

Memoria Virtuale qualche dettaglio

Bit di validità

0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H

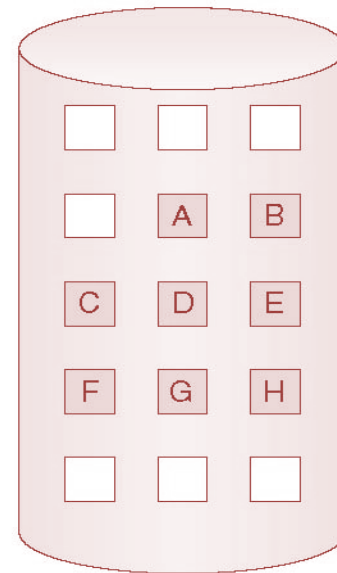
logical
memory

valid-invalid bit	
frame	bit
0	4 v
1	i
2	6 v
3	i
4	i
5	9 v
6	i
7	i

page table

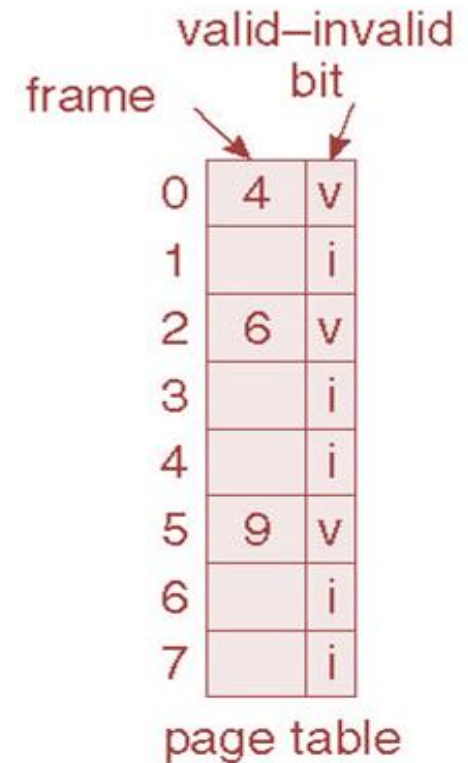
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

physical memory



Page Fault

- Quando il processo **richiede un indirizzo**:
 - Se tenta di accedere a un indirizzo con **bit di validità 1**, procede come nella paginazione normale, la pagina c'è e l'indirizzo logico viene convertito in indirizzo fisico.
 - Se tenta di accedere a un indirizzo con **bit di validità 0**, viene generato un trap denominato **page fault** che può essere dovuto ad un errore nella gestione degli indirizzi oppure alla necessità di caricare in memoria la pagina mancante.



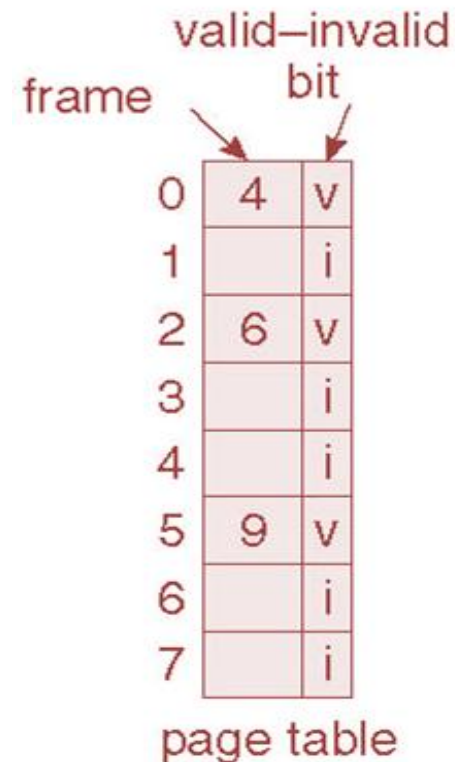
The diagram shows a page table with 8 rows, indexed 0 to 7. Each row has two columns: 'frame' and 'valid-invalid bit'. Arrows point from the column headers to their respective columns. The 'frame' column contains the values 4, 6, and 9 for rows 0, 2, and 5 respectively, and is empty for rows 1, 3, 4, 6, and 7. The 'valid-invalid bit' column contains 'v' for rows 0, 2, and 5, and 'i' for rows 1, 3, 4, 6, and 7. The entire table is labeled 'page table' at the bottom.

	frame	valid-invalid bit
0	4	v
1		i
2	6	v
3		i
4		i
5	9	v
6		i
7		i

page table

Page Fault

- Riassumendo:
 - Bit di validità 1 (v), pagina in memoria
 - Bit di validità 0 (i), pagina NON in memoria
 - Inizialmente il valore del bit di validità è 0 (i) per tutte le pagine: nessuna è caricata
- Durante la traduzione degli indirizzi effettuata dalla MMU se il bit di validità è 0 (i) viene generato un page fault, che come tutti i trap interrompe l'esecuzione del processo e produce l'esecuzione del gestore delle interruzioni.



The diagram shows a page table with 8 entries (frames 0 to 7). Each entry consists of a frame number and a valid-invalid bit. The valid-invalid bit is labeled 'v' for valid and 'i' for invalid. The frame numbers are labeled 'frame' and the valid-invalid bit is labeled 'valid-invalid bit'. The page table is labeled 'page table' at the bottom.

frame	valid-invalid bit
0	4 v
1	i
2	6 v
3	i
4	i
5	9 v
6	i
7	i

Gestione del Page Fault

- La procedura di gestione del **page fault** (avviata come gestore delle interruzioni) è la seguente:
 1. Viene controllata la tabella interna del processo per stabilire se l'indirizzo logico è valido.
 2. Se il riferimento non era valido il processo viene terminato. Altrimenti se il riferimento è valido viene iniziato il trasferimento della pagina.

