Virtuelna memorija

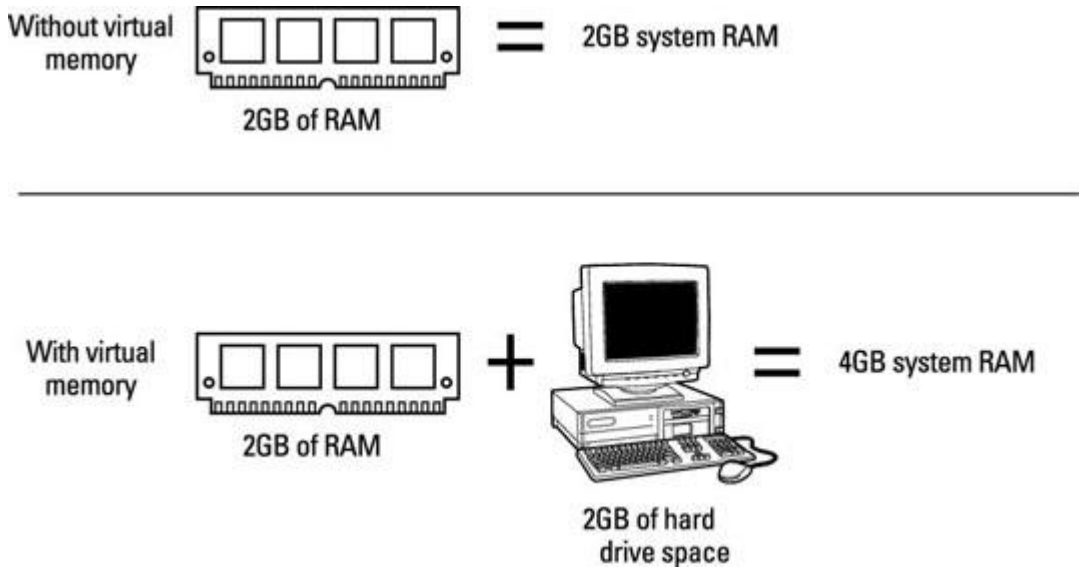
Nemanja Maček

Sadržaj

- Učitavanje stranica prema potrebi
- Alternativne tehnike učitavanja stranica
- Zamena stranica
- Raspodela okvira po procesima i efekat zasićenja

Šta je virtuelna memorija?

- Tehnika koja dozvoljava izvršavanje procesa čiji delovi mogu biti smešteni na diskovima.
- Virtuelna memorija:
  - Formira apstrakciju (logička memorija = operativna memorija + sekundarna memorija).
  - Omogućava izvršenje programa većih od same fizičke memorije.
  - Dozvoljava smeštaj znatno većeg broja procesa u memoriju (konkretno, delova procesa).

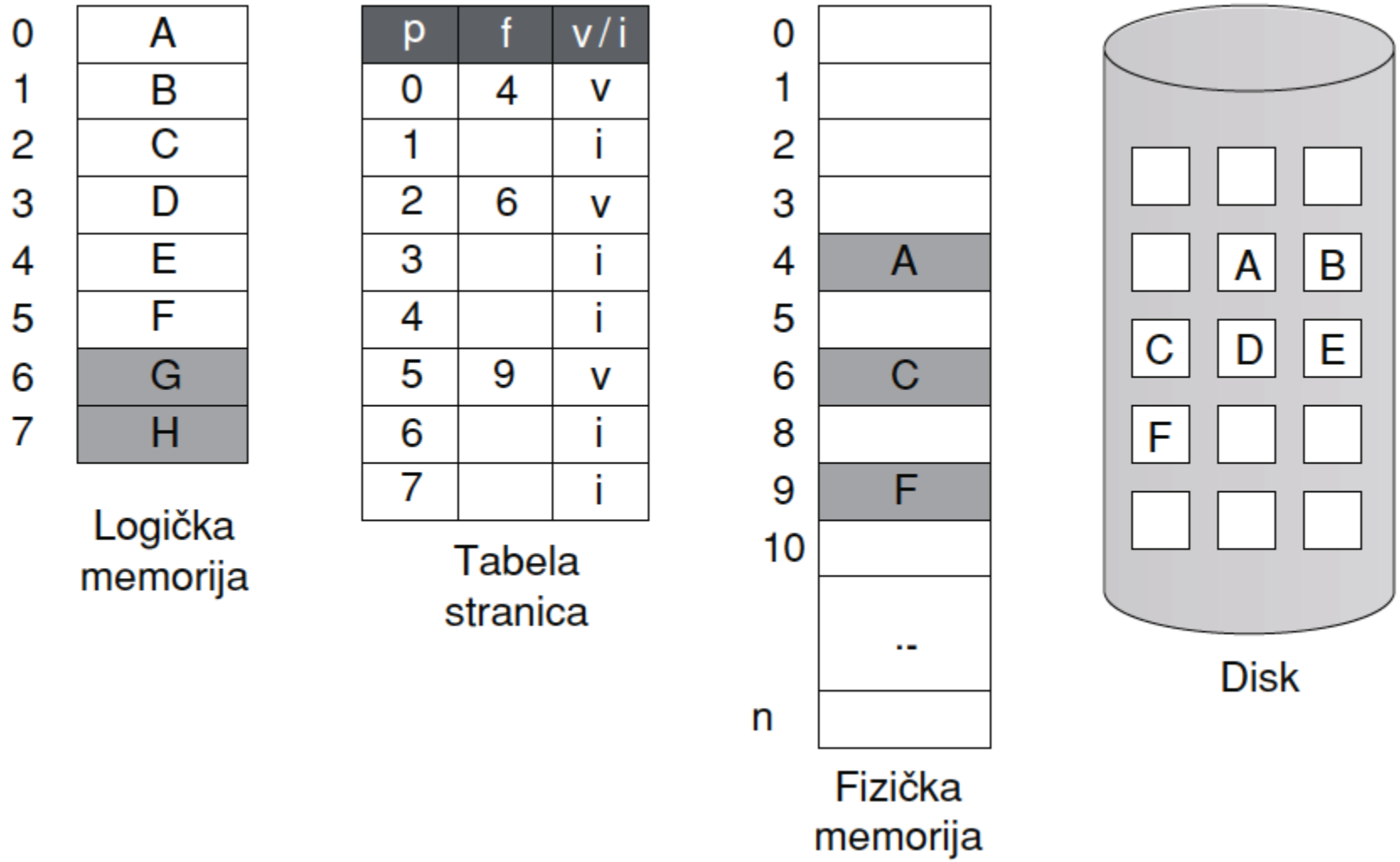


Učitavanje stranica prema potrebi

- Virtuelna memorija se najčešće realizuje tehnikom **učitavanja stranica prema potrebi**.
- DP (engl. *demand paging*) sistem podseća na straničenje sa razmenjivanjem.
- DP funkcioniše na sledeći način:
  - Memorija i prostor na disku koji se koristi za razmenjivanje su izdeljeni na stranice.
  - U fizičku memoriju se ne prebacuje ceo proces.
  - Prebacuju se samo stranice koje se trenutno traže.
    - Najčešće se prebacuje samo ona koja je neophodna.
    - Može prebaciti i više stranica u memoriju na osnovu pretpostavke o stranicama koje će biti potrebne procesu.
- Šta je prednost ovog sistema u odnosu na swap isključivo celih procesa?
  - Izbegava se nepotrebno čitanje sa diska.
  - Smanjuje se potrebna količina fizičke memorije.

