CALCOLATORI ELETTRONICI - DOMANDE FREQUENTI

1. Che cosa si intende per interruzione. Come vengono gestite le interruzioni multiple?

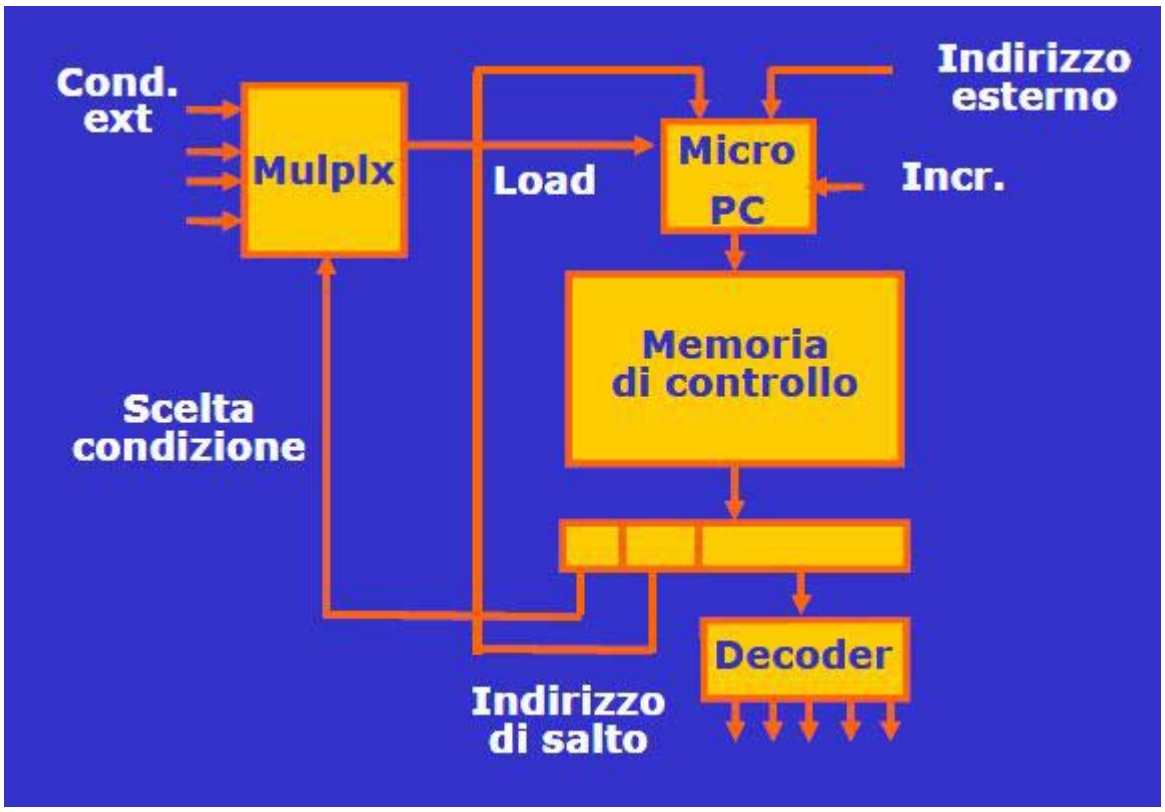
Le interruzioni sono un evento infrequente ed eccezionale, generato internamente o esternamente. L'interruzione è utilizzata per la gestione degli apparati di I/O, per ottenere risposte rapide nella gestione di dischi, tastiere, modem, etc. Questi elementi devono richiamare l'attenzione della CPU molto rapidamente e il trasferimento dei dati da e verso questi organi deve avvenire nel modo più efficiente possibile. Quindi è necessario assegnare urgenze o priorità diverse dalle varie richieste di intervento se contemporanee. Le interruzioni possono essere "interne", quando provengono dall'interno della CPU in condizione di errori o malfunzionamento hardware, o "esterne", cioè la gestione delle I/O mediante interruzioni. La gestione delle interruzioni multiple avviene utilizzando criteri di priorità e il meccanismo di annidamento. Gli indirizzi di ritorno e i relativi contesti possono essere immagazzinati nello stack. Però presenta alcuni problemi: se è in fase di esecuzione un attività con caratteristiche stringenti di tempi di sevizio, l'interruzione di tale attività è impossibile.

2. Descrivere la tecnica set-associative per l'accesso ai dati di una cache e illustrarne i vantaggi.

Le linee nella cache sono suddivise in gruppi, dove ogni gruppo può ospitare linee della memoria principale individuate secondo la tecnica di accesso diretto. All'interno di ogni gruppo la linea viene individuata in modo associativo. Questa tecnica insieme alle altre serve per trovare il dato richiesto più rapidamente nella memoria cache, questo perché la CPU può modificare un dato nella cache e noi non sappiamo come viene aggiornata la memoria principale.

3. Unità di controllo microprogrammata.

Ogni istruzione (CPU) è realizzata da una sequenza di microistruzioni. Ogni microistruzione implementa una o più microoperazioni concorrenti. Ogni microistruzione attiva le microoperazioni con un insieme di linee di controllo. La sequenza di controllo è immagazzinata in una memoria di controllo (CM – control memory). La memoria di controllo può essere RAM o ROM. Il costo maggiore di una unità microprogrammata è rappresentata dalla memoria di controllo. Il costo dipende da due parametri: W la lunghezza della microistruzione e N numero di microistruzioni. La minimizzazione di N è associata all'algoritmo, al parallelismo delle operazioni, etc; la minimizzazione di W è associata al livello possibile di codifica.

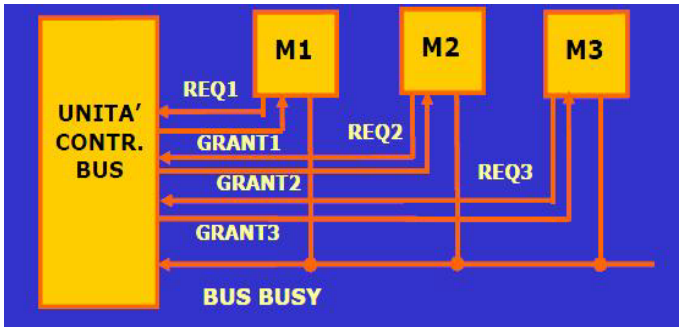


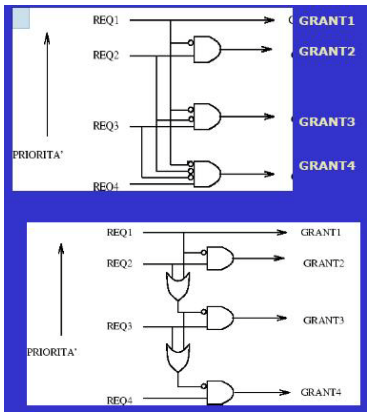
4. Arbitri: spiegare le funzioni svolte e i possibili modi di implementazione (concentrato o distribuito).

