Usr e Call

Come già preannunciato nel numero scorso, in questa puntata parleremo di un argomento molto importante e del quale finora nessuno aveva mai parlato: la chiamata di subroutine in linguaggio macchina da programma per mezzo delle istruzioni USR e CALL.

La novità consiste non certo nelle istruzioni, tanto ben note quanto di oscure caratteristiche che quasi nessun manuale riporta, ma nel fatto che finalmente si potranno usare nel modo giusto.

Facciamo stretto riferimento al "BASIC 80 Reference Manual" della Microsoft, manuale che non tutti possiedono e le cui notizie riteniamo senz'altro degne di essere divulgate.

A complemento di quanto abbiamo detto la scorsa puntata, aggiungiamo che, a seconda del tipo del valore numerico, si avrà un'occupazione di memoria differente da tipo a tipo e con disposizione dei byte secondo la tabella che seguirà le prossime considerazioni.

Per i numeri **interi** sappiamo che abbiamo a disposizione 2 byte: come è prassi consolidata nel campo dei microprocessori, analizzando la memoria byte dopo byte (e cioè per indirizzi crescenti) troveremo prima l'LSB e poi l'MSB del valore desiderato.

I numeri in **singola precisione,** come abbiamo visto, occuperanno quattro byte e rispettivamente conterranno la mantissa (prima l'LSB, poi il byte centrale ed infine l'MSB) e poi il byte di esponente.

I numeri in **doppia precisione** infine occupano 8 byte così predisposti: prima i 7 byte della mantissa (a partire dall'LSB per arrivare all' MSB) e per ultimo il byte di esponente.

In pratica il numero viene posto in memoria apparentemente al rovescio, rispetto a come avevamo indicato la scorsa puntata: in fondo si è ancora in accordo con la regola che vuole prima i byte meno significativi e poi quelli più significativi.

In tabella possiamo sintetizzare perciò quello che leggeremo scandendo le celle di memoria che contengono un certo valore:

intero	singola precisione	doppia precisione
LSB MSB	LSB \ NSB | mantissa MSB / esponente	LSB \ | ... | mantissa ... | MSB / esponente

Nozioni preliminari

Innanzitutto ricordiamo che già nel numero 34 di MC, nella rubrica "I trucchi del CP/M", abbiamo suggerito un metodo per disporre di una certa zona di memoria in cui porre la nostra subroutine in Assembler (dello Z80).

Inoltre, quando effettuiamo la chiamata (nel modo che vedremo in seguito) alla subroutine, abbiamo a disposizione 8 livelli di stack, per eventuali salvataggi di registri (PUSH) oppure chiamate di subroutine in linguaggio Assembler (CALL).

Se invece tale spazio non bastasse, allora si salverà lo Stack Pointer (SP) da qualche parte in memoria ed un nuovo stack potrà essere perciò generato, in funzione dello spazio di memoria disponibile alla routine stessa.

Al termine della routine, prima di ripassare il controllo al Basic, bisognerà ripristinare il vecchio valore dello SP.

L'istruzione USR

Come è ben noto, il formato dell'istruzione in esame è il seguente:

USR[<cifra>] (argomento)

dove <**cifra**> può variare da 0 a 9 e l'<**argomento**> è una qualsiasi espressione numerica o di stringa.

In particolare sappiamo anche che <**cifra**> specifica quale delle 10 possibili funzioni USR è stata chiamata, corrispondentemente alla cifra fornita con l'istruzione DEF USR relativa a tale USR.

Inoltre è ancora ben noto che, se la <**cifra**> manca, allora verrà considerata automaticamente l'USRO: in ogni caso l'indirizzo di partenza della nostra particolare USR è determinato con la DEF USR (ad esempio DEF USR5 = &H9000).

Vediamo ora, sempre all'atto della chiamata, qual è il contenuto dei registri dello Z80, contenuto che varierà in funzione del valore di <**argomento**>.

In particolare, dopo la chiamata della subroutine il registro A (l'accumulatore) conterrà un valore strettamente legato al "tipo" dell'<**argomento**> fornito, secondo la convenzione:

Valore di A	Tipo dell'argomento
2	Intero (complemento a 2)
3	Stringa
4	Singola precisione
8	Doppia precisione

Analizziamo per prima cosa i casi in cui l'argomento è un numero, per poi vedere cosa succede nel caso delle stringhe.

In particolare la coppia HL conterrà l'indirizzo del cosiddetto "FAC" (Floating Point Accumulator), cioè una zona di memoria, formata da 8 byte consecutivi, utilizzata dal Basic per effettuare i suoi calcoli matematici interni: è in pratica l'accumulatore (e abbiamo il sospetto che ve ne sia più di uno...) tramite il quale vengono calcolate somme, sottrazioni, moltiplicazioni, divisioni, funzioni trigonometriche, ecc.

