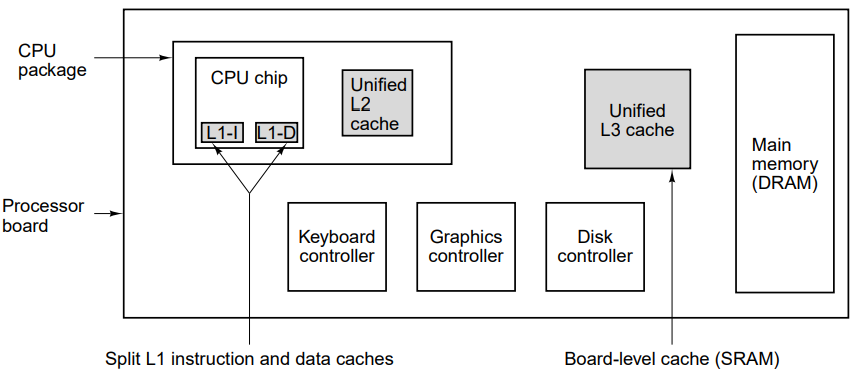
# MEMORIA CACHE



* REGISTRI-RAM: **Memoria Cache Contiene i dati che verranno usati con più probabilità dalla CPU.**

La CPU produce un indirizzo di memoria logico, il quale verrà opportunamente mappato in Cache.

Nella Cache vi è la corrispondenza tra indirizzo logico e fisico della MM

Utilizza una tecnologia SRAM, più veloce e resistente della DRAM. La sua capacità dipende dal suo livello e funzione.

* + **Cache L1** = interna alla CPU
  + **Cache L2** = interna alla scheda madre. Cache intermedia tra multicore per limitare il più possibile il traffico nei bus di sistema
  + **Cache L3** = gestione con la RAM

Quando la CPU chiede dei dati

**Convenienza nell’uso della Cache**

**H = probabilità di Cache HIT  determina l’efficienza di una Cache (ottimale 80-95%)**

**Tc = Tempo di accesso alla cache**

**Tp = tempo di accesso alla RAM**

**Tm = tempo medio di accesso alla RAM in presenza della Cache**

**= Tc + (1-h) \* Tp  Tm < Tp  h > Tc/Tp**

Vengono usate due regole statistiche:

* **Località temporale:** istruzioni e dati che sono usati di recente hanno probabilità di essere richiamati. MRU (Most Recently Used). È possibile che delle **istruzioni** **diverse** **possano** **essere** **interpretate** dalla CPU come **loop** di **microistruzioni** **costanti**, perciò la Cache è efficiente.
* **Località spaziale:** istruzioni e dati contigui a quelli appena usati possono essere chiamati con maggiore probabilità. Molte istruzioni vengono salvate in maniera sequenziale per massimizzare l’efficienza della contiguità durante l’esecuzione di una macroistruzione.

**Località Spaziale**

Si ottiene creando delle “**linee di cache**”

Queste linee contengono 32-64Bytes di locazioni contigue della RAM.

In caso di un Cache Miss viene aggiornata l’intera linea con un’altra appena prelevata dalla RAM.

* **FULLY ASSOCIATIVE:** ogni blocco di memoria RAM può essere memorizzato ovunque nella Cache. Viene inoltre salvato l’intero indirizzo di memoria.

La Cache è divisa in 2 campi

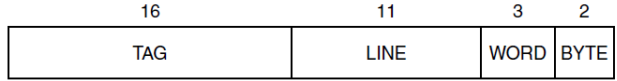
* + **Directory:**
    - **Attribute: contiene gli stati della Cache**
      * **M = Modified, D = Dirty =** il dato è modificato solo all’interno della Cache e la modifica non è stata salvata della RAM
      * **O = Owner, SD (Shared Dirty), SM (Shared Modified), T (Tagged). Owner possiede il dato originale,** mentre le altre linee di cache possono contenere lo stesso dato ma con minore grado di importanza.

