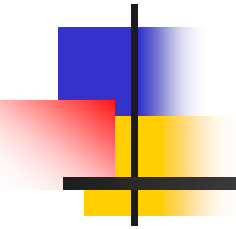
Corso di Calcolatori Elettronici (per Elettronici)



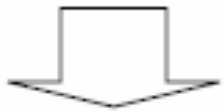
Parte I : Organizzazione generale del Calcolatore

Prof. Giandomenico Spezzano

Problema

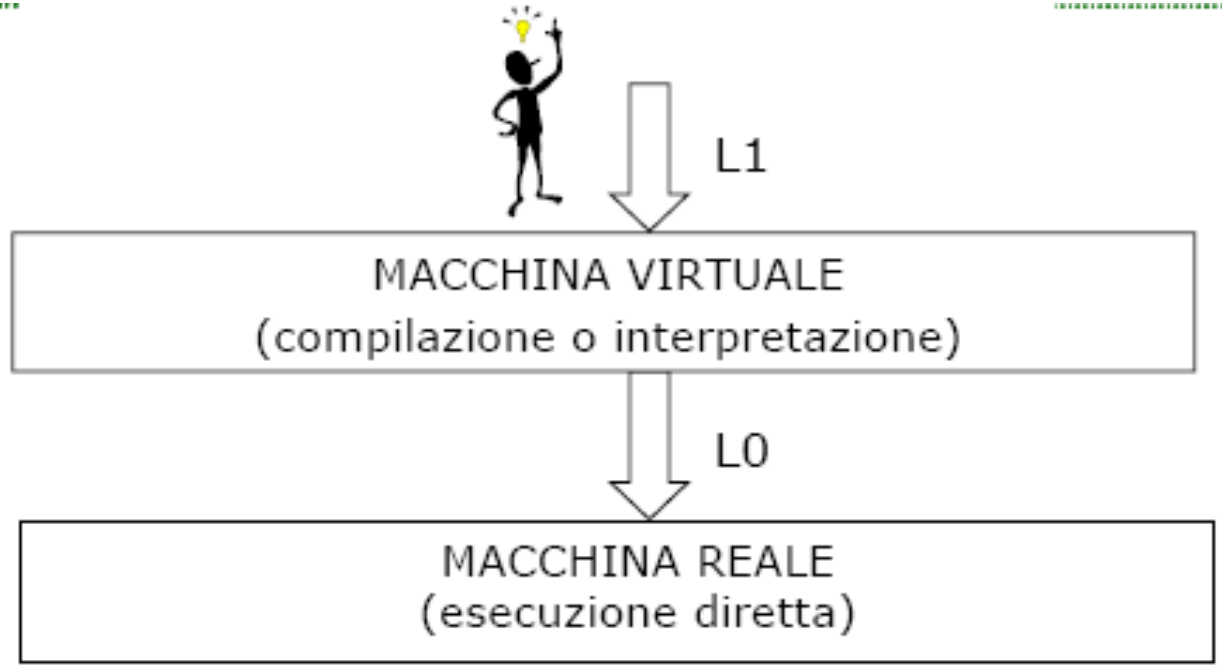
Voglio fare X (programma complesso in un linguaggio L1 di alto livello)

