coordinamento di Andrea de Prisco

OCCAM: parallelismo «esplosivo»

di Luciano Macera

Il nostro viaggio all'interno di OCCAM continua questo mese con un argomento che lascerà, i più, sconcertati. Parleremo del cosiddetto «parallelismo esplosivo», eufemismo non troppo scientifico che utilizziamo per indicare quella capacità di OCCAM di generare processi paralleli a tempo di esecuzione ed in numero noto solo in quel momento. Vedremo dunque come con poche righe di codice si riescono ad implementare meccanismi ad elevato parallelismo e dalla potenza a dir poco impressionante

L'augurio

OCCAM è, fortunatamente per certi versi, purtroppo per altri, più potente dei transputer sui quali gira. In hardware è stato sì possibile implementare tutte le funzionalità richieste, ma alcune di queste, per ovvie ragioni, con qualche limitazione.

Ci riferiamo essenzialmente al parallelismo dei programmi OCCAM che, lanciati sul singolo transputer sono eseguiti «solo» in parallelismo simulato, mentre per un'esecuzione su rete di transputer è necessario allocare staticamente, sui più chip, i vari processi.

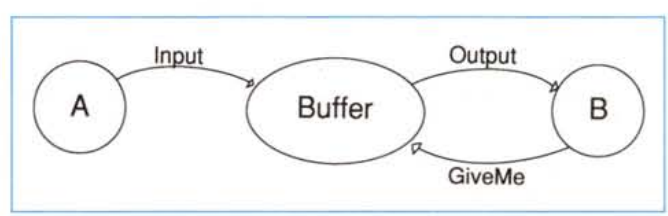
I link fisici, infatti, sono utilizzati essenzialmente per scambiare messaggi tra processi in esecuzione su nodi di elaborazione diversi, ma non per far migrare a tempo di esecuzione processi da eseguire parallelamente. Questo sicuramente a causa del fatto che, sempre attualmente, l'overhead dovuto a questa sorta di migrazione di processi sarebbe troppo oneroso e magari (anzi sicuramente) annullerebbe i vantaggi della parallelizzazione reale.

Ma... c'è un «ma». Anche se pieno zeppo di fantaSCIENZA. Quando i progettisti della INMOS hanno dato un nome alla loro creatura hanno scelto la parola transputer. Fusione di altri due termini molto noti che sono «transistor» e «computer». Ciò ad indicare che il «chippetto» è un computer (e anche buono) ma nasce per essere utilizzato come una specie di transistor ovvero come mattoncino «materia prima» di qualcosa di ben più potente del transputer stesso: la rete di transputer. Questa, ha dalla sua il fatto di poter essere estesa infinitamente e con essa la potenza di calcolo che può offrire. L'analogismo è praticamente perfetto: come con i transistor possiamo ipotizzare (e realizzare...) circuiti elettronici complicatissimi formati da decine, migliaia, milioni di semiconduttori, con i transputer è possibile costruire sistemi di calcolo a parallelismo massiccio della potenza desiderata, senza limiti né teorici né tecnici.

E fin qui non stiamo ancora sognando. Ma proviamo a spingere la nostra fantasia oltre. Qual è stato il cosiddetto «passo successivo» per il mondo dei transistor ? La risposta è semplice: il circuito integrato. Un «pezzetto di silicio» sul quale vengono «intarsiati» migliaia, oggi anche milioni, di transistor per concentrare in pochissimo spazio circuiti elettronici dalle dimensioni e costi svariati ordini di grandezza superiori.

E chi ci vieta di augurare al transputer la stessa sorte? O forse non è facile ipotizzare intere reti di transputer sul singolo chip? Magari prima 8, 16, 32 unità sullo stesso silicio ma più in là (quando probabilmente non si tratterà più di silicio...) migliaia o anche milioni di unità di calcolo pronte ad assecondare, e quindi a soddisfare, le già citate «esplosioni» di parallelismo possibili in OCCAM.

