Strutture di Memoria 1

Architettura degli Elaboratori e Laboratorio

17 Maggio 2013

Classificazione delle memorie

Funzionalitá:

- **Sola lettura** ROM, Read Only Memory, generalmente usata per contenere le routine di configurazione e controllo di basso livello (Firmware)
- **Lettura/Scrittura** tutti gli altri tipi di memorie

Tecnologia:

- **Elettronica** basata su semi-conduttori, RAM, ROM, Flash ...
- **Megnetica** basata su materiali polarizzabili, Dischi rigidi, Nastri
- **Ottica** CD

Modalitá di accesso:

- **Casuale** tempo di accesso indipendente dalla cella indirizzata, lo sono tutte le elettroniche
- **Sequenziale** l'accesso avviene scorrendo il supporto fino alla cella desiderata, nastro
- **Per Contenuto** CAM (Content Addressable Memory), risponde restituendo l'indirizzo che contiene il dato richiesto

Località dei dati

Generalmente, si considerano veri i due principi di località:

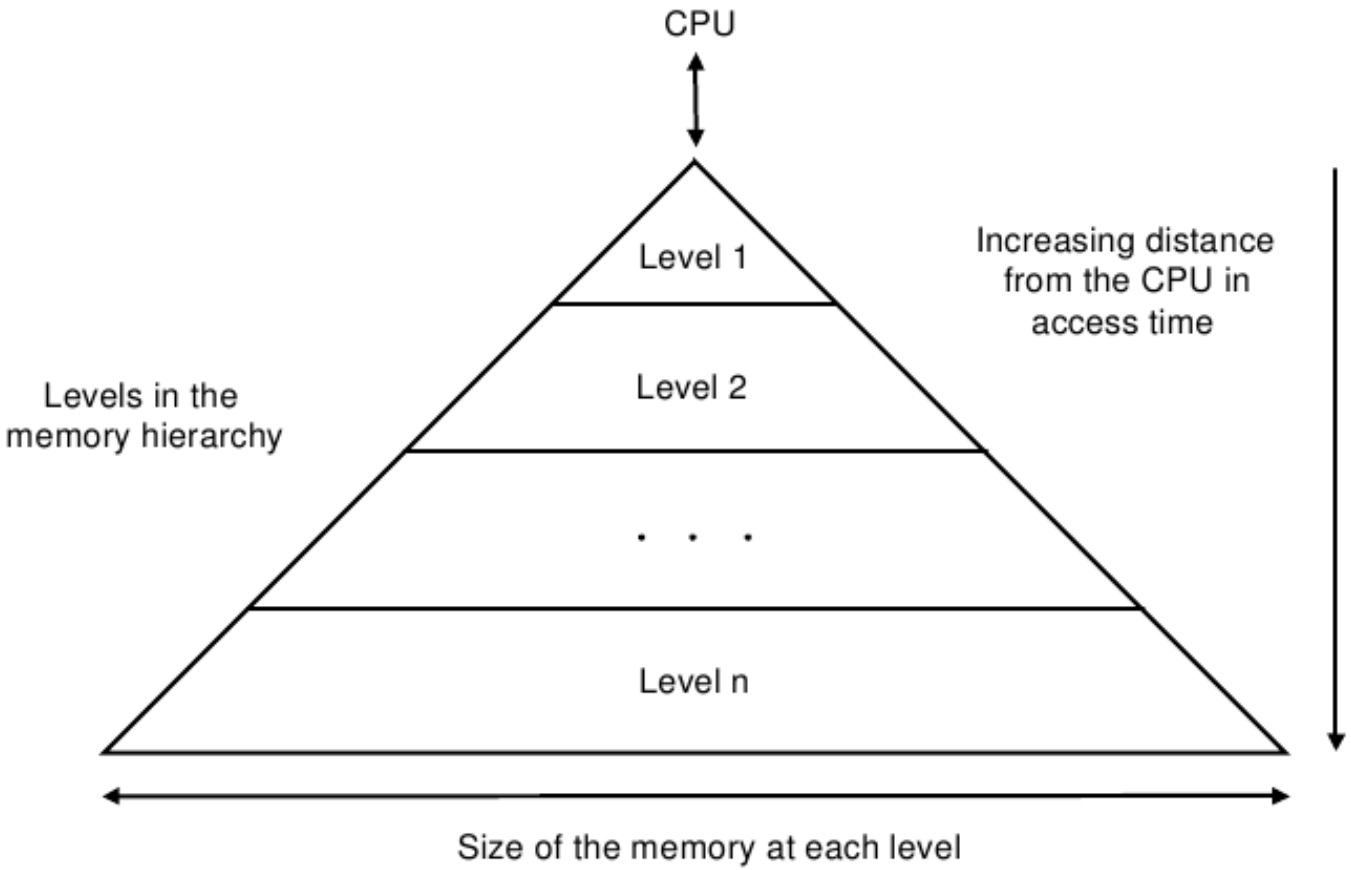
temporale Se è richiesto di accedere ad un dato, è probabile che debba accedervi nuovamente in breve tempo.

spaziale Se accedo ad un dato, è probabile che debba accedere anche a quelli memorizzati in indirizzi vicini.

