da m&p COMPUTER 22 del Settembre 1982 – Rubrica SOA

LA PROGRAMMAZIONE SINTETICA SU TI-59

Perché?

La programmazione sintetica non è nata per caso, bensì dal bisogno di ogni utente di calcolatore di scoprire sempre più sulla sua macchina e di sezionarla, per così dire, fino all'ultimo bit.
Anche nel nostro caso questo bisogno ha spinto a fare alcune considerazioni sulla struttura della TI-59 e delle sue istruzioni, considerazioni non riportate su "Elaborazione dei Dati Personale", il manuale fornito dalla Texas Instruments assieme alla macchina e delle quali si parlerà nei prossimi capitoli.
Bisogna comunque notare due cose:

a) la TI ha fatto uso di programmazione sintetica, come testimonia il contenuto di alcuni programmi del modulo Solid Sate "Math-Utilities";
b) il materiale riguardante questa scoperta è stato pubblicato in modo frammentario e limitatamente a certi argomenti (quelli più utili del resto: funzione HIR), non è mai stato raccolto ordinatamente com'è stato fatto per l'HP-41C (vedi "Synthetic Programming on the HP-41C").

Questo scritto ha appunto lo scopo di riunificare quanto è stato fin qui scoperto e di aggiungere qualcosa di nuovo su questo argomento.
A ben guardare la programmazione sintetica non permette cose trascendentali, ma amplia comunque le vaste capacità della TI-59+ PC-100C:

- controllo e manipolazione dei registri del Sistema Operativo Algebrico;
- controllo e manipolazione delle stringhe alfanumeriche;
- uso delle funzioni di editing in un programma come funzioni "normali";
- autocancellazione del programma, codifica di un programma in registri, estensione di DSZ e LBL.

In ogni caso l'uso di queste tecniche non può danneggiare in alcun modo la TI-59 che al massimo segnalerà errore facendo lampeggiare il display; è da sottolineare, inoltre, l'utile apporto della stampante che ha permesso la decodifica di istruzioni e programmi con notevole chiarificazione degli stessi. Alla descrizione della programmazione sintetica di anteporrà una breve descrizione hardware della macchina, che ho ritenuto utile inserire per una più facile comprensione di alcuni concetti.

Note e ringraziamenti

Nello scritto vengono usate alcune convenzioni qui di seguito riportate:

- S.O è l'abbreviazione di Sistema Operativo;
- "default" indica la ripartizione iniziale della TI-59 cioè 479.59;
- "b" all'interno delle stringhe indica lo spazio bianco;
- la funzione sintetica HIR viene indicata indifferentemente con HIR, H, registro del S.O.;
- T&P significa uso sia da tastiera che da programma;
- f.s. e p.s. stanno per funzione sintetica e programmazione sintetica.

Ringrazio per l'aiuto gli amici Franchini Giuseppe e Apollonio Paolo che, prestandomi il primo la propria TI-59 ed il secondo la propria TI-58C, hanno permesso il controllo dell'esattezza di tutte le informazioni contenute in questo scritto.
Fornisco pure una piccola bibliografia che credo risulterà utile a chi volesse approfondire:

- P. Pannunzi: "Ancora sulla cresta dell'onda la TI-59" pagg. 35/41 su M&P computer n° 3 Febbraio-Marzo 1980
- Club STI-59: "L'uso dell'istruzione HIR nella manipolazione dei codici alfanumerici" su Bollettino STI-59, Aprile 1980 vol. 2 n°2
- P.Pannunzi:"S.O.A.: note" pag. 14 su M&P Computer n°12 Maggio/Giugno 1981
- P.Pannunzi: "L'angolo delle TI" pag. 61 su MC Micromputer n°4 Dicembre 1981

Dentro la TI-59

In questo paragrafo si darà una breve descrizione hardware della TI-59 che sarà senza dubbio utile per comprendere i principi della p.s. e per capire come sia possibile effettuare talune operazioni e come, invece, non si possa compierne altre.
La struttura della TI-59 è in regola con quella di un qualsiasi micro che si rispetti (Fig. 1): la esamineremo pezzo per pezzo.

- *La C.P.U.* E', com'è noto, il cervello della macchina e si divide in due parti fondamentali: la "Arithmetic Logic Unit" che manipola e gestisce i dati a 8 bit e la Memoria Macchina (RAM e ROM) cui si appoggia l'A.L.U. nella esecuzione delle istruzioni, T&P. Il S.O. è contenuto nella ROM e ad esso spetta il compito di dirigere il traffico di bit tra le varie parti del sistema. In seguito esamineremo il suo comportamento in LRN mode. Nella ROM risiedono pure tutte le micro-operazioni eseguibili sulla TI-59; la RAM, invece, è quella che ci interessa poiché, come si vedrà, è possibile interagire con essa: vi sono gli otto registri del S.O., tanti quanti sono le operazioni che possono essere tenute in sospeso, i registri delle parentesi, quelli dei flag, quelli del ritorno delle subroutines, il registro T e quello del visualizzatore sui quali il manuale si sofferma anche se in modo incompleto. Si specifica comunque che i flag sono 10 e, com'è noto, sono di tipo logico potendo sempre assumere i due valori logici VERO e FALSO.

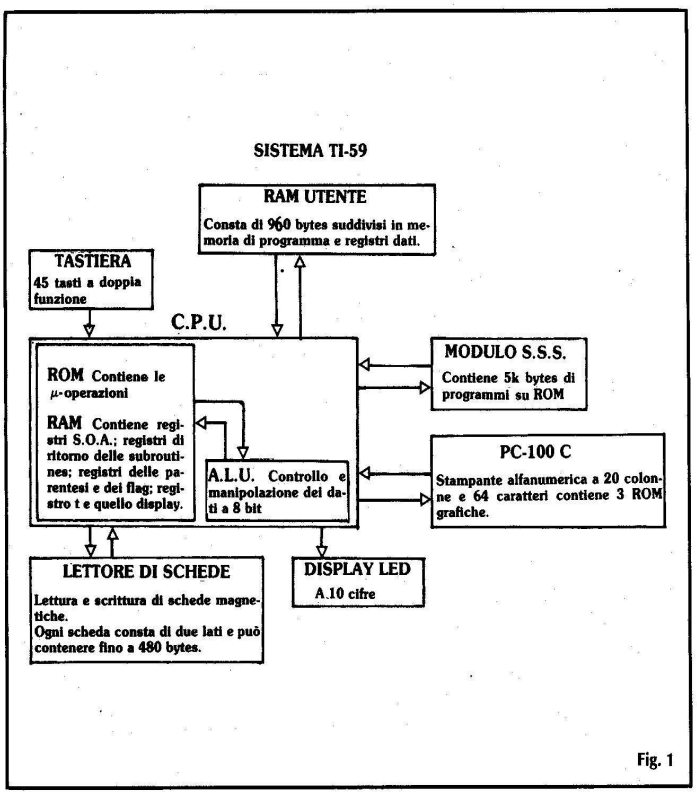


Fig. 1

- *La RAM utente*. Qui vengono memorizzati i dati introdotti come programma e come registri di memoria. Il tutto avviene su quattro RAM ognuna di 240 locazioni, cosicché in tutto vi sono 960 locazioni divisibili tramite la funzione Op 17 in area di programma o area di memoria. Nei paragrafi successivi si vedrà come sfruttare questo fatto.
- *Il display.* E' a led con 12 posizioni che vengono sfruttate diversamente a seconda del "modo" in cui opera (normale, esponenziale, fissaggio dei decimali, LRN-mode, elaborazione). PC-100C riconosce il "modo" del display: infatti facendo l'hard-copy con PRINT si vede che la stampa avviene in colonne diverse; inoltre si può capire anche come con OP 06 si possono usare solo i 4 caratteri sulla destra: il quinto (16° colonna) è a dibito alla stampa di "?" in caso di errore e quindi di display lampeggiante.
- *Le altre periferiche.* Sono la tastiera, la stampante, il lettore/registratore di schede e il modulo Solid State intercambiabile. Su queste non vi è niente da aggiungere a quanto afferma il manuale se non che la TI-59 riconosce solo se PC-100C è o non è attaccata ad essa (non c'è riempimento dei buffer di stampa) ma non riconosce se la stampante attaccata è accesa o spenta.

