

ARHITEKTURA I ORGANIZACIJA RAČUNARA

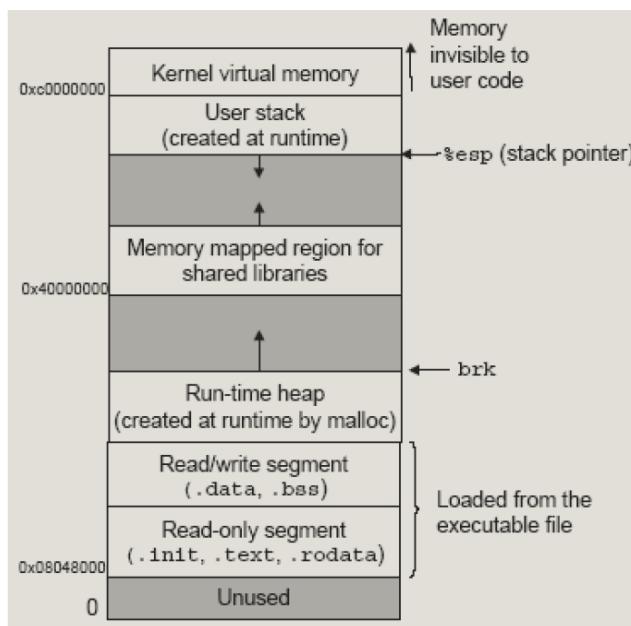
MEMORIJSKI SISTEM(5) VIRTUELNA MEMORIJA

VIRTUELNA MEMORIJA

- * Do prelaska na izvršenje, programi se obično drže u sekundarnoj memoriji u svom izvršnom objektnom obliku.
- * Prelasku na izvršenje programa prethodi punjenje glavne memorije programom. Sastavni deo punjenja je dodela memorije programu, tj. određivanje slobodnog bloka glavne memorije u koji se prenosi (kopira) program iz sekundarne memorije.
- * Po završetku izvršenja programu se "oduzima" memorija, tj. blok glavne memorije koji mu je bio dodeljen proglašava se slobodnim.

Operativni sistem upravlja memorijom

- * Poslovi dodele memorije programima, punjenja memorije i oslobađanja memorije od okončanih programa su u nadležnosti operativnog sistema računara. Uzeti zajedno, ovi poslovi nazivaju se **upravljanje memorijom**.
- * Oni se detaljno izučavaju u kursevima o operativnim sistemima računara, tako da o tome ovde nećemo detaljnije govoriti.
- * Međutim, upravljanje memorijom zahteva i određena arhitekturna i implementaciona rešenja, kojima ćemo se baviti u izlaganju koje sledi.



Memorijska slika procesa pod Linuxom

- Programi uvek počinju od virtuelne adrese 0x08048000.
- Stek korisnika uvek počinje od virtuelne adrese 0xbfffffff.
- Deljeni objekti se uvek pune u oblast počev od virtuelne adrese 0x40000000.

Virtuelna memorija kao apstrakcija glavne memorije

Upravljanje memorijom postaje složeno kada

- u glavnoj memoriji nema dovoljno prostora za držanje celih programa i njima pridruženih podataka u toku izvršenja programa, ili
- u glavnoj memoriji smenjuje se više procesa (programa u izvršenju) pri multiprogramsном radu računara.

U težnji da se korisnici računara **oslobode svih zaduženja vezanih za upravljanje memorijom, kao apstrakcija glavne memorije nastala je **virtuelna memorija**, razvijena sa ciljem da upravljanje memorijom učini efikasnijim i sa manje grešaka.**

Virtuelna memorija kao složeni sistem

- * U sistemima koji rade u multiprogramsном režimu upravljanje memorijom mora obezbititi izolovanje adresnih prostora različitih programa, ali i dozvoliti kontrolisanu deobu memorije za kooperaciju i sinhronizaciju programa.
- * Virtuelna memorija uspešno objedinjuje hardverske izuzetke, hardverom potpomognuto prevođenje adresa, glavnu memoriju, disk memoriju i jezgo operativnog sistema da svakom procesu obezbedi veliki, uniformni i privatni adresni prostor.

Odvajanje logičkih od fizičkih adresa

- * Ključna koncepcija na kojoj se zasniva virtuelna memorija je **odvajanje** virtuelnih, odnosno logičkih adresa, koje se koriste u programu, od adresa lokacija glavne memorije i sekundarne memorije u kojima se programi čuvaju.
- * Virtuelne adrese sistem generiše pri prevodenju i/ili povezivanju programa. Ove adrese na jedinstven način određuju elemente koji pripadaju jednom logičkom adresnom prostoru.

