5. UPRAVLJANJE MEMORIJOM

U toku razvoja računarskih sistema iznos glavne memorije instalirane u računaru se povećavao, ali se takođe i obim programa u poređenju sa dostupnom instaliranom memorijom sve više povećavao. Prvi pristup da se premosti ograničenje obima memorije zasniva se na primeni tehnike preklapanja (overlays). Smisao se sastoji u deljenju programa na nekoliko delova. Jedan od ovih delova je uvek prisutan u memoriji i upravlja punjenjem drugih delova sekundarne memorije (disk, traka, itd.) u glavnu memoriju. Ostali delovi se mogu puniti u glavnoj memoriji, pri čemu oni koriste istu memorijsku oblast, kao i prethodno korišćeni delovi, pa zbog toga kažemo da dolazi do preklapanja. Kada sa računarskim sistemom istovremeno radi veći broj korisnika smanjuje se deo glavne memorije koji je dodeljen svakom korisniku, a sa druge strane neophodno je uvesti neke mehanizme zaštite, koji će štititi (zaokruživati kao celinu) aktivnosti jednog korisnika od drugog.

Imajući u vidu prethodno pomenuto, sistem za zaštitu memorije treba ispuni sledeće funkcije:

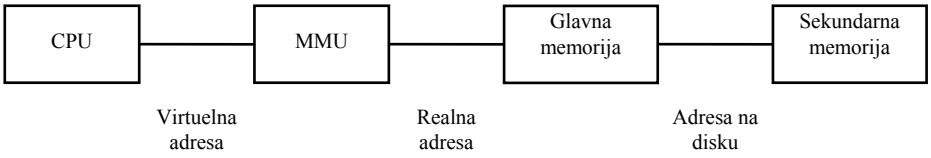
- **Zaštita** (protection) - ako nekoliko procesa deli glavnu memoriju (GM), procesima se ne sme dozvoliti da menjaju lokacije koje nisu njima dodeljene. Ovim se ostvaruje određeni oblik privatnosti procesa. Zaštita se može primeniti na dva nivoa: na prvom nivou, zaštita obezbeđuje da se lokacijama ne može pristupati od strane neautorizovanih korisnika; na drugom nivou, autorizovani pristupi su dalje ograničeni specifičnim pravima pristupa (samo-čitanje, samo-upis,...).
- **Deoba** (sharing) - često je potrebno da korisnici budu u mogućnosti da dele i ažuriraju informaciju - na primer, sistem baze podataka. Šta više, nije potrebno raspolagati sa nekoliko kopija jedinstvene "reentrant" rutine, ako je svim korisnicima dozvoljeno da koriste ovu rutinu. No, zaštita zbog nenamernih promena mora biti prisutna, čime se upravlja pristupima deljivim delovima meorije.
- **Relokacija** - kod multiprogramskog režima rada zahteva se da po nekoliko procesa istovremeno bude prisutno u memoriji. Proces ne zna unapred gde će biti napunjen, tako da nije poželjno koristiti apsolutne memorijske adrese.
- **Fizička organizacija memorije** - visoka cena brze GM-e obično nameće potrebu da se GM proširi sa jevtinijom i sporijom sekundarnom memorijom (SM) koja kao medijum za zapis koristi diskove, trake i dr. Ovakva memorijska hijerarhija, koja je prisutna kod svih savremenih računara, nameće potrebu za upravljanje tokom podataka između pomenuta dva nivoa fizičkih medijuma za zapis.
- **Logička organizacija meorije** - najveći broj GM-a je organizovan kao linearni prostor, sa sekvencijalnim adresama od 0 do odgovarajuće maksimalne adrese. Ovakav pristup u velikoj meri odražava (ili bi trebalo da odražava) način pisanja programa, a sastoji se u korišćenju logičkih programa i struktura podataka kao što su moduli, procedure i polja. Ako sistem za upravljanje memorijom (SMM) obezbeđuje nekoliko adresnih prostora, svaka logička struktura predstavljaće nezavisnu celinu (koju obično nazivamo segment). Ovakav pristup ima nekoliko prednosti: segmenti se mogu kompilovati i puniti nezavisno (linkovanje segmenata se izvodi u toku izvršenja programa), svaki segment može imati svoja sopstvena prava pristupa (samo-čitanje, čitanje-i-izvršenje, i dr.). Najjednostavnija segmentaciona šema obično koristi dva segmenta, jedan je programski (može se čitati i izvršavati) a drugi je segment podataka (moguć je upis i čitanje). Obično složenija rešenja koriste izdvojene segmente za svaku logičku strukturu.

5.1. Koncept virtuelne memorije

Na koncept virtuelne memorije ukazaćemo sa dva aspekta: Prvi se odnosi na translaciju adresa (određuje kako se adrese generisane od strane procesa (koji se izvršava) prevode u memorijske adrese) a drugi na radni model (na kojim se politikama zasniva MM).

Translacija adrese

Na slici 5.1 prikazana je adresna hijerarhija računara. Adrese koje generiše proces (koji se izvršava) se zovu *virtuelne adrese* (VA), a one se mogu razlikovati od adresa pomoću kojih se vrši pristup GM-i.



Sl. 5.1. Adresna hijerarhija.

Ukupan adresni prostor koji je dostupan za izvršenje nekom procesu se zove *virtuelni adresni prostor* (VAP). Adrese koje se koriste za pristup GM-i se zovu *realne adrese* (RA), zbog toga za svaku RA postoji realna odgovarajuća lokacija. Kada ne postoji relokacija, virtuelne i realne adrese su identične. Ako postoji relokacija, virtuelne adrese je potrebno prevesti u realne, a taj proces se zove *translacija adresa*. Translacija se izvodi hardverskom jedinicom koja se zove MMU (Memory Management Unit).

Zbog ograničenog obima GM-e, za objekat (npr. podatak ili program) se može desiti da ne bude prisutan u GM-i već u SM-i. Ovaj objekat se mora preneti (kada se zahteva njegovo korišćenje) iz SM-e u GM-u, npr. specificiranjem adrese objekta na disku. Kako se ava aktivnost izvodi transparentno u odnosu na proces koji se izvršava, kažemo da se radi o *virtuelnom memorijskom sistemu* (VMS). Često se termini *logičke adrese* (LA) i *logički adresni prostor* (LAP) koriste za sisteme koji ne podržavaju virtuelni memorijski sistem, dok se termini virtuelna adresa i virtuelni adresni prostor koriste kod sistema koji koriste virtuelni memorijski sistem. U daljem izlaganju koristićemo uglavnom termine virtuelna adresa i virtuelni adresni prostor.

