


La struttura delle memorie

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimento Patterson v.5: 2.11 – 5.2, 5.5, 5.12, B8, B9.


Sommario

DRAM

Codici di errore

Esempi

Miss penalty: esempio

Tempi di accesso:

1 ciclo di clock per inviare l'indirizzo.

15 cicli di clock per ciascuna attivazione della Memoria (lettura di parola, dettata dal cammino critico in lettura).

1 ciclo di clock per trasferire una parola al livello superiore (cache).

Blocco di cache di 4 parole, blocco di memoria (cache secondaria) di 1 parola

⇓

Miss_Penalty = 1 + 15 * 4 *(parole)* + 1 * 4 *(parole)* = *65 cicli_clock*

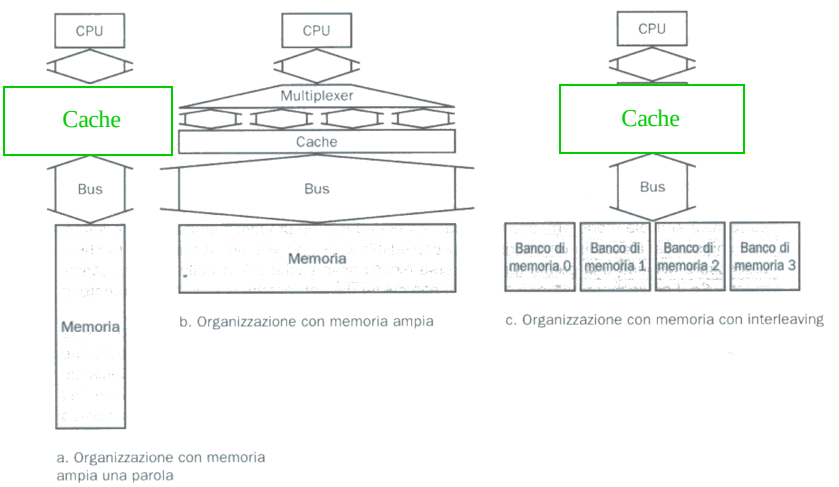
#byte / ciclo_clock = 4(*parole*) * 4(*byte/parola*) / 65(*cicli_clock*) ≅ 0,25(*byte / ciclo_clock*)

Obbiettivi:

•Diminuire la penalità di fallimento (miss_penalty).

•Diminuire il tasso di fallimento (miss_rate).

Riduzione del miss penalty



a. Organizzazione con memoria ampia una parola

b. Organizzazione con memoria ampia

c. Organizzazione con memoria con interleaving

Interleaving (interallacciamento). Banchi che potrebbero essere trasferiti in parallelo alla cache.

Valutazione della riduzione della "miss_penalty"

Architettura standard: penalità di miss è di 65 cicli_clock.

Maggiore ampiezza della memoria:
- Organizzazione della Memoria Principale per blocchi.
- Bus più ampio (bus dati largo un blocco, 4 parole).
- Per blocchi di Memoria di 4 parole, blocchi di cache di 4 parole:

  Miss_penalty = 1 + 15 * 1 + 1 * 1 = 17 cicli_clock.

  #byte / ciclo_clock = 4(*parole*) * 4(*byte/parola*) / 16(*cicli_clock*) = 0,94(*byte / ciclo_clock*)

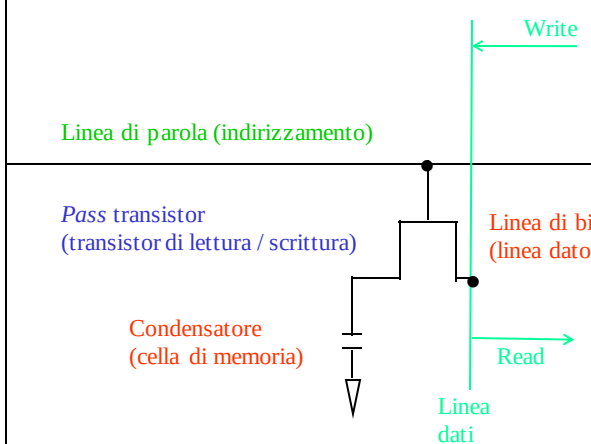
Interleaving:
- Organizzazione della Memoria Principale per banchi con accesso indipendente alla memoria (interleaving).
- Bus standard (trasferimento di 1 parola alla volta).
- Per blocchi di Memoria di 1 parola, blocchi di cache di 4 parole:

  Miss_penalty = 1 + 15 * 1 + 1*4 = 20 cicli_clock.

