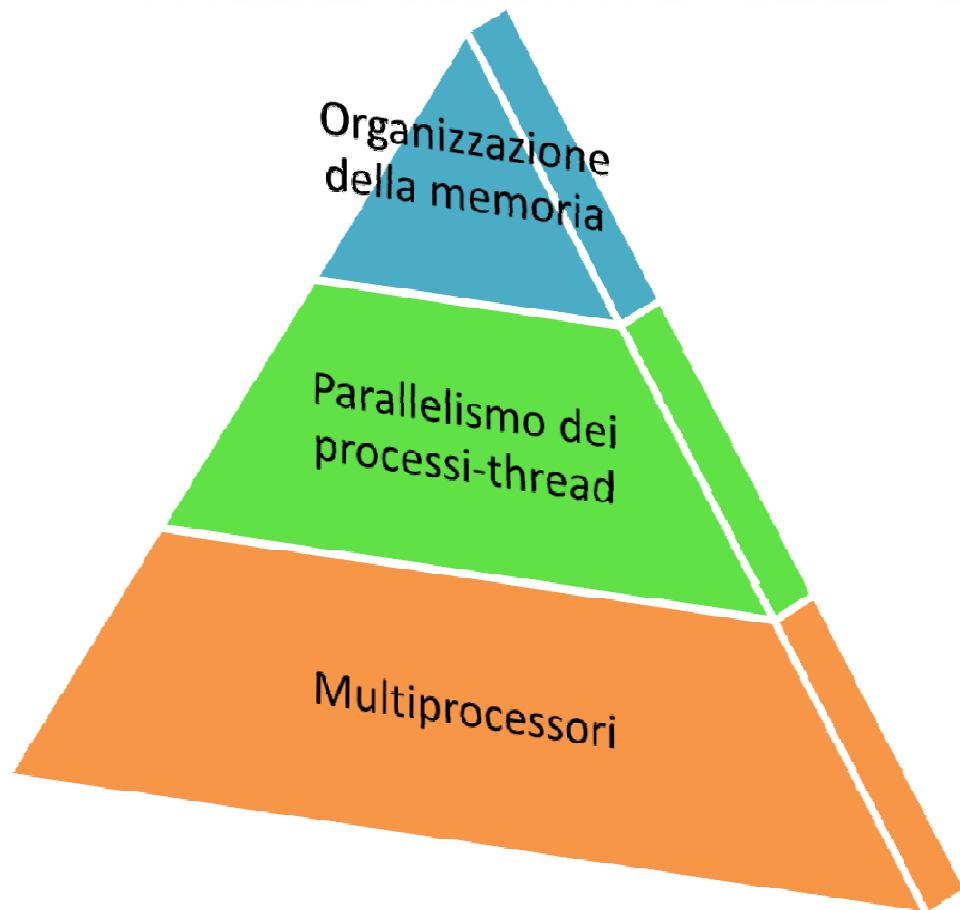




Architettura degli Elaboratori Multiprocessori

ARGOMENTI DELLA LEZIONE

- Multiprocessori
- Parallelismo dei processi-thread
- Organizzazione della memoria





Architettura degli Elaboratori

Classificazione



MULTIPROCESSORI

Generalità

- ❑ Nonostante i progressi fatti nel campo tecnologico si è arrivati a dover affrontare dei **limiti** tecnici e economici come:
 - ❑ L'incremento della velocità di trasmissione
 - ❑ l'impossibilità di miniaturizzare i componenti al di sotto di una certa soglia (per motivi fisici)
 - ❑ La scarsa convenienza commerciale per le case costruttrici nel produrre processori sempre più veloci ed esponenzialmente più costosi, in quanto non sarebbero più prodotti competitivi e non rientrerebbero nella fascia di mercato dominata dal consumatore medio
- ❑ Per questo motivo si è aumentato il numero di processori



MULTIPROCESSORI

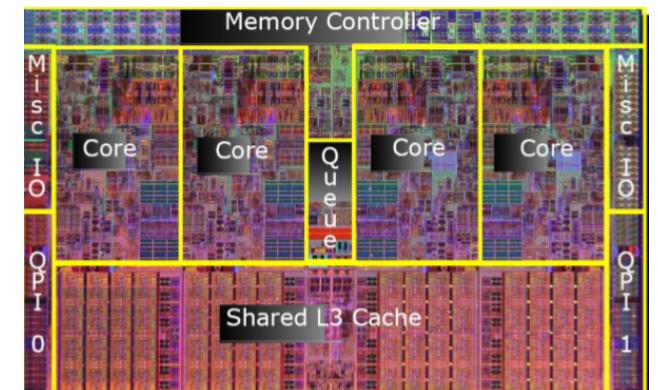
Definizione

- I sistemi **multiprocessore** e **multicore**, sono definiti come:
 - **Multiprocessore:** “*Sistema equipaggiato di almeno due processori completi operanti in parallelo*”
 - **Multicore:** “*Sistema composto da almeno due core, ovvero da due, o più, parti essenziali della CPU (cioè quelle in grado di svolgere una istruzione) residenti in un chip*”
- **PRO**
 - clock a frequenze maggiori
 - uso efficiente delle memorie cache
 - tempi di risposta migliori con carichi di lavoro intensi
 - riduzione del circuito stampato
 - possibilità di lavorare a correnti più basse, quindi una dispersione di calore minore e minor consumo elettrico
- **CONTRO**
 - modifiche al sistema operativo e ai programmi pre-esistenti
 - in generale sono più difficili da gestire rispetto ai sistemi monoprocessori
 - Scarsa ridondanza

Intel Xeon(4 processori)



Intel i7 (4 core)



MULTIPROCESSORI - MULTICORE

Quanti core ci sono nelle recenti architetture?





MULTIPROCESSORI

Vecchia classificazione

- Questi concetti erano stati, in parte, utilizzati da Michael J. Flynn nel 1966 per definire la tassonomia più completa e popolare (ma datata) sulle architetture parallele
- La classificazione si basa sulla nozione di **flusso di informazioni**
 - In un processore sono presenti due tipologie di flusso di informazioni: **istruzioni** e **dati**
 - Concettualmente questi possono essere pensati come due flussi indipendenti, a seconda che essi scorrono su due cablature differenti o meno

