Operativni sistemi
- Virtuelna memorija -

Veljko Stanković

Uvod

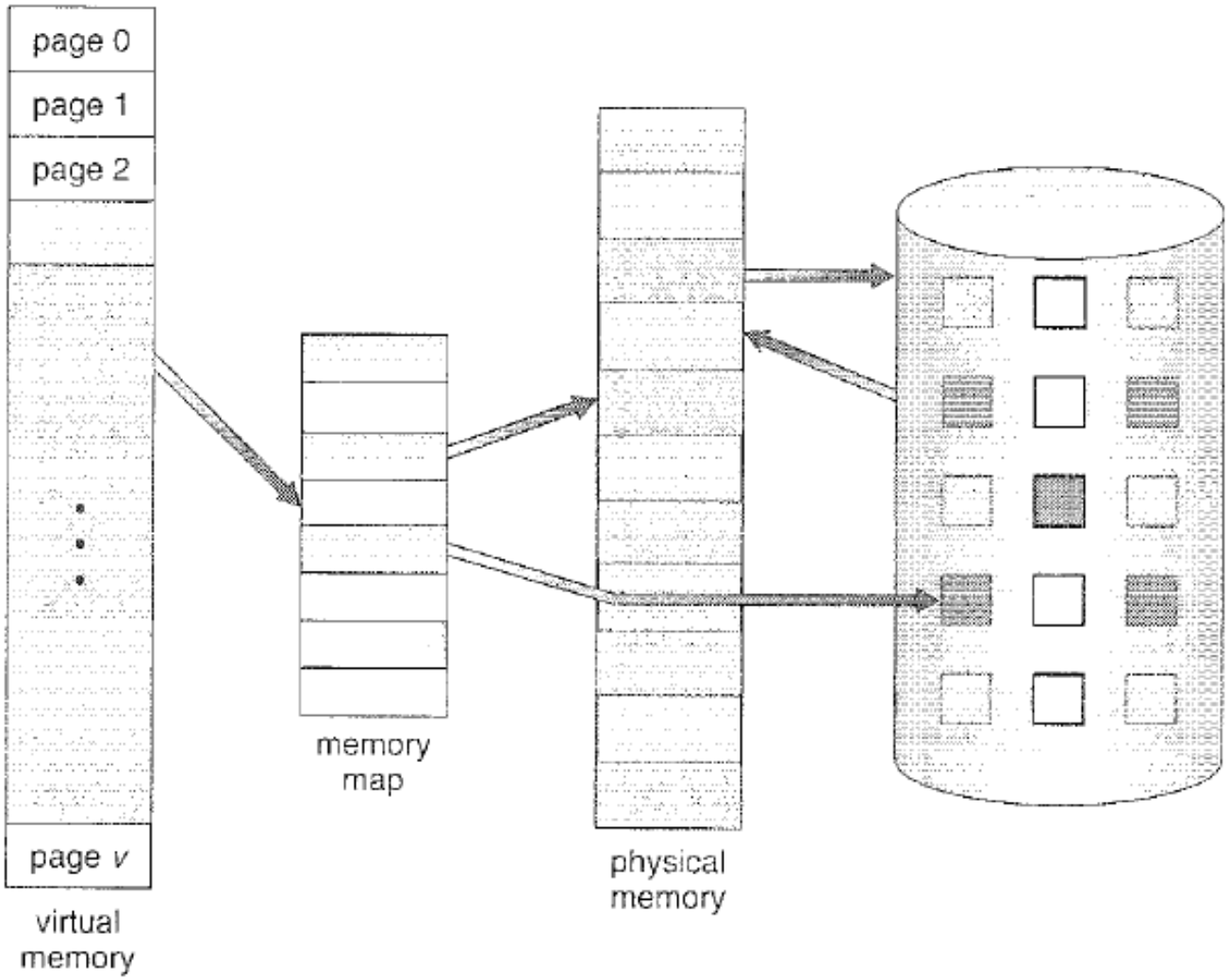
- Virtuelna memorija (VM) je tehnika koja omogucava izvrsavanje programa koji se ne nalaze u potpunosti u glavnoj memoriji.
  - Programi mogu biti veci od glavne memorije.
  - Prav ise razlika izmedju logicke i fizicke memorije.
  - Omogucava da programi veoma lako dele fajlove i laku implementaciju deljive memorije.
  - VM nije laka za implemetnaciju i moze stvoriti dosta problema koa se ne vodi racuna o tome kako se koristi.
  - Razlozi uvodjenja VM
    - ☑ Delovi koda koji se koriste za obradu gresaka se retko izvrsavaju i ne moraju biti stalno u GM.
    - ☑ Vektorima, listama i matricama se obicno dodeljuje vise memorije nego sto je potrebno.
    - ☑ Neki delovi kdoa koji se retko koriste ne mroaju biti u GM.

Uvod

— Prednosti koriscenja VM

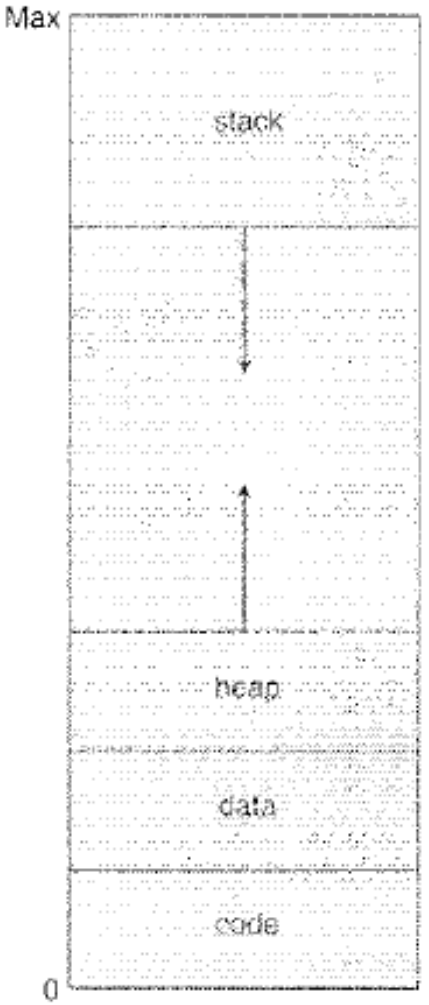
- Korisnici mogu da pisu programe koji bi koristili veoma veliki virtuelni adresni prostor.
- Vise programa se moze izvrsavati istovremeno posto zauzimaju manje fizicke memorije.
- Programi se izvrsavaju brze jer je ucitavanje programa u GM brze.

