OPERATIVNI SISTEMI

Slajdovi su kreirani na osnovu knjige "Operativni sistemi, principi unutrašnje organizacije i dizajna, 7. izdanje", William Stallings, CET, Beograd, 2013.

Virtuelna memorija

Karakteristike upravljanja memorijom

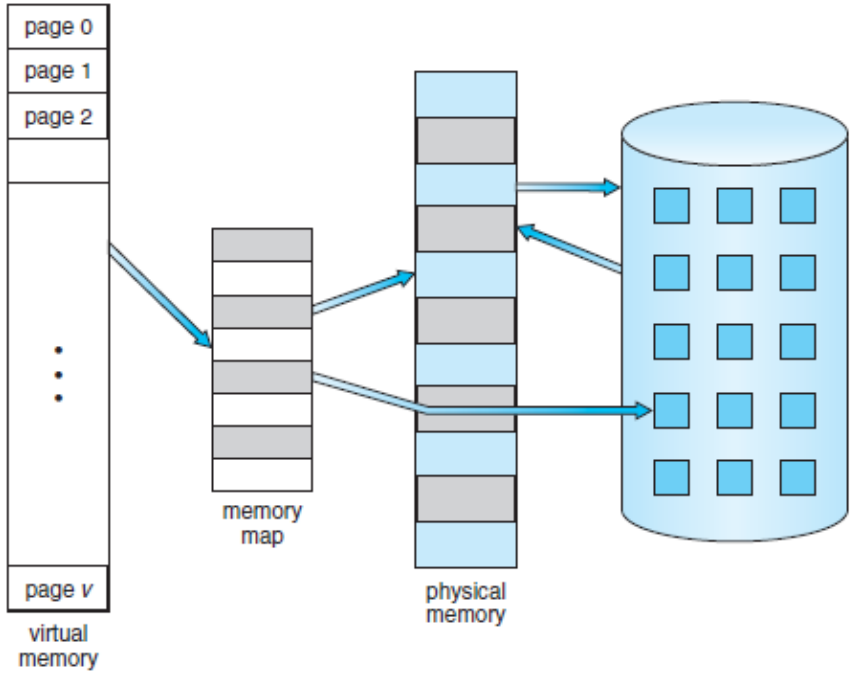
- Memorijske reference su logičke adrese koje se dinamički prevode u fizičke adrese u toku izvršavanja
  - Proces može da bude ubacivan i izbacivan u/iz glavne memorije u toku izvršavanja
  - Svaki put može biti smešten u različit deo memorije
- Proces se može izdeliti na manje delove koji ne moraju zauzimati uzastopne lokacije u memoriji

Karakteristike upravljanja memorijom

- Ako su **obe** karakteristike prisutne
  - Tada nije neophodno da svi delovi procesa istovremeno budu u glavnoj memoriji u toku izvršavanja
  - Ne koriste se svi delovi procesa uvek i jednako često
  - Nema razloga da delovi koji nisu u upotrebi zauzimaju memoriju

Virtuelna memorija

- Stvarna memorija
  - Glavna memorija
- Virtuelna memorija
  - Deo diska (fajl ili posebna particija) namenjen za smeštanje delova onih procesa koji se trenutno izvršavaju
  - Omogućuje efikasno multiprogramiranje
  - Sa stanovišta korisnika umanjuje ograničenja glavne memorije

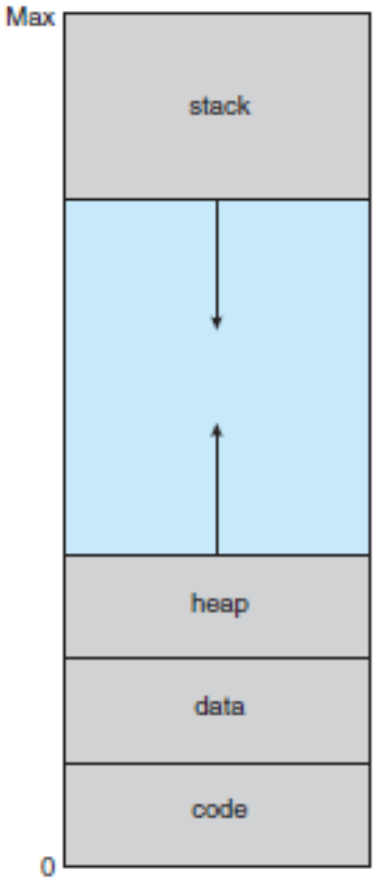


Prednosti nove strategije upravljanja memorijom

- Veći broj procesa može da se održava u glavnoj memoriji
  - Od svakog procesa se učitava samo deo, pa više različitih procesa može da bude učitano
  - Sa većim brojem procesa veća je šansa da će u svakom trenutku bar neki od njih biti u stanju Spreman
- Proces može biti veći od ukupne glavne memorije
- Sa stanovišta korisnika
  - Nova strategija upravljanja memorijom je transparentna
  - Korisnik vidi virtuelni adresni prostor sa kontinualnim logičkim adresama koje kreću od 0

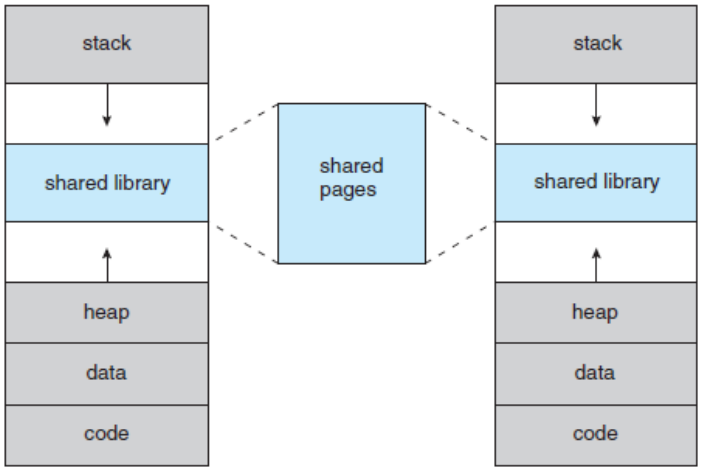
Prednosti nove strategije upravljanja memorijom

- *Heap* i *stack* memorija dodeljena procesu dinamički se menja
  - Prostor za ovu memoriju je deo virtuelnog adresnog prostora
  - Prostor zahteva fizičke stranice u memoriji samo ako je popunjen



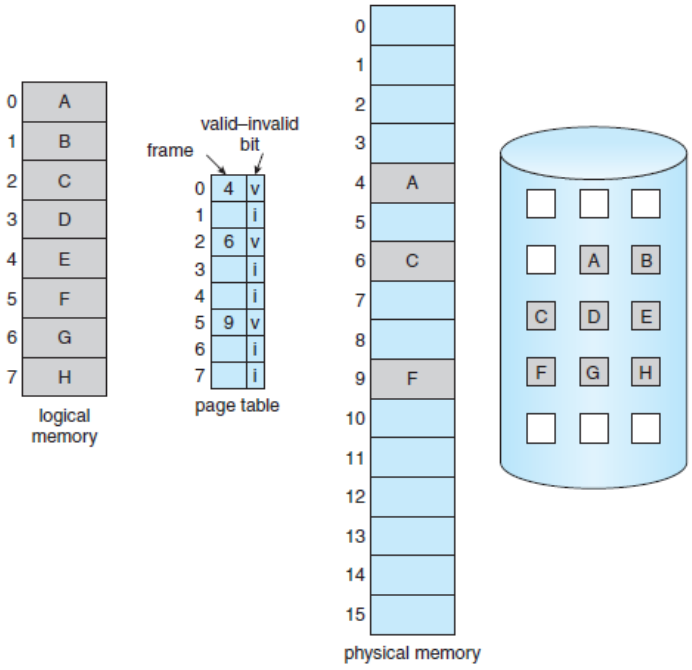
Prednosti nove strategije upravljanja memorijom

- Deo memorije može biti deljen od strane više procesa
  - Ista stranica se mapira u virtuelni prostor više procesa
  - Primenjuje se za
    - Sistemske biblioteke koje deli više procesa
    - Komunikaciju između procesa putem deljene memorije
    - Deljenje istog koda od strane više instanci istog procesa



Izvršavanje procesa

- Rezidentni skup
  - delovi procesa koji su u glavnoj memoriji u određenom trenutku
  - Postoji indikacija za svaku stranicu da li je u memoriji



Izvršavanje procesa

- Ako se pri izvršavanju procesa referencira stranica koja je u glavnoj memoriji, pristupa se traženoj lokaciji i nastavlja izvršavanje

Izvršavanje procesa

- Šta ako se referencira stranica koja nije u glavnoj memoriji

Izvršavanje procesa

- Ako se referencira stranica koja nije u glavnoj memoriji
  - To je greška stranice (eng. *page fault*)
  - Hardver utvrđuje da je za tu stranicu postavljen invalid bit
  - Šalje se zahtev OS da dobavi potrebnu stranicu

Izvršavanje procesa

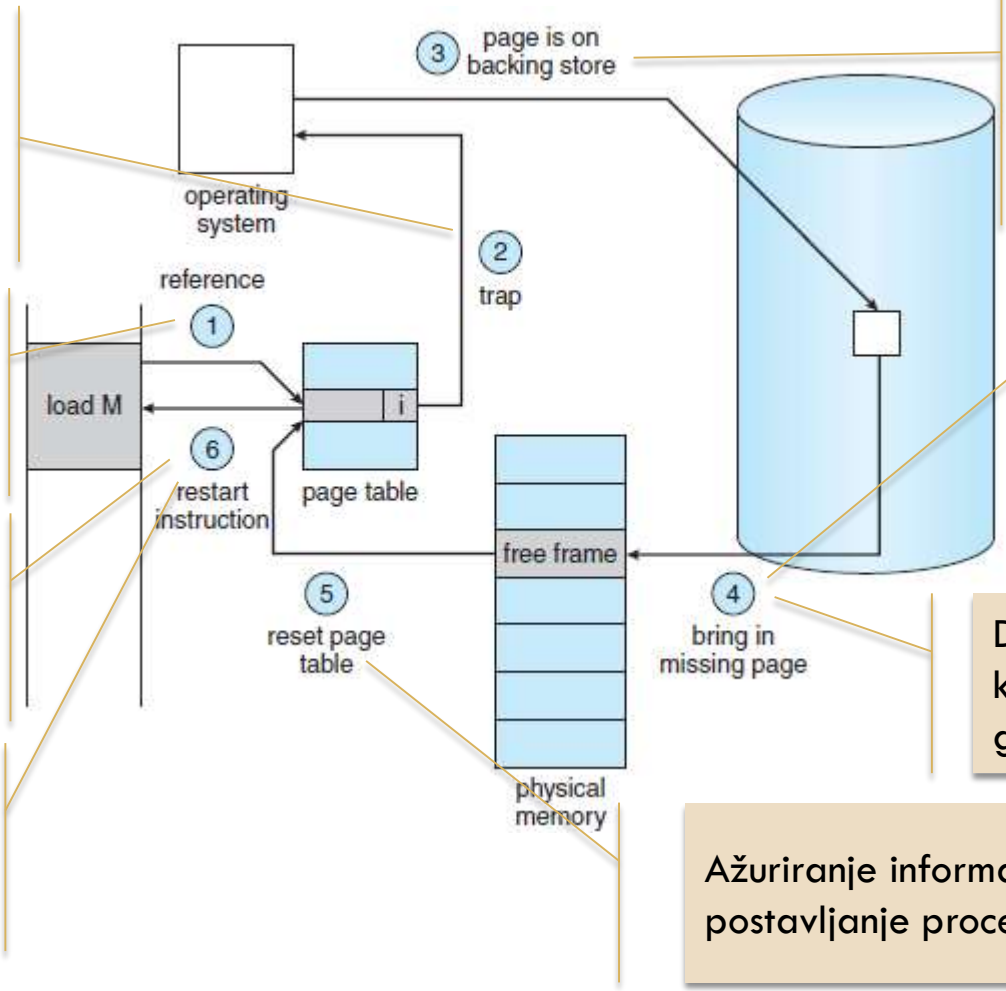
- Procedura za obradu greške stranice

Šalje se zahtev OS da dobavi stranicu. Proces se postavlja u stanje blokiran

Procesor generiše prekid pri referenciranju nevalidne stranice

Čeka se na procesor da opet rasporedi proces

Ponovo se izvršava instrukcija koja je izazvala grešku stranice



OS šalje zahtev disku za učitavanje potrebne stranice (čekanje u redu na uređaj)

