Управување со меморија

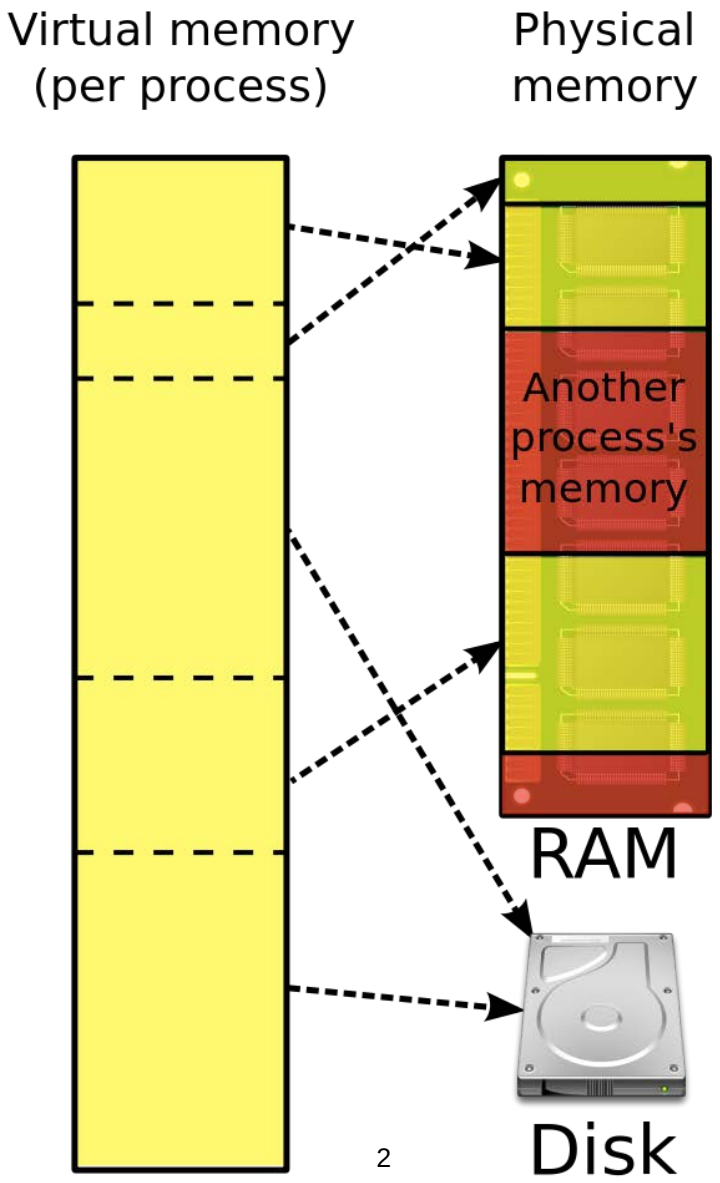
Оперативни системи 2024

проф. д-р Димитар Трајанов,
проф. д-р Невена Ацковска,
проф. д-р Боро Јакимовски,
проф д-р Весна Димитрова,
проф. д-р Игор Мишковски,
проф. д-р Сашо Граматиков,
вонр. проф. д-р Милош Јовановиќ,
вонр. проф. д-р Ристе Стојанов,
доц. д-р Костадин Мишев

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Потреба од меморија

- Потреба од мемориски простор кој е поголем од физичката меморија



Основи на замена на страници

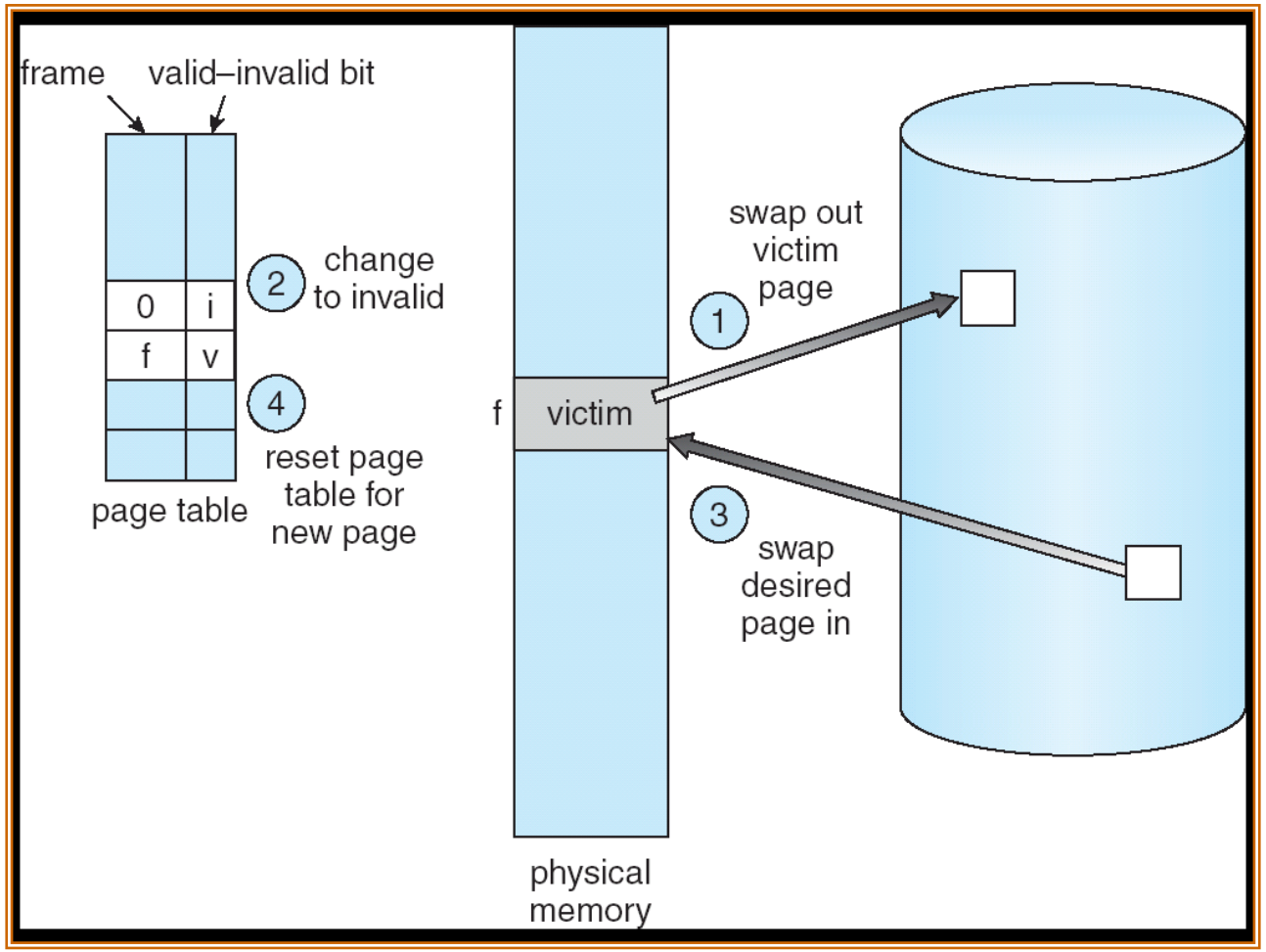
1. Најди ја локацијата на посакуваната страница на диск

