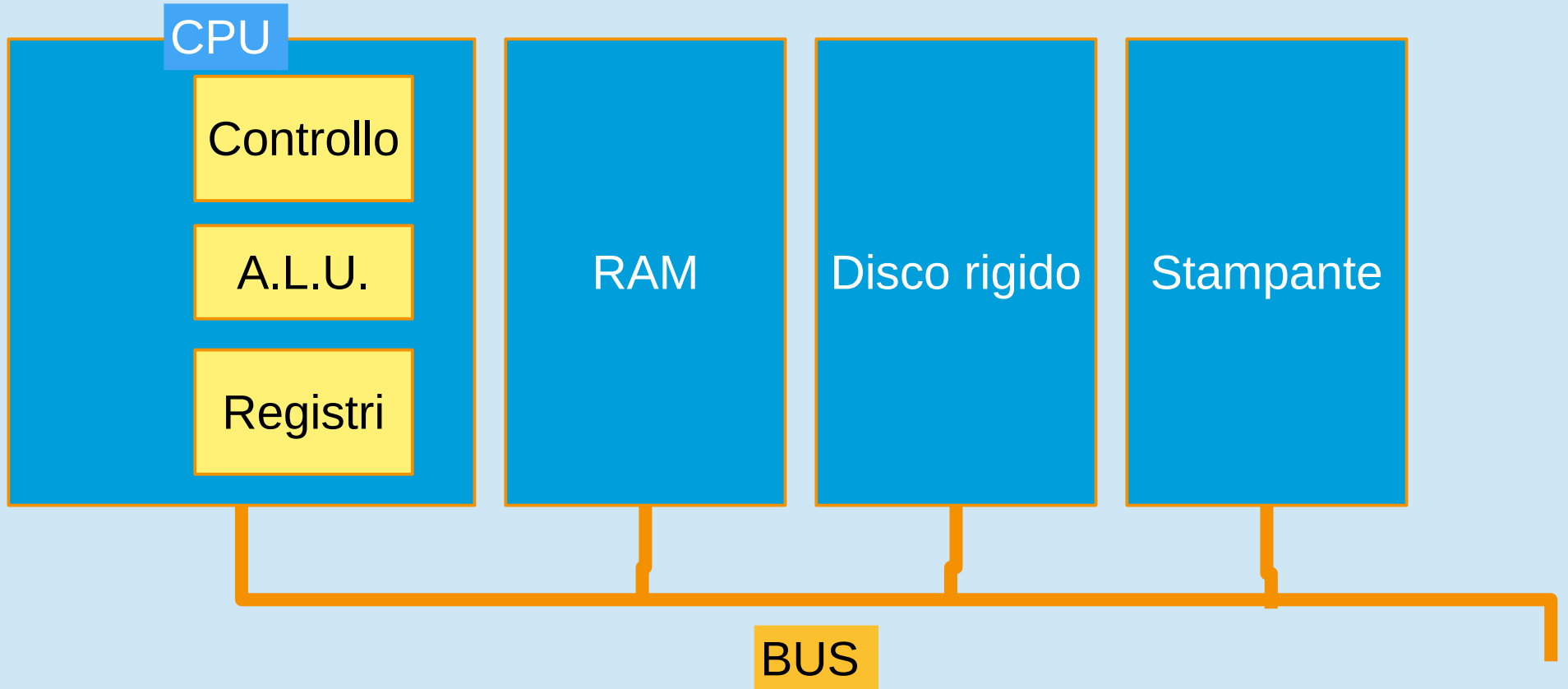


Architetture di calcolo

01 Processori

Organizzazione del Computer

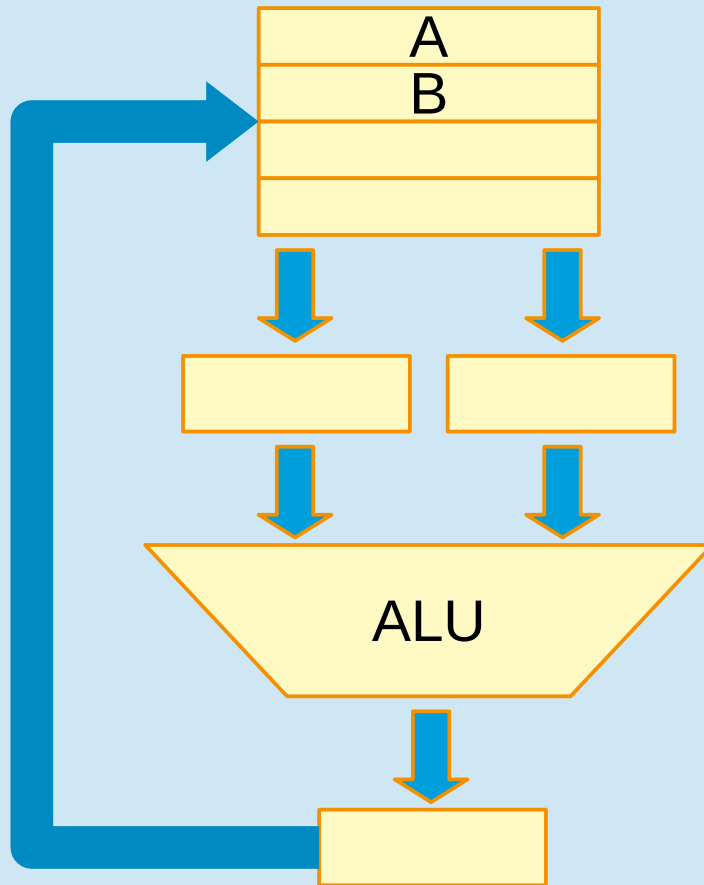


