Ricordate l'interfaccia per «pilotare il mondo» con l'Amiga di cui abbiamo parlato un po' di numeri fa su MC? L'autore di quell'infernale aggeggio torna questo mese alla carica con un prodotto leggermente più «morbido»: un bel programma per formattare dischetti. Se vi state chiedendo se disponevate o meno già di un prodotto simile, ossia state facendo gli «spiritosi», vi ricordo che questa rubrica non serve principalmente per espandere la propria biblioteca software, ma, speriamo, per insegnare qualcosa a chi l'Amiga non solo lo usa ma lo programma. E il valore didattico di quest'articolo non è certo da poco.
Leggete, leggete...

XFormat

di Alessandro Manotti - Roma

Il programma (come il suo nome fa dedurre) serve a formattare un disco. Premetto subito che il progetto originale era molto più ampio, ma ciò avrebbe causato la scrittura di una mole di dati molto superiore a quella attuale, quindi avrebbe perso il suo secondo scopo: quello didattico, in quanto troppo lungo e complesso.

In compenso vi fornirò tutte quelle le idee che mi erano venute in mente, così che potrete realizzarle voi, magari abbellendolo anche con della grafica.

A proposito: vorrei ringraziare il sig. Maurizio Mangrella per l'articolo sulla trackdisk.device (MC n° 83) ed il sig. Luca Ceccatelli per l'articolo sull'Amiga Filing System (MC n° 78), che sono stati a dir poco vitali per la creazione di XFormat.

La teoria

Cosa significa (e a cosa serve) formattare un disco?

Beh, cercate di paragonare un disco appena comprato (quindi senza alcun tipo di formattazione) ad un enorme foglio bianco. Se voi provate a scrivere dei conti su un foglio del genere, dovete ammettere che sarebbe piuttosto scomodo sia per la grandezza del foglio stesso (magari inizereste a scrivere di qua e di là creando un grande «casino») sia per la mancanza dei quadretti. Per programmi (come il comando DIR, oppure TYPE, oppure COPY, etc...) che gestiscono i file tramite le funzioni dell'AmigaDOS (tipo Read, Write, Open, etc...) si viene a creare una situazione

ROOT BLOCK

LONG WORD N°...	VALORE	COMMENTO
1	00 00 00 02	Identificatore blocco.
4	00 00 00 48	???
6	.. ..	Checksum.
7 - 78	.. ..	Files della Root Directory.
79	FF FF FF FF	???
80	.. ..	Puntatore FREE SPACE.
106	.. ..	Data alla creazione (od ultimo salvataggio) del file.
107	.. ..	Ora e minuti alla creazione (od ultimo salvataggio) del file.
108	.. ..	Secondi alla creazione (od ultimo salvataggio) del file.
109 - 116	.. ..	Nome del file.
128	00 00 00 01	Identificatore blocco.

Figura 1

```
DiskReq -> iotd_Req . io_Data = (APTR)MainBuffer;
DiskReq -> iotd_Req . io_Command = (UWORD)ETD_FORMAT;
DiskReq -> iotd_Req . io_Length = 11264;    /* cioè 22 settori = 1 cilindro */
DiskReq -> iotd_Req . io_Offset = offset;     /* offset del cilindro da
                                                 formattare */
DoIO(DiskReq);
```

Dove:
MainBuffer è un puntatore all' area di memoria CHIP (primi 512 KBytes) utilizzata come buffer del disco.

E' dato per scontato che sia stato già inizializzato il ChangeNum e che il trackdisk.device sia stato già aperto.

Figura 2

molto simile; l'unica differenza è che voi potete comunque scrivere qualcosa sul foglio, mentre tali funzioni non saprebbero dove mettere le mani... oops... la testina (del drive ovviamente!!!). Bisogna quindi suddividere questo disco «vergine» in tanti «fogli» più piccoli e tracciare su questi ultimi tanti quadretti, infine bisogna scrivere in alcuni quadretti di certe pagine (vedremo dopo quali) delle parole chiave, che permettano all'Amiga di sapere che in quel disco ci sono effettivamente dei dati scritti da un altro Amiga. A questo punto qualcuno si chiederà: ma come si può formattare un disco se le funzioni dell'AmigaDOS non permettono di scrivere su tale supporto (domanda da 100 milioni!)? Risposta: semplice! Non si utilizzano le funzioni dell'AmigaDOS!!!

La trackdisk.device

È a questo punto che in entra in gioco la trackdisck.device. Cos'è? Per una descrizione accurata rivedete MC n. 83 sulla trackdisk.device. Brevemente (per chi non avesse sottomano quel numero) dirò che — «un device è una parte del Sistema Operativo che quest'ultimo provvede a caricare ogni qual volta ne ha bisogno». — «Un device è, in sostanza, solo un programma con il quale il S.O. si interfaccia tramite un normale port, attraverso cui vengono passati i comandi e gli argomenti organizzati in una struttura dati detta IORequest».

