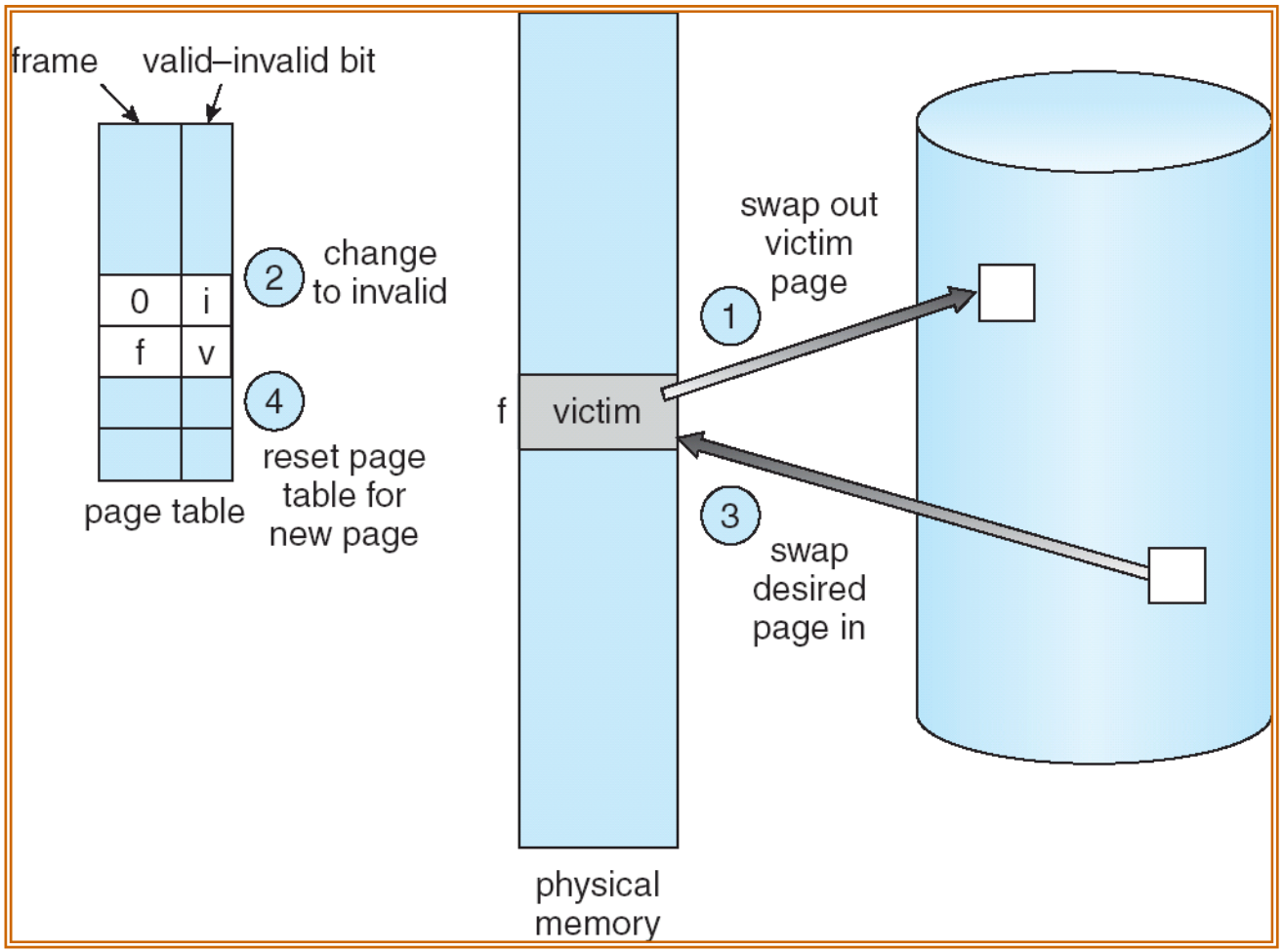
Основи на замена на страници

1. Најди ја локацијата на посакуваната страница на диск
2. Најди слободна рамка:
   - Ако има слободна рамка искористи ја
   - Ако нема, најди рамка жртва со користење на некој од алгоритмите за замена на страница
3. Прочитај ја посакуваната страница во ново-ослободената рамка. Ажурирај ги табелите на страници и рамки
4. Рестартирај го процесот

Која страница да се замени?

- При настанувањето на Page Fault треба се избере која страница да се отстрани за да се направи место за новата страница
- Променетите страници мора да се снимат на диск
- Алгоритам за оптимална замена на страници
  - Треба да се замени страницата која би била потребна најдалеку во иднината
  - Оптимално, но невозможно да се направи

Замена на страници



Алгоритми за замена на страници

- Not recently used – Не користен скоро
- First-In, First-Out
- Втора шанса
- Часовник (Clock)
- Least recently used – Најмалку користен скоро
- Работно множество (Working set)
- WSClock

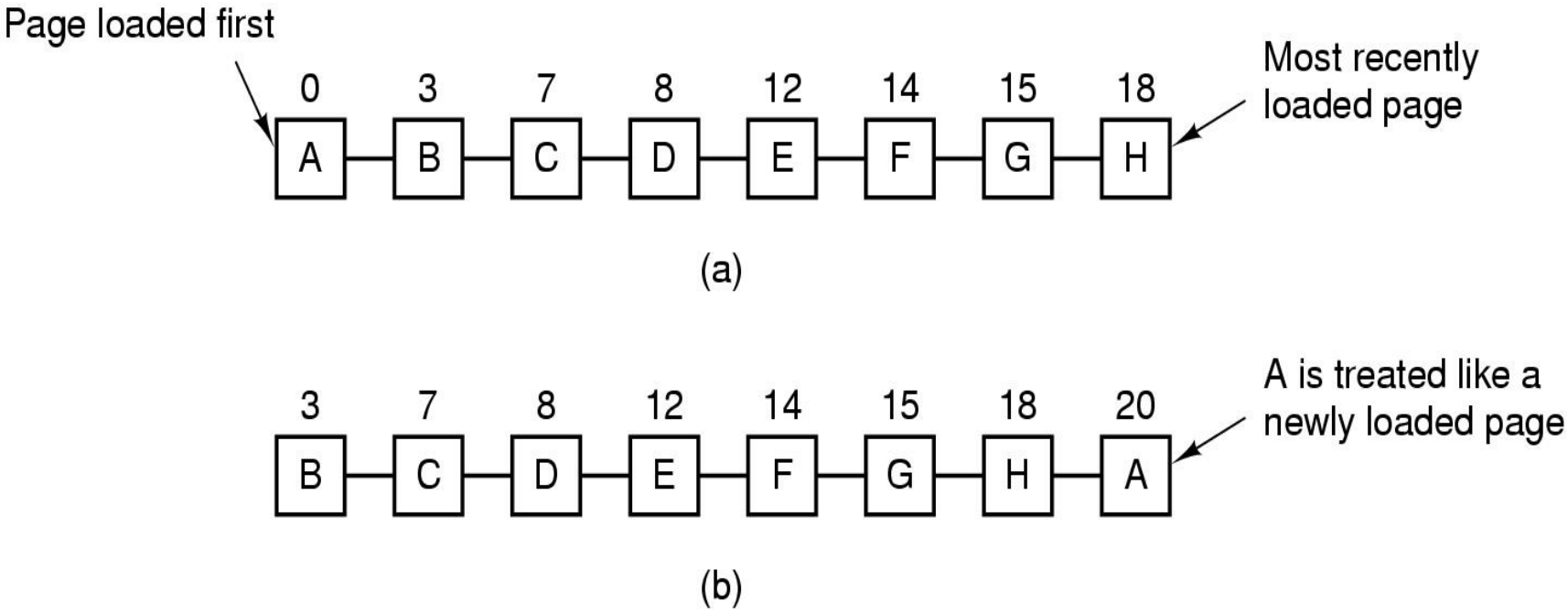
Алгоритам за замена на страници: Not Recently Used (NRU)

