ARCHITETTURE PER IL CALCOLO PARALLELO
(Prima Parte)

Argomenti

- INTRODUZIONE
  - Classificazione di Flynn (1966)
- PARALLELISMO NEL CHIP
  - Parallelismo a livello d'istruzioni
  - Multithreading nel chip
  - Multiprocessori in un solo chip
- COPROCESSORI
- MULTIPROCESSORI
  - Hardware
    - UMA con architettura basata sul bus
    - UMA con crossbar switch
    - UMA con multistage switching network
    - NUMA
  - Tipi di sistemi operativi
  - Sincronizzazione

Obiettivi

- Introduzione delle motivazioni che hanno spinto i progettisti verso la costruzione di architetture parallele.
- Capire il funzionamento del parallelismo all'interno di un singolo chip.
- Descrizione dell'hardware dei multiprocessori.
- Introduzione ai tipi di sistemi operativi e alla sincronizzazione dei multiprocessori.

INTRODUZIONE

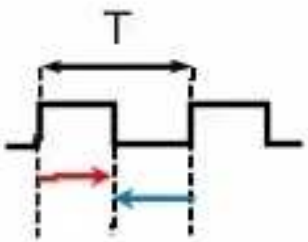
- L'obiettivo principale dell'industria dei computer è da sempre orientata verso l'incremento delle performance.
- In passato questo è stato possibile incrementando la frequenza del clock.
- Si è giunti così ad un limite fisico del materiale poiché, in accordo con la teoria della relatività di Einstein, nessun segnale elettrico può propagarsi più velocemente della velocità della luce (circa **20 cm/nsec** in un cavo di rame o nella fibra ottica).
- Questo significa che in un computer con un clock di 10 GHz, i segnali non possono percorrere più di 2cm.

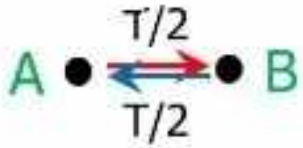
$$F = 10\,GH_z \rightarrow T = \frac{1}{F} = 0.1\,ns$$

$$spazio = velocità \cdot tempo = 20 \cdot \frac{cm}{ns} \cdot 0.1ns = 2cm$$

INTRODUZIONE

- Un computer con un clock di 1 THz (10^3 GHz) dovrebbe essere più piccolo di 100 μm, per permettere al segnale di andare avanti ed indietro in un ciclo di clock.





$$T = \frac{1}{10^{12}} = 10^{-3} ns$$

$$spazio = 20 \cdot \frac{cm}{ns} \cdot 10^{-3} ns = 2 \cdot 10^{-4} m = 200 \mu m$$

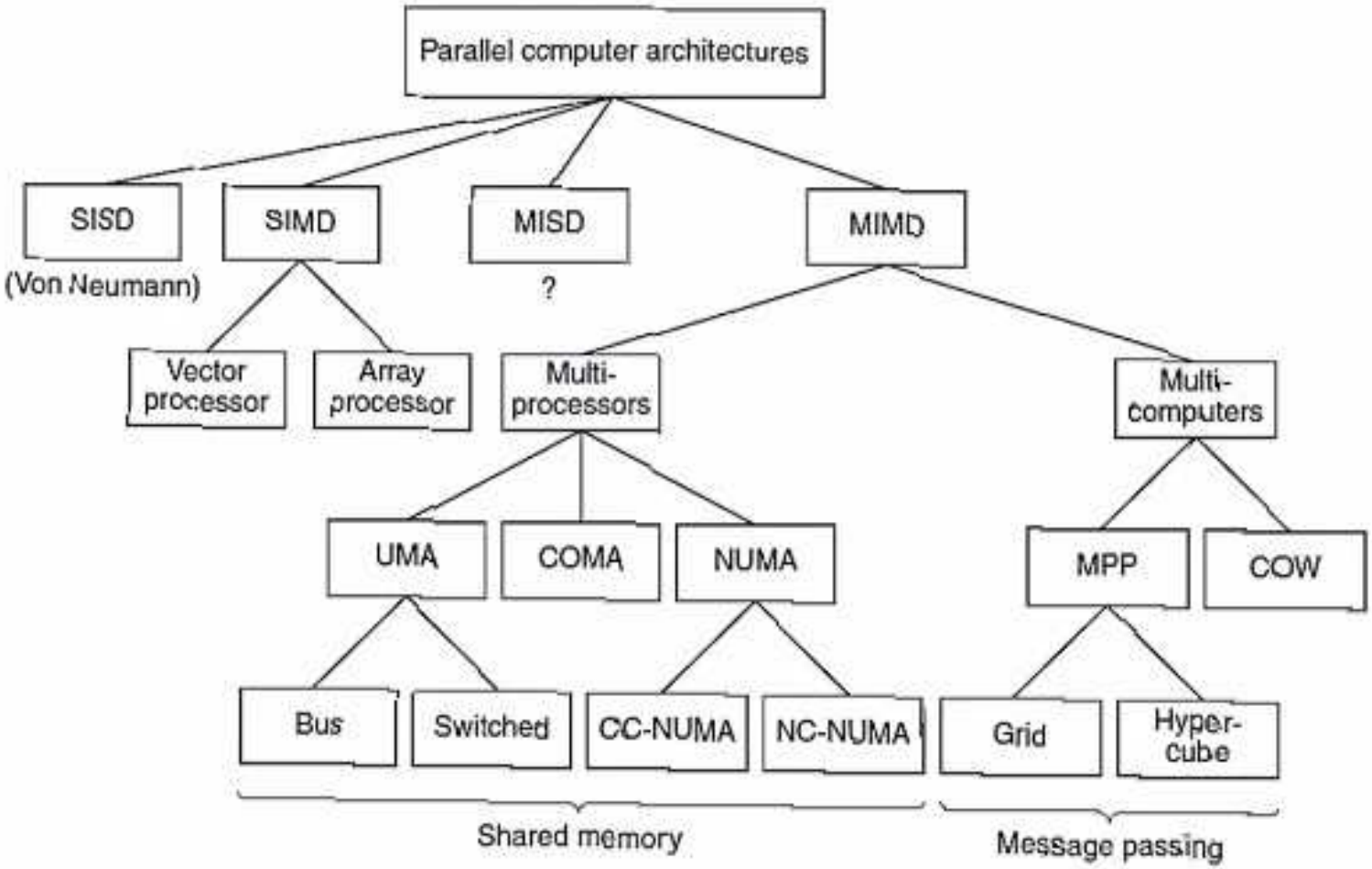
- Rendere computer così piccoli è possibile ma andiamo incontro ad un nuovo problema: la dissipazione del calore.
- Un modo per ottenere una maggiore potenza di calcolo è attraverso l'utilizzo delle architetture parallele: molte CPU (con velocità normale) che collaborano per il conseguimento del medesimo obiettivo.

Classificazione di Flynn (1966)

- La classificazione si basa su due concetti: il flusso di istruzioni ed il flusso dei dati.

