

Cache

COSA È RICHIESTO SAPERE ALL'ESAME:

- **TIPI DI CACHE**
- **TIPI DI INDIRIZZAMENTO (DIRECT MAPPED ESERCIZI, I RESTANTI SOLO TEORIA)**
- **TECNICHE DI INDIRIZZAMENTO**
- **STRATEGIA DI AGGIORNAMENTO**

TEORIA PRELIMINARE

COS'È UNA CACHE

Una cache è una piccola area di memoria ad alta velocità che viene utilizzata per archiviare temporaneamente i dati più frequentemente utilizzati o richiesti da una CPU (unità di elaborazione centrale). La cache funziona come uno strato intermedio tra la CPU e la memoria principale, che solitamente è più grande ma più lenta. Il suo scopo principale è quello di ridurre il tempo di accesso ai dati, migliorando le prestazioni complessive del sistema.

PRINCIPIO DI LOCALITÀ SPAZIALE E TEMPORALE + LRU

Principio di località spaziale e temporale dei dati in una cache

Il principio di località spaziale e temporale è un concetto chiave su cui si basa l'accesso dei dati in cache da parte della CPU. Ci sono due tipi di principi:

1. Località spaziale

se la CPU accede a un dato particolare, è probabile che accederà anche a dati adiacenti nello spazio di indirizzamento, ovvero resterà in uno spazio locale. Sfruttando la località spaziale, la cache può memorizzare un blocco di memoria (ovvero una linea di dati) invece di un singolo dato, in modo che quando viene richiesto un dato, la cache ha già in memoria i dati adiacenti, riducendo così i tempi di accesso successivi.

2. Località temporale

se una CPU accede a un dato in un certo momento, è probabile che ci acceda nuovamente in un futuro vicino. Sfruttando la località temporale, una cache mantiene in memoria i dati recentemente utilizzati, in modo da poterli restituire rapidamente in caso di accesso successivo. Ciò riduce i tempi di accesso alla memoria principale.

L' LRU (Least Recently Used) è un algoritmo di gestione della cache che fa riferimento al principio di località temporale dei dati. L'obiettivo dell'algoritmo LRU è quello di identificare i dati meno recentemente utilizzati nella cache in modo da poterli sostituire con nuovi dati quando la cache è piena e si verifica un "miss" (ossia un fallimento nella ricerca dei dati richiesti nella cache).

Rimpiazzare intelligente non è facile perchè si dovrebbe prevedere il futuro e questo non si può fare; l'LRU, per rimpiazzare intelligentemente, guarda al passato: vede da quanto tempo quel dato viene chiesto oppure no e in base a quello sostuisce un dato piuttosto che un altro. l'LRU, infatti, è basato sull'assunzione che i dati utilizzati di recente abbiano una maggiore probabilità di essere richiamati di nuovo nel prossimo futuro. Pertanto, memorizzare i dati più frequentemente utilizzati e sostituire quelli meno recentemente utilizzati può aiutare a migliorare l'efficacia della cache nel soddisfare le richieste dei dati.

ALTRA DEFINIZIONE DI LRU:

LRU(Least Recently Used): uno schema di sostituzione in cui il blocco sostituito è quello rimasto inutilizzato per più tempo

Esistono due tipi di miss:

