Lezione 26

La memoria - tecnologie

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

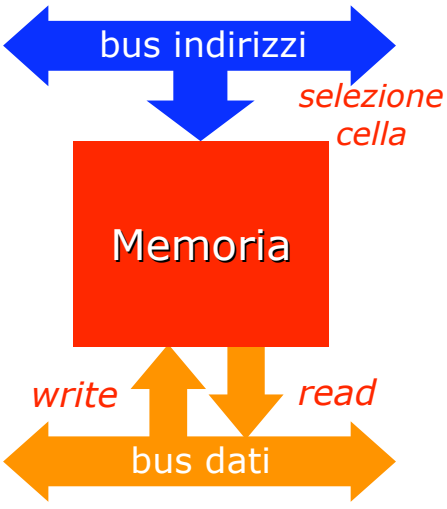
Struttura della memoria

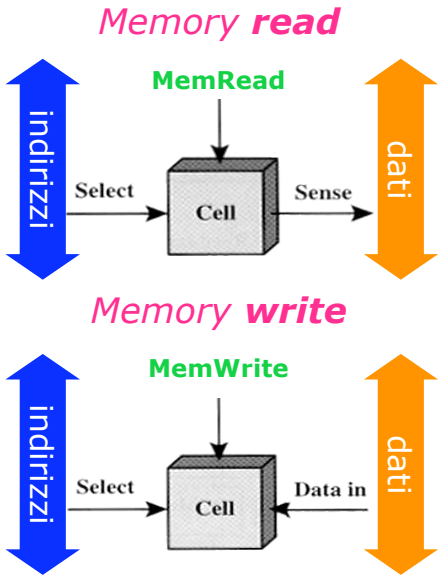
2 porte di comunicazione:

- **INDIRIZZO**: selezione di cella
- **DATI**: contenuto della cella

2 operazioni:

WRITE: scrittura nella cella
READ: lettura della cella





Organizzazione della memoria

La memoria è organizzata in ***2^K parole (word)*** di ***M bit***

M : ampiezza della memoria [bit/byte]

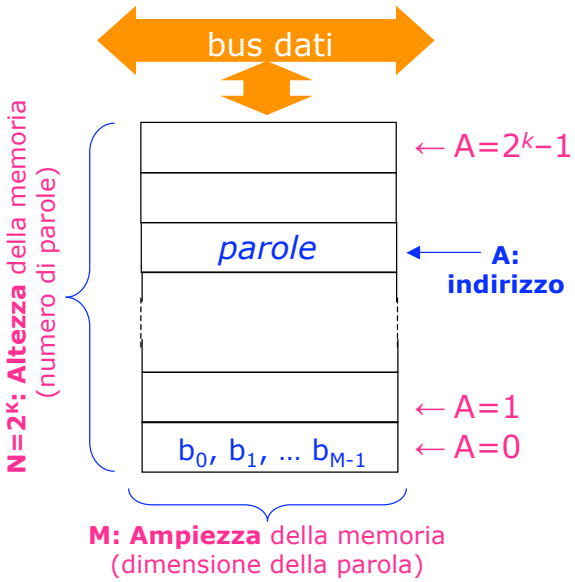
$N=2^K$: altezza della memoria [words]

- La dimensione della parola di memoria coincide con la dimensione dei registri della *CPU (CPU word)*, in modo che le operazioni di *load/store* avvengono in un singolo ciclo

C: Capacità della memoria

***C = N x M** [bytes]*

- Esempio: M=32 bit, k=32
- ***C*** $= 2^{32}$ *x 32 bit*
  $=$ *4 Gwords x 4 bytes*
  $=$ ***16 GB***



