Figura X – Die dello Z80 con le parti evidenziate [sito baltazar]

Per quanto sia chiaramente ben documentata l’architettura dello Z80, non ci sono informazioni ufficiali sulla sua organizzazione.

Tutte le informazioni che sono apparse negli anni successivi derivano da un processo di reverse engineering del die. Cioè si è analizzato al microscopio il chip e dalle tracce ottiche lasciate dai vari passaggi produttivi si è risaliti alla sua organizzazione a livello dei singoli transistor. Da lì si sono fatti i vari passaggi di astrazione sino a definire le unità operative e le loro connessioni. Di notevole importanza è stato il lavoro di Ken Shirriff, che nel suo blog ha scritto vari post al riguardo [http://www.righto.com/search/label/Z-80].

Figura XX – Schema blocchi ALU [http://www.righto.com/search/label/Z-80]

Come prima scoperta si è visto che l’ALU dello Z80 non è a 8 bit, come lo sono le parole, ma a 4 bit.  
Di conseguenza la ALU svolge le operazioni un nibble alla volta. Per prima cosa la ALU carica gli operandi provenienti dal bus in due latch, uno per ogni operando. Ogni latch è diviso in nibble più significativo e meno significativo. Le uscite di ogni latch sono collegate a due multiplexer divisi per i due operandi, in questo modo l’ALU può scegliere con quale parte operare. Dopodiché la ALU prima compie l’operazione sui nibble inferiori e salva il risultato su un latch. Poi somma le parti superiori e mette sul bus sia il latch della parte inferiore che la parte appena calcolata.  
I flag P di P/V, N, S e Z vengono generati direttamente dagli operandi. Il flag H viene generato al primo passaggio. Mentre il flag V di P/V e C vengono aggiornati solo nel secondo passaggio.  
Per le operazioni sui bit come SET, RST e BIT, la ALU compie delle operazioni di AND e OR con delle maschere generate in base al bit selezionato. La decodifica della selezione del bit avviene direttamente nell’ALU leggendo il campo corrispondente dall’istruzione, invece di ricevere dei segnali di controllo dal decoder.  
A differenza di altri processori come l’Intel 8085, successore dell’Intel 8080, lo Z80 non ha un’ALU dedicata per svolgere le operazioni di rotazione e scorrimento. Queste, essendo tutte ad un bit, sono svolte nel caricamento degli operandi, per cui l’ALU non deve svolgere nessuna operazione successiva. Per lo scorrimento come cifre BCD, che è svolto dalle istruzioni RRD e RLD, l’ALU compie la stessa operazione scambiando nei latch i nibble con quelli di un indirizzo in memoria.[ http://www.righto.com/2013/09/the-z-80-has-4-bit-alu-heres-how-it.html]

Figura X^3 – Circuito completo dell’inc/dec [http://www.righto.com/2013/11/the-z-80s-16-bit-incrementdecrement.html]

Figura X^4 – Inc/dec a 2 bit base dell’inc/dec [http://www.righto.com/2013/11/the-z-80s-16-bit-incrementdecrement.html]

Vicino al latch degli indirizzi è presente un incrementer/decrementer dedicato, abbrev. inc/dec, a 16 bit per svolgere tutte le operazioni del gruppo INC e DEC.  
In questo inc/dec viene caricato il valore del bus indirizzi, dopo il latch dedicato, e fornisce il nuovo valore sul bus interno. La necessità di avere un inc/dec dedicato deriva dal poter svolgere un tipo rudimentale di pipeline. Questo consiste nella sovrapposizione delle fasi di fetch e decode. Queste si sovrappongono nell’ultimo T-cycle in cui si sta effettuando l’operazione di refresh e la decodifica o intera esecuzione dell’istruzione appena ottenuta. Per cui si sfrutta l’inc/dec per caricare in PC il nuovo valore subito dopo il fetch e per caricare il valore di R durante il primo T-cycle dell’esecuzione o decodifica.  
La cella base dell’inc/dec è a due bit con carry in ingresso e sfrutta la tecnica del carry-skip per ridurre il ritardo di propagazione.  
Poi questi inc/dec a 2 bit sono collegati riportando il serie il carry in gruppi di due o tre. Per ognuno di questi gruppi c’è un circuito di carry-look ahead, CLA. Quest’ultimo sfrutta il fatto che se c’è almeno uno 0 non viene generato un carry in caso di incremento mentre se c’è almeno un 1 non viene generato un borrow in caso di decremento. Nel primo circuito c’è un segnale che se abilitato inibisce il carry.  
Quest’ultima caratteristica è utile per l’incremento di R. R viene passato all’inc/dec come coppia I-R e quando viene incrementato non si deve intaccare I. In questo modo settando quel particolare bit non avviene il carry oltre il bit 6 per cui non si modifica il registro I e nemmeno i bit più alti di R.  
All’interno dello Z80 non sono presenti due bus a 16 bit per i dati così da poter svolgere le operazioni di load sui registri in un’unica soluzione. L’unico presente è quello per gli indirizzi. Per cui un’altra caratteristica dell’inc/dec è la possibilità di non fare nessuna operazione. Questo, collegato al fatto che l’inc/dec per sua configurazione fa da retroazione del latch dell’indirizzo sul bus indirizzi interno, permette di salvare il valore di un registro a 16 bit e poi trasferirlo su un altro.  
L’ultimo elemento in più collegato è un riconoscitore del valore 0001H. Questo è sfruttato nelle operazioni di copia e ricerca di blocchi che ciclano sintantoché il contatore contenuto in BC arriva 0000H. In realtà lo Z80 controlla che il registro sia a 0001H per poi svolgere l’operazione come se fosse quella finale invece di controllare solo dopo il decremento.  
A differenza dell’ALU, l’inc/dec non modifica i flag. [http://www.righto.com/2013/11/the-z-80s-16-bit-incrementdecrement.html]

