di Andrea de Prisco

M.I.P.S.

A partire da questo mese, Appunti di Informatica si occuperà di processori. L'acronimo che dà il titolo a questa prima puntata, MIPS, sta per milioni di operazioni al secondo, una quantità che misura, in un certo senso, la velocità di un processore. Nel corso di questo primo articolo scopriremo appunto qual è il «senso» della frase precedente. Buona lettura.

MIPS, chi era costui?

Se qualche anno fa andavano molto di moda termini come «software», «hardware», «floppy disk», oggi parole come MIPS, RISC, MMU sembrano aver preso un posto assai preponderante nel comune lessico dei computer-freak dell'ultima generazione (era «post-Amiga»). Oggi questi Signori non si accontentano più di 16,32 o 64 bit, clock iper galattici da 16 MHz, coprocessori matematici, grafici o sonori... oggi un computer per essere «figo» (dicono loro) deve avere perlomeno un processore RISC da 150 MIPS (poveri illusi) o tutt'al più una decina di Transputer che lavorano in parallelo per darti la massima libidine computereccia (possibile e immaginabile...). A cosa servano poi i MIPS o cosa sia un processore RISC o un Transputer... nessuno lo sa.

MIPS, come detto, vuol dire milioni di operazioni al secondo. Se dunque un processore «va» a 3 mips vuol dire che compie la bellezza di 3 milioni di operazioni al secondo. Se prendiamo un altro processore che «corre» a 6 mips, la prima cosa che ci viene in mente è che vada il doppio «più veloce» del processore di prima. Sbagliato: poi vedremo perché.

Un discorso simile si potrebbe fare circa il clock: un processore a 4 MHz non è detto che sia più veloce di un altro processore a 1 MHz: un esempio classico è la coppia Z80-6502, il primo generalmente clock-ato a 4 MHz, il secondo ad 1... anni ed anni di micro informatica ad 8 bit non hanno affatto chiarito quale dei due fosse più veloce.

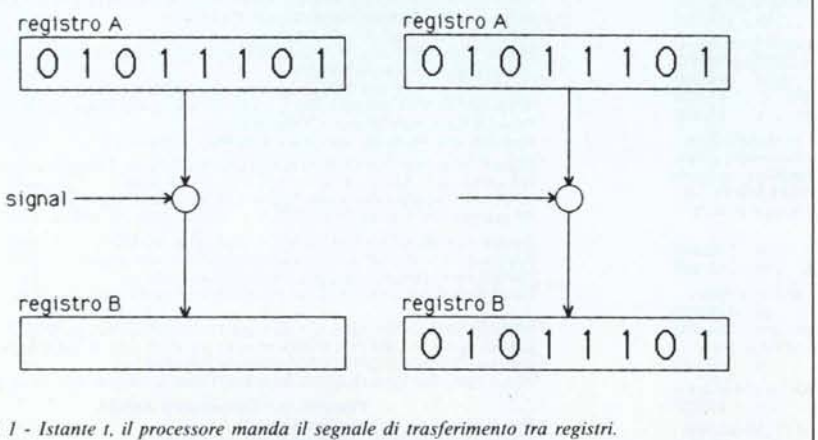

Figura 1 - Istante t, il processore manda il segnale di trasferimento tra registri.
Figura 2 - Istante t + 1, il dato è disponibile anche nel registro B.

Cominciamo dal Clock

Le operazioni che avvengono all'interno di un computer sono tutte sincronizzate da un orologio (clock) che ad intervalli di tempo regolari invia un impulso alle varie componenti della macchina (memoria, processori, bus, ecc.). Il ciclo di clock è dunque l'unità di tempo dei sistemi di calcolo: una determinata funzione potrà essere svolta in 1 ciclo, in 2 cicli, in 7 cicli... a seconda della durata di questi e della complessità delle operazioni da svolgere. All'interno di una CPU, le operazioni più brevi sono certamente quelle di lettura/scrittura nei registri interni e sono compiute in un solo ciclo di clock. Se il processore mantiene qualcosa in un registro e «vuole» copiarne il contenuto in un altro registro «sprecherà» al più un ciclo: se all'istante t (si veda figura 1 e 2) ha «innescato» il procedimento (ovvero la parte controllo ha inviato il «signal» alla porta posta tra i due registri), all'istante t + 1 il dato sarà disponibile nel re-

gistro destinazione. Si noti che, nella frase precedente, istante è sinonimo di ciclo di clock: come già detto un ciclo di clock è l'unità di tempo.

