

# LE MEMORIE

Prof. CAPEZIO Francesco

Quest'opera è soggetta alla licenza **Creative Commons**  
*[Attribuzione – Non Commerciale](#)*



# Introduzione

Le memorie di un computer possono essere divise tra **centrali** e **secondarie**. La memoria centrale di un computer è anche detta **memoria di lavoro** e, sostanzialmente, è quella sulla quale la CPU elabora le istruzioni necessarie all'esecuzione dei programmi.

Nella memoria di lavoro possono essere presenti le **istruzioni** dei programmi e i **dati** necessari all'elaborazione.

Possiamo dunque considerare memoria di lavoro la **memoria principale** (comunemente detta RAM) e la memoria **CACHE**.

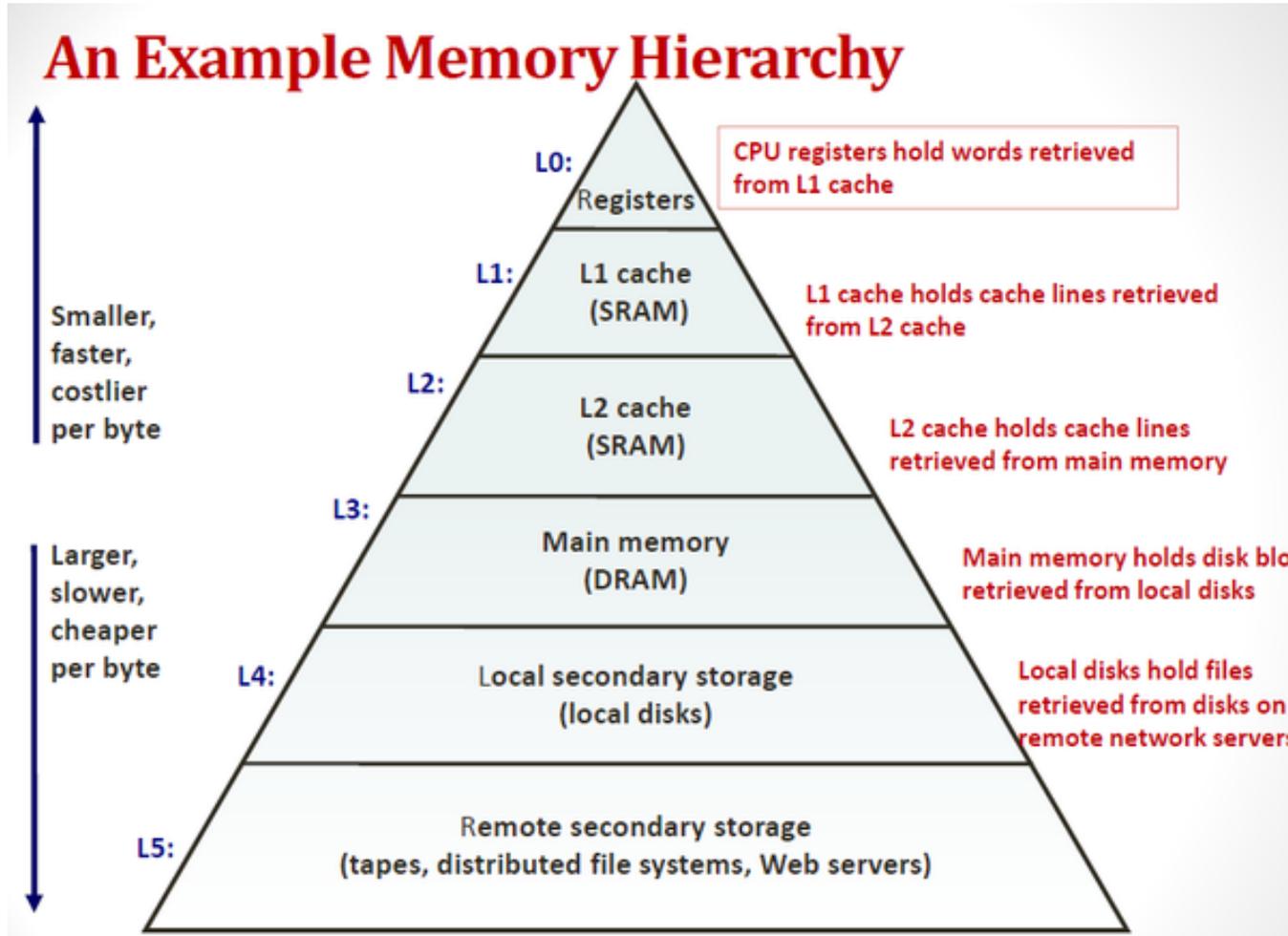
# Introduzione

Le memorie **secondarie** sono invece quelle comunemente usate per contenere tutti i dati e i programmi che possono essere utili al funzionamento del computer e dunque l'Hard Disk Drive (HDD) il Solid State Disk (SSD) e i dispositivi rimovibili.