Figura X^5 – Data bus interni. In rosso, verde e arancio le tre sezioni del bus a 8 bit. In fucsia, il bus indirizzi a 16 bit. [http://www.righto.com/2014/10/how-z80s-registers-are-implemented-down.html]

Il data bus interno a 8 bit è diviso in tre parti. Una prima che dall’esterno va verso la logica di decodifica e l’ALU poiché da sola decodifica le istruzioni sui bit. Una seconda, parte dalla prima divisa per mezzo di un latch e va verso una pagina dei registri. Dalla seconda ne parte una terza, divisa anch’essa con un latch verso l’altra pagina dei registri e l’ALU. La presenza dei tre bus permette di svolgere operazioni in parallelo all’interno della CPU. [http://www.righto.com/2014/09/why-z-80s-data-pins-are-scrambled.html]

Figura X^6 – Struttura dei registri dello Z80[http://www.righto.com/search/label/Z-80]

La struttura interna dei registri non è esattamente come quella mostrata nella Figura X##Quella del PDF## poiché non vi è una vera distinzione tra registri principali e ombra. Tutti i registri A, F, B, C, D, E, H ed L sono ravvicinati e collegati allo stesso bus a 16 bit come alla stessa coppia di bus a 8 bit. I bus a 8 bit sono divisi in maniera tale che uno punti ai registri che rappresentano la parte alta quindi A, B, D ed H mentre l’altra ai rimanenti F, C, E ed L. I due bus sono separati da degli interruttori che possono unirli o meno così da permettere operazioni in simultanea sui due gruppi di registri.  
Poi per ogni registro ce n’è uno identico indistinguibile dal primo. Per cui all’arrivo dell’istruzione EXX, che scambia tra registri principali ed ombra per BC, DE ed HL, viene semplicemente modificato un flag che seleziona un gruppo o l’altro senza distinzione. Allo stesso modo funziona l’istruzione EX AF, AF’, che scambia le coppie AF, e l’istruzione EX DE, HL, che con un altro flag instrada le istruzioni verso DE su HL e viceversa senza distinguere tra gli originali registri o meno.  
Sempre collegati a questo gruppo ci sono i registri IX, IY ed SP. Vi sono anche due registri temporanei a 8 bit non visibili al programmatore chiamati W ed Z. Questi sono usati come appoggio nelle istruzioni di lettura di indirizzi, dati o per lo scambio. Vengono usati per esempio nell’istruzione JP, abbrev. di Jump, che esegue un salto verso un indirizzo a 16 bit che segue l’opcode. In quel caso l’indirizzo a 16 bit viene caricato durante la lettura su WZ, prima di caricarlo in PC.  
A differenza dei precedenti, i registri PC e I-R sono separati dagli altri e isolabili dal bus a 16 bit per mezzo di interruttori per permettere le operazioni di incremento descritte prima.  
Infine il registro F presenta una copia nell’ALU. Questa viene caricata all’inizio delle operazioni per permettere il corretto utilizzo durante l’elaborazione dell’unità. [http://www.righto.com/2014/10/how-z80s-registers-are-implemented-down.html]

Per quanto riguarda le istruzioni che gestiscono l’I/O, nel datasheet è detto che possano indirizzare solo per mezzo degli 8 bit meno significativi. Invece, più avanti nello stesso datasheet, è affermato che l’indirizzamento diretto usa A come parte alta mentre l’indirizzamento indiretto per mezzo di C usa la coppia BC.

Lo Z80 presenta dei comportamenti non documentati nei datasheet ma dipendenti da residui nella progettazione del decoder delle istruzioni. Quest’analisi è stata ben documentata [z80-documented.pdf] e spazia dalla presenza di istruzioni particolari sino ad un comportamento non descritto per la gestione degli interrupt.  
Di nota sono l’esecuzione dell’istruzione DAA, la presenza dell’istruzione SLL, abbrev. di Shift logical left, controparte logica dello scorrimento aritmetico a sinistra SLA, ed infine la presenza di flip-flops sui pin nINT, nNMI e nBUSREQ.  
Per cui le richieste di interrupt possono avvenire in qualsiasi istante, la richiesta viene immagazzinata nei FFs e servita a tempo debito.

In conclusione, non si conosce completamente la vera organizzazione del decoder delle istruzioni. Dal die si riconosce una sezione di timing che conta i vari M-cycles e T-cycles. Il conteggio viene trasmesso a un PLA che fornisce i segnali di controllo. Di conseguenza si può dire solamente che è sicuro che lo Z80 sfrutti un controllo cablato e non microprogrammato.