3 – Gestione e organizzazione della memoria

Organizzazione e gestione della memoria
Gerarchie di memoria
Allocazione di memoria contigua e non contigua
Mono utente
Multiprogrammazione con partizioni fisse e con partizioni variabili
Swapping di memoria

Memoria Virtuale
- **paginazione**
- **località**

Mapping dei blocchi
Paginazione

- **Traduzione di indirizzi di Paginazione:**
  - **Mapping diretto**
  - **Mapping associativo**
  - **Mapping combinato**
  - **Tabelle di pagine multilivello**
  - **Tabelle inversa delle pagine**
  - **Condivisione nei sistemi di paginazione**

3 – Gestione e organizzazione della memoria

Paginazione: algoritmi di sostituzione di pagina
- **Random**
- **First-In-First-Out (FIFO) e anomalie**
- **Least-Recently-Used (LRU)**
- **Least-Frequently-Used (LFU)**
- **Not-Recently-Used (NRU)**
- **Modifica a FIFO: Seconda-Chance e pagina clock**
- **Pagina lontana**

Modello Working Set

Sostituzione di pagina con Page-Fault-Frequency (PFF)

Problemi di progettazione di sistemi con Paginazione
- **Dimensione della pagina**
- **Comportamento dei programmi con paginazione**
- **Sostituzione di pagina globale vs. locale**

3 – Gestione e organizzazione della memoria

Segmentazione

- **Traduzione di indirizzi**
- **Condivisione nei sistemi con segmentazione**
- **Protezione e controllo degli accessi**

Sistemi con Segmentazione/Paginazione

- **Traduzione dinamica degli indirizzi**
- **Condivisione**
- **Protezione e controllo degli accessi**

Obbiettivi

- Memoria Virtuale vantaggi e svantaggi della paginazione a previsione e a richiesta
- sistemi di memoria virtuale di
  - con paginazione
  - con segmentazione
  - combinato segmentazione /paginazione

- problemi nella sostituzione delle pagine.
- confronto tra strategie di sostituzione delle pagine e ottimizzazione
- impatto della dimensione pagina su prestazioni della memoria virtuale
- comportamento del programma nella paginazione

- condivisione e protezione dei sistemi di memoria virtuale
- hardware che rende i sistemi di memoria virtuale possibili

Memoria virtuale

- Risolve problemi di spazio di memoria limitato
- Crea l'illusione che esista più memoria di quella disponibile nel sistema

- Due tipi di indirizzi nei sistemi di memoria virtuale
  - Indirizzi **virtuali**
    - riferiti dai processi
  - Indirizzi **fisici**
    - descrive indirizzi nella memoria principale

- Memory management unit (MMU)
  - Componente hw dedicata alla traduzione di indirizzi virtuali in indirizzi fisici

Memoria virtuale - Introduzione

Posizione e funzioni della MMU

Memoria virtuale - Introduzione

Evoluzione delle organizzazioni di memoria

<table>
<tr><th>Real</th><th colspan="4">Real</th><th colspan="3">Virtual</th></tr>
<tr><td>Single-user dedicated systems</td><td colspan="4">Real memory multiprogramming systems</td><td colspan="3">Virtual memory multiprogramming systems</td></tr>
<tr><td></td><td colspan="2">Fixed-partition multi-programming</td><td colspan="2">Variable-partition multi-programming</td><td>Pure paging</td><td>Pure segmentation</td><td>Combined paging and segmentation</td></tr>
<tr><td></td><td>Absolute</td><td>Re-locatable</td><td colspan="4"></td><td colspan="3"></td></tr>
</table>

© Deitel & Ass. Inc.

Memoria virtuale: concetti di base

- Spazio di indirizzamento virtuale *V*
  - Intervallo di indirizzi virtuali ai quali un processo può fare riferimento
- Spazio di indirizzamento reale *R*
  - Intervallo di indirizzi fisici disponibili su un particolare sistema di computer
- Meccanismo di traduzione dinamica degli indirizzi (DAT – Dynamic Address Translation)
  - Converte indirizzi virtuali in indirizzi fisici durante l'esecuzione del programma

Memoria virtuale: concetti di base

- Spesso | *V* | >> | *R* |
  - S.O. deve memorizzare parti di *V* per ogni processo al di fuori della memoria principale
  - Schema di memoria a due livelli
    - S.O. scambia porzioni di *V* tra memoria principale (e la cache) e memoria secondaria

Introduzione

Relazione fra
indirizzo virtuale
e indirizzo reale
fisico di memoria
dato dalla
Tabella delle pagine

Introduzione

Operazioni interne di una MMU con16 4 KB pagine

Tabella delle pagine

Tipica riga della tabella delle pagine

Memoria virtuale: concetti di base

Memoria a due livelli

Memoria virtuale: concetti di base

Esistono parti dello spazio di indirizzamento in memoria e in memoria virtuale

© Deitel & Ass. Inc.

Mapping dei blocchi

- Mappa di traduzione degli indirizzi
  - Indica quali regioni dello spazio di indirizzamento virtuale di un processo, *V*, sono attualmente in memoria principale e dove si trovano
  - Informazioni organizzati in **blocchi**
  - Dimensione dei blocchi
    - Frammentazione interna
    - Tempo di trasferimento dei blocchi
    - Variabile o fissa

Mapping dei blocchi

Memoria Virtuale

Memoria Fisica

Mappa degli indirizzi virtuali in indirizzi reali

Mapping dei blocchi

- Contiguità artificiale
  - indirizzi virtuali contigui potrebbero non corrispondere agli indirizzi di memoria reale contigui
  - Il programmatore vede gli indirizzi virtuali contigui

Mapping dei blocchi

Contiguità artificiale

© Deitel & Ass. Inc.

