Per descrivere lo Z80 faccio uso della distinzione tra architettura e organizzazione.  
Con architettura ci si riferisce all’insieme di attributi e risorse visibili al programmatore [Buso, p.91] o che hanno un impatto decisivo sull’esecuzione del programma[Comp. Organization and Archietcture p26].  
Mentre con organizzazione ci si riferisce alle effettive unità operative e alle loro connessioni per sviluppare l’architettura[Comp. Organization and Archietcture p26].

A questo punto, si può analizzare lo Z80 secondo quattro elementi strettamente connessi tra loro: la piedinatura, o pinout; le risorse visibili al programmatore, detta per sineddoche architettura; la gestione degli eventi di interrupt; il set di istruzioni. Inizialmente farò riferimento a quanto riportato sul datasheet[rif. Datasheet] e il manuale del microprocessore[rif. Manuale].

Pinout

Figura XXX – Pinout [datasheet p.1]

Lo Z80 ha la possibilità di indirizzare 64kB di memoria o registri di interfaccia con le periferiche di input e output (I/O). Usa un bus a 16 bit denominato address bus A che ha la possibilità di andare in alta impedenza (tecnologia three-state) per permettere l’uso dello stesso bus ad altri microprocessori o ad un DMA. Per comunicare con la memoria e gli altri dispostivi usa un bus bidirezionale dati da 8 bit detto Data Bus D, anch’esso three-state per lo stesso motivo di A.

Gli altri pin sono organizzati in tre gruppi e sono tutti attivi bassi.

Il gruppo System Control raggruppa i segnali d’uscita che permettono il controllo del sistema collegato alla CPU.  
Per gestire la lettura e scrittura (R/W) con l’esterno fa uso di due pin distinti: nRD e nWR, con cui seleziona il verso della comunicazione. Quando nRD, abbrev. di Read, è attivo, la periferica selezionata deve mettere sul bus D il dato richiesto. Mentre quando nWR, abbrev. di Write, è attivo, la CPU ha fornito sul bus D un dato valido e comunica alla periferica che può essere letto.  
Lo Z80 indirizza gli I/O allo stesso modo della memoria, per cui vede quest’ultimi come dei registri indirizzabili. Per distinguere tra le operazioni di R/W verso memoria o I/O utilizza due pin: nMREQ e nIORQ. Con nMREQ, abbrev. di Memory request, la CPU comunica che l’indirizzo presente su A è utilizzabile per indirizzare una memoria. Per cui può essere usato come segnale di abilitazione delle memorie. Mentre con nIORQ, abbrev. di I/O request, si possono comunicare due eventi. Il primo è la controparte di nMREQ, cioè comunica all’I/O che l’indirizzo presente sul bus A è valido per essere letto. Il secondo è per comunicare il ricevimento di una richiesta di interruzione, interrupt acknowledge. Durante l’interrupt acknowledgement non viene attivato il pin nRD sebbene la CPU stia richiedendo alla periferica un dato poiché si vuole che non venga fatta confusione un’operazione di normale lettura. Al contrario si attiva il pin nM1, abbrev. di Machine cycle one, che comunica proprio quest’avvenimento. Lo stesso pin viene usato per segnalare che l’operazione di lettura della memoria che si sta effettuando è un’operazione di recupero dell’istruzione, instruction fetch, o come viene chiamata nel datasheet opcode fetch. Quest’informazione è utile per le periferiche, specialmente quelle della famiglia Z80, che riconoscono lo stato della macchina.  
L’ultimo pin di questo gruppo è nRFSH, abbrev. di Refresh. Questo pin comunica alle memorie collegate ad A che sul bus è presente un indirizzo per le operazioni di refresh. Quest’operazione è indispensabile per le Dynamic RAM, DRAM, che altrimenti perderebbero il loro contenuto. La presenza di una logica di refresh già integrata nel microprocessore è stato un altro punto di forza dello Z80 rispetto all’Intel 8080. L’indirizzo di refresh presenta solo i 7 bit meno significativi poiché al tempo non era utile avere più bit per fare il refresh delle DRAM sul mercato.

