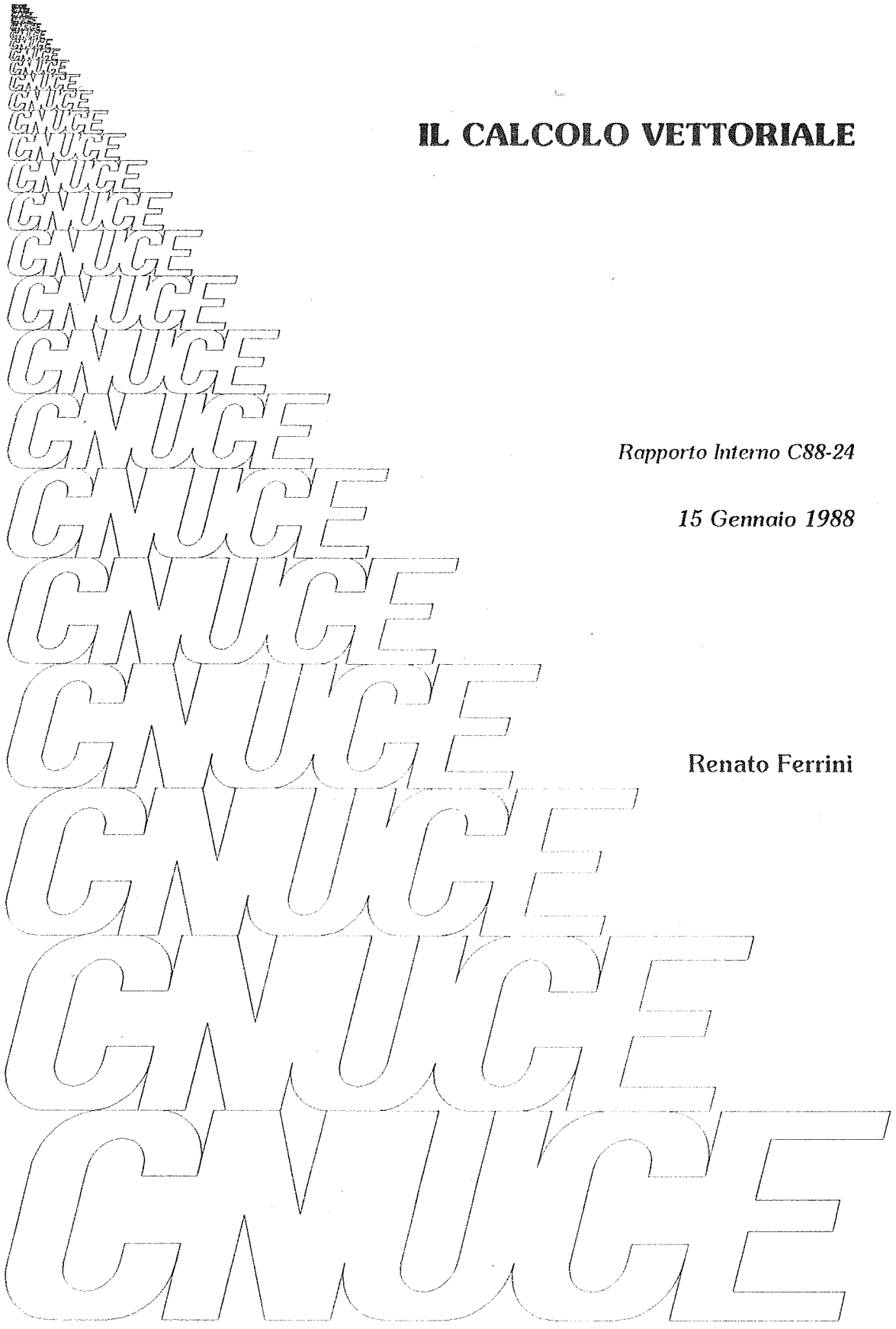

IL CALCOLO VETTORIALE

Rapporto Interno C88-24

15 Gennaio 1988

Renato Ferrini

IL CALCOLO VETTORIALE

15 Gennaio 1988

Renato Ferrini

CNUCE - Istituto del CNR
via Santa Maria 36
56100 PISA (Italy)

Prefazione

Una delle piu' importanti novita' introdotte in informatica negli ultimi anni e' rappresentata dal *Calcolo Parallelo*. L'elaborazione parallela dei dati ha permesso di raggiungere delle potenze di calcolo molto elevate che hanno consentito di far fronte alle sempre piu' pressanti esigenze delle applicazioni scientifiche.

Oggi e' possibile trovare sul mercato una larga varieta' di macchine che operano con tecniche parallele e la situazione e' in continua evoluzione.

In questo documento si vogliono affrontare i principi su cui si basano le tecniche del parallelismo, con particolare riguardo a quelle del calcolo vettoriale.

Contenuto

Figure

Introduzione ai calcolatori paralleli

In questi ultimi anni la richiesta di potenze di elaborazione sempre piu' elevate, necessarie in alcune discipline scientifiche come l'aerodinamica, la metereologia ed il trattamento delle immagini, ha favorito l'apparizione di nuovi tipi di calcolatori. L'aumento delle capacita' di calcolo ottenuto su tali macchine non e' dovuto tanto nelle innovazioni tecnologiche, che, anche se presenti, non sono essenziali, ma nella definizione di nuove architetture.

Il principio che consente di ottenere velocita' molto elevate si basa infatti su tecniche di funzionamento che permettono l'esecuzione parallela sia a livello di programma, o parti di programma, che a livello di singola istruzione.
Dato che il tipo di architettura gioca un ruolo fondamentale, cerchiamo adesso di indagare maggiormente su questo elemento.

Architettura di un calcolatore

L'architettura di un calcolatore puo' essere definita dai seguenti elementi:

- numero e tipo dei componenti hardware di base (es. processori, unita' di controllo, memorie, ecc...)
- tipo di struttura in cui i suddetti componenti si trovano connessi
- metodologie con cui e' possibilire utilizzare tali componenti (es. tipo di istruzioni, meccanismi di indirizzamento, modo di realizzazione delle operazioni di I/O, ecc...).

Da quanto esposto e' facile dedurre che l'architettura e' cio' che si presenta ad un programmatore che lavora a livello di linguaggio macchina, per cui e' necessario avere una conoscenza molto approfondita della macchina prima di scrivere la prima riga di codice. Anche se l'evoluzione dei linguaggi e dei compilatori ha ridotto l'influenza della architettura sulla programmazione, c'e' pero' da dire che, allo stato attuale, la programmazione vettoriale risente ancora delle implicazioni architetturali. Per tale motivo viene fornita di seguito una breve descrizione dei vari tipi di architetture utilizzate per la costruzione della maggior parte dei calcolatori presenti sul mercato.

Processori scalari

I processori scalari sono costituiti da un'unica unita' di elaborazione. Il loro funzionamento prevede l'esecuzione di un'istruzione alla volta che effettua operazioni logiche o aritmetiche su singoli dati. Per tale motivo questi tipi di processori sono anche classificati come SISD (Single Instruction stream/Single Data stream).

Anche sui calcolatori scalari monoprocessori sono state realizzate delle tecniche software, come la multiprogrammazione ed il time-sharing, che consentono di ottenere l'esecuzione *simultanea* di piu' processi. Pero' questa metodologia di funzionamento non puo' essere assimilata al concetto di parallelismo realizzato mediante tecniche hardware che permettono l'esecuzione *contemporanea* di piu' istruzioni.

In Figura 1 e' schematizzata l'architettura di un processore scalare.