Figura 1



```
[100]CHAN OF INT coda: -- si dichiara un array di canali di interi
                       -- ogni processo lanciato utilizzera' un
                       -- canale di ingresso e un canale d'uscita
                       -- di quest'array implementando una pipeline
PAR
  PAR i=0 FOR 100        -- lancio 100 volte il seguente processo
    WHILE TRUE
      INT x:
      SEQ
        coda[i  ] ? x          -- leggo il valore da bufferizzare
        coda[i+1] ! x          -- spedisco il valore da bufferizzare

  INT x:                       -- primo processo del buffer che
  SEQ                          -- interagisce col processo mittente
    input ? x
    coda[0] ! x

  INT x:                       -- ultimo processo del buffer che
  SEQ                          -- interagisce col processo destinatario
    coda[99] ? x
    output ! x

:
```

Figura 2

Un buffer parallelo

Sul numero scorso di MC, quale primo esempio di programmazione OCCAM abbiamo mostrato come risolvere il problema della sincronicità delle comunicazioni interponendo tra processo mittente e processo destinatario un processo Buffer. All'interno di questo (figura 1) i messaggi inviati dal processo mittente venivano parcheggiati in un array di messaggi (utilizzato circolarmente) in attesa che questi venissero richiesti dal processo destinatario. Per utilizzare, infatti, un comando alternativo con guardie d'ingresso era necessario che il processo destinatario inviasse una richiesta al buffer per ricevere un nuovo messaggio in arrivo. Quindi la soluzione, per quanto valida dal punto di vista OCCAM, aveva il piccolo difetto di rendere necessaria anche una lieve modifica al processo destinatario prima di utilizzare il buffer vero e proprio.

La soluzione che mostreremo questo mese sposa le caratteristiche di OCCAM in maniera più spinta utilizzando il già citato «parallelismo esplosivo» ed avendo anche il vantaggio di non richiedere alcuna modifica al processo destinatario che vedrà arrivare i suoi messaggi esattamente come se non esistesse alcun buffer tra esso e il processo mittente.

In figura 2 è mostrato il listato del nuovo programma buffer, in figura 3 sono rappresentati i processi coinvolti una volta lanciato il processo buffer. Vediamo come funziona. Quello che succede è molto semplice: invece di utilizzare un array di messaggi in cui depositare gli arrivi e prelevare le partenze utilizzando un unico canale di ingresso ed un altrettanto unico canale di uscita, si utilizza direttamente un array di diciamo 100 canali utilizzati da altrettanti processi creati dal processo buffer. Ognuno di questi processi figli non farà altro che ricevere un messaggio dal canale (i)esimo e rispedire immediatamente il messaggio ricevuto sul canale (i+1)esimo. Pensate, ad esempio, ad un certo numero di operai che scaricano, manualmente, un camion di mattoni passandoseli di mano in mano dal cassone sino al punto dove è necessario stiparli. L'operaio «mittente» sarà quello posizionato sul camion, l'operaio «destinatario» quello che posiziona uno dopo l'altro tutti i mattoni ricevuti. Tutti gli operai «intermedi» bufferizzano l'operazione così il mittente può iniziare a «spedire» anche se il destinatario non è pronto essendo costretto ad arrestarsi solo nel caso in cui il destinatario non riceve e tutti gli operai sono rimasti con un mattone in mano.

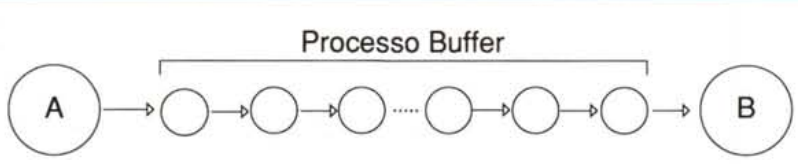


Figura 3

Nel listato di figura 2 troviamo innanzitutto la dichiarazione dell'array di canali necessari, come detto, alla bufferizzazione. Segue un primo PAR che lancia in parallelo i tre processi in esso contenuti: il primo è in buffer vero e proprio che commenteremo tra breve, il secondo e il terzo l'interfaccia di ingresso e d'uscita del buffer che dialogano con i veri processi «A» e «B» che devono comunicare. Corrispondono, in pratica, ai due operai citati precedentemente: il primo sul camion il secondo nel luogo dove occorre stipare i mattoni in arrivo.

Per quanto riguarda il buffer vero e proprio, notiamo subito che esso è formato da un «replicator» applicato al costrutto PAR. Come spiegato due numeri fa con questo meccanismo è possibile creare un numero qualsiasi di processi figli ognuno dei quali identificato da un ben preciso indice della variabile di ciclo. Come i «FOR» dei vari linguaggi sequenziali con la «semplice» differenza che le varie istanze di ciclo non sono più eseguite sequenzialmente ma tramutate in altrettanti processi e in quanto tali eseguiti parallelamente. Ognuno dei processi creati (sempre figura 2) è un loop infinito (WHILE TRUE) su una coppia di operazioni che prelevano il messaggio in arrivo sul già citato canale (i)esimo e spediscono il messaggio così ricevuto sul canale (i+1)esimo.