- За секоја страница има Reference bit и Modified bit
  - Битовите се поставуваат кога се пристапува или се модифицира станицата
  - Периодично овие битови се бришат
- Страниците се класифицираат во класи на следниов начин
  1. not referenced, not modified
  2. not referenced, modified
  3. referenced, not modified
  4. referenced, modified
- NRU ги отстранува страниците по случаен избор почнувајќи од најниската не празна класа

Алгоритам за замена на страници: FIFO

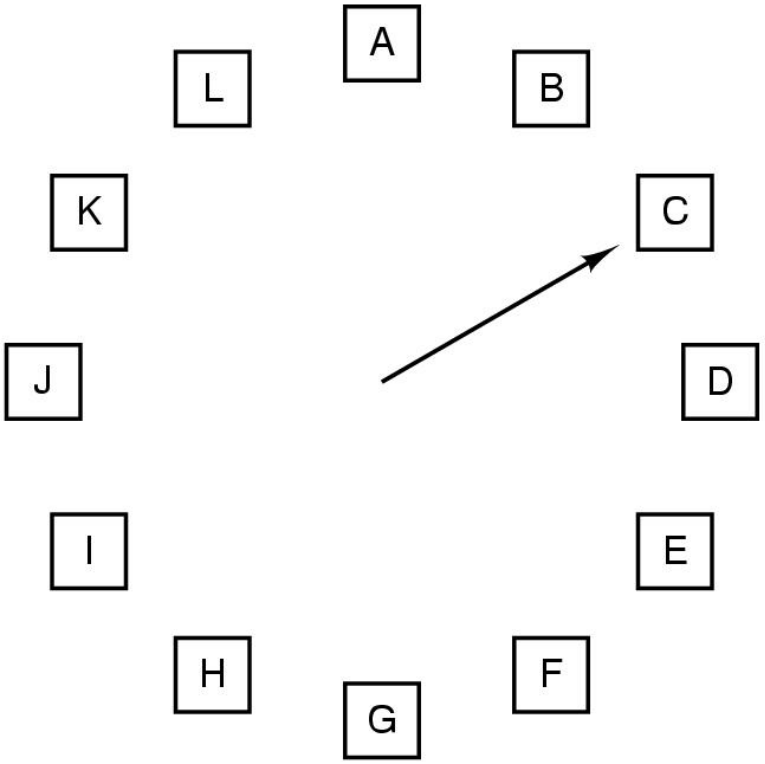
- Се чува поврзана листа од сите страници
  - Тие се подредуваат по редоследот по кој доаѓаат во меморијата
- Се отстранува страницата која е на почетокот на листата
- Недостаток
  - Страницата која е најдолго време во меморијата може да биде нај користена

Алгоритам за замена на страници: Втора шанса (Second Chance)



- Страниците се сортирани по FIFO
  - При настанувањето на Page fault се проверува дали R битот е поставен; ако е поставен страницата се става на крај и се прави обид да се отстрани наредната страница која има *R=0*

Алгоритам за замена на страници: Часовник (Clock)

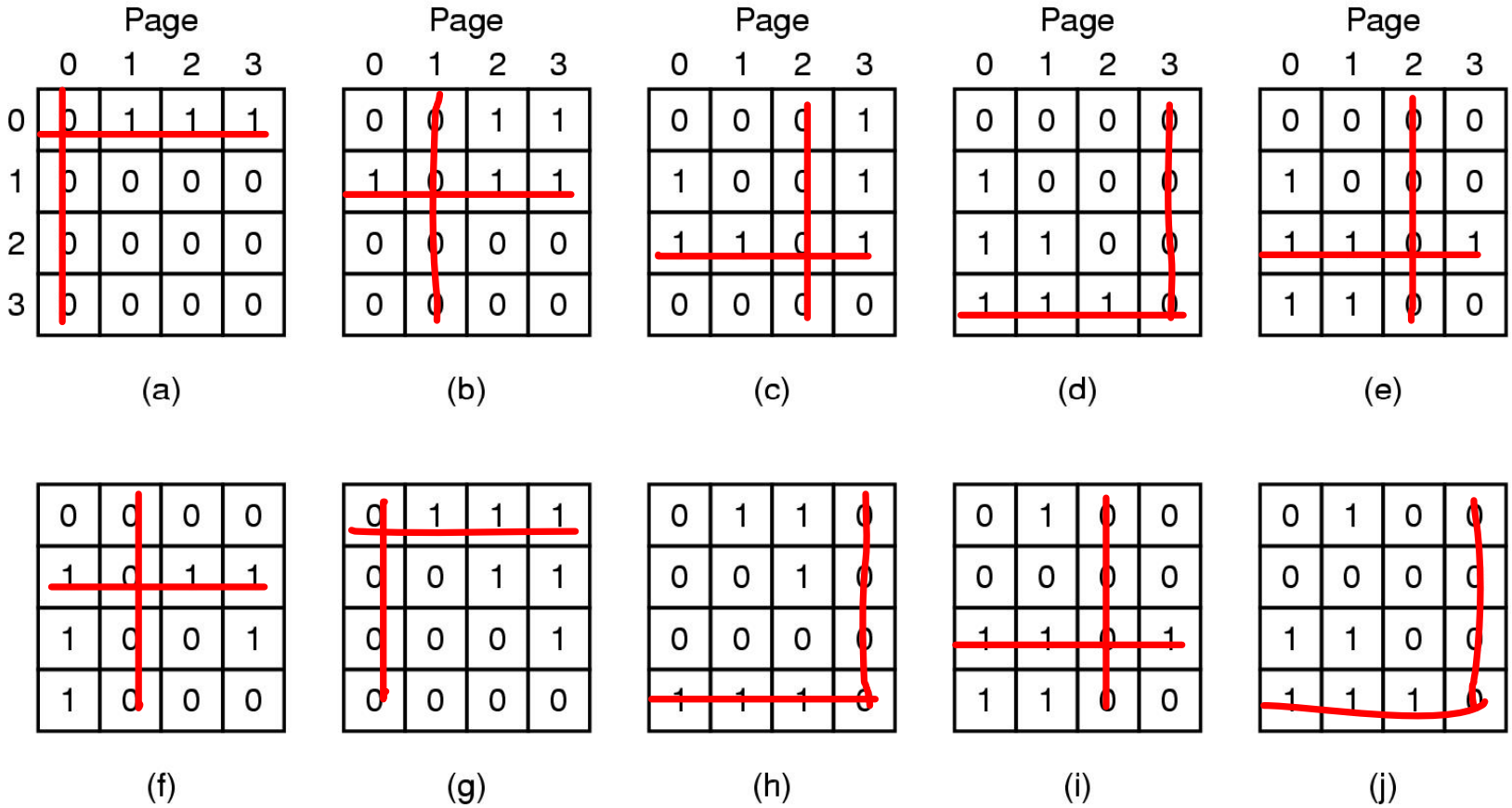


- Кога ќе се случи грешка (page fault) се проверува каде покажува последно. Која акција ќе се преземе зависи од R битот:
  - R=0 Отстрани ја страницата
  - R=1, Исчисти R, постави на 0 и оди на следната страница во кружната листа