Sommario

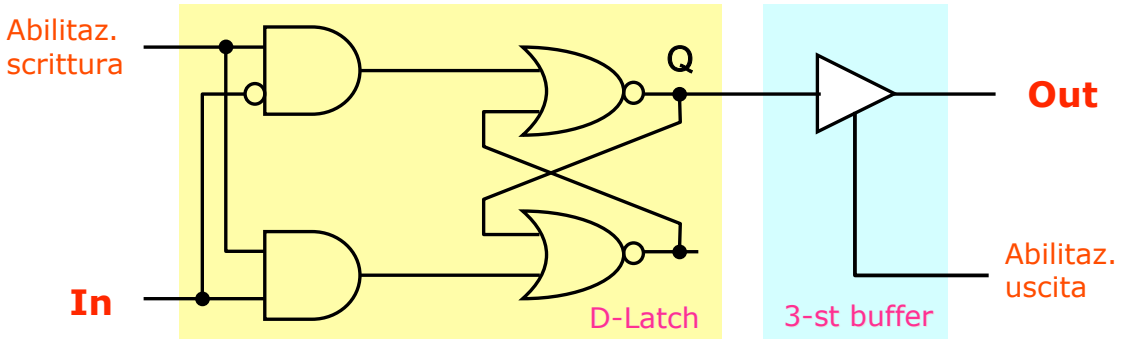
- Memorie ROM
- Memorie RAM
  - La RAM statica (SRAM)
  - La RAM dinamica (DRAM)
- Memorie con controllo degli errori (ECC)

Memorie ROM / RAM

- **Read-Only Memory, memoria di sola lettura**:
- 2^n celle, n linee di ingresso, m linee di uscita (**ampiezza**)
  - a ciascuna delle 2^n (**altezza**) configurazioni di ingresso (parole di memoria) è associata permanentemente una combinazione delle m linee di uscita
  - **Ad ogni parola corrisponde un mintermine, definito dal suo indirizzo di memoria**
- L' ingresso seleziona la parola da leggere di m bit
  - Il **contenuto della parola** di memoria corrisponde all'**uscita** relativa a tale mintermine.

  Decoder $n \rightarrow 2^n$ seguito da una matrice di m porte OR
- ***RAM – Random Access Memory***
  - **Il tempo di accesso** alla memoria è **fisso** e **indipendente dalla posizione** della parola alla quale si accede.

SRAM – RAM statica



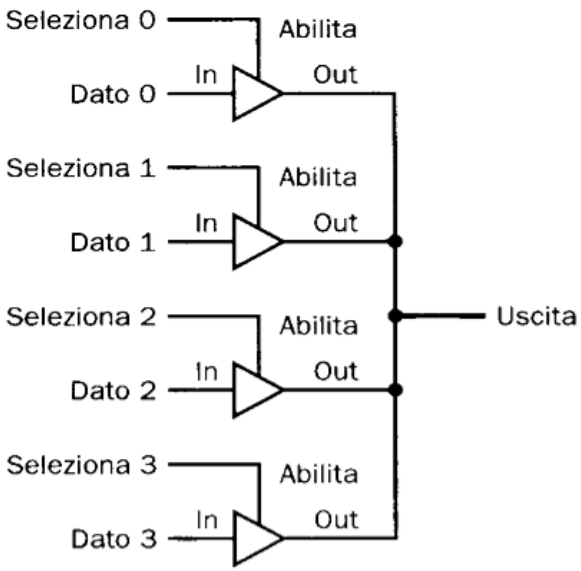
Struttura: Latch di tipo D

Velocità di una SRAM:

- Tempo di lettura: tempo di abilitazione del buffer di uscita: t_{buf}
- Tempo di scrittura: deve rispettare: $t_{set\text{-}up} + t_{hold}$
- Non si può utilizzare un array 1-D (come nel Register File)
  - Memorie di 64 kB richiederebbero un MUX a 64k vie!

Memorie: **uscita three-state**

- Si utilizza la tecnologia three-state, con un buffer three-state.
- Tutte le uscite delle celle sono collegate ad un'uscita comune
  - necessario evitare conflitti fra le uscite
  - → **uscite "isolate" con porte *three-state***
  - → **seleziono una sola cella alla volta**