Uvod


page 0
page 1
page 2
page v
virtual memory
memory map
physical memory

Uvod

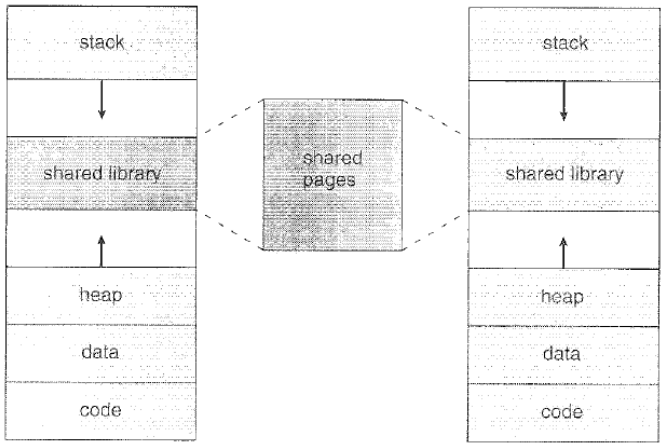
- Virtuelna memorija omogucava odvajanje logickog i ifizickog adresnog prostora.
  - Programi mogu obuhvatati veliki logicki adresni prostor pri cemu koriste malo deo fizickog adresnog prostora.
- Virtuelni adresni prostor procesa se odnosi virtuelni nacin smestanja procesa u memoriju.
  - Proces pocinje od neke odredjene adrese i zauzima kontinualan skup adresa.
  - MMU ima zadatak preslikavanja logickih adresa u fizicke.
- Primer virtuelnog adresnog prostora ⇨
  - Adresni prostor heap-a raste ka visim adresama
    - ☑ Koristi se za dinamicku dodelu prostora
  - Adresni prostor magacina (stack) raste ka nizim adresama



Uvod

— Mogucnost deobe fizicke memorije od strane dva i vise procesa ima sledece prednosti:

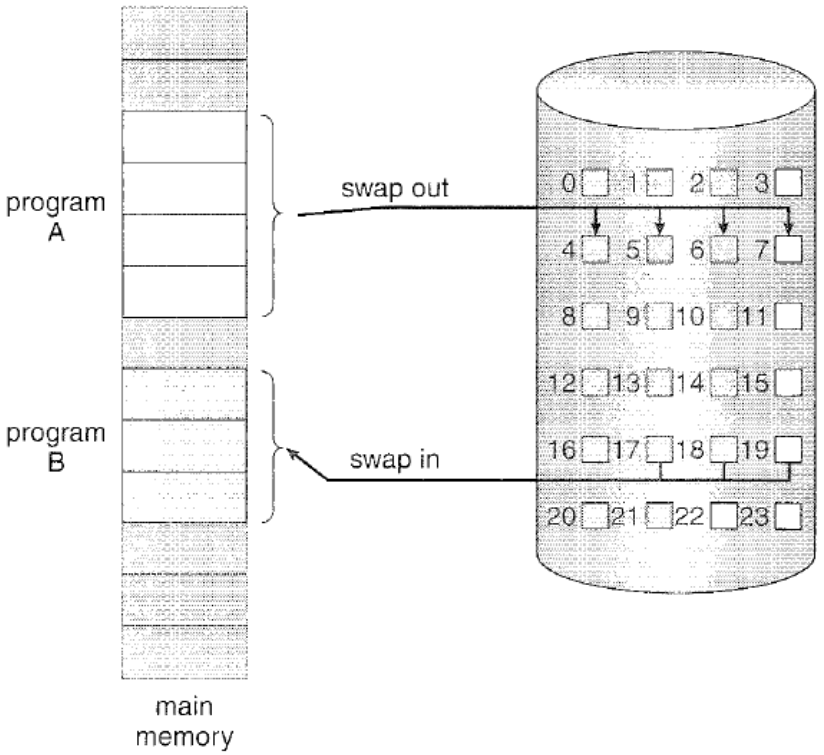
- Sistemske biblioteke mogu da koriste vise procesa preslikavanjem zajednickog objekta u virtuelni adresni prostor.
  - ☑ Sistemske bibilioteke su smestene na stranice virtuelne memorije zajednicke za sve programe.
- VM omogucava deobu memorije izmedju procesa.
  - ☑ Jedan proces moze da kreira region koji ce se koristiti za komunikaciju.
  - ☑ Fizicki adresni prostor je zajednicki dok ga svaki proces posmatra kao deo svog logickog adresnog prostora.
- Omogucava brze kreiranje procesa.



Stranicenje na zahtev
(Demand paging)

1010011
1110100
1100001
1010110

— Stranice programa se ucitavaju u GM samo kada se javi potreba za izvrsavanjem koda koji one sadrze.
  - Proces se nalazi na sekundarnoj memoriji (HDD) i po potrebi se ucitavaju stranice u GM.
  - Proces koji ucitava stranice u GM se naziva lazy swapper (lenji svaper).
  - Posto se u GM ucitavaju samo stranice procesa a ne citav proces umesto termina swapper (koji radi sa celim procesom) koristicemo termin pager (pejdzer).