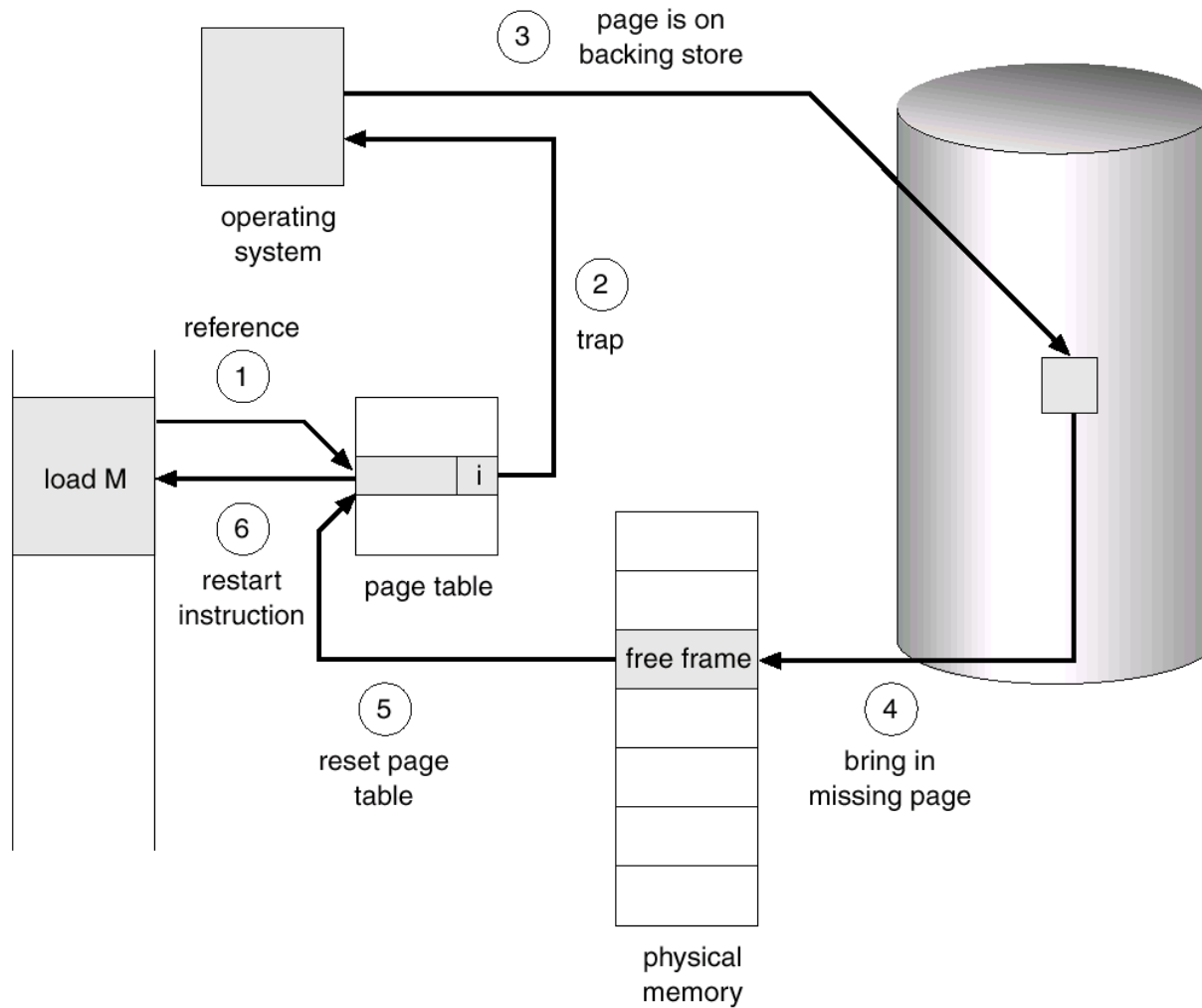
Gestione del Page Fault (segue)

3. Viene individuato un frame libero.
4. Viene caricata la pagina mancante, dal disco al frame scelto.
5. Viene modificata la tabella interna del processo e la tabella generale delle pagine.
6. Viene riavviata l'esecuzione interrotta con il trap di page fault. A questo punto il programma può accedere all'indirizzo che è in memoria centrale.

On demand puro

- È possibile avviare l'esecuzione di un processo senza pagine in memoria:
 - Quando il SO carica nel program counter l'indirizzo della prima istruzione, genera un primo page fault
 - Una volta portata la prima pagina in memoria, le altre vengono caricate quando sono indirizzate perché contengono dati o codice.
- Questo schema di esecuzione è detta **paginazione su richiesta pura**

Page Fault



Hardware

- L'**hardware** richiesto per la gestione della memoria virtuale tramite demand paging è dunque sostanzialmente lo stesso necessario per la **paginazione** e lo **swapping**:
 - **Tabella delle pagine**, per il reperimento delle pagine, a cui è aggiunto il bit di validità.
 - **Backing store**, realizzato su supporti di memoria secondaria, per la memorizzazione stabile delle pagine
 - **Apposita MMU**.

Page Fault

- Il **page fault** (come tutti i trap e gli interrupt):
 - Interrompe l'esecuzione del processo;
 - Induce il salvataggio dello stato del processo (registri, program counter, ecc.)
 - Avvia la routine di gestione del trap.
- Al termine dell'esecuzione della routine di gestione il processo può essere riavviato esattamente dalla istruzione macchina in cui si era interrotto, con la differenza che la locazione richiesta (o meglio la pagina in cui essa è contenuta) è stata caricata in memoria.

