Software RPN

a cura di Paolo Galassetti

Prima di questo articolo vi consigliamo di leggere, in questo stesso numero, la recensione del libro "Synthetic Programming on the HP 41 C". Si tratta di una pubblicazione che spiega le "istruzioni segrete" della 41 C, alla quale si è fatto riferimento per la redazione di questo articolo. Il libro conferisce alla 41 C, specie in alcuni campi, una flessibilità ancora maggiore. Come sempre, però ... non è tutto oro quello che luccica; anche la programmazione sintetica ha i suoi inconvenienti.
Rinnoviamo, dunque, l'invito a leggere prima la recensione del libro (peraltro interessantissimo, se non altro per cultura).

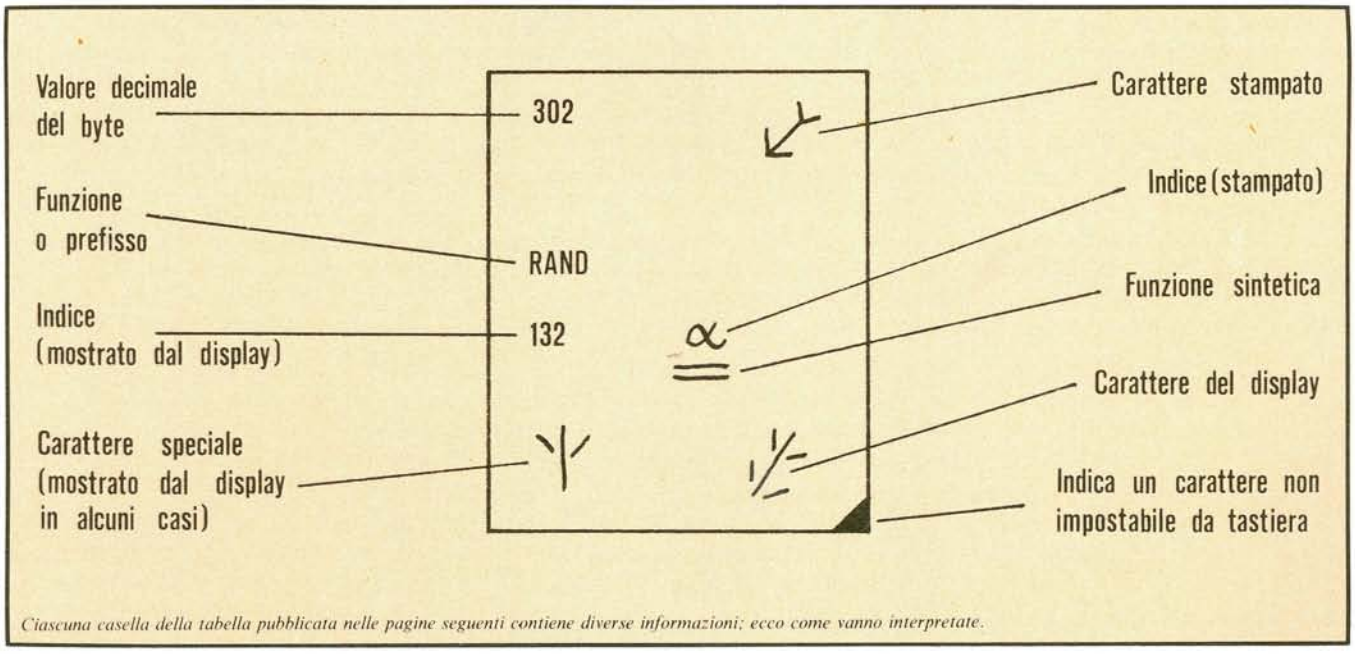
Chiunque abbia un minimo di conoscenza circa il funzionamento di un calcolatore elettronico sa che questo non è un "cervello" mostruoso o, peggio ancora, il frutto di chissà quale magia, ma semplicemente l'unione di un gran numero di elementi capaci di lavorare velocemente con informazioni elementari di tipo binario.

