

LA MEMORIA VIRTUALE ALGORITMI DI SOSTITUZIONE DELLE PAGINE

Danilo Croce

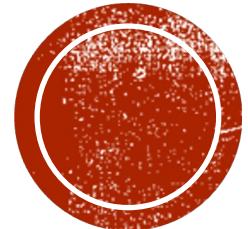
Novembre 2023



Negli anni '80 molte università utilizzavano un sistema timesharing con dozzine di utenti (più o meno soddisfatti) che lavoravano contemporaneamente su un sistema VAX da 4 MB.

Oggi Microsoft raccomanda di avere almeno 2 GB per un sistema a 64 bit con Windows 10.





VIRTUAL MEMORY

GESTIONE DELLA MEMORIA: OUTLINE

- Memory Abstraction
- **Virtual Memory**
- Algoritmi di sostituzione delle pagine
- Problemi di Progettazione per Sistemi di Paging



IL PROBLEMA DEL BLOATWARE E LA CRESCITA DELLA MEMORIA

- Necessità di **gestire programmi che superano la capacità della memoria** disponibile.
 - Problema dei programmi più grandi della memoria esiste sin dalle origini, specialmente nelle scienze e nell'ingegneria.
- Negli anni '60, introduzione di **tecniche per dividere programmi in parti gestibili**.
 - **Overlay**: sono piccole parti o segmenti di un programma.
 - **Solo l'overlay necessario viene caricato in memoria**.
 - Overlay successivi sovrascrivono o coesistono con quelli precedenti.
 - Gli overlay vengono scambiati tra memoria e disco.
- Originariamente, i programmatori dovevano suddividere manualmente i programmi in overlay.
 - Questa soluzione era tediosa e soggetta a errori.



MEMORIA VIRTUALE

- La **memoria virtuale** estende l'idea dei registri base e limite.
 - Ogni programma ha un proprio spazio degli indirizzi suddiviso in "**pagine**", che sono intervalli di indirizzi contigui.
 - Non tutte le pagine devono essere contemporaneamente nella memoria fisica:
 - l'hardware crea una mappa di quelle direttamente in memoria
 - se una pagina manca, il sistema operativo interviene
- La maggior parte dei sistemi moderni usa il "**paging**" (paginazione)
 - divisione dello spazio degli indirizzi in unità di dimensione fissa, es. 4 KB.
- **Alternativa:** "**segmentazione**" con unità di dimensione variabile:
 - **ora meno comune.**



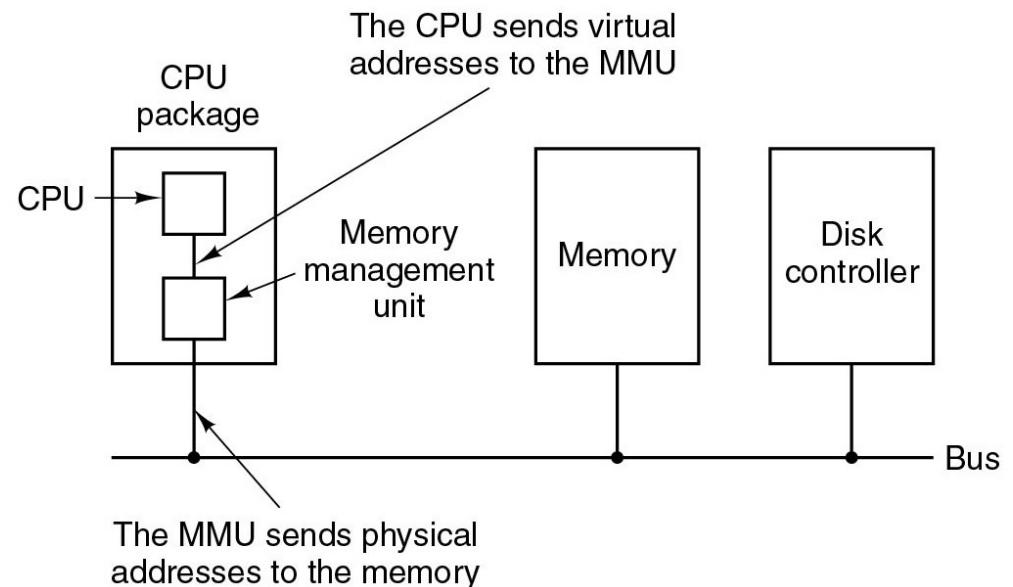
MEMORIA VIRTUALE (2)

- **Problema:**

- Finora la memoria può essere assegnata ai processi solo in blocchi contigui.

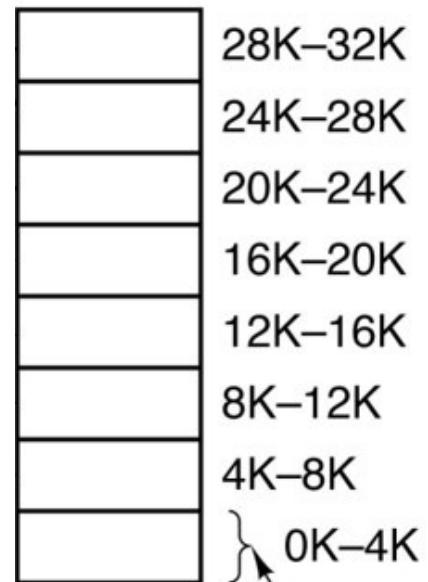
- **Soluzione (e vantaggio dell'uso della Memoria Virtuale):**

- Creare per il processo l'illusione di uno spazio di indirizzi ampio (ad esempio indicizzabile con 48 bit!!!).
- Questo spazio è noto come spazio di indirizzi virtuale
- La **RAM** (molto più limitata) è nota come **memoria fisica**.
- **Memory Management Unit (MMU):** traduce gli indirizzi virtuali (come usati dal processo) in indirizzi fisici



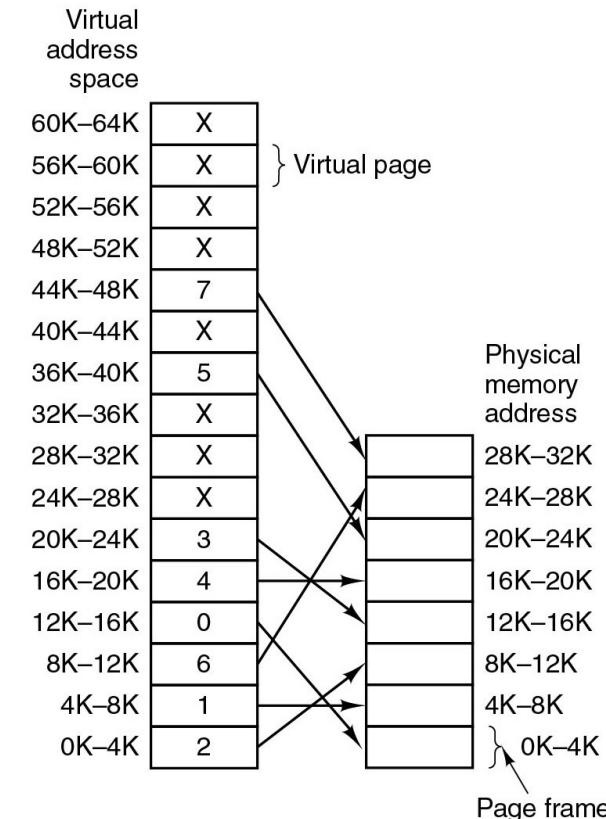
MEMORIA VIRTUALE E PAGINAZIONE

- I sistemi moderni utilizzano la **paginazione** (o *paging*):
 - Dividendo la memoria fisica e virtuale in pagine di dimensioni fisse
 - ad esempio 4096 byte o 4 KB
 - Traducendo le pagine virtuali in **pagine fisiche (frame)**



SPAZIO DI INDIRIZZAMENTO VIRTUALE VS. SPAZIO DEGLI INDIRIZZI FISICI E PAGE TABLE

- **Mappatura Memoria:**
 - 16 pagine virtuali possono essere mappate in 8 frame fisici usando la MMU.
 - Tuttavia, non tutte le pagine virtuali sono mappate fisicamente
 - quelle NON mappate sono contrassegnate con una X.
- Se un programma fa riferimento a una pagina non mappata, si verifica un «**Page fault**». Il sistema operativo allora:
 - Sposta un frame raramente usato su disco, se serve
 - Ma quale scegliere? Vedi più avanti
 - Carica la pagina richiesta nel frame libero o liberato.
 - Aggiorna la mappa della MMU per riflettere i cambiamenti.