Sorting parallelo

L'esempio appena visto funziona sì, ma crediamo che mai a nessuno venga in mente di implementarlo per davvero. Si tratta, infatti, del monumento allo spreco e, per questo, anche all'inefficienza. Quindi didatticamente validissimo ma ben ancorato a lavagne e pubblicazioni varie che, come noi, devono solo mostrare l'essenza di OCCAM. Ha infatti la caratteristica di essere sì particolarmente parallelo ma utilizza questo parallelismo non per migliorare le prestazioni del sistema mittente-destinatario ma solo per semplificare al massimo la stesura del programma. Pensate al fatto che un messaggio, nel caso di buffer vuoto e destinatario pronto a ricevere, prima di arrivare a destinazione deve «attraversare» più di cento processi (quante sono le posizioni del buf-

fer utilizzato) con altrettante operazioni di send e receive da eseguire. Non ve l'abbiamo confidato subito per non togliervi il gusto di studicchiarvi con noi il listato di figura 2.

Il prossimo esempio è, però, molto più serio. Discendente diretto del programma appena visto, ha l'importante proprietà di sfruttare il parallelismo nel giusto verso: incrementare la potenza di calcolo per ottenere in minor tempo i risultati voluti. È un semplice programma di ordinamento numerico che ha la particolare caratteristica di avere un tempo di computazione eccezionalmente basso, dovuto al fatto che grazie al parallelismo dei processi più operazioni vengono effettuate contemporaneamente.

Vedete a cosa servirebbe, subito, un chippetto con un milione di transputer dentro ? Ad ordinare, al volo, fino ad un milione di dati.

Già, i più svegli avranno già capito, è necessario creare un processo per ogni dato da ordinare: solo così, con un funzionamento pipeline del sorting è possibile ridurre drasticamente il tempo di esecuzione.

Come detto prima il funzionamento è simile a quello del buffer appena descritto. Ogni processo figlio ha però al suo interno una variabile temporanea nella quale mantiene il dato appena letto per confrontarlo col dato successivo. Di volta in volta leggerà un dato e con-

