## Stack

Segmento di memoria centrale chiamato stack segment dove vengono salvate variabili locali e parametri di un sottoprogramma.  
  
Il programma non può scrivervi direttamente ma solo in modalità LIFO (impilando, anche noto come Last In First Out)

L’indirizzo dello stack è salvato nel registro SS(**Stack Segment**)

Il registro SP(**Stack Pointer**) gestisce gli offset dello stack rispetto all’inizio del segmento

Il segmento di stack viene **gestito** in maniera **differente** rispetto ad altri segmenti:

* **Dimensione tipica di 1024 bytes** ma lo stesso modificabile
* Lo **Stack Pointer punta all’ultima cella** (1024 esima) contenente il valore EOS (End of Stack)

Possiamo effettuare **2 azioni** nello stack:

1. **Push (scrittura)**: **Decrementa SP** di una **dimensione pari al tipo di dato** e inserisce il **dato in cima allo stack**. **Può causare stack overflow** (stack pieno) se SP arriva a 0.
2. **Pop (lettura)**: **Incrementa SP** di una **dimensione pari al tipo di dato**, **rende disponibile lo spazio** occupato dal dato **puntato da SP** prima dell'incremento, **senza cancellare il dato stesso**. **Può causare stack underflow** (stack vuoto) se SP arriva al valore massimo.

**Scrittura:** la CPU decrementa il puntatore dello stack pointer e trasferisce il dato puntato dallo stack pointer.

**Lettura:** La cpu trasferisce il dato dall’indirizzo di memoria a cui punta SP, e successivamente incrementa quest'ultimo.

Lo stack viene utilizzato nei linguaggi ad alto livello per allocare variabili locali di un metodo/funzione oppure per salvare lo stato di un processo al verificarsi di un interrupt.

Ogni tipologia di variabile ha il suo peso. Es: int 4 byte, float 4 byte, double 8 byte, long 8 byte, short 2 byte, char 1 byte, bool 1 byte, void 0 byte, long double 16 byte…

All’interno della memoria i dati non vengono cancellati e verranno poi sovrascritti. (Es. dopo una POP la PUSH sovrascriverà).  
  
Lo stack per convenzione viene decrementato dal basso verso l'alto quindi 1000 in fondo e 0000 in cima (img sotto, FFFF è il max). Architetture di questo tipo sono per esempio la x86



## Memorie a semiconduttore

Si dividono in volatili e non volatili.

### Memoria RAM

Mantiene i dati e istruzioni dei programmi in esecuzione.  
La RAM è volatile, riscrivibile, veloce, meno capace, costosa, ad accesso diretto

Si divide in due gruppi principali:

* Dynamic RAM (DRAM)
* Static RAM (SRAM)

La densità di memoria rappresenta la quantità di informazioni che possono essere immagazzinate in una determinata area di memoria.

+Densità –> +Grande la quantità di dati che posso inserire in un singolo chip di silicio.

La DRAM ha una densità di memoria molto maggiore rispetto alla SRAM

La DRAM ha bisogno di grandi quantità di energia per i costanti refresh.

Le prestazioni della SRAM la rendono migliore per applicazioni che richiedono più velocità in accesso grazie alla sua natura statica e la mancanza di refresh costante (grazie ai transistor), consuma anche meno energia. Spesso viene usata come memoria di cache per compensare le sue limitazioni, viene sfruttata la sua velocità per fornire un accesso rapido ai dati più utilizzati.

Il valore di bit viene salvato tramite flip-flop, mantenendo il valore fino a quando è presente l’alimentazione e ha una densità bassa.

La SRAM ha 6 transistor.

