IL µP Z80

Caratteristiche generali

Lo Z80, realizzato in tecnologia NMOS, è uno dei microprocessori ad 8 bit più diffusi; con le sue 16 linee di indirizzo è in grado di gestire fino a 64 kbyte di memoria. Introdotto sul mercato dalla Zilog viene prodotto anche da altre Case costruttrici (*second source* o fonti alternative), come la SGS, alcune delle quali forniscono anche la versione CMOS (Toshiba e Sharp).
La frequenza massima di clock è di 2,5 MHz, per la versione base; essa sale a 4 MHz, 6 MHz e 8 MHz rispettivamente per le versioni Z80A, Z80B e Z80H. Necessita di una tensione di alimentazione $V_{CC} = +5$ V ed assorbe una corrente massima di 150 mA per la versione base, 200 mA per le versioni più veloci. I livelli delle tensioni sui suoi pin sono TTL compatibili. Dispone di due banchi di registri di lavoro, interscambiabili fra loro ed è provvisto di circuito interno per il rinfresco delle memorie dinamiche.
Il suo set di istruzioni è piuttosto vasto, essendo costituito da ben 158 istruzioni.
La famiglia Z80 comprende, oltre alla CPU, diversi dispositivi di supporto: il PIO (Parallel Input/Output Controller), il SIO (Serial Input/Output Controller), il CTC (Counter/Timer Circuit), il DMA (Direct Memory Access Controller) e il DART (Dual Asynchronous Receiver/Transmitter).

Piedinatura e segnali

La fig. 1 illustra la funzionalità e la disposizione dei 40 pin dello Z80. Si riconoscono innanzi tutto il terminale di alimentazione (+5 V, pin 11) e quello di massa (GND, pin 29). Il pin 6 (CLK) è il terminale a cui deve essere inviato il segnale di clock, il cui periodo minimo, per la versione base, è di 0,4 µs. Gli altri terminali sono raggruppati secondo la loro funzionalità.

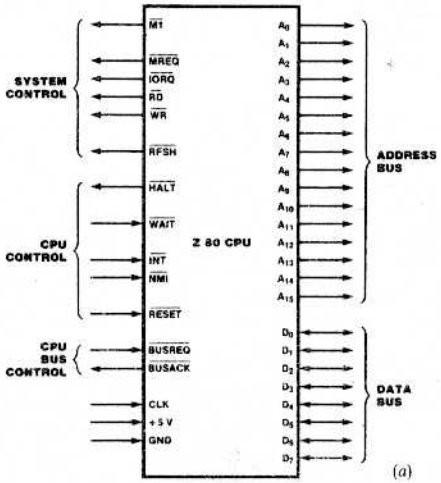


Fig. 1 - (*a*) Funzionalità e (*b*) disposizione dei terminali dello Z80.

Le *linee di bus* comprendono:

A_0-A_{15} (*Address Bus*): sono le 16 linee di indirizzo, di cui A_0 è la meno significativa. Sono uscite di tipo tri-state attive alte che vengono usate per indirizzare la memoria (massima capacità 64 kbyte) ed i dispositivi di I/O. In questo secondo caso vengono impiegate solo le 8 linee meno significative (A_0-A_7), con la possibilità di gestire pertanto 256 dispositivi di I/O.
Durante l'operazione di rinfresco, che viene eseguita direttamente dalla CPU, le 7 linee A_0-A_6 contengono l'indirizzo della riga di memoria dinamica da rinfrescare.

D_0-D_7 (*Data Bus*): sono le otto linee usate per lo scambio di dati con la memoria e i dispositivi di I/O. Sono linee bidirezionali (di ingresso e di uscita), le cui uscite sono di tipo tri-state attive alte.

Le linee di uscita per il *controllo del sistema* comprendono:

$\overline{M1}$ (*Machine Cycle One*, primo ciclo macchina): questa uscita si porta nello stato attivo, ossia al livello basso, ogni qualvolta la CPU legge un codice operativo in memoria, cioè durante il primo ciclo macchina (fase di fetch) di ogni istruzione (vedi Temporizzazioni). Se il codice operativo è di due byte, il segnale va basso per due volte consecutive. $\overline{M1}$ viene inoltre attivato insieme con $\overline{IORQ}$ durante il ciclo di riconoscimento di una interruzione (*vedi* Interruzioni).

$\overline{MREQ}$ (*Memory Request*, richiesta di memoria): segnala, attivandosi, che sul bus degli indirizzi è presente un indirizzo valido per un'operazione di lettura o scrittura in memoria.

$\overline{IORQ}$ (*I/O Request*, richiesta di I/O): segnala, passando al livello basso, che sulle otto linee basse degli indirizzi è presente un indirizzo valido per operazioni di scrittura o lettura in dispositivi di I/O. Viene inoltre attivato con $\overline{M1}$ durante il ciclo di riconoscimento di una interruzione.

$\overline{RD}$ (*Read*, lettura): la sua attivazione (livello basso) indica che la CPU è pronta a leggere un dato dalla memoria o da un dispositivo di I/O.

$\overline{WR}$ (*Write*, scrittura): indica che la CPU ha presentato sul bus un dato da trasferire in memoria o ad un dispositivo di I/O.

$\overline{RFSH}$ (*Refresh*, rinfresco): viene attivato, insieme con $\overline{MREQ}$, nel ciclo di rinfresco, durante il quale le sette linee di indirizzo più basso A_0-A_6 contengono l'indirizzo della riga di memoria dinamica da rinfrescare.

Le linee di controllo della CPU sono, ad eccezione di *HALT*, linee di ingresso che agiscono sulla CPU secondo diverse modalità. Sono tutte attive basse e comprendono:

$\overline{HALT}$: indica, quando è basso, che la CPU, in seguito ad una istruzione di *HALT*, si trova in uno stato di attesa, dal quale può uscire solo tramite i segnali di interruzione $\overline{NMI}$ o $\overline{INT}$ e, ovviamente di $\overline{RESET}$. Nello stato di attesa la CPU esegue in continuazione una particolare istruzione, la NOP (*no operation*), con lo scopo di rinfrescare la memoria dinamica.

$\overline{WAIT}$ (attesa): quando è basso, questo segnale provoca l'inserimento di cicli di attesa nel ciclo macchina della CPU, che rimane nello stato di attesa finché il segnale non torna nello stato alto di riposo.

$\overline{NMI}$ (*Non-Maskable Interrupt*, interruzione non mascherabile): il segnale, attivo basso, comunica alla CPU la richiesta di interruzione non mascherabile da parte di dispositivi esterni. Terminata l'istruzione in corso, la CPU viene forzata a ripartire dalla locazione 66_H.

$\overline{INT}$ (*Interrupt Request*, richiesta di interruzione): la richiesta di interruzione mascherabile, che ha priorità inferiore a $\overline{NMI}$, viene accettata solo se si è provveduto in precedenza ad abilitare le interruzioni via software, tramite l'istruzione IE. La CPU segnala l'accettazione dell'interruzione attivando $\overline{M1}$ e $\overline{IORQ}$ = INTACK

$\overline{RESET}$: inizializza la CPU nel seguente modo: 1) azzera il Program Counter (PC), sicché la CPU ripartirà dall'istruzione contenuta nella locazione 00_H; 2) disabilita le interruzioni e le predispone nel Modo 0 di funzionamento (*vedi* Interruzioni); 3) azzera il registro delle interruzioni I e il registro di rinfresco R (*vedi* Registri); 4) durante il ciclo di reset le linee dati e indirizzi sono poste in alta impedenza e le linee di controllo sono nello stato di riposo. Si noti che il $\overline{RESET}$, per poter essere riconosciuto dalla CPU, deve rimanere attivo per almeno tre cicli di clock completi.

