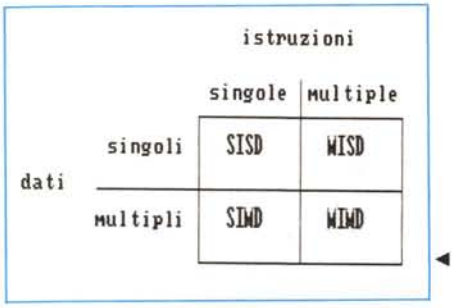
coordinamento Andrea de Prisco

Architettura e programmazione dei sistemi multiprocessore

di Giuseppe Cardinale Ciccotti

I sistemi multiprocessore sono le architetture parallele sicuramente più diffuse tanto che spesso tali sistemi vengono indicati come «i computer paralleli».
Come abbiamo visto, invece, esistono diverse classi di strutture parallele e i sistemi multiprocessori costituiscono una di queste. Una classificazione molto semplice, tuttavia per forza di cose non esaustiva, è quella fornita da Flinn riportata in figura 1.
In tale tassonomia i sistemi multiprocessore cadono nella classe dei MIMD computer.
La caratteristica fondamentale di tale classe è appunto quella di eseguire contemporaneamente istruzioni differenti fra loro su dati diversi. Tale possibilità consente ovviamente il massimo grado di flessibilità in quanto per sfruttare il parallelismo non si è più costretti (almeno in linea teorica) a organizzare logicamente e fisicamente i dati (e spesso anche la sequenza delle istruzioni).
Tuttavia i limiti fisici dell'hardware impediscono che questo concetto trovi reale rispondenza nelle capacità elaborative delle macchine reali, analizzeremo perciò tali problemi e le soluzioni adottate



◀ *Figura 1 - Tassonomia di Flinn. Le macchine di calcolo sono divise in quattro categorie, secondo il proprio modo di funzionamento.*

Strutture funzionali

I sistemi a multiprocessore, o multiprocessori, si distinguono dalle altre classi di architetture parallele per la caratteristica delle comunicazioni inter-PE. Abbiamo visto come negli Array Processor le comunicazioni inter-PE fossero un evento indesiderato e perciò il programmatore deve evitare un numero eccessivo di scambi di dati fra i PE, allocando i dati stessi in modo opportuno. Nei multiprocessori le comunicazioni sono invece parte peculiare del sistema stesso, tanto che quasi mai il programmatore deve preoccuparsi di gestire lo scambio di dati e, stavolta, anche di messaggi. I messaggi sono indispensabili poiché i PE del multiprocessore sono asincroni fra loro, e altrimenti non potrebbero essere visto che in genere eseguono istruzioni diverse che quindi hanno tempi diversi. Questo modo di processare le informazioni porta ad implementare sofisticati sistemi hardware-software per la gestione delle comunicazioni. Due differenti tipologie di architetture risolvono il problema con approcci diametralmente opposti: i sistemi «accoppiati strettamente» e quelli «accoppiati lascamente». I primi comunicano attraverso una memoria principale condivisa fra i vari PE, perciò la velocità con la quale i dati possono essere trasmessi da un PE all'altro dipende essenzialmente dalla larghezza di banda della memoria. Spesso ogni PE è fornito da una memoria locale molto veloce, ed è connesso direttamente alla memoria. Per questo motivo la memoria stessa deve essere di tipo multiporta cioè permettere un accesso simultaneo a due o più PE; l'alternativa consiste nel predisporre una rete di interconnessione tra i PE e la memoria e dei dispositivi che ne disciplinino l'accesso. Uno dei fattori limitanti all'espandibilità di un multiprocessore «strettamente accoppiato» è quindi il progressivo degrado delle prestazioni dovuto alla crescita delle collisioni negli accessi a memoria quando il numero dei PE aumenta.