Flusso di Istruzioni	Flusso di Dati	Nome	Esempio
Singolo	Singolo	SISD	Modello di Von Neumann
Singolo	Multiplo	SIMD	Supercomputer vettoriali
Multiplo	Singolo	MISD	Non sono note
Multiplo	Multiplo	**MIMD**	Multiprocessori e Multicomputer

Tassonomia dei calcolatori paralleli



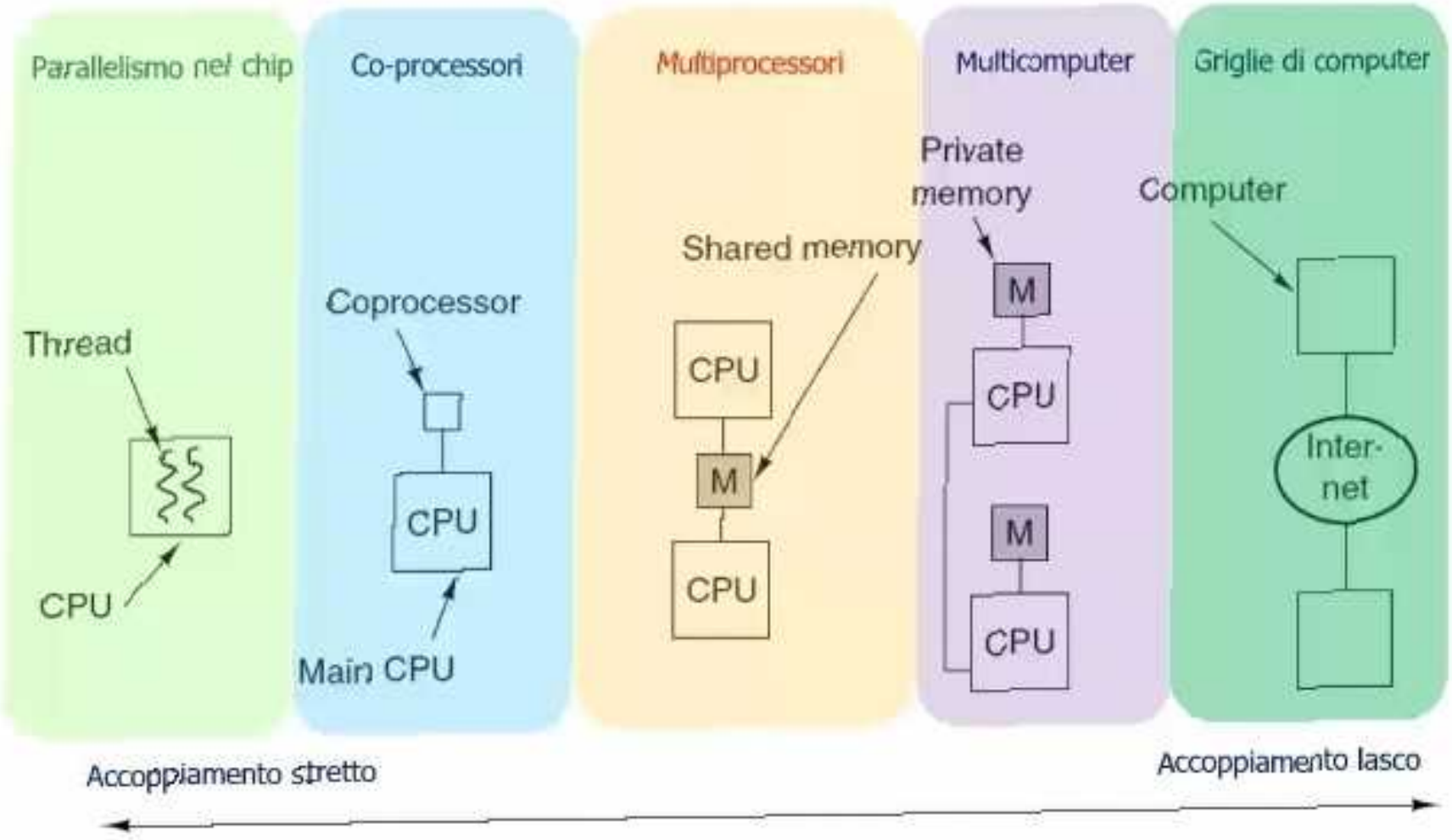
Architetture per il calcolo parallelo

- Il parallelismo nel chip aiuta a migliorare le performance della CPU: con il pipelining e le architetture superscalari si può arrivare ad un fattore di miglioramento da **5** a **10**.
- Per incrementare drasticamente le performance di un calcolatore occorre progettare sistemi con molte CPU, in questo caso di può arrivare ad ottenere un incremento di **50**, **100**, o anche di più
- Esistono così tre differenti approcci:
  - **Data Parallel Computers** (SIMD)
  - **Multiprocessors** (MIMD)
  - **Multicomputers** (MIMD)

Architetture per il calcolo parallelo

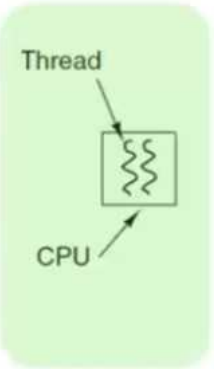
- Il parallelismo nel chip aiuta a migliorare le performance della CPU: con il pipelining e le architetture superscalari si può arrivare ad un fattore di miglioramento da **5** a **10**.
- Per incrementare drasticamente le performance di un calcolatore occorre progettare sistemi con molte CPU, in questo caso di può arrivare ad ottenere un incremento di **50**, **100**, o anche di più
- Esistono così tre differenti approcci:
  - **Data Parallel Computers** (SIMD)
  - **Multiprocessors** (MIMD)
  - **Multicomputers** (MIMD)

Quadro di sintesi delle architetture



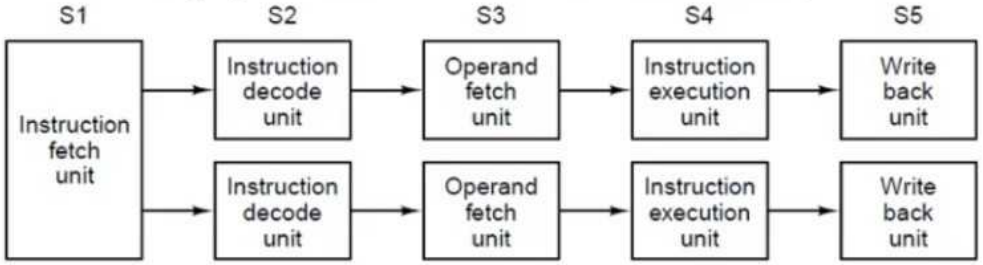
Parallelismo nel chip

- **Parallelismo a livello delle istruzioni**
  - L'idea è di emettere più istruzioni per ciclo di clock utilizzando processori superscalari e processori con parole di istruzione molto lunghe (Very Long Instruction Word)
- **Multithreading nel chip**
  - Superare lo stallo che si verifica nella pipeline quando si accede ad un indirizzo assente nelle cache (cache miss)
- **Multiprocessori in un solo chip**
  - più core sullo stesso chip che condividono le cache (primo e secondo livello) e la memoria principale

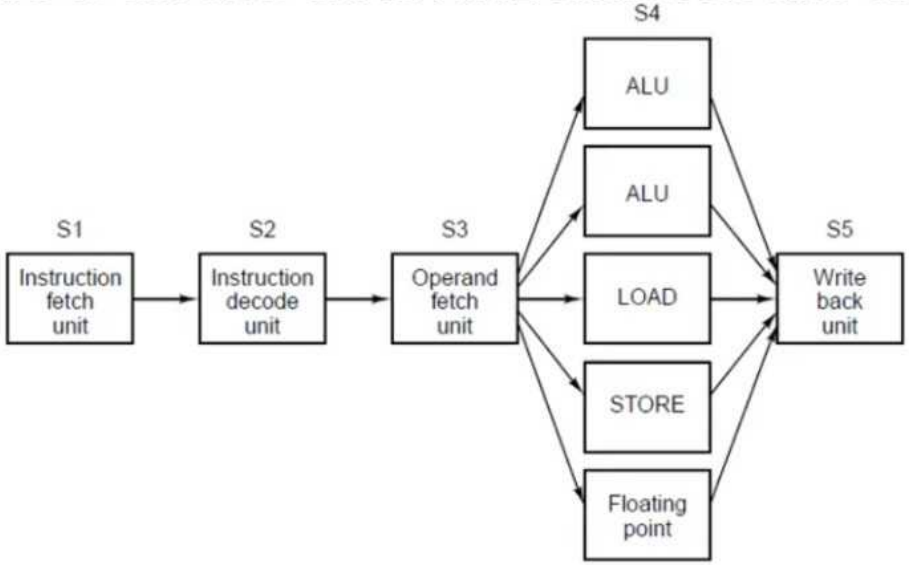


Parallelismo a livello delle istruzioni

- CPU superscalari: pipeline + unità funzionali parallele



- Processori VLIW (Very Long Instruction Word) in grado di indirizzare le diverse unità funzionali con una sola linea di pipeline