Алгоритам за замена на страници: Least Recently Used (LRU)

- Се претпоставува дека станиците кои се користени скоро ќе се користат пак
  - Се отфрлаат страниците кои не се користени најдолго време
- Решение 1– неекономично
  - Да се чува сортирана листа од страници – најскоро достапените на почеток
  - Листата да се ажурира на секој пристап до меморијата
- Решение 2
  - За секоја страница се чува бројач (кој се зголемува при секоја инструкција) и се запишува вредноста кога пристапува до неа
  - Кога ќе се случи page fault, се гледа која страница има најмала вредност на бројачот

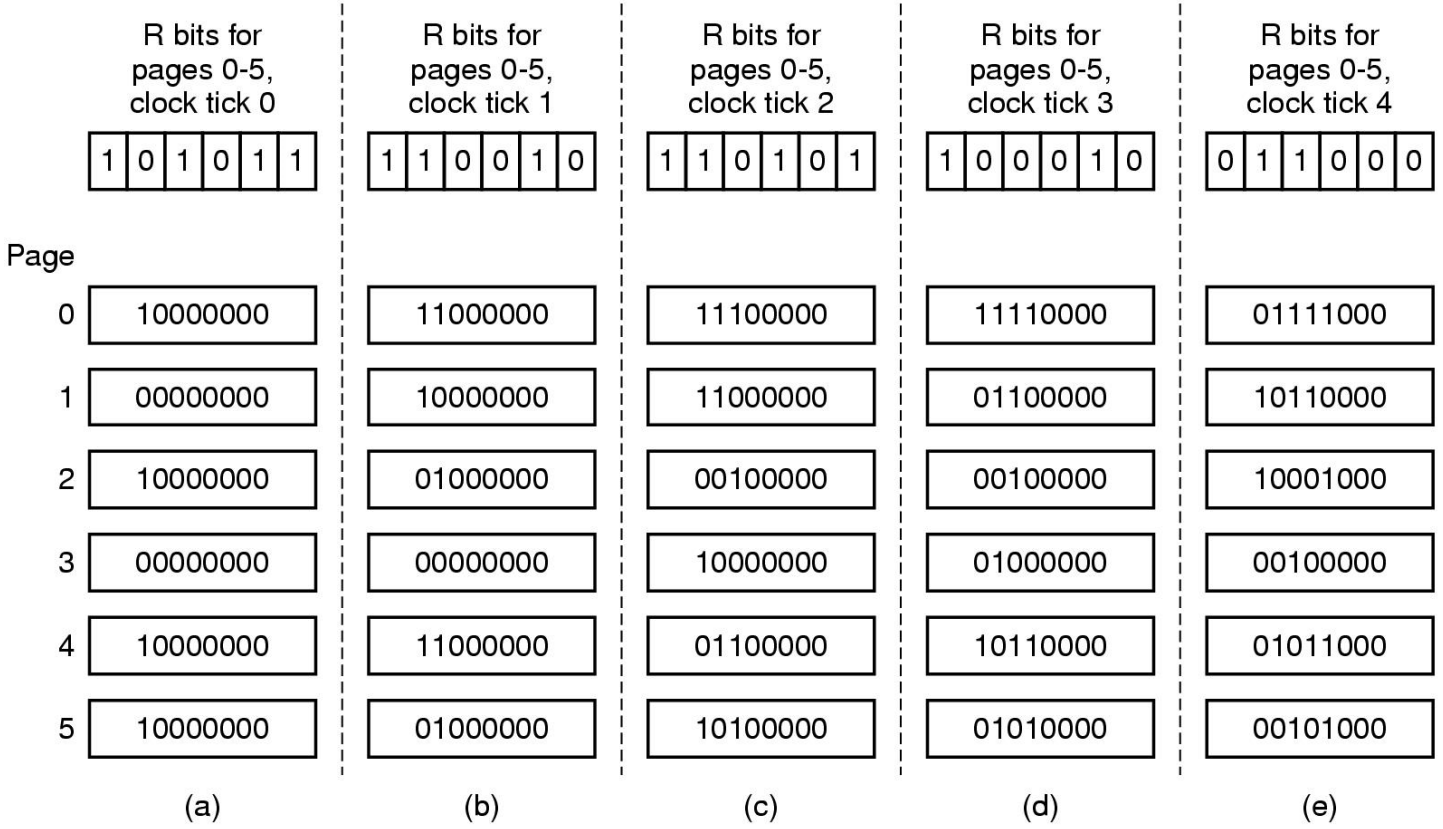
Решение 3: LRU со користење на бит матрици



Страниците се референцирани по редослед 0,1,2,3,2,1,0,3,2,3

Алгоритам за замена на страници:

Simulating LRU – Aging

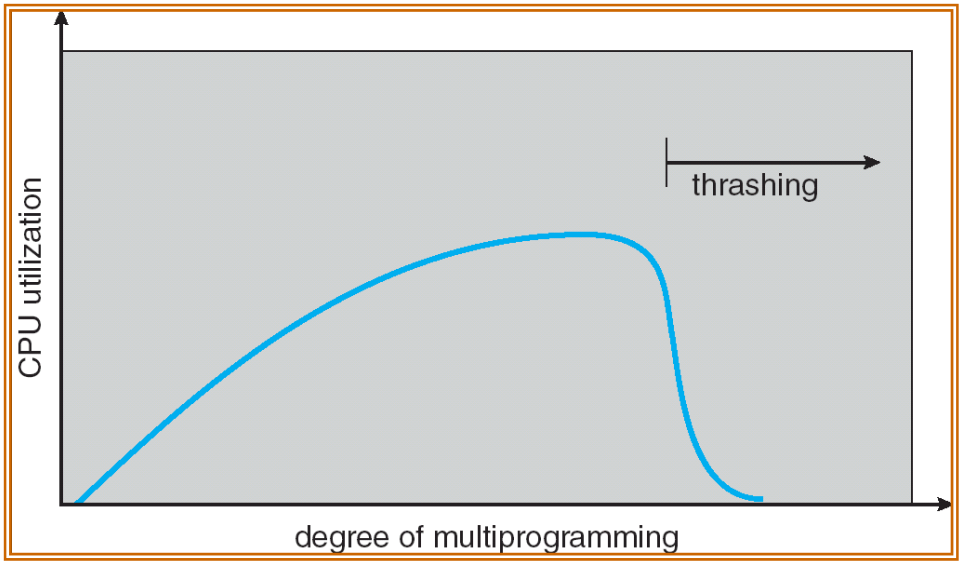


- Се мери временската староста на страниците
- Може да се имплементира и софтверски

Алгоритам Работно множество

- Почнуваме со page fault за првата страница со инструкции, па глобални променливи, стекови итн.
  - Стратегија страничење по барање (demand paging)
- Локалност на референцирање
  - Во секој момент процесот се обраќа само кон мал број на неговите страници
- Работно множество е множество страници кои процесот во еден момент ги користи

Thrashing



- Ако меморијата е мала да го зачува целото работно множество ќе имаме многу нови побарување и page faults – trashing

