JOVAN ĐORĐEVIĆ

ARHITEKTURA I ORGANIZACIJA RAČUNARA

HIJERARHIJSKA ORGANIZACIJA MEMORIJE

PRIRUČNIK ZA SIMULACIJU SA ZADACIMA

Beograd 1999.

Sadržaj

1. UVOD

Neka priča o hijerarhiji memorija: virtuelna, keš i operativna memorija sa preklapanjem pristupa modulima.

Memorija u najvećem broju računara se organizuje sa ciljem da se obezbedi velika brzina pristupa operativnoj memoriji i veliki adresni prostor. Ostvarivanje ovih ciljeva sa brzim memorijama velikog kapaciteta je moguće samo dotle dok je cena tako realizovane memorije u nekoj prihvatljivoj srazmeri sa cenom ostalih delova računara. Stoga se u računarima koriste određene tehnike kojima se na drugačiji, po ceni daleko prihvatljiviji, način ubrzava vreme pristupa operativnoj memoriji i povećava adresni prostor. Najčešće korišćene tehnike ubrzavanja pristupa operativnoj memoriji su tehnike keš memorije i preklapanja pristupa memorijskim modulima. Povećavanje adresnog prostora se realizuje korišćenjem tehnike virtuelne memorije.

U ovoj glavi se razmatraju preklapanje pristupa memorijskim modulima, keš memorija i virtuelna memorija. Preklapanje pristupa memorijskim modulima i kep memorija su hardverske tehnike kojima se ubrzava pristup operativnoj memoriji. Virtuelna memorija je softversko-hardverska tehnika kojom se uspostavlja korespondencija između adresa korisničkog i fizičkog adresnog prostora.

2. VIRTUELNA MEMORIJA I JEDINICE ZA PRESLIKAVANJE

2.1. OPŠTE NAPOMENE

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.

2.1.1. 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.

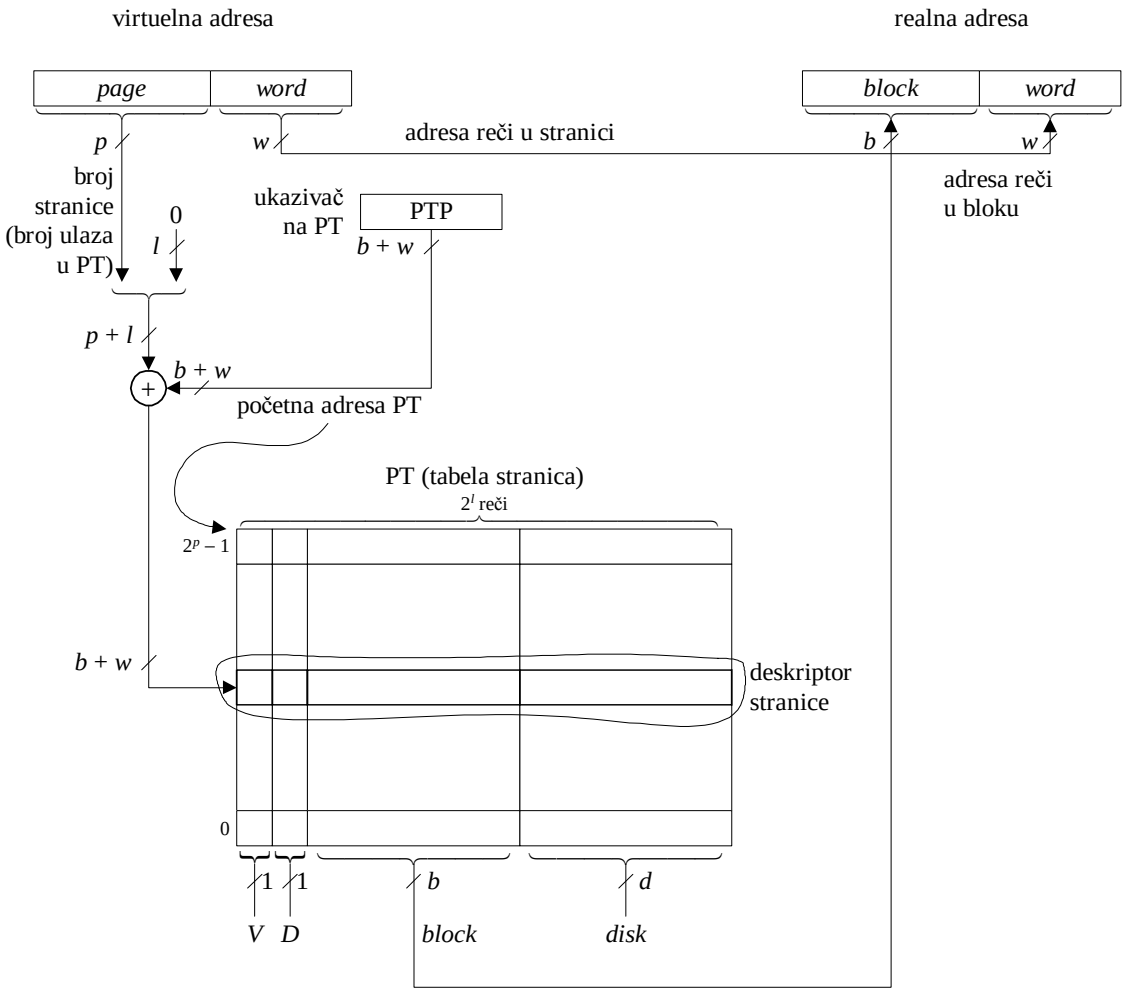
2.1.1.1. STRANIČNA ORGANIZACIJA

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:

1. 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.
2. 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.
3. 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.
4. 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.

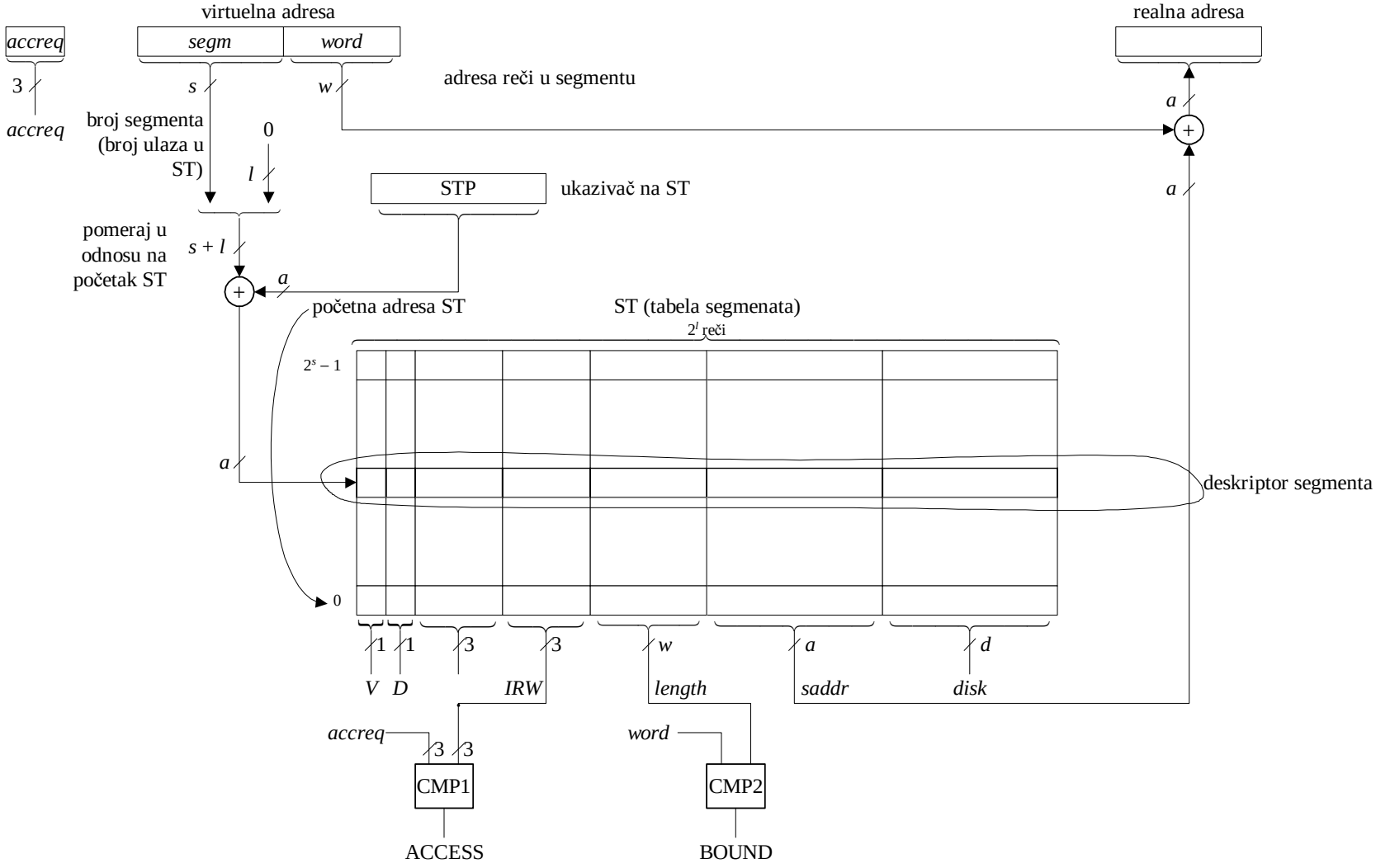
U nekom trenutku dati proces ponovo dobija procesor. Dati proces ponovo generiše adresu za koju je bilo utvrđeno da je iz stranice koja nije bila u memoriji. Pošto je sada data stranica u memoriji ponavljaju se koraci za slučaj kada je stranica u memoriji.

2.1.1.2. SEGMENTNA ORGANIZACIJA

Kod segmentne organizacije virtuelne memorije virtuelna adresa ima dva polja: broj segmenta (*segm*) dužine *s* bita i adresa reči u segmentu (*word*) dužine *w* bita. Kako je veličina virtuelnog adresnog prostora 2^{s+w} reči i sastoji se od 2^s segmenata maksimalne veličine 2^w reči, to su veličine polja *segm* i *word* *s* i *w* bita, respektivno. Kako je veličina realnog adresnog prostora 2^a reči to je veličina realne adrese *a* bita. Struktura virtuelne i realne adrese i dužine u bitovima delova virtuelne i realne adrese su prikazani na slici 2.

Tabela segmenata (ST) jednog procesa je data na slici 2. Tabela segmenata ima poseban ulaz za svaki segment procesa. U njima se nalaze informacije neophodne za preslikavanje segmenata virtuelnog adresnog prostora procesa u delove fizičke memorije veličine segmenata. Te informacije se nazivaju deskriptori segmenata. S obzirom da u virtuelnom adresnom prostoru procesa ima 2^s segmenata, to i tabela segmenata ima 2^s ulaza. Početna adresa tabele segmenata sadržana je u posebnom registru procesora koji se naziva ukazivač na ST (STP). Na osnovu sadržaja ukazivača na ST i broja segmenta, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora segmenta, a time i do informacija neophodnih za preslikavanje.

U fizičkoj memoriji postoji posebna tabela segmenata svakog procesa. Početne adrese tabela segmenata 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 ST upisuje početna adresa tabele segmenata procesa kome se dodeljuje procesor. Time se obezbeđuje da jedinica za preslikavanje pristupa tabeli segmenata procesa kome je dodeljen procesor.


virtuelna adresa
realna adresa
accreq
segm
word
3
s
w
adresa reči u segmentu
accreq
broj segmenta (broj ulaza u ST)
0
l
a
STP
ukazivač na ST
a
pomeraj u odnosu na početak ST
s + l
a
početna adresa ST
ST (tabela segmenata)
2^l reči
2^s – 1
a
deskriptor segmenta
0
1
1
3
3
w
a
d
V
D
IRW
length
saddr
disk
accreq
word
3
3
CMP1
CMP2
ACCESS
BOUND

Slika **2** Virtuelna adresa, realna adresa i tabela stranica za segementnu organizaciju virtuelne memorije

Polja deskriptora segmenta su:

* *V* (1 bit)—segment u memoriji,
* *D* (1 bit)—segment modifikovan,
* *IRW* (3 bita)—bitovi prava pristupa,
* *length* (*w* bita)—veličina segmenta,
* *saddr* (*a* bita)—početna adresa segmenta u operativnoj memoriji i
* *disk* (*d* bita)—adresa segmenta na disku.

Polje *V* označava da li je dati segment u operativnoj memoriji. Postavlja ga operativni sistem prilikom dovlačenja datog segmenta 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 dati segment modifikovan. Postavlja ga jedinica za preslikavanje ako je bilo operacija upisa u dati segment. Koristi ga operativni sistem onda kada odabere dati segment za zamenu i to ako

* je postavljen, da dati segment vrati na disk i
* nije postavljen, da dati segment ne vraća na disk.

Polje *IRW* označava dozvoljen pristup datom segmentu. Uzeto je da

* postavljen bit *I* označava da iz njega smeju da se čitaju samo instrukcije,
* postavljen bit *R* označava da iz njega smeju da se čitaju podaci i
* postavljen bit *W* označava da iz njega smeju da se upisuju podaci.

Polje *IRW* formira operativni sistem prilikom formiranja tabele segmenata datog procesa. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem registra *accreg*, utvrdi da li zahtevani pristup segmentu odgovara dozvoljenom pristupu segmentu.

Polje *length* označava veličinu segmenta. Polje *length* postavlja operativni sistem prilikom formiranja tabele segmenata datog procesa. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem polja *word* virtuelne adrese, utvrdi da li je adresa reči u segmentu unutar specificirane veličine segmenta.

Polje *saddr* označava početnu adresu dela operativne memorije u kome se nalazi dati segment. Vrednost u ovom polju ima smisla jedino ako je polje *V* postavljeno. Polje *saddr* postavlja operativni sistem prilikom dovlačenja segmenta 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 datog segmenta na disku. Polje *disk* postavlja operativni sistem prilikom formiranja tabele segmenata datog procesa. Koristi ga samo operativni sistem i to radi lociranja datog segmenta na disku prilikom

* dovlačenja segmenta sa diska u odabrani deo operativne memorije i
* vraćanja modifikovanog segmenta iz dela operativne memorije odabranog za zamenu na disk.

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

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

1. Polje *segm* virtuelne adrese predstavlja broj ulaza u tabelu segmenata u kome se nalazi deskriptor datog segmenta. S obzirom na to da deskriptor segmenta zauzima 2^l reči potrebno je broj ulaza u tabelu segmenata pretvoriti u pomeraj u odnosu na početak tabele segmenata. To se realizuje pomeranjem ulevo za *l* mesta sadržaja polja *segm* vitruelne adrese.
2. Sadržaj registra ukazivač na ST predstavlja početnu adresu tabele segmenata. Pomeraj u odnosu na početak tabele segmenata formiran u prethodnom koraku se sabira sa sadržajem registra ukazivač na ST i time dobija adresa deskriptora datog segmenta. Počev od formirane adrese treba očitati odgovarajući broj reči, jer se u njima nalaze polja *V*, *IRW*, *length* i *saddr* koja koristi jedinica za preslikavanje.
3. Polje *V* deskriptora ukazuje na to da li je dati segment 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 segment u memoriji utvrđuje se da je polje *V* postavljeno, pa se produžava sa sledećim koracima.
4. Polja *IRW*, *length* i *saddr* deskriptora sadrže bitove prava pristupa, veličinu segmenta i početnu adresu segmenta u fizičkoj memoriji, respektivno. Stoga se sada čita potreban broj reči deskriptora da bi se do njih došlo. Sada se istovremeno:
   * sabiranjem polja *saddr* iz deskriptora segmenta i polja *word* iz virtuelne adrese formira realna adresa,
   * upoređivanjem polja *IRW* iz deskriptora segmenta i sadržaja registra *accreg* utvrđuje da li se zahteva pristup koji je dozvoljen za dati segment i
   * upoređivanjem polja *length* iz deskriptora segmenta i polja *word* iz virtuelne adrese utvrđuje da li je adresa u segmentu unutar specificirane veličine segmenta.

Ukoliko su signali **ACCESS** i **BOUND** neaktivni, zahtevani pristup segmentu i virtuelna adresa su korektni, pa se formirana realna adresa koristi za pristup operativnoj memoriji. U suprotnom nema obraćanja operativnoj memoriji već se generiše prekid. Sve ostalo, što za ovaj slučaj potom treba uraditi, radi operativni sistem i to u sledećim koracima:

- Oduzima procesor datom procesu, njegov kontekst čuva i terminira procesa.
- Šalje poruku na konzolu da je dati proces terminiran.
- Dodeljuje 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.

U slučaju da segment nije u memoriji realizuju se samo prva tri od četiri koraka za slučaj kada je segment u memoriji. U koraku 3. 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 operativni sistem i to u sledećim koracima:

- Oduzima procesor datom procesu, njegov kontekst čuva i proces stavlja u red blokiranih procesa.
- Organizuje dovlačenje datog segmenta sa diska u neki od delova operativne memorije veličine segmenta.

- Dodeljuje 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.

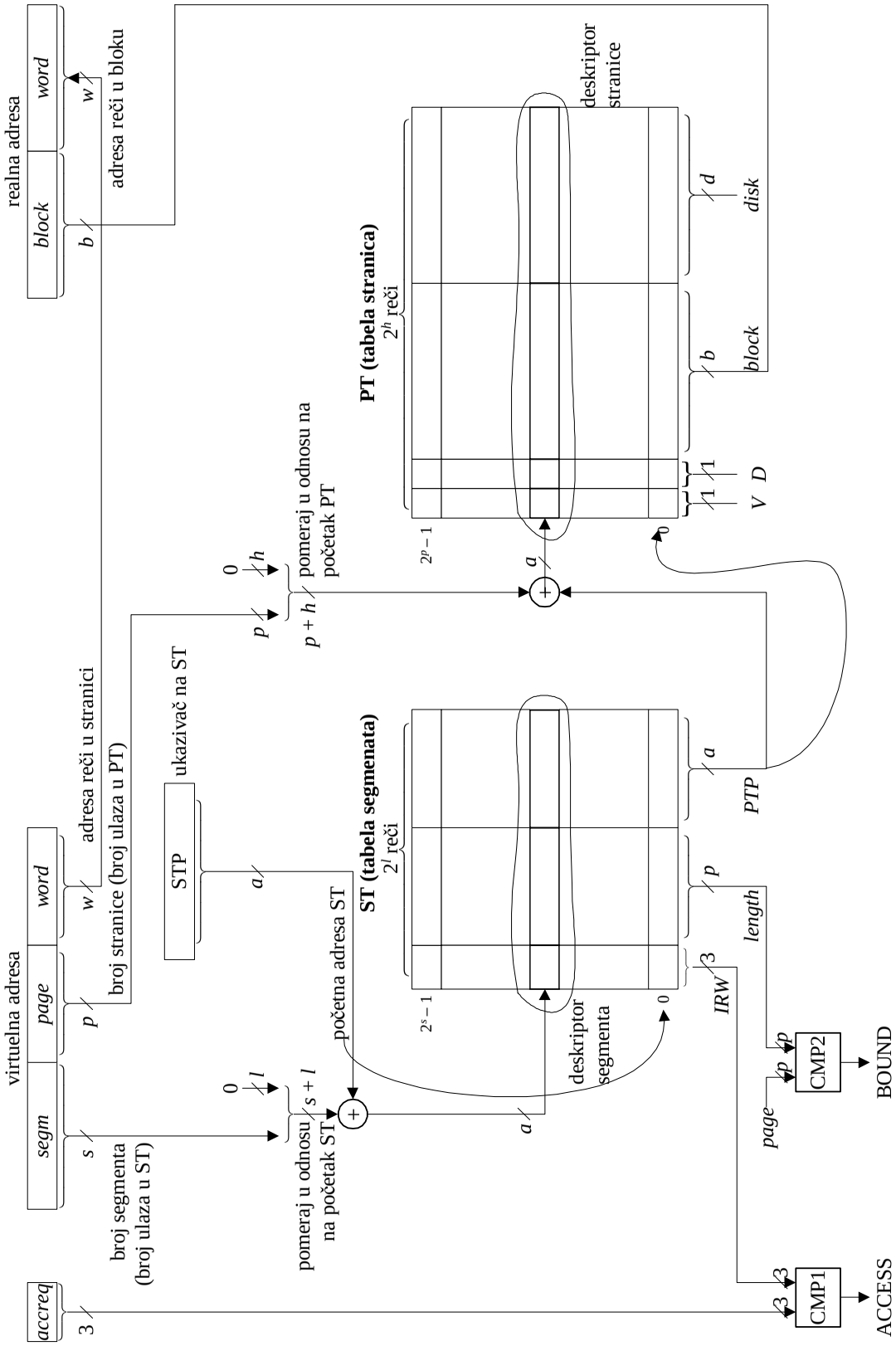
Dovlačenje segmenta sa diska u neki od delova operativne memorije se realizuje u sledećim koracima:

- Traži se deo operativne memorije u koji će se smestiti dati segment. Pri tome se izvršava jedan od sledeća dva koraka:
  - Ukoliko ima slobodnih delova operativne memorije po nekom algoritmu se odlučuje koji deo treba dodeliti datom segmentu.
  - Ukoliko nema slobodnih delova operativne memorije po nekom algoritmu se odlučuje koji segment treba izbaciti iz operativne memorije da bi se deo operativne memorije u kome se on nalazi oslobodio i dodelio datom segmentu. Ukoliko je segment odabran za izbacivanje modifikovan mora se vratiti na disk pre nego što se deo operativne memorije u kome se on nalazi koristi da se u njega dovuče novi segment. Adresa na disku segmenta odabranog za izbacivanje dobija se iz polja *disk* deskriptora segmenta. Po završenom vraćanju datog segmenta na disk, u polje *V* deskriptora segmenta koji je vraćen na disk upisuje se nula.
- Dovlači se dati segment sa diska u odabrani deo operativne memorije. Adresa datog segmenta na disku dobija se iz polja *disk* deskriptora segmenta. Po završenom dovlačenju datog segmenta sa diska u polje *saddr* deskriptora segmenta upisuje se početna adresa dela segmenta u operativnoj memoriji, a u polje *V* jedinica.
- Proces za koga je dovučen segment 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 segmenta koji nije bio u operativnoj memoriji. Pošto je sada dati segment u operativnoj memoriji ponoviće se koraci za slučaj kada je segment u memoriji.

2.1.1.3. SEGMENTNO-STRANIČNA ORGANIZACIJA

Kod segmentno-stranične organizacije virtuelne memorije virtuelna adresa ima tri polja: broj segmenta (*segm*) dužine *s* bita, broj stranice (*page*) dužine *p* bita i adresa reči u stranici (*word*) dužine *w* bita. Kako je veličina virtuelnog adresnog prostora 2^{s+p+w} reči i sastoji se iz 2^s segmenata maksimalne veličine 2^w reči, a oni su dalje podeljeni na 2^p stranica veličine 2^w reči, to su veličine polja *segm, page* i *word s, p* i *w* bita, respektivno. Kod segmentno-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 3.

Kod segmentno-stranične organizacije virtuelne memorije svaki proces ima jednu tabelu segmenata i onoliko tabela stranica koliko ima segmenata u virtuelnom adresnom prostoru procesa. Za dati procesor postoji jedna tabela segmenata i maksimalno 2^p tabela stranica.



Slika 3 Virtuelna adresa, realna adresa i tabela stranica za segmentno-straničnu organizaciju virtuelne memorije

Tabela segmenata (ST) jednog procesa je data na slici 3. Tabela segmenata ima poseban ulaz za svaki segment procesa. U njima se nalaze informacije neophodne za preslikavanje segmenata virtuelnog adresnog prostora procesa u blokove fizičke memorije. Te informacije se nazivaju deskriptori segmenata. S obzirom da u virtuelnom adresnom prostoru procesa ima 2^s segmenata, to i tabela segmenata ima 2^s ulaza. Početna adresa tabele segmenata sadržana je u posebnom registru procesora koji se naziva ukazivač na ST (STP). Na osnovu sadržaja ukazivača na ST i broja segmenta, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora segmenta, a time i do informacija neophodnih za preslikavanje.

U fizičkoj memoriji postoji posebna tabela segmenata svakog procesa. Početne adrese tabela segmenata 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 ST upisuje početna adresa tabele segmenata procesa kome se dodeljuje procesor. Time se obezbeđuje da jedinica za preslikavanje pristupa tabeli segmenata procesa kome je dodeljen procesor.

Polja deskriptora segmenta su:

* *IRW* (3 bita)—bitovi prava pristupa,
* *length* (*p* bita)—veličina segmenta i
* *PTP* (*b* + *w* bita)—početna adresa tabele stranica u fizičkoj memoriji.

Polje *IRW* označava dozvoljen pristup datom segmentu. Uzeto je da

* postavljen bit *I* označava da iz njega smeju da se čitaju samo instrukcije,
* postavljen bit *R* označava da iz njega smeju da se čitaju podaci i
* postavljen bit *W* označava da iz njega smeju da se upisuju podaci.

Polje *IRW* formira operativni sistem prilikom formiranja tabele segmenata datog procesa. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem registra *accreg*, utvrdi da li zahtevani pristup segmentu odgovara dozvoljenom pristupu segmentu.

Polje *length* označava veličinu segmenta u stranicama. Polje *length* postavlja operativni sistem prilikom formiranja tabele segmenata datog procesa. Koristi ga jedinica za preslikavanje da, upoređivanjem sa sadržajem polja *page* virtuelne adrese, utvrdi da li je broj stranice u segmentu unutar specificirane veličine segmenta.

Polje *PTP* označava početnu adresu operativne memorije u kome se nalazi tabela stranica datog segmenta. Polje *PTP* postavlja operativni sistem prilikom formiranja tabele segmenata i tabela stranica. Koristi ga jedinica za preslikavanje da formira realnu adresu.

Uzeto je da se *STP* i *PTP* predstavljaju u brojevima blokova, a *length* u broju stranica.

Tabela stranica (PT) jednog segmenta je data na slici 3. Tabela stranica ima poseban ulaz za svaku stranicu datog segmenta. U njima se nalaze informacije neophodne za preslikavanje stranica segmenta u blokove fizičke memorije. Te informacije se nazivaju deskriptori stranica. S obzirom da segment ima najviše 2^p stranica, to i tabela stranica može da ima najviše 2^p ulaza. Početna adresa tabele stranica sadržana je u polju *PTP* deskriptora datog segmenta. Na osnovu sadržaja polja *PTP* deskriptora datog segmenta i broja stranice, jedinica za preslikavanje virtuelnih adresa u realne adrese dolazi do deskriptora stranice, a time i do informacija neophodnih za preslikavanje.

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:

1. Polje *segm* virtuelne adrese predstavlja broj ulaza u tabelu segmenata u kome se nalazi deskriptor datog segmenta. S obzirom na to da deskriptor segmenta zauzima 2^l reči potrebno je broj ulaza u tabelu segmenata pretvoriti u pomeraj u odnosu na početak tabele segmenata. To se realizuje pomeranjem ulevo za *l* mesta sadržaja polja *segm* vitruelne adrese.
2. Sadržaj registra ukazivač na ST predstavlja početnu adresu tabele segmenata. Pomeraj u odnosu na početak tabele segmenata formiran u prethodnom koraku se sabira sa sadržajem registra ukazivač na ST i time dobija adresa deskriptora datog segmenta. Počev od formirane adrese treba očitati potreban broj reči, jer se u njima nalaze polja *IRW*, *length* i *PTP* koja koristi jedinica za preslikavanje.
3. Polja *IRW*, *length* i *PTP* deskriptora sadrže bitove zaštite, veličinu segmenta i početnu adresu tabele stranica u fizičkoj memoriji, respektivno. Stoga se sada čita potreban broj reči da bi se do njih došlo. Sada se istovremeno:

* upoređivanjem polja *IRW* iz deskriptora segmenta i sadržaja registra *accreg* utvrđuje da li se zahteva pristup koji je dozvoljen za dati segment i
* upoređivanjem polja *length* iz deskriptora segmenta i polja *page* iz virtuelne adrese utvrđuje da li je broj stranice u segmentu unutar specificirane veličine segmenta.

Ukoliko su signali ACCESS i BOUND neaktivni, zahtevani pristup segmentu je korektan, pa se pristupa deskriptoru stranice u tabeli stranica na način prikazan počev od koraka 4. U suprotnom nema obraćanja tabeli stranica već se generiše prekid i prelazi na aktivnosti prikazane počev od koraka 8.

4. 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^h reči potrebno je broj ulaza u tabelu stranica pretvoriti u pomeraj u odnosu na početak tabele stranica. To se realizuje pomeranjem ulevo za *h* mesta sadržaja polja *page* vitruelne adrese.
5. Sadržaj polja *PTP* koji predstavlja početnu adresu tabele stranica sabira se sa pomerajem u odnosu na početak tabele stranica koji je formiran u prethodnom koraku i time se dobija adresa deskriptora date stranice. Počev od formirane adrese treba očitati potreban broj reči, jer se u njima nalaze polja *V* i *block* koja koristi jedinica za preslikavanje.
6. 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 utvrđuje se da je polje *V* postavljeno, pa se produžava sa sledećim koracima.
7. 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.

   Ovo je kraj preslikavanja virtuelne u realnu adresu za slučaj da je stranica u memoriji.
8. U slučaju generisanog prekida, zbog pojave aktivne vrednosti jednog ili oba signala **ACCESS** ili **BOUND**, sve što za ovaj slučaj potom treba uraditi, radi operativni sistem i to u sledećim koracima:

   * Oduzima procesor datom procesu, njegov kontekst čuva i terminira procesa.
   * Šalje poruku na konzolu da je dati proces terminiran.
   * Dodeljuje 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.

U slučaju da stranica nije u memoriji realizuju se svi koraci sem koraka 7 kao i za slučaj kada je stranica u memoriji. Moguće su dve situacije. U prvoj situaciji se ide od koraka 1 do koraka 3 u kome se utvrđuje da je jedan od ili oba signala **ACCESS** ili **BOUND** aktivni, pa se prelazi na korak 8. U drugoj situaciji se ide od koraka 1 do koraka 6. U koraku 6 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 operativni sistem i to 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 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 će ponovo generisati adresu za koju je bilo utvrđeno da je iz stranice koja nije bila u memoriji. Pošto je sada data stranica u memoriji ponoviće se koraci za sličaj kada je stranica u memoriji.

2.1.2. 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 2.3.4, 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.

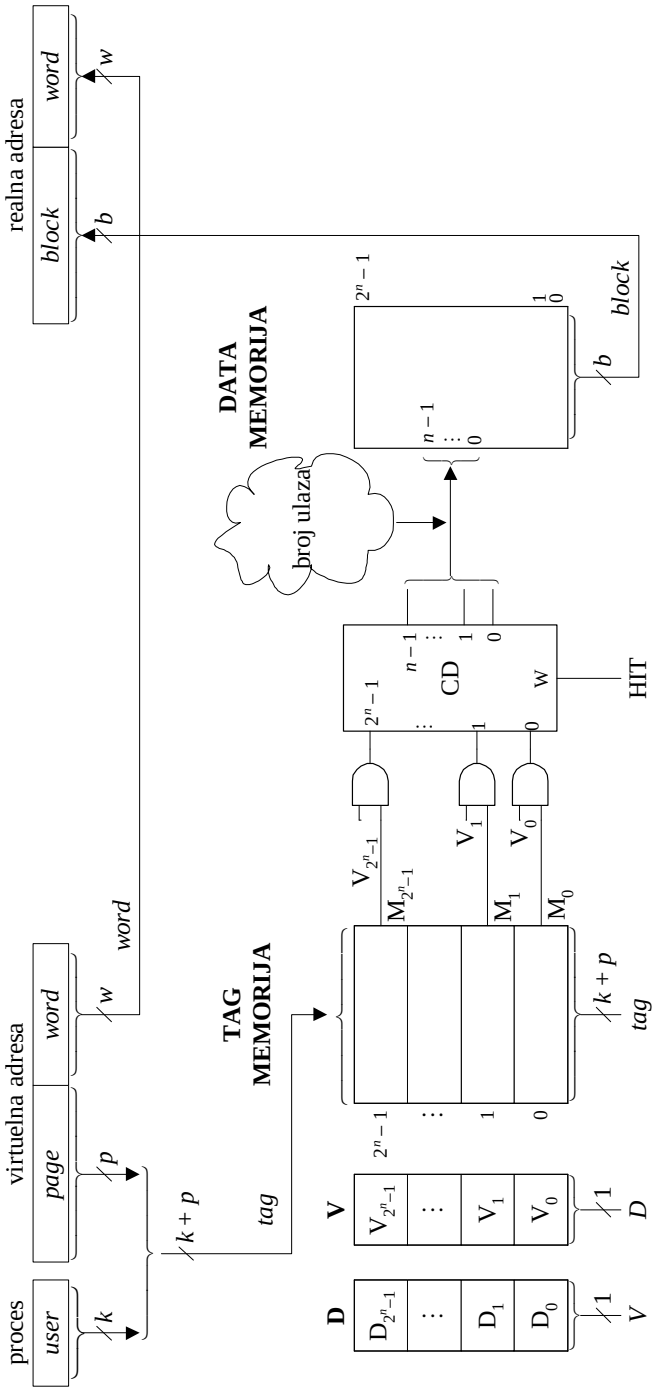
2.1.2.1. JEDINICA SA ASOCIJATIVNIM PRESLIKAVANJEM

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 4, *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 **4** 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 2.1.1.1. 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 2.1.1.1.

2.1.2.2. JEDINICA SA DIREKTNIM PRESLIKAVANJEM

S obzirom:

* da u virtuelnom adresnom prostoru procesa ima 2^s segmenata,
* da je broj procesa 2^k,
* da se u jedinici za preslikavanje mogu istovremeno čuvati deskriptori segmenata različitih procesa,

* da je jedinica za preslikavanje realizovana u tehnici direktnog preslikavanja i
* da se u jedinici za preslikavanje mogu istovremeno čuvati relevantni delovi deskriptora 2^m najčešće korišćenih segmenata do 2^k procesa,

onda je 2^s segmenata svih 2^k procesa grupisano u 2^m grupa sa po 2^{k+s-m} segmenata u grupi. Zaključuje se, kao što je prikazano na slici 5, da su:

* polje *tag* dužine $k + s – m$ bita i
* polje *ulaz* dužine m bita.

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

Jedinica za preslikavanje se sastoji iz sledećih delova:

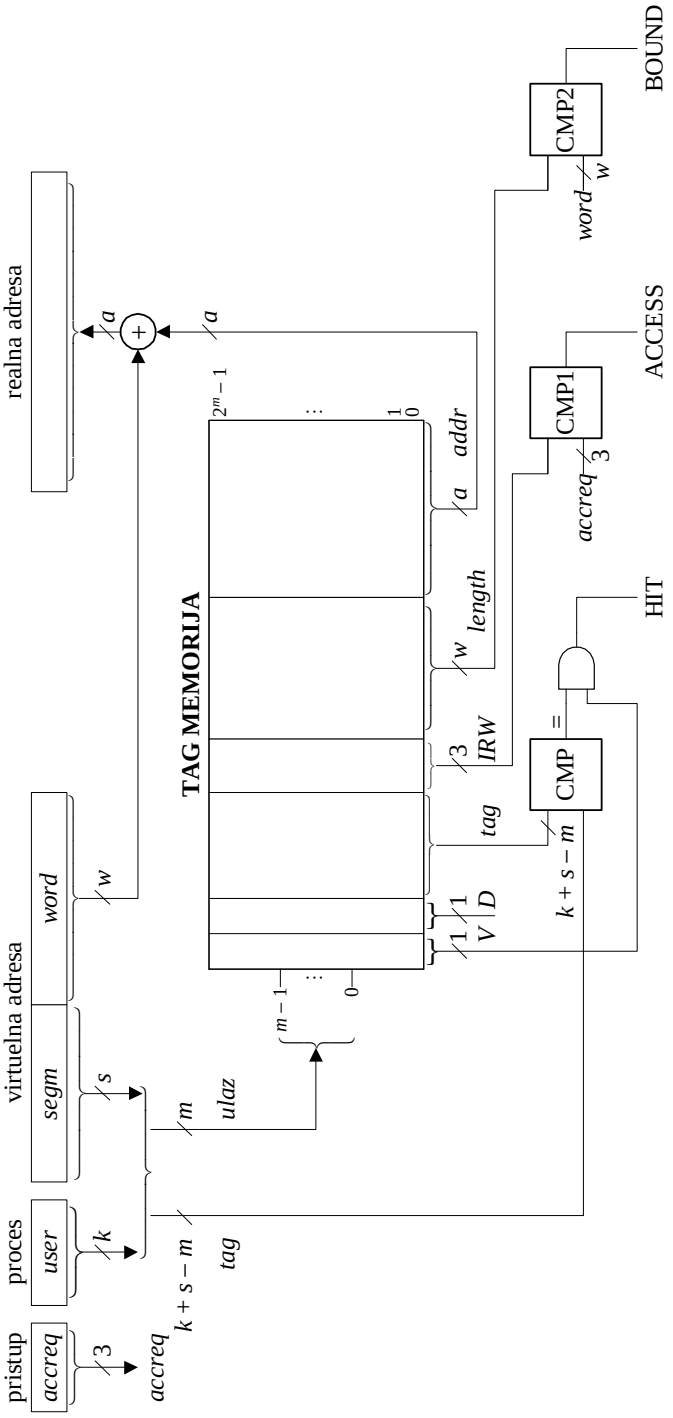
* TAG MEMORIJA—RAM memorija kapaciteta 2^m reči širine $1 + 1 + (k + s – m) + 3 + w + a$ bita, koja se sastoji iz polja:
  * *V* (valid bitovi) dužine 1 bit,
  * *D* (dirty bitovi) dužine 1 bit,
  * *tag* dužine $k + s – m$ bita,
  * *IRW* dužine 3 bita,
  * *length* dužine w bita i
  * *addr* dužine a bita,
* CMP—($k + s – m$)-bitni komparator,
* CMP1—3-bitni komparator i
* CMP2—w-bitni komparator.

TAG MEMORIJA služi za čuvanje delova *IRW*, *length* i *addr* 2^m deskriptora segmenata zajedno sa odgovarajućim *V*, *D* i *tag* bitovima. Značenje ovih bitova je sledeće:

* *V* bitovi označavaju za svaki od 2^m ulaza jedinice za preslikavanje da li je odgovarajući ulaz TAG MEMORIJE važeći,
* *D* bitovi označavaju za svaki od 2^m ulaza jedinice za preslikavanje da li je bilo upisa u neku od lokacija datog segmenta,
* *tag* bitovi služe za čuvanje 2^m *tag* polja segmenata čiji se delovi deskriptora nalaze u poljima *IRW*, *length* i *addr* TAG MEMORIJE,
* *IRW* bitovi označavaju dozvoljen pristup datom segmentu,
* *length* bitovi označavaju dužinu segmenta i
* *addr* bitovi označavaju početnu adresu dela operativne memorije u kome se nalazi dati segment.

CMP služi za generisanje aktivne vrednosti signala saglasnosti **HIT** ukoliko:

* *tag* polje generisane adrese je isto kao sadržaj *tag* polja TAG MEMORIJE adresiran poljem *ulaz* generisane adrese i
* *V* bit adresiran poljem *ulaz* generisane adrese je postavljen.



Slika **5** Jedinica sa direktnim preslikavanjem za virtuelnu memoriju segmentne organizacije

CMP1 služi za generisanje aktivne vrednosti signala **ACCESS** ukoliko je sadržaj registara *accreg* isti kao sadržaj polja *IRW* TAG MEMORIJE adresiran poljem *ulaz* generisane adrese.

CMP2 služi za generisanje aktivne vrednosti signala **BOUND** ukoliko je sadržaj polja *word* virtuelne adrese manji od sadržaja polja *length* TAG MEMORIJE adresiran poljem *ulaz* generisane adrese.

Bitovi *ulaz* iz generisane adrese se koriste kao adresa za TAG MEMORIJU. Bitovi *tag* očitani iz TAG MEMORIJE se porede sa bitovima *tag* iz generisane adrese. Ako se otkrije da postoji saglasnost i da je *V* bit, adresiran bitovima *ulaz* generisane adrese, postavljen, signal saglasnosti **HIT** postaje aktivan.

Očitano polje *addr* i polje *word* iz virtuelne adrese se sabiraju i daju *a* bitova realne adrese.

Sve ove aktivnosti se realizuju hardverski.

Da bi se segmenti različitih procesa koji imaju isti broj preslikali svaki u svoj deo memorije, u formiranju *tag* polja učestvuje ne samo polje *segm* 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 delovi deskriptora ne nalaze u jedinici za preslikavanje, ide se hardverski u tabelu segmenata i čita deskriptor na način prikazan u odeljku 2.1.1.2. Ukoliko se utvrdi da se segment nalazi u operativnoj memoriji, polja *IRW*, *length* i *addr* deskriptora se dovlače u jedinicu za preslikavanje. U ulaz DATA MEMORIJE određen *ulaz* bitovima virtuelne adrese se upisuju polja *IRW*, *length* i *addr* deskriptora. U isti ulaz TAG MEMORIJE se upisuje polje *tag* generisane adrese. U isti ulaz *V* bitova se upisuje 1, a u isti ulaz *D* bitova se upisuje 0. Sve ovo se realizuje hardverski. Ukoliko se utvrdi da se segment ne nalazi u operativnoj memoriji, generiše se prekid. Operativni sistem dovlači dati segment sa diska na način prikazan u zadatku 2.1.1.2.

2.1.2.3. JEDINICA SA SET-ASOCIJATIVNIM PRESLIKAVANJEM

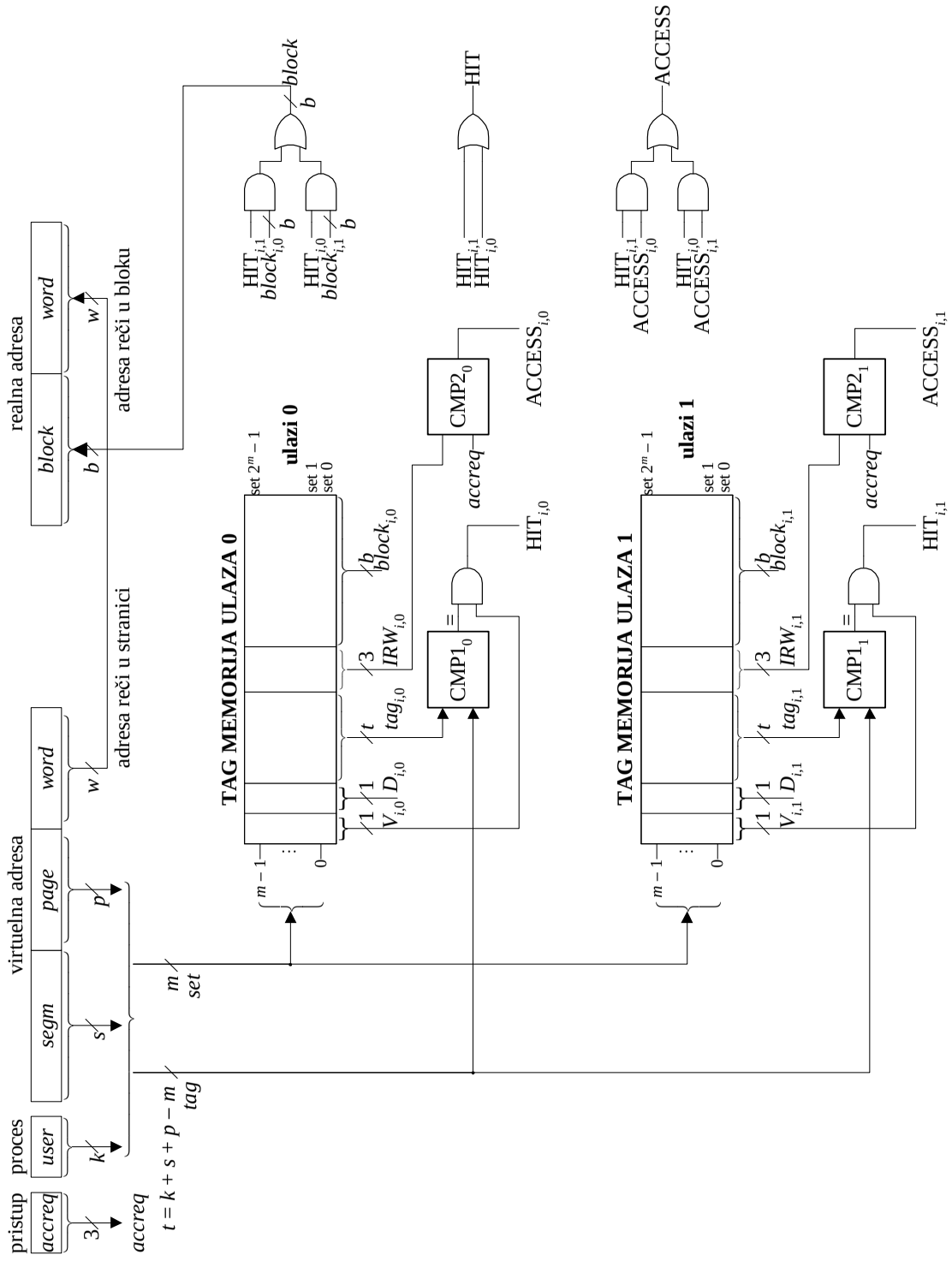
Jedinica sa set-asocijativnim preslikavanjem se razmatra za slučaj da postoje dva ulaza po setu, jer je to slučaj koji se najčešće sreće u praktičnim realizacijama.

S obzirom da u adresnom prostoru procesa ima 2^{s+p} stranica, da je broj procesa 2^k, da se u jedinici za preslikavanje mogu istovremeno čuvati deskriptori stranica različitih procesa, da je broj setova u jedinici za preslikavanje 2^m i da postoje dva ulaza po setu (2^1 ulaza), onda je 2^{s+p} stranica svih 2^k procesa grupisano u $2^{k+s+p-m}$ grupa sa po 2^m stranica u grupi. Zaključuje se da je polje TAG dužine $k + s + p - m$ bita i da je polje *set* dužine *m* bita kao što je prikazano na slici 6.

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

Jedinica za preslikavanje se sastoji iz sledećih delova:

* TAG MEMORIJA ULAZA 0—RAM memorija kapaciteta 2^m reči širine $1 + 1 + (k + s + p - m) + 3 + b$ bita, koja se sastoji iz polja:
  * *D* (dirty bitovi) dužine 1 bit,
  * *V* (valid bitovi) dužine 1 bit,
  * *tag* dužine $k + s + p - m$ bita,
  * *block* dužine *m* bita i
* TAG MEMORIJA ULAZA 1—RAM memorija kapaciteta 2^m reči širine $1 + 1 + (k + s + p - m) + 3 + b$ bita, koja se sastoji iz istih polja kao i TAG MEMORIJA ULAZA 0.
* CMP_0—komparator TAG MEMORIJE ULAZA 0.
* CMP_1—komparator TAG MEMORIJE ULAZA 1.



Slika **6** Jedinica sa set-asocijativnim preslikavanjem za virtuelnu memoriju segmentno-stranične organizacije

TAG MEMORIJA ULAZA 0 služi za čuvanje 2^m *block* polja deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima setova SET_0 do SET_{2^m-1} smeštenih u ulaze 0 jedinice za preslikavanje.

Bitovi *D* označavaju za svaki od 2^m setova ulaza 0 jedinice za preslikavanje da li je bilo upisa u neku od lokacija date stranice.

Bitovi *V* označavaju za svaki od 2^m setova ulaza 0 jedinice za preslikavanje da li je odgovarajući ulaz TAG MEMORIJE ULAZA 0 važeći.

Bitovi *tag* služe za čuvanje 2^m *tag* polja stranica čija se *block* polja deskriptora nalaze u odgovarajućim *block* poljima TAG MEMORIJE ULAZA 0.

Bitovi *block* služe za čuvanje 2^m *block* polja deskriptora stranica koja se nalaze u ulazima 0 jedinice za preslikavanje.

CMP_0 služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_{i,0}}$ za ulaze 0 jedinice za preslikavanje ukoliko:

* *tag* polje generisane adrese je isto kao sadržaj $tag_{i,0}$ polja TAG MEMORIJE ULAZA 0 adresiran poljem *set* generisane adrese i
* $V_{i,0}$ bit adresiran poljem *set* generisane adrese je postavljen.

TAG MEMORIJA ULAZA 1 služi za čuvanje 2^m *block* polja deskriptora stranica zajedno sa odgovarajućim *V*, *D* i *tag* bitovima setova SET_0 do SET_{2^m-1} smeštenih u ulaze 1 jedinice za preslikavanje. Funkcije polja *V*, *D*, *tag* i *block* su iste kao u slučaju TAG MEMORIJE ULAZA 0.

CMP_1 služi za generisanje aktivne vrednosti signala saglasnosti $\mathbf{HIT_{i,1}}$ za ulaze 1 jedinice za preslikavanje na isti način kao signal saglasnosti $\mathbf{HIT_{i,0}}$.

Bitovi *set* iz generisane adrese se koriste kao adresa za TAG MEMORIJU ULAZA 0 i TAG MEMORIJU ULAZA 1. Bitovi $tag_{i,0}$ očitani iz TAG MEMORIJE ULAZA 0 se porede sa bitovima *tag* iz generisane adrese. Ako se otkrije da postoji saglasnost i da je bit $V_{i,0}$, adresiran bitovima *set* generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT_{i,0}}$ postaje aktivan. Bitovi $tag_{i,1}$ očitani iz TAG MEMORIJE ULAZA 1 se porede sa bitovima *tag* iz generisane adrese. Ako se otkrije da postoji saglasnost i da je bit $V_{i,1}$, adresiran bitovima *set* generisane adrese, postavljen, signal saglasnosti $\mathbf{HIT_{i,1}}$ postaje aktivan. Signal saglasnosti **HIT** je aktivan ukoliko je aktivan jedan od signala $\mathbf{HIT_{i,0}}$ i $\mathbf{HIT_{i,1}}$.

U formiranju *b* najstarijih bitova realne adrese učestvuju bitovi $block_{i,0}$ iz TAG MEMORIJE ULAZA 0 ukoliko je signal $\mathbf{HIT_{i,0}}$ aktivan, odnosno bitovi $block_{i,1}$ iz TAG MEMORIJE ULAZA 1 ukoliko je signal $\mathbf{HIT_{i,1}}$ aktivan. Polje *word* iz virtuelne adrese daje *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 polja *tag* učestvuje 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 2.1.2.3. Ukoliko se utvrdi da se stranica nalazi u operativnoj memoriji, polje *block* deskriptora se dovlači u hardver. Polje *set* generisane adrese određuje u koji od 2^m seta se smešta polje *block* deskriptora. Pošto za svaki set postoje ulazi 0 i 1, korišćenjem FIFO ili LRU algoritma zamene u okviru datog seta bira se ulaz jedinice za preslikavanje u koji se smešta deskriptor. U polje *block* datog ulaza jedinice za preslikavanje se upisuje polje *block* deskriptora. U isti ulaz se upisuje polje *tag* generisane adrese. U isti ulaz *V* bitova se upisuje 1, a u isti ulaz *D* bitova 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 2.1.2.3.

2.2. STRANIČNA ORGANIZACIJA VIRTUELNE MEMORIJE

U ovoj glavi se daju opšte napomene o usvojenoj organizaciji virtuelnoj memorije stranične organizacije i realizacija sistema preslikavanja.

2.2.1. OPŠTE NAPOMENE

U ovom odeljku se daju opšte napomene o usvojenoj virtuelnoj memorije stranične organizacije. U okviru toga se daju karakteristike virtuelnog adresnog prostora procesa, fizičkog adresnog prostora i struktura tabele preslikavanja virtuelnog adresnog prostora u fizički adresni prostor.

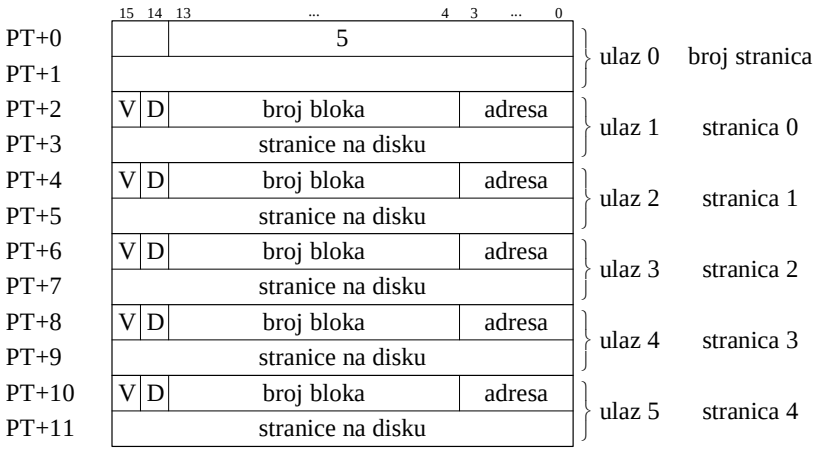
Adresabilna jedinica u virtuelnom adresnom prostoru i fizičkom adresnom prostoru je reč dužine dva bajta (1W=2B). Veličina virtuelnog adresnog prostora je 16M 16-bitnih reči, a veličina jedne stranice je 1KW. Shodno tome virtuelna adresa je dužine 24 bita i ima sledeću strukturu:

- viših m = 14 bitova ozanačavaju broj stranice i
- nižih k = 10 bitova označavaju adresu unutar stranice.

Veličina fizičkog adresnog prostora je 1MW 16-bitnih reči, a veličina jednog bloka je 1KW. Shodno tome fizička adresa je dužine 20 bita i ima sledeću strukturu:

- viših b=10 bitova označavaju broj bloka i
- nižih k=10 bitova označavaju adresu unutar bloka.

Preslikavanje virtuelnih adresa u fizičke adrese se realizuje delom softverski i delom hardverski. Softverski deo preslikavanja realizuje operativni sistem, dok hardverski deo preslikavanja realizuje jedinica za preslikavanje TLB (*Translation Look-aside Buffer*). Preslikavanje se realizuje na osnovu tabele preslikavanja stranica PT (*Page map Table*). PT se nalazi u operativnoj memoriji u delu odvojenom za operativni sistem. Usvojeno je da se PT sastoji od ulaza sa po dve reči (slika 7). U nultom ulazu se nalazi broj stranica virtuelnog adresnog prostora procesa, pri čemu se koriste samo bitovi 13 do 0 niže reči, dok se preostali bitovi niže reči i bitovi više reči ne koriste. U nenultim ulazima se nalaze deskriptori stranica. U prvom ulazu PT se nalazi deskriptor nulte stranice, u drugom deskriptor prve stranice itd. U nižoj reči nenultog ulaza se nalazi prva reč deskriptora, a u višoj druga. Bitovi 15 i 14 niže reči deskriptora predstavljaju V indikator (*Valid*) i D indikator (*Dirty*), respektivno. V indikator vrednostima 1 i 0 označava da li je stranica u operativnoj memoriji ili nije u operativnoj memoriji. D indikator vrednostima 1 i 0 označava da li je bilo upisa u stranicu ili nije bilo upisa u stranicu. Bitovi 13 do 4 niže reči deskriptora označavaju broj bloka operativne memorije u koji je operativni sistem smestio stranicu sa diska i predstavljaju važeći sadržaj onda kada je stranica dovučena sa diska u operativnu memoriju. Preostali bitovi niže reči i svi bitovi više reči predstavljaju adresu stranice na disku. Ovim se obezbeđuje da PT ima onoliko ulaza koliko ima stranica u virtuelnom adresnom prostoru procesa plus nulti ulaz. Izgled PT za primer programa od pet stranica prikazan je na slici 2.1.



Slika 7 Tabela stranica PT za proces od pet stranica

Prilikom kreiranja procesa operativni sistem, između ostalog, formira i tabelu stranica PT procesa sa onoliko ulaza koliko dati proces ima stranica plus jedan. U nulti ulaz se popunjava broj koji ukazuje koliko stranica ima dati proces. U ostale ulaze se za svaku stranicu popunjava njena adresa na disku, a za sve stranice se V i D indikatori postavljaju na 0. Početna adresa tabele stranica se čuva u deskriptoru procesa (*process control block*).

Preslikavanje virtuelnih u fizičke adrese se realizuje na osnovu sadržaja jedinice za preslikavanje u koju se smeštaju deskriptori najčešće korišćenih stranica različitih procesa. Pri preslikavanju za operaciju upisa jedinica za preslikavanje, između ostalog, u tabeli stranica postavlja na 1 indikator D u deskriptoru date stranice.

Deskriptore dovlači jedinica za preslikavanje pri pokušaju preslikavanja stranice za koju se deskriptor ne nalazi u jedinici za preslikavanje. Pri svakom dovlačenju deskriptora stranice iz tabele stranica, najpre se čita nulti ulaz tabele stranica da bi se izvršila proveru da li data stranica postoji u virtuelnom adresnom prostoru procesa. Ukoliko ne postoji, generiše se prekid zbog pristupa nepostojećoj stranici AV (*access violation*).

Ukoliko postoji, na osnovu adrese tabele stranica i broja stranice virtuelne adrese pristupa se deskriptoru stranice u tabeli stranica. Ukoliko je bit V nula, stranica nije u operativnoj memoriji, pa se generiše prekid zbog pristupa stranici koja nije u operativnoj memoriji PF (*page fault*). Kao reakcija na ovaj prekid, operativni sistem dovlači zahtevanu stranicu u neki blok operativne memorije. Tom prilikom mogu da se jave dve situacije. Prva je da u oepративnoj memoriji ima slobodnih blokova. U tom slučaju operativni sistem po nekom kriterijumu uzima jedan od tih blokova, u njega dovlači stranicu sa diska i u tabeli stranica datog procesa popunjava ulaz date stranice tako što upisuje broj bloka u koji je stranica dovučena i postavlja bit V na jedan. Druga je da u operativnoj memoriji nema slobodnih blokova. U tom slučaju operativni sistem po nekom kriterijumu nekoj stranici nekog od procesa oduzima blok. Tom prilikom se bit V deskriptora stranice kojoj se oduzima blok postavlja na 0 i proverava bit D. Ukoliko je bit D nula, nije bilo upisa u datu stranicu i nema potrebe da se data stranica vraća na disk. Ukoliko je bit D jedan, bilo je upisa u datu stranicu i data stranica mora prvo da se vrati na disk.

Ukoliko je bit V jedan, stranica je u operativnom memoriji, pa se broj bloka i bitovi V i D dovlače u neki od ulaza jedinice za preslikavanje.

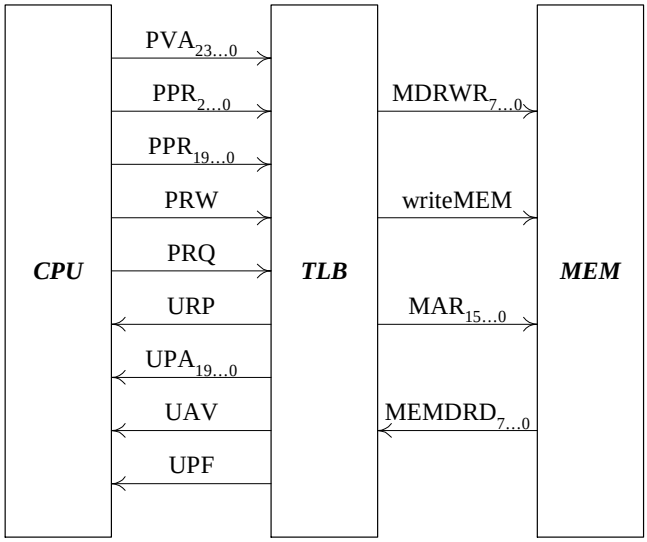
2.2.2. SISTEM ZA PRESLIKAVANJE

U ovom odeljku se daje realizacija sistema za preslikavanje virtuelnog u fizički adresni prostor. Sistem koji se posmatra (slika 8) se sastoji iz:

- procesora ***CPU***,
- operativne memorije ***MEM*** i
- jedinice za preslikavanje ***TLB***.

Uzeto je da ceo sistem radi sinhrono sa zajedničkim signalom takta.

U daljem tekstu se najpre razmatraju samo delovi procesora ***CPU*** bitni za njegov rad sa jedinicom za preslikavanje ***TLB***, zatim se daje realizacija operativne memorije ***MEM*** i na kraju prikazaje realizacija jedinice za preslikavanje ***TLB***.



Slika 8 Struktura sistema

2.2.2.1. PROCESOR *CPU*

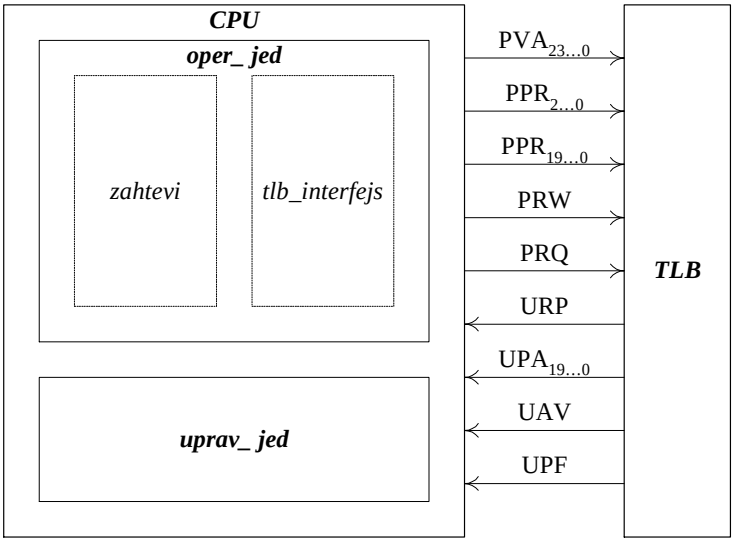
Procesor ***CPU*** (slika 9) se obraća jedinici ***TLB*** sa zahtevom za preslikavanja virtuelne adrese u fizičku adresu pre svakog čitanja podatka iz operativne memorije i pre svakog upisa podatka u operativnu memoriju. Procesor ***CPU*** šalje jedinici ***TLB*** 24-bitnu virtuelnu adresu po linijama $\mathbf{PVA_{23...0}}$, broj tekućeg procesa po linijama $\mathbf{PPR_{2...0}}$, adresu tabele stranica tekućeg procesa po linijama $\mathbf{PPT_{19...0}}$, indikator operacije upisa ili čitanja po liniji **PRW** i aktivnu vrednost signala **PRQ** trajanja jedna perioda signala takta kojim se u jedinici ***TLB*** startuje peslikavanje. Preslikavanje može da bude realizovano uspešno ili neuspešno. U slučaju uspešnog preslikavanja jedinica ***TLB*** vraća procesoru ***CPU*** 20-bitnu fizičku adresu po linijama $\mathbf{UPA_{19...0}}$ i aktivnu vrednost signala **URP** trajanja jedna perioda signala takta. U slučaju neuspešnog preslikavanja, koje može da nastane zbog toga što je jedinica ***TLB*** otkrila ili pokušaj pristupa nepostojećoj stranici (*access violation*) ili pokušaj pristupa stranici koja nije u operativnoj memoriji (*page fault*), jedinica ***TLB*** vraća procesoru ***CPU*** aktivnu vrednost ili signala UAV (*unit access violation*) ili signala UPF (*unit page fault*), respektivno, trajanja jedna perioda signala takta.

Procesor ***CPU*** (slika 9) se sastoji iz:

- operacione jedinice ***oper_jed*** i
- upravljačke jedinice ***uprav_jed***.

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija na osnovu upravljačkih signala koji dolaze iz upravljačke jedinice i generisanje signala logičkih uslova koji se vode u upravljačku jedinicu. Upravljačka jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala na osnovu algoritma generisanja upravljačkih signala zahteva za preslikavanje procesora i signala logičkih uslova.

Struktura i opis operacione jedinice i upravljačke jedinice se daju u daljem tekstu.



Slika 9 Procesor ***CPU***

2.2.2.1.1. Operaciona jedinica

Operaciona jedinica ***oper_jed*** (slika 9) se sastoji iz sledećih blokova:

- blok *tlb_ interfejs* i
- blok *zahtevi*.

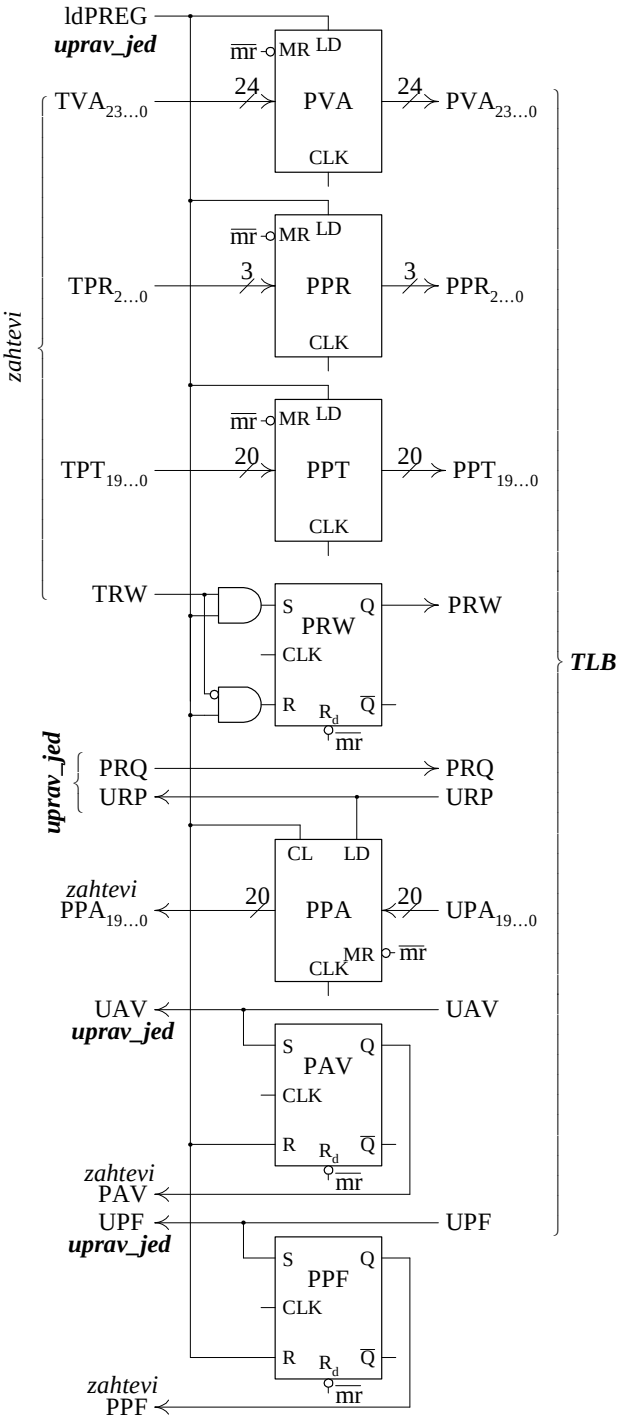
Blok *tlb_ interfejs* služi za povezivanje procesora ***CPU*** i jedinice ***TLB***. Blok *zahtevi* služi za generisanje zahteva za preslikavanje virtuelnih adresa za operacije čitanja i upisa. Struktura i opis blokova operacione jedinice ***oper_jed*** se daju u daljem tekstu.

2.2.2.1.1.1. Blok *tlb_ interfejs*

Blok *tlb_ interfejs* (slika 10) sadrži registre PVA, PPR, PPT i PPA i flip-flopove PRW, PAV i PPF.

Registar $PVA_{23...0}$ (*Processor Virtual Address register*) služi za čuvanje virtuelne adrese koju jedinica ***TLB*** treba da preslika u fizičku adresu. Virtuelna adresa dolazi na ulaze registra $PVA_{23...0}$ po linijama $\mathbf{TVA_{23...0}}$ iz bloka *zahtevi*, a u registar $PVA_{23...0}$ se upisuje pri aktivnoj vrednosti signala **ldPREG** trajanja jedna perioda signala takta. Sadržaj registra $PVA_{23...0}$ se po linijama $\mathbf{PVA_{23...0}}$ vodi u blok *cpu_interfejs* jedinice ***TLB***.

Registar $PPR_{2...0}$ (*Processor PRocess register*) služi za čuvanje broja tekućeg procesa. Broj tekućeg procesa dolazi na ulaze registra $PPR_{2...0}$ po linijama $\mathbf{TPR_{2...0}}$ iz bloka *zahtevi*, a u registar $PPR_{2...0}$ se upisuje pri aktivnoj vrednosti signala **ldPREG** trajanja jedna perioda signala takta. Sadržaj registra $PPR_{2...0}$ se po linijama linije $\mathbf{PPR_{2...0}}$ vodi u blok *cpu_interfejs* jedinice ***TLB***.



Slika 10 Blok tlb_interfejs

Registar $PPT_{19...0}$ (*Processor Page map Table address register*) služi za čuvanje početne adrese tabele stranica tekućeg procesa. Adresa dolazi na ulaze registra $PPT_{19...0}$ po linijama $TPT_{19...0}$ iz bloka *zahtevi*, a u registar $PPT_{19...0}$ se upisuje pri aktivnoj vrednosti signala **ldPREG** trajanja jedna perioda signala takta. Sadržaj registra $PPT_{19...0}$ se po linijama $\mathbf{PPT_{19...0}}$ vodi u blok *cpu_interfejs* jedinice ***TLB***.

Registar $PPA_{19...0}$ (*Processor Physical Address register*) služi za čuvanje fizičke adrese koja je generisana u jedinici ***TLB*** u slučaju uspešnog preslikavanja virtuelne adrese u fizičku adresu. Fizička adresa dolazi na ulaze registra $PPA_{19...0}$ po linijama $\mathbf{UPA_{19...0}}$ iz bloka *cpu_interfejs* jedinice ***TLB***, a u registar $PPA_{19...0}$ se upisuje pri aktivnoj vrednosti signala **URP** trajanja jedna perioda signala takta koji dolazi iz bloka *cpu_interfejs* jedinice ***TLB***. Aktivnom vrednošću signala **ldPREG** trajanja jedna perioda signala takta se na početku svakog novog zahteva za preslikavanje u sve razrede registra $PPA_{19...0}$ upisuje 0. Zbog toga će na kraju svakog zahteva za preslikavanje u registru $PPA_{19...0}$, u zavisnosti od toga da li je preslikavanje realizovano uspešno ili neuspešno, biti ili fizička adresa ili nule. Sadržaj registra $PPA_{19...0}$ se po linijama $\mathbf{PPA_{19...0}}$ vodi u blok *zahtevi*.

Flip-flop PRW (*Processor Read Write flag*) vrednostima 0 i 1 predstavlja indikator operacije čitanja ili upisa, respektivno, koja treba da se realizuje po preslikavanju virtuelne adrese u fizičku adresu. Vrednost indikatora dolazi na ulaze flip-flopa PRW po liniji **TRW** iz bloka *zahtevi*, a u flip-flopa PRW se upisuje pri aktivnoj vrednosti signala **ldPREG** trajanja jedna perioda signala takta. Sadržaj flip-flopa PRW se po liniji **PRW** vodi u blok *cpu_interfejs* jedinice ***TLB***.

Flip-flop PAV (*Procesor Address Violation flag*) vrednostima 0 i 1 predstavlja indikator da li je jedinica ***TLB*** uspešno realizovala preslikavanje ili je to uradila neuspešno jer je otkrila pokušaj pristupa nepostojećoj stranici (*access violation*), respektivno. Aktivnom vrednošću signala **ldPREG** trajanja jedna perioda signala takta se na početku svakog novog zahteva za preslikavanje u flip-flop PAV upisuje 0. Aktivnom vrednošću signala **UAV** trajanja jedna perioda signala takta, koji dolazi iz bloka *cpu_interfejs* jedinice ***TLB*** na kraju svakog neuspešnog preslikavanja zbog pokušaja pristupa nepostojećoj stranici (*access violation*), u flip-flop PAV se upisuje 1. Na kraju svakog preslikavanja u flip-flopu PAV će, u zavisnosti od toga da li je preslikavanje realizovano uspešno ili neuspešno zbog pokušaja pristupa nepostojećoj stranici (*access violation*), biti 0 ili 1, respektivno. Sadržaj flip-flopa PAV se po liniji **PAV** vodi u blok *zahtevi*.

