


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

Memory miss

SRAM

DRAM

Trasferimento dati

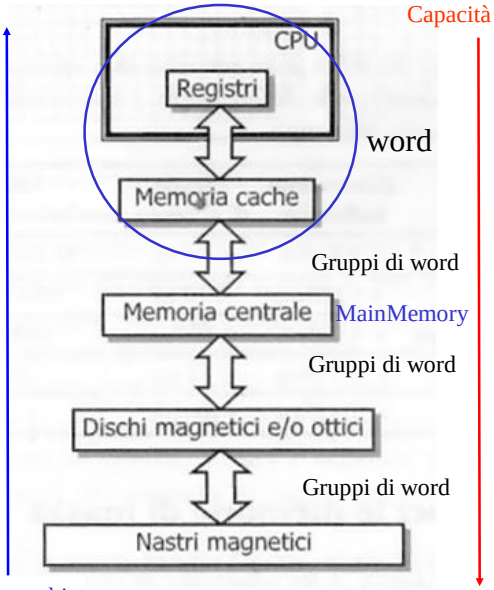
Gerarchia di memorie

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 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!!

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

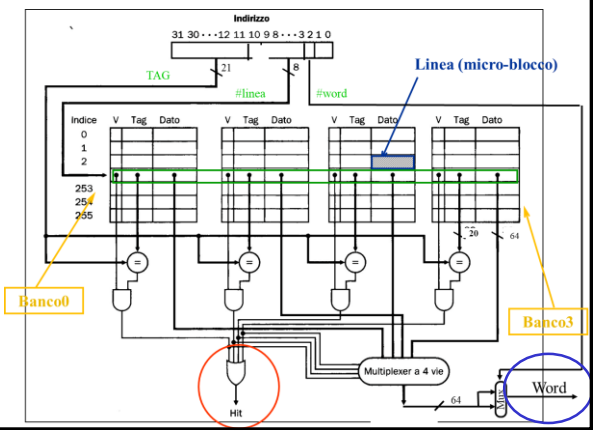
Meccanismo per decidere se una linea va scaricata: **dirty bit** = 1 se la linea è stata modificata dalla CPU (bit di validità + dirty bit).

INPUT al controllore (I):

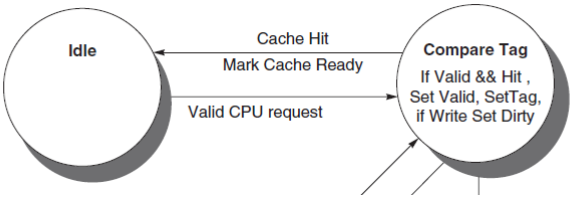
- Read/Write MM
- DirtyBit linea cache
- MM Ready

OUTPUT del controllore (Y):

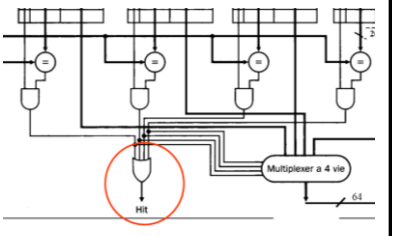
- Hit/miss (il dato presente in uscita è quello richiesto dalla CPU).



Gestione di una Hit



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?**)

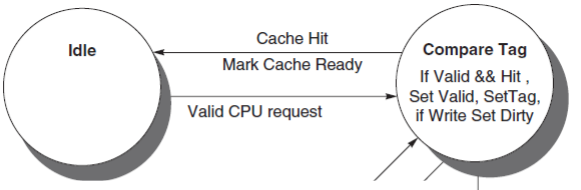


Se la condizione è verificata si ha una hit -> il dato in uscita dalla cache è quello richiesto dalla CPU.

Se la condizione non è verificata si ha una miss.

Il controllore della cache ritorna nello stato idle.

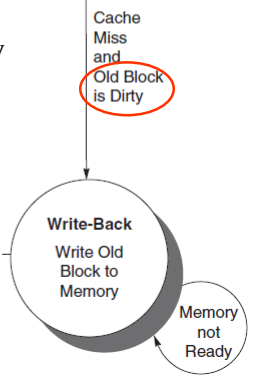
Scrittura di una linea di cache in MM



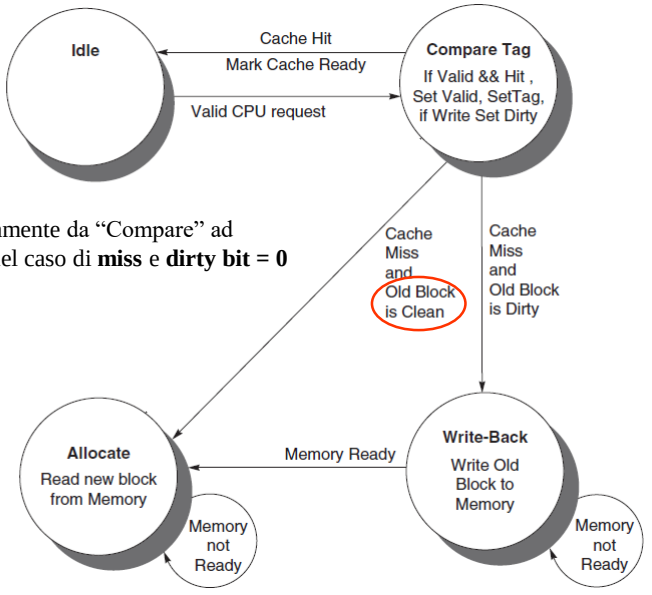
Se si verifica una **miss** e la linea di cache è stata modificata (**dirty bit** = 1),

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).

