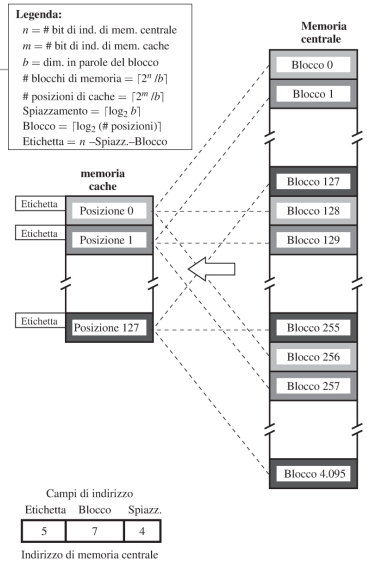
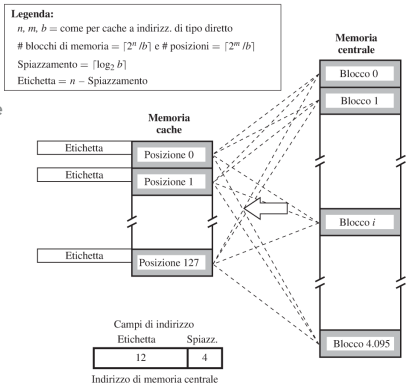
Indirizzamento diretto

- Si abbia una **cache** di **128 posizioni**, ciascuna che può contenere un **blocco** di **16 parole**, quindi un totale **2K parole**
- Si abbia una **memoria** centrale con **64K parole** (indirizzi di **16** bit), quindi può contenere **4K blocchi**
- **Indirizzamento diretto**: ogni blocco di memoria centrale può essere caricato in una sola posizione in cache, data dal resto della divisione fra numero del blocco e 128 (*i mod 128*). Più blocchi occupano la stessa posizione in cache in momenti diversi
- L'indirizzo della parola di memoria viene suddiviso in tre parti (campi)
- Il ***campo blocco*** *dell'indirizzo* (7 bit) individua la **posizione** in cache (1 fra 128 posizioni), il ***campo spiazzamento*** (4 bit) individua la **parola** nel blocco (1 fra 16 parole), il ***campo etichetta*** (5 bit) è registrato come etichetta associata al blocco della cache
- Decisione hit/miss semplice e rapida: per il blocco in cache individuato dal campo blocco dell'indirizzo si confronta il campo etichetta con l'etichetta presente nella cache



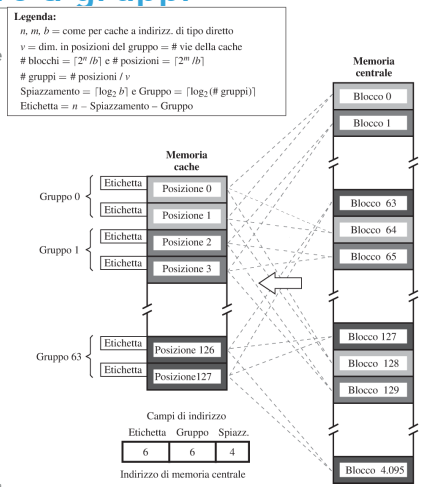
Indirizzamento associativo

- Ogni blocco è caricabile in qualsiasi posizione in cache. Si sostituisce un blocco solo se tutta la cache è occupata, ovvero lo spazio della cache è caricato in modo efficiente
- La valutazione di cache hit/miss è più costosa: si devono **confrontare tutte le etichette**, si deve avere un algoritmo di sostituzione (es. LRU)
- L'indirizzo di memoria è diviso in due campi: ***etichetta*** e ***spiazzamento***. Il campo *etichetta* è di 12 bit per identificare quale fra i 4K=2^12 blocchi della memoria sia caricato in cache
- Per stabilire se la parola cercata è in cache occorre confrontare il campo etichetta dell'indirizzo con tutte le 128 etichette di posizione
- Tale ricerca si chiama **ricerca associativa** e va condotta in parallelo su tutte le etichette