Flip-flop PPF (*Procesor Page Fault flag*) vrednostima 0 i 1 predstavlja indikator da li je jedinica ***TLB*** uspešno realizovala preslikavanje ili je to uradila neuspešno jer je otkrila pokušaj pristupa stranici koja nije u operativnoj memoriji (*page fault*), respektivno. Aktivnom vrednošću signala **ldPREG** trajanja jedna perioda signala takta se na početku svakog novog zahteva za preslikavanje u flip-flop PPF upisuje 0. Aktivnom vrednošću signala **UPF** trajanja jedna perioda signala takta, koji dolazi iz bloka *cpu_interfejs* jedinice ***TLB*** na kraju svakog neuspešnog preslikavanja zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*), u flip-flop PPF se upisuje 1. Na kraju svakog preslikavanja u flip-flopu PPF će, u zavisnosti od toga da li je preslikavanje realizovano uspešno ili neuspešno zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*), biti 0 ili 1, respektivno. Sadržaj flip-flopa PPF se po liniji **PPF** vodi u blok *zahtevi*.

Upravljački signal **PRQ** (*Processor ReQuest*) se koristi da procesor ***CPU*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira jedinici ***TLB*** da se na linijama $\mathbf{PVA_{23...0}}$, $\mathbf{PPR_{2...0}}$, $\mathbf{PPT_{19...0}}$ i **PRW** nalaze sadržaji potrebni za preslikavanja virtuelne adrese u fizičku adresu i da jedinica ***TLB*** treba ove sadržaje da upiše u svoje odgavarajuće registre i startuje preslikavanje.

Upravljački signal **URP** (*Unit RePly*) koji dolazi iz bloka *cpu_interfejs* jedinice ***TLB*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da je peslikavanje uspešno realizovano i da se na linijama $\mathbf{UPA_{19...0}}$ nalazi generisana fizička adresa.

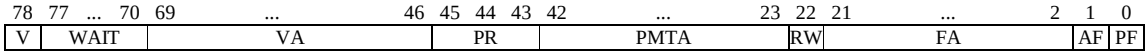
Upravljački signal **UAV** (*Unit Address Violation*) koji dolazi iz bloka *cpu_interfejs* jedinice ***TLB*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa nepostojećoj stranici (*access violation*).

Upravljački signal **UPF** (*Unit Page Fault*) koji dolazi iz bloka *cpu_interfejs* jedinice ***TLB*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*).

2.2.2.1.1.2. Blok *zahtevi*

Blok *zahtevi* (slika 12) sadrži memoriju TAB, brojače ECNT i WCNT, registar TMP i flip-flop END.

Memorija TAB se koristi za generisanje zahteva za preslikavanje. Jedan ulaz memorije TAB (slika 11) sadrži informacije na osnovu kojih se generiše jedan zahtev za preslikavanje. Memorija TAB ima 32 ulaza, pa najviše 32 zahteva za preslikavanje mogu da budu generisana.



Slika 11 Struktura jednog ulaza u memoriji TAB

Polje V (*Valid*) označava da li je odgovarajući ulaz memorije TAB važeći. Procesor ***CPU*** kreće sa generisanjem zahteva za preslikavanje od ulaza 0 memorije TAB i zaustavlja se ili na onom ulazu na kome otkrije da bit V ima vrednost nula ili po generisanju zahteva za ulaz 31. Bit polja V se pojavljuje na liniji $\mathbf{DO_{78}}$ i označen je kao signal **TV**.

Polje WAIT (*WAIT between requests*) određuje koliko perioda signala takta nakon što je iz memorije TAB očitan važeći ulaz treba da se sačeka pre nego što se pređe na generisanje novog zahteva za preslikavanje. Bitovi polja WAIT se pojavljuju na linijama $\mathbf{DO_{77...70}}$ i označeni su kao signali $\mathbf{TWAIT_{7...0}}$.

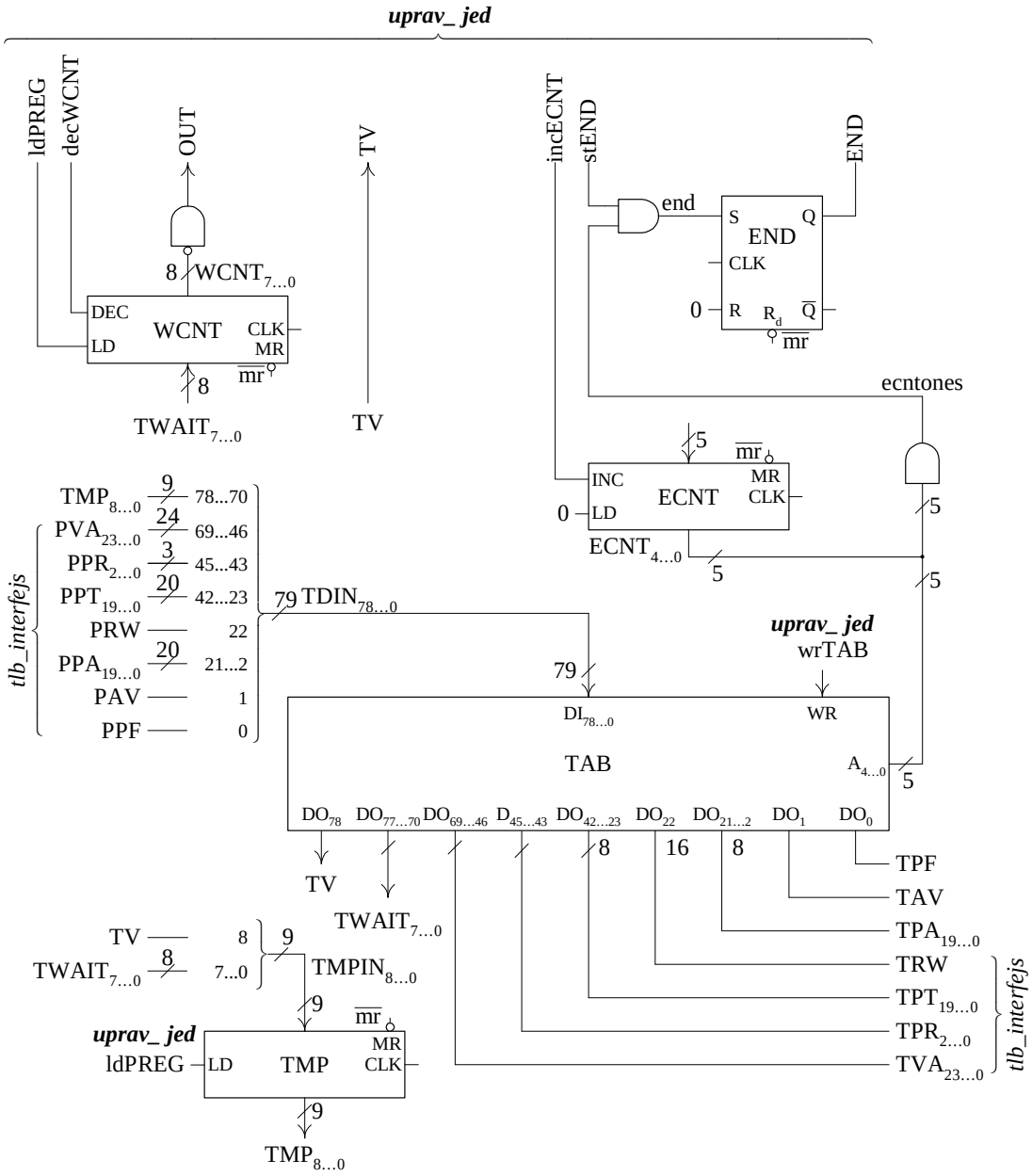
Polje VA (*Virtual Address*) sadrži virtuelnu adresu za koju preslikavanje u fizičku adresu treba da se realizuje. Bitovi polja VA se pojavljuju na linijama $\mathbf{DO_{69...46}}$ i označeni su kao signali $\mathbf{TVA_{23...0}}$.

Polje PR (*PRocess number*) sadrži broj procesa virtuelne adrese. Bitovi polja PR se pojavljuju na linijama $\mathbf{DO_{45...43}}$ i označeni su kao signali $\mathbf{TPR_{2...0}}$.

Polje PT (*Page map Table address*) sadrži početnu adresu tabele preslikavanja stranica tekućeg procesa. Bitovi polja PT se pojavljuju na linijama $\mathbf{DO_{42...23}}$ i označeni su kao signali $\mathbf{TPT_{19...0}}$.

Polje RW (*Read Write*) označava da li po preslikavanju virtuelne u fizičku adresu treba da se realizuje čitanje iz memorije ili upis u memoriju. Bit polja RW se pojavljuje na liniji $\mathbf{DO_{22}}$ i označen je kao signal **TRW**.

Polje PA (*Physical Address*) sadrži fizičku adresu u koju je preslikavana virtuelna adresa. Bitovi polja PA se pojavljuju na linijama $\mathbf{DO_{21...2}}$ i označeni su kao signali $\mathbf{TPA_{19...0}}$.



Slika 12 Blok *zahtevi*

Polje AV (*Access Violation*) sadrži indikator koji vrednošću 1 označava da peslikavanje nije uspešno realizovano zbog pokušaja pristupa nepostojećoj stranici (*access violation*). Polje AV vrednošću 0 označava da prilikom peslikavanja nije bilo pokušaja pristupa nepostojećoj stranici (*access violation*). Bit polja AV se pojavljuje na liniji **DO_1** i označen je kao signal **TAV**.

Polje PF (*Page Fault*) sadrži indikator koji vrednošću 1 označava da peslikavanje nije uspešno realizovano zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*). Polje AV vrednošću 0 označava da prilikom peslikavanja nije bilo pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*). Bit polja PF se pojavljuje na liniji **DO_0** i označen je kao signal **TPF**.

Pretpostavlja se da je na neki način, koji nije predmet ovih razmatranja, u ulaze memorije TAB smešten željeni sadržaj na osnovu koga se generišu zahtevi za preslikavanje.

Čitanje iz memorije TAB i upis u memoriju TAB se realizuje vrednostima 0 i 1, respektivno, signala **wrTAB** na upravljačkoj liniji **WR** memorije TAB. Lokacija memorije TAB sa koje se čita ili u koju se upisuje određena je vrednostima signala $\mathbf{ECNT_{4...0}}$ prisutnim na adresnim linijama $\mathbf{A_{4...0}}$ memorije TAB. Pročitani sadržaj, koji se pojavljuje na izlaznim linijama podataka $\mathbf{DO_{78...0}}$ memorije TAB, grupisan je po poljima ulaza memorije TAB u grupe linija **TV**, $\mathbf{TWAIT_{4...0}}$, $\mathbf{TVA_{23...0}}$, $\mathbf{TPR_{2...0}}$, $\mathbf{TPT_{19...0}}$, **TRW**, $\mathbf{TPA_{19...0}}$, **TVA** i **TPF**. Na početku generisanja zahteva za preslikavanje se, pri aktivnoj vrednosti signala **ldPREG** trajanja jedna perioda signala takta, sadržaj sa linija **TV** i $\mathbf{TWAIT_{4...0}}$ upisuje u registar $TMP_{8...0}$, sadržaj sa linija $\mathbf{TVA_{23...0}}$, $\mathbf{TPR_{2...0}}$, $\mathbf{TPT_{19...0}}$ i **TRW**, upisuje u registre $PVA_{23...0}$, $PPR_{2...0}$ i $PPT_{19...0}$ i flip-flop TRW bloka *tlb_interfejs*, dok se sadržaj sa linija $\mathbf{TPA_{19...0}}$, **TVA** i **TPF** tada ne koristi. Na kraju generisanja zahteva za preslikavanje se, pri aktivnoj vrednosti signala **wrTAB** trajanja jedna perioda signala takta, u ulaz memorije TAB, određen vrednostima signala $\mathbf{ECNT_{4...0}}$ na adresnim linijama $\mathbf{A_{4...0}}$ memorije TAB, upisuje sadržaj sa linija $\mathbf{TDIN_{78...0}}$ prisutan na ulaznim linijama podataka $\mathbf{DI_{78...0}}$ memorije TAB. Sadržaj na linijama $\mathbf{TDIN_{78...0}}$ se formira od sadržaja registra $TMP_{8...0}$ i registara $PVA_{23...0}$, $PPR_{2...0}$, $PPT_{19...0}$, flip-flopa PRW, registra $PPA_{19...0}$ i flip-flopova PAV i PPF bloka *tlb_interfejs*, respektivno. Ovim se obezbeđuje da se na kraju generisanja zahteva u ulazu memorije TAB, na osnovu čijeg je sadržaja generisan zahtev za preslikavanje, nađe i vrednost preslikane fizičke adrese i vrednosti indikatora koji ukazuju da li je zahtevano preslikavanje nepostojeće stranice ili stranice koja nije u memoriji.

Registar $TMP_{8...0}$ (*TeMPorary*) služi za čuvanje sadržaja iz polja V i WAIT ulaza memorije TAB adresiranog sadržajem registra $ECNT_{4...0}$. U registar $TMP_{8...0}$ se pri aktivnoj vrednosti signala **ldPREG** trajanja jedna perioda signala takta upisuje sadržaj sa linija $\mathbf{TMPIN_{8...0}}$ formiran od signala **TV** i $\mathbf{TWAIT_{4...0}}$ sa izlaznih linija podataka $DO_{78...70}$ memorije TAB. Sadržaj registra $TMP_{8...0}$ se preko linija $\mathbf{TDIN_{78...70}}$ vodi na ulazne linije podataka $DI_{78...70}$ memorije TAB.

Brojač $ECNT_{4...0}$ (*Entry CouNTer*) služi za generisanje adresa ulaza memorije TAB. Signali $\mathbf{ECNT_{4...0}}$ sa izlaza ovog brojača se vode na adresne linije $A_{4...0}$ memorije TAB. Početna vrednost brojača je nula, a njegov sadržaj se inkrementira pri aktivnoj vrednosti signala **incECNT**.

Brojač $WCNT_{7...0}$ (*Wait CouNTer*) služi za realizaciju čekanja između generisanja dva zahteva. Dužina čekanja zavisi od sadržaja polja WAIT ulaza memorije TAB adresiranog sadržajem brojača $ECNT_{4...0}$. Vrednost dužine čekanja se iz polja WAIT adresiranog ulaza memorije TAB kao signali $\mathbf{TWAIT_{7...0}}$ vodi na ulaze brojača $WCNT_{7...0}$ i upisuje u njega pri aktivnoj vrednosti signala **ldWCNT**. Dekrementiranje brojača $WCNT_{7...0}$ se realizuje pri aktivnoj vrednosti signala **decWCNT**. Kada sadržaj brojača $WCNT_{7...0}$ postane nula signal **OUT** dobija aktivnu vrednost.

Flip-flop END (*generation of requests ENDed*) služi kao indikacija da li se završilo sa generisanjem zahteva iz svih ulaza memorije TAB. Ovaj flip-flop je na početku generisanja zahteva neaktivan, a postavlja se na aktivnu vrednost onda kada signal **end** postane aktivan. Signal **end** je aktivan ukoliko su i signal **ecntones** i signal **stEND** aktivni. Signal **ecntones** postaje aktivan kada sadržaj brojača $ECNT_{4...0}$ postane sve jedinice, što znači da se prešlo na generisanje zahteva iz zadnjeg ulaza memorije TAB. Signal **stEND** postaje aktivan jednu perioda signala takta u zadnjem koraku generisanja zanteva iz svakog ulaza memorije TAB.

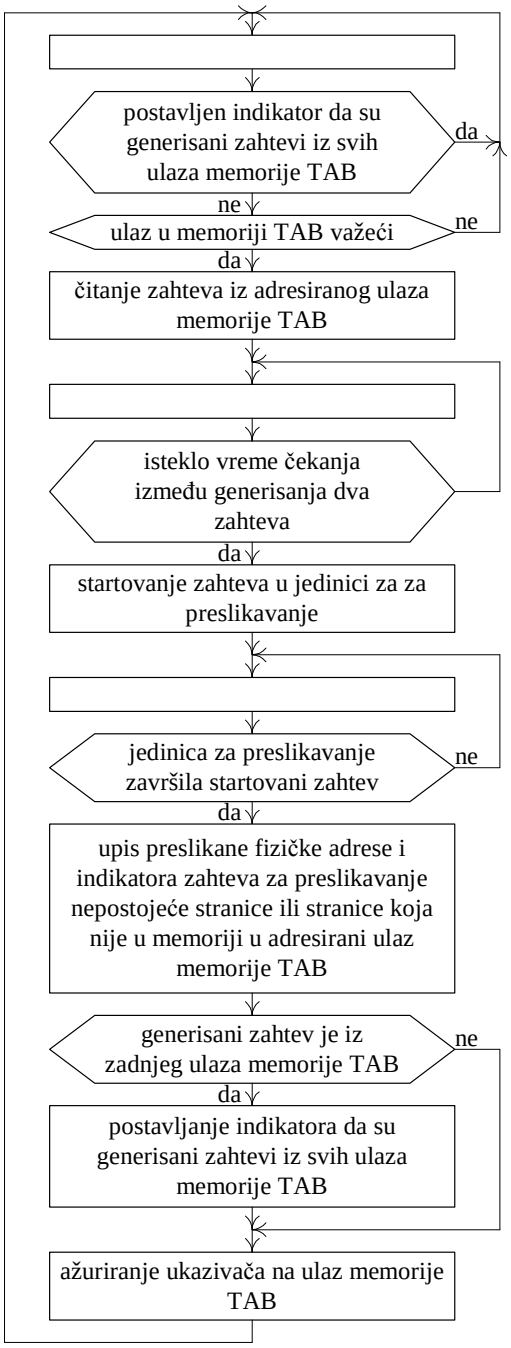
Stoga signal **end** postaje aktivan jednu periodu signala takta tek u zadnjem koraku generisanja zahteva iz zadnjeg ulaza memorije TAB, pa se tada na signal takta upisuje aktivna vrednost u flip-flop END.

2.2.2.1.2. Upravljačka jedinica

U ovom poglavlju se daju dijagram toka generisanjem zahteva za preslikavanje, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.

2.2.2.1.2.1. Dijagram toka generisanja zahteva

U početnom koraku se vrši provera da li je postavljen indikator da su generisani zahtevi za preslikavanje iz svih ulaza memorije TAB (slika 13).



Slika 13 Dijagram toka generisanja zahteva za preslikavanje

Ukoliko je indikator postavljen, ostaje se u početnom koraku i nema generisanja zahteva za preslikavanje. Ukoliko indikator nije postavljen, vrši se provera da li je ulaz memorije TAB na koji ukazuje ukazivač ulaza memorije TAB važeći. Ukoliko ulaz memorije TAB nije važeći, ostaje se u početnom koraku i nema generisanja zahteva za preslikavanje.

Ukoliko je ulaz važeći, prelazi se na korake generisanja zahteva za preslikavanje. Iz ulaza memorije TAB se, najpre, čita sadržaj neophodan za generisanje zahteva za preslikavanje, zatim se čeka da istekne vreme čekanja između generisanja dva zahteva i potom zahtev startuje u jedinici preslikavanja. Po prijemu indikacije da je jedinica preslikavanja završila dato preslikavanje procesor produžava sa preostalim koracima. Najpre se preslikana fizička

adresa i indikatori zahteva za preslikavanje nepostojeće stranice i stranice koja nije u memoriji upisuju u ulaz memorije TAB na osnovu čijeg sadržaja je i generisan zahtev za preslikavanje. Potom se, u slučaju da je zahtev generisan na osnovu sadržaja iz zadnjeg ulaza memorije TAB, postavlja indikator da su generisani zahtevi iz svih ulaza memorije TAB. Na kraju se vrši ažuriranje ukazivača na ulaz memorije TAB i prelazi na početni korak.

2.2.2.1.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture operacione jedinice (poglavlje 2.2.2.1.1) i dijagrama toka generisanja zahteva za preslikavanje (poglavlje 2.2.2.1.2.1). Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je sledeća:

<korak>: <bezuslovni_signali> <uslovni_signali> <sledeći_korak>

pri čemu je:

<bezuslovni_signali> := <signal>, {<signal>,} | <prazno>
<uslovni_signali> := <uslovni_signal>,{<uslovni_signal>,}|<prazno>
<uslovni_signal> := *if* (<uslov>, <signal> {,<signal>})
<sledeći_korak > := *br* <uslovni_korak>
<uslovni_korak> := <korak>|| (*if* <uslov> *then* < uslovni_korak>
else <uslovni_korak>)
<korak> := simboličke oznake za korake od 0 do 7 $step_0$ do $step_7$
<signal> := svi signali operacione jedinice koje generiše upravljačka jedinica.
<uslov> :=izrazi formirani od signala logičkih uslova koje generiše operaciona jedinica
<prazno> :=

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *if* (**TV**·$\overline{\textbf{END}}$, **ldPREG**)
br (*if* **TV**·$\overline{\textbf{END}}$ *then* $step_1$ *else* $step_0$)

! Od koraka $step_0$ se kreće prilikom generisanja svakog novog zahteva za preslikavanje. U korak $step_0$ se dolazi iz koraka $step_4$ po kompletiranju generisanja zahteva. U koraku $step_0$ se proverava da li signali **TV** i $\overline{\textbf{END}}$ bloka *zahtevi* imaju aktivne vrednosti što se dešava u slučaju kada je ulaz memorije TAB adresiran sadržajem registra $ECNT_{4...0}$ važeći i kada još uvek nije generisan zahtev za zadnji ulaz memorije TAB, respektivno. U slučaju da signali **TV** i $\overline{\textbf{END}}$ imaju aktivne vrednosti generiše se aktivna vrednost upravljačkog signala **ldPREG** blokova *zahtevi* i *tlb_interfejs*. Signalom **ldPREG** trajanja jedna perioda signala takta se u brojač $WCNT_{7...0}$ bloka *zahtevi* upisuje vrednost iz polja WAIT memorije TAB, u registar $TMP_{8...0}$ bloka *zahtevi* upisuju polja V i WAIT memorije TAB, u registre $PVA_{19...0}$, $PPR_{2...0}$ i $PPT_{19...0}$ i flip-flop PRW bloka *tlb_interfejs* upisuju polja VA, PR, PT i RW memorije TAB, u registar $PPA_{19...0}$ i flip-flopove PAV i PPF upisuju nule i prelazi na korak $step_1$. U suprotnom slučaju ostaje se u koraku $step_0$. !

$step_1$: *if* ($\overline{\textbf{OUT}}$, **decWCNT**),
if (**OUT**, **PRQ**),
br (*if* **OUT** *then* $step_2$ *else* $step_1$)

! U korak $step_1$ se dolazi iz koraka $step_0$. U ovom koraku se vrši provera vrednosti signala **OUT** bloka *zahtevi*. Neaktivna vrednost signala **OUT** označava da brojač WCNT još uvek nije došao do nule. U tom slučaju se generiše aktivna vrednost signala **decWCNT** bloka *zahtevi* i njome na signal takta dekrementira sadržaja brojača WCNT. Pri neaktivnoj vrednosti signala **OUT** ostaje se u koraku $step_1$. Aktivna vrednost signala **OUT** označava da je brojač WCNT kao rezultat dekrementiranja došao do nule. U tom slučaju se generiše aktivna vrednost signala **PRQ** bloka *tlb_interfejs* trajanja jedna perioda signala takta. Time se startuje jedinica ***TLB*** da izvrši preslikavanje. Pri aktivnoj vrednosti signala **OUT** prelazi se na korak $step_2$. !

$step_2$: *br* (*if* (**URP** + **UAV** + **UPF**) *then* $step_3$ *else* $step_2$)

! U korak $step_2$ se dolazi iz koraka $step_1$. U koraku $step_2$ se vrši provera signala **URP**, **UAV** i **UPF** bloka *tlb_interfejs* koje jedinica ***TLB*** šalje procesoru ***CPU***. Neaktivna vrednost signala **URP** je indikacija da jedinica ***TLB*** nije još uvek završila startovano preslikavanje, dok je aktivna vrednost indikacija da je preslikavanje

uspešno završeno. U slučaju da je preslikavanje uspešno završeno, fizička adresa, koja se po linijama $\mathbf{UPA_{19...0}}$ bloka *cpu_interfejs* jedinice ***TLB*** šalje u processor ***CPU***, upisuje se aktivnom vrednošću signala **URP** u registar $PPA_{19...0}$ bloka *tlb_interfejs*. Neaktivna vrednost signala **UAV** je indikacija da prilikom peslikavanja u jedinici ***TLB*** nije bilo pokušaja pristupa nepostojećoj stranici (*access violation*). Aktivna vrednost signala **UAV** trajanja jedna perioda signala takta je indikacija da je preslikavanje neuspešno realizovano zbog pokušaja pristupa nepostojećoj stranici (*access violation*). Aktivnom vrednošću signala **UAV** se u flip-flop PAV bloka *tlb_interfejs* upisuje aktivna vrednost. Neaktivna vrednost signala **UPF** je indikacija da prilikom peslikavanja u jedinici ***TLB*** nije bilo pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*). Aktivna vrednost signala **UPF** trajanja jedna perioda signala takta je indikacija da je preslikavanje neuspešno realizovano zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*). Aktivnom vrednošću signala **UPF** se u flip-flop PPF bloka *tlb_interfejs* upisuje aktivna vrednost. U koraku $step_2$ se ostaje sve dok su neaktivne vrednosti signala **URP**, **UAV** i **UPF**. Kada jedan od signala **URP**, **UAV** ili **UPF** postane aktivan, prelazi se na korak $step_3$. !

$step_3$: **wrTAB**,
br $step_4$

! U korak $step_3$ se dolazi iz koraka $step_2$. U ovom koraku se generiše aktivna vrednost signala **wrTAB** bloka *zahtevi* trajanja jedna perioda signala takta. Signalom **wrTAB** se fizička adresa iz registra $PPA_{19...0}$ i indikatori pokušaja pristupa nepostojećoj stranici iz flip-flopa PAV i pokušaja pristupa stranici koja nije u operativnoj memoriji iz flip-flopa PPF upisuju u odgovarajuća polja ulaza memorije TAB bloka *zahtevi*. Signalom **wrTAB** se, takođe, potvrđuju sadržaji u ostalim poljima istog ulaza memorije TAB tako što se u njih upisuju sadržaji registara $TMP_{8...0}$, $PVA_{23...0}$, $PPR_{2...0}$, $PPT_{19...0}$ i flip-flopa PRW. U koraku $step_3$ se ostaje samo jedna perioda signala takta i bezuslovno se prelazi u korak $step_4$. !

$step_4$: **incECNT**, **stEND**,
br $step_0$

! U korak $step_4$ se dolazi iz koraka $step_3$. U ovom koraku se bezuslovno generišu aktivne vrednost signala **incECNT** i **stEND** bloka *zahtevi* trajanja jedna perioda signala takta. Aktivnom vrednošću signala **incECNT** se inkrementira sadržaj brojača $ECNT_{4...0}$ i time njegova vrednost podešava na prvi sledeći ulaz memorije TAB bloka *zahtevi*. Aktivnom vrednošću signala **stEND** treba da se u ovom zadnjem koraku generisanja zahteva u slučaju kada je zahtev iz zadnjeg ulaza memorije TAB u flip-flop END upiše aktivna vrednost i time po prelasku na korak $step_0$ zaustavi rad procesora ***CPU***. U koraku $step_4$ se ostaje samo jadna perioda signala takta i bezuslovno se prelazi u korak $step_0$. !

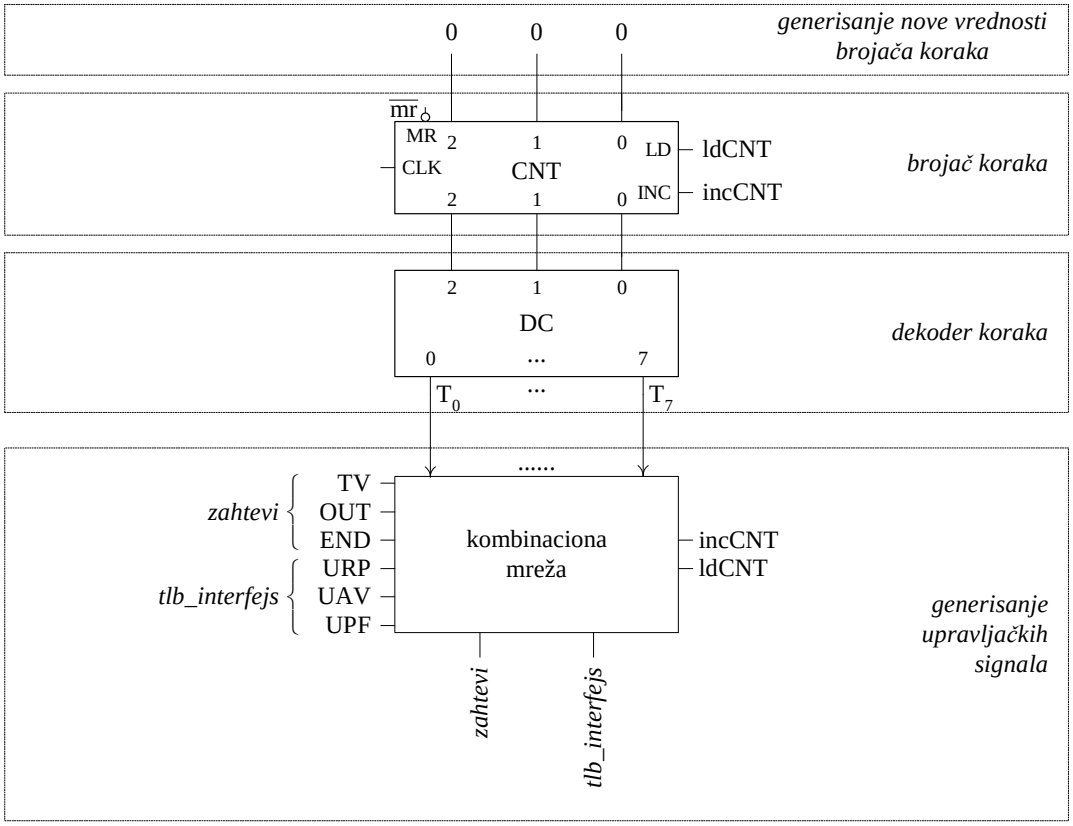
2.2.2.1.2.3. Struktura upravljačke jedinice

Upravljačka jedinica (slika 14) se sastoji od sledećih blokova:

- blok *generisanje nove vrednosti brojača koraka*,
- blok *brojač koraka*,
- blok *dekoder koraka* i
- blok *generisanje upravljačkih signala*.

Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.

Blok *generisanje nove vrednosti brojača koraka* služi za generisanje vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja onda kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje 2.2.2.1.2.2) se utvrđuje da je 0 vrednost koju treba upisati u brojač CNT. Ta vrednost je ožičena na ulazima 0, 1 i 2 brojača CNT.



Slika 14 Struktura upravljačke jedinice

Blok *brojač koraka* sadrži brojač CNT. Brojač CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač CNT može da radi u sledećim režimima:

- režim inkrementiranja,
- režim skoka i
- režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača CNT za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.1.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **incCNT**.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač CNT. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.1.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ldCNT**.

U režimu mirovanja pri pojavi signala takta ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **incCNT** i **ldCNT**.

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali $\mathbf{T_0}$ do $\mathbf{T_7}$ na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.1.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$, itd.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$ do $\mathbf{T_7}$ koji dolaze sa bloka *dekoder koraka* upravljačke jedinice, signala logičkih uslova koji dolaze iz blokova operacione jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje 2.2.2.1.2.2) generišu upravljačke signale. Za svaki od upravljačkih signala treba posmatrati u kojim koracima i pod kojim logičkim uslovima prema algoritmu generisanja upravljačkih signala dati upravljački signal treba da ima aktivnu vrednost. Dati upravljački signal se u opštem slučaju dobija kao unija proizvoda signala dekodovanih stanja brojača CNT i signala logičkih uslova pod kojim u datom koraku dati upravljački signal treba da bude aktivan. Da li će to uvek tako da bude zavisi od toga da li je neki upravljački signal aktivan u više ili samo u jednom koraku i da li je aktivan uslovno ili bezuslovno. Prema algoritmu generisanja upravljačkih signala jedan isti upravljački signal može da bude aktivan samo u jednom, a ne u više koraka. Stoga se uslovno generisani upravljački signali dobijaju kao proizvod signala dekodovanog stanja brojača CNT i signala logičkog uslova pod kojim u datom koraku dati upravljački signal treba da bude aktivan, dok se bezuslovno generisani upravljački signali dobijaju samo na osnovu signala dekodovanog stanja brojača CNT. Na primer, upravljački signal **PRQ** treba da je aktivan u koraku $step_1$ ukoliko je signal logičkog uslova **OUT** aktivan, pa se ovaj upravljački signal generiše prema relaciji $\mathbf{PRQ} = \mathbf{T1 \cdot OUT}$, dok upravljački signal **wrTAB** treba da je uvek aktivan u koraku $step_3$, pa se ovaj upravljački signal generiše prema relaciji $\mathbf{wrTAB} = \mathbf{T_3}$.

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:

- upravljačke signale operacione jedinice i
- upravljačke signale upravljačke jedinice.

Upravljački signali operacione jedinice se daju posebno za svaki blok operacione jedinice i to:

- blok *tlb_interfejs* i
- blok *zahtevi*.

Upravljački signali bloka *tlb_interfejs* se generišu na sledeći način:

- $\mathbf{ldPREG} = \mathbf{T_0 \cdot TV \cdot \overline{END}}$ i
- $\mathbf{PRQ} = \mathbf{T_3 \cdot OUT}$.

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **TV** — blok *zahtevi*,
- **END** — blok *zahtevi* i
- **OUT** — blok *zahtevi*.

Upravljački signali bloka *zahtevi* se generišu na sledeći način:

- $\mathbf{ldPREG} = \mathbf{T_0 \cdot TV \cdot \overline{END}}$,
- $\mathbf{decWCNT} = \mathbf{T1 \cdot \overline{OUT}}$,
- $\mathbf{wrTAB} = \mathbf{T_3}$,
- $\mathbf{incECNT} = \mathbf{T_3}$ i
- $\mathbf{stEND} = \mathbf{T_4}$.

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **TV** — blok *zahtevi*,
- **END** — blok *zahtevi* i
- **OUT** — blok *zahtevi*.

Upravljački signali upravljačke jedinice se generišu na sledeći način:

- **ldCNT** = $\mathbf{T_4}$ i
- **incCNT** = $\mathbf{T_0 \cdot TV \cdot \overline{END} + T_1 \cdot OUT + T_2 \cdot (URP + UAV + UPF) + T_3}$.

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

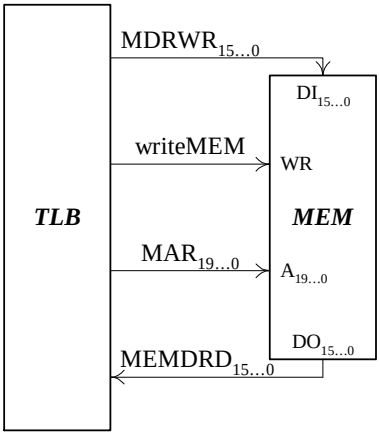
- **TV** — blok *zahtevi*,
- **END** — blok *zahtevi*,
- **OUT** — blok *zahtevi*,
- **URP** — blok *tlb_interfejs*,
- **UAV** — blok *tlb_interfejs* i
- **UPF** — blok *tlb_interfejs*,

2.2.2.2. MEMORIJA ***MEM***

Memoriji ***MEM*** (slika 15) se obraća jedinica za preslikavanje ***TLB***

- kod dovlačenja nultog ulaza tabele stranica radi provere da li je preslikavanje zadato za postojeću stranicu,
- kod prebacivanja deskriptora stranice iz tabele stranica u neki od ulaza jedinice za preslikavanje,
- kod dovlačenja deskriptora stranice iz tabele stranica u jedinicu za preslikavanje radi modifikovanja D indikatora na vrednost 1 i
- kod vraćanja deskriptora stranice sa modifikovanim D indikatorom iz jedinice za preslikavanje u tabelu stranica.

U prva tri slučaja sa memorijom ***MEM*** se realizuje operacija čitanja, a u četvrtom slučaju se realizuje operacija upisa.



Slika 15 Memorija ***MEM***

Kod operacije čitanja jedinica ***TLB*** šalje 20-bitnu adresu po linijama $\mathbf{MAR_{19...0}}$ i drži neaktivnu vrednost upravljačkog signala **writeMEM**. Očitani 16-bitni podatak se vraća po linijama $\mathbf{MEMDRD_{15...0}}$. Kod operacije upisa jedinica ***TLB*** šalje 20-bitnu adresu po linijama $\mathbf{MAR_{19...0}}$, 16-bitni podatak po linijama $\mathbf{MDRWR_{15...0}}$ i drži aktivnu vrednost upravljačkog signala **writeMEM**. Uzeto je da je vreme pristupa memorije ***MEM*** trajanja četiri periode signala takta.

2.2.2.3. JEDINICA *TLB*

Date su realizacije tri jedinice ***TLB*** za preslikavanje virtuelnih u fizičke adrese i to:

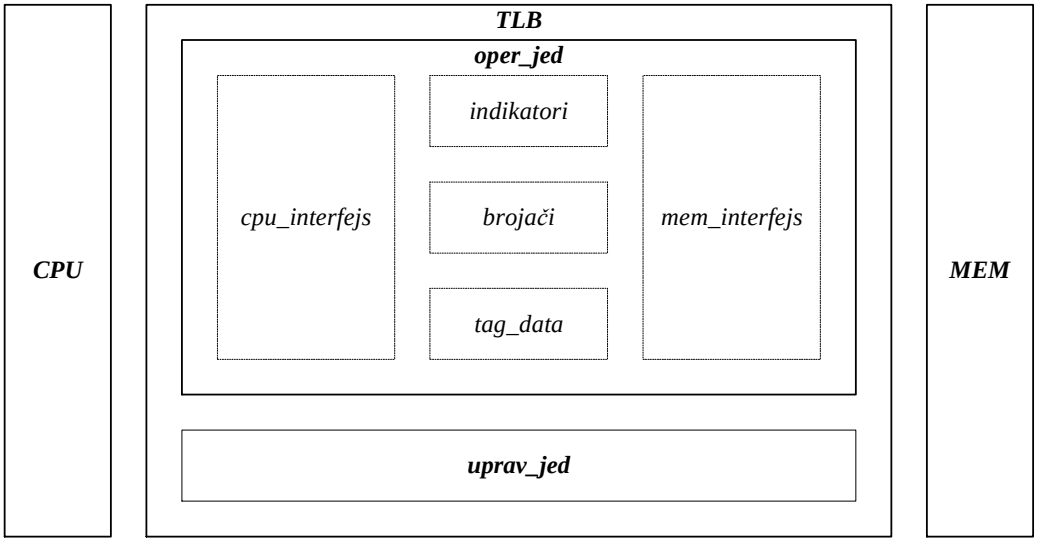
- jedinica sa asocijativnim preslikavanjem,
- jedinica sa direktnim preslikavanjem i
- jedinica sa set-asocijativnim preslikavanjem.

Usvojeno je da svaka od jedinica ***TLB*** ima 8 ulaza u kojima mogu da se čuvaju deskriptori stranica različitih procesa. Zbog toga se prilikom prebacivanja procesora sa procesa na proces ulazi u jedinici ***TLB*** ne proglašavaju za nevažeće. Usvojeno je da je broj procesa 8.

2.2.2.3.1. JEDINICA SA ASOCIJATIVNIM PRESLIKAVANJEM

Jedinica sa asocijativnim preslikavanjem ***TLB*** (slika16) se sastoji iz:

- operacione jedinice ***oper_jed*** i
- upravljačke jedinice ***uprav_jed***.



Slika 16 Jedinica sa asocijativnim preslikavanjem

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu asocijativnog preslikavanja virtuelne adrese u fizičku adresu. Sve sekvencijalne mreže unutar ***TLB*** su sinhrone.

2.2.2.3.1.1. OPERACIONA JEDINICA

Operaciona jedinica ***oper_jed*** se sastoji iz sledećih blokova:

- blok *cpu_interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag_data i*
- blok *cpu_ interfejs.*

Blok *cpu_interfejs* služi za razmenu podataka i upravljačkih signala sa procesorom ***CPU***.

Blok *indikatori* služi za vođenje evidencije za svaki od 8 ulaza jedinice o tome da li je u ulazu važeći descriptor i da li je sadržaj stranice modifikovan.

Blok *brojači* služi za odbrojavanje onoliko perioda signala takta koliko je vreme pristupa memoriji ***MEM*** i za određivanje broja ulaza jedinice za upis novog deskriptora po FIFO (*First In First Out*) algoritmu.

Blok *tag_data* služi za čuvanje 8 deskriptora stranica.

Blok *cpu_ interfejs* služi za razmenu podataka i upravljačkih signala sa memorijom ***MEM***.

Struktura i opis blokova operacione jedinice ***oper_jed*** se daju u daljem tekstu.

2.2.2.3.1.1.1. Blok *cpu_interfejs*

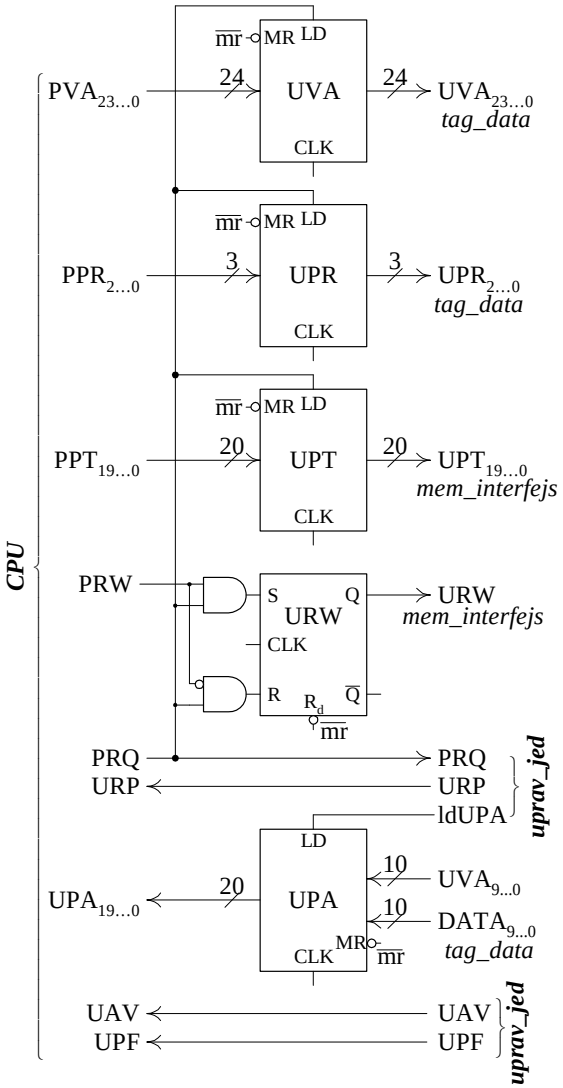
Blok *cpu_interfejs* (slika 17) sadrži registre $UVA_{23...0}$, $UPR_{2...0}$, $UPT_{19...0}$ i $UPA_{19...0}$ i flip-flop URW.

Registar $UVA_{23...0}$ (*Unit Virtual Address register*) služi za čuvanje virtuelne adrese koju jedinica ***TLB*** treba da preslika u fizičku adresu. Virtuelna adresa dolazi na ulaze registra $UVA_{23...0}$ po linijama $\mathbf{PVA_{23...0}}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UVA_{23...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj registra $UVA_{23...10}$ se vodi u blok *tag_data* gde se koriste za utvrđivanje saglasnosti sa nekim od ulaza memorije TAG i upisivanje u memoriju TAG prilikom dovlačenja deskriptora stranice. Sadržaj razreda $UVA_{9...0}$ se takođe vodi na ulaze 9 do 0 registra $UPA_{19...0}$ gde se koristi kao adresa u bloku prilikom formiranja fizičke adrese.

Registar $UPR_{2...0}$ (*Unit PRoces register*) služi za čuvanje broja tekućeg procesa. Broj tekućeg procesa dolazi na ulaze registra $UPR_{2...0}$ po linijama $\mathbf{PPR_{2...0}}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UPR_{2...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj registra $UPR_{2...0}$ se vodi u blok *tag_data* gde se koristi za utvrđivanje saglasnosti sa nekim od ulaza memorije TAG i upisivanje u memoriju TAG prilikom dovlačenja deskriptora stranice.

Registar $UPT_{19...0}$ (*Unit Page map Table address register*) služi za čuvanje početne adrese tabele stranica tekućeg procesa. Adresa dolazi na ulaze registra $UPT_{19...0}$ po linijama $PPT_{19...0}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UPT_{19...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj registra $UPT_{19...0}$ se po vodi u blok *mem_interfejs* gde se koristi kao adresa za čitanje nultog ulaza tabele stranica radi provere da li se radi o pokušaju pristupa nepostojećoj stranici (*access violation*) i za formiranje adrese deskriptora stranice u tabeli stranica radi čitanja deskriptora ili upisa u deskriptor.

Registar $UPA_{19...0}$ (*Unit Physical Address register*) služi za čuvanje fizičke adrese koja se generiše u slučaju uspešnog preslikavanja virtuelne adrese u fizičku adresu. Fizička adresa se generiše tako što se na ulaze 19 do 10 registra $UPA_{19...0}$ dovodi sadržaj sa izlaznih linija $\mathbf{DATA_{9...0}}$ memorije DATA bloka *tag_data*, a na ulaze 9 do 0 registra $UPA_{19...0}$ dovodi sadržaj sa linija $\mathbf{UVA_{9...0}}$. Generisana fizička adresa se u registar $UPA_{19...0}$ upisuje pri aktivnoj vrednosti signala **ldUPA**. Sadržaj registra $UPA_{19...0}$ se vodi u blok *tlb_interfejs* procesora ***CPU***.



Slika 17 Blok *cpu_interfejs*

Flip-flop URW (*Unit Read Write flag*) vrednostima 0 i 1 predstavlja indikator operacije čitanja ili upisa, respektivno, koja treba da se realizuje po preslikavanju virtuelne adrese u fizičku adresu. Vrednost indikatora dolazi na ulaze flip-flopa URW po liniji PRW iz bloka *tlb_interfejs* procesora ***CPU***, a u flip-flop URW se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj flip-flopa URW se vodi u blok *indikatori*.

Upravljački signal **PRQ** (*Processor ReQuest*) koji dolazi iz bloka *tlb_inerfejs* procesora ***CPU*** se koristi da procesor ***CPU*** aktivnom vrednošću ovog signala trajanja jedna perioda izvrši upis sadržaja sa linija $\mathbf{PVA_{23...0}}$, $\mathbf{PPR_{2...0}}$, $\mathbf{PPT_{19...0}}$ i **PRW** bloka *tlb_interfejs* u registre $UVA_{23...0}$, $UPR_{2...0}$ i $UPT_{19...0}$ i flip_flop URW jedinice ***TLB*** i da se startuje preslikavanje.

Upravljački signal **URP** (*Unit RePly*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da je peslikavanje uspešno realizovano i da se na linijama $\mathbf{UPA_{19...0}}$ nalazi generisana fizička adresa.

Upravljački signal **UAV** (*Unit Address Violation*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda

signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa nepostojećoj stranici (*access violation*).

Upravljački signal **UPF** (*Unit Page Fault*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*).

2.2.2.3.1.1.2. Blok *indikatori*

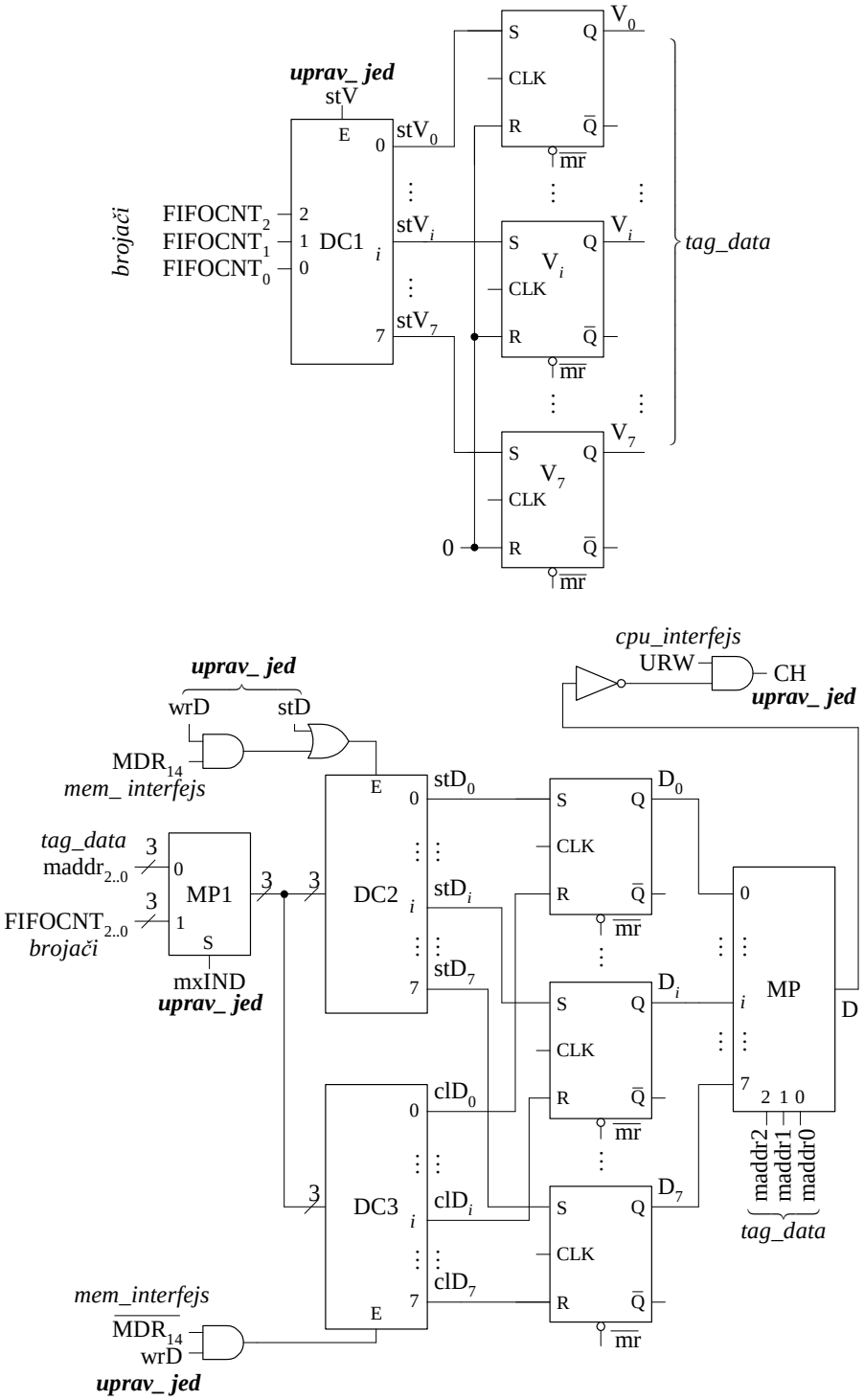
Blok *indikatori* (slika 18) se sastoji od flip_flopova V_0 do V_7 sa dekoderom DC1 i flip_flopova D_0 do D_7 sa dekoderima DC2 i DC3 i multiplekserima MP1 i MP2.

Flip-flopovi V_0 do V_7 (*Valid*) se koriste kao indikatori za svaki od ulaza 0 do 7 jedinice ***TLB*** da li je ulaz važeći ili nije važeći. Flip-flop V_i, i=0,…7, se postavlja na 1 aktivnom vrednošću signala $\mathbf{stV_i}$ trajanja jedna perioda signala takta prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti ulaz jedinice ***TLB***. Signali V_0 do V_7 se vode u blok *tag_data* gde se koriste za utvrđivanje saglasnosti.

Dekoder DC1 se koristi da se, formiranjem aktivne vrednosti jednog od signala $\mathbf{stV_0}$ do $\mathbf{stV_7}$ na izlazima dekodera DC1 trajanja jedna perioda signala takta, jedan od flip-flopova V_0 do V_7 postavi na 1. Ovo se realizuje prilikom dovlačenja deskriptora stranice generisanjem aktivne vrednosti signala **stV** trajanja jedna perioda signala takta koji se pojavljuje kao aktivna vrednost na izlazu dekodera DC1 određenom sadržajem brojača za zamenu $FIFOCNT_{2...0}$ koji se dovodi na ulaze dekodera DC1 iz bloka *brojači*.

Flip-flopovi D_0 do D_7 (*Dirty*) se koriste kao indikatori za svaki od ulaza 0 do 7 jedinice ***TLB*** da li je bilo upisa u stranicu čiji se deskriptor nalazi u datom ulazu ili ne. Flip-flop D_i, i=0,…7, se postavlja na 1 aktivnom vrednošću signala $\mathbf{stD_i}$ trajanja jedna perioda signala i na 0 aktivnom vrednošću signala $\mathbf{clD_i}$ trajanja jedna perioda signala takta. Ovi signali se koriste da se prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti ulaz jedinice ***TLB*** flip-flop D_i postavi na vrednost bita D (*Dirty*) deskriptora stranice, koji se nalazi u razredu MDR_{14} registra $MDR_{15...0}$ bloka *mem_interfejs*, i da se prilikom generisanja fizičke adrese radi upisa, flip-flop D_i postavi na 1.

Dekoder DC2 se koristi da se, formiranjem aktivne vrednosti jednog od signala $\mathbf{stD_0}$ do $\mathbf{stD_7}$ na izlazima dekodera DC2 trajanja jedna perioda signala takta, jedan od flip-flopova D_0 do D_7 postavi na 1. Dekoder DC3 se koristi da se, formiranjem aktivne vrednosti jednog od signala $\mathbf{clD_0}$ do $\mathbf{clD_7}$ na izlazima dekodera DC3 trajanja jedna perioda signala takta, jedan od flip-flopova D_0 do D_7 postavi na 0. Multiplekser MP1 se koristi da se vrednostima 0 i 1 signala **mxIND** kroz multiplekser na ulaze dekodera DC2 i DC3 propusti ili vrednost $\mathbf{maddr_{2..0}}$ iz bloka *tag_data*, koja predstavlja broj ulaza u kome je otkrivena saglasnost, ili vrednost $\mathbf{FIFOCNT_{2...0}}$ bloka *brojači*, koja predstavlja broj ulaza koji je odabran za zamenu. Signal **mxIND** je normalno 0, pa se pri otkrivanju saglasnosti kroz multiplekser propušta $\mathbf{maddr_{2..0}}$ bloka *tag_data*. Signal **mxIND** je 1 kada se, zbog toga što nije otkrivena saglasnost, dovlači deskriptor, pa se kroz multiplekser tada propušta $\mathbf{FIFOCNT_{2..0}}$ bloka *brojači*.



Slika 18 Blok *indikatori*

Prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-tom ulazu jedinice ***TLB***, *i*=0,…7, treba da se generiše aktivna vrednost signala **stD*i*** i da se flip-flop D*i* postavi na 1. Tada se generiše signal **stD** koji se pojavljuje kao aktivna vrednost signala **stD*i*** na izlazu dekodera DC2 određenom sadržajem **maddr$_{2..0}$** bloka *tag_data*, koji prolazi kroz multiplekser MP1 i pojavljuje se na ulazima dekodera DC2.

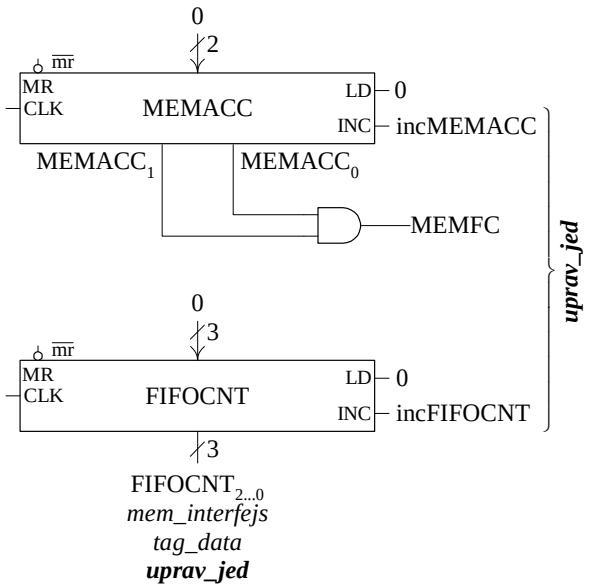
Prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti ulaz jedinice ***TLB***, treba da se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, generiše aktivna vrednost signala $\mathbf{stD_i}$ ili $\mathbf{clD_i}$, respektivno, i da se flip-flop D_i postavi na vrednost bita D (*Dirty*) deskriptora stranice. Tada se generiše signal **wrD** koji se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, pojavljuje kao aktivna vrednost ili signala $\mathbf{stD_i}$ na izlazu dekodera DC2 ili signala $\mathbf{clD_i}$ na izlazu dekodera DC3. Izlazi dekodera DC2 i DC3 su određeni sadržajem $FIFOCNT_{2..0}$ bloka *tag_data*, koji prolazi kroz multiplekser MP2 i pojavljuje se ulazima dekodera DC2 i DC3.

Multiplekser MP2 se koristi da se signalima $\mathbf{maddr_{2..0}}$ iz bloka *tag_data*, koji predstavljaju broj *i*-tog ulaza, *i*=0,…7, u kome je otkrivena saglasnost, na izlazu multipleksera MP2 kao signal **D** selektuje indikator D_i. Signal **D** sa signalom **URW** bloka *cpu_interfejs* formira signal **CH** (*change*), pri čemu signal **URW** vrednostima 0 i 1 ukazuje da li sa preslikane fizičke adrese treba realizovati čitanje ili upis, respektivno. Signal **CH** ima vrednost 1 ukoliko signal **URW** ima vrednost 1, jer se radi o upisu, i ukoliko signal **D** ima vrednost 0, jer nije bilo upisa u stranicu čiji se deskriptor nalazi u *i*-tom ulazu jedinice ***TLB***. Tada treba bit D deskriptora date stranice u tabeli stranica postaviti na 1. U svim ostalim slučajevima signal **CH** ima vrednost 0 i tada ne treba ništa raditi.

Signal **CH** se koristi prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-tom ulazu jedinice ***TLB***. Ukoliko je bit D 0, to znači da nije bilo upisa u datu stranicu i da treba bit D deskriptora date stranice u tabeli stranica postaviti na 1. Kako je za operaciju upisa indikator URW bloka *cpu_interfejs* 1, to je i signal **CH** 1. Ukoliko je bit D 1, to znači da je bilo upisa u datu stranicu i da je bit D deskriptora date stranice u tabeli stranica već postavljen 1. Kako je za operaciju upisa indikator URW bloka *cpu_interfejs* 1, to je i signal **CH** 0.

2.2.2.3.1.1.3. Blok *brojači*

Blok *brojači* (slika 19) se sastoji od brojača vremena pristupa operativnoj memoriji $MEMACC_{1...0}$ (*MEMory ACCess time*) i brojača broja ulaza za zamenu $FIFOCNT_{2...0}$ (FIFO CouNTer).



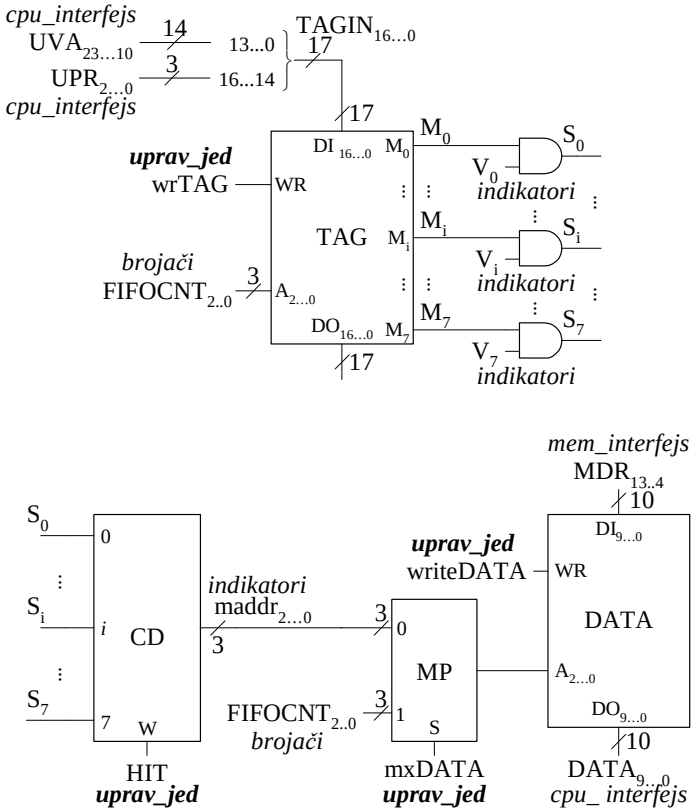
Slika 19 Blok *brojači*

Brojač $MEMACC_{1...0}$ se koristi da odbroji četiri signala takta kod obraćanja jedinice ***TLB*** memoriji ***MEM*** radi čitanja ili upisa, jer je usvojeno da je vreme pristupa memoriji ***MEM*** četiri periode signala takta. Inkrementiranje brojača se realizuje pri aktivnoj vrednosti signala **incMEMACC**. Signal **MEMFC** (*MEMory Function Completed*) postaje aktivan na treći signal takta kada brojač $MEMACC_{1...0}$ inkrementiranjem pređe u stanje tri. Na četvrti signal takta brojač $MEMACC_{1...0}$ se vraća na stanje nula, a signal **MEMFC** postaje neaktivan. Signal **MEMFC** služi upravljačkoj jedinici ***uprav_jed*** kao indikacija da je pristup memoriji ***MEM*** završen.

Brojač $FIFOCNT_{2...0}$ se koristi za realizaciju FIFO algoritma zamene. Tekuća vrednost brojača $FIFOCNT_{2...0}$ se koristi kada nema saglasnosti u jedinici ***TLB*** za određivanje ulaza jedinice ***TLB*** u koji treba dovući deskriptor stranice. Sadržaj brojača $FIFOCNT_{2...0}$ se inkrementira aktivnom vrednošću signala **incFIFOCNT** po dovlačenju deskriptora stranice.

2.2.2.3.1.1.4. Blok *tag_data*

Blok *tag_data* (slika 20) sadrži asocijativnu memoriju TAG, 8 logičkih I kola, koder CD, RAM memoriju DATA i multiplekser MP.



Slika 20 Blok *tag_data*

Memora TAG čuva za svaki od 8 ulaza jedinice ***TLB*** broj procesa i broj stranice čiji se descriptor trenutno nalazi u istom ulazu memorije DATA. Kapacitet memorije TAG je 8 reči širine 17 bitova. Memorija TAG ima sledeće ulazne i izlazne linije:

- adresne linije $\mathbf{A_{2...0}}$,
- ulazne linije podataka $\mathbf{DI_{16...0}}$,
- izlazne linije podataka $\mathbf{DO_{16...0}}$,
- izlazne linije podudaranja $\mathbf{M_{7...0}}$ i

- upravljačku liniju **WR**.

Upis u memoriju TAG se realizuje aktivnom vrednošću signala **wrTAG**, a čitanje njegovom neaktivnom vrednošću. Pored toga u memoriji TAG se realizuje i upoređivanje sadržaja svake od osam lokacija sa sadržajima na ulaznim linijama podataka $\mathbf{DI_{16...0}}$. Na osnovu toga se za osam ulaza memorije TAG formiraju signali podudaranja $\mathbf{M_{7...0}}$ (*Match*). Memoriji TAG se pristupa u sledećim slučajevima:

1. *Utvrđivanje saglasnosti sadržaja na ulaznim linijama podataka* $DI_{16...0}$ *sa sadržajima svih osam ulaza memorije TAG*. U ovom slučaju na ulazne linije podataka $\mathbf{DI_{16...0}}$ se vode signali $\mathbf{TAGIN_{16...0}}$, gde signali $\mathbf{TAGIN_{16...14}}$ predstavljaju signale broja procesa $\mathbf{UPR_{2...0}}$ bloka *cpu_interfejs* i signali $\mathbf{TAGIN_{13...0}}$ predstavljaju signale broja stranice $\mathbf{UVA_{23...10}}$ bloka *cpu_interfejs*. Ukoliko se deskriptor date stranice datog procesa nalazi u nekom od osam ulaza memorije DATA, doći će do podudaranja sadržaja datog ulaza memorije TAG i sadržaja na ulaznim linijama podataka $\mathbf{DI_{16...0}}$ i odgovarajući signal na izlaznim linijama podudaranja $\mathbf{M_{7...0}}$ će postati aktivan. U suprotnom slučaju svi signali $\mathbf{M_{7...0}}$ će biti neaktivni.

2. *Upisivanje broja procesa i broja stranice kod dovlačenja deskriptora stranice*. Signali na ulaznim linijama podataka $\mathbf{DI_{16...0}}$ imaju isto značenje i formiraju se na isti način kao u slučaju 1. Sadržaj sa linija $\mathbf{DI_{16...0}}$ se upisuje u isti ulaz memorije TAG kao što je i ulaz memorije DATA u koji se upisuje deskriptor. Taj ulaz je određen vrednošću signala $\mathbf{FIFOCNT_{2...0}}$ bloka *brojači* koji se vode na adresne linije $\mathbf{A_{2...0}}$ memorije TAG.

Logička I kola služe za formiranje signal saglasnosti $\mathbf{S_7}$ do $\mathbf{S_0}$ za svaki od osam ulaza jedinice za preslikavanje. Ovi signali se formiraju prema relaciji $\mathbf{S}_i = \mathbf{M}_i \cdot \mathbf{V}_i$, gde je $i = 0,...,7$. Da bi u i – tom ulazu bila otkrivena saglasnost ($\mathbf{S}_i = 1$), potrebno je ne samo da za i – ti ulaz postoji podudaranje ($\mathbf{M}_i = 1$), već i da je i – ti ulaz važeći ($\mathbf{V}_i = 1$). Signali $\mathbf{V_7}$ do $\mathbf{V_0}$ bloka *indikatori* ukazuju da li je odgovarajući ulaz važeći ili ne.

Koder CD služi za formiranje signala saglasnosti **HIT** i signala binarne vrednosti ulaza jedinice za preslikavanje u kome je otkrivena saglasnost $\mathbf{maddr_{2...0}}$. Signal **HIT** je aktivan ukoliko je jedan od signala $\mathbf{S_7}$ do $\mathbf{S_0}$ aktivan. U suprotnom slučaju, signal **HIT** je neaktivan. Signali $\mathbf{maddr_{2...0}}$ predstavljaju binarnu vrednost linije kodera CD na kojoj jedan od signala $\mathbf{S_7}$ do $\mathbf{S_0}$ ima aktivnu vrednost.

Memorija DATA čuva za svaki od 8 ulaza jedinice za preslikavanje deskriptor stranice koji predstavlja broj bloka u koji je preslikana stranica, pri čemu se njen broj zajedno sa brojem procesa kome stranica pripada nalazi u istom ulazu memorije TAG. Kapacitet memorije DATA je 8 reči širine 10 bitova. Memorija DATA ima sledeće ulazne i izlazne linije:

- adresne linije $\mathbf{A_{2...0}}$,
- ulazne linije podataka $\mathbf{DI_{9...0}}$,
- izlazne linije podataka $\mathbf{DO}_{9...0}$ i
- upravljačku liniju **WR**.

Upis u memoriju DATA se realizuje aktivnom vrednošću signala **wrDATA**, a čitanje njegovom neaktivnom vrednošću.

Memoriji DATA se pristupa u sledećim slučajevima:

1. *Čitanje broja bloka i formiranja fizičke adrese pri utvrđenoj saglasnosti u memoriji TAG*. U ovom slučaju signali na adresnim linijama $\mathbf{A_{2...0}}$ memorije DATA predstavljaju signale $\mathbf{maddr_{2...0}}$ koji se kroz multiplekser MP propuštaju neaktivnom vrednošću upravljačkog signala **mxADATA**. Očitana vrednost sa izlaza $\mathbf{DO}_{9...0}$ memorije DATA, koja predstavlja broj bloka u koji se preslikava stranica za koju je utvrđena saglasnost u memoriji

TAG, vodi se u blok *cpu_interfejs* na ulaze 19…10 registra $URA_{19...10}$ radi formiranja fizičke adrese.

2. *Upisivanje broja bloka kod dovlačenja deskriptora stranice*. U ovom slučaju signali na adresnim linijama $\mathbf{A_{2...0}}$ memorije DATA predstavljaju signale $\mathbf{FIFOCNT_{2...0}}$ bloka *brojači* koji se kroz multiplekser MP propuštaju aktivnom vrednošću upravljačkog signala **mxADATA**. Podatak za upis koji predstavlja broj bloka u koji se stranica preslikava i koji se vodi na ulazne linije podataka $\mathbf{DI_{9...0}}$ memorije DATA predstavlja razrede $\mathbf{MDR_{13...4}}$ pročitanog deskriptora stranice iz registra $\mathbf{MDR_{15...0}}$ bloka *mem_interfejs*.

2.2.2.3.1.1.5. Blok *mem_interfejs*

Blok *mem_interfejs* (slika 21) sadrži registar $MAR_{19...0}$ sa multiplekserom MP1, registar $MDR_{15...0}$ sa multiplekserom MP2, registar $UPG_{13...0}$, sabirač ADD i komparator CMP.

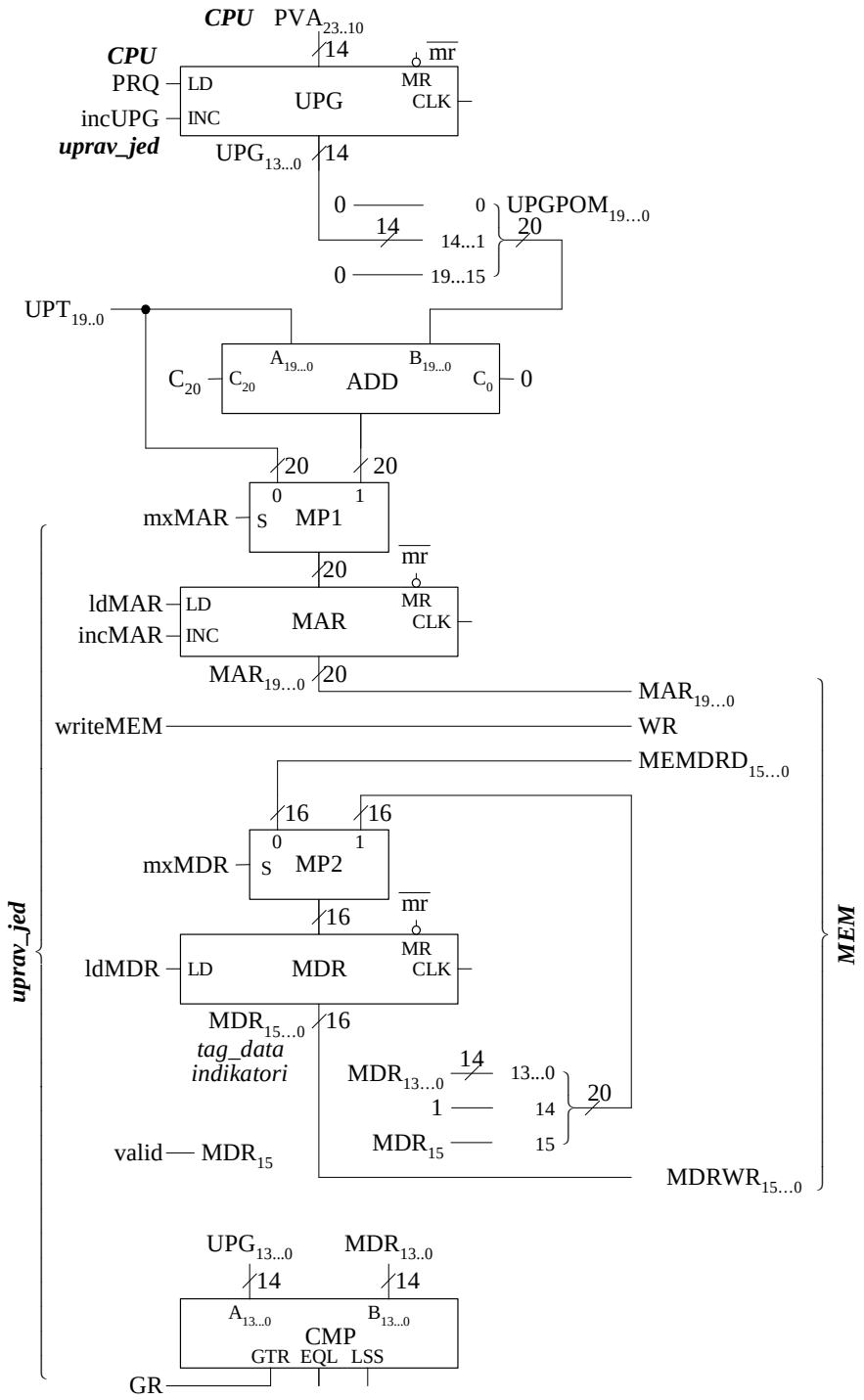
Registar $UPG_{13...0}$ (*Unit PaGe register*) služi za čuvanje broja stranice virtuelne adrese. Broj stranice dolazi po linijama $PVA_{23...10}$ procesora ***CPU*** i upisuje se u registar $UPG_{13...0}$ pri aktivnoj vednosti signala **PRQ** procesora ***CPU***. Sadržaj registra $UPG_{13...0}$ se inkrementira pri aktivnoj vednosti signala **incUPG** trajanja jedna perioda signala takta. Ovaj registar se koristi za formiranje signala pomeraja $\mathbf{UPGPOM_{19...0}}$ u odnosu na početnu adresu tabele stranica sa kojim treba očitati descriptor stranice iz tabele stranica. Pošto je veličina jednog ulaza u tabeli stranica dve reči i pošto se u nultom ulazu tabele stranica nalazi ukupan broj stranica procesa na koji se tabela odnosi, pomeraj $UPGPOM_{19...0}$ se formira kao $(UPG_{13...0}+1)\cdot 2$. Zbog toga se aktivnom vrednošću signala **incUPG** najpre inkrementira sadržaj registra $UPG_{13...0}$, pa se posle toga sadržaj registra $UPG_{13...0}$ pomeren ulevo za jedno mesto, čime se vrši množenje broja stranice sa dva, koristi za formiranje signala pomeraja $\mathbf{UPGPOM_{19...0}}$.