Mapping dei blocchi

- Pagine
  - I blocchi sono di dimensione fissa
  - La tecnica è chiamata **paginazione**
- Segmenti
  - I blocchi possono avere dimensioni diverse
  - La tecnica è chiamata **segmentazione**
- Mappa dei blocchi
  - Sistema che rappresenta gli indirizzi come coppie ordinate

Mapping dei blocchi

Formato dell'indirizzo virtuale in un sistema con mapping dei blocco

Numero di blocco	Spostamento
b	*d*

Indirizzo virtuale $v = (b, d)$

Mapping dei blocchi

- Dato un indirizzo virtuale $v = (b, d)$ si traduce nell'indirizzo reale r
  - tabella della mappa dei blocchi per ogni processo
  - Una riga per ogni blocco del processo, ordinate
  - Indirizzo reale *a* della tabella del nuovo processo da caricare
  - Registro origine della tabella della mappa dei blocchi

- Traduzione
  - Si aggiunge ad *a* il numero di blocco, *b*, per individuare la riga nella tabella di mappa dei blocchi
  - La riga produce l'indirizzo, *b'*, inizio del blocco *b* in memoria principale
  - Si aggiunge lo spostamento *d* a *b'* per formare l'indirizzo reale, *r*

Mapping dei blocchi

Traduzione di indirizzo virtuale con mapping dei blocchi

Paginazione

- Paginazione usa il mapping di blocchi a dimensione fissa
  - L'indirizzo virtuale nel sistema di paginazione è una coppia ordinata

    $v = (p, d)$

    - p è il numero della pagina in memoria virtuale su cui risiede il dell'elemento di riferimento
    - d è lo spostamento dall'inizio della pagina p in cui si trova l'elemento di riferimento

Paginazione

Formato dell'indirizzo virtuale in un sistema con paginazione puro

Numero di pagina	Spostamento
p	*d*

Indirizzo virtuale $v = (p, d)$

Paginazione

- Page frame
  - Blocchi della memoria principale a dimensione fissa destinati a contenere pagine
  - Inizia a un indirizzo di memoria principale che è un multiplo intero della dimensione fissa della pagina (p_s)
  - Il sistema può posizionare le pagine in un qualsiasi page frame libero

Paginazione

Memoria principale divisa in page frames

Page frame number	Page frame size	Range of physical memory addresses
0	p_s	$0 \rightarrow p_s - 1$
1	p_s	$p_s \rightarrow 2p_s - 1$
2	p_s	$2p_s \rightarrow 3p_s - 1$
3	p_s	$3p_s \rightarrow 4p_s - 1$
4	p_s	$4p_s \rightarrow 5p_s - 1$
5	p_s	$5p_s \rightarrow 6p_s - 1$
6	p_s	$6p_s \rightarrow 7p_s - 1$
7	p_s	$7p_s \rightarrow 8p_s - 1$
⋮		

© Deitel & Ass. Inc.

Paginazione

Corrispondenza

fra

indirizzi di memoria virtuale e

indirizzi di memoria fisica

in un sistema di paginazione puro

Paginazione

- Tabella della pagine pagina virtuale page frame
  - ma non tutte le pagine risiedono in memoria principale

- **Page table entry (PTE)**

  (riga della tabella delle pagine)
  - Indica che la pagina virtuale *p* è nel page frame *p'*
  - Contiene un bit di residenza, *r*, per indicare se la pagina è in memoria principale
    - *r* = 1, PTE memorizza il numero di page frame *p'*
    - *r* = 0, PTE memorizza la posizione della pagina in memoria secondaria

Paginazione

Riga della tabella delle pagine

Bit di residenza	Indirizzo di memoria secondaria (se la pagina non è in memoria principale)	Page frame (se la pagina è in memoria principale)
r	*s*	*p'*

Bit di residenza della pagina

r = 0 la pagina non è in memoria principale

r = 1 la pagina è in memoria principale

© Deitel & Ass. Inc.

Paginazione
traduzione dell'indirizzo con mapping diretto

- Mapping Diretto
  - Traduzione dinamica degli indirizzi da virtuali a fisici con paginazione - simile alla traduzione di indirizzi di blocchi
  - La tabella delle pagine contiene una riga per ogni pagina dello spazio virtuale *V*
  - Richiede l'accesso alla memoria per ogni mapping di una pagina
  - I riferimenti di processo con indirizzamento virtuale $v = (p, d)$
    - DAT (Dynamic Address Translation) aggiunge l'indirizzo base della tabella delle pagine del processo, *b*, al numero di pagina riferito, *p* (riga nella tabella)
    - $b + p$ forma l'indirizzo di memoria principale della PTE per la pagina *p*
    - Il sistema concatena il page frame corrispondente, *p'*, con lo spostamento, *d*, per formare l'indirizzo reale, *r*

Traduzione dell'indirizzo di paginazione con mapping diretto

© Deitel & Ass. Inc.

Traduzione dell'indirizzo di paginazione con mapping associativo

- Mantenere tutta la tabella delle pagine in memoria cache spesso non è praticabile
  - A causa di costo della memoria cache ad alta velocità, per spazi di indirizzamento di programmi relativamente grandi

- Aumentare le prestazioni di traduzione dinamica degli indirizzi
  - Posizionare l'intera tabella delle pagine nella **memoria associativa** con indirizzamento per contenuto
  - Ogni posizione nella memoria associativa è cercata contemporaneamente
  - Costo elevato della memoria cache con indirizzamento per contenuto

Traduzione dell'indirizzo di paginazione con mapping associativo

Paginazione con traduzione di indirizzo con mapping associativo puro

© Deitel & Ass. Inc.

Traduzione dell'indirizzo di paginazione con mapping diretto/associativo

- Compromesso fra costo e prestazioni
  - PTE sono prevalentemente memorizzati nelle tabelle direttamente mappate nella memoria principale
  - PTE utilizzate più di recente sono memorizzati nella memoria cache associativa ad alta velocità chiamata

    Traduzione Lookaside Buffer (TLB)
    - Se PTE non si trova nel TLB, il meccanismo DAT ricerca nella tabella nella memoria principale
  - Può portare ad alte prestazioni con un TLB relativamente piccolo, grazie al principio di **località** temporale
    - È probabile che una pagina riferita recentemente da un processo sia nuovamente riferita in tempi brevi
  - TLB è controllabile sia hardware (parte delle MMU) sia dal s.o.