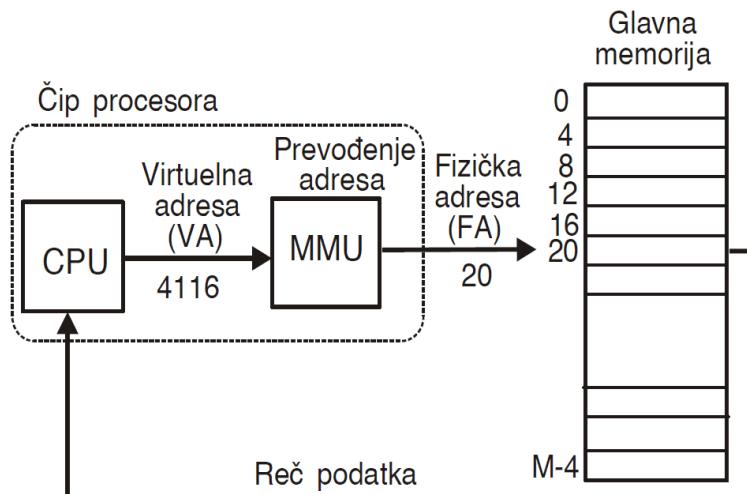
Prevođenje adresa pri svakom obraćanju memoriji

- * Sistem odlaže konkretnu dodelu memorije programu **sve do trenutka kada počinje izvršenje programa**. Naime tek tada operativni sistem, prema trenutnom zauzeću memorije, može na najbolji način dodeliti memoriju tom programu.
- * Pri izvršenju programa **svako obraćanje memoriji zahteva prevođenje virtuelne adrese**, koju generiše procesor, u fizičku adresu, određenu dodelom memorije programu i relativnom adresom instrukcije ili podatka u programu.

Jedinica za upravljanje memorijom (MMU)

- * Prevođenje virtuelnih u fizičke adrese vrši jedinica za upravljanje memorijom (engl. **memory management unit, skr. MMU**), koja je obično implementirana u čipu procesora.
- * Mogućnost da se programi izvršavaju pri bilo kojoj dodeli memorije programu zahteva i poštovanje sledećeg **ograničenja**: u programima se **ne smeju koristiti načini adresiranja sa fiksnim adresama**, kao što je to direktno odnosno apsolutno adresiranje.
- * Najpogodniji način adresiranja je **bazno adresiranje**, koje je u savremenim procesorima često i jedini način adresiranja podataka u memoriji

Preslikavanje virtuelne u fizičku adresu pri izvršenju instrukcije Load



Segmentna virtuelna memorija

- * Virtuelna memorija kod koje se virtuelni adresni prostor deli u segmente čije su dužine određene dužinom programa naziva se segmentna virtuelna memorija (engl. **segmented virtual memory**).
- * Kod nje se i dodela memorije programima vrši u takvim segmentima.
- * Pojava velikog broja malih slobodnih blokova memorije, koji se pojedinačno ne mogu iskoristiti za nove dodele, a zbirno mogu činiti značajan deo memorije - **spoljašnja fragmentacija memorije**.

Stranična virtuelna memorija

- * Dodata memorije programima može se pojednostaviti ako se svaki program "upakuje" u izvestan broj stranica fiksnih dužina. Za datu veličinu stranica i dužinu programa potreban broj stranica određuje se tako da njihova zbirna dužina bude jednaka ili veća od dužine programa.
- * Virtuelna memorija kod koje se virtuelni adresni prostor deli u ovakve stranice fiksnih dužina naziva se stranična virtuelna memorija (engl. **paged virtual memory**).

Unutrašnja fragmentacija memorije

- * Kod nje se i memorija dodeljuje programu u stranicama. Pri tome se takođe javljaju neiskorišćeni delovi memorije, ovog puta u vidu **neiskorišćenih delova poslednjih stranica dodeljenih programima.**
- * Ova pojava naziva se **unutrašnja fragmentacija memorije.**

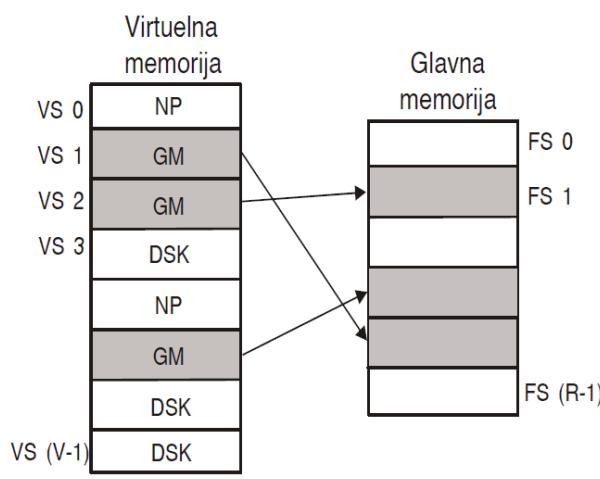
Stranična virtuelna memorija

- * Kod stranične virtuelne memorije virtuelni adresni prostor podeljen je na **virtuelne stranice**, a fizički adresni prostor na fizičke stranice, za koje se često koristi i termin **okvir stranice**.
- * Veličina stranica L određena je celobrojnim stepenom dvojke, i najčešće iznosi $L=4$ KB.
- * Virtuelni adresni prostor veličine 2^V bajtova sadrži V virtuelnih stranica označenih brojevima 0, 1, ..., $V-1$.