Sabirač ADD se koristi za izračunavanje adrese sa koje treba iz tabele stranica pročitati deskriptor stranice. Na ulaze $A_{19...0}$ sabirača se dovodi sadržaj registra adrese tabele stranica $UPT_{19...0}$ bloka *cpu_interfejs*, a na ulaze $B_{19...0}$ sadržaj signala pomeraja $\mathbf{UPGPOM_{19...0}}$. Na izlazima sabirača ADD se formira adresa sa koje treba pročitati descriptor stranice.

Registar $MAR_{19...0}$ (*Memory Address Register*) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba pročitati podatak, koji može biti ili niža reč deskriptora stranice ili niža reč nultog ulaza tabele stranica, ili adrese lokacije memorije ***MEM*** u koju treba upisati nižu reč deskriptora stranice u kojoj je bit D izmenjen na 1. Adresa se upisuje u registar $MAR_{19...0}$ pri aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje može biti ili sadržaj registra $UPT_{19...0}$, koji predstavlja adresu niže reči nultog ulaza u tabeli stranica, ili sadržaj formiran na izlazima sabirača ADD, koji predstavlja adresu niže reči deskriptora stranice u tabeli stranica. Selekcija jedne od te dve vrednosti kroz multiplekser MP1 se realizuje upravljačkim signalom **mxMAR**. Sadržaj registra $MAR_{19...0}$ se vode na adresne linije memorije ***MEM***.

Registar $MDR_{15...0}$ (*Memory Data Register*) služi za čuvanje podatka koji je pročitan iz memorije ***MEM*** i za čuvanje podatka koji treba upisati u memoriju ***MEM***. Podatak pročitan iz memorije ***MEM***, koji može biti ili niža reč deskriptora stranice ili niža reč nultog ulaza tabele stranica, dolazi po linijama podataka $\mathbf{MEMDRD_{15...0}}$ iz memorije ***MEM***. Podatak koji treba upisati u memoriju ***MEM*** je sadržaj registra $MDR_{15...0}$ u kome je bit 14 postavljen na 1 i predstavlja nižu reč deskriptora stranice u kojoj je bit D postavljen na 1. Podatak se upisuje u registar $MDR_{15...0}$ na signal takta pri aktivnoj vrednosti signala **ldMDR**. Selekcija jedne od te dve vrednosti kroz multiplekser MP2 se realizuje upravljačkim signalom **mxMDR**. Izlazi registra $MDR_{15...0}$ se vode na ulazne linije podataka memorije ***MEM***. Izlazi registra $MDR_{13...4}$ se vode i u memoriju DATA bloka *tag_data* radi upisa broja bloka deskriptora stranice u

memoriju DATA, izlaz MDR_{14} se vodi u blok *indikatori* radi upisa bita D deskriptora stranice u odgovarajući flip-flop D_7 do D_0 i izlazi $MDR_{15...0}$ se, sa razredom MDR_{14} postavljenim na 1, preko multipleksera MP2 vraćaju u registar $MDR_{15...0}$.



Slika 21 Blok *mem_interfejs*

Komparator CMP poredi broj stranice virtuelne adrese uvećan za jedan iz registra $UPG_{13...0}$ i broj stranica procesa iz razreda 13 do 0 registra $MDR_{15...0}$, koji predstavljaju broj stranica procesa pročitan iz nultog ulaza tabele stranica, i na izlazu GTR generiše signal greške **GR**

zbog zahteva za preslikavanje nepostojeće stranice. Signal **GR** se koristi kao signal logičkog uslova u upravljačkoj jedinici. Ako je signal **GR** na aktivnoj vrednosti, generiše se aktivna vrednost signala **UAV** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice.

Prilikom čitanja deskriptora stranice u razredu MDR_{15} se nalazi indikator koji ukazuje da li je stranica u operativnoj memoriji. Signal iz razreda MDR_{15} daje signal greške **valid** zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji. Signal **valid** se koristi kao signal logičkog uslova u upravljačkoj jedinici. Ako je signal **valid** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **UPF** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji.

2.2.2.3.1.2. UPRAVLJAČKA JEDINICA

U ovom poglavlju se daju dijagram toka zahteva, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.

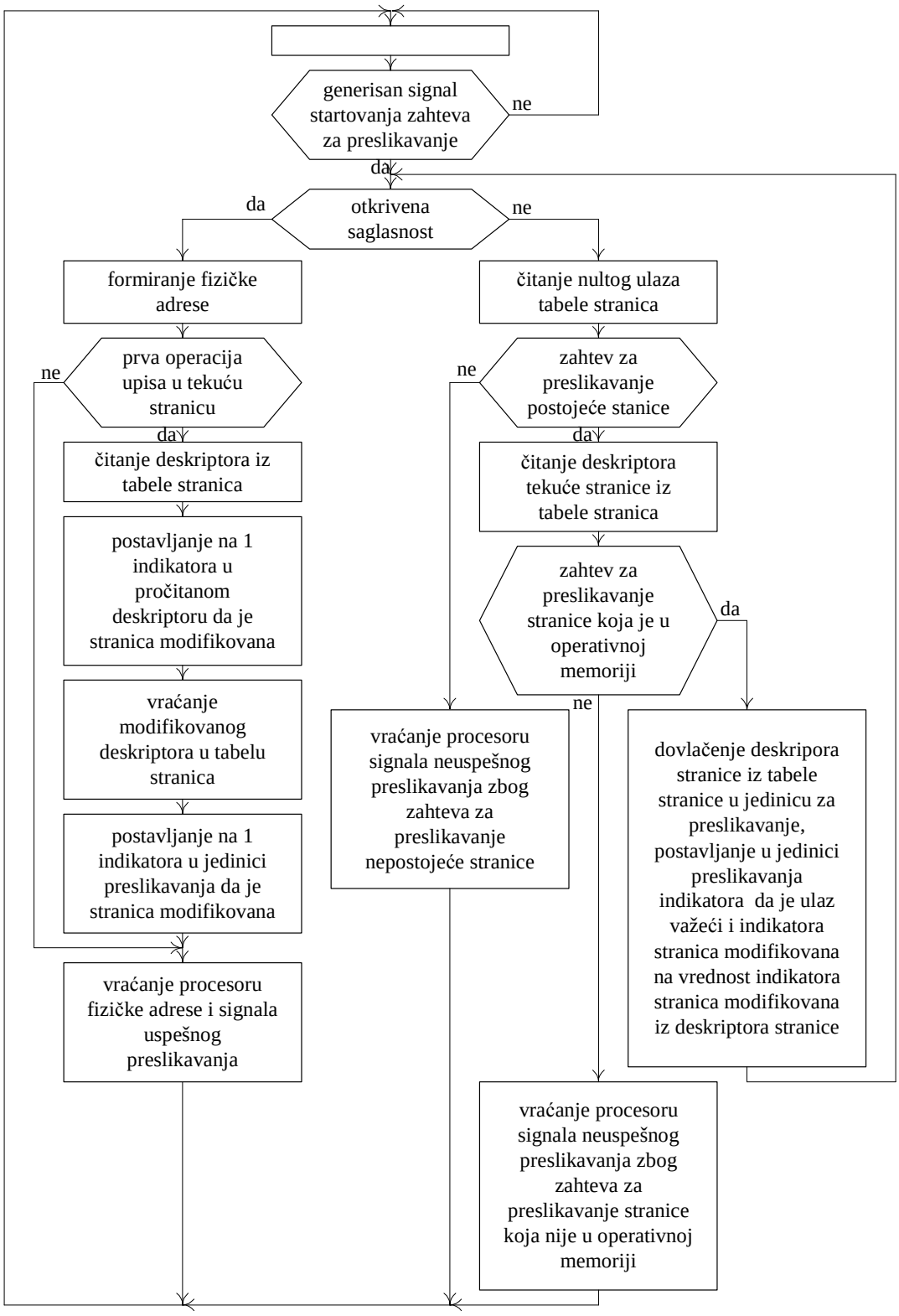
2.2.2.3.1.2.1. Dijagram toka zahteva

Na početku dijagrama toka se čeka pojava aktivne vrednosti signala startovanja zahteva za preslikavanje (slika 22). Pri njenom pojavljivanju iz procesora se u registre jedinice za preslikavanje upisuju virtuelna adresa, broj procesa, početna adresa tabele stranica i indikator operacije čitanja ili upisa i startuje zahtev u jedinici za preslikavanje.

Najpre se proverava da li postoji saglasnost virtuelne adrese sa sadržajem nekog od ulaza jedinice za preslikavanje, čime se utvrđuje da li se deskriptor stranice nalazi u jedinici preslikavanja.

Ukoliko postoji saglasnost vrši se formiranje fizičke adrese. Zatim se proverava da li je preslikavanje realizovano za operaciju upisa i to za prvu operaciju upisa na tekućoj stranici. Ukoliko jeste, iz tabele stranica se dovlači deskriptor, zatim se u njemu postavlja na 1 indikator da je stranica modifikovana i na kraju modifikovani deskriptor stranice vraća u tabelu stranica. Pored toga u jedinici preslikavanja na 1 se postavlja i indikator da je stranica modifikovana. Na kraju zahteva se procesoru vraća fizička adresa i signal da je preslikavanje uspešno realizovano i prelazi na početni korak u kome se čeka pojava aktivne vrednosti signala startovanja sledećeg zahteva za presliikavanje. Ukoliko se ne radi o prvoj operaciji upisa na datoj stranici, odmah se procesoru vraća fizička adresa i signal da je preslikavanje uspešno realizovano i prelazi na početni korak.

Ukoliko ne postoji saglasnost prelazi se na dovlačenje niže reči nultog ulaza tabele stranica u kome se nalazi ukupan broj stranica procesa i poređenje ukupnog broja stranica procesa sa brojem stranice iz virtuelne adrese. Ukoliko je broj stranice virtuelne adrese veći od broja stranica procesa, radi se o zahtevu za preslikavanje nepostojeće stranice, pa se procesoru šalje signal da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice i prelazi na početni korak. U suprotnom slučaju se produžava sa preslikavanjem tako što se čita deskriptor stranice iz tabele stranica i proverava da li se stranica nalazi u operativnoj memoriji. Ukoliko se stranica ne nalazi u operativnoj memoriji, radi se o zahtevu za preslikavanje stranice koja nije u operativnoj memoriji, pa se procesoru šalje signal da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji i prelazi na početni korak.



Slika 22 Dijagram toka operacija

U suprotnom slučaju se produžava sa preslikavanjem tako što se čita deskriptor stranice iz tabele stranica i dovlači u jedinicu za preslikavanje. Pored toga u jedinici preslikavanja se postavljaju i indikator da je ulaz važeći na 1 i indikator da je stranica modifikovana na

vrednost indikatora da je stranica modifikovana iz deskriptora stranice. Potom se vraća na korak u kome se proverava da li postoji saglasnost. Pošto se sada utvrđuje da postoji saglasnost, jer se deskriptor nalazi u jedinici preslikavanja, produžava se sa već opisanim koracima za slučaj kada se deskriptor nalazi u jedinici preslikavanja.

2.2.2.3.1.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture operacione jedinice (poglavlje 2.2.2.3.1.1) i dijagrama toka zahteva (poglavlje 2.2.2.3.1.2.12.2.2.1.2.1). Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je identična kao i notacija za algoritam generisanja upravljačkih signala za upravljačku jedinicu procesora ***CPU*** (poglavlje 2.2.2.1.2.2).

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *br* (*if* (**PRQ**) *then* $step_1$ *else* $step_0$)

! U koraku $step_0$ se čeka da se aktivom vrednošću signala **PRQ** procesora ***CPU*** iz registara procesora ***CPU*** u registre jedinice ***TLB*** upišu podaci neophodni za preslikavanje i startuje samo preslikavanje. Pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta virtuelna adresa, broj tekućeg procesa, početna adresa tabele stranica i tip operacije se upisuju u registre $UVA_{23...0}$, $UPR_{2...0}$ i $UPT_{19...0}$ i flip-flop URW bloka *cpu_interfejs*, respektivno, dok se samo broj stranice virtuelne adrese upisuje u registar $UPG_{13...0}$ bloka *mem_interfejs*. Pri aktivnoj vrednosti signala **PRQ** se prelazi na korak step1, dok se u suprotnom slučaju ostaje u koraku $step_0$. !

$step_1$: *if* (**HIT**, **ldUPA**),
if ($\overline{\textbf{HIT}}$, **ldMAR**, **incUPG**),
br (*if* **HIT** *then* step2 *else* step6)

! U korak $step_1$ može da se dođe ili iz koraka $step_0$ ili iz koraka $step_9$. Iz koraka $step_0$ se dolazi po startovanju svakog novog zahteva za preslikavanje. Iz koraka $step_9$ se dolazi po nekom zahtevu za preslikavanje za koji je, pri prethodnom prolasku kroz $step_1$, otkriveno da nema saglasnosti, pa se išlo na dovlačenje deskriptora stranice iz memorije ***MEM*** u jedinicu ***TLB***. U koraku $step_1$ se vrši provera saglasnosti i na osnovu toga formira vrednost signala **HIT** bloka *tag_data*. Ako se pojavi aktivna vrednost signala **HIT**, što znači da je otkrivena saglasnost, generiše se aktivna vrednost signala **ldUPA** bloka *cpu_interfejs* trajanja jedna perioda signala takta. Njome se fizička adresa, formirana od broja bloka pročitanog iz DATA memorije koji dolazi po linijama $\textbf{DATA}_{\textbf{9...0}}$ bloka *tag_data* i adrese reči u bloku virtuelne adrese koja dolazi po linijama $UVA_{9...0}$ bloka *cpu_interfejs*, upisuje u registar $UPA_{19...0}$ bloka *cpu_interfejs*. U slučaju da je neaktivna vrednost signal **HIT**, što znači da nije otkrivena saglasnost, mora da se prvo prođe kroz korake $step_6$, $step_7$, $step_8$ i $step_9$ i da se u njima iz tabele stranica u jedinicu za preslikavanje dovuče deskriptor tekuće stranice, pa da se zatim ponovo dođe u korak step1. Stoga se tada generiše aktivna vrednost signala **ldMAR** trajanja jedna perioda signala takta bloka *mem_interfejs* kojom se u registar $MAR_{19...0}$ upisuje sadržaj registra $UPT_{19...0}$ u kome se nalazi adresa nultog ulaza tabele stranica. Takođe se generiše i aktivna vrednost signala **incUPG** trajanja jedna perioda signala takta bloka *mem_interfejs* kojom se za jedan uvećava sadržaj registra $UPG_{13...0}$. Time se obezbeđuje da se u registru $UPG_{13...0}$ nalazi broj stranice virtuelne adrese plus 1. Ovo se radi zbog toga što se deskriptor *i*-te stranice nalazi u (*i*+1)-om ulazu tabele stranica. Pri aktivnoj vrednosti signala **HIT** se prelazi na korak $step_2$, dok se u suprotnom slučaju prelazi na korak $step_6$. !

$step_2$: *if* ($\overline{\textbf{CH}}$, **URP**),
if (**CH**, **mxMAR**, **ldMAR**),
br (*if* **CH** *then* $step_3$ *else* $step_0$)

! U korak $step_2$ može da se dođe jedino iz koraka $step_1$ i to samo onda kada je u koraku $step_1$ otkrivena saglasnost i preslikavanje uspešno realizovano. U koraku $step_2$ se proverava vrednost signala **CH** bloka *tag_data*, koji može da ima aktivnu vrednosti samo ukoliko je preslikavanje realizovano za operaciju upisa i to za prvu operaciju upisa na tekućoj stranici. U slučaju da je signal **CH** bloka *tag_data* na neaktivnoj vrednosti, generiše se aktivna vrednosta signala **URP** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje uspešno realizovano i da se u registru $UPA_{19...0}$ bloka *cpu_interfejs* nalazi fizička adresa. U slučaju da je signal **CH** na aktivnoj vrednosti, mora da se prođe kroz korake $step_3$, $step_4$ i $step_5$ i da se u njima najpre u bit D deskriptora tekuće stranice u tabeli stranica upiše vrednost 1, pa da se tek onda generiše aktivna vrednosta

signala **URP**. Stoga se u koraku $step_2$ generišu aktivne vrednosti signala **mxMAR** i **ldMAR** bloka *mem_interfejs* kojima se adresa ulaza u tabelu stranica formirana na izlazu sabirača ADD i propušta kroz multiplekser MP1 i upisuje u registar $MAR_{19...0}$. Pri aktivnoj vrednosti signala **CH** se prelazi na korak $step_3$, dok se u suprotnom slučaju prelazi na korak $step_0$. !

$step_3$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* step4 *else* step3)

! U korak $step_3$ se dolazi samo iz koraka $step_2$. U koraku $step_3$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* kojom se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači*, generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs*, čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost očitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_4$, dok se u suprotnom slučaju ostaje u koraku $step_3$. !

$step_4$: **mxMDR**, **ldMDR**, **stD**,
br $step_5$

! U korak $step_4$ se dolazi samo iz koraka $step_3$. U registru $MDR_{15...0}$ bloka *cpu_interfejs* se nalazi pročitan deskriptor stranice čiji bit D treba postaviti na 1. Stoga se bezuslovno generišu aktivne vednosti signala **mxMDR** i **ldMDR** bloka *mem_interfejs* kojima se 16-bitna reč, koja na pozicijama 15, 13…0 ima razrede MDR_{15} i $MDR_{13...0}$ i na poziciji 14 ima 1, propušta kroz multiplekser MP2 i upisuje u registar $MDR_{15...0}$. Pored toga bezuslovno se generiše i aktivna vednost signala **stD** bloka *indikatori* kojom se postavlja na aktivnu vrednost jedan od flip-flopova V_0 do V_7 bloka *indikatori* koji odgovara ulazu jedinice ***TLB*** u kome je prilikom preslikavanja otkrivana saglasnost. Iz koraka $step_4$ se uvek prelazi na korak $step_5$. !

$step_5$: **writeMEM**, **incMEMACC**,
if (**MEMFC**, **URP**),
br (*if* **MEMFC** *then* $step_0$ *else* $step_5$)

! U korak $step_5$ se dolazi samo iz koraka $step_4$. U koraku $step_5$ se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *mem_interfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se modifikovani sadržaj registra $MDR_{15...0}$ bloka *mem_interfejs* upisuje u memoriju ***MEM*** na adresi određenoj sadržajem registra $MAR_{19...0}$ bloka *mem_interfejs*. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači* generiše se aktivna vrednost signala **URP** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje uspešno realizovano i da se u registru $UPA_{19...0}$ bloka *cpu_interfejs* nalazi fizička adresa. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_0$, dok se u suprotnom slučaju ostaje u koraku $step_5$. !

$step_6$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* $step_7$ *else* $step_6$)

! U korak $step_6$ se dolazi iz koraka $step_1$ kada se utvrdi da nema saglasnosti, pa se prelazi na korake dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu za preslikavanje. U koraku $step_6$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs* čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost pročitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč nultog ulaza tabele stranica tekućeg procesa. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_7$, dok se u suprotnom slučaju ostaje u koraku $step_6$. !

$step_7$: *if* (**GR**, **UAV**),
if ($\overline{\textbf{GR}}$, **mxMAR**, **ldMAR**),
br (*if* **GR** *then* $step_0$ *else* $step_8$)

! U korak $step_7$ se dolazi samo iz koraka $step_6$. U koraku $step_7$ se proverava da li se zahtev za preslikavanje odnosi na postojeću stranicu procesa. Signal **GR** bloka *mem_interfejs* predstavlja izlaz GTR komparatora CMP koji poredi broj stranice virtuelne adrese uvećan za jedan iz registra $UPG_{13...0}$ i broj stranica procesa iz registra $MDR_{13...0}$. Ako je signal **GR** na aktivnoj vrednosti, generiše se aktivna vrednost signala **UAV** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice. Ako je siganal **GR** na neaktivnoj vrednosti, produžava se sa koracima dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu jedinicu za preslikavanje tako što se generišu aktivne vrednosti signala **mxMAR** i **ldMAR** bloka *mem_interfejs* kojima se adresa ulaza u tabelu stranica

formirana na izlazu sabirača ADD propušta kroz multiplekser MP1 i upisuje u registar $MAR_{19...0}$. Pri aktivnoj vrednosti signala **GR** se prelazi na koraka $step_0$, dok se u suprotnom slučaju prelazi na korak $step_8$. !

$step_8$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* $step_9$ *else* $step_8$)

! U korak $step_8$ se dolazi samo iz koraka $step_7$. U koraku $step_8$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs* čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost pročitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_9$, dok se u suprotnom slučaju ostaje u koraku $step_8$. !

$step_9$: *if* ($\overline{\textbf{valid}}$, **UPF**),
if (**valid**, **wrTAG**, **wrDATA**, **mxDATA**, **stV**, **wrD**, **mxIND**, **incFIFOCNT**),
br (*if* **valid** *then* $step_1$ *else* $step_0$)

! U korak $step_9$ se dolazi samo iz koraka $step_8$. U koraku $step_9$ se proverava da li se zahtev za preslikavanje odnosi na stranicu procesa koja je u operativnoj memoriji. Signal **valid** bloka *mem_interfejs* predstavlja vrednost V bita deskriptora stranice koji vrednostima 0 i 1 određuje da se stranica ne nalazi i nalazi u operativnoj memoriji, respektivno. Ako je signal **valid** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **UPF** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje za stranicu koja nije u operativnoj memoriji. Ako je siganal **valid** na neaktivnoj vrednosti, produžava se sa koracima dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu za preslikavanje tako što se generišu aktivne vrednosti signala **wrTAG**, **wrDATA**, **mxDATA**, **stV, wrD**, **mxIND** i **incFIFOCNT**. Aktivna vrednost signala **wrTAG** bloka *tag_data* omogućuje da se u TAG memoriju upišu broj procesa i broj stranice sa linija $\textbf{TAGIN}_{\textbf{16...0}}$, dok aktivna vrednost signala **wrDATA** bloka *tag_data* omogućuje da se u DATA memoriju upiše broj bloka sa linija $\textbf{MDR}_{\textbf{13...4}}$ bloka *mem_interfejs*. Adresa ulaza TAG memorije i DATA memorije je određena sadržajem brojača za zamenu $FIFOCNT_{2...0}$ bloka *brojači*. Sadržaj brojača za zamenu $FIFOCNT_{2...0}$ se na adresne linije memorije TAG vodi direktno, dok se na adresne linije linije memorije DATA vodi sa izlaza multipleksera MP kroz koji se propušta aktivnom vrednošću signala **mxDATA**. Aktivnom vrednošću signala **stV** bloka *indikatori* se jedan od flip-flopova V_0 do V_7, adresiran sadržajem brojača za zamenu $FIFOCNT_{2...0}$, postavlja na 1. Aktivnom vrednošću signala **wrD** bloka *indikatori* jedan od flip-flopova D_0 do D_7, adresiran sadržajem brojača za zamenu $FIFOCNT_{2...0}$, se postavlja na vrednost indikatora D dovučenog deskriptora stranice koji se nalazi u razredu MDR_{14} bloka *mem_interfejs*. Sadržaj brojača za zamenu $FIFOCNT_{2...0}$ se vodi na ulaze dekodera DC2 i DC3 sa izlaza multipleksera MP kroz koji se propušta aktivnom vrednošću signala **mxIND**. Aktivnom vrednošću signala **incFIFOCNT** bloka *brojači* uvećava se sadržaj brojača za zamenu $FIFOCNT_{2...0}$. Pri aktivnoj vrednosti signala **valid** se prelazi na korak $step_1$, dok se u suprotnom slučaju prelazi na korak $step_0$. !

2.2.2.3.1.2.3. Struktura upravljačke jedinice

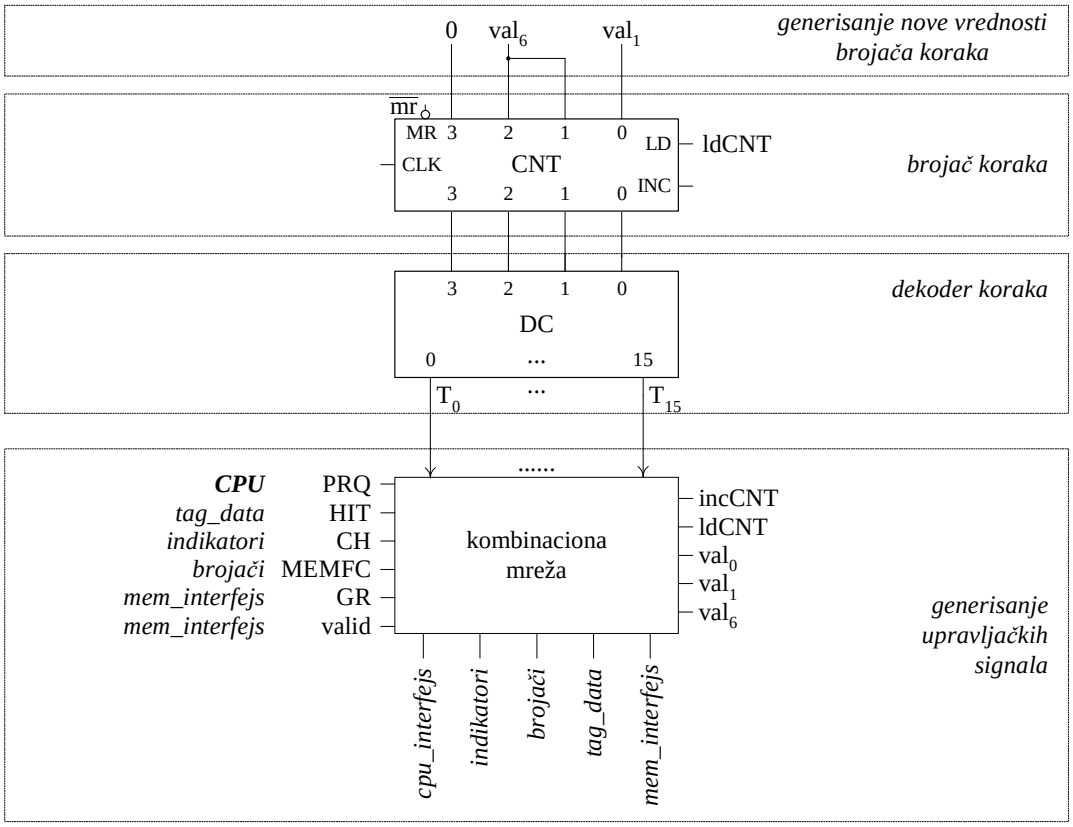
Upravljačka jedinica (slika 23) se sastoji od sledećih blokova:

- blok *generisanje nove vrednosti brojača koraka*,
- blok *brojač koraka*,
- blok *dekoder koraka* i
- blok *generisanje upravljačkih signala*.

Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.

Blok *generisanje nove vrednosti brojača koraka* služi za generisanje vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje 2.2.2.3.1.2.2) se utvrđuje da su 0, 1 i 6 vrednosti koje treba upisati u brojač CNT da bi se realizovala odstupanja od sekvencijalnog generisanja upravljačkih signala. Te vrednosti se formiraju na ulazima 3, 2, 1 i 0 brojača CNT pomoću signala $\textbf{val}_\textbf{0}$, $\textbf{val}_\textbf{1}$ i $\textbf{val}_\textbf{6}$ pri čemu signal $\textbf{val}_\textbf{0}$ ima vrednost 1 samo onda kada treba upisati 0, signal $\textbf{val}_\textbf{1}$ ima vrednost 1 samo onda kada treba upisati 1 i signal $\textbf{val}_\textbf{6}$ ima vrednost 6 samo

onda kada treba upisati 6. Time se obezbeđuje da na ulazima 3, 2, 1 i 0 brojača CNT budu vrednosti 0, 0, 0 i 0 onda kada je signal **val$_0$** ima vrednost 1, vrednosti 0, 0, 0 i 1 onda kada je signal **val$_1$** ima vrednost 1 i vrednosti 0, 1, 1 i 0 onda kada je signal **val$_6$** ima vrednost 1.



Slika 23 Struktura upravljačke jedinice

Blok *brojač koraka* sadrži brojač CNT. Brojač CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač CNT može da radi u sledećim režimima:

- režim inkrementiranja,
- režim skoka i
- režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača CNT za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **incCNT**. Signal **incCNT** je uvek neaktivan sem kada treba obezbediti režim inkrementiranja.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač CNT. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ldCNT**. Signal **ldCNT** je uvek neaktivan sem kada treba obezbediti režim skoka.

U režimu mirovanja pri pojavi signala takta ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **incCNT** i **ldCNT**. Ovi signali su neaktivni kada se čeka pojava aktivne vrednosti signala startovanja zahteva za preslikavanje

PRQ procesora ***CPU*** ili pojava aktivne vrednosti signala **MEMFC** bloka *brojači* kojim se označava da je završena operacija čitanja iz memorije ***MEM*** ili upisa u memoriju ***MEM***

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali $\mathbf{T_0}$ do $\mathbf{T_{15}}$ na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$, itd.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$ do $\mathbf{T_{15}}$ koji dolaze sa bloka *dekoder koraka* upravljačke jedinice, signala logičkih uslova koji dolaze iz blokova operacione jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2) generišu upravljačke signale. Upravljački signali se generišu na identičan način kao i upravljački signali procesora ***CPU*** (poglavlje 2.2.2.1.2.2).

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:

- upravljačke signale operacione jedinice ***oper_jed*** i
- upravljačke signale upravljačke jedinice ***uprav_jed***.

Upravljački signali operacione jedinice ***oper_jed*** se daju posebno za svaki blok operacione jedinice i to:

- blok *cpu_interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag_data i*
- blok *cpu_ interfejs.*

Upravljački signali bloka *cpu_interfejs* se generišu na sledeći način:

- $\mathbf{ldUPA = T_1 \cdot HIT}$
- $\mathbf{URP = T_2 \cdot \overline{CH} + T_5 \cdot MEMFC}$
- $\mathbf{UAV = T_7 \cdot GR}$
- $\mathbf{UPF = T_9 \cdot \overline{valid}}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **HIT** — blok *tag_data*,
- **CH** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **GR** — blok *mem_interfejs* i
- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *indikatori* se generišu na sledeći način:

- $\mathbf{stD = T_4}$
- $\mathbf{stV = T_9 \cdot valid}$
- $\mathbf{wrD = T_9 \cdot valid}$
- $\mathbf{mxIND = T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *brojači* se generišu na sledeći način:

- $\mathbf{incMEMACC = T_3 + T_5 + T_6 + T_8}$

- **incFIFOCNT** = $\mathbf{T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *tag_data* se generišu na sledeći način:

- **wrTAG** = $\mathbf{T_9 \cdot valid}$
- **wrDATA** = $\mathbf{T_9 \cdot valid}$
- **wxDATA** = $\mathbf{T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *mem_interfejs* se generišu na sledeći način:

- **incUPG** = $\mathbf{T_1 \cdot \overline{HIT}}$
- **mxMAR** = $\mathbf{T_2 \cdot CH + T_7 \cdot \overline{GR}}$
- **ldMAR** = $\mathbf{T_2 \cdot CH + T_7 \cdot \overline{GR} + T_1 \cdot \overline{HIT}}$
- **mxMDR** = $\mathbf{T_4}$
- **ldMDR** = $\mathbf{T_3 \cdot MEMFC + T_4 + T_6 \cdot MEMFC + T_8 \cdot MEMFC}$
- **writeMEM** = $\mathbf{T_5}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **HIT** — blok *tag_data*
- **CH** — blok *tag_data*.
- **GR** — blok *mem_interfejs* i
- **MEMFC** — blok *brojači*.

Upravljački signali upravljačke jedinice ***uprav_jed*** se generišu na sledeći način:

- **ldCNT** = $\mathbf{val_1 + val_0 + val_6}$
- **incCNT** = $\mathbf{T_0 \cdot PRQ + T_1 \cdot HIT + T_2 \cdot CH + T_3 \cdot MEMFC + T_4 + T_6 \cdot MEMFC + T_7 \cdot \overline{GR} + T_8 \cdot MEMFC}$
- $\mathbf{val_6}$ = $\mathbf{T_1 \cdot \overline{HIT}}$
- $\mathbf{val_0}$ = $\mathbf{T_2 \cdot \overline{CH} + T_5 \cdot MEMFC + T_7 \cdot GR + T_9 \cdot \overline{valid}}$
- $\mathbf{val_1}$ = $\mathbf{T_9 \cdot \overline{valid}}$

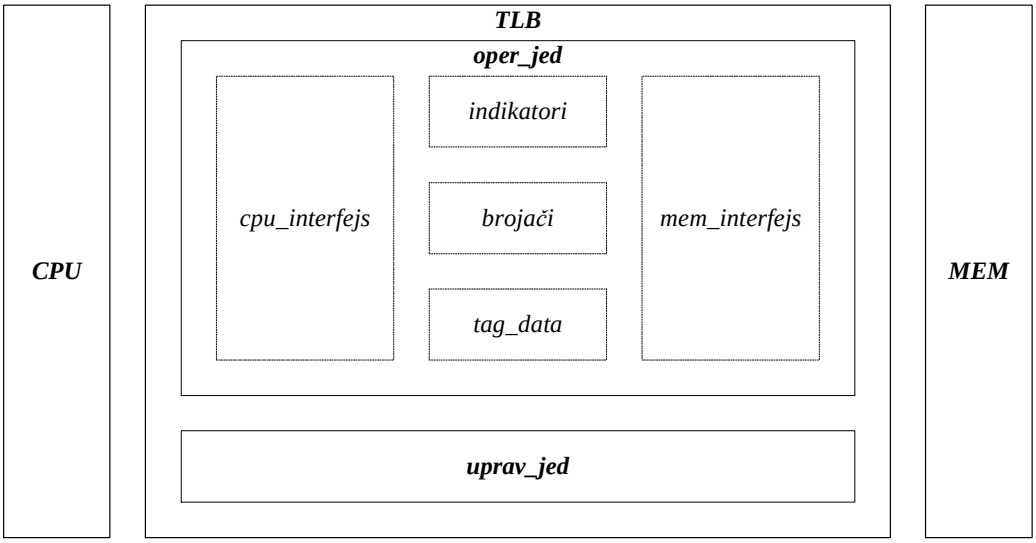
Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **PRQ** — blok *cpu_interfejs*,
- **HIT** — blok *tag_data*,
- **CH** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **GR** — blok *mem_interfejs*.

2.2.2.3.2. JEDINICA SA DIREKTNIM PRESLIKAVANJEM

Jedinica sa direktnim preslikavanjem ***TLB*** (slika 24) se sastoji iz:

- operacione jedinice ***oper_jed*** i
- upravljačke jedinice ***uprav_jed***.



Slika 24 Jedinica sa direktnim preslikavanjem

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu asocijativnog preslikavanja virtuelne adrese u fizičku adresu. Sve sekvencijalne mreže unutar ***TLB*** su sinhrone.

2.2.2.3.2.1. OPERACIONA JEDINICA

Operaciona jedinica ***oper_jed*** se sastoji iz sledećih blokova:

- blok *cpu_interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag_data i*
- blok *cpu_ interfejs.*

Blok *cpu_interfejs* služi za razmenu podataka i upravljačkih signala sa procesorom ***CPU***.

Blok *indikatori* služi za vođenje evidencije za svaki od 8 ulaza jedinice o tome da li je u ulazu važeći descriptor i da li je sadržaj stranice modifikovan.

Blok *brojači* služi za odbrojavanje onoliko perioda signala takta koliko je vreme pristupa memoriji ***MEM***.

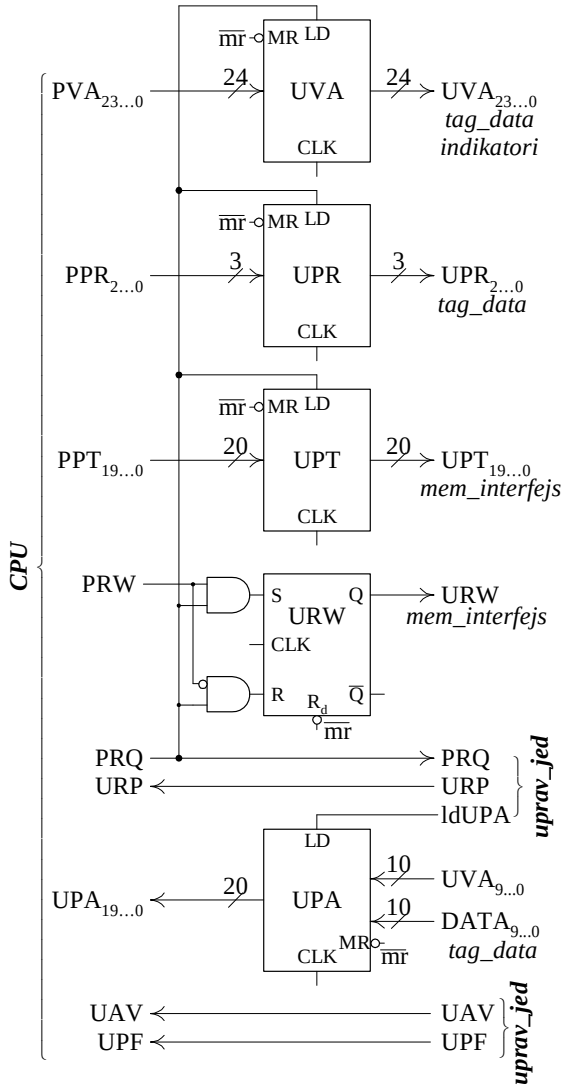
Blok *tag_data* služi za čuvanje 8 deskriptora stranica.

Blok *cpu_ interfejs* služi za razmenu podataka i upravljačkih signala sa memorijom ***MEM***.

Struktura i opis blokova operacione jedinice ***oper_jed*** se daju u daljem tekstu.

2.2.2.3.2.1.1. Blok *cpu_interfejs*

Blok *cpu_interfejs* (slika 25) sadrži registre $UVA_{23...0}$, $UPR_{2...0}$, $UPT_{19...0}$ i $UPA_{19...0}$ i flip-flop URW.



Slika 25 Blok *cpu_interfejs*

Registar $UVA_{23...0}$ (*Unit Virtual Address register*) služi za čuvanje virtuelne adrese koju jedinica ***TLB*** treba da preslika u fizičku adresu. Virtuelna adresa dolazi na ulaze registra $UVA_{23...0}$ po linijama $\mathbf{PVA_{23...0}}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UVA_{23...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Razredi registra $UVA_{12...10}$ se vode u blok *tag_data* gde se koriste za adresiranje ulaza memorije TAGDATA i u blok *indikatori* gde se koriste za adresiranje flip-flopova V_0 do V_7 i D_0 do D_7. Razredi registra $UVA_{23...13}$ se takođe vode u blok *tag_data* gde se koriste za utvrđivanje saglasnosti sa adresiranim ulazom memorije TAGDATA i upisivanje u memoriju TAGDATA prilikom dovlačenja deskriptora stranice. Pored toga razredi registra $UVA_{9...0}$ se vode na ulaze 9 do 0 registra $UPA_{19...0}$ gde se koriste kao adresa u bloku prilikom formiranja fizičke adrese.

Registar $UPR_{2...0}$ (*Unit PRoces register*) služi za čuvanje broja tekućeg procesa. Broj tekućeg procesa dolazi na ulaze registra $UPR_{2...0}$ po linijama $\mathbf{PPR_{2...0}}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UPR_{2...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj registra $UPR_{2...0}$ se vodi u blok *tag_data* gde se koristi za utvrđivanje saglasnosti sa adresiranim

ulazom memorije TAGDATA i upisivanje u memoriju TAGDATA prilikom dovlačenja deskriptora stranice.

Registar $UPT_{19...0}$ (*Unit Page map Table address register*) služi za čuvanje početne adrese tabele stranica tekućeg procesa. Adresa dolazi na ulaze registra $UPT_{19...0}$ po linijama $PPT_{19...0}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UPT_{19...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj registra $UPT_{19...0}$ se po vodi u blok *mem_interfejs* gde se koristi kao adresa za čitanje nultog ulaza tabele stranica radi provere da li se radi o pokušaju pristupa nepostojećoj stranici (*access violation*) i za formiranje adrese deskriptora stranice u tabeli stranica radi čitanja deskriptora ili upisa u deskriptor.

Registar $UPA_{19...0}$ (*Unit Physical Address register*) služi za čuvanje fizičke adrese koja se generiše u slučaju uspešnog preslikavanja virtuelne adrese u fizičku adresu. Fizička adresa se generiše tako što se na ulaze 19 do 10 registra $UPA_{19...0}$ dovodi sadržaj sa izlaznih linija $\mathbf{DATA_{9...0}}$ memorije TAGDATA bloka *tag_data*, a na ulaze 9 do 0 registra $UPA_{19...0}$ dovodi sadržaj sa linija $\mathbf{UVA_{9...0}}$. Generisana fizička adresa se u registar $UPA_{19...0}$ upisuje pri aktivnoj vrednosti signala **ldUPA**. Sadržaj registra $UPA_{19...0}$ se vodi u blok *tlb_interfejs* procesora ***CPU***.

Flip-flop URW (*Unit Read Write flag*) vrednostima 0 i 1 predstavlja indikator operacije čitanja ili upisa, respektivno, koja treba da se realizuje po preslikavanju virtuelne adrese u fizičku adresu. Vrednost indikatora dolazi na ulaze flip-flopa URW po liniji PRW iz bloka *tlb_interfejs* procesora ***CPU***, a u flip-flop URW se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj flip-flopa URW se vodi u blok *indikatori*.

Upravljački signal **PRQ** (*Processor ReQuest*) koji dolazi iz bloka *tlb_inerfejs* procesora ***CPU*** se koristi da procesor ***CPU*** aktivnom vrednošću ovog signala trajanja jedna perioda izvrši upis sadržaja sa linija $\mathbf{PVA_{23...0}}$, $\mathbf{PPR_{2...0}}$, $\mathbf{PPT_{19...0}}$ i **PRW** bloka *tlb_interfejs* u registre $UVA_{23...0}$, $UPR_{2...0}$ i $UPT_{19...0}$ i flip_flop URW jedinice ***TLB*** i da se startuje preslikavanje.

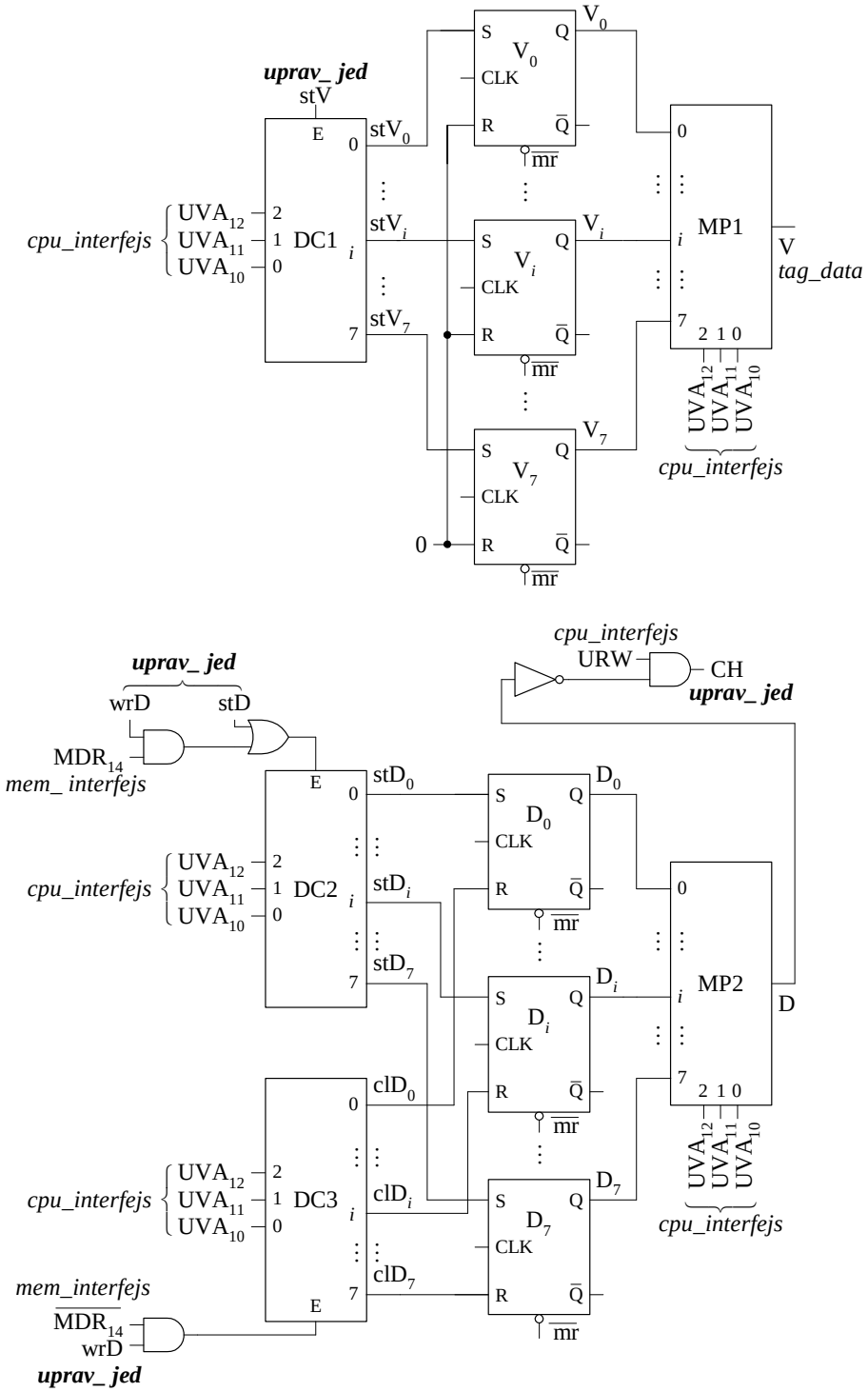
Upravljački signal **URP** (*Unit RePly*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da je peslikavanje uspešno realizovano i da se na linijama $\mathbf{UPA_{19...0}}$ nalazi generisana fizička adresa.

Upravljački signal **UAV** (*Unit Address Violation*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa nepostojećoj stranici (*access violation*).

Upravljački signal **UPF** (*Unit Page Fault*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*).

2.2.2.3.2.1.2. Blok *indikatori*

Blok *indikatori* (slika 26) se sastoji od flip_flopova V_0 do V_7 sa dekoderom DC1 i multiplekserom MP1 i flip_flopova D_0 do D_7 sa dekoderima DC2 i DC3 i multiplekserom MP2.



Slika 26 Blok *indikatori*

Flip-flopovi V_0 do V_7 (*Valid*) se koriste kao indikatori za svaki od ulaza 0 do 7 jedinice ***TLB*** da li je ulaz važeći ili nije važeći. Flip-flop V_i, i=0,…7, se postavlja na 1 aktivnom vrednošću signala $\mathbf{stV}_i$ trajanja jedna perioda signala takta prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u i-ti ulaz jedinice ***TLB***.

Dekoder DC1 se koristi da se, formiranjem aktivne vrednosti jednog od signala $\mathbf{stV_0}$ do $\mathbf{stV_7}$ na izlazima dekodera DC1, jedan od flip-flopova V_0 do V_7 postavi na 1. Ovo se realizuje

prilikom dovlačenja deskriptora stranice generisanjem aktivne vrednosti signala **stV** trajanja jedna perioda signala takta koji se pojavljuje kao aktivna vrednost na izlazu dekodera DC1 određenom sadržajem razreda $\mathbf{UVA_{12...10}}$ koji se dovodi na ulaze dekodera DC1 iz bloka *cpu_interfejs*.

Multiplekser MP1 se koristi da se signalima $\mathbf{UVA_{12...10}}$ iz bloka *cpu_interfejs*, koji predstavljaju broj *i*-tog ulaza, *i*=0,…7, u kome se proverava saglasnost, na izlazu multipleksera MP1 kao signal **V** selektuje vrednost flip-flopa V_i. Signal **V** se vodi u blok *tag_data* gde sa signalom **EQL** sa izlaza komparatora CMP formira signal saglasnosti **HIT**.

Flip-flopovi D_0 do D_7 (*Dirty*) se koriste kao indikatori za svaki od ulaza 0 do 7 jedinice ***TLB*** da li je bilo upisa u stranicu čiji se deskriptor nalazi u datom ulazu ili ne. Flip-flop D_i, *i*=0,…7, se postavlja na 1 aktivnom vrednošću signala $\mathbf{stD_i}$ trajanja jedna perioda signala i na 0 aktivnom vrednošću signala $\mathbf{clD_i}$ trajanja jedna perioda signala takta. Ovi signali se koriste da se prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti ulaz jedinice ***TLB*** flip-flop D_i postavi na vrednost bita D (*Dirty*) deskriptora stranice, koji se nalazi u razredu MDR_{14} registra $MDR_{15...0}$ bloka *mem_interfejs*, i da se prilikom generisanja fizičke adrese radi upisa, flip-flop D_i postavi na 1.

Dekoder DC2 se koristi da se, formiranjem aktivne vrednosti jednog od signala $\mathbf{stD_0}$ do $\mathbf{stD_7}$ na izlazima dekodera DC2 trajanja jedna perioda signala takta, jedan od flip-flopova D_0 do D_7 postavi na 1. Dekoder DC3 se koristi da se, formiranjem aktivne vrednosti jednog od signala $\mathbf{clD_0}$ do $\mathbf{clD_7}$ na izlazima dekodera DC3 trajanja jedna perioda signala takta, jedan od flip-flopova D_0 do D_7 postavi na 0. Prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-tom ulazu jedinice ***TLB***, *i*=0,…7, treba da se generiše aktivna vrednost signala **stD*i*** i da se flip-flop D*i* postavi na 1. Tada se generiše signal **stD** koji se pojavljuje kao aktivna vrednost signala **stD*i*** na izlazu dekodera DC2 određenom signalima $\mathbf{UVA_{12...10}}$ bloka *cpu_interfejs* koji se dovode na ulaze dekodera DC2. Prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti ulaz jedinice ***TLB***, treba da se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, generiše aktivna vrednost signala $\mathbf{stD_i}$ ili $\mathbf{clD_i}$, respektivno, i da se flip-flop D_i postavi na vrednost bita D (*Dirty*) deskriptora stranice. Tada se generiše signal **wrD** koji se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, pojavljuje kao aktivna vrednost ili signala $\mathbf{stD_i}$ na izlazu dekodera DC2 ili signala $\mathbf{clD_i}$ na izlazu dekodera DC3 određenom signalima $\mathbf{UVA_{12...10}}$ bloka *cpu_interfejs* koji se dovode na ulaze dekodera DC2 i DC3.

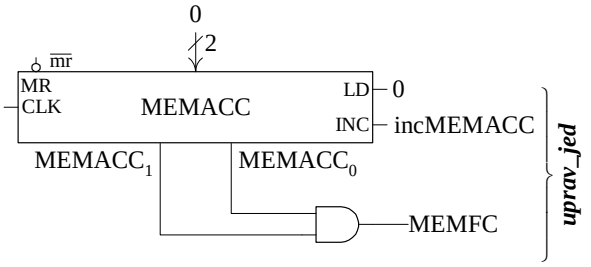
Multiplekser MP2 se koristi da se signalima signalima $\mathbf{UVA_{12...10}}$ bloka *cpu_interfejs*, koji predstavljaju broj *i*-tog ulaza, *i*=0,…7, u kome se proverava saglasnost, na izlazu multipleksera MP2 kao signal **D** selektuje indikator D_i. Signal **D** sa signalom **URW** bloka *cpu_interfejs* formira signal **CH** (*change*), pri čemu signal **URW** vrednostima 0 i 1 ukazuje da li sa preslikane fizičke adrese treba realizovati čitanje ili upis, respektivno. Signal **CH** ima vrednost 1 ukoliko signal **URW** ima vrednost 1, jer se radi o upisu, i ukoliko signal **D** ima vrednost 0, jer nije bilo upisa u stranicu čiji se deskriptor nalazi u *i*-tom ulazu jedinice ***TLB***. Tada treba bit D deskriptora date stranice u tabeli stranica postaviti na 1. U svim ostalim slučajevima signal **CH** ima vrednost 0 i tada ne treba ništa raditi.

Signal **CH** se koristi prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-tom ulazu jedinice ***TLB***. Ukoliko je bit D 0, to znači da nije bilo upisa u datu stranicu i da treba bit D deskriptora date stranice u tabeli stranica postaviti na 1. Kako je za operaciju upisa indikator URW bloka *cpu_interfejs* 1, to je i signal **CH** 1. Ukoliko je bit D 1, to znači da je bilo upisa u datu stranicu i da je bit D deskriptora date stranice u tabeli

stranica već postavljen 1. Kako je za operaciju upisa indikator URW bloka *cpu_interfejs* 1, to je i signal **CH** 0.

2.2.2.3.2.1.3. Blok *brojači*

Blok *brojači* (slika 27) se sastoji od brojača vremena pristupa operativnoj memoriji $MEMACC_{1...0}$ (*MEMory ACCess time*).



Slika 27 Blok *brojači*

Brojač $MEMACC_{1...0}$ se koristi da odbroji četiri signala takta kod obraćanja jedinice ***TLB*** memoriji ***MEM*** radi čitanja ili upisa, jer je usvojeno da je vreme pristupa memoriji ***MEM*** četiri periode signala takta. Inkrementiranje brojača se realizuje pri aktivnoj vrednosti signala **incMEMACC**. Signal **MEMFC** (*MEMory Function Completed*) postaje aktivan na treći signal takta kada brojač $MEMACC_{1...0}$ inkrementiranjem pređe u stanje tri. Na četvrti signal takta brojač $MEMACC_{1...0}$ se vraća na stanje nula, a signal **MEMFC** postaje neaktivan. Signal **MEMFC** služi upravljačkoj jedinici ***uprav_jed*** kao indikacija da je pristup memoriji ***MEM*** završen.

2.2.2.3.2.1.4. Blok *tag_data*

Blok *tag_data* (slika 28) sadrži RAM memoriju TAGDATA, komparator CMP i logičko I kolo.

Memorija TAGDATA čuva za svaki od 8 ulaza jedinice za preslikavanje broj procesa, broj stranice čiji se descriptor trenutno nalazi u datom ulazu memorije DATA i deskriptor stranice koji predstavlja broj bloka u koji je preslikana stranica. Kapacitet memorije TAGDATA je 8 reči širine 24 bita. Memorija TAGDATA ima sledeće ulazne i izlazne linije:

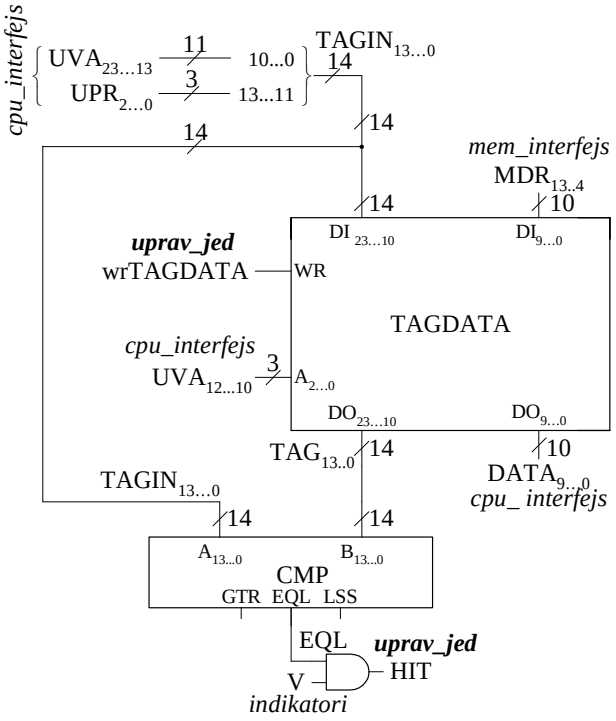
- adresne linije $\mathbf{A_{2...0}}$,
- ulazne linije podataka $\mathbf{DI_{23...0}}$,
- izlazne linije podataka $\mathbf{DO_{23...0}}$ i
- upravljačku liniju **WR**.

Na adresne linije $\mathbf{A_{2...0}}$ se dovode signali $\mathbf{UVA_{12...10}}$ bloka *cpu_interfejs* koji predstavljaju 3 najmlađa bit broja stranice. Ulazne linije podataka su grupisane u linije $\mathbf{DI_{23...10}}$ i linije $\mathbf{DI_{9...0}}$. Na ulazne linije podataka $\mathbf{DI_{23...10}}$ se vode signali $\mathbf{TAGIN_{13...0}}$, gde signali $\mathbf{TAGIN_{13...11}}$ predstavljaju signale broja procesa $\mathbf{UPR_{2...0}}$ bloka *cpu_interfejs* i signali $\mathbf{TAGIN_{10...0}}$ predstavljaju signale 11 najstarijih bitova broja stranice $\mathbf{UVA_{23...13}}$ bloka *cpu_interfejs*. Na ulazne linije podataka $\mathbf{DI_{9...0}}$ se vode signali $\mathbf{MDR_{13...4}}$ pročitanog deskriptora stranice iz registra $\mathbf{MDR_{15...0}}$ bloka *mem_interfejs* koji predstavljaju broj bloka u koji se stranica preslikava. Izlazne linije podataka su grupisane u linije $\mathbf{DO_{23...10}}$ i linije $\mathbf{DO_{9...0}}$. Na izlaznim linijama podataka $\mathbf{DO_{23...10}}$ se pojavljuju signali $\mathbf{TAG_{13...0}}$, gde signali $\mathbf{TAG_{13...11}}$ predstavljaju signale broja procesa i signali $\mathbf{TAGIN_{10...0}}$ predstavljaju signale 11 najstarijih bitova broja stranice adresiranog ulaza. Na izlaznim linijama podataka $\mathbf{DO}_{9...0}$ se pojavljuju

signali **DATA**$_{9...0}$ koja predstavlja broj bloka u koji se preslikava stranica. Upis u memoriju TAGDATA se realizuje aktivnom vrednošću signala **wrTAGDATA**, a čitanje njegovom neaktivnom vrednošću. Memoriji TAGDATA se pristupa u sledećim slučajevima:

1. *Utvrđivanje saglasnosti i formiranja fizičke adrese pri utvrđenoj saglasnosti*. Iz memorije TAGDATA se čita. Pročitani sadržaj sa linija **TAG$_{13...0}$** se vodi na komparator CMP gde se upoređuje sa sadržajem sa linija **TAGIN$_{13...0}$**. Ukoliko saglasnost postoji, pročitani sadržaj na linijama **DATA**$_{9...0}$ predstavlja broj bloka u koji se preslikava stranica, pa se sadržaj sa linija **DATA**$_{9...0}$ vodi u blok *cpu_interfejs* na ulaze 19…10 registra URA$_{19...10}$ radi formiranja fizičke adrese.

2. *Upisivanje broja procesa, broja stranice i broja bloka kod dovlačenja deskriptora stranice*. U memoriju TAGDATA se upisuje. Sadržaj koji se upisuje je **TAGIN$_{13...0}$** i **MDR$_{13...4}$**, gde signali **TAGIN$_{13...11}$** predstavljaju signale broja procesa, signali **TAGIN$_{10...0}$** predstavljaju signale 11 najstarijih bitova broja stranice i signali **MDR$_{13...4}$** predstavljaju signale broja bloka u koji se stranica preslikava.

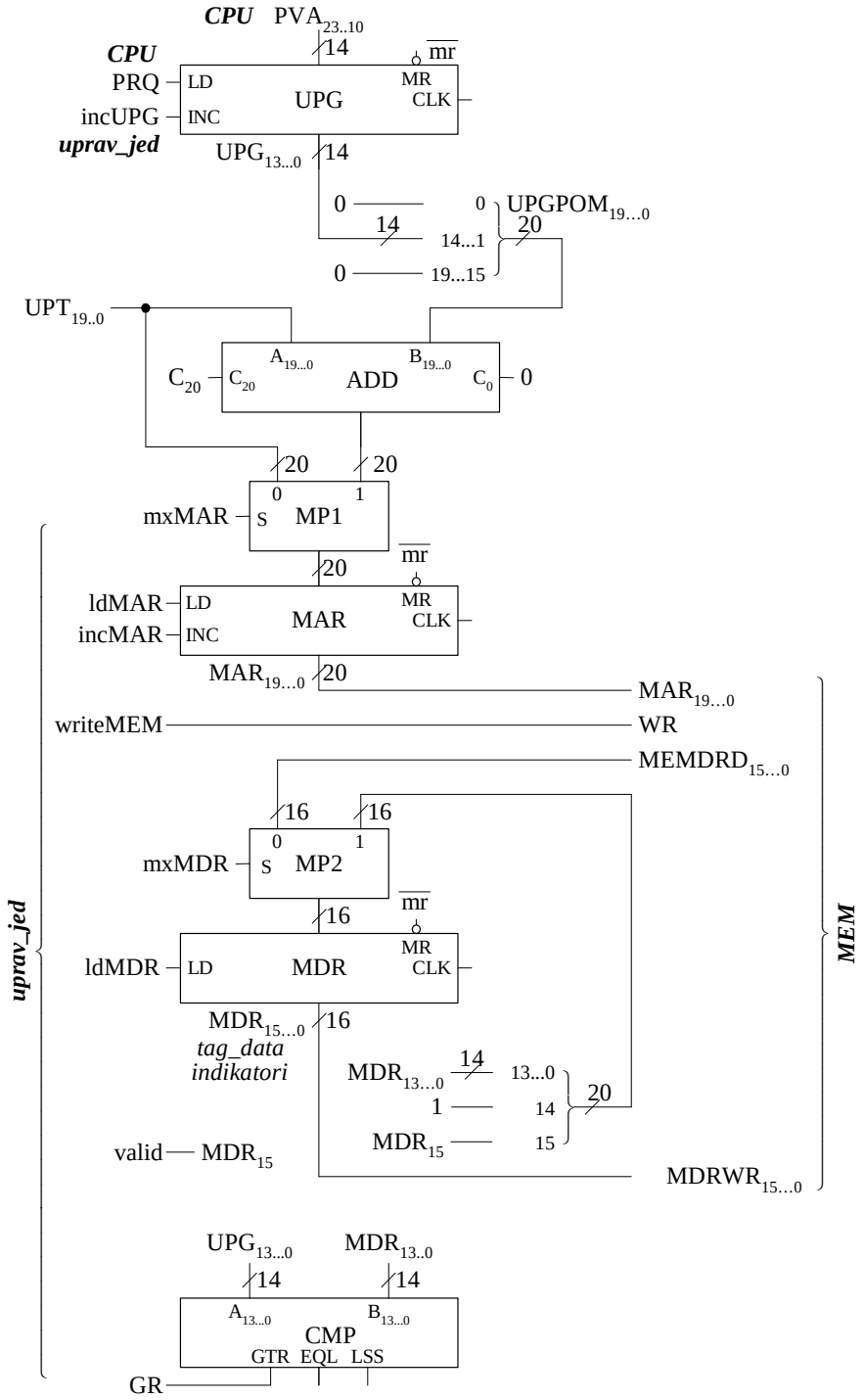


Slika 28 Blok *tag_data*

Komparator CMP poredi sadržaj na linijama **TAGIN$_{13...0}$**, gde signali **TAGIN$_{13...11}$** predstavljaju signale broja procesa i signali **TAGIN$_{10...0}$** predstavljaju signale 11 najstarijih bitova broja stranice zahteva za preslikavanje, i sadržaj na linijama **TAG$_{13...0}$**, gde signali **TAG$_{13...11}$** predstavljaju signale broja procesa i signali **TAGIN$_{10...0}$** predstavljaju signale 11 najstarijih bitova broja stranice adresiranog ulaza memorije TAGDATA. Ukoliko se sadržaji slažu, signal **EQL** na izlazu komparatora CMP ima vrednost 1. Ukoliko je i signal **V** bloka *indikatori* 1, što ukazuje da je adresirani ulaz memorije TAGDATA važeći, na izlazu I kola se formira vrednost 1 signal **HIT**, čime se utvrđuje da postoji saglasnost.

2.2.2.3.2.1.5. Blok *mem_interfejs*

Blok *mem_interfejs* (slika 29) sadrži registar MAR$_{19...0}$ sa multiplekserom MP1, registar MDR$_{15...0}$ sa multiplekserom MP2, registar UPG$_{13...0}$, sabirač ADD i komparator CMP.



Slika 29 Blok *mem_interfejs*

Registar $UPG_{13...0}$ (*Unit PaGe register*) služi za čuvanje broja stranice virtuelne adrese. Broj stranice dolazi po linijama $PVA_{23...10}$ procesora ***CPU*** i upisuje se u registar $UPG_{13...0}$ pri aktivnoj vednosti signala **PRQ** procesora ***CPU***. Sadržaj registra $UPG_{13...0}$ se inkrementira pri aktivnoj vednosti signala **incUPG** trajanja jedna perioda signala takta. Ovaj registar se koristi za formiranje signala pomeraja $\mathbf{UPGPOM_{19...0}}$ u odnosu na početnu adresu tabele stranica sa kojim treba očitati descriptor stranice iz tabele stranica. Pošto je veličina jednog ulaza u tabeli stranica dve reči i pošto se u nultom ulazu tabele stranica nalazi ukupan broj stranica procesa

na koji se tabela odnosi, pomeraj $UPGPOM_{19...0}$ se formira kao $(UPG_{13...0} +1)\cdot 2$. Zbog toga se aktivnom vrednošću signala **incUPG** najpre inkrementira sadržaj registra $UPG_{13...0}$, pa se posle toga sadržaj registra $UPG_{13...0}$ pomeren ulevo za jedno mesto, čime se vrši množenje broja stranice sa dva, koristi za formiranje signala pomeraja $\mathbf{UPGPOM_{19...0}}$.

Sabirač ADD se koristi za izračunavanje adrese sa koje treba iz tabele stranica pročitati deskriptor stranice. Na ulaze $A_{19...0}$ sabirača se dovodi sadržaj registra adrese tabele stranica $UPT_{19...0}$ bloka *cpu_interfejs*, a na ulaze $B_{19...0}$ sadržaj signala pomeraja $\mathbf{UPGPOM_{19...0}}$. Na izlazima sabirača ADD se formira adresa sa koje treba pročitati descriptor stranice.

Registar $MAR_{19...0}$ (*Memory Address Register*) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba pročitati podatak, koji može biti ili niža reč deskriptora stranice ili niža reč nultog ulaza tabele stranica, ili adrese lokacije memorije ***MEM*** u koju treba upisati nižu reč deskriptora stranice u kojoj je bit D izmenjen na 1. Adresa se upisuje u registar $MAR_{19...0}$ pri aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje može biti ili sadržaj registra $UPT_{19...0}$, koji predstavlja adresu niže reči nultog ulaza u tabeli stranica, ili sadržaj formiran na izlazima sabirača ADD, koji predstavlja adresu niže reči deskriptora stranice u tabeli stranica. Selekcija jedne od te dve vrednosti kroz multiplekser MP1 se realizuje upravljačkim signalom **mxMAR**. Sadržaj registra $MAR_{19...0}$ se vode na adresne linije memorije ***MEM***.

Registar $MDR_{15...0}$ (*Memory Data Register*) služi za čuvanje podatka koji je pročitan iz memorije ***MEM*** i za čuvanje podatka koji treba upisati u memoriju ***MEM***. Podatak pročitan iz memorije ***MEM***, koji može biti ili niža reč deskriptora stranice ili niža reč nultog ulaza tabele stranica, dolazi po linijama podataka $\mathbf{MEMDRD_{15...0}}$ iz memorije ***MEM***. Podatak koji treba upisati u memoriju ***MEM*** je sadržaj registra $MDR_{15...0}$ u kome je bit 14 postavljen na 1 i predstavlja nižu reč deskriptora stranice u kojoj je bit D postavljen na 1. Podatak se upisuje u registar $MDR_{15...0}$ na signal takta pri aktivnoj vrednosti signala **ldMDR**. Selekcija jedne od te dve vrednosti kroz multiplekser MP2 se realizuje upravljačkim signalom **mxMDR**. Izlazi registra $MDR_{15...0}$ se vode na ulazne linije podataka memorije ***MEM***. Izlazi registra $MDR_{13...4}$ se vode i u memoriju DATA bloka *tag_data* radi upisa broja bloka deskriptora stranice u memoriju DATA, izlaz MDR_{14} se vodi u blok *indikatori* radi upisa bita D deskriptora stranice u odgovarajući flip-flop D_7 do D_0 i izlazi $MDR_{15...0}$ se, sa razredom MDR_{14} postavljenim na 1, preko multipleksera MP2 vraćaju u registar $MDR_{15...0}$.

Komparator CMP poredi broj stranice virtuelne adrese uvećan za jedan iz registra $UPG_{13...0}$ i broj stranica procesa iz razreda 13 do 0 registra $MDR_{15...0}$, koji predstavljaju broj stranica procesa pročitan iz nultog ulaza tabele stranica, i na izlazu GTR generiše signal greške **GR** zbog zahteva za preslikavanje nepostojeće stranice. Signal **GR** se koristi kao signal logičkog uslova u upravljačkoj jedinici. Ako je signal **GR** na aktivnoj vrednosti, generiše se aktivna vrednost signala **UAV** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice.

Prilikom čitanja deskriptora stranice u razredu MDR_{15} se nalazi indikator koji ukazuje da li je stranica u operativnoj memoriji. Signal iz razreda MDR_{15} daje signal greške **valid** zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji. Signal **valid** se koristi kao signal logičkog uslova u upravljačkoj jedinici. Ako je signal **valid** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **UPF** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji.

2.2.2.3.2.2. UPRAVLJAČKA JEDINICA

U ovom poglavlju se daju dijagram toka zahteva, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.