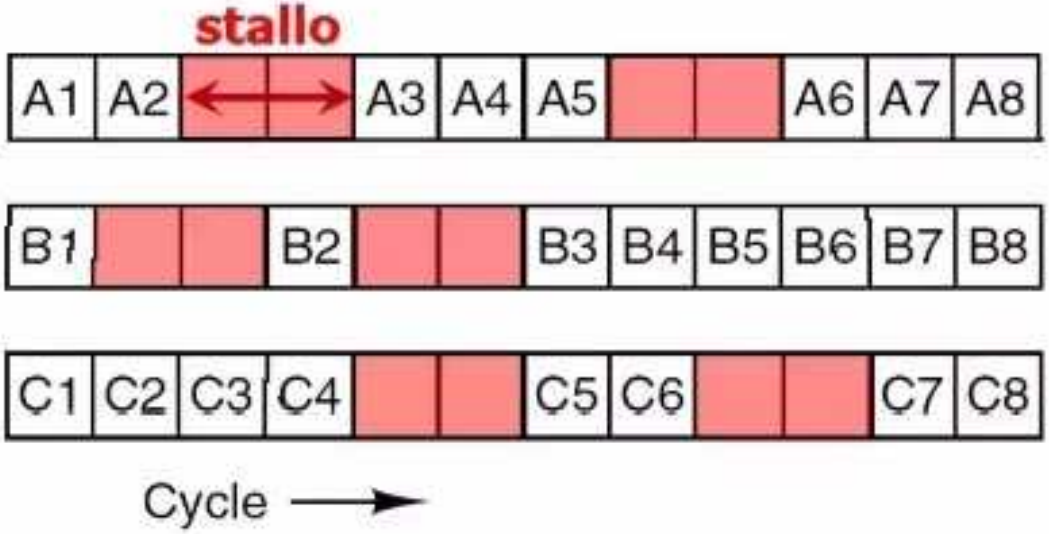
Il problema dello stallo della pipeline

- Il problema dello stallo si verifica quando una CPU tenta di accedere ad un riferimento in memoria che non è nella cache, quindi deve attendere il caricamento prima di riprendere l'esecuzione.

- Il **Multithreading nel chip** permette di mascherare queste situazioni attraverso lo switch tra thread, esistono differenti approcci:
  - **Multithreading a grana fine**
  - **Multithreading a grana grossa**
  - **Multithreading simultaneo**

Il problema dello stallo della pipeline

- Supponiamo di avere una CPU che emette una istruzione per ciclo di clock con tre thread A, B e C.
- Durante il primo ciclo A, B e C eseguono le istruzioni A1, B1 e C1
- Purtroppo al secondo ciclo di clock A2 fa un riferimento non presente nella cache di primo livello e deve attendere 2 cicli per recuperarlo dalla cache di secondo livello.



Multithreading a grana fine

- Sono eseguite ogni ciclo di clock, a turno, le singole istruzioni dei thread

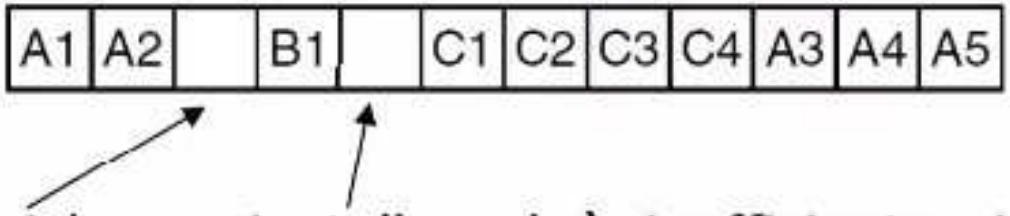
A1	B1	C1	A2	B2	C2	A3	B3	C3	A4	B4	C4

- Nei due cicli di stallo la CPU è occupata a svolgere le istruzioni degli altri due thread.

- Poiché non c'è alcuna relazione tra i thread, ciascuno ha il proprio insieme di registri. Il numero massimo di thread concorrenti è definito a priori in fase di progettazione del chip.
- Le operazioni in memoria non sono l'unica ragione di stallo: alcune istruzioni condizionano l'esecuzione di altre.
- Nella pipeline non ci sarà mai più di una istruzione per thread e quindi il numero massimo di thread è pari al numero di stadi della pipeline (non è detto che ci siano tanti thread quanti gli stadi della pipeline).

Multithreading a grana grossa

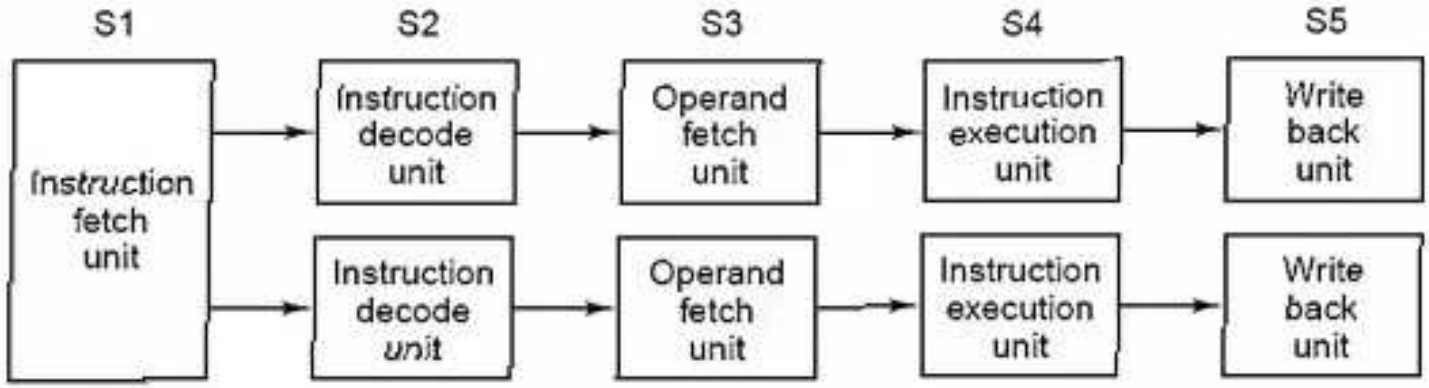
- Un thread va avanti finché non raggiunge uno stallo, perde un ciclo e commuta su un altro thread.



- Perde un ciclo ogni stallo ed è inefficiente rispetto al multithreading a grana fine.
- richiede meno thread per mantenere occupata la CPU
- Esiste una variante che permette di "guardare avanti" le istruzioni anticipando lo stallo e approssimando il multithreading a grana fine.

