

ZX MASTER MIND



Master Mind. Il gioco in scatola

Il gioco, nella versione classica, consiste di una tavola di gioco e da pioli colorati di due tipi:

- Il primo tipo, sono quelli più grandi e sono di 6 colori diversi, vengono usati per la composizione del codice segreto e dei codici tentativi.
 - Il secondo tipo, sono più piccoli e di colore bianco e nero, vengono usati per la composizione dei codici chiave.
- (Nell'immagine sopra la mia scatola nella variante 8 colori comprata nei primi anni 80')

Regolamento del gioco

Il Master Mind è un gioco di logica dove due giocatori si alternano nei ruoli di Codificatore e Decodificatore. Il Codificatore è colui che sceglie e nasconde il codice segreto e il Decodificatore è il giocatore che lo deve indovinare.

L'obiettivo è quello di indovinare il Codice Segreto nel minore numero di tentativi possibili.

I due giocatori decideranno all'inizio quante partite effettuare e al termine confronteranno la somma dei tentativi.

Chi avrà il numero minore, vince.

Il codice segreto è composto da quattro pioli colorati tenuti nascosti da un coperchietto della tavola di gioco.

Il Decodificatore per tentativi successivi giocherà dei codici, anch'essi composti da pioli colorati e il Codificatore porrà dei Pioli Chiave di colore Bianco e Nero per ogni tentativo.

In base alle risposte del Codificatore, il Decodificatore deciderà il Codice successivo da giocare.

I Codici Chiave in risposta ad ogni tentativo seguono queste regole:

- Tanti pioli neri quanti sono i colori del Codice tentativo che sono presenti nel Codice Segreto e che sono anche nella posizione giusta.
- Tanti pioli bianchi quanti sono i colori presenti nel Codice Segreto ma che sono nella posizione sbagliata.
- Nessun Piolo Chiave per i colori del codice giocato non presenti nel Codice Segreto
- Nota: I Pioli Bianchi e Neri indicano la quantità di pioli colorati presenti nel Codice Segreto ma non la loro posizione.

Per evitare errori nelle risposte in caso di colori ripetuti, è preferibile che il Codificatore, metta prima i pioli neri e poi quelli bianchi facendo attenzione per quest'ultimi di escludere i colori già considerati nella risposta dei pioli neri. Nel seguente esempio il Codificatore mette un solo piolo nero per il blu presente e in posizione giusta. Nessun piolo bianco per l'altro blu. Nessun altro piolo per gli altri colori in quanto non sono presenti nel codice segreto



Possibili combinazioni di codici segreti.

Nella versione classica, il numero totale dei possibili Codici Segreti è calcolato secondo la formula del calcolo combinatorio delle Disposizioni con Ripetizione: $D=n^k$ dove nel nostro caso, n è il numero di colori e k il numero di posizioni dove i colori possono essere collocati, quindi 6^4 e cioè 1296 possibilità di codici segreti.

Per comodità, anche se impropriamente, tali disposizioni verranno chiamate combinazioni.

Esistono altre varianti del gioco con più colori e posizioni o versioni facilitate con meno colori/posizioni per bambini. Nella versione classica i colori sono bianco, blu, giallo, nero, verde. La variante a 8 colori aggiunge l'arancione e il marrone.

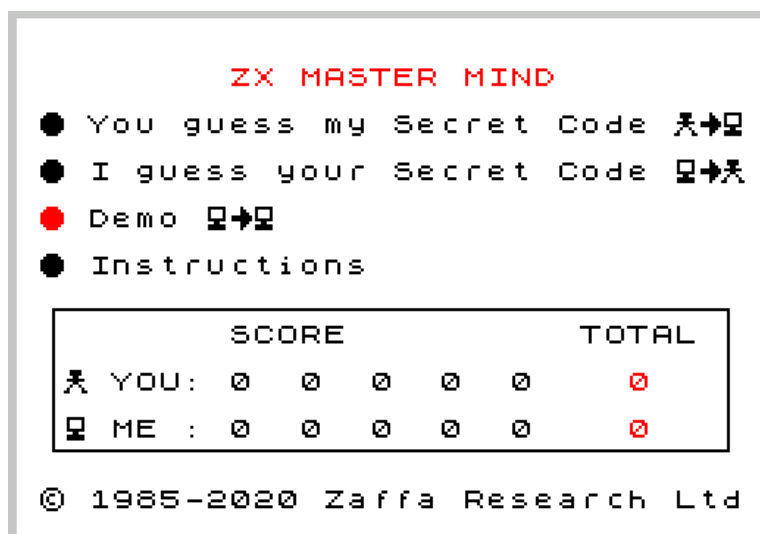
Rispetto ai colori del gioco in scatola, i colori usati dal programma, sono quelli dello ZX Spectrum e cioè: blu, rosso, magenta, verde, azzurro e giallo. I colori bianco e nero vengono usati solo per i codici chiave.

Modalità di gioco e opzioni del programma "ZX Master Mind"

Dal menu del programma sono disponibili le seguenti opzioni:

- Il giocatore indovina il codice segreto scelto in modo casuale dal programma
- Il programma indovina il codice segreto scelto dal giocatore
- Modalità Demo – Il programma indovina un codice segreto casuale
- Istruzioni

Sotto al menu è visualizzato il punteggio per le partite giocate tra il giocatore e il programma.



Di seguito i dettagli per ogni opzione:

Il giocatore indovina il codice segreto.

Il giocatore è il decodificatore e il programma il codificatore

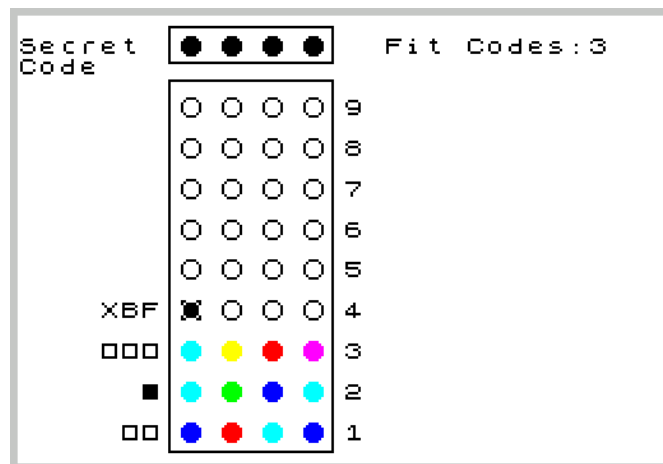
Il programma sceglierà un codice segreto casuale che terrà oscurato nella parte alta della tavola.

Ad ogni codice tentativo del giocatore, il programma attribuirà un codice chiave (pioli bianchi e neri).

L'opzione "X" abbandona la partita e si ritorna al menu.

Per le opzioni "B" ed "F", vedere il paragrafo: "Opzioni F e B"

In alto a destra vengono visualizzati i "Fit Codes" che sono il numero dei codici segreti rimanenti dopo ogni tentativo.

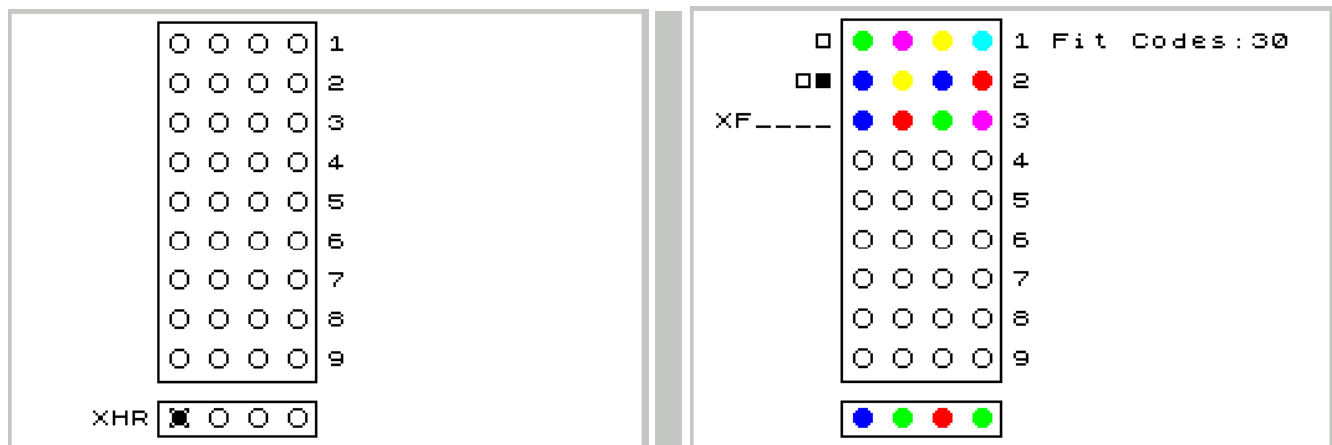


Il programma indovina il codice segreto.

Il programma è il decodificatore e il giocatore il codificatore.

All'inizio della partita il giocatore ha tre possibilità per scegliere il codice segreto:

- 1) Scegliere un colore per ogni posizione
- 2) Usare l'opzione "R" (Random) per generare un codice segreto casuale
- 3) Usare l'opzione "H" (Hidden) per non inserire il proprio codice segreto ma scriverlo su un foglio.



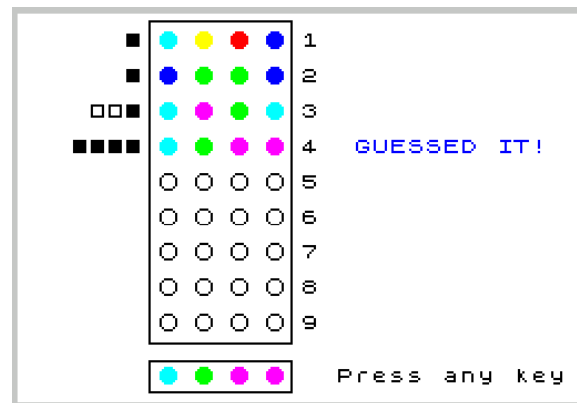
Dopo aver inserito il codice segreto, il programma inizierà a giocare il primo tentativo a cui si dovrà fornire un codice chiave.