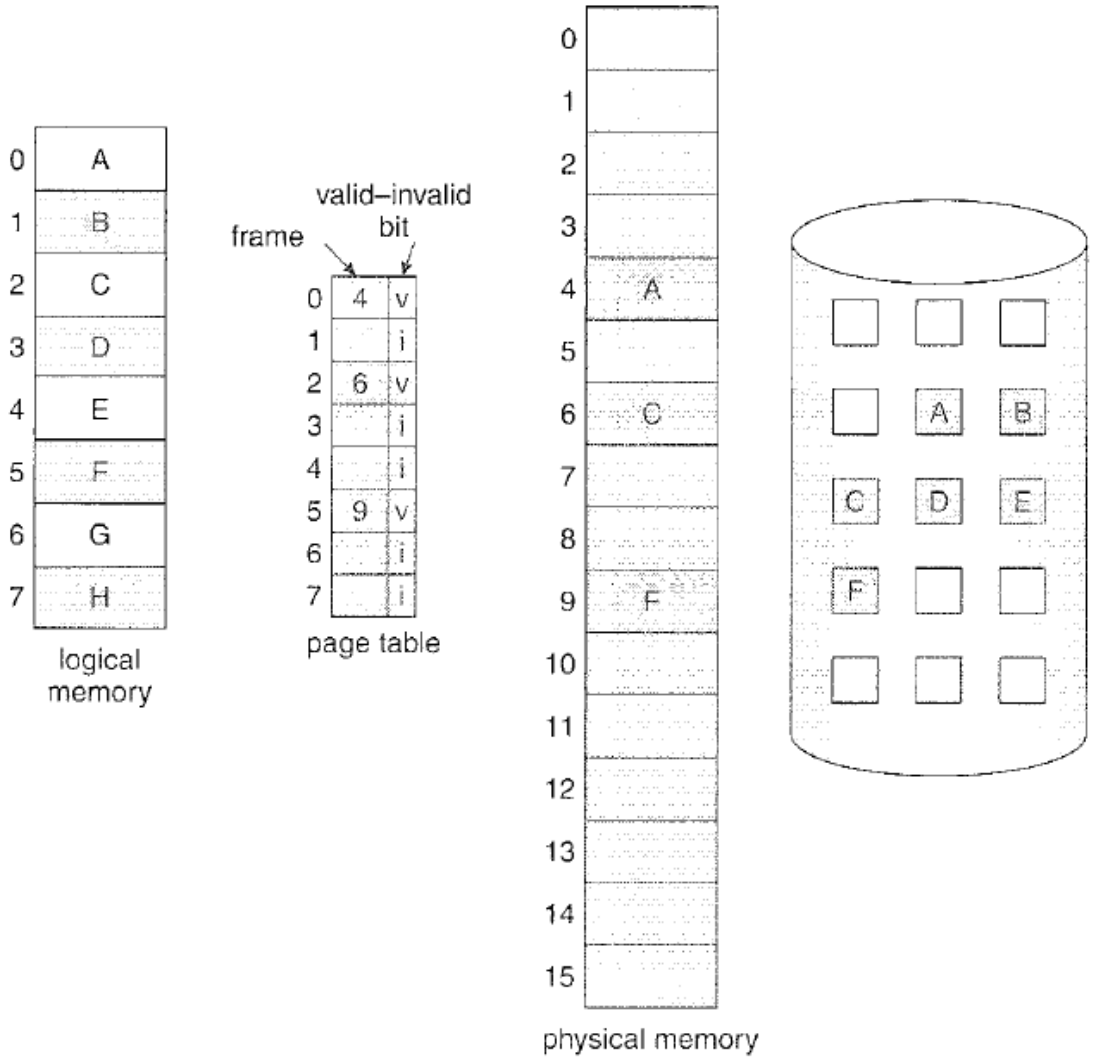
Stranicenje na zahtev
(Demand paging)

1010011
1110100
1100001
1010110

— Pre ucitavanja u GM pager pokusava da pogodi koje ce se stranice koristiti.
  - Smanjuje se vreme ucitavanja i velicina fizicke memorije koja ce biti zauzeta.
— U tabeli stranicenja uvodi se valid/invalid bit
  - Kada je postavljen na vrednost valid, stranica procesa je ucitana u memoriju
  - Kada je bit postavljen na vrednost invalid, stranica je ili izvan logickog adresnog prostora ili se nalazi na sekundarnoj memoriji.
— Ako proces pokusa da pristupi stranici koja nije ucitana u glavnu memoriju aktivira se **page-fault trap**.
  - OS proverava u tabeli stranicenja da li je bit postavljen na vrednsot invalid
    - ☑ Moze biti samo jedan bit koji pokazuje da stranica nije u GM ili adresa stranice na sekundarnoj memoriji.

Stranicenje na zahtev
(Demand paging)