Ma all'interno di un processore possono succedere anche cose più complesse di un semplice trasferimento. Ad esempio un bella addizione: il contenuto del registro A deve essere sommato al registro B e il risultato deve rimanere in A. Come dire «somma ad A il contenuto di B». All'interno della CPU esiste la cosiddetta ALU, unità aritmetico logica, preposta allo svolgimento delle operazioni aritmetiche. Per eseguire una somma occorre presentare all'ingresso della ALU i due operandi e prelevare al ciclo di clock successivo il risultato. Di queste cose abbiamo già parlato sul numero 55 di MC (sempre in «Appunti...») quindi vi rimandiamo lì per ulteriori chiarimenti a riguardo. Comunque, tornando alla somma di prima, possiamo identificare tre fasi:

— scrivere A e B negli ingressi della ALU
— eseguire la somma
— trasferire il risultato in A

tre operazioni elementari che potremo eseguire in tre cicli di clock successivi. Si noti il parallelismo reale della prima microistruzione: dato che le operazioni di scrittura di A e scrittura di B sono indipendenti, nulla ci vieta di effettuarle contemporaneamente. Nelle figure 3...6 abbiamo «fotografato» i quattro eventi corrispondenti allo stato iniziale e nei 3 cicli di clock successivi.

Da quanto detto, si evincerebbe che più alto è il clock, maggiore è la «velocità». Questo è vero solo se stiamo considerando lo stesso processore: è ovvio! Ma molti non lo capiscono... vediamo insieme perché.

Primo: all'interno di un processore, quello che succede per eseguire una determinata operazione è tutt'altro che standard. Per fare subito un esempio, prima abbiamo dato per scontato che la scrittura del registro A nel primo operando della ALU e la scrittura di B nel secondo possa avvenire contemporaneamente. Ma non è detto: se il processore ha una architettura interna con collegamenti molto ripartiti (ne abbiamo parlato lo scorso mese) potrebbe non essere possibile la scrittura in parallelo. In questo caso sprecheremmo un ciclo di clock in più: la prima fase si sdoppierebbe in:

— scrivere A nel primo ingresso della ALU
— scrivere B nel secondo ingresso della ALU

Secondo: non sta scritto da nessuna parte che una ALU esegua una addizione in un solo ciclo. Potrebbe impiegarne 2, 3 7... o quanti vuole lei.

Terzo: l'esecuzione di una istruzione LM si compone di una prima fase di decodifica (il processore, letta l'istruzione dalla memoria, deve capire innanzitutto quello che deve fare), seguita dalla esecuzione vera e propria che abbiamo già trattato. Ovvero un certo numero di cicli (maggiore di zero!) sono utilizzati per «capire il da farsi». Quanti?

Ed è qui che casca l'asino, o meglio, gli asini. Dipende. Varia infatti da processore a processore e l'unico modo per scoprirlo è leggere (attentamente) le specifiche del processore in questione.

Ricapitolando potremmo avere processori che in 4 cicli di clock eseguono una addizione registro-registro, processori che per fare lo stesso ne impiegano 10. Ma i «cicli» possono essere più o meno brevi, si parla di 1 MHz, 4.77, 8, 16 MHz... dunque nella valutazione della velocità di elaborazione dobbiamo tener conto di vari parametri.

Usiamo allora i mips!

Se togliamo da mezzo clock e cicli impiegati, un modo per tagliare apparentemente la testa al toro potrebbe essere quella di «misurare» i processori in mips. Se un processore esegue una operazione di addizione registro-registro in 8 cicli di clock e la frequenza di questi è pari a 4 MHz (ovvero un ciclo dura un quarto di microsecondo), espresso in unità di tempo la durata di una addizione è:

$$8 \cdot 1/4 \cdot 10 - {}^{6} \text{ secondi}$$

pari dunque a 2 microsecondi. Questo è effettivamente un risultato tangibile: se un altro processore per fare la stessa cosa impiega un microsecondo siamo autorizzati a pensare che sia il doppio più veloce... quando si tratta di eseguire addizioni registro-registro. Volendo avere un parametro globale, per misurare la velocità (di uno speci-

