Architettura degli Elaboratori Elettronici

Multiprocessori e Multicore

Dott. Franco Liberati
liberati@di.uniroma1.it

ARGOMENTI DELLA LEZIONE

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

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 e di elaborazione dati
  - l'impossibilità di miniaturizzare i componenti al di sotto di una certa soglia (per motivi fisici) quindi architetture più complesse e ricche di registri ausiliari
  - 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 quindi nella fascia di mercato dominata dal consumatore medio
- Per questo motivo si è **aumentato il numero di processori** ed è stato introdotto il **calcolo parallelo**

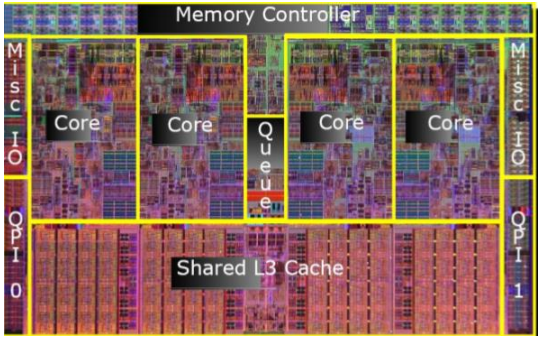
MULTIPROCESSORI

Definizione

- I sistemi **multiprocessore** e **multi-core**, sono definiti come:
  - **Multiprocessore**: "*Sistema equipaggiato con 2 processori, o più, operanti in parallelo*"
    - **PRO**
      - Alta ridondanza (se si rompe un processore si può continuare a lavorare a fronte di prestazioni inferiori)
    - **CONTRO**
      - uso non vantaggioso della memoria cache
      - Deduplicazione del circuito stampato
      - Maggior consumo elettrico
  - **Multicore**: "*Sistema composto da due o più **core**, cioè un nucleo del processore (UC e ALU), montato sullo stesso chip*"
    - **PRO**
      - 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
      - Scarsa ridondanza

Intel Xeon(4 CPU)

Intel i7 (4 core)



MULTIPROCESSORI

Curiosità

- Quanti core può avere un sistema di calcolo?

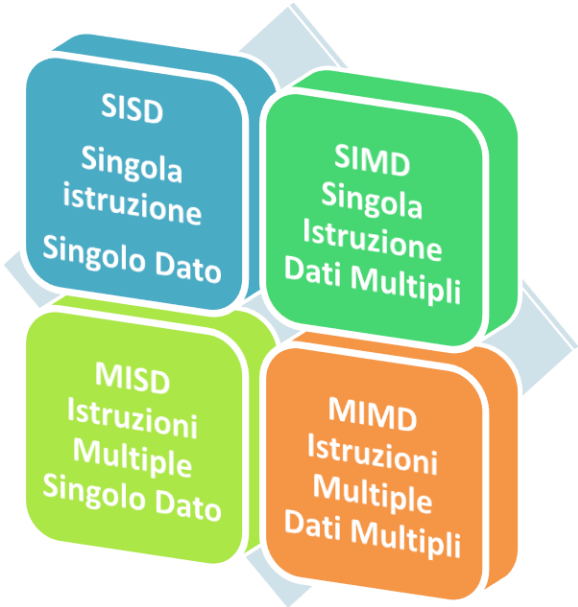
https://top500.org/lists/top500/2023/11/

MULTIPROCESSORI

Classificazione di Flynn per Architettura Parallela

- Nel 1966 Michael J. Flynn definì la tassonomia più completa e popolare 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 scorrano su due cablature differenti o meno

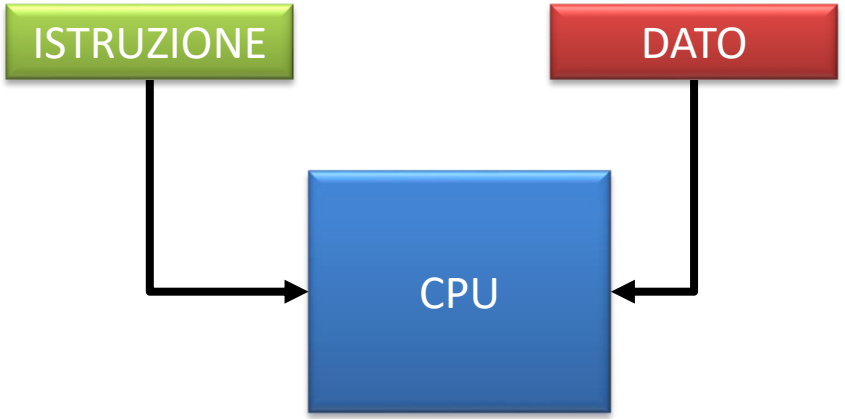
Tassonomia di Flynn



MULTIPROCESSORI

Architettura Single Instruction Single Data

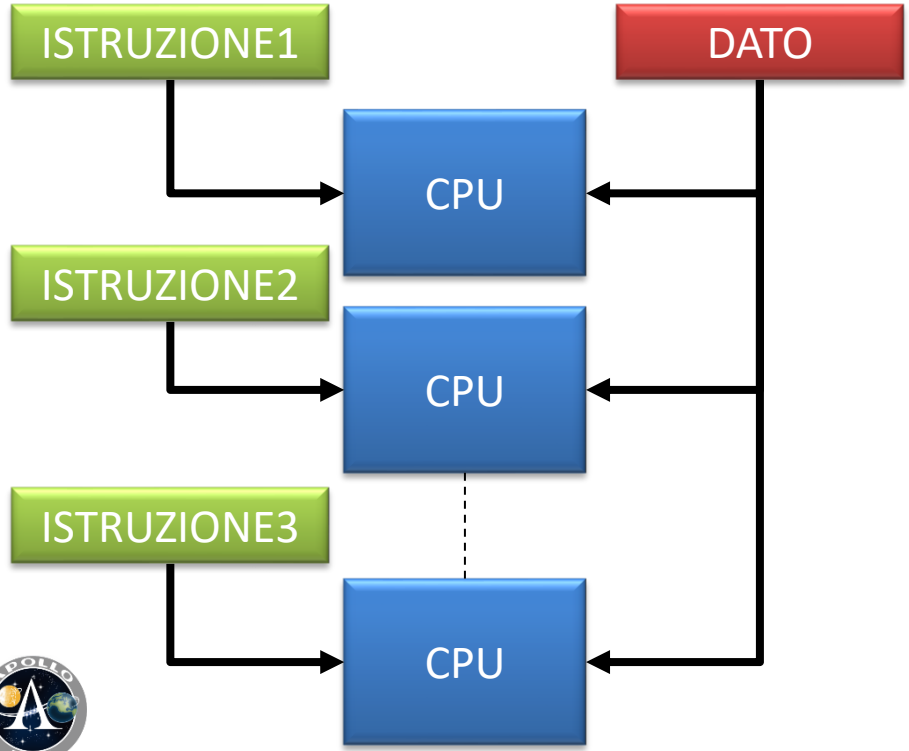
- Una **architettura SISD** è costituita da un solo processore, il quale riceve un unico flusso di istruzioni che opera su un solo flusso di dati
- Gli elaboratori fino alla fine degli anni ’80 hanno aderito a questo modello che corrisponde a quello proposto dal matematico e informatico John von Neumann negli anni ’40
- Gli **algoritmi** che sono eseguiti su questo tipo di architettura sono detti **sequenziali** (o seriali), in quanto non contengono alcun parallelismo
- Esempi di architetture SISD sono gli elaboratori, le workstation e i mainframe dotati di una singola CPU