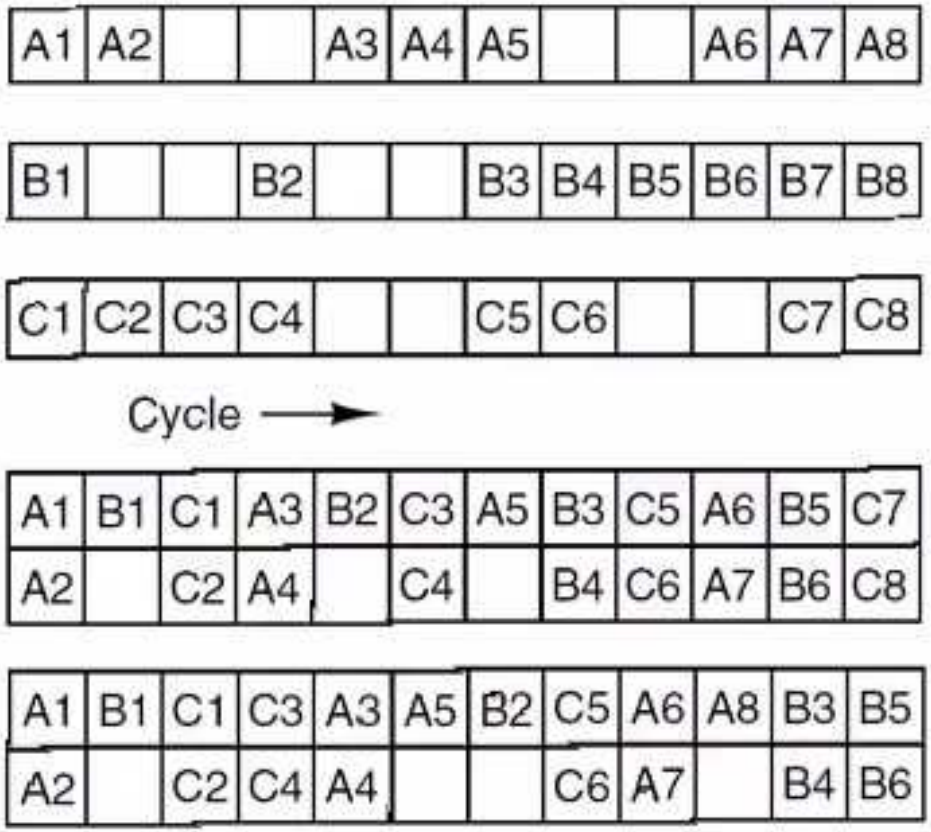
Multithreading in CPU superscalari

- Le CPU possono avere più unità in parallelo ed emettere più istruzioni per ciclo



Multithreading in CPU superscalari

- In ogni thread sono eseguite due istruzioni per ciclo finché non si raggiunge uno stallo:



Esempio iniziale

Multithreading a Grana fine

Multithreading a Grana grossa

Multithreading simultaneo

A1	B1	C1	A3	B2	C3	A5	B3	C5	A6	B5	C7
A2		C2	A4		C4		B4	C6	A7	B6	C8

Multithreading a Grana fine

A1	B1	C1	C3	A3	A5	B2	C5	A6	A8	B3	B5
A2		C2	C4	A4			C6	A7		B4	B6

Multithreading a Grana grossa

- Ciascun thread emette due istruzioni per ciclo finché non si raggiunge uno stallo

A1	B1	C2	C4	A4	B2	C6	A7	B3	B5	B7	C7
A2	C1	C3	A3	A5	C5	A6	A8	B4	B6	B8	C8

- A questo punto si passa all'istruzione del thread che segue affinché la CPU rimanga impiegata

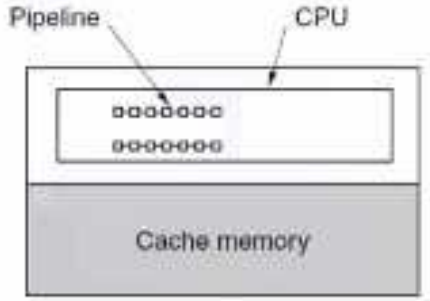
Multiprocessori in un solo chip

- Con l'andare avanti della tecnologia, i transistor sono diventati più piccoli ed è stato possibile incrementare il loro numero in un singolo chip.
- A causa del raggiungimento dei limiti fisici dei materiali (dissipazione del calore e distanza percorsa dai segnali all'interno dei semiconduttori) non è stato possibile incrementare la frequenza di clock per ottenere CPU più veloci.
- Per tale ragione le industrie produttrici di chip hanno incominciato ad inserire più CPU (chiamate **core**) all'interno dello stesso chip (o meglio **die**).
- I multiprocessori possono essere realizzati con core identici (**multiprocessori omogenei**) oppure con core con specifiche funzionalità (**multiprocessori eterogenei**).

Multiprocessori omogenei in un solo chip

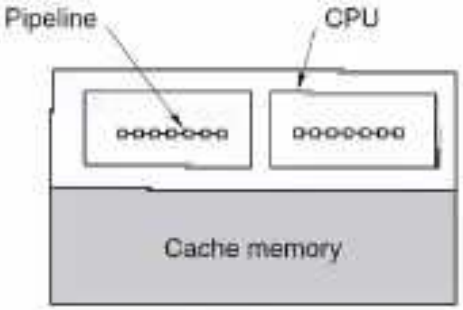
- Le CPU che contengono più processori e che condividono le cache e la memoria principale sono dette multiprocessori.
- Esistono due tecnologie *di* multiprocessori in un solo chip:

1) Quelle con una sola CPU e più pipeline che possono moltiplicare il throughput in base al numero di pipeline. Le unità funzionali sono intimamente correlate.

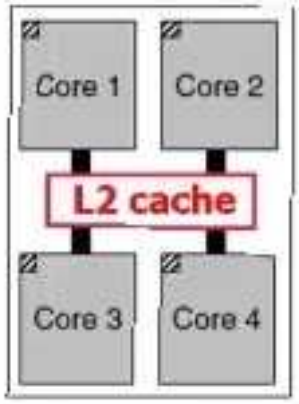


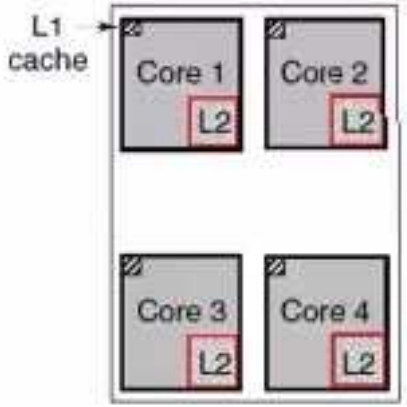
2) Quelle che hanno più CPU (chiamate core) ciascuna con la propria pipeline.

   L'interazione tra CPU non è semplice poiché risultano maggiormente disaccoppiate.



Multiprocessori omogenei in un solo chip





- Mentre le CPU possono o meno condividere le cache, esse condividono sempre la memoria principale

- Il processo di **snooping** (eseguito dall'hardware) garantisce che se una parola è presente in più cache e una CPU modifica il suo valore in memoria, essa è automaticamente rimossa in tutte le cache in modo da garantire consistenza.

Multiprocessori eterogenei in un solo chip

- Oltre ai multicore simmetrici esiste un altro tipo di chip multicore in cui ogni core ha un compito specifico (come ad esempio decoder audio/video, criptoprocessore, interfacce di rete).
  - Poichè queste architetture realizzano un vero e proprio calcolatore completo in un singolo chip sono spesso dette **system on a chip**.
- Come accaduto spesso nel passato, l'**hardware è molto più avanti del software:** mentre sono attualmente disponibili dei chip multicore, non abbiamo applicazioni in grado di sfruttare queste nuove caratteristiche.
- Pochi programmatori sono in grado di scrivere algoritmi paralleli che gestiscano correttamente la competizione delle risorse condivise.