L'opzione "X" abbandona la partita.

L'opzione "F" verrà trattata nel paragrafo: "Opzioni F e B"

Il programma indovina un codice segreto casuale – Modalità Demo

Il programma giocherà sia da Codificatore che da Decodificatore. Sceglierà in modo casuale un Codice Segreto e giocherà fino a quando non lo indovina. Questa modalità può essere usata per capire come si svolge il gioco.

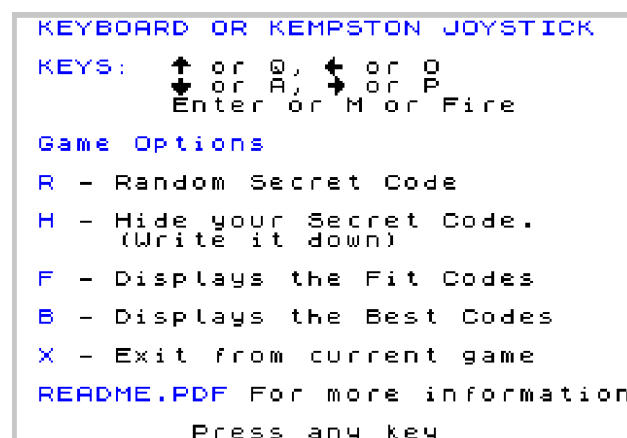


Istruzioni. Uso della tastiera e del Joystick Kempston

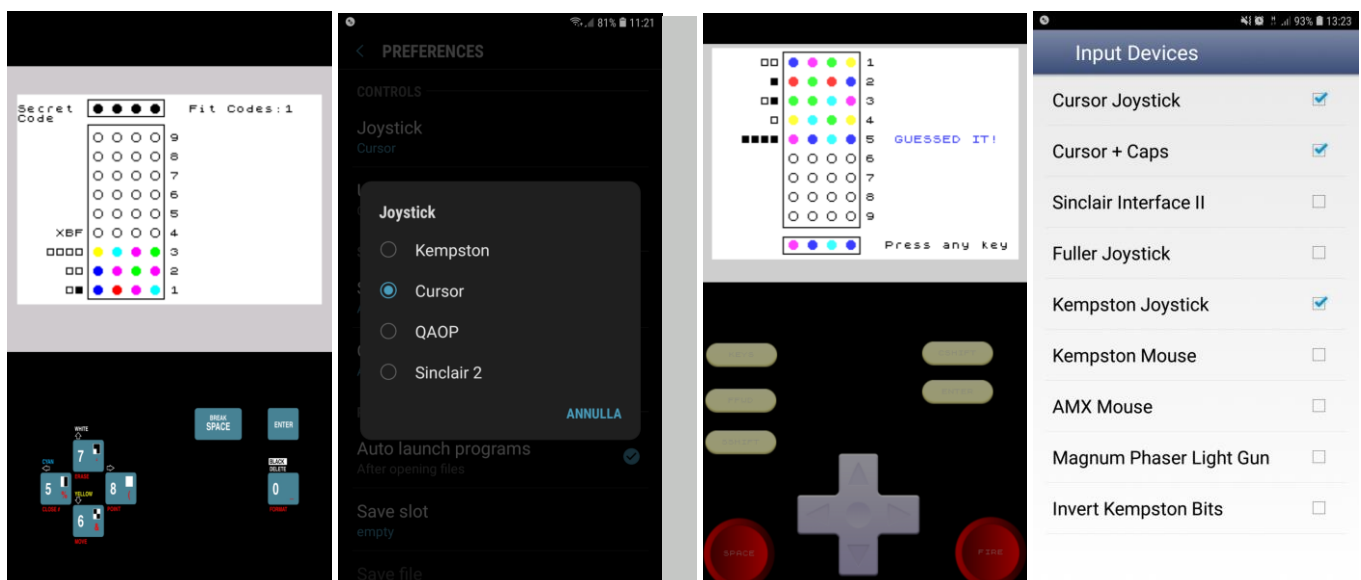
Per selezionare i pioli colorati e le opzioni del programma, usare i tasti cursori o il Joystick Kempston.

La scelta viene confermata con il tasto "Invio" o "Fire" del Joystick.

Giocando con un Spectrum con tasti in gomma è meglio usare i tasti alternativi: Q A O P come tasti cursore e M come INVIO.



Utilizzando un emulatore dello ZX Spectrum per Android è preferibile scegliere come dispositivo di input nelle preferenze i "tasti cursore". L'immagine che segue fa riferimento a: "Unreal Speccy Portable" e a "Speccy"



Caricare ed eseguire il programma.

Per gli emulatori ZX Spectrum, aprire il file auto avviante MMEMUXX.TAP (XX è la versione).

È stato testato con Fuse per Windows, Speccy e Unreal Speccy per Android.

Per lo ZX Spectrum, il classico LOAD "" e poi riprodurre il file MMVAWXX.WAV da un lettore MP3/Smartphone/PC collegato tramite cavo Jack alla porta "EAR" dello Spectrum. Testato sul mio ZX Spectrum +.

Opzioni "F" e "B"

Queste opzioni non sono necessarie per il normale svolgimento del gioco, tuttavia posso essere usate se si è in difficoltà a trovare un codice tentativo oppure per migliorare la propria strategia di gioco.

Opzione "F" (Fit Codes)

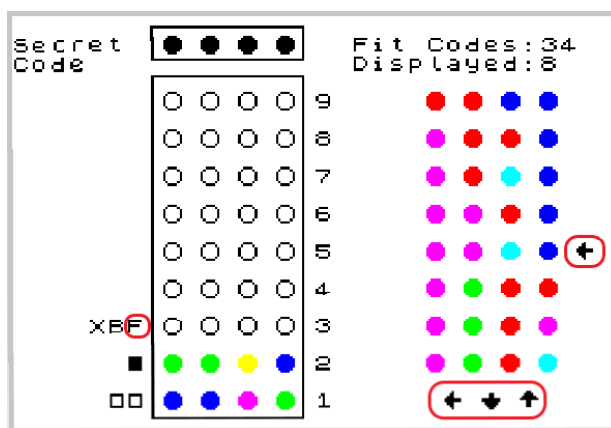
Questa opzione visualizza i codici segreti possibili chiamati dal programma Fit Codes. Tra questi c'è il reale codice segreto.

In alto a destra è visualizzato il totale dei Fit Codes e quanti sono stati visualizzati.

Sono visualizzati a gruppi da otto. Si passa al gruppo successivo usando il tasto freccia "Giù".

La freccia "Su" abilita un cursore che può scorrere la lista dei fit codes per scegliere il codice tentativo che deve poi essere confermato con il tasto INVIO.

Per tornare indietro, premere il tasto cursore con freccia a sinistra.



Opzione "B" (Best Codes)

Visualizzazione dei Best Codes. I Best Codes sono selezionati da un algoritmo che individua i codici migliori da giocare.

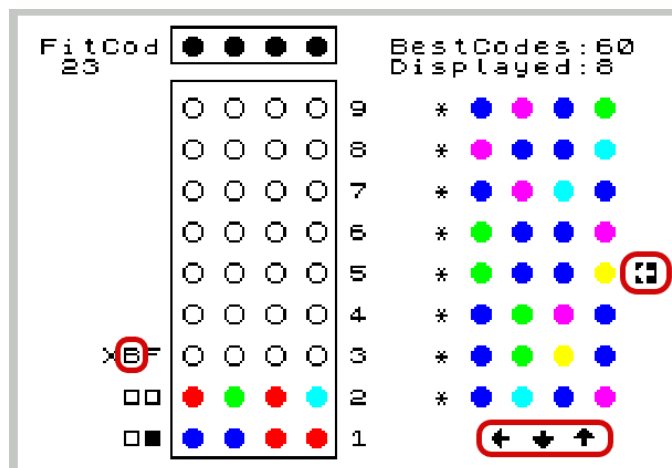
Durante l'elaborazione, un contatore indica la progressione dei codici da processare.

Giocare i Best Codes, in termini probabilistici, riduce il numero di tentativi per indovinare il codice segreto.

Se i Best Codes sono contrassegnati con un asterisco, sono di tipo inconsistente, cioè non potranno indovinare il codice segreto nel momento in cui vengono giocati ma offrono comunque migliori probabilità di arrivare al codice segreto con un minor numero di tentativi rispetto alla scelta di un codice tentativo tra fit codes.

In alto a destra è visualizzato il totale dei Best Codes e quanti di questi sono stati visualizzati.

Per navigare tra Best Codes e fare la selezione, le modalità sono le stesse come per i Fit Codes



Come funziona l'algoritmo "Best Codes"

Iniziamo a capire come vengono individuati i Fit Codes assumendo che il programma che gioca come decodificatore.

1) All'inizio il programma genera tutti i 1296 codici che tiene in memoria. In questa fase questi codici sono tutti Fit Codes.

2) Alla prima giocata il programma seleziona un Fit Code in modo casuale ad esclusione per i codici con un colore ripetuto 3 o 4 volte. Sono i codici peggiori per iniziare.

3) Confronta il suo Codice tentativo con ognuno dei Fit Codes. Tutti quelli che generano un Codice chiave diverso da quello ottenuto dal giocatore, vengono scartati.

4) Il programma gioca un Fit Code di quelli rimasti.

5) Il programma ripete il procedimento dal punto 3).

