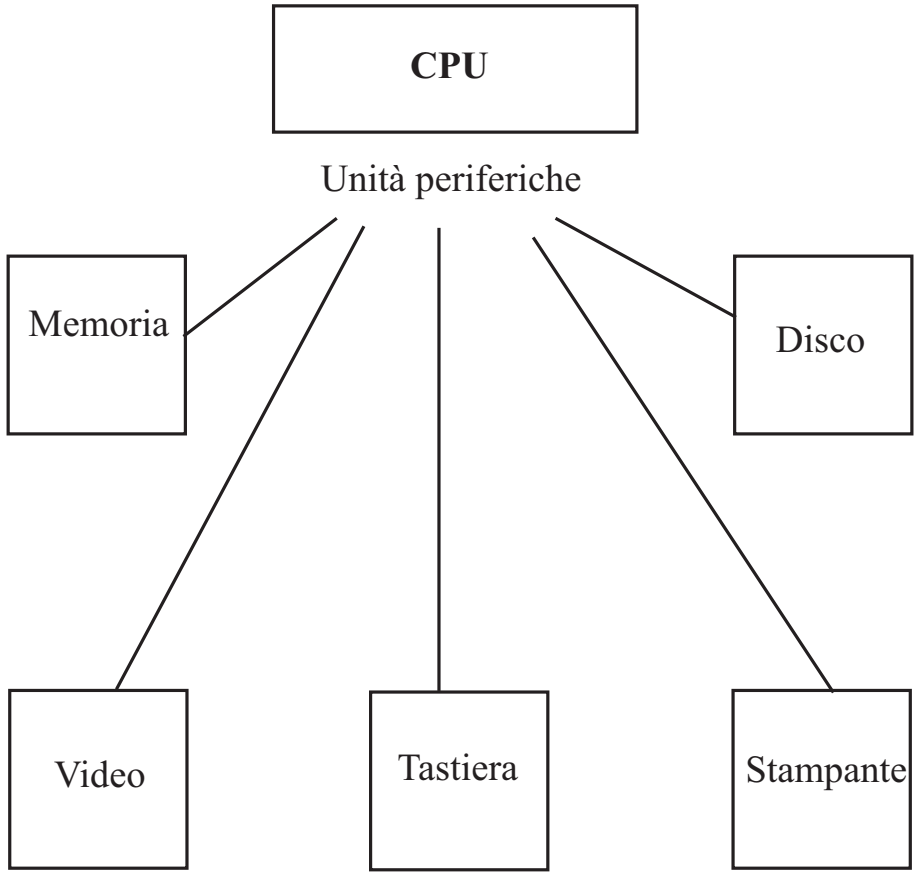
Il microprocessore Z80

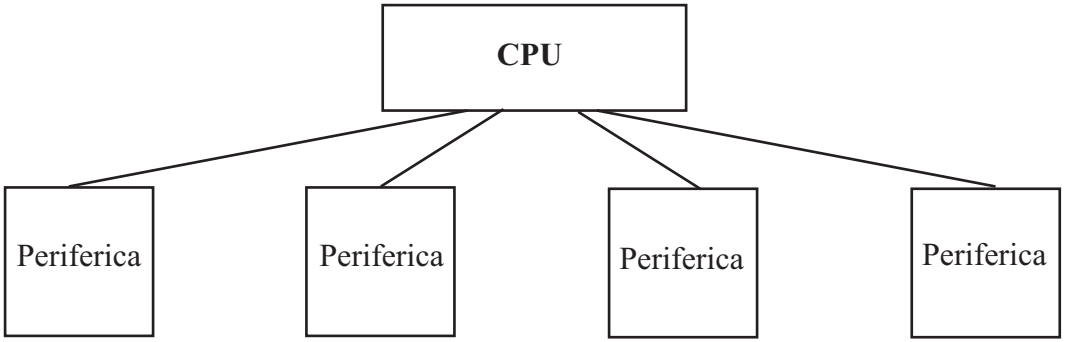
10 dicembre 2013

Architettura di un computer



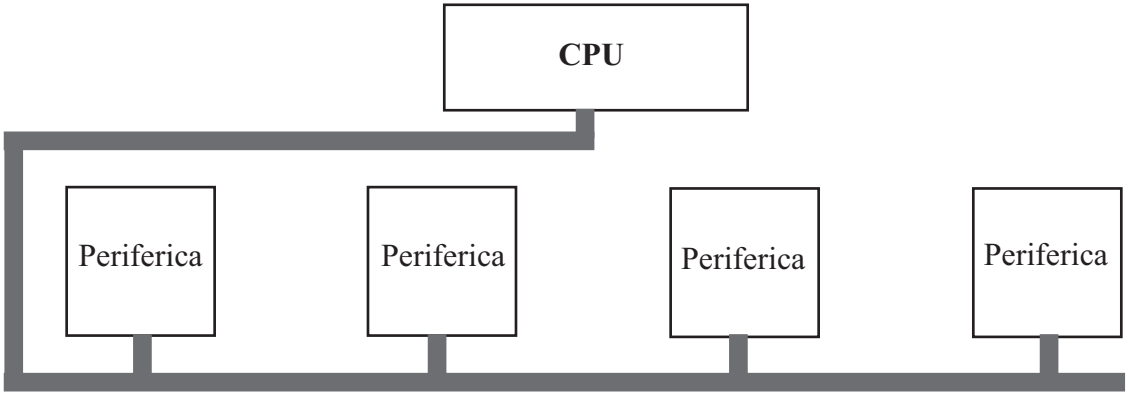
In passato una CPU era costituita da un enorme volume di circuiti elettronici; oggi, i progressi fatti nel campo dell'integrazione hanno reso possibile racchiudere tutte le funzioni in un unico chip di silicio, in cui sono racchiuse migliaia o decine di migliaia di porte logiche elementari. Questi integrati prendono il nome di microprocessori.

Connessione tra CPU e periferiche - I



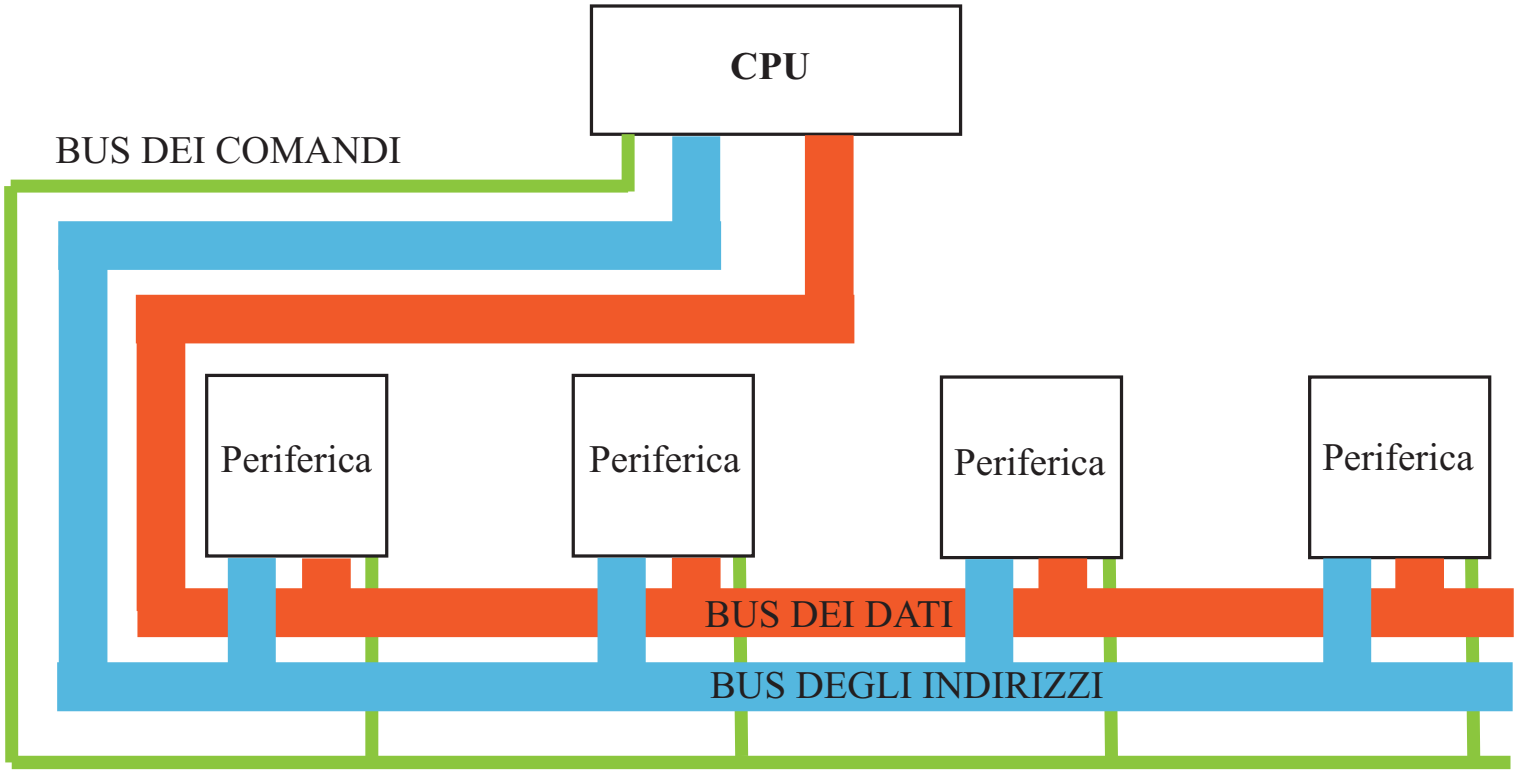
Ogni periferica ha un suo canale di comunicazione con la CPU

Connessione tra CPU e periferiche- II



Tutte le periferiche condividono il canale di comunicazione con la CPU (Bus)

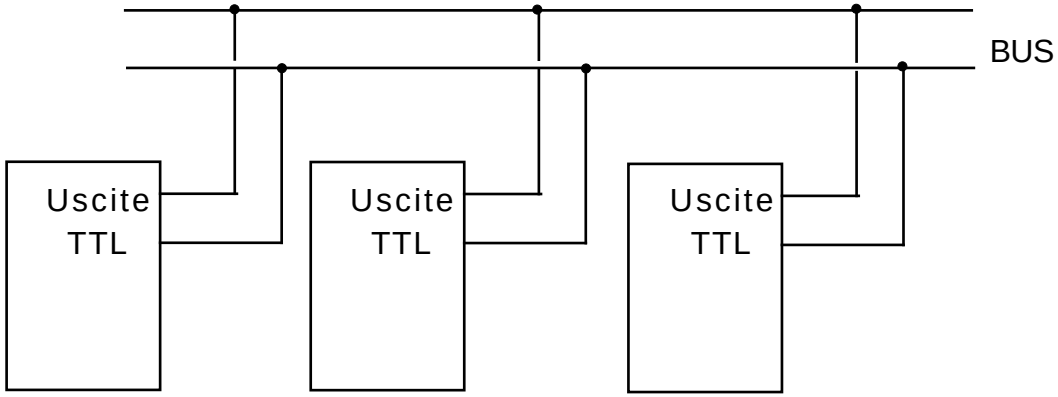
BUS



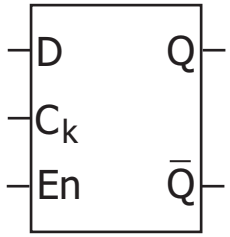
Ogni periferica ha un suo indirizzo univoco

Logica 3-state

Le porte logiche non possono essere connesse ad un bus in modo immediato, perché potrebbero sorgere dei conflitti.



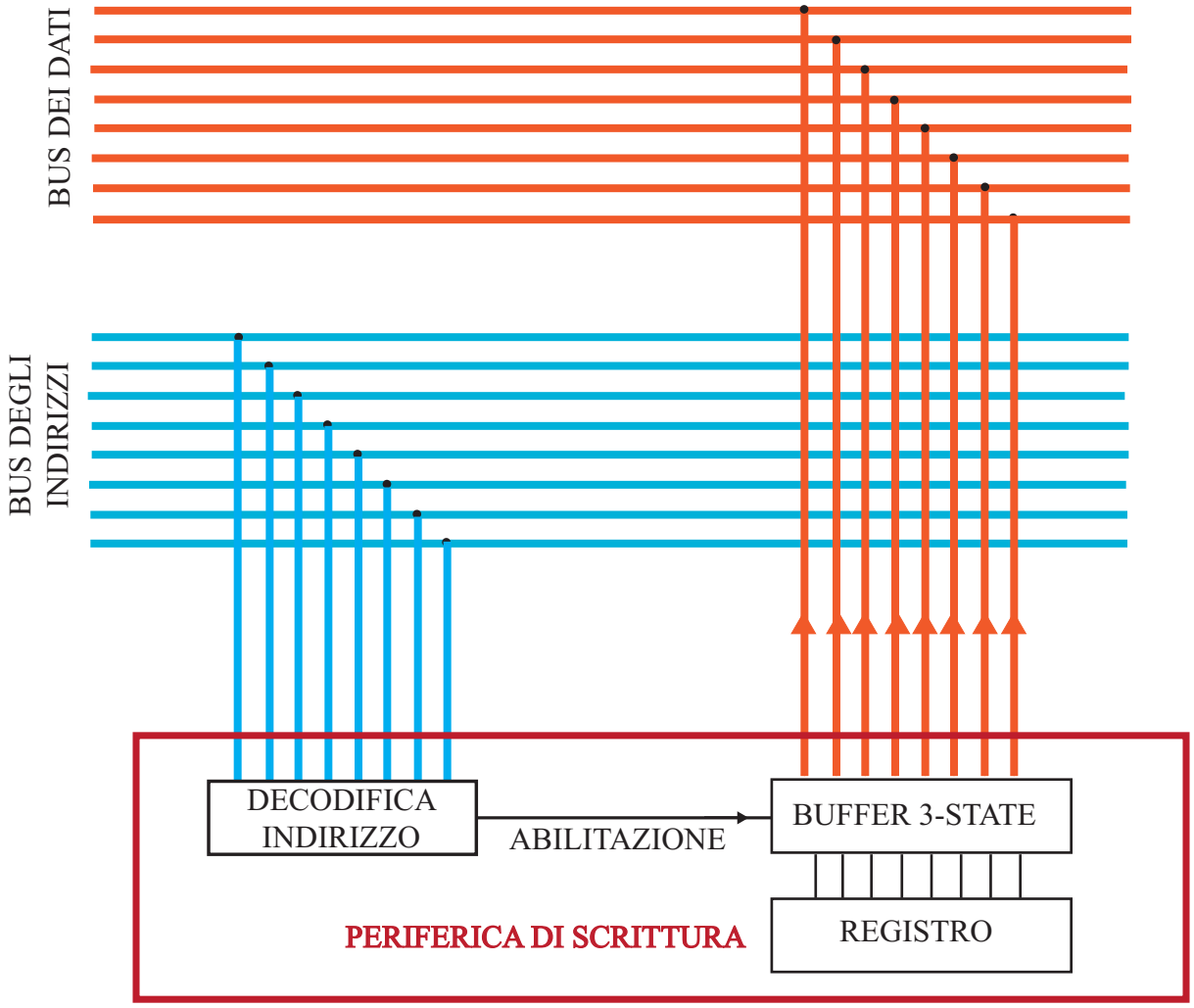
Si deve evitare che due o più periferiche scrivano simultaneamente sul bus
Una soluzione è data dalla logica "3-state":



Enable	D_n	Q_{n+1}
1	0	0
1	1	1
0	x	H

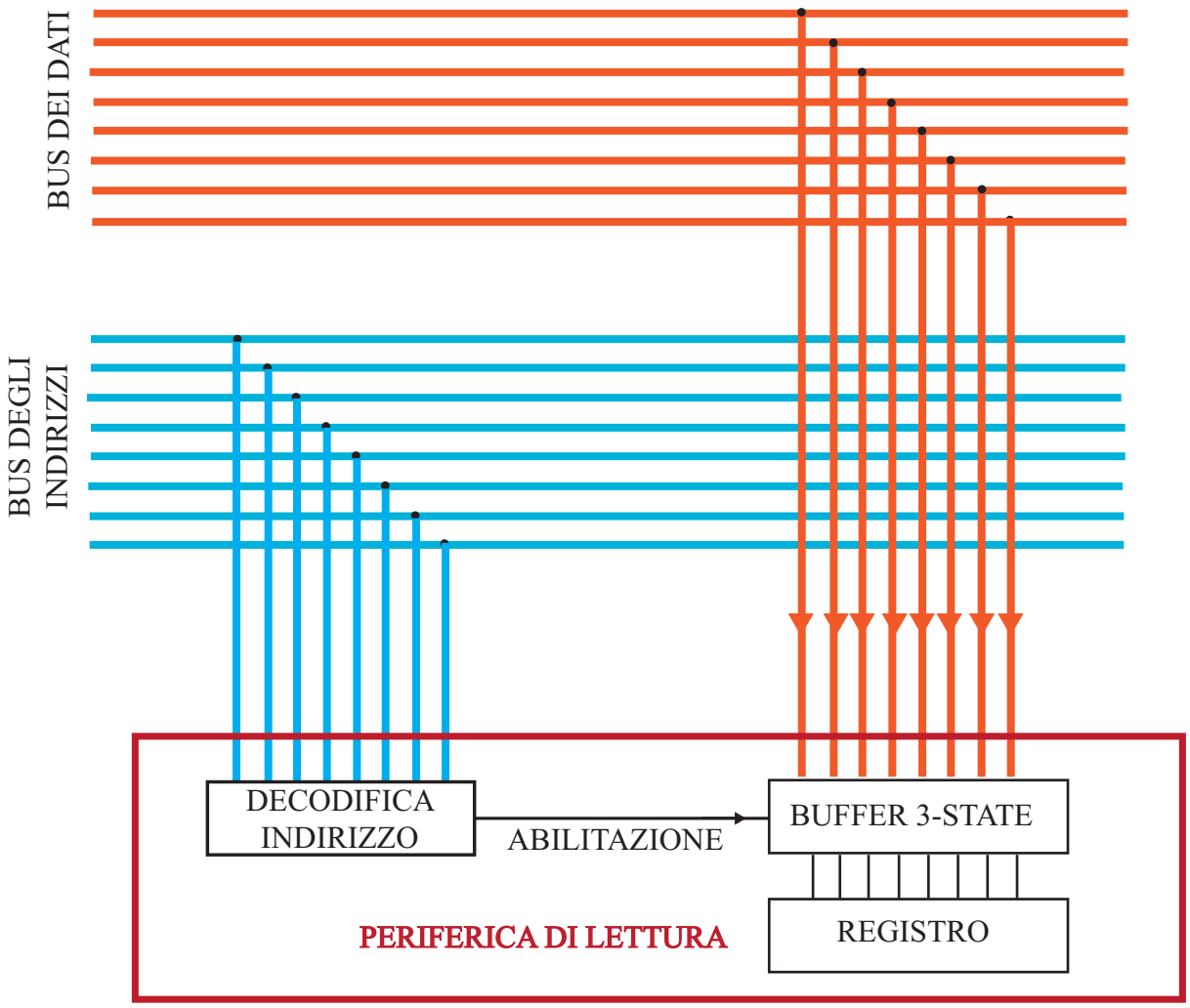
Quando $En = 0$ l'uscita e' in alta impedenza.
Con la logica "3-state" solo una periferica per volta è abilitata, le altre devono essere in alta impedenza: questo si ottiene col meccanismo dell'indirizzamento.

Periferica di scrittura



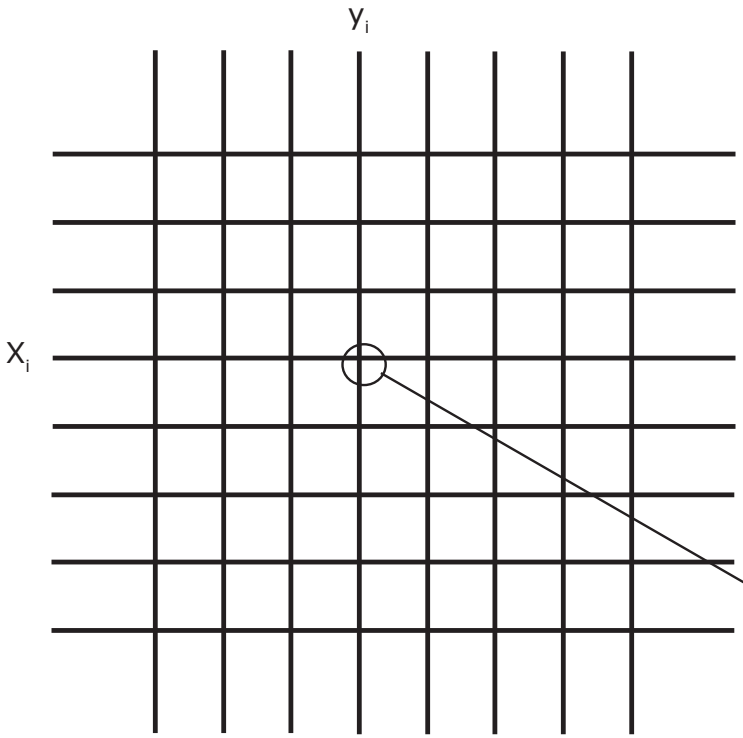
Quando l'indirizzo presente sul bus corrisponde a quello della periferica il buffer è abilitato e scrive sul bus dei dati il contenuto del registro.
Quando la periferica non è indirizzata le uscite del buffer sono in alta impedenza.

Periferica di lettura



Quando l'indirizzo presente sul bus corrisponde a quello della periferica
il buffer è abilitato e scrive sul registro della periferica il contenuto del bus dei dati.
Quando la periferica non è indirizzata le uscite del buffer sono in alta impedenza.

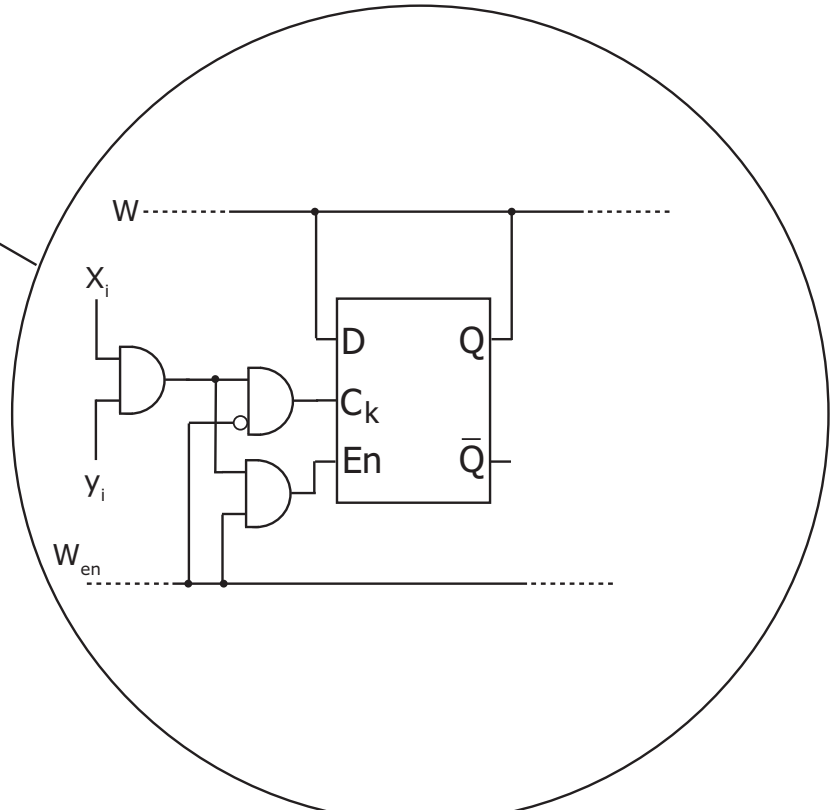
Indirizzamento della memoria



INDIRIZZAMENTO DI UNA CELLA DI MEMORIA

(SCHEMA CONCETTUALE)

In realta' un gruppo di 8 celle (byte) condivide lo stesso indirizzo.
Ci sono 8 linee W con cui si legge (o si scrive) un byte



Microprocessori

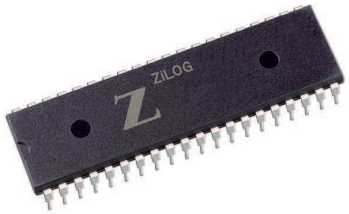
I computer (e i microprocessori) si contraddistinguono dalla dimensione del bus dei dati e da quella del bus degli indirizzi.

Costruttore	Sigla	Dati (bit)	Indirizzi (bit)
Intel	8080	8	16
Motorola	6800	8	16
Zilog	Z80	8	16
Intel	8086	16	16
Motorola	68000	16	16
Intel	80386	32	32
Motorola	68020	32	32

Tabella 1: *Alcuni microprocessori "storici"*

I computer (e i microprocessori) sono macchine "sincrone": la sequenza delle operazioni è scandita da un clock.

