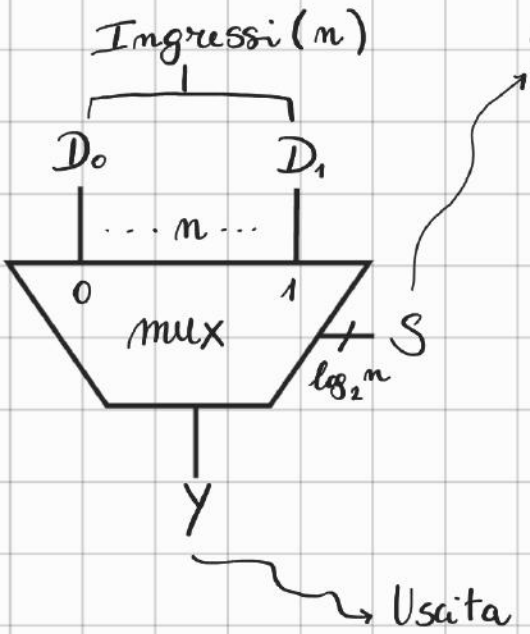
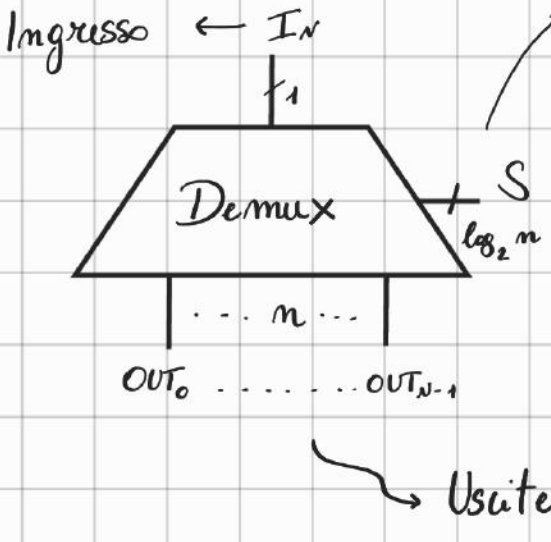
Multiplexer (mux)



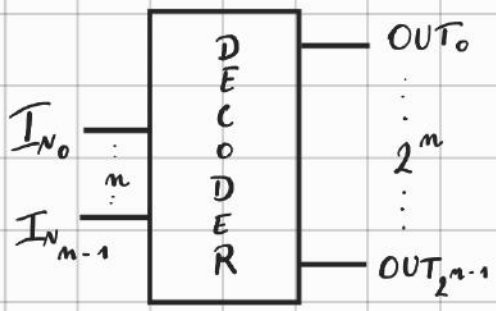
Il multiplexer è una rete combinatoria in grado di scegliere un'uscita a partire da un certo numero di ingressi possibili basandosi sul valore di un selettore.
Se S=0, Y sceglie D_0, D_1 altrimenti.

Demultiplexer



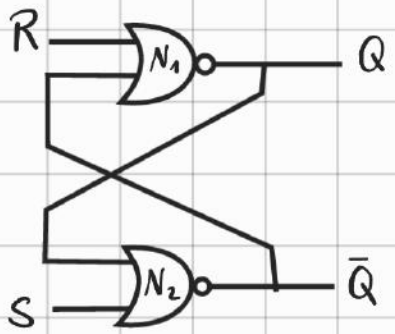
Il Demultiplexer è una rete combinatoria che prende un ingresso in input ed è in grado di scegliere una fra le possibili uscite grazie ad un selettore.

Decoder



Il decoder ha n ingressi e 2^n uscite e attiva una delle sue uscite a seconda delle combinazioni dei valori in ingresso. Le uscite sono dette "One Hot".

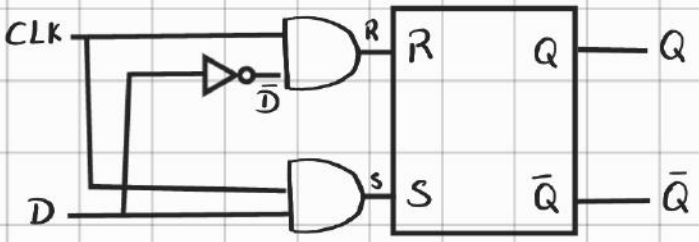
Latch - SR



Il Latch-SR ha due ingressi, S e R e due uscite Q e $\bar{Q}$. S e R attivano (set) e disattivano (reset) l'uscita Q.

1) R=1, S=0 → Q=F, $\bar{Q}$=T
2) R=0, S=1 → Q=T, $\bar{Q}$=F
3) R=1, S=1 → Q=F, $\bar{Q}$=F
4) R=0, S=0 → Se Q=0 → $\bar{Q}$=T, Q=F
   → Se Q=1 → $\bar{Q}$=F, Q=T

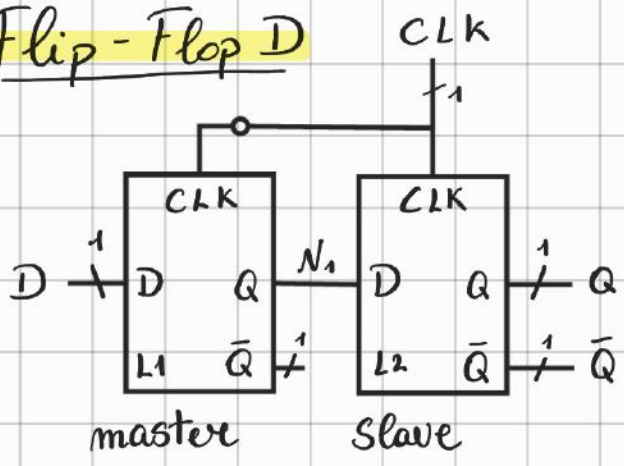
Latch - D



Il Latch-D ha due ingressi: un ingresso dati D che controlla il prossimo stato e un ingresso clock CLK che controlla il momento del cambio di stato.

Se CLK=1 il Latch è trasparente ed i dati scorrono da D verso Q come se il Latch fosse un buffer. Se CLK=0 il Latch è opaco, il passaggio dei dati viene bloccato e Q mantiene il valore precedente.

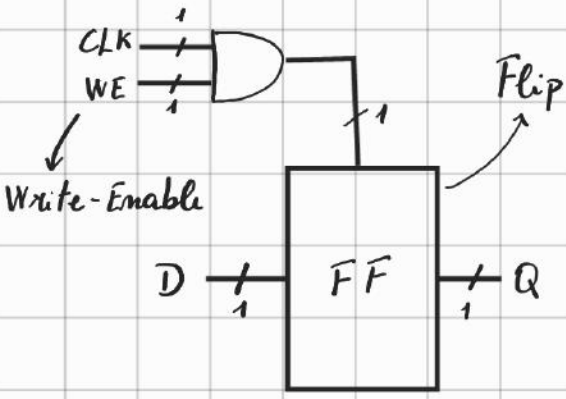
Flip - Flop D



Un Flip-Flop D è costruito a partire da due Latch-D controllati da due segnali di clock complementari. Il primo Latch L1 è detto master e L2 è detto slave, N_1 è il nodo che li unisce.

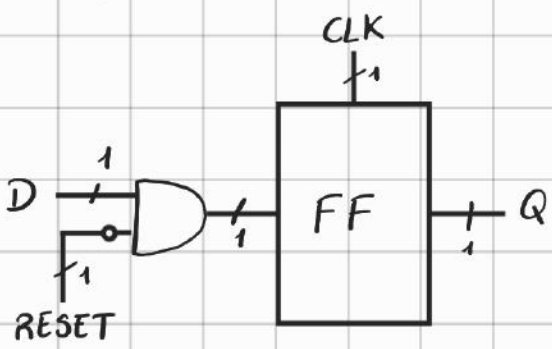
Se CLK = 0 L1 é trasparente ed L2 é opaco quindi qualsiasi valore di D viene portato ad N_1. Se CLK = 1 L1 é opaco e L2 é trasparente quindi il valore di N_1 viene trasmesso a Q ma N_1 rimane isolato da Q. In altre parole viene copiato D su Q al fronte di salita del clock e viene ricordato il suo stato in tutti gli altri casi.