Dok se čeka na uređaj, raspoređuje se drugi proces

Disk će postaviti prekid kada prebaci stranicu u glavnu memoriju

Ažuriranje informacije o stranici i postavljanje procesa u stanje Spreman

Efikasnost virtuelne memorije

- Ako potrebni deo procesa nije u glavnoj memoriji
  - mora se dobaviti sa diska
- Ako nema mesta u glavnoj memoriji za novi deo
  - deo nekog procesa se mora izbaciti iz glavne memorije

Efikasnost virtuelne memorije

- Zavisi od verovatnoće pojave greške stranice

$$0 \leq p \leq 1$$

- Prosečno efektivno vreme pristupa

$$t = (1 - p) \times mat + p \times pft$$

mat – vreme pristupa memoriji

pft – vreme obrade greške stranice

Efikasnost virtuelne memorije

- Tri grupe operacija pri obradi greške stranice:
  1. Obrada prekida za grešku stranice
  2. Dobavljanje stranice sa diska
  3. Restartovanje instrukcije
- Druga stavka je vremenski zahtevna

Efikasnost virtuelne memorije

- Ako je trajanje
  - Obrade greške stranice 8 ms
  - Pristupa memoriji 200 ns
- Tada je prosečno efektivno vreme pristupa

$$t = (1 - p) \times 200\,ns + p \times 8\,ms$$
$$= (1 - p) \times 200\,ns + p \times 8{,}000{,}000\,ns$$
$$= 200\,ns + p \times 7{,}999{,}800\,ns$$

- Vreme pristupa je direktno proporcionalno učestalosti greške stranice

Efikasnost virtuelne memorije

- Ako je $p = 0.001$
  - Jedno od hiljadu referenciranja izaziva grešku stranice
- Tada je $t = 8.2\ \mu s$
- To je usporenje od 40 puta u odnosu na vreme potrebno samo za pristup radnoj memoriji
- Da bi usporenje bilo manje od 10%

$$220 > 200 + 7{,}999{,}800 \times p$$

$$20 > 7{,}999{,}800 \times p$$

$$p < 0.0000025$$

  - Manje od jedno od 400000 referenciranja sme da izazove grešku stranice ako želimo dobre performanse

Brbljanje (*Thrashing*)

- Kada sistem provodi najveći deo vremena razmenjujući delove između stvarne i virtuelne memorije umesto u izvršavanju instrukcija
- Izbegavanje brbljanja
  - OS izbacuje stranicu za koju utvrdi da je najmanja verovatnoća da će biti uskoro referencirana
  - Predviđanje se vrši na osnovu nedavne istorije

Princip lokalnosti

- Program i reference podataka unutar procesa teže da se grupišu

Princip lokalnosti

- Tokom dužeg perioda menjaju se delovi procesa koji se koriste
- U kratkom intervalu procesor uglavnom radi sa malim i ograničenim skupom adresa
- Samo nekoliko delova procesa je potrebno u kratkom vremenskom periodu

Princip lokalnosti

- Zašto postoji lokalnost u programima?
  1. Izvršavanje programa je skoro uvek sekvencijalno
     - Naredna adresa je najčešće ona koja sledi trenutnoj
     - Izuzetak su grananja i pozivi funkcija
  2. U kratkom periodu program lokalizovan na određenu funkciju
     - U funkciji se pristupa ograničenom skupu njenih parametara, lokalnih promenljivih i podskupa globalnih promenljvih
     - Retko u programu postoji dugačak uzastopan niz poziva funkcija
     - Uglavnom program ograničen na uzak nivo dubine poziva funkcija

Princip lokalnosti

- Zašto postoji lokalnost u programima?
  3. Zbog iterativnih delova koda
     - Petlje se uglavnom sastoje od malog broja instrukcija koje se ponavljaju mnogo puta
     - Obrada je ograničena na mali susedni deo adresa
  4. Često se podaci čuvaju u strukturi koja čuva podatke u uzastopnim lokacijama
     - Podaci kojima se pristupa su blisko locirani
     - Npr. rad sa nizovima

Princip lokalnosti

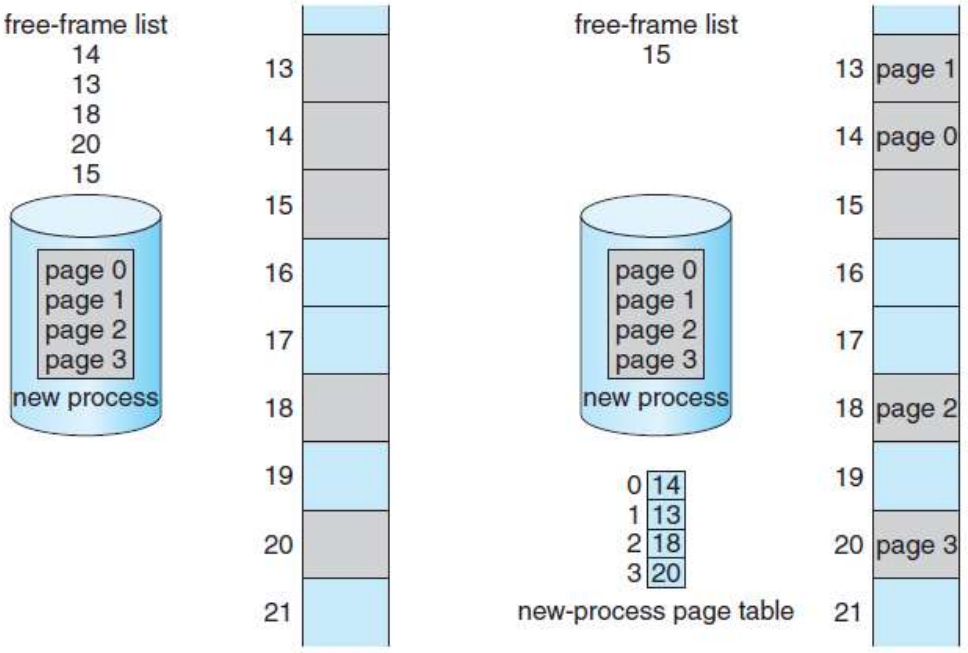
- Zato što postoji lokalnost, moguće je napraviti pretpostavku koji delovi procesa će biti potrebni uskoro i tako izbeći brbljanje
- Zbog principa lokalnosti, virtuelna memorija je efikasna

Podrška za virtuelnu memoriju

- **Hardverska**
  - Hardver procesora mora da podrži adresiranje stranica/segmenata
- Softverska
  - OS mora da pomera stranice/segmente između glavne i sekundarne memorije

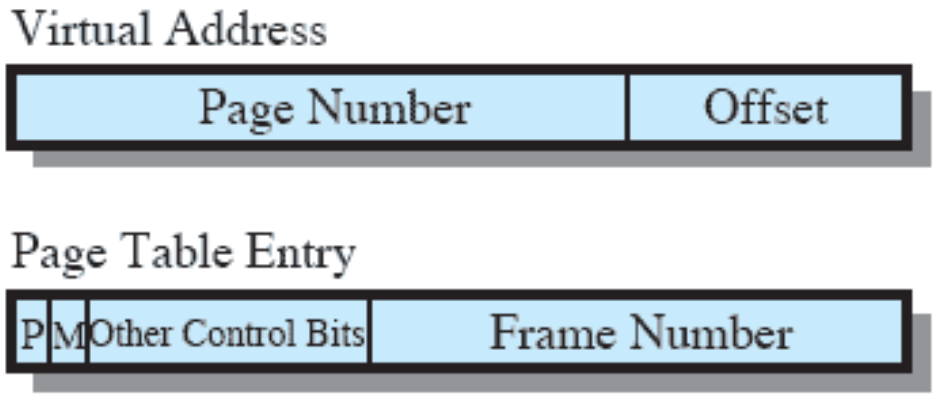
Straničenje

- Proces je podeljen na stranice
  - Sve stranice su smeštene na disku
  - Deo stranica je u glavnoj memoriji
- Tabela stranica
  - Evidencija u kojim okvirima su stranice smeštene
  - Svaki proces ima svoju



Stavka tabele stranica

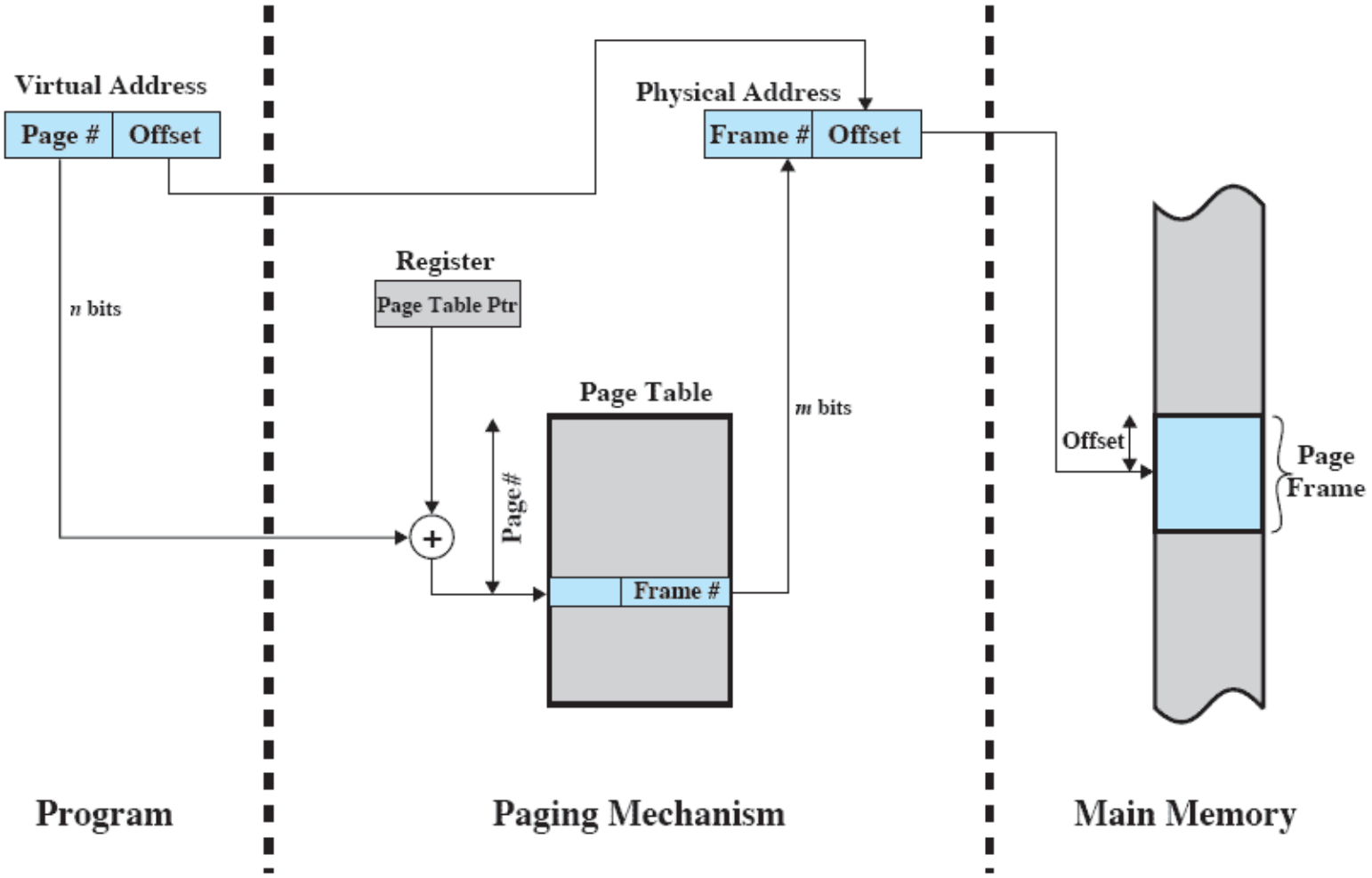
- Podaci potrebni za evidenciju stranice
  - Broj okvira ako je stranica u glavnoj memoriji
  - P – da li je stranica prisutna u memoriji
  - M – da li je stranica menjana otkako je učitana u glavnu memoriju. Ako jeste, potrebno je pre izbacivanja upisati izmenu na disk



