Routine per HP 41

Francesco Balena - Bari

Spett.le MCmicrocomputer
accludo alla presente del materiale per la rubrica di software RPN; si tratta di un paio di routine e della descrizione di alcuni "trucchi" che è possibile usare programmando con la HP 41 C/CV.

Per questi ultimi non si tratta di niente di eccezionale, alcuni suggerimenti che però prima o poi vengono utili, e che hanno il pregio, almeno per quanto mi è dato di conoscere, di essere inediti (con l'eccezione del primo argomento del paragrafo "Qualche istruzione in più", che ho ritrovato su "Calculators tips & routines").

Segue un programma per la risoluzione di equazioni diofantine lineari, cioè di equazioni della forma: Ax + By = C dove si impone che le radici siano entrambe intere. Anche se il termine "diofantino" è poco noto, queste equazioni sono alla base della risoluzione di molti problemi di matematica ricreativa e di enigmistica; quasi sempre esse vengono risolte per tentativi (almeno le più semplici) perché il metodo generale è poco noto ed è comunque abbastanza complicato. Una volta tanto la HP 41 non solo permette di risparmiare tempo e calcoli noiosi, ma affronta un problema che l'utente medio non saprebbe come risolvere da solo.

Il secondo programma che vi presento è una brevissima routine che aggiunge una potente macro-istruzione disponibile solo su pochi grossi computer. Le istruzioni "non ortodosse" sono facili da ottenere per chi si intende anche un minimo di Programmazione Sintetica; poiché MCmicrocomputer ha già presentato il Byte Jumper ed il metodo per ottenerlo, al listing è accluso un elenco di operazioni da eseguire per sintetizzare tutte le istruzioni sintetiche, facendo uso esclusivamente del Byte Jumper.

Qualche istruzione in più....

1) Come sappiamo, nel set di istruzioni della HP 41 C mancano i test di confronto X> = Y? e X> = 0?; essi possono essere simulati nel modo seguente

```
.                    .
.                    .
X≠Y?          e      X≠0?
X>Y?                 X>0?
.                    .
.                    .
```

2) Chi, come il sottoscritto, si trova ogni tanto a tradurre in RPN dei programmi scritti per le Texas TI 58/59, avrà notato l'assenza sulla HP 41 C dell'istruzione analoga a "INV DSZ nn", che funziona esattamente al contrario della istruzione "DSE"; la linea di programma seguente viene eseguita se il contenuto del registro nn è minore o uguale a zero.

Sulla HP 41 questa istruzione può essere simulata dalla sequenza:

```
.
.
DSE nn
FS? 52
GTO 99
.
.
```

Il contenuto di R nn è decrementato ed il salto alla etichetta 99 viene eseguito solo se esso è ora uguale o minore di zero.

Ad esempio, nel programma:

```
10
STO 00
LBL 01
DSE 00
FS? 52
GTO 02
.
.
.
GTO 01
```

le istruzioni al posto dei puntini vengono eseguite nove volte, dopo di che si salta alla etichetta 02.

Analogamente funziona la sequenza:

```
.
.
IGS nn
FS? 52
.
.
```

3) Un'altra istruzione assente sulla HP 41 C è quella che permette di azzerare i registri che contengono gli indirizzi di ritorno delle subroutine, mentre le TI 58/59 sono dotate dell'istruzione RST (che comunque oltre a fare ciò azzera i flag e posiziona il Program Counter sulla linea 000).

Una istruzione del genere può risultare molto utile dopo che abbiamo pasticciato un po' troppo confondendo "XEQ" e "GTO"; ad esempio, quando usciamo da una subroutine mediante una "GTO" in seguito ad una condizione eccezionale, ed in tal caso rimane un indirizzo di ritorno in sospeso.

La seguente routine, richiamabile da un programma mediante la semplice istruzione XEQ "RST", cancella tutti gli indirizzi di ritorno in sospeso, non disturba la catasta operativa (ad eccezione del registro L) ed è lunga solo 20 byte.

Se invece vogliamo incorporare questa sequenza all'interno del programma principale, possiamo eliminare le linee 01 e 10, mentre il "6" alla linea 04 deve essere sostituito da un "7" (con una occupazione di memoria di 12 byte soltanto).

```
01♦LBL "RST"   06♦LBL 00
02 SIGN        07 DSE L
03 CLX         08 XEQ 00
04 6           09 RTN
05 X<> L       10 END
```

Equazioni diofantine lineari

Sebbene la risoluzione di molti problemi dipenda dalla ricerca di radici intere di una equazione lineare a due incognite, a molti il termine "equazione diofantina" risulterà nuovo, e saranno ancora meno coloro che conoscono un metodo efficiente (che non sia cioè per tentativi) per risolvere questa classe di equazioni.

