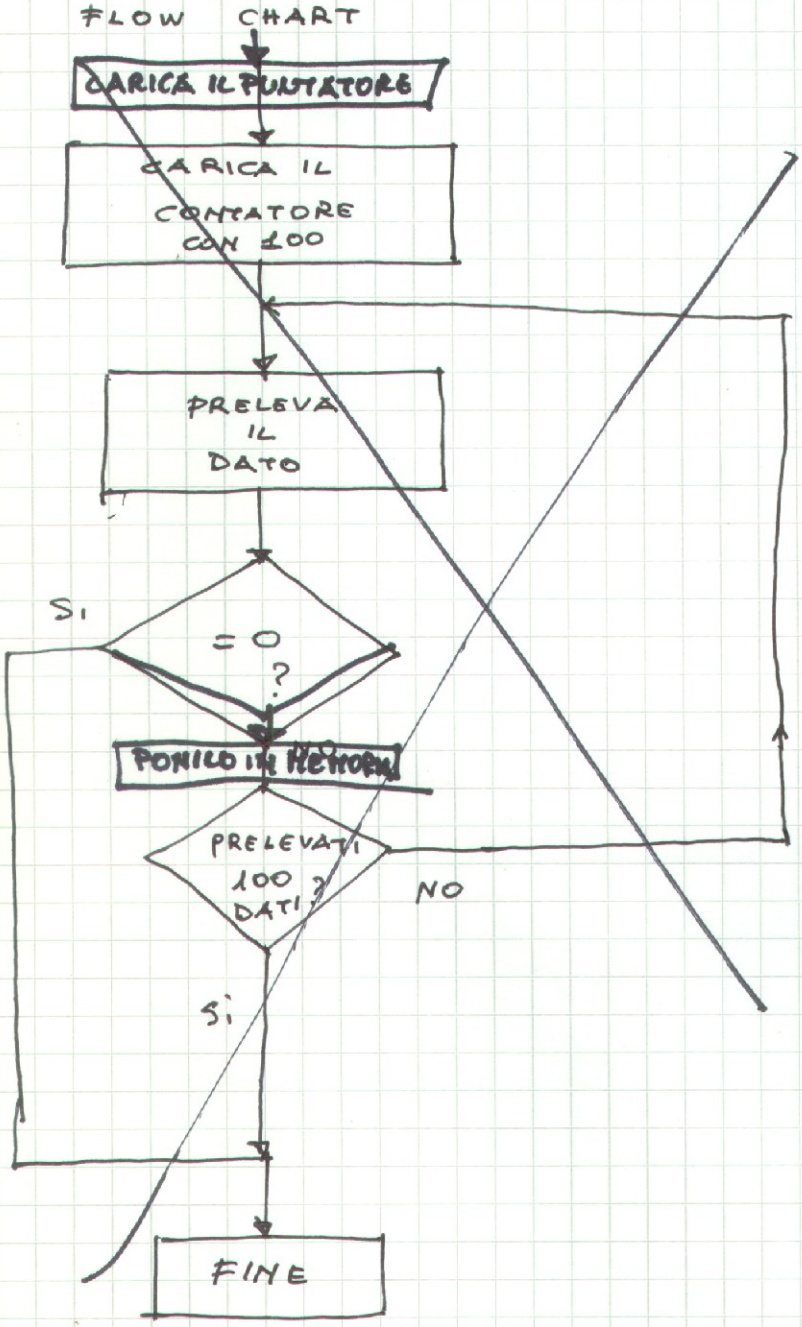
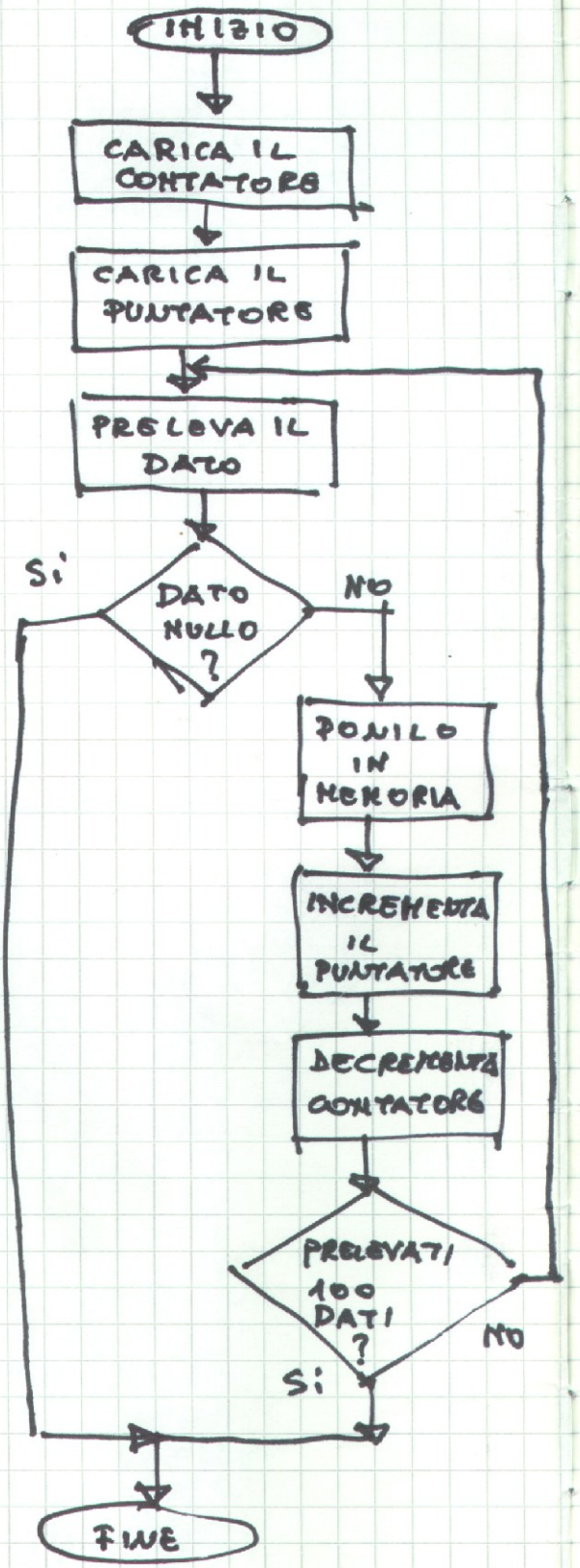
PROGRAMMA 2

Si prelevano 100 dati dalla porta d'indirizzo 20H e si scrivono in memoria.
Se il dato prelevato è 00H si interrompe il programma

FLOW CHART





CODIFICA

```
ORG  LD B, 64H ; carica il contatore B con il numero di dati
                ; da prelevare
     LD HL, 1B00H ; carica il puntatore alla zona di memoria
                ; dove si dovranno inserire i dati
LOOP: IN A, (20H) ; porta nell'accumulatore il dato
     LD (HL), A  ; salvalo in memoria
     CP 00H      ; controlla se il contenuto dell'accumulatore è zero
     JR Z, FINE  ; se si vai alla fine
     INC HL      ; altrimenti incrementa il puntatore
     DJNZ LOOP   ; controlla se B≠0 e preleva un nuovo dato
FINE: HALT
```

```
LD E,L
DEC B
LD H,00H
LD D,00H
LOOP:
ADD HL,DE
DEC B
JP NZ,LOOP
POP DE
POP BC
RET
```

Ex 1

Un µP rileva la temperatura da un sensore tramite un ADC ad 8 bit che presenta gli ingressi SOC (start of conversion) OE (output enable) e l'uscita EOC (end of conversion).
La temperatura viene rilevata ogni 30 minuti mediante interruzione del CTC.
La lettura di un valore $\geq 30_H$ dell'ADC significa che va attivata una sirena.

1. Progettare il circuito di interfacciamento del µP con CTC, ADC e sirena.
2. Scrivere il programma di inizializzazione della scheda.

il programma di gestire delle interruzioni del CTC.
Fare ipotesi sui dettagli non specificati dalla traccia.

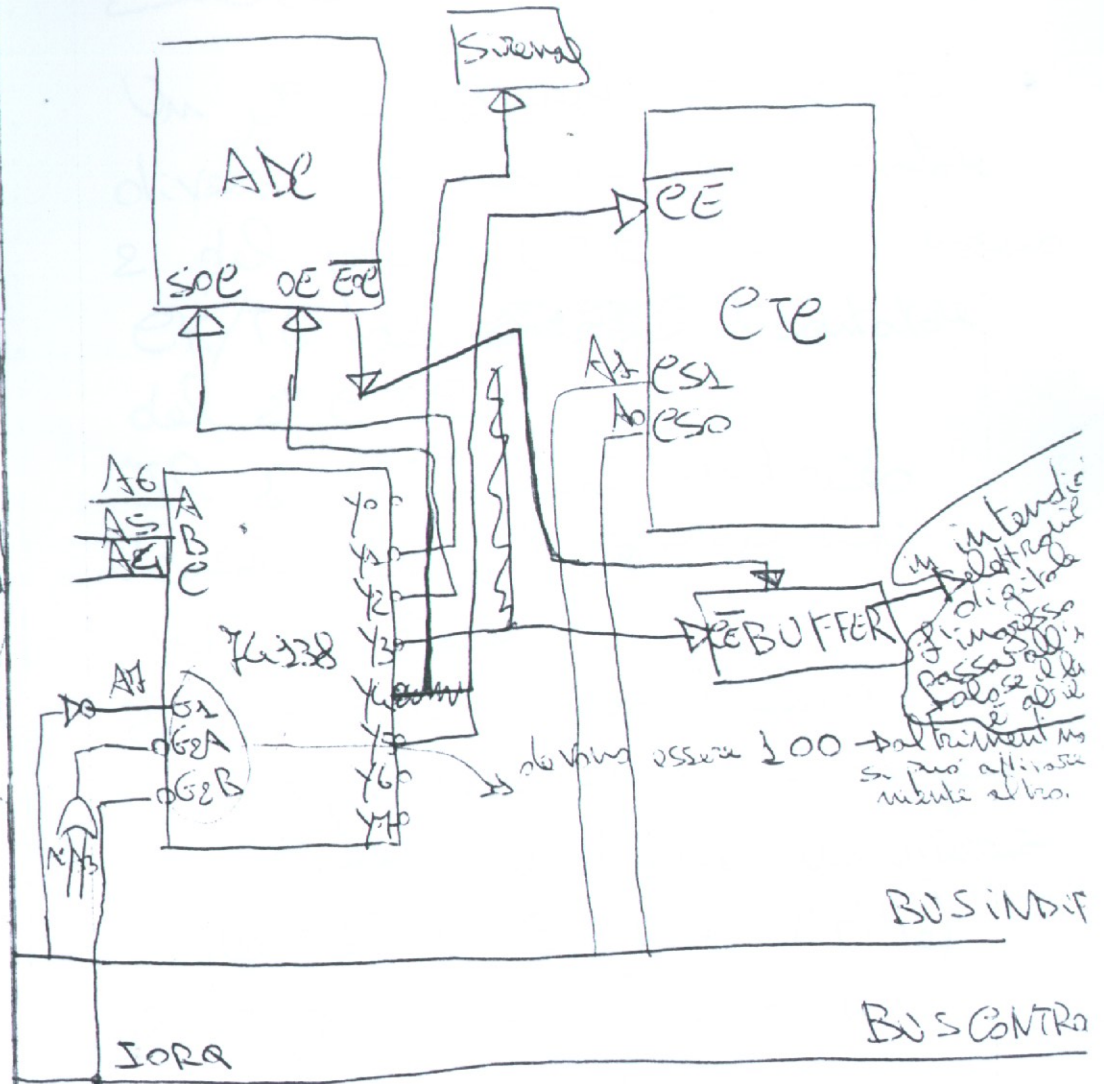
Dobbiamo supporre i questi segnali sono attivi bassi altrimenti dovremmo mettere i not.

SOE		10_H
OE		20_H
EOC		30_H
SIRENA		40_H
CTC		50_H

sono sempre = nella mostra mappa quindi le utilizziamo per i segnali G1 G2A G2B

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	1	0	0	0	0	
0	0	1	0	0	0	0	0	
0	0	1	1	0	0	0	0	
0	1	0	0	0	0	0	0	
0	1	0	1	0	0	0	0	— Dcont0
						0	1	— Dcont1
						1	0	— Dcont2
						1	1	— Dcont3

1 $D0_H$



Se non prendiamo in considerazione A_7 e al posto suo mettiamo 1 resistenza, il circuito non se ne accorge ed è indifferente se è 0 o 1 che viene attivato il decoder. Quindi il

contatore è come se avesse 2 indirizzi 50_H e $D0_H$, non succede niente però basta che non facciamo coincidere $D0_H$ con qualche altro indirizzo.
Questo viene chiamato "decodifica parziale degli indirizzi", e significa che gli indirizzi saranno simili.

Programm. inviolato

```
ORG 0000H
LD HL,2000H
LD (1A0CH),HL
LD A,1AH
LD I,A
LD A,47H
OUT (50H),A
LD A,00H
OUT (50H),A
LD A,67H
```

Esercizio

Un µP è collegato a 2 CTC diversi: l'uscita del contatore 2 del primo CTC va in ingresso CK/TG del [redacted] 1 contatore del 2° CTC.
Il 1 CTC ha indirizzo base 60_H, il 2° CTC ha indirizzo base 80_H.

① Quale è l'intervallo di Tempo massimo che si può misurare mediante un'interrupt del 2° CTC ($f_{CK} = 2MHz$).
② Progettare il circuito di interfacciamento.
③ Programmare i CTC.

deve avere un'uscita 3 state che rimane collegata al BUS, quindi il µP manda OE per farsi scaricare dal bus dati il dato da fare la conversione.

apparecchio contatore = quando il µP ci scrive un numero dentro, incomincia a contare a decrescere ogni ms. e quando è arrivato a 0 manda un'interruzione.

Nel nostro caso questo ADC è collegato ad un sens di temperatura, quando il risultato della conversione è ≥ a 60_H vuol dire che la temperatura

Programma:

Un µP è collegato ad un ADC l'indirizzo per far partire la conversione SOC = 20_H, 60_H l'indirizzo di lettura OE = 30_H all'indirizzo 60_H è collegato invece EOC all'indirizzo 50_H Si trova una periferica un apparecchio che funziona da contatore. ADC è collegato un sens di temper quando il risultato della conversione è ≥ a 60_H vuol dire che la temperatura letta è troppo elevata, deve suonare una sirena che si trova all'indirizzo 60_H *

Per leggere il risultato della conversione il µP va 30_H e leggi i

risultato che OG ha dato in uscita.

* Questa scheda deve controllare la temperatura ogni 500ms e deve avvenire con la tecnica delle interruzioni. Il uP fa partire il contatore quando arriva l'interruzione dal contatore va a vedere il risultato della conversione. Il programma che gestisce l'interruzione si occupa del risultato della conversione, se deve o meno attivare la sirena, questo è a 3000H, la tabella dei vettori delle interruzioni è a 1A00H

Programma di inizializzazione

della scheda cioè quando accende la scheda deve mettere tutto a posto, cioè il uP deve seguire appena si accende.

ORG 0000H direttiva all'assemblatore non è un istruzione che deve seguire il uP ma è a direttiva che traduce in linguaggio macchina dall'indirizzo 0000H.

1A00	00
1A01	30

Nella tabella dei vettori si deve essere scritto l'indirizzo del programma per gestire l'interruzione.

Quindi la 1ª cosa da fare

andiamo a scrivere nella tabella dei vettori l'indirizzo 3000H a partire da 1A00

ORG 0000H
LD HL, 3000H
LD (1A00H), HL → mette la parte bassa di 1A00 e quella alta a 1A01. Se ci fossero state + cause di interruzioni avremmo dovuto fare tante volte questa operazione fino ad aver scritto tutti gli indirizzi dove ci stanno i programmi per gestire quelle cause di inter.

Ricordiamo che arriviamo a 1A00 perché 00H ci viene dato dalla periferica, 1AH ci viene dato dal registro I, nel registro I, 1AH viene scritto dal μP stesso.
Quindi andiamo a scrivere 1AH in I.

ORG 0000H
LD HL, 3000H
LD (1A00H), HL
LD A, 1AH
LD I, A

La periferica restituisce 00H perché sono io a dire alla periferica di darmi 00H.

andiamo a scrivere nella tabella dei vettori l'indirizzo 3000_H a partire da $1A00$

ORG 0000_H
LD HL, 3000_H
LD ($1A00_H$), HL → mette la parte bassa di $1A00$ e quella alta di $1A01$.

Se ci fossero state + cause di interruzioni avremmo dovuto fare tante volte questa operazione fino ad aver scritto tutti gli indirizzi dove ci stanno le i programmi per gestire quella causa di inter.

Ricordiamo che arriviamo a $1A00$ perché 00_H ci viene dato dalla periferica, $1A_H$ ci viene dato dal registro I, nel registro I, $1A_H$ viene scritto dal μP stesso.
Quindi andiamo a scrivere $1A_H$ in I.

ORG 0000_H
LD HL 3000_H
LD ($1A00_H$), HL
LD A, $1A_H$
LD I, A

La periferica restituisce 00_H perché sono io a dire alla periferica di darmi 00_H.

Programma di inizializzazione

```
ORG 0000H
LD HL, 3009H
LD (1A00H), HL
LD A, 1AH
LD I, A
LD A, 00H
LD (50H), A
LD A, 66H
LD (50H), A
IM 2
EI
ET: JP ET
```

partirà il contatore dato che all'inizio c'è 00 e quando arriva a 0 deve generare un'interruzione

Noi vogliamo gestire l'interruzione col modo 2, però all'inizio tutti i registri sono a 0 quindi anche IM è a 0 cioè significa che gestirebbe il problema delle interruzioni con il modo 0 quindi siamo noi a dire di gestire le interruzioni col modo 2.

IM 2

Questo vale anche per le interruzioni, che all'inizio sono disabilitate quindi noi dobbiamo abilitarle. Da questo momento in poi il µP starà in attesa di ricevere un'interruzione. Quindi possiamo mettere un'istruzione di salto indietro.

gramma che gestisce l'interruzione che si trova all'indirizzo 3000H quindi usiamo la direttiva:

ORG 3000H → cioè non è un'istruzione ma serve all'assemblatore per depositare le istruzioni da partire proprio da 3000H.

Per far partire la conversione analogico-digitale basta scrivere qualcosa all'indirizzo 20H.

OUT (20H), A

Per controllare lo stato di end of conversion EOC si deve leggere all'indirizzo 40H

ET1: IN A, (40H)

quindi nell'accumulatore ci sarà scritto il contenuto del bus dati in modo particolare il bit D0 conterrà il contenuto di end of conversion EOC, adesso dobbiamo testarlo.

Bit 0, A

se il bit testato è a D0 significa che è finita la conversione perché il segnale EOC è attivo dato che è attivo basso. Adesso per far aspettare il μP: JP NZ, ET1.

ETS va su IN a non su Bit perché altrimenti controllerò sempre lo stesso bit, quindi io devo prima andare a leggere il dato e poi vado a ~~[illegible]~~ controllare il mio bit.

Se andiamo avanti significa che l'ADC contiene il segnale in digitale.

IN A, (30H)

significa che quando sarà attivo $\overline{OE}$ l'ADC manderà il dato sul BUS DATI che andrà poi scritto nell'accumulatore.

CP 60H → devo controllare se l'accumulatore è ≥ di 60H questo modificherà il RDF.

se il flag di segno è 0.

```
JP M, ETS
OUT (60H), A
ETS: LD A, 64H
     OUT (50H), A
     RETI
```

a questo punto ho riattivato il contatore e quindi RETI.

```
LD E,L
DEC B
LD H,00H
LD D,00H
LOOP:
ADD HL,DE
DEC B
JP NZ,LOOP
POP DE
POP BC
RET
```

Ex 1

Un µP rileva la temperatura da un sensore tramite un ADC ad 8 bit che presenta gli ingressi SOC (start of conversion) OE (output enable) e l'uscita EOC (end of conversion).
La temperatura viene rilevata ogni 30 minuti mediante interruzione del CTC.
La lettura di un valore $\geq 30_H$ dell'ADC significa che va attivata una sirena.

① Progettare il circuito di interfacciamento del µP con CTC, ADC e sirena.

② Scrivere il programma di inizializzazione della scheda.

programma

Il μP legge continuamente 1 byte inviato dal 1 porta di indirizzo (20H). In questo byte il bit da 0 a 3 indicano il valore delle uscite dei sensori i quali quando sono ad 1 indicano che vi sono stato un'intrusione nel locale control dal μP.

I restanti bit vengono dai sensori antincendio, quando sono a 1 indicano che c'è pericolo di incendio.

Al μP è collegato 1 sirena il μP vede la sirena come 1 porta di indirizzo 30H.

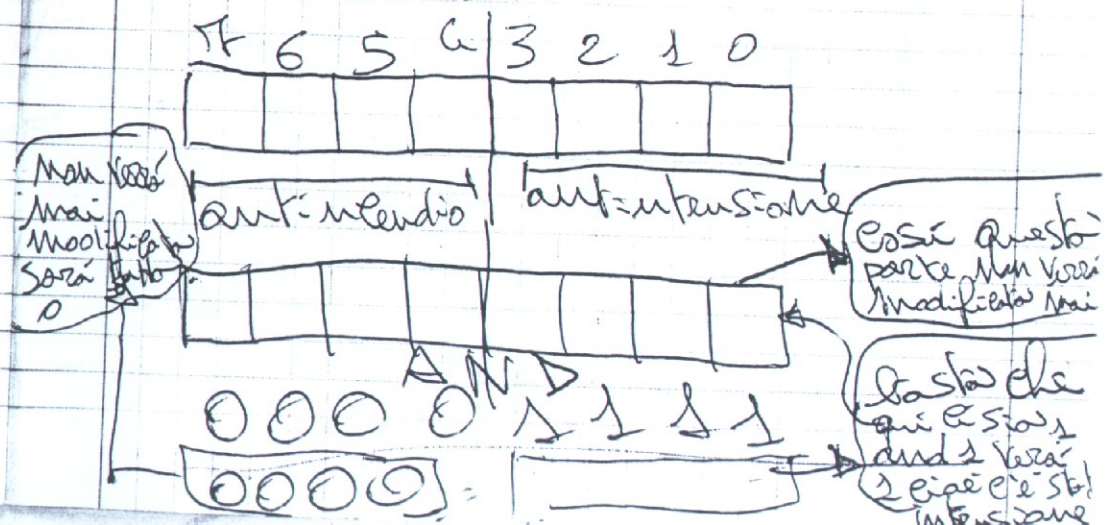
Per azionare la sirena il μP

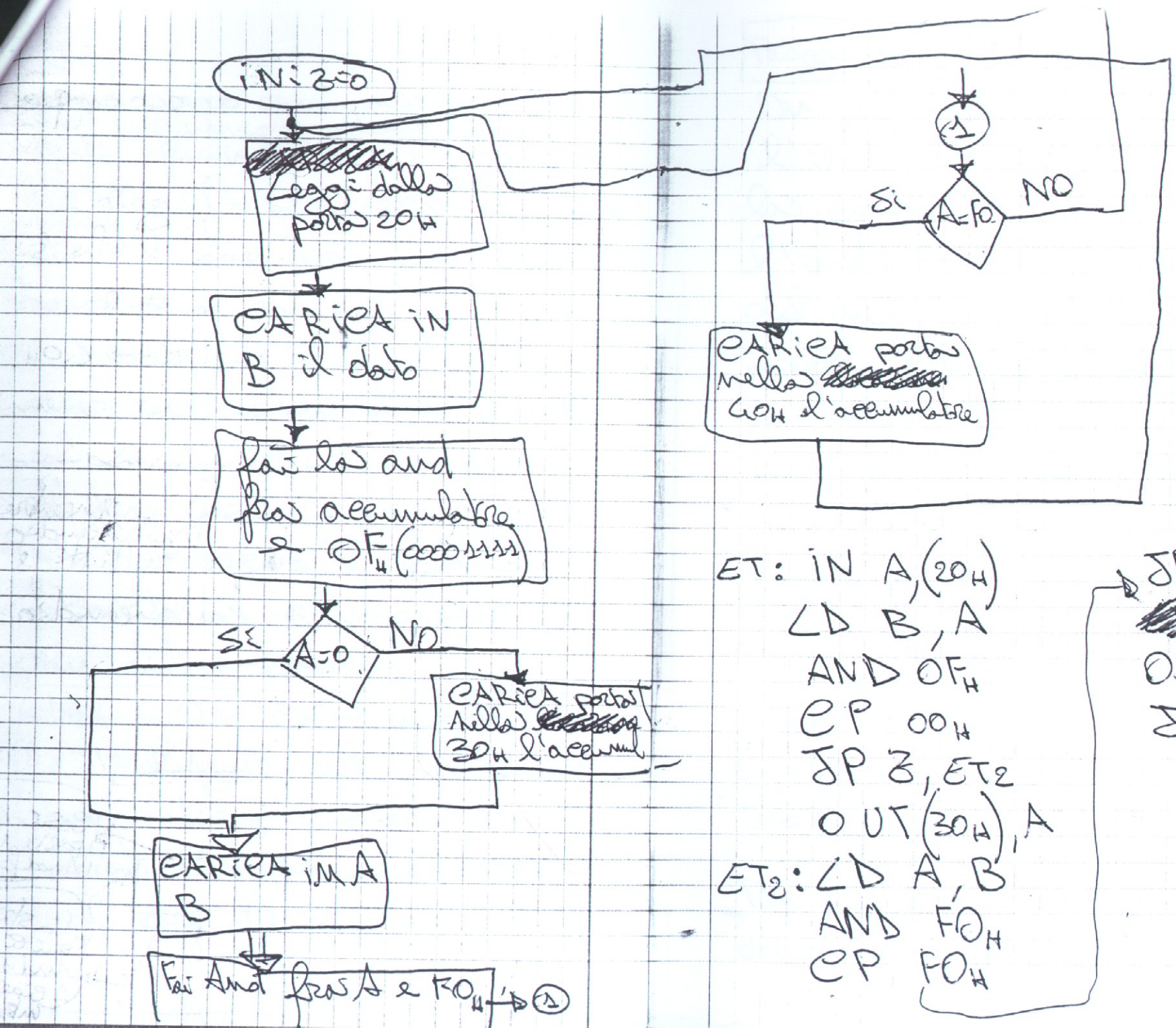
sono collegati i sensori.

deve scrivere 1 dato qualunque all'indirizzo 30H.

Infine al μP è collegato 1 sistema antincendio che si aziona scrivendo 1 dato qualsiasi all'indirizzo 60H.

Si vuole azionare la sirena se almeno 1 dei sensori antintrusione è a 1 mentre si vuole azionare l'antincendio se tutti e quattro i valori antincendio sono a 1.






```
ET: IN A,(20H)
    LD B,A
    AND 0FH
    CP 00H
    JP Z,ET2
    OUT (30H),A
ET2: LD A,B
    AND F0H
    CP F0H
    JP NZ,ET
    OUT (60H),A
    JP ET
```

Programma

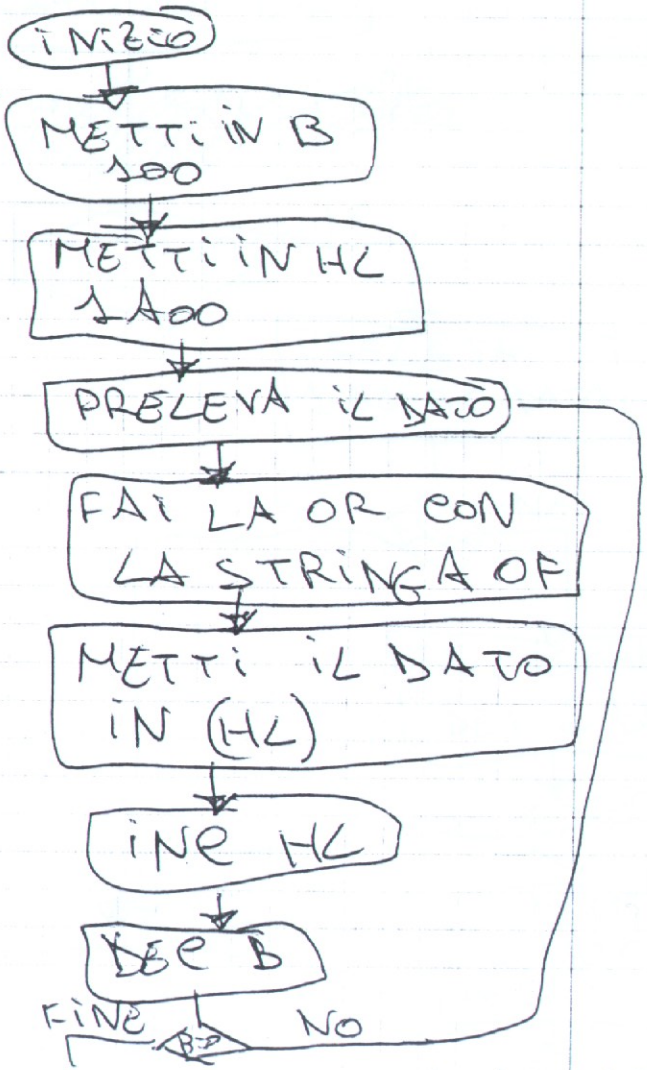
A partire dalla locazione 8000_H ci sono 100 dati esadecimali dei quali occupa 1 byte. Si vuole fare il quadrato di ciascuno che occupa 2 byte e devono essere messi nell'indirizzo 8200_H, quindi bisogna usare il sottoprogramma Quad.

```
    LD B, 64H
    LD DE, 8000H
    LD IX, 8200H
TE: LD A, (DE)
    LD L, A
    CALL QUAD
    LD (IX+0), L
    LD (IX+1), H
    INC DE
    INC IX
    INC IX
    DEC B
    JP NZ, ETE
    HALT
```

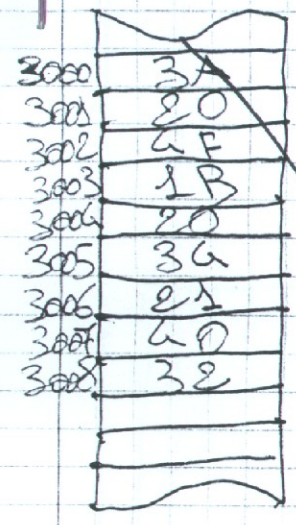
Programma

100 dati a partire dalla locazione 1A00 dobbiamo cambiare a ogni dato il nibble inferiore e deve essere 1111.

```
    LD B, 64H
    LD HL, 1A00
ET: LD A, (HL)
    OR 0F
    LD (HL), A
    INC HL
    DEC B
    JPNZ ET
    HALT
```



Es



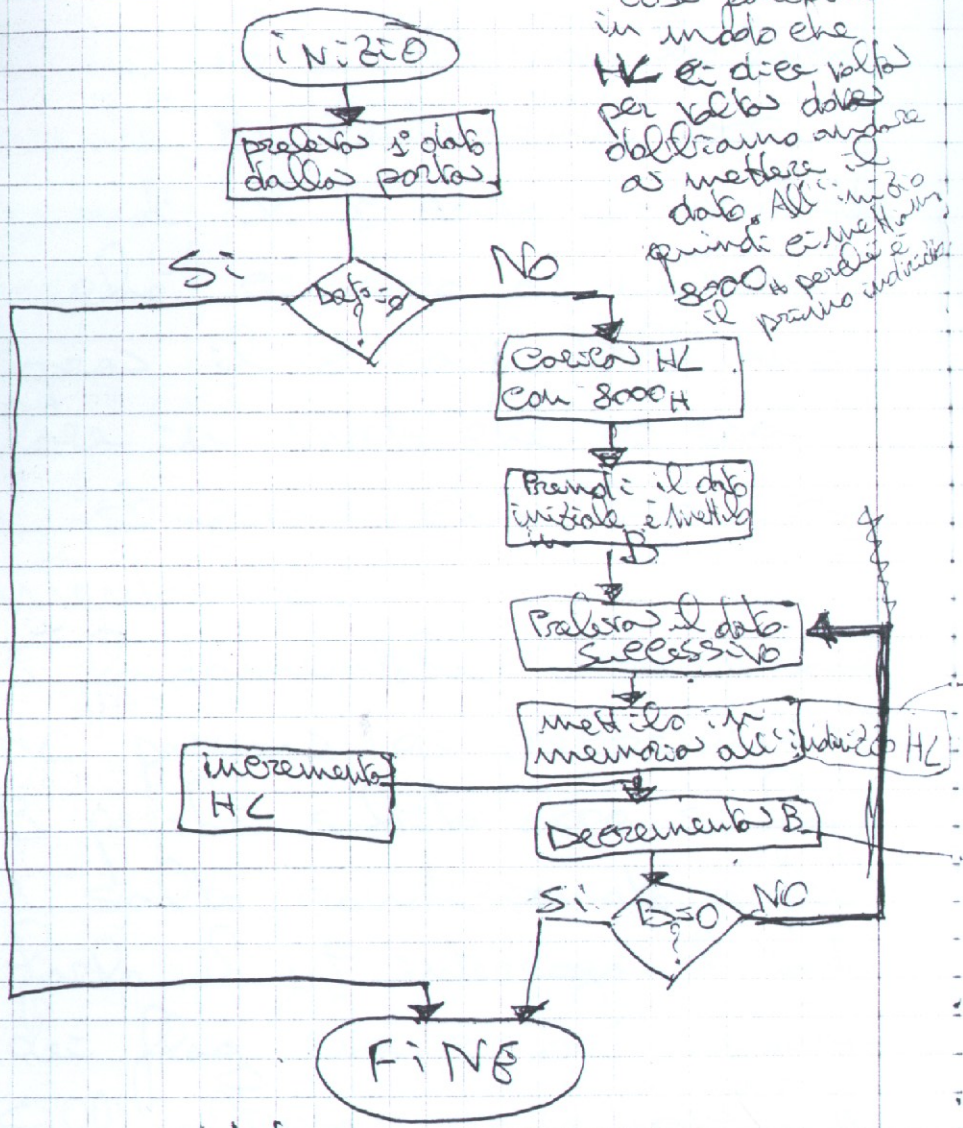
A 31
B 2A
C 10
D 32
E 12
H 30
L 00
IX 3002
IY 3000

1) ADD A, 20H
2) ADD A, B
3) ADD A, (HL)
4) ADD A, (IX+2)
5) ADD A, (IY+3)
6) ADD A, (IX+5)
7) ADD A, (IY+1)
8) ADD HL, BC
9) LD A, 20H
10) LD A, B
11) LD A, (HL)
12) LD A, (IX+2)
13) LD A, (IY+3)
14) LD A, (IX+5)
15) LD A, (IY+1)
16) LD HL, BC

Si devono prelevare dati dal unità I/O (è l'unità che invia il dato da prelevare) porta di indirizzo 50H e vanno immagazzinati in memoria a partire dall'indirizzo 8000H. Il primo dato che viene prelevato dalla porta 50H (non è ancora immag.) indica quanti sono i dati da prelevare successivamente. Per cui se è 0 significa che non ci sono dati da prelevare e quindi dobbiamo chiudere il programma.

Diagramma di flusso

Diagramma di flusso.



così facciamo in modo che HL ci dice volta per volta dove dobbiamo andare a mettere il dato. All'inizio quindi ci mette il dato 8000H perché è il primo indirizzo

B = è un contatore è un registro dove andiamo a mettere il numero dei dati da prelevare che ogni volta che preleviamo 1 dato scala preleva un altro e scala e così via. Così arrivati a 0 significa che non c'è nessun altro dato e il programma deve finire.

poiché ho fatto in modo che HL contenesse 8000H il primo dato verrà messo lì.

x sapere se ho finito devo prima decrementare x poi vedere se B=0 o meno. Se B=0 il programma finisce se invece è ≠0 devo tornare a prelevare il dato successivo.

Però in questo diagramma c'è stesso
Perché io su HL non ho fatto
niente HL conterrebbe 8000H
quindi il 1 dato si perderebbe
perché ci vado a mettere
sopra il 2° dato. Dobbiamo
fare in modo che ci sia
1 istruzione che decrementa
mento HL.
È fondamentale incrementare
HL prima di decrementare
B perché subito dopo si fa
1 salto condizionato e per
farlo il µP interroga il registro
dei flag che ci dà informaz
sull'ultima operazione fatta
quindi se incrementa HL
lo mettiamo dopo di decremento

l'informaz
B del registro dei flag si riferisce
all'operazione incrementa HL.
Per questo è fondamentale,
perché quando facciamo il
salto condizionato B=0 il
registro dei flag dato che si
riferisce all'operazione sbagliata
cioè incrementa HL ci dà
un informazione sbagliata
in quanto non darà mai B=0
perché si riferisce a HL che
non sarà mai 0 perché si
incrementa sempre.

Adesso dobbiamo codificare
il programma. Nello 080 la
lettura di un dato dalla porta

Si userà l'istruzione IN, quando si preleva dati dai registri o dalla memoria si userà l'istruzione LD (load) se dobbiamo scrivere in una porta si userà OUT.

IN A,(50_H)

preleva il dato e mettilo nel registro A all'indirizzo (50_H) → indirizzo della porta.

Se il dato volessimo caricarlo in H dobbiamo per forza prima caricare l'indirizzo nel registro C e poi andiamo a prelevare il dato lo cerchiamo in H all'indirizzo che ci verrà detto da C. perché non possiamo mettere direttamente l'indirizzo

LD C, 50_H
IN H, (C)

IN A, (50_H).