2.2.2.3.2.2.1. Dijagram toka zahteva

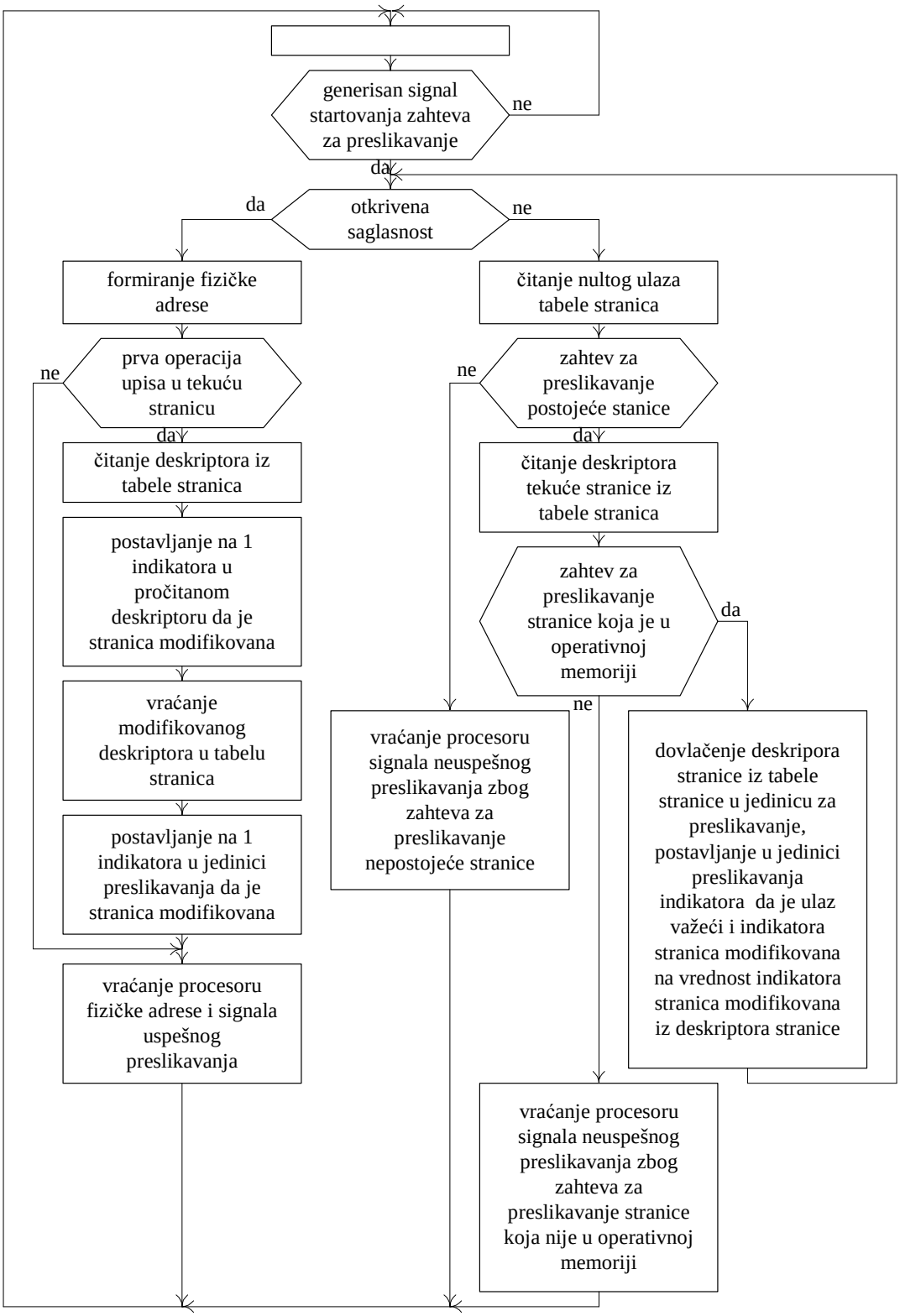
Na početku dijagrama toka se čeka pojava aktivne vrednosti signala startovanja zahteva za preslikavanje (slika 30). Pri njenom pojavljivanju iz procesora se u registre jedinice za preslikavanje upisuju virtuelna adresa, broj procesa, početna adresa tabele stranica i indikator operacije čitanja ili upisa i startuje zahtev u jedinici za preslikavanje.

Najpre se proverava da li postoji saglasnost virtuelne adrese sa sadržajem nekog od ulaza jedinice za preslikavanje, čime se utvrđuje da li se deskriptor stranice nalazi u jedinici preslikavanja.

Ukoliko postoji saglasnost vrši se formiranje fizičke adrese. Zatim se proverava da li je preslikavanje realizovano za operaciju upisa i to za prvu operaciju upisa na tekućoj stranici. Ukoliko jeste, iz tabele stranica se dovlači deskriptor, zatim se u njemu postavlja na 1 indikator da je stranica modifikovana i na kraju modifikovani deskriptor stranice vraća u tabelu stranica. Pored toga u jedinici preslikavanja na 1 se postavlja i indikator da je stranica modifikovana. Na kraju zahteva se procesoru vraća fizička adresa i signal da je preslikavanje uspešno realizovano i prelazi na početni korak u kome se čeka pojava aktivne vrednosti signala startovanja sledećeg zahteva za presliikavanje. Ukoliko se ne radi o prvoj operaciji upisa na datoj stranici, odmah se procesoru vraća fizička adresa i signal da je preslikavanje uspešno realizovano i prelazi na početni korak.

Ukoliko ne postoji saglasnost prelazi se na dovlačenje niže reči nultog ulaza tabele stranica u kome se nalazi ukupan broj stranica procesa i poređenje ukupnog broja stranica procesa sa brojem stranice iz virtuelne adrese. Ukoliko je broj stranice virtuelne adrese veći od broja stranica procesa, radi se o zahtevu za preslikavanje nepostojeće stranice, pa se procesoru šalje signal da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice i prelazi na početni korak. U suprotnom slučaju se produžava sa preslikavanjem tako što se čita deskriptor stranice iz tabele stranica i proverava da li se stranica nalazi u operativnoj memoriji. Ukoliko se stranica ne nalazi u operativnoj memoriji, radi se o zahtevu za preslikavanje stranice koja nije u operativnoj memoriji, pa se procesoru šalje signal da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji i prelazi na početni korak.

U suprotnom slučaju se produžava sa preslikavanjem tako što se čita deskriptor stranice iz tabele stranica i dovlači u jedinicu za preslikavanje. Pored toga u jedinici preslikavanja se postavljaju i indikator da je ulaz važeći na 1 i indikator da je stranica modifikovana na vrednost indikatora da je stranica modifikovana iz deskriptora stranice. Potom se vraća na korak u kome se proverava da li postoji saglasnost. Pošto se sada utvrđuje da postoji saglasnost, jer se deskriptor nalazi u jedinici preslikavanja, produžava se sa već opisanim koracima za slučaj kada se deskriptor nalazi u jedinici preslikavanja.



Slika 30 Dijagram toka operacija

2.2.2.3.2.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture operacione jedinice (poglavlje 2.2.2.3.2.1) i dijagrama toka zahteva (poglavlje 2.2.2.3.2.2.1). Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je identična kao i notacija za algoritam generisanja upravljačkih signala za upravljačku jedinicu procesora ***CPU*** (poglavlje 2.2.2.1.2.2).

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *br* (*if* (**PRQ**) *then* $step_1$ *else* $step_0$)

! U koraku $step_0$ se čeka da se aktivom vrednošću signala **PRQ** procesora ***CPU*** iz registara procesora ***CPU*** u registre jedinice ***TLB*** upišu podaci neophodni za preslikavanje i startuje samo preslikavanje. Pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta virtuelna adresa, broj tekućeg procesa, početna adresa tabele stranica i tip operacije se upisuju u registre $UVA_{23...0}$, $UPR_{2...0}$ i $UPT_{19...0}$ i flip-flop URW bloka *cpu_interfejs*, respektivno, dok se samo broj stranice virtuelne adrese upisuje u registar $UPG_{13...0}$ bloka *mem_interfejs*. Pri aktivnoj vrednosti signala **PRQ** se prelazi na korak step1, dok se u suprotnom slučaju ostaje u koraku $step_0$. !

$step_1$: *if* (**HIT**, **ldUPA**),
if ($\overline{\textbf{HIT}}$, **ldMAR**, **incUPG**),
br (*if* **HIT** *then* step2 *else* step6)

! U korak $step_1$ može da se dođe ili iz koraka $step_0$ ili iz koraka $step_9$. Iz koraka $step_0$ se dolazi po startovanju svakog novog zahteva za preslikavanje. Iz koraka $step_9$ se dolazi po nekom zahtevu za preslikavanje za koji je, pri prethodnom prolasku kroz $step_1$, otkriveno da nema saglasnosti, pa se išlo na dovlačenje deskriptora stranice iz memorije ***MEM*** u jedinicu ***TLB***. U koraku $step_1$ se vrši provera saglasnosti i na osnovu toga formira vrednost signala **HIT** bloka *tag_data*. Ako se pojavi aktivna vrednost signala **HIT**, što znači da je otkrivena saglasnost, generiše se aktivna vrednost signala **ldUPA** bloka *cpu_interfejs* trajanja jedna perioda signala takta. Njome se fizička adresa, formirana od broja bloka pročitanog iz DATA memorije koji dolazi po linijama $\textbf{DATA}_{\textbf{9...0}}$ bloka *tag_data* i adrese reči u bloku virtuelne adrese koja dolazi po linijama $\textbf{UVA}_{\textbf{9...0}}$ bloka *cpu_interfejs*, upisuje u registar $UPA_{19...0}$ bloka *cpu_interfejs*. U slučaju da je neaktivna vrednost signal **HIT**, što znači da nije otkrivena saglasnost, mora da se prvo prođe kroz korake $step_6$, $step_7$, $step_8$ i $step_9$ i da se u njima iz tabele stranica u jedinicu za preslikavanje dovuče deskriptor tekuće stranice, pa da se zatim ponovo dođe u korak step1. Stoga se tada generiše aktivna vrednost signala **ldMAR** trajanja jedna perioda signala takta bloka *mem_interfejs* kojom se u registar $MAR_{19...0}$ upisuje sadržaj registra $UPT_{19...0}$ u kome se nalazi adresa nultog ulaza tabele stranica. Takođe se generiše i aktivna vrednost signala **incUPG** trajanja jedna perioda signala takta bloka *mem_interfejs* kojom se za jedan uvećava sadržaj registra $UPG_{13...0}$. Time se obezbeđuje da se u registru $UPG_{13...0}$ nalazi broj stranice virtuelne adrese plus 1. Ovo se radi zbog toga što se deskriptor *i*-te stranice nalazi u (*i*+1)-om ulazu tabele stranica. Pri aktivnoj vrednosti signala **HIT** se prelazi na korak $step_2$, dok se u suprotnom slučaju prelazi na korak $step_6$. !

$step_2$: *if* ($\overline{\textbf{CH}}$, **URP**),
if (**CH**, **mxMAR**, **ldMAR**),
br (*if* **CH** *then* $step_3$ *else* $step_0$)

! U korak $step_2$ može da se dođe jedino iz koraka $step_1$ i to samo onda kada je u koraku step1 otkrivena saglasnost i preslikavanje uspešno realizovano. U koraku $step_2$ se proverava vrednost signala **CH** bloka *tag_data*, koji može da ima aktivnu vrednosti samo ukoliko je preslikavanje realizovano za operaciju upisa i to za prvu operaciju upisa na tekućoj stranici. U slučaju da je signal **CH** bloka *tag_data* na neaktivnoj vrednosti, generiše se aktivna vrednosta signala **URP** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje uspešno realizovano i da se u registru $UPA_{19...0}$ bloka *cpu_interfejs* nalazi fizička adresa. U slučaju da je signal **CH** na aktivnoj vrednosti, mora da se prođe kroz korake $step_3$, $step_4$ i $step_5$ i da se u njima najpre u bit D deskriptora tekuće stranice u tabeli stranica upiše vrednost 1, pa da se tek onda generiše aktivna vrednosta signala **URP**. Stoga se u koraku $step_2$ generišu aktivne vrednosti signala **mxMAR** i **ldMAR** bloka *mem_interfejs* kojima se adresa ulaza u tabelu stranica formirana na izlazu sabirača ADD i propušta kroz multiplekser MP1 i upisuje u registar $MAR_{19...0}$. Pri aktivnoj vrednosti signala **CH** se prelazi na korak $step_3$, dok se u suprotnom slučaju prelazi na korak $step_0$. !

$step_3$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),

br (*if* **MEMFC** *then* step4 *else* step3)

! U korak $step_3$ se dolazi samo iz koraka $step_2$. U koraku $step_3$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* kojom se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači*, generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs*, čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost očitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_4$, dok se u suprotnom slučaju ostaje u koraku $step_3$. !

$step_4$: **mxMDR**, **ldMDR**, **stD**,
br $step_5$

! U korak $step_4$ se dolazi samo iz koraka $step_3$. U registru $MDR_{15...0}$ bloka *cpu_interfejs* se nalazi pročitan deskriptor stranice čiji bit D treba postaviti na 1. Stoga se bezuslovno generišu aktivne vednosti signala **mxMDR** i **ldMDR** bloka *mem_interfejs* kojima se 16-bitna reč, koja na pozicijama 15, 13,…0 ima razrede MDR_{15} i $MDR_{13...0}$ i na poziciji 14 ima 1, propušta kroz multiplekser MP2 i upisuje u registar $MDR_{15...0}$. Pored toga bezuslovno se generiše i aktivna vednost signala **stD** bloka *indikatori* kojom se postavlja na aktivnu vrednost jedan od flip-flopova V_0 do V_7 bloka *indikatori* koji odgovara ulazu jedinice ***TLB*** u kome je prilikom preslikavanja otkrivana saglasnost. Iz koraka $step_4$ se uvek prelazi na korak $step_5$. !

$step_5$: **writeMEM**, **incMEMACC**,
if (**MEMFC**, **URP**),
br (*if* **MEMFC** *then* $step_0$ *else* $step_5$)

! U korak $step_5$ se dolazi samo iz koraka $step_4$. U koraku $step_5$ se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *mem_interfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se modifikovani sadržaj registra $MDR_{15...0}$ bloka *mem_interfejs* upisuje u memoriju ***MEM*** na adresi određenoj sadržajem registra $MAR_{19...0}$ bloka *mem_interfejs*. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači* generiše se aktivna vrednost signala **URP** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje uspešno realizovano i da se u registru $UPA_{19...0}$ bloka *cpu_interfejs* nalazi fizička adresa. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_0$, dok se u suprotnom slučaju ostaje u koraku $step_5$. !

$step_6$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* $step_7$ *else* $step_6$)

! U korak $step_6$ se dolazi iz koraka $step_1$ kada se utvrdi da nema saglasnosti, pa se prelazi na korake dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu za preslikavanje. U koraku $step_6$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs* čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost pročitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč nultog ulaza tabele stranica tekućeg procesa. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_7$, dok se u suprotnom slučaju ostaje u koraku $step_6$. !

$step_7$: *if* (**GR**, **UAV**),
if ($\overline{\textbf{GR}}$, **mxMAR**, **ldMAR**),
br (*if* **GR** *then* $step_0$ *else* $step_8$)

! U korak $step_7$ se dolazi samo iz koraka $step_6$. U koraku $step_7$ se proverava da li se zahtev za preslikavanje odnosi na postojeću stranicu procesa. Signal **GR** bloka *mem_interfejs* predstavlja izlaz GTR komparatora CMP koji poredi broj stranice virtuelne adrese uvećan za jedan iz registra $UPG_{13...0}$ i broj stranica procesa iz registra $MDR_{13...0}$. Ako je signal **GR** na aktivnoj vrednosti, generiše se aktivna vrednost signala **UAV** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice. Ako je siganal **GR** na neaktivnoj vrednosti, produžava se sa koracima dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu jedinicu za preslikavanje tako što se generišu aktivne vrednosti signala **mxMAR** i **ldMAR** bloka *mem_interfejs* kojima se adresa ulaza u tabelu stranica formirana na izlazu sabirača ADD propušta kroz multiplekser MP1 i upisuje u registar $MAR_{19...0}$. Pri aktivnoj vrednosti signala **GR** se prelazi na koraka $step_0$, dok se u suprotnom slučaju prelazi na korak $step_8$. !

$step_8$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* $step_9$ *else* $step_8$)

! U korak $step_8$ se dolazi samo iz koraka $step_7$. U koraku $step_8$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje

vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs* čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost pročitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_9$, dok se u suprotnom slučaju ostaje u koraku $step_8$. !

$step_9$: *if* ($\overline{\textbf{valid}}$, **UPF**),
if (**valid**, **wrTAGDATA**, **stV**, **wrD**),
br (*if* **valid** *then* $step_1$ *else* $step_0$)

! U korak $step_9$ se dolazi samo iz koraka $step_8$. U koraku $step_9$ se proverava da li se zahtev za preslikavanje odnosi na stranicu procesa koja je u operativnoj memoriji. Signal **valid** bloka *mem_interfejs* predstavlja vrednost V bita deskriptora stranice koji vrednostima 0 i 1 određuje da se stranica ne nalazi i nalazi u operativnoj memoriji, respektivno. Ako je signal **valid** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **UPF** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje za stranicu koja nije u operativnoj memoriji. Ako je siganal **valid** na neaktivnoj vrednosti, produžava se sa koracima dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu za preslikavanje tako što se generišu aktivne vrednosti signala **wrTAGDATA**, **stV** i **wrD**. Aktivna vrednost signala **wrTAGDATA** bloka *tag_data* omogućuje da se u memoriju TAGDATA upišu broj procesa i 11 starijih bitova broj stranice sa linija $\textbf{TAGIN}_{\textbf{13...0}}$ i broj bloka sa linija $\textbf{MDR}_{\textbf{13...4}}$ bloka *mem_interfejs*. Adresa ulaza memorije TAGDATA je određena sa 3 najmlađa bita broja stranice $UVA_{2...0}$ bloka *cpu_interfejs*. Aktivnom vrednošću signala **stV** bloka *indikatori* se jedan od flip-flopova V_0 do V_7, adresiran sa 3 najmlađa bita broja stranice $UVA_{2...0}$, postavlja na 1, dok se aktivnom vrednošću signala **wrD** bloka *indikatori* jedan od flip-flopova D_0 do D_7, adresiran sa 3 najmlađa bita broja stranice $UVA_{2...0}$, postavlja na vrednost indikatora D dovučenog deskriptora stranice koji se nalazi u razredu MDR_{14} bloka *mem_interfejs*. Pri aktivnoj vrednosti signala **valid** se prelazi na korak $step_1$, dok se u suprotnom slučaju prelazi na korak $step_0$. !

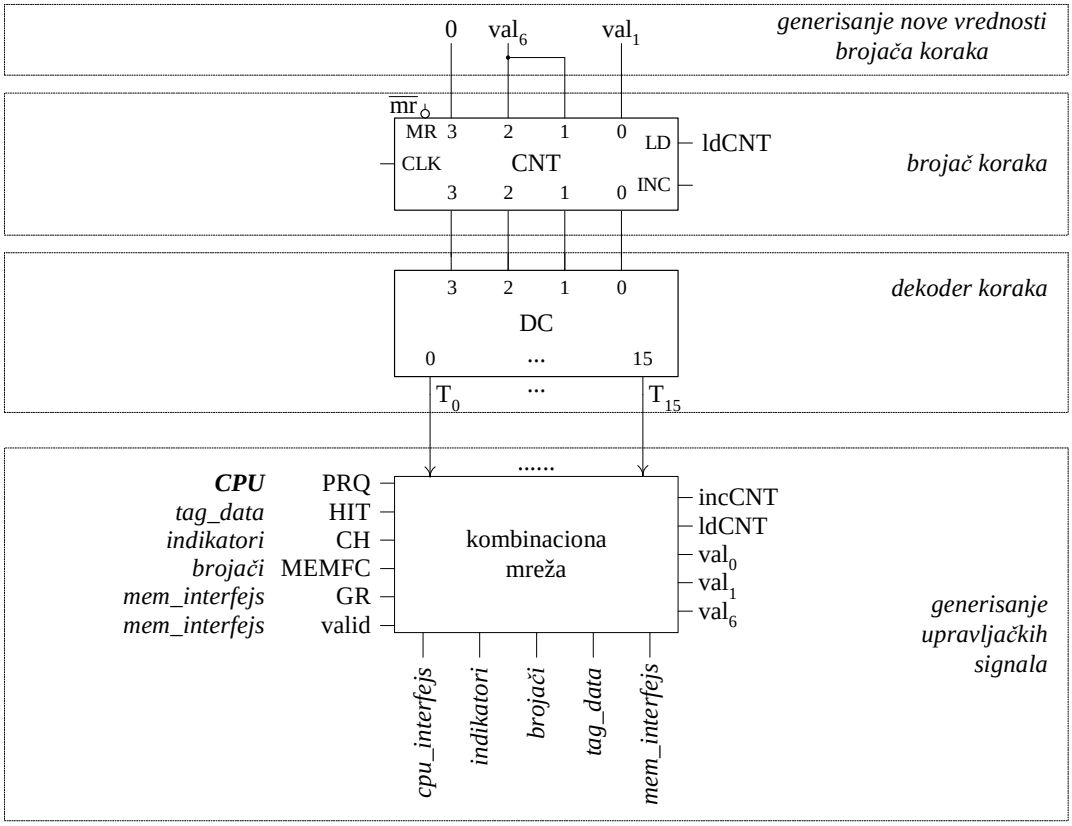
2.2.2.3.2.2.3. Struktura upravljačke jedinice

Upravljačka jedinica (slika 31) se sastoji od sledećih blokova:

- blok *generisanje nove vrednosti brojača koraka*,
- blok *brojač koraka*,
- blok *dekoder koraka* i
- blok *generisanje upravljačkih signala*.

Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.

Blok *generisanje nove vrednosti brojača koraka* služi za generisanje vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje 2.2.2.3.1.2.2) se utvrđuje da su 0, 1 i 6 vrednosti koje treba upisati u brojač CNT da bi se realizovala odstupanja od sekvencijalnog generisanja upravljačkih signala. Te vrednosti se formiraju na ulazima 3, 2, 1 i 0 brojača CNT pomoću signala $\textbf{val}_\textbf{0}$, $\textbf{val}_\textbf{1}$ i $\textbf{val}_\textbf{6}$ pri čemu signal $\textbf{val}_\textbf{0}$ ima vrednost 1 samo onda kada treba upisati 0, signal $\textbf{val}_\textbf{1}$ ima vrednost 1 samo onda kada treba upisati 1 i signal $\textbf{val}_\textbf{6}$ ima vrednost 6 samo onda kada treba upisati 6. Time se obezbeđuje da na ulazima 3, 2, 1 i 0 brojača CNT budu vrednosti 0, 0, 0 i 0 onda kada je signal $\textbf{val}_\textbf{0}$ ima vrednost 1, vrednosti 0, 0, 0 i 1 onda kada je signal $\textbf{val}_\textbf{1}$ ima vrednost 1 i vrednosti 0, 1, 1 i 0 onda kada je signal $\textbf{val}_\textbf{6}$ ima vrednost 1.



Slika 31 Struktura upravljačke jedinice

Blok *brojač koraka* sadrži brojač CNT. Brojač CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač CNT može da radi u sledećim režimima:

- režim inkrementiranja,
- režim skoka i
- režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača CNT za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **incCNT**. Signal **incCNT** je uvek neaktivan sem kada treba obezbediti režim inkrementiranja.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač CNT. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ldCNT**. Signal **ldCNT** je uvek neaktivan sem kada treba obezbediti režim skoka.

U režimu mirovanja pri pojavi signala takta ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **incCNT** i **ldCNT**. Ovi signali su neaktivni kada se čeka pojava aktivne vrednosti signala startovanja zahteva za preslikavanje **PRQ** procesora ***CPU*** ili pojava aktivne vrednosti signala **MEMFC** bloka *brojači* kojim se označava da je završena operacija čitanja iz memorije ***MEM*** ili upisa u memoriju ***MEM***

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali $\mathbf{T_0}$ do $\mathbf{T_{15}}$ na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.1.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$, itd.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$ do $\mathbf{T_{15}}$ koji dolaze sa bloka *dekoder koraka* upravljačke jedinice, signala logičkih uslova koji dolaze iz blokova operacione jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje 2.2.2.3.2.2.2) generišu upravljačke signale. Upravljački signali se generišu na identičan način kao i upravljački signali procesora ***CPU*** (poglavlje 2.2.2.1.2.2).

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:

- upravljačke signale operacione jedinice ***oper_jed*** i
- upravljačke signale upravljačke jedinice ***uprav_jed***.

Upravljački signali operacione jedinice ***oper_jed*** se daju posebno za svaki blok operacione jedinice i to:

- blok *cpu_interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag_data i*
- blok *cpu_ interfejs.*

Upravljački signali bloka *cpu_interfejs* se generišu na sledeći način:

- $\mathbf{ldUPA = T_1 \cdot HIT}$
- $\mathbf{URP = T_2 \cdot \overline{CH} + T_5 \cdot MEMFC}$
- $\mathbf{UAV = T_7 \cdot GR}$
- $\mathbf{UPF = T_9 \cdot \overline{valid}}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **HIT** — blok *tag_data*,
- **CH** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **GR** — blok *mem_interfejs* i
- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *indikatori* se generišu na sledeći način:

- $\mathbf{stD = T_4}$
- $\mathbf{stV = T_9 \cdot valid}$
- $\mathbf{wrD = T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *brojači* se generišu na sledeći način:

- $\mathbf{incMEMACC = T_3 + T_5 + T_6 + T_8}$

Pri njihovom generisanju se ne koriste signali logičkih uslova.

Upravljački signali bloka *tag_data* se generišu na sledeći način:

- $\mathbf{wrTAGDATA = T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *mem_interfejs* se generišu na sledeći način:

- **incUPG** = $\mathbf{T_1 \cdot \overline{HIT}}$
- **ldUPG** = $\mathbf{T_9 \cdot valid}$
- **mxMAR** = $\mathbf{T_2 \cdot CH + T_7 \cdot \overline{GR} + T_1 \cdot \overline{HIT}}$
- **ldMAR** = $\mathbf{T_2 \cdot CH + T_7 \cdot \overline{GR}}$
- **mxMDR** = $\mathbf{T_4}$
- **ldMDR** = $\mathbf{T_3 \cdot MEMFC + T_4 + T_6 \cdot MEMFC + T_8 \cdot MEMFC}$
- **writeMEM** = $\mathbf{T_5}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **HIT** — blok *tag_data*
- **CH** — blok *tag_data*.
- **GR** — blok *mem_interfejs* i
- **MEMFC** — blok *brojači*.

Upravljački signali upravljačke jedinice ***uprav_jed*** se generišu na sledeći način:

- **ldCNT** = $\mathbf{val_1 + val_0 + val_6}$
- **incCNT** = $\mathbf{T_0 \cdot PRQ + T_1 \cdot HIT + T_2 \cdot CH + T_3 \cdot MEMFC + T_4 + T_6 \cdot MEMFC + T_7 \cdot \overline{GR} + T_8 \cdot MEMFC}$
- $\mathbf{val_6}$ = $\mathbf{T_1 \cdot \overline{HIT}}$
- $\mathbf{val_0}$ = $\mathbf{T_2 \cdot \overline{CH} + T_5 \cdot MEMFC + T_7 \cdot GR + T_9 \cdot \overline{valid}}$
- $\mathbf{val_1}$ = $\mathbf{T_9 \cdot \overline{valid}}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

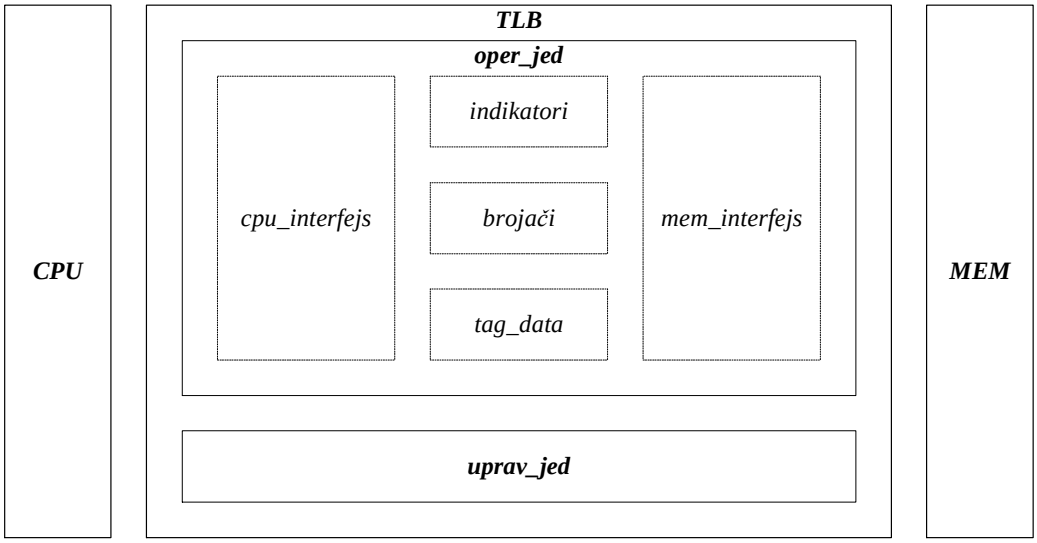
- **PRQ** — blok *cpu_interfejs*,
- **HIT** — blok *tag_data*,
- **CH** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **GR** — blok *mem_interfejs*.

2.2.2.3.3. JEDINICA SA SET ASOCIJATIVNIM PRESLIKAVANJEM

Jedinica sa set asocijativnim preslikavanjem ***TLB*** (slika 32) se sastoji iz:

- operacione jedinice ***oper_jed*** i
- upravljačke jedinice ***uprav_jed***.

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu set asocijativnog preslikavanja virtuelne adrese u fizičku adresu. Sve sekvencijalne mreže unutar jedinice ***TLB*** su sinhrone.



Slika 32 Jedinica sa set asocijativnim preslikavanjem

2.2.2.3.3.1. OPERACIONA JEDINICA

Operaciona jedinica ***oper_jed*** se sastoji iz sledećih blokova:

- blok *cpu_interfejs,*
- blok *indikatori,*
- blok *brojači,*
- blok *tag_data i*
- blok *cpu_ interfejs.*

Blok *cpu_interfejs* služi za razmenu podataka i upravljačkih signala sa procesorom ***CPU***.

Blok *indikatori* služi za vođenje evidencije za svaki od 8 ulaza jedinice, organizovanih u četiri seta sa po dva ulaza po setu, o tome da li je u ulazu važeći descriptor i da li je sadržaj stranice modifikovan.

Blok *brojači* služi za odbrojavanje onoliko perioda signala takta koliko je vreme pristupa memoriji ***MEM*** i realizaciju FIFO algoritma zamene za četiri seta sa po dva ulaza po setu.

Blok *tag_data* služi za čuvanje 8 deskriptora stranica organizovanih u četiri seta sa po dva ulaza po setu.

Blok *cpu_ interfejs* služi za razmenu podataka i upravljačkih signala sa memorijom ***MEM***.

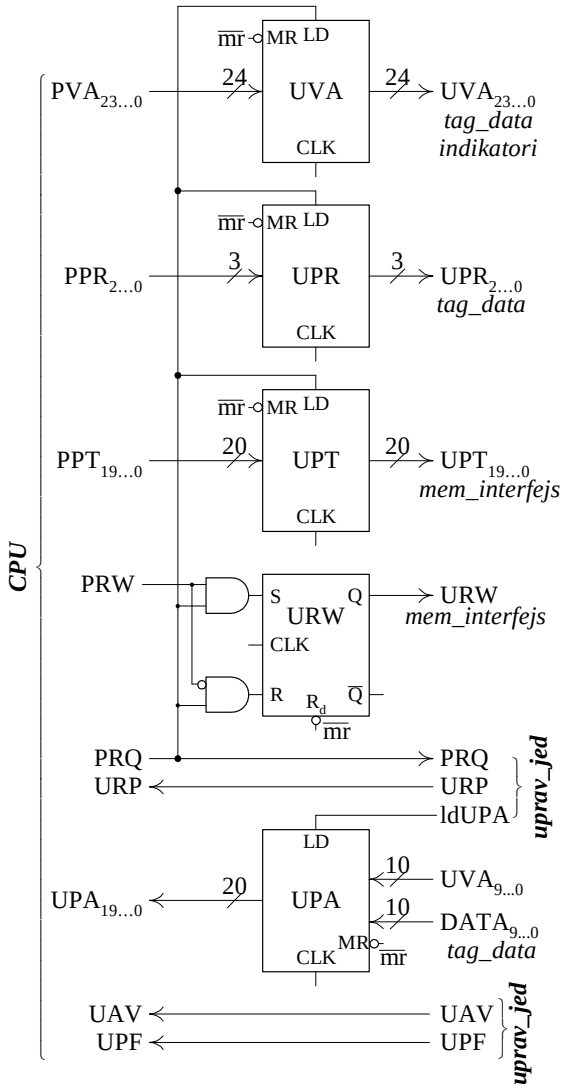
Struktura i opis blokova operacione jedinice ***oper_jed*** se daju u daljem tekstu.

2.2.2.3.3.1.1. Blok *cpu_interfejs*

Blok *cpu_interfejs* (slika 33) sadrži registre $UVA_{23...0}$, $UPR_{2...0}$, $UPT_{19...0}$ i $UPA_{19...0}$ i flip-flop URW.

Registar $UVA_{23...0}$ (*Unit Virtual Address register*) služi za čuvanje virtuelne adrese koju jedinica ***TLB*** treba da preslika u fizičku adresu. Virtuelna adresa dolazi na ulaze registra $UVA_{23...0}$ po linijama $\mathbf{PVA_{23...0}}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UVA_{23...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Razredi registra $UVA_{11...10}$ sadrže broj seta. Razredi registra $UVA_{11...10}$ se vode u blok *tag_data* gde se koriste za adresiranje memorije

TAGDATA0 ulaza nula i memorije TAGDATA1 ulaza jedan četiri seta. Pored toga razredi registra $UVA_{11...10}$ se vode i u blok *indikatori* gde se koriste za adresiranje flip-flopova $V0_0$ do $V0_3$ ulaza nula, flip-flopova $V1_0$ do $V1_3$ ulaza jedan, flip-flopova $D0_0$ do $D0_3$ ulaza nula i flip-flopova $D1_0$ do $D1_3$ ulaza jedan četiri seta. Razredi registra $UVA_{23...12}$ se takođe vode u blok *tag_data* gde se sa razredima registra $UPR_{2...0}$ koriste u dvema situacijama. Razredi registara $UPR_{2...0}$ i $UVA_{23...12}$ se koriste za utvrđivanje saglasnosti sa adresiranim ulazom ili memorije TAGDATA0 ulaza nula ili memorije TAGDATA1 ulaza jedan i čitanje broja bloka u koji se stranica preslikava. Pored toga razredi registra $UPR_{2...0}$ i $UVA_{23...12}$ se koriste i za upisivanje u adresirani ulaz ili memorije TAGDATA0 ulaza nula ili memorije TAGDATA1 ulaza jedan prilikom dovlačenja deskriptora stranice. Razredi registra $UVA_{9...0}$ se vode na ulaze 9 do 0 registra $UPA_{19...0}$ gde se koriste kao adresa u bloku prilikom formiranja fizičke adrese.



Slika 33 Blok *cpu_interfejs*

Registar $UPR_{2...0}$ (*Unit PRoces register*) služi za čuvanje broja tekućeg procesa. Broj tekućeg procesa dolazi na ulaze registra $UPR_{2...0}$ po linijama $\mathbf{PPR_{2...0}}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UPR_{2...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Razredi registra $UPR_{2...0}$ se vode u blok *tag_data* gde se sa razredima registra $UVA_{23...12}$ koriste za utvrđivanje saglasnosti sa adresiranim ulazom ili memorije TAGDATA0 ili memorije TAGDATA1 i

upisivanje ili u memoriju TAGDATA0 ili u memoriju TAGDATA1 prilikom dovlačenja deskriptora stranice.

Registar $UPT_{19...0}$ (*Unit Page map Table address register*) služi za čuvanje početne adrese tabele stranica tekućeg procesa. Adresa dolazi na ulaze registra $UPT_{19...0}$ po linijama $PPT_{19...0}$ iz bloka *tlb_interfejs* procesora ***CPU***, a u registar $UPT_{19...0}$ se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj registra $UPT_{19...0}$ se po vodi u blok *mem_interfejs* gde se koristi kao adresa za čitanje nultog ulaza tabele stranica radi provere da li se radi o pokušaju pristupa nepostojećoj stranici (*access violation*) i za formiranje adrese deskriptora stranice u tabeli stranica radi čitanja deskriptora ili upisa u deskriptor.

Registar $UPA_{19...0}$ (*Unit Physical Address register*) služi za čuvanje fizičke adrese koja se generiše u slučaju uspešnog preslikavanja virtuelne adrese u fizičku adresu. Viši bitovi fizičke adresa se generišu tako što se na ulaze 19 do 10 registra $UPA_{19...0}$ dovodi sadržaj $\mathbf{DATA_{9...0}}$ bloka *tag_data* koji predstavlja broj bloka u koji se stranica preslikava. Sadržaj $\mathbf{DATA_{9...0}}$ je dobijen ili iz memorije TAGDATA0 ili memorije TAGDATA1 u zavisnosti od toga u kojoj je od ove dve memorije za zadati set otkrivena saglasnost. Niži bitovi fizičke adresa se generišu tako što se na ulaze ulaze 9 do 0 registra $UPA_{19...0}$ dovodi sadržaj sa linija $\mathbf{UVA_{9...0}}$. Generisana fizička adresa se u registar $UPA_{19...0}$ upisuje pri aktivnoj vrednosti signala **ldUPA**. Sadržaj registra $UPA_{19...0}$ se vodi u blok *tlb_interfejs* procesora ***CPU***.

Flip-flop URW (*Unit Read Write flag*) vrednostima 0 i 1 predstavlja indikator operacije čitanja ili upisa, respektivno, koja treba da se realizuje po preslikavanju virtuelne adrese u fizičku adresu. Vrednost indikatora dolazi na ulaze flip-flopa URW po liniji PRW iz bloka *tlb_interfejs* procesora ***CPU***, a u flip-flop URW se upisuje pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta koji dolazi iz bloka *tlb_interfejs* procesora ***CPU***. Sadržaj flip-flopa URW se vodi u blok *indikatori*.

Upravljački signal **PRQ** (*Processor ReQuest*) koji dolazi iz bloka *tlb_inerfejs* procesora ***CPU*** se koristi da procesor ***CPU*** aktivnom vrednošću ovog signala trajanja jedna perioda izvrši upis sadržaja sa linija $\mathbf{PVA_{23...0}}$, $\mathbf{PPR_{2...0}}$, $\mathbf{PPT_{19...0}}$ i **PRW** bloka *tlb_interfejs* u registre $UVA_{23...0}$, $UPR_{2...0}$ i $UPT_{19...0}$ i flip_flop URW jedinice ***TLB*** i da se startuje preslikavanje.

Upravljački signal **URP** (*Unit RePly*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da je peslikavanje uspešno realizovano i da se na linijama $\mathbf{UPA_{19...0}}$ nalazi generisana fizička adresa.

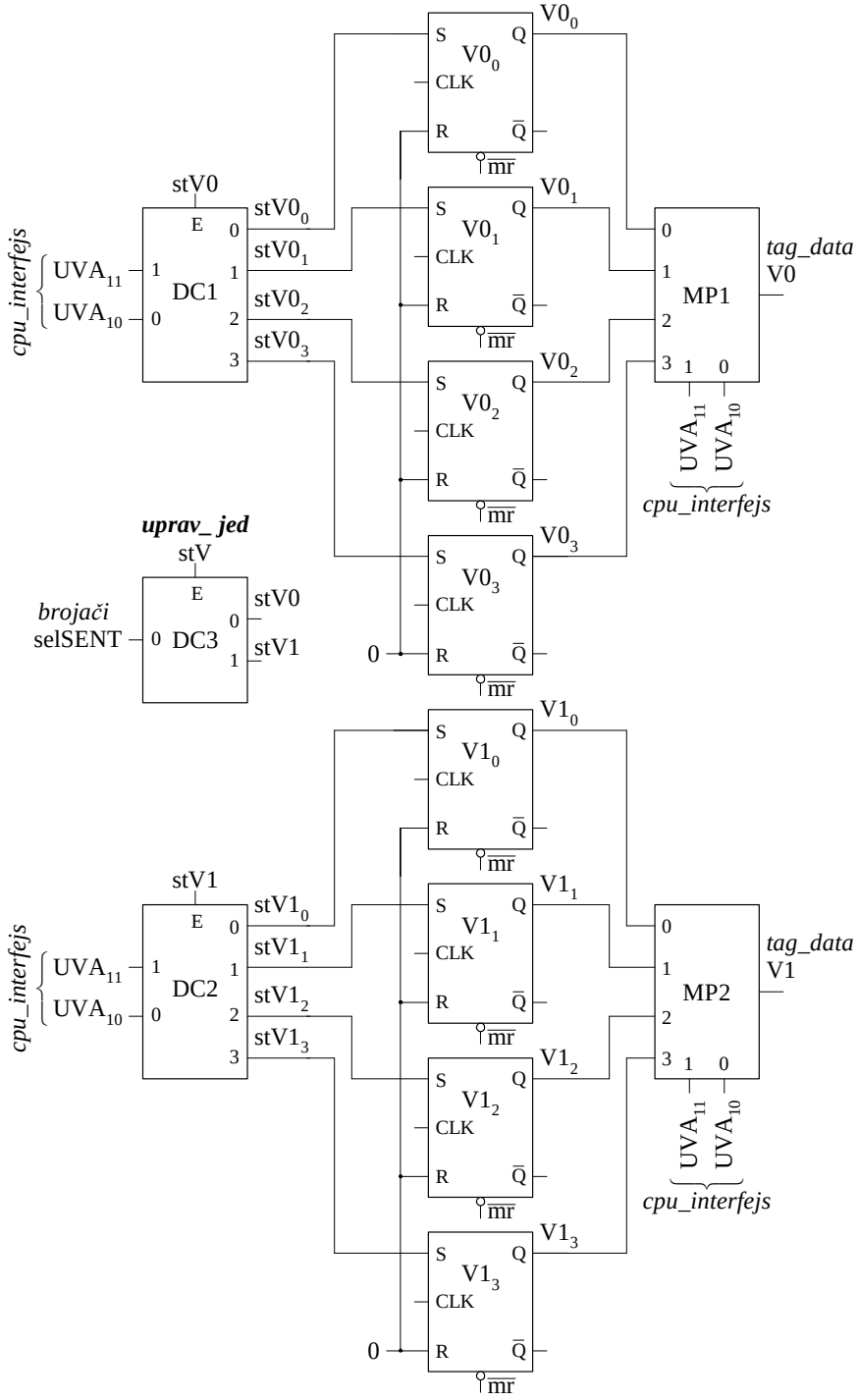
Upravljački signal **UAV** (*Unit Address Violation*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa nepostojećoj stranici (*access violation*).

Upravljački signal **UPF** (*Unit Page Fault*) koji se vodi u blok *tlb_inerfejs* procesora ***CPU*** se koristi da jedinica ***TLB*** aktivnom vrednošću ovog signala trajanja jedna perioda signala takta signalizira procesoru ***CPU*** da peslikavanje nije uspešno realizovano zbog pokušaja pristupa stranici koja nije u operativnoj memoriji (*page fault*).

2.2.2.3.3.1.2. Blok *indikatori*

Blok *indikatori* (slike 34 i 35) se sastoji od flip_flopova $V0_0$ do $V0_3$ sa dekoderom DC1 i multiplekserom MP1, flip_flopova $V1_0$ do $V1_3$ sa dekoderom DC2 i multiplekserom MP2,

dekodera DC3 (slika 34), flip_flopova $D0_0$ do $D0_3$ sa dekoderima DC1 i DC2 i multiplekserom MP1, flip_flopova $D1_0$ do $D1_3$ sa dekoderima DC3 i DC4 i multiplekserom MP2, dekodera DC5 i DC6 i multipleksera MP3 sa logičkim I kolom (slika 35).



Slika 34 Blok *indikatori* (prvi deo)

Flip-flopovi $V0_0$ do $V0_3$ (*Valid*) se koriste kao indikatori za svaki od sadržaja sa lokacija 0 do 3 memorije TAGDATA0 ulaza nula koje pripadaju setovima 0 do 3, respektivno, da li je važeći ili nije važeći. Flip-flop $V0_i$, i=0,…3, se postavlja na 1 aktivnom vrednošću signala

stV0$_i$ trajanja jedna perioda signala takta prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti lokaciju memorije TAGDATA0 ulaza nula koja pripada *i*-tom setu.

Dekoder DC1 se koristi da se, formiranjem aktivne vrednosti jednog od signala **stV0$_0$** do **stV0$_3$** na izlazima dekodera DC1, jedan od flip-flopova V0$_0$ do V0$_3$ postavi na 1. Ovo se realizuje prilikom dovlačenja deskriptora stranice generisanjem aktivne vrednosti signala **stV0** trajanja jedna perioda signala takta koji se pojavljuje kao aktivna vrednost na izlazu dekodera DC1 određenom sadržajem razreda **UVA$_{11...10}$** koji predstavlja broj seta i koji se dovodi na ulaze dekodera DC1 iz bloka *cpu_interfejs*.

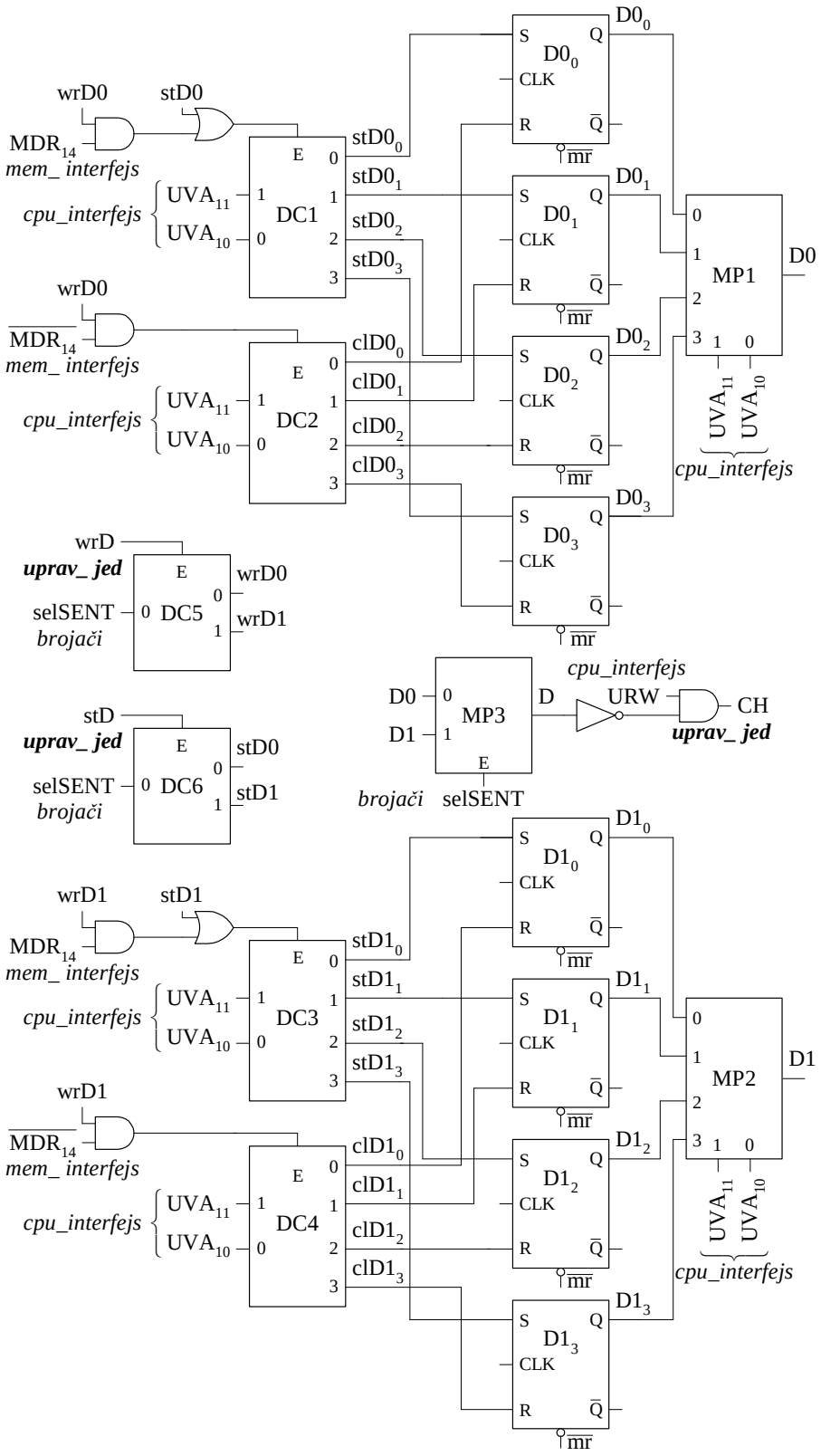
Multiplekser MP1 se koristi da se signalima **UVA$_{11...10}$** iz bloka *cpu_interfejs*, koji predstavljaju adresu *i*-te lokacije memorije TAGDATA0 ulaza nula koja pripada *i*-tom setu, *i*=0,…3, za koji se i proverava saglasnost, na izlazu multipleksera MP1 kao signal **V** selektuje vrednost flip-flopa V0$_i$. Signal **V0** se vodi u blok *tag_data* gde sa signalom **EQL0** sa izlaza komparatora CMP0 formira signal saglasnosti **HIT0** za memoriju TAGDATA0 ulaza nula.

Flip-flopovi V1$_0$ do V1$_3$ (*Valid*) se koriste kao indikatori za svaki od sadržaja sa lokacija 0 do 3 memorije TAGDATA1 ulaza jedan koje pripadaju setovima 0 do 3, respektivno, da li je važeći ili nije važeći. Flip-flop V0$_i$, *i*=0,…3, se postavlja na 1 aktivnom vrednošću signala **stV1$_i$** trajanja jedna perioda signala takta prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-ti lokaciju memorije TAGDATA1 ulaza nula koja pripada *i*-tom setu.

Dekoder DC2 se koristi da se, formiranjem aktivne vrednosti jednog od signala **stV1$_0$** do **stV1$_3$** na izlazima dekodera DC2, jedan od flip-flopova V1$_0$ do V1$_3$ postavi na 1. Ovo se realizuje prilikom dovlačenja deskriptora stranice generisanjem aktivne vrednosti signala **stV1** trajanja jedna perioda signala takta koji se pojavljuje kao aktivna vrednost na izlazu dekodera DC2 određenom sadržajem razreda **UVA$_{11...10}$** koji predstavlja broj seta i koji se dovodi na ulaze dekodera DC2 iz bloka *cpu_interfejs*.

Multiplekser MP2 se koristi da se signalima **UVA$_{11...10}$** iz bloka *cpu_interfejs*, koji predstavljaju adresu *i*-te lokacije memorije TAGDATA1 ulaza jedan koja pripada *i*-tom setu, *i*=0,…3, za koji se i proverava saglasnost, na izlazu multipleksera MP2 kao signal **V** selektuje vrednost flip-flopa V1$_i$. Signal **V1** se vodi u blok *tag_data* gde sa signalom **EQL1** sa izlaza komparatora CMP1 formira signal saglasnosti **HIT1** za memoriju TAGDATA1 ulaza jedan.

Dekoder DC3 se koristi da se pri aktivnoj vrednosti signala **stV**, a u zavisnosti od toga da li je vrednost signala **selSENT** bloka *brojači* nula ili jedan, formira aktivna vrednost ili signala **stV0** ili signala **stV1**, respektivno. Aktivna vrednost signala **stV** se generiše pri dovlačenju deskriptora stranice iz tabele stranica u jedinicu za preslikavanje da bi se na vrednost 1 postavio jedan od flip-flopova V0$_0$ do V0$_3$ i V1$_0$ do V1$_3$. Signal **selSENT** vrednostima 0 i 1 određuje da li deskriptor stranice treba upisati u memoriju TAGDATA0 ulaza nula i postaviti na vrednost 1 jedan od flip-flopova V0$_0$ do V0$_3$ ili u memoriju TAGDATA1 ulaza jedan i postaviti na vrednost 1 jedan od flip-flopova V1$_0$ do V1$_3$. Aktivna vrednost signala **stV0** daje, u zavisnosti od vrednosti signala broja seta UVA$_{11...10}$, aktivnu vrednost jednog od signala **stV0$_0$** do **stV0$_3$** čime se realizuje postavljanje na vrednost 1 jednog od flip-flopova V0$_0$ do V0$_3$, dok aktivna vrednost signala **stV1** daje, u zavisnosti od vrednosti signala broja seta **UVA$_{11...10}$**, aktivnu vrednost jednog od signala **stV1$_0$** do **stV1$_3$** čime se realizuje postavljanje na vrednost 1 jednog od flip-flopova V1$_0$ do V1$_3$.



Slika 35 Blok *indikatori* (drugi deo)

Flip-flopovi $D0_0$ do $D0_3$ (*Dirty*) se koriste kao indikatori za svaku od stranica čiji se deskriptori nalaze na lokacijama 0 do 3 memorije TAGDATA0 ulaza nula koje pripadaju

setovima 0 do 3, respektivno, da li je bilo upisa u stranicu ili ne. Flip-flop $D0_i$, i=0,…3, se postavlja na 1 aktivnom vrednošću signala **stD0*i*** trajanja jedna perioda signala i na 0 aktivnom vrednošću signala **clD0*i*** trajanja jedna perioda signala takta. Ovi signali se koriste da se prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-tu lokaciju memorije TAGDATA0 ulaza nula koja pripada *i*-tom setu flip-flop $D0_i$ postavi na vrednost bita D (*Dirty*) deskriptora stranice, koji se nalazi u razredu MDR_{14} registra $MDR_{15...0}$ bloka *mem_interfejs*, i da se prilikom generisanja fizičke adrese radi upisa, flip-flop $D0_i$ postavi na 1.

Dekoder DC1 se koristi da se, formiranjem aktivne vrednosti jednog od signala **stD0$_0$** do **stD0$_3$** na izlazima dekodera DC1 trajanja jedna perioda signala takta, jedan od flip-flopova $D0_0$ do $D0_3$ postavi na 1. Dekoder DC2 se koristi da se, formiranjem aktivne vrednosti jednog od signala **clD0$_0$** do **clD0$_3$** na izlazima dekodera DC2 trajanja jedna perioda signala takta, jedan od flip-flopova $D0_0$ do $D0_3$ postavi na 0. Prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-tu lokaciju memorije TAGDATA0 ulaza nula koja pripada *i*-tom setu, treba da se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, generiše aktivna vrednost signala **stD0*i*** ili **clD0*i***, respektivno, i da se flip-flop $D0_i$ postavi na vrednost bita D (*Dirty*) deskriptora stranice. Tada se generiše signal **wrD0** koji se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, pojavljuje kao aktivna vrednost ili signala **stD0*i*** na izlazu dekodera DC1 ili signala **clD0*i*** na izlazu dekodera DC2 određenom signalima **UVA$_{11...10}$** bloka *cpu_interfejs* koji se dovode na ulaze dekodera DC1 i DC2. Prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-toj lokaciji memorije TAGDATA0 ulaza nula koja pripada *i*-tom setu, i=0,…3, treba da se generiše aktivna vrednost signala **stD0*i*** i da se flip-flop $D0_i$ postavi na 1. Tada se generiše signal **stD0** koji se pojavljuje kao aktivna vrednost signala **stD0*i*** na izlazu dekodera DC1 određenom signalima **UVA$_{11...10}$** bloka *cpu_interfejs* koji se dovode na ulaze dekodera DC1.

Multiplekser MP1 se koristi da se signalima signalima **UVA$_{11...10}$** bloka *cpu_interfejs*, koji predstavljaju broj *i*-te lokacije memorije TAGDATA0 ulaza nula koja pripada *i*-tom setu, i=0,…3, u kome se proverava saglasnost, na izlazu multipleksera MP1 kao signal **D0** selektuje indikator $D0_i$.

Flip-flopovi $D1_0$ do $D1_3$ (*Dirty*) se koriste kao indikatori za svaku od stranica čiji se deskriptori nalaze na lokacijama 0 do 3 memorije TAGDATA1 ulaza jedan koje pripadaju setovima 0 do 3, respektivno, da li je bilo upisa u stranicu ili ne. Flip-flop $D1_i$, i=0,…3, se postavlja na 1 aktivnom vrednošću signala **stD1*i*** trajanja jedna perioda signala i na 0 aktivnom vrednošću signala **clD1*i*** trajanja jedna perioda signala takta. Ovi signali se koriste da se prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-tu lokaciju memorije TAGDATA1 ulaza jedan koja pripada *i*-tom setu flip-flop $D1_i$ postavi na vrednost bita D (*Dirty*) deskriptora stranice, koji se nalazi u razredu MDR_{14} registra $MDR_{15...0}$ bloka *mem_interfejs*, i da se prilikom generisanja fizičke adrese radi upisa, flip-flop $D1_i$ postavi na 1.

Dekoder DC3 se koristi da se, formiranjem aktivne vrednosti jednog od signala **stD1$_0$** do **stD1$_3$** na izlazima dekodera DC3 trajanja jedna perioda signala takta, jedan od flip-flopova $D1_0$ do $D1_3$ postavi na 1. Dekoder DC4 se koristi da se, formiranjem aktivne vrednosti jednog od signala **clD1$_0$** do **clD1$_3$** na izlazima dekodera DC4 trajanja jedna perioda signala takta, jedan od flip-flopova $D1_0$ do $D1_3$ postavi na 0. Prilikom dovlačenja deskriptora stranice iz memorije ***MEM*** u *i*-tu lokaciju memorije TAGDATA1 ulaza jedan koja pripada *i*-tom setu, treba da se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, generiše aktivna vrednost signala **stD1*i*** ili **clD1*i***, respektivno, i da se flip-flop $D1_i$ postavi na vrednost bita D (*Dirty*) deskriptora

stranice. Tada se generiše signal **wrD1** koji se, u zavisnosti od toga da li je bit MDR_{14} 0 ili 1, pojavljuje kao aktivna vrednost ili signala **stD1*i*** na izlazu dekodera DC3 ili signala **clD1*i*** na izlazu dekodera DC4 određenom signalima **UVA11...10** bloka *cpu_interfejs* koji se dovode na ulaze dekodera DC3 i DC4. Prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-toj lokaciji memorije TAGDATA1 ulaza jedan koja pripada *i*-tom setu, *i*=0,…3, treba da se generiše aktivna vrednost signala **stD1*i*** i da se flip-flop $D1_i$ postavi na 1. Tada se generiše signal **stD1** koji se pojavljuje kao aktivna vrednost signala **stD1*i*** na izlazu dekodera DC3 određenom signalima **UVA11...10** bloka *cpu_interfejs* koji se dovode na ulaze dekodera DC3.

Multiplekser MP2 se koristi da se signalima signalima **UVA11...10** bloka *cpu_interfejs*, koji predstavljaju broj *i*-te lokacije memorije TAGDATA1 ulaza jedan koja pripada *i*-tom setu, *i*=0,…3, u kome se proverava saglasnost, na izlazu multipleksera MP2 kao signal **D1** selektuje indikator $D1_i$.

Dekoder DC5 se koristi da se pri aktivnoj vrednosti signala **wrD**, a u zavisnosti od toga da li je vrednost signala **selSENT** bloka *brojači* nula ili jedan, formira aktivna vrednost ili signala **wrD0** ili signala **wrD1**, respektivno. Aktivna vrednost signala **wrD** se generiše pri dovlačenju deskriptora stranice iz tabele stranica u jedinicu za preslikavanje da bi se jedan od flip-flopova $D0_0$ do $D0_3$ i $D1_0$ do $D1_3$ postavio na vrednost signala MDR_{14} koji predstavlja bit D (*Dirty*) deskriptora stranice. Signal **selSENT** vrednostima 0 i 1 određuje da li deskriptor stranice treba upisati u memoriju TAGDATA0 ulaza nula i postaviti na vrednost bita MDR_{14} jedan od flip-flopova $D0_0$ do $D0_3$ ili u memoriju TAGDATA1 ulaza jedan i postaviti na vrednost bita MDR_{14} jedan od flip-flopova $D1_0$ do $D1_3$. Aktivna vrednost signala **wrD0** daje saglasno vrednostima signala broja seta $UVA_{11...10}$ ili aktivnu vrednost jednog od signala **stD0$_0$** do **stD0$_3$**, ukoliko je signal MDR_{14} jedan, ili aktivnu vrednost jednog od signala **clD0$_0$** do **clD0$_3$**, ukoliko je signal MDR_{14} nula, čime se realizuje postavljanje na vrednost signala MDR_{14} jednog od flip-flopova $D0_0$ do $D0_3$. Na isti način aktivna vrednost signala **wrD1** daje saglasno vrednostima signala broja seta $UVA_{11...10}$ ili aktivnu vrednost jednog od signala **stD1$_0$** do **stD1$_3$**, ukoliko je signal MDR_{14} jedan, ili aktivnu vrednost jednog od signala **clD1$_0$** do **clD1$_3$**, ukoliko je signal MDR_{14} nula, čime se realizuje postavljanje na vrednost signala MDR_{14} jednog od flip-flopova $D1_0$ do $D1_3$.

Dekoder DC6 se koristi da se pri aktivnoj vrednosti signala **stD**, a u zavisnosti od toga da li je vrednost signala **selSENT** bloka *brojači* nula ili jedan, formira aktivna vrednost ili signala **stD0** ili signala **stD1**, respektivno. Aktivna vrednost signala **stD** se generiše prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u *i*-toj lokaciji ili memorije TAGDATA0 ulaza nula ili memorije TAGDATA1 ulaza jedan koja pripada *i*-tom setu, *i*=0,…3, da bi se na vrednost 1 postavio jedan od flip-flopova $D0_0$ do $D0_3$ i $D1_0$ do $D1_3$. Signal **selSENT** vrednostima 0 i 1 određuje da li se upis realizuje u stranicu čiji se deskriptor nalazi u memoriju TAGDATA0 ulaza nula, pa na vrednost 1 treba postaviti jedan od flip-flopova $D0_0$ do $D0_3$, ili u stranicu čiji se deskriptor nalazi u memoriju TAGDATA1 ulaza jedan, pa na vrednost 1 treba postaviti jedan od flip-flopova $D1_0$ do $D1_3$. Aktivna vrednost signala **stD0** daje, u zavisnosti od vrednosti signala broja seta $UVA_{11...10}$, aktivnu vrednost jednog od signala **stD0$_0$** do **stD0$_3$** čime se realizuje postavljanje na vrednost 1 jednog od flip-flopova $D0_0$ do $D0_3$, dok aktivna vrednost signala **stD1** daje, u zavisnosti od vrednosti signala broja seta **UVA11...10**, aktivnu vrednost jednog od signala **stD1$_0$** do **stD1$_3$** čime se realizuje postavljanje na vrednost 1 jednog od flip-flopova $D1_0$ do $D1_3$.

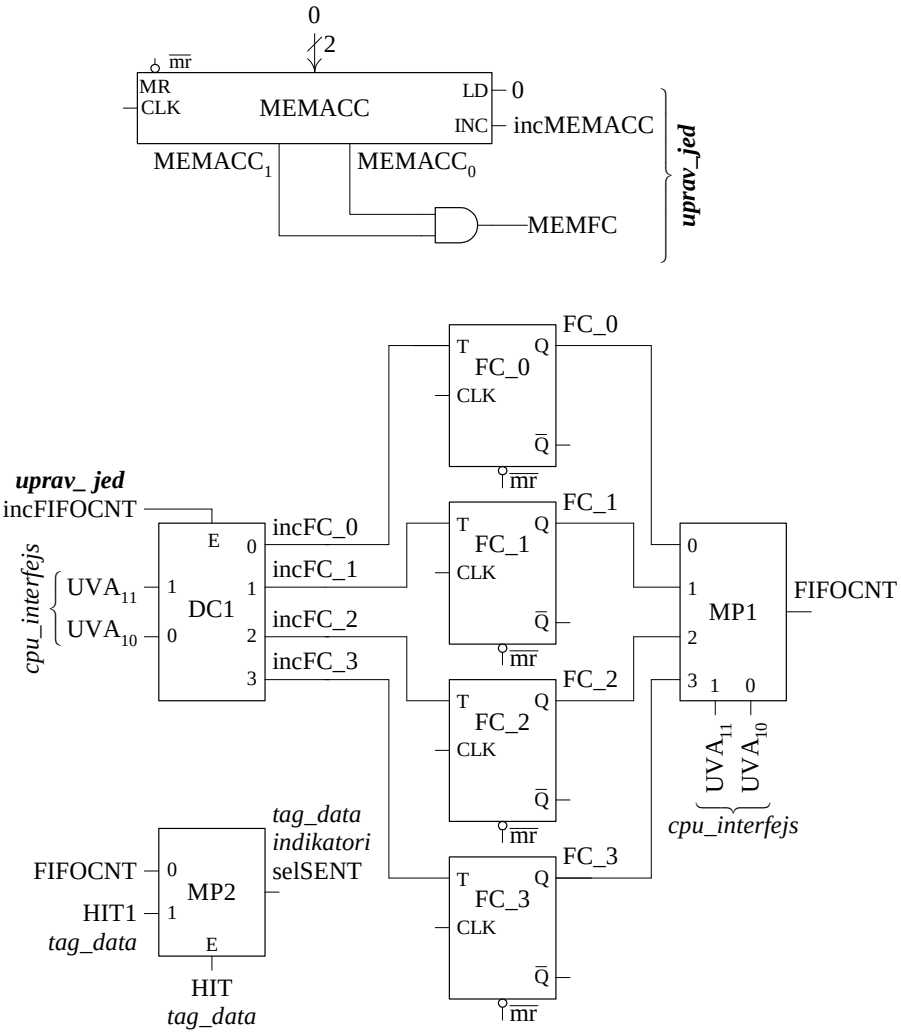
Multiplekser MP3 se koristi da se vrednostima 0 i 1 signala **selSENT** bloka *brojači* na izlazu multipleksera MP3 kao signal **D** selektuje jedan od signala **D0** ili **D1**, respektivno.

Signal **selSENT** vrednostima 0 i 1 određuje da li se deskriptor stranice za koju je otkrivena saglasnost i u koju treba upisati nalazi u memoriju TAGDATA0 ulaza nula, pa kao signal **D** treba koristiti signal **D0**, ili u memoriju TAGDATA1, pa kao signal **D** treba koristiti signal **D1**. Signal **D** sa signalom **URW** bloka *cpu_interfejs* formira signal **CH** (*change*), pri čemu signal **URW** vrednostima 0 i 1 ukazuje da li sa preslikane fizičke adrese treba realizovati čitanje ili upis, respektivno. Signal **CH** ima vrednost 1 ukoliko signal **URW** ima vrednost 1, jer se radi o upisu, i ukoliko signal **D** ima vrednost 0, jer nije bilo upisa u stranicu za koju je otkrivena saglasnost i čiji se deskriptor nalazi u jedinice ***TLB***. Tada treba bit D deskriptora date stranice u tabeli stranica postaviti na 1. U svim ostalim slučajevima signal **CH** ima vrednost 0 i tada ne treba ništa raditi.

Signal **CH** se koristi prilikom generisanja fizičke adrese radi upisa u stranicu čiji se deskriptor nalazi u jedinici ***TLB***. Ukoliko je bit D 0, to znači da nije bilo upisa u datu stranicu i da treba bit D deskriptora date stranice u tabeli stranica postaviti na 1. Kako je za operaciju upisa indikator URW bloka *cpu_interfejs* 1, to je i signal **CH** 1. Ukoliko je bit D 1, to znači da je bilo upisa u datu stranicu i da je bit D deskriptora date stranice u tabeli stranica već postavljen 1. Kako je za operaciju upisa indikator URW bloka *cpu_interfejs* 1, to je i signal **CH** 0.

2.2.2.3.3.1.3. Blok *brojači*

Blok *brojači* (slika 36) se sastoji od $MEMACC_{1...0}$ (*MEMory ACCess time*), flip-flopova FC_0, FC_1, FC_2 i FC_3, dekodera DC1 i multipleksera MP1 i MP2.



Slika 36 Blok *brojači*

Brojač $MEMACC_{1...0}$ se koristi da kao brojač vremena pristupa operativnoj memoriji odbroji četiri signala takta kod obraćanja jedinice ***TLB*** memoriji ***MEM*** radi čitanja ili upisa, jer je usvojeno da je vreme pristupa memoriji ***MEM*** četiri periode signala takta. Inkrementiranje brojača se realizuje pri aktivnoj vrednosti signala **incMEMACC**. Signal **MEMFC** (*MEMory Function Completed*) postaje aktivan na treći signal takta kada brojač $MEMACC_{1...0}$ inkrementiranjem pređe u stanje tri. Na četvrti signal takta brojač $MEMACC_{1...0}$ se vraća na stanje nula, a signal **MEMFC** postaje neaktivan. Signal **MEMFC** služi upravljačkoj jedinici ***uprav_jed*** kao indikacija da je pristup memoriji ***MEM*** završen.

Flip-flopovi FC_0, FC_1, FC_2 i FC_3, dekoder DC1 i multiplekseri MP1 i MP2 se koriste za realizaciju FIFO algoritma zamene za setove 0 do 3 koji imaju ulaze 0 i 1. Flip-flopovi FC_0, FC_1, FC_2 i FC_3 se koriste kao FIFO brojači po modulu 2 za setove 0 do 3, respektivno. U slučaju da u *i*-tom setu, *i*=0..3, nema saglasnosti, u zavisnosti od toga da li je vrednost flip-flopa FC_*i* 0 ili 1 deskriptor stranice se dovlači u *i*-tu lokaciju memorije TAGDATA0 ulaza nula ili memorije TAGDATA1 ulaza jedan, respektivno, i sadržaj flip-flopa FC_*i* inkrementira sa 0 na 1 i sa 1 na 0, respektivno. Inkrementiranje flip-flopa FC_*i*, *i*=0,…3, se realizuje aktivnom vrednošću signala **incFC_*i***, trajanja jedna perioda signala takta.

Dekoder DC1 se koristi da se, formiranjem aktivne vrednosti jednog od signala **incFC_0** do **incFC_3** na izlazima dekodera DC1, jedan od flip-flopova FC_0, FC_1, FC_2 i FC_3 inkrementira. Ovo se realizuje prilikom dovlačenja deskriptora stranice generisanjem aktivne vrednosti signala **incFIFOCNT** trajanja jedna perioda signala takta koji se pojavljuje kao aktivna vrednost na izlazu dekodera DC1 određenom sadržajem razreda $\mathbf{UVA_{11...10}}$ koji predstavlja broj seta i koji se dovodi na ulaze dekodera DC1 iz bloka *cpu_interfejs*.

Multiplekser MP1 se koristi da se signalima $\mathbf{UVA_{11...10}}$ iz bloka *cpu_interfejs*, koji predstavljaju broj *i*-tog seta, *i*=0,…3, u koji se dovlači deskriptor stranice, na izlazu multipleksera MP1 kao signal **FIFOCNT** selektuje vrednost flip-flopa FC_*i*.

