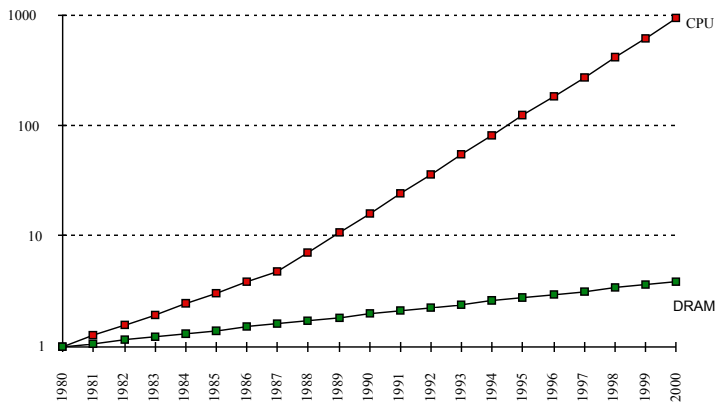
La gerarchia di Memoria

Gap delle prestazioni DRAM - CPU

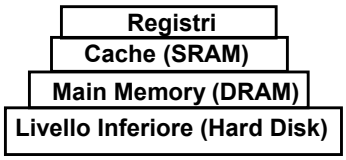


Località ed Organizzazione Gerarchica

- Le RAM statiche sono veloci ma consumano molto ed offrono densità medio/basse
- Le RAM dinamiche consumano poco ed offrono densità molto alte ma sono lente rispetto alla CPU (ed ogni anno il divario aumenta

Località

- I programmi godono della proprietà di località sia Spaziale sia Temporale
- **Località temporale:**
  - È molto probabile che un'istruzione verrà referenziata nuovamente a breve
- **Località spaziale**
  - È molto probabile che vengano referenziate istruzioni vicine a quella attualmente in esecuzione
- La località dei programmi suggerisce una gerarchia di memoria



Gerarchia di Memoria

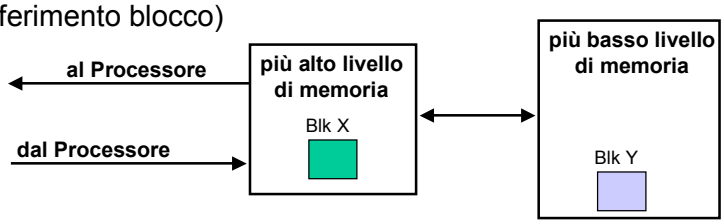
- Livelli più alti
  - Alta Velocità, Alto Costo, Piccole Dimensioni
- Livelli più bassi
  - Bassa Velocità, Basso Costo, Grandi Dimensioni
- Ordine di ricerca di un dato in memoria:
  dai livelli più alti ai più bassi
- Il trasferimento di informazioni all'interno della gerarchia avviene tra livelli adiacenti.
- Ogni livello è organizzato in blocchi di n byte

Tempo medio di accesso alla memoria

$$t_{AMAT} = \text{Hit time} + \text{miss rate} \bullet \text{miss penalty}$$

dove:

- **hit rate:** tentativi riusciti/numero di tentativi
- **miss rate:** miss rate= 1 – hit rate
- Hit time = (tempo di accesso + tempo per determinare se dato è in livello attuale)
- Miss penalty= (tempo accesso al livello inferiore + tempo trasferimento blocco)



Quattro domande per chi progetta la gerarchia di memoria

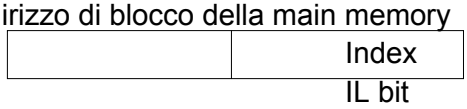
- Q1: Dove piazzare un blocco ?
  (Block placement)
- Q2: Come faccio a sapere se un blocco è presente?
  (Block identification)
- Q3: Quale blocco devo sostituire nel caso di Miss?
  (Block replacement)
- Q4: Cosa succede in scrittura ?
  (Write strategy)

Q1: Dove piazzare un blocco ?

