La struttura delle memorie cache

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

Università degli Studi di Milano

Riferimento Patterson: 5.2, 5.4, 5.9, B.9

Sommario

Gestione delle hit

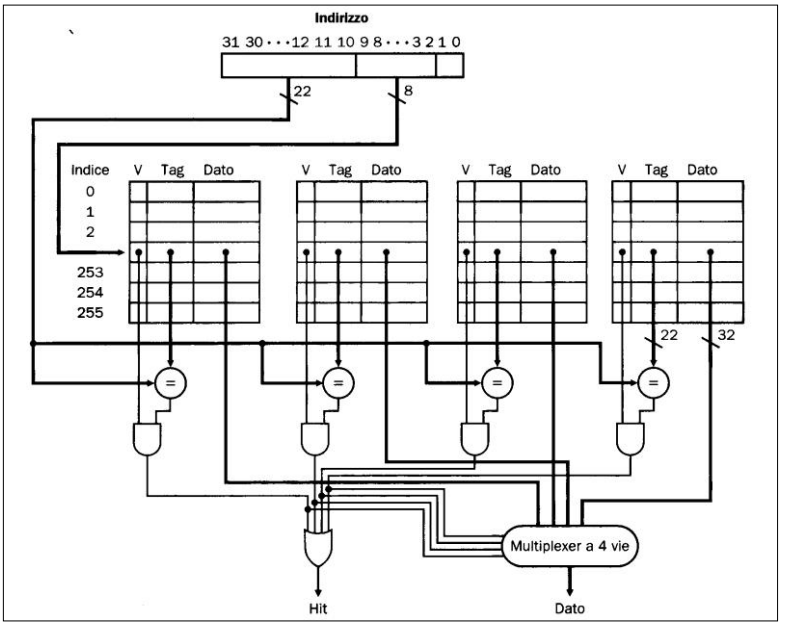
Gestione delle miss

SRAM

DRAM

Trasferimento dati

Cache n-associativa



Come si trova un blocco di MM in cache?

Corrispondenza diretta:

Indicizzazione mediante {#riga, #colonna}.
Controllo del tag + bit validità del blocco (1 comparazione).

Associativa: ricerca in tutte le n linee della cache

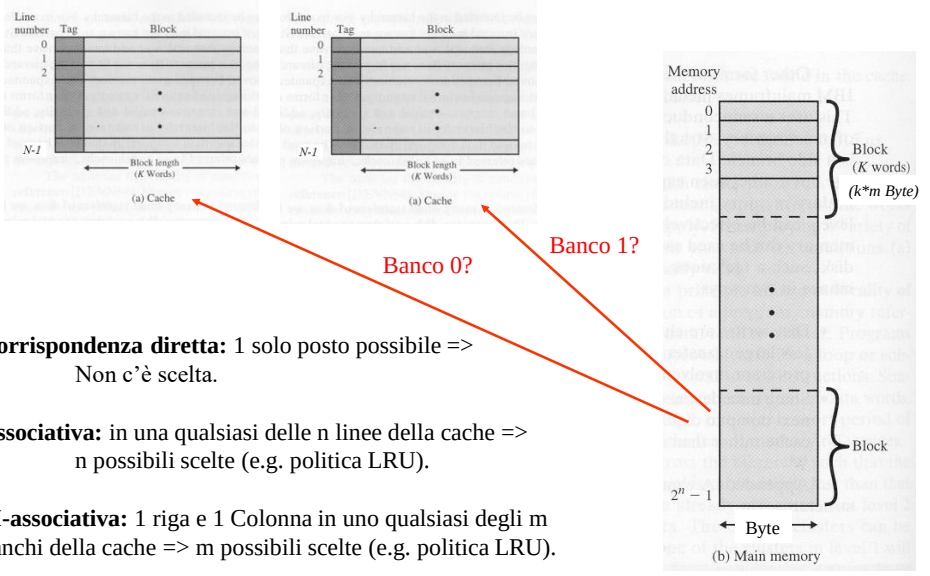
n comparazioni: controllo di tutti i tag + bit di validità + indicizzazione colonna.
Il TLB della memoria virtuale è di questo tipo.

m-associativa: ricerca negli m insiemi (banchi)

Indicizzazione mediante {#riga, #colonna}.
Controllo del tag + bit validità della linea sugli m insiemi (m comparazioni).

Politiche di sostituzione di una linea di cache

In quale banco inserisco in cache il micro-blocco letto dalla MM?



Corrispondenza diretta: 1 solo posto possibile =>
Non c'è scelta.

Associativa: in una qualsiasi delle n linee della cache =>
n possibili scelte (e.g. politica LRU).

M-associativa: 1 riga e 1 Colonna in uno qualsiasi degli m banchi della cache => m possibili scelte (e.g. politica LRU).

Tassonomia del funzionamento

HIT Successo nel tentativo di accesso ad un dato: è presente al livello superiore della gerarchia.

MISS Fallimento del tentativo di accesso al livello superiore della gerarchia => il dato o l'indirizzo devono essere cercati al livello inferiore.

HIT_RATE Percentuale dei tentativi di accesso ai livelli superiori della gerarchia che hanno avuto successo.
HIT_RATE = Numero_successi / Numero_accessi_memoria

MISS_RATE Percentuale dei tentativi di accesso ai livelli superiori della gerarchia che sono falliti
MISS_RATE = Numero_fall. / Numero_accessi_memoria

HIT_RATE + MISS_RATE = 1

HIT TIME Tempo richiesto per verificare se il micro-blocco contenente il dato è presente al livello attuale della memoria.

MISS_PENALTY Tempo richiesto per sostituire il micro-blocco di memoria mancante al livello superiore.

Criteri di progettazione

Parametri di definizione della struttura di una cache:
- Capacità
- Grado di associatività

Cache primaria: massimizzo Hit rate.

Cache secondaria: minimizzo Miss penalty (massimizzo transfer rate).

Tipi di miss di una cache

3-C miss model: compulsory, capacity and conflict.

Miss obligate (compulsory). Quando vengono caricati dei dati **per la prima volta** in una linea. Sono chiamate anche miss da partenza a freddo (**cold-start**).

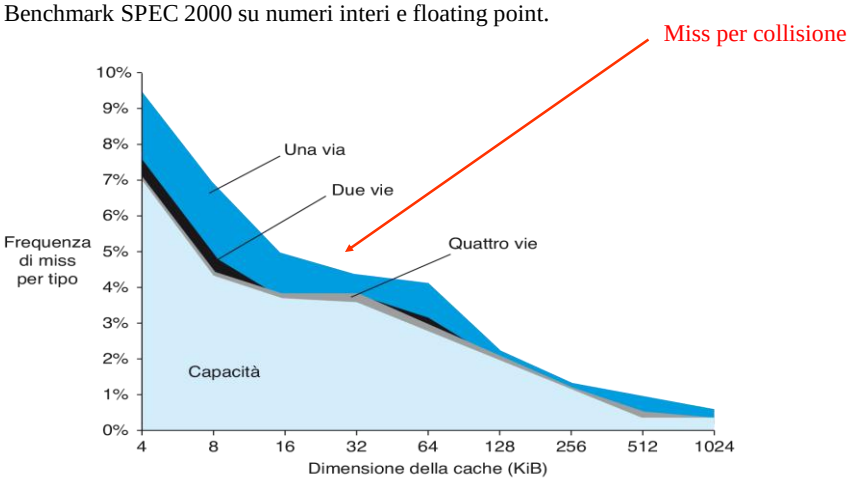
Miss per capacità (capacity). Inevitabili. La cache non può contenere tutti i micro-blocchi di dati e/o istruzioni che servono per l'esecuzione di un programma: il micro-blocco viene caricato in cache, poi scaricato e poi caricato ancora.