Multiprocessori «lascamente accoppiati»

I sistemi multiprocessori «lascamente accoppiati», invece, presentano, di norma, meno conflitti di memoria di quelli «strettamente accoppiati» perché le comunicazioni non sono gestite tramite una memoria principale condivisa ma avvengono tramite un «sistema a trasferimento di messaggi», che abbrevieremo con STM. Ogni PE ha un insieme di dispositivi di I/O e una memoria principale locale dove mantiene dati e programma, tutto ciò costituisce un «modulo». Processi eseguiti sui moduli differenti possono comunicare tramite l'STM. In figura 2 è mostrato un modulo di un sistema multiprocessore lascamente accoppiato non gerarchico. Consiste di un PE, una memoria locale ML, alcuni dispositivi di I/O ed un'interfaccia verso altri moduli. In genere quest'interfaccia è un canale e un arbitro CA. Per canale si intende un dispositivo logico di comunicazione, che può essere realizzato fisicamente con un bus, una memoria principale o secondaria, etc. Nella figura 3 è illustrato il caso di connessione fra i moduli e l'STM. Se le richieste da due o più moduli collidono nell'accesso di uno stesso segmento fisico dell'STM, cioè se richiedono comunicazione allo stesso modulo, l'arbitro di tale modulo accorda il canale ad una delle richieste secondo il tipo di politica di accesso stabilito dal progettista in base alla caratteristiche fisiche del sistema. L'STM può essere di varia natura, un bus a partizione di tempo come nel PDP-11 o un sistema a memoria condivisa. Fin dai primi articoli di questa serie sulle macchine ed il calcolo parallelo, abbiamo posto l'accento sui canali di comunicazione fra PE e memoria e abbiamo visto come il collo di bottiglia risieda proprio nel limitato numero di canali disponibili. Anche nel caso di sistemi multiprocessore, l'STM è il fattore determinante della limitazione delle prestazioni del sistema. Se l'STM è un bus time-shared, le performance di-

pendono dalla frequenza dei messaggi trasmessi sul bus, dalla lunghezza del messaggio e dalla capacità del bus (bits/s). Quando il numero dei moduli aumenta, sicuramente il primo di questi fattori cresce e l'ultimo può risultare insufficiente per assicurare le prestazioni richieste. Se invece l'STM è realizzata con banchi di memoria condivisa, l'unico fattore limitante, sono i conflitti di memoria imposti dal grado di parallelizzazione delle connessioni fisiche moduli-memoria condivisa. Per questo motivo sono state introdotte le «memorie multiporta» che permettono per la loro particolare struttura l'accesso simultaneo di due o, raramente, più PE. La contesa per l'accesso a una stessa locazione è quindi risolta da un circuito di arbitraggio integrato nella memoria stessa.

Da un punto di vista concettuale un sistema di comunicazione come l'STM, può essere visto come un insieme di «porte» per ogni PE. Processi allocati ad un certo PE possono comunicare con altri processi allocati allo stesso PE, o su altri PE, tramite una porta di input ad essi associata. Tali porte ovviamente devono essere nello spazio di memoria riservato al PE, su cui i processi sono eseguiti. In questo modo i messaggi sono spediti dal processo mittente alla porta di input del processo destinatario.

Naturalmente se i due processi vengono assegnati allo medesimo PE, la comunicazione avverrà interamente nella memoria locale: se invece, il processo destinatario è allocato ad un altro PE, il messaggio verrà diretto alla porta di input propria del PE relativo al processo destinatario, nella memoria condivisa e da lì smistato alla porta di input del processo nella memoria privata del PE. La figura 4 mostra la struttura logica delle comunicazioni fra i PE.

Un processo allocato a PE1 mette un messaggio nella porta di input di un processo relativo allo stesso PE con il trasferimento indicato dalla freccia a. La freccia b invece, illustra un'azione in due passi che permette di trasferire un messaggio tra processi allocati a PE1 e PE2.