Le più grandi differenze tra queste due memorie sono:

- Le memorie secondarie sono lente, economiche e permanenti (i dati restano salvati anche a computer spento)
- Le memorie centrali sono veloci, costose e volatili (i dati vengono persi quando si spegne il computer)

# GERARCHIA DELLE MEMORIE



Le memorie di un computer sono diverse, da quelle lente come gli HDD ai registri del processore (piccole memorie che contengono singole istruzioni o singoli dati).

E' possibile classificare queste memorie tramite diversi schemi, il più comune dei quali è una piramide.

Alla base della piramide ci sono le memorie più lente e che sono comunemente più grandi (costando meno)

In cima alla piramide ci sono le memorie molto veloci, comunemente piccole di dimensioni

# GERARCHIA DELLE MEMORIE

MEMORIA	TEMPO DI ACCESSO AL DATO	DIMENSIONE TIPICA	VOLATILITA'	CONTENUTO
REGISTRI	Istantanei	Pochi Byte	Volatile (memoria del processore)	Il dato o l'istruzione utilizzata in questo istante
CACHE L1	1 nSec	8 MB	Volatile (memoria del processore)	Dati e istruzioni frequentemente utilizzati
CACHE L2 e L3	5 nSec	64 MB	Volatitè (memoria centrale)	Dati e istruzioni frequentemente utilizzati
MEMORIA PRINCIPALE	50 nSec	8 GB	Volatile (memora centrale)	Dati e istruzioni in esecuzione
SSD	0,1 mSec	250 GB	Permanente (mem. Secondaria)	Tutti i dati e i programmi
HDD	10 mSec	1 TB	Permanente (mem. Secondaria)	Tutti i dati e i programmi

# L'HARD DISK DRIVE (HDD)

Tra le memorie secondarie la più utilizzata è l'HDD (recentemente sostituito dal Solid State Disk – SSD).

L'HDD è composta da un **insieme di dischi** (detti piatti) di materiale ferromagnetico, che vengono scritti e letti da una **testina** in grado di magnetizzare (con carica positiva o negativa) il materiale attraverso il passaggio di corrente.

La **formattazione** di un HDD è l'operazione necessaria per creare una struttura in grado di facilitare la divisione dei dati all'interno del disco

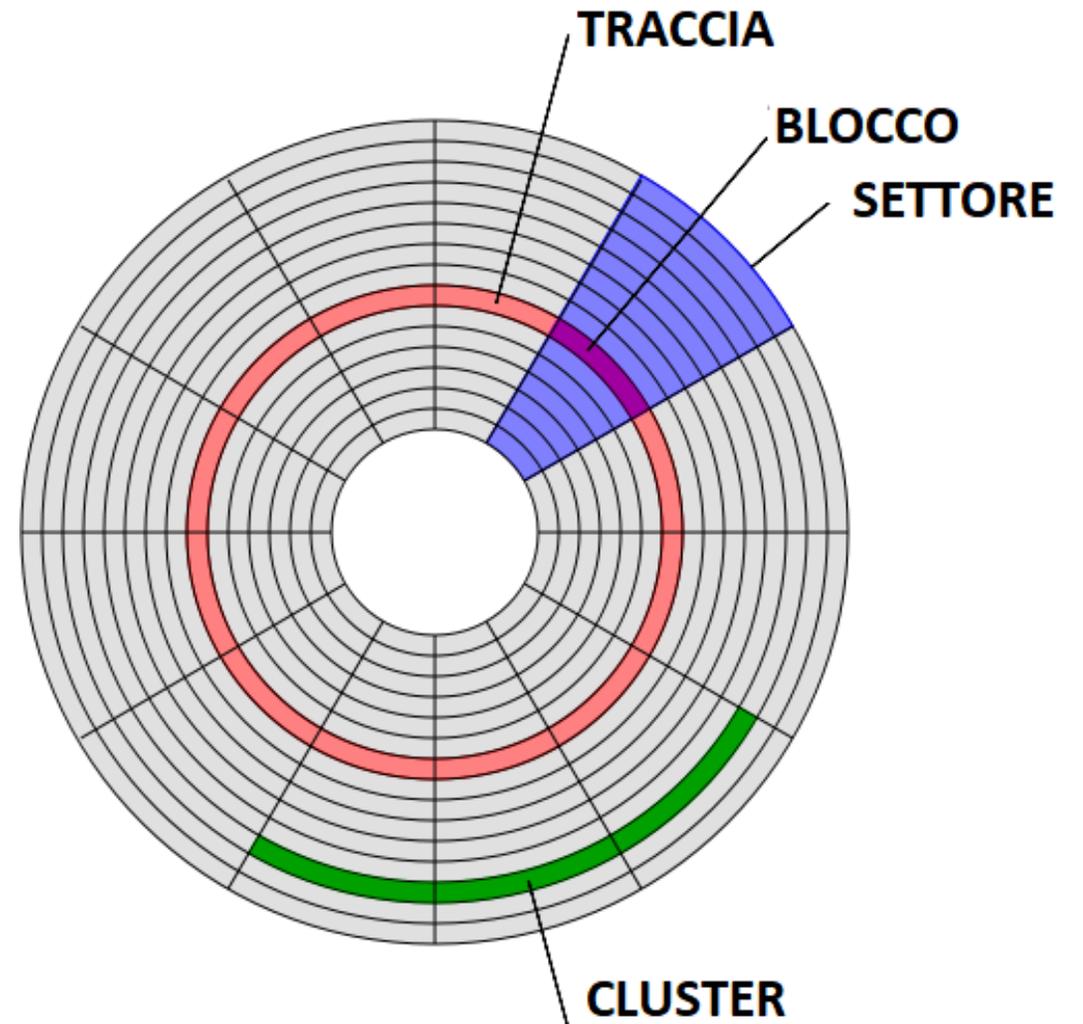
# L'HARD DISK DRIVE (HDD)