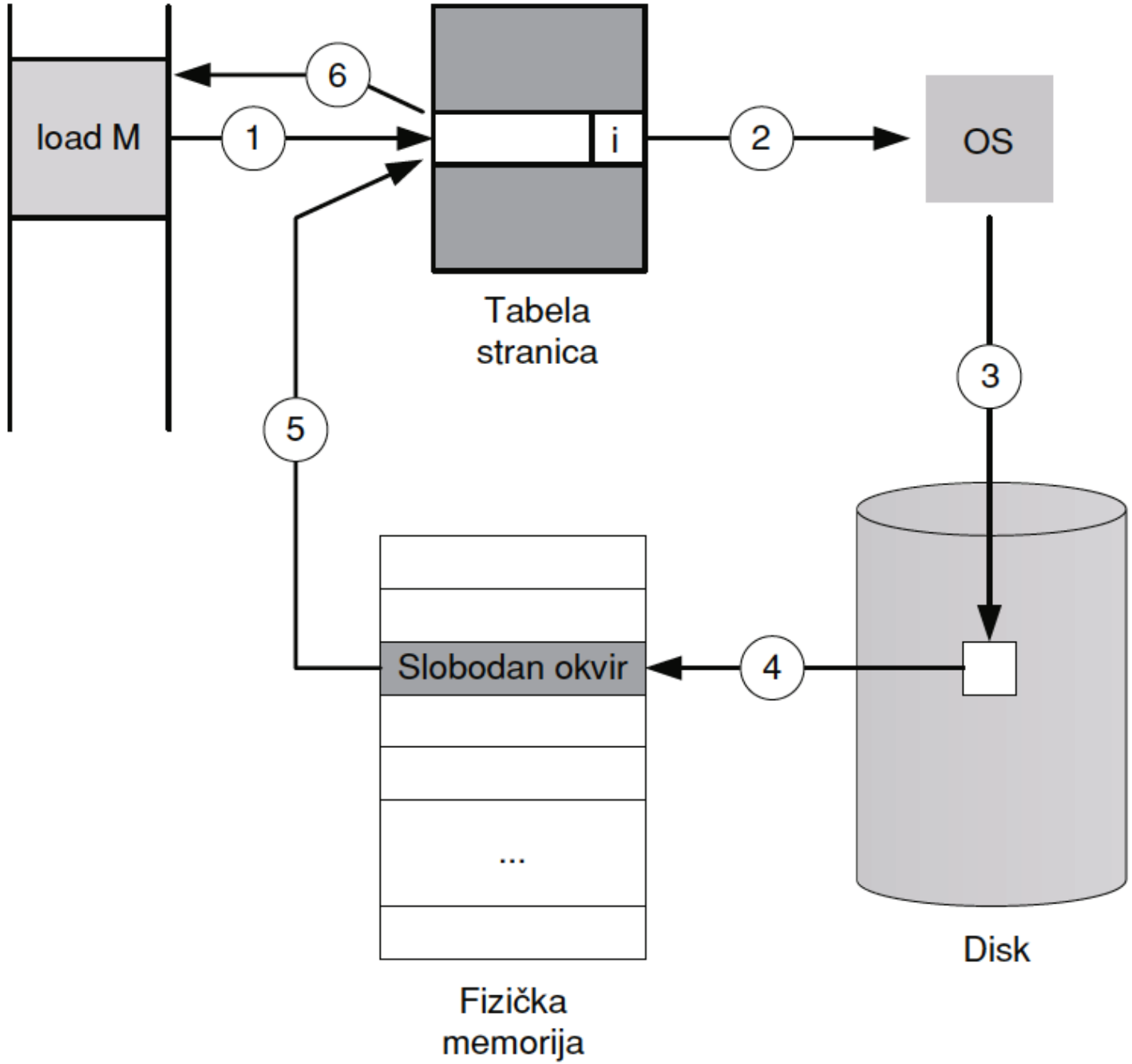
Koji je hardver neophodan za realizaciju DP tehnike?

- Unija hardvera potrebnog za straničenje i razmenjivanje.
- Sledeće dve komponente su apsolutno neophodne:
- **Tabela stranica**.
  - Eksplicitno se zahteva prisustvo **bita validnosti** u tabeli stranica.
  - Pomoću bita validnosti može se opisati **trenutni položaj** stranice.
    - Vrednost bita **v** (**valid**) ukazuje da se logička stranica nalazi u memoriji.
    - Vrednost bita **i** (**invalid**) ukazuje da se:
      - Stranica ne nalazi u memoriji već u swap prostoru na disku.
      - Takođe može označavati takođe da je to stranica koja ne pripada adresnom prostoru diska.
- **Sekundarna memorija**.
  - Služi za smeštaj svih stranica koje nisu prisutne u memoriji.
  - Uređaj koji se, po pravilu, koristi je disk (hard disk ili SSD).


0
1
2
3
4
5
6
7
A
B
C
D
E
F
G
H
Logička memorija
p
f
v/i
0 4 v
1 i
2 6 v
3 i
4 i
5 9 v
6 i
7 i
Tabela stranica
0
1
2
3
4
5
6
8
9
10
n
A
C
F
..
Fizička memorija
A
B
C
D
E
F
Disk

Prebacivanje stranica sa diska u fizičku memoriju

- Posmatrajte sledeće dve situacije:
  - Proces se izvršava ili pristipa stranici koja je **u memoriji**.
    - Kako teče izvršavanje programa?
      - Izvršavanje programa teče normalno.
  - Proces pokušava da pristupa stranici koja **nije u memoriji** (nego na disku).
    - Kako sada teče izvršavanje programa?
      - Proces pristupa logičkoj stranici čija je vrednost bita validnosti i.
      - To izaziva prekidni signal **PF** (engl. *page-fault trap*).
      - Operativni sistem tada poziva **rutinu za opsluživanje PF**.
      - Rutina za opsluživanje PF prebacuje stranicu sa diska u memoriju.
- Kako se obavlja prebacivanje stranice sa diska u fizičku memoriju?


load M
6
1
i
2
OS
Tabela
stranica
3
5
Slobodan okvir
4
...
Disk
Fizička
memorija

Prebacivanje stranica sa diska u fizičku memoriju

- Kako se obavlja prebacivanje stranice sa diska u fizičku memoriju?

1. Referenca (load M) je prouzrokovala prekid PF.
   - Prilikom čitanja stranice u tabeli detektovan je invalid bit.
2. OS poziva sistemsku rutinu za obradu PF.
   - Ukoliko referenca nije validna, proces se prekida jer sadrži pogrešnu instrukciju.
   - Ukoliko je referenca validna, PF započinje učitavanje stranicu u memoriju.
3. PF rutina pronalazi stranicu na disku u swap prostoru.
4. PF rutina traži slobodan okvir u fizičkoj memoriji i prebacuje stranicu sa diska u okvir.
5. PF rutina ažurira tabelu stranica. Na ulazu koji je napravio PF prekid, upisuje se adresa okvira i poništava invalid bit (postavlja se v bit).
6. Prekinura instrukcija koja je uzrokovala PF prekid se izvršava iz početka, s tim što sada ima sve što joj treba u memoriji.

- Postupak se ponavlja za svaku stranicu procesa koja nije u memoriji, pri čemu svaki PF prekid učitava samo jednu stranicu sa diska.

Učitavanje stranica prema potrebi

- **Performanse DP tehnike** zavise od:
  - Verovatnoće da se dogodi PF greška ($p \in [0,1]$)
  - Trajanja memorijskog ciklusa (t_{MA})
  - Trajanja obrade PF prekida (t_{PF}).
- Efektivno vreme pristupa memoriji (ukoliko se koristi DP tehnika) je:
  - $t_{EA} = (1 - p)\, t_{MA} + p\, t_{PF.}$
- Od čega **zavisi trajanje obrade PF prekida** (t_{PF})?
- Uprošćeno, komponente obrade PF prekida su:
  - Servisiranje PF prekida
  - Čitanje stranice
  - Ponovno izvršenje procesa koji je izazvao PF.

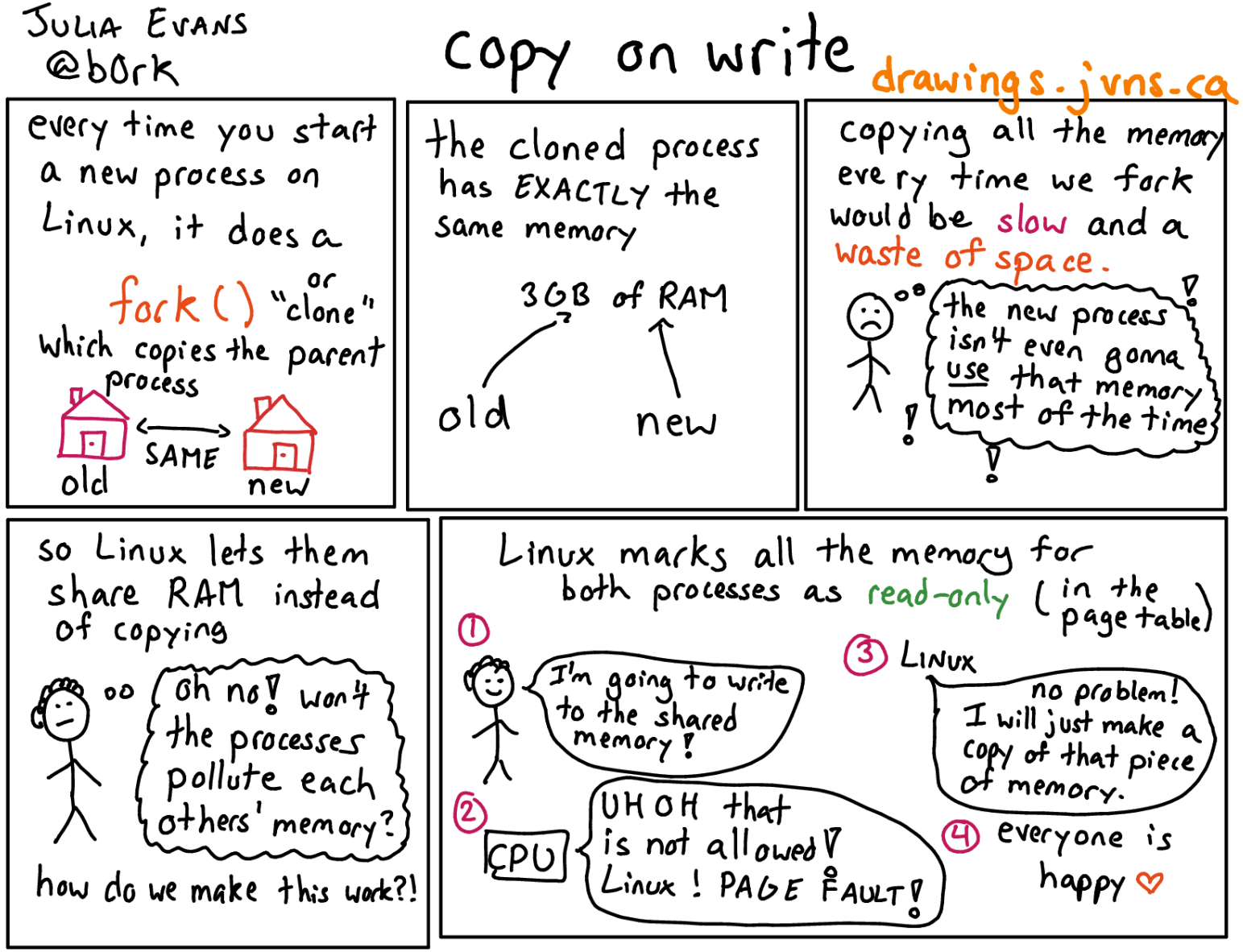
Prebacivanje stranica sa diska u fizičku memoriju