In particolare la freccia b1 manda il messaggio alla porta di input del PE2 nella memoria condivisa e la frecca b2 indica il trasferimento del messaggio alla parte di input, nella memoria locale di PE2, del processo destinatario.

Sistemi multiprocessore «strettamente accoppiati»

A causa della grande variabilità dei tempi di interferenza fra i processi, il through-put dei multiprocessori «lascamente accoppiati», può risultare troppo basso per quelle applicazioni che richiedono risposte in tempi molto brevi (applicazioni real-time). Multiprocessori più adatti a questi compiti sono quelli detti «strettamente accoppiati». Un'architettura tipo è mostrata in figura 5. Essa consiste di p PE, 1 moduli di memoria e di canali di I/O. Queste unità sono connesse attraverso un insieme di tre reti:

— la rete di interconnessione Processore-Memoria (RIPM);
— la rete di interconnessione I/O-Processore (RIIOP);
— la rete di interconnessione dei segnali di interrupt (RISI).

La RIPM può essere considerata come una schiera di switch ciascuno connettente un PE con un modulo della memoria condivisa. Tale schiera può essere realizzata crossbar con p ingressi ed l uscite e quindi p*l «incroci». Tenete presente che ogni incrocio è costituito da (n+k) singole connessioni dove n sono i fili di indirizzo del modulo di memoria e k i fili del bus dati. Perciò il sistema crossbar ha una complessità dell'ordine di p*l*(n+k); per valori grandi di p ed l, il costo della RIPM diventa dominante rispetto al costo totale del sistema multiprocessore. Naturalmente anche in questo caso deve essere risolta la richiesta contemporanea dell'accesso di un modulo di memoria da parte di più PE. Tale conflitto è in genere arbitrato dalla RIPM stessa. Vengono presi diversi accorgimenti per limitare i conflitti di memoria, tra cui la possibilità di comunicazione tra un PE e più moduli di memoria contemporaneamente (broadcasting) e spesso, il numero di questi ultimi, l, è preso pari a p, numero dei PE. Mantenendo poi i dati e le istruzioni del kernel del sistema operativo in una memoria locale (ML), ogni PE ha necessità di accedere ai moduli di memoria solo per i dati e le istruzioni del programma uten-

Figura 2 - Esempio di Modulo. Un sistema «lascamente accoppiato» può essere costruito con questi «mattoni».

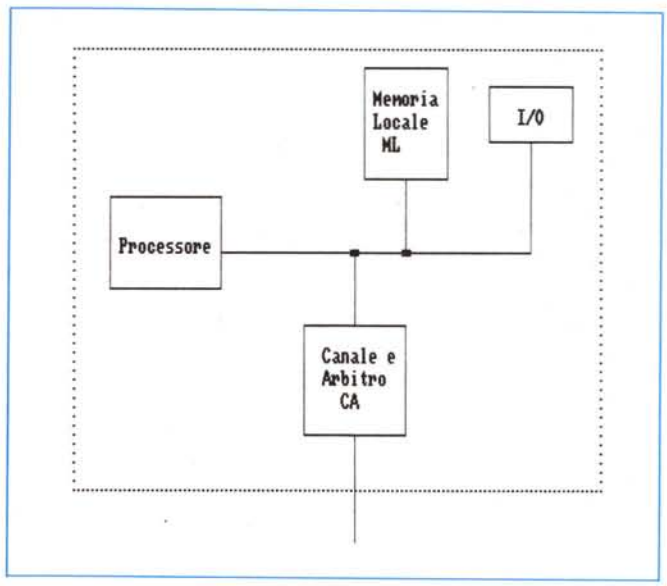
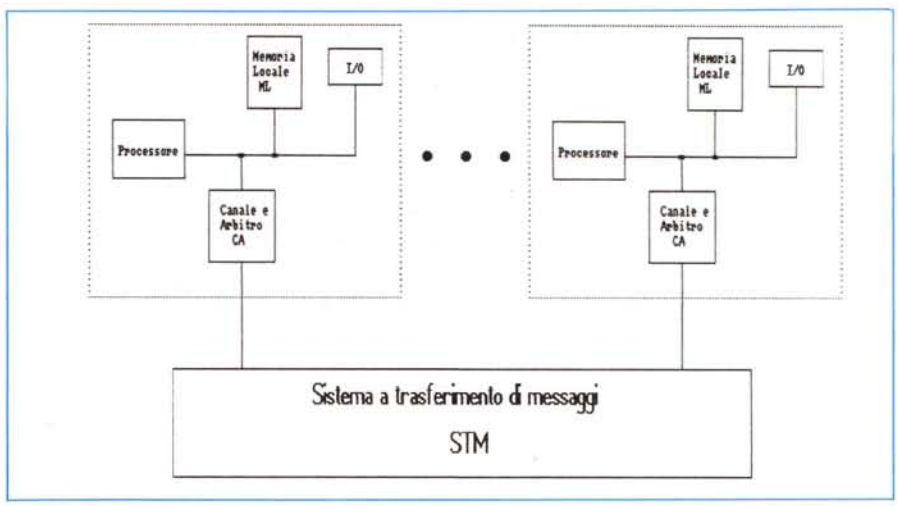