**Tracce:** sono circonferenze a diversa distanza dal centro. (16 mila tracce)

**Settore:** E' una porzione del di disco delimitata da un certo angolo (spicchio). (16 settori)

**Blocco:** è la porzione di traccia contenuta in un settore

**Cluster:** è un insieme di blocchi contenente lo stesso dato



# L'HARD DISK DRIVE (HDD)

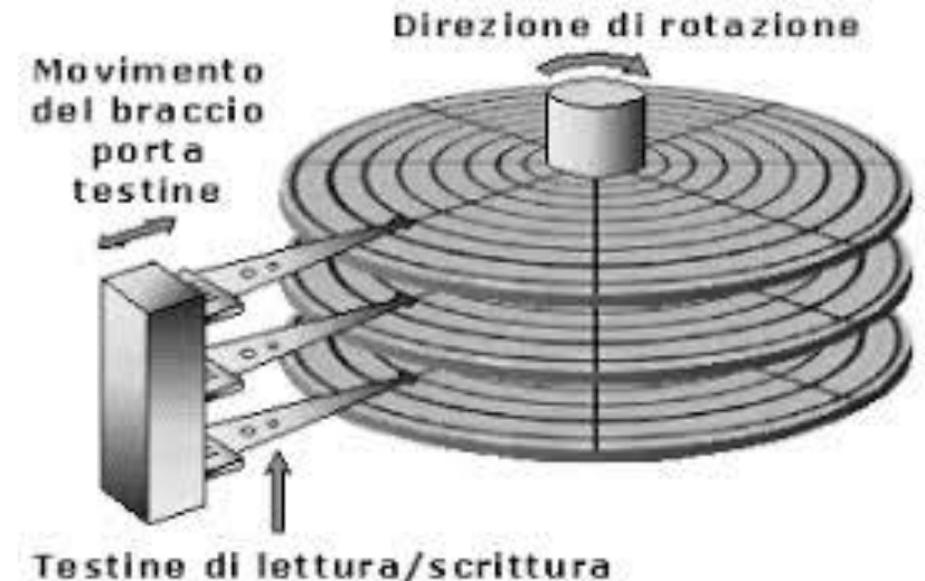
La scrittura di un HDD avviene in realtà in modo **parallelo** su ogni piatto: le testine si muovono sincronizzate in modo da dividere il dato sui diversi dischi, diminuendo il tempo di scrittura totale.

Blocchi e cluster di un HDD sono in effetti considerati l'insieme su ogni piatto.

La scelta del settore deriva dalla rotazione del disco (**tempo di latency**)

La scelta della traccia deriva dal movimento della testina (**tempo di seek**)

Un parametro importante sulla velocità di un disco è la velocità di rotazione: tipicamente **7200 rpm**



# MEMORIA CENTRALE

La memoria centrale di un computer è anche detta **memoria di lavoro** e, sostanzialmente, è quella sulla quale la CPU elabora le istruzioni necessarie all'esecuzione dei programmi.

Nella memoria di lavoro possono essere presenti le **istruzioni** dei programmi e i **dati** necessari all'elaborazione.

Possiamo dunque considerare memoria di lavoro la **memoria principale** (comunemente detta RAM) e la memoria **CACHE**.

Includiamo tra le memorie centrali anche la memoria ROM, contenente il BIOS della scheda madre

# MEMORIA PRINCIPALE e MEMORIA CACHE

- La memoria principale è quella comunemente chiamata RAM
- È la memoria nella quale la CPU legge istruzioni e dati dei programmi in esecuzione
- Per velocizzare l'esecuzione delle istruzioni quelle **più frequentemente utilizzate** vengono scritte nella CACHE, una memoria molto veloce ma costosa (dunque piccola)
- La memoria CACHE è divisa in tre livelli: il primo è solitamente integrato nel processore (dimensione tipica 64MB)
- Quando la RAM trova l'istruzione desiderata in CACHE ha fatto un **hit**, quando non la trova è un **miss** e l'istruzione viene cercata nella RAM

# MEMORIA CACHE

Quali sono le istruzioni più frequentemente utilizzate?

## **principio di località**

- **Località temporale:** le istruzioni usate recentemente è probabile che vengano riutilizzate a breve (rispetto ad istruzioni che usate molto tempo prima)
- **Località spaziale:** le istruzioni «vicine» (nell'esecuzione del programma) a quella attuale è probabile vengano utilizzate a breve.

