A cura di Pierluigi Panunzi

Il calcolo delle espressioni (1ª parte)

La scorsa puntata abbiamo analizzato insieme le caratteristiche della routine che implementa il comando LET e più genericamente un'assegnazione di una certa espressione ad una variabile prefissata: la presenza o meno del comando LET, come ben sappiamo, è facoltativa nella versione standard dell'MBASIC, mentre è obbligatoria in altri computer (tipo lo Spectrum), non dotati di Basic standard. Diciamo subito che l'analisi di un'espressione è alquanto complicata, anche perché, come di consueto, noi effettueremo in questa sede l'«analisi dell'analizzatore di un'espressione» e perciò ci dovremo immedesimare in tale oggetto: come dati in input vedremo i caratteri o meglio la codifica dei caratteri costituenti una certa linea di programma, all'interno della quale compare un'espressione e viceversa, dopo una più o meno gravosa elaborazione, avremo come output il valore del FAC (Floating Accumulator), che come sappiamo contiene appunto il risultato finale da «ricopiare» nell'apposito spazio di memoria riservato alla variabile indicata nel comando stesso.

Nel nostro studio della routine ci addentreremo quasi senza saperlo in quel «campo minato» rappresentato dalla ricorsività, che non pochi programmatori associano quasi per default al Pascal, demandando a tale linguaggio ad alto livello il compito di sbrogliarsela in situazioni difficili.

Nel nostro caso troveremo, caso assai raro nell'MBASIC, una routine («principale»)... che chiama con una CALL una subroutine, all'interno della quale si trova nientemeno che una nuova CALL alla routine. No, il redattore non ha preso un abbaglio: viceversa il tutto è condito con un sapientissimo gioco di salvataggi e ripristini di registri vari nello stack, grazie ai quali si memorizza lo «stato» di una certa situazione prima di effettuare una chiamata ricorsiva alla stessa routine da cui si era partiti. Il tutto è a sua volta legato all'ampiezza dello stack e perciò alla quantità di memoria compresa tra la fine della zona riservata al nostro programma e l'inizio del CP/M vero e proprio: a tale scopo come vedremo in seguito esiste un'apposita subroutine «sentinella» avente il compito di controllare che vi sia sufficiente spazio di memoria ed in caso contrario di interrompere l'elaborazione con un perentorio «Out of Memory Error».

Il fatto che tendenzialmente un'espressione richieda una certa dose di ricorsività non è nemmeno tanto appariscente, ma salta fuori solo dopo aver analizzato una certa espressione (anche banale, come vedremo) con «gli occhi del computer». Supponiamo di voler analizzare l'espressione facile facile

```
5 + sin (x + 3)
```

Dato per scontato che si tratta di un'espressione e che perciò ci troveremo ad eseguire appunto la routine di calcolo di un'espressione, ad un certo momento, dopo aver letto il byte che codifica la funzione «seno» (e perciò ben all'interno della routine «calcolo di espressione»), troveremo un'altra espressione, da calcolare a parte, in quanto tale è l'argomento di una funzione trigonometrica.

Arrivati a questo punto la routine relativa al «seno» (alla quale siamo entrati tramite un salto, grazie alla «jump table» oramai famosa), richiederà ancora una volta l'intervento della routine di analisi di un'espressione, che questa volta deve «ricominciare da zero» un nuovo calcolo...

Lasciamo al lettore il compito di estrapolare il ragionamento alle funzioni «ingegneristiche» del tipo «seno di logaritmo di coseno di esponenziale di arcotangente»: ogni volta che si incontra una funzione si deve salire di livello per calcolare una nuova espressione. Al termine della valutazione dell'ultimo argomento si procederà automaticamente a ritroso scendendo da un livello a quello precedente, per ottenere alla fine un risultato unico.

Il tutto, lo ripetiamo ancora una volta, con *un'unica* routine di valutazione di un'espressione...

Per arrivare ora ad analizzare come è implementata la ricorsività nell'Assembler dell'MBASIC, dobbiamo premettere parecchie considerazioni: in particolare in questa puntata andremo ad analizzare in quale modo vengono codificate le costanti numeriche che appaiono nelle nostre linee di programma, anche perché riteniamo utile vedere che cosa si sono inventati i progettisti della Microsoft nell'implementare l'MBASIC.