Le istruzioni della TI-59

Le istruzioni della TI-59 in LRN mode sono espresse da un codice a due cifre, cosicché le istruzioni sono 100 (da 00 a 99). Evidentemente "istruzione" in questa sede è il singolo byte senza gli eventuali suffissi, ad es. STO XY e STO PQ sono "istruzioni uguali" nel senso che entrambe eseguono un caricamento in memoria.

Secondo la regola della matrice, in generale si riesce a ricavare, senza imparare niente a memoria, il codice di quasi tutte le istruzioni. Si è detto "quasi", poiché vi sono eccezioni a tale regola: i codici composti ottenuti contraendo alcune istruzioni con *Ind* che, detto tra noi, risultano molto utili permettendo il risparmio di un buon numero di bytes. C'è in più da considerare le "funzioni di editing" che non sono introducibili in un programma in quanto di esse ci si serve appunto per la correzione di programmi stessi.

Di tutto questo tiene conto la TI quando nel manuale a pag. V-50 fornisce l'elenco dei codici in ordine numerico: è evidente la mancanza di alcuni di questi. Questo fatto ha dato l'avvio alla caccia dei codici "fantasma", è cioè l'inizio della p.s.

Con la tecnica del "cancellatore di byte" si può ottenere la sequenza richiesta: 21, 26, 31, 41, 51, 56, 82 che listata su stampante dà luogo al listato n°1.

```
000   21 2ND
001   26 2ND
002   31 LRN
003   41 SST
004   46 INS
005   51 BST
006   56 DEL
007   82 HIR
```

Listato n. 1

Ora che l'elenco delle istruzioni è finalmente completo (si noti l'estensione della regola della matrice anche alle funzioni di editing) si può dare la tabella completa dei codici ove il singolo quadrato va interpretato come nello schema a corredo (fig. 2).

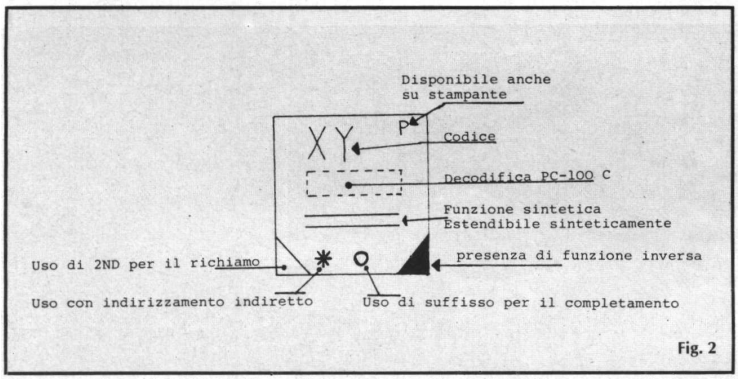


Fig. 2

Il tutto è visibile nella fig. 3

Fig. 3

00 0	01 1	02 2	03 3	04 4	05 5	06 6	07 7	08 8	09 9
10 E'	11 A	12 B	13 C	14 D	15 E	16 A'	17 B'	18 C'	19 D'
20 CLR	21 2ND	22 INV	23 LNX	24 CE	25 CLR	26 2ND	27 INV	28 LOG	29 CP
30 TAN	31 LRN	32 X⇌T	33 x^2	34 $\sqrt{x}$	35 1/X	36 PGM o	37 P→R	38 SIN	39 COS
40 IND o	41 SST	42 STO o	43 RCL o	44 SUM o	45 Y^x	46 INS	47 CMS	48 EXC o	49 PRD o
50 |x|	51 BST	52 EE	53 (	54)	55 ÷	56 DEL	57 ENG	58 FIX ✱ o	59 INT
60 DEG	61 GTO o	62 PG✱ o	63 EX✱ o	64 PD✱	65 X	66 PAU	67 EQ ✱ o	68 NOP	69 OP o
70 RAD	71 SBR ✱ o	72 ST✱ o	73 RC✱ o	74 SM✱ o	75 −	76 LBL o	77 GE ✱ o	78 $\Sigma+$	79 $\bar{x}$
80 GRD	81 RST	82 HIR ✱ o	83 GO✱ o	84 OP✱ o	85 +	86 STF ✱ o	87 IFF ✱ o	88 DMS	89 π
90 LST	91 R/S	92 RTN	93 .	94 +/−	95 =	96 WRT	97 DSZ ✱ o	98 P ADV	99 P PRT

D'ora in poi per funzione sintetica si intenderà un'istruzione non impostabile direttamente da tastiera e ottenibile solo con tecniche particolari; sarà compito della p.s. usare queste f.s. per ottenere nuove possibilità di programmazione.

Ancora sulla RAM utente.

Si è già detto della suddivisione in area dati/area programma della RAM; spingendosi un po' più a fondo ci si può chiedere in che rapporto stiano queste due zone.
E' facile rispondere usando la funzione Op 17: infatti 6 Op 17 dà 479.59 mentre 7 Op 17 dà 559.49 ed è evidente che "perdendo" 10 registri dati si sono "guadagnati" 80 passi di programma; poiché un passo corrisponde ad un byte allora un registro dati sarà composto da 8 bytes.
Risulta perciò possibile fissare una corrispondenza quasi biunivoca tra passi e registri, come mostra il manuale: ma poiché non esistono indirizzamenti da tre cifre, non possono essere ottenuti in alcun modo i registri 100-119; questo spiega la ripartizione massima della 59 e cioè la 10 Op 17 (159.899).

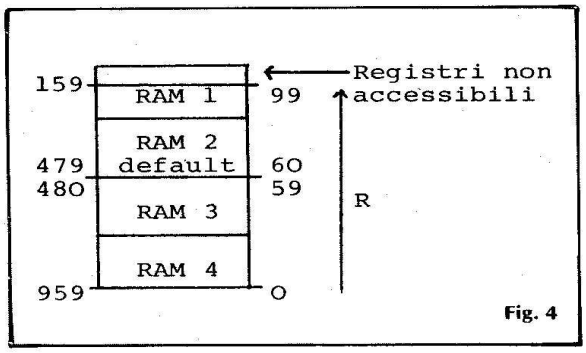
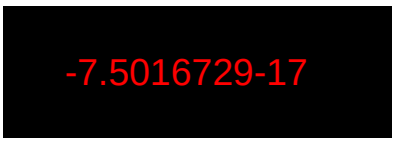


Fig. 4

Ci si può chiedere cosa capita ai registri "nascosti" cioè, se si è in default, ai registri da 60 in su. Qui ci viene in aiuto la corrispondenza di cui sopra secondo la quale è evidente che questi registri "non esistono", o meglio sono a disposizione SOLO come passi di programma (infatti il loro richiamo provoca errore); stesso discorso si può fare per le locazioni oltre la 479 usabili solo come registri dati. Ma noi siamo testardi e scriviamo dal passo 160 in su il seguente programma (in default):

LBL A STO 00 CP x=t <175> RCL 00 x Dsz 00 <168> 1 = R/S

cioè un programmino facile facile per il calcolo del fattoriale.
Facciamo 10 Op 17 e apparirà 159.99 sul display, richiamiamo il registro 99 e, sorpresa, (o no?) il display mostra



Cosa rappresenta questo numero?

Memorizzazione dei codici

Consideriamo i vari codici del programma suddetto e scriviamoli otto a otto:

```
76 11 42 00 29 67 01 75
43 00 65 97 00 01 68 01
95 91 00 00 00 00 00 00
```

Chiameremo l'insieme di 8 bytes "maxibyte"; perciò il nostro programma è composto da 3 maxibyte: allora riscriviamo ogni maxibyte in senso inverso

```
75 01 67 29 00 42 11 76 (M1)
01 68 01 00 97 65 00 43 (M2)
00 00 00 00 00 00 91 95 (M3)
```

e confrontiamoli rispettivamente con le memoria 99, 98 e 97:

```
 -7.5016729-17   (R99)
-1680.100977     (R98)
         9-31    (R97)
```

poiché è noto che ad ogni registro dati corrispondono 8 byte è evidente che ad ogni maxibyte corrisponde un singolo registro:

M1 ↔R99
M2 ↔R98
M3 ↔R97

Ma come corrispondono? La risposta guardando M1 e R99 è semplice: basta prendere i primi otto numeri 75016729, porre la virgola tra il primo ed il secondo

7.5016729

e considerare il 14°e 15°numero (17) come esponen ziale :

75016729 17

dunque, tranne i segni, c'è corrispondenza perfetta.
Mancano pure delle cifre, e precisamente la 9°, 10° , 11°, 12°, 13°, 16° (e cioè 0, 0, 4,2,1,6): in que sto caso è il display che non le mostra poiché può contenere solo 8 +2 cifre in notazione esponenziale.
Generalmente è la tredicesima cifra resta non visibile (per noi e non per la TI-59 ovviamente) e pure la sedicesima: anzi questa determina i segni da porre nel numero in questione secondo la seguente regola:

se la sedicesima cifra è 0 o 1 i segni sono + per la mantissa e + per l'esponenziale; se è 2 o 3 i segni sono – e +; se è 4 o 5 i segni sono + e -; se è 6 o 7 i segni sono – e – (come nel nostro caso); se infine c'è 8 o 9 il registro corrispondente al maxibyte richiesto dà overflow.