Una buona gestione di questa politica di scelta delle istruzioni **aumenta la probabilità di hit intorno all' 85% - 90 %**, diminuendo dunque il **tempo medio di accesso all'istruzione:**

$$T_a = P * T_c + (1 - P) * T_r$$

T<sub>a</sub>: tempo medio di accesso

P: probabilità di hit

T<sub>c</sub>: tempo di accesso alla cache

T<sub>r</sub>: tempo di accesso alla ram

# Memorizzazione di un bit

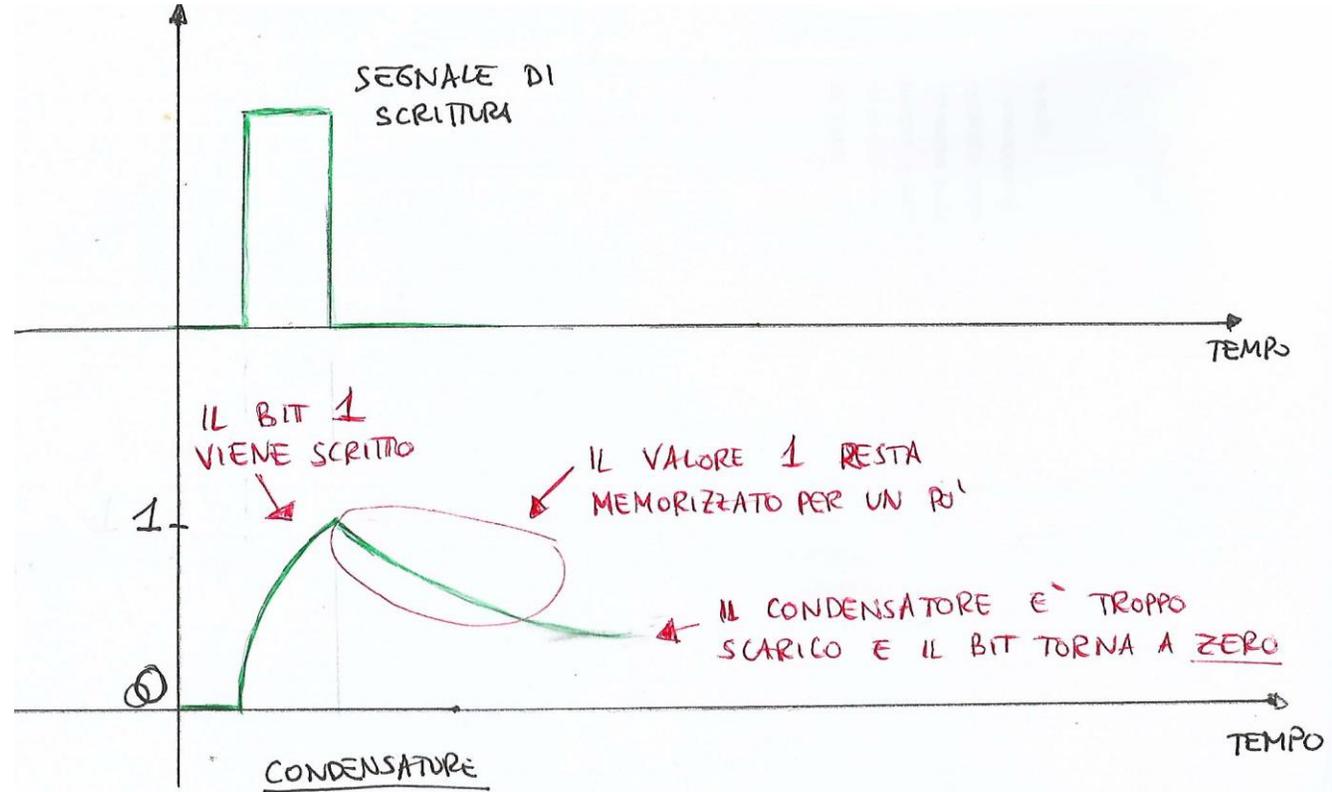
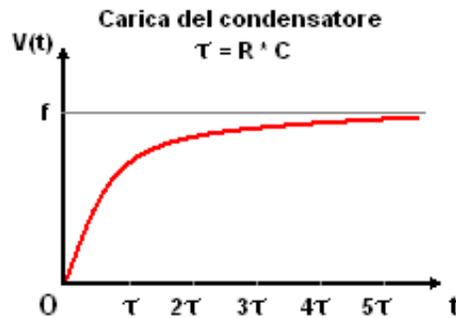
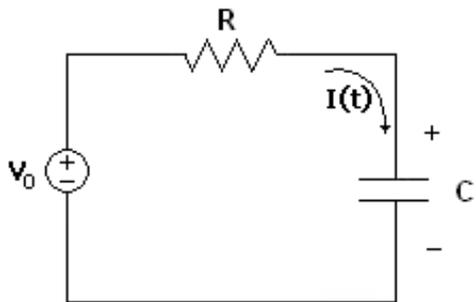
Memorizzare un bit significa impostare un valore 1 oppure 0 in un dispositivo e garantire che questo valore resti invariato anche terminata la scrittura.