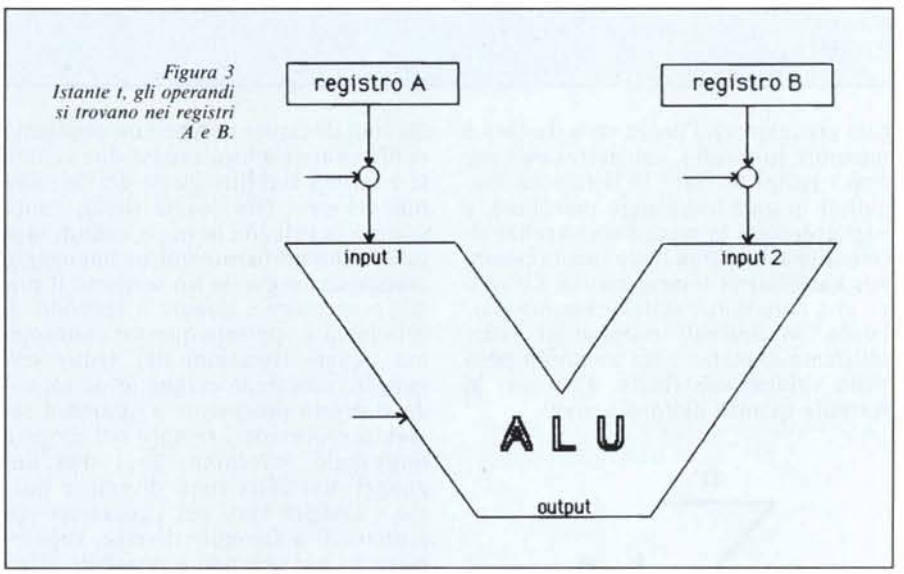

Figura 3
Istante t, gli operandi si trovano nei registri A e B.

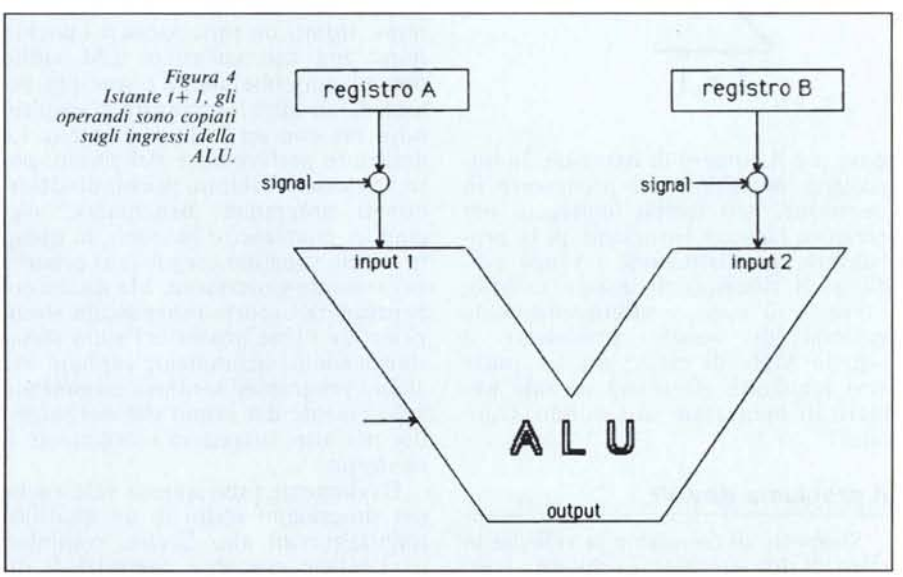

Figura 4
Istante t+ 1, gli operandi sono copiati sugli ingressi della ALU.

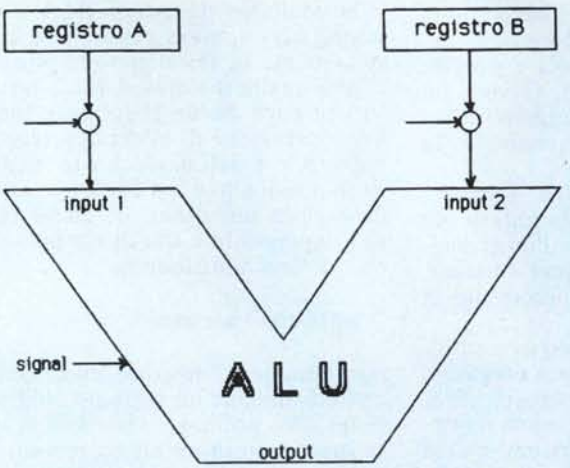

Figura 5 - Istante t + 2, si segnala alla ALU di eseguire l'addizione.