In questo caso specifico ho necessità di sapere se c'è il dato o in A però in generale per vedere se in un registro c'è un dato specifico, uso un istruzione di confronto CP (compare). Il 1° operando non si scrive perché implicitamente è A il 2° operando è il dato con cui voglio fare il confronto.

CP α_H A - α_H.

questa istruzione CP fa una sottrazione ma non la mette nell'accumulatore A, va a cambiare solo il registro dei flag, ma nell'accumulatore rimarrà ciò che c'era prima.

```
IN  A,(50H)
CP  00H
JP  Z, Fine
```

quando fa l'istruzione CP sta a 0 quindi: la sottrazione 0 - 0 sarà 0 il registro dei flag cambierà perché il flag di Z sarà 1 e quindi se il flag è 1 significa che il dato era 0 e quindi questo salto porterà alla fine del programma.

```
    IN  A,(50H)
    CP  00H
    JP  Z, Fine
    LD  HL, 8000H
    LD  B,A
Ciclo: IN  A,(50H)
    LD  (HL), A
    INC HL
    DEC B
    JP  NZ, Ciclo
FINE: HALT
```

INC → incrementa

DEC decrementa

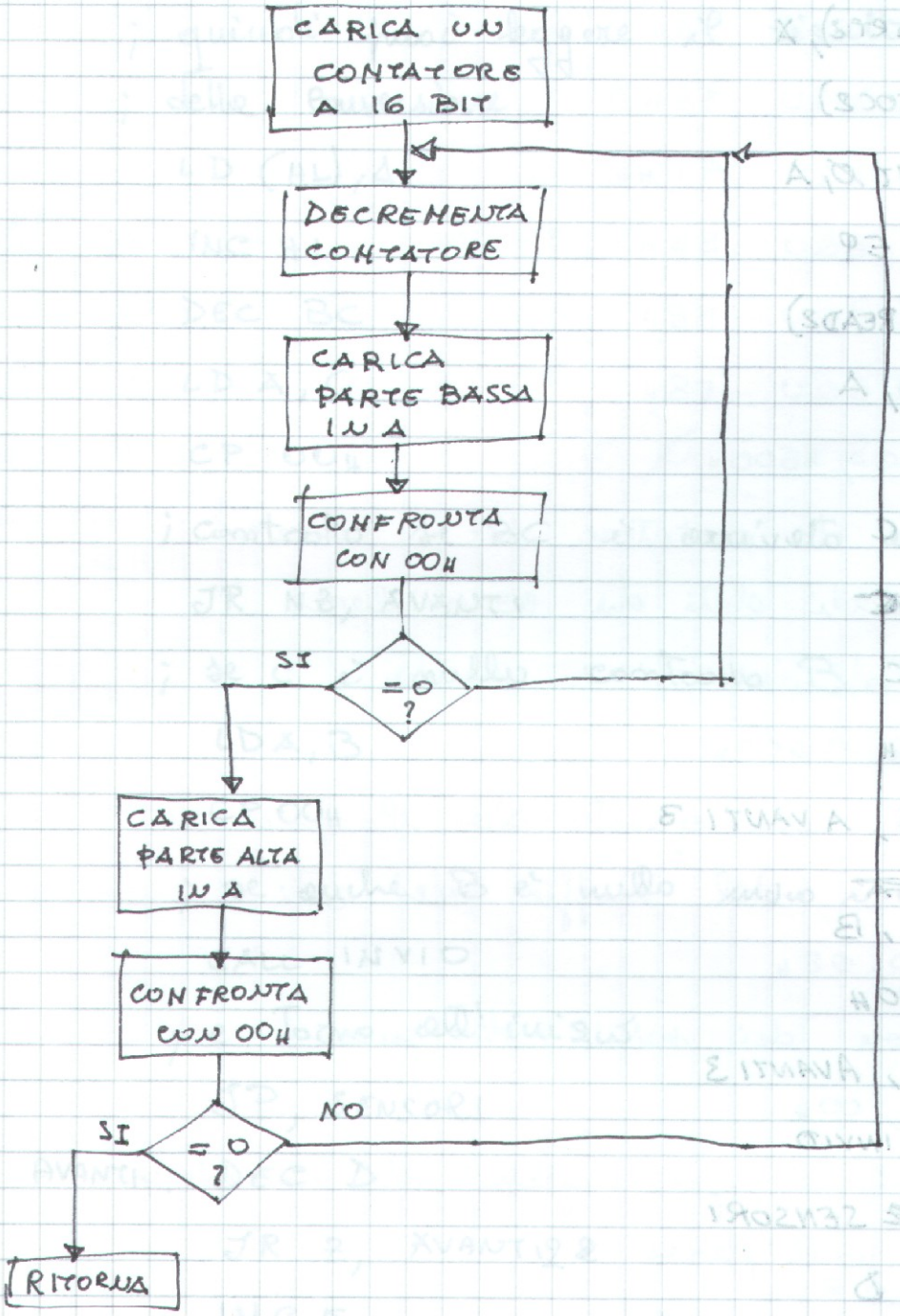
Ciclo: scrivi in memoria all'indirizzo puntato da HL

non è 0.

Se è non è 0 vai a ciclo se è 0 FINE: HALT

PROGRAMMA 53

Scrivere un sottoprogramma di tipo che crea un ritardo di 1



```
RITARDO : ~~LD BC,~~
          PUSH BC ;
          LD BC, XXXX ;
E1 :      DEC BC ,
          LD C, A
          CP 00H
          JR NZ, E1
          ~~CP 00H~~
          LD A, B
          CP 00H
          JR NZ, E1
          POP BC
          RET
```

ciclo interno

Durata ciclo interno

DEC BC	6 TCK
LD C, A	4 TCK
CP 00H	7 TCK
	17 TCK

supponendo $f_{CK} = 1\,MHz \rightarrow T_{CK} = 1\,\mu s$

$17\,T_{CK} = 17\,\mu s$

Supponendo che C contenga FFH in totale si ha

$17\,\mu s \times 255 = 4335\,\mu s$

Il ciclo esterno dura

$4335\,\mu s +$

LD A, B $4\,\mu s +$

Durata ciclo interno

DEC	BC	6 T_{CK}
LD	C, A	4 T_{CK}
CP	00_H	7 T_{CK}
JR NZ, E1		12 T_{CK}
		29 T_{CK}

Supponendo $f_{CK} = 1\ MHz \rightarrow T_{CK} = 1\ \mu s$

$29\ T_{CK} = 29\ \mu s$

Supponendo $C = FF_H$ si ha una durata complessiva

$29 \times 255 =$

$= 7395\ \mu s$

Per avere un ritardo di un 1 s

Il ciclo più esterno dura

	7395 μs +
LD A, B	4 μs +
CP 00_H	7 μs +
JR NZ, E1	12 μs =
	7418 μs

Per avere un ritardo di 1 s tale ciclo deve durare

$1\ s = 1000.000\ \mu s : 7418\ \mu s =$

$\simeq 135$

in B deve essere $135_{/10}$

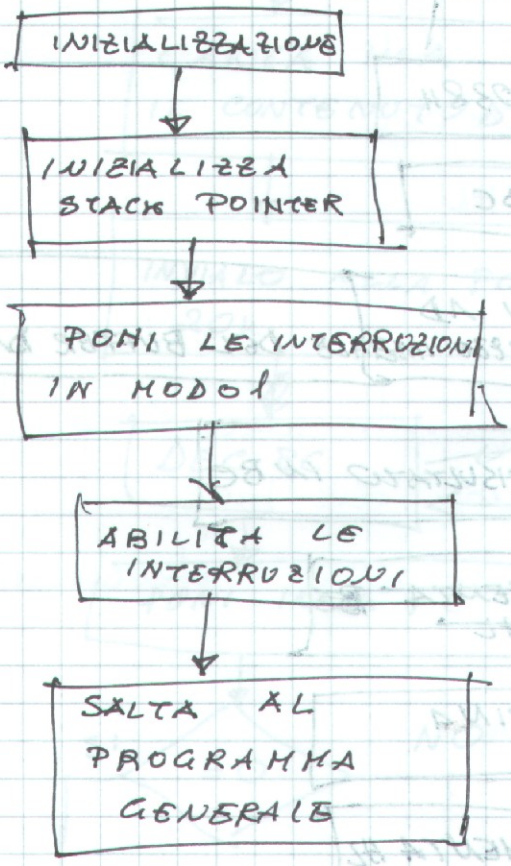
$$\begin{array}{r|l} 135 & 16 \\ 7 & 8 \end{array}$$

in BC deve esserci $87FF_H$:

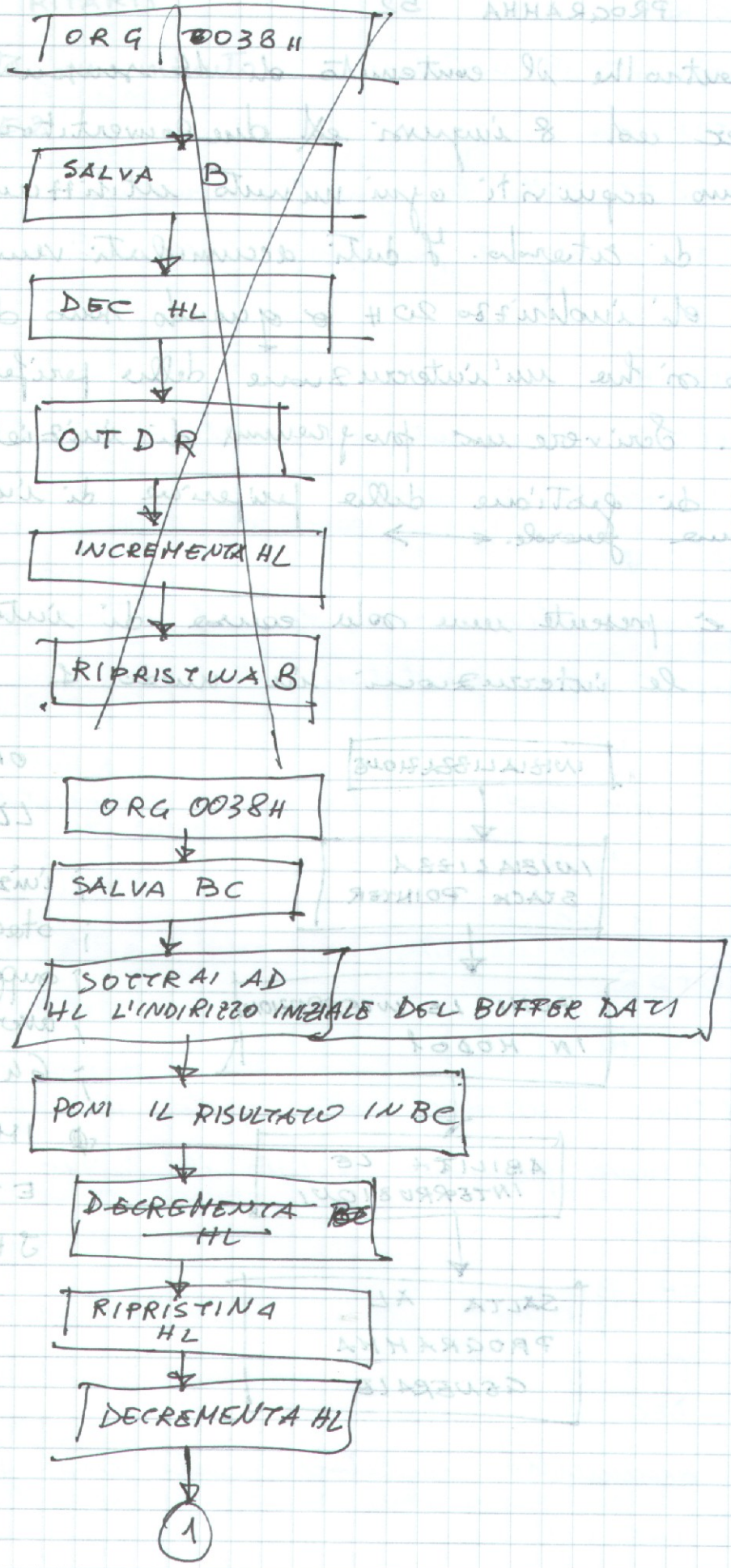
PROGRAMMA 52

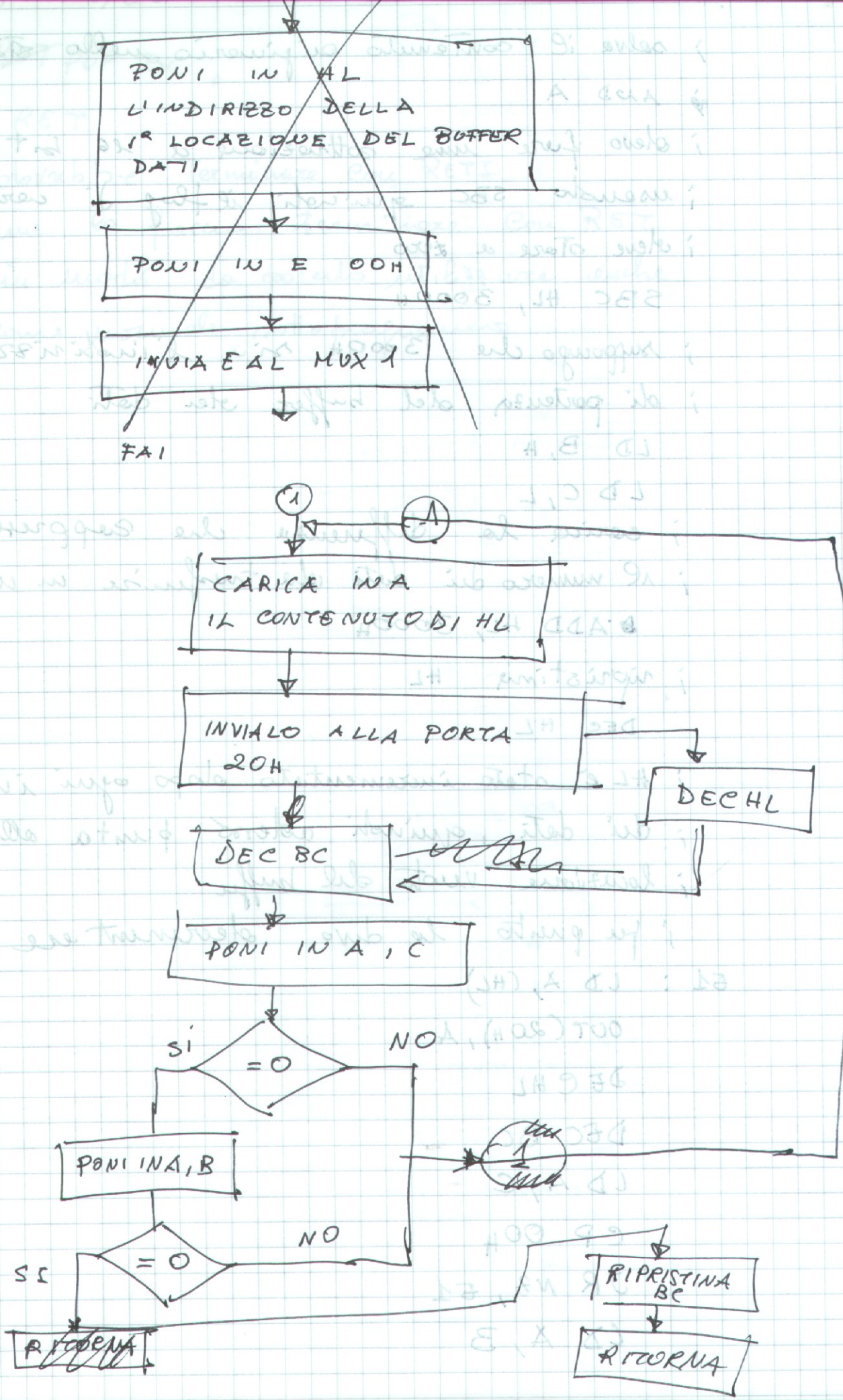
Un µP controlla il contenuto di 16 sensori attraverso due multiplexer ad 8 ingressi ed due convertitori A/D. I dati vengono acquisiti ogni minuto utilizzando un sottoprogramma di ritardo. I dati accumulati vengono spediti alla porta di indirizzo 20H e quando sono diventati 1600 e quando si ha un'interruzione della periferica di indirizzo 30H. Scrivere un programma di inizializzazione e programma di gestione della periferica di indirizzo 30H. e programma generale. ←→

Poiché è presente una sola causa di interruzione usiamo le interruzioni in modo 1




```
ORG  0000H
LD SP, 0FFFFH
; inizializzo lo
; stack pointer
; supponendo di
; avere tutti i
; 64 K di memoria
IM 1
EI
JP SENSORI
```

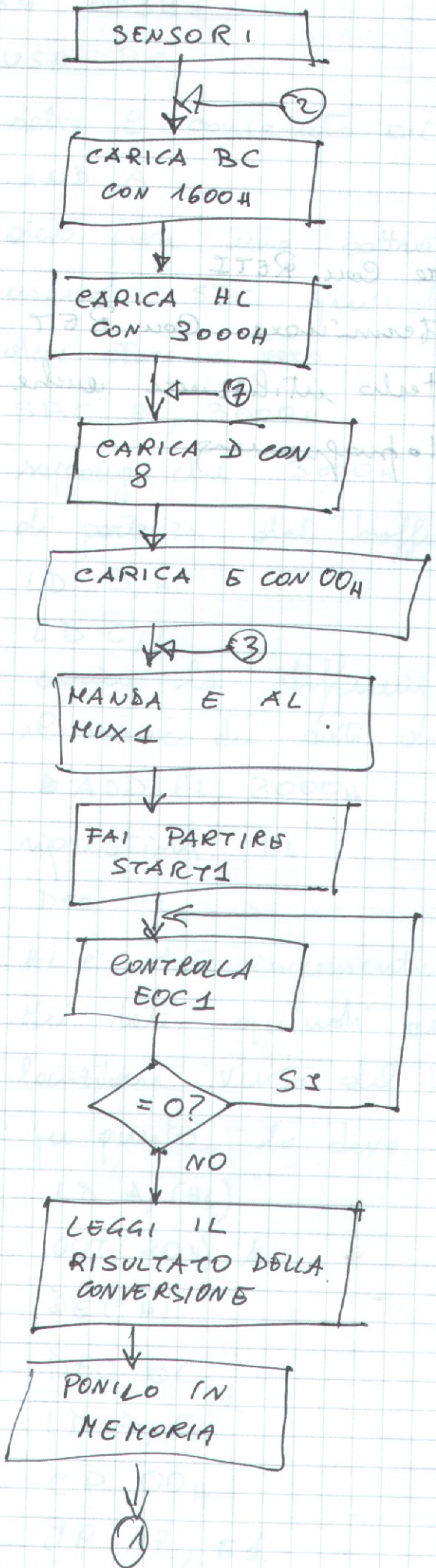

ORG 0038H
SALVA B
DEC HL
OTDR
INCREMENTA HL
RIPRISTINA B
ORG 0038H
SALVA BC
SOTTRAI AD HL L'INDIRIZZO INIZIALE DEL BUFFER DATI
PONI IL RISULTATO IN BC
DECREMENTA HL
RIPRISTINA HL
DECREMENTA HL
1

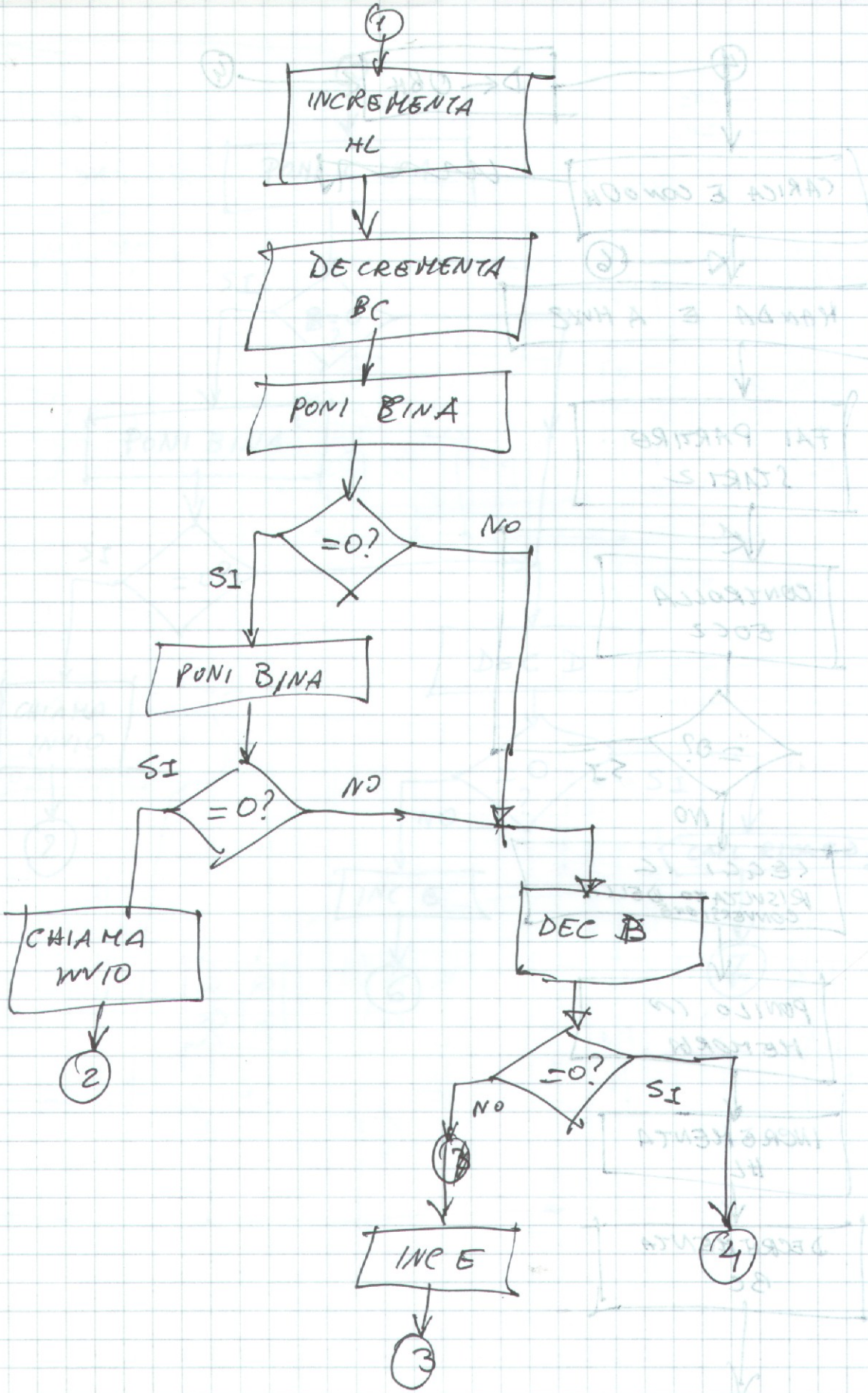

PONI IN HL L'INDIRIZZO DELLA 1ª LOCAZIONE DEL BUFFER DATI
PONI IN E 00H
INVIA E AL MUX 1
FAI
1
A
CARICA IN A IL CONTENUTO DI HL
INVIALO ALLA PORTA 20H
DEC HL
DEC BC
PONI IN A, C
SI
=0
NO
PONI IN A, B
1
=0
NO
SI
RITORNA
RIPRISTINA BC
RITORNA

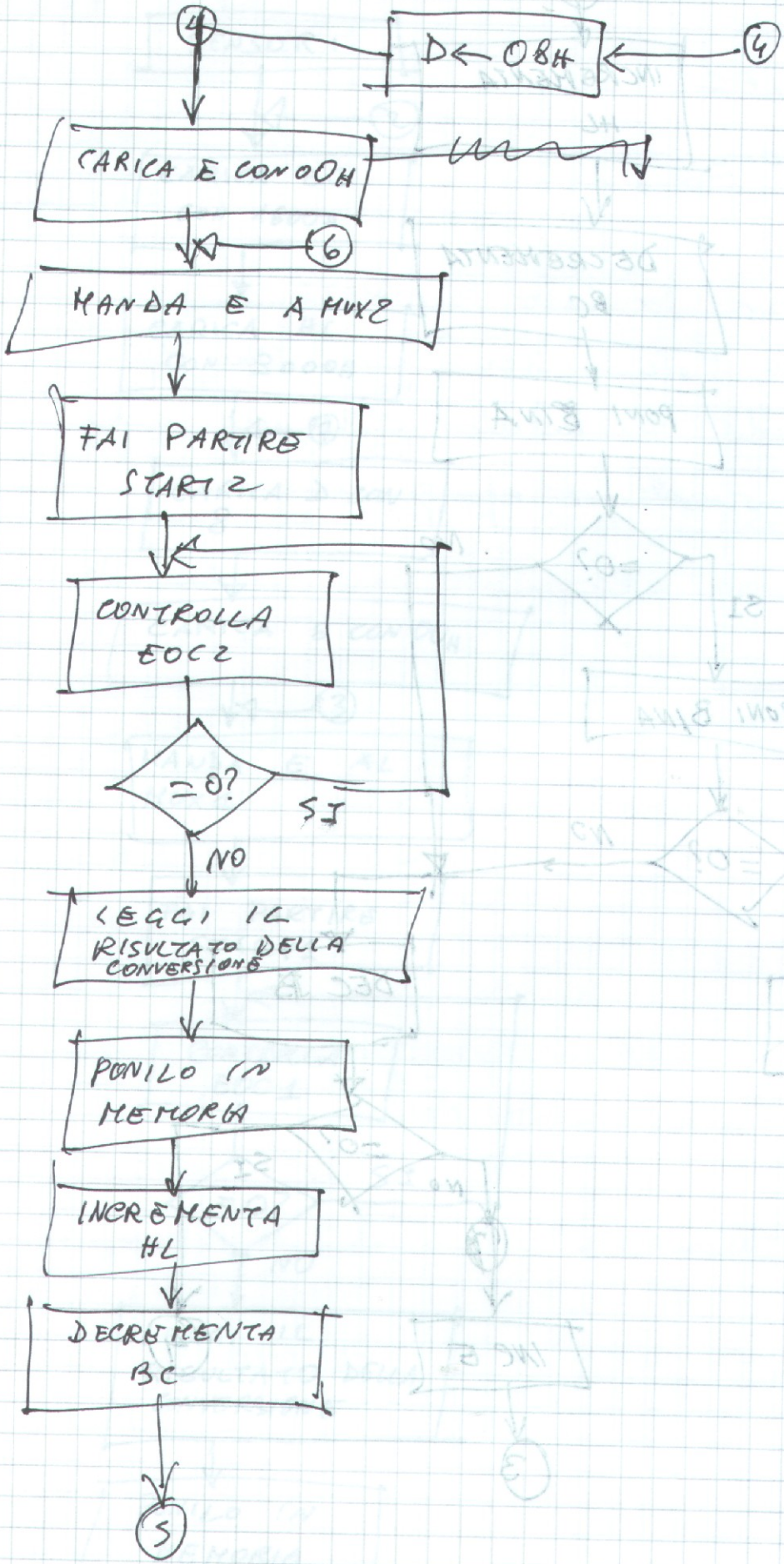
```
        ORG  0038H
INVIO : PUSH BC
        ; salva il contenuto originario nello stack
        ; AND A
        ; devo fare una sottrazione a 16 bit
        ; usando SBC quindi il flag di carry
        ; deve stare a zero
          SBC HL, 3000H
        ; suppongo che 3000H sia l'indirizzo
        ; di partenza del buffer dei dati
          LD B, H
          LD C, L
        ; carico la differenza che rappresenta
        ; il numero dei dati che trasferire in uscita
          ADD HL, 3000H
        ; ripristino HL
          DEC HL
        ; HL è stato incrementato dopo ogni i/o
        ; dei dati, quindi adesso punta alla pr
        ; locazione vuota del buffe
        ; per questo lo devo decrementare
E1 :    LD A, (HL)
        OUT (20H), A
        DEC HL
        DEC BC
        LD A, C
        CP 00H
        JR NZ, E1
```

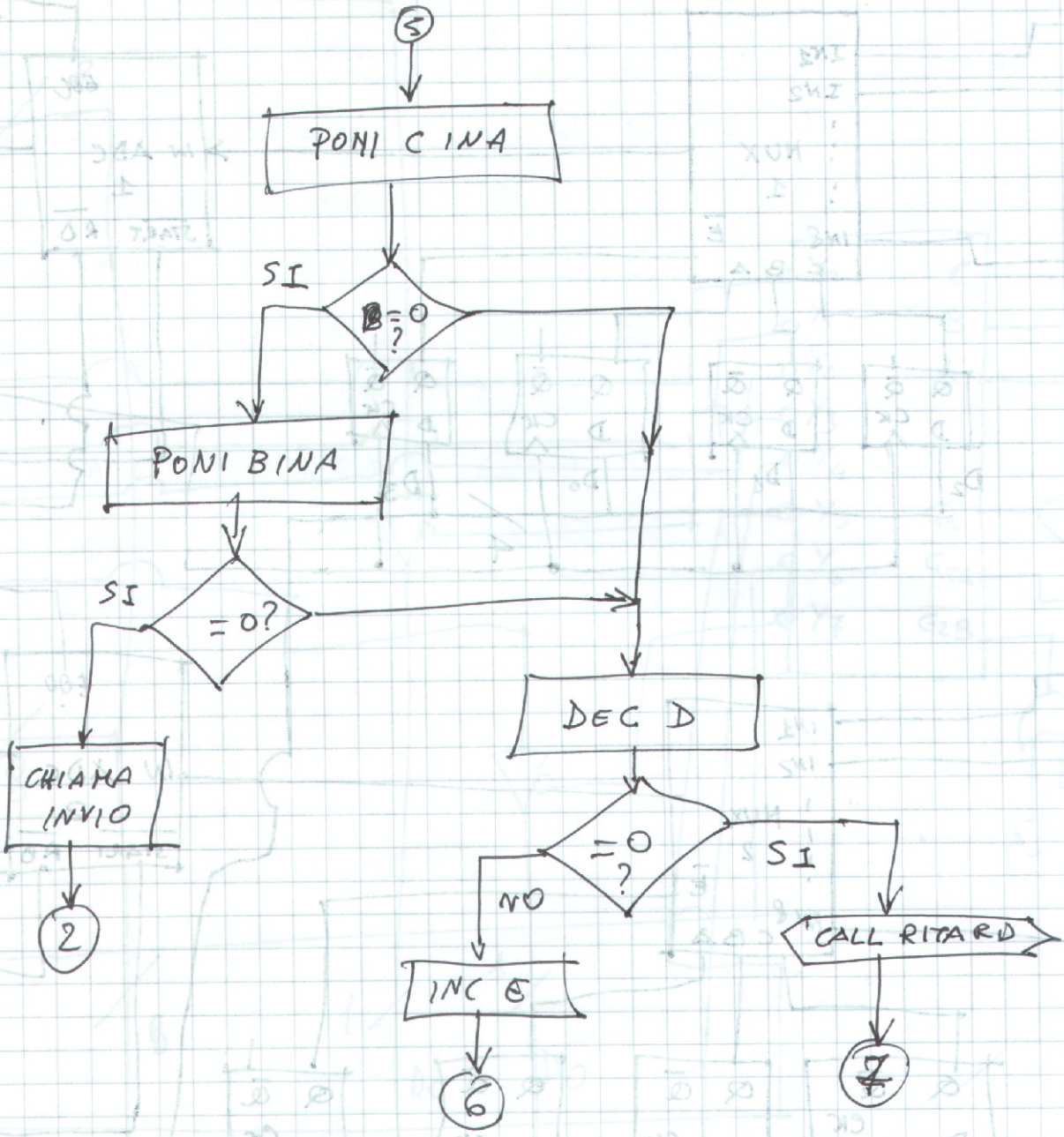
```
INVIO : PUSH BC
        ; salva il contenuto originario nello stack
        ; AND A
        ; devo fare una sottrazione a 16 bit
        ; usando SBC quindi il flag di carry
        ; deve stare a zero
          SBC HL, 3000H
        ; suppongo che 3000H sia l'indirizzo
        ; di partenza del buffer dei dati
          LD B, H
          LD C, L
        ; carica la differenza che rappresenta
        ; il numero dei dati che trasferire in uscita
          ADD HL, 3000H
        ; ripristina HL
          DEC HL
        ; HL è stato incrementato dopo ogni immissione
        ; dei dati, quindi adesso punta alla prima
        ; locazione vuota del buffer
        ; per questo lo devo decrementare
E1 :    LD A, (HL)
        OUT (20H), A
        DEC HL
        DEC BC
        LD A, C
        CP 00H
        JR NZ, E1
        LD A, B
```

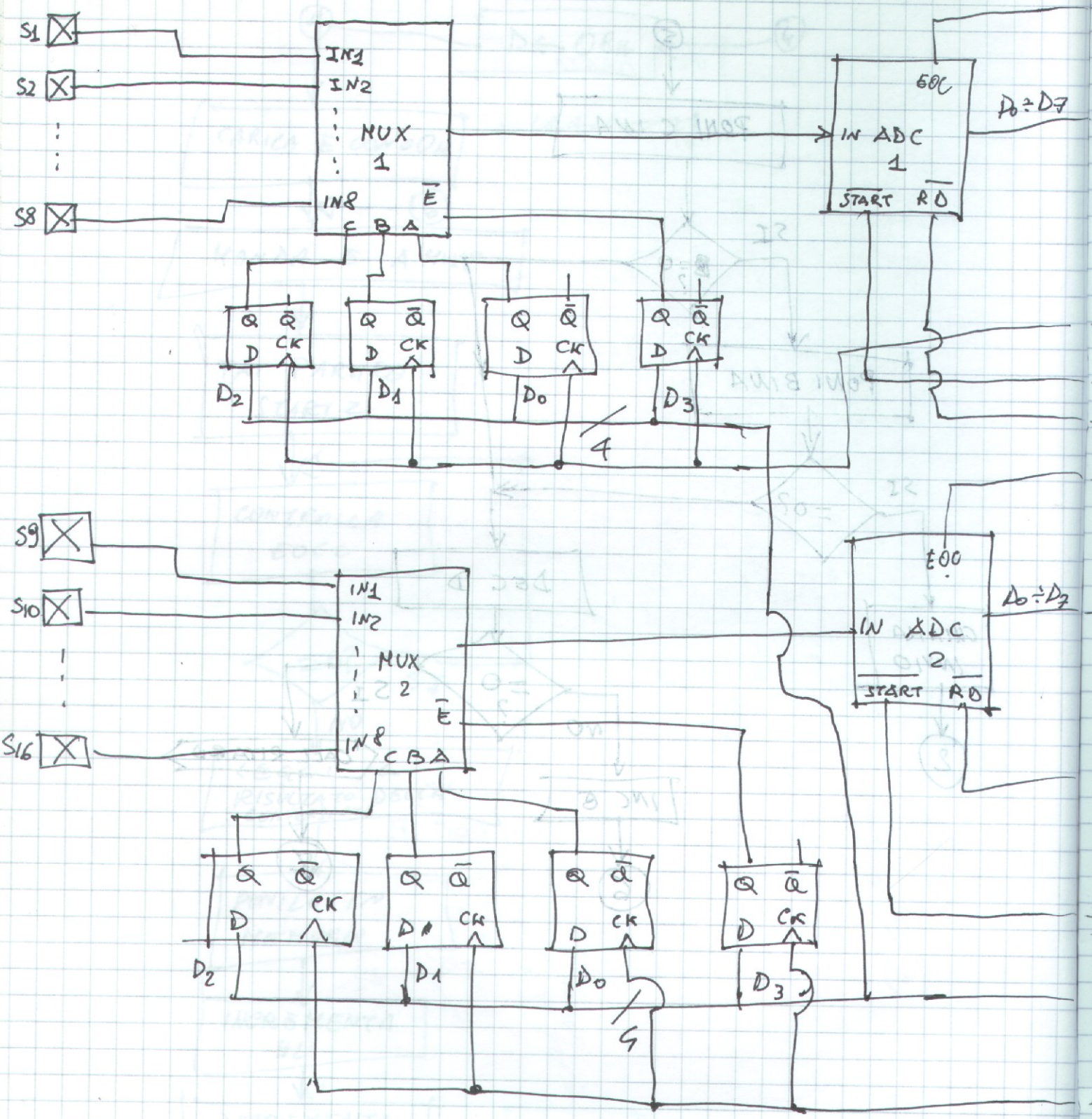
```
CP 00H
JR NZ, E1
~~RET~~ POP BC
; RET
; dovrebbe Terminare con RETI
; ma lo faccio Terminare con RET
; in modo da poterlo utilizzare anche
; come normale sottoprogramma
```