Miss per collisione (conflict or collision). Più micro-blocchi devono essere trasferiti nella stessa linea. In questo caso i trasferimenti devono essere accodati e sui micro-blocchi successivi si verificano miss.

Quale pesa di più?

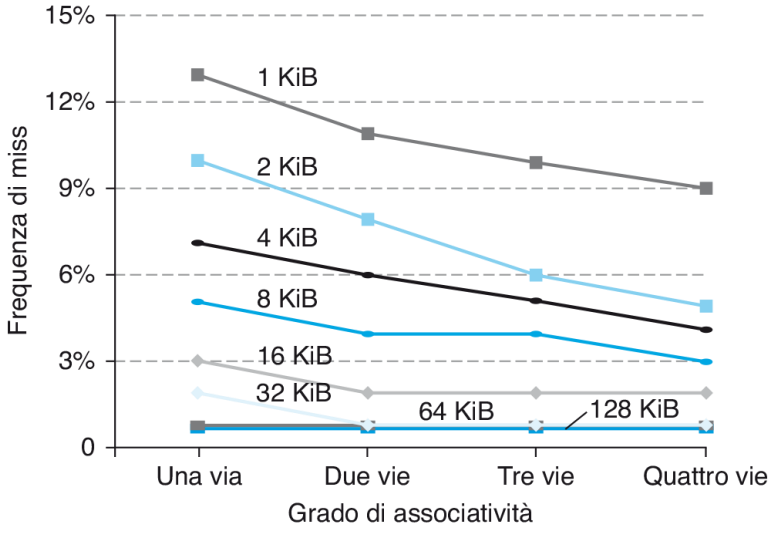
Valutazione su benchmark

Benchmark SPEC 2000 su numeri interi e floating point.



Cold-start miss: 0,006%. Incremento oltre le 4-vie non è apprezzabile.

Quanta associatività?



All'aumentare dell'associatività, aumenta la complessità e aumenta il tempo di accesso.

Mappatura diretta

4 linee di 1 word ➔ Capacità della cache = 4 * 4 Byte = 16 Byte

Codice:
lw $t0, 0($0)
lw $t1, 32($0)
lw $t2, 0($0)
lw $t3, 24($0)
lw $t4, 32($0)

0/16 = **0** R = 0
0/4 = 0 (linea 0)

32/16 = **2** R = 0
0/4 = 0 (linea 0)

0/16 = **0** R = 0
0/4 = 0 (linea 0)

24/16 = **1** R = 8
8/4 = 2 (linea 2)

32/16 = 2 R = 0
0/4 = 0 (linea 0)

Indirizzo MM	Hit o Miss	Contenuto delle linee di cache dopo ogni lw			
		0	1	2	4
0	Miss	Mem[0]			
32	Miss	Mem[32]			
0	Miss	Mem[0]			
24	Miss	Mem[0]		Mem[24]	
32	Miss	Mem[32]		Mem[24]	

4 Linee di 1 parola

5 Miss (2 miss per cold-start, 3 miss per collisione)

A 2-vie – prima possibilità

2 Banchi da 2 linee di 1 word ciascuna ➔ Capacità della linea = 4 Byte; capacità del banco = 8 Byte; Capacità della cache = 16 Byte.

Codice:
lw $t0, 0($0)
lw $t0, 32($0)
lw $t2, 0($0)
lw $t3, 24($0)
lw $t4, 32($0)

Indirizzo MM	Hit o Miss	Contenuto delle linee di cache dopo ogni lw			
		Banco 0_{lin0}	Banco 0_{lin1}	Banco 1_{lin0}	Banco 1_{lin1}
0	Miss	Mem[0]			
32	Miss	Mem[0]		Mem[32]	
0	Hit	Mem[0]		Mem[32]	
24	Miss	Mem[0]		Mem[24]	
32	Miss	Mem[32]		Mem[24]	

0/8 = 0 R = 0
0/4=0 (linea 0)->banco 0)

32/8 = 4 R = 0
0/4 = 0 (linea 0)
➔ banco 1 (la linea 0 del banco 0 è già occupata)

0/8 = 0 R = 0
0/4 = 0 (linea 0)
➔ banco 0) - Hit

24/8 = 3 R = 0
0/4 = 0 (linea 0)
➔ banco 0 o banco 1

32/8 = 4 R = 0
0/4 = 0 (linea 0)
➔ banco 0 (la linea 0 del banco 1 è già stata occupata e di recente)

4 Miss (2 miss di cold-start, 2 per collisione)

A 2-vie ottimizzata

2 Banchi da 2 linee di 1 word ciascuna ➔ Capacità della linea = 4 Byte; capacità del banco = 8 Byte; Capacità della cache = 16 Byte.

Codice:
lw $t0, 0($0)
lw $t1, 32($0)
lw $t2, 0($0)
lw $t3, 24($0)
lw $t4, 32($0)

Indirizzo MM	Hit o Miss	Contenuto delle linee di cache dopo ogni lw			
		Banco 0_{lin0}	Banco 0_{lin1}	Banco 1_{lin0}	Banco 1_{lin1}
0	Miss	Mem[0]			
32	Miss	Mem[0]		Mem[32]	
0	Hit	Mem[0]		Mem[32]	
24	Miss	Mem[24]		Mem[32]	
32	Hit	Mem[24]		Mem[32]	

0/8 = 0 R = 0
0/4 = 0 (linea 0, banco 0)

32/8 = 4 R = 0
0/4 = 0 (linea 0, banco 1)

0/8 = 0 R = 0
0/4 = 0 (linea 0, banco 0)

24/8 = 3 R = 0
0/4 = 0 (linea0 , banco 0)

32/8 = 4 R = 0
0/4 = 0 (linea0 , banco 0) - Hit

3 Miss (2 Miss da cold-start, 1 miss da collisione)

Ottimizzazione della scelta del banco
Se LRU -> ordinamento delle lw per ottimizzare gli accessi

A 4-vie (completamente associative)

4 Banchi da 1 linea di 1 word ciascuno ➔ Capacità della linea = 4 Byte; Capacità del banco = 4 Byte; Capacità della cache = 16 Byte.

