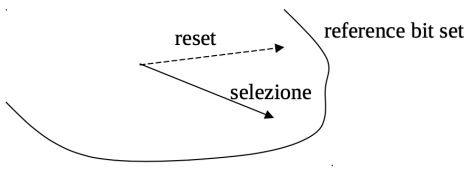
Algoritmo dell’orologio (Not Recently Used)

Andiamo a vedere come all'interno di un sistema operativo reale e moderno viene ad essere eseguita la sostituzione delle pagine all'interno degli schemi di memoria virtuale.

La vittima è una pagina che non è stata utilizzata di recente. Ma non è nemmeno la pagina che è stata utilizzata più tempo fa, è una non utilizzata di recente.
Per ogni frame associamo un "reference bit": ci dice se quel frame è stato utilizzato o no.
Nelle architetture moderne, questo reference bit, lo abbiamo rappresentato esattamente all'interno della page table: all'interno della page table c'è un bit che viene posto ad 1 quando il sottosistema di controllo del mio processore va ad utilizzare la entry sulla page table stessa, per effettuare la traduzione tra un indirizzo logico e il corrispettivo indirizzo fisico.
Quindi quando per esempio questa entry della tabella viene ad essere utilizzata in termini di caricamento anche all'interno del TLB.
Il bit che noi associamo al frame corrisponde esattamente al frame che noi identifichiamo nell'indirizzo fisico.
Quindi il frame F che è presente eventualmente nella nostra RAM.
Questa informazione di controllo che ci serve per gestire l'algoritmo dell'orologio ce lo abbiamo già all'interno della page table, e viene portata da 0 a 1 quando il sottosistema di controllo del processore accede alla page table.

- un **reference bit** è associato a ciascun frame
- il reference bit è impostato ad 1 ogni volta che la pagina in quel frame e referenziata in lettura o scrittura
- periodicamene, il sistema operativo scorre parte dell’insieme dei reference bit e li imposta a 0 (lancetta del reset)
- quando una sostituzione di pagina è necessaria allora il sistema operativo scorre i reference bit fino a che non ne trova uno impostato a 0 (lancetta di selezione)
- la pagina corrispondente viene selezionata per la sostituzione durante quest’ultimo scorrimento, reference bit ad 1 vengono resettati



La lancetta di selezione fa un giro completo in caso lo use bit di tutti i frame sia reimpostato ad 1 dall’ultimo reset

Supponiamo che poi sia necessaria una sostituzione di pagina.
Per selezionare la vittima scorriamo questi reference bit fino a che non né troviamo uno impostato a 0. E la pagina corrispondente a quel frame viene selezionata per la sostituzione.
Mentre la lancetta di selezione scorre i bit fino a che non ne trova uno impostato a zero, resetta questi reference bit che sono a 1 con la lancetta di reset.
La speranza è che così se troviamo tutti questi frame utilizzati di recente, dopo un giro magari riusciamo ad essere tranquilli che troviamo il valore 0 per uno di questi che abbiamo correntemente resettato.

Pagine che erano state portate all'interno della RAM e che correntemente poi sono state resettate nel bit di utilizzo e poi non più accedute, questa lancetta di selezione quando scorrerà se le troverà col bit a zero, e quindi se le troverà come buone vittime per eliminarle.

Algoritmo dell’orologio con bit addizionali

Il Dirty Bit ci dice se una pagina è stata toccata in scrittura. Quindi se consideriamo entrambi ossia il "Reference Bit" che ci dice se questa pagina è stata toccata di recente, e il "Dirty Bit" che ci dice se comunque la pagina è stata sporcata, abbiamo 4 combinazioni:

- il **reference bit** viene manipolato come nel caso dell’algorimo dell’orologio classico
- il **dirty bit** viene usato in combinazione al **reference bit** per determinare quale pagina sostituire

Possibili combinazioni

(rb=0, db=0) (rb=1, db=0) (rb=0, db=1) (rb=1, db=1)

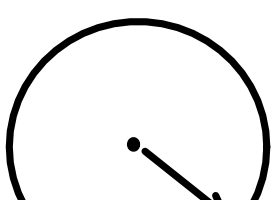
Pagina preferita per la sostituzione perchè, oltre a non essere stata usata di recente, non necessita di essere ricopiata su disco

Es. Sistemi della Apple

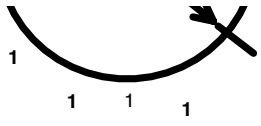
Qui la selezione della vittima la possiamo fare utilizzando anche il dirty bit, scegliendo come vittima una pagina il cui dirty bit sia impostato a zero.
La pagina migliore è quella in cui il dirty bit è zero, e il reference bit è zero: questa è una pagina che è in memoria, non è stata mai aggiornata (db=0) e non è stata toccata di recente (rb=0).
Questa è un'ottima vittima perché se dobbiamo caricare qualcosa all'interno di quel corrispettivo frame della RAM, non abbiamo neanche bisogno di portare il contenuto corrente fuori dalla RAM.
Quel contenuto non è aggiornato. Quindi non abbiamo bisogno di allineare il relativo contenuto all'interno di una swap area con il nuovo contenuto che avevamo generato all'interno della RAM.

È sempre possibile identificare la vittima?

Se noi consideriamo la lancetta di reset che gira, questa lancetta che gira sul quadrante porta da 1 a 0 tutti i bit che vengono incrociati e dopo un giro io spererei di trovare lo zero nel punto in cui ero passato in precedenza e da dove ero partito, è sempre possibile avere questo tipo di comportamento all'interno di un sistema operativo?



La risposta è SI nel caso in cui abbiamo una esecuzione atomica. Atomica significa dire che mentre questa lancetta sta effettuando le sue attività, il che implica dire che sta girando un thread a livello kernel del S.O per eseguire questa selezione e quindi per fare il giro su questi bit e aggiornarli, non può succedere nulla all'interno dell'esecuzione su questa macchina.
Perché se qualcos'altro può succedere è possibile che mentre questo thread gira in



modo kernel per cercare la vittima, su un altro CPU-CORE altri thread stanno eseguendo. Il che implica dire mentre faccio il giro con la lancetta cercando di resettare l'utilizzo dei frame, questi thread riutilizzano questi frame.
Implica dire che, chiuso il giro e ritorno esattamente nel punto iniziale, posso non trovare lo zero che avevo settato al giro prima.
Questo significa eseguire in maniera NON ATOMICA.

- no nel caso di esecuzione non atomica
  - ✓ reset del reference bit eseguito sul thread X
  - ✓ set dello stesso bit rieseguito sul thread Y