Struktura tabele stranica

- Pri izvršavanju procesa, najčešće se početna adresa tabele stranica drži u registru
  - Ova adresa se čuva u upravljačkom bloku procesa
  - Pri komutaciji se ubacuje u procesorski registar
- Broj stranice iz virtuelne adrese se koristi za indeksiranje tabele stranica i preuzimanje broja okvira
- Broj okvira se kombinuje sa brojem pomeraja za dobijanje fizičke adrese

Adresiranje sa virtuelnom memorijom



TLB Bafer

- Ako se koristi virtuelna memorija, pri svakom referenciranju podataka trebaju dva pristupa memoriji:
  - Jedan da se preuzme stavka iz tabele stranica
  - Jedan da se pročita vrednost sa izračunate fizičke adrese
- Da bi se prevazišao ovaj problem koristi se specijalni brzi keš za stavke tabele stranica
  - Zove se Translation Lookaside Buffer (TLB)
  - Sadrži stavke tabele stranica koje su najskorije korišćene
  - U savremenim procesorima najčešće postoji više nivoa TLB
    - Hijerarhijski organizovani po veličini i brzini
    - Svaki sledeći nivo veći, ali sporiji
  - Obično postoje odvojeni TLB za adrese koje se odnose na instrukcije i one koje se odnose na podatke

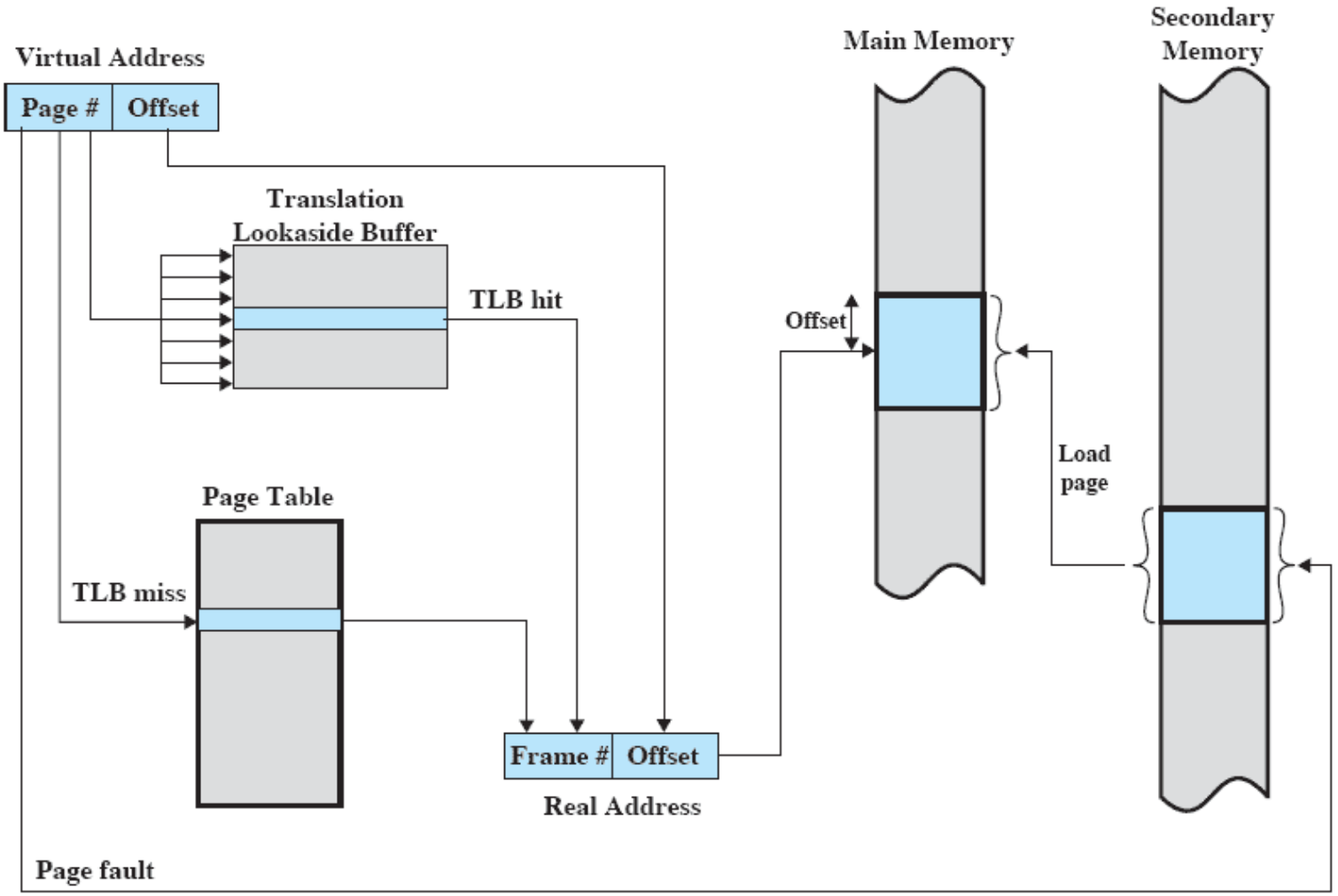
Referenciranje uz korišćenje TLB

- Pri referenciranju na osnovu virtuelne adrese
  - procesor najpre proverava TLB
- Ako se stavka za referenciranu stranicu nalazi u TLB
  - preuzima se broj okvira i formira stvarna adresa
- Ako se stavka za referenciranu stranicu ne nalazi u TLB
  - pristupa se stavci u tabeli stranica

Referenciranje uz korišćenje TLB

- Proverava se da li je stranica u glavnoj memoriji
  - Ako nije, generiše se greška stranice (*page fault*)
  - OS je zadužen da prebaci stranicu iz spoljne u radnu memoriju i da ažurira tabelu stranica
- TLB se ažurira da uključi referenciranu stavku tabele stranica

Referenciranje uz korišćenje TLB

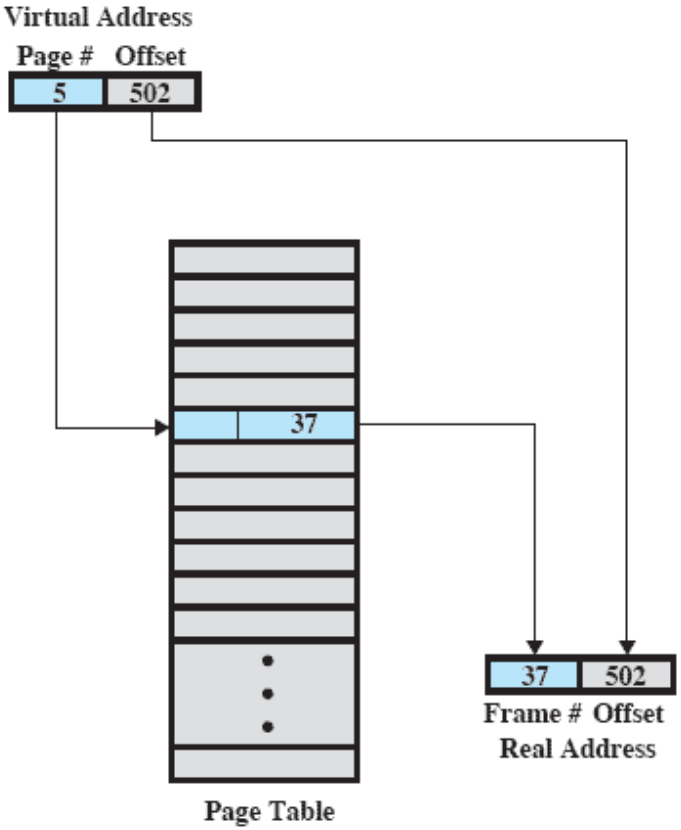


TLB – asocijativno preslikavanje

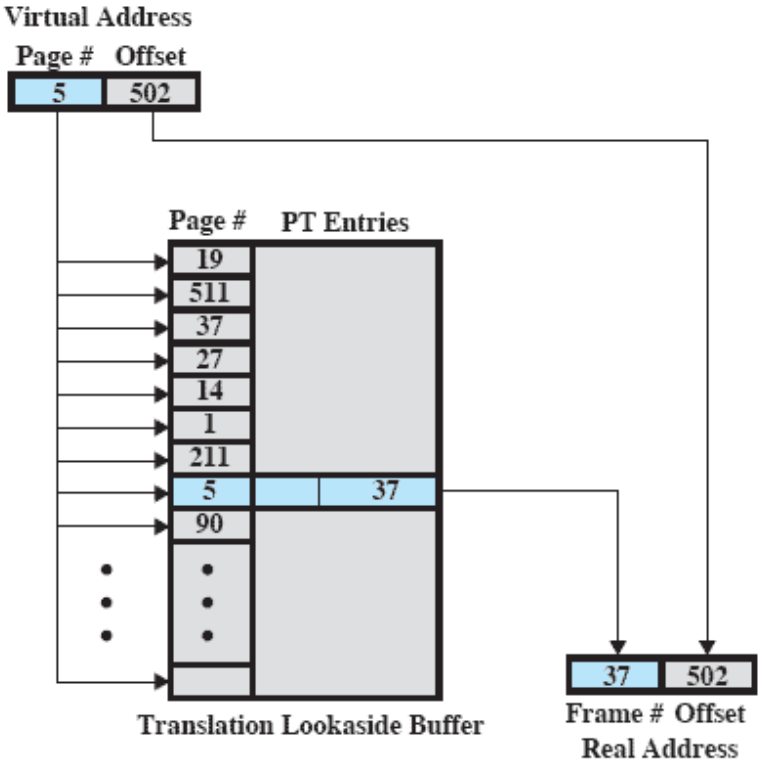
- TLB sadrži samo neke od stavki tabele stranica
  - ne može se stavka jednostavno pronaći indeksiranjem na osnovu broja stranice
  - stavka TLB mora da sadrži broj stranice i kompletnu stavku tabele stranica
- Procesor može istovremeno da ispita više stavki TLB da pronađe odgovarajuću stranicu
  - ova tehnika se zove asocijativno preslikavanje