Le costanti numeriche in un programma

Per quanto riguarda la gestione di costanti numeriche, l'MBASIC, a differenza di altre versioni di Basic non standard, è senz'altro il più «intelligente», in quanto codifica le costanti già all'atto dell'impostazione di una linea di programma, a seguito della pressione del tasto RETURN e non al momento dell'esecuzione del programma, risparmiando spazio di memoria ed in particolare sul tempo di esecuzione del programma, non dovendo appunto calcolare i valori delle costanti all'atto dell'esecuzione.

Questo è senz'altro un vantaggio in quanto non ci accorgiamo minimamente di quanto succede a seguito della pressione del tasto RETURN, tanto

```
1305H   INC HL
1306H   LD A,(HL)
        CP 3AH
        RET NC
130AH   CP 20H
        JP Z,1305H
        JP NC,1386H
        OR A
        RET Z
        CP 0BH
        JP C,1381H
        CP 1EH
        JP NZ,1323H
        LD A,(0A6CH)
        OR A
        RET
1323H   CP 10H
        JP Z,1360H
        PUSH AF
        INC HL
        LD (0A6CH),A
        SUB 1CH
        JP NC,1366H
        SUB 0F5H
        JP NC,133EH
        CP 0FEH
        JP NZ,1352H
        LD A,(HL)
        INC HL
133EH   LD (0A6AH),HL
        LD H,0
1343H   LD L,A
        LD (0A6EH),HL
        LD A,2
        LD (0A6DH),A
        LD HL,138CH
        POP AF
        OR A
        RET
1352H   LD A,(HL)
        INC HL
        INC HL
        LD (0A6AH),HL
        DEC HL
        LD H,(HL)
        JP 1343H
135DH   CALL 138EH
1360H   LD HL,(0A6AH)
        JP 1306H
1366H   INC A
        RLCA
        LD (0A6DH),A
        PUSH DE
        PUSH BC
        LD DE,0A6EH
        EX DE,HL
        LD B,A
        CALL 28BEH
        EX DE,HL
        POP BC
        POP DE
        LD (0A6AH),HL
        POP AF
        LD HL,138CH
        OR A
        RET
1381H   CP 09H
        JP NC,1305H
1386H   CP 30H
        CCF
        INC A
138CH   DEC A
        RET
```

Figura 1 - Listato della subroutine dell'MBASIC, posta a partire dall'indirizzo 1305H, che effettua la scansione del testo codificato di un programma e decodifica i valori delle costanti.

ci sembra istantanea la risposta del calcolatore.

In altri Basic invece le costanti vengono espresse in memoria come una sequenza di caratteri ASCII e sono convertite solo in fase di esecuzione: in quest'ultimo caso chi trae vantaggio dalla codifica «posticipata» è il comando LIST, che semplicemente riporterà su video quanto memorizzato (almeno per ciò che riguarda le costanti, in quanto sappiamo che le funzioni ed i comandi sono comunque codificati); invece nel caso dell'MBASIC la routine che implementa il comando LIST è, come vedremo in una prossima puntata, alquanto più complessa, richiedendo la decodifica delle costanti.

Ora analizzeremo la codifica delle costanti, codifica che si differenzia enormemente in base al «tipo» della costante in esame; per effettuare l'analisi di tali codifiche abbiamo lavorato in questa maniera:

— abbiamo impostato di volta in volta delle linee di programma molto semplici, contenenti varie assegnazioni di costanti e variabili, cercando di coprire tutte le combinazioni possibili

— fatto ciò, con il comando SYSTEM siamo tornati al CP/M da dove abbiamo eseguito il comando «ZSID MBASIC. COM», con il che abbiamo caricato in memoria sia l'MBASIC che l'analizzatore ZSID, senza però perdere il contenuto della memoria, laddove era stato memorizzato il nostro programmino: in particolare tale zona nel nostro caso (Osborne I con MBASIC.COM release 5.21) iniziava a partire dall'indirizzo 6540H

— grazie allo ZSID abbiamo quindi analizzato tale zona ottenendo dei risultati di cui ora andremo a riferire: da questi ci accorgeremo che l'MBASIC tratta in modo differente anche costanti dello stesso tipo.