Insegnare a queste macchine come compiere una determinata operazione, sia pure una semplice addizione, significa fornire loro tutte le istruzioni necessarie a compiere l'operazione nell'unico linguaggio da esse conosciuto: il codice binario. Per semplificare le cose, le macchine vengono costruite in modo da interpretare le istruzioni fornite dall'operatore in un linguaggio più comprensibile all'uomo, e tradurle automaticamente in linguaggio macchina. È così che si dimentica facilmente che dietro una istruzione "GTO", "SIN" o "RTN" la macchina lavora con una serie interminabile di zeri ed uno. Le programmabili, senza fare eccezione, fanno anch'esse corrispondere a ciascuna istruzione impostata il relativo codice binario che, anche in questo caso è l'unico linguaggio comprensibile alla macchina.

Normalmente il linguaggio macchina viene ignorato quando si lavora con una programmabile, dato che essa stessa provvede a formare i codici necessari senza che l'operatore debba preoccuparsene. Ciò non toglie che qualcuno possa essere interessato ai codici usati dalla macchina, tenuto conto che tale conoscenza permette in alcuni casi di ottenere risultati interessanti. Per quanto riguarda la 41-C, il discorso è particolarmente valido, poiché una conoscenza particolareggiata dei codici usati permette di ottenere operazioni altrimenti impossibili.

Le istruzioni di programma, i caratteri e i dati vengono memorizzati dalla 41-C sotto forma di una lunga fila di bit che possiamo considerare raggruppati in byte (8 bit), ciascuno dei quali risulta diviso in due "nybble" (un nybble = 4 bit), ad esempio, l'istruzione "STO 12", nella memoria della macchina è rappresentata dal byte 2C composto dai nybble 2 (in binario 0010) e C (in binario 1100). Non tutte le istruzioni però sono codificate in un solo byte, alcune, come risulta anche dalle indicazioni fornite dal manuale d'uso della 41-C richiedono due o più byte. Vediamo quindi i codici da essa usati per memorizzare istruzioni e dati. Partendo dalla cosa più semplice, cominciamo col vedere come viene codificato un numero impostato, per esempio +1,364582197E-13. Considerando che 4 bit possono assumere valori da 0 a 15 (in esadecimale da 0 a F), un nybble è più che sufficiente a memorizzare tutti i numeri da 0 a 9; facendo i conti, per memorizzare il numero preso nell'esempio, occorrono: 1 nybble per il segno della mantissa, 10 nybble per la mantissa (se essa è costituita da meno di 10 cifre, vengono inseriti zeri fino a formare comunque una mantissa di 10 cifre), 1 nybble per il segno dell'esponente, 2 nybble per l'esponente; in totale 14 nybble cioè 7 byte il che coincide con quello che tutti sanno: un registro della 41-C occupa 7 byte. Per ciascuna cifra, il rispettivo nybble assume il suo valore binario (per esempio 7 sarà 0111), per i segni, la 41-C assume per il segno "meno" il codice 1001 e per il segno "più" 0000.

Per i programmi, la macchina assume come unità elementare il byte, indicato (da noi) con i valori esadecimali dei due nybble che lo compongono. Come strumento col quale ricavare il codice relativo a ciascuna



Ciascuna casella della tabella pubblicata nelle pagine seguenti contiene diverse informazioni; ecco come vanno interpretate.

istruzione, risulta utilissima la tabella riportata nella figura a fondo pagina; il valore esadecimale posto a fianco di ogni riga rappresenta il primo nybble del byte in questione; il valore posto sopra ogni colonna, invece, rappresenta il secondo nybble; per esempio la funzione "TAN" è codificata con il byte 5B: tutte le volte che la 41-C incontra, durante lo svolgimento di un programma, il byte 5B, essa calcolerà la tangente del valore impostato in X. A meno che il byte 5B sia preceduto per esempio da un byte 90, perchè in tal caso, essendo tale byte corrispondente al prefisso "RCL", il 5B non verrà interpretato più come "TAN" ma come indice 91, per cui l'istruzione 90 5B sarà RCL 91, chiaro? Evidentemente, un unico byte, a seconda del prefisso cui segue o del modo in cui viene utilizzato (ALPHA per esempio), può avere diversi significati, come, del resto, la tabella mostra chiaramente facendo corrispondere più funzioni allo stesso byte. Per vederne uno, il byte 5A vale: "COS" se è solo, 90 se preceduto da un prefisso (sarà quindi RCL 90 o STO90) e "Z" come ALPHA DATA o carattere da stampare.

