Programmi per il Copper

di Maurizio Mangrella - Eboli (SA)

Maurizio Mangrella ha colpito ancora. Non contento delle sue gesta contro Modi Grafici, Scrolling, Copper et similia, questa volta ci offre sul solito piatto d'argento (ah, se tutti fossero precisi come lui!) una scorpacciata di altre importanti informazioni, davvero tante e tutte accompagnate da degno software, che vi stilleremo mese per mese come una preziosa medicina. Oggi l'antipasto: un semplice Assembler Disassembler per scrivere i vostri programmi Copper (Amiga non solo è multitasking, ma da un po' di tempo anche multiprocessor!) per sbizzarrirvi nei modi grafici più strani. Per motivi di spazio non possiamo pubblicare il listato, ma tranquilli... le prossime utility del Mangrella, come detto già in nostro possesso, sono di dimensioni minori...

Riprendo il discorso sul Copper presentandovi alcuni programmi di supporto che (udite udite!) vanno da un Assembler simbolico (capace cioè di gestire label a cui associare particolari valori) a un Disassembler, passando per un interessante demo e un caricatore dei programmi realizzati «in proprio».

Il Copper: parte seconda

Il Copper (come già dissi in un altro articolo) è un piccolo ma velocissimo microprocessore RISC dedicato, il quale è in grado di interpretare solo tre istruzioni, sincronizzandosi con la posizione del Video Beam.
Le istruzioni e il loro OpCodes sono: ▼

Istruzione	OpCode	Funzione
MOVE	0	Trasferisce un dato immediato in un registro dei chip custom
WAIT	1	Aspetta che il Video Beam raggiunga una determinata posizione : nel caso questa sia stata superata, passa direttamente all'istruzione seguente
SKIP	2	Salta la prossima istruzione nel momento in cui viene raggiunta una data posizione del Video Beam o nel caso questa sia stata già superata.

Il formato tipico di una istruzione (anche questo scoperto da me dopo lunghe, estenuanti e «GuruMedit-ose» ricerche) è

```
1a Word     ISTRUZIONE
2a Word     Primo Dato
3a Word     Secondo dato
```

Per la MOVE il primo dato è la word meno significativa dell'indirizzo del registro su cui operare, mentre il secondo dato è, appunto, il dato da trasferire; per WAIT e SKIP, invece, primo e secondo dato sono, rispettivamente, ordinata e ascissa della posizione che il raster deve raggiungere. Quindi gli schemi tipici sono

```
MOVE    registro, dato
WAIT    ordinata, ascissa
SKIP    ordinata, ascissa
```

Il Copper accede ai registri dei chip custom (la maggior parte dei quali è a sola scrittura) durante la scrittura sul video di una riga di schermo, mentre tutti gli altri dispositivi dell'Amiga (microprocessore, canali DMA, etc.) accedono alla memoria durante gli Horizontal Blanks, che durano circa 63-64 microsecondi.

Ogni ViewPort è dotata di una o più CopLists, che sono i programmi del Copper: una CopList è un insieme contiguo di istruzioni, che vengono eseguite tutte in fila, dalla prima all'ultima (il Copper non è dotato di istruzioni di salto); si deve anche ricordare che il Copper non possiede registri interni, alla maniera di un normale microprocessore: il che, viste le applicazioni tipiche, non è una grave limitazione.

Una CopList si compone di una «introduzione» (una struttura dati che ne definisce le caratteristiche) e delle istruzioni vere e proprie; senza troppo dilungarmi, ponendo che sia cpl& l'indirizzo di una struct CopList, i parametri principali saranno i seguenti:

```
PEEKL (cpl&+12)
fornisce l'indirizzo della prima istruzione
PEEKW (cpl&+28)
fornisce il numero di istruzioni di cui si compone la CopList
PEEKW (cpl&+30)
non so precisamente cosa sia (il C lo chiama Max-Count), comunque, in genere, è superiore di una unità al valore precedente.
```

Delle tre CopLists presenti in una ViewPort (DspIns per lo schermo, SprIns per gli sprite e ClrIns per i colori) la più importante è DspIns, in quanto è l'unica effettivamente attiva: tutte le CopLists scritte dal Sistema Operativo (già, è proprio così...) terminano con una

```
WAIT        10000, 255
```

istruzione palesemente scorretta (il Video Beam non raggiunge mai l'ordinata

È disponibile, presso la redazione, il disco con il programma pubblicato in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 265.

10000...) e le SprIns e le ClrIns si compongono, in effetti, di questa sola istruzione. Secondo me (parere puramente personale) questa istruzione vale come marker di fine programma.