- I principi sono espressi in termini di probabilità, non è un discorso deterministico!
- Simili ipotesi si possono fare per le istruzioni.

Questi principi sono alla base dell'introduzione delle gerarchie di memoria.

Gerarchie di memoria (I)



Gerarchie di memoria (I)

I vari tipi di memoria sono realizzati con tecnologie diverse:

- Costo per bit immagazzinato
- Tempo di accesso (o latenza)

Inoltre, si differenziano per:

- Memorie a semiconduttore con tecnologia VLSI (memoria cache e DRAM)
- Memorie ottiche o magnetiche (memoria secondaria)

Cache e località

L'algoritmo di caching mira a sfruttare i principi di località spaziale e temporale:

- mantenendo i dati richiesti recentemente "vicino" alla cpu (località temporale)
- muovendo blocchi contigui di memoria che contengono il dato richiesto (località spaziale)

Parametri di valutazione

Se il processore richiede alla cache una parola e questa è già presente nella cache si ha un hit (“colpito”).

Analogamente, se il processore richiede alla cache una parola e questa non è già presente nella cache si ha un miss (“mancato”).

Si possono quindi calcolare:

$$\text{Frequenza di hit} = \frac{\text{Accessi risolti CON SUCCESSO}}{\text{Numero totale di accessi}}$$

$$\text{Frequenza di miss} = \frac{\text{Accessi risolti SENZA SUCCESSO}}{\text{Numero totale di accessi}}$$

Tempo di hit Tempo necessario a prelevare il dato, comprende il tempo necessario a determinare se l’accesso è un hit o un miss.

Tempo di miss Tempo necessario a sostituire un dato con il blocco corrispondente nel livello inferiore (contenente il dato richiesto), più il tempo necessario per consegnare il dato al livello richiedente.

Cache

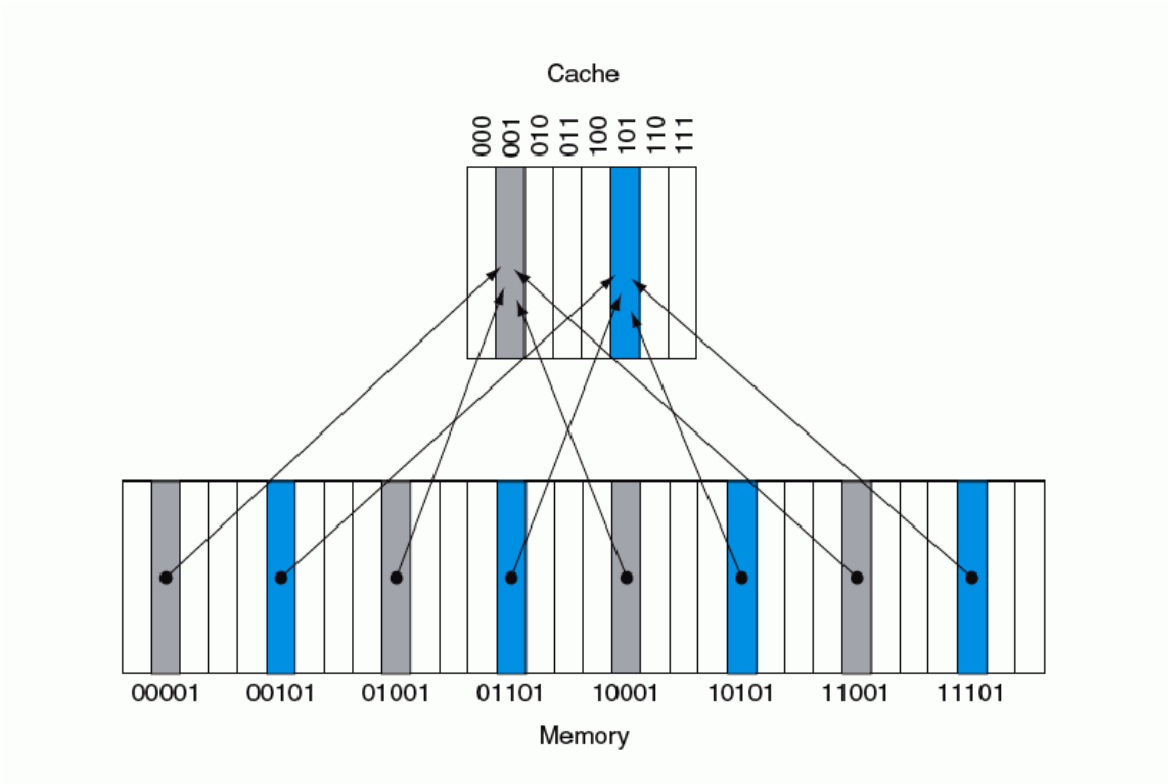
A fronte di una richiesta di dati da parte della CPU, la cache deve essere in grado di rispondere alle due domande:

1. Il dato è presente in memoria?
2. Se è presente, dove si trova?

Principali esempi di cache:

- Direct Mapped
- Fully Associative
- Set Associative

Direct Mapped Cache (I)



Cache direct mapped: considerando un blocco di dati, la sua posizione in cache è univocamente determinata dal suo indirizzo:

$$\text{pos} = \text{addr} \mod (\text{Nblocks})$$

$$1 \mod (8) = 1 \qquad 5 \mod (8) = 5 \qquad 9 \mod (8) = 1 \ldots$$

Direct Mapped Cache (II)

Abbiamo visto che, mediante i bit meno significativi dell'indirizzo del dato richiesto, possiamo risalire alla posizione del dato in cache.

Ora più blocchi sono destinati alla stessa posizione in cache, come riconoscere il dato richiesto?

Osserviamo di nuovo l'indirizzo del dato richiesto, i bit più alti vengono detti tag e utilizzati come identificativo.

Seguendo l'esempio precedente:

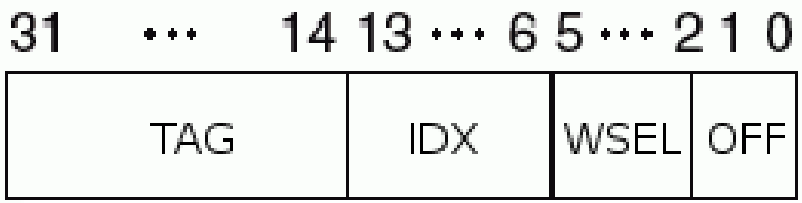
ADDR	→	TAG	IDX
00001	→	00	001
01001	→	01	001
10001	→	10	001
11001	→	11	001

Direct Mapped Cache (III)

Consideriamo una cache Direct Mapped, uno spazio di indirizzamento a 32 bit, parole di 32 bit:
Nell'indirizzo richiesto, identifichiamo:

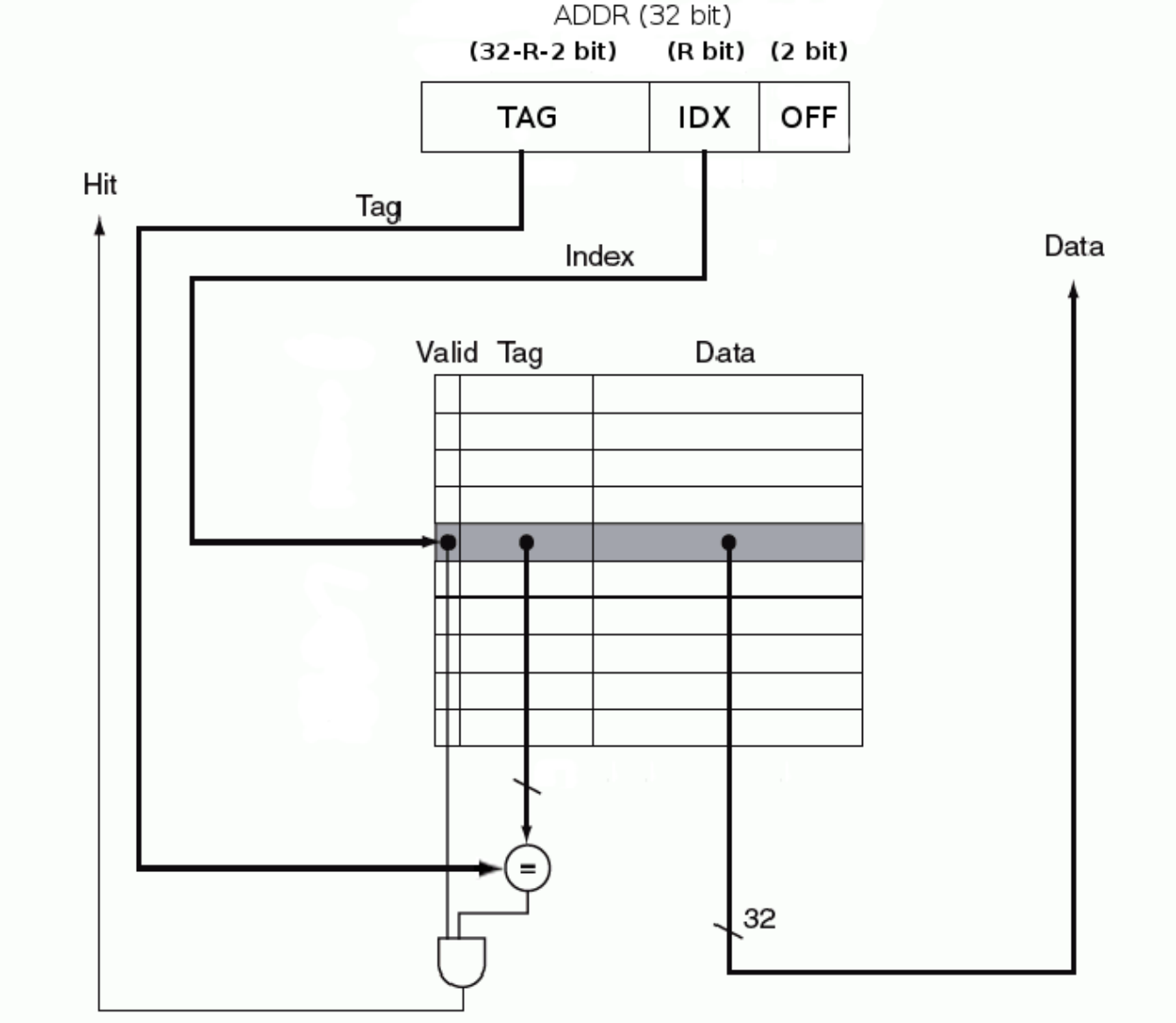
- OFFSET (2 bit) per selezionare il byte richiesto nella word
- WSEL (M bit) per selezionare la word nella riga
- IDX (R bit) per selezionare la riga nella cache
- TAG ($32 - 2 - R - M$ bit) per l' etichetta