TLB asocijativno preslikavanje

Direktno preslikavanje



Asocijativno preslikavanje



- N mogućih lokacija u kojima može biti tražena stavka
- Zavisno od procesora, N može biti od 1 do cele veličine TLB

Stavke TLB bafera

- Stavke u TLB baferu
  - mogu da se odnose na sve procese u sistemu
  - da sadrže podatke o stranicama aktivnog procesa
- Ako se stavke odnose na različite procese
  - U svakoj stavci je potrebna informacija na koji se proces odnosi
  - Pri referenciranju, pronalaženje je uspešno samo ako se nađe stavka koja se odnosi na referenciranu stranicu konkretnog procesa
- Ako TLB ne podržava stavke koje se odnose na različite procese
  - Pri komutaciji procesa mora se TLB isprazniti

TLB bafer – procenat pogodaka

- Procenat pogodaka – koji deo ukupnih referenciranja stranica se pronađe u TLB baferu
- Cilj je ostvariti što veći procenat pogodaka
  - Tada nije potrebno pristupati tabeli stranica
  - Time se ubrzava pristup referenciranoj adresi

Vreme pristupa memoriji

- Efektivno vreme pristupa memoriji
  - Ukupno vreme potrebno da se pročita/upiše podatak u memoriju
  - Vreme uključuje pristup tabeli stranica i traženoj memorijskoj lokaciji
- Primer
  - 100 ns za pristup radnoj memoriji
  - Ako se stranica ne pronađe u TLB, efektivno vreme pristupa je 200 ns
  - 100 ns za pristup tabeli stranica i 100 ns za pristup traženoj lokaciji
- Prosečno efektivno vreme pristupa
  - Uzima u obzir procenat pogodaka
  - Ako je procenat pogodaka 80% i 100 ns vreme pristupa memoriji

$$t = 0.80 \times 100 + 0.20 \times 200 = 120\,ns$$

Tabela stranica

- Tabela stranica može da bude prevelika da bi se čuvala u glavnoj memoriji
- Jedna varijanta rešenja je smeštanje tabele stranica u virtuelnu memoriju
  - tabela stranica se deli u stranice
- Pri izvršavanju procesa deo tabele stranica mora da bude u glavnoj memoriji
  - potrebno je pristupiti stavci koja se odnosi na stranicu koja se trenutno referencira

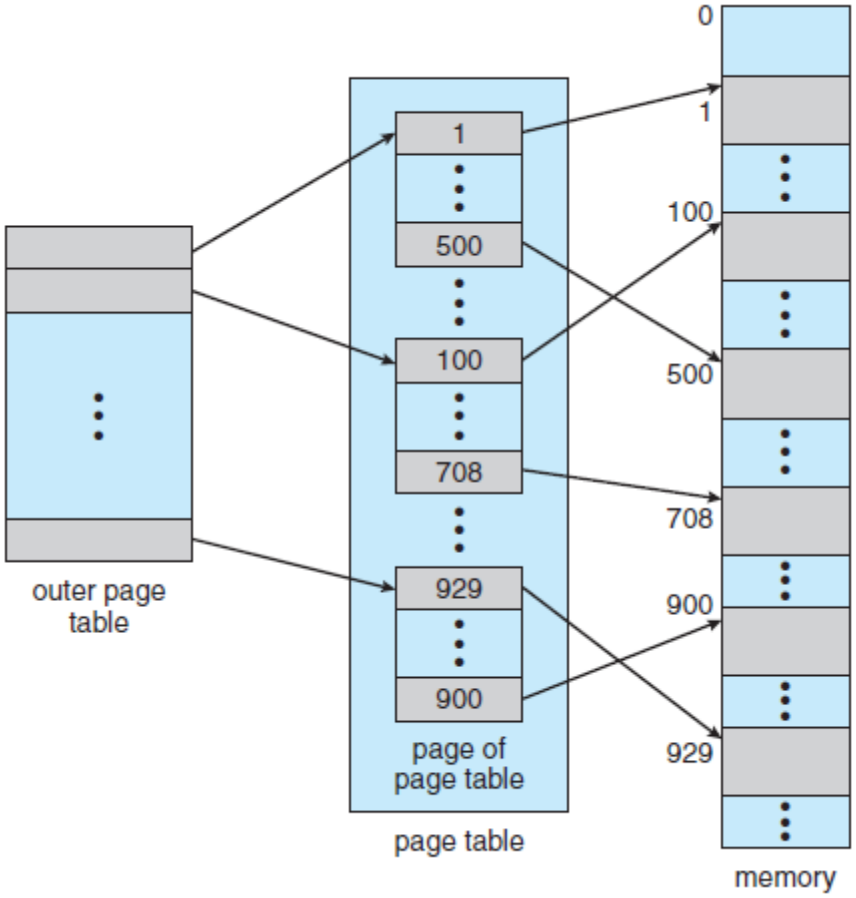
Hijerarhijska tabela stranica

- Druga varijanta za prevelike tabele stranica
- Mala osnovna tabela stranica
- Stavke te tabele stranica pokazuju na stranicu u drugoj tabeli stranica
- U toj drugoj tabeli stranica su podaci o stranicama procesa
- Može se tako realizovati u N nivoa
  - Tabele stranica na svim nivoima se čuvaju u radnoj memoriji
  - Najčešće ima 3 nivoa

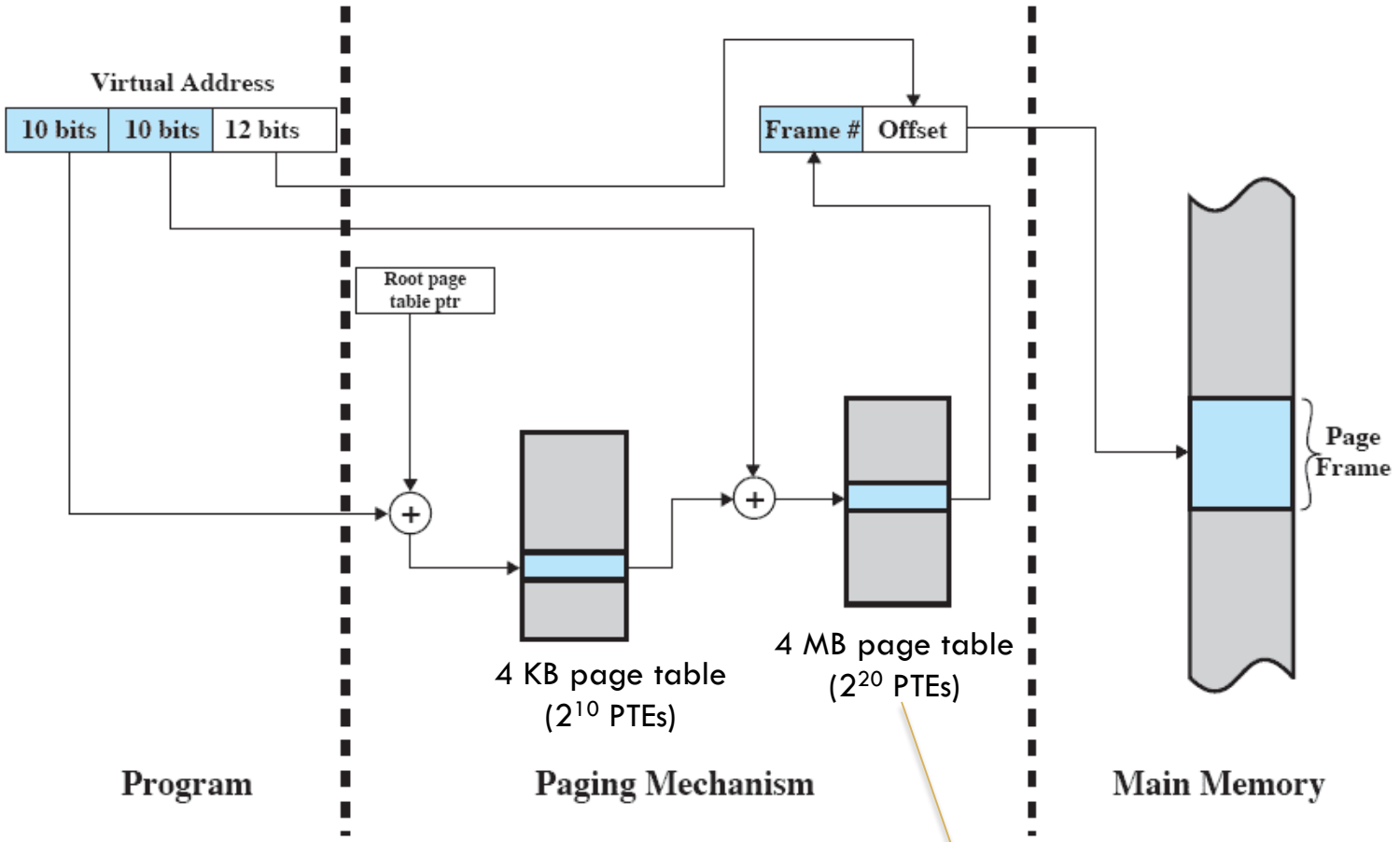
Hijerarhijska tabela stranica

- OS ne kreira sve delove tabela stranica na nižim nivoima
  - Npr. ne kreiraju se stavke tabele stranica za stranice iz heap memorije koje se ne koriste
  - Na ovaj način se troši manje radne memorije u odnosu na klasičnu tabelu stranica u jednom nivou

Hijerarhijska tabela stranica u 2 nivoa



Hijerarhijska tabela stranica u 2 nivoa



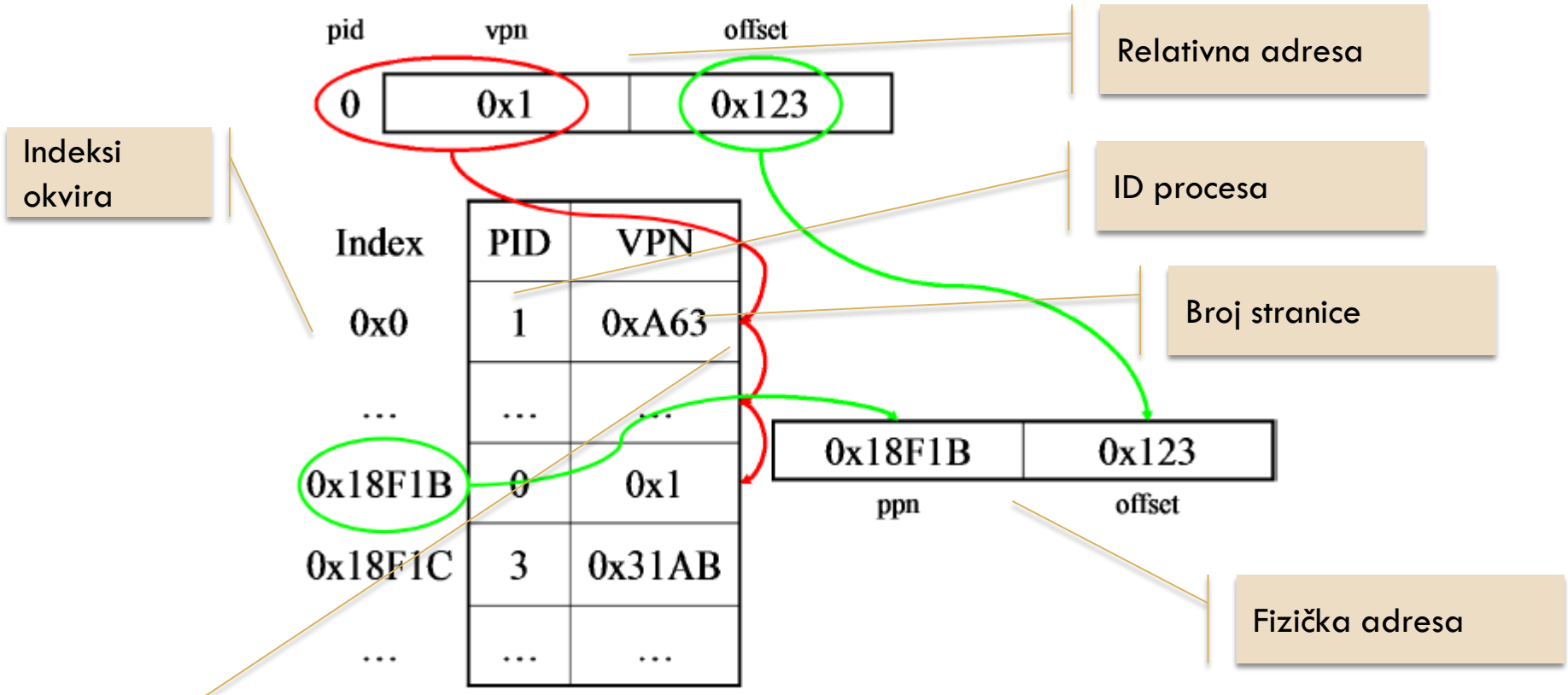
2^{10} stranica sa po 2^{10} stavki

Veličina tabele stranica

- Loša strana klasične tabele stranica je da je njena veličina proporcionalna veličini virtuelnog adresnog prostora