Tale "accumulatore", in base al tipo degli operandi coinvolti nelle operazioni, viene utilizzato in parte o tutto, secondo le regole viste la scorsa puntata.

In particolare se l'argomento è di tipo **intero** (che richiede perciò solo 2 byte), saranno utilizzati solo i due byte di indirizzo FAC-3 e FAC-2 rispettivamente per la parte meno significativa e la più significativa del numero in questione, che ricordiamo essere espresso in complemento a 2.

Ad esempio subito dopo la chiamata dell'istruzione

```
B% = 1000
A = USR1 (B%)
```

a disposizione della routine in Assembler, avremo la seguente situazione:

```
FAC-7 xx
FAC-6 xx
FAC-5 xx
FAC-4 xx
FAC-3 E8
FAC-2 03
FAC-1 xx
FAC  xx  ← HL
```

Infatti sappiamo che 1000, posto in una variabile intera vale in esadecimale 03E8H: la coppia HL punta comunque all'indirizzo di FAC e perciò per puntare all'LSB del valore bisogna indirizzare la cella HL-3.

Invece se l'argomento è espresso in singola precisione allora avremo ancora che HL puntano all'indirizzo di FAC, ma per quest'ultimo avremo un'utilizzazione di 4 byte e per la precisione le celle da FAC-3 a FAC, in perfetto accordo con il metodo di memorizzazione di un numero intero.

Ad esempio, se eseguiamo le seguenti istruzioni:

```
P = 50000
C = USR0(P)
```

avremo, appena chiamata la USR0, HL che puntano al FAC il quale conterrà

```
FAC-7 xx
FAC-6 xx
FAC-5 xx
FAC-4 xx
FAC-3 00
FAC-2 50
FAC-1 43
FAC   90  ← HL
```

in quanto già la scorsa puntata avevamo visto che 50000 veniva codificato internamente con i quattro byte 90 43 50 00, che però compaiono in memoria in ordine inverso.

Infine se l'argomento è in doppia precisione, allora il FAC verrà utilizzato interamente, ancora in accordo con la codifica di un valore in doppia precisione.

Ad esempio, ponendo

```
S# = 5
B = USR9(S#)
```

si avrà un FAC così occupato

```
FAC-7 00
FAC-6 00
FAC-5 00
FAC-4 00
FAC-3 00
FAC-2 00
FAC-1 20
FAC   83 ← HL
```

dal momento che il valore 5 viene codificato con 83 20 00 00 00 00 00 00, al solito "rovesciato".

Un discorso a parte si ha nel caso in cui l'argomento dell'istruzione USR è una stringa.

In particolare, subito dopo la chiamata dell'istruzione stessa ed ovviamente ancora a livello Assembler,avremo, oltre all'accumulatore che contiene il valore 3 (come visto), la coppia di registri DE, la quale punta al cosiddetto "string descriptor" e cioè "descrittore della stringa": questo è formato da 3 byte (da cui il valore 3 dell'accumulatore) che rispettivamente contengono la lunghezza della stringa, l'LSB ed infine l'MSB dell'indirizzo di memoria in cui è posta la stringa.

In questo caso bisogna porre una particolarissima attenzione perché l'indirizzo della stringa può essere "all'interno dell'area di programma" nel caso in cui avessimo delle linee del genere:

```
200 A$ = "PIPPO"
210 X$ = USR6(A$)
```

In questo caso infatti l'indirizzo della stringa, posto nello "string descriptor" è proprio quello del byte contenente la prima "P" di "PIPPO": se per caso la routine in Assembler manipola in qualche modo la stringa "PIPPO", allora inevitabilmente si altererà se non distruggerà fatalmente il programma stesso.

Per evitare ciò è consigliabile cambiare l'assegnazione di linea 200 con la seguente:

```
200 A$ = "PIPPO" + ""
```

oppure con

```
200 A$ = "PIPP" + "O"
```

in modo tale che l'MBASIC sia costretto a "costruire" la stringa "PIPPO" in un'altra parte della memoria, per l'appunto nell'apposito "string space": in tal modo si previene una fastidiosa variazione del testo del programma durante la chiamata della subroutine.

Detto questo, vediamo ora, una volta eseguita la routine, quale sarà il valore fornito dalla routine stessa, da restituire al Basic.

In particolare si ha che il valore "di uscita" sarà sempre dello stesso tipo dell'argomento che era stato passato alla subroutine e cioè intero, stringa, reale o in doppia precisione.