Traduzione o interpretazione



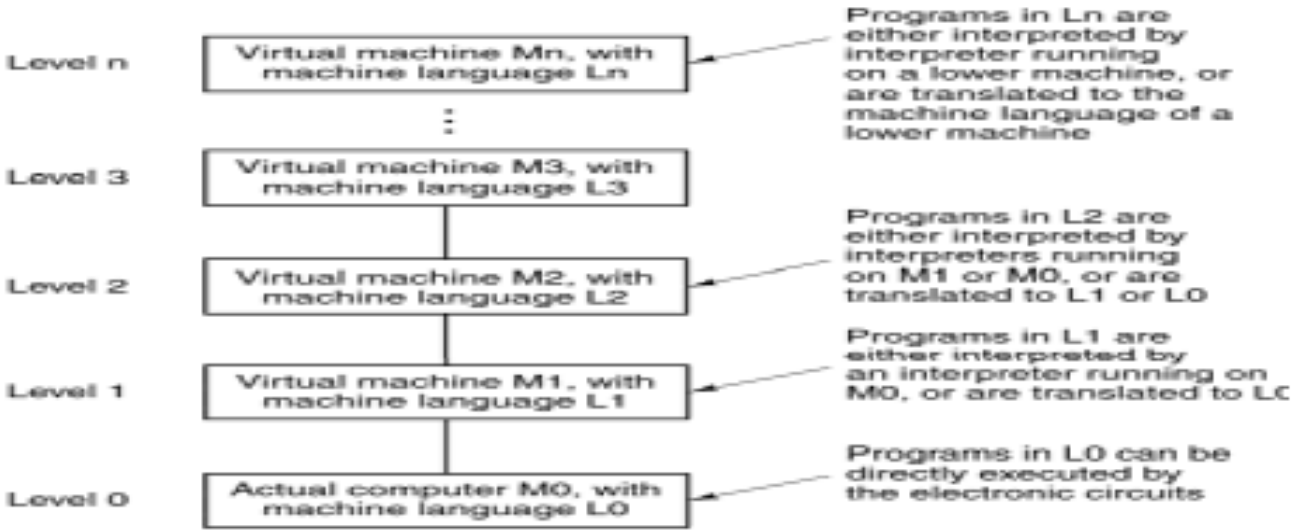
So fare Y (programma di istruzioni semplici in un linguaggio macchina L0)

Soluzione adottata



Se L0 ed L1 sono troppo diversi il problema si decompone introducendo livelli intermedi

Architettura a livelli

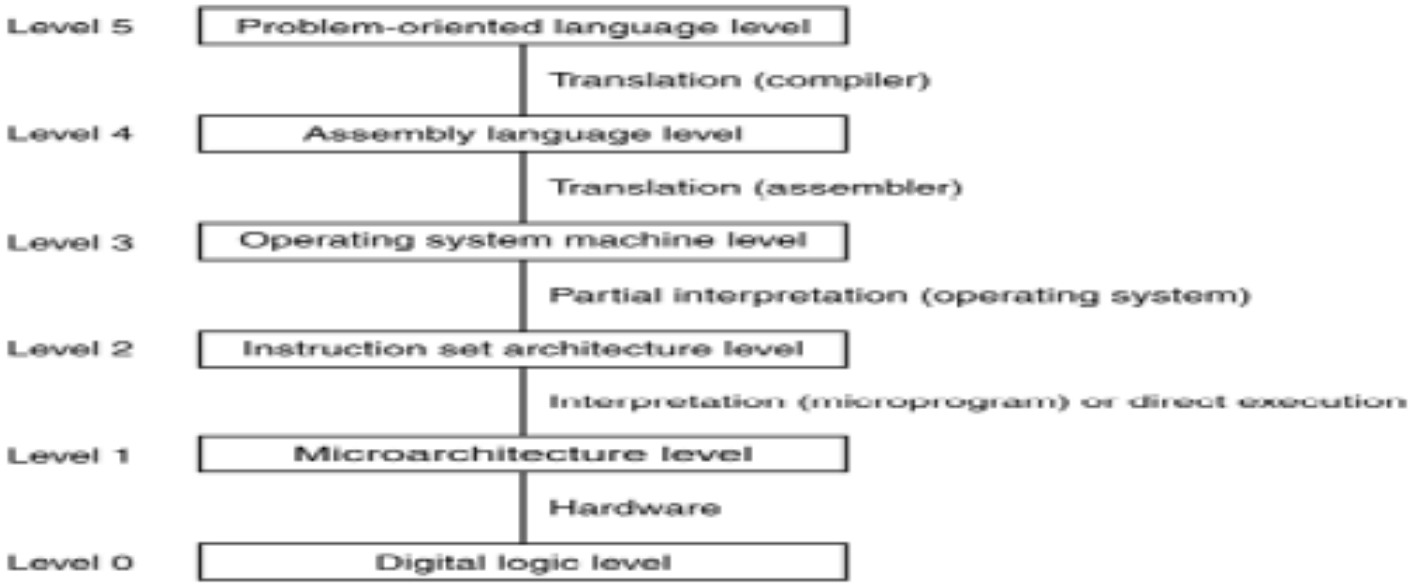


- Al livello i corrispondono una macchina virtuale M_i ed un linguaggio L_i
- Il linguaggio L_i è tradotto nel linguaggio L_{i-1} o interpretato da un programma che gira sulla macchina M_{i-1}