Per attivare una CopList (a patto di possedere un po' di documentazione sui registri interni non dovrebbe essere difficile scriverne una adatta alle proprie necessità) basta fare i «parassiti», ovvero sostituirla alla normale DspIns: dopo averla inserita da qualche parte della memoria (servitevi della AllocMem, richiedendo ASSOLUTAMENTE la Chip Memory) daremo

```
POKEL     cpl&+12,cpi&
POKEW     cpl&+28,n%
POKEW     cpl&+30,n%+1
```

dove cpl& è l'indirizzo della DspIns CopList, dato da

```
cpl& = PEEKL (ViewPort&+8)
```

cpl& è l'indirizzo della prima istruzione e n% è il numero di istruzioni presenti nel programma. A questo punto daremo

```
CALL MrgCop (View&)
```

(MrgCop () è una routine della graphics.library) per ottenere magicamente l'effetto. View& è la View attiva, data da

```
View& = ViewAddress&(0)
```

(ViewAddress& () è una routine della intuition.library). Ci sarebbe anche un altro modo (BASTAAAA! Per favore...), ma è piuttosto scomodo e sconveniente.

CopAssembler

CopAssembler è un Assembler per i programmi del Copper, il quale riconosce le tre istruzioni del microprocessore più un certo numero di direttive. Queste sono:

```
.ORG <indirizzo>
setta l'indirizzo d'inizio del programma (vedi dopo)
.DEF <label> <valore/label>
setta la prima label al valore <valore> o al valore della seconda label specificata
.END
termina il programma.
```

Un valore immediato può essere introdotto in decimale (normalmente) o in esadecimale (facendolo precedere dal segno $). Una label ha priorità su un valore numerico : ad esempio, nel seguente programma,

```
.DEF      $AAAA 5
MOVE      $F100 $AAAA
```

la label $AAAA avrà priorità sul valore esadecimale $AAAA (−21846 dec., considerando il segno) e la MOVE di cui sopra sarà trasformata in

```
MOVE      $F100 5
```

Tenetelo ben presente! Inoltre, ogni quantità numerica deve essere introdotta secondo le convenzioni del Basic, cioè con il segno . Per l'istruzione MOVE c'è una eccezione: le due istruzioni

```
MOVE      $100 $0200
e
MOVE      $F100 $0200
```

sono equivalenti, e trasferiscono — per la cronaca — la quantità $0200 nel registro $DFF100 (Denise ViewPort Mode, vedi dopo). È possibile ridefinire una label, ma l'ultima definizione vale retrospettivamente per tutto il programma: cioè, nel seguente «stralcio»,

```
.DEF      START −1
WAIT      START 0
  .
  .
.DEF      START 0
```

la WAIT sarà trasformata in

```
WAIT      0 0
```

perché l'ultima definizione vale per tutto il programma. Questo perché l'editor elabora le .DEF man mano che vengono inserite, e l'ultima sarà quella effettivamente disponibile all'assemblatore.

Per cancellare l'ultima riga introdotta, basta digitare D <RET>.

Alla partenza del programma, vi troverete davanti lo schermo vuoto con la sola presenza, in basso a sinistra, di un «_». Scrivete le righe del programma e premete <RET>: quanto scritto sarà elaborato e trasferito direttamente nel listato in alto. L'ultima riga dello schermo ospita eventuali messaggi d'errore: una riga con sintassi non corretta non viene accettata, e offre «un'altra possibilità».

Dopo aver digitato .END apparirà un piccolo menu: tramite questo potrete assemblare il programma (premendo A il programma vi chiederà il nome del file contenente il codice oggetto), stampare o salvare il sorgente (quest'ultima operazione forse è inutile, dal momento che, allo stato attuale, il mio programma non consente di caricare un sorgente) o terminare il lavoro. Un'ultima nota sull'editor: eventuali argomenti in eccesso dopo una istruzione o una direttiva vengono semplicemente ignorati; inoltre, si ricordi che per cancellare un carattere errato non servono i tasti cursore ma solo il classico BackSpace.

Il formato del file oggetto è il seguente:

```
LONG      Indirizzo di inizio
WORD      Istruzione
WORD      Primo dato
WORD      Secondo dato
etc.
```

L'indirizzo di inizio, a causa dell'allocazione dinamica della memoria, viene normalmente ignorato: un programma per il Copper, data l'assoluta mancanza di istruzioni di salto, è sempre perfettamente rilocabile.

CopLoader, il piccolo caricatore che allego (completo delle dichiarazioni di libreria e di tutto quanto serve a farlo funzionare) offre una interessante prerogativa: quella di correggere automaticamente gli indirizzi dei bitplane, qualora si sia fatto riferimento a questi; nel caso non si desideri questa opportunità, basta cancellare l'IF block posto quasi al centro della routine di caricamento.