Elementi virtuelne i fizičke adrese

- * Fizički adresni prostor veličine 2^r bajtova sadrži R fizičkih stranica označenih brojevima 0, 1, ..., R-1.
- * Data virtuelna adresa a_v , deli se na pomeraj unutar stranice p i broj virtuelne stranice BVS.
- * Pri svakom obraćanju memoriji ova virtuelna adresa preslikava se u fizičku adresu a_f , koja se takođe deli na pomeraj unutar stranice p i broj fizičke stranice BFS.

Preslikavanje virtuelnih u fizičke stranice

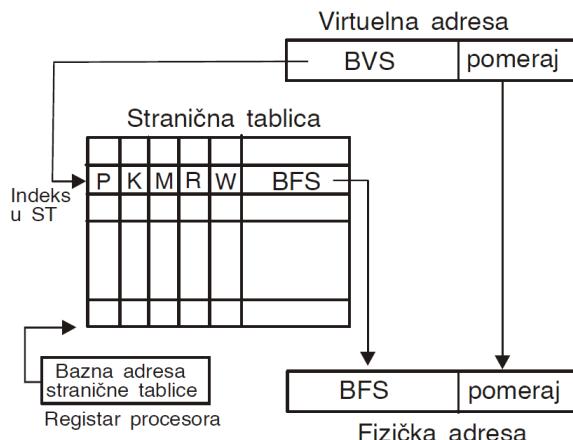


- GM- virtuelne stranice prisutne u glavnoj memoriji,
- DSK- virtuelne stranice prisutne na disku,
- NP- nepostojeće virtuelne stranice.

Stranična tablica

- * Prevođenje virtuelnih u fizičke adrese vrši se pri svakom obraćanju memoriji. Ovo prevođenje vrši se korišćenjem **stranične tablice (ST)**, koja za svaku virtuelnu stranicu ima po jednu stavku.
- * Stavka stranične tablice (SST) adresira se brojem virtuelne stranice BVS, dobijene iz virtuelne adrese a_v izdvajanjem pomeraja iz nje.
- * Ova stavka stranične tablice sadrži broj fizičke stranice BFS u koju se preslikava virtuelna stranica BVS.
- * Pomeraj u virtuelnoj adresi a_v se neizmenjen prenosi u polje pomeraja fizičke adrese.

Korišćenje stranične tablice pri prevođenju virtuelne u fizičku adresu



- P - prisutna u glavnoj memoriji,
- K-korišćena u proteklom periodu,
- M-modifikovana u glavnoj memoriji,
- R-dozvoljen pristup radi čitanja, i
- W-dozvoljen pristup radi upisa.

Greška stranice

- * Ako je virtuelna stranica prisutna u glavnoj memoriji, tj $P(SST)=1$, imamo pogodak stranice. $P(SST)=0$ signalizira da stranica nije prisutna u glavnoj memoriji, i tada se javlja greška stranice (engl. page fault).
- * Pored ove stranične tablice, mora postojati još jedna tablica koja za svaku kreiranu virtuelnu stranicu sadrži njenu adresu na disku.

Prevođenje adresa pri pogodku

- * Prevođenjem virtuelne adrese u slučaju pogotka upravlja MMU kroz sledeće korake, prikazane na slici a:

Korak 1: Procesor formira virtuelnu adresu VA i šalje je u MMU.

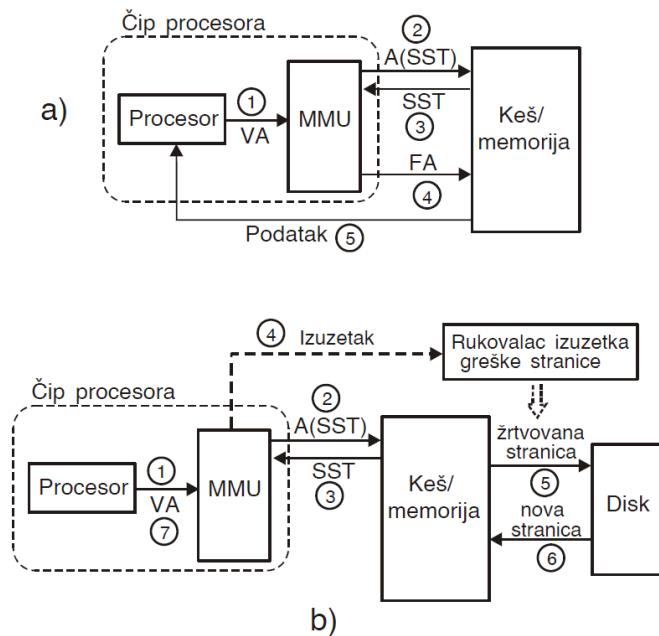
Korak 2: MMU formira adresu stavke stranične tablice A(SST) i upućuje zahtev za pribavljanje SST iz keša/glavne memorije.