Perchè la stratificazione

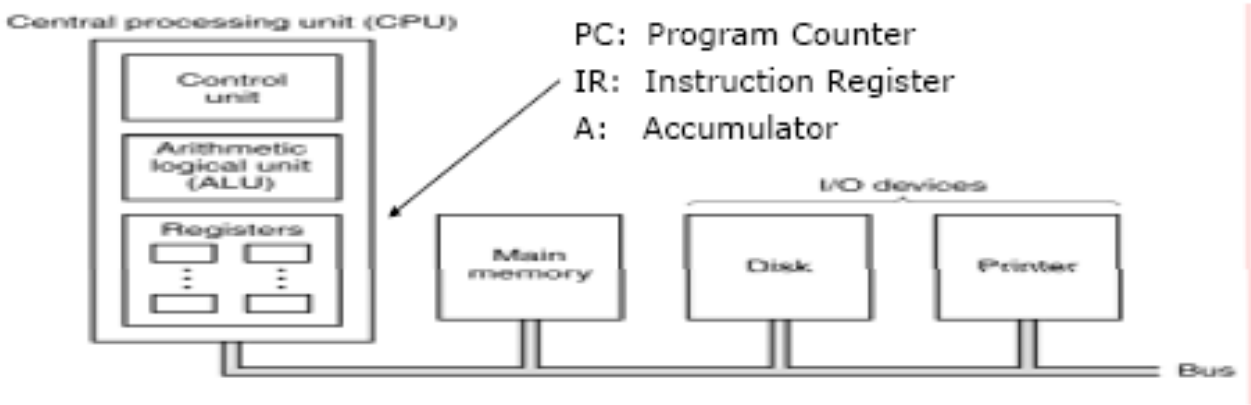
- M_0 è facilmente realizzabile in hardware, ma difficile da programmare
- M_n è facile da programmare ma impossibile da realizzabile in hardware
- Implementazione progressiva e modulare
- Trasparenza per l'utente e le applicazioni
- Il linguaggio L_n non dipende dalla piattaforma (hardware) M_0:
  - Diversi linguaggi disponibili sulla stessa piattaforma
  - Lo stesso linguaggio disponibile su diverse piattaforme

Tipica strutuura a livelli



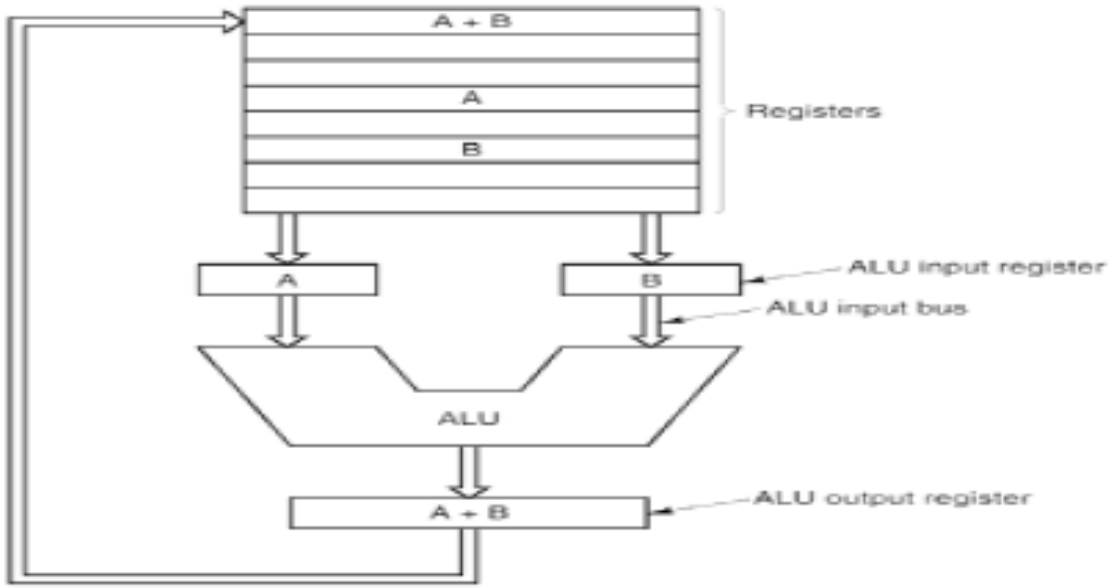
- Il livello 2 è il più basso al quale un utente può programmare la macchina (confine tra software e hardware)
- Normalmente si programma a livello 5