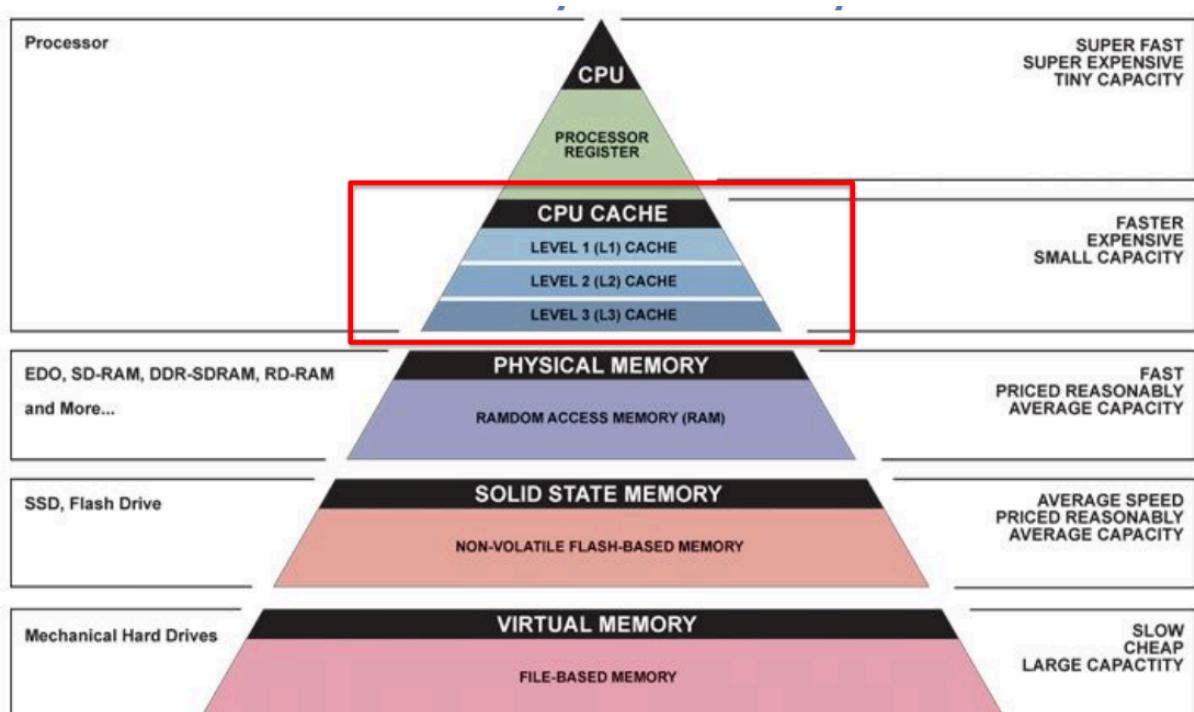
- Miss per "prima load" → la cache era vuota e ho dovuto fare una load
- Miss per "LRU" → la cache era piena ma non c'era il dato che cercavo (sovrascrivo il dato che cercavo con uno della cache; il dato sostituito nella cache è quello che non ho utilizzato per più tempo)

HIT: ho trovato nella cache il dato che stavo cercando

(Cache = memoria veloce e piccola; RAM = memoria (principale) più lenta ma più grande)

Durante l'esecuzione di un programma bisogna tenere in considerazione il fatto che esiste una "gerarchia di memoria" che stabilisce diverse probabilità sul fatto che si possa accedere ad un dato o no.

- Più la memoria è vicino alla CPU → veloce e piccola.
- Più ci si allontana dalla CPU → lenta e grande.



Le memorie principali in un computer sono:

- **Interna** → è dentro la CPU ed è costituita da registri
- **Centrale** → **RAM**
- **Secondarie** → alta capacità, bassi costi e non volatili

Tendenzialmente una memoria **cache** sta tra la memoria interna e quella centrale. Essa è trasparente (nascosta) al programmatore. Quello che fa è semplicemente copiare i dati che sono all'interno della RAM. Questo velocizza il

processo di acquisizione dei dati da parte della CPU in quanto non serve andare di nuovo in RAM per accedere ad un dato nuovo o magari appena usato. Ovviamente funziona solo se vengono implementate certe funzioni ...

Bisogna tenere in considerazione 2 principi, detti di località. Essi sfruttano la gerarchia della memoria per ottenere il massimo delle prestazioni.

- **Località Temporale:** quando si fa riferimento a un elemento c'è la tendenza a fare riferimento allo stesso elemento dopo poco tempo.
- **Località Spaziale:** quando si fa riferimento a un elemento c'è la tendenza a fare riferimento poco dopo ad altri elementi che hanno l'indirizzo vicino ad esso

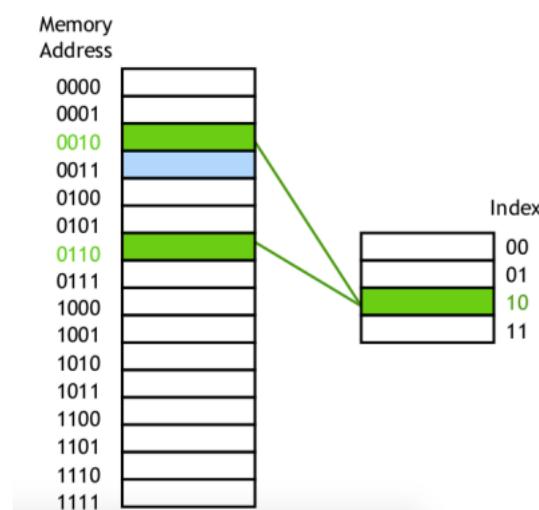
Questi principi, applicati ad un contesto reale come la cache, si traducono con:

- **Località Temporale:** mantenere i dati appena usati "vicino" alla CPU.
- **Località Spaziale:** muovere blocchi contigui di memoria che contendono il dato richiesto. Quindi già che c'è muove anche i blocchi vicini oltre a quello che gli serve.