Noi vedremo solo due metodi per memorizzare: tramite un **condensatore** e tramite un circuito detto **Latch**

# Condensatore

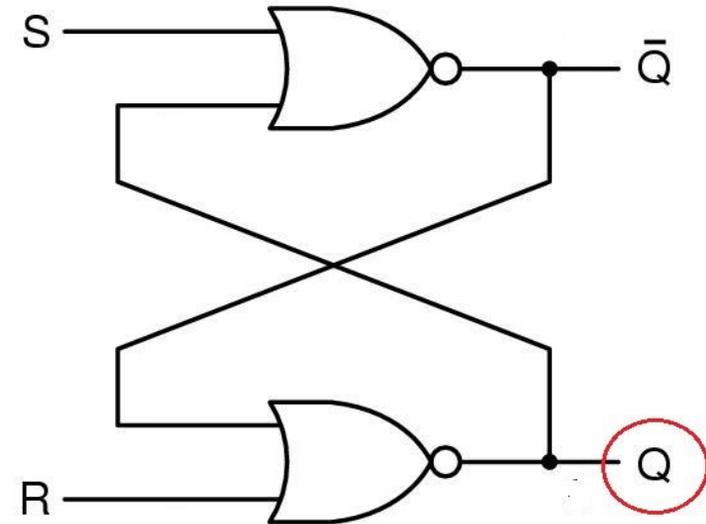
- Il condensatore è un dispositivo in grado di **immagazzinare della carica elettrica**.
- Il condensatore si carica applicando una tensione ai suoi capi.

Tolta la tensione, di carica il condensatore resta carico per un po' per poi scaricarsi: in questo modo ho "memorizzato" il bit 1 (Condensatore carico)

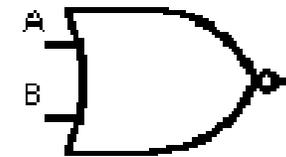


# Circuito di Latch-SR

- Il **Latch Set-Reset** è un circuito logico **bistabile** costituito da due porte **NOR**.
- Ha due ingressi: **S** (set) e **R** (reset)
- Ha due uscite ma ne consideriamo solo una: **Q**
- Quando gli ingressi sono a zero **ho due stati stabili**: l'uscita Q può essere 1 o 0 (situazione di memorizzazione)
- I due stati stabili possono essere cambiati utilizzando gli ingressi S e R



La porta NOR è il negato della porta OR, fornisce 0 quando almeno uno degli ingressi è 1 e fornisce 1 se entrambi gli ingressi sono 0



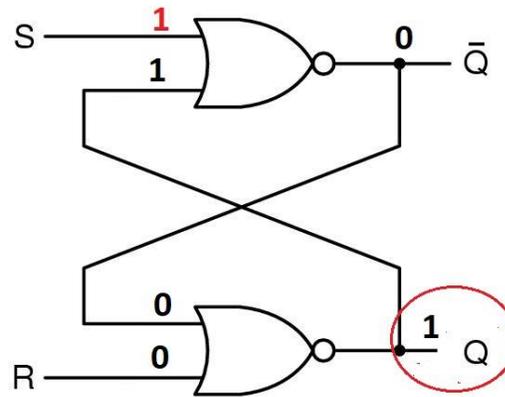
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

# Circuito di Latch-SR

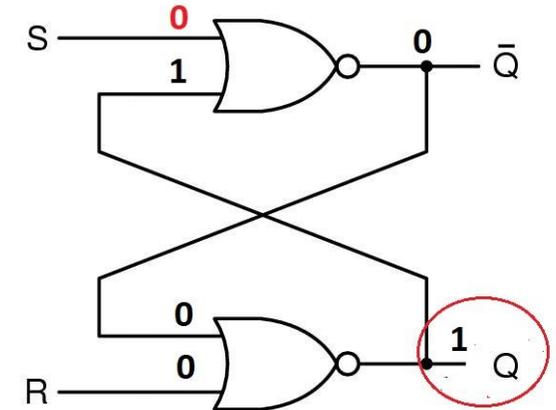
- Lo stato iniziale del circuito non può essere definito a priori (può essere uno dei due stati stabili)
- L'ingresso S forza il bit Q a 1
- L'ingresso R forza il bit Q a 0
- Gli ingressi a zero non cambiano lo stato
- Non è previsto lo stato S=1 R=1

S	R	Q	$\bar{Q}$	Q	$\bar{Q}$	funzione
0	0	1	0	0	1	Memorizza
1	0	1	0	1	0	Set
0	1	0	1	0	1	Reset
1	1	1	1	0	0	Non ammesso

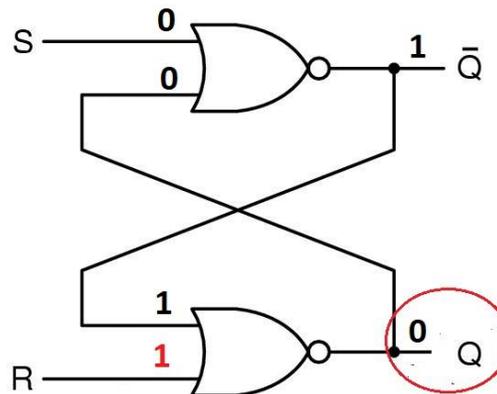
**SET: Scrittura del bit a 1**  
Ingresso S = 1 (R=0)