- Detaljnije, PF rutinu čini sledeća sekvenca aktivnosti:

1. Prelazak u operativni sistem izazvan prekidnim signalom PF.
2. Čuvanje konteksta prekinutog procesa u registrima procesora.
3. Određivanje da li je referenca legalna i ako jeste određivanje lokacije stranice na disku.
4. Čitanje stranice sa diska koje izaziva čekanje (PF rutina prelazi u stanje WAIT).
5. Dodela procesora drugom procesoru, jer tekući proces (PF rutina) čeka na resurs.
6. Spašavanje konteksta tekućeg procesa nakon prekidnog signala koji znači da je čitanje stranice sa diska završeno.
7. Korekcija tabele stranica.
8. Čekanje da se procesor ponovo dodeli procesu koji je izazvao PF.
9. Obnova konteksta procesa.

Alternativne tehnike učitavanja stranica: CoW

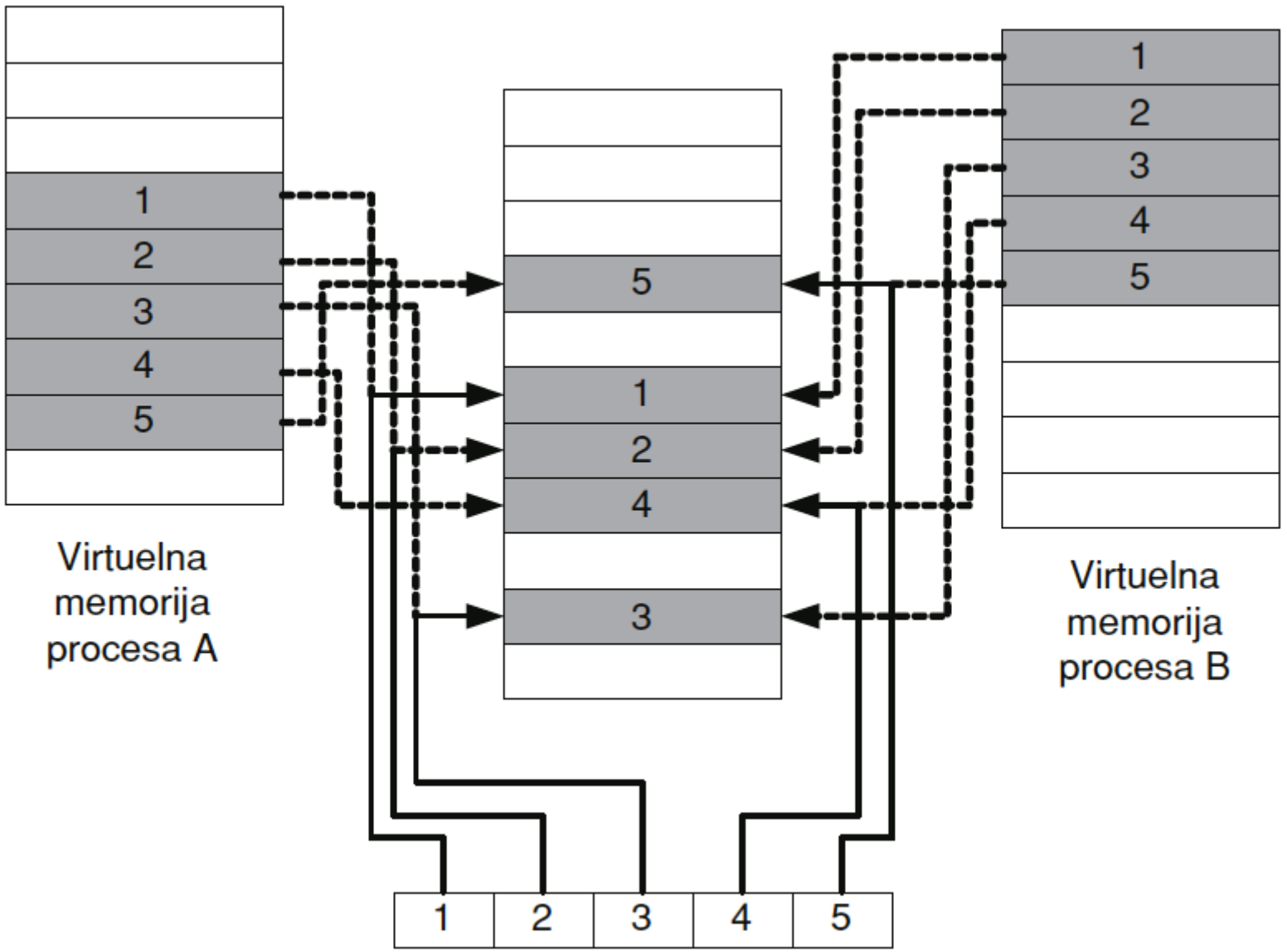
- Kopiranje na ciklus upisa (*CoW*, engl. ***Copy-on-Write***).
  - Sistemski poziv fork duplira adresni prostor procesa roditelja.
  - Posle toga, potrebno je formirati i kopirati sve stranice koje pripadaju roditelju.
  - Kako se može izbeći DP tehnika u ovom slučaju?
    - Stranice proces roditelja se inicijalno ne kopiraju.
    - Nova memorija se inicijalno ne dodeljuje proces detetu.
    - Procesi roditelj i dete inicijalno dele sve stranice, koje se označavaju kao **CoW**.
    - Ako bilo koji proces pokuša da modifikuje stranicu, najpre se kreira kopija stranice koja se dodeljuje tom procesu.
    - Nakon toga proces može da modifikuje svoju kopiju.



* Slika preuzeta sa stranice: https://drawings.jvns.ca/copyonwrite/

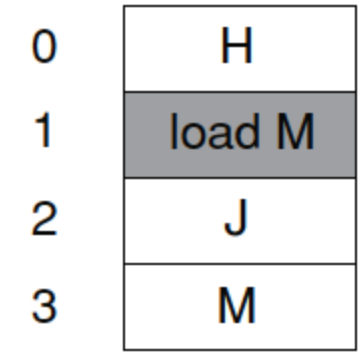
Memorijski mapirane datoteke

- Koncept virtuelne memorije dozvoljava korisnicima da datotekama pristupaju preko **memorijskih referenci**.
- To se postiže pomoću **memorijski mapiranih datoteka**.
  - Deo virtuelnog memorijskog prostora dodeli se datotekama.
  - Slika stranice na disku predstavlja deo datoteke a ne stranicu u swap prostoru.
  - Inicijalni pristup datoteci se odvija preko DP tehnike koja će izazvati PF prekid i kao rezultat te greške pročitaće se deo datoteke u memoriju.
  - Sledeći delovi datoteke dobijaju se na isti način.
- Više procesa može deliti datoteke u memoriji na ovaj način.
  - Datoteka se pročita u memoriju pomoću DP tehnike.
  - Svaki proces podesi svoj adresni prostor tako da ukazuje na datoteku.
  - Upis koji uradi bilo koji proces, automatski je vidljiv svim ostalim procesima.