Vanno introdotti ora i 3 principali tipi di cache:

DIRECT MAPPED

A ciascun blocco nella memoria centrale corrisponde una specifica locazione nella cache.



L'indirizzo di memoria è costituito da 4 elementi principali:

- TAG (Etichetta) → contiene informazioni per cercare la parola/word specifica. Si trova nei Bit più significativi.
- OFFSET → ulteriore criterio di ricerca, in caso ci siano più TAG uguali ne determina la scelta.
- INDICE → seleziona il blocco della cache.
- Bit di Validità →

Passaggi:

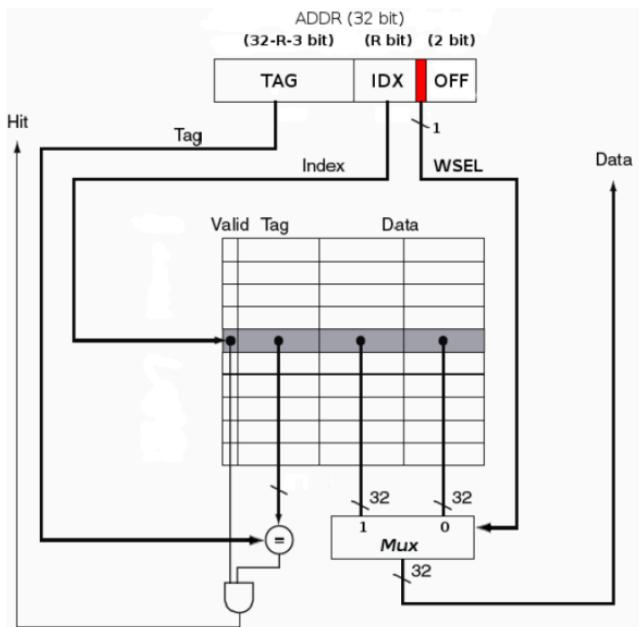
1. Uso dell'INDICE per trovare i blocchi di memoria (primo filtro)
2. Uso del TAG per selezionare un solo indirizzo
3. CONFRONTO DEI TAG
4. TAG AND Bit di validità → Genero un Hit oppure un Miss

In caso di MISS:

CPU va in stallo → MMU si occupa di reperire il dato → Caricamento in Cache → CPU riprende l'esecuzione

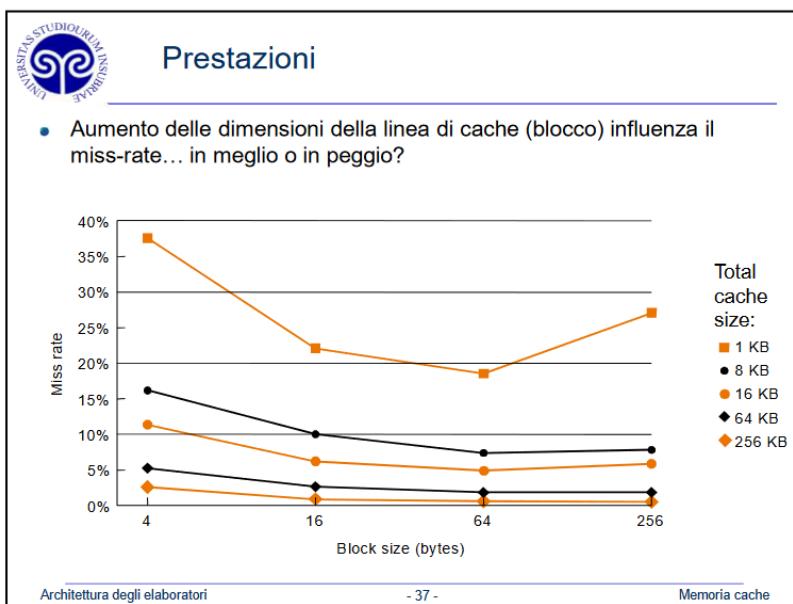
MMU = È un'altra memoria interna alla CPU. Funzione: Gestione delle cache: La MMU può interagire con la cache per ottimizzare l'accesso alla memoria.

2-WORD DIRECT MAPPED CACHE



Rapporto tra la dimensione della cache e i miss rate:

Un blocco molto grande permette di sfruttare la località spaziale.