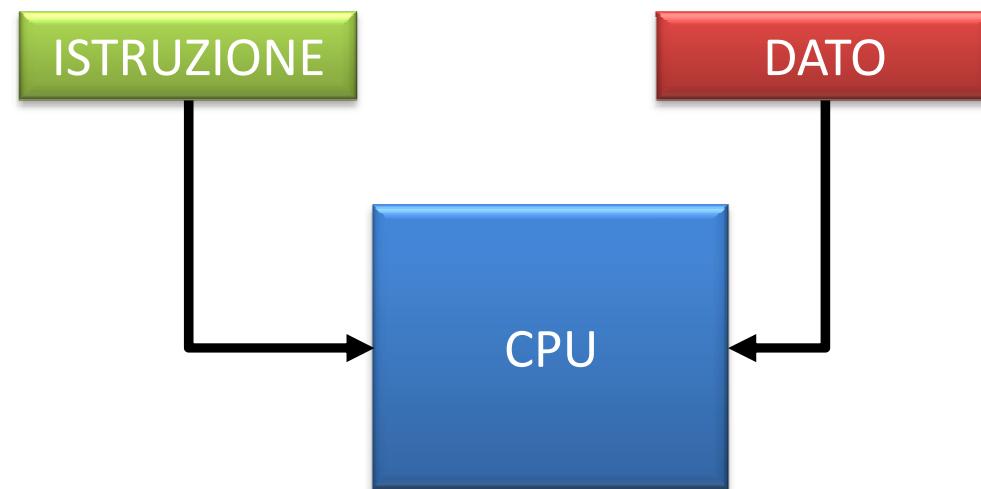
Tassonomia di Flynn



MULTIPROCESSORI

Architettura Single Instruction Single Data

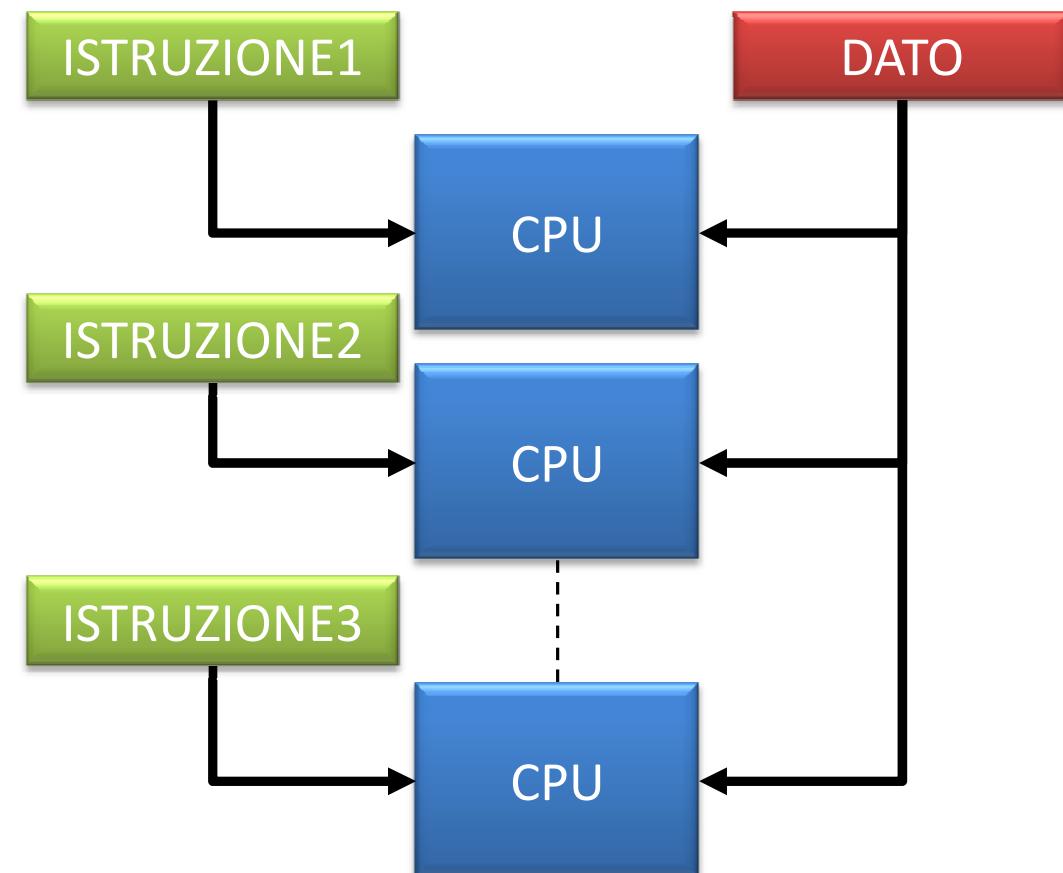
- Una **architettura SISD** è costituita da una singola unità di elaborazione, la quale riceve un unico flusso di istruzioni che opera su un unico flusso di dati
- Questo modello corrisponde a quello proposto nel progetto RISC (in parte richiamando la logica di Von Neumann)
- Gli **algoritmi** che sono eseguiti su questo tipo di architettura sono detti **sequenziali**, in quanto non contengono alcun parallelismo
- Esempi di architetture SISD sono gli elaboratori, le workstation e i mainframe dotati di una singola CPU



MULTIPROCESSORI

Architettura Multiple Instruction Single Data

- Nell'architettura **MISD** n processori, ciascuno dotato di una propria CPU, condividono un'unica unità di memoria
- Ad ogni ciclo di clock, il dato ricevuto dalla memoria è elaborato da tutte le CPU simultaneamente secondo le istruzioni ricevute dalla propria unità di controllo
- Con questa architettura si ha il **parallelismo a livello di istruzioni**
- Questa classe di elaboratori eseguono operazioni diverse su input identici e per questo sono soliti essere usati nella classificazione dei problemi computazionali o nella crittografia dei dati



MULTIPROCESSORI

Architettura Multiple Instruction Single Data: esempio

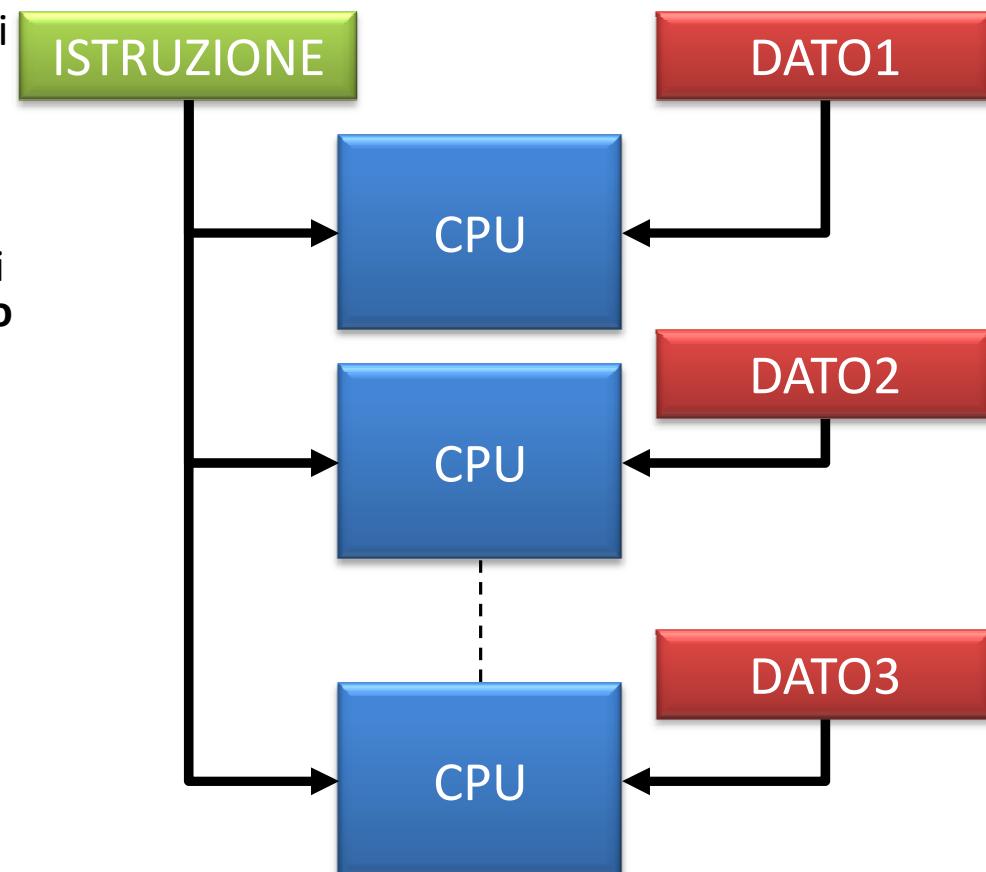
Dato: "Franco Liberati"

ALGORITMO	RISULTATO	ALGORITMO	RISULTATO
MD4	9e04da9f42791b76a8d52ffab53f9040	CRC32	ce126edc
MD6-128	ed8f4bb7ae251c1e8a2b1ef7952324ec	Whirlpool	6a8a65a3e807b2a4aaaf7eb5c2b7c4e1d787e71 08ef0eb0a4315f94629413c5432d2cea1d39d1a09 88a3d67a06f137336e9a20ccafee8eafb41ba1cc63 cda9e
MD6-512	f38aa64f3366d8e209d2568916e29d588242761c5 232c3f7c1533d72fdbf802c31abfcc5776ab8f40a21 560a1030da26d902a8aa7be0ae33f0d5f62aef6170 6c	Adler32	2bf105a6
RipeMD-160	077fe5765e6afcc53db9da2d2f2a4d660cd97007	SHA3-256	5ad12383ed19ec6e1414c2355cddf13e108ee034 d476c62d1834b38d496f82d8
RipeMD-320	f953beb75ea2a941fdd771a1b809c86fe278457e95 a9ad24213b8105b2d6e1f6e84200dbb4f9cb01	NTLM	1509AC5DA79E4B3235CD6A9645C0E7CE

MULTIPROCESSORI

Architettura Single Instruction Multiple Data

- ❑ Una architettura SIMD è composta da n processori identici, ognuno dotato di una propria memoria locale. Tutti i processori lavorano sotto il controllo di un flusso di istruzioni unico e rilasciato da un'unità di controllo centrale. Inoltre sono presenti n flussi di dati, uno per ogni processore
- ❑ I processori funzionano parallelamente: ad ogni passo, tutti i processori eseguono la stessa istruzione ma su dati diversi. Questa architettura è un esempio di **parallelismo a livello dati** (DLP)
 - ❑ PRO: Facilità di progettazione, analisi e implementazione
 - ❑ CONTRO: Si possono risolvere algoritmi del tipo *divide et impera*: cioè problemi che possono essere suddivisi in una serie di sottoproblemi tutti identici, ognuno dei quali è poi eseguito contemporaneamente tramite lo stesso insieme di istruzioni
- ❑ Le architetture SIMD (tipica dei processori vettoriali) sono usati nel campo dell'elaborazione grafica e delle applicazioni multimediali





MULTIPROCESSORI

SIMD:esempio di programma

- Il problema SOMMATORIA può essere così formulato:
Istanza: interi $a[1], a[2], \dots, a[n]$

Risposta: $S = \sum_{k=1}^n a[k]$

for (i=1; i≤n;i++) {tot=tot+a[i];}
Ordine di grandezza = **O(n-1)**

- ALGORITMO in parallelo

for (j=1; j≤log₂n;j++)
{
for all p processor (1 ≤ p ≤ n/2^j) do in parallel
 $\{a[2^j p] = a[2^j p] + a[2^j p - 2^{j-1}]\}$
}
Ordine di grandezza = **O(log₂n)**