  #byte / ciclo_clock = 4(*parole*) * 4(*byte/parola*) / 20(*cicli_clock*) = 0,80(*byte / ciclo_clock*)

Memorie DRAM - scrittura

Dynamic RAM. Condensatore che viene caricato. ∀bit (1 condensatore contro 4-6 SRAM).



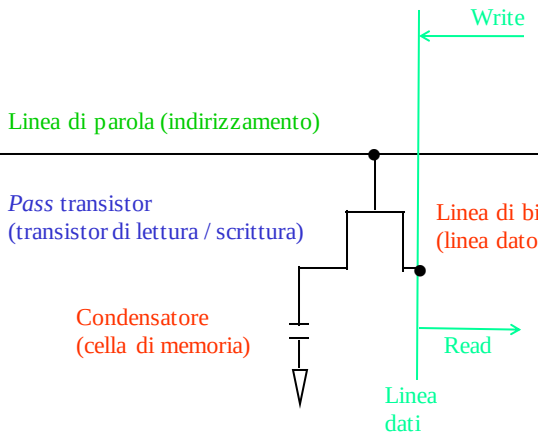
Scrittura: carica / scarica

La scrittura carica (memoria = 1) o scarica (memoria = 0) il condensatore.

1 *Pass* transistor + 1 condensatore.

.

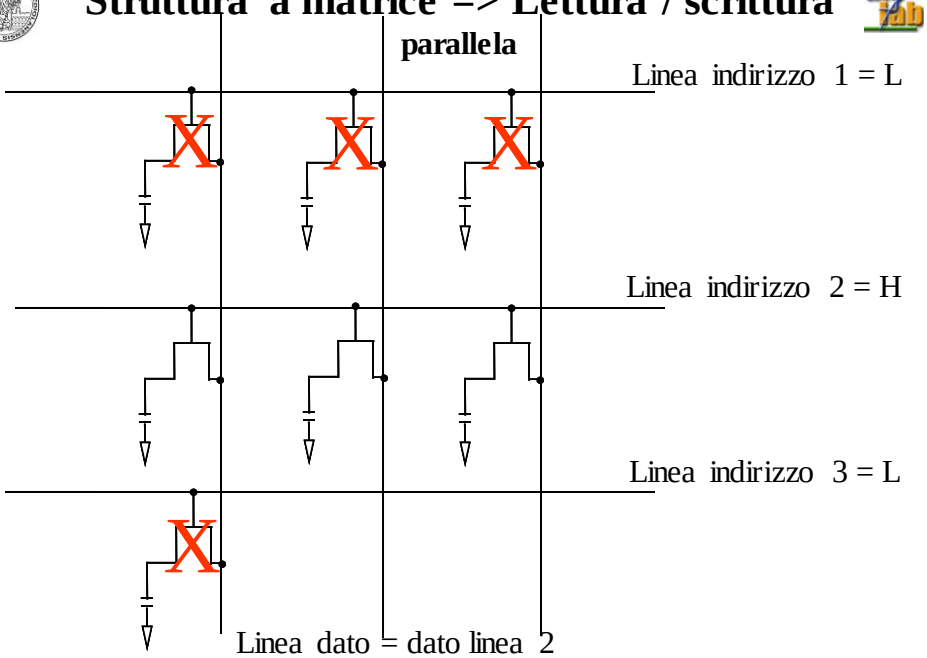
Memorie DRAM - lettura



Lettura: la linea di bit è portata ad una tensione intermedia, e poi tirata verso low o high a seconda che il condensatore sia carico / scarico: amplificazione della carica.

1 *Pass* transistor + 1 condensatore.
La lettura scarica la memoria che deve essere ricaricata.