Se più master sono collegati al bus, una fase di arbitrazione deve decidere quale unità avrà il controllo:

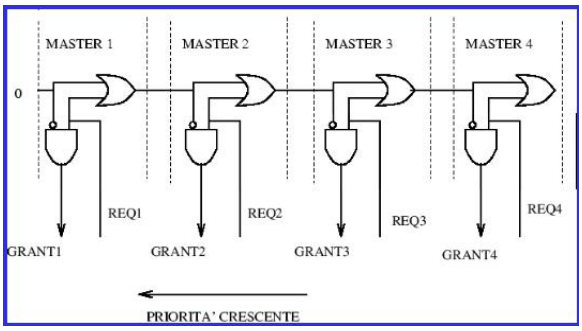
- Arbitro centralizzato: ha la possibilità di modificare agevolmente la politica di assegnazione ed ha una maggiore osservabilità.
- Arbitro distribuito: ha una maggiore modularità e possibilità di espansione e riconfigurazione e ha una migliore tolleranza ai guasti.

Nell'arbitro concentrato, la politica di assegnazione delle risorse è definita all'interno dell'allocatore.



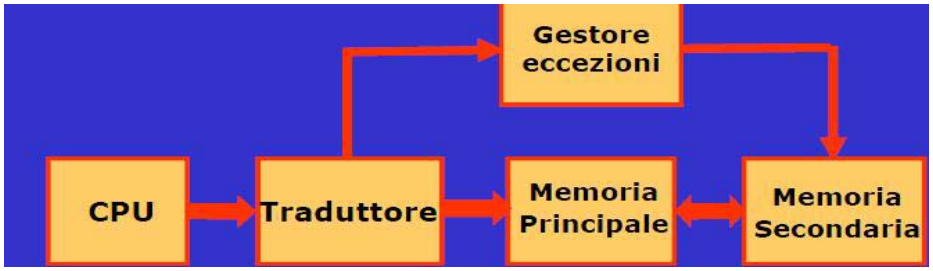


Il circuito dell'arbitro distribuito è uguale a quello concentrato, varia la distribuzione delle parti hardware.

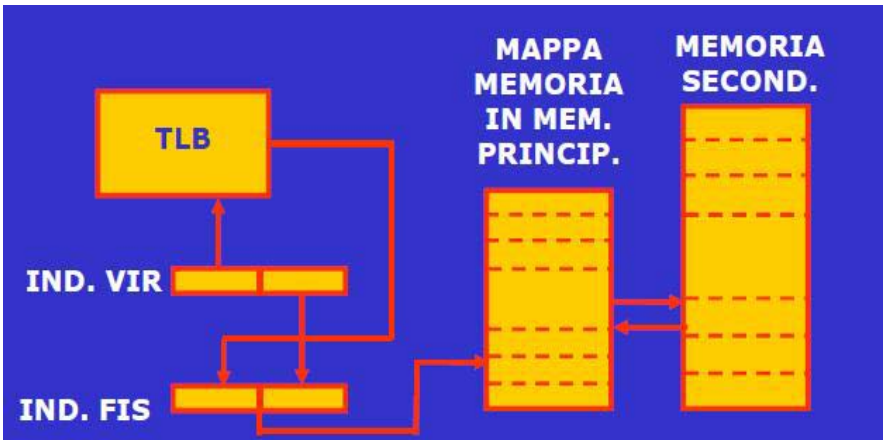


5. La memoria virtuale (ragioni e meccanismi).

Il meccanismo di gestione delle gerarchia formata da Memoria principale e Memoria secondaria prende il nome di Memoria Virtuale. Ogni attività eseguita dal processore vede uno spazio di indirizzi virtuale o logico; la memoria è indirizzata tramite indirizzi fisici. Esiste una corrispondenza (allocazione) tra indirizzo logico e indirizzo fisico. L'allocazione è una funzione f che associa a un indirizzo logico un indirizzo fisico.
Questo libera il programmatore dalla necessità di occuparsi direttamente dello spazio della memoria fisica e facilita la gestione comune della memoria tra più utenti (task).
La memoria virtuale rende il programma indipendente dalle dimensioni della singola macchina e sfrutta efficacemente la gerarchia.



Indirizzo virtuale è l'indirizzo di blocco virtuale da cui si estrae l'indirizzo fisico di blocco.
Lo spiazzamento è usato direttamente come offset all'interno del blocco fisico.



TLB è una memoria temporanea che effettua velocemente la traduzione dall'indirizzo virtuale all'indirizzo fisico.

6. Le topologie e le caratteristiche delle strutture di interconnessione dei sistemi paralleli.

La struttura di interconnessione è un elemento essenziale in un sistema di elaborazione distribuito e spesso condiziona le prestazioni complessive in modo rilevante. A questo livello di astrazione un sistema distribuito è formato da nodi (di elaborazione) e da rami per il trasporto dell'informazione.
Le caratteristiche principali sono :

- La topologia di rete
- La modularità e riconfigurabilità
- Le prestazioni in termini di banda passante
- La latenza

Le prestazioni della rete di interconnessione sono definite dall'interazione di alcune scelte progettuali, che sono:

- Topologia, cioè la struttura fisica del grafo di interconnessione della rete;
- Algoritmo di instradamento, che definisce i percorsi che i messaggi possono seguire tramite determinati algoritmi;
- Strategie di commutazione, cioè come i dati attraversano un percorso o un nodo;
- Controllo di flusso, cioè cosa succede quando i dati attraversano un nodo.

Le reti di interconnessione possono essere "statiche", la topologia di interconnessione formata da archi punto-punto non cambia durante l'esecuzione di un programma, o "dinamiche", canali commutati consentono alla struttura di comunicazione di adattarsi alle esigenze specifiche del programma.

Le Caratteristiche

- Grado del Nodo: numero di rami che fanno capo al singolo nodo, associazione al costo e alla modularità del sistema; da lui dipende il suo costo.
- Diametro della Rete: il massimo dei cammini minimi tra due nodi della rete, associato alla latenza del processo di comunicazione; esso influisce sul ritardo massimo presentato dalle informazioni che circolano sulla rete.