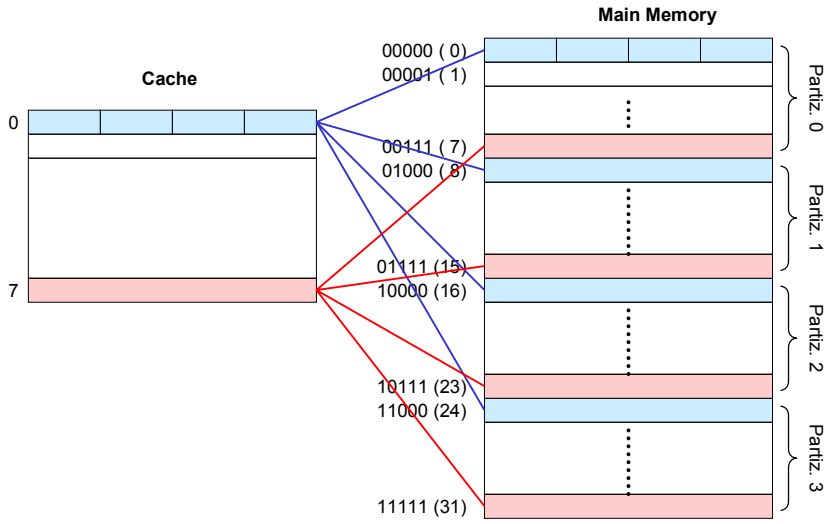
- Full Associative:
  - Un blocco della main memory può essere mappato in un blocco qualsiasi della cache
- Direct Mapped:
  - Data una memoria cache di NB blocchi, il blocco della memoria principale di indice j può essere mappato solo nel blocco della memoria cache di indice

    Index = j modulo NB
  - L'indice del blocco in cache è ottenuto considerando gli IL=$\log_2$NB bit meno significativi dall'indice del blocco della memoria principale

Indirizzo di blocco della main memory



Cache direct-mapped



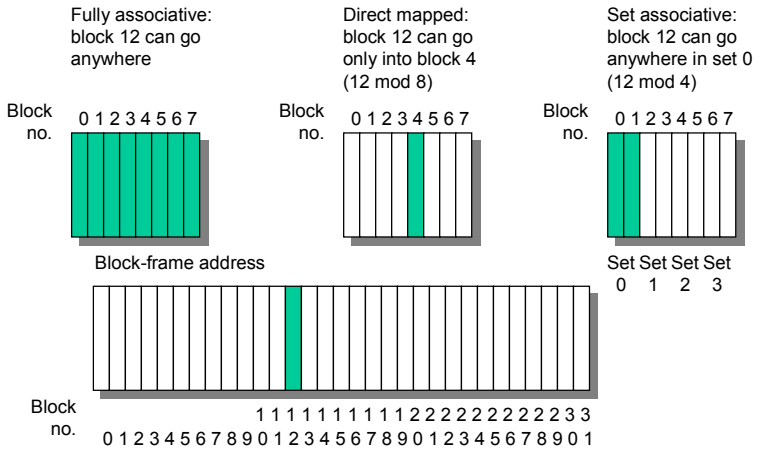
Q1: Dove piazzare un blocco ?

- N-way Set Associative
  - Data una memoria cache di NS set, ciascuno di N blocchi, il blocco della memoria principale di indice j può essere mappato nel set della memoria cache di indice

    Index = j modulo NS
  - All'interno del set un blocco può essere piazzato in una posizione qualsiasi.
  - L'indice del set in cache è ottenuto considerando gli IL=$\log_2$NS bit meno significativi dall'indice del blocco della memoria principale.

Q1: Dove piazzare un blocco ?

- Block 12 placed in 8 block cache:
  - Fully associative, direct mapped, 2-way set associative
  - S.A. Mapping = Block Number Modulo Number Sets



Q2: Come faccio a sapere se un blocco è presente?

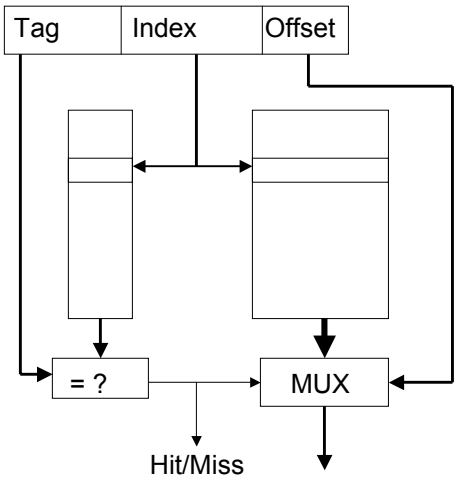
- Data una cache direct mapped di NB blocchi e un main memory di NM blocchi, nello stesso blocco della cache possono essere mappati NM div NB blocchi.
- Per sapere se il blocco presente in cache è quello effettivamente cercato, oltre a memorizzare il blocco, viene memorizzata una informazione supplementare, il tag del blocco ovvero la parte più significativa dell'indirizzo del blocco che si sta cercando.
- La memorizzazione del Tag richiede IB= $\log_2$ (NM div NB) bit