MULTIPROCESSORI

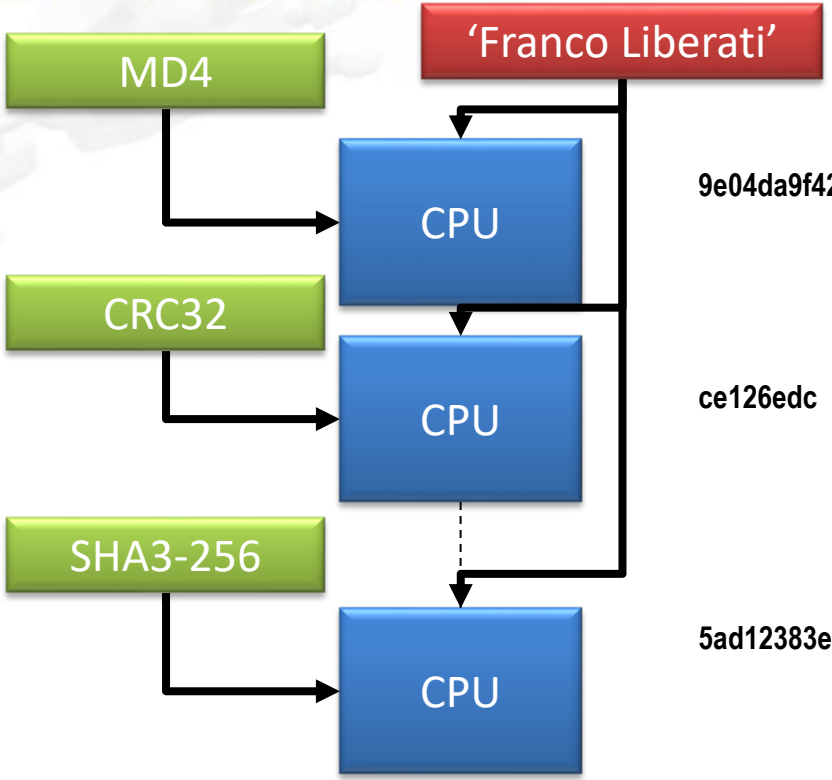
Architettura Multiple Instruction Single Data

- Nell'**architettura MISD** *n* processori, condividono la Memoria Centrale
- Ad ogni ciclo di clock, il dato ricevuto dalla memoria è elaborato da tutti i processori simultaneamente, ciascuno secondo le istruzioni ricevute nella propria Unità di Controllo
- Il **parallelismo a livello di istruzioni** si ottiene consentendo a differenti processori di elaborare diverse operazioni sullo stesso dato
- Questa classe di elaboratori eseguono operazioni diverse su input identici e per questo si usano nella classificazione dei problemi computazionali o nella crittografia dei dati
- Architetture usate su alcuni elaboratori del progetto Apollo (ma sono per lo più teoriche)



MULTIPROCESSORI

Architettura Multiple Instruction Single Data: esempio



9e04da9f42791b76a8d52ffab53f9040

ce126edc

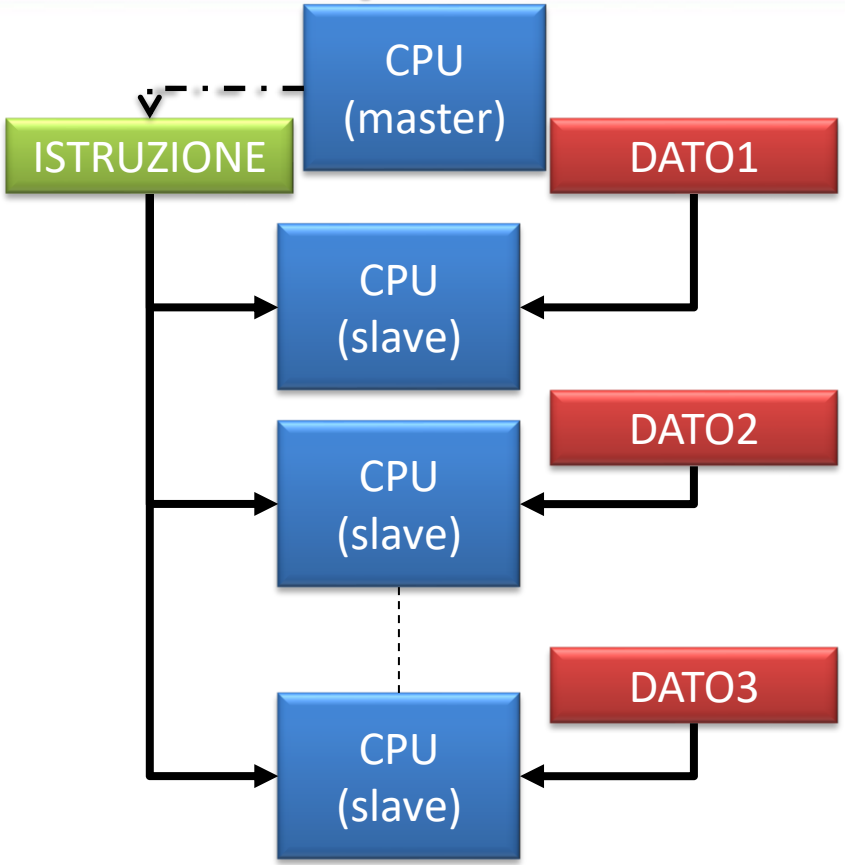
5ad12383ed19ec6e1414c2355cddf13e108ee034d476c62d1834b38d496f82d8

https://www.browserling.com/tools/all-hashes

MULTIPROCESSORI

Architettura Single Instruction Multiple Data

- Una **architettura SIMD** è composta da *n* **processori serventi** (*slave*), ognuno dotato di una propria memoria locale. Tutti i processori lavorano sotto il controllo di un flusso di istruzioni unico e rilasciato da un **processore padrone** (*master*). Inoltre sono presenti *n* flussi di dati, uno per ogni processore
- I processori funzionano contemporaneamente: ad ogni colpo di clock, tutti i processori eseguono la stessa istruzione ma su dati diversi. Questo 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 risolto contemporaneamente tramite lo stesso insieme di istruzioni
- Le architetture SIMD (tipica dei processori vettoriali) negli ultimi anni hanno preso piede nel campo dell'elaborazione grafica e delle applicazioni multimediali (es.: Intel MMX e MIPS-3D). Un esempio è l'APE 1000



MULTIPROCESSORI

SIMD:esempio di programma

- Il problema SOMMATORIA può essere così formulato:
  Problema: interi a[1], a[2], .., a[n]

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

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

- ALGORITMO in parallelo

  for (j=1; j≤log$_2$n;i++)
  {
  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**(log$_2$n)