1010011
1110100
1100001
1010110



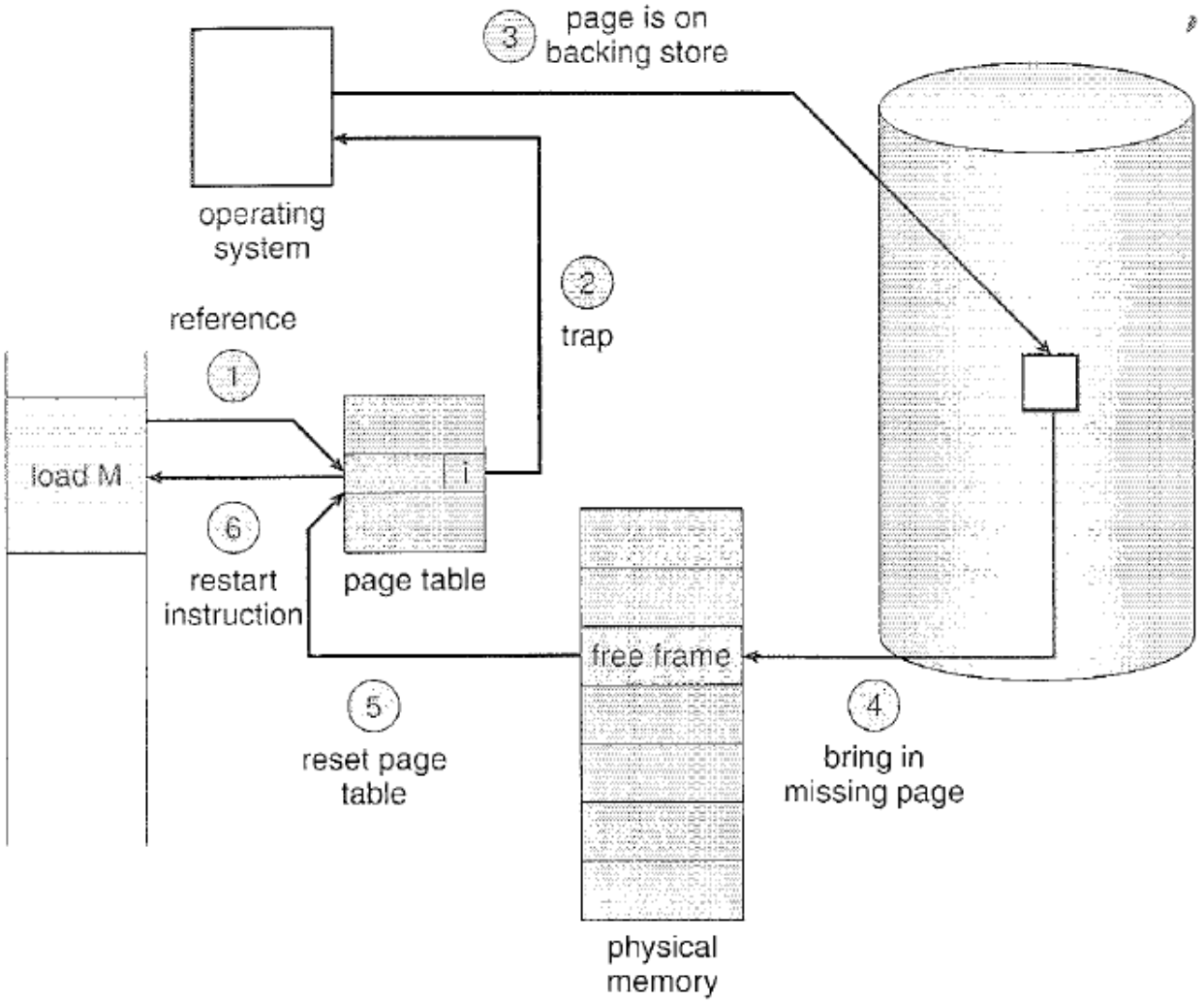
Stranicenje na zahtev
(Demand paging)

1010011
1110100
1100001
1010110

- Obrada page fault trapa
  - Proveravamo u tabeli koja se obicno cuva u okviru PCB da li je stranica validana ili ne.
  - Ako je stranica nevalidna (izvan adresnog protora) proces se prekida. Ako je stranica validna ali nije u GM ona se ucitava u GM.
  - Pronalazi se slobodan ram.
  - Ucitava se stranica sa sek. mem. u slobodan ram u fizickoj memoriji.
  - Nakon ucitavanja, modifikuje se tabela stranicenja kako bi pokazivala da je stranica u GM.
  - Restartuje se instrukcija cije je izvrsavanje prekinuto trapom. Proces sada moze da pristupi stranici kao da se vec nalazila u GM.
- Retko se desava da sa svakom novom instrukcijom proces pristupa novoj stranici.
  - Programi imaju osobinu lokalnosti referenci (locality of reference).

Stranicenje na zahtev
(Demand paging)

1010011
1110100
1100001
1010110



Stranicenje na zahtev
(Demand paging)

— Performanse stranicenja po zahtevu

- Znacajno utice na performanse sistema
- Efektivno vreme pristupa
  - ☑ Vreme pristupa memoriji – ma
  - ☑ Vreme obrade page fault trapa – pft
  - ☑ Verovatnoca pojave trapa – p

$$T_{\text{eff}} = (1 - p)T_{\text{ma}} + pT_{\text{pft}}$$

- Vreme obrade page fault trapa se sastoji iz tri osnovne komponente:
  - ☑ Opsluzivanje page fault trapa
  - ☑ Vreme ucitavanja stranice u GM
  - ☑ Restartovanje procesa
- Vreme pristupa i ucitavanja stranice iz sek. mem. u GM ima dominantan uticaj na efektivno vreme pristupa.

Stranicenje na zahtev
(Demand paging)

— Pojava page fault trapa dovodi do sledecih aktivnosti:

- Trap ka OS-u
- Sacuvaj sadrzaj registara i stanje procesa
- Utvrdi da li je interapt posledica page fault trapa
- Utvrdi da li je pristup stranici legalan i njenu lokaciju na sekundarnoj memoriji
- Izvrsi ucitavanja stranice u slobodan ram
  - ☑ Sacekaj u redu cekanja za sekundarnu memoriju dok se ne opsluzi zahtev za pristup
  - ☑ Sacekaj vreme potrebno za pristup sekundarnoj memoriji
  - ☑ pocni prenos u sobodan ram
- Dok ceka dodeli CPU nekom drugom procesu
- Prijem interapta da ju I/o sa sek. mem. zavrsen
- Sacuvaj stanje drugog procesa koji se izvrsavao
- Utvrdi da li je interapt potekao od sek. mem.
- Azuriraj tabelu stranicenja da je trazena stranica sada u GM
- Sacekaj da CPU bude ponovo dodeljen posmatranom procesu.

Zamena stranica
(Page replacement)

- Time sto u memoriju ucitavamo samo deo stranica programa, omogucavamo istovremeno izvrsavanje vise programa cime se povecava iskoriscenost sistema.
  - Sto je broj strnica programa koje drzimo u GM manji to mozemo vise programa istovremeno da izvrsavamo pri cemu svaki program zauzima deo GM.
  - Ako smestimo previse programa u GM moze se deisti da veoma cesto moramo da vrsimo svapovanje potrebnih stranica sa HDD u GM, cime gubimo dosta vremena.
  - Mora postojati kompromis izmedju broja programa u GM i broja strnica svakog programa koje se smestaju u GM.
- Deo memorije se koristi kao bafer pri I/O
  - Koliko memorije dodeliti programima a koliko za potrebe opsluzivanja I/O?!
- Sta raditi ako kada zelimo da svapujemo stranicu a nema slobodnih ramova?