Il microprocessore Zilog Z80



Lo Z80 nacque nel luglio del 1976 per opera di Federico Faggin che, lasciata la Intel dopo aver lavorato sull'8080, aveva fondato la Zilog. Era progettato per offrire compatibilità binaria con l'Intel 8080 in modo che il codice 8080 (in particolare il sistema operativo CP/M) potesse essere eseguito sullo Z80 senza modifiche.

In breve lo Z80 conquistò il mercato dell'8080, e divenne la più popolare CPU a 8 bit di tutti i tempi (e, tenendo in considerazione la dimensione del mercato di allora, la CPU più popolare in generale). Versioni successive dello Z80 ne hanno aumentato la velocità dai 2,5 MHz iniziali fino a 20 MHz.

Viene prodotto in grandi volumi ancora oggi (2009), finora ne sono stati prodotti più di 2 miliardi di unità. Lo si produce ancora e in tali quantità soprattutto per usarlo come microcontrollore o per motivi di istruzione: la semplicità di programmazione, la semplice architettura a 8 bit e la grande quantità di informazioni lo rendono uno strumento perfetto per l'insegnamento dell' elettronica programmata.

Zilog concedeva in licenza il core dello Z80 senza royalty a tutte le aziende che volessero costruire il chip. Questo fece si che il prodotto della piccola azienda guadagnasse consensi nel mercato mondiale, in quanto aziende di gran lunga più grandi come Toshiba iniziarono a produrre lo Z80.

Numerazione esadecimale

La notazione binaria e' molto pesante, conviene usare la notazione esadecimale.

Num. decimale	Num. binaria	Num. esadecimale
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
...	...	...
255	11111111	FF
256	100000000	100
...	...	

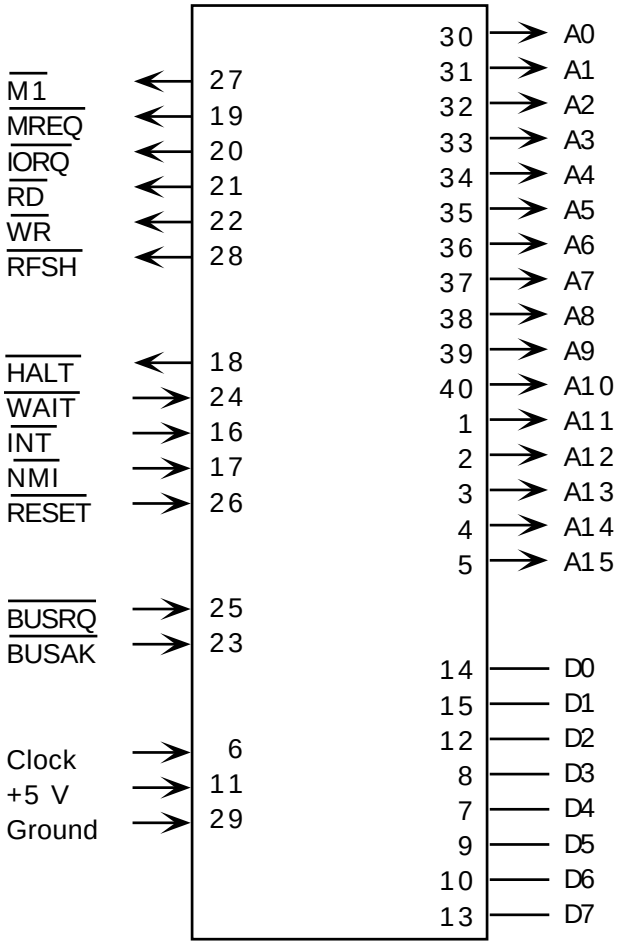
Tipi di memoria

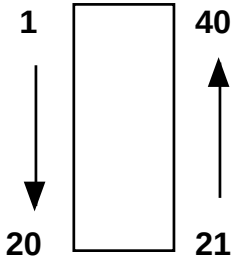
Memorie statiche (SRAM)
Flip-flop tipo D semplificato - 6 transistor per cella

Memorie dinamiche (DRAM)
Un transistor MOS e una capacita' MOS
(necessitano di rinfresco)

Memorie Flash
Una capacita' MOS

Struttura dello Z80 - I



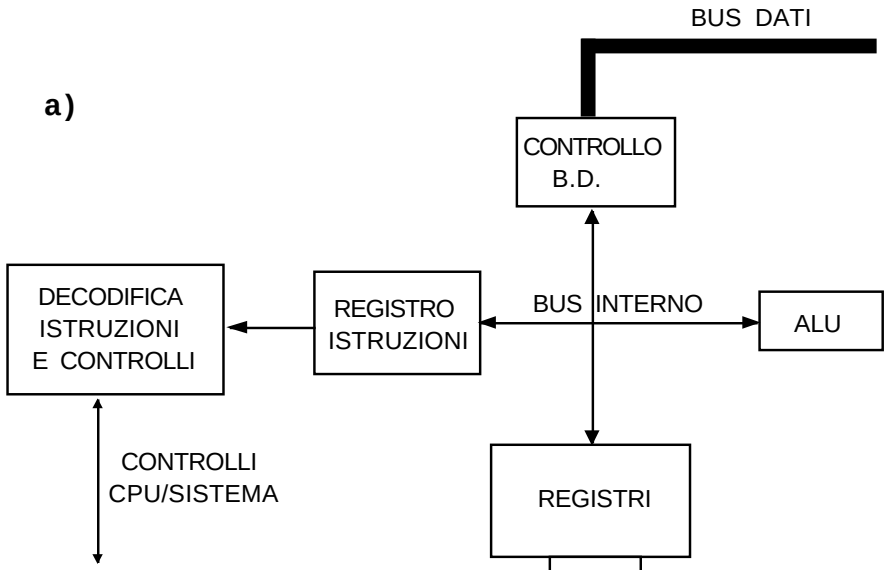


NUMERAZIONE
DEI PIEDINI

I segnali di controllo sono "attivi bassi": perchè?

Struttura dello Z80 - II

a)



b)

Registri principali		Registri secondari	
A	F	A'	F'
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'
	I	R	
Index Register IX			
Index Register IY			
Stack Pointer			
Program Counter			

Il registro F

Bit number	Simbolo	Significato
0	C	Carry
1	N	Add/Subtract
2	P/V	Parity/Overflow
3	X	Non usato
4	H	Half carry
5	X	Non usato
6	Z	Zero
7	S	Sign

Programmazione dello Z80 - I

Indirizzo	Istruzione	Mnemonico	Commento
0000	00	NOP	Non fa niente
0001	C3	JP 0000	Salta alla locazione 0000
0002	00		
0003	00		

Il microprocessore esegue un compito definito leggendo dalla memoria una serie di istruzioni, che costituiscono il programma.

Fornendo un comando di $\overline{RESET}$, lo Z80 comincia ad eseguire il programma partendo dalla locazione 0000.

Le istruzioni devono essere date in linguaggio macchina, cioe' in codice binario: e' quindi utile al programmatore fare ricorso ad una forma mnemonica per indicare ogni istruzione, e scrivere anche delle note.

Non tutte le istruzioni dello Z80 si esauriscono in un byte; ci sono invece istruzioni piu' complesse che richiedono 2,3 o anche 4 bytes. Di conseguenza l'istruzione deve essere memorizzata in piu' locazioni di memoria consecutive.

Programmazione dello Z80 - II

Indirizzo	Istruzione	Mnemonico	Commento
0000	06	LDB, 64	Carica nel registro B il numero
0001	64		esadecimale 64
0002	05	DEC B	Decrementa il registro B di 1
0003	C2	JPNZ, 0002	Salta alla locazione 002 se
0004	02		l'ultima operazione non ha dato
0005	00		risultato zero
0006	76	HALT	Si ferma

Questo programma esegue un loop 64 volte (esadecimale) e poi si ferma. In esso abbiamo usato istruzioni lunghe 1,2 e 3 bytes. Abbiamo caricato in B un numero e successivamente lo abbiamo decrementato a passi di 1, eseguendo ogni volta un test per verificare il contenuto del registro. Questo tipo di istruzioni usa proprio il bit 6 del registro F, che viene posto ad 1 quando l'ultima operazione fatta (in questo caso DEC B) fornisce un risultato zero. Si noti poi il modo con cui abbiamo dato l'indirizzo nell'istruzione JPNZ, 0002: prima il byte meno significativo (02) e poi quello piu' significativo (00). Questa convenzione e' del tutto generale e deve essere seguita in tutte le istruzioni che contengono un indirizzo.

Programmazione dello Z80 - III

Indirizzo	Label	Istruzione	Mnemonico	Commento
0000		06	LDB, 64	Carica nel reg. B
0001		64		il numero esadec. 64
0002	loop	05	DEC B	Decrementa il reg. B di 1
0003		C2	JPNZ, loop	Salta alla loc. 002 se
0004		02		l'ultima operazione non ha
0005		00		dato risultato zero
0006		76	HALT	Si ferma

Alla locazione 0002 (il bersaglio del salto) abbiamo dato il nome loop: questo rende piu' facile rileggere il programma e capirne il funzionamento.

Programmazione dello Z80 - IV

Indirizzo	Label	Istruzione	Mnemonico	Commento
0100		3A	LDA,(0200)	Carica in A
0101		00		il primo addendo
0102		02		
0103		2A	LDHL, 0201	Carica in HL
0104		01		l'indirizzo del
0105		02		secondo addendo
0106		86	ADDA,(HL)	Somma
0107		32	LD (0202),A	Scrive in memoria
0108		02		il risultato
0109		02		
010A		76	HALT	

Il programma e' memorizzato alla locazione 0100; gli operandi nelle locazioni di memoria 0200 e 0201, mentre il risultato viene salvato nella locazione 0202.

Qui abbiamo usato varie tecniche di indirizzamento

Come facciamo a mettere in esecuzione questo programma?

Indirizzo	Label	Istruzione	Mnemonico	Commento
0000		C3	JP, 0100	Salta a 0100
0001		00		
0002		01		

Classificazione delle istruzioni - I

Caricamento a 8 o 16 bit
: Queste istruzioni muovono i dati tra i vari registri della CPU, ovvero tra i registri e la memoria. Questo gruppo comprende pure istruzioni di caricamento immediato del dato specificato nell'istruzione stessa in uno dei registri o in una qualsiasi posizione di memoria. Le istruzioni di scambio (Exchange) consentono poi di effettuare lo scambio tra il contenuto di due registri. In questo ambito rientrano anche istruzioni che, ad esempio, consentono di scambiare il contenuto dei registri principali con quello dei registri secondari.

Istruzioni aritmetiche e logiche
: Operano su dati posti nell'accumulatore e in un altro qualsiasi dei registri di uso generale, oppure su dati posti in accumulatore e una qualsiasi locazione di memoria. Sono anche possibili somme e sottrazioni a 16 bit.

Istruzioni di rotazione e scorrimento
: Permettono di far ruotare verso destra o verso sinistra il contenuto di qualunque registro o locazione di memoria, con o senza l'utilizzo del bit di Carry del registro F. Inoltre sono possibili rotazioni separate del gruppo meno significativo di 4 bit (nibble).

Classificazione delle istruzioni - II

Manipolazioni di singoli bit

E' possibile esaminare, porre a 1 (set), o porre a 0 (reset), singoli bit di ogni registro, o di qualunque locazione di memoria.

Istruzioni di salto, chiamata, o ritorno

E' possibili fare salti (Jump), ovvero chiamate di subroutines (Call) con relativi ritorni. I salti possono essere condizionati al valore di specifici bits del registro F, in particolare il bit Z (zero) o il bit C (carry).

Istruzioni di input/output

Esistono varie istruzioni per effettuare operazioni di I/O con dispositivi esterni, che vengono selezionati utilizzando gli 8 bit meno significativi del bus degli indirizzi. E' quindi possibile in linea di principio avere 256 dispositivi di I/O diversi. Torneremo piu' avanti su queste istruzioni.

Trasferimento e ricerca di blocchi di dati

E' possibile trasferire un blocco di dati di qualsiasi dimensioni tra due diverse posizioni di memoria. Inoltre e' possibile analizzare un blocco di memoria per ricercare una particolare configurazione di 8 bit.

Modi di indirizzamento - I

Esistono vari modi di indirizzamento della memoria, come abbiamo gia' imparato attraverso gli esempi visti:

Indirizzamento immediato:
Il dato (1 byte) viene fornito direttamente e caricato sul registro. Sono quindi istruzioni del tipo:
LDr, n: carica il byte n nel registro r
Il registro puo' essere A,B,C,D,E,H,L

Indirizzamento immediato esteso:
LDdd, nn: carica i due bytes nn nella coppia di registri dd
La coppia di registri puo' essere BC,DE,HL oppure il registro SP.
Ovviamente i due bytes vanno dati nel consueto ordine (LSB e MSB)

Indirizzamento esteso:
LDA, (nn): carica in A il contenuto della locazione di memoria nn
Questo schema è usato anche nelle istruzioni di salto

Indirizzamento implicito:
LDr, r': il contenuto del registro r' e' copiato in r
I registri utilizzabili sono A,B,C,D,E,H,L

Indirizzamento indiretto:
LDr,(dd): carica nel registro r il contenuto della locazione di memoria il cui indirizzo e' nella coppia di registri dd.
Normalmente la coppia usata e' HL.

Modi di indirizzamento - II

Indirizzamento indicizzato:

LDr,(IX+d): carica nel registro r il contenuto della locazione di memoria indirizzata da IX piu' un offset d;

d e' un byte che viene fornito direttamente, ed e' interpretato come un numero di 7 bit con segno. Quindi l'offset possibile e' ± 127.

Indirizzamento relativo:

Si applica solo nelle istruzioni di salto; consente un salto di ± 127 locazioni *relativamente* alla locazione corrente. E' un'istruzione del tipo:

JR e: salta dell'offset e (fornito direttamente)

ovvero

JRcc, e: salto condizionato (cc puo' essere Z,NZ,C,NC)

Modificato in pagina zero:

Esiste solo per l'istruzione RST (restart): il programma salta ad alcune locazioni predefinite della pagina zero (cioe' della parte iniziale della memoria). Poiche' e' un'istruzione a un solo byte, la sua esecuzione e' piu' veloce del normale salto, e puo' quindi essere utile quandi si deve velocemente reagire ad un interrupt esterno. L'istruzione quindi e' scritta come:

RSTgg, dove gg puo' essere $00, 08, 10, 18, 20, 28, 30, 38$.