Infine le linee di controllo dei bus comprendono:

BUSREQ (*Bus Request*, richiesta dei bus): segnala, portandosi al livello basso, la richiesta di un dispositivo periferico di gestire i bus degli indirizzi e dei dati. La CPU accetta la richiesta al termine del ciclo macchina in corso ponendo in alta impedenza le linee di indirizzo, le linee dei dati e le linee $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ e $\overline{WR}$, che in questo modo possono essere controllate direttamente dalla periferica. La CPU cessa inoltre di operare il rinfresco della memoria, che deve pertanto essere gestito dalla periferica.

BUSAK (*Bus Acknowledge*, riconoscimento della richiesta dei bus): segnala che la CPU ha lasciato la gestione dei bus; da questo momento la periferica richiedente può assumerne il controllo.

Temporizzazioni

L'esecuzione di un'istruzione avviene secondo una successione di operazioni elementari, *i cicli macchina*, la cui tipologia comprende:

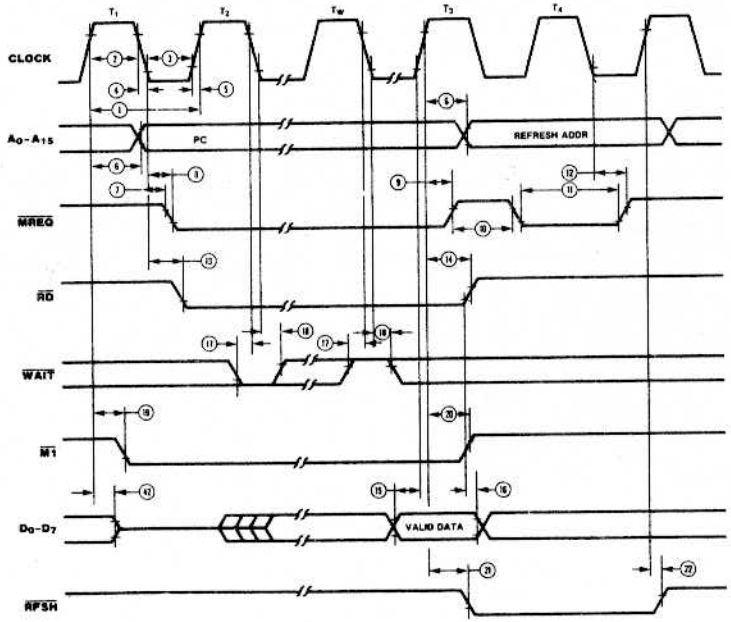
- prelievo del codice operativo dalla memoria;
- lettura e scrittura in memoria;
- lettura e scrittura nei dispositivi di I/O;
- riconoscimento dell'interruzione;
- riconoscimento della richiesta di bus;
- ciclo di halt;
- ciclo di reset.

Verranno ora esaminati gli andamenti nel tempo (*timing*) dei segnali generati dalla CPU in alcuni dei cicli macchina fondamentali; le temporizzazioni degli altri cicli macchina verranno trattate nei paragrafi successivi. Nei fogli tecnici inseriti alla fine della Nota Applicativa sono riportati i valori dei parametri dinamici indicati nei diagrammi temporali.

FETCH

Prelievo del codice operativo. In fig. 2 è illustrata la temporizzazione relativa alla fase di prelievo del codice operativo (fetch) da parte della CPU, ovvero relativa al *primo ciclo macchina* (CM1) di ogni istruzione.

Come si vede la lettura del codice operativo ha una durata di quattro periodi di clock (non si



NOTE: T_W-Wait cycle added when necessary for slow ancilliary devices.

Fig. 2 - Ciclo di lettura del codice operativo (fase di fetch).

consideri lo stato di attesa T_W). Supponendo allora di lavorare con una frequenza di clock di 2,5 MHz e quindi con un periodo di 0,4 µs, si ottiene per il ciclo di fetch il valore 1,6 µs. Nel primo ciclo T_1, sul fronte di salita del clock, la CPU pone il contenuto del Program Counter (PC) sul bus indirizzi (A_0-A_{15}); sul diagramma viene evidenziato l'intervallo di tempo (6) dopo il quale gli indirizzi possono ritenersi validi. Dopo mezzo periodo, sul fronte di discesa di T_1 va basso $\overline{MREQ}$, che pertanto può essere usato come segnale di abilitazione o di strobe per la memoria. L'attivazione quasi contemporanea di $\overline{RD}$ indica che i dati possono essere presentati dalla memoria sul bus dati.

Successivamente sul fronte negativo di T_2 la CPU controlla la linea $\overline{WAIT}$ e, se questa è alta (inattiva), legge con il fronte positivo di T_3 il dato nel frattempo presentato dalla memoria sul bus dati (D_0-D_7). Se viceversa la linea $\overline{WAIT}$ è tenuta bassa, la CPU inserisce prima della fase T_3 un ciclo di attesa T_W. Sul fronte di discesa di quest'ultimo viene nuovamente campionato il segnale di $\overline{WAIT}$ e, solo se questo è alto, si passa alla fase T_3; in caso contrario viene inserito un altro ciclo di wait, e così via.

Si noti come la lettura del codice operativo sia segnalata dall'attivazione della linea $\overline{M1}$, che scende al livello basso sul fronte positivo di T_1 e rimane bassa per i due primi cicli di clock.

Durante le fasi T_3 e T_4 la CPU internamente provvede alla decodifica del codice operativo ed esternamente opera il rinfresco della memoria dinamica. Più precisamente sulle sette linee meno significative del bus indirizzi pone il contenuto di un apposito *registro di rinfresco* (R), che viene di volta in volta incrementato, ed abbassa la linea $\overline{RFSH}$. Anche in questa fase, stabilizzatisi gli indirizzi, viene attivata la linea $\overline{MREQ}$. I 128 indirizzi che, uno per ogni primo ciclo macchina (CM1), vengono rinfrescati successivamente, corrispondono alle righe della memoria dinamica.

Lettura e scrittura in memoria. Queste operazioni durano ciascuna tre cicli di clock, se non intervengono cicli di attesa T_W; le temporizzazioni relative sono riportate in fig. 3.