Il secondo gruppo, CPU control, invece raggruppa quattro ingressi e un’uscita che controllano il comportamento del microprocessore.  
Il pin nRESET, se mantenuto attivato per almeno tre cicli di clock, avvia la procedura di reset del microprocessore che azzera tutti i registri. Durante il reset della CPU, i bus A e D vanno in alta impedenza e i segnali di controllo diventano inattivi.  
Con una particolare istruzione, si può portare la CPU in uno stato di attesa in cui continua ad eseguire operazioni nulle per continuare a svolgere il refresh della memoria. La CPU comunica l’ingresso in questo stato attivando l’uscita nHALT. Per uscire da questo stato bisogna fornire alla CPU un interrupt attivo.  
Lo Z80 distingue tra due tipi di interruzioni, interrupt: quelli mascherabili via software che vengono richiamati abbassando il pin nINT; e quelli non mascherabili sul pin nNMI. I secondi hanno una priorità maggiore degli altri e vengono serviti per primi. Gli interrupt vengono serviti solamente alla fine di un’istruzione.  
L’ultimo pin del gruppo è l’ingresso nWAIT. Questo pin viene attivato dalle memorie o dagli I/O su cui sta avvenendo un’operazione R/W. La periferica abbassa questo pin per comunicare che non è pronta ad accettare il dato e lo mantiene abbassato. La CPU da parte sua rimane in un ciclo di attesa fintato che non vede disattivarsi il segnale. Il mantenere la CPU in uno stato di attesa per periodi prolungati può portare a problemi di refresh poiché la CPU non lo effettua con la periodicità corretta.

L’ultimo gruppo gestisce la possibilità di avere più microprocessori o un DMA collegati agli stessi bus e si chiama CPU bus control.  
In questo gruppo ci sono due segnali: nBUSREQ, abbrev. di Bus request, e nBUSACK, abbrev. di Bus acknowledge.  
Con il primo segnale una periferica richiede di poter usare i bus in modo esclusivo gestendo lei gli indirizzamenti, i dati e i controlli. Questo evento ha una priorità maggiore di nNMI e viene servito alla fine dei cicli macchina delle istruzioni.  
Quando la CPU accetta la richiesta, lo comunica attivando il pin nBUSACK e mette i bus A, D e il gruppo System Control in alta impedenza. Durante questo periodo la CPU rimane in attesa non svolgendo nessuna attività per poi ripartire dallo stesso punto in cui si era interrotta. Come per i cicli di nWAIT, periodi prolungati di nBUSREQ possono portare a problemi di refresh poiché la CPU non effettua più l’operazione richiesta con la corretta periodicità.

Architettura

Figura X – Diagramma a blocchi [datasheet p.2]

L’architettura dello Z80 è di tipo CISC, Complex Instruction Set Computer, perché presenta 158 istruzioni di lunghezza variabile la cui esecuzione può durare da uno a cinque cicli macchina [Buso p.96]. Inoltre ci sono istruzioni che possono venire eseguite più volte fino al verificarsi di una condizione. Ad esempio una di queste è l’istruzione LDI che esegue una copia di una locazione di memoria in un'altra avanzando di locazione in locazione e decrementando un contatore. Quando quest’ultimo arriva a 0 si termina l’esecuzione e si passa all’istruzione successiva.

Lo Z80 divide il tempo in cui esegue un’istruzione in due modi: cicli di clock, detti T-cycles, che vanno da un fronte positivo del segnale di clock al successivo, e cicli macchina, detti M-cycles, che raggruppano i T-cycles in base alle operazioni che compie. Le operazioni di R/W su memoria o I/O, interrupt acknowledgement e opcode fetch durano tutte un singolo ciclo macchina che però ha durata variabile da 3 a 5 cicli di clock.  
I cicli macchina vengono numerati da 1 a 6 in base alla durata di un’operazione. Per esempio, tutte le operazioni aritmetiche a 8 bit tra l’accumulatore e un altro registro durano tutte 1 M e 4 T. Mentre l’istruzione EX (SP), HL , che scambia il contenuto di HL con la testa della pila, dura 6 M e 23 T.  
M1 è sempre un ciclo di opcode fetch. Nell’esecuzione delle istruzioni, ci possono essere dei cicli macchina di lunghezza variabile da 1 a 6 T per svolgere delle operazioni interne. Per questo motivo nel datasheet sono fornite le durate delle istruzioni sia in M-cycles sia T-cycles.

Lo Z80 usa la tecnica dell’I/O isolato poiché per comunicare con gli I/O usa delle istruzioni dedicate. [Comp. Organization and Archietcture p262] Inoltre il datasheet afferma che è usabile solamente la parte meno significativa dell’indirizzo per selezionare i dispositivi che di conseguenza possono esser al massimo 256.