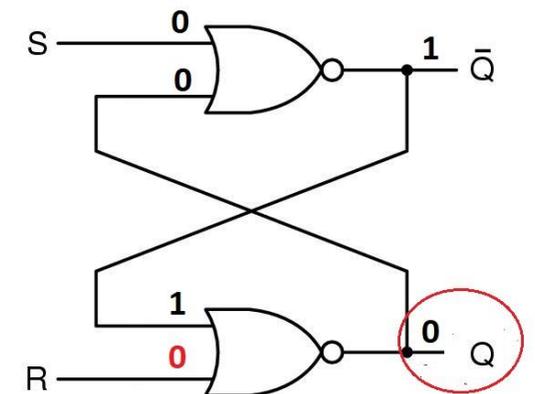
**Memorizzazione**  
Riporto S=0



**RESET: Scrittura del bit a 0**  
Ingresso R = 1 (S=0)



**Memorizzazione**  
Riporto R=0



# DRAM – Dynamic RAM

La **DRAM** è un tipo di memoria basata sull'utilizzo di **microcondensatori** (comandate da transistor)

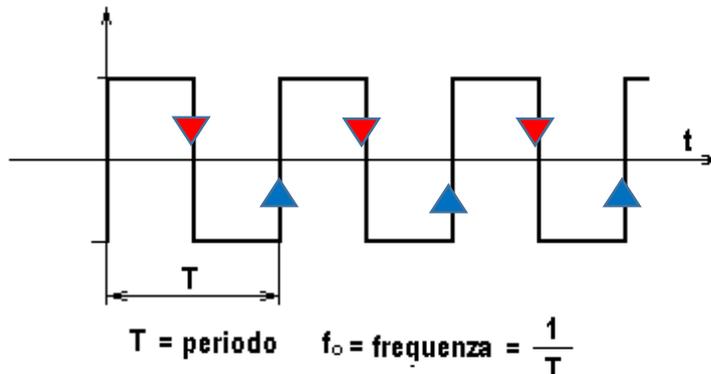
- Ogni condensatore memorizza un bit
- Il condensatore tende a scaricarsi ed è necessario, periodicamente, fare un **refresh** della memoria
- Il refresh **rallenta** l'esecuzione dei programmi
- è una memoria relativamente poco costosa



**Memoria Principale  
(RAM)**

La **SDRAM (Synchronous DRAM)** esegue il refresh della memoria in modo sincrono con il lavoro della CPU (gestito da un segnale di **clock**). In questo modo velocizzo l'esecuzione delle istruzioni.

Segnale di  
clock



▼ La CPU legge il contenuto della cella

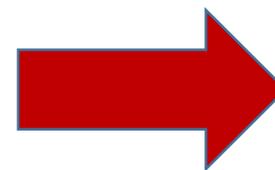
▲ La SDRAM esegue il refresh

P.S. La frequenza della CPU (es 2,2GHz è proprio la frequenza di questo segnale (nell'esempio: 2.2miliardi di oscillazioni al secondo, ocorrispondente, semplificando, a 2.2 miliardi di operazioni al secondo

# SRAM – Static RAM

La **SRAM** è un tipo di memoria basata sull'utilizzo di **flip-flop** circuito simili al **latch**

- Ogni flip-flop memorizza un bit
- Un flip-flop è composto da 6 transistor
- Ne consegue che la memoria è più **costosa** e più **piccola**
- Il circuito non perde il dato e non ha bisogno di refresh dunque è molto più **veloce**



**Memoria  
CACHE**

# Indirizzamento Memoria CACHE

Come detto, la memoria **cache** contiene alcuni **blocchi** della memoria principale, in particolare quelli utilizzati più di frequente.

Quando al **CPU** fa **riferimento** ad un indirizzo della memoria principale, si procede prima ad una ricerca di quell'indirizzo nella memoria cache. Questo può dare un **hit** o un **miss**.

Il procedimento di ricerca dell'indirizzo nella memoria cache è **trasparente** all'utente, ovvero nella programmazione non è necessario preoccuparsi di dove siano allocati gli indirizzi (tra memoria principale e memoria cache)

Esistono tre principali modi di **indirizzare** la memoria cache:

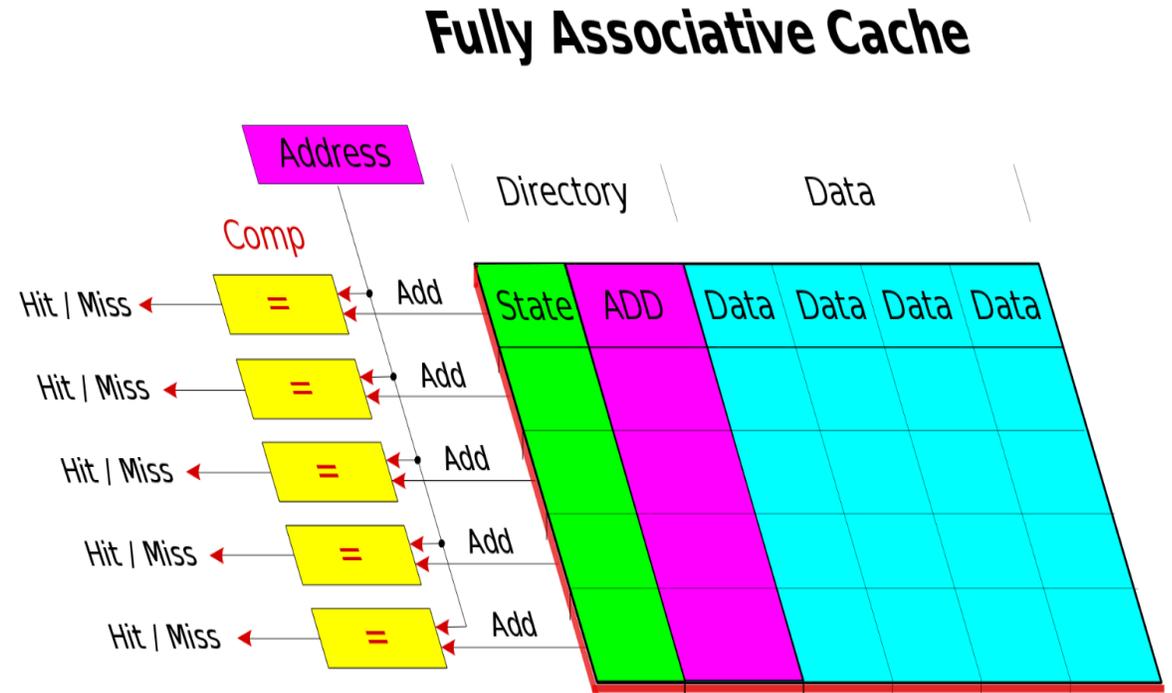
- Indirizzamento **Fully Associative**
- Indirizzamento **Direct Mapping**
- Indirizzamento **Set Associative**

# Indirizzamento Fully Associative

Ogni blocco della memoria principale può essere messo in **uno qualsiasi dei blocchi** della cache.

Quando si fa riferimento ad un indirizzo è necessario **controllare tutta la cache** per vedere se quell'indirizzo è contenuto.

Si controlla quindi un **COMPARE** tra l'indirizzo cercato e un campo **ADD** (address) di ogni cella della cache. Se uno dei confronti dà un HIT (successo) allora l'indirizzo cercato era nella cache. (ovviamente bisogna che sia valido, ovvero che il flag STATE sia a 1)



Questo metodo è molto **flessibile** (posso mettere i blocchi della RAM nella cella che risulta più adatta) ma rallenta la ricerca in cache (aumenta il tempo di accesso alla cache) in quanto è necessario controllare tutti i blocchi alla ricerca di quello giusto)