Codice:
lw $t0, 0($0)
lw $t0, 32($0)
lw $t2, 0($0)
lw $t3, 24($0)
lw $t4, 32($0)

Indirizzo MM	Hit o Miss	Contenuto delle linee di cache dopo ogni lw			
		Banco 0	Banco 1	Banco 2	Banco 3
0	Miss	Mem[0]			
32	Miss	Mem[0]	Mem[32]		
0	Hit	Mem[0]	Mem[32]		
24	Miss	Mem[0]	Mem[32]	Mem[24]	
32	Hit	Mem[0]	Mem[32]	Mem[24]	

0/4 = 0 R = 0
0/4 = 0 (linea0, banco 0)

32/4 = 4 R = 0
0/4 = 0 (linea 0, banco 1)

0/4 = 0 R = 0
0/4 = 0 (linea0, banco 0) - Hit

24/4 = 3 R = 0
0/4 = 0 (linea 0, banco 2)

32/4 = 4 R = 0
0/4 = 0 (linea 0, banco 1) - Hit

3 Miss per cold-start

Effetto della modifica della struttura

Cambiamento nel progetto	Effetto sulla frequenza di miss	Eventuale effetto negativo sulle prestazioni
Aumento della dimensione della cache	Diminuiscono le miss di capacità	Può aumentare il tempo di accesso
Aumento del grado di associatività	Diminuisce la frequenza delle miss causate da conflitti	Può aumentare il tempo di accesso
Aumento della dimensione del blocco	Diminuisce la frequenza delle miss per un'ampia gamma di dimensioni dei blocchi a causa della località spaziale	Aumenta la penalità di miss. Blocchi molto grandi potrebbero aumentare la frequenza di miss

La capacità della cache L2 e L3 e l'ampiezza della loro linea, aumenta con continuità. Aumenta di poco la latenza, ma diminuiscono le miss.

La capacità della cache L1 è rimasta pressochè costante sia in dimensioni che in struttura (32 Kbyte, 1, 2, 4 vie).

Sommario

Gestione delle hit

Gestione delle miss

SRAM

DRAM

Trasferimento dati

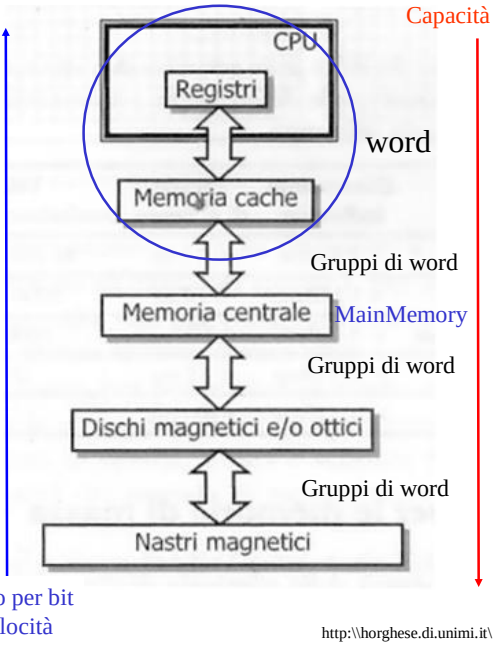
Gerarchia di memorie

Livelli multipli di memorie con diverse dimensioni e velocità.

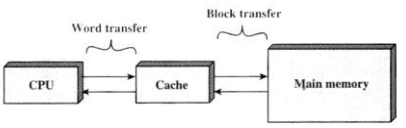
Nel livello superiore troviamo un sottoinsieme dei dati del livello inferiore.

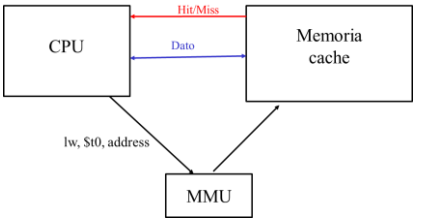
Ciascun livello vede il livello inferiore e viceversa.

Cache (memoria nascosta)



Gestione dei fallimenti di una cache





La gestione avviene tra CPU e MMU.

Hit – è quello che vorremmo ottenere, il funzionamento della CPU non viene alterato.

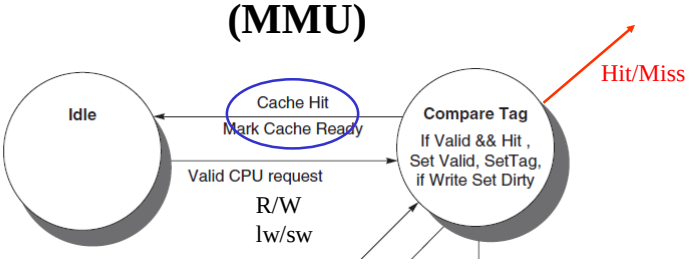
Miss – in lettura devo aspettare che il dato sia pronto in cache -> eccezione particolare della CPU che crea uno **stallo** della CPU. **Nelle CPU super-scalari si sfrutta l'esecuzione fuori ordine per nascondere questa latenza.**

Passi da eseguire in caso di Miss (miss penalty):

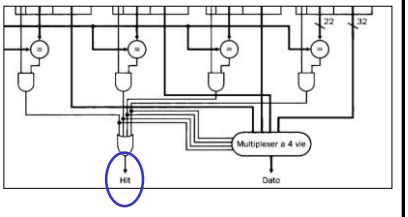
1) Bloccare tutte le istruzioni nella pipeline (blocco dei registri di pipeline per uno o più cicli di clock)
2) Richiedere che la MM legga e porti fuori il micro-blocco contenente il dato da leggere.
3) Trasferire il micro-blocco in cache, aggiornare i campi validita' e tag.
4) Riavviare l'esecuzione della pipeline.

NB Il programma non può continuare!!

Gestione di una miss lettura/scrittura (MMU)



A seguito di una richiesta di lettura/scrittura, il controllore della cache si sposta nello stato in cui viene controllato se il dato presente in cache è quello richiesto (**c'è il dato in cache?**)

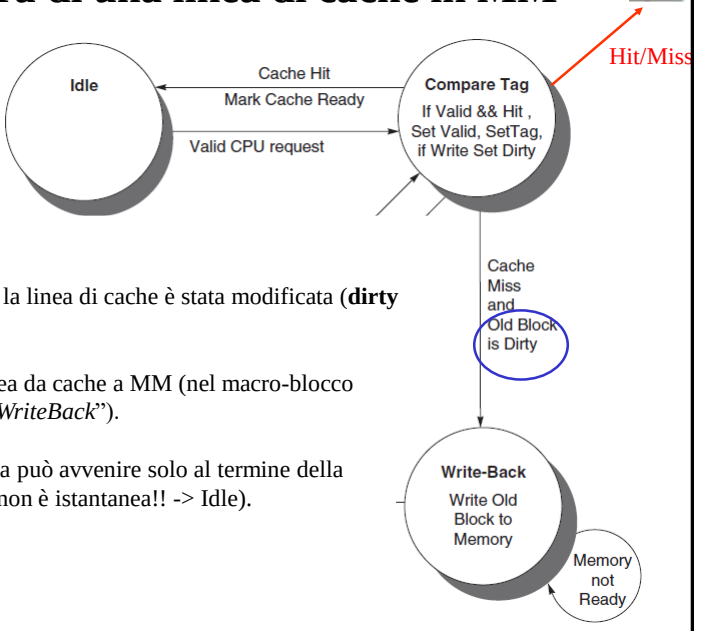


Se la condizione è verificata si ha una hit -> il dato in uscita dalla cache è quello richiesto dalla CPU.

Se la condizione non è verificata si ha una miss.

Il controllore della cache ritorna nello stato idle.

