Scuola di videogame
... Moltiplicando sprite

Questo mese concludiamo il discorso portato avanti nelle ultime tre puntate con l'esposizione di altre due tecniche relative alla moltiplicazione degli sprite

Se infatti il listato proposto precedentemente è in grado di effettuare questa delicata operazione è anche vero che esso presenta notevolissime limitazioni. Il principale inconveniente è quello che i 24 sprite non possono essere posizionati liberamente nell'intera area dello schermo. Il motivo di questa mancanza è molto semplice ed è dovuto alla particolare tecnica utilizzata. Il programma effettua una divisione dello schermo in tre «zone», in ognuna delle quali un gruppo di otto sprite è libero di «vagare». Un'operazione del genere è necessaria in quanto, come noto, il 64 non può visualizzare più di otto sprite se non riutilizzandoli nella stessa «pennellata video», in posizioni diverse (cfr. MC n. 71; pag. 176). Il programma quindi è stato studiato per sfruttare questa caratteristica, ma lo fa nel modo più elementare possibile e cioè impostando tre diverse interruzioni raster ed effettuando, al verificarsi di ciascuna di esse, un cambiamento delle sole coordinate degli otto sprite, che devono necessariamente essere limitate in modo da far visualizzare ogni gruppo di sprite nell'area ad esso dedicata dalle due IRQ tra le quali è compreso. Si ha quindi la visualizzazione di 24 sprite, ma questi non hanno né forma né colori indipendenti da gruppo a gruppo e quindi risultano gemelli a triplette. Tuttavia quest'ultimo «inconveniente» non dipende dalla tecnica utilizzata, ma solo dalla semplificazione fatta al listato per renderlo comprensibile anche ai meno addentrati nel linguaggio macchina del C-64. Tenete presente infatti che per molti lettori è stata la prima «lezione» e a mio avviso siamo partiti anche troppo di corsa.

Prima di menzionare le tecniche alternative vorrei concludere il discorso della puntata precedente.

Ricordate quando parlavamo della BPL? Dicevamo che si trattava di una istruzione che eseguiva un «salto condizionato». La figura 1 mostra un elenco completo delle istruzioni del 6510 con relativo breve commento della loro funzione. In questo elenco troviamo altre sette istruzioni simili alla BPL, ma aventi ognuna un «tipo di condizionamento» diverso. Queste istruzioni effettuano le loro scelte basandosi sui valori contenuti nei bit del registro di stato (status register). Nel numero di aprile avevo promesso che avrei riparlato di questo particolare registro... Si tratta per l'appunto di un «byte» che contiene dei «bit segnalatori», detti anche «flag». La figura 2 illustra la posizione di ognuno di questi flag. Essi vengono alterati ogni qualvolta viene eseguita una istruzione che ne implica l'intervento. Ad esempio la «LDA» altera i contenuti del flag N e del flag Z. Il flag N è il segno del byte caricato nell'accumulatore che corrisponde al valore del bit più significativo (l'ottavo). Il flag Z indica (quando è posto a «uno») che il byte caricato ha contenuto nullo (uguale a «zero»).

A proposito del flag di segno (N): per rappresentare numeri con segno negativo il 6510 (ma anche molti altri microprocessori) utilizza la tecnica del complemento a due, che consente di rappresentare, con un solo byte, numeri che vanno da −128 a +127. Detta

Registro di stato (status register)

7	6	5	4	3	2	1	0
N	V	-	B	D	I	Z	C

7) Segno
6) Overflow
5) NON UTILIZZATO
4) Break
3) Decimale
2) Interrupt
1) Zero
0) Carry

Figura 2

Esempi di numeri negativi ottenuti con la tecnica del complemento a due

Numero positivo		Inversione dei bit	Numero negativo
Decimale	Binario		Addizione di "uno"
10	00001010	11110101	11110110
2	00000010	11111101	11111110
1	00000001	11111110	11111111

Figura 3

tecnica attribuisce convenzionalmente al bit più significativo di ogni byte il «segno» del valore che esso rappresenta. È chiaro quindi che così facendo il valore numerico è espresso dai soli primi 7 bit. Con sette bit sono possibili 128 combinazioni quindi se il bit più significativo è a zero il valore rappresentato può variare da 0 a 127, viceversa se il bit più significativo è a uno il valore rappresentato varia da −128 a −1. È da notare che non è sufficiente cambiare il valore del bit più significativo per invertire il segno del numero in questione (ad esempio per passare da +5 a −5). Infatti per eseguire questa operazione con la tecnica del complemento a due occorre invertire tutti i bit del byte e aggiungere «uno». La figura 3 mostra alcuni esempi di inversioni di segno. Un sistema pratico per effettuare questo calcolo consiste nel sottrarre da 255 il valore del quale si vuole ottenere il complemento a due e poi aggiungere 1. Qualcuno potrà anche chiedersi il perché di tutta questa macchinosità e soprattutto l'utilità della cosa. Vi ricorderete senz'altro la subroutine presa in esame la puntata scorsa. Essa eseguiva un LOOP (ciclo) utilizzando il registro X come contatore. Quando il registro X diventava «più piccolo di zero» il loop finiva. Il valore immediatamente più piccolo di 0 è −1, che in complemento a due si scrive 255. Infatti decrementando di uno un byte avente valore uguale a zero si passa al valore 255 (come volevasi dimostrare!). Dopo questo esempio pratico dovrebbe essere chiaro che il complemento a due non è solo un sistema utilizzato per confonderci le idee.