Indirizzo di blocco generato dal processore

Tag	Index	Offset

- Mediante l'index viene individuato il blocco in cache, mediante il Tag viene verificato se il blocco presente è quello cercato
- L'offset individua la word all'interno del blocco

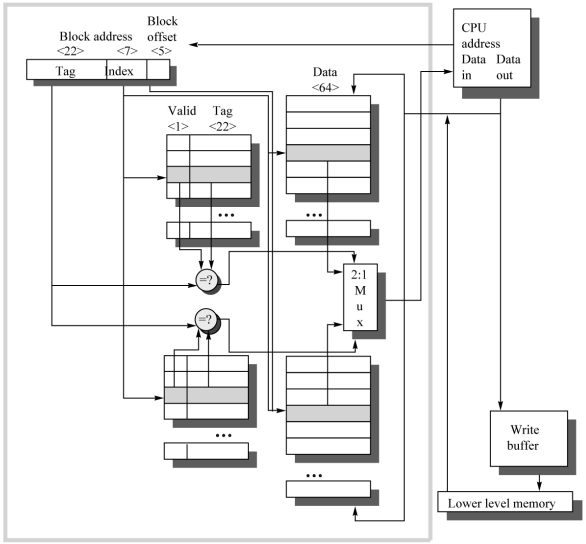
Q2: Come faccio a sapere se un blocco è presente?



Q2: Come faccio a sapere se un blocco è presente?

- Data una cache Set Associative a N vie di NS set e un main memory di NM blocchi, nello stesso set della cache possono essere mappati NM div NS blocchi.
- La memorizzazione del Tag richiede IB= $\log_2$ (NM div NS) bit
- Fissata la dimensione della memoria cache, all'aumentare del numero di blocchi di ciascun set diminuisce il numero di set presenti (diminuisce il numero di bit dell'index e aumenta quello del tag)
- Nel caso di memoria full associative possiamo pensare la cache con un unico set e pertanto il tag coincide con l'indirizzo del blocco

Q2: Come faccio a sapere se un blocco è presente?



Q3: Quale blocco devo sostituire nel caso di Miss?

- Random
- Least Recently Used

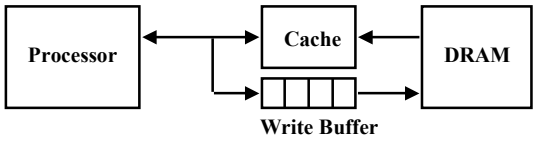
Associativity:		2-way		4-way		8-way
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.18%	5.69%	4.67%	5.29%	4.39%	4.96%
64 KB	1.88%	2.01%	1.54%	1.66%	1.39%	1.53%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Q4: Cosa succede in scrittura ?

- Write through: L'informazione e' scritta sia in cache sia nel livello inferiore di memoria.
- Write back: L'informazione e' scritta solo in cache. Il blocco della cache modificato è scritto in main memory solo quando è sostituito

- WT è combinato con un write buffers in modo da non aspettare il livello inferiore di memoria.

Write buffer



- Un Write Buffer è richiesto tra Cache e Memoria
  - Processore: scrive I dati sulla cache e sul write buffer
  - Memory controller: scrive il contenuto del buffer in memoria
- Il Write buffer è una coda :
  - Tipica dimensione : 4
  - Lavora bene se la frequenza di scrittura << 1 / DRAM write cycle

Write miss policy

- Write allocate
  - Il blocco viene scritto in cache
- No Write allocate

  Il blocco viene scritto direttamente in main memory

Tipicamente il write allocate è associato al write back, mentre il no write allocate è associato al write through

