per vedere su quanti byte deve assemblare l'istruzione. Se per esempio incontra una istruzione del tipo XOR A sa che verrà compilata su un solo byte, mentre se incontra BIT 1, (HL) sa che richiede 2 byte.

In questo modo, sommando istruzione per istruzione i byte necessari per assemblarle, sa in qualsiasi punto del programma quale è l'indirizzo relativo ad ogni label.
Come vedremo in seguito, ogni label può essere al massimo di 6 caratteri, perciò l'assembler richiede 8 byte per ogni label che deve mettere nella tabella.
I primi 6 servono per memorizzare in codice ASCII il nome della label (1 byte per ogni carattere) gli ultimi due contengono rispettivamente la parte bassa e la parte alta dell'indirizzo corrispondente.
Il numero massimo di label che l'assembler accetta dipende da quanta RAM c'è a disposizione.
L'assembler prima di iniziare il passo 1 fa un'indagine per vedere se la scheda ha montato le memorie da 4 K oppure quelle da 16 K.
Vediamo nei due casi il numero massimo di label accettate dall'assembler.

A) 4 K di RAM

Numero di byte disponibili = 4096.-1024. = 3072.
(1024 byte sono usati dall'assembler come SCRATCHPAD-RAM)
Numero label = 3072.:8. = 384.

B) 16 K di RAM

Numero di byte disponibili = 16384.-1024. = 15360.
Numero label = 15360.:8.=1920.

Questi calcoli valgono qualora l'assembler sia residente su ROM.
Nel caso l'assembler risieda su RAM bisogna avere a disposizione obbligatoriamente 16 K di RAM e le label accettate dall'assembler sono:

- Numero di byte disponibili = 16384.-5120. = 11264.
- Numero label = 11264.:8.= 1408.

Se si tenta di compilare un programma con un numero di label superiore al massimo accettato viene data una segnalazione di errore e viene interrotta la compilazione, in quanto l'assembler non può garantire la corretta compilazione di tutti gli statements.

3.2.2 Passo 2

Questo passo può essere eseguito correttamente solo se è già stato eseguito il passo 1. Durante l'esecuzione di questo passo viene riletto il nastro contenente il source e come risultato si ottiene un nastro in formato binario contenente la compilazione in codice macchina di tutte le istruzioni contenute nel source. E' indispensabile aver già fatto il passo 1, in quanto, se il source contiene delle istruzioni che fanno riferimento a delle labels (per esempio: JP TIØ; LD A, (SAVE); JR Z, XX ecc.), l'assembler deve aver accessibile e già completa la tabella delle labels (creata dal passo 1) per sapere a quale indirizzo fa riferimento ciascuna label. Durante il passo 2 vengono anche segnalati gli eventuali errori che il passo 1 non è in grado di riconoscere e più precisamente i riferimenti illegali. Per capire cosa si intende per riferimento illegale vediamo un esempio:

```
JP    XX
```

Se la label XX non esiste nel source, il passo 1 non la può inserire nella tabella delle labels e quando il passo 2 cerca di compilarla, non trova l'indirizzo corrispondente a XX, perciò non essendo in grado di assemblare l'istruzione dà una segnalazione di riferimento illegale.

Il passo 1 non dà l'errore perchè la label XX si può trovare nella parte di nastro che non ha ancora control lato. Questi errori che trova il passo 2 non vengono dati durante l'esecuzione del passo e con il formato standard, ma vengono dati come un numero cumulativo alla fine del passo. Questo perchè,se viene scelta come periferica di uscita del nastro binario la teletype, verrebbero perforati gli errori in mezzo al binario rovinando perciò il nastro binario.

3.2.3 Passo 3

Questo passo può essere eseguito correttamente solo se è già stato eseguito il passo 1. Le sequenze legali dei vari passi sono perciò solo due:

Passo 1, Passo 2, Passo 3

oppure

Passo 1, Passo 3, Passo 2

Durante l'esecuzione del passo 3 viene riletto il nastro che contiene il source e come risultato si ottiene una lista che oltre alle istruzioni,contiene tutte le informazioni necessarie per il debugging del programma. Vengono rilevati gli stessi errori che rileva

il passo 2 (riferimenti illegali) e vengono segnalati sulla lista con una lettera che precede la riga nella quale si trovano. Oltre ai riferimenti illegali vengono segnalati anche gli errori che ha trovato il passo 1 sempre con una lettera (che identifica il tipo di errore) che precede la riga in cui si trovano . Gli errori non vengono segnalati nel formato standard per non avere delle pagine più lunghe dalle altre nel caso si trovino degli errori perchè la segnalazione standard comporta una riga di scrittura.

3.2.4 Passo CTRL/C

Questo passo permette di ripassare il controllo al MDL. Viene utilizzato quando si finisce di usare il programma assembler e si vuole utilizzare qualche altro programma.

3.2.5 Passo X

Questo passo permette di cambiare le periferiche di ingresso e di uscita in quanto fa ripartire il programma assembler dall'inizio, quando cioè chiede quali sono le periferiche da utilizzare.

3.3 CARATTERI LEGALI

Il programma assembler accetta sul nastro del source i caratteri in codice ASCII . Questi caratteri possono essere di 7 oppure 8 bit (ci può cioè essere o no il bit di parità) in quanto il programma toglie sempre il bit più significativo per rendere tutti i nastri compatibili. Non tutti i caratteri del codice ASCII sono legali per l'assembler, alcuni di questi non possono venire utilizzati.

3.3.1 Caratteri letterali

Di questi caratteri sono legali solamente le lettere maiuscole, quelle cioè che vanno da A (codice ASCII di 7 bit = = 41H;codice ASCII di 8 bit = C1H)fino a Z (codice ASCII di 7 bit = 5AH;codice ASCII di 8 bit = DAH).Le lettere minuscole vengono accettate e considerate legali solo nella zona del commento.

3.3.2 Caratteri numerici

Questi caratteri sono considerati tutti legali e sono quelli che vanno da Ø (codice ASCII di 7 bit = 3ØH; codice ASCII di 8 bit = BØH) fino a 9 (codice ASCII di 7 bit = 39H; codice ASCII di 8 bit = B9H).

3.3.3 Caratteri speciali

Di questi caratteri solo alcuni sono considerati legali e sono quelli elencati nella seguente tabella:

Carattere	Nome	Cod. ASCII 7 bit	Cod. ASCII 8 bit	Funzione
Return	Ritorno carrello	ØD	8D	Indicano la fine dello
Line - Feed	Avanzam. carta	ØA	8A	statement
:	Due Punti	3A	BA	Ind.la fine di 1 label
TAB	Tabul. orizz.	Ø9	89	Dividono i vari campi
Space	Spazio	2Ø	AØ	di uno statement
(	parent.sinistra	28	A8	Indicano una operazione
)	" destra	29	A9	"indiretta"
,	virgola	2C	AC	divide i due operanti delle istruzioni
;	punto-virgola	3B	BB	indica l'inizio di un commento
+	più	2B	AB	operatori per operazioni aritmetiche
-	meno	2D	AD	
.	punto	2E	AE	indica che un numero è decimale
'	apostrofo	27	A7	indicatore singolo di carattere ASCII
"	virgoletta	22	A2	indicatore doppio di carattere ASCII

Tutti i caratteri speciali che non compaiono nella tabella sono considerati illegali e accettati solo come commento.

Durante la lettura del nastro del source vengono ignorati completamente i caratteri blank (codice ASCII = ØH) e Rubout (codice ASCII = 7FH opp. FFH). L'eventuale ritrovamento di uno di questi due caratteri da parte dell'Assembler non dà origine a nessuna segnalazione di errore e non preclude il corretto assemblaggio del nastro.

3.4 FORMATO DEGLI STATEMENTS

Il nastro del source è formato da una serie di righe di scrittura in codice ASCII. Ogni riga corrisponde ad uno statement e può essere al massimo di 72. caratteri. Se l'Assembler incontra, durante la compilazione degli statements più lunghi di 72. caratteri non dà nessuna segnalazione di errore, ma si limita ad ignorare i caratteri dal settantatreesimo in poi.
Questo perchè quasi sempre i caratteri oltre i 72. accettati fanno parte del commento e di conseguenza anche se vengono ignorati non precludono la corretta compilazione del programma.
Ogni statement deve terminare con i caratteri RETURN e LINE-FEED nella sequenza in cui sono scritti. Sono accettati anche degli statement vuoti purchè contengano almeno i due caratteri terminali (RETURN e LINE-FEED). Questi statements non danno origine a nessuna compilazione, se ne terrà conto solo sulla lista dove verrà stampata una riga vuota che verrà però conteggiata. Ogni statement deve essere completato entro la riga, non vengono perciò accettati statements su più righe.
Ogni statement può essere diviso in quattro campi distinti:

1) LABEL
2) OPERATORE
3) OPERANDO
4) COMMENTO

Di questi quattro campi due (LABEL e COMMENTO) non sono obbligatori e si possono omettere senza incorrere in errori di compilazione . L'OPERATORE si scrive sempre mentre l'OPERANDO si può omettere se l'istruzione di cui corrisponde l'Operatore non lo richiede.

Per chiarire questo concetto vediamo alcune istruzioni con solo operatore ed altre con operatore e operando:

OPERATORE	OPERATORE	OPERANDO
SCF	OR	A
HALT	CP	B
EXX	AND	(HL)
RLA	RET	NZ
OTIR	RL	A

In parecchi casi l'operatore richiede due operandi; entrambi fanno parte del campo OPERANDO. Vediamo alcuni esempi:

OPERATORE	OPERANDI
LD	A,B
BIT	1,C
ADD	HL, HL
SET	2, (HL)
EX	DE, HL

I quattro campi in cui viene diviso ogni statement sono separati tra di loro da spazi oppure da tab.

La separazione minima, per non incorrere in errori di compilazione, è di uno spazio o un tab; se ci sono più spazi o più tab la separazione è considerata legale; lo stesso se ci sono spazi e tab assieme. Per avere una lista ordinata e di conseguenza facilmente leggibile, si consiglia di scrivere il source nel seguente modo:

LABEL (tab) OPERATORE (tab) OPERANDO (1 o 2 tab) COMMENTO

dopo il campo OPERANDO si mette un tab se i caratteri che compongono l'operando sono 8 o più di 8, si mettono due tab se sono meno di 8.

Nel caso non ci sia la LABEL, si mette un tab prima dell'operatore; se non c'è l'operando si mettono 3 tab dopo l'operatore. Questo permette di avere una lista con i vari campi perfettamente incolonnati e che iniziano sempre allo stesso punto è cioè:

COLONNA	1	9	17	33
CAMPI	LABEL	OPERATORE	OPERANDO	COMMENTO

Nella tabella che segue sono descritti alcuni esempi di statement scritti con il criterio sopracitato. Il carattere ⌊⌋ sta ad indicare un tab:

```
TESTA: LD      A,B          ; CARICO REG.A
       NOP                  ; NO-OPERATION
T1 :   BIT     1,(IX+2CH)   ; TEST BIT 1
       JR      Z, TESTA     ; SALTO SE Ø
```

Vediamo ora in dettaglio le caratteristiche e le limitazioni dei vari campi.

3.4.1 Label

La label è un nome creato dall'utente che serve per identificare un ben determinato statement. Durante il passo 1, della compilazione ad ogni label, viene assegnato un valore, che corrisponde alla locazione di memoria in cui andrà a finire l'istruzione cui fa riferimento la label, e viene memorizzato nella tabella dei simboli. Ogni label deve indicare un solo statement, perciò all'interno di un programma non ci possono essere label uguali tra di loro. Il nome di una label può essere uno qualsiasi purchè risponda alle seguenti restrizioni:

A) non deve superare in lunghezza i 6 caratteri.

B) deve essere composto solo da caratteri Alfanumerici; sono cioè legali tutte le lettere da A a Z e tutti

i numeri da Ø a 9. Può iniziare indifferentemente con una lettera o con un numero.

C) deve finire sempre con il carattere: (due punti). Questo carattere non è compreso nel conteggio dei caratteri perciò si può avere un massimo di 6 caratteri più i : .