Chip-level Multiprocessor

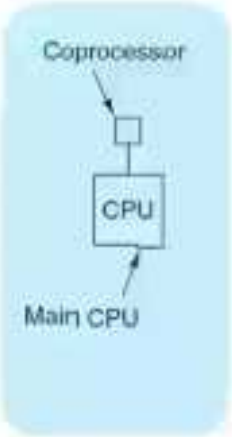
- I chip multicore sono come dei piccoli multiprocessori e per questa ragione vengono chiamati **CMP** (**Chip-level MultiProcessors** ovvero multiprocessori a livello di chip).
- Dal punto di vista software essi non non sono così differenti dai multiprocessori a bus o a reti di switch.
- Rispetto a multiprocessori a bus che hanno una cache per ogni CPU, potrebbero avere delle prestazioni degradate sulla cache condivisa nelle situazioni in cui un core "ingordo" saturi la cache L2.
- Altra differenza rispetto ai multiprocessori è la minore tolleranza ai malfunzionamenti causata dalla stretta connessione dei core (un errore su uno potrebbe propagarsi negli altri).

Chip-level Multiprocessor

- I chip multicore sono come dei piccoli multiprocessori e per questa ragione vengono chiamati **CMP** (**Chip-level MultiProcessors** ovvero multiprocessori a livello di chip).
- Dal punto di vista software essi non non sono così differenti dai multiprocessori a bus o a reti di switch.
- Rispetto a multiprocessori a bus che hanno una cache per ogni CPU, potrebbero avere delle prestazioni degradate sulla cache condivisa nelle situazioni in cui un core "ingordo" saturi la cache L2.
- Altra differenza rispetto ai multiprocessori è la minore tolleranza ai malfunzionamenti causata dalla stretta connessione dei core (un errore su uno potrebbe propagarsi negli altri).

Coprocessori

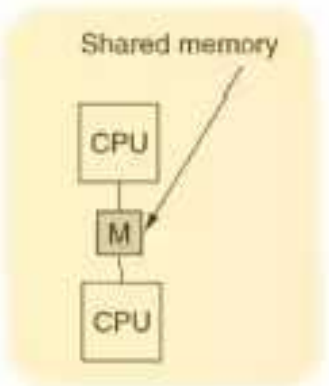
- L'utilizzo di un processore dedicato *a* specifiche funzioni (**coprocessore**) premette *di* migliorare le performance di un calcolatore.

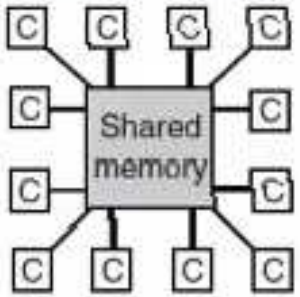


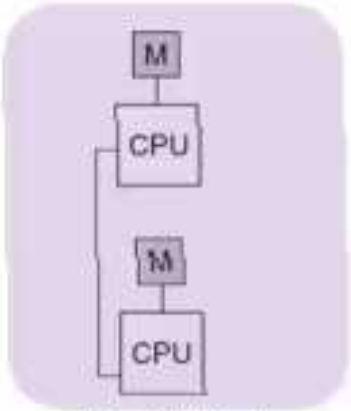
- Esistono molte varianti di coprocessori:
  - Processori di rete.
  - Processori grafici.
  - Crittoprocessori.

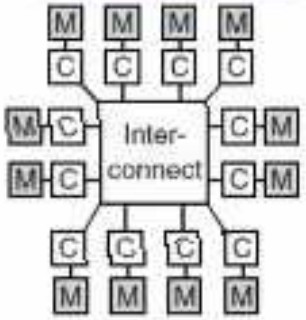
MIMD

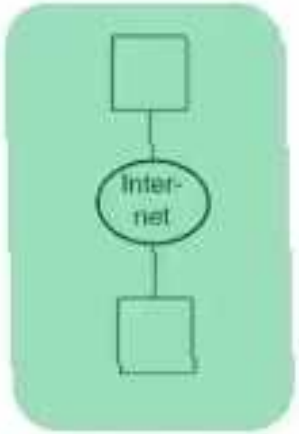
- Tutte le comunicazioni tra componenti elettronici (o ottici) avvengono attraverso lo scambio di messaggi.
- Le principali differenze tra MIMD risiedono nella base dei tempi, nella scala delle distanze e nell'organizzazione logica utilizzata

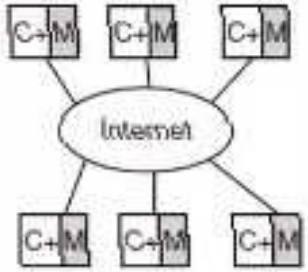






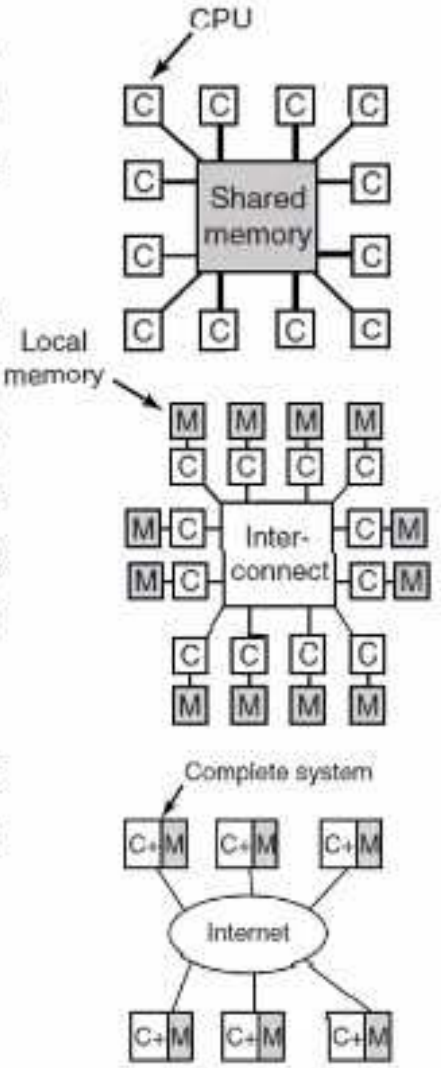






Multiprocessori e Multicomputer (MIMD)

1) **Multiprocessori a memoria condivisa**, meno di **1000** CPU comunicano attraverso una memoria condivisa; il tempo di accesso ad una parola di memoria è di **2-10** ns.
2) **Multicomputer a scambio di messaggi**, un certo numero di coppie memoria-CPU sono collegati attraverso una rete di interconnessione ad alta velocità; le comunicazioni avvengono attraverso brevi messaggio che possono essere spediti in **10-50** µs; ogni memoria è locale a ciascuna CPU.
3) **Sistemi distribuiti**, computer distribuiti su un'area geografica estesa come internet, i messaggi impiegano da **10** a **100** ms.



Multiprocessori

- Un multiprocessore a memoria condivisa (o semplcemente multiprocessore) è un calcolatore in cui tutte le CPU condividono una memoria comune.

- L'unica proprietà inusuale che ha un multiprocessore è che una CPU può scrivere un valore in una parola di memoria e tovarvi un differente valore alla successiva lettura (perché un'altra CPU può averla modificata).

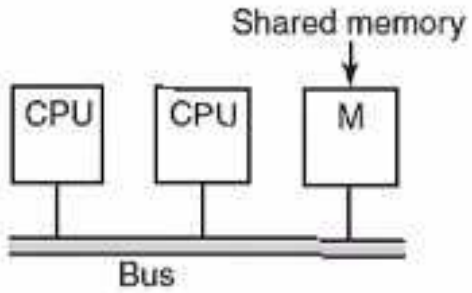
- I sistemi operativi utilizzati nei Multiprocessori sono normali sistemi operativi.

Hardware dei multiprocessori

- Esistono due differenti tipologie di multiprocessori:
  - **Multiprocessori UMA** (Uniform Memory Access) in cui ogni parola di memoria può essere letta alla stessa velocità.
  - **Multiprocessori NUMA** (Nonuniform Memory Access) in cui non tutte le parole di memoria possono essere lette alla medesima velocità.