Scrittura di una linea di cache in MM

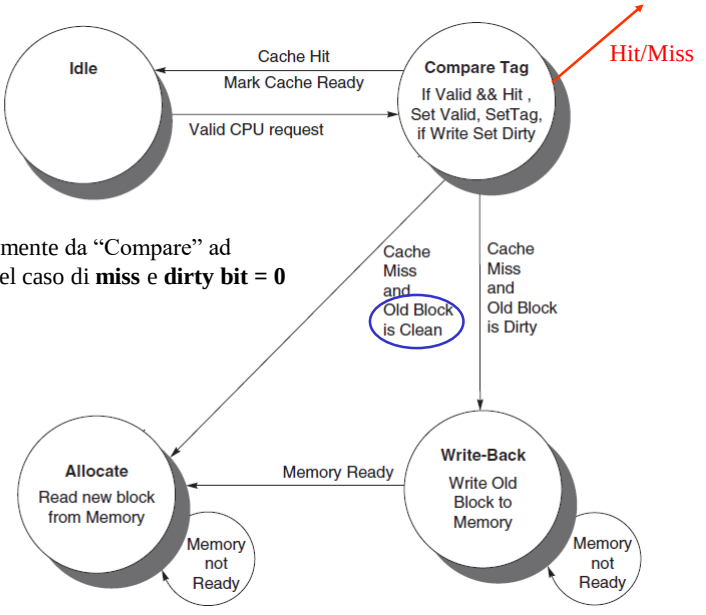


Se si verifica una **miss** e la linea di cache è stata modificata (**dirty bit** = 1),

Va prima trasferita la linea da cache a MM (nel macro-blocco identificato dal TAG – "*WriteBack*").

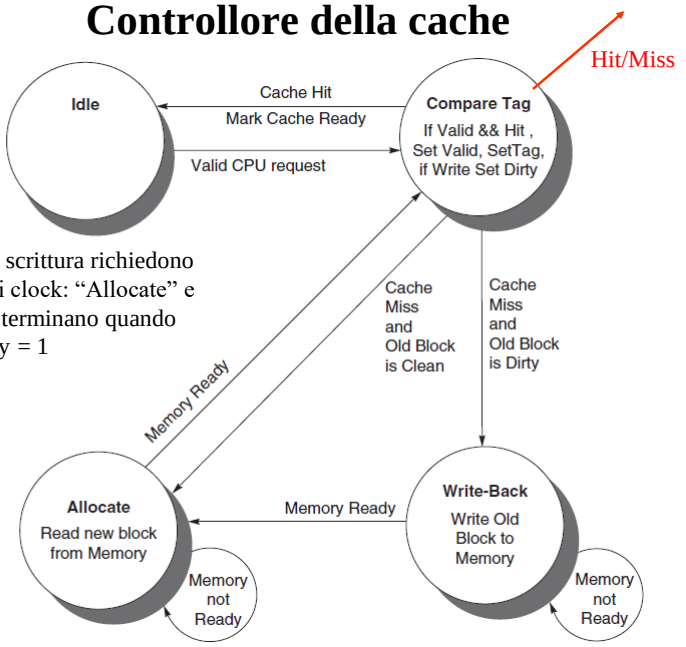
La transazione successiva può avvenire solo al termine della scrittura della MM (che non è istantanea!! -> Idle).

Lettura di una linea di cache dalla MM



Passo direttamente da "Compare" ad "Allocate" nel caso di **miss** e **dirty bit = 0**

Controllore della cache



La lettura e la scrittura richiedono diversi cicli di clock: "Allocate" e "Write-back" terminano quando MemoryReady = 1

Controllo mediante FSM nella MMU

Stato = situazione

<S, I, Y, f(S,I), g(S), S_0>

MMU

STATI del controllore (S):

- "Idle": non ci sono richieste alla cache
- "Compare": identificazione di Hit / Miss
- Scrittura della linea di cache in MM
- Lettura della linea di cache dalla MM

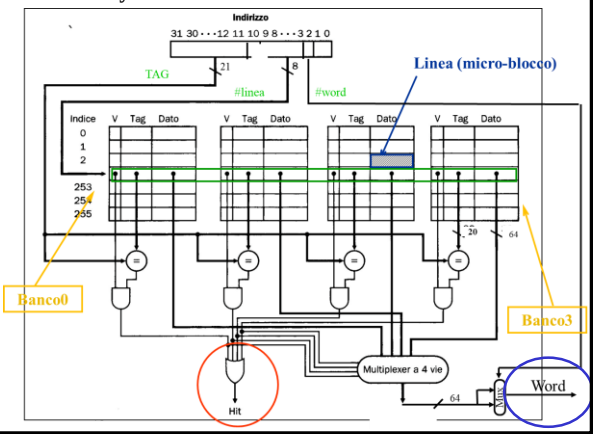
Meccanismo per decidere se una linea va scaricata: **dirty bit** = 1 se la linea è stata modificata dalla CPU (bit di validità + dirty bit).

INPUT al controllore (I):

- Read/Write MM
- DirtyBit linea cache
- MM Ready
- Hit/Miss (dalla Cache)

OUTPUT del controllore (Y):

- Hit/miss (il dato presente in uscita è quello richiesto dalla CPU).



Sommario

Gestione delle hit

Gestione delle miss

SRAM

DRAM

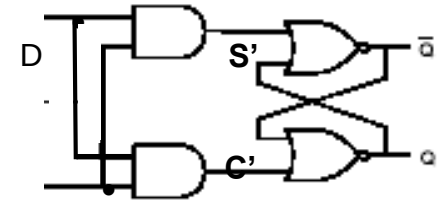
Trasferimento dati

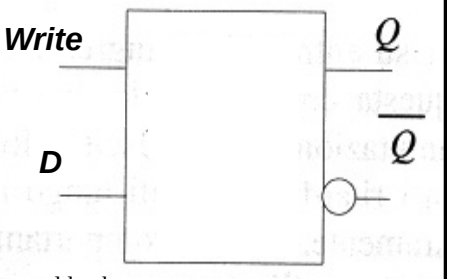
Cella SRAM

E' trasparente quando Write = 1

Se Write = 1 $Q_{t+1} = D$

Se Write = 0 $Q_{t+1} = Q_t$





Lettura - sempre disponibile in uscita

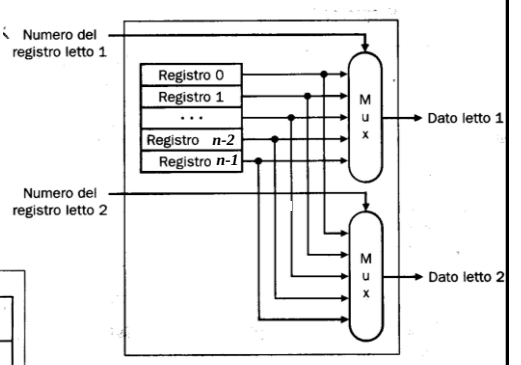
Scrittura – segnale esplicito («apertura porta di accesso al latch»

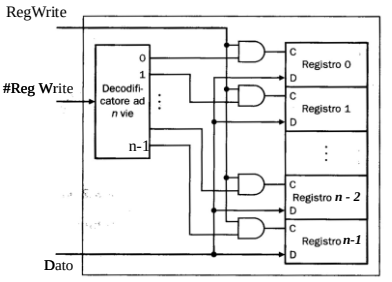
Register file

Il tempo di lettura dipende dal cammino critico dei Mux.

Il tempo di scrittura dipende dal cammino critico del Decoder.

Numero_registro = selettore.





Lettura - sempre disponibile in uscita (dopo tempo di commutazione del MUX)

Scrittura – segnale esplicito (in AND con il clock in caso di memoria sincrona).