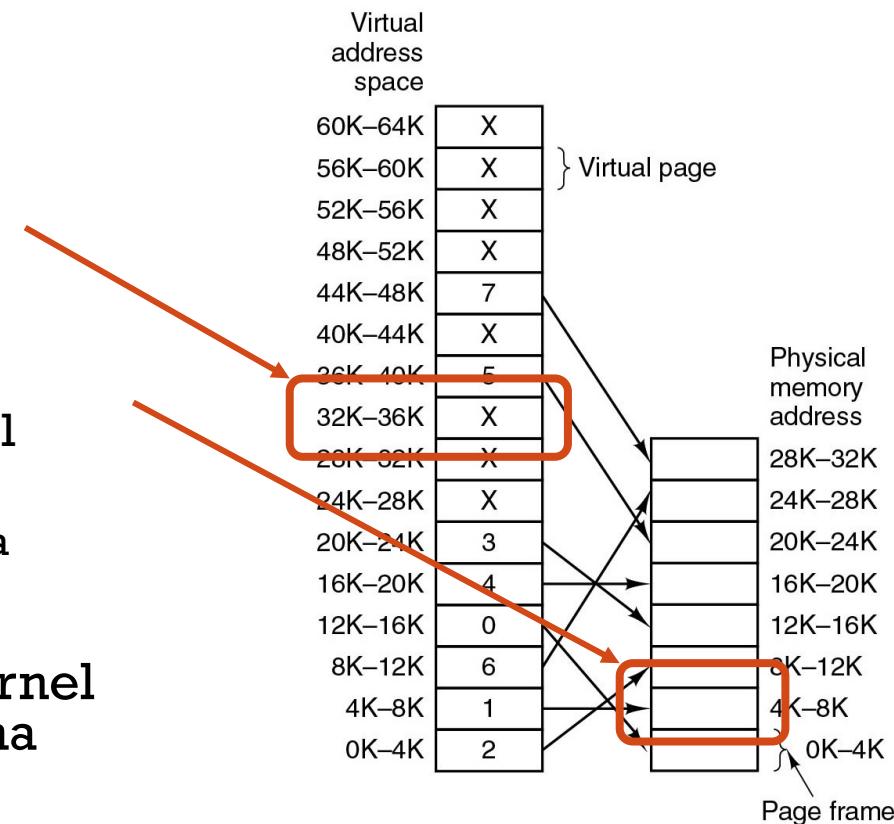


La **relazione** tra gli indirizzi di memoria virtuale e fisica è data dalla **Page Table**.



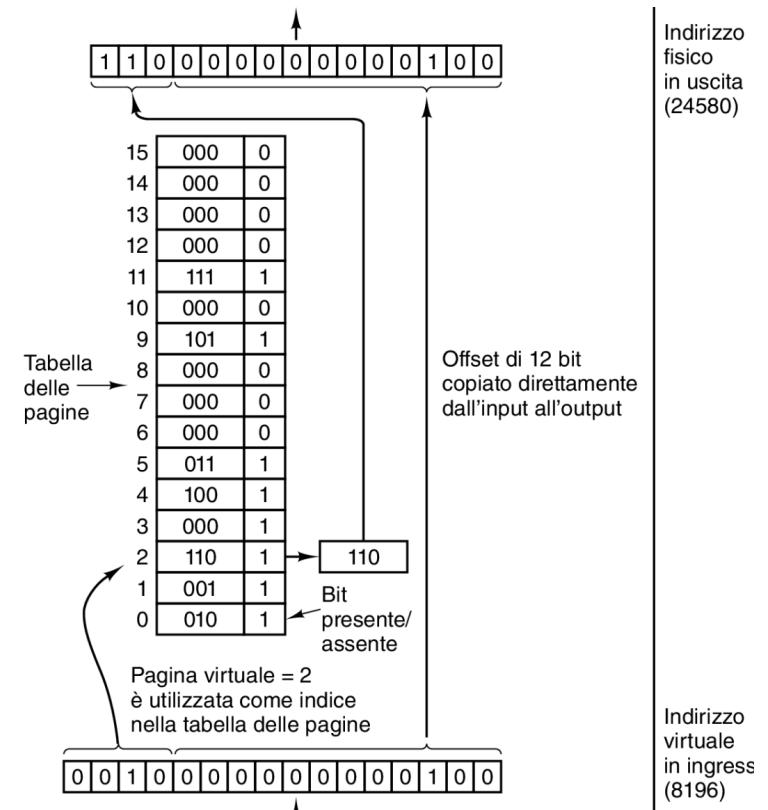
SPAZIO DI INDIRIZZAMENTO VIRTUALE VS. SPAZIO DEGLI INDIRIZZI FISICI E PAGE TABLE

- **Esempio:** gestire istruzione
 - MOV REG, 32780
- Fa riferimento alla pagina virtuale 8.
 - Indirizzo 12 della pagina
 - $32780 - 2^{15}$ (**32768**) = 12
- Se non è mappata, il sistema operativo potrebbe decidere di sostituire il frame 1
 - Spostando il precedente su disco
 - Popolando il nuovo frame e puntando poi a
 - $4108 = 4096 + 12$
- Il page fault avviene nello spazio kernel durante il «**trap**» eseguito dal sistema operativo



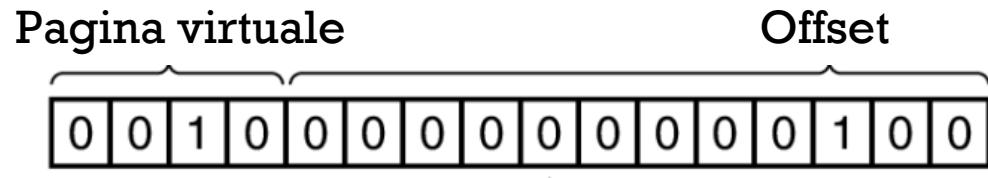
FUNZIONAMENTO INTERNO DELLA MMU

- **Indirizzo Virtuale:** 8196
 - **Rappresentazione Binaria:**
 - 0010 00000000100
 - **Suddivisione dell'Indirizzo Virtuale:**
 - **Numero di pagina:** 4 bit (permette di gestire 16 pagine)
 - **Offset:** 12 bit (indirizza 4096 byte per pagina che compongono ogni frame)
 - **Mappatura tramite la Tabella delle Pagine:**
 - Numero di pagina →
 - Indice nella tabella delle pagine →
 - Numero di frame



Funzionamento interno della MMU con 16 pagine da 4 kB.

EVOLUZIONE DEGLI INDIRIZZI E TABELLA DELLE PAGINE

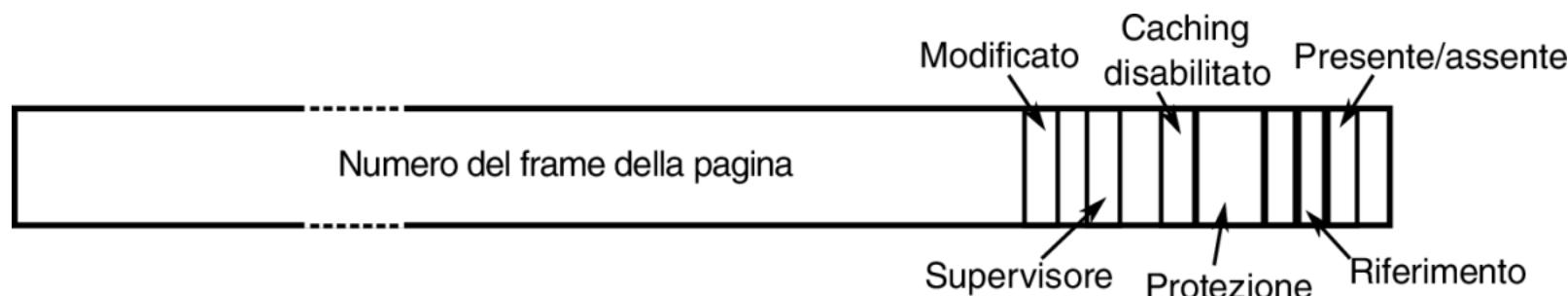


- **Indirizzi nei nostri esempi:** 16 bit (per chiarezza nelle illustrazioni)
- **Moderni PC:**
 - Usano indirizzi a 32 o 64 bit.
 - Con 32 bit e pagine da 4 KB:
 - 12 bit per indirizzare 4096 byte per pagina
 - Tabella delle pagine di $2^{(32-12)} = 2^{20} = 1.048.576$ voci! Una taglia di 4GB è «fattibile» anche per PC con «pochi» GB di RAM.
- **Indirizzi a 64 bit e pagine da 4 KB:**
 - Richiede 2^{52} voci ($\sim 4,5 \times 10^{15}$) nella tabella.
 - In realtà nei sistemi a 64 bit si usano 48 bit
 - 256 Terabyte bastano e avanzano... gli altri bit sono riservati per il futuro ...