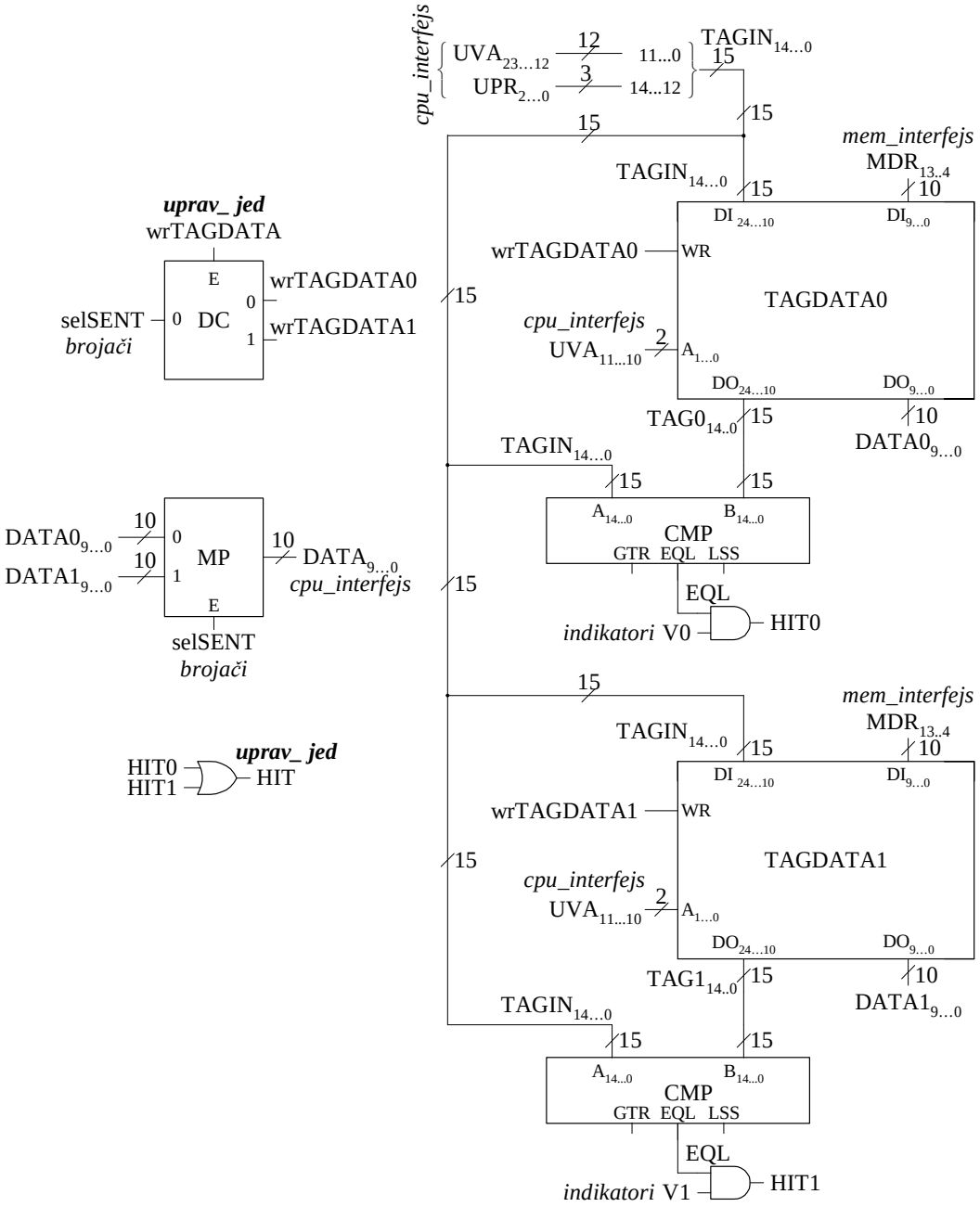
Multiplekser MP2 se koristi da se vrednostima 0 i 1 signala **HIT** bloka *tag_data* kao signal **selSENT** selektuje jedan od signala **FIFOCNT** ili signal **HIT1**, respektivno. U slučaju da nema saglasnosti signal **HIT** je nula, pa se tada kao vrednost signala **selSENT** pojavljuje vrednost signala **FIFOCNT**. Stoga kada nema saglasnosti, vrednostima 0 i 1 signala **FIFOCNT** se određuje da li treba pristupati memoriji TAGDATA0, flip-flopovima $V0_0$ do $V0_3$ i $D0_0$ do $D0_3$ ulaza nula ili memoriji TAGDATA1, flip-flopovima $V1_0$ do $V1_3$ i $D1_0$ do $D1_3$ ulaza jedan, respektivno. U slučaju da ima saglasnosti signal **HIT** je jedan, pa se tada kao vrednost signala **selSENT** pojavljuje vrednost signala **HIT1**. Stoga kada ima saglasnosti vrednostima 0 i 1 signala **HIT1** se određuje da li treba pristupati memoriji TAGDATA0, flip-flopovima $V0_0$ do $V0_3$ i $D0_0$ do $D0_3$ ulaza nula ili memoriji TAGDATA1, flip-flopovima $V1_0$ do $V1_3$ i $D1_0$ do $D1_3$ ulaza jedan, respektivno.

2.2.2.3.3.1.4. Blok *tag_data*

Blok *tag_data* (slika 37) sadrži RAM memorije TAGDATA0 i TAGDATA1, komparatore CMP0 i CMP1, logička I i ILI kola, dekoder DC i koder CD.

Memorija TAGDATA0 ima četiri memorijske lokacije širine 25 bitova koje predstavljaju ulaze nula setova 0 do 3. U *i*-toj lokaciji memorije TAGDATA0 koja pripada *i*-tom setu, *i*=0,…3, se za stranicu koja se nalazi u *i*-tom setu čuva broj procesa, broj stranice čiji se descriptor trenutno nalazi u datoj lokaciji memorije TAGDATA0 i deskriptor stranice koji predstavlja broj bloka u koji je preslikana stranica. Memorija TAGDATA0 ima sledeće ulazne i izlazne linije:

- adresne linije $\mathbf{A_{1...0}}$,
- ulazne linije podataka $\mathbf{DI_{24...0}}$,
- izlazne linije podataka $\mathbf{DO_{24...0}}$ i
- upravljačku liniju **WR**.



Slika 37 Blok *tag_data*

Na adresne linije **A1...0** se dovode signali **UVA11...10** bloka *cpu_interfejs* koji predstavljaju 2 najmlađa bit broja stranice. Ulazne linije podataka su grupisane u linije **DI24...10** i linije **DI9...0**. Na ulazne linije podataka **DI24...10** se vode signali **TAGIN14...0**, gde signali **TAGIN14...12** predstavljaju signale broja procesa **UPR2...0** bloka *cpu_interfejs* i signali **TAGIN11...0** predstavljaju signale 12 najstarijih bitova broja stranice **UVA23...12** bloka *cpu_interfejs*. Na ulazne linije podataka **DI9...0** se vode signali **MDR13...4** pročitanog deskriptora stranice iz registra **MDR15...0** bloka *mem_interfejs* koji predstavljaju broj bloka u koji se stranica preslikava. Izlazne linije podataka su grupisane u linije **DO24...10** i linije **DO9...0**. Na izlaznim linijama podataka **DO24...10** se pojavljuju signali **TAG014...0**, gde signali **TAG014...12** predstavljaju signale broja procesa i signali **TAG011...0** predstavljaju signale 12

najstarijih bitova broja stranice adresiranog ulaza. Na izlaznim linijama podataka $\mathbf{DO}_{9...0}$ se pojavljuju signali $\mathbf{DATA0}_{9...0}$ koja predstavlja broj bloka u koji se preslikava stranica. Upis u memoriju TAGDATA0 se realizuje aktivnom vrednošću signala **wrTAGDATA0**, a čitanje njegovom neaktivnom vrednošću. Memoriji TAGDATA0 se pristupa u sledećim slučajevima:

1. *Utvrđivanje saglasnosti i formiranja fizičke adrese pri utvrđenoj saglasnosti.* Iz memorije TAGDATA0 se čita. Pročitani sadržaj sa linija $\mathbf{TAG0}_{14...0}$ se vodi na komparator CMP0 gde se upoređuje sa sadržajem sa linija $\mathbf{TAGIN}_{14...0}$. Ukoliko saglasnost postoji, pročitani sadržaj na linijama $\mathbf{DATA0}_{9...0}$ predstavlja broj bloka u koji se preslikava stranica, pa se sadržaj sa linija $\mathbf{DATA0}_{9...0}$ propušta kroz multiplekser MP i kao sadržaj $\mathbf{DATA}_{9...0}$ vodi u blok *cpu_interfejs* na ulaze 19…10 registra $\mathrm{URA}_{\mathbf{19...10}}$ radi formiranja fizičke adrese.

2. *Upisivanje broja procesa, broja stranice i broja bloka kod dovlačenja deskriptora stranice.* U memoriju TAGDATA0 se upisuje. Sadržaj koji se upisuje je $\mathbf{TAGIN}_{14...0}$ i $\mathbf{MDR}_{13...4}$, gde signali $\mathbf{TAGIN}_{14...12}$ predstavljaju signale broja procesa, signali $\mathbf{TAGIN}_{11...0}$ predstavljaju signale 12 najstarijih bitova broja stranice i signali $\mathbf{MDR}_{13...4}$ predstavljaju signale broja bloka u koji se stranica preslikava.

Komparator CMP0 poredi sadržaj na linijama $\mathbf{TAGIN}_{14...0}$, gde signali $\mathbf{TAGIN}_{14...12}$ predstavljaju signale broja procesa i signali $\mathbf{TAGIN}_{11...0}$ predstavljaju signale 12 najstarijih bitova broja stranice zahteva za preslikavanje, i sadržaj na linijama $\mathbf{TAG0}_{14...0}$, gde signali $\mathbf{TAG0}_{14...12}$ predstavljaju signale broja procesa i signali $\mathbf{TAG0}_{11...0}$ predstavljaju signale 12 najstarijih bitova broja stranice adresirane lokacije memorije TAGDATA0 ulaza nula. Ukoliko se sadržaji slažu, signal **EQL0** na izlazu komparatora CMP0 ima vrednost 1. Ukoliko je i signal **V0** bloka *indikatori* 1, što ukazuje da je sadržaj adresirane lokacije memorije TAGDATA0 ulaza nula važeća, na izlazu I kola se formira vrednost 1 signal **HIT0**, čime se utvrđuje da postoji saglasnost u memoriji TAGDATA0 ulaza nula.

Memorija TAGDATA1 ima četiri memorijske lokacije širine 25 bitova koje predstavljaju ulaze jedan setova 0 do 3. U *i*-toj lokaciji memorije TAGDATA1 koja pripada *i*-tom setu, *i*=0,…3, se za stranicu koja se nalazi u *i*-tom setu čuva broj procesa, broj stranice čiji se deskriptor trenutno nalazi u datoj lokaciji memorije TAGDATA1 i deskriptor stranice koji predstavlja broj bloka u koji je preslikana stranica. Memorija TAGDATA1 ima sledeće ulazne i izlazne linije:

- adresne linije $\mathbf{A}_{1...0}$,
- ulazne linije podataka $\mathbf{DI}_{24...0}$,
- izlazne linije podataka $\mathbf{DO}_{24...0}$ i
- upravljačku liniju **WR**.

Na adresne linije $\mathbf{A}_{1...0}$ se dovode signali $\mathbf{UVA}_{11...10}$ bloka *cpu_interfejs* koji predstavljaju 2 najmlađa bit broja stranice. Ulazne linije podataka su grupisane u linije $\mathbf{DI}_{24...10}$ i linije $\mathbf{DI}_{9...0}$. Na ulazne linije podataka $\mathbf{DI}_{24...10}$ se vode signali $\mathbf{TAGIN}_{14...0}$, gde signali $\mathbf{TAGIN}_{14...12}$ predstavljaju signale broja procesa $\mathbf{UPR}_{2...0}$ bloka *cpu_interfejs* i signali $\mathbf{TAGIN}_{11...0}$ predstavljaju signale 12 najstarijih bitova broja stranice $\mathbf{UVA}_{23...12}$ bloka *cpu_interfejs*. Na ulazne linije podataka $\mathbf{DI}_{9...0}$ se vode signali $\mathbf{MDR}_{13...4}$ pročitanog deskriptora stranice iz registra $\mathbf{MDR}_{15...0}$ bloka *mem_interfejs* koji predstavljaju broj bloka u koji se stranica preslikava. Izlazne linije podataka su grupisane u linije $\mathbf{DO}_{24...10}$ i linije $\mathbf{DO}_{9...0}$. Na izlaznim linijama podataka $\mathbf{DO}_{24...10}$ se pojavljuju signali $\mathbf{TAG1}_{14...0}$, gde signali $\mathbf{TAG1}_{14...12}$ predstavljaju signale broja procesa i signali $\mathbf{TAG1}_{11...0}$ predstavljaju signale 12 najstarijih bitova broja stranice adresiranog ulaza. Na izlaznim linijama podataka $\mathbf{DO}_{9...0}$ se pojavljuju signali $\mathbf{DATA1}_{9...0}$ koja predstavlja broj bloka u koji se preslikava stranica. Upis u memoriju

TAGDATA1 se realizuje aktivnom vrednošću signala **wrTAGDATA1**, a čitanje njegovom neaktivnom vrednošću. Memoriji TAGDATA1 se pristupa u sledećim slučajevima:

1. *Utvrđivanje saglasnosti i formiranja fizičke adrese pri utvrđenoj saglasnosti*. Iz memorije TAGDATA1 se čita. Pročitani sadržaj sa linija $\mathbf{TAG1_{14...0}}$ se vodi na komparator CMP1 gde se upoređuje sa sadržajem sa linija $\mathbf{TAGIN_{14...0}}$. Ukoliko saglasnost postoji, pročitani sadržaj na linijama $\mathbf{DATA1}_{9...0}$ predstavlja broj bloka u koji se preslikava stranica, pa se sadržaj sa linija $\mathbf{DATA1}_{9...0}$ propušta kroz multiplekser MP i kao sadržaj $\mathbf{DATA}_{9...0}$ vodi u blok *cpu_interfejs* na ulaze 19…10 registra $URA_{19...10}$ radi formiranja fizičke adrese.

2. *Upisivanje broja procesa, broja stranice i broja bloka kod dovlačenja deskriptora stranice*. U memoriju TAGDATA1 se upisuje. Sadržaj koji se upisuje je $\mathbf{TAGIN_{14...0}}$ i $\mathbf{MDR_{13...4}}$, gde signali $\mathbf{TAGIN_{14...12}}$ predstavljaju signale broja procesa, signali $\mathbf{TAGIN_{11...0}}$ predstavljaju signale 12 najstarijih bitova broja stranice i signali $\mathbf{MDR_{13...4}}$ predstavljaju signale broja bloka u koji se stranica preslikava.

Komparator CMP1 poredi sadržaj na linijama $\mathbf{TAGIN_{14...0}}$, gde signali $\mathbf{TAGIN_{14...12}}$ predstavljaju signale broja procesa i signali $\mathbf{TAGIN_{11...0}}$ predstavljaju signale 12 najstarijih bitova broja stranice zahteva za preslikavanje, i sadržaj na linijama $\mathbf{TAG1_{14...0}}$, gde signali $\mathbf{TAG1_{14...12}}$ predstavljaju signale broja procesa i signali $\mathbf{TAG1_{11...0}}$ predstavljaju signale 12 najstarijih bitova broja stranice adresirane lokacije memorije TAGDATA1 ulaza jedan. Ukoliko se sadržaji slažu, signal **EQL1** na izlazu komparatora CMP1 ima vrednost 1. Ukoliko je i signal **V1** bloka *indikatori* 1, što ukazuje da je sadržaj adresirane lokacije memorije TAGDATA1 ulaza jedan važeći, na izlazu I kola se formira vrednost 1 signal **HIT1**, čime se utvrđuje da postoji saglasnost u memoriji TAGDATA1 ulaza jedan.

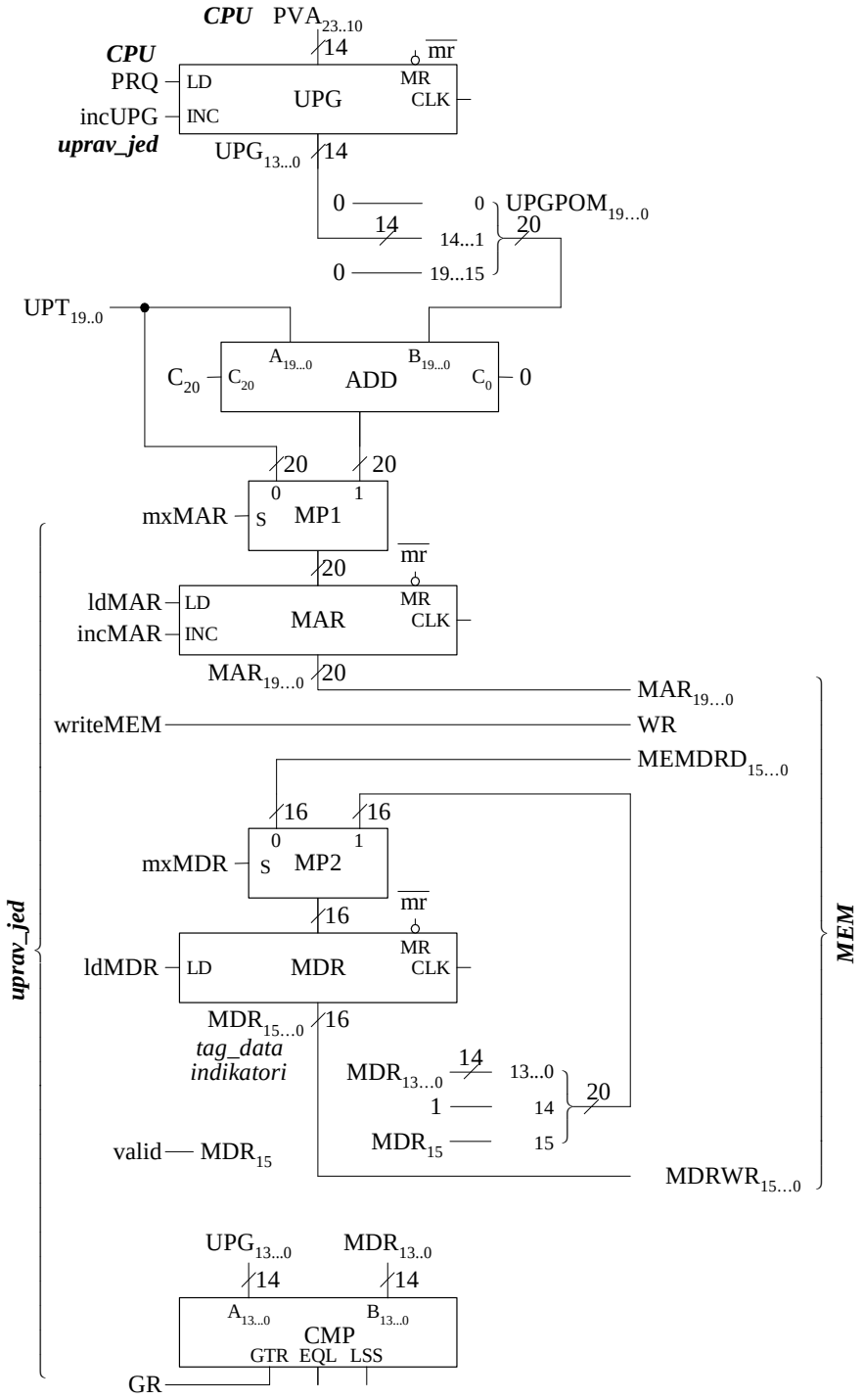
Dekoder DC se koristi da se pri aktivnoj vrednosti signala **wrTAGDATA**, a u zavisnosti od toga da li je vrednost signala **selSENT** bloka *brojači* nula ili jedan, formira aktivna vrednost ili signala **wrTAGDATA0** ili signala **wrTAGDATA1**, respektivno. Aktivna vrednost signala **wrTAGDATA** se generiše pri dovlačenju deskriptora stranice iz tabele stranica u jedinicu za preslikavanje da bi se broj procesa, broj stranice i broj bloka upisali u *i*-tu lokaciji memorije TAGDATA0 ulaza nula ili u *i*-tu lokaciji memorije TAGDATA1 ulaza jedan koje pripadaju *i*-tom setu, *i*=0,…3. Signal **selSENT** vrednostima 0 i 1 određuje da li upis treba da se realizuje u memoriju TAGDATA0 ulaza nula ili memoriju TAGDATA1 ulaza jedan.

Multiplekser MP se koristi da se vrednostima 0 i 1 signala **selSENT** bloka *brojači* na izlaznim linijama $\mathbf{DATA}_{9...0}$ multipleksera MP selektuje jedan od sadržaja na linijama $\mathbf{DATA0}_{9...0}$ ili $\mathbf{DATA1}_{9...0}$, respektivno. Signal **selSENT** vrednostima 0 i 1 određuje da li je otkrivena saglasnost u memoriju TAGDATA0 ulaza nula, pa kao sadržaj $\mathbf{DATA}_{9...0}$ treba koristiti sadržaj sa linija $\mathbf{DATA0}_{9...0}$, ili u memoriju TAGDATA1, pa kao sadržaj $\mathbf{DATA}_{9...0}$ treba koristiti sadržaj sa linija $\mathbf{DATA1}_{9...0}$. Sadržaj $\mathbf{DATA}_{9...0}$ se vodi u blok *cpu_interfejs* na radi formiranja fizičke adrese.

Logičko I kolo se koristi za formiranja signala saglasnosti HIT jedinice za preslikavanje. Saglasnost je otkrivena i signal HIT ima vrednost jedan ukolikoje saglasnost otkrivena bilo u *i*-tu lokaciji memorije TAGDATA0 ulaza nula ili u *i*-tu lokaciji memorije TAGDATA1 ulaza jedan koje pripadaju *i*-tom setu, *i*=0,…3 , pa vrednost 1 ima ili signal HIT0 ili signal HIT1, reapektivno.

2.2.2.3.3.1.5. Blok *mem_interfejs*

Blok *mem_interfejs* (slika 38) sadrži registar $MAR_{19...0}$ sa multiplekserom MP1, registar $MDR_{15...0}$ sa multiplekserom MP2, registar $UPG_{13...0}$, sabirač ADD i komparator CMP.



Slika 38 Blok *mem_interfejs*

Registar $UPG_{13...0}$ (*Unit PaGe register*) služi za čuvanje broja stranice virtuelne adrese. Broj stranice dolazi po linijama $PVA_{23...10}$ procesora ***CPU*** i upisuje se u registar $UPG_{13...0}$ pri aktivnoj vednosti signala **PRQ** procesora ***CPU***. Sadržaj registra $UPG_{13...0}$ se inkrementira pri aktivnoj vednosti signala **incUPG** trajanja jedna perioda signala takta. Ovaj registar se koristi za formiranje signala pomeraja $\mathbf{UPGPOM_{19...0}}$ u odnosu na početnu adresu tabele stranica sa kojim treba očitati descriptor stranice iz tabele stranica. Pošto je veličina jednog ulaza u tabeli

stranica dve reči i pošto se u nultom ulazu tabele stranica nalazi ukupan broj stranica procesa na koji se tabela odnosi, pomeraj $UPGPOM_{19...0}$ se formira kao $(UPG_{13...0} +1)\cdot 2$. Zbog toga se aktivnom vrednošću signala **incUPG** najpre inkrementira sadržaj registra $UPG_{13...0}$, pa se posle toga sadržaj registra $UPG_{13...0}$ pomeren ulevo za jedno mesto, čime se vrši množenje broja stranice sa dva, koristi za formiranje signala pomeraja $\mathbf{UPGPOM_{19...0}}$.

Sabirač ADD se koristi za izračunavanje adrese sa koje treba iz tabele stranica pročitati deskriptor stranice. Na ulaze $A_{19...0}$ sabirača se dovodi sadržaj registra adrese tabele stranica $UPT_{19...0}$ bloka *cpu_interfejs*, a na ulaze $B_{19...0}$ sadržaj signala pomeraja $\mathbf{UPGPOM_{19...0}}$. Na izlazima sabirača ADD se formira adresa sa koje treba pročitati descriptor stranice.

Registar $MAR_{19...0}$ (*Memory Address Register*) služi za čuvanje ili adrese lokacije memorije ***MEM*** sa koje treba pročitati podatak, koji može biti ili niža reč deskriptora stranice ili niža reč nultog ulaza tabele stranica, ili adrese lokacije memorije ***MEM*** u koju treba upisati nižu reč deskriptora stranice u kojoj je bit D izmenjen na 1. Adresa se upisuje u registar $MAR_{19...0}$ pri aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje može biti ili sadržaj registra $UPT_{19...0}$, koji predstavlja adresu niže reči nultog ulaza u tabeli stranica, ili sadržaj formiran na izlazima sabirača ADD, koji predstavlja adresu niže reči deskriptora stranice u tabeli stranica. Selekcija jedne od te dve vrednosti kroz multiplekser MP1 se realizuje upravljačkim signalom **mxMAR**. Sadržaj registra $MAR_{19...0}$ se vode na adresne linije memorije ***MEM***.

Registar $MDR_{15...0}$ (*Memory Data Register*) služi za čuvanje podatka koji je pročitan iz memorije ***MEM*** i za čuvanje podatka koji treba upisati u memoriju ***MEM***. Podatak pročitan iz memorije ***MEM***, koji može biti ili niža reč deskriptora stranice ili niža reč nultog ulaza tabele stranica, dolazi po linijama podataka $MEMDRD_{\mathbf{15...0}}$ iz memorije ***MEM***. Podatak koji treba upisati u memoriju ***MEM*** je sadržaj registra $MDR_{15...0}$ u kome je bit 14 postavljen na 1 i predstavlja nižu reč deskriptora stranice u kojoj je bit D postavljen na 1. Podatak se upisuje u registar $MDR_{15...0}$ na signal takta pri aktivnoj vrednosti signala **ldMDR**. Selekcija jedne od te dve vrednosti kroz multiplekser MP2 se realizuje upravljačkim signalom **mxMDR**. Izlazi registra $MDR_{15...0}$ se vode na ulazne linije podataka memorije ***MEM***. Izlazi registra $MDR_{13...4}$ se vode i u memoriju DATA bloka *tag_data* radi upisa broja bloka deskriptora stranice u memoriju DATA, izlaz MDR_{14} se vodi u blok *indikatori* radi upisa bita D deskriptora stranice u odgovarajući flip-flop D_7 do D_0 i izlazi $MDR_{15...0}$ se, sa razredom MDR_{14} postavljenim na 1, preko multipleksera MP2 vraćaju u registar $MDR_{15...0}$.

Komparator CMP poredi broj stranice virtuelne adrese uvećan za jedan iz registra $UPG_{13...0}$ i broj stranica procesa iz razreda 13 do 0 registra $MDR_{15...0}$, koji predstavljaju broj stranica procesa pročitan iz nultog ulaza tabele stranica, i na izlazu GTR generiše signal greške **GR** zbog zahteva za preslikavanje nepostojeće stranice. Signal **GR** se koristi kao signal logičkog uslova u upravljačkoj jedinici. Ako je signal **GR** na aktivnoj vrednosti, generiše se aktivna vrednost signala **UAV** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice.

Prilikom čitanja deskriptora stranice u razredu MDR_{15} se nalazi indikator koji ukazuje da li je stranica u operativnoj memoriji. Signal iz razreda MDR_{15} daje signal greške **valid** zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji. Signal **valid** se koristi kao signal logičkog uslova u upravljačkoj jedinici. Ako je signal **valid** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **UPF** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji.

2.2.2.3.3.2. UPRAVLJAČKA JEDINICA

U ovom poglavlju se daju dijagram toka zahteva, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.

2.2.2.3.3.2.1. Dijagram toka zahteva

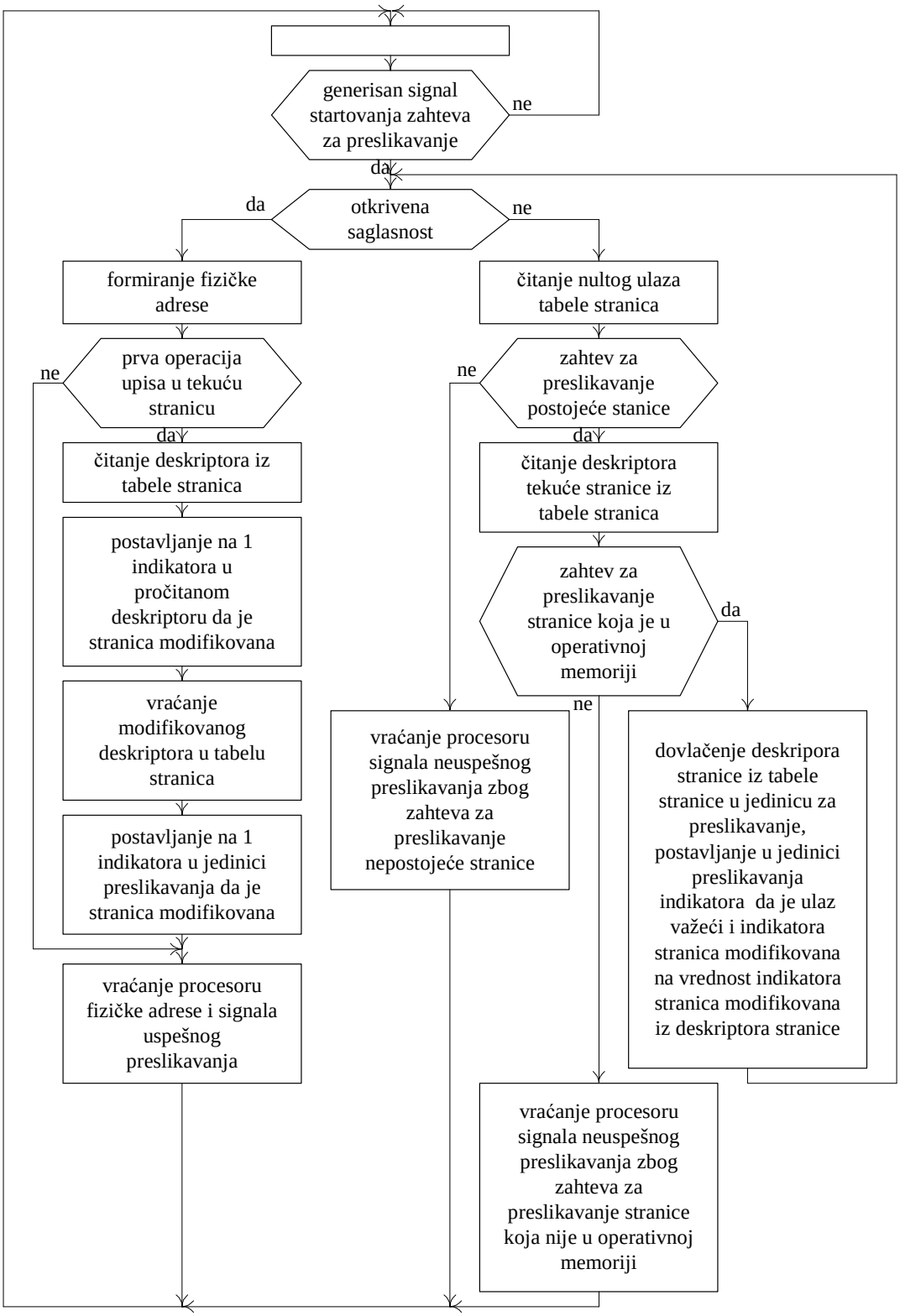
Na početku dijagrama toka se čeka pojava aktivne vrednosti signala startovanja zahteva za preslikavanje (slika 39). Pri njenom pojavljivanju iz procesora se u registre jedinice za preslikavanje upisuju virtuelna adresa, broj procesa, početna adresa tabele stranica i indikator operacije čitanja ili upisa i startuje zahtev u jedinici za preslikavanje.

Najpre se proverava da li postoji saglasnost virtuelne adrese sa sadržajem ulaza nula ili ulaza jedan zadatog seta jedinice za preslikavanje, čime se utvrđuje da li se deskriptor stranice nalazi u jedinici preslikavanja.

Ukoliko postoji saglasnost vrši se formiranje fizičke adrese. Zatim se proverava da li je preslikavanje realizovano za operaciju upisa i to za prvu operaciju upisa na tekućoj stranici. Ukoliko jeste, iz tabele stranica se dovlači deskriptor, zatim se u njemu postavlja na 1 indikator da je stranica modifikovana i na kraju modifikovani deskriptor stranice vraća u tabelu stranica. Pored toga u jedinici preslikavanja na 1 se postavlja i indikator da je stranica modifikovana. Na kraju zahteva se procesoru vraća fizička adresa i signal da je preslikavanje uspešno realizovano i prelazi na početni korak u kome se čeka pojava aktivne vrednosti signala startovanja sledećeg zahteva za presliikavanje. Ukoliko se ne radi o prvoj operaciji upisa na datoj stranici, odmah se procesoru vraća fizička adresa i signal da je preslikavanje uspešno realizovano i prelazi na početni korak.

Ukoliko ne postoji saglasnost prelazi se na dovlačenje niže reči nultog ulaza tabele stranica u kome se nalazi ukupan broj stranica procesa i poređenje ukupnog broja stranica procesa sa brojem stranice iz virtuelne adrese. Ukoliko je broj stranice virtuelne adrese veći od broja stranica procesa, radi se o zahtevu za preslikavanje nepostojeće stranice, pa se procesoru šalje signal da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice i prelazi na početni korak. U suprotnom slučaju se produžava sa preslikavanjem tako što se čita deskriptor stranice iz tabele stranica i proverava da li se stranica nalazi u operativnoj memoriji. Ukoliko se stranica ne nalazi u operativnoj memoriji, radi se o zahtevu za preslikavanje stranice koja nije u operativnoj memoriji, pa se procesoru šalje signal da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje stranice koja nije u operativnoj memoriji i prelazi na početni korak.

U suprotnom slučaju se produžava sa preslikavanjem tako što se čita deskriptor stranice iz tabele stranica i dovlači u jedinicu za preslikavanje. Pored toga u jedinici preslikavanja se postavljaju i indikator da je ulaz važeći na 1 i indikator da je stranica modifikovana na vrednost indikatora da je stranica modifikovana iz deskriptora stranice. Potom se vraća na korak u kome se proverava da li postoji saglasnost. Pošto se sada utvrđuje da postoji saglasnost, jer se deskriptor nalazi u jedinici preslikavanja, produžava se sa već opisanim koracima za slučaj kada se deskriptor nalazi u jedinici preslikavanja.



Slika 39 Dijagram toka operacija

2.2.2.3.3.2.2. Algoritam generisanja upravljačkih signala

Algoritam generisanja upravljačkih signala je formiran na osnovu strukture operacione jedinice (poglavlje 2.2.2.3.3.1) i dijagrama toka zahteva (poglavlje 2.2.2.3.3.2.1). Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je identična kao i notacija za algoritam generisanja upravljačkih signala za upravljačku jedinicu procesora ***CPU*** (poglavlje 2.2.2.1.2.2).

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

$step_0$: *br* (*if* (**PRQ**) *then* $step_1$ *else* $step_0$)

! U koraku $step_0$ se čeka da se aktivom vrednošću signala **PRQ** procesora ***CPU*** iz registara procesora ***CPU*** u registre jedinice ***TLB*** upišu podaci neophodni za preslikavanje i startuje samo preslikavanje. Pri aktivnoj vrednosti signala **PRQ** trajanja jedna perioda signala takta virtuelna adresa, broj tekućeg procesa, početna adresa tabele stranica i tip operacije se upisuju u registre $UVA_{23...0}$, $UPR_{2...0}$ i $UPT_{19...0}$ i flip-flop URW bloka *cpu_interfejs*, respektivno, dok se samo broj stranice virtuelne adrese upisuje u registar $UPG_{13...0}$ bloka *mem_interfejs*. Pri aktivnoj vrednosti signala **PRQ** se prelazi na korak step1, dok se u suprotnom slučaju ostaje u koraku $step_0$. !

$step_1$: *if* (**HIT**, **ldUPA**),
if ($\overline{\textbf{HIT}}$, **ldMAR**, **incUPG**),
br (*if* **HIT** *then* $step_2$ *else* $step_6$)

! U korak $step_1$ može da se dođe ili iz koraka $step_0$ ili iz koraka $step_9$. Iz koraka $step_0$ se dolazi po startovanju svakog novog zahteva za preslikavanje. Iz koraka $step_9$ se dolazi po nekom zahtevu za preslikavanje za koji je, pri prethodnom prolasku kroz $step_1$, otkriveno da nema saglasnosti, pa se išlo na dovlačenje deskriptora stranice iz memorije ***MEM*** u jedinicu ***TLB***. U koraku $step_1$ se vrši provera saglasnosti i na osnovu toga formira vrednost signala **HIT** bloka *tag_data*. Ako se pojavi aktivna vrednost signala **HIT**, što znači da je otkrivena saglasnost, generiše se aktivna vrednost signala **ldUPA** bloka *cpu_interfejs* trajanja jedna perioda signala takta. Njome se fizička adresa, formirana od broja bloka pročitanog iz memorije TAGDATA0 ulaza nula ili memorije TAGDATA1 ulaza jedan koji dolazi po linijama $\textbf{DATA}_{\textbf{9...0}}$ bloka *tag_data* i adrese reči u bloku virtuelne adrese koja dolazi po linijama $\textbf{UVA}_{\textbf{9...0}}$ bloka *cpu_interfejs*, upisuje u registar $UPA_{19...0}$ bloka *cpu_interfejs*. U slučaju da je neaktivna vrednost signal **HIT**, što znači da nije otkrivena saglasnost, mora da se prvo prođe kroz korake $step_6$, $step_7$, $step_8$ i $step_9$ i da se u njima iz tabele stranica u jedinicu za preslikavanje dovuče deskriptor tekuće stranice, pa da se zatim ponovo dođe u korak $step_1$. Stoga se tada generiše aktivna vrednost signala **ldMAR** trajanja jedna perioda signala takta bloka *mem_interfejs* kojom se u registar $MAR_{19...0}$ upisuje sadržaj registra $UPT_{19...0}$ u kome se nalazi adresa nultog ulaza tabele stranica. Takođe se generiše i aktivna vrednost signala **incUPG** trajanja jedna perioda signala takta bloka *mem_interfejs* kojom se za jedan uvećava sadržaj registra $UPG_{13...0}$. Time se obezbeđuje da se u registru $UPG_{13...0}$ nalazi broj stranice virtuelne adrese plus 1. Ovo se radi zbog toga što se deskriptor *i*-te stranice nalazi u (*i*+1)-om ulazu tabele stranica. Pri aktivnoj vrednosti signala **HIT** se prelazi na korak $step_2$, dok se u suprotnom slučaju prelazi na korak $step_6$. !

$step_2$: *if* ($\overline{\textbf{CH}}$, **URP**),
if (**CH**, **mxMAR**, **ldMAR**),
br (*if* **CH** *then* $step_3$ *else* $step_0$)

! U korak $step_2$ može da se dođe jedino iz koraka $step_1$ i to samo onda kada je u koraku $step_1$ otkrivena saglasnost i preslikavanje uspešno realizovano. U koraku $step_2$ se proverava vrednost signala **CH** bloka *tag_data*, koji može da ima aktivnu vrednosti samo ukoliko je preslikavanje realizovano za operaciju upisa i to za prvu operaciju upisa na tekućoj stranici. U slučaju da je signal **CH** bloka *tag_data* na neaktivnoj vrednosti, generiše se aktivna vrednosta signala **URP** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje uspešno realizovano i da se u registru $UPA_{19...0}$ bloka *cpu_interfejs* nalazi fizička adresa. U slučaju da je signal **CH** na aktivnoj vrednosti, mora da se prođe kroz korake $step_3$, $step_4$ i $step_5$ i da se u njima najpre u bit D deskriptora tekuće stranice u tabeli stranica upiše vrednost 1, pa da se tek onda generiše aktivna vrednosta signala **URP**. Stoga se u koraku $step_2$ generišu aktivne vrednosti signala **mxMAR** i **ldMAR** bloka *mem_interfejs* kojima se adresa ulaza u tabelu stranica formirana na izlazu sabirača ADD i propušta kroz multiplekser MP1 i upisuje u registar $MAR_{19...0}$. Pri aktivnoj vrednosti signala **CH** se prelazi na korak $step_3$, dok se u suprotnom slučaju prelazi na korak $step_0$. !

$step_3$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),

br (*if* **MEMFC** *then* step4 *else* step3)

! U korak $step_3$ se dolazi samo iz koraka $step_2$. U koraku $step_3$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* kojom se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači*, generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs*, čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost očitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_4$, dok se u suprotnom slučaju ostaje u koraku $step_3$. !

$step_4$: **mxMDR**, **ldMDR**, **stD**,
br $step_5$

! U korak $step_4$ se dolazi samo iz koraka $step_3$. U registru $MDR_{15...0}$ bloka *cpu_interfejs* se nalazi pročitan deskriptor stranice čiji bit D treba postaviti na 1. Stoga se bezuslovno generišu aktivne vednosti signala **mxMDR** i **ldMDR** bloka *mem_interfejs* kojima se 16-bitna reč, koja na pozicijama 15, 13,…0 ima razrede MDR_{15} i $MDR_{13...0}$ i na poziciji 14 ima 1, propušta kroz multiplekser MP2 i upisuje u registar $MDR_{15...0}$. Pored toga bezuslovno se generiše i aktivna vednost signala **stD** bloka *indikatori* kojom se postavlja na aktivnu vrednost jedan od flip-flopova V_0 do V_7 bloka *indikatori* koji odgovara ulazu jedinice ***TLB*** u kome je prilikom preslikavanja otkrivana saglasnost. Iz koraka $step_4$ se uvek prelazi na korak $step_5$. !

$step_5$: **writeMEM**, **incMEMACC**,
if (**MEMFC**, **URP**),
br (*if* **MEMFC** *then* $step_0$ *else* $step_5$)

! U korak $step_5$ se dolazi samo iz koraka $step_4$. U koraku $step_5$ se bezuslovno generišu aktivne vrednosti signala **writeMEM** bloka *mem_interfejs* i **incMEMACC** bloka *brojači*. Aktivnom vrednošću signala **writeMEM** se modifikovani sadržaj registra $MDR_{15...0}$ bloka *mem_interfejs* upisuje u memoriju ***MEM*** na adresi određenoj sadržajem registra $MAR_{19...0}$ bloka *mem_interfejs*. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** bloka *brojači* generiše se aktivna vrednost signala **URP** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje uspešno realizovano i da se u registru $UPA_{19...0}$ bloka *cpu_interfejs* nalazi fizička adresa. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_0$, dok se u suprotnom slučaju ostaje u koraku $step_5$. !

$step_6$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* $step_7$ *else* $step_6$)

! U korak $step_6$ se dolazi iz koraka $step_1$ kada se utvrdi da nema saglasnosti, pa se prelazi na korake dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu za preslikavanje. U koraku $step_6$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs* čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost pročitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč nultog ulaza tabele stranica tekućeg procesa. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_7$, dok se u suprotnom slučaju ostaje u koraku $step_6$. !

$step_7$: *if* (**GR**, **UAV**),
if ($\overline{\textbf{GR}}$, **mxMAR**, **ldMAR**),
br (*if* **GR** *then* $step_0$ *else* $step_8$)

! U korak $step_7$ se dolazi samo iz koraka $step_6$. U koraku $step_7$ se proverava da li se zahtev za preslikavanje odnosi na postojeću stranicu procesa. Signal **GR** bloka *mem_interfejs* predstavlja izlaz GTR komparatora CMP koji poredi broj stranice virtuelne adrese uvećan za jedan iz registra $UPG_{13...0}$ i broj stranica procesa iz registra $MDR_{13...0}$. Ako je signal **GR** na aktivnoj vrednosti, generiše se aktivna vrednost signala **UAV** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje nepostojeće stranice. Ako je siganal **GR** na neaktivnoj vrednosti, produžava se sa koracima dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu jedinicu za preslikavanje tako što se generišu aktivne vrednosti signala **mxMAR** i **ldMAR** bloka *mem_interfejs* kojima se adresa ulaza u tabelu stranica formirana na izlazu sabirača ADD propušta kroz multiplekser MP1 i upisuje u registar $MAR_{19...0}$. Pri aktivnoj vrednosti signala **GR** se prelazi na koraka $step_0$, dok se u suprotnom slučaju prelazi na korak $step_8$. !

$step_8$: **incMEMACC**, *if* (**MEMFC**, **ldMDR**),
br (*if* **MEMFC** *then* $step_9$ *else* $step_8$)

! U korak $step_8$ se dolazi samo iz koraka $step_7$. U koraku $step_8$ se bezuslovno generiše aktivna vrednost signala **incMEMACC** bloka *brojači* čime se obezbeđuje da se inkrementira sadržaj brojača $MEMACC_{1...0}$ koji određuje vreme pristupa memoriji ***MEM***. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR** bloka *mem_interfejs* čime se obezbeđuje da se u registar $MDR_{15...0}$ upiše vrednost pročitana iz memorije ***MEM*** sa adrese određene sadržajem registra $MAR_{19...0}$, što je u ovom slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se prelazi na korak $step_9$, dok se u suprotnom slučaju ostaje u koraku $step_8$. !

$step_9$: *if* ($\overline{\textbf{valid}}$, **UPF**),
if (**valid**, **wrTAGDATA**, **stV**, **wrD, incFIFOCNT**),
br (*if* **valid** *then* $step_1$ *else* $step_0$)

! U korak $step_9$ se dolazi samo iz koraka $step_8$. U koraku $step_9$ se proverava da li se zahtev za preslikavanje odnosi na stranicu procesa koja je u operativnoj memoriji. Signal **valid** bloka *mem_interfejs* predstavlja vrednost V bita deskriptora stranice koji vrednostima 0 i 1 određuje da se stranica ne nalazi i nalazi u operativnoj memoriji, respektivno. Ako je signal **valid** na neaktivnoj vrednosti, generiše se aktivna vrednost signala **UPF** bloka *cpu_interfejs* kojom jedinica ***TLB*** daje procesoru ***CPU*** indikaciju da je preslikavanje neuspešno realizovano zbog zahteva za preslikavanje za stranicu koja nije u operativnoj memoriji. Ako je siganal **valid** na neaktivnoj vrednosti, produžava se sa koracima dovlačenja deskriptora tekuće stranice iz tabele stranica u jedinicu za preslikavanje tako što se generišu aktivne vrednosti signala **wrTAGDATA**, **stV, wrD** i **incFIFOCNT**. Aktivna vrednost signala **wrTAGDATA** bloka *tag_data* omogućuje da se u memoriju TAGDATA0 ulaza nula ili memoriju TAGDATA1 ulaza jedan upišu broj procesa i broj stranice sa linija $\textbf{TAGIN}_{\textbf{14...0}}$ i broj bloka sa linija $\textbf{MDR}_{\textbf{13...4}}$ bloka *mem_interfejs*. Selekcija memorije TAGDATA0 ulaza nula ili memorije TAGDATA1 ulaza jedan se vrši na osnovu sadržaja brojača za zamenu zadatog seta. Aktivnom vrednošću signala **stV** bloka *indikatori* se u jedan od flip-flopova $V0_0$ do $V0_3$ ulaza nula ili u jedan od flip-flopova $V1_0$ do $V1_3$ postavlja na 1. Selekcija flip-flopova $V0_0$ do $V0_3$ ulaza nula ili flip-flopova $V1_0$ do $V1_3$ ulaza jedan se vrši na osnovu sadržaja brojača za zamenu zadatog seta. Aktivnom vrednošću signala **wrD** bloka *indikatori* se jedan od flip-flopova $D0_0$ do $D0_3$ ulaza nula ili jedan od flip-flopova $D1_0$ do $D1_3$ postavlja na vrednost indikatora D dovučenog deskriptora stranice koji se nalazi u razredu MDR_{14} bloka *mem_interfejs*. Selekcija flip-flopova $D0_0$ do $D0_3$ ulaza nula ili flip-flopova $D1_0$ do $D1_3$ ulaza jedan se vrši na osnovu sadržaja brojača za zamenu zadatog seta. Aktivnom vrednošću signala **incFIFOCNT** bloka *brojači* inkrementira se sadržaj brojača za zamenu zadatog seta. Pri aktivnoj vrednosti signala **valid** se prelazi na korak $step_1$, dok se u suprotnom slučaju prelazi na korak $step_0$. !

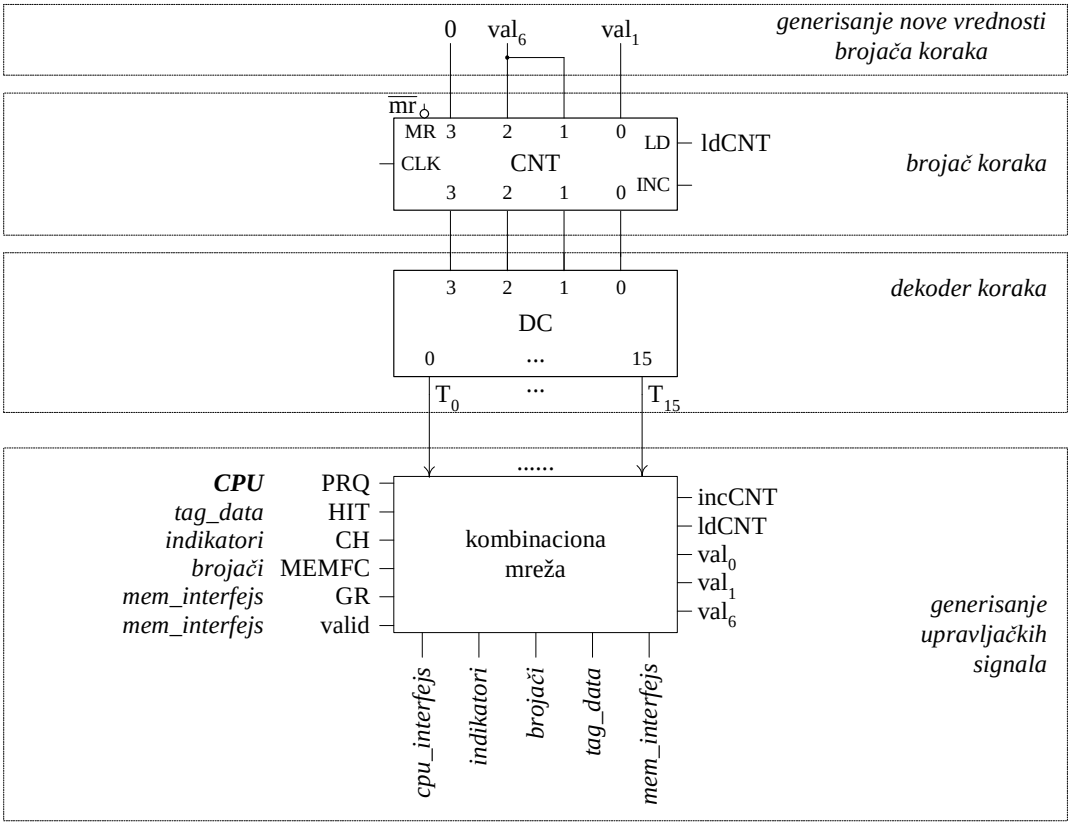
2.2.2.3.3.2.3. Struktura upravljačke jedinice

Upravljačka jedinica (slika 40) se sastoji od sledećih blokova:

- blok *generisanje nove vrednosti brojača koraka*,
- blok *brojač koraka*,
- blok *dekoder koraka* i
- blok *generisanje upravljačkih signala*.

Struktura i opis blokova upravljačke jedinice se daju u daljem tekstu.

Blok *generisanje nove vrednosti brojača koraka* služi za generisanje vrednosti koju treba upisati u brojač CNT. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog generisanja upravljačkih signala. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje 2.2.2.3.3.2.2) se utvrđuje da su 0, 1 i 6 vrednosti koje treba upisati u brojač CNT da bi se realizovala odstupanja od sekvencijalnog generisanja upravljačkih signala. Te vrednosti se formiraju na ulazima 3, 2, 1 i 0 brojača CNT pomoću signala $\textbf{val}_\textbf{0}$, $\textbf{val}_\textbf{1}$ i $\textbf{val}_\textbf{6}$ pri čemu signal $\textbf{val}_\textbf{0}$ ima vrednost 1 samo onda kada treba upisati 0, signal $\textbf{val}_\textbf{1}$ ima vrednost 1 samo onda kada treba upisati 1 i signal $\textbf{val}_\textbf{6}$ ima vrednost 6 samo onda kada treba upisati 6. Time se obezbeđuje da na ulazima 3, 2, 1 i 0 brojača CNT budu vrednosti 0, 0, 0 i 0 onda kada je signal $\textbf{val}_\textbf{0}$ ima vrednost 1, vrednosti 0, 0, 0 i 1 onda kada je signal $\textbf{val}_\textbf{1}$ ima vrednost 1 i vrednosti 0, 1, 1 i 0 onda kada je signal $\textbf{val}_\textbf{6}$ ima vrednost 1.



Slika 40 Struktura upravljačke jedinice

Blok *brojač koraka* sadrži brojač CNT. Brojač CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač CNT može da radi u sledećim režimima:

- režim inkrementiranja,
- režim skoka i
- režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta vrši se uvećavanje sadržaja brojača CNT za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.3.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **incCNT**. Signal **incCNT** je uvek neaktivan sem kada treba obezbediti režim inkrementiranja.

U režimu skoka pri pojavi signala takta vrši se upis nove vrednosti u brojač CNT. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.3.2.2). Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ldCNT**. Signal **ldCNT** je uvek neaktivan sem kada treba obezbediti režim skoka.

U režimu mirovanja pri pojavi signala takta ne menja se vrednost brojača CNT. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **incCNT** i **ldCNT**. Ovi signali su neaktivni kada se čeka pojava aktivne vrednosti signala startovanja zahteva za preslikavanje **PRQ** procesora ***CPU*** ili pojava aktivne vrednosti signala **MEMFC** bloka *brojači* kojim se označava da je završena operacija čitanja iz memorije ***MEM*** ili upisa u memoriju ***MEM***

Blok *dekoder koraka* sadrži dekoder DC. Na ulaze dekodera DC vode se izlazi brojača CNT. Dekodovana stanja brojača CNT pojavljuju se kao signali $\mathbf{T_0}$ do $\mathbf{T_{15}}$ na izlazima dekodera DC. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje 2.2.2.3.3.2.2) dodeljen je jedan od ovih signala i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$, itd.

Blok *generisanje upravljačkih signala* sadrži kombinacione mreže koje pomoću signala $\mathbf{T_0}$ do $\mathbf{T_{15}}$ koji dolaze sa bloka *dekoder koraka* upravljačke jedinice, signala logičkih uslova koji dolaze iz blokova operacione jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje 2.2.2.3.3.2.2) generišu upravljačke signale. Upravljački signali se generišu na identičan način kao i upravljački signali procesora ***CPU*** (poglavlje 2.2.2.1.2.2).

Blok *generisanje upravljačkih signala* generiše dve grupe upravljačkih signala i to:

- upravljačke signale operacione jedinice ***oper_jed*** i
- upravljačke signale upravljačke jedinice ***uprav_jed***.

Upravljački signali operacione jedinice ***oper_jed*** se daju posebno za svaki blok operacione jedinice i to:

- blok *cpu_interfejs*,
- blok *indikatori*,
- blok *brojači*,
- blok *tag_data i*
- blok *cpu_ interfejs.*

Upravljački signali bloka *cpu_interfejs* se generišu na sledeći način:

- $\mathbf{ldUPA = T_1 \cdot HIT}$
- $\mathbf{URP = T_2 \cdot \overline{CH} + T_5 \cdot MEMFC}$
- $\mathbf{UAV = T_7 \cdot GR}$
- $\mathbf{UPF = T_9 \cdot \overline{valid}}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **HIT** — blok *tag_data*,
- **CH** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **GR** — blok *mem_interfejs* i
- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *indikatori* se generišu na sledeći način:

- $\mathbf{stD = T_4}$
- $\mathbf{stV = T_9 \cdot valid}$
- $\mathbf{wrD = T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *brojači* se generišu na sledeći način:

- $\mathbf{incMEMACC = T_3 + T_5 + T_6 + T_8}$
- $\mathbf{incFIFOCNT = T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *tag_data* se generišu na sledeći način:

- **wrTAGDATA** = $\mathbf{T_9 \cdot valid}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **valid** — blok *mem_interfejs*.

Upravljački signali bloka *mem_interfejs* se generišu na sledeći način:

- **incUPG** = $\mathbf{T_1 \cdot \overline{HIT}}$
- **mxMAR** = $\mathbf{T_2 \cdot CH + T_7 \cdot \overline{GR}}$
- **ldMAR** = $\mathbf{T_2 \cdot CH + T_7 \cdot \overline{GR}}$
- **mxMDR** = $\mathbf{T_4}$
- **ldMDR** = $\mathbf{T_3 \cdot MEMFC + T_4 + T_6 \cdot MEMFC + T_8 \cdot MEMFC}$
- **writeMEM** = $\mathbf{T_5}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **HIT** — blok *tag_data*
- **CH** — blok *tag_data*.
- **GR** — blok *mem_interfejs* i
- **MEMFC** — blok *brojači*.

Upravljački signali upravljačke jedinice ***uprav_jed*** se generišu na sledeći način:

- **ldCNT** = $\mathbf{val_1 + val_0 + val_6}$
- **incCNT** = $\mathbf{T_0 \cdot PRQ + T_1 \cdot HIT + T_2 \cdot CH + T_3 \cdot MEMFC + T_4 + T_6 \cdot MEMFC + T_7 \cdot \overline{GR} + T_8 \cdot MEMFC}$
- $\mathbf{val_1}$ = $\mathbf{T_1 \cdot \overline{HIT}}$
- $\mathbf{val_0}$ = $\mathbf{T_2 \cdot \overline{CH} + T_5 \cdot MEMFC + T_7 \cdot GR}$
- $\mathbf{val_6}$ = $\mathbf{T_9 \cdot \overline{valid}}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz blokova operacione jedinice i to:

- **PRQ** — blok *cpu_interfejs*,
- **HIT** — blok *tag_data*,
- **CH** — blok *indikatori*,
- **MEMFC** — blok *brojači*,
- **GR** — blok *mem_interfejs*.

2.3. SEGMENTNA ORGANIZACIJA VIRTUELNE MEMORIJE

U ovoj glavi se daju opšte napomene o usvojenoj organizaciji virtuelnoj memorije stranične organizacije i realizacija sistema preslikavanja.

2.3.1. OPŠTE NAPOMENE

U ovom odeljku se daju opšte napomene o usvojenoj virtuelnoj memorije segmentne organizacije. U okviru toga se daju karakteristike virtuelnog adresnog prostora procesa,

fizičkog adresnog prostora i struktura tabele preslikavanja virtuelnog adresnog prostora u fizički adresni prostor.

Adreasibilna jedinica je reč dužine dva bajta (1W=2B). Veličina virtuelnog adresnog prostora je 16M 16-bitnih reči, a maksimalna veličina jednog segmenta je 64KW. Shodno tome virtuelna adresa je dužine 24 bita, i ima sledeću strukturu:

- višihm = 8 bita označavaju broj segmenta
- nižih k = 16 bita označavaju adresu unutar segmenta

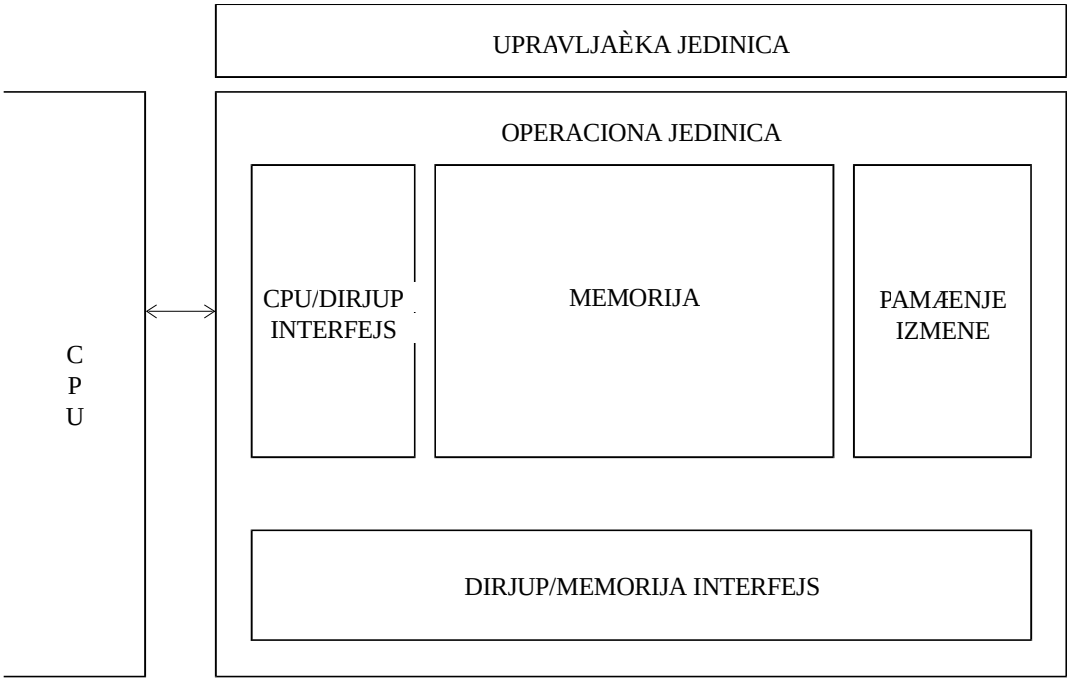
Veličina fizičkog adresnog prostora je 1MW. Realna adresa ima 20 bita i dobija se kao pocetna adresa segmenta u operativnoj memoriji + adresa unutar segmenta.

Struktura jedinice sa direktnim preslikavanjem (u daljem tekstu – **DIRJUP** jedinica), prikazana je na slici 2.1.

DIRJUP jedinica se sastoji iz:

- **operacione jedinice**
- **upravljačke jedinice**

Operaciona jedinica se sastoji iz kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje aritmetičkih, logičkih i drugih mikrooperacija i generisanje logičkih uslova.



Slika 2.1. Strukturna šema DIRJUP jedinice

Upravljačka jedinica se sastoji iz kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu direktnog preslikavanja virtuelne adrese u realnu.

Funkcija i struktura svakog od ovih delova biće objašnjena u posebnom poglavlju.

Operativni sistem i **DIRJUP**, za realizaciju virtuelne memorije, pristupaju zajedničkoj strukturi **SMT** (**S**egment **M**ap **T**able) u kojoj su informacije o preslikavanju segmenata. **SMT** se nalazi u **operativnoj memoriji** u onom delu koji je odvojen za **operativni sistem**. **SMT** se, u ovoj realizaciji, sastoji iz ulaza od po dve reči. U svakom (nenultom) ulazu se čuva deskriptor odgovarajućeg segmenta (koji je takođe dve reči). U nultom ulazu u nižoj reči, operativni sistem postavi broj koji označava koliko segmenata ima program. Taj broj koristi **DIRJUP** za proveru prava pristupa. U prvom ulazu **SMT**-a se nalazi deskriptor nultog segmenta, u drugom deskriptor drugog i.t.d. **SMT** je dugačka onoliko ulaza koliko segmenata ima dati program i plus jedan za nulti ulaz. U nižoj reči nenultog ulaza se nalazi prva reč deskriptora, a u višoj druga.

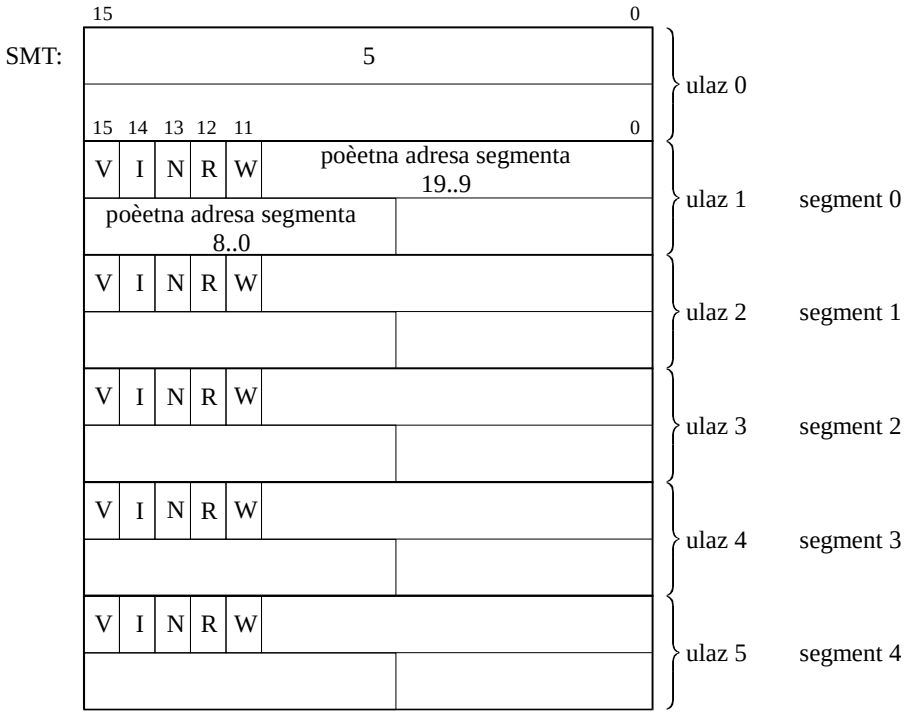
Najviših pet bita prve reči deskriptora su:

- 15-ti V-bit (valid bit) označava da je segment u operativnoj memoriji
- 14-ti I-bit označava da je u segment upisivano (vršen je bar jedan upis u njega) i da taj segment operativni sistem pri izbacivanju iz memorije mora vratiti na disk.
- 13-ti N-bit (i**N**struction bit) označava da je taj segment sadrži isključivo instrukcije, kada je ovaj bit nula to znači da segment sadrži isključivo podatke.
- 12-ti R-bit (Read data) označava da je segment sa podacima i da je dozvoljeno čitanje
- 11-ti W-bit (Write data) označava da je segment sa podacima i da je dozvoljeno čitanje.

Biti N,R i W su biti zaštite. Dozvoljene kombinacije ovih bita date su u sledećoj tabeli:

N	R	W
0	1	0
0	1	1
1	0	0

Inicijalno kada se startuje odgovarajući program, **operativni sistem** formira **SMT** sa potrebnim brojem ulaza, upiše broj segmenata u nižu reč nultog ulaza i postavi sve **V-bit**-e na nula. Onda (pri startovanju tog programa) procesor zadaje virtuelne adrese, pa pri preslikavanju u **DIRJUP**-u, ako segment nije dovučen u memoriju, **DIRJUP** zadaje prekid **S**egment **F**ault koji kao rezultat ima da **operativni sistem** dovuče adresirani segment u **operativnu memoriju** i u ulaz u **SMT**-u tog segmenta postavi odgovajajuće informacije deskriptora čime se i **V-bit** deskriptora setuje na jedan. Izgled **SMT-**a za primer programa od pet segmenata prikazan je na slici 2.2.



Slika 2.2. Izgled SMT za primer programa od pet segmenata

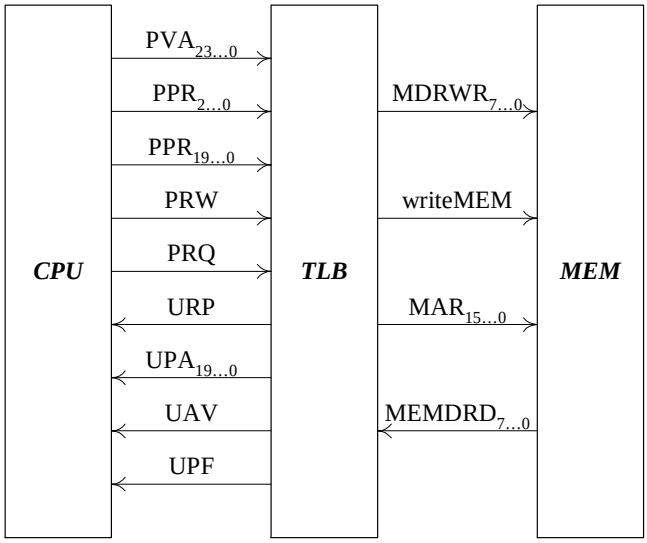
2.3.2. SISTEM ZA PRESLIKAVANJE

U ovom odeljku se daje realizacija sistema za preslikavanje virtuelnog u fizički adresni prostor. Sistem koji se posmatra (slika 8) se sastoji iz:

- procesora ***CPU***,
- operativne memorije ***MEM*** i
- jedinice za preslikavanje ***TLB***.

Uzeto je da ceo sistem radi sinhrono sa zajedničkim signalom takta.

U daljem tekstu se najpre razmatraju samo delovi procesora ***CPU*** bitni za njegov rad sa jedinicom za preslikavanje ***TLB***, zatim se daje realizacija operativne memorije ***MEM*** i na kraju prikazaje realizacija jedinice za preslikavanje ***TLB***.



Slika 41 Struktura sistema

2.3.2.1. PROCESOR CPU

Procesor CPU se obraća DIRJUP onda kada treba očitati podatak iz operativne memorije ili upisati podatak u operativnu memoriju, pa je potrebno izvršiti peslikavanje virtuelne adrese u realnu adresu.

Procesor CPU šalje 24-bitnu virtuelnu adresu po linijama **$PVAR_{23...0}$**, oznaku tekućeg korisnika po linijama **$PUSRR_{2...0}$**, adresu početka tabele segmenata tekućeg korisnika po linijama **$PSMTAR_{19...0}$**, podatak o vrsti operacije (upis ili čitanje) po liniji **R/W**, podatak o tome da li je procesor u fazi izvrsavanja ili fecovanja instrukcije po liniji **E/F** i generiše aktivnu vrednost signala **PREQ**.U slučaju uspešnog prevođenja adrese, generisana 20-bitna realna adresa se vraća po linijama **$URAR_{19...0}$**. Prevođenje je uspešno završeno i na linijama **$URAR_{19...0}$** je važeći podatak onda kada DIRJUP generiše aktivnu vrednost signala **URP**, a flip-lfopovi PAVF i PSFF su postavljeni na neaktivnu vrednost.

Procesor CPU u delu za povezivanje sa DIRJUP jedinicom koristi:

- **registre** PVAR, PUSRR, PSMTAR i PRAR **za čuvanje virtuelne adrese i podataka o korisniku, odnosno za smeštanje generisane realne adrese u slučaju uspešno izvedenog preslikavanja,**
- **brojac** PCNT **za simuliranje pauze izmedju dva zahteva za preslikavanjem,**
- **flip-flopove** PRW, PAVF, PSFF, PEF **za indikaciju o tipu operacije odosno o razlogu nemogućnosti uspešnog preslikavanja i**
- **upravljačke signale** PREQ, AV, SF,URP i URP2 **za sinhronizaciju sa DIRJUP jedinicom**

Registar **PVAR** (*Processor Virtual Address Register*) služi za čuvanje virtuelne adrese lokacije memorije MEM sa koje treba očitati podatak u slučaju operacije čitanja ili adrese lokacije memorije MEM u koju treba upisati podatak u slučaju operacije upisa. Izlazne linije ovog registra se vode kao 24 linije **$PVAR_{23...0}$** u CPU/DIRJUP interfejs direktne jedinice. Pretpostavlja se da je procesor CPU pre obraćanja DIRJUP jedinici već upisao virtuelnu

adresu u registar PVAR, tako što je generisao aktivnu vrednost signala **ldPVAR** i na signal takta **CLK** izvršio upis sadržaja sa linija $\mathbf{PVARIN_{19...0}}$.

Registar **PUSR** (*Processor USer Register*) sadrži oznaku tekućeg korisnika procesora, a registar **PSMTAR** (*Processor Segment Map Table Address Register*) pokazuje na početak tabele segmenata tekućeg korisnika.Pretpostavlja se da je operativni sistem, pri dodeljivanju procesora korisniku, upisao potrebne podatke u ove registre, tako što je generisao aktivnu vrednost signala **ldPUSR** i na signal takta izvršio upis sadržaja sa linija $\mathbf{PUSR_{2...0}}$ u slučaju registra PUSR, odnosno generisao aktivnu vrednost signala **ldPSMTAR** i na signal takta izvršio upis sadržaja sa linija $\mathbf{PSMTAR_{19...0}}$ u slučaju registra **PSMTAR**.

Registar **PRAR** (*Processor Real Address Register*) služi za čuvanje realne adrese koja je generisana u slučaju uspešnog preslikavanja virtuelne u realnu adresu. Generisana adresa se iz **CPU/DIRJUP** interfejsa direktne jedinice vodi po linijama $\mathbf{URAR_{19...0}}$ na ulaze registra PRAR. Aktivna vrednost signala **URP** se koristi da se na signal takta **CLK** izvrši upis realne adrese sa linija $\mathbf{URAR_{19...0}}$ u registar **PRAR,** pod uslovom da su flip-flopovi **PAVF i PSFF** postavljeni na nulu.

Brojac **PCNT** (*Processor CouNTer)* sluzi za simuliranje pauze izmedju dva uzastopna zahteva za preslikavanjem. Aktivnom vrednoscu signala **URP** u **PCNT** se upisuje 4-bitni podatak **PCNTIN** koji se nalazi u tabeli zahteva i oznacava vreme koje je potrebno da prodje izmedju dva uzastopna zahteva za preslikavanjem. **PCNT** je brojac unazad i na svaki signal takta odbroji za 1 unazad. Kada se na izlazu **PCNT** pojavi 0 aktivira se signal **URP2** koji signalizira procesoru da moze da posalje sledeci zahtev.

Flip-flop **PR/WF** (*Processor Read/ Write Flag*) služi za čuvanje podatka o tipu operacije u kojoj se koristi zadata virtuelna adresa. Aktivnom vrednošću signala **R/W** (*Read/Write operation*), koji se dobija sa izlaza **PR/WF** flip-flopa, se specificira operacija čitanja, a neaktivnom operacija upisa. Pretpostavlja se da je procesor **CPU** pre toga generišući aktivnu vrednost ili signala **setR/W** ili signala **clR/W** trajanja jedne periode signala takta **CLK** postavio flip-flop **PR/WF** ili na aktivnu vrednost ili na neaktivnu vrednost, respektivno. Podatak o tipu operacije prosleđuje se **DIRJUP** jedinici na signal takta **CLK**, ako je aktivan signal **PREQ**.