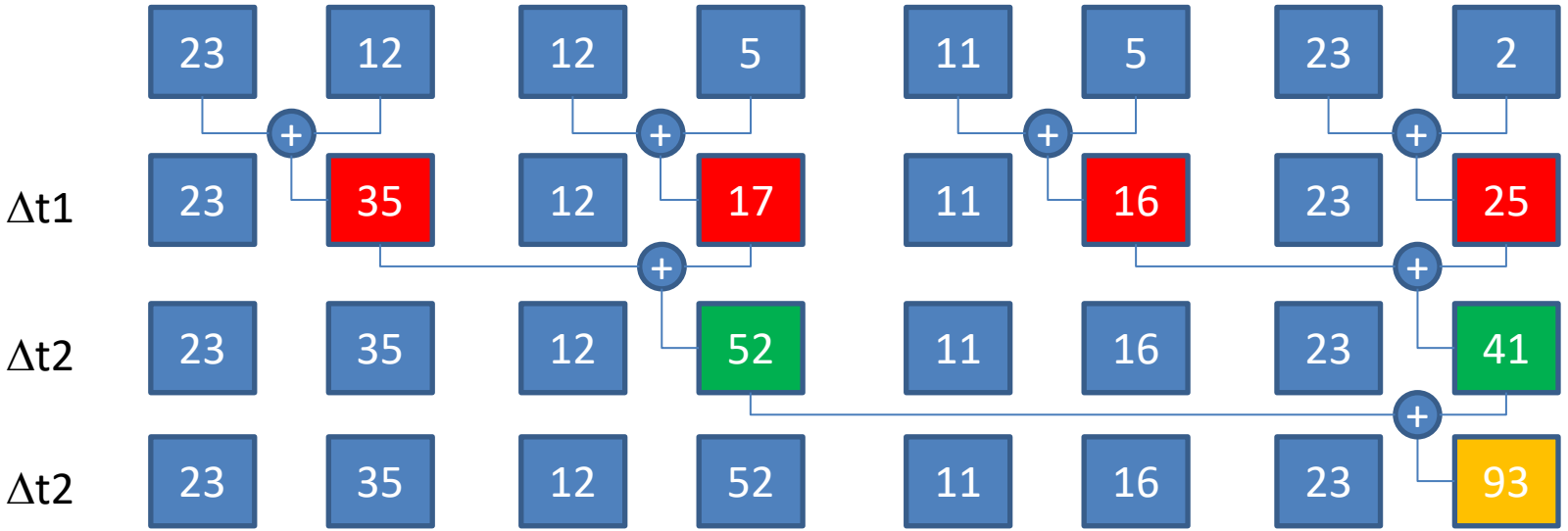
MULTIPROCESSORI

SIMD:esempio di programma

for (j=1; j≤log$_2$n;i++)
{

for all k processor (1 ≤k ≤n/2^j) do in parallel
{a[2^{j}k] =a[2^{j}k]+a[2^{j}k-2^{j-1}]}

}



MULTIPROCESSORI

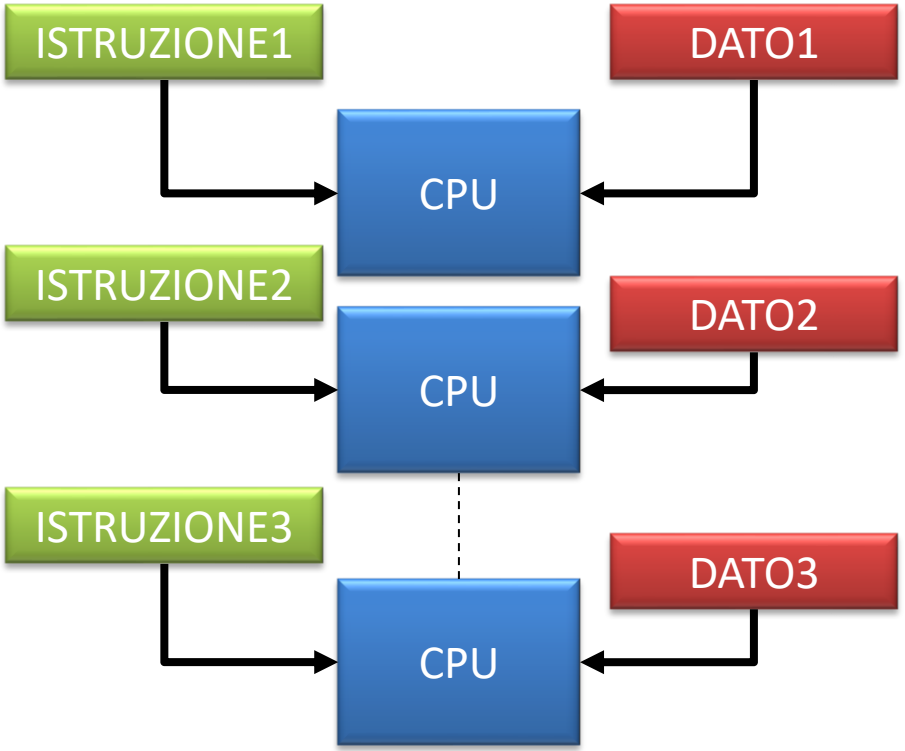
Architettura Single Instruction Multiple Data: P-RAM

- 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): affinché il processore k e il processore j si scambino un valore, è sufficiente 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

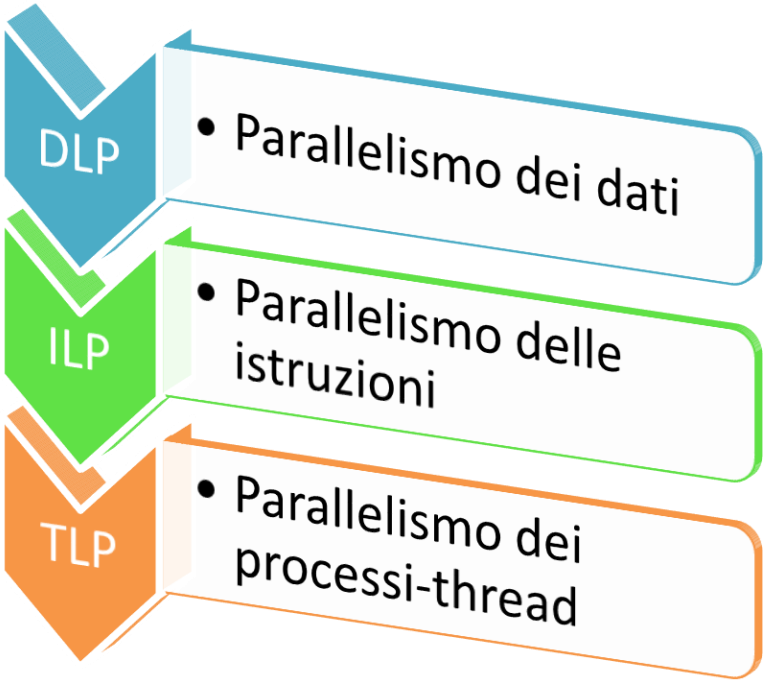
- L'**architettura MIMD** è la più generale e più potente nella classificazione di Flynn. Ci sono *n* processori, *n* flussi di istruzioni e *n* flussi di dati. Ogni processore possiede una propria unità di controllo e una propria memoria locale
- Ogni processore opera sotto il controllo di un flusso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



MULTIPROCESSORI

Classificazione di Almasi per Architettura Parallela

- 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
  - **Instruction Level Parallelism (ILP)**: le istruzioni, che compongono il programma, sono distribuite ed eseguite contemporaneamente da più processori
  - **Thread Level Parallelism (TLP)**: le applicazioni utilizzano thread/processi concorrenti, cioè thread/processi che sono eseguiti in parallelo da più processori



MULTIPROCESSORI

Organizzazione Memoria

- Nei multiprocessori si può avere una **memoria condivisa** o **distribuita**.

