Appunti di Calcolatori Elettronici
Esecuzione di istruzioni in parallelo

Introduzione

L'obbiettivo primario dei progettisti dei calcolatori è stato sempre, fin dagli albori dell'informatica, quello di rendere i calcolatori quanto più veloci possibile: a tal proposito, ci si rese conto subito che le accelerazioni ottenibili tramite un miglioramento dell'hardware sono comunque limitate; ad esempio, le leggi della fisica dicono che niente può andare più veloce della luce (cioè circa 30 cm per ns nel vuoto e 20 cm in un filo di rame), il che comporta che, per realizzare un calcolatore con un tempo di esecuzione di 1 ns per istruzione, la distanza totale su cui il segnale elettrico può viaggiare, all'interno della CPU, verso la memoria e di ritorno dalla memoria, non può essere maggiore di 20 cm; in poche parole, calcolatori molto veloci devono anche essere molto piccoli.

Non basta però dire che minori dimensioni equivalgono a maggiori velocità: infatti, i calcolatori veloci producono più calore di quelli lenti, per cui la riduzione delle dimensioni complica i meccanismi di dissipazione del calore. Proprio per questo, alcuni supercalcolatori sono immersi nel *freon liquido*, usato come liquido di raffreddamento, per riuscire a trasferire all'esterno il calore il più velocemente possibile.

Sulla scorta di queste ed altre considerazioni relative ai vincoli fisici di realizzazione dei calcolatori, si è pensato che, al posto di costruire un'unica CPU ad alta velocità, si potrebbe costruire una macchina con molte ALU più lente (e quindi più economiche e facili da realizzare) o perfino con diverse CPU complete, da collegare e far funzionare in modo da ottenere la stessa potenza di calcolo ma ad un costo minore. Si parla in questo caso genericamente di **macchine parallele**;

vogliamo dare dei cenni ad alcune tecniche di realizzazione di questo tipo di macchine.

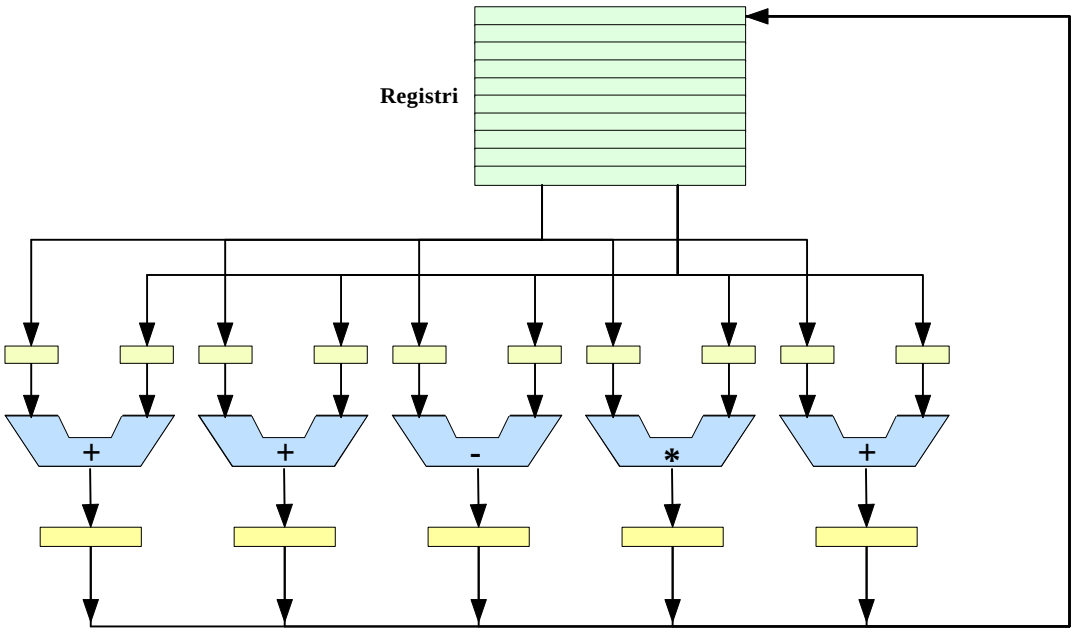
Classificazione di Flynn

In primo luogo, le macchine parallele possono essere divise in tre categorie, a seconda di quanti flussi di istruzioni e di dati sono presenti in esse (**classificazione di Flynn**):

- **macchine SISD** (*Single Instruction Single Data*): flusso di istruzioni unico e flusso di dati unico;
- **macchine SIMD** (*Single Instruction Multiple Data*): flusso di istruzioni unico e flusso di dati multiplo;
- **macchine MIMD** (*Multiple Instruction Multiple Data*): flusso di istruzioni multiplo e flusso di dati multiplo.