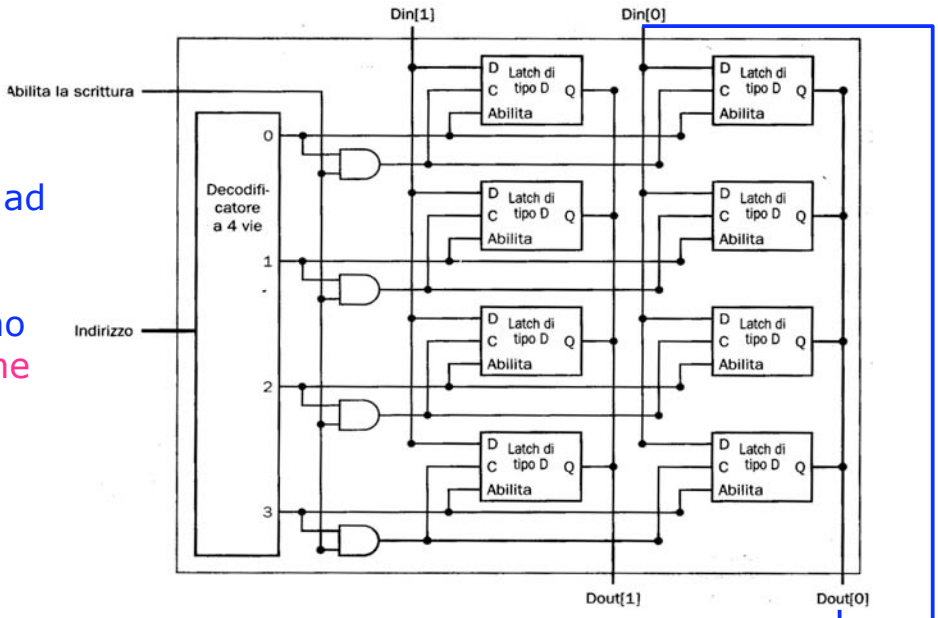


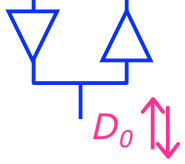
Esempio di SRAM

Esempio:
SRAM
4 celle x 2 bit

Struttura simile ad un Register File

Le linee dati sono spesso in comune





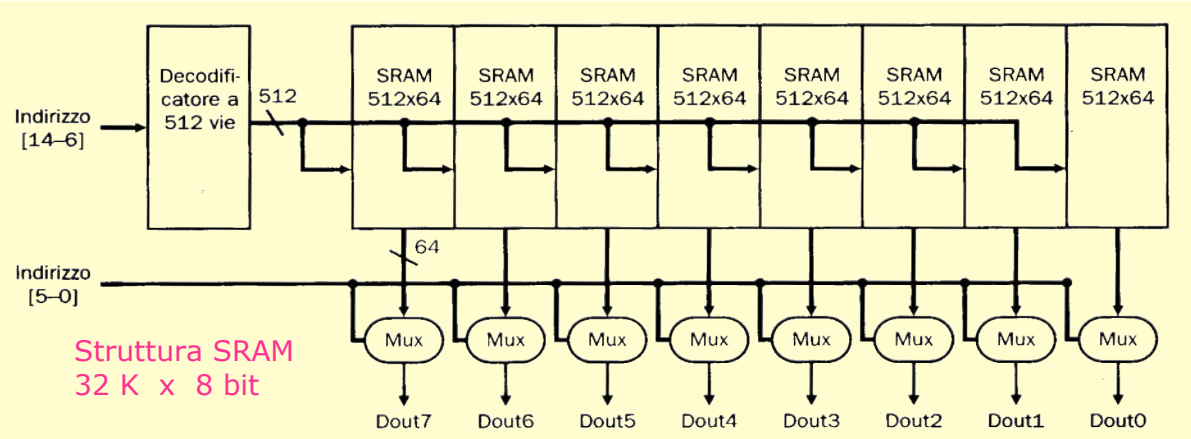
- Problema: es. SRAM 32K x 8 bit
  - Decodificatore a 15 bit → **32 K** linee di abilitazione, **32 K** uscite

Indirizzamento SRAM a banchi

- Esempio: **RAM 32 k x 8 bit** → 8 x (32 k x 1 bit)

  Considero che: **32 k x 8 bit = 512 x 512 bit**

  - Per ogni bit di ampiezza, ho 512 banchi di 64 bit
    - 1 DEMUX a 15 bit (32 K uscite)
    - 1 DEMUX a 9 bit (512 uscite) + 8 MUX a 6 bit → (64 ingressi)