Figura 3 - Sistema multiprocessore «lascamente accoppiato». Il sistema non è gerarchico perché i moduli sono allo stesso livello. Sono possibili però configurazioni gerarchizzate su più livelli, l'STM potrà anch'essa avere più livelli.



Bibliografia
*Hwang K., Briggs F. «**Computer architecture and parallel processing**», Mc Graw-Hill, 1988.*

te. In alcuni casi possono venire predisposte «cache memory», speciali memorie veloci che mantengono i dati più frequentemente adoperati.

La RISI permette ad ogni PE, di asserire un interrupt ad un qualsiasi altro PE del sistema; tale rete permette una efficace sincronizzazione dei processi su differenti PE. La RIIOP consente la comunicazione di ciascun processore con un canale di I/O che è connesso ai dispositivi periferici. La RISI può essere, a sua volta, realizzata con un bus time-shared o con un sistema crossbar. Per esempio, nell'Univac 1100/80 e nell'Honeywell 60/66, una connessione può essere stabilita tra ciascuna coppia di PE. Tuttavia anche se la soluzione su bus comporta i problemi già esposti, la relativa bassa frequenza degli interrupt rende questa soluzione praticabile.

Sistemi simmetrici e asimmetrici

Il sistema multiprocessore si dice «omogeneo» se le unità funzionali fisiche assemblate nel sistema sono identiche oppure «eterogeneo» se sono di tipo diverso. Anche se omogeneo, un sistema può essere «asimmetrico», quando, ad esempio, le unità funzionali hanno diversa dimensione, spazio di indirizzamento e/o di I/O differente. Nella maggior parte dei casi, tale differenza è trasparente ai processi utente, e interessa soltanto le attività del sistema operativo, specialmente rispetto al fattore di carico. Se un sistema è asimmetrico rispetto all'I/O si ha una struttura tipo quella in figura 6. Tuttavia tale sistema presenta la caratteristica di rendere inaccessibili le periferiche connesse ad un PE se questo va in errore o si rompe. Viene perciò tollerata una certa ridondanza in modo da rendere il sistema protetto da un certo livello di errori. Ovviamente la massima protezione si ottiene connettendo i PE con i dispositivi di I/O tramite crossbar.

Caratteristiche dei processori per sistemi multiprocessore

La maggior parte dei sistemi multiprocessore della penultima generazione, erano costruiti su PE non progettati specificatamente per architetture parallele. Negli ultimi anni sono stati presentati processori che hanno caratteristiche che li rendono molto adatti a strutture multiprocessore. Vediamo perciò quali sono queste essenziali caratteristiche.