Una raccomandazione per quanto riguarda i vostri listati: nel caso stiate scrivendo una CopList da sostituire a DspIns, mettete DspIns in testa al programma, ovvero fate precedere le vostre istruzioni da quelle normalmente contenute in questa CopList. Questo per non creare inutilmente problemi (ogni volta che si effettua una operazione sullo schermo il S.O. aggiorna automaticamente le CopLists, e si aspetta di trovare quelle «originali»).

CopDisassembler

Il disassemblatore funziona molto più semplicemente: vi chiede il nome di una struttura dati che contiene CopList (potete rispondere con UCopList, CopList o ViewPort) e il suo indirizzo; provvede da solo a scandagliare le strutture dati: ad esempio, con una ViewPort, si ha:

```
Struttura dati    ViewPort
Indirizzo         XXXXXX
```

DspIns	a XXXXXX
SprIns	a XXXXXX
ClrIns	a XXXXXX
UCopIns	a XXXXXX
Struttura dati (UCopList, CopList)	CopList
Indirizzo	XXXXXX

(come schema tipico di risposta). Dopo aver inserito l'ultimo dato appare, in cima al video, l'indirizzo (esadecimale) della CopList, dunque le istruzioni (che «escono» ad una ad una con la pressione di un tasto). Questo programma può rivelarsi utile per conoscere le CopLists «standard» e per modificarle nella maniera opportuna.

Qualche registro

Facendo pratica con il Copper ho anche individuato le locazioni di alcuni registri hardware interessanti. Cominciamo con quelli video: gli indirizzi dei bitplane si trovano nelle longword da DFF0E0 a DFF0F7; poi c'è il ViewPort Mode, sito a DFF100. In quest'ultimo registro, il bit 15 attiva la Hi-Res, i bit 12-14 selezionano il numero di bitplane, il bit 11 attiva l'HAM, il 10 il Dual Playfield, il 7 l'HalfBrite e il 2 l'InterLace. Il bit 9 deve essere sempre (???) settato. I colori sono memorizzati nelle word da DFF180 a DFF1BF.

A partire da DFF120 ci sono i registri per gli sprite: da DFF120 a DFF13F abbiamo 8 long-word che puntano alla descrizione della forma (word alternate per il primo e secondo bitplane, terminate da una long-word a 0), poi 8 gruppi di word così conformati:

SPR0POS DFF140
posizione dello sprite 0 in uno schermo 320×256: i bit 15-8 regolano l'ascissa, quelli da 7 a 0 l'ordinata. La posizione è riferita in maniera assoluta all'inizio della schermata (per un WorkBench normale l'offset per l'ascissa è 64 e per l'ordinata è 44)
SPR0CTL DFF142
valore terminale della y per lo sprite 0 nei bit da 15 a 8, bit di controllo da 7 a 3 (il bit 7, SPRITE_ATTACHED, fonde una coppia di sprite e ne forma uno dotato di 15 colori e il trasparente). I bit da 2 a 0 sono, rispettivamente, i «noni bit» dell'ordinata finale, dell'ordinata iniziale e dell'ascissa
SPR0DATa DFF144
Word di dati per il primo bitplane (sola scrittura)
SPR0DATb DFF146
Word di dati per il secondo bitplane (sola scrittura)

I gruppi sono così localizzati:

Sprite 0	DFF140
Sprite 1	DFF148
Sprite 2	DFF150
Sprite 3	DFF158
Sprite 4	DFF160
Sprite 5	DFF168
Sprite 6	DFF170
Sprite 7	DFF178

Poi (non c'entra molto...) vi do gli indirizzi dei registri di controllo del DMA:

DMACON DFF096
gestisce il DMA: il bit 9 (MASTER) abilita i canali in generale, mentre i bit da 0 a 8 abilitano i singoli canali, secondo il seguente schema:
— Bit 0 Canale audio 0
— Bit 1 Canale audio 1
— Bit 2 Canale audio 2
— Bit 3 Canale audio 3
— Bit 4 Dischi (read/write)
— Bit 5 Sprite
— Bit 6 Blitter
— Bit 7 Copper (!)
— Bit 8 Display video (Raster)
il bit 15 è il solito Set/Reset (vedi dopo)
ADKCON DFF09E
gestisce la modulazione dei canali: i bit da 0 a 3 attivano la modulazione in ampiezza, mentre quelli da 4 a 7 gestiscono quella in frequenza. Gli altri bit gestiscono l'UART e il read/write sul disco. Il bit 15 funziona come sopra (vedi dopo).