Torniamo alle istruzioni di salto condizionato e al registro di stato. Come detto questo registro memorizza degli indicatori; ora vedremo che cosa «indicano». Il flag N e il flag Z sono già stati esaminati. Il flag V (overflow) indica che c'è stato un «riporto» dal bit 6 al bit 7: è utile quando si effettuano delle operazioni in complemento a due (addizioni o sottrazioni) per controllare un eventuale errore nel risultato prodotto. Infatti nel caso in cui detto bit vale «1» ciò signifi-

Figura 1

Elenco delle istruzioni del 6510

ADC: somma il contenuto di un indirizzo di memoria con l'accumulatore, più il flag di carry.
AND: effettua un AND logico tra l'accumulatore e il dato fornito.
ASL: sposta tutti i bit contenuti nell'accumulatore, o in una locazione di memoria, di un bit verso sinistra. L'ottavo bit viene depositato nel flag di carry mentre il primo viene sostituito con uno «zero».
BCC: effettua un test del flag di carry. Se il flag è a «zero» effettua una «diramazione». Se è a «uno» continua il programma con l'istruzione successiva.
BCS: simile alla «BCC», ma opera in modo opposto, ovvero dirama se a «uno» e continua se a «zero».
BEQ: effettua un test del flag zero. Se è a «uno» dirama, altrimenti prosegue.
BIT: esegue un AND logico tra l'accumulatore e il dato fornito, ma non memorizza il risultato. Questa operazione tuttavia altera il flag di zero (lo pone a «uno» per un AND che dà risultato zero, viceversa lo pone a «zero») e i flag «Overflow» e «segno» nei quali deposita, rispettivamente, il bit 6 e il bit 7 del risultato.
BMI: effettua una diramazione se il flag di segno è settato.
BNE: esegue una diramazione se il flag di zero non è settato (operazione inversa di «BEQ»).
BPL: esegue una diramazione se il flag di segno non è settato (operazione inversa di «BMI»).
BRK: genera un interrupt forzata. Il contenuto delle locazioni $FFF e $FFFF viene depositato nel PROGRAM COUNTER, mentre il contenuto di quest'ultimo viene depositato nell'area di stak. Anche il registro di stato è memorizzato nell'area di stack, con il flag di BREAK settato.
BVC: controlla il flag di overflow. Se non è settato effettua una diramazione.
BVS: simile alla «BVC», ma dirama se il flag di overflow è settato.
CLC: azzera il flag di carry.
CLD: azzera il flag D.
CLI: azzera il flag I.
CLV: azzera il flag di overflow.
CMP: sottrae il dato fornito al contenuto dell'accumulatore, ma senza depositare il risultato. Tuttavia i flag «SEGNO», «ZERO» e «CARRY» vengono normalmente alterati.
CPX: come la «CMP», ma la sottrazione avviene tra il dato fornito e il registro X.
CPY: come la «CPX», ma la sottrazione avviene tra il dato fornito e il registro Y.
DEC: il valore contenuto nella locazione di memoria specificata viene decrementato di «uno».
DEX: simile alla «DEC», ma decrementa di «uno» il registro X.
DEY: simile alla «DEX», ma decrementa il registro Y.
EOR: effettua un «OR-ESCLUSIVO» tra il valore fornito e l'accumulatore.
INC: simile a «DEC», ma esegue l'operazione inversa (incrementa).
INX: simile a «INC», ma opera sul registro X.
INY: simile a «INX», ma opera sul registro Y.
JMP: effettua un salto a una nuova locazione di memoria.
JSR: effettua un salto a una nuova locazione di memoria, ma, a differenza della «JMP», ritorna «indietro» quando incontra una «RTS».
LDA: deposita il dato fornito nell'accumulatore.
LDX: simile alla «LDA», ma deposita nel registro X.
LDY: simile alla «LDX», ma deposita nel registro Y.
LSR: simile alla «ASL», ma lo spostamento avviene verso destra e nel flag di carry viene depositato il valore del primo bit. L'ottavo bit viene azzerato.
NOP: non effettua alcuna operazione.
ORA: esegue un OR logico tra l'accumulatore e il dato fornito.a
PHA: deposita nell'area di stack il valore contenuto nell'accumulatore senza alterare quest'ultimo.
PHP: simile alla «PHA», ma deposita il registro di stato.
PLA: simile alla «PHA», ma riprende un valore dall'area di stack e lo deposita nell'accumulatore.
PLP: simile alla «PLA», ma il valore viene depositato nel registro di stato.
ROL: simile alla «ASL», ma nel primo bit viene depositato il valore contenuto nel flag di carry prima che in quest'ultimo venga depositato il valore dell'ottavo bit.
ROR: simile alla «ROL», ma lo spostamento è effettuato verso destra.
RTI: il registro di stato e il program counter vengono alterati con valori provenienti dall'area di stack.
RTS: utilizza in abbinamento con la «JSR».
SBC: esegue una sottrazione tra dato fornito e accumulatore, con riporto.
SEC: setta il flag di carry.
SED: setta il flag D.
SEI: setta il flag I.
STA: simile alla «LDA», ma effettua l'operazione inversa (l'accumulatore viene depositato nella locazione di memoria specificata).
STX: simile alla «STA», ma deposita il registro X.
STY: simile alla «STX», ma deposita il registro Y.
TAX: copia l'accumulatore nel registro X.
TAY: copia l'accumulatore nel registro Y.
TSX: copia lo STACK POINTER nel registro X.
TXA: copia il registro X nell'accumulatore.
TXS: copia il registro X nello stack pointer.
TYA: copia il registro Y nell'accumulatore.