Ci sono diverse topologie:

- Array lineare: Grado = 2, Diametro = N-1
- Anello circolare: Grado = 2, Diametro = N (comunicazione monodirezionale) o N/2 (comunicazione bidirezionale)
- Albero: Grado = 2, Diametro da calcolare
- Ipercubo: Grado = Diametro = log2(N)

7. Critiche al parallelismo: la legge di Amdhal

Considera gli effetti di operazioni sequenziali su di un sistema di elaborazione parallelo. Se P è il numero di processori, T(P) il tempo di esecuzione di un programma a parallelismo P, f la frazione di programma che deve essere eseguita sequenzialmente, il tempo di esecuzione del programma su di una architettura a parallelismo P è:

$T(P) = fT(1) + (1-f)[T(1)/P]$

Speed Up:

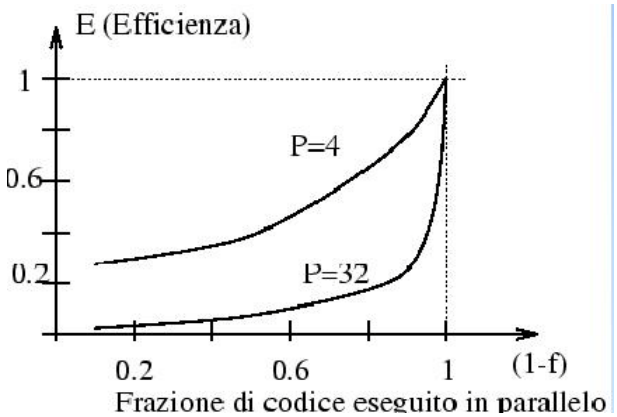
$S(P) = T(1)/T(P) = 1/[f + (1-f)/P] = P/[1 + f(P-1)]$

Efficienza:

$E(P) = S(P)/P = 1/[1 + f(P-1)]$

Le conseguenze sono:

Una piccola porzione di codice non parallelizzabile può avere un grande effetto globale. Riuscire a parallelizzare una porzione di codice sequenziale consente di migliorare significativamente le prestazioni.



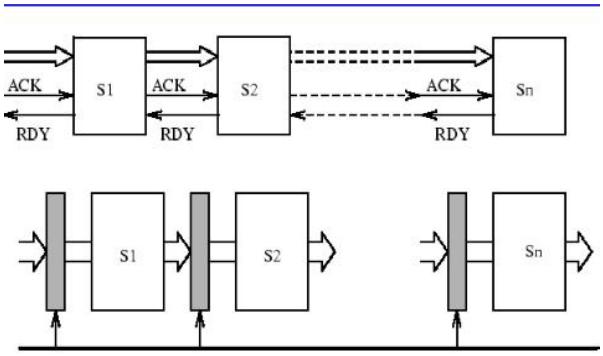
8. Indicare le principali caratteristiche delle architetture pipeline e i vantaggi ottenuti dalla loro introduzione.

L'architettura a pipeline è la più significativa soluzione architetturale che consente di aumentare la velocità di una CPU. Riprende concetti ampiamente applicati in altri settori, come ad esempio una catena di montaggio industriale: la catena di montaggio consente di aumentare la frequenza con la quale vengono ultimati i prodotti (throughput), mantiene inalterato il tempo di realizzazione del prodotto (latenza) e mantiene inalterata la velocità degli operatori (clock).

Il modello della pipeline è diviso in cinque stati:

- IF = fetch o prelievo istruzione
- ID = decodifica e lettura registri
- EX = esecuzione
- ME = accesso alla memoria
- WB = scrittura registro destinazione

Esistono due strutture di pipeline: asincrona e sincrona. Sono così fatte:



Una pipeline a k stadi elabora n istruzione in: $T_k = [k + (n-1)]\tau$

Con uno speed-up di: $S_k = T_1/T_k = nk\tau / [k\tau + (n-1)\tau] = nk/(k + n - 1)$

I vantaggi sono che, rispetto all'esecuzione in sequenza, le prestazioni sono k volte superiori purché gli stadi delle pipeline siano sempre tutti attivi. Non è possibile mantenere la pipeline sempre piena! Facciamo un riassunto dei vantaggi e svantaggi:

Vantaggi :

- Istruzioni con la stessa temporizzazione
- Numero limitato di formati per le istruzioni
- Architettura di tipo load/store

Svantaggi:

- Necessità di unità funzionali condivise
- Variazioni nell'ordine di esecuzione delle istruzioni
- Dipendenza dalle istruzioni dei dati

9. Descrivere i meccanismi di funzionamento di una memoria cache

La cache è una memoria veloce e di piccole dimensioni posta fra la CPU e la memoria principale. Essa forma, insieme alla memoria principale, una gerarchia di memoria. Le prestazioni di una memoria cache dipendono anche dalla sua posizione rispetto alla CPU (su scheda o su chip). La presenza di una memoria cache sullo stesso chip del processore rappresenta la soluzione che garantisce la migliore efficienza. Vediamo le operazioni:

- La cache confronta la parte rilevante dell'indirizzo
- Se vi è {hit} viene completato il ciclo
- In caso di {miss}:
  1 – la cache inizia la lettura da memoria principale di dati che comprendono quello richiesto;
  2 – la politica di sostituzione ha gli stessi problemi di quella preemptive della memoria virtuale.

10. Classificazione delle architetture parallele e distribuite

È possibile classificare le architetture distribuite secondo diversi criteri:

- Classificazione di Enslow
- Classificazione di Flynn

La classificazione di Enslow vuole individuare le caratteristiche che deve avere un'architettura per dirsi distribuita. La classificazione di Enslow colloca le architetture in uno spazio tridimensionale: “un sistema è completamente distribuito se lo è nelle tre dimensioni”.

Essa si può vedere come:

DISTRIBUZIONE DELLA ELABORAZIONE

- Unità di elaborazione singola
- Unità di elaborazione singola ma con più unità funzionali
- Unità di elaborazione multiple omogenee o eterogenee

DISTRIBUZIONE DATI

- Base di dati centralizzata
- Base di dati distribuita fisicamente con direttorio centralizzato
- Base di dati distribuita fisicamente senza direttorio centralizzato

CONTROLLO

- Punto di controllo unico
- Relazioni di tipo master/slave statiche o dinamiche
- Punto di controllo autonomi e/o cooperanti

Secondo la classificazione di Flynn un sistema di elaborazione opera su un flusso di dati in base a un flusso di istruzioni acquisite in memoria. È possibile una classificazione basata sul grado di parallelismo dei due flussi.