Traduzione dell'indirizzo di paginazione con mapping diretto/associativo

Traduzione di indirizzo combinata con mapping associativa/diretta

© Deitel & Ass. Inc.

TLBs – Translation Lookaside Buffers

Valido	Pagina virtuale	Modificato	Protezione	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

Una TLB per accelerare la paginazione

Tabelle di Pagine multilivello

- Tabelle di pagine multilivello
  - Sistema in grado di memorizzare locazioni non contigue di memoria quelle parti della tabella delle pagine di processo che il processo sta utilizzando
  - Riduce il numero di righe della tabella in memoria
  - Gerarchia delle tabelle di pagina
    - Ogni livello contiene una tabella che memorizza i puntatori alle tabelle del livello inferiore
    - I livelli più bassi hanno tabelle con le traduzioni di indirizzi
  - Può ridurre l'overhead di memoria rispetto al sistema di mappatura diretta
  - Richiede un accesso alla memoria in più per il mapping, limitabile con località e uso di TLB

Introduzione

Indirizzi di 32 bit con 2 page campi di tabella

Tabelle di pagine a due livelli

Tabelle di Pagine multilivello

Traduzione di indirizzi

© Deitel & Ass. Inc.

Tabella inversa delle pagine

- Tabelle inversa delle pagine
  - La tabella multilivello riduce il numero di righe della tabella in memoria, utile se il processo usa solo una parte dello spazio di indirizzamento
  - Altrimenti non è conveniente
  - La tabella **inversa** memorizza un PTE (riga della tabella delle pagine) in memoria per ogni page frame del sistema
  - Il numero di righe della tabella in memoria dipende non dallo spazio, *V*, ma dal numero di page frame (memoria fisica), *R*
  - È l'inverso rispetto ad una tabella di pagine tradizionale
    - Tradizionale: pagina virtuale page frame
    - Inverse: page frame pagina virtuale
  - Non memorizza informazioni relative alla memoria secondaria
  - Utilizza funzioni hash per mappare l'indirizzo virtuale nella riga nella tabella inversa delle pagine

Tabella inversa delle pagine

Traduzione degli indirizzi di pagina con le tabelle di pagina invertite

© Deitel & Ass. Inc.

Tabella inversa delle pagine

Confronto tra una tabella delle pagine tradizionali e una tabella inversa delle pagine

Tabella inversa delle pagine

- Hashing può causare collisioni che aumentano il tempo di traduzione dell'indirizzo aumentando il numero di accessi alla memoria
  - **Collisione**: quando più indirizzi virtuali tradotti con hash danno lo stesso valore (numero di page frame)
  - Concatenamento (chaining) per evitare sovrascritture di elementi in collisione
  - Possibile diminuzione delle collisioni aumentando l'intervallo dei valori della funzione hash
  - Non si può aumentare la dimensione della tabella inversa delle pagine perché deve memorizzare esattamente una PTE per ogni page frame
  - Hash Anchor Table (HAT) aumenta l'intervallo dei valori della funzione hash con l'aggiunta di un ulteriore livello indiretto
  - La dimensione deve essere scelta attentamente per bilanciare la frammentazione della tabella, l'overhead e le prestazioni
  - Esempi: architetture IA-64 Intel, PA-RISC HP, PowerPC Apple

Tabella inversa delle pagine

Tabella inversa delle pagine con hash anchor table

© Deitel & Ass. Inc.

Condivisione in un sistema di paginazione

- Condivisione nei sistemi multiprogrammati
  - Riduce la memoria consumata dai programmi che utilizzano i dati e/o istruzioni in comune
  - Richiede che il sistema identifichi ogni pagina come condivisibile o non condivisibile
  - Procedure o dati
  - Procedure non modificabili: *rientranti*
  - Problema della modificabilità

Condivisione in un sistema di paginazione

Condivisione in un sistema puro di paginazione

© Deitel & Ass. Inc.

Condivisione in un sistema di paginazione

- Esempio
  - `fork` in Unix crea un processo duplicato inizialmente identico
    - Tecnica del *copy-on-write*
  - Nel momento della modifica viene creata una copia
  - Miglior uso della memoria
  - Maggior velocità di creazione dei processi
  - Overhead nel momento della copia

Sostituzione in un sistema di paginazione

- Strategie di **sostituzione**
  - Tecnica usata da un sistema per selezionare le pagine da sostituire quando la memoria è piena
  - Determina **dove** caricare in memoria principale una pagina o un segmento da inserire
- Strategie di **Fetch**
  - Determina **quando** le pagine o segmenti devono essere caricati nella memoria principale
  - Strategie a richiesta
    - nel momento in cui il processo vi fa riferimento
  - Strategie a previsione
    - Uso di euristiche per prevedere a quali pagine un processo farà presto riferimento e caricamento di quelle pagine o segmenti
    - Maggior overhead

Località

- Processo tende a fare riferimento alla memoria secondo modelli altamente localizzati
  - Località spaziale
  - Località temporale
  - In sistemi con paginazione, i processi tendono a favorire determinati sottoinsiemi delle loro pagine, e tali pagine tendono ad essere adiacenti nello spazio di indirizzamento virtuale del processo
  - Osservazioni empiriche

Paginazione **a richiesta**

- Quando un processo inizia l'esecuzione, il sistema carica in a memoria principale la pagina che contiene la sua prima istruzione

- Successivamente il sistema carica una pagina dalla memoria secondaria alla memoria principale solo quando il processo vi fa esplicito riferimento

- Richiede al processo di accumulare pagine una alla volta

Paginazione a richiesta

© Deitel & Ass. Inc.