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”
   - Процесот треба да продолжи од каде што настанал page fault

Замена на страници



Алгоритами за замена на страница

- 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

Алгоритми за замена на страници

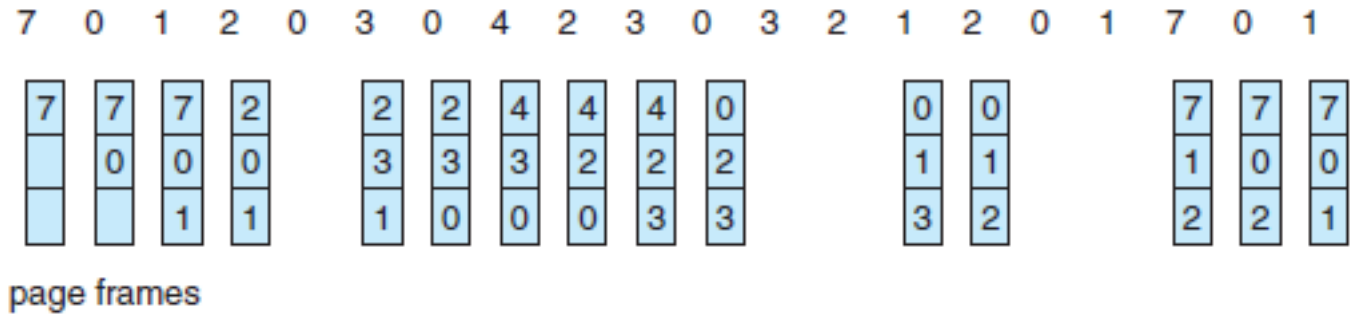
- Not recently used page replacement
- First-In, First-Out page replacement
- Second chance page replacement
- Clock page replacement
- Least recently used page replacement
- Working set page replacement
- WSClock page replacement

Not Recently Used (NRU)

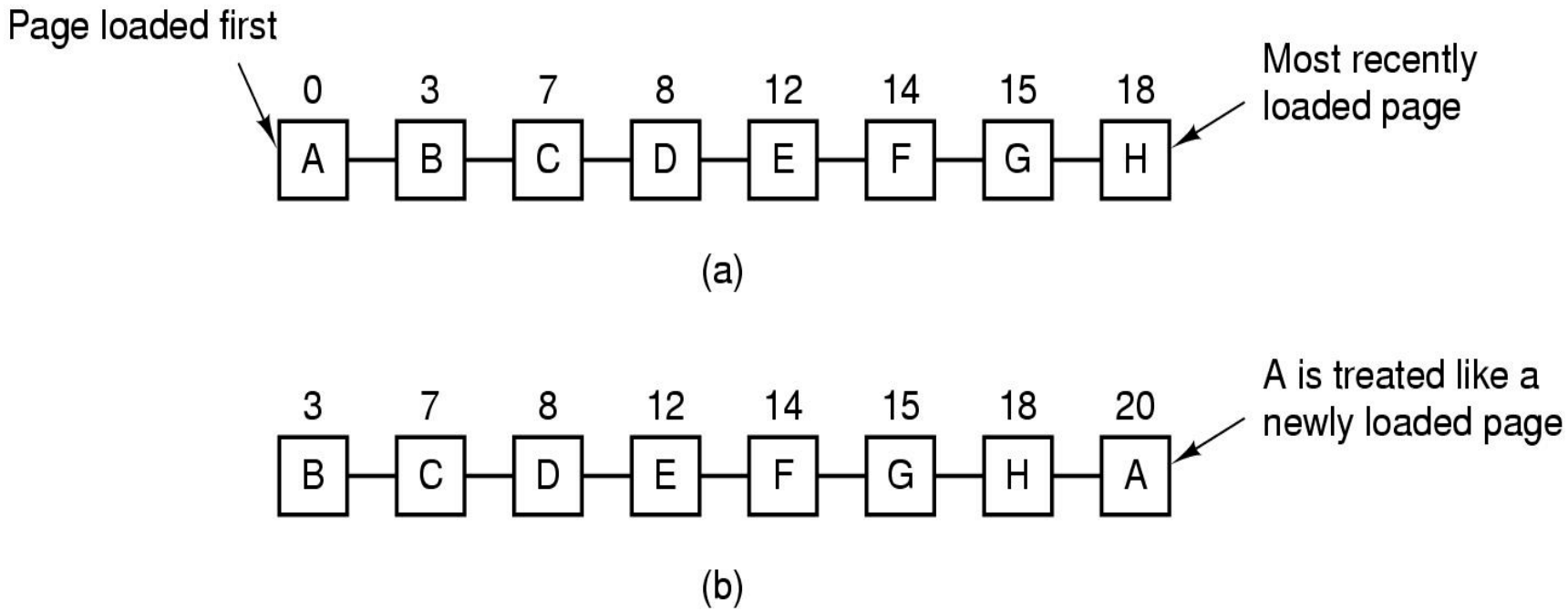
- За секоја страница има **Reference bit** и **Modified bit**
  - Битовите се поставуваат кога се пристапува или се модифицира станицата
  - Периодично овие битови се бришат
- Страниците се класифицираат во класи на следниов начин
  1. not referenced, not modified
  2. not referenced, modified
  3. referenced,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