Accessi

- Un programma può accedere a più locazioni di memoria con una sola istruzione, ovvero può provocare più di un page fault (e conseguentemente più di una interruzione) per ogni istruzione.
- Questo fenomeno è in realtà molto limitato grazie al fatto che i programmi tendono ad avere **località di riferimento**, ovvero ad allocare in posizioni vicine, riferimenti “vicini”.

Prestazioni

- La paginazione su richiesta può avere un effetto notevole sulle **prestazioni** di un SO:
 - Se la pagina è in memoria il tempo di accesso corrisponde al tempo di accesso alla memoria.
 - Se la pagina non è in memoria e si verifica il page fault, il tempo di accesso include il tempo di caricamento della pagina in memoria e il tempo di accesso alla memoria.

Prestazioni

- Supponiamo che **p** sia la probabilità di avere un page fault. Ovviamente $0 \leq p \leq 1$.
- Indichiamo con **ma** il tempo di accesso in memoria e con **pf** il tempo di accesso con page fault.
- Allora il tempo di accesso effettivo medio sarà:
$$(1-p) \times \mathbf{ma} + p \times \mathbf{pf}$$
- Il tempo di accesso effettivo dipende dal tempo di gestione del page fault e dalla frequenza dei page fault.

Eventi scatenati dal page fault (1/3)

- La sequenza di azioni che vengono effettuate in seguito al page fault è la seguente:
 1. Viene generato un trap
 2. Vengono salvati i registri e lo stato
 3. Viene determinata la natura del fault
 4. Viene controllata la validità del riferimento
 -

Eventi scatenati dal page fault (2/3)

5. Viene caricata la pagina sul frame libero:
 - Attesa che la richiesta di accesso al disco sia servita (scheduling del disco)
 - Attesa del posizionamento e/o latenza
 - Inizio trasferimento della pagina sul frame.
6. Durante l'attesa la CPU può essere rischedulata ad un altro processo
7. Interrupt per l'I/O del disco completato
8. Salvataggio dei registri dell'altro processo in esecuzione

Eventi scatenati dal page fault (3/3)

9. Determinazione della natura dell'interrupt
10. Correzione della tabella delle pagine
11. Attesa che la CPU venga di nuovo allocata a questo processo
12. Recupero dello stato del processo e dei suoi registri e ripresa dell'esecuzione interrotta.

Morale

- In un sistema con memoria virtuale basata su paginazione su richiesta è importante **tenere basso il numero dei page fault** altrimenti il tempo effettivo di accesso aumenta a dismisura e si rallenta notevolmente l'esecuzione dei processi.
- E' cruciale **scegliere bene il meccanismo di sostituzione** delle pagine!

Prestazioni

- la paginazione su richiesta ha anche **effetti positivi**:
 - Se un processo utilizza solo una parte delle pagine, la paginazione su richiesta consente di eliminare i tempi di caricamento delle pagine che non verranno utilizzate.
 - Questo significa poter aumentare il grado di multiprogrammazione e quindi anche l'utilizzo e il throughput della CPU.

Copiatura su scrittura

- La chiamata alla fork genera un processo figlio che usa una copia dello spazio degli indirizzi del processo genitore, generato duplicando le pagine del processo genitore.
- Solitamente però lo spazio degli indirizzi viene sostituito con una exec e dunque la generazione di un processo figlio produce a stretto giro un page fault e rende inutile la copiatura fatta in precedenza.

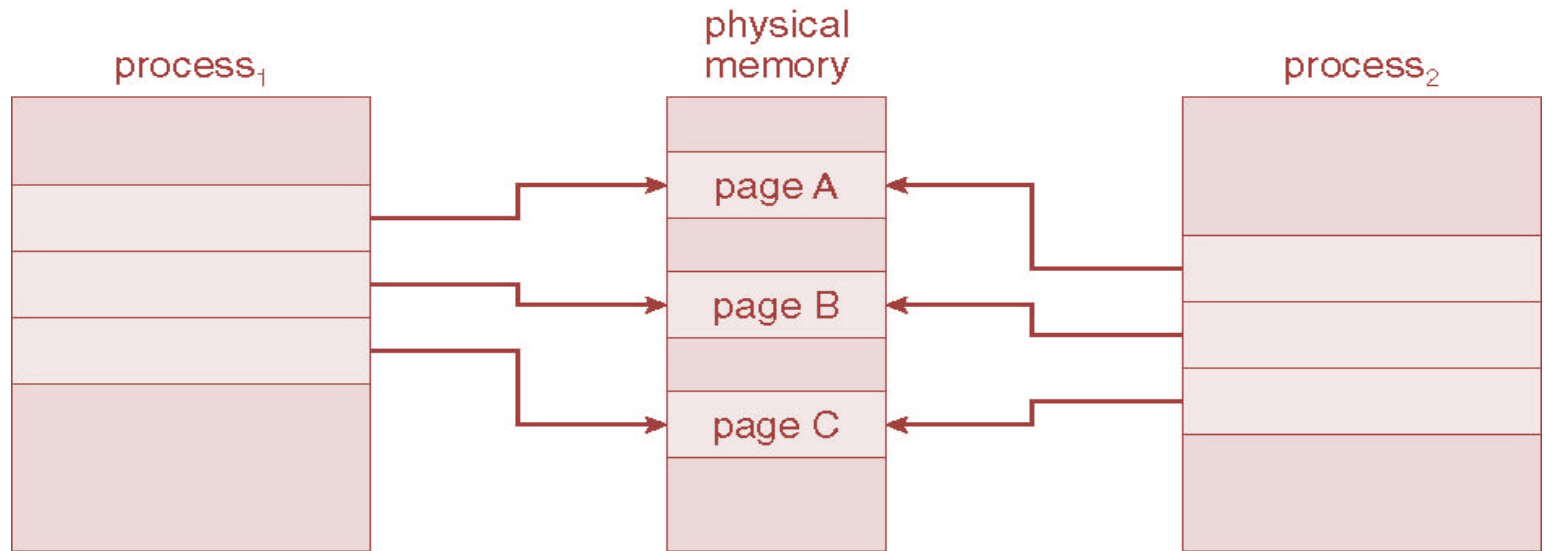
Copiatura su scrittura

- Per evitare questo fenomeno si utilizza una tecnica detta **copiatura su scrittura** (copy on write) basata sulla iniziale condivisione delle pagine da parte del processo padre e del processo figlio.
- Le pagine condivise vengono marcate come pagine da copiare (duplicare) in caso di scrittura, ad indicare che se/quando uno dei due processi tenta una scrittura, allora i due spazi degli indirizzi si devono separare ovvero il SO deve fare una copia della pagina per ciascuno dei due processi e modificare quella opportuna.