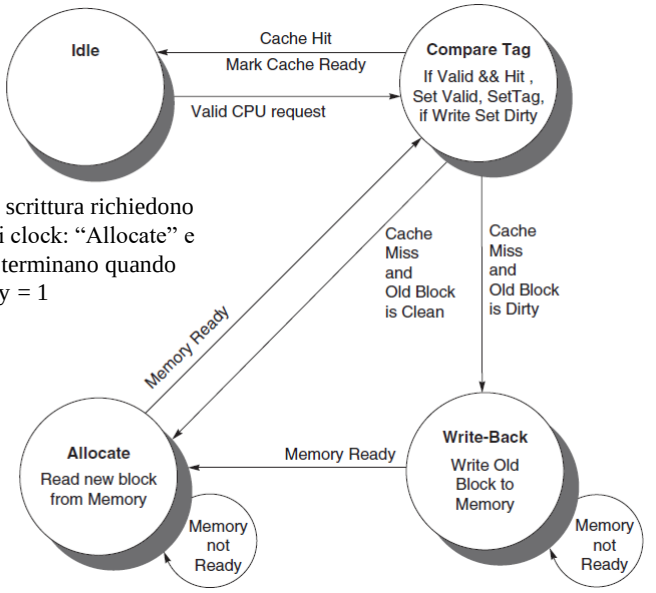


Lettura di una linea di cache dalla MM



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

Sommario

Memory miss

SRAM

DRAM

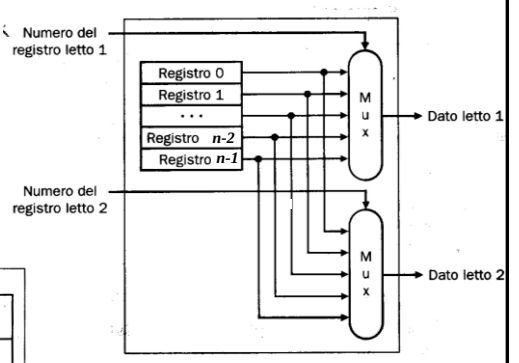
Trasferimento dati

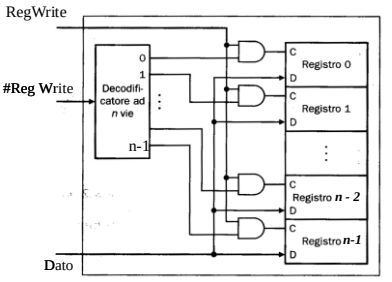
Register file

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)

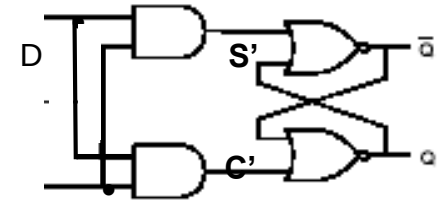
Scrittura – segnale esplicito (in AND con il clock in caso di memoria sincrona).

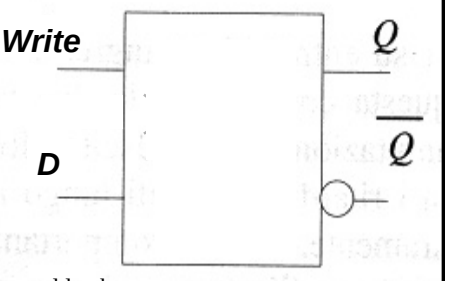
Cella SRAM

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»

SRAM oggi

Static RAM. Bistabili (4-6 **transistor** per cella, 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)

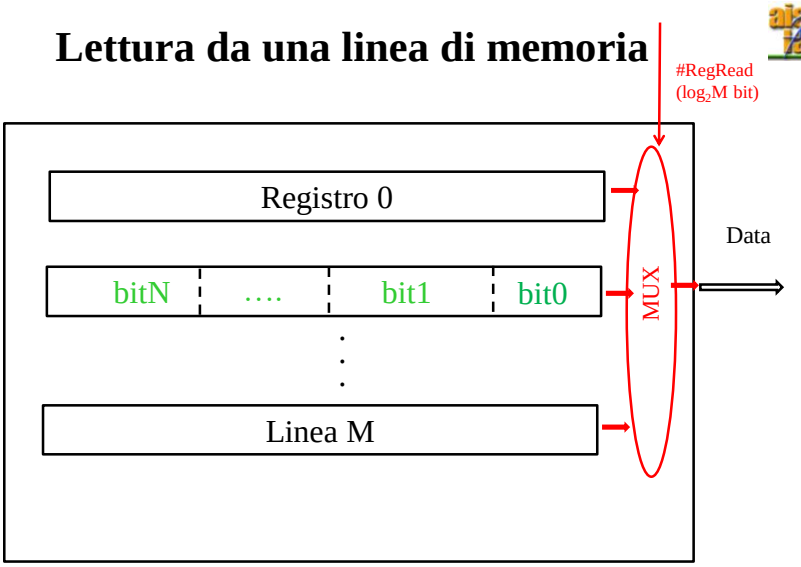
Con la tecnologia CMOS, consumano energia solo quando commutano. Poca energia viene consumata per mantenere il dato (**standby**).

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

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

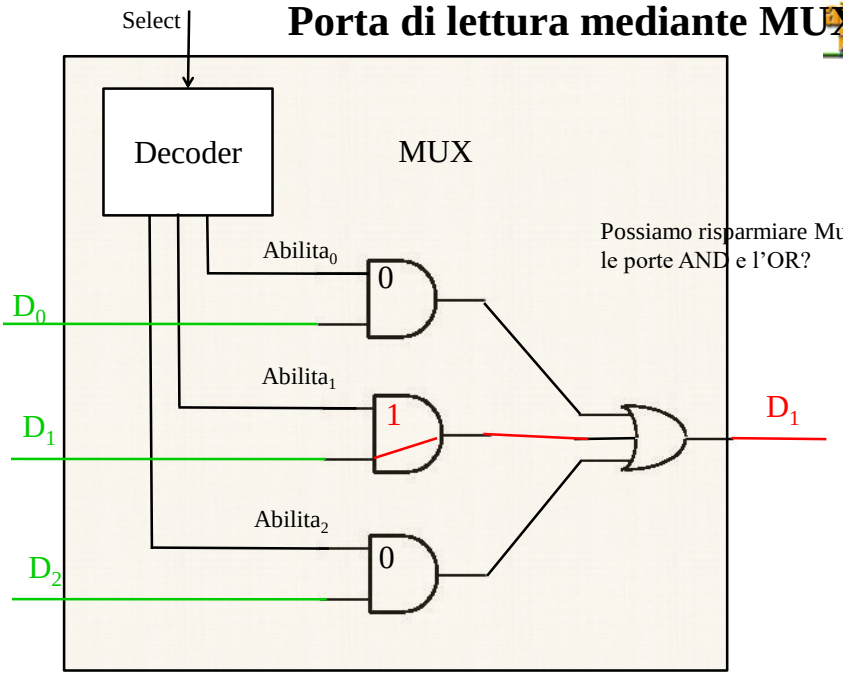
Memorie veloci per dispositivi embedded.

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 L1-L3.

Porta di lettura mediante MUX



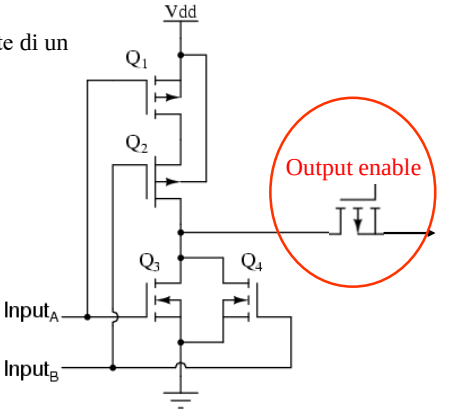
Possiamo risparmiare Mux, le porte AND e l'OR?