ca che c'è stato il «coinvolgimento» dell'ottavo bit che in queste operazioni, invece, deve essere usato solo come segno. Il successivo flag è quello di break (B): esso viene posto a uno dall'istruzione «BRK» per differenziare l'interrupt che essa genera da un interrupt hardware. Il bit 3 (quarto bit) indica che è stato attivato il modo decimale (BCD) e che quindi le operazioni vengono fatte seguendo tale metodo (torneremo a discuterne). Il flag I (interrupt) viene condizionato dal programmatore con le istruzioni «SEI» e «CLI» per disabilitare o riabilitare le interruzioni. Infine il flag di carry, che indica un riporto o un prestito e quindi viene attivato (posto a «uno») quando si passa dal valore 255 al valore 0 (con un incremento) e viceversa. Inoltre detto flag viene coinvolto nelle operazioni di «scorrimento» quali «ASL», «LSR», «ROL» e «ROR». Spiegato finalmente il significato e la funzione del registro di stato saranno ora più chiare anche le istruzioni di salto condizionato. C'è da spendere, in ogni modo, ancora qualche parola su queste istruzioni. Esse non effettuano un vero e proprio «salto», nel senso che non possono saltare liberamente a tutte le locazioni

```
:8D97 EA31 11 D0 A9 F0 8D
,8000 78        SEI
,8001 A9 7F     LDA #$7F
,8003 8D 0D DC STA $DC0D
,8006 A9 24     LDA #$24
,8008 A2 80     LDX #$80
,800A 8D 14 03 STA $0314
,800D 8E 15 03 STX $0315
,8010 A9 00     LDA #$00
,8012 8D 12 D0 STA $D012
,8015 AD 11 D0 LDA $D011
,8018 09 80     ORA #$80
,801A 8D 11 D0 STA $D011
,801D A9 01     LDA #$01
,801F 8D 1A D0 STA $D01A
,8022 58        CLI
,8023 60        RTS
,8024 A9 FF     LDA #$FF
,8026 8D 19 D0 STA $D019
,8029 A2 10     LDX #$10
,802B BD 00 81 LDA $8100,X
,802E 9D 00 D0 STA $D000,X
,8031 CA        DEX
,8032 10 F7     BPL $802B
,8034 A2 07     LDX #$07
,8036 BD 11 81 LDA $8111,X
,8039 9D F8 07 STA $07F8,X
,803C CA        DEX
,803D 10 F7     BPL $8036
,803F A2 07     LDX #$07
,8041 BD 20 81 LDA $8120,X
,8044 9D 27 D0 STA $D027,X
,8047 CA        DEX
,8048 10 F7     BPL $8041
,804A A9 57     LDA #$57
,804C A2 80     LDX #$80
,804E 8D 14 03 STA $0314
,8051 8E 15 03 STX $0315
,8054 4C 31 EA JMP $EA31
,8057 A9 FF     LDA #$FF
,8059 8D 19 D0 STA $D019
,805C A2 10     LDX #$10
,805E BD 30 81 LDA $8130,X
,8061 9D 00 D0 STA $D000,X
,8064 CA        DEX
,8065 10 F7     BPL $805E
,8067 A2 07     LDX #$07
,8069 BD 41 81 LDA $8141,X
,806C 9D F8 07 STA $07F8,X
,806F CA        DEX
,8070 10 F7     BPL $8069
,8072 A2 07     LDX #$07
,8074 BD 50 81 LDA $8150,X
,8077 9D 27 D0 STA $D027,X
,807A CA        DEX
,807B 10 F7     BPL $8074
,807D A9 24     LDA #$24
,807F A2 80     LDX #$80
,8081 8D 14 03 STA $0314
,8084 8E 15 03 STX $0315
,8087 4C 31 EA JMP $EA31
,808A EA        NOP
,808B EA        NOP
,808C EA        NOP
,808D EA        NOP
,808E EA        NOP
,808F EA        NOP
```