Costanti intere

Sono considerate di questo tipo tutte le costanti senza «punto decimale», di valore assoluto compreso tra Ø e 32768 (per la precisione comprese tra −32768 e 32767). A loro volta, a seconda se il valore assoluto è minore o maggiore di 256, allora vengono codificate con uno o due byte rispettivamente.

A loro volta, ancora, le quantità che potrebbero essere codificate con un byte comportano un'ulteriore distinzione: se le costanti sono ad una sola cifra allora si ha una certa codifica, che è diversa da quella che si ha se le cifre sono due o tre.

Facciamo subito vari esempi iniziando da valori ad una cifra per arrivare al massimo consentito:

Valore digitato (decimale)	Codifica (esadecimale)
Ø	11
1	12
2	13
...	
9	1A
10	ØF ØA
11	ØF ØB
12	ØF ØC
...	
24	ØF 18
...	
128	ØF 8Ø
...	
255	ØF FF
256	1C ØØ Ø1
257	1C Ø1 Ø1
258	1C Ø2 Ø1
...	
1000	1C E8 Ø3
...	
32767	1C FF FF

Osservando la tabella troviamo che:

— le costanti di valore compreso tra 0 e 9 vengono codificate con valori compresi tra 11H ed 1AH

— le costanti di valore compreso tra 10 e 255 vengono codificate con il loro byte esadecimale, preceduto dal prefisso ØFH

— le costanti di valore compreso tra 256 e 32768 vengono infine codificate con due byte esadecimali, stavolta preceduti da un prefisso pari ad 1CH.

Tutto ciò accade indipendentemente dal tipo della variabile che appare nell'assegnazione, come dire che anche nel caso di una linea di programma del tipo di:

```
10 A = 20 : B% = 20 : C = 20
```

il valore 2Ø è *sempre* espresso come ØFH 14H, anche se prima del segno «=» abbiamo rispettivamente variabili reali, intere ed in doppia precisione: ci penserà la routine che abbiamo visto la volta scorsa all'interno della «LET» ad effettuare le dovute conversioni.

Come caso a parte si hanno le costanti espresse direttamente in notazione esadecimale o ottale, rispettivamente grazie agli operatori «&H» ed «&O» (oppure più semplicemente «&»).

Ad esempio si ha:

Valore digitato	Codifica (esadecimale)
&H23	ØC 23 ØØ
&H1BCD	ØC CD 1B
&077 oppure &77	ØB 3F ØØ
&05 oppure &5	ØB 05 ØØ

Dimenticavamo di aggiungere che nel caso di costanti negative, l'MBASIC semplicemente traduce il «−» con il byte esadecimale F3H mentre effettua la codifica del valore assoluto secondo le modalità già viste.

Costanti reali

Sono considerati tali i valori contenenti il punto decimale oppure la lettera «e» ad indicare la notazione esponenziale: la codifica avviene in 4 byte, secondo le modalità di cui abbiamo parlato nel numero 42 di MC, preceduti da un prefisso dato dal byte esadecimale 1DH: ad esempio si ha:

Valore digitato (reale)	Codifica (esadecimale)
2.3	1D 33 33 13 82
.5	1D ØØ ØØ ØØ 8Ø
2e-5	1D AB C5 27 71

dove ancora una volta abbiamo indicato successivamente i byte così come si leggono scandendo la memoria per indirizzi crescenti.

Costanti in doppia precisione

Per questo tipo di costante riportiamo un unico esempio:

Valore digitato	Codifica (esadecimale)
	1F 84 47 1B 47
2d-5	AC C5 27 71

dove gli ultimi quattro byte ricordano la codifica in singola precisione di 2e-5 e dove il prefisso vale 1FH.

Ecco che perciò possiamo redigere una tabellina di trascodifica, che in

base al valore trovato subito dopo al segno dell'« = » (codificato con FØH) oppure in corrispondenza di una costante, fornisce il tipo e talvolta già il valore della costante stessa:

Prefisso o valore (esa)	Costante
ØB	ottale («&Ø» o «&»)
ØC	esadecimale («&H»)
ØD	—
ØE	—
ØF	intera ad un byte
1Ø	—
11	Ø
12	1
13	2
14	3
15	4
16	5
17	6
18	7
19	8
1A	9
1B	—
1C	intera a due byte
1D	singola precisione
1E	—
1F	doppia precisione

Ai lettori lasciamo il divertente compito di analizzare cosa succede provando ad usare i «prefissi» apparentemente inutilizzati e cioè ØDH, ØEH, 1ØH, 1BH ed 1EH.