Flip-flop **PAVF** (*Procesor Address Violation Flag*) služi za indikaciju procesoru da je generisana virtuelna adresa neispravna tj. da je broj segmenta na koju se adresa odnosi veći od ukupnog broja segmenta tekućeg korisnika, ili da pristup datom segmentu nije dozvoljen. Ovaj flip-flop postavlja se na aktivnu vrednost na signal takta **CLK** pri aktivnoj vrednosti signala **AV**. Signal **AV** generiše **DIRJUP** jedinica ako u procesu preslikavanja adrese otkrije da je došlo do Address Violation. Resetovanje **PAVF** se vrši pri postavljanju novog zahteva za preslikavanje tj. pri aktivnoj vrednosti signala **PREQ**.

Flip-flop **PSFF** (*Procesor Segment Fault Flag*) služi za indikaciju procesoru da se segment na koji se generisana virtuelna adresa odnosi ne nalazi u operativnoj memoriji. Procesor na osnovu aktivne vrednosti sadržaja **PSFF** generiše Segment Fault prekid, a operativni sistem obrađujući ovaj prekid suspenduje tekućeg korisnika i organizuje prebacivanje segmenta sa diska u operativnu memoriju. Ovaj flip-flop postavlja se na aktivnu vrednost na signal takta **CLK** pri aktivnoj vrednosti signala **SF**. Signal **SF** generiše **DIRJUP** jedinica ako u procesu preslikavanja adrese otkrije da je došlo do Segment Fault-a. Resetovanje **PSFF** se vrši pri postavljanju novog zahteva za preslikavanje tj. pri aktivnoj vrednosti signala **PREQ**.

Upravljački signal **PREQ** (*Processor REQuest*) se koristi da procesor **CPU** aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira **DIRJUP** jedinici da se na linijama $\mathbf{PVAR_{23...0}}$, $\mathbf{PUSR_{2...0}}$, $\mathbf{PSMTAR_{19...0}}$,**E/W** i **R/W** nalaze podaci potrebni za operaciju preslikavanja virtuelne u realnu adresu, i da treba, na signal takta **CLK**, ove sadržaje upisati u odgavarajuće registre i time pokrenuti preslikavanje.

Upravljački signali **AV** (*Address Violation*) i **SF** (*Segment Fault*) se koriste da DIRJUP jedinica aktivnom vrednošću ovih signala trajanja jedne periode signala takta **CLK** signalizira procesoru **CPU** da se preslikavanje zadate adrese ne može uspešno izvršiti jer je došlo do prekoračenja dozvoljenog opsega adresa ili je došlo do segmentnog prekida. U slučaju aktivne vrednosti signala **AV** na signal takta se upisuje aktivna vrednost u **PAVF** flip-flop, a u slučaju aktivne vrednosti signala **SF** aktivna vrednost se upisuje u **PSFF** flip-flop.

Upravljački signal **URP** (*Unit RePly*) se koristi da **DIRJUP** jedinica aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira procesoru **CPU** da je peslikavanje adrese završeno - ili uspešno ili neuspešno (što se vidi po vrednostima upisanim u **PAVF** i **PSFF**). U slučaju uspešnog peslikavanja iz **DIRJUP** jedinice aktivnom vrednošću signala **URP** se signalizira i da se na linijama $\mathbf{URAR_{19...0}}$ nalazi generisana realna adresa i da aktivnom vrednošću signala **URP** treba na signal takta **CLK** ovu adresu upisati u registar **PRAR**.

Upravljacki signal **URP2** oznacava zakasnjen signal **URP** za onoliko taktova koliki je sadrzaj ulaza u tabelu zahteva **PCNTIN**. Aktivna vrednost signala **URP2** oznacava procesoru da posalje sledeci zahtev za preslikavanje adrese.

2.3.2.2. OPERACIONA DIRJUP JEDINICA

Operaciona **DIRJUP** jedinica (slika 2.1.) sastoji se iz sledećih blokova:

1. **CPU/DIRJUP interfejs**
2. **Memorija**
3. **DIRJUP/MEMORIJA interfejs**
4. **pamćenje izmene**

2.3.2.2.1. CPU/DIRJUP INTERFEJS

Strukturna šema bloka **CPU/DIRJUP interfejsa** prikazana je na slici 2.3.

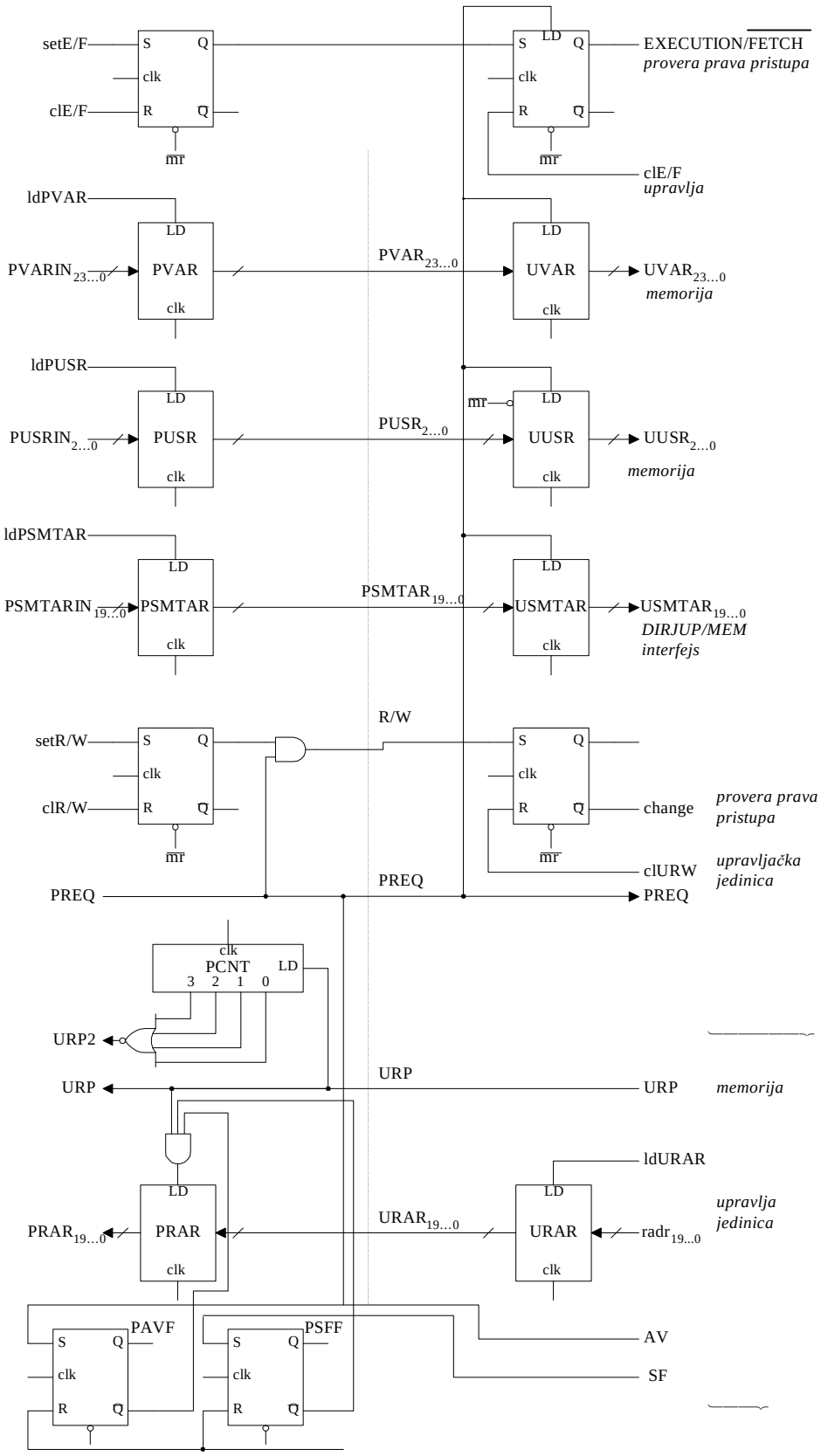
Interfejs sačinjavaju registri:

- **UVAR** (**V**irtual **A**ddress **R**egister)
- **URAR** (**R**eal **A**ddress **R**egister)
- **USMTAR** (**S**egment **M**ap **T**able **A**ddress **R**egister)

Registri **UVAR** i **URAR** služe za prihvatanje virtuelne i realne adrese, respektivno.

Registar **USMTAR** pokazuje na pocetak **SMT** (**S**egment **M**ap **T**able) tabele. Takodje se u njega upisuje vrednost samo pri inicijalizaciji sistema.

CPU i **DIRJUP** rade sa istim signalom takta.


setE/F
clE/F
S
Q
clk
R
mr
LD
EXECUTION/FETCH
provera prava pristupa
clE/F
upravlja
ldPVAR
PVARIN23...0
PVAR
PVAR23...0
UVAR
UVAR23...0
memorija
ldPUSR
PUSRIN2...0
PUSR
PUSR2...0
UUSR
UUSR2...0
memorija
ldPSMTAR
PSMTARIN19...0
PSMTAR
PSMTAR19...0
USMTAR
USMTAR19...0
DIRJUP/MEM
interfejs
setR/W
clR/W
R/W
change
provera prava
pristupa
clURW
upravljačka
jedinica
PREQ
PCNT
3 2 1 0
URP2
URP
memorija
ldURAR
PRAR19...0
PRAR
URAR19...0
URAR
radr19...0
upravlja
jedinica
PAVF
PSFF
AV
SF

Slika 2.3. CPU/DIRJUP interfejs

2.3.2.2.2. MEMORIJA

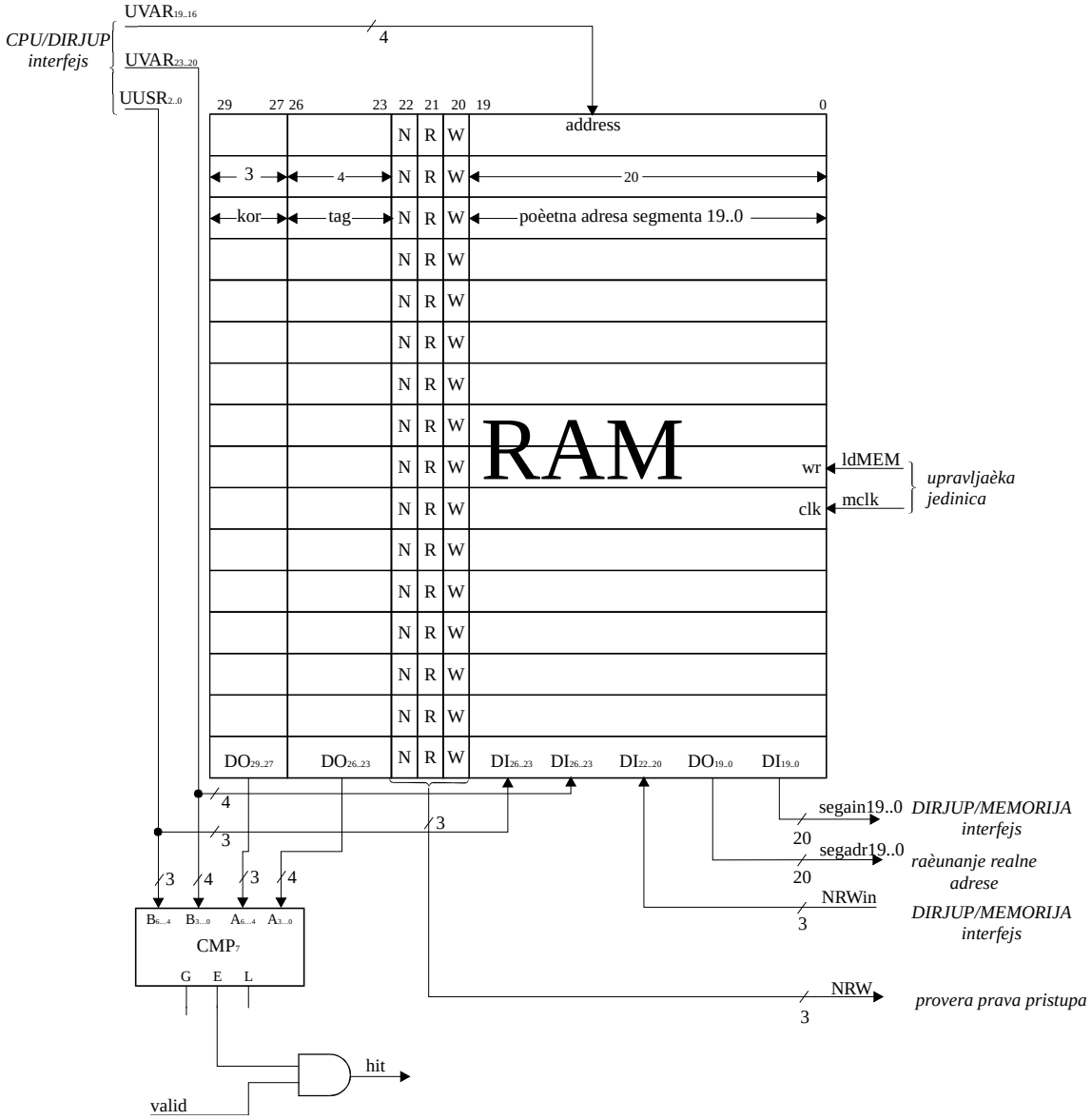
Strukturna šema bloka **MEMORIJA** prikazana je na slikama 2.4., 2.5., 2.6. i 2.7.

Blok **MEMORIJA** sastoji se iz:

- memorije DIRJUP-a
- sabirača za računanje realne adrese
- kola za proveru prava pristupa
- kola sa flip–flop-ovima V0..V15

2.3.2.2.2.1. MEMORIJA DIRJUP-a

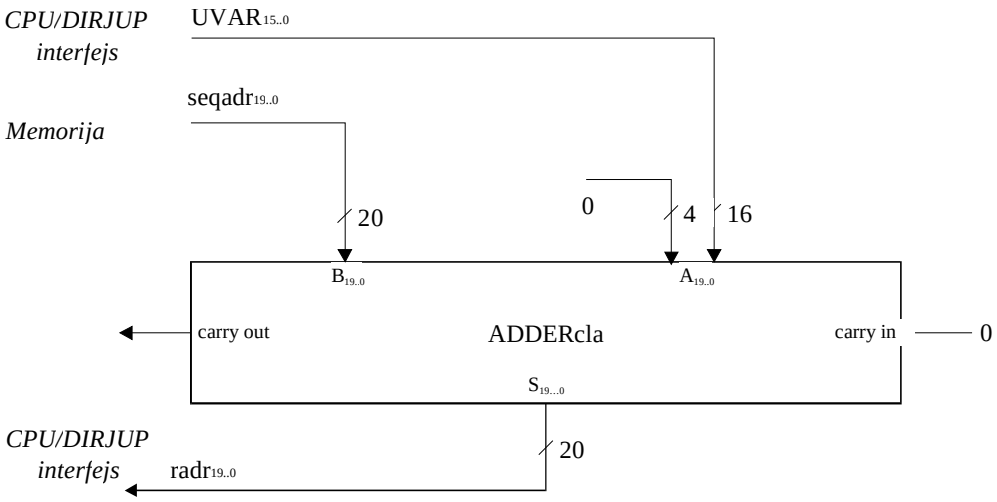
Ovaj blok (slika 2.4.) se sastoji iz brze **RAM** memorije (iste one sa kojom se projektuje i **KEŠ** memorija) i sedmobitnog komparatora i **AND** logičkog kola. Na adresne linije memorije se dovode signali **UVAR23..20**. Posle vremena očitavanja na izlazima **DI29..0** se pojavi sadržaj adresiranog ulaza. Vrednost **DO29..23** (**user+tag**) se poredi sa **UUSR2:0+UVAR23..20** i ako su jednaki i ako je **valid** (deskriptor adresiranog segmenta nalazi se u **DIRJUP**-u tj. odgovarajući **V** flip–flop je na jedinici) , dobija se **hit**.



Slika 2.4. Memorija DIRJUP-a

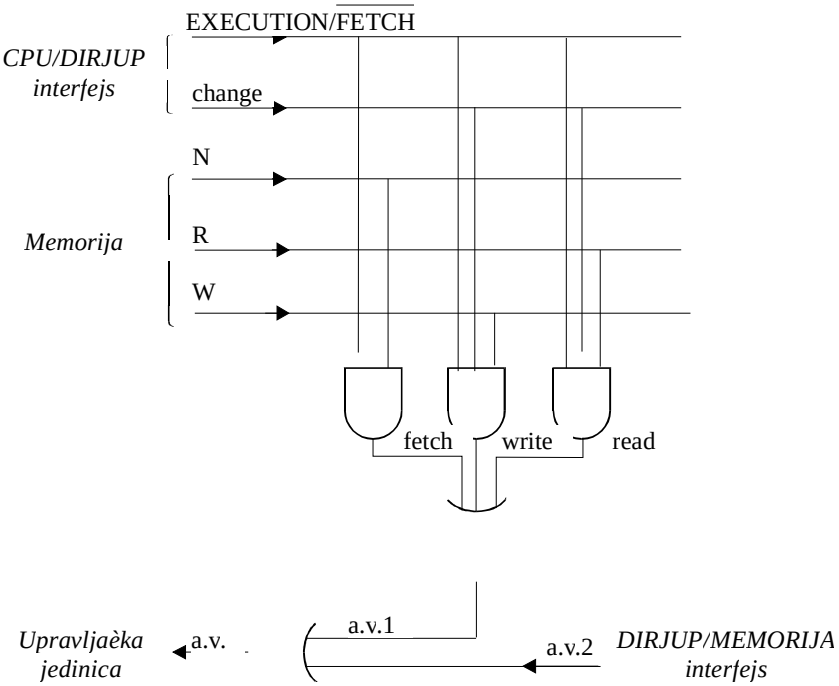
2.3.2.2.2.2. SABIRAČ ZA RAČUNANJE REALNE ADRESE

Sabirač za računanje realne adrese (slika 2.5.) mora biti što brži, jer je on zajedno sa memorijom usko grlo. On je realizovan kao **C**arry **L**ook **A**head sabirač. On radi paralelno sa određivanjem **hit** signala, pa ako je hit=1, onda će se njegov rezultat smatrati važećim. Na ulaz B sabirača dovodi se početna adresa segmenta, a na ulaz A adresa u okviru segmenta. Na izlazu je realna adresa.



Slika 2.5. Računanje realne adrese

2.3.2.2.2.3. KOLO ZA PROVERU PRAVA PRISTUPA



Slika 2.6. Provera prava pristupa

Ovo kolo (slika 2.6.) služi za proveru prava pristupa kada procesor generiše virtuelnu adresu. Da se ne bi registrovala povreda prava pristupa **DIRJUP** mora biti uvek u jednom od sledeća tri stanja:

Fetch: Procesor je u fazi fecovanja instrukcije i obratio se segmentu sa instrukcijama

Write: Procesor je u fazi egzekucije i obratio se segmentu sa zahtevom da piše u njega, a u segmentu je dozvoljeno upisivanje (čim je W bit deskriptora setovan na jedan to znači da je to segment sa podacima i da je dozvoljen upis).

Read: Procesor je u fazi egzekucije i obratio se segmentu sa zahtevom da čita iz njega, a u segmentu je dozvoljeno čitanje (čim je R bit deskriptora setovan na jedan to znači da je to segment sa podacima i da je dozvoljeno čitanje).

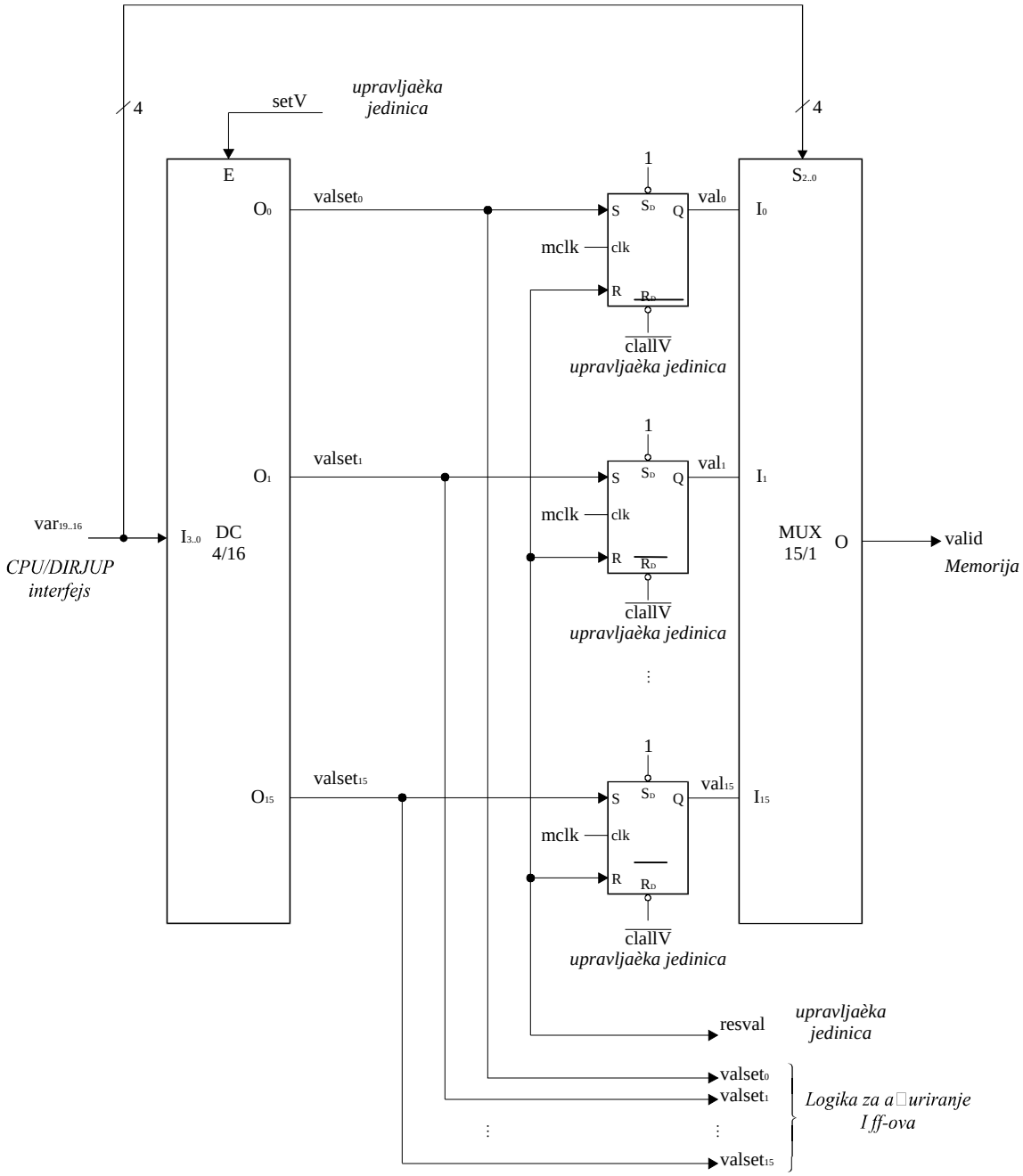
Kada **DIRJUP** nije ni u jednom od ova tri stanja registruje se nedozvoljen pristup. To će prouzrokovati generisanje prekida procesoru.

2.3.2.2.2.4. KOLO SA FLIP–FLOP-ovima V0..V15

Ovo kolo služi za pamćenje i određivanje validnosti ulaza memorije **DIRJUP**-a (slika 2.7).

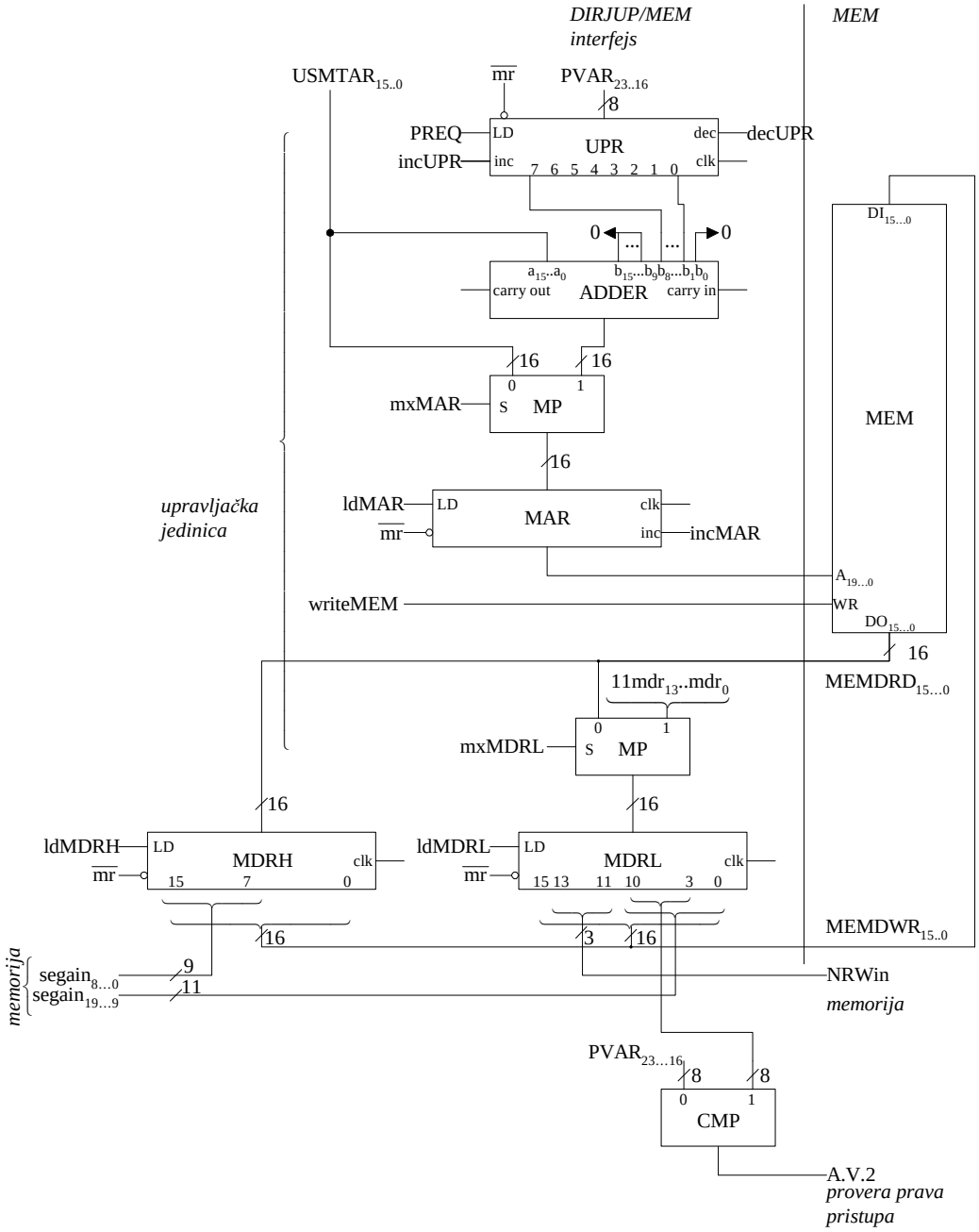
Kada je neki **V** flip–flop setovan to znači da je odgovatajući ulaz memorije **DIRJUP**-a validan tj. da sadrži deskriptor. Validnost određenog ulaza se proverava tako što se adresa ulaza **var19..16** dovede na **S** ulaze **MUX16/1**. Kao rezultat se dobija **valid** koji služi za određivanje **hit** signala.

Kada se dobije da je **miss** (**hit=0**), to znači da odgovarajući deskriptor nije u **memoriji DIRJUP**-a i da se mora ulaziti u **SMT** (koja je u operativnoj memoriji) da bi se proverilo da li je segment u operativnoj memoriji. Ako nije (**V-bit** deskriptora je nula), **DIRJUP** zadaje prekid **S**egment **F**ault koji za posledicu ima dovlačenje adresiranog segmenta sa diska. **Operativni sistem** kada dovuče segment, napravi deskriptor segmenta i stavi ga na njegovo mesto u **SMT**. Sada će **DIRJUP** (kada mu se procesor opet obrati sa istom virtuelnom adresom) kada ode u **SMT** imati **V-bit** deskriptora na jedinici te će moći da dovuče deskriptor (deo koji mu je potreban) u **memoriju DIRJUP-**a. Nakon toga će se setovati odgovarajući **V**-flip–flop na jedan, da bi se označilo da na tom ulazu **memorije DIRJUP**-a postoji validan deskriptor. To setovanje se vrši preko dekodera **DC4/16** zadavanjem upravljačkog signala **setV**.



Slika 2.7. V flip–flop-ovi

2.3.2.2.3. DIRJUP/MEMORIJA interfejs



Slika 2.8. DIRJUP/MEMORIJA interfejs

Blok DIRJUP/MEM interfejs sadrži registar MAR sa multiplekserom, registar MDRL sa multiplekserom, registar MDRH, komparator CMP8, sabirač i registar UPR (slika 2.8).

Registar UPR (*Unit Page Register*) služi za čuvanje broja segmenta na kojoj se nalazi virtuelna adresa čije se preslikavanje u realnu adresu trenutno izvršava. Ovaj registar se koristi kao pomoćni pri formiranju adrese ulaza utabelu segmenata. Broj segmenta se upisuje u registar UPR na signal takta CLK pri aktivnoj vednosti signala PREQ tj. pri iniciranju procesa

preslikavanja. Pošto je valičina jednog ulaza u tabeli dve reči, a u nultom ulazu se nalazi ukupan broj stranica korisnika na kojeg se tabela odnosi, adresa ulaza za određenu stranicu se formira na sledeći način: adresa_ulaza = USMTAR + 2*(UPR+1).

Ovo izračunavanje adrese se obavlja u sabiraču. Na A ulaz sabirača se dovodi sadržaj registra USMTAR (*Unit Segment Map Table Address Register*). Na B ulaz sabirača se dovodi inkrementirana vrednost sadržaja registra UPR, i to na pozicije $B_{10}…B_1$ bitovi $UPR_9…UPR_0$ čime se vrši množenje sa dva broja stranica uvećanog za jedan.

Registar MAR (*Memory Address Register*) služi za čuvanje ili adrese lokacije memorije MEM sa koje treba očitati podatak, koji može biti ili deskriptor segmenta ili niža reč nultog ulaza tabele segmenata, ili adrese lokacije memorije MEM u koju treba upisati podatak - deskriptor segmenta u koji je utisnut bit izmene. Adresa se upisuje u registar MAR pri pojavi signala takta **CLK** pri aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje može biti ili sadržaj registra USMTAR ili adresa ulaza u tabelu stranica formirana u sabiraču. Upravljačkim signalom **mxMAR** se selektuje kroz multiplekser jedna od te dve vrednosti. Kada je ovaj signal neaktivan, na ulazima 15…0 registar MAR prisutno je $USMTAR_{15...0}$.Kada treba propustiti vrednost $S_{15...0}$, signal **mxMAR** dobija aktivnu vrednost. Izlazi registra MAR se vode na adresne linije memorije MEM.

Registar MDRL (*Memory Data Register Low*) služi za čuvanje nize reci podatka koji je očitan iz memorije MEM i eventualno utiskivanje bita izmene, ili za čuvanje podatka koji treba upisati u memoriju MEM. U slučaju da se u registar upisuje sadržaj nulotg ulaza tabele segmenata ili deskriptor, segmenta iz tabele segmenata, na ulaze registra MDR se dovodi po linijama $\mathbf{DATA_{15...0}}$ 16-bitna vrednost sa izlaznih linija podataka memorije MEM. U slučaju da je potrebno utisnuti bit izmene u deskriptor segmenta koji je očitan iz memorije MEM i koji se nalazi u registru MDR, na ulaze registra MDR dovodi se podatak formiran od 14 najnižih bitova ovog registra i dve jedinice na dve najviše pozicije. Upravljačkim signalom mxMDR se selektuje kroz multiplekser koja će od ove dve vrednosti biti upisana u registar MDR. Sam upis će se obaviti na signal takta **CLK** ukoliko je aktivan signal **ldMDR**. Izlazi registra MDRWR se vodi na ulazne linije podataka memorije MEM. Izlazi registra MDR $MDR_{13...8}$ se vode i u DATA memoriju, a izlaz MDR_{14} u blok indikatora, radi upisa broja bloka i postavljanja odgovarajuće vrednosti I indikatora pri upisu dela deskriptora segmenta u DIRJUP jedinicu.

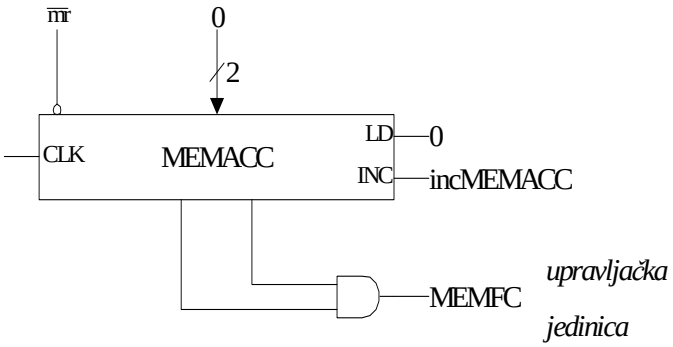
Registar MDRH (*Memory Data Register High)* sluzi za cuvanje vise reci podatka koji je ocitan iz memorije MEM ili za cuvanje podatka koji treba upisati u memoriju MEM.

Komparator CMP8 poredi broj segmenta iz virtuelne adrese i broj segmenata koji se nalazi u memoriji i utvrdjuje da li je adresa ispravna. Kao izlaz se dobija signal A.V.2. koji je 1 ukoliko je broj segmenta iz virtuelne adrese manji od broja segmenata u memoriji.

2.3.2.2.3.1. Brojac MEMACC

Brojač vremena pristupa operativnoj memoriji MEMACC se koristi kod čitanja ili upisa u memoriju MEM da odbroji četiri signala takta CLK. Usvojeno je da toliko iznosi vreme pristupa memoriji. Signal MEMFC (*MEMory Function Completed*) postaje aktivan kada brojač MEMACC pređe u stanje tri i koristi se kao signal logičkog uslova da je pristup memoriji završen. Generisanje i korišćenje ovog signala prikazano je na slici 2.9. Uzeto je da se korak $step_i$ koristi za pristup memoriji. Ulaskom u ovaj korak kreće se sa odbrojavanjem četiri signala takta CLK i ostaje u koraku $step_i$. Posle trećeg signala takta CLK signal MEMFC

postaje aktivan. Na četvrti signal takta CLK brojač MEMACC se vraća na stanje nula, signal MEMFC postaje neaktivan i prelazi se na korak $step_{i+1}$.



slika 2.9. BROJAC MEMACC

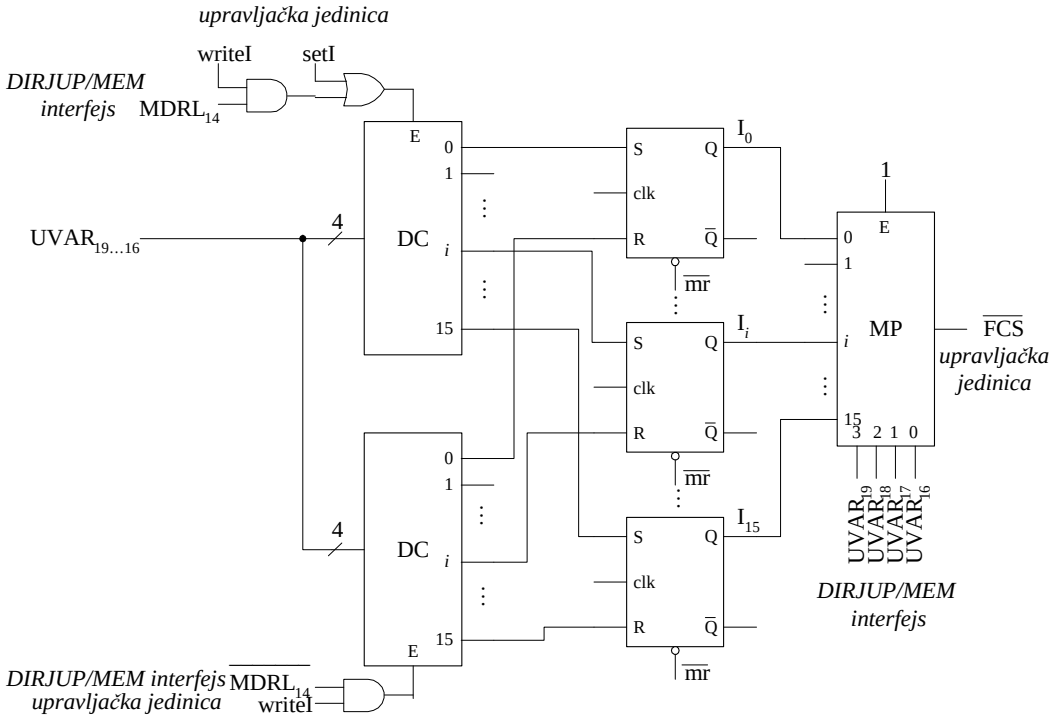
2.3.2.2.4. PAMCENJE IZMENE

Za pamcenje izmene koriste se flip-flopovi I0..I15. Ovi flip–flop-ovi su inicijalno postavljeni na nulu. Kada su setovani označavaju da je bilo upisa u segment, čiji se deskriptor (deo, jer ceo deskriptor je u **SMT**) nalazi u odgovarajućem ulazu **memorije DIRJUP-a**. Ova informacija je vrlo bitna jer kada **operativni sistem** treba da izbaci neki blok iz memorije, on će prvenstveno gledati da izbaci onaj blok koji nije menjan, jer ga nije potrebno vraćati na disk.

Kada se deskriptor dovlači u **memoriju DIRJUP-a** iz **SMT**, setuje se, ili resetuje odgovarajući **I** flip–flop (određenog ulaza za upis deskriptora) u zavisnosti od **I-bit-**a koji se nalazi u deskriptoru u **SMT**. Ako je **I-bit** jedan, znači da smo se obratili segmentu koja se još uvek nalazi u operativnoj memoriji i prethodno je menjan. Ako je **I-bit** nula, znači da smo se obratili segmentu koji prethodno nije menjan.

Ako se procesor obrati **DIRJUP**-u sa namerom da u neki segment nešto upiše, i ako imamo deskriptor tog segmenta u **memoriji DIRJUP-a**, postoje dva slučaja:

- odgovatajući **I** flip–flop je na nuli, što znači da se prvi put obraćamo tom segmentu sa zahtevom da u njega nešto upišemo. Tada se postavlja taj **I** flip–flop na jedan, i **DIRJUP** mora obaviti setovanje **I-bit**-a u odgovarajućem deskriptoru u **SMT**.
- odgovatajući **I** flip–flop je na jedinici, što znači da se obraćamo stranici u okviru koje je već upisivano, pa je samim tim i u **SMT** tabeli **I-bit** na jedinici te nema potrebe ga setovati.



Slika 2.10. I flip–flopo-vi

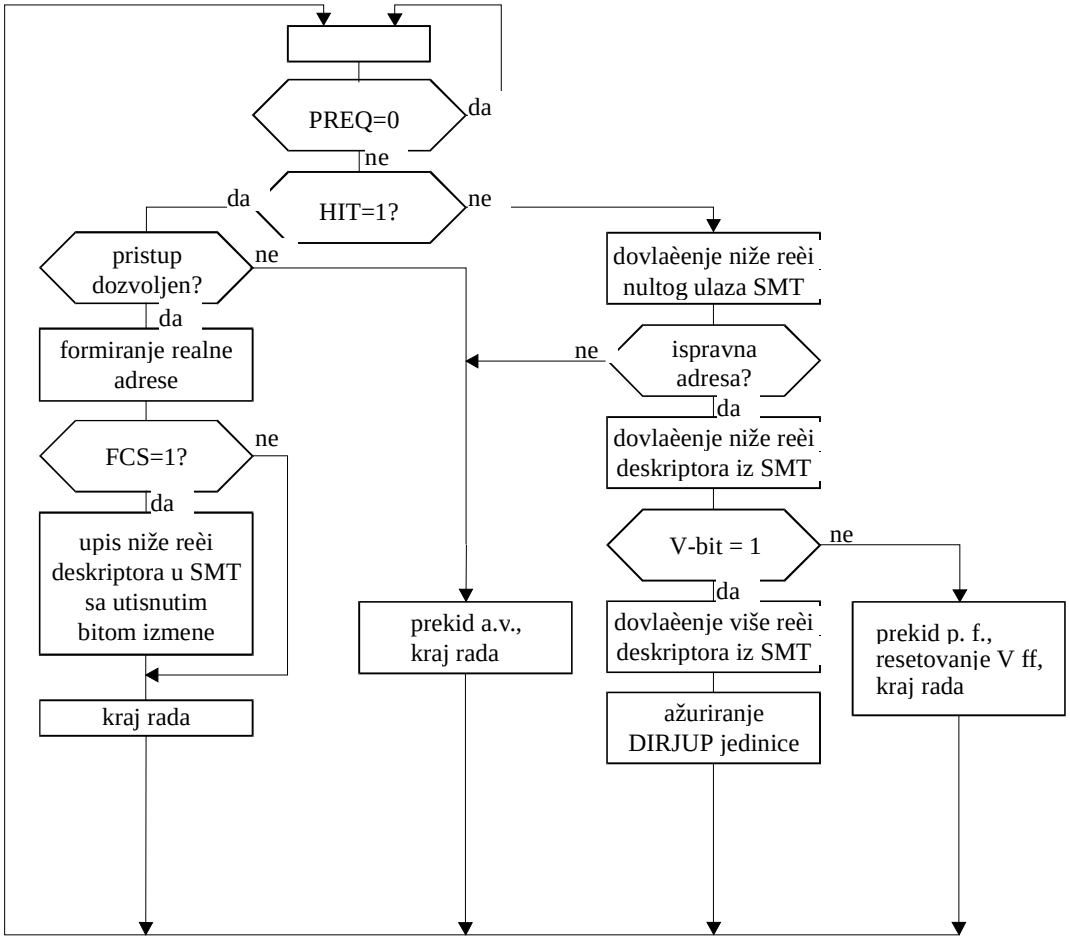
2.3.2.3. UPRAVLJAČKA DIRJUP JEDINICA

Upravljača jedinica **DIRJUP**-a služi za generisanje upravljačkih signala prema algoritmu direktnog preslikavanja virtuelne adrese u realnu. U ovom poglavlju će biti prikazani i objašnjeni:

- dijagram toka operacija
- vremenski oblici signala
- algoritam generisanja upravljačkih signala
- strukturna šema upravljačke jedinice

2.3.2.3.1. DIJAGRAM TOKA OPERACIJA

Dijagram toka operacija preslikavanja virtuelne adrese u realnu dat je na slici 2.11.



Slika 2.11. DIJAGRAM TOKA OPERACIJA

– **Da li je PREQ jednak jedan?**

  Na početku procesa preslikavanja proverava se da li je odgovarajuća virtuelna adresa upisana u registar **UVAR**. Ako jeste, onda je signal **PREQ** postavljen na jedan (procesor postavljanjem signala **PREQ** na 1startuje jedinicu za preslikavanje virtuelne adrese u realnu), i može da se predje na sledeću operaciju. Ukoliko je **PREQ** jednak nula, treba sačekati da se u registar **UVAR** upiše nova virtuelna adresa, i time **PREQ** postavi na jedan.

– **Da li saglasnost postoji (da li je hit=1)?**

  Ako postoji, to znači da je u određenom ulazu **memorije DIRJUP-a** otkrivena saglasnost, što znači da se u dijagramu toka ide na formiranje realne adrese. Ako saglasnost ne postoji, to se u dijagramu toka ide na dovlačenje niže reči nultog ulaza **SMT**-a.

– **Pristup dozvoljen?**

  Proverava se procesor obratio segmentu sa korektnim zahtevom.

– **Formiranje realne adrese**

  Obavlja se na sabiraču i prosledjuje **CPU/DIRJUP interfejsu**

– **Da li je FCS=1?**

Proverava se da li je procesor pristupio segmentu sa zahtevom da u njega nešto upiše i da li je to prvi upis u taj segment. Ako je to slučaj, ažurira se niža reč deskriptora u **SMT**-u (na **I**-bit deskriptora se utisne jedinica).

– **Dovlačenje niže reči nultog ulaza SMT-a**

Pošto je **hit=0**, dovlači se niža reč nultog ulaza **SMT**-a u **VMDRL** radi utvrđivanja tačnosti zadate adrese.

– **Adresa ispravna?**

Ako adresa nije ispravna generiše se prekid **a.v.** i završava sa radom. Ako je adresa ispravna, dovlači se deskriptor adresiranog segmenta iz **SMT**-a.

– **Da li je V-bit=1?**

Ako je **V**-bit **VMDRL**-a jedan, to znači da smo adresirali segment koja se nalazi u operativnoj memoriji te treba izvršiti ažuriranje **DIRJUP** jedinice, a ako nije, generiše se prekid **p.f.**, resetuju **V** flip–flop-ovi i to je kraj rada.

– **Ažuriranje DIRJUP jedinice**

Podrazumeva upisivanje dovučenog deskriptora u **memoriju DIRJUP-**a, postavljanjem odgovarajućeg V flip–flopa, kao i setovanje ili resetovanje odgovarajućeg **I** flip–flopa u zavisnosti od **I-bita VMDRL**-a. Upravljačka jedinica se restartuje jer sada ima sve potrebne podatke za preslikavanje zadate adrese u **memoriji DIRJUP-a**.

– **Kraj rada**

Postavljanjem signala **URP** na 1, jedinica signalizira procesoru da je adresa prevedena.

2.3.2.3.2. ALGORITAM GENERISANJA UPRAVLJACKIH SIGNALA

Na osnovu dijagrama toka operacija (slika 2.11.) i strukture operacione jedinice DIRJUP formiran je algoritam generisanja upravljačkih signala operacione jedinice. Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je sledeća:

<korak>: <bezuslovni_signal> <uslovni_signal> <sledeći_korak>

pri čemu su:

<bezuslovni_signal> := <signal>, {<signal>,} | <prazno>

<uslovni_signal> := *if* (<uslov>, <signal>, {signal,}) | <prazno>

< sledeći_korak > := *br* <korak> |
br (*if* <uslov>*then* <korak> *else* <korak>) |
br (*if* <uslov>*then* <korak>*else* (*if* <uslov> *then* <korak> *else* <korak>))

<korak> := simboličke oznake za korake od 0 do 13 step0 do step13

<signal> := svi signali operacione jedinice koje generiše upravljačka jedinica.

<uslov> := izrazi formirani od signala logičkih uslova koje generiše upravljačka jedinica

<prazno> :=

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

step0: *br* (*if* (**PREQ**) *then* step1 *else* step0)

! U koraku step0 se čeka da iz procesora stigne signal zahteva za preslikavanje virtuelne u realnu adresu **PREQ.** Ako se pojavi signal **PREQ** na signal takta **CLK** virtuelna adresa se upisuje u registar UVAR, broj segmenta u registar UPR, oznaka tekućeg korisnika se upisuje u registar UKOR, adresa početka tabele stranica tekućeg korisnika se upisuje u registar UPMTAR, tip operacije upisuje u flip-flop UR/WF i upisuje se podatak u flip-flop EXECUTION/FETCH. Ako se pojavi signal **PREQ** na signal takta **CLK** se prelazi na korak step1. U suprotnom slučaju se ostaje u koraku step0.

step1: *if* (**HIT**, **ldURAR**), **incUPR**

if(,**ldMAR**)

br (*if* **HIT** *then* step2 *else* step7)

! U korak step1 može da se dođe ili iz koraka step0 ili iz koraka step13. Iz koraka step0 se dolazi po svakom novom zahtevu za preslikavanje adrese. Iz koraka step13 se dolazi po nekom zahtevu za preslikavanje adrese za koji je pri prethodnom prolasku kroz step1 otkriveno da nema saglasnosti, pa se išlo na dovlačenje deskriptora segmenta iz operativne memorije u DIRJUP jedinicu. U koraku step1 se vrši provera saglasnosti i na osnovu toga formira vrednost signala **HIT**. Ako se pojavi aktivna vrednost signala **HIT**, to znači da je otkrivena saglasnost. U tom slučaju se generiše aktivna vrednost signala **ldURAR**, pa se na signal takta **CLK** realna adresa, koja se dobija kao pocetna adresa segmenta iz DIRJUP memorije+ adresa unutar segmenta iz registra UVAR, upisuje u registar URAR. Ako signal **HIT** ima neaktivnu vrednost tj. ako nema saglasnosti, generiše se aktivna vrednost signala **ldMAR**, kojom se, na signal takta **CLK** u registar MAR upisuje adresa nultog ulaza tabela segmenata. U koraku step1 takođe se generiše i aktivna vrednost signala **incUPR** čime se na signal takta **CLK** za jedan uveća broj segmenta koji se nalazi u registru UPR. Ovo se radi zbog kasnijeg određivanja tačne adrese odgovarajućeg ulaza u tabelu stranica.Ako se pojavi aktivna vrednost signala **HIT**, na signal takta **CLK** se prelazi na korak step2. U suprotnom slučaju se prelazi na korak step7.

step2: *if (***A.V.1., AV)**

br(if **A.V.1** *then step 13 else step 3)*

! U korak step2 moze da se dodje samo iz koraka step1. U koraku step2 se proverava da li je pristup dozvoljen. O dozvoli pristupa govori signal **A.V.1.** koga generise kombinaciona mreza sa slike 3.7. Ukoliko je ovaj signal aktivan, znaci da je doslo do greske **AV** i ide se na korak step 11. U suprotnom ide se na korak step3.

step3: *if*($\overline{\textbf{FCS}}$,**URP**,**clR/W,clE/F**),

if (**FCS**, **mxMAR**, **ldMAR**),

br (*if* **FCS** *then* step4 *else* step0)

! U korak step3 može da se dođe jedino iz koraka step2 i to samo onda kada je u koraku step2 otkriveno da je pristup dozvoljen. Ukoliko je aktivna vrednost signala

FCS to znači da se radi o operciji upisa, i to o prvoj takvoj operaciji na tekućoj stranici. Poterebno je iz operativne memorije učitati deskriptor segmenta i u njega utisnuti bit izmene. Generišu se aktivne vrednosti signala **mxMAR** i **ldMAR**. Aktivnom vednošću signala **mxMAR**, u bloku DIRJUP/MEM interfejs, kroz multiplekser se propušta adresa ulaza u tabelu segmenata formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** ta adresa se na signal takta **CLK** upisuje u registar MAR. U slučaju da je signal **FCS** na neaktivnoj vrednosti, generišu se aktivne vrednosti signala **UPR** i **clUR/WF.** Aktivnom vrednošću signala **UPR** DIRJUP jedinica daje procesoru indikaciju o kraju rada, a aktivnom vrednošću signala **clR/W** resetuje se flip-flop UR/WF koji ukazuje na tip operacije koja se izvodi. Na signal takta **CLK** se prelazi iz koraka step2 u korak step3 ako je signal **FCS** na aktivnoj vrednosti, odnosna u korak step0 ako je signal **FCS** na neaktivnoj vrednosti.

step4: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then* step5 *else* step4)

! U korak step4 se dolazi samo iz koraka step3. U koraku step4 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovm slučaju niža reč deskriptora tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step4 u korak step5. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step4.

step5: **mxMDR**, **ldMDRL**, **setI**,*br* step6

! U korak step5 se dolazi samo iz koraka step4. U registru MDR nalazi se učitan deskriptor segmenta u koji je potrebno utisnuti bit izmene. Bezuslovno se generišu aktivne vednosti signala **mxMDR**, **ldMDR** i **setI**. Aktivnom vrednošću signala **mxMDR** kroz multiplekser u bloku DIRJUP/MEM interfejs se propušta 16-bitna reč sastavljena od 14 nižih bitova registra MDR i dve jedinice na dve najviše pozicije.Aktivna vrednost signala **ldMDR** omogućuje da se na signal takta **CLK** ova vrednost upiše u registar MDR čime će najviša dva bita (V i I biti u deskriptoru) biti postavljeni na jedan. Aktivnom vednošću signala setI na signal takta **CLK** postaviće se na jedinicu I indikator vezan za ulaz AJUP jedinice dodeljen tekućoj stranici. Na signal takta **CLK** se uvek prelazi iz koraka step5 u korak step6.

step6: **writeMEM**, **incMEMACC**, *if* (**MEMFC**, **URP**, **clR/W, clE/F**), *br* (*if* **MEMFC** *then* step0 *else* step6)

! U korak step6 se dolazi samo iz koraka step5. U koraku step6 se bezuslovno generišu aktivne vrednosti signala **writeMEM** i **incMEMACC**. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDR upisuje u memoriju MEM na adresi određenoj sadržajem registra MAR. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta **CLK** inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji MEM. Aktivnom vrednošću signala **MEMFC** pri pojavi signala takta **CLK** završava se rad - generišu se aktivne vrednosti signala **URP** i **clR/W**. Aktivnom vrednošću signala URP daje indikacija o kraju rada, a aktivnom vrednošću signala **clR/W** resetuje se flip-flop UR/WF koji ukazuje na tip operacije koja se izvodi. Pri aktivnoj vrednosti signala **MEMFC** na signal takta **CLK** se prelazi iz koraka step0.Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step6.

step7: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then* step8 *else* step7)

! U korak step7 se dolazi iz koraka step1.Iz koraka step1 se dolazi kada se pri otkrivenoj nesaglasnosti odmah prelazi na dovlačenje deskriptora tekuće stranice iz operativne memorije u DIRJUP jedinicu. U koraku step7 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju niža reč nultog ulaza tabele segmenta tekućeg korisnika. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step7 u korak step8. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step7.

step8: *if* (**A.V.2**, **AV**),

if (~~**A.V.2**~~, **mxMAR**, **ldMAR**),

br (*if* **A.V.2** *then* step13 *else* step9)

! U korak step8 se dolazi samo iz koraka step7. U koraku step8 se ispituje ispravnost virtuelne adrese tj. da li je broj segmenta u dozvoljenom opsegu za tekućeg korisnika. Signal **A.V.2** predstavlja izlaz iz komparatora u kojem se porede broj segmenta i niža reč nultog ulaza tabele segmenataa gde je upisan ukupan broj segmenata korisnika. Ako je signal **A.V.2** na aktivnoj vrednosti to znači da je došlo do greške, pa se generiše aktivna vrednost signala **AV** kojom se postavlja odgovarajući indikator u procesoru. Ako je siganal **AV** na neaktivnoj vrednosti generišu se aktivne vrednosti signala **mxMAR** i **ldMAR**. Aktivnom vrednošću signala **mxMAR** se kroz odgovarajuće multiplekser u bloku DIRJUP/MEM interfejs propušta adresa ulaza u tabelu stranica formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** se obezbeđuje da se na signal takta **CLK** ta adresa upiše u registar MAR. Na signal takta **CLK** se prelazi iz koraka step8 u korak step9 ukoliko je adresa ispravna tj. signal **A.V.2** je na neaktivnoj vrednosti. U suprotnom prelazi se u korak step11.

step9: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then* step10 *else* step9)

! U korak step9 se dolazi samo iz koraka step8. U koraku step9 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovm slučaju niža reč deskriptora tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step9 u korak step10. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step9.

step10: *if* (**valid**, **incMAR)**,

if (☐, **SF, clallV**),

br (*if* **valid** *then* step11 *else* step13)

! U korak step10 se dolazi samo iz koraka step9. U koraku step10 se ispituje da li se tekući segment nalazi u operativnoj memoriji. Signal **valid** predstavlja vrednost V-bita deskriptora segmenta. Ako ima aktivnu vrednost, segment je u memoriji i generiše se aktivna vrednost signala **incMAR**. Aktivne vrednost signala **incMAR** omogućuje da se pri sledecek taktu **CLK**, procita visa rec deskriptora segmenta iz SMT. Na signal takta **CLK** prelazi se iz koraka step10 u korak step12 ako je

signal valid na aktivnoj vrednosti, odnosno u korak step11 ako je signal valid na neaktivnoj vrednosti.

step11: **incMEMACC,** *if***(MEMFC,ldMDRH)**

*br***(***if* **MEMFC** *then* step12 *else* step11**)**

! U korak step11 se dolazi samo iz koraka step10. U koraku step11 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovm slučaju visa reč deskriptora tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step11 u korak step0 pri cemu se prethodno azurira DIRJUP memorija. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step11.

step12: **ldMEM, decUPR, setV, writeI,** *br* step1

! U korak step12 se dolazi iz koraka step11. U koraku step12 se se bezuslovno generisu signal **ldMEM** cime se postize da se azurira sadrzaj DIRJUP memorije.

Zatim se zbog povratka u korak step1, gde se bezuslovno generise **incUPR,** generise signal **decUPR**, kako bi se izbeglo da se registar **UPR** inkrementira po drugi put.

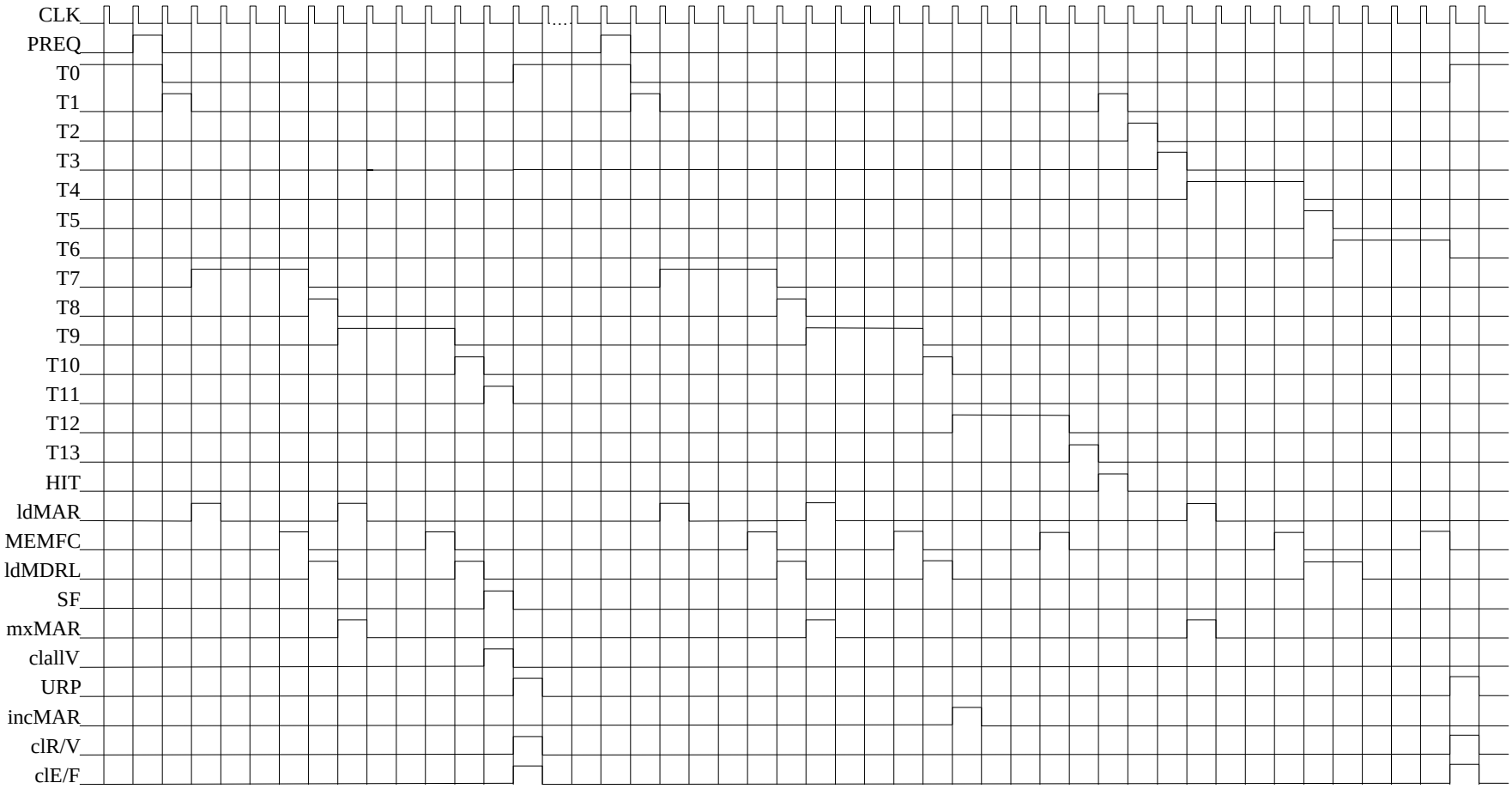
step13: **URP**, **clR/W**, **clE/F** *br* step0

! U korak step13 se dolazi ili iz koraka step8,step 10 ili iz koraka step2. U sva tri slučaja došlo je do neuspešnog okončavanja preslikavanja. Indikatori koji govore o tipu problema su postavljeni u ranijim koracima, pa se u koraku step10 samo daje indikacija o kraju rada postavljanjem na aktivnu vrednost signala **URP**, i resetuje UR/WF flip-flop koji ukazuje na tip operacije koja se izvodi, postavljanjem na aktivnu vrednost signala **clUR/WF**. Na signal takta **CLK** uvek se prelazi iz koraka step13 u korak step0.

2.3.2.3.3. VREMENSKI OBLICI SIGNALA

Vremenski oblici signala na osnovu kojih se dobijaju svi upravljački signali dati su na slici **Error! Reference source not found.**. Vremenski oblici signala su dati za primer kada se pri izvršavanju zahteva za preslikavanje otkrije da saglasnosti nema i da segment nije u memoriji,

a zatim se, nakon što operativni sistem dovuče segment u memoriju, generiše novi zahtev i preslikavanje uspešno završava.

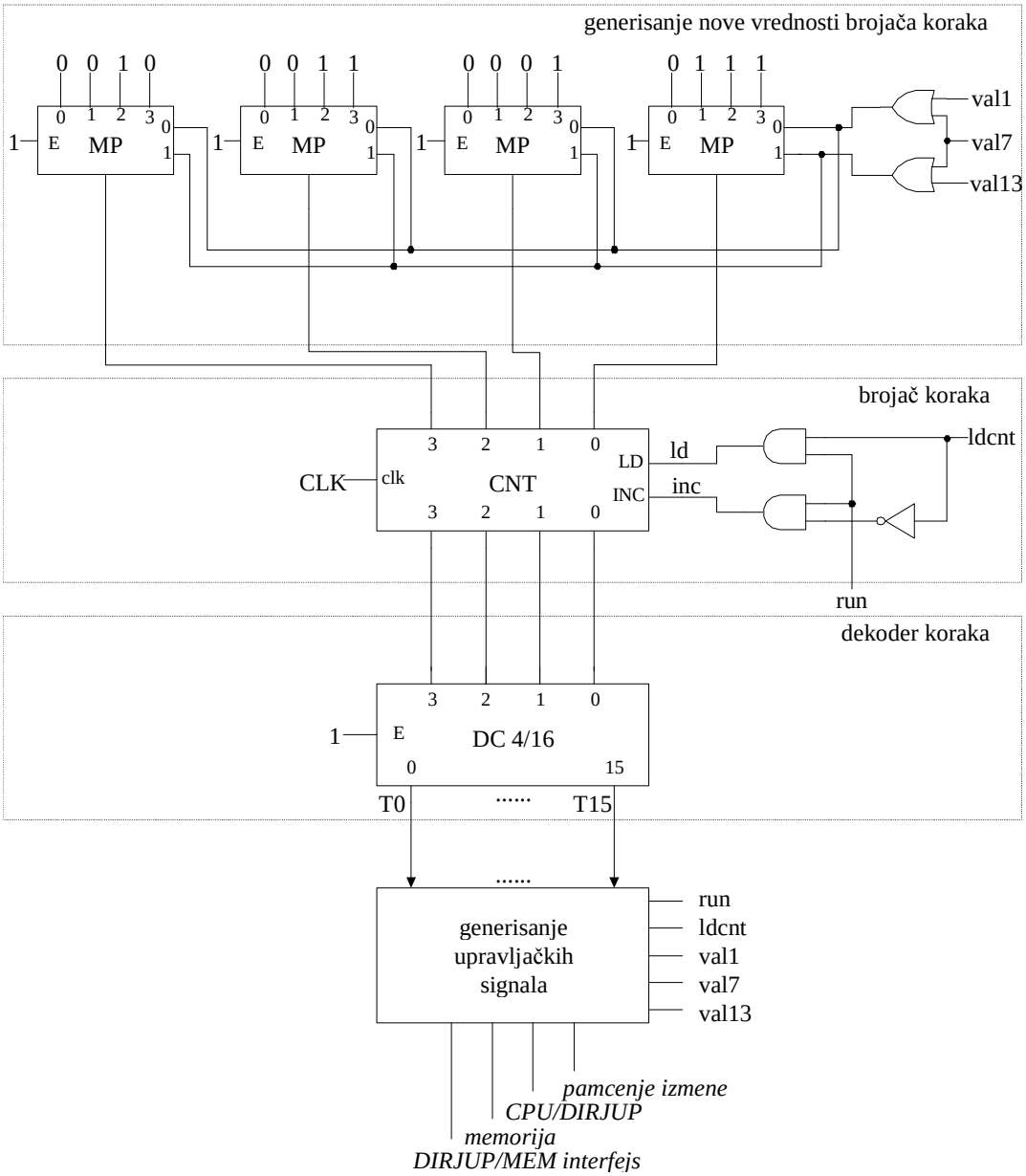

CLK
PREQ
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
T13
HIT
ldMAR
MEMFC
ldMDRL
SF
mxMAR
clallV
URP
incMAR
clR/V
clE/F

Slika 2.12. Vremenski oblici signala

2.3.2.3.4. Struktura upravljačke jedinice

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 2.13. Upravljačka jedinica se sastoji od sledećih blokova:

— generisanje nove vrednosti brojača koraka,
— brojač koraka,
— dekoder koraka i
— generisanje upravljačkih signala



Slika 2.13 Struktura upravljačke jedinice

2.3.2.3.4.1. Generisanje nove vrednosti brojaca koraka

Ovaj blok se sastoji od multipleksera i služi za generisanje i selekciju vrednosti koju treba upisati u brojač koraka. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Analizom algoritma generisanja upravljačkih signala operacione jedinice se utvrđuje da su 0, 1, 7 i 13 vrednosti koje treba upisati u brojač koraka da bi se realizovala odstupanja od sekvencijalnog izvršavanja mikrooperacija. Te vrednosti su ožičene na ulazima 0, 1, 2 i 3 multipleksera. Selekcija jedne od te četiri vrednosti se postiže odgovarajućim vrednostima signala $\mathbf{val_1}$, $\mathbf{val_7}$ i $\mathbf{val_{13}}$. Ako su sva tri signala neaktivna kroz multipleksere se propušta vrednost 0, a aktivnom vrednošću samo jednog od signala $\mathbf{val_1}$, $\mathbf{val_7}$ ili $\mathbf{val_{13}}$ kroz multiplekser se propuštaju vrednosti 1, 7 i 13, respektivno.

2.3.2.3.4.1.1. Brojač koraka

Brojač koraka CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač koraka može da radi u sledećim režimima:

— režim inkrementiranja,
— režim skoka i
— režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta **CLK** vrši se uvećavanje sadržaja brojača koraka za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz sekvence upravljačkih signala po koracima. Režim inkrementiranja je najčešći režim rada brojača koraka. Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **inc**. Signal **inc** je aktivan ako je signal **run** aktivan, i ako je signal **ldCNT** neaktivan. Signal **run** je uvek aktivan sem kada treba obezbediti režim mirovanja. Signal **ldCNT** je uvek neaktivan sem kada treba obezbediti režim skoka.

U režimu skoka pri pojavi signala takta **CLK** vrši se upis nove vrednosti u brojač koraka. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz sekvence upravljačkih signala po koracima. Režim skoka se javlja samo onda kada u brojač koraka treba upisati novu vrednost. Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ld**. Signal **ld** je aktivan ako su signali **run** i **ldCNT** aktivni.

U režimu mirovanja pri pojavi signala takta **CLK** ne menja se vrednost brojača koraka. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **inc** i **ld**. Ovi signali su neaktivni kada je signal **run** neaktivan. Signal **run** je uvek aktivan sem kada treba obezbediti režim mirovanja, što se dešava kada se čeka:

— pojava signala zahteva za pevođenje adrese **PREQ** ili
— pojava signala **MEMFC** kojim se označava da je čitanje ili upis u MEM obavljen.

2.3.2.3.4.1.2. Dekoder koraka

Dekodovana stanja brojača koraka pojavljuju se kao signali $\mathbf{T_0}$ do $\mathbf{T_{15}}$ na izlazima dekodera koraka. Svakom koraku iz algoritma generisanja upravljačkih signala dodeljen je jedan od

ovih signala i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$…, s tim da se signali $\mathbf{T_{14}}$ i $\mathbf{T_{15}}$ ne koriste.

2.3.2.3.4.1.3. Generisanje upravljačkih signala

Ovaj blok se sastoji od kombinacionih mreža koje pomoću signala $\mathbf{T}_0$ do $\mathbf{T_{13}}$ koji dolaze sa dekodera koraka, signala logičkih uslova koji dolaze iz procesora CPU i operacione jedinice DIRJUP jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje **Error! Reference source not found.**) generišu upravljačke signale. Ovaj blok generiše dve grupe upravljačkih signala i to upravljačke signale:

— operacione jedinice DIRJUP-a i
— upravljačke jedinice DIRJUP-a.

2.3.2.3.4.1.3.1. Upravljački signali operacione jedinice DIRJUP jedinice

Za svaki od ovih signala treba posmatrati u kojim koracima i pod kojim logičkim uslovima prema algoritmu generisanja upravljačkih signala dati signal treba da ima aktivnu vrednost. Dati signal se dobija kao unija proizvoda signala dekodovanih stanja brojača koraka i logičkih uslova pod kojim u datom koraku dati signal treba da bude aktivan. Na primer, signal **ldMAR** je aktivan u koraku $\mathbf{T_1}$ ako je neaktivan signal **HIT,** u koraku $\mathbf{T_3}$ ako je aktivan signal **FCS**, a u koraku $\mathbf{T_8}$ ako je neaktivan signal **A.V.2**, pa se stoga ovaj signal generiše prema relaciji **ldMAR** = $\mathbf{T_1 \cdot \overline{HIT}}$ + $\mathbf{T_3 \cdot}$ **FCS +** $\mathbf{\overline{T_8}}$ **A.V.2.** Ovi signali su dati posebno za procesor CPU, memoriju MEM i blokove operacione jedinice **DIRJUP** jedinice.

Upravljački signali operacione jedinice **DIRJUP** jedinice podeljeni su u grupe koje odgovaraju blokovima operacione jedinice i to:

— CPU/DIRJUP interfejs,
— memorija,
— pamcenje izmene i
— DIRJUP/MEM interfejs.

2.3.2.3.4.1.3.1.1. CPU/DIRJUP interfejs

U blok CPU/DIRJUP interfejs se šalju sledeći upravljački signali:

— ldURAR kojim se obezbeđuje da se na signal takta CLK u registar URAR upiše realna adresa formirana u DIRJUP jedinici,
— clR/W kojim se obezbeđuje da se na signal takta CLK upiše neaktivna vrednost u flip-flop R/W,
— clE/W kojim se obezbedjuje da se na signal takta CLK upise neaktivna vrednost u flip-flop E/W,
— AV kojim se, na signal takta CLK , upisuje aktivna vrednost u flip-flop PAVF, i time signalizira procesoru do je došlo do *Address Violation* greške,
— SF kojim se, na signal takta CLK , upisuje aktivna vrednost u flip-flop PSFF, i time signalizira procesoru do je došlo do *Segment Fault* greške,
— URP kojim se signalizira procesoru CPU da je operacija preslikavanja adrese završena, i u slučaju uspešnog preslikavanja upisuje realna adresa u registar PRAR

Ovi signali se generišu prema sledećim relacijama:

ldURAR = $\mathbf{T_1 \cdot HIT}$

clR/W = $\mathbf{T_3 \cdot \overline{FCS + T_6} \cdot MEMFC + T_{13}}$

AV = $\mathbf{T_8 \cdot A.V.2 + T_2 \cdot A.V.1}$

SF = $\mathbf{T_{10} \cdot}$ □

URP = $\mathbf{T_3 \cdot \overline{FCS + T_6} \cdot MEMFC + T_{13}}$

Kod njihovog generisanja kao signali logičkih uslova korišćeni su signali:

— HIT koji ukazuje da li postoji saglanost,
— FCS koji ukazuje da je u toku operacija prvog upisa na tekućem segmentu,
— A.V.1 koji ukazuje da je zabranjen pristup datom segmentu (*Address V iolation)*
— A.V.2 koji ukazuje da je broj stranice iz virtuelne adrese koja se preslikava veći od dozvoljenog (*Address Violation*)
— valid koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji i
— MEMFC koji ukazuje da su protekle četiri periode signala takta CLK koliko traje pristup memoriji MEM.