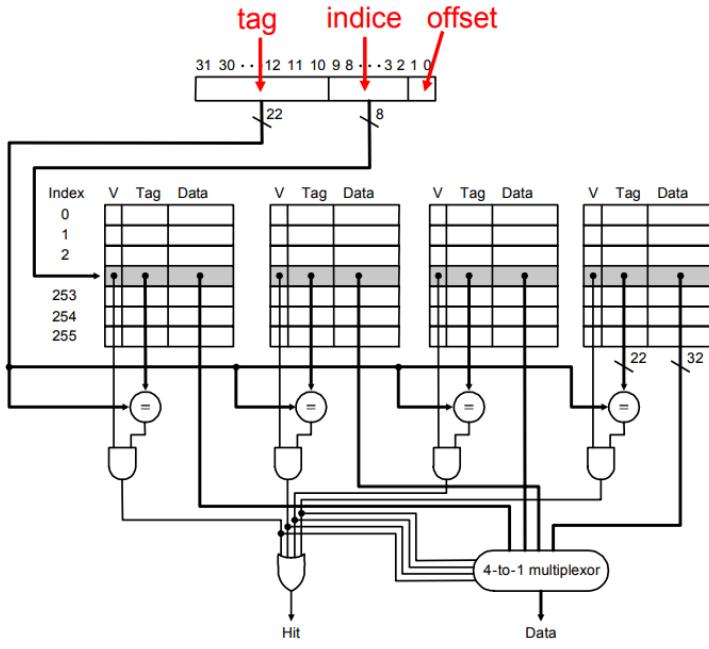
Aumento delle dimensioni della linea di cache (blocco) influenza il miss-rate... in meglio o in peggio?

- Dimensione della cache: diminuisce il miss rate, ma il beneficio diminuisce all'aumentare della dimensione aumenta il costo (ovviamente) e lo hit-time (memoria più grande)
- Dimensioni del blocco molto grandi = miss penalty, se è troppo grosso rispetto alla dimensione della cache allora anche + miss rate
- Indirizzamento diretto: semplice e veloce

- Indirizzamento associativo: costoso oppure lento

SET-ASSOCIATIVE

È una via di mezzo tra la Direct Mapped e la Fully Associative. Ciascun blocco della memoria ha a disposizione un numero fisso (≥ 2) di locazioni in cache. In generale, le cache set-associative sono un compromesso tra le cache completamente associative (massima flessibilità, ma più complesse) e le cache a mappatura diretta (più semplici, ma meno flessibili). La dimensione del set (il numero di vie all'interno di un insieme) può variare a seconda delle specifiche del sistema e delle esigenze di progettazione.



- I blocchi appartenenti alla cache sono raggruppati in SET.
- Ogni SET dunque riferenzia N blocchi in memoria.

Il tag viene confrontato con i tag delle quattro linee presenti nell'insieme corrispondente. Se il tag corrisponde a uno dei tag presenti, viene trovato un hit e i dati vengono restituiti dalla cache. Se il tag non corrisponde a nessuno dei tag presenti nell'insieme, si verifica un miss.

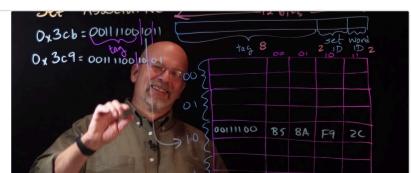
Tra i vantaggi della Set-Associative c'è il fatto che il TAG ha ora ben 22 bit con cui lavorare. Se si guarda la 2-Word Direct Mapped si può notare che il controllo dell'indice è fatto con 1 istruzione. Fin qui tutto apposto, ma poi quando si controlla il TAG esso viene fatto in contemporanea in modo da risparmiare molto tempo.

Inoltre + associatività significa una diminuzione del MISS-RATE

Ep 076: Set-Associative Caches

Set-associative caches blend the organizations of direct mapped and fully associative caches to reduce the consequences of those two architectures. The result is a cache with nearly the speed of a direct mapped cache while greatly reducing the risk of thrashing.

<https://youtu.be/gr5M9CULUzw>



STRATEGIE DI AGGIORNAMENTO

Richiede che la CPU venga fermata/stallo per aggiornare i valori. Ci sono vari metodi:

Write-through

I dati sono scritti nel blocco della cache e nel blocco del livello inferiore (= memoria; cache collegata alla memoria).

- SVANTAGGI: si fa contemporaneamente la scrittura in cache e in mem centrale, aumentato esponenzialmente il traffico nei bus.
- VANTAGGI: semplice da fare, coerenza dati, minore rischio perdita dati e maggiore affidabilità dei dati

Write Back

I dati sono scritti solo nel blocco della cache. Il blocco modificato viene scritto nel livello inferiore della gerarchia solo quando deve essere sostituito

- Svantaggi: tanto tempo, perdita di dati, complessiva hardware
- Vantaggi: Riduzione del traffico di scrittura, Riduzione della latenza delle scritture.