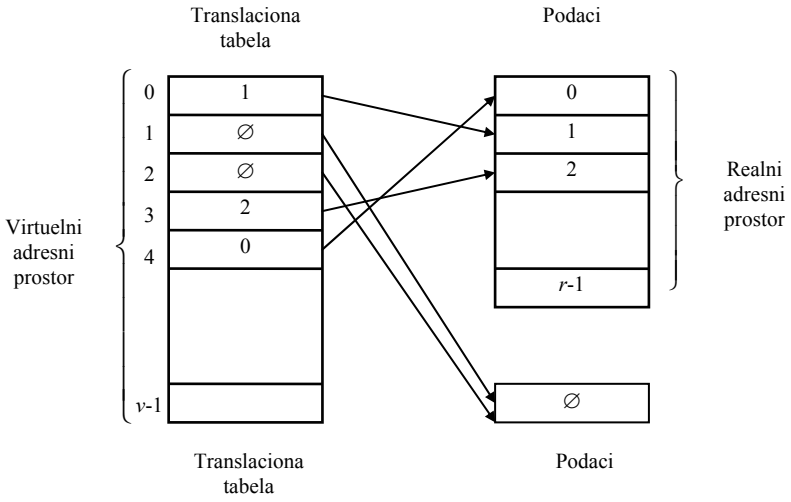
Kao što smo već ukazali, relokacija je potrebna kada je neophodno obaviti neki vid translacije adresa. Ovo se može izvesti u nekoliko koraka u toku generisanja i/ili izvršenja programa. Kada je alocirana memorijska oblast poznata u toku procesa punjenja, loader može menjati virtuelna adrese u realne, tako da se realne adrese generišu u toku izvršenja programa; ovo se zove *statička relokacija*. Kada alocirana memorijska oblast nije poznata u toku punjenja, proces koji se izvršava koristiće virtuelne adrese, koje se relociraju u trenutku izvršenja, a to je poznato kao *dinamička relokacija*. Pošto se dinamička relokacija obavlja u trenutku izvršenja za svaku adresu, neophodno je koristiti specijalan hardver tipa MMU, koji će ovu translaciju obaviti veoma brzo.

Formalno, funkcija MMU-a se može opisati na sledeći način: Skup virtuelnih adresa V={0,1,...,v-1} se preslikava u skup alociranih realnih memorijskih adresa R={0,1,...,r-1} korišćenjem funkcije $f : V \rightarrow R$. Funkcija f se definiše kao:

$f(x) = y$	ako je podatak na virtuelnoj adresi x u R na realnoj adresi y
$= \varnothing$	ako podatak na virtuelnoj adresi x ne postoji u R

Ako je $f(x)=0$, javlja se greška zbog promašaja a podatak kome se obraćamo treba dobaviti u GM-u iz SM-e koristeći se rutinom za obradu greške.

Pojednostavljeni primer MMU-a prikazan je na slici 5.2. virtuelna adresa 4 se translira u realnu adresu 0 ($f(4)=0$), dok se virtuelna adresa 3 translira u realnu adresu 2 ($f(3)=2$). Pristup virtuelnoj adresi 1 generiše grešku usled promašaja jer je $f(1)=0$. Virtuelne adrese se obično dele na dva dela: broj-virtuelnog-bloka i ofset. Broj virtuelnog bloka se preslikava u realnu adresu bloka, dok se ofset koristi za selekciju reči unutar ovog bloka (termine ofset (pomeraj) i razmeštaj (displacement) koristićemo ravnopravno). Rad sa blokovima ima prednost koja se ogleda u tome da se obim translacione tabele smanjuje u značajnoj meri. Ako se 1MB virtuelnog prostora translira bez korišćenja blokova (obim bloka je 1), translaciona tabela treba da ima 1M ulaza, ali ako su blokovi obima 4096B broj ulaza se smanjuje na 256. Izbor obima bloka je često uslovljen vremenom pristupa i vremenom transfera iz/ka SM-i, kao i obimom GM-e.



Sl. 5.2. Primer MMU.

5.2. Radni model

Ako se sagleda ponašanje programa, a posebno sekvenca obraćanja memoriji, ustanoviće se da ova sekvenca nije u potpunosti slučajna, tj. na neki način ona je predvidljiva. Ovo ukazuje na *princip lokalnosti*, koji sugeriše da u toku određenog vremenskog intervala program ima tendenciju da grupiše obraćanja memoriji na mali deo memorije u odnosu na ukupni raspoloživi prostor. U okviru ove lokalnosti razlikujemo dve komponente: vremensku i prostornu lokalnost. *Vremenska lokalnost* ukazuje na lokalnost u vremenu u bliskoj budućnosti. Ona ukazuje da će se program obraćati onoj programskoj sekvenci i onim podacima kojima se obraćao u bliskoj prošlosti. Ova lokalnost može biti rezultat obraćanja instrukcijama (kod programskih petlji) ili obraćanja podacima (kod zapisa ili magacina). *Prostorna lokalnost* ukazuje na lokalnost u prostoru: u bliskoj budućnosti, program še se obraćati onim programima i podacima čije su adrese bliske onima kojima se zadnji put obraćao. Ovo je uslovljeno sekvencijalnim izvršenjem programskog koda i strukturom podataka kao što su polja. Naravno, različiti programi pokazuju različiti stepen lokalnosti, ali vremenska i prostorna lokalnost mogu egzistirati istovremeno. Stepen prostorne lokalnosti je važan faktor kod određivanja obima bloka.

U toku nekog vremenskog intervala program se obraća samo podskupu njegovog virtuelnog adresnog prostora (vremenska lokalnost). Skup blokova u trenutku t, kojima se vršilo obraćanje u okviru zadnjih h obraćanja, zove se *radni skup* $W(t,h)$. Naravno, h se može izraziti u vremenskim jedinicama umesto u obraćanjima memoriji.