Noi abbiamo provato alcune combinazioni, partendo dalla linea di programma

«1Ø A = &77»

La cui costante viene codificata con ØBH 3FH ØØH e sostituendo via via il prefisso ØBH con gli altri valori, per mezzo di opportune POKE: abbiamo poi listato le linee di programma così ottenute e non ancora contenti le abbiamo eseguite.

Ecco cosa abbiamo ricavato:

Byte di prefisso	listing	dopo l'esecuzione
ØB	1Ø A = &077	A = 63
ØC	1Ø A = &H3F	A = 63
ØD	1Ø A = 51461	overflow in 1Ø
ØE	1Ø A = 63	A = 63
ØF	1Ø A = 63	A = 63
1Ø	1Ø A = 3Ø	missing operand in 1Ø
1B	1Ø A = 1Ø?	syntax error
1E	1Ø A = 3Ø	missing operand in 1Ø

A parte perciò i casi leciti (ottale, esadecimale ed intero ad un byte), negli altri casi si sono ottenuti risultati divertenti e strani:

— che relazione ci sarà mai tra il valore 51461 ed i byte esadecimali di codifica? Sembra corretta in questo caso la segnalazione di overflow

— da dove esce fuori quel «3Ø» quando usiamo i prefissi 1ØH ed 1EH ed ancora, chi è l'operando che manca?

— sarà proprio un «?» vero e proprio quello che segue il 1Ø, altrettanto misterioso come origine, tanto che in esecuzione si ottiene «syntax error»? Ricordiamo a tale proposito che 3FH è la codifica ASCII del «?»: ma gli altri byte prima e dopo?

Dopo queste «note di colore» torniamo al nostro problema: la scansione del testo con successiva decodifica di eventuali costanti viene effettuata da una nostra «vecchia conoscenza», la routine che parte all'indirizzo 13Ø5H e che abbiamo infatti già incontrato nelle scorse puntate: mentre finora dicevamo che genericamente tale routine scandisce byte dopo byte il testo, ora andiamo proprio ad analizzarla.

La routine 13Ø5H: scan del testo e decodifica delle costanti

Osserviamo il listato di figura 1: innanzitutto ricordiamo che stiamo analizzando byte dopo byte una linea di programma, avendo come puntatore HL: si tratta in definitiva di una routine formata da una lunga serie di test, per ottenere un certo funzionamento in base al valore del byte.

Subito troviamo l'incremento del puntatore HL, per poter accedere al successivo byte, quello dunque da analizzare. La routine verrà subito abbandonata nel caso che il byte letto abbia un valore maggiore di 3AH e cioè nel caso di lettere (nomi di variabili) e nel caso di codifica di funzioni o comandi (vedasi in tale proposito l'ormai nota «jump table», del numero 38 di MC): tali casi non interessano alla routine in questione, ma verranno processati da altre subroutine.

Subito dopo troviamo il test se il byte letto è un «blank», nel qual caso verrà ignorato per passare al byte successivo, oppure se il valore è maggiore di 2ØH, ma pur sempre minore di 3AH, a causa del test precedente: in questo caso significa che il byte è un qualsiasi simbolo («!», «(», «%», ecc.) oppure una cifra tra 0 e 9 nel qual caso si ha l'uscita dalla routine rispettivamente con il Carry non settato (simboli) oppure settato (cifre), entrambe con la condizione di «NZ».

Il test successivo riguarda il caso in cui il byte è nullo, indicante la fine fisica della linea di programma, nel qual caso si uscirà con la condizione ovvia di «Z»: tale condizione sarà, per la routine di calcolo di un'espressione, un'indicazione di errore in quanto significa che si è arrivati alla fine della linea di programma quando ancora ci si aspettava la continuazione dell'espressione.

Il test successivo riguarda il caso in cui il byte letto sia o un «TAB» (Ø9H) oppure un «Line Feed» (ØAH), che servono unicamente per migliorare la leggibilità del programma e come tali vengono ignorati; altri valori compresi tra Ø1H e Ø8H provocano l'uscita dal programma con la condizione «NZ» e «NC».