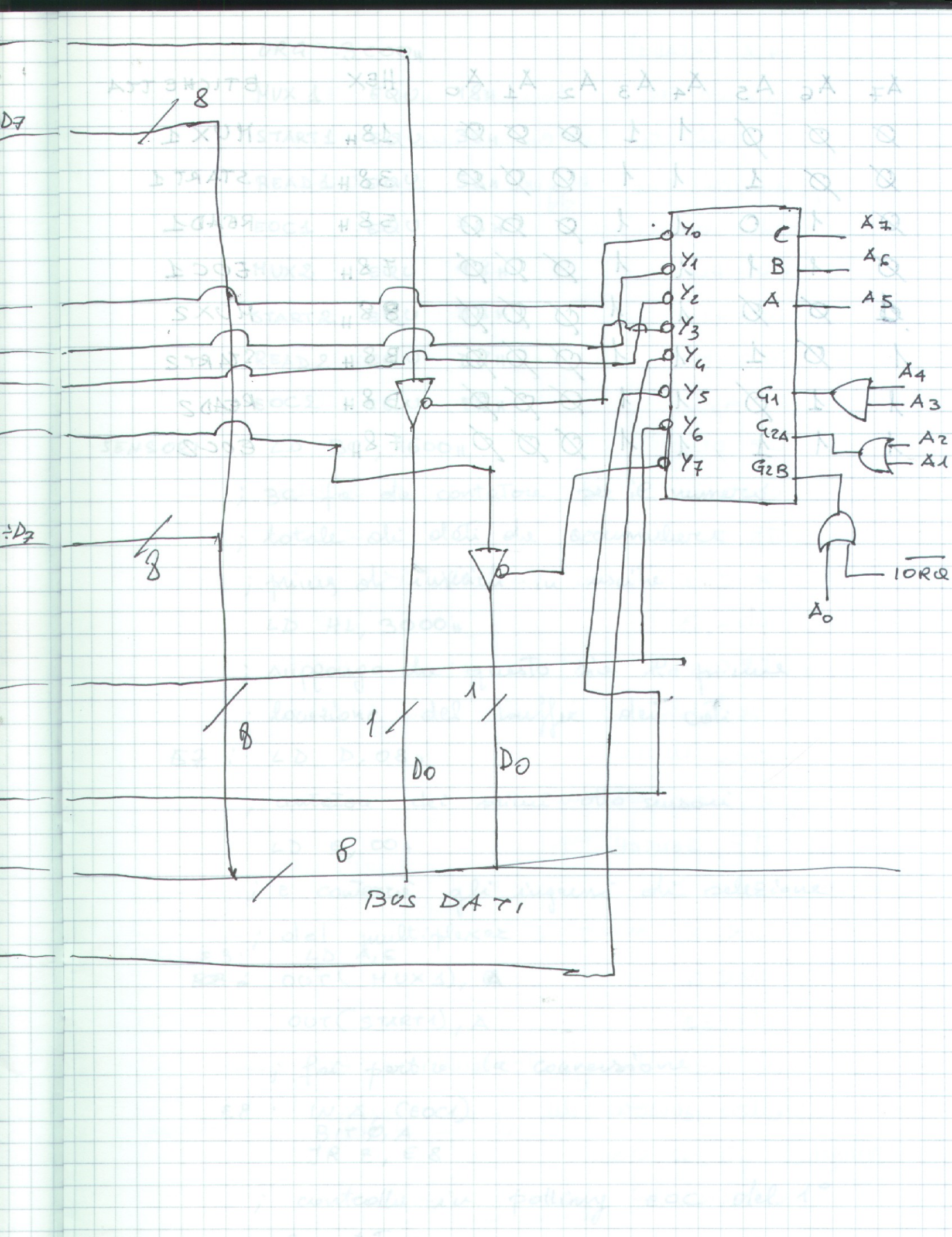

SENSORI
2
CARICA BC CON 1600H
CARICA HL CON 3000H
7
CARICA D CON 8
CARICA E CON 00H
3
MANDA E AL MUX1
FAI PARTIRE START1
CONTROLLA EOC1
= 0?
SI
NO
LEGGI IL RISULTATO DELLA CONVERSIONE
PONILO IN MEMORIA
1


1
INCREMENTA HL
DECREMENTA BC
PONI B/NA
=0?
NO
SI
PONI B/NA
SI
=0?
NO
CHIAMA INVIO
2
DEC B
=0?
SI
NO
INC E
3
4


4
D ← 08H
4
CARICA E CON 00H
6
MANDA E A MUX2
FAI PARTIRE START2
CONTROLLA EOC2
=0?
SI
NO
LEGGI IL RISULTATO DELLA CONVERSIONE
PONILO IN MEMORIA
INCREMENTA HL
DECREMENTA BC
5


5
PONI C INA
SI
B=0
?
PONI BINA
SI
=0?
CHIAMA
INVIO
2
DEC D
=0
?
SI
NO
CALL RITARD
INC E
6
7


S1
S2
S8
IN1
IN2
MUX 1
IN8
E
C B A
EOC
D0 ÷ D7
IN ADC 1
START RD
Q Q̄ D CK
D2
D1
D0
D3
4
S9
S10
S16
MUX 2
EOC
D0 ÷ D7
IN ADC 2
START RD
D2
D1
D0
D3
4


8
D7
÷D7
8
8
1
1
1
D0
D0
8
BUS DATI
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
C
B
A
G1
G2A
G2B
A7
A6
A5
A4
A3
A2
A1
IORQ
A0

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX	ETICHETTA
0	0	0	1	1	0	0	0	18_H	MUX1
0	0	1	1	1	0	0	0	38_H	START1
0	1	0	1	1	0	0	0	58_H	READ1
0	1	1	1	1	0	0	0	78_H	EOC1
1	0	0	1	1	0	0	0	98_H	MUX2
1	0	1	1	1	0	0	0	$B8_H$	START2
1	1	0	1	1	0	0	0	$D8_H$	READ2
1	1	1	1	1	0	0	0	$F8_H$	EOC2

```
        ORG 3C00H
MUX1    EQU 18H
START1  EQU 38H
READ1   EQU 58H
EOC1    EQU 78H
MUX2    EQU 98H
START2  EQU B8H
READ2   EQU D8H
EOC2    EQU F8H
SENSORI : LD BC, 1600H
        ; BC fa da contatore per il numero
        ; totale di dati da accumulare
        ; prima di inviarli in uscita
          LD HL, 3000H
        ; suppongo che questo sia la prima
        ; locazione del buffer dei dati
E7 :    LD D, 08H
        ; contatore dei primi otto sensori
          LD E, 00H
        ; E conterrà gli ingressi di selezione
        ; del multiplexer
E3:       LD A,E
          OUT(MUX1), A
          OUT(START1), A
        ; fai partire la conversione
E8 :    IN A, (EOC1)
          BIT 0, A
          JR Z, E8
        ; controlla in polling EOC del 1°
```

```
MUX1    EQU  18H
START1  EQU  38H
READ1   EQU  58H
EOC1    EQU  78H
MUX2    EQU  98H
START2  EQU  B8H
READ2   EQU  D8H
EOC2    EQU  F8H
SENSORI : LD BC, 1600H
        ; BC fa da contatore per il numero
        ; totale di dati da accumulare
        ; prima di inviarli in uscita
          LD HL, 3000H
        ; suppongo che questo sia la prima
        ; locazione del buffer dei dati
E7 :    LD D, 08H
        ; contatore dei primi otto sensori
          LD E, 00H
        ; E conterrà gli ingressi di selezione
        ; del multiplexer
E3:       LD A, E
E8 :    OUT (MUX1), A
          OUT (START1), A
        ; fai partire la conversione
E8 :    IN A, (EOC1)
          BIT 0, A
          JR Z, E8
        ; controlla in polling EOC del 1°
        ; convertitore
```

```
        IN A, (READ1)
        ; quando EOC è tornato ad 1
        ; è terminata la conversione
        ; quindi puoi leggere il risultato
        ; della conversione
        LD (HL), A
        INC HL
        DEC BC
        LD A, C
        CP 00H
        ; controllo se BC è arrivato a zero
        JR NZ, AVANTI
        ; se C è nullo controllo B
        LD A, B
        CP 00H
        ; se anche B è nullo invio i dati
        CALL INVIO
        ; e torno all'inizio
        JP, SENSORI
AVANTI: DEC D
        JR Z, AVANTI2
        INC E
        JP, E3
        ; se non ho finito continuo con un
        ; altro sensore del primo gruppo
AVANTI2: LD D, 08H
        ; altrimenti passo ai sensori
```

```
; quando EOC è tornato ad 1
; è terminata la conversione
; quindi puoi leggere il risultato
; della conversione
    LD (HL), A
    INC HL
    DEC BC
    LD A, C
    CP 00H
; controllo se BC è arrivato a zero
    JR NZ, AVANTI
; se C è nullo controllo B
    LD A, B
    CP 00H
; se anche B è nullo invio i dati
    CALL INVIO
; e torno all'inizio
    JP, SENSORI
AVANTI: DEC D
    JR Z, AVANTI2
    INC E
    JP, E3
; se non ho finito continuo con un
; altro sensore del primo gruppo
AVANTI2: LD D, 08H
; altrimenti passo ai sensori
; del secondo gruppo
```

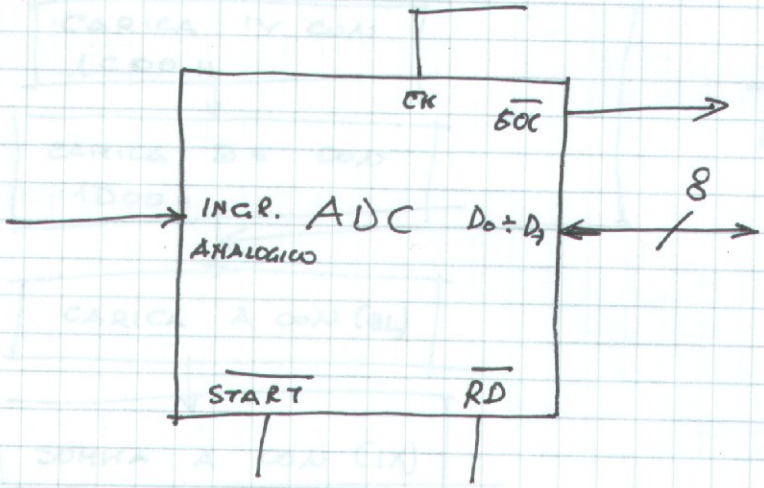
```
        LD E, 00H
E6 :    LD A, E
        OUT (MUX2), A
        OUT (START2), A
E9 :    IN A, (EOC2)
        JR Z BIT 0, A
        JR NZ, E9
        IN A, (READ2)
        LD (HL), A
        INC HL
        DEC BC
        LD A, C
        CP 00H
        JR NZ, AVANTI 3
        LD A, B
        CP 00H
        JR NZ, AVANTI 3
        CALL INVIO
        JP, SENSORI
AVANTI 3 : DEC D
        JR Z, AVANTI 4
        INC E
        JP, E6
AVANTI 4 : CALL RITARD
        ; sono finiti anche i sensori del 2° gruppo
        ; e aspetto un minuto
```

```
        OUT (MUX2), A
        OUT (START2), A
E9 :    IN A, (EOC2)
        JR Z BIT 0, A
        JR NZ, E9
        IN A, (READ2)
        LD (HL), A
        INC HL
        DEC BC
        LD A, C
        CP 00H
        JR NZ, AVANTI 3
        LD A, B
        CP 00H
        JR NZ, AVANTI 3
        CALL INVIO
        JP, SENSORI
AVANTI 3 : DEC D
        JR Z, AVANTI 4
        INC E
        JP, E6
AVANTI 4 : CALL RITARD
        ; sono finiti anche i sensori del 2° gruppo
        ; e aspetto un minuto
        JP, E7.
```

PROGRAMMA 51

Un µP Z80 controlla 128 sensori analogici tramite due multiplexer analogici ad 8 ingressi e due convertitori A/D; scrivere un programma che acquisisca i dati ogni minuto; si ha a disposizione un sottoprogramma RITARD. Acquisiti 100 dati, questi vengono inviati alla porta FOH e si ricomincia da capo.

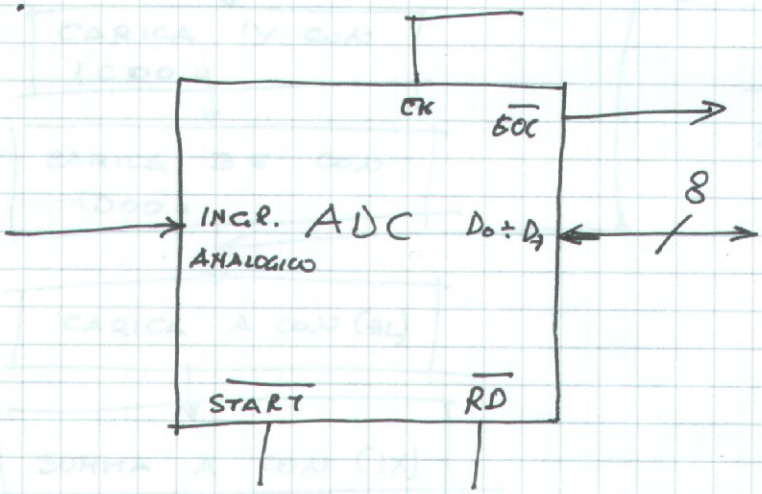
Questo esercizio implica delle considerazioni sull'hardware. Un convertitore A/D è un dispositivo elettronico che presenta un ingresso a cui giunge un segnale analogico il cui valore verrà convertito in un dato numerico. L'ADC presenta 8 linee di uscita digitali sulle quali scaricherà il dato numerico risultato della conversione. Queste uscite sono three-state in modo da consentire l'interfacciamento con il bus dati di un µP.



L'ADC presenta anche un ingresso di START per far partire la conversione e un ingresso di RD per leggere il risultato della conversione. Il processo di conversione è temporizzato da un segnale di clock la cui frequenza massima dipende dal tipo di ADC. La durata della conversione dipende dalla frequenza del clock ed è in genere espressa in N cicli di clock. In genere l'ADC

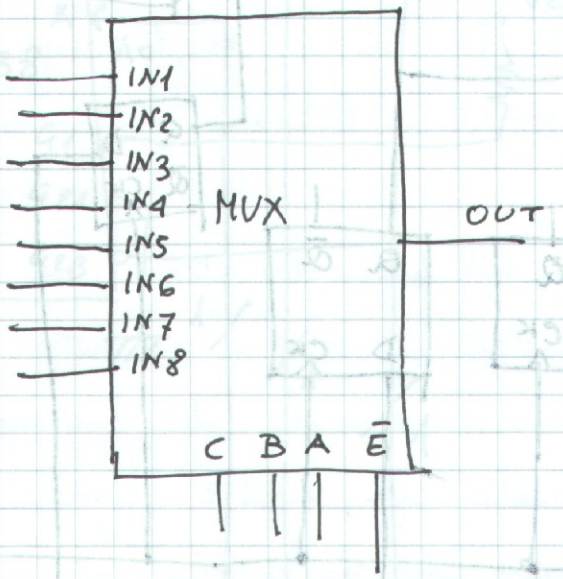
ad 8 ingressi e due convertitori A/D; scrivere un programma che acquisisca i dati ogni minuto; si ha a disposizione un sottoprogramma RITARD. Acquisiti 100 dati, questi vengono inviati alla porta F0H e si ricomincia da capo.

Questo esercizio implica delle considerazioni sull'hardware. Un convertitore A/D è un dispositivo elettronico che presenta un ingresso a cui giunge un segnale analogico il cui valore verrà convertito in un dato numerico. L'ADC presenta 8 linee di uscita digitali sulle quali scaricherà il dato numerico risultato della conversione. Queste uscite sono three-state in modo da consentire l'interfacciamento con il bus dati di un µP.



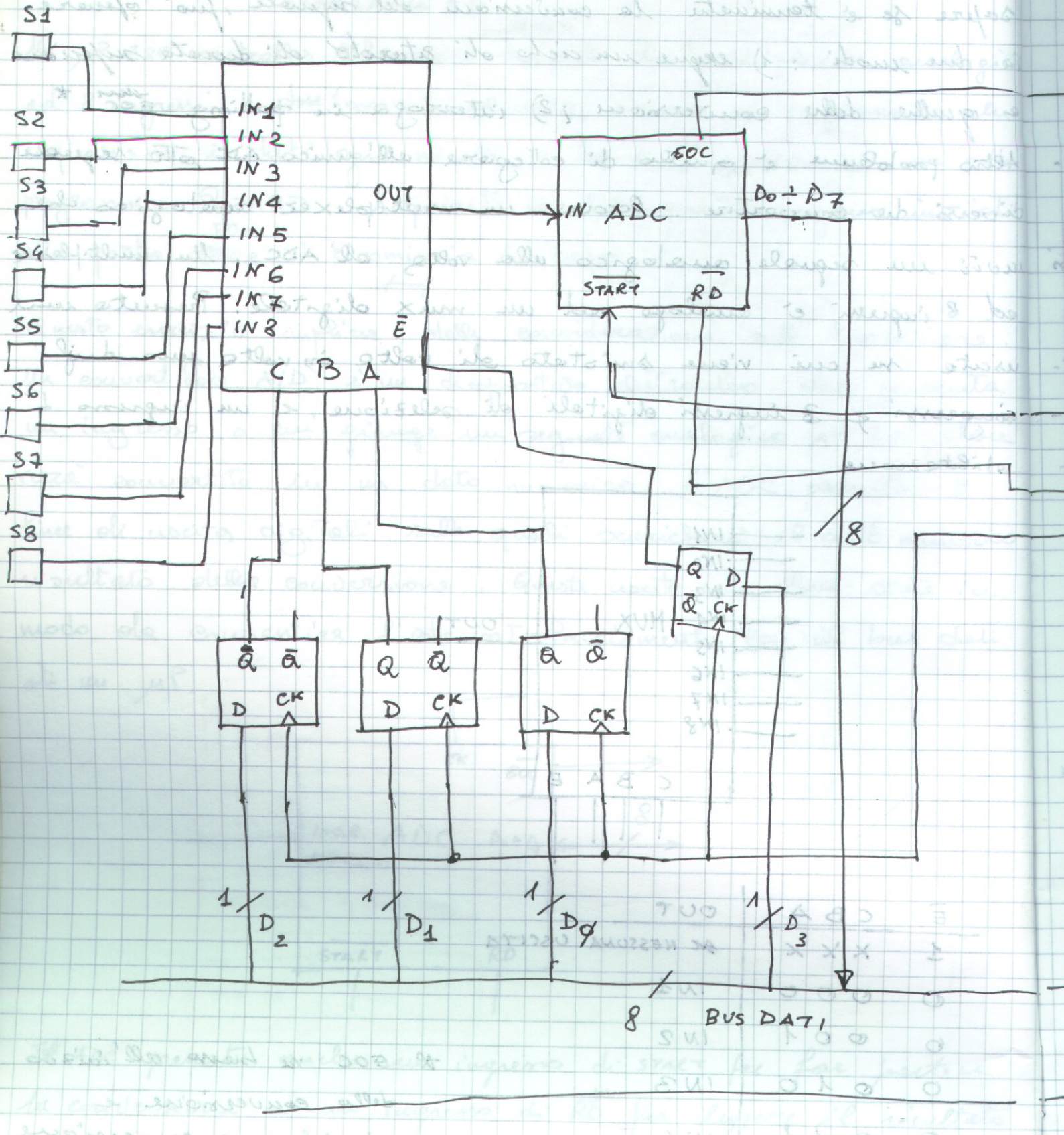
L'ADC presenta anche un ingresso di START per far partire la conversione e un ingresso di RD per leggere il risultato della conversione. Il processo di conversione è temporizzato da un segnale di clock la cui frequenza massima dipende dal tipo di ADC. La durata della conversione dipende dalla frequenza del clock ed è in genere espressa in N cicli di clock. In genere l'ADC presenta una uscita EOC che segnala la fine della conversione. Il µP, per

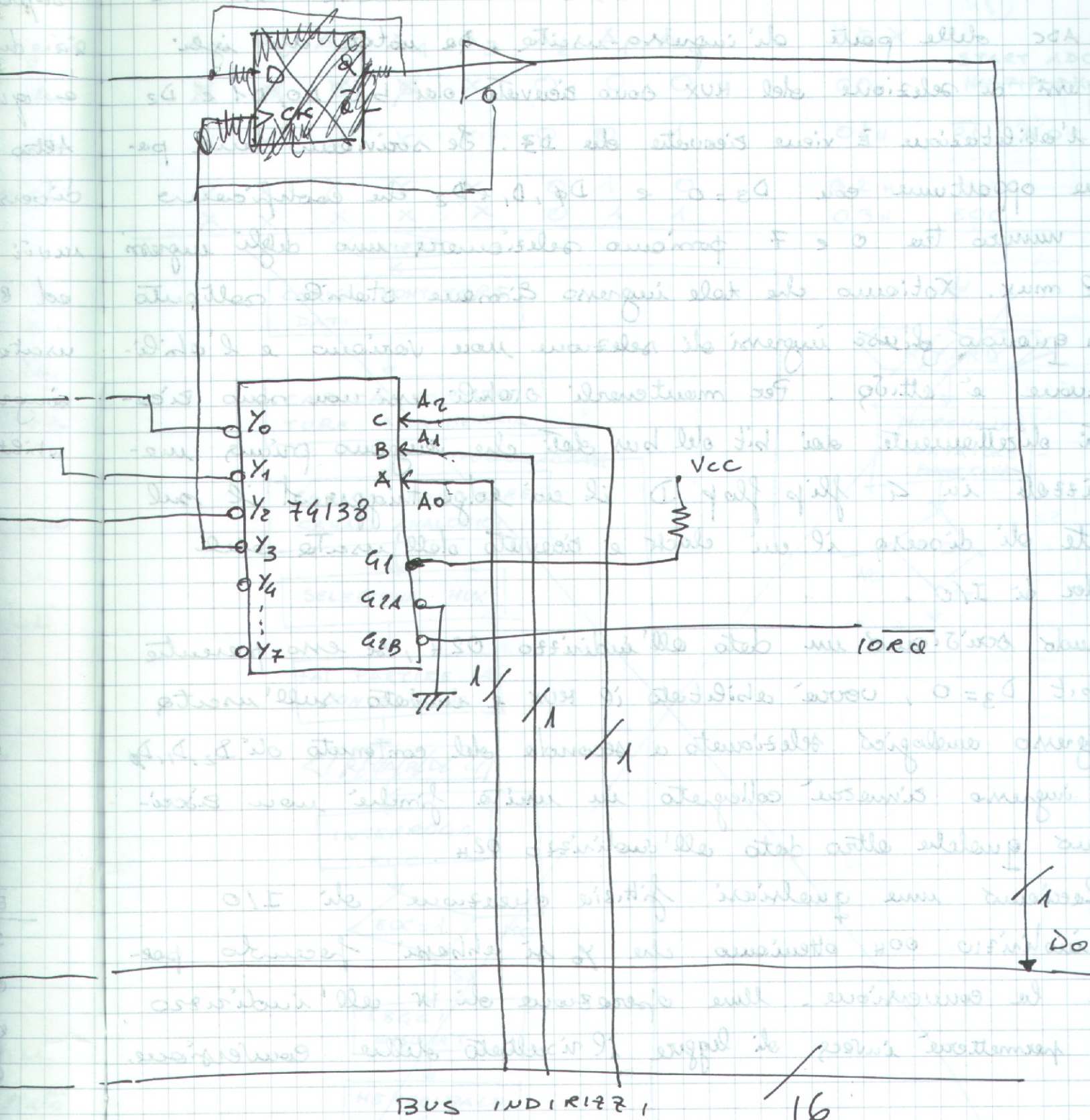
sapere se è terminata la conversione del segnale, può operare in due modi: 1) esegue un ciclo di ritardo di durata superiore a quella della conversione; 2) interroga in polling $\overline{EOC}$.*
Altro problema è quello di collegare all'unico ADC otto segnali diversi da convertire. Occorre un multiplexer analogico che invii un segnale analogico alla volta all'ADC. Un multiplexer ad 8 ingressi è analogo ad un mux digitale. Premesso una usata su cui viene suddiviso di volta in volta uno degli ingressi a 3 ingressi digitali di selezione e un ingresso di abilitazione



Ē	C B A	OUT
1	X X X	~~X~~ NESSUNA USCITA
0	0 0 0	IN1
0	0 0 1	IN2
0	0 1 0	IN3
0	0 1 1	IN4
⋮	⋮	⋮
0	1 1 1	IN8

* EOC va basso all'inizio della conversione e si rialza a conversione terminata


S1
S2
S3
S4
S5
S6
S7
S8
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
OUT
C B A
Ē
EOC
IN ADC
D0 ÷ D7
START
RD
8
Q D
Q̄ CK
Q̄ Q̄
D CK
Q Q̄
Q Q̄
1
1/ D2
1/ D1
1/ D0
1/ D3
8
BUS DATI


D
Q
CK
Q
Y0
Y1
Y2
Y3
Y4
Y7
74138
C
B
A
G1
G2A
G2B
A2
A1
A0
Vcc
IORQ
1
1
1
1
D0
BUS INDIRIZZI
16

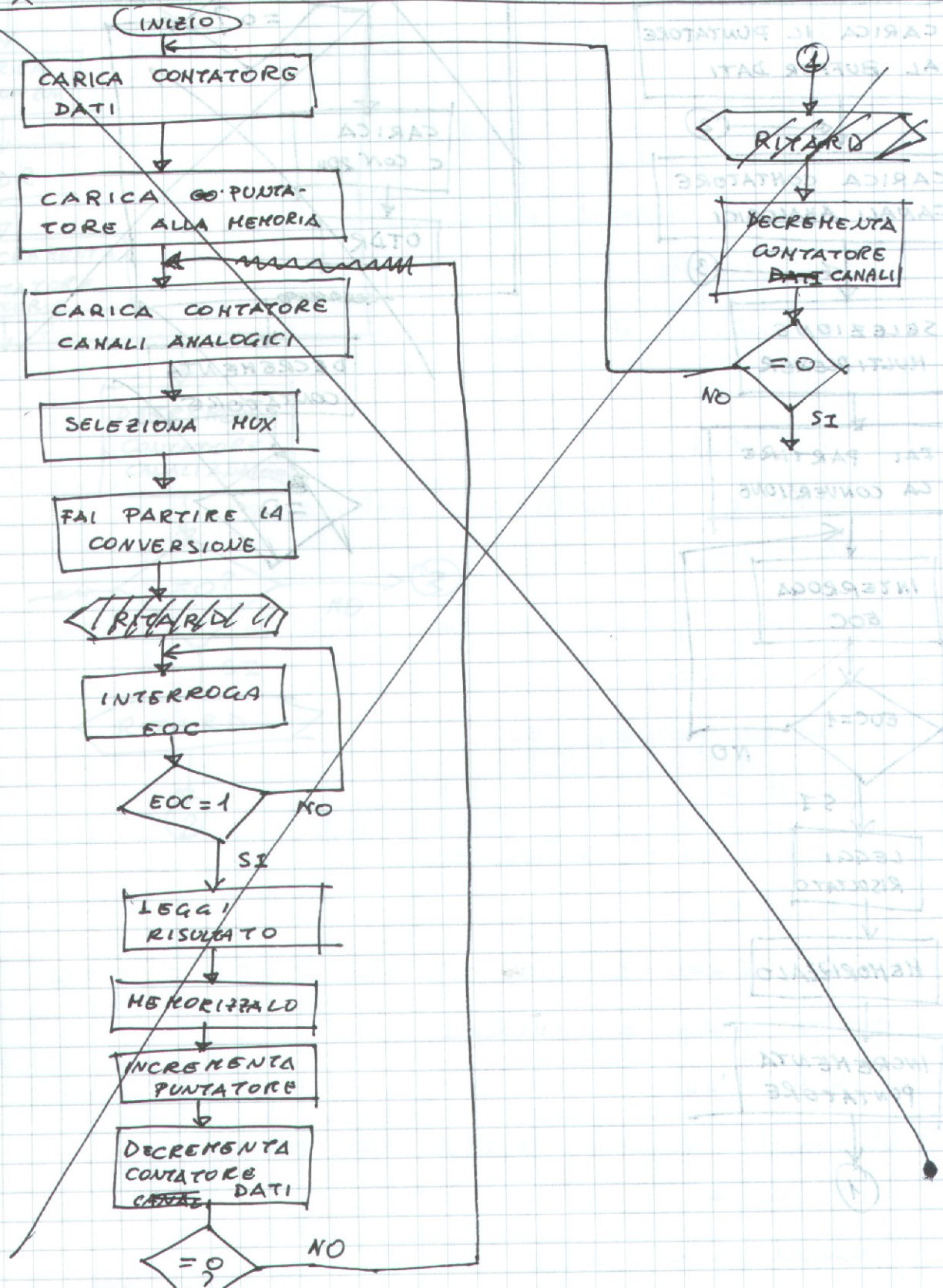
Nello schema vediamo che abbiamo reso sia il MUX che l'ADC delle porte di ingresso-uscita. Da notare che gli ingressi di selezione del MUX sono ricevuti dai bit D0, D1 e D2 e l'abilitazione E viene ricevuta da D3. Se scriviamo una parola opportuna con $D_3 = 0$ e D_0, D_1 e D_2 che codificano un numero tra 0 e 7 possiamo selezionare uno degli ingressi del mux. Notiamo che tale ingresso rimane stabile soltanto fin quando gli ingressi di selezione non variano e l'abilitazione è attiva. Per mantenerli stabili essi non sono ricevuti direttamente dai bit del bus dati che vengono prima memorizzati in 4 flip flop D il cui edge triggered il fronte di discesa il cui clock è ricevuto dall'uscita del decoder di I/O.

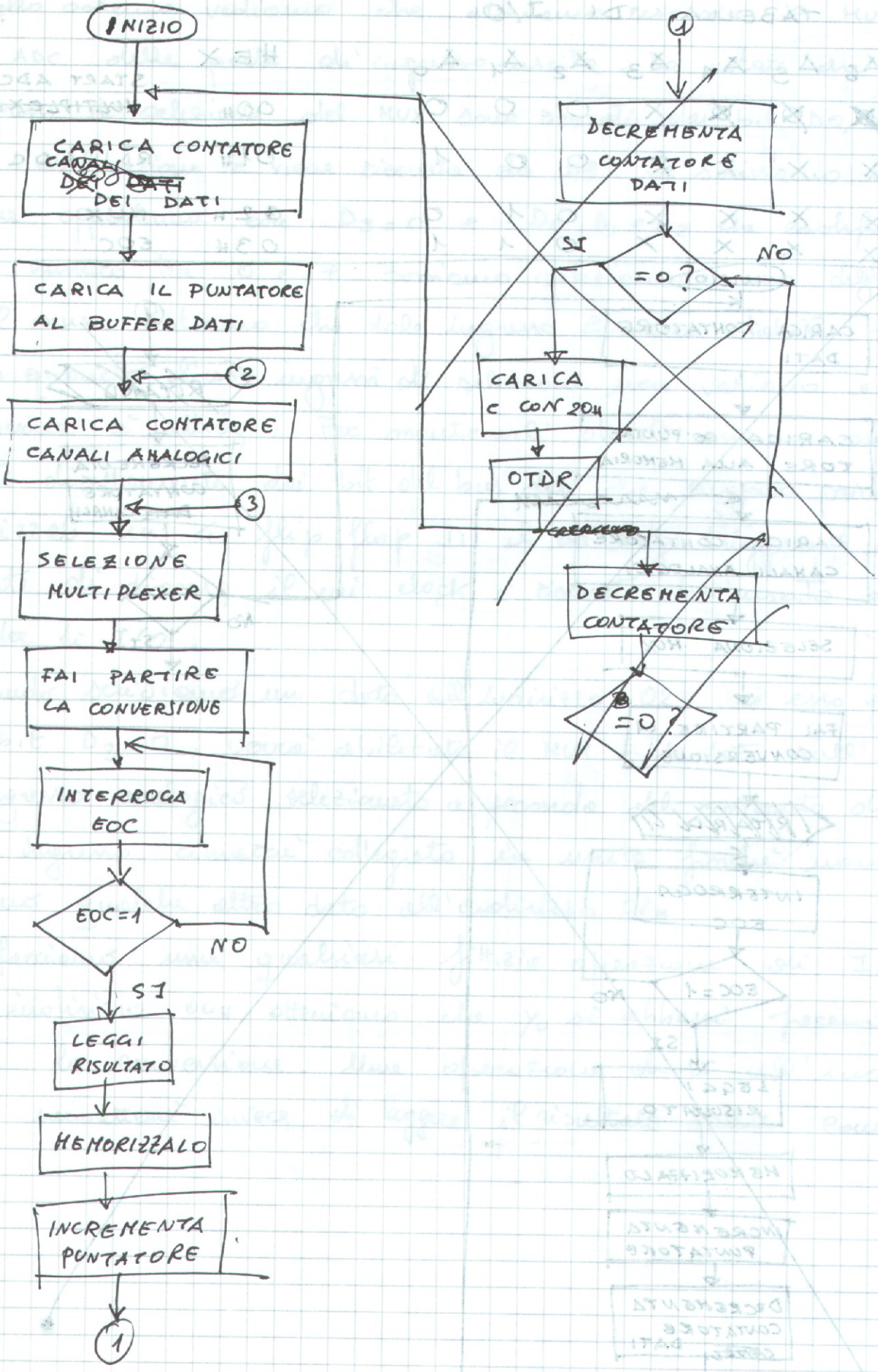
Quando scriviamo un dato all'indirizzo 02_H, se esso presenta il bit $D_3 = 0$, verrà abilitato il MUX e inviato sull'uscita l'ingresso analogico selezionato a seconda del contenuto di D_2, D_1. Tale ingresso rimarrà collegato in uscita finché non riscriveremo qualche altro dato all'indirizzo 02_H.

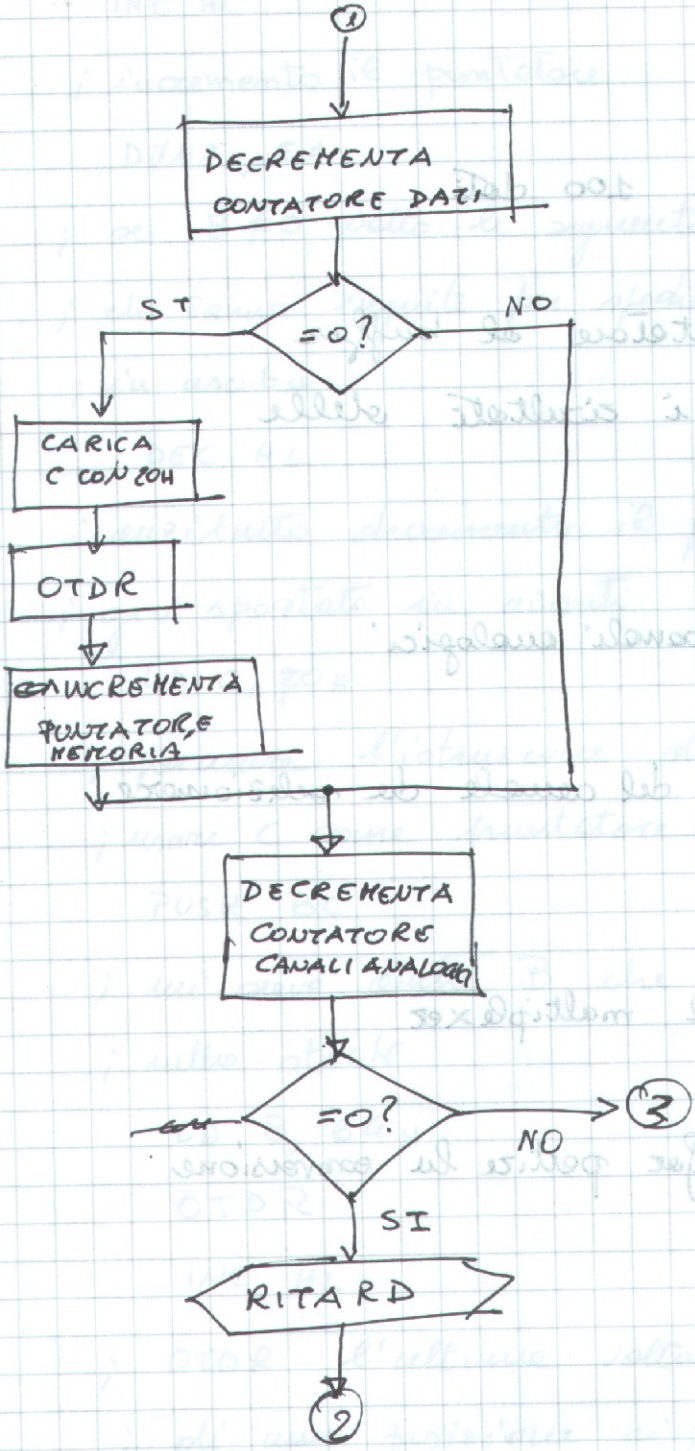
Se facciamo una qualsiasi fittizia operazione di I/O all'indirizzo 00_H otteniamo che Vo si abbassi facendo partire la conversione. Una operazione di IN all'indirizzo 01_H permetterà invece di leggere il risultato della conversione

TABELLA DI I/O

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	HEX	
~~X~~	~~X~~	~~X~~	~~X~~	X	0	0	0	00H	START ADC ~~MULTIPLEXER~~
X	X	X	X	X	0	0	1	01H	$\overline{RD}$ ADC
X	X	X	X	X	0	1	0	02H	MUX
X	X	X	X	X	0	1	1	03H	EOC




INIZIO
CARICA CONTATORE CANALI DEI DATI
CARICA IL PUNTATORE AL BUFFER DATI
2
CARICA CONTATORE CANALI ANALOGICI
3
SELEZIONE MULTIPLEXER
FAI PARTIRE LA CONVERSIONE
INTERROGA EOC
EOC=1
NO
SI
LEGGI RISULTATO
MEMORIZZALO
INCREMENTA PUNTATORE
1
1
DECREMENTA CONTATORE DATI
SI
=0 ?
NO
CARICA c CON 20μ
OTDR
DECREMENTA CONTATORE
=0 ?