WRITE BUFFER CON WRITE THROUGH

Dati sia nella cache che nel write buffer. Gestito in modalità FIFO. Frequenza scrittura inferiore a qualcosa della DRAM, solo così è efficiente

write miss → scrivere in un indirizzo che non è ancora contenuto in cache.

- Le scritture possono indurre **write miss**
- Soluzioni possibili:
 - **Write allocate**: il blocco viene caricato in cache e si effettua la scrittura
 - **No-write allocate**: il blocco viene scritto direttamente nella memoria di livello inferiore, senza essere trasferito in cache

https://lpg.unibs.it/calca/Lucidi/L18_EserciziCache0809.pdf

ESERCIZI

ESERCIZI MISS E HIT

- Indirizzo di blocco = indirizzo parola / num word per blocco
- Blocco della Cache = Indirizzo di blocco % dim cache
- vedere se gli indirizzi sono al byte (fare * 4 alle word per blocco)

ALTRÉ FORMULE

- INDICE = log 2 di dim cache
- Indirizzi di parola = bit che compongono un blocco della cache (INDICE+TAG+WSEL)
- OFFSET/WSEL = (in byte)
- NUMERO MEDIO CICLI ACCESSO MEMORIA = (CPI* Hit Prob) + (Miss Prob * Miss Pen)
- CPI =
- Velocità Media (Istruz/sec) = Freq Clock / Num Med Cicli
- 50 ns = 5×10^{-8} secondi
- Miss prob lo ricavi dall'hit prob (100% - hit prob)

Domanda

224 di 271

Si supponga di scrivere dati in maniera sequenziale in un vettore posizionato in memoria da 0x00100000 a 0x0010007FF (estremi inclusi, indirizzamento al byte). Vi sia una cache a mappaggio diretto con 16 blocchi da 4 word; se la cache, inizialmente vuota, alloca sulla scrittura e adotta una politica di write-back, quanti write-back ci saranno dopo aver scritto tutto il vettore?

- 120
- 2048
- 112
- 512
- 8

- Converti l'indirizzo finale in decimale MA solo le ultime 3 cifre e sommo 1 → 0x7FF + 1 → 2048
- X = Indirizzo in Decimale / numero word in un blocco (ma in byte, quindi *4) → 2048 / (4*4) = 128
- X - dim cache → 128 - 16 = 112

ES. TEMA D'ESAME FERRETTI) NON VIENE SPECIFICATO NIENTE → per default è una cache a MAPPAGGIO DIRETTO

Considerando una memoria cache di 8K blocchi, e ogni blocco di 8 parole, si chiede di indicare quanti bit sono necessari per il TAG sapendo che un indirizzo di memoria è rappresentato su 32 bit

PER DEFAULT → INDIRIZZO DI BYTE

$K = \text{kilo} = (2^{10}) = 2048 \Rightarrow \text{n° blocchi} = 8K = 8 \cdot 2048 = 8192$

$8192 = 2^{13} \rightarrow 23 \text{ bit sono riservati all'Index}$

L'offset si calcola con il metodo di Ferretti

⇒ (2 word = 1 parola = 32 bit = 4 byte)

8 parole → $8 \cdot 4 \text{ byte} = 32 \text{ byte} = 2^5 \rightarrow \text{Offset} = 5$

Bit riservati al Tag = $32 - 13 - 5 = 14$

Considerando di avere 8 blocchi da 16 word, a che blocco viene mappato l'indirizzo 256?

X = INDIRIZZO CHE TI DANNO / word in 1 blocco (al byte) → 256/64 = 4

TAG = X % dim cache → 4 % 8 = 4

Considerando una memoria cache di 8k blocchi, e ogni blocco di 8 parole, si chiedere di indicare quanti bit sono necessari per il TAG sapendo che un indirizzo di memoria e' rappresentato su 32 bit?

Scegli un'alternativa:

- nessuna delle altre risposte è corretta
- 13
- 14
- non rispondo
- 18
- 32

- 8 WORD, quindi $8 \times 4 \rightarrow 32 \rightarrow 2^5 \rightarrow 5$ bit di OFFSET
- OPPURE c'è un altro metodo, 3 bit perchè $2^3=8$. Così trovi le word, poi devi trovare i byte, 1 parola è 4 byte, quindi servono 2^2 . Allora $2+3=5 \rightarrow$ OFFSET
- $\log 2$ di 8k = 13
- $32 - 5 - 13 = 14$

