**MEMORIA RAM (RANDOM ACCESS MEMORY)**

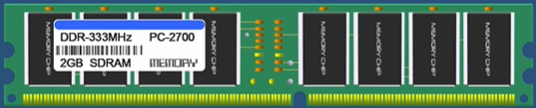
La memoria RAM è la memoria principale del PC, sul quale la CPU va a cercare la maggior parte dei dati o programmi che gli servono in ogni istante.

Esiste una gerarchia di memoria all’interno di un PC.

È **volatile**, cioè perde il contenuto informativo se non c’è alimentazione.

La RAM si divide in:

* **Statica SRAM**
* **Dinamica DRAM**

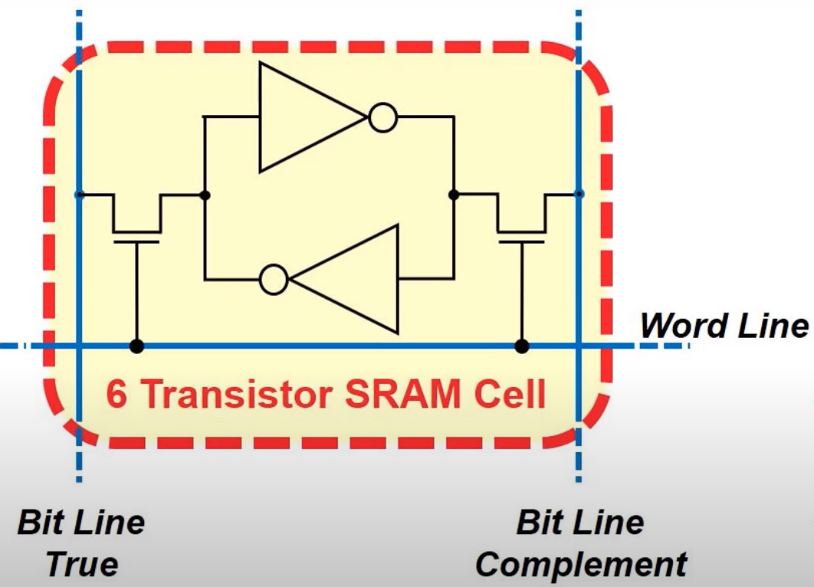


Il numero dopo DDR indica la frequenza di operatività, quello dopo PC indica il larghezza di Banda = bitrate

**SRAM**

Una cella di memoria è composta da 2 transistor e 1 LATCH/FLIP-FLOP (4 transistor).

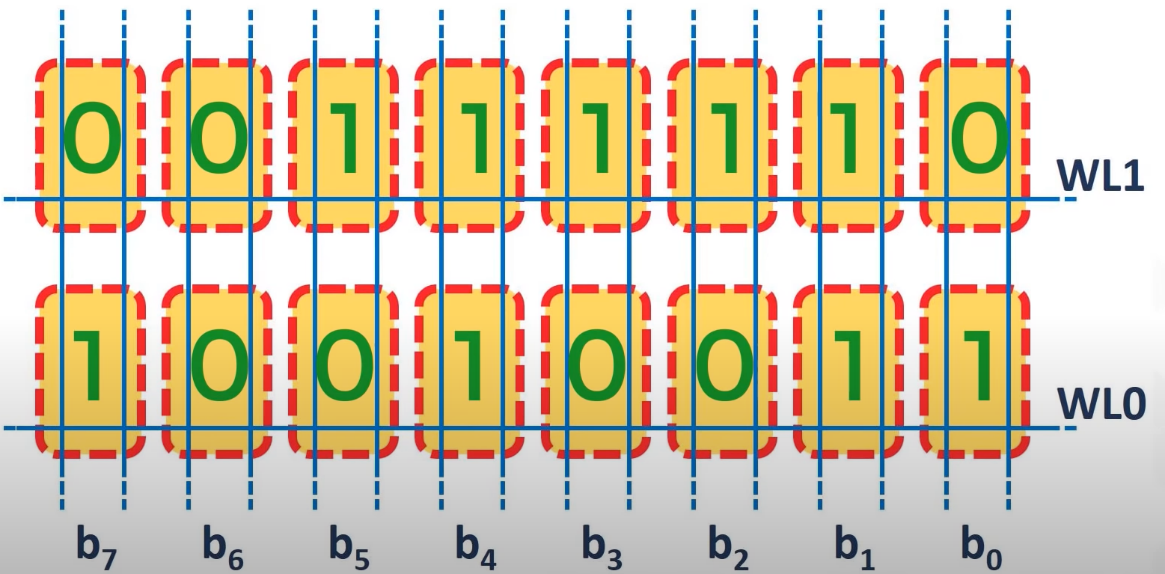
Ogni cella memorizza 1 BIT.