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 најкористена

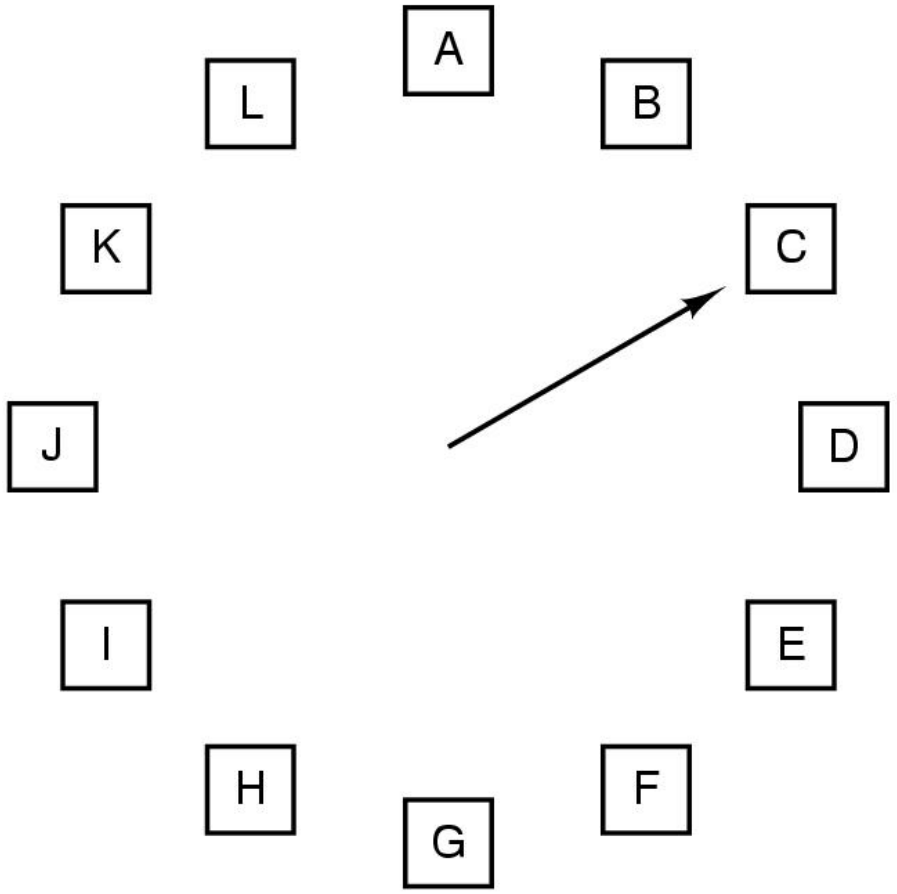


Second Chance



- 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*

Clock



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

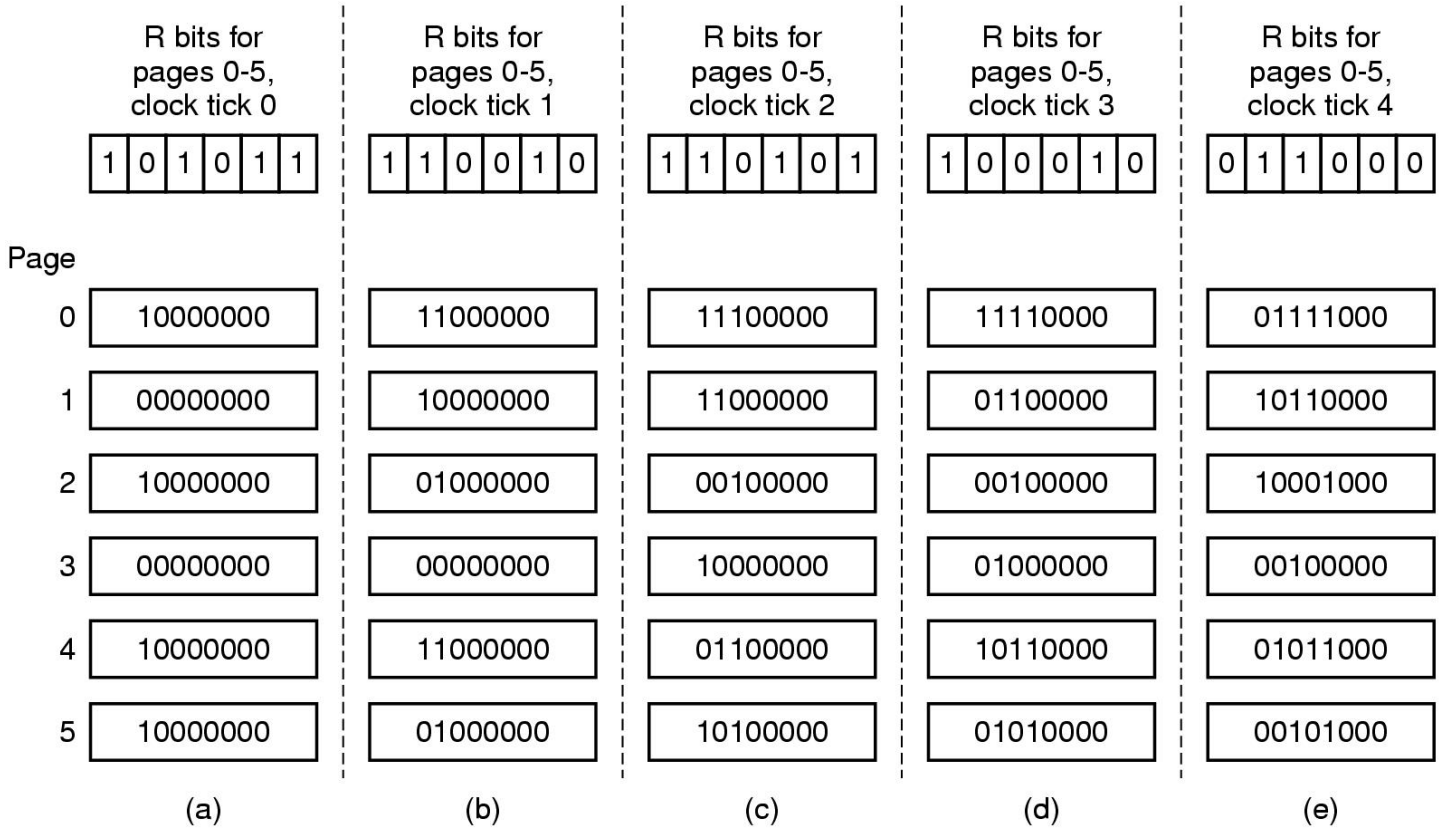
R = 0: Evict the page

R = 1: Clear R and advance hand

Least Recently Used (LRU)

- Се претпоставува дека станиците кои се користени неодамна ќе се користат пак
  - Се отфрлаат страниците кои не се користени најдолго време
- Решение 1
  - Да се чува сортирана листа од страници врз основа на бројот на пристапи до нив
  - Листата да се ажурира на секој пристап до меморијата
- Решение 2
  - За секоја страница се чува бројач кој се зголемува при секој пристап до неа
  - Периодично се ресетира бројачот