Temporizzazione: ciclo M1

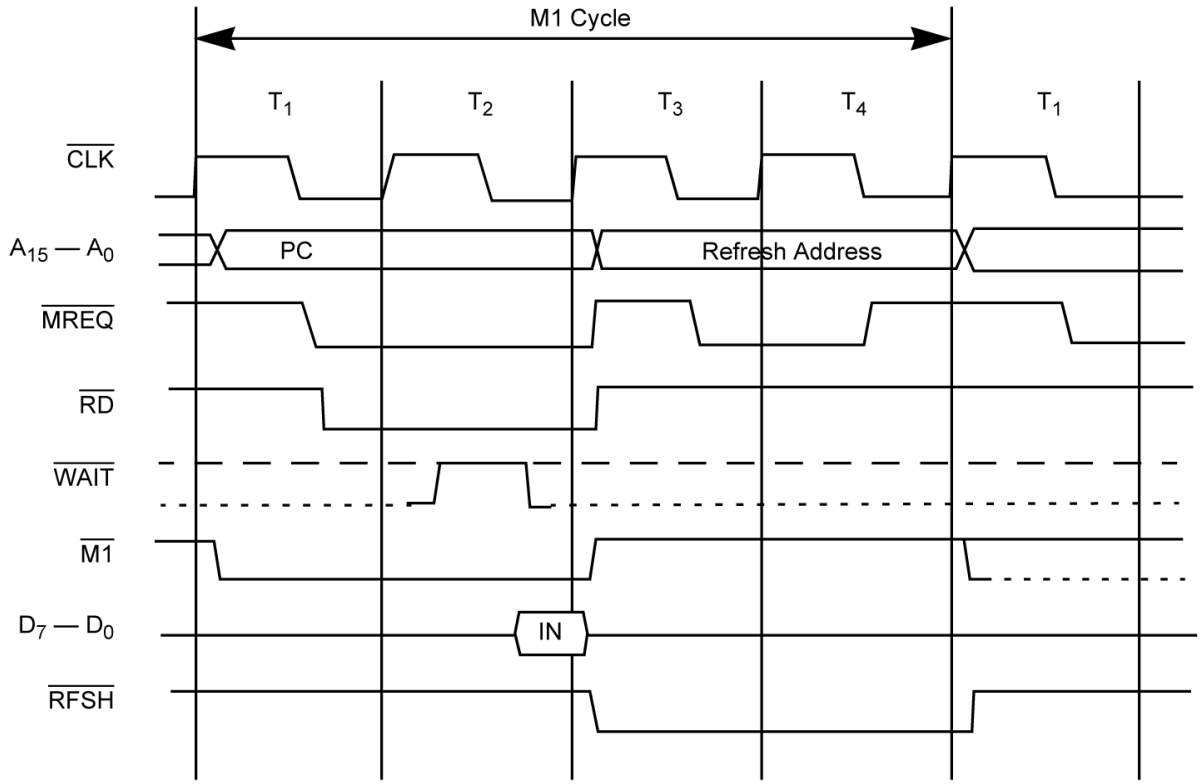


Figure 5. Instruction Op Code Fetch

Temporizzazione: ciclo memory read/write

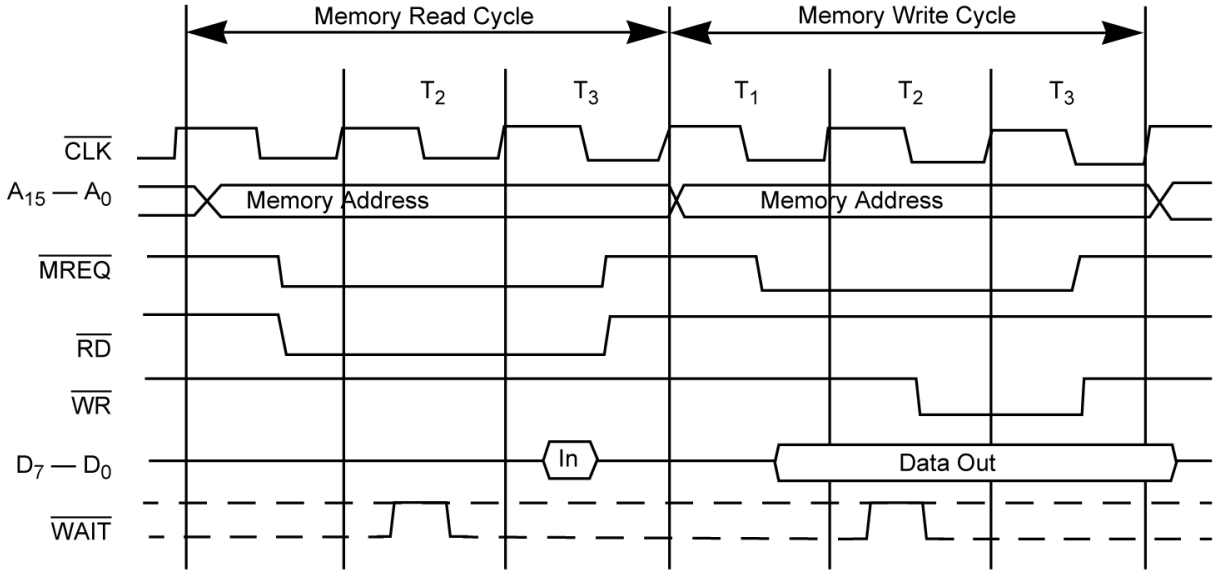


Figure 6. Memory Read or Write Cycle

Temporizzazione: ciclo I/O read/write

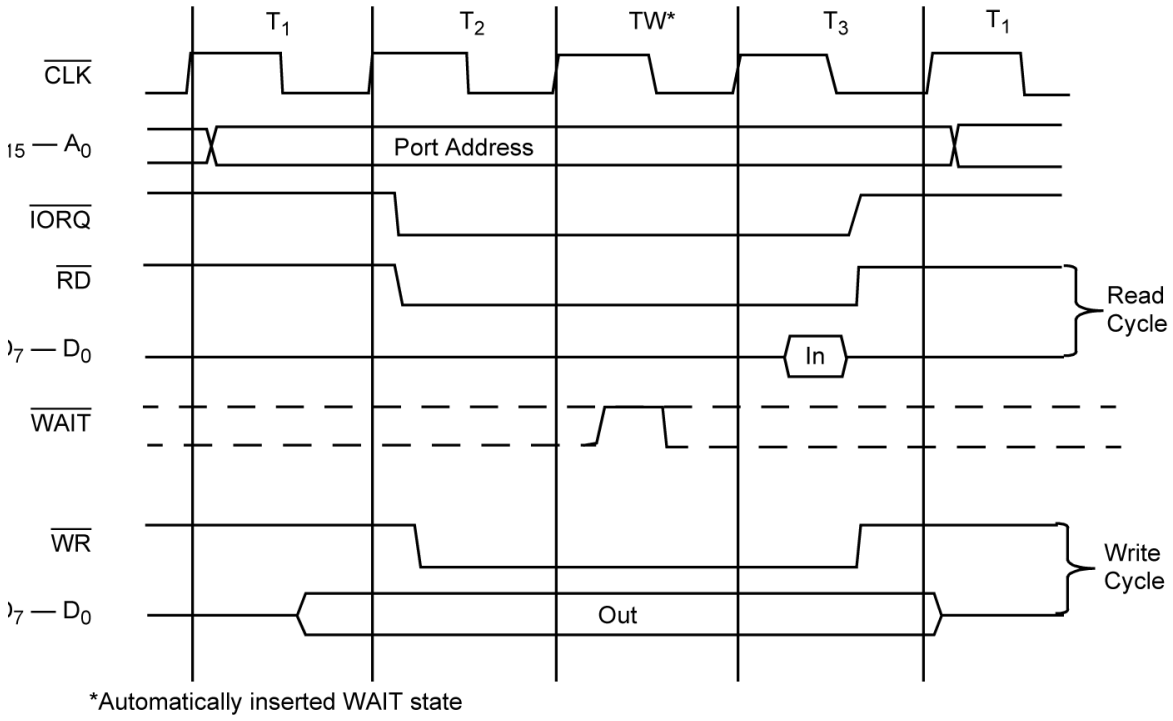


Figure 7. Input or Output Cycles

Temporizzazione ciclo BUSREQUEST

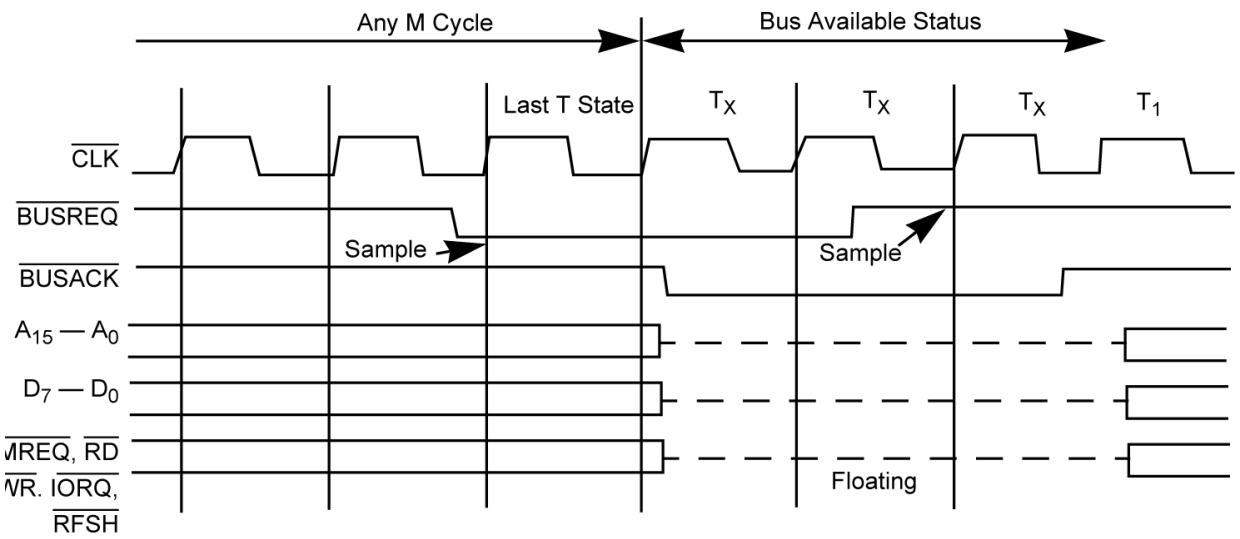


Figure 8. Bus Request/Acknowledge Cycle

Codici istruzione

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LD BC, **	LD (BC),A	INC BC	INC B	DEC B	LD B,*	RLCA	EXAF, AF'	ADD HL, BC	LD A, (BC)	DEC BC	INC C	DEC C	LD C, *	RRCA
1	DJNZ *	LD DE, **	LD(DE) ,A	INC DE	INC D	DEC D	LD D, *	RLA	JR *	ADD HL , DE	LD A, (DE)	DEC DE	INC E	DEC E	LD E, *	RRA
2	JRNZ *	LD HL, **	LD (**), HL	INC HL	INC H	DEC H	LD H, *	DAA	JRZ, *	ADD HL, HL	LD HL, (**)	DEC HL	INC L	DEC L	LD L, *	CPL
3	JRNC *	LD SP, **	LD (**), A	INC SP	INC (HL)	DEC (HL)	LD (HL), *	SCF	JRC, *	ADD HL, SP	LD A, (**)	DEC SP	INC A	DEC A	LD A, *	CCF
4	LD B, B	LDB , C	LD B, D	LD B, E	LD B, H	LDB , L	LD B, (HL)	LD B, A	LD C, B	LD C, C	LD C, D	LD C, E	LD C, H	LD C, L	LD C, (HL)	LD C, A
5	LD D, B	LD D, C	LD D, D	LD D, E	LD D , H	LD D, L	LD D, (HL)	LD D, A	LD E, B	LD E, C	LD E, D	LD E, E	LD E, H	LD E, L	LD E, (HL)	LD E, A
6	LD H, B	LD H, C	LD H, D	LD H, E	LD H, H	LD H, L	LD H, (HL)	LD H, A	LD L, B	LD L, C	LD L, D	LD L, E	LD L,H	LD L, L	LD L, (HL)	LD L, A
7	LD (HL), B	LD (HL), C	LD (HL), D	LD (HL), E	LD (HL), H	LD (HL), L	HALT	LD (HL), A	LD A, B	LD A, C	LD A, D	LD A, E	LD A, H	LD A, L	LD A, (HL)	LDA, A
8	ADD A, B	ADD A, C	ADD A, D	ADD A, E	ADD A, H	ADD A, L	ADD A, (HL)	ADD A, A	ADC A, B	ADC A, C	ADC A, D	ADC A, E	ADC A, H	ADC A, L	ADC A, (HL)	ADC A, A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB (HL)	SUB A	SBC A, B	SBC A, C	SBC A, D	SBC A, E	SBC A, H	SBC A, L	SBC A, (HL)	SBC, A
A	AND B	AND C	AND D	AND E	AND H	AND L	AND (HL)	AND A	XOR B	XOR C	XOR D	XOR E	XOR H	XOR L	XOR (HL)	XOR A
B	OR B	OR C	OR D	OR E	OR H	OR L	OR (HL)	OR A	CP B	CP C	CP D	CP E	CP H	CP L	CP (HL)	CP A
C	RETNZ	POP BC	JPNZ, **	JP **	CALL NZ, **	PUSH BC	ADD A, *	RST 0H	RET Z	RET	JPZ, **	[1]	CALL Z, **	CALL **	ADC A, *	RST 8H
D	RET NC	POP DE	JPNC, **	OUT (*), A	CALL NC, **	PUSH DE	SUB *	RST 10H	RET C	EXX	JPC, **	IN A, (*)	CALL C, **	[1]	SBC A, *	RST 18H
E	RET PO	POP HL	JP PO, **	EX(SP) HL	CALL PO, **	PUSH HL	AND *	RST 20H	RET PE	JP (HL)	JP PE, **	EX DE,HL	CALL PE,**	[1]	XOR *	RST 28H
F	RET P	POP AF	JP P, **	DI	CALL P, **	PUSH AF	OR *	RST 30H	RET M	LD SP, HL	JPM, *	EI	CALL m, **	[1]	CP *	RST 38H

Altri esempi di programmi

Somma di due numeri a 16 bit:
gli operandi sono nelle locazioni di memoria ADR1 e ADR2.

```
LDA,(ADR1)      carica la meta' bassa di OP1
LD HL,ADR2      indirizza la meta' bassa di OP2
ADDA,(HL)       somma
LD(ADR3),A      salva in ADR3 il risultato (basso)
LDA,(ADR1-1)    carica la meta' alta di OP1
DEC HL          indirizza la meta' alta di OP2
ADC A,(HL)      somma le parti alte di OP1, OP2 e carry
LD(ADR3-1),A    salva in ADR3-1 il risultato (alto)
```

Sottrazione di due numeri a 16 bit:
gli operandi sono nelle locazioni di memoria ADR1 e ADR2.

```
LD HL,(ADR1)   carica in HL il contenuto di ADR1 e ADR1+1
LD DE,(ADR2)   carica in DE il contenuto di ADR2 e ADR2+1
AND A          azzera riporto
SBC HL,DE      sottrae gli operandi
LD(ADR3),DE    salva in ADR3
```

Istruzioni di input - output

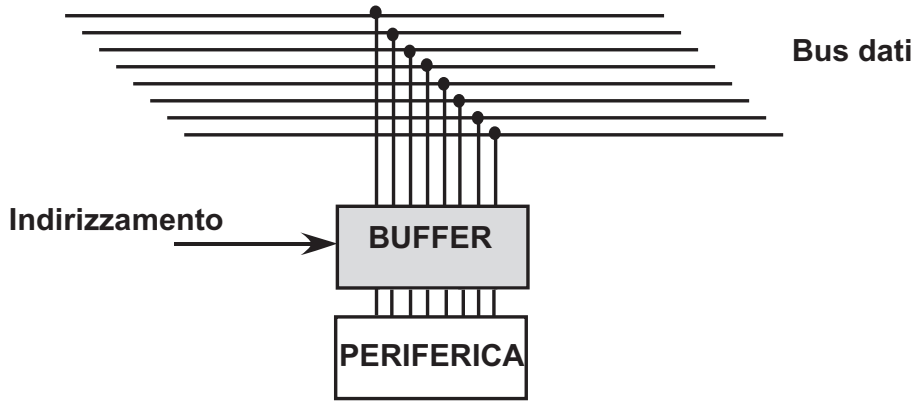
Queste istruzioni attivano i cicli di read o write con periferiche (linea $\overline{IORQ}$).

Lettura di una periferica

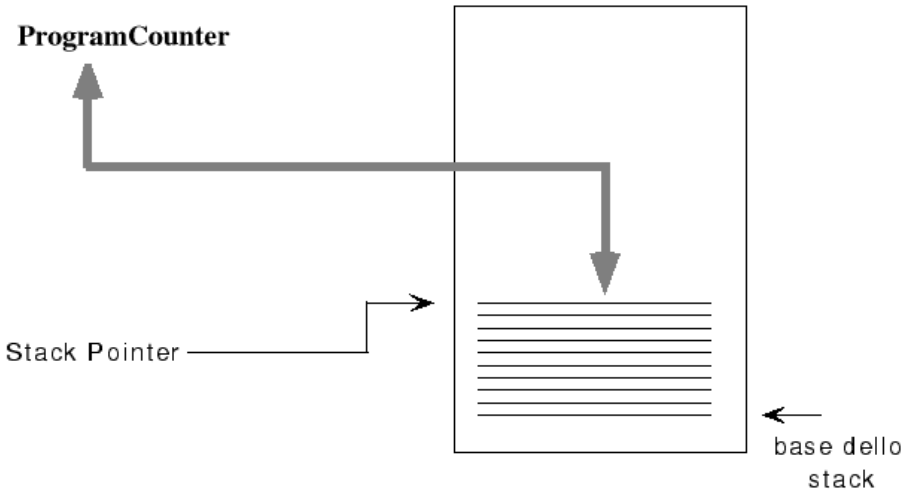
```
 Codice      Mnemonico
  DB         IN A,(nn)  La periferica nn e' letta ed
  nn                    il risultato e' caricato in A
```

Scrittura di una periferica

```
 Codice      Mnemonico
  D3         OUT(nn),A  Il contenuto di A e' scritto
  nn                    sulla periferica nn
```



Lo stack (catasta)



Subroutine

Lo Z80 consente al programmatore di costruire programmi strutturati con subroutines.
In sostanza, e' possibile effettuare salti con la possibilita' di ritornare al punto di partenza (che deve quindi essere ricordato).
Il meccanismo che consente cio' e' quello dello stack, alla cui gestione e' deputato lo StackPointer.
Occorre, inizialmente, caricare nello StackPointer un indirizzo corrispondente ad una zona della memoria libera, cioe' non utilizzata per altri scopi. Quando, nel corso del programma, viene incontrata l'istruzione

```
Codice          Mnemonico
 CD .. ..        CALL nn
```

lo Z80 effettua le seguenti operazioni:
- il contenuto del ProgramCounter viene trasferito nella locazioni (SP-1) e (SP-2);
- il contenuto dello StackPointer viene decrementato di 2;
- nn viene trasferito nel ProgramCounter.

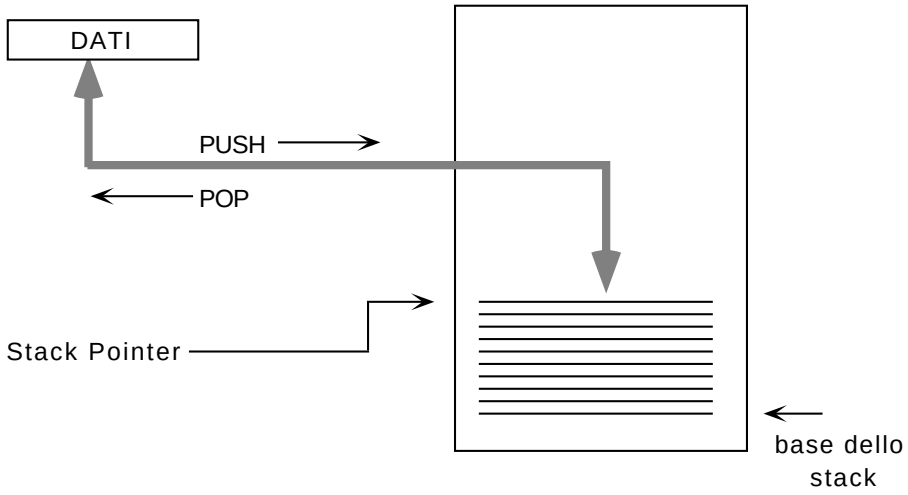
L'ultima istruzione della subroutine deve essere l'istruzione

```
Codice    Mnemonico
 C9           RET
```

- il contenuto delle locazioni (SP) e (SP+1) e' trasferito nel ProgramCounter;
- il contenuto dello StackPointer viene incrementato di 2.

Lo stack puo' essere utilizzato anche per immagazzinamento dati, gestione degli interrupts.
Lo stack puo' essere utilizzato con le istruzioni PUSH e POP.
PUSH pq: il contenuto della coppia di registri pq e' scritto nello stack e lo StackPointer e' decrementato di 2.

POP pq: il contenuto dello stack e' scritto nella coppia di registri pq e lo Stack-Pointer e' incrementato di 2.



Interrupts

L'interrupt e' il meccanismo con cui lo Z80 puo' interagire in modo asincrono con l'ambiente esterno.
L'interrupt e' generato da una periferica e richiede allo Z80 di compiere specifiche azioni, ovvero eseguire uno specifico programma.
Esistono 3 meccanismi di interruzione, attivati rispettivamente da

$\overline{BUSRQ}$
$\overline{NMI}$
$\overline{INT}$.

Interrupts - II

$\overline{BUSRQ}$
E' il meccanismo a piu' alta priorita'.
$\overline{NMI}$
E' accettato dopo la fine dell'istruzione corrente, se non e' attivo $\overline{BUSRQ}$. Se $\overline{BUSRQ}$ e' attivo $\overline{NMI}$ e' eseguito dopo.
Provoca un trasferimento automatico del ProgramCounter nello stack; poi viene caricato nel ProgramCounter il contenuto delle loazioni di memoria 0066 e 0067.
Quindi il programmatore deve opportunamente scrivere una routine per servire l'interrupt e metterne l'indirizzo di partenza nella locazione di memoria 0066 e 0067.
$\overline{INT}$
Puo' essere mascherato ponendo a "0" il contenuto dei bit IFF1 e IFF2 con l'istruzione DI.
Al contrario, si abilita con l'istruzione EI, che li mette a "1".
Esistono 3 modalita' di risposta, selezionate con le istruzioni

```
Codice      Mnemonico
ED 46       IM0          seleziona modo 0
ED 56       IM1          seleziona modo 1
ED 5E       IM2          seleziona modo 2
```

Interrupts - III

Interrupt di modo 0

Lo Z80 legge il bus dei dati e interpreta il dato letto come un'istruzione da eseguire.

Generalmente la periferica porra' sul bus dei dati un istruzione tipo RST, che ramifica a 8 possibili indirizzi:

Codice	Mnemonico	
C7	RST 00	Salta alla locazione 00
CF	RST 08	Salta alla locazione 08
D7	RST 10	Salta alla locazione 10
DF	RST 18	Salta alla locazione 18
E7	RST 20	Salta alla locazione 20
EF	RST 28	Salta alla locazione 28
F7	RST 30	Salta alla locazione 30
FF	RST 38	Salta alla locazione 38

Interrupt di modo 1
Lo Z80 esegue l'istruzione

```
Codice      Mnemonico
FF           RST 38     Salta alla locazione 38
```

Interrupt di modo 2
Lo Z80 legge il bus dei dati e carica sul ProgramCounter un indirizzo formato dal contenuto del registro I (parte piu' significativa) e dal dato letto (parte meno significativa).
in questo modo possono essere serviti in modo differenziato molti dispositivi di I/O.

Quando lo Z80 riceve un interrupt, interrompe il processo in esecuzione e si dedica al servizio dell'interrupt eseguendo il programma a cio' predisposto.
Alla fine, dovra' riprendere il processo che era stato abbandonato.
Per consentire di interrompere e riprendere in modo ordinato e semplice un processo, lo Z80 utilizza i registri secondari, tramite le istruzioni:

Codice	Mnemonico	
08	EX AF, AF'	Scambia il contenuto di (AF) con quello di (AF)'
D9	EXX	scambia il contenuto di (BC),(DE),(HL) con quello di (BC)',(DE)',(HL)'

Quindi il programma di gestione di un interrupt dovra' prevedere: - salvataggio dei registri; - gestione dell'interrupt; - ripristino dei registri;

La scheda didattica Z80 - Schema a blocchi

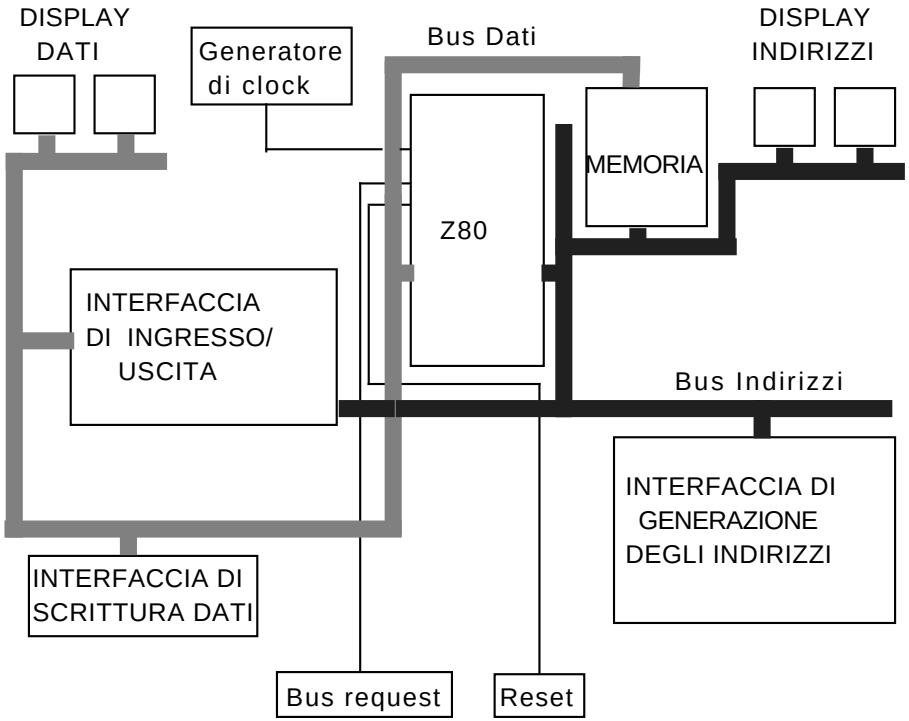
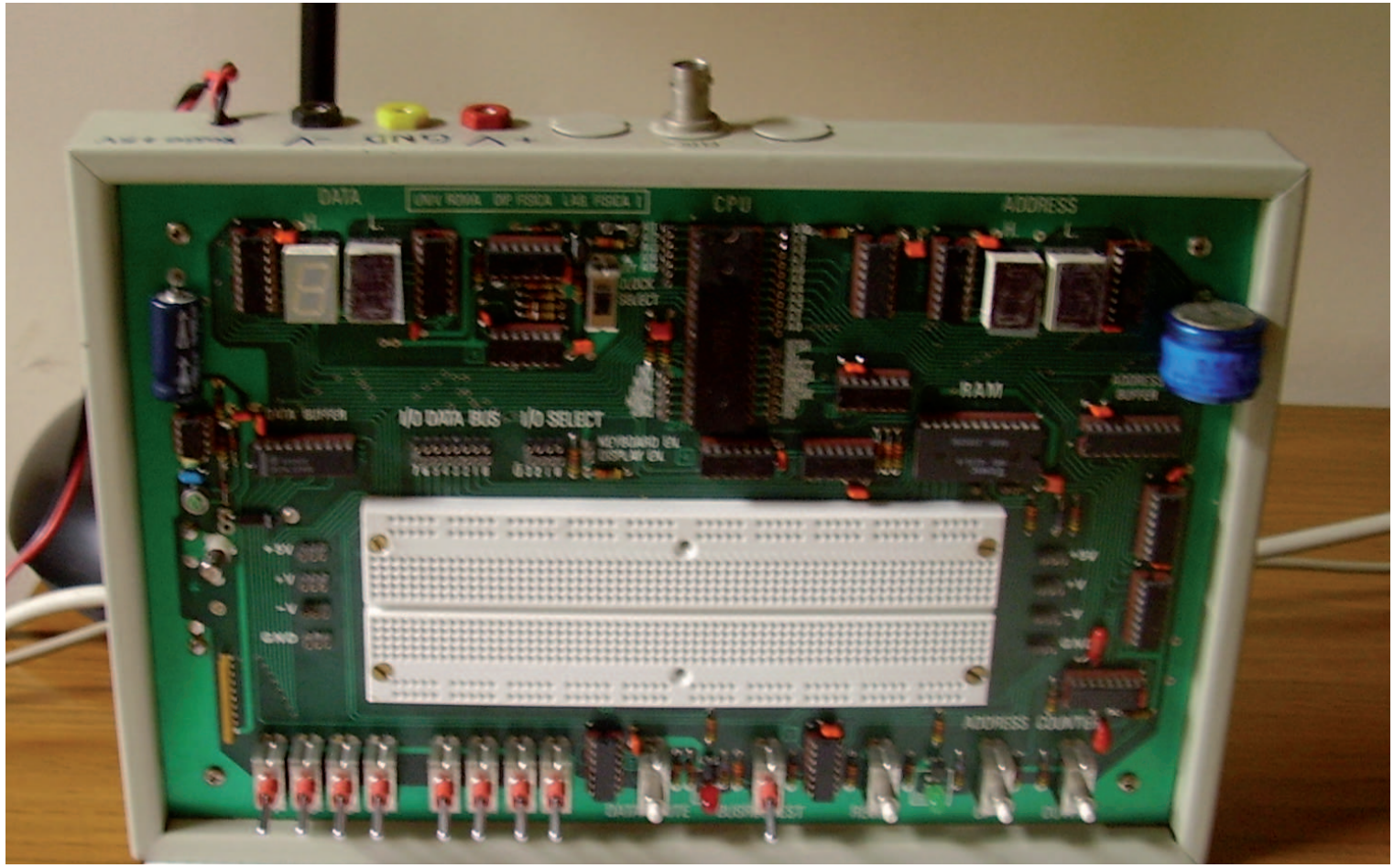
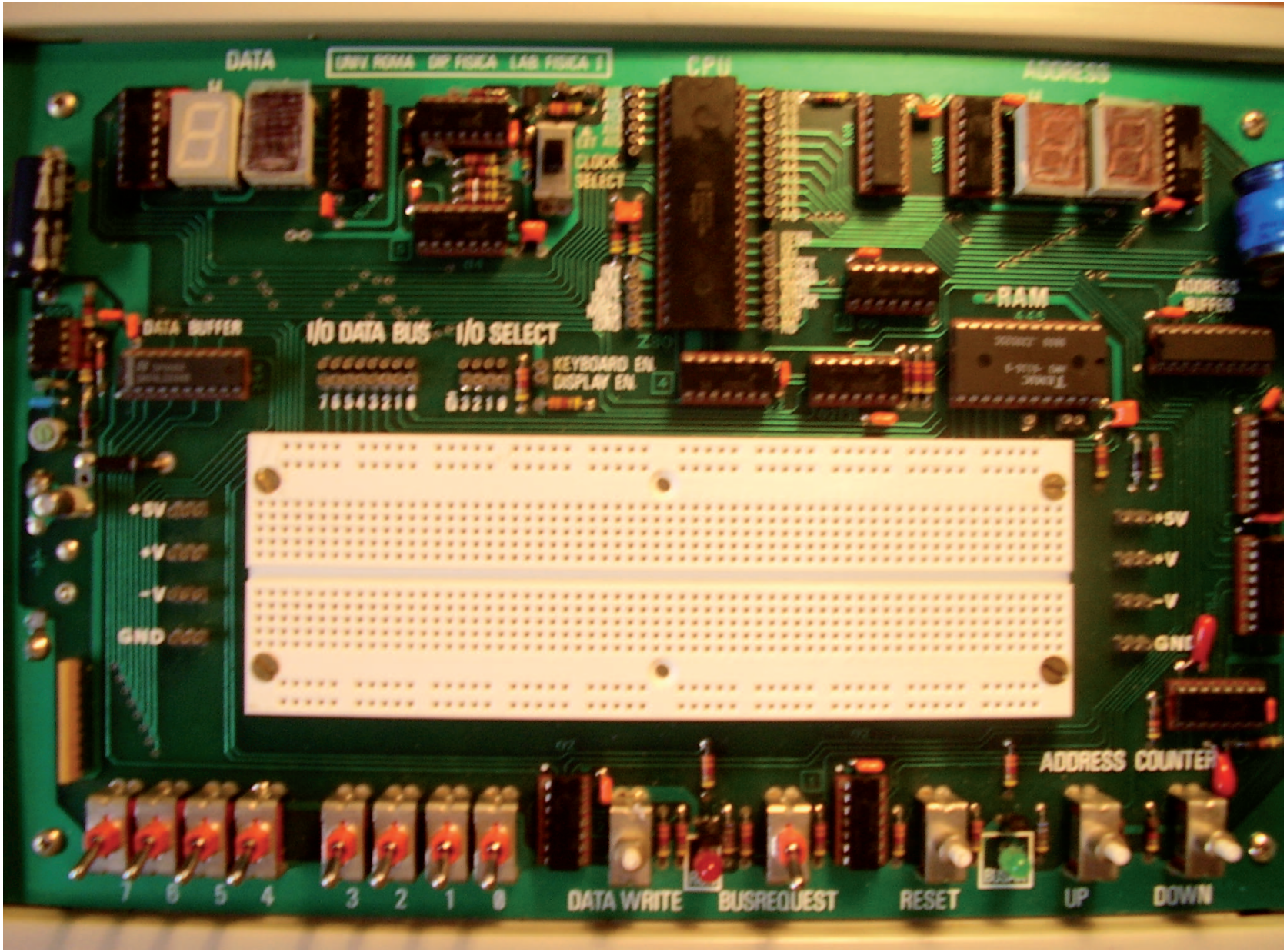