SRAM oggi

Register File. Latch (4 **porte logiche** per cella (16 transistor), ottimizzazione della cella rispetto al latch, cf. Register File).

Tempo di accesso uguale per ogni dato.

Informazione stabile (non ci sono disturbi da parte di quello che succede intorno)

Non c'è bisogno di rinfrescare il contenuto della memoria (refresh)

Sono memorie volatili (dipendono dall'alimentazione)

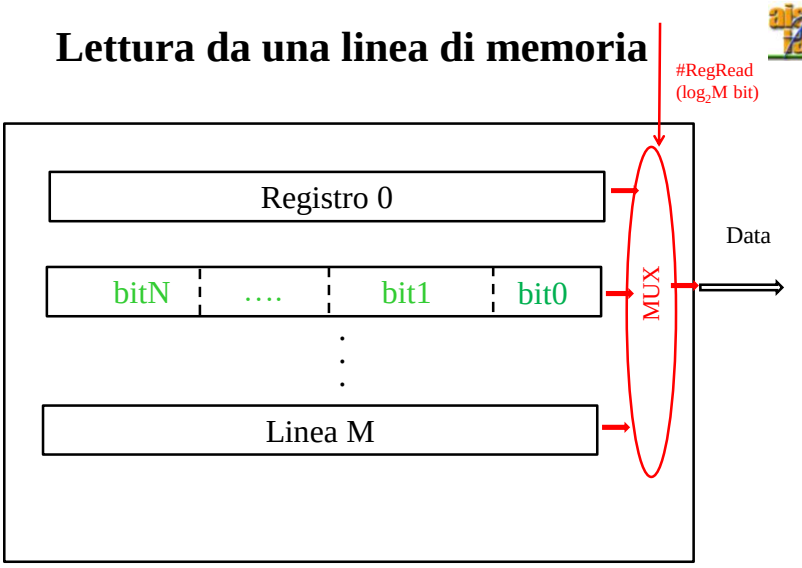
Con la tecnologia CMOS, consumano energia solo quando commutano. Poca energia viene consumata per mantenere il dato (**standby**).

SRAM - 1 sola porta di lettura / scrittura (o leggo o scrivo o non faccio nulla: {lw, sw, nulla} - 2 segnali)

Cache -> inserita nella CPU -> Produzione delle SRAM è principalmente da parte dei produttori di CPU.

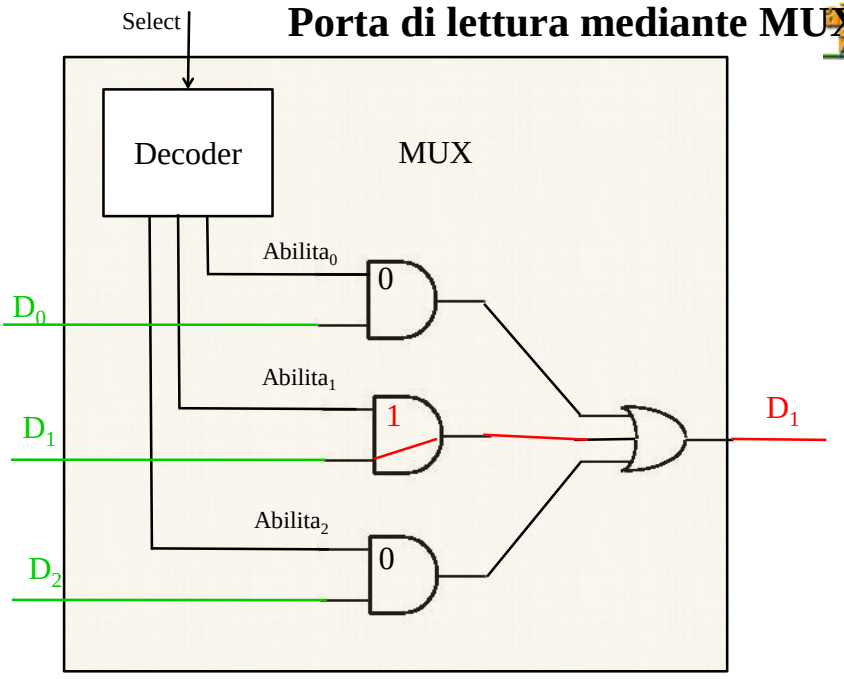
Memorie veloci per dispositivi embedded.

Lettura da una linea di memoria



Seleziono uno dei registri = porto in uscita l'uscita di tutti i bit del registro selezionato.
Il mux è pratico per 32 registri, ma non per migliaia di linee come nelle cache L2-L3.

Porta di lettura mediante MUX



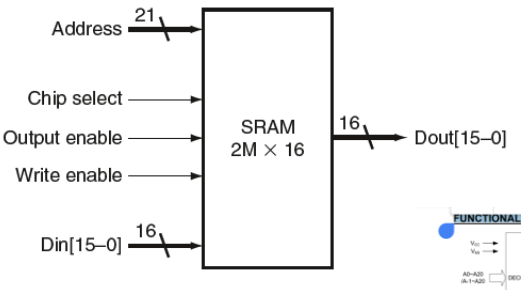
Chip di SRAM

- Sono chip con parole di ampiezza ridotta (si devono adattare a memorie di ampiezza diversa)
- 8/16 bit è un'ampiezza che consente la rilevazione degli errori HW

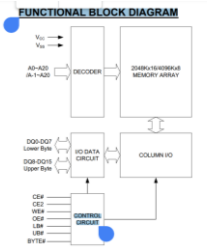
Alliance AS6C3216A-55TIN
SRAM 32M,
2Mlinee x 16 bit
2,7V – 3,6V
55ns
Asyncronous

Un chip di SRAM::esempio

48-Pin



Alliance AS6C3216A-55TIN
SRAM 32M,
2Mlinee x 16 bit
2,7V – 3,6V
55ns
Asyncronous



PIN DESCRIPTION

SYMBOL	DESCRIPTION
A0 – A20	Address Inputs(word mode)
A-1 – A20	Address Inputs(byte mode)
DQ0 – DQ15	Data Inputs/Outputs
CE#, CE2	Chip Enable Input
WE#	Write Enable Input
OE#	Output Enable Input
LB#	Lower Byte Control
UB#	Upper Byte Control
BYTE#	Byte Enable
V_{CC}	Power Supply
V_{SS}	Ground

Altezza (2 M linee) x Ampiezza (16 bit)

Altezza (2M linee) ➔ 21 bit di indirizzamento
Ampiezza (16 bit) → Bus dati su 16 bit.

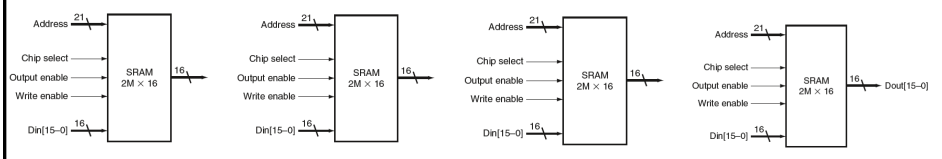
https://www.mouser.it/datasheet/2/12/AS6C3216A-55TIN-1265400.pdf

Chip select -> Abilitazione del chip (lettura e scrittura).
Write enable (abilitazione scrittura), Output enable (abilitazione lettura!).