COME È COMPOSTA UNA VOCE DELLA PAGE TABLE?

- Ogni voce ha informazioni cruciali come il numero del frame (es 12 bit per 4KB), come:
 - **Bit Presente/Assente**: Indica se la pagina virtuale è **in memoria**.
 - **Bit Protezione**: Specifica i **tipi di accesso consentiti** (lettura, scrittura, esecuzione).
 - **Bit Supervisor**: Stabilisce se la pagina è **accessibile** solo al sistema operativo o anche ai programmi utente.
 - **Bit Modificato (M) e Riferimento (R)**: Registrano l'uso della pagina.
 - Il bit Modificato si attiva quando la pagina viene scritta,
 - il bit Riferimento viene impostato ogni volta che si accede alla pagina.



- **Nota:** per un processo l'indirizzo in memoria della «sua» tabella delle pagine è scritto nel registro Page Table Base Register (PTBR)

VELOCIZZARE LA PAGINAZIONE - PROBLEMI CHIAVE

- **Mappatura Veloce:** Necessaria a ogni riferimento alla memoria. Ogni istruzione può richiedere più riferimenti alla tabella delle pagine.
 - **Sfida:** Se un'istruzione impiega 1 ns, la ricerca nella tabella delle pagine deve essere inferiore a 0,2 ns per evitare colli di bottiglia.
- **Dimensione della Tabella delle Pagine:**
 - **Contesto:** Con 48 bit di indirizzamento e pagine di 4 KB, ci sono 64 miliardi di pagine. Una tabella delle pagine per questo spazio indirizzi richiederebbe voci enormi.
 - **Problema:** Usare centinaia di gigabyte solo per la tabella delle pagine è impraticabile. Ogni processo richiede una propria tabella delle pagine.



APPROCCI ALLA SOLUZIONE

- **Tabella delle Pagine in Registri Hardware:**

- **Funzionamento:** Un registro hardware per ogni pagina virtuale, caricato all'avvio del processo.
- **Vantaggi:** Semplice, non richiede accessi alla memoria durante la mappatura.
- **Svantaggi:** Costoso con tabelle delle pagine grandi, ricaricare l'intera tabella ad ogni cambio di contesto è inefficiente.

- **Tabella delle Pagine in Memoria Principale:**

- **Funzionamento:** La tabella delle pagine è interamente in RAM, con un registro che punta al suo inizio.
- **Vantaggi:** Facile da cambiare a ogni cambio di contesto, richiede solo il ricaricamento di un registro.
- **Svantaggi:** Richiede accessi frequenti alla memoria, rendendo la mappatura più lenta.



PROBLEMA DELLA PAGINAZIONE E TLB

- **Problema di Prestazioni nella Paginazione:**
 - **Ogni istruzione richiede l'accesso alla memoria** per prelevare l'istruzione stessa e un ulteriore accesso per la tabella delle pagine.
 - Raddoppio degli accessi alla memoria **riduce le prestazioni** di metà.
- **Ma ...**
 - I programmi tendono a fare **molti riferimenti a un piccolo numero di pagine**.
 - **Solo una parte limitata** delle voci della tabella delle pagine viene **utilizzata frequentemente**.
- **Introduzione del *Translation Lookaside Buffer (TLB)*:**
 - Dispositivo hardware che mappa indirizzi virtuali in fisici senza passare dalla tabella delle pagine.
 - Riduce gli accessi alla memoria durante la paginazione.



FUNZIONAMENTO E GESTIONE DEL TLB

- **Struttura:**

- **Piccolo numero** di voci (es. 8-256), ciascuna con numero di pagina virtuale, **bit modificato**, **codice di protezione**, e frame fisico.

- **Funzionamento:**

- Alla richiesta di un indirizzo virtuale, la MMU controlla prima nel TLB.
- Se trovato e valido, il frame è prelevato direttamente dal TLB.
- Se non trovato (**TLB miss**), avviene una ricerca normale nella tabella delle pagine e la voce trovata rimpiazza una voce nel TLB.

- **Gestione delle Modifiche:**

- Le modifiche ai permessi di una pagina nella tabella delle pagine richiedono l'aggiornamento del TLB.
- Per garantire la coerenza, la voce corrispondente nel TLB viene eliminata o aggiornata.

Valid	Virtual Page	Modified	Protection	Page Frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75



GESTIONE SOFTWARE DEL TLB

- **TLB in Architetture RISC:**

- Alcune macchine RISC come SPARC, MIPS, e HP PA gestiscono le voci del TLB tramite software.

- **Processo in Caso di TLB Miss:**

- Un TLB miss non porta a una ricerca automatica nella tabella delle pagine da parte della MMU.
- Invece, si genera un errore di TLB e il sistema operativo deve intervenire.
- Il sistema operativo cerca la pagina, aggiorna il TLB, e riavvia l'istruzione.



TIPOLOGIE DI MISS E IMPLICAZIONI

- **Frequenza dei TLB Miss:**

- I TLB miss sono comuni a causa del numero limitato di voci nel TLB (es. 64 voci).
- Aumentare la dimensione del TLB è costoso e richiede compromessi nella progettazione dei chip.

- **Soft Miss vs Hard Miss:**

- **Soft Miss:** La pagina è in memoria ma non nel TLB. Richiede solo l'aggiornamento del TLB.
- **Hard Miss:** La pagina non è in memoria e richiede un accesso alla memoria non volatile (disco o SSD).
 - Un hard miss è significativamente più lento di un soft miss.

- **Page Table Walk e Diverse Tipologie di Miss:**

- La ricerca nella gerarchia delle tabelle delle pagine è chiamata "**page table walk**".
- I miss possono variare in «gravità» da minori (pagina in memoria ma non nella tabella delle pagine) a maggiori (pagina da caricare dalla memoria non volatile).
- Un **accesso a un indirizzo non valido** può portare a un **segmentation fault** e alla **terminazione del programma**.