Miglioramento delle performance della cache

t_{AMAT} = Hit time + miss rate•miss penalty

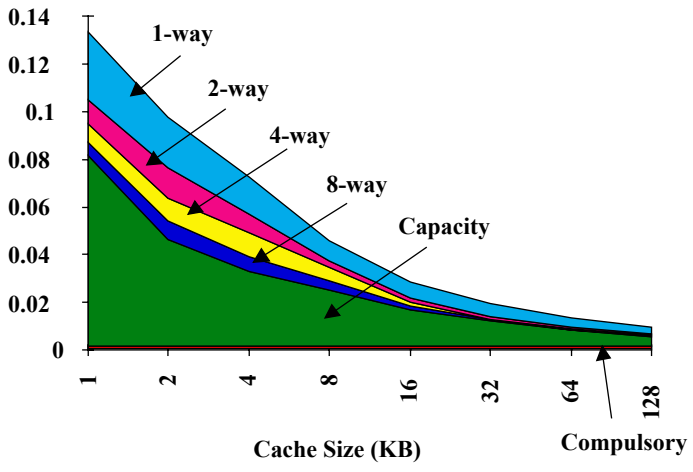
Il miglioramento può essere ottenuto:

1. Riducendo il miss rate,
2. Riducendo il miss penalty
3. Riducendo il tempo di hit in cache.

Cause Miss Rate

- Compulsary
  - Dovuto al riempimento iniziale della cache nella fase di avvio
- Capacity
  - Dovuto alla limitata dimensione della cache
- Conflict
  - Dovuto al conflitto tra più blocchi della main memory sullo stesso blocco della cache

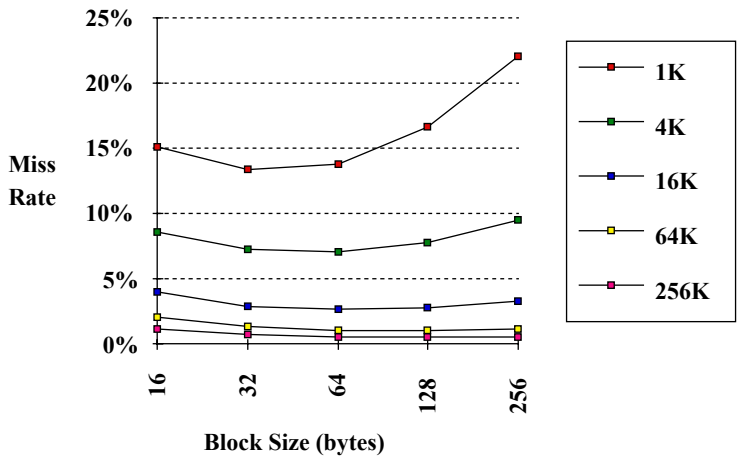
Cause Miss Rate



Elementi che influenzano il miss

- Dimensione della cache
- Dimensione del blocco
- Associatività

DIMENSIONE DEL BLOCCO



Associatività di una cache

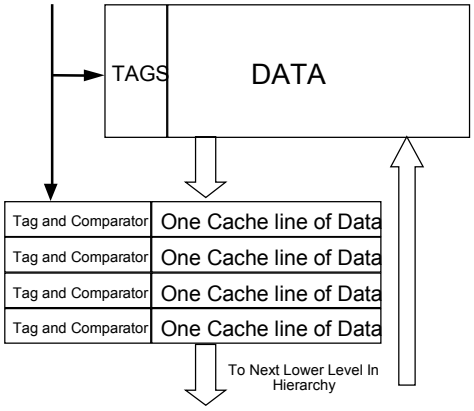
- Il miss rate di una cache di dimensione NM diminuisce all'aumentare dell'associatività
- Poiché contemporaneamente aumenta l'hit time, ciò che bisogna valutare è l'AMAT.

Tempo medio di accesso vs Miss Rate

Cache Size	Associativity			
(KB)	1-way	2-way	4-way	8-way
1	2.33	2.15	2.07	2.01
2	1.98	1.86	1.76	1.68
4	1.72	1.67	1.61	1.53
8	1.46	1.48	1.47	1.43
16	1.29	1.32	1.32	1.32
32	1.20	1.24	1.25	1.27
64	1.14	1.20	1.21	1.23
128	1.10	1.17	1.18	1.20

Ridurre il Miss Rate mediante Victim Cache

- *Come combinare un hit time come nella direct mapped riducendo i conflict misses?*
- Si aggiunge un buffer dove piazzare i dati scartati dalla cache
- Jouppi [1990]: Una victim cache di 4 blocchi rimuove dal 20% al 95% dei conflitti per una direct mapped data cache di 4 KB.