1
2
3
4
5
Virtuelna memorija procesa A
5
1
2
4
3
1
2
3
4
5
Virtuelna memorija procesa B
1
2
3
4
5

Zamena stranica

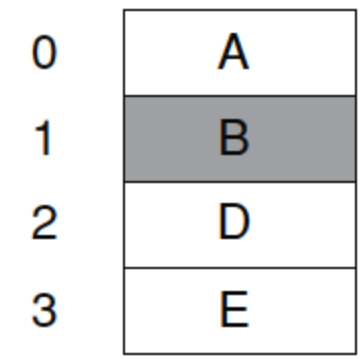
- Proces može da se izvršava ako se samo deo procesa nalazi u memoriji.
- Šta je posledica toga?
  - Povećanje broja aktivnih procesa u memoriji.
  - Povećanje stepena multiprogramiranja.
- Koje pojave prate sisteme sa virtuelnom memorijom?
  - Iznenadna pojavljivanja velikog broja PF prekidnih signala.
  - Potpuno zauzeće cele fizičke memorije.
- Kako rešavamo **problem zauzeća memorije**?
- Primer:
  - Imamo dva procesa sa po 4 logičke stranice i memoriju sa 8 okvira koja je trenutno popunjena.
  - Proces P1 izvršava instrukciju load M koja ima referencu za stranicu 3.
  - Stranica 3 se ne nalazi u fizičkoj memoriji, nego na disku, tako da nastupa PF prekidni signal.
  - Kako nema slobodnih okvira, jedan mora da se žrtvuje i da se oslobodi za stranicu M.



Logički adr. prostor
(korisnik 1)

p	f	v/i
0	3	v
1	4	v
2	5	v
3		i

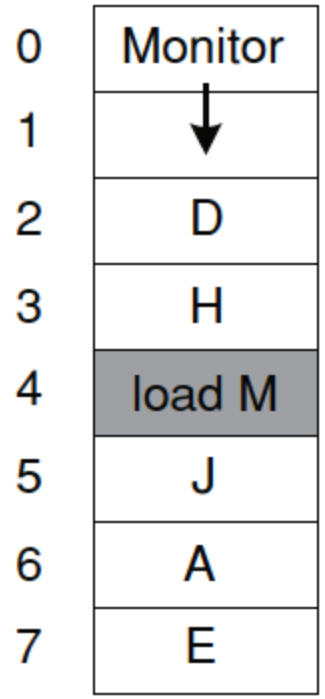
Tabela stranica
korisnik (1)



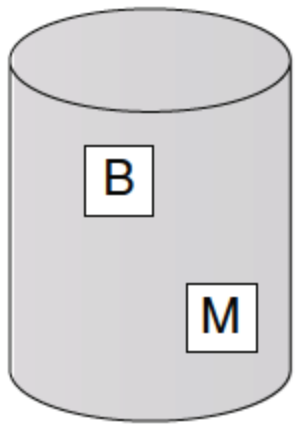
Logički adr. prostor
(korisnik 2)

p	f	v/i
0	6	v
1		i
2	2	v
3	7	v

Tabela stranica
(korisnik 2)



Fizički adresni
prostor

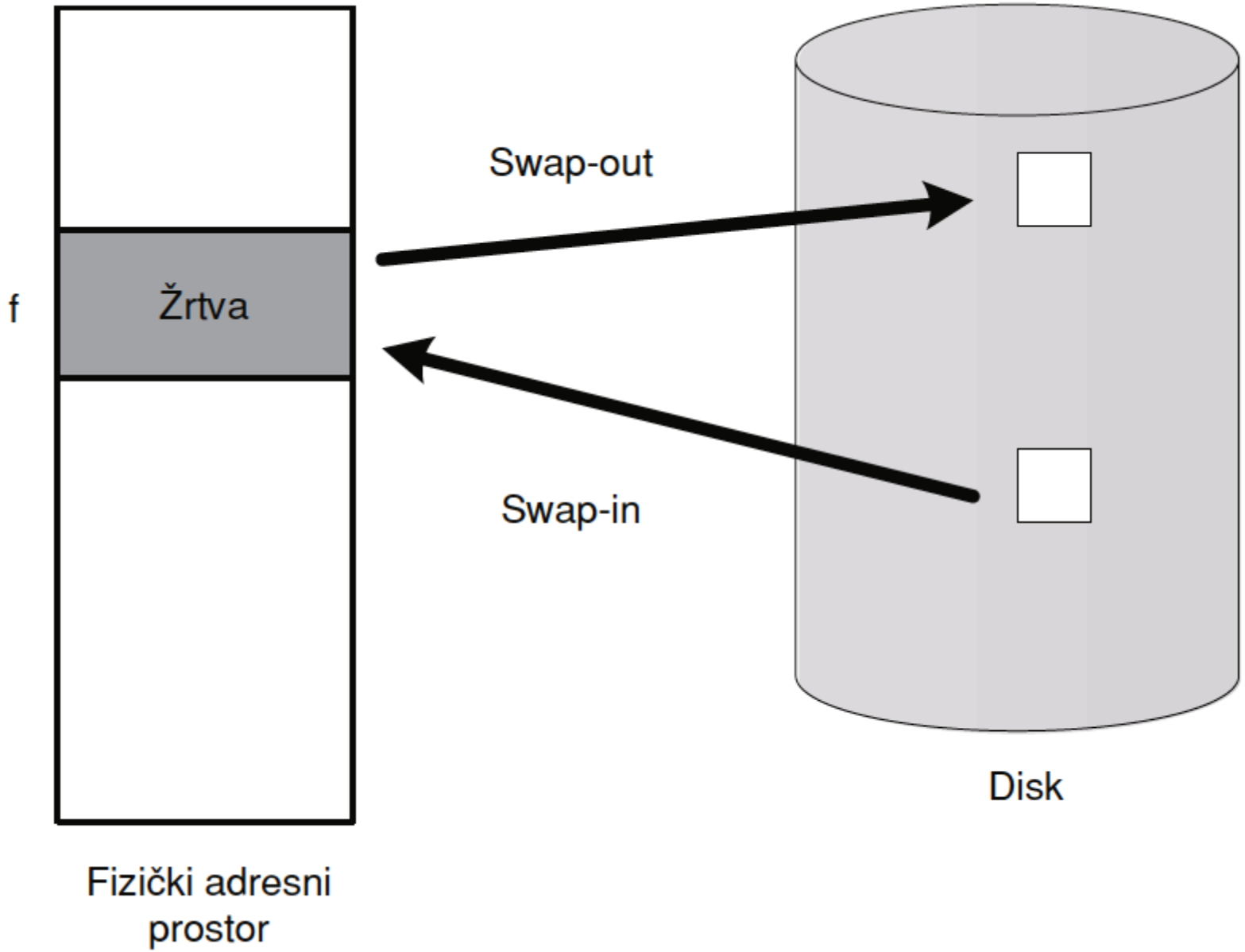


Disk

Zamena stranica

- Kako funkcioniše zamena stranica?

1. Najpre se bira okvir u koji će biti smeštena željena stranica.
   - Ako ima slobodnih okvira, uzima se jedan od njih.
   - Ako nema slobodnog, pomoću algoritama za zamenjivanje stranica bira se **žrtva**.
     - Žrtva = okvir u kome se nalazi stranica koja će biti zamenjena.
2. Žrtvovana stranica se upisuje na disk.
   - Ako sadržaj okvira nije modifikovan u odnosu na postojeću sliku na disku upis se ne radi!
     - Ovaj korak zahteva da se u tabelu stranica uvede **bit čistoće** (engl. *dirty flag*).
     - Znatno povećava performanse!
3. Žrtvovani okvir se u tabeli stranica obeležava kao **nevalidan**.
4. Željena stranica se učitava u oslobođeni okvir.
5. Tabela stranica se ažurira i nova stranica se obeležava kao **validna** bitom validnosti.


f
Žrtva
Swap-out
Swap-in
Disk
Fizički adresni prostor

Algoritmi za izbor žrtve

- U slučaju pojave PF prekidnog signala smanjuju se performanse procesa.
- Broj PF prekida mora se održati na što manjem nivou.
- Potrebno je naći algoritam koji će za datu sekvencu referenci da postigne **najmanju verovatnoću pojave PF grešaka**.
- Algoritmi za izbor žrtve:
  - FIFO
  - Optimalni
  - LRU
  - Druga šansa
  - NRU
  - Frekvencijski algoritmi (LFU, MFU)

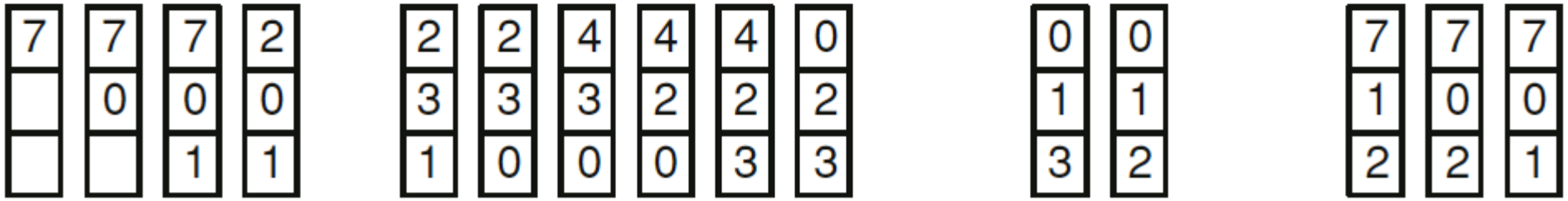
Algoritmi za izbor žrtve: FIFO

- Formira se FIFO red na čiji se kraj stavljaju stranice prilikom učitavanja u memoriju.
- Prilikom zamene, memoriju napušta stranica koja se nalazi na početku reda.
- Primer:

Niz referenci:

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

Okviri:

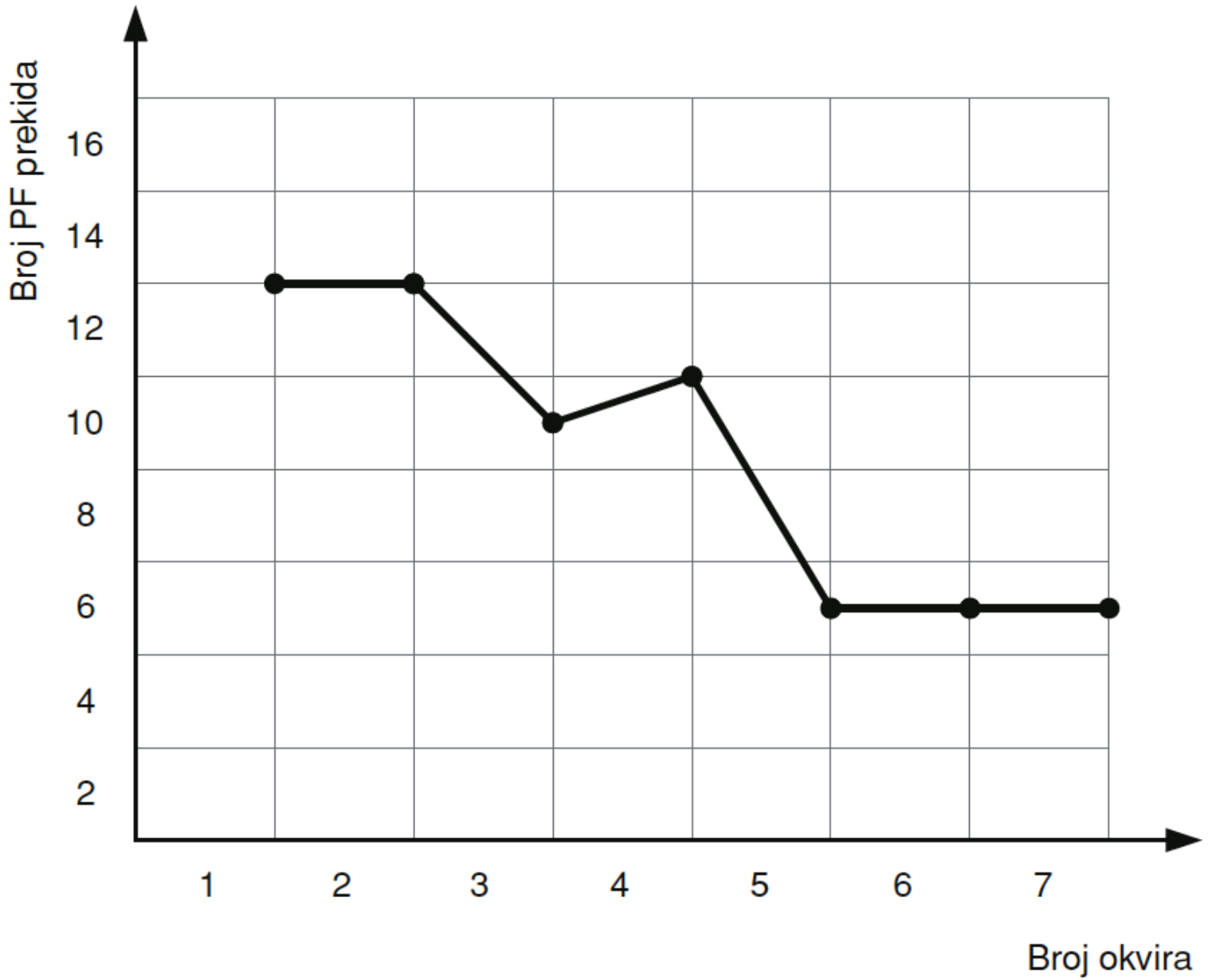


Algoritmi za izbor žrtve: FIFO

- Objašnjenje primera:
  - Prve tri reference napraviće tri uzastopne PF greške i popuniti sva tri okvira.
  - Četvrta reference usloviće PF grešku koja mora da izazove zamenu stranice.
    - Kao žrtva, bira se okvir 0 u kome se nalazi najstarija stranica (stranica 7).
  - Peta referenca ne izaziva PF jer se stranica 0 nalazi u drugom okviru u memoriji.
  - Šesta referenca izaziva novi PF i novu zamenu.
    - Kao žrtva bira se okvir 1 koji sadrži stranicu 0.
  - ...
  - Broj PF prekida za datu sekvencu: 15

Algoritmi za izbor žrtve: FIFO

- Šta je **Belady-jevu anomalija**?
- Da li broj PF grešaka strogo opada sa povećanjem broja okvira u memoriji?
- Ne mora da znači!
- Primer:
  - Niz refernci: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.
  - Broj okvira: 1, 2, ... 7.
  - Analiza pokazuje da FIFO algoritam izaziva veći broj PF grešaka u slučaju memorije sa 4 okvira u odnosu na memoriju sa tri okvira!


Broj PF prekida
16
14
12
10
8
6
4
2
1
2
3
4
5
6
7
Broj okvira

Algoritmi za izbor žrtve: optimalni algoritam

- FIFO svoj kriterijum formira na istoriji referenci koja je uvek poznata.
- OPT algoritam svoj kriterijum formira **na budućnosti**!
  - OPT mora unapred da poznaje sve procese, što **nije moguće**!
  - Ovo naročito važi ako se radi o interaktivnim sistemima
- U čemu je štos?
  - OPT žrtvuje stranicu koja se neće koristiti najduže vremena.
  - Time se maksimalno odlaže pojavljivanje sledeće PF greške.
  - Primer:
    - Stranica A će biti potrebna posle 100 instrukcija.
    - Stranica B će biti potrebna posle 1000 instrukcija.
    - Bolje je žrtvovati stranicu B jer se izazivanje sledeće PF greške duže odlaže.
- OPT proizvodi minimum PF grešaka ali **NE može da se implementira**!

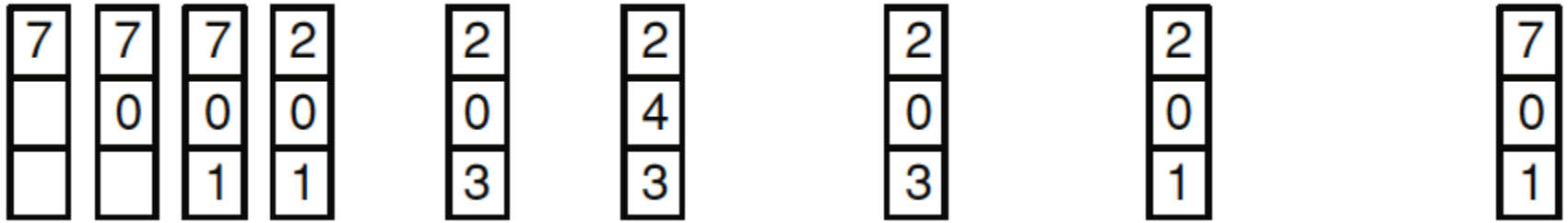
Algoritmi za izbor žrtve: optimalni algoritam

- Primer:

Niz referenci:

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

Okviri:



- Broj PF prekida za datu sekvencu: 9

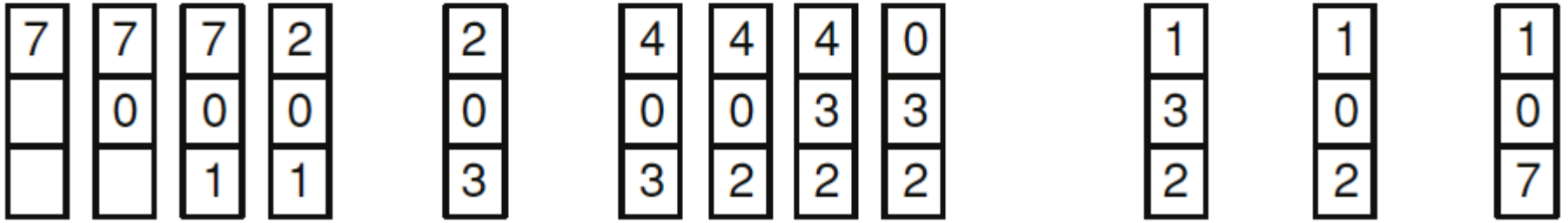
Algoritmi za izbor žrtve: LRU

- LRU algoritam (engl. *least recently used*) žrtvuje stranicu koja najduže nije korišćena.
  - LRU svakoj stranici dodeljuje vreme poslednjeg korišćenja.
  - Vreme poslednjeg korišćenja se ažurira sa sa svakim pristupom stranici.
- Primer (za datu sekvencu LRU generiše 12 PF prekida, dok za istu FIFO generiše 15 a OPT 9):