Zamena stranica
(Page replacement)

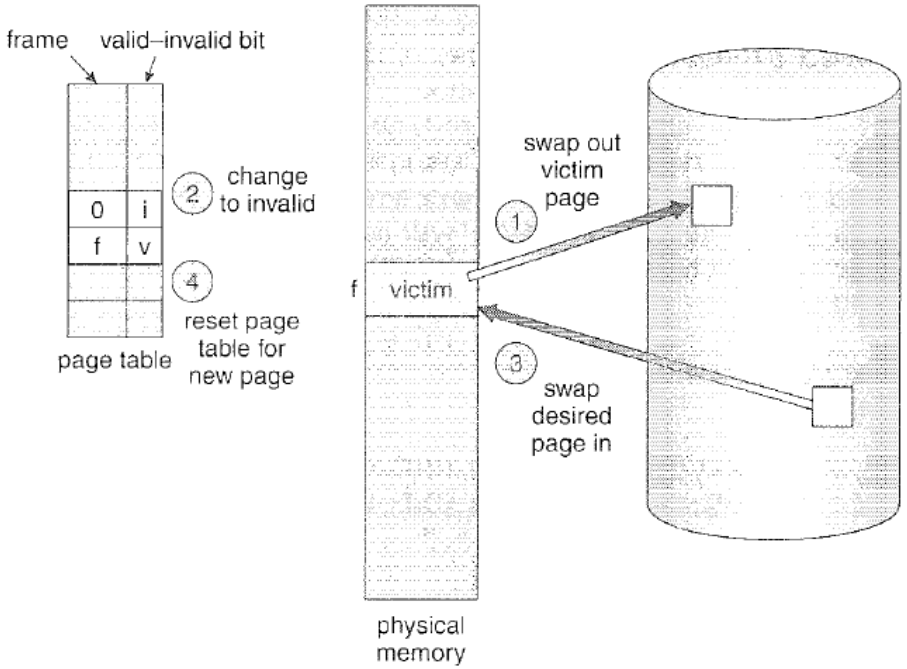
1010011
1110100
1100001
1010110

- Ako nema slobodnih ramova u glavnoj memoriji, mi mozemo:
  - Da prekinemo izvrsavanje procesa
    - ☑ Nije bas prava opcija jer time narusavamo izvrsavanje procesa a ne pomazemo njegovo izvrsavanje.
  - Da svapujemo ceo proces na HDD
  - Da svapujemo stranicu koja se trenutno ne koristi.
- Modifikujemo rutinu za opsluzivanje page fault trap-a
  - Nadji lokaciju trazene stranice na disku
  - Nadji slobodan ram
    - ☑ Ako ima slobodnih ramova, tada iskoristi neki od njih
    - ☑ Ako nema slobodnih ramova, odaberi ram uz pomoc algoritma za zamenu
    - ☑ Smesti sadrzaj odabranog rama na HDD i azuriraj tabele stranicenja
  - Ucitaj zeljenu stranicu u slobodan ram i azuriraj tabele stranicenja
  - Restartuj korisnicki proces

Zamena stranica
(Page replacement)

— Opisani proces zahteva dva pristupa ucitavanje sa HDD u GM

- Moze se ubrzati tako sto bi posmatrali da li je stranica od kada je ucitana u GM modifikovana ili ne.
  - ☑ Ako nije modifikovana, samo ucitavamo novu stranicu
  - ☑ Ako je modifikovana, izvrsavamo celi proces (staru stranicu smestamo na disk a ucitavamo novu)



Zamena stranica
(Page replacement)

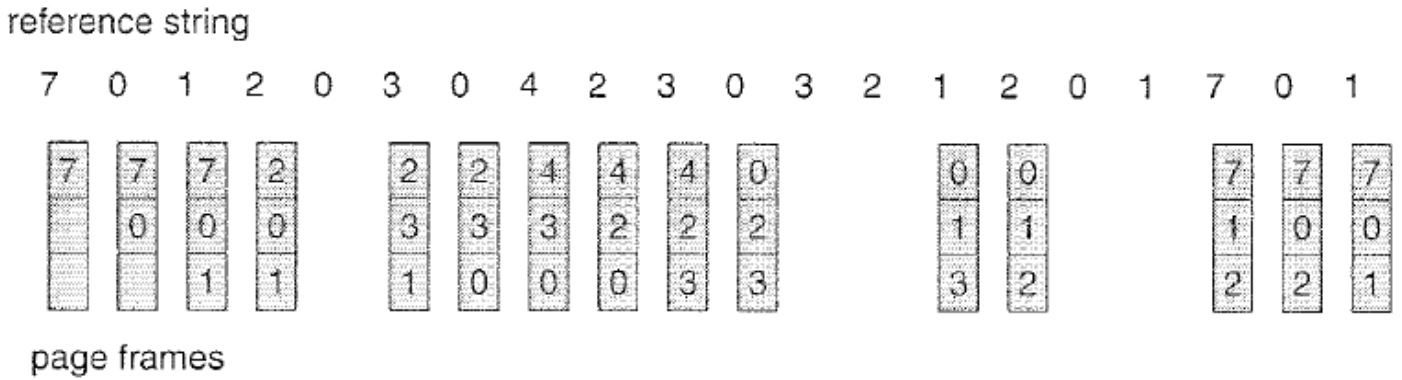
1010011
1110100
1100001
1010110

- Implementacija zamene stranica zahteva primenu:
  - Algoritama za dodelu ramova
    - ☑ Moramo odluciti koliko ramova cemo dodeliti svakom procesu
  - Algoritama za zamenu stranica
    - ☑ Moramo odluciti koje ramove cemo zameniti u GM.
- Performanse algoritama za zamenu stranica ispitujemo posmatranjem nekog niza pristupa odredjenim adresama.
  - Umesto niza adresa, posmatramo niz indeksa stranica jer pristup adresama u okviru jedne stranice ne zahteva promenu stranice ili zamenu.