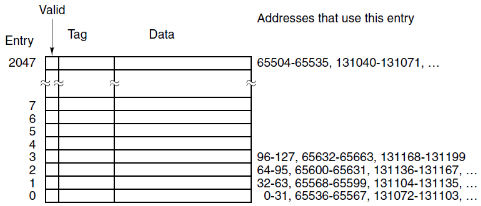
In caso di **richiesta di lettura** sul **BUS** il **dato** viene **fornito dall’Owner**

* + - * **E (Exclusive), R (Reserved), V (Valid), C (Clean):** il dato è valido aggiornato in RAM
      * **Invalid :** se il dato non è presente in Cache, oppure è presente ma non è valido perché modificato o alterato
      * **VALIDITY = 1 bit 0= linea non valida non ancora caricata correttamente 1= linea valida**
    - **INDIRIZZO: indirizzo del blocco di memoria principale**
  + **Data: lunghezza 32/64 Byte,**

Per cercare un dato si deve comparare il campo TAG con tutte le linee di Cache.

Il campo OFFSET esprimerà quale dei N Byte del campo DATA andare a selezionare. Sarà lungo log2 N e i restanti bit andranno a definire il campo TAG



* **ACCESSO DIRETTO**

* **TAG: 16 bit** contengono l’indirizzo della memoria RAM mappato nella linea di cache
* **LINE: 11 bit** individuano la linea di cache
* **WORD: 3 bit.** Individua quale delle 8 WORD nel campo DATA contiene il dato che sta cercando
* **BYTE: 2bit.** Individua quale dei 4 Byte nella WORD si cerca

ESEMPIO DI CODIFICA

Indirizzi a 32 bit perciò spazio di indirizzamento totale 2^32 Bytes, visto che si tratta di un Byte addressing. = 4GB massimi

Ogni linea di Cache può contenere 2^16 blocchi di memoria diversi.

I blocchi di memoria sono da 32 celle.

Le locazioni di memoria contigue vengono salvate in linee di cache contigue, fino all’esaurimento delle linee, e si ricomincia da capo.

Fissato un numero di linea a 11 bit = 2^11 linee totali = 2048

ogni linea possiede 2 ^ 16 combinazioni diverse del campo TAG

Questa situazione genera dei conflitti perché è possibile che 2 indirizzi di memoria appartengano alla stessa linea ma possiedono campi TAG diversi, costringendo un reload della cache.

LINEA = 1024 = 10000000000

TAG = 1024 = 0000010000000000

WORD = 3 = 011

BYTE = 2 = 10

INDIRIZZO COMPLETO = TAG + LINEA + WORD + BYTE

= 0000010000000000 10000000000 011 10

gli indirizzi completi vanno a 0 a 4294967295 e corrispondono a 2^32 indirizzi totali.

La distanza tra due locazioni di memoria all’interno della stessa linea di cache è 65536 = 2^16 = 2^(LINEA+WORD+BYTE)

Aumentando progressivamente il campo TAG, che sono i 16 bit più significativi dell’indirizzo totale a 32 bit, si trovano le locazioni codificate e mappate nella linea di riferimento.

WORD seleziona quale delle 8 WORD = 32 BYTE del campo DATA andare a selezionare.

BYTE seleziona quale dei 4 Byte della WORD selezionata prima andare a leggere

100 01 = 2° byte della 5° WORD. 0 è il 1° indice.

Quando la CPU cerca un dato controlla la linea di cache. Se VAL = 0 Cache MISS, altrimenti controlla il campo LINEA.

Una volta trovata la linea di cache controlla i 2 campi TAG, quello della Cache e quello dell’indirizzo TOTALE. Se i campi corrispondono si avrà un HIT, altrimenti MISS.

**ESEMPIO**

**RAM 64KB indirizzabile a 1 BYTE  WORD = 1Byte. Dimensione di blocco 8Bytes. Si deve mappare la RAM in una cache di 256Bytes a indirizzamento diretto**

**64KB = 2^6 \* 2^10 Bytes = 2^16 Bytes Indirizzi di memoria a 16bits**

**I 16 bits devono essere divisi in 3 campi, TAG, LINE e WORD.**

**BLOCCO = 8 Byte = 2^3 Byte il campo WORD della linea sarà di 3 bit**

**N linee = Capacità Cache / Dimensione Blocco = 256Bytes / 8 Bytes = 32 = 2^5**

**Campo LINE = 5bit**

**Campo TAG = 16 – 5 – 3 = 8 bits**

* **CACHE ASSOCIATIVA A N VIE**

Riduce i conflitti dovuti a più TAG per stessa LINEA

Sfrutta anche la località temporale

Una linea di cache contiene n tabelle di dimensione uguale a quelle della Cache ad ACCESSO DIRETTO.

In caso di Cache MISS viene applicata una delle politiche di rimpiazzo per ricaricare i dati nella cache.

Una linea di Cache possiede N SET