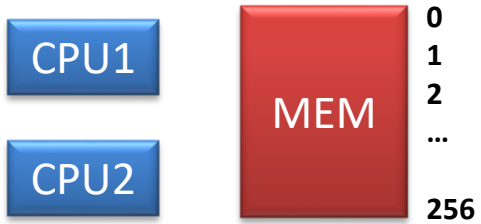
MULTIPROCESSORI

Organizzazione Memoria

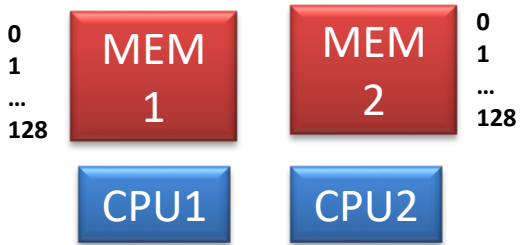
- La **differenza tra memoria condivisa e distribuita sta nella struttura della memoria virtuale**, ossia la memoria vista dalla prospettiva del processore
- Ciò che distingue una memoria condivisa da una memoria distribuita è come **il sottosistema di memoria interpreta un indirizzo generato da un processore**
- 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** di tale struttura dati. Una macchina a memoria distribuita, invece, deve creare **copie dei dati condivisi** in ciascuna memoria locale. Queste copie sono create inviando, da un processore all'altro, un messaggio contenente i dati da condividere
  - Un inconveniente della memoria distribuita è che a volte questi messaggi possono essere molto grandi e richiedere tempi di trasferimento lunghi

```
lw $a0,1000 #carica in $a0
            #l'operando contenuto
            #nella locazione 1000 di
            #memoria principale
```

Memoria condivisa: $a0 ha lo stesso valore



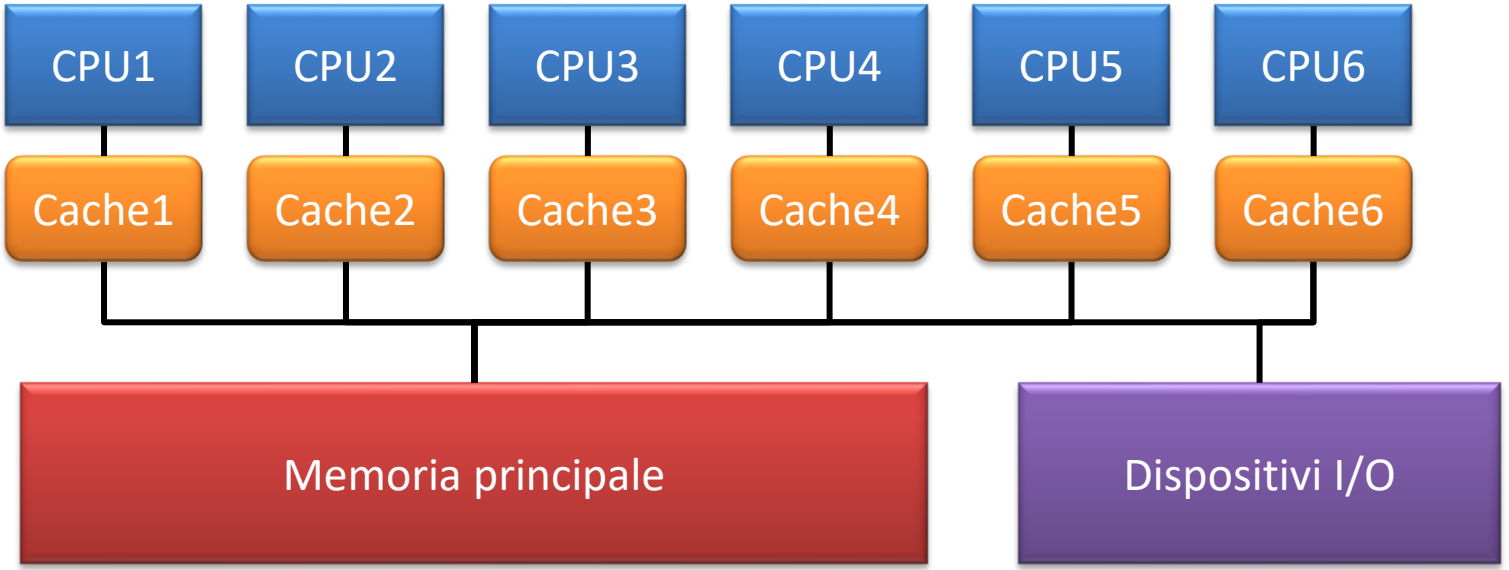
Memoria distribuita: $a0 ha valori diversi



MULTIPROCESSORI

Memoria Condivisa

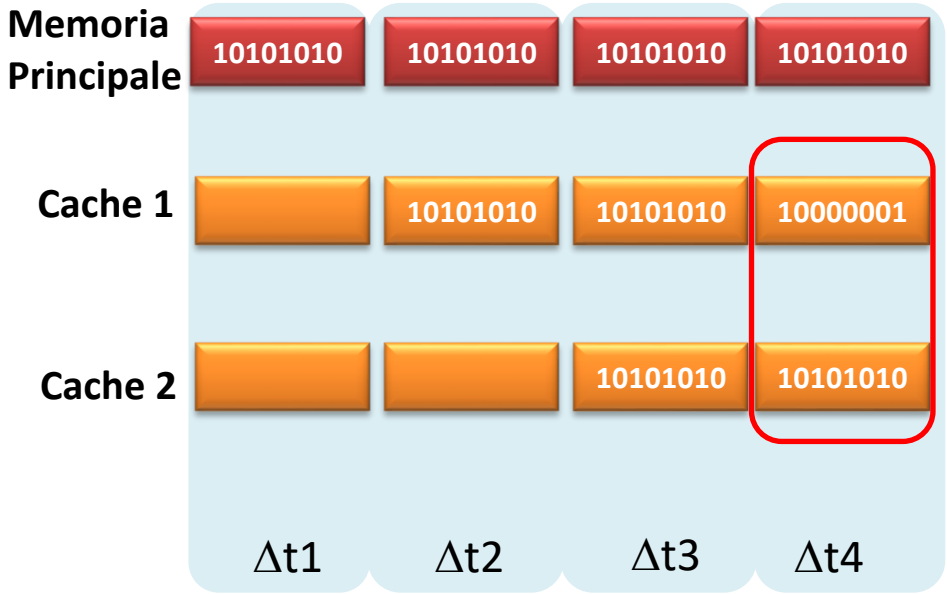
- Un modo semplice per collegare più processori insieme per costruire un multiprocessore a memoria condivisa è utilizzando un bus (**multiprocessore a bus unico**)
- Ad ogni processore può essere associata una memoria cache in quanto si presume che la probabilità che un processore necessiti di un dato 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 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, similarmente a quelli che si ha con i thread a livello di programmazione



MULTIPROCESSORI

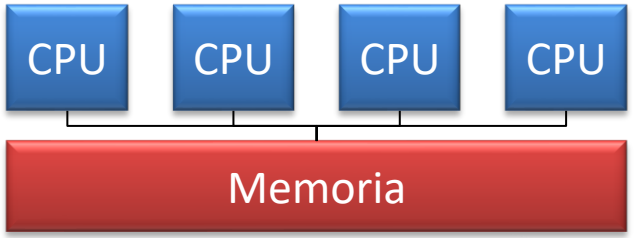
Memoria Condivisa: caratteristiche

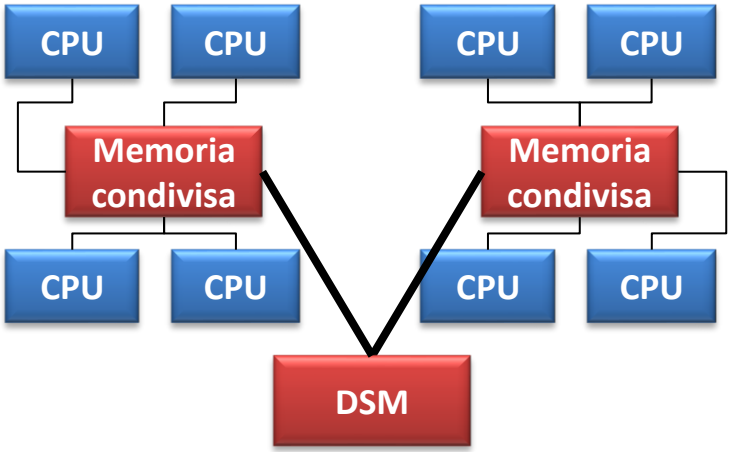
- Caratteristiche dei **sistemi multiprocessore a memoria condivisa**:
  - La **memoria logica è la stessa per tutti i processori**; ad esempio, tutti i processori associati alla stessa struttura dati lavoreranno con gli stessi indirizzi logici, in quanto globali, accedendo così alle stesse locazioni di memoria.
  - La **sincronizzazione** è ottenuta analizzando i compiti 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 compito mentre un altro compito 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 compiti è veloce;infatti il tempo necessario alle attività per comunicare tra loro è il tempo che una di esse 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 compiti che connessioni alla memoria. In queste situazioni si avranno dei processori in stato d'attesa e quindi tempi di latenza maggiori.