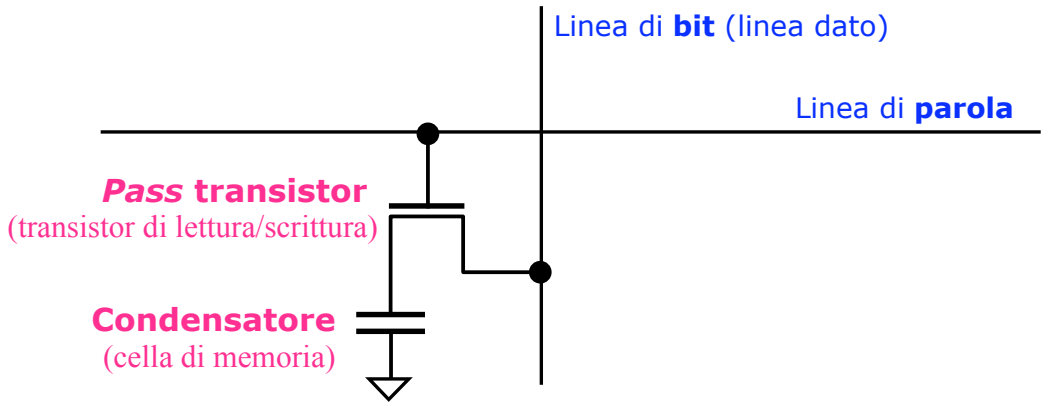
Sommario

- RAM Statiche (SRAM)
- RAM Dinamiche (DRAM)
- Correzione degli errori

Memorie RAM dinamiche (DRAM)

Funzionamento:

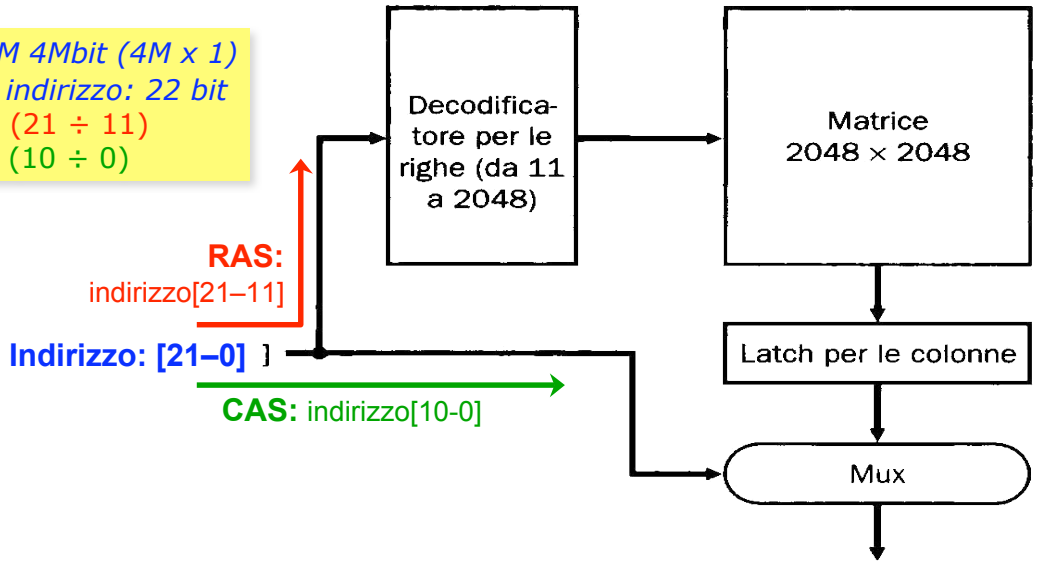
- Elemento di memoria: carica di un condensatore
  - condensatore CARICO → 1
  - condensatore SCARICO → 0
- Cella DRAM: **1 transistor + 1 condensatore**
  - La lettura scarica la memoria che deve essere ricaricata
  - Necessario un refresh periodico (gestito autonomamente dal controllore della memoria)



Struttura di una DRAM

- Struttura a blocchi, "**quadrata**" (come SRAM):
- **Capacità DRAM: 2^{2N} bit → matrice quadrata** di **2^N x 2^N bit**
- Accesso a bit: selezione **riga (N bit):** segnale **RAS** (Row Access Strobe)
  selezione **colonna (N bit)**: segnale **CAS** (Column Access Strobe)