Struttura a matrice => Lettura / scrittura



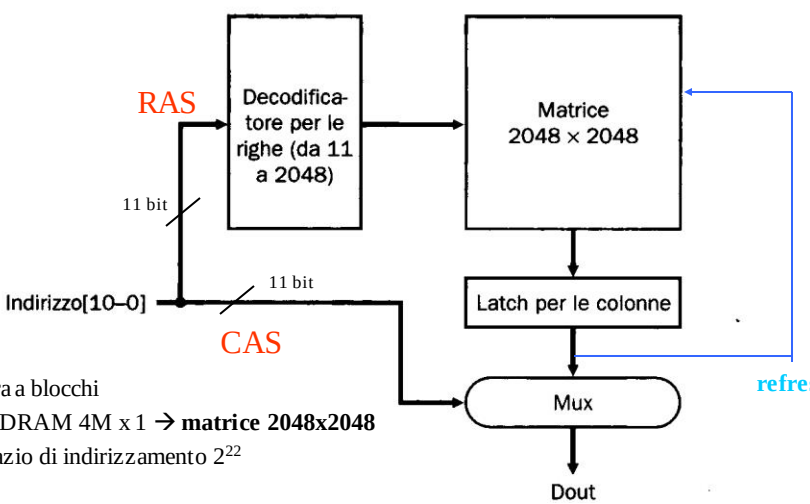
I problemi delle DRAM

I condensatori …. si scaricano (qualche millisecondo)

refresh gestito autonomamente dal controllore della memoria mediante ciclo lettura/scrittura

Cosa leggo/scrivo? Quale/i bit?

Struttura a 2 livelli (matrice) di una DRAM



- Struttura a blocchi
  - es. DRAM 4M x 1 → **matrice 2048x2048**
  - Spazio di indirizzamento 2^{22}

•Accesso:
selezione riga (RAS) + selezione colonna (CAS)
Efficiente per il refresh (refresh di riga) – (35-70ms, tempo di carica dei condensatori).
Utilizzo per la Memoria Principale.

Specifiche delle DRAM (Patterson and Hennessy, 2008)

Year introduced	Chip size	$ per GB	Total access time to a new row/column	Column access time to existing row
1980	64 Kbit	$1,500,000	250 ns	150 ns
1983	256 Kbit	$500,000	185 ns	100 ns
1985	1 Mbit	$200,000	135 ns	40 ns
1989	4 Mbit	$50,000	110 ns	40 ns
1992	16 Mbit	$15,000	90 ns	30 ns
1996	64 Mbit	$10,000	60 ns	12 ns
1998	128 Mbit	$4,000	60 ns	10 ns
2000	256 Mbit	$1,000	55 ns	7 ns
2004	512 Mbit	$250	50 ns	5 ns
2007	1 Gbit	$50	40 ns	1.25 ns

Tempo di accesso della cache primaria = 0,16ns
Tempo di accesso della cache secondaria = 0.3ns

Evoluzione delle SRAM e DRAM

Synchronous version.

Trasferimento ***a burst*** o a pagina: trasferimento consecutivo di parole ad indirizzi consecutivi (e.g. RAM EDO).

SDRAM (Synchronous DRAM)

La fase di indirizzamento e di recupero dei dati vengono separate in modo da ridurre al minimo l'impatto della latenza.

Tra l'indirizzamento ed il recupero dei dati, il processore puo' eseguire altri compiti (NB il processore puo' essere la CPU o il controllore della memoria, o altro: il dispositivo che controlla la memoria).

DDR-SDRAM. Riescono a trasferire 2 bit per ciclo di clock. Frequenza doppia rispetto alla frequenza del clock del bus.

Prestazione di una memoria

Tempo di accesso.

Per una memoria a random-access, è il tempo richiesto per eseguire un'operazione di lettura / scrittura. Più precisamente è il tempo che intercorre dall'istante in cui l'indirizzo si presenta alla porta di lettura della memoria all'istante in cui il dato diventa disponibile.

Per una memoria non random-access, è il tempo richiesto per posizionare il dispositivo di lettura / scrittura in corrispondenza dell'informazione da leggere / scrivere.

Memory cycle time: si applica ad una memoria random-access. E' il tempo di accesso più il tempo necessario perchè possa avvenire un secondo accesso a memoria.

Transfer rate: quantità di informazione trasferita nell'unità di tempo si misura in:

Numero_dati (in byte) / s.

Organizzazione della memoria

Organizzazione a matrice (cf. cache)

Organizzazione gerarchica.

Sommario

DRAM.

Codici di errore

Esempi

Errori nella memoria

Mean Time To Failure (MTTF) di un disco arriva a 1.000.000 di ore = 114 anni.

Un grosso centro di calcolo può contenere 50.000 server, ciascuno contiene facilmente 2 dischi per un totale di 100.000 dischi.

Numero di ore in un anno è: 24 x 365 = 8.760 ore / anno
MTTF = 1.000.000 di ore corrisponde a un Annual Failure Rate (AFR) =
8.760 / 1.000.000 = 0,876%

Un AFR di 0,876% corrisponde per 100.000 dischi a una media di 876 guasti per anno, cioè più di 2 guasti al giorno!

Da qui la ragione di proteggere i dati.