Страничење по барање и Thrashing

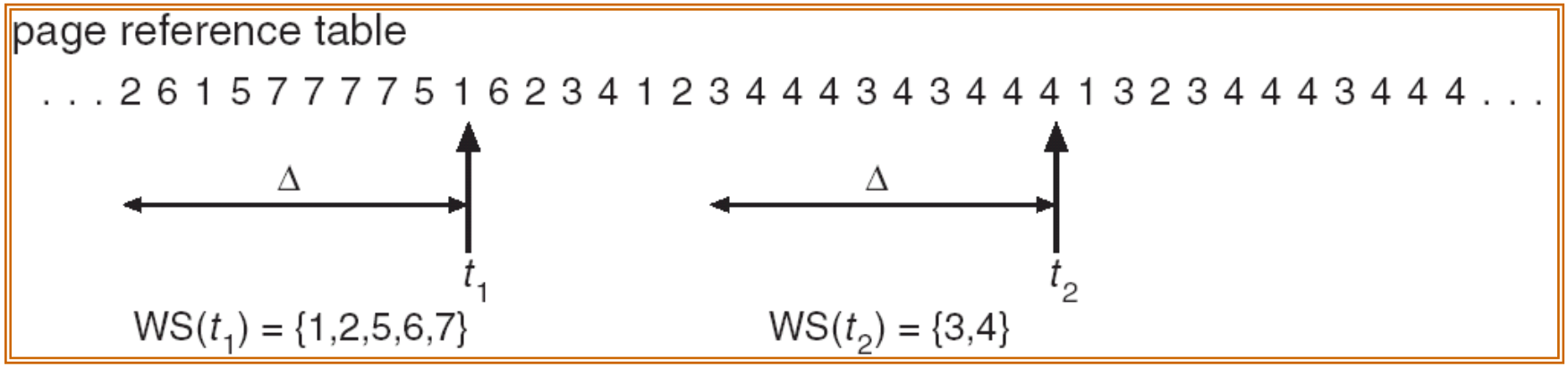
- Зошто страничењето по барање функционира?
  Заради моделот на локалност
  - Процесот мигрира од една локалност (страници тековно барани од процесот) во друга

- Зошто се случува thrashing?
  Σ големината на локалноста $>$ вкупната големина на меморијата

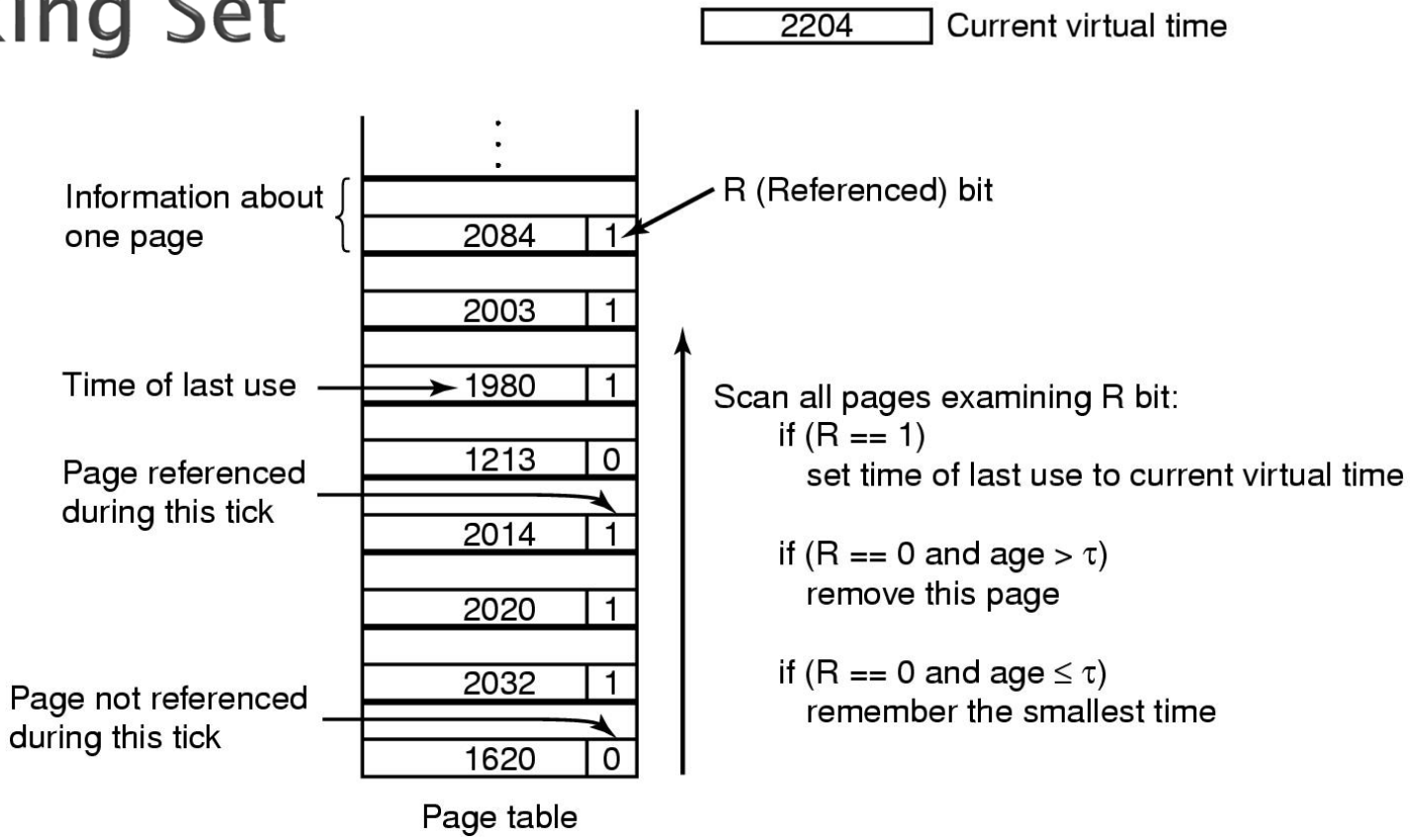
Модел на работно множество

- $\Delta \equiv$ прозорец на работно множество $\equiv$ фиксен број на референци кон страници
  Пример: 10,000 инструкции
- WSS_i (working set of Process P_i) $=$
  вкупен број на страници референцирани во последниот Δ (се менува)
  - Ако Δ е премало нема да ја содржи целокупната локалност
  - Ако Δ е преголемо ќе содржи повеќе локалности
  - Ако $\Delta = \infty \Rightarrow$ ќе го содржи целиот програм
- $D = \Sigma\ WSS_i \equiv$ вкупен број на побарани рамки
- Ако $D > m$ (вкупен број рамки) $\Rightarrow$ Thrashing
- Политика ако $D >$ m, суспендирај еден од процесите