1
DECREMENTA CONTATORE DATI
=0?
SI
NO
CARICA C CON 20H
OTDR
INCREMENTA PUNTATORE MEMORIA
DECREMENTA CONTATORE CANALI ANALOGICI
=0?
3
NO
SI
RITARD
2

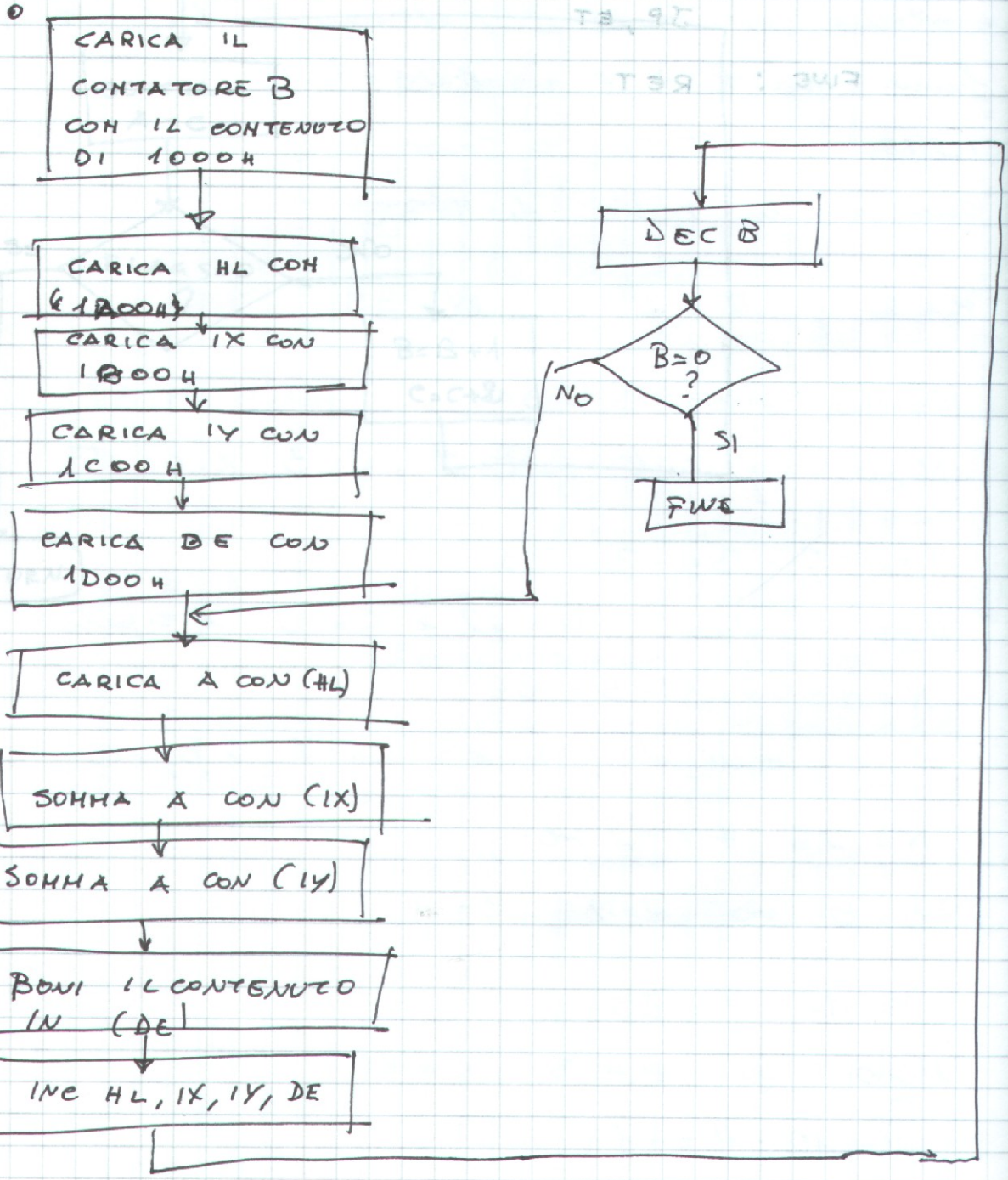
```
        ORG  XXXX;
        LD   B, 64H;
    ; B è il contatore dei 100 dati
        LD   HL, 1800H
    ; usiamo HL come puntatore al buffer
    ; che riempiremo con i risultati delle
    ; conversioni
E2 :    LD   D, 08H
    ; D è il contatore dei canali analogici
        LD   E, 00H
    ; E contiene il numero del canale da selezionare
E3 :    LD   A, E
        OUT (02H), A
    ; selezione il canale del multiplexer
        OUT (00H), A
    ; scrittura fittizia per far partire la conversione
E1 :    IN   A, (03H)
    ; lettura per prelevare EOC
        BIT  0, A
    ; il bit 0 di A contiene EOC
        JR   Z, E1
    ; se flag di zero = 1 vuol dire che il bit 0 è a zero
    ; quindi ritorno a controllare
        IN   A, (01H)
    ; altrimenti leggo il risultato della conversione
        LD   (HL), A
    ; memorizzo nel buffer
```

```
        INC HL
    ; incremento il puntatore
        DJNZ, E4
    ; se B ≠ 0 salto le seguenti istruzioni
    ; che sono eseguite per spedire i 100 dati accumulati
    ; in uscita
        DEC HL
    ; anzitutto decremento il puntatore che bravo
    ; già spostato in avanti
        LD C, 70H
    ; per usare l'istruzione di out ricordo devo
    ; usare C come puntatore alla porta di I/O
        PUSH BC
    ; mi serve anche B che salvo momentaneamente
    ; nello stack
        LD B, 64H
        OTDR
        INC HL
    ; OTDR l'ultima volta sposta HL all'indietro
    ; di una posizione rispetto all'inizio del
    ; buffer
        POP BC
    ; ripristino B
E4 : DEC D
    ; decremento contatore cicli esterno
        INC E
    ; incremento l'indirizzo del ciclo esterno
```

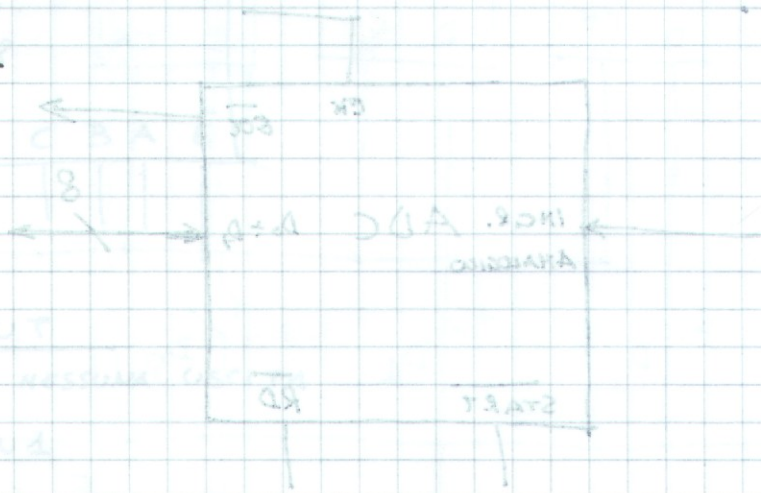
```
CALL   RITARD
; aspetta un minuto
JP , E2
```

PROGRAMMA 50

Sommare insieme gli elementi di posto corrispondente di 3 tabelle che iniziano rispettivamente agli indirizzi 1A00H, 1B00H, 1C00H; i risultati vanno posizionati in memoria a partire dall'indirizzo 1D00H. La lunghezza di queste 3 tabelle è contenuta nella locazione d'indirizzo 1000H



```
        ORG XXXX
        LD B, (1000H)
        LD HL, 1A00H
        LD IX, 1B00H
        LD IY, 1C00H
        LD DE, 1D00H
ET:     LD A, (HL)
        ADD A, (IX)
        ADD A, (IY)
        LD (DE), A
        INC HL
        INC IX
        INC IY
        INC DE
        DJNZ, ET
        HALT.
```



PROGRAMMA 49

Scrivere un sottoprogramma che esegue l'estrazione della radice quadrata approssimata all'intero più piccolo.

L'algoritmo è il seguente: 1) si pone il risultato R ad 1 ed un contatore C ad 1, si

2) si sottrae a C ad N

3) se il risultato è ≥ 0, incremento R di 1, e C di 2 e ritorno al punto 2, altrimenti l'operazione si ferma ed R è il risultato.

ESEMPIO

Calcolare la radice di N = 35

R = 1, C = 1, N = 35

N = N - C = 35 - 1 = 34

$N \geq 0$ quindi R = 2, C = 3 e ripeto

N = N - C = 34 - 3 = 31

$N \geq 0$ quindi R = 2 e C = 5 e ripeto

N = N - C = 31 - 5 = 26

$N \geq 0$ quindi R = 3 C = 7 e ripeto

N = 26 - 7 = 19 > 0

$N \geq 0 \Rightarrow$ R = 4 C = 9

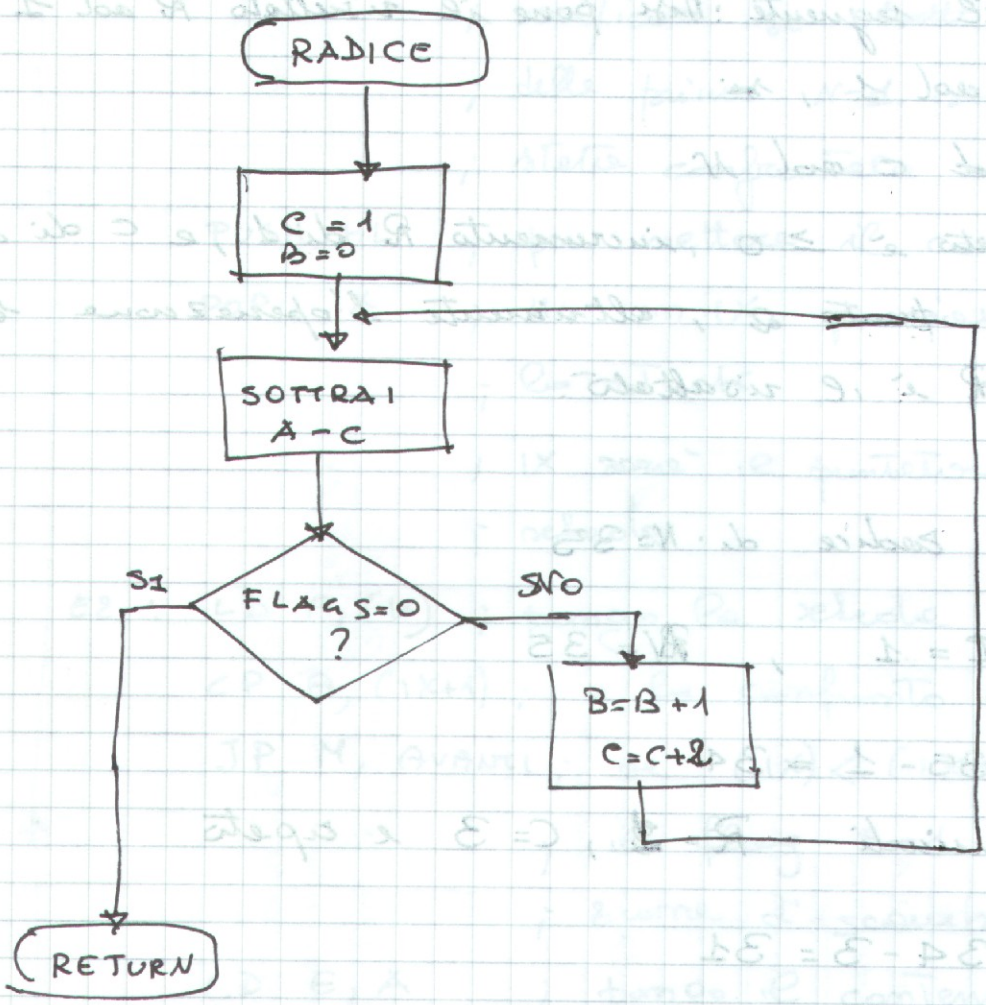
N = 19 - 9 = 10

$N \geq 0 \Rightarrow$ R = 5, C = 11

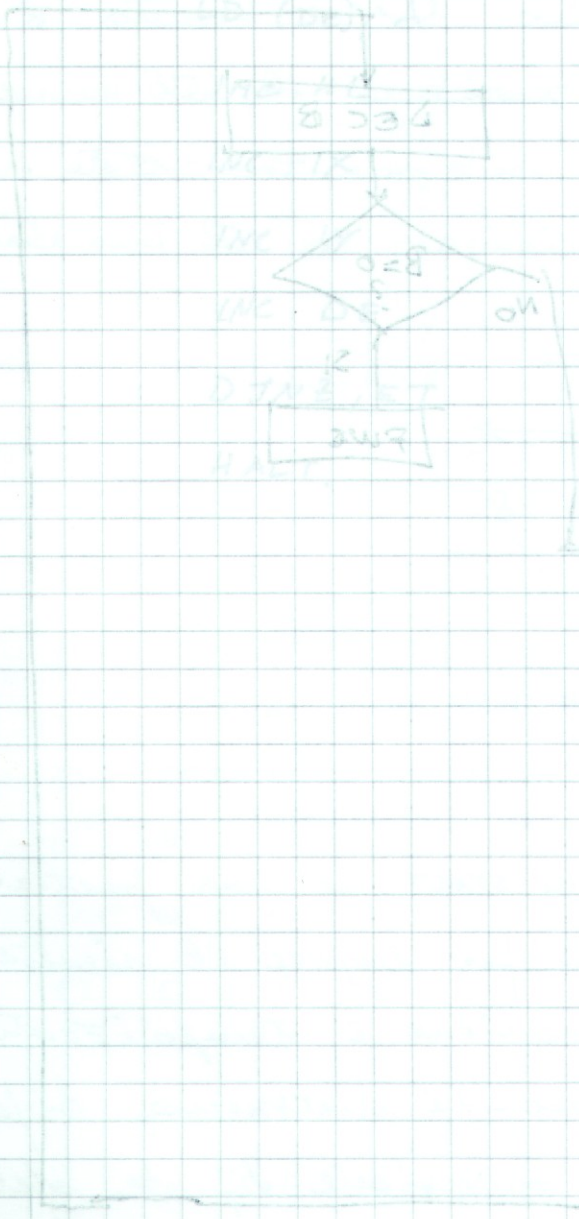
N = 10 - 11 = -1

N < 0 il programma si blocca

Suppongo che il numero di cui eseguire la radice sia un intero
col 8 bit senza segno presente in A all'atto della chiamata
del sottoprogramma, e che porremo il risultato in B



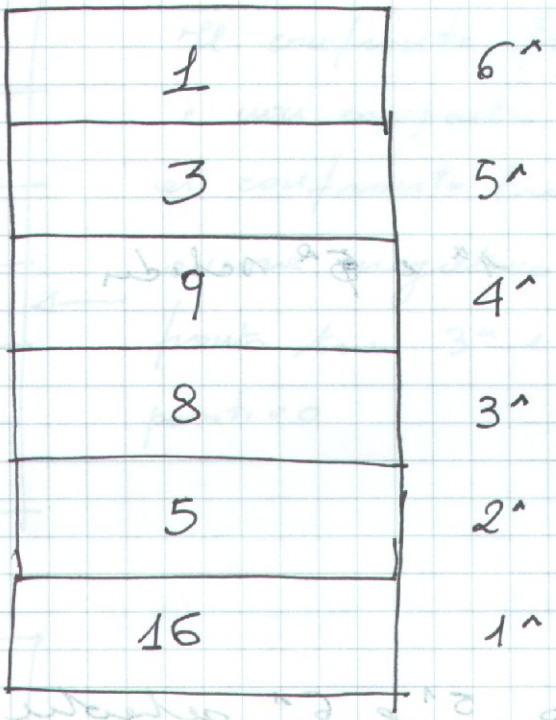
```
RADICE :  LD C, 01H
          LD B, 00H
ET :      SUB A, C
          JP M, FINE
          INC B
          INC C
          INC C
          JP, ET
FINE :    RET
```



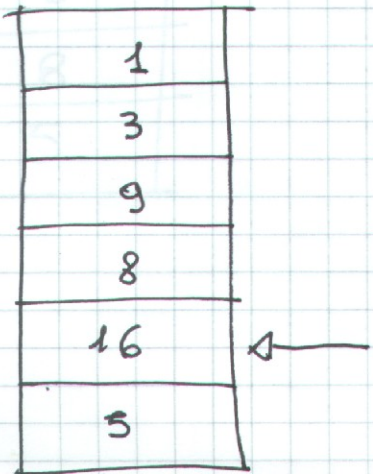
PROGRAMMA 48

Scrivere un sottoprogramma che ordini in senso crescente un'area di memoria, il cui indirizzo di partenza è contenuto in H+L e la cui ampiezza è contenuta nel registro B.

Possiamo utilizzare la tecnica del bubble-sort che illustriamo con un esempio. Supponiamo di voler ordinare in senso crescente la seguente pila di 6 schede



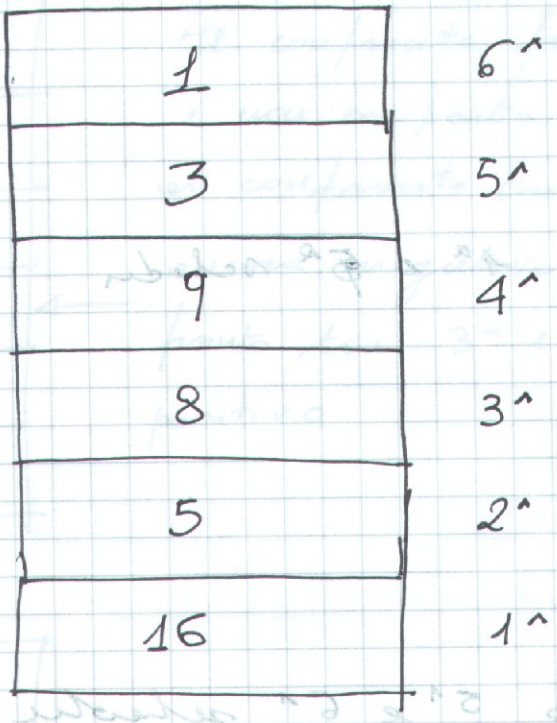
Puntiamo sulla prima scheda e chiediamoci se il suo contenuto è superiore a quello della seconda scheda. Se la risposta è sì scambiamo ~~le~~ di i due contenuti.



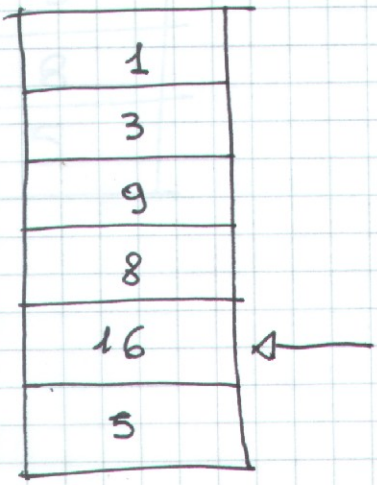
Puntiamo ora sulla seconda scheda e ci chiediamo se il suo contenuto è superiore a quello della successiva; se la

area di memoria, il cui indirizzo di partenza è contenuto in H L e la cui ampiezza è contenuta nel registro B.

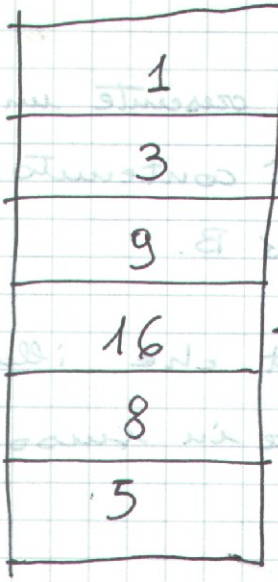
Possiamo utilizzare la tecnica del bubble-sort che illustriamo con un esempio. Supponiamo di voler ordinare in senso crescente la seguente pila di 6 schede



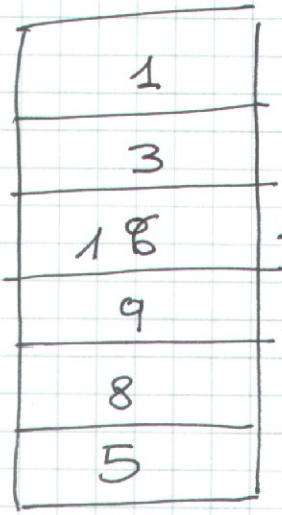
Puntiamo sulla prima scheda e chiediamoci se il suo contenuto è superiore a quello della seconda scheda. Se la risposta è si scambiamo ~~le~~ di i due contenuti.



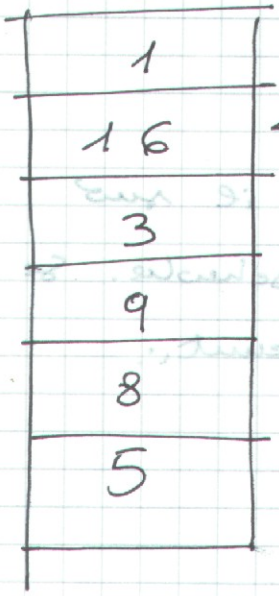
puntiamo ora sulla seconda scheda e ci chiediamo se il suo contenuto è superiore a quello della successiva; se la risposta è si effettuo lo scambio



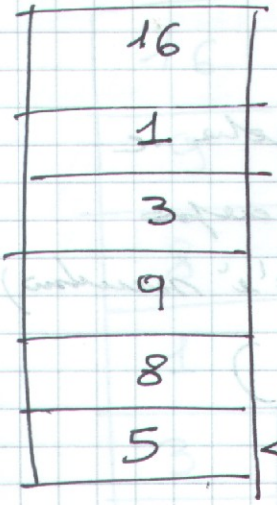
← effettuiamo lo stesso ragionamento confrontando 3ª e 4ª scheda



← confrontiamo 4ª e 5ª scheda

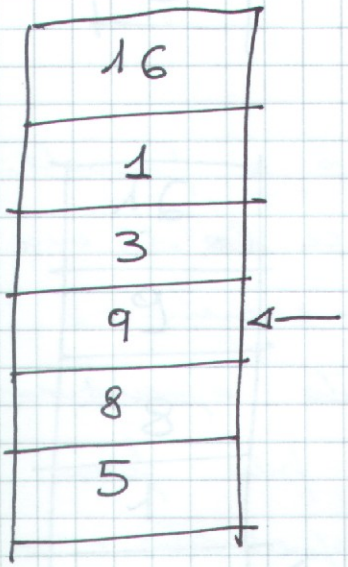


← confrontiamo 5ª e 6ª scheda

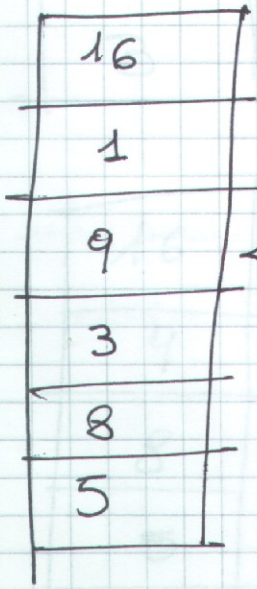


Ora il numero più grande fra quelli presenti nelle schede è sicuramente finito in cima alla pila.

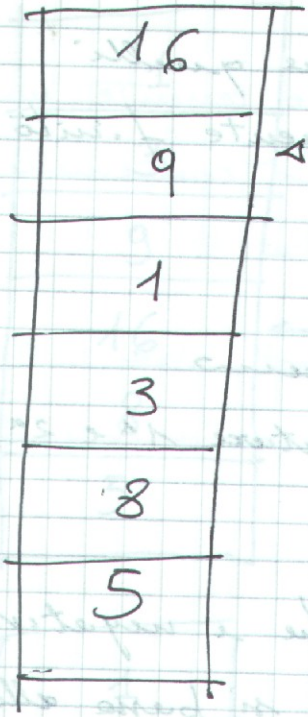
Per ordinare la pila dobbiamo ritornare all'inizio e confrontare 1ª e 2ª scheda.



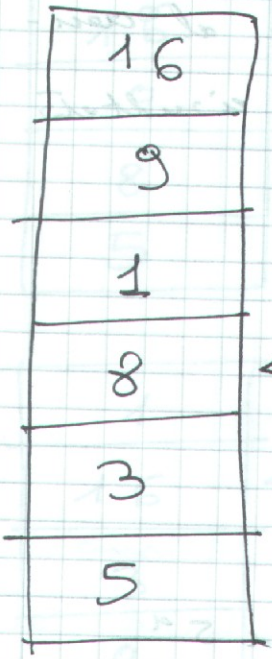
Il confronto fra 1ª e 2ª scheda è negativo e non comporta lo scambio; si passa allora al confronto fra 2ª e 3ª scheda che è ancora negativo. Allora si passa al confronto fra 3ª e 4ª scheda che dà risultato positivo



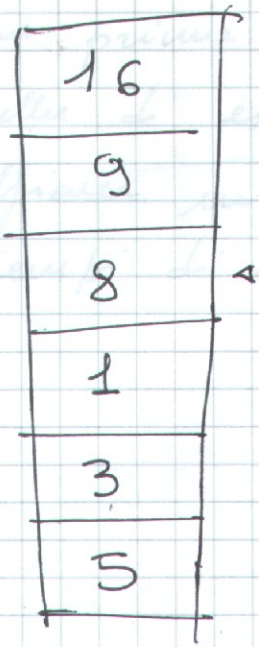
anche il confronto fra 4ª e 5ª scheda è positivo e ci fa fare lo scambio



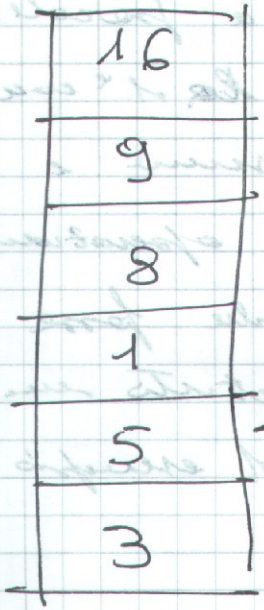
← il confronto tra 5ª e 6ª scheda è negativo e ricominciamo da capo confrontando 1ª e 2ª scheda (non c'è scambio) 2ª e 3ª (non c'è scambio) ~~3ª e 4ª~~



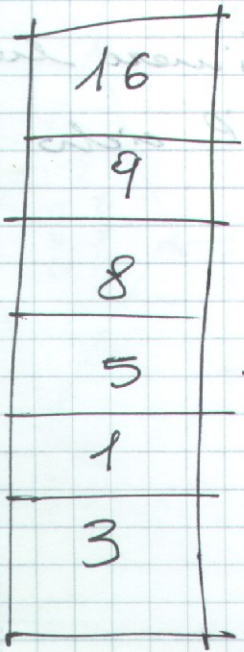
← fra 3ª e 4ª c'è scambio



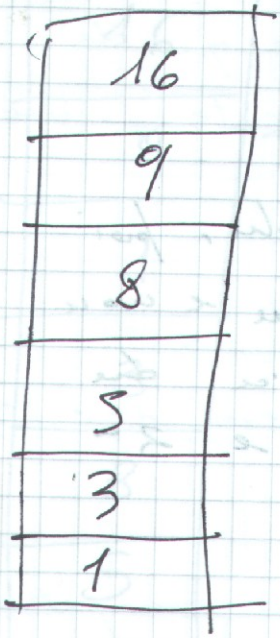
confrontiamo 4ª e 5ª chiave, poi la 5ª e la 6ª, ~~infine la~~ e non c'è scambio. Si ricomincia da capo riconfrontando 1ª e 2ª



anche il confronto tra la 2ª e 3ª comporta lo scambio

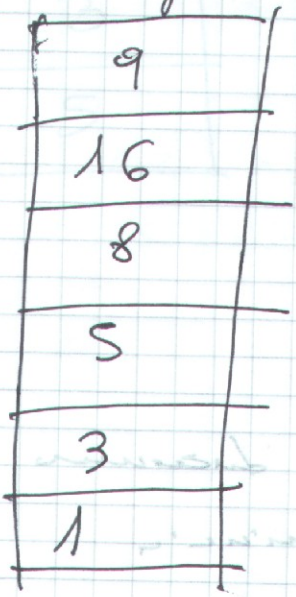


tutti gli altri confronti danno esito negativo e si ricomincia da capo



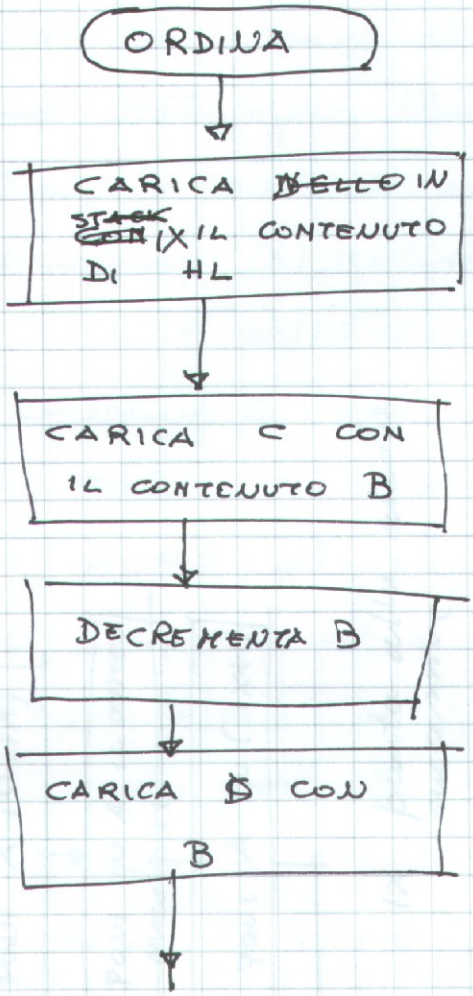
Dopo l'ultima pressata la lista è ordinata.

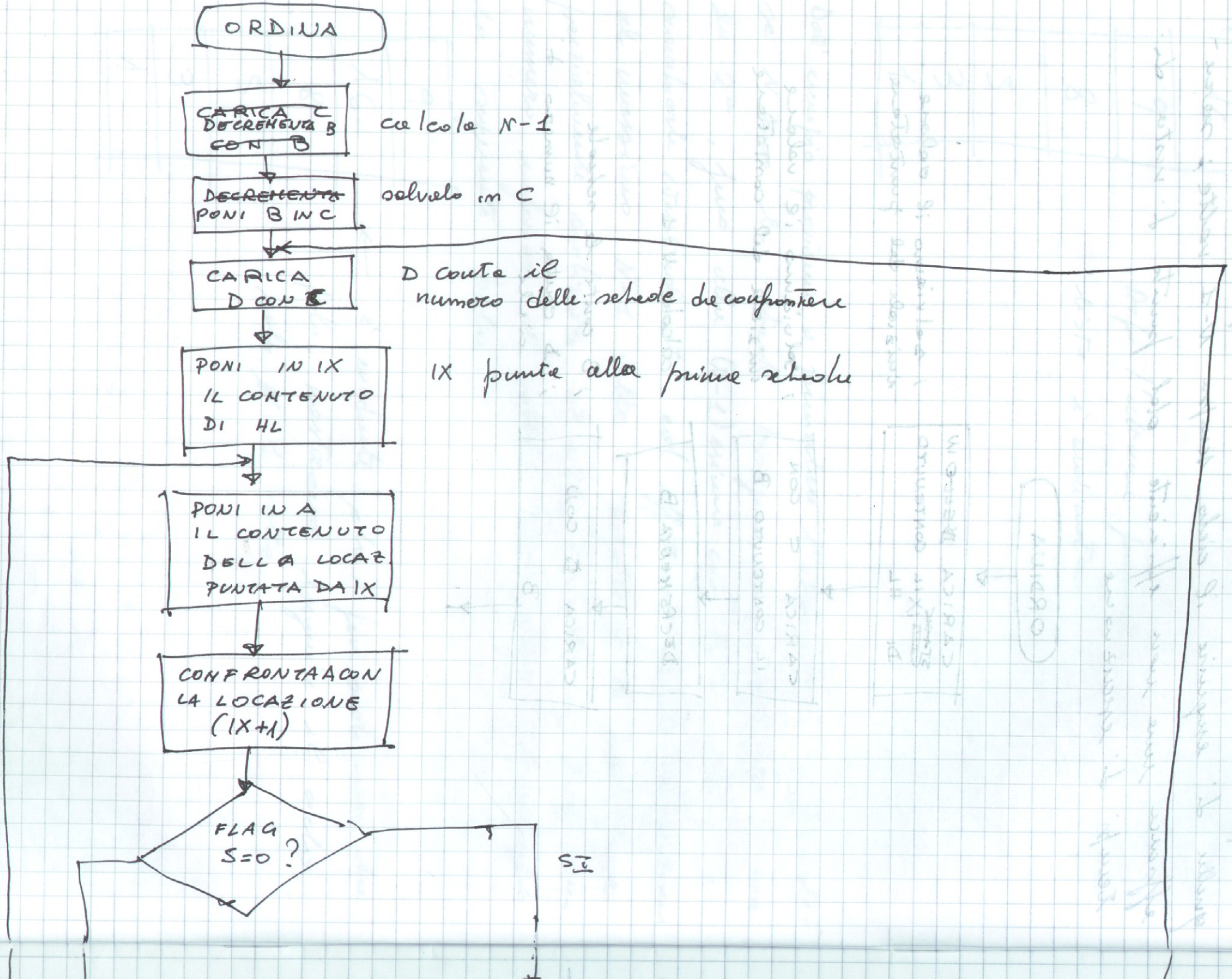
Dall'esempio proviamo a formulare l'algoritmo a parole se N sono le schede bisogna confrontare la 1ª con la 2ª, poi la 2ª con la 3ª fino alla (N-1)esima con la N-esima e scambiarle eventualmente, ed effettuare queste operazioni al massimo N-1 volte. Tuttavia se la pila fosse parzialmente ordinata il ciclo andrebbe ripetuto un numero inferiore di volte. Prendiamo ad esempio la seguente pila

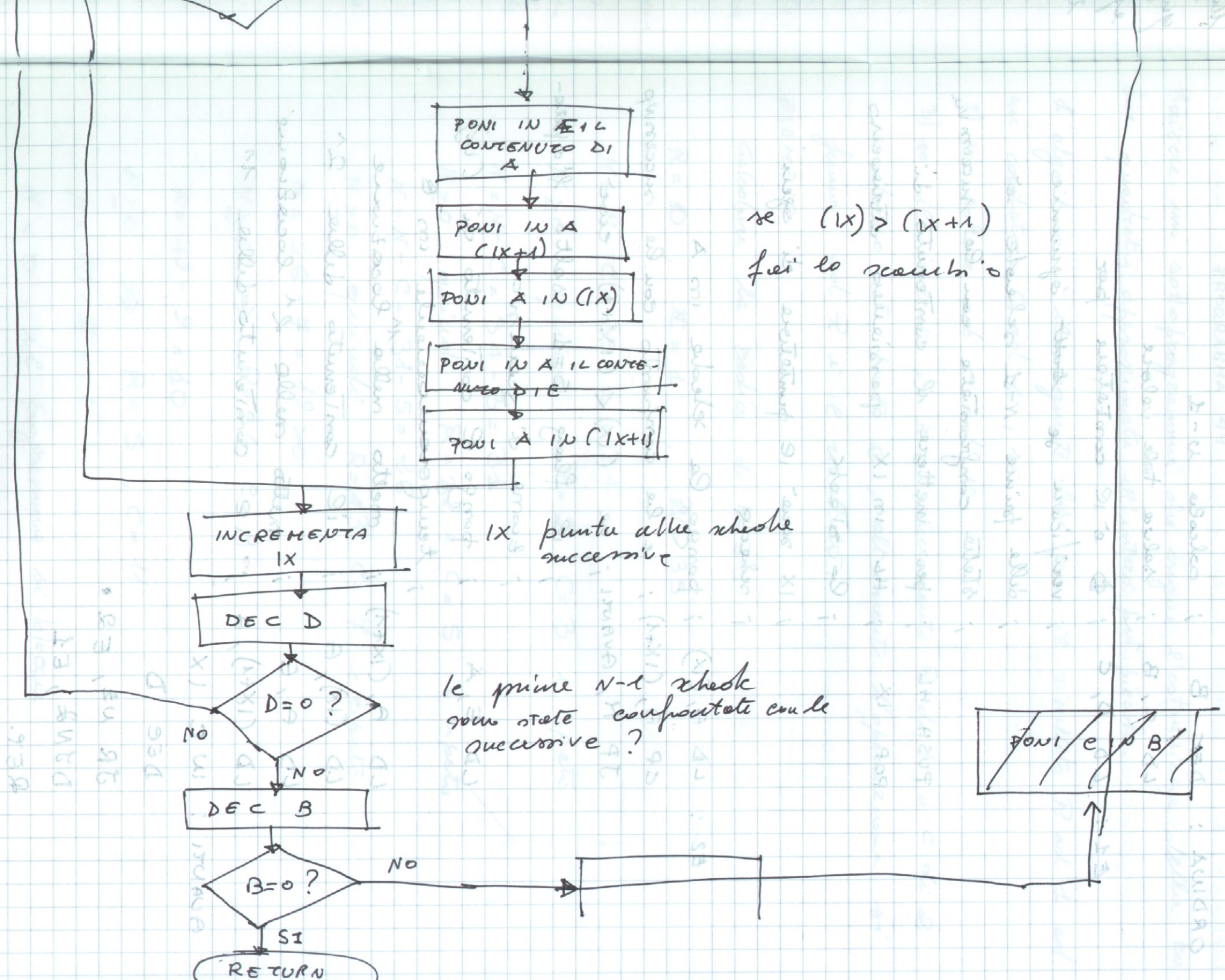


← è evidente che, per ordinare la pila, basterà eseguire il ciclo solo 1 volta.