È molto dispendioso dal punto di vista economico.

**WORD LINE:** si attiva quando viene decodificato l’indirizzo di memoria, corrispondente a una locazione.

**BIT LINE**: contiene il contenuto informativo da LEGGERE o SCRIVERE in memoria.

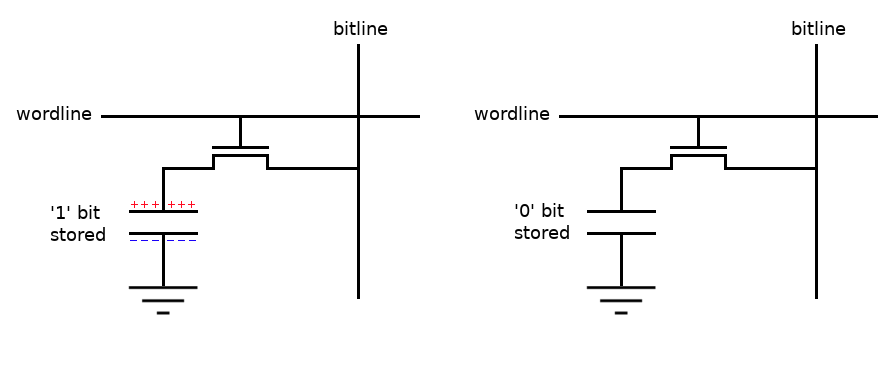
Quando la WL = 1 i transistor si chiudono.

* **Lettura:** la BL riceve il valore contenuto nella cella
* **Scrittura:** la BL invia il valore nella cella

**Le operazioni di lettura e scrittura possono avvenire all’infinito senza causare danni ai circuiti interni. Tempo Lettura = Tempo Scrittura**

* **Synchronous RAM:** architettura basata sulle pipeline, cioè in parallelo.
  + **BUS** grandi con **trasferimenti** fino a **Gb/s**
  + Molto **costose** per essere sviluppate a causa del **grande** **numero** di **componenti** richiesto (6 transistor = 1 bit)
* **Asynchronus RAM:** 
  + **velocità** di trasferimento **modesta**
  + **Risparmio energetico**

**DRAM**



Come nelle SRAM sono presenti la WORD LINE e la BIT LINE, e hanno la stessa funzione.

In questo caso **1 bit** viene **salvato** all’interno di un **condensatore**, che se è **carico = 1**, **scarico = 0**.

Un condensatore però non mantiene la sua carica all’infinito, ma si scarica in un tempo di 5Τ = 5 \* (R\*C).

Esso può scaricarsi sulla BL in caso di LETTURA, oppure sul transistor quando esso è aperto, quindi la WL = 0. Ciò vuol dire che non è stata selezionata la locazione.

Se si scarica si perde il contenuto informatico, perciò è necessaria una operazione di REFRESH circa 16 volte al secondo = 16Hz -> 1 / 16 = 64ms