D) Non può terminare con la lettera H se i caratteri che la precedono formano un numero esadecimale. Per chiarire il concetto vediamo un esempio:

AB6H:

Questa label è considerata illegale perchè il compilatore non può sapere se è veramente una label oppure se è il numero esadecimale AB6. Se infatti il programmatore la usa come label e la richiama ad esempio con un Jump:

JP AB6H

il compilatore non può sapere se il programmatore vuole saltare all'indirizzo di memoria AB6 in esadecimale oppure all'indirizzo cui corrisponde la label AB6H. Per evitare questa incongruenza non si può usare come label un nome che termina con H, a meno che non sia preceduto da caratteri non

esadecimali. Per esempio sono legali i seguenti nomi:

ABXH:

12H5:

123XYH:

La seguente tabella riporta alcuni nomi di labels illegali ed il motivo per cui lo sono:

Nome	Motivo dell'illegalità
ABC1234:	il numero dei caratteri supera il 6
AB $X:	contiene un carattere non alfanumerico "$'
12ACH:	termina con la lettera H preceduta da caratteri che formano un numero esadecimale
XYZ	non termina con il carattere :

3.4.2 Operatore

L'operatore è quella parte dello statement che stabilisce l'azione che compie l'istruzione. Tutti i tipi di operatori che formano il linguaggio macchina dello Z 80 sono descritti nel capitolo 2; a questi si sommano alcuni operatori che servono per descrivere delle pseudo-istruzioni.

Questi pseudo-operatori sono i seguenti:

ORG
DEFB
DEFW
DEFS
DEFM
END
EQU

e verranno trattati dettagliatamente nel paragrafo 3.6.

L'operatore viene preceduto da una label qualora in altri punti del programma si voglia fare riferimento a questo statement, e viene seguito da uno o due operandi qualora il tipo di istruzione che si vuole usare lo richieda. Due delle pseudo-istruzioni viste e precisamente ORG e END non possono essere precedute da una label in quanto non danno origine a nessun codice binario. Il compilatore considera terminato il campo operatore quando incontra uno spazio oppure un tab.

3.4.3 Operando

L'operando contiene le variabili che l'operatore deve manipolare o valutare.

Non sempre il campo operando deve essere descritto,

ci sono infatti delle istruzioni e delle pseudo-istruzioni che non lo richiedono. Per sapere se una certa istruzione richiede l 'operando o no, bisogna leggere il capitolo 2 che riporta tutte le istruzioni nella loro forma corretta. Per quanto riguarda le pseudo-istruzioni ce n'è solo una che in alcuni casi non richiede l'operando ed è: END.

Il campo operando può contenere uno o due operandi. In quest'ultimo caso i due operandi vengono divisi tra loro da una virgola. Prima e dopo la virgola non vengono accettati nè spazi nè tab.

L'operando si considera terminato quando si incontra uno spazio o un tab. E' importante ricordare che il campo operando può esistere solo se esiste il campo operatore; un campo operando senza il relativo campo operatore è considerato illegale.

3.4.4 Commento

Il commento inizia dopo il campo operando e finisce con il termine dello statement. Deve iniziare sempre con il carattere; (punto e virgola) e si considera finito quando si incontra il carattere return. All'interno del campo del commento sono considerati validi tutti i caratteri

del codice ASCII ad eccezione del return (che serve per terminarlo) e del Line-Feed. Questa parte dello statement è l'unica che non influisce sulla compilazione; infatti se una istruzione è seguita o no dal commento la compilazione è la stessa. Se il commento supera il 72esimo carattere della riga si può proseguire nello statement successivo scrivendo come primo carattere un punto e virgola (;).

3.4.5 Formati legali

Uno statement è formato da diversi campi, ma non sempre questi campi sono tutti necessari. La seguente tabella riporta tutte le combinazioni tra i vari campi che sono considerati legali dal compilatore:

1)	LABEL:	OPERATORE	OPERANDO	COMMENTO
2)		OPERATORE	OPERANDO	COMMENTO
3)		OPERATORE	OPERANDO	
4)		OPERATORE		
5)	LABEL:	OPERATORE	OPERANDO	
6)	LABEL :	OPERATORE		
7)				COMMENTO
8)		OPERATORE		COMMENTO
9)	LABEL:	OPERATORE		COMMENTO

tutte le altre combinazioni sono considerate illegali.
In particolar modo bisogna ricordare che è illegale una label seguita da un commento perchè la label deve sempre far riferimento ad una istruzione.

3.5 ISTRUZIONI

Tutte le istruzioni legali accettate dall'Assembler sono riportate nell'appendice A in fondo al capitolo. Gran parte di queste sono fisse come scrittura; di seguito sono riportati alcuni esempi:

```
ADC     A, (HL)
LD      A, B
INC     IX
JP      (IY)
```

Come si vede chiaramente, queste istruzioni possono essere scritte nel solo modo riportato. Le due parentesi stanno ad indicare che il valore che contengono non è l'operando, bensì l'indirizzo dell'operando.

Ci sono altre istruzioni il cui formato non è standard, ma può essere scritto in diversi modi:

3.5.1 IX+OFF , IY + OFF

In tutte le istruzioni dove compaiono come operandi i registri IX e IY più l'offset, quest'ultimo può essere descritto in diversi modi:

1. Decimale

   In questo caso il numero deve essere seguito dal punto . Affinchè il numero venga compilato correttamente deve stare tra - 128. e + 127.

Viene data una segnalazione di errore qualora il numero in valore assoluto supera il contenuto di un Byte e cioè 255.

Bisogna stare attenti che se il numero è compreso tra + 127. e +255.oppure tra - 128. e - 255. l'offset cambia di segno.

Se per esempio di scrive:

```
LD  A, (IX + 128.)
```

viene compilato come se si scrivesse:

```
LD  A, (IX - 128.)
```

perchè + 128. vale in esadecimale ØØ8ØH mentre - 128. vale FF8ØH, perciò considerando il solo byte basso si ottiene lo stesso valore.

2. Esadecimale

In questo caso il numero deve essere seguito dalla lettera H. Viene data una segnalazione di errore se il numero supera il contenuto di un byte e cioè FFH. Vale il discorso fatto per i numeri decimali se il numero è compreso tra + 7 FH e +FFH oppure tra - 80H e - FFH. Infatti scrivere:

```
LD  A, (IX + 8ØH)
```

equivale a scrivere

```
LD  A, (IX - 8ØH)
```

3. Label

In questo caso è indispensabile che in qualche punto del programma venga assegnato un valore alla label con la pseudo-istruzione EQU .

Vediamo un esempio:

```
ABC:  EQU  100.
      LD   A, (IY - ABC)
```

I due statements equivalgono al seguente:

```
LD    A, (IY - 100.)
```

Il valore può essere assegnato alla label in decimale oppure in esadecimale e deve rispettare le regole viste nei paragrafi 3.5.1.1 e 3.5.1.2 . L'offset può essere anche omesso; in questo caso il compilatore non dà nessuna segnalazione di errore e considera l'offset uguale a zero.

Scrivere per esempio:

```
LD    A, (IX)
```

equivale a scrivere:

```
LD    A, (IX + Ø.)
```

Nella tabella seguente sono riportati alcuni esempi corretti:


```
      BIT    1, (IX + 1Ø.)
      LD     B, (IX - 6BH)
      AND    (IY + OFFSET)
      RRC    (IY - XYZH)
      SET    6, (IX)
OFFSET: EQU  100.
XYZH:   EQU  - 5H
```

3.5.2 NN

NN rappresenta un valore che può essere contenuto da una parola di 16 bit.

Questo numero può essere espresso in decimale, esadecimale oppure con una label.

Se il numero viene espresso come un valore decimale può andare da Ø. a 65535., se viene espresso in esadecimale può andare da ØH a FFFFH, infine se viene espresso con una label questa deve essere obbligatoriamente descritta in qualche punto del programma.

Un caso particolare è rappresentato da due caratteri preceduti dalle due virgolette. Vediamo un esempio:

```
LD    HL, "AB
```

Questo tipo di scrittura sta ad indicare che NN è formato da due byte che contengono il codice ASCII a 7 bit dei due caratteri che seguono le due virgolette; in questo caso A e B. Il primo carattere(in ordine di scrittura) rappresenta la parte alta di NN, mentre il secondo rappresenta la parte bassa. Lo stesso risultato si può ottenere scrivendo:

```
LD    HL, 4142H
```

dove 41 e 42 è il valore in esadecimale del codice ASCII dei caratteri A e B. Il fatto di poter scrivere direttamente i due caratteri è un vantaggio per il programmatore che, in caso contrario, dovrebbe scrivere i codici ASCII dei caratteri. Sono considerati legali tutti i caratteri ad eccezione del RETURN e del LINE-FEED.

NN può essere preceduto dal segno - , che sta ad indicare che il valore compilato non è quello rappresentato da NN, bensì il suo complemento a due. Se NN è rappresentato da una label che corrisponde ad un valore negativo, ed è preceduto dal segno - , il valore compilato diventa positivo. Per chiarire il concetto vediamo un esempio:

```
ABCD:  EQU    - 1ØØØH
       LD     HL, - ABCD
```

Alla label ABCD corrisponde il valore esadecimale - 1ØØØH = FØØØH, ma il valore compilato è

- ABCD = - (-1ØØØH) = 1ØØØH

Nella tabella seguente sono riportati alcuni esempi corretti utilizzando NN:

```
BEGIN:  LD     HL, 1359 .
ROUT:   LD     BC, -674AH
        LD     SP, STACK
        JP     BEGIN
STACK:  DEFS   1Ø.
        CALL   ROUT
        LD     DE, "12
```

3.5.3 N

N rappresenta un valore che può essere contenuto da un byte di 8 bit.

Questo numero può essere espresso in decimale, esadecimale oppure con una label.

Se viene espresso in decimale, il suo valore può andare da Ø. a 255. , se viene espresso in esadecimale può andare da ØH a FFH, infine se viene espresso con una label, questa deve essere obbligatoriamente descritta in qualche punto del programma.

Un caso particolare è rappresentato da un carattere preceduto dall'apostrofo. Vediamo un esempio:

```
LD   A, '1
```

Questo tipo di scrittura sta ad indicare che il valore di N è il codice ASCII a 7 bit del carattere che segue l'apostrofo. Lo stesso risultato si può ottenere scrivendo:

```
LD   A, 31H
```

dove 31H è il valore esadecimale del codice ASCII del carattere 1. Sono considerati legali tutti i caratteri ad eccezione del RETURN e del LINE-FEED.

N può essere preceduto dal segno -; in questo caso il valore assemblato non è N, bensì il suo completamento a 2. Nella tabella seguente sono riportati alcuni esempi corretti che utilizzano N:


```
        LD      A, 10.
        LD      E, FCH
        CP      'A
        OR      - 51
        SUB     LABEL
LABEL:  EQU     76H
```

3.5.4 DIS1, DIS2

DIS1 e DIS2 seguono le stesse regole viste per NN.

3.6 PSEUDO-ISTRUZIONI

Le pseudo-istruzioni sono degli statements che non danno origine una volta compilati a dei codici macchina che vengono eseguiti dallo Z80, ma servono per dare delle direttive all'Assembler. Due di questi (ORG e END) devono essere descritti in ogni programma, gli altri vengono usati solo se servono.

3.6.1 ORG XX

Questa pseudo-istruzione ha lo scopo di indicare all'assembler l'indirizzo dal quale deve cominciare la compilazione degli statements che seguono. Se per una dimenticanza del programmatore questa pseudo-istruzione non compare, l'assembler inizia la compilazione dalla locazione di indirizzo Ø.

XX è un numero che può essere rappresentato con 16 bits e segue le stesse regole e le stesse limitazioni viste in precedenza per NN.

All'interno di un programma ci possono essere altre pseudo-istruzioni ORG; in questo caso l'assembler, dopo averle incontrate, prosegue la compilazione degli statements che seguono iniziando dall'indirizzo descritto con la pseudo-istruzione.

Per fare un esempio, si può vedere un caso che si incontra abbastanza facilmente; quello cioè di dover caricare in memoria delle costanti non contigue al programma che le usa:

```
        ORG     66H
        JP      NM INT
        ORG     2ØØH
START : LD      SP, START
```

In questo esempio il programma vero e proprio inizia dalla locazione 200H in poi, però quando si carica in memoria bisogna caricare anche a partire dall'indirizzo 66H, il salto per servire degli eventuali interrupt non mascherabili.

L'assembler durante la compilazione incontra per primo l'indirizzo 66H e di conseguenza compila l'istruzione che segue (JP NMINT) nelle locazioni 66H, 67H e 68H; il successivo statement è un altro ORG e di conseguenza le istruzioni che seguono vengono compilate non da 69H in poi, bensì da 200 H in poi. Questa pseudo-istruzione non può essere preceduta dalla label, in quanto non dà origine a nessun codice binario e di conseguenza a nessun indirizzo cui la label si possa riferire.