Una procedura impiega 100 millisecondi per essere eseguito su una macchina senza cache, e con un tempo di accesso alla memoria di 50ms. Si esegue il programma su una macchina con lo stesso tipo di memoria RAM e con una cache ad 1ms. Se "hit rate" durante l'esecuzione e' 0.8, qual'e' il tempo di esecuzione del programma?
(Il tempo dipende solo dagli accessi in memoria, trascurando istruzioni)

$$\text{ACCESSI} = \frac{T_{\text{PROGRAMMA}}}{T_{\text{ACCESO RAM (NO CACHE)}}} = \frac{0,1}{50 \cdot 10^{-3}} = 2000 \text{ ACC}$$

CALCOLA HIT

$$\text{ACCESSI} * \text{HIT RATE} = y$$

$$y * \text{TEMPO ACCESSO CACHE} = m$$

$$2000 \text{ ACC} * 0,95 = 1900 \text{ ACC}$$

$$1900 \text{ ACC} * 1 \text{ ms} = 1,9 \cdot 10^{-3}$$

CALCOLA MISS

$$\text{ACCESSI} * \text{MISS RATE} = h$$

$$h * \text{TEMPO ACCESSO RAM} = m$$

$$2000 \text{ ACC} * 0,05 = 100 \text{ ACC}$$

$$100 \text{ ACC} * 50 \text{ ms} = 0,005 \text{ ms}$$

$$1,9 \cdot 10^{-3} + 0,005 \text{ ms} = T \cdot 10^{-3} = T \text{ ms}$$

$$T = m + m$$

ESERCIZIO 1a

- In una cache direct-mapped con 32 blocchi (da 1 word), quali indirizzi di parole di memoria sono mappati nel blocco 13_{10} ?
 - Indirizzi di Parole di memoria = log 2 di **dim cache** $\rightarrow 2^5$
 - 5 sono i bit che servono per mappare tutte le parole di memoria

ESERCIZIO 1b

- L'indirizzo della parola di memoria 301_{10} fa parte degli indirizzi mappati nel blocco 13_{10} di una cache direct-mapped di 32 blocchi (da 1 word) e di una cache direct-mapped di 64 blocchi (di una word)?
- Poichè $301 = 256 + 0 + 0 + 32 + 0 + 8 + 4 + 0 + 1$
la rappresentazione binaria di 301_{10} è $(100101101)_2$

Tale indirizzo fa quindi parte degli indirizzi mappati nel blocco 13_{10} in una cache direct-mapped a 32 blocchi (avendo gli ultimi 5 bit uguali a 01101)

NON fa invece parte degli indirizzi mappati nel blocco 13_{10} in una cache direct-mapped a 64 blocchi (non avendo gli ultimi 6 bit uguali a 001101)

INDICE = Log 2 di dim cache

ESERCIZIO 1c (facile)

- Qual è l'indirizzo della word contenuta nel blocco con indice IND e con etichetta TAG, se la cache è a x blocchi da 1 word?
- Ad esempio, se la cache ha 8 blocchi
 - quali sono indice e etichetta dell'indirizzo 19_{10} ?
 - a quale indirizzo corrisponde il blocco con TAG=11 e IND=011?

ESERCIZIO 2

- Consideriamo una cache direct-mapped a 8 blocchi da 1 word, inizialmente vuota
- Data la sequenza di accessi alle word con indirizzo 22, 26, 22, 26, 16, 3, 16, 18, 16
- Qual è la sequenza di hit/miss?
- Come cambia il contenuto della cache ad ogni miss?

- $8 = 2^3$ e blocchi da una word
 → 3 bit low order dell'indirizzo, corrispondono al numero del blocco (INDEX)
 → restanti bit alti dell'indirizzo, corrispondono al TAG (etichetta) del blocco

22, 26, 22, 26, 16, 3, 16, 18, 16

- Riferimento a word con indirizzo 18 (10010) → miss



Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	N		
010	Y	11 _{two}	Memory (11010 _{two})
011	N		
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

d. After handling a miss of address (10000_{two})

Index	V	Tag	Data
000	Y	10 _{two}	Memory (10000 _{two})
001	N		
010	Y	10 _{two}	Memory (10010 _{two})
011	Y	00 _{two}	Memory (00011 _{two})
100	N		
101	N		
110	Y	10 _{two}	Memory (10110 _{two})
111	N		

f. After handling a miss of address (10010_{two})

Ricordarsi solo che si guarda prima l'INDICE, se è già occupato si sostituisce (scrivendo MISS) e poi, dato che è una cache ad 1 word il TAG non serve a un cazzo

ESERCIZIO 3

- In una cache direct-mapped in cui ogni blocco contiene **due word**, qual è il valore del TAG (etichetta) per la parola di memoria con indirizzo 301_{10} ?

