VERZIJA:7.11.2005.

5. MEMORIJA

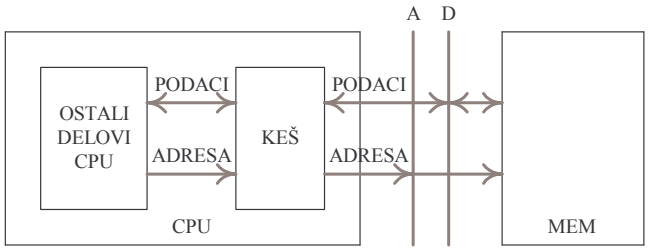
U okviru ove glave razmatraju se tehnike kojima se rešavaju dva problema pri radu sa memorijom i to vreme pristupa i veličina adresnog prostora. U okviru rešavanja problema vremena pristupa razmatraju se dve tehnike i to preklapanje pristupa memorijskim modulima i keš memorija. U okviru rešavanja problema veličine adresnog prostora korisnika razmatra se virtuelna memorija.

5.1. KEŠ MEMORIJA

U ovom poglavlju se najpre daju osnovni pojmovi vezani za ubrzavanje pristupa memorijskim lokacijama korišćenjem keš memorije. Zatim se razmatraju osnovna pitanja vezana za realizaciju keš memorije, i to tehnika preslikavanja, zamena blokova keš memorije i ažuriranje sadržaja operativne memorije. Na kraju se daju neka specifična rešenja čijim se korišćenjem poboljšava rad keš memorija.

5.1.1. Osnovni pojmovi

Mehanizam keš memorije podrazumeva da u procesoru postoji posebna memorija, koja se naziva keš memorija (slika 1). Keš memorija se realizuje sa memorijskim komponentama čije je vreme pristupa daleko manje od vremena pristupa memorijskih komponenata operativne memorije i veoma blisko vremenu prenosa podataka između registara. Zbog toga je cena po bitu keš memorije daleko veća od cene po bitu operativne memorije. To ima za posledicu da je kapacitet keš memorije daleko manji od kapaciteta operativne memorije.



Slika 1 Asocijativno preslikavanje

Algoritam izvršavanja instrukcija u računarima sa keš memorijom je takav da se, pri svakom generisanju adrese operativne memorije od strane nekog dela procesora radi čitanja ili upisa, najpre, vrši provera da li se sadržaj sa generisane adrese nalazi u keš memoriji. Na početku dati sadržaj nije u keš memoriji. Zbog toga se sadržaj sa date i nekoliko susednih adresa, koje čine blok, najpre, prebacuje iz operativne u keš memoriju, pa se potom ili čita sadržaj iz keš memorije i prosleđuje onom delu procesora koji je adresu generisao ili upisuje u keš memoriju sadržaj iz onog dela procesora koje je adresu generisao.

Za nekoliko sledećih generisanih adresa operativne memorije verovatno će se i dalje utvrđivati da se sadržaji takođe ne nalaze u keš memoriji, pa će se odgovarajući blokovi dovlačiti iz operativne memorije u keš memoriju i sa keš memorijom realizovati čitanje ili upis. Time će se broj blokova operativne memorije dovučenih u keš memoriju povećavati. Zbog toga će se od određenog trenutka za neke od generisanih adresa operativne memorije utvrđivati da se sadržaji nalaze u keš memoriji. U svakoj takvoj situaciji sadržaj će se ili čitati iz keš memorije umesto iz operativne memorije ili upisivati u keš memoriju umesto u operativnu memoriju. S obzirom na to da je keš memorija daleko brža od operativne

memorije, u svako takvoj situaciji čitanje ili upis sadržaja će se realizovati sa vremenom pristupa keš memorije umesto sa vremenom pristupa operativne memorije.

Efikasnost mehanizma keš memorije direktno zavisi od toga koliko često će se utvrđivati da se sadržaj sa generisane adrese nalazi u keš memoriji. Analiza tragova generisanih adresa prilikom izvršavanja tipičnih programa ukazuju na dva efekta koji direktno utiču na efikasnost mehanizma keš memorije. Prvi je da se jedanput generisana adresa obično posle toga još nekoliko puta generiše. Ovaj efekat se naziva vremenski lokalitet programa i javlja se kod generisanja adresa instrukcija i skalarnih veličina u petlji. Drugi je da se posle neke generisane adrese veoma često generišu adrese koje slede sekvencijalno. Ovaj efekat se naziva prostorni lokalitet programa i javlja se kod sekvencijalnog izvršavanja instrukcija i sekvencijalnog pristupa podacima koji predstavljaju elemente vektora. Da bi se iskoristio potencijal prvof efekta potrebno je sadržaje memorijskih lokacija kojima se pristupa čuvati u keš memoriji sa sledeće pristupe, a da bi se iskoristio potencijal drugog efekta potrebno je dovlačiti ne samo sadržaj memorijske lokacije kojoj se pristupa, već blok od nekoliko susednih memorijskih lokacija.

Naziv za keš memoriju odražava činjenicu da je mehanizam keš memorije "skriven" od programera i da programer ne može programskim putem da utiče na ono što se dešava u keš memoriji. Provera da li se sadržaj sa generisane adrese nalazi u keš memoriji, eventualno dovlačenje bloka podataka iz operativne memorije u keš memoriju, čitanje podatka iz keš memorije i upis podatka u keš memoriju realizuju se kompletno hardverski. Programer je "svestan" postojanja mehanizma keš memorije indirektno na osnovu toga što će se program izvršavati brže u procesoru sa keš memorijom nego u procesoru bez keš memorije. Zbog toga mehanizam keš memorije ne pripada arhitekturi već organizaciji procesora.

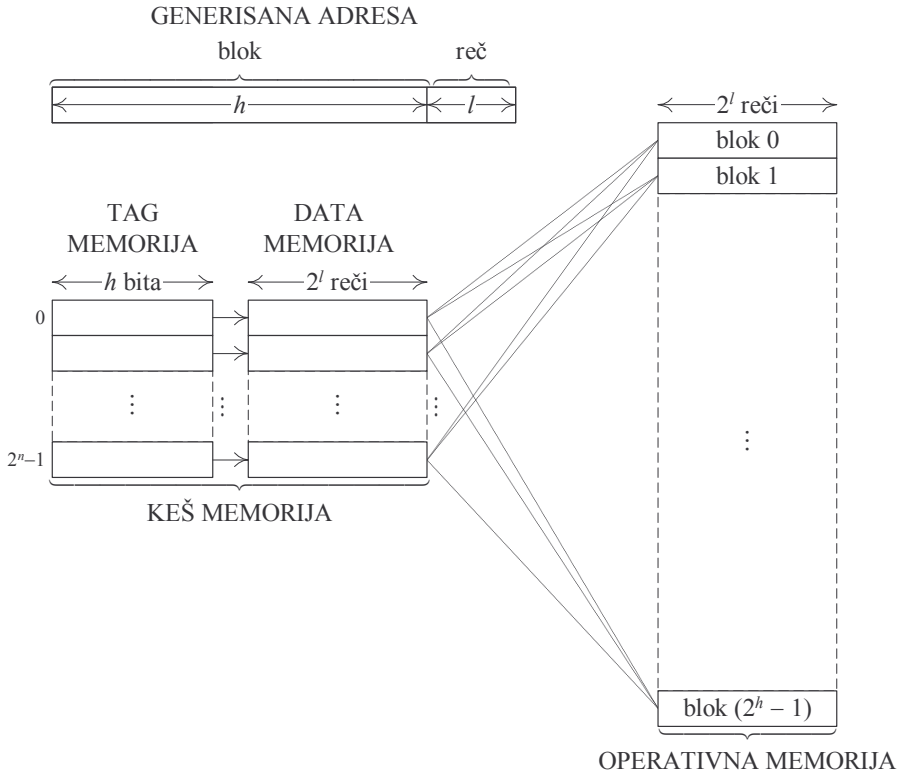
Prilikom realizacije keš memorije postoje tri osnovna pitanja koja treba rešiti. Kapacitet keš memorije je manji od kapaciteta operativne memorije i postoji potreba da se u keš memoriji vodi evedincija o tome koji se blokovi operativne memorije nalaze u keš memoriji i gde se nalaze u keš memoriji. Ovo vođenje evidencije se naziva tehnika preslikavanja. Kapacitet keš memorije je manji od kapaciteta operativne memorije, pa će se posle određenog vremena dešavati da se generišu adrese sa kojih se sadržaji ne nalaze u keš memoriji, a keš memorija je popunjena. Tada postoji potreba da se odluči koji će se blok izbaciti iz keš memorije da bi se u njoj stvorio prostor za dovlačenje bloka iz operativne memorije kome pripada generisana adresa. Ovo odlučivanje se realizuje prema nekom od algoritama zamene. Prilikom operacija upisa i utvrđivanja da u keš memoriji postoji blok kome pripada generisana adresa, upis će se realizovati u keš memoriju. Time se javlja razlika u vrednosti kopije sadržaja sa određene adrese u keš memoriji i u operativnoj memoriji. Sadržaji određenih lokacija operativne memorije se nalaze u keš memoriji samo privremeno radi ubrzanja pristupa. Zbog tog postoji potreba da se na neki način za sve generisane adrese za koje je upis izvršen u kopiju sadržaja u keš memoriji obezbedi ažuriranje i kopije sadržaja u operativnoj memoriji. Načini realizacije zavise od usvojene tehnike ažuriranja sadržaja operativne memorije. Ova tri pitanja su predmet razmatranja u sledeća tri poglavlja.