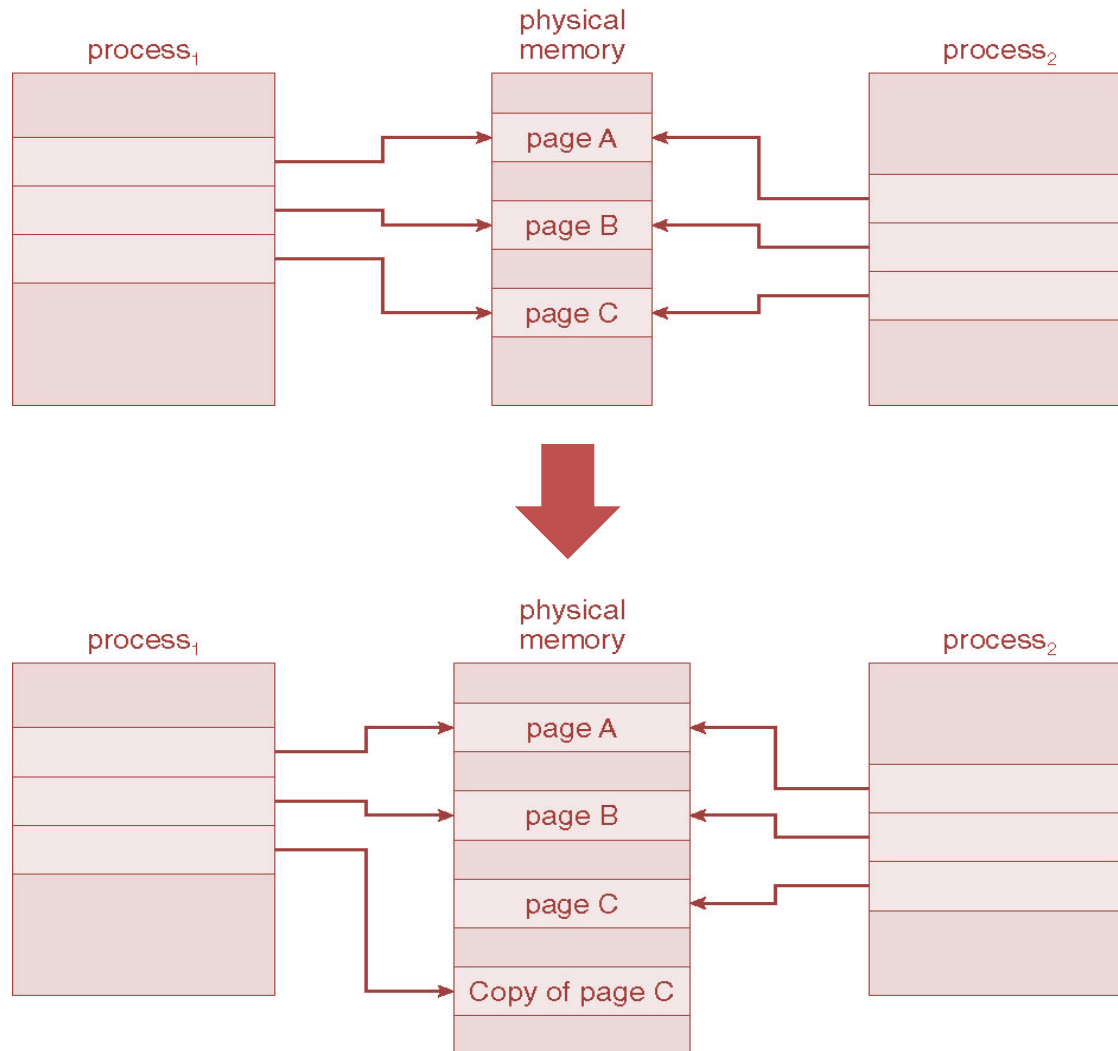
Copiatura su scrittura

- Per esempio se il processo figlio tenta di modificare una pagina dello stack, il SO considera la pagina (marchiata come copy on write) da copiare nello spazio degli indirizzi del processo figlio.
- Il processo figlio modifica la sua copia della pagina e non quella del processo padre
- È importante capire dove recuperare la pagina libera necessaria per il copy on write; di solito **si usano pagine prelevate da un pool** che il SO tiene libere per questo tipo di esigenze