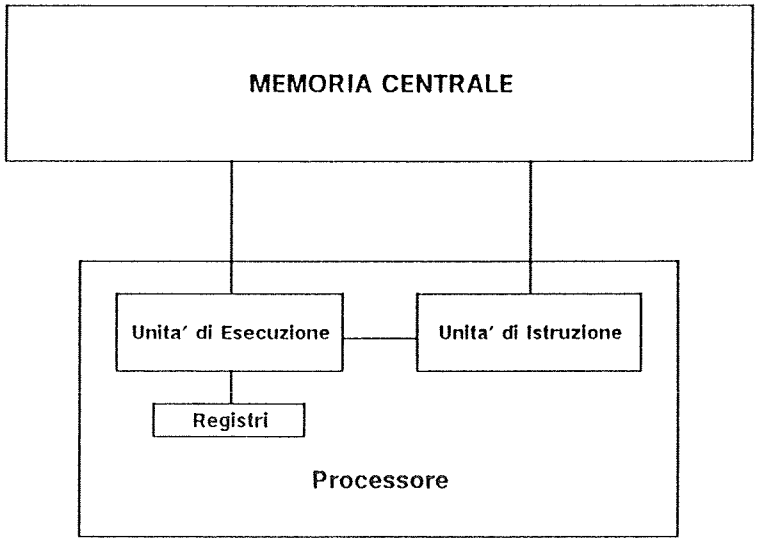

Figura 1. Processore scalare

Array Processor

I processori che appartengono a questa categoria usano un'architettura che consente di ottenere un parallelismo a livello della singola istruzione. Gli *Array Processor* sono composti da piu' processori in grado di eseguire operazioni aritmetiche e logiche ed ognuno dei quali e' corredato di una propria memoria. In altre parole, i vari processori corrispondono alle *Unita' di Esecuzione* dei processori scalari e sono gestiti da un'unita' di controllo a cui e' demandato il compito di provvedere alla distribuzione del calcolo tra le varie unita' di elaborazione. Inoltre l'unita' di controllo e' predisposta all'esecuzione su singoli dati **(istruzioni scalari).**

In figura Figura 2 a pag. 3 e' riportato lo schema di un Array processor.

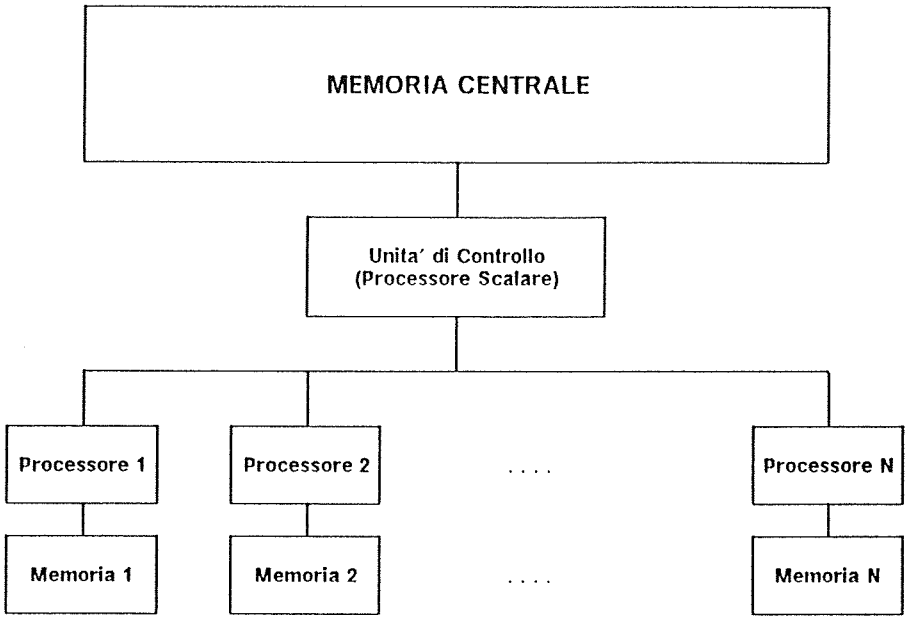

Figura 2. Array Processor

Cerchiamo di spiegare il funzionamento di un Array Processor mediante un esempio. Prendiamo il ciclo:

```
      DO  1  I = 1, 200
      A(I) = A(I) * B(I)
1     CONTINUE
```

Come si puo' vedere la moltiplicazione A(I)*B(I) deve essere ripetuta per i 200 elementi dei vettori A e B. In questo caso l' *Unita' di Controllo* decide di smistare le 200 operazioni tra i vari processori assegnando al primo la moltiplicazione A(1)*B(1), al secondo la moltiplicazione A(2)*B(2) e cosi' via. Si ottiene pertanto l'esecuzione simultanea di piu' operazioni e tale tecnica di funzionamento realizza un tipo di un parallelismo che viene definito **spaziale.**

Possiamo immaginare che durante l'esecuzione del suddetto ciclo DO, nell'Array Processor si presenti una situazione simile a quella riportata in Figura 3 a pag. 4

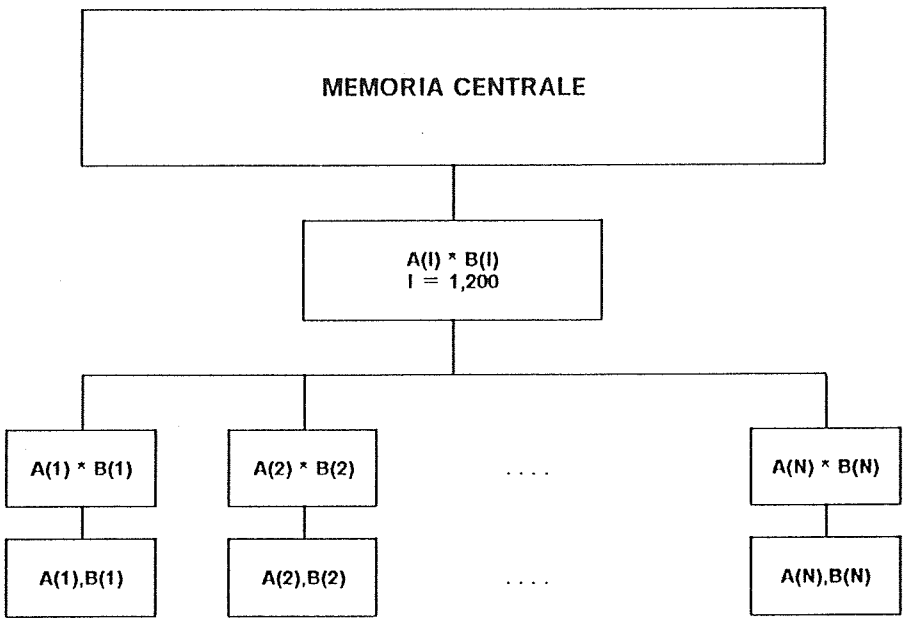

Figura 3. Funzionamento di un Array Processor

Dal funzionamento dell'Array Processor si deduce una considerazione importante: si ha un aumento della velocita' di esecuzione solo quando si deve ripetere una stessa operazione aritmetica su piu' dati. Per questo motivo e' necessario che il programma contenga dei **vettori** ad una o piu' dimensioni (Matrici), sui quali devono essere svolte le stesse operazioni aritmetiche mediante dei cicli DO.

La proprieta' di eseguire operazioni a livello di vettori ha introdotto in informatica una nuova terminolgia, per cui si parla di calcolatori vettoriali, di programmazione vettoriale, ecc... Inoltre per questa caratteristica gli Array Processor vengono definiti come macchine di tipo SIMD (Single Instruction stream/Multiple Data stream).