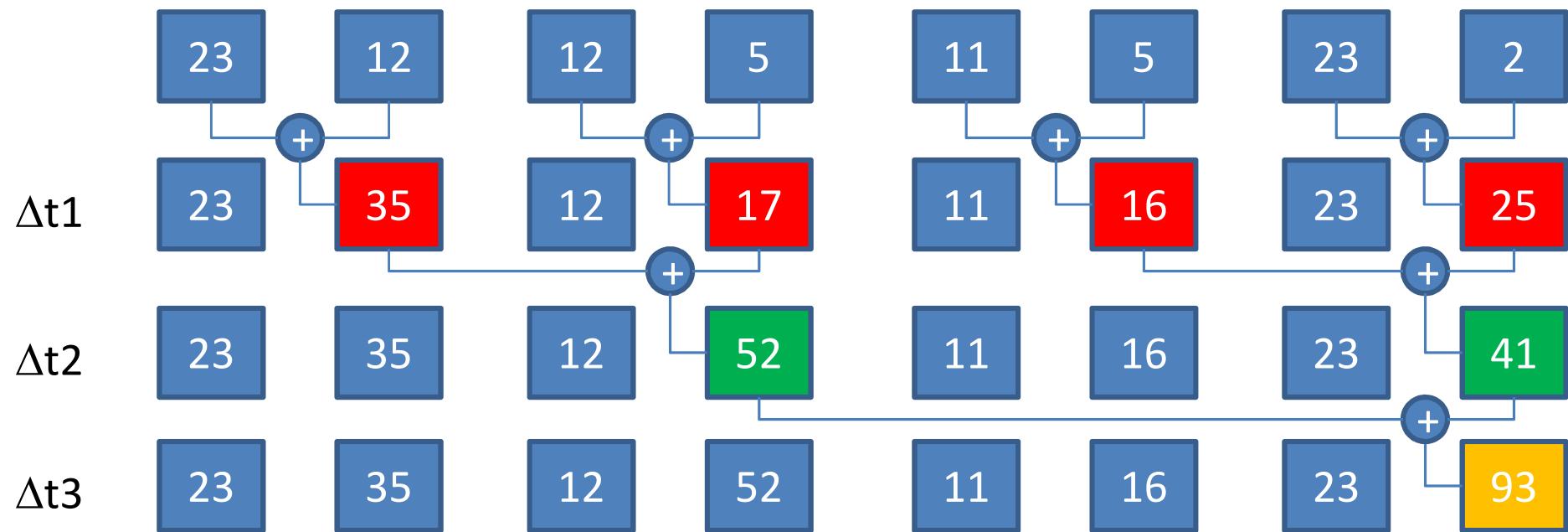
MULTIPROCESSORI

SIMD: esempio di programma

```

for (j=1; j≤log2n; i++)
{
    for all p processor ( $1 \leq p \leq n/2^j$ ) do in parallel
        { $a[2^j p] = a[2^j p] + a[2^j p - 2^{j-1}]$ }
}

```

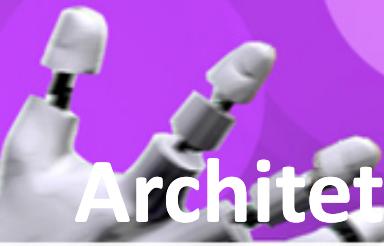




MULTIPROCESSORI

Architettura Single Instruction Multiple Data: P-RAM

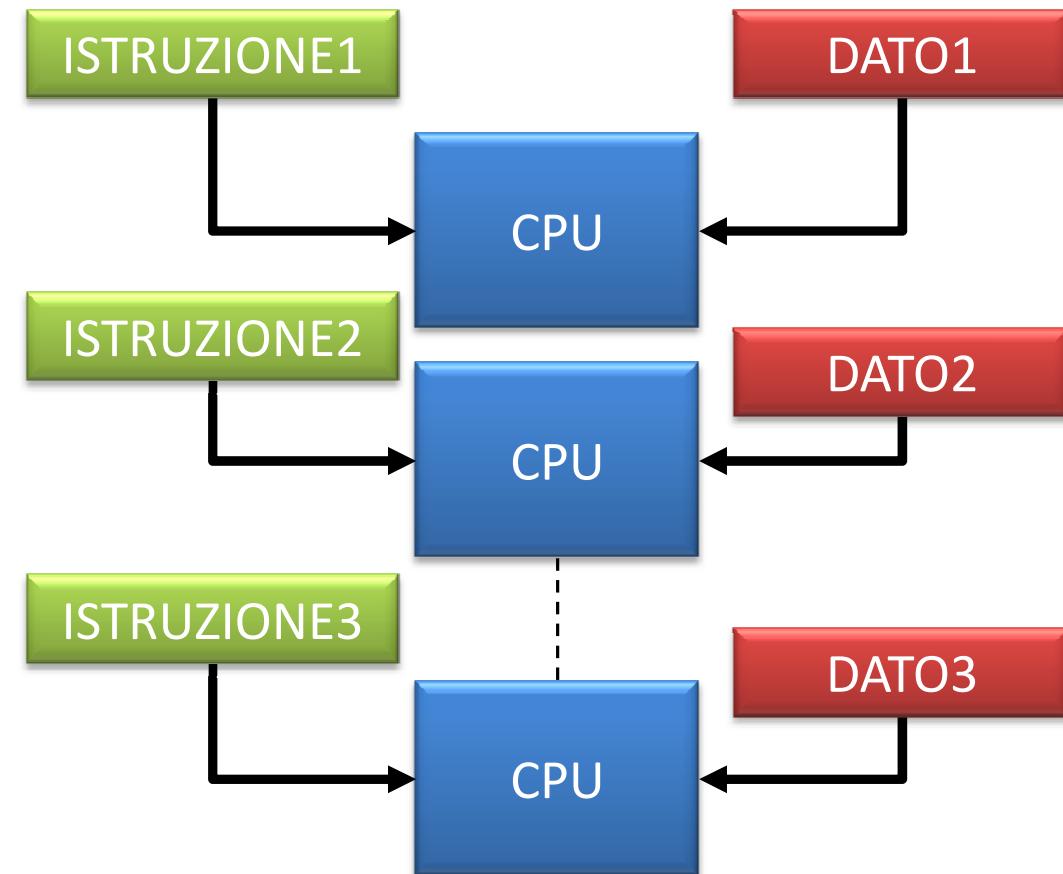
- Il modello SIMD più noto è il **P-RAM** che consiste di p -processori con una memoria globale, condivisa da tutti
- La memoria globale è usata dai processori per scambiarsi dati in tempo costante $O(1)$: perché il processore k e il processore j si scambino un valore, basta che il processore k scriva tale valore in una **variabile condivisa** e il processore j vi acceda in lettura
- Il calcolo procede per passi. Ad ogni passo ogni processore può fare una operazione sui dati che possiede, oppure può leggere o scrivere nella memoria condivisa. In particolare, è possibile selezionare un insieme di processori che eseguono tutti la stessa istruzione (su dati generalmente diversi), mentre gli altri processori restano inattivi; i processori attivi sono sincronizzati, nel senso che eseguono la stessa istruzione simultaneamente e l'istruzione successiva può essere eseguita solo quando tutti hanno terminato l'esecuzione
- Si possono ulteriormente specificare vari modelli di PRAM, in termini di limitazione agli accessi a memoria condivisa:
 1. **EREW** (Exclusive Read Exclusive Write): non è ammesso l'accesso contemporaneo da parte di più processori alla stessa locazione di memoria.
 2. **CREW** (Concurrent Read Exclusive Write): l'accesso contemporaneo è permesso in lettura.
 3. **CRCW** (Concurrent Read Concurrent Write): l'accesso contemporaneo è permesso in lettura ed in scrittura



MULTIPROCESSORI

Architettura Multiple Instruction Multiple Data

- L'architettura **MIMD** è la più generale e più potente nella classificazione di Flynn. Ci sono n processori, n flussi di istruzioni e n flussi di dati. Ogni processore possiede una propria unità di controllo e una propria memoria locale
- Ogni processore opera sotto il controllo di un flusso di istruzioni rilasciato dalla propria CPU: pertanto, i processori possono potenzialmente eseguire programmi diversi su dati diversi, risolvendo sottoproblemi differenti ma facenti parte di un unico problema più grande
- Si ha un **parallelismo a livello di thread/processi (TPL)**
 - **PRO:** tale architettura risolve in parallelo quei problemi che non hanno una struttura regolare
 - **CONTRO:** gli algoritmi sono difficili da progettare, analizzare e implementare