Organizzazione del Computer

Instruction SET

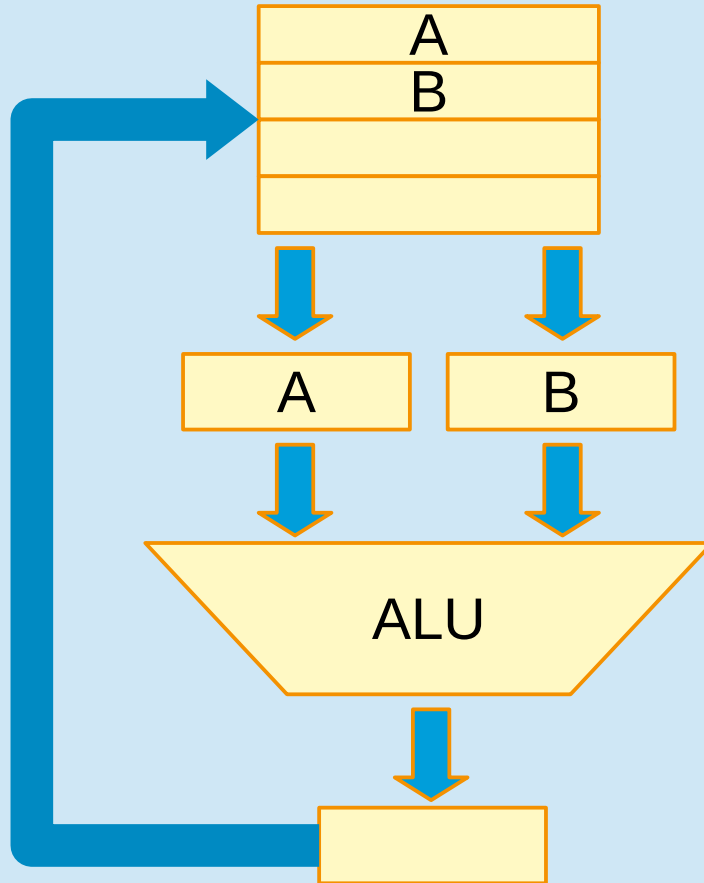
E' l'insieme (set) di istruzioni supportate dalla CPU.