Cerchiamo adesso di quantificare l'aumento della velocita' di elaborazione ottenuta in un'operazione di moltiplicazione tra due vettori. In un calcolatore scalare, se T e' il tempo per eseguire tale operazione ed N e' il numero degli elementi del vettore, si puo' stimare il tempo di esecuzione dell'intero ciclo in N*T. In un Array Processor avente M processori della stessa potenza di quello scalare si possono presentare due casi:

- *Numero dei processori maggiore del numero degli elementi del vettore.*
  In questo caso tutte le N operazioni vengono eseguite in un unico passo per cui il tempo totale sara' T.

- *Numero dei processori minore del numero degli elementi del vettore.*

In questo caso saranno necessari N/M passi per completare l'operazione su tutti gli elementi del vettore, per cui il tempo totale risultera' di T*(N/M).

Il rapporto tra l'esecuzione vettoriale e quella scalare **(speedup),** che indica il guadagno di potenza, risulta pertanto di 1/N nel primo caso e di 1/M nel secondo.

Tutto questo pero' avviene in presenza di istruzioni su vettori **(istruzioni vettoriali),** altrimenti le istruzioni scalari vengono gestite direttamente dall'Unita' di Controllo in modo sequenziale e non portano quindi alcun beneficio nella velocita' di esecuzione.

Processori di tipo Pipeline

Anche i processori di tipo *pipeline* realizzano un parallelismo a livello della singola istruzione come gli Array Processor; ma, a differenza di questi ultimi, il parallelismo e' ottenuto mediante la *sovrapposizione* delle operazioni da eseguire. Questa tecnica permette di ottenere un parallelismo di tipo **temporale.**

Andiamo adesso ad analizzare in dettaglio il funzionamento dei processori di tipo pipeline; cosa che pero' richiede la conoscenza preliminare di alcune informazioni riguardo alla struttura interna di un processore. In un processore, l'esecuzione delle operazioni e' *scandita* da un tempo che si rende necessario per la propagazione dei segnali da un circuito ad un altro e che e' chiamato **ciclo base.** Normalmente l'esecuzione di un'istruzione aritmetica si scompone in varie sottooperazioni, ognuna delle quali necessita di uno o piu' cicli base. Ad esempio la moltiplicazione in virgola mobile tra due numeri reali si suddivide nelle seguenti sottooperazioni di un ciclo base ciascuna:

- *Operazione 1 (OP1):* Aggiustamento delle caratteristiche.
- *Operazione 2 (OP2):* Moltiplicazione delle mantisse.
- *Operazione 3 (OP3):* Addizione delle caratteristiche.
- *Operazione 4 (OP4):* Normalizzazione del risultato.

In un processore scalare la moltiplicazione in virgola mobile tra due vettori A e B, puo' essere rappresentata come in Figura 4 a pag. 6 dove M1 indica la moltiplicazione tra i primi elementi dei vettori, M2 tra i secondi elementi, ecc...

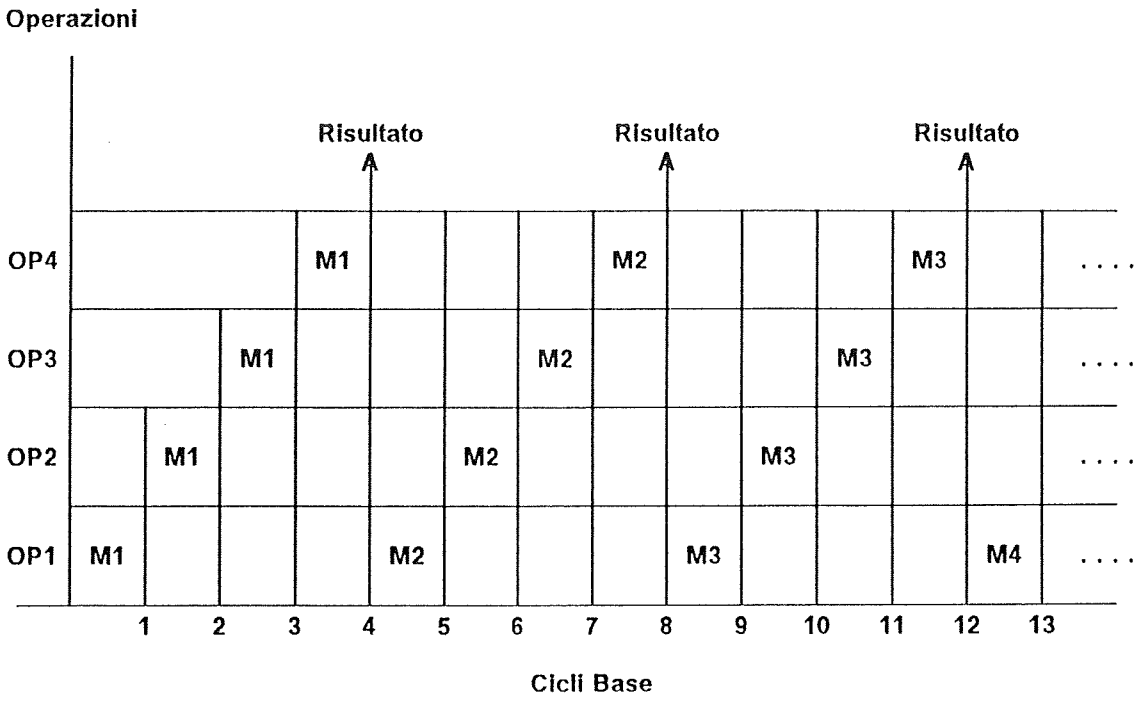

Figura 4. Esecuzione della moltiplicazione in un processore scalare

Come si puo' notare dalla figura, in un processore scalare la moltiplicazione successiva non inizia fino a quando tutte le operazioni precedenti non sono state completate. In questo caso, se N e' il numero di elementi dei due vettori, si puo' stimare il tempo necessario per l'esecuzione di tutta l'istruzione $A(I) = A(I)*B(I)$ in 4*N cicli base.

Per migliorare la velocita' di esecuzione si e' pensato di dotare i processori di una struttura, chiamata **pipeline**, per mezzo della quale e' possibile sovrapporre i vari passi dell'esecuzione per i diversi elementi del vettore. In questo modo, quando e' in esecuzione la seconda sottooperazione della prima moltiplicazione, si puo' iniziare il primo passo per la seconda moltiplicazione e cosi' via, come e' mostrato in Figura 5 a pag. 7