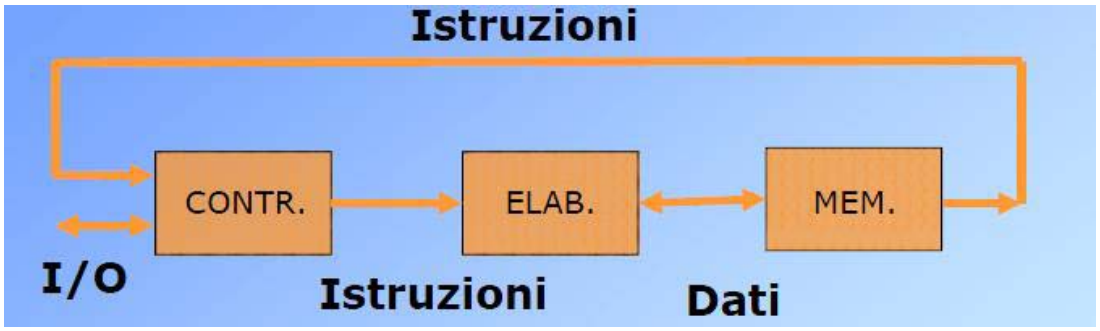
- SISD : singolo flusso di istruzioni – singolo flusso di dati
- SIMD: singolo flusso di istruzioni – flusso di dati multiplo
- MISD: flusso di istruzioni multiplo – singolo flusso di dati
- MIMD: flusso di istruzioni multiplo – flusso di dati multiplo

11. Classificazione di Flynn per le architetture parallele (Riportare lo schema di massima delle singole architetture).

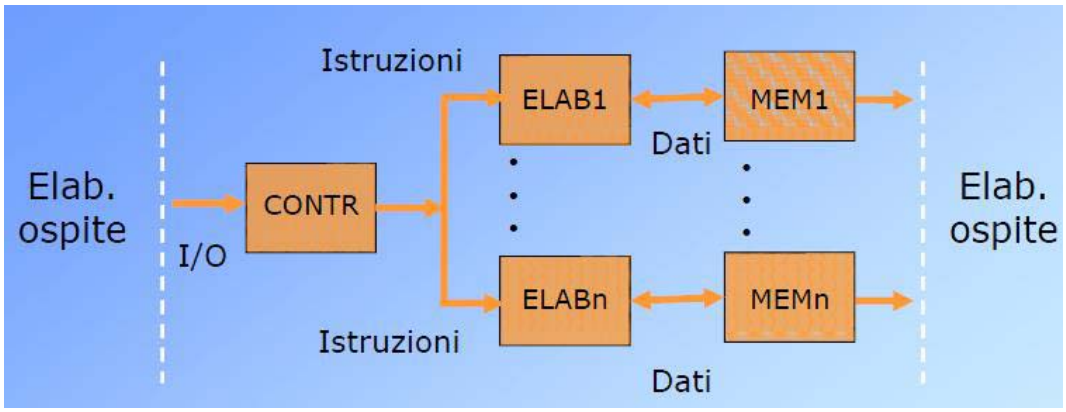
Secondo la classificazione di Flynn un sistema di elaborazione opera su un flusso di dati in base a un flusso di istruzioni acquisite in memoria. È possibile una classificazione basata sul grado di parallelismo dei due flussi.

- SISD : singolo flusso di istruzioni – singolo flusso di dati
- SIMD: singolo flusso di istruzioni – flusso di dati multiplo
- MISD: flusso di istruzioni multiplo – singolo flusso di dati
- MIMD: flusso di istruzioni multiplo – flusso di dati multiplo

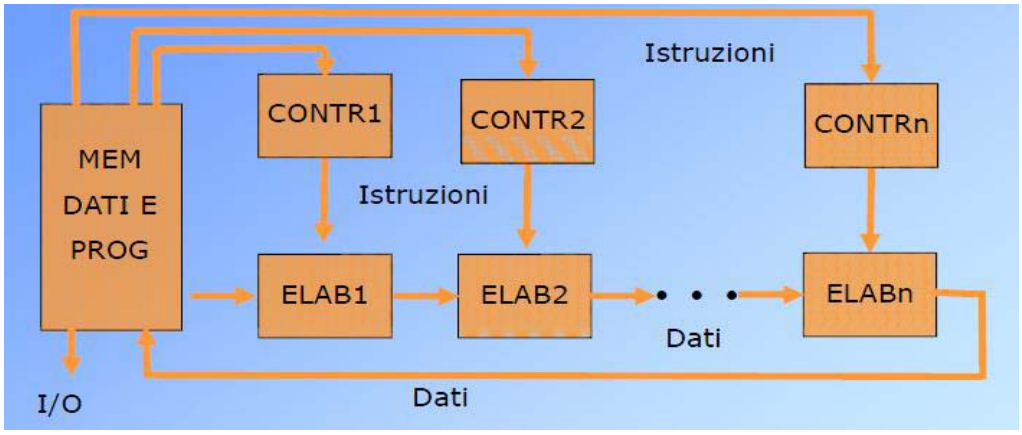
SISD: architettura tradizionale con singola CPU che elabora una istruzione alla volta operando su un dato alla volta.



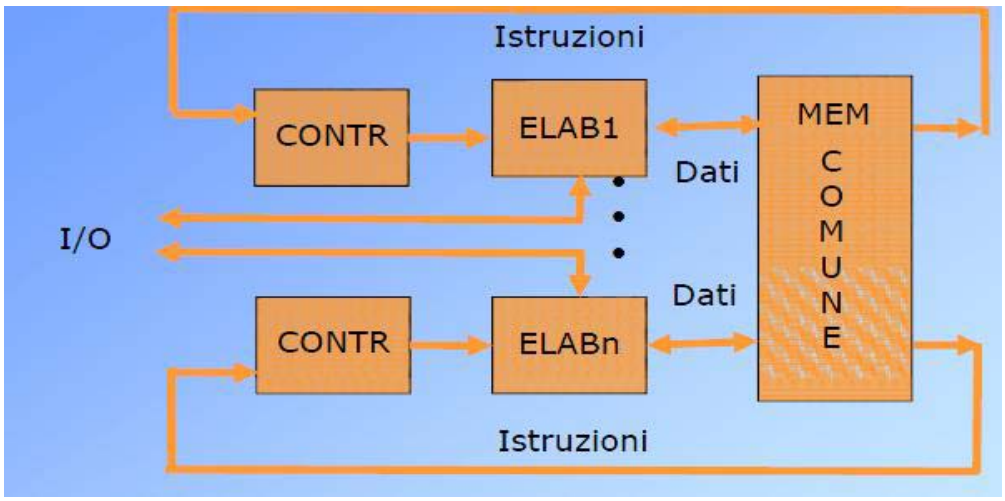
SIMD: più CPU operano in modo sincrono eseguendo la stessa istruzione su dati diversi.