Il bit 15 serve per manipolare i bit di un registro (per quei registri che sono dotati di questa funzione): scrivendo una particolare matrice di bit nel registro, i suoi bit settati setteranno quelli del registro se il bit 15 è resettato, e viceversa. Confusi? È comprensibile!

Per quanto riguarda gli altri registri, quelli audio sono disposti «a blocchetti», uno per ogni canale. Un «blocchetto» è così costituito:

AC0PTR_H	DFF0A0
AC0PTR_L	DFF0A2
AC0LEN	DFF0A4
AC0PER	DFF0A6
AC0VOL	DFF0A8
AC0DAT	DFF0AA

puntatore alla tavola di word (parte alta)
puntatore alla tavola di word (parte bassa)
lunghezza della tavola
periodo (numero di sample tra una conversione e un'altra)
volume (0-64)
dati (a sola scrittura)

Questo era il blocco per il canale 0: per i canali 1, 2 e 3 cominciano, rispettivamente, agli indirizzi DFF0B0, DFF0C0 e DFF0D0.

Per una descrizione completa di questi registri, si rimanda all'articolo «DMA Music Composer» di Dante Sbrega, pubblicato sul numero 74 di MC. Infine, i registri del Blitter, che dovreste già conoscere (B... come Blitter, di Paolo Russo), e che vanno usati con una certa attenzione (leggi: anche troppa...).

Conclusioni

A questo punto penso proprio di aver finito. Non allego i listati dei programmi, altrimenti l'articolo diventerebbe chilometrico, ma discuto solo delle variabili e delle procedure (vedi figura a fianco).

MC

```
  CopAssembler .

defind$, defdat$        Indice e dati per le labels
ps, ps1, ps2            Posizioni all'interno di una stringa
ind, stp                Variabili indice per i passi di programma
st                      Numero massimo di steps (400, modificabile)
pin$(stp,2)             Il testo del sorgente : pin$(stp,0) contiene tutta
                        la riga del listato, mentre pin$(stp,1) e pin$(stp,2)
                        contengono, rispettivamente, primo e secondo dato
                        dell'istruzione
in$                     Stringa in input
in$(2)                  Argomenti della stringa in input
pr$                     Stringa da stampare nel listato
ln                      Linea di schermo a cui stampare
orgf%                   Flag di ORG già definita
errf%                   Flag di errore

Assemble:               Assembla il programma
Saves:                  Salva il sorgente
Prints:                 Stampa il sorgente
Ends:                   Termina il programma
FindData:               Trova un dato (numerico o label)
GetKey:                 Attende la pressione di un tasto

  CopDisassembler .

vp&                     ViewPort
cpl&                    CopList
ucp&                    UCopList
cpi&                    Istruzioni del Copper
ad$                     Indirizzo (hex.)
in                      Istruzione
addrvwait, datahwait    Primo e secondo dato
i$(2)                   Nomi delle istruzioni

  CopLoader .

f$                      Nome del file da caricare
length&                 Lunghezza del file
cpi&                    Indirizzo della prima istruzione
cpl&                    Indirizzo di struct CopList
vport&                  La ViewPort
view&                   La View attiva
```

telefon market

TEL. 0461-932424

IL MODO PIU' SEMPLICE PER ACQUISTARE

TUTTI I GIORNI DALLE 9 ALLE 21

PERSONAL COMPUTERS

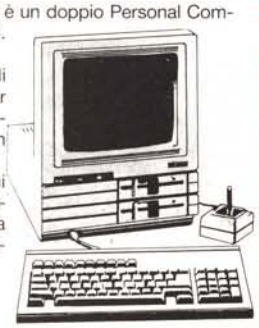
cordata

Una gamma completa di PERSONAL COMPUTERS di provenienza USA (non Taiwan), certificati FCC, di marca, a prezzi assolutamente competitivi. TELEFONATECI, Vi invieremo materiale illustrativo e Listino Prezzi.

CORDATA VI OFFRE UNA GRANDE OCCASIONE

L'UNICO AL MONDO IL DOPPIO P.C.

Basato sulla struttura di un CS 40 il CORDATA Bridge è un doppio Personal Computer perchè in grado di funzionare sia come P.C. MS DOS che come Apple II.
La struttura elettronica di questo P.C. incorpora gli elementi base sia dell'uno che dell'altro computer consentendo all'utilizzatore di disporre concretamente di 2 diversi Personal Computers: un PC XT e un Apple II.
Unico al mondo, è il P.C. ideale per la Scuola, per chi vuole unire l'utile al dilettevole (in modo Apple è prevista la porta "joystick") e per chi debba o voglia passare da un sistema all'altro senza dover rinunciare ai programmi ed all'esperienza acquisiti.