Indirizzamento associativo a gruppi

- Le **posizioni di cache sono riunite in gruppi** e la relazione è fra blocchi in memoria e gruppi in cache
- Quindi si può copiare il blocco in una qualsiasi posizione del gruppo, e si ha corrispondenza associativa per blocchi all'interno di un gruppo, e corrispondenza diretta per blocchi sui gruppi
- Si **riduce il conflitto** dell'indirizzamento diretto, ma è meno flessibile dell'indirizzamento associativo
- L'indirizzo è suddiviso in tre campi: *spiazzamento*, *gruppo* e *etichetta*
- Si confrontano tutte le etichette di un gruppo per cercare se la parola è in cache
- La cache si dice **a n vie**, con **n numero di posizioni** del gruppo, nell'esempio è una cache a due vie



Dati scaduti

- Problema di **coerenza** della cache: due unità funzionali diverse (DMA e processore) devono far riferimento a copie identiche di blocchi di dati
- Il DMA trasferisce dati fra memoria centrale e disco
- Nel caso di **scrittura di blocchi** di memoria da parte del DMA (che legge dal disco), le corrispondenti copie in cache devono essere marcate come **non valide**, poiché contengono **dati scaduti** (*stale data*), ovvero non più presenti in memoria
- A ciascuna posizione in cache è associato un **bit di validità** (oltre al *bit di modifica*, e all'etichetta)
- Nel caso di **lettura di blocchi** di memoria da parte del DMA (che scrive sul disco), se la cache è gestita in modalità *scrittura differita* (*write back*), deve prima essere *svuotata la cache* (*cache flush*), in modo che i cambiamenti più recenti (marcati dal bit di modifica) siano copiati nella memoria centrale
  - L'operazione di svuotamento della cache è svolta dal sistema operativo, non comporta rallentamenti significativi poiché lo svuotamento della cache è molto più veloce rispetto allo svuotamento della memoria centrale

Algoritmi di sostituzione

- Permettono di scegliere in quale posizione della cache caricare un blocco di memoria
  - Se si usa l'indirizzamento diretto, la posizione è determinata da tale schema
  - Altrimenti, occorre usare un algoritmo di sostituzione che determina quale blocco sostituire quando tutte le posizioni allocabili contengono blocchi validi
- L'algoritmo **LRU** (least recently used) sostituisce il blocco meno recentemente usato
- Per una memoria cache associativa a gruppi, con 4 posizioni per gruppo (a 4 vie), per ogni posizione (del gruppo) vi sarà un contatore da due bit (modulo 4), inizialmente vale 0
  - Se si ha una **cache hit**, il contatore della posizione interessata va messo a 0, i contatori del gruppo con valore minore (prima dell'azzeramento) sono incrementati di 1
  - Se si ha una **cache miss** e si ha una posizione libera nel gruppo, si carica il blocco, il contatore della sua posizione vale zero e gli altri contatori sono incrementati di 1
  - Se si ha una **cache miss** e il **gruppo è pieno**, la posizione con contatore massimo (valore 3) va liberata e riempita col nuovo blocco, il contatore corrispondente vale 0, e gli altri contatori vanno incrementati di 1
  - Si può dimostrare che in ciascun gruppo i quattro contatori sono sempre diversi fra loro

Considerazioni di prestazione (S. 8.7)

- La gerarchia di memoria permette di minimizzare il rapporto costo/prestazione, fa sì che il processore veda una memoria con accesso breve e capacità grande
- Un ottimo indicatore di efficacia per la memoria cache è il **tasso di successo** o **hit rate**, **h**, ovvero il rapporto fra numero di accessi con hit e numero totale di accessi
- **Tasso di insuccesso**, o **miss rate** è pari a **1 - h**
- Le prestazioni degradano nel caso di evento di miss
- Il tempo totale di accesso quando accade una miss è detto **penalità di miss**, **M** (ovvero, il tempo necessario a caricare un blocco da memoria centrale a cache, e il **tempo di accesso alla cache**, **C**)
- Tempo medio di accesso alla memoria $t_{avg} = hC + (1 - h)M$