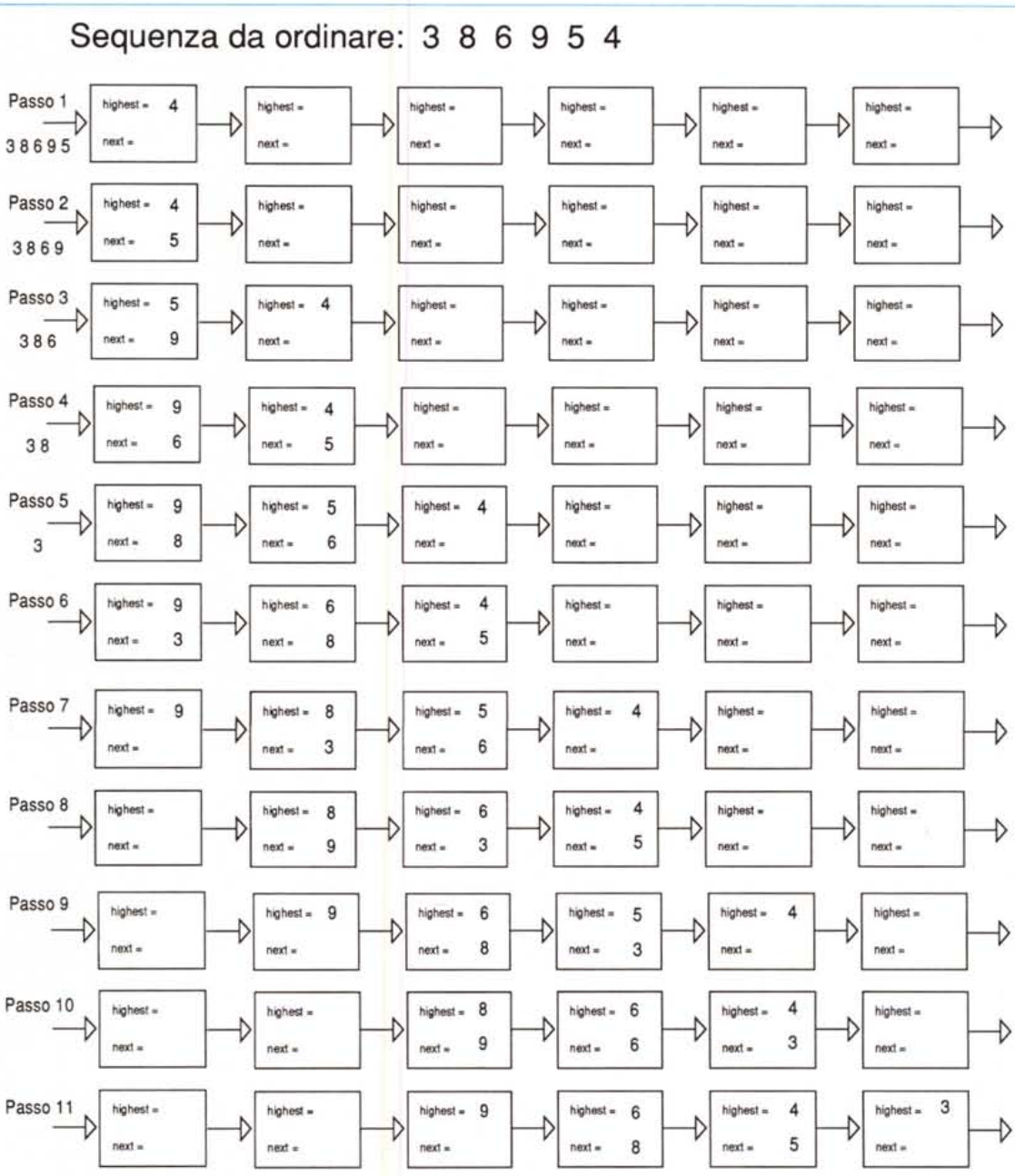


Figura 4

Figura 5A

```
INT highest:          -- variabile per il valore piu' alto ricevuto
SEQ
  input ? highest     -- il primo valore inizializza highest
  SEQ j=0 FOR 5       -- per i rimanenti 5 valori...
    INT next:
    SEQ
      input ? next          -- leggo il prossimo valore
      IF
        next <= highest     -- se e' minore di highest
          output ! next     -- lo inoltro al processo successivo
        next > highest      -- se e' maggiore di highest
          SEQ
            output ! highest  -- inoltro quest'ultimo
            highest := next   -- e assumo un nuovo highest
  output ! highest  -- per finire inoltro il "vero" highest
:
```

Figura 5B

```
[101]CHAN OF INT pipe: -- si dichiara un array di canali di interi
                       -- ogni processo lanciato utilizzera' un
                       -- canale di ingresso e un canale d'uscita
                       -- di quest'array implementando una pipeline

PAR i=0 FOR 100        -- lancio 100 volte il seguente processo
  INT highest:         -- variabile per il valore piu' alto ricevuto
  SEQ
    pipe[i] ? highest  -- il primo valore inizializza highest
    SEQ j=0 FOR 99     -- per i rimanenti 99 valori...
      INT next:
      SEQ
        pipe[i] ? next         -- leggo il prossimo valore
        IF
          next <= highest      -- se e' minore di highest
            pipe[i+1] ! next   -- lo inoltro al processo successivo
          next > highest       -- se e' maggiore di highest
            SEQ
              pipe[i+1] ! highest  -- inoltro quest'ultimo
              highest := next      -- e assumo un nuovo highest
    pipe[i+1] ! highest  -- per finire inoltro il "vero" highest
:
```

frontandolo col dato immagazzinato propagherà quello più alto o più basso tra i due a seconda se l'ordinamento è ascendente o discendente. Un po' come se gli operai di prima scaricassero pietre dal camion con la raccomandazione di far giungere all'ultimo della coda le pietre in ordine di grandezza. Certo, ci vorrebbero tanti operai quante sono le pietre, ma raggiungerebbero il loro scopo in un batter d'occhio e soprattutto senza fare confusione: ogni operaio la prima pietra che riceve la conserva in tasca e man mano che passano le altre pietre esegue un rapido test ponendole una per una nella tasca attualmente vuota e cedendo al successivo la più leggera tra le pietre nelle due tasche.

Tornando alla realtà, in figura 4 è mostrato un esempio «grafico» del procedimento di sort parallelo (pipeline per la precisione) applicato ad una sequenza di sei interi da ordinare. Saranno necessari 6 processi (il cui codice OCCAM è listato in figura 5A) ognuno dei quali proclamerà, in al più cinque passi il massimo elemento transitato al suo interno. Naturalmente, dato che il procedimento si svolge «in parallelo» in ogni passo saranno effettuati contemporaneamente più confronti all'interno di processi differenti ma eseguendo questi simultaneamente comunque utilizzeranno il tempo necessario ad un singolo confronto.

