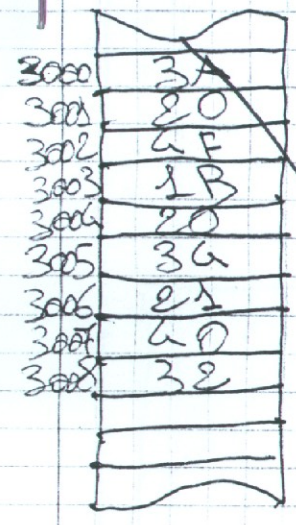
Es



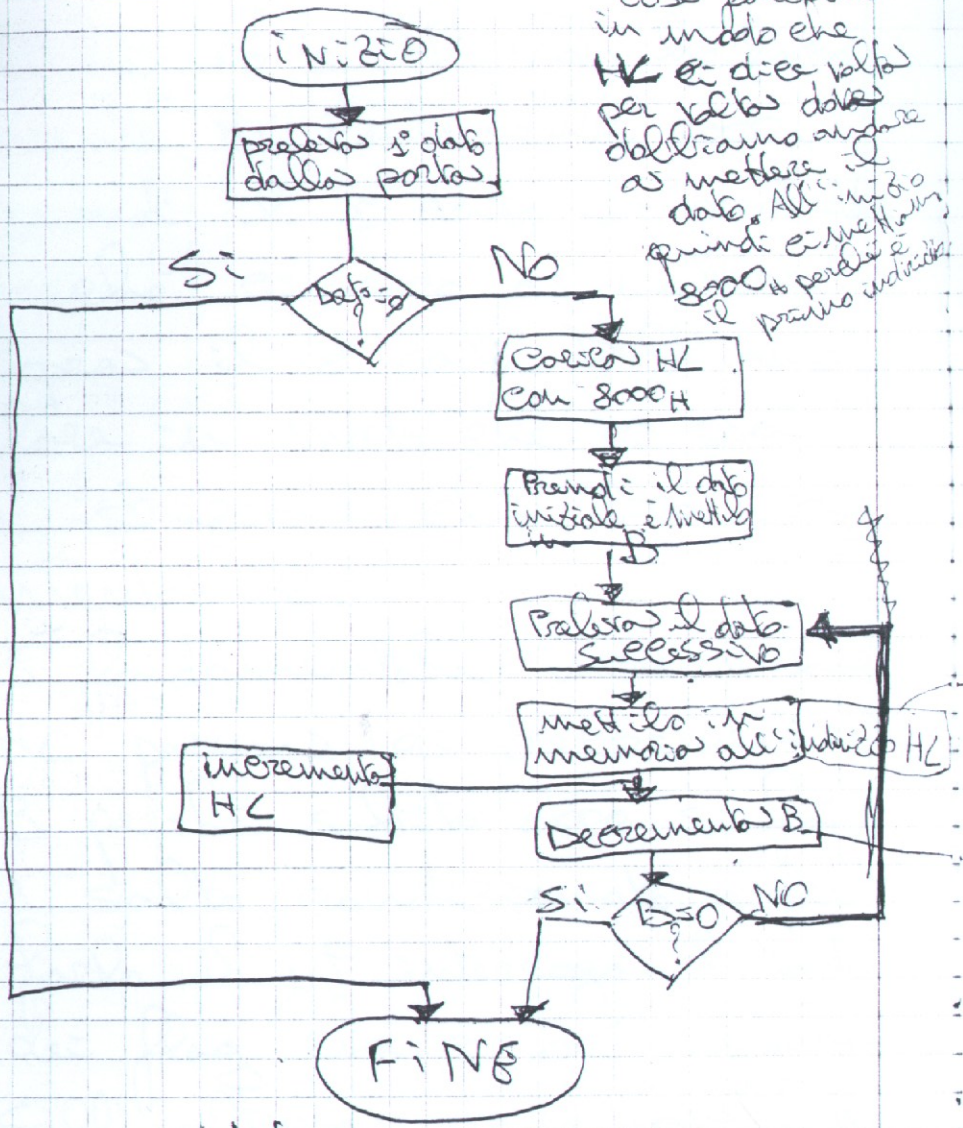
A 31
B 2A
C 10
D 32
E 12
H 30
L 00
IX 3002
IY 3000

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 prelevamo il 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 e 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