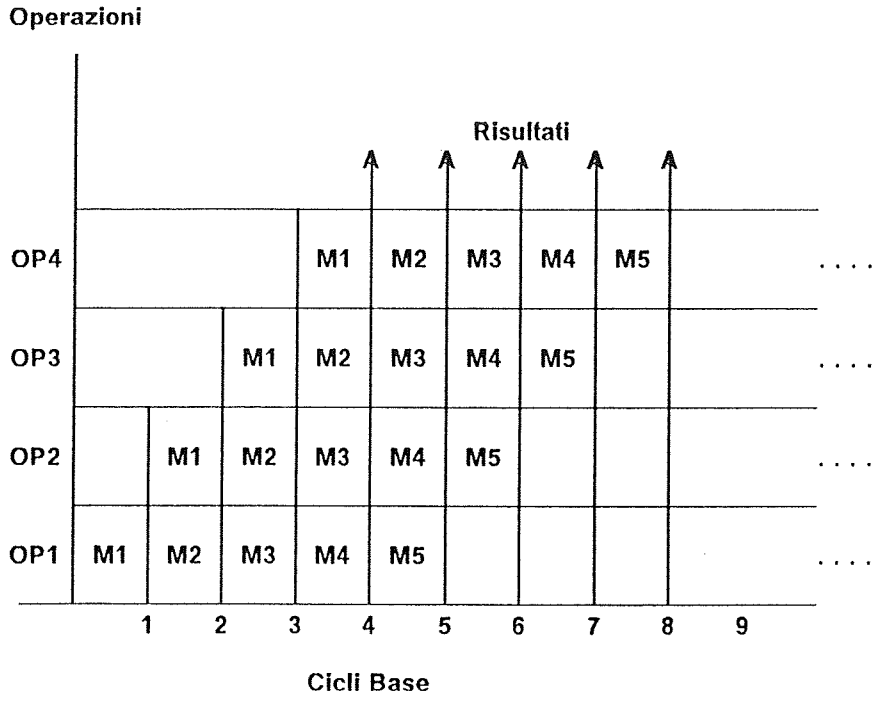

Figura 5. Esecuzione della moltiplicazione in un processore pipeline

Dal funzionamento della pipeline si puo' dedurre che, dopo i cicli spesi per l'inizializzazione **(Startup),** che nel caso della moltiplicazione in virgola mobile sono 3, si ottiene un risultato per ogni ciclo base. Quindi il tempo totale per ottenere i risultati di una moltiplicazione di due vettori aventi N elementi e' di 3+N cicli, raggiungendo uno *speedup* di (3+N)/N*4 che per valori di N molto grandi si avvicina ad 1/4.

Per il modo di lavorare i processori di tipo pipeline sono classificati SIMD come gli Array Processor. Inoltre i processori pipeline hanno in comune con gli Array Processor un altro elemento e cioe' la caratteristica di ottenere un guadagno di potenza solo in presenza di istruzioni che lavorano su vettori.

Multiprocessori

I *Multiprocessori* hanno la pecularieta' di avere un'architettura comprendente piu' processori di tipo scalare che si dividono la stessa memoria centrale. La differenza tra questi ultimi e gli Array Processor consiste nel fatto che le unita' di elaborazione dei Multiprocessori sono indipendenti le une dalle altre poiche' ognuna e' provvista di una propria *Unita' di Istruzione* e di una propria *Unita' di Esecuzione.*

Questa caratteristica permette percio' l'esecuzione parallela di insiemi di istruzioni diverse che lavorano su dati diversi, per cui il

Multiprocessori vengono classificati come processori di tipo MIMD (Multiple Instruction stream/Multiple Data stream).

In Figura 6 e' riportato lo schema di un Multiprocessore.

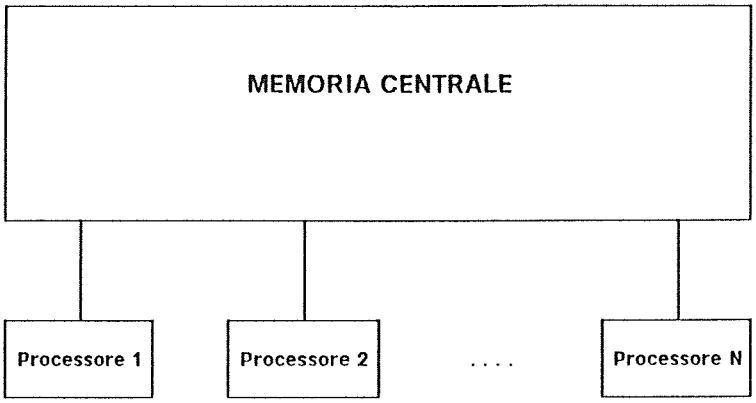

Figura 6. Multiprocessore

Per la loro architettura, i Multiprocessori possono eseguire contemporaneamente piu' programmi, uno per ogni processore presente **(Calcolo Parallelo).** Il compito di gestire questo meccanismo e' affidato completamente al sistema operativo ed e' quindi trasparente all'utente.

Il parallelismo puo' pero' avvenire anche a livello di parti di programma ed in questo caso e' necessario che il programmatore indichi, con opportune istruzioni, quali unita' di programma possono essere eseguite in maniera indipendente.

A differenza degli Array Processor e dei processori pipeline, nei multiprocessori la potenza di calcolo e' indipendente dal tipo di elaborazione, anche se spesso qualcuno dei processori e' provvisto di una pipeline.

Architettura e organizzazione dell'IBM 3090-VF

Generalita'

Il calcolatore IBM 3090 nasce in seguito alla richiesta di utenti di sistemi IBM per la realizzazione di una macchina che avesse le seguenti caratteristiche:

- buone prestazioni scalari
- medie prestazioni vettoriali
- architettura multiprocessore
- taglia di memoria centrale fino a 256 MB
- precisione standard di 64 bit
- trasparenza per i sistemi operativi IBM
- costo ragionevole

Prima di spiegare che cosa si intende per medie prestazioni vettoriali e' necessario introdurre due nuovi concetti: il **Megaflops** ed il **livello di vettorizzazione.**

Il Megaflops e' l'unita' di potenza usata per i processori vettoriali e rappresenta il numero di milioni di operazioni aritmetiche in virgola mobile per secondo. Questa nozione costituisce pero' un valore indicativo in quanto puo' essere influenzato dai tempi di accesso ai dati, dal tipo di rappresentazione dei numeri e dal tipo delle operazioni aritmetiche.

Il livello di vettorizzazione di un programma fornisce invece una valutazione del numero di operazioni su vettori che possono essere gestite da un Array Processor o da un processore di tipo pipeline. Questo dato e' espresso dalla percentuale del tempo di esecuzione scalare che puo' essere svolto in modo vettoriale; quindi dire che un programma e' vettorizzabile al 50% significa che meta' del suo tempo di esecuzione e' vettorizzabile.

I concetti appena spiegati consentono adesso di precisare che l'IBM 3090-VF e' stato progettato specificatamente per l'esecuzione di programmi che hanno un livello di vettorizzazione compreso tra il 50% ed il 70%, poiche' questo tipo di applicazioni sono tra le piu'

comuni nel campo scientifico. Con tale tipologia di carico l'IBM 3090-VF riesce ad ottenere una potenza effettiva di 30 Megaflops.

Un processore cosi' strutturato si pone percio' come soluzione intermedia tra i Multiprocessori ed i super-calcolatori capaci di raggiungere una potenza di diverse centinaia di Megaflops. Quanto detto e' schematizzabile come riportato in Figura 7

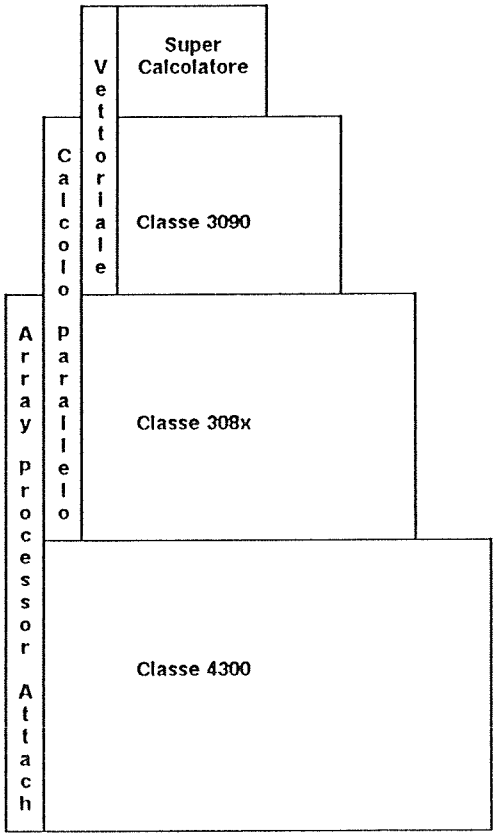

Figura 7. Posizione del 3090-VF tra gli elaboratori scientifici

Struttura scalare del 3090

Non e' nostra intenzione entrare nei dettagli per quanto riguarda la tecnologia costruttiva della serie 3090; diciamo semplicemente che l'unita' centrale e' a base di chips ECL (Emitter Coupled Logic) assemblati su TCM (Thermal Conduction Module) e che il ciclo base e' di 17.2 ns, contro i 24 ns del 3081-KX. E' importante invece elencare gli elementi che fanno parte della struttura del 3090.