Le operazioni aritmetiche e logiche a 8 bit sono basate sull’accumulatore. Per cui hanno sempre l’accumulatore come uno degli operandi e sempre lo stesso registro come risultato. Mentre le operazioni a 16 bit usano i registri HL, IX o IY allo stesso scopo.  
Gli indirizzi, come un generale i valori a 16 bit, vengono letti e scritti nella memoria secondo la tipologia Little Endian in cui all’indirizzo minore corrisponde la parte bassa del dato.

Come si vede dalla figura X, all’interno dello Z80 vi sono tre blocchi adibiti al controllo della CPU scandendo il ritmo delle operazioni e decodificando le istruzioni lette. Questi blocchi sono connessi per mezzo di un bus interno alle interfacce dati e indirizzi ma anche all’unità aritmetica ALU e ai registri.

Figura XX – Registri della CPU [datasheet p.X]

Lo Z80 mette a disposizione 6 registri nominati B, C, D, E, H, L. Questi sono registri a scopo generico, general purpose, che possono essere usati singolarmente a 8 bit o accoppiati a 16 bit come BC, DE, HL. I registri a 16 bit si possono usare per l’indirizzamento.  
Assieme a questi registri general purpose c’è la coppia formata dall’accumulatore A e dal registro F che contiene i contrassegni derivanti dalle operazioni dell’ALU, detti flag. Assieme formano la coppia AF che non può essere usata per indirizzare. Nel registro F sono contenuti i flag:  
S sign flag, che segnala il segno del risultato;  
Z zero flag, che segnala se il risultato è 0;   
P/V parity-overflow flag, che segnala se nell’ultima operazione aritmetica è avvenuto un overflow o se il risultato dell’ultima operazione logica è pari;  
H half-carry flag, che segnala se nell’ultima operazione si è verificato un carry tra il primo o il secondo nibble, cioè tra i quattro bit meno significativi e i successivi;  
N subtract flag, che segnala se nell’ultima operazione è stata effettuata una sottrazione;  
C carry flag, che segnala se è stato prodotto un riporto dal bit più significativo.  
Per ognuno di questi otto registri c’è una copia detti registri alternativi od ombra. I registri possono essere scambiati con quelli principali per mezzo di una coppia di istruzioni di scambio, Exchange. La presenza di questo set alternativo permette di risolvere velocemente un interrupt senza dover salvare il contesto su una pila oppure di svolgere delle operazioni con tecniche di background-foreground.

In aggiunta a questi registri ci sono altri registri con scopi precisi.  
La coppia di registri d’indicizzazione indipendenti IX e IY, entrambi a 16 bit, sono usabili come indirizzi specialmente come base per puntare un’area di memoria attraverso delle istruzioni di indicizzazione.  
Il registro SP, stack pointer, a 16 bit, punta alla prima posizione libera in testa ad una pila LIFO (Last-In First-Out). La pila è gestita attraverso una coppia di istruzioni PUSH e POP. La gestione automatica è utile per il salvataggio del contesto in caso di interrupt e il passaggio di parametri alle subroutine richiamate. Lo stack cresce verso gli indirizzi minori per cui un’operazione di PUSH decrementa SP.  
Il program counter PC è contenuto in un registro a 16 bit che può essere scritto e modificato con delle operazioni di salto. Il PC viene incrementato alla fine di ogni operazione di lettura o scrittura dalla memoria e sovrascritto in caso di salto.  
Il registro R, a 7 bit, contiene l’indirizzo di refresh che viene messo sul bus. Viene automaticamente incrementato alla fine dell’operazione e può essere gestito con un registro a 8 bit in relazione con l’accumulatore.  
Si può gestire allo stesso modo di R, il registro I detto Interrupt Page Address. Il registro contiene la parte alta di un indirizzo che punta ad una voce di una tabella contenente informazioni per gestire una richiesta di interrupt collegata. Però questo avviene solo in una delle tre modalità di gestione degli interrupt.

Ai precedenti registri si aggiungono due coppie di flip-flops.  
La prima coppia detta Interrupt Status FFs fa da maschera per gli interrupt normali. Il primo bit se abilitato permette la ricezione degli interrupt di nINT da parte della CPU. Il secondo bit serve come salvataggio del primo quando vengono servite le richieste da parte di nNMI.  
La seconda coppia detta Interrupt Mode FFs definisce la tipologia di servizio degli interrupt usati.