Per avere un listing comprensibile e chiaro è consigliabile scrivere ORG nel campo operatore e XX nel campo operando ; il tutto può essere seguito da un eventuale commento. Nella tabella che segue sono riportati alcuni esempi corretti:

```
        ORG     100H
        ORG     1000.
        ORG     START
START:  EQU     12ØØH
```

3.6.2 END

Questa pseudo-istruzione ha lo scopo di informare l'assembler che il programma da compilare è finito e di conseguenza tutti gli statements che si trovano dopo vengono ignorati. La pseudo-istruzione END può essere seguita da un operatore che rappresenta un indirizzo a 16 bits; questo indirizzo è quello di auto-start. Se viene descritto significa che alla fine del caricamento in memoria (mediante il Loader) del nastro binario, il programma va subito in esecuzione a partire dall'indirizzo che segue la pseudo-istruzione END. Se invece si descrive solo END, alla fine del caricamento del nastro binario il controllo verrà ridato dal LOADER al DEBUGGER. L'operatore che segue la pseudo-istruzione END deve seguire le stesse regole e le stesse restrizioni viste per NN.

Per avere un listing comprensibile e chiaro è consigliabile scrivere END nel campo operatore e l'eventuale indirizzo di auto-start nel campo operando ; il tutto può essere seguito da un commento.

Nello statement che contiene le pseudo-istruzioni END non si deve mettere la label perchè non dà origine a nessun codice binario e di conseguenza a nessun indirizzo cui può fare riferimento la label.

Nella tabella che segue sono riportati alcuni esempi corretti:

```
        END
        END     START
        END     2ØØH
```

3.6.3 EQU

Questa pseudo-istruzione viene utilizzata per assegnare ad una label un certo valore. La sua forma corretta è la seguente:

```
LABEL:  EQU   YY
```

Questo tipo di scrittura indica che nella tabella dei simboli la label non sarà seguita dall'indirizzo a cui si trova, ma bensì dal valore YY.

All'interno del programma ogni volta che si descrive la label non si farà riferimento al suo indirizzo, ma al valore che il programmatore gli ha assegnato. La variabile YY può essere descritta come un valore decimale oppure come un valore esadecimale e può rappresentare un numero contenuto in 8 bits oppure in 16 bits. Nel caso si assegni alla label un valore superiore a 255. o FFH ci sono delle limitazioni; infatti non si può fare riferimento a quella label in una istruzione che opera sui bytes. Vediamo un esempio illegale:

```
ABCD:   EQU   1ØØØH
        LD    A, ABCD
```

infatti l'assembler si comporta come se incontrasse il seguente statement :

```
LD   A, 1ØØØH
```

e perciò dà una segnalazione di errore.

E' invece legale il contrario; cioè richiamare in una istruzione che opera sulle parole (16 bits) una label a cui è stato assegnato un valore inferiore a 255. o FFH.

Vediamo un esempio:

```
ABCD:   EQU   65H
LD      HL, ABCD
```

l'assembler si comporta come se incontrasse il seguente statement:

```
LD   HL, ØØ65H
```

perciò accetta lo statement.

Nella tabella che segue sono riportati alcuni esempi corretti:

```
XX:     EQU   1650.
ACD:    EQU   67H
YY:     EQU   1A6BH
```

3.6.4 DEFB X

Questa pseudo-istruzione permette al programmatore di assegnare il valore X ad un certo byte del programma; permette cioè di compilare delle costanti.

L'indirizzo a cui la costante X viene compilata è quello successivo all'ultimo statement incontrato dall'assembler prima di DEFB. Vediamo un esempio pratico:

```
        ORG     1ØØH
START:  LD      A,(XYZ)
        HALT
XYZ:    DEFB    55.
```

L'istruzione di LD viene compilata nelle locazioni 1ØØH, 1Ø1H e 1Ø2H, l'istruzione HALT nella locazione 1Ø3H e la pseudo-istruzione DEFB in quella successiva, cioè 1Ø4H. L'operando X può essere al massimo di 255. o FFH e segue le stesse regole e le stesse restrizioni viste per N. Per chiarezza della lista e compatibilità con le altre istruzioni è consigliabile descrivere DEFB nel campo operatore e X nel campo operando; il tutto può essere preceduto da una label e seguito da un commento.

Nella tabella che segue sono riportati alcuni esempi corretti:

```
        DEFB    10.
        DEFB    - 67H
        DEFB    XYZ
        DEFB    F6H
XYZ:    EQU     12H
```


3.6.5 DEFW XY

Questa pseudo-istruzione permette al programmatore di assegnare il valore Y ad un certo byte del programma ed il valore X al byte successivo. XY viene descritto come un unico valore e segue le stesse regole e le stesse restrizioni viste per NN. L'indirizzo a cui la costante XY viene compilata è quello successivo all'ultimo statement incontrato dall'assembler prima di DEFW, ed il successivo. La parte bassa della costante viene messa nella prima cella, la parte alta nella seguente. Vediamo un esempio:

```
        ORG     1ØØH
START:  LD      HL, XYZ
        HALT
XYZ:    DEFB    67A2H
```

L'istruzione LD viene compilata alla locazione 100H, 101H e 102H, l'istruzione HALT alla locazione 103H, la parte bassa della costante e cioè A 2H alla locazione 104H e la parte alta 67H alla locazione 1Ø5H.

E' consigliabile descrivere DEFW nel campo operatore e XY nel campo operando; il tutto può essere preceduto da una label e seguito da un commento.

La tabella che segue riporta alcuni esempi corretti:

```
        DEFW   10000.
        DEFW   123H
        DEFW   XYZ
        DEFW   - 1243.
XYZ:    EQU    674H
```

3.6.6 DEFS XX

Questa pseudo-istruzione ha lo scopo di riservare delle locazioni vuote nel programma. XX rappresenta il numero di byte che vengono compilati al valore di ØØH dall'assembler. XX è un numero contenuto da una parola a 16 bits e nella scrittura segue le stesse regole e le stesse restrizioni viste per NN. Se il valore di XX supera il numero 4. nel listing vengono rappresentate col valore di ØØH solo le prime quattro celle, le altre non vengono stampate per non appesantire troppo il listing, ma l'assembler le conteggia.

Vediamo un esempio:

```
        ORG    1ØØH
START:  LD     A, COST
BUFER:  DEFS   1ØØH
COST:   DEFB   76H
```


L'istruzione LD viene compilata nelle locazioni 100H, 101H e 102H, la prima locazione lasciata a ØØH ha come indirizzo 103H, l'ultima 202H, mentre la costante definita con DEFB viene compilata all'indirizzo 203H. Delle 1ØØH locazioni compilate a ØØH solo le prime 4. compaiono nel listing, le altre non vengono scritte, però ne viene tenuto conto; infatti DEFB viene compilato dopo 100H bytes. Nel binario invece compaiono tutte e 1ØØ H. E' consigliabile descrivere DEFS nel campo operatore e XX nel campo operando; il tutto può essere preceduto da una label e seguito da un commento.

La tabella che segue riporta alcuni esempi corretti:

```
        DEFS   10.
        DEFS   2ØØH
        DEFS   XYZ
XYZ:    EQU    120H
```

3.6.7 DEFM Stringa

Questa pseudo-istruzione permette di caricare in celle di memoria successive delle costanti che corrispondono al codice ASCII a 7 bit dei caratteri che formano la stringa. Il primo carattere della stringa viene utilizzato dall'Assembler come carattere delimitante perciò non viene caricato come costante. Dal secondo in poi, ogni carattere viene caricato in un byte di memoria.

L'indirizzo a cui viene compilato il primo carattere è quello successivo all'ultimo statement incontrato dall'assembler prima di DEFM.

La stringa viene considerata conclusa quando l'assembler incontra un carattere uguale al primo. Neppure questo ultimo carattere viene messo in memoria perchè è un carattere delimitante. Il carattere che indica l'inizio e la fine della stringa può essere uno qualsiasi ad eccezione di SPAZIO, TAB, RETURN, LINE-FEED.

I primi due non si possono usare perchè vengono utilizzati per dividere DEFM dalla stringa, gli altri due perchè indicano la fine dello statement.

Vediamo un esempio:

```
        ORG     1ØØH
START:  CALL TALK
        DEFM / ABC /
```

L'istruzione CALL viene assemblata nelle locazioni 100H, 101H, e 102H, il codice ASCII e 7 bits del carattere A e cioè 41H viene compilato nella locazione 103H, quello della B (42H) nella 104H e quello della C (43H) nella 105H. I due caratteri / non vengono compilati perchè fungono da caratteri delimitanti. La lunghezza della stringa è limitata solo dalla lunghezza massima di uno statement (72 caratteri). Se la stringa con

tiene più di 4. caratteri nel listing compaiono solo i primi 4. ; l'assembler tiene però conto degli altri.

E' importante ricordare che il carattere che viene usato come carattere delimitante, non può essere usato all'interno della stringa. La tabella che segue riporta alcuni esempi corretti.

```
        DEFM    / ASCII + /
        DEFM    A (R/L) A
        DEFM    @ESEMPIO DI DEFM @
        DEFM    1 76541
```

3.7 FORMATO DELLA LISTA

La lista è la stampa che si ottiene durante il passo 3 ed ha lo scopo di aiutare il programmatore durante la fase di debugging del programma. Sulla lista sono riportate tutte le informazioni che permettono al programmatore di controllare il programma, di sapere dove ogni istruzione viene caricata in memoria, di calcolare l'occupazione di memoria e altre cose. Il formato della lista si può vedere nella appendice B che riporta l'esempio di una pagina di lista.

All'inizio di ogni pagina della lista compare la scritta:

PAG. XXX

dove XXX rappresenta il contatore delle pagine. Alla fine di ogni pagina compare la scritta:

ERR. YYY

dove YYY è il numero di errori trovati dall'assembler. Questo numero non si riferisce agli errori trovati nella pagina, ma è un contatore cumulativo; se per esempio nella prima pagina vi sono 2 errori e nella seguente 3 errori, alla fine della prima pagina avrò la scritta:

ERR. ØØ2

mentre alla fine della seconda pagina avrò:

ERR. ØØ5

cioè il numero di tutti gli errori incontrati fino ad allora. Questo permette al programmatore di avere alla fine dell'ultima

pagina della lista il numero totale degli errori che ci sono nel programma.

Per facilitare la comprensione della lista è utile dividere la riga in diversi campi, che partendo da sinistra sono:

A) CAMPO ERRORE

B) CAMPO CONTATORE DI RIGA

C) CAMPO INDIRIZZO

D) CAMPO CONTENUTO

E) CAMPO SOURCE

Vediamo di seguito i vari campi e le loro caratteristiche.

3.7.1 Campo errore

Questo campo compare all'inizio della riga solamente quando durante l'assemblaggio viene riscontrato un errore nella riga stessa. E' formato da una sola lettera che indica il tipo di errore ed è diviso dal campo successivo da uno spazio. I vari tipi di errori riportati nel capitolo 3.1Ø sono formati da due lettere; nella lista compare solo la seconda delle due. Per fare un esempio il tipo di errore IC (carattere illegale) viene riportato nella lista solo con la lettera C. Se non viene trovato nessun errore nella riga, il campo errore è formato da uno spazio.

3.7.2 Campo contatore di riga

Questo campo segue il campo errore ed è formato da tre caratteri che formano un numero. Questo numero rappresenta il conteggio (in decimale) degli statements del programma e può arrivare fino a 999. Se il programma supera le 999. righe il conteggio riprende da Ø. Questo campo è diviso dal successivo da uno spazio.

3.7.3 Campo indirizzo

Questo campo segue il campo contatore di riga ed è formato da quattro caratteri esadecimali che rappresentano l'indirizzo di memoria a cui corrisponde il contenuto del campo che segue. Il primo valore del campo indirizzo viene fornito all'assembler della pseudo-istruzione ORG ; gli altri indirizzi degli statements si ottengono incrementando il precedente di 1,2, 3 o 4 a secondo che l'istruzione precedente venga compilata su 1,2, 3 o 4 byte. Se viene incontrato un nuovo ORG il campo indirizzo riprende il conteggio dal nuovo valore. Se all'inizio del programma non c'è la pseudo-istruzione ORG il conteggio inizia da ØØØØH. Questo campo è diviso dal successivo da uno spazio.

3.7.4 Campo contenuto

Questo campo contiene il codice macchina in esadecimale dell'istruzione rappresentata dallo statement. La lunghezza del campo contenuto varia a seconda del tipo di istruzione e può essere di 2,4,6 o 8 caratteri esadecimali a seconda che l'istruzione richieda 1, 2, 3, o 4 byte per essere assemblata; ogni byte viene descritto con due caratteri esadecimali. Per esempio l'istruzione:

```
INC     A
```

richiede un solo byte, perciò viene descritta con due soli caratteri:

```
3C
```

mentre l'istruzione :

```
BIT   Ø, (IX + 5H)
```

richiede 4 byte perciò viene descritta con 8 caratteri esadecimali:

```
DDCB0546
```

Questo campo è diviso dal successivo da un tab.