5.3. Zaštita i deoba

Veoma je važno razumeti tačno značenje termina zaštita i deoba. U tom smislu, zaštita znači da nije, u toku izvršenja nekog procesa nad određenim onjektom, dozvoljen pristup radi čitanja, upisa ili izvršenja, a drugim procesima su dozvoljene operacije čitanje, upis ili izvršenje. Deoba ukazuje na mogućnost pristupa objektu koji pripada jednom procesu od strane drugih procesa. Zaštita i deoba su dve važne (često konfliktne) osobine kod upravljanja memorijom. Da bi smo razumeli zašto su to konfliktne osobine, razmatramo šta će se desiti u idealnoj situaciji jednoj od ovih osobina. Ako se zaštita podržava u optimalnom smislu, procesu će biti dozvoljeno da pristupa isključivo svojim a ne i drugim objektima. Nasuprot tome, najidealnija situacija deobe se javlja kada bilo koji proces neće "kvariti" objekte drugih procesa. Na nesreću, ova druga situacija nije realna, dok je kod prvog slučaja znatno otežan rad. Zbog toga, neophodno je naći kompromis između ove dve krajnosti, pri čemu će kontrolisana deoba biti moguća. Pod pojmom kontrolisanje se podrazumeva da jedan proces može dozvoliti nekim procesima pristup pojedin ili svim njegovim objektima, dok drugi moguma to mogu zabraniti. Ako je pristup dozvoljen on može biti tipa: čitanje, upis, ili izvršenje.

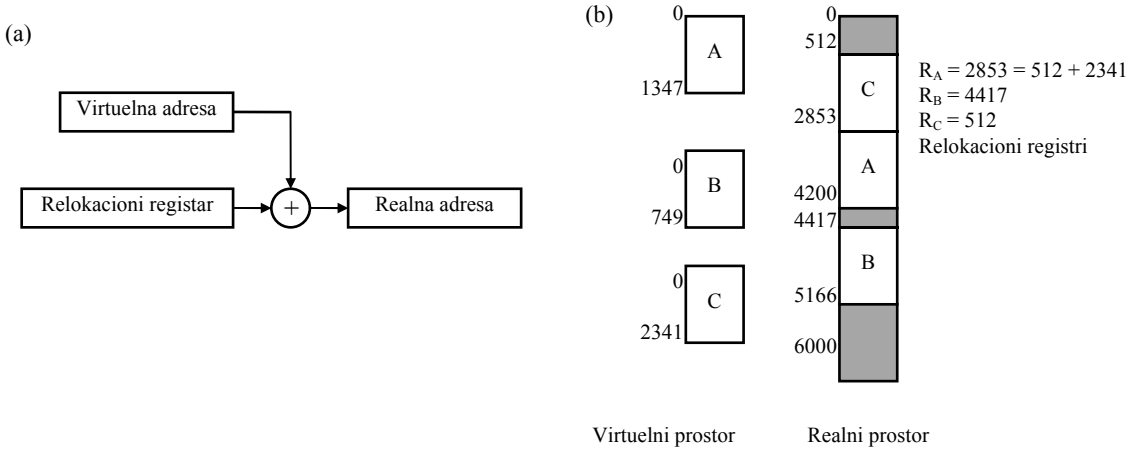
5.4. Mehanizmi adresne translacije

Translacija adresa koja je potrebna kada se obavlja dinamička relokacija, može se izvesti na nekoliko načina. Standardni mehanizmi translacije adresa su: relokacija straničenje, segmentacija i segmentacija sa straničenjem.

5.4.1. Relokacija

Kod starijih računara, cilj upravljanja memorijom kod dinamičke relokacije često se realizovao pomoću *relokacionih registara*. Proces koristi virtuelni adresni prostor koji je manji ili jednak realnom adresnom prostoru, pri čemu se počinje sa VA=0. Ako se deo virtuelnog prostora puni bilo gde u memoriji, tada se realna adresa koja odgovara prvoj lokaciji smešta u relokacioni registar. Nakon toga, u toku svakog obraćanja memoriji, virtuelna adresa se dodaje relokacionom registru pa se na ovaj način dobija realna adresa (slika 5.3). Na ovaj način nekoliko procesa može istovremeno da bude prisutno u memoriji, pri čemu svaki počinje sa svoje VA=0. Svaki put kada procesor komutira na drugi proces, ažurira se relokacioni registar. Relokacija procesa sastoji se u ažuriranju relokacionog registra. Na slici 5.3b prikazana je situacija kada u memoriji postoje tri procesa, zajedno sa svojim virtuelnim prostorima. Da bi se smanjio obim i vreme potrebno za relokaciono sabiranje, LS k bitova relokacionog registra treba inhibirati čime se omogućava punjenje programa samo na onim lokacijama koje su umnožak od 2^k.

Na žalost, ova šema ne nudi zaštitu: proces C može da promeni svoju virtuelnu lokaciju 2500, tako da proces A može da krahira ako u sledećem trenutku izvršava naredbu sa svoje VA=159. Ovo se može izbeći ako se upari relokacioni registar sa odgovarajućim *registrom zaštite* u kome se čuva realna adresa zadnje lokacije koja je dodeljena procesu. Ako se proces C (sa slike 5.3b) izvršavao, u njegov registar zaštite biće upisana vrednost 2853. Kod svakog obraćanja memoriji, virtuelna adresa se dodaje relokacionom registru, a rezultat se proverava sa vrednošću upisanom u registru zaštite. Ako provera ukaže na nekoreknost u radu ne vrši se pristup memoriji, a signalizira se greška.



Sl. 5.3. Dobijanje realne adrese na osnovu virtuelne.

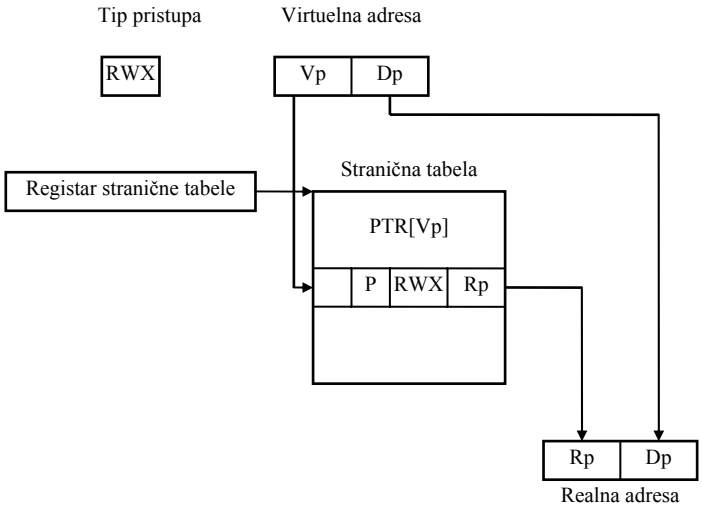
5.4.2. Straničenje

Kod šeme relokacija-zaštita javljaju se sledeći problemi:

- svakom procesu se mora dodeliti kontinualni prostor realne memorije,
- virtuelnom adresnom prostoru nije dozvoljeno da premaši iznos memorije koja mu se dodeljuje.