TABELLA NR. 1

16a cifra	segno mantissa	segno esponente
0 / 1	+	+
2 / 3	–	+
4 / 5	+	–
6 / 7	–	–
8 / 9	" OVERFLOW "	

Tab. 1

Si propone una tabella riassuntiva (tab. 1) che controlliamo su M2:

016810097650043

cioè

-0.1680100 +4

che la TI-59 automaticamente trasforma in

-1680.100977

```
000  76 LBL
001  11  A
002  65  ×
003  08  8
004  94 +/-
005  85  +
006  09  9
007  05  5
008  09  9
009  85  +
010  53  (
011  24 CE
012  75  -
013  07  7
014  54  )
015  55  ÷
016  01  1
017  00  0
018  00  0
019  00  0
020  95  =
021  58 FIX
022  03  03
023  91 R/S
024  76 LBL
025  12  B
026  75  -
027  09  9
028  05  5
029  09  9
030  95  =
031  55  ÷
032  08  8
033  94 +/-
034  95  =
035  59 INT
036  91 R/S
```

list. 8

Il "7" finale è in realtà il "65" arrotondato per necessità del visualizzatore: sarà visibile facendo INV Int su R98

-0.1009765

Idem si farà per ciò che riguarda M3:

0000000000009195

che diventerebbe

0.0000000 -19

che la TI-59 ancora una volta trasforma in

9 -31

cioè proprio R97. In questo caso il 9 che in condizioni normali non sarebbe stato visibile qui lo è essendo la prima cifra diversa da zero incontrata nel maxibyte. Sembra evidente che la TI-59 disporrà di una apposita rappresentazione interna per distinguere i numeri non

visibili (cifre di scorta) e i segni (distinguere cioè se un + significa avere come sedicesima cifra un 4 o un 5) e per la condizione di overflow.

Come supporto finale al paragrafo si riporta un programma che consente di individuare assegnato un registro il maxibyte corrispondente (Lbl A) oppure dato il passo consente di individuare il registro di appartenenza (Lbl B). A tale proposito vedere il listato n°8.
Ad esempio, al registro 45 corrisponde il maxibyte individuato dai passi tra 592 e 599 (formato del display 599.592).

In conclusione è utile insistere sul fatto che i passi tra 000 e 159 non possono essere richiamati come registri per l'inesistenza di registri con indirizzamento a tre cifre.
Si riporta pure un programma di esempio («Sistemi non lineari a 2 equazioni in 2 incognite» realizzato dall'autore) e la sua codifica in registri (listati n°2 e n°3).

240	76	LBL	290	55	÷	340	43	RCL	390	54	)
241	11	A	291	43	RCL	341	04	04	391	42	STO
242	42	STO	292	04	04	342	95	=	392	19	19
243	01	01	293	95	=	343	42	STO	393	85	+
244	91	R/S	294	42	STO	344	22	22	394	43	RCL
245	76	LBL	295	20	20	345	53	(	395	01	01
246	12	B	296	53	(	346	43	RCL	396	95	=
247	42	STO	297	43	RCL	347	02	02	397	42	STO
248	02	02	298	02	02	348	85	+	398	01	01
249	91	R/S	299	85	+	349	43	RCL	399	43	RCL
250	76	LBL	300	43	RCL	350	04	04	400	05	05
251	13	C	301	04	04	351	54	)	401	65	×
252	42	STO	302	54	)	352	42	STO	402	43	RCL
253	03	03	303	42	STO	353	11	11	403	22	22
254	93	.	304	11	11	354	17	B'	404	75	-
255	00	0	305	16	A'	355	75	-	405	43	RCL
256	00	0	306	75	-	356	43	RCL	406	06	06
257	00	0	307	43	RCL	357	06	06	407	65	×
258	00	0	308	05	05	358	95	=	408	43	RCL
259	01	1	309	95	=	359	55	÷	409	20	20
260	42	STO	310	55	÷	360	43	RCL	410	95	=
261	04	04	311	43	RCL	361	04	04	411	55	÷
262	43	RCL	312	04	04	362	95	=	412	43	RCL
263	03	03	313	95	=	363	42	STO	413	19	19
264	91	R/S	314	42	STO	364	23	23	414	85	+
265	76	LBL	315	21	21	365	43	RCL	415	43	RCL
266	14	D	316	53	(	366	06	06	416	02	02
267	53	(	317	43	RCL	367	65	×	417	95	=
268	43	RCL	318	01	01	368	43	RCL	418	42	STO
269	01	01	319	85	+	369	21	21	419	02	02
270	85	+	320	43	RCL	370	75	-	420	43	RCL
271	43	RCL	321	04	04	371	43	RCL	421	03	03
272	04	04	322	54	)	372	05	05	422	32	X↕T
273	54	)	323	42	STO	373	65	×	423	43	RCL
274	42	STO	324	10	10	374	43	RCL	424	05	05
275	10	10	325	43	RCL	375	23	23	425	50	I×I
276	43	RCL	326	02	02	376	95	=	426	77	GE
277	02	02	327	42	STO	377	55	÷	427	14	D
278	42	STO	328	11	11	378	53	(	428	43	RCL
279	11	11	329	17	B'	379	43	RCL	429	06	06
280	16	A'	330	75	-	380	20	20	430	50	I×I
281	75	-	331	43	RCL	381	65	×	431	77	GE
282	43	RCL	332	01	01	382	43	RCL	432	14	D
283	01	01	333	42	STO	383	23	23	433	43	RCL
284	42	STO	334	10	10	384	75	-	434	01	01
285	10	10	335	17	B'	385	43	RCL	435	99	PRT
286	16	A'	336	42	STO	386	22	22	436	43	RCL
287	42	STO	337	06	06	387	65	×	437	02	02
288	05	05	338	95	=	388	43	RCL	438	99	PRT
289	95	=	339	55	÷	389	21	21	439	91	R/S

list. 2

list. 3

9.1990244-31	65
7.750064315	66
-4.3320343 50	67
43851.94356	68
6.5064375-50	69
9.9999999 99?	70
4.2542144-37	71
2.343652-59	72
-2.3436505 14	73
-6.5064323 44	74
-5.5950644 14	75
-5.4044385 32	76
-4.2950444 64	77
1.7104201 71	78
-4.202431 44	79
8.5014353-50	80
4.3559505 61	81
-4.2540444 35	82
2.0429504-50	83
-4.2161042-51	84
1.1420243-40	85
4.3850144 69	86
0.343044201	87
-930342137.7	88
-4.2127691-17	89

Per utilizzare il programma basta caricare le due equazioni f(x,y)=0 e g(x,y)=0 come Lbl A' e Lbl B' dal passo 000 in poi (terminandole al solito con INV SBR) utilizzando RCL 10 come x e RCL 11 come y. Poiché il programma utilizza il metodo iterativo di Newton-Fourier, bisogna impostare il punto di innesco $P(x_0,y_0)$ come x_0 A, y_0 B, l'errore ε richiesto C e far partire l'elaborazione con D. La soluzione trovata sarà stampata.

Tecniche di p.s.

Prima di iniziare a parlare delle semplicissime tecniche di p.s. occorre spendere due parola in più sul S.O. e la funzione in LRN mode.
Un gruppo di istruzioni ha bisogno dell'indirizzamento per essere completate; indirizzamento che può essere a 2 o 3 byte a seconda del tipo – vedi tabella n° 2; cosicché il S.O. accetta QUALSIASI byte do po l'istruzione segnalando errore solo nel modo di elaborazione; ad es. uno STO 73 in default, un GTO x^2 se non esiste la label x^2 e così via.