Flip-Flop ENABLE



Un Flip-Flop EN aggiunge un altro ingresso WE per determinare se memorizzare o no il dato sul fronte del clock. Se WE = T si comporta come un normale Flip-Flop D, se WE = F ignora il clock e mantiene il proprio stato.

Flip-Flop RESET

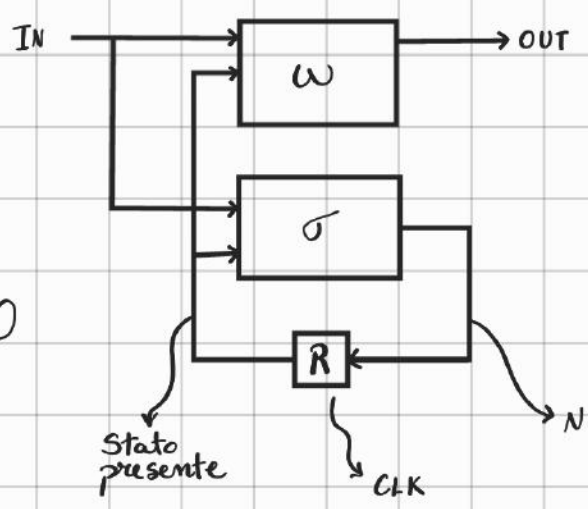


Un Flip-Flop RESET aggiunge un ingresso RESET. Se RESET = F si comporta come un normale Flip-Flop D, se RESET = T viene ignora D e viene settata l'uscita Q a 0.

Mealy

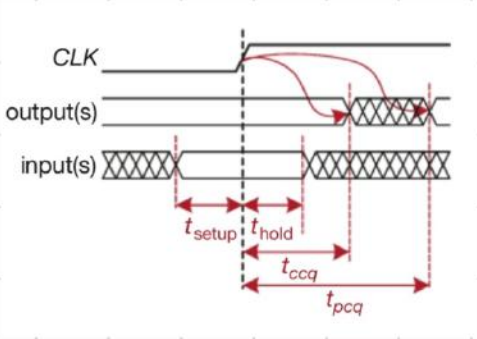
Γ = inputs
O = outputs
Σ = stati

$\omega : \Gamma \times \Sigma \rightarrow O$
$\sigma : \Gamma \times \Sigma \rightarrow \Sigma$



Nelle macchine alla Mealy le uscite dipendono sia dallo stato presente che dagli input attuali.

Temporizzazioni



Quando il clock presenta il fronte di salita, l'usaita o le uscite iniziano a cambiaree dopo il ritardo di contaminazione da clock a Q (t_{ccq}) e devono stabilizzarsi sul valoree definitivo entro il ritardo di propagazione da clock a Q (t_{pcq}). Perché la rete interpreti correttamente l'ingresso o gli ingressi, questi devono stabilizzaresi almeno entro il tempo di setup (t_{setup}) prima del fronte di salita del clock e devono rimanere stabili per la durata almeno del tempo di hold (t_{hold}) dopo il fronte di salita del clock.

Periodo di clock

Si indica con T_c ed é il tempo fra i fronti di salita del segnale periodico del clock. Il suo reciproco, $f = \frac{1}{T_c}$ é la frequenza del clock.

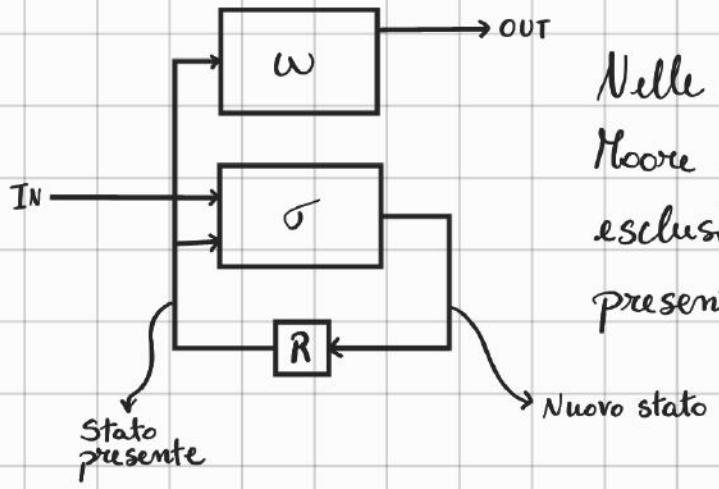
Sfasamento del clock

Nella realtá, il tempo con la quale il segnale di clock raggiunge tutti i registri mostra una certá variabilitá. Questo fa sí che i fronti di clock non si presentino tutti nello stesso istante: tale fenomeno prende il nome di sfasamento del clock.

Moore

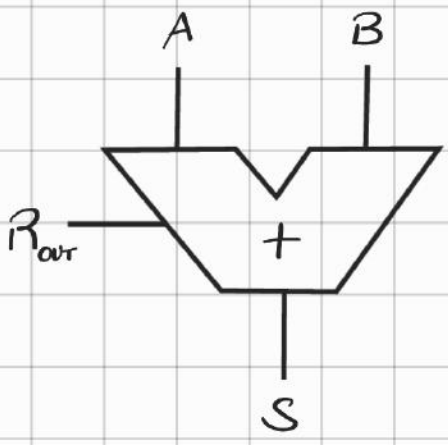
Γ = inputs
O = outputs
Σ = stati

$\omega : \Sigma \rightarrow O$
$\sigma : \Gamma \times \Sigma \rightarrow \Sigma$



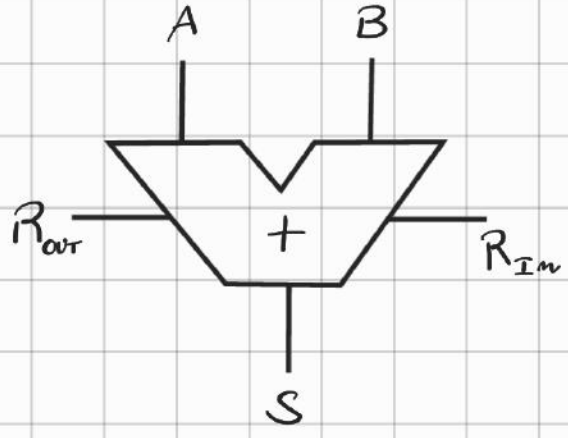
Nelle macchine alla Moore le uscite dipendono esclusivamente dallo stato presente della macchina.

Half - adder



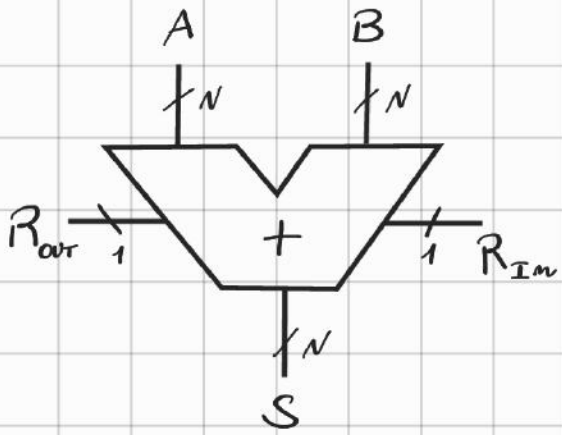
L'half-adder ha due ingressi A e B e due uscite S e R_{OUT}. S é la somma tra A e B e R_{OUT} é il riporto in uscita.