5.1.2. Tehnike preslikavanja

Tehnika preslikavanja određuje način vođenja evidencije o tome koji se blokovi operativne memorije nalaze u pojedinim blokovima keš memorije. Koriste se tri tehnike preslikavanja i to: asocijativno, direktno i set–asocijativno.

5.1.2.1. Asocijativno preslikavanje

Kod tehnike asocijativnog preslikavanja, keš memorija se sastoji iz DATA MEMORIJE i TAG MEMORIJE (slika 2). U DATA MEMORIJU se smeštaju blokovi preneti iz operativne u keš memoriju. U TAG MEMORIJU se, za blokove prenete iz operativne u keš memoriju, smeštaju brojevi blokova operativne memorije, koji se nazivaju *tag*–ovi. Za realizaciju DATA MEMORIJE i TAG MEMORIJE koriste se RAM memorija i asocijativna memorija, respektivno. Ukoliko u DATA MEMORIJU može da se smesti 2^n blokova, kaže se da keš memorija ima 2^n ulaza. Za svaki ulaz DATA MEMORIJE postoji odgovarajući ulaz TAG MEMORIJE.



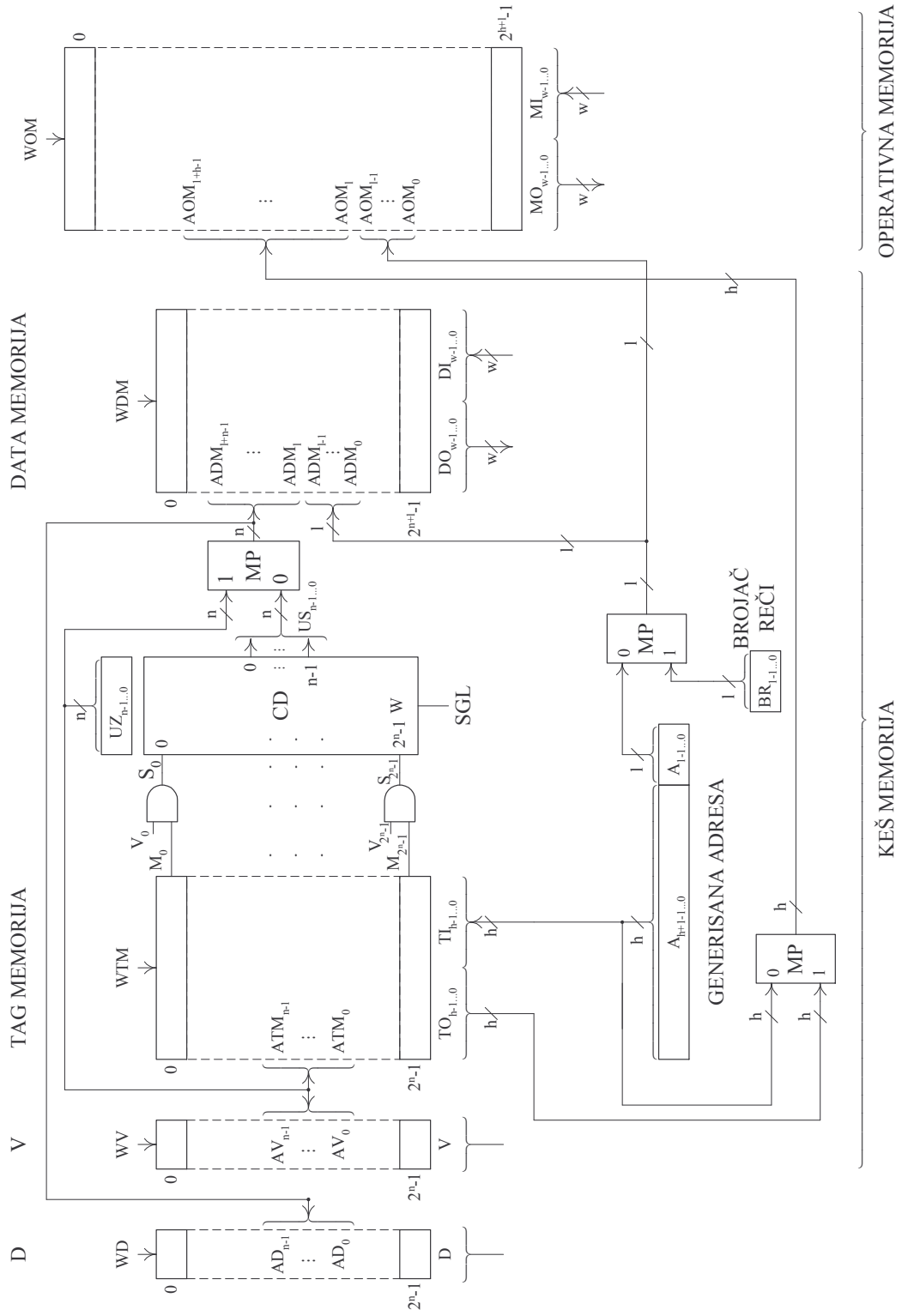
Slika 2 Asocijativno preslikavanje

U slučaju keš memorije realizovane u tehnici asocijativnog preslikavanja, zamišlja se kao da je operativna memorija podeljena na 2^h blokova. Ukoliko se uzme da je dimenzija bloka 2^l reči, tada generisana adresa ima sledeću strukturu: nižih l bitova određuju adresu reči unutar bloka i viših h bitova određuju broj bloka u operativnoj memoriji.

U tehnici asocijativnog preslikavanja bilo koji blok operativne memorije može da se smesti u bilo koji ulaz keš memorije, prilikom dovlačenja iz operativne u keš memoriju. Pošto u i–tom ulazu DATA MEMORIJE može da se nađe bilo koji od 2^h blokova operativne memorije, u i–tom ulazu TAG MEMORIJE se čuva broj bloka operativne memorije kome

pripada dati blok. S obzirom da u operativnoj memoriji ima 2^h blokova, širina memorijske reči TAG MEMORIJE je *h* bita.