Korak 3: Keš/glavna memorija vraćaju u MMU zahtevanu SST.

Korak 4: MMU formira fizičku adresu i dostavlja je kešu/glavnoj memoriji.

Korak 5: Keš/glavna memorija vraćaju procesoru zahtevanu reč podatka.

Učešće
hardvera
računara u
prevođenju
virtuelne
adrese
a) pri pogotku
stranice,
b) pri grešci
stranice



Prevođenje adresa pri grešci stranice

Greška stranice zahteva sadejstvo hardvera i kernela operativnog sistema, definisano sledećim koracima, ilustrovanim prethodnom slikom b.

- **Koraci 1÷3:** Istovetni su sa koracima 1÷3 pri pogotku stranice.
- **Korak 4:** Bit P(SST)=0 pa MMU aktivira izuzetak, koji predaje upravljanje procesorom rukovaocu izuzecima greške stranice u kernelu operativnog sistema.
- **Korak 5:** Rukovalac izuzecima greške stranice pronalazi žrtvovanu stranicu u glavnoj memoriji (onu koja će biti zamenjena), i ako je ona modifikovana, vraća je na disk.

Prevođenje adresa pri grešci stranice

Korak 6: Rukovalac izuzecima greške stranice pribavlja novu stranicu u glavnu memoriju i ažurira u njoj i odgovarajuću SST.

Korak 7: Rukovalac izuzecima greške stranice vraća upravljanje procesu koji ga je aktivirao, dovodeći do restartovanja instrukcije na kojoj se javio izuzetak. Procesor ponovo šalje istu virtuelnu adresu u MMU. Međutim, sada je virtuelna stranica prisutna u glavnoj memoriji pa se javlja pogodak, pri kome se zahtevani podatak dostavlja procesoru na već opisani način.

Linearna stranična tablica

- * Ovakva linearna stranična tablica ili potpuna stranična tablica, ima V stavki.
- * Za 32-bitni virtuelni adresni prostor sa adresiranjem na nivou bajtova i stranice veličine $L=4\text{ KB}$, $V=2^{32}/2^{12}=2^{20}=1\text{ M}$.
Ako je svaka stavka stranične tablice dužine četiri bajta, cela stranična tablica je veličine 4 MB.

Držanje stranične tablice u memoriji

- * Po jedna ovakva stranična tablica potrebna je za svaki proces u sistemu.
- * Stranične tablice ovih veličina ne mogu se držati u registrima procesora, već u glavnoj memoriji računara.
- * Onda svako obraćanje memoriji za pribavljanje instrukcije ili podatka zahteva po jedno dodatno obraćanje memoriji za prevođenje virtuelne u fizičku adresu.

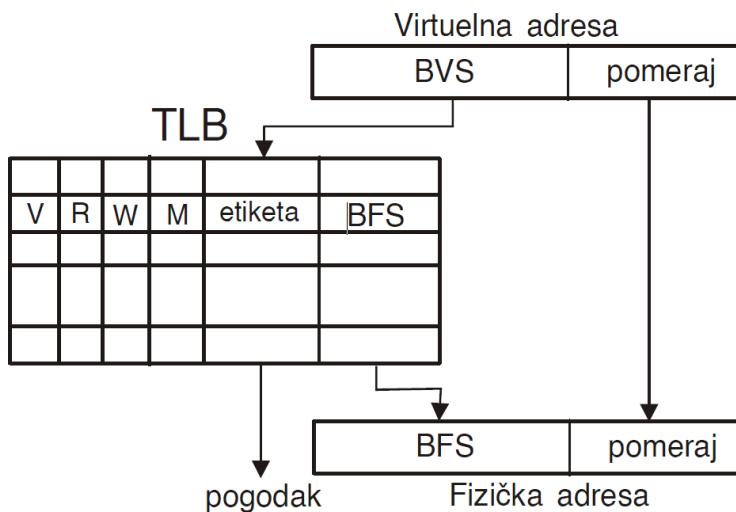
TLB - keš za prevođenje virtuelnih u fizičke adrese

- * Ovaj problem može se delimično rešiti korišćenjem posebne keš memorije u kojoj se drže najaktuelnije stavke stranične tablice.
- * Keš memorija sa ovom namenom poznata je kao **Translation Lookaside Buffer, TLB**.
- * Pri svakom obraćanju memoriji najpre se pristupa TLB-u, sa ciljem da se u njemu nađe broj fizičke stranice u kojoj je prisutna tekuća virtuelna stranica.
- * U slučaju pogotka, nije potrebno obraćanje straničnoj tablici u memoriji, pa se prevođenje adrese znatno ubrzava.