3.7.5 Campo Source

Questo campo è la riproduzione della statement che si si trova sul nastro del source. Questo campo permette al programmatore di vedere gli eventuali errori che si trovano nel source.

NOTA: Il campo indirizzo ed il campo contenuto possono non venire descritti qualora lo statement da compilare dia origine a nessun codice macchina. Per esempio se lo statement contiene solo il campo commento non si ha nessun codice macchina e perciò i campi sopracitati vengono sostituiti con degli spazi.

3.8. FORMATO DEL BINARIO

Il nastro binario si ottiene durante il passo 2 e viene perforato in codice ASCII.

Il formato è il seguente:

:NN IIII XXDD . . . . DDCC

:NN IIII XXDD . . . . DDCC

All'inizio di ogni riga di binario ci sono i caratteri RETURN, LINE-FEED e due punti (:).

I caratteri che seguono hanno il seguente significato:

NN Sono due caratteri che formano un numero esadecimale. Questi rappresentano il numero di byte di binario contenuti dalla riga.
Il massimo numero di byte che può contenere una riga è 10H.

IIII Sono quattro caratteri esadecimali che indicano l'indirizzo corrispondente al primo byte di dati della riga.

XX Sono due caratteri che vengono sempre messi a zero. Servono solo per compatibilità con i binari ottenuti con altri assembler.

DD..DD Sono diversi caratteri esadecimali che presi a due a due indicano il contenuto dei vari byte del binario.
Il numero di questi caratteri è sempre il doppio di NN perchè per rappresentare un byte in esadecimale sono necessari due caratteri in codice ASCII.

CC Sono due caratteri esadecimali che rappresentano il checksum.
Questo numero è ottenuto facendo la somma di tutti i

numeri della riga, prendendo i caratteri a due a due, partendo da NN, e facendo poi il complemento a due del risultato.
Si usa il complemento a due per facilitare il compito del loader.
E' infatti sufficiente che questi sommi tutti i numeri della riga, compreso il checksum e controlli che il risultato sia zero; se è diverso da zero c'è stato un errore di lettura.

Alla fine dell'ultima riga del binario viene aggiunto un RETURN ed un LINE-FEED.

Un caso particolare è rappresentato dall'AUTO-START la cui riga di binario è sempre l'ultima ed ha il seguente formato:

:00 IIII XXCC

Il loader la distingue dalle altre perchè il valore di NN è sempre zero, perciò interpreta il valore di IIII non come indirizzo di caricamento dei dati, ma come indirizzo di AUTO-START.

Se alla fine del binario c'è l'AUTO-START, il loader alla fine del caricamento dei dati in memoria esegue un JP all'indirizzo di AUTO-START.
In caso contrario ritorna il controllo di M0-Z.

3.9 USO DELL'ASSEMBLER

Per utilizzare l'assembler bisogna tenere presenti due possibilità:

1) se il programma risiede su ROM si fa partire mediante il comando A.
2) Se il programma risiede su nastro di carta o su nastro magnetico viene caricato mediante il LOADER. Alla fine del caricamento l'assembler va in esecuzione mediante l'autostart.

Una volta che l'assembler è partito, i comandi da dare sono gli stessi per i due tipi di assembler. Per prima cosa il programma pone all'utente una serie di domande che riguardano le periferiche di ingresso-uscita.
La prima domanda è:

IN (L/H/C)

e riguarda la periferica di ingresso dalla quale si vuole leggere il nastro contenente il source.

Le risposte possibili sono tre:

C = Registratore a cassetta

H = Lettore veloce

L = Lettore lento (Teletype)

Qualsiasi altro carattere viene interpretato dall'assembler come L .

La seconda domanda è:

BIN (L/C)

e riguarda la periferica di uscita dalla quale si vuole ottenere il nastro binario. Le risposte possibili sono due:

C = Registratore a cassetta

L = Perforatore lento (Teletype)

Qualsiasi altro carattere viene interpretato dall'assembler come L .

La terza domanda è:

LIS (L/C)

e riguarda la periferica di uscita dalla quale si vuole ottenere la stampa della lista.

Le risposte possibili sono due:

C = Registratore a cassetta

L = Teletype

Qualsiasi altro carattere viene interpretato dall'assembler come L .

A questo punto l'assembler chiede quale passo deve eseguire con la domanda:

PASS ?

Le risposte legali sono cinque:

1 = PASSO 1
2 = PASSO 2
3 = PASSO 3
M = PASSO CTRL/C
X = PASSO X

I vari passi sono già stati visti dettagliatamente nel paragrafo 3.2. Se si risponde alla domanda con un carattere diverso dai cinque visti, l'assembler rifà la domanda PASS?

NOTE : 1) I nastri magnetici si possono riavvolgere soltanto mentre l'assembler attende la risposta alla domanda PASS ?. Prima di dare la risposta bisogna far partire il registratore.

2) I nastri di carta vanno messi sul lettore prima di dare la risposta alla domanda PASS? dell'assembler.

3.1 Ø ERRORI

Gli errori vengono segnalati dall'Assembler in modo diverso a seconda del passo che sta eseguendo quando li incontra. Durante il passo 1 vengono segnalati con una scritta sulla teletype, qualsiasi sia la periferica di lettura del source che è stata selezionata ed hanno il seguente formato:

XX A LABEL + YY

XX identifica il tipo di errore incontrato

LABEL identifica l'ultima label incontrata dall'assembler

YY identifica quante righe dopo la label è stato incontrato l'errore.

Questo tipo di segnalazione permette di risalire facilmente alla riga in cui si trova l'errore, anche se non si ha a disposizione la lista: è infatti sufficiente avere la stampa del source, cercare la label indicata dalla scritta e contare avanti YY righe per trovare quella contenente l'errore. Se l'errore viene incontrato prima della prima label del programma, l'assembler stampa al posto della label, la scritta BEGIN, che sta ad indicare che il numero YY è ottenuto partendo dall'inizio del programma.

I tipi di errori che vengono segnalati con XX sono i seguenti:

1) IC Questo tipo di errore viene segnalato se l'assembler incontra un carattere diverso da quello che si aspetta. Vediamo alcuni esempi che darebbero origine a questo tipo di errore e a fianco gli stessi statements corretti:

ERRATI		CORRETTI	
DJNC	LABEL	DJNZ	LABEL
BIT	7L	BIT	7, L
NPP		NOP	
RRRA		RRCA	

2) ID Questo tipo di errore viene segnalato se l'assembler non trova i caratteri (spazio o tab) di divisione tra i vari campi dello statement. Vediamo degli esempi di statements che dovrebbero originare questo tipo di errore e a fianco gli stessi statements corretti:

ERRATI	CORRETTI	
LDA,B	LD	A, B
INCHL.	INC	HL
ABC:NOP	ABC:	NOP
RETZ	RET	Z

3) IN Questo tipo di errore viene segnalato se l'assembler incontra un riferimento errato dopo la pseudo-istruzione END. vediamo alcuni esempi:

ERRATI		CORRETTI	
END	65AHH	END	65AH
END	6A7.	END	607.

4) IT Questo tipo di errore viene segnalato se l'assembler incontra una label che non risponde a tutti i requisiti visti nel paragrafo 3.4.1.

5) DT Questo tipo di errore viene segnalato se l'assembler incontra una label che era già stata usata nel programma. Questa label viene ignorata e viene considerata valida, ai fini dell'assemblaggio, la prima.

6) TT Questo tipo di errore viene segnalato se l'assembler incontra un numero di label superiore alla capacità della tabella dei simboli. Questo tipo di errore è l'unico che fa interrompere la compilazione perché è l'unico che l'assembler non può in nessun modo ignorare. Gli altri errori vengono segnalati, ma l'assembler riserva, al posto dello statement sbagliato, dei byte nei quali si può mettere il codice corretto in fase di debugging del programma.

Il numero dei byte che l'assembler riserva in caso di errore e che possono non essere a zero, dipende dal tipo di istruzione che l'assembler aveva iniziato a decodificare prima di trovare l'errore.
Durante il passo 2 gli errori non vengono segnalati durante l'assemblaggio per non rovinare il nastro binario, se questi si perfora con la teletype, ma alla fine della compilazione viene dato il numero complessivo degli errori trovati. Questo numero può essere superiore a quello rilevato durante il passo uno, perché ci possono essere due tipi in più di errori rilevati:

1) Riferimento illegale.
Questo errore viene segnalato se una istruzione fa riferimento ad una label che non è descritta nel programma. Un esempio molto semplice è quello del JP ad una label che non esiste.

2) Offset illegale.
Questo errore viene segnalato se una istruzione del tipo JR o DJNZ fa riferimento ad una label che non esiste, oppure se fa riferimento ad una label che si trova più lontano di 128 byte rispetto al punto di richiamo.

Durante il passo tre, gli errori vengono segnalati nella lista nell'apposito campo errore con una lettera che i-

dentifica il tipo. Per esempio la lettera C indica che l'errore è del tipo IC, la lettera D che l'errore è del tipo ID. Alla fine di ogni pagina viene dato il numero complessivo degli errori trovati fino a quel punto.

NOTA : Gli errori del tipo DT e IT non vengono segnalati durante il passo 2 e 3.


```
001                             ;
002                             ; TABELLA DELLE ISTRUZIONI PER Z80
003                             ; ================================
004                             ;
005                             ;
006 0000 8E                     ADC     A,(HL)
007 0001 DD8E05                 ADC     A,(IX+IND)
008 0004 FD8E05                 ADC     A,(IY+IND)
009 0007 8F                     ADC     A,A
010 0008 88                     ADC     A,B
011 0009 89                     ADC     A,C
012 000A 8A                     ADC     A,D
013 000B 8B                     ADC     A,E
014 000C 8C                     ADC     A,H
015 000D 8D                     ADC     A,L
016 000E CE20                   ADC     A,N
017 0010 ED4A                   ADC     HL,BC
018 0012 ED5A                   ADC     HL,DE
019 0014 ED6A                   ADC     HL,HL
020 0016 ED7A                   ADC     HL,SP
021                             ;
022 0018 86                     ADD     A,(HL)
023 0019 DD8605                 ADD     A,(IX+IND)
024 001C FD8605                 ADD     A,(IY+IND)
025 001F 87                     ADD     A,A
026 0020 80                     ADD     A,B
027 0021 81                     ADD     A,C
028 0022 82                     ADD     A,D
029 0023 83                     ADD     A,E
030 0024 84                     ADD     A,H
031 0025 85                     ADD     A,L
032 0026 C620                   ADD     A,N
033 0028 09                     ADD     HL,BC
034 0029 19                     ADD     HL,DE
035 002A 29                     ADD     HL,HL
036 002B 39                     ADD     HL,SP
037 002C DD09                   ADD     IX,BC
038 002E DD19                   ADD     IX,DE
039 0030 DD29                   ADD     IX,IX
040 0032 DD39                   ADD     IX,SP
041 0034 FD09                   ADD     IY,BC
042 0036 FD19                   ADD     IY,DE
043 0038 FD29                   ADD     IY,IY
044 003A FD39                   ADD     IY,SP
045                             ;
046 003C A6                     AND     (HL)
047 003D DDA605                 AND     (IX+IND)
048 0040 FDA605                 AND     (IY+IND)
049 0043 A7                     AND     A
050 0044 A0                     AND     B
051 0045 A1                     AND     C
052 0046 A2                     AND     D
053 0047 A3                     AND     E
054 0048 A4                     AND     H
055 0049 A5                     AND     L
056 004A E620                   AND     N
057                             ;
058 004C CB46                   BIT     0,(HL)
059 004E DDCB0546               BIT     0,(IX+IND)
060 0052 FDCB0546               BIT     0,(IY+IND)
```