Tab. 2

TABELLA NR. 2 - LUNGHEZZA DI SUFFISSI

funzione		numero di byte
	STO	1
	RCL	1
• inverse	SUM	1
• inverse	PRD	1
	SBR	1 (indir. per label)/2 (indir.assoluto)
	GTO	1 (indir. per label)/2 (indir.assoluto)
	FIX	1 (numero tra 0 e 9)
• inverse	STF	1 (numero tra 0 e 9)
• inverse	IFF	1 (numero tra 0 e 9)
	DSZ	2 (reg.+label)/3 (reg.+ind. assoluto)
	PGM/HIR	1
	GE/EQ	1 (indir. per label)/2 (indir. assoluto)

N.B. Si esclude il caso di indirizzamenti indiretti.

Questo fatto ci permette appunto l'introduzione di f.s. usando gli indirizzamenti di dette istruzioni (di solito STO e RCL) e poi cancellare con Del il codice di troppo: è la cosiddetta tecnica del "cancellatore di byte": è, evidentemente, una tecnica artificiale non prevista dal manuale.
Come si vedrà più avanti l'istruzione HIR richiede pure un suffisso ad 1 byte e questo è il modo per ottenerlo: sia da codificare HIR 38; si scrivono in LRN mode le seguenti istruzioni:

STO 82 STO 38 aventi codice
42 82 42 38

si ritorna indietro con BST BST si fa Del, poi BST BST, ancora Del, quindi SST finché non ci si riporta alla locazione che segue quella contenete il codice 38.
Esiste pure un'altra tecnica più "furba": quella della *sostituzione*.
Notando infatti che la maggior parte delle volte l'indirizzo da dare coincide con un codice funzione, si può immettere direttamente il codice funzione al posto dell'indirizzo senza che succeda nulla: in un certo senso si "truffa" il S.O.
Nel caso precedente 38 è Sin, perciò si farà

STO 82 Sin XX aventi codice
42 82 38 XX

e basterà fare BST BST BST Del e riportarsi in XX con SST.

Tab. 3

TABELLA NR. 3 - SUFFISSI DI HIR

X	OPERAZIONE
0	STO
1	RCL
3	SUM
4	PRD
5	INV SUM
6,7,8,9	INV PRD

Il fatto notevole è che si può usare questa tecnica per snellire l'inserimento di funzioni normali; per immettere STO 25 si può premere STO CLR dato che, avendo bisogno STO di un suffisso, il CLR verrà tratto appunto come il 25 richiesto.
Unico requisito, ovvio d'altronde, è la conoscenza dei codici, o almeno della maggior parte di essi.

Il codice 82

La scoperta di "cosa facesse" il codice 82 ha richiesto molta fantasia, anche per le sue caratteristiche peculiari.
Innanzitutto il nome HIR deriva dalla decodifica di PC-100C e si può vedere che richiede un indirizzamento ad un byte,

listando ad esempio uno dei programmi del modulo MU che usano questa funzione.
Prendendo il listato di MU-06 (quello del programma “SORTING”) si può notare che l’HIR “sembra usato” come registro e si possono compiere le solite operazioni in memoria, anche se in modo particolare.
E’ così infatti: si può notare che il suffisso XY indica sia l’operazione sia l’operazione da fare (X) che il registro interessato (Y); senza dilungarci diremo che (quante prove!) Y va da 1 a 8 e che X va da 0 a 9 in accordo con la tabella 3.

E’ facile poi vedere che, in effetti, il fatto che gli y possibili siano 8 fa pensare che HIR coincida proprio con i registri del S.O.. Infatti in HIR si depositano i valori intermedi nel caso di operazioni in sospeso e di apertura di parentesi. Inoltre, come specifica il manuale, negli ultimi quattro HIR si depositano i codici alfanumerici corrispondenti ai quattro Op 01.. Op 04. Si comprende allora come sia possibile manipolare questi codici direttamente in ambiente HIR. Ma c’è di più: sebbene il manuale taccia, pure le funzioni D.MS, P/R (con le inverse) e le funzioni statistiche appoggiano i risultati intermedi in HIR.

Un po’ di HIR

Vedremo qui, gli usi possibili di HIR.

- *Dove usare HIR?* Da quanto detto finora sulle tecniche di p.s. HIR risulta usabile comodamente in un programma; il manuale nell’appendice D (quella dedicata alla correzione dei programmi) afferma che un programma può essere eseguito manualmente mediante l’uso di SST a partire da qualsiasi locazione: questo fatto ci è utile poiché una qualsiasi istruzione può essere ottenuta da tastiera tramite un programma usando SST. Se si inserisce nella locazione 043 il codice 82, per eseguire da tastiera HIR 15 si farà

  GTO 043 SST (°) 15

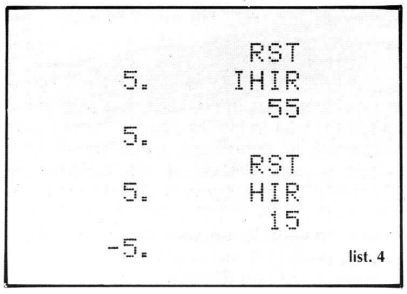
  In (°) il S.O. ha ricevuto l’ordine di eseguire un HIR ed attende il byte successivo, dopo la pressione del 5 sul display apparirà il contenuto del 5° registro del S.O. Tutto questo è fattibile con un’etichetta, ma in generale la situazione più comoda si ha ponendo 82 nella locazione 000 e l’esempio precedente diventa

  RST SST 15

  risparmiando la pressione di due tasti (ma attenti ai flag!). Da notare pure che suffissi indifferenti per HIR sono quelli per cui Y=0, Y=2 e X=2; da qui deriva l’impossibilità di eseguire l’indirizzamento indiretto da programma (HIR 40 equivale ad un Nop), cosa possibile invece da tastiera: se si pone infatti 15 in R23 e 82 in 000 facendo

  RST SST Ind 23

  si otterrà sul display sempre il contenuto di H5. Inoltre è possibile l’uso, T&P, di HIR preceduta da INV per ciò che riguarda le operazioni aritmetiche in memoria: l’uso di INV HIR 3Y anziché HIR 5Y e INV HIR 4Y anziché HIR 6Y farà comparire sulla printer, in TRACE, il simbolo IHIR come si può vedere da listato 4. Comunque in TRACE, PC-100C pone sempre una I dinanzi ad una funzione inversa anche se questa non ha alcun senso: è possibile avere cose del tipo I+ e simili.



list. 4

- *Uso di HIR come registri.* L’utilizzo è del tutto immediato dopo aver imparato la tabella 3: 25 HIR 31 aggiunge 25 al 3° registro del S.O., HIR 07 richiama il contenuto del 7° ecc. Esistono purtroppo, alcune situazioni non del tutto gradite: mentre le operazioni di richiamo e memorizzazione sono del tutto analoghe a STO e RCL, quelle aritmetiche in memoria sono “strane” nel senso che se il numero

oggetto dell'operazione è in valore assoluto minore di 1 viene modificato; nessun problema, invece, se il numero ha modulo maggiore di 1. Se ad esempio si vuole sommare 0.2 in H1 facendo HIR 31 si sommerà in realtà 20, T&P, se si vuole sommare 0.004 in realtà si somma 4000, ecc.
L'inghippo è presto trovato e la regola generale è la seguente:

se si compiono operazioni aritmetiche in memoria con numeri di modulo minore di 1 questi "vengono usati" moltiplicati per 10^{2n+2} ove n è il numero di zeri della parte decimale.

Nel caso di 0.004 (n=2) si trova infatti 0.004 * 10^6=4000. Tutto questo non serve se il numero è impostato in notazione esponenziale.
Occorre prestare molta attenzione all'uso di HIR in un programma: non bisognerà usare i primi 2/3 registri H in quanto destinati probabilmente alle operazioni in sospeso, perciò resteranno 5/6 registri H usabili come registri normali (senza stampa).