Struktura stavki TLB-a

- * Nastoji se da se obraćanje TLB-u svede na samo jedan taktni ciklus, što upućuje da se TLB implementira kao mala keš memorija sa potpunim asocijativnim preslikavanjem.
- * Stavka TLB-a sadrži kao etiketu broj virtualne stranice na koju se odnosi, broj fizičke stranice u kojoj je prisutna ta virtualna stranica, i indikatore V, R, W i M.

TLB u postupku prevođenja virtualnih adresa



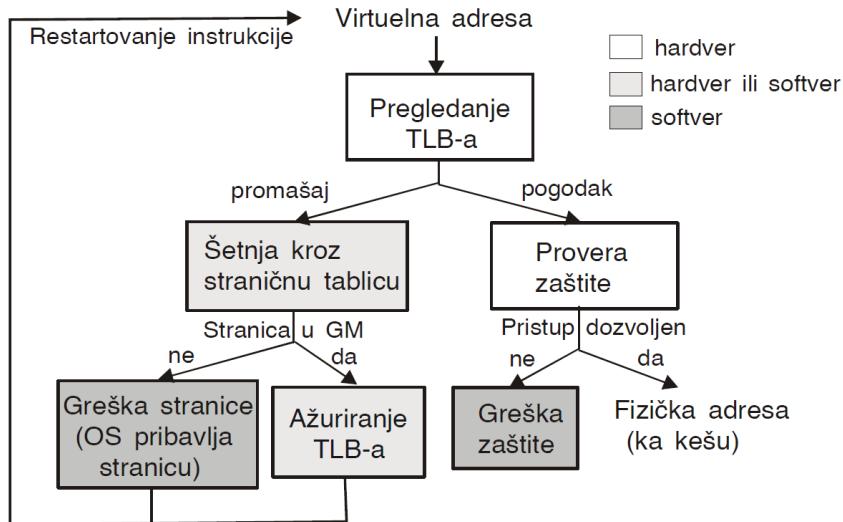
Grešku stranice opslužuje operativni sistem

- * Ako se javi promašaj u TLB-u sledi pristup straničnoj tablici iz koje dobijamo informaciju da li je virtualna stranica prisutna u memoriji i gde, ili da ona tamo nije prisutna.
- * U drugom slučaju javlja se **greška stranice** (page fault), koja dovodi do istoimenog izuzetka, za čije opsluživanje se poziva operativni sistem.
- * Pošto opsluživanje greške stranice traje dugo (nekoliko ms), dolazi do promene konteksta, do okončanja ovog opsluživanja.

Promašaj TLB-a

- * Opslugivanje greške stranice uključuje prenošenje promašene stranice u glavnu memoriju, ažuriranje stranične tablice i TLB-a.
- * Kada se javi promašaj u TLB-u ali se u straničnoj tablici nađe fizička stranica u kojoj je prisutna virtualna stranica, ažurira se TLB zamenom sadržaja polja u izabranoj stavci.
- * U pogodenoj stavci TLB-a ili stranične tablice proveravaju se bitovi zaštite memorije R i W, I ako je pokušan nedozvoljeni pristup memoriji, javlja se izuzetak povrede zaštite memorije.

Aktivnosti pri prevođenju virtuelne adrese uz učešće TLB-a



Sadržaj TLB-a pri promeni konteksta procesa

- * Pošto TLB sadrži stavke stranične tablice tekućeg procesa, promena konteksta procesa zahteva izmenu celokupnog sadržaja TLB-a.
- * Pri napuštanju tekućeg procesa moraju se sve stavke TLB-a proglašiti nevažećim, što se postiže jednostavnim brisanjem indikatora V u svim njegovim stavkama.
- * Alternativno rešenje je da se etiketi u svakoj stavci TLB-a doda identifikator procesa PID, koji bi učestvovao u traženju stavke TLB-a koja određuje zahtevano prevođenje virtuelne adrese.