Porte three-state

- **Stato 0** – spento – connesso a massa (0V)
- **Stato 1** – acceso – connesso all'alimentazione (+Vcc)

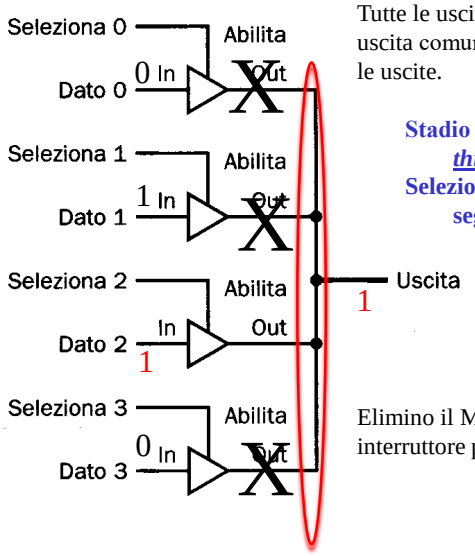
- **Stato disabilitato** – disconnesso – non connesso a nulla, tensione floating (alta impedenza). Interruttore di accensione del circuito.

I circuiti a valle vedono l'equivalente di un "filo tagliato".



- Low
- High
- Staccato

Memoria three-state (soluzione HW)

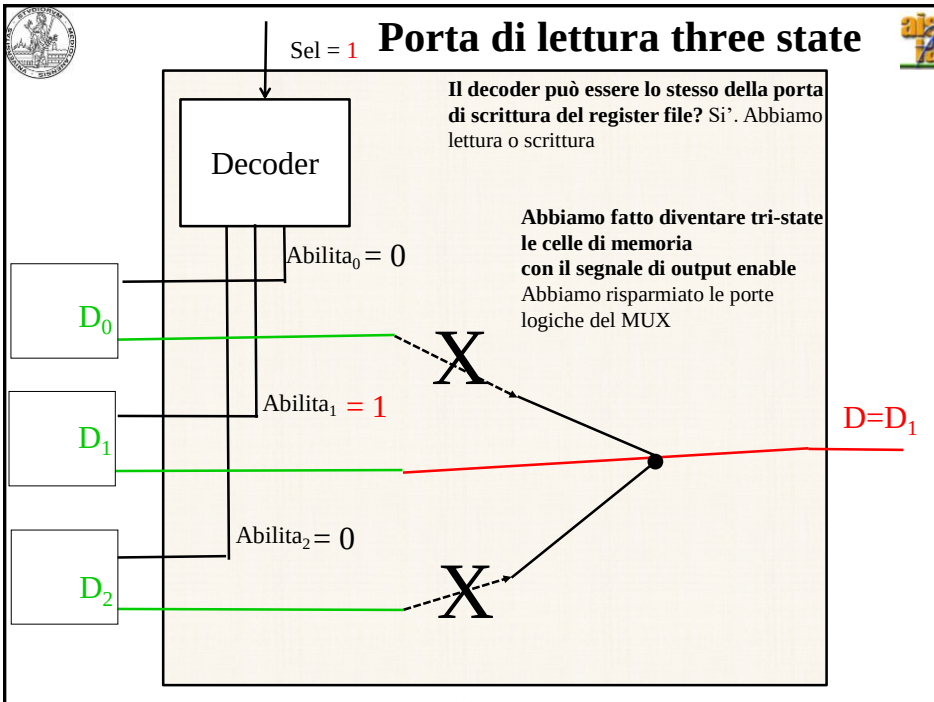


Tutte le uscite delle celle sono collegate ad un'unica uscita comune => E' necessario evitare conflitti fra le uscite.

Stadio di uscita "isolata" con porte di uscita *three-state*
Seleziono una sola cella alla volta mediante segnale di abilitazione.

Elimino il MUX -> sostituito da un transistor-interruttore per bit, che lavorano in parallelo.

Porta di lettura three state

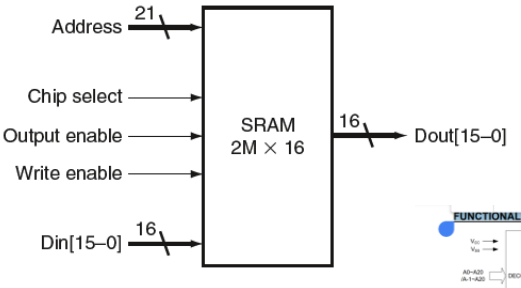


Il decoder può essere lo stesso della porta di scrittura del register file? Si'. Abbiamo lettura o scrittura

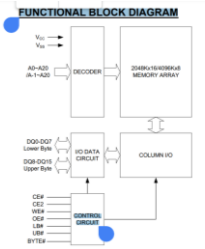
Abbiamo fatto diventare tri-state le celle di memoria con il segnale di output enable
Abbiamo risparmiato le porte logiche del MUX

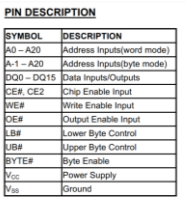
Un chip di SRAM

48-Pin



Alliance AS6C3216A-55TIN
SRAM 32M,
2,7V – 3,6V
55ns
2048K x 16
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 Mlinee) x Ampiezza (16 bit)

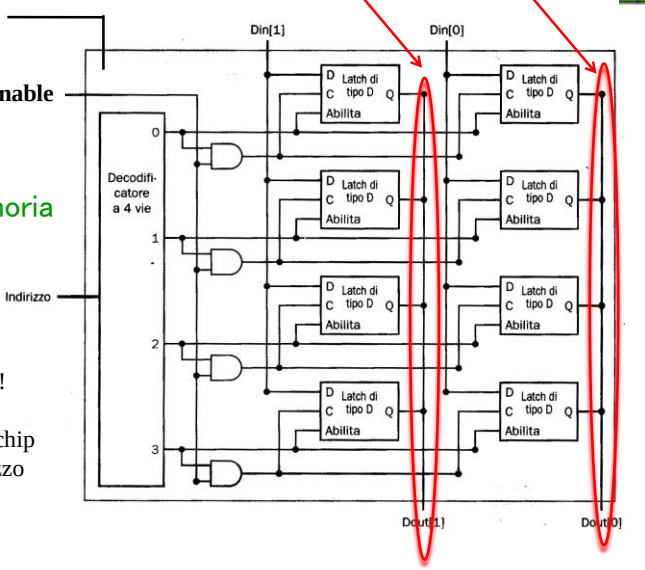
Altezza (2 linea) ➔ 21 bit di indirizzamento
Ampiezza (16 bit) ➔ Bus dati su 16 bit.