Ora, grazie alla trackdisk.device possiamo pilotare direttamente le testine del drive ed i dati che esse scrivono o leggono, abbiamo cioè il drive nelle nostre mani (ghigno maligno)!!!

Come far riconoscere all'Amiga un disco...

... ovvero come non far apparire sul monitor quella maledetta finestrella che al primo «SAVE AS...» di un menu di un programma ci esclama qualcosa del tipo: «mi dispiace, ma il disco non è AmigaDOS, quindi non posso salvare il tuo lavoro di una giornata su tale supporto» ed è a questo punto che avreste voglia di smontare il vostro Amiga tasto per tasto, e tutto questo solo perché vi siete dimenticati di formattare un misero disco. Cos'è che non ha trovato

XFormat

```
#include "stdio.h"
#include "exec/types.h"
#include "exec/memory.h"
#include "exec/ports.h"
#include "exec/io.h"
#include "devices/trackdisk.h"

#define NO_ERR          0
#define ERR             1
#define CYLINDER        11264
#define SECTOR          1024

struct MsgPort *TrackPort , *CreatePort();
struct IOExtTD *DiskReq , *CreateExtIO();

UBYTE *MainBuffer;
UBYTE CheckOpenDevice;

Memory()
{
        if((MainBuffer = (UBYTE *)AllocMem(CYLINDER , MEMF_CHIP ! MEMF_CLEAR)) == NULL) return(ERR);
        return(NO_ERR);
}

void ExitClear()
{
        if(MainBuffer) FreeMem(MainBuffer , CYLINDER);
        if(CheckOpenDevice) CloseDevice(DiskReq);
        if(DiskReq) DeleteExtIO(DiskReq , sizeof(struct IOExtTD));
        if(TrackPort) DeletePort(TrackPort);
        printf("\033[0m");
        _exit();
}

void Syntax()
{
        printf("\n\n\033[33m*** Errore. Parametri errati.");
        printf("\n\n\033[0mSINTASSI:\033[33m XFormat\033[0m <drive> <nome>");
        printf("\n<drive> = '0' per il drive DF0: oppure '1' per il drive DF1:, etc...");
        printf("\n<name>  = nome del disco.");
        printf("\n\nP.S.: XFormat formatta solamente MICROFLOPPY (80 cilindri, 1760 settori).");
        printf("\n\n");
        _exit();
}

void Motor()
{
         DiskReq -> iotd_Req . io_Length = 0;
         DiskReq -> iotd_Req . io_Command = (UWORD)TD_MOTOR;
         DoIO(DiskReq);
}

OpenTrackDisk(DriveSel)
int DriveSel;
{
        int ChangeNum;

        if((TrackPort = CreatePort("DiskPort" , NULL)) == NULL) return(ERR);
        if((DiskReq = CreateExtIO(TrackPort , sizeof(struct IOExtTD))) == NULL) return(ERR);
        CheckOpenDevice = OpenDevice((char *)TD_NAME , DriveSel , DiskReq , NULL);
        DiskReq -> iotd_Req . io_Command = TD_CHANGENUM;
        DoIO(DiskReq);
        ChangeNum = DiskReq -> iotd_Req . io_Actual;
        DiskReq -> iotd_Count = ChangeNum;
        if(DiskReq -> iotd_Req . io_Error) return(ERR);
        return(NO_ERR);
}
```

(continua a pag. 258)

l'Amiga per essere costretto ad inviarvi quel messaggio? RISPOSTA:
1) i primi quattro byte del blocco zero (per intenderci, il primo dei due blocchi in cui si annidano i virus e molti caricatori) devono contenere i valori (in esadecimale): **44 4F 53 00** oppure la stringa: **DOS**\0 (lo «\0» è il codice ASCII 0 utilizzato per informare che la stringa è terminata);
2) il blocco 880 deve essere riconosciuto dall'Amiga come il ROOT-BLOCK (figura 1);
3) si utilizza un blocco (di solito 1' 881) per memorizzare la Free Space (vedremo dopo di cosa si tratta) puntata da una long word (4 byte) che si trova nel ROOT-BLOCK.

Dal dire al fare

Come dice il proverbio? Ah, sì: tra il dire ed il fare c'è di mezzo... l'Amiga (beh, non era proprio così, comunque...). Passiamo quindi al fare.