UMA con architettura basata sul bus

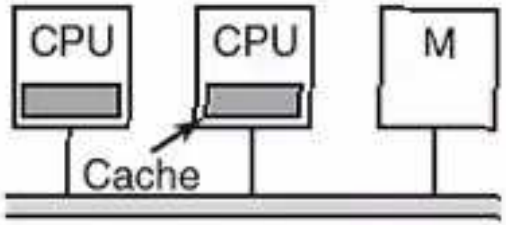
- I più semplici multiprocessori sono basati su un singolobus che interconnnette tutte le CPU alla memoria condivisa.



- Una CPU che vuole leggere/scrivere una parola in memoria, se il bus è occupato, deve attendere che si liberi.
- Questa architettura funziona bene con due o tre CPU (il contenzioso sull'accesso al bus può essere gestito agevolmente).
- Per poter incrementare il numero di CPU, e quindi le performance, occorre aggiungere delle ulteriori memorie.

UMA con singolo bus e cache nelle CPU

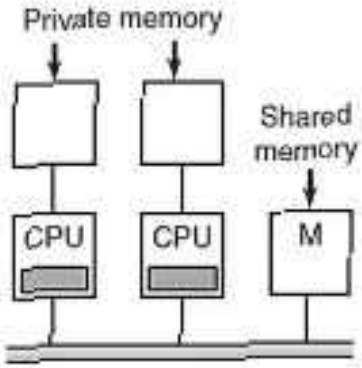
- L'aggiunta di una memoria cache all'interno delle singole CPU può ridurre il traffico sul bus e il sistema può supportate anche più di tre CPU:



- Il Caching non viene eseguito sulle singole parole di memoria ma su blocchi di 32 o 64 byte. Quando una parola è referenziata, l'intero blocco che la contiene, chiamato **linea di cache**, è caricato nella CPU che l'ha richiesta.
- Ogni blocco di cache è contrassegnato come READ-ONLY oppure READ/WRITE.

UMA con singolo bus e CPU dotate di RAM

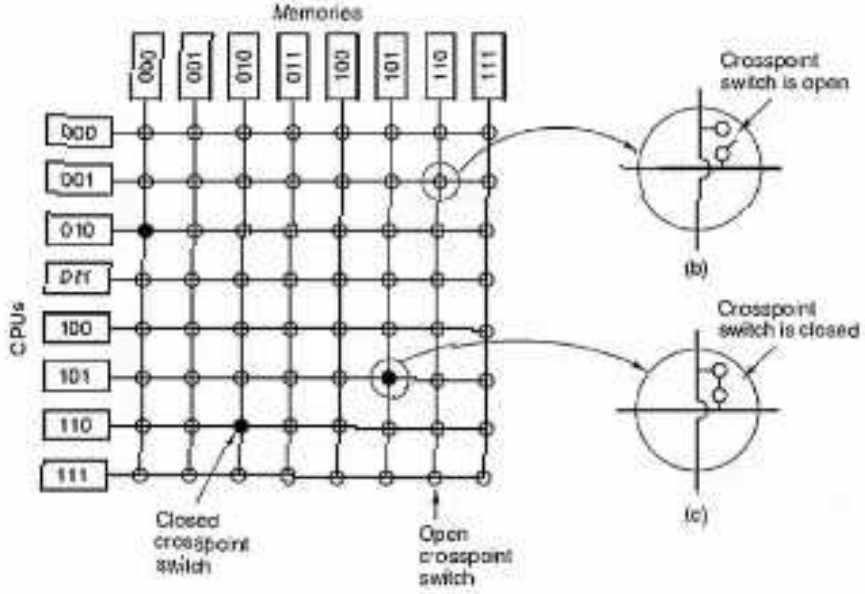
- Un altra soluzione è con l'aggiunta di memorie RAM in un bus interno dedicato per ogni CPU



- A questo punto la memoria condivisa è utilizzata esclusivamente per scrivere variabili condivise (globali).
- Questa soluzione riduce il traffico sul bus ma richiede una collaborazione attiva del compilatore che deve separare gli oggetti locali (il programma, lo stack, le variabili locali,...) da quelli globali.

UMA con crossbar switch

- Anche con le migliori tecniche di caching, l'uso di un singolo bus di interconnessione limita la dimensione del multiprocessore UMA a 16 o 32 CPU.
- Per andare oltre occorre utilizzare una differente rete di interconnessione.
- Il circuito più semplice che permette di collegare n CPU a k memorie è il **crossbar switch**.



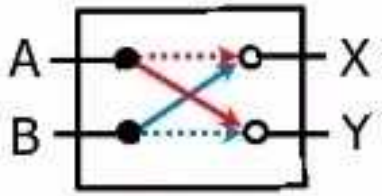
- In ogni intersezione delle linee orizzontali (CPU) con quelle verticali (RAM) c'è un piccolo interruttore (pilotato elettricamente) che permette di stabilire il collegamento tra CPU e RAM.

UMA con crossbar switch

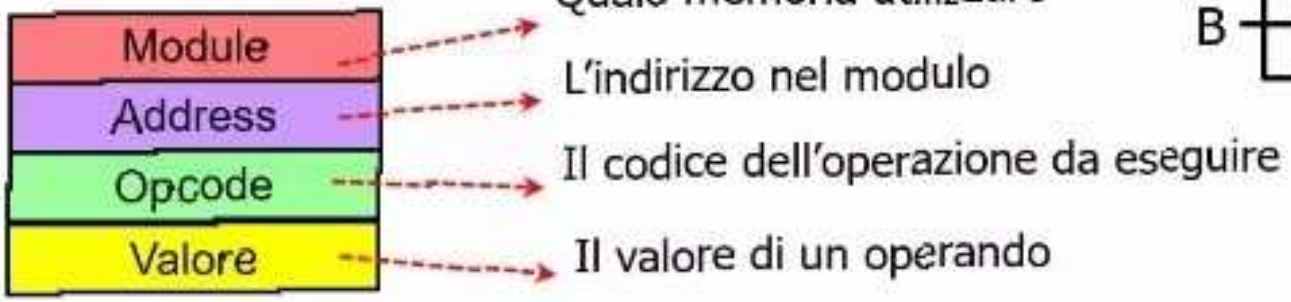
- Attraverso questo sistema si realizza una **rete non bloccante**: ad alcuna CPU è mai vietata la connessione verso una memoria di cui ha bisogno perché qualche crosspoint o linea è già occupata.
- Non è necessaria alcuna azione di pianificazione anticipata. È sempre possibile connettere una CPU alla memoria aprendo o chiudendo un interruttore.
- Rimane il problema della competizione per la memoria, qualora due, o più, CPU vogliono accedere allo stesso modulo nel medesimo istante (partizionando la memoria in n unità, la competizione si riduce di un fattore n rispetto al modello con bus singolo).
- Una delle peggiori caratteristiche di questo schema è che il numero degli incroci cresce come n^2.
  - con 1000 CPUs e 1000 moduli di memoria occorrono un milione di crosspoints. Costruire una crossbar di queste dimensioni non è fattibile.

UMA con rete di commutazione a più stadi

- Una progettazione di multiprocessori completamente differente è basata su switch 2x2 (due input e due output).
- I messaggi che arrivano nelle due linee di ingresso possono scambiarsi in una delle due linee di uscita.



