Upravljanje memorijom

Vladimir Filipović

Upravljanje memorijom

- Memorija predstavlja jedan od osnovnih resursa računarskog sistema.
- Svaka procesorska instrukcija se mora naći u memoriji, a za izvršavanje većine instrukcija potrebni su podaci iz memorije koji se moraju doneti (učitati) u procesor ili iz procesora prebaciti u memoriju.
- Iako današnji prosečni računari imaju nekoliko hiljada puta više memorije od najmoćnijih računara iz šezdesetih godina dvadesetog veka, programi postaju sve zahtevniji, a podaci sa kojima operišu postaju sve veći.

Upravljanje memorijom (2)

- Kako je memorija ograničeni resurs, njeno efikasno korišćenje je jedan od osnovnih zadataka operativnih sistema. Ovim problemom se obično bavi poseban modul operativnog sistema koji se naziva menadžer memorije.
- Memorija je bilo koji fizički uređaj koji može privremeno ili trajno da čuva podatke.
- Na osnovu načina čuvanja podataka dele se na:
  - memorije koje privremeno čuvaju podatke (dok je računar uključen) i
  - memorije koje imaju mogućnost trajnog čuvanja podataka.

Vrste memorije

- Osnovna podela memorije obično je na osnovu brzine pristupa:
    1. Registri;
    2. Keš memorija;
    3. Primarna memorija;
    4. Sekundarna memorija.

1. Najbrži su registri - memorijske ćelije ugrađene u procesor i u njima se nalaze podaci koje procesor trenutno obrađuje. Primer: ako je potrebno izvršiti aritmetičku operaciju nad dva broja, ti brojevi se najpre smeštaju u registre. Registri su memorija koja privremeno čuva podatke.

Vrste memorije (2)

2. Keš memorija je veoma brza memorija koja takođe čuva podatke privremeno i obično se nalazi u samom procesoru.
   Ona predstavlja bafer između primarne memorije i procesora.
   Keš memorija sadrži delove podataka za koje se pretpostavlja da će ih procesor često koristiti.
   Na ovaj način se smanjuje broj pristupa sporijim memorijama, čime se povećava iskorišćenost procesora i poboljšava efikasnost sistema.

Vrste memorije (3)

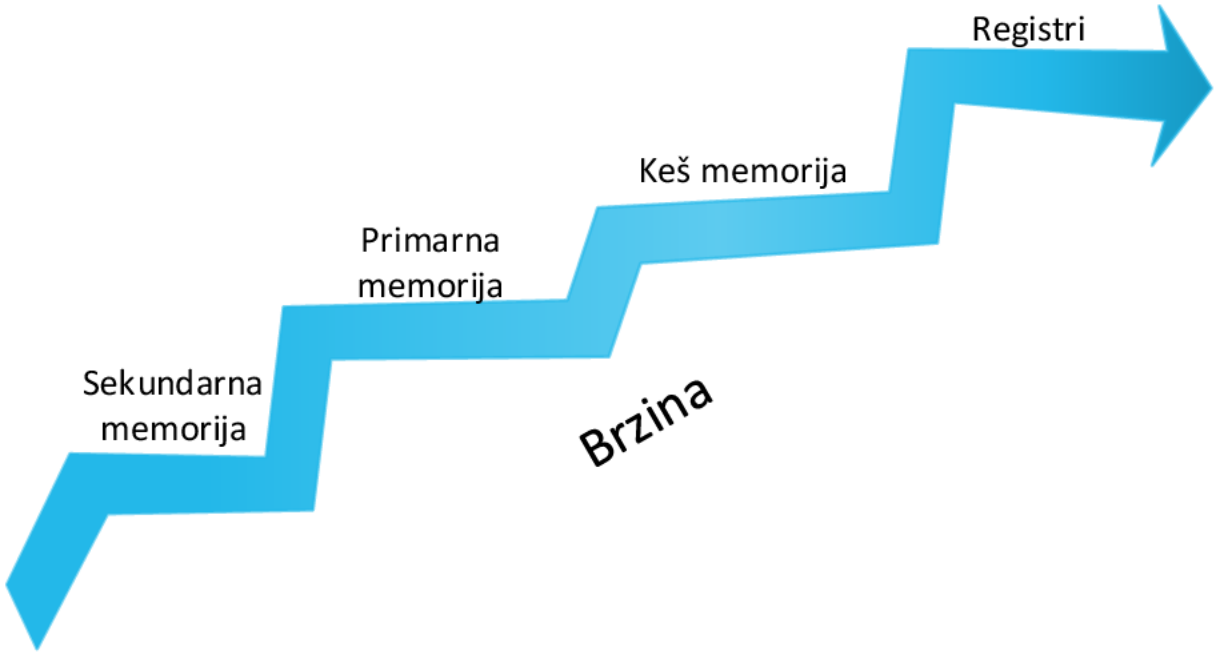
3. Primarna (unutrašnja) memorija sadrži instrukcije i podatke sa kojima procesor trenutno operiše, pa se još naziva radnom memorijom.
   Ova memorija je sporija od keš memorije, ali je značajno brža od sekundarne memorije.
   Vrsta primarne memorije je RAM memorija. RAM memorija privremeno čuva podatke i bez nje računar ne može da funkcioniše.
   ROM memorija je vrsta primarne memorije koja čuva podatke kada je računar isključen. Moguće je samo čitanje podataka iz ROM-a.

Vrste memorije (4)

4. Sekundarna memorija ne gubi sadržaj nakon prestanka rada računara, pa se zbog toga koristi za trajno čuvanje podataka.
   Da bi procesor pristupio podacima iz sekundarne memorije, najpre je potrebno da se oni učitaju u radnu memoriju, pa se sekundarna memorija naziva i spoljašnja memorija.
   Primeri sekundarne memorije su hard disk, CD-ROM, DVD, itd.
   Vreme pristupa podacima iz ove memorije zavisi od lokacije podataka, kao i od vrste sekundarne memorije.

Vrste memorije (5)

Sledeća slika pokazuje vrste memorije i njihove brzine.



Brzina memorije

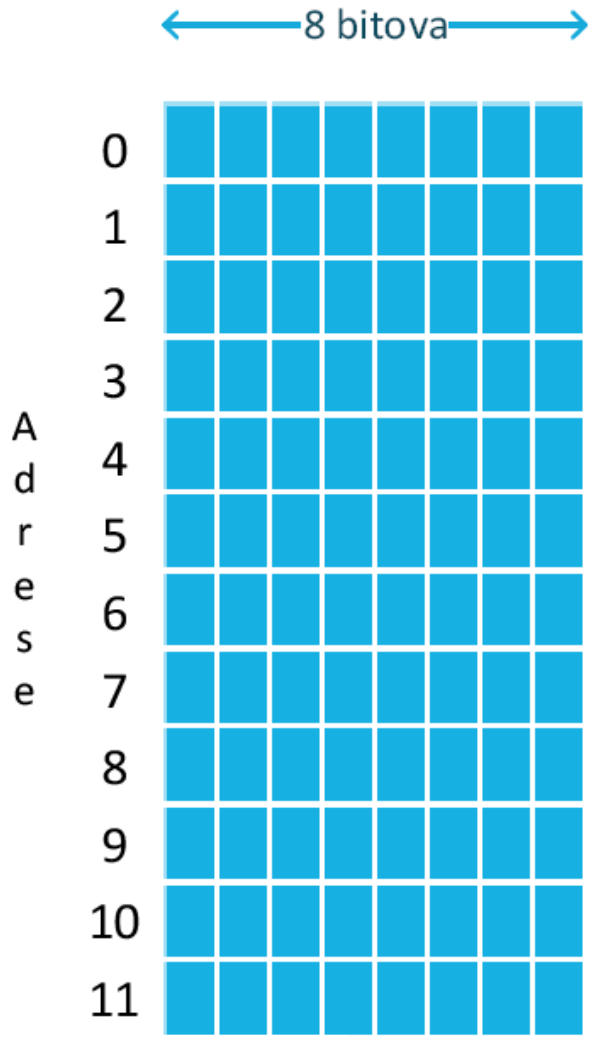
Organizacija memorije

- Memorija se sastoji od niza memorijskih ćelija koje mogu da sačuvaju najmanju količinu podataka – bit.
- Bitovi se organizuju u bajtove – grupe od po 8 bitova.
- Na većini računara bajt je najmanja adresibilna jedinica, što znači da svaki bajt ima svoju adresu i može mu se direktno pristupiti.
- Memorijska reč određuje količinu podataka koje procesor može da obradi u jednom trenutku. Njena veličina je određena arhitekturom konkretnog računara, ranije obično 8, 16, 32 bita.

Organizacija memorije (2)

- Na primer, 64-bitni računar je u mogućnosti da jednom instrukcijom sabere dva 64-bitna broja (brojevi predstavljeni sa 64 binarnih cifara), dok će 32-bitni računar biti u stanju da isto to uradi sa brojevima veličine najviše 32 bita.
- Često se, kada se govori o veličini memorijske reči, misli i na adresibilni prostor računara.
- Na primer, 32-bitni računar najčešće dozvoljava 32-bitne memorijske adrese, pa je u mogućnosti da podrži 2^{32}= 4.294.967.296 bajtova (4GB) memorije.

Organizacija memorije (3)



Memorija kao matrica

- Radna memorija se može predstaviti kao matrica bitova. Zbog pojednostavljenja može se pretpostaviti da je ta matrica dimenzija $N \times 8$, pa se memorija može shvatiti kao niz bajtova.
- Redni broj bajta zapisan u binarnom sistemu predstavlja adresu tog bajta. Na osnovu te adrese procesor može da pristupi proizvoljnom bajtu.

Povezivanje adresa

- Svaki program koji se izvršava mora se učitati u radnu memoriju.
- U trenutku pisanja programa nije poznato u kom delu memorije će program biti smešten. Zato se koriste simboličke adrese, a to su imena promenljivih. Kada se program učita u memoriju on mora baratati sa realnim fizičkim adresama.
- Proces prevođenja simboličkih u fizičke adrese se naziva povezivanje adresa (address binding) i može se obaviti u različitim trenucima, zavisno do toga da li je u vreme prevođenja poznato u kom delu memorije će se učitati taj program.

Povezivanje adresa (2)

- Ukoliko je za vreme prevođenja programa poznato u kom delu memorije će se učitati, onda se povezivanje adresa može obaviti u tom trenutku.
- U suprotnom (najčešćem) slučaju, prevodilac generiše relativne adrese u odnosu na početak dela memorije koja se dodeljuje procesu. Ove adrese se još nazivaju i relokatibilne adrese.

Povezivanje adresa (3)

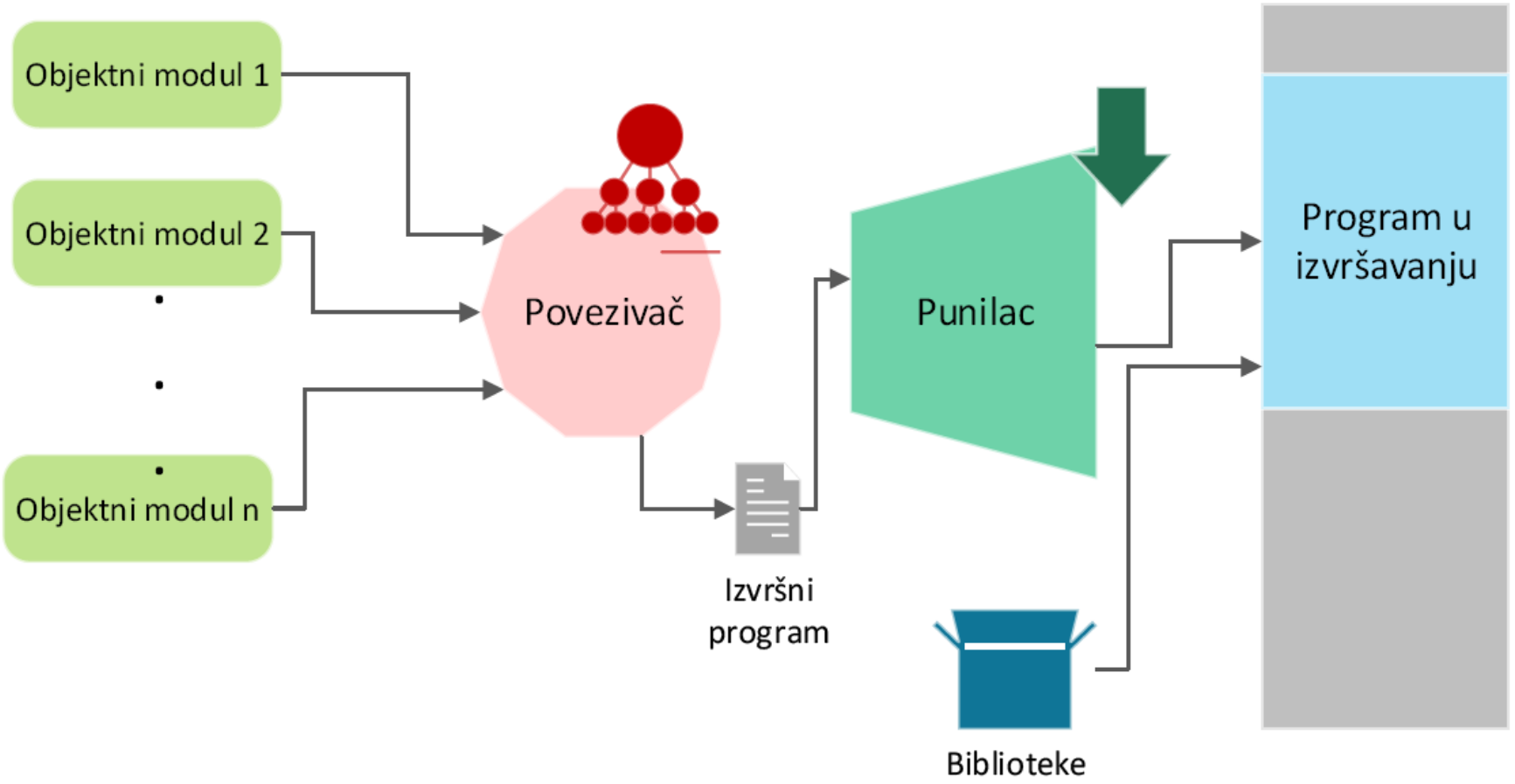
- Prevodilac prevodi izvorni kôd u objektni modul, a program se može sastojati iz više objektnih modula.
- Povezivač (linker) povezuje sve objektne module u izvršni program.
- Prilikom učitavanja izvršnog programa, punilac (loader) preslikava relativne adrese u fizičke.
- Ako se program koji se izvršava može premeštati za vreme izvršavanja, tada se povezivanje adresa obavlja za vreme izvršavanja programa.