Модел на работно множество



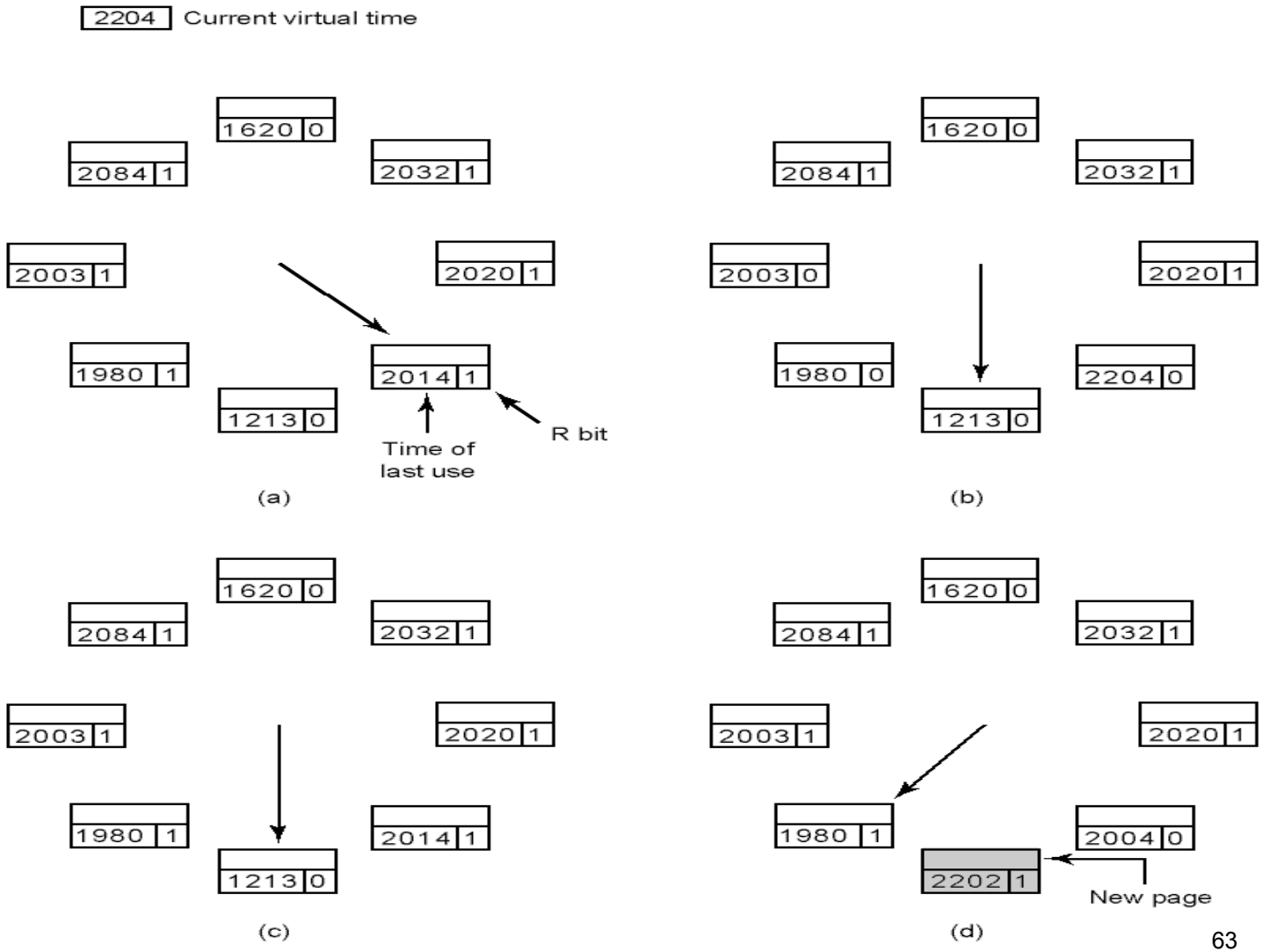
Алгоритам за замена на страници: Working Set



- Имплементација со проверка на времето на старост на страниците
- Се проверуваат сите времиња за да се одреди која страница да се отстрани

Алгоритам за замена на страници: WSClock

- Се отстранува првата страница која го задоволува старосниот критериум



Споредба на алгоритамите за замена на страници

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

Дизајн елементи за системот за страничење

- Глобална и локална замена
- Контрола на оптоварувањето
- Големина на страниците
- Поделба на адресните простори за податоци и програми
- Споделени страници
- Споделени библиотеки

Дизајн на системот за замена на страници

- Локална замена
  - На секој процес му се отстапува фиксен простор во меморија
  - Процесот заменува некоја од неговите страници

- Глобална замена
  - Динамички алоцира замена меѓу сите процеси кои се извршуваат
  - Бројот на рамки доделени на секој процес е променлива категорија

Дизајн на системот за замена на страници
Глобална и локална замена

	Age
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

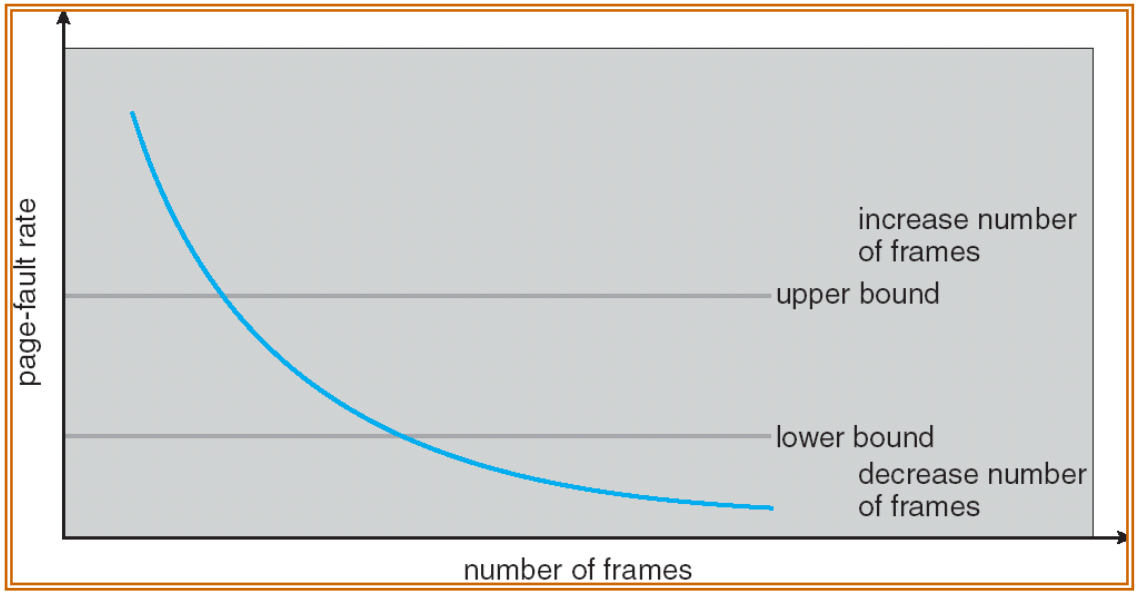
(b)

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

(c)

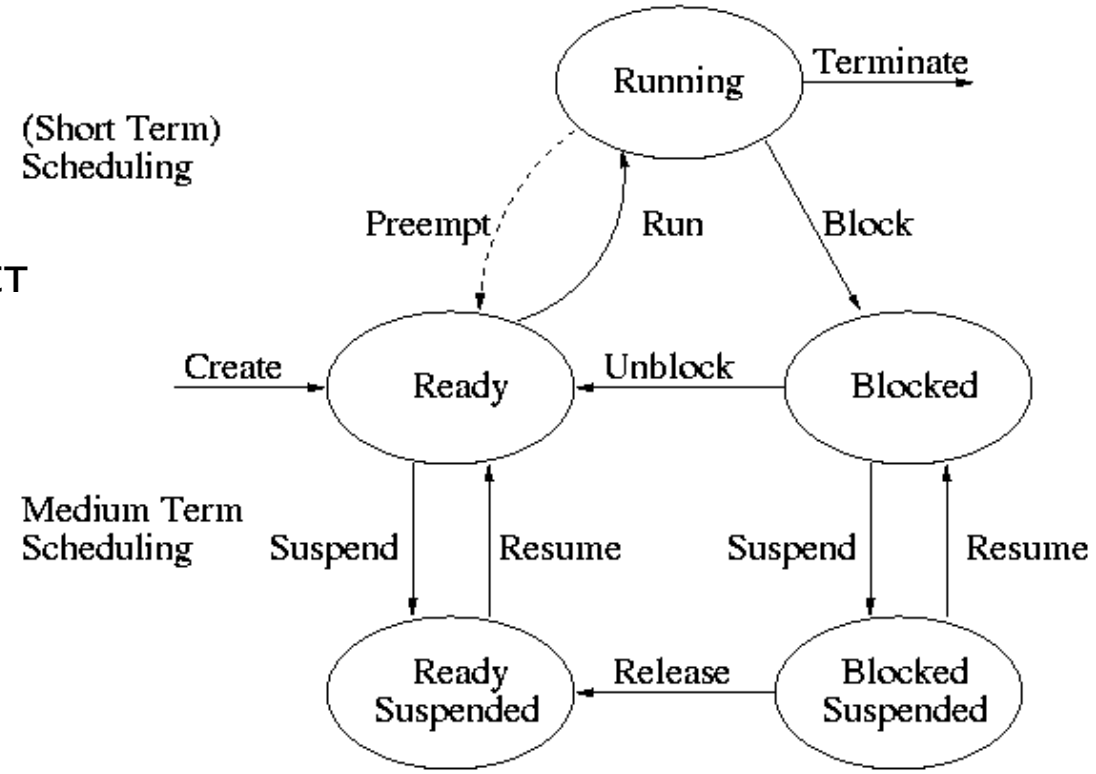
Шема на фреквенција на грешки во страничење