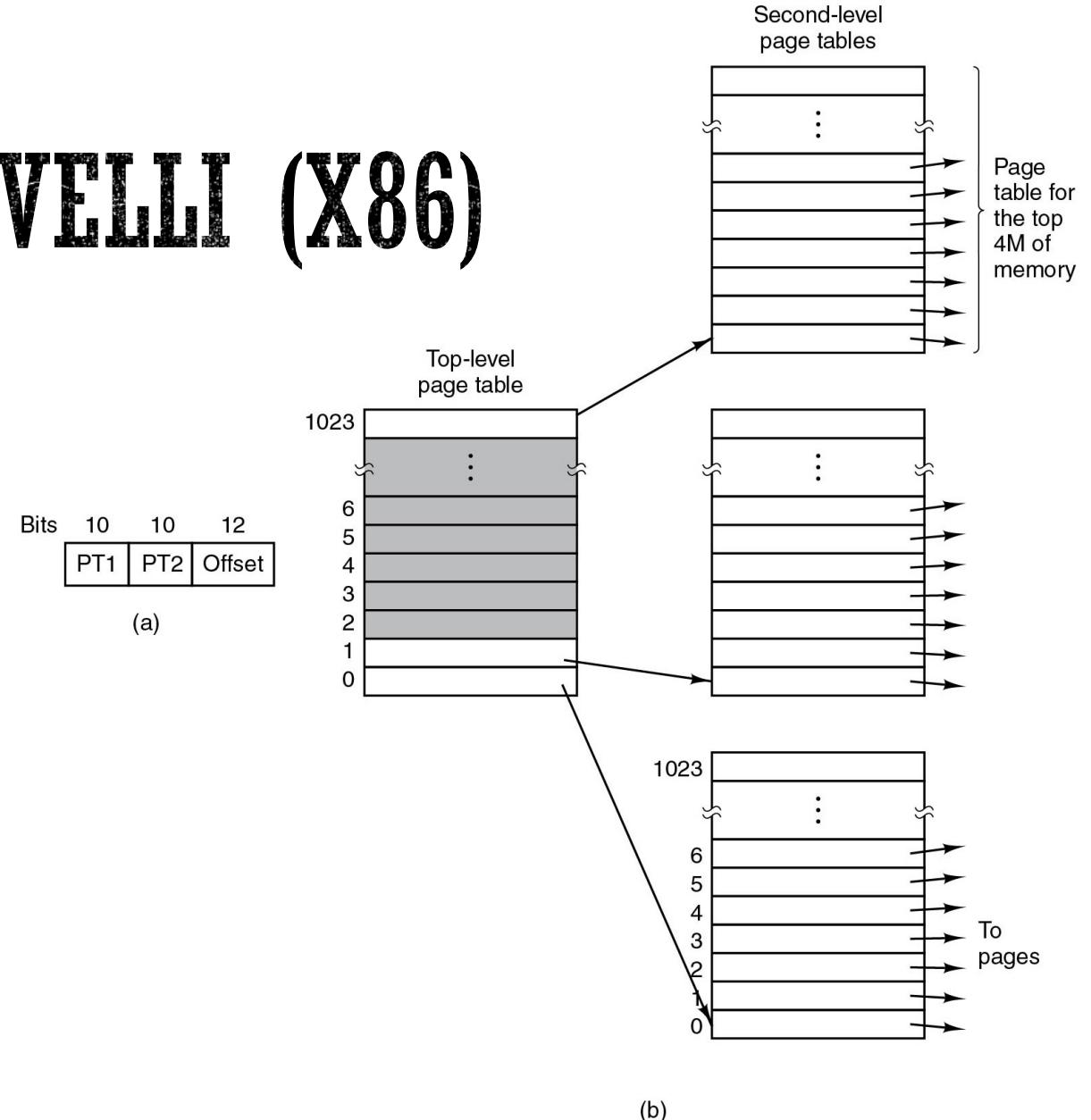
PAGE TABLE SIZES

- Poche slide fa: «*Con 32 bit e pagine da 4 KB*»:
 - 12 bit per indirizzare 4096 byte per pagina
 - tabella delle pagine di $2^{(32-12)} = 2^{20} = 1.048.576$ voci! Fattibile per PC con GB di RAM.
- Uno spazio di indirizzi virtuali molto grande porterebbe a una tabella di pagine molto grande
 - **Spreco di memoria** (senza contare cosa succederebbe per 64 bit!)
- **Possibili soluzioni**
 - Multi-level page table



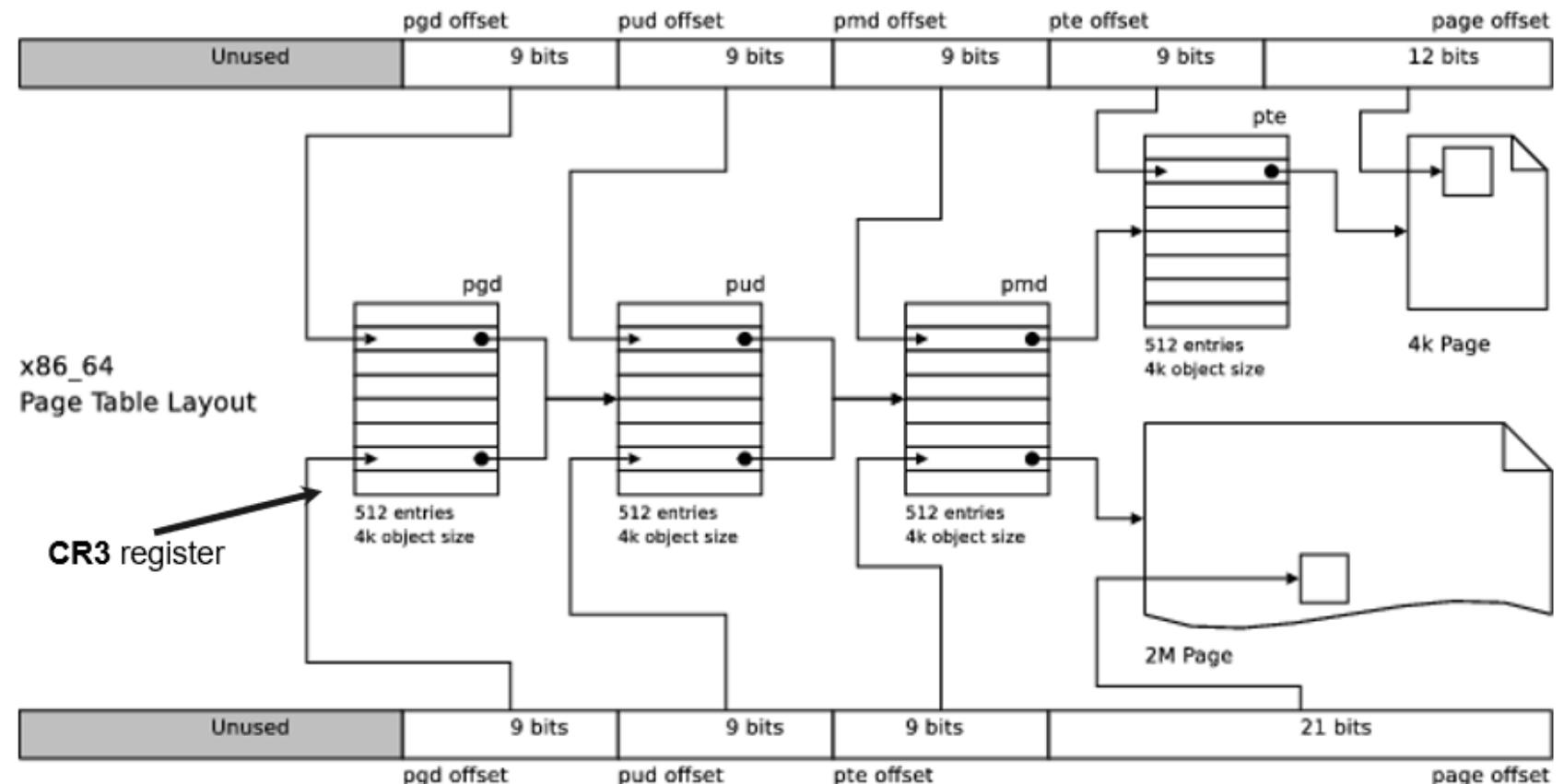
PAGE TABLE A DUE LIVELLI (X86)

- Le Page tables sono “attraversate” (“walked”) dal Memory Management Unit
- CR3 register**
 - Registro speciale per puntare al vertice della gerarchia delle tabelle di pagina
- Esempio
 - a) Un indirizzo a 32-bit con due campi
 - 10 + 10 bit
 - b) Una page table a due livelli



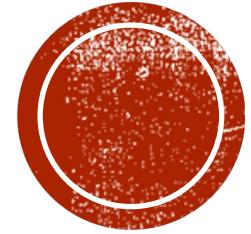
64 BIT: PAGE TABLE A 4 LIVELLI

- **PGD:** Page Global Directory
- **PUD:** Page Upper Directory
- **PMD:** Page Mid-level Directory
- **PTE:** page table entry



Nota: $2^9 \times 2^9 \times 2^9 \times 2^9 \times 2^{12} = 2^{48}$ byte. Ricordate i 48 bit?
Permettono di puntare per il momento, 256 TB di memoria





ALGORITMI DI SOSTITUZIONE DELLE PACINE

GESTIONE DELLA MEMORIA: OUTLINE

- Memory Abstraction
- Virtual Memory
- **Algoritmi di sostituzione delle pagine**
- Problemi di Progettazione per Sistemi di Paging



PAGE REPLACEMENT

- Il computer potrebbe utilizzare più memoria virtuale di quanta ne abbia di fisica.
- La paginazione crea l'illusione di una memoria praticamente illimitata a disposizione dei processi utente.
- Quando una pagina logica non è in memoria (scambiata o *swapped* con un file/partizione), il sistema operativo deve caricarla in memoria in caso di **page fault**.
- Un'altra pagina logica potrebbe essere scambiata... Ma quale?