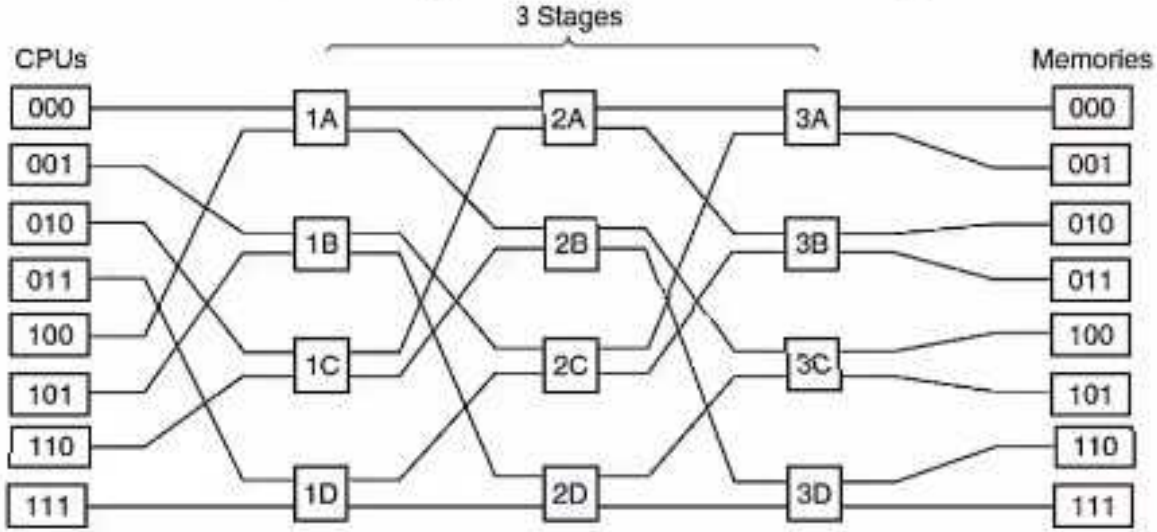
- Ogni messaggio contiene:



- Lo switch utilizza il campo Module per scegliere dove spedire il messaggio (su X o su Y).
- Gli switch possono essere organizzati in vari modi per costruire complesse reti di commutazione a più stadi.

UMA con rete di commutazione a più stadi

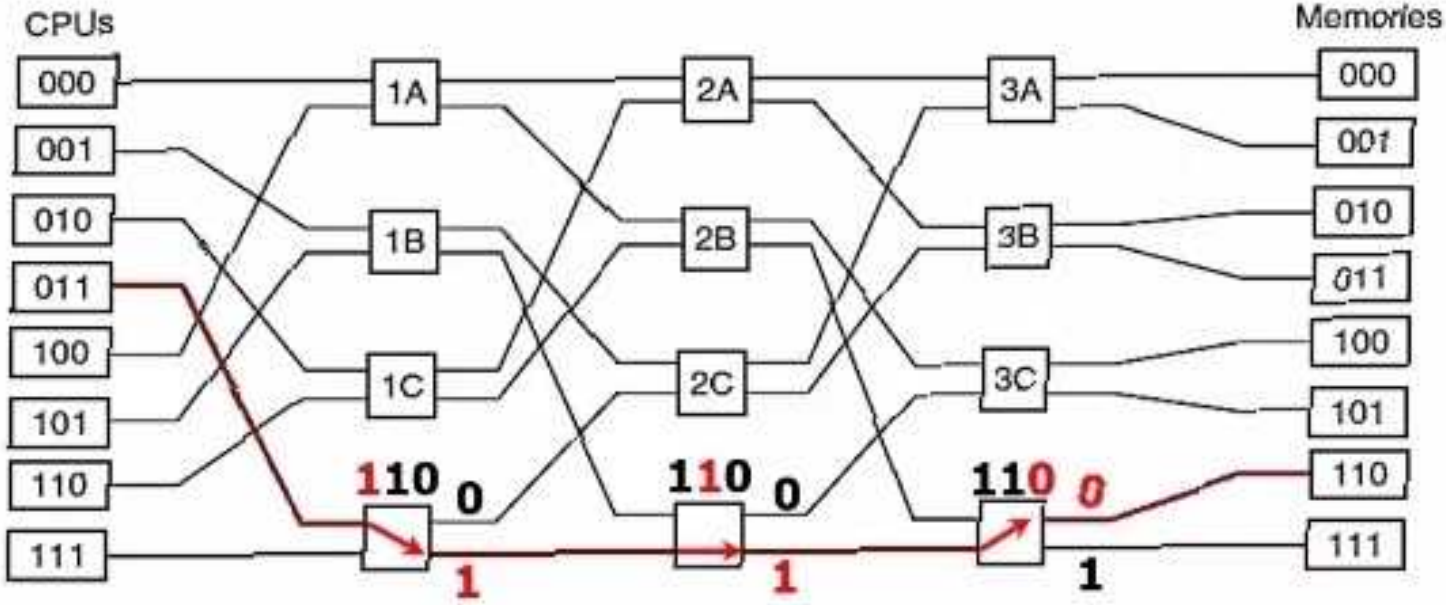
- Una rete economica e semplice è la **rete omega**.



- In questo esempio sono state connesse 8 CPU ad 8 memorie utilizzando 12 switch.
- Per n CPU e n memorie sono necessari $\log_2 n$ stadi (o stage), con n/2 switch per stadio, per un totale di **$(n/2)\log_2 n$** switch, che è decisamente meglio di **n^2** crosspoint, soprattutto per valori grandi di n.

UMA con rete di commutazione a più stadi

- Lo schema di connessione di una rete omega è detto **shuffle** (o miscuglio) **perfetto**.
- Per vedere come una rete omega lavora supponiamo che la CPU numero 3 (011_2) voglia leggere una parola nella memoria 6 (110_2).



UMA con rete di commutazione a più stadi

- A differenza dell'interconnessione con crossbar switch, la rete omega è una rete **bloccante**: non tutti gli insiemi di richieste possono essere processati contemporaneamente.

- È auspicabile distribuire i riferimenti alla memoria in modo uniforme rispetto ai moduli. Un sistema di memorie in cui le parole consecutive sono in moduli diversi è detto **interleaved**.

- Le memorie interleaved massimizzano il parallelismo perché la maggior parte dei riferimenti è in indirizzi consecutivi.

- È sempre possibile progettare reti di switch non bloccanti.

Multiprocessori NUMA

- Con i processori UMA a singolo bus si può arrivare a connettere insieme fino a 12 CPU, con una rete di interconnessione, a causa dei costi dell'hardware, si può arrivare a connettere insieme meno di 100 CPU.
- Per eccedere questo valore occorre introdurre una nuova architettura in cui si rinuncia a qualcosa: che i tempi di accessi della memoria siano uniformi.
- In un processore NUMA il tempo di accesso ai moduli di memoria locale è minore rispetto a quello dei moduli remoti.
- Tutti i programmi che girano su multiprocessori UMA continuano a girare sulle macchine NUMA, ma naturalmente hanno performance degradate.

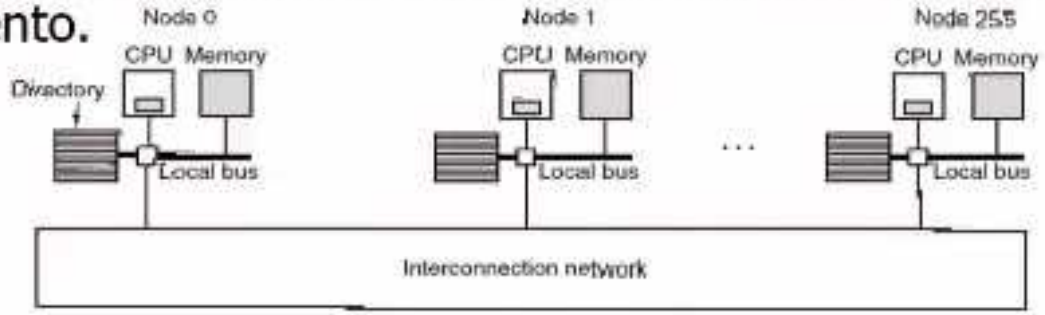
Multiprocessori NUMA

- Le macchine NUMA hanno tre caratteristiche chiavi:
  1) C'è un unico spazio di indirizzamento visibile a tutte le CPU.
  2) L'accesso alla memoria remota è attraverso istruzioni di LOAD e STORE.
  3) L'accesso alla memoria remota è più lento dell'accesso rispetto alla memoria locale.