Recupero di processi

L'architettura dei processori usati in sistemi multiprocessore, dovrebbe riflettere il fatto che il processo e il processore sono due entità differenti. Se il processore, per un qualsiasi motivo si interrompe, dovrebbe essere possibile per un altro processore recuperare il processo interrotto cosicché l'esecuzione del processo stesso possa continuare. Senza tale caratteristica infatti, se di n processi cooperanti o semplicemente comunicanti, anche un solo è interrotto, tutti i restanti processi vengono a catena interrotti. Il problema può essere risolto se i registri che contengono lo stato del processo in esecuzione sono posti fisicamente fuori dal processore.

Efficiente «context switching»

Per «context switching» si intende il cambio del contenuto dei registri di un processore contenenti lo stato del processo in esecuzione, quando tale processo passa dallo stato di esecuzione a quello di attesa (o terminazione) ed un altro processo è mandato in esecuzione. In un ambiente multiprogrammato caratteristico delle macchine MIMD, tale operazione è eseguita assai di frequente e richiede grandi aree per lo stack e la coda necessari a mantenere l'ambiente e l'ordine dei processi. Nel processore Cyber-170, è prevista una istruzione apposita chiamata *central exchange jump* che scambia l'insieme dei registri interni del processore con un'area di memoria specificata. Una soluzione efficiente potrebbe essere quella di disporre un solo registro interno nel quale memorizzare il puntatore ad una zona di memoria vista come insieme di registri di stato, in questo modo si può riservare una zona per ciascun processo. In ogni caso tale operazione dovrebbe essere molto rapida per non aggiungere overhead.

Supporto memoria virtuale

Il frazionamento dei processi per sfruttare il parallelismo di un algoritmo, spinge verso una modularizzazione del codice e dei dati in segmenti logicamente disgiunti. Se il processore supporta una memoria virtuale, può indirizzare tramite una unità MMU (Memory Management Unit) più locazioni di quante fisicamente sono disponibili, allora tale caratteristica consente una efficiente programmazione ed esecuzione dei processi.

Primitive di sincronizzazione

La cooperazione dei processi e la loro esecuzione contemporanea richiede una efficiente sincronizzazione. In particolare se i processi scambiano dati è necessario che la zona di memoria in cui essi risiedono sia assegnata in mutua esclusione. Potrebbe altrimenti accadere che un processo legga dei dati parzialmente aggiornati utilizzandoli altresì come corretti. Allora l'accesso alla zona di memoria in questione assegnata ad un processo è vietato finché il processo stesso non la rilascia. Questo meccanismo può essere ottenuto tramite dei «semafori». Ogni semaforo è una variabile booleana con una coda associata in cui sono posti i processi in attesa di accedere alla risorsa. Sul semaforo possono essere eseguite solo due operazioni, dove s è un semaforo:

l'operazione P(s) codificata come
if s=1 then s=0
l'operazione V(s) codificata come s=1

L'operazione P(s) testa il semaforo, se è «verde» allora lo mette a «rosso»; questo indica ovviamente il possesso esclusivo della risorsa. L'operazione V (s) pone invece il semaforo a verde e perciò indica il rilascio della risorsa. Perciò un processo che volesse usare una risorsa, dovrebbe seguire questo tipo di procedura:

```
P(s);
corpo istruzioni;
V(s);
corpo istruzioni;
.
.
```