Figura 1: *La scheda didattica Z80: schema a blocchi.*

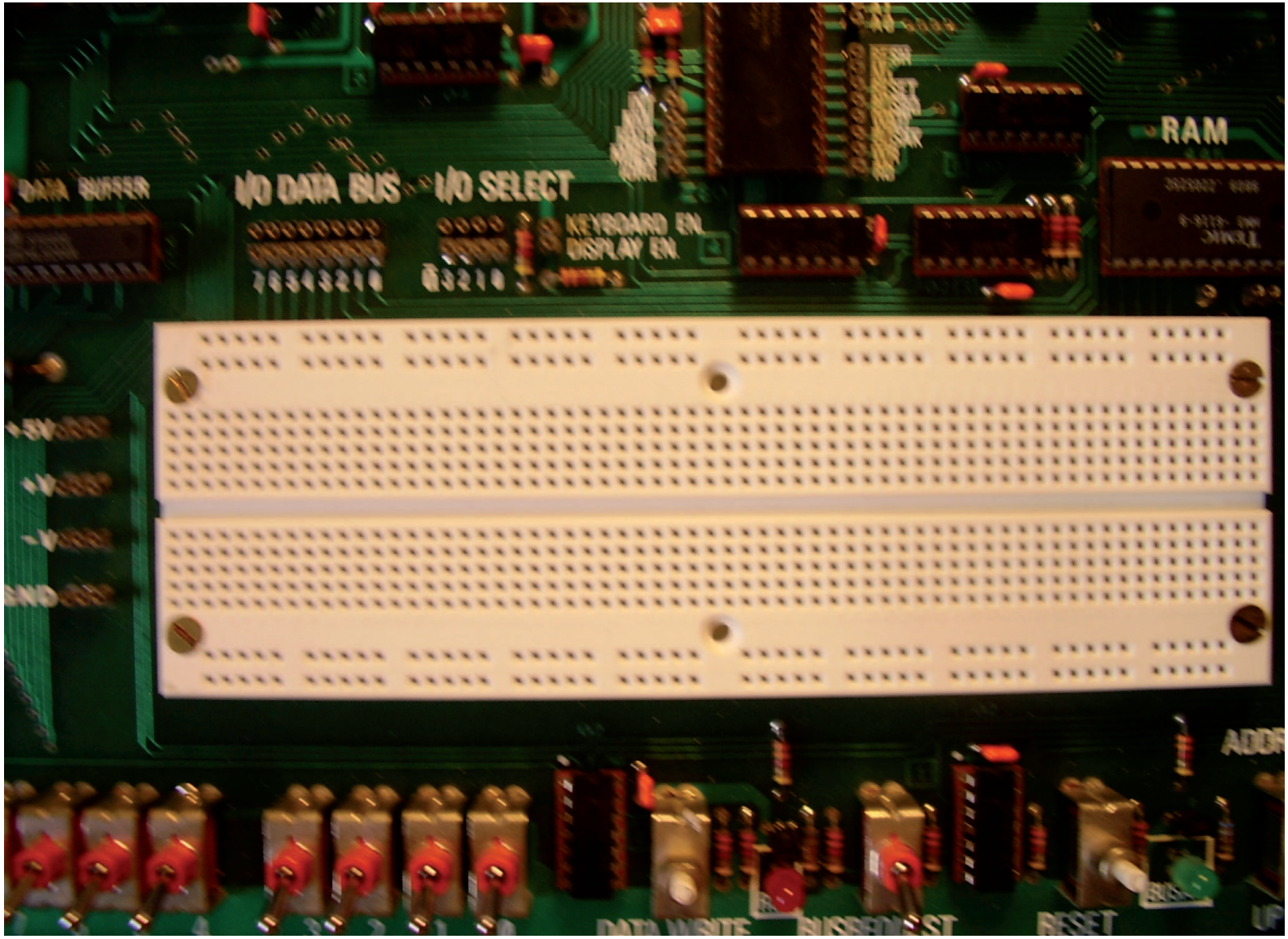
La scheda didattica Z80 - Foto



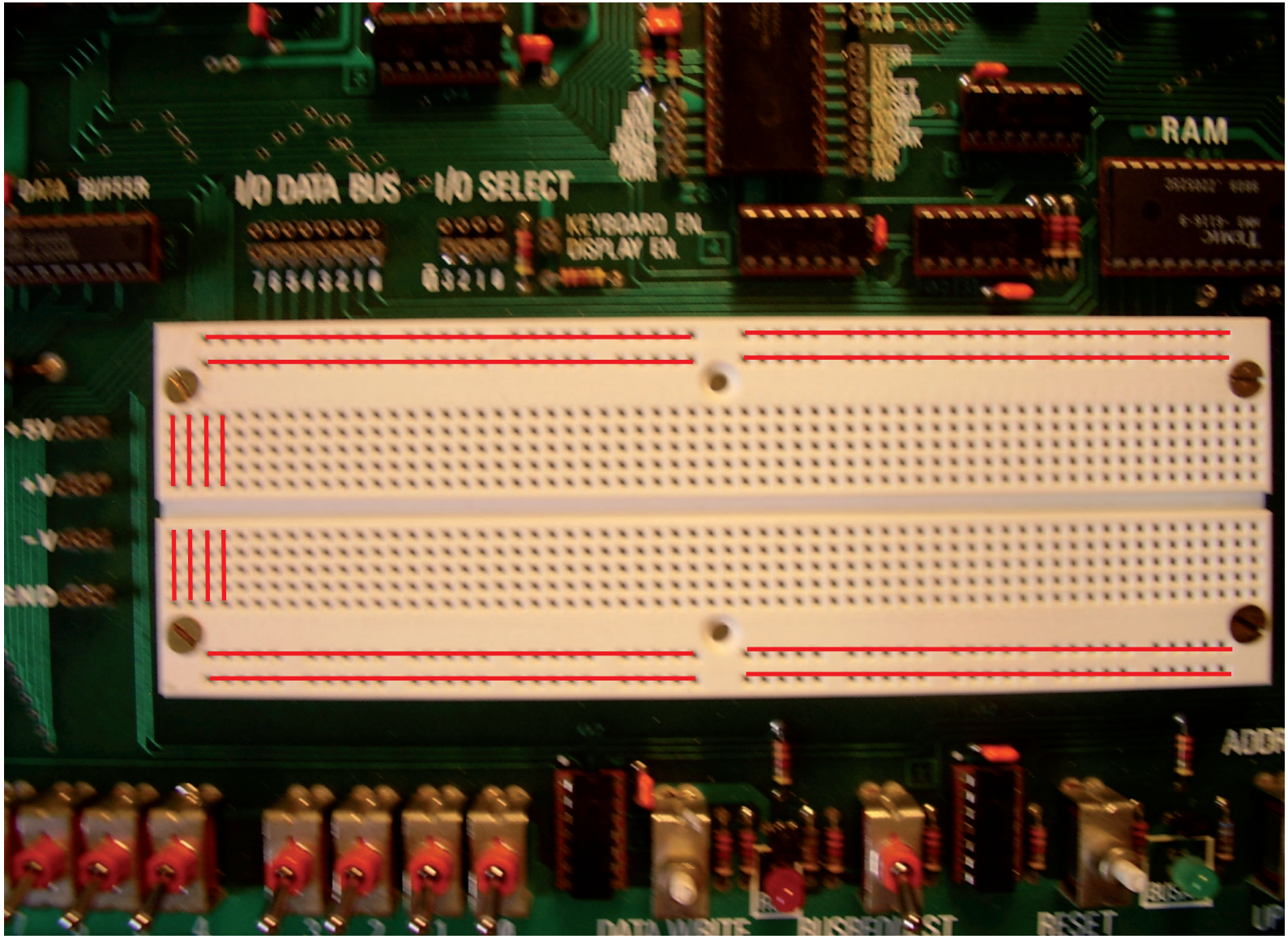
La scheda didattica Z80 - Foto



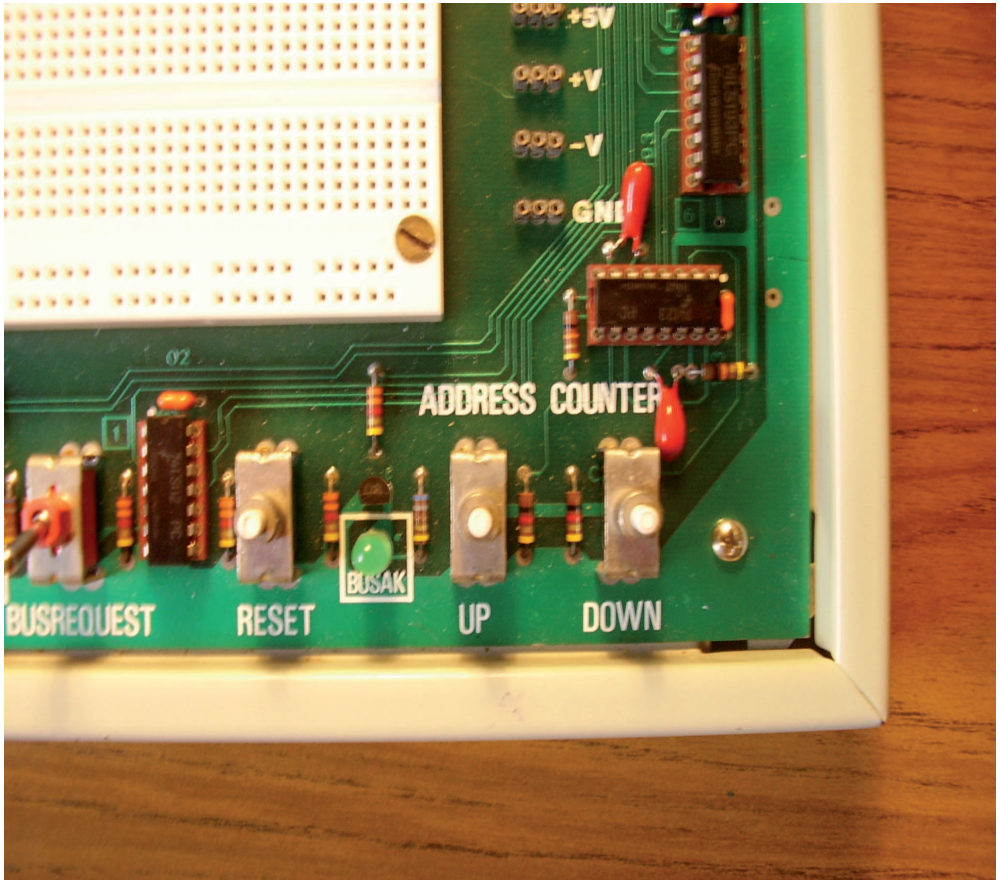
La scheda didattica Z80 - Foto



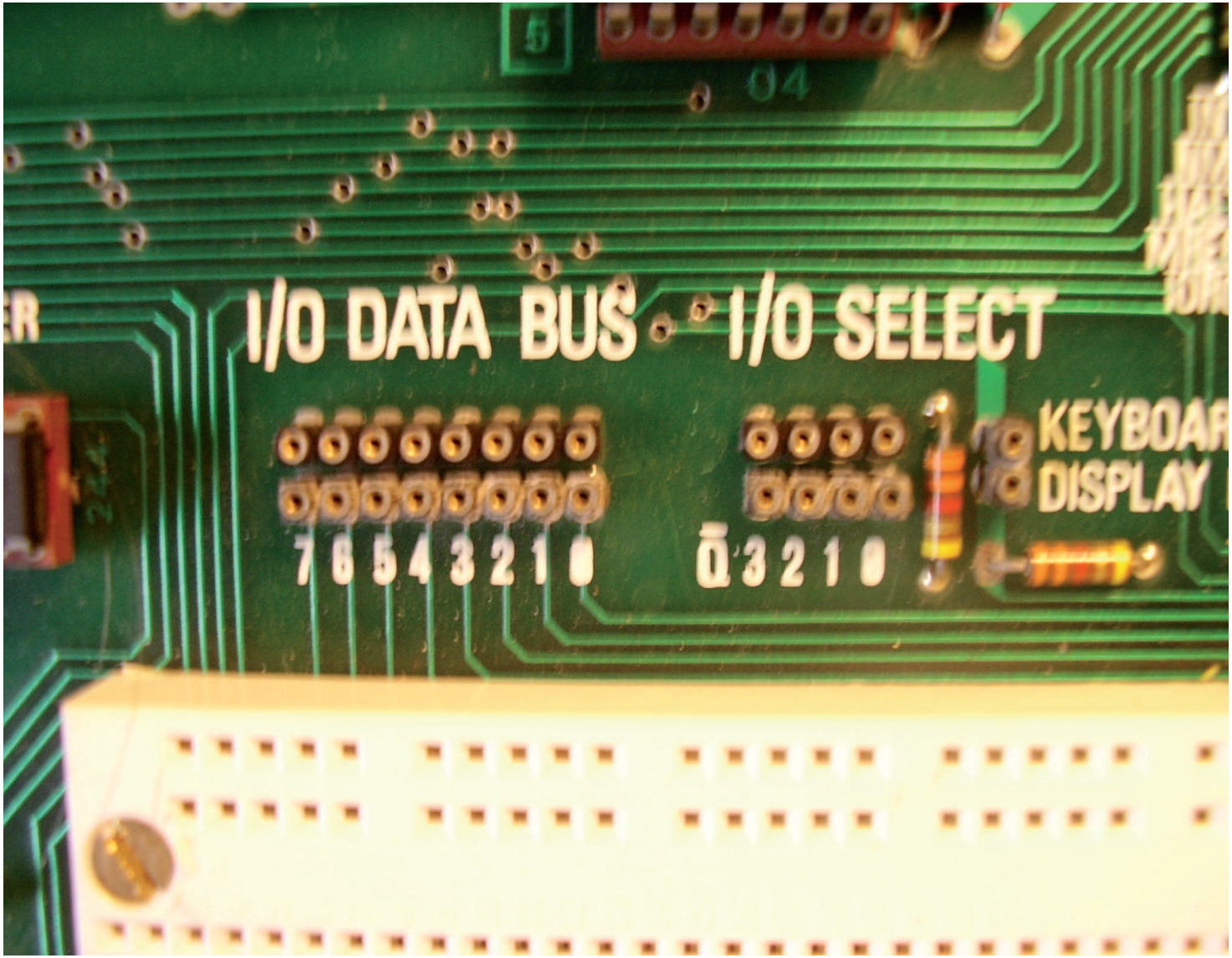
La scheda didattica Z80 - Foto



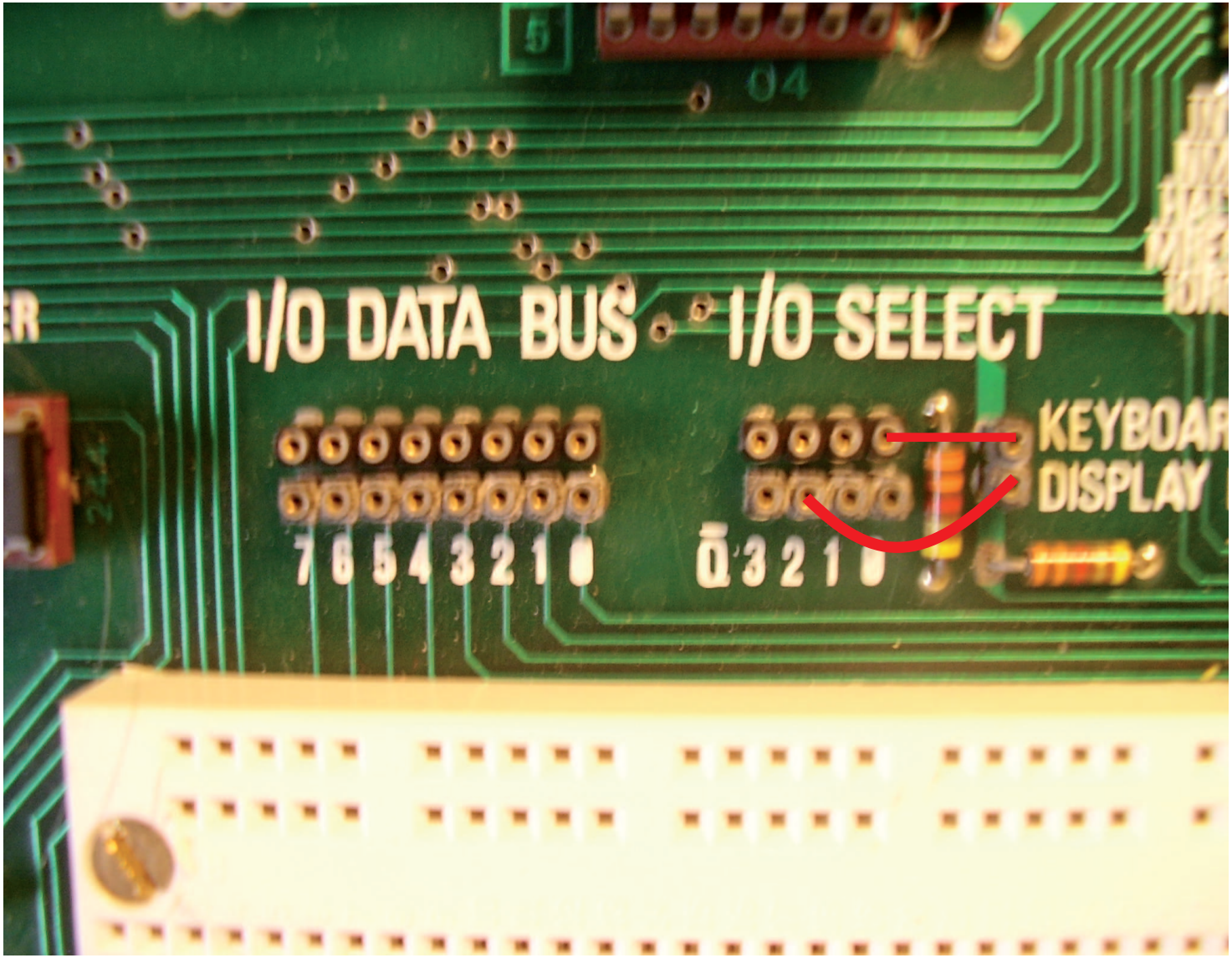
La scheda didattica Z80 - Foto



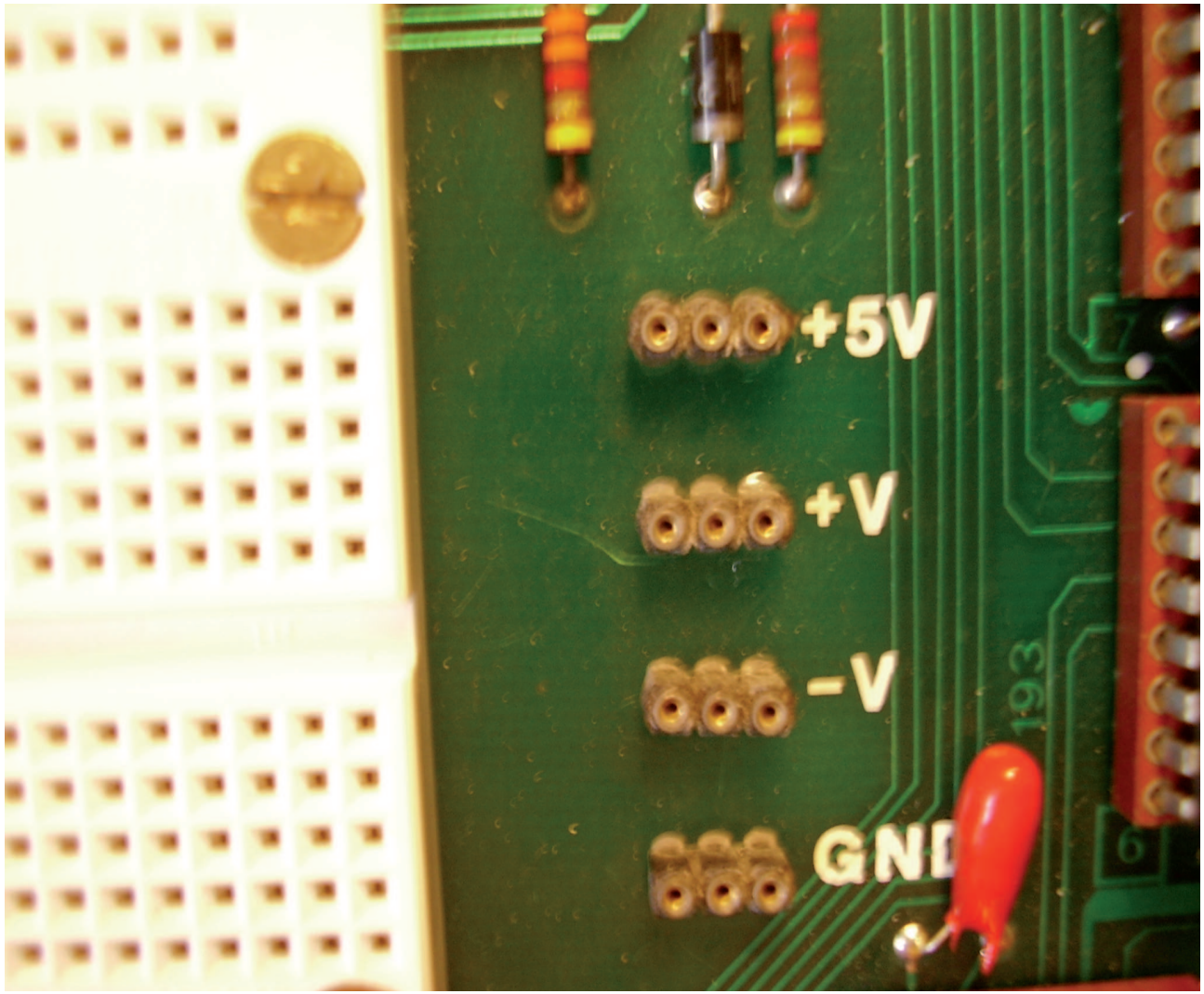
La scheda didattica Z80 - Foto



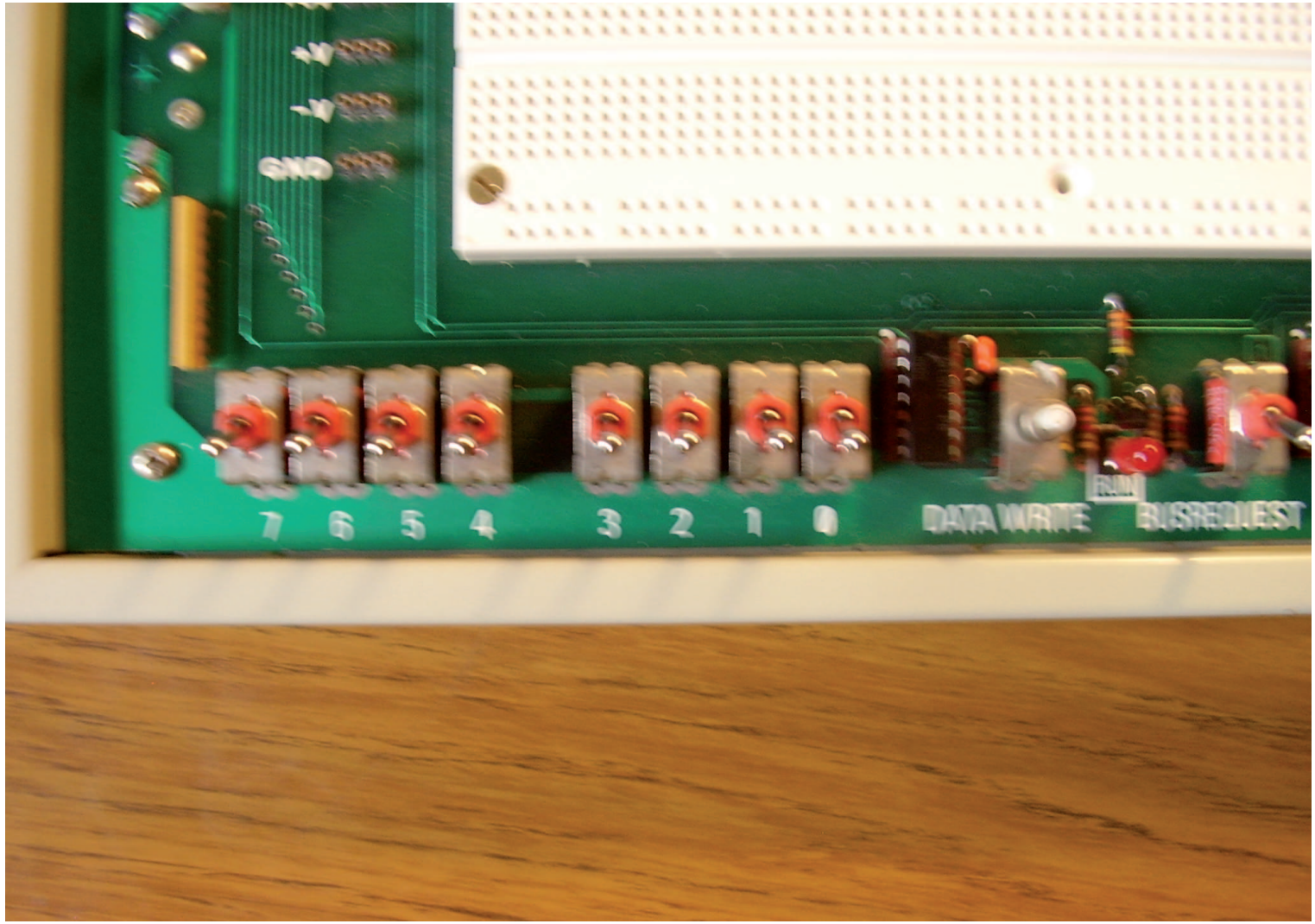
La scheda didattica Z80 - Foto



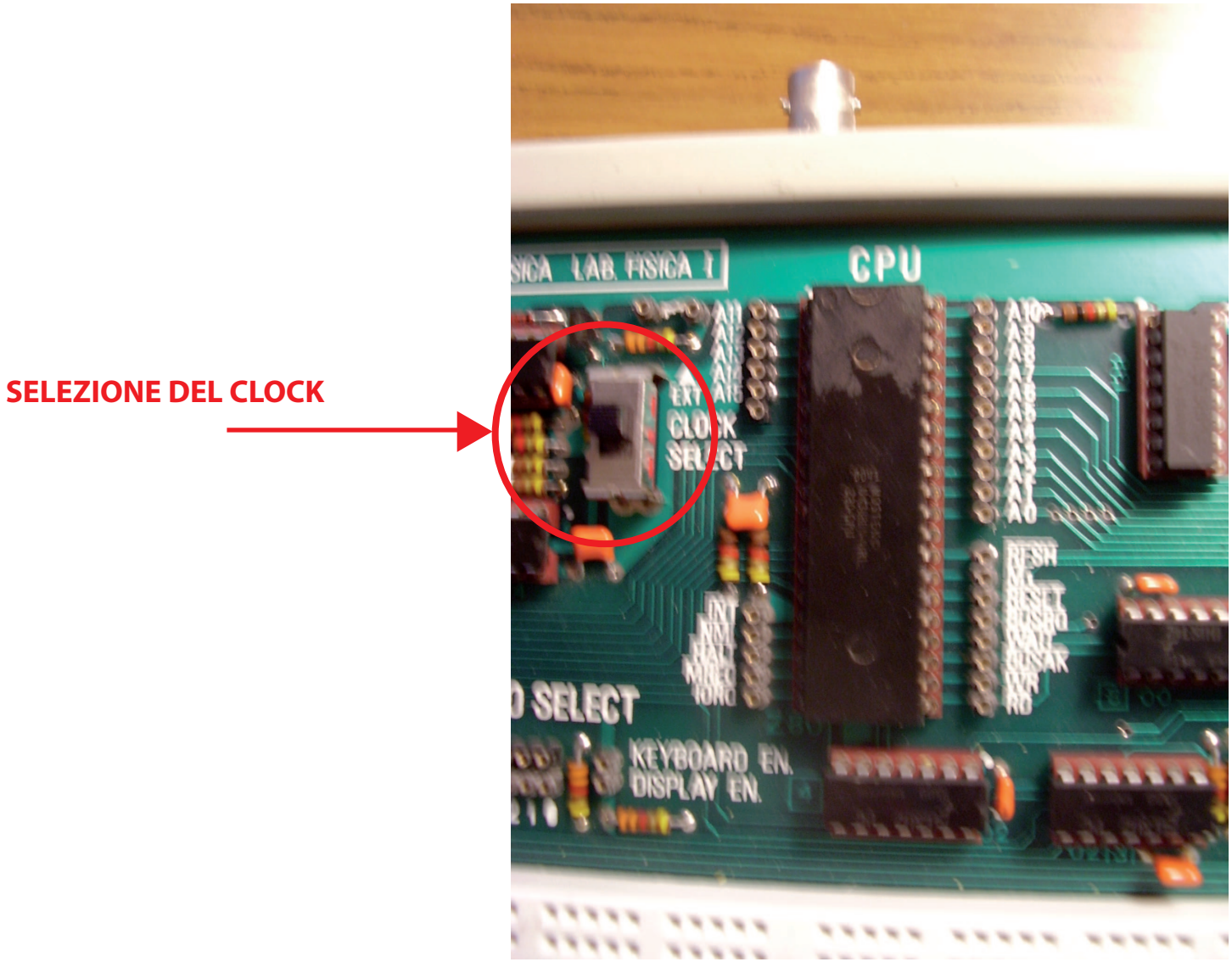
La scheda didattica Z80 - Foto



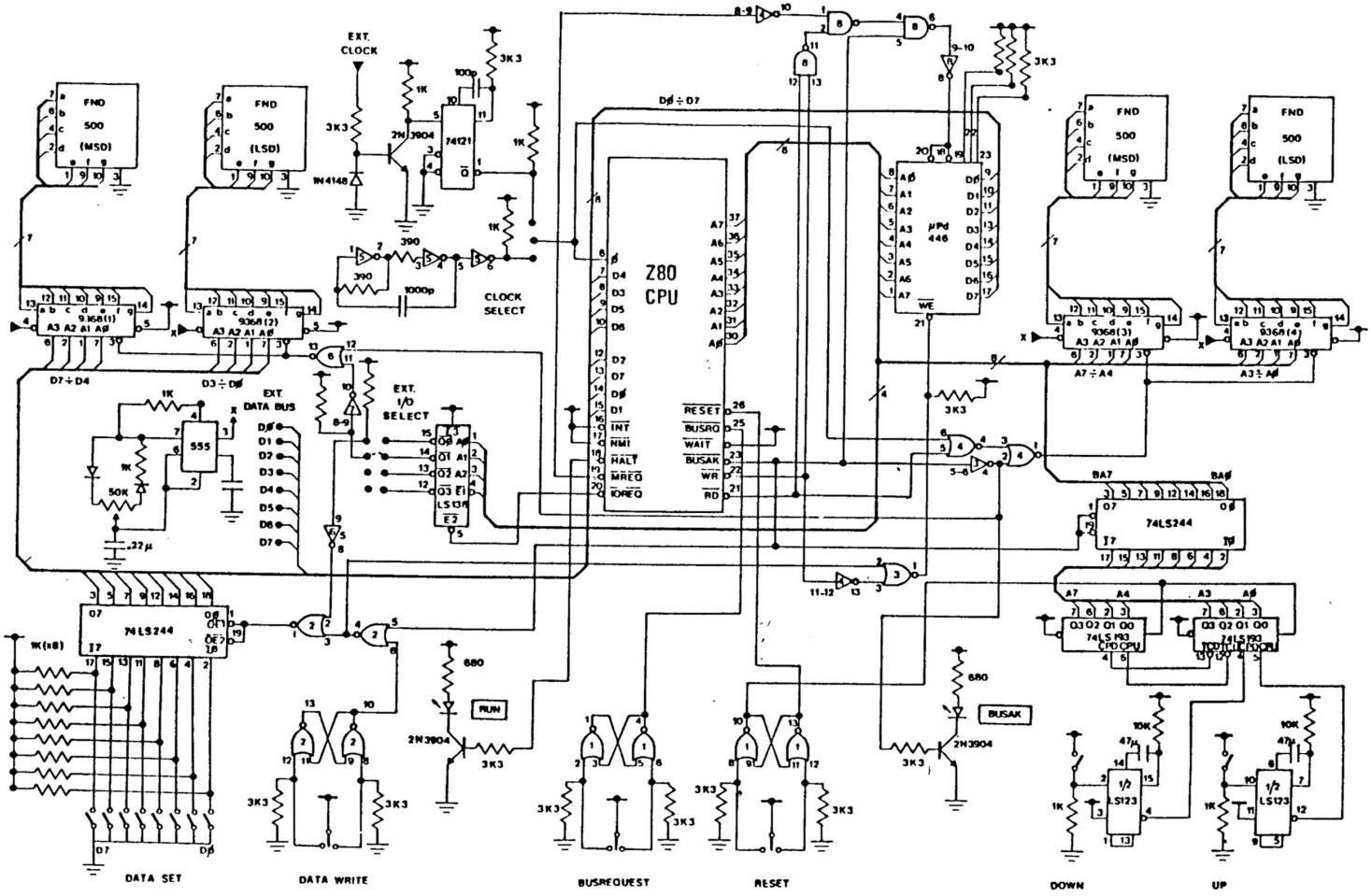