Povezivanje adresa (4)

- Program može pozivati neke biblioteke pod određenim uslovima.
- Te biblioteke se ne učitavaju sa učitavanjem programa da ne bi nepotrebno zauzimale prostor.
- Kada se ostvare određeni uslovi, biblioteke se učitavaju u memoriju i pozivaju.
- Dakle, u slučaju ovakvog korišćenja biblioteka se povezivanje adresa obavlja za vreme izvršavanja programa.

Povezivanje adresa (5)

Slika pokazuje izvršavanje programa u memoriji.



Izvršni program u memoriji

Povezivanje adresa (6)

- Pri izvršavanju procesa, interakcija sa memorijom se odvija kroz niz čitanja ili pisanja u lokacije koje se nalaze na određenoj adresi.
- Procesor uzima instrukcije i podatke iz memorije ili smešta podatke u memoriju.
- U tim akcijama procesor ne manipuliše fizičkim adresama, već logičkim koje sam generiše.
- Skup svih logičkih adresa naziva se logički ili virtuelni adresni prostor.
- Svakoj logičkoj adresi odgovara fizička adresa.

Povezivanje adresa (7)

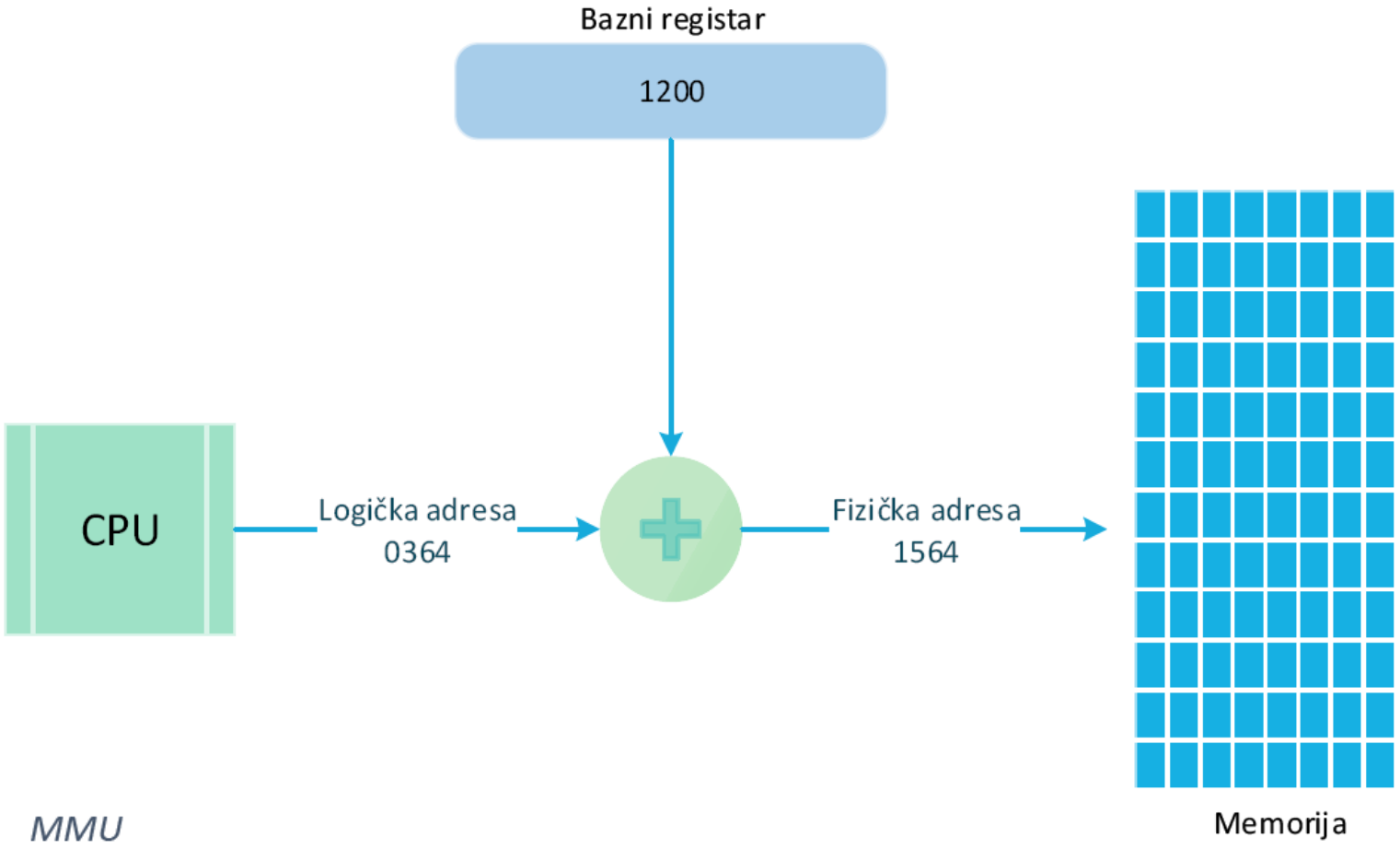
- Skup svih fizičkih adresa koje odgovaraju adresama logičkog adresnog prostora naziva se fizički adresni prostor.
- Ukoliko se povezivanje adresa obavlja za vreme prevođenja i punjenja programa, logički i fizički adresni prostori se podudaraju. To nije slučaj kada se povezivanje adresa obavlja za vreme izvršavanja.

Jedinica za upravljanje memorijom

- Hardverski uređaj koji preslikava logičke u fizičke adrese naziva se jedinica za upravljanje memorijom (Memory-Management Unit - MMU).
- Jednostavni MMU se može predstaviti kao uređaj koji poseduje poseban registar koji se naziva registar relokacije:
  - MMU dodaje vrednost registra relokacije na logičku adresu i tako generiše fizičku adresu.
  - Korisnički program barata logičkim adresama, koje on vidi u opsegu od 0 do max.
  - Ako je sa R označena vrednost registra relokacije, MMU preslikava logičke adrese u fizičke koje su u opsegu od R + 0 do R + max.

Jedinica za upravljanje memorijom (2)

Slika pokazuje strukturu jednostavnog MMU.



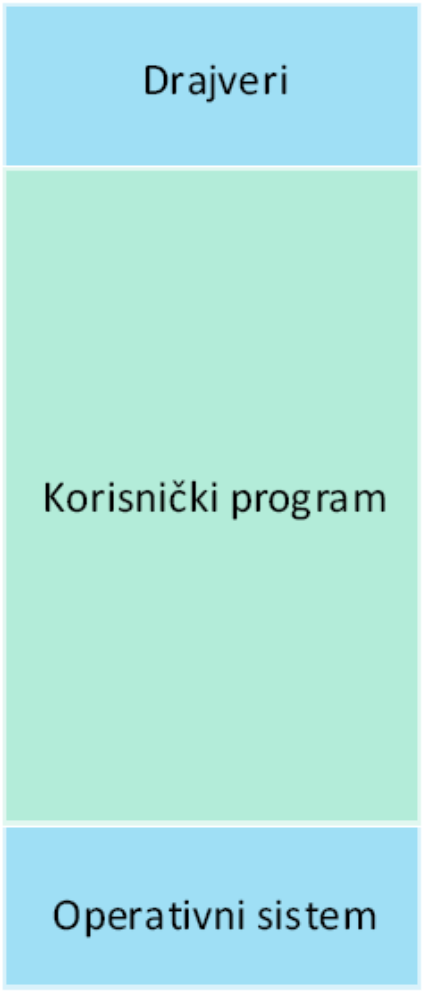
Jedinica za upravljanje memorijom (3)

- Prilikom upravljanja memorijom, operativni sistem bi trebalo da ispuni sledeća dva uslova:
  1. Svaki proces mora imati dovoljno memorije za izvršavanje i ne sme pristupati memoriji drugog procesa kao što drugi proces ne sme pristupati njegovoj memoriji.
  2. Različite vrste memorija unutar računarskog sistema moraju biti korišćene tako da se svaki proces izvršava najefikasnije moguće.

MMU kod monoprogramiranja

- Najjednostavniji pristup upravljanja memorijom podrazumeva da se u memoriji može naći samo jedan proces.
- Tada se jedan deo memorije odvaja za operativni sistem, dok ostatak memorije koristi proces koji se izvršava.
- U tom slučaju se obično deo operativnog sistema učitava na niže adrese memorije, drajveri se učitavaju na adrese koje se nalaze na vrhu memorije, a ostatak služi za korisnički program.
- Ovakvu organizaciju rada sa memorijom ima i operativni sistem MS DOS.

MMU kod monoprogramiranja (2)



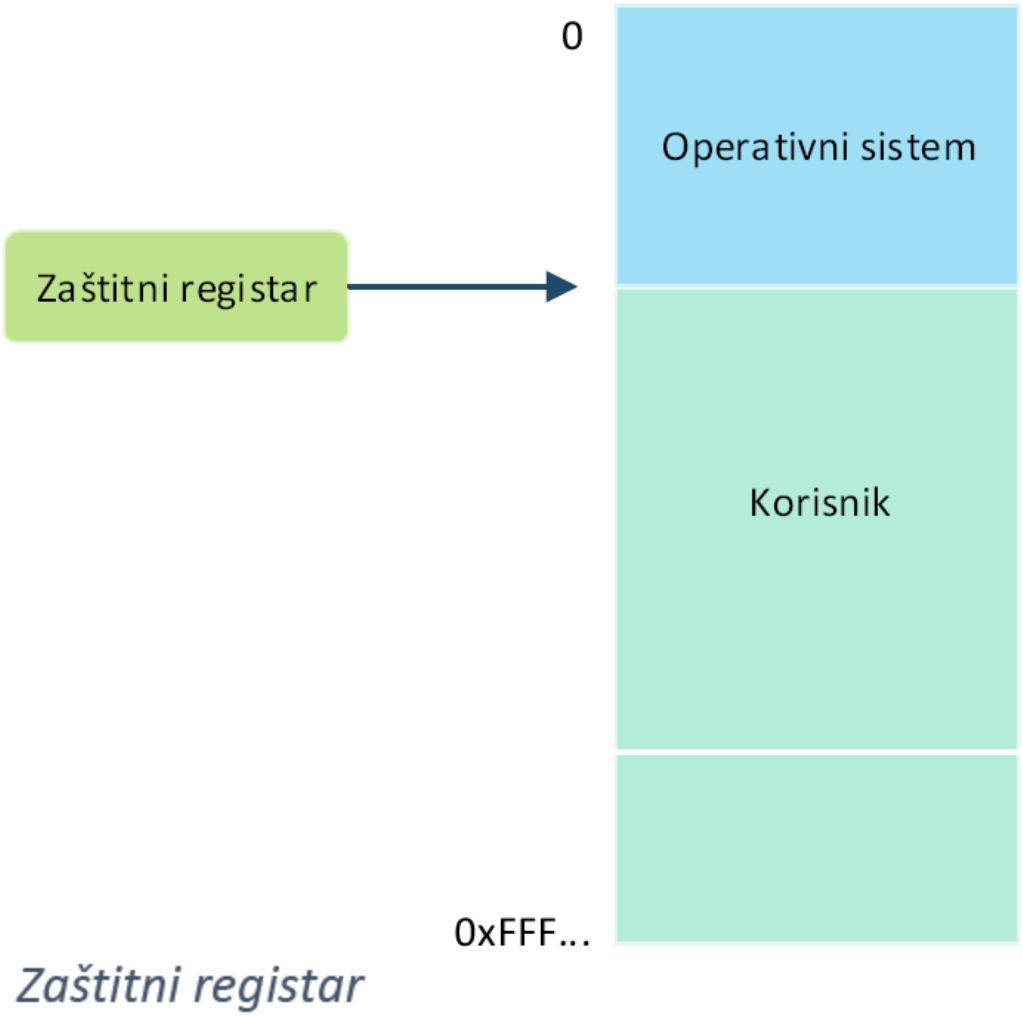
Monoprogramiranje

- Kod MS-DOS sistema:
  1. korisnik unosi komandu za pokretanje programa.
  2. Operativni sistem kopira zahtevani program u memoriju i izvršava ga.
  3. Kada proces završi sa radom, operativni sistem štampa "prompt" karakter i čeka unos sledeće komande.
  4. Kada se unese sledeća komanda, operativni sistem učitava novi program pri čemu prepisuje prethodni sadržaj memorije.