Zamena stranica
(Page replacement)

— FIFO zamena stranica

- Uz svaku stranicu se pridruzuje vreme koje je proteklo od trenutka kada je ona ucitana u GM
- Kada se javi potreba za zamenom, menja se stranica koja je najvise vremena provela u GM
- Koristimo FIFO red, prilikom zamene uklanja se stranica sa vrha reda, a nova stranica se upisuje na kraj reda cekanja.
  - ☑ Nema potrebe za eksplicitnim tajmerom

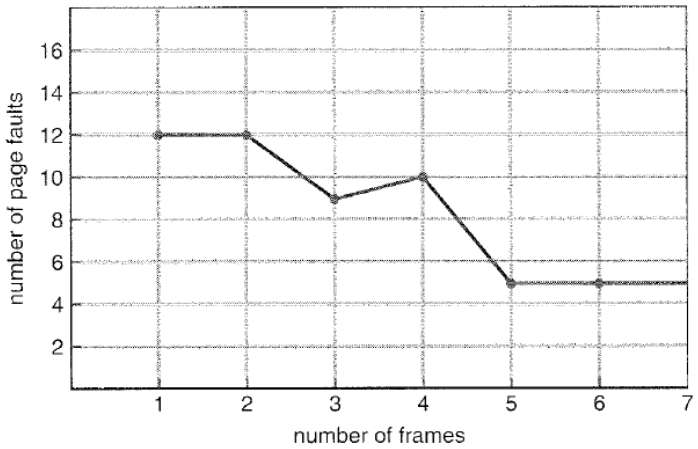


Zamena stranica
(Page replacement)

1010011
1110100
1100001
1010110

— FIFO zamena stranica

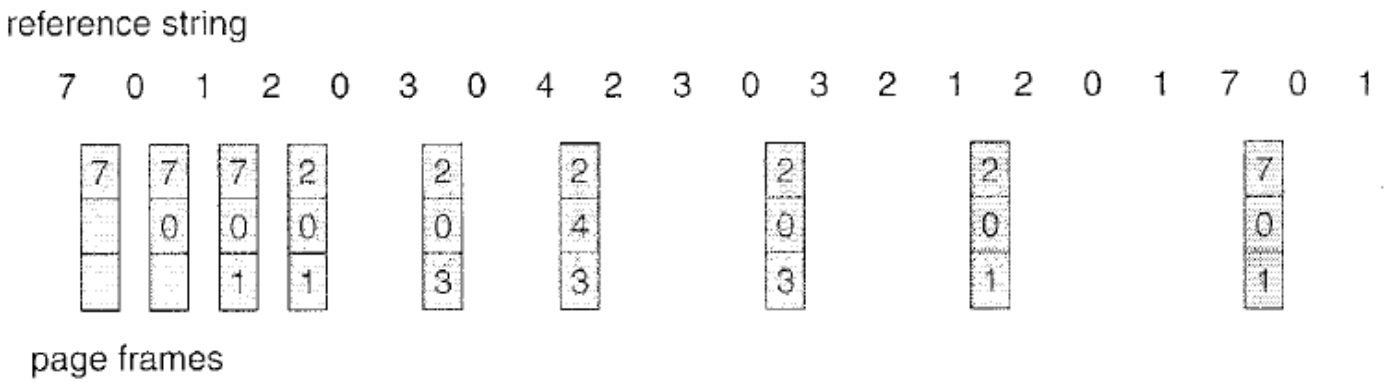
- Lak za razumevanja i implemetaciju
- Nema dobre performanse
- Pri zameni stranice ne vodi se racuna o sadrzaju i ucestanosti upotrebe stranice bez obzira na vreme upisa.
  - ☑ U odredjenim slucajevima dolazi do loseg izbora stranica za zamenu cime se povecava broj poziva page fault trapa.
- Belady-eva anomalija
  - ☑ Broj page fault trapova je veci kada se procesu dodeli 4 ramova nego kada se dodele 3 rama



Zamena stranica
(Page replacement)

— Optimalni algoritam za zamenu stranica

- **Zameni stranicu koja se nece koristiti najduzi vremenski period.**
  - ☑ Najmanji broj poziva page fault trapa
  - ☑ Nema Belady-eve anomalije
- Tezak za implemetaciju jer zahteva poznavanje buducih pristupa stranicama.
  - ☑ Koristi se samo ka oreferenca za poredjenje performansi.



Zamena stranica
(Page replacement)

— LRU zamena stranica

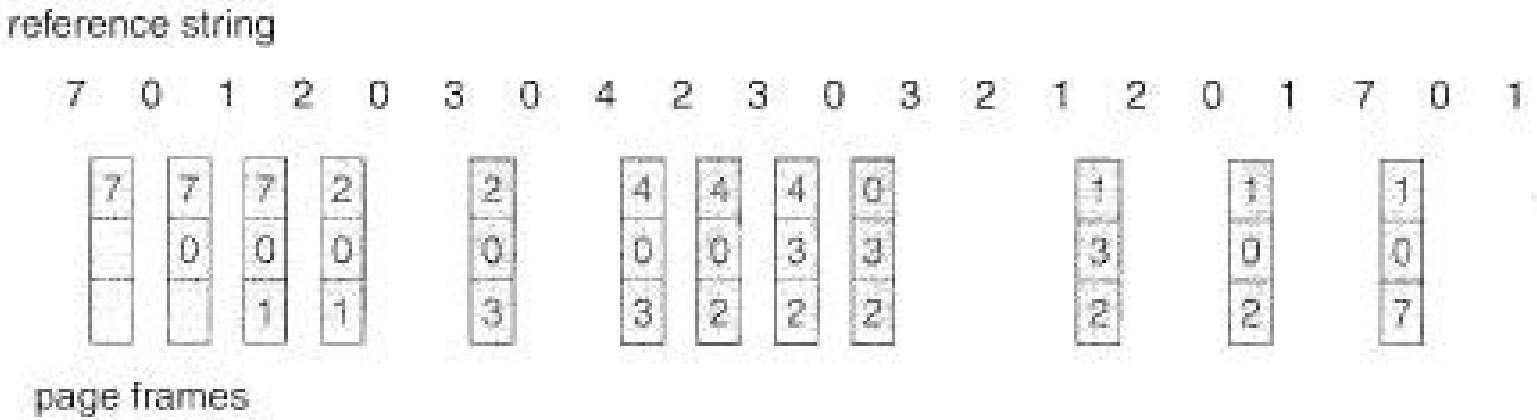
- Least recently used – LRU
- Menja se ona stranica koja je najmanje koriscena u nekom periodu vremena.
- Svakoj strnici se pridruzuje brojac koji pokazuje vreme od kada je ona zadnji put koriscena.
- Ima bolje performanse od FIFO
- Kako odrediti ram koji je najmanje koriscen
  - ☑ Svakoj stranici se dodeljuje brojac koji prati broj pristupa toj stranici
    - Menja se stranica sa najmanjim brojem pristupa
    - Moramo voditi racuna o prebacivanju brojaca (overflow)
  - ☑ Svaki put kad ase pristupi nekoj stranici ona se smesta na vrh magacina.
    - Na vrhu magacina se nalazi najskorije koriscena stranica.
    - Svaka zamena stranice u magacinu zahteva izmenu velikog broja pointera.