Esercizio 3 – soluzione

- TAG = parte alta dell'indirizzo (sequenza di bit non usati nell'indice del blocco)
- Per conoscere il TAG serve quindi conoscere il numero di bit usati per l'indice (quest'ultimo ricavabile dal numero di blocchi della cache, oppure dalla dimensione totale della cache e della dimensione di ogni blocco)
- Se ad esempio sono usati per l'indice 4 bit (informazione non disponibile in questo esercizio)

$$301 = 256 + 0 + 0 + 32 + 0 + 8 + 4 + 0 + 1 = (1001\ 0110\ 1)_2$$

il valore del TAG è 1001, l'indice è 0110 e il bit meno significativo è usato per individuare la parola

indirizzo 32 bit	il resto boh?	4 bit TAG	1 bit: WSEL
------------------	---------------	-----------	-------------

Ricordarsi che c'è sempre 1 bit per il WSEL (Sceglie quale word va usata)

ESERCIZIO 4a

- Sia data una cache direct-mapped con **blocchi di 4 parole** e una dimensione totale di 32 parole
- Assumendo **indirizzi (di parola) a 16 bit**, determinare il numero di bit riservati all'etichetta, all'indice di blocco e alla parola

Dim Cache = Dimensioni totale RAM / num word per blocco

32 parole / 4 parole = 8 = 2^3 → da notare che sono tutte in word, se fossero in byte, avremmo dovuto cambiare tutto in byte

A quel punto calcolo INDICE che è 3

indirizzo 16 bit: etichetta 11, indice 3, parola 2 (perchè sono 4 word e 2 in binario può rappresentare fino a 4)

ESERCIZIO 4b

Esercizio 4b

- Sia data una cache direct-mapped con **blocchi di 4 parole** e una dimensione totale di **32 parole** e la seguente sequenza di **indirizzi di parola** a cui si intende accedere:

1, 4, 8, 5, 33, 66, 32, 56, 9, 11, 4, 43, 88, 6, 32

- Determinare il numero di miss e hit alla cache, assumendo che la cache sia inizialmente vuota

Dim Cache = $32/4 = 8$ quindi la cache è da 8 blocchi

Troviamo poi l'*indirizzo di blocco (l'indice)* = *indirizzo parola/ampiezza di blocco* \rightarrow prendi solo parte intera

indirizzo parola	1	4	8	5	33	66	32
indirizzo di blocco	0	1	2	1	8	16	8
blocco di cache	0	1	2	1	0	0	0

Poi calcoliamo il blocco della cache in cui starà: *indirizzo di blocco % dimensione cache*

Ora possiamo simulare l'accesso alla memoria.

Blocco	1	2	3	4	5	6	7	8
0	mem[0]	mem[0]	mem[0]	mem[0]	mem[8]	mem[16]	mem[8]	mem[8]
1		mem[1]	mem[1]	mem[1]	mem[1]	mem[1]	mem[1]	mem[1]
2			mem[2]	mem[2]	mem[2]	mem[2]	mem[2]	mem[2]
3								
4								
5								
6								mem[14]
7								

miss miss miss hit miss miss miss miss

Blocco	9	10	11	12	13	14	15
0	mem[8]	mem[8]	mem[8]	mem[8]	mem[8]	mem[8]	mem[8]
1	mem[1]	mem[1]	mem[1]	mem[1]	mem[1]	mem[1]	mem[1]
2	mem[2]	mem[2]	mem[2]	mem[10]	mem[10]	mem[10]	mem[2]
3							
4							
5							
6	mem[14]	mem[14]	mem[14]	mem[14]	mem[22]	mem[22]	mem[22]
7							

hit hit hit miss miss hit hit

ESERCIZIO 5

- Si supponga di avere una cache a mappaggio diretto con 4 blocchi da 2 word
- Se la cache parte vuota e si esegue 10 volte un loop che accede alle word che si trovano agli indirizzi 0, 1, 2, 3, 4, quanti miss ci saranno ?

indirizzo = 0, 1, 2, 3, 4

$$\text{Indirizzo di blocco} = \frac{\text{indirizzo parola}}{\text{word all'interno di un blocco}}$$

indirizzo parola / word all'interno di un blocco

Blocco della cache = indirizzo di blocco % dim cache (numero blocchi nella cache)

Indirizzi di blocco	0	0	1	1	2
Blocco della cache	0	0	1	1	2

cache	1	2	3	4	5	6	7
0	mem[0]	mem[0]	mem[0]	mem[0]	mem[0]	mem[0]	mem[0]
1			mem[1]	mem[1]	mem[1]	mem[1]	mem[1]
2					mem[2]	mem[2]	mem[2]
HIT/MISS	miss	hit	miss	hit	miss	hit	hit