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!).

Esempio di SRAM in scrittura

Risparmio il Mux di uscita



Esempio: SRAM
4 elementi di memoria
x 2 bit ciascuno

Indirizzo -> Enable
Abilito sempre un'uscita!

Chip enable? Abilita il chip se l'indirizzo è un indirizzo nel chip.

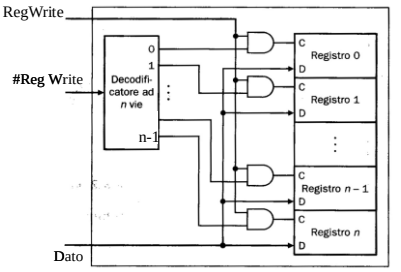
Il decodificatore è una schiera di AND che lavorano in parallelo: meno complesso e più veloce di un Mux. **Not enough!**

SRAM a matrice

Una **SRAM 4M x 8** avrebbe bisogno di un decoder a 22 bit -> 4M linee, con 4M porte AND ciascuna con 22 bit in ingresso.

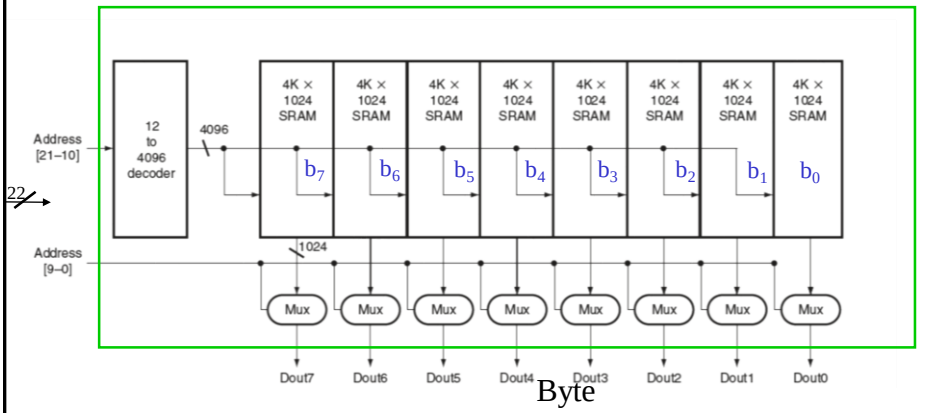
C'e' un limite elettrico al numero di linee che si possono collegare assieme e aumenta il tempo di commutazione.

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

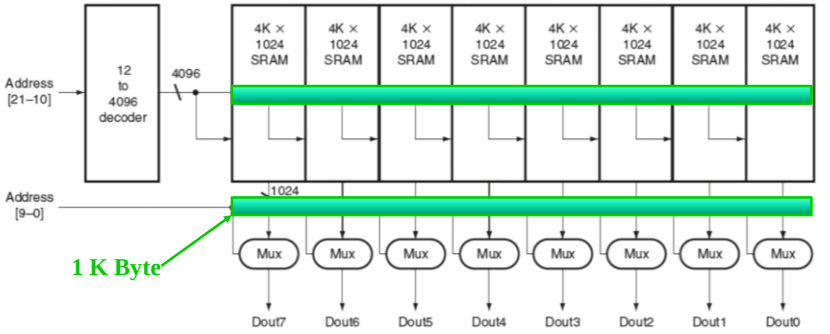
SRAM 4M x 8. Ad ogni bit assegno un chip (banco) di 4K linee x 1024 bit per linea (4M bit => internamente è una memoria 4K linee x 1.024 bit.



Il decodificatore sarà a 12 bit ($\log_2$ 4K) per selezionare una delle 4096 linee (ciascuna di 1024 bit).
Ciascuna linea fornisce 1024 bit.
La stessa linea sarà selezionata in ogni banco di memoria → porto fuori 1.024 x 8 = 8 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.

A. Nell'approccio non a matrice avrei avuto bisogno di un decodificatore a 22 bit ($\log_2$4M).

Modalità burst

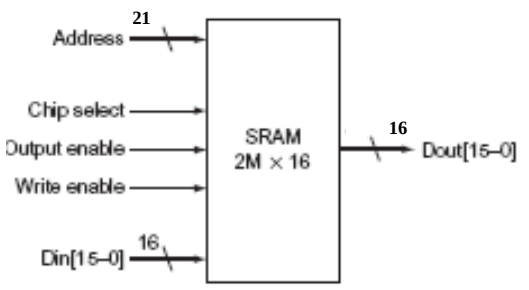


Leggo prima una intera linea e poi attraverso i mux di uscita leggo la parola alla volta (1 Byte)
Lettura gerarchica in 2 passi.

Syncronous Static RAM (SSRAM) -> trasferisco **burst**.
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**.

Chip di SRAM (2M x 16)



Tempo di accesso:
da Address a Dout.

Altezza (2 Mbit) x Ampiezza (16 bit) – Oggi 1-4 bit

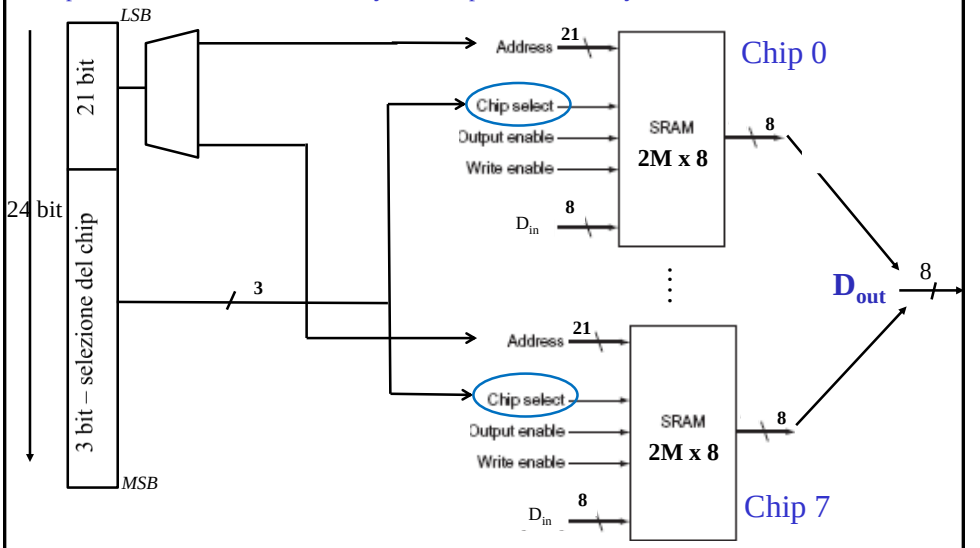
Altezza (2 Mbit) ➔ 21 bit di indirizzamento
Ampiezza (16 bit) → Bus dati su 16 bit.