ALU - 1



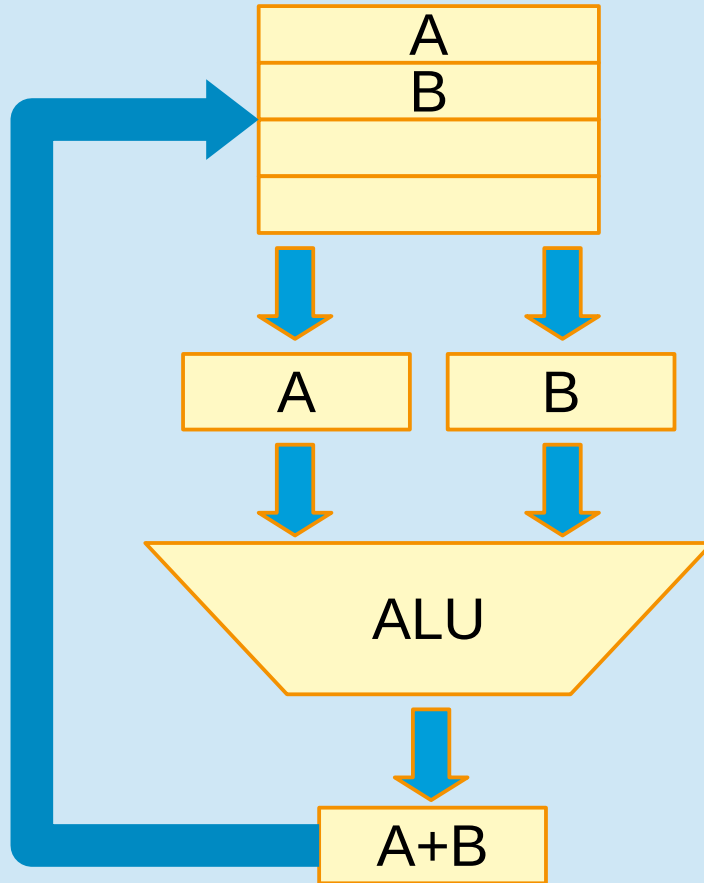
Operazione
selezionata: SOMMA

ALU - 1



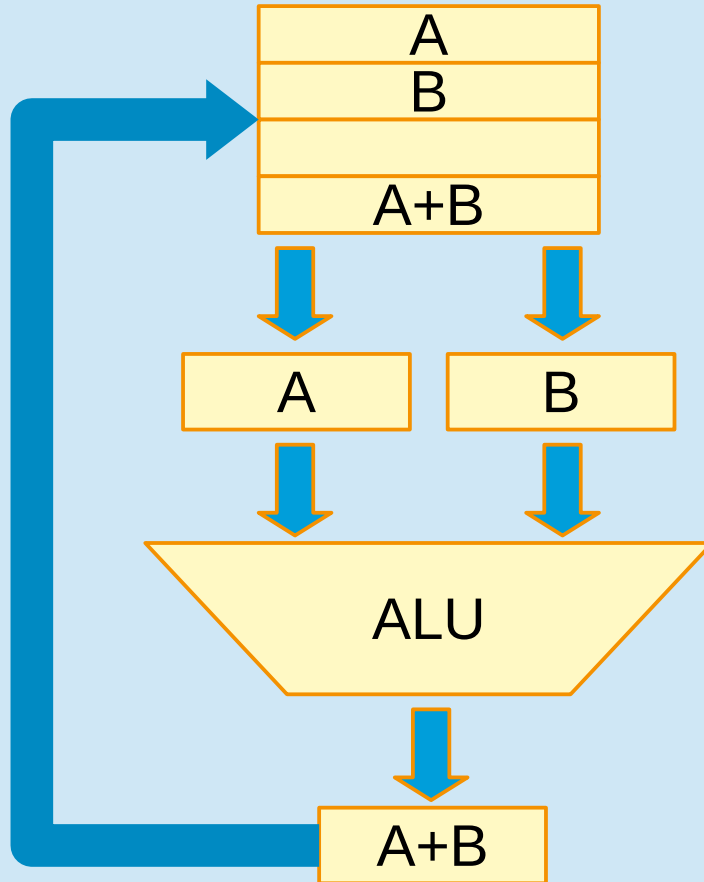
Load operandi da registri CPU a registri ALU

ALU - 2



Esecuzione

ALU - 3



Copia risultato da
registri ALU a
registri CPU

Esecuzione istruzioni

Fasi di esecuzione

- 1.FETCH : Legge istruzione
- 2.DECODE : Decodifica istruzione
- 3.MEMORY : Legge da memoria
- 4.EXECUTE : Esegue istruzione
- 5.WRITE BACK : Scrive su memoria

Interprete in versione Software

L'esecuzione istruzioni può essere eseguita anche da un **programma software**, invece che da strutture hardware.

In questo caso il programma si chiama **'INTERPRETE'** perché interpreta i comandi

Esempio di interprete software

```
Int PC = 0 // La prima istruzione numero 0
Int AC // Registro accumulazione
Int INSTR
Int INSTR_TYPE
Int DATA_ADDRESS
Int DATA

While True:
    INSTR = Memory(PC)
    PC = PC + 1
    INSTR_TYPE = Decode(INSTR)
    DATA_ADDRESS = Fetch(INSTR_TYPE)
    if DATA_ADDRESS > 0:
        data = Memory(DATA_ADDRESS)
    Execute(INSTR_TYPE, DATA)
```

Fase Execute: Scrive internamente sul registro AC

Fase Write Back: Assente perche non scrive su memoria

Motivi di un interprete software

L'interprete dei comandi, realizzato in software, e' utile perche consente di ampliare l'INSTRUCTION SET senza realizzare nuovo hardware (sarebbe molto costoso) .

L'interprete si comporta come una CPU simulata, che viene chiamata 'Virtual Machine'.