Il ciclo di lettura è sostanzialmente simile a quello di prelievo del codice operativo; si notano tuttavia due differenze: la lettura dei dati presentati dalla memoria avviene sul fronte di discesa di T_3,

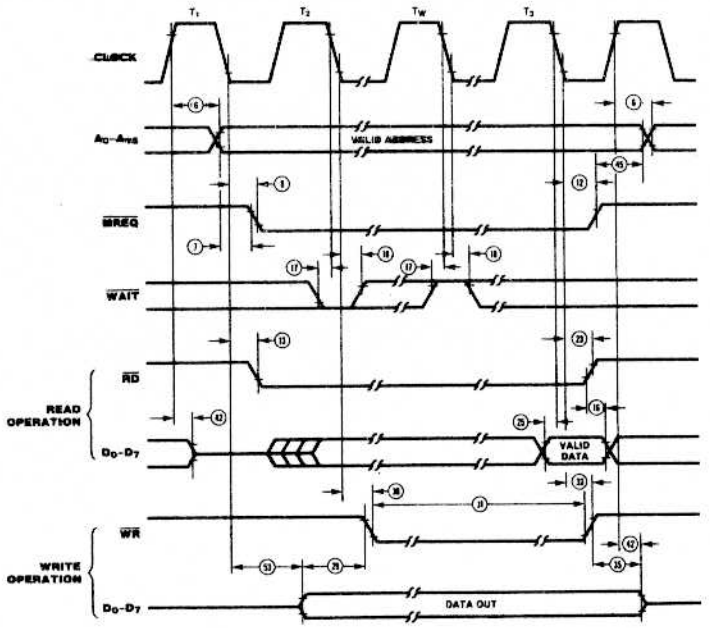
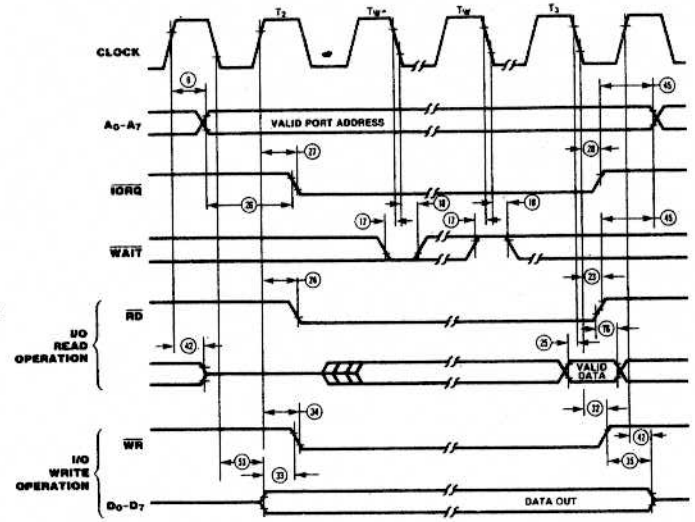


Fig. 3 - Ciclo di lettura e di scrittura in memoria.



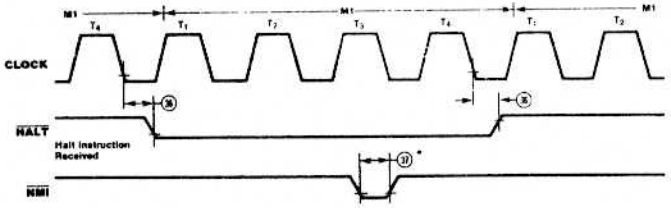
NOTE: T_W* = One Wait cycle automatically inserted by CPU.

Fig. 4 - Ciclo di ingresso e di uscita.

anziché su quello di salita, e la linea $\overline{M1}$, non indicata in figura, rimane inattiva (alta).
Per il ciclo di scrittura, sempre riportato in fig. 3, al posto di $\overline{RD}$, è interessata la linea $\overline{WR}$, che viene attivata (livello basso) dopo che i dati posti dalla CPU sul bus si sono stabilizzati. Il segnale $\overline{WR}$ può pertanto essere usato come segnale per l'abilitazione alla scrittura per la memoria. Nelle operazioni di lettura e scrittura l'introduzione di cicli di attesa T_W avviene secondo le stesse modalità viste per il ciclo di fetch.

Lettura e scrittura nei dispositivi di I/O. Queste operazioni, chiamate anche cicli di *ingresso* (lettura) e di *uscita* (scrittura), sono sostanzialmente analoghe alle precedenti che operano sulla memoria; al posto di $\overline{MREQ}$ viene però attivata la linea $\overline{IORQ}$ (*vedi* fig. 4).
Inoltre, poiché i dispositivi periferici in genere sono più lenti delle memorie, viene automaticamente introdotto un ciclo di attesa T_W^*. Naturalmente ulteriori cicli di attesa possono essere aggiunti tramite la linea $\overline{WAIT}$.

Ciclo di HALT. Questo ciclo, illustrato in fig. 5, si verifica dopo che la CPU ha decodificato l'istruzione HALT. Una volta terminato il ciclo CM1 relativo al fetch di questa istruzione, la CPU esegue in continuazione cicli NOP (4 periodi di clock) e contemporaneamente mantiene attivata la



NOTE: $\overline{INT}$ will also force a Halt exit.

Fig. 5 - Ciclo di HALT.

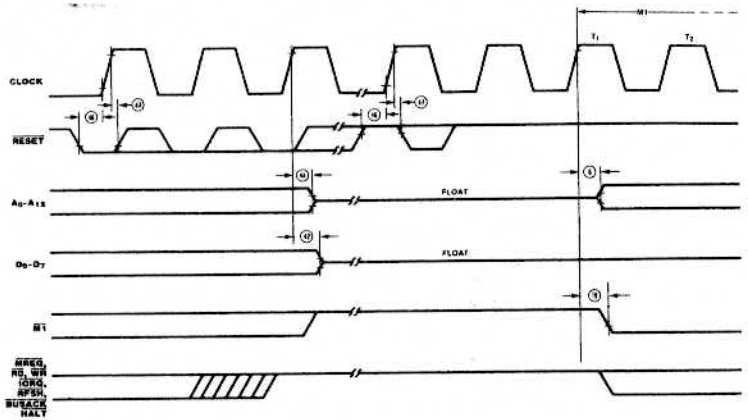


Fig. 6 - Ciclo di Reset.

linea di uscita $\overline{HALT}$.
Da questo stato la CPU può uscire solo se vengono attivati i segnali di interruzione $\overline{NMI}$ o $\overline{INT}$.

Ciclo di RESET. In fig. 6 è riportata la temporizzazione del ciclo di reset, che ha inizio solo quando la linea $\overline{RESET}$ rimane attivata per almeno tre cicli di clock. Durante questo ciclo le linee di bus vengono portate in stato di alta impedenza (fluttuante) e le linee di controllo nello stato di riposo. Dopo che il $\overline{RESET}$ è tornato inattivo, deve trascorrere un intervallo di tre cicli di clock prima che la CPU riprenda il normale funzionamento, ricominciando con un ciclo di fetch alla locazione 0000_H.

Registri

Lo Z80 presenta (*vedi* fig. 7) un gruppo di registri di uso speciale e due banchi di registri di uso generale, uno principale e uno secondario, scambiabili fra di loro.

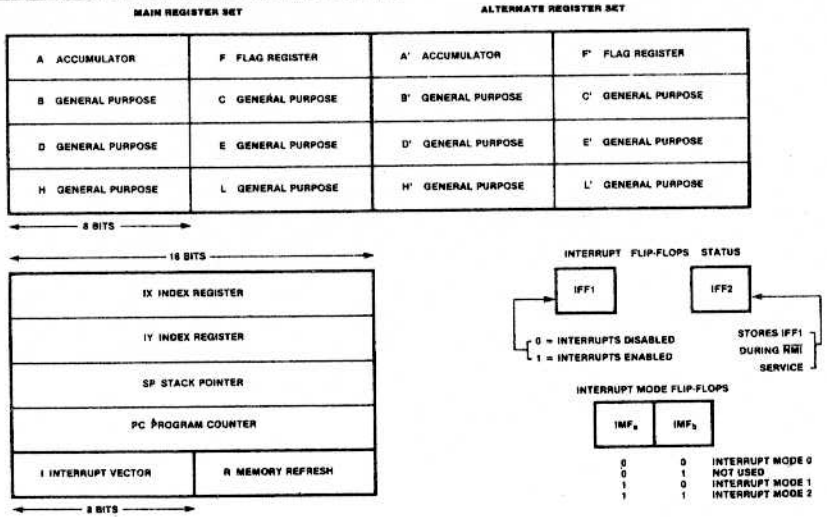


Fig. 7 - I registri della CPU.