Esempio:

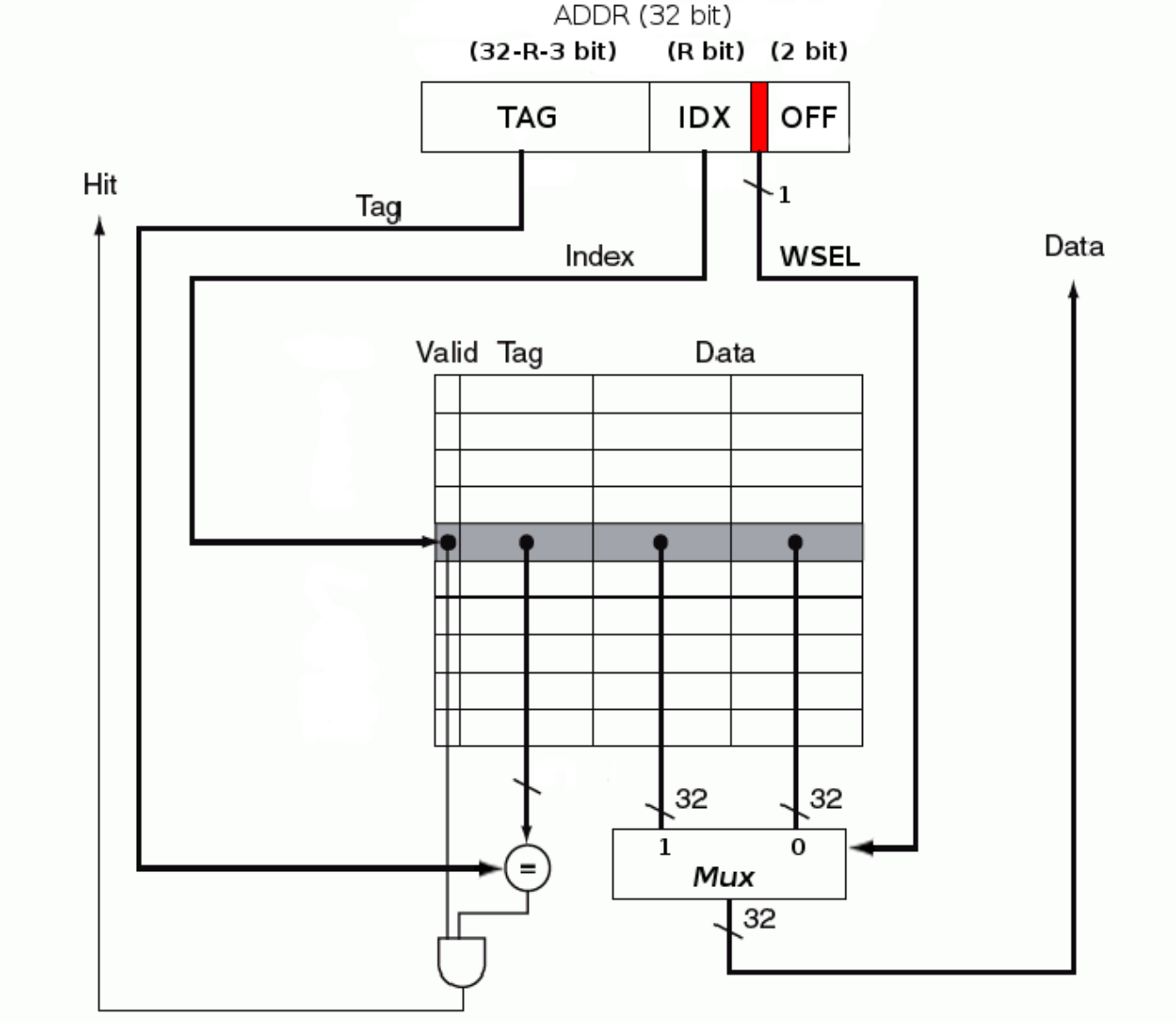


$M = 4$: 2^4 word per riga
$R = 8$: 2^8 righe
tag di 18 bit

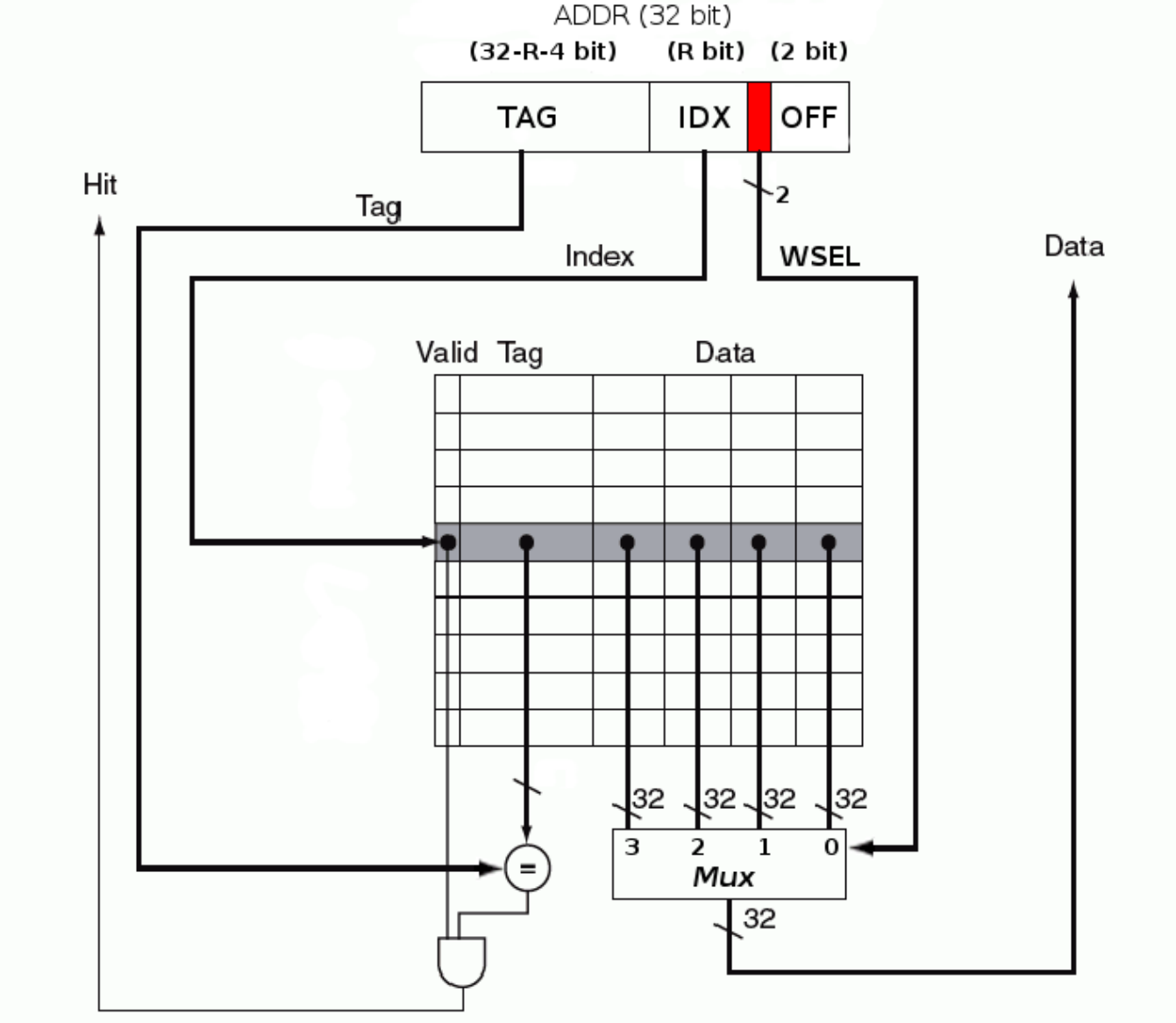
1-Word Direct Mapped Cache



2-Word Direct Mapped Cache



4-Word Direct Mapped Cache



Fully Associative Cache (I)

Ogni blocco può essere collocato in qualsiasi locazione della cache

Pro massima utilizzazione della cache

Contro per ricercare un blocco nella cache è necessario cercarlo in tutte le linee della cache.