Da notare che ogni indice presente nelle righe da 0 a 7 è ripetuto nelle righe da 8 a F; questo perchè gli indici della prima metà della tabella (0-7) valgono come indice diretto e quelli della seconda metá come indice indiretto; per esempio, 9A 34 corrisponde ad ASTO 52 ma 9A 4B vale ASTO *IND* 52. Nelle righe 0, 2 e 3, potrebbero far sorgere qualche dubbio le istruzioni ad un solo byte come RCL 10, STO 02 eccetera, dato che pocanzi avevo detto che una istruzione del genere è composta da due byte (un prefisso ed un indice); questa è una soluzione adottata dalla 41-C per permetterci di ottenere codici brevi e quindi esecuzioni rapide e poco spazio occupato in memoria, per gli accessi ai registri da R00 a R15. Per quanto riguarda le LABEL ora dovrebbe essere chiaro il motivo per cui nel manuale si parla di "etichette in forma breve" e "etichette in forma lunga": le prime (LBL 00÷15) occupano un solo byte, le seconde (LBL 16÷99) occupano due byte. Le etichette in forma lunga possono anche assumere come indici i byte da 66 a 6F e da 7B a 7F definendo così le cosiddette "label alpha locali" da LBL A a LBL J e da LBL a a LBL e. Anche per le istruzioni "GTO" vale il discorso fatto per le etichette. Nella riga B vediamo 15 istruzioni GTO complete di indice (00÷14) per cui lo spazio da esse occupato è di un solo byte; tuttavia quando un GTO 00÷14 viene impostato in memoria, la 41-C lascia vuoto il byte immediatamente successivo, per cui in effetti, l'istruzione di GTO 00÷14 occupa due byte; il secondo byte è lo spazio in cui verrà poi immagazzinata l'informazione relativa alla lunghezza e alla direzione del salto necessari a raggiungere direttamente la LBL indirizzata, senza doverla ricercare sequenzialmente ogni volta. Nel caso delle istruzioni GTO da due byte la massima lunghezza del salto che può essere memorizzata (in un byte) è di 16 registri, per cui, se la label ricercata distasse più di 16 registri (112 byte) dall'istruzione di GTO, il puntatore ogni volta ricercherebbe sequenzialmente l'etichetta, con notevole spreco di tempo. In grado di registrare distanze ben più lunghe (fino a 512 registri,

Questa tabella risulta uno strumento indispensabile per ricavare i codici esadecimali corrispondenti a ciascuna istruzione.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	8 ◆ NULL 00 _	1 × LBL 00 01	2 x̄ LBL 01 02	3 ← LBL 02 03	4 α LBL 03 04	5 β LBL 04 05	6 Γ LBL 05 06	7 ↓ LBL 06 07	8 Δ LBL 07 08	9 σ LBL 08 09	10 ◆ LBL 09 10	11 λ LBL 10 11	12 μ LBL 11 12	13 ∠ LBL 12 13	14 ↑ LBL 13 14	15 ↕ LBL 14 15	0
1	16 ⊟ 0 16	17 Ω 1 17	18 δ 2 18	19 Á 3 19	20 á 4 20	21 Ä 5 21	22 ä 6 22	23 Ö 7 23	24 ö 8 24	25 Ü 9 25	26 ü . 26	27 fE EEX 27	28 ce CHS 28	29 ≠ GTO α 29	30 £ XEQ α 30	31 ⊠ SPARE 31	1
2	32 RCL 00 32 (space)	33 ! RCL 01 33	34 " RCL 02 34	35 # RCL 03 35	36 $ RCL 04 36	37 % RCL 05 37	38 & RCL 06 38	39 ' RCL 07 39	40 (RCL 08 40	41) RCL 09 41	42 * RCL 10 42	43 + RCL 11 43	44 , RCL 12 44	45 − RCL 13 45	46 . RCL 14 46	47 / RCL 15 47	2
3	48 0 STO 00 48	49 1 STO 01 49	50 2 STO 02 50	51 3 STO 03 51	52 4 STO 04 52	53 5 STO 05 53	54 6 STO 06 54	55 7 STO 07 55	56 8 STO 08 56	57 9 STO 09 57	58 : STO 10 58	59 ; STO 11 59	60 < STO 12 60	61 = STO 13 61	62 > STO 14 62	63 ? STO 15 63	3
4	64 @ + 64	65 A - 65	66 B * 66 n	67 C / 67	68 D X<Y? 68	69 E X>Y? 69	70 F X≤Y? 70	71 G Σ+ 71	72 H Σ- 72	73 I HMS+ 73	74 J HMS- 74	75 K MOD 75	76 L % 76	77 M %CH 77	78 N P-R 78	79 O R-P 79	4
5	80 P LN 80	81 Q X^2 81	82 R SQRT 82	83 S Y^X 83	84 T CHS 84	85 U e^X 85	86 V LOG 86	87 W 10^X 87	88 X e^X-1 88	89 Y SIN 89	90 Z COS 90	91 [TAN 91	92 \ ASIN 92	93] ACOS 93	94 ↑ ATAN 94	95 _ DEC 95	5
6	96 ` 1/X 96	97 a ABS 97	98 b FACT 98	99 c X≠0? 99	100 d X>0? 00 100	101 e LN1+X 01 101	102 f X<0? A 102	103 g X=0? B 103	104 h INT C 104	105 i FRC D 105	106 j D-R E 106	107 k R-D F 107	108 l HMS G 108	109 m HR H 109	110 n RND I 110	111 o OCT J 111	6
7	112 p CL Σ T	113 q X<>Y Z	114 r PI Y	115 s CLST X	116 t R↑ L	117 u RDN M	118 v LASTX N	119 w CLX O	120 x X=Y? P	121 y X≠Y? Q	122 z SIGN R	123 { X≤0? a	124 | MEAN b	125 → SDEV c	126 Σ AVIEW d	127 ⊢ CLD e	7
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