C**hip select -> Abilitazione del chip (per risparmiare energia, indirizzo nel range del chip).**

Write enable (abilitazione scrittura), Output enable (abilitazione lettura!).

Memoria SRAM come insieme di chip

Capacità totale = 8 Banchi * 2 MByte => Capacità di 16 MByte

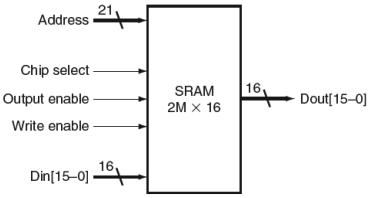


Principio di località: i dati all'interno di un chip verranno utilizzati negli istanti di tempo successivi. Abilitazione e disabilitazione sono eventi relativamente poco frequenti.

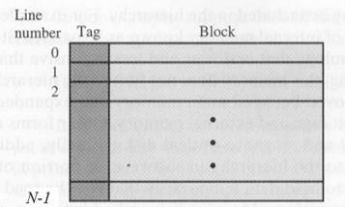
Esempio

Cache a mappatura diretta
Capacità della cache: 128 MByte
Linee di 4 word e word di 4 Byte (dim_linea = 16 Byte).
#Linee di cache: 128 Mbyte / 16 Byte = 8 Mlinee
Costruzione mediante SRAM da 2 Mlinee x 16 bit
Capacità del chip: 4 MByte

Chip di SRAM



Ampiezza 2 Byte

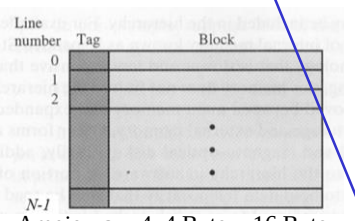


Ampiezza = 4x4 Byte = 16 Byte

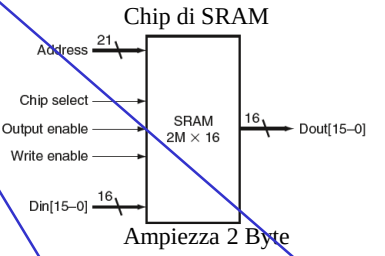
"Tessellazione della cache con i chip"

Dimensionamento

Capacità della cache: 128 MByte
Linee di 4 word e word di 4 Byte (dim_linea = 16 Byte).
#Linee di cache: 128 Mbyte / 16 Byte = 8 Mlinee
Costruzione mediante SRAM da 2 Mlinee x 16 bit
Capacità del chip: 4 MByte
#Chip = 128 Mbyte / 4 MByte = 32 chip



Ampiezza = 4x4 Byte = 16 Byte



Ampiezza 2 Byte

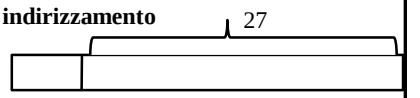
Servono 8 chip di SRAM affiancati per realizzare una linea di cache (8 chip x 2 Byte = 16 Byte).

Servono 4 chip "inpilati" per ottenere 8 Mlinee (2Mlinee x 4 = 8 Mlinee)

I chip saranno disposti in una matrice 4 x 8.

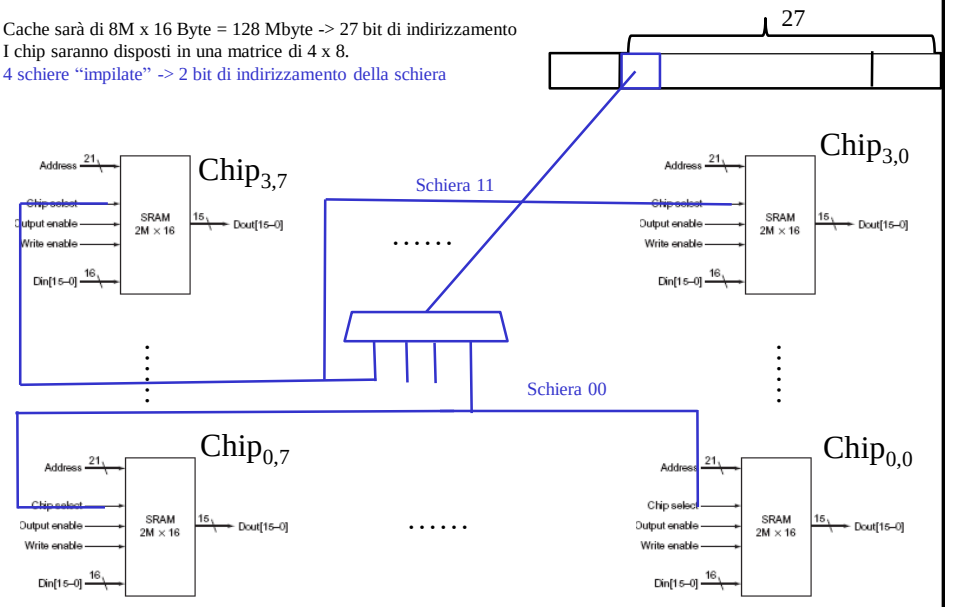
Indirizzamento

Cache sarà di 8M x 16 Byte = 128 Mbyte -> **27 bit di indirizzamento**
I chip saranno disposti in una matrice di 4 x 8
(4 schiere di 8 chip)