Invece valori del byte letto compresi tra ØBH ed 1FH (quasi tutti prefissi a parte le già note eccezioni) riescono a superare anche quest'ultimo test: in particolare però i valori 1EH prima e 10H vengono processati a parte.

Ecco che finalmente rimangono i valori tra ØBH e ØFH, tra 11H e 1DH nonché 1FH: con altri test, preceduti da sottrazioni, si suddividono in altre quattro parti, a seconda, lo sappiamo già, del tipo di costante a cui si riferiscono.

Se il byte vale 1CH, 1DH o 1FH, allora si salta all'indirizzo 1366H, nel quale troviamo una routine che copia rispettivamente i successivi 2, 4 o 8 byte, tanti quanti devono essere i byte per delle costanti intere, reali o in doppia precisione in un «accumulatore temporaneo» (che chiameremo in seguito AT). Se invece il byte era compreso tra 11H e 1BH allora (a parte il caso di 1BH) si trattava della codifica di una «costante con una sola cifra»; un byte pari a ØFH invece comporta il caricamento del byte successivo (caso della «costante intera ad un solo byte») nel registro L, con contemporaneo caricamento di H con Ø, per ricostruire così in HL un valore completo esadecimale, che verrà poi depositato nell'AT.

Infine se il byte era compreso tra ØBH e ØEH, allora si ottiene la memorizzazione in HL del valore della costante, rappresentata dai due byte successivi, anch'esso poi memorizzato nell'AT. Sia nel caso di costanti intere, reali, ed in doppia precisione, sia nel caso in cui il contenuto di HL viene posto nell'AT, si avrà un'uscita dalla subroutine «13Ø5H» con le condizioni «NC» e «NZ».

Riassumendo, a seconda del o dei byte incontrati, si avranno le seguenti condizioni:

Byte incontrato	Condizioni	
cifra tra 0 e 9 in ASCII	CY	NZ
simbolo generico, lettera o codice di comando o funzione	NC	NZ
fine della linea	NC	Z

Tali condizioni ci torneranno molto utili nell'analisi della routine di calcolo di un'espressione, che vedremo nella prossima puntata. MC

CEIN S.r.l. CENTRO INFORMATICA

DIV. VENDITA PER CORRISPONDENZA

Vendita per corrispondenza: Via Merano, 1/2 - 16154 GENOVA - Tel. 010-673522
Esposizione e punto vendita: Via Merano, 3r - 16154 GENOVA
ESTRATTO DAL NOSTRO CATALOGO GENERALE

Hardware Amstrad

HA1001	Interfaccia seriale RS 232 con software	189.000
HA1002	Interfaccia parallela Centronics 8 bit	85.000
HA1003	Interfaccia 2 vie da 8 bit	99.900
HA1011	Cavi per stampante parallela Centronics	48.500
HA1012	Stampante Riteman per Amstrad completa di cavi	899.000
HA1021	Secondo drive 5" 1/4 Timatic con software	599.000
HA1031	Light pen con software	85.000
HA1041	Sintonizzatore per ricevere i programmi tv direttamente sul monitor	285.000
HA1051	Sintetizzatore vocale stereo completo di 2 casse acustiche	149.000
HA1061	Registratore per Amstrad CPC664/6128 con cavo	79.000