Da bi se rešili ovi problemi koristi se *straničenje*. Straničenje deli virtuelni adresni prostor na blokove, koji se nazivaju *stranice*, pri čemu je svaka stranica istog obima. Glavna memorija, kojoj se pristupa preko realnog adresnog prostora, deli se na blokove istog obima koji se zovu *okviri stranica*. Translacija virtuelne adrese u realnu adresu obavlja se pomoću tabele transliranja koja se zove *tabela straničenja*. Svaki proces ima sopstvenu tabelu straničenja, lociranu bilo gde u memoriji koja se adresira preko *registra tabele straničenja* (PTR) (kao što je prikazano na slici 5.4). Nakon komutacije procesa, PTR se postavlja da ukaže na tabelu straničenja procesa koji se aktivira.

Virtuelni adresni prostor se deli na blokove čija veličina odgovara stranici. Virtuelnu adresu čini virtuelni broj stranice, Vp, i razmeštaj u okviru stranice, Dp. Ulaz u tabelu stranice (PTE) čini realna adresa stranice, Rp, prava pristupa koje proces ima na toj stranici i marker prisutnosti, P, koji ukazuje da li se ta stranica nalazi u glavnoj memoriji. Translacija adrese se obavlja na sledeći način: Vp se koristi kao indeks u stranici, na koju ukazuje PTR. Zatim se pristupa ulazu tabele stranice PTR[Vp], ali se prethodno proveravaju prava pristupa (čitanje, upis ili izvršenje (RWX)) nad operacijom koja se obavlja. Polje Rp se izdvaja od PTE-a i pripaja se polju Dp koje pripada virtuelnoj adresi, pa se na taj način dobija realna adresa.



Sl. 5.4. Prevođenje virtuelne u realnu adresu.

5.4.3. Segmentacija

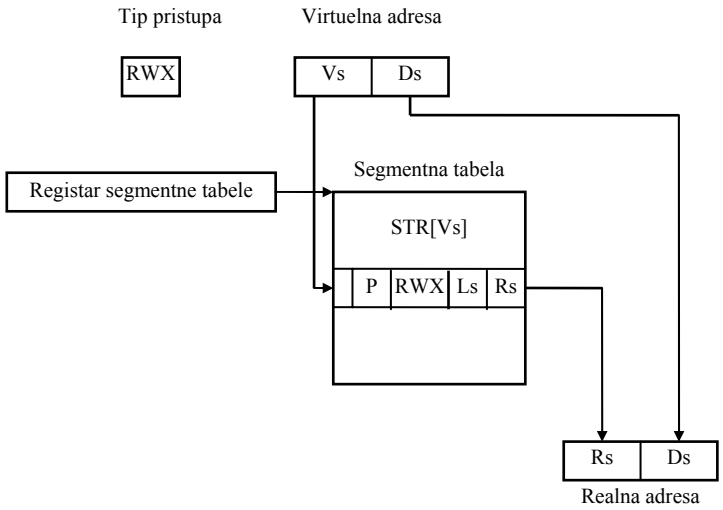
Da bi se postigla logička organizacija virtuelnog adresnog prostora koji će podsećati na to kako korisnik (treba da) vidi memoriju, uvodi se koncept segmentacije. Deljenje programa na delove jednakih dužina nije prirodno. Uobičajeno, blok-strukturni jezici, kao što su Pascal i C, omogućavaju kreiranje logičkih celina kao što su procedure i moduli različitih dužina. Sa ovakvim logičkim celinama može se manipulisati od strane mehanizma za upravljanje memorijom, memorisanjem ovih celina u segmentima. Nasuprot stranici, segment se može povećavati i smanjivati (u toku njegovog života), kao što je slučaj sa magacinskim segmentom.

Jednostavnu implementaciju uobičajeno čini nekoliko parova relokaciono-zaštitinih registara, pri čemu se po jedan par dodeljuje svakom segmentu. Nedostatak ovog pristupa se sastoji u tome što broj segmenta mora biti mali (i fiksan) zbog toga što je broj relokaciono-zaštitnih registara obično ograničen.

Savršenija implementacija segmentacije omogućava da se svakom procesu dodeli veliki broj segmenata. Na slici 5.5 prikazan je način realizacije. virtuelna adresa se razlaže na segmentni broj Vs i razmeštaj u okviru segmenta Ds. Koristeći *segmentnu tabelu*, na koju se ukazuje *registrom segmentne tabele* (STR), segmentnim brojem Vs selektuje se segmentni tabelarni ulaz (STE), koji se zove *deskriptor segmenta*. Izabrani STE sadrži: realnu segmentnu adresu (Rs), markere zaštite i upravljanja, slične onima kao kod PTE-a. Pošto prostor dodeljen segmentu može biti promenljive dužine, u STE mora da se čuva i informacija o dužini tog segmenta Ls. Proces čini skup segmenata, a svaki segment ima svoja prava pristupa.

I pored toga što su mehanizmi za translaciju adresa, prikazani na slici 5.4 i slici 5.5, na prvi pogled isti (oba podržavaju virtuelnu memoriju), koncepti straničenja i segmentacije su u suštini veoma različiti.

- Straničenje deli virtuelni adresni prostor, fizički adresni prostor i prostor sekundarne memorije na blokove (nazvani stranice) jednakih dužina. Ovo omogućava da se kontinualni virtuelni adresni prostor dodeljen procesu rasipa (ne mora biti kontinualno raspoređen) u realnom adresnom prostoru i sekundarnoj memoriji. Čak se stranica, kao pojam, odnosi na memoriju a ne na logičke objekte koji su vidljivi na programskom nivou. Straničenje je, zbog toga, transparentno procesu kod koga je to straničenje izvršeno.
- Segmentacija deli virtuelni adresni prostor na blokove (segmente) koji direktno odgovaraju objektima na programskom nivou. Zbog toga segment nema fiksnu dužinu, pa se čak i obim segmenta može menjati u toku izvršenja programa. Zaštita i deoba su, zbog toga, mogući na objektnom nivou, i vidljivi su procesi kod koga je izvršena segmentacija.