ALGORITMI DI SOSTITUZIONE DELLE PAGINE

- Algoritmo ottimale
- Not Recently Used (NRU)
- First-In, First-Out (FIFO) algorithm
- Second-chance algorithm
- Clock algorithm
- Least recently used (LRU) algorithm
- Working set algorithm
- WS Clock algorithm



ALGORITMO DI SOSTITUZIONE DELLE PAGINE OTTIMALE

- **Concetto:** Scegliere la pagina con il riferimento più distante nel futuro da rimuovere.
- **Idealmente**, si rimuove la pagina che non sarà usata per il maggior numero di istruzioni future.
- **Esempio:** «*Se una pagina non sarà usata per 8 milioni di istruzioni e un'altra per 6 milioni, si rimuove la prima*».
- **Problema:** È impossibile per il sistema operativo prevedere il momento del prossimo riferimento per ciascuna pagina.

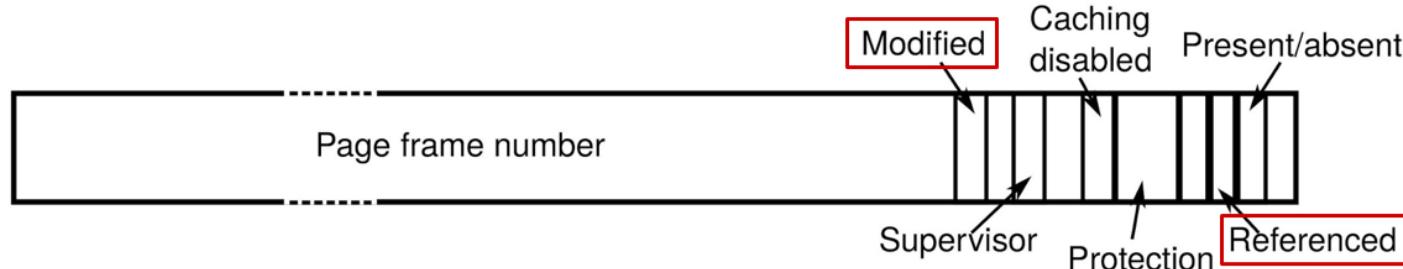


LIMITI PRATICI E VALUTAZIONE DEGLI ALGORITMI

- Il metodo ottimale **non è realizzabile** in pratica perché richiede la previsione del futuro utilizzo delle pagine.
- *Possibile* implementazione su un simulatore per valutare le prestazioni rispetto agli algoritmi reali.
- **Valutazione:** Se un sistema ha prestazioni inferiori dell'1% rispetto all'ottimale, il miglioramento massimo teorico è dell'1%.
- Gli algoritmi reali devono essere valutati per la loro applicabilità pratica, non per l'ottimalità teorica.



UN BREVE RECAP



Bit della Page Table Entry utili per gli algoritmi di sostituzione delle pagine:

- **Modified (M)**: Impostato quando una pagina viene modificata (conosciuto anche come “dirty” bit)
- **Referenced (R)**: Impostato quando la pagina viene acceduta (conosciuto anche come “accessed” bit)



CONCETTO E FUNZIONAMENTO DI NRU (NOT RECENTLY USED)

- **Obiettivo:** Trovare le pagine non modificate che non sono state accedute «recentemente».
- Vengono usati i **Bit di Stato R e M:**
 - R indica l'accesso alla pagina,
 - M segnala le modifiche.
- **Aggiornamento Hardware:** I bit vengono impostati dall'hardware a ogni accesso.
- **Reset Periodico:** Il bit R viene periodicamente ripulito per identificare pagine non recentemente usate.
 - per esempio a ogni interrupt del clock
- **Classificazione delle Pagine** in base ai bit R e M
 - le pagine sono divise in 4 classi (da 0 a 3) in funzione dell'uso e delle modifiche.



CLASSIFICAZIONE DELLE PAGINE E SCELTA DI RIMOZIONE

- **Classi di Pagine:**

- **Classe 0:** Non referenziata, non modificata.
- **Classe 1:** Non referenziata, modificata.
- **Classe 2:** Referenziata, non modificata.
- **Classe 3:** Referenziata, modificata.

- Le pagine di **classe 1** sembrano a prima vista **impossibili**, ma appaiono quando un interrupt del **clock** **azzerà il bit R** di una pagina di classe 3.

- Gli interrupt del clock non azzerano il bit M perché questa informazione è necessaria per sapere se la pagina deve essere riscritta su disco o meno.

- **Selezione per Rimozione:**

- NRU rimuove una pagina casuale dalla classe più bassa non vuota.
- Azzerare R ma non M produce una pagina di classe 1: una pagina di classe 1 è stata modificata molto tempo fa e da allora non è stata più toccata.

- **Vantaggi di NRU:** Semplicità, efficienza implementativa e prestazioni accettabili



ALGORITMO FIFO (FIRST-IN, FIRST-OUT)

- **Descrizione:** FIFO è un algoritmo di paginazione che elimina la pagina più vecchia in memoria.
- **Implementazione:** Il sistema operativo rimuove la pagina in testa alla lista (la più vecchia) durante un page fault, aggiungendo la nuova pagina in coda.
- **Problema di FIFO:** Nel contesto informatico, la pagina più vecchia potrebbe ancora essere frequentemente utilizzata, rendendo FIFO poco efficace.
- **Conclusione:** A causa di queste limitazioni, FIFO è raramente utilizzato nella sua forma più semplice.



SECONDA CHANCE - MIGLIORAMENTO DELL'ALGORITMO FIFO

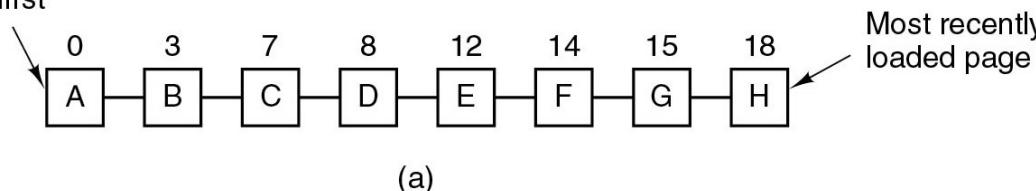
- **Principio:** Controllo del bit R (bit di lettura) della pagina più vecchia per decidere la rimozione.
- **Funzionamento:**
 - Se $R = 0$: la pagina è vecchia e non usata di recente, quindi viene sostituita.
 - Se $R = 1$: il bit R viene azzerato, la pagina è reinserita in fondo alla lista e considerata come appena caricata.

a) Pagine ordinate in ordine FIFO.

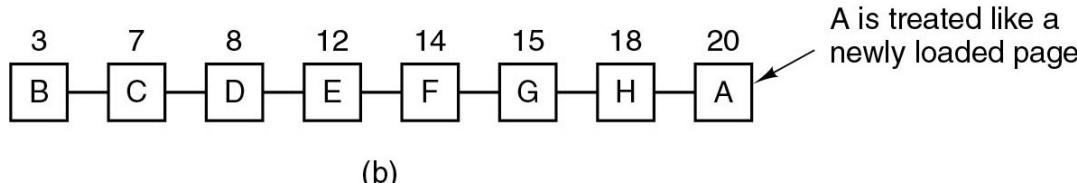
b) Elenco delle pagine se si verifica un errore di pagina al tempo 20 e A ha il bit R impostato.

I numeri sopra le pagine sono i loro tempi di caricamento.

Page loaded first



(a)



(b)

OPERATIVITÀ E CASO PEGGIORE DI SECONDA CHANCE

- **Azioni:**

- Se la pagina A ha $R = 0$, viene rimossa (scritta su memoria non volatile se modificata, altrimenti scartata).
- Se A ha $R = 1$, viene messa in fondo alla lista e il suo timestamp di caricamento aggiornato.