- Ci sono due tipi di macchine NUMA
  1) **No Cache** - Nonuniform Memory Access (NC-NUMA).
  2) **Cache Coherent** - Nonuniform Memory Access (CC-NUMA).

Multiprocessori NUMA

- L'approccio più comune utilizzato per costruire grandi multiprocessori CC-NUMA è il multiprocessore basato sulle directory (**directory-based multiprocessor**).
- L'idea di fondo è di mantenere una base dati delle linee di memoria ed il loro stato. Quando è referenziata una linea di cache, viene interrogata la base dati per sapere dove si trova e se è "pulita" o "sporca".
- Poiché ogni istruzione che accede alla memoria interroga questo database, occorre che sia gestito su un hardware specifico velocissimo.
- L'unione di tutte le memorie dei nodi costituisce l'intero spazio di indirizzamento.



Tipi di sistemi operativi multiprocessore

- Ci sono tre differenti approcci per i sistemi operativi multiprocessori:

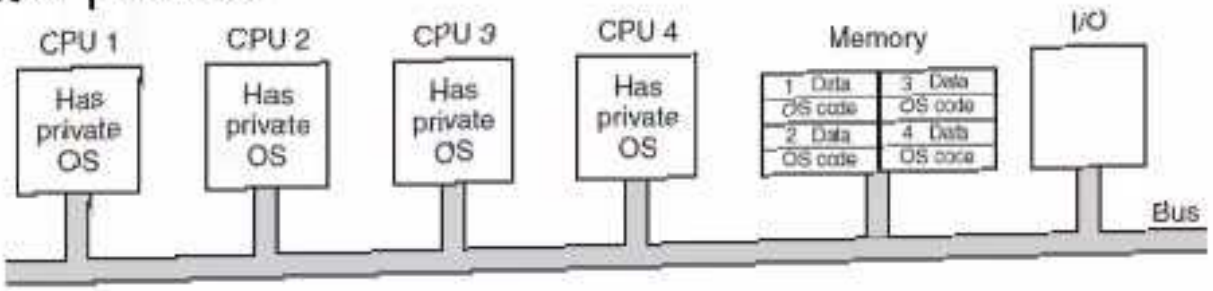
1) **Ogni CPU ha un proprio sistema operativo**

2) **Multiprocessori master-slave**

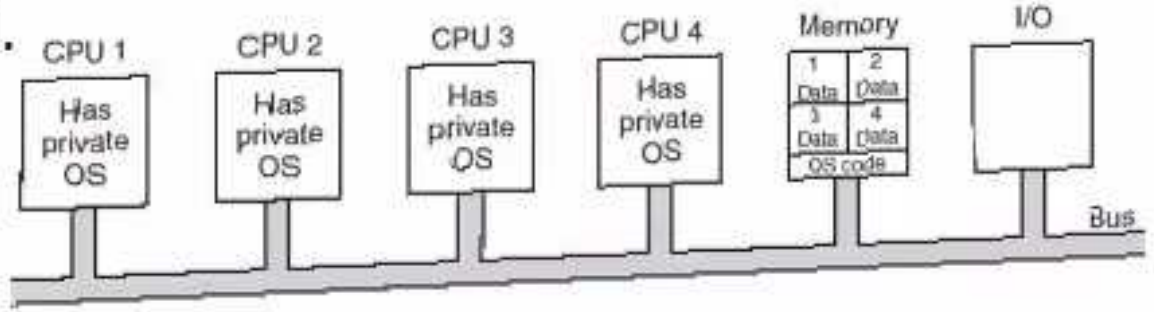
3) **Multiprocessori simmetrici**

Ogni CPU ha un proprio sistema operativo

- La soluzione più semplice è di **dividere staticamente** la memoria in tante partizioni quante sono le CPU.
- Ogni CPU ha un proprio sistema operativo e una a propria memoria privata.



- Le CPU operano con computer indipendenti, una possibile ottimizzazione è attraverso la condivisione del sistema operativo tha CPU.

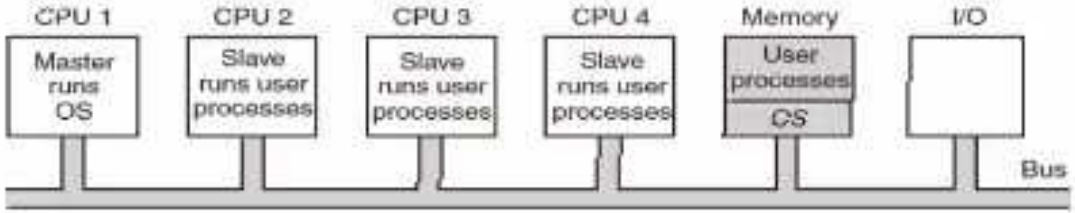


Ogni CPU ha un proprio sistema operativo

- Problematiche:
  1) Quando un processo esegue una system call, questa è gestita dalla CPU che l'ha lanciato utilizzando le strutture dati presenti nelle tabelle di quel sistema operativo.
  2) Non c'è condivisione di processi, ogni CPU gestisce indipendentemente i propri processi. Quindi non è possibile il bilanciamento del carico (una CPU potrebbe essere scarica mentre un'altra satura).
  3) Sistemi operativi indipendenti non condividono pagine di memoria. Una CPU può eseguire molto swap a causa della carenza di spazio libero mentre un'altra può disporre di pagine libere.
  4) Se un sistema operativo mantiene un buffer cache dei blocchi del disco usati recentemente independentemente dagli altri questo può condurre a letture inconsistenti (causate da letture sporche nei vari buffer).

Multiprocessori master-slave

- Il sistema operativo e le sue tabelle è presente solo sulla CPU 1 (**master**). Tutti le *system call* sono redirette sulla CPU 1 per essere processate



- Il modello master-slave resolve la maggior parte dei problemi del modello precedente:
  - C'è una singola struttura dati che che tiene traccia dei processi pronti.
  - La CPU master può bilanciare il carico sulle varie CPU ma può anche diventarne il collo di bottiglia.
  - Le pagine sono allocate dinamicamente tra i processi e c'è una sola buffer cache che evita le inconsistenze.
- Proprio perché la CPU master può diventare critica, il modello funziona con un numero ridotto di CPU.

La sincronizzazione dei multiprocessori

- Le CPU in un multiprocessore hanno bisogno di sincronizzarsi: le regioni critiche del kernel e le tabelle devono essere protette da mutex.
- Se si disabilitano gli interrupt di una CPU, le altre CPU continuano a funzionare normalmente e possono accedere ad una regione critica.
- In un mono processore l'istruzione TSL (Test e Set Lock), impiegata nei mutex, permette ad una parola di memoria di essere controllata e impostata in una sola operazione indivisibile.
- Con più CPU, l'istruzione TSL deve prima bloccare il bus, evitando l'accesso di altre CPU, poi accedere alla memoria e quindi sbloccare il bus.

Conclusioni

- È stata analizzata la modalità con la quale si realizza il parallelismo all'interno di un singolo chip.
- Introdotte le diverse architetture di calcolo parallelo.
- Studiato i diversi hardware dei multiprocessori.
- Introdotto i tipi di sistemi operativi utilizzati dai multiprocessori.