Unita' centrale del 3090

La configurazione dell'unita' centrale del 3090, che e' schematizzata in Figura 8 a pag. 11, si compone di:

- da 1 a 6 unita' di calcolo
- un controllore di sistema
- un processore di canali
- una memoria centrale
- una memoria estesa (opzionale).

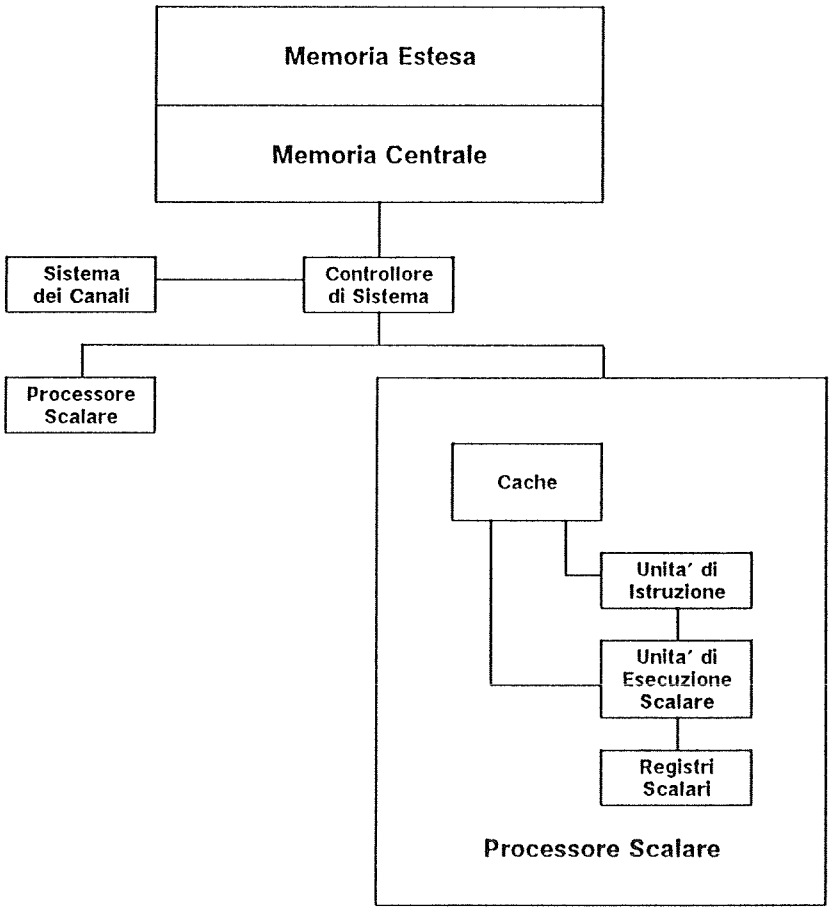

Figura 8. Struttura del 3090 scalare

L'organizzazione generale e' detta *diadica*, cioe' le unita' centrali, se presenti, sono identiche e si dividono l'accesso alla memoria centrale ed all'insieme dei canali di I/O. La gestione delle unita' centrali e del processore dei canali e' a carico del controllore di sistema che provvede anche ad un utilizzo ottimale di tali risorse.

Gerarchie della memoria.

La struttura della memoria si compone di piu' livelli ognuno dei quali costituisce un elemento della gerarchia e quindi gioca un ruolo importante nelle prestazioni globali del sistema. I vari livelli formano una struttura piramidale nella quale gli elementi piu' alti

rappresentano le memorie piu' veloci, ma anche le piu' costose e quindi le piu' limitate come dimensioni.

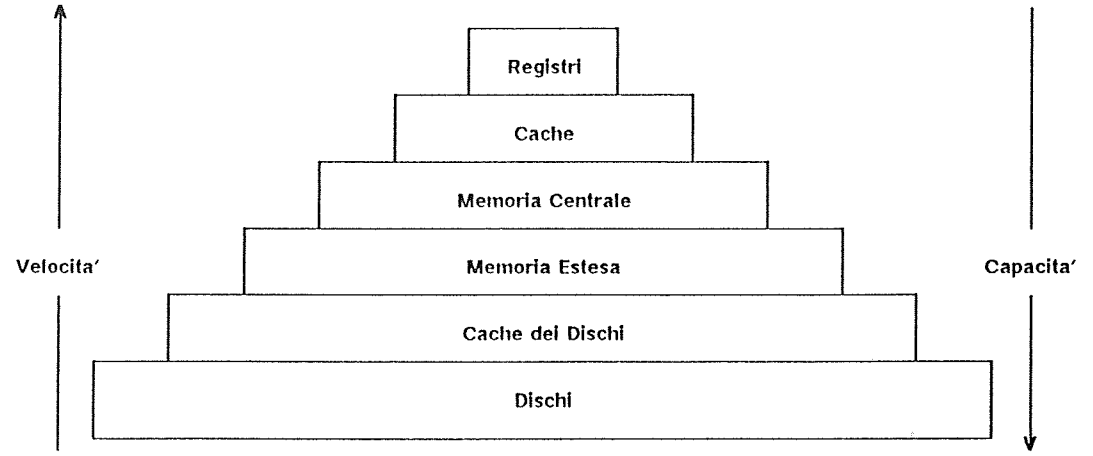

Figura 9. Gerarchia della memoria.

L'accesso alle informazioni, siano istruzioni di un programma o dati da elaborare, rappresenta uno dei problemi piu' importanti che influiscono sulle prestazioni di una macchina. Attualmente la tecnologia usata nella costruzione dei processori consente di effettuare le operazioni logiche o aritmetiche in pochi nanosecondi, mentre la richiesta di dati e la loro ricezione dalle unita' di memorizzazione richiede dei tempi che possono andare da qualche centinaio di nanosecondi, per la memoria centrale, a qualche millisecondo, per i dischi. Questa differenza dei tempi di accesso tra i vari supporti di memorizzazione puo' provocare degli squilibri durante la fase di esecuzione, a causa dei quali l'unita' di calcolo potrebbe passare la maggior parte del suo tempo ad aspettare la disponibilita' delle informazioni. D'altra parte le memorie veloci necessitano di una tecnologia sofisticata e quindi costosa, che limita necessariamente la loro capienza.

Si e' quindi pensato di risolvere il problema inserendo nell'architettura piu' unita' di memorizzazione aventi tempi di accesso sempre crescenti e creando quindi una gerarchia di memoria cosi' composta:

- *Registri*

  I registri sono delle zone di memoria situate all'interno dell'unita' centrale, con la quale condividono il ciclo base e cioe' il tempo che intercorre tra la richiesta di un'informazione e la sua disponibilita' alle unita' di calcolo. L'elaborazione dei dati contenuti nei registri consente pertanto di operare in maniera sincrona con le unita' di calcolo e cio' rappresenta la situazione piu' favorevole per le prestazioni di una macchina. I registri possono anche essere gestiti da un programmatore che lavora a livello di linguaggio macchina.