MMU kod monoprogramiranja (3)

- Učitavanje operativnog sistema u memoriju izvršava se prilikom uključivanja sistema.
- Tada se aktivira punilac (bootstrap loader) koji ima zadatak da prenese operativni sistem iz sekundarne memorije u radnu memoriju.
- Ovaj postupak se zove podizanje sistema. Po učitavanju operativnog sistema, korisnički program (odnosno proces) se puni na adrese odmah posle operativnog sistema, ili na drugi kraj memorije.

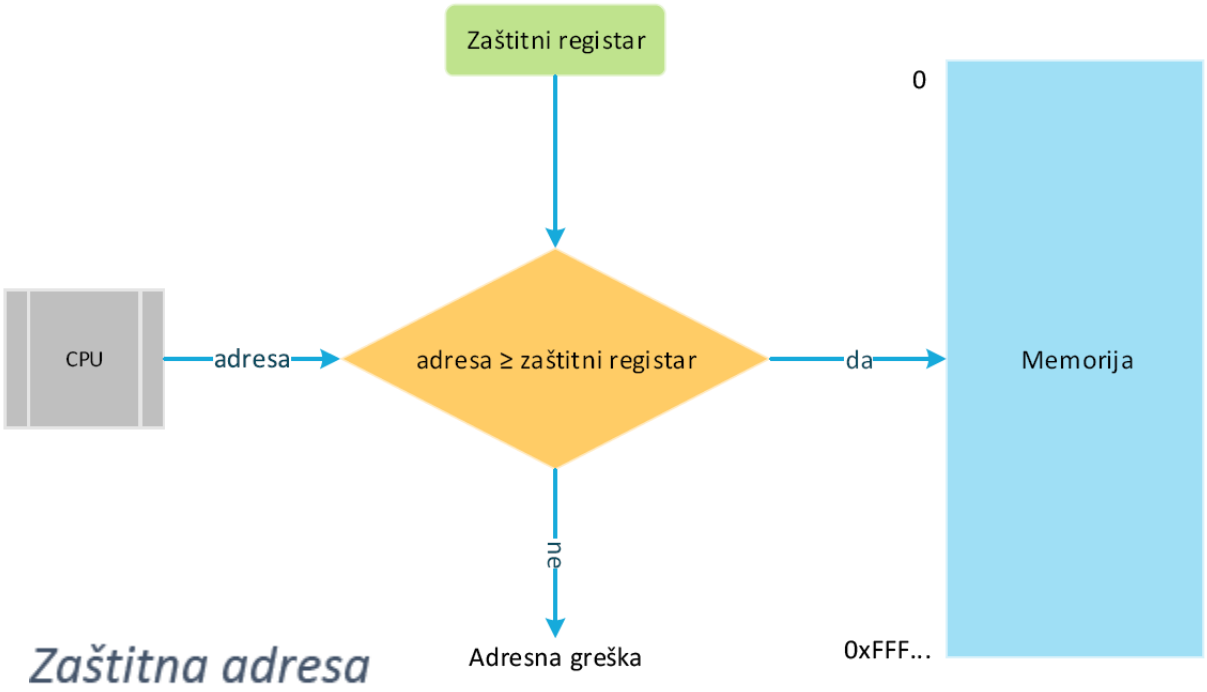
MMU kod monoprogramiranja (4)



- Pri ovakvom načinu upravljanja memorijom potrebno je zaštiti kôd i podatke operativnog sistema od slučajnih ili namernih izmena od strane korisničkog programa.
- Ovu zaštitu obično pruža hardver, a najjednostavniji pristup podrazumeva da se svaka adresa koju generiše korisnički program poredi sa sadržajem zaštitnog registra.

MMU kod monoprogramiranja (5)

- Ukoliko je adresa manja od sadržaja zaštitnog registra generiše se prekid (tipa pogrešna adresa) i operativni sistem preduzima odgovarajuće akcije. Obično je reakcija na takvu vrstu greške da se prekine korisnički program uz izdavanje odgovarajuće poruke o problemu.



MMU kod monoprogramiranja (6)

- Iako adresni prostor počinje od adrese 0, prva adresa u korisničkom programu nije 0, već prva adresa posle vrednosti zaštitnog registra.
- Povezivanje instrukcija i podataka sa memorijskim adresama može se izvršiti ili u vreme kompilacije ili u vreme punjenja.
    1. Ako je vrednost zaštitne adrese poznata u vreme kompilacije može se generisati apsolutni kôd, a adrese će se nizati počev od adrese koja se nalazi u zaštitnom registru.
    2. Alternativa je da kompilator generiše relokatibilni kôd. Tada se povezivanje odlaže do momenta punjenja.

MMU kod monoprogramiranja (7)

- U oba prethodna slučaja, vrednost zaštitne adrese mora da bude fiksirana tokom izvršenja programa.
- U nekim situacijama, u toku izvršavanja programa, potrebno je da se menja veličina dela memorije gde je smešten operativni sistem, a samim time i zaštitna adresa.
- Promena u vrednosti granične adrese zahteva izmenu sadržaja zaštitnog registra, ali i premeštanje korisničke memorije na nove lokacije u odnosu na novu vrednost granične adrese.
- Ovakva operacija je vremenski veoma zahtevna jer se dosta vremena troši na kopiranje memorije.

MMU kod multiprogramiranja

- Izvršavanje procesa veoma često može biti blokirano, npr. zbog čekanja na neke ulazno - izlazne operacije.
- Kod monoprogramiranja tada procesor prestaje sa radom, pa je stepen njegove neiskorišćenosti veliki.
- Kod savremenih računara, da bi se povećala iskorišćenost procesora, koriste se napredniji pristupi upravljanja procesima zasnovani na multiprogramiranju.
- Ovakvi pristupi omogućavaju da se više procesa može naći u memoriji i da se prividno izvršavaju istovremeno.

MMU kod multiprogramiranja (2)

- Kada se tekući proces zaustavi zbog čekanja na neku spoljašnju operaciju, procesor može da izvršava instrukcije nekog drugog procesa.
- U nekim slučajevima proces troši i do 80% ukupnog vremena izvršavanja zbog čekanja na neke spoljašnje operacije (npr. ulazno - izlazne operacije), dok se samo 20% vremena koristi za pravo izračunavanje.
- Ako bi u procesor istovremeno izvršavao 5 programa, deluje da bi iskorišćenost bila 100%. Ovo zahteva da ni u jednom trenutku ne postoje dva ili više procesa koji se nalaze u stanju čekanja.

MMU kod multiprogramiranja (3)

- Ako se p označi procenat vremena koje proces troši na čekanje i ako postoji n procesa u memoriji, tada je verovatnoća da se svih n procesa nađe u stanju čekanja p^n.
- Odavde se dobija da je procenat vremena u kojem procesor nije besposlen je: 1− p^n. Ovaj procenat se naziva iskorišćenost procesora, a broj n se naziva stepen multiprogramiranja.
- Navedena formula za iskorišćenost procesora je aproksimativne prirode, jer u stvarnosti procesi nemaju isto vreme čekanja (p).

MMU kod multiprogramiranja (4)

- Različite tehnike koje obezbeđuju memoriju potrebnu procesima za izvršavanje u sistemima zasnovanim multiprogramiranju:
  1. Prebacivanje
  2. Particije
     - Statičke particije
     - Dinamičke particije
  3. Straničenje
  4. Segmentacija

Prebacivanje

- Za vreme izvršavanja, program (proces) mora biti učitan u memoriju.
- Međutim, dok proces čeka na neke resurse nije neophodno da se nalazi u memoriji.
- Prebacivanje (swaping) je jednostavna tehnika upravljanja memorijom/procesima koju koristi operativni sistem u cilju povećanja iskorišćenosti procesora. Ideja je da se blokirani proces privremeno premesti u sekundarnu memoriju (disk), a da se u memoriju učita novi program.

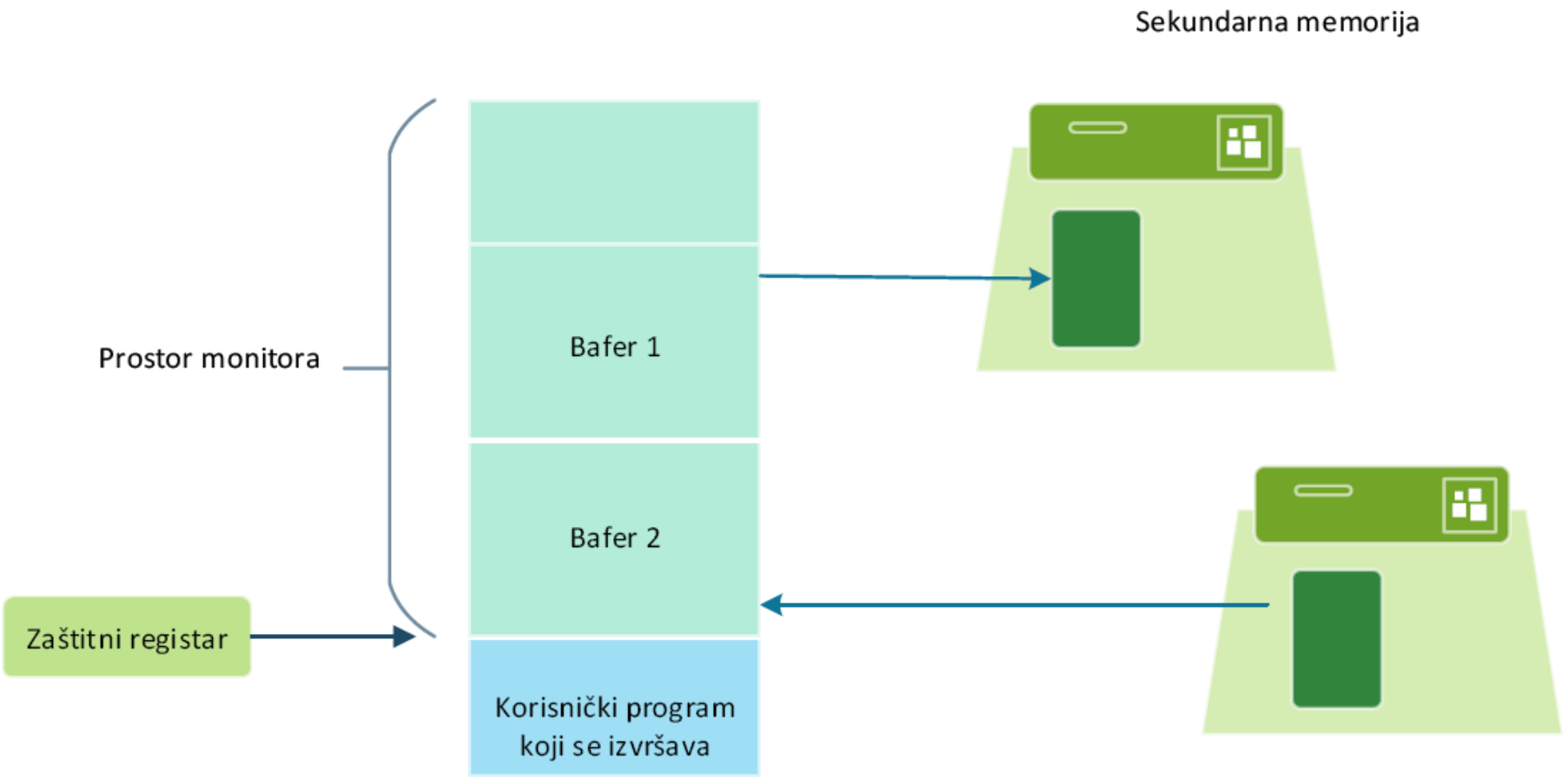
Prebacivanje (2)

- Na ovaj način se formira red privremeno blokiranih procesa. Procesi iz reda blokiranih procesa, koji su u mogućnosti da nastave sa radom, formiraju red spremnih procesa.
- Nakon prebacivanja blokiranog procesa u sekundarnu memoriju OS odlučuje da li će učitati novi program u memoriju, ili će aktivirati neki iz reda spremnih procesa.
- Efikasnost prebacivanja zavisi od brzine sekundarne memorije, a i da bude i dovoljno velika da prihvati memorijske slike za sve procese.