Gestione degli eventi di interrupt

Come già detto in precedenza, lo Z80 distingue tra interrupt mascherabili, INT, e non, NMI. Entrambi gli interrupt vengono serviti solo nel caso in cui non sia stato attivato nBUSREQ in precedenza e alla fine delle istruzioni.

Figura XXXX – Diagramma di flusso della gestione degli interrupt[THE Z80 FAMILY PROGRAM

INTERRUPT STRUCTURE p.4]

Gli interrupt non mascherabili, NMI, sono stati pensati per servire solo interrupt di alta priorità e che richiedono un servizio immediato. Un esempio è un problema all’alimentazione che richiede un immediato salvataggio del contesto da parte della CPU. Per questo non sono mascherabili via software. Essendo considerati come eventi unici hanno una sola strategia di servizio.  
Quando viene accettato l’interrupt, viene eseguita in automatico l’istruzione RST 66H. L’istruzione RST hh, abbrev. di Restart, ha un solo parametro (hh) che appartiene ad un insieme di otto indirizzi da 00H a 38H distanti ognuno 8 byte. L’esecuzione è simile ad un’istruzione di CALL e consiste nel salvare il PC nello stack e caricare l’indirizzo puntato dal parametro nel PC.  
Nel caso di un NMI, come si vede nella Figura X^5, la CPU esegue un opcode fetch ignorando l’istruzione letta. Poi salva PC nello stack, disabilita gli INT resettando IFF1 e salvando nell’altro, IFF2, lo stato del primo per ripristinarlo alla fine della routine. Infine salta alla locazione 66H a cui si deve trovare la routine di servizio.  
Per uscire da un NMI si usa l’istruzione apposita RETN, abbrev. per Return NMI. L’istruzione ricarica in PC il valore salvato nello stack, esattamente al contrario di RST hh. In aggiunta ad una normale istruzione di ritorno RET, ricarica il valore di IFF2 in IFF1 così da ripristinare l’abilitazione degli INT allo stato prima dell’interrupt.

Figura XXXXX – Sequenza di servizio di un interrupt non mascherabile [THE Z80 FAMILY PROGRAM

INTERRUPT STRUCTURE p.5]

Gli interrupt mascherabili, INT, sono abilitati o meno dallo stato di IFF1. Lo stato dei bit di IFF è selezionato da due istruzioni DI, abbrev. di Disable Interrupt, che resetta i bit disabilitando gli interrupt e EI, abbrev. Enable Interrupt, che setta i bit abilitando gli interrupt. In ogni caso gli INT rimangono disabilitati fino alla fine dell’istruzione successiva. Questo per permettere il corretto funzionamento delle istruzioni della famiglia RET, che solitamente seguono le precedenti alla fine delle routine di servizio degli interrupt.  
La possibile disabilitazione degli interrupt permette di garantire che alcune porzioni di codice vengano eseguite nel tempo corretto senza interruzioni. Inoltre, a differenza degli NMI, gli interrupt non vengono disabilitati automaticamente all’inizio della routine di servizio per cui si può implementare una logica di interrupt nidificato.[Buso p.178]  
Lo Z80 per mantenere la compatibilità con i sistemi bassati sull’Intel 8080, ma anche per avere una modalità preferenziale per le sue periferiche, permette tre modi diversi di gestire un interrupt. In ogni caso però l’asserzione avviene allo stesso modo: avviene una lettura di un dato attivando solo nIORQ e nM1 così da segnalare alla periferica interessata che la CPU sta prendendo in carico la richiesta. In aggiunta, la scelta della modalità avviene via software per mezzo di tre istruzioni dedicate: IM0, IM1 e IM2. Le quali settano il valore dei bit IMF.