cioè più dell'intera memoria della 41-C) sono le istruzioni GTO di tre byte della riga D. Per ora non ci soffermiamo sul metodo usato dalla 41-C per codificare l'informazione relativa alla distanza di salto. Le istruzioni "XEQ" della riga E lavorano in modo analogo ai GTO da tre byte, ma, in più, memorizzano in appositi registri l'indirizzo di ritorno dalla subroutine.

Il byte AE vale GTO IND oppure XEQ IND a seconda che venga seguito da un indice compreso nelle righe da Ø a 7 o che venga seguito da un indice compreso nelle righe da 8 a F. La riga F è la riga relativa alle istruzioni di programma contenenti caratteri alpha; per esempio TEXT 5 significa che i cinque byte seguenti nella memoria di programma vanno interpretati come caratteri ALPHA, se impostiamo perciò la riga "PIPPO", la 41-C memorizzerà i byte: F5 (TEXT 5) 5Ø (P) 49 (I) 5Ø (P) 5Ø (P) 4F (0). I conti tornano: infatti il numero massimo di caratteri che noi possiamo memorizzare in una linea di programma è proprio 15, al quale corrisponderà un codice FF seguito dai 15 byte relativi al resto della riga. Nelle caselle relative ad alcuni byte figurano caratteri del display che non possono essere ottenuti direttamente da tastiera, tuttavia essi vengono visualizzati se il loro codice viene incontrato dopo il byte "Fn" (n è il numero esadecimale che va da Ø a F); tali simboli sono distinti da un triangolino nero presente nell'angolo basso a destra della relativa casella. Il byte 7F normalmente è "CLD", ma se esso segue un byte Fn, assume il significato di "APPEND"; per cui impostando la parola "TOM" memorizzeremo F3 54 4F 4D, ma impostando "APPEND TOM" il codice sarà F4 7F 54 4F 4D. I byte da CO a CD (GLOBAL) possono valere come END o ALPHA LABEL; in entrambi i casi il secondo, il terzo e il quarto nybble (seguenti C) indicano la distanza che separa la "GLOBAL" in questione da quella che la precede nella memoria di programma; se il terzo byte non è un Fn allora la "GLOBAL" è un END, se invece il terzo byte è un Fn allora si tratta di una LABEL ALPHA dove il byte successivo a Fn memorizza l'eventuale assegnazione di quella etichetta ad un tasto, ed i rimanenti n-1 byte costituiscono il testo della label. Per esempio LBL "TEST" sarà: Ca bc F5 de 54 45 53 54, dove abc contengono l'informazione relativa alla distanza dalla "GLOBAL" precedente e ne indentificano l'eventuale assegnazione ad un tasto. Infine, i byte nulli ØØ vengono introdotti dalla 41-C per facilitare l'editing pur rimanendo invisibili; tali byte, quando in fase di PACKING sono giudicati superflui dalla macchina, vengono eliminati automaticamente. Per ora abbiamo visto come la 41-C codifica le istruzioni e i dati impostati nella sua memoria. In seguito vedremo come la memoria stessa viene ripartita ed utilizzata. Tutto ciò deve considerarsi un "corso" di approfondimento sulla 41-C che ci permetterà poi di utilizzarla manipolando i singoli byte, il tutto teso ad ottenere prestazioni ancora più sorprendenti di quelle attuali.