Prebacivanje (3)

- Vreme potrebno za prebacivanje konteksta nije zanemarljivo, pase teži da vremena trajanja procesorskih aktivnosti procesa budu srazmerno duža od vremena potrebnog za prebacivanja konteksta.
- Napredniji pristup, kada je prebacivanje u pitanju, podrazumeva da se efikasnost sistema poboljša preklapanjem prebacivanja jednog procesa sa izvršavanjem drugog procesa. Za realizaciju ovakvog pristupa potrebno je rezervisati bar dva memorijska regiona za prihvat dve memorijske slike procesa. Ovi regioni se obično nazivaju baferi.

Prebacivanje (4)



Baferi i sekundarna memorija

Prebacivanje (5)

- Sledećim pristupom se preklapa izvršavanje jednog procesa sa učitavanjem i upisivanjem drugih procesa u memoriju:
  1. Kada proces koji se izvršava oslobodi procesor on se premešta u slobodni bafer i pokreće se njegovo pomeranje na sekundarnu memoriju.
  2. Naredni proces se pomera iz bafera u oblast memorije predviđenu za izvršavanje.
  3. Zatim se u odgovarajući bafer učitava proces koji je sledeći na redu za izvršavanje.

Particije

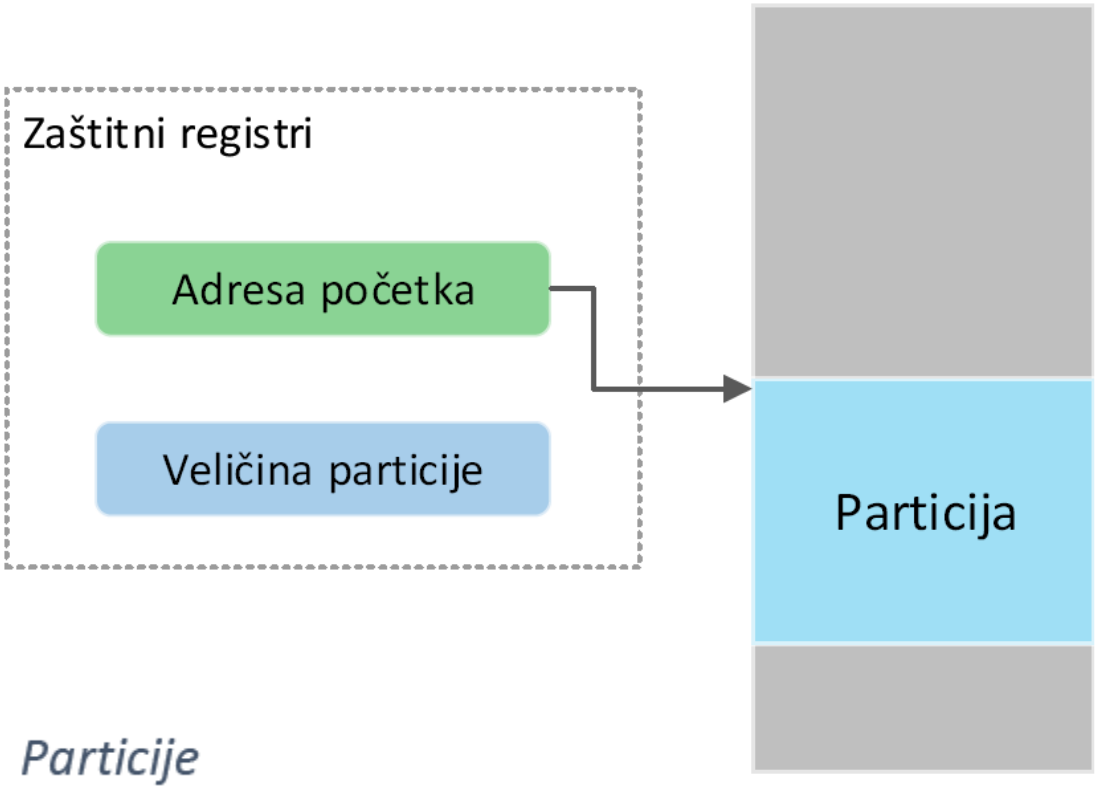
- Tehnika prebacivanja podrazumeva da se, pored operativnog sistema, u memoriji najčešće nalazi samo jedan proces.
- Porast kapaciteta memorije omogućio je da se umesto prebacivanja koristi efikasnija tehnika koja se naziva particionisanje memorije.
- Osnovna ideja je da se memorija podeli na n particija koje predstavljaju neprekidne delove memorije.
- U svaku particiju se može smestiti po jedan proces.
- Stepen multiprogramiranja u ovom slučaju je n.

Particije (2)

- Ako su sve particije zauzete, a neki program treba da započne izvršavanje, onda se formira red spremnih procesa. Kada se neka od zauzetih particija oslobodi, učitava se neki od spremnih procesa.
- Potrebno je particije zaštititi od pristupa drugih procesa. Ova zaštita se realizuje (najčešće) uz pomoć hardverske podrške koja se sastoji od po dva registra za svaku particiju.
- U zavisnosti od implementacije, ti registri mogu sadržati fizičke adrese početka i kraja particije, ali i adresu početka i veličinu particije.

Particije (3)

Svakom procesu je ograničen pristup samo na adrese koje odgovaraju njegovoj particiji. Ukoliko proces pokuša da pristupi nedozvoljenoj adresi, OS preduzima akcije (npr. prekida izvršavanje procesa).

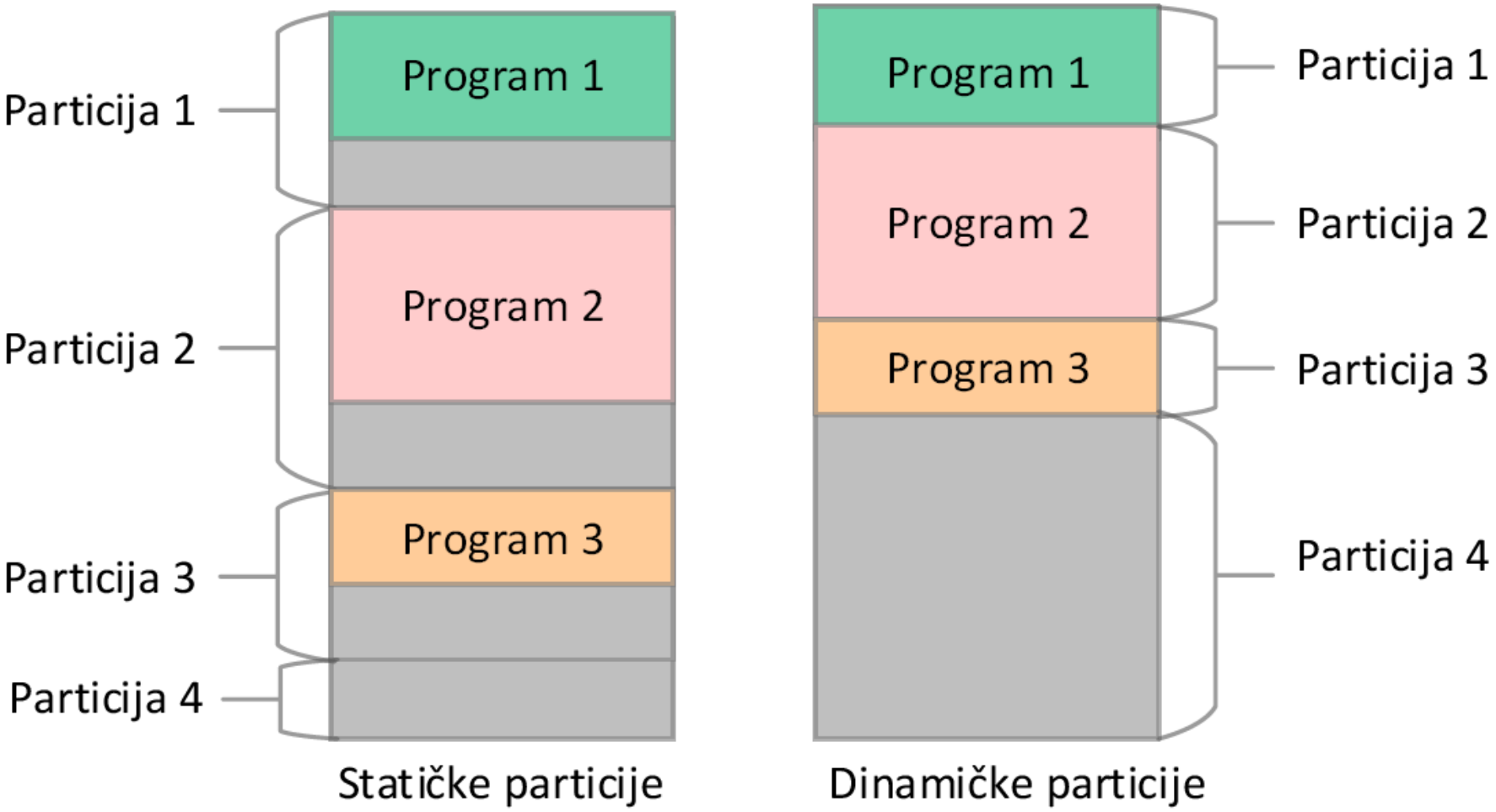


Particije

Particije (4)

- Postoje dva pristupa kada je patricionisanje memorije u pitanju:
  - Statički - podrazumeva da se veličina particija ne menja u toku rada.
  - Dinamički – dozvoljava se promena veličine memorijskih oblasti u zavisnosti od potreba procesa i stanja u sistemu.

Particije (5)



Statičke i dinamičke particije

Statičke particije

- Kod statičkog particionisanja radna memorija se statički deli na fiksni broj particija, gde svaka particija može sadržati proces.
- Broj particija zavisi od različitih faktora, kao što su: prosečna veličina procesa, veličina radne memorije, brzina procesora, itd.
- Veličine particija su fiksne i najčešće nisu jednake veličine.
- Particije manje veličine služe za manje i kratke procese, dok se veće particije ostavljaju za zahtevnije procese.

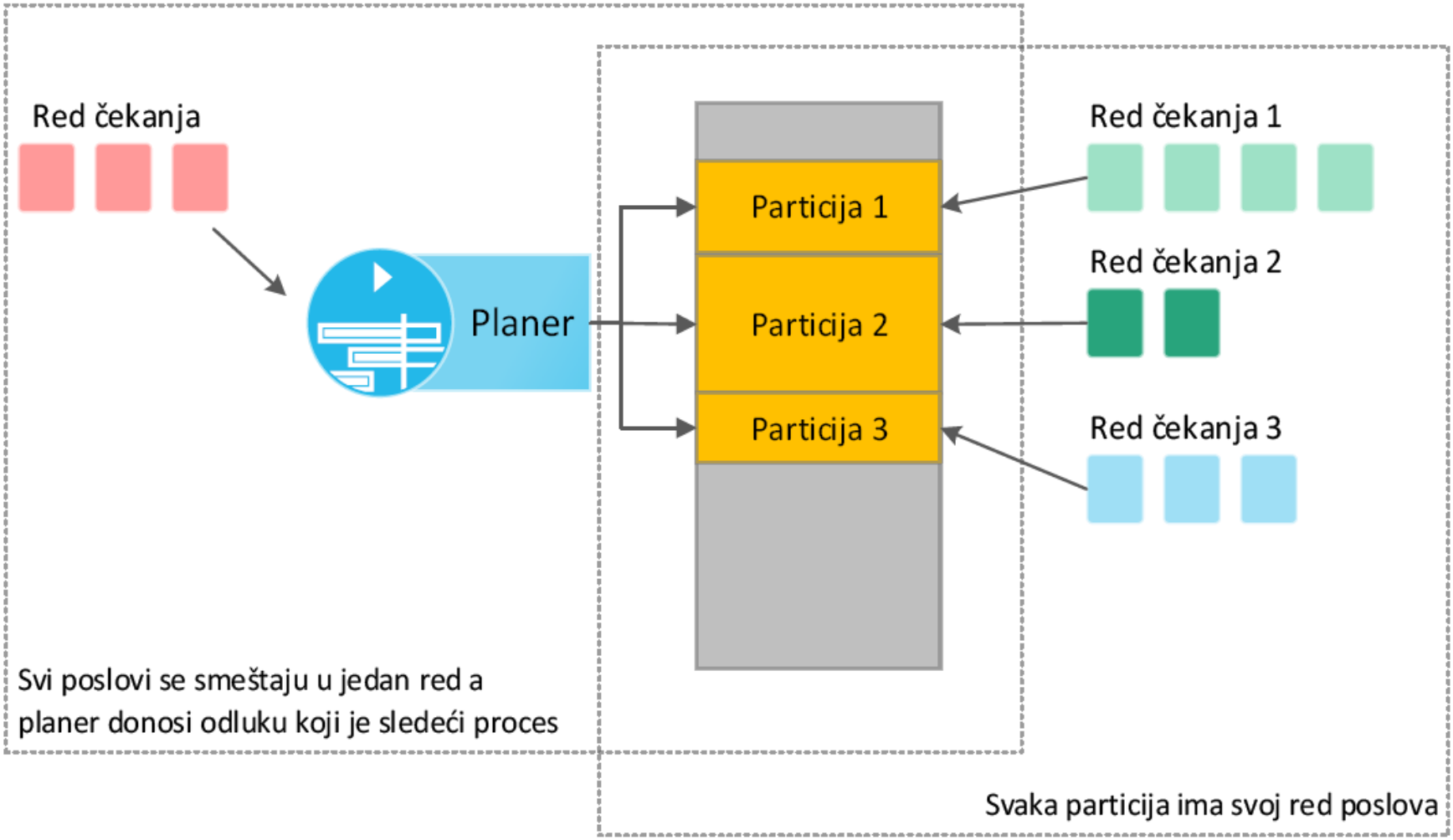
Statičke particije (2)

- Sistemi koji podržavaju statičke particije, procese koji pristižu smeštaju u red spremnih poslova.
- Planeri vode računa o memorijskim potrebama svakog od procesa kao i o raspoloživim particijama kako bi odredili kojim procesima treba dodeliti koju paticiju.
- Kada se procesu dodeli memorijski prostor, on se smešta u odgovarajuću particiju. Tek tada taj proces može konkurisati za dobijanje procesora i ostalih resursa koji su mu potrebni za izvršavanje.
-

Statičke particije (3)

- Kada proces završi sa svojim radom, on oslobađa particiju u kojoj se izvršavao i onda se ona može dodeliti drugom procesu.
- Postoje dve strategije za odabir particije u koju bi proces trebalo da bude smešten:
  1. Prva podrazumeva da svaka particija ima svoj red poslova u koji se smeštaju procesi čiji memorijski zahtevi odgovaraju veličini odgovarajuće particije.
  2. Drugi pristup je da se svi poslovi smeštaju u jedan red poslova, a da planer donosi odluku koji je sledeći proces koji bi trebalo da se izvrši.