MULTIPROCESSORI

Memoria Condivisa: organizzazione

- **Classificazione memoria condivisa:**
  - **Uniform Memory Access**: 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**: 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)
    - Scalabile ma complessa da sviluppare

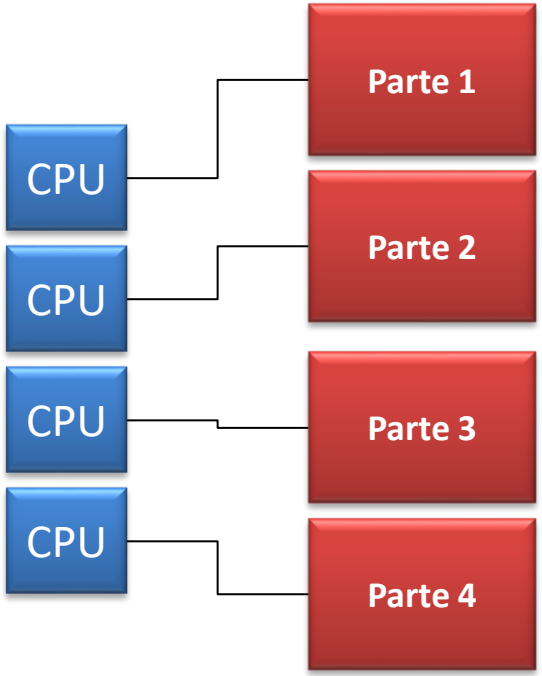




MULTIPROCESSORI

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 "**multicompute**r", 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

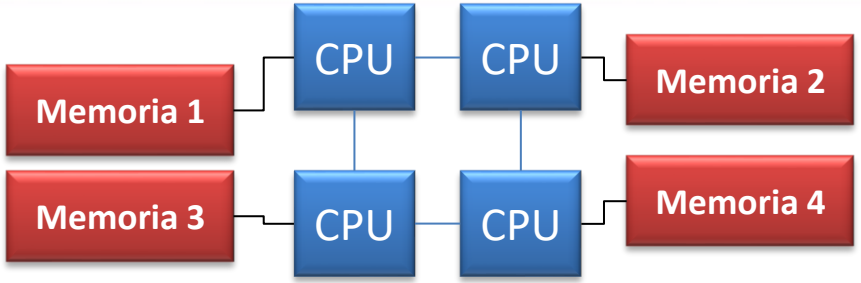
Memoria Distribuita: caratteristiche

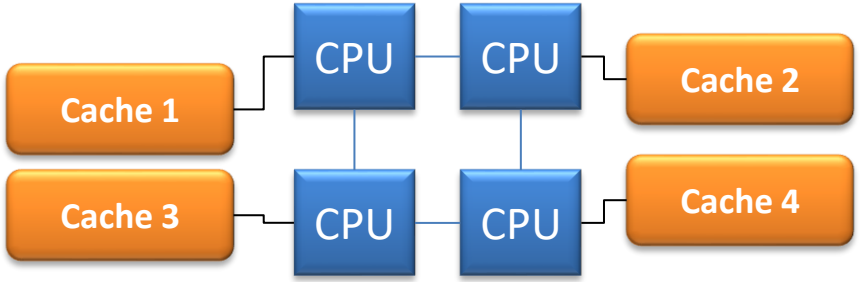
- Caratteristiche dei **sistemi multiprocessore a memoria distribuita**:
  - La memoria è fisicamente distribuita tra i vari processori; ogni memoria locale è accessibile direttamente solo dal suo processore
  - La **sincronizzazione è ottenuta mediante lo spostamento di dati** (un messaggio) tra i processori
    - 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: 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 mantenendo solo le memorie cache

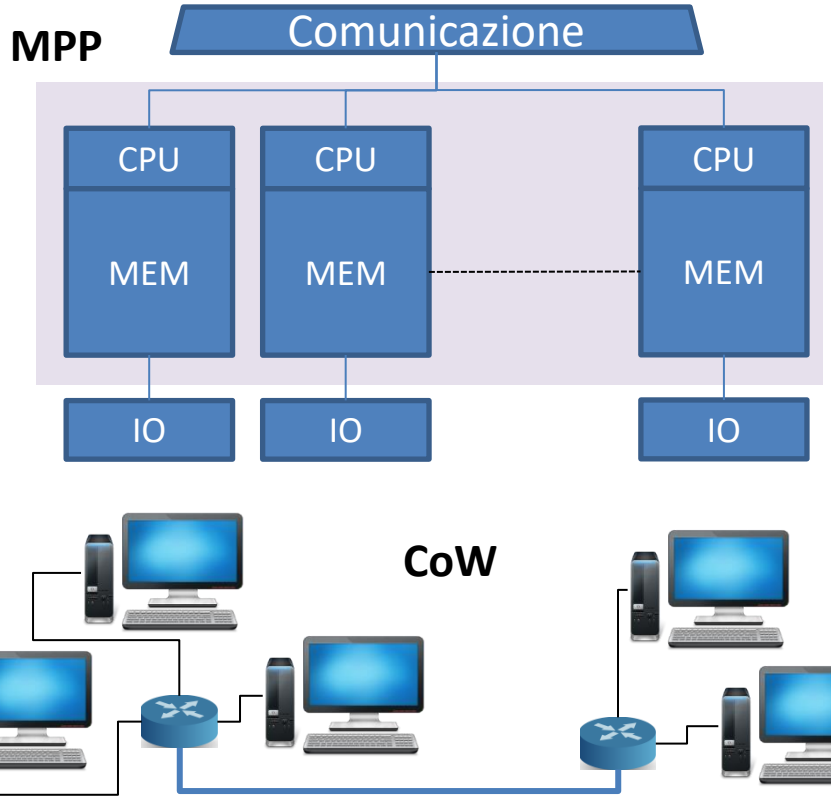




MULTIPROCESSORI

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 sono basate su queste architetture
  - **Cluster of Workstations** : le architetture CoW sono sistemi di elaborazione basati su calcolatori collegati da reti di comunicazione. I cluster di calcolo ricadono in questa classificazione



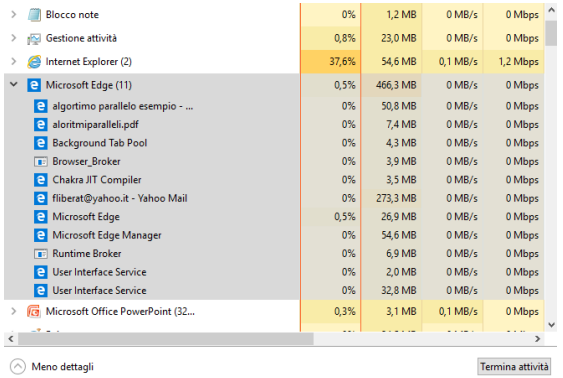
MULTIPROCESSORI

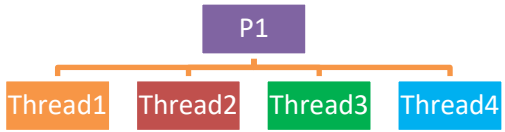
Parallelismo dei processi e dei thread

- Un **processo** (**task**),è un programma in memoria
- Un **thread** è una suddivisione di un processo in due o più sottoprocessi

Esempio

In un'applicazione di elaborazione di immagini, un thread è dedicato alla gestione dell'interfaccia utente, mentre un altro potrebbe gestisce l'applicazione di filtri complessi alle immagini. Questo permette all'utente di continuare a interagire con l'applicazione mentre i filtri sono applicati.