SRAM

- Visione logica
  - struttura rettangolare o lineare
  - Indirizzo come {numero di linea, numero di colonna} o numero di elemento
- Visione fisica
  - La memoria è costituita da chip
    - Struttura rettangolare o lineare
    - Indirizzo della linea
- Come costruiamo una memoria logica con i chip di memoria?

Organizzazione fisica della memoria



Stessi 21 bit di indirizzamento -> indirizzamento della linea (2 M linee)

La linea dati e' ampia 2 Byte (ampiezza memoria sul chip) x 4 (#chip) = 8 byte = 2 word.

Capacità totale della cache: 2 M linee x 8 Byte / linea = 16 M Byte

Numero di bit richiesti per indirizzare i dati all'interno di questa cache su 4 chip: 24 bit: 21 bit per la linea, 1 bit per la parola, 2 bit per il byte interno alla parola.

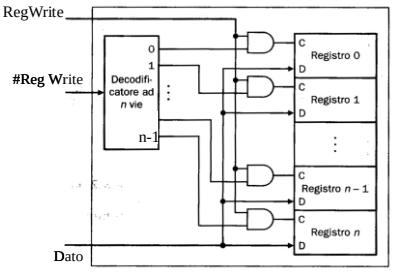
Aumentando il numero di chip in parallelo, aumento l'ampiezza della linea della memoria.

SRAM a matrice (indirizzamento)

Una **SRAM 4M x 8** avrebbe bisogno di un decoder a 22 bit -> 4M line => Serve quindi un decoder a 4M vie: 4M porte AND ciascuna con 22 bit in ingresso.

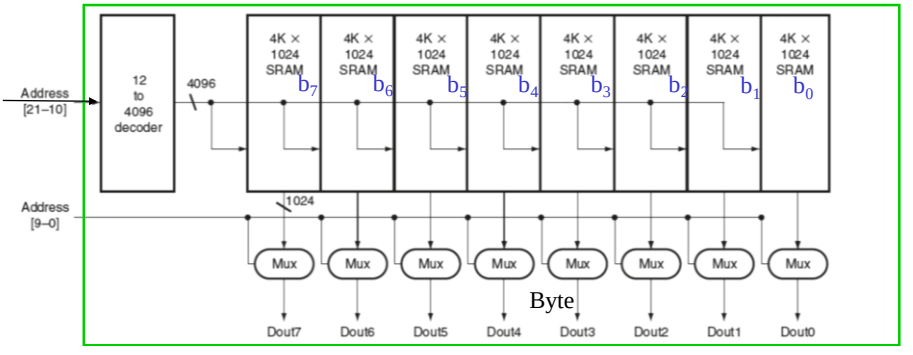
C'e' un limite elettrico al numero di linee che si possono collegare assieme e aumenta il tempo di commutazione.

E' più conveniente costruire una matrice e separare la lettura delle linee dalla lettura delle colonne (estrazione di una linea «lunga» dalla memoria – cf. Memoria cache).



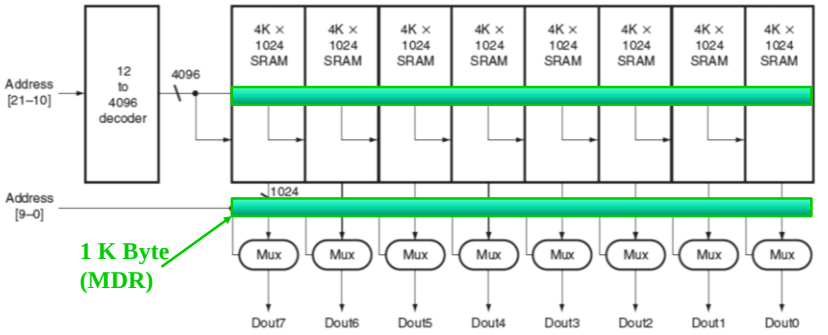
SRAM a matrice

Voglio costruire una SRAM 4M x 8 = 32 Mbit.
Ho a disposizione chip (banchi) di 4K linee x 1024 bit per linea (4K x 1K = 4 Mbit
➔ Mi servono 8 banchi in parallelo).



Il decodificatore sarà a 12 bit ($\log_2$ 4K) per selezionare una delle 4K linee (ciascuna di 1024 bit).
La stessa linea sarà selezionata in ogni banco di memoria ➔ porto fuori 1.024 x 8 = 8 Kbit = 1 KByte.
Seleziono 1 bit da ogni banco con il Mux (controllato dai 10 bit meno significativi, $\log_2$ 1.024).
=> Ottengo 1 x 8 = 1 Byte.
Nell'approccio non a matrice avrei avuto bisogno di un decodificatore a 22 bit ($\log_2$4M).

Modalità burst



Leggo prima una intera linea e poi attraverso i mux di uscita leggo 1 parola alla volta (1 Byte)
Lettura gerarchica in 2 passi.

Syncronous Static RAM (SSRAM) -> trasferisco **burst (diminuisco la miss penalty).**
Burst: indirizzo iniziale (e.g. indirizzo del primo elemento di un vettore) + lunghezza del burst (numero byte consecutivi. In questo caso al massimo saranno 1K elementi pari alla lunghezza della linea di uscita).

- Non viene richiesto di specificare un nuovo indirizzo per ogni byte -> sono adiacenti.
- E' una tecnica potente per portare fuori da una SRAM e leggere **blocchi di memoria**.

Sommario

Gestione delle hit

Gestione delle miss

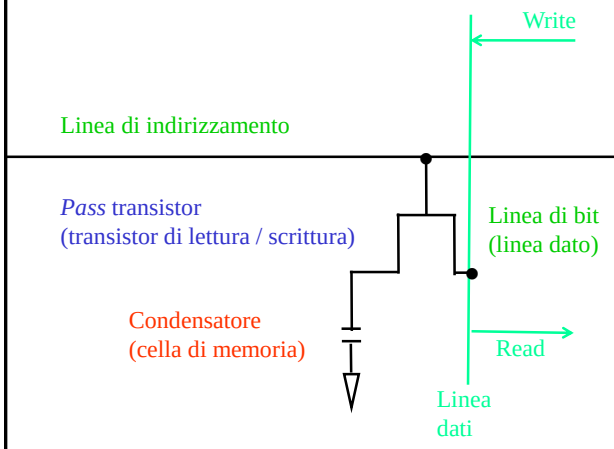
SRAM

DRAM

Trasferimento dati

Memorie DRAM

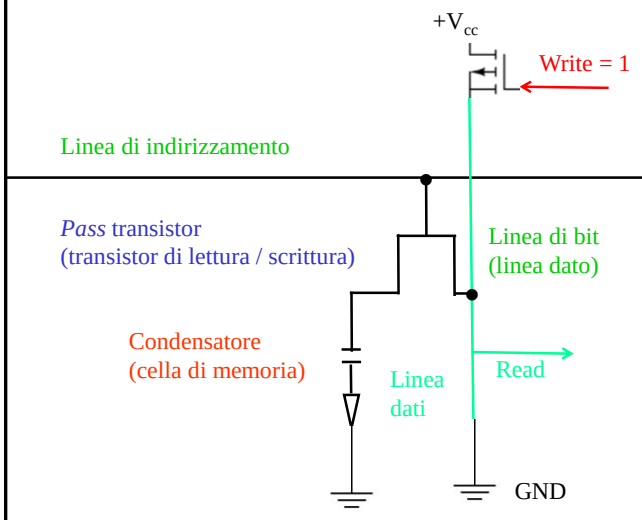
Dynamic RAM. 1 transistor per bit (contro 4-6 transistor della SRAM).