Questa tendenza negli anni 80 ha causato la proliferazione di sistemi con moltissime istruzioni complesse chiamate **CISC**:
Complex Instruction Set Computer

Struttura a strati con interprete

Linguaggio Arricchito - da Interpretare

Interprete (Macchina Virtuale)

Linguaggio semplice - Macchina CPU

CPU (Macchina Fisica)

CISC e RISC

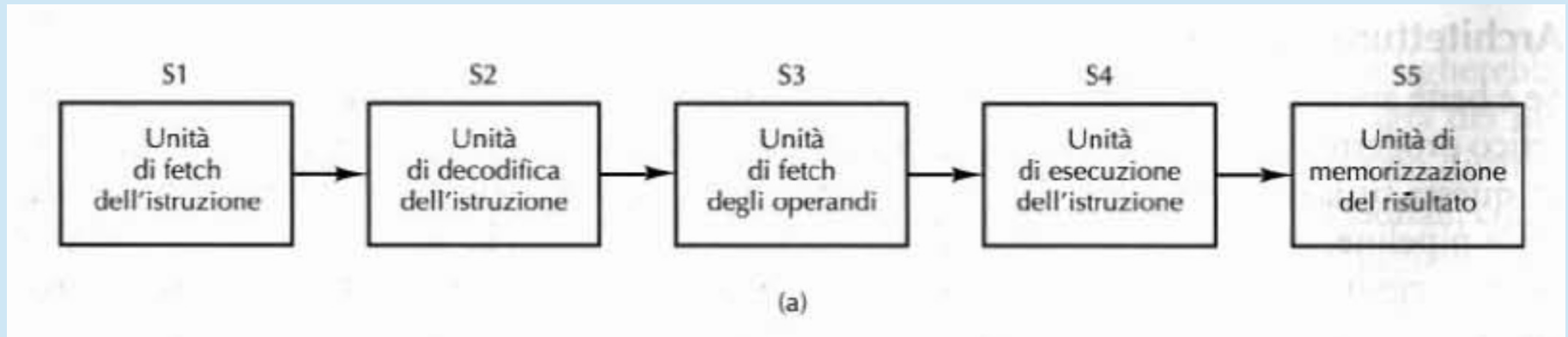
Durante la fase storica in cui i sistemi CISC si stavano diffondendo, altri sperimentavano sistemi con meno istruzioni, più semplici, il cui hardware però era molto più veloce: I sistemi **RISC**
Reduced Instruction Set Computer.

CISC e RISC

LA SCELTA DI INTEL: Sistema Ibrido:

- CISC lente, interpretate: solo per istruzioni rare
- RISC veloci, non interpretate: per istruzioni frequenti

Pipeline



Origini: Prefetch Buffer

Inizialmente era necessario velocizzare l'operazione FETCH, la quale legge dalla memoria l'istruzione corrente.

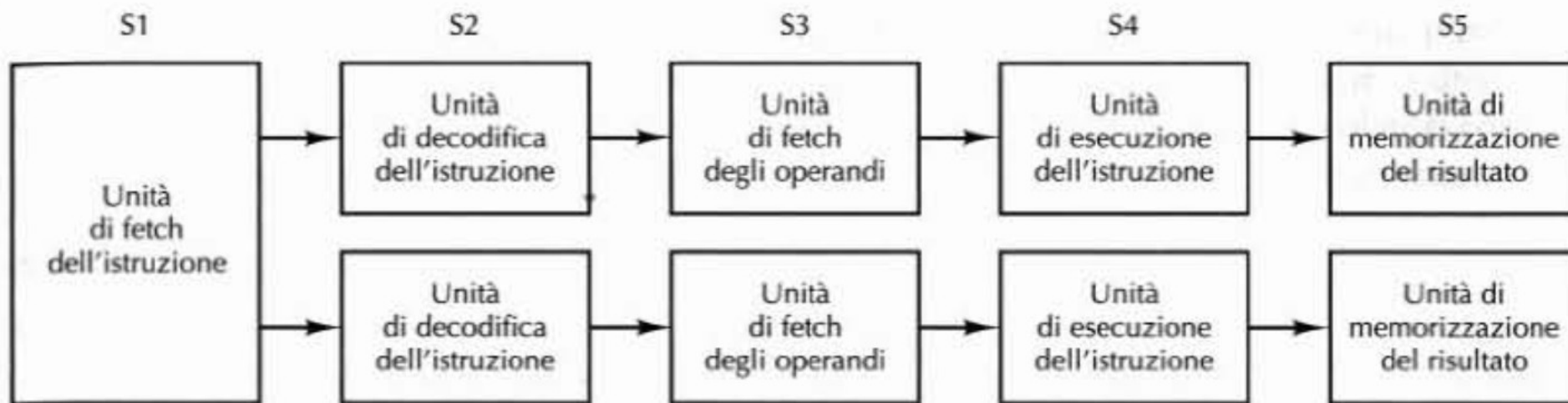
Venne inventato il PREFETCH BUFFER.