Simulating LRU - Aging



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

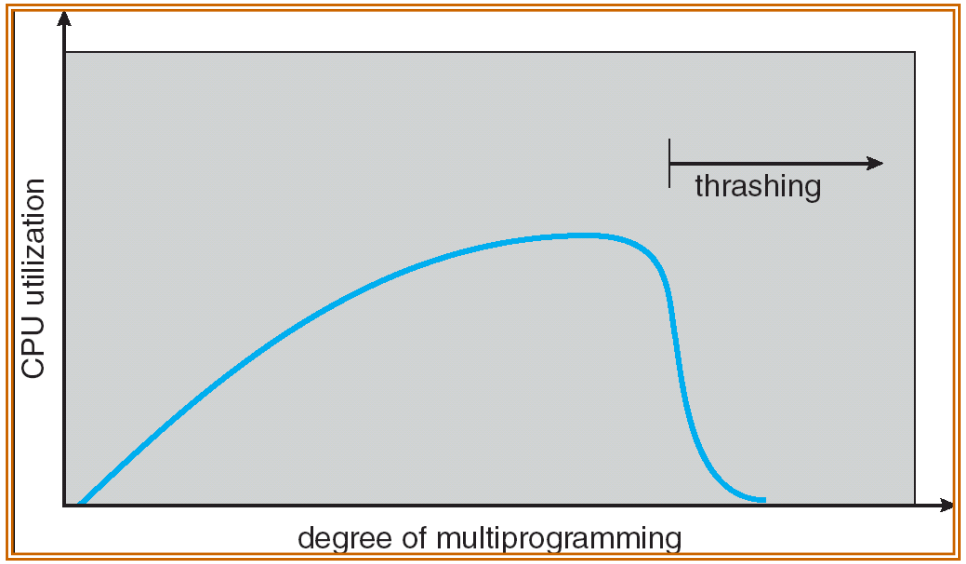
Работно множество

- 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

Модел на работно множество

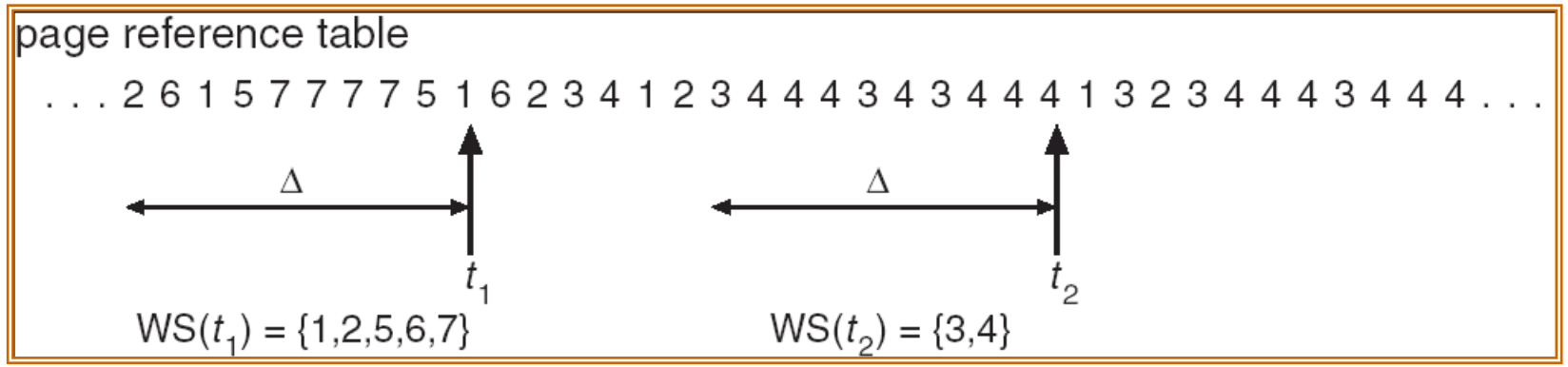
- $\Delta \equiv$ прозорец на работно множество $\equiv$ фиксен број на референци кон страници
  Пример: 10,000 инструкции
- WSS_i (working set of Process P_i) =
  вкупен број на страници референцирани во последниот Δ (се менува)
  - Ако Δ е премало нема да ја содржи целокупната локалност
  - Ако Δ е 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