I registri di uso speciale comprendono:

Program Counter (PC). Il contatore di programma, a 16 bit, contiene, come è noto, l'indirizzo dell'istruzione da prelevare in memoria. L'aggiornamento del PC avviene dopo la decodifica del codice operativo dell'istruzione prelevata, alla fine del ciclo di fetch CM1; pertanto il PC contiene già l'indirizzo di una nuova istruzione mentre è ancora in corso la fase di esecuzione dell'istruzione precedente.

Stack pointer (SP). Il puntatore di stack, anch'esso a 16 bit, contiene l'indirizzo dell'ultima locazione occupata dall'area di stack (*vedi* istruzioni PUSH e POP).

Index register IX e IY. I due registri indice vengono usati nelle istruzioni con indirizzamento indicizzato. Essi contengono l'indirizzo base a cui deve essere sommato il *displacement* (spiazzamento) per ottenere l'indirizzo effettivo di un operando.

Interrupt Register (I). Questo registro, a 8 bit, viene usato per memorizzare la parte alta del vettore di interruzione nel modo 2 (*vedi* Interruzioni).

Refresh register (R). Il registro contiene l'indirizzo di rinfresco per la memoria dinamica. Degli 8 bit contenuti, vengono utilizzati i 7 meno significativi.

Registri di flag (F e F'). I *registri di stato* contengono i flag secondo la disposizione illustrata in fig. 8, dove con b_0 si intende il bit meno significativo e con b_7 il più significativo. Il significato dei bit di flag è il seguente:

b_7							b_0
S	Z	X	H	X	P/V	N	C

Fig. 8 - Registro di stato.

C (carry, riporto): $C = 1$ quando un'operazione genera un riporto/prestito dal/al bit più significativo (b_7) dell'operando o del risultato. Viene inoltre usato come nono bit in operazioni di scorrimento e rotazione.
N (addizione/sottrazione): $N = 1$ dopo un'operazione di sottrazione.
P/V (parità e overflow): nelle operazioni logiche indica la parità del risultato; più precisamente $P = 1$ se il numero degli 1 presenti nel risultato è pari, $P = 0$ se è dispari.
Nelle operazioni aritmetiche $V = 1$ indica che si è verificato un overflow ed il risultato pertanto non è corretto.
H (half carry, riporto parziale): $H = 1$ se un'operazione produce un riporto/prestito dal/al quarto bit (b_3) dell'accumulatore. Viene usato nelle operazioni in formato BCD.
Z (zero): $Z = 1$ se il risultato di un'operazione è 0, altrimenti $Z = 0$.
S (segno): $S = 1$ se il risultato è negativo (in complemento a 2 l'MSB di un numero negativo è per l'appunto 1), altrimenti è $S = 0$.
X: sono flag non significativi.

I registri di *uso generale* o di *lavoro* sono disposti in un banco principale e in uno ausiliario, perfettamente uguali e fra loro intercambiabili mediante istruzioni. Sono tutti registri a 8 bit e fra questi si distingue per il suo particolare ruolo l'accumulatore (A); gli altri sono indicati con B, C, D, E, H, L.
Possono essere usati a coppie, BC, DE, HL (e i corrispondenti BC', DE', HL'), in modo da formare registri a 16 bit. In questo caso il byte più significativo del dato (HI) risiede nel primo registro, B o D o H, il meno significativo (LO) nel secondo registro, C o E oppure L.

Istruzioni

Il set di istruzioni dello Z80 comprende istruzioni di lunghezza variabile da un byte (solo codice operativo) ad un massimo di 4 byte.
Sono riportate nelle pagine finali della NA, suddivise per gruppi:

- caricamento a 8 bit;
- caricamento a 16 bit;
- scambio, trasferimento e ricerca di gruppo;
- aritmetiche e logiche a 8 bit;
- aritmetiche di uso generale e controllo della CPU;
- aritmetiche a 16 bit;
- rotazione e scorrimento;
- manipolazione di bit;
- salto;
- chiamata e ritorno;
- ingresso e uscita.

Per ogni istruzione vengono forniti il codice mnemonico, la descrizione in formato simbolico, lo stato dei flag, il codice operativo (in binario ed esadecimale) ed inoltre il numero dei byte, dei cicli macchina (M) e degli stati o periodi di clock (*T*), oltre ad un commento che permette di completare il codice stesso.

Si noti il significato delle parentesi presenti nel codice mnemonico e nella descrizione simbolica. L'espressione HL, ad esempio, indica il contenuto della coppia di registri HL, mentre (HL) indica il contenuto della locazione puntata da HL, ovvero della locazione il cui indirizzo è contenuto in HL. Così mentre *nn* rappresenta un dato di due byte, l'espressione (*nn*) indica il contenuto della locazione di indirizzo *nn*.
Per il significato dei simboli usati nella colonna dei flag si veda la notazione riportata nell'ultima pagina dei fogli delle istruzioni. In ogni caso la doppia freccia indica che il flag assume un valore coerente con i risultati dell'operazione; il punto indica viceversa che il flag è rimasto inalterato; la *X* significa che il suo valore è privo di significato.

Caricamento ad 8 bit. Queste istruzioni consentono il trasferimento dei dati fra memoria e registri e fra registri stessi. Vengono tutte indicate con la sigla LD, abbreviazione di LOAD (caricare). Si noti che nel codice mnemonico viene sempre indicato per primo il registro (o la locazione) *destinazione* e poi il *sorgente*.
La prima istruzione **LD r, r'** carica nel registro *r* il contenuto del registro *r'*. I registri possono essere qualsiasi fra A, B, C, D, E, H, L e ciascuno è contraddistinto da un suo codice, indicato nella colonna dei commenti.
Il codice operativo, in cui la posizione dei bit è indicata dai numeri 7...0 posti in testa alla colonna, varia ovviamente con i registri usati.
Ad esempio impiegando i registri A e B, l'istruzione diventa

Mnemonico	7 6 5 4 3 2 1 0	Hex	
LD A, B	0 1 1 1 1 0 0 0	78	[1]

L'istruzione, della lunghezza di un byte, comprende un solo ciclo macchina, la lettura del codice operativo, della durata di 4 periodi di clock.
L'istruzione successiva **LD r, n** è di tipo immediato e carica nel registro *r* il dato *n*, ad 8 bit. Così se si deve caricare in B il dato esadecimale $A3_H$, l'istruzione diviene

Mnemonico	7 6 5 4 3 2 1 0	Hex	
LD B, A3H	0 0 0 0 0 1 1 0	06	[2]
	1 0 1 0 0 0 1 1	A3	