Software Amstrad CPC 464/664/6128

SA1001	Dun Darach	Arcade/Avventura	12.000
SA1002	Super Pipeline II	Arcade	10.000
SA1003	Rocco	Sportivo	12.000
SA1004	Frank Bruno's boxing	Sportivo	13.000
SA1005	War Lord	Avventura	12.000
SA1006	Rally II	Sportivo	12.000
SA1007	Fantastic Voyage	Arcade/Avventura	12.000
SA1008	A view to a kill	Arcade	10.000
SA1009	Daley Thompson Dechatlon	Sportivo	10.000
SA1010	Alien 8	Arcade/Avventura	13.000
SA1011	Knight Lore	Arcade/Avventura	13.000
SA1012	Gilligan's Gold	Arcade	10.000
SA1013	Fruit Frank	Arcade	11.000
SA1014	Sorcery	Arcade/Avventura	10.000
SA1015	Jet Boot Jack	Arcade	10.000
SA1016	Arnheim	War Games	12.000
SA1017	Air Traffic Control	Simulazione	10.000
SA1018	The Hobbit	Avventura	12.000
SA1019	Gremlins	Avventura	13.000
SA1020	Jewels of Babilon	Avventura	10.000
SA1021	Fighter Pilot	Simulazione	10.000
SA1022	Android I	Arcade	10.000
SA1023	Confuzion	Strategia	13.000
SA1024	Psychedelia	Grafica	13.000
SA1025	Starion	Arcade	13.000
SA1026	Project Future	Arcade	10.000
SA1027	Macadam Bumpers	Flipper	13.000
SA1028	Beach Head	Strategia/Arcade	15.000
SA1029	3D Boxing	Sportivo	14.000
SA1030	Everyone's a Wally	Avventura/Arcade	15.000
SA1031	The Hacker	Strategia	15.000
SA1032	Copy V 2.0	Copiatore universale	30.000
SA1033	Syclone II	Copiatore	25.000

Hardware Commodore 64/128

HC1001	Interfaccia parallela Centronics Cardco	199.000
HC1011	Stampante Riteman C+	767.000
HC1021	Fast load: Velocizza 5 volte il tuo 1541	45.000
HC1022	Smagic: Copia da disco a disco e cassetta	99.900
HC1023	Speeddos: Velocizza 20 volte il tuo 1541	159.000
HC1031	Pad numerico con software	99.000
HC1032	Mouse con software	189.000
HC1033	Super sketch: Tavoletta grafica con software	249.000
HC1034	Light pen con software	95.000
HC1035	La moviola per rallentare fino a fermare i video games	69.900
HC1036	Tastiera Music 64 con software	240.000
HC1051	Voice master interfaccia per parlare al tuo Commodore	230.000
HC1052	Sintetizzatore vocale Currah con software	119.000
HC1061	Copia programmi da registratore a registratore	29.500
HC1062	Azimuth controller per registratore 1530 e 1531	18.000
HC1071	Digitalizzatore di immagini con software	449.000
HC1081	Dust cover ripara il tuo computer dalla polvere (anche per C16)	12.900

Software Commodore 64/128

SC1001	Franck Brunos Boxing	Sportivo	12.000
SC1002	Winter Games	Sportivo	12.000
SC1003	Summer Games II	Sportivo	12.000
SC1004	Speed King	Sportivo	12.000
SC1005	Five A Side	Sportivo	10.000
SC1006	A View to a Kill	Arcade	10.000
SC1007	Barry Mc Gillian Boxing	Sportivo	10.000
SC1008	Exploding Fist	Sportivo	12.000
SC1009	Frankie Goes to Hollywood	Avventura	14.000
SC1010	Beach Head II	Arcade	12.000
SC1011	Tour de France	Sportivo	10.000
SC1012	Blackwhyce	Arcade	12.000
SC1013	Rescue on Fractalus	Arcade	10.000
SC1014	Who dares I	Arcade	10.000
SC1015	Who dares II	Arcade	12.000
SC1016	Kennedy Approach	Simulazione	10.000
SC1017	Karateka	Sportivo	12.000
SC1018	Paradroid	Labirinto	10.000
SC1019	Everyone's a Wally	Arcade/Avventura	12.000
SC1020	Tir na nog	Arcade/Avventura	12.000
SC1021	Spy Vs Spy I	Arcade	12.000
SC1022	Spy Vs Spy II	Arcade/Avventura	12.000
SC1023	Gremlins	Avventura	12.000
SC1024	Spiderman	Avventura	10.000
SC1025	Hulk	Avventura	10.000
SC1026	Castle of Terror	Avventura	10.000
SC1027	Lords of Midnight	Avventura	10.000

Hardware Spectrum

HS 1001	Interfaccia joystick configurazione Kempston	29.900
HS1002	Interfaccia monitor	35.000
HS1003	Interfaccia ram Turbo	89.000
HS1004	Interfaccia sonora per Spectrum	59.000
HS1021	Wafadrives: Doppio drive per spectrum 128K + 128K + word processor	399.000
HS1031	Light pen con software	72.000
HS1051	Sintetizzatore vocale Currah	99.000
HS1061	Registratore per Spectrum	69.000
HS1091	Joystick con interfaccia incorporata	44.500