Full - adder



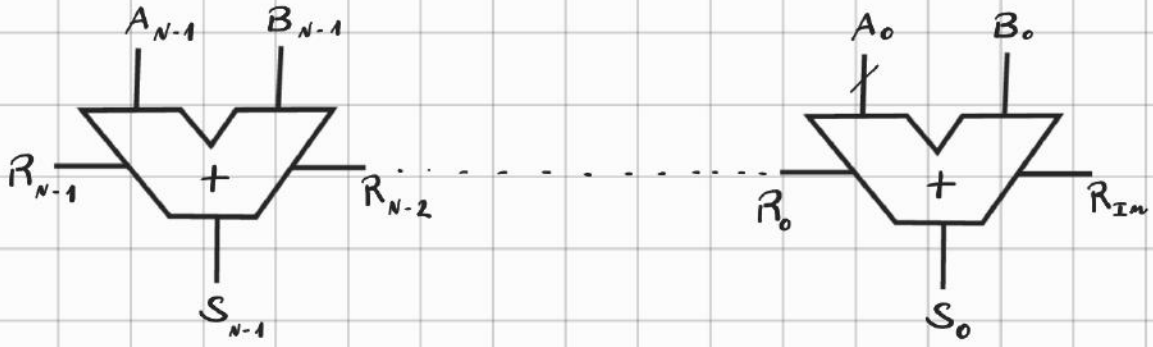
Il Full-adder ha tre ingressi A, B e R_{IN} e due uscite S e R_{OUT}. S é la somma tra A e B, R_{OUT} é il riporto in uscita e R_{IN} é il riporto in ingresso.

Carry Propagate adder



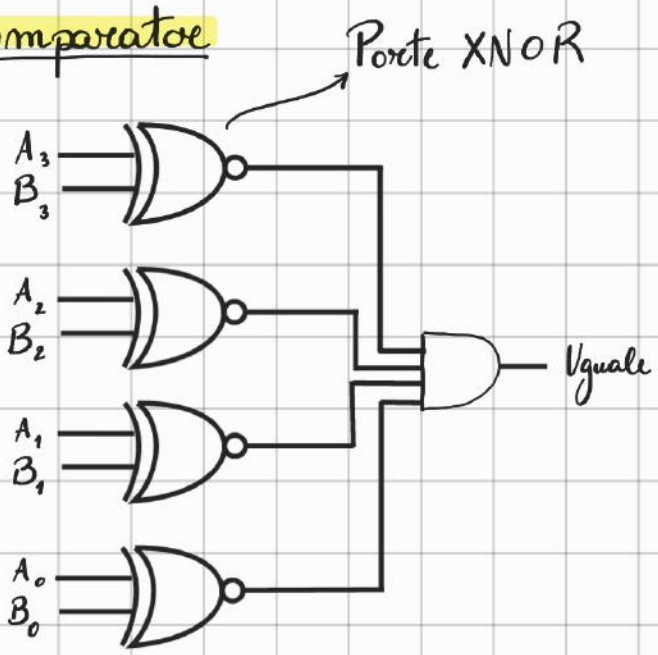
Un carry propagate adder ha due ingressi a N bit A e B e R_{Im} a 1 bit. Produce un risultato a N bit S e R_{out} a 1 bit. Il riporto di un bit si propaga nel bit successivo.

Ripple carry adder



Per costruire un ripple carry adder a N bit bisogna collegare N full adder in cascata. R_{out} di uno stato è R_{im} di un altro.

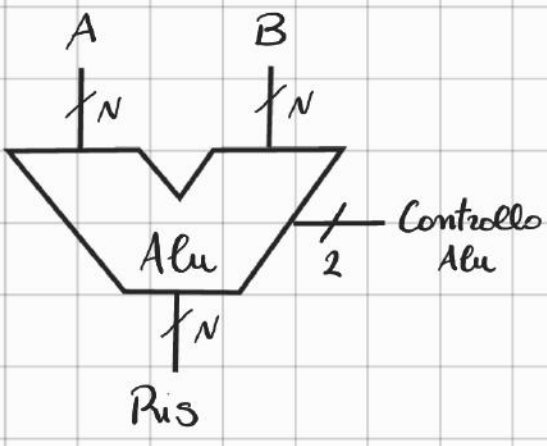
Comparatore



Un comparator riceve due numeri binari a N bit A e B e determina se sono uguali. Esistono due modi:

- Usare due porte XNOR e confrontare colonna per colonna
- Fare A - B e guardare il segno $\begin{cases} \ominus \rightarrow A < B \\ \oplus \rightarrow A \geq B \end{cases}$

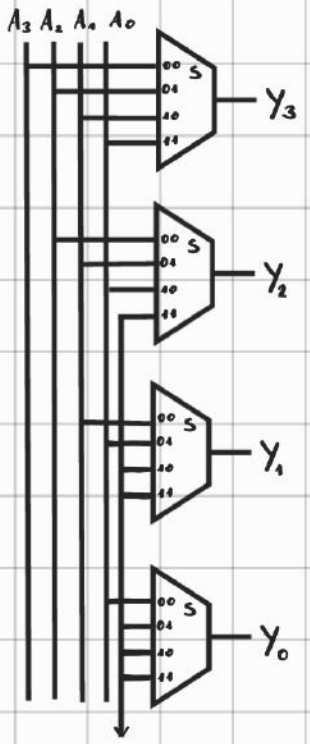
Alu



Un'Alu unisce all'interno di una singola unitá una serie di operazioni logiche e matematiche. Prende in input A e B di N bit e Controllo Alu di 2 bit ed in base al valore di controllo Alu sceglie l'operazione da svolgere tra A e B.

Controllo Alu	Operazione
0 0	Addizione
0 1	Sottrazione
1 0	AND
1 1	OR

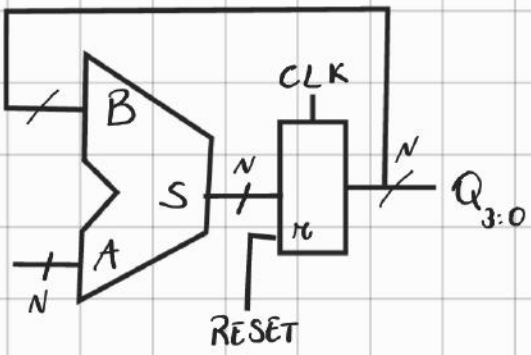
Shifter



Gli shifter traslano un numero binario a destra o a sinistra di uno specifico numero di posizioni. Ne esistono 3 tipi:

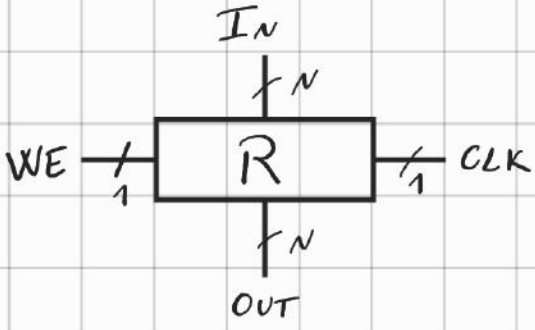
- Shifter logico: trasla un numero verso sinistra o destra e riempie gli spazi vuoti con degli 0.
- Shifter aritmetico: Uguale allo shifter logico se trasliamo verso sinistra ma verso destra riempie i bit piú significativi con una copia del precedente bit piú significativo.
- Rotator: trasla un numero verso sinistra o destra circolarmente in modo che gli spazi vuoti vengano riempiti dai bit all'estremitá opposta del numero.