- Alternativa je invertovana tabela stranica
  - Jedna stavka za svaki okvir
  - Značajno manje stavki od klasične tabele stranica
  - Stavka specificira koja stranica kojeg procesa je trenutno smeštena u okvir

Linearna invertovana tabela stranica



- Linearno se prolazi kroz sve okvire da se nađe okvir u kojem je referencirana stranica
- Sporo!

Heširana invertovana tabela stranica

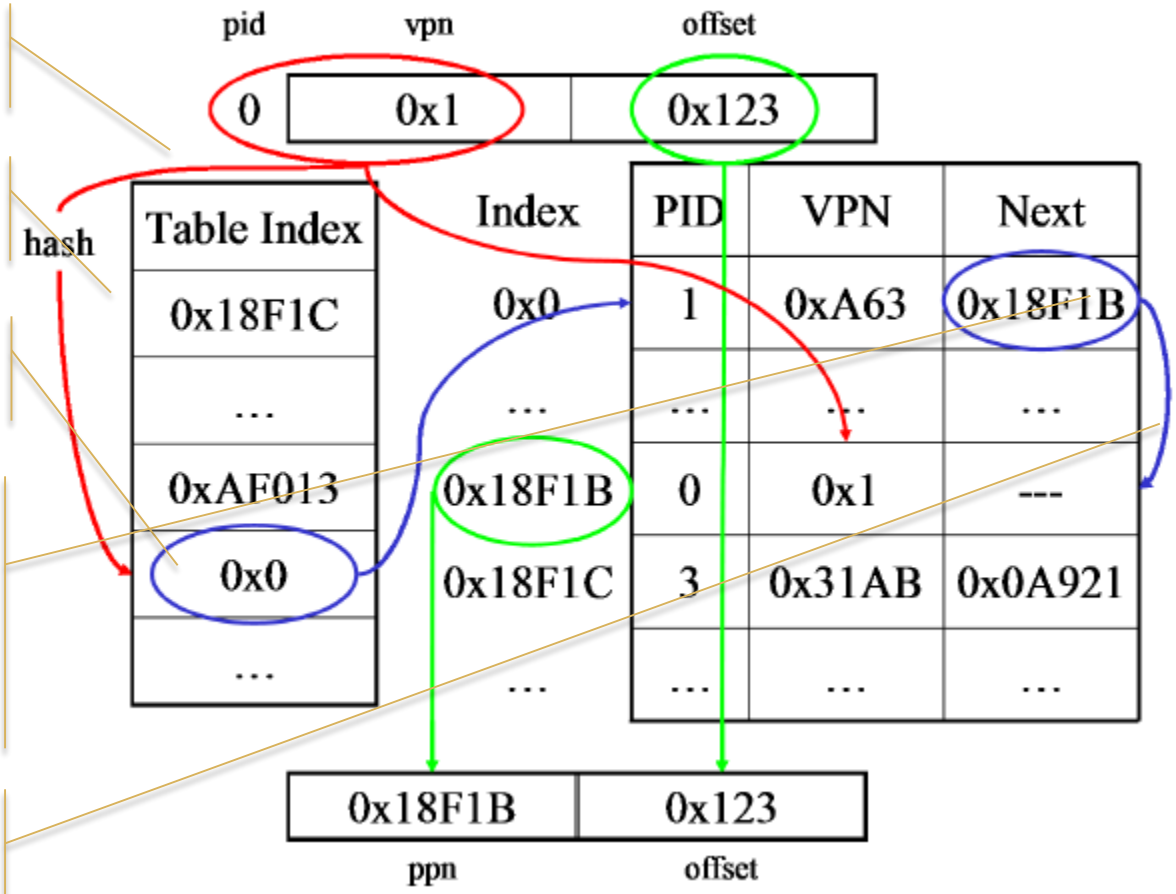
Hash anchor table

Indeks stavke je broj stranice konvertovan hash funkcijom

Sadržaj stavke je okvir u kojem je stranica smeštena

Više stranica može hesh funkcijom da se preslika u istu stavku. Zato se ulančavaju sve stranice koje imaju isti hash

Prolazi se kroz lanac da se nađe stranica koja je referencirana



Zaštita memorije

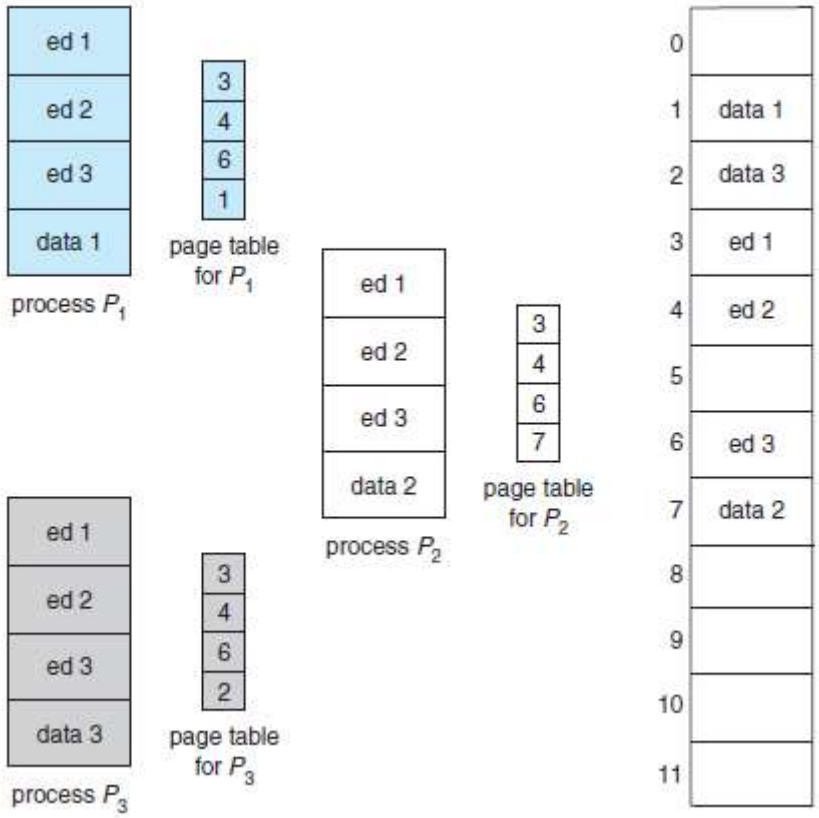
- Da li je određena memorijska lokacija
  - Samo za čitanje
  - Čitanje i upis
  - Samo za izvršavanje
- Implementira se dodatnim bitovima u stavci tabele stranica
- Po jedan bit za svaki od tri tipa zaštite

Zaštita memorije

- Zabrana pristupa stranicama koje ne pripadaju procesu
  - Implicitno ugrađena u logiku rada virtuelne memorije
  - Proces vidi samo svoj virtuelni adresni prostor
  - Proces nema mehanizam da pristupi nečemu što nije deo njegovog adresnog prostora
    - Jer referencira virtuelne adrese, a hardver procesora ih mapira na fizičke adrese u radnoj memoriji

Deljene stranice

- Straničenje omogućuje deljenje zajedničkog koda
- Zajednički kod se smešta samo jednom u glavnu memoriju
- Različiti procesi u svojoj tabeli stranica referenciraju isti okvir sa deljenim kodom
- Primer
  - Tri instance istog editora
  - Samo je sekcija za podatke svakom procesu različita



Veličina stranice za različite familije procesora

Architecture	Page Size	Huge Page Size	Large Page Size
32-bit x86	4 KB	4 MB	
x86-64	4 KB	2 MB	1 GB
IA-64 (Itanium)	4 KB	8 KB, 64 KB, 256 KB, 1 MB, 4 MB, 16 MB, 256 MB	-
Power Architecture	4 KB	64 KB, 16 MB	16 G
SPARC	8 KB	-	64 KB, 4 MB, 256 MB, 2 GB
ARMv7	4 KB	1 MB, 16 MB	-

Aspekti upravljanja memorijom

- Hardverski aspekt
  - odgovornost hardvera procesora
- **Softverski aspekt**
  - odgovornost softvera operativnog sistema

OS podrška za upravljanje memorijom

- Preduslov je da hardver
  - koristi tehnike virtuelne memorije
  - podržava straničenje, segmentaciju ili oboje
- Softver OS je odgovoran za algoritme koji se primenjuju pri upravljanju memorijom
- Najvažnije je pitanje performanse
  - potrebno je minimizovati učestalost grešaka stranica
  - OS donosi odluke kada, kako i koju stranicu da zameni
  - Dok se zamenjuje stranica, OS raspoređuje drugi proces

Politika donošenja

- Odlučuje kada stranica treba da se donese u glavnu memoriju
- Dve osnovne varijante
  - Straničenje po zahtevu
  - Predstraničenje

Politika donošenja

- Straničenje po zahtevu
  - stranica se donosi u glavnu memoriju samo kada se napravi referenca na lokaciju u toj stranici
  - Mnogo grešaka stranica na početku rada programa
  - Kasnije taj broj opada
- Predstraničenje
  - Donose se unapred stranice koje nisu zahtevane
  - Efikasnije je doneti više stranica odjednom, ako se na disku nalaze u susednim lokacijama
  - Može nepotrebno doneti stranicu koja se neće koristiti

Politika smeštanja

- Određuje gde u glavnoj memoriji proces treba da bude smešten
- Ako sistem koristi straničenje (što je slučaj u većini savremenih OS) ovo pitanje je nevažno
  - hardver za prevođenje adresa radi jednako bez obzira u kojem je konkretno okviru stranica smeštena