La densità influenza il costo e le prestazioni. Le prestazioni della memoria sono influenzate da latenza e larghezza di banda

Latenza (tempo di accesso alle informazioni): specifica il numero di cicli di clock necessari per eseguire una operazione (-latenza + velocemente si eseguono le operazioni)

La maggiore densità per quanto comporti dimensioni maggiori porta a una congestione dei circuiti aumentando la latenza

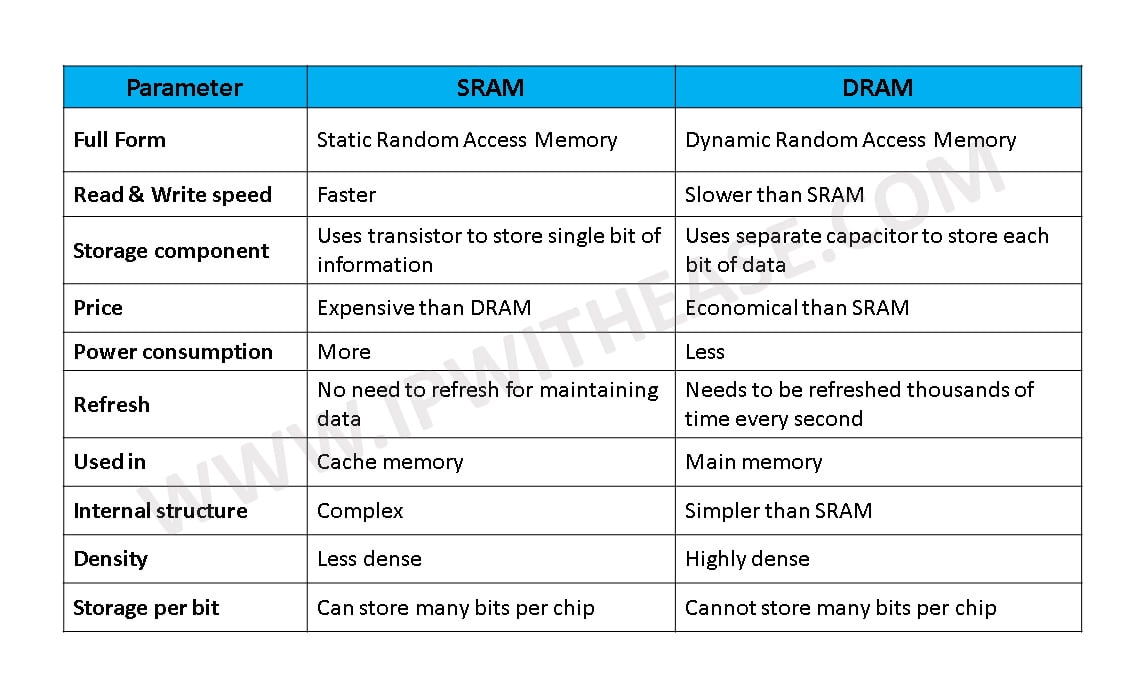
In conclusione l’uso tra queste due ram dipende in base al tipo di esigenze dell’applicazione.

La DRAM consente di memorizzare ogni bit di dati all’interno di un circuito integrato, viene usata come memoria standard per i computer desktop moderni (forma completa memoria ad accesso casuale dinamico). La presenza di bit viene indicata tramite la presenza di carica in un condensatore, richiede un refresh periodico per mantenere il valore (un condensatore carico tende a disperdere carica) e ha un’alta densità, consuma anche più energia. Ha anche più latenza rispetto alla SRAM perché più densità crea più latenza.

1 condensatore e 1 transistor

Refresh: il dato viene letto e poi riscritto.

La DRAM compensa la latenza con una larghezza di banda (frequenza) molto elevata.



Capacità: SRAM 1mb -> 16mb

DRAM 1gb -> 16gb

La SRAM è memoria su chip mentre DRAM è off-chip

NON HO SEGNATO LE LATENZE DI OGNUNA!!!

Le DRAM possono essere classificate in 3 tipi:

* SDRAM
* DDR SDRAM
* RDRAM

La SDRAM esegue il refresh in modo sincrono alla CPU (prima la CPU legge, poi refresh).