Contatore



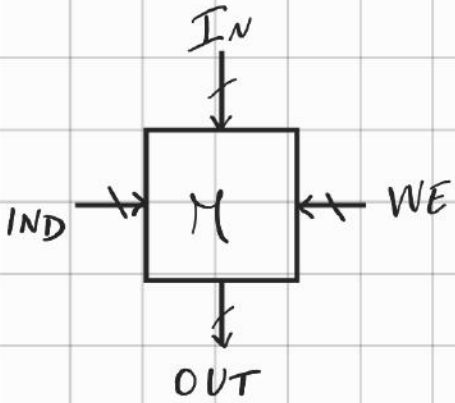
Un contatore a N bit prende in input un CLK e un RESET e produce un output Q a N bit. Il RESET inizializza l'output a 0, successivamente il contatore genera i 2^N possibili valori di uscita in ordine crescente aumentando di 1 ad ogni fronte di salita del clock. Ad ogni ciclo del clock il contatore aggiunge 1 al valore immagazzinato nel registro.

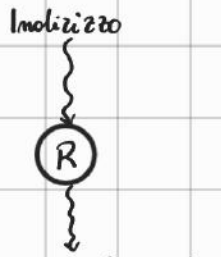
Registro

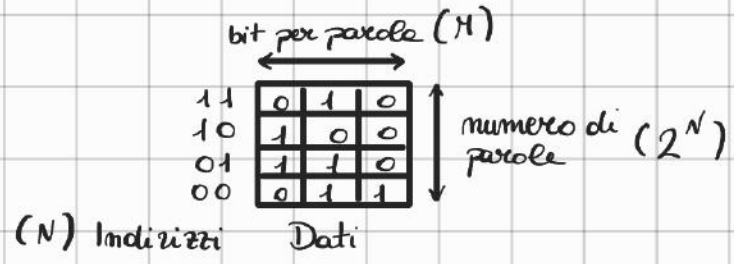


Un registro prende in input un clock e un WE, se entrambi valgono 1 il registro produce in uscita lo stesso valore ottenuto in input sennó rimane stabile.

Memoria (Generica)





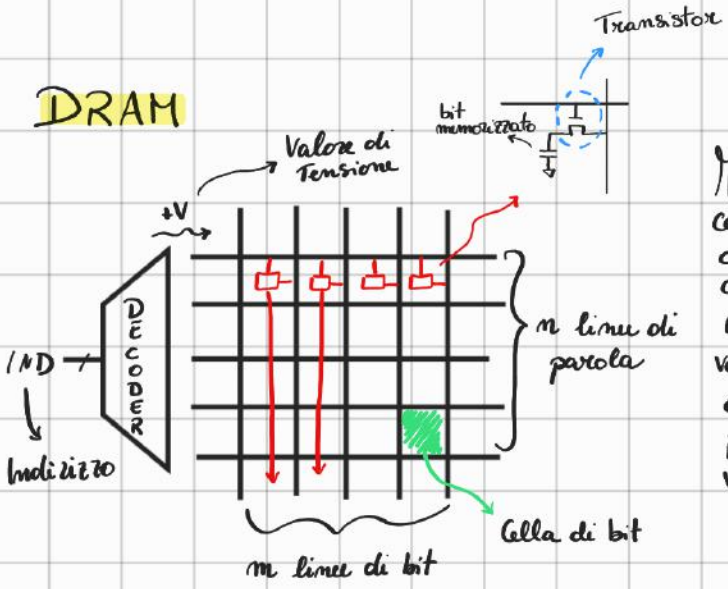


$2^N \times M$
(Dimensione)

$OUT = M[IND_1] \longrightarrow if(WE)\ M[IND_2] = IN;$

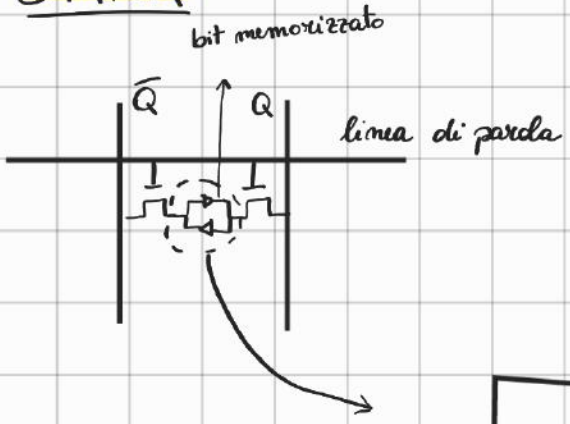
RAM
- DRAM
- SRAM

DRAM

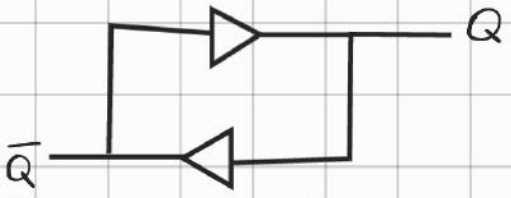


Memorizza un bit come presenza o assenza di carica in un condensatore. Il transistor si comporta come un interruttore che commette o discommette il condensatore dalla linea di bit. Quando la linea di parola è attiva il transistor si accende e il valore del bit viene trasferito alla o dalla linea di bit (lettura - scrittura). Il bit vale 1 se il condensatore è caricato a V_{DD}, altrimenti, se viene scaricato, il bit vale 0.

SRAM

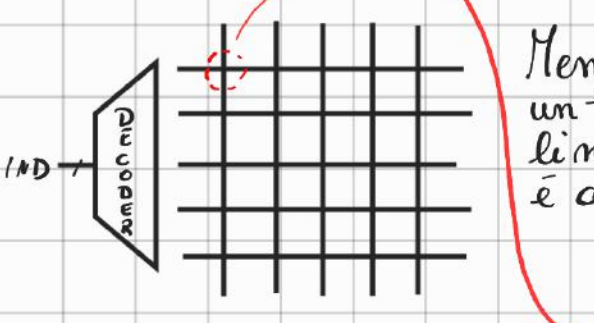


I bit immagazzinati non hanno bisogno di essere ricaricati. Il bit viene immagazzinato grazie a due negatori a croce, ogni cella possiede due uscite Q e $\bar{Q}$ e quando la linea di parola viene attivata entrambi i transistor si accendono e i bit di dato vengono trasferiti da o verso le linee di bit (lettura-scrittura).

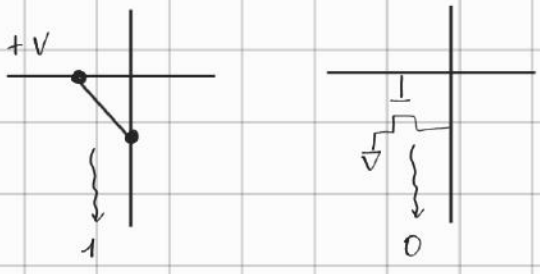


Rende stabile l'informazione contenuta nei due transistor.

ROM

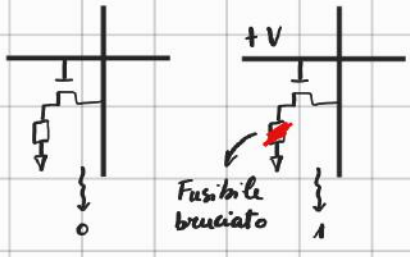


Memorizza un bit come presenza o assenza di un transistor. Se è presente, questo spinge la linea di bit ad un valore BASSO (0), se invece è assente la linea di bit resta ALTA (1).