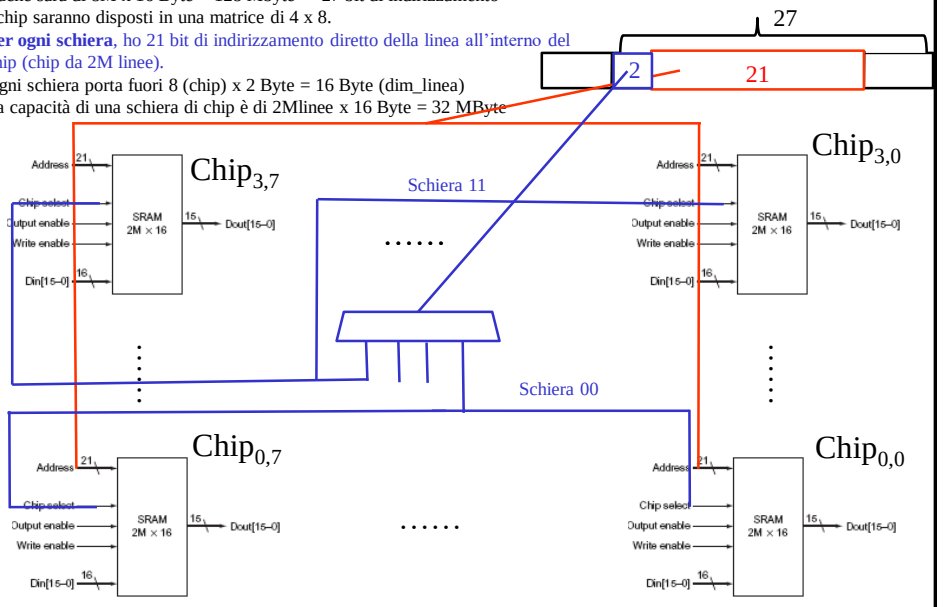
Selezione della schiera di chip

Cache sarà di 8M x 16 Byte = 128 Mbyte -> 27 bit di indirizzamento
I chip saranno disposti in una matrice di 4 x 8.
4 schiere "impilate" -> 2 bit di indirizzamento della schiera



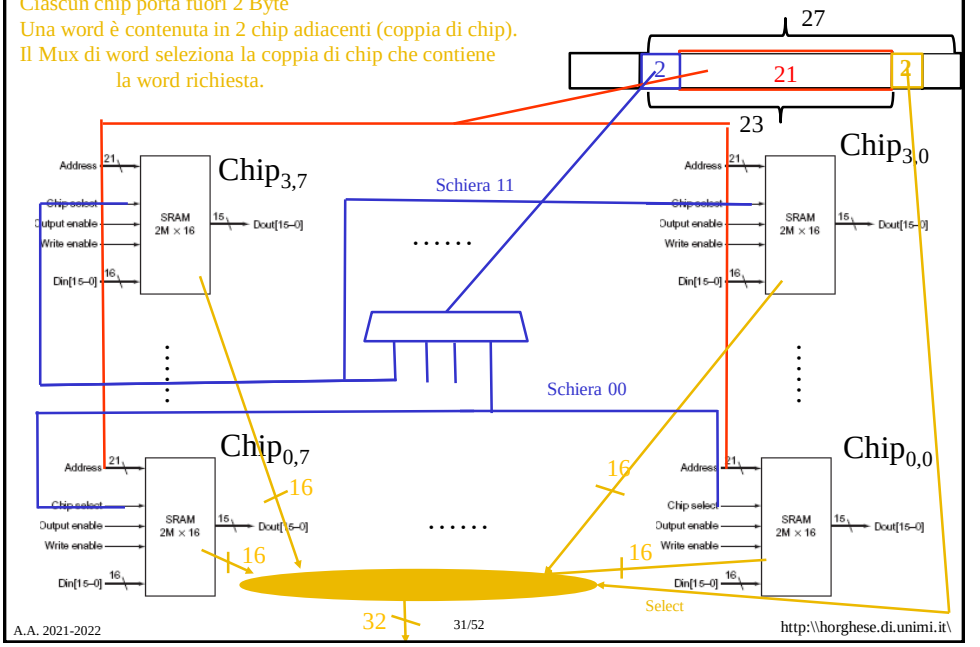
Selezione della linea

Cache sarà di 8M x 16 Byte = 128 Mbyte -> 27 bit di indirizzamento
I chip saranno disposti in una matrice di 4 x 8.
Per ogni schiera, ho 21 bit di indirizzamento diretto della linea all'interno del chip (chip da 2M linee).
Ogni schiera porta fuori 8 (chip) x 2 Byte = 16 Byte (dim_linea)
La capacità di una schiera di chip è di 2Mlinee x 16 Byte = 32 MByte



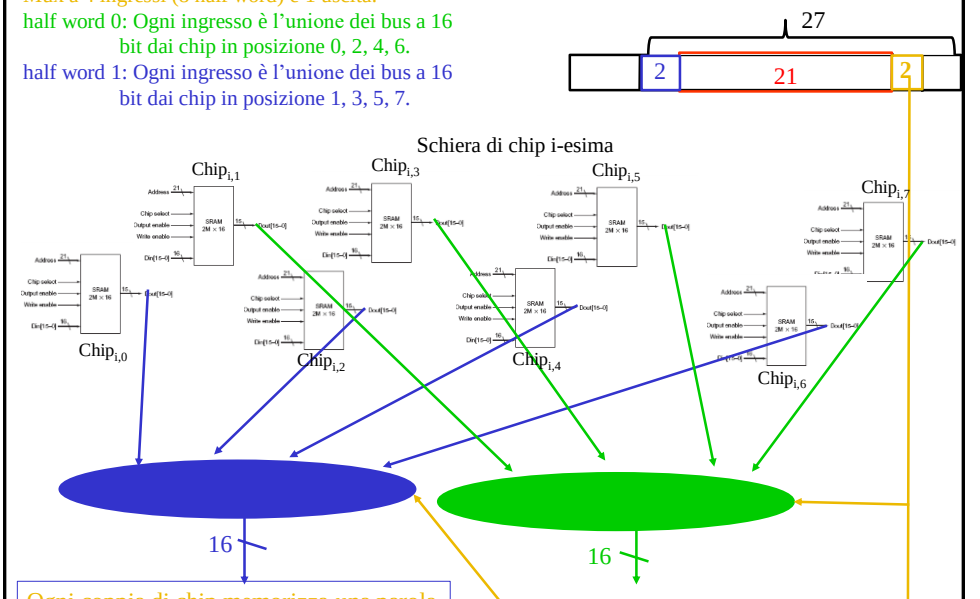
Selezione della colonna

Ciascun chip porta fuori 2 Byte
Una word è contenuta in 2 chip adiacenti (coppia di chip).
Il Mux di word seleziona la coppia di chip che contiene la word richiesta.



Selezione della Colonna: dettaglio

Mux a 4 ingressi (8 half word) e 1 uscita.
half word 0: Ogni ingresso è l'unione dei bus a 16 bit dai chip in posizione 0, 2, 4, 6.
half word 1: Ogni ingresso è l'unione dei bus a 16 bit dai chip in posizione 1, 3, 5, 7.