Per i dati in Memoria a semiconduttore si usano i codici di parità.

ECC

- Errori dovuti a malfunzionamenti HW o SW.
  - Date le dimensioni delle memorie (**10^{10} celle**) la probabilità d'errore non è più trascurabile.
  - Per applicazioni sensibili, è di fondamentale importanza gestirli.
- Codici rivelatori d'errore
  - Es: codice di parità.
  - Consente di individuare errori singoli in una parola.
  - Non consente di individuare su quale bit si e' verificato l'errore.
- Codici correttori d'errore (error-correcting codes – ECC)
  - Consentono anche la correzione degli errori.
  - Richiedono più bit per ogni dato (più ridondanza)
    - Per la correzione di 1 errore per parole e l'individuazione di 2 errori, occorrono 8bit /128 bit.

Codice rilevatore di errore: il codice di parità

- **Es: Bit di parità (even):**
  - aggiungo un bit ad una sequenza in modo da avere un n. pari (*even*) di "1"
    - `0000 1010 0` ← `bit di parità`
    - `0001 1010 1`
  - Un errore su uno dei bit porta ad un n. dispari di "1"
- Prestazioni del codice
  - mi accorgo dell'errore, ma non so dov'è
  - **rivelo ma non correggo errori singoli**
  - **COSTO: 1 bit aggiuntivo ogni 8 → 9/8 = +12,5%**

Codici correttori d'errore

- **Es: Codice a ripetizione**
  - Ripeto ogni singolo bit della sequenza originale per altre 2 volte → triplico ogni bit

    `0 00 1 11 1 11 0 00 1 11 0 00 0 00 1 11 ...`

  - Un errore su un bit di ciascuna terna può essere corretto:

    `000 → 010 → 000`

    `111 → 110 → 111`

- Prestazioni del codice
  - **rivelo e correggo errori singoli**
  - **COSTO: 2 bit aggiuntivi ogni 1 → 3/1 = +200%**

Definizioni

- **Distanza di Hamming**, **d** (tra 2 sequenze di N bit)
  - il numero di cifre differenti, giustapponendole

    `01001000`

    `01000010` → `d = 2`

- **Distanza minima** di un codice, d_{MIN}
  - il valor minimo di **d** tra tutte le coppie di sequenze di un codice

- Capacità di rivelazione di un codice: $t = d_{MIN} - 1$
- Capacità di correzione di un codice: $r = (d_{MIN}-1)/2$

- Esempi:
  - Codice a bit di **parità**: $d_{MIN} = 2$ → t=1, r=0
  - Codice a **ripetizione (3,1)**: $d_{MIN} = 3$ → t=2, r=1

Codice ECC di Hamming (Turing award del 1968)

Come calcolare il codice ECC Hamming

1) Enumeriamo i bit partendo da sinistra: (d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8).
2) Segnare le posizioni potenze di 2 come posizione dei bit di parità (1, 2, 4, 8, 16…)
3) I bit dei dati vanno inseriti in tutti gli altri bit (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, …)
4) La posizione del bit di parità determina di quali bit sarà responsabile (p_1 di tutti i bit che hanno indirizzo che termina con 1: i numeri dispari; p_2 di tutti i bit che hanno il secondo bit da destra = 1: 2,3 – 10,11; 6,7 – 110,111; 10,11 – 1010,1011,….).

Bit position		1	2	3	4	5	6	7	8	9	10	11	12
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity bit coverage	p1	X		X		X		X		X		X	
	p2		X	X			X	X			X	X	
	p4				X	X	X	X					X
	p8								X	X	X	X	X

Le quattro cifre p1p2p3p4 ci dicono dove c'è un errore eventualmente e quindi ci dà la possibilità di correggerlo (e.g. se valessero 1011, l'errore sarebbe in posizione 11 (d_7).

Con un parola a 12 bit possiamo correggere errori su dati a 8 bit.