- Page Fault Frequency алгоритам – Се разгледува ратата на Page fault за секој процес
- Идејата е да се најде “прифатлива” рата на грешки
  - Ако е мала, на процесот треба да му се одземе рамка
  - Ако е голема, процесот добива рамка



Контрола на оптоварувањето

- Иако се добро дизајнирани системите може да се заглават во постојана замена на страници (thrash)
- PFF алгоритамот може да покажува дека
  - Некои процеси бараат меморија
  - Никој процес нема можност да ослободи меморија
- Решение:
  Да се намали бројот на процеси во меморијата

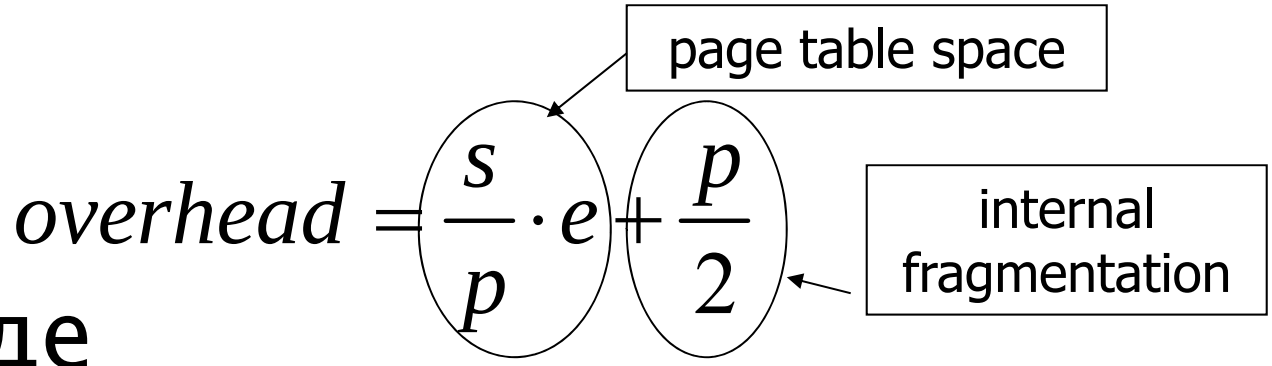


Големина на страниците

- Мали страници
- Предности
  - Помала внатрешна фрагментација
  - Помалку делови од програмите кои не се користат се во меморијата
- Недостатоци
  - Голем број на страници а со тоа и голема табела на станици

Големина на страниците

- Overhead за еден процес би бил

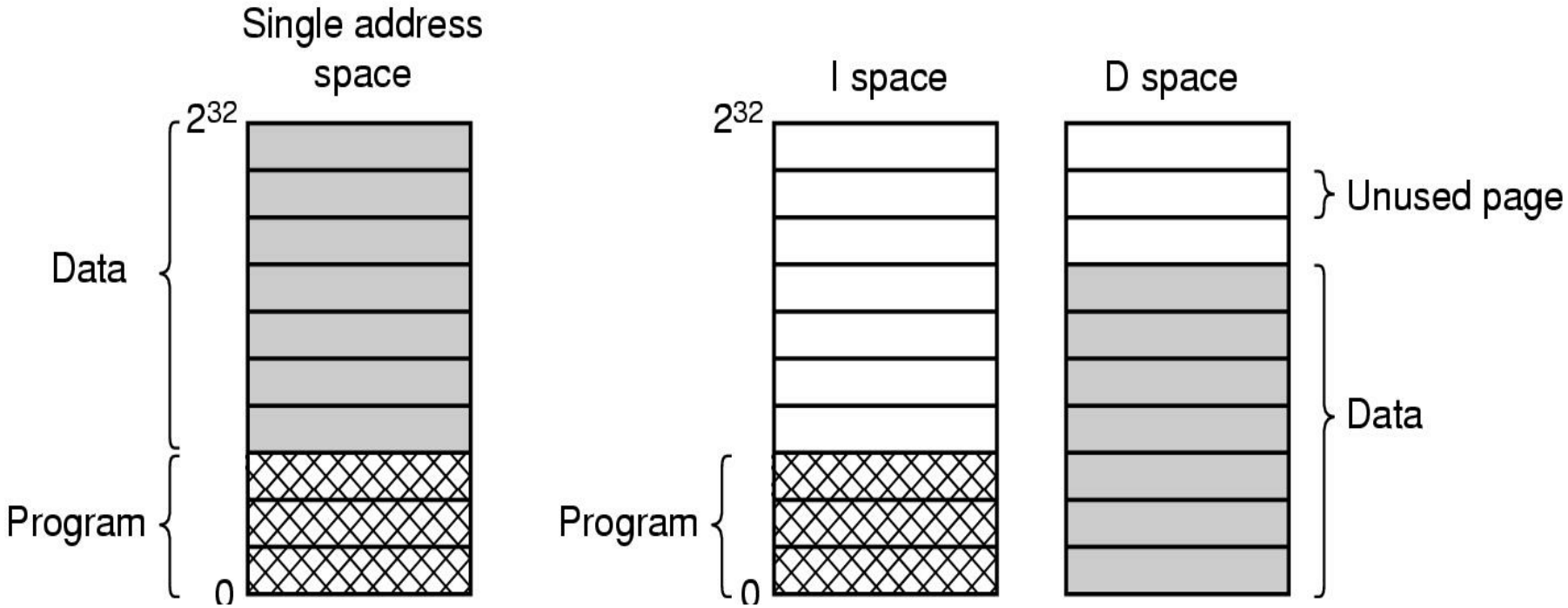


- Каде
  - s = големина на процесот во бајти
  - p = големина на страницата во бајти
  - e = големина на редот од табелата на страници во бајти