Realizacija keš memorije sa asocijativnim preslikavanjem je data na slici 3.



Slika 3 Keš memorija sa asocijativnim preslikavanjem

S obzirom da je operativna memorija kapaciteta 2^{h+l} reči i da je veličina bloka 2^l reči, uzima se kao da je operativna memorija organizovana u 2^h blokova veličine 2^l reči. Stoga se adresa operativne memorije dužine h+l bita može podeliti na sledeći način: viših h bitova označavaju broj bloka i nižih l bitova označavaju adresu reči u bloku.

Pri generisanju zahteva za čitanje od strane procesora, viših h bitova generisane adrese vodi se na ulazne linije podataka $TI_{h-1...0}$ TAG MEMORIJE da bi se, njihovim istovremenim upoređivanjem sa sadržajima svih 2^n ulaza TAG MEMORIJE, utvrdilo da li postoji saglasnost sa sadržajem nekog ulaza. Za svaki od 2^n ulaza TAG MEMORIJE postoji poseban signal saglasnosti M_0 do $M_2{}^n{}_{-1}$, koji svojom aktivnom vrednošću određuje da je na datom ulazu otkrivena saglasnost. Otkrivena saglasnost je važeća, i jedan od signala važećih saglasnosti S_0 do $S_2{}^n{}_{-1}$ je aktivan, ukoliko je odgovarajući indikator važećih ulaza V aktivan. Signal saglasnosti SGL, koji se dobija sa izlaza W kodera CD, ima aktivnu vrednost, ukoliko jedan od signala S_0 do $S_2{}^n{}_{-1}$ ima aktivnu vrednost. Binarna vrednost broja ulaza u kome je otkrivena saglasnost je određena sa n bita sa izlaza kodera CD na osnovu signala S_0 do $S_2{}^n{}_{-1}$. Ukoliko postoji saglasnost, sa n bitova sa izlaza kodera CD i l nižih bitova generisane adrese, adresira se reč DATA MEMORIJE i obavlja čitanje.

Ukoliko saglasnost ne postoji u jedan od ulaza keš memorije treba iz operativne memorije da se dovuče blok u kome se nalazi željeni sadržaj. Ulaz u koji se dovlači blok naziva se ulaz za zamenu. Broj ulaza je određen vrednošću $UZ_{n-1...0}$ do koje se dolazi na osnovu nekog od algoritama zamene. Pre dovlačenja željenog bloka proverava se da li se u ulazu koji je odabran za zamenu nalazi blok koji je nekom od operacija upisa modifikovan. Ovo se utvrđuje na osnovu sadržaja indikatora D adresiranog vrednošću $UZ_{n-1...0}$. Ukoliko je indikator D ulaza odabranog za zamenu 1, blok je modifikovan, pa prvo treba dati blok vratiti u operativnu memoriju pa tek onda dovući željeni blok. Ukoliko je indikator D ulaza odabranog za zamenu 0, blok nije modifikovan, pa se željeni blok odmah dovlači.

Prilikom vraćanja bloka odabranog za zamenu 2^l reči datog bloka se čita iz DATA MEMORIJE sa adresa formiranih od vrednosti $UZ_{n-1...0}$ koja daje n starijih bitova adrese i vrednosti $BR_{l-1...0}$ koja je daje l mlađih bitova adrese. Date reči se upisuju u OPERATIVNU MEMORIJU na adresama formiranih od vrednosti na linijama $TO_{h-1...0}$ koja daje n starijih bitova adrese i vrednosti $BR_{l-1...0}$ koja je daje l mlađih bitova adrese. Vrednost na linijama $TO_{h-1...0}$ pročitana je iz TAG MEMORIJE sa adrese određene vrdnošću $UZ_{n-1...0}$.

Prilikom dovlačenja željenog bloka 2^l reči datog bloka se upisuje u DATA MEMORIJU na adresama formiranih kao i u slučaju vraćanja bloka od vrednosti $UZ_{n-1...0}$ koja daje n starijih bitova adrese i vrednosti $BR_{l-1...0}$ koja je daje l mlađih bitova adrese. Date reči se čitaju iz OPERATIVNE MEMORIJE sa adresa formiranih od vrednosti bitova $A_{h-1...0}$ generisane adrese koja daje n starijih bitova adrese i vrednosti $BR_{l-1...0}$ koja je daje l mlađih bitova adrese. Pored toga bitovi $A_{h-1...0}$ generisane adrese se upisuju u ulaz TAG MEMORIJE čija je adresa određena vrednošću $UZ_{n-1...0}$. Indikatori V i D ulaza adresiranih vrednošću $UZ_{n-1...0}$ postavljaju se na 1 i 0, respektivno.

Po dovlačenju bloka podataka u keš memoriju, utvrđuje se da sada postoji saglasnost, pa se na već opisani način adresira reč DATA MEMORIJE i obavlja čitanje.

Pri generisanju zahteva za upis od strane procesora, na isti način se ispituje saglasnost sa sadržajem keš memorije kao u slučaju operacije čitanja. Ukoliko postoji saglasnost, na isti način se adresira reč DATA MEMORIJE i vrši upis, pri čemu se sada indikator modifikovanog ulaza D, adresiran vrednošću $US_{n-1...0}$ sa izlaza kodera CD i koja predstavlja broj ulaza u kome je otkrivena saglasnost, postavlja na 1.

Ako saglasnost ne postoji, na identičan način kao i za operaciju čitanja, se, najpre, blok iz ulaza keš memorije, određenog vrednošću $UZ_{n-1...0}$, vraća u operativnu memoriju, ukoliko je modifikovan, a zatim, u isti ulaz keš memorije, dovlači novi blok iz operativne memorije. Potom se ponovo, na već opisani način, vrši provera da li postoji saglasnost, utvrđuje da postoji saglasnost i realizuje upis.

5.1.3. Zamena blokova keš memorije

Pri generisanju zahteva za upis ili čitanje od strane procesora može se utvrditi da se blok u kome je zahtevana reč ne nalazi u bloku keš memorije koji je predviđen odabranom tehnikom preslikavanja. Tada se jedan blok keš memorije mora vratiti u operativnu memoriju, da bi se u keš memoriji napravio prostor za blok iz operativne memorije u kome se zahtevana reč nalazi. Ovaj blok se određuje korišćenjem jednog od algoritama zamene koji se hardverski realizuju u keš memoriji.

Kod keš memorije sa direktnim preslikavanjem algoritam zamene je trivijalan jer je blok za zamenu određen brojem bloka generisane adrese. Kod keš memorije sa asocijativnim i set-asocijativnim preslikavanjem algoritmom zamene se za zamenu bira jedan od svih blokova keš memorije sa asocijativnim preslikavanjem i jedan od svih blokova seta, određenog brojem seta generisane adrese, keš memorije sa set-asocijativnim preslikavanjem. Stoga se algoritam zamene realizuje za celu keš memoriju sa asocijativnim preslikavanjem, a posebno za svaki set keš memorije sa set-asocijativnim preslikavanjem.