Copiatura su scrittura



Copiatura su scrittura

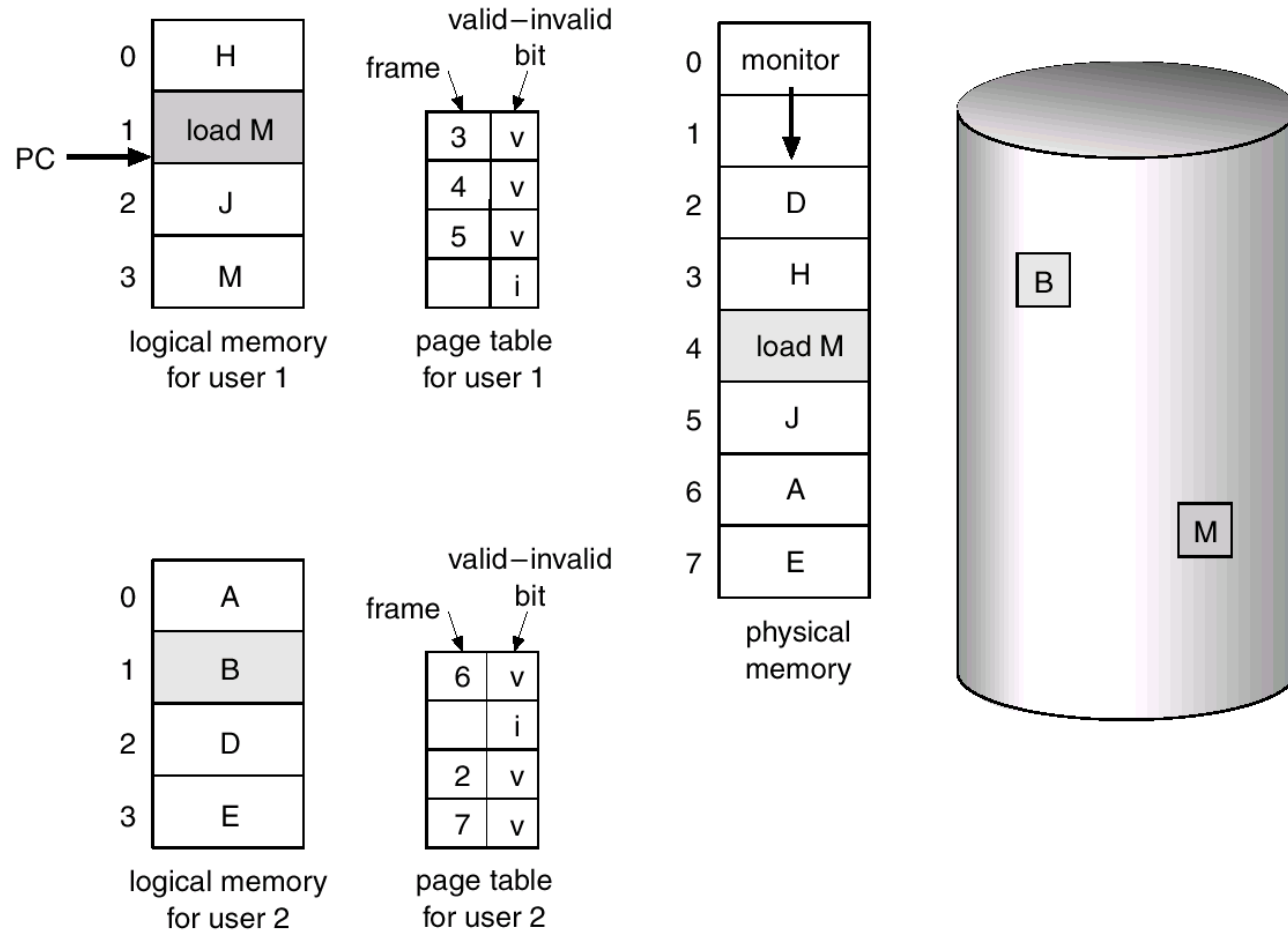


Sostituzione delle pagine

- Se nessun frame è libero è possibile liberarne uno occupato scaricando il suo contenuto dalla memoria centrale alla backing store.
- La **sostituzione delle pagine** (una è stata scaricata – **swap out** - e una è stata caricata – **swap in** -) deve poi essere registrata nelle tabella delle pagine.