Per formattare un disco tramite la trackdisk.device, si utilizzano gli stessi comandi usati per scrivere una traccia, con la sola differenza che invece di utilizzare il «comando» ETD_WRITE si utilizza il «comando» ETD_FORMAT (da notare che esiste anche il «comando» standard CMD_FORMAT, ma quest'ultimo non controlla se il disco sia stato cambiato o meno). La figura 2 illustra tutti i parametri da definire per formattare un cilindro, semplice no! I più attenti avranno notato che viene fornito un puntatore ad un buffer, perché? In verità il «comando» ETD_FORMAT non è altro che un rozzo ETD_WRITE, ma ci sono due sostanziali differenze; primo: se, per esempio, nello scrivere un cilindro tramite ETD_WRITE, si incontra un settore che contiene uno o più errori (per es. un «bad sector sum»), non solo verrà ritornato il numero dell'errore (cosa utile), ma il settore non verrà scritto con i dati richiesti neanche se pregate l'Amiga in ginocchio, invece al «comando» ETD_FORMAT non importa assolutamente niente se ci sono degli errori nel settore, lui scrive e basta; secondo: ETD_FORMAT scrive i dati nel disco in modo più rapido di ETD_WRITE (molto probabilmente perché ETD_WRITE si preoccupa di controllare la validità di ogni settore in cui scrive, rallentando quindi, il processo di scrittura dei dati).

Free Space

Chi di voi avrà utilizzato almeno una volta lo Smart Disk, avrà sicuramente utilizzato l'opzione Bitmap. Essa permette di visualizzare quali sono i blocchi liberi e quelli occupati di un dato disco; come fa Smart Disk ad avere tali informazioni? Legge la Free Space!

Ma cominciamo dall'inizio.

La Free Space occupa un solo settore ed indica quali sono i blocchi disponibili in un disco (settandola opportunamente si può ingannare l'Amiga facendogli credere che il disco sia completamente pieno anche se non contiene alcun file!). È proprio a questo settore che

(segue da pag. 257)

```
AnalSin(argc , argv)
int argc;
char *argv[];
{
        printf("\n\033[33mXFormat V1.1.\033[0m Creato Nel 1990 Da Alessandro Manotti.");
        printf("\nSOFTWARE MADE IN ITALY.");
        if(argc != 3) Syntax();
        if((strcmp(argv[1] , "0") == NULL)) return(0);
        else if((strcmp(argv[1] , "1") == NULL)) return(1);
        else if((strcmp(argv[1] , "2") == NULL)) return(2);
        else if((strcmp(argv[1] , "3") == NULL)) return(3);
        else Syntax();
}

FormatDisk(argv)
char *argv[];
{
        register int i = 433 , NewSum = 0 , offset , errs = 0;
        register char *byte , *name , x = 0 , y = 0;
        long int *lbyte;

        DiskReq -> iotd_Req . io_Data = (APTR)MainBuffer;
        DiskReq -> iotd_Req . io_Command = (UWORD)ETD_FORMAT;
        DiskReq -> iotd_Req . io_Length = CYLINDER;

        byte = (char *)MainBuffer;
        lbyte = (long int *)MainBuffer;
        name = argv[2];

        printf("\n     00 01 02 03 04 05 06 07 08 09");
        printf("\n   0 \033[43mFF\033[0m ");
        *(lbyte) = 0x444F5300;
        DiskReq -> iotd_Req . io_Offset = 0;
        DoIO(DiskReq);
        if(DiskReq -> iotd_Req . io_Error)
        {
                errs ++;
                printf("\b\b\b\033[42m??\033[0m ");
        }
        for(offset = 0 ; offset < 5 ; offset++) *(byte + offset) = 0;
        for(offset = 1 ; offset < 80 ; offset++)
        {
                if(x == 9)
                {
                        x = 0;
                        y += 10;
                        printf("\n  %2d \033[43mFF\033[0m " , y);
                }
                else
                {
                        x ++;
                        printf("\033[43mFF\033[0m ");
                }
                DiskReq -> iotd_Req . io_Offset = offset * CYLINDER;
                DoIO(DiskReq);
                if(DiskReq -> iotd_Req . io_Error)
                {
                        errs ++;
                        printf("\b\b\b\033[42m??\033[0m ");
                }
        }
        printf("\n\nInizializzazione del disco...");
        *(byte + 3)   = 0x02;
        *(byte + 15)  = 0x48;
        *(byte + 312) = 0xFF;
        *(byte + 313) = 0xFF;
```

fanno riferimento programmi che, per esempio, permettono di sapere quanti Kbyte sono ancora disponibili nel disco. La Free Space è utilizzata anche da alcune funzioni dell'AmigaDOS come Write(), etc... (cioè le funzioni che scrivono dati nel disco) per sapere in quali blocchi si possono salvare i dati.

Ma vediamo com'è strutturata e com'è possibile salvare in un solo settore (512 byte) le informazioni dei 1760 settori che compongono l'intero disco.

Ogni long word è composta da 32 bit, quindi, se utilizziamo ogni bit per rappresentare un settore, possiamo imporre che, se tale bit è posto ad 1, significa che il settore corrispondente è libero, cioè può essere riempito con dei dati, mentre se è posto a 0, significa che dei dati sono già presenti in tale settore. Tenendo in considerazione che abbiamo a che fare con 1760 settori, significa che avremo bisogno di 55 long word per descrivere la bitmap dell'intero disco. La cosa è più facile a farsi che a spiegarsi, spero comunque che abbiate capito; facciamo comunque un esempio utilizzando lo schema della figura 3.