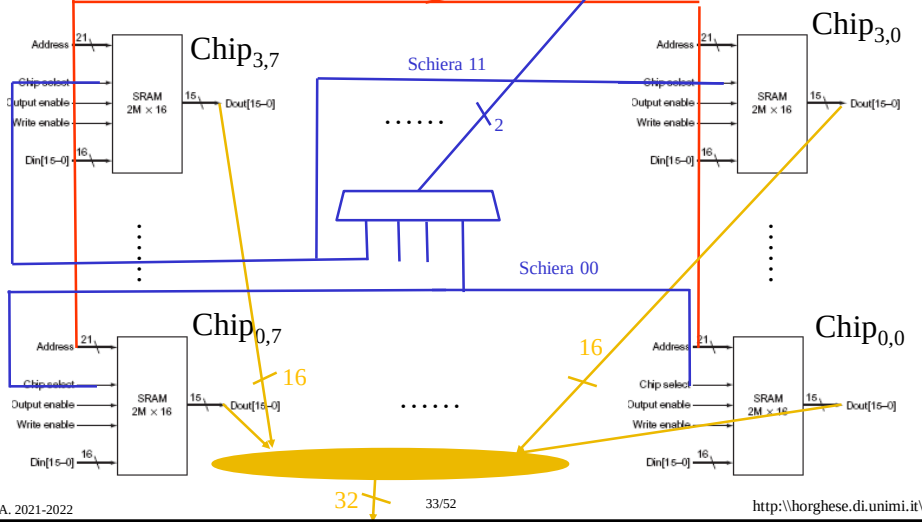


Ogni coppia di chip memorizza una parola

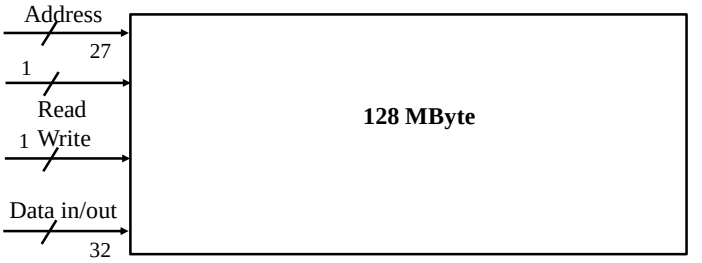
Posizione della parola

lw $t1, 172($0) -> parsing dell'indirizzo
Macro-blocco 0 di MM (indirizzo < 128 Mbyte)
Schiera 0 dei chip (indirizzo < 32 Mbyte – capacità di una schiera)
Linea 10 all'interno del chip (172 / 16 Byte = 10, R = 12)
Parola 3 => Chip 6 e 7 (12 Byte / 4 Byte / word = 3).

00000	00	00000 ……0000	1010	11	00



Indirizzamento fisico e logico



Esercizio

Memoria cache ad accesso diretto con linee di 8 word e word di 8 Byte.
Costruzione mediante 64 chip di SRAM da 1 M x 16
lw $t1, 1452($0)

Sommario

Memory miss

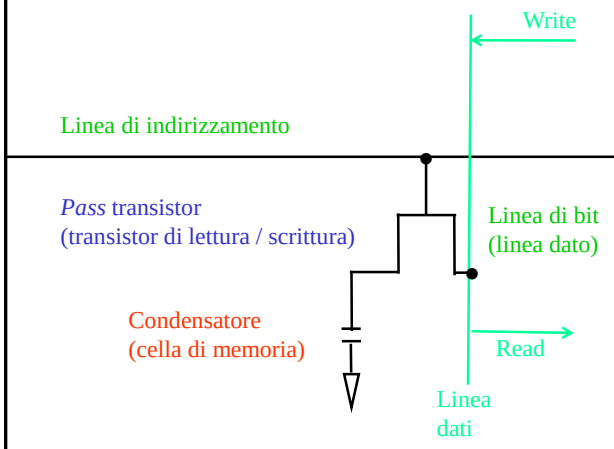
SRAM

DRAM

Trasferimento dati

Memorie DRAM

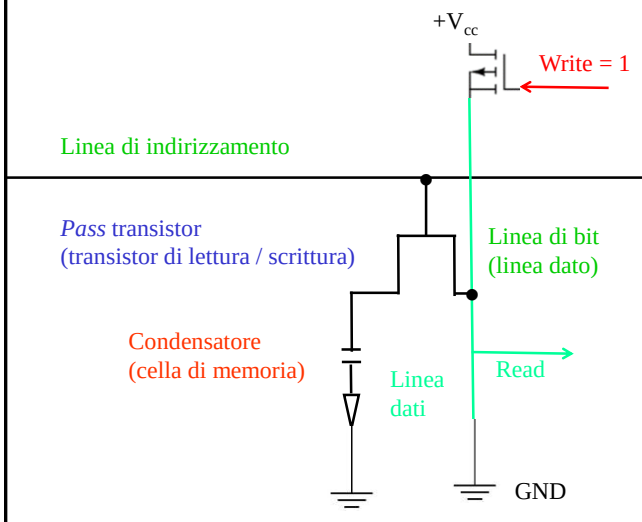
Dynamic RAM. 1 transistor per bit (contro 4-6 transistor della SRAM).



1 *Pass* transistor + 1 condensatore.

.