Una prima implementazione dell'algoritmo, può essere quella di eseguire il ciclo sempre N-1 volte; sarà efficace ma non efficiente dal punto di vista dei tempi di esecuzione




ORDINA
CARICA C DECREMENTATA B CON B
calcola N-1
DECREMENTA PONI B IN C
salvalo in C
CARICA D CON B
D conta il numero delle schede da confrontare
PONI IN IX IL CONTENUTO DI HL
IX punta alla prima scheda
PONI IN A IL CONTENUTO DELLA LOCAZ. PUNTATA DA IX
CONFRONTA A CON LA LOCAZIONE (IX+1)
FLAG S=0 ?
SI


PONI IN A IL CONTENUTO DI A
PONI IN A (IX+1)
PONI A IN (IX)
PONI IN A IL CONTENUTO DI E
PONI A IN (IX+1)
se (IX) > (IX+1)
fai lo scambio
INCREMENTA IX
IX punta alla scheda successiva
DEC D
D=0 ?
NO
NO
DEC B
B=0 ?
NO
SI
RETURN
le prime N-1 schede sono state confrontate con le successive ?
PONI E IN B

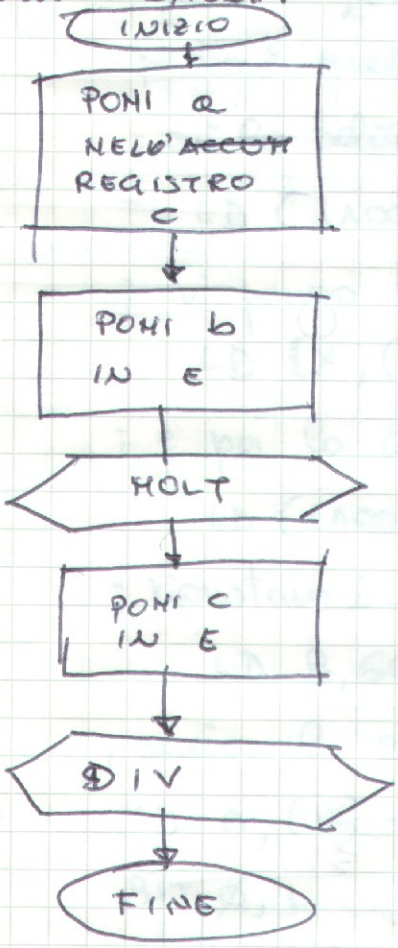
```
ORDINA : DEC  B      ; calcolo n-1
         LD   C, B   ; salvo tale valore
E1 :     LD   D, C   ; D è il contatore per
                     ; verificare se ogniuno
                     ; delle prime N-1 schede è
                     ; stata confrontata con la successiva
         PUSH HL     ; per mettere il contenuto di
         POP  IX     ; HL in IX possiamo attraverso
                     ; lo stack
                     ; IX sarà il puntatore di ogni
                     ; scheda
E2 :     LD   A, (IX)   ; pongo la scheda in A
         CP   A, (IX+1) ; e la confronto con la successiva
         JP   M, AVANTI ; se (IX) < (IX+1) cioè
                        ; il flag S=1 salto l'opera-
                        ; zione di scambio
         LD   E, A      ; pongo il contenuto di (IX)
                        ; temporaneamente in E
         LD   A, (IX+1) ; metto nella 1^ locazione
         LD   (IX), A   ; il contenuto della 2^
         LD   A, E      ; metto nella 2^ locazione
         LD   (IX+1), A ; il contenuto della 1^
AVANTI : INC  IX
         DEC  D
         JR   NZ, E2
         DJNZ, E1
         RET.
```

PROGRAMMA 47.

Si ha a disposizione il sotto programma DIV che effettua la divisione fra il contenuto del registro BC e quello del registro E e pone il risultato in C, e il sotto programma HOLT che effettua la moltiplicazione fra i registri C ed E e pone il risultato in BC, effettuare l'operazione

$$\frac{a \cdot b}{c}$$

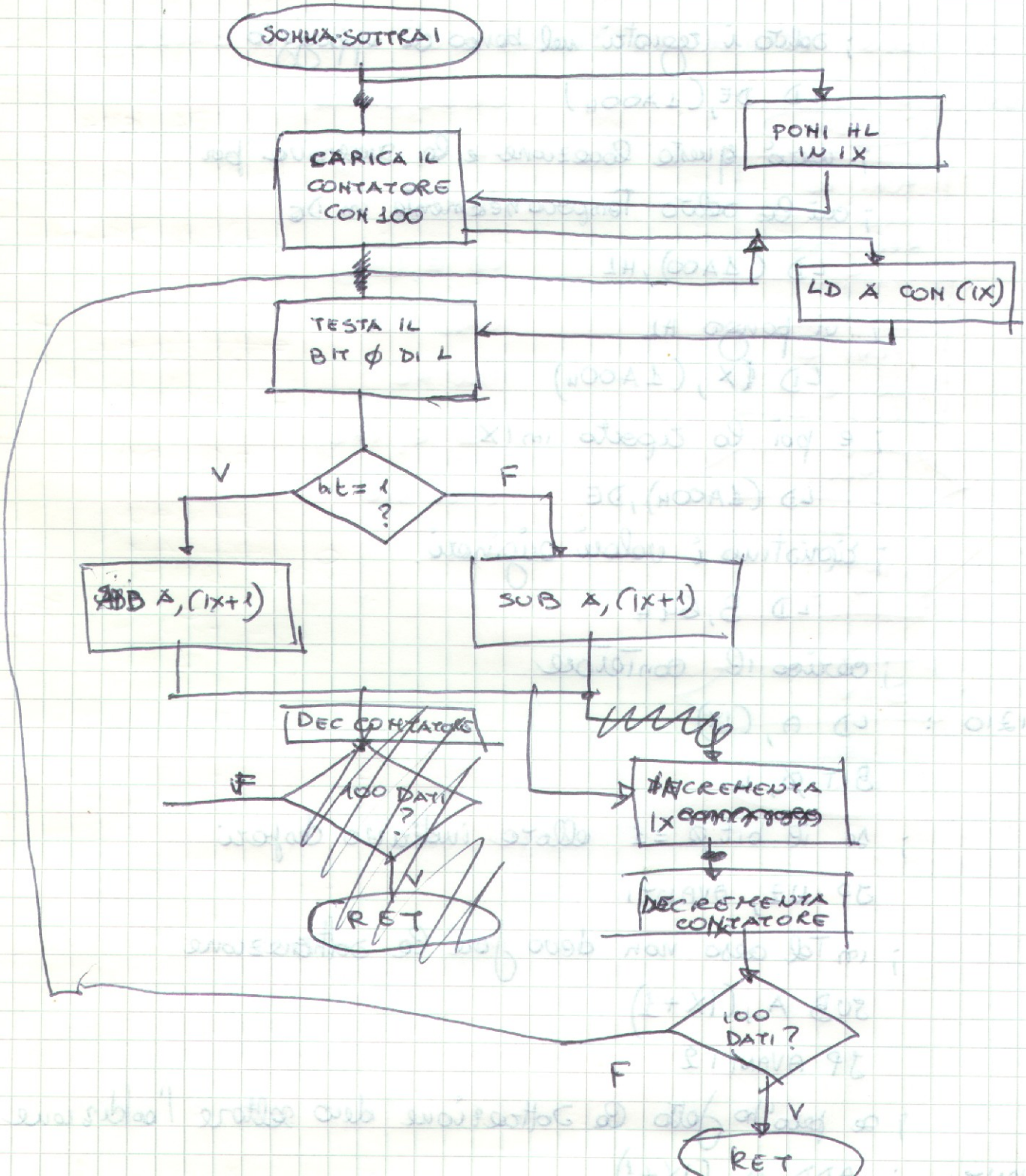
dove a si trova all'indirizzo $1A00_H$, b all'indirizzo $1A01_H$, e all'indirizzo $1A02_H$.




```
LD HL, 1A00H
LD C, (HL)
INC HL
LD E, (HL)
CALL HALT
INC HL
LD E, (HL)
CALL DIV
HALT.
```

PROGRAMMA 46

Si hanno cento dati in memoria a partire dalla locazione ~~1000H~~ contenuta in HL: ogni dato va sostituito dalla somma fra se stesso e il successivo, se occupa una locazione d'indirizzo pari o dalla differenza fra se stesso e il successivo se occupa una locazione d'indirizzo dispari. Scrivere un sottoprogramma che effettui tale operazione.



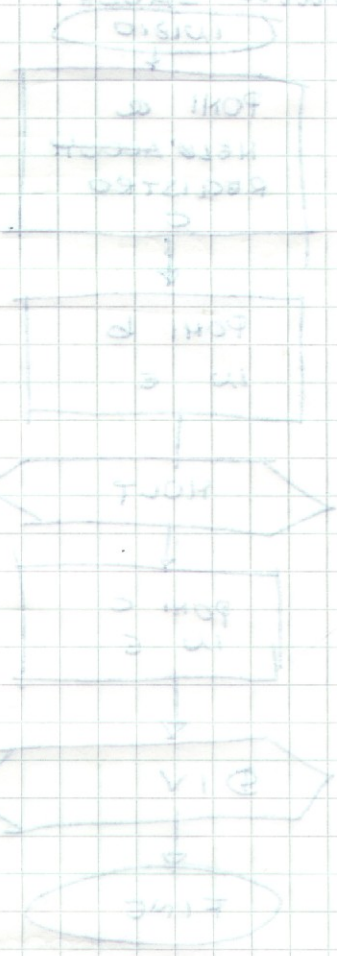
```
SOMMA-SOTTRAI:  EX (SP), HL
                ; pongo HL in testa allo stack
                EX (SP), IX
                ; e lo posto in IX

SOMMA-SOTTRAI:  EX AF, AF'
                EXX
                ; salvo i registri nel banco di appoggio
                LD DE, (1A00H)
                ; uso questa locazione e la successiva per
                ; cui la salvo temporaneamente in DE
                LD (1A00), HL
                ; vi pongo HL
                LD IX, (1A00H)
                ; e poi lo sposto in IX
                LD (1A00H), DE
                ; ripristino i valori originari
                LD B, 64H
                ; uso il contatore
INIZIO:         LD A, (IX)
                BIT 0, L
                ; se il bit 0 = 1 allora indirizzo dopo
                JP NZ, AVANTI
                ; in tal caso non devo fare la sottrazione
                SUB A, (IX+1)
                JP AVANTI2
                ; se però ho fatto la sottrazione devo settare l'addizione
AVANTI:         ADD A, (IX+1)
```

```
                EX (SP), IX
                ; e lo porto in IX
SOMMA-SOTTRAI : LD EX AF, AF'
                EXX
                ; salvo i registri nel banco di appoggio
                LD DE, (1A00H)
                ; uscirò questa locazione e la successiva per
                ; cui le salvo temporaneamente in DE
                LD (1A00), HL
                ; vi pongo HL
                LD IX, (1A00H)
                ; e poi lo riporto in IX
                LD (1A00H), DE
                ; ripristino i valori originari
                LD B, 64H
                ; carico il contatore
INIZIO :        LD A, (IX)
                BIT 0, L
                ; se il bit 0 = 1 allora indirizzo dispari
                JP NZ, AVANTI
                ; in tal caso non devo fare la sottrazione
                SUB A, (IX+1)
                JP AVANTI2
                ; se però ho fatto la sottrazione devo settare l'addizione
AVANTI :        ADD A, (IX+1)
AVANTI2 :       INC IX
                INC HL
```

DJNZ , 0000 0010

RET.



PROGRAMMA 45

In un ciclo continuo il µP preleva 4 dati alla volta dalla porta 20H, ne fa la somma a 8 bit e invia il risultato alla porta 30H, Interrompere il programma se c'è overflow

```
        ORG XXXX
        LD C,(20H)
INIZIO: XOR A          ; pone A=0
        LD B,04H       ; contatore
        IN B,(C)
INDIETRO: ADD A,B      ; poi di volta in volta la somma
        JP PO, FINE
        DJNZ, INDIETRO
        OUT(30H),A     ; fatta la somma si invia in uscita
        JP, INIZIO
; salto incondizionato poiché è un ciclo
; continuo
FINE: HALT
```

Programma 44

Scrivere un programma che effettui la somma tra due operandi composti ciascuno da 4 byte, registrati in memoria a partire, rispettivamente, dagli indirizzi 1A00H e 1A06H. immagazzinando il riporto al posto del primo dato

```
ORG xxxx
LD B, 04H      ; contatore dei byte
LD DE, 1A00H
LD HL, 1A06H
AND A          ; azzera il flag di carry
LOOP: LD A, (DE)
ADC A, (HL)    ; somma i due byte più l'eventuale
               ; riporto della somma precedente
LD (DE), A     ; le istruzioni che seguono
INC DE         ; non modificano il flag di carry
INC HL
DJNZ, LOOP
HALT.
```

PROGRAMMA 45

Il µP preleva dati da una porta di indirizzo 20H; questi dati sono il risultato della conversione digitale delle temperature rilevate da un sensore. I dati vanno rilevati ogni minuto e immagazzinati in memoria. Se il dato è superiore a 8F, ciò indica una situazione di pericolo per cui il µP attiva una sirena scrivendo un dato qualsiasi all'indirizzo di I/O 30H. In ogni caso ~~ogni volta~~ quando abbiamo soltanto un'interruzione o un'altra esterna che costringe il µP ad inviare alla porta di uscita 40H i dati accumulati

Supponiamo di operare con un'unica interruzione per cui la possiamo ~~applicare~~ gestire in modo 1 (indirizzo 0038H)

```
~~LOOP~~   ORG XXXX
~~dato~~   LD HL, 1800H    ; suppongo che il buffer che conterrà
                           ; ~~questi~~ i dati del sensore parta da
                           ; questa locazione. Questa istruzione occupa 3 byte
           ~~IN A, (20H)~~
~~ORG~~    IM 1            ; imposta il modo 1 (2 byte)
           ~~IN A, (20H)~~ ; ~~preleva il dato (2 byte)~~
INIZIO     LD DE, 102      ; questo ciclo genera un ritardo
LOOP1      LD BC, 32767    ; approssimativo di 60 s
LOOP2      DEC BC          ; occupa 12 byte di memoria
           JR NZ, LOOP2
           DEC DE
           JR NZ, LOOP1
           IN A, (20H)     ; preleva il dato (2 byte)
           CP 8FH          ; controlla se è troppo alto (2 byte)
           JR NC, AVANTI   ; se è più piccolo o uguale a 8F
                           ; allora è posto a zero (2 byte)
```

```
        OUT (30H), A        ; altrimenti avvia la scrittura (2 byte)
AVANTI: JP INIZIO           ; apri il file (3 byte)
AVANTI: LD (HL), A          ; memorizza il dato (1 byte)
        INC HL              ; 1 byte
        JP INIZIO           ; 3 byte
```

Poiché sono 29 byte non si arriva alla locazione 0038H

```
        ORG 0038H           ; routine gestione interruzione
        DEC HL              ; punta alla locazione non riempita
        LD BC, 1800H        ; punta all'inizio
        XOR A               ; poni il flag di carry a zero
        SBC HL, BC          ; ora in HL c'è il numero di dati accumulati
        LD B, H             ; poniamo questo dato in BC che possiamo
        LD C, L             ; usare come contatore
        LD HL, 1800H        ; fai puntare HL all'inizio della zona in
                            ; cui sono accumulati i dati
ET      LD A, (HL)          ; preleva il dato
        OUT (40H), A        ; spedisce in uscita
        INC HL              ; incrementa il puntatore
        DEC BC
        JR NZ, ET
        RETI                ; ritorna da interruzione mascherabile.
        JR NZ, ET           ; se ci non è ancora nullo salta all'inizio
        LD C, FFH           ; altrimenti riporta C al valore massimo
        DJNZ, ET            ; decrementa B e salta all'indietro
                            ; se B ≠ 0 cioè BC ≠ 0.
```

PROGRAMMA 39

Si hanno 100 dati immagazzinati in memoria a partire da 1400H fare in modo che il nibble inferiore di ogni dato sia sostituito dalla stringa 1111

PROGRAMMA 40

Si prelevano dati dalla porta 20H e si scrivono in memoria dall'indirizzo 1B00, non si sa a priori quanti dati si devono prelevare; il prelievo dei dati termina se il dato ricevuto è 00H

PROGRAMMA 41

Si ha a disposizione un sottoprogramma QUAD che prende il dato che si trova nel registro C e pone in DE il quadrato. Si ha inoltre a disposizione il sottoprogramma RAD che prende il contenuto del registro BC, ne fa la radice e pone il risultato in C. Usare questi sottoprogrammi per effettuare la seguente operazione

$$\sqrt{a^2+b^2}$$

dove a è il contenuto della locazione 1B00, b è il contenuto di 1B01.

PROGRAMMA 42

Si ha un dato formato da 6 byte memorizzato in memoria dalla locazione 1A00, alla locazione 1A05. Si effettui una rotazione a sinistra del dato complessivo 5 volte.

PROGRAMMA 39

```
        LD B, 64H
        LD HL, 1400H
CICLO:  LD A, (HL)
        OR 0FH ; setto le nibble inferiore
        LD (HL), A
        INC HL
        DJNZ, CICLO
```

PROGRAMMA 40

```
        LD  HL, 1B00H
INIZIO: IN  A, (20H)
        CP  00H
        JP  Z, FINE
        LD  (HL), A
        INC HL
        JP  INIZIO
        HALT
```

PROGRAMMA 41

```
LD A, (1B00H)
LD C, A
CALL QUAD   ; ora DE contiene a^2
LD H, D
LD L, E     ; ora HL contiene a^2
LD A, (1B01H)
LD C, A
CALL QUAD   ; ora DE contiene b^2
AND A       ; azzera il carry
SBC HL, DE  ; ora HL contiene a^2 - b^2
LD B, H
LD C, L     ; ora BC contiene a^2 - b^2
CALL RAD
HALT
```

PROGRAMMA A2

```
        LD  D, 05H
CICLOEXT: LD  C, 08H

        LD  HL, 1A00H

        BIT 7, (HL)     ; teste il bit + significativo
        JP NZ, AVANTI   ; e se è nullo
        AND A           ; azzera il flag di carry
        JP AVANTI2
AVANTI: SCF             ; se il bit è 1 setta il flag di carry
AVANTI2: LD B, 06H      ; conto i byte
        LD HL, 1A00H
CICLO:  SRA (HL)        ; sposto il primo byte
                        ; e infilo dentro il bit meno
        INC HL          ; significativo dato dal carry
        DJNZ, CICLO
        DEC C
        JP NZ, CICLOEXT ; bisogna farlo 8 volte
        DEC D
        JP NZ, CICLOEXT ; abbiamo fatto 5 colonne?

        HALT
```

PROGRAMMA 38

Scrivere un programma che legga un dato dalla porta d'indirizzo 80H ~~ciclo ogni minuto~~, se il dato è uguale a zero viene scartato, altrimenti viene scritto in memoria a partire dall'indirizzo 1B00H; arrivati a 100 dati, si mandano i dati raccolti alla porta di indirizzo 30H e il ciclo riprende.

```
        ORG XXXX
INIZIO  LD C, 20H
        LD HL, 1B00H
        LD ~~B, 100~~ LD B, 100
        ~~LD C, 20H~~
LOOP    IN A, (C)
        JR Z, ~~XXXXXX~~ LOOP   ; se il dato prelevato è = 0 prelevo
                                ; un'altro dato
        LD (HL), A              ; altrimenti scrivilo in memoria
        INC HL
        DJNZ, LOOP              ; vedi se sono 100 dati raccolti
        DEC HL                  ; fai puntare HL all'ultimo dato inserito
        LD C, 30H
        LD B, 100
        OTDR                    ; invia i 100 dati in uscita
        JP INIZIO
```

PROGRAMMA 37

Si ha un dato scritto in memoria su 4 byte della locazione 1B00 alla locazione 1B03; controllare un bit alla volta di questo dato a partire dal più significativo, se il bit è uguale ad 1 effettuare la somma fra i registri A e B, altrimenti effettuare la somma fra A e C. Ogni somma va memorizzata a partire dall'indirizzo 1B04.

```
        LD E, 04H   ; quattro byte da controllare
        LD HL, 1B03H ; byte più significativo
CICLO:  LD B, 08H   ; otto bit da controllare
        SLA (HL)    ; shifta a sinistra
                    ; in modo che il bit + significativo
                    ; vada nel carry
        JP C, AVANTI
        ADD A, C    ; se FC = 0  effettua A + C
        JP POI
AVANTI: ADD A, B    ; se FC = 1  effettua A + B
POI:    DJNZ, CICLO ; controlla gli 8 bit
        DEC HL      ; passa all'altro byte
        DEC C
        JP NZ, CICLO
        HALT
```

PROGRAMMA 36

Scrivere un programma che prelevi 20 dati dalla porta d'indirizzo 20_H e li scriva in memoria a partire dalla locazione 1A00. Successivamente verificare se c'è qualche dato pari a $3F_H$, se la risposta è negativa inviare i dati alla porta 80_H e mettere a zero le locazioni di memoria utilizzate

```
        LD C, 20H
        LD HL, 1A00H
        LD B, 14H
        INIR
        LD A, 3FH   ; carico il dato da cercare
        DEC HL      ; durante il caricamento dei dati
                    ; si era spostato già avanti di una
                    ; posizione
        CPDR        ; controlla tutti i dati
                    ; al termine se il flag Z=1
                    ; c'è un dato pari a 3FH
        JP NZ, FINE ; se si termina programma
        INC HL      ; CPDR ha portato HL troppo indietro
        LD C, 80H   ; indirizzo porta di uscita
        LD B, 14H
        XOR A       ; azzera A
CICLO : LD D, (HL)
        OUT (C), D
        LD (HL), A
        INC HL
        DJNZ, CICLO
```

PROGRAMMA 35

Si prelevano dati in modo continuo dalla porta d'indirizzo 20_H, prelevati due dati se ne fa la somma e la si pone in un byte in memoria a partire dalla locazione 1800_H e il ciclo ripreude, si esce dal programma se viene prelevato uno zero dalla porta o si ha errore di overflow nell'addizione.

```
        ORG XXXX
        LD HL, 1800H
        IN A,(200H)  LD C, 20H
INIZIO  LD B,A       IN A, (C)   ; si deve usare questa istruzione
        IN A,(20H)               ; perchè modifica ie flag
        ADD A,B      JR Z, FINE  ; salta alla fine se si è prelevato
                                 ; un dato nullo
        IN B, (C)
        JR Z, FINE
        ADD A, B     ; si sommano i due dati
        JP PE, FINE  ; controllare se vi è stato overflow
        LD (HL), A   ; se non vi è problema memorizzare
                     ; la somma
        INC HL
        JP INIZIO
FINE    HALT
```

PROGRAMMA 34

Si hanno 100 dati scritti in memoria a partire dalla locazione 1A00; di ogni dato va analizzato ogni bit a partire da quello meno significativo; se il bit è uguale ad 1 si prelevano dati dalla porta di indirizzo 20_H e si memorizzano a partire dalla locazione $1B00_H$ altrimenti non si fa niente

```
        ORG XXXX

        LD HL, 1A00H    ; punta ai dati da controllare bit a bit
        LD DE, 1B00H    ; punta alle locazioni da riempire
        LD B, 64H       ; contatore dei dati da controllare

        ~~LD C, 8~~

LOOP1   LD A, (HL)      ; preleva il dato
        LD C, 8         ; conta i bit controllati
LOOP2   RRCA            ; manda il bit nel flag di carry
        JR NC, AVANTI   ; se CF=0 non fare niente altrimenti
        EX AF, AF'      ; se è CF=1, salva A
        IN A, (20H)     ; preleva il dato dalla porta
        LD (DE), A      ; mettilo in memoria
        EX AF, AF'      ; recupera A
        INC DE          ; incrementa il puntatore alla locazione da riempire
AVANTI  DEC C           ; controlla se abbiamo controllato tutto il byte
        JR NZ, LOOP2    ; se no torna indietro
        INC HL          ; se sì punta al byte successivo
        DJNZ, LOOP1
        HALT
```

PROGRAMMA 33

Il µP legge continuamente un byte inviato dalla porta di indirizzo 20H; i bit 0, 1, 2, 3 sono uscite di sensori ON-OFF i quali, quando sono ad 1 indicano che vi è stata una intrusione in locali da essi controllati; i restanti bit vengono da sensori antincendio e ad 1 indicano che vi è un pericolo di incendio; il µP ha a disposizione una sirena che vede come porta di indirizzo ~~8004~~ 30H, se il µP scrive un dato qualsiesi a questo indirizzo la sirena si aziona, si ha inoltre a disposizione un sistema antincendio che si aziona scrivendo un dato qualsiesi all'indirizzo di I/O 40H; si vuole azionare (solo) la sirena se è ~~il sistema~~ se almeno uno dei sensori antintrusione ad 1 mentre si vuole azionare la sirena e il sistema antincendio se almeno due sensori antincendio sono ad 1.

```
        ORG XXXX
INIZIO  IN A, (20H)
        LD C, 0          ; C conta i sensori antincendio ad 1
        LD B, 4          ; controlla i sensori antintintrusione
LOOP1   RRCA
        JR C, SIRENA     ; suona la sirena se un bit è ad 1
        DJNZ, LOOP1
        LD B, 4          ; controlla i sensori antincendio
LOOP2   RRCA
        JR NC, AVANTI    ; se il bit è a zero non si incrementa il contatore
        INC C
AVANTI  EX AF, AF'       ; salva A
        LD A, C          ;
        CP 02H
        JP Z, INCENDIO   ; se il contatore è ad 2 azione
```

EX AF, AF'

AVANTI O DJNZ, LOOP2

~~SIRENA~~

```
INCENDIO  OUT(40H), A    ; azione il sistema antincendio
SIRENA    OUT(30H), A    ; e/o la sirena
          JP INIZIO      ; riprendi il controllo
```

PROGRAMMA 33

Il µP legge continuamente un byte inviato dalla porta di indirizzo 20H; i bit 0, 1, 2, 3 sono uscite di sensori ON-OFF i quali, quando sono ad 1 indicano che vi è stata una intrusione in locali da essi controllati; i restanti bit vengono da sensori antincendio e ad 1 indicano che vi è un pericolo di incendio; il µP ha a disposizione una sirena che vede come porta di indirizzo ~~8004~~ 30H, se il µP scrive un dato qualsiesi a questo indirizzo la sirena si aziona, si ha inoltre a disposizione un sistema antincendio che si aziona scrivendo un dato qualsiesi all'indirizzo di I/O 40H; si vuole azionare (solo) la sirena se è ~~il sistema~~ se almeno uno dei sensori antintrusione ad 1 mentre si vuole azionare la sirena e il sistema antincendio se almeno due sensori antincendio sono ad 1.

```
        ORG XXXX
INIZIO  IN A, (20H)
        LD C, 0          ; C conta i sensori antincendio ad 1
        LD B, 4          ; controlla i sensori antintintrusione
LOOP1   RRCA
        JR C, SIRENA     ; suona la sirena se un bit è ad 1
        DJNZ, LOOP1
        LD B, 4          ; controlla i sensori antincendio
LOOP2   RRCA
        JR NC, AVANTI    ; se il bit è a zero non si incrementa il contatore
        INC C
AVANTI  EX AF, AF'       ; salva A
        LD A, C          ;
        CP 02H
        JP Z, INCENDIO   ; se il contatore è ad 2 azione
```

EX AF, AF'

AVANTI O DJNZ, LOOP2

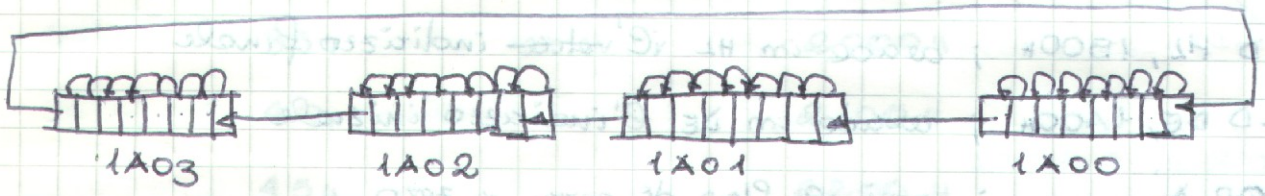
~~SIRENA~~

```
INCENDIO  OUT(40H), A    ; azione il sistema antincendio
SIRENA    OUT(30H), A    ; e/o la sirena
          JP INIZIO      ; riprendi il controllo
```

PROGRAMMA 32

Scrivere un programma che effettui la rotazione completa a sinistra di un dato di 4 byte scritto in memoria a partire dalla locazione 1A00

Cerchiamo di analizzare la situazione graficamente



Considerato come un unico dato a 32 bit q, per avere un unica rotazione, ogni byte va shiftato a sinistra di una posizione e in coda deve finire il bit più significativo del byte precedente per ottenere una rotazione completa questa operazione va fatta 32 volte.

```
        ORG XXXX

        LDB,32      ; contatore delle rotazioni
        OR A        ; carry flag a zero
LOOP1   LD C,04H    ; contatore degli shift

        LD HL,1A00H ; facciamo puntare al 1° byte
        [illegible]
        [illegible] ; inizialmente il carry flag deve essere a zero
                    e si pone a zero, [illegible]
LOOP2   SLA (HL)    ; si shifta a sinistra il generico byte, in coda
                    ; finisce uno zero
        LD A,(HL)   ; si pone il dato shiftato in A
        ADC A,00H   ; serve a sommare al dato il carry flag cioè
                    ; a spostare nel byte il bit più significativo
        LD (HL),A
        INC HL      ; punta al byte successivo
        DEC C

        JR NZ,LOOP2 ; shifta il byte successivo

        DJNZ ,LOOP1 ; fai un altra rotazione

        HALT
```

PROGRAMMA 31

Scrivere un programma che verifichi se nelle locazioni 1A00 - 1B00 ci sia il valore 00H; se ciò non si verifica i dati vanno inviati alla porta di uscita 80H.

```
        ORG XXXX
        LD HL, 1B00H    ; carica in HL l'indirizzo finale
        LD DE, 1A00H    ; carica in DE l'indirizzo iniziale
        XOR A           ; poni le flag di carry a zero
        SBC HL, DE      ; calcola la differenza fra indirizzo finale e
                        ; iniziale, questa è pari al numero di locazioni
                        ; da controllare meno uno
        LD C, L         ; sposta questo dato in BC
        LD B, H
        INC BC          ; ora in BC c'è il numero di dati da controllare
                        ; per cui può funzionare da contatore
        PUSH BC         ; salva questo valore
        ADD HL, DE      ; ripristina il vecchio valore di HL
        LD A, 00H       ; si vuole controllare se nelle in una delle
                        ; locazioni ci sia un valore nullo
        CPDR            ; confronta finché BC = 0 oppure A = (HL)
        JR Z, FINE      ; se un dato era nullo Z = 1
                        ; per cui termina il programma
        INC HL          ; altrimenti fai puntare HL alla 1ª locazione
        LD C, 80H       ; e C alla porta
        OTDR            ;
        POP BC          ; ripristina il contatore
INDIETRO LD A, (HL)
        OUT (80H), A    | ciclo
        DEC BC          |
```

di uscita 80H.

```
        ORG XXXX
        LD HL, 1B00H  ; carico in HL l'~~valore~~ indirizzo finale
        LD DE, 1A00H  ; carico in DE l'indirizzo iniziale
        ~~X~~ORA      ; pongo il flag di carry a zero
        SBC HL, DE    ; calcolo la differenza fra indirizzo finale e
                      ; iniziale, questa è pari al numero di locazioni
                      ; da controllare meno uno
        LD C, L       ; sposto questo dato in BC
        LD B, H
        INC BC        ; ora in BC c'è il numero di dati da controllare
                      ; per cui può funzionare da contatore
        PUSH BC       ; salvo questo valore
        ADD HL, DE    ; ripristino il vecchio valore di HL
        LD A, 00H     ; si vuole controllare se ~~nelle~~ in una delle
                      ; locazioni ci sia un valore nullo
        CPDR          ; confronto finché BC = 0 oppure A = (HL)
        JR Z, FINE    ; se un dato era nullo Z = 1
                      ; per cui termina il programma
        INC HL        ; altrimenti fai puntare HL alla 1ª locazione
        ~~LD C, 80H~~   ~~; e C alla porta~~
        ~~OUT (C), A~~
        POP BC        ; ripristino il contatore
INDIETRO LD A, (HL)
        OUT (80H), A  | sbagliato
        DEC BC
        INC HL
        JR NZ, INDIETRO
FINE    HALT
```

PROGRAMMA 30.

In una piastra vi sono 5 periferiche in grado di generare interruzioni

INDIRIZZO PERIFERICA	INDIRIZZO ROUTINE DI GESTIONE
20 H	1800 H
30 H	1900 H
35 H	2000 H
40 H	2100 H
45 H	2200 H

Scrivere il programma di inizializzazione delle schede supponendo di porre la tabella delle interruzioni all'indirizzo 1000 H.

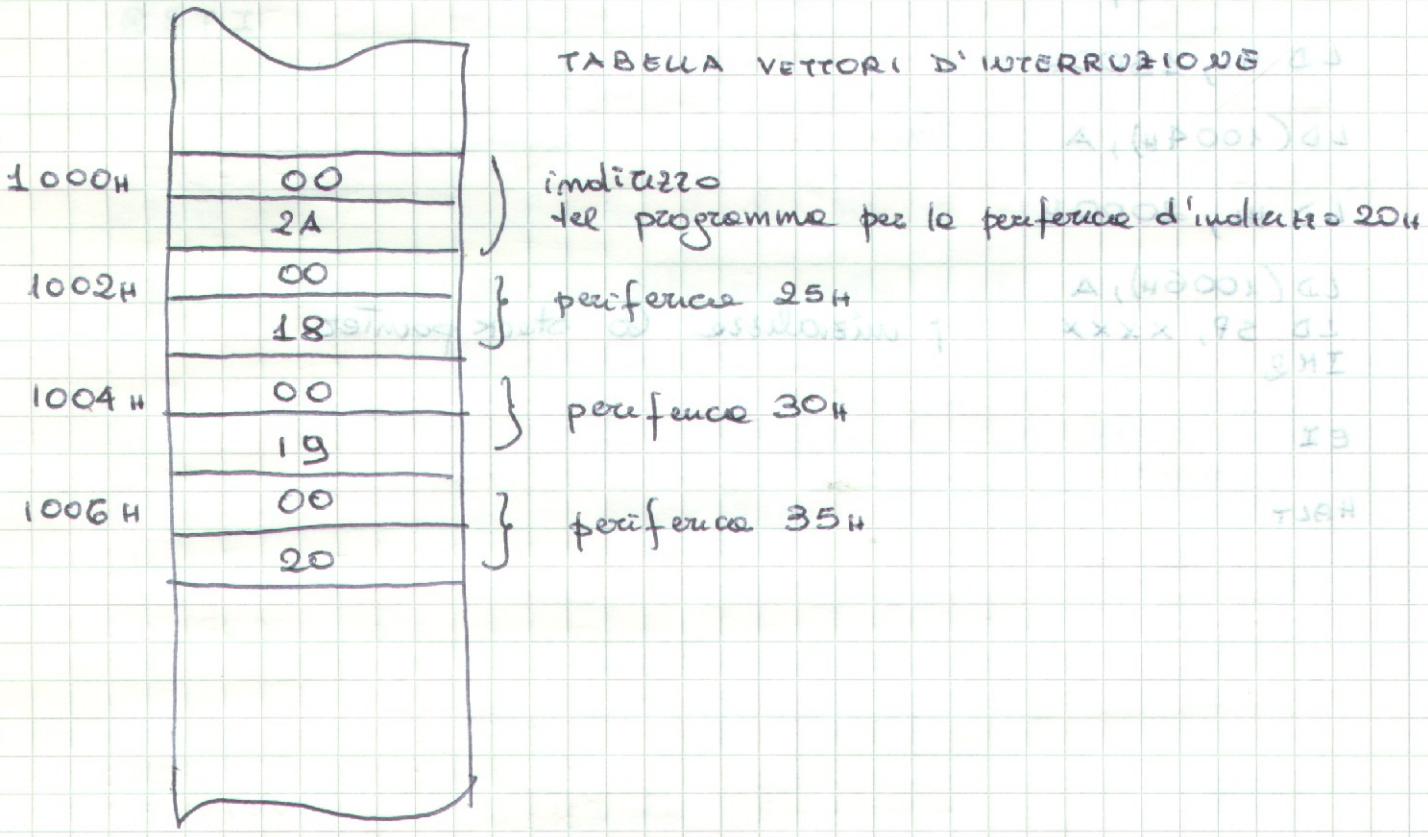
PROGRAMMA 29

La periferica d'indirizzo 20_H, quando vuole inviare dati al uP, genera un segnale d'interrupt. Nella scheda vi sono altre 3 periferiche che generano interruzioni secondo la seguente Tabella

INDIRIZZO PERIFERICA	INDIRIZZO PROGRAMMA
25_H	1800_H
30_H	1900_H
35_H	2000_H

Quando la periferica (20_H) genera l'interruzione il uP preleverà dalla stessa 100 dati e li immagazzinerà in memoria a partire dall'indirizzo 3000_H. Il programma che gestisce questa interruzione si trova all'indirizzo $2A00_H$.