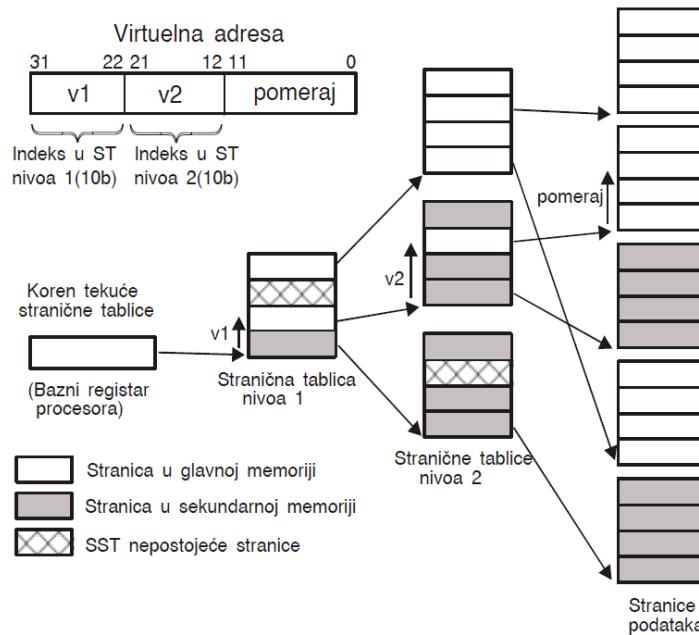
Organizacija TLB-a

- * Broj stavki TLB-a kreće se od 32 do 256.
- * TLB-i manjeg obima implementiraju se kao keš memorije sa potpunim asocijativnim preslikavanjem, a oni većeg obima sa skupnoasocijativnim preslikavanjem i asocijativnošću $2 \div 4$.
- * Mnogi procesori sa posebnim keš memorijama instrukcija i podataka imaju i odvojene TLB-e za prevođenje virtuelnih adresa instrukcija i podataka.

Hijerarhijska stranična tablica

- * Razmotrimo virtualnu memoriju sa 32-bitnim virtualnim adresnim prostorom, adresivom na nivou bajtova, i stranicama veličine 4KB. Stranična tablica ima 1M stavki.
- * Linearna struktura ovakve stranične tablice može se zameniti hijerarhijskom strukturom, naprimer u dva nivoa, na sledeći način.
- * Neka je svaki blok stranične tablice sa po 1K njenih sukcesivnih stavki (nazovimo ga ST-blok) adresiran jednom stavkom pomoćne stranične tablice, nazovimo je stranična tablica nivoa 1 (skr. STN1), koja takođe ima 1K stavki. Sa stawkama dužine 4B, STN1 je veličine jedne stranice, i može se čuvati u glavnoj memoriji.

Hijerarhijska stranična tablica u dva nivoa



Dobre i loše strane...

- * Hijerarhijska stranična tablica može smanjiti zauzeće glavne memorije delovima straničnih tablica procesa.
- * Ako je lokalnost obraćanja procesa memoriji slabo izražena, aktivni ST-blokovi se često menjaju.
- * Ovo povlači potrebu za čestom razmenom prethodno aktivnih ST-blokova i novih aktivnih ST-blokova između glavne memorije i diska, što povećava premašenje operativnog sistema računara.

... hijerarhijske stranične tablice

- * Značajan nedostatak hijerarhijske organizacije stranične tablice je povećan broj obraćanja memoriji pri prevođenju u slučaju promašaja TLB-a, jednak broju nivoa stranične tablice.
- * Novi 64-bitni procesori sa vrlo velikim virtuelnim adresnim prostorom zahtevaju tri i više nivoa stranične tablice, što znatno produžuje prevođenje virtuelnih adresa.

Strategije pribavljanja i zamene podataka u memoriji

- * Podaci se sa diska u glavnu memoriju mogu pribavljati po zahtevu, unapred, ili njihovom kombinacijom.
- * Strategijom pribavljanje po zahtevu virtualna stranica se sa diska prenosi u memoriju posle nastanka greške stranice. Ona nastaje u okviru instrukcije a ne po njenom završetku.
- * Procesor po nastanku ove greške, koju zapaža kroz pojavu izuzetka greške stranice, pamti informacije potrebne za restartovanje instrukcije u kojoj je došlo do greške stranice i poziva rukovoca izuzetka.

Pribavljanje po zahtevu i pribavljanje unapred

- * Po opsluženom izuzetku obnavlja informacije o prekinutoj instrukciji i restartuje njeno izvršenje.
- * Pribavljanje unapred zasniva se na prostornoj lokalnosti: ako se program obratio stranici i , vrlo je verovatno da će se uskoro obratiti i susednim stranicama.... $i-2, i-1, i+1, i+2, \dots$ itd.
- * Pribavljanje ovih stranica unapred može isključiti greške stranica pri skorom obraćanju njima. U tome mogu biti od velike koristi keš memorije diskova

Izbor fizičke stranice za dodelu

- * Izbor fizičke stranice za dodelu vrši algoritam zamene stranica.
- * Sistem vodi evidenciju o nedodeljenim stranicama naprimer u obliku posebne liste nedodeljenih stranica. Kada ova lista nije prazna, onda se za dodelu uzima fizička stranica sa početka liste.
- * Ako se pri tome takva lista formira za svaki proces odvojeno, algoritam zamene je *lokalni*. Ako je lista zajednička za sve proceze, algoritam zamene je *globalni*.