Pri izboru algoritma zamene treba voditi računa o dva zahteva. Prvi je da on treba da obezbedi minimalnu verovatnoću da će blok koji je odabran za zamenu i vraćen iz keš u operativnu memoriju ubrzo morati ponovo da se dovuče iz operativne u keš memoriju. Drugi je da cena hardvera potrebnog za njegovu realizaciju bude što je moguće niža. Ova dva zahteva su kontradiktorna, jer je cena hardvera algoritama zamene koji bolje ispunjavaju prvi zahtev viša u odnosu na cenu hardvera algoritama zamene koji to čine lošije. Ovde se razmatraju četiri algoritma zamene i to RANDOM, FIFO, LRU i PSEUDO LRU i daju njihova moguća realizacija. Razmatranja se odnose na keš memoriju sa asocijativnim preslikavanjem i važe i za jedan set keš memorije sa set-asocijativnim preslikavanjem.

5.1.3.1. RANDOM

RANDOM algoritmom zamene za zamenu se slučajno bira blok korišćenjem jednog od postojećih generatora slučajnih brojeva. Mogući način realizacije je da se koristi brojač po modulu 2^n, gde 2^n predstalja broj ulaza keš memorije sa asocijativnim preslikavanjem i broj ulaza po setu keš memorije sa set-asocijativnim preslikavanjem. Može se odabrati neki proizvoljan signal i koristiti za inkrementiranje brojača. U trenutku kada nema saglasnoti i treba odabrati blok za zamenu, trenutna vrednost brojača određuje ulaz za zamenu. Posle toga se produžava se inkrementiranje brojača do sledećeg tenutka kada nema saglasnosti i kada ponovo na osnovu vrednosti brojača treba odabrati ulaz za zamenu.

Ovaj algoritam ne vodi računa o tome da će blok koji je odabran za zamenu i vraćen iz keš u operativnu memoriju možda ubrzo morati ponovo da se dovuče iz operativne u keš memoriju. Međutim, hardver za njegovu realizaciju je jedostavan.

5.1.3.2. FIFO

FIFO (*first in—first out*) algoritmom zamene za zamenu se bira blok koji je najranije unet iz operativne memorije u keš memoriju. Mogući način realizacije je da se koristi brojač po modulu 2^n, gde 2^n predstalja broj ulaza keš memorije sa asocijativnim preslikavanjem i broj

ulaza po setu keš memorije sa set-asocijativnim preslikavanjem. U trenutku kada nema saglasnoti i treba odabrati blok za zamenu, trenutna vrednost brojača određuje ulaz za zamenu. Pri tome se i vrši inkrementiranje brojača.

Ovaj algoritam ne vodi računa o tome da će blok koji je odabran za zamenu i vraćen iz keš u operativnu memoriju možda ubrzo morati ponovo da se dovuče iz operativne u keš memoriju, već za zamenu bira blokove po onom redosledu po kome su i dovlačeni iz operativne memorije u keš memoriju. Međutim, hardver za njegovu realizaciju je jedostavan.

5.1.3.3. LRU

LRU (*least recently used*) algoritmom zamene za zamenu se bira blok kome se najduže vremena nije pristupalo. Mogući način realizacije je da se koristi 2^n brojača po modulu 2^n, gde 2^n predstalja broj ulaza keš memorije sa asocijativnim preslikavanjem i broj ulaza po setu keš memorije sa set-asocijativnim preslikavanjem. Svakom od 2^n ulaza keš memorije dodeljuje se jedan od 2^n brojača po modulu 2^n. Brojači se tokom rada tako ažuriraju da je u njima uvek 2^n različitih vrednosti i da brojač ulaza kome se najduže vremena nije pristupalo ima sve jedinice i time vrednost 2^n-1. U trenutku kada ima saglasnoti sadržaj brojača ulaza u kome je otkrivena saglasnot se upoređuje sa sadržajima brojača svih preostalih ulaza. Brojači koji imaju manju vrednost od brojača ulaza u kome je otkrivena saglasnost se inkrementiraju, brojači koji imaju veću vrednost od brojača ulaza u kome je otkrivena saglasnost se ne menjaju i brojač ulaza u kome je otkrivena saglasnost se postavlja na nulu. U trenutku kada nema saglasnosti za zamenu se bira ulaz čiji brojač ima sve jedinice i time vrednost 2^n-1, brojači svih preostalih ulaza imaju manju vrednost od brojača ulaza koji je odabran za zamenu pa se inkrementiraju i brojač ulaza koji je odabran za zamenu se postavlja na nulu. Na početku rada brojači 2^n ulaza treba tako da se inicijalizuju da u njima bude 2^n različitih vrednosti, pri čemu brojače ulaza treba inicijalizovati na vrednosti 2^n-1, 2^n-2, ..., 1 i 0 saglasno redosledu po kome se želi popunjavanje ulaza. Time se i za popunjavanje keš memorije i za zamenu blokova popunjene keš memorije koristi isti mehanizam.

Ovaj algoritam je baziran na pretpostavci da onim blokovima kojima je više pristupano u prošlosti verovatno će biti pristupano i u budućnosti, pa se za zamenu bira blok kome se najmanje skoro pristupalo. Međutim, hardver za njegovu realizaciju je složen.

5.1.4. Ažuriranje operativne memorije

Ažuriranje operativne memorije određuje kako se kod operacije upisa menja sadržaj u operativnoj memoriji. Pri tome se, kod zahteva za upis, mogu javiti dve situacije. Prva je da u keš memoriji postoji saglasnost, a druga da nema saglasnosti.

Za slučaj kada je u keš memoriji otkrivena saglasnost, postoje dva pristupa i to *upiši skroz* (*write through* ili *store through*) i *vrati nazad* (*write back* ili *copy back*). Kod pristupa *upiši skroz*, pri svakom zahtevu za upis istovremeno se vrši upis i u keš memoriju i u operativnu memoriju. Kod pristupa *vrati nazad*, pri svakom zahtevu za upis vrši se upis samo u keš memoriju, pa odgovarajući sadržaj u operativnoj memoriji nije ažuran. Zbog toga se za svaki blok u keš memoriji vodi evidencija o tome da li je modifikovan ili ne. Ukoliko je kasnije potrebno dovući novi blok iz operativne memorije na mesto nekog bloka u keš memoriji koji je nekim od prethodni upisa modifikovan, potrebno je, najpre, dati blok keš memorije vratiti u operativnu memoriju i time obezbediti da i sadržaj u operativnoj memoriji bude ažuran. Pored toga, kada se nekom procesu oduzima procesor, treba proveriti koji su blokovi u keš memoriji modifikovani, pa ih, radi ažuriranja sadržaja u operativnoj memoriji, vratiti iz keš memorije u

operativnu memoriju. Stoga kod keš memorija koje koriste ovaj pristup ažuriranja sadržaja operativne memorije, pored zahteva za čitanje i upis, postoje i zahtevi za selektivno i kompletno vraćanje blokova iz keš memorije u operativnu memoriju (*flush*).

Prednost pristupa *upiši skroz* je u tome da je operativna memorija uvek ažurna čime je obezbeđena konzistentnost sadržaja operativne i keš memorije. Nedostatak ovog pristupa je u obraćanju operativnoj memoriji pri svakom upisu u keš memoriju, čime se bespotrebno opterećuje magistrala upisuivanjem međurezultata u operativnu memoriju.

Prednost pristupa *vrati nazad* je u tome što se operativnoj memoriji i magistrali pristupa samo onda kada se blok vraća iz keš memorije u operativnu memoriju što rezultuje u manjem saobraćaju na magistrali. Nedostatak ovog pristupa je potreba da se blok koji se izbacuje iz keš memorije mora najpre vratiti u operativnu memoriju, pa tek onda dovući novi, što znatno usporava odziv keš memorije u slučaju promašaja.

Ovde se vidi da su sve prednosti jednog pristupa ujedno i nedostaci drugog. Stoga se pristup *vrati nazad* koristi tamo gde je magistrala usko grlo sistema, a pristup *upiši skroz* gde magistrala to nije.