Statičke particije (4)



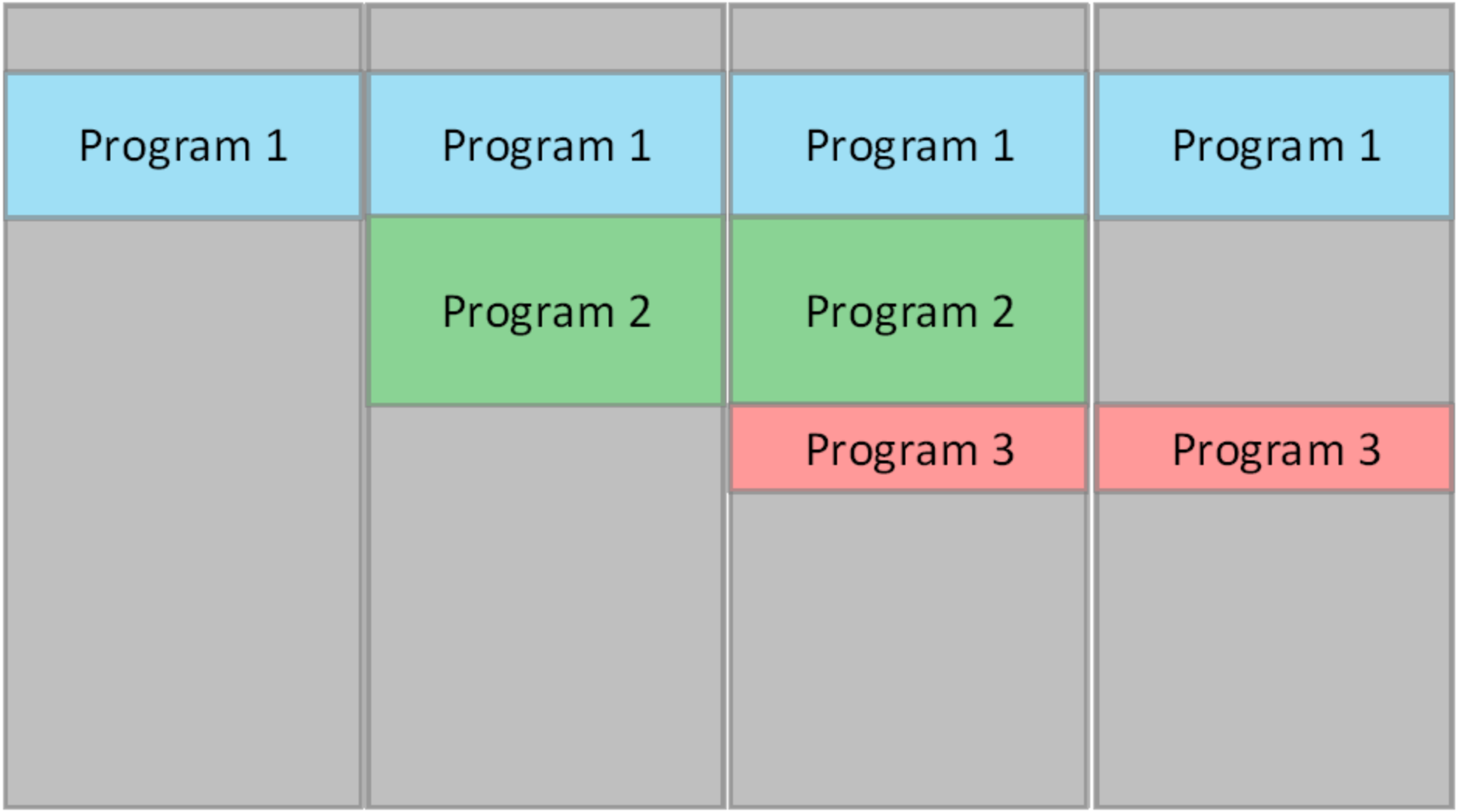
Dinamičke particije

- Dinamički pristup kod particionisanja memorije je sličan statičkom, osim što veličine particija nisu fiksirane.
- Sistemi koji podržavaju dinamičke particije moraju voditi računa o slobodnoj i zauzetoj memoriji.
- Dinamičke particije se realizuju na sledeći način:
  - Pri pokretanju računara, u memoriju se učitava operativni sistem.
  - Ostatak memorije koji je slobodan obrazuje jednu slobodnu particiju.

Dinamičke particije (2)

- Dalja aktivnost je:
  - Kada se pokrene proces koji je prvi predviđen za izvršavanje, slobodna particija se deli na dva dela: jedan deo je onaj koji se dodeljuje procesu, a drugi deo ostaje slobodan za ostale.
  - Vremenom neki procesi završavaju svoje izvršavanje, ostavljajući pri tome prazne particije koje su koristili.
  - Kada novi proces dođe na red za izvršavanje, pronalazi se dovoljno velika slobodna particija, a zatim se iz nje izdvaja onoliko memorije koliko je procesu potrebno.
  - Nakon završetka izvršavanja, proces oslobađa memoriju. Ako pored upravo oslobođene memorije već postoji slobodna particija, oslobođena memorije se objedinjava sa susednom slobodnom particijom.

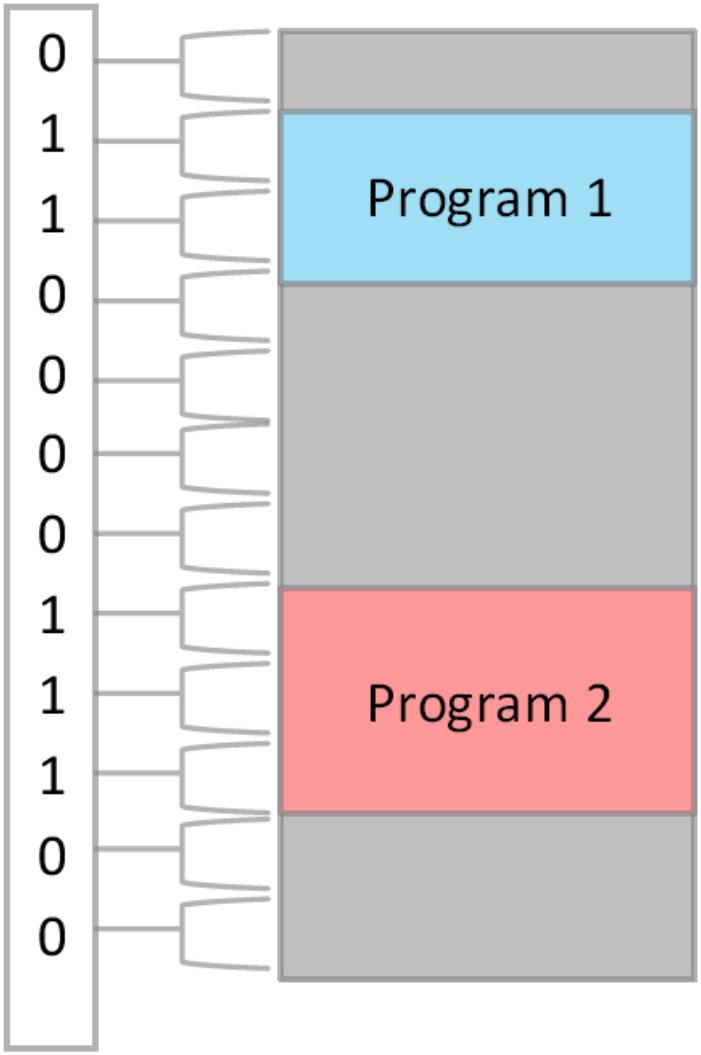
Dinamičke particije (3)



Dinamičke particije (4)

- Evidenciju o slobodnim i zauzetim delovima memorije operativni sistem obično vodi korišćenjem bit mapa ili povezanih listi.
  - Bit mape podrazumevaju da se memorija podeli na jednake delove i da se svakom od tih delova pridruži po jedan bit.
    - Ovaj bit ima vrednost 1 ukoliko je odgovarajući deo memorije zauzet a u suprotnom je 0.
    - Za punjenje procesa u memoriju potrebno je pronaći dovoljno veliki niz nula u bit mapi.
    - Implementacija bit mapa je jednostavna, ali je vreme potrebno za pretraživanje i pronalažene slobodne particije dosta veliko.

Dinamičke particije (5)

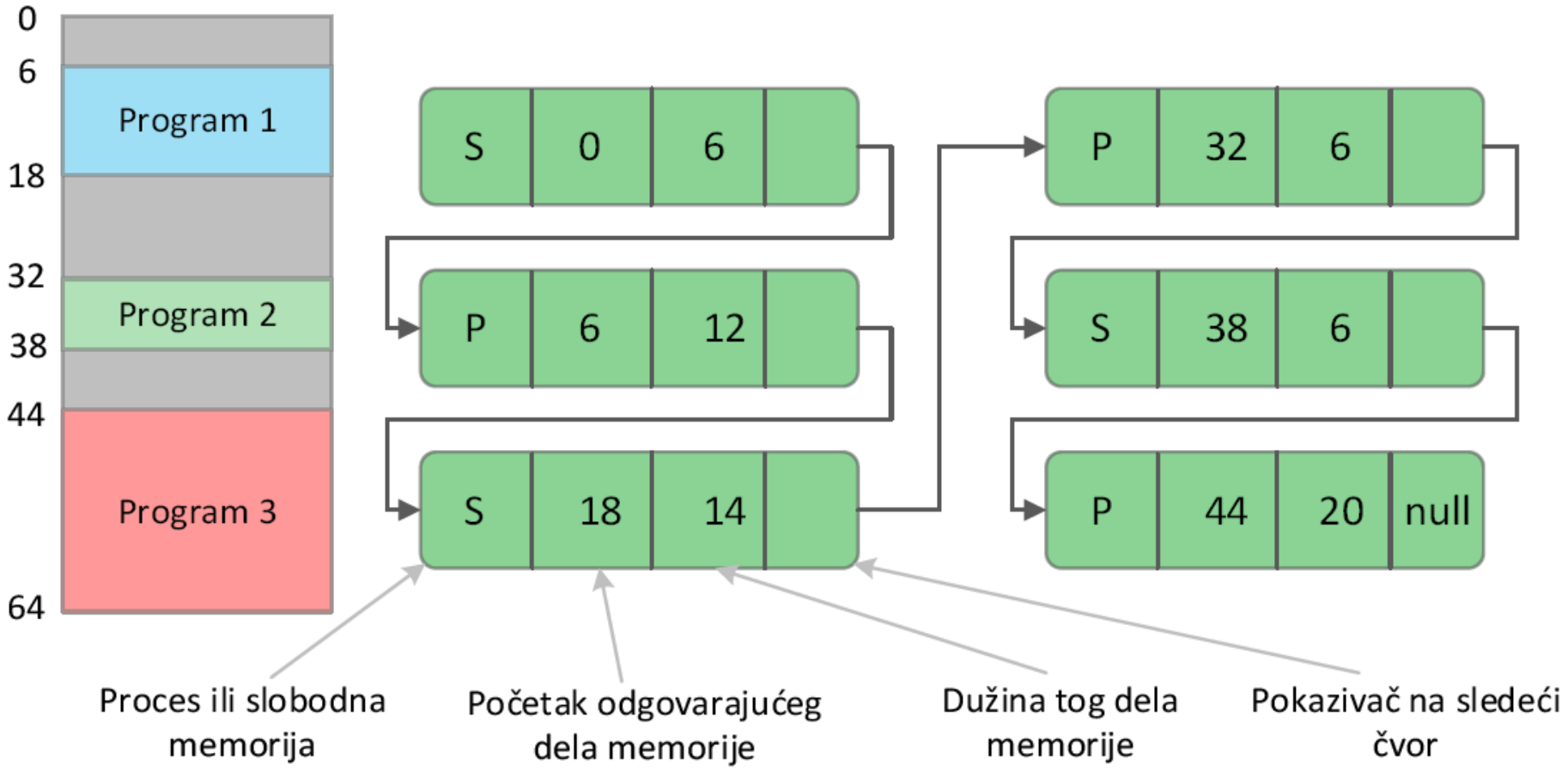


Bit mapa

Dinamičke particije (6)

- Povezana lista sastoji se od čvorova, pri čemu svaki čvor sadrži sledeće podatke:
  - Da li je u odgovarajući deo na koji se čvor odnosi smešten proces ili je u pitanju slobodna memorija;
  - Početak odgovarajućeg dela memorije;
  - Dužina tog dela memorije;
  - Pokazivač na sledeći čvor koji čuva informacije o narednom delu memorije.
- U nekim implementacijama se liste slobodnih i zauzetih delova memorije implementiraju odvojeno. Time se olakšava pretraživanje slobodnih delova, ali se otežava ažuriranje obe liste.