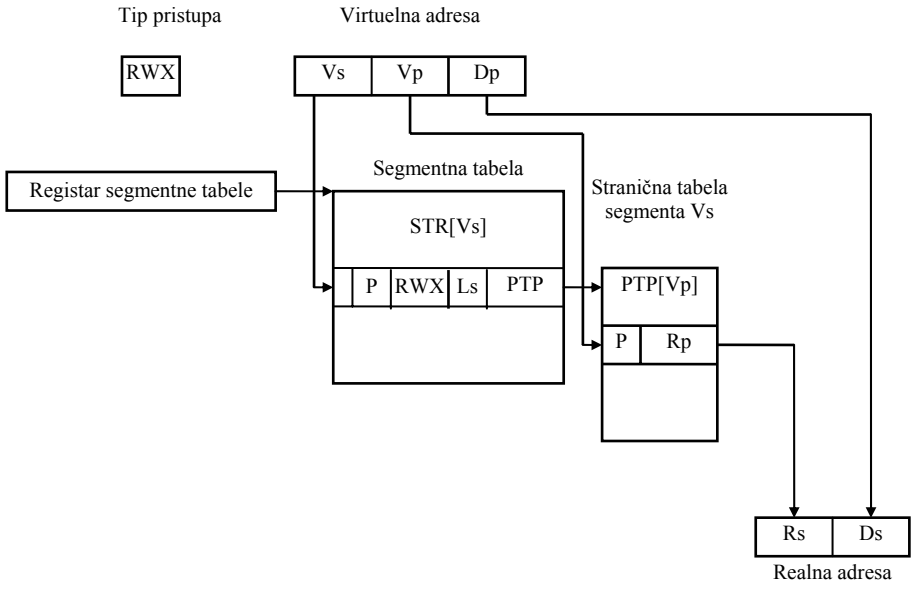
Sl. 5.5. Adresna translacija kod sistema koji se zasniva na segmentaciji.

5.4.4. Segmentacija sa straničenjem

Često se, radi dobijanja boljih performansi, vrši kombinacija mehanizma straničenja i segmentacije. Segmenti se dele na stranice (jednakog obima), a svaka virtuelna adresa se razlaže na tri dela (Vs, Vp, Dp) gde: Vs označava broj segmenta, Vp broj stranice u okviru segmenta, a Dp je razmeštaj u okviru stranice. Na slici 5.5 prikazan je mehanizam adresne translacije kod koje STR ukazuje na segmentnu tabelu. Koristeći Vs kao prvi indeks, izdvaja se STE koji ukazuje na tabelu stranice tog segmenta. Zatim se koristi Vp da bi se odredila adresa realne stranice Rp u tabeli stranica. Konačno se koristi Dp da bi se odredio razmeštaj u okviru stranice.

Kada je proces aktivan, njegova segmentna tabela mora biti prisutna u glavnoj memoriji. Obim tabele se reducira na taj način što je neophodno čuvati samo tabele stranica datih segmenata koji pripadaju radnom skupu procesa, koji su prisutni u glavnoj memoriji. Ovo se ostvaruje markerom prisustva u STE-u.

Na slici 5.6, prava segmenata i obim segmenata se specificiraju pomoću STE-ova, tako da je pristup regulisan na nivou segmenta. Sve stranice segmenta, zbog toga imaju ista prava pristupa. Ova šema, kod koje segmentacija ima dominantni faktor zove se *segmentacija sa straničenjem*.



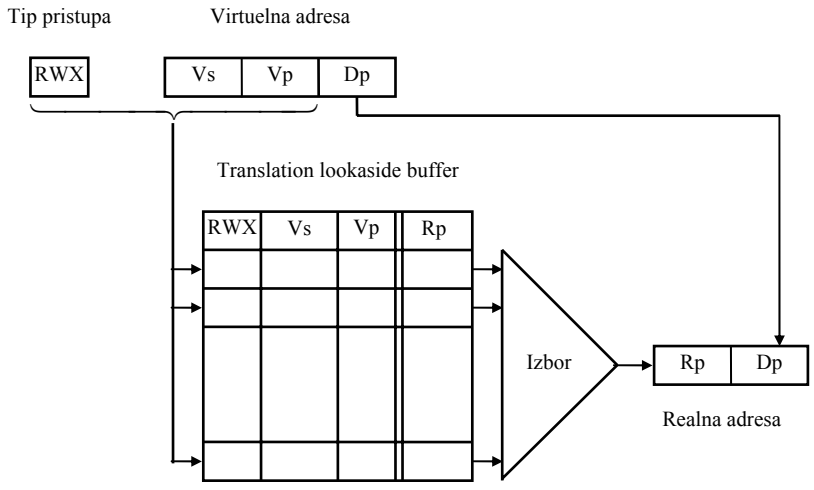
Sl. 5.6. Prevođenje adresa kod segmentacije sa straničenjem.

5.4.5. Mehanizmi ubrzanja adresne translacije

Mehanizam adresne translacije o kome smo govorili do sada koristi direktno preslikavanje. Kod direktnog preslikavanja, broj ulaza u translacionoj tabeli (TTB) jednak je broju blokova u virtuelnom adresnom prostoru. Pri tome se u toku svakog memorijskog pristupa ispituje TTB što unosi kašnjenje od jednog memorijskog ciklusa (ako se koriste dva ili veći broj nivoa translacije). Kako se memorijski ciklusi obavljaju često, ovaj oblik adresne translacije smanjiće značajno performanse sistema. Kod sistema sa relativno malim virtuelnim adresnim prostorom - na primer 2^{24} bajtova - i blokovima relativno velikog obima - na primer 2^{12} bajtova - potrebna je samo 2^{12} ulaza u tabelu. Ova informacija se može čuvati u brzoj memoriji (obično ne glavnoj) sa ciljem da se ubrza operacija adresne translacije. Za sisteme sa velikim virtuelnim adresnim prostorom ovo nije dobro rešenje, jer TTB postaje suviše velika, a shodno tome suviše spora (zbog vremena potrebnog za adresno dekodiranje) i skupa (zbog kapaciteta tabela). Nekoliko akceleracionih mehanizama postoji za proces adresne translacije, a to su: TLB (Translation Lookaside Buffers), IPT (Inverted Page Tables) i MVS (Multiple Virtual Spaces).