Paginazione **a previsione**

prefetching

- Il Sistema Operativo cerca di prevedere di quali pagine un processo avrà bisogno e pre-carica queste pagine quando vi è spazio di memoria disponibile

- Strategia di paginazione a previsione
  - Deve essere progettato attentamente in modo che l'overhead per implementarla non riduca le prestazioni del sistema
  - Quanta **memoria** pre-allocare
  - Quante **pagine** pre-caricare ogni volta
  - Quale **politica**

Sostituzione di pagina

- Quando un processo genera un *page fault*, il gestore della memoria deve
  - **individuare** la pagina di memoria secondaria a cui fa riferimento,
  - **caricarla** nel page frame della memoria principale e
  - **aggiornare** la riga corrispondente della tabella delle pagina
- Bit di modifica (*dirty bit*)
  - 1 se la pagina è stata modificata, 0 altrimenti
  - aiuta il sistema a determinare rapidamente la modifica di pagina
- Scrittura su disco (*flush*) di una pagina modificata: *overhead*
  - Sistemi con *flush* periodico automatico (preventivo)

Strategie di sostituzione delle pagine

- Strategia di sostituzione di pagina **Ottima** (OPT o MIN)
  - Ottiene prestazioni ottime, sostituisce la pagina alla quale non verrà fatto riferimento nuovamente più avanti nel futuro
  - Minimo numero di page fault => minimo tempo di attesa
  - Ideale
  - Conoscenza del comportamento futuro esatto del processo
- Strategie di sostituzione di pagina: caratteristiche
  - Euristiche per selezionare una pagina da sostituire
  - Approssimazioni della previsione del comportamento futuro
  - Overhead

Sostituzione casuale delle pagine (RAND)

- Strategia di sostituzione di pagina con basso overhead e semplice
- Veloce
- Non discrimina tra i processi

- Equa: ogni pagina nella memoria principale ha una uguale probabilità di essere selezionata per la sostituzione

- Potrebbe facilmente selezionare come prossima pagina da sostituire anche la pagina che alla quale verrà fatto il prossimo riferimento

- Poco usata

Sostituzione delle pagine First-In-First-Out (FIFO)

- Sostituisce la pagina che è stata nel sistema il **più a lungo** (ordine in base al tempo di arrivo)
- Può sostituire anche le pagine molto utilizzate
- Può essere implementata con overhead relativamente basso
- Non conveniente (non praticabile) per la maggior parte dei sistemi
- Base per altre strategie

Sostituzione delle pagine First-In-First-Out (FIFO)

Page reference	Result
A	Fault
B	Fault
C	Fault
A	No fault
D	Fault
A	Fault

FIFO page replacement with three pages available

A	–	–
B	A	–
C	B	A
C	B	A
D	C	B
A	D	C
⋮	⋮	⋮

A is replaced

B is replaced

© Deitel & Ass. Inc.

11.6.3 Anomalia della strategia FIFO

- Anomalia di Belady (o FIFO)
  - Alcuni modelli di riferimento di pagina effettivamente causano un aumento del numero di errori di pagina all'aumentare del numero di page frame assegnati a un processo
  - controintuitivo

Anomalia della strategia FIFO

I page faults possono aumentare con il numero di page frame allocati.

Page reference	Result	FIFO page replacement with three pages available			Result	FIFO page replacement with four pages available			
A	Fault	A	–	–	Fault	A	–	–	–
B	Fault	B	A	–	Fault	B	A	–	–
C	Fault	C	B	A	Fault	C	B	A	–
D	Fault	D	C	B	Fault	D	C	B	A
A	Fault	A	D	C	No fault	D	C	B	A
B	Fault	B	A	D	No fault	D	C	B	A
E	Fault	E	B	A	Fault	E	D	C	B
A	No fault	E	B	A	Fault	A	E	D	C
B	No fault	E	B	A	Fault	B	A	E	D
C	Fault	C	E	B	Fault	C	B	A	E
D	Fault	D	C	E	Fault	D	C	B	A
E	No fault	D	C	E	Fault	E	D	C	B

Three "no faults" Two "no faults"

© Deitel & Ass. Inc.

Anomalia della strategia FIFO

All pages frames initially empty

		0	1	2	3	0	1	4	0	1	2	3	4
Youngest page		0	1	2	3	0	1	4	4	4	2	3	3
			0	1	2	3	0	1	1	1	4	2	2
Oldest page				0	1	2	3	0	0	0	1	4	4
		P	P	P	P	P	P	P			P	P	

9 Page faults

(a)

		0	1	2	3	0	1	4	0	1	2	3	4
Youngest page		0	1	2	3	3	3	4	0	1	2	3	4
			0	1	2	2	2	3	4	0	1	2	3
Oldest page				0	1	1	1	2	3	4	0	1	2
					0	0	0	1	2	3	4	0	1
		P	P	P	P			P	P	P	P	P	P

10 Page faults

(b)

- FIFO con 3 page frame
- FIFO con 4 page frame
- *P* indica quale riferimento di pagina mostra page fault

Sostituzione di pagina Least-Recently-Used (**LRU**)

- Sfrutta la località **temporale** sostituendo la pagina che ha trascorso più tempo in memoria senza essere riferita

- Può portare a prestazioni migliori rispetto FIFO

- Maggior overhead di sistema
  - Aggiornamento dei link ad ogni accesso alla memoria
  - Lista delle pagine
  - Oppure uso di un contatore per ogni riga di pagina nella tabella

- LRU può portare a scarse prestazioni se la pagina meno utilizzata di recente è la prossima pagina a cui fa riferimento un programma in una iterazione all'interno di un ciclo che fa riferimento a diverse pagine
  - Ad ogni sostituzione la pagina se riferita va ricaricata

Sostituzione di pagina Least-Recently-Used (**LRU**)

Page reference	Result	LRU page replacement with three pages available		
A	Fault	A	–	–
B	Fault	B	A	–
C	Fault	C	B	A
B	No fault	B	C	A
B	No fault	B	C	A
A	No fault	A	B	C
D	Fault	D	A	B
A	No fault	A	D	B
B	No fault	B	A	D
F	Fault	F	B	A
B	No fault	B	F	A

© Deitel & Ass. Inc.