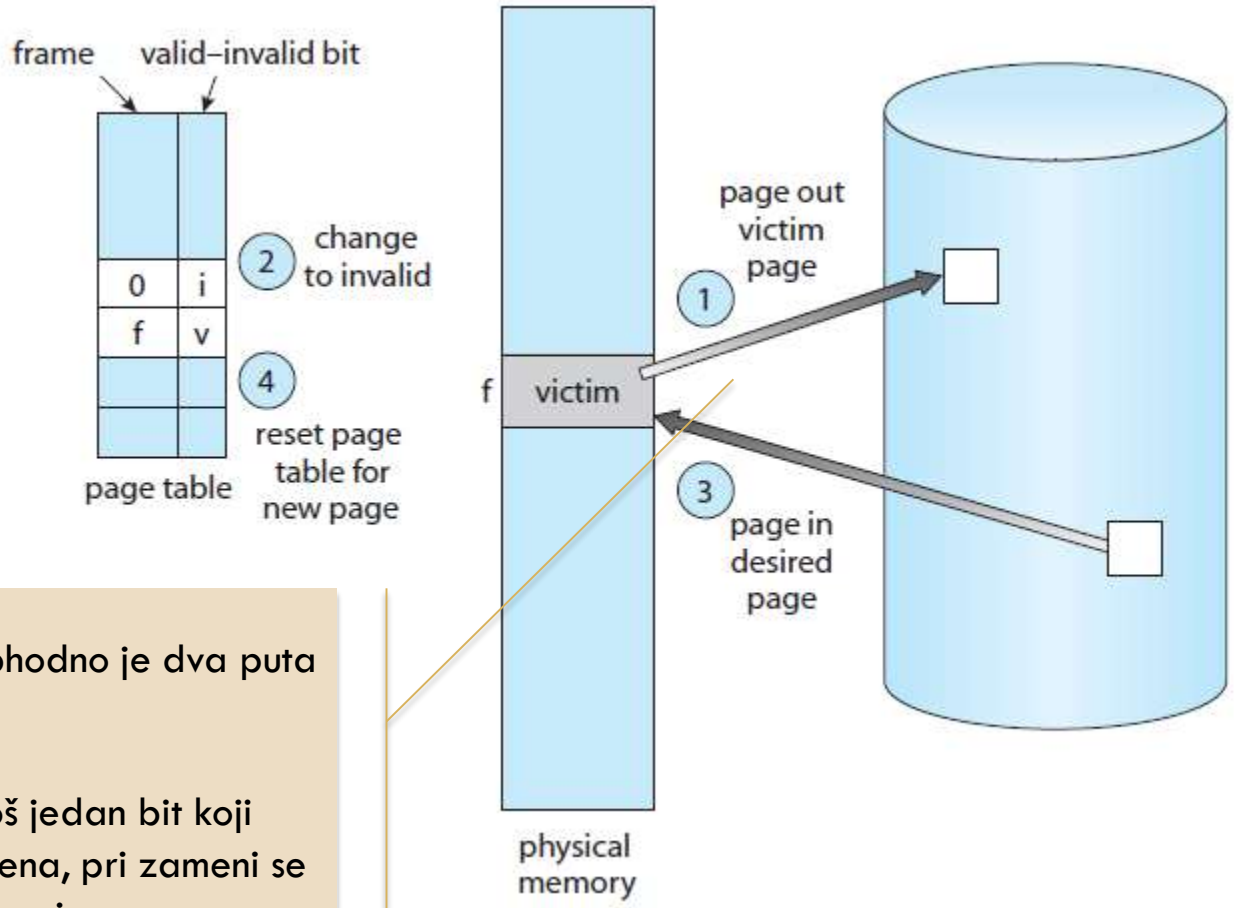
Politika zamene

- Kada su svi okviri u glavnoj memoriji zauzeti potrebno je da se donese nova stranica
- Politika zamene određuje koja stranica iz glavne memorije treba da bude zamenjena

Politika zamene

- Cilj je da se minimizuje broj grešaka stranice tako što se zameni ona za koju je najmanje verovatno da će uskoro biti referencirana
  - Kako da se to utvrdi?
  - Princip lokalnosti može da pomogne
- Većina politika pokušava da predvidi buduće ponašanje na osnovu ponašanja u prošlosti

Zamena stranica



- Ako nema slobodnih okvira, neophodno je dva puta pristupiti disku
- Ako se uvede u tabelu stranica još jedan bit koji označava da li je stranica izmenjena, pri zameni se na disk upisuju samo izmenjene stranice

Politika zamene – zaključavanje okvira

- Ako je okvir zaključan, stranica smeštena u njemu ne može da se zameni
- U zaključanim okvirima se čuvaju
  - kernel OS
  - glavne upravljačke strukture
  - U/I baferi
- Mogu se zaključati i tek donešene stranice da ne bi bile zamenjene pre korišćenja zato što drugi proces treba slobodan okvir
- Zaključavanje se postiže pridruživanjem bita zaključavanja svakom okviru
  - Mora se oprezno koristiti jer se zaključavanjem okvira smanjuje broj raspoloživih okvira za smeštanje stranica

Politike za izbor stranice za zamenu

- Optimalna
- Najmanje skoro korišćena
  - *Last recently used* (LRU)
- Prva unutra, prva napolje
  - *First-in-first-out* (FIFO)
- Časovnik

Niz referenci

- Program pri izvršavanju redom referencira određene lokacije u memoriji
- Politike zamene se porede kroz broj grešaka stranice na određenom nizu referenci
- Kod referenci nas interesuju samo adrese stranica
  - Pomeraj unutar stranice sada nije važan
- Uzastopne reference na istu stranicu se mogu posmatrati kao jedna referenca
- Primer niza referenci
  - 2 3 2 1 5 2 4 5 3 2 5 2

Optimalna politika

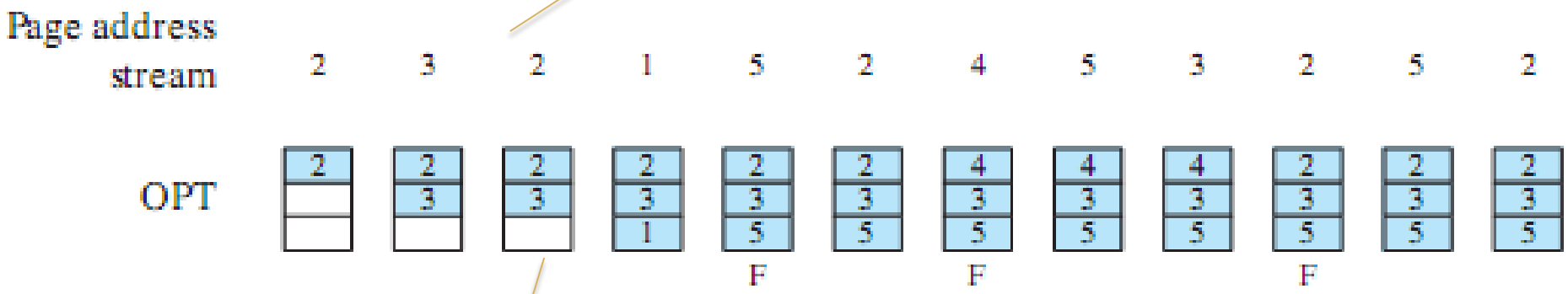
- Bira se ona stranica za koju je vreme do sledeće reference najduže

Optimalna politika

- Ova politika ne može da se implementira jer zahteva od OS savršeno poznavanje budućih događaja
- Služi kao standard u odnosu na koji se procenjuju algoritmi koje je moguće implementirati

Optimalna politika - primer

Adrese koje se referenciraju u toku izvršavanja programa



Procesu su dodeljena tri okvira u radnoj memoriji

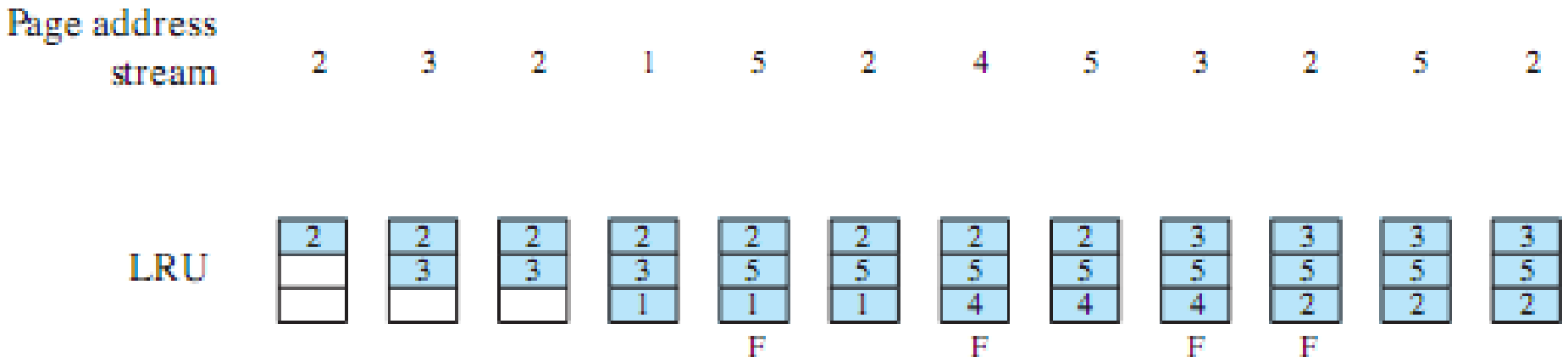
Greška stranice nastupa kada se referencira stranica koja nije u glavnoj memoriji. Tada se tražena stranica ubacuje u glavnu memoriju, a neka od stranica izbacuje

Najmanje skoro korišćena

- Ako sistem ne poznaje buduće događaje, ima evidenciju prošlih
- Zamenjuje stranicu koja najduže nije bila korišćena pod pretpostavkom da ni uskoro neće biti

Najmanje skoro korišćena

- Zamenjuje stranicu koja najduže nije bila referencirana



Najmanje skoro korišćena

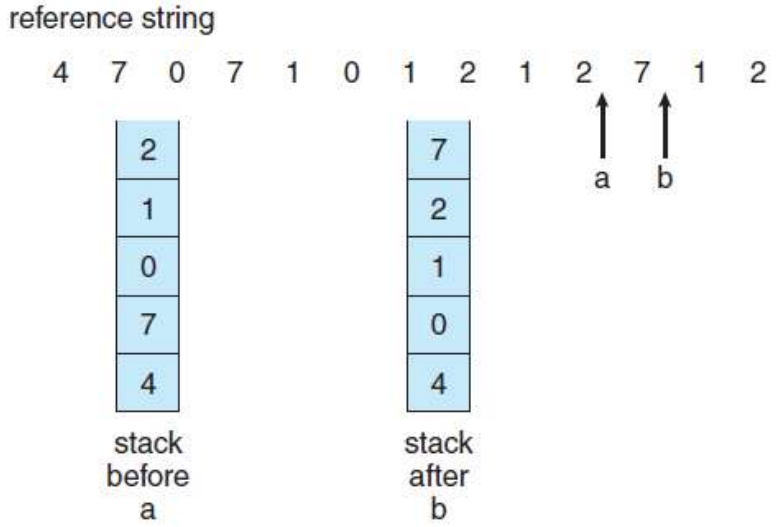
- Po principu lokalnosti, trebalo bi da bude zamenjena stranica za koju je najmanje verovatno da će biti referencirana u bliskoj budućnosti
- Radi skoro jednako dobro kao optimalna politika
- Teško za implementaciju
- Dve varijante implementacije
  1. Brojač
  2. Stek

Najmanje skoro korišćena

- Implementacija preko brojača
  - Procesor evidentira broj otkucaja
  - U tabeli stranica se uz svaku stranicu doda polje koje označava trenutak poslednjeg referenciranja
  - Pri svakom referenciranju se upiše trenutak referenciranja
  - Zamenjuje se stranica sa najmanjim trenutkom vremena

Najmanje skoro korišćena

- Implementacija preko steka
  - Čuva se stek referenciranih stranica
  - Kada se stranica referencira stavlja se na vrh steka
  - Ako je već bila referencirana izbacuje se iz nekog dela steka
  - Ovaj stek se može implementirati kao dvostruko spregnuta lista