Esempio

Dato: 1 0 0 1 1 0 1 0 = 154 in base 10

d1 d2 d3 d4 d5 d6 d7 d8

1 2 3 4 5 6 7 8 9 10 11 12

0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100

p1 p2 d1 p3 d2 d3 d4 p4 d5 d6 d7 d8

Determino i 4 bit di parità:

p_1. Considero i bit la cui posizione contiene un 1 nel bit meno significativo: 1, 3, 5, 7, 9, 11:

$\underline{p_1}$ p_2 $\underline{1}$ p_3 $\underline{0}$ 0 $\underline{1}$ p_4 $\underline{1}$ 0 $\underline{1}$ 0 => bit di parità $p_1 = 0$

p_2. Considero i bit la cui posizione contiene un 1 nel secondo bit meno significativo: 2, 3, 6, 7, 10, 11:

p_1 $\underline{p_2}$ $\underline{1}$ p_3 0 $\underline{0}$ $\underline{1}$ $\underline{p_4}$ 1 $\underline{0}$ $\underline{1}$ 0 => bit di parità $p_2 = 1$

p_3. Considero i bit la cui posizione contiene un 1 nel terzo bit meno significativo: 4, 5, 6, 7, (12, 13, 14, ...):

p_1 p_2 1 $\underline{p_3}$ $\underline{0}$ $\underline{0}$ $\underline{1}$ p_4 1 0 1 0 => bit di parità $p_3 = 1$

p_4. Considero i bit la cui posizione contiene un 1 nel quarto bit meno significativo: 8,9,10,11,..

p_1 p_2 1 p_3 0 0 1 $\underline{p_4}$ $\underline{1}$ $\underline{0}$ $\underline{1}$ 0 => bit di parità $p_4 = 0$

Errore di 1 bit

Codice memorizzato:

0 1 1 1 0 0 1 0 1 0 1 0

1 2 3 4 5 6 7 8 9 10 11 12

p1 p2 d1 p3 d2 d3 d4 p4 d5 d6 d7 d8

Supponiamo un errore nel bit d6 (bit 10) -> dato letto: 1001 1**1**10 invece di 1001 1010. La parola di memoria conterrà:

0 1 1 1 0 0 1 0 1 1 1 0

1 2 3 4 5 6 7 8 9 10 11 12

p1 p2 d1 p3 d2 d3 d4 p4 d5 d6 d7 d8

Analizzo i bit di parità (parità pari, 0 = numero pari di 1)

	prima	dopo											
P1:	0	-> 0	OK	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**
P2:	1	-> 0	Not OK	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**
P4:	1	-> 1	OK	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**
P8:	0	-> 0	Not OK	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**

I bit di parità che sono cambiati sono: 2 e 8 -> il bit dati che ha errore è il bit 2+8 = 10.

Esempio

Dato: 10011010 = 154 in base 10, dato letto: 10011**1**10

Confronto i 2 codici:

Codice memorizzato: 0 1 1 1 0 0 1 0 1 0 1 0

Codice memorizzato con errore: 0 0 1 1 0 0 1 1 1 1 1 0

Dato: 10011010 = 154 in base 10. Dato letto con errore: 1**1**011010 (bit 5 del codice)

Confronto i 2 codici:

Codice memorizzato: 0 1 1 1 0 0 1 0 1 0 1 0

Codice memorizzato con errore: 1 1 1 0 0 0 1 0 1 1 1 0

Analizzo i bit di parità

P1: 0-> 1 Not OK

P2: 1->1 Ok

P4: 1->0 Not OK

P8: 0->0 OK

I bit di parità che sono cambiati sono: 1 e 4 -> il bit dati che ha errore è il bit 1+4 = 5 (cioè d2).

Dimensione di codici ECC

- Conviene applicare ECC a parole più lunghe possibile → aggiungo meno ridondanza → maggiore efficienza del codice
  - A costo di complessità maggiori di codifica/decodifica

	Single-Error Correction		Single-Error Correction/ Double-Error Detection	
Data Bits	Check Bits	% Increase	Check Bits	% Increase
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

E' conveniente avere parole di memoria lunghe.

Applicazioni nelle memorie

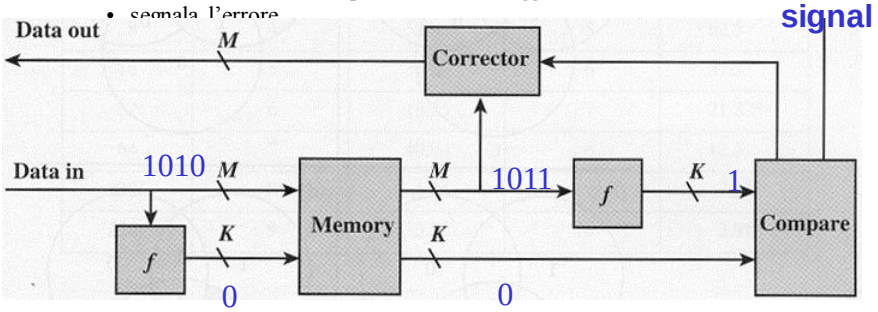
- RAM con controllo di parità
  - Aggiungo un bit di parità ad ogni byte
  - Es: RAM 1 M x 9 bit (8+1)