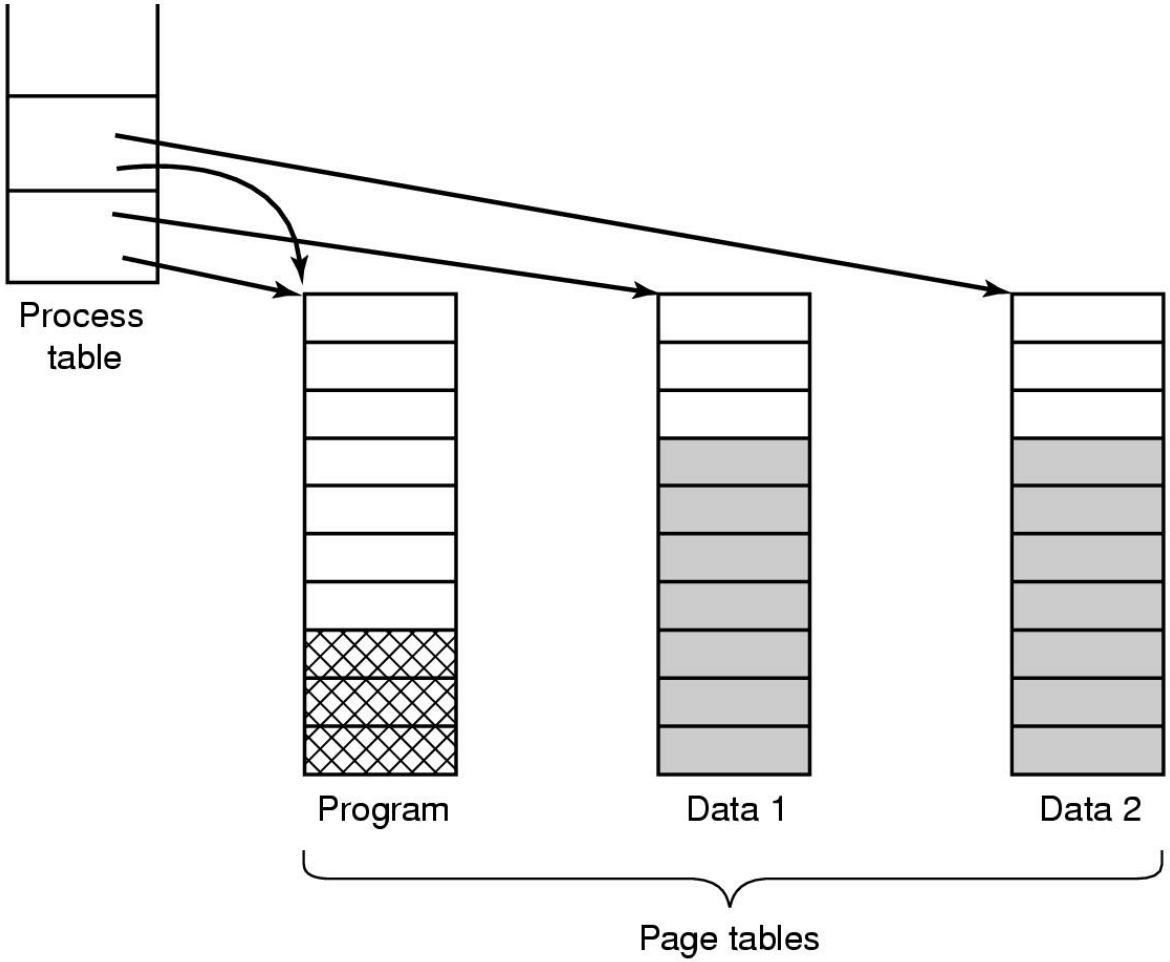
Оптимална вредност

$$p = \sqrt{2se}$$

Поделба на адресните простори за податоци и програми



Деливи страници



Повеке процеси користат исти страници

Shared Libraries

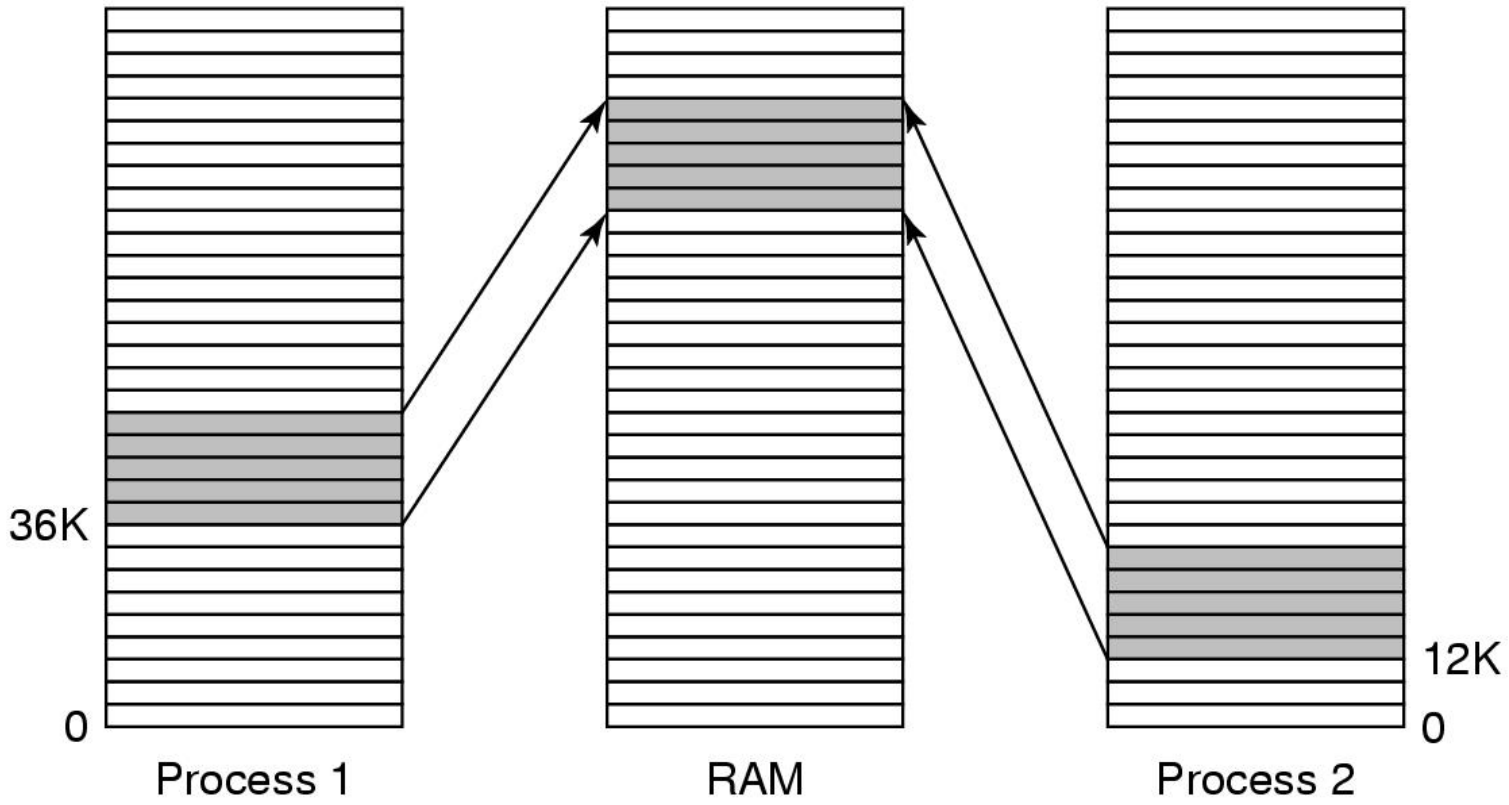


Figure 3-27. A shared library being used by two processes.