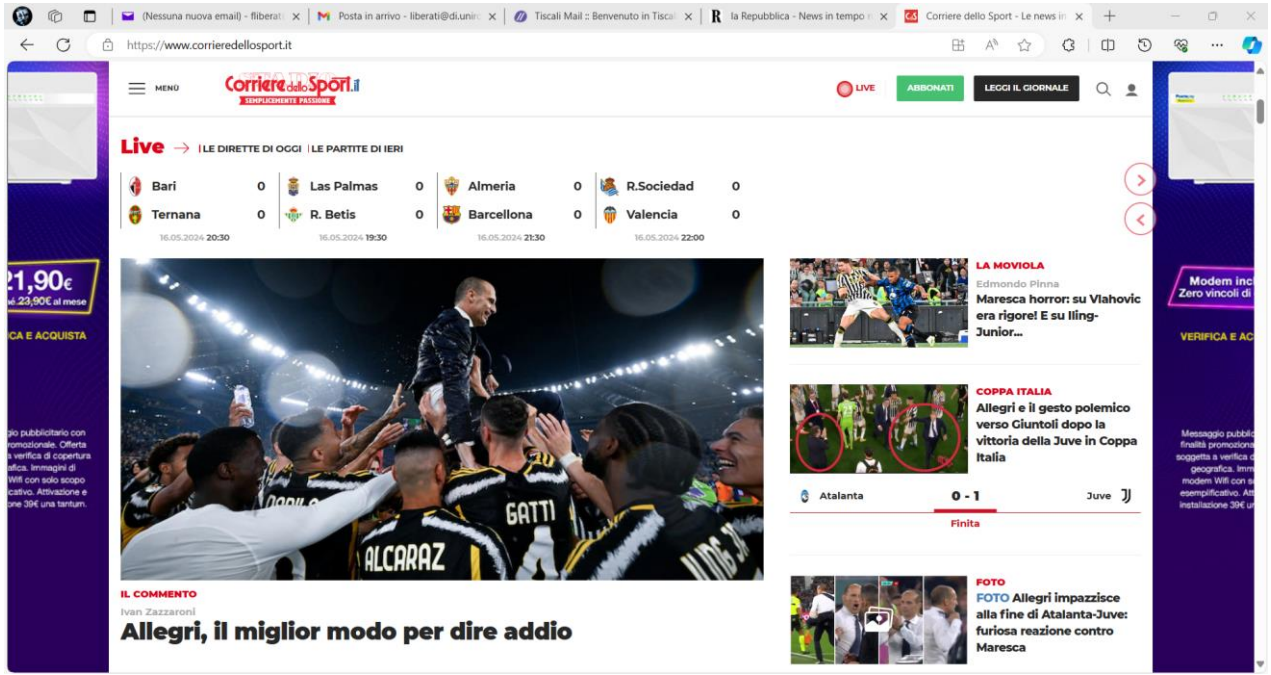




MULTIPROCESSORI

Parallelismo dei processi e dei thread

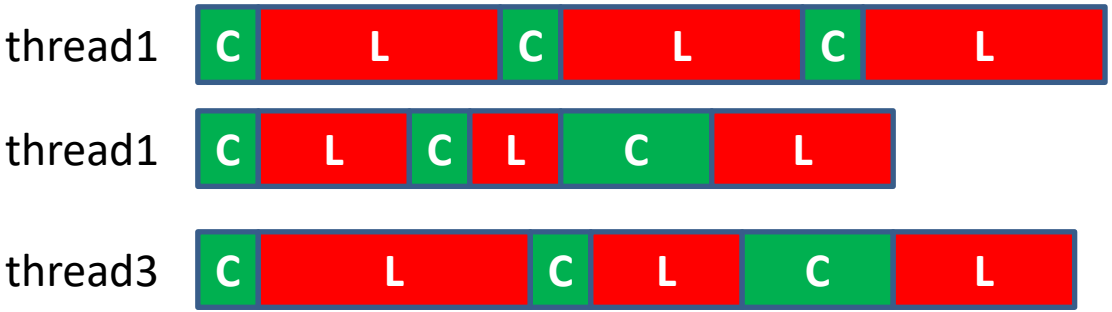
- I **thread** sono eseguiti in concorrenza da un sistema di elaborazione monoprocessore o parallelamente in un multiprocessore o multicore



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 migliore”*
- 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 thread a livello parallelo** (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 *reservation station*)
Execute	L'istruzione attende nella coda finché i suoi operandi di input non sono disponibili. L'istruzione lasciare 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 WB.

MULTIPROCESSORI

Parallelismo dei processi e dei thread

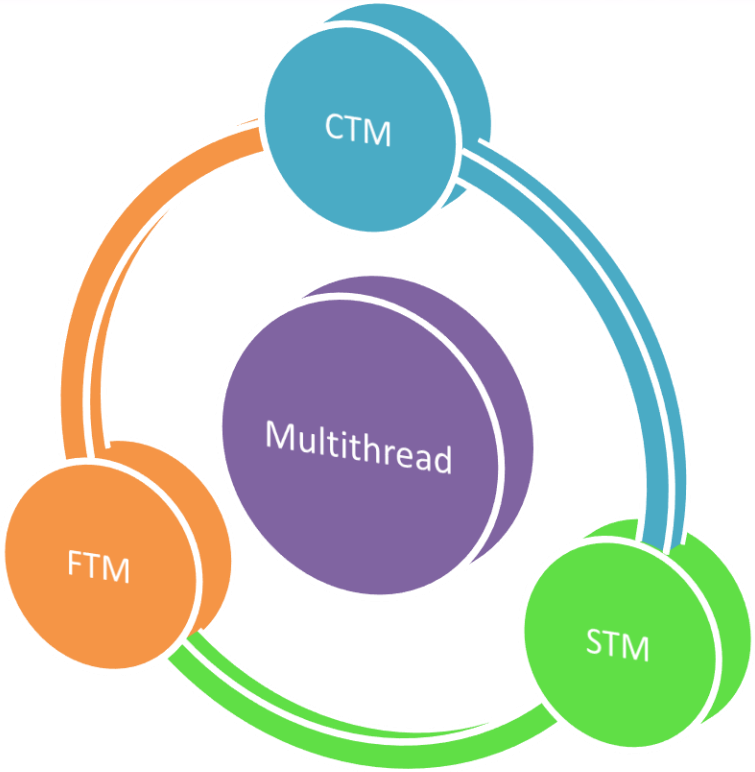
- Paradigma *Out of Order Processing*

```
X=A+B
Y=C+D
W=X+Y

lw $t0,A          lw $t1, B          #lettura a e lettura b
lw $t2,C          lw $t3, D          #lettura c e lettura d
add $t4,$t0,$t2   add $t5,$t3,$t1    #x=a+b e y=c+d
add $t7,$t4,$t5                      #x+y

CALCOLO SOLUZIONE AX^2+BX+C

lw $t0,C          lw $t1, A          #lettura a e lettura c
lw $t2,B          mul $t3,$t0,$t2    #lettura b e calcolo ac
Mul $t2,£t2,$t2   mul $t3,$t3,4      #calcolo b^2 e 4ac
sub $t3,$t1,$t3                      #calcolo b-4ac
mfc1 $f0,$t3                         #calcolo Delta non eseguibile in parallelo
cvt.s.d $f0,$f0                      #
sqrt.s $f0,$f0                       #

mfc1 $f9,$t0      mul $t1,$t1,-1     #spostamento a in coprocessore calcolo -b
mfc1 $f8,$t1                         #spostamento -b in coprocessore
cvt.s.d $f9,$f9   cvt.s.d $f8,$f8    #conversione a e -b
mul.s $f9,$f9,2                                  #calcolo 2a
sub.s $f5,$f8,$f0 add.s $f6,$f8,$f0  # -b-delta  -b+delta
div.s $f5,$f5,$f9 div.s $f6,$f6,$f9  #-b-delta/2a -b+delta/2a
```

