Out e Wait

Proseguiamo la nostra analisi dell'MBASIC, in linea con ciò che ci siamo proposti la scorsa puntata e cioè, in definitiva, conoscere i segreti di tale interprete analizzando un po' alla volta la sua struttura interna.

Ricordiamo ancora, e ci teniamo a farlo, che le notizie riportate sono assolutamente inedite, nel senso che le informazioni non provengono da alcun testo, ma dallo studio compiuto con l'ausilio di programmi tipo DDT o ZSID.

In tale ottica, lo ripeteremo fino alla noia, non è improbabile la presenza di imprecisioni nelle nostre analisi: altrettanto monotonamente auspichiamo un aiuto da parte dei lettori nel segnalare inesattezze, ovviamente suffragate da opportune controdeduzioni, nonché nuove idee e spunti da presentare nell'ambito della rubrica.

Infine consigliamo i lettori di tenere sottomano la rivista n. 38 nella quale, a pagina 164, troviamo la «jump table» relativa alla release 5.21 dell'MBASIC.COM.

L'istruzione OUT

Dopo aver visto la POKE, eccoci dunque ad un'altra istruzione speciale dell'MBASIC, forse non molto usata se non in particolarissimi casi in cui da programma bisogna controllare un'apparecchiatura connessa ad una porta di I/O: tale porta avrà un certo indirizzo compreso tra Ø e 255, che in genere viene riportato dai manuali d'uso del proprio personal.

La sintassi dell'istruzione OUT è la seguente:

OUT <porta>, <valore>

dove tanto <porta> quanto <valore> devono essere due numeri interi compresi tra Ø e 255: <porta> è appunto l'indirizzo della porta di I/O mentre <valore> è appunto il valore che si vuole inviare a tale porta.

Andiamo dunque ad analizzare la routine relativa ad OUT, che in base alla già citata «jump table» è posta a partire dall'indirizzo 1 FEEH.

In questo caso, come per la prossima funzione, riporteremo a fianco delle istruzioni mnemoniche sia l'indirizzo che il contenuto delle celle di memoria, entrambi espressi in esadecimale.

1FEE	CD 53 ØØ	CALL 2Ø53H
1FF1	D3 ØØ	OUT (ØØ),A
1FF3	C9	RET

Andiamo perciò ad esplorare la routine 2Ø53H.

2Ø53	CD 66 2Ø	CALL 2Ø66H
2Ø56	32 ØB 2Ø	LD (2ØØB), A
2Ø59	32 F2 1F	LD (1FF2),A
2Ø5C	CD C7 43	CALL 43C7H
2Ø5F	2C	DEFB ','
2Ø6Ø	C3 66 2Ø	JP 2Ø66H

In particolare troviamo all'indirizzo 2Ø53H la chiamata alla 2Ø66H, che già sappiamo che calcola un'espressione e ne ricava un valore intero compreso tra Ø e 255, posto alla fine dell'accumulatore.

All'uscita di tale routine il valore così ottenuto viene posto nelle locazioni 2ØØBH, che incontreremo più avanti, e 1FF2H: analizzando in quale contesto si trova tale indirizzo, scopriamo che è il secondo byte di un'istruzione di «OUT».

In tal modo il valore che imposteremo come porta verrà forzato all'indirizzo 1FF2H: ad esempio scrivendo in Basic

OUT 25,Ø

otterremo, come prima cosa, la trasformazione di 25 in esadecimale (grazie alla 2Ø66H) e cioè in 19H, valore che verrà poi posto nella locazione di cui sopra per ottenere un'istruzione Assembler

D3 19 OUT (19H),A

Ma che cosa inviamo a tale porta? Proseguiamo l'analisi per scoprirlo.

All'indirizzo 2Ø5CH troviamo la chiamata a 43C7H seguita da un byte che rappresenta la «virgola» in codice ASCII: già conosciamo tale subroutine, la quale verifica che nel testo Basic, subito dopo <porta>, si trovi una «virgola» di separazione.

Infine la JP 2Ø66H non è altro che una furba abbreviazione di

CALL 2Ø66H
RET

in quanto saltando direttamente alla 2Ø66H, la nostra subroutine 2Ø53H terminerà laddove termina, con una RET, la routine 2Ø66H... Anche in questo caso sappiamo che tale subroutine analizza il testo, calcola un'espressione intera da porre in un byte, che nel caso nostro è l'accumulatore.

Ritornando (proprio a causa di una RET!) al programma chiamante, troviamo appunto quella «OUT (xx),A», dove «xx» è stato appena riempito dalla LD (1FF2H),A e dove ora l'accumulatore contiene proprio il valore da inviare alla porta in questione.