Poiché questa istruzione è composta da un ciclo di lettura del codice operativo (durata 4*T*) e da un ciclo di lettura in memoria dell'operando (durata 3*T*), la sua durata complessiva è di 7*T*.
La terza istruzione è un esempio di indirizzamento indiretto da coppia di registri; **LD r, (HL)** carica in *r* il contenuto della locazione puntata dalla coppia di registri HL, ossia della locazione il cui indirizzo è contenuto nel registro a 16 bit HL. Il significato delle parentesi risulta così chiaro.
L'istruzione **LD r, (IX+d)** è di tipo indicizzato; essa carica in *r* il contenuto della locazione il cui indirizzo si ottiene sommando al contenuto del registro indice IX il valore dello spiazzamento *d*. Lo spiazzamento è un numero relativo di 8 bit espresso in complemento a 2, compreso pertanto fra -128 e $+127$.
L'istruzione **LD A, (BC)** carica nell'accumulatore il contenuto della locazione di memoria puntata dalla coppia di registri BC.
La successiva **LD A, (nn)** carica in A il contenuto della locazione di memoria il cui indirizzo è *nn*. Nella formazione del codice il primo byte costituisce il codice operativo, il secondo la parte bassa dell'indirizzo, il terzo la parte alta. Ad esempio se l'indirizzo è $203F_H$, la codifica avviene nel seguente modo

Mnemonico	Hex	
LD A, (203FH)	3A	
	3F	[3]
	20	

Quest'ordine di codifica, ossia prima il byte basso e poi quello alto, viene mantenuto in tutte le operazioni che coinvolgono due byte.
Infine le ultime istruzioni effettuano il trasferimento fra accumulatore A e, rispettivamente, registro interruzioni (I) e registro di rinfresco (R).
Dall'esame di questo primo gruppo di istruzioni si nota come il trasferimento fra la memoria ed un registro *r* qualsiasi può avvenire solo in maniera indiretta, usando come puntatori i registri indice e HL. Solo il trasferimento preferenziale fra memoria e accumulatore può essere realizzato in maniera diretta oppure in maniera indiretta usando come puntatori anche le coppie di registri BC e DE.
Caricamento a 16 bit. La prima delle istruzioni a 16 bit, **LD dd, nn**, è di tipo immediato e carica il dato a 16 bit *nn* nelle coppie di registri BC, DE, HL o nello Stack Pointer (SP), indicate genericamente con *dd*. Al solito la parte bassa del dato occupa il secondo byte dell'istruzione, dopo il codice operativo, e la parte alta il terzo byte. Così l'istruzione LD SP, 4500H si codifica come

Mnemonico	Hex	
LD SP, 4500H	31	
	00	[4]
	45	

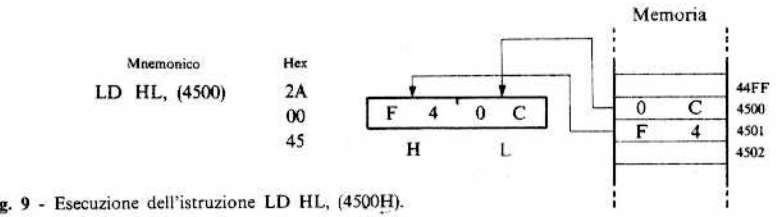


Fig. 9 - Esecuzione dell'istruzione LD HL, (4500H).

L'istruzione **LD HL, (nn)** carica in L il contenuto della locazione di indirizzo *nn* e in H quello di indirizzo *nn* + 1, come mostrato in fig. 9.
Le successive istruzioni non necessitano di particolare commento; si noti solo come il caricamento dell'SP possa avvenire o in modo immediato, o direttamente da memoria oppure tramite i registri indice e HL.
Le ultime istruzioni del gruppo gestiscono l'area di stack.
PUSH qq «spinge» nello stack il contenuto di una delle coppie di registri AF, BC, DE, HL. Più precisamente la parte alta dei registri va nella locazione puntata da SP − 1, la parte bassa nella locazione precedente, di indirizzo SP − 2. Ad esempio se HL contiene il dato A34F e SP contiene l'indirizzo 0409, l'istruzione PUSH HL opera come indicato in fig. 10; dopo l'esecuzione SP conterrà il nuovo indirizzo dello stack 0407.
L'istruzione **POP qq** compie l'operazione inversa, cioè estrae dallo stack l'ultimo dato immesso. Ad esempio supponendo, come illustrato in fig. 11, che lo stack pointer SP sia carico con l'indirizzo 450D (ultima locazione occupata dello stack), l'istruzione POP BC trasferisce FF in C e AA in B. Dopo l'esecuzione dell'istruzione SP, incrementato di due, punterà alla locazione 450F.

Scambio, trasferimento e ricerca di gruppo. Le prime due istruzioni, **EX DE, HL** e **EX AF, AF'**, operano lo scambio dei contenuti dei registri indicati. L'istruzione **EXX**, a sua volta, opera lo scam-

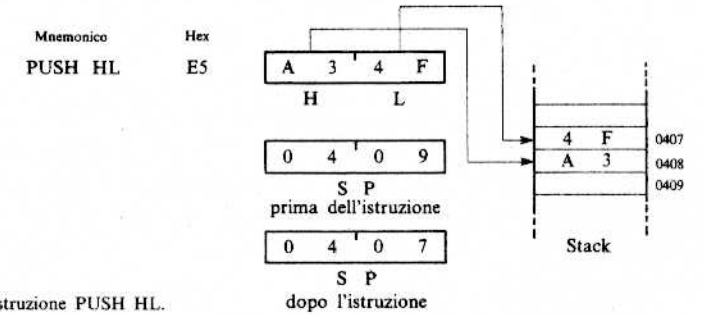


Fig. 10 - Istruzione PUSH HL.

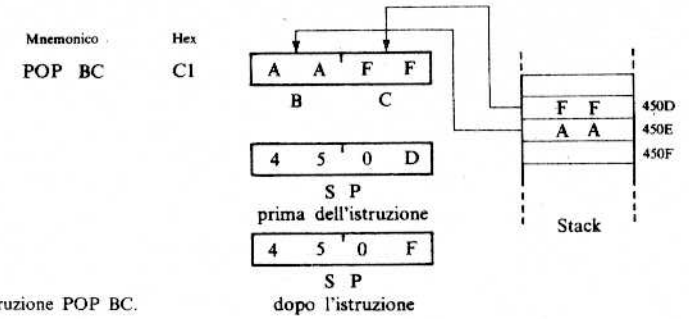


Fig. 11 - Istruzione POP BC.