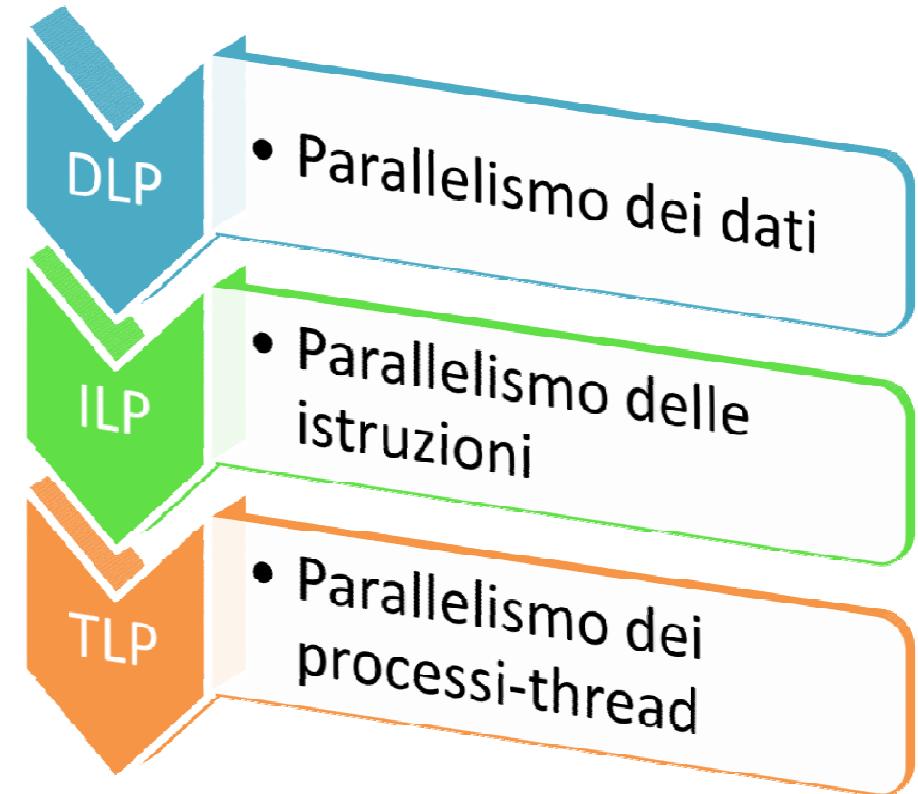




MULTIPROCESSORI

Nuova Classificazione

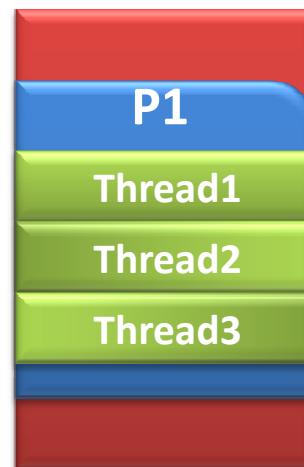
- Una architettura **parallela** (definita da George S. Almasi e Allan Gottlieb nel 1989 come: "*Insieme di elementi di elaborazione che cooperano e comunicano per risolvere velocemente problemi di dimensioni considerevoli, talvolta intrattabili su macchine sequenziali*") è caratterizzata da **tre livelli di parallelismo**:
 - **Data Level Parallelism (DLP)**: i dati, su cui il programma lavora, sono distribuiti ed elaborati contemporaneamente da più processori o core
 - **Instruction Level Parallelism (ILP)**: le istruzioni, che compongono il programma, sono distribuite ed eseguite contemporaneamente da più processori o core
 - **Thread Level Parallelism (TLP)**: le applicazioni utilizzano thread/processi concorrenti, cioè thread/processi che sono eseguiti in parallelo da più processori o core



MULTIPROCESSORI

Parallelismo dei processi e dei thread

- Un **processo (task)**, è un programma in memoria
- Un **thread** è una suddivisione di un processo in due o più sottoprocessi
- In una architettura monoprocessoressore due o più thread sono eseguiti in pseudo-parallelismo; mentre in un sistema multiprocessore o multicore i thread sono elaborati parallelamente



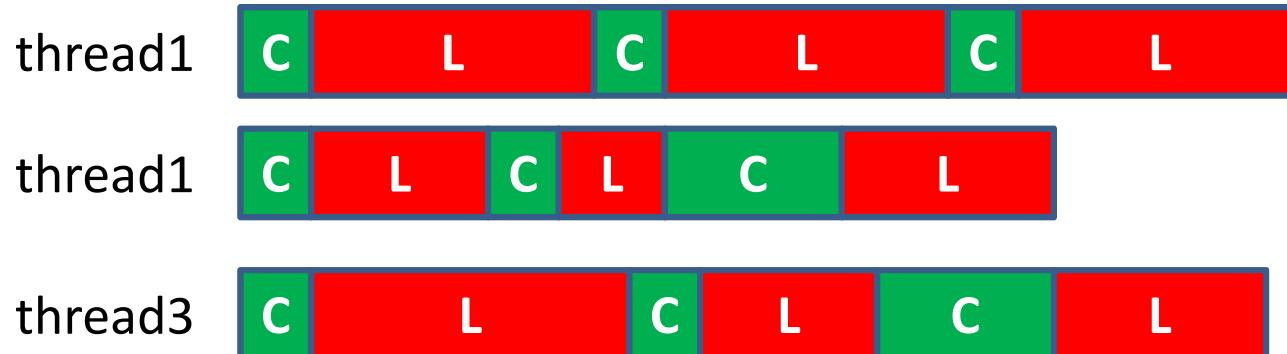
		0%	1,2 MB	0 MB/s	0 Mbps
>	Blocco note	0,8%	23,0 MB	0 MB/s	0 Mbps
>	Gestione attività	0%	7,4 MB	0 MB/s	0 Mbps
>	Internet Explorer (2)	37,6%	54,6 MB	0,1 MB/s	1,2 Mbps
>	Microsoft Edge (11)	0,5%	466,3 MB	0 MB/s	0 Mbps
e	algoritmo parallelo esempio - ...	0%	50,8 MB	0 MB/s	0 Mbps
e	aloritmiparalleli.pdf	0%	4,3 MB	0 MB/s	0 Mbps
e	Background Tab Pool	0%	3,9 MB	0 MB/s	0 Mbps
Browser Broker		0%	3,5 MB	0 MB/s	0 Mbps
e	Chakra JIT Compiler	0%	273,3 MB	0 MB/s	0 Mbps
e	fiberat@yahoo.it - Yahoo Mail	0%	26,9 MB	0 MB/s	0 Mbps
e	Microsoft Edge	0%	54,6 MB	0 MB/s	0 Mbps
e	Microsoft Edge Manager	0%	6,9 MB	0 MB/s	0 Mbps
Runtime Broker		0%	2,0 MB	0 MB/s	0 Mbps
e	User Interface Service	0%	32,8 MB	0 MB/s	0 Mbps
>	Microsoft Office PowerPoint (32...)	0,3%	3,1 MB	0,1 MB/s	0 Mbps



MULTIPROCESSORI

Processi e thread concorrenziali

- Per **multithread** si intende la “capacità di un processore di passare da un thread all’altro. Questa capacità è usata quando uno dei thread è in uno stato di stallo, ad esempio perché i dati necessari non sono ancora disponibili o si sta facendo una accesso ad un dispositivo di I/O Passare ad un altro thread, le cui istruzioni possono essere eseguite, comporterà un’elaborazione dei dati più efficiente”
- In altre parole il multithreading permette a diversi thread di condividere le risorse hardware di un unico processore in modo tale che esso possa eseguirli in parallelo. Affinché questa tecnica sia efficace, è necessario che il sistema operativo e le applicazioni la supportino. Quindi il software non dovrà più essere di tipo sequenziale ma bensì programmato secondo una logica parallela. Inoltre, anche l’hardware dovrà essere progettato in modo da poter passare velocemente da un thread all’altro



C: tempo di computazione
L: tempo di latenza (conflitto
acceso in memoria,
fallimento accesso in cache,
interruzione, attesa I/O)



MULTIPROCESSORI

Parallelismo dei processi e dei thread

- Il multithreading soddisfa il parallelismo a livello di thread (TLP) ed è stato introdotto quando ci si è accorti che per aumentare le prestazioni dei calcolatori c'era bisogno di qualcosa in più del parallelismo a livello di istruzioni (ILP)
 - Con il solo utilizzo dell'ILP si arrivò a situazioni conflittuali che comportavano di conseguenza inutili sprechi di risorse.
 - Paradigmi di processi sono stati inventati per migliorare l'ILP ed aumentare il volume di istruzioni eseguibili simultaneamente, come l'*Out of Order Processing*, ma si è visto che le prestazioni ottenute non potevano giustificare le costose migliorie da fare a livello hardware. Perciò si è optato per il multithreading come soluzione finale.



MULTIPROCESSORI

Parallelismo dei processi e dei thread

□ Paradigma *Out of Order Processing*

In order	Out of order
Fetch	Recupero delle istruzioni
Decode	Invio delle istruzioni verso una coda di istruzioni (chiamata anche buffer di istruzioni o <i>reservation station</i>)
Execute	L'istruzione attende nella coda finché i suoi operandi di input non sono disponibili. L'istruzione lascia la coda
Mem	L'istruzione è inviata all'unità funzionale appropriata ed eseguita da tale unità
WB	I risultati sono in coda
	Solo dopo che tutte le istruzioni precedenti hanno riportato i risultati nei registri avviene il salvataggio in memoria

MULTIPROCESSORI

Parallelismo dei processi e dei thread

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

In Order Processing

I1=b·b
 I2=a·c
 I3=4·I2
 I4=I1-I3
 I5=sqrt(I4)
 I6=b -1
 I7=2 ·a
 I8=I6+I5
I9=I8/I7
 I10=I6-I5
I11=I10/I7