Thrashing



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

Модел на работно множество

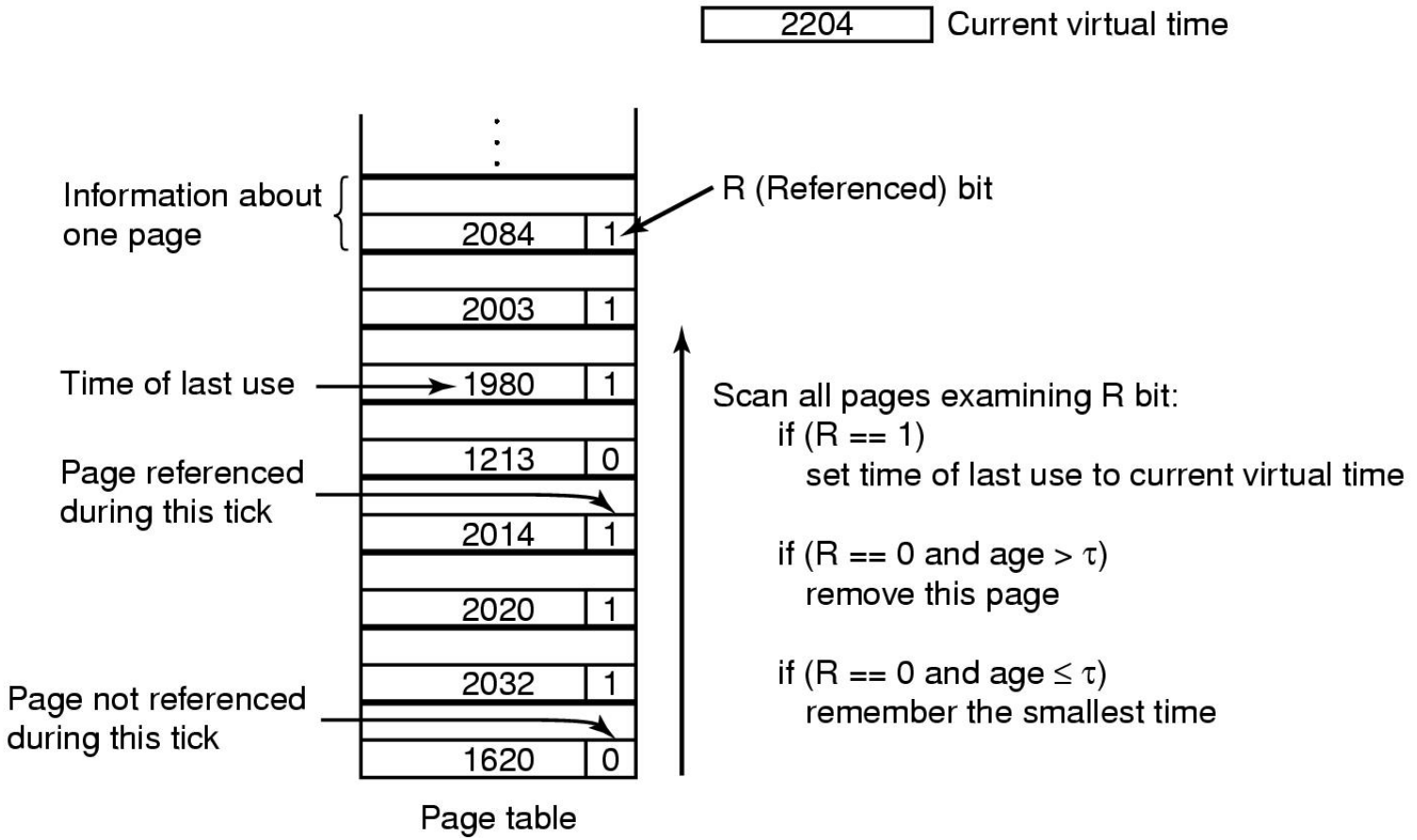


Страничење по барање и Thrashing

- 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**

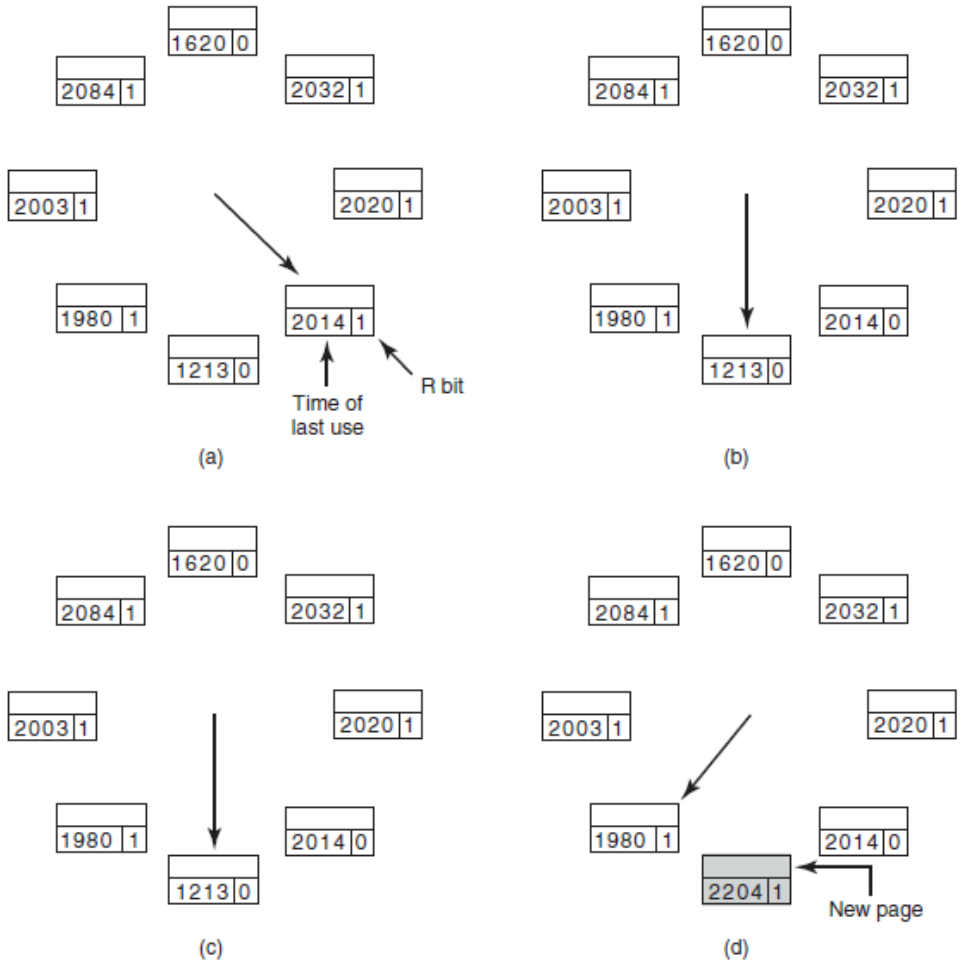
Working Set



Имплементација со проверка на времето на старост на страниците

Се проверуваат сите времиња за да се одреди која страница да се отстрани

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

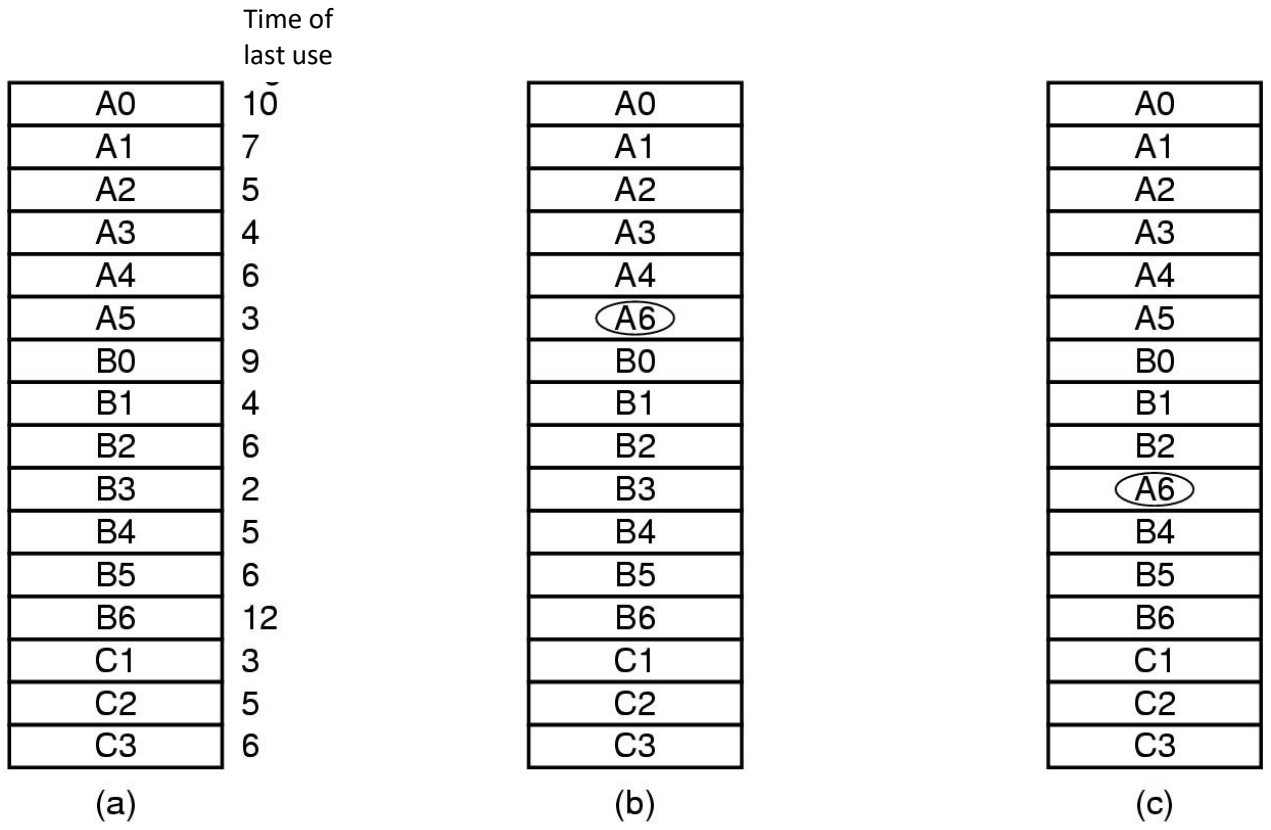
Дизајн елементи за системот за страничење

- Глобална и локална замена
- Контрола на оптоварувањето
- Големина на страниците
- Поделба на адресните простори за податоци и прогрми
- 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