Struttura del computer



- La memoria contiene sia i dati che le istruzioni
- Il contenuto dei registri può essere scambiato con la memoria e l'I/O
- Le istruzioni trasferiscono i dati e modificano il contenuto dei registri
- Registri particolari:
  - PC: indirizza l'istruzione corrente
  - IR: contiene l'istruzione corrente

Struttura della CPU



- Esecuzione di operazioni aritmetiche e logiche sui dati contenuti nei registri
- Spostamento di dati fra registri e fra registri e memoria
- Ciclo elementare: due operandi sono inviati alla ALU e il risultato e messo in un registro

Il ciclo Fetch-Decode-Execute

L'esecuzione di ciascuna istruzione nella CPU consta dei seguenti passi:

1. Carica l'istruzione da memoria in IR (Instruction Register) (**Fetch**)
2. Incrementa PC (Program Counter)
3. Decodifica l'istruzione (**Decode**)
4. Se l'istruzione usa un dato in memoria calcolane l'indirizzo
5. Carica l'operando in un registro
6. Esegui l'istruzione (**Execute**)
7. Torna al passo **1.** Per l'esecuzione dell'istruzione successiva

Accessi alla memoria sono effettuati sempre al passo 1, e non sempre ai passi 4 e 5

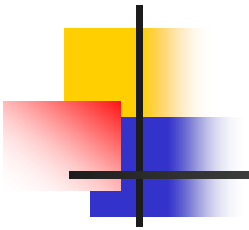
Esecuzione e interpretazione

Esecuzione diretta

- Le istruzioni possono venire eseguite direttamente dai circuiti hardware
- Approccio molto complesso:
  - Repertorio di istruzioni limitato
  - Progettazione dell'HW complessa
  - Esecuzione molto efficiente

Interpretazione

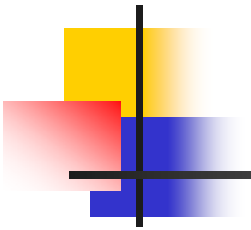
- L'hardware può eseguire solo alcune operazioni elementari molto semplici dette microistruzioni
- Ciascuna istruzione è scomposta in una successione di microistruzioni poi eseguite dall'hardware
- Vantaggi:
  - Repertorio di istruzioni esteso
  - HW più compatto
  - Flessibilità di progetto



La microprogrammazione

- Eseguire direttamente le istruzioni più frequenti
- Massimizzare la frequenza alla quale le istruzioni sono eseguite misurata in MIPS (Millions of Instr. per Second)
- Semplificare la decodifica delle istruzioni: formati molto regolari
- Limitare i riferimenti alla memoria alle sole LOAD e STORE
- Ampliare il numero di registri per limitare l'uso di LOAD e STORE

N.B. Questi principi sono tipici della filosofia RISC ma anche le architetture CISC vi si adeguano almeno in parte



CISC e RISC

Architetture **RISC** (*Reduced Instruction Set Computer*):

- Esecuzione diretta
- Repertorio ristretto (alcune decine)
- Istruzioni prevalentemente su registri
- *Una istruzione per ciclo di macchina*

Architetture **CISC** (*Complex Instruction Set Computer*)

- Interpretazione tramite microprogramma
- Repertorio esteso (alcune centinaia)
- Istruzioni anche su memoria
- *Molti cicli di macchina per istruzione*

Esempi:

- Alpha (DEC), SPARC: RISC
- Pentium II/III/IV (Intel): CISC

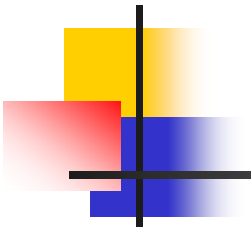
All'inizio degli anni '80 i progettisti di sistemi veloci riconsiderano l'approccio dell'*esecuzione diretta*