**VANTAGGI E SVANTAGGI DRAM**

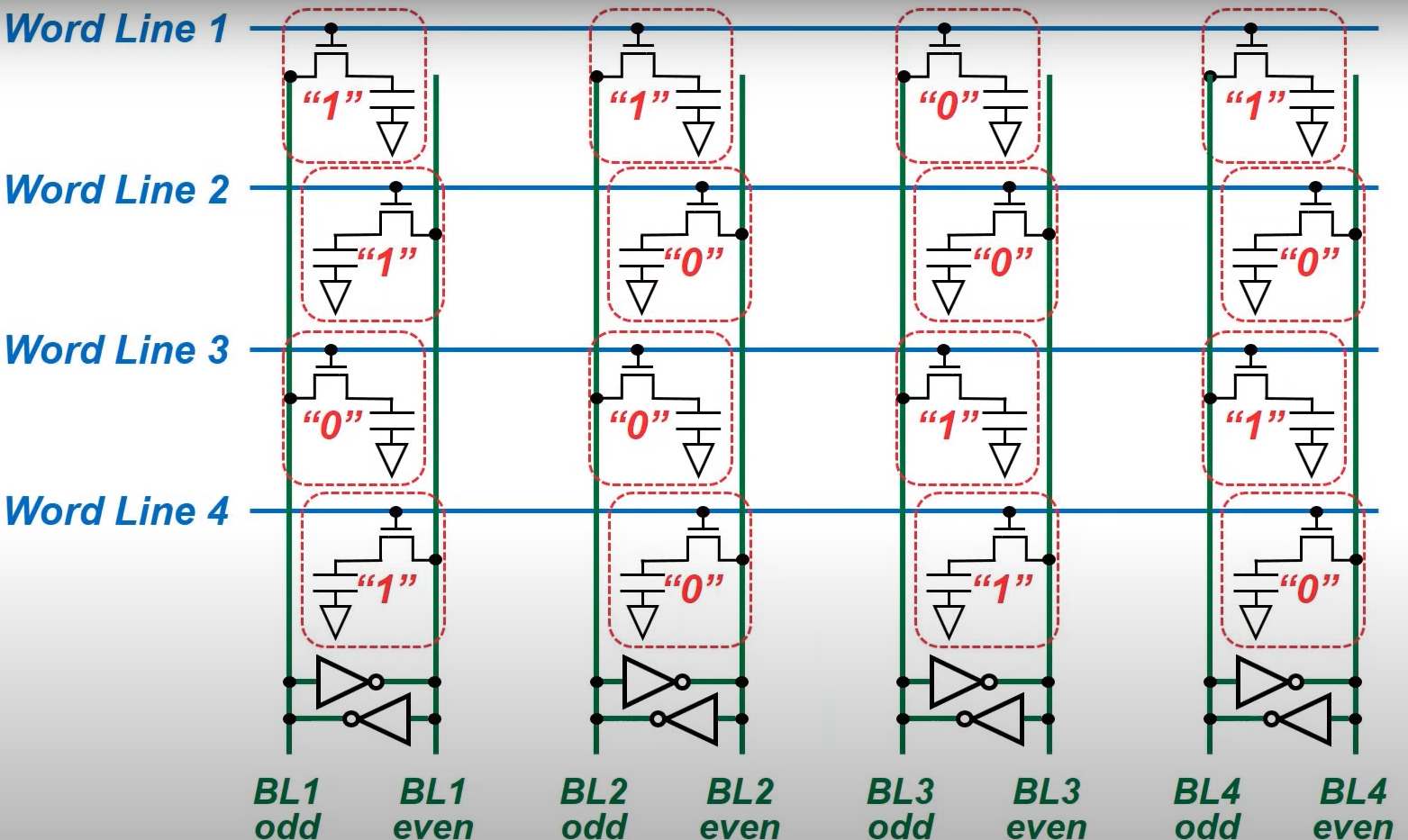
* **Più** **lente** delle **SRAM**
* **Più** **economiche**, richiedono **meno** **componenti**: 1 transistor e 1 Condensatore
* **Complessità** della **conservazione** dello stato: **operazione** di **REFRESH** **onerosa**