Scrivere il programma di inizializzazione supponendo che la Tabella delle interruzioni parta dall'indirizzo 1000_H e il programma che gestisce la periferica d'indirizzo 20_H.



```
ORG 0000
LD A, 20H
LD I, A ; carica in I la parte alta dell'indirizzo
        ; della tabella
LD A, 00H
OUT (20H), A ; invia la parte bassa alla periferica
LD A, 02H ; invia 1
OUT (25H), A ; invia la parte bassa alla 2ª periferica
LD A, 04H
OUT (30H), A ; invia la parte bassa alla 3ª periferica
LD A, 06H
OUT (35H), A ; invia la parte bassa alla 4ª periferica
LD HL, 2A00H ; riempio la tabella
LD (1000H), A ; con gli indirizzi dei sottoprogrammi
LD HL, 1800H
LD (1002H), A
LD HL, 1900H
LD (1004H), A
LD HL, 2000H
LD (1006H), A
LD SP, XXXX ; inizializza lo stack pointer
IM 2
EI
HALT
```

```
ORG 2A00
LD B, 64H
LD HL, 3000H
LD C, 20H
INIR
HALT
```

```
ORG 2A00H
; questa procedura modifica B, C, HL
 PUSH BC
 PUSH HL
 LD HL, 3000H
 LD C, 20H
 INIR
 POP BC
 POP HL
 RETI
```

PROGRAMMA 28

In una scheda abbiamo un contatore all'indirizzo 30_H, quest quando si invia il dato 13_H al contatore, questo si attiva e dopo circa un minuto genera un'interruzione; all'indirizzo 1800_H c'è il programma di gestione dell'interruzione il cui scopo è quello di prelevare 100 dati dalla porta di indirizzo 25_H e metterli in memoria a partire dall'indirizzo 3000_H.

1- Scrivere il programma di inizializzazione della scheda.
2- Scrivere la routine di gestione dell'interruzione.

Il programma di inizializzazione deve partire dalla locazione 0000_H perché è il 1° compito che il µP deve eseguire

```
        ORG 0000H.
        LD A, 20H; suppongo che la tabella delle interruzioni
                 ; cominci a 2000H
        LD I, A  ; carico la parte alta dell'indirizzo in I
        LD A, 00H; nella tabella c'è solo l'indirizzo della routine
                 ; del contatore
        OUT(30H), A; invio la parte bassa del vettore d'interruzione
                   ; al contatore
                   ; si suppone che abbia un solo indirizzo d'ingresso
        LD HL, 1800H; in HL pongo l'indirizzo della
                    ; routine che gestisce l'interruzione del
                    ; contatore
        LD (2000H), HL; pongo in 2000H e 2001H l'indirizzo
                      ; della routine
        IM2           ; imposto il modo 2
        LD SP, XXXX   ; inizializzo lo stack
        EI            ; abilito le interruzioni
```

```
OUT(30H), A     ; faccio partire la 1ª volta il contatore
HALT            ; la scheda si blocca e attende un'interruzione

ORG 1800H       ; programma del contatore
LD B, 100       ; inizializzo il contatore
LD C, 25H       ; faccio puntare C alla porta di uscita
LD HL, 3000H    ; HL punta alla zona di destinazione
INIR            ; prelevo i miei dati
LD A, 13H
OUT(30H), A     ; faccio ripartire il contatore
EI              ; abilito le interruzioni
RETI
```

PROGRAMMA 23

Scrivere un programma che prelevi 100 dati che arrivano porta d'indirizzo 20H; il programma controlla il dato ricevuto e se non è zero lo invia in uscita alla porta 30H;

PROGRAMMA 24 X

Scrivere un programma che pone a zero 100 locazioni di memoria a partire dall'indirizzo 1A00H;

PROGRAMMA 25 X

A partire dalla locazione 1A00H preleva 100 dati, complementali e riscrivili a partire dalla locazione 1B00.

PROGRAMMA 26

Scrivere un programma che effettui la seguente operazione $(a^2+b^2)^2$ dove a e b sono dati scritti in memoria alle locazioni 1A00 e 1A01, si ha a disposizione un sottoprogramma QUAD che aspetta il dato nel registro L e restituisce il quadrato in HL.

PROGRAMMA 27 X

Si hanno 100 dati in memoria a partire dalla locazione 1A00, si controlla ogni dato e se è diverso da zero lo spedisce alla porta 10H.

PROGRAMMA 28

PROGRAMMA 23

```
        LD B, 64H
CICLO:  IN A, (20H)
        CP 00H
        JP Z, AVANTI
        OUT (30H), A    ; se il dato non è nullo
AVANTI: DJNZ, CICLO     ; lo spedisci in uscita
```

PROGRAMMA Q4

```
        LD  B, 64H
        LD  HL, 1400H
        XOR A ; azzera l'accumulatore
CICLO:  LD (HL), A
        INC HL
        DJNZ, CICLO
        HALT
```

PROGRAMMA 25

```
        LD  B, 64H

        LD  HL, 1A00H
        LD  DE, 1B00H
CICLO:  LD  A, (HL)
        CPL
        LD  (DE), A
        INC HL
        INC DE
        DJNZ, CICLO
        HALT
```

PROGRAMMA 26

```
LD A, (1A00H)
LD L, A
CALL QUAD      ; ora HL contiene a^2
LD B, H
LD C, L        ; lo sposto in BC
LD A, (1A01H)
LD L, A
~~ADD HL, BC~~
CALL QUAD      ; ora HL contiene b^2
ADD HL, BC     ; ora HL contiene a^2 + b^2
CALL QUAD      ; questo programma
               ; presuppone che comunque
               ; i quadrati occupino un byte
HALT
```

PROGRAMMA 27

```
        LD  B, 64H
        LD  HL, 1A00H
CICLO:  LD  A,(HL)
        CP  00H
        JP  Z, AVANTI
        OUT (10H), A ; Se il dato non è nullo
                     ; spediscilo
AVANTI: INC HL
        DJNZ, CICLO
        HALT
```

PROGRAMMA 22

Si ha a disposizione un sottoprogramma QUAD che prende il dato che si trova nel registro E e pone in DE il quadrato. Usarlo per effettuare l'operazione

$$(a^2 + b^2 + c^2)^2$$

dove a è il dato nella locazione 1A00, b nella locazione 1A01, c nella locazione 1A02.

(Si lascia la soluzione al lettore).

```
LD  HL, 1A00H
LD  E, (HL)
CALL QUAD      ; DE contiene a^2
LD  B, D       ; e lo sposto in BC
LD  C, E
INC HL
LD  E, (HL)
CALL QUAD      ; ora DE contiene b^2
LD  A, E       ; e sommo BC e DE
ADD A, C       ; in modo che poi
LD  C, A       ; BC contenga a^2 + b^2
LD  A, B
ADC A, D
LD  B, A
INC HL
LD  E, (HL)    ; carico c
CALL QUAD
LD  H, D
LD  L, E       ; ora HL contiene c^2
ADD HL, BC     ; sommo c^2 + (a^2 + b^2)
```

effettuare l'operazione

$$(a^2 + b^2 + c^2)^2$$

dove a è il dato nella locazione 1A00, b nella locazione 1A01, c nella locazione 1A02.

(Si lascia la soluzione al lettore).

```
LD  HL, 1A00H
LD  E, (HL)
CALL QUAD      ; DE contiene a^2
LD  B, D       ; e lo sposto in BC
LD  C, E
INC HL
LD  E, (HL)
CALL QUAD      ; ora DE contiene b^2
LD  A, E       ; e sommo BC e DE
ADD A, C       ; in modo che poi
LD  C, A       ; BC contenga a^2 + b^2
LD  A, B
ADC A, D
LD  B, A
INC HL
LD  E, (HL)    ; carica c
CALL QUAD
LD  H, D
LD  L, E       ; ora HL contiene c^2
ADD HL, BC     ; sommo c^2 + (a^2 + b^2)
LD  D, H
LD  E, L
CALL QUAD
HALT
```

PROGRAMMA 21

Si ha a disposizione il sotto programma DIV che effettua la divisione fra il contenuto del registro BC e quello del registro E e pone il risultato in C, e il sotto programma MOLT che effettua la moltiplicazione fra i registri C ed E e pone il risultato in DE; effettuare la operazione

$$\frac{ab}{c} + \frac{de}{f}$$

dove a è il contenuto di 1A00, b di 1A01, c di 1A02, d di 1A03, e di 1A04, f di 1A05.

CODIFICA

~~LD HL, 1~~

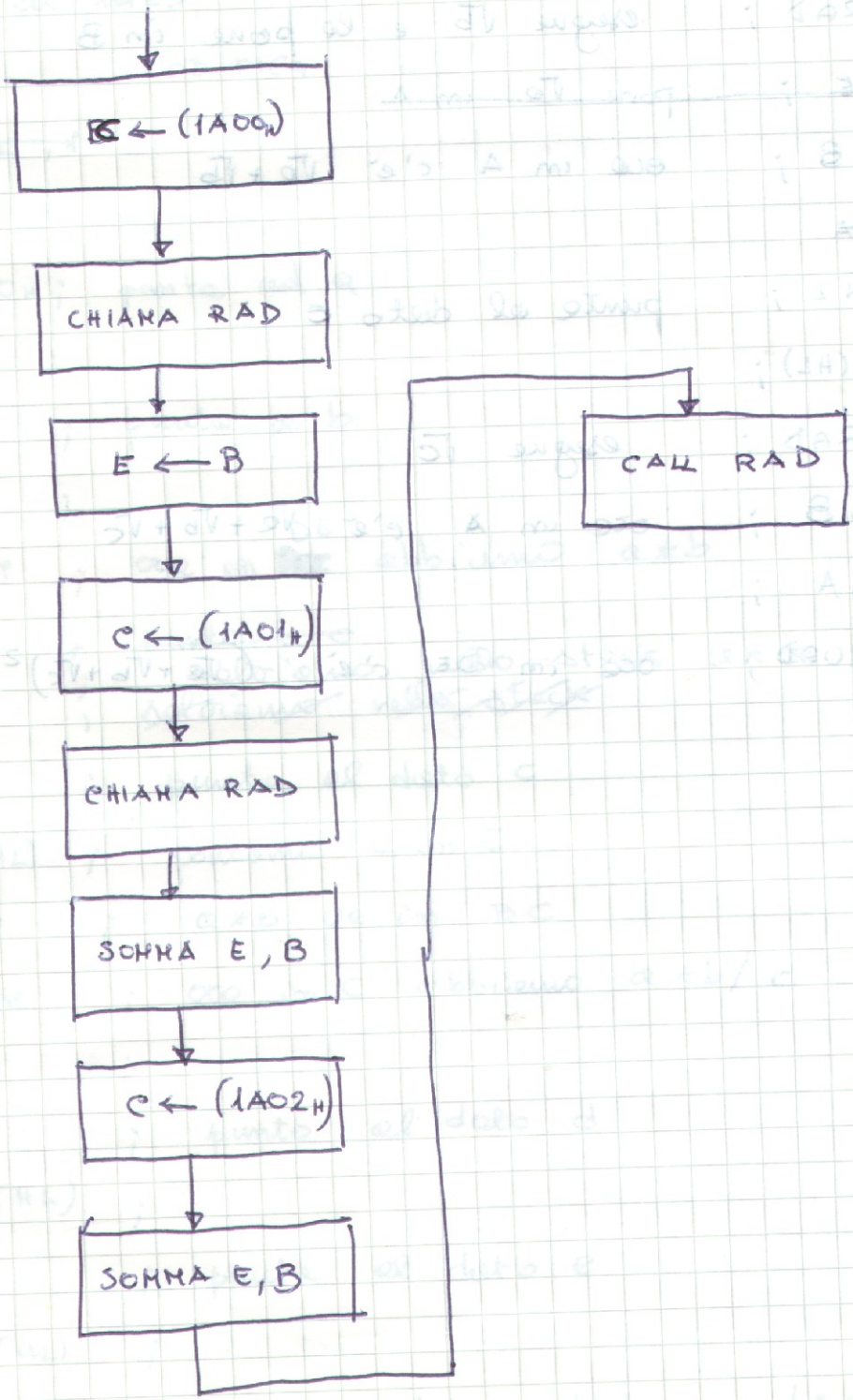
```
ORG XXXX
LD HL, 1A00H ; punta ad a
LD C, (HL)   ;
INC HL       ; punta a b
LD E, (HL)   ;
CALL MOLT    ; ora in DE abbiamo a*b
~~INC HL~~   ; ~~punta a c~~
PUSH DE      ; salviamo nello stack a*b
~~PUSH DE~~  ; ~~salviamo nello stack~~
INC HL       ; punta al dato c
LD E, (HL)   ; poniamo c in E
POP BC       ; a*b va in BC
CALL DIV     ; ora in C abbiamo a*b/c
LD A, C      ;
INC HL       ; punta al dato d
LD C, (HL)   ;
INC HL       ; punta al dato e
LD E, (HL)   ;
```

```
PUSH DE ;  e lo salviamo nello stack
INC HL  ; punta al dato f
LD E, (HL) ;
POP BC   ; ora in BC abbiamo d*e
CALL DIV ; ora in C abbiamo d*e/f
ADD A, C ;  in A abbiamo a*b/c + d*e/f
HALT
```

PROGRAMMA 20

Il sottoprogramma QUAD prende il contenuto del registro E, ne fa il quadrato e lo pone nel registro DE, il sottoprogramma RAD prende il contenuto del registro C, ne fa la radice e la pone nel registro B, eseguire con questi sottoprogrammi l'operazione $(\sqrt{a}+\sqrt{b}+\sqrt{c})^2$ dove a è il contenuto della locazione 1A00, b il contenuto di 1A01 e c il contenuto di 1A02.

FLOW CHART



CODIFICA

```
ORG XXXX
LD HL, 1A00H ; punta al dato a
LD C, (HL) ; carica a in nel registro C
CALL RAD ; esegue √a
LD A, B ; adesso in A c'è √a
INC HL ; punta al dato b
LD C, (HL) ;
CALL RAD ; esegue √b e la pone in B
~~LD A, E ; pone √a in A~~
ADD A, B ; ora in A c'è √a + √b
~~LD E, A~~
INC HL ; punta al dato c
LD C, (HL) ;
CALL RAD ; esegue √c
ADD A, B ; ora in A c'è √a + √b + √c
LD E, A ;
CALL QUAD ; ora in DE c'è (√a + √b + √c)^2
```

PROGRAMMA 19

Si hanno 100 dati in memoria a partire dalla locazione 1A00, si controlla se ogni dato è diverso da zero e lo si spedisce alla porta 10_H.

CODIFICA

```
    LD B, 64H
    LD HL, 1A00H
LOOP LD A, (HL)
    CP 00
    INC HL
    CP 00H   ; controlla se il dato prelevato è nullo
    JR NZ, LOOP ; in tal caso si passa al successivo
    OUT(10H), A ; altrimenti lo si spedisce in uscita
    DJNZ LOOP
    HALT
```

PROGRAMMA 18

Si ha una periferica che si trova all'indirizzo 20H; appena si scrive la stringa 11001100 B sulla periferica questa viene attivata e dopo 10 minuti 20 dati genera un'interrupt. Sfruttando questa periferica si vuole prelevare ogni 10 minuti 20 dati dalla periferica d'indirizzo 40H con la tecnica dell'interrupt modo 2. Scrivere il programma di inizializzazione e il programma di gestione dell'interruzione.

~~CODIFICA~~ ANALISI

Supponiamo che la tabella dei vettori d'interruzione si trovi in RAM a partire dalla locazione 1800H, e che il programma di gestione si trovi a partire dalla locazione 1900H.

CODIFICA

```
ORG 0000
LD HL, 1900H
LD (1900), HL ; si scrive in mem tabella delle interruzioni
; l'indirizzo del programma di gestione delle interruzioni
IM 2          ; si pongono le interruzioni in modo 2
EI            ; si abilitano le interruzioni
OUT (20), CCH ; si attiva la periferica
  :
ORG 1900
LD B, 14H     ; si carica il contatore
LD HL, 1A00H  ; HL punta alla locazione in cui si vuole
              ; memorizzare il dato
LD C, 40H     ; e punta alla porta di I/O
INIR
OUT (20), CCH ; occorre riattivare la periferica
              ; per avere una interruzione dopo altri 10 minuti
```

PROGRAMMA 17

Si scrive il sotto programma QUAD descritto all'esercizio precedente.

ANALISI

Per effettuare una elevazione al quadrato basta sommare HL con se stesso tante volte quanto è scritto in L.

CODIFICA

```
        ORG  XXXX
        LD   B,L        ; carica il contatore
        ~~ADD  HL,HL~~
        LD   H,00H      ; per un funzionamento corretto occorre
        ; assicurarsi che la parte alta di HL sia nulla
LOOP :  ADD  HL,HL
        DJNZ LOOP
        RET
```

PROGRAMMA 16

A partire dalla locazione 1A00 ci sono 100 dati da 1 byte; a partire dalla locazione 1B00 si devono immagazzinare su 2 byte i quadrati dei 100 dati precedenti. Si fa a disposizione un sottoprogramma QUAD che prende un numero immagazzinato nel registro L e restituisce nel registro HL il suo quadrato.

CODIFICA

```
        LD B, 64H       ; contatore
        LD DE, 1A00H    ; punta ai dati da elevare al quadrato
        LD IX, 1B00H    ;
LOOP:   LD A, (DE)      ; carica in A il dato da elevare al quadrato
        LD L, A         ; sposta in L
        CALL QUAD       ; chiama QUAD
        LD (IX), L      ; scrivi il byte inferiore in memoria
        LD (IX+1), H    ; scrivi il byte superiore
        INC IX
        INC IX          ; incrementa di due il puntatore
        DJNZ LOOP
        HALT
```

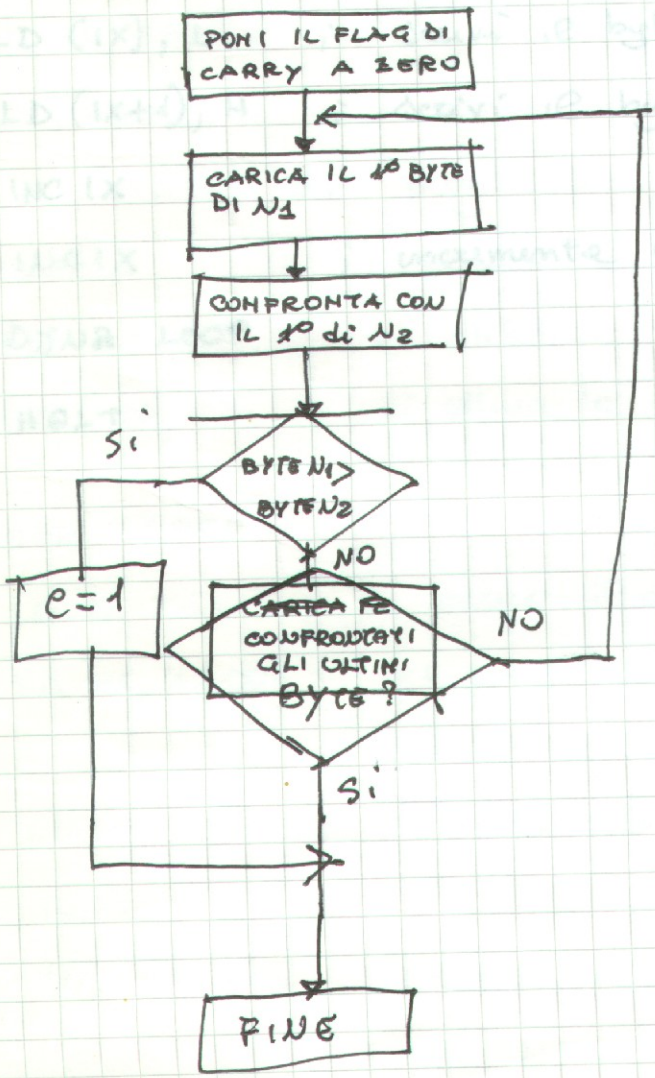
PROGRAMMA 15

Scrivere un programma per controllare se $N_1 > N_2$ dove N_1 e N_2 sono due dati rispettivamente di 4 byte scritti in memoria, il primo a partire dalla locazione 1A00 (byte meno significativo), il secondo a partire dalla locazione 1A04 (byte meno significativo).

ANALISI

Il programma deve comunicare in qualche modo all'ambiente esterno che $N_1 > N_2$. Si può usare, ad esempio, il flag C in modo che C=1 indica che $N_1 > N_2$ e C=0 indica che $N_1 \not> N_2$. Il controllo si può effettuare sott confrontando prima i due byte più significativi, se il byte più significativo di N_1 è maggiore di quello di N_2 si esce dal programma dopo aver settato C altrimenti si prosegue controllando gli altri byte.

FLOW CHART

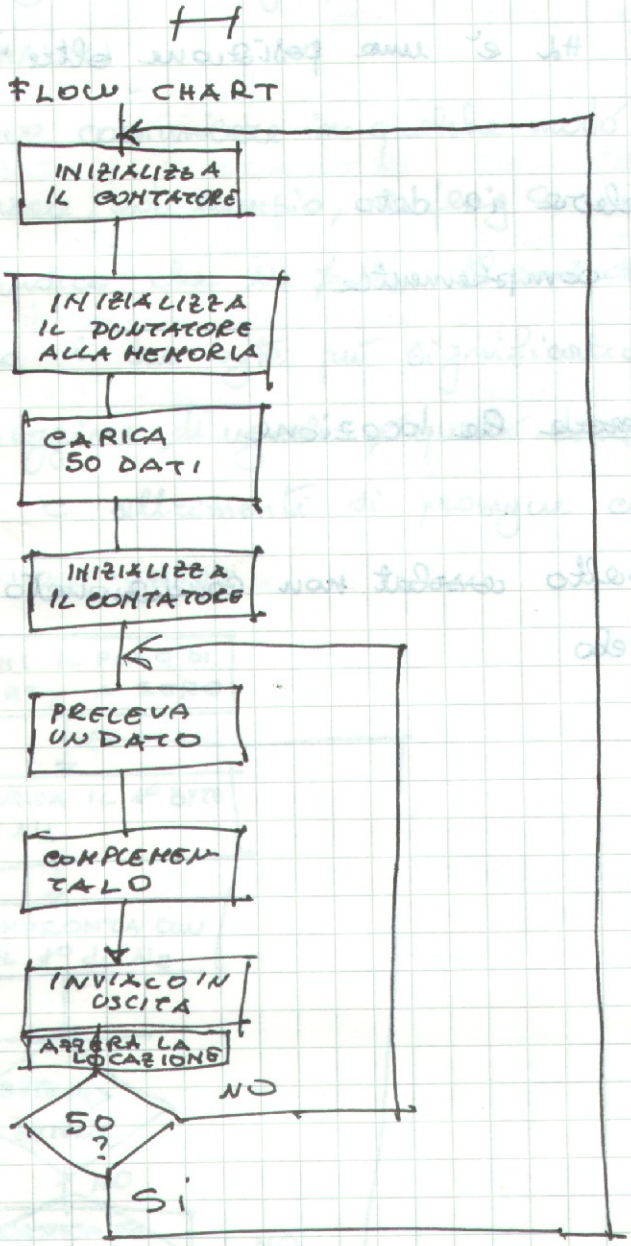


```
        LD B,04H        ;
        LD DE,1403H     ; DE punta ai byte di N1
        LD HL,1407H     ; HL punta ai byte di N2
        AND A           ; non cambia A serve solo a porre C=0
LOOP:   LD A,(DE)       ; ora A contiene l'MS byte di N1
        CP (HL)         ; se N1>N2 il flag di segno sarà positivo
        JP P,FINE1      ; se S=0 vai alla fine per settare C
        DEC DE          ; altrimenti decrementa i puntatori
        DEC HL          ; per controllare gli altri byte
        JR Z,FINE       ; se N1 è addirittura minore di N2 esci
        DJNZ,LOOP       ; altrimenti vai a controllare i successivi
FIN1:   CCF             ; poni C=1
FINE:   HALT
```

PROGRAMMA 14

Si prelevano ~~100~~ dati dalla porta d'indirizzo 20H, quando si sono immagazzinati in memoria 50 dati, si complementano e si inviano in uscita alla porta di indirizzo 30H; si azzerano le locazioni utilizzate e si riprende il ciclo

FLOW CHART



CODIFICA

```
INIZ: LD C, 20H      ; C punta alla porta da cui prelevare i dati
      LD HL, 1A00H   ; HL punta all'area di memoria in cui conservare i da
      LD B, 32H      ; B è il contatore dei 50 dati
      INIR           ; l'istruzione incrementa HL anche l'ultima volta p
      DEC HL         ; cui HL è una posizione oltre il 50mo dato
      LD B, 50H      ;
LOOP: LD A, (HL)     ; prelevo il dato
      CPL            ; lo complemento
      OUT (30H), A
      LD (HL), 00H   ; azzero la locazione
      DEC HL
      DJNZ, LOOP
      JR INIZ        ; salto assoluto non condizionato per ripartire
      H              ; cielo
```

PROGRAMMA 13

Si hanno 100 dati scritti in memoria a partire dall'indirizzo $1A00_H$; sostituire ogni dato con la somma fra se stesso e il successivo.

(Si lascia la soluzione allo studioso lettore)

```
        LD B, 64H
        LD IX, 1A00H
CICLO:  LD A, (IX)
        ADD A, (IX+1)
        LD (IX), A
        INC IX
        DJNZ, CICLO
        HALT
```

PROGRAMMA 12

Si suppone di avere a disposizione un sottoprogramma RAD che prende il contenuto del registro HL, ne fa la radice quadrata e la espone in HL. Scrivere un programma che effettui il calcolo $\sqrt{\sqrt{N_1} - \sqrt{N_2}}$ dove N_1 è il contenuto della locazione 1800 ed N_2 è il contenuto della locazione 1801.

(Si lascia lo svolgimento allo studioso lettore)

```
LD DE, 1800H
LD A, (DE)
LD L, A
LD H, 00H
CALL RAD
LD B, H
LD C, L
INC DE
LD A, (DE)
LD L, A
CALL RAD
LD D, H
LD E, L
LD H, B
LD L, C
AND A
SBC HL, DE
CALL RAD
HALT
```

PROGRAMMA 11

Si prelevano dati da una porta di indirizzo 20H e si immagazzinano in memoria a partire da 1800H; il primo dato prelevato dice quanti sono i dati successivi da prelevare. Se è pari a 00H, si interrompe il programma.

CODIFICA

```
LD C, 20H      ; C punta alla porta
LD A, (C)
CP 00H         ; controlla se il 1° dato è 00
JR Z, FINE     ; se sì, termina il programma
LD B, A        ; altrimenti carica il contatore B
LD HL, 1800H   ; carica il puntatore alla memoria
INIR           ; caricamento ricorsivo
FINE : HALT
```

PROGRAMMA 10 *

Si hanno 100 dati immagazzinati in memoria agli indirizzi a partire da $1B00_H$; fare in modo che il nibble inferiore di ogni dato sia sostituito dalla stringa 0000
(lo svolgimento si lascia allo studioso e solerte studente)

```
        LD  B, 64H
        LD  HL, 1B00H
CICLO:  LD  A, (HL)
        AND F0H
        LD  (HL), A
        INC HL
        DJNZ, CICLO
```

PROGRAMMA 9

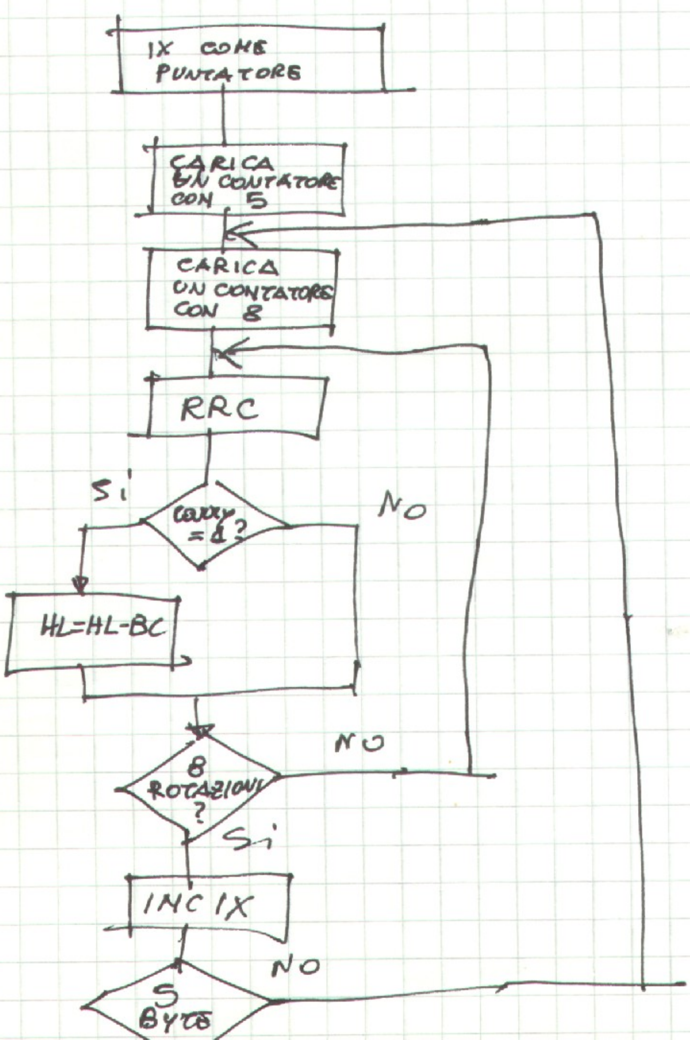
Si ha un dato scritto in memoria a partire dalla locazione 1800 fino alla locazione 1804; controllare un bit alla volta di questo dato a partire dall'LSB di 1800; se il bit è 1 effettuare la sottrazione fra i registri HL e BC.

ANALISI

Per analizzare il contenuto di questi bit basta effettuare ~~una~~ otto rotazioni a destra di ogni locazione con l'istruzione RRC in modo che ogni volta LSB finisce nel flag di carry che si può testare con un JR condizionato.
Non si può ~~FLO~~ usare HL come puntatore alla memoria perché viene modificato nel corso del programma. Non si può usare B come contatore per ~~FLOW CHART~~ lo stesso motivo

FLOW CHART

x 99



522111

CODIFICA

```
        LD IX, 1800H    ; IX punta ai byte di memoria
        LD D, 05H       ; D conta il numero di byte
LOOP1:  LD E, 08H       ; E conta il numero di rotazioni di 1 byte
LOOP2:  RRD (IX)        ; ruota il byte a destra
        JR NC, AVANTI   ; se il carry=0 non esegui la sottrazione
        SUB HL, BC      ;
AVANTI: DEC E           ; controlla se il byte è stato ruotato
        JR NZ, LOOP2    ; 8 volte
        ~~DEC~~ INC IX  ; passa al byte successivo
        DEC D
        JR NZ, LOOP1    ; controlla se sono finiti i byte
        HALT
```

PROGRAMMA 8

Scrivere un programma che pone a zero 100 locazioni di memoria a partire dall'indirizzo 1A00

CODIFICA

```
        LD  HL, 1A00H
        LD  B, 64H
~~LOOP~~  LD  A, 00H
LOOP    LD  (HL), A
        INC HL
        DJNZ LOOP
```

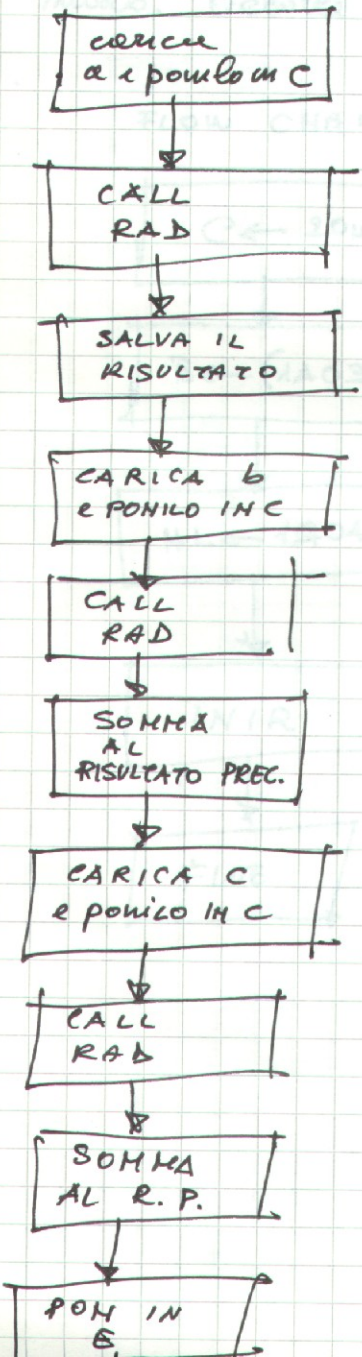
PROGRAMMA 4

Il programma QUAD prende il contenuto del registro E, ne fa' il quadrato e lo pone nel registro DE, il programma RAD prende il contenuto del registro C, ne fa' la radice e la pone nel registro B; eseguire con questi sottoprogrammi l'operazione

$$(\sqrt{a} + \sqrt{b} + \sqrt{c})^2$$

dove a è il contenuto della locazione 1A00, b il contenuto della locazione 1A01, e il contenuto della locazione 1A02

FLOW CHART



```
LD HL, 1A00 ;
LD C, (HL) ; preleva a
CALL RAD ;
LD A, B ; salva √a in A
INC HL ;
LD C, (HL)
CALL RAD ; adesso in B c'è √b
ADD A, B ; adesso in A c'è √a + √b
INC HL
CALL RAD ; adesso in B c'è √c
ADD A, B ; adesso in A c'è √a + √b + √c
LD E, A ;
CALL QUAD ; adesso in DE c'è (√a + √b + √c)²
HALT
```

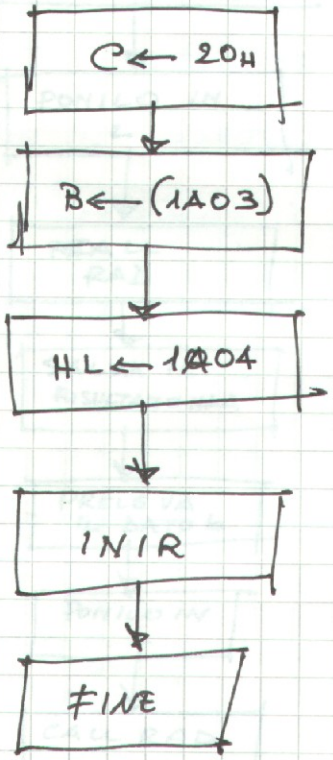
PROGRAMMA 6

Si effettuano prelievi di dati da una porta di indirizzo 20H e si scrivono in memoria a partire dall'indirizzo 1A04H, nel byte ~~da 1A00 a~~ 1A03 c'è scritto il numero di dati da acquisire

ANALISI

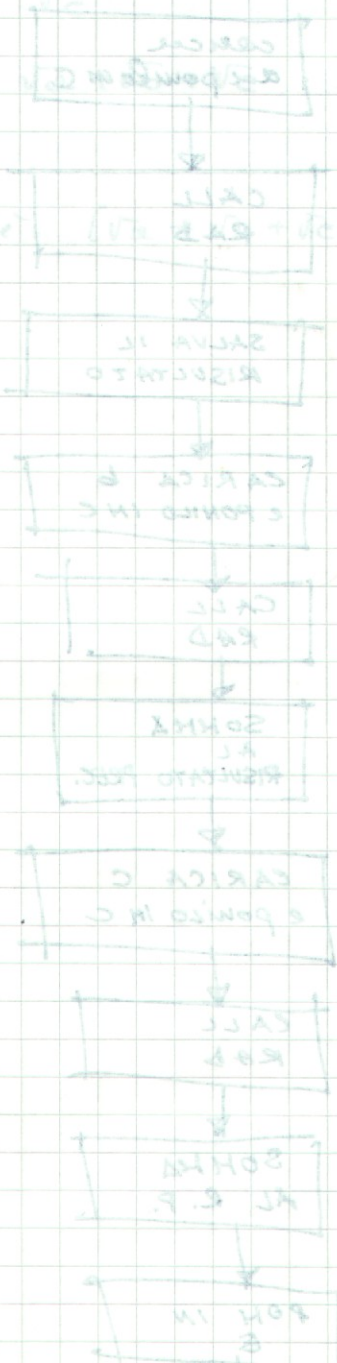
Il programma deve solo acquisire dati per cui conviene usare l'istruzione ricorsiva INIR che preleva i dati dalla porta il cui indirizzo è nel registro C e li pone nella locazione di memoria puntata da HL, incrementa HL e ripete fino a che il registro non diventa nullo. Occorre caricare B con il contenuto della locazione 1A03.