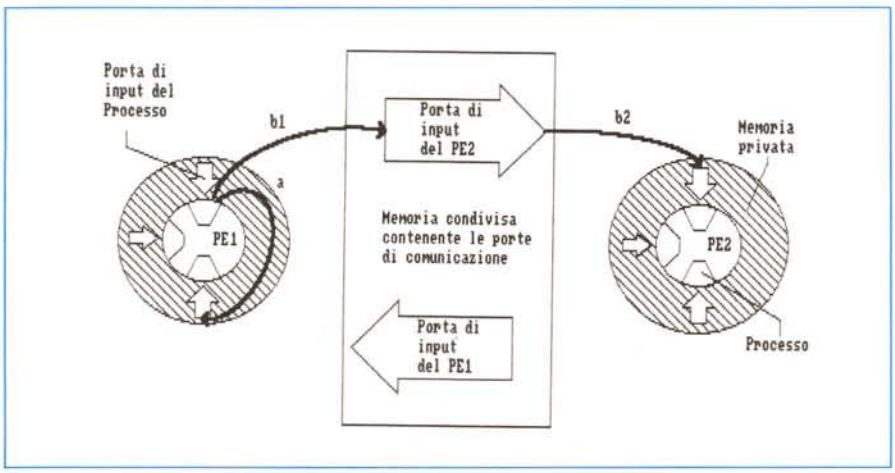


Figura 4 - Schema concettuale di comunicazione tra processi in ambiente multiprocessore.

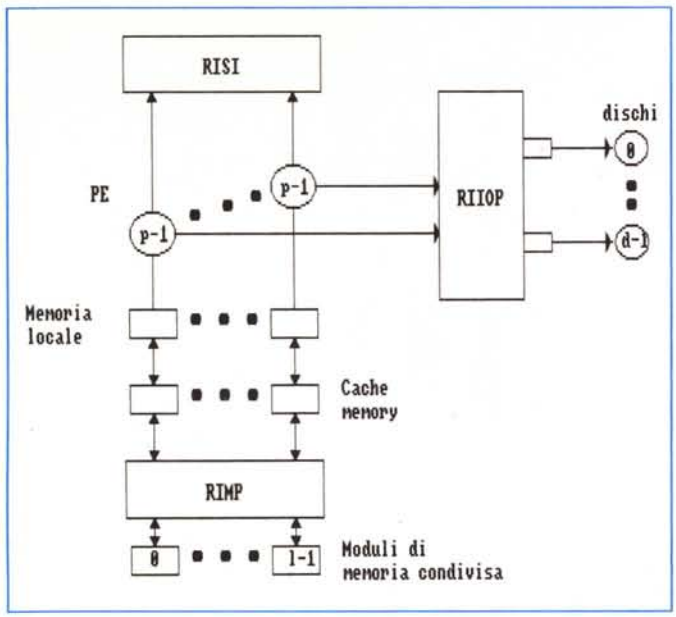


Figura 5 - Schema di un multiprocessore «strettamente accoppiato».

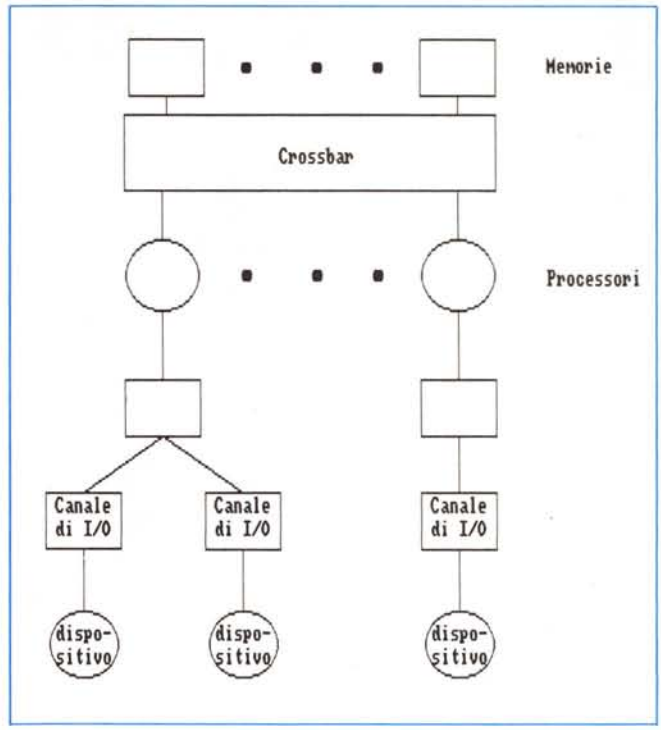


Figura 6 - Sistema multiprocessore con I/O asimmettrico. ►

Dove con corpo istruzioni si intende un generico insieme di istruzioni. Tuttavia il sistema potrebbe perdere il suo effetto se il processo fosse interrotto durante l'esecuzione dell'IF THEN del costrutto P(s). Una possibile codifica Assembler 68000 di tale istruzione è infatti divisa in più operazioni:

```
test:
cmp s,0
breq test
move 0,s
```