BRIDGE

PERSONAL COMPUTER MONOBLOCCO BIVALENTE MS/DOS e APPLE II con Monitor Integrato

Specifiche tecniche e configurazione:

PC MS/DOS
- INTEL 8088-2 a 4,77 e 8 Mhz
- 512 K RAM espandibile a 768 K
- Monitor alta risoluz. 640×400
- Scheda grafica CGA e OLIVET.M24
- Porta per monitor colori CGA
- Porte Parallela e Seriale
- 2 FDD 5,25" da 360 KBytes

APPLE II
- 65C02 a 1.0 Mhz Apple mode
- 6502 Apple display contr.
- 128 KB Apple RAM
- Apple Disk support
- Apple Game port

L. 2.625.000 + IVA

Ma per chi ordina o prenota "BRIDGE" entro il 30 novembre uno sconto speciale di L. 400.000 (per tutto novembre BRIDGE a Lire 2.225.000 + IVA).

COPROCESSORI MATEMATICI

8087	- 5 MHZ	299.000
8087/2	- 8 MHZ	399.000
8087/1	- 10 MHZ	599.000
80287	- 6/8 MHZ	449.000
80287/8	- 8/10 MHZ	699.000
80287/10	- 10/13 MHZ	799.000
80387/16	- 16 MHZ	1.199.000

HOME COMPUTER (Grandi Occasioni)

sinclair

PACK1	SPECTRUM 128 Plus.2 con registratore di cassette incorporato + Stampante SEIKOSHA GP50S (40 col. - 50 CPS) + 20 Giochi. GARANZIA PER 1 ANNO	490.000

Commodore

PACK2	Commodore 64 (Tipo vecchio) Nuovo di Fabbrica + Joystick + 4 Giochi. GARANZIA PER 1 ANNO	255.000

MOUSE

Commodore

MOUSE	Per Commodore C/64/128 completo di software per disegnare sia su floppy che su cassetta	99.000

LOGITECH

Il MOUSE PROFESSIONALE PIÙ VENDUTO AL MONDO. Tutti i Mouse LOGITECH sono dotati di manuale (italiano o inglese) e di software di base.

MOUSE3	Bus Mouse (Mouse + Scheda) Ingl.	179.000
MOUSE3I	Bus Mouse (Mouse + Scheda) Ital.	199.000
MOUSE1	Mouse Seriale C7 per PC/XT e AT Ingl.	179.000
MOUSE1I	Mouse Seriale C7 per per PC/XT e AT Ital.	199.000
MOUSE4	Mouse per PS/2 IBM Inglese	139.000
MOUSE4I	Mouse per PS/2 IBM Italiano	169.000
MOUSEHR	Mouse High Resolution Inglese	219.000
MOUSEHRI	Mouse High Resolution Italiano	249.000
	SOLAMENTE ACQUISTANDO UN MOUSE LOGITECH È POSSIBILE ACQUISTARE IL SEGUENTE SOFTWARE DA ABBINARE:	
MOUSE11	Logipaint Inglese	40.000
MOUSE11I	Logipaint Italiano "Logipaint" è un programma che consente di disegnare con il P.C.	53.000
MOUSE12	Publisher Inglese	80.000
MOUSE12I	Publisher Italiano "Publisher" il primo Desk Top Publishing	99.000
MOUSE13	LogiCAD Inglese "LogiCAD" il primo CAD, programma per disegno tecnico	99.000

STAMPANTI

SP180	**SEIKOSHA SP180** 80 colonnne = 100CPS/20NLQ specificare se con Interfaccia Parallela IBM o Commodore)	299.000
LC10	**STAR LC10** 80 Colonne = 8 Fonti di carattere = 144CPS/36NLQ (Interfaccia Parallela IBM o Commodore)	519.000
	IDEM a colori	619.000
DM292	**OLIVETTI DM 292** 136 Colonne = 240CPS/40NLQ Interfaccia Parallela IBM	899.000

CONDIZIONI DI VENDITA

Contributo Spese di spedizione:
- Pacco postale normale — NESSUNA SPESA
- Pacco postale urgente — Lire 5.000
- Corriere Espresso TRACO — Lire 10.000

Pagamenti: CONTRASSEGNO

Tutti i prezzi sono IVA esclusa

TELEFON MARKET ITALIA s.r.l.
38100 TRENTO - Via Chini, 1
Casella Postale 334
Telex 400838 TRAVAI I

Desidero ricevere materiale informativo

Nome Società Qualifica Indirizzo CAP Città Telefono