Za slučaj kada je u keš memoriji nije otkrivena saglasnost, postoje dva pristupa i to *dovuci blok* (*write allocate*) i *ne dovlači blok* (*no write allocate*). Kod pristupa *dovuci blok*, blok se dovlači iz operativne u keš memoriju, čime se obezbeđuje da se sada u keš memoriji otkriva saglasnost. Dalji postupak odgovara prethodno opisanoj situaciji za operaciju upisa i otkrivenu saglasnost, u kojoj se upis vrši u keš memoriju, a za ažuriranje sadržaja operativne memorije koristi pristup *upiši skroz* ili *vrati nazad*. Kod pristupa *ne dovlači blok*, blok se ne dovlači iz operativne u keš memoriju, već se upis vrši samo u operativnu memoriju.

Obično se uz pristup *vrati nazad* (*write back* ili *copy back*) koristi pristup *dovuci blok* (*write allocate*), dok se uz pristup *upiši skroz* (*write through* ili *store through*) koristi pristup *ne dovlači blok* (*no write allocate*).

5.1.5. Neka razmatranja u vezi realizacije keš memorije

U osnovni mehanizam funkcionisanja keš memorije moguće je uvesti neka poboljšanja koja skraćuju vreme čitanja iz i upisa u keš memoriju.

Moguće poboljšanje je u tome da keš memorija, ako se radi o operaciji upisa, odmah dozvoli procesoru da produži sa izvršavanjem tekuće instrukcije bez obzira na to da li je upis zaista izvršen ili nije. Time će paralelno keš memorija obavljati upis a procesor izvršavati instrukciju. Keš memorija neće moći da prihvati novi zahtev za upis ili čitanje ukoliko se prethodno započeti upis nije završio.

Poboljšanje je moguće učiniti i u slučaju operacije čitanja kada traženi blok nije u keš memoriji, već ga treba dovući iz operativne memorije. Tada procesor ne mora da čeka da ceo blok bude prenesen iz operativne u keš memoriju i da tek tada dobije traženi sadržaj. Keš memorija može procesoru dostaviti traženi sadržaj čim on stigne iz operativne u keš memoriju. U tom slučaju procesor može ranije da nastavi izvršavanje tekuće instrukcije, a da se paralelno s time ostatak bloka prenese iz operativne u keš memoriju. Pri tome dovlačenje reči bloka treba započeti od reči čije je čitanje zahtevano. Kao u prethodnom slučaju, keš memorija opet ne može prihvatiti novi zahtev za čitanje ili upis sve dok se prenos prethodnog bloka ne obavi do kraja. Ova tehnika naziva se *by-pass*.

Sledeće poboljšanje je moguće ostvariti u slučajevima kada je potrebno izvršiti vraćanje modifikovanog bloka u operativnu memoriju. Da bi se ubrzao taj postupak moguće je u

poseban bafer privremeno smestiti ceo blok koji se vraća i odmah preći na dovlačenje novog bloka iz operativne memorije. Tek po završetku dovlačenja novog bloka prelazi se na vraćanje u operativnu memoriju bloka koji se nalazi u baferu. I ovde keš memorija ne može prihvatiti novi zahtev za čitanje ili upis sve dok se cela operacija ne završi do kraja. Ovo poboljšanje se naziva *baferisanje*.

Sva navedena poboljšanja imaju za cilj da se procesor što manje zadržava prilikom obraćanja keš memoriji. Pri tome se pretpostavlja da se procesor vrlo verovatno neće uskoro ponovo obraćati keš memoriji, pa će do sledećeg obraćanja procesora keš memoriji, keš memorija moći da obavi prethodno započetu operaciju do kraja.

U slučaju operacije upisa postoji više načina da se promene sadržaja operativne i keš memorije realizuju. Ako se koristi pristup *vrati nazad* onda se promena u operativnoj memoriji ostvaruje samo u slučaju vraćanja bloka u operativnu memoriju. Što se tiče keš memorije kod ovog pristupa se promena u keš memoriji ostvaruje uvek i to i u slučaju da ima saglasnosti i u slučaju da nema saglasnosti, pri čemu se u drugom slučaju to čini tek pošto se blok prenese iz operativne u keš memoriju. Ako se koristi pristup *upiši skroz* onda se promena u operativnoj memoriji ostvaruje uvek. Što se tiče keš memorije kod ovog pristupa se u slučaju saglasnosti ili upisuje novi sadržaj u keš memoriju ili se ulaz keš memorije proglašava nevažećim. U slučaju da nema saglasnosti u nekim situacijama ažurirani blok operativne memorije se dovlači u keš memoriju, dok se u drugim ažurirani blok operativne memorije ne dovlači u keš memoriju.

U osnovni mehanizam funkcionisanja keš memorije je moguće uvesti neka poboljšanja, koja skraćuju vreme čitanja iz i upisa u keš memoriju. Moguće poboljšanje je u tome da keš memorija, ako se radi o operaciji upisa, upis izvrši u bafer podatka i odmah dozvoli procesoru da produži sa izvršavanjem tekuće instrukcije. Time će se paralelno obavljati upis iz bafera podatka u operativnu memoriju i izvršavati instrukcije procesora. Ove tehnike se nazivaju *baferovanje podatka* i *rani start procesora*. Poboljšanje je moguće učiniti i u slučaju operacije čitanja kada traženi blok nije u keš memoriji, već ga treba dovući iz operativne memorije. Tada procesor ne mora da čeka da ceo blok bude dovučen iz operativne memorije u keš memoriju i da tek tada dobije traženi sadržaj. Keš memorija može procesoru dostaviti traženi sadržaj čim on stigne iz operativne u keš memoriju. U tom slučaju, procesor može ranije da nastavi izvršavanje tekuće instrukcije i da se paralelno ostatak bloka prenosi iz operativne u keš memoriju. Pri tome, dovlačenje reči bloka treba započeti od reči čije je čitanje zahtevano. Ove tehnike se nazivaju *prosleđivanje* i *rani start procesora*. Sledeće poboljšanje je moguće ostvariti u slučajevima kada je potrebno izvršiti vraćanje modifikovanog bloka odabranog za zamenu iz keš memorije u operativnu memoriju. Da bi se ubrzao taj postupak, moguće je postaviti bafer bloka, koji će prihvatiti ceo blok koji se vraća, i odmah preći na dovlačenje bloka iz operativne memorije. Tek po završetku dovlačenja bloka iz operativne u keš memoriju, prelazi se na vraćanje bloka iz bafera bloka u operativnu memoriju. Ovo poboljšanje se naziva *baferovanje bloka podataka*.

5.2. VIRTUELNA MEMORIJA I JEDINICA ZA UBRZAVANJE

Kod velikog broja savremenih računara ne postoji jedan prema jedan korespondencija između adresa koje se generišu u programu i adresa operativne memorije. Adrese koje se generišu u programu zovu se virtuelne adrese, a adrese operativne memorije realne adrese.

Opseg adresa koje se generišu u programu se zove virtuelni adresni prostor, a opseg adresa operativne memorije realni adresni prostor.

U ovim razmatranjima se uzima da se kompletan virtuelni adresni prostor dodeljuje svakom procesu. Kompletni programi i podaci svakog procesa nalaze se na disku, a samo njihovi delovi za kojima u određenom trenutku postoji potreba dovlače se sa diska i smeštaju u neki deo operativne memorije. Zbog toga kod ovih računara postoji potreba da se za svaki proces vodi evidencija o tome koji se njegovi delovi nalaze u operativnoj memoriji i u kom njenom delu. To se obično realizuje pomoću posebnih tabela koje se formiraju u operativnoj memoriji za svaki proces i koje se nazivaju tabele preslikavanja. U zavisnosti od toga kako se virtuelni adresni prostor radi dovlačenja sa diska u operativnu memoriju deli na delove, razlikuju se tri tipa virtuelnih memorija, i to virtuelne memorije sa straničnom, segmentnom i segmentno-straničnom organizacijom.