ERR. 000

```
061 0056 CB47           BIT     0,A
062 0058 CB40           BIT     0,B
063 005A CB41           BIT     0,C
064 005C CB42           BIT     0,D
065 005E CB43           BIT     0,E
066 0060 CB44           BIT     0,H
067 0062 CB45           BIT     0,L
068 0064 CB4E           BIT     1,(HL)
069 0066 DDCB054E       BIT     1,(IX+IND)
070 006A FDCB054E       BIT     1,(IY+IND)
071 006E CB4F           BIT     1,A
072 0070 CB48           BIT     1,B
073 0072 CB49           BIT     1,C
074 0074 CB4A           BIT     1,D
075 0076 CB4B           BIT     1,E
076 0078 CB4C           BIT     1,H
077 007A CB4D           BIT     1,L
078 007C CB56           BIT     2,(HL)
079 007E DDCB0556       BIT     2,(IX+IND)
080 0082 FDCB0556       BIT     2,(IY+IND)
081 0086 CB57           BIT     2,A
082 0088 CB50           BIT     2,B
083 008A CB51           BIT     2,C
084 008C CB52           BIT     2,D
085 008E CB53           BIT     2,E
086 0090 CB54           BIT     2,H
087 0092 CB55           BIT     2,L
088 0094 CB5E           BIT     3,(HL)
089 0096 DDCB055E       BIT     3,(IX+IND)
090 009A FDCB055E       BIT     3,(IY+IND)
091 009E CB5F           BIT     3,A
092 00A0 CB58           BIT     3,B
093 00A2 CB59           BIT     3,C
094 00A4 CB5A           BIT     3,D
095 00A6 CB5B           BIT     3,E
096 00A8 CB5C           BIT     3,H
097 00AA CB5D           BIT     3,L
098 00AC CB66           BIT     4,(HL)
099 00AE DDCB0566       BIT     4,(IX+IND)
100 00B2 FDCB0566       BIT     4,(IY+IND)
101 00B6 CB67           BIT     4,A
102 00B8 CB60           BIT     4,B
103 00BA CB61           BIT     4,C
104 00BC CB62           BIT     4,D
105 00BE CB63           BIT     4,E
106 00C0 CB64           BIT     4,H
107 00C2 CB65           BIT     4,L
108 00C4 CB6E           BIT     5,(HL)
109 00C6 DDCB056E       BIT     5,(IX+IND)
110 00CA FDCB056E       BIT     5,(IY+IND)
111 00CE CB6F           BIT     5,A
112 00D0 CB68           BIT     5,B
113 00D2 CB69           BIT     5,C
114 00D4 CB6A           BIT     5,D
115 00D6 CB6B           BIT     5,E
116 00D8 CB6C           BIT     5,H
117 00DA CB6D           BIT     5,L
118 00DC CB76           BIT     6,(HL)
119 00DE DDCB0576       BIT     6,(IX+IND)
120 00E2 FDCB0576       BIT     6,(IY+IND)
```



```
121 00E6 CB77           BIT     6,A
122 00E8 CB70           BIT     6,B
123 00EA CB71           BIT     6,C
124 00EC CB72           BIT     6,D
125 00EE CB73           BIT     6,E
126 00F0 CB74           BIT     6,H
127 00F2 CB75           BIT     6,L
128 00F4 CB7E           BIT     7,(HL)
129 00F6 DDCB057E       BIT     7,(IX+IND)
130 00FA FDCB057E       BIT     7,(IY+IND)
131 00FE CB7F           BIT     7,A
132 0100 CB78           BIT     7,B
133 0102 CB79           BIT     7,C
134 0104 CB7A           BIT     7,D
135 0106 CB7B           BIT     7,E
136 0108 CB7C           BIT     7,H
137 010A CB7D           BIT     7,L
138                     ;
139 010C DC8805         CALL    C,NN
140 010F FC8805         CALL    M,NN
141 0112 D48805         CALL    NC,NN
142 0115 CD8805         CALL    NN
143 0118 C48805         CALL    NZ,NN
144 011B F48805         CALL    P,NN
145 011E EC8805         CALL    PE,NN
146 0121 E48805         CALL    PO,NN
147 0124 CC8805         CALL    Z,NN
148                     ;
149 0127 3F             CCF
150                     ;
151 0128 BE             CP      (HL)
152 0129 DDBE05         CP      (IX+IND)
153 012C FDBE05         CP      (IY+IND)
154 012F BF             CP      A
155 0130 B8             CP      B
156 0131 B9             CP      C
157 0132 BA             CP      D
158 0133 BB             CP      E
159 0134 BC             CP      H
160 0135 BD             CP      L
161 0136 FE20           CP      N
162                     ;
163 0138 EDA9           CPD
164                     ;
165 013A EDB9           CPDR
166                     ;
167 013C EDA1           CPI
168                     ;
169 013E EDB1           CPIR
170                     ;
171 0140 2F             CPL
172                     ;
173 0141 27             DAA
174                     ;
175 0142 35             DEC     (HL)
176 0143 DD3505         DEC     (IX+IND)
177 0146 FD3505         DEC     (IY+IND)
178 0149 3D             DEC     A
179 014A 05             DEC     B
180 014B 0B             DEC     BC
```

```
181 014C 0D                     DEC     C
182 014D 15                     DEC     D
183 014E 1B                     DEC     DE
184 014F 1D                     DEC     E
185 0150 25                     DEC     H
186 0151 2B                     DEC     HL
187 0152 DD2B                   DEC     IX
188 0154 FD2B                   DEC     IY
189 0156 2D                     DEC     L
190 0157 3B                     DEC     SP
191                             ;
192 0158 F3                     DI
193                             ;
194 0159 10FE           DIS1:   DJNZ    DIS1
195                             ;
196 015B FB                     EI
197                             ;
198 015C E3                     EX      (SP),HL
199 015D DDE3                   EX      (SP),IX
200 015F FDE3                   EX      (SP),IY
201 0161 08                     EX      AF,AF'
202 0162 EB                     EX      DE,HL
203                             ;
204 0163 D9                     EXX
205                             ;
206 0164 76                     HALT
207                             ;
208 0165 ED46                   IM      0
209 0167 ED56                   IM      1
210 0169 ED5E                   IM      2
211                             ;
212 016B ED78                   IN      A,(C)
213 016D DB20                   IN      A,(N)
214 016F ED40                   IN      B,(C)
215 0171 ED48                   IN      C,(C)
216 0173 ED50                   IN      D,(C)
217 0175 ED58                   IN      E,(C)
218 0177 ED60                   IN      H,(C)
219 0179 ED68                   IN      L,(C)
220                             ;
221 017B 34                     INC     (HL)
222 017C DD3405                 INC     (IX+IND)
223 017F FD3405                 INC     (IY+IND)
224 0182 3C                     INC     A
225 0183 04                     INC     B
226 0184 03                     INC     BC
227 0185 0C                     INC     C
228 0186 14                     INC     D
229 0187 13                     INC     DE
230 0188 1C                     INC     E
231 0189 24                     INC     H
232 018A 23                     INC     HL
233 018B DD23                   INC     IX
234 018D FD23                   INC     IY
235 018F 2C                     INC     L
236 0190 33                     INC     SP
237                             ;
238 0191 EDAA                   IND
239                             ;
240 0193 EDBA                   INDR
```



```
241                             ;
242 0195 EDA2                   INI
243                             ;
244 0197 EDB2                   INIR
245                             ;
246 0199 E9                     JP      (HL)
247 019A DDE9                   JP      (IX)
248 019C FDE9                   JP      (IY)
249 019E DA8805                 JP      C,NN
250 01A1 FA8805                 JP      M,NN
251 01A4 D28805                 JP      NC,NN
252 01A7 C38805                 JP      NN
253 01AA C28805                 JP      NZ,NN
254 01AD F28805                 JP      P,NN
255 01B0 EA8805                 JP      PE,NN
256 01B3 E28805                 JP      PO,NN
257 01B6 CA8805                 JP      Z,NN
258                             ;
259 01B9 38FE           DIS2:   JR      C,DIS2
260 01BB 18FC                   JR      DIS2
261 01BD 30FA                   JR      NC,DIS2
262 01BF 20F8                   JR      NZ,DIS2
263 01C1 28F6                   JR      Z,DIS2
264                             ;
265 01C3 02                     LD      (BC),A
266 01C4 12                     LD      (DE),A
267 01C5 77                     LD      (HL),A
268 01C6 70                     LD      (HL),B
269 01C7 71                     LD      (HL),C
270 01C8 72                     LD      (HL),D
271 01C9 73                     LD      (HL),E
272 01CA 74                     LD      (HL),H
273 01CB 75                     LD      (HL),L
274 01CC 3620                   LD      (HL),N
275 01CE DD7705                 LD      (IX+IND),A
276 01D1 DD7005                 LD      (IX+IND),B
277 01D4 DD7105                 LD      (IX+IND),C
278 01D7 DD7205                 LD      (IX+IND),D
279 01DA DD7305                 LD      (IX+IND),E
280 01DD DD7405                 LD      (IX+IND),H
281 01E0 DD7505                 LD      (IX+IND),L
282 01E3 DD360520               LD      (IX+IND),N
283 01E7 FD7705                 LD      (IY+IND),A
284 01EA FD7005                 LD      (IY+IND),B
285 01ED FD7105                 LD      (IY+IND),C
286 01F0 FD7205                 LD      (IY+IND),D
287 01F3 FD7305                 LD      (IY+IND),E
288 01F6 FD7405                 LD      (IY+IND),H
289 01F9 FD7505                 LD      (IY+IND),L
290 01FC FD360520               LD      (IY+IND),N
291 0200 328805                 LD      (NN),A
292 0203 ED438805               LD      (NN),BC
293 0207 ED538805               LD      (NN),DE
294 020B 228805                 LD      (NN),HL
295 020E DD228805               LD      (NN),IX
296 0212 FD228805               LD      (NN),IY
297 0216 ED738805               LD      (NN),SP
298 021A 0A                     LD      A,(BC)
299 021B 1A                     LD      A,(DE)
300 021C 7E                     LD      A,(HL)
```

```
301 021D DD7E05          LD      A,(IX+IND)
302 0220 FD7E05          LD      A,(IY+IND)
303 0223 3A8805          LD      A,(NN)
304 0226 7F              LD      A,A
305 0227 78              LD      A,B
306 0228 79              LD      A,C
307 0229 7A              LD      A,D
308 022A 7B              LD      A,E
309 022B ED57            LD      A,I
310 022D ED5F            LD      A,R
311 022F 7C              LD      A,H
312 0230 7D              LD      A,L
313 0231 3E20            LD      A,N
314 0233 46              LD      B,(HL)
315 0234 DD4605          LD      B,(IX+IND)
316 0237 FD4605          LD      B,(IY+IND)
317 023A 47              LD      B,A
318 023B 40              LD      B,B
319 023C 41              LD      B,C
320 023D 42              LD      B,D
321 023E 43              LD      B,E
322 023F 44              LD      B,H
323 0240 45              LD      B,L
324 0241 0620            LD      B,N
325 0243 ED4B8805        LD      BC,(NN)
326 0247 018805          LD      BC,NN
327 024A 4E              LD      C,(HL)
328 024B DD4E05          LD      C,(IX+IND)
329 024E FD4E05          LD      C,(IY+IND)
330 0251 4F              LD      C,A
331 0252 48              LD      C,B
332 0253 49              LD      C,C
333 0254 4A              LD      C,D
334 0255 4B              LD      C,E
335 0256 4C              LD      C,H
336 0257 4D              LD      C,L
337 0258 0E20            LD      C,N
338 025A 56              LD      D,(HL)
339 025B DD5605          LD      D,(IX+IND)
340 025E FD5605          LD      D,(IY+IND)
341 0261 57              LD      D,A
342 0262 50              LD      D,B
343 0263 51              LD      D,C
344 0264 52              LD      D,D
345 0265 53              LD      D,E
346 0266 54              LD      D,H
347 0267 55              LD      D,L
348 0268 1620            LD      D,N
349 026A ED5B8805        LD      DE,(NN)
350 026E 118805          LD      DE,NN
351 0271 5E              LD      E,(HL)
352 0272 DD5E05          LD      E,(IX+IND)
353 0275 FD5E05          LD      E,(IY+IND)
354 0278 5F              LD      E,A
355 0279 58              LD      E,B
356 027A 59              LD      E,C
357 027B 5A              LD      E,D
358 027C 5B              LD      E,E
359 027D 5C              LD      E,H
360 027E 5D              LD      E,L
```



```
361 027F 1E20            LD      E,N
362 0281 66              LD      H,(HL)
363 0282 DD6605          LD      H,(IX+IND)
364 0285 FD6605          LD      H,(IY+IND)
365 0288 67              LD      H,A
366 0289 60              LD      H,B
367 028A 61              LD      H,C
368 028B 62              LD      H,D
369 028C 63              LD      H,E
370 028D 64              LD      H,H
371 028E 65              LD      H,L
372 028F 2620            LD      H,N
373 0291 2A8805          LD      HL,(NN)
374 0294 218805          LD      HL,NN
375 0297 ED47            LD      I,A
376 0299 DD2A8805        LD      IX,(NN)
377 029D DD218805        LD      IX,NN
378 02A1 FD2A8805        LD      IY,(NN)
379 02A5 FD218805        LD      IY,NN
380 02A9 6E              LD      L,(HL)
381 02AA DD6E05          LD      L,(IX+IND)
382 02AD FD6E05          LD      L,(IY+IND)
383 02B0 6F              LD      L,A
384 02B1 68              LD      L,B
385 02B2 69              LD      L,C
386 02B3 6A              LD      L,D
387 02B4 6B              LD      L,E
388 02B5 6C              LD      L,H
389 02B6 6D              LD      L,L
390 02B7 2E20            LD      L,N
391 02B9 ED4F            LD      R,A
392 02BB ED7B8805        LD      SP,(NN)
393 02BF F9              LD      SP,HL
394 02C0 DDF9            LD      SP,IX
395 02C2 FDF9            LD      SP,IY
396 02C4 318805          LD      SP,NN
397                      ;
398 02C7 EDA8            LDD
399                      ;
400 02C9 EDB8            LDDR
401                      ;
402 02CB EDA0            LDI
403                      ;
404 02CD EDB0            LDIR
405                      ;
406 02CF ED44            NEG
407                      ;
408 02D1 00              NOP
409                      ;
410 02D2 B6              OR      (HL)
411 02D3 DDB605          OR      (IX+IND)
412 02D6 FDB605          OR      (IY+IND)
413 02D9 B7              OR      A
414 02DA B0              OR      B
415 02DB B1              OR      C
416 02DC B2              OR      D
417 02DD B3              OR      E
418 02DE B4              OR      H
419 02DF B5              OR      L
420 02E0 F620            OR      N
```

```
421                             ;
422 02E2 EDBB                   OTDR
423                             ;
424 02E4 EDB3                   OTIR
425                             ;
426 02E6 ED79                   OUT     (C),A
427 02E8 ED41                   OUT     (C),B
428 02EA ED49                   OUT     (C),C
429 02EC ED51                   OUT     (C),D
430 02EE ED59                   OUT     (C),E
431 02F0 ED61                   OUT     (C),H
432 02F2 ED69                   OUT     (C),L
433 02F4 D320                   OUT     (N),A
434                             ;
435 02F6 EDAB                   OUTD
436                             ;
437 02F8 EDA3                   OUTI
438                             ;
439 02FA F1                     POP     AF
440 02FB C1                     POP     BC
441 02FC D1                     POP     DE
442 02FD E1                     POP     HL
443 02FE DDE1                   POP     IX
444 0300 FDE1                   POP     IY
445                             ;
446 0302 F5                     PUSH    AF