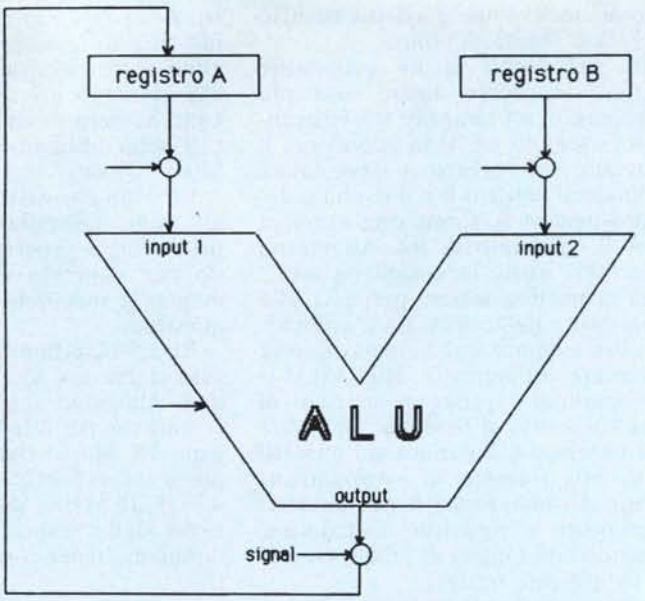

Figura 6 - Istante t + 3, si trasferisce il risultato nel registro A.

fico processore), l'unica cosa da fare è misurare (o meglio, calcolare) uno per uno i tempi di tutte le istruzioni eseguibili in quel linguaggio macchina, e rispolverando le nostre conoscenze di statistiche, fare una bella media pesata per calcolare il tempo medio. Ovvero se una istruzione, statisticamente parlando, in normali programmi viene utilizzata di meno, avrà un minor peso nella valutazione finale. Espresso in formule quanto detto si scrive:

$$\sum_{i=1}^{n} t_i \cdot p_i$$

dove n è il numero di istruzioni in linguaggio macchina del processore in questione, t_i il tempo impiegato per eseguire l'iesima istruzione, p_i la probabilità che l'istruzione i venga eseguita. Il risultato di questo calcolo, espresso in mips, è effettivamente la velocità di «quel» processore a «quei» MHz di clock: un computer così realizzato effettuerà un tale numero di operazioni al secondo. Contenti?

Il problema dov'è?

Supposto di conoscere la velocità in mips di due processori «diversi», cerchiamo di capire perché non possiamo confrontare tra loro queste due velocità e quindi stabilire quale dei due «va più veloce». Noi, come detto, conosciamo la velocità in mips, quindi sappiamo quante istruzioni in linguaggio macchina esegue in un secondo il primo processore e quante il secondo. Il problema è appunto questo: conosciamo quante istruzioni nel «suo» linguaggio macchina esegue in un secondo il primo processore e quante il secondo processore, sempre nel proprio linguaggio macchina. Se i due linguaggi macchina sono diversi, e questo è sempre vero per processori appartenenti a famiglie diverse, converrete con noi che non è possibile effettuare confronti basandosi solo sui mips. Infatti un processore a «pochi» mips, ma con istruzioni LM molto potenti potrebbe anche essere più veloce di un altro processore a «molti» mips ma con un LM assai scarso. La maggiore performance del primo, potrebbe essere valutata provando determinati programmi benchmark, saggiando, cronometro in mano, in quanto tempo vengono eseguiti sul primo e sul secondo processore. Ma anche coi benchmark occorre usare molta attenzione: se i due processori sono abbastanza simili, sicuramente capiterà che alcuni programmi saranno eseguiti più velocemente dal primo che dal secondo, per altri succederà esattamente il contrario.