Figura XXXXXX – Sequenze di servizio di un interrupt mascherabile [THE Z80 FAMILY PROGRAM

INTERRUPT STRUCTURE p.6]

La prima modalità detta Mode 0, selezionata con IM0, è completamente compatibile con le periferiche dell’Intel 8080. Durante il ciclo di interrupt acknowledge, la periferica mette sul bus D un’istruzione che viene eseguita dalla CPU. Nel caso di istruzioni che richiedono più byte, solo la prima operazione di lettura è di interrupt acknowledge mentre le altre sono semplici letture dalla memoria. Per cui la soluzione migliore è che la periferica fornisca un’istruzione RST hh verso la propria routine. Ogni routine ha di base 8 byte in cui può avvenire la disabilitazione degli altri interrupt e il salvataggio del contesto prima di saltare all’effettiva routine. Nell’ipotesi vantaggiosa che le periferiche sfruttino l’istruzione RST hh, si possono servire 8 diversi interrupt.  
La modalità Mode 1, selezionata con IM1, invece è simile a come vengono servirti gli NMI. Dopo il ciclo di interrupt acknowledge viene eseguito RST 38H, servendo quindi un unico tipo di interrupt.  
La modalità Mode 2, selezionata con IM2, è quella tipica delle periferiche dello Z80 e attua un interrupt vettorizzato.[Buso p.173] Durante il ciclo di interrupt acknowledge, la periferica carica sul bus l’indice di una entry in una tabella di cui verranno letti solo i sette bit più significativi. La CPU nel ciclo successivo salva nello stack il valore di PC. Poi legge un indirizzo contenuto in una tabella puntandolo per mezzo della giustapposizione tra il registro I e il valore fornito dalla periferica. Dopo aver letto il nuovo indirizzo, che punta alla routine di servizio adatta, effettua l’opcode fetch e ne inizia l’esecuzione. Con questa modalità si possono servire 128 interrupt diversi con una latenza minima di cinque cicli macchina.

Figura X^6 – Sequenza di servizio degli interrupt vettorizzati [THE Z80 FAMILY PROGRAM

INTERRUPT STRUCTURE p.9]

Il salvataggio del contesto non viene fatto automaticamente e può essere fatto attraverso l’uso dei registri ombra oppure sullo stack. Quest’ultima opzione è l’unica in caso di interrupt nidificati.  
Per ritornare dalle routine di interrupt si possono usare due istruzioni di ritorno che svolgono la stessa funzione dal punto di vista della CPU: RET e RETI. Entrambe ricaricano in PC il valore salvato nello stack. La seconda è pensata appositamente per le periferiche della famiglia Z80 poiché riconoscono l’istruzione diversa e si comportano di conseguenza.

Figura X^7 – Esempio di connessione daisy chain ##DA MODIFICARE##

Per permette di collegare più INT con priorità differenti si usa una daisy chain. La daisy chain è una connessione in serie delle periferiche sulla stessa linea di controllo. Ogni nodo della daisy chain condivide una connessione al data bus, un’uscita per la richiesta di interrupt, un’ingresso e un’uscita di un segnale di interrupt acknowledge. Tutte le uscite di richiesta interrupt sono collegate assieme per mezzo di una wired-AND. Mentre la linea di acknowledge esce dallo Z80 ed entra nel primo blocco, da questo esce un altro acknowledge e così via. La priorità tra gli interrupt viene dettata dalla vicinanza alla CPU: il più vicino ha priorità maggiore.  
La CPU quando asserisce una richiesta, invia il segnale di interrupt acknowledge, che nel caso dello Z80 avviene con nIORQ e nM1 attivi contemporaneamente. Dal primo nodo, viene ricevuto l’acknowledge e se la periferica effettivamente ha generato un’interrupt, mette sul bus il proprio indirizzo della tabella e non trasmette l’acknowledge al nodo successivo. Se invece non l’ha generato, trasmette al blocco successivo il segnale. Per servire altri interrupt, che continuano a tenere bassa la linea nINT, lo Z80 continua a svolgere i cicli di interrupt acknowledge.

Normalmente per ogni periferica bisogna creare una piccola interfaccia che svolga questa funzione. Nel caso delle periferiche della famiglia Z80, in particolare PIO, SIO e CTC, queste mettono a disposizione due linee: IEI, abbrev. di Interrupt Enable Input; IEO, abbrev. di Interrupt Enable Output. Questi pin sono collegati in modo che l’IEO della periferica precedente sia collegato con l’IEI della successiva. La prima periferica è collegata in modo da avere IEI sempre abilitato.  
Ogni periferica riconosce autonomamente la condizione di interrupt aclknowledge perché sono collegate al gruppo System Control. Se avviene un interrupt, la periferica interessata disabilita IEO altrimenti quest’ultimo ha lo stesso valore di IEI. Se IEI è disabilitato, nel caso di interrupt richiesto, la periferica non risponde all’interrupt acknowledge. Nel momento in cui una periferica, con IEI abilitato, risponde all’interrupt acknowledge, mette sul bus dati l’indirizzo della entry. Alla fine della routine, che si chiuderà con l’istruzione adatta RETI, la periferica riconosce l’istruzione e abilita IEO permettendo alla successiva periferica di servire il proprio interrupt.