Sostituzione di pagina Least-Frequently-Used (**LFU**)

- Sostituisce la pagina che è **meno** intensamente riferita
- Basato sul fatto che una pagina alla quale non si fa riferimento spesso non è probabile che sia riferita in futuro
- Potrebbe facilmente selezionare la pagina sbagliata per la sostituzione
  - Una pagina che è stata molto riferita in passato può non essere più riferita di nuovo, ma rimarrà in memoria mentre altre pagine attive più recenti vengono sostituite

Sostituzione di pagina Not-Frequently-Used (**NFU**)

- Sostituisce la pagina che è **non è stata** recentemente riferita

- Usa un contatore associato ad ogni pagina e inizializzato a 0
- Ad ogni interrupt del clock il s.o. fa la scansione dei bit delle pagine di memoria e somma il bit R al contatore, che tiene così traccia del numero di riferimenti
- Si sceglie la pagina con il valore del contatore minimo

Sostituzione di pagina Not-Recently-Used (**NRU**)

- Approssima la strategia LRU con un overhead minimo utilizzando il bit di riferimento e il bit di modifica per determinare quale pagina non è stata utilizzata di recente e può essere rapidamente sostituita
  - bit di riferimento = 0 se la pagina non è stata riferita,
    1 altrimenti (bit di accesso)
  - bit di modifica = 0 se la pagina non è stata modificata
    1 altrimenti
  - due bit hw nella tabella delle pagine
  - cerca una pagina non riferita (approssima LRU)
  - se non la trova cerca un pagina non modificata

- Può essere implementata su macchine che non dispongono di bit di riferimento e / o bit di modifica hardware (simulabile sw)

- Con molti utenti le pagine vengono spesso tutte riferite

- Periodicamente si possono azzerare i bit di riferimento

Sostituzione di pagina Not-Used-Recently (**NRU**)

Tipi di pagine con NUR

Classe	Bit di riferimento	Bit di modifica	Descrizione
Classe 0	0	0	Migliore scelta per la sostituzione
Classe 1	0	1	Sembra non realistico
Classe 2	1	0	
Classe 3	1	1	Peggiore scelta per la sostituzione

Varianti della strategia FIFO
Second-Chance e Sostituzione a orologio

- Strategia di sostituzione di pagina Second chance
  - Esamina il bit di riferimento della **pagina più vecchia**
    - Se è off (0) la strategia seleziona la pagina per la sostituzione
    - Se è on (1) la strategia azzera il bit e sposta la pagina in coda FIFO e si considera come un nuovo arrivo
  - Assicura che le pagine attive siano quelle con minor probabilità di essere sostituite

- Strategia di sostituzione a orologio
  - Simile a second chance, ma organizza le pagine in una lista circolare invece della lista lineare
  - Puntatore che scorre la lista ad ogni page fault; se bit ref =0 si scorre (simula FIFO)

Varianti della strategia FIFO: Second-Chance

Operazioni di una strategia second chance

a) pagine ordinate secondo FIFO

b) la lista delle pagine se il fault accade al tempo 20, <u>*A*</u> ha bit *R=1*

(i numeri sopra alle pagine sono i tempi di caricamento)

Varianti della strategia FIFO: Sostituzione a orologio

Quando si verifica un page fault,
viene analizzata la pagina
cui sta puntando la lancetta.
L'azione intrapresa dipende dal bit R:
R = 0: Rimuovi la pagina
R = 1: Azzera R e fai avanzare la lancetta

Sostituzione delle pagine *Far*

- Idea: **lontananza** delle pagine
- Crea un grafo di accesso che rappresenta il modello dei riferimenti di un processo
- Grafo:
  - nodi=pagine
  - archi=riferimenti
- Sostituisce la pagina non referenziato che è più lontana nel grafo di accesso da qualsiasi pagina riferita
- Prestazioni a livelli quasi ottimali
- Non è stata solitamente implementata in sistemi reali
  - La ricerca e gestione del grafo di accesso è complessa e senza supporto hardware

Sostituzione delle pagine *Far*

Esempio di grafo di accesso nella strategia di sostituzione *far*

© Deitel & Ass. Inc.

Modello *Working Set*

- Per eseguire un programma in modo efficiente
  - Il sistema deve mantenere in memoria principale quel **sottoinsieme di pagine favorito** del programma

- Altrimenti
  - Il sistema potrebbe sperimentare un'eccessiva attività di paginazione causando una bassa utilizzazione del processore chiamato thrashing quando il programma richiede ripetutamente pagine in memoria secondaria

Modello *Working Set*

Modello di riferimenti alla memoria che mostra la località

(Reprinted by permission from IBM Systems Journal. © 1971 by International Business Machines Corporation.)

Modello *Working Set*

- Poiché i processi mostrano **località**

  aumentare il numero di page frame assegnati ad un processo
  oltre una certa soglia
  ha poco o nessun effetto sul tasso di page fault

Modello *Working Set*

Dipendenza del tasso di page fault sulla quantità di memoria per le pagine di un processo.

© Deitel & Ass. Inc.

Modello *Working Set*

- Il **working set** delle pagine di un processo

$$W(t, w)$$

- insieme di pagine alle quali il processo fa riferimento durante l'intervallo di tempo $[t - w, t]$

Modello *Working Set*

- La dimensione del working set del processo aumenta asintoticamente con la dimensione del programma del processo all'aumentare della finestra del working set

Modello *Working Set*

Dimensione del working set in funzione della dimensione della finestra

- working set: insieme di pagine usate dai *w* più recenti riferimenti
- *W(t,w)* dimensione del *working set* al tempo *t*

© Deitel & Ass. Inc.