Ora, premesso che ogni Fit Code ha la stessa probabilità degli altri Fit Codes di indovinare il Codice Segreto, nel caso però che non lo indovini, in base al Codice Chiave ricevuto rimarranno tanti o meno Fit Codes per il tentativo successivo.

L'obiettivo dell'algoritmo è quello di far rimanere, in termini probabilistici, il minor numero di Fit Codes.

Ovviamente, la componente fortuna non può essere eliminata e mantiene un ruolo significativo.

Prima versione dell'algoritmo:

Sviluppai questa versione nel 1985. È la routine "Simula" inclusa nei vecchi listati che riporto nelle immagini più avanti. Nel programma attuale questa routine è stata migliorata.

Una copia della lista dei Fit Codes viene usata come lista dei Codici Segreti possibili.

Per ogni Fit Code viene associato un punteggio che viene determinato come segue: viene simulato di giocare il primo Fit Code e viene ipotizzato che il Codice Segreto sia il primo della sua lista. In base al Codice Chiave generato vengono contati tutti i Fit Codes compatibili con il Codice Chiave. Il numero ottenuto va a sommarsi al punteggio del primo Fit Code.

Questa operazione viene ripetuta ipotizzando che il codice segreto sia il secondo dalla lista ottenendo un nuovo numero da sommare al punteggio del primo Fit Code.

Terminati tutti i Codici Segreti, si passa al secondo Fit Code e si ripete la simulazione per tutti i Codici Segreti per ottenere il punteggio anche per il secondo Fit Code.

Il procedimento sopra, viene ripetuto per i Fit Codes rimanenti.

Al termine, i Fit Codes con il punteggio più basso saranno eletti Best Codes.

Se ci saranno più Fit Codes con lo stesso punteggio, saranno tutti eletti a Best Codes.

Questo algoritmo indovina il Codice Segreto in media alla 4,54 giocata. Risultato ricavato da 1000 partite giocate.

Versione migliorata:

Viene fatta una copia della lista dei Fit Codes e questa viene considerata la lista dei Codici Segreti Possibili.

Ora viene simulato di giocare il primo Fit Code della lista e che il Codice Segreto sia il primo della rispettiva lista. In questo caso, poiché i due codici sono identici, il Codice Chiave sarà 4 neri.

Per ogni Fit Code, viene preventivamente definita una tabella come quella in figura sotto che contiene tutti i Codici Chiave possibili e ogni casella è valorizzata a zero.

In questo caso verrà incrementata casella "4 neri" di uno.

	0 Neri	1 Nero	2 Neri	3 Neri	4 neri
0 Bianchi	0	0	0	0	0
1 Bianco	0	0	0		
2 Bianchi	0	0	0		
3 Bianchi	0	0			
4 Bianchi	0				

L'operazione viene ripetuta tenendo fisso il primo Fit Code e passando al secondo Codice Segreto.

Anche qua, in base al Codice Chiave ottenuto, verrà incrementata la casella corrispondente.

Il processo viene ripetuto per tutti i Codici Segreti.

Al termine, i valori nelle caselle indicheranno il numero dei Fit Codes che rimarrebbero se venisse giocato il primo Fit Code e il Codice Chiave fosse quello relativo alla casella stessa.

Ad esempio, se la casella 2 Neri e 2 Bianchi conterrà 3, significa che giocando il primo Fit Code e ricevendo come Codice Chiave 2 Neri e 2 Bianchi, i Fit Codes rimanenti per la giocata successiva saranno 3. (Ovviamente se non viene indovinato il codice segreto)

A questo punto, l'algoritmo associa al primo Fit Code il valore massimo (MAX) che trova in tabella. MAX è quindi il numero massimo di Fit Codes che possono rimanere se viene giocato il primo Fit Code. L'algoritmo ripete ora lo stesso processo per il Fit Code successivo, ottenendone il relativo MAX. Continua così fino al termine di tutti i Fit Codes.

Ora verranno controllati tutti i Fit Codes. Quello che avrà il valore MAX più basso, sarà eletto Best Code. Se altri Fit Codes avranno lo stesso MAX, anche questi saranno Best Codes.

La base di elaborazione dei Fit Codes può essere estesa a tutti i codici (1296) mettendo in concorrenza i Fit Codes con i codici inconsistenti, (codici che non possono essere il codice segreto). Ebbene, talvolta i codici inconsistenti hanno un MAX minore di quello ottenuto dai Fit Codes, e quindi potranno essere giocati lasciando un minor numero di Fit Codes per il tentativo successivo.

Se parte dei Fit Codes e alcuni codici inconsistenti riceveranno lo stesso valore di MAX, solo i primi verranno eletti a Best Codes quanto questi hanno anche la possibilità di indovinare il codice segreto.

Poiché il tempo di elaborazione aumenta in base al numero di codici processati, occorre limitare l'uso dell'algoritmo per evitare tempi di attesa eccessivi che possano annoiare il giocatore.

In caso il programma è decodificatore, l'algoritmo viene eseguito se i fit codes sono inferiori o uguali a 317.

Se i fit codes sono minori o uguali a 50, l'algoritmo considera tutti i 1296 codici per l'elaborazione.

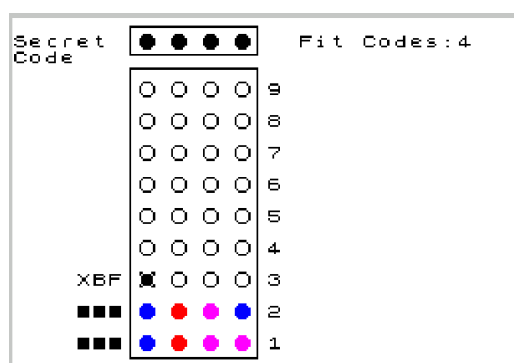
Stesse modalità sopra quando è il giocatore ad essere il decodificatore e seleziona l'opzione "B".

Scegliendo come primo tentativo un codice che non sia un colore ripetuto 4 volte, rimangono al massimo 317 Fit Codes e quindi, a meno del caso sopra, al secondo tentativo può essere eseguito l'algoritmo.

Quando poi nei tentativi successivi il numero di Fit Codes scenderà, 50 sarà un buon compromesso per estendere l'uso dell'algoritmo a tutti i 1296 codici con la possibilità di avere come Best Codes anche codici inconsistenti.

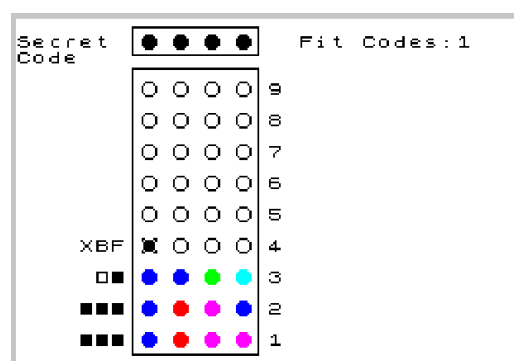
Esempio di utilizzo di un codice inconsistente.

Ipotizza di esser giunto a questa situazione di gioco:



Non sembra male, 3 neri al primo tentativo e anche al secondo che confermano che i primi 3 colori sono giusti e anche al posto giusto. A questo punto basta ripetere il codice cambiando l'ultimo colore con uno dei 4 possibili. Così facendo però, hai il 25% di probabilità di indovinare il codice segreto al terzo tentativo e se sei sfortunato, lo indovinerai al sesto tentativo.

Se invece giochi un codice inconsistente come quello nella figura sotto:

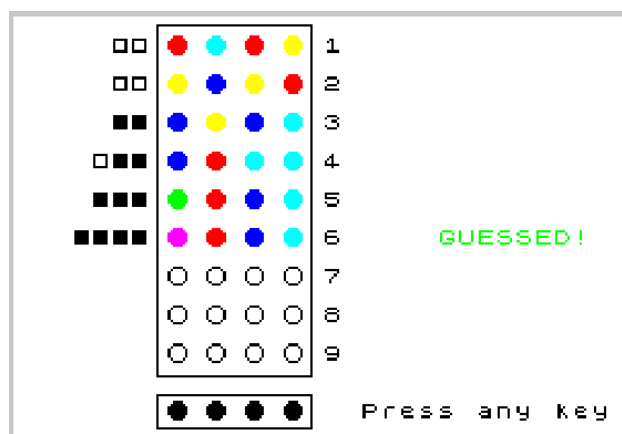


Avrai la certezza di indovinare il codice segreto al quarto tentativo.

Al quinto tentativo, se nella medesima situazione il primo codice giocato sono 4 colori diversi.

Quanti tentativi servono al programma per indovinare il codice segreto?

Al massimo 5. La media è di 4,47. Il programma potrebbe far meglio ma i tempi di attesa per l'elaborazione dei codici si allungherebbe come spiegato in precedenza. In rari casi e solo se il tentativo iniziale è del tipo 1123, cioè con un solo colore ripetuto, potrebbero essere necessari 6 tentativi come nell'esempio sotto.