2.3.2.3.4.1.3.1.2. Memorija

U blok memorija se salju sledeci upravljacki signali:

—**setV** koji obezbedjuje da se na signal takta **CLK** upise aktivna vrednost u adresirani flip-flop V,

—**clallV** koji obezbedjuje da se na signal takta **CLK** upise neaktivna vrednost u sve V flip-flopove,

—**ldMEM** koji obezbedjuje da se na signal takta **CLK** u adresirani ulaz memorije **DIRJUP-a** .

Ovi signali se generisu prema sledecim relacijama:

setV = $\mathbf{T_{12}}$

clallV = $\mathbf{T_{10} \cdot}$ □

ldMEM = $\mathbf{T_{12}}$

Kod njihovog generisanja kao signali logičkih uslova korišćen je signal:

— valid koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji.

2.3.2.3.4.1.3.1.3. Pamcenje izmene

U blok pamcenje izmene se salju sledeci upravljacki signali:

— setI koji obezbeđuje da se na signal takta CLK upiše aktivna vrednost u adresirani flip-flop I i
— writeI koji obezbeđuje da se na signal takta CLK upiše vrednost u adresirani flip-flop I, i to vrednost koja se nalazi u deskriptoru segmenta u operativnoj memoriji (I-bit).

Ovi signali se generišu prema sledećim relacijama:

setI = T_5

writeI = T_{12}

2.3.2.3.4.1.3.1.4. DIRJUP/MEM interfejs

U blok **DIRJUP/MEM** interfejs se salju sledeci upravljacki signali:

— incMEMACC kojim se obezbeđuje da se na signal takta CLK inkrementira sadržaj brojača MEMACC.
— mxMAR čijom se aktivnom vrednošću kroz multiplekser propušta adresa ulaza u tabelu stranica formirana u sabiraču,
— mxMDR čijom se aktivnom vrednošću kroz multiplekser propušta modifikovan sadržaj registra MDR (utisnute jedinice na najviša dva mesta),
— ldMAR kojim se obezbeđuje da se na signal takta CLK upiše adresa u registar MAR,
— ldMDRL kojim se obezbeđuje da se na signal takta CLK upiše podatak u registar MDRL,
— ldMDRH kojim se obezbedjuje da se na signal takta CLK upise podatak u registar MDRH i
— writeMEM čijom se aktivnom vrednošću realizuje upis u memoriju MEM**.**

Ovi signali se generišu prema sledećim relacijama:

incMEMACC = $T_4 + T_6 + T_7 + T_9 + T_{11}$

mxMAR = T_3·FCS + T_8· A.$\overline{V.2}$

mxMDR = T_5

ldMAR = T_3 · FCS + T_8· A.$\overline{V.2}$

ldMDRL = T_4 · MEMFC + T_7 · MEMFC + T_9 · MEMFC + T_5

ldMDRH = T_{11} · MEMFC

writeMEM = T_6

Kod njihovog generisanja kao signali logičkih uslova korišćeni su signali:

— **FCS** koji ukazuje da je u toku operacija prvog upisa na tekućem segmentu,
— A.V.2 koji ukazuje da je broj stranice iz virtuelne adrese koja se preslikava veći od dozvoljenog (*Address Vioaltion*) i
— MEMFC koji ukazuje da su protekle četiri periode signala takta CLK koliko traje pristup memoriji MEM.

2.3.2.3.4.1.3.2. Upravljački signali upravljačke jedinice DIRJUP jedinice

Upravljački signali upravljačke jedinice su:

— **ldCNT** čijom se aktivnom vrednošću obezbeđuje upis vrednosti 0, 1, 7 ili 13 u brojač koraka CNT, a neaktivnom vrednošću obezbeđuje inkrementiranje tekuće vrednosti brojača koraka CNT,

— **run** čijom se aktivnom vrednošću omogućuje promena vrednosti brojača koraka CNT i to na način određen vrednošću signala **ldCNT**, a neaktivnom vrednošću onemogućava promena vrednosti brojača koraka CNT, bez obzira na vrednost signala **ldCNT** i
— **val_1**, **val_7** i **val_{13}**, koji obezbeđuju vrednosti 0, 1, 7 i 13 za upis u brojač koraka CNT.
Ovi signali se generišu prema sledećim relacijama:

ldCNT = $T_1 \cdot$ □ + $T_2 \cdot$ **A.V.1** + $T_3 \cdot \overline{FCS} + T_6 \cdot$ **MEMFC** + $T_8 \cdot$ **A.V.2** + $T_{10} \cdot \overline{valid} + T_{12} + T_{13}$

run = $T_0 \cdot$**PREQ** + $T_1 + T_2 + T_3 + T_4 \cdot$**MEMFC** + $T_5 + T_6 \cdot$**MEMFC** + $T_7 \cdot$**MEMFC** + $T_8 + T_9 \cdot$ **MEMFC** + $T_{10} + T_{11} \cdot$**MEMFC** + $T_{12} + T_{13}$

$val_1 = T_{12}$

$val_7 = T_1 \cdot$ □

$val_{13} = T_2 \cdot$ **A.V.1** + $T_8 \cdot$ **A.V.2** + $T_{10} \cdot \overline{valid}$

Kod njihovog generisanja kao signali logičkih uslova korišćeni su signali:

— □ **koji ukazuje da nema saglasnosti,**
— PREQ **kojim se startuj operacija preslikavanja adrese,**
— MEMFC **kojim se specificira završetak pristupa memoriji MEM**
— A.V.1 **koji ukazuje da nije dozvoljen pristup tekucem segmentu.(** ***Address Violation*** **) i**
— A.V.2 **koji ukazuje da je broj segmenta iz virtuelne adrese koja se preslikava veći od dozvoljenog (** ***Address Vioaltion*** **)**

2.3.3. JEDINICA SA ASOCIJATIVNIM PRESLIKAVANJEM

U okviru ovog poglavlja se najpre daju karakteristike razmatrane operativne memorije i jedinice za ubrzavanje preslikavanja. Zatim se daju detalji operacione i upravljačke jedinice razmatrane jedinice za ubrzavanje preslikavanja.

2.3.3.1. KARAKTERISTIKE OPERATIVNE MEMORIJE I JEDINICE ZA UBRZAVANJE PRESLIKAVANJA

Jedinica za ubrzavanje preslikavanja koja se razmatra realizovana je tehnikom asocijativnog preslikavanja za straničnu organizaciju virtuelne memorije.

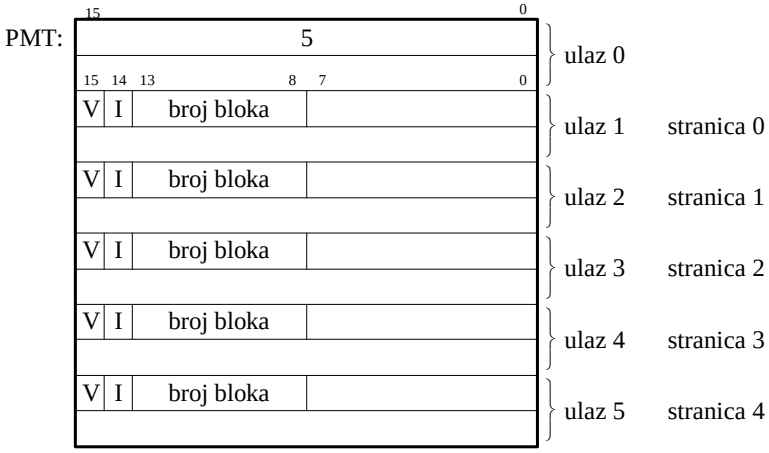
Adreasbilna jedinica je reč dužine dva bajta (1W=2B). Veličina virtuelnog adresnog prostora je 1M 16-bitnih reči, a veličina jedne stranice je 1KW. Shodno tome virtuelna adresa je dužine 20 bita, i ima sledeću strukturu:

- viših m = 10 bita ozančavaju broj stranice
- nižih k = 10 bita označavaju adresu unutar stranice

Veličina fizičkog adresnog prostora je 64KW, a veličina jednog bloka je 1KW.Realna adresa ima 16 bita i ima sledeću strukturu:

- viših m=6 bita označavaju broj stranice
- nižih k=10 bita označavaju adresu u okviru bloka.

Operativni sistem i asocijativna jedinica za ubrzavanje preslikavanja (u daljem tekstu AJUP jedinica), za realizaciju virtuelne memorije pristupaju zajedničkoj strukturi PMT (Page Map Table) u kojoj su informacije o



Slika 2.1 Izgled PMT za primer programa od pet stranica

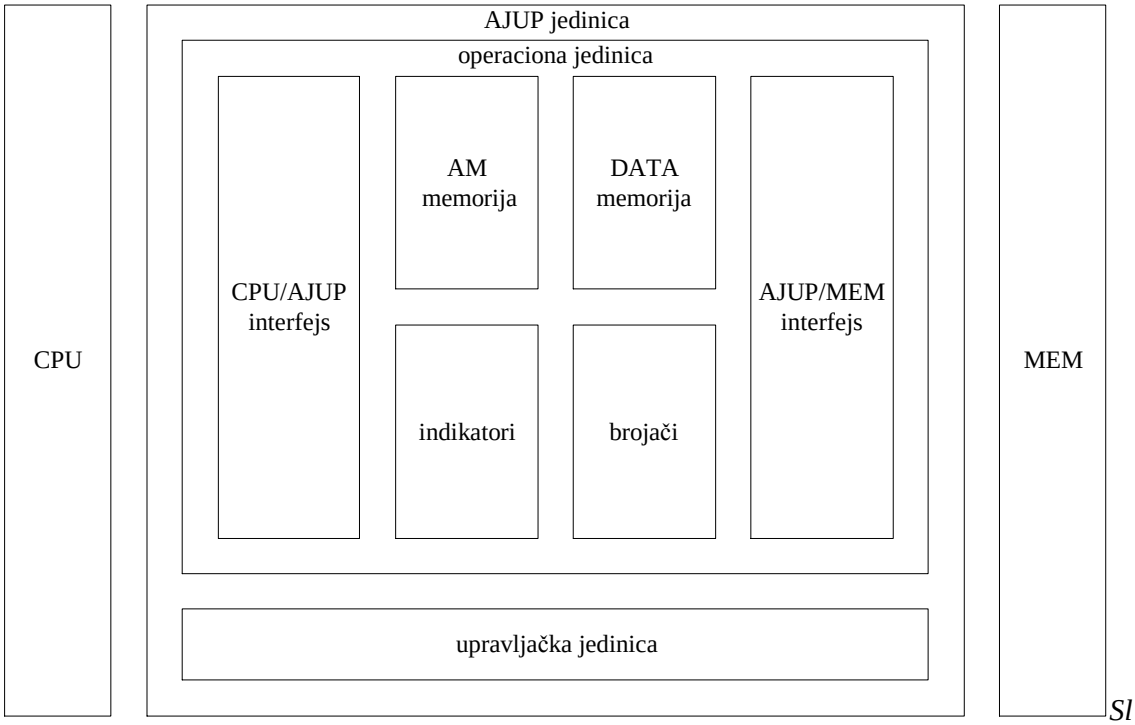
preslikavanju stranica u blokove. PMT se nalazi u operativnoj memoriji u onom delu odvojenom za operativni sistem. PMT se, u ovoj realizaciji, sastoji od ulaza od po dve reči. U svakom (nenultom) ulazu se čuva deskriptor odgovarajuće tsranice (koji je takođe dve reči). U nultom ulazu u nižoj reči, operativni sistem postavi broj koji označava koliko stranica ima program. Taj broj koristi AJUP jedinica za proveru prava pristupa. U prvom ulazu PMT se nalazi deskriptor nulte stranice, u drugom deskriptor prve …PMT je dugačka onoliko ulaza koliko ima stranica dati program i plus jedan za nulti ulaz.U nižoj reči nenultog ulaza se nalazi prva reč deskriptora, a u višoj druga. Petnaesti i šesnaesti bit prve reči deskriptora predstavljaju V-bit i I-bit respektivno. V-bit deskriptora odgovarajuće stranice (kada je na jedinici) označava da je operativni sistem dovukao tu stranicu sa diežska i smestio je u operativnu memoriju. I-bit deskriptora odgovarajuće stranice označava da je stranica menjana (vršen je bar jedna upis na nju), i da je operativni sistem pri izbacivanju iz operativne memorije mora vratiti na disk. Inicijalno kada se startuje program, operativni sistem firmira PMT sa potrebnim brojem ulaza, upiše broj stranica u nižu reč nultog ulaza i postavi sve V-bite na nula. Onda (pri startovanju otg programa) procesor zadaje virtuelne adrese, pa pri preslikavanju u AJUP jedinici, ako stranica nije dovučena u memoriju, AJUP jedinica zadaje prekid Page Fault koji kao rezultat ima da operativni sistem dovuče adresiranu stranicu u operativnu memoriju i u ulaz u PMT te stranice postavi odgovarajuće informacije deskriptora čime se i V-bit deskriptora setuje na jedan. Izgled PMT za primer programa od pet stranica prikazan je na slici 2.1.

Sistem koji se posmatra (slika2.2) se sastoji iz:

— procesora CPU,
— memorije MEM i
— jedinice za ubrzavanje preslikavanja AJUP.

Uzeto je da ceo sistem radi sinhrono sa zajedničkim signalom takta **CLK**.

U daljem tekstu će se najpre razmotriti samo delovi procesora CPU bitni za njegov rad sa AJUP jedincom, zatim će biti data uprošćena varijanta realizacije memorije MEM i na kraju će biti prikazana realizacija AJUP jedinice.



Slika 2.2 Struktura sistema

2.3.3.2. PROCESOR CPU

Procesor CPU se obraća AJUP onda kada treba očitati podatak iz operativne memorije ili upisati podatak u operativnu memoriju, pa je potrebno izvršiti peslikavanje virtuelne adrese u realnu adresu.

Procesor CPU šalje 20-bitnu virtuelnu adresu po linijama **PVAR$_{19...0}$**, oznaku tekućeg korisnika po linijama **PUSRR$_{2...0}$**, adresu početka tabele stranica tekućeg korisnika po linijama **PPMTAR$_{15...0}$**, podatak o vrsti operacije (upis ili čitanje) po liniji **R/W**, i generiše aktivnu vrednost signala **PREQ**.U slučaju uspešnog prevođenja adrese, generisana 16-bitna realna adresa se vraća po linijama **URAR$_{15...0}$**. Prevođenje je uspešno završeno i na linijama **URAR$_{15...0}$** je važeći podatak onda kada AJUP generiše aktivnu vrednost signala **URP**, a flip-lfopovi PAVF i PPFF su postavljeni na neaktivnu vrednost.

Procesor CPU u delu za povezivanje sa AJUP jedinicom koristi:

* registre PVAR, PUSRR, PPMTAR i PRAR za čuvanje virtuelne adrese i podataka o korisniku, odnosno za smeštanje generisane realne adrese u slučaju uspešno izvedenog preslikavanja,
* flip-flopove PRW, PAVF, PPFF za indikaciju o tipu operacije odosno o razlogu nemogućnosti uspešnog preslikavanja i

*upravljačke signale **PREQ**, **AV**, **PF** i **URP** za sinhronizaciju sa AJUP jedinicom

Registar PVAR (*Processor Virtual Address Register*) služi za čuvanje virtuelne adrese lokacije memorije MEM sa koje treba očitati podatak u slučaju operacije čitanja ili adrese lokacije memorije MEM u koju treba upisati podatak u slučaju operacije upisa. Izlazne linije ovog registra se vode kao 20 linija $\mathbf{PVAR_{19...0}}$ u CPU/AJUP interfejs asocijativne jedinice. Pretpostavlja se da je procesor CPU pre obraćanja keš memoriji već upisao virtuelnu adresu u registar PVAR, tako što je generisao aktivnu vrednost signala **ldPVAR** i na signal takta **CLK** izvršio upis sadržaja sa linija $\mathbf{PVARIN_{19...0}}$.

Registar PUSR (*Processor USer Register*) sadrži oznaku tekućeg korisnika procesora, a registar PPMTAR (*Processor Page Map Table Address Register*) pokazuje na početak tabele stranica tekućeg korisnika.Pretpostavlja se da je operativni sistem, pri dodeljivanju procesora korisniku, upisao potrebne podatke u ove registre, tako što je generisao aktivnu vrednost signala **ldPUSR** i na signal takta izvršio upis sadržaja sa linija $\mathbf{PUSR_{2...0}}$ u slučaju registra PUSR, odnosno generisao aktivnu vrednost signala **ldPPMTAR** i na signal takta izvršio upis sadržaja sa linija $\mathbf{PPMTAR_{15...0}}$ u slučaju registra PPMTAR.

Registar PRAR (*Processor Real Address Register*) služi za čuvanje realne adrese koja je generisana u slučaju uspešnog preslikavanja virtuelne u realnu adresu. Generisana adresa se iz CPU/AJUP interfejsa asocijativne jedinice vodi po linijama $\mathbf{URAR_{16...0}}$ na ulaze registra PRAR. Aktivna vrednost signala **URP** se koristi da se na signal takta **CLK** izvrši upis realne adrese sa linija $\mathbf{URAR_{16...0}}$ u registar PRAR, pod uslovom da su flip-flopovi PAVF i PPFF postavljeni na nulu.

Flip-flop PR/WF (*Processor Read/ Write Flag*) služi za čuvanje podatka o tipu operacije u kojoj se koristi zadata virtuelna adresa. Aktivnom vrednošću signala **R/W** (*Read/Write operation*), koji se dobija sa izlaza PR/WF flip-flopa, se specificira operacija čitanja, a neaktivnom operacija upisa. Pretpostavlja se da je procesor CPU pre toga generišući aktivnu vrednost ili signala **setR/W** ili signala **clR/W** trajanja jedne periode signala takta **CLK** postavio flip-flopPR/WF ili na aktivnu vrednost ili na neaktivnu vrednost, respektivno. Podatak o tipu operacije prosleđuje se AJUP jedinici na signal takta **CLK**, ako je aktivan signal **PREQ**.

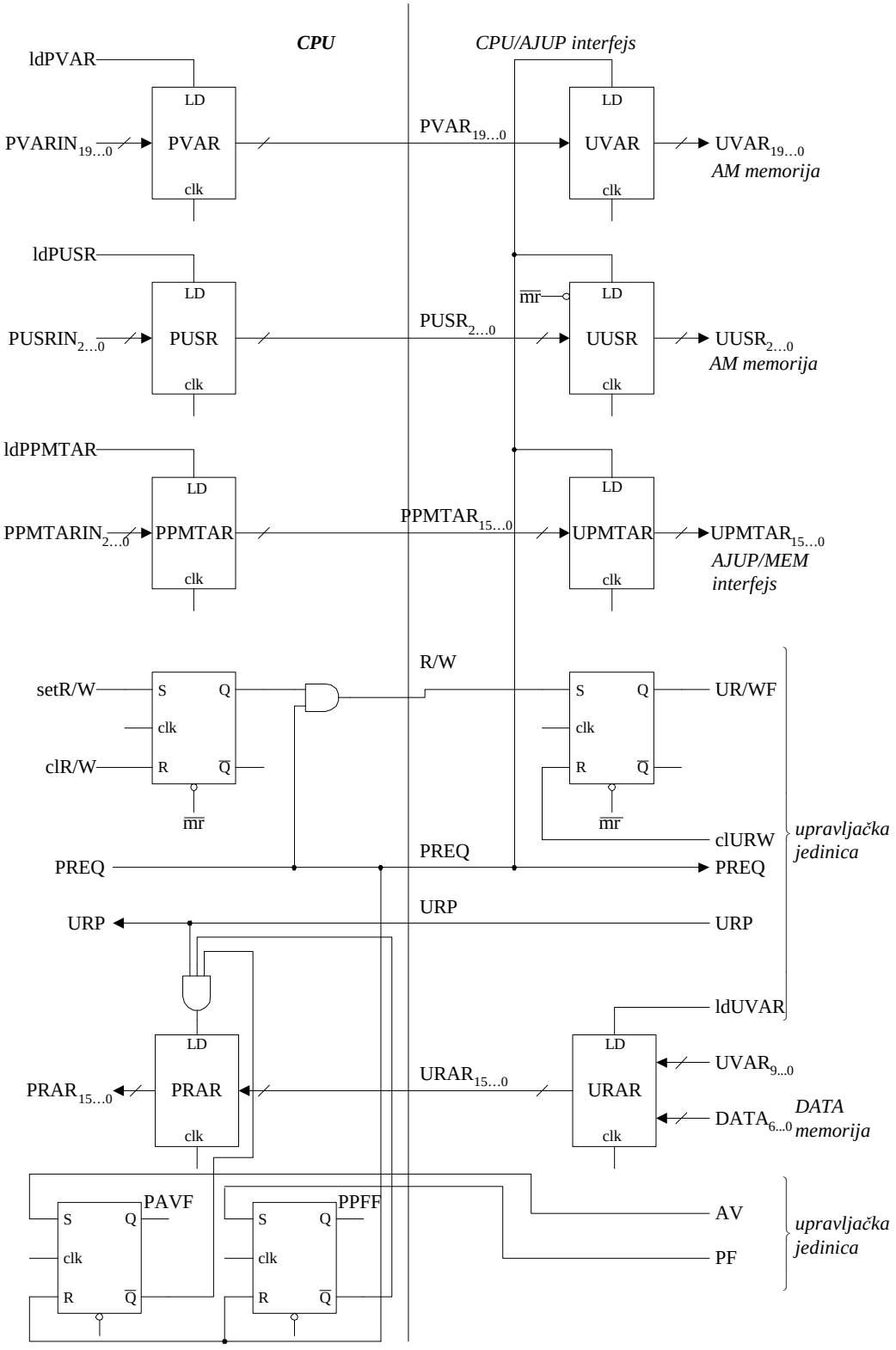
Flip-flop PAVF (*Procesor Address Violation Flag*) služi za indikaciju procesoru da je generisana virtuelna adresa neispravna tj. da je broj stranice na koju se adresa odnosi veći od ukupnog broja stranica tekućeg korisnika. Ovaj flip-flop postavlja se na aktivnu vrednost na signal takta **CLK** pri aktivnoj vrednosti signala **AV**. Signal **AV** generiše AJUP jedinica ako u procesu preslikavanja adrese otkrije da je došlo do Address Violation. Resetovanje PAVF se vrši pri postavljanju novog zahteva za preslikavanje tj. pri aktivnoj vrednosti signala **PREQ**.

Flip-flop PPFF (*Procesor Page Fault Flag*) služi za indikaciju procesoru da se stranica na koju se generisana virtuelna adresa odnosi ne nalazi u operativnoj memoriji. Procesor na osnovu aktivne vrednosti sadržaja PPFF generiše Page Fault prekid, a operativni sistem obrađujući ovaj prekid suspenduje tekućeg korisnika i organizuje prebacivanje stramice sa diska u operativnu memoriju. Ovaj flip-flop postavlja se na aktivnu vrednost na signal takta **CLK** pri aktivnoj vrednosti signala **PF**. Signal **PF** generiše AJUP jedinica ako u procesu preslikavanja adrese otkrije da je došlo do Page Fault-a. Resetovanje PPFF se vrši pri postavljanju novog zahteva za preslikavanje tj. pri aktivnoj vrednosti signala **PREQ**.

Upravljački signal **PREQ** (*Processor REQuest*) se koristi da procesor CPU aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira AJUP jedinici da se na linijama $\mathbf{PVAR_{19...0}}$, $\mathbf{PUSR_{2...0}}$, $\mathbf{PPMTAR_{15...0}}$ i **R/W** nalaze podaci potrebni za

operaciju preslikavanja virtuelne u realnu adresu, i da treba, na signal takta **CLK**, ove sadržaje upisati u odgavarajuće registre i time pokrenuti preslikavanje.

Upravljački signali **AV** (*Address Violation*) i **PF** (*Page Fault*) se koriste AJUP jedinica aktivnom vrednošću ovih signala trajanja jedne periode signala takta **CLK** signalizira procesoru CPU da se preslikavanje zadate adrese ne može uspešno izvršiti jer je došlo do prekoračenja dozvoljenog opsega adresa ili je došlo do straničnog



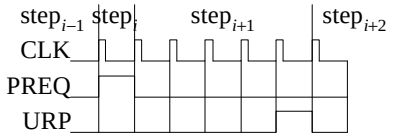
Slika 2.3 Procesor CPU i blok CPU/AJUP interfejs

prekida. U slučaju aktivne vrednosti signala **AV** na signal takta se upisuje aktivna vrednost u PAVF flip-flop, a u slučaju aktivne vrednosti signala **PF** aktivna vrednost se upisuje u PPFF flip-flop.

Upravljački signal **URP** (*Unit RePly*) se koristi da AJUP jedinica aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira procesoru CPU da je peslikavanje adrese završeno - ili uspešno ili neuspešno (što se vidi po vrednostima upisanim u PAVF i PPFF). U slučaju uspešnog peslikavanja iz AJUP

jedinice aktivnom vrednošću signala **URP** se signalizira i da se na linijama $\mathbf{URAR_{15...0}}$ nalazi generisana realna adresa i da aktivnom vrednošću signala **URP** treba na signal takta **CLK** ovu adresu upisati u registar PRAR.

Način razmene upravaljačkih signala između procesora CPU i AJUP jedinice na strani procesora CPU je prikazan na slici 2.4. Procesor CPU i operativni sistem u koracima pre koraka $step_i$ moraju generisati određene kombinacije signala **ldPVAR**, **ldPUSR**, **ldPPMTAR**,**setR/W** i **clR/W** da obezbede u relevantnim registrima za povezivanje sa AJUP jedinicom i u flip-flopu PR/WF vrednosti potrebne za preslikavanje adrese. Prelaskom u korak $step_i$, u kome ostaje samo jednu periodu signala takta **CLK**, procesor generiše aktivnu vrednost signala **PREQ** trajanja jedne periode takta i prelazi u korak $step_{i+1}$. U koraku $step_{i+1}$ procesor čeka pojavu aktivne vrednosti signala **URP** trajanja jedne periode signala takta **CLK**. Kada se javi aktivna vrednost signala **URP**, trajanja jedne periode signala trakta **CLK**, procesor prelazi u korak $step_{i+1}$ i produžava sa radom. Tom prilikom se u slučaju uspešno završenog preslikavanja na taj isti signal takta još i vrši upis generisane realne adrese u registar PRAR.



Slika 2.4 Razmena signala između procesora CPU i AJUP na strani procesora CPU

2.3.3.3. MEMORIJA MEM

Memorija MEM se obraća AJUP jedinici ili kod prebacivanja deskriptora stranice sa utisnutim bitom izmene iz AJUP jedinice u memoriju MEM kada treba realizovati operacije upisa u memoriju MEM ili kod prebacivanja deskriptora stranice ili niže reči nultog ulaza tabele stranica iz memorije MEM u AJUP jedinicu kada treba realizovati operacije čitanja iz memorije MEM (slika **Error! Reference source not found.**).

Kod operacije čitanja AJUP jedinica šalje 16-bitnu adresu po linijama $\mathbf{MAR_{15...0}}$ i drži neaktivnu vrednost upravljačkog signala **writeMEM**. Očitani 16-bitni podatak se vraća po linijama $\mathbf{MEMDRD_{15...0}}$. Kod operacije upisa AJUP jedinica šalje 16-bitnu adresu po linijama $\mathbf{MAR_{15...0}}$, 16-bitni podatak po linijama $\mathbf{MDRWR_{7...0}}$ i drži aktivnu vrednost upravljačkog signala **writeMEM**. Uzeto je da je vreme pristupa memorije MEM trajanja četiri periode signala takta **CLK**.

2.3.3.4. ASOCIJATIVNA JEDINICA ZA UBRZAVANJE PRESLIKAVANJA AJUP

Asocijativna jedinica za ubrzavanje preslikavanja AJUP (slika **Error! Reference source not found.**) se sastoji iz:

— operacione jedinice i
— upravljačke jedinice.

Operaciona jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje mikrooperacija i generisanje signala logičkih uslova. Upravljačka jedinica je kompozicija kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu asocijativnog preslikavanja virtuelne adrese u realnu. Sve sekvencijalne mreže unutar AJUP su sinhrone, a signal takta **CLK** je zajednički za procesor CPU i AJUP.

Funkcija i struktura operacione i upravljačke jedinice biće objašnjeni u daljem tekstu.

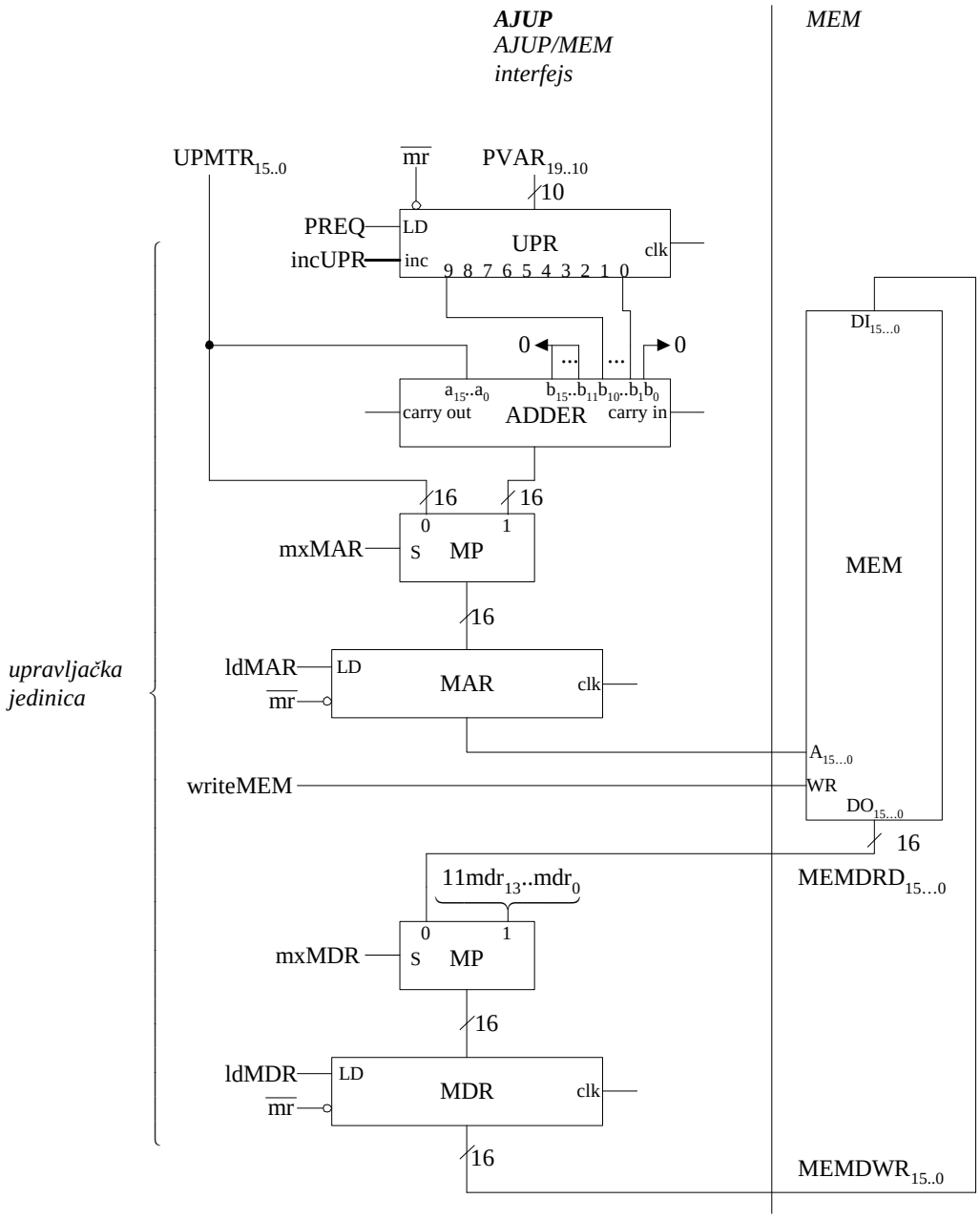
2.3.3.4.1. OPERACIONA JEDINICA AJUP JEDINICE

Operaciona jedinica se sastoji iz sledećih blokova:

— **CPU/AJUP interfejs**
— **indikatori**
— **brojači**
— **AM memorija**
— **DATA memorija**
— **AJUP/MEM interfejs**

Blok **CPU/AJUP interfejs** služi za razmenu adresa, podataka, upravljačkih signala i za sinhronizaciju rada sa procesorom CPU.

Blok **indikatori** služi za vođenje evidencije za svaki od 8 ulaza asocijativne jedinice o tome da li je ulaz zauzet i da li je sadržaj stranice čiji se deskriptor nalazi u ulazu modifikovan nekim prethodnim upisom.



Slika 2.5 Memorija MEM i blok AJUP/MEM interfejs

Blok **brojači** služi za odbrojavanje onoliko perioda signala takta **CLK** koliko je vreme pristupa memoriji MEM, i za određivanje broj ulaza asocijativne jedinice za upis novog deskriptora po FIFO (*First In First Out*) principu.

Blok **AM** služi za smeštanje 8 brojeva stranica.

Blok **DATA memorija** služi za smeštanje 8 brojeva blokova u koje se odgovarajuće stranice iz asocijativnog dela preslikavaju.

Blok **AJUP/MEM interfejs** služi za razmenu adresa, podataka i upravljačkih signala za sinhronizaciju rada sa operativnom memorijom MEM.

2.3.3.4.1.1. CPU/AJUP interfejs

Blok CPU/AJUP interfejs sadrži registre UVAR, URAR, UUSRR i UPMTAR i flip-flop UR/WF (slika 2.5).

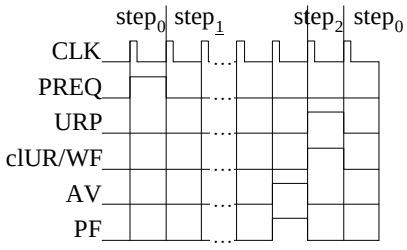
Registar UVAR (*Unit Virtual Address Register*) služi za čuvanje virtuelne adrese lokacije memorije MEM sa koje treba očitati podatak u slučaju operacije čitanja ili adresu lokacije memorije MEM u koju treba upisati podatak u slučaju operacije upisa. Na ulaze registra UVAR dovodi se 20-bitna adresa iz procesora CPU po linijama $\mathbf{PVAR_{19...0}}$. Ova adresa se upisuje u registar UVAR pri pojavi signala takta **CLK** ukoliko je aktivan signal **PREQ** koji dolazi od procesoraCPU. Razredi $\mathbf{UVAR_{19...10}}$ se vode u AM memoriju radi utvrđivanja saglasnosti, a kod upisivanja deskriptora stranice u AJUP jedinicu, i radi njihovog upisivanja u AM memoriju. Razredi $\mathbf{UVAR_{9...0}}$ se vode na ulaze 9…0 registra URAR.

Registar UUSR (*Unit USer Register*) sadrži oznaku tekućeg korisnika procesora, a registar UPMTAR (*Unit Page Map Table Address Register*) pokazuje na početak tabele stranica tekućeg korisnika. Na ulaze registra UUSR dovodi se sadržaj sa linija $\mathbf{PUSR_{2...0}}$, a na ulaze registra UPMTAR sadržaj sa linija $\mathbf{PPMTAR_{15...0}}$. Upis u ove registre vrši se na signal takta **CLK** pri aktivnoj vrednosti signala **PREQ**. Razredi $\mathbf{UUSR_{2...0}}$ se vode u AM memoriju radi utvrđivanja saglasnosti, a kod upisivanja deskriptora stranice u AJUP jedinicu, i radi njihovog upisivanja u AM memoriju. Razredi $\mathbf{UPMTAR_{15...0}}$ se vode u blok AJUP/MEM interfejs gde se koriste za formiranje adresa ulaza u tabelu stranica.

Registar URAR (*Unit Real Address Register*) služi za čuvanje realne adrese koja je generisana u slučaju uspešnog preslikavanja virtuelne u realnu adresu. Na ulaze registra $\mathbf{URAR_{15...10}}$ dovode se izlazne linije DATA memorije $\mathbf{DATA_{5...0}}$, a na ulaze $\mathbf{URAR_{9...0}}$ dovode se linije $\mathbf{UVAR_{9...0}}$. Upis se vrši na signal takta **CLK** pri aktivnoj vrednosti signala **ldURAR**. Generisana adresa se iz CPU/AJUP interfejsa asocijativne jedinice vodi po linijama $\mathbf{URAR_{15...0}}$ na ulaze registra PRAR.

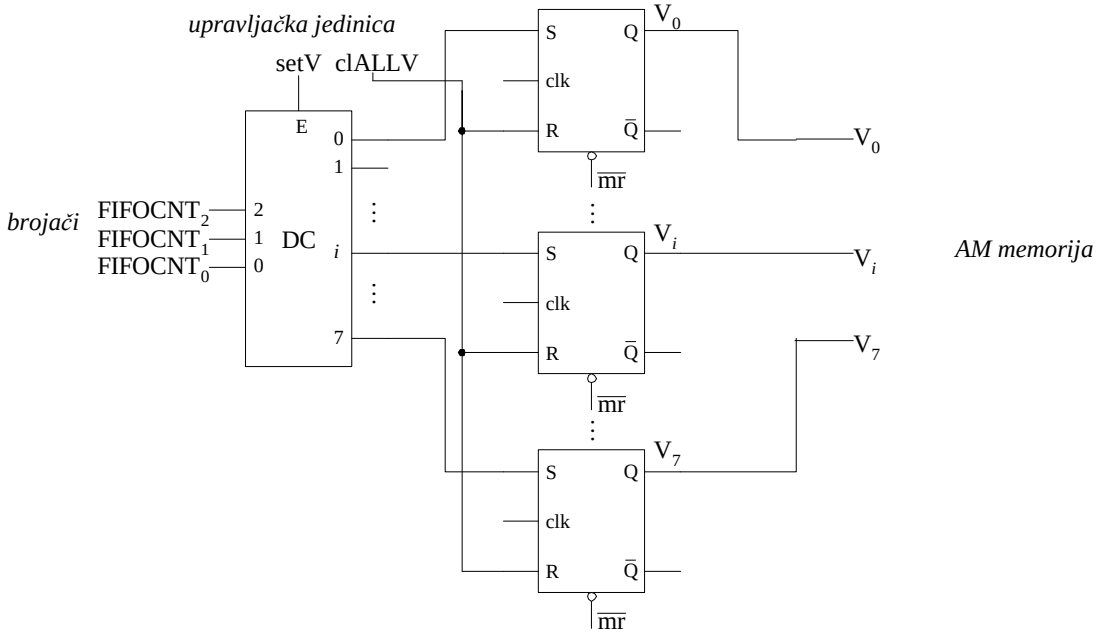
Upravljački signal **URP** (*Unit RePly*) se koristi da AJUP jedinica aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira procesoru CPU da je peslikavanje adrese završeno - ili uspešno ili neuspešno (što se vidi po vrednostima upisanim u PAVF i PPFF). U slučaju uspešnog peslikavanja iz AJUP jedinice aktivnom vrednošću signala **URP** se signalizira i da se na linijama $\mathbf{URAR_{15...0}}$ nalazi generisana realna adresa i da aktivnom vrednošću signala **URP** treba na signal takta **CLK** ovu adresu upisati u registar PRAR.

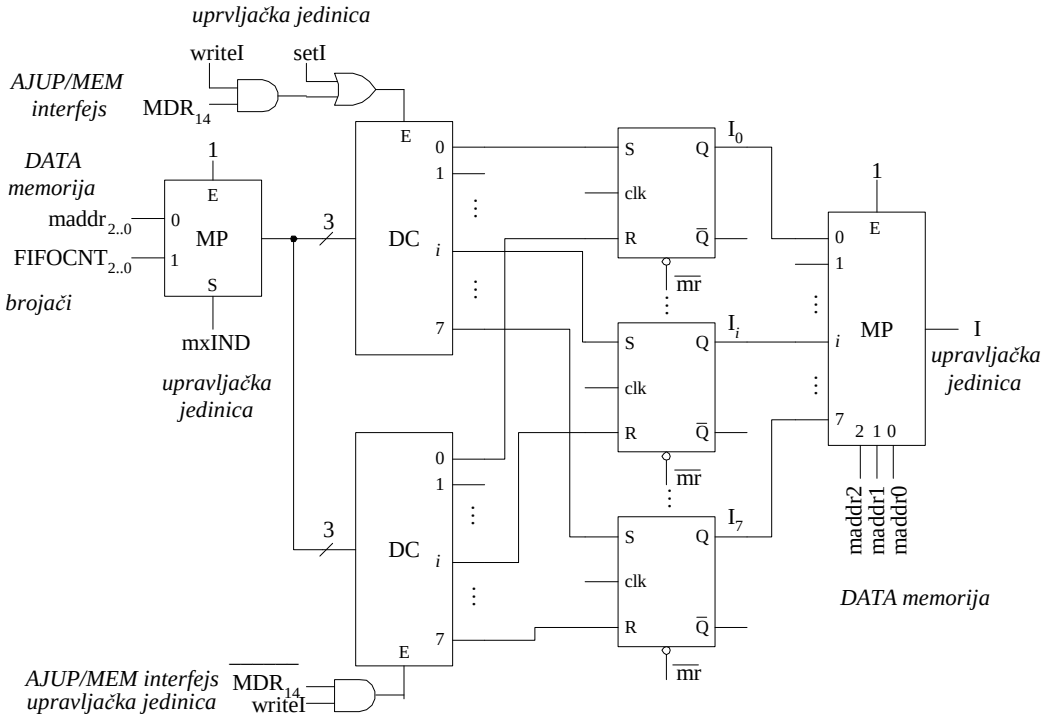
Flip-flop UR/WF (*Unit Read/ Write Flag*) služi za čuvanje podatka o tipu operacije u kojoj se koristi zadata virtuelna adresa. Aktivnom vrednošću signala **R/W** (*Read/Write operation*), koji se dobija sa izlaza PR/WF flip-flopa, se specificira operacija čitanja, a neaktivnom operacija upisa. Podatak o tipu operacije upisuje se u UR/WF na signal takta **CLK**, ako je aktivan signal **PREQ**. Ovaj flip-flop se postavlja na neaktivnu vrednost na aktivnu vrednost signala **clUR/WF**. Izlazni signal UR/WF vodi se u upravljačku jedinicu gde se koristi za generisanje signala **CH**.



Slika 2.6 Razmena signala između procesora CPU i AJUP jedinice na strani AJUP jedinice

Način razmene upravljačkih signala između procesora CPU i AJUP jedinice na strani AJUP jedinice je prikazan na slici 2.6. AJUP jeidinica u koraku $step_0$ čeka pojavu aktivne vrednosti signala **PREQ**, pa na signal





Slika 2.7 Indikatori

takta **CLK** prelazi na korak $step_1$. U slučaju saglasnosti, i pod uslovom da se ne radi o prvoj operaciji upisa na stranici, na prvi sledeći signal takta **CLK** se prelazi na korak $step_2$, u kome se ostaje samo jednu periodu signala takta **CLK**. U koraku $step_2$ se generišu aktivne vrednosti signala **URP** i **clUR/W** trajanja jedne periode signala takta **CLK**. Na sledeći signal takta **CLK** prelazi se u korak $step_0$. U slučaju da nema saglasnosti, ili se radi o prvoj operaciji upisa na stranici, iz koraka $step_1$ se prelazi na neki drugi korak da bi se AJUP jedinica po završenom preslikavanju ponovo vratila u korak $step_1$, a iz njega na već opisani način za slučaj saglasnosti, prešla u korak $step_2$ i, konačno, $step_0$. U slučaju da AJUP jedinica nije u stanju da uspešno završi preslikavanje, po povratku u stanje $step_1$ AJUP jedinica generiše generiše aktivnu vrednost signala **AV** ili **PF** trajanja jedne periode signala takta **CLK,** kojima procesoru daje indikaciju o razlogu neuspešno završenog preslikavaja.

2.3.3.4.1.2. Indikatori

Blok indikatori se sastoji od indikatora V_0 do V_7 i I_0 do I_7 (slika 2.7).

Indikatorima V_0 do V_7 se za svaki ulaz AJUP jedinice vodi evidencija da li je važeći ili ne. Ovi indikatori su realizovani sa 8 flip-flopova. Prilikom upisivanja deskriptora stranice iz operativne memorije u ulaz AJUP jedinice, adresiran sadržajem brojača FIFOCNT, generiše se aktivna vrednost signala **writeV**, pa se na signal takta **CLK** upisuje aktivna vrednost u indikator V koji odgovara datom ulazu. Prilikom provere saglasnosti vrednostima indikatora V_0 do V_7 uslovljavaju se signali mat_0 do mat_7 da bi se utvrdilo da li je sadržaj u ulazu važeći.

U slučaju da je došlo do Page Fault-a potrebno je sve ulaze AJUP jedinice proglasiti nevažećim. U tu svrhu generiše se aktivna vrednost signala **clallV**, pa se na signal takta upisuje neaktivna vrednost u sve V indikatore.

Indikatorima I_0 do I_7 se za svaki ulaz AJUP jedinice evidencija da li je bilo upisa u stranicu čiji je deskriptor u datom ulazu ili ne. Ova evidencija je potrebna operativnom sistemu pri izbacivanju stranice iz memorije. Ako je bilo upisa, tada pre dovlačenja nove stranice u operativnu memoriju i smeštanja njenog sadržaja u neki memorijski blok (naravno ako nema slobodnih blokova), najpre treba vratiti stranicu čiji je sadržaj bio u tom bloku na disk. Ako nije bilo upisa sadržaj nove stranice se odmah upisuje u izabrani blok. Pri upisu deskriptora stranice u neki od ulaza AJUP jedinice, I indikator vezan za taj ulaz postavlja se na vrednost I-bita iz deskriptora stranice. Ovo postavljanje obavlja se na sledeći način. Generiše se aktivna vrednost signala **mxIND** čime se na dekodere kojima se selektuje jedan od I indikatora propušta sadržaj brojača FIFOCNT koji služi za adresiranje ulaza AJUP jedinice pri upisu deskriptora. Generiše se aktivna vrednost signala **writeI** i na signal takta **CLK** se u adresirani flip-flop upiše vrednost I-bita iz deskriptora stranice koji je u tom trenutku u registru MDR (bit MDR_{14}).

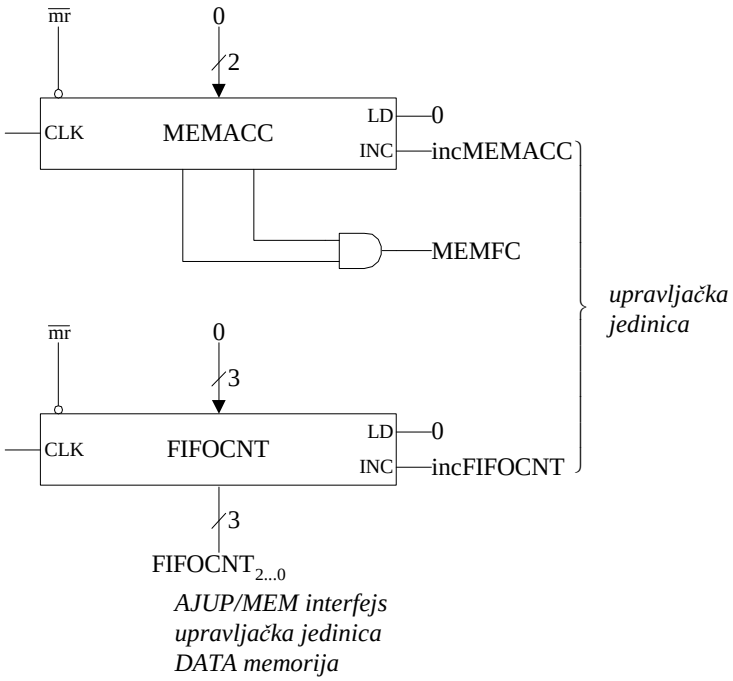
Kada se u AJUP jedinici, pri proveri saglasnosti, utvrdi da saglasnost postoji, na osnovu vrednosti I indikatora ulaza u kojem je saglasnost otkrivena i sadržaja flip-flopa URW koji ukazuje na tip operacije (čitanje/upis) formira se signal **CH**. Aktivna vrednost ovog signala predstavlja znak da se izvodi prva operacija upisa na tekućoj stranici. Tada je potrebno dovući deskriptor stranice iz operativne memorije i u njega utisnuti bit izmene tj. postaviti I-bit tog deskriptora na jedan. Takođe je potrebno I indikator ulaza AJUP jedinice u kojem je smešten deskriptor postaviti na jedan. To se postiže postavljanjem na aktivnu vrednost signala **setI** i upis se vrši na signal takta **CLK**. U ovoj situaciji, kada postoji saglasnost, selekcija željenog I indikatora vrši se 3-bitnom vrednošću $maddr_{2...0}$ iz bloka DATA memorija.

2.3.3.4.1.3. Brojači

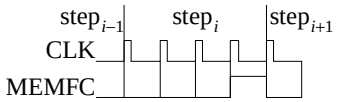
Modul brojači se sastoji od brojača vremena pristupa operativnoj memoriji MEMACC i brojača za generisanje adrese upisa u AJUP jedinicu FIFOCNT (slika2.8).

Brojač vremena pristupa operativnoj memoriji MEMACC se koristi kod čitanja ili upisa u memoriju MEM da odbroji četiri signala takta **CLK**. Usvojeno je da toliko iznosi vreme pristupa memoriji. Signal **MEMFC** (*MEMory Function Completed*) postaje aktivan kada brojač MEMACC pređe u stanje tri i koristi se kao signal logičkog uslova da je pristup memoriji završen. Generisanje i korišćenje ovog signala prikazano je na slici 2.9. Uzeto je da se korak $step_i$ koristi za pristup memoriji. Ulaskom u ovaj korak kreće se sa odbrojavanjem četiri signala takta **CLK** i ostaje u koraku $step_i$. Posle trećeg signala takta **CLK** signal **MEMFC** postaje aktivan. Na četvrti signal takta **CLK** brojač MEMACC se vraća na stanje nula, signal **MEMFC** postaje neaktivan i prelazi se na korak $step_{i+1}$.

Pošto se kod upisa deskriptora stranica u AJUP jedinicu kao algoritam zamene koristi FIFO (*First In First Out*) algoritam, za adresiranje ulaza u koji se vrši upis koristi se 3-bitni brojač FIFOCNT. Sadržaj ovog brojača $FIFOCNT_{2...0}$ vodi sa na adresne linije AM i DATA memorija, kao i u blok indikatora gde služi za selekciju. Inkremetiranje brojača FIFOCNT vrši se na signal takta **CLK** ako je aktivan signal **incFIFOCNT**.



Slika 2.8 Brojači



*Slika 2.9 Generisanje signala **MEMFC***

2.3.3.4.1.4. AM memorija

Blok AM memorija se sastoji od ascijativnog memorijskog modula AM i logike za generisanje signala saglasnosti HIT(slika 2.10).

Memorijski modul AM čuva brojeve 8 stranica čiji se deskriptori nalaze u AJUP jedinici, i oznake korisnika kojima te stranice pripadaju. Ovaj modul ima 8 lokacije širine 13 bita. Adresna reč modula DATA je širine 3 bita.

Memorijski modul DATA ima sledeće ulaze i izlaze:

* adresne linije **A** širine 3 bita,
* ulazne linije podataka **DI** širine 13 bita,
* izlazne linije podataka **DO** širine 13 bita,
* upravljački ulaz **WR** i
* izlazne linije $mat_0 \ldots mat_7$.

Upis u memorijski modul DATA se realizuje aktivnom vrednošću signala **writeDATA**, a čitanje njegovom neaktivnom vrednošću.

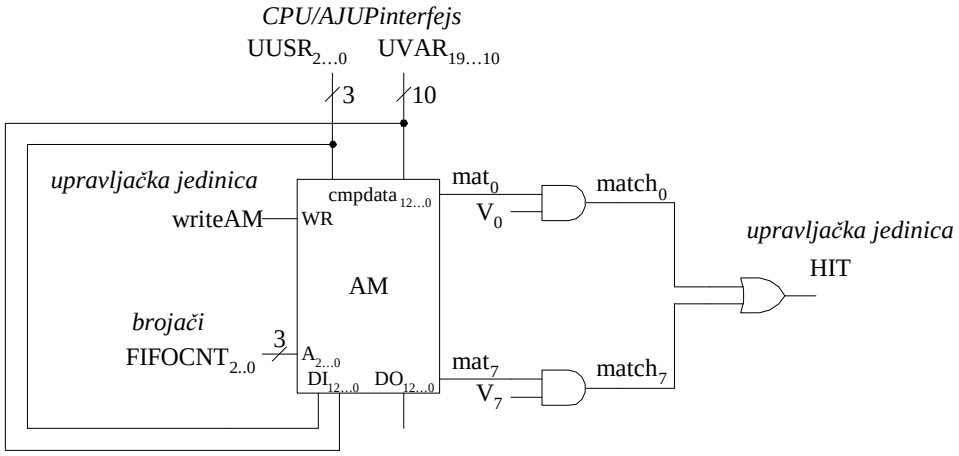
Nad asocijativnom memorijom AM se izvršavaju dve operacije:

1. Utvrđivanje saglasnosti sadržaja na ulazu cmpdata sa sadržajima svih osam ulaza asocijativne memorije

2. Upisivanje novog broja stranice i oznake korisnika u onaj ulaz asocijativne memorije koji je adresiran sa $A_{2...0}$.

Na ulaze AM memorije $cmpdata_{12...10}$ i $DI_{12...10}$ se dovodi oznaka korisnika iz registra UUSR ($UUSR_{2...0}$), a na ulaze $cmpdata_{9...0}$ i $DI_{9...0}$ broj stranice iz registra UVAR ($UVAR_{19...10}$). Obadva ova registra nalaze se u bloku CPU/AJUP iterfejs.

Utvrđivanje saglasnosti vrši se poređenjem 13-bitne vrednosti sa ulaza $cmpdata_{12...0}$ sa sadržajima svih osam ulaza asocijativne memorije. Ukoliko se u nekom ulazu postigne saglasnost, signal mat_i koji odgovara tom ulazu postaće jednak 1. Signali mat_i se na AND kolima uslovljavaju vrednošću V indikatora za dati ulaz koji pokazuje da li je sadržaj tog ulaza važeći. Najzad se od ovako dobijenih signala $match_i$ na OR kolu generiše signal saglasnosti HIT.



Slika 2.10AM memorija

Pri upisivanju novog broja stranice i oznake korisnika u asocijativnu memoriju, podatak sa ulaznih linija $DI_{12...0}$ upisuje se u lokaciju adresiranu sa $A_{2...0}$. Na ove linije dovodi se vrednost sadržaja brojača FIFOCNT iz bloka brojači, jer ovaj brojač služi za adresiranje ulaza kod upisa novog deskriptora stranice u AJUP jedinicu. Upis u am memoriju obavlja se na signal takta CLK pri aktivnoj vrednosti signala writeAM.

2.3.3.4.1.5. DATA memorija

Blok DATA memorija se sastoji od memorijskog modula DATA multipleksera i kodera (slika 2.11).

Memorijski modul DATA čuva brojeve 8 memorijskih blokova u kojima je smešten sadržaj 8 stranica čiji se deskriptori nalaze u AJUP jedinici. Ovaj modul ima 8 lokacije širine 6 bita. Adresna reč modula DATA je širine 3 bita.

Memorijski modul DATA ima sledeće ulaze i izlaze:

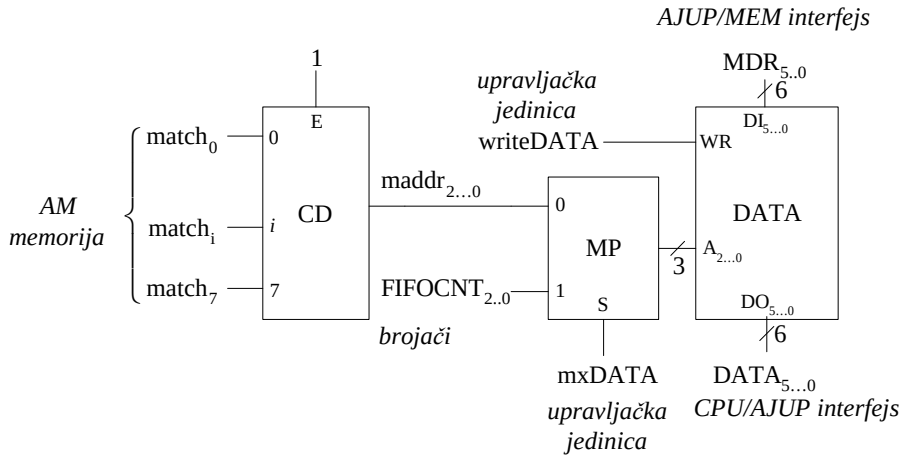
* adresne linije **A** širine 3 bita,

* ulazne linije podataka **DI** širine 6 bita,

* izlazne linije podataka **DO** širine 6 bita i

* upravljački ulaz **WR**.

Upis u memorijski modul DATA se realizuje aktivnom vrednošću signala **writeDATA**, a čitanje njegovom neaktivnom vrednošću.



Memorijskom modulu DATA pristupa se u sledećim slučajevima:

3. *Pri čitanju broja bloka kod formiranja realne adrese ako je otkrivena saglasnost*. U ovom slučaju adresa memorijskog modula DATA se formira od signala $match_0$…$match_7$ iz bloka AM memorija. Očitana vrednost sa izlaza **DO** memorijskog modula DATA vodi se u blok CPU/AJUP interfejs na ulaze 15…10 registra URAR.
4. *Pri upisu broja bloka kod upisivanja deskriptora stranice u AJUP jedinicu ako nema saglasnosti, ali se stranica nalazi u operativnoj memoriji*. Adresa lokacije u memorijskom modulu DATA u koju se upisuje broj bloka se formira od izlaza brojača FIFOCNT. Na ulaze **DI** memorijskog modula DATA propušta se sadržaj registra $MDR_{13...8}$ bloka AJUP/MEM interfejs.

Adresne linije **A** memorijskog modula DATA vezane su na sledeći način. Uslučaju 1 na adresne linije dovodi se 3-bitna adresa $maddr_{2...0}$ formirana u koderu od signala $match_0$…$match_7$ iz bloka AM memorija, a u slučaju 2 3-bitna adresa formirana od signala $FIFOCNT_{2...0}$ iz bloka brojači. Zato se na ulaze $A_{2...0}$ ove adrese vode preko multipleksera kojim se upravlja signalom **mxDATA**.

2.3.3.4.1.6. AJUP/MEM interfejs

Blok AJUP/MEM interfejs sadrži registar MAR sa multiplekserom, registar MDR sa multiplekserom, sabirač i registar UPR (slika 2.5).

Registar UPR (*Unit Page Register*) služi za čuvanje broja stranice na kojoj se nalazi virtuelna adresa čije se preslikavanje u realnu adresu trenutno izvršava. Ovaj registar se koristi kao pomoćni pri formiranju adrese ulaza utabelu stranica. Broj stranice se upisuje u registar

UPR na isgnal takta CLK pri aktivnoj vednosti signala PREQ tj. pri iniciranju procesa preslikavanja. Pošto je valičina jednog ulaza u tabeli dve reči, a u nultom ulazu se nalazi ukupan broj stranica korisnika na kojeg se tabela odnosi, adresa ulaza za određenu stranicu se formira na sledeći način: adresa_ulaza = UPMTAR + 2*(UPR+1).

Ovo izračunavanje adrese se obavlja u sabiraču. Na A ulaz sabirača se dovodi sadržaj registra UPMTAR (*Unit Page Map Table Address Register*). Na B ulaz sabirača se dovodi inkrementirana vrednost sadržaja registra UPR, i to na pozicije $B_{10}…B_1$ bitovi $UPR_9…UPR_0$ čime se vrši množenje sa dva broja stranica uvećanog za jedan.

Registar MAR (*Memory Address Register*) služi za čuvanje ili adrese lokacije memorije MEM sa koje treba očitati podatak, koji može biti ili deskriptor stranice ili niža reč nultog ulaza tabele tsranica, ili adrese lokacije memorije MEM u koju treba upisati podatak - deskriptor stranice u koji je utisnut bit izmene. Adresa se upisuje u registar MAR pri pojavi signala takta **CLK** pri aktivnoj vrednosti signala **ldMAR**. Adresa koja se upisuje može biti ili sadržaj registra UPMTAR ili adresa ulaza u tabelu stranica formirana u sabiraču. Upravljačkim signalom **mxMAR** se selektuje kroz multiplekser jedna od te dve vrednosti. Kada je ovaj signal neaktivan, na ulazima 15…0 registar MAR prisutno je $UPMTAR_{15...0}$.Kada treba propustiti vrednost $S_{15...0}$, signal **mxMAR** dobija aktivnu vrednost. Izlazi registra MAR se vode na adresne linije memorije MEM.

Registar MDR (*Memory Data Register*) služi za čuvanje podatka koji je očitan iz memorije MEM i eventualno utiskivanje bita izmene, ili za čuvanje podatka koji treba upisati u memoriju MEM. U slučaju da se u registar upisuje sadržaj nulotg ulaza tabele tsranica ili deskriptor, stranice iz tabele stranica, na ulaze registra MDR se dovodi po linijama $\mathbf{DATA_{15...0}}$ 16-bitna vrednost sa izlaznih linija podataka memorije MEM. U slučaju da je potrebno utisnuti bit izmene u deskriptor stranice koji je očitan iz memorije MEM i koji se nalazi u registru MDR, na ulaze registra MDR dovodi se podatak formiran od 14 najnižih bitova ovog registra i dve jedinice na dve najviše ppozicije. Upravljačkim signalom mxMDR se selektuje kroz multiplekser koja će od ove dve vrednosti biti upisana u registar MDR. Sam upis će se obaviti na signal takta **CLK** ukoliko je aktivan signal **ldMDR**. Izlazi registra MDRWR se vodi na ulazne linije podataka memorije MEM. Izlazi registra MDR $MDR_{13...8}$ se vode i u DATA memoriju, a izlaz MDR_{14} u blok indikatora, radi upisa broja bloka i postavljanja odgovarajuće vrednosti I indikatora pri upisu dela deskriptora stranice u AJUP jedinicu.

2.3.3.4.2. UPRAVLJAČKA JEDINICA AJUP

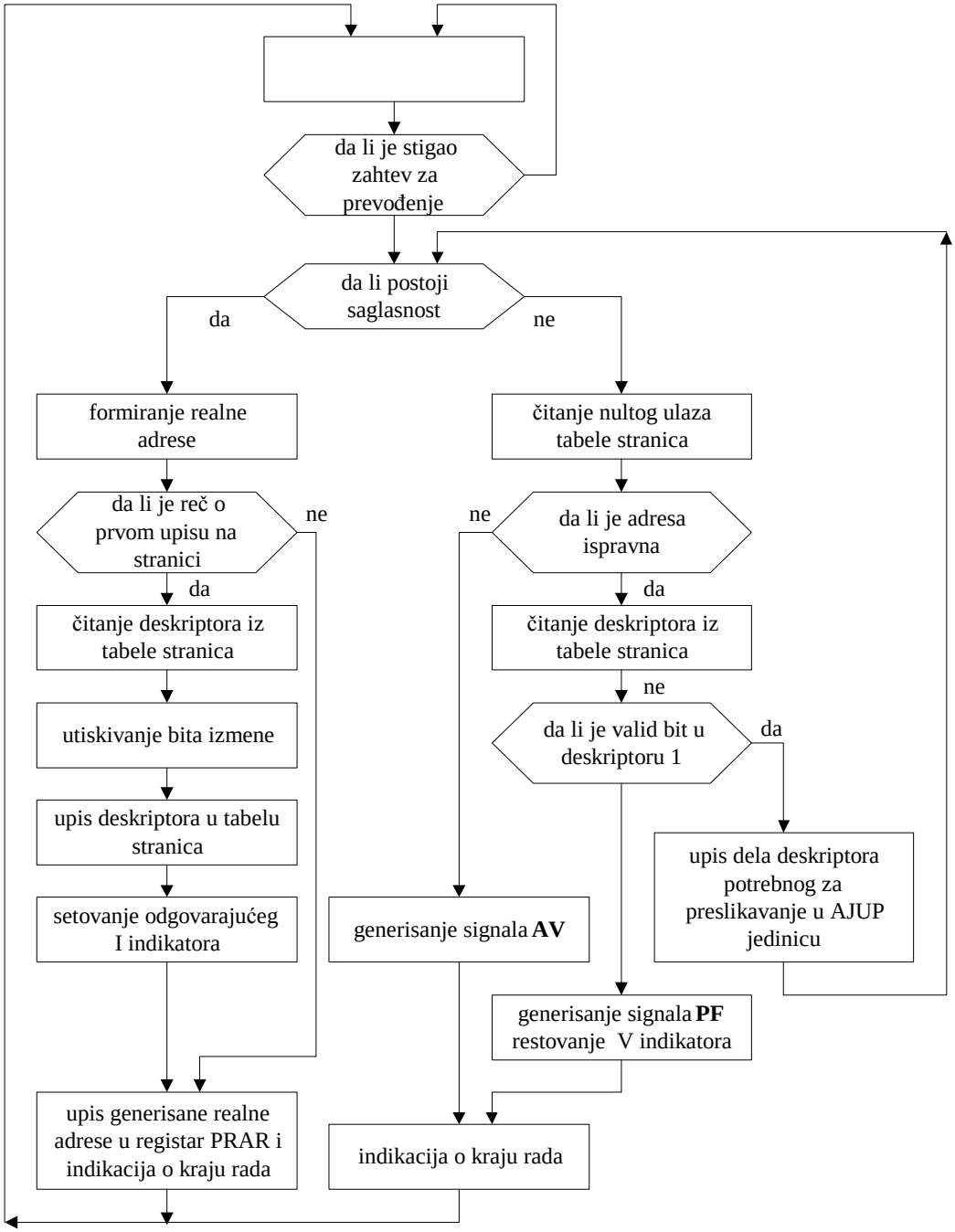
Upravljačka jedinica AJUP služi za generisanje upravljačkih signala prema algoritmu direktnog peslikavanja. U ovom poglavlju će biti prikazani dijagrami toka operacija, algoritam generisanja upravljačkih signala operacione jedinice, vremenski oblici signala i struktura upravljačke jedinice.

2.3.3.4.2.1. Dijagram toka operacija

Dijagram toka operacija je dat na slici 2.12.

Na početku upravljačka jedinica AJUP jedinice čeka pojavu signala zahteva za preslikavanje adrese **PREQ**. Pri pojavi ovog zahteva virtuelna adresa koju šalje procesor se upisuje u prihvatni registar virtuelne adrese AJUP jedinice, oznaka korisnika se upisuje u

registar UUSR, adresa početka tabele stranica se upisuje u registar UPMTAR, a oznaka operacije u flip-flop UR/WF.



Slika 2.12 Dijagram toka operacija

Sledeći korak je provera da li se deskriptor stranice na koju pokazuje virtuelna adresa nalazi u AJUP jedinici. Ova provera će se dalje nazivati provera da li postoji saglasnost. Ukoliko postoji saglasnost vrši se formiranje realne adrese. Zatim se vrši ispitivanje da li se radi o prvoj operaciji upisa na tekućoj stranici, i ako je odgovor da, iz tabele stranica se dovlači deskriptor i u njega utiskuje bit izmene. Deskriptor se po utiskivanju bita izmene vraća u memoriju i istovremeno se na aktivnu vrednost postavlja I indikator vezan za ulaz

AJUP jedinice u kojem je smešten deo deskriptora potreban za preslikavanje. Tada se prelazi na upisivanje generisane realne adrese u registar PRAR i indikaciju o kraju rada. Ako se na radi o prvoj operaciji upisa na stranici, na ovaj korak se dolazi odmah po generisanju realne adrese.rije u koji je izvršen upis u slučaju zahteva za upis.

Ukoliko ne postoji saglasnost prelazi se na dovlačenje niže reči nultog ulaza tabele stranica i poređenje ukupnog borja stranica koji je tu zapisan sa brojem stranice iz virtuelne adrese radi provere ispravnosti adrese.Ako je došlo do prekoračenja tj. Address Violation daje se indikacija o tome generisanjem aktivne vrednosti signala **AV** i završava se rad. Ako je adresa ispravna, iz tabele stranica se čita deskriptor stranice na koju ukazuje virtuelna adresa. Ispiruje se V-bit deskriptora čija aktivna vrednost označava da je stranica u memoriji. U slučaju da stranica nije u memoriji generiše se aktivna vrednost signala **PF** i završava rad. Ukoliko je stranica u memoriji, deo dskriptora koji je potreban za preslikavanje upisuje se u AJUP jedinicu i prelazi se na već opisani korak u kome se vrši provera da li se deskriptor stranice nalazi u AJUP jedinici. Pošto su potrebni podaci

dovučeni iz operativne memorije u AJUP jedinicu, sada će postojati saglasnost, pa će se dalji rad odvijati na već opisani način za slučaj saglasnosti.

Nakon indikacije kraja rada, AJUP jedinica prelazi u stanje u kojem čeka novi zahtev za preslikavanje adrese.

2.3.3.4.2.2. Algoritam generisanja upravljačkih signala operacione jedinice

Na osnovu dijagrama toka operacija (poglavlje **Error! Reference source not found.**) i strukture operacione jedinice AJUP (poglavlje **Error! Reference source not found.**) formiran je algoritam generisanja upravljačkih signala operacione jedinice. Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je sledeća:

<korak>: <bezuslovni_signal> <uslovni_signal> <sledeći_korak>

pri čemu su:

<bezuslovni_signal> := <signal>, {<signal>,} | <prazno>

<uslovni_signal> := *if* (<uslov>, <signal>, {signal,}) | <prazno>

< sledeći_korak > := *br* <korak> |
br (*if* <uslov>*then* <korak> *else* <korak>) |
br (*if* <uslov>*then* <korak>*else* (*if* <uslov> *then* <korak> *else* <korak>))

<korak> := simboličke oznake za korake od 0 do 7 step0 do step7

<signal> := svi signali operacione jedinice koje generiše upravljačka jedinica.