La sua velocità è anch'essa sincronizzata con la velocità del clock nel pc.

Nata per adeguarsi alla velocità degli altri componenti di un computer.

DDR(Double Data Rate): presenta varianti dall’ 1 al 5.

Anche detta DDR SDRAM ovvero con un solo passaggio si fa una doppia lettura.

**METTO TABELLA DIFFERENZE SDRAM E DDR!!**

RDRAM(Rambus DRAM) era un alternativa alla SDRAM a causa del loro costo e del copyright costavano troppo e quindi cadute in disuso.

DDR ha voltaggio più basso, velocità clock e dati maggiore, più pin e più recente.

Le DDR prima gen, possiede una larghezza di banda maggiore rispetto alla SDRAM. Utilizza entrambe le fasi del ciclo del Clock in modo da fare un doppio giro per ogni ciclo di clock.

Questo tipo di architettura semplifica la loro costruzione e aumenta il loro risparmio energetico, hanno bus da 64 bit e una memoria media di 512Mb. Avevano un buffer di prefetch di 2 bit

Il prefetch è la capacità di trasferire bit di dati dell’array di memoria al buffer interno di input/output per ogni ciclo di clock in anticipo

DDR2: due volte più veloce della prima gen. con buffer di prefetch di 4 bit. La velocità di clock tra 133 - 200 MHz, come le prime DRR.

Velocità di trasferimento: 533 - 800 MT/s.  
Transfer rate pari a 4 volte la frequenza del clock, connettore a 240 pin. Velocità pari alla DDR ma considerando che aveva un transfer rate migliore la rendeva più veloce.

DDR3: consumo energetico ridotto del 40%, velocità di trasferimento: 800 - 1600 MT/s. Buffer di prefetch di 8 bit.

Hanno aggiunto 2 funzioni: ASR (Automatic Self Refresh) e STR, fanno sì che la memoria controlli la frequenza di clock si aggiorna a seconda della temperatura.

DDR4: Tensione operativa più bassa (1,2 V) e maggiore velocità di trasferimento 2133 - 3200 MT/s.

Aggiunge 4 nuove tecnologie Bank Group: Ogni banco di memoria ha la funzione single head. Le funzioni sono: **DBI**(Data bus inversion), **CRC**(Cyclic Redundancy Check) e **CA** Parity. Migliorano la stabilità della trasmissione e accesso dei dati.

Possono elaborare 4 dati diversi in un ciclo di clock, quindi l'efficienza è molto maggiore delle DDR3.

DDR5: prelettura 16 bit, tensione di 1,1V

RDRAM (Rambus DRAM): Il suo costo è molto più elevato delle SDRAM a causa costi di produzione e costi di licenza, vecchia alternativa alle SDRAM. Venivano usate nei server di fascia alta ma ora sono state superate.

Riguardo ai pin ci sono due tipi:

* SIMM (Single In-Line Memory Module)
* DIMM (Dual In-Line Memory Module)

Le DIMM sono migliori perchè mi permettono di raddoppiare i dati che possono essere gestiti senza aumentare la frequenza di clock, tramite pin da entrambi lati della RAM.

DIMM: doppio bus a 64bit –> fino a 128gb

SIMM: un bus a 32 bit

Le SIMM sono obsolete e poco usate.

Il numero di slot DIMM è collegato alla capacità della RAM (+slot –> +ram installabile)

Nella storia i processori sono sempre stati più veloci mentre le memorie non hanno avuto lo stesso incremento.

METTO GRAFICO DELLE SLIDES!!

Gerarchie di memoria (**MANCANO DEI PEZZI**)

Le caratteristiche delle varie memorie sono dimensione, tempo di accesso e costo per bit.

la velocità di trasferimento è molto inferiore all’elaborazione della cpu, le computazioni sono rallentate dall'accesso di memoria e non dalla CPU ed è improbabile che il gap venga colmato tecnologicamente.