Ciò che occorre è che alla fine HL puntino all'indirizzo di FAC oppure DE puntino allo "string descriptor": eventualmente, se la routine lo richiedesse, tanto il FAC che il descrittore possono essere posti in altri punti della memoria totalmente differenti da quelli originari: basta che il tutto sia fatto con le dovute cautele e secondo le regole di codifica più volte viste.

Non dimentichiamoci poi del valore particolare dell'accumulatore!

Controllato perciò che tutto sia a posto ed eventualmente dopo aver ripristinato lo stack pointer, possiamo ritornare al Basic con una semplicissima RET.

Se invece si volesse forzare il "risultato" della routine ad un valore intero, che deve essere posto in HL e ciò indipendentemente dal tipo dell'argomento della chiamata, allora sfrutteremo una particolarissima routine, chiamata MAKINT, che i lettori attenti ricorderanno senz'altro.

Infatti nel numero 38 di MC, nella rubrica "I trucchi del CP/M", abbiamo incontrato, all'indirizzo di memoria 0105H, proprio l'"entry point" di tale routine.

Ecco che ad esempio la nostra routine può terminare in questo modo

```
LD HL, valore intero
LD IX, (0105H)
JP (IX)
```

Infatti poniamo in IX l'indirizzo contenuto nelle celle 105H e 106H e poi saltiamo, con la JP (IX), proprio a tale routine.

Viceversa, possiamo anche forzare il "tipo" dell'argomento ad **intero** (ovviamente solo se **non** era di tipo stringa) effettuando, in testa alla subroutine, una chiamata all'altra routine citata nel numero 36 e che si chiama FRCINT: in parole povere, dato un argomento reale o in doppia precisione, tale subroutine trasformerà l'"argomento passato" in intero, sulla quale poi effettuare l'elaborazione, sapendo che tale valore è posto in HL.

L'"entry point" di tale routine è posto all'indirizzo 0103H ed un esempio di subroutine che sfrutta questa possibilità è il seguente:

```
LD BC, SUBR
PUSH BC
LD IX, (0103H)
JP (IX)
SUBR ...
```

Senza scendere troppo in particolari di questa piccolissima, ma complicatuccia routinetta, diciamo che il caricamento di BC con successivo salvataggio nello stack servono a far ritornare il programma proprio all'etichetta SUBR, al termine della subroutine richiamata con il solito trucco di IX.

Si deve fare così in quanto quest'ultima è una subroutine che perciò finisce con un RET: per far saltare dal RET al punto

dove vogliamo noi, non abbiamo altre strade che quella vista...

L'istruzione CALL

Siamo dunque arrivati alla seconda istruzione, per la precisione un comando, che ci consente di accedere ad una subroutine in linguaggio macchina.

Mentre la USR era una funzione, per cui la subroutine relativa poteva funzionare solo come tale, la CALL è come detto un comando e di conseguenza la subroutine ad essa relativa dovrà fungere da comando: questo fatto ha a sua volta la conseguenza che, mentre per la USR ha senso parlare di ritorno di un valore, secondo le regole viste, nel caso della CALL invece, dovendosi "eseguire" qualcosa, non si dovrà ritornare al Basic alcun valore ed il ritorno avverrà con una semplice RET.

La sintassi del comando CALL è la seguente:

CALL <nome di variabile> [(lista argomenti)]

Innanzitutto <**nome di variabile**> è appunto il nome di una variabile che contiene l'indirizzo di partenza della routine in linguaggio machina: può essere in particolare una variabile di qualunque tipo (meglio se intera, diciamo noi), ma non può essere l'elemento di un vettore o di una matrice.

Questo non è un problema in quanto, avendo ad esempio 10 subroutine in linguaggio macchina i cui indirizzi sono memorizzati in un vettore, basta semplicemente assegnare ad una variabile il contenuto dell'elemento desiderato del vettore ed il gioco è fatto: vedremo comunque in seguito un paio di esempi di chiamate.

Per quanto riguarda la <**lista argomenti**> che può anche mancare, valgono alcune regolette che ora andiamo ad analizzare.

Innanzitutto se mancano gli argomenti, ad esempio con la chiamata

```
SUBR = &H9000
CALL SUBR
```

il controllo passa semplicemente alla subroutine il cui indirizzo è posto in SUBR, per poi tornare all'MBASIC a seguito della già nominata RET.

Se invece gli argomenti esistono, allora bisogna fare una distinzione a seconda se sono uno, due o tre da un lato oppure più di tre dall'altro.