La long word numero 2 contiene il valore (in esadecimale) FFFFFFFE, ciò significa che sono liberi tutti i primi 32 blocchi del disco tranne il primo, perché il primo bit (ricordo che un numero binario va letto da destra a sinistra) è posto a zero (vedi il relativo valore binario sulla stessa figura); ora dobbiamo fare un'osservazione molto importante: per l'Amiga i primi 2 blocchi di ogni disco risultano SEMPRE occupati, cioè appena formattate un disco non potete salvare (utilizzando le normali funzioni dell'AmigaDOS) più di 900096 byte, contro i 901120 teorici (cioè 1760 blocchi per 512 byte di ogni settore). Non c'è modo di far «vedere» alle funzioni dell'AmigaDOS quei 2 blocchi (nessuna long word della Free Space può controllarli), quindi, se volete scriverci sopra qualcosa, dovete utilizzare le funzioni della trackdisk.device forzando la scrittura dei dati in tali blocchi (con il famoso (!) ETD_WRITE o... ETD_FORMAT!). Vediamo il perché di questa osservazione: poche righe fa abbiamo detto che il valore FFFFFFFE indica che sono liberi i primi 32 blocchi tranne il primo... ma non essendo controllabili i primi 2 (che quindi dobbiamo scartare), significa che i blocchi liberi sono i primi 34 meno il primo, il secondo ed il terzo. La long word numero 3 contiene il valore (in hex.) FFFFFFFC, quindi risultano liberi i blocchi compresi tra il 33 ed il 64 compresi, tranne il 33 ed il 34, ma, a causa dell'osservazione precedente, i blocchi liberi sono quelli compresi tra il 35 ed il 66 compresi, tranne il 35 ed il 36 (è tutto shiftato di due blocchi). Le long word dalla 4 alla 55 contengono (in hex.) FFFFFFFF ciò significa che tutti i blocchi oltre il 66 (osservazione compresa!!!) sono liberi.

Voglio ricordarvi che quando create una Free Space dovete organizzarla in modo che sia il ROOT-BLOCK sia il settore contenente la Free Space stessa devono essere «catalogati» come

```
        *(byte + 314) = 0xFF;
        *(byte + 315) = 0xFF;
        *(byte + 318) = 0x03;
        *(byte + 319) = 0x71;
        *(byte + 432) = strlen(argv[2]);
        while((i < 463) && (*name))
        {
                *(byte + i) = *name;
                i ++;
                name ++;
        }
        *(byte + 511) = 0x01;

        for(i = 0 ; i < 128 ; i++) NewSum -= *(lbyte + i);
        *(lbyte + 5) = NewSum;
        for(i = 128 ; i < 184 ; i++) *(lbyte + i) = 0xFFFFFFFF;
        *(lbyte + 128) = 0xC000C037;
        *(lbyte + 156) = 0xFFFF3FFF;
        *(lbyte + 183) = 0x3FFFFFFF;
        DiskReq -> iotd_Req . io_Offset = 450560;
        DoIO(DiskReq);
        if(DiskReq -> iotd_Req . io_Error)
        {
                printf("\n\033[33m*** Errore. Non posso inizializzare il disco.");
                printf("\n");
                Motor();
                ExitClear();
        }
        return(errs);
}

void main(argc , argv)
int argc;
char *argv[];
{
        int drive , errs;

        drive = AnalSin(argc , argv);

        printf("\n\nInserisci nel drive DF%d: il disco da formattare, poi premi RETURN" , drive);
        getchar();
        if(OpenTrackDisk(drive))
        {
                printf("\n\033[33m*** Errore. Non posso utilizzare il drive richiesto.");
                printf("\n\n\033[0m");
                _exit();
        }
        if(Memory())
        {
                printf("\n\033[33m*** Errore. Non posso allocare la memoria necessaria.");
                printf("\n\n\033[0m");
                ExitClear();
        }

        if(errs = FormatDisk(argv))
        {
                printf("\n\033[33m%d cilindri non sono stati formattati correttamente." , errs);
                printf("\n\033[0m");
                Motor();
                ExitClear();
        }
        printf("\nDisco formattato senza errori.");
        printf("\n\n");
        Motor();
        ExitClear();
}
```

LONG WORD	VALORE (hex.)	VALORE (bin.)
2	FFFFFFFE	1111 1111 1111 1111 1111 1111 1111 1110
3	FFFFFFFC	1111 1111 1111 1111 1111 1111 1111 1100
4 - 55	FFFFFFFF	1111 1111 1111 1111 1111 1111 1111 1111

Figura 3

blocchi occupati, altrimenti correte il rischio di creare nel disco un bel casino!!!

Dopo questa BOTTA intellettuale (sta friggendo il mio cervello solo per fare questi conticini... non oso immaginare le condizioni del vostro!) vi concedo 10 secondi di pausa per freddarvi le meningi: 1... 10! OK, si riparte!!!