CORE 1	CORE 2
I1	I2
I3	NOP
I4	NOP
I5	I6
I7	I8
I9	I10
I11	NOP

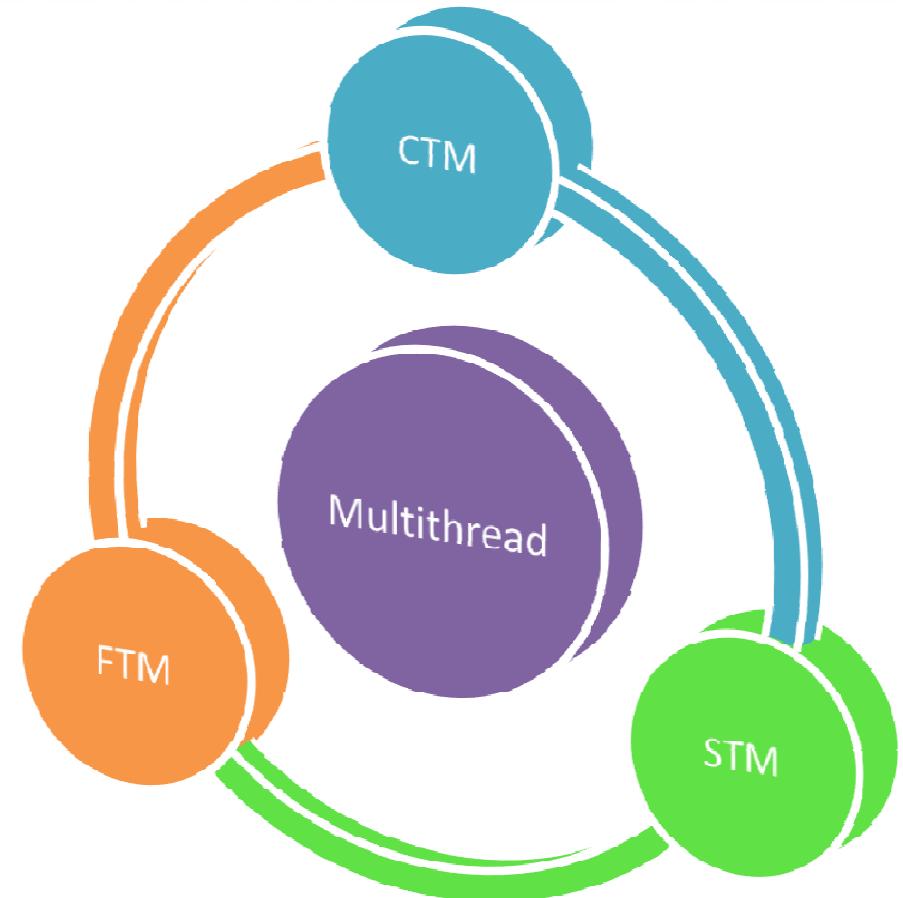
Out of Order Processing

CORE 1	CORE 2
I1	I2
I3	I4
I6	I5
I7	I8
I10	I9
I11	-

MULTIPROCESSORI

Parallelismo dei processi e dei thread

- I tre tipi principali di hardware multithreading sono:
 - il Coarse-Grained Multithreading
CMT
 - il Fine-Grained Multithreading
FMT
 - il Simultaneous Multithreading
SMT



MULTIPROCESSORI

Parallelismo dei processi e dei thread (caso studio)

- Dimensione verticale: istruzioni per ciclo di clock
 - Dimensione orizzontale: la sequenza dei cicli di clock
 - Ogni quadrante colorato indica l'esecuzione di un'istruzione, mentre ogni quadrante rosso rappresenta uno stato di inattività del processore
- OSS: Enorme spreco di risorse: a) i cicli di clock in cui tutti gli slot a disposizione sono utilizzati sono molto rari b) gli stalli possono paralizzare interi thread.





MULTIPROCESSORI

Multithread: Coars Grained MultiThreading (CMT)

- Il **multithreading a grana grossa** rende più efficiente l'utilizzo delle unità funzionali mediante l'esecuzione di un solo thread alla volta, per un certo numero di cicli di clock
- Questa tecnica è proficua laddove interi cicli di clock sono inattivi in quanto il thread che si stava eseguendo è in fase di stallo. L'efficienza è dovuta appunto all'eliminazione di questi cicli di clock inattivi
- il processore, se rileva che un thread è in stato di stallo, passa immediatamente all'esecuzione di un altro thread. Quindi non resta più inattivo aspettando che si compiano tutti i cicli di clock assegnati ad un thread, bensì continua a lavorare per un altro. Prima di passare al thread successivo, il processore salva lo stato di quello attuale facendo una copia delle istruzioni presenti nella pipeline e le unità funzionali in uso. Il tutto utilizzando un certo numero di set di registri
- Questa tecnica risulta inutile nel caso in cui gli stalli siano molto piccoli. in tal caso il tempo necessario per salvare lo stato e riempirla con le istruzioni del thread successivo sarebbe di gran lunga maggiore rispetto a quello che si dovrebbe aspettare per far passare i cicli di clock inattivi

MULTIPROCESSORI

Parallelismo dei processi e dei thread CMT



Δ1 Δ 2 Δ 3 Δ 4 Δ 5 Δ 6 Δ 7 Δ 8 Δ 9 Δ10 Δ11 Δ12 Δ13 Δ14

CORE 1



CORE 2



MULTIPROCESSORI

Multithread: Fine Grained MultiThreading (FMT)

- Il multithreading a grana fine consente l'alternanza di thread ad ogni ciclo di clock
- Questa alternanza è spesso creata in stile **roundrobin**, ovvero con un alternanza gestita come una coda circolare senza priorità, saltando qualsiasi thread si trovi in uno stato di stallo nel momento in cui dovrebbe essere eseguito
- L'hardware deve essere implementato in modo da poter passare da un thread all'altro ad ogni ciclo di clock. Il vantaggio principale del FMT è che, sia per stalli brevi che per stalli lunghi, è mantenuto un certo rendimento, in quanto in ogni caso il processore deve procedere eseguendo delle istruzioni
- Lo svantaggio più grande di questa tecnica è che essa rallenta l'esecuzione dei singoli thread, in quanto un thread che non presenta stalli verrà purtroppo ritardato dall'esecuzione delle istruzioni degli altri thread

MULTIPROCESSORI

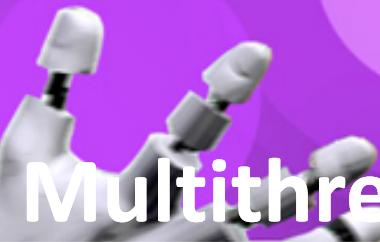
Parallelismo dei processi e dei thread FMT



$\Delta 1 \quad \Delta 2 \quad \Delta 3 \quad \Delta 4 \quad \Delta 5 \quad \Delta 6 \quad \Delta 7 \quad \Delta 8 \quad \Delta 9 \quad \Delta 10 \quad \Delta 11 \quad \Delta 12 \quad \Delta 13 \quad \Delta 14$

CORE 1





MULTIPROCESSORI

Multithread: Simultaneous MultiThreading (SMT)

- Il multithread simultaneo esegue istruzioni provenienti da diversi thread, in qualsiasi momento, in una qualsiasi unità funzionale. Nell'alternare i thread, un chip funziona come un processore FMT ma allo stesso tempo agisce come un processore CMT, eseguendo simultaneamente istruzioni proprie di thread diversi.
 - In questo multithreading il processore è in grado di gestire un parallelismo sia a livello di thread sia a livello di istruzioni. Va detto che, di solito, le architetture che utilizzano l'SMT dispongono di un numero di unità funzionali maggiore di quello di cui un singolo thread potrebbe necessitare in realtà.
- Come ci si aspetta, più thread sono in uso più alto sarà il rendimento complessivo; per questo motivo, l'SMT permette agli architetti di hardware di progettare core sempre più numerosi senza preoccuparsi che questi ne risentano in termini di rendimento
 - Però, per ottenere questo vantaggio i costruttori devono porre particolare attenzione al momento in cui dimensionano le cache nei vari livelli. Infatti incrementando o decrementando le dimensioni delle cache si possono ottenere diversi vantaggi o svantaggi

MULTIPROCESSORI

Parallelismo dei processi e dei thread STM



Δ1 Δ 2 Δ 3 Δ 4 Δ 5 Δ 6 Δ 7 Δ 8 Δ 9 Δ10 Δ11 Δ12 Δ13 Δ14

CORE 1



CORE 2

The background features a robotic hand on the left side, interacting with a circular interface composed of glowing purple and blue dots. Above this interface, the text is displayed against a dark purple gradient background.

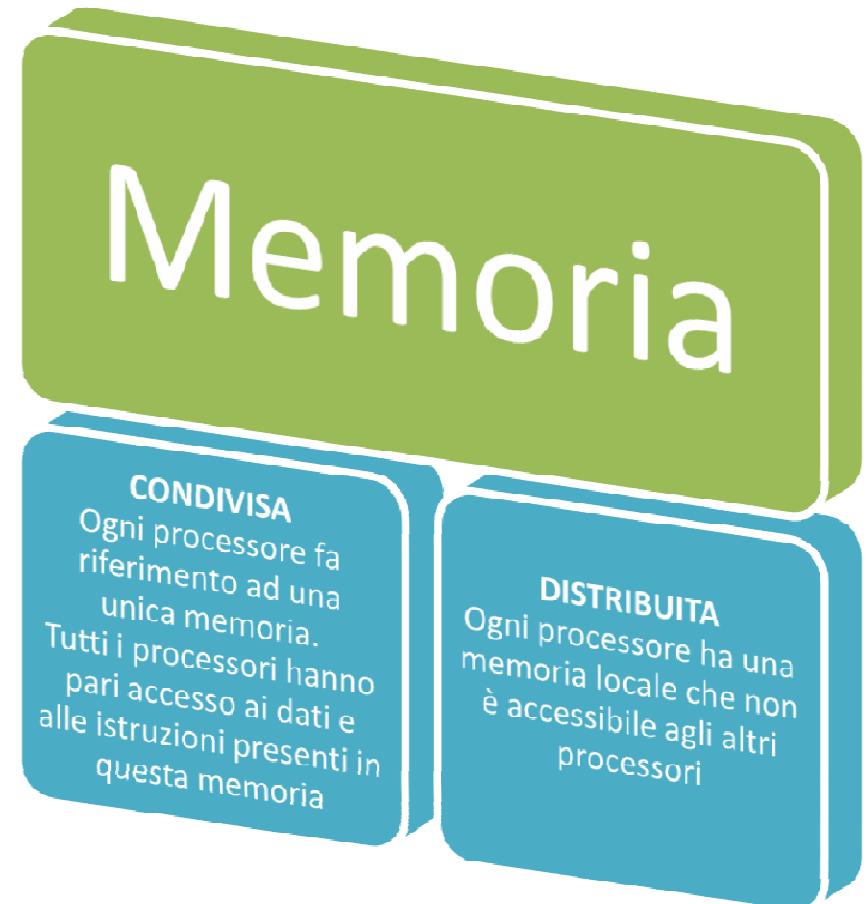
Memoria nei sistemi multiprocessori/multicore