Figura 4 - Listato del moltiplicatore in alternata.

```
:8D97 EA31 11 D0 A9 F0 8D
,9600 A9 17     LDA #$17
,9602 85 FC     STA $FC
,9604 A4 FC     LDY $FC
,9606 A2 00     LDX #$00
,9608 BD 00 95 LDA $9500,X
,960B DD 01 95 CMP $9501,X
,960E 90 0D     BCC $961D
,9610 85 FB     STA $FB
,9612 BD 01 95 LDA $9501,X
,9615 9D 00 95 STA $9500,X
,9618 A5 FB     LDA $FB
,961A 9D 01 95 STA $9501,X
,961D E8        INX
,961E 88        DEY
,961F F0 03     BEQ $9624
,9621 4C 08 96 JMP $9608
,9624 C6 FC     DEC $FC
,9626 A5 FC     LDA $FC
,9628 F0 03     BEQ $962D
,962A 4C 04 96 JMP $9604
,962D 4C 41 96 JMP $9641
,9630 A2 17     LDX #$17
,9632 BD C0 95 LDA $95C0,X
,9635 9D 00 95 STA $9500,X
,9638 9D 60 95 STA $9560,X
,963B CA        DEX
,963C 10 F4     BPL $9632
,963E 4C 00 96 JMP $9600
,9641 A0 00     LDY #$00
,9643 A2 17     LDX #$17
,9645 B9 00 95 LDA $9500,Y
,9648 DD 60 95 CMP $9560,X
,964B F0 03     BEQ $9650
,964D CA        DEX
,964E D0 F8     BNE $9648
,9650 A9 FF     LDA #$FF
,9652 9D 60 95 STA $9560,X
,9655 BD 78 95 LDA $9578,X
,9658 99 18 95 STA $9518,Y
,965B BD 90 95 LDA $9590,X
,965E 99 30 95 STA $9530,Y
,9661 BD A8 95 LDA $95A8,X
,9664 99 48 95 STA $9548,Y
,9667 C8        INY
,9668 C0 18     CPY #$18
,966A F0 03     BEQ $966F
,966C 4C 43 96 JMP $9643
,966F A9 FF     LDA #$FF
,9671 8D FF 95 STA $95FF
,9674 60        RTS
,9675 EA        NOP
,9676 EA        NOP
,9677 EA        NOP
,9678 EA        NOP
,9679 EA        NOP
,967A EA        NOP
,967B EA        NOP
,967C EA        NOP
,967D EA        NOP
,967E EA        NOP
,967F EA        NOP
,9680 78        SEI
,9681 A9 7F     LDA #$7F
,9683 8D 0D DC STA $DC0D
,9686 A9 B0     LDA #$B0
,9688 A2 96     LDX #$96
,968A 8D 14 03 STA $0314
,968D 8E 15 03 STX $0315
,9690 A9 00     LDA #$00
,9692 8D 12 D0 STA $D012
,9695 AD 11 D0 LDA $D011
,9698 29 7F     AND #$7F
```

```
,969A 8D 11 D0 STA $D011
,969D A9 01     LDA #$01
,969F 8D 1A D0 STA $D01A
,96A2 58        CLI
,96A3 60        RTS
,96A4 01 02     ORA ($02,X)
,96A6 04        ???
,96A7 08        PHP
,96A8 10 20     BPL $96CA
,96AA 40        RTI
,96AB 80        ???
,96AC EA        NOP
,96AD EA        NOP
,96AE EA        NOP
,96AF EA        NOP
,96B0 A9 FF     LDA #$FF
,96B2 8D 19 D0 STA $D019
,96B5 A9 00     LDA #$00
,96B7 8D 10 D0 STA $D010
,96BA A2 07     LDX #$07
,96BC A0 0E     LDY #$0E
,96BE BD 00 97 LDA $9700,X
,96C1 99 01 D0 STA $D001,Y
,96C4 BD 18 97 LDA $9718,X
,96C7 99 00 D0 STA $D000,Y
,96CA BD 30 97 LDA $9730,X
,96CD F0 09     BEQ $96D8
,96CF AD 10 D0 LDA $D010
,96D2 1D A4 96 ORA $96A4,X
,96D5 8D 10 D0 STA $D010
,96D8 BD 48 97 LDA $9748,X
,96DB 9D F8 07 STA $07F8,X