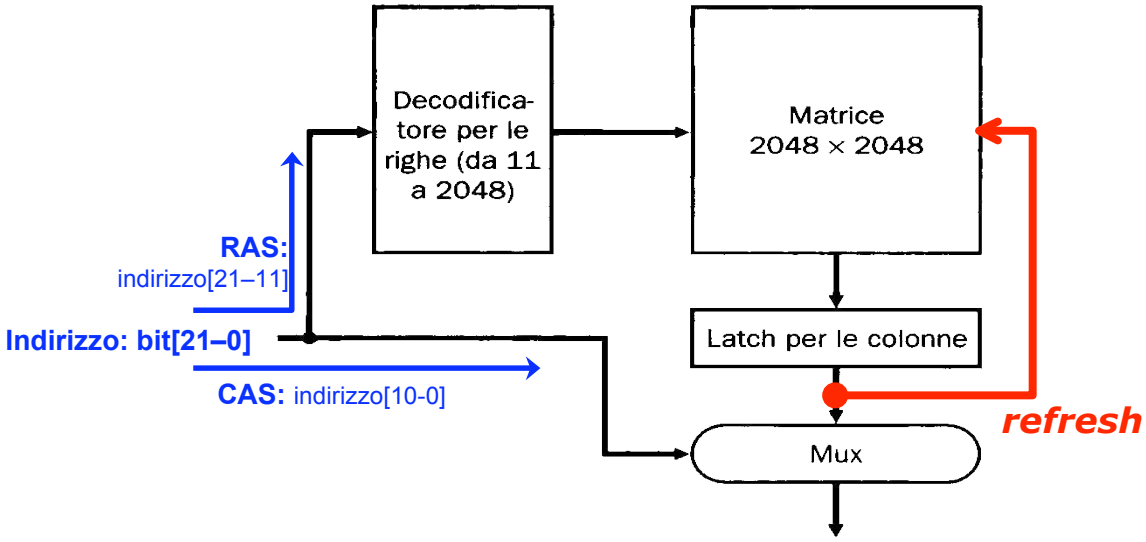
Esempio: DRAM 4Mbit (4M x 1)
4 M = 2^{22} → indirizzo: 22 bit
→ **RAS**: 11 bit (21 ÷ 11)
→ **CAS**: 11 bit (10 ÷ 0)



Struttura di una DRAM: refresh

Refresh di una RAM dinamica:

- Problema: il condensatore della cella si scarica in 30÷70 msec !!!
  - entro tale tempo devo riscrivere ("rinfrescare") il dato nella DRAM
  - REFRESH: ad ogni lettura di riga



Struttura DRAM: esempio

- *Frequenza di refresh di una RAM dinamica: frequenza di ripetizione delle operazione di refresh*

Esempio:

- *In una RAM dinamica di 4 M x 1 bit il tempo di scarica dei condensatori è di 64 millisecondi.*
  Calcolare la minima frequenza di refresh.

- Una memoria di 4 Mbit (2^{22}) è organizzata come matrice di 2^{11} = 2048 x 2048 celle.
- Ogni ciclo di refresh rigenera una riga; la stessa riga deve essere di nuovo "rinfrescata" dopo al più 64 ms. Quindi in 64 ms devo rigenerare 2048 righe.

$$T_{REFRESH,\ MAX} = 64\ ms / 2048 = 31.25\ \mu s$$
$$f_{REFRESH,\ MIN} = 1 / T_{REFRESH,\ MAX} = 32\ kHz$$

Evoluzione delle SRAM e DRAM

- Trasferimento a burst o a pagina: trasferimento consecutivo di parole ad indirizzi consecutivi.

- Synchronous DRAM (**SDRAM**)
  - L'accesso alla memoria è sincrono con il clock dato dalla CPU (mem. bus)
  - 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 può eseguire altri compiti (il processore può essere la CPU o il controllore della memoria, o altro: il dispositivo che controlla la memoria).

- **DDR-SDRAM** (Double-Data-Rate)
  - Riescono 2 trasferimenti per ciclo di clock.
  - Data-rate doppio rispetto alla frequenza del clock del bus.