Seguiamo quello che succede in figura 4. I dati da ordinare sono:

3 8 6 9 5 4

con in testa il 4 e in coda il 3. Al primo passo il 4 è inoltrato al processo uno che, ricevendo il suo primo dato, lo pone d'uffico nella sua variabile temporanea «highest». Al passo due anche il 5 giunge al processo uno potendo così effettuare il primo confronto tra i primi due dati. Naturalmente 4 è più piccolo di 5 quindi il primo è spedito al processo due (che, come prima, lo archivia d'ufficio) mentre il secondo diventa il nuovo «highest» del processo uno. Al terzo passo «entra» il 9 che prende il posto del 5 che viene spedito al processo due mentre arriva anche un nuovo dato, il 6. Al passo quattro è così possibile effettuare contemporaneamente due confronti, all'interno del processo uno e del processo due. Il procedimento «pipeline» si ripete per tutti i rimanenti numeri da ordinare all'interno di tutti i processi da attraversare. Notate, già al passo sei, la possibilità di effettuare tre confronti contemporaneamente (cioè tutti i dati coinvolti simultaneamente in un confronto). All'uscita dell'ultimo processo i dati saranno in ordine crescente e cioè nell'esatta sequenza:

9 8 6 5 4 3

Per finire diamo uno sguardo al programma di figura 5B. È la versione definitiva del sort parallelo appena descritto e mette in ordine 100 numeri. Naturalmente per sort più consistenti è sufficiente aumentare tale valore nella dichiarazione iniziale, nel PAR successivo e nel SEQ subito dopo. Highest, come detto prima, è la variabile che contiene costantemente il più grande numero passato all'interno del processo. La seconda variabile, next, contiene il dato in transito da confrontare con highest. Quindi il processo (i)esimo preleva dal canale (i)esimo un dato depositandolo momentaneamente nella variabile next, confronta questo con il contenuto di highest, se è più piccolo spedisce sul canale (i+1)esimo il valore di next se è più grande quello di highest assegnando poi a questo il dato trattenuto. Quando tutti i dati sono passati, non resta che inoltrare anche highest che, a quel punto, conterrà il numero più grande dell'intera sequenza ricevuta.

MS

EXpress Office Automation

I Cash&Carry dell'Informatica

ROMA
Via Tenuta di Torrenova, 28
TEL/FAX : 06 / 2040041

MILANO
Via Mecenate, 76/4
TEL/FAX : 02 / 58010800

TORINO
Via Umberto Giordano, 5/A
TEL : 011 / 2473160
FAX : 011 / 2473137

PARMA
Via Buffolara, 68
TEL: 0521 / 290517
FAX: 0521 / 96412

REGGIO EMILIA
Via Umbria, 10
TEL : 0522 / 512751
FAX : 0522 / 513129

FOGGIA
Via Vittime Civili, 66/A
TEL/FAX : 0881 / 694412

PROSSIME APERTURE
MODENA, BOLOGNA, FERRARA
MESSINA, CATANIA, TRAPANI
PESCARA, NOVARA, ...

Siamo presenti a:
MILANO - dal 3 al 7 Ottobre allo SMAU 91
al Padiglione: 15/II Stand: G08

P.C. EXPRESS - 286 Sistema ENTRY

CPU 286 a 16Mhz*	1 Mbyte RAM	Tastiera Italiana 101 Tasti
1 Porta Joystick	40 Mb HD Fast	FDC & IDE Controller
HGC 720 x 348 2 col	Drive 3" ½ 2Mb	Mouse 3 tasti bi-standard
Cabinet Desktop	1 porta Stampante	2 porte RS-232C
	L. 990,000	
Con VGA aggiungere:	L. 100,000	*MS-DOS 5.0 TESTED*

P.C. EXPRESS - 386 Sistema ENTRY

CPU 386 a 27Mhz*	1 Mbyte RAM	Tastiera Italiana 101 Tasti
1 Porta Joystick	40 Mb HD Fast	FDC & IDE Controller
HGC 720 x 348 2 col	Drive 3" ½ 2Mb	Mouse 3 tasti bi-standard
Cabinet MiniTower	1 porta Stampante	2 porte RS-232C
	L. 1,999,000	
Con VGA aggiungere:	L. 100,000	*MS-DOS 5.0 TESTED*

P.C. EXPRESS - Sistema ENTRY per CAD

CPU 386 a 40Mhz*	2 Mb RAM SIMM	Tastiera Italiana 101 Tasti
1 Porta Joystick	40 Mb HD Fast	Cabinet Tower Medio
VGA 1024 x 768 16 col	Drive 3" ½ 2Mb	Mouse 3 tasti 2600 dpi
2 porte RS-232C	1 porta Stampante	FDC & IDE Controller
	L. 2,499,000	
Con 387 aggiungere:	L. 500,000	*MS-DOS 5.0 TESTED*