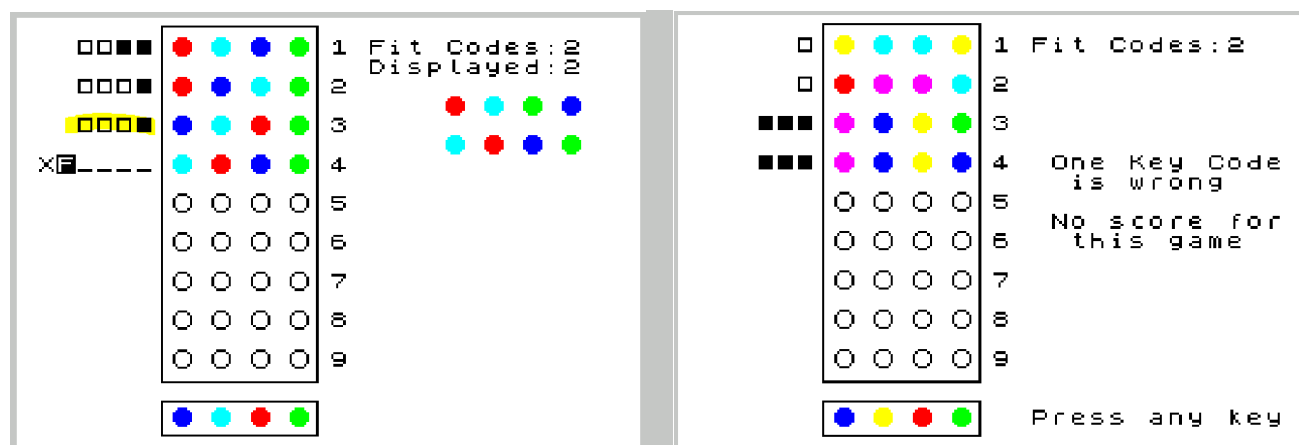


I tentativi iniziali di tipo 1112 e 1111, sono quelli che implicano un maggior rischio di arrivare ad indovinare il codice segreto al sesto tentativo o oltre e quindi non sono mai giocati dal programma.

Si può barare dando codici chiave errati?

Nell'immagine sotto, il programma indovina il codice segreto al terzo tentativo ma il giocatore bara rispondendo con tre bianchi e un nero. Poiché questo codice chiave lascia ancora due possibilità, il programma procede con il tentativo successivo.

Nel caso invece, un codice chiave errato non lascia tentativi possibili, il gioco si interrompe e un messaggio avvertirà il giocatore che ha sbagliato almeno un Codice Chiave (Key Code).



"ZX Master Mind". Breve storia di come è andata.

Negli anni 80', microcomputer come ZX Spectrum e Commodore 64, diedero alla mia generazione la possibilità di avere un computer a casa senza spendere grandi cifre e poter imparare a programmare, anche se la grande diffusione di questo tipo di computer era più legata alla possibilità di utilizzarsi come console per giochi. Giochi che si potevano acquistare in originale o trovare in audio cassette allegati a riviste acquistate in edicola.

Iniziai a scrivere il programma per il Master Mind nel 1985 quando ero studente alle scuole superiori a indirizzo Informatico. Trovavo sfidante implementare nei programmi la logica dei giochi come ad esempio Battaglia Navale, Forza 4 e Tetris. Scrivevo la parte della logica dei giochi, quella che mi dava più soddisfazione, ma poi perdeva interesse e i giochi rimanevano incompleti.

Master Mind fu quello che mi impegnò per più tempo perché l'obiettivo di fare un programma che potesse giocare meglio di un buon giocatore umano, mi piaceva molto.

Da ciò avrei cercato di imparare le strategie di gioco migliori per diventare più bravo anzi il più bravo tra gli amici. :-). Non sapevo fin dove sarei riuscito ad arrivare, ma mi accorsi ben presto che non sarebbe stato facile perseguire il mio obiettivo. Il codice sviluppato in Basic era troppo lento per le elaborazioni che avevo in mente e così la parte di calcolo dovetti convertirla in Assembler.

Un'altra sfida era la gestione della memoria, dovevo riuscire a far stare in memoria il programma in basic, il sorgente in assembler e il compilato, due assembleri: Zeus, che ho usato per scrivere l'assemblatore e Champ, per la fase di debug istruzioni per istruzione e infine le 1296 combinazioni che occupavano circa 13K.

Alla fine, il "cuore" del programma, che erano poi le routine in Assembler c'erano. Ma come per altri programmi, rimase incompleto.

Da allora non ho più fatto nulla e le uniche volte che lo Spectrum mi è capitato per le mani, è stato per spostarlo da cantine a soffitte.

"Prima o poi mi piacerebbe accenderlo e farci qualcosa", era il pensiero di quando lo vedevo. Poi finalmente un po' di tempo fa è arrivato il momento giusto mentre facevo pulizia tra vecchie scartoffie, ritrovai i vecchi listati di programmi tra cui quelli delle routine in assembler del Master Mind. Speravo di ripartire dal programma che avevo lasciato, ma accendendo il mio Spectrum mi accorsi subito che aveva problemi e quindi dovetti rinunciare.

Decisi allora di trovare un emulatore dello Speccy su PC, per riscrivere il programma partendo dalla base dei listati ritrovati.

Ecco i listati delle routine. C'è anche la struttura dati dei codici e come organizzavo la memoria per contenere il codice sorgente, il compilato e i due Assembleri:

The image shows two pages of handwritten assembly code and data structure diagrams for the ZX Spectrum Master Mind program.

Left Page:

- ① CP Inizializzazione del campo "PUNTAZIONE" ⑤**
ORG 2
START EQU 4000 CP EQU ...
ADDCP EQU START+10
LD BC, 1295
LD DE, 10 ; DE = incremento
LD IX, CP-12 ; IX = al campo puntatore
LD HL, CP-10 ; HL = Valore base da mettere nel campo puntatore
Loop ADD IX, DE
ADD HL, DE
LD (IX), HL
LD (IX+1), HL
DEC BC
LD A, B
OR C
JR NZ, Loop
LD (IX), FF
RET
- ① CP Programma Principale**
ORG 1
CALL ORG 2 ; Inizializzazione campo "PUNTAZIONE"
CALL ORG 3 ; Inizializzazione campo "POSIZIONE COLORI"
CALL ORG 4 ; Inizializzazione campo "N° COLORI LOW"
CALL ORG 5 ; Inizializzazione campo "N° COLORI HIGH"
RET
- Data Structure Diagram:**
A diagram showing memory layout with labels: N° COLORI, POSIZIONE COLORI, PUNTAZIONE, and PROBABILITY. It includes pointers like CP1, CP2, CP3, CP4, CP5, CP6, and CP7, and addresses like ORG 4, ORG 5, ORG 3, and ORG 2.

Right Page:

- Inizializzazione campo "POSIZIONE COLORI" ③**
START EQU 4000 ORG 3 40050-9C72-189
ADDCP EQU START+10 CP EQU 40000 = A2FB
LD HL, CP
LD D, 20
LD E, 20
LD B, 20
LD C, 40
DEBC NEXT SRL C
LD C, 20
SRL B
LD B, 20
SRL E
LD E, 20
SRL D
RET
- Bit Manipulation and Loop:**
A series of instructions for bit manipulation and looping, including: JR 2, N1, SET 7, (HL), N1 BIT 5, E, JR 2, N2, SET 6, (HL), N2 BIT 5, B, JR 2, N3, SET 5, (HL), N3 BIT 5, C, JR 2, N4, SET 4, (HL), N4 BIT 4, D, JR 2, N5, SET 3, (HL), N5 BIT 4, E, JR 2, N6, SET 2, (HL), N6 BIT 4, B, JR 2, N7, SET 1, (HL), N7 BIT 4, C, JR 2, N8, SET 0, (HL), N8 INC HL, BIT 1, D, JR 2, N9, SET 7, (HL), N9 BIT 1, E, JR 2, N10, SET 6, (HL), N10 BIT 1, B, JR 2, N11, SET 5, (HL), N11 BIT 1, C, JR 2, N12, SET 4, (HL), N12 BIT 1, D, JR 2, N13, SET 3, (HL), N13 BIT 1, E, JR 2, N14, SET 2, (HL), N14 BIT 1, B, JR 2, N15, SET 1, (HL), N15 BIT 1, C, JR 2, N16, SET 0, (HL), N16 INC HL, BIT 1, D, JR 2, N17, SET 7, (HL), N17 BIT 1, E, JR 2, N18, SET 6, (HL), N18 BIT 1, B, JR 2, N19, SET 5, (HL), N19 BIT 1, C, JR 2, N20, SET 4, (HL), N20 BIT 1, D, JR 2, N21, SET 3, (HL), N21 BIT 1, E, JR 2, N22, SET 2, (HL), N22 BIT 1, B, JR 2, N23, SET 1, (HL), N23 BIT 1, C, JR 2, N24, SET 0, (HL), N24 INC HL, BIT 1, D, JR 2, N25, SET 7, (HL), N25 BIT 1, E, JR 2, N26, SET 6, (HL), N26 BIT 1, B, JR 2, N27, SET 5, (HL), N27 BIT 1, C, JR 2, N28, SET 4, (HL), N28 BIT 1, D, JR 2, N29, SET 3, (HL), N29 BIT 1, E, JR 2, N30, SET 2, (HL), N30 BIT 1, B, JR 2, N31, SET 1, (HL), N31 BIT 1, C, JR 2, N32, SET 0, (HL), N32 INC HL, BIT 1, D, JR 2, N33, SET 7, (HL), N33 BIT 1, E, JR 2, N34, SET 6, (HL), N34 BIT 1, B, JR 2, N35, SET 5, (HL), N35 BIT 1, C, JR 2, N36, SET 4, (HL), N36 BIT 1, D, JR 2, N37, SET 3, (HL), N37 BIT 1, E, JR 2, N38, SET 2, (HL), N38 BIT 1, B, JR 2, N39, SET 1, (HL), N39 BIT 1, C, JR 2, N40, SET 0, (HL), N40 INC HL, BIT 1, D, JR 2, N41, SET 7, (HL), N41 BIT 1, E, JR 2, N42, SET 6, (HL), N42 BIT 1, B, JR 2, N43, SET 5, (HL), N43 BIT 1, C, JR 2, N44, SET 4, (HL), N44 BIT 1, D, JR 2, N45, SET 3, (HL), N45 BIT 1, E, JR 2, N46, SET 2, (HL), N46 BIT 1, B, JR 2, N47, SET 1, (HL), N47 BIT 1, C, JR 2, N48, SET 0, (HL), N48 INC HL, BIT 1, D, JR 2, N49, SET 7, (HL), N49 BIT 1, E, JR 2, N50, SET 6, (HL), N50 BIT 1, B, JR 2, N51, SET 5, (HL), N51 BIT 1, C, JR 2, N52, SET 4, (HL), N52 BIT 1, D, JR 2, N53, SET 3, (HL), N53 BIT 1, E, JR 2, N54, SET 2, (HL), N54 BIT 1, B, JR 2, N55, SET 1, (HL), N55 BIT 1, C, JR 2, N56, SET 0, (HL), N56 INC HL, BIT 1, D, JR 2, N57, SET 7, (HL), N57 BIT 1, E, JR 2, N58, SET 6, (HL), N58 BIT 1, B, JR 2, N59, SET 5, (HL), N59 BIT 1, C, JR 2, N60, SET 4, (HL), N60 BIT 1, D, JR 2, N61, SET 3, (HL), N61 BIT 1, E, JR 2, N62, SET 2, (HL), N62 BIT 1, B, JR 2, N63, SET 1, (HL), N63 BIT 1, C, JR 2, N64, SET 0, (HL), N64 INC HL, BIT 1, D, JR 2, N65, SET 7, (HL), N65 BIT 1, E, JR 2, N66, SET 6, (HL), N66 BIT 1, B, JR 2, N67, SET 5, (HL), N67 BIT 1, C, JR 2, N68, SET 4, (HL), N68 BIT 1, D, JR 2, N69, SET 3, (HL), N69 BIT 1, E, JR 2, N70, SET 2, (HL), N70 BIT 1, B, JR 2, N71, SET 1, (HL), N71 BIT 1, C, JR 2, N72, SET 0, (HL), N72 INC HL, BIT 1, D, JR 2, N73, SET 7, (HL), N73 BIT 1, E, JR 2, N74, SET 6, (HL), N74 BIT 1, B, JR 2, N75, SET 5, (HL), N75 BIT 1, C, JR 2, N76, SET 4, (HL), N76 BIT 1, D, JR 2, N77, SET 3, (HL), N77 BIT 1, E, JR 2, N78, SET 2, (HL), N78 BIT 1, B, JR 2, N79, SET 1, (HL), N79 BIT 1, C, JR 2, N80, SET 0, (HL), N80 INC HL, BIT 1, D, JR 2, N81, SET 7, (HL), N81 BIT 1, E, JR 2, N82, SET 6, (HL), N82 BIT 1, B, JR 2, N83, SET 5, (HL), N83 BIT 1, C, JR 2, N84, SET 4, (HL), N84 BIT 1, D, JR 2, N85, SET 3, (HL), N85 BIT 1, E, JR 2, N86, SET 2, (HL), N86 BIT 1, B, JR 2, N87, SET 1, (HL), N87 BIT 1, C, JR 2, N88, SET 0, (HL), N88 INC HL, BIT 1, D, JR 2, N89, SET 7, (HL), N89 BIT 1, E, JR 2, N90, SET 6, (HL), N90 BIT 1, B, JR 2, N91, SET 5, (HL), N91 BIT 1, C, JR 2, N92, SET 4, (HL), N92 BIT 1, D, JR 2, N93, SET 3, (HL), N93 BIT 1, E, JR 2, N94, SET 2, (HL), N94 BIT 1, B, JR 2, N95, SET 1, (HL), N95 BIT 1, C, JR 2, N96, SET 0, (HL), N96 INC HL, BIT 1, D, JR 2, N97, SET 7, (HL), N97 BIT 1, E, JR 2, N98, SET 6, (HL), N98 BIT 1, B, JR 2, N99, SET 5, (HL), N99 BIT 1, C, JR 2, N100, SET 4, (HL), N100 BIT 1, D, JR 2, N101, SET 3, (HL), N101 BIT 1, E, JR 2, N102, SET 2, (HL), N102 BIT 1, B, JR 2, N103, SET 1, (HL), N103 BIT 1, C, JR 2, N104, SET 0, (HL), N104 INC HL, BIT 1, D, JR 2, N105, SET 7, (HL), N105 BIT 1, E, JR 2, N106, SET 6, (HL), N106 BIT 1, B, JR 2, N107, SET 5, (HL), N107 BIT 1, C, JR 2, N108, SET 4, (HL), N108 BIT 1, D, JR 2, N109, SET 3, (HL), N109 BIT 1, E, JR 2, N110, SET 2, (HL), N110 BIT 1, B, JR 2, N111, SET 1, (HL), N111 BIT 1, C, JR 2, N112, SET 0, (HL), N112 INC HL, BIT 1, D, JR 2, N113, SET 7, (HL), N113 BIT 1, E, JR 2, N114, SET 6, (HL), N114 BIT 1, B, JR 2, N115, SET 5, (HL), N115 BIT 1, C, JR 2, N116, SET 4, (HL), N116 BIT 1, D, JR 2, N117, SET 3, (HL), N117 BIT 1, E, JR 2, N118, SET 2, (HL), N118 BIT 1, B, JR 2, N119, SET 1, (HL), N119 BIT 1, C, JR 2, N120, SET 0, (HL), N120 INC HL, BIT 1, D, JR 2, N121, SET 7, (HL), N121 BIT 1, E, JR 2, N122, SET 6, (HL), N122 BIT 1, B, JR 2, N123, SET 5, (HL), N123 BIT 1, C, JR 2, N124, SET 4, (HL), N124 BIT 1, D, JR 2, N125, SET 3, (HL), N125 BIT 1, E, JR 2, N126, SET 2, (HL), N126 BIT 1, B, JR 2, N127, SET 1, (HL), N127 BIT 1, C, JR 2, N128, SET 0, (HL), N128 INC HL, BIT 1, D, JR 2, N129, SET 7, (HL), N129 BIT 1, E, JR 2, N130, SET 6, (HL), N130 BIT 1, B, JR 2, N131, SET 5, (HL), N131 BIT 1, C, JR 2, N132, SET 4, (HL), N132 BIT 1, D, JR 2, N133, SET 3, (HL), N133 BIT 1, E, JR 2, N134, SET 2, (HL), N134 BIT 1, B, JR 2, N135, SET 1, (HL), N135 BIT 1, C, JR 2, N136, SET 0, (HL), N136 INC HL, BIT 1, D, JR 2, N137, SET 7, (HL), N137 BIT 1, E, JR 2, N138, SET 6, (HL), N138 BIT 1, B, JR 2, N139, SET 5, (HL), N139 BIT 1, C, JR 2, N140, SET 4, (HL), N140 BIT 1, D, JR 2, N141, SET 3, (HL), N141 BIT 1, E, JR 2, N142, SET 2, (HL), N142 BIT 1, B, JR 2, N143, SET 1, (HL), N143 BIT 1, C, JR 2, N144, SET 0, (HL), N144 INC HL, BIT 1, D, JR 2, N145, SET 7, (HL), N145 BIT 1, E, JR 2, N146, SET 6, (HL), N146 BIT 1, B, JR 2, N147, SET 5, (HL), N147 BIT 1, C, JR 2, N148, SET 4, (HL), N148 BIT 1, D, JR 2, N149, SET 3, (HL), N149 BIT 1, E, JR 2, N150, SET 2, (HL), N150 BIT 1, B, JR 2, N151, SET 1, (HL), N151 BIT 1, C, JR 2, N152, SET 0, (HL), N152 INC HL, BIT 1, D, JR 2, N153, SET 7, (HL), N153 BIT 1, E, JR 2, N154, SET 6, (HL), N154 BIT 1, B, JR 2, N155, SET 5, (HL), N155 BIT 1, C, JR 2, N156, SET 4, (HL), N156 BIT 1, D, JR 2, N157, SET 3, (HL), N157 BIT 1, E, JR 2, N158, SET 2, (HL), N158 BIT 1, B, JR 2, N159, SET 1, (HL), N159 BIT 1, C, JR 2, N160, SET 0, (HL), N160 INC HL, BIT 1, D, JR 2, N161, SET 7, (HL), N161 BIT 1, E, JR 2, N162, SET 6, (HL), N162 BIT 1, B, JR 2, N163, SET 5, (HL), N163 BIT 1, C, JR 2, N164, SET 4, (HL), N164 BIT 1, D, JR 2, N165, SET 3, (HL), N165 BIT 1, E, JR 2, N166, SET 2, (HL), N166 BIT 1, B, JR 2, N167, SET 1, (HL), N167 BIT 1, C, JR 2, N168, SET 0, (HL), N168 INC HL, BIT 1, D, JR 2, N169, SET 7, (HL), N169 BIT 1, E, JR 2, N170, SET 6, (HL), N170 BIT 1, B, JR 2, N171, SET 5, (HL), N171 BIT 1, C, JR 2, N172, SET 4, (HL), N172 BIT 1, D, JR 2, N173, SET 3, (HL), N173 BIT 1, E, JR 2, N174, SET 2, (HL), N174 BIT 1, B, JR 2, N175, SET 1, (HL), N175 BIT 1, C, JR 2, N176, SET 0, (HL), N176 INC HL, BIT 1, D, JR 2, N177, SET 7, (HL), N177 BIT 1, E, JR 2, N178, SET 6, (HL), N178 BIT 1, B, JR 2, N179, SET 5, (HL), N179 BIT 1, C, JR 2, N180, SET 4, (HL), N180 BIT 1, D, JR 2, N181, SET 3, (HL), N181 BIT 1, E, JR 2, N182, SET 2, (HL), N182 BIT 1, B, JR 2, N183, SET 1, (HL), N183 BIT 1, C, JR 2, N184, SET 0, (HL), N184 INC HL, BIT 1, D, JR 2, N185, SET 7, (HL), N185 BIT 1, E, JR 2, N186, SET 6, (HL), N186 BIT 1, B, JR 2, N187, SET 5, (HL), N187 BIT 1, C, JR 2, N188, SET 4, (HL), N188 BIT 1, D, JR 2, N189, SET 3, (HL), N189 BIT 1, E, JR 2, N190, SET 2, (HL), N190 BIT 1, B, JR 2, N191, SET 1, (HL), N191 BIT 1, C, JR 2, N192, SET 0, (HL), N192 INC HL, BIT 1, D, JR 2, N193, SET 7, (HL), N193 BIT 1, E, JR 2, N194, SET 6, (HL), N194 BIT 1, B, JR 2, N195, SET 5, (HL), N195 BIT 1, C, JR 2, N196, SET 4, (HL), N196 BIT 1, D, JR 2, N197, SET 3, (HL), N197 BIT 1, E, JR 2, N198, SET 2, (HL), N198 BIT 1, B, JR 2, N199, SET 1, (HL), N199 BIT 1, C, JR 2, N200, SET 0, (HL), N200 INC HL, BIT 1, D, JR 2, N201, SET 7, (HL), N201 BIT 1, E, JR 2, N202, SET 6, (HL), N202 BIT 1, B, JR 2, N203, SET 5, (HL), N203 BIT 1, C, JR 2, N204, SET 4, (HL), N204 BIT 1, D, JR 2, N205, SET 3, (HL), N205 BIT 1, E, JR 2, N206, SET 2, (HL), N206 BIT 1, B, JR 2, N207, SET 1, (HL), N207 