TLB (Translation Lookaside Buffers)

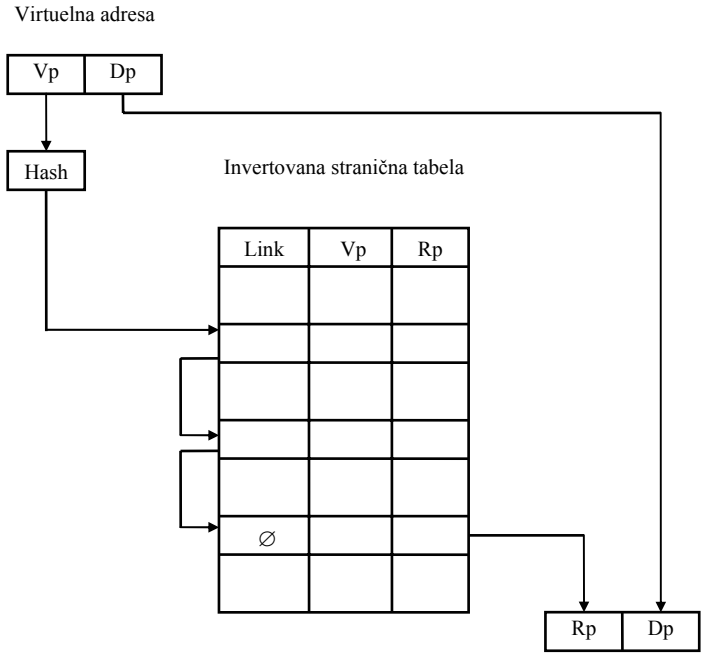
Mehanizam ubrzanja adresne translacije koji koriste TLB zasniva se na principu vremenske lokalnosti. TLB se može posmatrati kao hardverski keš koji se koristi da čuva najskorije korišćenu virtuelno-u-realnu adresnu translaciju). Za čuvanje N najskorije transliranih adresa koristi se asocijativna memorija. Na slici 5.7 prikazan je TLB za sistem sa slike 5.6. Svaki TLB ulaz se može posmatrati kao kombinacija STE/PTE sa slike 5.6. TLB ulaz sadrži: segmentni broj, Vs, broj stranice, Vp, markere koji ukazuju na prava pristupa, i realnu adresu stranice, Rp. Adresna translacija se obavlja na sledeći način: vrši se paralelno poređenje svih N ulaza u TLB-u (poređenje se odnosi na tip pristupa koji se obavlja, virtuelni segmentni broj Vs i virtuelni broj stranice Vp, virtuelnog prostora). Kada dođe do uparivanja TLB ulaza, realna adresa stranice Rp uparenog ulaza se pridružuje Dp-u, u cilju formiranja realne adrese. Memorija koja se koristi za čuvanje TLB ulaza se takođe zove CAM (Contents Addressable Memory) ili asocijativna memorija (AM), jer se željeni izlaz Rp nalazi pridruživanjem njegovog sadržaja sa ulaznim podatkom. kada ne postoji uparivanje, kažemo da se javila greška ili promašaj, koristi se adresno translacioni mehanizam sa slike 5.6 (segmentna tabela i neke tabele stranica nalaze se u glavnoj memoriji). Rezultat ove translacije se koriste za ažuriranje TLB-a.



Sl. 5.7. TLB za segmentaciju sa straničenjem.

Invertovana stranična tabela (Inverted Page Table - IPT)

Kod IPT postoji po jedan ulaz za svaku realnu memorijsku stranicu, umesto po jedan ulaz za svaku virtuelnu stranicu. Ovaj pristup ima prednost koja se ogleda u tome da se zauzima od strane tabele samo fiksirani (i ograničeni) deo memorije, nezavisno od oblika korišćenog virtuelnog adresnog prostora. Ova tabela se može implementirati brzom memorijom, pa se na taj način ubrzava operacija adresne translacije.



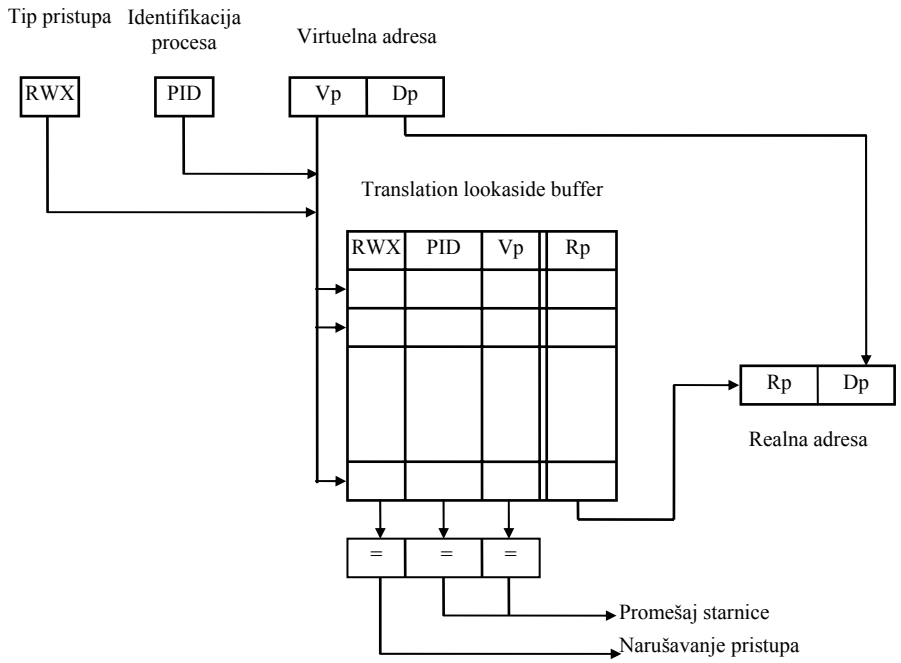
Sl. 5.8. Prevođenje adrese pomoću IPT.