Instruction set

Lo Z80 mette a disposizione 158 diverse istruzioni in cui l’opcode è per lo più contenuto in un solo byte. Il set di istruzioni contiene come sottoinsieme il set dell’Intel 8080. La lunghezza effettiva delle istruzioni dipende dal metodo di indirizzamento dei dati. Per cui si può passare da istruzioni di un singolo byte come LD A, B in cui l’indirizzamento dei registri è diretto sino a LD (IX+d), n che per contenere l’offset d per IX e il valore n occupa 4 byte.  
Le istruzioni hanno la forma generale:

<menmonico> {<operando\_1>{, <operando\_2>}}

In base all’istruzione, possono essere presenti entrambi gli operandi, solo il primo oppure nessuno. Nel caso di istruzioni con due operandi, si ha sempre che il secondo va a modificare il primo. Ad esempio in ADD HL, BC si ha che in HL viene caricato il valore della somma tra HL stesso e BC a 16 bit.

<operando\_1>, <operando\_2>: <r> | (HL) | (IX+ <d> ) | (IY+ <d> ) | (C) | (A) | <n> | <nn> | (<nn>) | (BC) | (DE) | <b> | <e> | <cc> | <qq> | <ss> | <pp> | <rr> | <s> | <m>

Tabella – Sommario degli operandi ##DA FARE DA QUELLA A PAGINA 53 PDF um0080##

Gli operandi invece possono essere direttamente valori, puntatori ad uno specifico registro o bit, oppure puntatori a locazioni di memoria, riconoscibili perché contenuti in parentesi tonde.  
Le istruzioni si dividono in 11 gruppi: caricamenti a 8 bit e a6 bit; exchange, trasferimenti a blocchi e ricerca; operazioni a 8 bit e 16 bit; istruzioni di controllo della CPU; rotazioni e scorrimenti; operazioni sui bit; salti; chiamate, ritorni e restart; operazioni sugli I/O.  
Di nota sono le operazioni di trasferimento di blocchi e ricerca. Queste sono operazioni che continuano ad essere eseguite fintantoché non si verifica una condizione. La condizione comune a tutte è quella che un contatore arrivi a 0. Nel caso in cui dopo ogni singola esecuzione non si verifichi la condizione, viene ricaricato nel PC la posizione dell’istruzione. Per cui la CPU continua a fare il fetch della stessa istruzione per eseguire il ciclo.  
Inoltre lo Z80 presenta una specifica integrazione per svolgere sia somme che sottrazioni con operandi in BCD. Per mezzo di un’istruzione di decimal adjust, DAA, che in base allo stato dei flag H ed N ed al valore dell’accumulatore, ricostruisce il risultato BCD dell’operazione.  
Principalmente segue questa logica:  
se N è 1 tutte le operazioni successive sono di sottrazione invece che di somma;  
se la parte alta del risultato nell’accumulatore è maggiore di 9 o C è settato, viene sommato 60H;  
se la parte bassa del risultato nell’accumulatore è maggiore di 9 o H è settato, viene sommato 6H;  
nei casi previsti si eseguono entrambe le somme.

Lo Z80 permette ben 10 metodi diversi di indirizzare gli operandi.  
Permette l’indirizzamento immediato sia di valori a 8 bit o 16 bit fornendoli nei byte successivi all’opcode.  
Allo stesso modo si possono indirizzare per mezzo di un campo dell’istruzione un registro, una coppia di registri o un singolo bit in un byte. Alcune istruzioni come quelle aritmetico-logiche a 8 bit, indirizzano implicitamente l’accumulatore.  
Come indirizzamento indiretto lo Z80 permette di usare preferibilmente il registro HL per mantenere l’indirizzo oppure SP per quanto riguarda lo stack.  
Nel caso particolare dei registri IX e IY, di può attuare un indirizzamento indicizzato. Si può per esempio puntare alla locazione (IX+d) con d un indice da -127 a +128 da sommare a IX solo per creare il puntatore.