Kod virtuelnih memorija stranične organizacije virtuelni adresni prostor se deli na delove fiksne veličine koji se nazivaju stranice, a realni adresni prostor se deli na delove fiksne veličine koji se nazivaju blokovi. Veličina stranice odgovara veličini bloka. Pojedine stranice svih procesa se po potrebi smeštaju u raspoložive blokove operativne memorije. Kada se svi blokovi operativne memorije popune stranicama različitih procesa, neka od tih stranica se vraća na disk da bi se oslobodio blok u operativnoj memoriji za neku novu stranicu.

Kod virtuelnih memorija segmentne organizacije virtuelni adresni prostor se deli na delove promenljive veličine koji se nazivaju segmenti. Pojedini segmenti svih procesa se po potrebi smeštaju u delove operativne memorije koji po veličini odgovaraju veličinama segmenata koji se u njih smeštaju. Kada je kompletna operativna memorija popunjena segmentima različitih procesa, jedan ili više segmenata se vraća na disk da bi se u operativnoj memoriji oslobodio prostor dovoljan za smeštanje segmenta koji se dovlači.

Kod segmentno-stranične organizacije virtuelne memorije virtuelni adresni prostor se deli na delove promenljive veličine koji se nazivaju segmenti, a onda se segmenti dele na delove fiksne veličine koji se nazivaju stranice. Realni adresni prostor se deli na delove fiksne veličine koji se nazivaju blokovi. Veličina stranice odgovara veličini bloka. Pojedine stranice pojedinih segmenata svih procesa se po potrebi smeštaju u raspoložive blokove operativne memorije. Kada se svi blokovi operativne memorije popune stranicama segmenata procesa, neka od stranica se vraća na disk da bi se oslobodio blok za neku novu stranicu negog segmenta nekog procesa.

Razdvajanjem virtuelnog i realnog adresnog prostora javlja se potreba za preslikavanjem virtuelnih adresa u realne korišćenjem tabela preslikavanja. S obzirom da se sve tabele preslikavanja nalaze u operativnoj memoriji, preslikavanje virtuelnih adresa u realne bi drastično usporilo vreme izvršavanja instrukcija. To je razlog što kod računara sa virtuelnom memorijom postoje posebne jedinice, koje će se u daljem tekstu nazivati jedinice za preslikavanje, čiji je zadatak da ubrzaju postupak preslikavanja virtuelnih adresa u realne.

U daljem tekstu se daju osnovne karakteristike virtuelnih memorija i jedinica za preslikavanje.

5.2.1. Organizacija virtuelne memorije

Postoje tri osnovne vrste organizacije virtuelnih memorija, i to:

stranična,

segmentna i

segmentno-stranična.

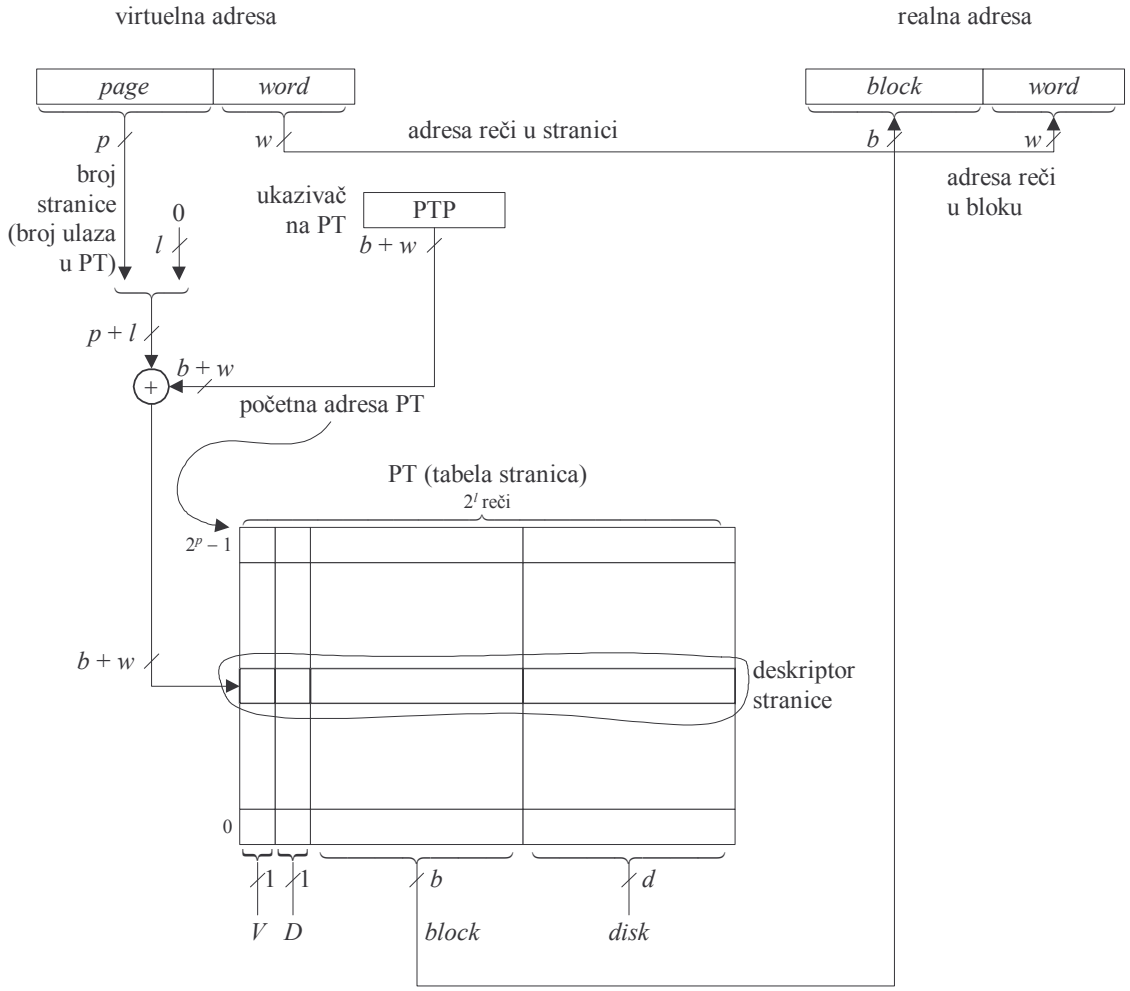
U daljem tekstu se daju samo one osobine svake od navedenih organizacija virtuelne memorije koje su relevantne za realizaciju jedinica za preslikavanje.

5.2.1.1. Stranična organizacija

Kod stranične organizacije virtuelne memorije virtuelna adresa ima dva polja: broj stranice (*page*) dužine *p* bita i adresa reči u stranici (*word*) dužine *w* bita. Kako je veličina virtuelnog adresnog prostora 2^{p+w} reči, a stranice 2^{w} reči, to su veličine polja *page* i *word p* i *w* bita, respektivno. Kod stranične organizacije virtuelne memorije realna adresa ima dva polja: broj bloka (*block*) dužine *b* bita i adresa reči u bloku (*word*) dužine *w* bita. Kako je veličina realnog adresnog prostora 2^{b+w} reči, a bloka 2^{w} reči, to su veličine polja *block* i *word b* i *w* bita, respektivno. Struktura virtuelne i realne adrese i dužine u bitovima delova virtuelne i realne adrese su prikazani na slici 1.