- La ricerca sequenziale è troppo lenta
- Per velocizzare la ricerca è necessario effettuarla in parallelo, associando un comparatore a ciascuna posizione della cache. COSTO MOLTO ELEVATO

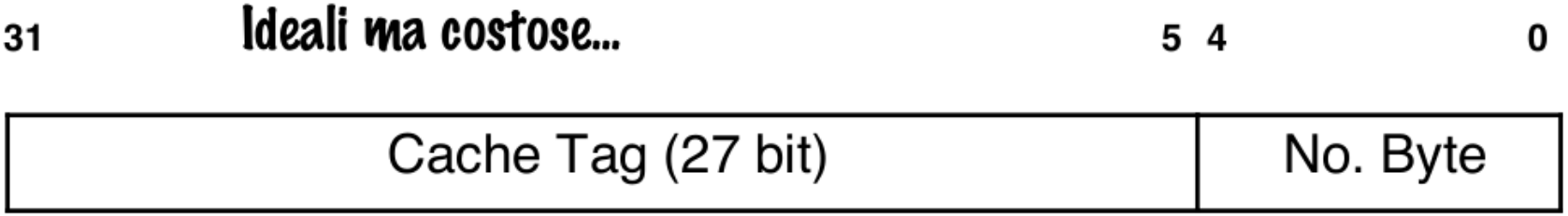
Fully Associative Cache (II)

Consideriamo una cache Fully Associative, uno spazio di indirizzamento a 32 bit, parole di 32 bit:
Nell'indirizzo richiesto, identifichiamo:

OFFSET (5 bit) per selezionare il byte richiesto nella word

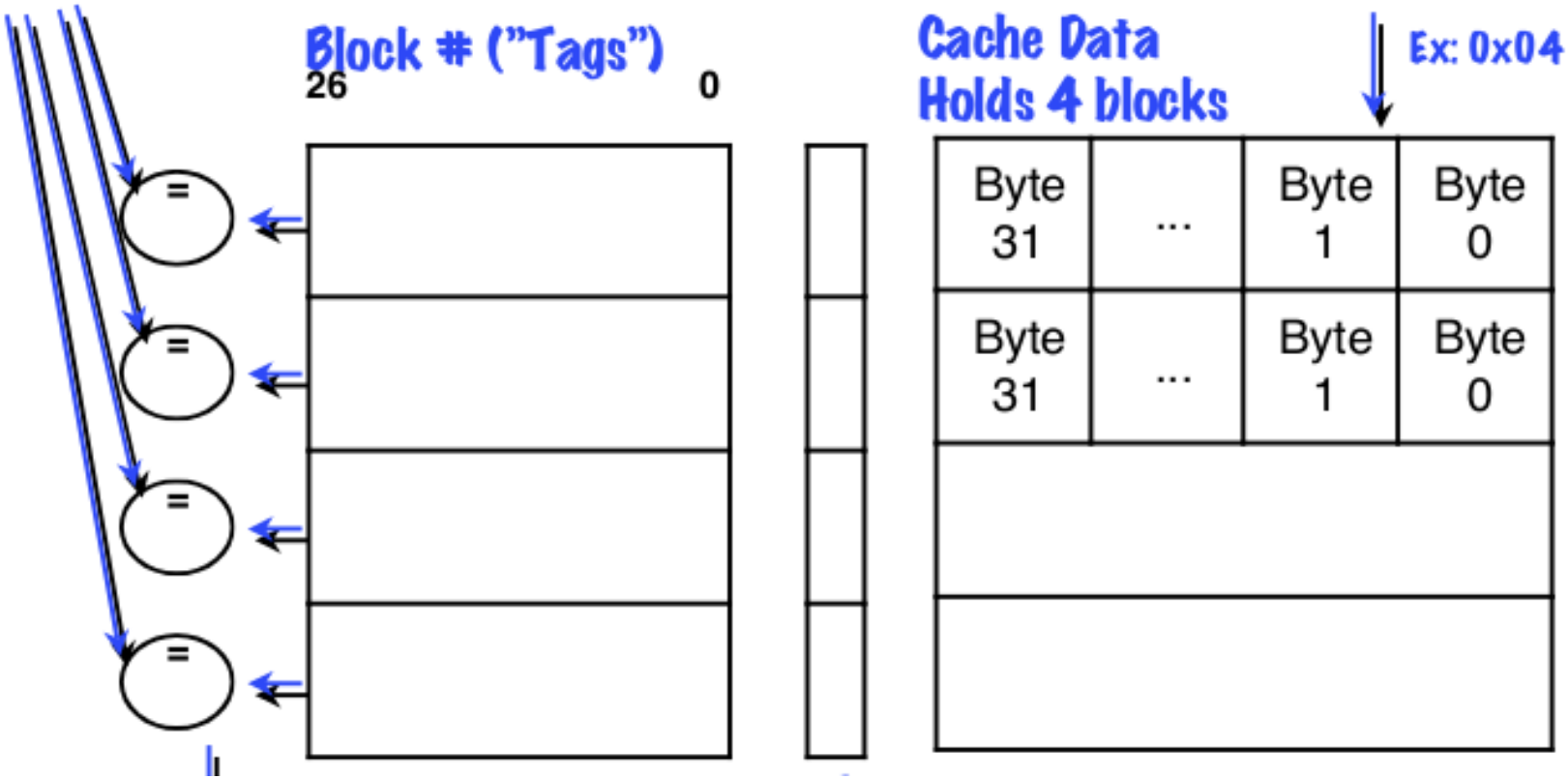
TAG ($32 - 5$ bit) per l' etichetta

Esempio:



$M = 5$: 2^5 word per riga

Fully Associative Cache (III)



Set Associative Cache (I)

- Nelle cache direct mapped a ciascun blocco della memoria corrisponde una specifica locazione nella cache.
- In una cache fully associative ogni blocco può essere collocato in qualsiasi locazione della cache.