Rimpiazzamento delle Pagine

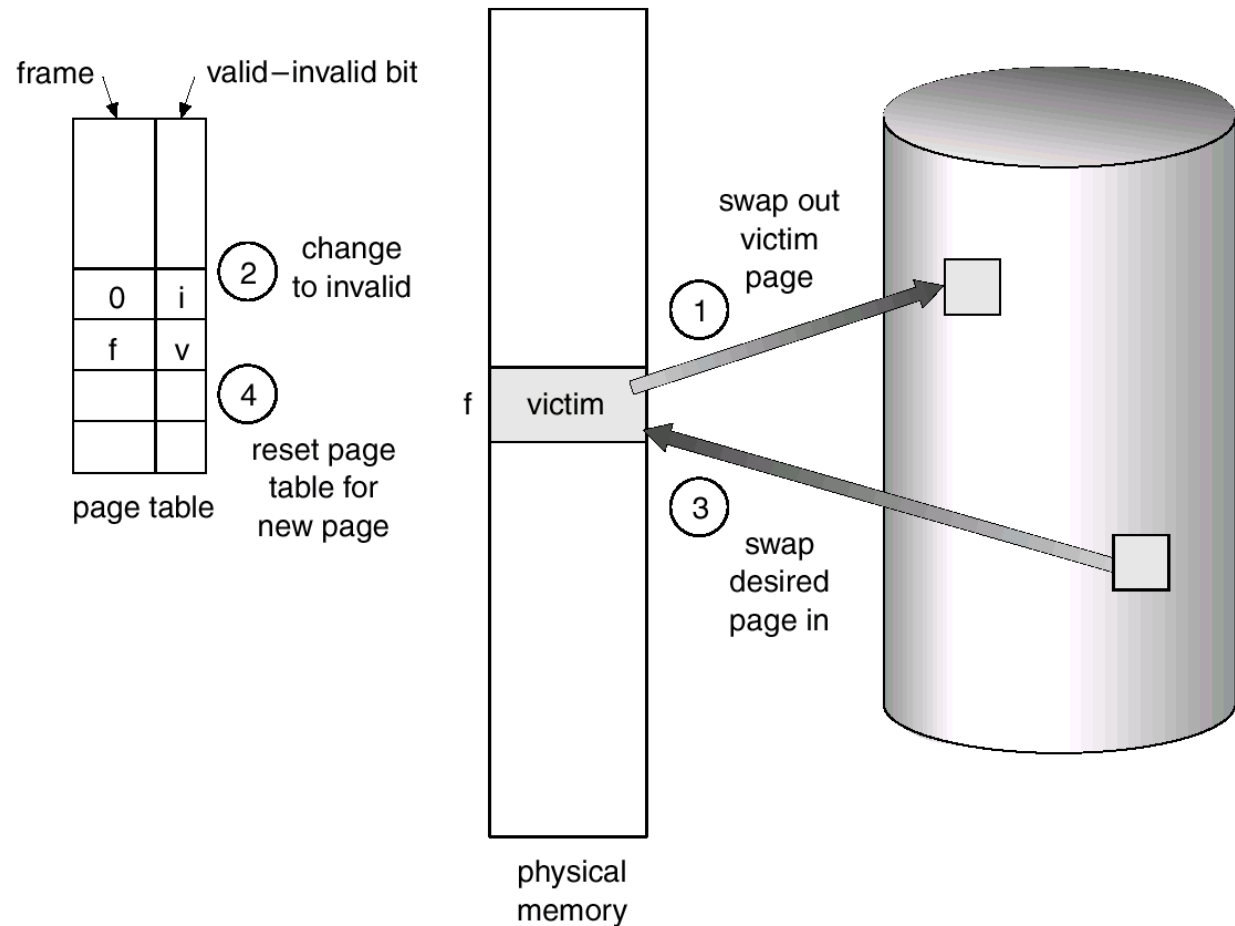
Page Replacement



Sostituzione delle pagine

- La sostituzione delle pagine avviene quindi nel modo seguente:
 1. Individuazione della locazione richiesta
 2. Individuazione del frame libero
 - a. Se esiste un frame libero viene utilizzato.
 - b. Altrimenti viene selezionata una **vittima** utilizzando un apposito algoritmo
 - c. La vittima viene scaricata (cioè scritta su disco) e vengono opportunamente aggiornate le tabelle
 3. La pagina richiesta viene carica e vengono opportunamente aggiornate le tabelle
 4. Viene riavviato il processo utente.

Page Replacement



Dirty bit

- Lo scaricamento può essere evitato se la pagina da scaricare non è stata modificata.
- Per verificare questa condizione occorre aggiungere alla pagina un **bit di modifica (dirty bit)** che viene gestito via hardware:
 - Quando la pagina è caricata viene messo a 0
 - Quando la pagina è scritta viene messo a 1
 - Quando la pagina è scaricata viene fatta la scrittura su disco solo se il bit di modifica è a 1.

Paginazione a richiesta

- Il meccanismo di sostituzione illustrato sino ad ora necessita della definizione di due componenti fondamentali:
 - L'algoritmo per l'**allocazione dei frame** che decide quanti frame devono essere allocati a ciascun processo.
 - L'algoritmo per la **sostituzione delle pagine** che decide quale pagina debba essere scaricata se non ci sono frame liberi.

Sostituzione delle pagine

- Il numero di page fault è influenzato fortemente dai ricaricamenti ovvero dalla scelta di fare swap out di una pagina che servirà entro breve e dovrà pertanto rifare swap in.
- Confrontiamo i diversi algoritmi di sostituzione delle pagine per capire quanti fault generano.
- Consideriamo generalmente un algoritmo tanto più buono quanti meno page fault genera (perché miriamo a minimizzare il tempo effettivo di accesso).