Principi progettuali dei computer moderni

- Eseguire direttamente le istruzioni più frequenti
- Massimizzare la frequenza alla quale le istruzioni sono eseguite misurata in MIPS (Millions of Instr. per Second)
- Semplificare la decodifica delle istruzioni: formati molto regolari
- Limitare i riferimenti alla memoria alle sole LOAD e STORE
- Ampliare il numero di registri per limitare l'uso di LOAD e STORE

N.B. Questi principi sono tipici della filosofia RISC ma anche le architetture CISC vi si adeguano almeno in parte

Vari tipi di parallelismo



Il parallelismo è ormai l'unica strada per aumentare le prestazioni
Limite di un'esecuzione sequenziale: velocità della luce (30 cm in 1 nsec)

Due tipi di parallelismo:

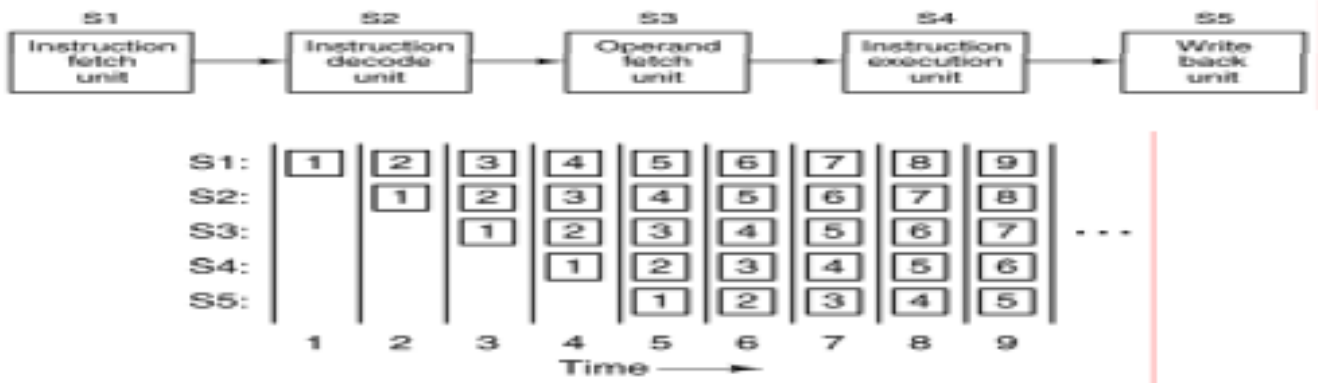
A) a livello di istruzioni

- *Diverse istruzioni* eseguite insieme
- *Diverse fasi* della stessa istruzione eseguite concorrentemente

B) a livello di processori

- *Molti processori* lavorano insieme allo stesso problema
- Fattori di parallelismo molto elevati
- Diversi tipi di interconnessione e di cooperazione (più o meno stretta)

Pipeling



- Ciascuna istruzione è divisa in fasi
- L'esecuzione avviene in una *pipeline* a più stadi
- Più istruzioni in esecuzione contemporanea
- Una istruzione completata per ogni ciclo

N.B. Si guadagna un fattore pari al numero di stadi della pipeline

Caratteristiche di una pipeline

Una pipeline consente un compromesso tra:

- Latenza: tempo per eseguire una istruzione
- Ampiezza di banda: numero di istruzioni completate per unità di tempo misurata in MIPS (milioni di istruzioni al secondo)

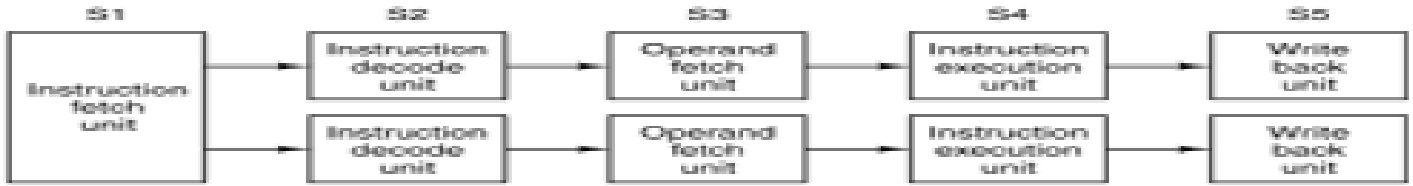
Con:

- Velocità di clock = T nsec
- Numero di stadi = n

Abbiamo:

- Latenza = nT
- Ampiezza di banda = 1 istr. ogni T nsec, ovvero: $10^9/T$ istr. ogni sec., ovvero: $1000/T$ MIPS

Architetture superscalari



Si aumenta il parallelismo avendo più di una pipeline nel microprocessore

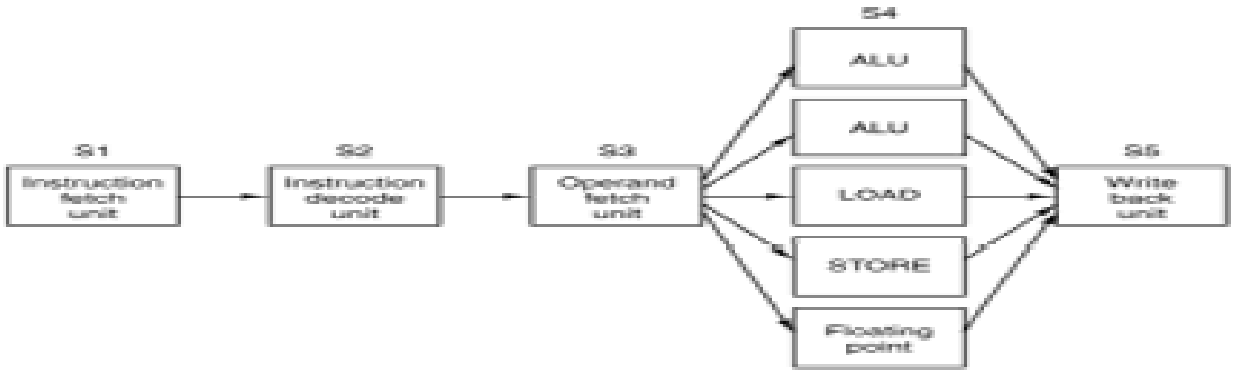
Le pipeline possono essere specializzate:

- Una versione del Pentium ha due pipeline a più stadi
- Una può eseguire solo istruzioni su interi

Problema: compatibilità dell'esecuzione parallela

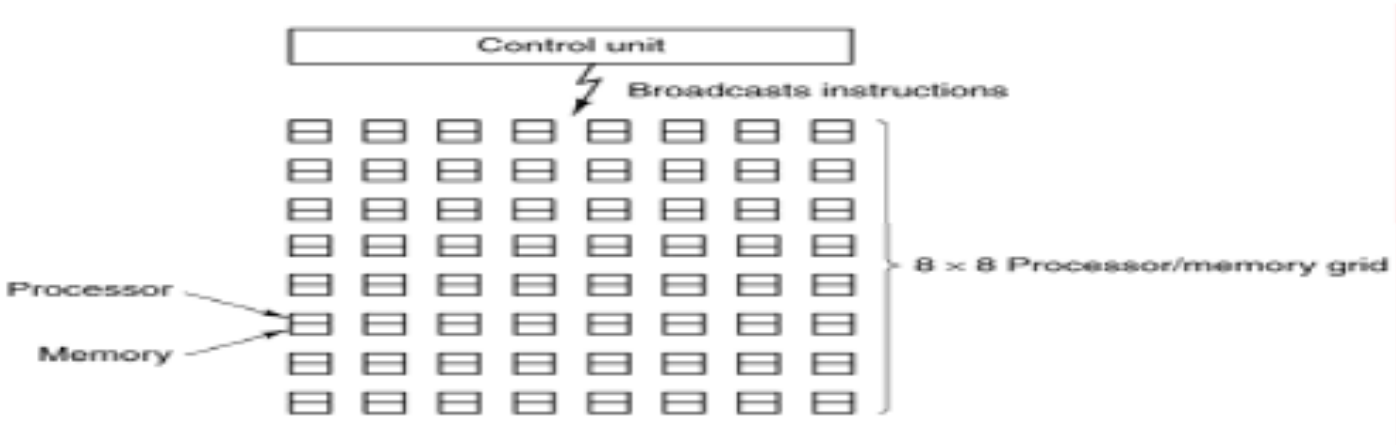
- Indipendenza tra le istruzioni
- Ciascuna istruzione non deve utilizzare i risultati dell'altra