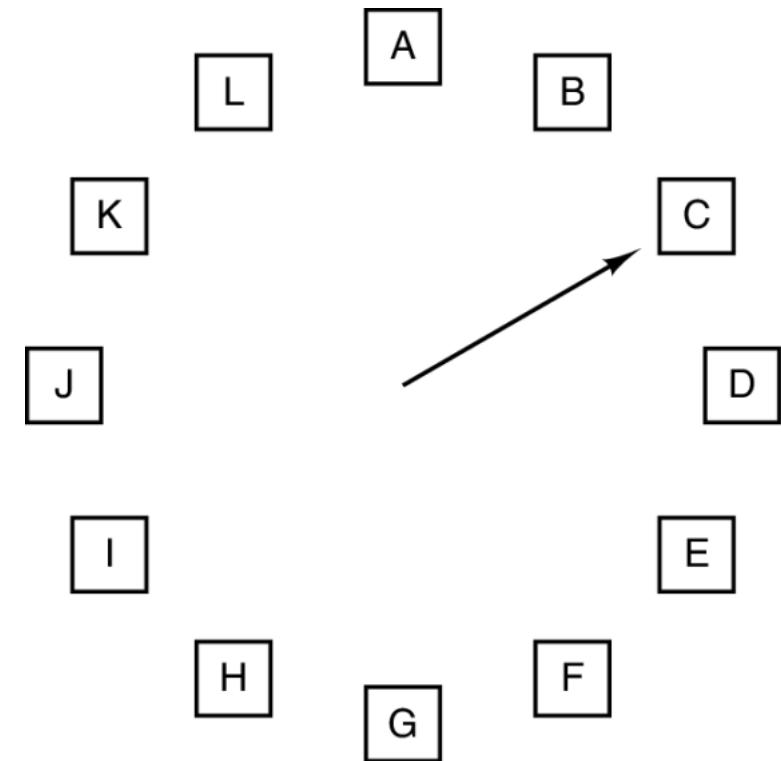
- **Scenari Possibili:**

- Se trova una pagina non referenziata, la rimuove.
- Se tutte le pagine sono state referenziate, Seconda Chance opera come FIFO puro, con un ciclo completo di reset dei bit R prima di rimuovere la pagina iniziale.



ALGORITMO DI CLOCK PER LA SOSTITUZIONE DELLE PAGINE

- **Funzionamento:** Lista circolare dei frame di pagina con un puntatore simile a una lancetta d'orologio per identificare la pagina più vecchia.
- **Page Fault:**
 - Se il bit R della pagina puntata è 0, la pagina viene rimossa e sostituita con la nuova, poi il puntatore avanza.
 - Se $R = 1$, il bit viene azzerato e il puntatore si sposta alla pagina successiva.
- **Concetto:** Ripete il processo finché non trova una pagina con $R = 0$
- **Vantaggio:** Elimina l'inefficienza della continua riallocazione delle pagine lungo la lista.
 - **Efficiente e più performante rispetto a Seconda Chance e FIFO.**



LEAST RECENTLY USED (LRU) – TRA TEORIA E «PRATICA»

■ Teoria

- **Fondamento LRU:** Pagine non usate di recente sono candidate alla sostituzione.
- **Possibile Implementazione:** Lista delle pagine con quelle più usate in testa e quelle meno usate in coda.
- **Aggiornamenti:** Ogni riferimento richiede l'aggiornamento della lista (uno *stack*) e copia di pagine intere, operazione costosa anche con hardware dedicato.
- Sebbene tendente all'ottimo, praticamente non efficiente e non utilizzato
- Esistono però altri metodi per implementare l'LRU con hardware speciale:
 - Uso di un contatore a 64 bit per ogni riferimento a memoria.
 - **Selezione LRU:** Alla generazione di un page fault, si rimuove la pagina con il contatore più basso, indicando l'uso meno recente



SIMULAZIONE SOFTWARE DI LRU - ALGORITMO NOT FREQUENTLY USED

- **NFU (Not Frequently Used)**: Associa un contatore a ogni pagina, incrementato con ogni interrupt del clock in base al bit R.
 - Tanti accessi ad una pagina => Alto valore di «frequenza» assegnato alla pagina => Minore possibilità di rimozione
- **Limite di NFU**: Non dimentica l'uso passato, può portare a scelte subottimali in ambienti multi-pass o in fase di boot
 - Esempio: una pagina utilizzata con altissima frequenza in un determinato periodo e poi «abbandonata» potrebbe non venire sostituita
- **Miglioramento di NFU => Aging**
 - Numero di bit fisso, esempio 8 bit
 - Ad ogni interrupt del clock i bit vengono spostati a destra
 - Prima dello shift dei contatori, il bit R viene aggiunto al lato sinistro.
- **Effetto dell'Aging**: Emula LRU, dando meno peso agli usi passati e preferendo le pagine meno referenziate di recente.



NFU E AGING IN AZIONE

- Simula l'LRU via software, es: **Pagina 1**
 - a) NON è modificata ed ha valore 0000000
 - b) viene modificata e diventa 10000000
 - c) viene modificata e diventa 11000000
 - d) NON è modificata e diventa 01100000
- Consideriamo le **Pagine 3 e 5**:
 - (c) Entrambe hanno avuto accesso
 - (d) e (e) Nessuna delle due ha avuto riferimenti
- Registrando un solo bit per intervallo di tempo non potremmo distinguere fra riferimenti in tempi recenti o meno
- Con NFU e aging, la **pagina 3 viene rimossa** poiché la pagina 5 ha avuto riferimenti in (a) prima e la pagina 3 no.

	R bits for pages 0-5, clock tick 0	R bits for pages 0-5, clock tick 1	R bits for pages 0-5, clock tick 2	R bits for pages 0-5, clock tick 3	R bits for pages 0-5, clock tick 4
Page	1 0 1 0 1 1	1 1 0 0 1 0	1 1 0 1 0 1	1 0 0 0 1 0	0 1 1 0 0 0
0	10000000	11000000	11100000	11110000	01110000
1	00000000	10000000	11000000	01100000	10110000
2	10000000	01000000	00100000	00010000	10001000
3	00000000	00000000	10000000	01000000	00100000
4	10000000	11000000	01100000	10110000	01011000
5	10000000	01000000	10100000	01010000	00101000

(a) (b) (c) (d) (e)



LIMITI E PRATICITÀ DELL'AGING

- **Differenze da LRU:** Aging non distingue l'ordine esatto dei riferimenti recenti e ha un orizzonte temporale limitato.
 - Non è necessariamente un male, anzi
- **Fattibilità:** 8 bit sono generalmente sufficienti per un buon compromesso tra accuratezza e uso di memoria.



IL CONCETTO DI WORKING SET

- **Definizione di Working Set:**

- Insieme delle pagine attualmente usate da un processo.
- Rappresenta la località di riferimento, ovvero le pagine a cui un processo fa riferimento durante una fase dell'esecuzione.