Modello *Working Set*

- Allo spostarsi tra working sets di un processo, il sistema mantiene temporaneamente in memoria le pagine di che non sono più nel working set corrente del processo
  - L'obiettivo della gestione della memoria del working set è di ridurre questa cattiva allocazione
  - Tenendo traccia del working set al momento di un page fault si elimina una pagina esterna al w.s.
  - Approssimazioni del w.s. per maggior efficienza
  - Uso del tempo di esecuzione anziché del numero di riferimenti
  - Tempo virtuale attuale (corrente): tempo di CPU usata effettivamente dal suo avvio.
  - Il w.s. è l'insieme di pagine referenziate negli ultimi τ sec. di tempo virtuale

Algoritmo di sostituzione di pagina con Working Set

Modello *Working Set*

Allocazione della memoria principale con gestione della memoria del working set

© Deitel & Ass. Inc.

Strategia di sostituzione WSClock (Clock e **Working Set**)

- Per non memorizzare la tabella delle pagine per intero da scandire ad ogni page fault
- Uso dell'algoritmo clock unito al working set – lista circolare

Se R==1 non la seleziona
Lo setta a 0 e avanza

Strategia di sostituzione WSClock (Clock e **Working Set**)

(c)

(d)

Se (R==0 e età > τ)
rimuovi questa pagina

Se (R==0)
avanza la lancetta

Se scorre tutte le pagine e torna al punto iniziale:

-Se qualche scrittura è schedulata continua a scorrere finché non trova una pagina con R==0

-Altrimenti tutte le pagine sono nel w.s., se ne sceglie una qualunque (prima pulita, oppure l'attuale)

Strategia di sostituzione Page-Fault-Frequency (**PFF**)

- Regola l'insieme di pagine residenti di un processo
  - Basato sulla frequenza in cui il processo ha un fault
  - Basato sul tempo tra page faults, chiamato tempo interfault del processo

- Vantaggio di PFF sulla sostituzione delle pagine del working set
  - Minor overhead
    - PFF regola set l'insieme delle pagine residenti solo dopo ogni page fault
    - Il meccanismo working set deve operare dopo ogni riferimento alla memoria

Rilascio delle pagine

- Le pagine inattive possono rimanere in memoria principale a lungo finché la strategia di gestione rileva che il processo non ne ha più bisogno
  - Un modo per risolvere il problema
    - Il processo chiede una sostituzione volontaria di pagina per liberare un page frame del quale sa di non aver più bisogno
    - Elimina il ritardo causato dal graduale rilascio delle pagine del working set del processo
    - La miglior opportunità è nel supporto del compilatore e del sistema operativo

Sintesi e confronto fra le strategie di sostituzione

Algoritmo	Commento
Ottimale	Non implementabile, ma utile come indice di confronto e valutazione
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
LRU (Last Recently Used	Eccellente, ma difficile da implementare con precisione
NFU (Not 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

Dimensione della pagina

- Alcuni sistemi migliorano le prestazioni e l'utilizzo della memoria, fornendo diverse dimensioni di pagina
  - dimensione **piccola** di pagina
    - Minor frammentazione interna
    - Può ridurre la quantità di memoria necessaria per contenere working set di un processo
    - Maggior memoria disponibile per altri processi – minor spreco di memoria
    - Tabella delle pagine più grande

Dimensione della pagina

- dimensione **grande** di pagina
  - Riduce la memoria sprecata dalla frammentazione della tabella
  - Abilita ogni riga della TLB a mappare una regione della memoria più grande, migliorando le prestazioni
  - Riduce il numero di operazioni di I/O del sistema per caricare il working set di un processo in memoria
- dimensione **multipla** di pagina
  - Possibile frammentazione esterna

Dimensione della pagina

Frammentazione interna in un sistema con paginazione e segmentazione

© Deitel & Ass. Inc.

Dimensione della pagina

Dimensioni della pagina in varie architetture di processori

Manufacturer	Model	Page Size	Real address size
Honeywell	Multics	1KB	36 bits
IBM	370/168	4KB	32 bits
DEC	PDP-10 and PDP-20	512 bytes	36 bits
DEC	VAX 8800	512 bytes	32 bits
Intel	80386	4KB	32 bits
Intel / AMD	Pentium 4 / Athlon XP	4KB or 4MB	32- or 36 bits
Sun	UltraSparc II	8KB, 64KB, 512KB, 4MB	44 bits
AMD	Opteron / Athlon 64	4KB, 2MB and 4MB	32, 40, or 52 bits
Intel-HP	Itanium, Itanium 2	4KB, 8KB, 16KB, 64KB, 256KB, 1MB, 4MB, 16MB, 64MB, 256MB	Between 32 and 63 bits
IBM	PowerPC 970	4KB, 128KB, 256KB, 512KB, 1MB, 2MB, 4MB, 8MB, 16MB, 32MB, 64MB, 128MB, 256MB	32 or 64 bits

© Deitel & Ass. Inc.

Comportamento del programma con paginazione

- I processi tendono a fare riferimento a una parte significativa delle loro pagine entro un breve periodo di tempo dopo l'inizio dell'esecuzione
- Accedono la maggior parte delle loro pagine rimanenti a un ritmo più lento

Percentuale di pagine del processo riferite nel tempo

© Deitel & Ass. Inc.

Comportamento del programma con paginazione

- Il tempo medio interfault in generale aumenta monotonicamente
  - Più page frame un processo ha, più lungo è il tempo che intercorre tra errori di pagina

Dipendenza del tempo interfault del numero di frame di pagina assegnato a un processo

© Deitel & Ass. Inc.