ESERCIZIO 6

- Considerando indirizzi di memoria a 32 bit per una cache direct-mapped con dimensione di ogni blocco di 4 parole e dimensione totale di **64 KiB**, qual è il numero di bit per i TAG?
- N.B. Se non è specificato diversamente, si assume **indirizzo al byte**

Visto che **stiamo usando i byte** dobbiamo trasformare tutto il resto in byte. Quindi:

Dim totale 64 KiB = $64 * 2^{10}$ Dim blocco = 4 word = 16 byte

Ricaviamo il numero dei blocchi dividendo dimensione totale in **byte**/ numero **byte** di ogni blocco.

$$\frac{64 \cdot 2^{10}}{16} = 4096 = 2^{12} \text{ blocchi} \rightarrow 12 \text{ è l'indice}$$

indirizzo 32 bit	tag 16 bit	indice 12 bit	4 bit
------------------	------------	---------------	-------

4 bit perchè: 2 come WSEL e 2 per individuare per individuare il byte all'interno della parola (forse perchè l'indirizzo è al byte?)

ESERCIZIO 7

BOH

ESERCIZIO 10

- Si considerino due cache, C1 e C2, costituite entrambe da **4 blocchi di due word**. La **cache C1 è direct-mapped**, la cache **C2 è fully-associative con politica di rimpiazzamento LRU**
- Determinare, nei due casi, il numero di miss per la seguente sequenza di operazioni di lettura di una word agli indirizzi:
508, 1016, 510, 24, 540, 1050, 1020, 24, 538
- Gli indirizzi sono al **byte**, espressi in **decimale**
- Si assuma che all'inizio le cache siano vuote
- Spiegare il ragionamento che ha portato alla risposta

- C1 → essendo che gli indirizzi sono al byte, dobbiamo anche moltiplicare per 4 il numero di word per blocco

Essendo i blocchi da 2 word e gli indirizzi al byte, dividiamo tutti gli indirizzi per 8 (e consideriamo la parte intera del risultato), dopodichè calcoliamo il blocco in cui è mappato l'indirizzo richiesto attraverso l'operazione di modulo 4 (numero dei blocchi di cache)

- $508/8 = 63$ $63 \% 4 = 3 \rightarrow$ miss
- $1016/8 = 127$ $127 \% 4 = 3 \rightarrow$ miss
- $510/8 = 63$ $63 \% 4 = 3 \rightarrow$ miss
- $24/8 = 3$ $3 \% 4 = 3 \rightarrow$ miss
- $540/8 = 67$ $67 \% 4 = 3 \rightarrow$ miss
- $1050/8 = 131$ $131 \% 4 = 3 \rightarrow$ miss
- $1020/8 = 127$ $127 \% 4 = 3 \rightarrow$ miss
- $24/8 = 3$ $3 \% 4 = 3 \rightarrow$ miss
- $538/8 = 67$ $67 \% 4 = 3 \rightarrow$ miss
- Si sarebbe potuto concludere che si hanno solo miss, osservando che tutti gli indirizzi sono mappati all'indice 3 e che non ci sono due accessi consecutivi che richiedono lo stesso indirizzo

- C2 → Essendo i blocchi mappati a caso, non usiamo neanche i blocchi della cache. Inoltre usiamo politica di LRU, dunque dato che la cache è grossa 4 blocchi allora appena c'è una MISS, LRU vede il blocco utilizzato da meno tempo e lo rimpiazza (in questo caso 127 viene rimpiazzato con 131 e **successivamente viene rimpiazzato da 127**).

• $508/8 = 63$	miss	($63 \rightarrow 0$)
• $1016/8 = 127$	miss	($127 \rightarrow 1$)
• $510/8 = 63$	hit	$\rightarrow 0$
• $24/8 = 3$	miss	($3 \rightarrow 2$)
• $540/8 = 67$	miss	($67 \rightarrow 3$)
• $1050/8 = 131$	miss	($131 \rightarrow 1$) LRU
• $1020/8 = 127$	miss	($127 \rightarrow 0$) LRU
• $24/8 = 3$	hit	
• $538/8 = 67$	hit	

63	63 +vecchio	127
127 +vecchio	131	131
3	3	3
67	67	67

Si supponga di avere una cache two-way set-associative, composta da 4 blocchi da una word. Quanti miss occorreranno nel leggere in sequenza gli indirizzi 0, 8, 0, 6 e 8?

Scegli un'alternativa:

- a. 3
 b. 2
 c. 5
 d. 4
 e. 1

0	0	0000000 00 0
	8	0000001 00 0
1	6	0000000 11 0
2		
3		