Supponiamo che un processo P1 esegua questa sequenza e trovi s uguale a 1; prima che possa eseguire l'istruzione move e mettere il semaforo a rosso, viene interrotto da un altro processo P2, che testando la variabile s e trovandola a 1 accede ai dati associati, modificandoli. P2 rilascia poi la zona eseguendo una V(s) e termina. P1 è allora rimesso in esecuzione ed esegue il move, accedendo ai dati. Tuttavia i dati stessi sono stati modificati e P1 non se ne è accorto perdendo i dati precedenti!

La soluzione sta nel prevedere nel set di istruzioni del microprocessore una istruzione indivisibile, cioè che non possa essere interrotta in nessuna condizione, tramite la quale implementare l'operazione P(s).

Questo tipo di istruzione è spesso indicata come *test-and-set*, il 68000 per esempio ha un'istruzione che assolve a questo compito, che ha mnemonico TAS.

Meccanismi di comunicazione interprocessore

L'insieme dei processori usati in un sistema multiprocessore deve avere un efficace sistema di comunicazione interprocessore.

Questo meccanismo dovrebbe essere implementato in hardware, in modo da ottenere la massima efficienza e semplicità nella comunicazione. Tuttavia nei sistemi «strettamente accoppiati» è possibile avere meccanismi software di comunicazione senza un esplicito meccanismo hardware.

Tale meccanismo è inefficiente perché ciascun processore deve periodicamente controllare le sue porte di input per verificare se c'è qualche messaggio in arrivo; questo sistema è detto «polling mailbox».

Naturalmente non è possibile pensare di adottare un sistema «mailbox» se il numero di processori è grande; infatti, maggiore è il numero di processori maggiore risulterà statisticamente il numero di comunicazioni e perciò maggiore deve essere la frequenza di interrogazione della mailbox con aumento dell'overhead dovuto a tale operazione.

Adeguato set di istruzioni macchina

Il set di istruzioni del processore dovrebbe facilitare l'implementazione di linguaggi ad alto livello che permettano un'effettiva concorrenza delle procedure e una efficiente manipolazione delle strutture dei dati. Dovrebbero essere presenti anche istruzioni per creare e terminare esecuzioni parallele di processi relativi ad uno stesso programma. Indispensabile si rivela un clock real-time per generare identificativi dei processi privi di ambiguità e per gestire segnali di time-out per la sincronizzazione dei processi. Un sistema multiprocessore infatti può conservare un corretto funzionamento tramite dei «watchdog», vale a dire timer associati con le varie risorse, che impediscono il blocco del sistema in attesa di eventi che ritardano o che non si verificano.

In definitiva un sistema multiprocessore è ben strutturato solo se ogni componente può controllare gli altri in maniera relativamente semplice e diretta.

Conclusioni

Le macchine MIMD presentano un alto grado di flessibilità e potenziale efficienza, tuttavia per questo si rivelano anche le più complicate da gestire e progettare, in quanto ogni componente del sistema deve riflettere tale flessibilità di funzionamento.

Nel prossimo articolo considereremo in particolare le reti di interconnessione e le organizzazioni di memoria che si possono adottare per aumentare l'efficienza del sistema; in seguito tratteremo gli aspetti software dei sistemi multiprocessore in particolare del sistema operativo, dei linguaggi e delle strutture di programmazione.

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*

Prezzi IVA compresa

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.

FLOPPERIA
SRL