Svaki ulaz IPT-a (slika 5.8) čini: polje za vezu (koje formira lanac IPT ulaza sa istom "hash" vrednošću), Vp polje, i Rp polje. Adresna translacija se obavlja metodom "hashing" (to je oblik skremblovanja) broja stranice sa virtuelnom adresom tako da se ona može koristiti sa virtuelnom adresom ulaza u IPT-u. Sadržaji Vp polja tog ulaza se kompariraju sa Vp brojem od virtuelne adrese. Nakon uparivanja, dobijena Rp vrednost se koristi za specifikaciju realne adrese stranice. Kod promašaja, polje za vezu (link) se koristi za proveru sledećeg IPT ulaza u lancu IPT ulaza sa nekom "hash" vrednošću. Ako uparivanje ne postoji, stranica nije prisutna u memoriji, pa se javlja greška stranice. I pored toga što dugi lanac ukazuje na dugo translaciono vreme, u najvećem broju slučajeva eksperimentalno je utvrđeno da je srednja dužina lanca između jedan i dva. Mehanizam adresne translacije se može implementirati hardverski, koristeći standardne memorije sa brzim pristupom u kojima se čuva IPT.

Višestruki virtuelni prostori (Multiple Virtual Spaces - MVS)

Opisani mehanizmi translacije adresa (TLB i IPT) omogućavaju translaciju samo jednog virtuelnog adresnog prostora. Nakon komutacije procesa, translacione tabele je neophodno brisati i ponovo puniti sa translacionom informacijom koja se odnosi na novo aktivirani proces. Jedan od načina da se štiti brisanje translacionih tabela (TLB ili IPT) je da se prošire ulazi poljem koje sadrži informaciju o identifikaciji procesa (PTD). Na slici 5.9 prikazana je TLB za virtuelni memorijski sistem koji koristi straničenje. Šema koristi asocijativno pretraživanje pomoću kojeg se zahteva i uparivanje sa PID brojem.

Kod velike TLB i česte procesne komutacije, PID koncept može poboljšati brzinu adresne translacije jer veliki broj ulaza reaktiviranih procesa može još biti prisutan u TLB, pa se na taj način stopa premašaja značajno smanjuje.



Sl. 5.9. TLB kod virtuelnog memorijskog sistema sa straničenjem.

5.4.6. Motorola MC68851 PMMU

MC68851 PMMU (Paged Memory Management Unit) ugrađuje se u sisteme zasnovane na MC68020 sa ciljem da se ostvari preslikavanje memorije i zaštita memorije. Upravljan pod kontrolom sistemskog softvera MC68851 implementira šemu memorijskog preslikavanja na taj način što obavlja translaciju logičkih adresa (pripadaju programu) u fizičke adrese (memorijske adrese). Mehanizam zaštite PMMU-a omogućava da izabrane oblasti glavne memorije budu zaštićene od pristupa programa koji se izvršavaju. Ovi pristupi se mogu ograničiti koristeći se korisničkim i supervizorskim načinom rada. Šta više, važeći adresni opseg koji je dodeljen programu može se strogo poveriti unapred određenom opsegu. MC68851 takođe podržava instrukcije Breakpoint (BKPT) i Call Module (CALLM) mikroporcesora MC68020. Tipičan način povezivanja PMMU i MC68020 prikazan je na slici 5.10.

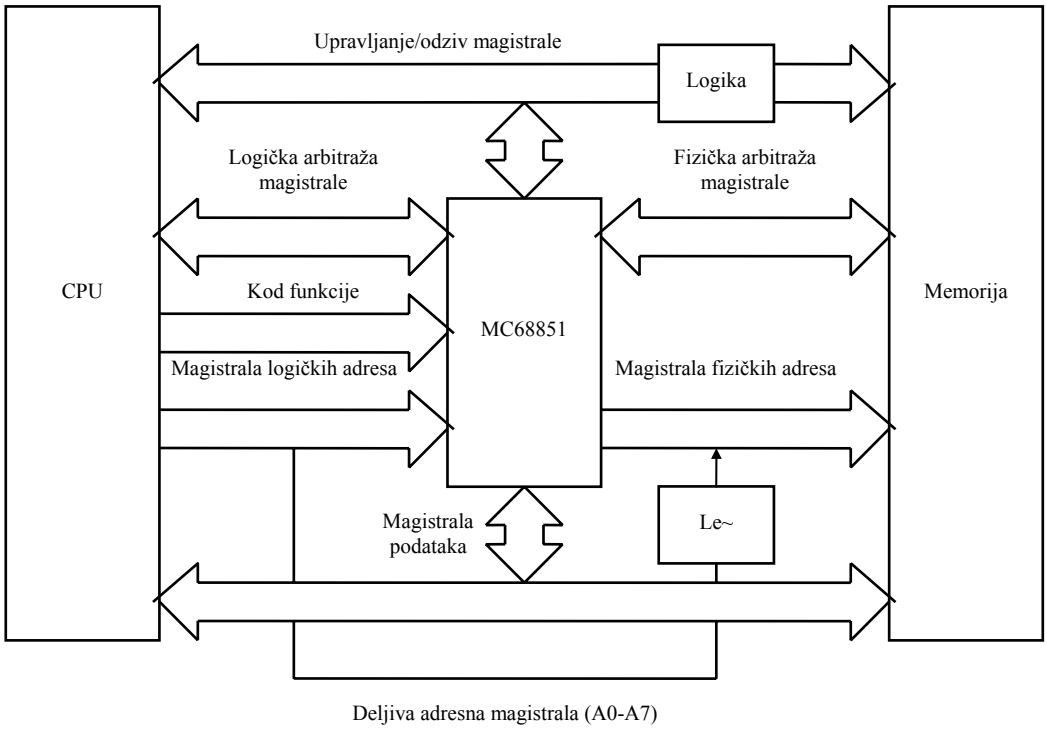
Logički adresni izlazi generisani od strane CPU-a (u toku izvršenja programa) nadgledaju se od strane MC68851. Svaka logička adresa se proverava u saglasnosti sa njenom privilegijom ili opsegom i, ako je važeća, prevodi se u fizičku adresu koja predstavlja memorijsku lokaciju ili lokaciju perifernog uređaja. Na taj način svi pristupi memoriji ili perifernim uređajima tipa čitanje ili upis se kontrolišu i potvrđuju od strane MC68851 kada je ona instalirana u sistem. MC68852 se zove "Paged Memory Managment Unit", jer logičke i fizičke adresne prostore vidi kao da su podeljeni na stranice fiksnog obima. Za svaku logičku stranicu, koju čini 2 bajta instrukcija ili podataka, postoji odgovarajuća stranica u fizičkoj memoriji istog obima koja se zove okvir stranice (page frame). Na slici 5.11 je prikazan konceptualni međusobni odnos između logičkog adresnog prostora koji sadrži jedan ili veći broj programa i fizičke memorije koja je podeljna na okvire stranica. Program u objektnom obliku (kodu) sadrži logičke adrese koje su dodeljene od strane programera, ili kompilatora ili "linkage" editora a čuvaju se u sekundarnoj "virtuelnoj" memoriji koja u najvećem broju slučajeva predstavlja disk. Fizičke adrese se dodeljuju od strane PMMU-a koristeći translacionu tabelu u toku izvršenja programa. Nakon što se program prvo puni u jedan ili veći broj okvira stranica (ovu aktivnost obavlja OS) upravljanje se prepušta programu na taj način što se PC loaduje (postavlja) na početnu LA tog programa.