- *Uso di HIR con i codici alfanumerici.* Come già detto i codici alfanumerici introducibili con gli Op 01..Op 04 sono contenuti in H5, H6, H7, H8 ("buffer di stampa"); si tratta di vedere come ciò avvenga.
Introduciamo 1314151617 (ABCDE) nell'ultimo buffer con Op 04 e, usando HIR da tastiera come precedentemente descritto, si faccia HIR 18 e sul display apparirà

.0013141516

cioè il codice moltiplicato per 10^{-12}. Ma non c'è da preoccuparsi: non si è persa l'ultima lettera (E → 17) poiché i registri della TI sono a 12 cifre, mentre il display ne visualizza solo 10. Premendo Op 05 si estrarranno i singoli caratteri dalla ROM grafica di PC-100C e verrà stampato ABCDE nell'ultimo quarto di carta termica.
Questo fatto suggerisce di saltare l'impostazione dei codici tramite Op e di usare HIR direttamente introducendo 1314151617 x 1 EE +/- in H8 e premendo Op 05; ma così facendo non si ottiene l'effetto desiderato ed esce una strana scritta (U*%^{2}b).
Infatti per uno strano scherzo dei progettisti la stampa senza Op avviene solo aggiungendo un 1 nel buffer interessato, in questo caso H8: 1 HIR 38, solo così si sistema tutto.
L'utilità delle HIR si esplica ancor di più nello scorrimento del buffer a destra o sinistra o nella sostituzione di qualche carattere. Se per es. da ABCDE volessi ottenere solo BCDEb (scorrimento a sinistra), supposto di continuare l'esempio precedente, basterebbe fare:

1 HIR 58 100 HIR 48 1 HIR 38 Op 05

per ottenere l'effetto voluto. Nello stesso modo per eliminare E; si noti che non ha importanza che le prime due cifre dopo l'uno iniziale siano diverse da zero; se in H8 c'è

1.1314151617

verrà stampato ancora BCDEb.
Per cambiare carattere, ad es. da A in P basterà fare la differenza dei codici (33-13=20) e aggiungerla tenendo presente che:

se si cambia rispettivamente il 1°, il 2°, il 3°, i l 4° o il 5° carattere si moltiplica la differenza nell'ordine per 1EE 4+/-, 1EE 6+/-,1EE 8+/-,1EE 10+/-, 1EE 12+/-; in generale se il codice da cambiare è in posizione n si moltiplica la differenza per $10^{-(2n+2)}$.

Nel nostro caso si farà

<u>1 HIR 58</u> 1 + 20 EE 4 +/- = HIR 38 Op 05

Volendo è possibile saltare il primo termine, poiché la somma non interferisce con la parte intera del registro H interessato.
Analoghe considerazioni si possono fare volendo far scorrere o sostituire due o più caratteri. Si noti, comunque, che tali operazioni sono possibili pure senza uso di p.s., ma risultano più costose in termini di tempo. Infatti per sostituire A con P, bisogna appoggiare il codice 13143151617 su un registro

qualsiasi, poi aggiungere 2000000000 a questo registro e infine immetterlo nel buffer desiderato oppure riscrivere di nuovo il codice alfanumerico.

- *HIR e funzioni varie*. Come già accennato alcune funzioni si appoggiano, a causa della loro complessità, su alcuni registri H: si tratta delle funzioni D.MS, P/R (e relative inverse) e tutte le funzioni statistiche (Σ+, $\overline{x}$, Op 11, Op 12, Op 13 e le loro inverse, se esistono). Più in dettaglio D.MS si appoggia a H1, H2, H8 così pure l'inversa; P/R ad H7, H8 i quali contengono rispettivamente sin α, R ed α; INV P/R invece usa H1, H7, H8 che contengono rispettivamente y, x e y.
  Ad esempio sia da convertire 45.33333333 con D.MS → 45.55925925 (cioè da sessagesimale a sessadecimale), si trova in H1 1640.133333; in H2 0.33333333 ed in H8 33.333333.
  Se si vuole trasformare in coordinate cartesiane R = 5 e α=18° si usa P/R (x=4.755282581 e y =1.545084972) e si trova 0.3090169944 (sin 18°) in H1, 5 (R) in H7 e 18° (α) in H8. Per ciò che riguarda le funzioni statistiche l'occupazione degli H è molto più complicata ed è in rapporto col tipo di operazione eseguita; in generale si può dire che vengono occupati H1, H2, H3, H4, H7, H8; si può vedere questo su un esempio preso dal manuale a pag. V-36 e del quale il tabulato mostra l'occupazione degli HIR man mano che si procede nell'uso delle funzioni statistiche (vedi listato n°5).

Listato N. 5 - Occupazione degli hir con funzioni statistiche - a: impostazione dati (+) - b: media - c: varianza e deviazione standard peso N (Op 11) - d: varianza e deviazione standard peso N-1 (INV x̄) - e: regressione lineare (Op 12) - f: correlazione (Op 13).

```
list. 5
                0.                  1454.833334
                0.                    12952801.
                0.                           0.
                0.                           0.
                0.                           0.
                0.                           0.
            101.1                        101.1
          60560.5   a                        5.   d

             3599.                  -891.0382576
                0.                      60185.2
                0.                    360960.64
                0.                           0.
                0.                           0.
                0.                           0.
            101.1                        101.1
          60560.5   b                        5.   e

         -10026.68444                187.5333334
            60185.2                  25.09333334
                0.                     2160255.
                0.                    12952801.
                0.                           0.
                0.                           0.
            101.1                        101.1
          60560.5   c                        5.   f
```

HIR e fissaggio dei decimali

Un ultimo aspetto sul quale ci si sofferma è il rapporto tra il formato di stampa e l'istruzione FIX, che riguarda pure, com'è evidente, la f.s. HIR.
Se si fissano due decimali e si imposta 13 in Op 04 con Op 05 si ottiene la stampa

bbbAb (1) anziché
bbbbA

come ci si sarebbe potuto attendere. Da ciò si deduce che il fissaggio dei decimali fa scorrere a sinistra il codice tenendo conto degli zeri ad esso dovuti. Infatti per avere (1) si sarebbe dovuto impostare 1300 in Op 04 e invece si è impostato 13.00 da cui risulta che Op 05 non ha tenuto conto della virgola ed ha trattato 13.00 come l'intero 1300.
In base a queste considerazioni si possono dedurre alcune conseguenze:

1) Se dato FIX N con N pari, c'è uno scorrimento a sinistra di N zeri se il display può contenerli tutti; altrimenti c'è uno scorrimento di N – 2P zeri ove P è il numero dei codici impostati. Ad esempio, in un programma si voglia impostare Abbbb: si farà

   FIX 8 13 Op 04 Op 05 anziché
   13 00 00 00 00 Op 04 Op 05

   con un notevole risparmio di passi.

2) Se invece N è dispari c'è uno scorrimento errato dato che vengono aggiunti N zeri (dispari) mentre un codice è sempre formato da due numeri (pari): in FIX 3 si voglia stampare la stringa "π+e" di codice 534754, allora con 534754.000 Op 04 Op 05 si ottiene la stampa "4QΔ.b" che è stata così ottenuta:

```
05|34|73|40|00
 4|  Q|  Δ|  .|  b
```

Occorre notare che la memorizzazione della stringa in HIR avviene con il solito modo, tenendo conto degli zeri dovuti al fissaggio; "π+e" in FIX 3 è memorizzato x come

-0.000534754000

visibile dopo aver tolto il fissaggio.
La presenza del segno – è visibile anche in altre situazioni e precisamente (si suppone sempre di essere in Op 04):

- se la stringa è di un carattere avremo overflow in H8 con FIX 8; una memorizzazione negativa con FIX 7, FIX 6, FIX3 e FIX 2;
- se la stringa è due caratteri c'è il - con FIX 8, FIX 7, FIX 6, FIX3, FIX 2;
- se la stringa è di tre caratteri c'è il – solo con FIX3 e FIX 2;
- se la stringa è di quattro caratteri c'è il – con tutti i FIX eccetto FIX 1 e FIX 0;
- se la stringa è di cinque caratteri per ogni FIX H8 è positivo.