447 0303 C5                     PUSH    BC
448 0304 D5                     PUSH    DE
449 0305 E5                     PUSH    HL
450 0306 DDE5                   PUSH    IX
451 0308 FDE5                   PUSH    IY
452                             ;
453 030A CB86                   RES     0,(HL)
454 030C DDCB0586               RES     0,(IX+IND)
455 0310 FDCB0586               RES     0,(IY+IND)
456 0314 CB87                   RES     0,A
457 0316 CB80                   RES     0,B
458 0318 CB81                   RES     0,C
459 031A CB82                   RES     0,D
460 031C CB83                   RES     0,E
461 031E CB84                   RES     0,H
462 0320 CB85                   RES     0,L
463 0322 CB8E                   RES     1,(HL)
464 0324 DDCB058E               RES     1,(IX+IND)
465 0328 FDCB058E               RES     1,(IY+IND)
466 032C CB8F                   RES     1,A
467 032E CB88                   RES     1,B
468 0330 CB89                   RES     1,C
469 0332 CB8A                   RES     1,D
470 0334 CB8B                   RES     1,E
471 0336 CB8C                   RES     1,H
472 0338 CB8D                   RES     1,L
473 033A CB96                   RES     2,(HL)
474 033C DDCB0596               RES     2,(IX+IND)
475 0340 FDCB0596               RES     2,(IY+IND)
476 0344 CB97                   RES     2,A
477 0346 CB90                   RES     2,B
478 0348 CB91                   RES     2,C
479 034A CB92                   RES     2,D
480 034C CB93                   RES     2,E
```



```
481 034E CB94                   RES     2,H
482 0350 CB95                   RES     2,L
483 0352 CB9E                   RES     3,(HL)
484 0354 DDCB059E               RES     3,(IX+IND)
485 0358 FDCB059E               RES     3,(IY+IND)
486 035C CB9F                   RES     3,A
487 035E CB98                   RES     3,B
488 0360 CB99                   RES     3,C
489 0362 CB9A                   RES     3,D
490 0364 CB9B                   RES     3,E
491 0366 CB9C                   RES     3,H
492 0368 CB9D                   RES     3,L
493 036A CBA6                   RES     4,(HL)
494 036C DDCB05A6               RES     4,(IX+IND)
495 0370 FDCB05A6               RES     4,(IY+IND)
496 0374 CBA7                   RES     4,A
497 0376 CBA0                   RES     4,B
498 0378 CBA1                   RES     4,C
499 037A CBA2                   RES     4,D
500 037C CBA3                   RES     4,E
501 037E CBA4                   RES     4,H
502 0380 CBA5                   RES     4,L
503 0382 CBAE                   RES     5,(HL)
504 0384 DDCB05AE               RES     5,(IX+IND)
505 0388 FDCB05AE               RES     5,(IY+IND)
506 038C CBAF                   RES     5,A
507 038E CBA8                   RES     5,B
508 0390 CBA9                   RES     5,C
509 0392 CBAA                   RES     5,D
510 0394 CBAB                   RES     5,E
511 0396 CBAC                   RES     5,H
512 0398 CBAD                   RES     5,L
513 039A CBB6                   RES     6,(HL)
514 039C DDCB05B6               RES     6,(IX+IND)
515 03A0 FDCB05B6               RES     6,(IY+IND)
516 03A4 CBB7                   RES     6,A
517 03A6 CBB0                   RES     6,B
518 03A8 CBB1                   RES     6,C
519 03AA CBB2                   RES     6,D
520 03AC CBB3                   RES     6,E
521 03AE CBB4                   RES     6,H
522 03B0 CBB5                   RES     6,L
523 03B2 CBBE                   RES     7,(HL)
524 03B4 DDCB05BE               RES     7,(IX+IND)
525 03B8 FDCB05BE               RES     7,(IY+IND)
526 03BC CBBF                   RES     7,A
527 03BE CBB8                   RES     7,B
528 03C0 CBB9                   RES     7,C
529 03C2 CBBA                   RES     7,D
530 03C4 CBBB                   RES     7,E
531 03C6 CBBC                   RES     7,H
532 03C8 CBBD                   RES     7,L
533                             ;
534 03CA C9                     RET
535 03CB D8                     RET     C
536 03CC F8                     RET     M
537 03CD D0                     RET     NC
538 03CE C0                     RET     NZ
539 03CF F0                     RET     P
540 03D0 E8                     RET     PE
```

```
541 03D1 E0              RET     PO
542 03D2 C8              RET     Z
543                      ;
544 03D3 ED4D            RETI
545                      ;
546 03D5 ED45            RETN
547                      ;
548 03D7 CB16            RL      (HL)
549 03D9 DDCB0516        RL      (IX+IND)
550 03DD FDCB0516        RL      (IY+IND)
551 03E1 CB17            RL      A
552 03E3 CB10            RL      B
553 03E5 CB11            RL      C
554 03E7 CB12            RL      D
555 03E9 CB13            RL      E
556 03EB CB14            RL      H
557 03ED CB15            RL      L
558                      ;
559 03EF 17              RLA
560                      ;
561 03F0 CB06            RLC     (HL)
562 03F2 DDCB0506        RLC     (IX+IND)
563 03F6 FDCB0506        RLC     (IY+IND)
564 03FA CB07            RLC     A
565 03FC CB00            RLC     B
566 03FE CB01            RLC     C
567 0400 CB02            RLC     D
568 0402 CB03            RLC     E
569 0404 CB04            RLC     H
570 0406 CB05            RLC     L
571                      ;
572 0408 07              RLCA
573                      ;
574 0409 ED6F            RLD
575                      ;
576 040B CB1E            RR      (HL)
577 040D DDCB051E        RR      (IX+IND)
578 0411 FDCB051E        RR      (IY+IND)
579 0415 CB1F            RR      A
580 0417 CB18            RR      B
581 0419 CB19            RR      C
582 041B CB1A            RR      D
583 041D CB1B            RR      E
584 041F CB1C            RR      H
585 0421 CB1D            RR      L
586                      ;
587 0423 1F              RRA
588                      ;
589 0424 CB0E            RRC     (HL)
590 0426 DDCB050E        RRC     (IX+IND)
591 042A FDCB050E        RRC     (IY+IND)
592 042E CB0F            RRC     A
593 0430 CB08            RRC     B
594 0432 CB09            RRC     C
595 0434 CB0A            RRC     D
596 0436 CB0B            RRC     E
597 0438 CB0C            RRC     H
598 043A CB0D            RRC     L
599                      ;
600 043C 0F              RRCA
```



```
601                      ;
602 043D ED67            RRD
603                      ;
604 043F C7              RST     00H
605 0440 D7              RST     10H
606 0441 DF              RST     18H
607 0442 E7              RST     20H
608 0443 EF              RST     28H
609 0444 F7              RST     30H
610 0445 FF              RST     38H
611 0446 CF              RST     08H
612                      ;
613 0447 9E              SBC     A,(HL)
614 0448 DD9E05          SBC     A,(IX+IND)
615 044B FD9E05          SBC     A,(IY+IND)
616 044E 9F              SBC     A,A
617 044F 98              SBC     A,B
618 0450 99              SBC     A,C
619 0451 9A              SBC     A,D
620 0452 9B              SBC     A,E
621 0453 9C              SBC     A,H
622 0454 9D              SBC     A,L
623 0455 DE20            SBC     A,N
624 0457 ED42            SBC     HL,BC
625 0459 ED52            SBC     HL,DE
626 045B ED62            SBC     HL,HL
627 045D ED72            SBC     HL,SP
628                      ;
629 045F 37              SCF
630                      ;
631 0460 CBC6            SET     0,(HL)
632 0462 DDCB05C6        SET     0,(IX+IND)
633 0466 FDCB05C6        SET     0,(IY+IND)
634 046A CBC7            SET     0,A
635 046C CBC0            SET     0,B
636 046E CBC1            SET     0,C
637 0470 CBC2            SET     0,D
638 0472 CBC3            SET     0,E
639 0474 CBC4            SET     0,H
640 0476 CBC5            SET     0,L
641 0478 CBCE            SET     1,(HL)
642 047A DDCB05CE        SET     1,(IX+IND)
643 047E FDCB05CE        SET     1,(IY+IND)
644 0482 CBCF            SET     1,A
645 0484 CBC8            SET     1,B
646 0486 CBC9            SET     1,C
647 0488 CBCA            SET     1,D
648 048A CBCB            SET     1,E
649 048C CBCC            SET     1,H
650 048E CBCD            SET     1,L
651 0490 CBD6            SET     2,(HL)
652 0492 DDCB05D6        SET     2,(IX+IND)
653 0496 FDCB05D6        SET     2,(IY+IND)
654 049A CBD7            SET     2,A
655 049C CBD0            SET     2,B
656 049E CBD1            SET     2,C
657 04A0 CBD2            SET     2,D
658 04A2 CBD3            SET     2,E
659 04A4 CBD4            SET     2,H
660 04A6 CBD5            SET     2,L
```

```
661 04A8 CBDE              SET     3,(HL)
662 04AA DDCB05DE          SET     3,(IX+IND)
663 04AE FDCB05DE          SET     3,(IY+IND)
664 04B2 CBDF              SET     3,A
665 04B4 CBD8              SET     3,B
666 04B6 CBD9              SET     3,C
667 04B8 CBDA              SET     3,D
668 04BA CBDB              SET     3,E
669 04BC CBDC              SET     3,H
670 04BE CBDD              SET     3,L
671 04C0 CBE6              SET     4,(HL)
672 04C2 DDCB05E6          SET     4,(IX+IND)
673 04C6 FDCB05E6          SET     4,(IY+IND)
674 04CA CBE7              SET     4,A
675 04CC CBE0              SET     4,B
676 04CE CBE1              SET     4,C
677 04D0 CBE2              SET     4,D
678 04D2 CBE3              SET     4,E
679 04D4 CBE4              SET     4,H
680 04D6 CBE5              SET     4,L
681 04D8 CBEE              SET     5,(HL)
682 04DA DDCB05EE          SET     5,(IX+IND)
683 04DE FDCB05EE          SET     5,(IY+IND)
684 04E2 CBEF              SET     5,A
685 04E4 CBE8              SET     5,B
686 04E6 CBE9              SET     5,C
687 04E8 CBEA              SET     5,D
688 04EA CBEB              SET     5,E
689 04EC CBEC              SET     5,H
690 04EE CBED              SET     5,L
691 04F0 CBF6              SET     6,(HL)
692 04F2 DDCB05F6          SET     6,(IX+IND)
693 04F6 FDCB05F6          SET     6,(IY+IND)
694 04FA CBF7              SET     6,A
695 04FC CBF0              SET     6,B
696 04FE CBF1              SET     6,C
697 0500 CBF2              SET     6,D
698 0502 CBF3              SET     6,E
699 0504 CBF4              SET     6,H
700 0506 CBF5              SET     6,L
701 0508 CBFE              SET     7,(HL)
702 050A DDCB05FE          SET     7,(IX+IND)
703 050E FDCB05FE          SET     7,(IY+IND)
704 0512 CBFF              SET     7,A
705 0514 CBF8              SET     7,B
706 0516 CBF9              SET     7,C
707 0518 CBFA              SET     7,D
708 051A CBFB              SET     7,E
709 051C CBFC              SET     7,H
710 051E CBFD              SET     7,L
711                        ;
712 0520 CB26              SLA     (HL)
713 0522 DDCB0526          SLA     (IX+IND)
714 0526 FDCB0526          SLA     (IY+IND)
715 052A CB27              SLA     A
716 052C CB20              SLA     B
717 052E CB21              SLA     C
718 0530 CB22              SLA     D
719 0532 CB23              SLA     E
720 0534 CB24              SLA     H
```



```
721 0536 CB25              SLA     L
722                        ;
723 0538 CB2E              SRA     (HL)
724 053A DDCB052E          SRA     (IX+IND)
725 053E FDCB052E          SRA     (IY+IND)
726 0542 CB2F              SRA     A
727 0544 CB28              SRA     B
728 0546 CB29              SRA     C
729 0548 CB2A              SRA     D
730 054A CB2B              SRA     E
731 054C CB2C              SRA     H
732 054E CB2D              SRA     L
733                        ;
734 0550 CB3E              SRL     (HL)
735 0552 DDCB053E          SRL     (IX+IND)
736 0556 FDCB053E          SRL     (IY+IND)
737 055A CB3F              SRL     A
738 055C CB38              SRL     B
739 055E CB39              SRL     C
740 0560 CB3A              SRL     D
741 0562 CB3B              SRL     E
742 0564 CB3C              SRL     H
743 0566 CB3D              SRL     L
744                        ;
745 0568 96                SUB     (HL)
746 0569 DD9605            SUB     (IX+IND)
747 056C FD9605            SUB     (IY+IND)
748 056F 97                SUB     A
749 0570 90                SUB     B
750 0571 91                SUB     C
751 0572 92                SUB     D
752 0573 93                SUB     E
753 0574 94                SUB     H
754 0575 95                SUB     L
755 0576 D620              SUB     N
756                        ;
757 0578 AE                XOR     (HL)
758 0579 DDAE05            XOR     (IX+IND)
759 057C FDAE05            XOR     (IY+IND)
760 057F AF                XOR     A
761 0580 A8                XOR     B
762 0581 A9                XOR     C
763 0582 AA                XOR     D
764 0583 AB                XOR     E
765 0584 AC                XOR     H
766 0585 AD                XOR     L
767 0586 EE20              XOR     N
768                        ;
769                        ; COSTANTI
770                        ;
771 0588 0000        NN:   DEFS    2H
772                  IND:  EQU     5H
773                  N:    EQU     20H
774 058A 4141              DEFM    /AA/
775 058C 77                DEFB    77H
776 058D FFFF              DEFW    FFFFH
777                        END
```

```
601 06A9 FE33      T3:     CP      '3
602 06AB 20F0              JR      NZ,T4
603 06AD FE34              CP      '4
604 06AF 2004              JR      NZ,T6
605 06B1 3E02              LD      A,02H
606 06B3 18EA              JR      T2
607 06B5 3E10      T6:     LD      A,10H
608 06B7 18E6              JR      T2
609                ;
610                ;
611                ; ROUTINE PER FARE UN RITARDO DI 22.4 USEC
612                ;
613                ;
614 06B9 FDE5      RIT20:  PUSH    IY
615 06BB FDE1              POP     IY
616 06BD C9                RET
617                ;
618                ;
619                ;
620 06BE 79        SCRIVI: LD      A,C
621 06BF D30C              OUT     (0CH),A
622 06C1 78                LD      A,B
623 06C2 D30D              OUT     (0DH),A
624 06C4 C9                RET
625                ;
626                ;
627 06C5 ED53B707  INDIR:  LD      (SAVE),DE
628 06C9 E5                PUSH    HL
629 06CA 21B807            LD      HL,SAVE1
630 06CD CDE206            CALL    BINSTA
631 06D0 2B                DEC     HL
632 06D1 CDE206            CALL    BINSTA
633 06D4 E1                POP     HL
634 06D5 CDDC06            CALL    SPAZ
635 06D8 CDDC06            CALL    SPAZ
636 06DB C9                RET
637                ;
638                ;
639 06DC 3E20      SPAZ:   LD      A,20H
640 06DE CD7D07            CALL    WRITE
641 06E1 C9                RET
642                ;
643                ;
644 06E2 C5        BINSTA: PUSH    BC
645 06E3 CD9307            CALL    BINHEX
646 06E6 CD7D07            CALL    WRITE
647 06E9 78                LD      A,B
648 06EA CD7D07            CALL    WRITE
649 06ED C1                POP     BC
650 06EE C9                RET
651                ;
652                ;
653 06EF D9        TALK:   EXX
654 06F0 E3                EX      (SP),HL
655 06F1 46                LD      B,(HL)
656 06F2 23        TL2:    INC     HL
657 06F3 7E                LD      A,(HL)
658 06F4 CD7D07            CALL    WRITE
659 06F7 10F9              DJNZ    TL2
660 06F9 23                INC     HL
```