PROM

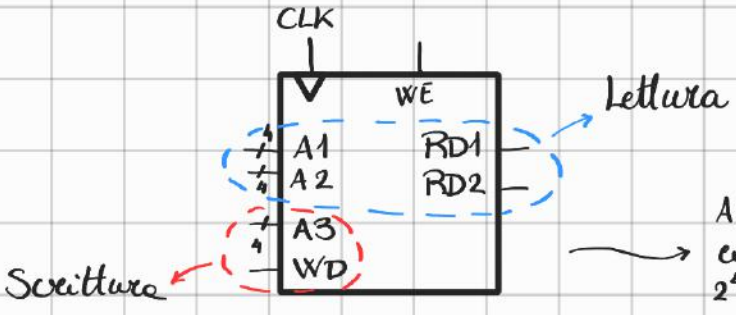
Ha un transistore in ogni cella ma consente di commettere o discommettere il transistore applicando un'alta tensione. Quindi inizialmente ogni cella é "inizializzata" a 0 e a scelta del programmatore é possibile "bruciare" il fusibile di una cella per ottenere il valore 1
Una volta bruciato il fusibile non puó piú essere utilizzato. Nelle EPROM questo procedimento si puó ripetere n-volte perché i fusibili vengono accesi o spenti e non bruciati.



Throughput

Quantitá di bit scambiabili per unitá di tempo.
Le DRAM hanno un throughput nettamente inferiore alle SRAM, perché devono "rinfrescare" periodicamente i dati, oltre che dopo ogni lettura.
Anche la latenza di una DRAM é piú lungo di quello di una SRAM perché la sua linea di bit non é pilotata in maniera attiva da un transistore.

Register file

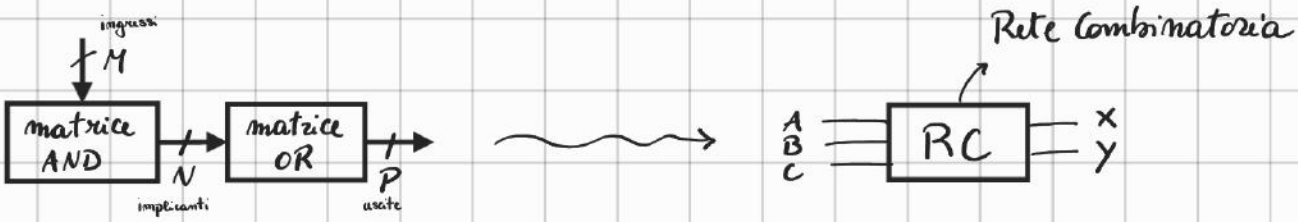


A1, A2, A3 avendo 4 bit ciascuno, possono accedere a $2^4 = 16$ registri.

I sistemi digitali usano spesso un certo numero di registri per immagazzinare variabili temporanee. Questo gruppo di registri prende il nome di "banco di registri", solitamente costruito come una piccola componente SRAM multi-porta poiché piú compatta rispetto ad una matrice di flip-flop.

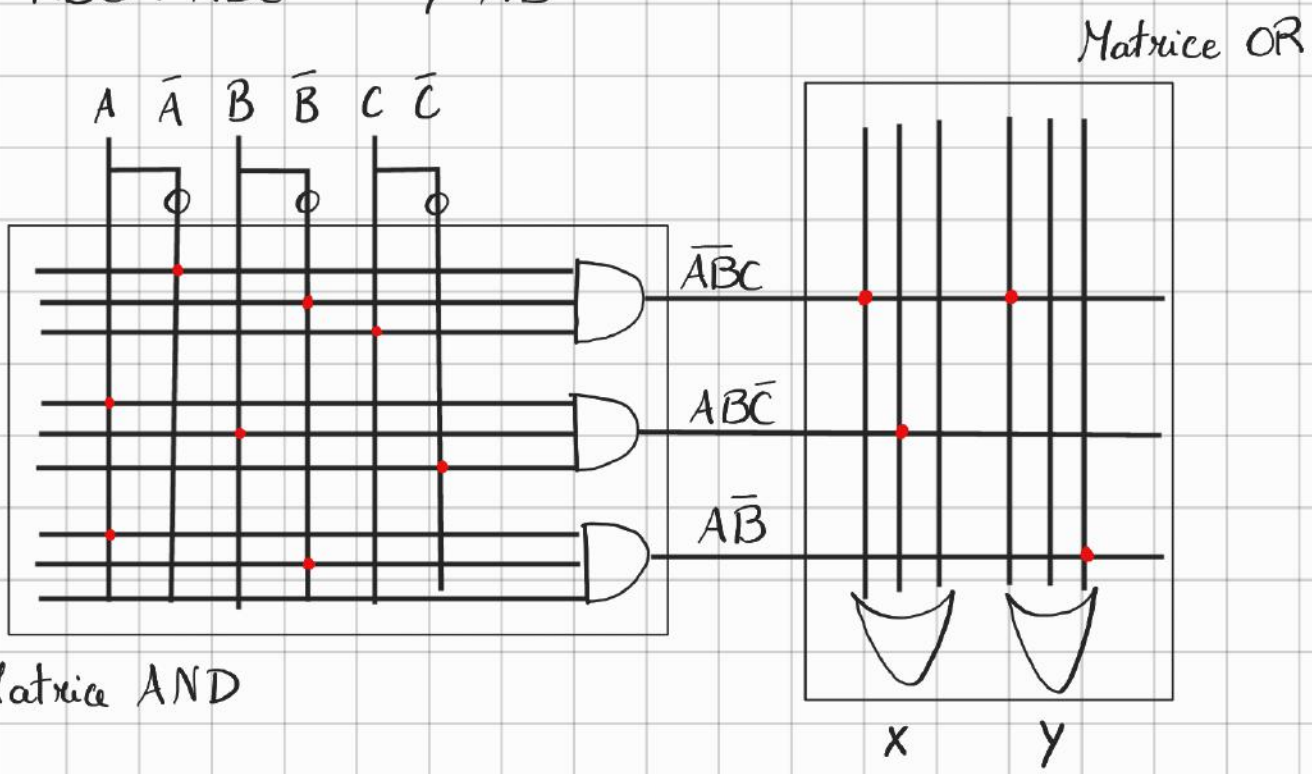
PLA

Realizzano funzioni combinatorie a due livelli nella forma somma di prodotti. Sono costituite da una matrice AND seguita da una matrice OR.



$M \times N \times P$

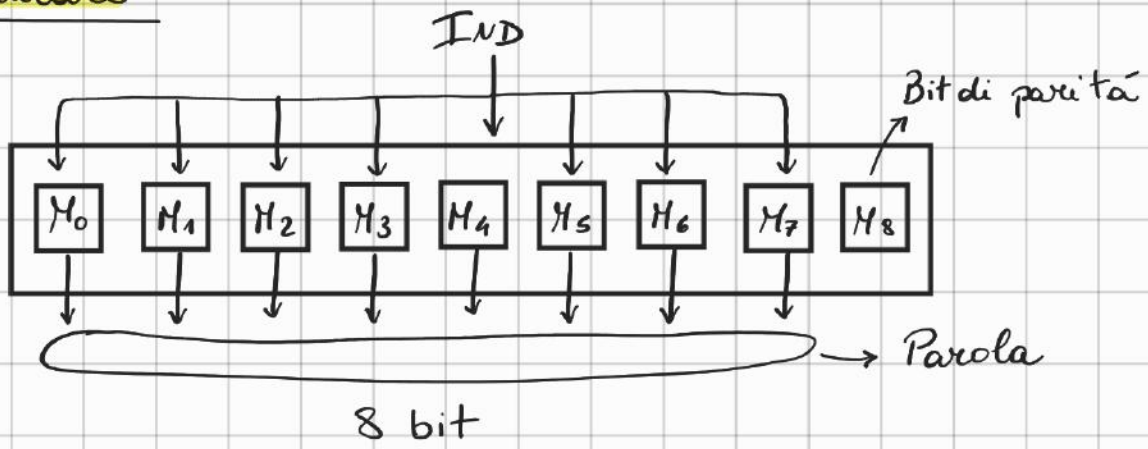
$X = \bar{A}\bar{B}C + AB\bar{C} \qquad Y = A\bar{B}$



SPLD

Sono PLA modificate che aggiungono dei registri e diverse altre caratteristiche alle matrici AND/OR. Tuttavia PLA e SPLD sono state rimpiazzate in larga misura dalle FPGA, che sono più flessibili ed efficienti per costruire grandi sistemi.