Имплементација на страничењето

Моменти на контакт со страничењето

1. Создавање на процес
   - Одреди големина на програм и податоци
   - Создади табела на страници
2. Извршување на процес
   - MMU се ресетира за нов процес
   - TLB flushed, табелата на страници на новиот процес се зема за тековна
3. Page fault time
   - Одреди ја виртуелната адреса која создава грешка
   - Замени ја одредената страница надвор, потребната внатре
4. Завршување на процес
   - Ослободи табела на страници, страници

Справување со Page Fault (1)

- Хардверот прави trap до јадрото, зачувувајќи го PC на стек.
- Се стартува асемблерска рутина за зачувување на регистрите за општа намена и другите променети информации.
- ОС детектира дека настанал page fault и се труди да види која виртуелна страница е потребна
- Откако ќе дознае за која виртуелна страница станува збор, ОС проверува дали адресата е валидна и дали заштитата е конзистенстна со достапот

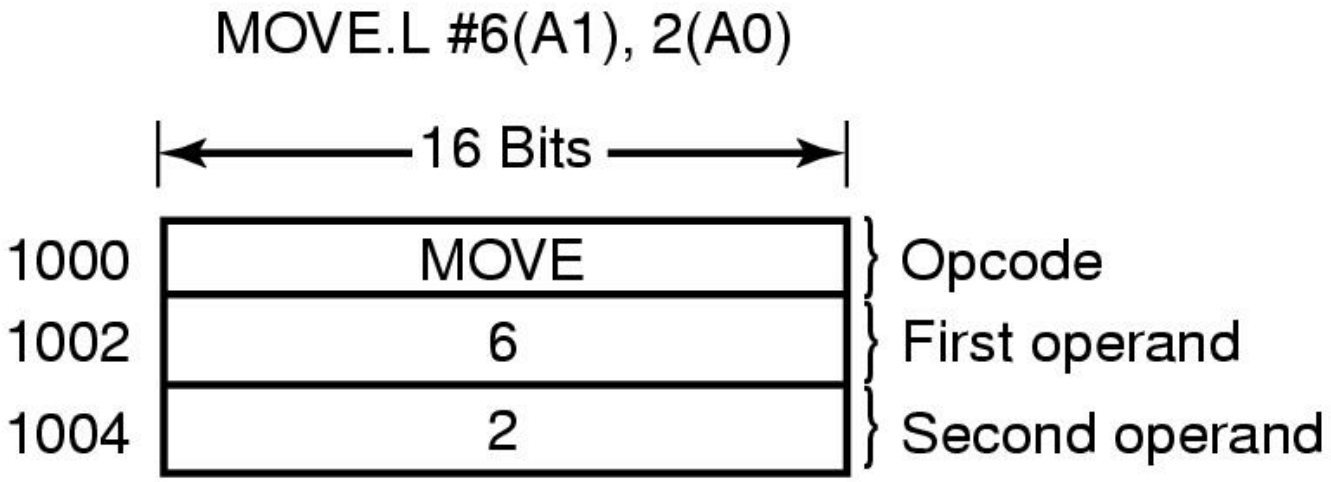
Справување со Page Fault (2)

- Ако рамката е модифицирана, страницата треба да се запише на диск и да настане промена на контекст
- Кога рамката не е модифицирана, ОС ја наоѓа диск адресата на бараната страница и закажува операција за читање од диск.
- Кога диск прекинот ќе укаже дека страницата е префрлена, табелата на страници се ажурира, рамката се означува дека е во нормална состојба

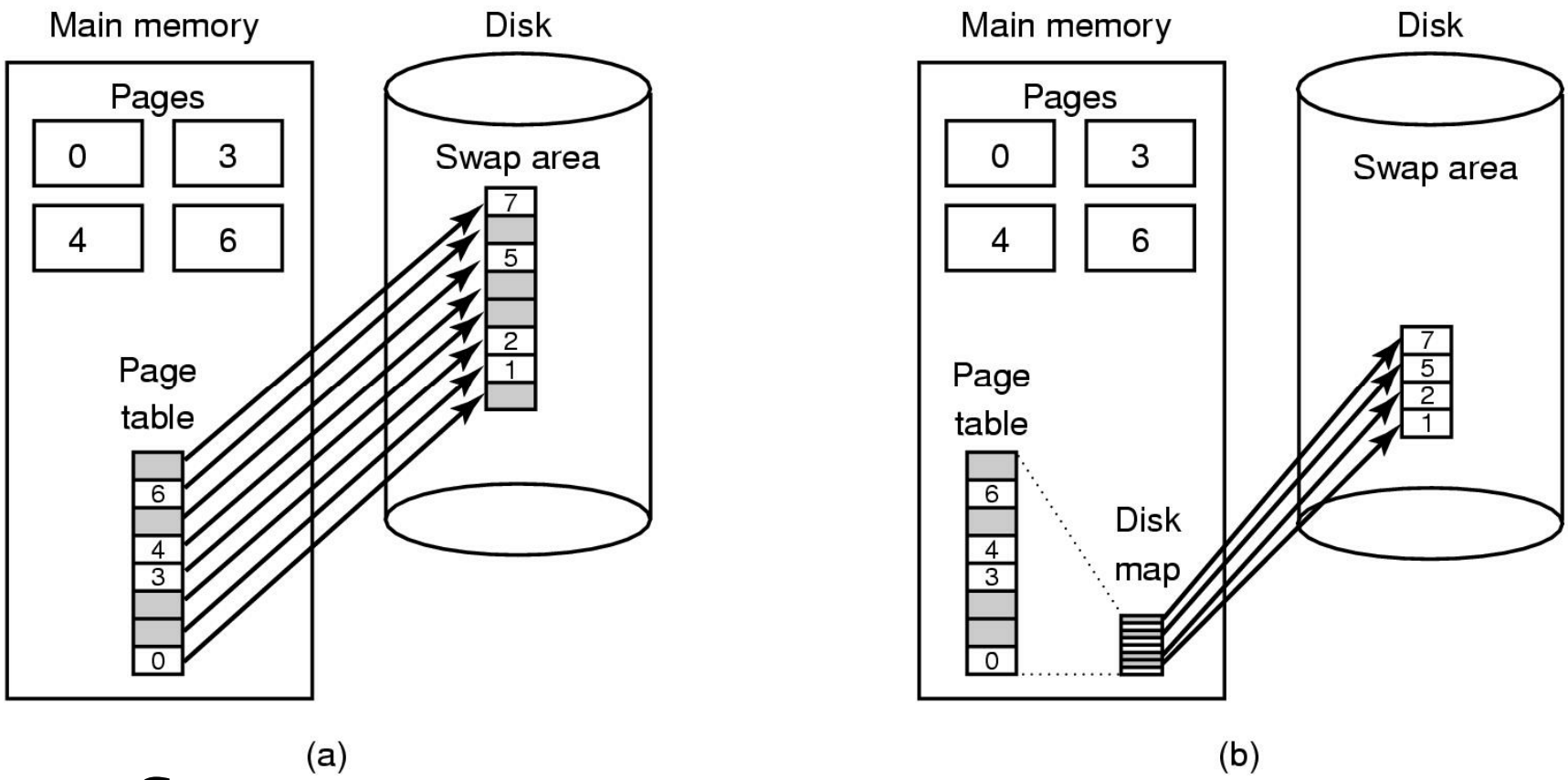
Справување со Page Fault (3)

- Инструкцијата која предизвикала грешка се враќа во состојбата пред прекинот и PC се поставува да покажува на таа инструкција.
- Процесот кој предизвикал грешка се распределува, а ОС се враќа на асемблерската рутина која го повика
- Оваа рутина ги запишува регистрите и другите информации за состојбата и врќа назад до корисничкиот простор за да продолжи извршувањето, како да не настанала page fault

Зачувување на инструкциите кои направиле page fault



Чување на страниците на диск



(a) Страничење со статички простор за замена

(b) Зачувување на страници динамички