Alcuni dati: aprile 2004

- *Da: http://www.samsung.com/Products/Semiconductor*
- **SRAM**
  - Sincrone, 1M x 36, 2M x 18, tempi di accesso: 2,6ns
  - High speed, 1M x 18 o 512K x 36, tempi di accesso: 1,6ns
  - Asincrone: 8M x 16, tempi di accesso: 10ns
  - Low power, 8M x 16 tempi di accesso: 70ns

- **DDR–SDRAM**
  - 128M x 8, rate: **266Mb/s** (133Mhz → T_C = 7,5 ns)
  - 16M x 16, rate: **400Mb/s** (200Mhz).
  - 3 clock di latenza → 7,5*3 = **22,5 ns**
  - 2-4-8: larghezza del burst

  Attualmente:
  - 1 GB DDR2 800 MHz burst rate: 6400 MB/s (128 M x 64 bit)

Prestazioni di una memoria

- Parola di memoria vs. Unità indirizzabile
  - Parola di memoria:
    L'unità naturale in cui la memoria viene organizzata (MIPS: 32 bit)
  - Unità indirizzabile: il minimo numero di unità contigue indirizzabili.
    In quasi tutti i sistemi si tratta del byte.

- Tempo di accesso (***access time***):
  - tempo richiesto per eseguire una lettura/scrittura:
    dall'istante in cui l'indirizzo si presenta alla porta di lettura ...
    ... all'istante in cui il dato diventa disponibile.

- *Memory **cycle time**:*
  - per memorie ad accesso casuale: è 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 [MB/s]
  - Random-access memory: $R = 1 / Memory\_cycle\_time$
  - Sequential memory: $R = 1 / [TA + N\ T_{TR}]$ T_{TR} : tempo di trasferim. N bytes

Sommario

- SRAM.
- DRAM.
- Correzione degli errori

ECC – Error Correction Codes

- 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 di controllo errori
  - 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.

Codici rivelatori d'errore

- **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
  - mi accorgo dell'errore, ma non so dov'è
  - **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: $r = d_{MIN} - 1$
- Capacità di correzione di un codice: $t = (d_{MIN}-1)/2$

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

Applicazioni nelle memorie

- RAM con controllo di parità
  - Aggiungo un bit di parità ad ogni byte

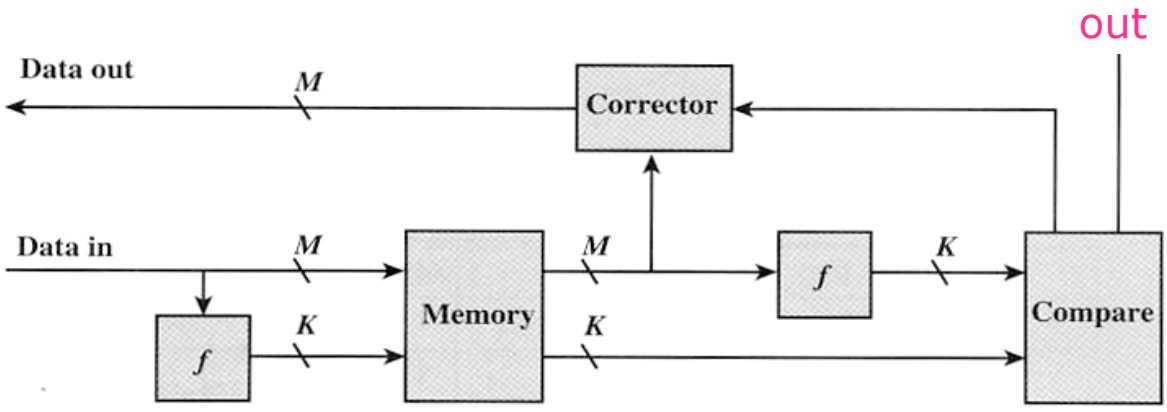
  Es: RAM 1 M x 9 bit (8+1)

  Detta p la probabilità che un bit venga sbagliato,
  - Probabilità di **un errore** in una parola: $P_1 = 9p(1-p)^8 \approx 9\,p$
  - Probabilità di **due errori** nella parola: $P_2 = 36p^2(1-p)^7 \approx 36\,p^2$

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

Correzione degli Errori

- **ECC memory –** possibili casi:
  - No errors detected: i dati letti possono essere inviati in uscita così come sono.
  - 1 errore 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: impossibilità di recupero – si segnala la condizione d'errore (eccezione!).



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