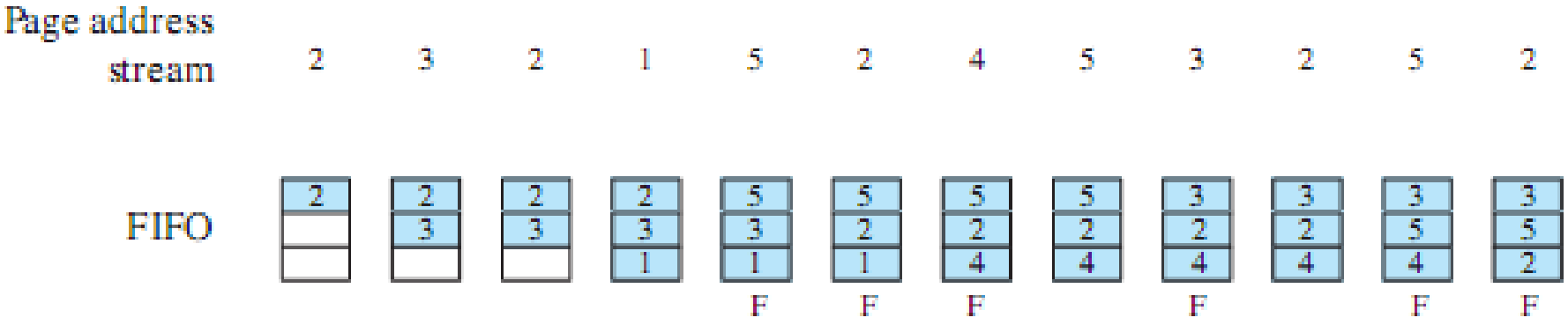
Najmanje skoro korišćena

- Ne može se koristiti u realnim sistemima
- Bilo koja implementacija ove politike veoma usporava sistem
- Ažuriranje trenutaka referenciranja ili steka mora biti obavljeno pri svakom referenciranju
  - To uvodi usporenje od najmanje 10 puta

Prva unutra, prva napolje

- Tretira okvire kao kružni bafer
- Postoji pokazivač na narednu stranicu koju treba ukloniti
  - Pokazivač kruži redom kroz okvire
  - Najjednostavnija politika za implementaciju
- Biće zamenjena stranica koja je najduže u memoriji
  - Često pogrešno
  - Stranice koje se često referenciraju će stalno biti izbacivane i ponovo ubacivane

Prva unutra, prva napolje - primer



- Više grešaka od LRU
- Za razliku od LRU ne prepoznaje stranice koje su češće referencirane od ostalih

Aproksimacije politike LRU

- Postoje politike koje pokušavaju da dobiju rezultate kao LRU, a da pri tome izbegnu usporenje koje klasična implementacija ove politike donosi
- Baziraju se na tome da hardver za svaku stranicu ažurira bit upotrebe
  - Kada se stranica donese u memoriju ili referencira, bit se postavlja na jedan
  - OS može da resetuje bit upotrebe za određenu stranicu
  - Dobijamo informaciju koje su stranice referencirane, ali ne i u kojem redosledu

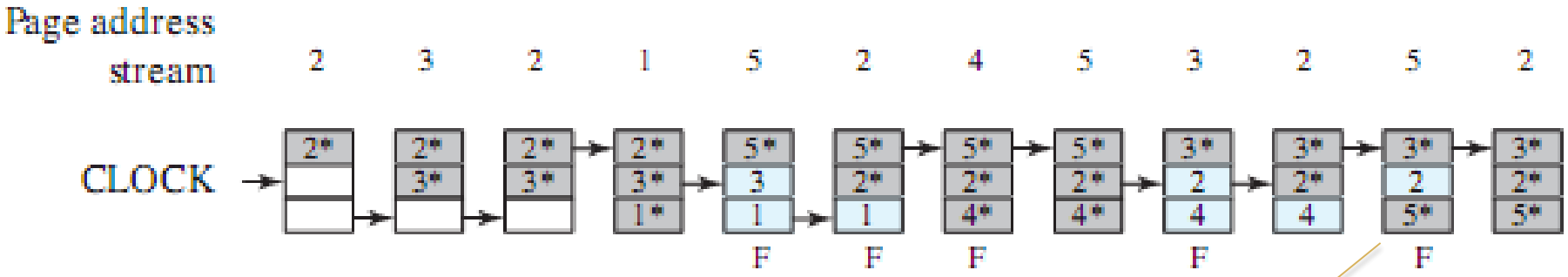
Politika dodatnih bitova upotrebe

- OS evidentira 8-bitnu reč za svaku stranicu
- U regularnim intervalima OS
  - Prebacuje bit upotrebe na najvišu poziciju u reči
  - Trenutne bite pomera na desno
  - Bit na najmanjoj poziciji ispada iz reči
- 8-bitna reč time sadrži istoriju referenciranja
- Ako 8-bitnu reč posmatramo kao dekadni broj
  - Stranica sa većom vrednošću reči je skorije referencirana
  - 11000100 je skorije referencirana od 01110111
  - Stranica sa najmanjom vrednošću će biti zamenjena

Politika časovnika

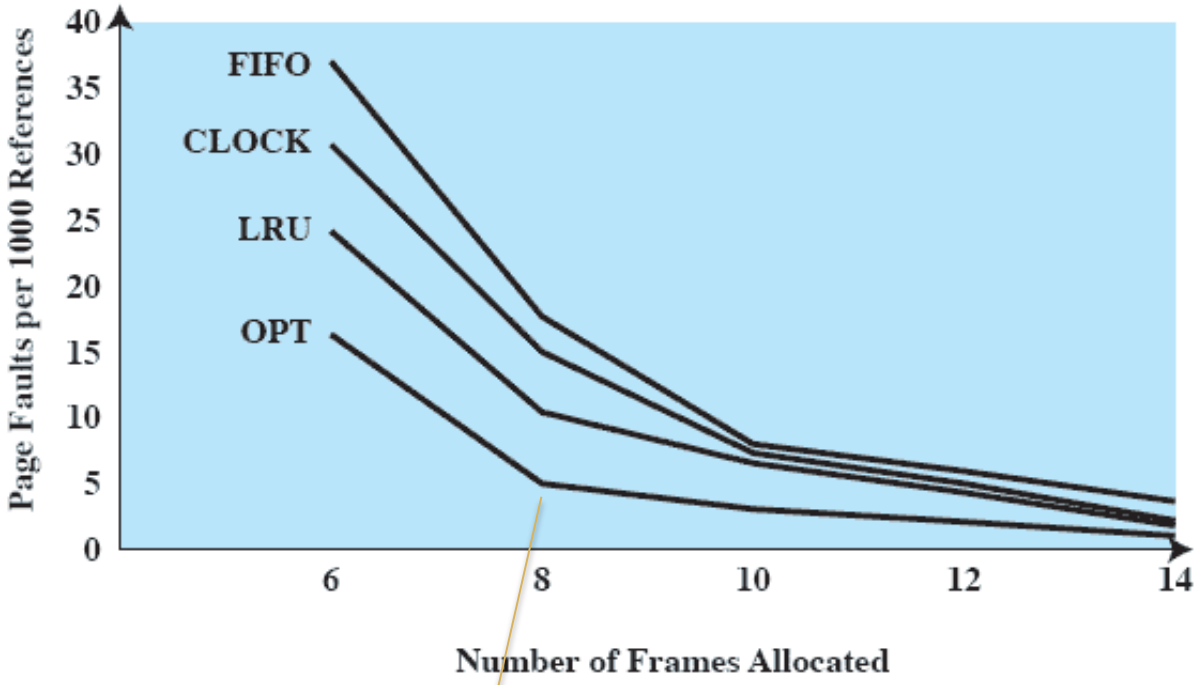
- Samo jedan bit upotrebe za svaku stranicu
- Kada je potrebno pronaći stranicu za zamenu
  - OS prolazi kružno kroz okvire
  - Kad god naiđe na okvir sa bitom upotrebe 1, postavi bit na 0
- Biće zamenjena stranica za koju je bit upotrebe 0
- Pošto ide u krug, u najgorem slučaju će u drugom prolasku pronaći okvir čiji je bit u prvom prolasku postavljen na 0
- Naredno pretraživanje će krenuti od okvira koji se nalazi posle okvira koji je zamenjen

Politika časovnika - primer



- FIFO bi zamenio 2, umesto 4
- Pošto je 2 skoro referencirana, algoritam časovnika je ne zamenjuje

Poređenje algoritama



Zbog lokalnosti, nakon određene granice, dalje povećavanje broja okvira dodeljenih procesu ima manji uticaj

Baferovanje stranica

- Deo glavne memorije se odvoji da radi kao keš stranica
- Zamenjena stranica se ubacuje u taj keš
  - Ako stranica nije menjana dodaje se u listu slobodnih stranica
  - Ako je stranica menjana dodaje se u listu menjanih stranica
- Kada se ponovo referencira stranica
  - postoji šansa da se stranica nalazi u baferu, pa je vraćanje u upotrebu brzo i jednostavno
  - Promenjene stranice se mogu upisivati na disk u grupama, što je brže

Upravljanje rezidentnim skupom

- OS za svaki proces treba da odluči koliko okvira da mu dodeli za smeštanje stranica
  - Što je manje memorije dodeljeno jednom procesu, to više procesa može istovremeno da bude u memoriji
  - Što je manji broj okvira dodeljen procesu, biće više grešaka stranice
  - Zbog principa lokalnosti, povećavanje broja dodeljenih okvira preko određene granice neće imati zapažen efekat na broj grešaka stranice

Minimalna veličina rezidentnog skupa

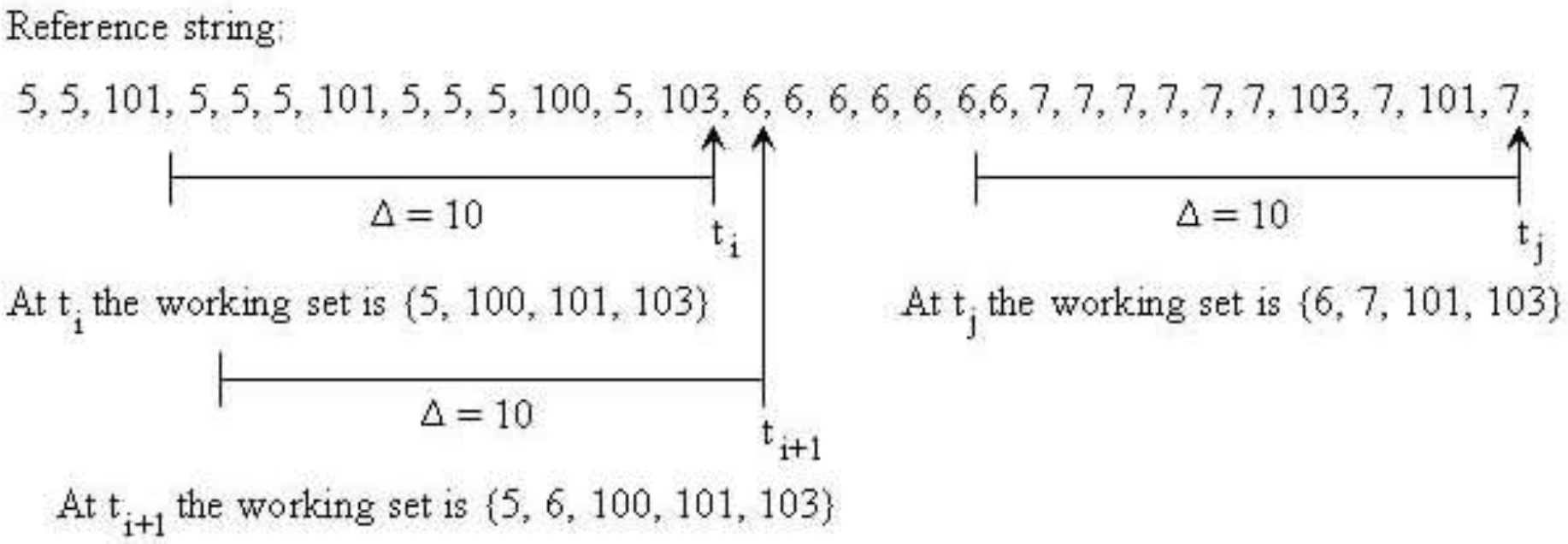
- Kada se desi greška stranice instrukcija se restartuje
  - Neophodno je da se procesu dodeli bar onoliko okvira koliko svaka pojedinačna instrukcija može da referencira
  - Arhitektura skupa instrukcija definiše minimalan broj okvira za proces
- Drugi parametar za određivanje minimalne veličine je da ne bude previše grešaka stranica