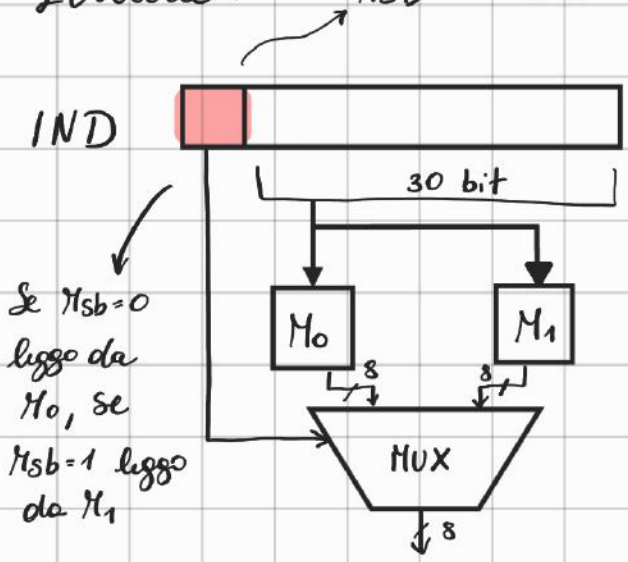
Memoria modulare



Preso un indirizzo in ingresso, accede ad ogni singola memoria e produce una parola da 8 bit formata dai valori di ogni memoria.

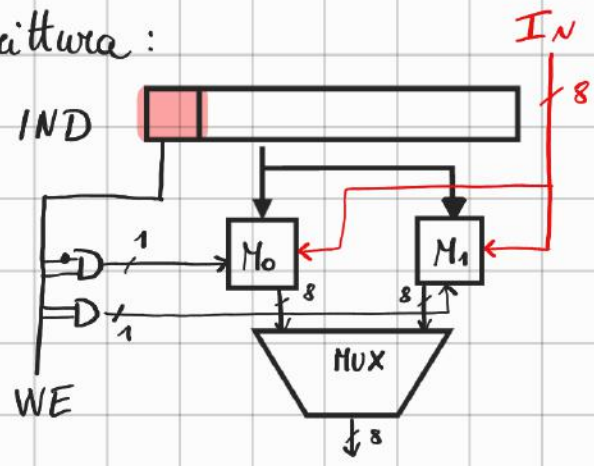
Memoria modulare sequenziale

Lettura: Msb → Indica il modulo di memoria
- Se Msb = 0 → M0
- Se Msb = 1 → M1



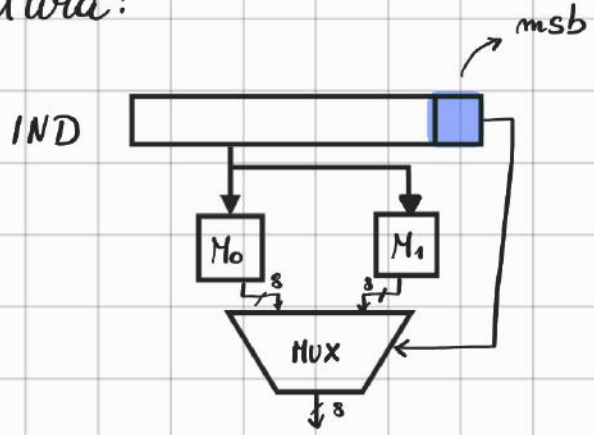
Nella memoria modulare sequenziale la memoria viene diviso in due moduli M0 e M1.

Scrittura:



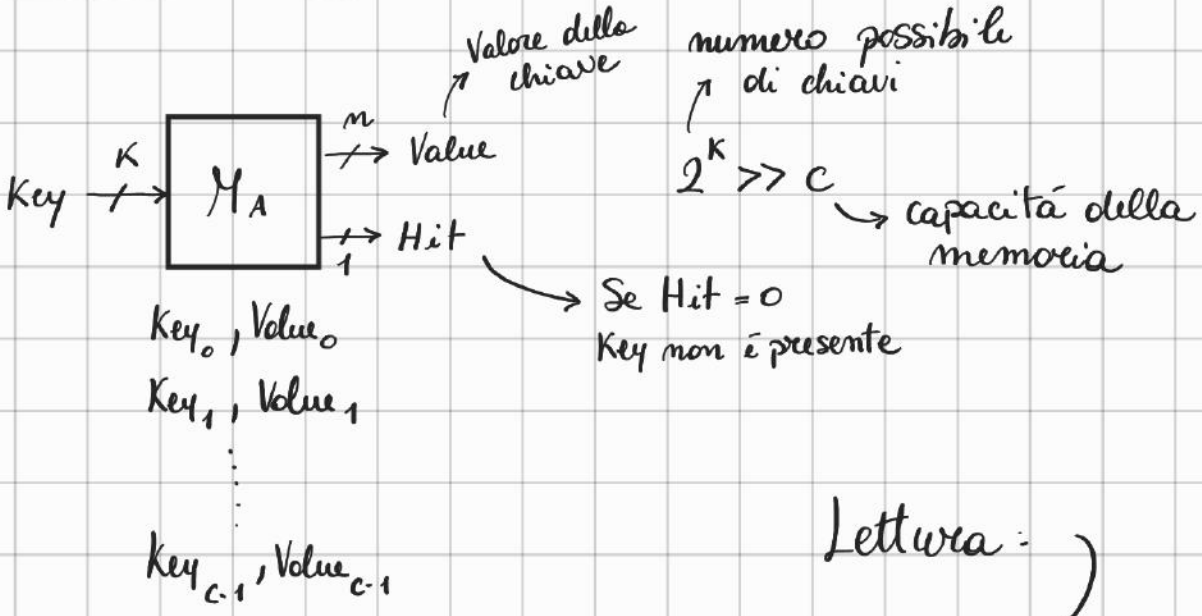
Memoria modulare interallacciata

Lettura:



Nella memoria modulare interallacciata invece di dividere la memoria a metá viene assegnata la prima parola al modulo M_0, la seconda al modulo M_1, la terza di nuovo a M_0 e cosi via ciclicamente.

Memoria Associativa

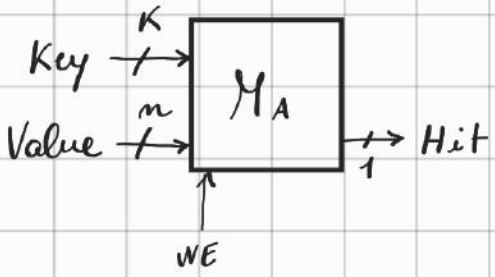


Lettura:

Prese un certo numero di chiavi e valori (c-1) implementati come registri, (Key_i corrisponde a $Value_i$), abbiamo bisogno di C confrontatori che prendono in input un ingresso da K bit che é la chiave corrispondente (Key_i) e la chiave che vogliamo cercare anch' esso da K bit. Ogni confrontatore darà in output 1 se la chiave cercata é presente in quella posizione (Key_i), 0 altrimenti e questi output vengono fatti passare da un OR. Prendiamo adesso un multiplexer che prenderá come input gli output dei registri valore e produrrá in output "value".

Inseriamo un encoder che riceverá in input c-bit e produrrá un output da $\log_2 c$ bit che entrerá come selettore nel nostro multiplexer.