Per equazioni diofantina (lineare) si intende una equazione del tipo:

$$Ax + By = C$$

(con A, B, C interi relativi) dove si impone che le radici x_0 e y_0 siano entrambe intere. Una equazione diofantina può non avere soluzioni; ciò avviene quando il M.C.D. dei valori assoluti di A e B non divide C, e in tal caso il programma termina visualizzando il messaggio "NO INT. ROOTS".

D'altra parte è facile vedere che se una equazione diofantina ammette una soluzione (x_0, y_0), allora ammette infinite soluzioni, date da

$$x = x_0 + ad$$
$$y = y_0 + bd$$

dove il parametro d può assumere tutti i valori interi positivi o negativi. Se l'equazione ammette soluzioni, il programma visualizza le due formule parametriche per il calcolo delle coppie di radici.

Molto spesso si richiede che le due radici oltre ad essere intere siano anche non negative. La routine B del programma risolve le disequazioni e visualizza la coppia di radici per cui x è minimo (ma non negativo), oppure visualizza il messaggio "NO POS. ROOTS" se nessun valore di d soddisfa la condizione imposta.

L'algoritmo usato nel programma sfrutta le proprietà delle frazioni continue; il metodo è stato tratto dal testo "FRAZIONI CONTINUE" di C.D.Olds (Zanichelli, serie Matematica Moderna), a cui si può fare riferimento per maggiori chiarimenti.

L'utilizzazione del programma è veramente molto semplice; è sufficiente caricare i coefficienti A, B, C, (completi di segno) nei registri Z, Y e X della catasta operativa

— impostare A, premere ENTER ↑
— impostare B, premere ENTER ↑
— impostare C, eseguire XEQ "DIEQ"

dopo alcuni secondi di elaborazione viene

Equazioni diofantine lineari

```
01♦LBL "DIEQ"   42♦LBL 02   83 ARCL 08   124 FRC
02 STO 02       43 RCL 03   84 RCL 01    125 X>0?
03 X<> Z        44 RCL 04   85 X>0?      126 SIGN
04 STO 00       45 MOD      86 "⊢+"      127 ABS
05 STO 03       46 X≠0?     87 RCL 08    128 +
06 SIGN         47 GTO 01   88 X=0?      129 INT
07 ST* 00       48 FS?C 20  89 " X= "    130 FS? 20
08 ST* 02       49 GTO 04   90 1         131 GTO A
09 ST* 03       50 RCL 03   91 RCL 01    132 X<>Y
10 *            51 RCL 04   92 X<0?      133 ENTER↑
11 STO 01       52 X=Y?     93 "⊢-"      134 FRC
12 ABS          53 GTO 03   94 ABS       135 X<0?
13 STO 04       54 ST- 03   95 X≠Y?      136 SIGN
14 CLX          55 ST- Y    96 ARCL X    137 ABS
15 STO 05       56 GTO 01   97 "⊢d"      138 -
16 STO 08       57♦LBL 03   98 PROMPT    139 INT
17 1            58 RCL 05   99 " Y= "    140 "NO POS"
18 STO 06       59 STO 07   100 RCL 07   141♦LBL 05
19 STO 07       60 RCL 06   101 X≠0?     142 "⊢. ROOTS"
20 FIX 0        61 STO 08   102 ARCL 07  143 X<>Y
21 CF 20        62♦LBL 04   103 X≠0?     144 PROMPT
22 GTO 02       63 RCL 04   104 "⊢+"     145♦LBL A
23♦LBL 01       64 ST/ 00   105 RCL 00   146 STO 02
24 FC?C 20      65 ST/ 01   106 1         147 "d="
25 SF 20        66 ST/ 02   107 X≠Y?     148 ARCL X
26 X<> 04       67 "NO INT" 108 ARCL 00  149 RCL 00
27 ENTER↑       68 RCL 02   109 "⊢d"      150 RCL 01
28 X<> 03       69 FRC      110 PROMPT   151 X<> Z
29 X<>Y         70 X≠0?     111♦LBL B    152 ST* Z
30 /            71 GTO 05   112 RCL 07   153 *
31 INT          72 RCL 01   113 CHS       154 RCL 07
32 ENTER↑       73 X<0?     114 RCL 00   155 +
33 X<> 07       74 SF 20    115 /        156 X<>Y
34 ST* 07       75 CHS      116 RCL 08   157 RCL 08
35 X<> 05       76 STO 01   117 CHS       158 +
36 ST+ 07       77 SIGN     118 RCL 01   159 "⊢ X="
37 RDN          78 RCL 02   119 /        160 ARCL X
38 X<> 08       79 ST* 08   120 FS? 20   161 "⊢ Y="
39 ST* 08       80 *        121 X<Y?     162 ARCL Y
40 X<> 06       81 ST* 07   122 X<>Y     163 PROMPT
41 ST+ 08       82 " X= "   123 ENTER↑   164 END
```