La macchina tradizionale di Von Neumann è ovviamente di tipo SISD: essa ha un unico flusso di istruzioni (cioè un unico programma) eseguito da una CPU ed ha poi un’unica memoria che contiene i dati usati dal programma; la prima istruzione viene presa dalla memoria e poi eseguita, dopodiché si passa al prelievo ed alla esecuzione dell’istruzione successiva e così via fino al termine del programma.

Si tratta perciò di un modello prettamente sequenziale. Nonostante questo, anche in un simile modello è possibile avere un limitato grado di parallelismo: infatti, si può tentare di prelevare una nuova istruzione e cominciare ad eseguirla prima che quella attualmente in esecuzione sia completata. Ad esempio, il calcolatore CDC 6600 ed alcuni dei suoi successori erano dotati di unità funzionali multiple (**ALU specializzate**), ognuna delle quali poteva eseguire una sola operazione ad alta velocità. Un esempio di una simile macchina è riportato nella figura seguente:



Una CPU con 5 unità funzionali che possono operare in parallelo

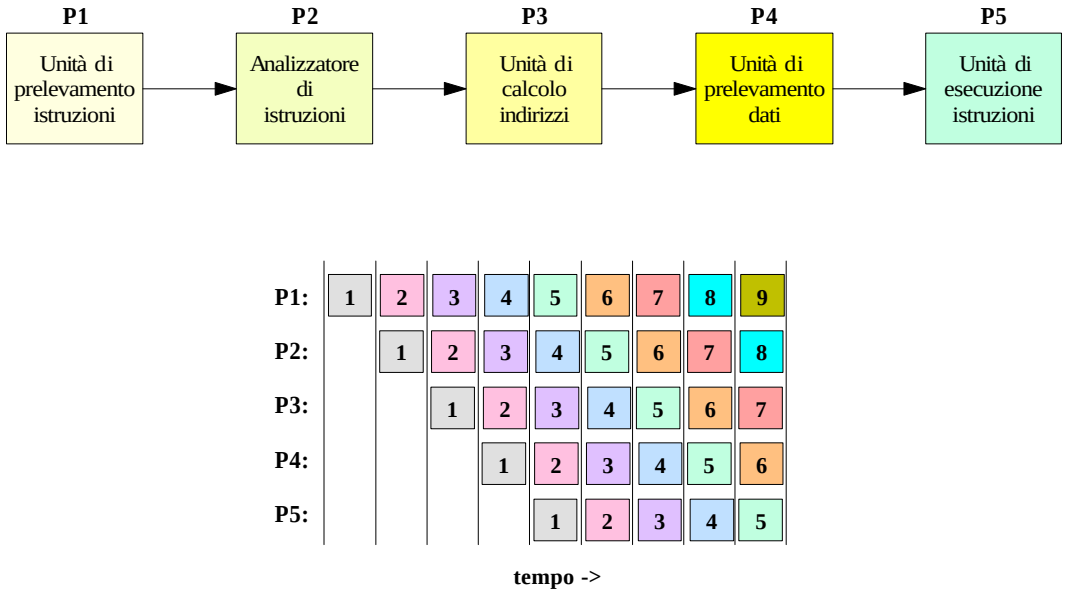
Come si vede, abbiamo 5 unità funzionali, tre per le operazioni di somma (molto frequenti) ed una ciascuno per le operazioni di sottrazione e moltiplicazione.

L'idea di fondo di una simile architettura è che l'unità di controllo vada a prendere una istruzione dalla memoria e poi la passi ad una delle unità funzionali per l'esecuzione; nel frattempo, l'unità di controllo prende l'istruzione successiva e la passa ad un'altra unità funzionale. Questo procedimento continua fin quando tutte le unità funzionali del tipo richiesto sono occupate oppure quando l'operando necessario ad una istruzione è ancora in fase di elaborazione.