Forse non lo avrete notato, ma nella figura 3 inizio la descrizione dalla long word numero 2... e la prima che fine ha fatto? Dovete sapere che anche i 32 bit della long word contenente il Checksum fanno parte della Bitmap, ciò significa che o si scrive nel settore un corretto Checksum, risultando però errata la bitmap dei blocchi 163-192 compresi, oppure si utilizza una corretta Bitmap di tali blocchi ottenendo però un errato Checksum. Per ovviare all'inconveniente i progettisti del S.O hanno utilizzato un trucchetto (a dire il vero un po' scomodo):

1) si calcola il Checksum del settore ponendo la long word numero 1 pari a zero;
2) si esegue la differenza tra il Checksum ottenuto (cioè quello errato) ed il Checksum da scrivere per ottenere una corretta Bitmap;
il risultato di tale differenza nella long word numero 1;
4) si scrive il corretto Checksum (cioè quello per ottenere una corretta Bitmap) al posto del vecchio.

Se ora provate a calcolare il nuovo Checksum vedrete che risulterà corretto.

A proposito: ho detto che la Free Space è scritta dalla long word numero 1 alla 55; da ciò che ho visto le long word rimanenti non servono a niente.

Il programma

Il programma, come qualsiasi programma in C che si rispetti, è suddiviso in più funzioni.

Riguardo la funzione **main** non c'è molto da dire, in quanto si limita a chiamare le altre funzioni e a verificarne il loro corretto svolgimento.

ExitClear esce dal programma chiudendo le porte e liberando la memoria.

Motor permette di far partire/fermare il motorino del drive.

AnalSin controlla la corretta sintassi del comando. A tal proposito voglio far notare che per segnalare al programma in quale drive si trova il dischetto da formattare non si deve inserire «df0:» o «df1:» ma semplicemente «0» od «1». Ho adottato questa soluzione perché la trovo molto più pratica, a chi non va può sempre modificare il programma ed utilizzare le parole che più ritiene opportune.

Syntax si limita a visualizzare il classico messaggio «USAGE» (cioè i parametri richiesti dal programma per il suo funzionamento).

OpenTrackDisk crea le porte necessarie ed apre il trackdisk.device e preleva il CHANGENUM del disco. Ciò significa che deve essere chiamata DOPO che il disco su cui si vuole operare è stato inserito.

Formatdisk è la funzione che provvede alla formattazione vera e propria del disco.

Le variabili «byte» ed «lbyte» sono dei puntatori alla zona di memoria utilizzata come buffer; la loro differenza è che la prima restituisce i valori contenuti nel buffer sotto forma di byte, mentre la seconda restituisce valori sotto forma di long word. Come si vede dal listato, viene innanzi tutto scritto il blocco zero con la parola chiave DOS (vedi paragrafo «Come far riconoscere all'Amiga un disco...»), poi grazie ad un ciclo for... next si formattano gli 80 cilindri, si scrive il ROOT-BLOCK con gli identificatori di tale blocco, si scrive il puntatore alla Free Space (XFormat utilizza come Free Space il blocco n. 881, cioè 0371 in hex.), si scrive il nome del disco (a questo proposito voglio dirvi che il nome deve essere preceduto dal numero di caratteri da cui è composto, pena la sua mancata lettura al momento del «validating» del disco al suo inserimento), ed infine si scrive la corretta Free Space.

Al lancio del programma dovete inserire il numero del drive da usare ed il nome del disco, pena un rimprovero da parte di XFormat ed un messaggio che illustra la corretta sintassi da utilizzare. Durante la formattazione appaiono due simboli: «FF» indica che il cilindro è stato formattato correttamente; «??» indica invece che il comando non ha potuto formattare correttamente quel determinato cilindro.

Riguardo la compilazione del programma non c'è nulla da dire: nessuna opzione è necessaria, mentre è sufficiente collegare le librerie «amiga.lib» ed «lc.lib» tramite il linker (es. BLINK).

Il compilatore da me utilizzato è la versione 5.0, comunque non credo che si possano avere problemi utilizzando le versioni precedenti... o successive.

Conclusioni

State calmi, lo so che vi avevo promesso delle idee, ma almeno datemi il tempo di raffreddare i polpastrelli! Ma prima devo dire una cosa...

In MC di dicembre 1989 (n. 91) nell'articolo dedicato al calcolo del Checksum di un settore (pag. 260), il sig. Giuliano Peritore afferma che il Checksum si calcola sottraendo al valore $FFFFFFFF tutte le long word costituenti il settore, eccetto quella contenente il Checksum. Invece, secondo i miei calcoli, il corretto valore a cui sottrarre tutte le long word, eccetto quella contenente il Checksum, è $00000000 (per il calcolo del Checksum del ROOT-BLOCK ho utilizzato questo valore con successo).