Radni skup

- Koji je kriterijum za određivanje odgovarajuće veličine rezidentnog skupa?
- Često korišćena je **strategija radnog skupa**
  - Zasniva se na principu lokalnosti
- Radni skup $W(t, \Delta)$
  - u trenutku vremena t, skup stranica koje su bile referencirane u poslednjih Δ jedinica vremena
  - vreme se meri referenciranjem stranica
  - jedno referenciranje je jedna jedinica vremena

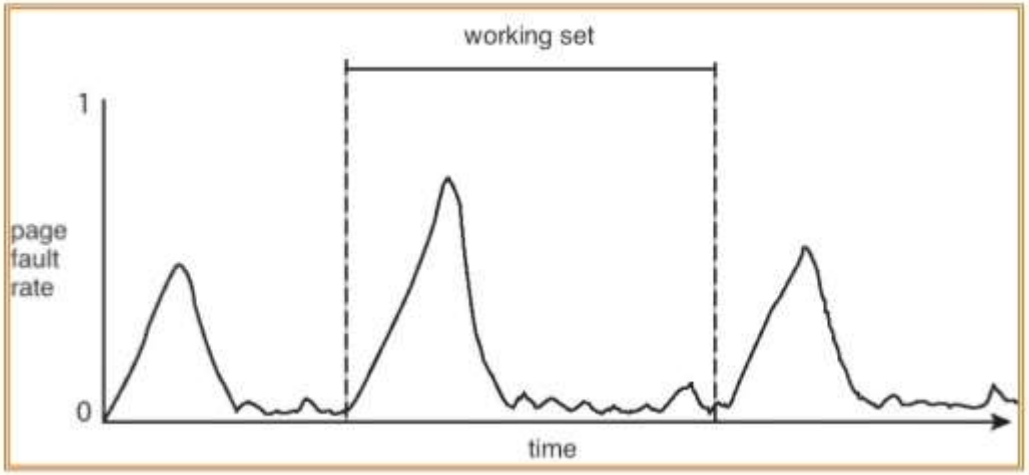
Radni skup - primer

- Pretpostavimo da je $\Delta = 10$ (nerealno malo)
- Izgled radnog skupa u trenucima t_i, t_{i+1} i t_j



Promena veličine radnog skupa

- Radni skup procesa
  - zbog principa lokalnosti ima relativno stabilne periode i
  - periode kada se drastično menja zbog prelaska na novu lokalnost



Strategija radnog skupa

- Nadgledati radni skup svakog procesa
- Periodično uklanjati iz memorije one stranice koje nisu u radnom skupu
- Problem
  - Merenje radnog skupa je nepraktično i zahtevno sa stanovišta performansi

Algoritam učestalosti greške stranice

- PFF – *Page fault frequency*
- Umesto nadgledanja radnog skupa
  - Nadgledati učestalost grešaka stranica
  - Učestalost grešaka stranice opada povećanjem rezidentnog skupa procesa
  - Ako je učestalost manja od nekog praga, smanjujemo rezidentni skup procesu
  - Ako je učestalost iznad nekog praga, povećavamo rezidentni skup

Algoritam učestalosti greške stranice

- Svakoj stranici se pridružuje bit upotrebe
- Brojač referenci koji broji koliko je proteklo referenci od poslednje greške stranice
- Kada se desi greška stranice, poredi se brojač sa pragom *F*
  - Ako je vreme od poslednje greške stranice manje od F, rezidentni skup se proširuje novom stranicom
  - Ako je vreme veće od F, odbacuju se stranice sa bitom upotrebe 0 i tako smanjuje rezidentni skup
  - Pri svakoj greški stranice, resetuje se na 0 bit upotrebe svim stranicama u rezidentnom skupu

Algoritam učestalosti greške stranice

- Ne radi dobro u toku prelaznih perioda kada se prelazi na novu lokalnost
  - Pri prelasku u novu lokalnost veliki broj grešaka stranica će povećavati rezidentni skup ne izbacujući odmah stranice stare lokalnosti
  - Kasnije će se rezidentni skup opet smanjivati, jer se stranica stare lokalnosti neće referencirati
  - Da bi se stranica izbacila iz radnog skupa potrebno je da prođe F jedinica virtuelnog vremena od kad je poslednji put referencirana

Radni skup sa promenljivim intervalom uzorkovanja

- VSWS – *Variable-interval sampled working set*
- Definiše se interval uzorkovanja
  - Na početku intervala, svim stranicama u rezidentnom skupu se resetuje bit upotrebe
  - U toku intervala svaka stranica koja prouzrokuje grešku stranice se dodaje u rezidentni skup
    - U toku intervala rezidentni skup može samo da raste
  - Na kraju intervala se izbacuju stranice koje imaju bit upotrebe 0
    - Na kraju intervala rezidentni skup može samo da se smanji

Radni skup sa promenljivim intervalom uzorkovanja

- U toku izvršavanja procesa meri se virtuelno vreme
- Ako je prošao maksimalni period od poslednjeg uzorkovanja
  - suspenduje se proces i skeniraju bitovi upotrebe
- Ako u toku intervala broj grešaka stranice pređe određeni prag
  - Ako je prošao minimalni period uzorkovanja, ponovo se vrši uzorkovanje
  - Ako nije prošao, čeka se da istekne minimalni period i tada se vrši uzorkovanje

Radni skup sa promenljivim intervalom uzorkovanja

- VSWS algoritam se bolje ponaša u prelazima između lokalnosti
- Učestalost uzorkovanja zavisi od učestalosti grešaka stranica
- Stranice stare lokalnosti će brže ispadati iz rezidentnog skupa nego kod PFF algoritma

Politika čišćenja

- Odlučuje kada se izmenjena stranica upisuje nazad u sekundarnu memoriju
- Čišćenje po zahtevu
  - Stranica se upisuje kada je izabrana za zamenu
- Predčišćenje
  - Stranice se upisuju unapred u paketima, pre nego što je traženo da se zamene
  - Ako se opet stranica izmeni, onda je prethodno upisivanje na disk bilo bespotrebno trošenje resursa

Politika čišćenja

- Najbolji pristup je baferovanje stranica
- Zamenjene stranice su u dve liste
  - Izmenjene i neizmenjene
- Stranice iz liste izmenjenih se periodično upisuju na disk
- Stranice iz liste neizmenjenih
  - vraćaju se u okvire dodeljene procesu ako se referenciraju ili
  - izbacuju se iz glavne memorije ako nema više mesta za njih u listi

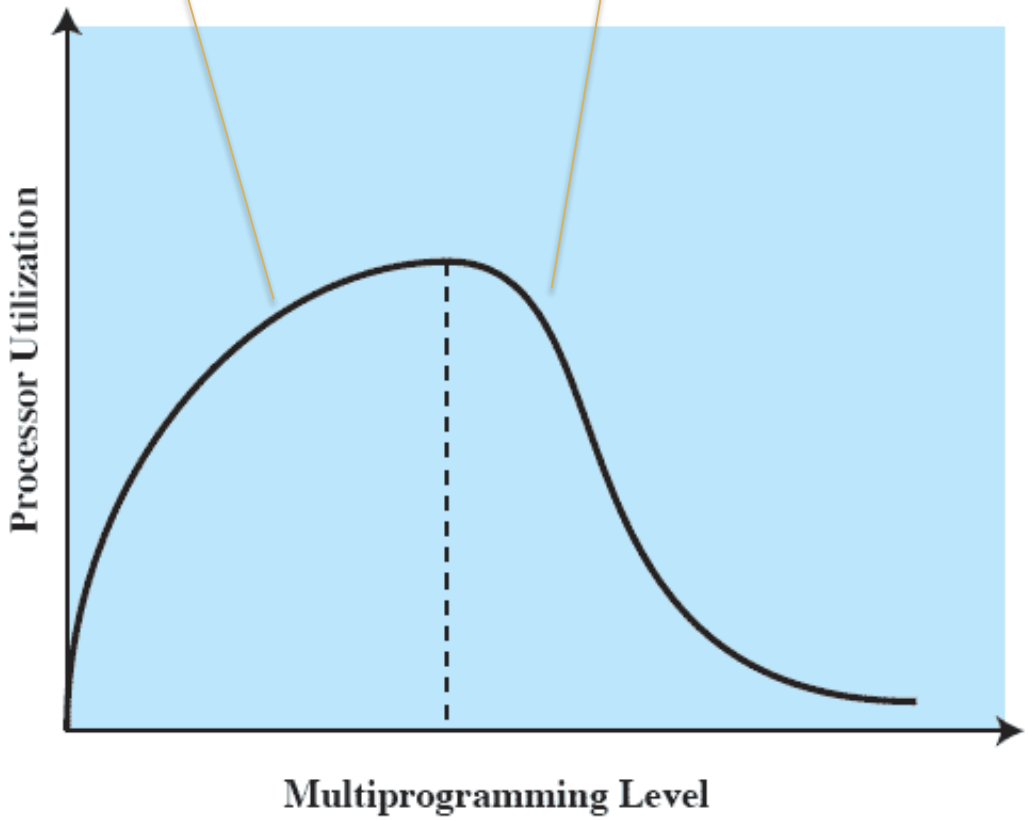
Upravljanje učitavanjem

- Bavi se nivoom multiprogramiranja
  - Koliko procesa će biti u glavnoj memoriji?
- Premalo procesa
  - često će svi procesi biti blokirani, pa će procesor biti besposlen
- Previše procesa
  - Veličina rezidentnog skupa svakog procesa je mala
  - Puno grešaka stranice
  - „Brbljanje“ da bi se dobavile stranice sa diska i upisivale nazad na disk

Nivo multiprogramiranja

Što više procesa u memoriji, procesor je iskorišćeniji

Nakon određenog broja procesa, sve više je „brbljanja“ i iskorišćenje procesora pada



Suspenzija procesa

- Ako je previše procesa u memoriji
  - Neki od procesa mora biti suspendovan
- Različite varijante izbora procesa koji će biti suspendovan

Izbor procesa za suspenziju

- Proces najnižeg prioriteta
- Proces koji je izazvao grešku
  - Proces očigledno nema radni skup u memoriji
  - Svakako bi bio blokiran čekajući zamenu stranice
- Proces koji najduže nije bio aktivan
  - Za ovaj proces je najverovatnije da neće imati svoj radni skup u memoriji

Izbor procesa za suspenziju

- Proces sa najmanjim rezidentnim skupom
  - Za ovaj proces je najmanji napor da se ponovo učita
- Najveći proces
  - Dobija se najviše slobodnih okvira
- Proces za koji je najmanje hitno da bude završen