In particolare ad ogni argomento esistente nella chiamata viene associato un parametro, formato da 2 byte, che contiene l'indirizzo del byte meno significativo dell'argomento stesso, data per scontata la codifica dei valori numerici secondo le regole già note.

Ora, se gli argomenti sono in numero minore o uguale a 3, i parametri relativi sono posti rispettivamente:

— in HL quello relativo al primo argomento

— in DE quello relativo all'eventuale secondo argomento

— ed infine in BC quello relativo all'eventuale terzo parametro.

Ad esempio, eseguendo il seguente frammento di programma

```
200 A% = 5 : B = 1.35 : C$ = "A"
210 ADDR = &H7000
230 CALL ADDR (A%, B, C$)
```

all'atto dell'esecuzione della routine in linguaggio macchina, posta all'indirizzo 7000H:

— la coppia HL punterà alla locazione di memoria contenente l'LSB del valore intero 5,

— la coppia DE punterà alla locazione di memoria da dove inizia la quaterna di byte costituenti il valore reale 1.35

— infine per la coppia BC confessiamo che il manuale non è molto esplicito: lasciamo perciò ai lettori il non difficile compito di determinare se l'indirizzo corrisponde allo "string descriptor" oppure alla prima locazione contenente la stringa stessa. Propendiamo di più per la prima versione e rimaniamo in attesa di riscontro da parte dei lettori, che così si potranno esercitare.

Invece nel caso che siano presenti più di tre argomenti allora i parametri sono distribuiti nel modo seguente:

— la coppia HL conterrà l'indirizzo relativo al primo argomento

— la coppia DE conterrà l'indirizzo relativo al secondo argomento

— la coppia BC infine punterà ad una zona di memoria in cui sono posti consecutivamente i parametri relativi ai rimanenti argomenti: in particolare BC punterà all'LSB del parametro del terzo argomento.

È importante notare che, a causa di questo schema generale, la subroutine **deve** sapere con esattezza quanti argomenti aspettarsi, proprio per ritrovarli tutti.

Viceversa il programma chiamante non ha alcun modo di testare se gli argomenti inviati alla subroutine sono in numero esatto e di tipo corretto.

Nel caso in cui non si avesse uniformità di numero e di tipi di argomenti, difficilmente si potrà salvare la situazione e nel caso più favorevole si avrà il crash del sistema!

Altra raccomandazione è di ricordarsi che i parametri associati ai vari argomenti sono degli indirizzi e non i valori degli argomenti stessi.

Concludiamo perciò questa puntata, che ci auguriamo abbia fornito parecchi spunti per programmatori, dai quali aspettiamo senz'altro risposta sotto forma di programmi, utility, ecc, che troveranno senz'altro posto nell'ambito di questa rubrica. MC

QUOTAZIONI sumus

Materiale nuovo imballato

CENTRO ASSISTENZA SPECTRUM

SUMUS s.r.l.
Via S. Gallo 16/r
50129 Firenze
tel. 055/29.53.61

IPEROFFERTE MAGIA SUMUS (QUANTITÀ LIMITATA)

Spectrum 48K con 6 games pack 279.000
Spectrum 48K plus con 6 games pack 339.000
Apple compatibile con tastiera separata di tipo professionale, 64K, doppio processore (6502 + Z-80) 799.000
PC IBM compatibile, 128K, doppio drive da 360K cad., clock calendario con batteria in tampone, interfaccia parallela e seriale 3.150.000

COMPATIBILE APPLE

LEMON II modelli vari ..... telefonare
MOUSE IC 64K biprocessore 679.000
MOUSE IIC biprocessore con tastiera separata ecc. . 799.000

ACCESSORI PER APPLE O COMPATIBILI

Floppy disk controller 79.000
Floppy disk drive (slim o standard) 349.000
Interfaccia stampante EPSON (grafica) 94.000
Interfaccia stampante EPSON con buffer 16K (espandibile on board a 64K con aggiunta integr.) 199.000
Interfaccia stampante CENTRONICS (non grafica) 73.000
Interfaccia stampante GRAPPLER (grafica) 94.000
Scheda CP/M (con Z-80), senza software 69.000
Scheda 80 colonne con soft switch 139.000
Scheda interfaccia seriale RS-232 (no buffer) 79.000
Scheda interfaccia Super Seriale (buffer) 180.000
Scheda espansione memoria + 128K 349.000
Scheda convertitore A/D 16 ingressi 125.000
Scheda musicale 109.000
Scheda sintesi vocale 69.000
Scheda orologio calendario con accumulatori 99.000
Scheda interfaccia monitor RGB 99.000
Scheda PAL (non raccomandata per il colore) 99.000
Scheda programmatore EPROM (2716/32/64) 99.000
Joystick plastico di precisione 42.000
Joystick metallico 37.000
Mouse con software 125.000
Modem con accoppiatore acustico ed interfaccia 259.000
Penna ottica con software 335.000
Language card (espande i vecchi 48K a 64K) 89.000