MISD: il medesimo flusso di dati viene elaborato da un insieme di processori che eseguono istruzioni diverse.



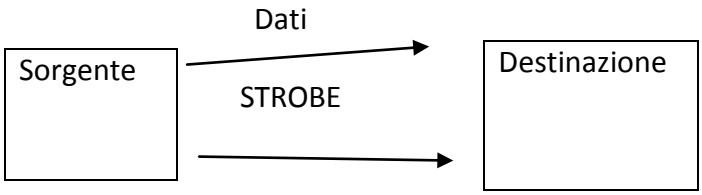
MIMD: unità di elaborazione diverse eseguono istruzioni diverse su dati diversi.



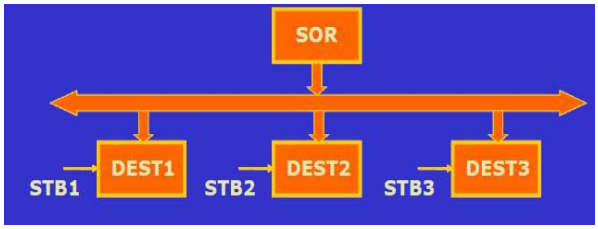
12. Sistemi a bus. Illustrare i protocolli per il trasferimento dei dati e le operazioni di lettura e scrittura.

Lo spostamento dei dati tra moduli e l'acquisizione di dati dall'esterno è uno degli aspetti più critici di un sistema di elaborazione. Ha grande influenza su prestazioni, costi, modularità e standardizzazione. Esistono vari protocolli di trasferimento dati:

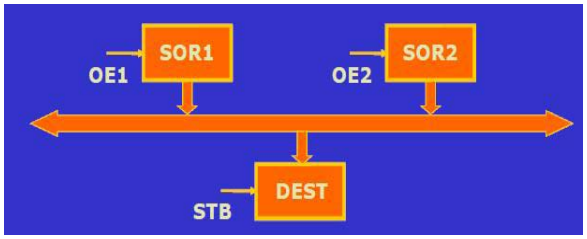
- PUNTO A PUNTO
  Il segnale STROBE comanda la memorizzazione del dato. Il trasferimento punto a punto richiede un solo segnale di STROBE per segnalare all'elemento ricevente la presenza del dato corretto.



- A DIFFUSIONE: Uno stesso dato invia dati a più destinatari



- A COLLETTORI: Più sorgenti possono inviare ad un unico destinatario



Il canale di comunicazione viene utilizzato per un solo trasferimento alla volta. Quando due elementi sorgente tentano di inviare dati contemporaneamente si ha un conflitto di accesso. Combinando i due meccanismi si ha un sistema in cui più moduli possono trasferirsi dati utilizzando una sola struttura di interconnessione: BUS.
I trasferimenti sono gestiti dalla combinazione dei segnali abilitazione (OE) e acquisizione (STB).

- Lettura: trasferimento dati da Slave a Matser
- Scrittura: trasferimento dati da Master a Slave

È bene ricordare che in ogni caso è sempre il modulo MASTER che attiva l'operazione e sceglie il modulo slave.

13. Architetture RISC.

RISC, cioè Reduced Instruction Set Computer, è un'architettura in cui tutte le istruzioni sono della stessa lunghezza, le operazioni da svolgere sono semplici e sono svolte tutte nello stesso numero di cicli. Le operazioni sono svolte su dati nei registri, le operazioni sui dati in memoria sono: Load e Store.
L'unità di controllo è semplice, è disponibile un grande numero di registri ed è disponibile una finestra che limita il numero dei registri visibili e migliora le chiamate a sottoprogramma.
Vediamo le ragioni del RISC:

- CARATTERISTICHE ARCHITETTURA ESTERNA
  Ha poche istruzioni e pochi modi di indirizzamento; accesso alla memoria limitato alle istruzioni di LOAD e di STORE e le operazioni logiche e aritmetiche svolte avendo come operandi solo registri interni.
- CARATTERISTICHE ARCHITETTURALI INTERNE
  Esecuzione veloce delle istruzioni; architettura interna organizzata a pipeline; codice macchina di lunghezza fissa e unità di controllo semplificata.

Un esempio di RISC è il RISC I progettato da D. Patterson avente un singolo chip con ALU a 32 bit. Il suo ciclo base era quello di lettura di due registri (Rs,S2), operazioni su ALU (F), e scrittura su registro /Rd): Rd <= F(Rs,S2)

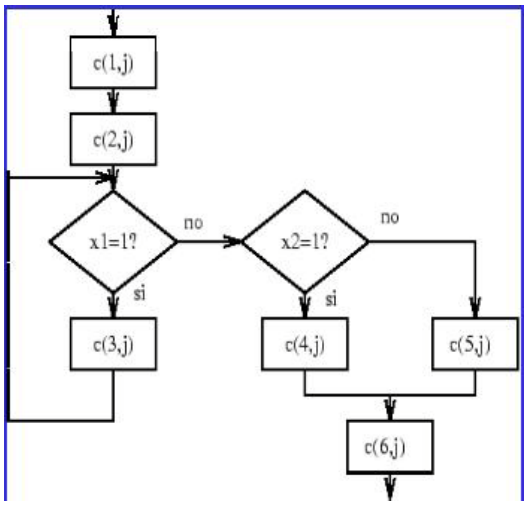
14. Processori RISC

Negli ultimi anni i calcolatori hanno seguito la legge di Moore: le prestazioni sono raddoppiate ogni 18-24 mesi, grazie soprattutto all'aumento del numero dei transistor sul chip, sempre più piccoli e veloci. Questa tendenza non può proseguire a causa della dissipazione di energia: POWER WALL.
Questo implica un evoluzione architetturale. Da qui si iniziarono a introdurre i primi sistemi mobili: con dimensioni ridotte, bassi consumi e bassi costi; "system-on-chip": con soluzione integrata, semplificazione e compatibilità.
Da qui nacque l'architettura ARM, cioè un RISC a 32 bit, con istruzioni a lunghezza fissa, architettura load/store, indirizzamenti semplici, esecuzione condizionale e supporto istruzioni comprese.
Ora vediamo in breve le evoluzioni dei processori ARM.