MULTIPROCESSORI

Parallelismo dei processi e dei thread

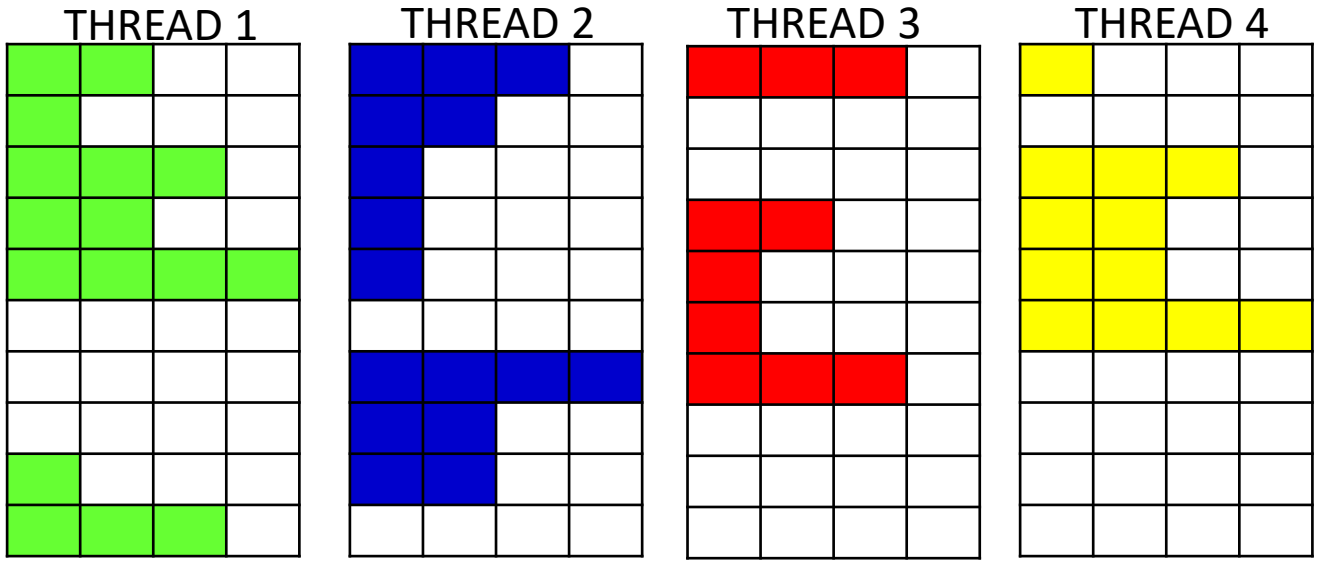
- I tre tipi principali di hardware multithreading sono:
  - **il Fine-Grained Multithreading**
  - **il Coarse-Grained Multithreadinge**
  - **il Simultaneous Multithreading**



MULTIPROCESSORI

Parallelismo dei processi e dei thread

- Ogni quadrante colorato indica l'esecuzione di un'istruzione, mentre ogni quadrante bianco rappresenta uno stato di inattività del processore
- 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.



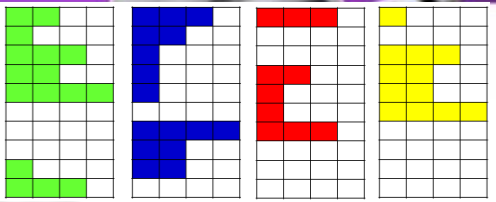
MULTIPROCESSORI

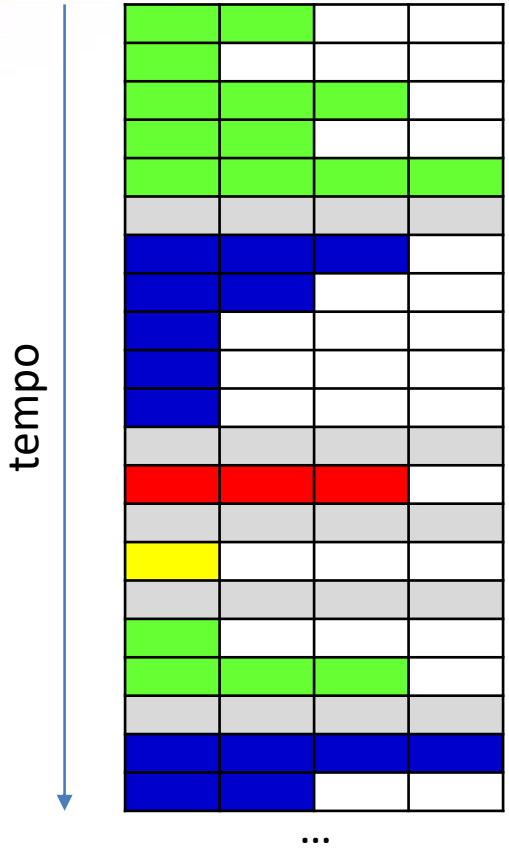
Multithread: Coars Grained MultiThreading (CMT)

- 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 valida 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, appena rileva che un thread è in stato di **stallo prolungati**, 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 totalmente inefficace nel caso in cui gli stalli siano molto piccoli. in tal caso il tempo necessario per salvare lo stato della pipeline 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