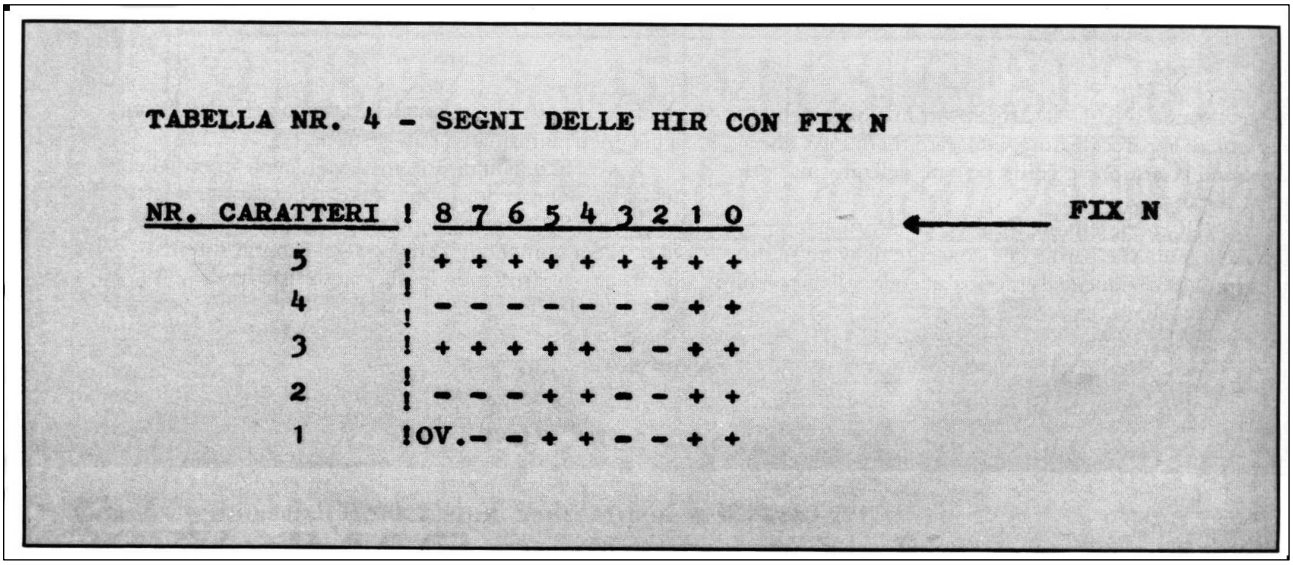


TABELLA NR. 4 - SEGNI DELLE HIR CON FIX N

NR. CARATTERI	8	7	6	5	4	3	2	1	0	← FIX N
5	+	+	+	+	+	+	+	+	+	
4	-	-	-	-	-	-	-	+	+	
3	+	+	+	+	+	-	-	+	+	
2	-	-	-	+	+	-	-	+	+	
1	OV.-		-	+	+	-	-	+	+	

Giocando con i bytes

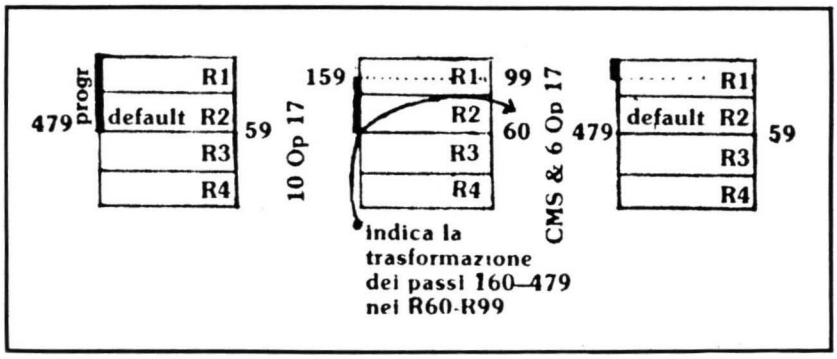
Portiamo a fondo il discorso già iniziato nei paragrafi precedenti dotando il software della TI-59 di una routine di autocancellazione del programma *da programma*.
Infatti, com'è noto, i tasti di cancellazione della TI sono i seguenti:

- CLR – cancella il visualizzatore (T&P)
- CMs – cancella i registri dati (T&P)
- RST – cancella flag e registri di ritorno delle subroutines (T&P)
- Op 05 – cancella i buffer di stampa (e i registri 5,6,78 del S.O.) (T&P)
- CP - cancella il registro T (T&P), il programma *solo da tastiera*.

Si vede perciò che CP è il punto debole poiché non può cancellare il programma in fase di elaborazione (ad es. a scopo di protezione del software).
Il problema si può superare ricorrendo alla già ricordata corrispondenza biunivoca tra passi e registri; infatti cambiando ripartizione si trasformano i passi di programma in registri che si possono cancellare tramite CMs, usabile anche in un programma; si scriverà perciò una certa routine, ad esempio:

Lbl A 10 Op 17 CMs 6 Op 17 R/S.

Questa routine posta *prima* del passo 160 azzera tutte le posizioni di programma dalla 160 in su e riporta in default (uso dinamico della memoria); dal punto di vista grafico la corrispondenza funziona così (supponendo di essere in default all'inizio):



Come si vede resta intatta la zona tra 000 e 159: l'eventualità di poterla cancellare si deve escludere non esistendo ripartizioni superiori a 159.99; è evidente che, si fosse disposto di un XY Op 17 che desse come ripartizione 0.119, il programma sarebbe caduto in errore appena effettuato l'XY Op 17 in quanto si sarebbe autoprivato dei passi indispensabili per eseguire l'istruzione seguente di cancellazione.

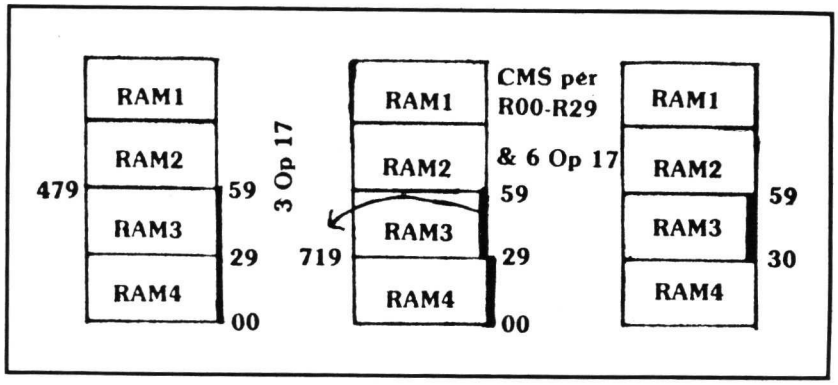
La routine data è ovviamente modificabile quanto si vuole; se in default si vogliono cancellare i passi 240-479 si farà

Lbl A 9 Op 17 CMs 6 Op 17 R/S

Quanto visto si inserisce comunque nel più vasto concetto di uso dinamico della memoria: si vogliano cancellare, ad esempio, solo i registri da 00 a 29, con situazione iniziale in default, si farà dunque:

Lbl A 3 Op 17 CMs 6 Op 17 R/S

E dal punto di vista grafico:



Si noti che in ogni caso cancellare aree dati che non iniziano da R00 risulta impossibile mediante l'uso dinamico della memoria e si deve ricorrere ad un ciclo.

Le funzioni di editing in un programma

L'uso delle funzioni di editing come f.s. presenta possibilità molto limitate. Infatti, al di fuori della funzione abituale, BST, Ins, Del e 2nd risultano agli "occhi" del S.O. come dei Nop; mentre LRN e SST a qualcosa possono servire: questo fatto si giustifica se si ricorda che da tastiera le suddette funzioni sono indifferenti per la RAM utente.

Per ciò che concerne LRN, inserendo il codice 31 in una qualsiasi parte del programma, purché non come suffisso, si verifica che il contatore di programma raggiunta quella locazione la esegue ed il calcolatore si pone automaticamente in LRN mode, uscendo perciò dal modo di elaborazione.

Se il pezzo di programma è ….. 85 38 95 66 31 XX …. Raggiunta la locazione XX c'è l'entrata in LRN mode e la visualizzazione sul display della locazione contenente XX; in un certo senso LRN è un surrogato di R/S.

Invece per SST c'è indifferenza per ogni istruzione che lo segue con la differenza che ….. SST R/S …. *non* blocca l'elaborazione e si può perciò usare in test che richiedano o non richiedano il blocco dell'elaborazione in seguito al verificarsi di una certa condizione.

Si veda il seguente problema elementare: bloccare l'elaborazione se R01 = 5 altrimenti visualizzare R00 in pausa e andare avanti.