Riguardo alle possibili modifiche da apportare al programma sono le seguenti:

1) installare il disco. L'operazione è possibile in due sistemi diversi: il primo consiste in una normale installazione con un programma in L.M. prefissato; il secondo sistema è quello di dare la possibilità all'utente di costruirsi una libreria su disco di molti programmi per il BOOT-BLOCK e di poter decidere di volta in volta quale utilizzare per installare un dato disco;
2) formattare il disco con o senza DOS;
3) decidere quali cilindri si vogliono formattare;
4) verifica di una corretta formattazione di ogni cilindro (tipo il classico format incluso nel disco del workbench). Per verificare la formattazione vi consiglio di: formattare il cilindro; rileggerlo tramite il «comando» ETD_READ e controllare se la funzione ritorna qualche errore; in caso negativo, verificare se i valori letti corrispondono ai valori che desideravate scrivere in esso.

Sperando che vi siate divertiti (!) vi prometto di farmi vivo di nuovo tra qualche mese con un programma che, se riesco ad ultimare, vi permetterà di ovviare al piccolo (!) inconveniente sulla lentezza dell'Amiga nel mostrare una directory. A presto!

COMPUTER

XT UniSystem 699.000
CPU Nec V20 con clock 4.77/12 MHz,
cabinet baby con alimentatore 200W,
tastiera 101 tasti,
640 KB Ram espandibili a 1 MB,
controller disk drive,
1 disk drive a scelta 360 K o 720 K,
scheda video duale Hercules+CGA,
porta parallela Centronics,
coprocessore opzionale 8087.

286 UniSystem 1.295.000
CPU 80286 con clock 6/12 MHz,
cabinet baby con alimentatore 200 W,
tastiera 101 tasti,
1 MB Ram espandibili a 8 MB EMS,
controller AT interleave 1:1,
1 disk drive a scelta 1.2 MB o 1.44 MB,
scheda video duale Hercules+CGA,
porta parallela Centronics,
coprocessore opzionale 80287,
0 wait states.

386-SX UniSystem 1.695.000
CPU 80386-SX con clock 8/16 MHz,
cabinet baby con alimentatore 200 W,
tastiera 101 tasti,
1 MB Ram espandibili a 8 MB EMS,
controller AT interleave 1:1,
1 disk drive a scelta 1.2 MB o 1.44 MB,
scheda video duale Hercules+CGA,
porta parallela Centronics,
coprocessore opzionale 80387-SX,
0 wait states.

386 UniSystem 2.595.000
CPU 80386 con clock 20/25 MHz,
cabinet tower con alimentatore 220 W,
tastiera 101 tasti,
1 MB Ram espandibili a 8 MB EMS,
controller AT interleave 1:1,
1 disk drive a scelta 1.2 MB o 1.44 MB,
scheda video duale Hercules+CGA,
porta parallela Centronics,
coprocessore opzionale 80387,
0 wait states.
386 UniSystem 32K cache ... 3.450.000

Prezzi IVA compresa

Amiga Action Replay

Finalmente! Una potentissima cartuccia utility+freezer+trainer!
Inserita nella porta di espansione del vostro Amiga 500, permette di:

- congelare e salvare su disco un programma caricato in memoria, per poterlo ricaricare quando volete fino a 4 volte più velocemente
- trovare le "poke" necessarie per ottenere vite infinite nei vostri giochi preferiti
- modificare e cambiare gli sprites di un gioco, per creare simpatiche versioni personalizzate o usare gli sprites nei vostri programmi
- avvertire della presenza di qualsiasi virus in memoria o sui vostri dischetti, distruggendo tutti i virus conosciuti
- salvare schermate e musiche su disco come files IFF, per poterle elaborare dai vostri programmi preferiti
- rallentare lo svolgimento dei giochi fino al 20% della velocità originale, per aiutarvi negli schermi più complicati
- usare il più potente monitor-disassembler per Amiga, con completo controllo dell'hardware e dei suoi registri (anche quelli "write-only"), uno strumento preziosissimo per il debugging dei vostri programmi: screen editor, breakpoint dinamici, assembler/disassembler delle istruzioni Copper, disk I/O con possibilità di alterare parametri quali sync o lunghezza della traccia, calcolatrice, notepad, ricerca di immagini o suoni in tutta la memoria, modifica caratteri in memoria, altera i registri della CPU, ed altro ancora.