visualizzata la formula parametrica per il calcolo di x; se invece l'equazione non ammette soluzioni intere viene mostrato il messaggio "NO INT. ROOTS".

Se l'equazione diofantina ammette soluzioni, facendo ripartire l'elaborazione si visualizza la formula parametrica per il calcolo di y. A questo punto possiamo ripetere all'infinito uno dei seguenti passi:

— impostare un valore (intero) ed eseguire XEQ A: viene mostrata la coppia (x, y) di radici corrispondenti a quel valore del parametro d.

— XEQ B per determinare il valore del parametro d per cui le radici sono entrambe non negative, e le radici stesse.

Se nessun valore (intero) di d soddisfa questa condizione viene visualizzato il messaggio "NO POS. ROOTS".

Il programma fa uso dei registri 00 ÷ 08 e del flag F 20.

Un punto a suo favore è sicuramente la alta velocità di esecuzione (massimo dieci secondi per le equazioni più complicate) e l'"eleganza" del suo output (buona parte del programma è dedicato alla composizione in ALPHA delle formule parametriche e dei messaggi di errore).

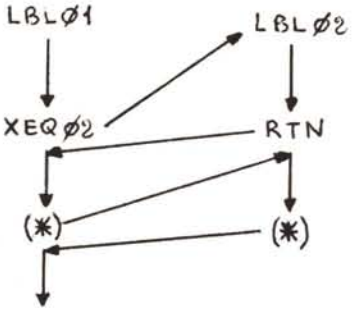
Nota: alle linee 120-122 viene implementato l'OR logico tra lo stato del flag 20 e il risultato di un test di confronto.

"Re": istruzione di 'resume'

Questo programma aggiunge una potente istruzione alla HP 41 C: si tratta dell'istruzione comunemente chiamata 'RESUME', che permette la costruzione delle cosiddette '*coroutines*'; essa è presente nel linguaggio assembler di alcuni grossi computer (ad es. il PDP-11, mentre persino l'IBM 360 ne è sprovvisto), ma a quanto ne so non è mai stata implementata su un personal, per non parlare ovviamente di calcolatrici programmabili.

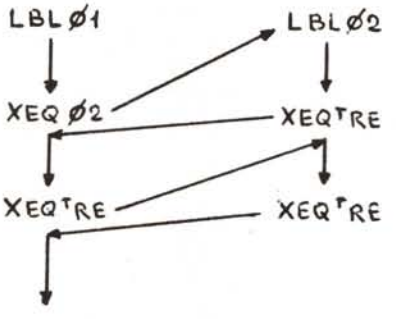
Supponiamo di avere in memoria due routine distinte; con il set di istruzioni che abbiamo a disposizione una delle due può chiamare l'altra come subroutine. Quando durante l'esecuzione viene incontrata una istruzione di "RTN" il controllo sarà riceduto alla routine chiamante, all'istruzione immediatamente successiva alla chiamata alla subroutine.

Supponiamo però di voler realizzare uno schema di esecuzione di questo tipo:



I primi due salti si possono effettuare facilmente mediante una "XEQ" e un "RTN". Analizziamo ora l'istruzione (*): essa non può essere un "XEQ 02" né può trattarsi di un "RTN", che in questo caso equivarrebbe ad uno "STOP". Il problema potrebbe aggirarsi mediante una serie di "GTO" che si riferiscono a delle etichette sparse nell'altra routine, ma questo sistema è quasi impossibile da realizzare se i salti sono condizionati, cioè se sono possibili flussi differenti.

Una soluzione alternativa, molto più pratica, veloce e meglio "strutturata" è l'introduzione di una speciale istruzione, che operando sul registro b della HP 41 C scambia tra di loro l'Address pointer ed il primo indirizzo di ritorno, permettendo di realizzare lo schema di flusso descritto sopra. Avremo cioè:



È importante notare che il primo salto deve essere effettuato con un normale "XEQ 02" poiché in quel momento non esiste alcun indirizzo di ritorno, ed una operazione di RESUME metterebbe uno "zero" 1 nell'Address Pointer provocando uno "STOP".