- RAM con codice correttore di errori (ECC)
  - si usa nelle memorie cache
  - codici ECC evoluti (alta efficienza)
    - Hamming, CCITT-32, Reed Solomon, …

Correzione degli Errori

Il controllore della memoria gestisce l'accesso e il riconoscimento degli errori.
OUT possibili:

- No errors detected
  - I dati letti possono essere inviati in uscita così come sono.
- 1 errore è stato individuato e corretto
  - I bit del dato, più il codice associato vengono inviati al correttore, il quale provvede a correggere il dato.
- 1 errore individuato, ma impossibile da correggere
  - segnala l'errore



f può essere ad esempio il codice di parità e K il codice di parità

Sommario

DRAM

Codici di errore

Esempi

Esempio di operazioni su cache

All'inizio il bit di data_valid = False. Suppongo cache di 8 linee di 1 byte.
Non importa quello che è contenuto. Cambio il contenuto di $t1.

			TAG	#L	Valid
• lb $t0, 0($t1)	$t1 = 22 (10 110)	MISS	10	6	0->1
• lb $t0, 0($t1)	$t1 = 26 (11 010)	MISS	10	4	0->1
• lb $t0, 0($t1)	$t1 = 22 (10 110)	HIT	10	6	1->1
• lb $t0, 0($t1)	$t1 = 26 (11 010)	MISS	11	4	1->1
• lb $t0, 0($t1)	$t1 = 16 (10 000)	MISS	10	0	0->1
• lb $t0, 0($t1)	$t1 = 10 (00 010)	MISS	00	2	0->1
• lb $t0, 0($t1)	$t1 = 7 (00 111)	MISS	00	7	0->1
• lb $t0, 0($t1)	$t1 = 16 (10 000)	HIT	10	0	1->1
• lb $t0, 0($t1)	$t1 = 18 (10 010)	MISS	10	2	1->1

Esempio di operazioni su cache

All'inizio il bit di data_valid = False. Suppongo cache di 8 linee di 2 byte.
Non importa quello che è contenuto. Cambio il contenuto di $t1.

			TAG	#L	#W
• lb $t0, 0($t1)	$t1 = 22 (1 011 0)	MISS	1	3	0
• lb $t0, 0($t1)	$t1 = 26 (1 101 0)	MISS	1	5	0
• lb $t0, 0($t1)	$t1 = 22 (1 011 0)	HIT	1	3	0
• lb $t0, 0($t1)	$t1 = 27 (1 101 0)	HIT	1	5	1
• lb $t0, 0($t1)	$t1 = 16 (1 000 0)	MISS	1	0	0
• lb $t0, 0($t1)	$t1 = 2 (0 001 0)	MISS	0	1	0
• lb $t0, 0($t1)	$t1 = 7 (0 011 1)	MISS	0	3	1
• sb $t0,0($t1)	$t1 = 8 (0 100 0)	n/a	0	4	0
• lb $t0, 0($t1)	$t1 = 16 (1 000 0)	HIT	1	0	0
• lb $t0, 0($t1)	$t1 = 18 (1 001 0)	MISS	1	1	0

Esercizi

Sia data una cache a corrispondenza diretta contenente 64Kbyte di dati e avente blocchi di 1 parola. Assumendo che gli indirizzi siano di 32 bit quale è il numero totale di bit richiesto per l'implementazione della cache?

Sia data una cache a 4-vie contenente 64Kbyte di dati complessivamente (16Kbyte per via) e avente blocchi di 1 parola. Assumendo che gli indirizzi siano di 32 bit quale è il numero totale di bit richiesto per l'implementazione della cache?

Quale richiede più bit? Perchè?

Esercizi

Supponendo che il MIPS abbia una cache di 512 Byte, indicare cosa succede nei campi della cache quando vengono eseguite le seguenti istruzioni:

lw $t1, 0x0000($t0) $t0 = 1kbyte = 1,024 byte
lw $t1, 0x0000($t0) $t0 = 0
lw $t1, 0x0202($t0) $t0 = 1kbyte = 1,024 byte
lw $t1, 0x0001($t0) $t0 = 0
lw $t1, 0x0201($t0) $t0 = 1kbyte = 1,024 byte

Costruire la porta di scrittura della Cache di primo livello.

Sommario

DRAM

Codici di errore

Esempi