Amiga Action Replay originale con manuale *in italiano* a sole 179.000

HARDWARE

Espansione da 2 MB per A-500, si inserisce nello slot sotto la tastiera al posto della vecchia espansione da 512K, completa di clock in tempo reale e batteria tampone 450.000
Espansione da 2 MB esterna per A-500 o A-1000 ..... 799.000
Hard disk GVP Impact 40 MB per A-500 ..... 1.550.000
Hard card GVP 40 MB per A-2000 ..... 1.480.000
Hard card GVP 100 MB per A-2000 ..... 2.550.000
Velocizzatore 68030 GVP A-3001 ..... da 1.440.000

SOFTWARE ALGOSYSTEM
MAGAZZINO/FATTURAZIONE/
ANAGRAFE CLIENTI
A SOLE L. 272.000

Dischi Fish di pubblico dominio aggiornati al n. 240

ACCESSORI

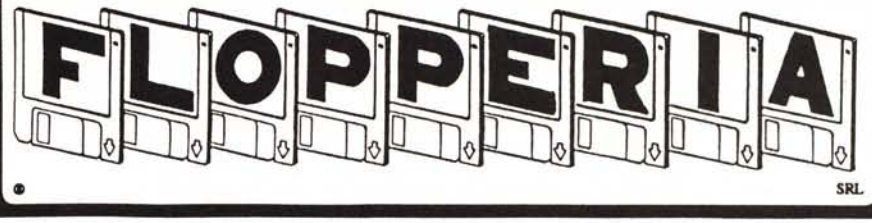
AMAS Sound Digitizer 299.000
Hard disk A-590 899.000
Espansione 2 MB per A-590 399.000
Mac-2-DOS con drive 950.000
Espansione 2 MB A-2000 799.000
DigiDroid 175.000
DigiView 4.0 450.000
Drive esterno con switch 179.000
Drive esterno TrackDisplay 259.000
Drive esterno 5"1/4 275.000
Flicker Fixer 950.000
Scanner A4 1.495.000

SYNCHRO EXPRESS
Eccezionale novità per Amiga: è finalmente disponibile il primo **copiatore hardware** per i dischetti Amiga! Con una speciale interfaccia collegata a 2 disk drives (quello interno al computer ed uno esterno), effettua copie di sicurezza, perfettamente funzionanti, di **qualsiasi software protetto** in meno di 50 secondi, compresi gli "impossibili" come Dragon's Lair.
89.000

FATTER AGNUS 8372-A
Il nuovo chip che permette di usare 1 MB di Chip Ram nel vostro Amiga, disponibile ora in kit di montaggio per l'installazione in tutti i modelli B-2000, ed A-500 (con piastra madre rev. 4 o 5) con inserita l'espansione A-501 da 512K.
159.000

Viale Monte Nero 31
20135 Milano
Tel. (02) 55.18.04.84
(4 linee ric. aut.)
Fax (02) 55.18.81.05 (24 ore)

Negozio aperto al pubblico tutti i giorni dalle **10** alle **13** e dalle **15** alle **19**.
Vendita per corrispondenza.
Sconti per quantità ai sigg. Rivenditori.



ELETTRONICA CENTOSTELLE s.r.l.

ZENITH	Lap top
TANDON	Desk top
ASEM	Desk top
NEC	Stampanti

Via Centostelle, 5/a - Firenze - Telefono (055) 61.02.51 - 60.81.07 - Fax 61.13.02

SOFTWARE

WORD PROCESSOR

Microsoft Word 5	it L.	712.000
Microsoft Word 5 euro	in L.	570.000
MicroPro Wordstar Prof. 5.5	it L.	590.000
MicroPro Wordstar Prof. 5.5	in L.	590.000
MicroPro Wordstar 2000 3.0	it L.	860.000
Lotus Manuscript 1.1	it L.	640.000
Lotus Manuscript 2.1	in L.	740.000
Ashton Tate Multimate adv. II	it L.	785.000
Ashton Tate Multimate adv. II	in L.	785.000
Ashton Tate Multimate 4.0	it L.	700.000
Ashton Tate Multimate 4.0	in L.	620.000
Ashton Tate Multimate LAN	it L.	1.500.000
Borland Sprint	in L.	330.000
Word Perfect 5.1	in L.	590.000

SPREADSHEET INTEGRATI

Microsoft Excel 2.1	it L.	712.000
Microsoft Excel Euro	in L.	640.000
Microsoft Excel 2.1 con Q+E	it L.	750.000
Microsoft Excel 2.1 Os/2	it L.	712.000
Microsoft Works	it L.	280.000
Microsoft Works	in L.	252.000
Lotus 1 2 3 Ver. 2.2	it L.	712.000
Lotus 1 2 3 Ver. 3.0	it L.	830.000
Lotus 1 2 3 Ver. 3.0	in L.	745.000
Lotus Symphony 2.0	it L.	840.000
Ashton Tate Framework III	it L.	850.000
Borland Quattro 1.0	it L.	320.000
Borland Quattro Profes. 2.0	it L.	720.000
Computer Ass. Supercalc 5	it L.	800.000

DATA BASE MANAGEMENT

Ashton Tate dBase III plus	it L.	880.000
Ashton Tate dBase IV 1.1	it L.	980.000
Ashton Tate dBase IV Dev. Ed.	it L.	1.850.000
Ashton Tate Rapid file	it L.	560.000
Borland Paradox	it L.	1.035.000
Borland Paradox (os/2)	it L.	1.240.000
Borland Paradox 386	it L.	1.240.000
Borland Reflex 2.0	it L.	340.000