Niz referenci:

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

Okviri:

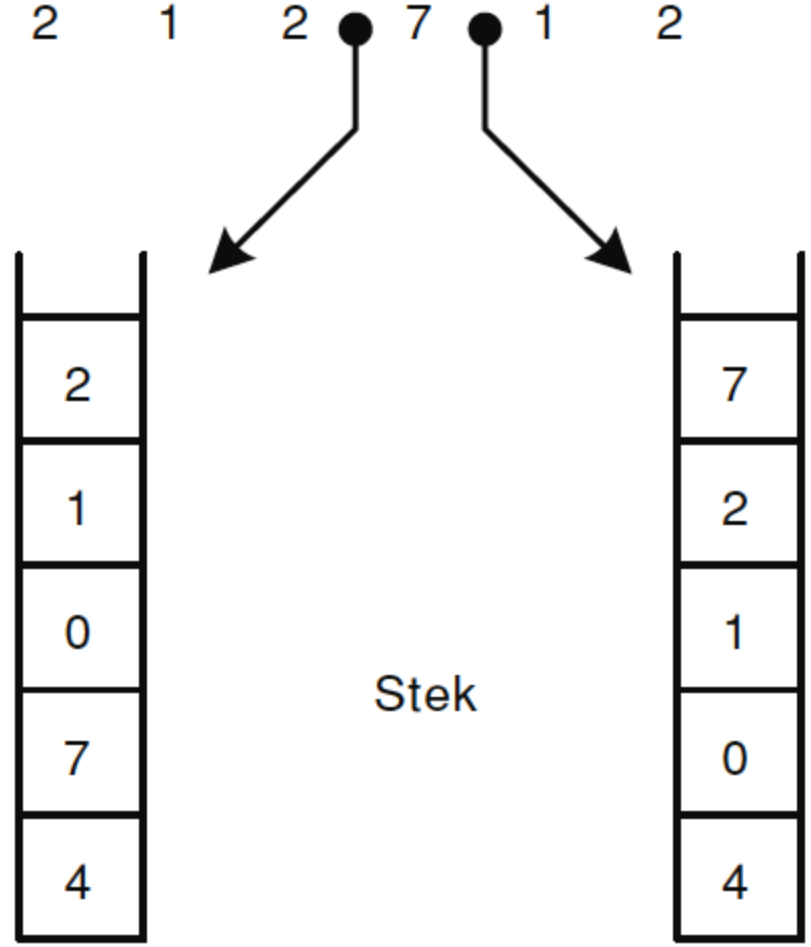


Algoritmi za izbor žrtve: LRU

- Algoritam se može realizovati na sledeća dva načina:
- **Pomoću brojača**.
  - Procesor ima brojač memorijskih instrukcija.
  - Posle svake memorijske instrukcije vrednost brojača se povećava za jedan.
  - Svaki okvir ima svoj interni brojač.
  - Svaki put kada se pristupi okviru sadržaj brojača instrukcija se kopira u interni brojač okvira.
  - Kada dođe do PF prekida, algoritam bira **stranicu čiji je broj najmanji**.
    - To znači da najduže nije korišćena.
- **Pomoću steka**.
  - Formira se stek koji opisuje redosled pristupanja stranicama.
  - Nakon PF prekida, žrtvuje se stranica sa vrha steka.
  - Na njeno mesto se stavlja stranica kojoj se pristupa.
  - Ova realizacija je veoma spora jer se pri svakom pristupu memoriji stek mora ažurirati.

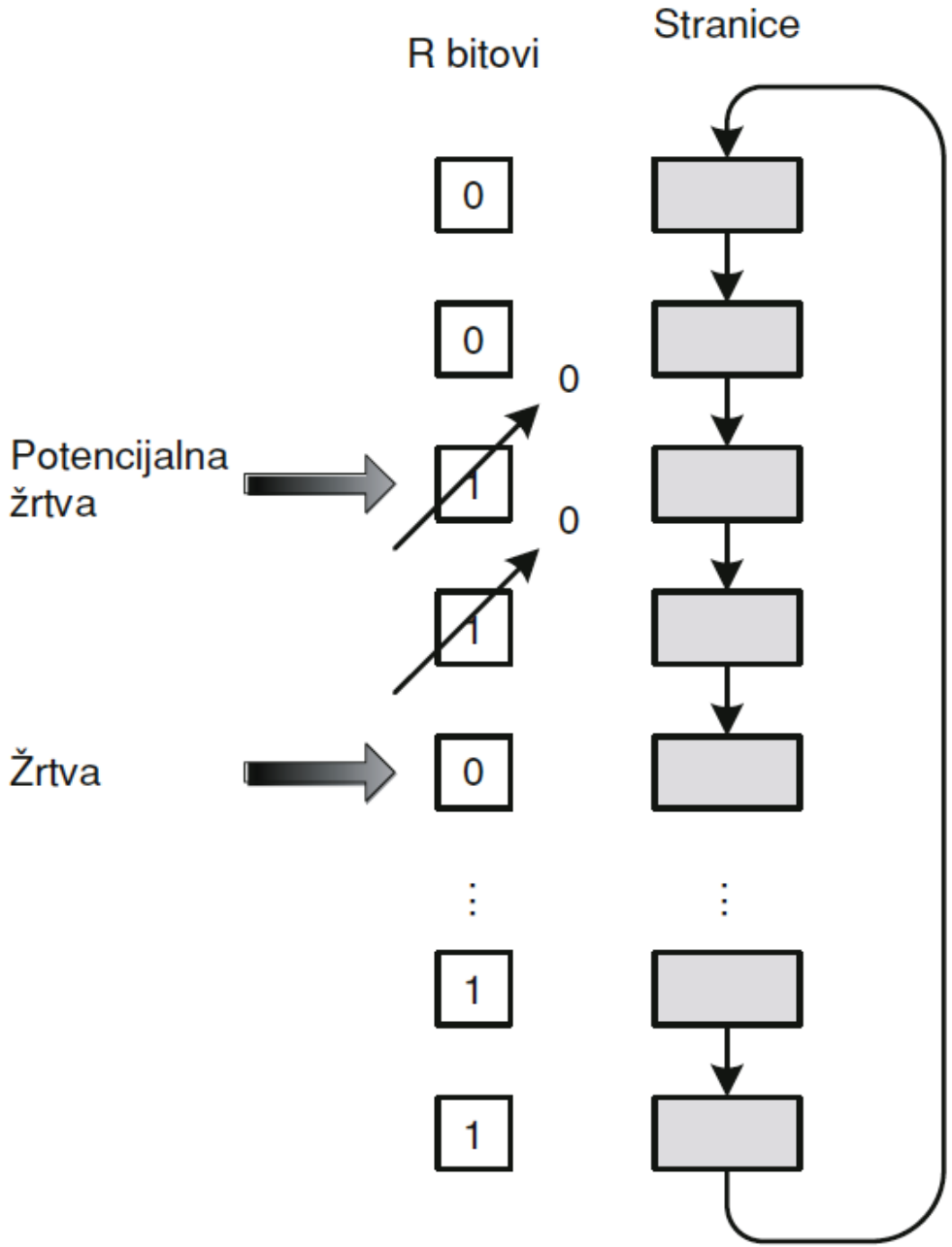
Niz referenci:

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



Algoritmi za izbor žrtve: druga šansa (satni algoritam)

- Druga šansa je varijanta FIFO algoritma koja u obzir uzima i korišćenost stranica.
- Formira se FIFO red na čiji se kraj stavljaju stranice prilikom učitavanja u memoriju.
- Koristi se vrednost **R** (*reference*) bitova:
  - R bitovi se postavljaju inicijano na 0.
  - Brisanje R bita nakon određenog vremena obavlja se pomoću tajmerskih prekida.
  - R bit se postavlja na 1 ako smo od poslednjeg brisanja bar jednom pristupili stranici.
- Potencijalna žrtva je stranica na kraju reda čekanja.
  - R=0: stranica se dugo nalazi u memoriji, nismo joj pristupili.
    - Stranica očigledno nije potrebna, tako da se može žrtvovati.
  - R=1: stranica je dugo u memoriji ali je i nedavno korišćena.
    - R bit te stranice biće resetovan posle provere.
    - Sama stranica biće prebačena na početak reda opsluživanja.
    - Stranica dobija drugu šansu.
    - Pretraga se nastavlja dok se ne naiđe na stranicu sa postavljenim R=0.