- ARM 6: processore RISC di tipo load/storecon indirizzi a dati a 32 bit; ALU solo per gli interi e 16 registri generali a 32bit; 25 istruzioni base modificabili.
- ARM 7: pipeline a 3 stadi, cache unificata e supporto Thumb.
- ARM 9:pipeline a 5 stadi, raddoppio della banda di memoria e supporto coprocessori integrato.
- ARM10:pipeline a 6 stadi, branch prediction e load/store non bloccanti.
- ARM11: pipeline a 8 stadi, branch prediction e possibile architettura a più processori.

15. Unità di controllo cablate

Il controllo cablato è un circuito sequenziale speciale che permette di realizzare la funzione per il controllo di esecuzione, ottenendo quindi i segnali di controllo desiderati. Per sistemi semplici è poco costoso, mentre di difficile progettazione per i sistemi complessi. Ci sono due principali metodi di progetto: macchine a stati finiti (progetto tradizionale di circuiti sequenziali) e generatori di sequenze basati su elementi di ritardo (asincroni) o contatori (sincroni). Una possibile descrizione del comportamento dell'unità di controllo è data dal Diagramma di Flusso.



16. Discutere brevemente le ragioni dell'introduzione delle architetture RISC e le principali differenze tra queste e le architetture tradizionali di tipo CISC.

Le ragioni dell'introduzione del RISC sono da ricercare nel fatto che le istruzioni di Load/Store sono preponderanti, quindi vanno ridotte (L/S 50%, Branch/Compare 20%, Aritmetiche 20%...).
L'architettura esterna è caratterizzata da poche istruzioni e pochi modi di indirizzamento; l'accesso alla memoria è limitato alle istruzioni L/S mentre le operazioni logiche vengono svolte con operandi presenti nei registri interni. L'architettura interna è organizzata a pipeline con un'unità di controllo semplificata e l'esecuzione delle istruzioni risulta molto veloce (una per ciclo); le istruzioni hanno inoltre lunghezza fissa.
CISC è l'acronimo di Complex Instruction Set Computer: tipicamente un processore di questo tipo implementa un numero relativamente scarso di registri di uso generale, ed ha una unità di controllo microprogrammata. Il set di istruzioni associato a CPU di tipo CISC è molto esteso e composto in genere di alcune centinaia di codici operativi diversi che svolgono funzioni anche molto complesse, fra cui sono caratteristici i trasferimenti memoria-memoria, assenti nei RISC; le istruzioni hanno lunghezza variabile e possono presentarsi in formati diversi, e sono necessari due o più cicli di clock per completare una istruzione. Il ridotto numero di registri interni obbliga questi processori a scrivere in memoria ogni volta che si verifica una chiamata di funzione o che viene salvato un registro nello stack.

17. Come vengono gestite le Interruzioni

La presenza di un interrupt è segnalata alla CPU da una linea proveniente dall'esterno. Il segnale viene memorizzato e testato dalla CPU alla fine di ogni ciclo di istruzione. La CPU risponde trasferendo il controllo ad un altro programma. L'evento che causa l'interruzione è asincrono

rispetto all'esecuzione del programma. Vediamo come reagisce la CPU alla richiesta di interruzione:

- Identificazione della sorgente di interruzione
- La CPU ottiene l'indirizzo di memoria del programma di servizio
- Il PC e altre informazioni essenziali relative alla CPU vengono memorizzate automaticamente
- Lo stato complessivo della CPU viene memorizzato a cura del programma di gestione
- Nel PC viene immagazzinato l'indirizzo del sottoprogramma di gestione
- L'esecuzione del sottoprogramma continua fino all'istruzione di RETURN che importa il controllo al programma interrotto

La gestione delle interruzioni può essere:

- CENTRALIZZATA: tutte le richieste sono gestite da un solo modulo centralizzato
- DISTRIBUITA: la selezione del periferico avviene con Polling (ogni elemento periferico dispone di un registro che contiene un bit che segnala la richiesta di interruzione) o Vettorizzazione (ogni periferico invia alla CPU un valore o vettore identificativo):

18. Gestione di I/O a controllo di programma

Tecnica più immediata per gestire I/O, il processore si limita a eseguire appositi sottoprogrammi di "ingresso" o "uscita" durante l'esecuzione di un programma. Con questa tecnica il processore, prima di effettuare l'operazione di I/O, controlla che la periferica sia pronta al trasferimento, e se ciò non si verifica, rimane in attesa finché dalla periferica stessa non giunge l'opportuno segnale di disponibilità. Questi cicli di attesa permettono di evitare di perdere i dati inviati ad una periferica che in quel momento è occupata. Possiamo usare un flag di disponibilità; forzato a 1 dalla periferica, quando questa è disponibile a ricevere/inviare un dato, e forzato a 0 dal processore durante l'operazione di trasferimento dati.
L'interfaccia deve fornire:

- Un registro per i dati trasferiti
- Un registro per i comandi alla periferica
- Un registro per lo stato della periferica

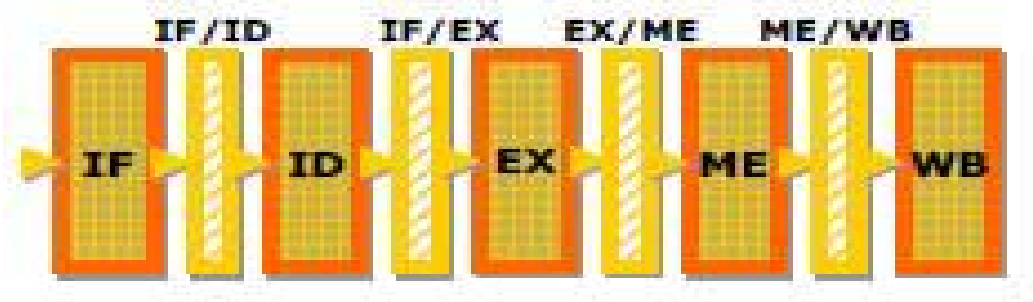
Questo metodo di gestione penalizza la velocità di elaborazione perché il processore entra in un ciclo di attesa inoperosa, per contro richiede però interfacce molto semplici e consente di effettuare direttamente da programma un controllo I/O.