Strategie di sostituzione Globali vs. Locali

- Implementazione di un sistema di memoria virtuale con paginazione
  - Strategie **Globali** di sostituzione di pagina: applicate a tutti i processi come unità
    - Tendono a ignorare le caratteristiche individuali di comportamento del processo
    - Strategia globale LRU (**gLRU**)
      - Sostituisce la pagina meno utilizzata di recente in tutto il sistema
    - **SEQ** strategia di sostituzione di pagina globale (sequenza)
      - Usa la strategia LRU per sostituire le pagine fino a quando non viene rilevato sequenza di errori di pagina a pagine contigue, a quel punto utilizza la strategia most-recently-used (**MRU**)
  - Strategie **Locali**: considerare ogni processo individualmente
    - Consente di regolare il sistema di allocazione della memoria in base alla importanza relativa di ciascun processo per migliorare le prestazioni

Confronto fra le strategie di sostituzione globali e locali

	Età
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(a)

A0
A1
A2
A3
A4
A6
B0
B1
B2
B3
B4
B5
B6
C1
C2
C3

(b)

A0
A1
A2
A3
A4
A5
B0
B1
B2
A6
B4
B5
B6
C1
C2
C3

(c)

a) Configurazione originale
b) Sostituzione locale di pagine
c) Sostituzione globale di pagine

Esempio: Sostituzione di pagina in Linux

- Linux utilizza una variante dell'algoritmo orologio per approssimare una strategia di sostituzione di pagina LRU

- Il gestore di memoria utilizza due liste collegate

  - lista attiva
    - Contiene le pagine attive
    - Le pagine usate più di recente sono in cima alla lista attiva

  - lista inattive
    - Contiene le pagine inattive
    - Le pagine meno usate di recente in fondo alla della lista inattiva
  - Vengono sostituite solo le pagine della lista inattiva

Esempio: Sostituzione di pagina in Linux

© Deitel & Ass. Inc.

Segmentazione

- **Segmento**
  - Contiene porzioni significative del programma (es. procedure, array, stack)
  - Composto da posizioni **contigue**
  - I segmenti non devono necessariamente essere della stessa dimensione né devono essere adiacenti in memoria principale
  - Il segmento è un concetto logico e non fisico (come per la pagina)

- Un processo può essere in esecuzione mentre le sue istruzioni correnti e i dati riferiti sono in segmenti di memoria principale

- Se il processo fa riferimento a dati in memoria secondaria il segmento corrispondente va caricato

Segmentazione

Allocazione di memoria non contigua in un sistema con segmentazione della memoria reale

© Deitel & Ass. Inc.

Segmentazione

- Un processo fa riferimento a un indirizzo di memoria virtuale $v = (s, d)$
  - s è il numero di segmento in memoria virtuale
  - d è lo spostamento all'interno del segmento s in cui si trova l'elemento indirizzato

Formato dell'indirizzo virtuale in un sistema di segmentazione pura

Segmentazione
traduzione dell'indirizzo con mapping diretto

- Il processo fa riferimento ad un indirizzo di memoria virtuale $v = (s, d)$
  - DAT aggiunge l'indirizzo di base della **tabella dei segmenti** del processo, b, al numero di segmento riferito, s
  - $b + s$ costituisce l'indirizzo di memoria principale della riga nella tabella dei segmenti per il segmento s
  - La riga contiene l'indirizzo iniziale del segmento in memoria, s'
  - Il sistema aggiunge s' allo spostamento, d, per formare l'indirizzo reale, r

Segmentazione –Traduzione dell'indirizzo con mapping diretto

Traduzione dell'indirizzo virtuale in un sistema con segmentazione puro

© Deitel & Ass. Inc.

Segmentazione
traduzione dell'indirizzo con mapping diretto

- Riga della tabella della mappa dei segmenti
  - Indica che quel segmento *s* inizia all'indirizzo di memoria reale *s'*
  - Contiene un bit di residenza, *r*, che indica se il segmento è in memoria
    - Se r = 1, *s'* memorizza l'indirizzo di base del segmento
    - Se r = 0 , *a* memorizza la posizione del segmento in memoria secondaria
  - Inoltre contiene un campo lunghezza, *l*, che indica la dimensione del segmento
    - Può essere usato per evitare che un processo riferisca indirizzi fuori dal segmento *(eccezione di overflow del segmento)*
  - Bit di protezione per controllare se il tipo di operazione è ammessa *(eccezione di protezione del segmento)*
    - (es. lettura, scrittura,..)

Segmentazione –traduzione dell'indirizzo con mapping diretto

Riga della tabella dei segmenti

Segment resident bit	Secondary storage address (if not in main memory)	Segment length	Protection bits	Base address of segment (if in main memory)
r	*a*	*l*		*s'*

r = 0 if segment is not in main memory
r = 1 if segment is in main memory

© Deitel & Ass. Inc.

Condivisione in un sistema con segmentazione

- La condivisione segmenti può causare meno overhead della condivisione nel sistema con paginazione pura e con mapping diretto
  - Potenzialmente pochi elementi della tabella delle mappa devono essere condivise
  - Controllo delle interferenze e meccanismi di protezione

Condivisione in un sistema con segmentazione

© Deitel & Ass. Inc.

Protezione e Controllo degli Accessi in Sistema con Segmentazione

- Uno schema per implementare la **protezione della memoria** nei sistemi di segmentazione è chiavi di protezione della memoria

- Chiave di protezione
  - valore associato ad un processo

- Se la chiave di protezione per il processore e il blocco richiesto sono gli stessi, il processo può accedere al segmento

- Chiavi manipolabili solo da istruzioni privilegiate

Protezione e Controllo degli Accessi in Sistema con Segmentazione

Protezione della memoria con le chiavi

in sistemi multiprogrammati

con di allocazione di memoria non contigua

© Deitel & Ass. Inc.

Protezione e Controllo degli Accessi in Sistema con Segmentazione

- Uno schema più comune è quello di utilizzare i **bit di protezione** che specificano se un processo è autorizzato a compiere operazioni
  e.g. se può leggere, scrivere, eseguire codice o aggiungere ad un segmento

- **diritti di accesso** ai segmenti

Type of access	Abbreviation	Description
Read	R	This segment may be read.
Write	W	This segment may be modified.
Execute	E	This segment may be executed.
Append	A	This segment may have information added to its end.

© Deitel & Ass. Inc.