Kako se program izvršava CPU pribavlja instrukcije čitanjem sa logičkih adresa. PMMU translira ove adrese u fizičke adrese. Na slici 5.11a prikazano je da stvarne lokacije stranica (koje su pridružene programu) ne zauzimaju kontinualne okvire stranica u memoriji. Na ovaj način dozvoljeno je OS-u da dodeli programske stranice (u memoriji) na nekonvencionalan način.

Na slici 5.11b prikazan je osnovni translacioni mehanizam LALFA. LA je podeljena na dva polja. Kada CPU generiše *n*-tobitnu adresu na svojim adresnim linijama, PMMU koristi MS *n* - *m* bitova da bi odredila vrednost u translacionoj tabeli. Stvarna lokacija u transalcionoj tabeli je

(root pointer)+p*(obim ulaza u bajtovima)

gde "root pointer" adresira bazu tabele a broj stranice *p* bira broj ulaza. Tabela ima 2 ulaza. Ova šema se zove direktno preslikavanje jer translacona tabela sadrži ulaz za svaku stranicu LAS-a (MC68851 podržava i druge (kompleksnije) načine preslikavanja). Ulaz svake fizičke stranice kod translacione tabele za direktno preslikavanje sadrži adresu okvira stranice koja je bazna memorijska adresa okvira stranice. Fizička adresa koja se koristi za pristup memoriji dobija se pridruživanjem adrese okvira stranice i LS *m* bitova logičke adrese.

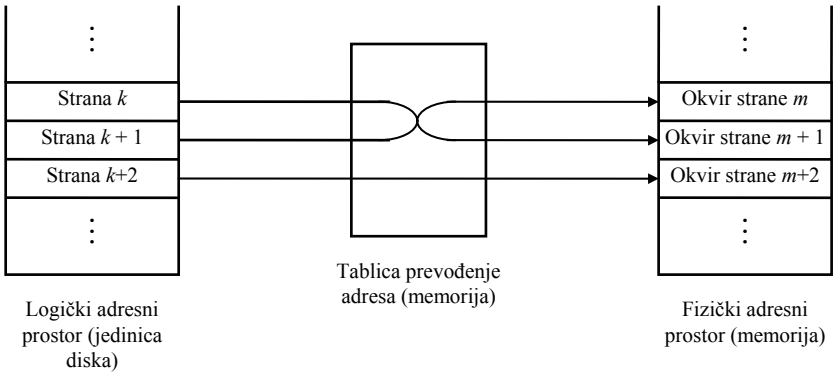


Sl. 5.10. MC68851 PMMU u sistemu zasnovanom na MC68020.

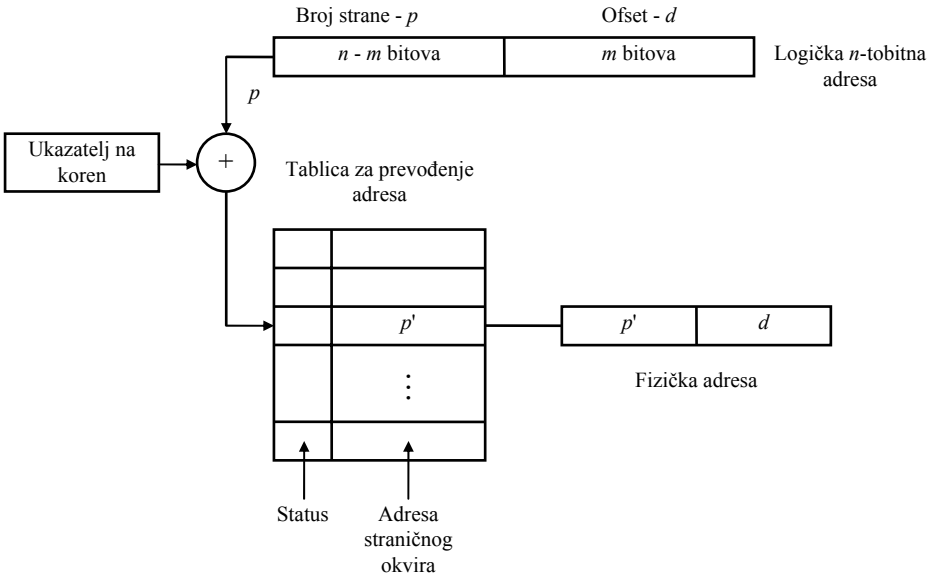
Na slici 5.11b adresom *p*' bira se okvir stranice a ofset *d* bira specifičnu lokaciju u okviru stranice. Obim stranice je 2 bajta. MC68851 sadrži veći broj programabilnih registara. Tri od registara se zovu "root pointer" registri i sadrže statusne i druge informacije koje se odnose na translacione tabele i bazne adrese. Instrukcija pomoću koje se upravlja MC68851 je PMOVE. Ona se koristi za prenos podataka CPU↔MC68851, kao na primer

```
PMOVE    (A1),CRP
```

Više detalja o načinu rada ovog čipa može se naći u kataloškim podacima firme Motorola koji se odnose na rad MC68851.



(a)



(b)

Napomene:
1. Translaciona tabela ima 2^{n-m} stavki u memoriji; šezdesetčetiri stavke se mogu držati u keš memoriji PPMU-a.
2. Veličina stranice je 2^m bajtova.
3. *p*' je bazna adresa izabranog straničnog okvira.

Sl. 5.11.