Questa strategia richiede che il tempo per eseguire una istruzione sia abbastanza maggiore di quello richiesto per prelevarla: per questo motivo, essa viene generalmente usata solo per operazioni in virgola mobile, che sono complesse e lunghe, e non per operazioni su interi, che sono invece semplici e veloci.

Macchine a pipeline

Una variante della strategia appena descritta consiste nel dividere l'esecuzione di ogni istruzione in più parti; ad esempio, nella prossima figura è illustrato quello che accade per una CPU suddivisa in 5 unità di elaborazione, denominate P1, P2, P3, P4 e P5:



Macchina a 5 unità con pipeline (figura in alto). Lo stato di ogni unità di elaborazione (figura in basso) viene illustrato in funzione del tempo

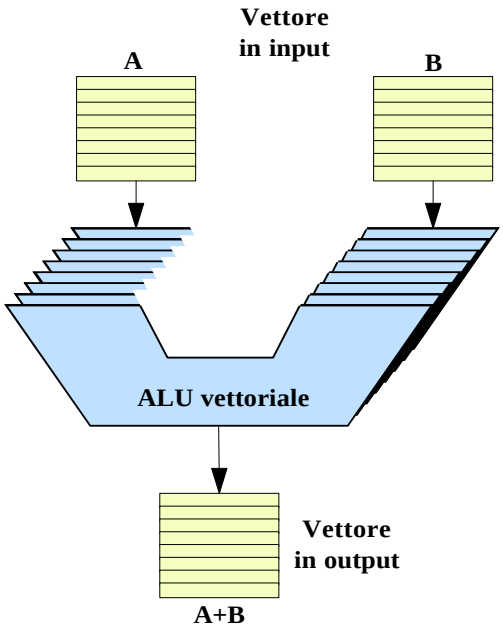
Durante il primo intervallo di tempo, l’istruzione viene prelevata dalla memoria dall’unità P1; nel secondo intervallo di tempo, l’istruzione viene passata a P2 per essere analizzata, mentre, in contemporanea, P1 prende una nuova istruzione. In ognuno degli intervalli successivi, una nuova istruzione è prelevata da P1, mentre le altre istruzioni sono passate ad una unità successiva lungo il percorso.

Una simile organizzazione è detta **macchina a pipeline**: se ciascun intervallo di tempo dura x ns, sono necessari $5x$ ns per eseguire una istruzione; tuttavia, all’unità P5 arriva una istruzione completa ogni x ns, ottenendo così un incremento di prestazioni di ben 5 volte (una volta raggiunta la condizione di regime, a partire dal quinto intervallo di tempo).

Una cosa importante da notare è che una macchina a pipeline, nonostante impieghi una sorta di parallelismo interno, è ancora una macchina SISD: c’è infatti un solo programma ed un solo insieme di dati.

Macchine vettoriali e Array Processor

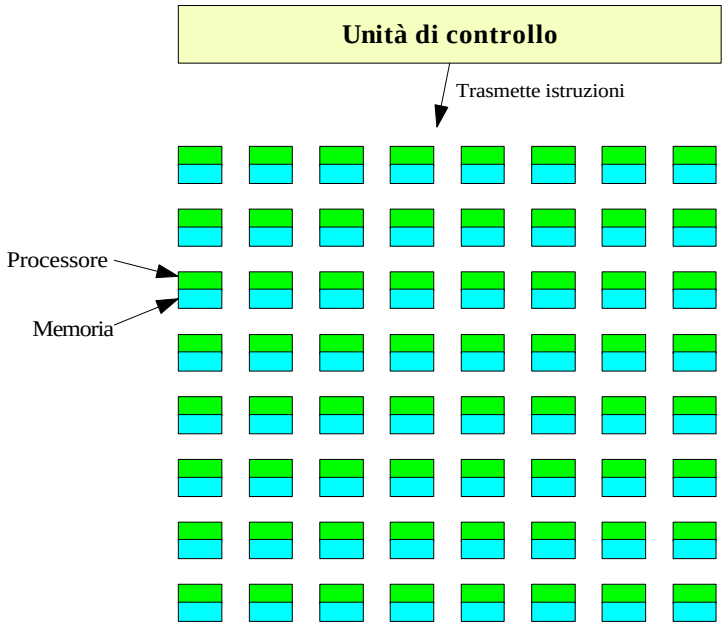
A differenze delle macchine SISD, le macchine SIMD operano invece in parallelo su insiemi multipli di dati. Una tipica architettura adatta a questo compito è la cosiddetta **macchina vettoriale**, schematizzata nella figura seguente:



Esempio di ALU vettoriale composta da 8 unità

In questo caso, il percorso dei dati a simile a quello delle macchine SISD tradizionali, ad eccezione del fatto che, invece di avere una sola variabile per ogni ingresso della ALU, abbiamo ora un vettore con N input. Ovviamente, la ALU è ora una **ALU vettoriale**, capace cioè di eseguire delle operazioni ricevendo in ingresso non più due scalari, bensì due vettori, e restituendo a sua volta un vettore.

Un approccio alternativo è costituito dal cosiddetto **Array Processor**, illustrato nella figura seguente:

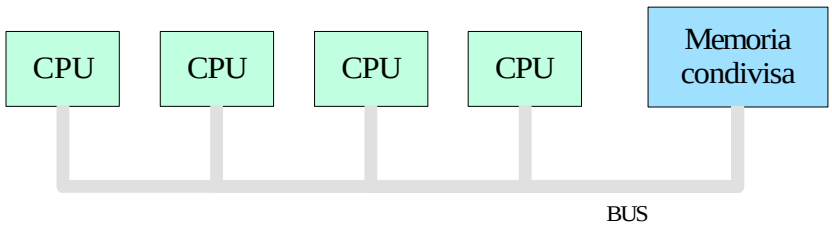


Griglia di processori/memoria di dimensione 8´8

Si tratta di una vera e propria griglia quadrata di elementi, ognuno dei quali è costituito da un processore e da una memoria. Esiste una singola unità di controllo, che trasmette le istruzioni, le quali sono eseguite da tutti i processori usando la propria memoria (caricata durante la fase di inizializzazione). Una simile architettura è particolarmente adatta per calcoli su matrici.

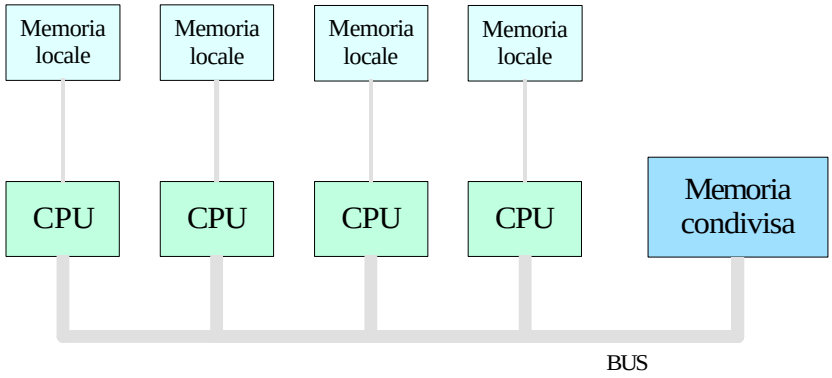
Macchine MIMD

La terza categoria di Flynn è costituita dalle macchine MIMD, in cui diverse CPU eseguono diversi programmi, a volte condividendo una parte di memoria comune. La figura seguente mostra ad esempio una macchina MIMD in cui più processori usano una memoria unica, cui accedono tramite un bus:



Multiprocessore di base

Il problema di una simile architettura è che, se le CPU sono molte veloci, i conflitti sull’uso del bus sono troppi. Una alternativa per aggirare questo problema sarebbe quella di dotata ciascun processore di una quantità di memoria locale propria, non accessibile agli altri, da affiancare alla memoria condivisa:



Multiprocessore con memorie locali

La memoria locale può essere usata per il codice del programma e, in generale, per tutti gli elementi che non devono essere condivisi.

Altri processori non usano addirittura un unico bus, ma diversi bus per ridurre ulteriormente il carico. Altri ancora usano la tecnica della **memoria cache**, in cui tenere le parole della memoria usate più frequentemente.

Autore: **Sandro Petrizzelli**
e-mail: sandry@iol.it
sito personale: http://users.iol.it/sandry