Scrittura:



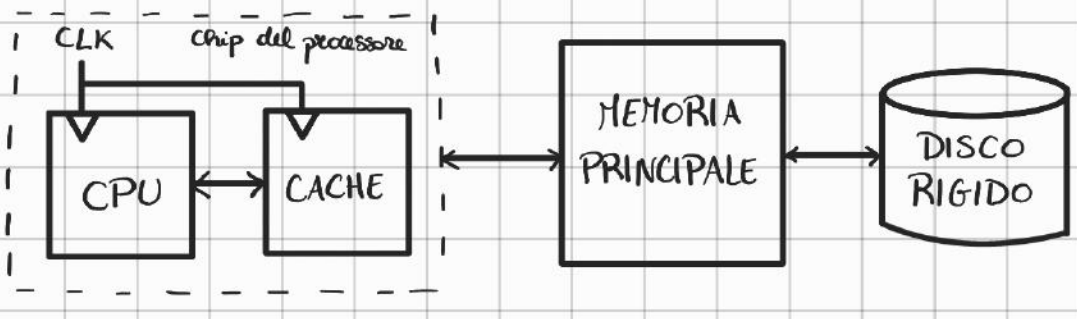
La differenza con la lettura é che il multiplexer viene sostituito da un demultiplexer che riceverá in ingresso il WE e come selettore avrá l'output dell'encoder.

Cache

Per rendere l'esecuzione di molti programmi piú rapida, i calcolatori memorizzano istruzioni e dati usati piú di frequente in una memoria piú veloce ma piú piccola, denominata cache e costituita generalmente da SRAM situata a bordo dello stesso chip del processore.

Ricerca di un dato in memoria

Il processore per prima cosa cerca il dato in cache. Se lo trova si parla di cache hit, se non lo trova si parla di cache miss e continua a cercare il dato nella memoria principale (fisica). Se non lo trova neanche lí, lo preleva dal disco rigido (virtuale) molto capiente ma lento.



Miss rate

$$\text{miss rate} = \frac{\#\ \text{miss}}{\#\ \text{Accessi in memoria}} = 1 - \text{hit rate}$$

Hit rate

$$\text{hit rate} = \frac{\#\ \text{hit}}{\#\ \text{Accessi in memoria}} = 1 - \text{miss rate}$$

Tempo medio di accesso in memoria

É il tempo medio di attesa da parte del processore per completare un'istruzione di lettura o scrittura in memoria.

$$AMAT = t_{cache} + TM_{cache}\,(t_{MP} + TM_{MP}\,t_{MV})$$

t_{cache} = tempo di accesso della cache
t_{MP} = tempo di accesso della memoria principale
t_{MV} = tempo di accesso della memoria virtuale
TM_{cache} = miss rate della cache
TM_{MP} = miss rate della memoria principale

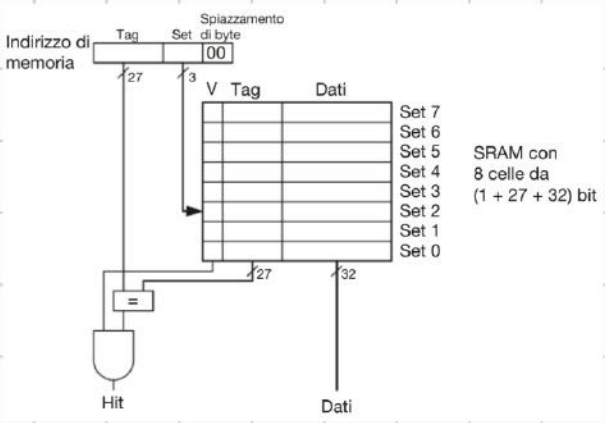
Memorie cache

Le cache usano la località spaziale e quella temporale per prevedere quali dati saranno saranno necessari nel prossimo futuro. Sono dimensionate in termini di capacità (C), numero di set (S), dimensione del blocco (b), numero di blocchi (B) e grado di associatività (N).
Località temporale significa che il processore ha un'elevata probabilità di accedere nuovamente nel prossimo futuro a un dato se lo ha utilizzato da poco.

Località spaziale significa che, quando il processore accede a un certo dato, ha un' elevata probabilità di accedere nel prossimo futuro ad altri dati in locazioni di memoria vicine al dato in questione.

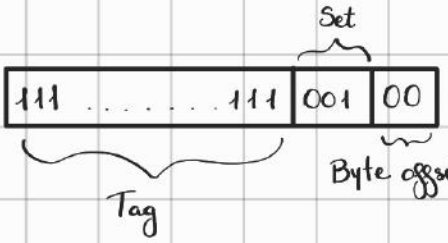
Direct mapped cache

Una cache a mappatura diretta ha un solo blocco in ogni set, quindi è organizzata in S=B set. Un qualsiasi indirizzo di memoria principale è mappato in un sdo blocco della cache.



Questo esempio raffigura una cache a mappatura diretta con capacità C = 8 parole e dimensione del blocco b = 1 parola. La cache ha quindi S = B = C/b set, ovvero 8, ciascuno contenente un blocco da una parola.

I due bit meno significativi dell'indirizzo sono sempre 00, perché le parole sono allineate ai bordi di parola. I successivi $\log_2 8 = 3$ bit indicano il set nel quale l'indirizzo di memoria viene mappato. I restanti bit più significativi sono denominati tag e indicano quale dei tanti indirizzi è presente in quel set.
In una cache a mappatura diretta l'indirizzo è quindi suddiviso nel seguente modo:



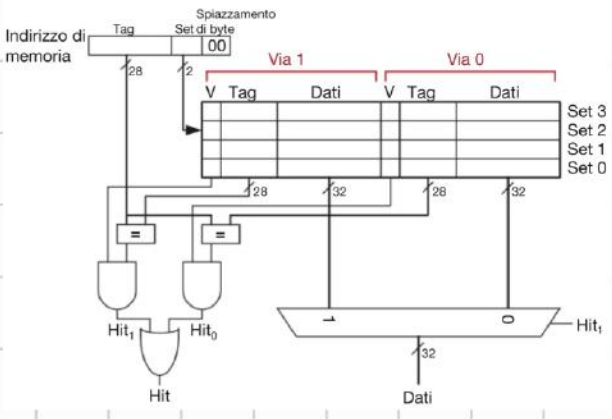
Il byte offset o spiazzamento di byte vale 00. (ogg)
Il Set si calcola con $\log_2 S$. (set)
Il tag sono i restanti bit → 32 - set - ogg

Dimensione in bit dell'indirizzo

La cache usa un bit di validitá per ogni set che indica se il set contiene dati significativi. Se il bit vale 0, il contenuto del set non é significativo.
Un' istruzione di lettura legge l'elemento della cache e controlla i bit di tag e il bit di validitá, se il tag corrisponde ai 27 bit piú significativi dell'indirizzo e il bit di validitá é 1, si é verificato un hit e il dato viene restituito al processore. In caso contrario, si é verificato un miss e il dato viene prelevato dalla memoria principale. Quando due indirizzi vengono mappati nello stesso blocco di cache si verifica un conflitto e il dato cui si accede per ultimo espelle il precedente dal blocco.

Set associative cache

Una cache parzialmente associativa a N vie ha S = B/N set e ogni set contiene N blocchi. L'indirizzo di memoria é mappato in un solo set ma il dato corrispondente a tale indirizzo puó finire in uno qualsiasi degli N blocchi di quel set.



Questo esempio raffigura una cache parzialmente associative con capacita C = 8, N = 2 vie e S = 4 set quindi oltre ai soliti due bit 00 del byte offset, qui si usano $\log_2 4 = 2$ bit per il set e il tag aumenta da 27 a 28 bit. Ogni via é costituita da un blocco di dati e dai bit di validitá e di tag, la cache legge i blocchi di entrambe le vie del set selezionato e controlla i tag e i bit di validitá: se si verifica un hit in una delle due vie, un multiplexer selezione il dato da tale via. Le cache parzialmente associative hanno

solitamente un miss rate inferiore alle cache a mappatura diretta di pari capacità perché presentano meno conflitti. Sono però più costose da realizzare a causa dei multiplexer di uscita e dei comparatori aggiuntivi.