bio fra i registri corrispondenti del banco principale e del banco ausiliare. Consente in questo modo di salvare nei registri ausiliari il contenuto dei registri principali, che possono così essere impiegati per un altro segmento del programma.
Le tre istruzioni seguenti operano lo scambio fra i registri indice e HL da un lato e le locazioni dello stack di indirizzo SP e SP + 1. Ad esempio se SP contiene l'indirizzo 0407, l'istruzione **EX (SP), HL** scambia il contenuto della locazione 0407 con quello di L e il contenuto della locazione 0408 con quello di H.
Segue un gruppo di istruzioni che consente il trasferimento di un blocco di dati da una zona della memoria ad un'altra. Implicano tre coppie di registri: il puntatore HL, che contiene l'indirizzo sorgente, il puntatore DC, che contiene l'indirizzo destinazione, e il contatore BC, che indica il numero delle locazioni da trasferire. Più precisamente l'istruzione **LDI** (*carica e incrementa*) trasferisce il contenuto della locazione puntata da HL nella locazione puntata da DE, incrementa i due registri che vengono così a puntare alle locazioni successive, e decrementa BC. Ripetendo questa istruzione fino a che BC = 0, si ottiene il trasferimento dell'intero blocco.
È proprio ciò che fa automaticamente l'istruzione successiva **LDIR** (*carica, incrementa e ripeti*). Le due istruzioni seguenti sono analoghe ma decrementano i puntatori, procedendo così nel trasferimento in senso opposto al precedente.
Le successive quattro istruzioni permettono di ricercare in un blocco di memoria il dato contenuto nell'accumulatore. **CPI** (*compara e incrementa*) compara il contenuto dell'accumulatore con quello della locazione puntata da HL, incrementa poi HL e decrementa BC. Se il confronto ha dato esito positivo, il flag *Z* va ad 1. **CPIR** è più completa in quanto compara, incrementa e ripete la comparazione finché il confronto ha dato esito positivo oppure il contenuto di BC è 0.
Le due istruzioni successive sono analoghe ma decrementano anziché incrementare il puntatore, ossia percorrono il blocco in senso decrescente.

Aritmetiche e logiche a 8 bit. L'operazione di addizione viene svolta da diverse istruzioni, tutte implicanti l'accumulatore. **ADD A, r** somma il contenuto del registro *r* a quello dell'accumulatore e pone il risultato nell'accumulatore stesso.
Vengono interessati praticamente tutti i flag di stato (*vedi* LE7-2).
ADD A, n è di tipo immediato e somma ad A l'operando *n*. Le successive istruzioni sono di tipo indiretto e indicizzato e consentono di sommare il contenuto dell'accumulatore a quello della locazione puntata dai registri. Le stesse varianti viste per ADD valgono per **ADC A, s** dove *s* può indicare sia un registro *r*, sia un dato *n*, sia (HL) ecc. Queste istruzioni eseguono l'addizione fra A, *s* ed il bit di carry (qui chiamato *CY*); vengono pertanto usate nell'addizione fra operandi costituiti da più byte, allorché occorre tener conto del riporto eventualmente generato dai byte meno significativi. I codici operativi sono gli stessi delle istruzioni ADD, dove si sostituisca il gruppo di bit 001 (in cornice) al gruppo 000 (pure in cornice) dei codici precedenti. Ad esempio

Mnemonico	7 6 5 4 3 2 1 0	Hex	
ADC A, 55H	1 1 0 0 1 1 1 0	CE	[5]
	0 1 0 1 0 1 0 1	55	

L'istruzione **SUB s** sottrae dall'accumulatore l'operando *s*, che al solito può essere il contenuto di un registro, di una locazione di memoria o un dato immediato. La **SBC A, s** sottrae dall'accumulatore, oltre all'operando *s*, anche il carry; viene pertanto usata nelle sottrazioni su operandi a più byte quando si debba tener conto del prestito richiesto dai byte meno significativi.
Seguono le operazioni logiche **AND s, OR s, XOR s**, che vengono effettuate fra i bit corrispondenti di A e di *s*; il risultato viene sempre posto in A.
L'istruzione **CP s** compara il contenuto di A con quello di *s*; più esattamente esegue la sottrazione A − *s* e se il risultato è 0 (i due contenuti sono uguali), porta ad 1 il flag di zero. Se viceversa A > *s*, la sottrazione fornisce *Z* = 0 e *C* = 0; infine nel caso A < *s*, la sottrazione genera un prestito e quindi *Z* = 0 e *C* = 1.
Seguono le istruzioni di incremento del contenuto di registro e di locazione puntata da registro ed infine quelle di decremento. Anche in queste, quando il risultato dell'operazione è 0, viene portato ad 1 il flag *C*.

Aritmetiche di uso generale e di controllo sulla CPU. L'istruzione **DAA** (*decimal adjustment accumulator*) opera l'aggiustamento decimale dell'accumulatore cioè ne converte il contenuto dal codice binario a BCD.
La seconda istruzione, **CPL**, esegue il complemento a 1 del contenuto dell'accumulatore, mentre la successiva, **NEG**, cambia il segno (negazione) al contenuto di A, ovvero esegue il suo complemento a 2.

CCF (*complement carry flag*) e **SCF** (*set carry flag*) rispettivamente complementano e portano a 1 il flag di carry.
Le istruzioni che seguono riguardano la gestione della richiesta di interruzione da parte di dispositivi periferici (*vedi* Interruzione); **DI** ed **EI** rispettivamente disabilitano ed abilitano le interruzioni mascherabili mentre **IM 0**, **IM 1** e **IM 2** selezionano il modo secondo cui deve operare l'interruzione.

Aritmetiche a 16 bit. Queste istruzioni non presentano particolari difficoltà di interpretazione; si noti solo che il registro HL è il più coinvolto da queste istruzioni e funge praticamente da accumulatore a 16 bit.
Rotazione e spostamento. Queste istruzioni sono chiaramente spiegate dagli schemi che le accompagnano. Le istruzioni di rotazione si distinguono in quelle di rotazione a sinistra (**RL**: *rotate left*) e in quelle di rotazione a destra (**RR**: *rotate right*). Possono inoltre includere nell'anello il bit di carry oppure non includerlo; in quest'ultimo caso sono dette *circolari* (**RLC** e **RRC**). Le rotazioni con carry al solito consentono di effettuare rotazioni su operandi costituiti da più byte.
Le istruzioni di rotazione possono operare sia sui registri, sia sulle locazioni di memoria, sia sull'accumulatore (**RLCA**, **RRCA**, **RLA**, **RRA**), nel qual caso sono più veloci durando un solo ciclo macchina.
Le istruzioni di spostamento (shift) possono essere di tipo aritmetico (**SLA** e **SRA**) o logico (**SRL**). **SLA** opera lo spostamento a sinistra di una posizione ed introduce uno 0 nella posizione meno significativa lasciata libera.
SRA opera uno spostamento a destra mantenendo lo stesso MSB; in questo modo conserva inalterato il segno dell'operando. Lo shift logico a destra **SRL** viceversa introduce uno 0 nella posizione più significativa.
Le ultime due istruzioni, **RLD** (*rotate left decimal*) e **RRD** (*rotate right decimal*), operano la rotazione dei due digit contenuti nella locazione di memoria puntata da HL, coinvolgendo nell'anello anche il digit meno significativo dell'accumulatore. Vengono usate per trattare dati in formato BCD.

Manipolazione di bit. Le istruzioni **BIT** consentono di testare un bit qualsiasi di un registro *r*, oppure di una locazione di memoria puntata dai registri indice o da HL. Il bit testato, la cui posizione è indicata da *b* (da 0 a 7), viene copiato nel flag *Z*.

Le istruzioni **SET** e **RES** forzano rispettivamente a 1 e a 0 il bit selezionato.