R bitovi
Stranice
0
0
0
Potencijalna
žrtva
1
0
1
Žrtva
0
1
1

Algoritmi za izbor žrtve: NRU

- Algoritam NRU (engl. not recently used) zahteva da se svakom okviru pridruže dva bita:
  - **R** (engl. *reference*) bit – označava da li smo pristupili tom okviru.
  - **M** (engl. *modified*) vit – označava da li smo menjali sadržaj tog okvira.
- Na početku rada procesa svi R i M bitovi dobijaju vrednost 0.
- Na osnovu vrednosti R i M bitova okvire delimo u sledeće klase:
  - **R=0, M=0**. Stranica nije skoro ni korišćena ni modifikovana. Idealna žrtva.
  - **R=0, M=1**. Stranica nije skoro korišćena, ali je modifikovana.
    - Nije pogodna za zamenu, jer mora da se upiše na disk, što će izazvati dva I/O ciklusa.
  - **R=1, M=0**. Stranica je skoro korišćena, ali nije modifikovana.
    - Najverovatnije će se koristiti ponovo.
  - **R=1, M=1**. Stranica je skoro korišćena i modifikovana.
    - Verovatno će se ponovo koristiti.
    - Nije pogodna za zamenu, jer mora da se upiše na disk, što će izazvati dva I/O ciklusa.
- Slično satnom algoritmu pretražuju se klasa (0,0), (0,1), ...

Algoritmi za izbor žrtve: frekvencijski bazirani algoritmi

- Prethodno algoritmi nisu uzimali u obzir koliko puta je stranica bila referencirana!
- Dva algoritma koji kriterijum formiraju na osnovu frekvencije korišćenja stranice su:
- **LFU** (engl. *least frequently used*)
  - LFU algoritam zamenjuje stranicu koja ima najmanji broj referenci u vremenu.
  - Svaka stranica koja se učita se određeno vreme mora štititi od samog algoritma!
    - Kada se pojavi njena frekvencija je jednaka 1, što je minimum.
    - Stranica je potenicjalna žrtva i ne bi dobila šansu da se takmiči sa drugim stranicama.
- **MFU** (engl. *most frequently used*)
  - MFU algoritam zamenjuje stranicu koja ima najveći broj referenci.
  - Algoritam smatra da su te stranice odavno učitane u memoriju i da procesima više nisu potrebne, dok su stranice sa malim brojem referencu relativno nove i potrebne.

Globalna i lokalna zamena stranica

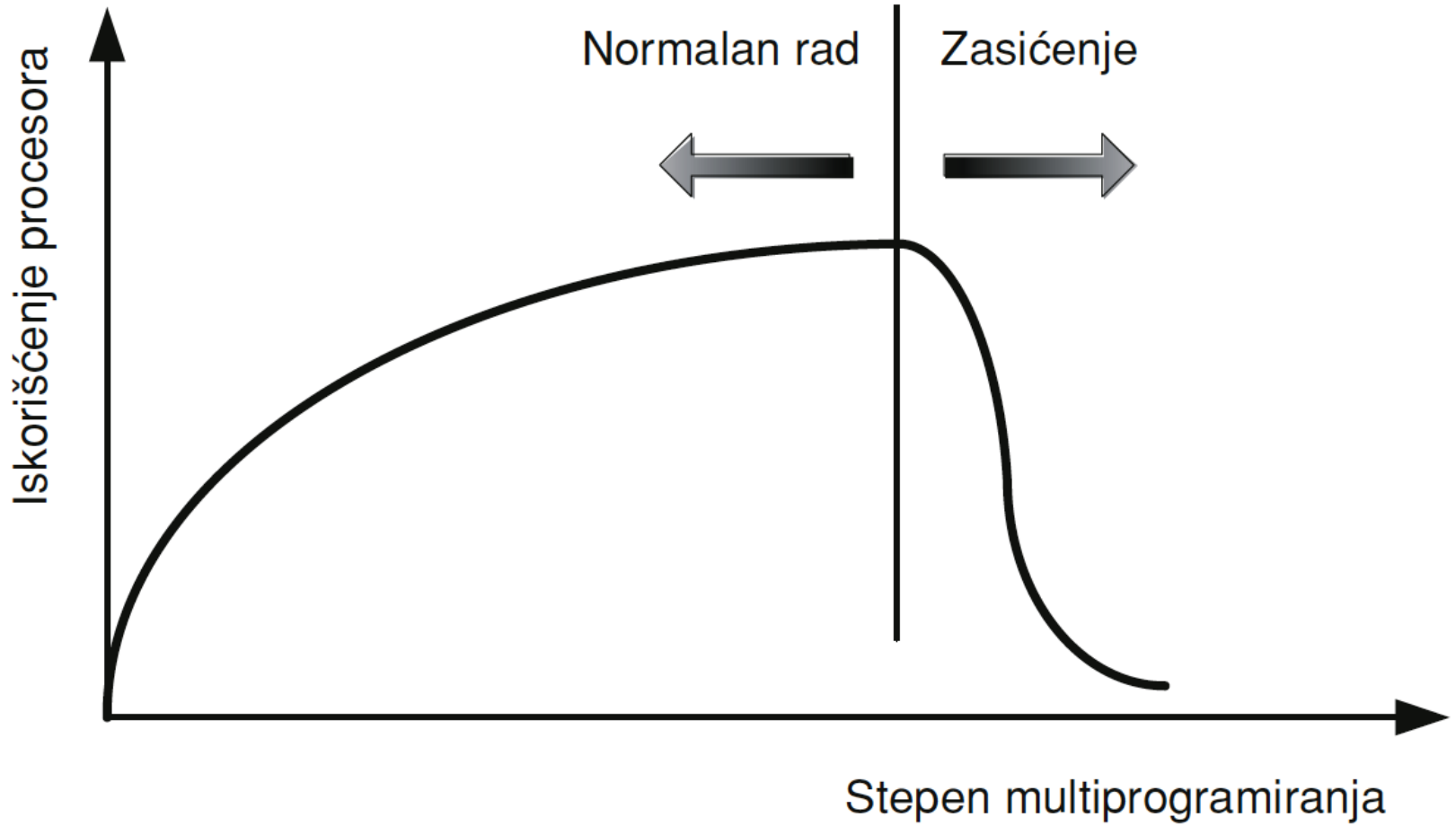
- **Globalna zamena**.
  - Jedan proces može žrtvovati tuđu stranicu (odnosno poslati je u *swap* prostor).
  - Broj okvira dodeljenih procesu može se menjati u vremenu.
  - Na broj PF prekida utiču i drugi procesi.
- **Lokalna zamena**.
  - Jedan proces može žrtvovati isključivo svoje stranice.
  - Broj okvira dodeljenih procesu se ne menja u vremenu.
  - U slučaju lokalne razmene proces sam diktira svoje PF prekide (drugi procesi nemaju uticaj na broj PF prekida koje taj proces izaziva).
  - Mana lokalne zamene**: blokiran proces** ne može ustupiti svoje okvire drugim procesima.
    - To znači da aktivan proces mora žrtvovati sopstvene stranice.
    - Zbog toga globalna razmena daje bolje rezultate.

Raspodela okvira po procesima

- Pitanje: kako rasporediti m okvira na n procesa?
  - Minimalni broj okvira po procesu zavisi od konkretne procesorske arhitekture.
  - Najmanji je dva (instrukcija od više bajtova može da bude na granici između dve stranice).
  - Maksimalni broj okvira koji proces može da dobije je ograničen veličinom fizičke memorije.
- **Metoda jednake raspodele**. Svakom procesu se dodeljuje jednaka količuna okvira m/n.
- **Metoda proporcionalne raspodele**.
  - m – ukupan broj okvira fizičke memorije
  - s_i – veličina procesa p_i
  - n – broj procesa
  - S – ukupna virtuelna memorija: $S = \sum_{i=1}^{n} s_i$
  - Svakom procesu pripašće sledeći broj okvira: $a_i = \frac{s_i}{\sum_{i=1}^{n} si} m$