- *Cache*

  Essendo i costi dei registri molto elevati, in genere il loro numero e' molto limitato per cui la maggior parte delle informazioni da elaborare dovrebbe essere situata nella memoria centrale introducendo cosi' dei tempi di ritardo per l'accesso dei dati di circa 10 volte il ciclo base dell'unita' centrale. Per ovviare a tale inconveniente e' stato trovato un compromesso con l'introduzione di una memoria di primo livello chiamata *Cache* (o *High Speed Buffer*), che ha una piccola capacita' (64 KB per processore nel 3090), ma e' molto veloce e direttamente accessibile dall'unita' di calcolo. La sua gestione, che lo fa sembrare come *una finestra di accelerazione* posta davanti alla memoria centrale, e' interamente a carico dall'hardware, rendendolo dunque completamente trasparente al programmatore.

- *Memoria Centrale*

  La programmazione e' facilitata dalla disponibilita' della memoria centrale di grandi dimensioni, che permette di ridurre, o di eliminare, la segmentazione dei programmi e di aumentare la quantita' di dati da elaborare contemporaneamente, consentendo una notevole diminuzione delle operazioni di I/O. La memoria centrale e' raggiungibile direttamente dalle unita' di calcolo anche se queste ultime devono sprecare alcuni cicli base per ottenere le informazioni richieste.
  Le taglie della memoria centrale disponibili sul 3090 possono andare da 32 MB sul modello piu' piccolo fino a 256 MB sul modello piu' grande.

- *Memoria Estesa*

  Attualmente quasi tutti i sistemi operativi adottano la tecnica della memoria virtuale per rendere disponibile agli utenti grandi quantita' di spazio indirizzabile. Tutto cio' e' ottenuto mediante una simulazione che prevede la suddivisione della memoria virtuale in due parti, di cui una e' residente nella memoria centrale e l'altra, generalmente la piu' consistente, e' memorizzata su disco. Il passaggio da un supporto all'altro dipende dalla utilizzazione o meno di certe zone della memoria virtuale ed avviene attraverso un'operazione di *paging* o di *swap*.

  Come gia' detto il tempo d'accesso alle informazioni memorizzate sui dischi si esprime in decine di millisecondi, che costituisce un tempo considerevole in rapporto al ciclo base dell'unita' centrale (qualche nanosecondo). Sulla serie 3090 e' stata dunque introdotta una memoria estesa che ha un tempo d'accesso inferiore al microsecondo. Tale memoria, che e' interna all'unita' centrale, viene gestita direttamente dal sistema operativo alla stessa stregua della memoria centrale, per cui, riducendo considerevolmente il carico di paginazione e di swap da e verso i dischi, si migliorano le prestazioni globali del sistema. Sui vari modelli della serie 3090 la memoria estesa puo' arrivare fino a 1024 MB.

- *Cache dei dischi*

  La *Cache dei dischi* e' un tipo di supporto strutturato come la memoria centrale con la sola differenza che e' situata al di fuori dell'unita' centrale, per cui e' necessaria un'operazione di I/O per l'accesso ai dati. Il vantaggio della cache rispetto ai dischi deriva dalla mancanza di operazioni meccaniche per il posizionamento sulle informazioni da trasferire; cio' permette di espletare l'operazione in pochi millisecondi.

- *Dischi*

  I dischi sono dei supporti esterni di memorizzazione la cui capacita' puo' arrivare a diverse centinaia di Megabyte a costi molto contenuti. Il reperimento delle informazioni richiede pero' una sequenza di operazioni sia meccaniche che elettroniche che determina dei tempi di accesso molto elevati rispetto a quelli dell'unita' centrale.

- *Ricerca di informazioni*

  Con una gerarchia di memoria cosi' strutturata, l'accesso ad una una informazione da parte dell'unita' centrale avviene nel seguente modo:

  - l'informazione si trova in un registro:
    in questo caso il tempo di accesso e' uguale al ciclo base dell'unita' centrale;

  - l'informazione e' nella memoria di primo livello (Cache):
    anche in questo caso il tempo di accesso e' uguale al ciclo base dell'unita' centrale;

  - l'informazione e' in memoria centrale:
    deve essere prelevata e scritta nella Cache e quindi il tempo di accesso e' uguale al ciclo della memoria centrale;

  - l'informazione e' in memoria virtuale:
    in questo caso, il sistema operativo deve intervenire per trasferirla in memoria reale; il tempo d'accesso sara' quindi dell'ordine dei microsecondi, se l'informazione e' memorizzata nella memoria estesa, o dei millisecondi, se l'informazione si trova nella cache dei dischi o su disco.

Il ruolo della memoria di primo livello e' particolarmente efficace nei casi in cui si referenzino le stesse istruzioni e/o dati molto frequentemente, poiche' aumenta la probabilita' che questi si trovino gia' nella Cache. Per tale motivo i benefici maggiori si hanno nei programmi scientifici, dove non e' raro passare piu' dell' 80% del tempo su qualche decina di istruzioni.

Architettura dell'unita' di calcolo vettoriale

L'unita' di calcolo vettoriale e' opzionale e puo' essere integrata in qualunque processore scalare presente nella configurazione del 3090-VF, come e' mostrato nella Figura 10 a pag. 16

La struttura dell'unita' di calcolo vettoriale del 3090-VF si basa sull'uso della pipeline per l'esecuzione delle operazioni aritmetiche su numeri interi o reali. Con tale organizzazione il processore vettoriale del 3090-VF e' di tipo SIMD, ma poiche' la **Vector Facility (VF)** puo' essere presente su piu' processori, la struttura globale della macchina puo' percio' essere classificata come MIMD.

Andiamo adesso ad analizzare le varie componenti della Vector Facility che sono schematizzate come in Figura 11