Esempio di stima del guadagno

- Si abbia: tempo di accesso τ alla cache, e 10τ alla memoria centrale
- Si abbiano 8 parole per blocco, allora il tempo totale di trasferimento del blocco da memoria a cache è $(10 + 7)\tau = 17\tau$, ovvero occorrono 10τ per la prima parola e 1τ per ogni parola successiva
- In una cache miss si hanno due accessi alla cache, quindi la penalità di miss è $M = \tau + 10\tau + 7\tau + \tau = 19\tau$
- Si supponga che ci siano 100 istruzioni e che il 30% di esse acceda alla memoria
- Si abbia un **tasso di hit** per le **istruzioni** pari a 0,95, e un **tasso di hit** per i **dati** pari a 0,9
  - Il tempo di accesso senza cache è $(100 + 30) \cdot 10\tau = 1300\tau$
  - Il tempo di accesso con cache è $100 \cdot (0{,}95\tau + 0{,}05 \cdot 19\tau) + 30 \cdot (0{,}9\tau + 0{,}1 \cdot 19\tau) = 190\tau + 84\tau = 274\tau$
- guadagno = tempo senza cache / tempo con cache = 4,7
- Ovvero, la cache fa apparire la memoria quasi cinque volte più veloce di quello che è

Miglioramento delle prestazioni

- Per migliorare il tasso di successo, si può aumentare la dimensione della cache, ma questo significa un maggior costo
  - Oppure, aumentare la dimensione dei blocchi per aver migliori effetti dati dalla località spaziale
  - Tuttavia, blocchi tanto grandi farebbero diminuire le hit, successivamente
  - In pratica si è visto che la dimensione migliore è fra 16 e 128 byte
  - Inoltre, si può ridurre la penalità di miss usando *load through*

Due livelli di cache

- Si abbia un processore con due cache L1, una per istruzioni e una per dati, e una cache L2 più grande
- Il tempo medio di accesso è $t_{avg} = h_1C_1 + (1 - h_1)(h_2C_2 + (1 - h_2)M)$
- Con h_1 tasso di hit per cache L1, h_2 tasso di hit per cache L2, C_1 tempo di accesso cache L1, C_2 tempo di accesso cache L2, M tempo di accesso alla memoria
- Se $(1 - h_1)(1 - h_2)$ è molto piccolo allora un'alta penalità di miss (ovvero M) è tollerabile, ovvero la memoria M lenta è meno critica
- Se $h_1 = h_2 = 0{,}9$ allora $(1 - h_1)(1 - h_2) = 0{,}1 \cdot 0{,}1 = 0{,}01$ quindi il numero di miss in L2 è l'1%. L'incidenza del tempo di accesso alla memoria centrale è ridotta all'1% dei casi

Calcolo tempi medi di accesso

- Si abbia: $C_1 = t$ (il tempo di accesso per le due cache L1), inoltre per trasferire un blocco da L2 a L1 occorre un tempo $C_2 = 15t$ e per trasferire un blocco da Memoria a L2 occorre un tempo $M = 100t$
- Si assuma che i tassi di hit siano uguali per istruzioni e dati e che per L1 e L2 siano $h_1 = 0{,}96$ e $h_2 = 0{,}80$ rispettivamente
- Il tempo medio di accesso visto dal processore è $t_{avg} = h_1C_1 + (1 - h_1)(h_2C_2 + (1 - h_2)M)$
- Quindi $t_{avg} = 0{,}96t + 0{,}04(0{,}80 \cdot 15t + 0{,}20 \cdot 100t) = 2{,}24t$
- Si supponga ora che L2 sia rimossa e che L1 sia più grande in modo da dimezzare il tasso di miss. Il tempo medio di accesso alla memoria è $t_{avg} = 0{,}98t + 0{,}02 \cdot 100t = 2{,}98t$