Software Spectrum

SS1001	Tau Ceti	Arcade	10.000
SS1002	The Way of Exploding Fist	Sportivo	10.000
SS1003	Daley Thompson Supertest	Sportivo	10.000
SS1004	I of the Mask	Arcade	10.000
SS1005	Mission Impossible	Arcade/Avventura	10.000
SS1006	Dam Busters	Simulazione	12.000
SS1007	Waterloo	Strategia	13.000
SS1008	Frankie Goes To Hollywood	Avventura	13.000
SS1009	Frank Bruno's Boxing	Sportivo	13.000
SS1010	XCel	Arcade	12.000
SS1011	Macadam Bumpers	Flipper	12.000
SS1012	Formula 1	Simulazione/Sportivo	11.000
SS1013	Rally Driver	Sportivo	10.000
SS1014	Colt	Compilatore	15.000
SS1015	Monty on the Run	Arcade	12.000
SS1016	One on One	Arcade	12.000
SS1017	Farlight	Arcade/Avventura	12.000
SS1018	Geoff Capes Strongman	Sportivo	12.000
SS1019	Beach Head II	Arcade	12.000
SS1020	Disc of Death	Arcade	12.000
SS1021	International Karate	Sportivo	12.000
SS1022	Rollercoaster	Arcade	10.000
SS1023	Saboteur	Arcade	10.000
SS1024	Critical Mass	Arcade	12.000
SS1025	International Rugby	Sportivo	10.000
SS1026	Terrormolinos	Avventura	12.000
SS1027	Jems of Blood	Avventura	10.000
SS1028	Gyroscope	Arcade	10.000
SS1029	Tomahawk	Simulazione	14.000
SS1030	Fightin Warriors	Sportivo/Epico	10.000
SS1031	Popeye	Arcade	10.000
SS1032	Robin of Sherwood	Avventura	12.000
SS1033	Blast 3.0	Compilatore	14.000
SS1034	Highway Encounter	Arcade	10.000
SS1035	Marsport	Arcade/Avventura	12.000
SS1036	Riddler's Den	Arcade	10.000
SS1037	Endurance	Simulazione/Sportivo	10.000

Hardware Vario

HV 1011	Nastri per stampante Mps 801	10.500
HV1012	Nastri per stampante Mps 802	13.500
HV1021	Floppy disk 5" 1/4 SS/DD in confezione da 50 dischetti	128.000
HV1022	Floppy disk 5" 1/4 SS/DD 10 dischetti con box plastico	30.000
HV1024	Floppy disk 3"	13.500
HV1081	Cavo di collegamento computer = > televisore 3mt.	9.900

Software C16

SD1001	Shoot it	Arcade	13.000
SD1002	Rig Attack	Arcade	13.000
SD1003	Super Gran	Arcade	13.000
SD1004	Climb it	Arcade	13.000
SD1005	Blage	Arcade	13.000
SD1006	Munch it	Arcade	13.000
SD1007	Zodiaco	Arcade	13.000
SD1008	Star Event	Arcade	14.000
SD1009	Wacky Painter	Arcade	14.000
SD1010	Xargon Wars	Arcade	14.000
SD1011	Calcio	Sportivo	15.000
SD1012	Roller Kong	Arcade	14.000
SD1013	Slippery Sid	Arcade	14.000

PREZZI IVA INCLUSA

Inoltre: *disponibilità software per* **ATOM ACORN** *e* **MSX**
Garanzia CeinPost: *la CEIN S.r.l. si impegna a sostituire o a rimborsare la merce purchè rispedita entro otto giorni dalla consegna.*
Catalogo: *per ricevere il nostro catalogo ed il notiziario CEINPAPER inviare L. 3000 rimborsabili con il primo ordine.*
Ordinazioni: *per una migliore gestione dei Vs ordini si prega di indicare chiaramente Nome, Cognome, Indirizzo, Numero telefonico e codice degli articoli da Voi scelti.*
Pagamento: *contrassegno al ricevimento della merce. Per importi inferiori a L. 300.000 saranno addebitate L. 5000 per spese di imballaggio e spedizione.*

CENTRO PILOTA AMSTRAD