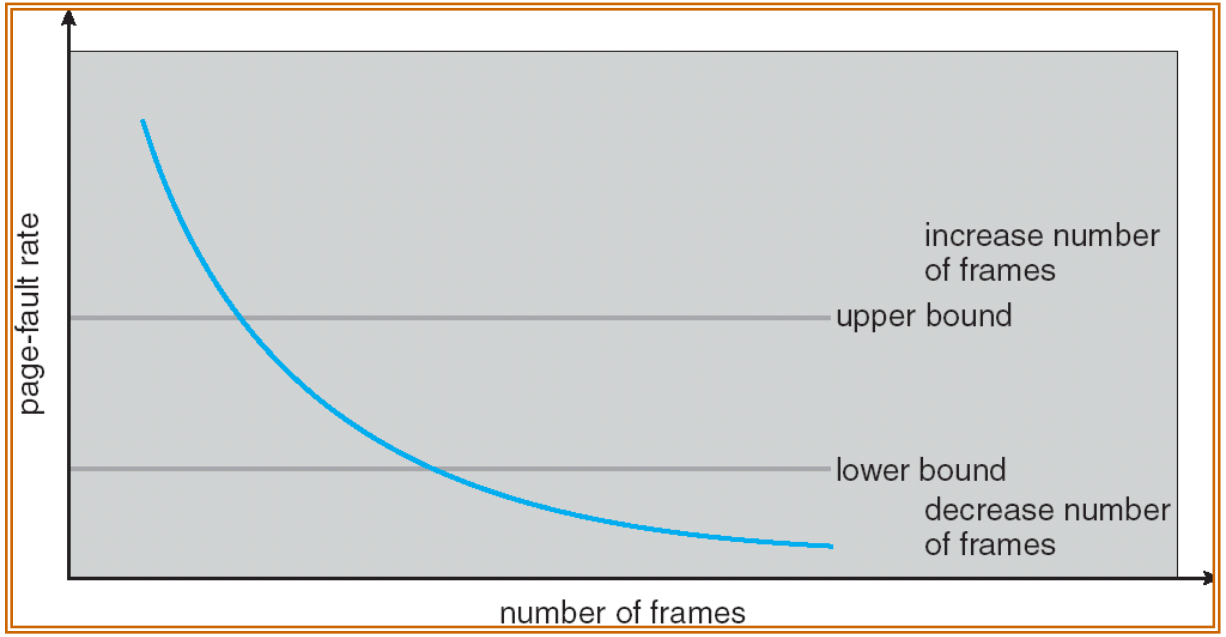


Локална замена

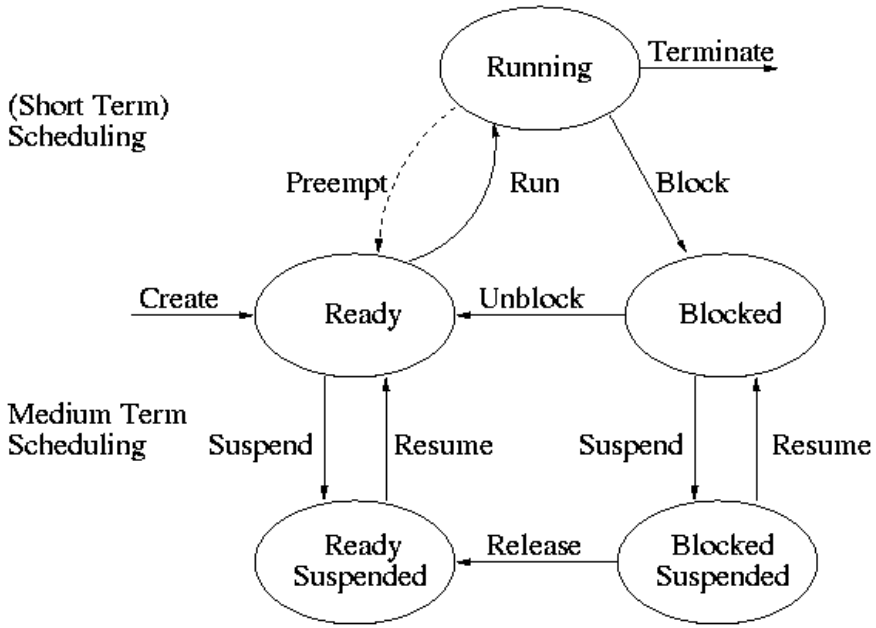
Глобална замена

Шема на фреквенција на грешки во страничење

- Page Fault Frequency алгоритам - Се разгледува стапката на промашување за секој процес
- Идејата е да се најде “прифатлива” стапка на промашување
  - 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 покажува дека
  - Некои процеси бараат меморија
  - Никој процес нема можност да ослободи меморија
- Решение:
  Да се намали бројот на процеси во меморијата

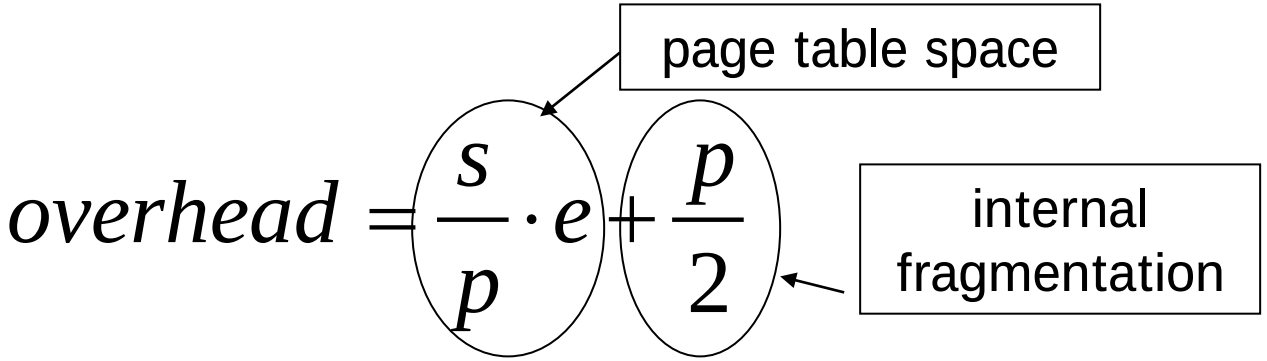
Големина на страниците

Мали страници

- Предности
  - Помала 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

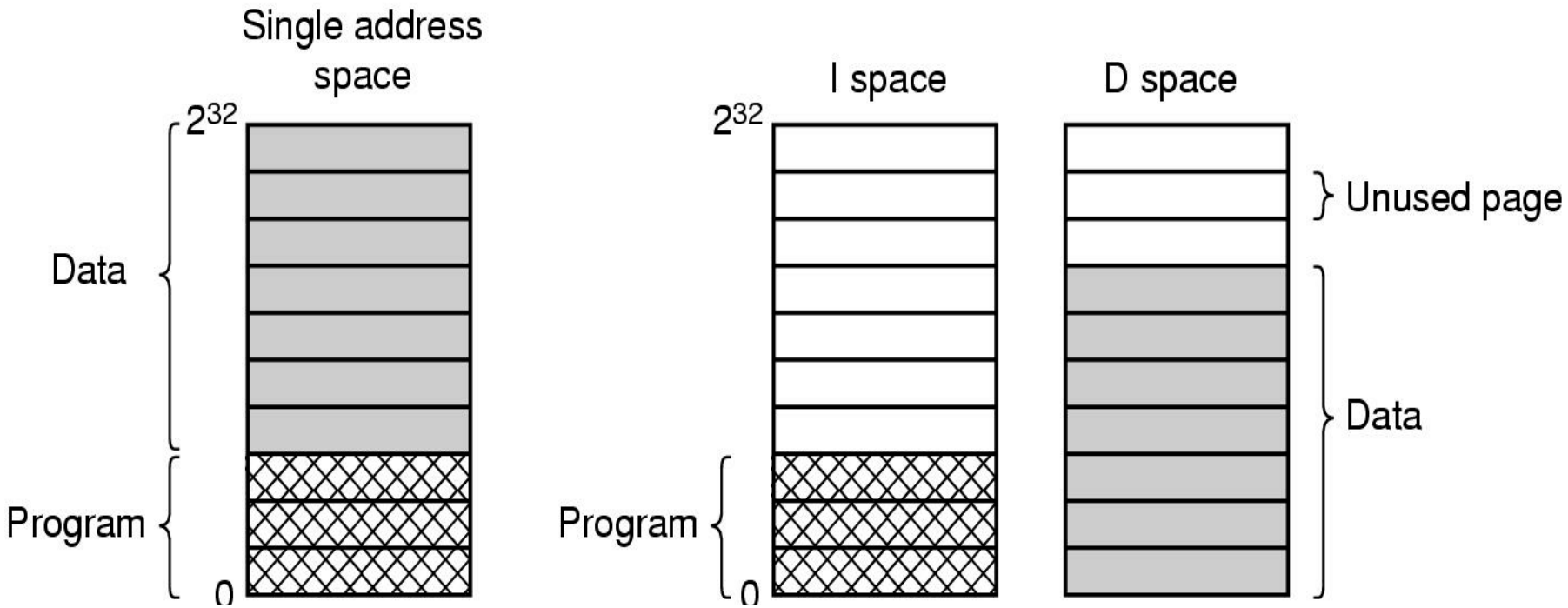


- Каде
  - s = големина на процесот во бајти
  - p = големина на страницата во бајти
  - e = големина на редот од табелата на страници во бајти