Non essendo possibile creare una…

CPU molto veloce, cache fra cpu e ram (molto veloce rispetto alla RAM ed ha capacità limitata)

Main memory: RAM, alta capacità poca velocità

Hard Drive: modulo I/O

### Memorie Cache

Obiettivo –> fornire i processori di una ridotta quantità di memoria interna molto veloce dove riversare i dati frequentemente utilizzati.

La cache memorizza i dati più usati della RAM in modo da incrementare le prestazioni del computer.

Contiene una copia delle informazioni in memoria, se non aggiornata costantemente si possono creare disallineamenti tra cosa c'è in cache e cosa in memoria centrale creando problemi di elaborazione.

E’ posto tra la CPU e la RAM, molto più veloce ma con meno capacità.

Permette di velocizzare le operazioni di LOAD/STORE per i dati più usati.

La CPU guarda prima nella cache se è presente il dato che serve (3-50 ns), se non è presente verrà ricercato nella RAM (100-250 ns) e lo trasferisce nella locazione più vecchia della cache.

Ci sono diversi livelli di cache:

* L1: interna al chip
* L2-L3: esterne al chip

Ogni core della CPU ha la propria cache L1 mentre la L2 e L3 sono condivise.

Per aumentare le prestazioni si può frammentare la cache e trasferirne una parte all’interno della CPU.

L1: dimensioni piccolissime (8 kb-64 kb) opera alla frequenza di clock.

L2: (128 kb-1 mb), a seconda delle architetture è posta all’interno o all’esterno, opera a frequenze differenti.

L3: generalmente è una estensione della L2 (circa 2MB).

(informazioni datate, adesso le cpu hanno all’incirca 30 MB o più di cache totale)

L’organizzazione dei dati è diversa rispetto alla RAM, insieme ai dati viene registrata la posizione nella RAM (tag) in modo da creare una corrispondenza diretta tra le due memorie.

Quando si preleva (lettura) un dato si effettua una ricerca nella cache anche se si conosce già l’indirizzo nella RAM e:

* SE trova il dato (cache hit), trasferimento rapido.
* SE non trova il dato (cache miss) si avvia il caricamento dei dati dalla RAM.

Il rapporto tra cache hit e accessi totali alla memoria, detto hit rate,

misura l'efficacia della cache.

La cache può essere usata sia per lettura che scrittura

È organizzata a blocchi che prendono il nome di linee.

Ad ogni accesso in memoria centrale, le informazioni vengono riportate anche in una linea ??

La cache memorizza coppie (Address, Value), dove Address è l’indirizzo della RAM e Value è la relativa word memorizzata.

In scrittura la cache può operare in 2 modi: **write through**, ogni scrittura della cache comporta una scrittura sulla ram.

**write back**, prima scrivo il dato in cache e solo quando deve essere tolto dalla cache allora verrà scritto in ram.

CACHE MISS: Un'operazione LDR (load), provoca la ricerca dell’indirizzo nella cache. Se l’indirizzo non viene trovato si verifica un evento di cache miss. Il cache controller recupera il dato dalla memoria e lo inserisce in cache.

CACHE HIT: Ricerca nella cache, se lo trova si verifica HIT e il trasferimento è velocissimo. Il rapporto HIT/tentativi totali da l’hit rate, che misura l’efficacia della cache.

CACHE WRITE: Un'operazione STR (store), provoca l’aggiornamento del valore associato all’indirizzo nella cache.

Il valore in memoria sarà differente ma poiché le LDR fanno riferimento alla cache otterranno sempre il dato aggiornato.

La locazione di memoria aggiornata viene marcata, nella cache, come **dirty** per segnalare che essa contiene un valore differente da quello in RAM.

CACHE THROUGH:Ogni scrittura sulla cache comporta una scrittura sulla RAM

CACHE BACK:Il dato viene effettivamente scritto sulla RAM solo quando deve essere eliminato dalla cache

CACHE FULL E WRITE BACK: Se non c'è più spazio nella cache occorre scegliere una entry da liberare e, se marcata dirty (disallineamento), occorre aggiornare il valore nella RAM.