FLOW CHART



CODIFICA

```
LD C, 204 ; carica C con l'indirizzo della porta
LD A, (1A03) ;
LD B, A   ; carica in B il numero di byte da prelevare
LD HL, 1A04 ; carica il puntatore
INIR
FINE
```



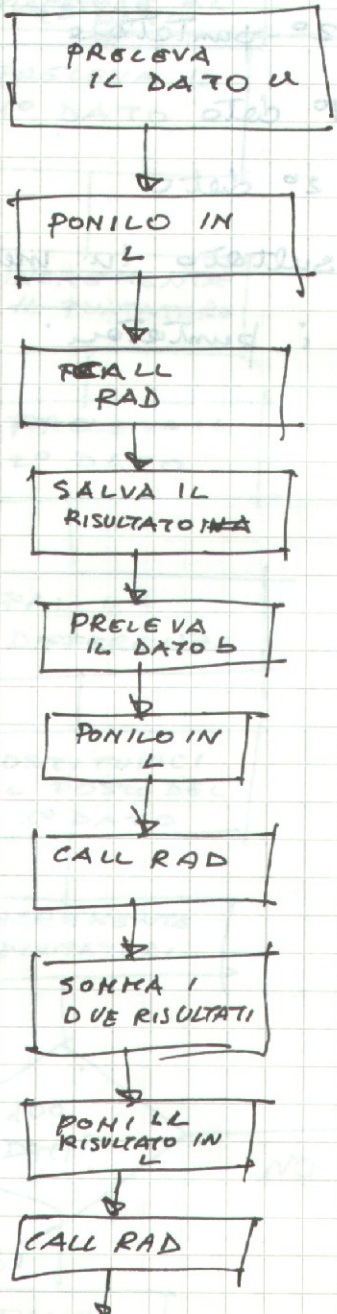
PROGRAMMA 5

Supporre di avere a disposizione un sottoprogramma RAD che prende il contenuto del registro L, ne fa la radice quadrata e la mette nel registro A; usarlo per scrivere un programma che effettui la seguente operazione

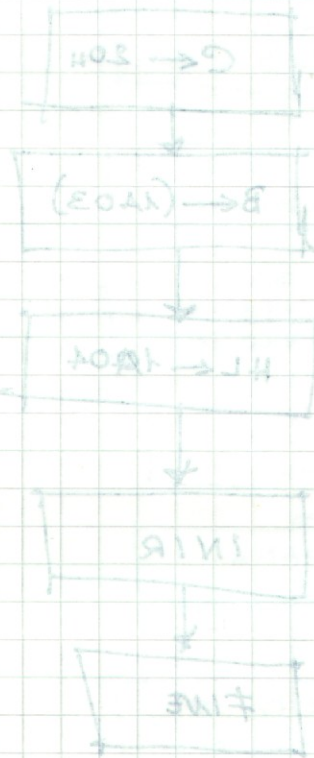
$$\sqrt{\sqrt{a} + \sqrt{b}}$$

dove a e b sono i contenuti dispettivamente delle locazioni 1A00 e 1A01.

FLOW CHART



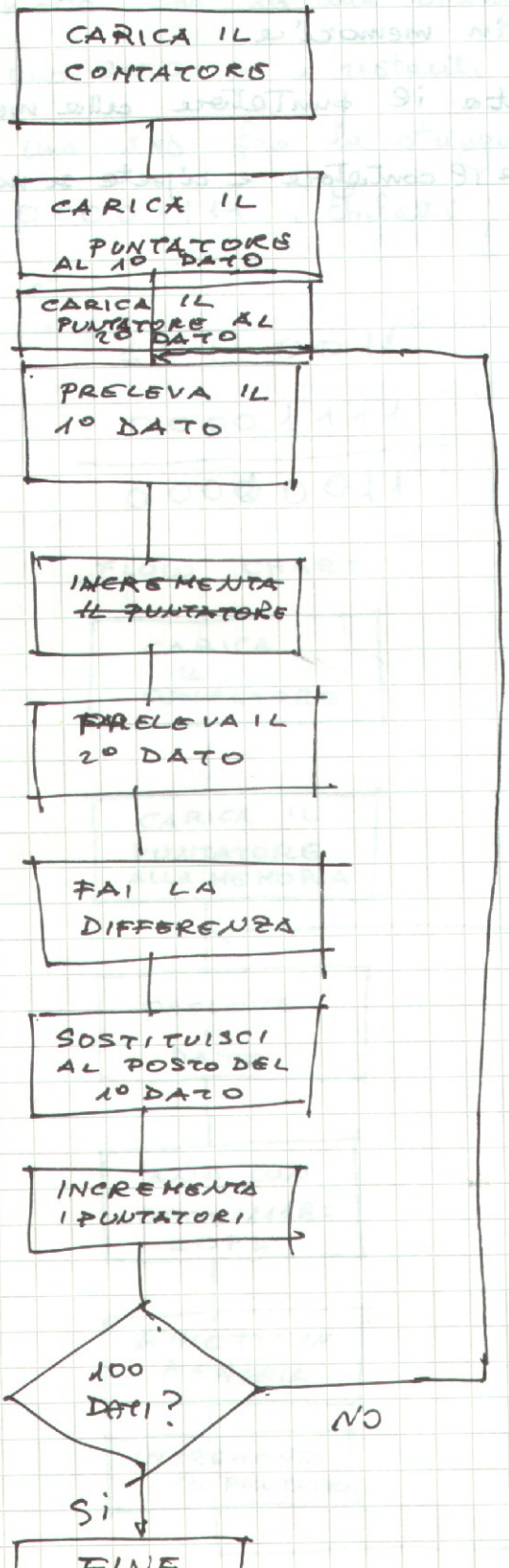
```
LD A, (1A00)    ; prelevo a
LD L, A         ;
CALL RAD
LD C, H         ; salvo √a in C
LD A, (1A01)    ; prelevo b
LD L, A
CALL RAD
LD A, H         ; adesso in A c'è √b
ADD C           ; esegue √b + √a e la pone nell'accumulatore
LD L, A         ;
CALL RAD        ; in H ci sarà √(√b + √a)
HALT
```



PROGRAMMA 4

Si hanno 100 dati scritti in memoria a partire dall'indirizzo 1A00H; sostituire ogni dato con la differenza fra se stesso e quello successivo.

FLOW CHART



ANALISI

Il metodo più rapido è avere due puntatori che si differenziano di 1, in modo che il 1° punta ad un dato ed il 2° punta a quello successivo. Poiché lo ~~Z80~~ 1° dato deve andare nell'accumulatore e lo Z80 permette di caricare l'accumulatore con l'indirizzamento indiretto usando BC e DE, usiamo DE come 1° puntatore (non BC poiché useremo B come contatore)

CODIFICA

```
        LD B, 64H   ; carica il contatore
        LD DE, 1A00H ; carica il 1° puntatore
        LD HL, 1A01H ; carica il 2° puntatore
LOOP:   LD A, (DE)  ; carica il 1° dato
        SUB A, (HL) ; sottrai il 2° dato
        LD (DE), A  ; metti il risultato in memoria
        INC DE      ; incrementa i puntatori
        INC HL
        DJNZ LOOP
        ~~BLUE~~ HALT
```

PROGRAMMA 3

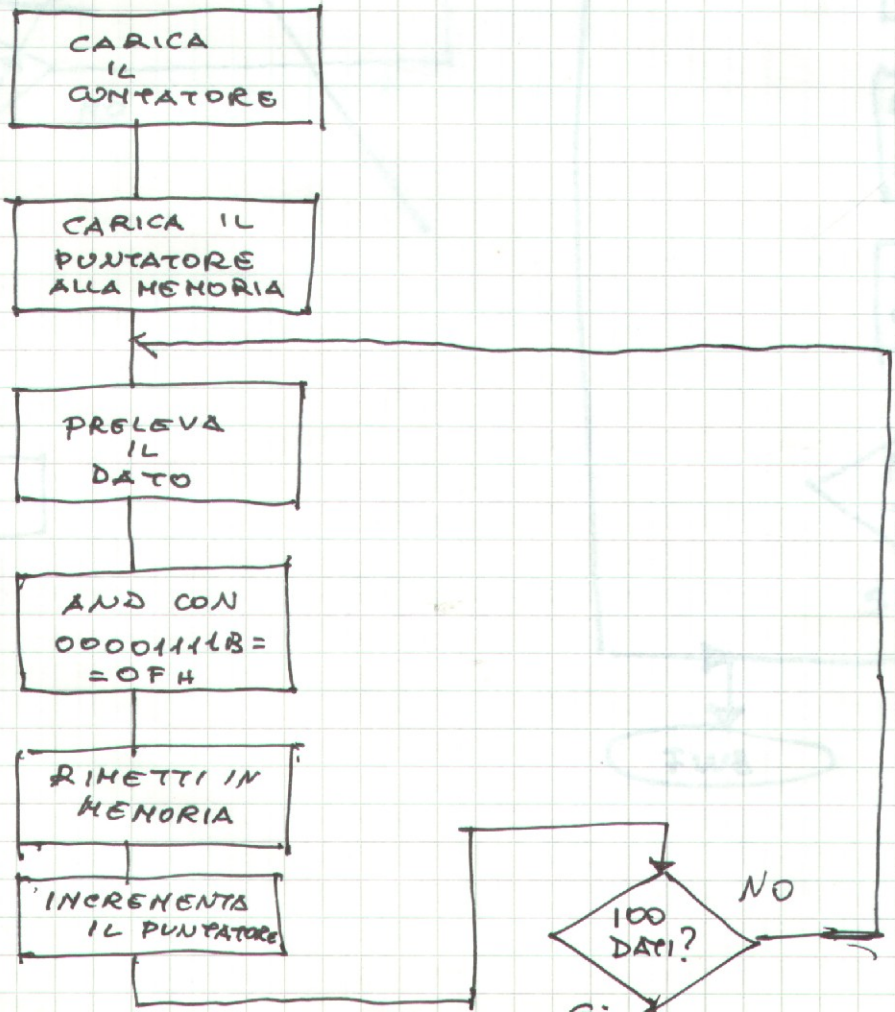
Si hanno 100 dati immagazzinati in memoria agli indirizzi a partire da 1A00H; fare in modo che il nibble superiore di ogni dato sia sostituito dalla stringa 0000 in binario.

ANALISI DEL PROBLEMA.

Per fare in modo che in una stringa di bit, i primi 4 siano sostituiti da uno zero e i restanti restino inalterati basta fare in modo che una AND fra la stringa da modificare e la maschera 00001111. Infatti, se, col esempio, la stringa di partenza è

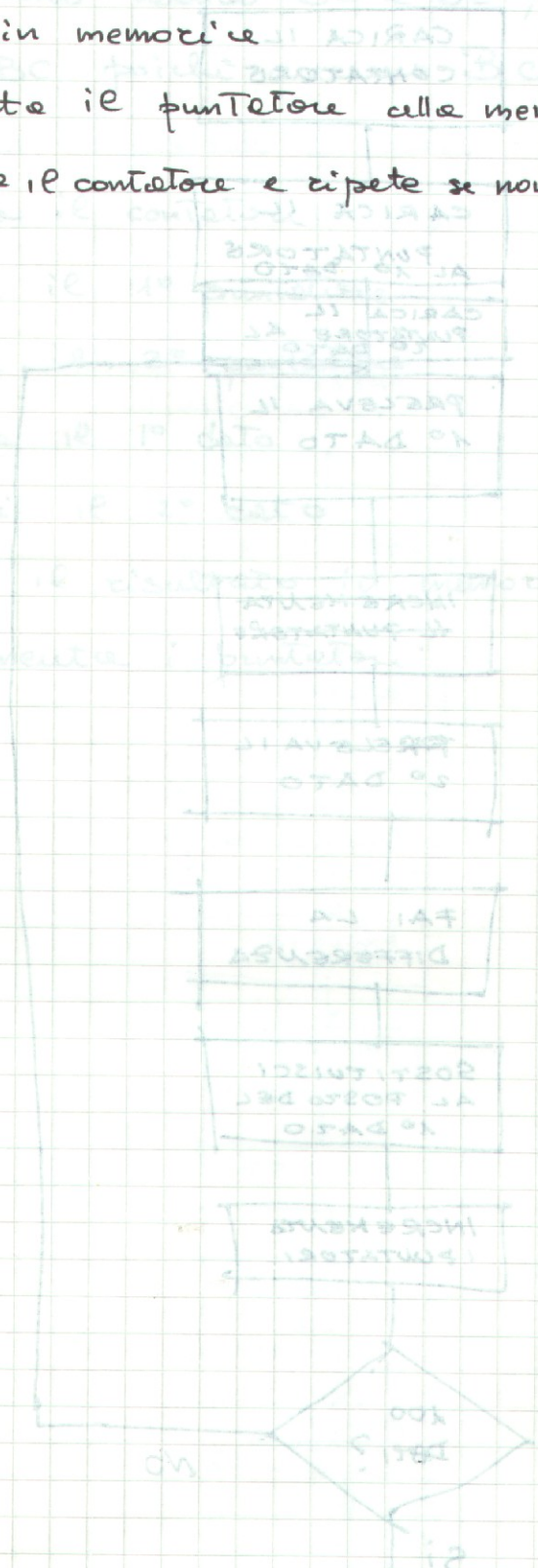
```
  10110011
AND 00001111
  --------
  00000011
```

FLOW CHART



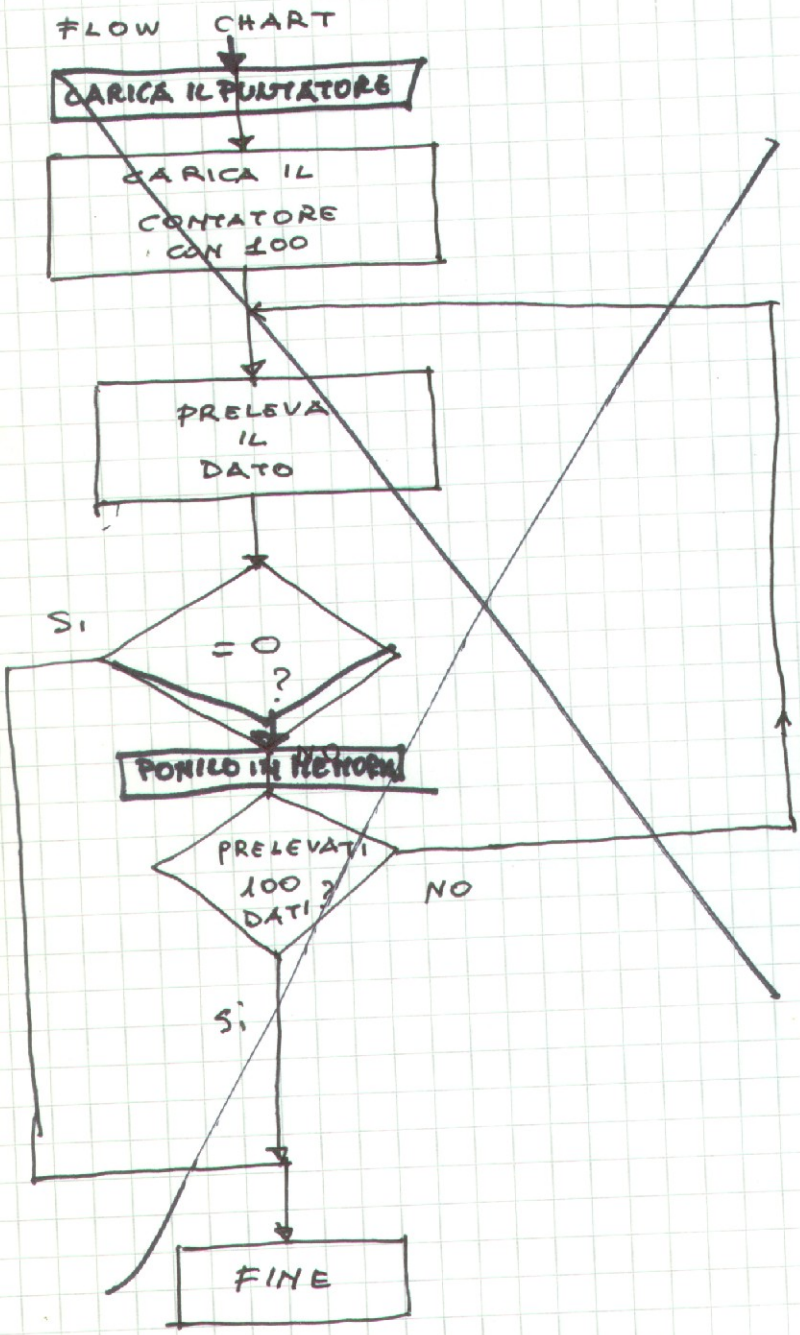
CODIFICA

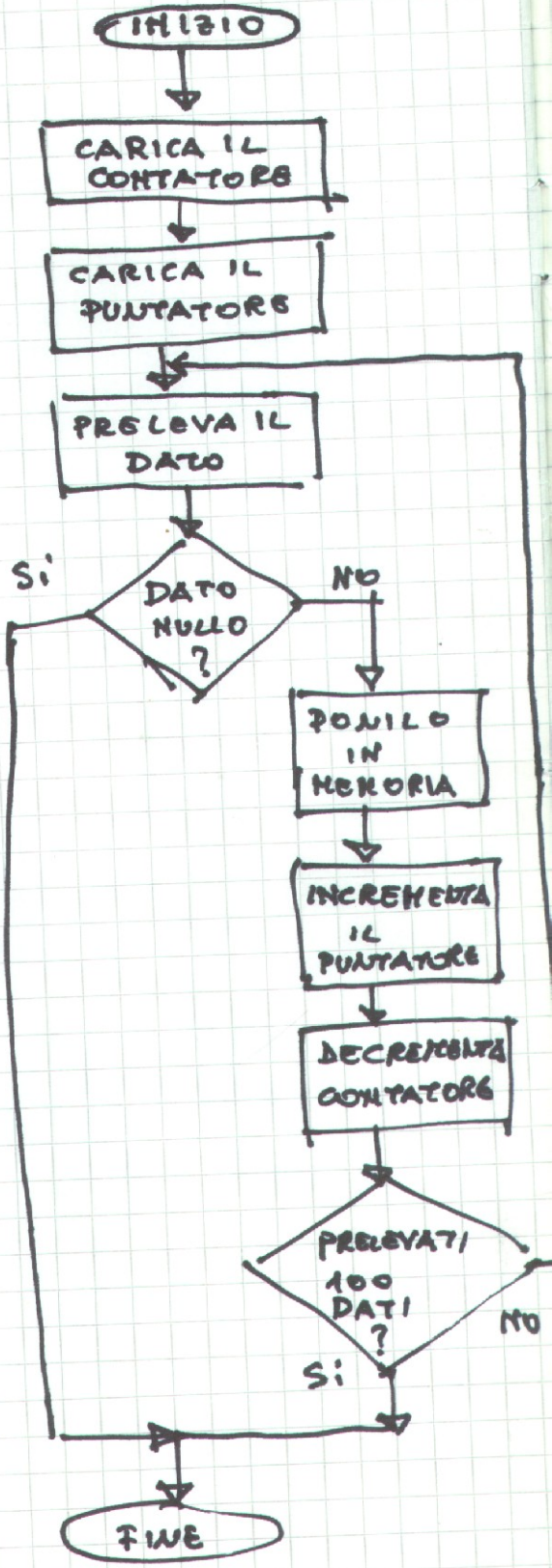
```
    LD B, 64H ;  carica il contatore
    LD HL, 1200H ;  carica il puntatore alla memoria
LOOP: LD A, (HL) ;  preleva il dato
    AND 0FH ;  fa la AND con la maschera
    LD (HL), A ;  rimette in memoria
    INC HL ;  incrementa il puntatore alla memoria
    DJNZ LOOP ;  decrementa il contatore e ripete se non sono 100 dati
    HALT
```



PROGRAMMA 2

Si prelevano 100 dati dalla porta d'indirizzo 20H e si scrivono in memoria.
Se il dato prelevato è 00H si interrompe il programma





CODIFICA

```
ORG   LD  B, 64H ; carica il contatore B con il numero di dati
                 ; da prelevare
      LD  HL, 1B00H ; carica il puntatore alla zona di memoria
                    ; dove si dovranno inserire i dati
LOOP: IN  A, (20H) ; porta nell'accumulatore il dato
      LD  (HL), A  ; salvalo in memoria
      CP  00H      ; controlla se il contenuto dell'accumulatore è zero
      JR  Z, FINE  ; se si vai alla fine
      INC HL       ; altrimenti incrementa il puntatore
      DJNZ LOOP    ; controlla se B≠0 e preleva un nuovo dato
FINE: HALT
```

PROGRAMMA N.1
MOLTIPLICAZIONE

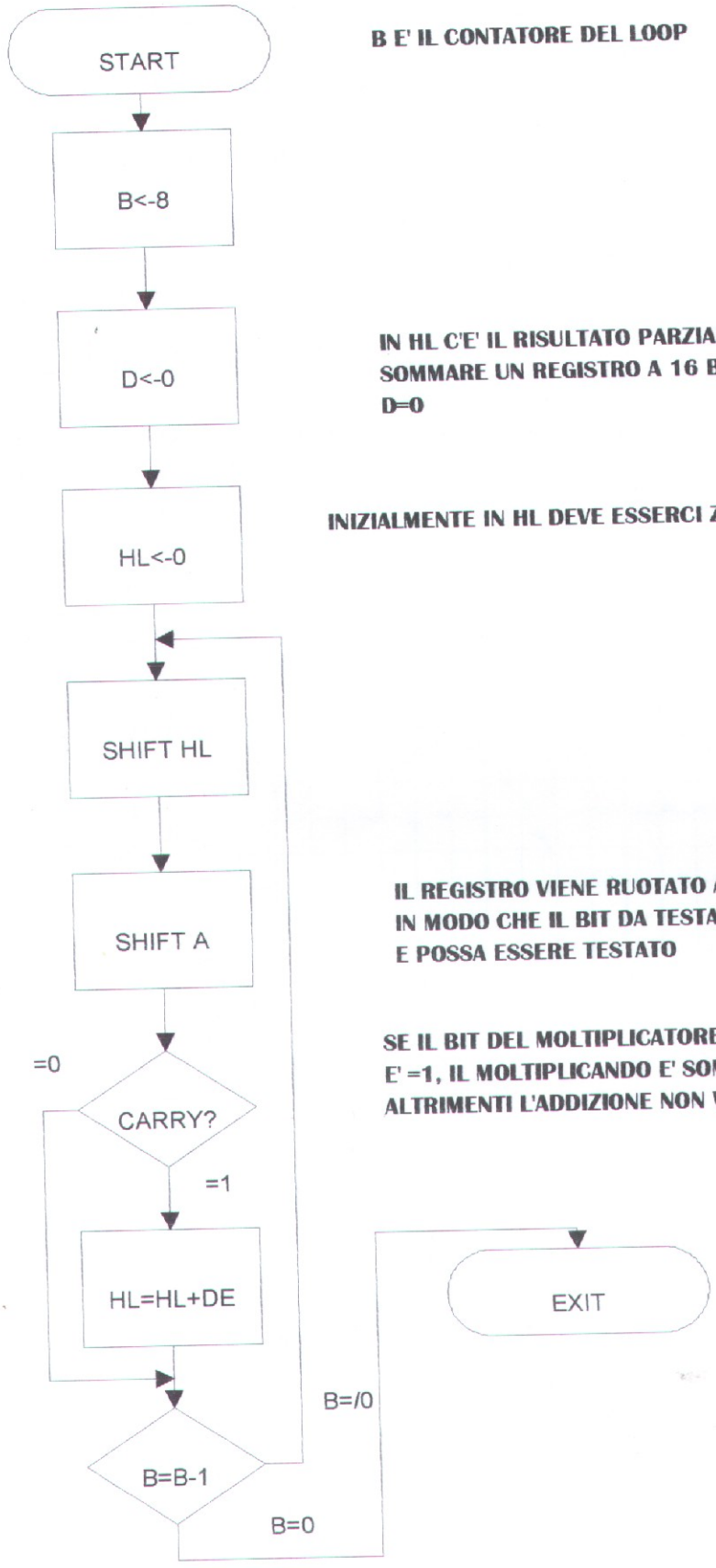
SI EFFETTUI LA MOLTIPLICAZIONE TRA IL MOLTIPLICANDO CONTENUTO NEL REGISTRO E ED IL MOLTIPLICATORE CONTENUTO NEL REGISTRO A ; IL RISULTATO VA MEMORIZZATO NEL REGISTRO HL.

ALGORITMO : SUPPONIAMO DI DOVER ESGUIRE LA SEGUENTE MOLTIPLICAZIONE IN BINARIO : 1100*0111

L'ALGORITMO E' LO STESSO CHE IMPLEMENTEREMMO IN DECIMALE

```
   1100*
   1101=
-----------
   1100+
  0000
 1100
1100
-----------
10011100
```

COME SI PUO' DEDURRE DALL'ESEMPIO ALLORA L'ALGORITMO PUO' ESSERE IL SEGUENTE : SI SCANDISCE LA STRINGA DI BIT DEL MOLTIPLICATORE DA DESTRA A SINISTRA : SE TROVIAMO IL BIT DI POSIZIONE N PARI AD 1 SOMMIAMO ALLA VARIABILE CHE DEVE CONTENERE IL RISULTATO FINALE, IL MOLTIPLICANDO SHIFTATO A SINISTRA DI N-1 POSIZIONI ALTRIMENTI NON FACCIAMO NIENTE. E' EQUIVALENTE ESEGUIRE LA STESSA COSA EFFETTUANDO UN CICLO CHE ANDRA' RIPETUTO UN NUMERO DI VOLTE PARI ALLA LUNGHEZZA IN BIT DEL MOLTIPLICATORE IN CUI SI SHIFTA A SINISTRA IL RISULTATO PARZIALE E SE IL BIT PRESO IN CONSIDERAZIONE DEL MOLTIPLICATORE E' PARI AD UNO AD ESSO SI SOMMA IL MOLTIPLICANDO. ALLA PAGINA SEGUENTE ABBIAMO LA FLOW-CHART.



B E' IL CONTATORE DEL LOOP

IN HL C'E' IL RISULTATO PARZIALE QUINDI AD ESSO DOBBIAMO SOMMARE UN REGISTRO A 16 BIT CIOE' DE PER CUI DEVE ESSERE D=0

INIZIALMENTE IN HL DEVE ESSERCI ZERO

IL REGISTRO VIENE RUOTATO A SINISTRA DI UNA POSIZIONE IN MODO CHE IL BIT DA TESTARE FINISCA NEL FLAG DI CARRY E POSSA ESSERE TESTATO

SE IL BIT DEL MOLTIPLICATORE ATTUALMENTE PIU' A SINISTRA E' =1, IL MOLTIPLICANDO E' SOMMATO AL RISULTATO PARZIALE ALTRIMENTI L'ADDIZIONE NON VIENE ESEGUITA.

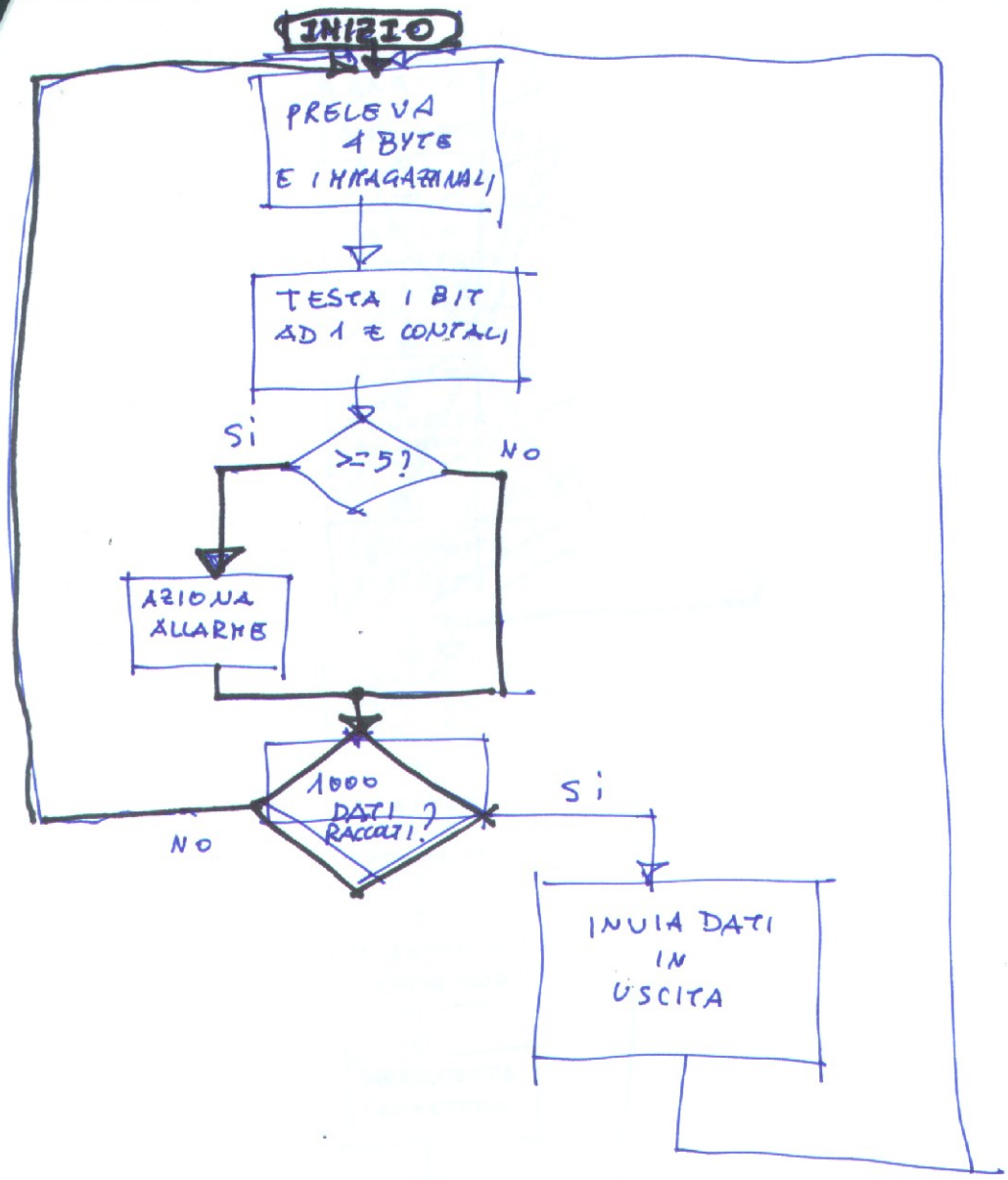
CODIFICA DEL PROGRAMMA.

```
; IL MOLTIPLICANDO SI TROVA IN E
; IL MOLTIPLICATORE SI TROVA IN A
; IL PRODOTTO NEL REGISTRO HL
        LD B,8              ;CARICA IL CONTATORE DEL LOOP
        LD D,0
;PER SOMMARE AD HL IL REGISTRO E DOBBIAMO SOMMARE IN
;REALTA' IL REGISTRO DE PER CUI DOBBIAMO ASSICURARCI CHE
;SIA D=0
        LD H,D
        LD L,D
; ALL'INIZIO ANCHE HL DEVE ESSERE SICURAMENTE A ZERO
LOOP    ADD HL,HL    ;SHIFTA HL A SINISTRA DI UNA POSIZIONE
        RLCA         ;RUOTA L'ACCUMULATORE A SINISTRA
;IL BIT PIU' SIGNIFICATIVO E' ORA NEL FLAG DI CARRY
        JR NC,NOADD
; TESTA IL FLAG DI CARRY E SE NON E' PARI AD UNO NON
; ESEGUE L'ADDIZIONE
        ADD HL,DE
NOADD   DINZ,LOOP    ;ESEGUE FINCHE' B E' DIVERSO DA ZERO
        HALT
```

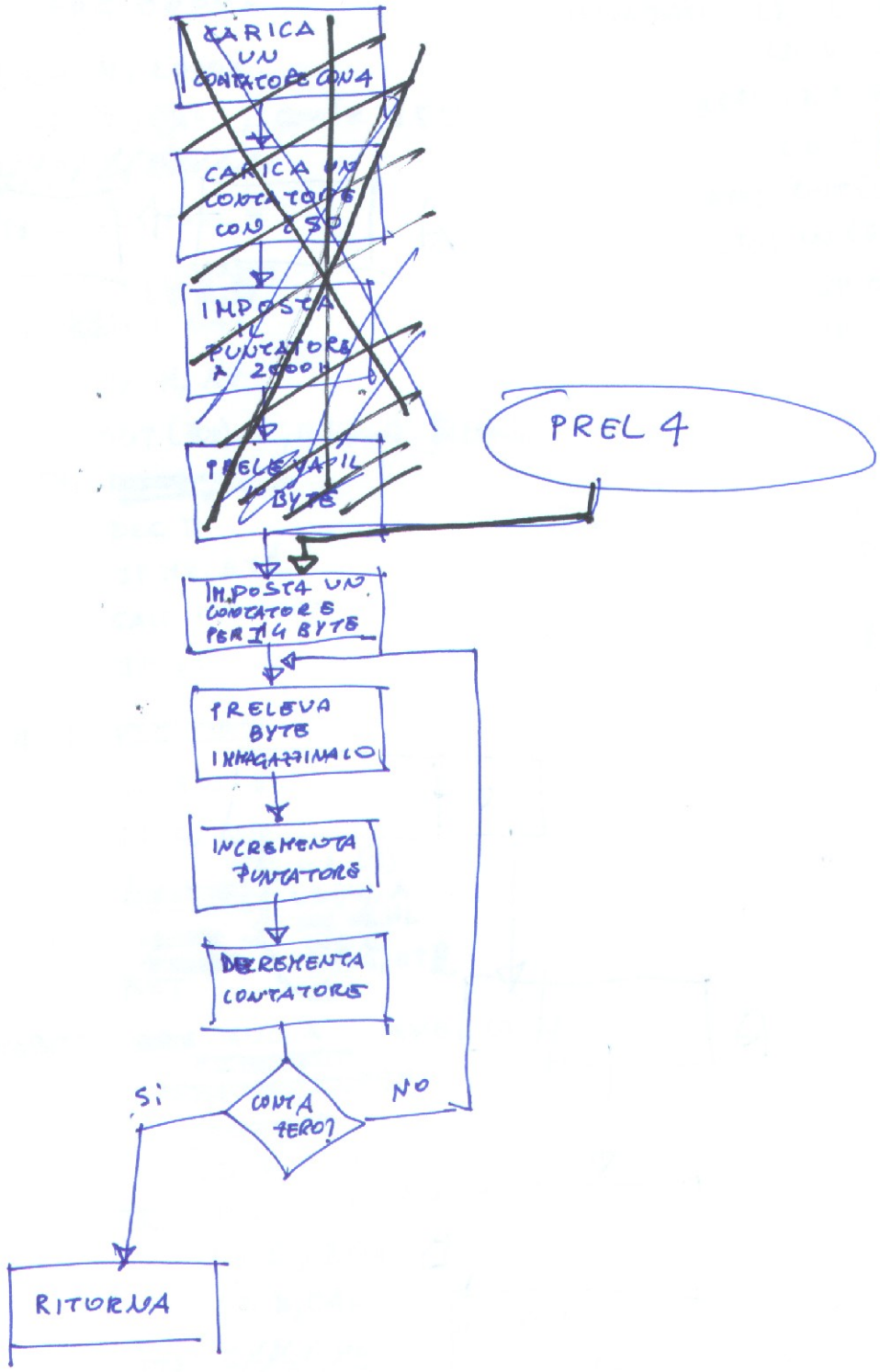
32 sensori A intelitem sono collegati in gruppi di 8 a 4 porte di indirizzo 20H, 21H, 22H, 23H. Questi sensori ad 1 indicano che le Temperature di un determinato punto del motore viene quel punto è eccessive. ~~il µP è collegato ad un~~ ~~contatore dovuto che, all'attraversamento~~ Il µP raccoglie questi dati e li immagazzina in memoria a partire dalla locazione 2000H. Raccolti 4 ~~da~~ byte dalle 4 porte, il µP ~~deve~~ testa ogni singolo b.i.t. Se almeno 5 sensori ~~segnalano~~ una Temperatura eccessiva il µP fa accendere una sirena di segnalazione. La sirena è collegata mediante circuito interfaccia al µP in modo da apparire come una porta di IO di indirizzo 30H. Il tentativo di scrittura di un qualunque dato su tale porta disattiva la sirena e permette ad un operatore di ridurre il numero di giri del motore. I dati immagazzinati non vanno altrimenti perché servono per creare un database.. Immagazzinati 1000 ~~dati~~ byte, il µP li invia ad una porta di uscita di indirizzo 40H costituita da un modem. La trasmissione è di tipo asincrono ~~tra~~, controllata dalla porta 50H. Per inviare un ~~dato~~ byte il µP deve prima inviare un numero 00000001 alla porta 50H che equivale ad un segnale Request to Send e poi deve attendere di ricevere una risposta 00000010 dalla ~~stessa~~ porta 50H (Clear to Send). Solo allora può inviare il dato in uscita alla porta 40H. Inviato il dato, il µP deve attendere dalla porta ~~50H~~ 50H il dato 00000100 che è un acknowledgment. Terminato l'invio dei dati il µP ~~si~~ riprende ad immagazzinare dati sulla Temperatura del motore ripetendo dalla locazione 2000H