Normalmente esistono due soluzioni

….. 5 x:t RCL 01 INV x=t <abc> R/S RCL 00 Pau … (con indirizzo assoluto)
⇑
abc

oppure

….. 5 x:t RCL 01 INV x=t A R/S Lbl A RCL 00 Pau … (con etichetta)

lunghi 12 e 13 passi rispettivamente.
Dal punto di vista sintetico si può fare:

….. 5 x:t RCL 01 x=t <abc> RCL 00 Pau SST R/S …
⇑
abc

di 12 passi, si noti la presenza di SST che "annulla" R/S.
Un esempio pratico ai applicazione che coinvolge le funzioni SST e LRN è il seguente.
Carichiamo il seguente programma a partire dalla locazione 000:

Lbl B GTO <008> Lbl A SST LRN ……R/S

lasciando a zero le locazioni dalla 009 alla 018.
Premendo A dopo un breve attimo il display mostrerà zero: l'istruzione SST ha fatto si che il programma ignorasse il LRN seguente. Premiamo ora B: vedremo apparire sul display

(siamo entrati in LRN mode). Impostiamo ora una sequenza di istruzioni (ad es. 3 + 2 = R/S); premiamo LRN e facciamo eseguire il programma dall'etichetta A.
Vedremo apparire sul display il risultato della sequenza precedentemente inserita: questa routine potrà quindi essere utile in tutti in quei casi in cui occorra inserire all'interno di un programma una sequenza di istruzioni (ad es. una funzione).

LBL & DSZ

Un altro aspetto della p.s. riguarda l'ampliamento delle possibilità di alcune istruzioni normali: Lbl e Dsz.
Per ciò che riguarda le etichette non è vero, come è già stato fatto notare, che le etichette usabili siano solo quelle indicate dal manuale a pag. V-43; si può vedere che tutte e cento le istruzioni della TI-59 possono essere usate come etichette, in quanto Op 08 riconosce qualsiasi codice dopo 76; ovviamente non tutte saranno operative:

- le etichette numeriche e quelle di editing + Ind non vengono considerate dal S.O. e dunque il loro uso è del tutto inutile;
- la f.s. HIR è usabile, ma come le funzioni composte (con Ind) non è richiamabile direttamente da tastiera;
- è possibile usare, T&P, le etichette R/S e Wrt (vedi Pgm 01 di molti moduli Solid State); in più 2nd CLR (codice 20) e 2nd INV (codice 27) sono usabili e la macchina le distingue da CLR e INV per la diversità dei codici.

Per introdurre LBL RC* cioè il codice 76 63 si può fare LBL RCL RCL Ind cioè 76 43 76 e poi cancellare il 43, oppure STO 76 STO 73 e cancellare i due 42.

Per quello che riguarda Dsz c'è da dire che qualsiasi registro (eccetto R40 che la macchina considera come Dsz Ind mancando una istruzione composta per Dsz Ind) è usabile con essa; contrariamente a ciò che afferma il manuale, il quale restringe i registri utili da 00 a 09: questo risultato si può ottenere con le due tecniche di p.s. introdotte.
Poiché anche STF usa solo i codici numerici da 0 a 9, si era tentati di estendere i risultati ottenuti con Dsz anche con questa funzione: ma si è visto in precedenza che esistono solo 10 registri di flag e questa limitazione hardware rende impossibile l'estensione; anzi si può vedere benissimo che la sequenza

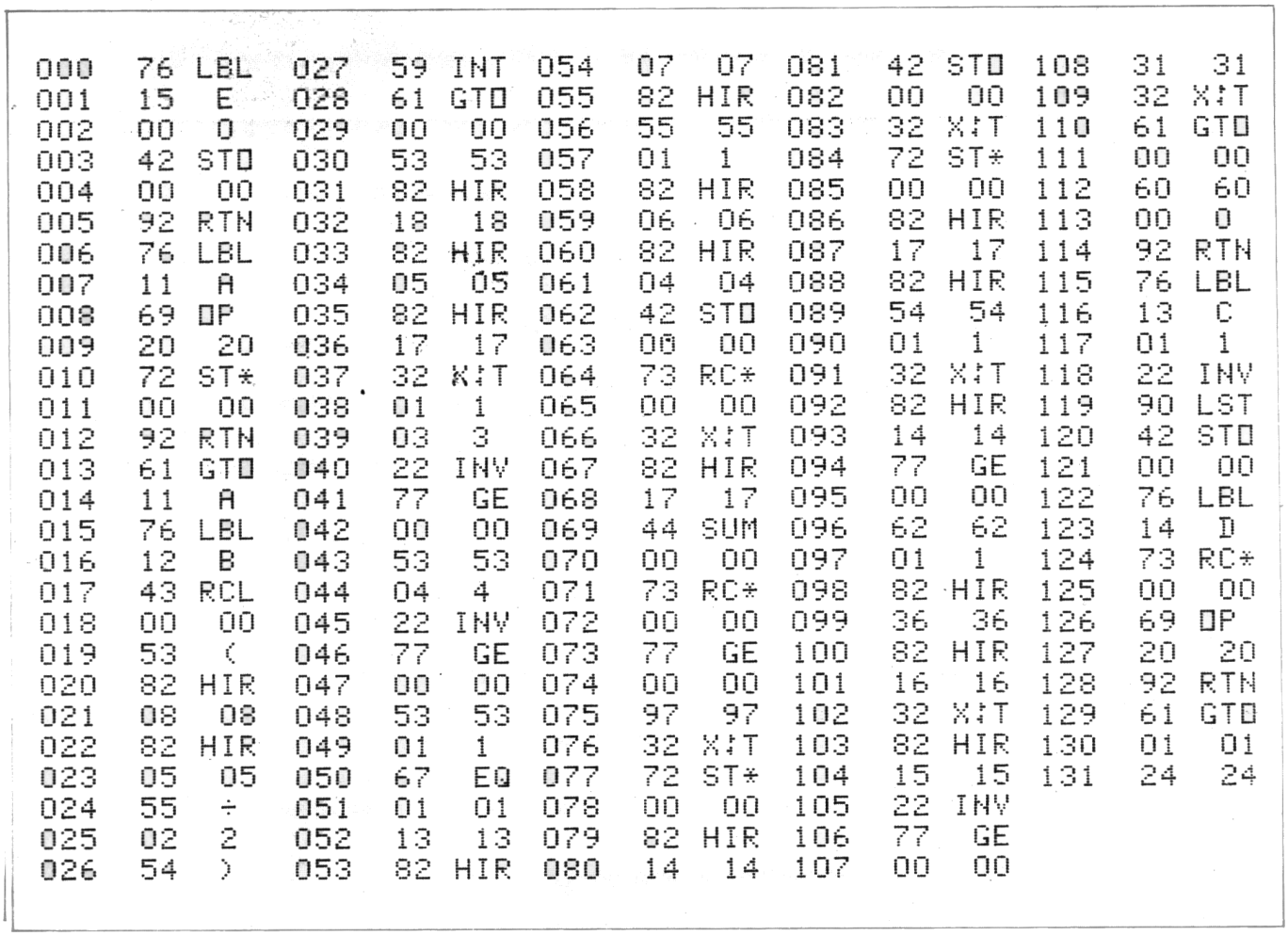
STF XY

attiva il flag Y ed X viene semplicemente ignorato. Idem per IFF e le relative inverse.

Come si vede le tecniche di p.s. danno molto spazio alle singole possibilità di inserimento e quindi all'autonomia del programmatore.

Conclusione

Come conclusione voglio riportare quello che considero come il capolavoro della p.s. e cioè il programma SORTING, 6° della biblioteca MATH/UTILITIES (vedi l istato 6).
Questo sorting, come dice il nome, ordina con il metodo di Shell un insieme di numeri (fino a 99) posti nella RAM utente usando come memoria di lavoro solo le HIR 4,5,6,7,8 oltre al registro 00.
Il suo uso è semplicissimo: si inizializza con E, si immettono con A tutti i numeri dell'insieme e si ordina con B. L'output può essere realizzato con C se di dispone di stampante o con D se si vogliono visualizzare gli elementi uno ad uno.