La scheda didattica Z80 - Foto



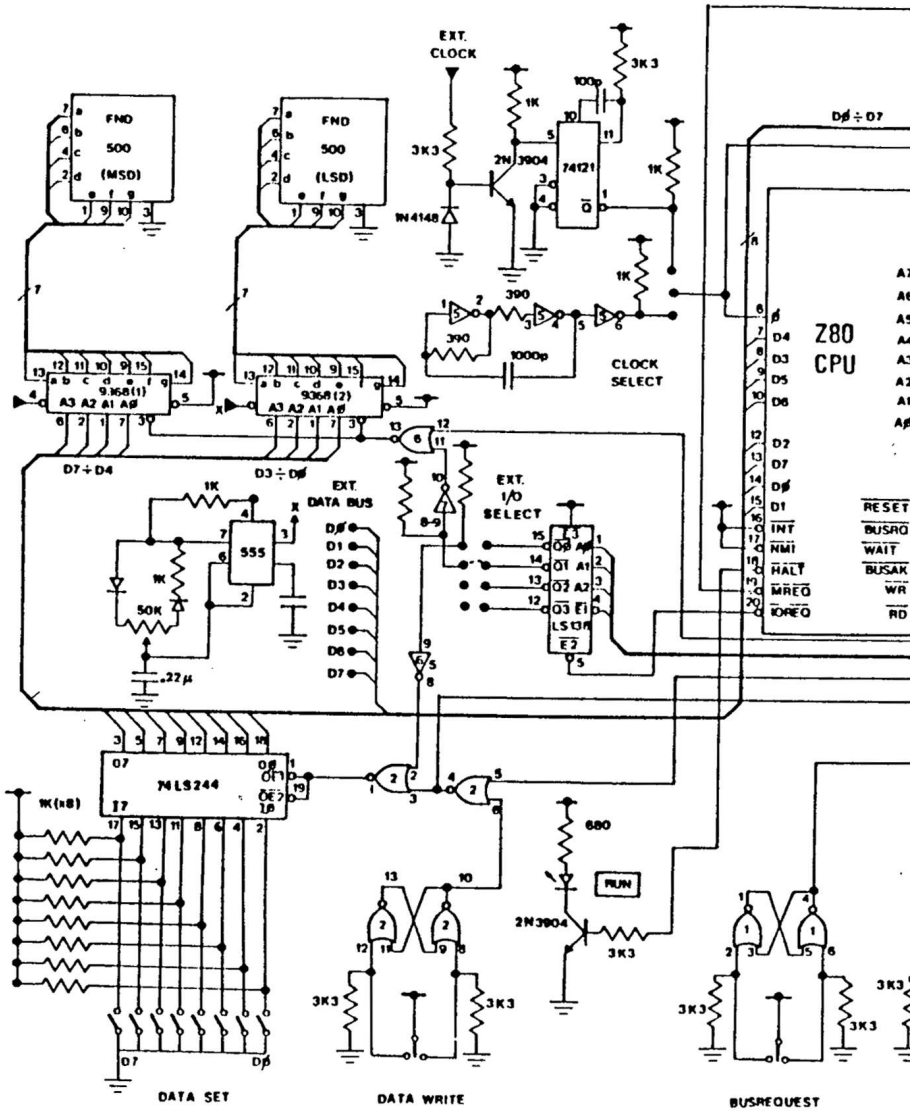
La scheda didattica Z80 - Foto



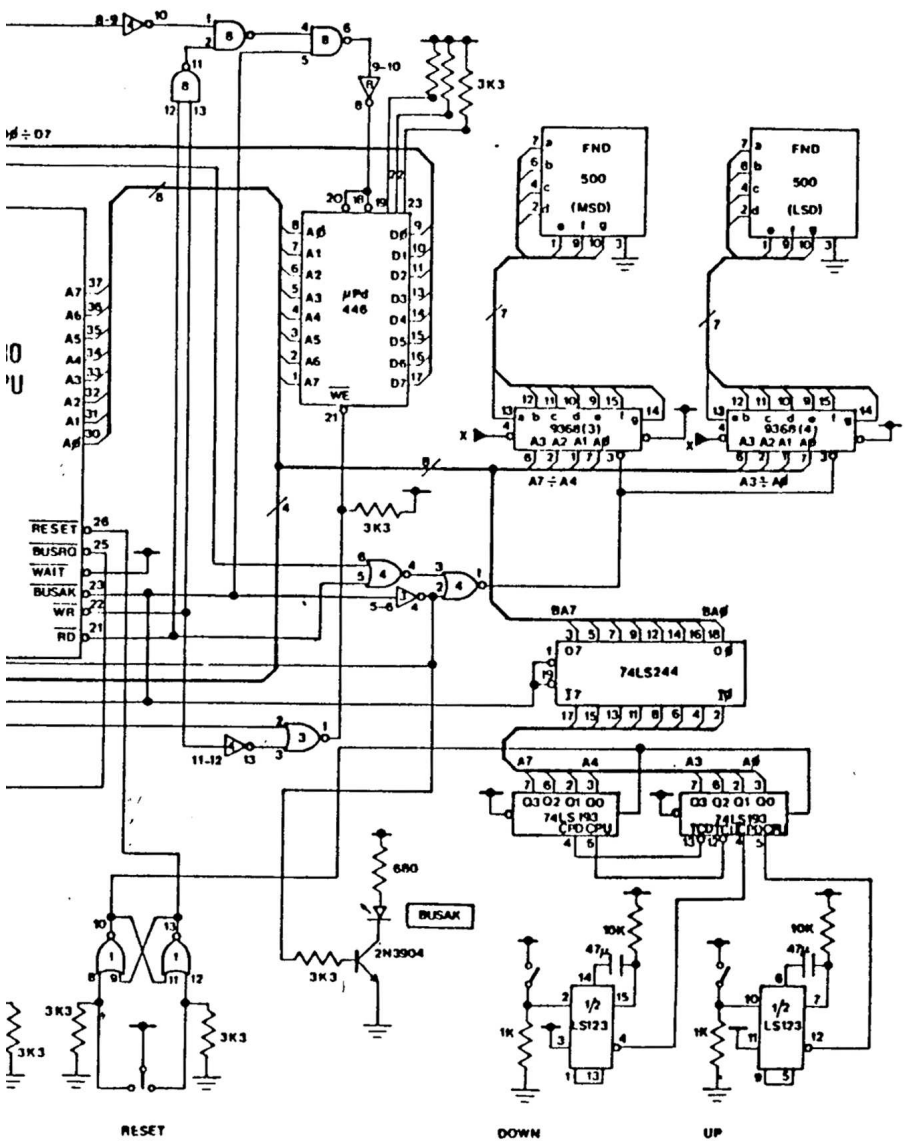
La scheda didattica Z80 - Schema



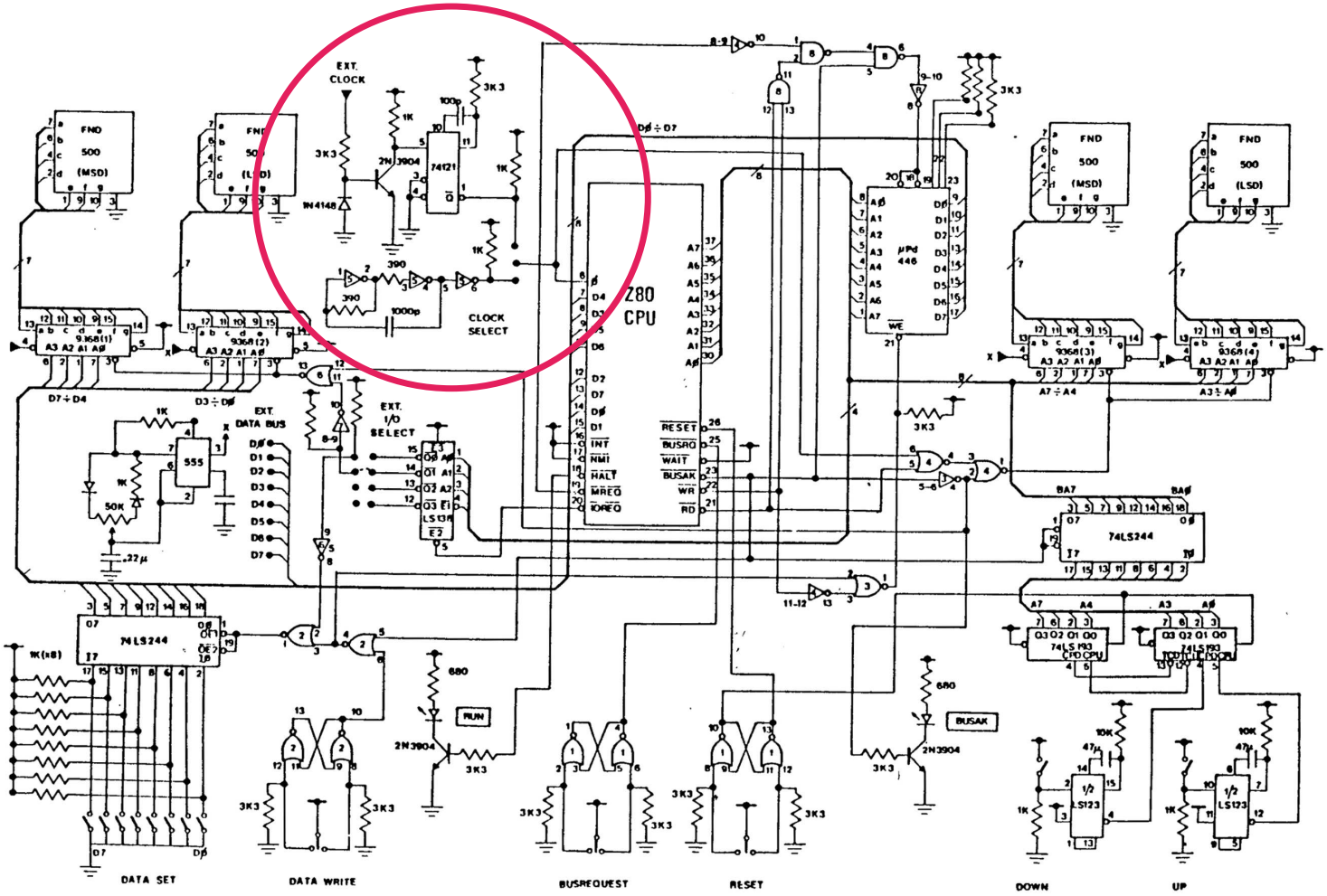
La scheda didattica Z80 - Schema



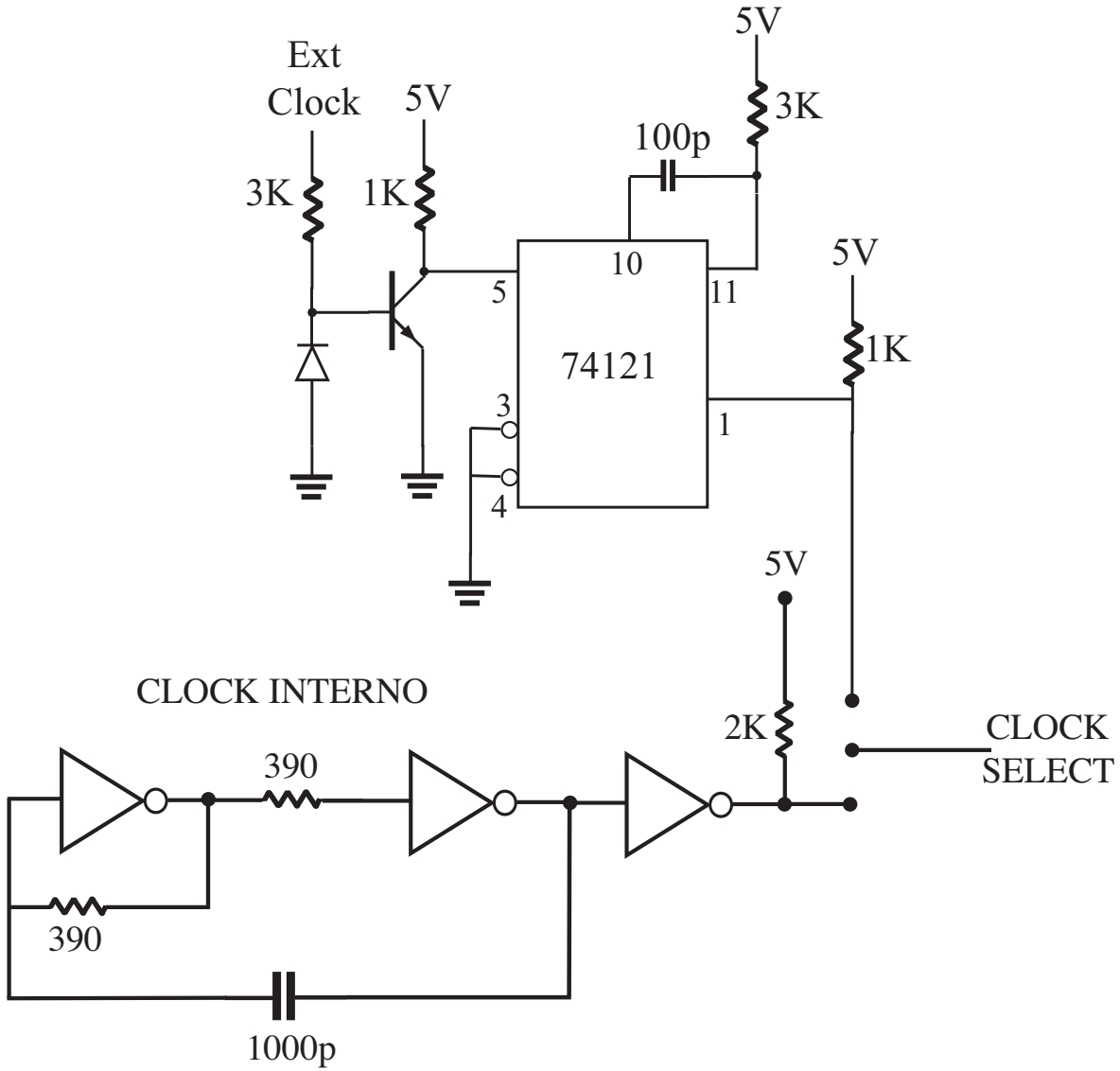
La scheda didattica Z80 - Schema



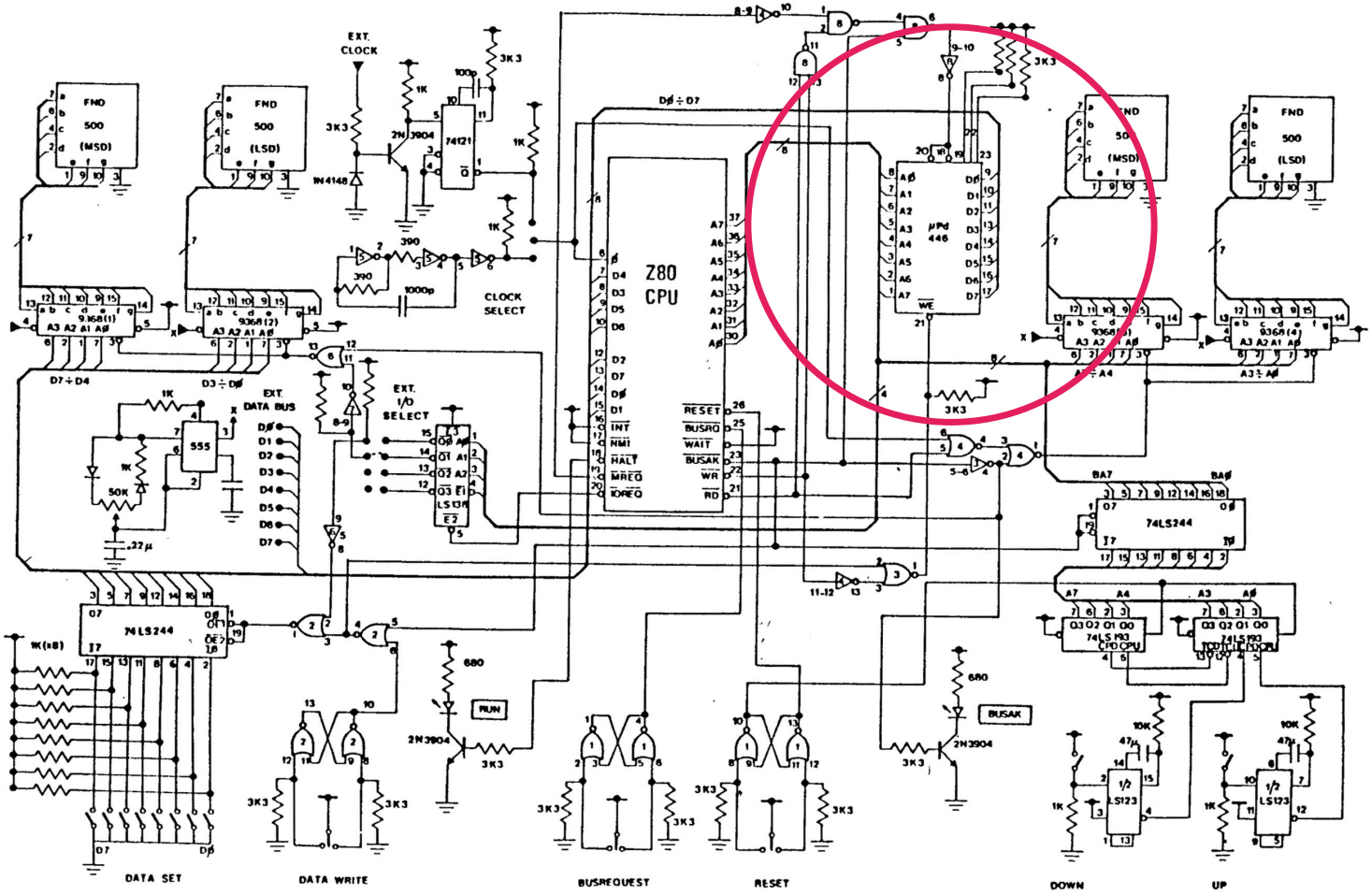
La scheda didattica Z80 - Clock



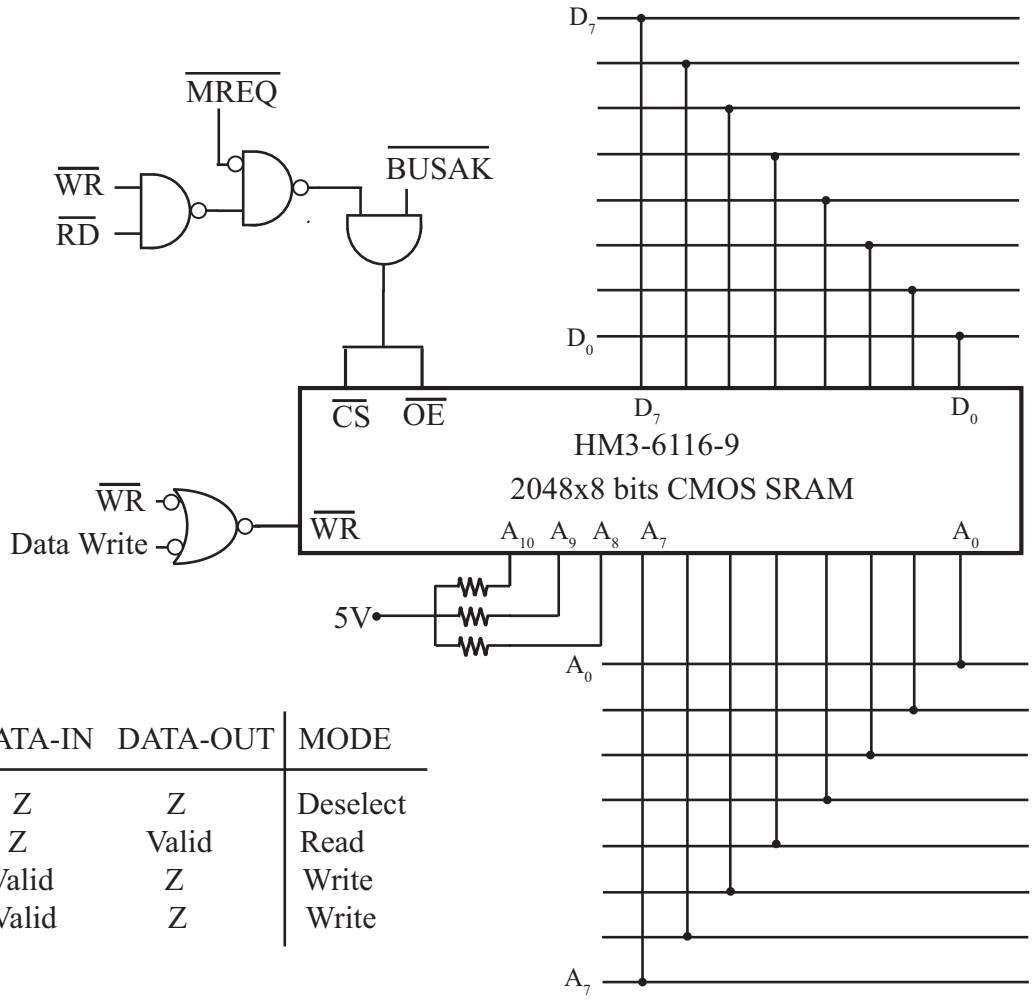
La scheda didattica Z80 - Clock



La scheda didattica Z80 - Memoria

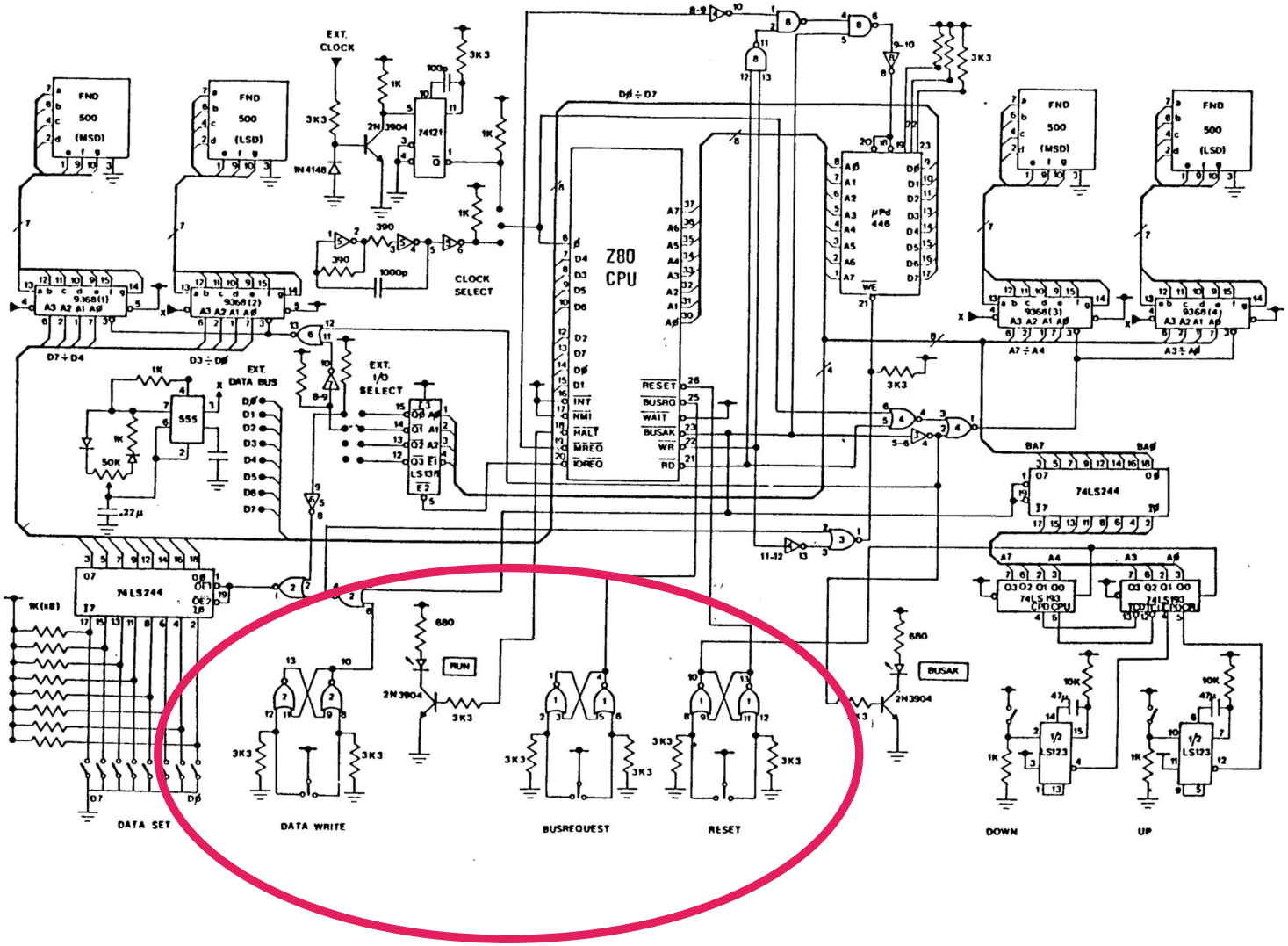


La scheda didattica Z80 - Memoria

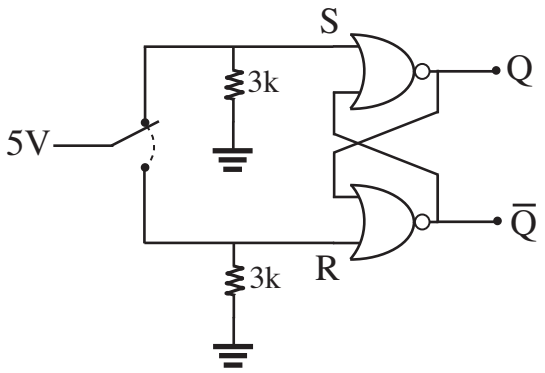