Salto. Sono istruzioni particolarmente interessanti perché permettono di interrompere la sequenzialità di un programma. Si distinguono in istruzioni di salto *assoluto* (JP: *jump*), che richiedono l'indirizzo *nn* della locazione a cui saltare, e istruzioni di salto *relativo* (JR: *jump relative*), in cui viene indicato di quante locazioni in avanti o indietro il programma deve saltare.
La prima istruzione **JP nn** esegue un salto all'indirizzo *nn*; al solito la parte bassa dell'indirizzo viene codificata nel secondo byte, subito dopo il codice operativo, la parte alta nel terzo byte.
Un salto relativo è invece realizzato dall'istruzione **JR e**, dove *e*, numero relativo di un byte espresso in complemento a 2 (quindi compreso fra -128 e $+127$), esprime il numero di locazioni di cui il programma deve avanzare o retrocedere, a partire dalla locazione che contiene il codice operativo. Ad esempio se l'istruzione JR 4 è memorizzata a partire dalla locazione 0405 (come è indicato in fig. 12), la sua esecuzione provocherà un salto a quattro locazioni più avanti, cioè alla 0409. A sua volta l'istruzione JR -4 provoca un salto di 4 locazioni indietro, cioè a 0401.
Si tenga presente che nel codificare l'istruzione occorre diminuire *e* di 2, poiché il PC, terminata la fase di fetch dell'istruzione di salto, viene automaticamente incrementato di 2 per puntare all'istruzione successiva. Pertanto nella prima istruzione $e-2=4-2=2$ (02_H) e nella successiva $e-2=$ $=-4-2=-6$ (FA_H).
I salti possono inoltre essere *incondizionati* o *condizionati* dal verificarsi di un evento precedente, indicato nel codice mnemonico da *cc*.
Le varie condizioni, caratterizzate ciascuna dal valore di un flag di stato, sono:

NZ: salta se il risultato precedente non è 0, ossia se il flag $Z=0$;
Z: salta se il risultato precedente è 0 ($Z=1$);
NC: salta se il risultato precedente non ha generato riporto, ossia se $C=0$;
C: salta se è stato generato riporto ($C=1$);
PO: salta se il risultato precedente ha parità dispari ($P/V=0$);
PE: salta se il risultato ha parità pari ($P/V=1$);
P: salta se il risultato precedente è positivo ($S=0$);
N: salta se il risultato è negativo ($S=1$).

L'ultima istruzione del gruppo, **DJNZ e**, è particolarmente utile nei cicli ripetuti (*loop*); essa decrementa il contenuto del registro B e compie un salto relativo solo se B è diverso da 0.

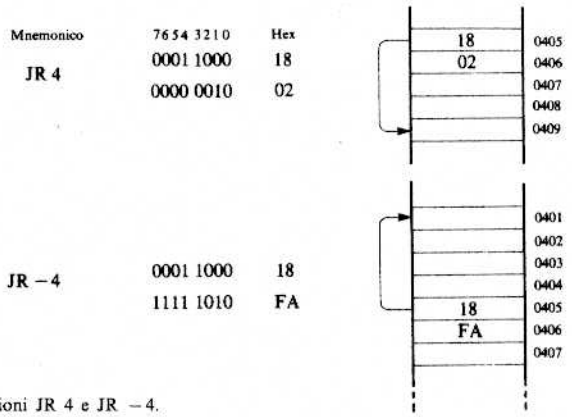


Fig. 12 - Istruzioni JR 4 e JR -4.

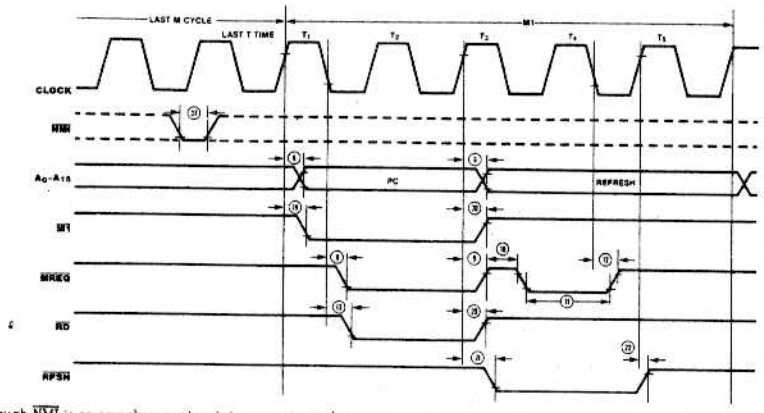
Chiamata e ritorno. Queste istruzioni consentono di abbandonare temporaneamente il programma principale (*main*) per eseguire *sottoprogrammi* (*vedi* Subroutine in NA7-2) e successivamente ritornare al programma principale.
L'istruzione **CALL**, nella forma incondizionata e condizionata, viene usata per chiamare una subroutine, ossia saltare all'indirizzo di partenza di un sottoprogramma. Automaticamente il contenuto del PC viene salvato nello stack; ciò consente, alla fine della subroutine, di riprendere l'esecuzione del programma principale esattamente da dove era stato abbandonato.
L'istruzione **RET** (*return*), che deve essere posta obbligatoriamente al termine di una subroutine, provvede a far riprendere il programma principale dall'istruzione successiva a CALL.
RETI e **RETN** sono le istruzioni di ritorno dalle subroutine che servono le interruzioni mascherabili e non mascherabili rispettivamente. Analogamente a **RET** devono costituire l'ultima istruzione di tali subroutine.
Le otto istruzioni **RST p** (*restart*) fanno ripartire il programma rispettivamente dalle locazioni 00_H, 08_H, 10_H, 18_H, 20_H, 28_H, 30_H e 38_H, a seconda di quanto specificato da *p*.

Ingresso e uscita. Queste istruzioni consentono il trasferimento di dati fra la CPU o le locazioni di memoria puntate e i dispositivi di I/O. Questi, come è noto, vengono indirizzati mediante gli 8 bit meno significativi (A_0-A_7) delle linee di indirizzi.
L'istruzione **IN A, (n)** trasferisce nell'accumulatore il dato del dispositivo di indirizzo *n*. **IN r, (C)** trasferisce nel registro *r* il dato del dispositivo I/O puntato dal registro C.
Le quattro istruzioni successive consentono il trasferimento di blocchi di dati provenienti dal dispositivo di I/O puntato da C nelle locazioni di memoria puntate da HL.
Analoghe sono le operazioni di uscita; ad esempio **OUT (n), A** trasferisce il contenuto dell'accumulatore nel dispositivo di I/O di indirizzo *n*. Le ultime quattro istruzioni provvedono al trasferimento di blocchi di dati.

Interruzioni

Lo Z80 consente due modalità di interruzione, la non mascherabile e la mascherabile; la richiesta avviene tramite l'attivazione di due linee distinte, $\overline{NMI}$ (*non-maskable interrupt*) e $\overline{INT}$ (*interrupt request*), il cui stato viene controllato dalla CPU al termine di ogni ciclo di istruzione.
L'interruzione non mascherabile (linea $\overline{NMI}$) ha priorità più elevata e viene incondizionatamente accettata dalla CPU; l'interruzione mascherabile (linea $\overline{INT}$) deve invece essere preventivamente abilitata via software mediante l'istruzione EI.