MULTIPROCESSORI

Organizzazione Memoria

- ❑ Nei multiprocessori si riferisce alla **memoria condivisa o distribuita**





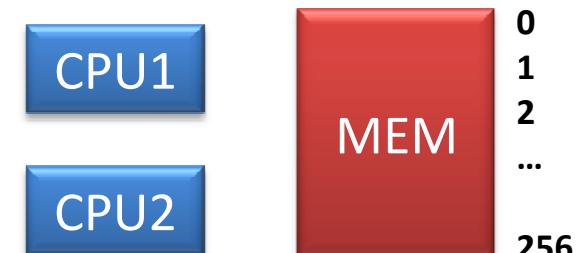
MULTIPROCESSORI

Organizzazione Memoria

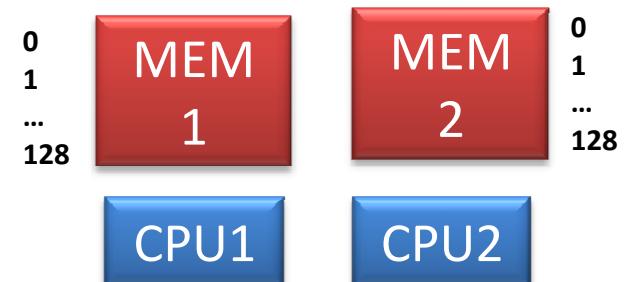
- Ciò che distingue una memoria condivisa da una memoria distribuita è come il **sottosistema di memoria interpreta un indirizzo generato da un processore**
- La distinzione tra memoria condivisa e memoria distribuita determina il modo in cui diverse parti di un programma parallelo devono comunicare
- In un sistema a memoria condivisa è sufficiente costruire una struttura dati in memoria e passare alle subroutine parallele le **variabili di riferimento**, ovvero gli indirizzi, di tale struttura dati. Una macchina a memoria distribuita, invece, necessita di scambiarsi le informazioni attraverso dei messaggi
 - Un inconveniente della memoria distribuita è che a volte questi messaggi possono essere molto grandi e richiedere tempi di trasferimento relativamente lunghi

Iw \$a0,100 #carica in \$a0
#l'operando contenuto
#nella locazione 100 di
#memoria principale

Memoria condivisa: \$a0 ha lo stesso valore



Memoria distribuita: \$a0 ha valori diversi





MULTIPROCESSORI

Memoria Condivisa: caratteristiche

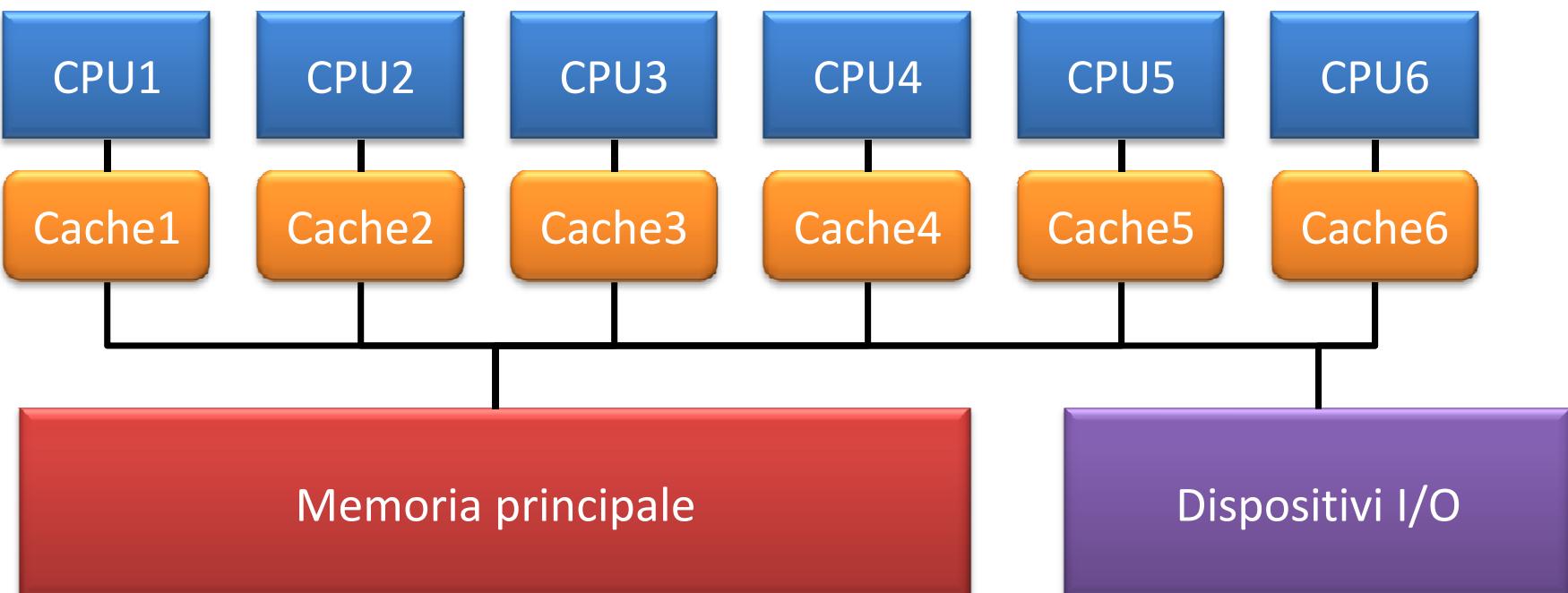
- Caratteristiche dei **sistemi multiprocessore a memoria condivisa**:
 - La **memoria logica** è la stessa per tutti i processori; ad esempio, tutti i processori associati alla stessa struttura dati lavoreranno con gli stessi indirizzi logici, in quanto globali, accedendo così alle stesse locazioni di memoria
 - La **sincronizzazione** è ottenuta analizzando i task dei vari processori e concedendo a turno la memoria condivisa; infatti i processori possono solo accedervi uno alla volta. Una locazione di memoria condivisa non deve essere modificata da un task mentre un altro task concorrente vi accede
 - Il programmatore è responsabile della gestione della sincronizzazione, inserendo opportuni controlli, semafori, lock, ecc nel programma che gestisce le risorse
 - La **condivisione dei dati** tra i vari task è veloce; infatti il tempo necessario ai task per comunicare tra loro è il tempo che uno di questi impiega per leggere una singola locazione (ciò dipende dalla velocità di accesso alla memoria)
 - La **scalabilità** è limitata dal numero di vie d'accesso alla memoria; questo limite si presenta soprattutto quando ci sono più task che connessioni alla memoria. In queste situazioni si avranno dei processori in stato d'attesa e quindi tempi di latenza maggiori.



MULTIPROCESSORI

Memoria Condivisa

- Un modo semplice per collegare più processori insieme per costruire un multiprocessore a memoria condivisa è utilizzando un bus (**multiprocessore a bus unico**)
- Ad ogni processore è associata una memoria cache (memoria locale) in quanto si presume che la probabilità che un processore necessiti di un dato o di un'istruzione presente nella memoria locale sia molto alta ($p>0.9$)