Algoritmi kandidati

- * Za izbor se mogu koristiti algoritmi zamene pominjani kod keš memorija (FIFO, RANDOM, LRU) ali i drugi koji su pogodni baš kod virtuelnih memorija.
- * Ovde se mogu koristiti složeniji algoritmi, koji daju bolje rezultate sa stanovišta smanjenja budućih grešaka stranica, jer sistem ima više vremena za analizu istorije korišćenja ranije dodeljenih fizičkih stranica.

Ipak, LRU algoritam

- * Ipak se i za zamenu stranica kod virtuelnih memorija najčešće koristi LRU algoritam.
- * Kako je ovde broj kandidata za zamenu veliki, striktna implementacija LRU algoritma nije moguća.
- * Zato se ovde koristi jedna aproksimacija LRU algoritma, koja je jednostavna za implementaciju a daje zadovoljavajuće rezultate.

Zaštita memorije od neovlašćenog korišćenja

- * Pri istovremenom izvršavanju više procesa u sistemu, postoji opasnost od neovlašćenog pristupa jednog procesa podacima drugog procesa.
- * Ali postoji i potreba da se omogući deoba nekih programa i/ili podataka u svrhu sinhronizacije procesa i razmene podataka između njih.
- * Pri tome pristupi deljenim programima ili podacima mogu biti ograničeni na određene aktivnosti.
- * Takva zaštita se u straničnoj virtualnoj memoriji lako izvodi na sledeći način.

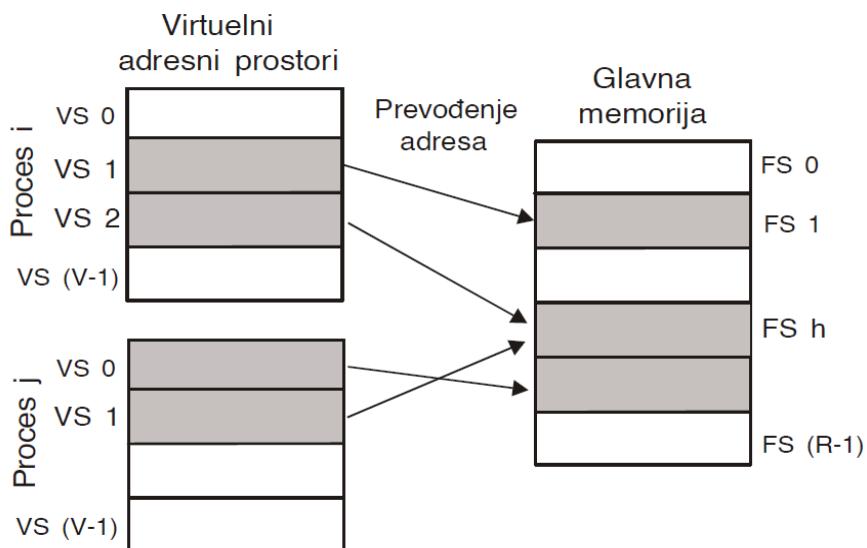
Deljive i privatne stranice

- * Neka procesi **Proces_i** i **Proces_j** imaju kako privatne, tako i deljive stranice podataka.
- * Pri dodeli memorije ovim procesima, deljivim stranicama podataka dodeljuju se iste fizičke stranice.
- * Na sledećem slajdu prikazan je primer dva procesa. **Proces_i** ima privatnu stranicu vs1 i deljivu stranicu vs2. **Proces_j** ima privatnu stranicu vs0 i deljivu stranicu vs1.

Zajednička fizička stranica

- * Deljive stranice su deljive baš između ova dva procesa.
- * Privatnim stranicama oba procesa dodeljuju se različite fizičke stranice, dok se deljivim stranicama dodeljuje ista fizička stranica.
- * Pri dodeli memorije ovim stranicama upisuju se i odgovarajuće vrednosti indikatora R i W u odgovarajućim stavkama stranične tablice odnosno TLB-a.