**TIPOLOGIE DI DRAM**

* **DDR = Double Data Rate:** i dati vengono trasmessi anche quando CLK è basso

Immagine che contiene testo, orologio

Descrizione generata automaticamente

REFRESH

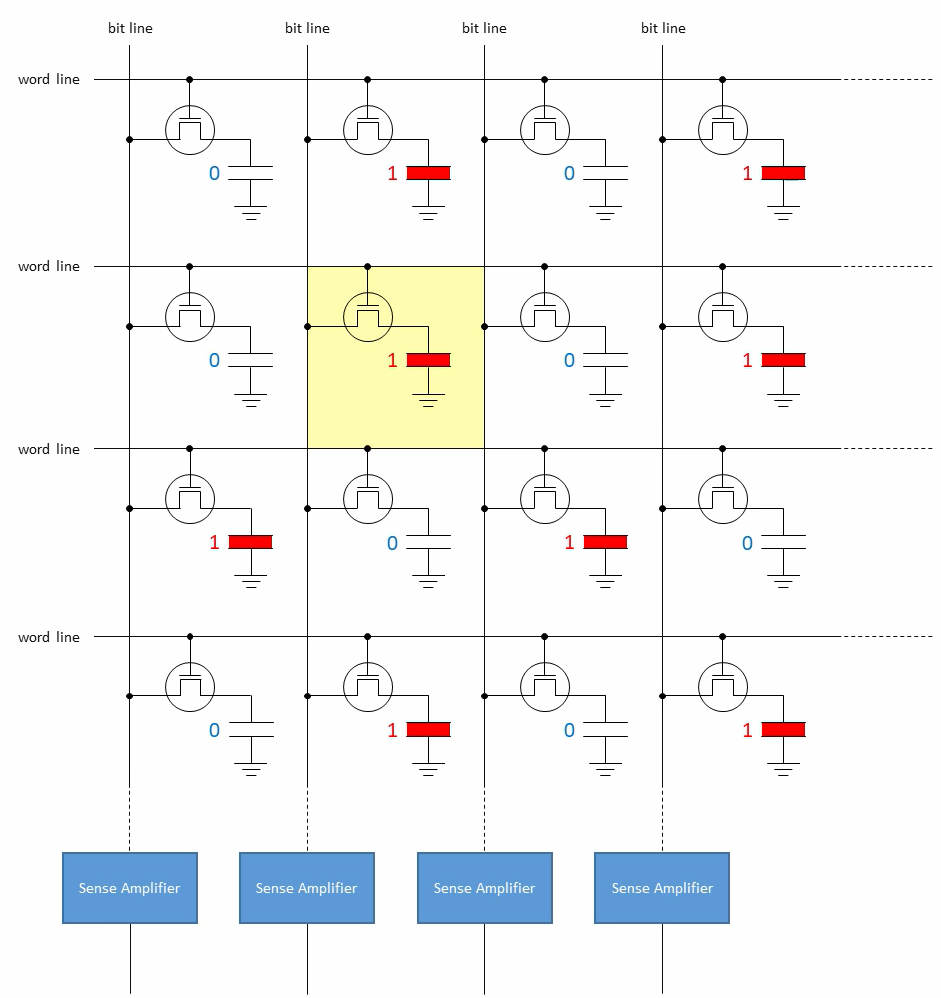
Ci sono **2 WORD LINE**, una per le **locazioni** **DISPARI** e una per quelle **PARI**.

Il circuito sotto è un **Inverter** **LOOP**.

Durante l’operazione di **REFRESH** i due **inverter** sono a una **tensione** **intermedia** **tra** i valori **0 e 1**, circa 0.5V. Se **BLodd = BLeven** si è in una **condizione** di **equilibrio**.

* Se **C = 1**, la sua tensione sarà maggiore di 0.5V, perciò **comincerà** ad **aumentare** quella sulla **BITLINE**. Il **LOOP** **incrementa** sempre di più **la tensione** fino ad arrivare a una condizione in cui una BL vale 1V e l’altra 0V.  
  Quando la **tensione** sulla **BL = 1V** **ricaricherà** il **C** che nel mentre si stava scaricando, **ripristinando il valore logico 1**, presente in precedenza.
* Se **C = 0**, la tensione sulla **BL = 0V**, perciò la tensione viene **subito ripristinata**.