Interruzione non mascherabile. Viene segnalata alla CPU, come si è detto, dall'attivazione (livello basso) della linea $\overline{NMI}$. Dopo aver salvato il contenuto del PC nello stack, il μP abbandona il programma principale e salta alla locazione 66_H, che deve contenere l'inizio della routine di gestione dell'interrupt. Tale routine deve terminare con l'istruzione RTN, che, ripristinando nel PC il valore salvato nello stack, fa riprendere alla CPU



*Although NMI is an asynchronous input, to guarantee its being recognized on the following machine cycle, NMI's falling edge must occur no later than the rising edge of the clock cycle preceding T_{LAST}.

Fig. 13 - Temporizzazioni dell'interrupt non mascherabile.

l'esecuzione del programma dall'istruzione successiva all'interruzione.
In fig. 13 è illustrata la temporizzazione dell'interruzione non mascherabile.

Interruzione mascherabile. Questo tipo di interruzione è segnalata alla CPU dall'abbassamento della linea $\overline{INT}$. La sua accettazione è controllata da un flip-flop IFF_1, che nello stato 1 abilita l'interruzione e nello stato 0 la disabilita (*vedi* fig. 7). Un secondo flip-flop IFF_2 ha il compito di memorizzare lo stato di IFF_1.
All'inizializzazione del sistema (linea di $\overline{RESET}$ attiva), i due flip-flop vengono posti a 0 e inibiscono l'accettazione di ogni interruzione mascherabile. L'abilitazione può avvenire solo tramite l'istruzione EI, che porta entrambi i flip-flop a 1. Viceversa, l'istruzione DI li resetta entrambi a 0, disabilitando l'interruzione.
Allorché viene eseguito un interrupt non mascherabile, IFF_1 viene posto a 0, dopo che il suo contenuto è stato salvato in IFF_2. In questo modo viene impedita l'accettazione di un eventuale successivo interrupt mascherabile. L'ultima istruzione della routine di servizio dell'interrupt non mascherabile, RTN, provvede a ripristinare lo stato precedente di IFF_1, trasferendo in esso il valore salvato in IFF_2.
L'accettazione di una interruzione mascherabile porta a 0 entrambi i flip-flop, ragion per cui lo stato precedente non viene più ripristinato al termine della routine di servizio.

L'esecuzione dell'interruzione può seguire tre diverse modalità, selezionabili mediante le istruzioni IM 0, IM 1 e IM 2. Anche in questo caso due flip-flop, IMF_a e IMF_b sovrintendono alla selezione.
Il **modo 1** è del tutto simile ad una interruzione non mascherabile, solo che il restart, ossia la partenza della subroutine di servizio, avviene alla locazione 38_H, anziché alla 66_H.
Gli altri due modi sono di tipo *vettorizzato*. Più precisamente nel **modo 0** il dispositivo periferico che ha generato l'interrupt, non appena ha ricevuto il segnale di accettazione della richiesta di interruzione, deve porre sul bus dati un'istruzione, anche a più byte, che viene eseguita dalla CPU. Normalmente viene utilizzata una delle otto istruzioni di restart (RST).
Nel **modo 2** il periferico, in seguito al segnale di accettazione dell'interruzione generato dalla CPU, pone sul bus dati un byte, il cui bit meno significativo deve essere 0, che viene interpretato dalla CPU come un indirizzo. Più precisamente questo byte va a costituire la parte bassa di un indirizzo completo, la cui parte alta è stata precedentemente caricata via software nel registro interruzioni I. L'indirizzo così ottenuto è quello di una delle locazioni pari di una tabella che contiene gli indirizzi, ovviamente a due byte, delle routine di servizio.
Naturalmente la tabella degli indirizzi deve essere stata preventivamente caricata in memoria. In fig. 14 viene meglio chiarita la modalità di questo

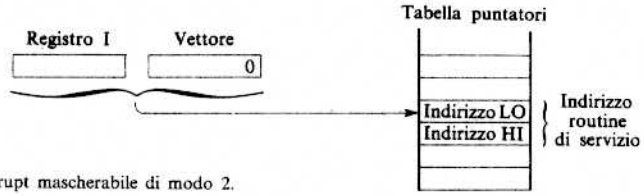
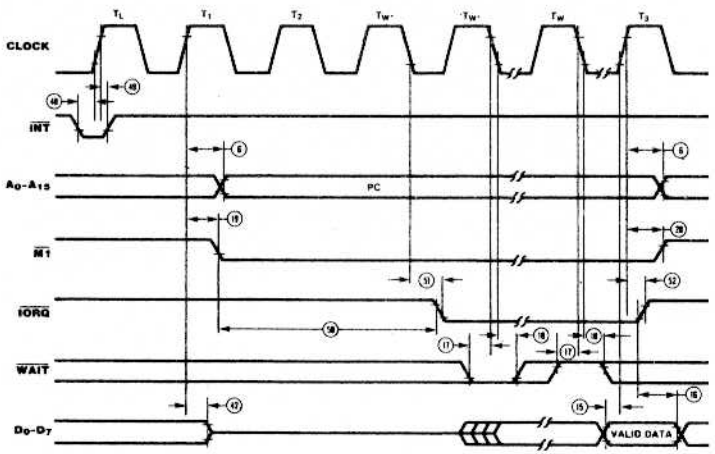


Fig. 14 - Interrupt mascherabile di modo 2.

indirizzamento.
In definitiva, fissato il contenuto di I, la tabella può essere composta da un massimo di 128 indirizzi e quindi può servire al massimo 128 dispositivi di I/O.
Si tenga infine presente che, per risolvere il problema della priorità fra più periferici, i dispositivi di I/O della famiglia Z80 sono predisposti per operare in *daisy-chain* (*vedi* par. 7.7).
Allorché una richiesta di interruzione mascherabile viene accettata, inizia un ciclo le cui temporizzazioni sono illustrate in fig. 15.
Sul fronte di salita dell'ultimo impulso di clock di ogni istruzione viene testata la linea $\overline{INT}$; se questa è bassa e se IFF_1 è a 1, inizia un particolare ciclo macchina CM1, durante il quale vengono attivati i segnali $\overline{M1}$ e $\overline{IORQ}$. Come si può vedere dai diagrammi, vengono automaticamente inseriti due cicli di attesa T_W^*, in modo da permettere al periferico, generalmente lento, di fornire il vettore di interruzione sul bus dati.

Richiesta del bus

In un sistema a μP può accadere che un periferico abbia bisogno di gestire i bus degli indirizzi e dei dati indipendentemente dalla CPU. Questo avviene ad esempio nelle operazioni di DMA, quando occorra trasferire quantità di dati direttamente dal periferico alla memoria senza passare attraverso la CPU, che rallenterebbe il trasferimento.
In questo caso il periferico attiverà la linea $\overline{BUSREQ}$, portandola bassa. Questa linea, che ha priorità più elevata di $\overline{NMI}$ e $\overline{INT}$, viene campionata dalla CPU sul fronte di salita dell'ultimo clock di ogni ciclo macchina; inizia allora il ciclo illustrato in fig. 16.
La CPU pone in alta impedenza le linee di bus e $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$, attivando contemporaneamente $\overline{BUSACK}$. Si noti che la linea $\overline{RFSH}$ viene disattivata; al rinfresco delle memorie dinamiche si deve pertanto provvedere dall'esterno.



NOTE: 1) T_L = Last state of previous instruction. 2) Two Wait cycles automatically inserted by CPU(*).

Fig. 15 - Temporizzazioni dell'interrupt mascherabile.