Tabela stranica (PT) jednog procesa je data na slici 1. Tabela stranica ima poseban ulaz za svaku stranicu procesa. U njima se nalaze informacije neophodne za preslikavanje stranica virtuelnog adresnog prostora procesa u blokove fizičke memorije. Te informacije se nazivaju deskriptori stranica. S obzirom da u virtuelnom adresnom prostoru procesa ima 2^{p} stranica, to i tabela stranica ima 2^{p} ulaza. Početna adresa tabele stranica sadržana je u posebnom registru procesora koji se naziva ukazivač na PT (PTP). Na osnovu sadržaja ukazivača na PT i broja stranice, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora stranice, a time i do informacija neophodnih za preslikavanje.

U operativnoj memoriji postoji posebna tabela stranica svakog procesa. Početne adrese tabela stranica svih procesa čuva operativni sistem. Prilikom prebacivanja procesora sa procesa na procesa operativni sistem, najpre, čuva kontekst procesa kome se oduzima procesor, a potom, restaurira kontekst procesa kome se dodeljuje procesor. Tom prilikom se u ukazivač na PT upisuje početna adresa tabele stranica procesa kome se dodeljuje procesor. Time se obezbeđuje da jedinica za preslikavanje pristupa tabeli stranica procesa kome je dodeljen procesor.



Slika 1 Virtuelna adresa, realna adresa i tabela stranica za straničnu organizaciju virtuelne memorije

Polja deskriptora stranice su:

V (1 bit)—stranica u memoriji,

D (1 bit)—stranica modifikovana,

block (*b* bita)—broj bloka i

disk (*d* bita)—adresa na disku.

Polje *V* označava da li je data stranica u operativnoj memoriji. Postavlja ga operativni sistem prilikom dovlačenja date stranice sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje i to ako

je postavljen, da formira realnu adresu i

nije postavljen, da generiše prekid.

Polje *D* označava da li je data stranica modifikovana. Postavlja ga jedinica za preslikavanje ako je bilo operacija upisa u datu stranicu. Koristi ga operativni sistem onda kada odabere datu stranicu za zamenu i to ako

je postavljen, da datu stranicu vrati na disk

nije postavljen, da datu stranicu ne vraća na disk.

Polje *block* označava broj bloka operativne memorije u kome se nalazi data stranica. Vrednost u ovom polju ima smisla jedino ako je polje *V* postavljeno. Polje *block* postavlja operativni sistem prilikom dovlačenja date stranice sa diska u operativnu memoriju. Koristi ga jedinica za preslikavanje, i to ako je polje V postavljeno, da formira realnu adresu.

Polje *disk* označava adresu date stranice na disku. Polje *disk* postavlja operativni sistem prilikom formiranja tabele stranice datog procesa. Koristi ga samo operativni sistem i to radi lociranja date stranice na disku prilikom

dovlačenja stranice sa diska u odabrani blok operativne memorije

vraćanja modifikovane stranice iz bloka operativne memorije odabranog za zamenu na disk.

Uzeto je da je veličina adrese stranice na disku *d* bitova.

Preslikavanje virtuelne u realnu adresu za slučaj da je stranica u memoriji realizuje se kompletno hardverski i to pomoću jedinice za preslikavanje. Ovo preslikavanje se realizuje u sledećim koracima:

Polje *page* virtuelne adrese predstavlja broj ulaza u tabelu stranica u kome se nalazi deskriptor date stranice. S obzirom na to da deskriptor stranice zauzima 2^l reči potrebno je broj ulaza u tabelu stranica pretvoriti u pomeraj u odnosu na početak tabele stranica. To se realizuje pomeranjem ulevo za *l* mesta sadržaja polja *page* vitruelne adrese.

Sadržaj registra ukazivač na PT predstavlja početnu adresu tabele stranica. Pomeraj u odnosu na početak tabele stranica formiran u prethodnom koraku se sabira sa sadržajem registra ukazivač na PT i time dobija adresa deskriptora date stranice. Počev od formirane adrese treba očitati odgovarajući broj reči da bi se došlo do polja *V* i *block* koja koristi jedinica za preslikavanje.

Polje *V* deskriptora ukazuje na to da li je data stranica u memoriji. Stoga se, najpre, sa adrese formirane u prethodnom koraku čita prva reč u kojoj se nalazi polje *V* deskriptora i vrši provera da li je ovo polje postavljeno. Za slučaj kada je stranica u memoriji utvrdiće se da je polje *V* postavljeno, pa se produžava sa sledećim koracima.

Polje *block* deskriptora sadrži broj bloka u kome se nalazi data stranica. Stoga se sada čita potreban broj reči deskriptora da bi se do njega došlo. Konkatenacijom polja *block* iz deskriptora stranice i polja *word* iz virtuelne adrese formira se realna adresa.

U slučaju da stranica nije u memoriji realizuju se samo prva tri od četiri koraka za slučaj kada je stranica u memoriji. U koraku tri se u ovom slučaju utvrđuje da polje *V* nije postavljeno, pa se generiše prekid. Svi koraci do generisanja signala prekida, uključujući i njegovo generisanje, realizuju se hardverski pomoću jedinice za preslikavanje. Sve ostalo, što za ovaj slučaj potom treba uraditi, radi se softverski i to radi poseban deo operativnog sistema. Ovo se realizuje u sledećim koracima:

Oduzima se procesor datom procesu, njegov kontekst čuva i proces stavlja u red blokiranih procesa.

Organizuje se dovlačenje date stranice sa diska u neki od blokova operativne memorije.

Dodeljuje se procesor nekom od radnosposobnih procesa. Tada se restaurira njegov kontekst. U okviru toga se u programski brojač upisuje nova vrednost i time kontrola predaje tom novom procesu.

Dovlačenje stranice sa diska u neki od blokova operativne memorije se realizuje u sledećim koracima:

Traži se blok u koji će se smestiti data stranica. Pri tome se izvršava jedan od sledeća dva koraka:

Ukoliko ima slobodnih blokova po nekom algoritmu se odlučuje koji blok treba dodeliti datoj stranici.

Ukoliko nema slobodnih blokova po nekom algoritmu se odlučuje koju stranicu treba izbaciti iz operativne memorije da bi se blok u kome se ona nalazi oslobodio i dodelio datoj stranici. Ukoliko je stranica odabrana za izbacivanje modifikovana mora se vratiti na disk pre nego što se blok u kome se ona nalazi koristi da se u njega dovuče nova stranica. Adresa na disku stranice odabrane za izbacivanje dobija se iz polja *disk* deskriptora date stranice. Po završenom vraćanju date stranice na disk, u polje V deskriptora stranice koja je vraćena na disk upisuje se nula.

Dovlači se data stranica sa diska u odabrani blok operativne memorije. Adresa date stranice na disku dobija se iz polja *disk* deskriptora stranice. Po završenom dovlačenju date stranice sa diska u polje *block* deskriptora stranice upisuje se broj bloka, a u polje V jedinica.

Proces za koji je dovučena stranica prevodi se u red radnosposobnih procesa.

U nekom trenutku dati proces ponovo dobija procesor. Dati proces ponovo generiše adresu za koju je bilo utvrđeno da je iz stranice koja nije bila u memoriji. Pošto je sada data stranica u memoriji ponavljaju se koraci za slučaj kada je stranica u memoriji.

5.2.2. Organizacija jedinica preslikavanja

Postoje tri osnovne vrste jedinica za preslikavanje i to:

jedinica sa asocijativnim preslikavanjem,

jedinica sa direktnim preslikavanjem i

jedinica sa set-asocijativnim preslikavanjem.