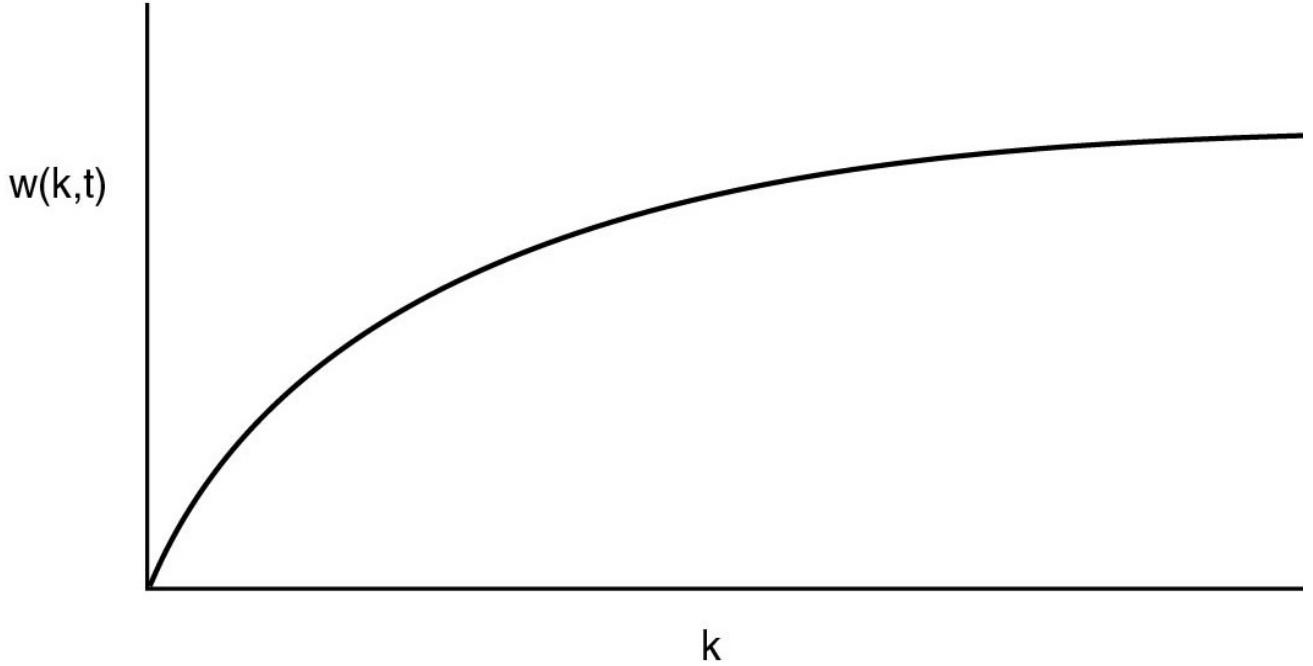
- **Demand Paging:**

- Le pagine sono caricate in memoria "on demand", solo quando necessario.
- Inizialmente, molti page fault si verificano finché non vengono caricate tutte le pagine necessarie.



CONCETTO E DINAMICA DEL WORKING SET

- **Definizione:** Working set $w(k, t)$ è l'insieme di pagine usate negli ultimi k riferimenti.
- **Monotonia:** $w(k, t)$ è monotona non decrescente al crescere di k .
- **Asintoto:** Il limite di $w(k, t)$ è finito, correlato allo spazio degli indirizzi del programma.
- **Implicazione:** C'è un ampio intervallo di k dove il working set resta invariato.



WORKING SET E PERFORMANCE

- **Gestione della Memoria e Page Fault:**

- Se il working set di un processo è completamente **in memoria**, si verificano **pochi page fault**.
- Se il working set è **più grande della memoria** disponibile, si verificano **frequenti page fault**, rallentando significativamente il processo
 - Fenomeno noto come *thrashing*.

- **Working Set Model:**

- Molti **sistemi operativi cercano di tracciare il working set** di ogni processo e di mantenerlo in memoria per ridurre i page fault.
- La **pre-paginazione carica in anticipo le pagine basandosi sul working set** del processo.



IMPLEMENTAZIONE E ALGORITMI DI SOSTITUZIONE

- **Tracciamento del Working Set:**

- Il working set è definito come l'insieme delle pagine usate negli ultimi k riferimenti alla memoria.
- **In pratica:** è spesso definito **in termini di tempo**, ad esempio, le pagine usate negli ultimi τ secondi di tempo di esecuzione.

- **Algoritmo di Sostituzione Basato sul Working Set:**

- Alla verifica di un page fault, si ricerca una pagina fuori dal working set per rimuoverla.
- Utilizza informazioni come il bit di riferimento e il tempo dell'ultimo utilizzo per determinare quali pagine rimuovere.



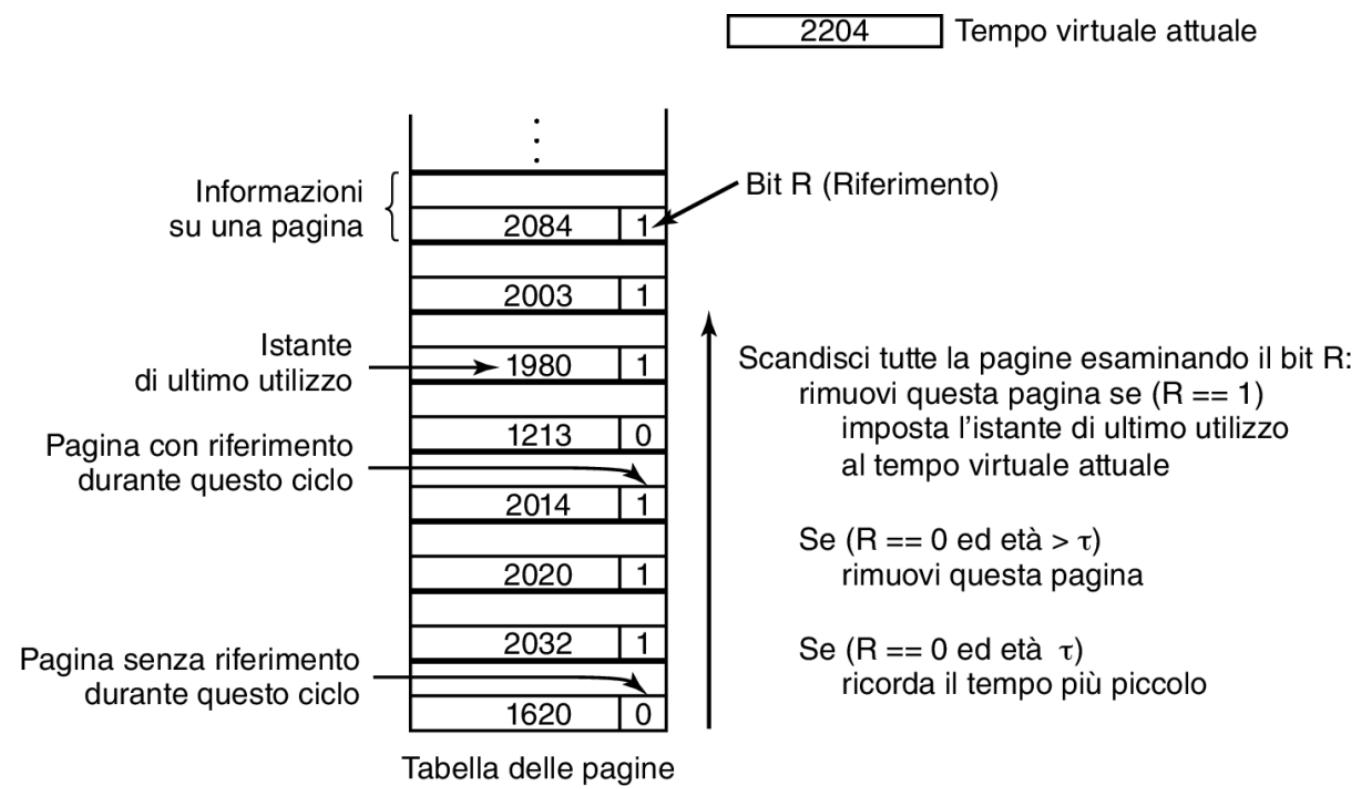
ALGORITMO WORKING SET: UN ESEMPIO

▪ Impostazione dei Bit R e M:

- Un **interrupt periodico** azzera il bit **R** a ogni ciclo di clock.

▪ Durante un Page Fault:

- Scansione delle pagine alla ricerca di una pagina da rimuovere.
- **Controllo del bit R per ogni pagina:**
 - **R = 1:** Aggiornamento del tempo dell'ultimo utilizzo, la pagina è nel working set.
 - **R = 0 e Età > τ :** La pagina non è nel working set e viene rimossa.
 - **R = 0 e Età $\leq \tau$:** La pagina rimane, ma si contrassegna la più vecchia per possibile rimozione.
- Se nessuna pagina è rimovibile, viene selezionata la più vecchia con $R = 0$
 - in caso contrario, una pagina a caso.



INTRODUZIONE ALL'ALGORITMO WSCLOCK

- **Miglioramento dell'Algoritmo Working Set:**

- WSClock è un'evoluzione dell'algoritmo Clock che integra informazioni del working set.
- Popolare per la sua semplicità e buone prestazioni.