BIT 1, C, JR 2, N208, SET 0, (HL), N208 INC HL, BIT 1, D, JR 2, N209, SET 7, (HL), N209 BIT 1, E, JR 2, N210, SET 6, (HL), N210 BIT 1, B, JR 2, N211, SET 5, (HL), N211 BIT 1, C, JR 2, N212, SET 4, (HL), N212 BIT 1, D, JR 2, N213, SET 3, (HL), N213 BIT 1, E, JR 2, N214, SET 2, (HL), N214 BIT 1, B, JR 2, N215, SET 1, (HL), N215 BIT 1, C, JR 2, N216, SET 0, (HL), N216 INC HL, BIT 1, D, JR 2, N217, SET 7, (HL), N217 BIT 1, E, JR 2, N218, SET 6, (HL), N218 BIT 1, B, JR 2, N219, SET 5, (HL), N219 BIT 1, C, JR 2, N220, SET 4, (HL), N220 BIT 1, D, JR 2, N221, SET 3, (HL), N221 BIT 1, E, JR 2, N222, SET 2, (HL), N222 BIT 1, B, JR 2, N223, SET 1, (HL), N223 BIT 1, C, JR 2, N224, SET 0, (HL), N224 INC HL, BIT 1, D, JR 2, N225, SET 7, (HL), N225 BIT 1, E, JR 2, N226, SET 6, (HL), N226 BIT 1, B, JR 2, N227, SET 5, (HL), N227 BIT 1, C, JR 2, N228, SET 4, (HL), N228 BIT 1, D, JR 2, N229, SET 3, (HL), N229 BIT 1, E, JR 2, N230, SET 2, (HL), N230 BIT 1, B, JR 2, N231, SET 1, (HL), N231 BIT 1, C, JR 2, N232, SET 0, (HL), N232 INC HL, BIT 1, D, JR 2, N233, SET 7, (HL), N233 BIT 1, E, JR 2, N234, SET 6, (HL), N234 BIT 1, B, JR 2, N235, SET 5, (HL), N235 BIT 1, C, JR 2, N236, SET 4, (HL), N236 BIT 1, D, JR 2, N237, SET 3, (HL), N237 BIT 1, E, JR 2, N238, SET 2, (HL), N238 BIT 1, B, JR 2, N239, SET 1, (HL), N239 BIT 1, C, JR 2, N240, SET 0, (HL), N240 INC HL, BIT 1, D, JR 2, N241, SET 7, (HL), N241 BIT 1, E, JR 2, N242, SET 6, (HL), N242 BIT 1, B, JR 2, N243, SET 5, (HL), N243 BIT 1, C, JR 2, N244, SET 4, (HL), N244 BIT 1, D, JR 2, N245, SET 3, (HL), N245 BIT 1, E, JR 2, N246, SET 2, (HL), N246 BIT 1, B, JR 2, N247, SET 1, (HL), N247 BIT 1, C, JR 2, N248, SET 0, (HL), N248 INC HL, BIT 1, D, JR 2, N249, SET 7, (HL), N249 BIT 1, E, JR 2, N250, SET 6, (HL), N250 BIT 1, B, JR 2, N251, SET 5, (HL), N251 BIT 1, C, JR 2, N252, SET 4, (HL), N252 BIT 1, D, JR 2, N253, SET 3, (HL), N253 BIT 1, E, JR 2, N254, SET 2, (HL), N254 BIT 1, B, JR 2, N255, SET 1, (HL), N255 BIT 1, C, JR 2, N256, SET 0, (HL), N256 INC HL, BIT 1, D, JR 2, N257, SET 7, (HL), N257 BIT 1, E, JR 2, N258, SET 6, (HL), N258 BIT 1, B, JR 2, N259, SET 5, (HL), N259 BIT 1, C, JR 2, N260, SET 4, (HL), N260 BIT 1, D, JR 2, N261, SET 3, (HL), N261 BIT 1, E, JR 2, N262, SET 2, (HL), N262 BIT 1, B, JR 2, N263, SET 1, (HL), N263 BIT 1, C, JR 2, N264, SET 0, (HL), N264 INC HL, BIT 1, D, JR 2, N265, SET 7, (HL), N265 BIT 1, E, JR 2, N266, SET 6, (HL), N266 BIT 1, B, JR 2, N267, SET 5, (HL), N267 BIT 1, C, JR 2, N268, SET 4, (HL), N268 BIT 1, D, JR 2, N269, SET 3, (HL), N269 BIT 1, E, JR 2, N270, SET 2, (HL), N270 BIT 1, B, JR 2, N271, SET 1, (HL), N271 BIT 1, C, JR 2, N272, SET 0, (HL), N272 INC HL, BIT 1, D, JR 2, N273, SET 7, (HL), N273 BIT 1, E, JR 2, N274, SET 6, (HL), N274 BIT 1, B, JR 2, N275, SET 5, (HL), N275 BIT 1, C, JR 2, N276, SET 4, (HL), N276 BIT 1, D, JR 2, N277, SET 3, (HL), N277 BIT 1, E, JR 2, N278, SET 2, (HL), N278 BIT 1, B, JR 2, N279, SET 1, (HL), N279 BIT 1, C, JR 2, N280, SET 0, (HL), N280 INC HL, BIT 1, D, JR 2, N281, SET 7, (HL), N281 BIT 1, E, JR 2, N282, SET 6, (HL), N282 BIT 1, B, JR 2, N283, SET 5, (HL), N283 BIT 1, C, JR 2, N284, SET 4, (HL), N284 BIT 1, D, JR 2, N285, SET 3, (HL), N285 BIT 1, E, JR 2, N286, SET 2, (HL), N286 BIT 1, B, JR 2, N287, SET 1, (HL), N287 BIT 1, C, JR 2, N288, SET 0, (HL), N288 INC HL, BIT 1, D, JR 2, N289, SET 7, (HL), N289 BIT 1, E, JR 2, N290, SET 6, (HL), N290 BIT 1, B, JR 2, N291, SET 5, (HL), N291 BIT 1, C, JR 2, N292, SET 4, (HL), N292 BIT 1, D, JR 2, N293, SET 3, (HL), N293 BIT 1, E, JR 2, N294, SET 2, (HL), N294 BIT 1, B, JR 2, N295, SET 1, (HL), N295 BIT 1, C, JR 2, N296, SET 0, (HL), N296 INC HL, BIT 1, D, JR 2, N297, SET 7, (HL), N297 BIT 1, E, JR 2, N298, SET 6, (HL), N298 BIT 1, B, JR 2, N299, SET 5, (HL), N299 BIT 1, C, JR 2, N300, SET 4, (HL), N300 BIT 1, D, JR 2, N301, SET 3, (HL), N301 BIT 1, E, JR 2, N302, SET 2, (HL), N302 BIT 1, B, JR 2, N303, SET 1, (HL), N303 BIT 1, C, JR 2, N304, SET 0, (HL), N304 INC HL, BIT 1, D, JR 2, N305, SET 7, (HL), N305 BIT 1, E, JR 2, N306, SET 6, (HL), N306 BIT 1, B, JR 2, N307, SET 5, (HL), N307 BIT 1, C, JR 2, N308, SET 4, (HL), N308 BIT 1, D, JR 2, N309, SET 3, (HL), N309 BIT 1, E, JR 2, N310, SET 2, (HL), N310 BIT 1, B, JR 2, N311, SET 1, (HL), N311 BIT 1, C, JR 2, N312, SET 0, (HL), N312 INC HL, BIT 1, D, JR 2, N313, SET 7, (HL), N313 BIT 1, E, JR 2, N314, SET 6, (HL), N314 BIT 1, B, JR 2, N315, SET 5, (HL), N315 BIT 1, C, JR 2, N316, SET 4, (HL), N316 BIT 1, D, JR 2, N317, SET 3, (HL), N317 BIT 1, E, JR 2, N318, SET 2, (HL), N318 BIT 1, B, JR 2, N319, SET 1, (HL), N319 BIT 1, C, JR 2, N320, SET 0, (HL), N320 INC HL, BIT 1, D, JR 2, N321, SET 7, (HL), N321 BIT 1, E, JR 2, N322, SET 6, (HL), N322 BIT 1, B, JR 2, N323, SET 5, (HL), N323 BIT 1, C, JR 2, N324, SET 4, (HL), N324 BIT 1, D, JR 2, N325, SET 3, (HL), N325 BIT 1, E, JR 2, N326, SET 2, (HL), N326 BIT 1, B, JR 2, N327, SET 1, (HL), N327 BIT 1, C, JR 2, N328, SET 0, (HL), N328 INC HL, BIT 1, D, JR 2, N329, SET 7, (HL), N329 BIT 1, E, JR 2, N330, SET 6, (HL), N330 BIT 1, B, JR 2, N331, SET 5, (HL), N331 BIT 1, C, JR 2, N332, SET 4, (HL), N332 BIT 1, D, JR 2, N333, SET 3, (HL), N333 BIT 1, E, JR 2, N334, SET 2, (HL), N334 BIT 1, B, JR 2, N335, SET 1, (HL), N335 BIT 1, C, JR 2, N336, SET 0, (HL), N336 INC HL, BIT 1, D, JR 2, N337, SET 7, (HL), N337 BIT 1, E, JR 2, N338, SET 6, (HL), N338 BIT 1, B, JR 2, N339, SET 5, (HL), N339 BIT 1, C, JR 2, N340, SET 4, (HL), N340 BIT 1, D, JR 2, N341, SET 3, (HL), N341 BIT 1, E, JR 2, N342, SET 2, (HL), N342 BIT 1, B, JR 2, N343, SET 1, (HL), N343 BIT 1, C, JR 2, N344, SET 0, (HL), N344 INC HL, BIT 1, D, JR 2, N345, SET 7, (HL), N345 BIT 1, E, JR 2, N346, SET 6, (HL), N346 BIT 1, B, JR 2, N347, SET 5, (HL), N347 BIT 1, C, JR 2, N348, SET 4, (HL), N348 BIT 1, D, JR 2, N349, SET 3, (HL), N349 BIT 1, E, JR 2, N350, SET 2, (HL), N350 BIT 1, B, JR 2, N351, SET 1, (HL), N351 BIT 1, C, JR 2, N352, SET 0, (HL), N352 INC HL, BIT 1, D, JR 2, N353, SET 7, (HL), N353 BIT 1, E, JR 2, N354, SET 6, (HL), N354 BIT 1, B, JR 2, N355, SET 5, (HL), N355 BIT 1, C, JR 2, N356, SET 4, (HL), N356 BIT 1, D, JR 2, N357, SET 3, (HL), N357 BIT 1, E, JR 2, N358, SET 2, (HL), N358 BIT 1, B, JR 2, N359, SET 1, (HL), N359 BIT 1, C, JR 2, N360, SET 0, (HL), N360 INC HL, BIT 1, D, JR 2, N361, SET 7, (HL), N361 BIT 1, E, JR 2, N362, SET 6, (HL), N362 BIT 1, B, JR 2, N363, SET 5, (HL), N363 BIT 1, C, JR 2, N364, SET 4, (HL), N364 BIT 1, D, JR 2, N365, SET 3, (HL), N365 BIT 1, E, JR 2, N366, SET 2, (HL), N366 BIT 1, B, JR 2, N367, SET 1, (HL), N367 BIT 1, C, JR 2, N368, SET 0, (HL), N368 INC HL, BIT 1, D, JR 2, N369, SET 7, (HL), N369 BIT 1, E, JR 2, N370, SET 6, (HL), N370 BIT 1, B, JR 2, N371, SET 5, (HL), N371 BIT 1, C, JR 2, N372, SET 4, (HL), N372 BIT 1, D, JR 2, N373, SET 3, (HL), N373 BIT 1, E, JR 2, N374, SET 2, (HL), N374 BIT 1, B, JR 2, N375, SET 1, (HL), N375 BIT 1, C, JR 2, N376, SET 0, (HL), N376 INC HL, BIT 1, D, JR 2, N377, SET 7, (HL), N377 BIT 1, E, JR 2, N378, SET 6, (HL), N378 BIT 1, B, JR 2, N379, SET 5, (HL), N379 BIT 1, C, JR 2, N380, SET 4, (HL), N380 BIT 1, D, JR 2, N381, SET 3, (HL), N381 BIT 1, E, JR 2, N382, SET 2, (HL), N382 BIT 1, B, JR 2, N383, SET 1, (HL), N383 BIT 1, C, JR 2, N384, SET 0, (HL), N384 INC HL, BIT 1, D, JR 2, N385, SET 7, (HL), N385 BIT 1, E, JR 2, N386, SET 6, (HL), N386 BIT 1, B, JR 2, N387, SET 5, (HL), N387 BIT 1, C, JR 2, N388, SET 4, (HL), N388 BIT 1, D, JR 2, N389, SET 3, (HL), N389 BIT 1, E, JR 2, N390, SET 2, (HL), N390 BIT 1, B, JR 2, N391, SET 1, (HL), N391 BIT 1, C, JR 2, N392, SET 0, (HL), N392 INC HL, BIT 1, D, JR 2, N393, SET 7, (HL), N393 BIT 1, E, JR 2, N394, SET 6, (HL), N394 BIT 1, B, JR 2, N395, SET 5, (HL), N395 BIT 1, C, JR 2, N396, SET 4, (HL), N396 BIT 1, D, JR 2, N397, SET 3, (HL), N397 BIT 1, E, JR 2, N398, SET 2, (HL), N398 BIT 1, B, JR 2, N399, SET 1, (HL), N399 BIT 1, C, JR 2, N400, SET 0, (HL), N400 INC HL, BIT 1, D, JR 2, N401, SET 7, (HL), N401 BIT 1, E, JR 2, N402, SET 6, (HL), N402 BIT 1, B, JR 2, N403, SET 5, (HL), N403 BIT 1, C, JR 2, N404, SET 4, (HL), N404 BIT 1, D, JR 2, N405, SET 3, (HL), N405 BIT 1, E, JR 2, N406, SET 2, (HL), N406 BIT 1, B, JR 2, N407, SET 1, (HL), N407 BIT 1, C, JR 2, N408, SET 0, (HL), N408 INC HL, BIT 1, D, JR 2, N409, SET 7, (HL), N409 BIT 1, E, JR 2, N410, SET 6, (HL), N410 BIT 1, B, JR 2, N411, SET 5, (HL), N411 BIT 1, C, JR 2, N412, SET 4, (HL), N412 BIT 1, D, JR 2, N413, SET 3, (HL), N413 BIT 1, E, JR 2, N414, SET 2, (HL), N414 BIT 1, B, JR 2, N415, SET 1, (HL), N415 BIT 1, C, JR 2, N416, SET 0, (HL), N416 INC HL, BIT 1, D,