Memorie DRAM - scrittura

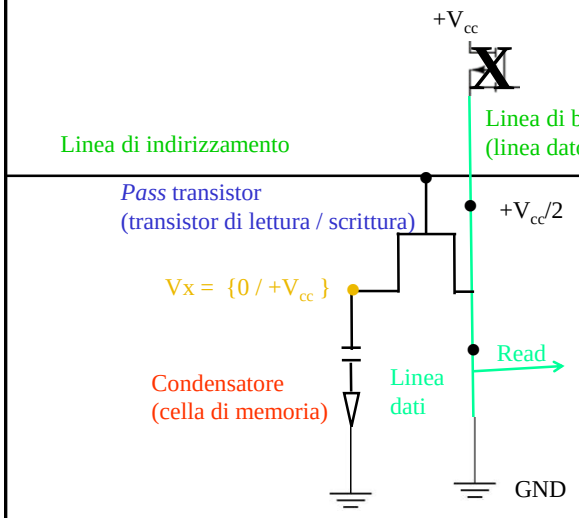


Scrittura: carica / scarica

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

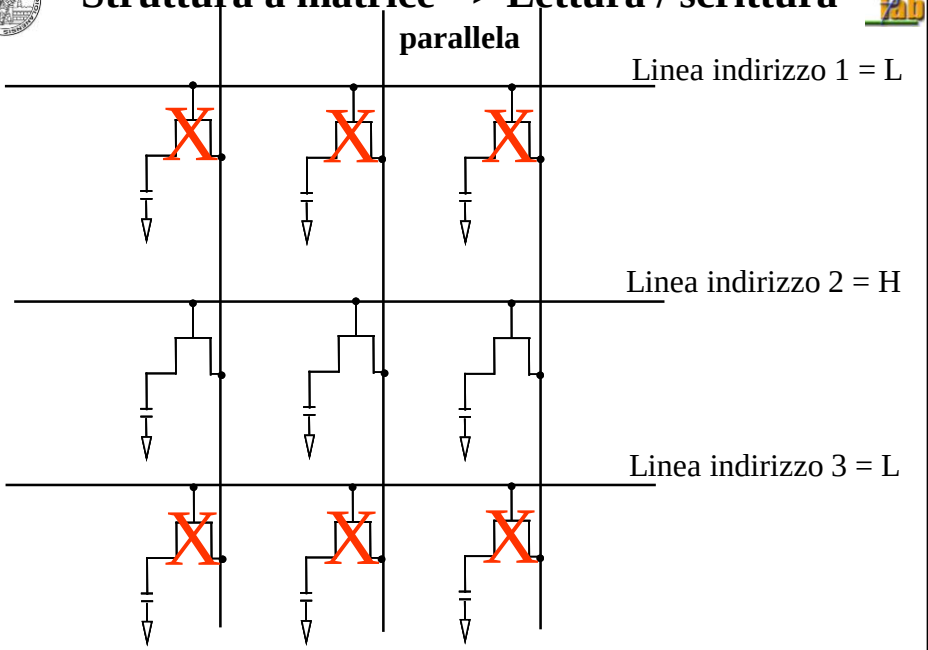
Se Vx < $V_{cc}/2$ la corrente è uscente dalla linea read.

=> 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.

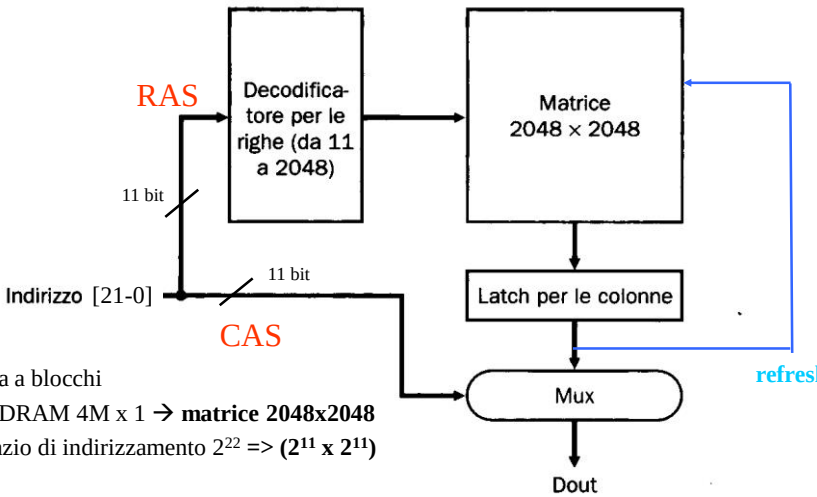
La lettura è distruttiva.

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



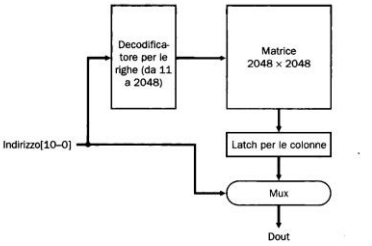
- Struttura a blocchi
  - es. DRAM 4M x 1 → **matrice 2048x2048**
  - Spazio di indirizzamento 2^{22} => **(2^{11} x 2^{11})**

•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

Tempo totale di accesso (dall'arrivo dell'indirizzo al dato in uscita): 35 ns
Tempo di accesso in modalità burst (accesso al buffer di linea di uscita): 0.8 ns

40 volte più veloce!

Organizzazione della memoria

Organizzazione a matrice (cf. cache)

Organizzazione gerarchica.

Sommario

Memory 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!!

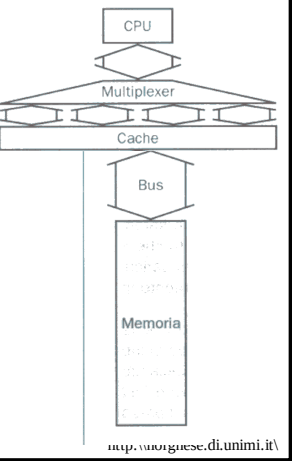
I componenti della Miss penalty

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



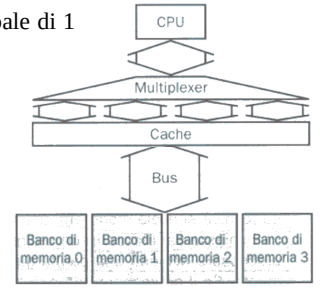
Obbiettivi:
•Diminuire la penalità di fallimento (miss_penalty).

Interleaving

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

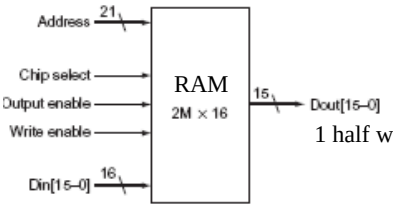
Miss_Penalty =

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

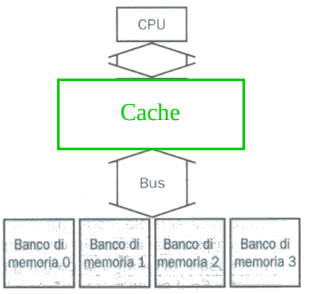


Interleaving (interallacciamento). Banchi che potrebbero essere trasferiti in parallelo alla cache. Si sposa perfettamente con la **struttura gerarchica** della memoria e con la **modalità di trasferimento a burst**.

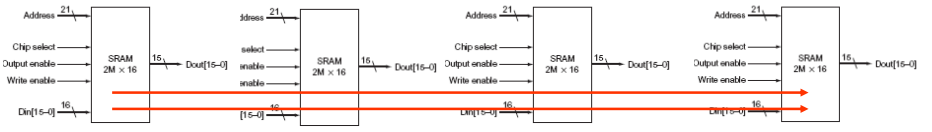
Osservazioni sull'interleaving



1 half word



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



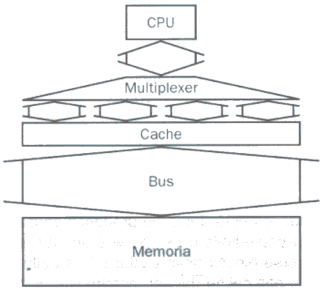
Il vettore di MM viene spalmato sulle linee.
Leggo dal MDR in uscita (velocità di 40x)

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).

Sommario

Memory miss

SRAM

DRAM

Trasferimento dati