Il solito RET termina perciò la routine di questa istruzione: non crediamo ci sia altro da aggiungere anche perché in fondo si trattava di un'istruzione molto semplice.

L'istruzione WAIT

Anche questa è un'istruzione raramente usata dal programmatore «medio» e decisamente «pericolosa» se usata maldestramente, al pari della POKE, in quanto può provocare nella maggior parte dei casi un «crash apparente» del sistema.

Ricordiamo perciò la sintassi della WAIT, per poi vedere la sua semantica:

WAIT <porta>, <valore>[,<maschera>]

In particolare la WAIT serve a monitorare lo stato di una <porta>, bloccando l'esecuzione del programma Basic fino a che il valore letto dalla porta, dapprima XOR-ato con <maschera> e poi posto in AND con <valore>, dia un risultato diverso da Ø. Nel caso in cui la <maschera> non compaia, allora per essa sarà considerato un valore nullo, che lascia inalterato l'altro operando nel corso del calcolo di un Or-Esclusivo.

Supponiamo perciò di voler sincronizzare il nostro programma Basic con un evento esterno, in particolare la transizione ad «1» di uno degli 8 bit (ad esempio il meno significativo, l'LSB) relativi alla porta di I/O di indirizzo 4Ø (28H).

Scrivendo perciò nel nostro programma

WAIT 4Ø,1

andiamo a leggere la porta 4Ø: supponiamo ora che la linea in esame sia ancora a «Ø». In questo caso dalla porta leggeremo un valore nullo, che in Or-esclusivo con Ø (la <masche-

ra> mancante) fornisce ancora un valore Ø, che a sua volta in AND con 1 (il <valore> impostato) dà un valore finale nullo.

Ciò comporta che l'istruzione non viene abbandonata, ma viene ripetuto il ciclo di operazioni a partire dalla lettura della porta: ciò si ripeterà all'infinito se il bit meno significativo della porta non cambierà mai di stato.

Ecco che perciò, impostando un valore di <porta> non opportuno, si entrerà fatalmente in un loop infinito, dal quale si esce *solo* con il reset manuale del computer: ricordiamo a tal proposito che un Control-C non serve assolutamente a nulla in questo frangente in quanto, come vedremo di seguito, all'interno del loop non viene abilitata la lettura della tastiera, così come nessun'altra operazione di «ripensamento».

Nel caso invece che nella nostra porta 4Ø il bit Ø passi alla condizione «1», allora come valore di input dalla porta avremo 1, che rimarrà tale sia dopo lo XOR con Ø (la <maschera> mancante), sia dopo l'AND con 1 (il <valore> impostato nell'istruzione): in tal modo si sarà ottenuto un valore non nullo con il che la routine Assembler relativa alla WAIT viene abbandonata, con l'esecuzione di una RET.

Ma andiamo a verificare tutto ciò analizzando la routine a partire dal suo entry point e cioè 1FF4H, guarda caso subito dopo la RET del comando OUT.

```
1FF4   CD 53 20   CALL 2053H
1FF7   F5         PUSH AF
1FF8   1E 00      LD E,0
1FFA   2B         DEC HL
1FFB   CD 05 13   CALL 1305
1FFE   CA 08 20   JP Z,2008H
2001   CD C7 43   CALL 43C7H
2004   2C         DEFB ','
2005   CD 66 20   CALL 2066H
2008   F1         POP AF
2009   57         LD D,A
200A   DB 00      IN A,(D0H)
200C   AB         XOR E
200D   A2         AND D
200E   CA 0A 20   JP Z,200AH
2011   C9         RET
```

Seguendola passo passo ritroveremo quanto già descritto in precedenza. Cominciamo dalla prima chiamata alla subroutine 2Ø53H: già sappiamo qual è il suo funzionamento ed in particolare ritroviamo che il valore calcolato e posto nell'accumulatore, viene depositato anche nella locazione 2ØØBH. In questo caso, analogamente all'istruzione OUT, si tratta ancora una volta del secondo byte di un'istruzione Assembler, che in questo caso è una IN che consente appunto di leggere dalla porta indicata nella linea di comando della WAIT.

Ritroviamo inoltre che la 2Ø53H legge e calcola un altro byte che pone in accumulatore: all'indirizzo 1FF7H troviamo un PUSH AF che ci consente di salvare il valore ottenuto (proprio il <valore> del comando Basic), il quale ci servirà in seguito per effettuare l'AND. Quindi si inizializza a Ø il registro E, rappresentante la <maschera> e poi si incontra il decremento della coppia HL con la successiva chiamata a 13Ø5H: questo serve per arretrare il puntatore interno alla linea analizzata, per andare a vedere se c'è una virgola.

Sappiamo infatti che la <maschera> può anche mancare: in tal caso il successivo salto condizionato verrà appunto effettuato per andare ad eseguire il resto della subroutine.