MONITOR - DEBUGGER - LOADER
(MDL)

INDICE

		PAG.
4.	MDL - Monitor, Debugger, Loader	4. -1÷2
4.1.	Comandi MDL	4.1. -1
	4.1.1. Comando L	-1
	4.1.2. Comando H	-1
	4.1.3. Comando C	-1
	4.1.4. Richiamo Programmi base	-1
4.2.	Comando "Memory Mode" (O)	4.2. -1
	4.2.1. Slash (/)	-1÷2
	4.2.1.1. Comandi da tastiera	-3
	4.2.1.2. Breakpoint	-4
	4.2.1.3. Comando di partenza (G)	-5
	4.2.1.4. Comando di continuazione	-6
	4.2.1.5. Memory Trace	-7
	4.2.1.6. Memory Trace Reset	-7
4.3.	Register Mode	4.3. -1÷3
	4.3.1. Register Trace	-4
	4.3.2. Register Trace Reset	-4
4.4.	Comando di ripetizione	4.4. -1
	4.4.1. Proceed Command	-1
4.5.	Single Step	4.5. -1
	4.5.1. Multiple step with Register Trace	-1÷2
4.6.	Ritorno all'MDL	4.6. -1
	4.6.1. Ritorno da un Breakpoint	-1
	4.6.2. Non maskable Interrupt	-1÷2
4.7.	Lista dei comandi dell'MDL	4.7. -1÷3

4. M0-Z- MONITOR - DEBUG - LOADER

L'M0-Z è uno strumento potente per la messa a punto di attrezzature per lo sviluppo di "software" sul microcalcolatore SGS-ATES basato sullo Z80 CPU.

Il programma M0-Z si trova su 2 K x 8 EPROM, PROM (o ROM). L'indirizzo di partenza dell'M0-Z è FC00 in esadecimale.

Usa anche le locazioni dalla 0070H alla 0200H della memoria RAM come stack e memoria temporanea.

Per questo i programmi dell'utente devono essere caricati sopra la locazione 0200H.

Quando inizialmente viene fatto partire l'M0-Z, quest'ultimo carica 3 bytes partendo all'indirizzo 0038H, con l'istruzione di un JUMP per il ritorno all'M0-Z da un "breakpoint".

Carica anche un JUMP incominciando dalla locazione speciale 0066H (non maskable interrupt) per il ritorno all'M0-Z da un "non maskable interrupt".

Le caratteristiche dell'M0-Z permettono di:

1. Esaminare e/o modificare qualsiasi locazione di memoria
2. Esaminare e/o modificare lo stato del CPU e registri di lavoro (principali ed alternativi).
3. Iniziare un programma dell'utente ad un indirizzo specificato.
4. Mettere un breakpoint
5. Ritornare dinamicamente da un "breakpoint" programmato salvando lo stato del processore e i registri di lavoro dell'utilizzatore.
6. Continuare dinamicamente un programma dell'utilizzatore dopo un breakpoint con una nuova sistemazione di breakpoint.

7. Eseguire una per volta in software le istruzioni del programma utente tramite semplici comandi da tastiera.
8. Ritornare dinamicamente all'M0-Z n risposta ad un "non maskable interrupt" salvando lo stato del processore e i registri di lavoro dell'utilizzatore.
9. Caricare i programmi assoluti (uscita dall'assemblatore) mediante il Loader.
10. Richiamare i programmi base.

4.1. Comandi dell'M0-Z

Quando il programma dell'M0-Z inizia, risponde con un asterisco (*) indicando che è pronto ad accettare i comandi dalla tastiera.

I sotto elencati comandi caricano il programma binario dal dispositivo d'ingresso in memoria.

4.1.1. * L Caricamento dal lettore lento (TTY)

4.1.2. * H Caricamento dal lettore veloce

4.1.3. * C Caricamento dalla cassetta magnetica

Il caricatore controlla ogni possibile errore di "checksum" e se tale errore esiste, ferma il caricamento e stampa un punto interrogativo (?) ritornando in "Command Mode".

Alla fine del caricamento il "Loader" fa partire il programma dell'utente o ritorna in "Command Mode".

4.1.4. Richiamo Programmi Base

* A - Assembler
* E - Editor
* D - Memory Dump

4.2. * O Memory Mode

In questo modo l'M0-Z accetta qualsiasi indirizzo esadecimale (ϕϕϕϕ-FFFF) e aspetta che uno dei seguenti caratteri speciali siano scritti dall'utente.

4.2.1. Barra (/)

O- XXXX/-YY-

Apre la locazione indicata dall'indirizzo esadecimale XXXX.

L'M0-Z stampa il contenuto in forma esadecimale (-YY-) e aspetta che l'utente carichi un nuovo valore o chiuda la locazione.

Se l'utente carica un nuovo valore, l'M0-Z sostituisce i contenuti vecchi della locazione corrente ma non la chiude.

In questo modo se l'entrata è sbagliata l'utente può caricare il valore esatto:

O-FCϕA/-AB-A1-AC-

Esempio: L'utente in risposta ad * batte sulla TTY i tasti "O" e "RETURN"

L'M0-Z stampa:

O-

L'utente scrive l'indirizzo della locazione

O-FCϕA

L'utente chiede all'M0-Z di stampare i contenuti di quella locazione battendo una barra (/)

O-FCϕA/-AB-

L'MDL stampa -AB-

L'utente scrive A1

O-FCϕA/-AB-A1-

La locazione FCϕA contiene ora A1.

L'utente scrive AC

O-FCØA/-AB-A1-AC-

La locazione FCØA contiene ora AC.

4.2.1.1. Comandi dalla tastiera in Memory Mode

A) RETURN

O-FCØA/-AC- (RETURN)

O-

Chiude la locazione corrente e aspetta che una nuova venga scritta dall'utente.

B) LINE FEED

O-FCØA/-AC- (LINE FEED)

FCØB/-FB-

Chiude la locazione corrente, apre quella successiva e stampa il contenuto di quest'ultima.

< Segno di minore

O-FCØA/AC - <

FCØ9/-3E-

Chiude la locazione, apre la precedente e stampa il contenuto di quest'ultima.

Qualsiasi altro carattere fa si che l'M0-Z riapra la locazione e stampi il contenuto.

4.2.1.2. Breakpoint

O-1CBØB

O-

In tale modo l'utente pone un breakpoint all'indirizzo 1CB0.

L'M0-Z salva i contenuti di questa locazione e carica una istruzione di re-start (RST 38H-codice FFH) la quale contiene l'indirizzo della routine al suo ritorno da un breakpoint.

Quando il programma dell'utente esegue il ritorno all'M0-Z,i contenuti originali sono reintegrati e così il breakpoint è rimosso.

Nota: I breakpoints devono essere messi a locazioni contenenti un "byte" che identifica una istruzione:

Esempio:

* C2 -JP NZ, 1234H

34

12

* 3E-LD A,ϕDH
ϕD

* 21 - LD HL, 1ϕϕϕH

ϕϕ

1ϕ

* 77 - LD (HL), A

* 23 - INC HL

* CD - CALL 5678H

78

56

L'asterisco indica i "bytes" che possono essere usati come breakpoints.