* secondo il LandMark Speed 1.14

MAINBOARD - CABINET - DRIVE - I/F VIDEO

Cabinet Mini Tower con 200W	199,000
Cabinet Big Tower con 200W	337,000
Monitor 14" HGC 720 x 348 2 colori	199,000
Monitor 14" VGA 640 x 480 Monocromo	233,000
Monitor 14" VGA 1024 x 768 16 colori	570,000
Piastra Madre 386 a 28Mhz* # Omaggio Licenza DOS 5.0 per ogni Piastra	633,000
Piastra Madre 386 a 40Mhz*	850,000
Piastra Madre 386 SX a 16Mhz*	478,000
VGA 1024 x 768 16 colori a 16 bit	186,000
Co-Proc. Matematico 80387 per CAD	509,000
Joystick Professionale a lunga durata	27,000

fino ad esaurimento scorte

PIU' DI 500 ARTICOLI A VOSTRA DISPOSIZIONE PER PICCOLE E GRANDI ESIGENZE

EXpress Office Automation

I prezzi indicati si intendono IVA esclusa
Sconti speciali per operatori di settore
Richiedete il listino e le condizioni per rivenditori

è una società del Gruppo PRIMA

LINEA AMIGA COMMODORE

GLI HARD DISK

IMPACT A2000 GVP II SERIE - HD Controller SCSI più esp. RAM 2-8 Mb per A2000 o ZBB con autoboot. 0 Kb **390.000** - 2Mb **590.000**

TOPCARD HARDITAL -
Controller Hard Disk SCSI per A2000. Autoboot autoconfig. Tutte le partizioni sono FFS e bootabili. Costruito complet. in tecnologia VLSI **240.000**

A2091 COMMODORE - Controller HD SCSI per A2000. Autoboot. Autoconfig.Con possibilità di inserire 2Mb di RAM. **290.000**

Con 2 Mb **490.000**
Hard Disk SCSI - 40 MB 3,5" 11ms-Quantum **470.000**
Hard Disk SCSI - 52 MB 3,5" 11ms-Quantum **540.000**
80 MB 3,5" 11ms-Quantum **820.000**
105 MB 3,5" 11ms-Quantum **910.000**
120 MB 3,5" 11ms-Quantum **1.080.000**
210 MB 3,5" 11ms-Quantum **1.440.000**

DOTTO HARDITAL - Micro HD controller in standard IDE AT BUS interno x A500/1000/2000 utilizza HD sia da 2,5 (7x11 cm) che da 3,5 - autobut e autoconfigurante - tutte le partizioni FSS **150.000**

HD consigliati - il PRATRIETER 20Mb - 23Ms **690.000**
SEAGATE 43 Mb 28Ms 3,5 inch **390.000**
QUANTUM 52Mb 11Ms 3,5 inch **540.000**
QUANTUM 105Mb 11Ms 3,5 inch **820.000**

A590 COMMODORE - HD controller più HD da 20 Mb con esp. Ram da 0 a 2Mb autoboot per A500 **690.000;** Con 2Mb di Ram **890.000**

HD2000 card - Controller e HD su scheda per AMSTRAD, IBM/XT o A2000 con Janus.HD2000 card 32Mb **490.000**

GLI EMULATORI MS-DOS

AT ONCE VORTEX 286 - Emulatore IBM/T per A500. Contenente la CPU 286 a 8MHz. Si inserisce all'interno del computer **359.000**

ADATTATORE PER AMIGA 2000 **120.000**

JANUS XT - Emulatore IBM/XT per A2000 + drive da 5,1/4 con garanzia Commodore Italia **580.000**

JANUS AT - Emulatore IBM/AT per A2000 + drive da 5,1/4 con garanzia Commodore Italia **849.000**

LE ESPANSIONI DI MEMORIA

MEGA AGNUS - Espansione di memoria da 2Mb di chip RAM interno per A2000 e A500 necessita del Fater Agnus 8373 **360.000**

AGNUS 8373 **220.000**

XPANDER - Esp. di memoria da 2Mb per A500/1000 di tipo slim. Esterna 0 Wait State. Munita di interruttore di disinserimento. Autoconfigurante. Dimensioni 13X10X2,5cm. **349.000**

AMINTERAM - Espansione di memoria per A500 da 512Kb. Si inserisce nell'apposito slot del computer **59.000;** Con orologio e batteria tampone **74.000**