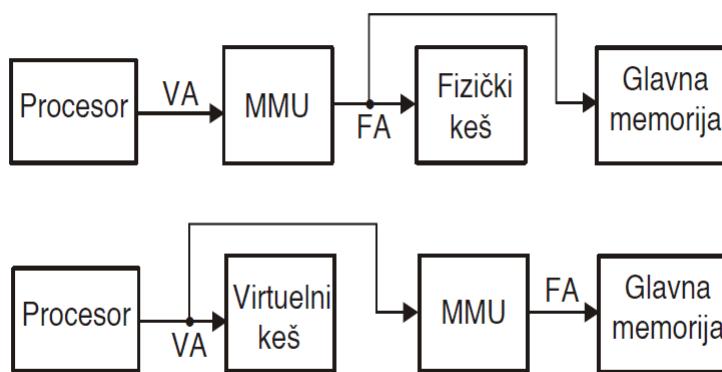
Dodata fizičkih stranica privatnim i deljivim virtuelnim stranicama.



Keš memorije u virtuelnom adresnom prostoru

- * Da li se keš memorija mora adresirati fizičkim adresama?
- * Svako prevodenje virtuelne adrese zahteva vreme, koje u tom slučaju odlaže i pristup kešu, što dovodi do zastoja u radu procesora.
- * Na sledećem slajdu prikazani su procesor, MMU, keš i glavna memorija u dve konfiguracije, koje se razlikuju po tome da li se keš adresira fizičkim adresama (gore) ili virtuelnim adresama (dole).

Adresiranje keša fizičkim adresama (gore) i virtuelnim adresama (dole)



Fizički keš

- * Keš adresiran fizičkim adresama (fizički keš) dolazi u tom lancu iza MMU-a.
- * Veliki nedostatak ovog rešenja je da se obraćanje kešu može započeti tek po okončanju prevođenja virtuelne adrese.
- * U ovom rešenju nema nikakvih ograničenja, a povoljno je i sa stanovišta eventualne direktne komunikacije keša i U-I sistema računara.

Virtuelni keš

- * Adresiranje keša može se započeti virtuelnom adresom, a dovršiti prevedenom fizičkom adresom (virtuelni keš).
- * U tom slučaju se pomeraj u virtuelnoj adresi, koji se neizmenjen prenosi u fizičku adresu , koristi za adresiranje skupa keš blokova i čitanje njihovih etiketa.
- * Po prevođenju virtuelne u fizičku adresu, ona se može uporediti sa pročitanim etiketama blokova u adresiranom skupu keš blokova.

Prevođenje sa zastojem ili bez zastoja

- * Ako se prevođenje adresa obavi sa pogotkom TLB-a, nema zastoja u pristupu kešu.
- * U suprotnom, dolazi do zastoja u pristupu kešu i u radu procesora.
- * Ovde postoji sledeće ograničenje Ako je l dužina polja pomeraja u virtuelnoj adresi, a r i b su dužine polja indeksa i adrese reči u bloku kod keš memorije, onda za ovakvo adresiranje keša mora biti ispunjen uslov

Ograničenje kod virtuelnog keša

- * $l \geq r+b$. Iz ovoga sledi $2^l \geq 2^{r+b}$.
- * $2^l = L$ je veličina stranice podataka, a $2^{r+b} = 2^r \times 2^b = S \times B$.
- * Onda je $L \geq S \times B$.
- * Ako obe strane nejednakosti pomnožimo sa A , dobićemo

$$L \times A \geq A \times S \times B = C$$

(A , S i B su asocijativnost, broj skupova keš blokova i dužina blokova keš memorije respektivno. C je kapacitet keša u bajtovima.)

- * Sledi ograničenje

$$C \leq L \times A$$

Prednosti i nedostaci virtuelnog keša

- * Prednost virtuelnog keša je pristup bez čekanja na okončanje prevodenja virtuelne adrese.
- * Nedostatak je nemogućnost direktnе komunikacije keša i U-I sistema.
- * Između keš memorija i virtuelnih memorija postoje sličnosti i razlike koje su navedene u tabeli na sledećem slajdu.
- * Veličine primarnih keš memorija su 4 KB÷64 KB, sekundarnih keš memorija su 128 KB÷4 MB, a glavne memorije 256 MB ÷ 16 GB.

Tipičan opseg parametara primarne i sekundarne keš memorije i virtuelne memorije

Parametar	Primarna keš memorija	Sekundarna keš memorija	Virtuelna memorija
Veličina bloka (stranice)	4÷32 B	32÷256 B	4096÷16.384 B
Vreme pogotka (TC)	1÷3	8÷15	50÷200
Promašajna kazna (TC)	10÷40	100÷500	1.000.000÷10.000.000
Faktor promašaja	0,5 ÷ 20 %	15÷30 %	0,00001 ÷ 0,001 %
Niži nivo memorije	sekundarni keš	glavna memorija	diskovi
Smeštanje blokova podataka	direktno ili skupno-asoc.	direktno ili skupno-asoc.	potpuno asocijativno
Zamena blokova	LRU	Random ili LRU	Aproximacija LRU