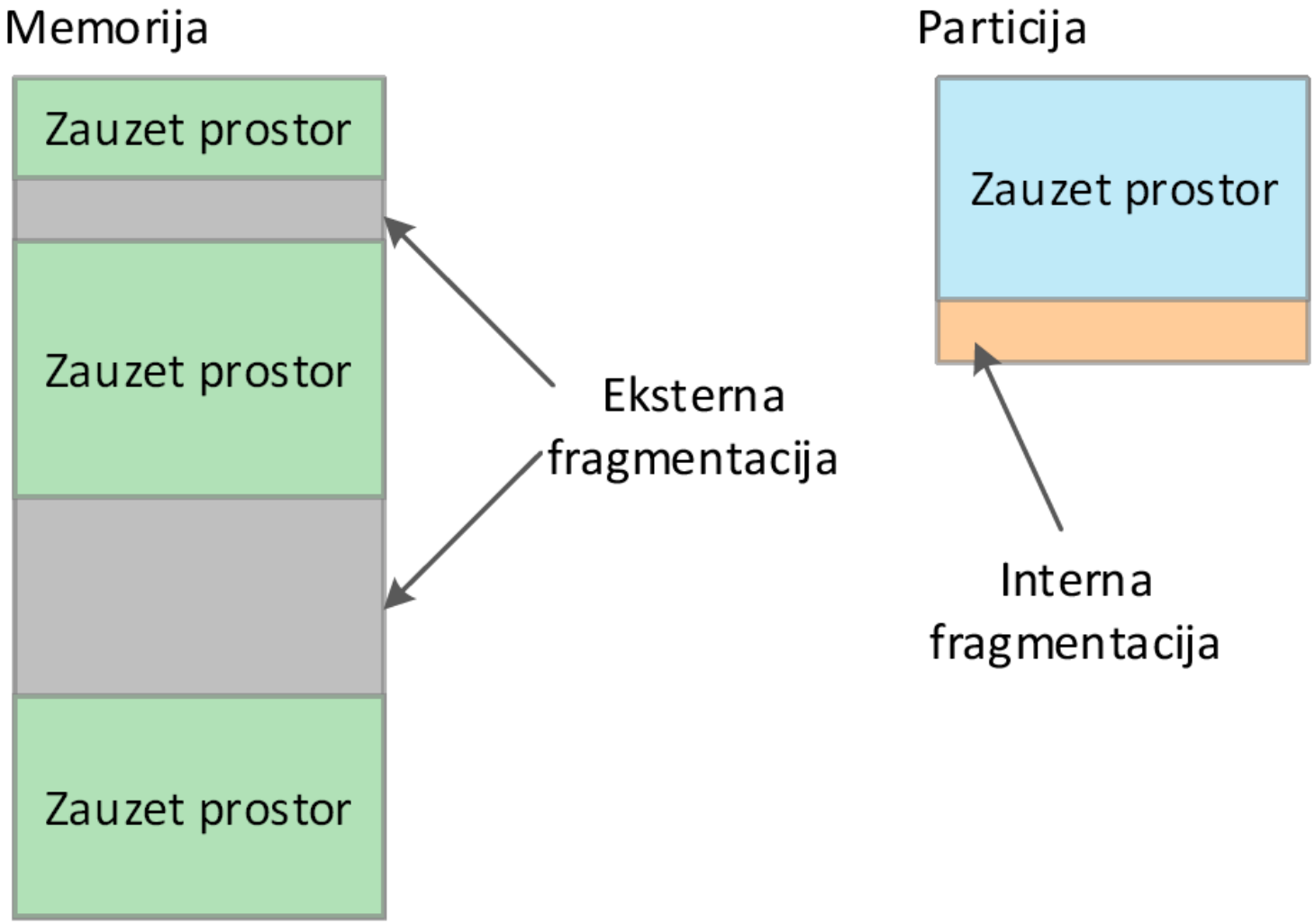
Dinamičke particije (7)



Fregmentacija

- Fragmentacija nastaje kada u računarskom sistemu postoje delovi memorije koji su slobodni, ali ih sistem ne može iskoristiti.
- Fragmenti su neiskorišćen slobodan memorijski prostor, iako postoje procesi koji čekaju da se izvrše.
- Fragmentacija može biti:
  - Interna se javlja kod statičkog particionisanja memorije i odnosi se na preostali neiskorišćen prostor koji ostaje kada se proces smesti u veću particiju.
  - Eksterna nastaje kod dinamičkog particionisanja, kada su sve slobodne lokacije zajedno dovoljne za izvršavanje procesa, ali ne formiraju jedinstvenu particiju.

Fragmentacija (2)

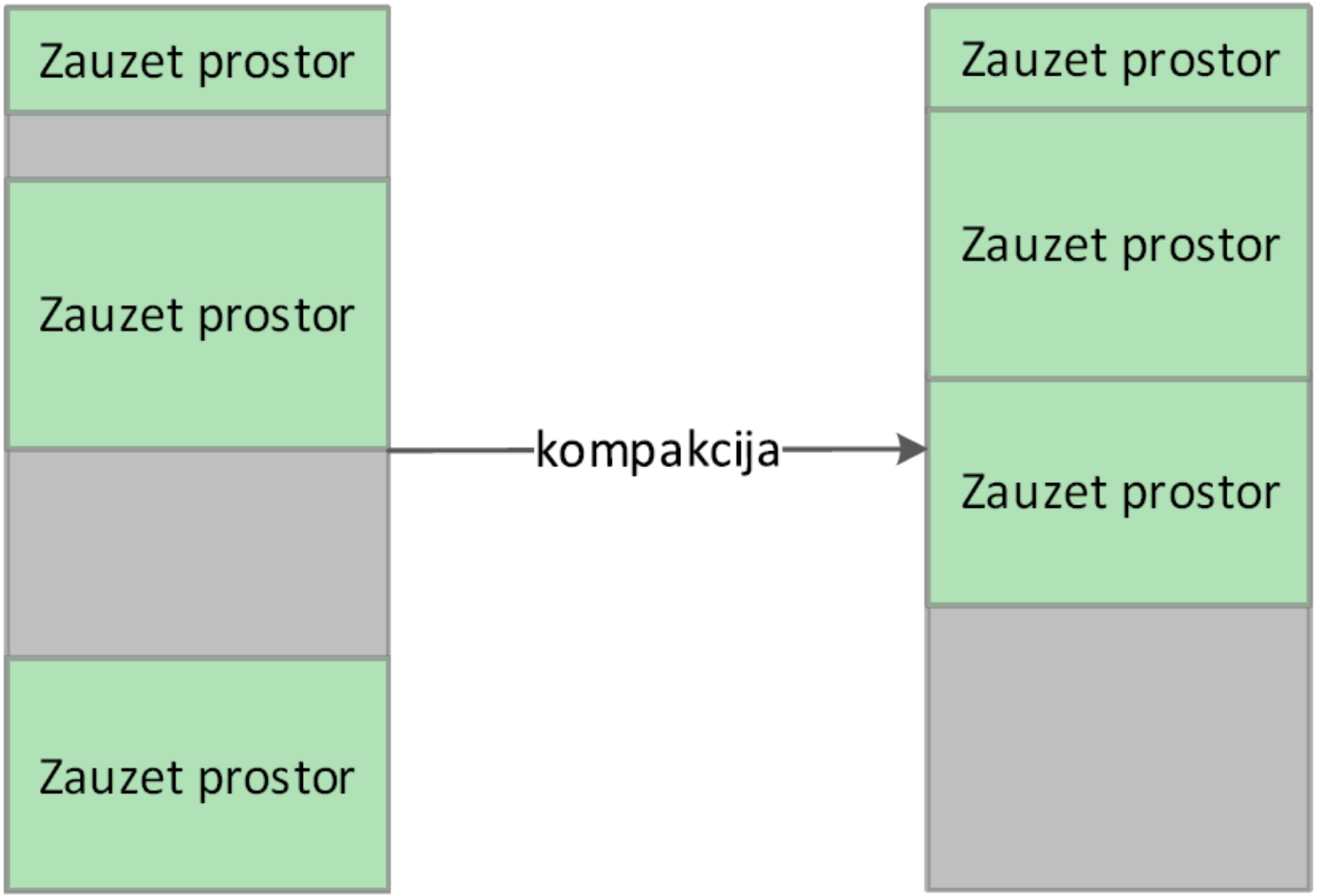


Fragmentacija

Fregmentacija (3)

- Način da se reši problem eksterne fragmentacije kod dinamičkih particija je kompakcija.
- Kompakcija podrazumeva da se sistem zaustavi i da se izvrši relokacija procesa tako da se oni sabiju na početak (ili kraj) memorije, pri čemu se slobodna memorija grupiše na drugi kraj memorije i formira jedinstvenu particiju.
- Ova operacija je vremenski dosta zahtevna i sistem će izgubiti na efikasnosti ako se često koristi.
- Upravljanje memorijom ne bi trebalo zasnivati na kompakciji.

Fragmentacija (4)



Kompakcija memorije

Algoritmi za dodelu memorije

- Procesu se na različite načine može dodeliti deo raspoložive memorije koji mu je potreban.
- Algoritam prvo poklapanje podrazumeva da se proces smesti u prvu slobodnu zonu koja je dovoljno velika da on može da stane u nju.
  - Pretraga za slobodnim prostorom kreće od početka memorije.
  - Po smeštanju se deo memorije koji je proces zauzeo označava kao zauzet, a preostali deo memorije i dalje ostaje na raspolaganju drugim procesima.
  - Vreme potrebno za dodelu memorije procesu je relativno malo, ali je zato loša iskorišćenost memorije zbog česte pojave fragmentacije.

Algoritmi za dodelu memorije (2)

- Modifikacija prethodnog algoritma: pretraga za slobodnom memorijom ne kreće svaki put od početka, već da se nastavi od lokacije gde je prethodna pretraga stala. Ovakav pristup se naziva sledeće poklapanje. Pristup sledećeg poklapanja ima malo lošije performanse od prvog poklapanja.
- Slučajna politika je realizacija dodele memorije koja podrazumeva da se od svih slobodnih i dovoljno velikih zona u memoriji slučajno izabere ona u koju će se smestiti proces.
  Ovakvo rešenje je dosta zahtevno, jer je potrebno odrediti delove memorije koji su kandidati za smeštanje procesa, ali i generisati slučajan broj.