,96DE 88        DEY
,96DF 88        DEY
,96E0 CA        DEX
,96E1 10 DB     BPL $96BE
,96E3 AD 08 97 LDA $9708
,96E6 E9 05     SBC #$05
,96E8 8D 12 D0 STA $D012
,96EB A9 60     LDA #$60
,96ED A2 97     LDX #$97
,96EF 8D 14 03 STA $0314
,96F2 8E 15 03 STX $0315
,96F5 4C BC FE JMP $FEBC
,96F8 EA        NOP
,96F9 EA        NOP
,96FA EA        NOP
,96FB EA        NOP
,96FC EA        NOP
,96FD EA        NOP
,96FE EA        NOP
,96FF EA        NOP

,9760 A9 FF     LDA #$FF
,9762 8D 19 D0 STA $D019
,9765 A9 00     LDA #$00
,9767 8D 10 D0 STA $D010
,976A A2 07     LDX #$07
,976C A0 0E     LDY #$0E
,976E BD 08 97 LDA $9708,X
,9771 99 01 D0 STA $D001,Y
,9774 BD 20 97 LDA $9720,X
,9777 99 00 D0 STA $D000,Y
,977A BD 38 97 LDA $9738,X
,977D F0 09     BEQ $9788
,977F AD 10 D0 LDA $D010
,9782 1D A4 96 ORA $96A4,X
,9785 8D 10 D0 STA $D010
,9788 BD 50 97 LDA $9750,X
,978B 9D F8 07 STA $07F8,X
```

```
,978E 88        DEY
,978F 88        DEY
,9790 CA        DEX
,9791 10 DB     BPL $976E
,9793 AD 10 97 LDA $9710
,9796 E9 00     SBC #$00
,9798 8D 00 D0 STA $D000
,979B A9 A8     LDA #$A8
,979D A2 97     LDX #$97
,979F 8D 14 03 STA $0314
,97A2 8E 15 03 STX $0315
,97A5 4C BC FE JMP $FEBC
,97A8 A9 FF     LDA #$FF
,97AA 8D 19 D0 STA $D019
,97AD A9 00     LDA #$00
,97AF 8D 10 D0 STA $D010
,97B2 A2 07     LDX #$07
,97B4 A0 0E     LDY #$0E
,97B6 BD 10 97 LDA $9710,X
,97B9 99 01 D0 STA $D001,Y
,97BC BD 28 97 LDA $9728,X
,97BF 99 00 D0 STA $D000,Y
,97C2 BD 40 97 LDA $9740,X
,97C5 F0 09     BEQ $97D0
,97C7 AD 10 D0 LDA $D010
,97CA 1D A4 96 ORA $96A4,X
,97CD 8D 10 D0 STA $D010
,97D0 BD 58 97 LDA $9758,X
,97D3 9D F8 07 STA $07F8,X
,97D6 88        DEY
,97D7 88        DEY
,97D8 CA        DEX
,97D9 10 DB     BPL $97B6
,97DB A9 00     LDA #$00
,97DD 8D 12 D0 STA $D012
,97E0 AD 11 D0 LDA $D011
,97E3 09 80     ORA #$80
,97E5 8D 11 D0 STA $D011
,97E8 A9 F5     LDA #$F5
,97EA A2 97     LDX #$97
,97EC 8D 14 03 STA $0314
,97EF 8E 15 03 STX $0315
,97F2 4C BC FE JMP $FEBC
,97F5 A9 FF     LDA #$FF
,97F7 8D 19 D0 STA $D019
,97FA AD FF 95 LDA $95FF
,97FD F0 0B     BEQ $980A
,97FF A2 5F     LDX #$5F
,9801 BD 00 95 LDA $9500,X
,9804 9D 00 97 STA $9700,X
,9807 CA        DEX
,9808 10 F7     BPL $9801
,980A A9 00     LDA #$00
,980C 8D 12 D0 STA $D012
,980F AD 11 D0 LDA $D011
,9812 29 7F     AND #$7F
,9814 8D 11 D0 STA $D011
,9817 A9 B0     LDA #$B0
,9819 A2 96     LDX #$96
,981B 8D 14 03 STA $0314
,981E 8E 15 03 STX $0315
,9821 A9 00     LDA #$00
,9823 8D FF 95 STA $95FF
,9826 8D 10 D0 STA $D010
,9829 4C 31 EA JMP $EA31
,982C EA        NOP
,982D EA        NOP
,982E EA        NOP
,982F EA        NOP
```