MONITORS

Monocromatici, vari tipi, primarie marche, da lire 152.000
A colori, vari tipi, primarie marche, da lire 455.000

STAMPANTI

Mannesmann Tally MT-80 (80 cps, 80/132 col.,grafica, Epson compatibile, foglio singolo e modulo continuo) ..... telefonare
Epson RX 80 F/T (stesse caratteristiche ma 100 cps) 737.000
Stampante Welco (stesse caratteristiche ma 130 cps) 699.000
Idem con interfaccia seriale anzichè parall. 730.000
Mitsui 2100, 120 cps, 80/132 colonne, near letter quality 999.000
Margherita, 18 cps 699.000
Idem con tastiera, usabile come macchina per scrivere intelligente o come stampante, completa di display multilinea a cristalli liquidi, correzione automatica 899.000

PLOTTERS

Plotter intelligente Mannesmann Tally Pixy 3, 3 penne formato A4 ..... 999.000
Plotterino/stampante Sony, 4 colori, veloce, possibilità di rotolo, larghezza 21 cm (A4), 80 colonne se usato in modo stampante 534.000

ACCESSORI PER PC/IBM E COMPATIBILI

Cavo stampante PC/stampante parallela 50.000
Unità a disco 5" ¼ aggiuntiva 399.000
Espansione di memoria +64K da montare sulla scheda già esistente 115.000

COMPUTER SANYO

MBC-550 - 16 bit - 128K RAM espandibili a 256 con incrementi da 64K - parzialmente IBM compatibile - grafica alta risoluzione 640 × 200 punti in 8 colori indipendenti - tastiera professionale - interfaccia stampante (senza cavo) - una unità a disco da 160K - compreso MS-DOS, Wordstar, Calcstar, BASIC 2.099.000
MBC-555 - come il precedente ma con due drive ed in più Datastar, Formsort, Reportstar, Spellstar, Mailm. .. 2.699.000
MBC-550/2 - come 550 ma con disco da 360K - comprende programmi "usa Sanyo PC", "programmo in BASIC", "disegno con il CAD", "contabilità", e manuali in italiano 2.450.000
Disk drive aggiuntivo (trasforma 550 in 555) 399.000
Disk drive aggiuntivo (trasforma 550/2 in 555/2) 450.000
Cavo stampante MBC/stampante parallela 59.000
Espansione di memoria, installata, 64K RAM 99.000
Interfaccia RS-232 per serie MBC 118.000

COMPUTERS PORTATILI

BONDWELL 12 - a valigia - 64K RAM - video incorporato da 9" - due unità a disco da 256K (non formattati) cadauno - secondo drive compatibile Spectravideo, Kaypro od Osborne con comando software - interfaccia parallela per stampante - due interfaccie seriali RS-232 - SINTETIZZATORE VOCALE INCORPORATO - uscita monitor supplementare - comprende CP/M, Wordstar, Calcstar, Datastar, Reportstar, Mailmerge 2.721.000

TAVOLI PER COMPUTERS

Ciatti mod. Memory (cm 60 × 82 × 115, piano scorrevole, disponibile bianco, nero, noce) 179.000
Ciatti mod. Logic, (ripiegabile, con ruote e supporto monitor, colori bianco e nero) 289.000
Eledra, tipo piccolo (circa 70 × 80 × 50), colore bianco, progettato per C64, adattissimo a Apple & C. e per stampanti 58.500
Socored, super professionale 346.000
Supporto in plexiglass per stampanti 61.016

PORTADISCHETTI E VARIE

In plexiglass, da 10 dischi 4.237
Da 40 dischi a vaschetta con serratura 24.576
Da 80 dischi a vaschetta con serratura 33.050
Pinza bucatrice per floppy. Consente di usare entrambi i lati del dischetto (Apple, Commodore) 5.932

PREZZI INCREDIBILI SU: APPLE - MACINTOSH - OLIVETTI M 24

SUMUS - LA PIÙ GRANDE ORGANIZZAZIONE DI VENDITA IN TOSCANA DI HOME & PERSONAL COMPUTERS - NON POSSIAMO ELENCARE TUTTO - VENITE A TROVARCI DI PERSONA - SIAMO APERTI ANCHE IL SABATO (fino a estate).

I prezzi qui indicati sono da intendersi franco negozio IVA esclusa. I prezzi e le disponibilità variano - telefonateci prima dell'ordine.