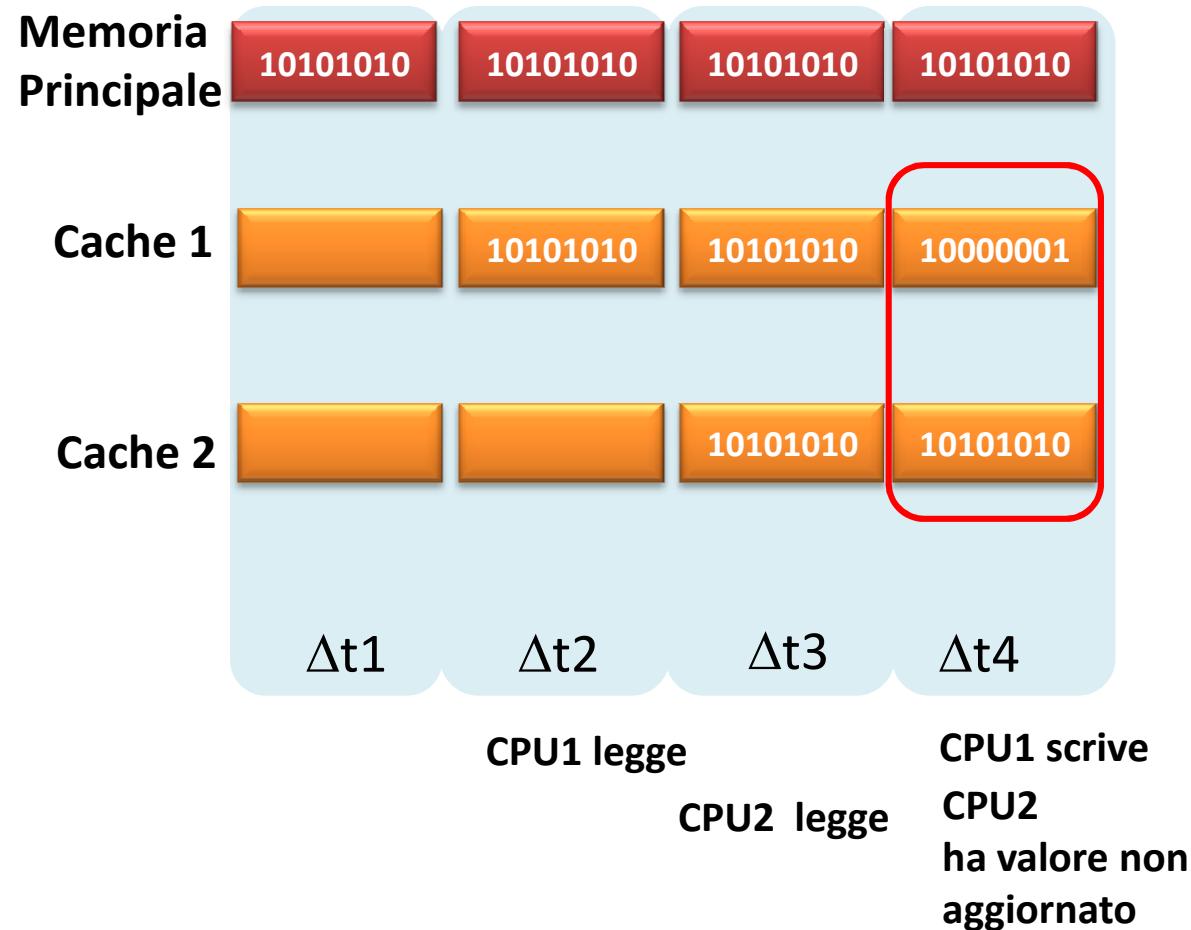




MULTIPROCESSORI

Memoria Condivisa

- Si incorre nel **problema di coerenza della cache** che sorge quando un processore modifica un dato della memoria principale mentre è simultaneamente utilizzato da altri processori
- Il nuovo valore passa dalla cache del processore che l'ha modificato alla memoria condivisa; in seguito, però, esso deve esser passato anche a tutti gli altri processori in modo che essi non lavorino con un valore obsoleto
- La risoluzione di questo problema richiede delle implementazioni hardware in grado di gestire problemi di concorrenza e sincronizzazione, similmente a quelli che si ha con i thread a livello di programmazione

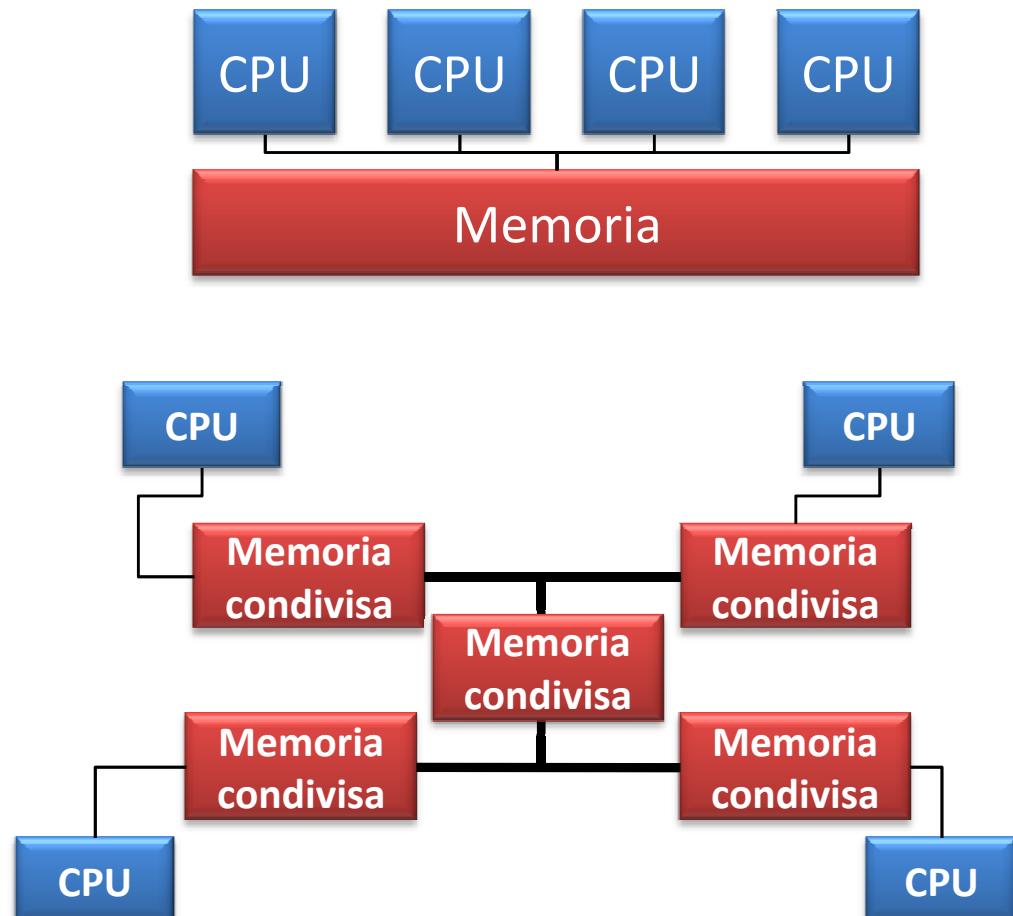


MULTIPROCESSORI

Memoria Condivisa: organizzazione

□ Classificazione memoria condivisa:

- **Uniform Memory Access (UMA):** il tempo di accesso alla memoria è costante per ogni processore e per qualsiasi locazione di memoria (Simmetric Multiprocessor,SMP)
 - Semplici da implementare ma non molto scalabili
- **NonUniform Memory Access (NUMA):** la memoria è suddivisa in una zona ad alta velocità assegnata singolarmente ad ogni processore ed una eventuale zona comune per lo scambio dei dati, ad accesso più lento (Distributed Shared Memory Systems, DSM)
 - Scalabili ma complessi da sviluppare





MULTIPROCESSORI

Memoria Distribuita: caratteristiche

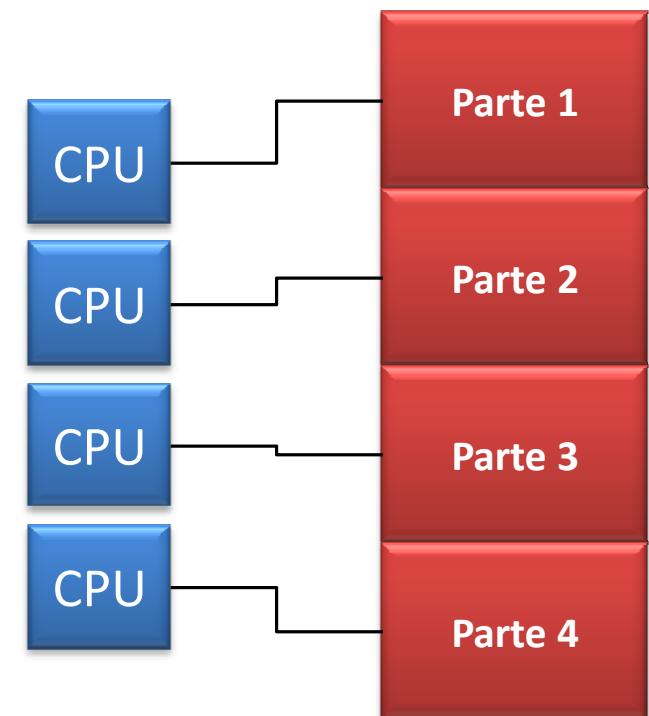
- Caratteristiche dei **sistemi multiprocessore a memoria distribuita**:
 - La memoria è fisicamente distribuita tra i vari processori; ogni memoria locale è accessibile direttamente solo dal suo processore
 - La **sincronizzazione** è ottenuta mediante lo **spostamento di dati** (un messaggio, *Message Passing*) tra i processori
 - Il **Message Passing** consiste nel far comunicare le CPU tra di loro tramite scambi di pacchetti dati. I messaggi trasmessi sono unità discrete di informazione; nel senso che hanno una identità ben definita, perciò deve essere sempre possibile poterli distinguere gli uni dagli altri
 - La suddivisione dei dati nelle memorie locali incide molto sulle prestazioni della macchina: è fondamentale fare una suddivisione accurata in modo da ridurre al minimo le comunicazioni tra le CPU. Inoltre, il processore che coordina queste operazioni di decomposizione e composizione deve comunicare efficacemente con i processori che operano sulle singole parti delle strutture dati