Figura 5 - Listato del moltiplicatore riordinatore.

I nuovi registri dei 24 sprite

Coordinate Y	Coordinate X	Coordinate X (MSB)	Forme
$95C0-$95D7	$9578-$958F	$9590-$95A7	$95A8-$95BF

Figura 6

di memoria possibili. Il loro salto è del tipo «relativo», ovvero è in funzione della posizione che esse occupano in memoria. Se difatti ci andiamo a riguardare il vecchio listato del moltiplicatore di sprite (MC n. 73), possiamo osservare che nella colonna relativa ai codici LM esadecimali (quella centrale), in corrispondenza di ciascuna «BPL», troviamo i valori $10 ed $F7. Il primo si riferisce al codice operativo, mentre il secondo è lo «scostamento» in complemento a due. Tradotto in altre parole, le istruzioni di salto condizionato possono effettuare spostamenti in avanti o indietro limitati ai valori +127 e −128. Il valore esadecimale $F7 corrisponde al valore (usando il complemento a due) −8. Ciò si traduce in un salto all'indietro di 8 locazioni di memoria. Le tre «BPL» delle rispettive tre subroutine hanno tutte uno scostamento di «−8» proprio perché devono tutte saltare di 8 locazioni indietro dal punto in cui si trovano! Non è necessario tuttavia, in fase di programmazione, calcolare l'esatta locazione di scostamento «a mano», in quanto praticamente tutti gli assemblatori effettuano detta operazione automaticamente (basta indicare la locazione di memoria alla quale vogliamo saltare). L'unico accorgimento da osservare riguarda appunto i «limiti» di salto che non devono essere oltrepassati, pena un «rifiuto di assemblaggio» da parte del nostro MONITOR-ASSEMBLATORE.

Del vecchio listato non resta da spiegare altro se non il significato delle «JMP $FEBC» e della «JMP $EA31». Dalla locazione $FEBC in poi troviamo una routine del sistema operativo che serve a ripristinare i valori dei registri X e Y e dell'accumulatore con i valori da questi memorizzati prima che avvenisse la IRQ e a ritornare al punto esatto in cui si era interrotto il programma principale. Dalla locazione $EA31 in poi troviamo invece la routine originale delle IRQ provocate dal timer. Essendo quest'ultima routine eseguita normalmente una volta ogni sessantesimo di secondo mentre le tre IRQ avvengono nel giro di un cinquantesimo di secondo, è opportuno, per non variare troppo il tempo di chiamata di detta routine, eseguirla una volta su tre.

E ora veniamo alle «tecniche alternative» (finalmente!).

La prima tecnica, fattami tornare a mente da un lettore di Roma, Massimiliano Leoni, consiste nel «duplicare» gli sprite alternando, con due gruppi di valori, i loro attributi ogni qualvolta si verifica il completamento del tragitto del pennello elettronico, quindi gli sprite vengono riutilizzati in «pennellate alterne». Il listato che realizza quanto detto è quello di figura 4. Il primo blocco di istruzioni (molto simile a quello già utilizzato per l'altro listato) si occupa della definizione di una nuova routine IRQ, che verrà attivata ogni qualvolta il raster arriverà alla posizione «256». Detta routine effettua un trasferimento delle coordinate, delle forme e dei colori, da tre tabelle ai relativi registri degli otto sprite e imposta l'indirizzo di una nuova routine IRQ nelle locazioni $0314 e $0315. Al verificarsi della nuova IRQ ci sarà un nuovo trasferimento, simile al precedente, ma questa volta le tabelle saranno diverse. Ci sarà inoltre il ripristino della routine IRQ che aveva abilitato quella attualmente in esecuzione (quindi la precedente) e il ciclo si ripeterà dall'inizio. Con questo sistema gli sprite sono liberi di muoversi lungo tutta l'area dello schermo, ma com'è facile intuire, «sfarfalleggiano» in modo visibilissimo e, tra l'altro, diventano anche trasparenti. Il trucco in pratica cerca di sfruttare il fenomeno della «persistenza delle immagini», ma ci riesce molto male. Meglio quindi avere una limitazione nel posizionamento degli sprite piuttosto che il fastidiosissimo effetto «fantasmasfarfallante».

La seconda tecnica si avvale sempre della «riutilizzazione degli sprite nella stessa pennellata video» (come faceva il listato del numero 73), ma in modo un po' più avanzato. Il listato che realizza ciò è quello di figura 5. Esso permette di visualizzare 24 sprite posizionabili liberamente su tutto lo schermo, ammesso che non ve ne siano più di otto con la «stessa» coordinata Y. Per farlo funzionare è necessario impostare una:

SYS 38528

Detto questo passiamo ad esaminare il principio di funzionamento. Il succo di tutto il programma sta nella routine che effettua il «riordinamento degli sprite», che consiste nel mettere in ordine crescente le coordinate Y degli sprite e in seguito di associare ad ogni coordinata Y gli altri attributi dello sprite corrispondente, quindi effettuando un riordinamento anche di questi. Così facendo avremo da una parte l'elenco originario dei nostri 24 sprite, che utilizzeremo come se fosse un gruppo di registri di I/O, e dall'altra un elenco in ordine «crescente di coordinata Y», che verrà utilizzato dalla routine moltiplicatrice funzionante in IRQ. Che cosa se ne fa la routine moltiplicatrice dell'elenco ordinato? Semplice: imposta tre diverse IRQ raster, ognuna corrispondente alla coordinata Y rispettivamente del primo, del nono e del diciassettesimo sprite. Per quello che riguarda la prima IRQ essa può avvenire anche alla posizione «zero» del pennello elettronico; le altre due avverranno un po' prima delle effettive coordinate degli sprite (a causa dei ritardi). Al verificarsi della prima IRQ, la routine abilitata esegue un trasferimento di dati degli attributi dei primi otto sprite, dalla tabella ordinata ai registri degli otto sprite «originali» e imposta la nuova posizione del raster alla quale avverrà la prossima interrupt. Inoltre effettua un cambiamento della routine IRQ. La prossima IRQ quindi «chiamerà» una nuova routine che questa volta effettuerà un trasferimento dei dati ordinati del secondo gruppo di otto sprite e, come la prima, muterà l'indirizzo della routine IRQ e la posizione raster alla quale avverrà l'interruzione. La terza routine, chiaramente, effettuerà l'ultimo trasferimento relativo al terzo gruppo di otto sprite. Verrà di seguito abilitata una ultima routine, la quale si occupa della comunicazione tra la routine riordinatrice (che non funziona in IRQ) e la routine moltiplicatrice. Questa routine di interfaccia è necessaria in quanto la routine di riordinamento impiega una frazione di tempo non trascurabile per eseguire la sua operazione e quindi se funzionasse in sincronismo con il raster ruberebbe troppo tempo al resto del programma. Così facendo, invece, la routine riordinatrice può essere chiamata quando vogliamo (ovvero ogni qualvolta c'è un aggiornamento, seppur minimo, delle caratteristiche dei 24 sprite), ma il suo «intralcio» sarà notevolmente ridotto. La comunicazione avviene tramite un byte. Quando la routine riordinatrice ha finito il suo compito essa pone a $FF il byte $95FF. Questo abilita la routine di interfaccia che effettua una copia della tabella ordinata, sicura che in questo momento è stata effettivamente resa tale, e questa copia sarà quella che verrà usata dal resto della routine IRQ. Inoltre detta routine imposterà a zero il menzionato byte $95FF avvisando quindi l'utente che è possibile effettuare un

nuovo riordinamento. A questo punto verrà riabilitata la prima routine IRQ e tutto ricomincerà da capo.

Osserviamo ora come viene effettuato il riordinamento. Innanzi tutto è necessaria una doppia copia delle coordinate Y. Una servirà per il riordinamento delle stesse e l'altra per riordinare anche il resto. Dette copie avvengono chiamando la routine posizionata a partire da $9630 con una

SYS 38448

Effettuata questa operazione preliminare il programma devierà alla locazione $9600 dove troviamo l'algoritmo di riordinamento. Spieghiamone il funzionamento. I 24 numeri, rappresentanti le coordinate Y di altrettanti sprite, sono disposti in ordine casuale. Il programma prende il primo numero della lista e lo confronta con il secondo. Se in detto confronto il primo numero risulta più grande del secondo il programma effettua uno scambio di locazione tra i due, altrimenti lascia inalterate le loro posizioni. Si passa quindi a confrontare il secondo con il terzo numero. Anche qui vale la stessa legge di prima: «scambio» se il secondo è più grande del terzo o altrimenti «posizioni inalterate». Si prosegue così fino a lista terminata. A questo punto il numero più grande della lista si ritroverà sicuramente per ultimo (provate con degli esempi pratici e vi convincerete). La lista comunque non è ancora ordinata. Si effettua quindi di nuovo lo stesso procedimento di «scambio/non scambio» dall'inizio della lista, ma questa volta tralasciando l'ultimo confronto (quello tra il ventitreesimo e il ventiquattresimo numero), dato che il più grande tra i due si trova già nella giusta posizione. Al termine di questa seconda «passata» il secondo numero più grande si ritroverà nella posizione 23. Ancora un'altra passata (tralasciando il confronto 22/23) e anche il terzo numero più grande si ritroverà nella sua giusta posizione (ovviamente la 22). Si procede così fin quando sarà possibile tralasciare anche lo scambio tra il primo e il secondo numero (nota di Andrea de Prisco: a dire il vero la lista risulta ordinata non appena si esegue una passata senza effettuare scambi, e ciò può succedere anche prima dell'ultimo «giro». Tale tecnica, nota in Informatica, si chiama «metodo del gorgogliamento»). A questo punto la lista sarà completamente ordinata. Questo apparentemente complesso sistema utilizzato è (strano a dirsi) quello che ci permette il riordinamento più rapido possibile (almeno spero...). La copia che inizialmente avevamo fatto si è trasformata (a meno che le coordinate non fossero già tutte in ordine) e quindi è servita per non intaccare la lista originale (quella di figura 6), che chiaramante deve restare inalterata. La seconda copia ci serve ora che il programma salta alla locazione $9641. Questa nuova routine si occupa del già citato riordinamento del «resto degli attributi». Dunque, prendiamo la prima coordinata Y dalla lista ordinata e andiamo a ricercare la sua posizione equivalente nella lista non ordinata (non quella originale, ma quella della seconda copia). Quando la troviamo possiamo cancellare (con un $FF), dalla copia della lista non ordinata, la coordinata Y, ma nello stesso tempo eseguiamo un trasferimento, dalla lista originale alla equivalente posizione ordinata (in questo caso la prima posizione), della coordinata X e della forma del corrispondente sprite. Il valore $FF, usato come «cancellino», serve ad eliminare dalla prossima ricerca i dati relativi a questo sprite, perché ormai sono già stati riordinati. È opportuno quindi non utilizzare come coordinata Y la posizione $FF in quanto altrimenti si creerebbe confusione tra dati riordinati e non (tra l'altro il valore $FF posizionerebbe lo sprite fuori schermo, quindi non serve). Terminata l'esecuzione di quest'ultima routine il programma impone, come detto, la locazione $95FF a $FF e in seguito ritorna al programma principale (che poi sarebbe il gioco). A questo punto, a partire dalla locazione $9500 fino alla $955F abbiamo gli attributi degli sprite completamente ordinati e quando avverrà l'IRQ alla posizione raster 256 l'apposita routine potrà usufruire dei nuovi dati. I più attenti avranno notato che i colori non vengono utilizzati, ma questa apparente limitazione non è difficile da eliminare (... lascio a voi il divertimento).