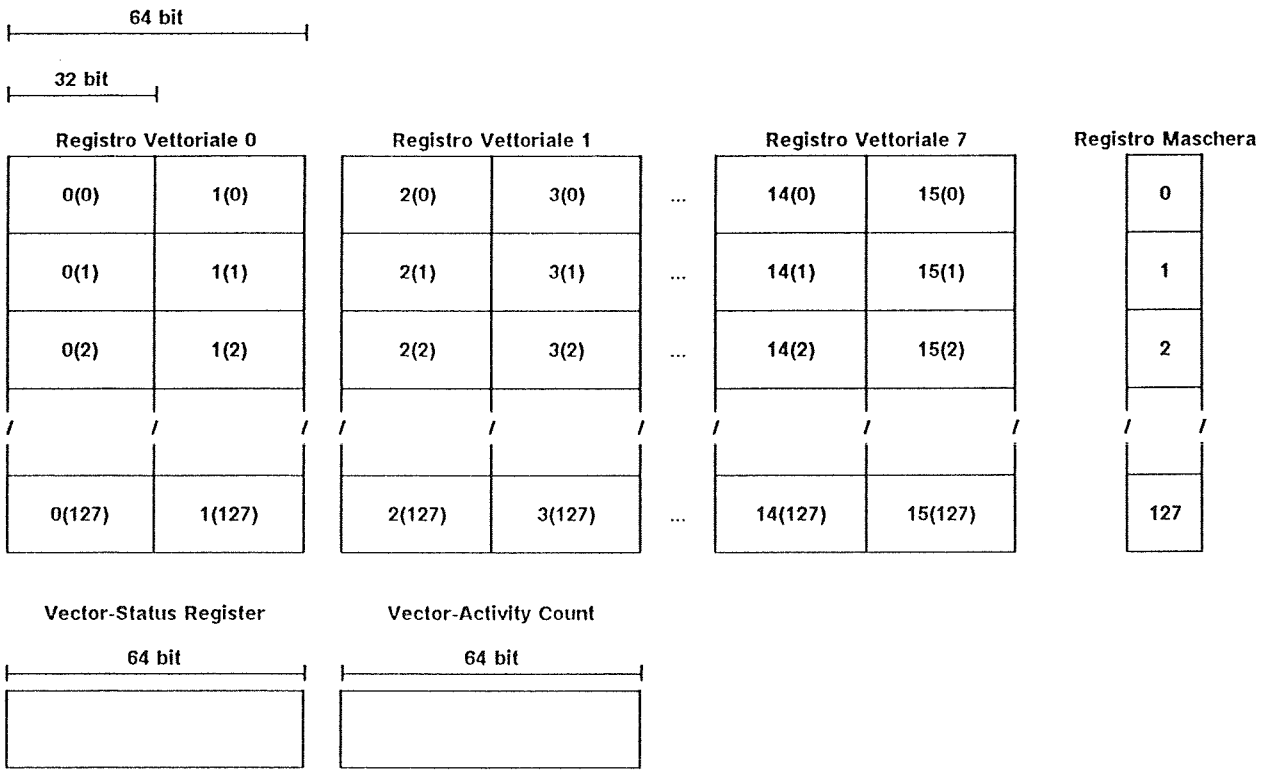

Figura 11. Schema dei registri dell'unita' vettoriale del 3090-VF

Registri vettoriali

Per poter fornire un risultato per ciclo, la pipeline deve essere alimentata in ingresso ad una velocita' sufficiente, ma poiche' il ciclo base dell'unita' centrale e' piu' rapido di quello della memoria centrale, e' stato necessario fornire la macchina di registri vettoriali. Il ruolo dei registri vettoriali, che sono delle memorie locali situate nella VF, unito a quello della Cache e' di eliminare i tempi di attesa della pipeline che pregiudicherebbero delle buone prestazioni.

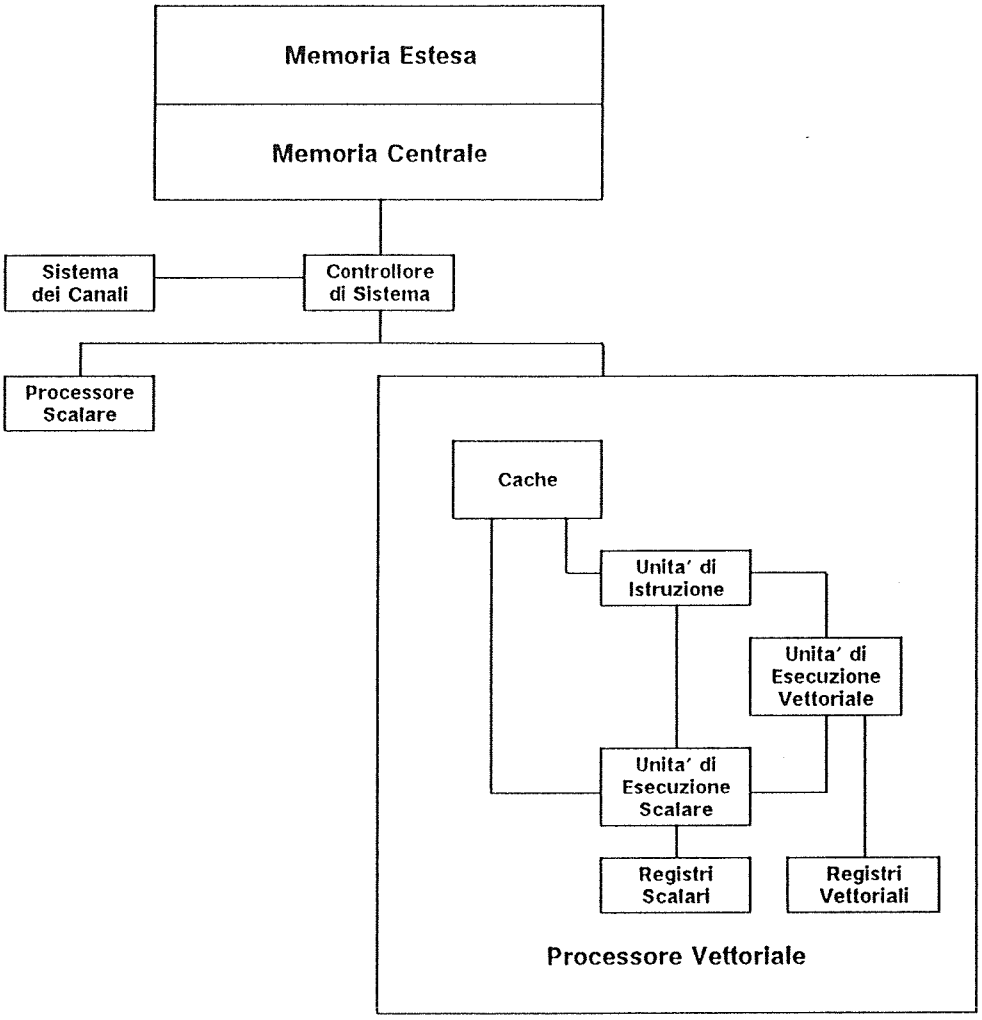

Figura 10. Processore vettoriale del 3090-VF

L'architettura vettoriale del 3090 dispone di 8 registri vettoriali in doppia precisione (64 bit), riconfigurabili anche come 16 registri in semplice precisione (32 bit). Ognuno dei registri puo' contenere 128 elementi aventi ciascuno una lunghezza di 32 o 64 bit. La scelta, per un modello particolare, della dimensione dei registri vettoriali e' in funzione del rapporto prezzo/prestazioni, in quanto, in generale, l'architettura vettoriale stabilisce che il numero di elementi contenuto in un registro vettoriale **(Section Size)** puo' essere una qualunque potenza di 2 compresa fra 8 e 512. E' percio' importante precisare che questa scelta di 128 elementi operata per il 3090-VF puo' essere cambiata su altri modelli.

Registro Maschera

Il *Registro Maschera* permette di elaborare un'operazione vettoriale solo su alcune coppie di elementi dei vettori in esame. Quindi in una situazione come la seguente:

Registro	Mascera	0	1	0	0	1	.
Vettore	A	A(1)	A(2)	A(3)	A(4)	A(5)	.....
Vettore	B	B(1)	B(2)	B(3)	B(4)	B(5)	.....

l'operazione vettoriale si verifichera' solo sulle coppie A(2)-B(2) , A(5)-B(5) , ecc...

Il *Registro Maschera* viene percio' utilizzato per le operazioni condizionali (come l'istruzione COMPARE) ed e' costituito di 128 bit, ognuno dei quali corrisponde ad un elemento del vettore. L'istruzione corrente sara' eseguita in base allo stato del bit (0 o 1) corrispondente all'elemento da elaborare.

Vector mask mode

Il *Vector mask mode* e' un indicatore di 1 bit che segnala se l'elaborazione deve interessare tutti gli elementi del vettore oppure deve coinvolgere un sottoinsieme di essi selezionato mediante l'uso del registro maschera.

Vector status register

Il *Vector Status Register* contiene le informazioni necessarie per il controllo della Vector Facility, come:

- l'indicazione se il *Vector Mask Mode* e' attivo o meno
- le informazioni sulle istruzioni vettoriali interrotte
- la lunghezza dell'attuale vettore da elaborare
- ecc...

Vector activity count.

Associato al Vector Status Register, il *Vector Activity Count* permette di distribuire e di controllare l'utilizzo delle risorse vettoriali (analisi dei lavori, tuning, ecc..).