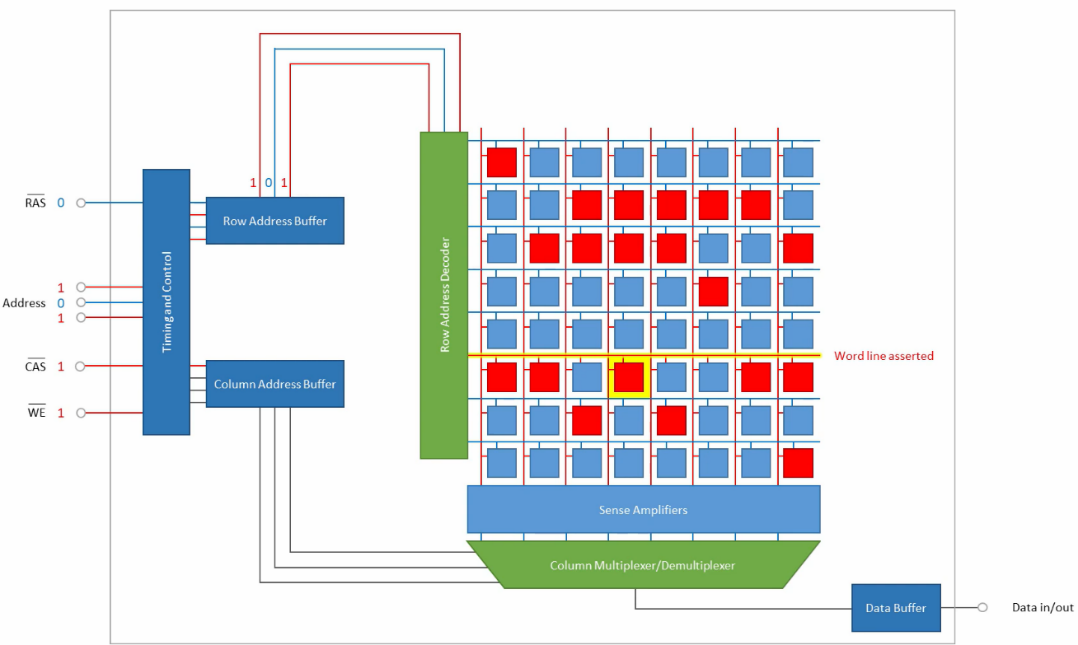
**ACCESSO IN MEMORIA**



I **SENSE AMPLIFIER** sono dei **circuiti logici** **contenenti** dei **LATCH** che **memorizzano** **temporaneamente** il **dato** presente nella cella.

**Dopo ogni** operazione di **LETTURA** il **dato** viene **perso**, perciò è **necessaria** una operazione di **REFRESH** immediata.

**LETTURA E SCRITTURA IN DRAM**



not(RAS) = flag che indica se è il momento di effettuare la fase RAS. Attivo BASSO

not(CAS) = flag che indica se è il momento di effettuare la fase CAS. Attivo BASSO

not(WE) = flag che permette o meno la scrittura in memoria. Attivo BASSO

ADDRESS = numero di bit che servono a codificare un indirizzo di memoria

**N bit -> 2^N allocazioni possibili**

La fase di accesso in memoria si divide in 2 fasi principali:

* **RAS (Raw Address Strobe)** : individua la RIGA (WORDLINE) da attivare
* **CAS (Coloumn Address Strobe)** : individua la cella nella Locazione

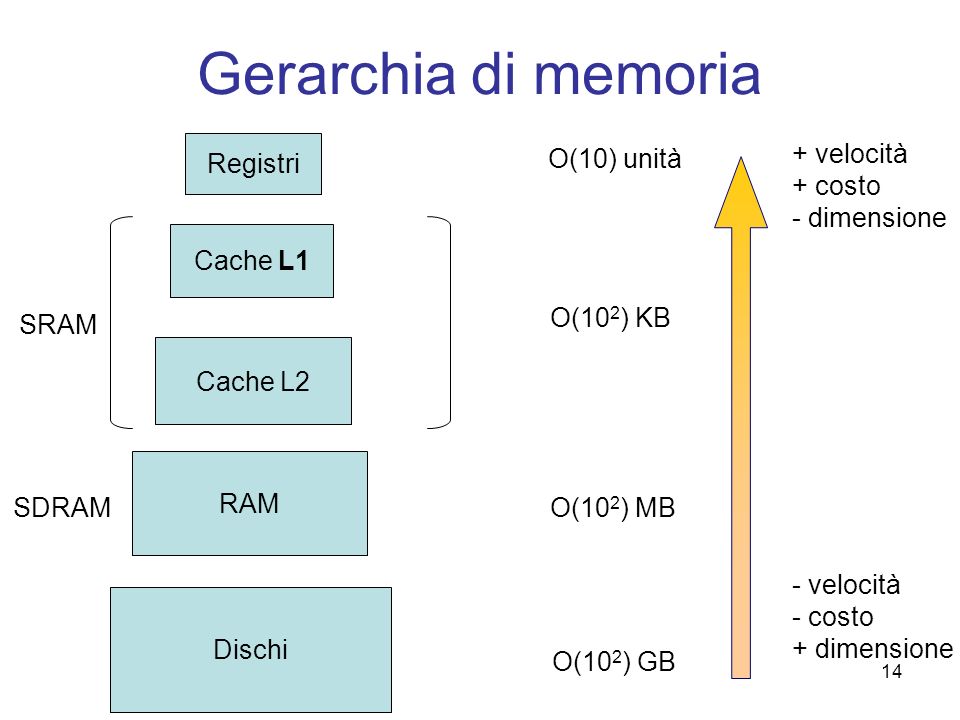
Il modulo di controllo determina le fasi dal punto di vista tempistico e logico, in base ai flag attivi.