Esistono delle limitazioni nel posizionamento degli sprite che derivano dal fatto che i 24 sprite devono essere posizionati in tre «blocchi» distinti. La prima è quella che, chiaramente, non è possibile posizionare più di otto sprite «fuori schermo» (cioè nel bordo superiore o inferiore... ai lati invece SI!). L'altra invece è che la routine non controlla automaticamente l'eventualità che più di otto sprite siano posizionati nello stesso «blocco» e quindi si «semi-blocca» se accade un avvenimento del genere, ovvero rallenta notevolmente l'esecuzione del programma, generando tra l'altro anche lo sfarfallamento degli sprite. Entrambe le limitazioni, tuttavia, sono eliminabili senza andare incontro a grossi problemi e i più bravi di voi (cioè i più interessati) sapranno certamente cavarsela da soli. In ogni modo, se ne farete richiesta, cercherò di accontentarvi. Buone vacanze!

Megaposta

Siamo due assidui lettori di questa bella rivista, in particolare seguiamo con vera trepidazione l'evolversi di questa sua iniziativa, che ci pare particolarmente interessante e azzeccata. Se scriviamo è per dare il nostro contributo convinti che la nostra esperienza possa essere di spunto ad altri intraprendenti. Ma passiamo adesso al dunque:

Abbiamo affrontato la situazione in maniera tecnica cercando dapprima di rimediare alle apparenti carenze strutturali del nostro amato Commodore 64. Infatti, come tutti ben sanno, la mancanza di hardware grafico limita a poche decine il numero di poligoni pieni tracciabili per secondo...

...Se lei desidera possiamo inviarle un dischetto contenente un dimostrativo delle suddette tecniche.

Tommaso e Luigi Bini, Pistoia.

Innanzi tutto mi scuso con voi per il ritardo con cui questa lettera appare sulle pagine di MC e del grande «taglio centrale» che ho dovuto apportare. Speriamo che nel frattempo non abbiate cambiato idea; sono molto curioso di vedere cosa siete riusciti a combinare!

Ciao Marco, voglio essere molto breve e dirti la ragione di questa lettera. Sto seguendo con molta curiosità i tuoi articoli e lo svolgersi del «Megagame 64». Ho capito che oltre al mitico 64 sei attratto in modo morboso dall'Amiga. Bene, premesso che sono un possessore felice di entrambi i computer e che ovviamente mi sento attratto dalla grafica dei 512 K ho pensato: perché non fare un videogioco per Amiga veloce, ben strutturato, bello graficamente e che ti prenda da morire come OUT RUN?...

Michele Signorile, Bari

Anche tu scusami per il «mutilamento», ma anche così il gioco farebbe una bellissima figura (senza andare a realizzare costose interfacce)...

Inoltre avverto Paolo Costabel, Roberto Sorgente e Marco Marinai che ho letto le loro lettere (anche quella di Marco su dischetto) ma non ho il tempo di rispondergli... alla prossima puntata.

...E infine un grazie a Raffaele Cirillo (Napoli) e a Tonino Librici (Torino) per aver inviato il loro contributo.

MC