Per stalli molto lunghi si passa al thread successivo

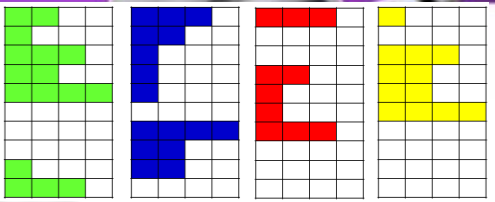
MULTIPROCESSORI

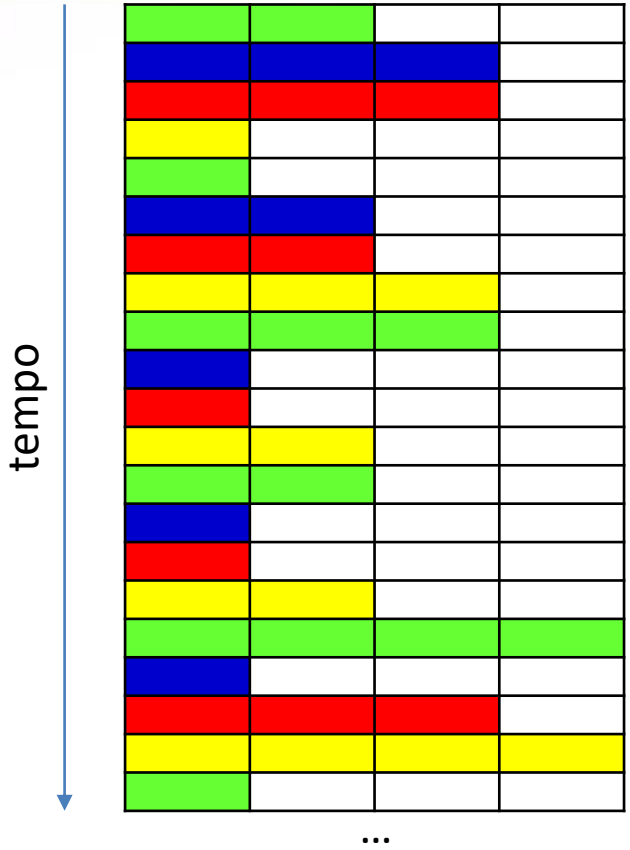
Multithread: Fine Grained MultiThreading (FMT)

- Il **multithreading a grana fine** consente l'alternanza di thread ad ogni ciclo di clock
- Questa alternanza è spesso creata usando l'**algoritmo round-robin** (il processore viene assegnato ai thread in relazione all'ordine d'arrivo nella Ready List; è presente un tempo limite in cui il thread può usare il processore saltando qualsiasi thread si trovi in uno stato di stallo nel momento in cui dovrebbe essere eseguito: scheduling FIFO (First In First Out) + timeslice)
- 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





FIFO+timeslice

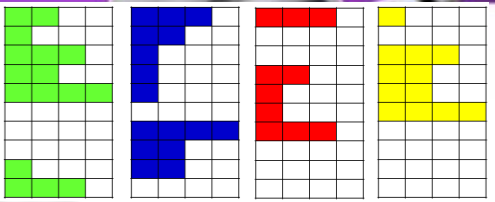
MULTIPROCESSORI

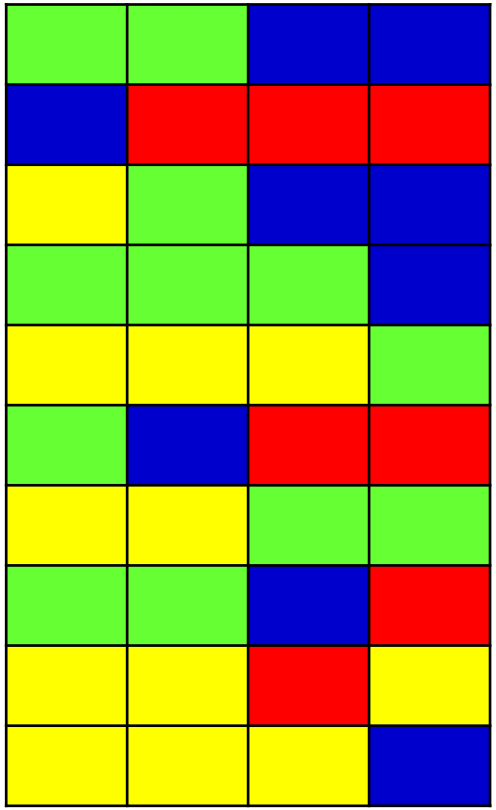
Multithread: Simultaneous MultiThreading (SMT)

- Il **mutithread 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 però 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 dell'elaboratore 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

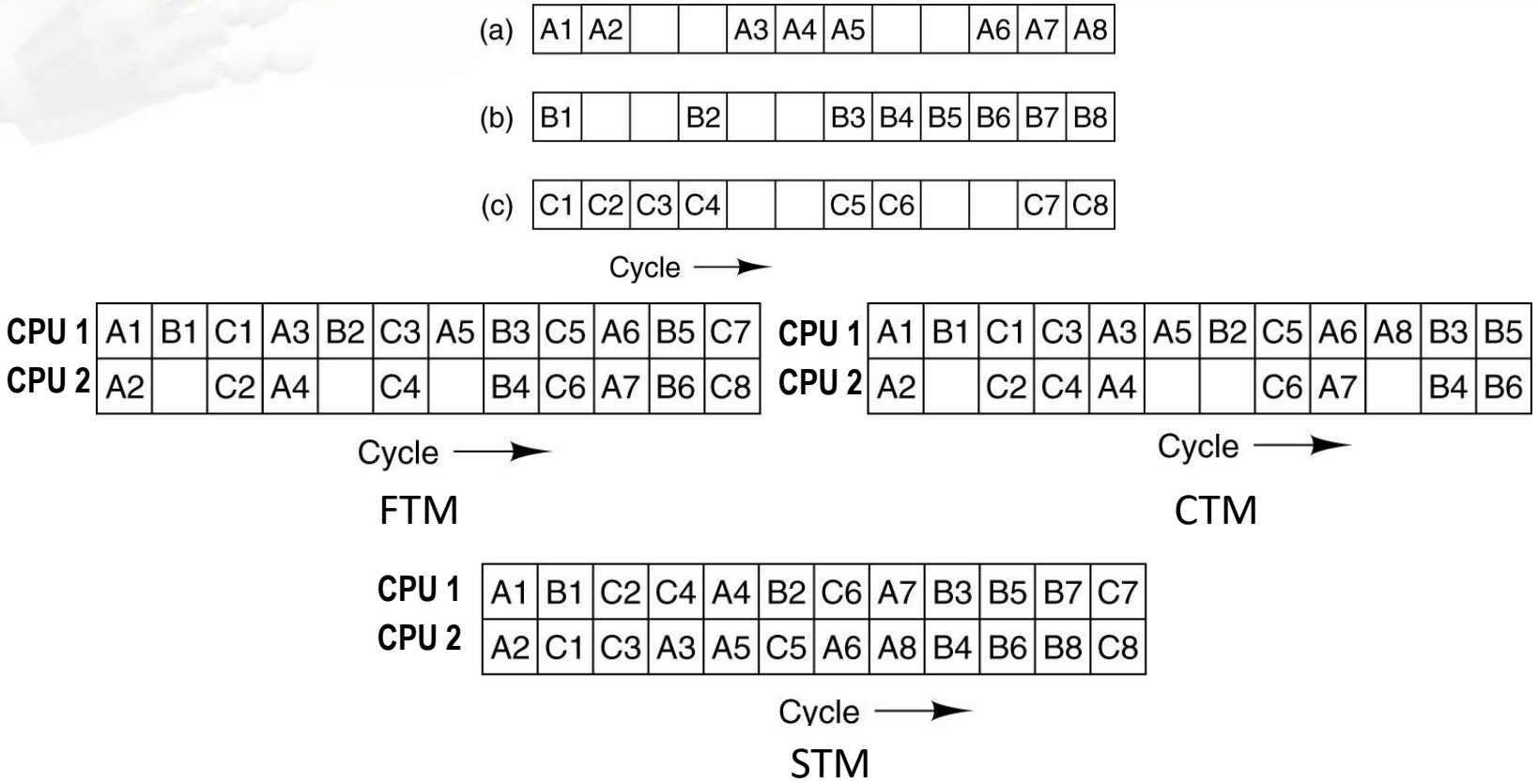




...

MULTIPROCESSORI

Parallelismo dei processi e dei thread FTM, CTM,STM




Fine