$\overline{CS}$	$\overline{OE}$	$\overline{WR}$	DATA-IN	DATA-OUT	MODE
H	X	X	Z	Z	Deselect
L	H	H	Z	Valid	Read
L	H	L	Valid	Z	Write
L	L	L	Valid	Z	Write

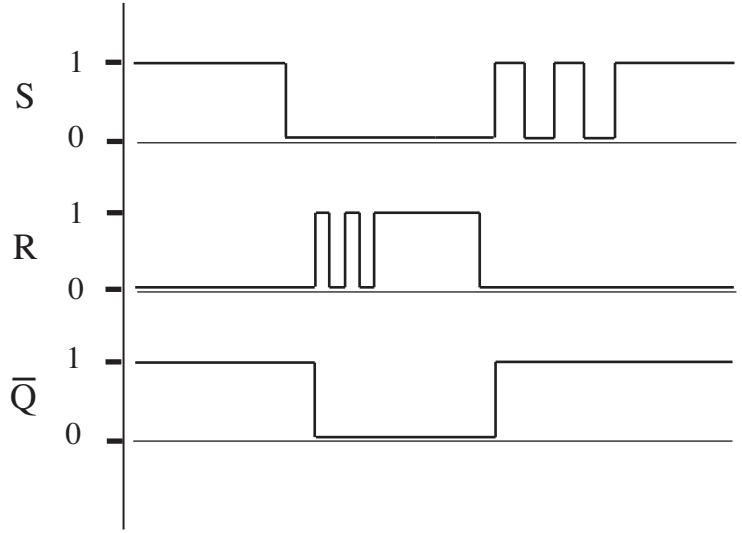
La scheda didattica Z80 - Pulsanti



La scheda didattica Z80 - Anti-rimbalzo

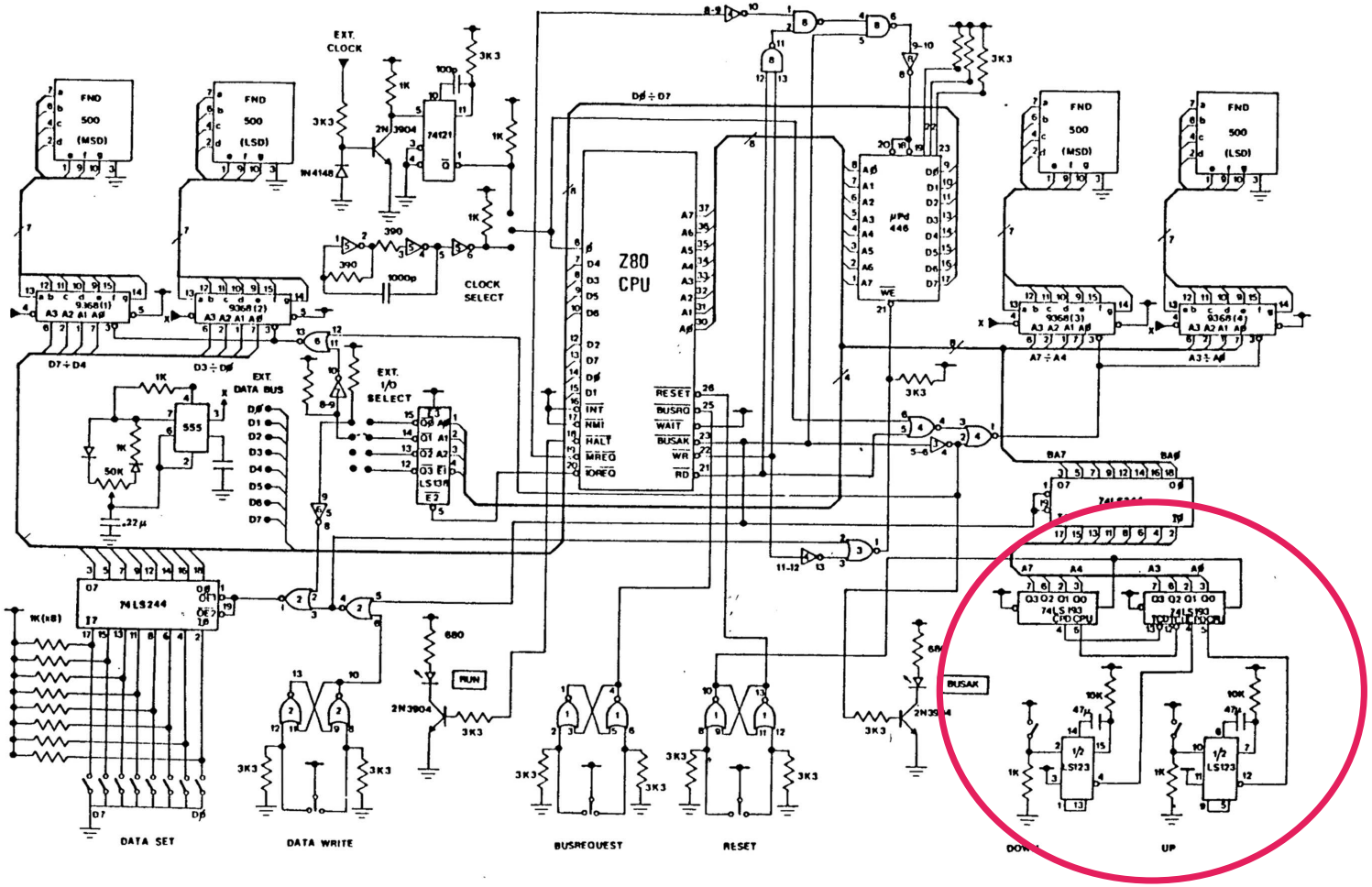


S	R	Q_n	$\overline{Q}_n$
0	0	Q_{n-1}	$\overline{Q}_{n-1}$
1	0	0	1
0	1	1	0
1	1	0	0

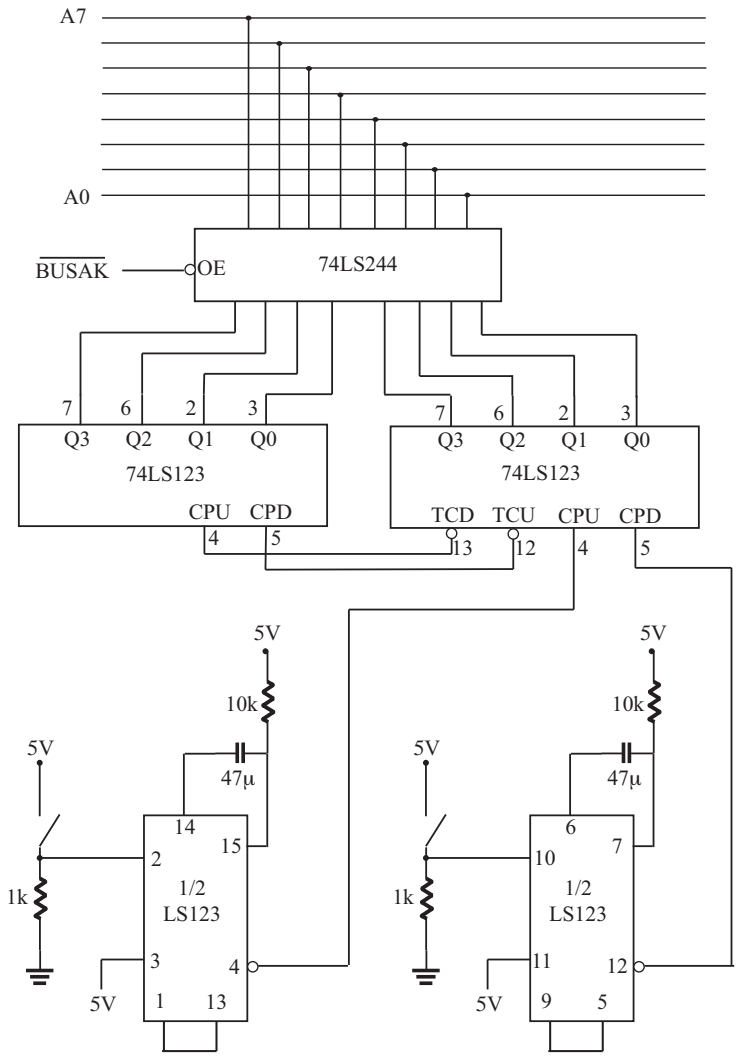


All'uscita $\overline{Q}$ si ha un unico impulso (basso)
i rimbalzi su S e su R non influiscono

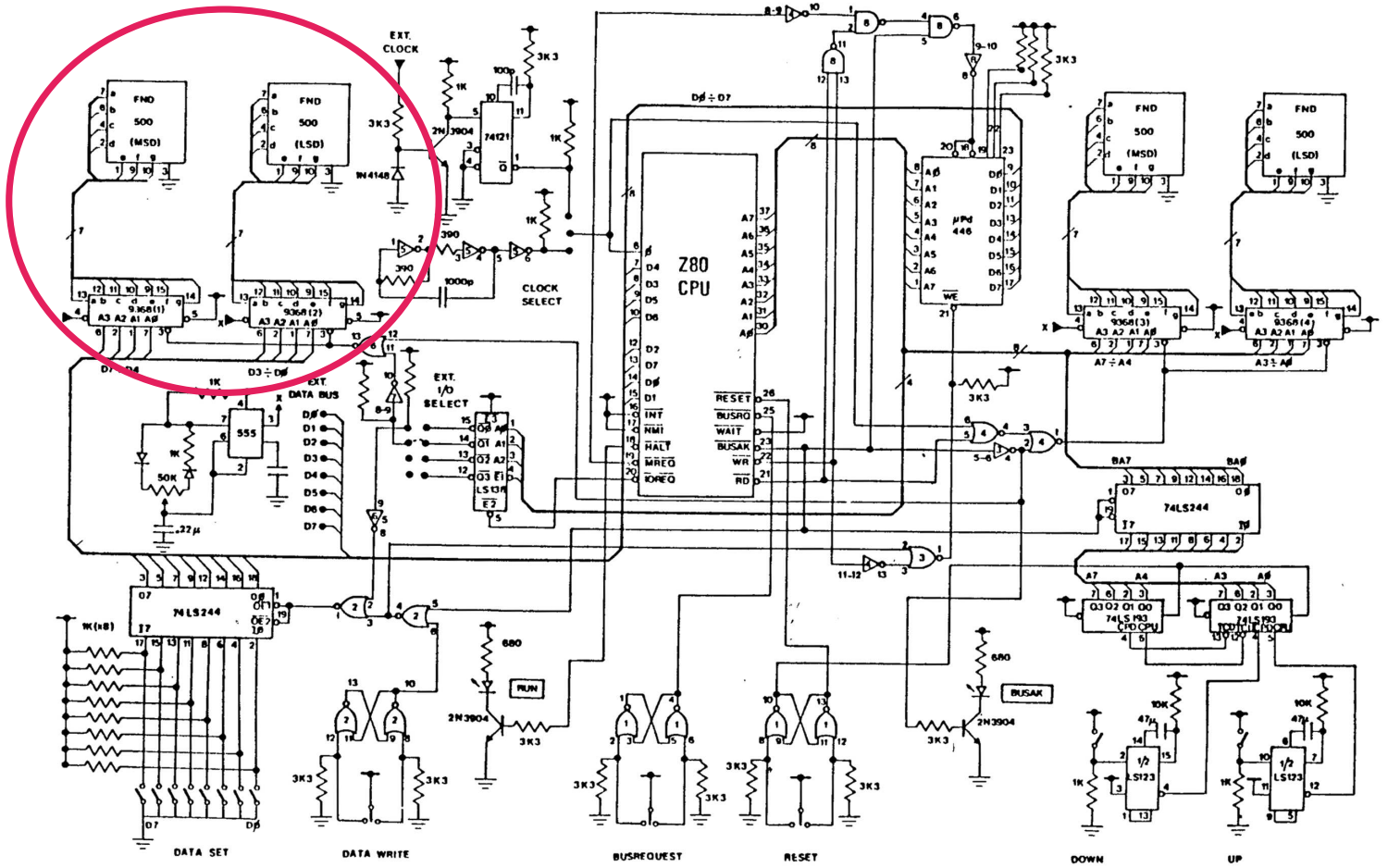
La scheda didattica Z80 - Address Counter



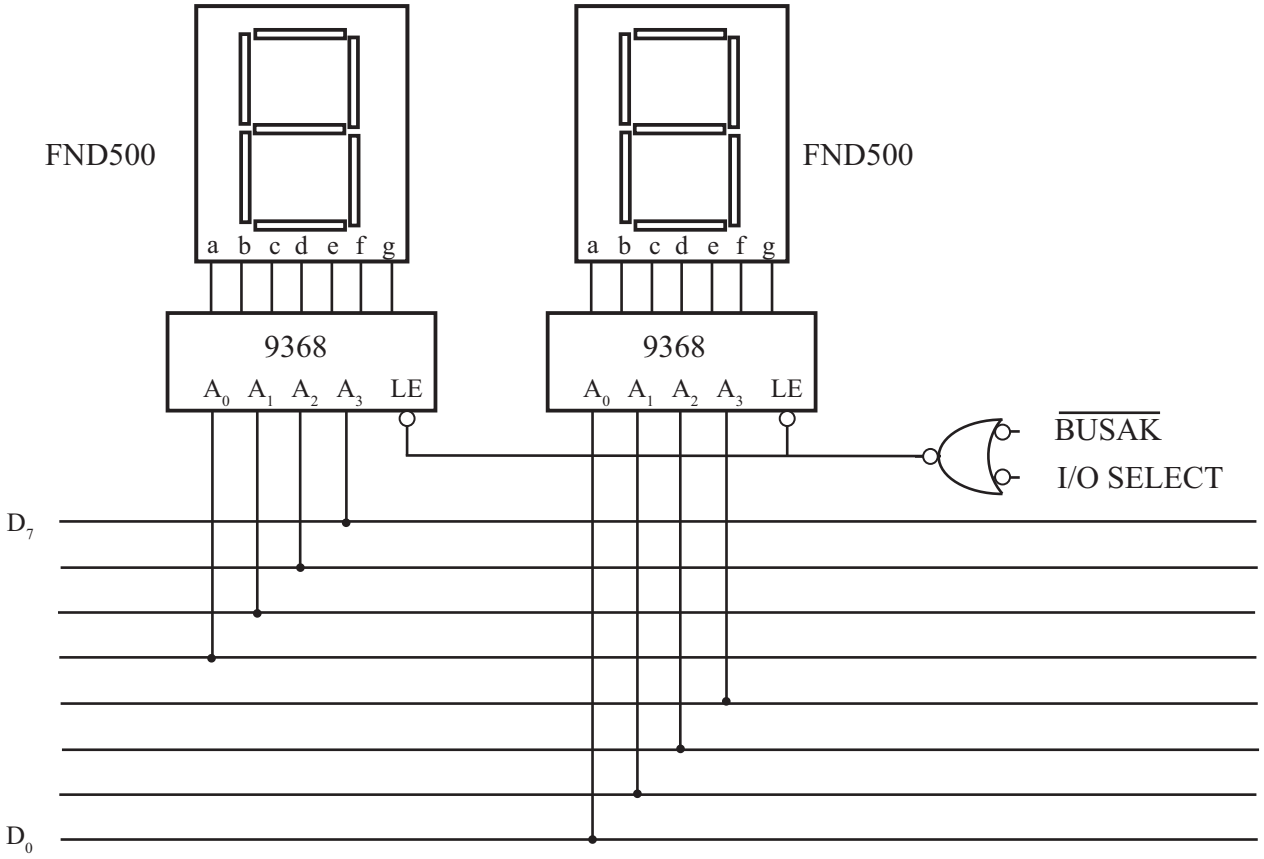
La scheda didattica Z80 - Address Counter