Bilo koja vrsta jedinice može da se koristi sa bilo kojom vrstom virtuelne memorije. Međutim, jedinica sa asocijativnim preslikavanjem, prikazana u odeljku **Error! Reference source not found.**, je realizovana sa virtuelnom memorijom stranične organizacije, jedinica sa direktnim preslikavanjem, prikazana u odeljku **Error! Reference source not found.**, je realizovana sa virtuelnom memorijom segmentne organizacije i jedinica sa set-asocijativnim preslikavanjem, prikazana u odeljku **Error! Reference source not found.**, je realizovana sa virtuelnom memorijom segmentno-stranične organizacije. Stoga su opšti principi realizacije tri tipa jedinica za preslikavanje dati za tri različita tipa virtuelnih memorija, saglasno navedenim realizacijama.

5.2.2.1. Jedinica sa asocijativnim preslikavanjem

S obzirom:

da u adresnom prostoru procesa ima 2^p stranica,

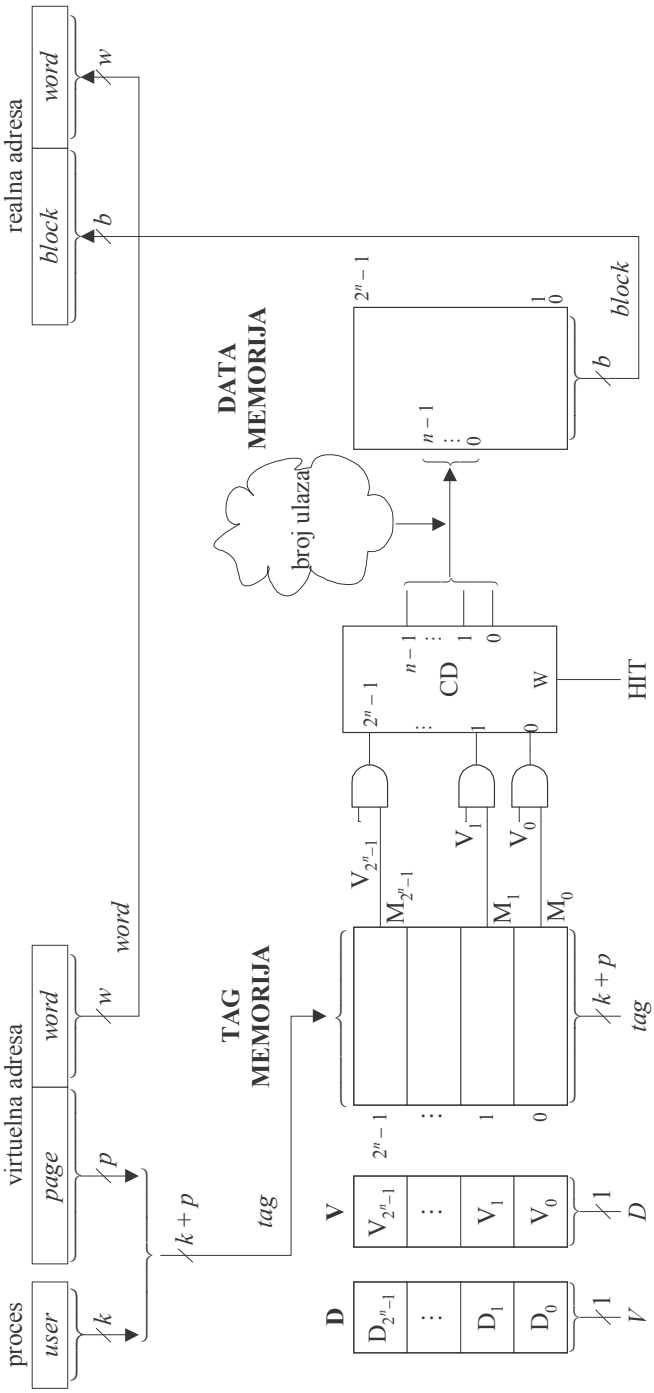
da je broj procesa 2^k i

da se u jedinici za preslikavanje mogu istovremeno čuvati deskriptori stranica različitih procesa,

zaključuje se da je, kao što je prikazano na slici 2, *tag* dužine $k + p$ bita i da je formiran tako da:

k najstarijih bitova predstavljaju broj procesa (*user*) i

p najmlađih bitova predstavljaju broj stranice (*page*) iz virtuelne adrese.



Slika 2 Jedinica sa asocijativnim preslikavanjem za virtuelnu memoriju stranične organizacije

Na osnovu zadatih karakteristika virtuelne memorije i jedinice za preslikavanje dolazi se do strukture jedinice za preslikavanje prikazane na istoj slici.

Jedinica za preslikavanje se sastoji iz sledećih delova:

$D_{0\ldots2^n-1}$ (dirty bitovi)—2^n flip-flopova,

$V_{0\ldots2^n-1}$ (valid bitovi)—2^n flip-flopova,

TAG MEMORIJA—asocijativna memorija kapaciteta 2^n reči širine $k + p$ bitova,

CD—koder i

DATA MEMORIJA—RAM memorija kapaciteta 2^n reči širine b bita.

Dirty bitovi označavaju za svaki od 2^n ulaza jedinice za preslikavanje da li je bilo upisa u neku od lokacija date stranice.

Valid bitovi označavaju za svaki od 2^n ulaza jedinice za preslikavanje da li je odgovarajući ulaz TAG MEMORIJE važeći.

TAG MEMORIJA služi za čuvanje 2^n TAG polja stranica čiji se delovi deskriptora nalaze u odgovarajućim ulazima DATA MEMORIJE i generisanje aktivnih vrednosti signala saglasnosti $\mathbf{M_{0\ldots2^n-1}}$ ukoliko postoji saglasnost TAG polja generisane adrese i sadržaja odgovarajućeg ulaza TAG MEMORIJE.

CD služi za generisanje aktivne vrednosti signala saglasnosti **HIT** i broja ulaza u jedinicu za preslikavanje za koji je otkrivena saglasnost.

DATA MEMORIJA služi za čuvanje 2^n deskriptora stranica.

TAG bitovi ($k + p$) iz generisane adrese se porede sa sadržajima svih 2^n ulaza u TAG MEMORIJI. Ako se sadržaj bilo kojeg ulaza slaže sa TAG bitovima generisane adrese i odgovarajući V bit je postavljen, signal saglasnosti HIT postaje aktivan.

Bitovi (n) koji označavaju broj ulaza u jedinicu za preslikavanje gde je otkrivena saglasnost dobijeni sa izlaza kodera se koriste kao adresa u DATA MEMORIJI i deskriptor se čita.

Očitano polje *block* daje b najstarijih bitova a polje *word* iz virtuelne adrese w najmlađih bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se stranice različitih procesa koje imaju isti broj preslikale svaka u svoj blok, u formiranju *tag* polja učestvuje ne samo polje *page* generisane adrese, već i vrednost registra *user* procesora. Kod prebacivanja procesora sa procesa na proces u ovaj registar procesora se upisuje broj procesa kome se dodeljuje procesor.

Ukoliko se utvrdi da se relevantni deskriptor ne nalazi u hardveru, ide se hardverski u tabelu stranica i čita deskriptor na način prikazan u odeljku 5.2.1.1. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Korišćenjem FIFO ili LRU algoritma zamene bira se ulaz jedinice za preslikavanje u koji se smešta deskriptor. U dati ulaz DATA MEMORIJE se upisuje polje *block* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* flip-flopova se upisuje 1, a u isti ulaz *D* flip-flopova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se stranica ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači datu stranicu sa diska na način prikazan u odeljku 5.2.1.1.