TEST2 Numero di X+0

ENTRATA	USCITE	MODIFICA:
IX+CP IX+CP C=0F	C=0F A=0+X	AF, BC, DE

ORG 7

LD E, 0 ; Nibble

LD D, 0 ; Nibble

LD E, 0F ; Nibble non interinato

LD A, D ; I Nibble

AND (IX+3)

LD B, A ; LD A, D

AND (IX+3)

CP B

JR C, E1

LD A, B

LD A, B ; SRAA

LD C, A ; SRAA SRAA

LD A, E ; II Nibble

AND (IX+3)

LD B, A ; LD A, E

AND (IX+3)

CP B

JR C, E2 ; SRAA SRAA

LD A, B

LD A, B

LD A, D ; III Nibble

AND (IX+4)

LD B, A

LD A, D

AND (IX+4)

CP B

JR C, E3

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, D ; IV Nibble

AND (IX+4)

LD B, A

LD A, D

AND (IX+4)

CP B

JR C, E4

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; V Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E5

LD A, B

LD A, B

LD A, D ; VI Nibble

AND (IX+5)

LD B, A

LD A, D

AND (IX+5)

CP B

JR C, E6

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; VII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E7

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; VIII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E8

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; IX Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E9

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; X Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E10

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XI Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E11

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E12

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XIII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E13

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XIV Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E14

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XV Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E15

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XVI Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E16

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XVII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E17

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XVIII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E18

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XIX Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E19

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XX Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E20

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XXI Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E21

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XXII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E22

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XXIII Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E23

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XXIV Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E24

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XXV Nibble

AND (IX+5)

LD B, A

LD A, E

AND (IX+5)

CP B

JR C, E25

LD A, B ; SRAA SRAA

LD A, B ; SRAA SRAA

LD C, A

LD C, A

LD A, E ; XXVI Nibble

AND (IX+5)

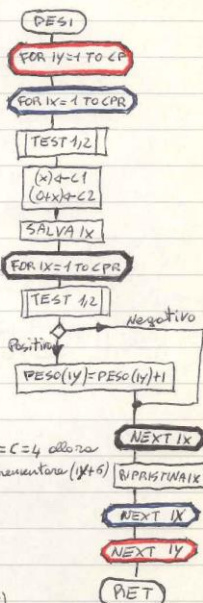
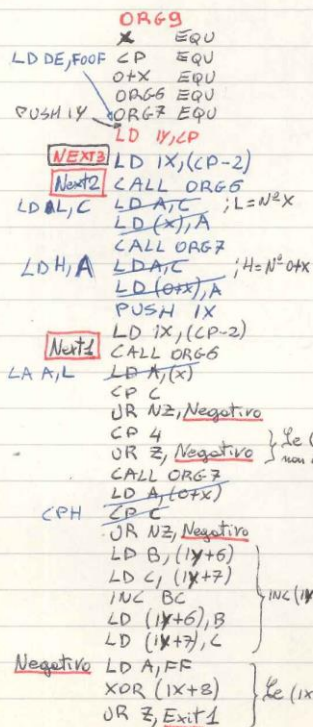
LD B, A