Fully associative cache

Una cache completamente associativa ha solo S = 1 set. Un dato può andare in uno qualsiasi dei B blocchi del set quindi può essere definita come una cache parzialmente associativa a B vie.

Questo esempio raffigura una cache completamente associativa con b = 8 blocchi. Alla richiesta di un dato, dlo confronti di tag devono essere fatti, dal momento che il dato potrebbe trovarsi in un blocco qualsiasi. Un multiplexer 8:1 seleziona il dato corretto in caso di hit, in conclusione queste cache hanno il minore numero di miss causati da conflitti ma richiedono hardware aggiuntivi per i confronti dei tag.

Miss penalty

Quando si ha una dimensione del blocco elevata, significa che a pari capacità di cache essa ha meno blocchi quindi può dare luogo ad un maggiore numero di conflitti. Inoltre serve più tempo in caso di miss per prelevare dalla memoria principale tutte le parole del blocco e trasferirle in cache: il tempo necessario per caricare in cache il blocco mancante viene

detto penalizzazione di miss.

LRU

In una cache a mappatura diretta, ogni indirizzo viene mappato in un unico blocco e set, quindi se un set è pieno e vogliamo caricare in cache un dato il blocco in quel set viene sostituito dal nuovo dato. In una cache parzialmente o completamente associativa, invece, bisogna scegliere quale blocco espellere se il set è pieno. La località temporale ci suggerisce che la scelta migliore sia quella di espellere il blocco utilizzato meno recentemente perché è quello che con meno probabilità verrà riutilizzato a breve. Questa tecnica prende il nome di LRU.
Per cache parzialmente associative a più di due vie si usa la politica pseudo-LRU che divide le vie in gruppi e U indica il gruppo di vie usato meno recentemente. Al momento dello sostituzione il nuovo blocco sostituisce un blocco a caso del gruppo U.

Riduzione del miss rate

Il primo passo per ridurre il miss rate è quello di capirne le cause. Esistono vari tipi di miss:

- miss inevitabile: avviene alla prima richiesta di un blocco di cache perché il blocco deve essere per forza letto dalla memoria principale.
- miss di capacità: avviene quando la cache è troppo piccola per contenere tutti i dati utilizzati in modo concorrente.
- miss di conflitto: avviene quando più indirizzi vengono mappati

nello stesso set ed espellono blocchi ancora necessari.

Al crescere delle capacità delle cache i miss di capacità diminuiscono. Aumentare il grado di associatività diminuisce il numero di miss di conflitto ma oltre i 4 o 8 si hanno minime riduzioni del miss rate.

Write - through

In una cache write-through, il dato viene scritto simultaneamente sia nella cache che nella memoria principale.

Write - back

In una cache write-back, un bit di modifica M (dirty bit) è associato a ogni blocco di cache: tale bit vale 1 se il blocco è stato modificato da almeno una scrittura, altrimenti vale 0.
I blocchi di cache modificati vengono riscritti in memoria principale solo al momento di essere espulsi dalla cache.
Pro e contro: una cache write-through non ha bisogno del dirty bit ma generalmente fa più accessi in memoria principale.

Memoria virtuale

La memoria virtuale è un'unità contenente uno o più dischi rigidi ciascuno dotato di una testina di lettura/scrittura posizionata sulla punta di un braccio triangolare. La testina si sposta sulla corretta locazione e sfrutta l'elettromagnetismo per leggere o scrivere dati sul disco in rotazione sotto di lei.
I programmi possono accedere a un qualsiasi dato della

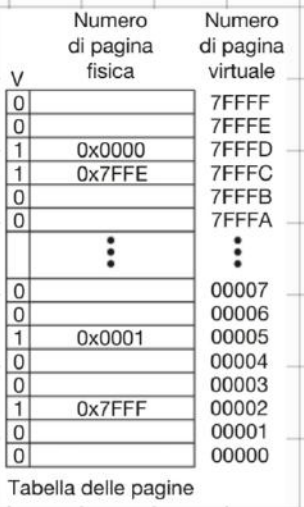
memoria virtuale utilizzando indirizzi virtuali che indicano le locazioni dei dati nella memoria virtuale.
É inoltre suddivisa in pagine virtuali, tipicamente da 4KB, esattamente come le pagine fisiche. Il processo di determinare l'indirizzo fisico a partire da quello virtuale prende il nome di traduzione dell'indirizzo. Se il processore tenta di accedere ad un indirizzo virtuale non presente in memoria fisica, si verifica il cosiddetto "page fault".
Per effettuare la traduzione dell'indirizzo si usa la tabella delle pagine che ha un elemento per ogni pagina virtuale.
Ogni elemento della tabella delle pagine indica in quale pagina fisica tale pagina virtuale si trova, oppure se é presente solo su disco. Ogni istruzione richiede quindi un accesso alla tabella e uno in memoria fisica, per velocizzare questo processo si usa un translation lookaside buffer (TLB) che tiene in cache gli elementi delle tabelle di utilizzo più frequente.

Traduzione dell'indirizzo

I bit più significativi dell'indirizzo virtuale e fisico indicano il numero di pagina mentre i bit meno significativi indicano la singola parola all'interno della pagina e sono denominati Spiazzamento della pagina.
Per determinare il numero di pagine (virtuali o fisiche) bisogna dividere la dimensione della memoria (virtuale o fisica) con la dimensione di una pagina (virtuale o fisica).

NPV: numero di pagina virtuale
NPF: numero di pagina fisica

Tabella delle pagine



V	Numero di pagina fisica	Numero di pagina virtuale
0		7FFFF
0		7FFFE
1	0x0000	7FFFD
1	0x7FFE	7FFFC
0		7FFFB
0		7FFFA
	⋮	⋮
0		00007
0		00006
1	0x0001	00005
0		00004
0		00003
1	0x7FFF	00002
0		00001
0		00000

Tabella delle pagine

La tabella delle pagine é dotata di un elemento per ogni pagina virtuale, ogni elemento contiene un numero di pagina fisica e un bit di validitá V: se tale bit vale 1, la pagina virtuale é mappata nella pagina fisica specificata, altrimenti la pagina virtuale va caricata da disco. Tale tabella viene tipicamente memorizzata in memoria fisica in un registro dedicato chiamato "registro di tabella delle pagine". Per effettuare una lettura o una scrittura, il processore deve prima tradurre l'indirizzo virtuale in fisico, poi accedere al dato in memoria fisica. Per fare questo estrae il NPV e lo somma al registro di tabella delle pagine per trovare l'indirizzo fisico dell'elemento della tabella delle pagine, poi legge tale elemento per ottenere il NPF e se l'elemento é valido viene unito il NPF con lo spiazzamento ottenendo l'indirizzo fisico. Questo processo comporta due accessi in memoria.

Translation lookaside buffer (TLB)

Il TLB é organizzato come una cache completamente associativa, in grado di memorizzare da 16 a 512 elementi: ogni elemento contiene un NPV e il corrispondente NPF. Si accede al TLB mediante il NPV, se si verifica hit, il TLB restituisce il corrispondente NPF altrimenti bisogna leggere la tabella delle pagine. Il TLB é abbastanza piccolo da potere essere interrogato in meno di un ciclo di clock.

Paging

Scrivere su disco una pagina fisica e caricare nella stessa pagine un'altra pagina virtuale. Il disco rigido usato nella memoria virtuale prende il nome di "swap area".
Il processore svuota una delle pagine fisiche usate meno di recente quando si verifica un page fault e vi carica la pagina virtuale mancante. Per supportare queste politiche di sostituzione, ogni elemento della tabella delle pagine contiene due ulteriori bit di stato: il bit di modifica M e il bit di utilizzo U.
Il bit di modifica vale 1 se una qualsiasi istruzione di scrittura in memoria ha modificato la pagina fisica.
Il bit di utilizzo vale 1 se la pagina fisica è stata recentemente utilizzata.