* **Indirizzi** entrano nel Modulo. Quando **RAS è attivo** **parte** la **fase di RAS**
* Nel **Raw Address Decoder** vengono **inseriti n bit** per la **codifica dell’indirizzo**.
  + **DECODER** = **n ingressi, 2^n uscite** -> **attiva l’uscita** corrispondente alla **codifica in binario degli ingressi**
  + Viene **attivata la RIGA** (WORDLINE) corrispondente alla codifica
  + **Tutta la LOCAZIONE** viene **salvata** all’interno dei **SENSE AMPLIFIER**
  + La **LOCAZIONE** viene **inserita** nel **Coloumn Address DE/Multiplexer**
* **LETTURA**:
  + **CAM**: viene **mandato** in **uscita** l’**ingresso** **corrispondente** alla **codifica** in binario **degli** ingressi **selettori** (n bit degli indirizzi)
* **SCRITTURA**
  + **CAD**: viene **mandato** in **una** delle 2^n **uscite**, **selezionata** dai **selettori**, il **segnale** di **ingresso**.

**MEMORIE PERMANENTI**

Sono memorie non volatili, che mantengono i dati anche in caso di mancanza di alimentazione.

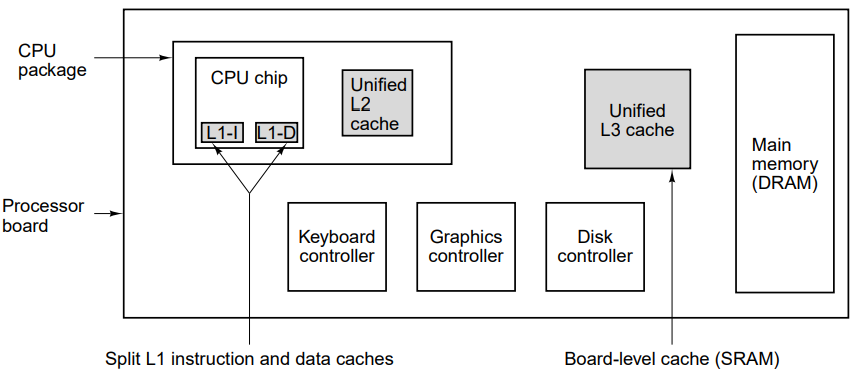
* **ROM** **(Read Only Memory)**: sono scritte nel momento in cui sono fabbricate e non possono essere modificate. Solitamente le ROM contengono le informazioni utili durante il BOOT della macchina.
  + **PROM (Programmable ROM)**: programmabili 1 volta
  + **EPROM (Eresable PROM)**: possono essere cancellate e riscritte
  + **EEPROM (Electrically EPROM)**: cancellabili elettricamente
  + **FLASH**
* **DISCHI RIGIDI**: usati per le memorie di massa ad alta capacità
  + **HDD Hard Disk Drive**: Alta capacità ma bassa velocità
  + **SSD Solid State Drive**: minore capacità ma alta velocità



Nei livelli più alti ci si avvicina alla CPU.

La CPU contatta prima i registri interni per cercare il dato occorrente.

In caso di mancanza del dato comincia a scendere nella piramide.



# LIVELLI INTERMEDI DI MEMORIE

* REGISTRI-RAM: **Memoria Cache**, suddivisa anch’essa in livelli.

Utilizza una tecnologia SRAM, più veloce e resistente della DRAM. La sua capacità dipende dal suo livello e funzione.