1 *Pass* transistor + 1 condensatore.

.

Memorie DRAM - scrittura

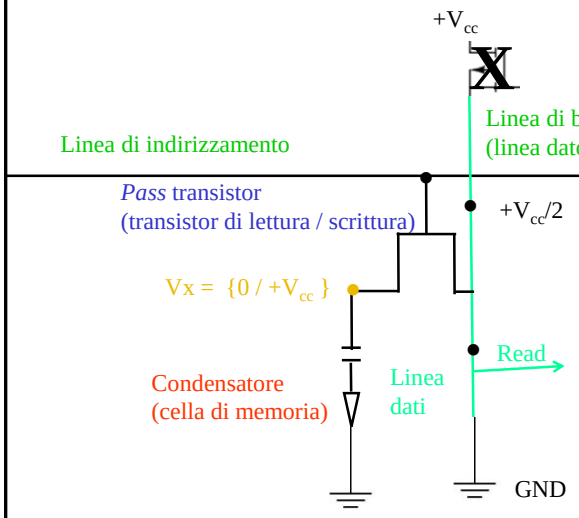


Scrittura: carica / scarica

La scrittura:
carica il condensatore da Vcc (memoria = 1) o
scarica il condensatore a GND (memoria = 0)

1 *Pass* transistor + 1 condensatore.

Memorie DRAM - lettura



Se Vx > $V_{cc}/2$ la corrente è **entrante** nella line Read

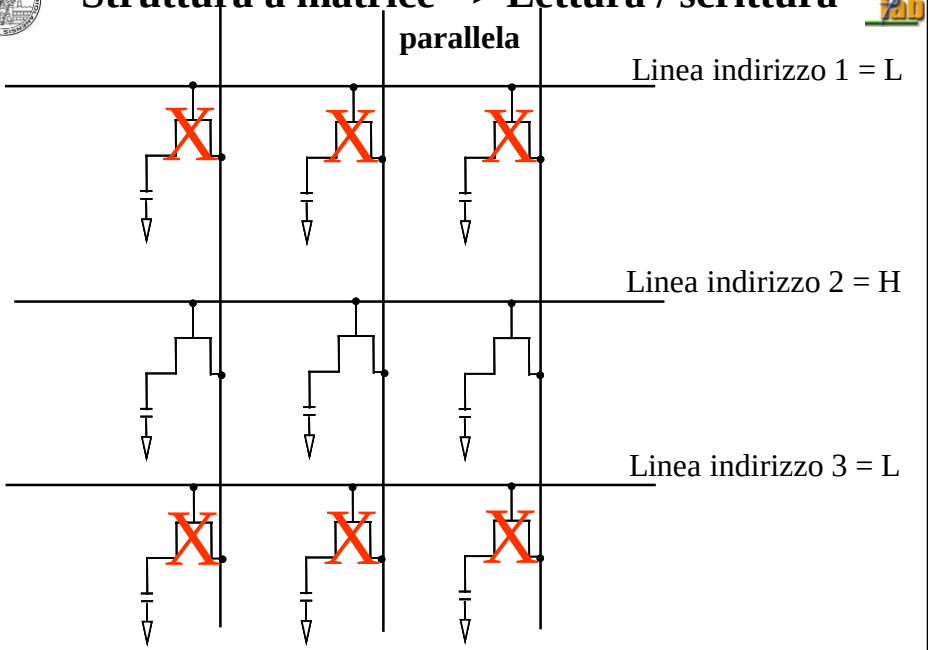
Se Vx < $V_{cc}/2$ la corrente è uscente dalla linea read.

=> Le variazioni di segno della corrente vengono trasformate in 0 / 1 da un amplificatore differenziale

La lettura è distruttiva e deve essere seguita da una lettura.

1 *Pass* transistor + 1 condensatore.

Struttura a matrice => Lettura / scrittura



I problemi delle DRAM

I condensatori sono componenti passivi -> richiedono tempo per caricarsi e scaricarsi.

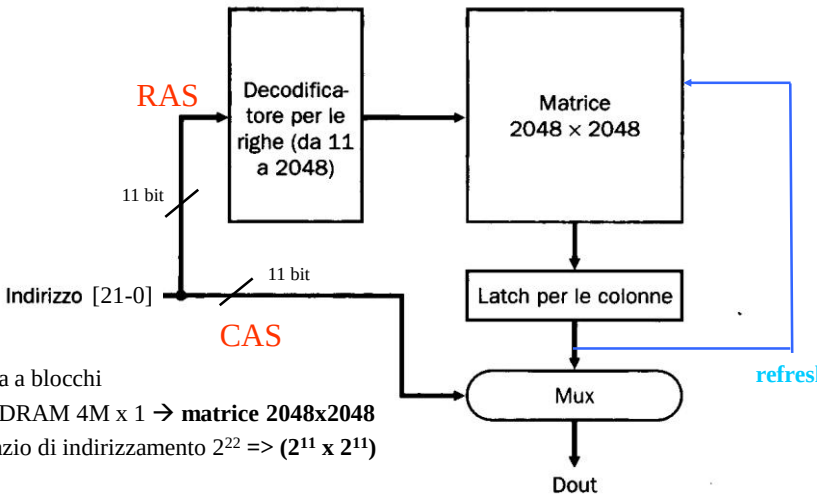
La lettura è distruttiva.

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



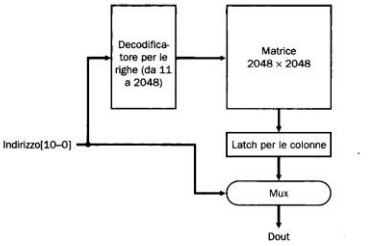
- Struttura a blocchi
  - es. DRAM 4M x 1 → **matrice 2048x2048**
  - Spazio di indirizzamento 2^{22} => **(2^{11} x 2^{11})**

•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.

Osservazioni

Latch di Colonna (MDR)

- Refresh
- Trasferimento in modalità burst (solo MUX di uscita, viene nascosta la latenza di lettura
- Modalità burst: indirizzo iniziale + lunghez del burst.
- Aiuta il blocking



SDRAM (Syncronous Dynamic RAM) sono memorie sincronizzate. I latch sono sincronizzati.
•Esiste un tempo certo per la scrittura e la lettura globale.
•Non c'è bisogno di procedure di conferma (protocolli hand-shaking).
•Il processore può fare altro.

DDR SDRAM (Double Data Rate SRAM). La sincronizzazione del funzionamento avviene su entrambi i fronti del clock. Velocità di trasferimento di picco di 3,2 Gbyte/s.