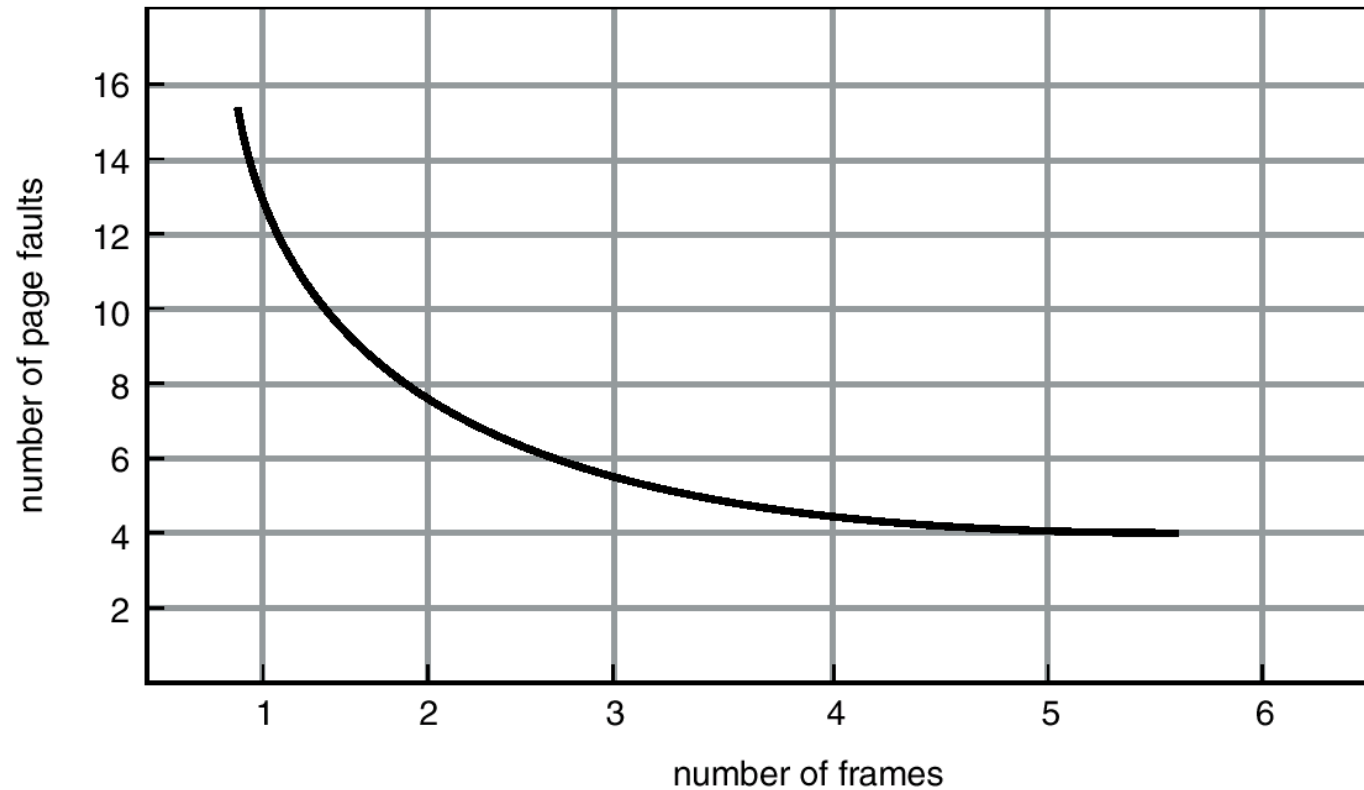
Sostituzione delle pagine

- Utilizzeremo una stessa sequenza tipo di accessi alla memoria (stringa dei riferimenti) per comparare le prestazioni dei diversi algoritmi su una memoria con 3 frame:

7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1

- **Nota bene:** In generale il numero dei page fault tenderebbe a diminuire aumentando il numero dei frame.

Page fault vs frame number



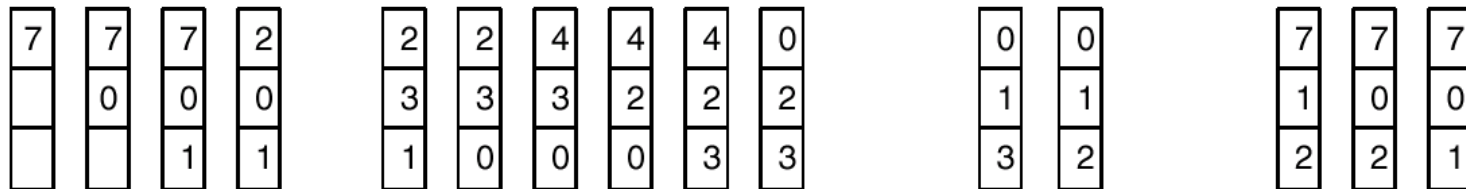
FIFO

- Un primo approccio alla scelta della pagina da scaricare è quello di usare un algoritmo di tipo **First In First Out (FIFO)**.
- L'algoritmo associa ad ogni pagina l'istante in cui è stata caricata in memoria e, se c'è necessità di scaricare una pagina, sceglie la pagina più vecchia, ovvero quella che ha l'istante di caricamento più basso.

FIFO

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



page frames

Page Fault

.....

NON Page Fault

FIFO

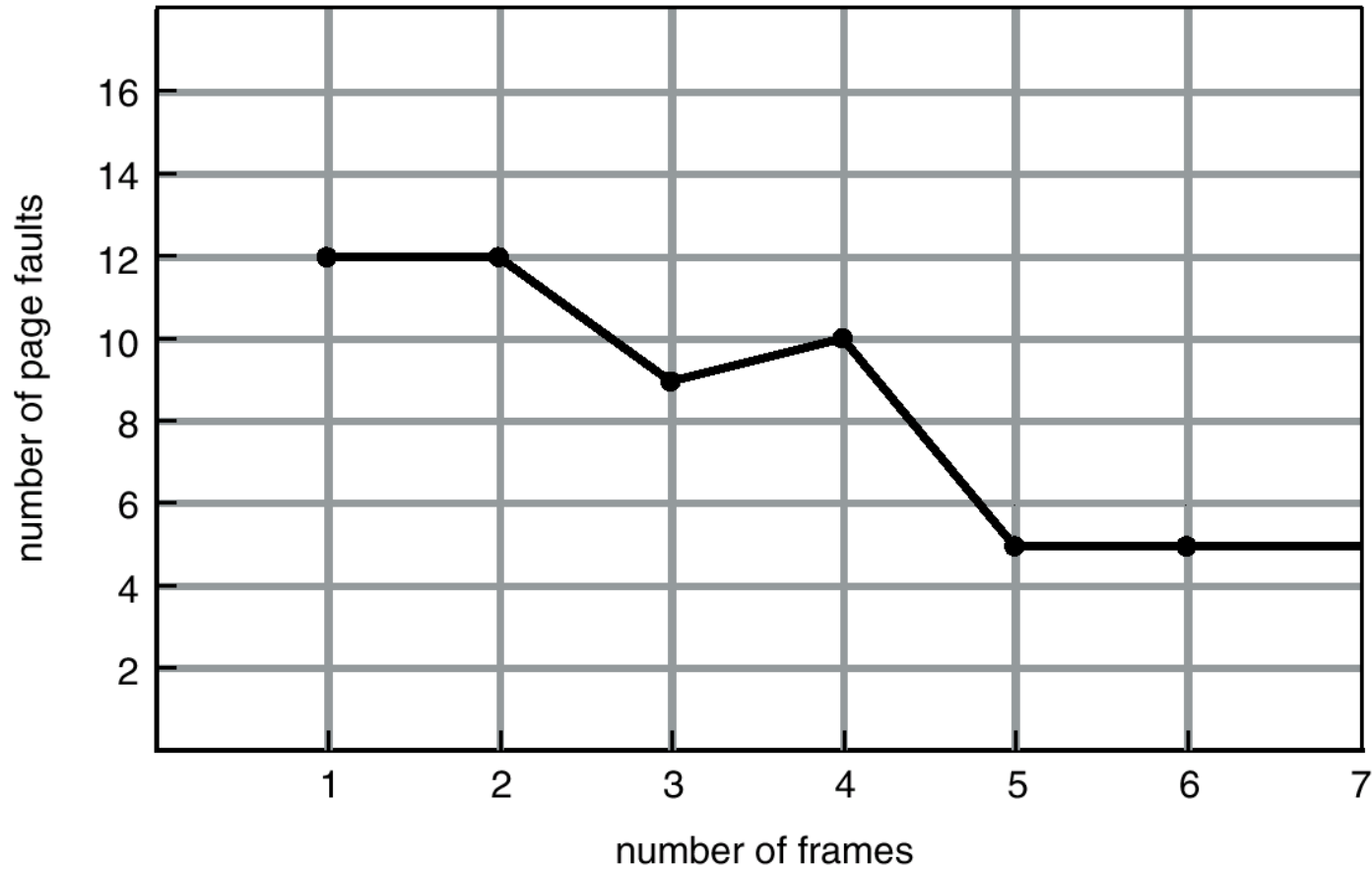
- L'algoritmo di tipo **FIFO** è facile da implementare ma fornisce prestazioni piuttosto basse:
 - La pagina sostituita potrebbe essere un modulo di inizializzazione caricato dal processo all'inizio e da tempo inutilizzato, e allora ci va bene.
 - Potrebbe però anche essere una variabile globale allocata dal processo all'inizio e ancora molto in uso, e allora la scelta è sfortunata.