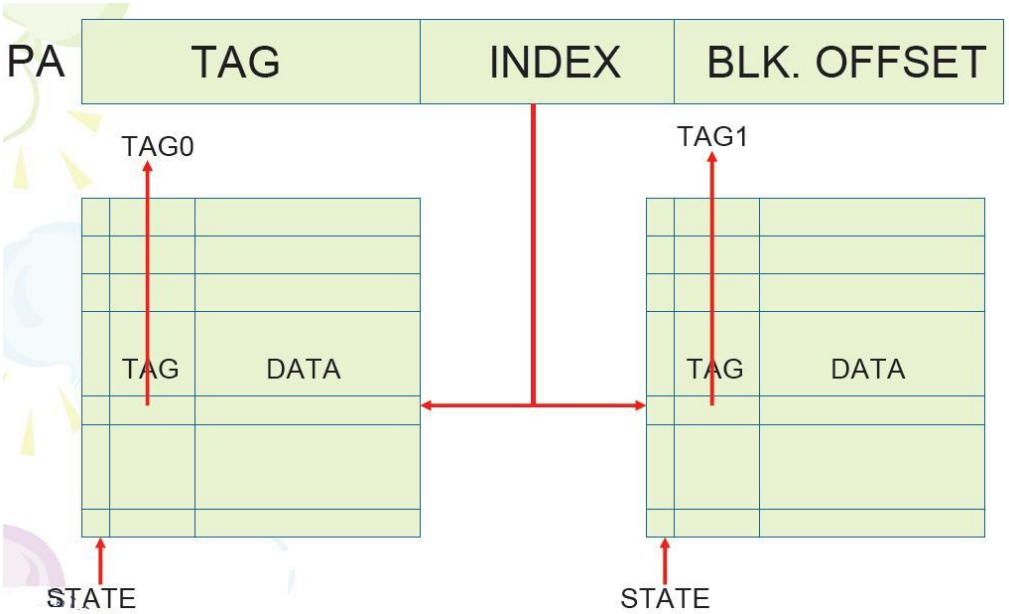
Un buon compromesso è costituito dalle cache set- associative (anche n-way set associative) : ciascun blocco di memoria ha a disposizione un numero fisso n (>=2) di locazioni in cache.
In questo caso la ricerca di un blocco richiede di confrontare il tag in n posizioni.
In questo modo si trae vantaggio della possibilità senza aumentare il costo di ricerca in modo inaccettabile

Set Associative Cache (II)