Nel caso in cui decidiamo di porre la <maschera> allora troveremo «qualcosa» nel testo, perciò il salto non verrà effettuato ed infine verrà eseguita la solita 43C7H che verifica appunto la presenza di una virgola.

Successivamente la 2Ø66H calcolerà il valore posto subito dopo la virgola ponendolo in accumulatore: da ciò che vedremo dopo, si intuisce che tale valore ottenuto è a monte posto anche nel registro E.

Infatti ci si può rendere conto di ciò andando un po' più avanti nella routine, dove troviamo una XOR E, dove perciò in E ci deve essere la <maschera>.

Siamo dunque arrivati all'indirizzo 2ØØ8H, al quale potevamo essere giunti nel caso di mancanza della <maschera>: il POP AF ripristina il valore del comando WAIT, che viene posto subito nel registro D.

Eccoci ora al ciclo fatidico: con la «IN» leggiamo la <porta>, con XOR E effettuiamo l'OR-esclusivo del valore letto con la <maschera> e con AND D andiamo a calcolare il successivo AND con il <valore> impostato nel comando.

Chiude e controlla il ciclo una JP Z,2ØØA, che impone imperterrita il ritorno indietro in caso di risultato nullo: ecco dunque il ciclo dal quale non si può più uscire, se dalla porta non abbiamo un valore opportuno, oppure se non premiamo il pulsante di reset del nostro personal computer.

MC

ARMONIA s.n.c.

Divisione Computers
IMPORT-EXPORT
COMPUTERS VIDEOGIOCHI ACCESSORI
NASTRI
CONEGLIANO (TV) VIALE CARDUCCI, 5/16
☎ 0438/24918-32988

VENDITA DIRETTA
SPEDIZIONE
IN TUTTA ITALIA

PREZZI IVA COMPRESA

AMSTRAD

CPC 6-128 m. a colori e fosfori verdi		telefonare
CPC 464 m. a colori	L.	900.000
CPC 464 m. a fosf. verdi	L.	700.000
Stampante DMP-1	L.	500.000

COMMODORE

C 128	L.	750.000
CBM 64 EXECUTIVE	L.	1.300.000
CBM 64 e C 16 solo rivenditori		telefonare
Floppy Driver 1541	L.	460.000
Stampante MPS 803	L.	460.000
Monitor 1702 a colori	L.	500.000
Monitor a colori per 64	L.	450.000
Monitor a fosf. verdi	L.	200.000
Commodore PC 10-PC 20		telefonare

SINCLAIR

QL	L.	700.000
Spectrum 48 K PLUS	L.	340.000
Spectrum 48 K	L.	240.000
Kit per trasf.lo Spectrum 48 K in Plus	L.	95.000
Espansion Sistem interfaccia uno + microdrive	L.	290.000
Disk drive 1 per QL	L.	800.000
OPUS discoveri 1	L.	630.000

STAMPANTI

Seikosha GP 50/S	L.	230.000
Seikosha SP 800 per PC IMB e Compatib.	L.	650.000
Mannesmann		telefonare
Honeywell		telefonare

ACCESSORI

Speed-64 velocizza 5 volte il drive con programmi	L.	60.000
Magic Mouse per 64	L.	150.000
Espans. Memoria 16K per VIC 20	L.	70.000
Trattore per MPS 803	L.	35.000
Joystick con interf. Spectrum	L.	35.000
Contenitore da 90 dischetti	L.	30.000
Diskettes 5'' ¼		
Sinclair SF DD (10p.)	L.	25.000
— Nashua SF DD (10 pz.)	L.	33.000
— Nashua DF DD (10 pz.)	L.	40.000
— Verbatim-Verex SF DD (10 pz.)	L.	33.000
— Verbatim-Verex DF DD (10 pz.)	L.	44.000
— Verbatim Datalife SF DD (10 pz.)	L.	40.000
— Verbatim Datalife DF DD (10 pz.)	L.	52.000
Dysan SF DD	L.	50.000

Vasto assortimento di Joystick, Paddle Videogiochi, Programmi, ecc.

PREZZI IVA COMPRESA
Pagamento: In contrassegno all'arrivo della merce, spese di spedizione L. 5.000 per importi inferiori a L. 100.000
Tutto il materiale sarà da noi preventivamente collaudato, l'eventuale materiale difettoso sarà sostituito tempestivamente.
Garanzia 3 mesi dalla consegna.

VENDITA ALL'INGROSSO
CONDIZIONI PARTICOLARI
AI RIVENDITORI

ARMONIA s.n.c.
Viale Carducci, 5 - 31015 Conegliano (TV)
Tel. 0438/24918 - 32988