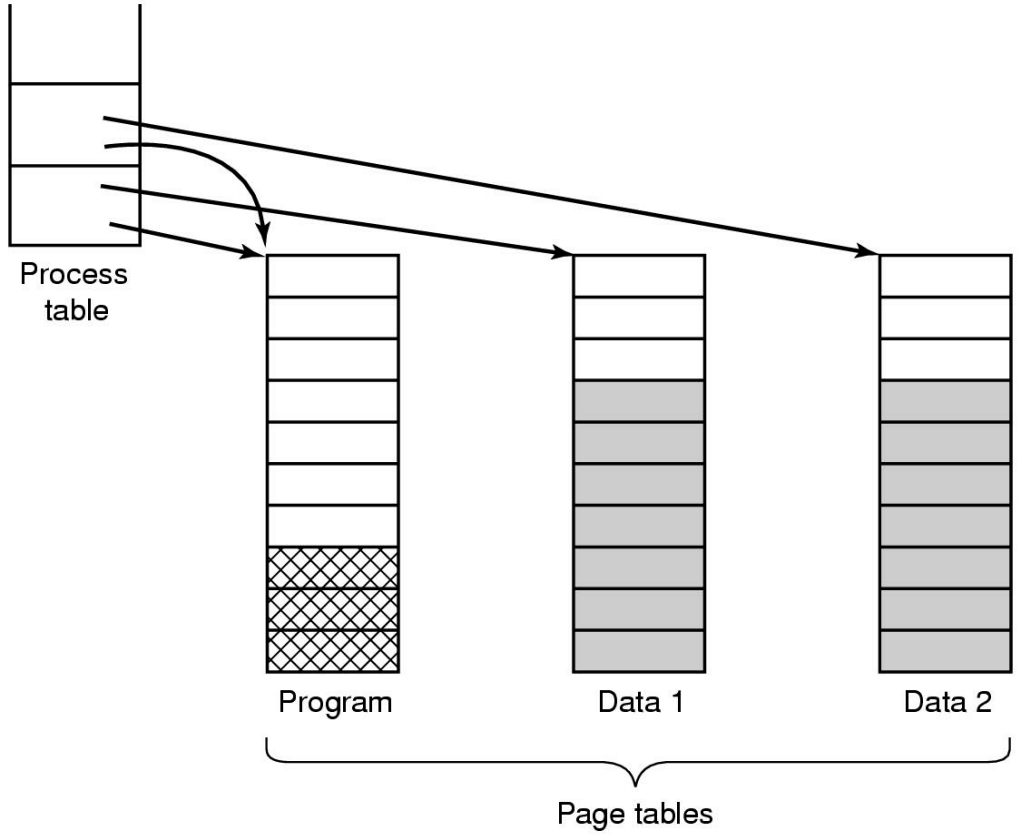
Оптимална вредност

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

Поделба на адресните простори за податоци и програми



Деливи страници



Повеќе процеси користат исти страници

Shared Libraries

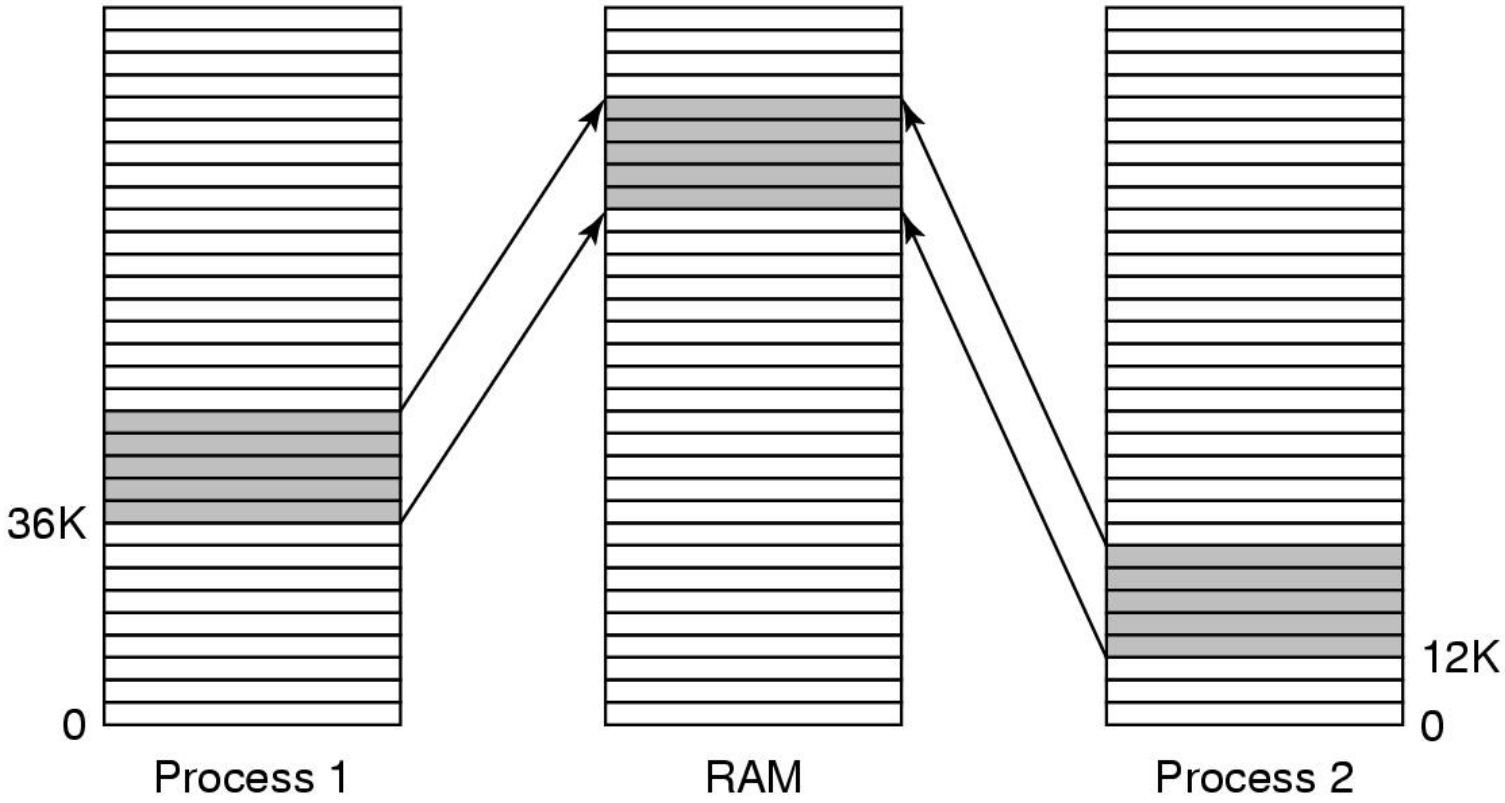


Figure 3-27. A shared library being used by two processes.