Efekat zasićenja

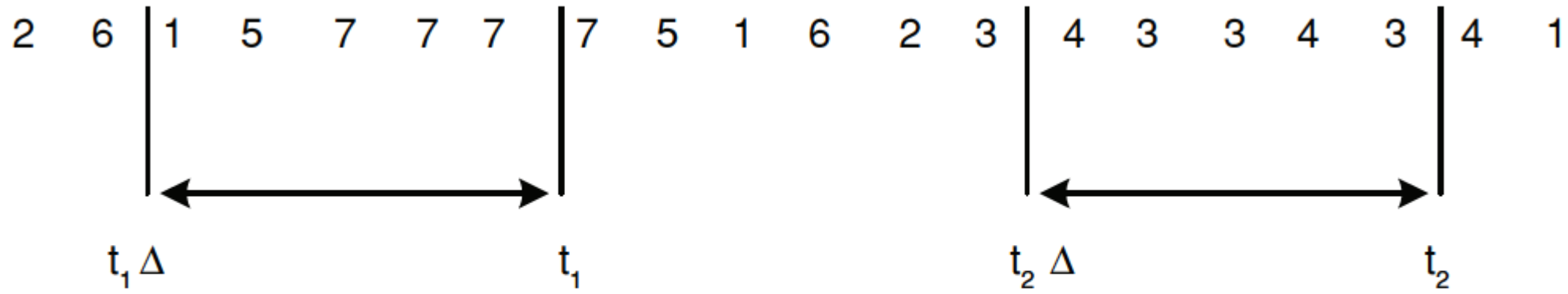
- Proces zahteva operativnu memoriju kao resurs (čista nuklearna fizika!)
- Šta se dešava ako **broj raspoloživih okvira za proces padne ispod minimalne vrednosti**?
  - Resurs (memorija) nije slobodan.
  - Proces ne može da se izvršava.
  - Proces se mora dovest u stanje WAIT.
- Šta se dešava ako je **broj okvira raspoloživih za proces veoma mali**?
  - Resurs (memorija) je slobodan.
  - Proces može da se izvršava.
  - Pojava PF grešaka je česta!
    - Proces veoma često zamenjuje svoje stranice (veliki broj U/I operacija sa diskom).
- Pojava čestog razmenjivanja stranica koja nastaje kao posledica visokog stepena multiprogramiranja naziva se **efekat zasićenja** (engl. *thrashing*).
  - U takvim situacijama javljaju se ozbiljni problemi u vidu degradacije performansi.


Normalan rad
Zasićenje
Iskorišćenje procesora
Stepen multiprogramiranja

Efekat zasićenja

- Uvešćemo model lokalnosti u svaki proces, koji će nam pomoći da objasnimo efekat zasićenja.
- **Lokalnost** je skup stranica koje proces koristi zajedno u jednom intervalu vremena.
- Proces može menjati svoje lokalnosti a one se mogu preklapati.
- Primer:
  - Svaki poziv potprograma napraviće novu lokalnost.
  - Kad se izađe iz tog potprograma lokalnost se menja.
- Zasićenje nastupa kada je **suma lokalnosti za sve aktivne procese** veća od fizičke memorije!
- Uvodimo novi model koga ćemo nazvati **radni model** (engl. *working-set model*).
  - **Prozor radnog skupa** Δ (engl. *working-set window*) je vreme u kome se izvrši određeni broj instrukcija.
  - Sve stranice referencirane u tom periodu predstavljaju **radni skup** (engl. *working-set*) za Δ.

Efekat zasićenja



$WS(t_1) = \{1, 5, 7\}$

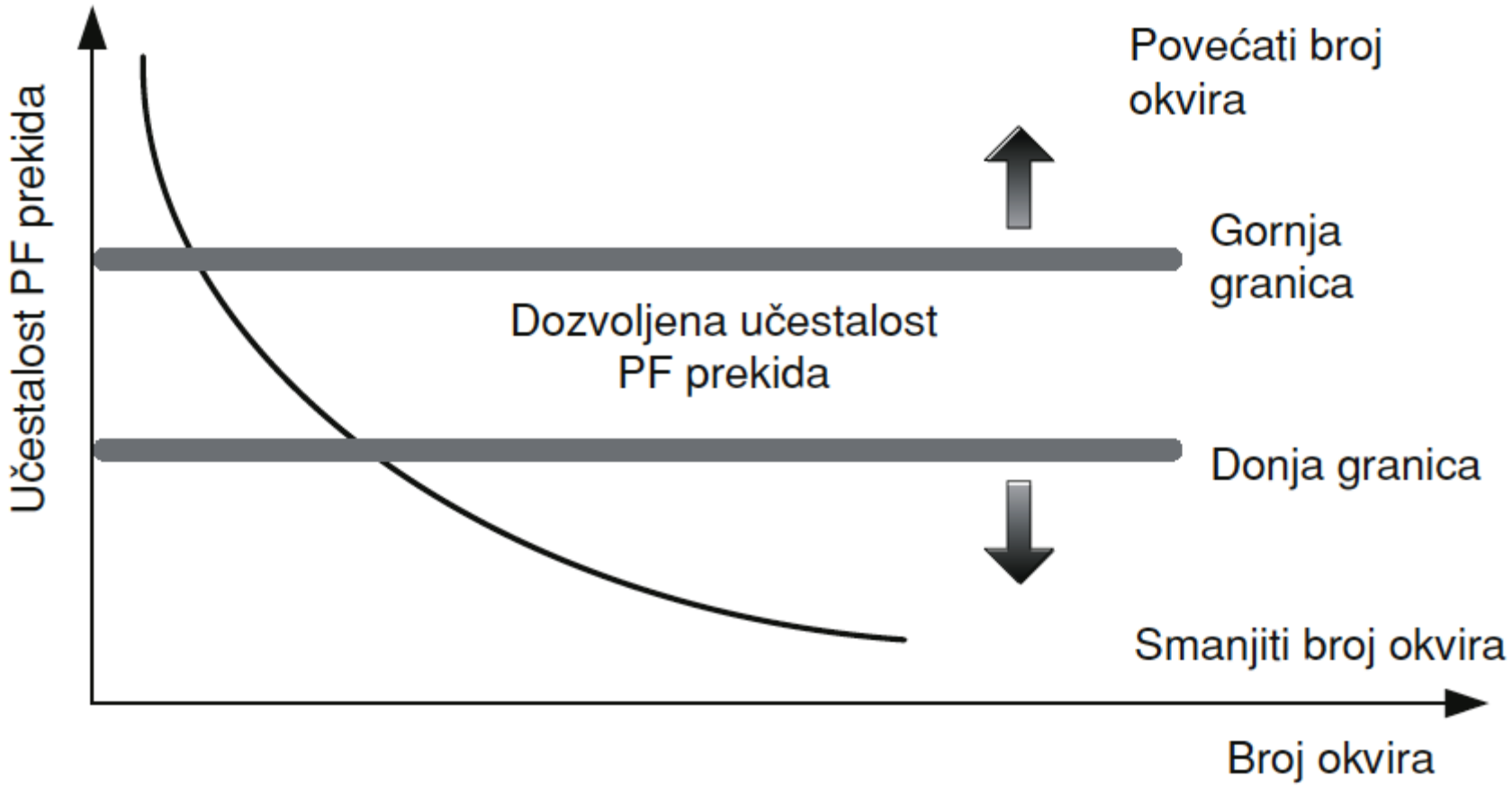
$WS(t_2) = \{3, 4\}$

- Primer: data su dva radna skupa za po 5 memorijskih referenci.
  - WS(t1) obuhvata stranice 1,5 i 7.
  - WS(t2) obuhvata stranice 3 i 4.

Efekat zasićenja

- U celoj ovoj priči najznačanija je veličina radnog skupa.
- **Veličina radnog skupa** WSS_i procesa P_i definiše se kao ukupan broj stranica koje proces traži u vremenskom prozoru Δ.
- Ukupna veličina radnog prostora u sistemu D je: $D = \sum_{i=1}^{n} WSS_i$

- Ako je D > m, nastupiće efekat zasićenja.
  - OS prati veličinu D.
  - Kada D dođe do veličine sistemske memorije:
    - Novi procesi se neće uvoditi.
    - Pojedini procesi moraju da se suspenduju na disk, kako bi se smanjila veličina radnog skupa.
- Efekat zasićenja se može sprečiti limitiranjem broja PF prekida koje jedan proces izaziva u okvirima dozvoljenog opsega.
  - Broj PF prekida je veliki u slučaju zasićenja!


Povećati broj okvira
Gornja granica
Dozvoljena učestalost PF prekida
Donja granica
Smanjiti broj okvira
Učestalost PF prekida
Broj okvira

Efekat zasićenja

- Koji se algoritam primenjuje?
  - Najpre se postavljanjem donje i gornje granice definiše **opseg dozvoljenog broja PF prekida** za jedan proces.
  - Zatim se prati broj, odnosno frekvencija izazivanja prekida.
  - Ako je broj PF grešaka procesa **manji od donje granice**:
    - Procesu je dodeljeno više okvira.
    - Poneki se može oduzeti i dodeliti drugim procesima.
  - Ako je broj PF grešaka **veći od gornje granice**:
    - Procesu nije dodeljen dovoljan broj okvira.
    - Proces može da dobije još ovira.
  - Ako se broj PF grešaka svakog procesa održava u dozvoljenom opsegu (između granica):
    - Efekat zasićenja se neće dogoditi.

Literatura

1. B. Đorđević, D. Pleskonjić, N. Maček (2005): Operativni sistemi: teorija, praksa i rešeni zadaci. Mikro knjiga, Beograd.
2. R. Popović, I. Branović, M. Šarac (2011): Operativni sistemi. Univerzitet Singidunum, Beograd.

Hvala na pažnji

Pitanja su dobrodošla.