Unità funzionali multiple



- La CPU contiene al suo interno diverse unità funzionali indipendenti
- Lo stadio più lento della pipeline (che condiziona la velocità) viene "multiplato" (parallelizzato)
- Architettura adottata nei supercomputer
- Prima forma di architettura superscalare

Array processor

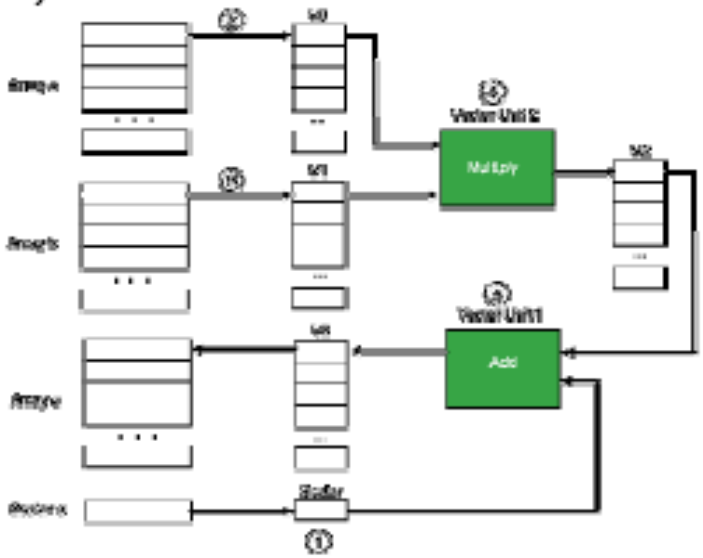


Array computer: processori identici ed autonomi eseguono le stesse istruzioni su dati diversi

- **ILLIAC IV** (1974): 50 Mflops

Vector processor

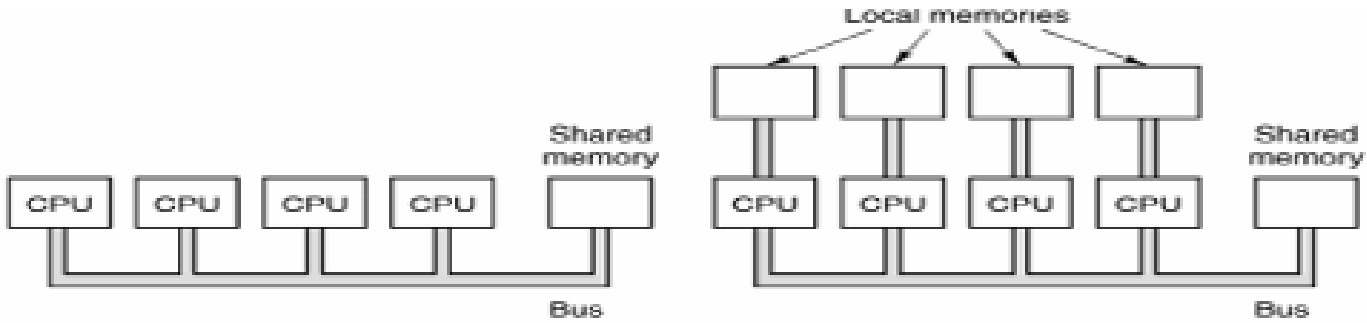
Vector processor: operano in parallelo su registri vettoriali (supercomputer)



Array e Vector processor sono esempi di:

SIMD (Single Instruction Multiple Data)

Multiprocessor



Le CPU lavorano indipendentemente

Shared memory: il bus può divenire collo di bottiglia

Private memory: contiene il codice e parte dei dati

Scambio dati tramite la shared memory

Multicomputers: i singoli elementi sono normali Workstation o PC

MIMD (Multiple Instruction Multiple Data)