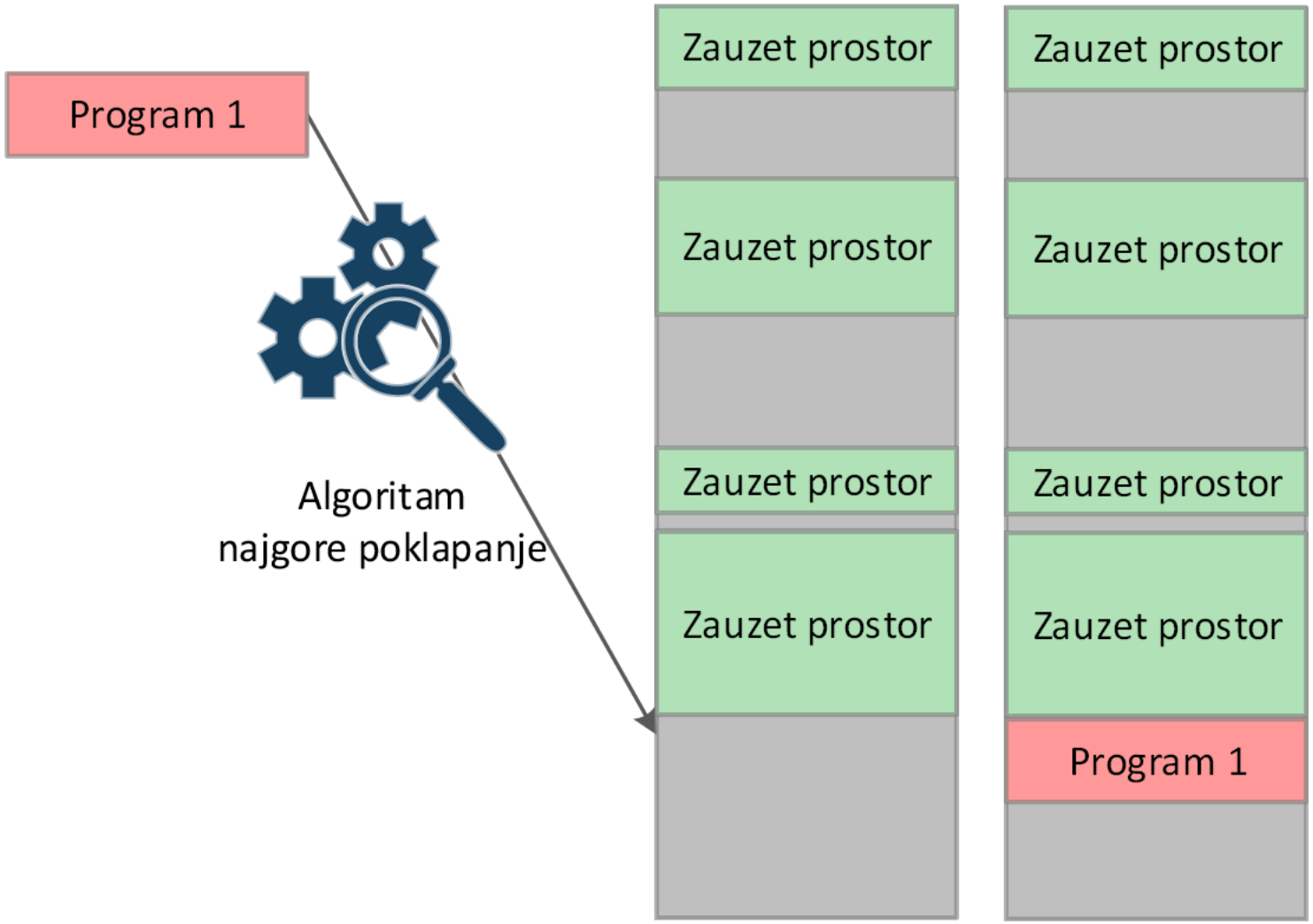
Algoritmi za dodelu memorije (3)

- Pristup najbolje poklapanje je zasnovan na ideji da se proces smesti u slobodni deo memorije koji je najmanji od svih slobodnih delova koji su veći od potreba procesa.
  Algoritam najboljeg poklapanja je dosta zahtevan - potrebno je proći kroz celu memoriju, izračunati veličinu svih slobodnih regiona, i pronaći najmanji koji je veći od memorijskih zahteva koje ima proces.
  U odnosu na pristup prvog poklapanja, ovaj algoritam zahteva nešto više vremena za dodelu memorije, ali su njegovi sveukupni rezultati veoma dobri.

Algoritmi za dodelu memorije (4)

- Algoritam najgore poklapanje zasniva se na ideji da se proces smesti u deo memorije u koji se najgore uklapa, tj. u region sa najvećim kapacitetom.
  Ovakva ideja proistekla je iz pretpostavke da bi ovakvim raspoređivanjem procesa u što veće particije slobodni ostatak mogao da se iskoristi za neki drugi proces.

Algoritmi za dodelu memorije (5)



Algoritam najgore poklapanje

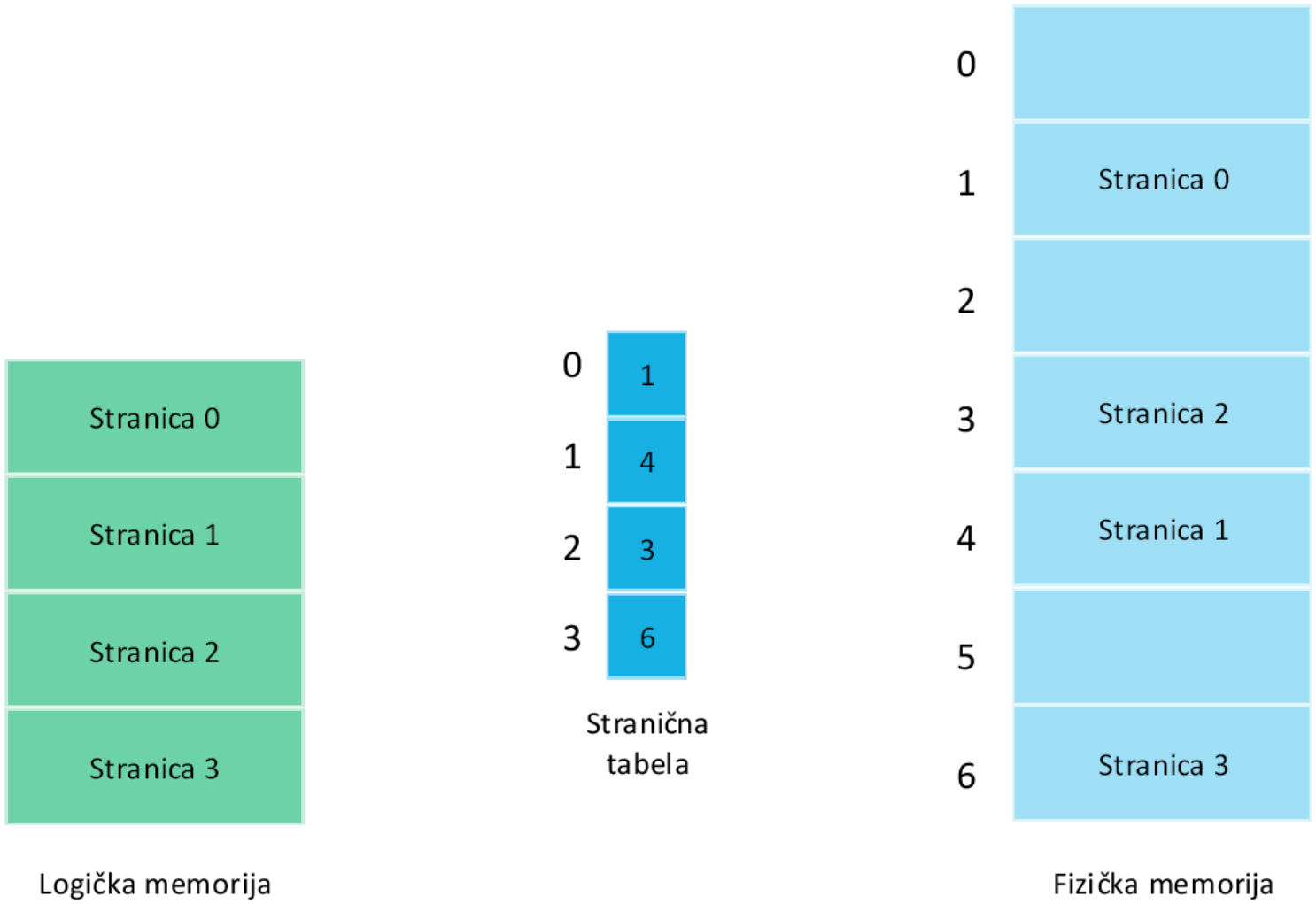
Straničenje

- Straničenje dozvoljava da memorija dodeljena procesu ne bude alocirana u jednom komadu, tj. isključivo na adresama koje se nalaze jedna za drugom.
- Ovakvim pristupom se omogućava da proces dobije memoriju, bez obzira gde se ona nalazi, ukoliko je slobodna.
- Osnovna ideja straničenja je da se radna memorija podeli na okvire fiksirane veličine, a da se logička memorija potrebna procesu podeli na stranice koje su iste veličine kao i okviri.

Straničenje (2)

- Kada proces treba da se učita u memoriju, pronalaze se slobodni okviri i u njih se smeštaju odgovarajuće stranice.
- Za vreme izvršavanja programa procesor radi sa logičkim adresama. Kod ovakve organizacije logička adresa se sastoji iz dva dela:
  1. Broja stranice - p,
  2. Pozicije u okviru stranice - d.
- Logičke adrese je potrebno prevesti u fizičke, a za to se koristi posebna struktura koja se naziva stranična tabela.

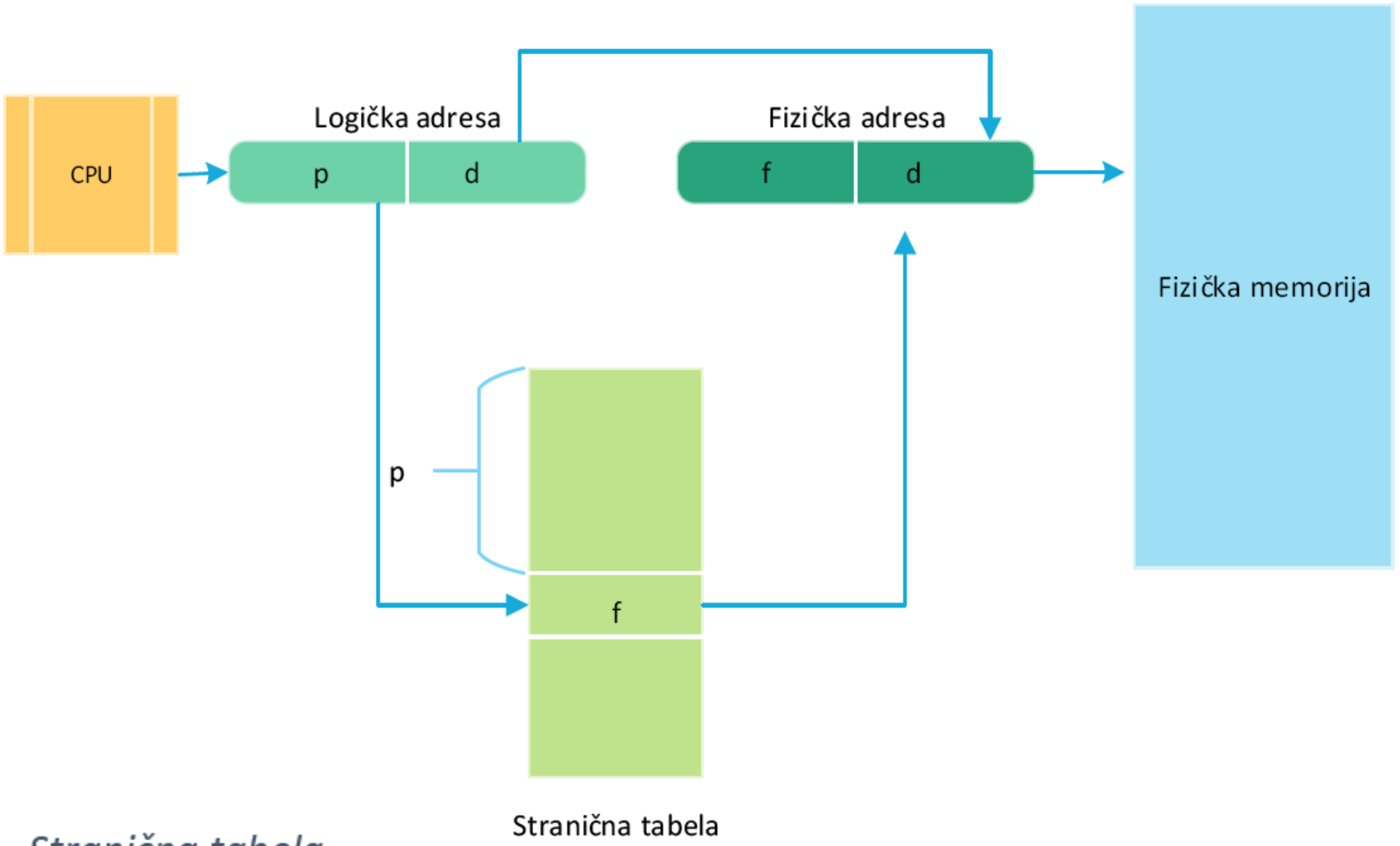
Straničenje (3)



Straničenje (4)

- Operativni sistem generiše po jednu straničnu tabelu za svaki proces koji se izvršava.
- Stranična tabela svakog procesa sadrži podatke o tome u kojim okvirima u memoriji su smeštene njegove stranice.
- Fizička adresa se dobija na osnovu ovih podataka dodavanjem vrednosti pozicije u stranici na adresu okvira u kojem se nalazi odgovarajuća stranica.

Straničenje (5)



Stranična tabela

Straničenje (6)

- Za određivanje koji proces je sledeći za izvršavanje odgovoran je planer poslova.
- Kada proces treba da krene sa izvršavanjem, planer izračunava njegovu veličinu izraženu u stranicama i proverava raspoložive memorijske okvire.
- Ukoliko postoji dovoljno slobodnih okvira, stranice procesa se pune u dodeljene okvire, a broj okvira se smešta u straničnu tabelu tog procesa.

Straničenje (7)

- Kod straničenja nema eksterne fragmentacije, jer svaki slobodni okvir može da bude dodeljen.
- Međutim, skoro sigurno postoji izvesna interna fragmentacija, jer obično poslednji okvir dodeljen procesu ne bude potpuno popunjen.
- Prednost straničenja je mogućnost deljenja zajedničkog koda - ukoliko se kôd ne samomodifikuje u toku izvršavanja, onda je moguće da više korisnika koristi istu kopiju tog koda u memoriji.