DESKTOP PUBLISHING

Ventura Publisher	it L.	1.480.000
Fonts Bitstream	it L.	550.000
Ashton Tate Byline	it L.	472.000

AMBIENTI OPERATIVI

Microsoft Project 3.0	it L.	760.000
Microsoft Project 4 Euro	in L.	680.000
Microsoft Windows 286	it L.	180.000
Microsoft Windows 386	it L.	280.000
Microsoft Windows 286 toolkit	in L.	680.000
Lotus Agenda	in L.	650.000

LINGUAGGI

Microsft Quick basic 4.5	in L.	145.000
Microsoft Quick C compiler	in L.	145.000
Microsoft Basic Compiler 6.0	in L.	380.000
Microsoft C Compiler 5.1	in L.	590.000
Microsoft Fortran Compiler	in L.	630.000
Microsoft Cobol Compiler V3	in L.	1.100.000
Microsoft Macro Assembler	in L.	240.000
Microsoft Pascal Compiler	in L.	550.000
Microsoft OS/2 toolkit	in L.	480.000
Borland turbo Pascal 5.5	it L.	240.000
Borland turbo basic	it L.	170.000
Borland turbo C 2.0	it L.	240.000
Borland turbo Prolog. 2.0	in L.	230.000
Borland turbo Assembler/debug	it L.	230.000
Borland turbo C professional	it L.	399.000
Borland turbo Pascal Profess.	it L.	399.000
Microsoft word per windows	in L.	780.000
Corel Draw! 1.1	in L.	850.000
Superbase 4	in L.	890.000
RM Cobol 85	in L.	2.765.000
RM Cobol Compiler	in L.	1.660.000
RM Fortran	in L.	1.405.000

UTILITIES

Norton Utilities	in L.	170.000
Norton Commander	in L.	170.000
PC Tools 6.0	in L.	200.000

GRAPHICS

Microsoft Chart 2	it L.	390.000
Microsoft Chart 3 euro	in L.	540.000
Lotus Freelance Plus	it L.	730.000
Paintbrush plus (per Wind.)	in L.	290.000
Gem Artline	in L.	1.350.000
Gem desktop publishers	in L.	650.000
Lotus GraphWriter II	it L.	723.000
Adobe Illustrator	in L.	1.390.000

SOFTWARE UPGRADE

Da DB III a DB IV	L.	400.000
Da Framework II a Framework III	L.	300.000
Aggiornamenti Quick Microsoft	L.	80.000

NOVITA'

Microsoft Quick Basic 4.5	it L.	195.000
Microsoft Quick Pascal 1.1	in L.	150.000
Microsoft Quick Pascal 1.1	it L.	195.000
Microsoft Quick MASM /c	in L.	290.000

AUTOCAD 10.0
per scuole ed università
L. 1.190.000

LEADER NEI COMPUTER PORTATILI
CONCESSIONARIO TOSHIBA

TOSHIBA tutti i modelli	Telefonare	
ZENITH 80286 HD 20MB	L.	3.493.000
80386 sx HD 40MB	L.	6.713.000
TANDON 80286 HD 20MB	L.	4.392.000
80386 sx HD 40MB	L.	5.432.000

SCANNER

LOGITECH SCANMAN PLUS PC	L.	408.000
LOGITECH SCANMAN PLUS PC + IMAGE IN	L.	790.000
LOGITECH SCANMAN PLUS PC + FINESSE 3.0	L.	720.000
TRACKMAN	L.	199.000
LOGITECH MOUSE + PAINT SHOW	L.	192.000

COPROCESSORI MATEMATICI
a basso consumo per PORTATILI

	IVA esclusa	IVA compresa
80C287/8	L. 480.000	L. 571.000
/10	L. 547.000	L. 651.000
/12	L. 675.000	L. 803.000
80C387/16	L. 840.000	L. 990.000

COPROCESSORI MATEMATICI INTEL

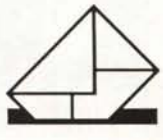
80287/8	L. 389.000	L. 463.000
/10	L. 439.000	L. 522.000
/16	L. 649.000	L. 772.000
/33	L. 1.160.000	L. 1.380.000

Confezioni originali

DIRETTAMENTE A CASA VOSTRA:
SOFTWARE E HARDWARE AI MIGLIORI PREZZI

PREZZI IVA ESCLUSA - PAGAMENTO CONTRASSEGNO, VISA - SPESE POSTALI L. 10.000

Consulenze gratuite, informazioni, ordini e conferme prezzi sulla nostra Hot Line Tel. 055/610251-608107



Ordini a mezzo poste :
Elettronica Centostelle
Via Centostelle 5/a
50137 Firenze

o tramite Fax
al numero
055/61.13.02