Specifiche delle DRAM

Anno introduzione	Dimensione chip	$ per GiB	Tempo accesso totale a nuova riga / colonna	Tempo medio di accesso alla riga esistente
1980	64 Kibibit	$1 500 000	250 ns	150 ns
1983	256 Kibibit	$ 500 000	185 ns	100 ns
1985	1 Mebibit	$ 200 000	135 ns	40 ns
1989	4 Mebibit	$ 50 000	110 ns	40 ns
1992	16 Mebibit	$15 000	90 ns	30 ns
1996	64 Mebibit	$10 000	60 ns	12 ns
1998	128 Mebibit	$ 4000	60 ns	10 ns
2000	256 Mebibit	$ 1000	55 ns	7 ns
2004	512 Mebibit	$ 250	50 ns	5 ns
2007	1 Gibibit	$ 50	45 ns	1,25 ns
2010	2 Gibibit	$ 30	40 ns	1 ns
2012	4 Gibibit	$ 1	35 ns	0,8 ns
2015	8 Gibibit	$ 7	30 ns	0,6 ns
2018	16 Gibibit	$ 6	25 ns	0,4 ns

Tempo totale di accesso (dall'arrivo dell'indirizzo al dato in uscita): 25 ns
Tempo di accesso in modalità burst (accesso al buffer di linea di uscita): 0.4 ns

60 volte più veloce!

Sommario

Memory miss

Gestione delle miss

SRAM

DRAM

Trasferimento dati

Gestione dei fallimenti di una cache

La gestione avviene tra CPU e MMU.

Hit – è quello che vorremmo ottenere, il funzionamento della CPU non viene alterato.

Miss – in lettura devo aspettare che il dato sia stato caricato nella linea di cache e sia pronto -> eccezione particolare della CPU che crea uno **stallo** della CPU e non un flush.
Nelle CPU super-scalari si sfrutta l'esecuzione fuori ordine per nascondere questa latenza.

Passi da eseguire in caso di Miss (miss penalty):

1) Bloccare tutte le istruzioni nella pipeline (blocco dei registri di pipeline per uno o più cicli di clock)
2) *Scaricare in MM la linea di cache interessata (micro-blocco).*
3) *Richiedere che la MM legga il micro-blocco contenente il dato da leggere e lo porti fuori nel MDR.*
4) ***Trasferire il micro-blocco in cache, aggiornare i campi validita' e tag.***
5) Riavviare l'esecuzione della pipeline.

NB Il programma non può continuare!!

I componenti della Miss penalty

Tempi di accesso:

1 ciclo di clock per inviare l'indirizzo.

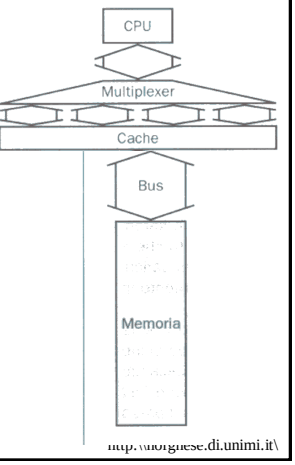
15 cicli di clock per ciascuna attivazione della Memoria Principale
(dall'invio dell'indirizzo alla parola in uscita)

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

Blocco di cache di 4 parole, blocco di memoria principale di 1 parola:

Miss_Penalty =

1 (invio indirizzo)
+
15 * 4 *(parole)* (lettura)
+
1 * 4 *(parole)* (trasferimento a cache)
=
65 cicli_clock



Obbiettivi:

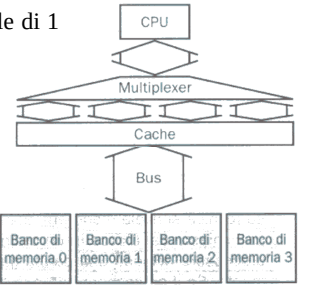
•Diminuire la penalità di fallimento (miss_penalty).

Interleaving

Blocco di cache di 4 parole, blocco di memoria principale di 1 parola:

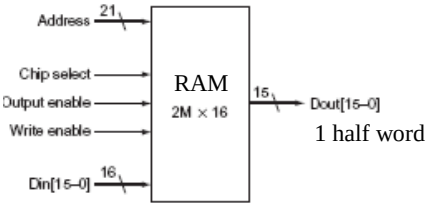
Miss_Penalty =

1 (invio indirizzo)
+
15 * 1 *(insieme di 4 parole)* (lettura)
+
1 * 4 *(parole)* (trasferimento a cache)
=
20 cicli_clock

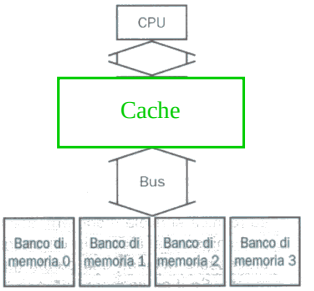


Interleaving (interallacciamento). Banchi che potrebbero essere trasferiti in parallelo alla cache. Si sposa perfettamente con la **struttura gerarchica** della memoria e con la **modalità di trasferimento a burst**.

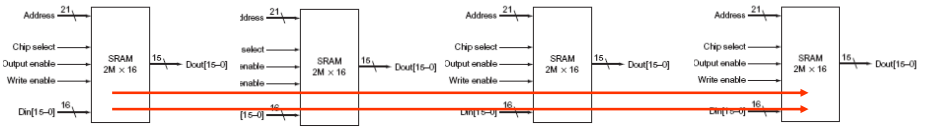
Osservazioni sull'interleaving



1 half word



Linea di cache di 4 parole = 16 Byte = 8 half word



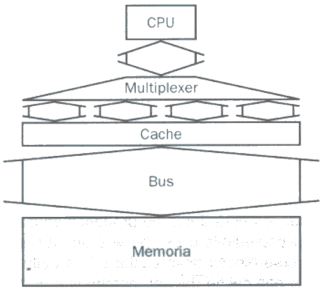
Il vettore di MM viene spalmato sulle linee.
Leggo dal MDR in uscita (velocità di 40x)

Bus ampio

Blocco di cache di 4 parole, blocco di memoria principale di 1 parola:

Miss_Penalty =

1 (invio indirizzo)
+
15 * 1 *(insieme di 4 parole)* (lettura)
+
1 * 1 *(insieme di 4 parole)* (trasferimento a cache)
=
17 cicli_clock

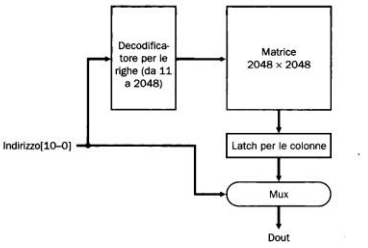


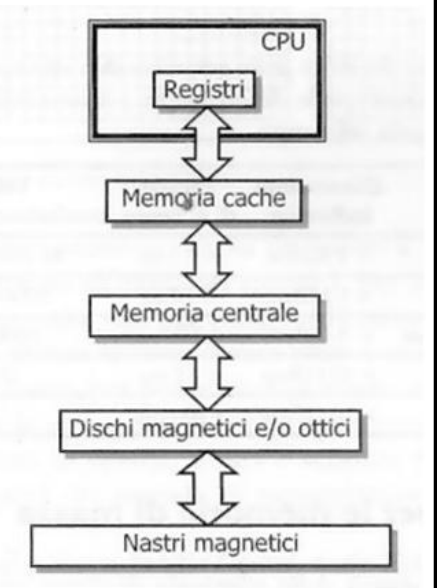
Complessità del bus non giustificata. Si va verso bus sempre piu' stretti (insiemi di bus seriali a 1 bit).

Organizzazione della memoria

Elementi chiave per nascondere la latenza:

- Organizzazione a matrice (cf. cache)
- Organizzazione gerarchica.





Sommario

Memory miss

SRAM

DRAM

Trasferimento dati