Protezione e Controllo degli Accessi in Sistema con Segmentazione

Combinazione di diritti di accesso in lettura, scrittura ed esecuzione per definire modalità di controllo di accesso utili

Mode	Read	Write	Execute	Description	Application
Mode 0	No	No	No	No access permitted	Security.
Mode 1	No	No	Yes	Execute only	A segment made available to processes that cannot modify it or copy it, but that can run it.
Mode 2	No	Yes	No	Write only	These possibilities are not useful, because granting write access without read access is impractical.
Mode 3	No	Yes	Yes	Write/execute but cannot be read	
Mode 4	Yes	No	No	Read only	Information retrieval.
Mode 5	Yes	No	Yes	Read/execute	A program can be copied or executed but cannot be modified.
Mode 6	Yes	Yes	No	Read/write but no execution	Protects data from an erroneous attempt to execute it.
Mode 7	Yes	Yes	Yes	Unrestricted access	This access is granted to trusted users.

© Deitel & Ass. Inc.

Protezione e Controllo degli Accessi in Sistema con Segmentazione

- I bit di protezione vengono aggiunti alle righe della tabella dei segmenti e controllati quando un processo fa riferimento a un indirizzo
  - Se il segmento non è in memoria, viene generato un fallimento di accesso al segmento *(missing-segment fault)*
  - Se *d* > *l*, viene generata un'eccezione di overflow del segmento (*overflow-segment exception)*
  - Se non è consentita l'operazione (es. *read, write, execute, append*), viene generata un'eccezione di protezione di segmento (*segment-protection exception)*

Protezione e Controllo degli Accessi in Sistema con Segmentazione

Riga della tabella dei segmenti con i bit di protezione

Confronto fra paginazione e segmentazione

Formato dell'indirizzo virtuale in un sistema con segmentazione/paginazione

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Segmentazione e frammentazione esterna

Sistemi con Segmentazione/Paginazione

- Per sfruttare i vantaggi delle due tecniche
- I segmenti occupano una o più pagine
- Tutte le pagine del segmento non devono necessariamente essere in memoria principale contemporaneamente

- Le pagine contigue nella memoria virtuale non devono essere contigui nella memoria principale
- Indirizzo di memoria virtuale è implementato come una tripla ordinata

  $v = (s, p, d)$

  – *s* è il numero del segmento
  – *p* è il numero di pagina all'interno del segmento
  – *d* è lo spostamento all'interno della pagina in cui trova l'elemento desiderato

Traduzione dinamica dell’indirizzo in un sistema con segmentazione/paginazione

Formato dell'indirizzo virtuale in un sistema con segmentazione/paginazione

Sistemi con segmentazione/paginazione

- Riferimenti del processo agli indirizzi di memoria virtuale

$$v = (s, p, d)$$

  - TLB tabella delle pagine più usate nella memoria associativa
  - Si cerca in TLB la pagina *(s, p)*
    - Se si trova si ottiene il page frame *p'* che concatenato a *d* forma l'indirizzo reale *r*
    - Se non si trova si applica la mappatura **diretta**:
  - DAT aggiunge l'indirizzo di base della tabella dei segmenti, *b*, al numero di segmento di riferimento, *s*
  - *b* + *s* costituisce l'indirizzo di memoria della riga della tabella dei segmenti per il segmento *s*
  - La riga della tabella dei segmenti memorizza l'indirizzo base della tabella delle pagine, *s´*
  - Il numero di pagina, *p*, è aggiunto all'indirizzo base *s´* per individuare la PTE (riga) per la pagina *p*, che memorizza il numero di page frame *p´*
  - Si concatena *p´* con lo spostamento, *d*, a formare indirizzo reale *r*
  - La traduzione è caricata in TLB

Sistemi con segmentazione/paginazione

- In caso di *segment-missing fault* il S.O. cerca il segmento in memoria secondaria
- In caso di *page-fault* cerca e carica la pagina dalla memoria secondaria
- Possibile *segment-overflow*
- Possibili eccezioni di protezione

Traduzione dinamica di indirizzi in un sistema con segmentazione/paginazione

traduzione degli indirizzi virtuale con mapping combinato associativo/diretto in un sistema di segmentazione/paginazione

© Deitel & Ass. Inc.

Traduzione dinamica di indirizzi in un sistema con segmentazione /paginazione

Struttura della tabella per un sistema con segmentazione / paginazione

© Deitel & Ass. Inc.

Condivisione e protezione in un sistema con segmentazione/paginazione

- Sistemi di paginazione e segmentazione
  - Due processi condividono la stessa memoria quando ogni processo ha una riga della tabella di mappa dei segmenti che punta alla stessa tabella delle pagine
  - La condivisione è facilitata dalla paginazione
  - Controllo degli accessi come da segmentazione

- La condivisione richiede una gestione attenta da parte del sistema operativo
  - Diversi processi possono condividere le pagine e in caso di modifica devono essere aggiornate le tabelle
  - Uso di liste di PTE che mappano una pagina condivisa per facilitare l'aggiornamento in caso di modifica

Condivisione e protezione in un sistema con segmentazione/paginazione

Due processi che condividono un segmento in un sistema con segmentazione/paginazione.

© Deitel & Ass. Inc.

Esempio: architettura di memoria virtuale IA-32 Intel

- Architettura IA-32 Intel supporta sia la segmentazione pura e segmentazione/paginazione della memoria virtuale
- Spazio di indirizzamento logico
  - Insieme di indirizzi contenuti in ogni segmento
- Segmenti
  - Situato in qualsiasi posizione disponibile nello spazio di indirizzamento lineare del sistema
- traduzione degli indirizzi del segmento
  - Eseguita con mapping diretto che utilizza registri del processore ad alta velocità per memorizzare la tabella dei segmenti all'inizio registrata nel descrittore di tabella globale (Tabella Descrittore Globale GDT) o locale (Tabella Descrittore Locale LDT)
- Paginazione
  - Supporta tabelle delle pagine più livelli e formati pagina multiple