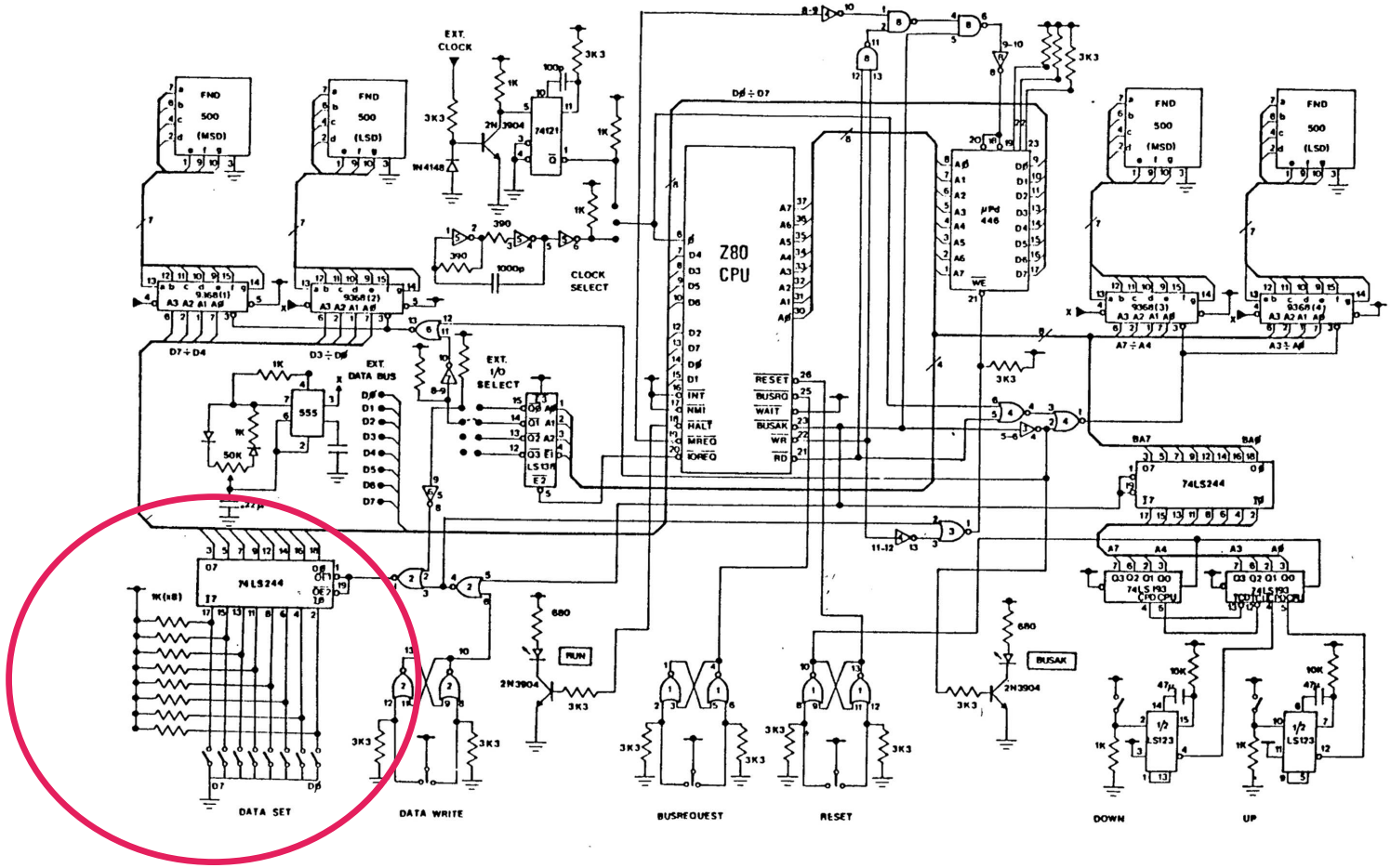
La scheda didattica Z80 - Data Bus Display



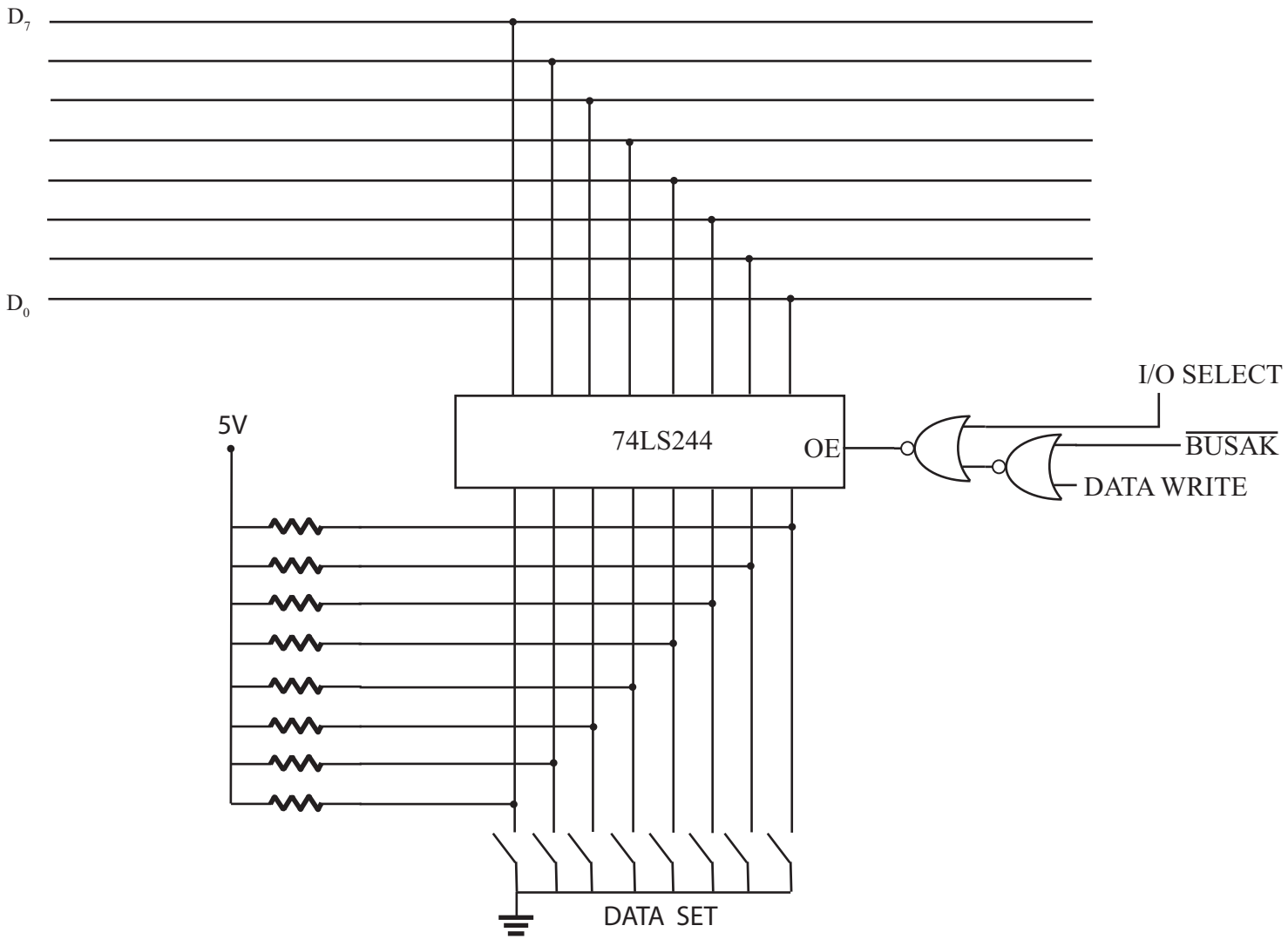
La scheda didattica Z80 - Data bus display



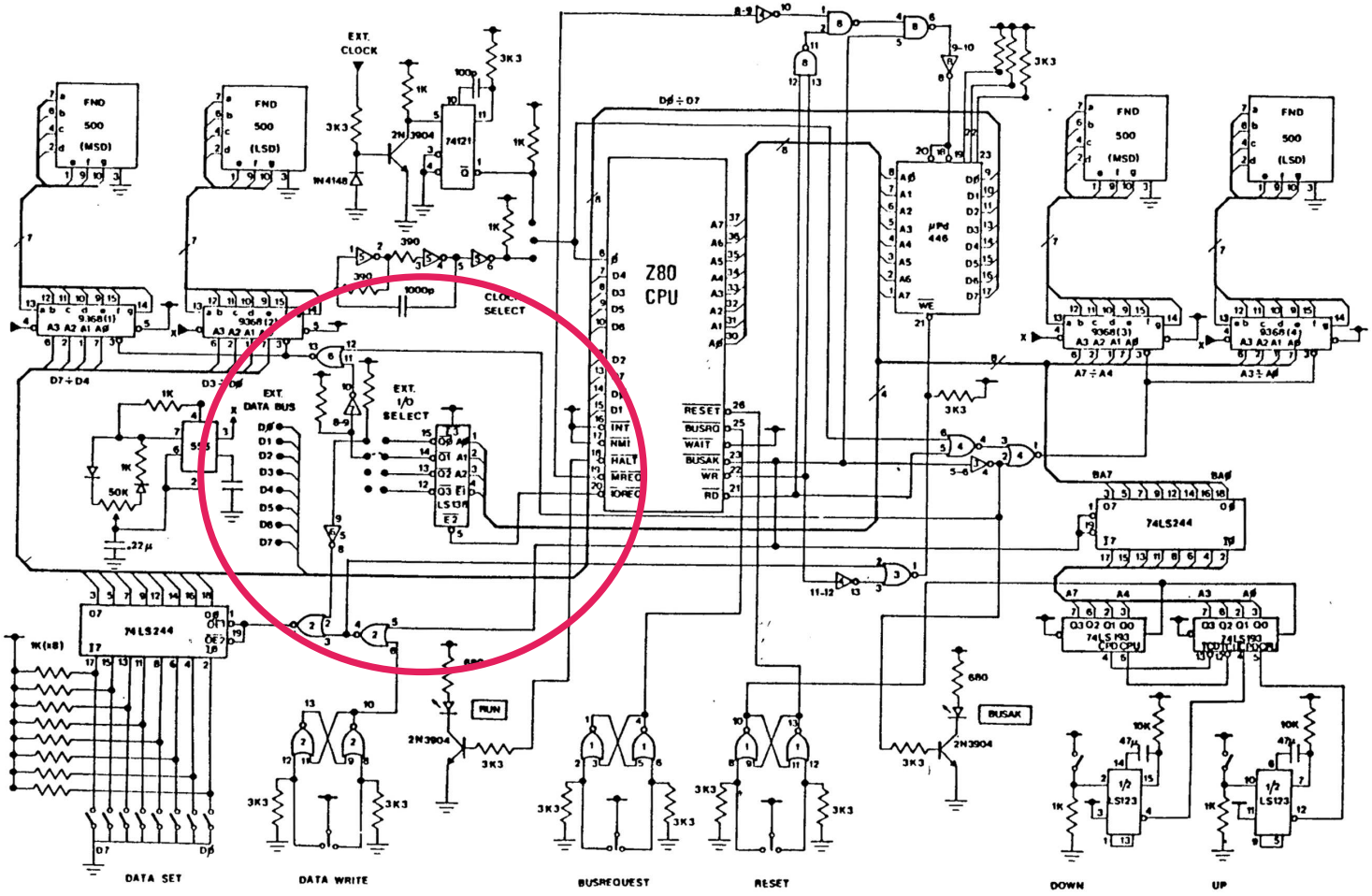
La scheda didattica Z80 - Data Bus Write



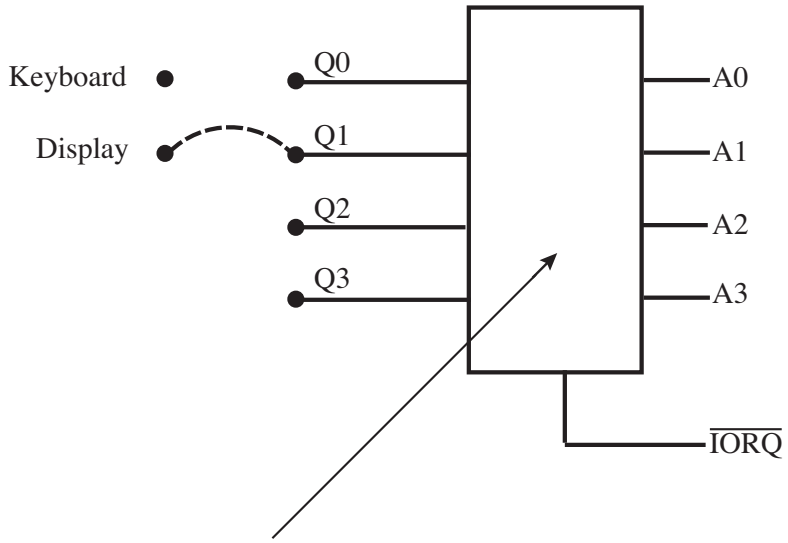
La scheda didattica Z80 - Data bus write



La scheda didattica Z80 - Input/Output Select

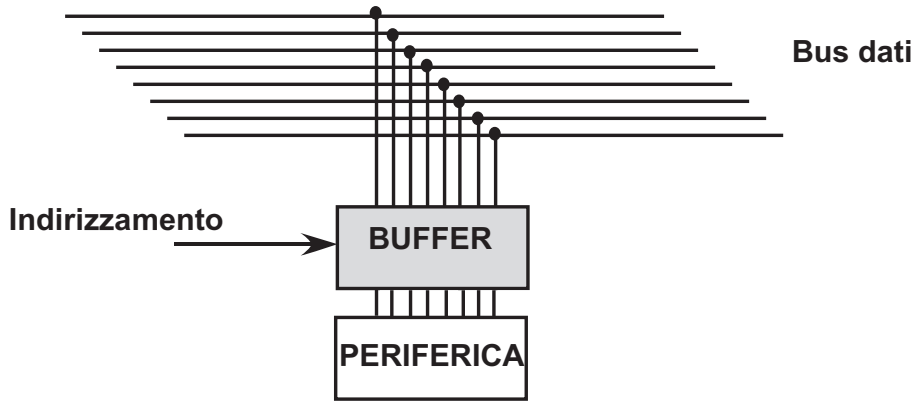


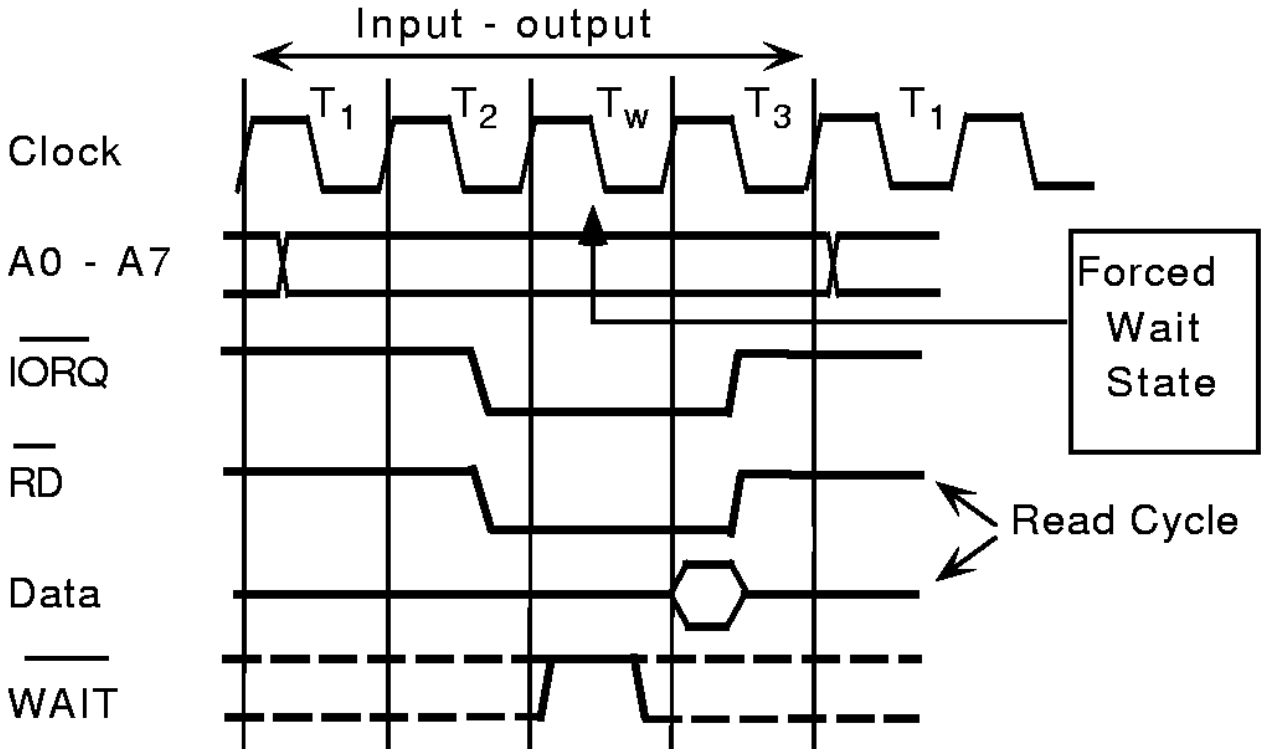
La scheda didattica Z80 - Input/Output Select



DECODER:
Seleziona una delle uscite Q in base
all'indirizzo presente sugli ingressi A.
E' abilitato da $\overline{\text{IORQ}}$

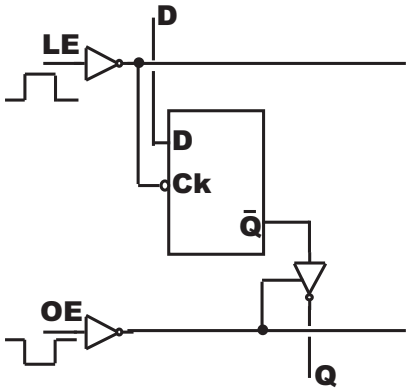
Dispositivi di I/O - Lettura - I





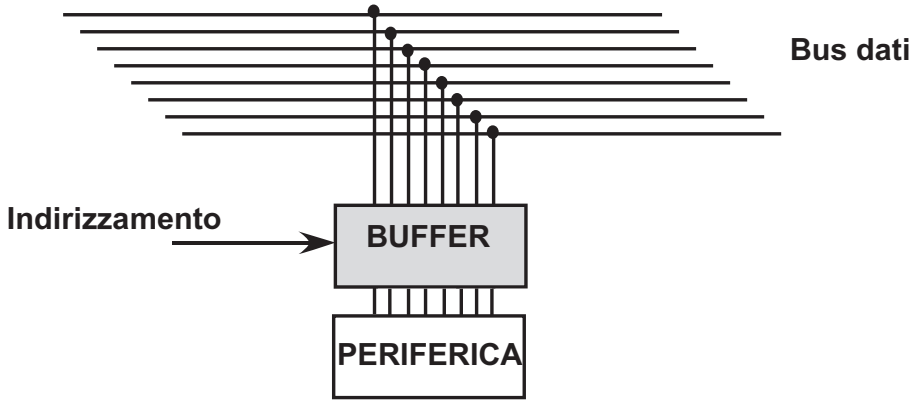
Dispositivi di I/O - Lettura - II

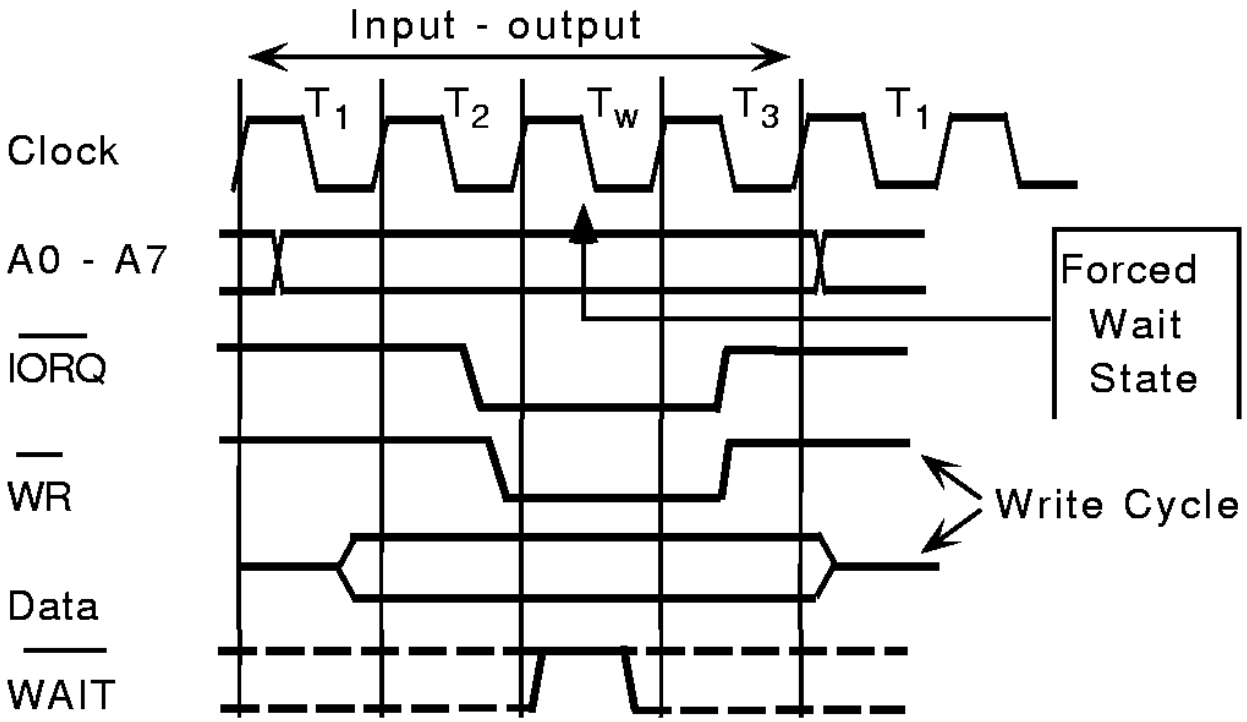
74LS373
OCTAL TRANSPARENT LATCH (3-state output)



D	LE	OE	Q
H	H	L	H
L	H	L	L
x	L	L	x
x	x	H	Hi Z

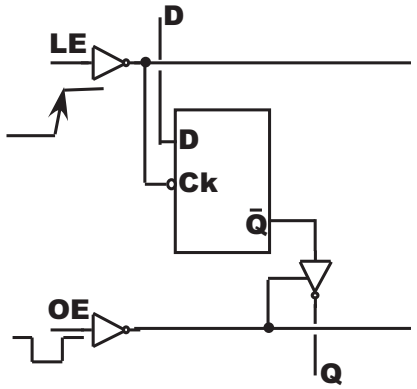
Dispositivi di I/O - Scrittura - I





Dispositivi di I/O - Scrittura - II

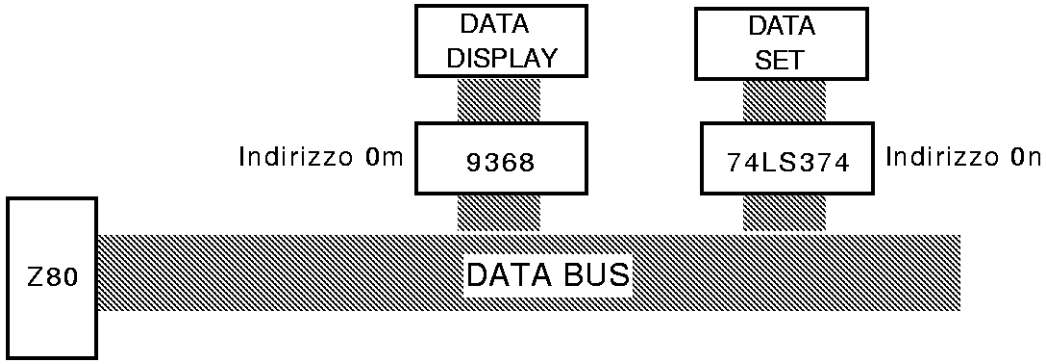
74LS374
OCTAL EDGE-TRIGGERED LATCH (3-state output)



D	LE	OE	Q
H	↑	L	H
L	↑	L	L
x	L	L	x
x	x	H	Hi Z

Z80: I esercitazione - I

Verifica della temporizzazione dei comandi e cicli di istruzione



Scrivere un semplice programma di loop infinito

indirizzo	dato	istruzione	commento
00	00	NOP	nessuna operazione
01	C3	JP0001	salto incondizionato
02	01		all'indirizzo 0001
03	00		

Z80: I esercitazione - II

Per immettere un programma e farlo funzionare si deve eseguire la seguente sequenza.
1) Prendere il controllo del bus mediante l'interruttore BUSREQUEST; si ha il controllo quando é acceso il led verde BUSAK;
2) Mediante i pulsanti UP o DOWN posizionare il contatore degli indirizzi (ADDRESS COUNTER) nella locazione di memoria da cui si desidera far partire il programma; l?indirizzo relativo appare sul visualizzatore degli ADDRESS in forma esadecimale.
3) Impostare (in forma binaria) i byte delle istruzioni del programma da eseguire mediante gli interruttori 0-7;
4) Trasferire nella locazione di memoria indirizzata il dato impostato mediante il pulsante DATA WRITE;
5) Incrementare di uno la posizione dell'ADDRESS COUNTER mediante il pulsante UP;
6) Ripetere la sequenza 3-4-5 fino al termine del programma;
7) Per controllare l'esattezza dei dati impostati si può decrementare l'ADDRESS COUNTER mediante il pulsante DOWN verificando, locazione per locazione, il contenuto della memoria e correggendo gli eventuali errori:
8) Restituire i bus alla CPU mediante l'interruttore BUSREQUEST (il led verde si spegne);
9) Premere momentaneamente il pulsante di RESET; si accende il led rosso di RUN e la CPU cerca la prima istruzione da eseguire in 0000.

