

# Politiche di rimpiazzamento (cache)

## MIN

Strategia ideale: scegliamo il blocco/pagina che non sarà utilizzato per il tempo più lungo. Difficile o impossibile da prevedere.

## Random

Funziona se il numero di vie è molto maggiore della frazione del WS presente in ciascuna via / se il numero di pagine fisiche per la chache è molto maggiore della dimensione del WS.

È semplice e veloce, perciò è utile nelle cache L1, dove c'è poco tempo per scegliere la vittima e il costo dei fault è limitato dagli altri livelli di cache.

## FIFO

La vittima è la linea / pagina caricata da più tempo.

Ha un caso pessimo comune: si scorre ripetutamente un array troppo grande per stare interamente in cache.

Ref.	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A			E				D			C				
2		B			A				E			D			
3			C			B			A			E			
4				D		C			B						

## Anomalia di Belady

La politica FIFO può comportarsi peggio con cache più grandi:

FIFO (3 slots)															
Ref.	A	B	C	D	A	B	E	A	B	C	D	E			
1	A			D			E					+			
2		B			A			+		C					
3			C			B		+		D					

  

FIFO (4 slots)															
Ref.	A	B	C	D	A	B	E	A	B	C	D	E			
1	A				+		E				D				
2		B				+		A				E			
3			C					B							
4				D					C						

## Least Recently Used

Per i principi di località, se un blocco è stato utilizzato di recente probabilmente è ancora nel working set, perciò la politica LRU è una buona approssimazione di MIN.

Svantaggi: costoso per la necessità di memorizzare e confrontare i tempi di accesso, stesso caso pessimo di FIFO:

LRU															
Ref.	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A			E			D			C					
2		B			A			E			D				
3			C			B			A			E			
4				D		C			B						

  

MIN															
Ref.	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
1	A					+					+				
2		B					+				+	C			
3			C					+	D			+			
4				D	E					+			+		

## Not Recently Used

Approssimazione di LRU: si mantiene un bit di accesso, che viene impostato a 1 ad ogni operazione di lettura/scrittura. Periodicamente tutti i bit vengono resettati a 0. Quando è necessario scegliere una vittima,

**scelta casuale** tra quelli con bit a 0 (o tra tutti se non ce n'è nessuno). Basso overhead, usato per le cache dati/istruzioni;

**clock algorithm / kth chance** il SO scorre periodicamente tutte le pagine, incrementando un contatore per quelle con  $U = 0$ , resettando il contatore a  $U$  per le altre (con conseguente TLB shootdown). Le pagine con contatore  $k$  sono marcate come vittime. Second chance: se  $U = 0$  allora vittima.