Ovviamente tutto questo vale anche per programmi scritti in un qualsiasi linguaggio ad alto livello, compilati ovviamente con due compilatori distinti ognuno per il processore sul quale dovrà «girare» il codice eseguibile. Lo stesso programma, compilato per il processore col linguaggio macchina più potente avrà un codice eseguibile formato da meno istruzioni del corrispondente eseguibile per il processore, con più mips, ma con istruzioni più semplici.

A cosa servono i mips?

La prima risposta che saremmo tentati di dare è certamente: «vendere computer...» ma non è vero. O meglio, attualmente sono molto sfruttati per questo scopo, ma originariamente non era così. I mips servono solo ed esclusivamente per valutare miglioramenti di prestazioni tra processori della stessa famiglia, ovvero con medesimo linguaggio macchina. Infatti uno stesso processore può essere realizzato con diverse tecniche, oppure si può pensare di mettere in uno stesso computer più processori in modo che eseguano parallelamente più processi.

A partire dal prossimo numero vedremo, partendo da un processore abbastanza «basico», come sia possibile aumentare i mips cambiando la strategia di progetto del processore stesso. In seguito proveremo anche ad aggiungere processori per ottenere sistemi multiprocessor. Scopriremo, tanto per anticipare subito i primi «colpi di scena», che 4 × 2 non fa 8 quando si tratta di utilizzare 4 processori da 2 mips ciascuno, ma molto, molto meno... Arrivederci!

Quando Lotus 123 non basta più:

I programmi accessori o "add-in" per Lotus 123 consentono di aumentarne ulteriormente la potenza e la flessibilità, mantenendone inalterate le caratteristiche di semplicità d'uso. Nati proprio per funzionare con Lotus, gli add-in presentano la stessa struttura a menu di selezione orizzontale che gli è caratteristica e posseggono tutti la funzione di guida d'aiuto sempre in linea che va ad integrare quella standard di Lotus 123.

Sideways: elimina la limitazione di stampa data dalla larghezza fisica del foglio. Ruotando di 90 gradi i caratteri, consente la stampa di fogli molto larghi sfruttando la lunghezza a modulo continuo.
Consente di assegnare attributi particolari ai caratteri di stampa, quali il grassetto ed il sottolineato e di definirne a piacere la grandezza.

Spreadlink: converte nel formato 123 qualsiasi file testo o rapporto generato da altri pacchetti. Spreadlink esegue una conversione "intelligente" dei dati, determinandone automaticamente il layout e riconoscendone automaticamente il tipo (testo, numero, data, etc.).

Goal Solutions: spesso si conosce il risultato che si desidera ottenere, mentre non si conoscono i dati di partenza. Goal Solutions aggiunge a Lotus 123 anche questa possibilità: attraverso una serie variabile di iterazioni di calcolo, consente, tra l'altro, di calcolare l'incremento nelle vendite richiesto per raggiungere un certo budget.

3-D Graphics: incrementa la potenza della funzione grafica di Lotus 123, dotandola anche della tridimensionalità.
Genera grafici tridimensionali ad istogramma, lineari e a superficie. Offre la possibilità di assegnare un fattore di rotazione al grafico, di variare il punto d'osservazione e di abilitare o disabilitare il tracciamento delle linee nascoste.

Inword: aggiunge a Lotus 123 un completo elaboratore di testi, consente di giustificare il testo, effettuare operazioni di ricerca e sostituzione, assegnare attribuiti come grassetti, corsivi e sottolineati, oltre che inserire nel testo dati tratti direttamente dal foglio elettronico. Ideale per lettere circolari personalizzate e per tutte quelle applicazioni di scrittura che richiedono di estrarre dei dati da Lotus 123.

Dejà: l'interfaccia ideale tra Lotus 123 e dBASE III. Consente di leggere file dBASE dall'interno di Lotus 123, di modificarne i dati, di inserire filtri e di aggiungere e cancellare i record. Aggiunge a Lotus nuove funzioni orientate alla gestione degli archivi, insieme a molti dei comandi propri di dBASE III.
Un prodotto indispensabile per ottenere la massima flessibilità dai propri dati.

Desidero ricevere materiale illustrativo sugli "Add-in".
Cognome e nome _____
Azienda _____
Via _____
CAP _____ Città _____

Compilare e spedire in busta chiusa a:

J.soft

Distributore per l'Italia

Viale Restelli, 5 - 20124 Milano
Tel. 02/6888228-683797-6880841/2/3