FIFO

- Se la scelta della pagina da sostituire è errata aumenta la frequenza del page fault.
- Con la nostra sequenza abbiamo avuto 15 page fault.
- **Anomalia di Belady:** l'algoritmo FIFO può avere prestazioni peggiori aumentando il numero di frame.
- Provare a casa, ad esempio, la stringa di riferimenti seguente (su 3 e 4 frame):

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

Anomalia di Belady



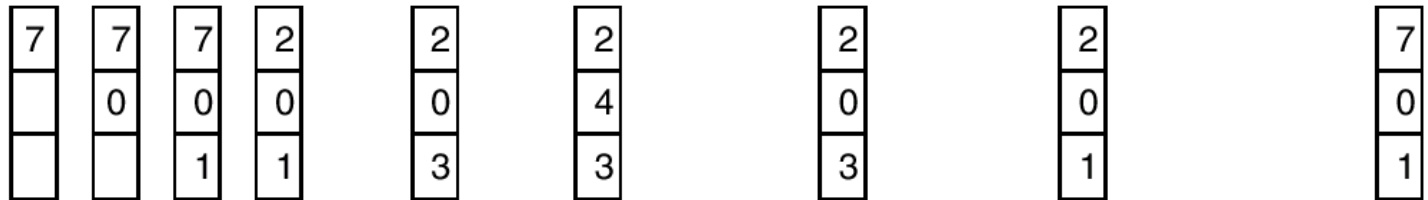
Algoritmo ottimale (non implementabile)

- Potendo conoscere la sequenza completa dei riferimenti richiesti, si può scrivere un **algoritmo ottimale** (o **algoritmo minimo**) che renda minimo il numero di page fault che avvengono complessivamente.
- L'algoritmo funziona scaricando la pagina che non verrà utilizzata per il periodo di tempo più lungo (ma si parla del futuro).

Algoritmo ottimale

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



page frames

Algoritmo ottimale

- L'**algoritmo ottimale** non soffre dell'anomalia di Belady.
- Rispetto all'algoritmo FIFO abbiamo ottenuto (sulla stringa di riferimenti di prova, con 3 frame) un miglioramento notevole: 9 page fault al posto di 15.
- E' però **impossibile da implementare** perché richiede conoscenza sui riferimenti richiesti nel futuro (che sono imprevedibili).
- Viene utilizzato come **lower bound**: meglio di così non si può fare.

Least Recently Used (LRU)

(usata meno recentemente)

- Usiamo come approssimazione di un futuro vicino (quale pagina dilazionerà la generazione del page fault) un passato recente (quale pagina è inutilizzata da più tempo).
- L'algoritmo **Least Recently Used (LRU)** scarica la pagina che non viene acceduta da più tempo, supponendo che se non la si usa da un po', probabilmente non la si userà per un altro po'.

LRU

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2					2		4	4	4	0					1		1		1		
	0	0	0					0		0	0	3	3					3		0		0		
		1	1					3		3	2	2	2					2		2		7		

page frames

LRU

- **LRU** rappresenta l'algoritmo ottimale tra quelli con analisi dei dati a ritroso.
- Rispetto all'algoritmo ottimale, ci sono casi in cui “sbaglia”, quando una pagina richiesta molto tempo fa viene scaricata ma sta in realtà per essere richiesta di nuovo.
- Non è soggetto all'anomalia di Belady.

LRU

- LRU rappresenta una buona politica di sostituzione (molto utilizzata) ma è costoso da implementare.
- Il problema di determinare l'ordine in cui rimuovere le pagine può essere risolto sostanzialmente in due modi:
 - Con **contatori**: la tabella delle pagine non viene ordinata.
 - Con uno **stack**: la tabella delle pagine viene ordinata.
 - Entrambi i modi necessitano, ad ogni accesso ad una pagina, di modificare qualcosa nella tabella delle pagine.
 - Richiederebbero hardware apposito, troppo costoso.