Subblock placement

- Non deve essere sostituito un intero blocco su un miss
- È presente un bit di validità per ogni sottoblocco
- Originariamente inventato per ridurre l'occupazione di memoria per il tag

Tag	validity bit		subblock				
201	1		1		1		1	
302	1		1		0		1	
202	0		0		1		1	
204	0		1		1		1	

Cache Pseudo-associativa

- Come combinare il veloce hit time della Direct Mapped con i ridotti conflict misses della 2-way Set Associative cache?
- La cache è divisa in due metà.
- La verifica dell'hit viene fatta in una metà come una direct-mapped. Nel caso di miss viene testata l'altra metà.
- Nel caso di hit nella seconda metà sia ha uno pseudo-hit (hit lento)

- Problemi: CPU pipeline è complicato poiché l'hit impiega 1 o 2 cicli
  - Ideale per cache non collegate direttamente al processore

Cache di secondo livello

$AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times Miss\ Penalty_{L1}$

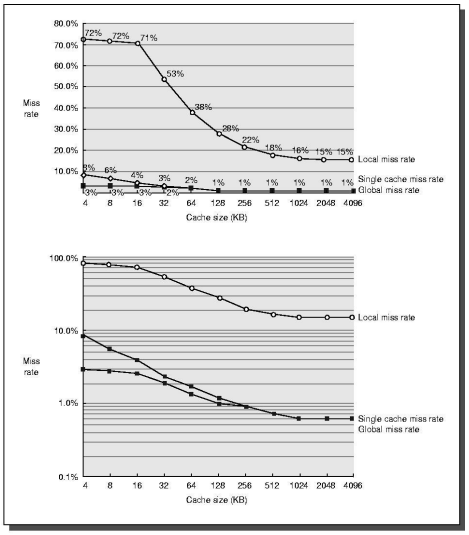
$Miss\ Penalty_{L1} = Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2}$

$AMAT = Hit\ Time_{L1} + Miss\ Rate_{L1} \times (Hit\ Time_{L2} + Miss\ Rate_{L2} \times Miss\ Penalty_{L2})$

- Definizioni:
  - *Local miss rate*— miss in questa cache diviso il numero di accessi a questa cache ($Miss\ rate_{L2}$)
  - *Global miss rate*—miss in questa cache diviso il numero totale di accessi generati dalla CPU ($Miss\ Rate_{L1} \times Miss\ Rate_{L2}$)

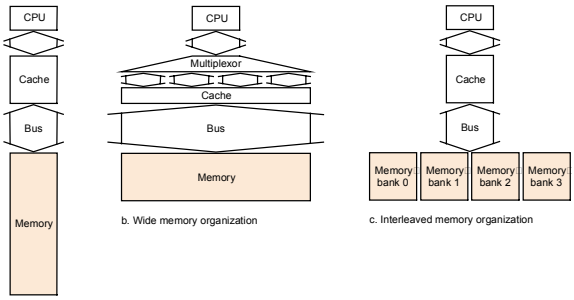
Confronto tra local e global miss rate

- 1° livello 32 KByte; 2° livello crescente
- Il global miss rate è vicino a quello di una cache con un solo livello se L2 >> L1
- Non bisogna usare il local miss rate
- Si hanno veloci hit time e un numero inferiore di miss
- Poiché gli hit (di L2) sono pochi, l'obiettivo è la riduzione del miss



Memory interleaved

- *Simple*:
  - CPU, Cache, Bus, Memory same width (32 or 64 bits)
- *Wide*:
  - CPU/Mux 1 word; Mux/Cache, Bus, Memory N words (Alpha: 64 bits & 256 bits; UtraSPARC 512)
- *Interleaved*:
  - CPU, Cache, Bus 1 word: Memory N Modules (4 Modules); example is *word interleaved*



Memory Interleaved

- Timing model (word size is 32 bits)
  - 1 to send address,
  - 6 access time, 1 to send data
  - Cache Block is 4 words

- *Simple M.P.* = 4 x (1+6+1) = 32
- *Wide M.P.* = 1 + 6 + 1 = 8
- *Interleaved M.P.* = 1 + 6 + 4x1 = 11