Unita' di esecuzione vettoriale

L'unita' di esecuzione vettoriale del 3090 e' composta dagli elementi rappresentati nella Figura 12 a pag. 18

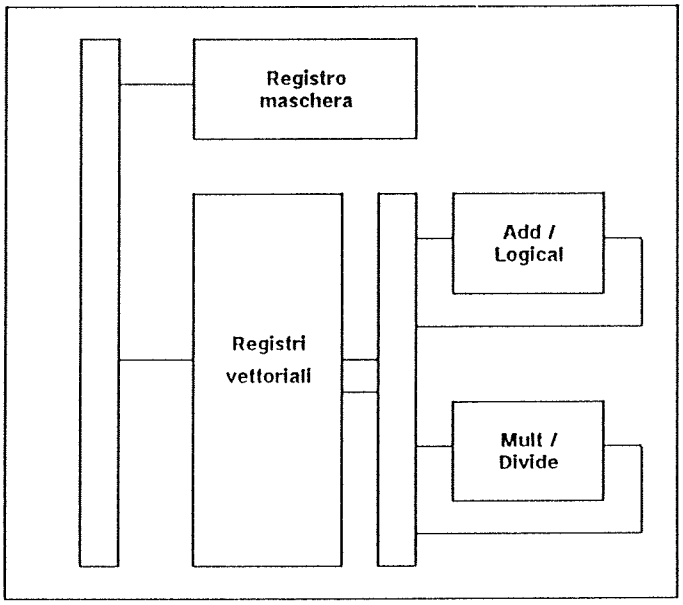

Figura 12. Unita' di esecuzione vettoriale

La Vector Facility del 3090 dispone di 2 pipeline, di cui la prima, chiamata *Add/Logical* e' utilizzata per le addizioni, le sottrazioni, i confronti e le operazioni logiche, mentre la seconda, chiamata *Mult/Divide* e' utilizzata per le operazioni di moltiplicazione e divisione.

Come gia' detto, la caratteristica di una pipeline e' che, una volta inizializzata, fornisce un risultato ad ogni ciclo-macchina; tuttavia, delle due pipeline presenti nel 3090, una sola puo' essere in funzione in un dato momento, a meno che non sia in corso di esecuzione una istruzione *composta,* costituita dalle operazioni di prodotto e somma, per cui avviene il **concatenamento.** Una istruzione composta, come il Multiply and Add, crea un flusso di dati tale che l'uscita della pipeline Mult/Divide corrisponde all'ingresso della pipeline Add/Logical per fornire un risultato ad ogni ciclo, benche' l'istruzione consista in due operazioni aritmetiche.

Istruzioni vettoriali

Per il supporto della Vector Facility sono state introdotte 171 nuove istruzioni di tipo vettoriale, utilizzabili a livello di linguaggio macchina. Queste nuove istruzioni elaborano i dati sia in singola che in doppia precisione ed inoltre possono eseguire operazioni aritmetiche e logiche su vettori posti sia nella memoria centrale che nei registri vettoriali. Altre istruzioni permettono di realizzare la funzione di concatenamento (MULTIPLY and ADD), che consente l'esecuzione di due operazioni aritmetiche per ciclo.

Le istruzioni vettoriali possono avere fino a 3 operandi per referenziare sia i registri vettoriali, sia i registri scalari, sia l'indirizzo di memoria contenente il primo elemento di un vettore. In

- *Con indirizzamento sequenziale:*
  quando gli elementi del vettore sono ordinati in posizioni contigue della memoria, o in posizioni separate distanti tra da loro da un *passo* costante. Questo passo costante viene chiamato **STRIDE** e vedremo in seguito che puo' avere un peso importante sulle prestazioni.

- *Con indirizzamento indiretto:*
  quando gli elementi sono posizionati senza un ordine preciso ed il loro caricamento avviene sotto il controllo del Vector Mask Register.

Poiche' la dimensione dei registri vettoriali e' di 128 elementi, l'elaborazione vettoriale viene effettuata a gruppi di 128 **(sezione del vettore)** con esclusione dell'ultimo gruppo che puo' avere una dimensione o sezione inferiore.

Potenza di calcolo vettoriale

A conclusione della descrizione delle caratteristiche del 3090-VF cerchiamo di calcolare la massima potenza istantanea **(potenza di picco)** della VF che si verifica durante un operazione di concatenamento, in cui vengono eseguite 2 istruzioni per ciclo, e quando i 128 elementi sono gia' caricati nel registro vettoriale. Poiche' il ciclo base e' di 17.2 ns, ne deriva che la VF puo' operare ad una velocita' massima di 116 Megaflops quando esegue una istruzione composta (2 istruzioni/17.2 nanosecondi). Tale risultato e' pero' ottenibile per un tempo di 2.2 microsecondi che e' il tempo necessario per l'elaborazione dei 128 elementi presenti nel registro vettoriale.
E' evidente che ci sara' interesse a sfruttare le prestazioni date dal concatenamento, ogni volta che sara' possibile.

Se si dispone di piu' unita' di calcolo vettoriale, che possono essere fino a 6, e si utilizzano le possibilita' del 'Multitasking', si potra' raggiungere una potenza *teorica* di 696 Megaflops.

Software disponibile sul 3090-VF

L'architettura del 3090-VF ha richiesto l'aggiornamento del software gia' esistente per la gestione sia dei nuovi elementi scalari, come la memoria estesa, che della Vector Facility. Di seguito daremo un cenno del nuovo software che permette di effettuare il calcolo vettoriale sul 3090-VF, poiche' alcuni prodotti saranno oggetto di una dettagliata analisi in seguito.

Software di base

Una delle caratteristiche di progettazione del 3090-VF era quella di essere gestito dai sistemi operativi tradizionali della IBM, in quanto cio' permette di far uso del calcolo vettoriale direttamente dall'am-

biente *general purpose*. L'unita' di calcolo vettoriale e' percio' supportata dai seguenti sistemi operativi:

- MVS
- VM

Compilatore VS FORTRAN

Poiche' le capacita' vettoriali dell' IBM 3090-VF sono rivolte soprattutto all'ambiente scientifico, il linguaggio che permette di utilizzare la Vector Facility dell'IBM 3090 e' il **Fortran.** Di questo prodotto e' stata creata una nuova versione che comprende:

- *Compilatore VS FORTRAN Versione 2*
  Questo compilatore e' interamente compatibile con il VS FORTRAN Versione 1 con l'aggiunta di un parametro che permette la vettorizzazione del codice fortran a vari livelli.
- *Nuovo Debug Interattivo (IAD)*
  Il Debug Interattivo Versione 1 e' stato esteso con il supporto di nuove possibilita'.
- *Nuova Libreria Fortran*
  La libreria Fortran e' stata aggiornata con l'inserimento di nuove funzioni e l'ottimizzazione di quelle gia' esistenti.

Libreria ESSL

E' stata inoltre creata una nuova libreria, chiamata *Engineering and Scientific Subroutine Library (ESSL)* che e' composta da un insieme di sottoprogrammi matematici scritti in modo da sfruttare, in maniera ottimale, le caratteristiche scalari e vettoriali dell'IBM 3090-VF.

Programma di Conversione LCP

Questo programma, chiamato *Language Conversion Program (LCP)* converte le istruzioni di codici sorgente scritti in Fortran IBM Livello 66 (Fortran G1, HX, Q) e del VS FORTRAN Versione 1 in istruzioni equivalenti del VS Fortran Livello 77, facilitando cosi' la standardizzazione dei programmi al livello 77.