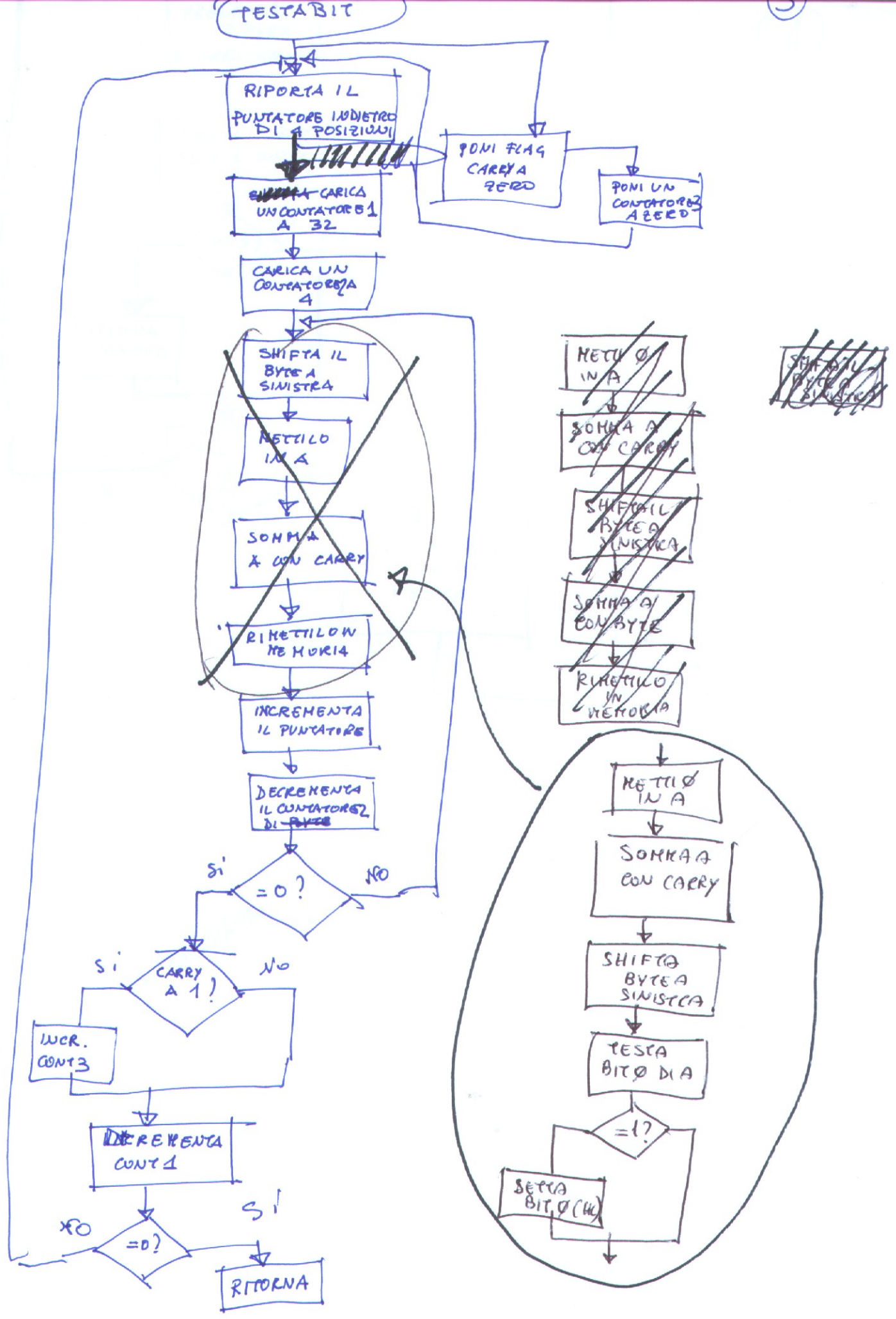
26/2/04

~~alvarez@tiscalinet.it~~


INIZIO
PRELEVA 4 BYTE E MEMORIZZALI
TESTA I BIT AD 1 E CONTALI
Si
>=5?
No
AZIONA ALLARME
1000 DATI RACCOLTI?
No
Si
INVIA DATI IN USCITA

1


CARICA UN CONTATORE CON 4
CARICA UN CONTATORE CON 0
IMPOSTA IL PUNTATORE A 2000H
PREL 4
PRELEVA IL 1° BYTE
IMPOSTA UN CONTATORE PER 14 BYTE
PRELEVA BYTE IMMAGAZZINALO
INCREMENTA PUNTATORE
DECREMENTA CONTATORE
CONTA ZERO?
SI
NO
RITORNA


TESTABIT
RIPORTA IL PUNTATORE INDIETRO DI 4 POSIZIONI
PONI FLAG CARRY A ZERO
PONI UN CONTATORE2 A ZERO
CARICA UN CONTATORE1 A 32
CARICA UN CONTATORE2 A 4
SHIFTA IL BYTE A SINISTRA
METTILO IN A
SOMMA A CON CARRY
RIMETTILO IN MEMORIA
INCREMENTA IL PUNTATORE
DECREMENTA IL CONTATORE2 DI BYTE
=0 ?
SI
NO
CARRY A 1 ?
SI
NO
INCR. CONT3
DECREMENTA CONT1
=0?
NO
SI
RITORNA
METTI Ø IN A
SOMMA A CON CARRY
SHIFTA BYTE A SINISTRA
SOMMA A CON BYTE
RIMETTILO IN MEMORIA
METTI Ø IN A
SOMMA A CON CARRY
SHIFTA BYTE A SINISTRA
TESTA BIT Ø DI A
=1?
SETTA BIT Ø (1)

```
ORG 0000H
ET: LD HL, 2000H
    LD D, FAH ; conta 250
ET1: CALL PREL 4
    CALL TESTABIT
    LD A,D
    CP 04H
    JP N, AV
    OUT (30H), A ; 2 uscite
AV: DJNZ, ET1
    DEC D
    JP NZ, ET1
    CALL INVIADATI
    JP ET

PREL 4 : PUSH BC
    LD B,04H
    LD C, 20H
ET2: IN A,(C)
    LD (HL), A
    INC HL
    INC C
    DJNZ, ET2
    RET

TESTABIT: SBC AND A ; azzera carry
    PUSH BC
    SBC HL, 0004H
    PUSH BC
    LD E, 00H
ET3: SBC HL, 0004H
    LD C, 20H
    LD B,04H
ET4: SLA (HL)
    LD A, (HL)
    ADC A, 00H
    LD (HL), A
AV3: INC HL
    DJNZ ET4
    JP NC, AV2
    INC E
AV2: DEC C
    JP NZ, ET3
    POP BC
    LD A, E
    BIT 0, A
    JP NZ, AV3
    SET 0, (HL)

INVIADATI: LD HL, 2000H
    LD D, 04H
ET8: LD B, FAH
    LD A, 01H
ET7: OUT (50H), A
ET6: IN (60H), A
    CP 02H
    JP NZ, ET6
    LD A, (HL)
    OUT (40H), A
    INC HL
    DJNZ, B ET7
    DEC D
    JP NZ, ET8
    RET
```

Esercizio

Il µP riceve un frame di dati dalla porta 30H. Il frame è organizzato in byte nel seguente modo: il primo byte dice la lunghezza del frame, l'ultimo byte è il checksum. Il checksum è ottenuto facendo la XOR fra inizialmente fra 00H e il 1° byte, successivamente fa XOR fra il risultato della XOR precedente e il byte successivo. Lo Z80 preleva tutti i dati a partire dalla locazione 1A00H. La lunghezza può essere al massimo di 200 byte. ~~immagazzinati i dati~~ Il µP controlla prima se la lunghezza massima è rispettata, se no invia un messaggio A3, e poi li immagazzina. Poi effettua il calcolo del checksum e lo confronta con quello inviato. Se tutto è ok viene inviato alla porta 30H un byte FF che è interpretato come un riscontro positivo. Se in caso ... [illegible] In ogni caso resta in attesa di nuovi dati. L'area di memoria utilizzata per immagazzinare i dati è di 10.000 locazioni. Quando si riempie il µP invia un messaggio FEH sulla porta per indicare che il buffer è pieno e non può ricevere altri frame.

Il byte di checksum non viene memorizzato

10.000

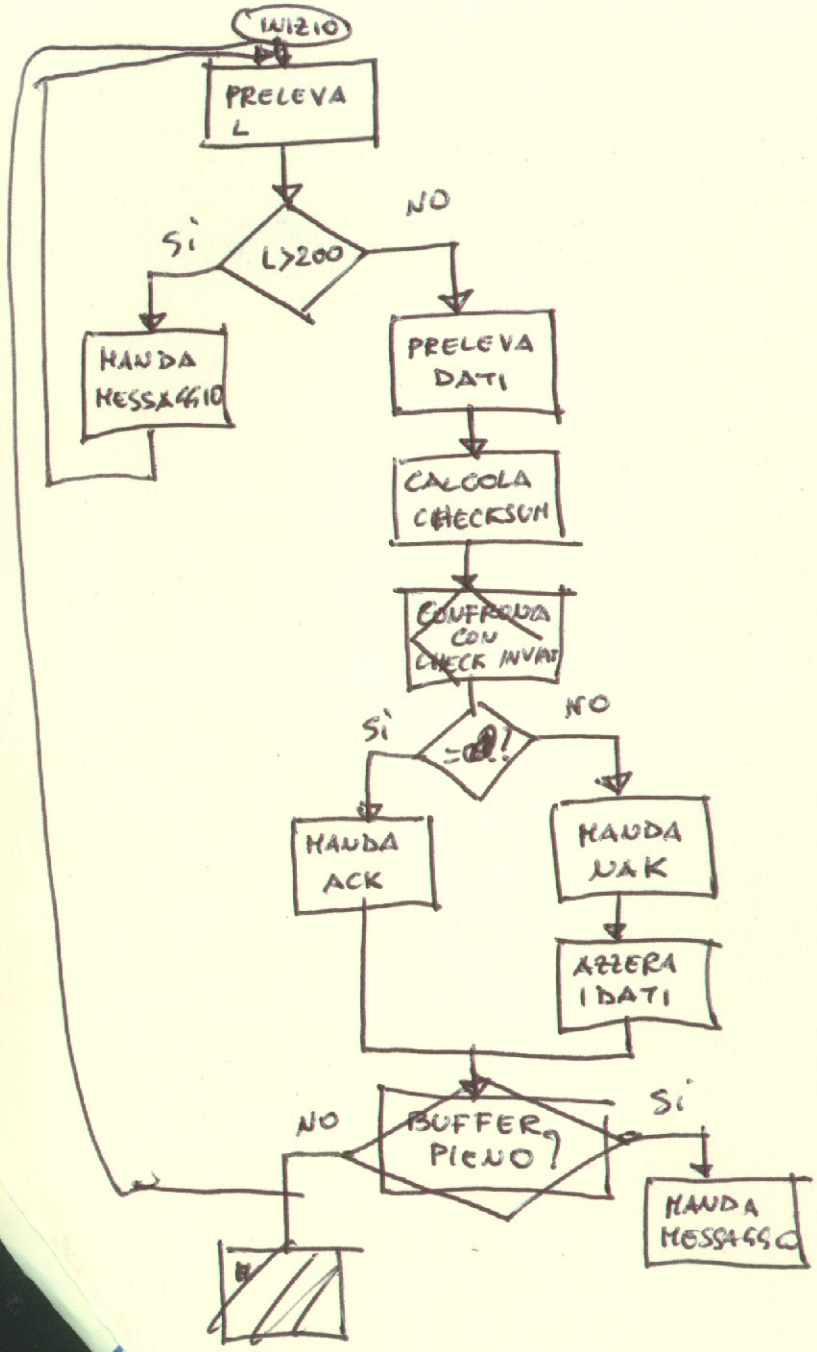
1A00H
270F
410F

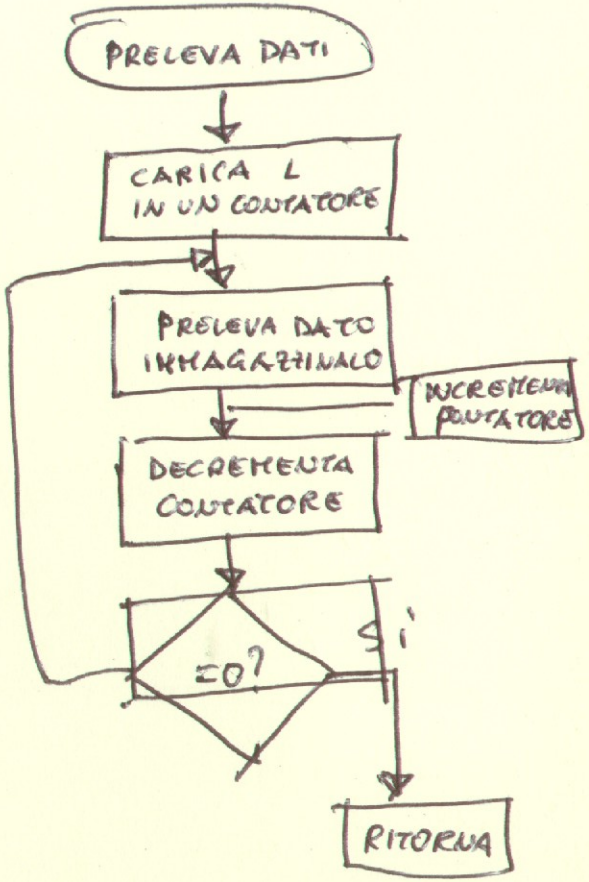
256

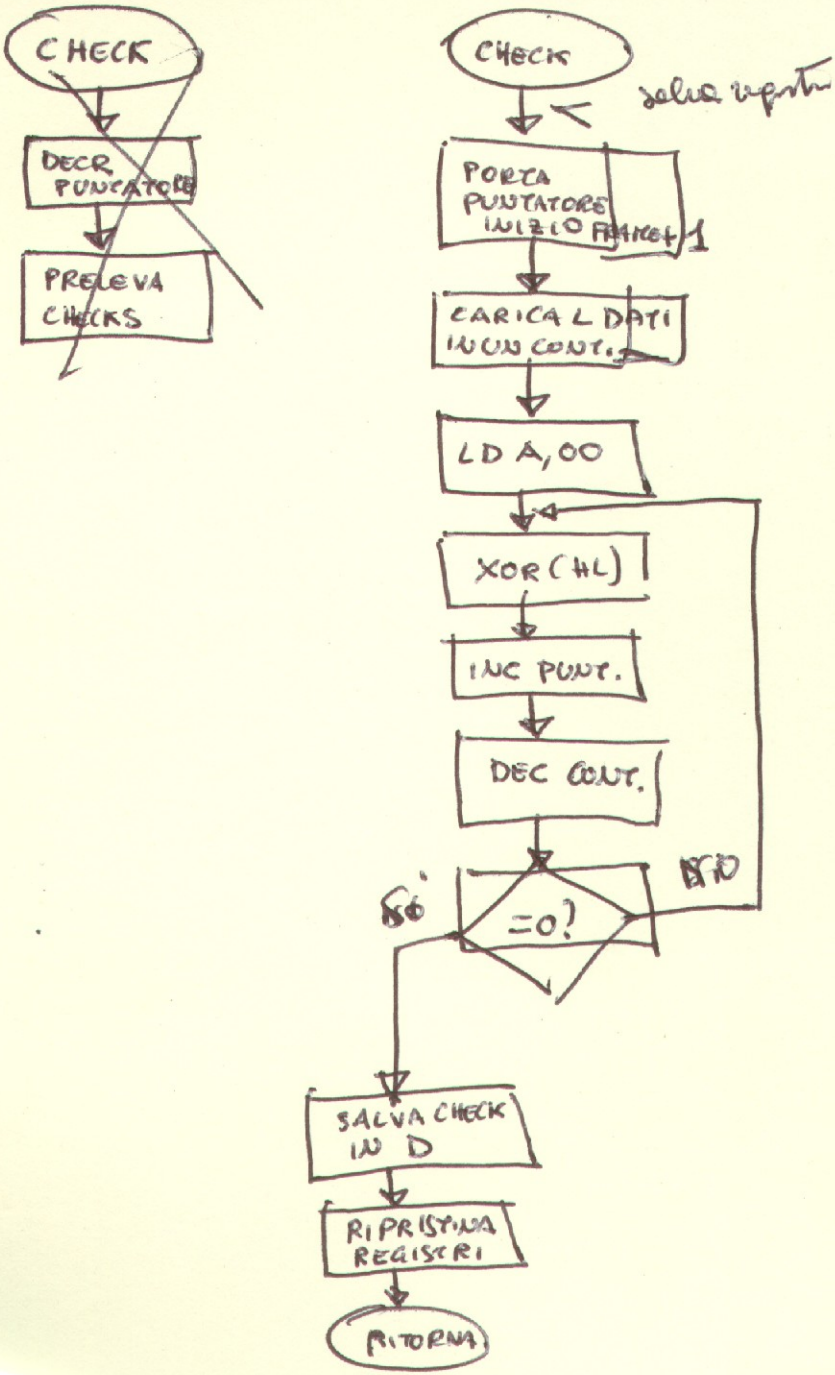
10

2710

EXL = A3
NAK FD
ACK FF
FBE FE

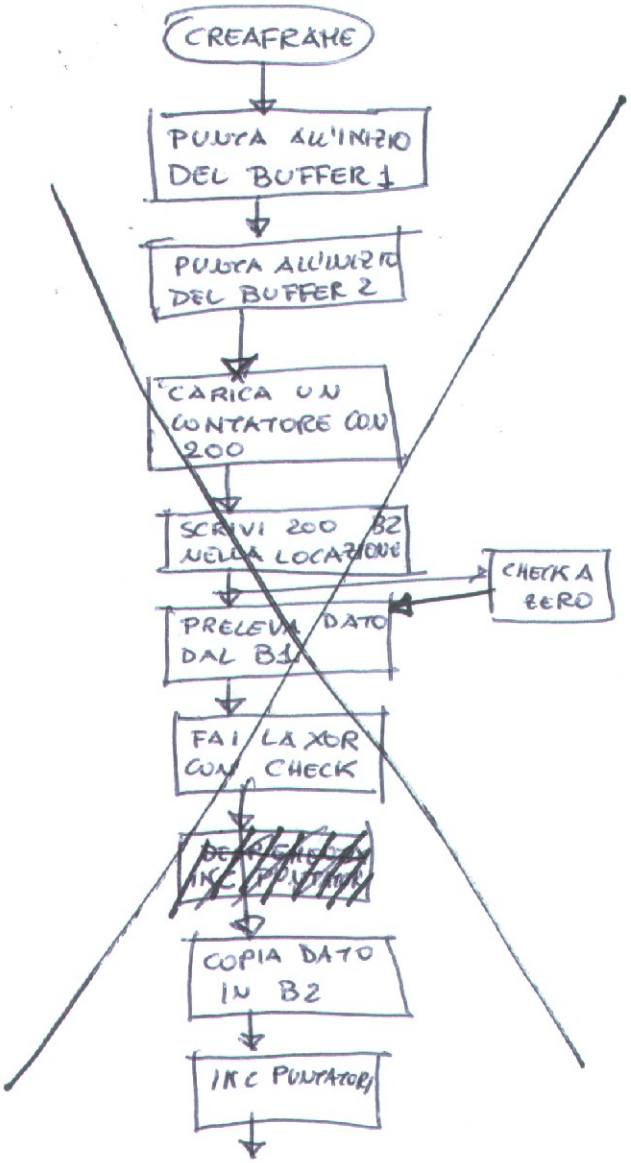

INIZIO
PRELEVA L
L>200
SI
NO
MANDA MESSAGGIO
PRELEVA DATI
CALCOLA CHECKSUM
CONFRONTA CON CHECK INVIATI
=0?
SI
NO
MANDA ACK
MANDA NAK
AZZERA I DATI
BUFFER PIENO?
NO
SI
MANDA MESSAGGIO

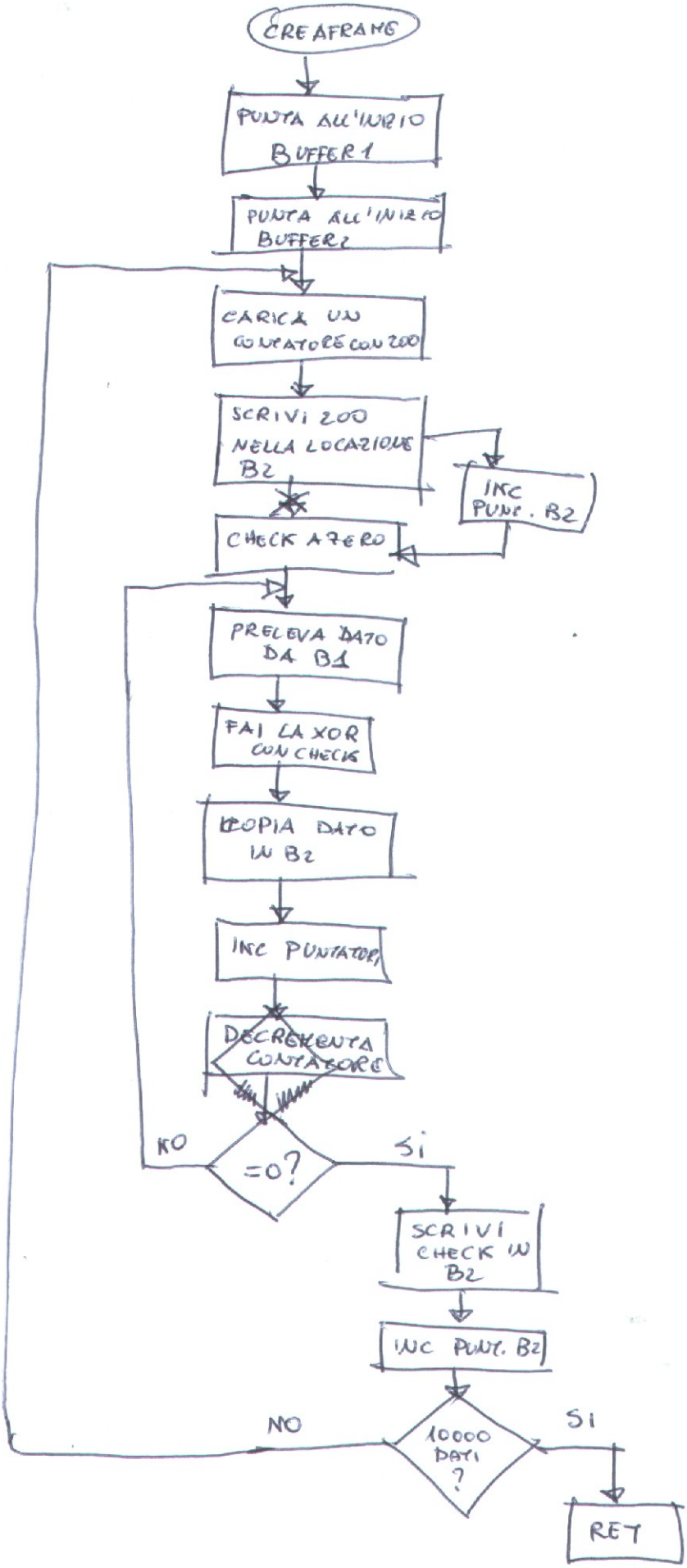

PRELEVA DATI
CARICA L IN UN CONTATORE
PRELEVA DATO IMMAGAZZINATO
INCREMENTA PUNTATORE
DECREMENTA CONTATORE
=0?
SI
RITORNA


CHECK
DECR PUNCATORE
PRELEVA CHECKS
CHECK
salva reg
PORTA PUNTATORE INIZIO
CARICA L DATI IN UN CONT.
LD A,00
XOR (HL)
INC PUNT.
DEC CONT.
=0?
SALVA CHECK IN D
RIPRISTINA REGISTRI
RITORNA

```
ORG 6000
IN A,(30H)
FRAME: LD HL,1A00H
ET: IN A,(30H)
    CP D2H
    JP N,ET1
    LD A,A3H
    OUT(30H),A
    JP ET
ET1: CALL PREL ; A contiene L
    CALL CHECK ; alla fine HL deve puntare al check
                 inviato
    DEC HL
    LD E,A  LD E,A
    LD A,D
    CP (HL)
    JP Z,ET2
    LD A,FDH
    OUT(30H),A
    LD E  LD A,E
    CALL AZZERA
    JP ET3
ET2: CALL LD A,FFH
    OUT(30H),A
ET3: LD A,E  AND A
    SBC HL  LD BC,4100H ; HL è andato
    SBC HL,BC             avanti di 1
    JP Z,FINE
    ADD HL,BC
    JP ET
FINE: LD A,FEH
    OUT(30H),A
    RET
```

Un microprocessore ha accumulato 10000 dati in un buffer di memoria che inizia alla locazione C000H. I dati vanno inviati ad una porta di uscita, ma vanno prima organizzati in pacchetti di 200 dati; ogni pacchetto va aggiunto un byte iniziale che indica la lunghezza complessiva del frame e un byte finale di checksum, il byte di checksum viene calcolato ponendolo prima a zero e poi ponendolo in OR esclusivo con ogni byte di dati. Scrivere un sottoprogramma CREAFRAME che legge i dati dal buffer e li organizza in frame e copia i frame in un buffer a partire dalla locazione D000H



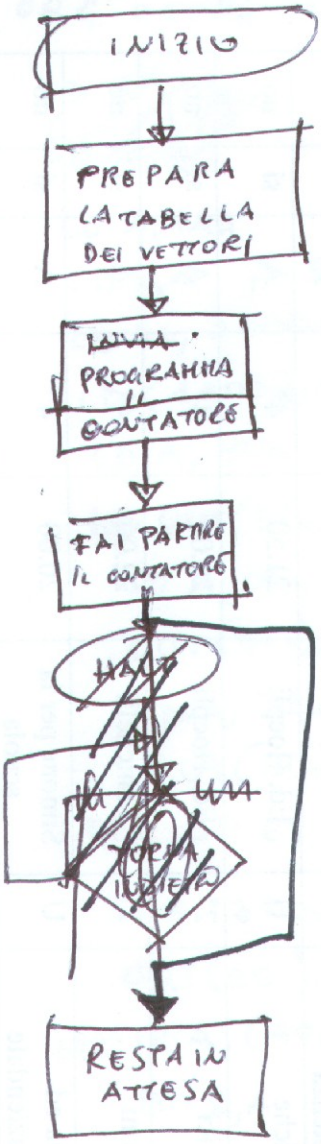

CREAFRAME
PUNTA ALL'INIZIO BUFFER 1
PUNTA ALL'INIZIO BUFFER 2
CARICA UN CONTATORE CON 200
SCRIVI 200 NELLA LOCAZIONE B2
INC PUNT. B2
CHECK A ZERO
PRELEVA DATO DA B1
FAI LA XOR CON CHECK
COPIA DATO IN B2
INC PUNTATORI
DECREMENTA CONTATORE
=0?
KO
SI
SCRIVI CHECK IN B2
INC PUNT. B2
10000 DATI ?
NO
SI
RET

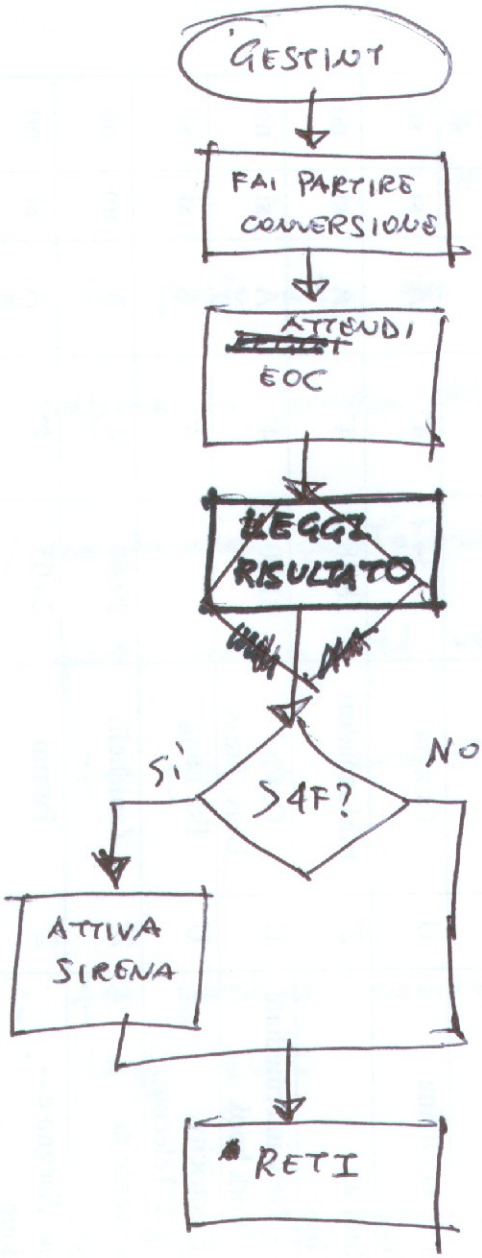
```
CREFRAME: LD HL, C000H
          LD IX, D000H

ET:  LD B, C8H
     LD (IX), B
     INC IX
     LD A, 00H
ET2: LD C, (HL)
     XOR C
     LD (IX), C
     INC IX
     INC HL
     DJNZ ET2
     LD (IX), A
     INC IX
     LD [illegible]
     LD [illegible]
     AND A
     SBC HL, E710H
     JP Z, FINE
     ADD HL, E710H
     JP ET
FINE: RET
```

1710
0000
E710

Un microprocessore è collegato ad un ADC, l'indirizzo per far partire una conversione è a 20H, basta effettuare una scrittura fittizia a questo indirizzo per far partire la conversione; l'indirizzo di lettura del risultato è 30H; l'indirizzo 40H è collegato al segnale $\overline{EOC}$ dell'ADC la lettura di un dato 00H che punta porta indica l'inizio che è terminata la conversione. All'indirizzo 50H si trova un contatore; ~~la scrittura~~ la scrittura di un numero su questo contatore comporta la partenza del contatore. ~~Quando~~ il quale decrementa il numero che vi è stato inserito ogni ~~1/5~~ ms. Quando il contatore arriva a zero, genera un'interruzione. Questa interruzione costringe il µP a ~~fa~~ far partire la conversione. Il µP usa il contatore per leggere i dati dell'ADC ogni 50 ms. ~~Un secondo contatore identico al I dati raccolti dal µP vengono immagazzinati in una zona di memoria a partire dall'indirizzo 2000H. Raccolti mille dati il µP smette di inviare alla porta all'indirizzo 60H.~~ Il µP è collegato ad una sirena di allarme in corrispondenza dell'indirizzo 60H. La scrittura di un dato qualsiasi su questo indirizzo fa scattare l'allarme. L'allarme deve partire quando il risultato della conversione supera il valore 4FH. ~~Usa la gestione delle int.~~ Il contatore usa la gestione delle interruzioni in modo ~~2~~ 2. L'indirizzo del suo programma di gestione delle interruzioni si trova all'indirizzo 3000H; la tabella dei vettori delle interruzioni si trova all'indirizzo 1A00H. Per programmare il contatore occorre scrivere all'indirizzo 51H un byte 80H che abilita la generazione delle interruzioni e successivamente allo stesso indirizzo la parte alta del vettore delle interruzioni.


INIZIO
PREPARA LA TABELLA DEI VETTORI
INVIA PROGRAMMA AL CONTATORE
FAI PARTIRE IL CONTATORE
HALT
RESTA IN ATTESA


GESTINT
FAI PARTIRE CONVERSIONE
ATTENDI EOC
LEGGI RISULTATO
SI'
NO
>4F?
ATTIVA SIRENA
IRET

```
ORG 0000H

LD A, 1AH
LD I, A ; salva la parte alta del vettore interruzioni
LD HL, 3000H
LD (1A00H), HL ; prepara la tabella del vettore interr.
LD A, 80H
OUT (51H), A ; programma contatore
  LD A, 00H
  OUT (51H), A ; parte bassa del vettore interruzioni
  LD A, 32H
  OUT (50H), A ; fai partire contatore
ET: JP ET

  ORG 3000H
  OUT (20H), A ; fai partire conversione
ET2: IN A, (40H) ; leggi EOC
  BIT 0, A
  JR NZ, ET2
  IN A, (30H) ; leggi risultato conversione
  CP 4FH
  JP M, ET3
  OUT (60H), A
ET3: RETI
```

50H
32H

```
LD B, L
LD E, L
DEC B
LD H, 00H
LD D, 00H
LOOP: ADD HL, DE
DEC B
JP NZ, LOOP
HALT
```

IN ALTERNATIVA ← JR NZ, LOOP

IN ALTERNATIVA ← DJNZ, LOOP

Scrivere un programma che fa il quadrato del numero contenuto in L e pone il risultato in HL

Programma 2

```
LD C, 50H
IN A, (C)
CP 00H
JP Z, FINE
LD B, A
LD HL, 8000H
LOOP: IN A, (C)
      LD (HL), A
      INC HL
      DEC B
      JP NZ, LOOP
FINE: HALT
```

PRIMA VERSIONE

IN ALTERNATIVA DJNZ, LOOP

```
LD C, 50H
IN A, (C)
CP 00H
JP Z, FINE
LD B, A
LD HL, 8000H
INIR
FINE: HALT
```

IN ALTERNATIVA → JRZ, FINE

Programma 1

Si prelevano dati da una porta di indirizzo 20_H e si immagazzinano in memoria a partire dall'indirizzo 8000_H; il primo dato prelevato da cui vuol dire i dati successivi da prelevare. Se è pari a 00_H, si interrompe il programma.

```
LD B, 64H
LD DE, 8000H
LD IX, 8200H
CICLO: LD A, (DE)
LD L, A
CALL QUAD ; HL contiene il quadrato
LD (IX+0), L
LD (IX+1), H
INC IX
INC IX
INC DE
DEC B
JP NZ, CICLO
HALT
ORG 3000H
QUAD: PUSH BC
PUSH DE
LD B, L
LD E, L
DEC B
LD H, 00H
LD D, 00H
LOOP: ADD HL, DE
DEC B
JP NZ, LOOP
POP DE
POP BC
RET
```

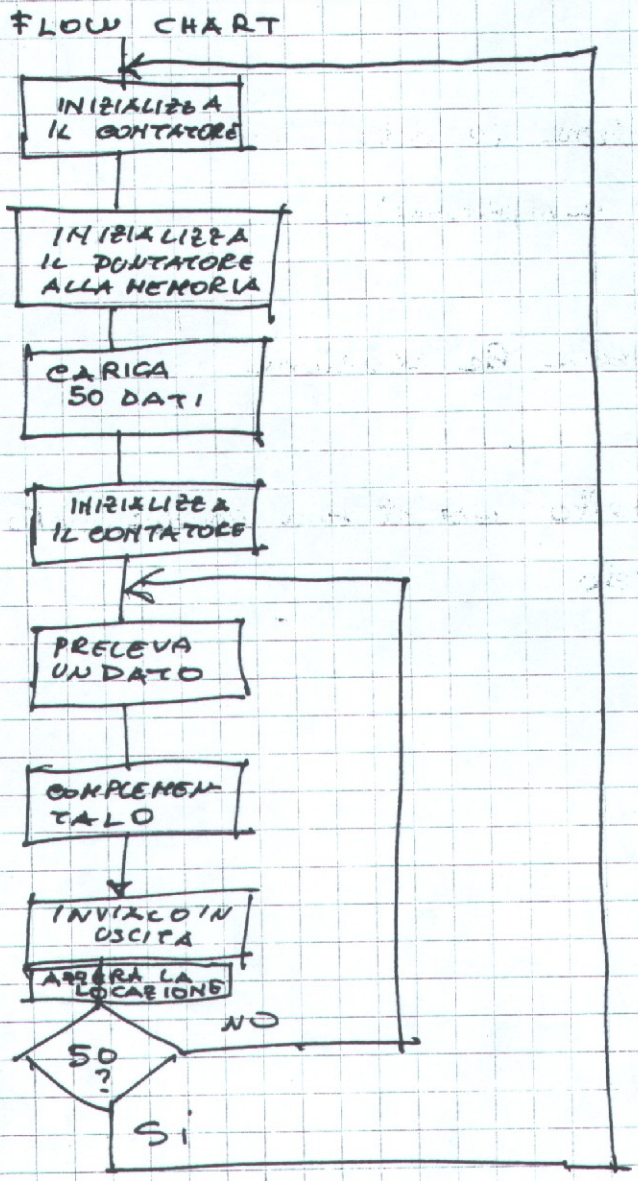
Programma 4

A partire dalla locazione 8000H ci sono 100 dati da 1 byte; a partire dalla locazione 8200H si devono memorizzare su 2 byte i quadrati dei 100 dati precedenti. Si usa a disposizione una sottoprogramma QUAD ~~dell'esempio precedente~~

PROGRAMMA 14

Si prelevano ~~100~~ dati dalla porta d'indirizzo 20H, quando si sono immagazzinati in memoria 50 dati, si complementano e si inviano in uscita alla porta di indirizzo 30H; si azzerano le locazioni utilizzate e si riprende il ciclo

FLOW CHART



CODIFICA

```
INIZ: LD C, 20H      ; C punta alla porta da cui prelevare i dati
      LD HL, 1A00H   ; HL punta all'area di memoria in cui conservare i dati
      HHHR LD B, 32H ; B è il contatore dei 50 dati
      INIR           ; l'istruzione incrementa HL anche l'ultima volta per
      DEC HL         ; cui HL è una posizione oltre il 50mo dato
      LD B, 50H      ;
LOOP: LD A, (HL)     ; prelevo il dato
      CPL            ; lo complemento
      OUT (30H), A
      LD (HL), 00H   ; azzero la locazione
      DEC HL
      DJNZ, LOOP
      JR INIZ HO     ; salto assoluto non condizionato per ripartire da
      H              ; zero
```