* + **Cache L1** = interna alla CPU
  + **Cache L2** = interna alla scheda madre. Cache intermedia tra multicore per limitare il più possibile il traffico nei bus di sistema
  + **Cache L3** = gestione con la RAM

**Convenienza nell’uso della Cache**

**H = probabilità di Cache HIT 🡪 determina l’efficienza di una Cache (ottimale 80-95%)**

**Tc = Tempo di accesso alla cache**

**Tp = tempo di accesso alla RAM**

**Tm = tempo medio di accesso alla RAM in presenza della Cache**

**= Tc + (1-h) \* Tp 🡪 Tm < Tp 🡪 h > Tc/Tp**

Contiene i dati che verranno usati con più probabilità. Vengono usate due regole statistiche:

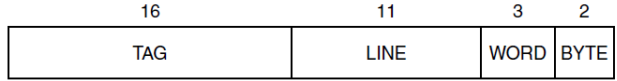
* **Località temporale:** istruzioni e dati che sono usati di recente hanno probabilità di essere richiamati. MRU (Most Recently Used). È possibile che delle istruzioni diverse possano essere interpretate dalla CPU come loop di microistruzioni costanti, perciò la Cache è efficiente.
* **Località spaziale:** istruzioni e dati contigui a quelli appena usati possono essere chiamati con maggiore probabilità. Molte istruzioni vengono salvate in maniera sequenziale per massimizzare l’efficienza della contiguità durante l’esecuzione di una macroistruzione.

**Località Spaziale**

Si ottiene creando delle “**linee di cache**”

Queste linee contengono 32-64Bytes di locazioni contigue della RAM.

In caso di un Cache Miss viene aggiornata l’intera linea con un’altra appena prelevata dalla RAM.



* **FULLY ASSOCIATIVE:** ogni blocco di memoria RAM può essere memorizzato ovunque nella Cache. Viene inoltre salvato l’intero indirizzo di memoria.

La Cache è divisa in 2 campi

* + **Directory:**
    - **Attribute: contiene gli stati della Cache**
      * **M = Modified, D = Dirty =** il dato è modificato solo all’interno della Cache e la modifica non è stata salvata della RAM
      * **O = Owner, SD (Shared Dirty), SM (Shared Modified), T (Tagged). Owner possiede il dato originale,** mentre le altre linee di cache possono contenere lo stesso dato ma con minore grado di importanza.

In caso di **richiesta di lettura** sul **BUS** il **dato** viene **fornito dall’Owner**

* + - * **E (Exclusive), R (Reserved), V (Valid), C (Clean):** il dato è valido aggiornato in RAM
      * **Invalid :** se il dato non è presente in Cache, oppure è presente ma non è valido perché modificato o alterato
    - **INDIRIZZO:**
  + **Data**
* **VALIDITY = 1 bit 0= linea non valida non ancora caricata correttamente**

**1= linea valida**

* **TAG: 16 bit** contengono l’indirizzo della memoria RAM mappato nella linea di cache
* **LINE: 11 bit** individuano la linea di cache
* **WORD: 3 bit** individua quante WORD (4BYTE) sono presenti nella linea di cache 2^3 = 8 WORD = 8\*4 = 32Bytes
* **BYTE: 2bit** può essere usato come OFFSET

Quando la CPU cerca un dato controlla la linea di cache. Se VAL = 0 Cache MISS, altrimenti controlla se l’indirizzo che cerca corrisponde al campo TAG della linea. Se corrisponde Cache HIT, altrimenti MISS

**ESEMPIO**

**RAM 64KB indirizzabile a 1 BYTE 🡪 WORD = 1Byte. Dimensione di blocco 8Bytes. Si deve mappare la RAM in una cache di 256Bytes a indirizzamento diretto**

**64KB = 2^6 \* 2^10 Bytes = 2^16 Bytes Indirizzi di memoria a 16bits**

**I 16 bits devono essere divisi in 3 campi, TAG, LINE e WORD.**

**BLOCCO = 8 Byte = 2^3 Byte il campo WORD della linea sarà di 3 bit**

**N linee = Capacità Cache / Dimensione Blocco = 256Bytes / 8 Bytes = 32 = 2^5**

**Campo LINE = 5bit**

**Campo TAG = 16 – 5 – 3 = 8 bits**

* RAM-MASSA: **Memoria Virtuale**
  + **Paginata:** crea una espansione virtuale alla RAM
  + **Segmentata:** organizza la memoria virtuale in segmenti