Zamena stranica
(Page replacement)

1010011
1110100
1100001
1010110

— LRU zamena stranica

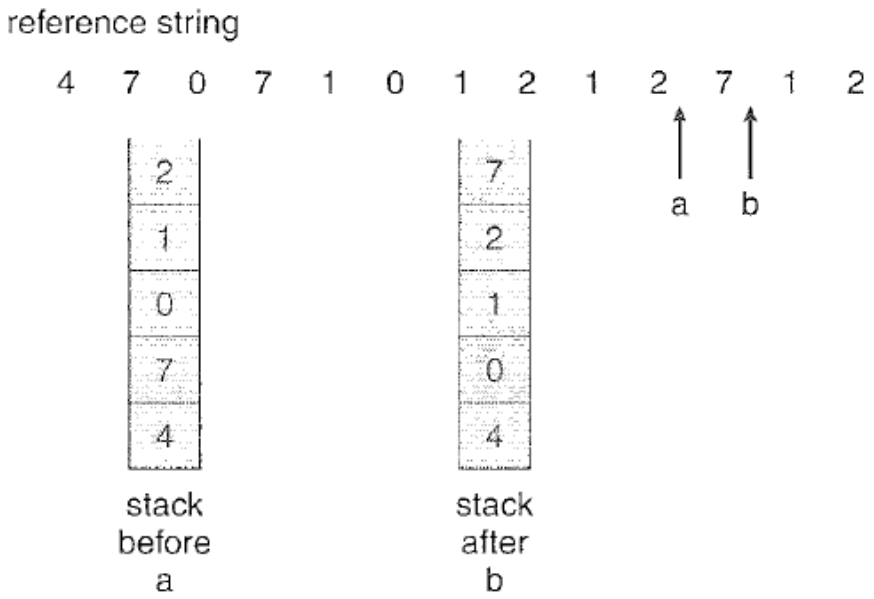
↳ Least recently used – LRU



Zamena stranica
(Page replacement)

— LRU zamena stranica

- LRU uz primenu magacina



- Problem je sto svaki pristup nekom podatku na nekoj stranici zahteva azuriranje brojaca ili magacina sto je veom neefikasno.

Zamena stranica
(Page replacement)

- Ako nije predvidjena hardverska podrska LRU, moramo koristiti FIFO.
- Obicno se svakoj stranici dodeljuje referentni bit (reference bit) koji pokazuje da li je data stranica modifikovana ili ne.
  - OS inicijalno postavlja sve referentne bitove na 0.
  - Posle odredjenog vremena mozemo da utvrdimo koje su stranice koristene a koje ne, iako ne znamo redosled pristupa tim stranicama.
  - Ovo je osnova za mnoge modifkovane LRU algoritme.

Zamena stranica
(Page replacement)

1010011
1110100
1100001
1010110

— Additional reference bits algorithm

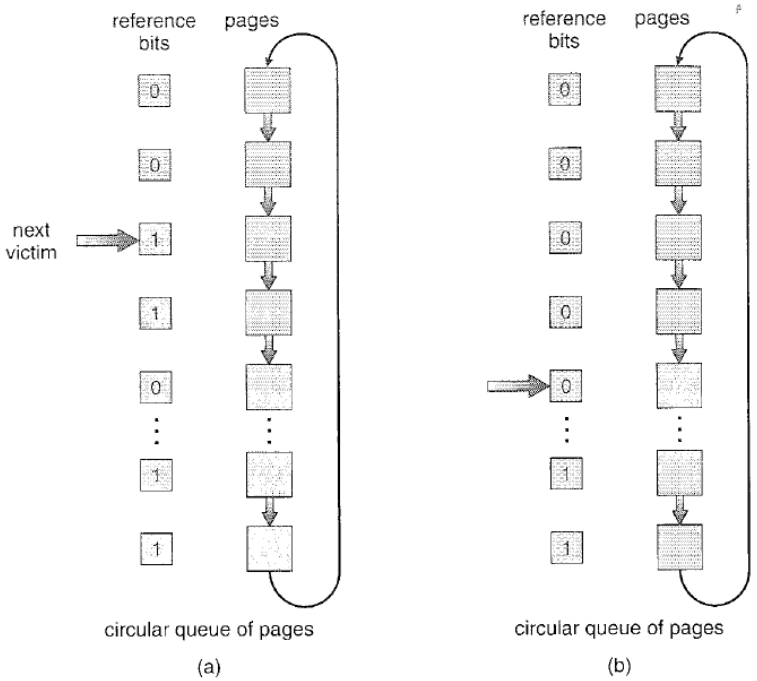
- Algoritam sa dodatnim referentnim bitovima
  - ☑ Umesto jednog referentnog bita, svakoj stranici se pridruzuje niz bita duzine *n*
  - ☑ U određenon vremenskom intervalu se proverava vrednost referentnih bita stranica u GM i njhiove vrednosti se upisuju u odgovarajuće vektore na pozicijama najveće težine, ostale vrednosti se (šiftaju) cirkularno pomeraju udesno i biti najmanje težine se brisu
  - ☑ Nizovi bita za svaku stranicu predstavljaju istoriju pristupanja toj stranici
  - ☑ Menja se ona stranica sa najmanjim brojem koji je predstavljen nizom bita.

Zamena stranica
(Page replacement)

1010011
1110100
1100001
1010110

— Second-chance algorithm

- Algoritam druge sanse
  - ☑ Modifikacija FIFO algoritma
  - ☑ Zamenjujemo prvu stranicu u redu ciji je referentni bit 0
  - ☑ Ako su svi referentni biti postavljeni na 1, algoritam postaje FIFO
    - Svaki put kada proveri vrednost referentnog bita neke stranice, postavlja ga na 0.