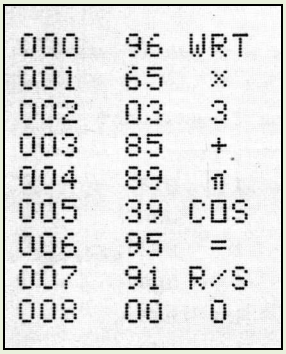


```
000  76 LBL  027  59 INT  054  07  07  081  42 STO  108  31  31
001  15  E   028  61 GTO  055  82 HIR  082  00  00  109  32 X:T
002  00  0   029  00  00  056  55  55  083  32 X:T  110  61 GTO
003  42 STO  030  53  53  057  01  1   084  72 ST*  111  00  00
004  00  00  031  82 HIR  058  82 HIR  085  00  00  112  60  60
005  92 RTN  032  18  18  059  06  06  086  82 HIR  113  00  0
006  76 LBL  033  82 HIR  060  82 HIR  087  17  17  114  92 RTN
007  11  A   034  05  05  061  04  04  088  82 HIR  115  76 LBL
008  69 DP   035  82 HIR  062  42 STO  089  54  54  116  13  C
009  20  20  036  17  17  063  00  00  090  01  1   117  01  1
010  72 ST*  037  32 X:T  064  73 RC*  091  32 X:T  118  22 INV
011  00  00  038  01  1   065  00  00  092  82 HIR  119  90 LST
012  92 RTN  039  03  3   066  32 X:T  093  14  14  120  42 STO
013  61 GTO  040  22 INV  067  82 HIR  094  77  GE  121  00  00
014  11  A   041  77  GE  068  17  17  095  00  00  122  76 LBL
015  76 LBL  042  00  00  069  44 SUM  096  62  62  123  14  D
016  12  B   043  53  53  070  00  00  097  01  1   124  73 RC*
017  43 RCL  044  04  4   071  73 RC*  098  82 HIR  125  00  00
018  00  00  045  22 INV  072  00  00  099  36  36  126  69 DP
019  53  (   046  77  GE  073  77  GE  100  82 HIR  127  20  20
020  82 HIR  047  00  00  074  00  00  101  16  16  128  92 RTN
021  08  08  048  53  53  075  97  97  102  32 X:T  129  61 GTO
022  82 HIR  049  01  1   076  32 X:T  103  82 HIR  130  01  01
023  05  05  050  67  EQ  077  72 ST*  104  15  15  131  24  24
024  55  ÷   051  01  01  078  00  00  105  22 INV
025  02  2   052  13  13  079  82 HIR  106  77  GE
026  54  )   053  82 HIR  080  14  14  107  00  00
```

TI 58C e 59

Grazie alla struttura pressoché uguale della TI-58C, quasi tutte le cose dette si trasportano facilmente al nostro caso:

1) La memoria costante della TI-58C non si estende alla memoria macchina, ma solo alla RAM utente.
2) Al contrario della TI-59 nella TI-58C c'è la perfetta corrispondenza tra passi di programma e registri dati, infatti si va da 0.59 (6 Op 17) a 479 (0 Op 17). Questo fatto può causare fastidi per la routine di autocancellazione del programma nel senso che non è possibile porre "6 Op 17" poiché la scomparsa dei passi di programma impedirà l'esecuzione delle istruzioni seguenti: meglio porre "5 Op 17" che cancellerà tutte le locazioni oltre il passo 079.
3) La perfetta corrispondenza tra passi e registri fa sì che sia possibile ottenere la codifica in maxibyte anche per le istruzioni poste all'inizio della memoria di programma.
4) Pur non essendo provvista di lettore/registratore di schede magnetiche la codifica della stampante fornisce sempre WRT per il codice 96, che d'altronde è impostabile direttamente: il S.O. della 58C lo considera equivalente ad un Int eccetto quando lo si usa come etichetta.



Conversioni e funzioni statistiche

L'appendice B del manuale in dotazione alle TI fornisce l'elenco delle condizioni di errore, cioè di quelle operazioni che effettuate fanno lampeggiare il display.
Tutte queste condizioni, meno una, non influenzano minimamente lo svolgimento dei calcoli nel senso che, pur lampeggiando il display, è possibile ottenere risultati corretti qualsiasi altra funzioni si usi; l'eccezione è costituita dal Pgm 01 del modulo MASTER e precisamente da quella parte di esso che contiene una lunga serie di routine etichettate da A a E e da A' a E'.
Ciascuna di queste routine ha la forma

Lbl <etichetta-definibile> Adv Prt Pgm Ind 00 <etichetta-definibile> Prt RTN

e quando si memorizza un numero in R00, si richiama Pgm 01e si preme uno qualsiasi dei tasti definibili della TI-59 , dopo aver eseguito Adv e Prt, elabora la corrispondente del Pgm Ind 00, stampa i risultati e ritorna.
Evidentemente si avrà un segnale di errore se il numero contenuto in R00 supera il numero massimo dei programmi contenuti nel modulo (nel nostro caso 25); in questo caso la macchina "impazzisce", non risponde più a certi comandi ed ha strane reazioni ad altri (vedi per esempio LRN e Op).
Esiste tuttavia una serie di istruzioni che si comporta in modo ancora più strano e precisamente le seguenti:

(A) P/R, INV P/R, D.MS, INV D.MS, Σ+, INV Σ+, $\overline{\mathbf{x}}$, INV $\overline{\mathbf{x}}$;

se si prova la sequenza

(1) (OFF/ON) *//serve per resettare la TI-59//* 99 STO 00 CLR Pgm 01 A $\overline{\mathbf{x}}$ LRN

si trova che, abbastanza misteriosamente, il contatore di programma si è portato nella locazione 067 se si fosse eseguito invece Σ+ ci si sarebbe trovati al passo 192 e così via.
Se per caso, invece, la ripartizione corrente non consente l'arrivo al passo di programma corrispondente all'istruzione eseguita, scatta un meccanismo che permette di leggere all'interno della ROM contente queste istruzioni: una sequenza utile è

(2) (OFF/ON) 99 STO 00 10 Op 17 CLR Pgm 01 A $\overline{\mathbf{x}}$ LRN

e stavolta il contatore di programma resta fermo nella posizione 000 (dovuta al reset della TI-59) e mostra il primo byte di questa ROM (82:HIR).
Il programma può essere letto passo a passo fino al byte 583, ma a noi interessa solo la parte fino al passo 379, inoltre il fatto che non si possa usare ne BST ne Ins ne tantomeno Del (che però riserva divertenti sorprese) ci dimostra che non si è affatto ricopiato nella RAM utente questa ROM, ma solo che si è aperta una finestra su di essa: infatti solo nella TI-58C, a manipolazione conclusa, si ritrova l'eventuale programma preesistente.
Tutto questo ci fa concludere che si è trovato una specie di monitor, dato che la sequenza (2) ci permette la lettura della ROM; si può affermare, inoltre, che le istruzioni (A) di cui sopra unitamente alle Op 11, Op 12, Op 13, Op 14, Op 15 costituiscono delle "utilities" di sistema codificate in SOA.
Come si può vedere dalla tabella sotto riportata compaiono nell'ordine le istruzioni:

ISTRUZIONI	PASSO	ISTRUZIONI	PASSO	ISTRUZIONI	PASSO
Op 14	000	Op 13	149	INV D.MS	341
Op 15	000	Σ+	192		
Op 12	002	INV Σ+	213		
Op 11	067	INV P/R	250		
x_m	067	P/R	284		
INV x_m	107	D.MS	303		

Si possono notare due cose interessanti, scorrendo il listato della ROM:

1) innanzitutto la presenza nelle locazioni 045 e 082 di un HIR 20 che, com'è noto dalla p.s., dovrebbe equivalere ad un Nop; vedendo invece che nelle routine che iniziano da 000, 002 e 067 esiste una parte iniziale comune e che l'HIR in questione appare nel punto di diramazione si può pensare che questo funzioni come un GTO indiretto in cui il registro che l'indirizzo del salto è un registro inaccessibile del S.O.
2) Nella sequenza relativa a D.MS non compare alcuna routine che tronca le cifre di scorta: ciò significa che questa caratteristica dia di D.MS che di EE è svolta ad un livello più basso.
3) E' ora possibile vedere di quali registri H si servono queste funzioni: non si può confermare quanto apparso nella prima parte di questo articolo con l'avvertenza che gli H di indice più basso (H1, H2, H3) sono dovuti non alle routine stesse bensì all'esistenza di livelli di parentesi e operazioni in sospeso in queste.