È inutile dire che per accedere al registro b occorrono delle tecniche di programmazione sintetica; la routine è breve e le istruzioni sintetiche sono circa una decina, e possono essere create senza particolari difficoltà mediante l'uso del Byte Jumper, seguendo le istruzioni riportate più avanti.

Re - istruzioni di resume

```
01♦LBL "RE"          15 X<> [   X<> M
02 SF 20             16 "⊢AAAAAA"
03 RCL b             17 X<> [   X<> M
04 STO [   STO M     18 "⊢AA"
05 RDN               19 STO [   STO M
06 FC? 20            20 RDN
07 CLA               21 "⊢AA"
08 FC?C 20           22 STO [   STO M
09 RTN               23 RDN
10 "⊢AAAAAA"         24 "⊢AA"
11 RCL \   RCLN      25 RCL \   RCLN
12 X<> [   X<>M      26 STO b
13 "⊢AAAAAA"         27 END
14 RCL \   RCLN
```

Chiariamo il funzionamento di questa routine.

Eseguendo un XEQ "RE" il corrente Address Pointer viene posto nel campo del primo indirizzo di ritorno, mentre il vecchio indirizzo di ritorno è "spinto" nel campo del 2° indirizzo di ritorno; quel che dobbiamo fare è scambiare tra loro il contenuto di questi due campi (che sono rispettivamente i byte 2-3 e 4-5 del registro b). Ciò può essere fatto caricando il contenuto del registro b in ALPHA e operando poi una serie di operazioni sulla stringa, usando i registri della catasta per conservare i risultati intermedi.

Alla fine dell'operazione il nuovo contenuto "modificato" del registro b è nel registro N (byte 8-14 del registro ALPHA) e quindi possiamo richiamarlo e memorizzarlo in b (linea 26); in questo preciso istante l'Address Pointer verrà caricato col valore che conteneva nel momento in cui si è effettuato il "RCL b" (linea 03) quindi l'elaborazione riprenderà da questa linea. L'uso del flag 20 permette di evitare un loop infinito: eseguendo un "RTN" nell'Address Pointer verrà automaticamente caricato il valore che ci interessa e l'elaborazione del programma principale riprenderà dalla istruzione successiva al penultimo XEQ "RE" incontrato.

Tra i vantaggi di questa routine vi è il fatto che non vengono utilizzati registri dati, ma solo i registri T e Z della catasta, in modo che i registri X, Y ed L possono essere usati per passare dei parametri tra le coroutine. Viene tuttavia usato il registro ALPHA, che viene "pulito" al termine della routine, ed il flag F 20, che viene lasciato disattivato.

Creazione delle istruzioni sintetiche

Invece di creare le istruzioni sintetiche una alla volta, le generiamo tutte insieme, dopo di che inseriremo le istruzioni "normali" usando il tasto SST. Il Byte Jumper (indicato con B.J. nelle istruzioni) deve essere assegnato ad un tasto qualsiasi.

Istruzioni preliminari: eseguire GTO... per posizionarsi in una zona libera della memoria di programma, entrare nel modo PRGM ed impostare le seguenti istruzioni: 1) LBL "RE" 2) SF 20 3) STO 01 4) "AB", poi eseguire GTO. 004 (sembra inutile ma non lo è), entrate nel modo USER per poter utilizzare il Byte Jumper e continuate con le istruzioni seguenti:

```
 1) PRGM(Off), B.J., PRGM(On)
 2) STO 25, SST, ←, ←          (sul display: 04 -)
 3) MEAN, RST                  (sul display: 04 "A■)
 4) PRGM(Off), B.J., PRGM(On)
 5) RCL 25, GTO 005, ←, LASTX, RST
 6) PRGM(Off), B.J., PRGM(On)
 7) STO 25, GTO 005, ←, RDN, RST
 8) PRGM(Off), B.J., PRGM(On)
 9) Esegui nuovamente le istruzioni ai punti 7 e 8
10) X  25, GTO 006, ←, RDN, RST
11) PRGM(Off), B.J., PRGM(On)
12) Esegui nuovamente le istruzioni ai punti 10 e 11
13) RCL 25, GTO 005, ←, LASTX, RST
14) PRGM(Off), B.J., PRGM(On)
15) X  25, GTO 006, ←, RDN, RST
16) PRGM(Off), B.J., PRGM(On)
17) Esegui nuovamente le istruzioni ai punti 13 e 14
18) Esegui nuovamente le istruzioni ai punti 7 e 8
19) RCL 25, GTO 005, ←, MEAN, RST
20) PRGM(Off), B.J., PRGM(On)
21) SIN, GTO 003, DEL 002
```

A questo punto possiamo inserire le altre istruzioni per completare la routine. Ad esempio, GTO.004 e caricare le istruzioni alle linee 05 ÷ 10 della routine, e così via.