MK

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
8	128 ◆ DEG 00	129 × RAD 01	130 x̄ GRAD 02	131 ← ENTER 03	132 α STOP 04	133 β RTN 05	134 Γ BEEP 06	135 ↓ CLA 07	136 Δ ASHF 08	137 σ PSE 09	138 ◆ CLRG 10	139 λ AOFF 11	140 μ AON 12	141 ∡ OFF 13	142 τ PROMPT 14	143 Φ ADV 15	8
9	144 Θ RCL 16	145 Ω STO 17	146 δ STO+ 18	147 Ȧ STO- 19	148 ȧ STO* 20	149 Ä STO/ 21	150 ä ISG 22	151 Ö DSE 23	152 ö VIEW 24	153 Ü ΣREG 25	154 ü ASTO 26	155 Æ ARCL 27	156 œ FIX 28	157 ≠ SCI 29	158 £ ENG 30	159 ▒ TONE 31	9
A	160 XROM 32	[illegible] ' XROM 33	162 " XROM 34	163 # XROM 35	164 $ XROM 36	165 % XROM 37	166 & XROM 38	167 ' XROM 39	168 (SF 40	169) CF 41	170 * FS?C 42	171 + FC?C 43	172 , FS? 44	173 – FC? 45	174 . GTO IND XEQ IND 46	175 / SPARE 47	A
B	176 0 SPARE 48	177 1 GTO 00 49	178 2 GTO 01 50	179 3 GTO 02 51	180 4 GTO 03 52	181 5 GTO 04 53	182 6 GTO 05 54	183 7 GTO 06 55	184 8 GTO 07 56	185 9 GTO 08 57	186 : GTO 09 58	187 ; GTO 10 59	188 < GTO 11 60	189 = GTO 12 61	190 > GTO 13 62	191 ? GTO 14 63	B
C	192 @ GLOBAL 64	193 A GLOBAL 65	194 B GLOBAL 66	195 C GLOBAL 67	196 D GLOBAL 68	197 E GLOBAL 69	198 F GLOBAL 70	199 G GLOBAL 71	200 H GLOBAL 72	201 I GLOBAL 73	202 J GLOBAL 74	203 K GLOBAL 75	204 L GLOBAL 76	205 M GLOBAL 77	206 N X < > 78	207 O LBL 79	C
D	208 P GTO 80	209 Q GTO 81	210 R GTO 82	211 S GTO 83	212 T GTO 84	213 U GTO 85	214 V GTO 86	215 W GTO 87	216 X GTO 88	217 Y GTO 89	218 Z GTO 90	219 [GTO 91	220 \ GTO 92	221] GTO 93	222 ↑ GTO 94	223 _ GTO 95	D
E	224 ᵀ XEQ 96	225 a XEQ 97	226 b XEQ 98	227 c XEQ 99	228 d XEQ 00 100	229 e XEQ 01 100	230 f XEQ A 102	231 g XEQ B 103	232 h XEQ C 104	233 i XEQ D 105	234 j XEQ E 106	235 k XEQ F 107	236 l XEQ G 108	237 m XEQ H 109	238 n XEQ I 110	239 o XEQ J 111	E
F	240 p TEXT 0 T	241 q TEXT 1 Z	242 r TEXT 2 Y	243 s TEXT 3 X	244 t TEXT 4 L	245 u TEXT 5 M [	246 v TEXT 6 N \	247 w TEXT 7 O]	248 x TEXT 8 P ↑	249 y TEXT 9 Q =	250 z TEXT 10 ⊢ ⊤	251 π TEXT 11 a	252 | TEXT 12 b	253 → TEXT 13 c	254 Σ TEXT 14 d	255 ⊢ TEXT 15 e	F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	