Z80: I esercitazione - III

Dopo aver memorizzato il programma farlo eseguire con il clock interno ad 1 MHz.
Sincronizzare esternamente l'oscilloscopio **analogico** con il segnale $\overline{M1}$;
visualizzare su una traccia il clock della CPU (pin 6) e sull'altra traccia i seguenti segnali:

- $\overline{M1}$
- $\overline{MREQ}$
- $\overline{RD}$
- D_0
- A_0
- $\overline{RFSH}$

Disegnare l'andamento, la fase ed i tempi rispetto al clock, delle forme d'onda osservate e verificarne l'accordo con i diagrammi temporali riportati nella descrizione generale dello Z80.
Inserendo invece un clock esterno a pochi Hz é possibile osservare la fase di fetch di ogni istruzione sul visualizzatore degli indirizzi.

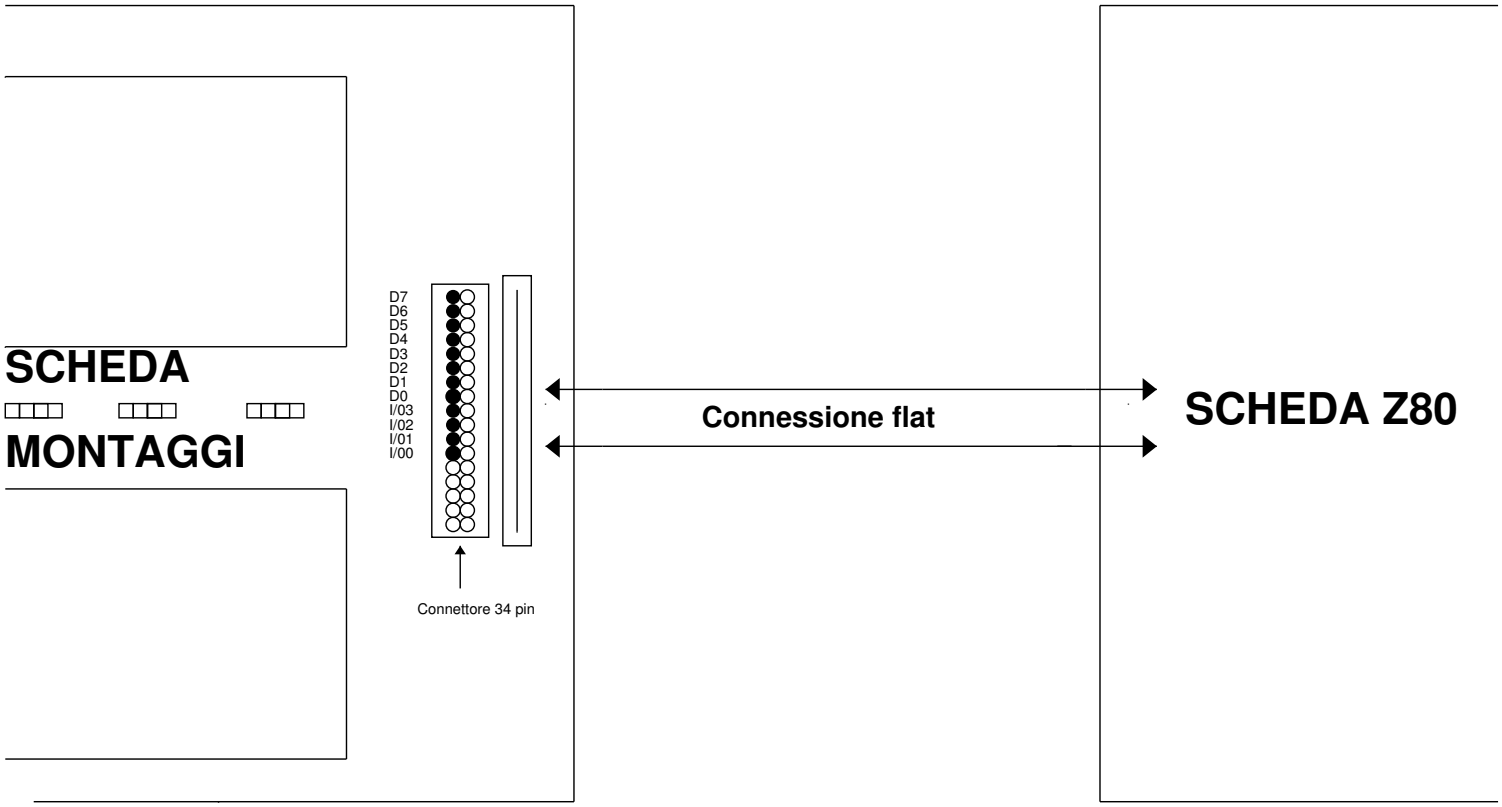
Z80: I esercitazione - IV

Acquisizione di un dato sulla porta di ingresso (interruttori) e visualizzazione sulla porta di uscita (display)

indirizzo	dato	istruzione	commento
00	06	LD B,xx	carica registro B con xx
01	xx		
02	05	DEC B	$B = B - 1$
03	C2	JPNZ 0002	salto condizionato a 0002
04	02		(salta se $B \neq 0$)
05	00		
06	DB	IN A,(0n)	carica accumulatore con il dato
07	0n		sul dispositivo d'ingresso (ind. 0n)
08	D3	OUT(0m),A	presenta contenuto accumulatore su
09	0m		dispositivo d'uscita (ind. 0m)
0A	C3	JP 0000	salto incondizionato a 0000
0B	00		
0C	00		

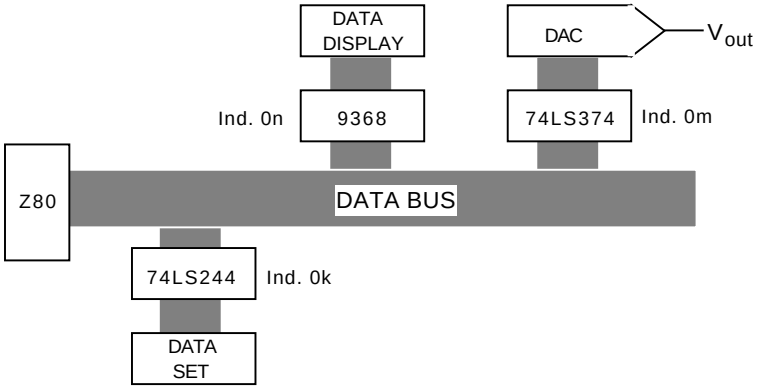
Verificare il flusso del programma con il CLOCK a bassa frequenza, in modo da poter cambiare il dato mentre il programma e' in RUN.

Z80: Connessione con la scheda montaggi

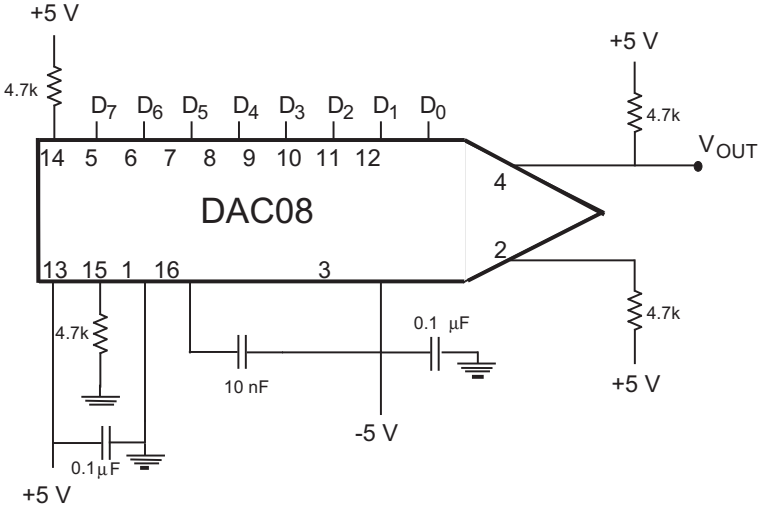


Per le successive esercitazioni utilizzeremo la scheda montaggi opportunamente connessa alla scheda Z80:
Portiamo sulla scheda montaggi il bus dei dati ($D0 - D7$) e le uscite del decodificatore di indirizzi ($I/00 - I/03$) con un cavo flat.

Z80: DAC



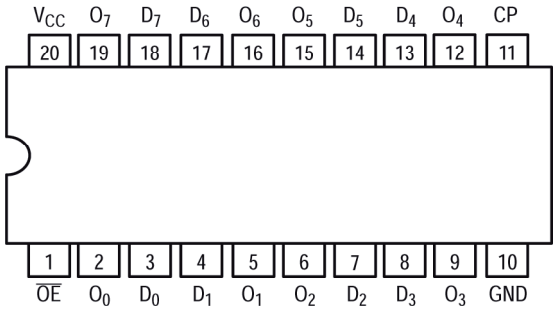
Utilizzeremo un DAC commerciale (DAC800 o DAC08). Con il montaggio mostrato il DAC fornisce:
+5V per ingresso 00000000
0V per ingresso 11111111



74LS374 e DAC08

74LS374

SN74LS374

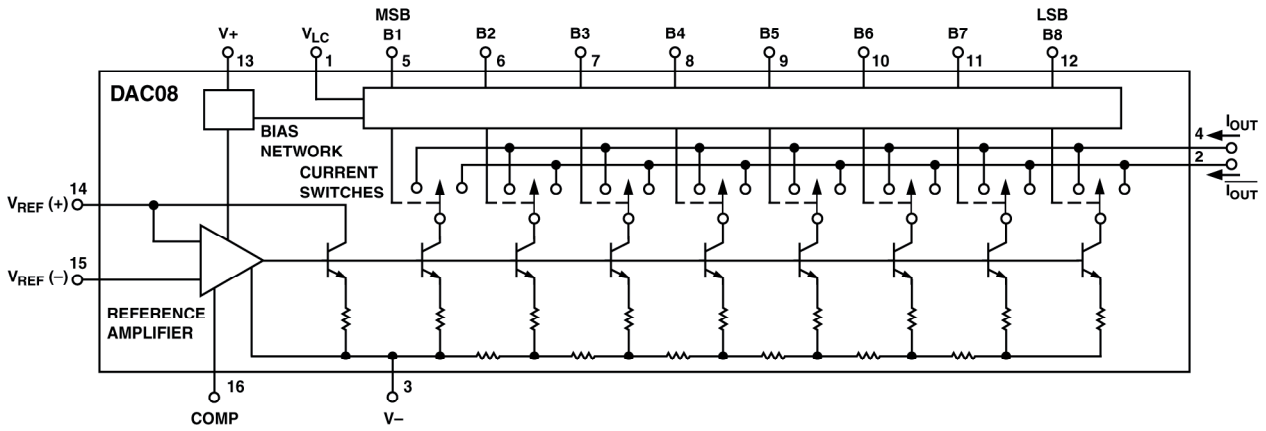


LS374

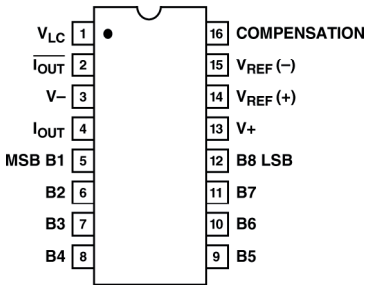
D_n	LE	$\overline{OE}$	O_n
H	⌐	L	H
L	⌐	L	L
X	X	H	Z*

DAC08

FUNCTIONAL BLOCK DIAGRAM



16-Lead Dual-In-Line Package (Q and P Suffix)



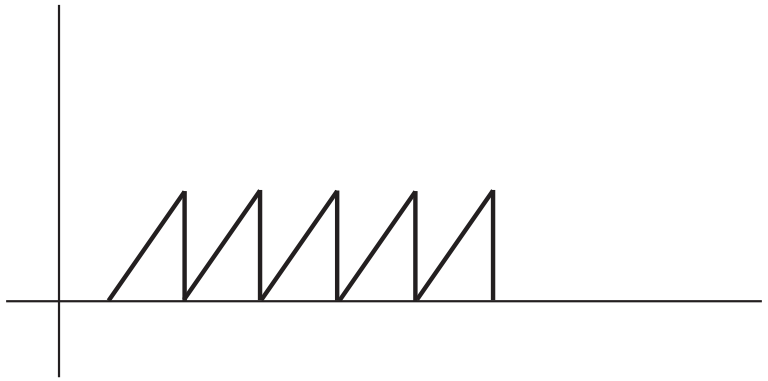
Calibrazione del DAC

indirizzo	dato	label	istruzione	commento
00	DB	start	IN A,(0k)	legge dato da tastiera
01	0k			
02	D3		OUT(0m),A	scrive dato su DAC
03	0m			
04	D3		OUT(0n),A	scrive dato su display
05	0n			
06	C3		JP start	torna a start
07	00			
08	00			

Rampa

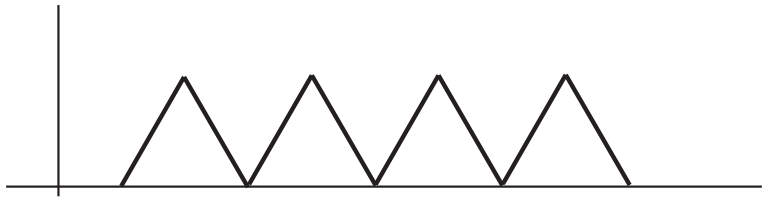
indirizzo	dato	label	istruzione	commento
00	3C	start	INC A	$A = A + 1$
01	D3		OUT(0m),A	scrive su DAC
02	0m			
03	C3		JP start	torna a start
04	00			
05	00			

Per avere una rampa negativa basta sostituire DEC A al posto di INC A.



Dente di sega

indirizzo	dato	label	istruzione	commento
00	D3	up	OUT(0m),A	scrive su DAC
01	0m			
02	3C		INC A	$A = A + 1$
03	C2		JPNZ up	ripete se $A \neq 0$
04	00			
05	00			
06	3D	down	DEC A	$A = A - 1$
07	D3		OUT(0m),A	scrive su DAC
08	0m			
09	C2		JPNZ down	ripete se $A \neq 0$
0A	06			
0B	00			
0C	C3		JP up	ripete il ciclo
0D	00			
0E	00			

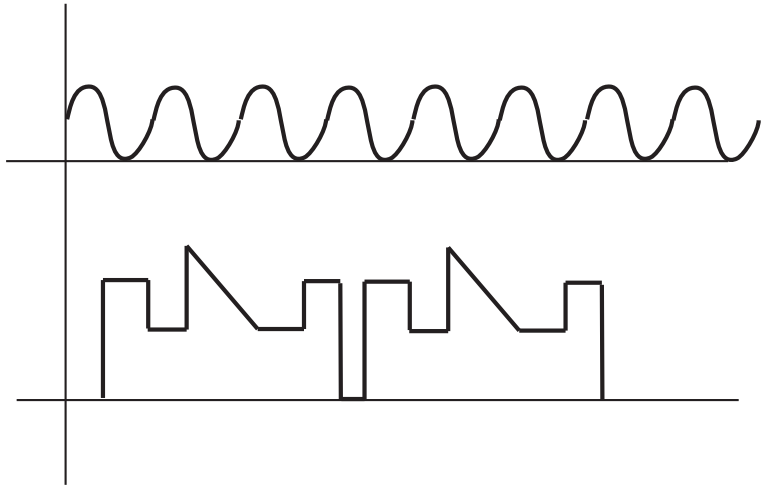


Forma d'onda costruita con dati memorizzati

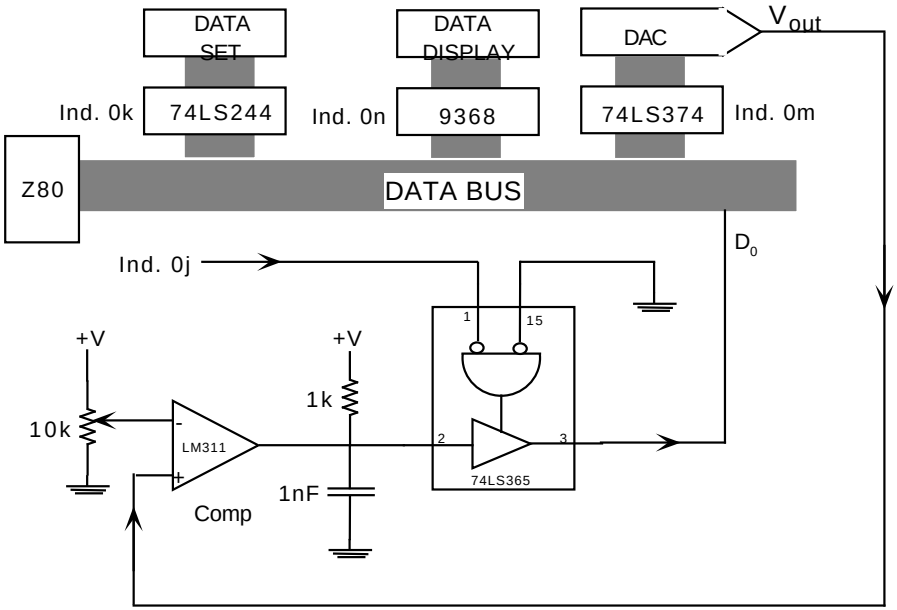
E' possibile costruire una forma d'onda a partire da una tabella contenuta nella memoria della scheda. Inserire in memoria jj dati (che rappresentano i campioni della forma d'onda desiderata) nelle locazioni da kk a kk + jj -1.

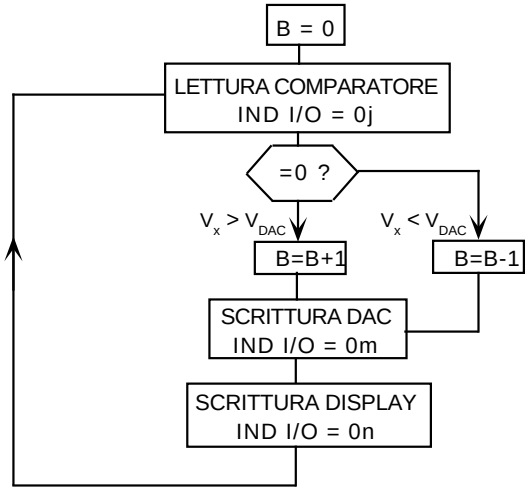
indirizzo	dato	label	istruzione	commento
00	06	start	LD B,jj	Carica Registro B con numero di dati
01	jj			
02	21		LD HL, 00kk	Carica registri HL con primo indirizzo dati
03	kk			
04	00			
05	7E	loop	LD A,(HL)	Copia (HL) in A
06	D3		OUT(0n),A	Scrive sul DAC
07	0n			
08	23		INC HL	Incrementa HL
09	05		DEC B	B = B-1
0A	C2		JPNZ loop	Ripete per il nuovo dato
0B	05			
0C	00			
0D	C3		JP start	Ripete tutto il ciclo
0E	00			
0F	00			

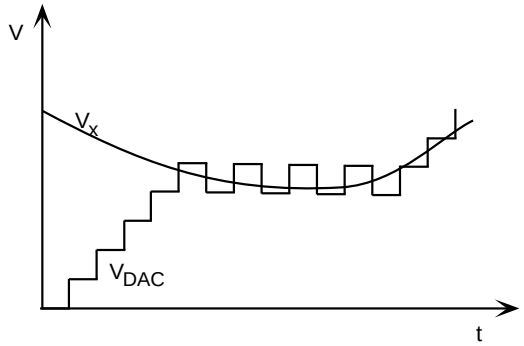
Forma d'onda costruita con dati memorizzati



Z80: ADC Tracking







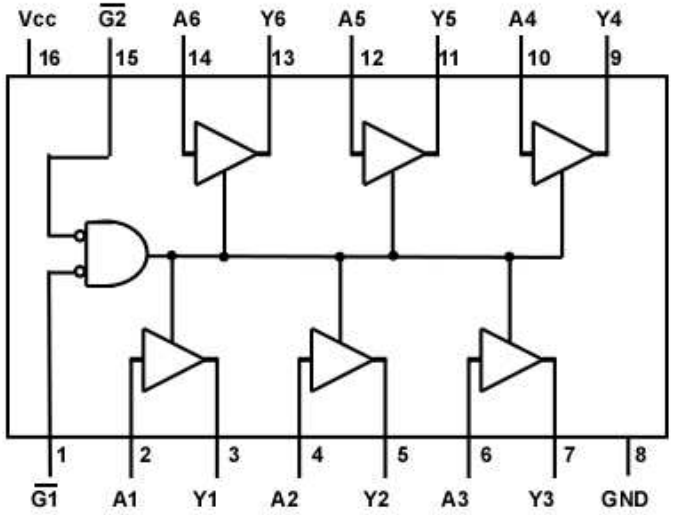
Z80: ADC Tracking - II

L'uscita del buffer viene collegata alla linea D0 del bus dei dati (bit meno significativo).
Dopo aver letto il bus dei dati occorre quindi "mascherare" gli altri bits, il cui contenuto non e' predicibile. Con l'istruzione:

```
Codice  Mnemonico
E6       AND 01
01
```

si effettua AND logico tra il contenuto di A e il numero 00000001. Il risultato viene messo in A.

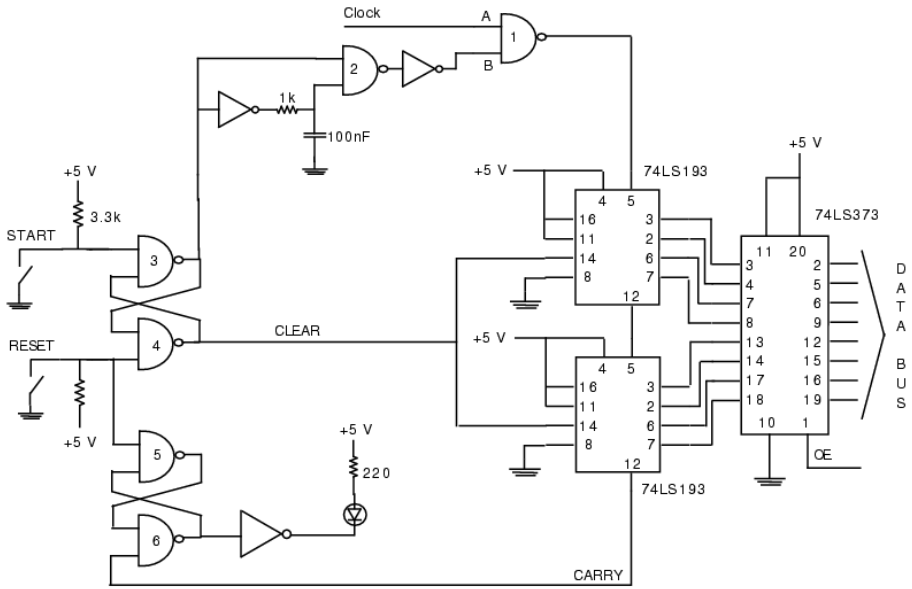
Per leggere l'uscita del comparatore utilizziamo un buffer 74LS365.
Questo integrato contiene 6 porte logiche con uscita 3-state (a noi ne serve una sola).



La tavola della verita' e': .

Input			Output
$\overline{G1}$	$\overline{G2}$	A	Y
H	X	X	Hi Z
X	H	X	Hi Z
L	L	H	H
L	L	L	L

Z80: TDC


Clock
A
B
1k
100nF
+5 V
3.3k
START
RESET
CLEAR
+5 V
220
74LS193
74LS193
74LS373
OE
DATA BUS
CARRY