Имплементација на страничењето

Моменти на контакт со страничењето

1. Креирање на процес
   - Одреди ја големината на програмскиот и податочниот дел
   - Креирај табела на страници
2. Извршување на процес
   - Реиницијализирај ја MMU за нов процес
   - Исчисти го TLB
3. Грешка на страница
   - Определи ја виртуелната адреса која ја предизвикала грешката
   - Донеси ја потребната страна, исфрли ја непотребната (swap)
4. Терминирање на процес
   - Ослободи ја табелата на страници и страниците

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 конзистентна со достапот. Ако има слободна рамка, ја става страницата во слободната рамка, ако нема, почнува некој алгоритам за замена на страница.

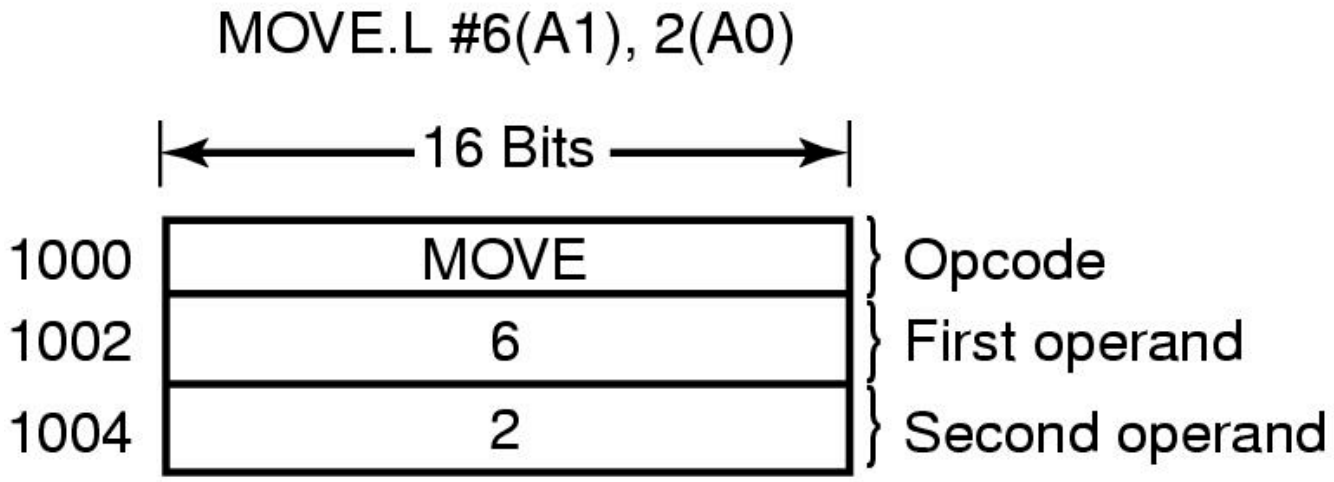
Справување со Page Fault (2)

- Ако рамката е модифицирана, страницата треба да се запише на диск и да настане промена на контекст
- Кога рамката е достапна (веднаш или откако е запишана на диск), 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

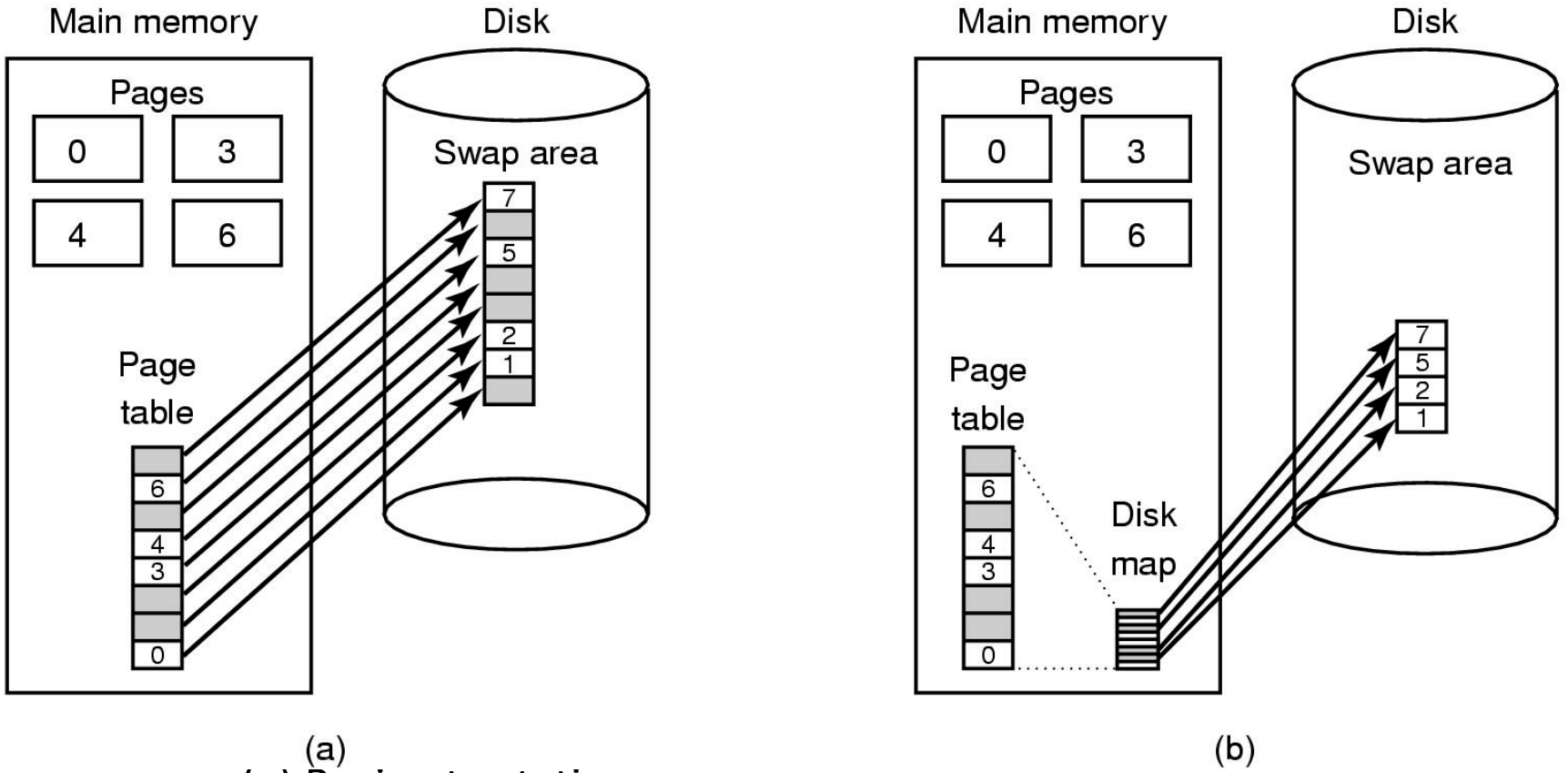
Справување со Page Fault (3)

- 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 враќа 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) Paging to static swap area

(b) Backing up pages dynamically