* * *

Alla chiara esposizione dell'autore aggiungo una raccomandazione, la solita, circa la routine "RE": attenzione a non commettere errori durante l'impostazione delle istruzioni sintetiche; se tutto procede bene, prima di eseguire ogni sequenza "PRGM(off), B.J., PRGM(on)" accertatevi che sul display compaia la stringa "TA*".

Per quanto riguarda il punto "Qualche istruzione in più", va notato che il Flag 52 risulta sempre spento alle interrogazioni, quindi inserire una istruzione FS? 52 in un programma serve solo a saltare l'istruzione successiva durante l'elaborazione. Se il passo FS? 52 viene posto subito dopo una istruzione di test condizionale, non è difficile comprendere come essa possa invertirne l'effetto. Per esempio, consideriamo la sequenza

```
.
.
TEST?
FS? 52
GTO 99
.
.
```

se la risposta al TEST? (volendo indicare con ciò un qualsiasi test condizionale) è SÌ, verrà eseguita l'istruzione FS? 52 e quindi l'esecuzione salterà oltre l'istruzione GTO 99; se la risposta è NO, dalla istruzione TEST? l'esecuzione salterà direttamente al passo GTO 99; nel nostro esempio l'istruzione GTO 99 rappresenta l'istruzione che va saltata o meno, a seconda dell'esito del test. Da ciò deriva che le due istruzioni X>=Y? e X>=0? possono essere ottenute anche con le sequenze

```
.                    .
.                    .
X<Y?                 X<0?
FS? 52       e       FS? 52
.                    .
.                    .
```

Superfattoriale

Luigi Juspa - Savignano Scalo (AV)

Come è risaputo, la funzione fattoriale di cui è dotata la 41C permette il calcolo di n! con n ≤ 69. Per valori maggiori la comparsa del messaggio "OUT OF RANGE" indica il raggiungimento del limite operativo. Il programma che mi accingo a illustrare permette invece il calcolo del fattoriale di numeri molto maggiori. L'idea che è alla base dell'algoritmo è nata nel realizzare alcuni programmi di calcolo combinatorio, e in particolare un programma sul calcolo del semifattoriale. Ricordo brevemente che per semifattoriale si intende una produttoria parzializzata ai soli termini pari o dispari a seconda di n (es. 7!! = 7·5·3·1; 6!! = 6·4·2). Si deduce facilmente da ciò che un qualsiasi fattoriale può essere espresso nella formula: n! = n!!·(n-1)!!. È possibile ovviamente adottare anche produttorie più parzializzate e in generale risulterà: n! = nr!·(n-1)r!·(n-2)r!.....[n-(r-1)] r! indicando con r il numero di parzializzazioni: es. per r = 3 n! = n!!!·(n-1)!!!·(n-2)!!!, per r = 4 n! = n!!!!·(n-1)!!!!·(n-2)!!!!·(n-3)!!!! e così via.

Si comprende a questo punto che introducendo tali parzializzazioni, le produttorie risulteranno meno sature al crescere di r, facendo aumentare così il valore di n massimo calcolabile. Il programma calcola per prima cosa il valore di r più idoneo, (r troppo piccolo causa l'overflow, r troppo grande un aumento del tempo di elaborazione) quindi le varie produttorie, ed infine esegue le moltiplicazioni tra queste usando l'artifizio di scindere ogni fattore in mantissa ed esponente tramite semplici operazioni logaritmico-esponenziali. Il prodotto delle mantisse e la somma degli esponenti daranno rispettivamente la mantissa e l'esponente del fattoriale cercato. Tre precisazioni sono d'obbligo a questo punto:

1) Il numero ottenuto è, con tutta evidenza, non utilizzabile per calcoli successivi.

2) Pur essendo tecnicamente possibile, è sconsigliabile calcolare il fattoriale di numeri superiori a 500, perché a partire da tale valore di n la giusta determinazione di r può risultare difficoltosa in qualche caso particolare. Inoltre il tempo di elaborazione aumenta apprezzabilmente.