19. Descrivere l'architettura di una CPU con organizzazione a pipeline. Individuare le funzioni svolte dai singoli stadi

Nelle architetture pipeline l'esecuzione di una singola istruzione consiste di più fasi, svolte in sequenza da sezioni diverse. In questo modo si aumenta il throughput senza aumentare la velocità dell'unità centrale. Si introduce tuttavia un ritardo tra ingresso e uscita dei dati.
Le fasi sono:

- FETCH: fetch dell'istruzione dalla memoria
- DECOD: decodifica dell'istruzione
- OPER: cattura dell'operando dalla memoria
- ESEC: esecuzione dell'istruzione
- WR: scrittura dei dati

Lo svolgimento della singola istruzione è una scelta di progetto.



20.Architetture superscalari. Mostrare un esempio di architettura superscalare e discuterne i vantaggi e i problemi introdotti

Un microprocessore con architettura superscalare supporta il calcolo parallelo su un singolo chip, permettendo prestazioni molto superiori a parità di clock rispetto ad una CPU ordinaria. In un processore superscalare istruzioni diverse trattano i propri operandi contemporaneamente, su diverse unità hardware all'interno dello stesso chip. In questo modo più istruzioni possono essere eseguite nello stesso ciclo di clock. Sono presenti diverse unità funzionali dello stesso tipo, con dispositivi addizionali per distribuire le istruzioni alle varie unità. Per esempio, sono generalmente presenti unità per il calcolo interno (AI), floating (AF), branch(B) e load/store (L/S). Si può creare parallelismo adottando più unità funzionali per non far pagare anche alle istruzioni più veloci il ritardo nella fase di EX (IF->ID->EX->ME->WB). Le operazioni AF, ad esempio, richiederanno sicuramente più cicli di clock rispetto alle operazioni AI, quindi da qui nasce il problema dell'ordine di completamento non mantenuto; infatti se in ingresso ho un operazione tra float e poi un'operazione tra interi, in uscita avrò prima il risultato della seconda e poi quella della prima. Il compilatore può cercare di riordinare le istruzioni in modo da ottimizzare l'uso delle entità ma in generale è meglio che la CPU faccia concludere le istruzioni in ordine.

21. Accesso diretto alla memoria. Illustrare l'architettura del modulo e i vantaggi di questa soluzione.

Noi potremmo trovarci davanti a dei problemi tipo l'organizzazione nella memoria cache dei dati provenienti dalla memoria principale oppure non sapere come viene aggiornata la memoria principale dopo che la CPU ha modificato un dato nella cache. In poche parole, a noi serve che il dato richiesto possa essere trovato rapidamente nella memoria cache. Questo è possibile grazie a tre diverse "tecniche":

- Tecniche associative
- Accesso diretto

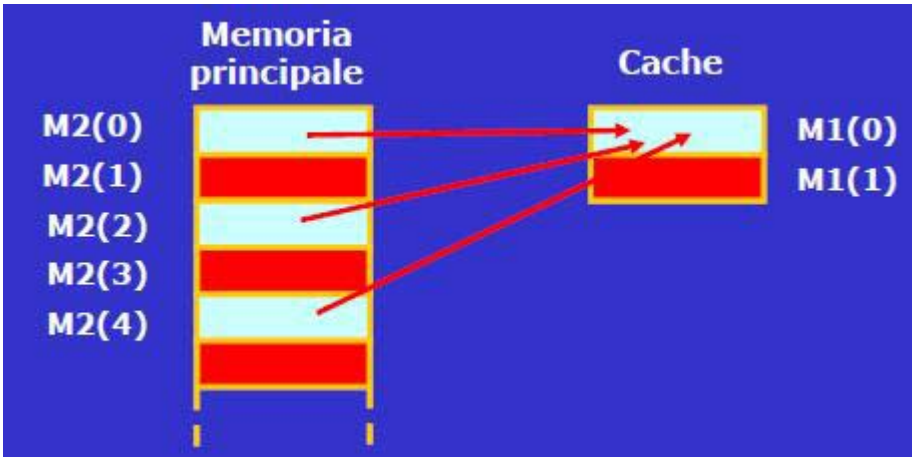
- Tecniche miste o set-associative

Nel metodo ad accesso diretto, la memoria cache M1 è suddivisa in S1 = 2^k pagine (o linee) di n parole. La memoria principale M2 è suddivisa in S2 regioni di egual dimensioni.

M1(1),…M1(i),…M1(S1): pagine in cache

M2(1),…M2(j),…M2(S2):pagine in memoria principale

M2(j) va in M1(i) con i=j (mod S1)



22. Architetture a cluster. Illustrare i vantaggi di questa soluzione architetturale.

I Cluster sono una classe di architetture di elaborazione parallele basate su un insieme di elaboratori indipendenti cooperante per mezzo di una rete di interconnessione e coordinato mediante un sistema operativo che lo rende in grado di operare su un singolo workload. Un cluster può essere utilizzato in molti modi:

- Elevate prestazioni su un singolo problema
- Elevato throughput o elevata capacità con un carico di lavoro a processi
- Elevata disponibilità mediante la ridondanza dei nodi
- Elevata banda ottenuta dalla molteplicità dei dischi o dei canali di I/O

Ma i vantaggi più importanti per giustificare l'interesse così forte sono:

1- *Buon rapporto prestazioni/prezzo:*
I commodity cluster possono migliorare il rapporto prestazioni/prezzo di un ordine di grandezza rispetto ad architetture diverse
2- *Rafforzamento del ruolo del personale:*
Sistemi a cluster a elevate prestazioni utilizzano moduli elementari che derivano dai sistemi di uso personale. L'amministrazione HW e SW richiedono competenze maggiormente disponibili sul mercato.
3- *Tempi e costi di sviluppo:*

Questi sistemi si basano su elementi disponibili sul mercato spesso per uso personale. L'integrazione non richiede moduli specializzati.

4- *Convergenza delle architetture:*
Per la prima volta gli utenti e venditori hanno a disposizione un'architettura parallela stabile e indipendente. Questo garantisce persistenza a lungo termine dell'architettura e giustifica investimenti software.