Una volta che il comando "B" è stato scritto il programma dell'utente è modificato (1 byte).

Se l'utente vuole rimuovere il breakpoint senza attuarlo, deve reintegrare i contenuti originali di quel "byte".

4.2.1.3. Partenza di un programma dell'utente

O-XXXXG

L'M0-Z fa partire il programma all'indirizzo esadecimale XXXX

4.2.1.4. Mettere un nuovo breakpoint e continuare l'esecuzione dal programma dell'utente dall'indirizzo di un precedente breakpoint

O-XXXXC

L'M0-Z mette un breakpoint all'indirizzo XXXX ed esegue un comando di "GO" all'indirizzo del precedente breakpoint.

Importante: il primo breakpoint deve sempre essere dichiarato usando il comando "B" dopo di che il comando "C" può essere eseguito tutte le volte che l'utente desidera.

Esempio:

* O

O-ϕ4ϕϕB
O-ϕ2ϕϕG
Al ritorno da un breakpoint l'M0-Z stampa:
B-ϕ4ϕϕ-AϕBϕ
e ritorna in modo di comando

*

L'utente desidera mettereun nuovo breakpoint e continuare l'esecuzione del suo programma dalla locazione ϕ4ϕϕH

* O

O-ϕ41ϕC

4.2.1.5. Memory Trace Command

Con questo comando l'utente può chiedere il contenuto di 16 locazioni consecutive.
L'M0-Z stampa i 16 contenuti in una singola riga.

O - ØØ2Ø/AA-T- ØØ -FF-AC- . . . . . BB-CC
ØØ3Ø/EE-

4.2.1.6. Memory Trace Reset Command

Questo comando permette all'utente di azzerare 16 locazioni consecutive

O-ØØ2Ø/AA-Z
ØØ2Ø/ØØ-ØØ-ØØ-. . . . . .ØØ-ØØ

ØØ3Ø/EE-

4.3. Register Mode

Dal command mode l'utente può entrare in register mode stampando il carattere "R"

* R

L'M0-Z risponde:

R-

In Register Mode l'M0-Z permette all'utente di esaminare tutti i registri del CPU (coppia principale ed alternativa) e modificarli se desidera.

I contenuti di questi registri sono salvati quando il programma dell'utente ritorna all'M0-Z da un breakpoint e sono reintegrati dai seguenti comandi:
"G", "C", "S ", "B", "Tn".

L'utente ha accesso ai contenuti dei seguenti registri:

Principali		Alternativi
AF	-	AF'
BC	-	BC'
DE	-	DE'
HL	-	HL'
IX	-	
IY	-	
I	-	
SP	-	

Per i registri AF, BC, DE, HL l'utente fornisce(dopo R-)il primo carattere, l'M0-Z risponde con il secondo e aspetta che uno dei seguenti caratteri sia stampato dall'utente.

/ - per i contenuti della coppia principale

' - per i contenuti delle coppie alternative

Esempio:

R-AF/-ϕ14C

L'utente stampa A in risposta a R-

L'M0-Z risponde con F

L'utente stampa /

L'M0-Z risponde con -ϕ14C in forma esadecimale

ϕ1 - contenuti del registro A

4C - contenuti del registro F

A questo punto l'utente può caricare un nuovo valore esadecimale oppure può battere RETURN; in tal caso l'M0-Z stampa R- ed aspetta che un altro registro sia specificato

R-DE'-Fϕ14-

L'utente stampa D

L'M0-Z risponde con E

L'utente richiede i contenuti della coppia alternativa DE' stampando "'"

L'M0-Z risponde con i contenuti -Fϕ14-

R-HL/-ϕϕϕϕ-F01B

R-

L'utente stampa H

l'M0-Z risponde con L

l'utente stampa "/"

l'M0-Z risponde con -ϕϕϕϕ-

L'utente stampa Fϕ1B

l'M0-Z risponde con

R-

a questo punto il registro HL contiene Fϕ1B

R-I/-ϕϕ-

L'utente richiede il contenuto del registro I "interrupt" stampando I/.

l'M0-Z risponde con -ϕϕ-

R-IX/-ϕϕ1A-

L'utente richiede i contenuti del registro indice IX stampando IX

L'M0-Z risponde con /-ϕϕ1A-

R-SP/-ϕ1ϕϕ-

L'utente richiede il contenuto dello "stack pointer" scrivendo S

LM0-Z risponde con P/-ϕ1ϕϕ-

ESCAPE fa tornare l'M0-Z in Command Mode.

4.3.1. Register Trace Command

Questo comando stampa il contenuto dei registri del CPU su una riga

R-T- XXXX - XXXX - . . . . . XXXX-XX

R-

nel seguente ordine

AF-HL-DE-BC-AF'-HL'-DE'-BC'-IX-IY-SP-I

4.3.2. Register Trace Reset Command

Questo comando permette di azzerare tutti i registri del CPU (eccetto SP) e di stamparne il contenuto nell'ordine sopra indicato

R-Z- ØØØØ-ØØØØ- . . . . . ØØØØ-XXXX-ØØ

R-

Dopo questi comandi l'M0-Z rimane in Register Mode.

4.4. Ripetere sequenza Breakpoint

* B

Con questo comando l'utente su ritorno da un breakpoint può ripetere lo stesso breakpoint.

4.4.1. Proceed Command

Dopo il ritorno da un breakpoint premendo il tasto "P" si fa in modo di eseguire l'istruzione dove il breakpoint era collocato; dopo di che l'MDL pone di nuovo il breakpoint nello stesso punto, continua l'esecuzione del programma dell'utente fino ad incontrare di nuovo tale breakpoint.

In pratica si riprende l'esecuzione del programma senza togliere il breakpoint dal punto in cui era stato messo.

4.5. Single Step

Dopo il ritorno all'M0-Z da un breakpoint si può continuare l'esecuzione del programma una istruzione per volta con il comando S

* S

L'istruzione viene eseguita e l'M0-Z stampa:

B-XXXX-YYYY

XXXX- Contenuto del registro PC

YYYY- Contenuto del registro AF.

4.5.1. Multiple Step con Register Trace

Questo comando permette di eseguire n (1 ÷ F) istruzioni una per volta e stampare il contenuto di tutti i registri del CPU dopo ogni istruzione

* Tn

n deve essere un numero esadecimale da 1 a F.

Dopo l'esecuzione di ogni istruzione M0-Z stampa:

```
B -XXXX-XXXX-XXXX. . . . XXXX-XX
B -XXXX-XXXX-XXXX. . . . XXXX-XX
:
.
B -XXXX-XXXX-XXXX. . . . XXXX-XX
```


L'ordine dei registri è la seguente:

PC-AF-HL-DE-BC-AF'-HL'-DE'-BC'-IX-IY-SP-I

Nota:
Questi due comandi (Single Step - Multiple Step con Register Trace) non devono essere usati per far eseguire le istruzioni DI, EI e LD I,A poichè queste ultime sono usate dall'M0-Z per realizzare questa funzione.

4.6. Ritorni all'M0-Z

4.6.1. Ritorno da un breakpoint

Su ritorno da un breakpoint l'M0-Z salva tutti i registri del CPU e rimuove il breakpoint dal programma dell'utente; in oltre

stampa sul terminale l'indirizzo del breakpoint e i contenuti della coppia AF e ritorna in Command Mode

B-XXXX - YYYY

*

XXXX indirizzo del breakpoint

YYYY contenuti della coppia registro AF

4.6.2. Nonmaskable interrupt

Premendo l'apposito pulsante l'utente genera un "nonmaskable interrupt".

Il CPU salva il contenuto del "program counter" nello stack dell'utente e salta alla locazione ϕϕ66H.

All'inizio l' M0-Z carica 3 bytes incominciando da quell'indirizzo (ϕϕ66) con un'istruzione di salto all'indirizzo del programma di "nonmaskable interrupt".

Questo programma salva tutti i registri dell'utente, stampa i contenuti del "program counter" e la coppia del registro

AF, e ritorna in Command Mode.

B-XXXX-YYYY

*

dove:

XXXX - program counter dell'utente al momento che il "nonmaskable interrupt" è generato.

YYYY - contenuti della coppia AF

Nota: Se il programma dell'utente occupa anche una sola delle locazioni ϕϕ66H, ϕϕ67H e ϕϕ68H il "nonmaskable interrupt" non deve essere usato.

Si ricorda anche che se l'interrupt viene generato prima di qualche breakpoint, questo non è rimosso dal programma dell'utente dall'M0-Z.

4.7. Lista comandi M0-Z e caratteri speciali

L . . . . . Caricamento dal lettore lento (TTY)

H . . . . . Caricamento dal lettore veloce

C . . . . . Caricamento dalla cassetta magnetica

O . . . . . Memory Mode

/ - Apre la locazione e stampa i contenuti

RETURN - Chiude la locazione e rimane in Memory Mode

LINE FEED - Chiude la locazione corrente, apre la successiva e stampa il contenuto.

< - Chiude la locazione corrente, apre la precedente e stampa il contenuto.

B - Programma il breakpoint

G - Inizia un programma

C - Programma un nuovo breakpoint e continua l'esecuzione di un programma da un breakpoint precedente.

R ---- - Register Mode

/ - Stampa il contenuto della coppia principale

' - Stampa il contenuto della coppia alternativa

T - Stampa il contenuto di tutti i registri

Z - Azzera tutti i registri (eccetto SP)

B ---- - Ripete la sequenza breakpoint

S ---- - Single step

Tn --- - Multiple step con Register Trace

ESCAPE - Ritorno in modo di comando

APPENDICE

Terminale "SILENT"

Normalmente il software del microcalcolatore utilizza come periferiche di I/O terminali standard (TTY, CRT e cassette magnetiche).

Esiste però la possibilità di utilizzare come periferica la "SILENT 7ØØ ASR" della T.I.

E' cura dell'utente inizializzare un flag per selezionare il tipo di periferica che ha a disposizione.

L'utente può configurare il sistema secondo il contenuto del flag nei seguenti modi (per fare ciò si usa l'MDL in MEMORY MODE):

Indirizzo Flag	Contenuto	Spiegazione
ØØA9H	ØFH	Aggiunge un ritardo di 2ØØ mS dopo un CR: da utilizzare nel caso che l'utente adoperi le normali cassette magnetiche e la "SILENT" come terminale.
ØØA9H	FFH	Aggiunge un ritardo di 2ØØ mS dopo un CR: da utilizzare nel caso che l'utente usa solamente la "SILENT" (comprese le cassette della "SILENT").
ØØA9H	ØØH	Configurazione standard (TTY - CRT - cassette magnetiche).

INDICE

		PAG.
5.1.	Introduzione	5.1.- 1
5.2.	Modi di operazione	5.2.- 1
5.3.	Comandi dell'Editor	5.3.- 1÷2
5.4.	Designazione Input/Output Device	5.4.- 1÷2
5.5.	Comandi del Text Input	5.5.- 1
5.5.1.	Comando Append	- 1
5.5.2.	Comando Read	- 2
5.5.3.	Comando Editing	- 3
5.5.4.	Comando Kill	- 3
5.5.5.	Comando Delete	- 3
5.5.6.	Comando Insert	- 4
5.5.7.	Comando Overlay	- 5
5.5.8.	Comando Change	- 5
5.5.9.	Ricerca di "String"	- 6
5.5.10.	Ricerca Buffer	- 7
5.6.	Comandi del Text Output	5.6.- 1
5.6.1.	Comando List	- 1
5.6.2.	Comando Write	- 1
5.6.3.	Comando Next	- 2
5.6.4.	Comando Quit	- 2
5.6.5.	Comando Exit	- 2
5.6.6.	Opzione "T"	- 3
5.6.7.	Leader/Trailer	- 3

./..

		PAG.
5.7.	Comandi e funzioni di tasti speciali	5.7.- 1
5.7.1.	Tasto RETURN	- 1
5.7.2.	Erase (CTRL/U)	- 2
5.7.3.	RUBOUT or DELETE	- 2
5.7.4.	ALTMODE or ESCAPE	- 2
5.7.5.	Contatore della riga corrente	- 3
5.7.6.	Slash (/)	- 4
5.7.7.	LINE FEED	- 4
5.7.8.	Segno di minore (<)	- 4
5.7.9.	Segno uguale (=)	- 5
5.7.10.	CTRL/BELL	- 5
5.7.11.	Tabulazione (CTRL/TAB)	- 6
5.7.12.	CTRL/C	- 6
5.8.	Messaggi diagnostici	5.8.- 1÷2