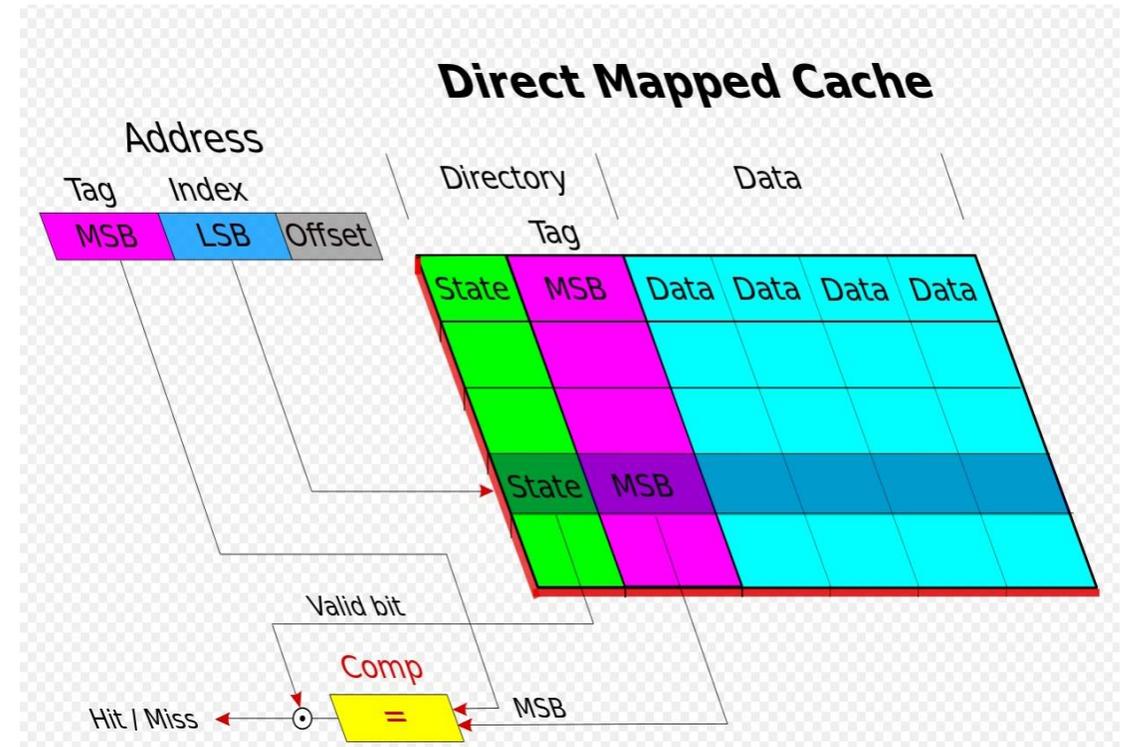
# Indirizzamento Direct Mapping

Un blocco della memoria principale (RAM) può essere messo solo nel blocco della cache identificato dalla parte **meno significativa** dell'indirizzo (**LSB**). Ad esempio i blocchi con indirizzo **3240** **5140** e **7240** potranno essere messi solo nell'indirizzo 40 della cache (immaginando una cache a 100 indirizzi, da 00 a 99).

Quando la CPU fa riferimento ad un certo indirizzo si va a controllare **solo un blocco** della CACHE (quello relativo al LSB) e si controlla se la parte più significativa dell'indirizzo (MSB) corrisponde.

Si può avere un HIT o un MISS

Questo metodo è molto meno **flessibile**: il blocco della RAM può essere messo **solo in una posizione** della cache, indipendentemente se ce ne sono altre migliori (ad esempio perchè libere). L'accesso alla Cache è però più **veloce** (tempi di accesso minori) in quanto si va a controllare un solo blocco e non tutta la Cache.



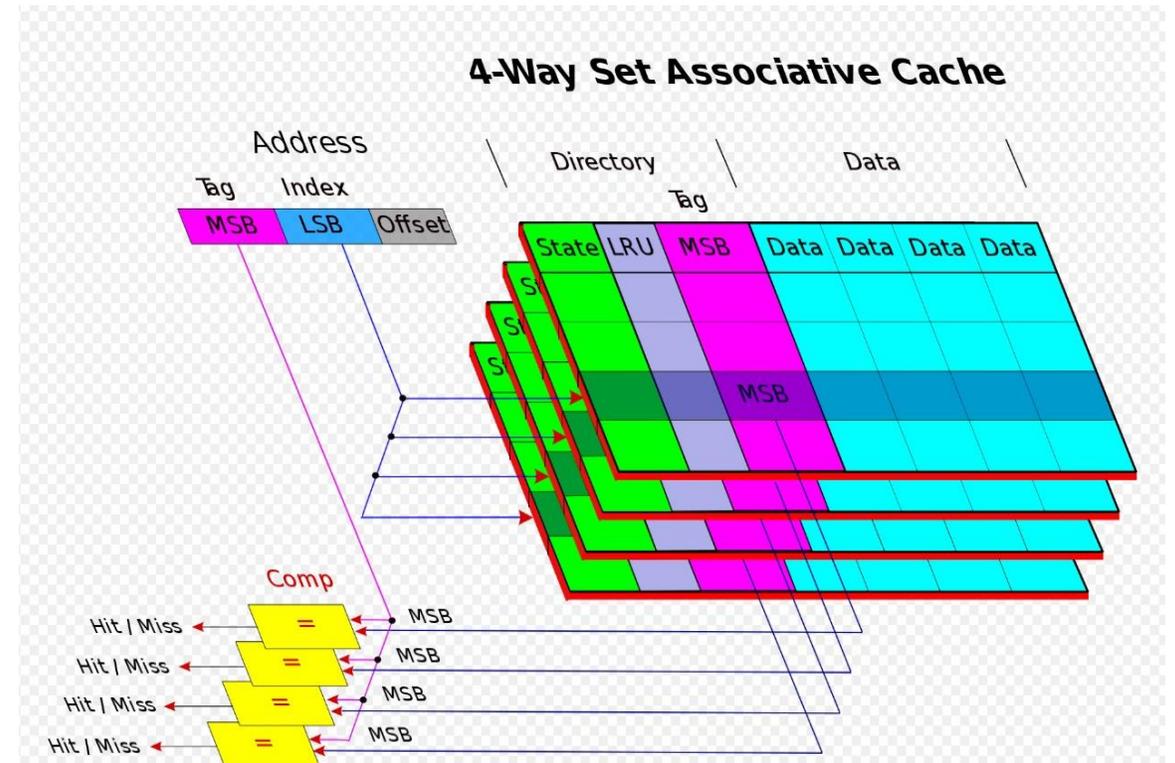
# Indirizzamento Set Associative a N-vie

Questo metodo è un mix dei due precedenti.

Si divide la memoria cache in diverse porzioni (vie) ad esempio 4 come in figura.

Ogni via è gestita come nel **Direct Mapping** ma per ogni blocco della memoria principale ho N posizioni a disposizione (ad esempio 4 come in figura) nella Cache, in quanto ogni porzione ripete gli indirizzamenti.

Facendo riferimento all'esempio precedente si avrebbero N posizioni «40» in cui poter mettere il blocco della cache



Questo metodo unisce le caratteristiche dei due precedenti e quindi è più flessibile del Direct Mapping puro e più veloce del Fully Associative puro