LD A, E

SIMULA

~~NOTIZIA~~ Edicola il peso di ogni CP

ENTRATE	USCITE	MODIFICA
/	/	tutti i registri e i campi "PESO"



CONTINUA

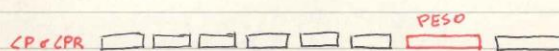
SIMULA

~~XXXXXX~~ (continuazione)

```
LD B, (1x+8)
LD C, (1x+8)
PUSH BC
POP ix
JR Next1 } 1x = (1x+8)

POP ix
LD A, FF
XOR (1x+8) } 1x = (1x+8) = FF End
JR Z, Exit2
LD B, (1x+8)
LD C, (1x+8)
PUSH BC
POP ix
JR Next2 } 1x = (1x+8)

LD A, FF
XOR (1x+8) } 1x = (1x+8) = FF Finisci
JR Z, FINE
LD BC, 10
ADD 1x, BC
JR Next3 } ELSE 1x = 1x + 10 e ripet.
```



MASTER MIND

STRUTTURA DEL FILE

START = 44000
ABEO

START	I NIBBLE	000		
START+1	II NIBBLE	101	POSIZIONE COLORI	} C
START+2	III NIBBLE	202		
START+3	IV NIBBLE	303		
START+4	V NIBBLE	404	NUMERO COLORI	
START+5	VI NIBBLE	505		
START+6	BYTE	606	X	
START+7	BYTE	707	O+X	
START+8		808		} PUNTATORE INIZIO FILE
START+9	ADDRESS	909	PUNTATORE	
START+10	05 NIBBLE	100A		} POSIZIONE COLORI
START+11	11 NIBBLE	110B		
START+12	25 NIBBLE	120C		
START+13	35 NIBBLE	130D		} NUMERO COLORI
START+14	45 NIBBLE	140E		
START+15	55 NIBBLE	150F		
START+16	6	160F		} CP
START+17	WORD	170F	PESSO	
START+18		180F		
START+19	ADDRESS	190F	PUNTATORE	} RECORD
START+20		200F		

VARIABILI

PUNTATORE

RECORD

PUNTATORE 1

K	EQU 44000
ADDL	EQU K
X	EQU K+6
D+X	EQU K+7
PUNT	EQU K+8
ADDCP	EQU K+10

MASTER MIND

SORBO 37000, 438
OGG O. 45000, 540
MZ

 $\frac{1}{2}$

SORGENTE

OGGETTO

FILE

37000-38058 CP REG. ALU Bytes = 114	DAG-2 ORG 3 ORG 4 ORG 5	43000-A2F8 ORG 1 Bytes = 13 43012-A804 43020-A806 ORG 2 Bytes = 32 43051-A82B 43070-A83E ORG 3 Bytes = 183 43258-A8FA 43270-A806 ORG 4 Bytes = 45 43314-A932 43320-A938 ORG 5 Bytes = 55 43374-A96E 43376-A970 ORG 6 Bytes = 35 43401-A994 43420-A92C ORG X Bytes = 113 43532-AA0C 43540-AA3C ORG 8 Bytes = 74 43653-AA85 43680-AAAD ORG 9 Bytes = 112 43721-AB0F 43850-AB54 ORG 10 Bytes = 53	74000-START AB 50 54076 AB 8C START+1 X 44072 AB 7C START+2 OX 44078 AB 7C START+8 PUNT 4408 AB 10 START+10 CP1 4408 AB 7C START+18 PUNT 44020 AB 4C CP 2 44028 AB 4C PUNT 2 44030 CP 3 44038 PUNT 3
38680 N° COLORI HIGH Bytes = 301	ADDCP		56050 DE 7E CP 1295 56038 DE 7E PUNT 1295 56050 DE 7E CP 1296 56067 DE 7E FFXX 56078
39000 TEST1 N X Bytes = 159	IN: M1P I1P C OUT C MAX W M1P I1P C D180 L10P OUT DE 0X ADDCP C1X 0X ORG 7 IN: (C) (X) (0X)	→	MEMORY MAPPING 23000 CHAMP 36582 39000 MASTER MIND SOURCE 42800 43000 MASTER MIND 43880 44000 MASTER MIND FILE 56908 57344-5000 ZEUS 65279
39800 CPR Bytes = 436			
40300 SIMULA Bytes = 576	ADDCP X 0X ORG 6 ORG 7		
40950 TROVA C Bytes = 272	ADDCP C		
41250 CLS PESI Bytes = 138		43910-AB86 ORG 11 Bytes = 23	

La ruggine era molta e ci ho messo un bel po' per riprendere confidenza con l'Assembler e per capire la logica delle routine che avevo scritto. Ripercorrere i ragionamenti fatti 35 anni prima è stato emozionante come rivedere delle foto di quei tempi.

Poi ho cercato di ricrearmi l'ambiente che avevo sullo Spectrum ma questa volta con l'emulatore FUSE e Zeus Assembler + CHAMP in versione TZX; Zeus per programmare e CHAMP per il debugging.

Dopo un po' mi sono reso conto che questo modo di procedere era troppo laborioso e lento e così ho cercato, e per fortuna trovato, un IDE (Integrated Development Environment) da utilizzare su PC Windows chiamato "BASIN".

Il resto è storia recente di tante notti passate a scrivere e soprattutto riscrivere, migliorare, velocizzare il codice e farcelo stare tutto nei 48K.

Grazie veramente agli autori di BasIn. Senza questo tool credo non mi sarebbe stato possibile terminare il progetto. Mi sento di consigliarlo vivamente a chi vuole sviluppare sia in Basic che Assembler.

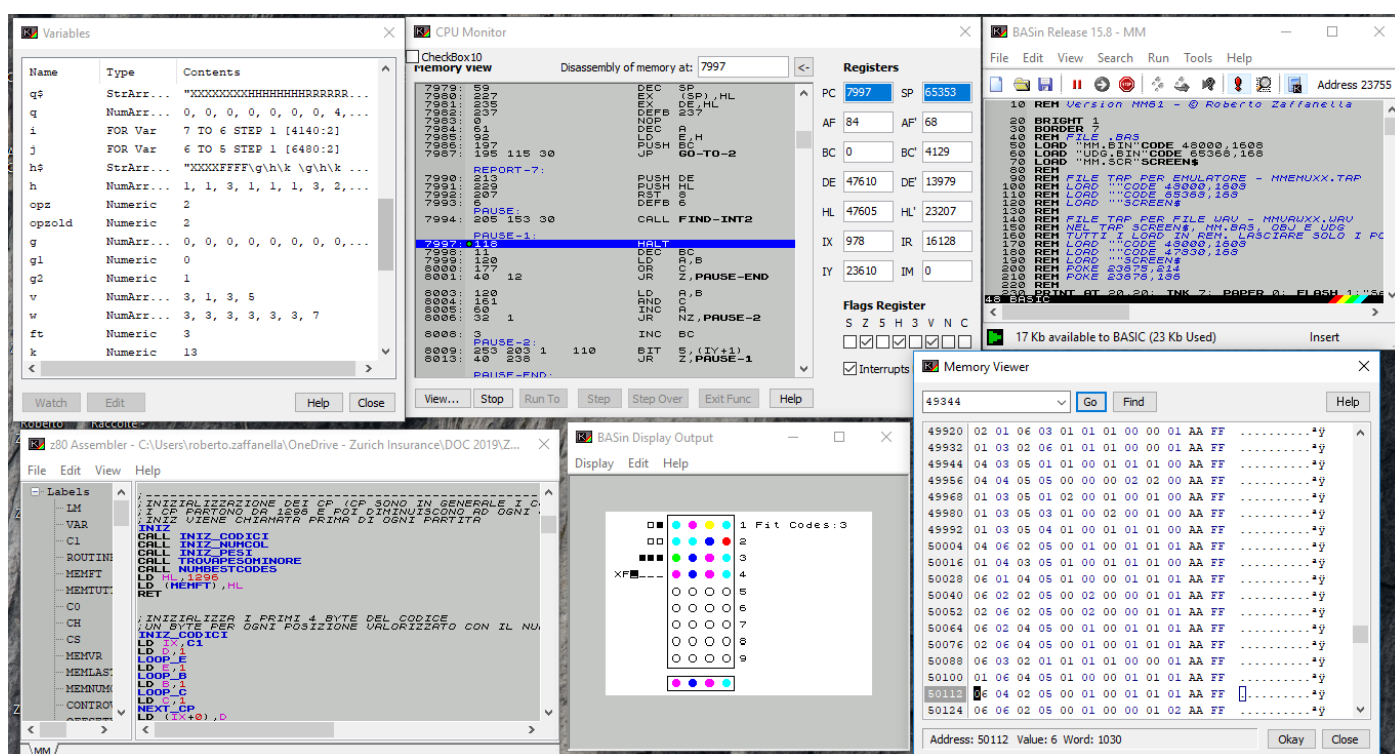
Nel caso però consiglio la versione 15.8. Con le versioni successive ho riscontrato dei problemi.

Qui il link per il download:

<https://sites.google.com/site/ulaplus/home/zx-spin-and-basin> 15.8

Si scompatta in una cartella e non necessità di installazione.

Di seguito, uno screenshot del programma all'opera. Si possono vedere i listati in Basic e Assembler, il programma in esecuzione, il valore delle variabili, il contenuto della memoria e l'esecuzione del codice oggetto:



Un'ultima nota sulla scelta di usare l'inglese per i menu e le altre parti del programma.

Lo spazio video dello Spectrum è molto limitato paragonato a quello a cui siamo abituati da tempo e l'inglese ben si presta per economizzare lo spazio video. Inoltre, il programma può utilizzato da persone non italiane con più facilità.

Domande, segnalazioni di bug o suggerimenti sono benvenuti. Scrivere a zaffaroby@gmail.com

Buon divertimento 😊