INSIDER 2 HARDITAL - Esp. di mem. da 2Mb per A500. Esp. la memoria a 2,3 Mb nei computer con i vecchi Agnus e a 2,5Mb in quelli con i nuovi BIG Agnus, di cui 1 Mb come chip RAM e 1,5 come fast RAM. Si inserisce nell'apposito slot del computer. Con orologio e batteria tampone. **259.000**

INSIDER 4 HARDITAL - Come sopra ma da 4 Mb **390.000**

INSIDER 6 HARDITAL - Come sopra ma da 6Mb **520.000**

SUPEROTTO HARDITAL - Esp. da 0-2-4-8-Mb sulla stessa scheda per A2000 o ZBB. Con display con indicazione della memoria disponibile e di led di autoconfiguraz. Zero wait state. 2Mb **280.000** - 4Mb **480.000** - 8Mb **790.000**

SUPEROTTO HD HARDITAL - Come sopra ma con integrato un controller per HD in tecnologia SCSI. Con 2 MB **490.000**

A2058 COMMODORE - Espansione da 2 a 8 Mb per A2000, 2Mb **850.000**

KICKROM 1,3 A1000 -
Kickstart 1.3 su Eprom senza saldature per A1000 con orologio tampone.Si inserisce sul connettore laterale del computer. Con connettore passante **149.000**

KICKROM 1.3 A500/A2000 - Kickstart 1.3 su Eprom interno per A500/2000. Con deviatore per Kickstart 1.2 **89.000**

KICKROM 2.0 - Kickstart 2.0 su Eprom interna per Amiga **140.000**

I DRIVE

ADRIVE - Drive da 3,5" esterno per A500/1000/2000. Con interruttore per il disins. e di connettore passante **119.000**

ADRIVE 2000 - Drive interno da 3,5" per A2000 **99.000**

SUPER DRIVE HARDITAL - rivoluzionario drive da 3,5" esterno per A 500/1000. Comprende un tasto plagio per duplicare qualsiasi programma, un tasto A.V. per impedire la contaminazione di virus sul dischetto, e di un tasto per disattivare il drive. Completo di connettore passante. **159.000**

ACCELERATORI - PROCESSORI - COPROCESSORI

BANG 2081/82 Hardital - Scheda accell. per A500/A2000 contenente 68020 a 16 MHz e 68881 **490.000;**

Per configuraz. con 68881 o 68882 con altre frequenze **Chiedere**

BIG BANG HARDITAL - Scheda accelleratrice contenente il 68030 e il 68882 per A 500/2000 con clock asincrono da 16 a 60 Mhz.
Burst mode design completa di memoria autoconfigurante a 32 bit da 1 a 8 MB. Switchabile tra 68000 e 68030. Rimappatura del Kikstart su RAM a 32 bit.
Caratteristica unica è che la memoria viene vista completamente anche in modo 68000. Attualmente è la più moderna e potente scheda accelleratrice del mercato. Completa di 68030 e 68882 a 25 Mhz e 2MB di RAM **990.000**

Con 4MB **1.390.000**
Con 8mb **1.790.000**
Per configurazione a 30, 37, 55 Mhz **Chiedere**
A 37 Mhz **1.690.000**
a 55 Mhz **1.990.000**

GVP 3001 - Scheda accell. per A2000 con 68030 e 68882 a 28 Mhz.Controller HD ed esp. di mem. a 32 bit da 4 Mb espandibili a 8. **2.690.000**

PROCESSORI: 68010 **29.000** - 68020 **190.000** - 68030 **290.000**

COPROCESSORI: 68881 16 Mhz **190.000;** 68882 16 Mhz **290.000**

A2630 - Scheda accelleratrice contenente 68030 e 68882 a 25Mhz + RAM a 32 bit da 2 Mb **1.760.000;** Con 4mb **2.050.000**

TURBO BOARD II SERIE con memoria da 1Mb
Espandibile a 16 - **Chiedere**

RAM X A 3000 1Mbx4 **49.000**

I DIGITALIZZATORI AUDIO VIDEO

GENLOCK CARD A 2300 Commodore -
Scheda Genlock semiprofessioale per Amiga 2000 **390.000**

FLICKER FIXER - Scheda da inserire nello slot video dell'A2000 ed elimina il flicker. **370.000**

FLICKER FIXER + Monitor Multisync **990.000**

I MONITORS

COMMODORE 1084 S -Monitor HiRes stereo per A500-A1000-A2000 **450.000**

PHILIPS 8833 - Monitor stereo per Amiga o PC **425.000**

LE STAMPANTI

STAR LC 10 **380.000**
STAR LC 10 color **450.000**
STAR LC 24-10 - 24 aghi 150 cps NLQ **610.000**
COMMODORE MPS 1230 **330.000**
COMMODORE MPS 1550 C COLOR **410.000**