La cache tiene traccia del dato da aggiornare impostando il dirty bit.

Per questo una ricerca fallita in write back spesso genera due accessi alla memoria.

PRINCIPI DI LOCALITA’

Riferimenti temporali: data una locazione X acceduta al tempo T, la probabilità che avvenga di nuovo a breve è molto elevata.

Riferimenti spaziali: data una locazione X acceduta al tempo T, la probabilità che vengano accedute locazioni adiacenti a X a breve è molto elevata.

Quando si accede ad un dato in memoria principale non viene portato in memoria solo il dato interessato ma un intero blocco di dati adiacenti perché molti programmi sono sequenziali per cui l’istruzione successiva sarà all'indirizzo di memoria successivo.

Quando l’assunto viene verificato:

* Il processore utilizza molto gli stessi dati → Località di riferimento temporale.
* Il processore richiede dati vicini in memoria in un breve intervallo di tempo → Località di riferimento spaziale.

Esempio: somma degli elementi di un vettore.

int sum = 0;

for(int i = 0; i < n; i++) {

sum = sum + V[i];

}

sum e i: Località temporale poiché vengono usate ad ogni iterazione.

V: Località spaziale poiché viene letta una posizione dopo l’altra (in successione), quindi caricata in blocchi.

ORGANIZZAZIONE DELLA CACHE nella località spaziale

La cache nel principio della locazione spaziale non è organizzata in locazioni ma in blocchi o pagine di dimensioni costanti e potenze di 2 (in genere 64 byte).

La cache include una funzione di **mapping** che lega il numero di un blocco della cache al numero del blocco della RAM e viceversa.

La cache tende a massimizzare la cache hit e minimizzare il cache miss tramite le località di riferimento.

CI SONO DIVERSI TIPI DI CACHE :

* a mappatura diretta
* completamente associativa
* associativa a k-vie

Per analogia un parcheggio è la nostra memoria quindi ogni cella contiene una macchina diversa (dato).

La cache è un parcheggio vuoto più piccolo ma più vicino a noi quindi più veloce.

CACHE A MAPPATURA DIRETTA:

Usiamo una regola fissa che decide dove mettere i dati nel parcheggio. Diamo a ogni riga della cache un indice e facciamo lo stesso nella RAM.

Quando leggiamo un dato dalla RAM possiamo scriverlo nella cache solo nell’indice uguale ad entrambi.

Creo una associazione fisica tra le due celle.

CACHE COMPLETAMENTE ASSOCIATIVA:

Non c'è nessuna regola che vincola dove memorizzare i dati nella cache. Quando la cache è libera possiamo scrivere un dato ovunque noi vogliamo, però quando è piena dobbiamo decidere quale posizione sovrascrivere:

* Possiamo cancellare la posizione più vecchia tramite FIFO;
* Possiamo sovrascrivere la posizione usata meno di recente LRU;
* Oppure sovrascrivere una cella a caso;

CACHE A K-VIE:

È un ibrido tra le due precedenti, la base è a mappatura diretta.

La novità è che la riga della cache è diventata un parcheggio multipiano. Ogni cella contiene una macchina per piano.

Quindi dovremo sempre usare lo stesso indice tra cache e RAM ma possiamo scegliere in quale piano parcheggiare.

La lunghezza delle word può variare a seconda delle macchine:

8 bit, 16 bit, 32 bit, 64 bit. Cambierà solo la grandezza del parcheggio.

Quando copiamo dati dalla RAM alla cache è conveniente trasferire anche i dati vincini tramite blocchi di word.

Questi blocchi verranno scritti in successione nella stessa riga della cache che si chiama LINE.

Non possiamo rimpiazzare la singola macchina ma solo l’intera linea.

es. → cache a 2 vie, ogni settore ha 2 piani e linee da 4 word