Funzionamento:

Mentre si sta eseguendo l'istruzione precedente, si legge la nuova istruzione in anticipo (fase FETCH anticipata)

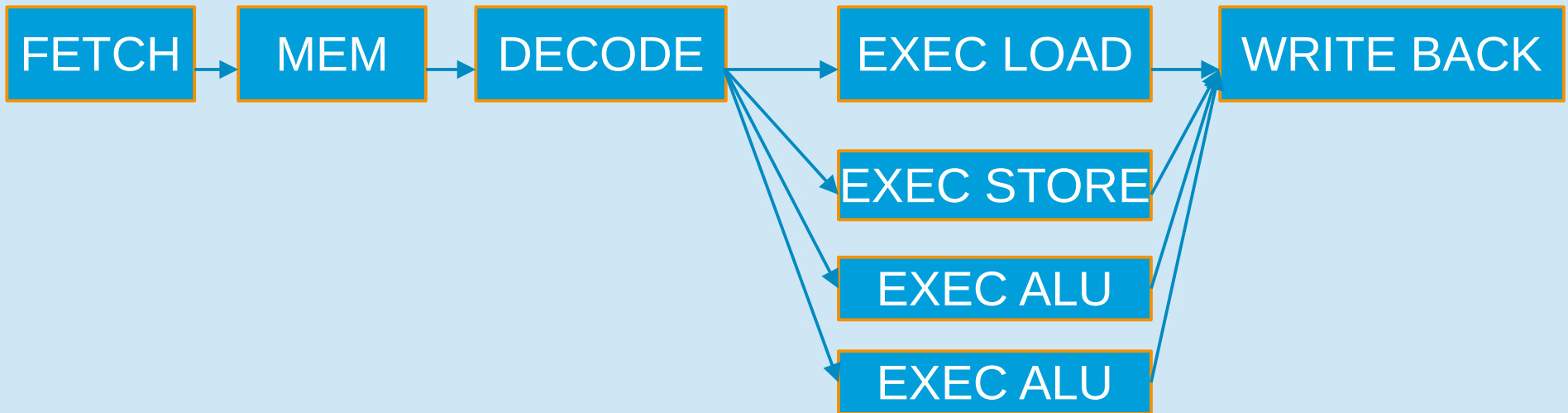


Superscalare (Stadi paralleli)



Superscalare (Stadi paralleli)

A volte viene duplicato solo lo stadio EXE essendo il più lento



Aumento di prestazioni CPU

Concetti importanti per aumentare prestazioni (MIPS) :

- Istruzioni direttamente eseguite da hw sono piu veloci
- Ottimizzare la velocita di INIZIO delle istruzioni (PIPELINE)
- Istruzioni facili da decodificare
- Solo load e store devono accedere alla memoria (sono lente)
- Molti registri evitano continui accessi lenti alla cache o alla RAM

Array Computer / Processore Vettoriale

Un modo per aumentare le prestazioni di calcolo, e' quello di usare nello stesso computer un **Array di Processori**, ai quali assegnare una operazione, che viene eseguita da ogni processori su dati diversi. Ogni processore processa il suo segmento dati.

Un altro modo di parallelizzare l'esecuzione di istruzioni, e' costruire un **Processore Vettoriale**, ovvero un processore che ha una ALU speciale che agisce su vettori di dati anziche singoli elementi.

Multiprocessori

Un altro sistema per effettuare più operazioni contemporaneamente è il **Multiprocessore**.

Esso è più versatile dei due sistemi descritti prima, in quanto permette a ogni CPU di eseguire un'istruzione diversa da quella eseguita dall'altra CPU.

Le CPU coinvolte condividono una stessa memoria.

Questa versatilità causa anche possibili conflitti, e il codice deve essere scritto appositamente, o compilato in modo ottimizzato ove possibile, affinché l'esecuzione sia corretta.

Multiprocessori

Usando più CPU in **parallelo**, si aumenta la quantità di istruzioni eseguite al secondo. Questo aumenta il **traffico sul BUS dati**, per cui spesso si aggiunge una **memoria locale** a ogni CPU. In questo modo, solo i dati condivisi sono memorizzati nella memoria condivisa, tutto il resto é locale e non genera traffico.