- **Struttura Dati:**

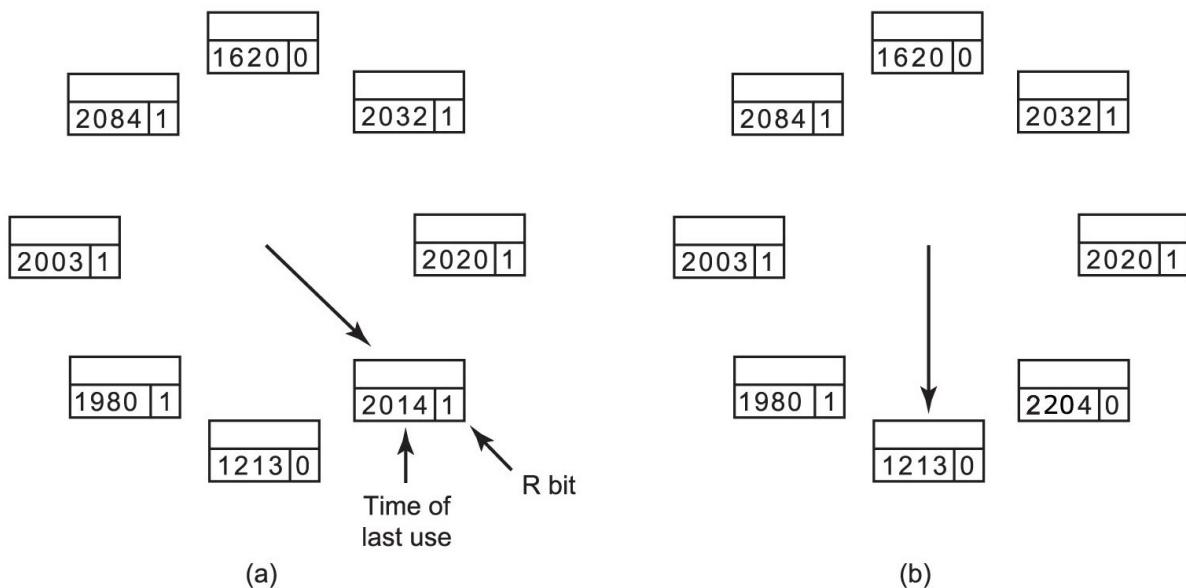
- Usa una lista circolare di frame, simile all'algoritmo Clock.
- Ogni frame nella lista contiene
 - il tempo dell'ultimo utilizzo
 - il bit R (Riferimento)
 - il bit M (Modificato)



WSCLOCK: UN ESEMPIO

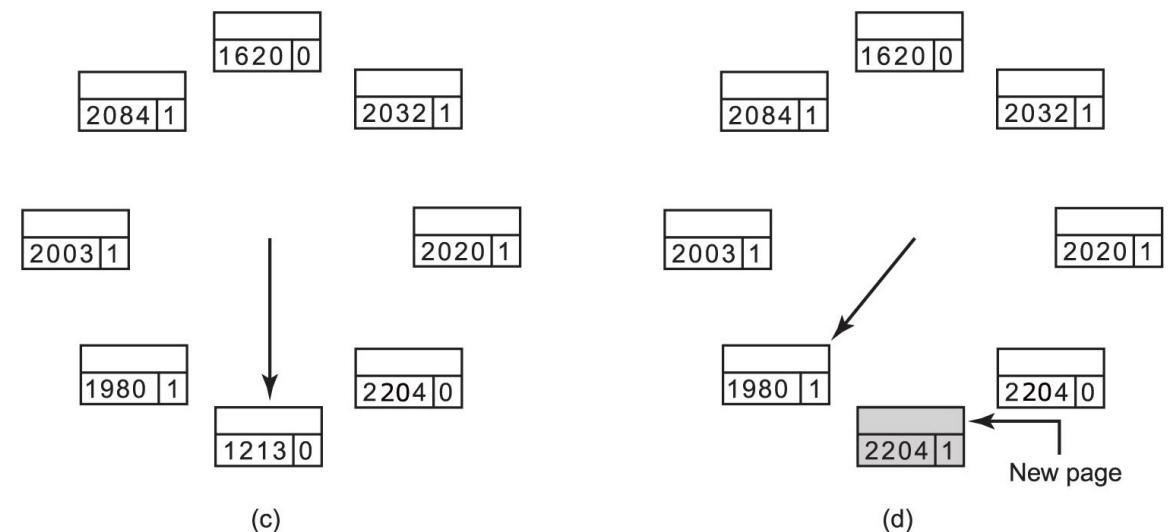
- Ad ogni page fault è esaminata per prima la pagina indicata dalla lancetta dell'orologio.
- Se il **bit R =1**, la pagina **NON** è la **candidata ideale** alla rimozione
 - è stata usata nel ciclo del clock
- Il bit R viene quindi impostato a 0
- La lancetta avanza alla pagina successiva e l'algoritmo viene ripetuto per la nuova pagina.
 - La situazione dopo questa sequenza è mostrata nella Figura (b)

2204 Current virtual time



WSCLOCK: UN ESEMPIO (2)

- Se la pagina indicata ha $R = 0$ (c) e se l'età è maggiore di τ
 - Se $M = 0$ (pagina pulita)
 - Non è nel set di lavoro e ne esiste una copia valida su memoria non volatile
 - Il frame viene semplicemente riciclato e vi viene posta la nuova pagina (d)
 - Se $M = 1$ invece la pagina è «sporca» (ovvero modificata)
 - non ne esiste una copia valida in memoria non volatile
 - non può essere sfrattata immediatamente.
- Per evitare «rallentamenti» (come un cambio di processo) la scrittura su memoria non volatile viene schedulata e rimandata
 - lungo la lista potrebbe esserci una pagina pulita e vecchia che può essere usata immediatamente
 - la lancetta avanza e l'algoritmo procede con la pagina successiva



GESTIONE SCRITTURE E SELEZIONE PAGINA IN WSCLOCK

- **Limitazione Scritture su Memoria Non Volatile:**
 - Possibilità di schedulare tutte le pagine per I/O su memoria non volatile in un ciclo di clock.
 - Per ridurre il traffico su disco/SSD, si imposta un limite massimo di scritture (n pagine).
 - Una volta raggiunto il limite n , ulteriori scritture non vengono schedulate.
- **Comportamento al Completamento del Giro di Orologio:**
 - **Quando ci Sono Scritture Pendenti**
 - La lancetta prosegue il suo giro **cercando pagine "pulite"** (non modificate).
 - **Non appena una** scrittura pendente **viene completata**, la pagina associata diventa "pulita".
 - La lancetta **seleziona la prima pagina pulita che incontra** e la rimuove dalla memoria.
 - **Quando NON ci Sono Scritture Pendenti**
 - Significa che tutte le pagine sono attivamente utilizzate ("nel set di lavoro").
 - La strategia diventa quella di **scegliere e rimuovere una pagina pulita a caso**.
 - Se non ci sono pagine pulite disponibili, la pagina corrente viene scelta per la rimozione e la sua copia viene scritta su disco.



ALGORITMI DI SOSTITUZIONE DELLE PAGINE: RIEPILOGO

Algoritmo	Commento
Ottimale	Non implementabile, ma utile come termine di confronto e valutazione
LRU (Last Recently Used)	Eccellente, ma difficile da implementare con precisione
NRU (Not Recently Used)	Approssimazione molto rozza dell'LRU
FIFO (First-In, First Out)	Potrebbe eliminare pagine importanti
Seconda chance	Deciso miglioramento rispetto al FIFO
Clock	Realistico
NFU (Non Frequently Used)	Approssimazione abbastanza rozza dell'LRU
Aging	Algoritmo efficiente che approssima bene l'LRU
Working set	Piuttosto dispendioso da implementare
WSClock	Algoritmo efficiente e buono



ALGORITMI DI SOSTITUZIONE DELLE PAGINE: RIEPILOGO (2)

- **Algoritmi Preferiti:**

- Aging e WSClock sono i «migliori» tra gli algoritmi analizzati.
- Entrambi basati rispettivamente su LRU e sull'idea di Working Set, con buone prestazioni e implementazione efficiente.
- La nozione di «migliore» è il risultato del trade-off tra la complessità del metodo e i vincoli hardware che il sistema operativo deve comunque rispettare.

- **Implementazioni nei Sistemi Operativi:**

- Sistemi come Windows e Linux adottano varianti di questi algoritmi, a volte combinando diversi elementi per ottimizzare le prestazioni in base a specifiche esigenze e al tipo di hardware.