Straničenje (8)

- Zaštita memorije vrši se uz pomoć zaštitnih bitova koji se pridružuju svakoj stranici i čuvaju u straničnoj tabeli.
- Jednim bitom se označava da li se određena stranica može čitati i modifikovati ili samo čitati.
- Pošto svaki pristup memoriji podrazumeva čitanje stranične tabele radi određivanja broja memorijskog okvira, tom prilikom se proverava da li se stranica može modifikovati ili je dozvoljeno samo njeno čitanje.

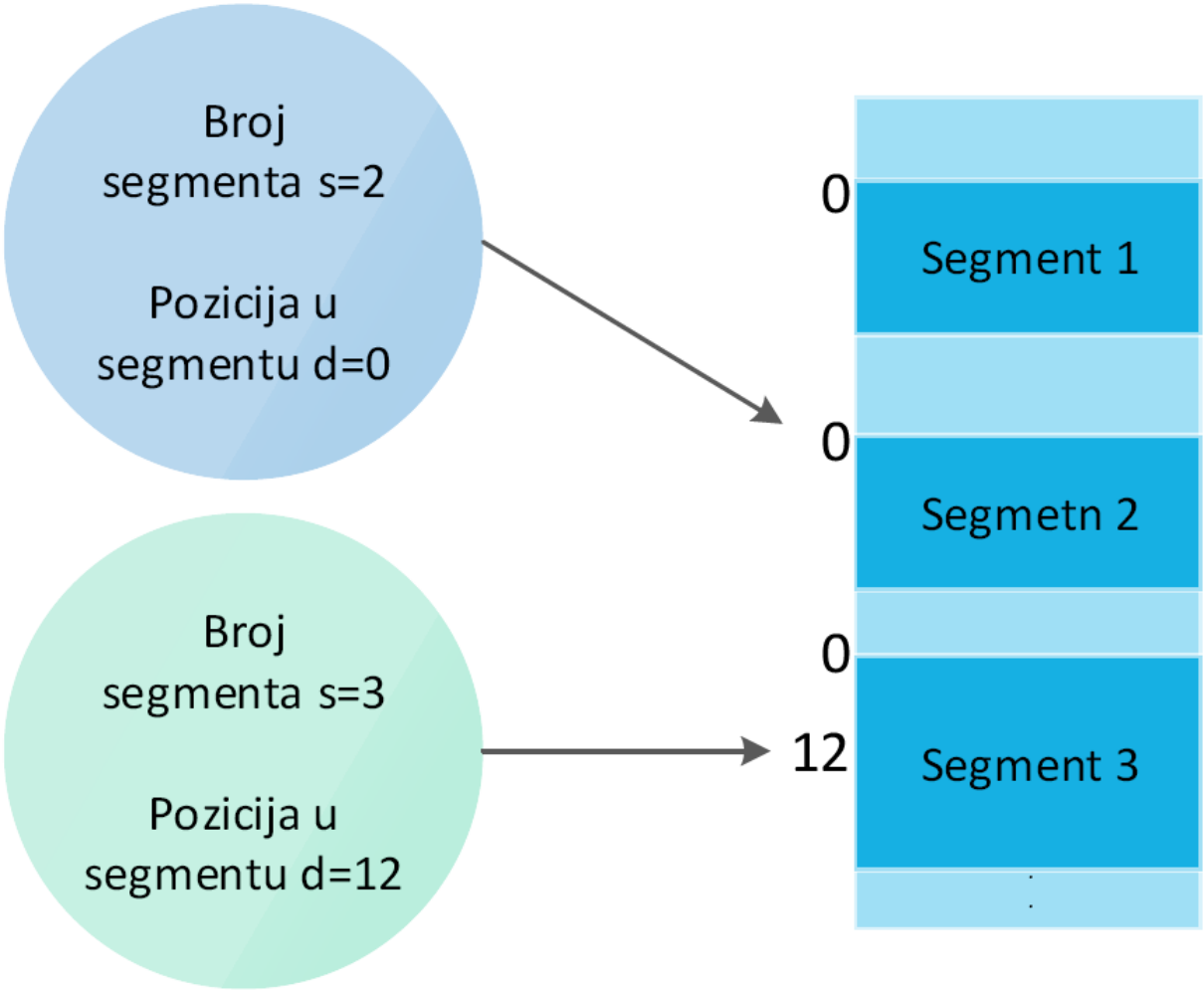
Segmentacija

- Prevedeni korisnički program se može posmatrati kao skup različitih logičkih celina – segmenata.
  - Na primer, neke od njih su: glavni program, stek, hip, biblioteke, globalni podaci, itd.
- Osnovna ideja segmentacije je da se svakom segmentu dodeli poseban memorijski prostor.
  - Svaki segment je atomičan, tj. ili će se ceo segment naći u memoriji ili neće uopšte biti učitan u memoriju.
  - Segmenti se mogu naći bilo gde u memoriji, ali jedan segment mora biti u neprekidnom memorijskom bloku.
  - Nije neophodno da segmenti budu iste veličine, već se svakom segmentu dodeljuje tačno onoliko memorije koliko je potrebno.

Segmentacija (2)

- Slično kao kod straničenja, logičke adrese kod segmentacije se sastoje iz dva dela:
  1. Broja segmenta - s,
  2. Pozicije u segmentu – d.
- Dvodimenzionalne logičke adrese potrebno je preslikati u jednodimenzione fizičke adrese. Za te potrebe uvodi se segmentna tabela.

Segmentacija (3)



Segmentacija

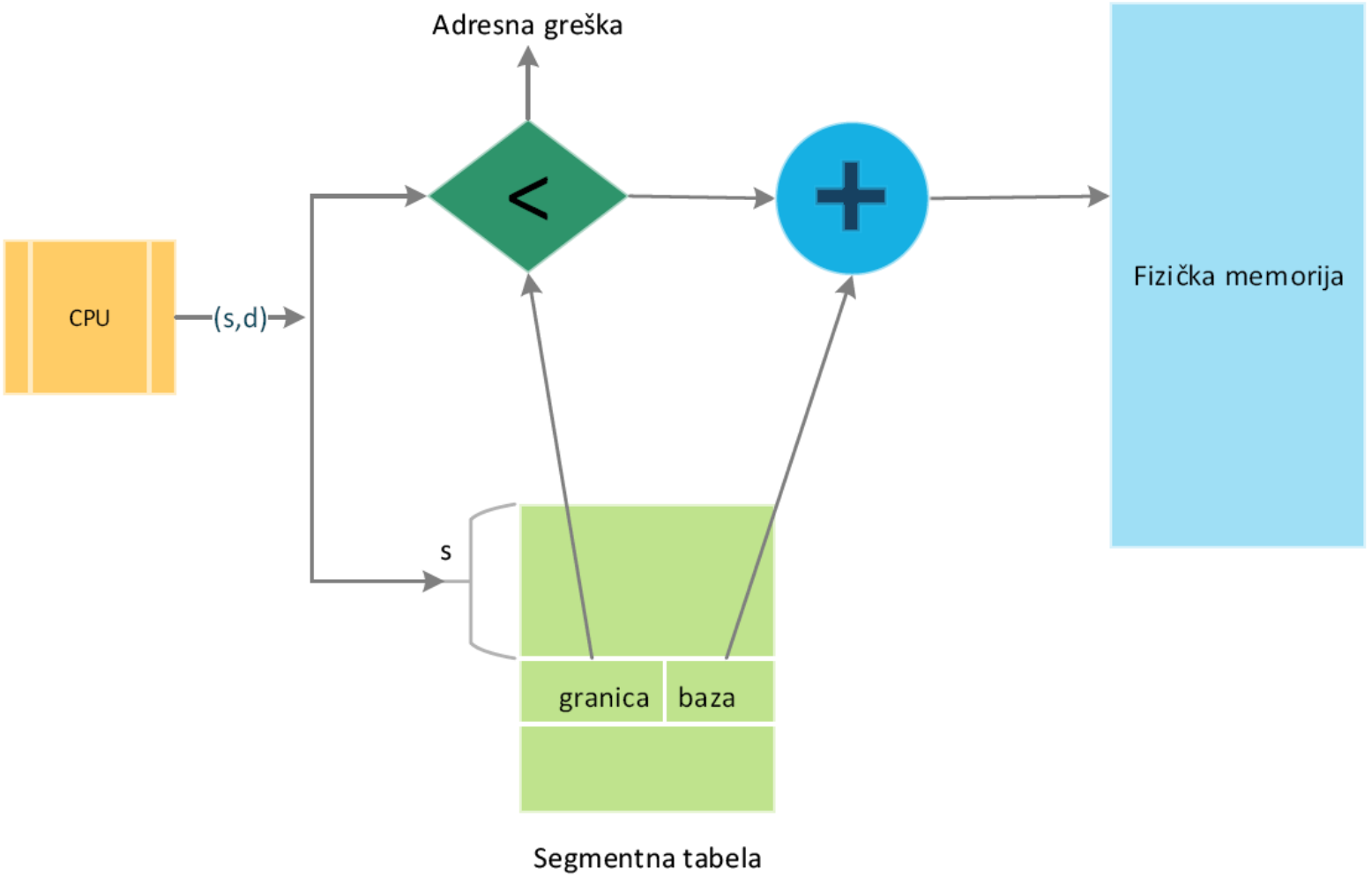
Segmentacija (4)

- Svakom procesu odgovara jedna segmentna tabela - po jedan red za svaki segment procesa.
  - Red segmentne tabele sadrži redni broj segmenta, adresu početka segmenta u memoriji i dužinu segmenta.
  - Kao i kod straničenja, segmentna tabela može biti smeštena u registrima, asocijativnoj memoriji ili u radnoj memoriji.
  - Bazni registar segmentne tabele STBR sadrži adresu lokacije gde je smeštena segmentna tabela.
  - Kako broj segmenata jednog procesa može da varira, potrebno je voditi računa o veličini segmentne tabele. Za to se obično koristi registar dužine segmentne tabele STLR .

Segmentacija (5)

- Mapiranje logičkih adresa u fizičke sastoji se od sledećih akcija:
  1. Najpre se za logičku adresu (s, d) proverava da li je broj segmenta validan, tj. da li je s < STLR.
  2. Ako je broj segmenta regularan, onda se on dodaje na vrednost registra STBR. Time se dobija memorijska adresa odgovarajućeg reda u segmentnoj tabeli.
  3. Sa te adrese se učitava dužina segmenta i fizička adresa početka segmenta. Prvo se proverava da li je pozicija u segmentu validna, tj. da li je d manje od dužine segmenta.
  4. Ako je to tačno, onda se na fizičku adresu početka segmenta dodaje *d* i dobija se odgovarajuća fizička adresa.

Segmentacija (6)



Preslikavanje logičkih adresa u fizičke

Segmentacija (7)

- Kao i kod straničenja, i kod segmentacije preslikavanje logičkih adresa u fizičke zahteva dva obraćanja memoriji, što se može ubrzati korišćenjem asocijativne memorije.
- Prednost segmentacije je i u zaštiti memorije:
  - S obzirom da segmenti predstavljaju logički definisane delove programa, verovatno je da će se sve stavke segmenta koristiti na isti način.
  - Dakle, očekivano je da neki segmenti sadrže instrukcije a neki podatke.
  - Segmenti koji sadrže instrukcije mogu se označiti tako da se mogu samo čitati i da se ne mogu modifikovati, čime se štite od neželjenih promena i pristupa.

Segmentacija (8)

- Segmentacija pruža mogućnost deljenja koda i podataka između različitih procesa.
  - Svaki proces ima svoju segmentnu tabelu.
  - Segmenti se dele ukoliko stavke u segmentnim tabelama dva različita procesa ukazuju na istu fizičku lokaciju.
- Dodeljivanje memorije je slično kao i kod straničenja.
  - Razlika je u tome što su segmenti različite veličine.
  - Potrebno je pronaći i dodeliti memoriju za sve segmente korisničkog programa.
  - Dodela memorije je problem dinamičke dodele memorije i rešava se prethodno opisanim algoritmima.

Segmentacija (9)

- Veličina segmenata je različita, pa segmentacija podseća na dinamičko particionisanje.
- Razlika između njih je u tome što se kod segmentacije proces ne nalazi u jednom memorijskom bloku, već se delovi procesa mogu naći bilo gde u memoriji.
- Kao i dinamičko particionisanje, segmentacija rešava problem interne fragmentacije ali ostaje problem eksterne fragmentacije.
- Međutim, eksterna fragmentacija je manje izražena jer su segmenti manji od kompletnog programa, pa se lakše mogu „umetnuti“.

Segmentacija (10)

- U nekim implementacijama kombinuju se segmentacija i straničenje kako bi se iskoristile prednosti i umanjili nedostaci koje imaju ova dva koncepta.
- Segmentacija sa straničenjem predstavlja pristup koji podrazumeva podelu segmenata na stranice čime se omogućava da segmenti ne moraju da se smeštaju u neprekidne memorijske blokove.
- Na ovaj način se veliki segmenti lakše smeštaju u memoriju i smanjuje se eksterna fragmentacija.

Zahvalnica

Najveći deo materijala iz ove prezentacije je preuzet iz knjige „Operativni sistemi“ autora prof. dr Miroslava Marića i iz slajdova sa predavanja koje je držao prof. dr Marić.

Hvala prof. dr Mariću na datoj saglasnosti za korišćenje tih materijala.