23. Spiegare che cosa si intende per memoria associativa e illustrare un esempio di applicazione.

Noi potremmo trovarci davanti a dei problemi tipo l'organizzazione nella memoria cache dei dati provenienti dalla memoria principale oppure non sapere come viene aggiornata la memoria principale dopo che la CPU modifica un dato nella cache. In poche parole a noi serve che il dato richeisto deve essere trovato rapidamente nella memoria cache. Questo è possibile grazie a tre diverse "tecniche":

- Tecniche associative
- Accesso diretto
- Tecniche miste o set-associative

Vediamo meglio la memoria associativa. Nella memoria tradizionale, in lettura, si restituisce un dato conoscendone la posizione (indirizzo); nella memoria associativa, invece, restituisce un'informazione associata fornendo un dato.

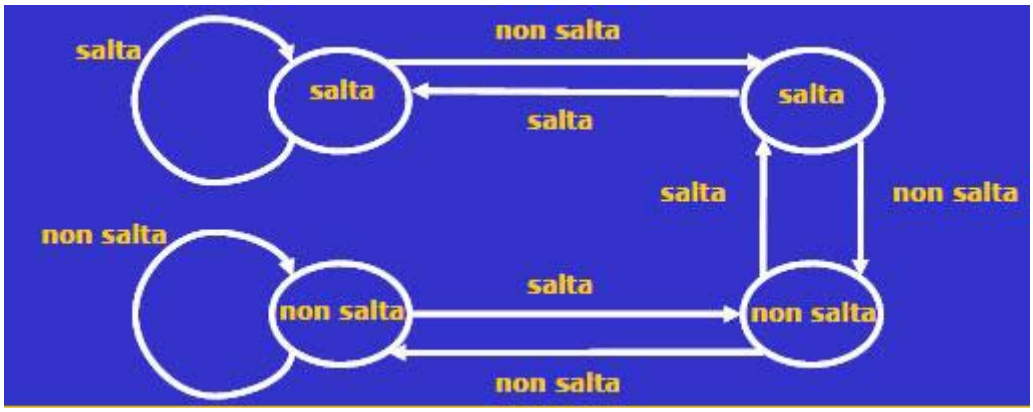
Nella memoria associativa il tag viene utilizzato come chiave di lettura. L'unico problema è che la tecnica associativa è molto costosa quando crescono le dimensioni della memoria.

24. La predizione dei salti in una architettura pipeline. Perché viene usata? Quali problemi risolve? Come si implementa?

Maggiori sono il grado di parallelismo e il numero di istruzioni pre-processate contemporaneamente, maggiore è la perdita relativa di performance dovuta alla non-predicibilità del flusso di istruzioni. Per questo è utile fare previsioni attendibili sui salti condizionati, per poter continuare a fare fetch e decodifica delle istruzioni seguenti prima di aver eseguito il salto. Tali previsioni si basano su informazioni statistiche molto semplici aggiornate dinamicamente per ciascuna istruzione di salto.
La predizione dei salti può essere di due tipi:

- Predizione statica: determinazione statica: si assume che il salto avvenga sempre o che non avvenga mai.
- Predizione dinamica: determinazione dell'esecuzione del salto sulla base della storia precedentemente memorizzata (ad esempio, nella cache).

La determinazione dinamica è basata sulla presenza di informazioni specifiche (2 bit) nella cache dove viene contenuto il codice sulla base di un particolare algoritmo sequenziale. Ogni volta che viene eseguita una particolare istruzione viene aggiornato lo stato dell'istruzione stessa.

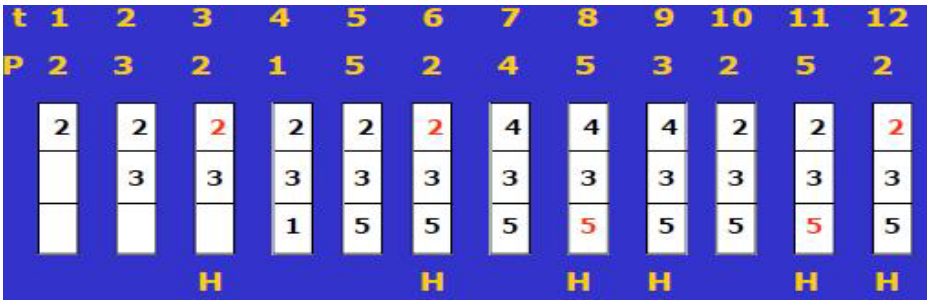


Una tecnica di implementazione è la **Branch Prediction**. È una tecnica di implementazione che aumenta di molto le prestazioni delle istruzioni di salto. Intuitivamente, la CPU "ricorda" l'esito di ogni istruzione di salto condizionato (quello incondizionato non ha bisogno di essere ricordato, ovviamente) salvandolo in una apposita cache interna (la Branch Cache).

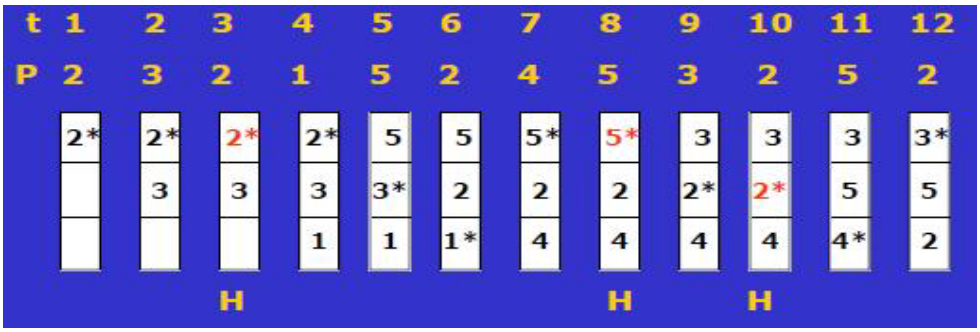
25. Le politiche di sostituzione in una struttura di memoria gerarchica.

L'obiettivo delle politiche di sostituzione è quello di minimizzare il numero di insuccessi (memory fault). L'ipotesi è di assumere massimo il tasso di successo H se si rende massimo l'intervallo tra due fault successivi.
Possiamo suddividere le politiche di sostituzione in:

- Ottima: è quella che richiede più passi:
  1- Eseguire il programma per ricavare le richieste di accesso
  2- Calcolare le sostituzioni ottima
  3- Eseguire il programma con la sequenza ottima
  Essa non è utilizzabile perché l'identificazione della strategia ottima richiede l'esecuzione preventiva del programma.



- FIFO – First In First Out: si associa al blocco un numero di sequenza e si sostituisce il blocco introdotto meno recentemente.



- LRU – Least Recently Used: si sostituisce il blocco che da più tempo non viene utilizzato.