Consideriamo una cache 2-Ways Set Associative, uno spazio di indirizzamento a 32 bit, parole di 32 bit:
Nell'indirizzo richiesto, identifichiamo:

- OFFSET (4 bit) per selezionare il byte richiesto nella word
- IDX (6 bit) per selezionare la riga nel gruppo
- TAG ($32 - 6 - 4$ bit) per l' etichetta

Esempio:

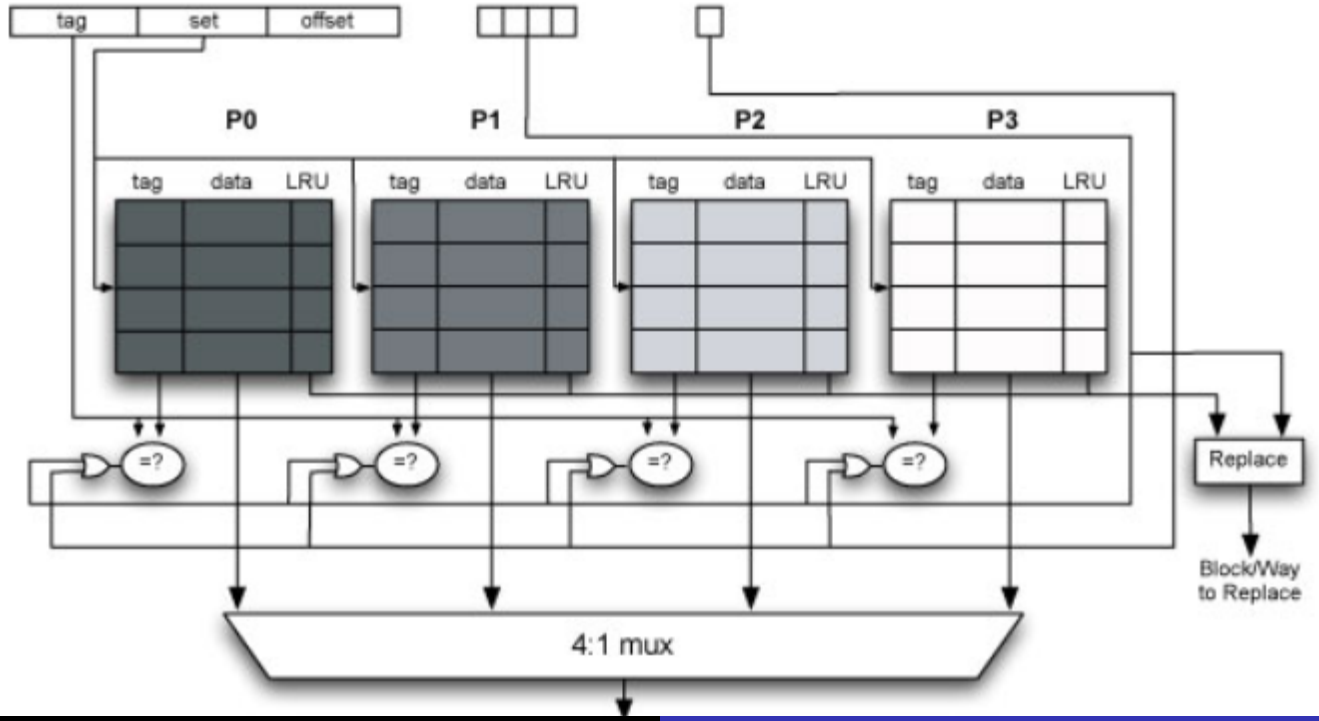


Set Associative Cache (II)

Consideriamo una cache 4-Ways Set Associative, uno spazio di indirizzamento a 32 bit, parole di 32 bit:
Nell'indirizzo richiesto, identifichiamo:

- OFFSET (4 bit) per selezionare il byte richiesto nella word
- IDX (6 bit) per selezionare la riga nel gruppo
- TAG ($32 - 6 - 4$ bit) per l' etichetta

Esempio:



Politiche di rimpiazzamento in cache

In caso di cache miss se non ci sono blocchi disponibili, quale blocco bisogna rimuovere dalla cache (per fare posto al blocco chiesto)?

LRU Least Recent Used, rimpiazza la linea non usata da piú tempo

RAND Random, rimpiazza una linea qualsiasi

Mediamente i due metodi si equivalgono in pretazioni.

Gestione delle scritture

In caso di cache hit:

write-through Le scritture vengono eseguite contemporaneamente sia in cache che in memoria principale, assicurando la consistenza dei dati.

write-back Le scritture aggiornano i dati solo nel blocco di cache aggiornando la memoria principale solo in caso di sovrascrittura di un dato modificato in precedenza.