I COMPUTER

COMMODORE CDTV **1.099.000**
AMIGA 500 - Con mouse, manuali e garanzia Commodore Italia **629.000**
Come sopra ma con espansione da 1Mb **675.000**
Come sopra ma con espanzione da 2.5 Mb **849.000**
AMIGA 2000 - Con mouse e manuali con garanzia Commodore Italia **1.340.000**
Come sopra più espansione da 2Mb **1.590.000**
+ HD autoboot SCSI QUANTUM 52 Mb da 2Mb **2.340.000**
AMIGA 3000 **Chiedere**
DISCHETTI SONY, BULK, DD-DS DA 3,5" -1 L.**790**- 10 L.**690**- 100 L.**640**- 1000 L.**560**

I PERSONAL COMPUTER IBM COMPATIBILI

HAR 286/16 - CPU 286/16Mhz 0 WAIT STATE - 1 Mb RAM - 2 seriali 1 parallela - Case Baby AT Alim. 200 W - Controller per floppy e host adapter per AT-BUS - 1 drive da1,44 mb - scheda VGA 800X600 - tast. est. 102 tasti **790.000**

Come sopra ma con CPU 286 da 20 MHz **840.000**

HAR 386 - CPU 386/16MHz 0 WAIT STATE - 1Mb RAM - 2 seriali 1 parallela - Case Baby AT alimentatore 200 W - Controller per floppy e Host Adapter per AT-BUS - 1 drive da 1.44Mb - scheda VGA 800X600 - tastiera estesa 102 tasti **1.040.000**

Come sopra ma con CPU 386 da 25 Mhz + 4Mb **1.540.000**
Come sopra ma con CPU 386 da 33 Mhz + 4Mb **1.890.000**

HAR 486 - CPU 486/25Mhz 0 Wait state - 4 Mb Ram - 2 seriali 1 parallela - Case Torre alim. 300 W - Controller per floppy e Host Adapter per AT Bus - 1 drive da 1,2 Mb - Scheda VGA 800X600 - Tastiera Estesa 102 tasti. **2.950.000**

NOTEBOOK - CPU 386/20 Mhz - LCD Display retro illuminato con risol. VGA 640X480 - 1 Mb Ram - 1 drive 3,5" - 1,44 Mb - con alim. batterie, borsa di trasporto HD 20 Mb **2.290.000**

Come sopra HD 40 Mb **3.490.000**

PERIFERICHE - DRIVE 1,2MB 5 1/4 **140.000**
COPROCESSORE MATEMATICO 287 10 Mhz **190.000**
COPROCESSORE MATEMATICO 387 20 Mhz **290.000**
COPROCESSORE MATEMATICO 387 25 Mhz 0A 33 Mhz **990.000**
HARD DISK SEAGATE ST124 21,4 Mb 3,5" ST- 412 60ms **320.000**
HARD DISK SEAGATE ST157A 43 Mb 3,5" AT-BUS 28ms **390.000**
HARD DISK SEAGATE 130 MB 3,5" AT-BUS 18ms **840.000**
HARD DISK SEAGATE ST 1239A 211Mb 3,5 AT-BUS 15ms **1.390.000**
MONITOR 14" MONO-DOPPIA FREQ.SCHERMO PIATTO **170.000**
MONITOR 14" MONO-VGA 640X480 **190.000**
MONITOR 14" COLOR MULTISYNC 1024X768 **349.000**
MONITOR 19" COLOR MULTISYNC 1024X768 **1.640.000**
SCHEDE VGA 256Kb 800X600 **120.000**
SCHEDE VGA 512 Kb 1024X768 **160.000**
SCHEDE VGA 1024 Kb 1024X768X256 COLORI **220.000**
PC NET RETE LOCALE A 1 MBits/s + acces. **290.000**
Handy scanner 200/300/400 dpi **290.000**
Handy scanner colori 200/300/400 dpi **890.000**
Digitaliz. 12"x12" comp. Summaskeck 1201 **440.000**
MOUSE GENIUS **40.000**

COMPUTER CENTER

PER INFORMAZIONI E/O ORDINAZIONI:
Via Forze Armate 260
20152 Milano - Tel. 02/4890213

HARDITAL
Show Room - Via G. Cantoni, 12
Milano - Tel. (02) 4983457-4983462

VENDITA SOLO PER CORRISPONDENZA
TUTTI I PREZZI SONO IVA COMPRESA