MULTIPROCESSORI

Memoria Distribuita

- In un sistema a **memoria distribuita** la memoria è associata ai singoli processori e un processore è solamente in grado di indirizzare la propria memoria. Questo tipo di sistema è detto anche “multicomputer”, riflettendo il fatto che i blocchi del sistema sono a loro volta piccoli sistemi completi di processore e memoria
- Questa organizzazione presenta diversi **vantaggi**:
 1. non vi sono conflitti a livello di bus o switch. Ogni processore può utilizzare l'intera larghezza di banda della propria memoria locale, senza subire interferenze da parte di altri processori
 2. la mancanza di un bus comune significa che non c'è limite intrinseco al numero di processori
 3. non ci sono problemi di coerenza della cache. Ogni processore è responsabile dei propri dati, e non deve preoccuparsi di aggiornare eventuali copie
- Lo **svantaggio** è la comunicazione interprocessore che è più difficile da implementare. Se un processore richiedesse dei dati presenti nella memoria di un altro processore, i due processori devono necessariamente scambiarsi dei messaggi tramite il **Message Passing**. Ciò introduce due fonti di rallentamento: per costruire e inviare un messaggio da un processore all'altro ci vuole tempo, e inoltre un qualsiasi processore deve essere interrotto al fine di gestire i messaggi ricevuti da altri processori





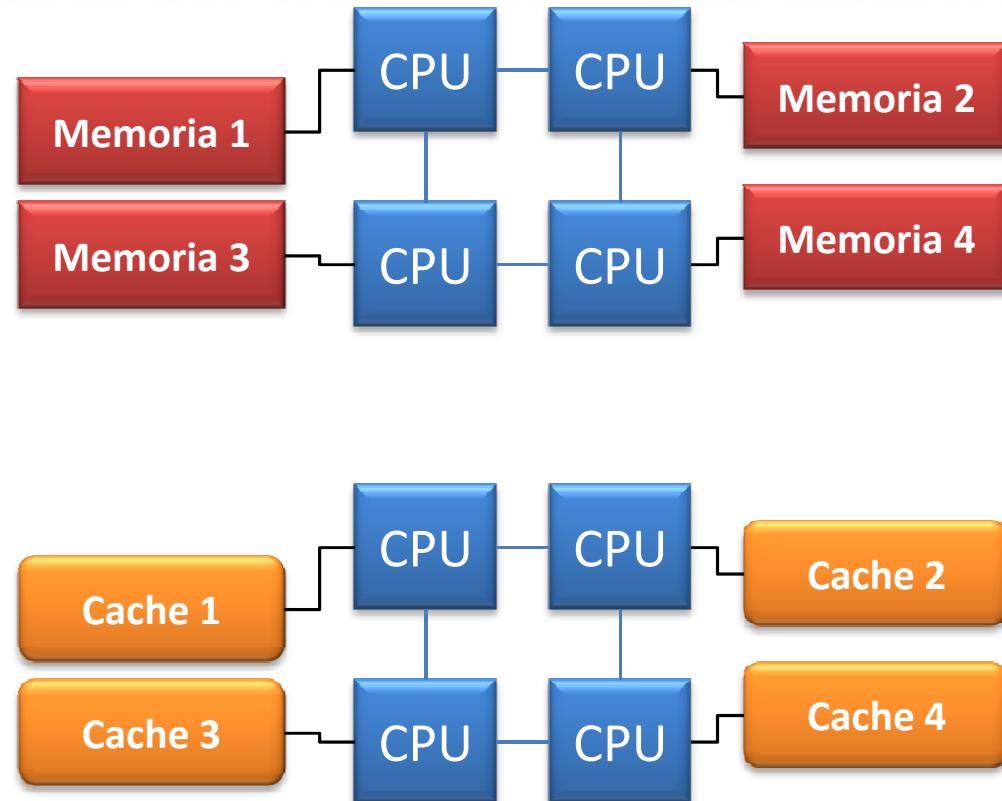
MULTIPROCESSORI

Memoria Distribuita: organizzazione

□ Classificazione memoria distribuita:

□ **NO-Remote Memory Access:** la memoria è distribuita fisicamente tra i processori (*local memory*). Tutte le memorie locali sono private e può accedervi solo il processore locale. La comunicazione tra i processori avviene tramite un protocollo di comunicazione per scambio di messaggi (*Message Passing*)

□ **Cache Only Memory Access:** questa tipologia di elaboratori sono dotati solamente di memorie cache. Analizzando le architetture NonUniform Memory Access si è notato che queste mantenevano delle copie locali dei dati nelle cache e che questi dati erano memorizzati come doppioni anche nella memoria principale. Questa architettura elimina i doppioni mantenendoli solo nelle memorie cache



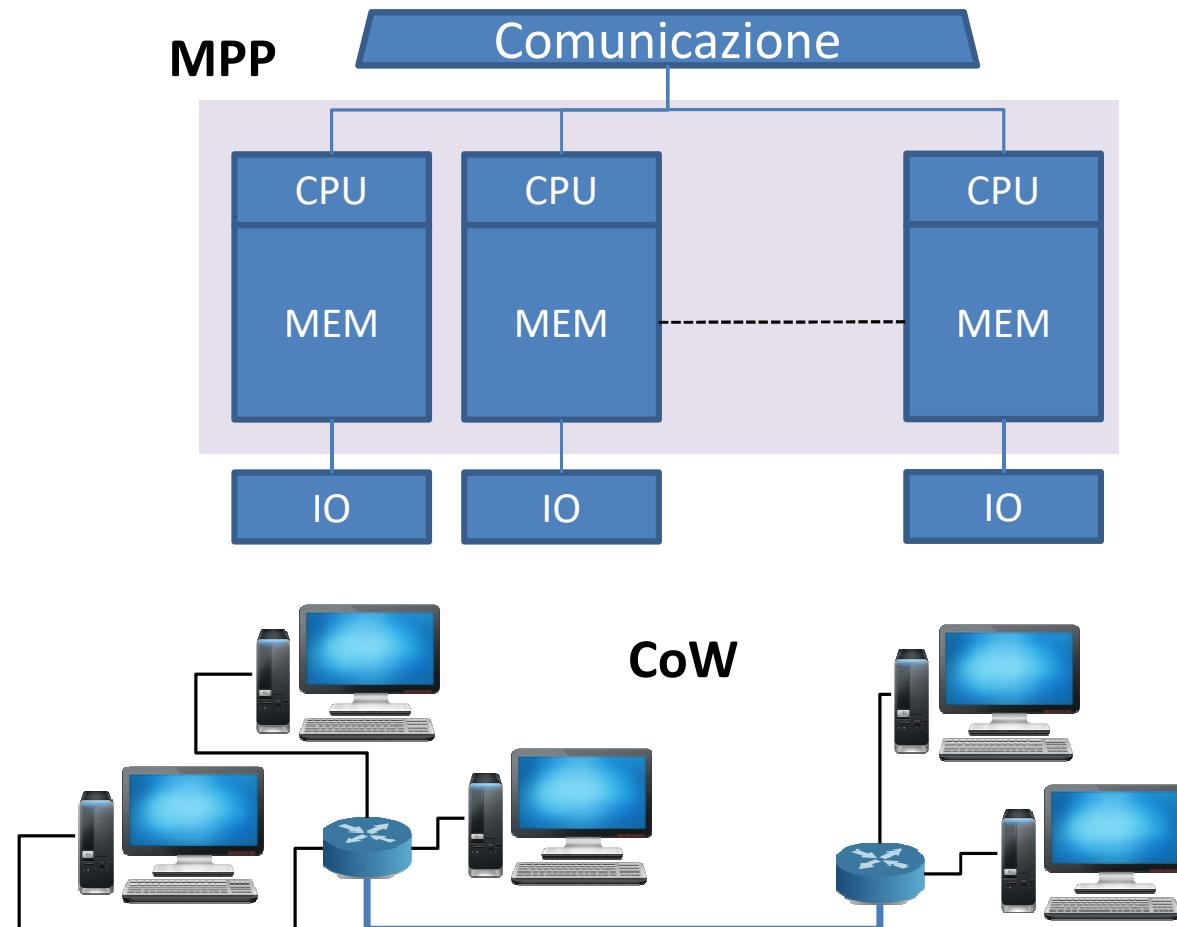
MULTIPROCESSORI

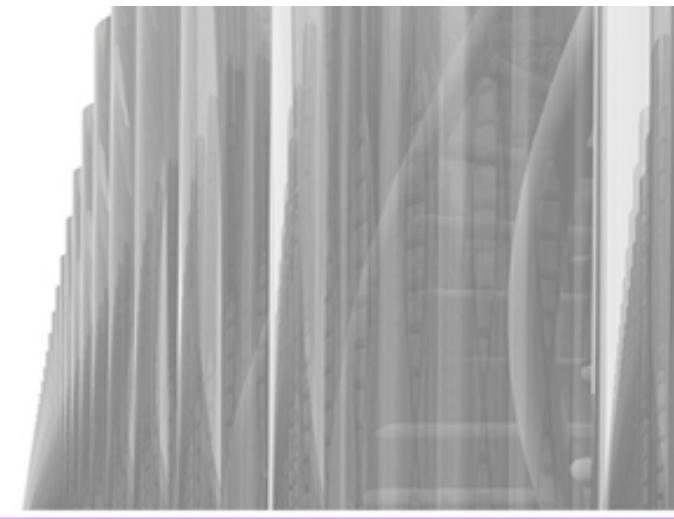
Memoria Distribuita: classificazione

- Classificazione dei sistemi multiprocessore a memoria distribuita:

- **Massively Parallel Processing:** composte da centinaia di processori (che possono diventare anche centinaia di migliaia in alcune macchine) collegati da una rete di comunicazione. Le macchine più veloci del pianeta (es.: Cray) sono basate su queste architetture

- **Cluster of Workstations :** le architetture CoW sono sistemi di elaborazione basati su calcolatori collegati da reti di comunicazione (si parla di cluster di calcolo)





Fine