Zamena stranica
(Page replacement)

— Enhanced second-chance algorithm

- Poboljsani algoritam druge sanse
- Posamtramo uredjeni par (reference bit, modify bit)
  - ☑ Imamo 4 opcije
    - (0,0) – stranici nije pritupano niti je modifikovana → najbolja stranica za zamenu
    - (0,1) – stranici nije skoro pristupano ali je modifikovana → zahteva njeno smestanje na HDD pri zameni
    - (1,0) – stranici je pristupano ali nije modifikovana → verovatno ce biti ponovo koriscena
    - (1,1) – stranici je pristupano i njen sadrzaj je menjan
  - ☑ Funkcionise na slican nacin, s tim sto se prednost daje stranicama koje su modifikovane cime se smanjuje broj I/O operacija

Dodela ramova
(Frame allocation)

1010011
1110100
1100001
1010110

- Kako raspodeliti fizicku memoriju konacne velicine izmedju vise procesa?
- Ogranicenja se odnose na maksimalni i minimalni broj stranica koje se mogu dodeliti procesima.
  - Maksimalni broj stranica je odredjen fizickom velicinom memorije.
  - Minimalni broj stranica je doredjen zeljenim performansama koje udredjene ucestanoscu pojavljivanja page fault trapova.

Dodela ramova
(Frame allocation)

1010011
1110100
1100001
1010110

- Ravnopravna dodela
  - Najjednostavniji nacin da se *m* ramova raspodeli skupu od *n* procesa jest da se svakom procesu dodeli podjednak broj ramova *m* / *n*.
- Proporcionalna dodela
  - Svakom proesu se dodeljuje deo memorije srazmerno njegovoj velicni.
  - Neka je velicina logicke memorije procesa p_i oznacena sa s_i.
  - Uvodimo velicinu

$$S = \sum s_i$$

  - Tada je broj ramova koje dodeljujemo procesu p_i, a_i, odredjen sa:

$$a_i = s_i / S \times m.$$

  - Procesi dele ramove srazmerno svojim potrebama.
- U oba slucaja svi procesi se tretiraju sa istim prioritetom.
  - Postoji potreba da se procesima sa visim prioritetom dodeli vise ramova

Dodela ramova
(Frame allocation)

1010011
1110100
1100001
1010110

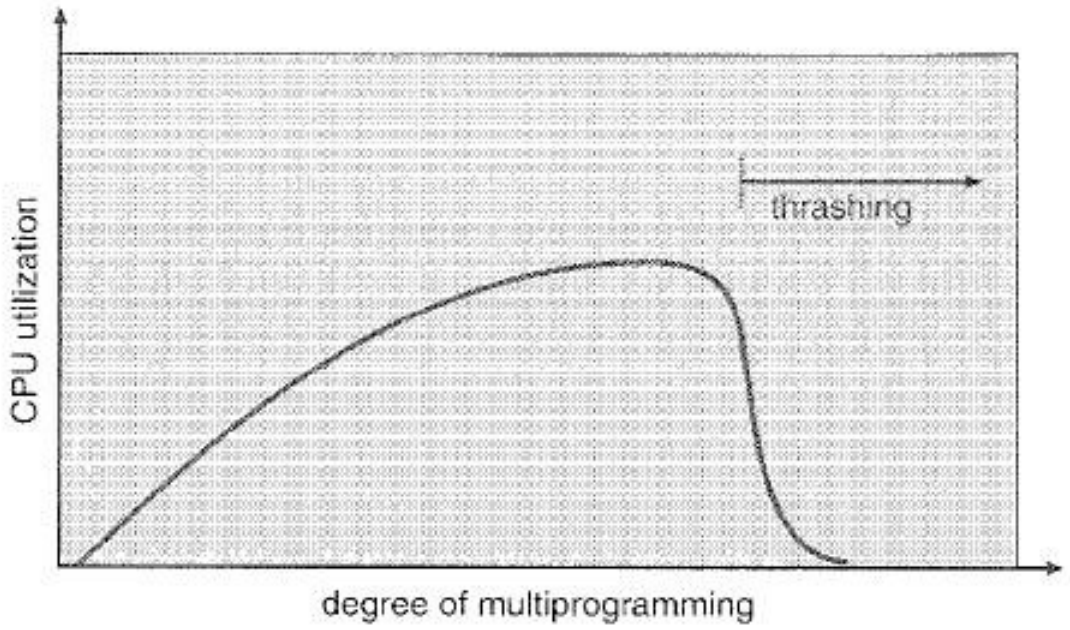
— Drugi bitan faktor koji utice na broj ramova koji se dodeljuju procesima jeste priroda algoritama za zamenu stranica.

- Globalni algoritmi za zamenu stranica omogucavaju da se pri zameni bira bilo koja od stranica bez obzira da li ona tenutno priada nekom drugom procesu.
  - ☑ Omogucava da se broj ramova koji su dodeljuje procesu menja tokom izvrsavanja.
  - ☑ Broj ramova dodeljenih jednom procesu zavisi od page fault ucestanosti svih procesa.
  - ☑ Reyultuje boljom iskorisc4enoscu sistema.
- Lokalni algoritmi za zamenu stranica vrse zamenu stranica samo iz skupa strania koje su dodeljene tom pocesu.
  - ☑ Broj ramova koji su dodeljeni procesima ostaje fiksan.

Thrashing

- Moze se desit da se broj ramova koji su dodeljni procesu smanji ispod odredjene granice.
  - Prekida se izvrsavanje procesa i oslobadjaju se njegove preostale opcije.
  - U opstem slucaju, ako proces nema dovoljno ramova za izvrsavanje, on ce provoditi vise vremena u zameni stranica (paging) nego izvrsavajuci se na CPU.
  - Pojava poznata pod nazivom **thrashing**.

Thrashing



Thrashing

1010011
1110100
1100001
1010110

Praznjenje
(Thrashing)

1010011
1110100
1100001
1010110

Domaci zadatak

1010011
1110100
1100001
1010110