3) Non si è ritenuto di dover adottare alcuna procedura di controllo sui dati in ingresso, ma chi avvertisse questa necessità può facilmente modificare il programma in maniera opportuna. Due parole, le ultime, sull'utilizzo che è banale. Posto in X il valore di n, si richiama il programma con XEQ "SFT", oppure più comodamente assegnando "SFT" ad un tasto.

* * *

Un programma che non richiede altre parole, né aggiunte alle istruzioni per l'uso che è privo di difficoltà.

Il risultato, fornito sul registro ALPHA sotto forma di stringa, è rintracciabile in forma numerica nei registri R02 (mantissa) e R03 (esponente). MC

Superfattoriale

```
01*LBL "SFT"   20 +        39 RCL Y
02 FIX 4       21 STO 00   40 LOG
03 SF 29       22 1        41 INT
04 ENTER↑      23 STO 02   42 LASTX
05 ENTER↑      24 0        43 FRC
06 ENTER↑      25 STO 03   44 10↑X
07 41.5        26*LBL 02   45 ST* 02
08 X>Y?        27 RCL 00   46 RDN
09 RDN         28 INT      47 ST+ 03
10 /           29 STO L    48 DSE 00
11 .45         30*LBL 01   49 GTO 02
12 +           31 LASTX    50 CLA
13 INT         32 RCL 01   51 ARCL 02
14 STO 01      33 -        52 "├ E"
15 RCL Y       34 RCL Y    53 FIX 0
16 X<>Y        35 X<>Y     54 CF 29
17 -           36 *        55 ARCL 03
18 1 E3        37 X>0?     56 AVIEW
19 /           38 GTO 01   57 END
```

Synthetic Corrige

Nel precedente numero di MC (numero 19) sono stati pubblicati due articoli riguardanti la programmazione sintetica.

Come certamente qualche lettore avrà notato, i due articoli che sono nelle pag. 64, 65 e seg. presentano alcuni errori di stampa non molto dannosi ai fini dell'articolo, ma che potrebbero risultare fastidiosi se non venissero notati.

Per questo motivo viene qui riportata una tabella di correzione:

(N° 19 di MICROCOMPUTER)

PAG.	COLONNA	N. RIGA	INVECE DI	LEGGERE
65	3°	3,4	GOTO 01	GOTO .001
65	3°	9,10,11	si eseguono le operazioni etc.	eseguire le seguenti istruzioni: *
67	I° (fig. 5)	3	REG	ΣREG
67	2°	3	GOTO ...	GOTO ...

* Tralasciare la lettura del testo delle righe 10,11; riprendere dalla riga 12 (Impostare un SIZE etc.).

Francesco Guarnieri

Che cosa ha in più Personal Kid?

PERSONAL KID

	PREZZO (IVA escl.)
CPU BOARD 48 K RAM	650.000
Tastiera ASCII con pad numerico esteso e tasti funzionali	210.000
Alimentatore 80 W	150.000
Alimentatore switching 75 W	200.000
Contenitore	120.000
UNITÀ CENTRALE (48 K RAM, interfaccia per registratore, input analogici, lettere minuscole, BASIC, monitor e disassembler) completa di alimentatore, tastiera ASCII dotata di pad numerico esteso e tasti funzionali, contenitore	
Con tastiera incorporata	1.210.000
Con tastiera separata	1.260.000
UNITÀ CENTRALE con monitor	
Con tastiera incorporata	1.450.000
Con tastiera separata	1.500.000
UNITÀ CENTRALE con monitor 12'', drive 5'' e interfaccia per due drive	
Con tastiera incorporata	2.250.000
Con tastiera separata	2.300.000
Monitor 12'' fosfori verdi o gialli	250.000
Drive 5''	710.000
Interfaccia doppio drive	120.000
Espansione 16 K RAM	150.000

- Costo Basso
- Lettere minuscole
- Tastiera con pad numerico + i segni delle operazioni
- Repeat automatico
- Set di tasti funzionali per l'esecuzione immediata dei principali comandi
- Diretto controllo del cursore
- Zoccolo per memoria EPROM
- Disponibilità del sistema in versione open frame o vestita in più configurazioni

Compatibile Apple

Marketing plan ANCONA

SIPREL s.r.l. Via Di Vittorio, 82 - Tel. 071/8046305- Zona Ind.le Baraccola - 60020 Candia di Ancona

Cercasi Concessionari