<uslov> := izrazi formirani od signala logičkih uslova koje generiše upravljačka jedinica

<prazno> :=

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

step0: *br* (*if* (**PREQ**) *then* step1 *else* step0)

! U koraku step0 se čeka da iz procesora stigne signal zahteva za preslikavanje virtuelne u realnu adresu **PREQ.** Ako se pojavi signal **PREQ** na signal takta **CLK** virtuelna adresa se upisuje u registar UVAR, broj stranice u registar UPR, oznaka tekućeg korisnika se upisuje u registar UKOR, adresa početka tabele stranica tekućeg korisnika se upisuje u registar UPMTAR i tip operacije upisuje u flip-flop UR/WF. Ako se pojavi signal **PREQ** na signal takta **CLK** se prelazi na korak step1. U suprotnom slučaju se ostaje u koraku step0.

step1: *if* (**HIT**, **ldURAR**), **incUPR**

if (, **ldMAR**)

br (*if* **HIT** *then* step2 *else* step6)

! U korak step1 može da se dođe ili iz koraka step0 ili iz koraka step10. Iz koraka step0 se dolazi po svakom novom zahtevu za preslikavanje adrese. Iz koraka step10 se dolazi po nekom zahtevu za preslikavanje adrese za koji je pri prethodnom prolasku kroz step1 otkriveno da nema saglasnosti, pa se išlo na dovlačenje deskriptora stranice iz operativne memorije u AJUP jedinicu. U koraku step1 se vrši provera saglasnosti i na osnovu toga formira vrednost signala **HIT**. Ako se pojavi aktivna vrednost signala **HIT**, to znači da je otkrivena saglasnost. U tom slučaju se generiše aktivna vrednost signala **ldURAR**, pa se na signal takta **CLK** realna adresa, formirana od broja bloka pročitanog iz DATA memorije i adrese u okviru bloka iz virtuelne adrese, upisuje u registar URAR. Ako signal **HIT** ima neaktivnu vrednost tj. ako nema saglasnosti, generiše se aktivna vrednost signala **ldMAR**, kojom se, na signal takta **CLK** u registar MAR upisuje adresa nultog ulaza tabela stranica. U koraku step1 takođe se generiše i aktivna vrednost signala **incUPR** čime se na signal takta **CLK** za jedan uveća broj stranice koji se nalazi u registru UPR. Ovo se radi zbog kasnijeg određivanja tačne adrese odgovarajućeg ulaza u tabelu stranica.Ako se pojavi aktivna vrednost signala **HIT**, na signal takta **CLK** se prelazi na korak step2. U suprotnom slučaju se prelazi na korak step6.

step2: *if* (, **UPR**, **clUR/WF**),
if (**CH**, **mxMAR**, **ldMAR**),

br (*if* **CH** *then* step3 *else* step0)

! U korak step2 može da se dođe jedino iz koraka step1 i to samo onda kada je u koraku step1 otkrivena saglasnost. Ukoliko je aktivna vrednost signala **CH** to znači da se radi o operciji upisa, i to o prvoj takvoj operaciji na tekućoj stranici. Poterebno je iz operativne memorije učitati deskriptor stranice i u njega utisnuti bit izmene. Generišu se aktivne vrednosti signala **mxMAR** i **ldMAR**. Aktivnom vednošću signala **mxMAR**, u bloku AJUP/MEM interfejs, kroz multiplekser se propušta adresa ulaza u tabelu stranica formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** ta adresa se na signal takta **CLK** upisuje u registar MAR. U slučaju da je signal **CH** na neaktivnoj vrednosti, generišu se aktivne vrednosti signala **UPR** i **clUR/WF.** Aktivnom vrednošću signala **UPR** AJUP jedinica daje procesoru indikaciju o kraju rada, a aktivnom vrednošću signala **clUR/WF** resetuje se flip-flop UR/WF koji ukazuje na tip operacije koja se izvodi. Na signal takta **CLK** se prelazi iz koraka step2 u korak step3 ako je signal **CH** na aktivnoj vrednosti, odnosna u korak step0 ako je signal **CH** na neaktivnoj vrednosti.

step3: **incMEMACC**, *if* (**MEMFC**, **ldMDRRD**),

br (*if* **MEMFC** *then* step4 *else* step3)

! U korak step3 se dolazi samo iz koraka step2. U koraku step3 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovm slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step3 u korak step4. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step3.

step4: **mxMDR**, **ldMDR**, **setI**,

br step5

! U korak step4 se dolazi samo iz koraka step3. U registru MDR nalazi se učitan deskriptor stranice u koji je potrebno utisnuti bit izmene. Bezuslovno se generišu aktivne vednosti signala **mxMDR**, **ldMDR** i **setI**. Aktivnom vrednošću signala **mxMDR** kroz multiplekser u bloku AJUP/MEM interfejs se propušta 16-bitna reč sastavljena od 14 nižih bitova registra MDR i dve jedinice na dve najviše pozicije.Aktivna vrednost signala **ldMDR** omogućuje da se na signal takta **CLK** ova vrednost upiše u registar MDR čime će najviša dva bita (V i I biti u deskriptoru) biti postavljeni na jedan. Aktivnom vednošću signala setI na signal takta **CLK** postaviće se na jedinicu I indikator vezan za ulaz AJUP jedinice dodeljen tekućoj stranici. Na signal takta **CLK** se uvek prelazi iz koraka step4 u korak step5.

step5: **writeMEM**, **incMEMACC**, *if* (**MEMFC**, **URP**, **clUR/WF**), *br* (*if* **MEMFC** *then* step0 *else* step5)

! U korak step5 se dolazi samo iz koraka step4. U koraku step5 se bezuslovno generišu aktivne vrednosti signala **writeMEM** i **incMEMACC**. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDR upisuje u memoriju MEM na adresi određenoj sadržajem registra MAR. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta **CLK** inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji MEM. Aktivnom vrednošću signala **MEMFC** pri pojavi signala takta **CLK** završava se rad - generišu se aktivne vrednosti signala **URP** i **clUR/WF**. Aktivnom vrednošću signala URP daje indikacija o kraju rada, a aktivnom vrednošću signala **clUR/WF** resetuje se flip-flop UR/WF koji ukazuje na tip operacije koja se izvodi. Pri aktivnoj vrednosti signala **MEMFC** na signal takta **CLK** se prelazi iz koraka step0.Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step5.

step6: **incMEMACC**, *if* (**MEMFC**, **ldMDRRD**),

br (*if* **MEMFC** *then* step7 *else* step6)

! U korak step6 se dolazi iz koraka step1.Iz koraka step1 se dolazi kada se pri otkrivenoj nesaglasnosti odmah prelazi na dovlačenje deskriptora tekuće stranice iz operativne memorije u AJUP jedinicu. U koraku step6 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju niža reč nultog ulaza tabele stranica tekućeg korisnika. Pri aktivnoj vrednosti signala **MEMFC** se na signal

takta **CLK** prelazi iz koraka step6 u korak step7. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step6.

step7: *if* (**GR**, **AV**),

if (□, **mxMAR**, **ldMAR**),

br (*if* **GR** *then* step10 *else* step8)

! U korak step7 se dolazi samo iz koraka step6. U koraku step7 se ispituje ispravnost virtuelne adrese tj. da li je broj stranice u dozvoljenom opsegu za tekućeg korisnika. Signal **GR** predstavlja izlaz iz komparatora u kojem se porede broj stranice i niža reč nultog ulaza tabele stranica gde je upisan ukupan broj stranica korisnika. Ako je signal **GR** na aktivnoj vrednosti to znači da je došlo do greške, pa se generiše aktivna vrednost signala **AV** kojom se postavlja odgovarajući indikator u procesoru. Ako je siganal **AV** na neaktivnoj vrednosti generišu se aktivne vrednosti signala **mxMAR** i **ldMAR**. Aktivnom vrednošću signala **mxMAR** se kroz odgovarajuće multiplekser u bloku AJUP/MEM interfejs propušta adresa ulaza u tabelu stranica formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** se obezbeđuje da se na signal takta **CLK** ta adresa upiše u registar MAR. Na signal takta **CLK** se prelazi iz koraka step7 u korak step8 ukoliko je adresa ispravna tj. signal **GR** je na neaktivnoj vrednosti. U suprotnom prelazi se u korak step10.

step8: **incMEMACC**, *if* (**MEMFC**, **ldMDRRD**),

br (*if* **MEMFC** *then* step9 *else* step8)

! U korak step9 se dolazi samo iz koraka step7. U koraku step8 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDR**, čime se obezbeđuje da se na signal takta **CLK** u registar MDR upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovm slučaju niža reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step8 u korak step9. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step8.

step9: *if* (**valid**, **writeAM**, **writeDATA**, **setV**, **writeI**, **incFIFOCNT**),

if (□, **PF, clallV**),

br (*if* **valid** *then* step1 *else* step10)

! U korak step9 se dolazi samo iz koraka step8. U koraku step9 se ispituje da li se tekuća stranica nalazi u operativnoj memoriji. Signal **valid** predstavlja vrednost V-bita deskriptora stranice. Ako ima aktivnu vrednost, stranica je u memoriji i generišu se aktivne vrednosti signala **writeAM**, **writeDATA**, **setV, writeI** i **incFIFOCNT**. Aktivne vrednosti signala **writeAM** i **writeDATA** omogućuju da se na signal takta **CLK**, u ulaz AJUP jedinice koji adresira brojač FIFOCNT, upišu broj stranice i broj njoj odgovarajućeg bloka uzet iz deskriptora stranice. Aktivnom vrednošću signala **setV** se na aktivnu vrednost postavlja V indikator vezan uz taj ulaz AJUP jedinice, a aktivnom vrednošću signala **writeI** u I flip-flop vezan za taj ulaz se prepisuje vrednost I-bita iz deskriptora. Aktivna vrednost signala **incFIFOCNT** pri signalu takta **CLK** uvećaće sadržaj ovag brojača. Ukoliko signal valid ima neaktivnu vrednost genriše se aktivna vrednost signala **PF** čime se postavlja odgovarajući indikator, i aktivna vrednost signala **clallV** kojim se resetuju svi V flip.flopovi. Na signal takta **CLK** prelazi se iz koraka step9 u korak step1 ako je signal valid na aktivnoj vrednosti, odnosno u korak step10 ako je signal valid na neaktivnoj vrednosti.

step10: **URP**, **clUR/WF**, *br* step0

! U korak step10 se dolazi ili iz koraka step7 ili iz koraka step9. U oba slučaja došlo je do neuspešnog okončavanja preslikavanja. Indikatori koji govore o tipu problema su postavljeni u ranijim koracima, pa se u koraku step10 samo daje indikacija o kraju rada postavljanjem na aktivnu vrednost signala **URP**, i resetuje UR/WF flip-flop koji ukazuje na tip operacije koja se izvodi, postavljanjem na aktivnu vrednost signala **clUR/WF**. Na signal takta **CLK** uvek se prelazi iz koraka step10 u korak step0.

2.3.3.4.2.3. Vremenski oblici signala

Vremenski oblici signala na osnovu kojih se dobijaju svi upravljački signali dati su na slici **Error! Reference source not found.**. Vremenski oblici signala su dati za primer kada se pri izvršavanju zahteva za preslikavanje otkrije da saglasnosti nema i da stranica nije u memoriji, a zatim se, nakon što operativni sistem dovuče stranicu u memoriju, generiše novi zahtev i preslikavanje uspešno završava.

Dati vremenski oblici signala odnose se na pet grupa operacija označenih sa ① do ⑤ i sa sledećim značenjem:

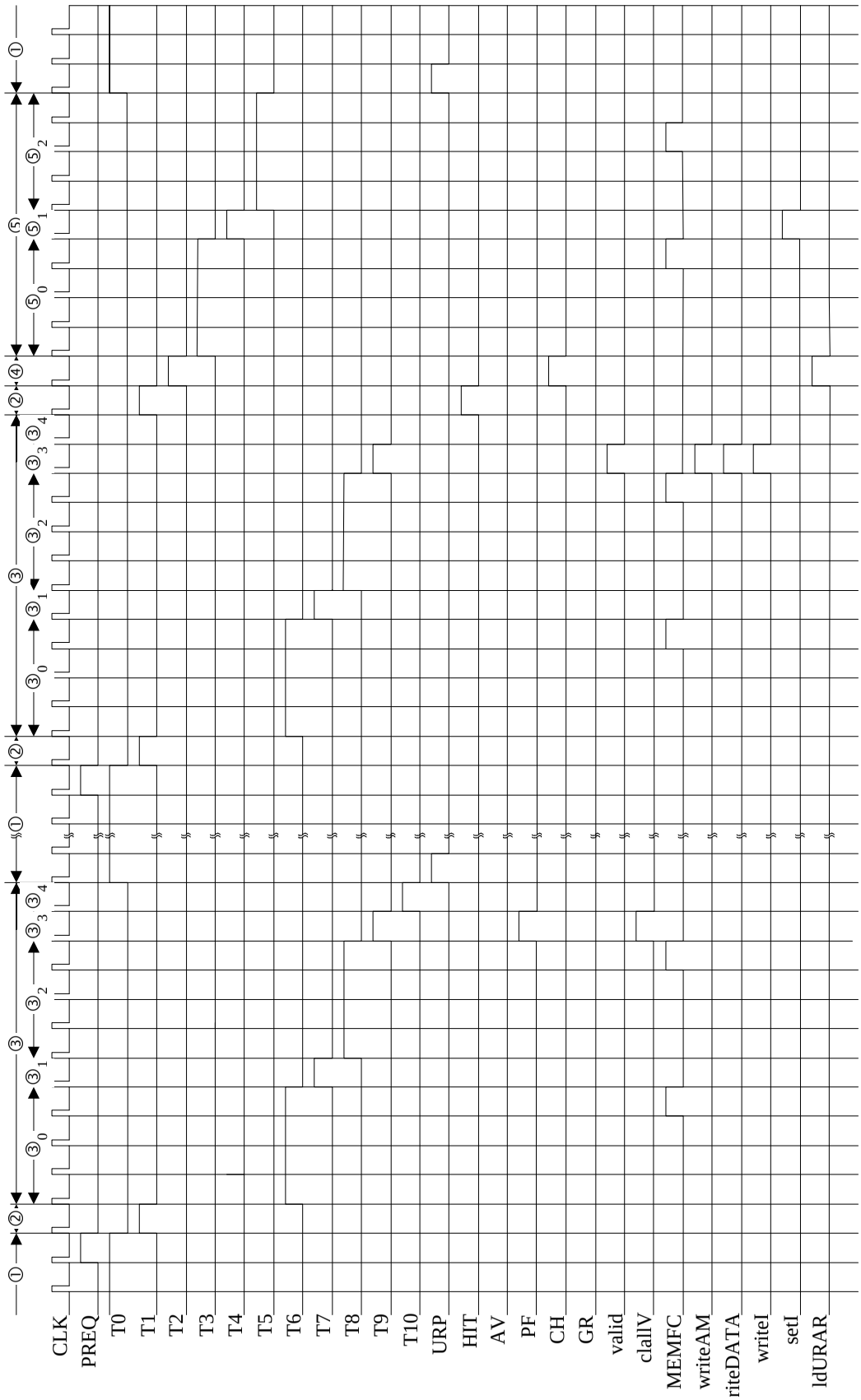
- ① čekanje na novi zahtev za preslikavanje virtuelne u realnu adresu
- ② ispitivanje postojanja saglasnosti
- ③ provera da li je stranica u operativnoj memoriji
- ④ ispitivanje da li se radi o prvoj operaciji upisa na stranici
- ⑤ utiskivanje bita izmene.

Operacije ③ se sastoji od po pet podoperacija označenih sa $③_0$ do $③_4$, a operacija ⑤ od tri podoperacije označene sa $⑤_0$ do $⑤_2$. Podoperacije operacije ③ su:

- $③_0$ čitanje nultog ulaza tabele stranica,
- $③_1$ ispitivanje ispravnosti adrese (da li je došlo do Address Violation),
- $③_2$ dovlačenje deskripotra stranice ,
- $③_3$ ispitivanje da li je stranica u memoriji,
- $③_4$ u slučaju da je stranica u memoriji, upis njenog deskriptora u AJUP jedinicu; ako stranica nije u memoriji završetak rada.

Podoperacije operacije ⑤ su:

- $⑤_0$ dovlačenje deskriptora stranice iz operativne memorije u AJUP jedinicu,
- $⑤_1$ utiskivanje bita izmene u deskriptor,
- $⑤_2$ vraćanje deskriptora sa utisnutim bitom izmene u operativnu memoriju i postavljanje odgovarajućeg I indikatora.

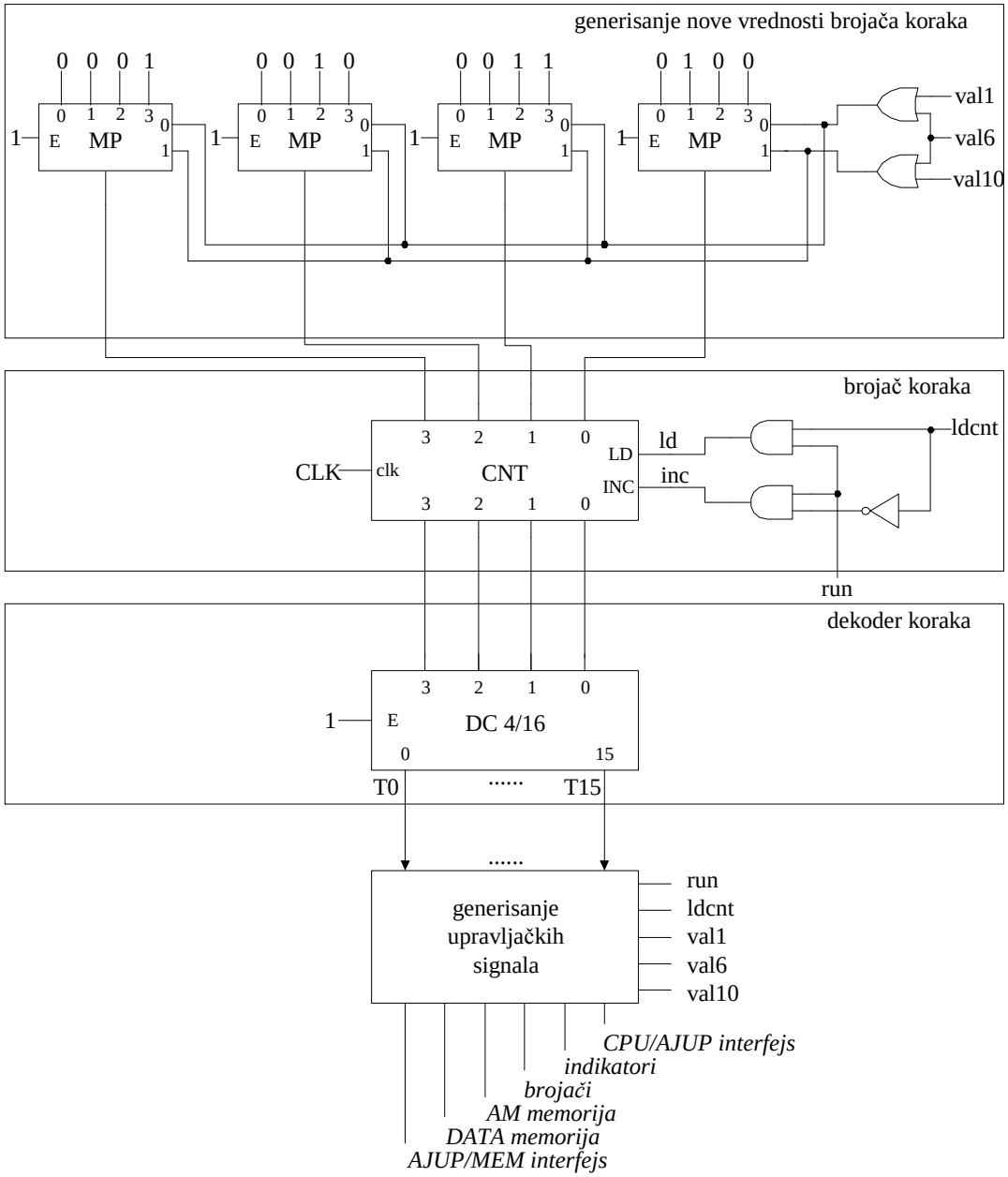


Slika 42 Vremenski oblici signala

2.3.3.4.2.4. Struktura upravljačke jedinice

Struktura upravljačke jedinice ožičene realizacije je prikazana na slici 2.13. Upravljačka jedinica se sastoji od sledećih blokova:

— generisanje nove vrednosti brojača koraka,
— brojač koraka,
— dekoder koraka i
— generisanje upravljačkih signala



Slika 2.13 Struktura upravljačke jedinice

2.3.3.4.2.4.1. Generisanje nove vrednosti brojača koraka

Ovaj blok se sastoji od multipleksera i služi za generisanje i selekciju vrednosti koju treba upisati u brojač koraka. Potreba za ovim se javlja kada treba odstupiti od sekvencijalnog izvršavanja mikrooperacija. Analizom algoritma generisanja upravljačkih signala operacione jedinice (poglavlje **Error! Reference source not found.**) se utvrđuje da su 0, 1, 6 i 11 vrednosti koje treba upisati u brojač koraka da bi se realizovala odstupanja od sekvencijalnog izvršavanja mikrooperacija. Te vrednosti su ožičene na ulazima 0, 1, 2 i 3 multipleksera. Selekcija jedne od te četiri vrednosti se postiže odgovarajućim vrednostima signala $\mathbf{val_1}$, $\mathbf{val_6}$ i $\mathbf{val_{10}}$. Ako su sva tri signala neaktivna kroz multipleksere se propušta vrednost 0, a aktivnom vrednošću samo jednog od signala $\mathbf{val_1}$, $\mathbf{val_6}$ ili $\mathbf{val_{10}}$ kroz multiplekser se propuštaju vrednosti 1, 6 i 10, respektivno.

2.3.3.4.2.4.2. Brojač koraka

Brojač koraka CNT svojom trenutnom vrednošću obezbeđuje aktivne vrednosti određenih upravljačkih signala. Brojač koraka može da radi u sledećim režimima:

— režim inkrementiranja,
— režim skoka i
— režim mirovanja.

U režimu inkrementiranja pri pojavi signala takta **CLK** vrši se uvećavanje sadržaja brojača koraka za jedan. Ovim režimom se obezbeđuje sekvencijalno generisanje upravljačkih signala iz sekvence upravljačkih signala po koracima. Režim inkrementiranja je najčešći režim rada brojača koraka. Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **inc**. Signal **inc** je aktivan ako je signal **run** aktivan, i ako je signal **ldCNT** neaktivan. Signal **run** je uvek aktivan sem kada treba obezbediti režim mirovanja. Signal **ldCNT** je uvek neaktivan sem kada treba obezbediti režim skoka.

U režimu skoka pri pojavi signala takta **CLK** vrši se upis nove vrednosti u brojač koraka. Ovim režimom se obezbeđuje odstupanje od sekvencijalnog generisanja upravljačkih signala iz sekvence upravljačkih signala po koracima. Režim skoka se javlja samo onda kada u brojač koraka treba upisati novu vrednost. Ovaj režim rada se obezbeđuje aktivnom vrednošću signala **ld**. Signal **ld** je aktivan ako su signali **run** i **ldCNT** aktivni.

U režimu mirovanja pri pojavi signala takta **CLK** ne menja se vrednost brojača koraka. Ovaj režim rada se obezbeđuje neaktivnim vrednostima signala **inc** i **ld**. Ovi signali su neaktivni kada je signal **run** neaktivan. Signal **run** je uvek aktivan sem kada treba obezbediti režim mirovanja, što se dešava kada se čeka:

— pojava signala zahteva za pevođenje adrese **PREQ** ili
— pojava signala **MEMFC** kojim se označava da je čitanje ili upis u MEM obavljen.

2.3.3.4.2.4.3. Dekoder koraka

Dekodovana stanja brojača koraka pojavljuju se kao signali $\mathbf{T_0}$ do $\mathbf{T_{15}}$ na izlazima dekodera koraka. Svakom koraku iz algoritma generisanja upravljačkih signala (poglavlje **Error! Reference source not found.**) dodeljen je jedan od ovih signala i to koraku $step_0$ signal $\mathbf{T_0}$, koraku $step_1$ signal $\mathbf{T_1}$…, s tim da se signali $\mathbf{T_{12}}$ do $\mathbf{T_{15}}$ ne koriste.

2.3.3.4.2.4.4. Generisanje upravljačkih signala

Ovaj blok se sastoji od kombinacionih mreža koje pomoću signala $\mathbf{T_0}$ do $\mathbf{T_{11}}$ koji dolaze sa dekodera koraka, signala logičkih uslova koji dolaze iz procesora CPU i operacione jedinice AJUP jedinice i saglasno algoritmu generisanja upravljačkih signala (poglavlje **Error! Reference source not found.**) generišu upravljačke signale. Ovaj blok generiše dve grupe upravljačkih signala i to upravljačke signale:

— operacione jedinice keš memorije i
— upravljačke jedinice keš memorije.

2.3.3.4.2.4.4.1. Upravljački signali operacione jedinice AJUP jedinice

Za svaki od ovih signala treba posmatrati u kojim koracima i pod kojim logičkim uslovima prema algoritmu generisanja upravljačkih signala (poglavlje **Error! Reference source not found.**) dati signal treba da ima aktivnu vrednost. Dati signal se dobija kao unija proizvoda signala dekodovanih stanja brojača koraka i logičkih uslova pod kojim u datom koraku dati signal treba da bude aktivan. Na primer, signal **ldMAR** treba da je uvek aktivan u koraku $\mathbf{T_6}$, u koraku $\mathbf{T_8}$ ako je neaktivan signal **GR**, a u koraku $\mathbf{T_2}$ ako je aktivan signal **CH**, pa se stoga ovaj signal generiše prema relaciji **ldMAR** = $\mathbf{T_2 \cdot CH + T_8 \cdot \overline{\quad} + T_6}$. Ovi signali su dati posebno za procesor CPU, memoriju MEM i blokove operacione jedinice AJUP jedinice.

Upravljački signali operacione jedinice keš memorije podeljeni su u grupe koje odgovaraju blokovima operacione jedinice i to:

— CPU/AJUP interfejs,
— indikatori,
— brojači,
— AM memorija,
— DATA memorija i
— AJUP/MEM interfejs.

2.3.3.4.2.4.4.1.1. CPU/AJUP interfejs

U blok CPU/AJUP interfejs se šalju sledeći upravljački signali:

— **ldURAR** kojim se obezbeđuje da se na signal takta **CLK** u registar URAR upiše realna adresa formirana u AJUP jedinici,
— **clUR/WF** kojim se obezbeđuje da se na signal takta **CLK** upiše neaktivna vrednost u flip-flop UR/WF,
— **AV** kojim se, na signal takta **CLK** , upisuje aktivna vrednost u flip-flop PAVF, i time signalizira procesoru do je došlo do *Address Violation* greške,
— **PF** kojim se, na signal takta **CLK** , upisuje aktivna vrednost u flip-flop PPFF, i time signalizira procesoru do je došlo do *Page Fault* greške,
— **URP** kojim se signalizira procesoru CPU da je operacija preslikavanja adrese završena, i u slučaju uspešnog preslikavanja upisuje realna adresau registar PRAR

Ovi signali se generišu prema sledećim relacijama:

ldURAR = $\mathbf{T_1 \cdot HIT}$

clUR/WF = $\mathbf{T_2 \cdot \overline{\quad} + T_5 \cdot MEMFC + T_{10}}$

AV = $\mathbf{T_7 \cdot GR}$

PF = $\mathbf{T_9}\cdot$□

URP = $\mathbf{T_2}\cdot$□ + $\mathbf{T_5}\cdot$**MEMFC** + $\mathbf{T_{10}}$

Kod njihovog generisanja kao signali logičkih uslova korišćeni su signali:

— **HIT** koji ukazuje da li postoji saglasnost,
— **CH** koji ukazuje da je u toku operacija prvog upisa na tekućoj stranici,
— **GR** koji ukazuje da je broj stranice iz virtuelne adrese koja se preslikava veći od dozvoljenog (*Address Vioaltion*)
— **valid** koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji i
— **MEMFC** koji ukazuje da su protekle četiri periode signala takta **CLK** koliko traje pristup memoriji MEM.

2.3.3.4.2.4.4.1.2. Indikatori

U blok indikatori se šalju sledeći upravljački signali:

— **setV** koji obezbeđuje da se na signal takta **CLK** upiše aktivna vrednost u adresirani flip-flop V,
— **clallV** koji obezbeđuje da se na signal takta **CLK** upiše neaktivna vrednost u sve V flip-flopove,
— **setI** koji obezbeđuje da se na signal takta **CLK** upiše aktivna vrednost u adresirani flip-flop I i
— **writeI** koji obezbeđuje da se na signal takta **CLK** upiše vrednost u adresirani flip-flop I, i to vrednost koja se nalazi u deskriptoru stranice u operativnoj memoriji (I-bit).

Ovi signali se generišu prema sledećim relacijama:

setV = $\mathbf{T_9}\cdot$**valid**

clallV = $\mathbf{T_9}\cdot$□

setI = $\mathbf{T_4}$

writeI = $\mathbf{T_9}\cdot$**valid**

Kod njihovog generisanja kao signali logičkih uslova korišćen je signal:

— **valid** koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji.

2.3.3.4.2.4.4.1.3. Brojači

U blok brojači se šalju sledeći upravljački signali:

— **incFIFOCNT** kojim se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača FIFOCNT i
— **incMEMACC** kojim se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača MEMACC.

Ovi signali se generišu prema sledećim relacijama:

incFIFOCNT = $\mathbf{T_9}\cdot$**valid**

incMEMACC = $\mathbf{T_3} + \mathbf{T_5} + \mathbf{T_6} + \mathbf{T_8}$

Kod njihovog generisanja kao signal logičkog uslova korišćen je signal:

— . **valid** koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji.

2.3.3.4.2.4.4.1.4. AM memorija

U blok AM memorija se šalje upravljački signal:

— **writeAM** kojim se obezbeđuje upis u AM memoriju.
Ovaj signal se generiše prema sledećoj relaciji:

writeAM = $\mathbf{T_9 \cdot valid}$

Kod njegovog generisanja kao signal logičkog uslova korišćen je signal:

— **valid** koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji.

2.3.3.4.2.4.4.1.5. DATA memorija

U blok DATA memorija se šalje upravljački signal:

— **writeDATA** kojim se obezbeđuje upis u DATA memoriju.
Ovaj signal se generiše prema sledećoj relaciji:

writeDATA = $\mathbf{T_9 \cdot valid}$

Kod njegovog generisanja kao signal logičkog uslova korišćen je signal:

— **valid** koji ukazuje da se stranica na kojoj se nalazi zadata adresa nalazi u operativnoj memoriji.

2.3.3.4.2.4.4.1.6. AJUP/MEM interfejs

U blok AJUP/MEM interfejs se šalju sledeći upravljački signali:

— **mxMAR** čijom se aktivnom vrednošću kroz multiplekser propušta adresa ulaza u tabelu stranica formirana u sabiraču,
— **mxMDR** čijom se aktivnom vrednošću kroz multiplekser propušta modifikovan sadržaj registra MDR (utisnute jedinice na najviša dva mesta),
— **ldMAR** kojim se obezbeđuje da se na signal takta **CLK** upiše adresa u registar MAR,
—**ldMDR** kojim se obezbeđuje da se na signal takta **CLK** upiše podatak u registar MDR i
— **writeMEM** čijom se aktivnom vrednošću realizuje upis u memoriju MEM.
Ovi signali se generišu prema sledećim relacijama:

mxMAR = $\mathbf{T_2 \cdot CH + T_7 \cdot}$ $\square$

mxMDR = $\mathbf{T_4}$

ldMAR = $\mathbf{T_2 \cdot CH + T_7 \cdot}$ $\square$ $\mathbf{+ T_6}$

ldMDRRD = $\mathbf{(T_3 + T_6 + T_8) \cdot MEMFC + T_4}$

writeMEM = $\mathbf{T_5}$

Kod njihovog generisanja kao signali logičkih uslova korišćeni su signali:

— **CH** koji ukazuje da je u toku operacija prvog upisa na tekućoj stranici,
— **GR** koji ukazuje da je broj stranice iz virtuelne adrese koja se preslikava veći od dozvoljenog (*Address Vioaltion*) i
— **MEMFC** koji ukazuje da su protekle četiri periode signala takta **CLK** koliko traje pristup memoriji MEM.

2.3.3.4.2.4.4.2. Upravljački signali upravljačke jedinice AJUP jedinice

Upravljački signali upravljačke jedinice su:

— **ldCNT** čijom se aktivnom vrednošću obezbeđuje upis vrednosti 0, 1, 6 ili 11 u brojač koraka CNT, a neaktivnom vrednošću obezbeđuje inkrementiranje tekuće vrednosti brojača koraka CNT,
— **run** čijom se aktivnom vrednošću omogućuje promena vrednosti brojača koraka CNT i to na način određen vrednošću signala **ldCNT**, a neaktivnom vrednošću onemogućava promena vrednosti brojača koraka CNT, bez obzira na vrednost signala **ldCNT** i
— $\mathbf{val_1}$, $\mathbf{val_6}$ i $\mathbf{val_{10}}$, koji obezbeđuju vrednosti 0, 1, 6 i 10 za upis u brojač koraka CNT.

Ovi signali se generišu prema sledećim relacijama:

ldCNT = $\mathbf{T_1}\cdot\square + \mathbf{T_5}\cdot\mathbf{MEMFC} + \mathbf{T_7}\cdot\mathbf{GR} + \mathbf{T_{10}}$

run = $\mathbf{T_0}\cdot\mathbf{PREQ} + \mathbf{T_1}\cdot\mathbf{HIT} + \mathbf{T_2} + \mathbf{T_3}\cdot\mathbf{MEMFC} + \mathbf{T_4} + \mathbf{T_5}\cdot\mathbf{MEMFC} + \mathbf{T_6}\cdot\mathbf{MEMFC} + \mathbf{T_7} + \mathbf{T_8}\cdot\mathbf{MEMFC} + \mathbf{T_9}\cdot\square$

$\mathbf{val_1}$ = $\mathbf{T_9}\cdot$**valid**

$\mathbf{val_6}$ = $\mathbf{T_1}\cdot\square$

$\mathbf{val_{10}}$ = $\mathbf{T_7}\cdot$ **GR**

Kod njihovog generisanja kao signali logičkih uslova korišćeni su signali:

— $\square$ koji ukazuje da nema saglasnosti,
— **PREQ** kojim se startuj operacija preslikavanja adrese,
— **MEMFC** kojim se specificira završetak pristupa memoriji MEM i
—**GR** koji ukazuje da je broj stranice iz virtuelne adrese koja se preslikava veći od dozvoljenog (*Address Vioaltion*)

2.3.4. JEDINICA SA SET-ASOCIJATIVNIM PRESLIKAVANJEM

U okviru ovog poglavlja se najpre daju karakteristike razmatrane operativne memorije i jedinice za ubrzavanje preslikavanja. Zatim se daju detalji operacione i upravljačke jedinice razmatrane jedinice za ubrzavanje preslikavanja.

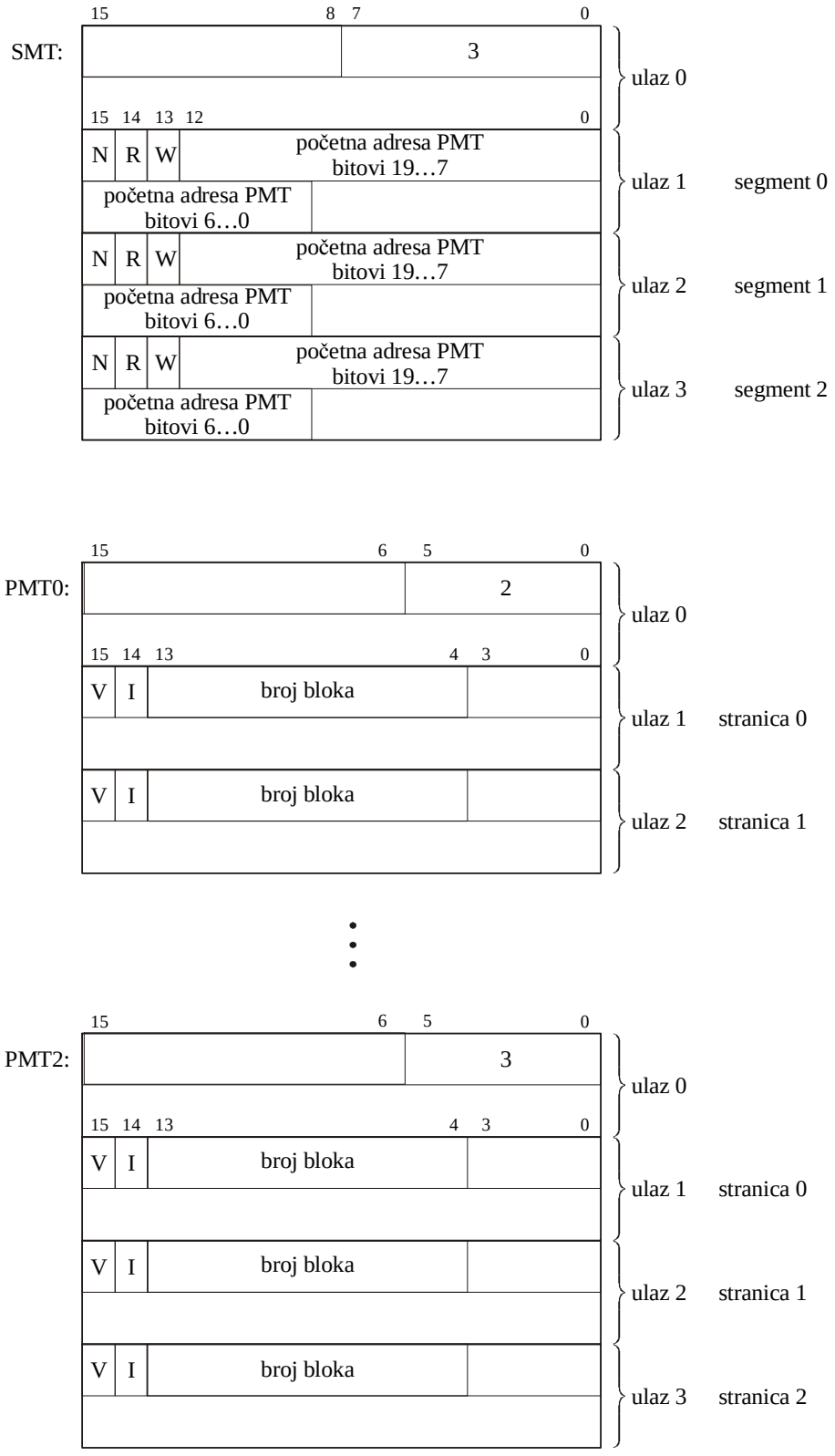
2.3.4.1. KARAKTERISTIKE OPERATIVNE MEMORIJE I JEDINICE ZA UBRZAVANJE PRESLIKAVANJA

Jedinica sa set-asocijativnim preslikavanjem (u daljem tekstu **jedinica SAJUP**) koristi se u procesoru segmentno-stranične organizacije virtuelne memorije. Virtuelna memorija je kapaciteta 16 M 16-bitnih reči i sastoji se od 256 segmenata, pri čemu je maksimalna veličina segmenta 64 stranice veličine 1 K reči. Operativna memorija je kapaciteta 1 M 16-bitnih reči i sastoji se od 1 K blokova pri čemu je veličina bloka 1 K reči. **Jedinica SAJUP** je realizovana sa 8 setova i 2 ulaza po setu.

Preslikavanje virtuelnih u realne adrese se ostvaruje kombinovanim aktivnostima operativnog sistema i **jedinice SAJUP**. Informacije neophodne za preslikavanje stranica segmenata u blokove nalaze se u tabeli segmenata koja će se u daljem tekstu označavati kao SMT (*Segment Map Table*), i tabelama stranica svakog segmenta, pri čemu će se u daljem tekstu svaka od njih označavati kao tabela PMT (*Page Map Table*). Ove tabele se nalaze u operativnoj memoriji i njima pristupaju i operativni sistem i **jedinica SAJUP**. Ovde će biti dati samo elementi tabele SMT i tabela PMT koji su neophodni za praćenje rada **jedinice SAJUP** prilikom preslikavanja virtuelnih adresa u realne.

Usvojeno je da tabela SMT ima onoliko ulaza koliko dati proces ima segmenata plus jedan ulaz u kome se čuva informacija o broju segmenata datog procesa. Svaki ulaz zauzima po dve reči. U nultom ulazu, i to u 8 nižih bitova niže reči, operativni sistem postavlja broj koji označava koliko ima segmenata proces kome pripada data tabela. Taj broj koristi **jedinica SAJUP** da prilikom pristupa tabeli SMT izvrši proveru da li je broj segmenta generisane virtuelne adrese unutar predviđenog broja segmenata datog procesa. U nenultim ulazima tabele SMT nalaze se deskriptori segmenata i to u prvom ulazu se nalazi deskriptor nultog segmenta, u drugom deskriptor prvog itd. U nižoj reči nenultog ulaza se nalazi prva reč deskriptora, a u višoj druga. Bitovi 15, 14 i 13 prve reči deskriptora sadrže bitove NRW sa identičnim značenjem kao kod segmentne organizacije virtuelne memorije (odeljak **Error! Reference source not found.**). Bitovi 12…0 prve reči deskriptora i bitovi 15…9 druge reči deskriptora predstavljaju bitove 19…7 i 6…0, respektivno, početne adrese tabele stranica datog segmenta.

Usvojeno je da tabela PMT ima onoliko ulaza koliko dati segment ima stranica plus jedan ulaz u kome se čuva informacija o broju stranica datog segmenta. Svaki ulaz zauzima po dve reči. U nultom ulazu, i to u 6 nižih bitova niže reči, operativni sistem postavlja broj koji



Slika 43 *Izgled tabele SMT sa svojim tabelama PMT za primer procesa od tri segmenta*

označava koliko ima stranica segment kome pripada data tabela. Taj broj koristi **jedinica SAJUP** da prilikom pristupa tabeli PMT izvrši proveru da li je broj stranice generisane virtuelne adrese unutar predviđenog broja stranica datog segmenta. U nenultim ulazima tabele PMT nalaze se deskriptori stranica i to u prvom ulazu se nalazi deskriptor nulte stranice, u drugom deskriptor prve itd. U nižoj reči nenultog ulaza se nalazi prva reč deskriptora, a u višoj druga. Petnaesti i šesnaesti bit prve reči deskriptora predstavljaju V-bit i I-bit, respektivno, sa identičnim značenjem kao kod stranične organizacije virtuelne memorije (odeljak **Error! Reference source not found.**).

Inicijalno kada se formira odgovarajući proces, operativni sistem za svaki proces formira tabelu SMT sa potrebnim brojem ulaza, upiše broj segmenata u nižu reč nultog ulaza i postavi sve NRW bite na odgovarajuće vrednosti. Pored toga, operativni sistem za svaki segment formira posebnu tabelu PMT sa potrebnim brojem ulaza, upiše broj stranica u nižu reč nultog ulaza i postavi sve V-bite i I-bite na vrednost 0. Početne adrese tabela PMT se upisuju u odgovarajuća polja ulaza tabele SMT sa deskriptorima segmenata. Kada dati proces dobije procesor, počinju da se generišu njegove virtuelne adrese. Svaka virtuelna adresa se preslikava pomoću **jedinice SAJUP**. Kad god **jedinica SAJUP** utvrdi da stranica nije u operativnoj memoriji, ona generiše prekid **PF** (*Page Fault*). Kao rezultat, operativni sistem dovlači adresiranu stranicu u operativnu memoriju i u ulaz tabele PMT te stranice postavi odgovajajuće informacije deskriptora u okviru čega se i V-bit deskriptora postavlja na vrednost jedan. Pri prvom upisu u neku stranicu **jedinica SAJUP** postavlja na vrednost 1 I-bit u deskriptoru date stranice u tabeli PMT. Izgled tabele SMT i tabela PMT za slučaj procesa od tri segmenta prikazan je na slici **Error! Reference source not found.**.

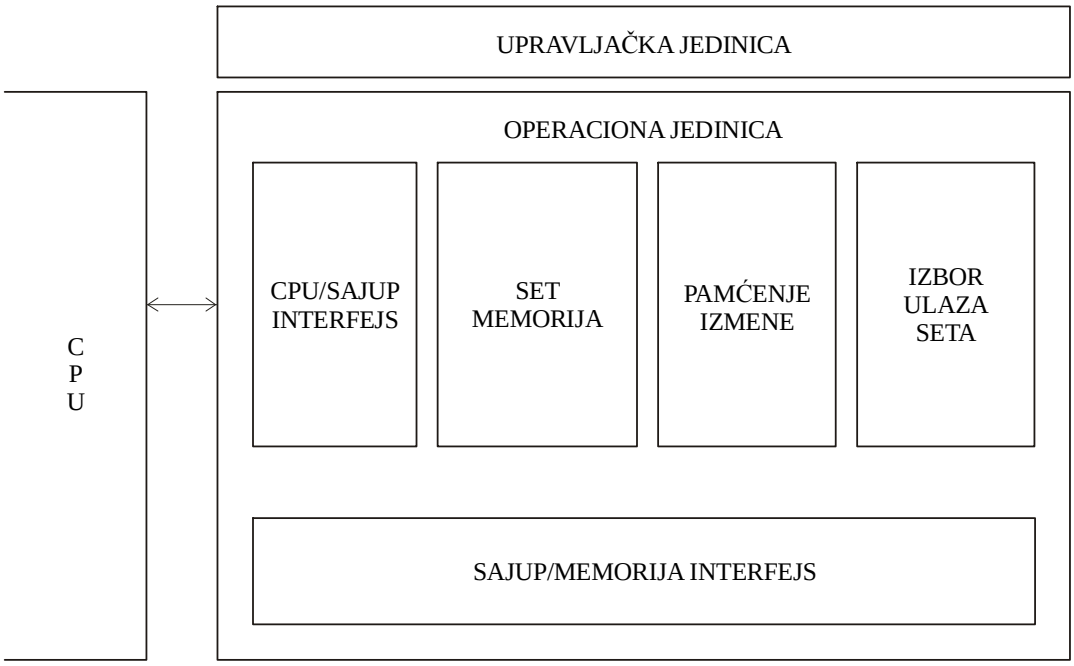
Osnovni funkcionalni delovi jedinice sa set-asocijativnim preslikavanjem, prikazane na slici **Error! Reference source not found.**, su:

- operaciona jedinica i
- upravljačka jedinica.

Operaciona jedinica se sastoji iz kombinacionih i sekvencijalnih prekidačkih mreža koje služe za pamćenje binarnih reči, izvršavanje aritmetičkih, logičkih i drugih mikrooperacija i generisanje signala logičkih uslova.

Upravljačka jedinica se sastoji iz kombinacionih i sekvencijalnih prekidačkih mreža koje služe za generisanje upravljačkih signala prema algoritmu set-asocijativnog preslikavanja virtuelne adrese u realnu.

Funkcije i struktura operacione i upravljačke jedinice dati su u posebnim poglavljima.



Slika 44 *Blok šema jedinice SAJUP*

2.3.4.2. PROCESOR CPU

Procesor CPU se obraća SAJUP onda kada treba očitati podatak iz operativne memorije ili upisati podatak u operativnu memoriju, pa je potrebno izvršiti preslikavanje virtuelne adrese u realnu adresu.

Procesor CPU šalje 24-bitnu virtuelnu adresu po linijama $\mathbf{PVAR_{23...0}}$, oznaku tekućeg korisnika po linijama $\mathbf{PUSR_{2...0}}$, adresu početka tabele segmenata tekućeg korisnika po linijama $\mathbf{PSMTAR_{19...0}}$, podatak o vrsti operacije (upis ili čitanje) po liniji **R/W**, podatak o tome da li je procesor u fazi izvršavanja ili fečovanja instrukcije po liniji **E/F** i generiše aktivnu vrednost signala **PREQ**. U slučaju uspešnog prevođenja adrese, generisana 20-bitna realna adresa se vraća po linijama $\mathbf{URAR_{19...0}}$. Prevođenje je uspešno završeno i na linijama $\mathbf{URAR_{19...0}}$ je važeći podatak onda kada SAJUP generiše aktivnu vrednost signala **URP**, a flip-flopovi PAVF i PPFF su postavljeni na neaktivnu vrednost.

Procesor CPU u delu za povezivanje sa SAJUP jedinicom koristi:

* **registre** PVAR, PUSR, PSMTAR i PRAR **za čuvanje virtuelne adrese i podataka o korisniku, odnosno za smeštanje generisane realne adrese u slučaju uspešno izvedenog preslikavanja,**
* **flip-flopove** PR/WF, PAVF, PPFF, PE/FF **za indikaciju o tipu operacije odosno o razlogu nemogućnosti uspepnog preslikavanja i**
* **upravljačke signale** PREQ, AV, PF i URP **za sinhronizaciju sa SAJUP jedinicom**

Registar **PVAR** (*Processor Virtual Address Register*) služi za čuvanje virtuelne adrese lokacije memorije MEM sa koje treba očitati podatak u slučaju operacije čitanja ili adrese lokacije memorije MEM u koju treba upisati podatak u slučaju operacije upisa. Izlazne linije ovog registra se vode kao

24 linije **$PVAR_{23...0}$** u CPU/SAJUP interfejs SAJUP jedinice. Pretpostavlja se da je procesor CPU pre obraćanja SAJUP jedinici već upisao virtuelnu adresu u registar PVAR, tako što je generisao aktivnu vrednost signala **ldPVAR** i na signal takta **CLK** izvršio upis sadržaja sa linija **$PVARIN_{19...0}$.**

Registar **PUSR** (*Processor USer Register*) sadrži oznaku tekućeg korisnika procesora, a registar **PSMTAR** (*Processor Segment Map Table Address Register*) pokazuje na početak tabele segmenata tekućeg korisnika. Pretpostavlja se da je operativni sistem, pri dodeljivanju procesora korisniku, upisao potrebne podatke u ove registre, tako što je generisao aktivnu vrednost signala **ldPUSR** i na signal takta izvršio upis sadržaja sa linija **$PUSRIN_{2...0}$** u slučaju registra PUSR, odnosno generisao aktivnu vrednost signala **ldPSMTAR** i na signal takta izvršio upis sadržaja sa linija **$PSMTAR_{19...0}$.** u slučaju registra **PSMTAR**.

Registar **PRAR** (*Processor Real Address Register*) služi za čuvanje realne adrese koja je generisana u slučaju uspešnog preslikavanja virtuelne u realnu adresu. Generisana adresa se iz **CPU/SAJUP** interfejsa **SAJUP** jedinice vodi po linijama **$URAR_{19...0}$** na ulaze registra PRAR. Aktivna vrednost signala **URP** se koristi da se na signal takta **CLK** izvrši upis realne adrese sa linija **$URAR_{19...0}$** u registar **PRAR,** pod uslovom da su flip-flopovi **PAVF** i **PPFF** postavljeni na nulu.

Flip-flop **PR/WF** (*Processor Read/ Write Flag*) služi za čuvanje podatka o tipu operacije u kojoj se koristi zadata virtuelna adresa. Aktivnom vrednošću signala **R/W** (*Read/Write operation*), koji se dobija sa izlaza **PR/WF** flip-flopa, se specificira operacija čitanja, a neaktivnom operacija upisa. Pretpostavlja se da je procesor **CPU** pre toga generišući aktivnu vrednost ili signala **setR/W** ili signala **clR/W** trajanja jedne periode signala takta **CLK** postavio flip-flop **PR/WF** ili na aktivnu vrednost ili na neaktivnu vrednost, respektivno. Podatak o tipu operacije prosleđuje se **SAJUP** jedinici na signal takta **CLK**, ako je aktivan signal **PREQ**.

Flip-flop **PAVF** (*Procesor Address Violation Flag*) služi za indikaciju procesoru da je generisana virtuelna adresa neispravna tj. da je broj segmenta veći od ukupnog broja segmenata tekućeg procesa ili da je broj stranice na koju se adresa odnosi veći od ukupnog broja stranica tekućeg segmenta korisnika, ili da pristup datom segmentu nije dozvoljen. Ovaj flip-flop postavlja se na aktivnu vrednost na signal takta **CLK** pri aktivnoj vrednosti signala **AV**. Signal **AV** generiše **SAJUP** jedinica ako u procesu preslikavanja adrese otkrije da je došlo do Address Violation. Resetovanje **PAVF** se vrši pri postavljanju novog zahteva za preslikavanje tj. pri aktivnoj vrednosti signala **PREQ**.

Flip-flop **PPFF** (*Procesor Page Fault Flag*) služi za indikaciju procesoru da se stranica na koju se generisana virtuelna adresa odnosi ne nalazi u operativnoj memoriji. Procesor na osnovu aktivne vrednosti sadržaja **PPFF** generiše Page Fault prekid, a operativni sistem obrađujući ovaj prekid suspenduje tekućeg korisnika i organizuje prebacivanje bloka sa diska u operativnu memoriju. Ovaj flip-flop postavlja se na aktivnu vrednost na signal takta **CLK** pri aktivnoj vrednosti signala **PF**. Signal **PF** generiše **SAJUP** jedinica ako u procesu preslikavanja adrese otkrije da je došlo do Page Fault-a. Resetovanje **PPFF** se vrši pri postavljanju novog zahteva za preslikavanje tj. pri aktivnoj vrednosti signala **PREQ**.

Upravljački signal **PREQ** (*Processor REQuest*) se koristi da procesor **CPU** aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira **SAJUP** jedinici da se na linijama **$PVAR_{23...0}$**, **$PUSR_{2...0}$**, **$PSMTAR_{19...0}$** , **E/F** i **R/W** nalaze podaci potrebni za operaciju preslikavanja virtuelne u realnu adresu, i da treba, na signal takta **CLK**, ove sadržaje upisati u odgovarajuće registre i time pokrenuti preslikavanje.

Upravljački signali **AV** (*Address Violation*) i **PF** (*Page Fault*) se koriste da SAJUP jedinica aktivnom vrednošću ovih signala trajanja jedne periode signala takta **CLK** signalizira procesoru **CPU**

da se preslikavanje zadate adrese ne može uspešno izvršiti jer je došlo do prekoračenja dozvoljenog opsega adresa ili je došlo do straničnog prekida. U slučaju aktivne vrednosti signala **AV** na signal takta se upisuje aktivna vrednost u **PAVF** flip-flop, a u slučaju aktivne vrednosti signala **PF** aktivna vrednost se upisuje u **PPFF** flip-flop.

Upravljački signal **URP** (*Unit RePly*) se koristi da **SAJUP** jedinica aktivnom vrednošću ovog signala trajanja jedne periode signala takta **CLK** signalizira procesoru **CPU** da je preslikavanje adrese završeno - ili uspešno ili neuspešno (što se vidi po vrednostima upisanim u **PAVF** i **PPFF**). U slučaju uspešnog preslikavanja iz **SAJUP** jedinice aktivnom vrednošću signala **URP** se signalizira i da se na linijama $\mathbf{URAR_{19...0}}$ nalazi generisana realna adresa i da aktivnom vrednošću signala **URP** treba na signal takta **CLK** ovu adresu upisati u registar **PRAR**.

2.3.4.3. OPERACIONA JEDINICA

Operaciona **SAJUP** jedinica sastoji se iz sledećih blokova:

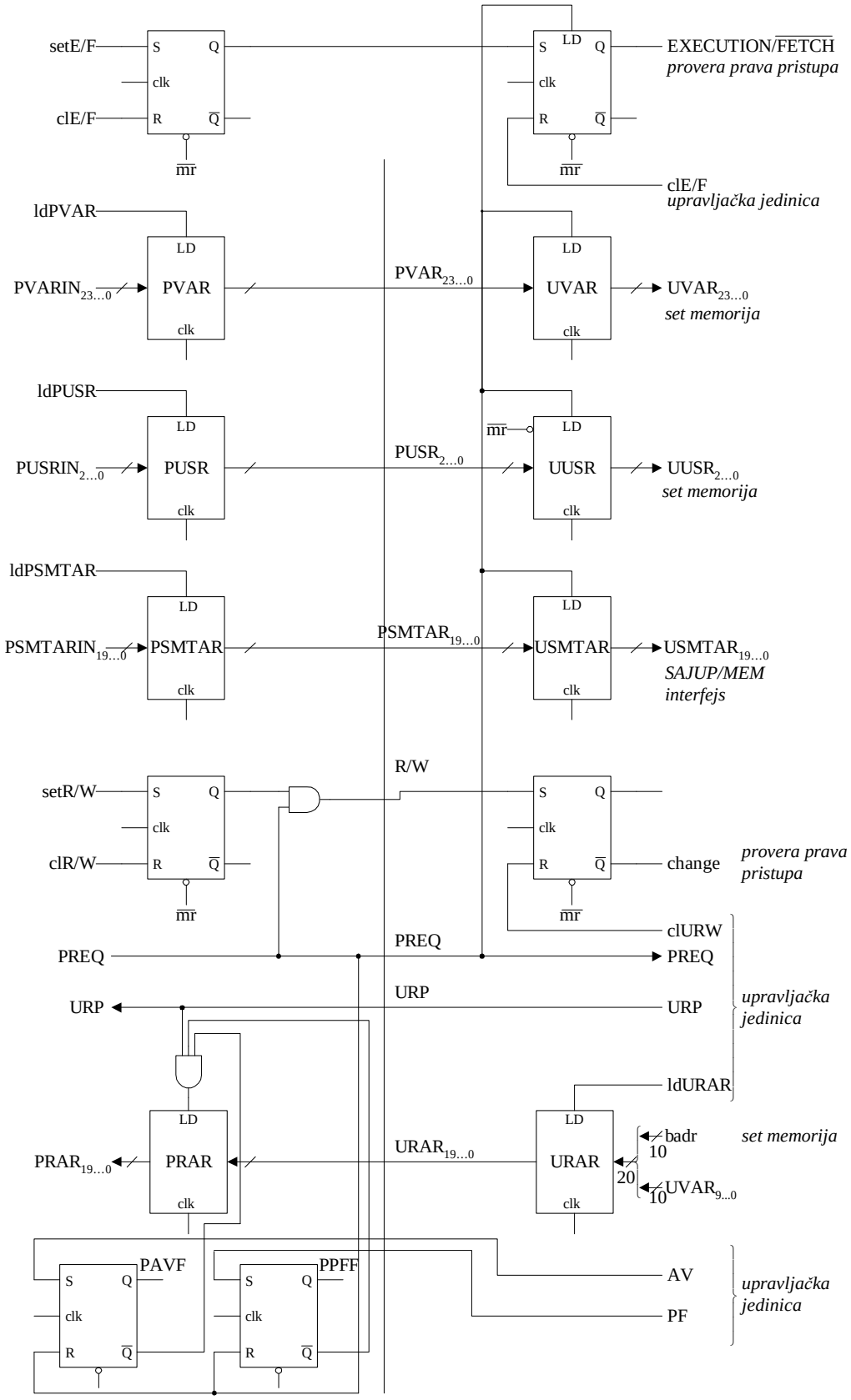
- **CPU/SAJUP interfejs**,
- **set memorija**,
- **SAJUP/MEMORIJA interfejs**,
- **pamćenje izmene** i
- **izbor ulaza seta**.

2.3.4.3.1. CPU/SAJUP interfejs

Strukturna šema bloka **CPU/SAJUP interfejsa** prikazana je na slici 2.3.

Interfejs sačinjavaju registri:

- **UVAR** (**V**irtual **A**ddress **R**egister)
- **URAR** (**R**eal **A**ddress **R**egister)
- **USMTAR** (**S**egment **M**ap **T**able **A**ddress **R**egister)
- **UUSR (U**ser **S**et **R**egister)


setE/F
clE/F
S
Q
clk
R
$\overline{Q}$
$\overline{mr}$
LD
EXECUTION/$\overline{FETCH}$
provera prava pristupa
clE/F
upravljačka jedinica
ldPVAR
PVARIN$_{23...0}$
PVAR
PVAR$_{23...0}$
UVAR
UVAR$_{23...0}$
set memorija
ldPUSR
PUSRIN$_{2...0}$
PUSR
PUSR$_{2...0}$
UUSR
UUSR$_{2...0}$
set memorija
ldPSMTAR
PSMTARIN$_{19...0}$
PSMTAR
PSMTAR$_{19...0}$
USMTAR
USMTAR$_{19...0}$
SAJUP/MEM interfejs
R/W
setR/W
clR/W
change
provera prava pristupa
clURW
PREQ
upravljačka jedinica
URP
ldURAR
PRAR
PRAR$_{19...0}$
URAR$_{19...0}$
URAR
badr
10
20
10
UVAR$_{9...0}$
set memorija
PAVF
PPFF
AV
PF
upravljačka jedinica

***Slika 3** CPU/SAJUP interfejs*

Registri **UVAR** i **URAR** služe za prihvatanje virtuelne i realne adrese, respektivno.

Registar **USMTAR** pokazuje na početak **SMT** (**S**egment **M**ap **T**able) tabele. Takođe se u njega upisuje vrednost samo pri inicijalizaciji sistema.

Registar **UUSR** sadrži oznaku tekućeg korisnika.

CPU i **SAJUP** rade sa istim signalom takta.

2.3.4.3.2. Set Memorija

Način funkcionisanja ovog bloka je isti kao i odgovarajućih blokova jedinice sa asocijativnim preslikavanjem (poglavlje **Error! Reference source not found.**) i jedinice sa direktnim preslikavanjem (poglavlje **Error! Reference source not found.**).

Treba napomenuti da se sadržaj registra UVAR koristi dvojako. U bloku **SAJUP/MEMORIJA interfejs** signali $\mathbf{UVAR_{23...16}}$ i $\mathbf{UVAR_{15...0}}$ sadrže broj segmenta i broj stranice pri pristupu tabelama SMT i PMT, respektivno. U bloku **set memorija** signali $\mathbf{UVAR_{23...13}}$ i $\mathbf{UVAR_{12...10}}$ sadrže tag i broj seta, respektivno, pri pristupu set memoriji.

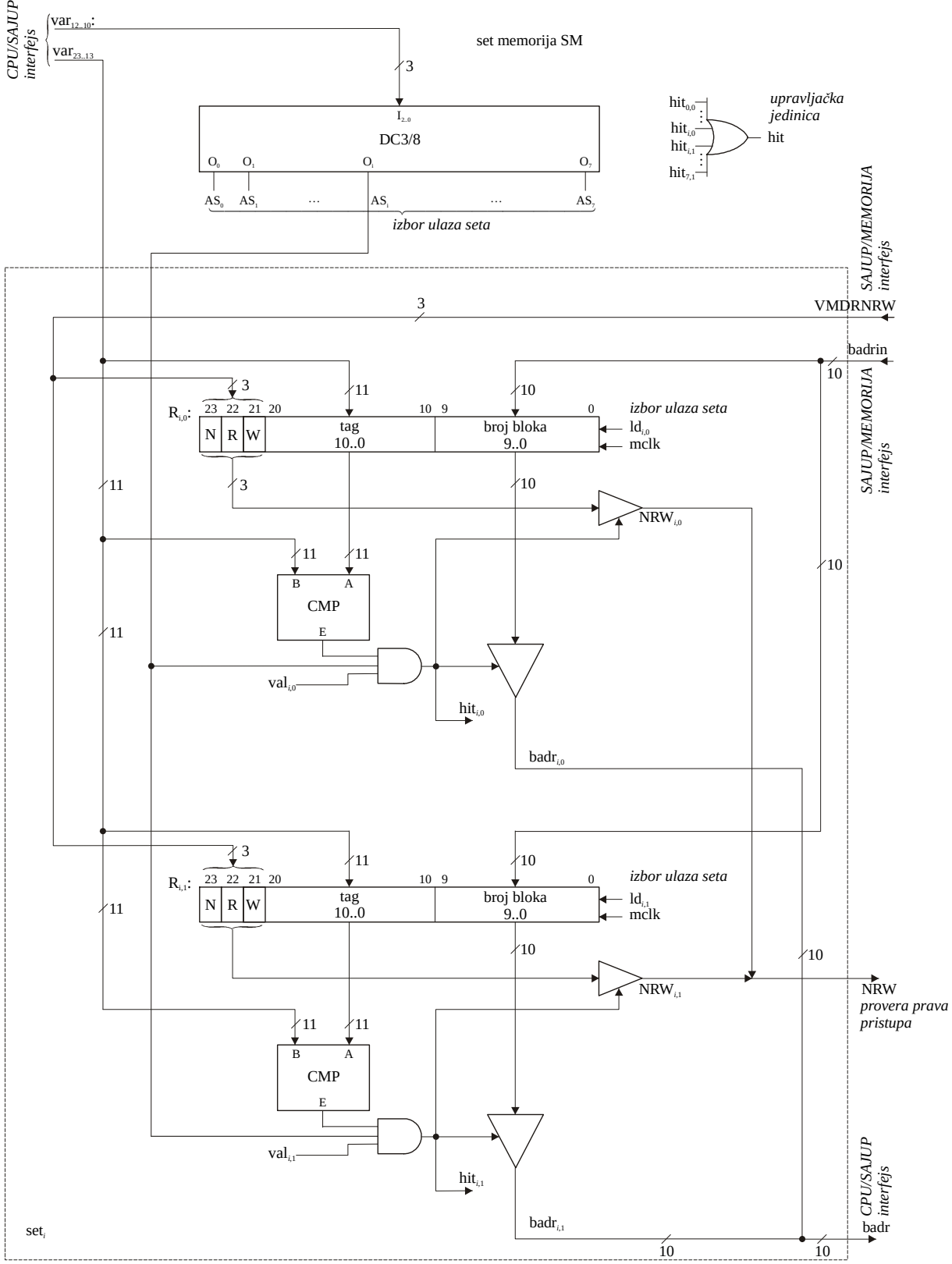
Blok **set memorija**, prikazan na slikama 4, 5 i 6, sačinjavaju:

- set memorija SM,
- kola za proveru prava pristupa i
- flip-flopovi $val_{0,0}$, $val_{0,1}$, …, $val_{7,1}$.

2.3.4.3.2.1. Set memorija SM

Set memorija SM se sastoji od 8 identičnih setova pri čemu su detalji realizacije dati samo za set set_i (slika 4). Ulaz svakog seta sadrži polje bitova prava pristupa *NRW*, polje *tag* i polje *broj bloka*. Od 8 setova u datom trenutku se pristupa samo jednom od njih i to onom koji je određen bitovima $\mathbf{UVAR_{12...10}}$, koji predstavljaju broj seta generisane virtuelne adrese. Ovi bitovi se vode na ulaze dekodera DC3/8 da bi se na izlazima dobili signali selekcije setova $\mathbf{AS_i}$, $i = 0, \ldots, 7$. Unutar jednog seta realizacija je kao i za slučaj asocijativnog preslikavanja sa 2 ulaza.

Nad set memorijom SM mogu da se realizuju samo dve operacije i to: utvrđivanje da li ima saglasnosti i ako ima dobijanje broja bloka i bitova prava pristupa, i upisivanje bitova prava pristupa *NRW*, bitova *tag* i bitova *broj bloka*.



Slika 4 *Set memorija SM*

Utvrđivanje da li ima saglasnosti se vrši za oba ulaza svih 8 setova istovremeno upoređivanjem signala koji se dobija koji se dobija konkatenacijom signala **UUSR$_{2..0}$** i **UVAR$_{23..13}$** generisane virtuelne adrese sa signalom koji se dobija konkatenacijom polja **user** i **tag** odgovarajućeg ulaya odabranog seta. U slučaju seta set$_i$, formiraju se dva signala

saglasnosti $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$ za ulaze 0 i 1, respektivno. Na formiranje ovih signala utiče signal selekcije seta $\mathbf{AS_i}$. Zbog toga ako je signal $\mathbf{AS_i}$ jedinica, jedan od signala saglasnosti $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$ seta set_i može da bude jedinica. Signali selekcije ostalih 7 setova su nule, pa su njihovi signali saglasnosti nule. Na signale $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$ utiču i signali $\mathbf{val_{i,0}}$ i $\mathbf{val_{i,1}}$, koji određuju, posebno za svaki od dva ulaza seta, da li je važeći.

Signal saglasnosti **hit** je ILI funkcija 8 parova signala saglasnosti za 8 setova set memorije SM.

Na sličan način se formiraju i bitovi prava pristupa **NRW** i broja bloka **badr**. Oni se posebno formiraju za svaka dva ulaza svih 8 setova. Svi se oni preko trostatičkih bafera vezuju međusobno i daju bitove prava pristupa **NRW** i broja bloka **badr** za kompletnu memoriju SM. U slučaju seta set_i formiraju se signali prava pristupa $\mathbf{NRW_{i,0}}$ i $\mathbf{NRW_{i,1}}$ i signali broja bloka $\mathbf{badr_{i,0}}$ i $\mathbf{badr_{i,1}}$ za ulaze 0 i 1, respektivno, i puštaju se na linije **NRW** i **badr** cele set memorije signalima $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$, respektivno. Ukoliko je set set_i selektovan, samo je signal selekcije seta $\mathbf{AS_i}$ jedinica, pa samo jedan od signala saglasnosti $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$ može da bude jedinica. Koji će od njih biti jedinica zavisi od toga da li je i u kom je od dva ulaza tog selektovanog seta otkrivena saglasnost. Ukoliko ni u jednom nema saglasnosti, signal saglasnosti **hit** cele set memorije SM je nula, pa signali **NRW** i **badr** cele set memorije ostaju u stanju visoke impedanse. Ako se u jednom od dva ulaza otkrije saglasnost, jedan od signala $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$ će postati jedinica, pa će se bitovi prava pristupa i broja bloka datog ulaza pojaviti na linijama **NRW** i **badr** cele set memorije SM. Treba pomenuti da je **jedinica SAJUP** tako isprojektovana da se ne može desiti da se, ako je set set_i selektovan, na oba ulaza otkrije saglasnost i da signali $\mathbf{hit_{i,0}}$ i $\mathbf{hit_{i,1}}$ budu istovremeno aktivni.

Operacija upisivanja bitova prava pristupa *NRW*, bitova *tag* i bitova *broj bloka* se realizuje na sledeći način. Bitovi prava pristupa i broja bloka se vode po linijama **VMDRNRW** i **badrin**, respektivno, iz očitanog deskriptora istovremeno na svaka dva ulaza svih 8 setova. To se isto čini sa bitovima *tag* koji se vode po linijama $\mathbf{UVAR_{23...13}}$ iz generisane virtuelne adrese. U koji će ulaz od 8 parova ulaza set memorije SM da se izvrši upis zavisi od signala **ld** ovih ulaza. Ako je set set_i selektovan signalom selekcije $\mathbf{AS_i}$ se obezbeđuje da samo signali $\mathbf{ld_{i,0}}$ i $\mathbf{ld_{i,1}}$ mogu da budu aktivni. Korišćenjem **FIFO** algoritma zamene unutar seta set_i bira se jedan od dva ulaza ovog seta za upis deskriptora i time obezbeđuje da samo jedan od signala $\mathbf{ld_{i,0}}$ i $\mathbf{ld_{i,1}}$ bude aktivan (slika 10).

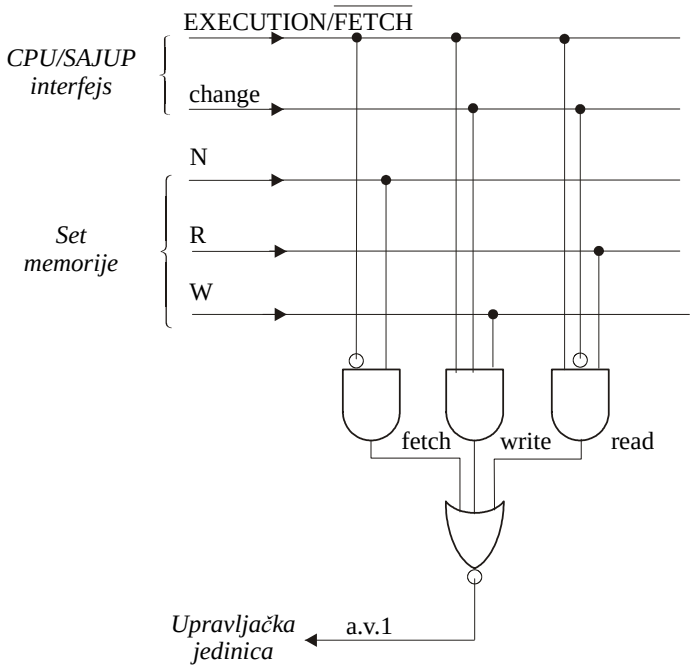
2.3.4.3.2.2. Kola za proveru prava pristupa

Kola za proveru prava pristupa (slika 5) su realizovana na isti način kao i odgovarajuća kola kod jedinice sa direktnim preslikavanjem za virtuelnu memoriju segmentne organizacije (poglavlje **Error! Reference source not found.**).

2.3.4.3.2.3. Flip-flopovi $\mathbf{val_{i,0}}$, $\mathbf{val_{i,1}}$, …, $\mathbf{val_{i,7}}$

Flip-flopovi $val_{i,0}$, $val_{i,1}$, i = 0, …, 7, (slika 6) služe za označavanje važećih ulaza set memorije SM pri čemu je usvojeno da vrednost 1 označava da je ulaz važeći, a vrednost 0 da je ulaz nevažeći. Ulaz je važeći kada je u njega upisan deskriptor neke stranice koja je u memoriji. Inicijalno ovi flip-flopovi su postavljeni na vrednost 0. Kada se u neki ulaz set memorije SM unese novi sadržaj, u flip-flop val koji odgovara tom ulazu se upiše vrednost 1. Upis vrednosti 1 u jedan od flip-flopova $val_{i,0}$, $val_{i,1}$, i = 0, …, 7, se realizuje generisanjem aktivne vrednosti jednog od signala $\mathbf{ld_{i,0}}$, $\mathbf{ld_{i,1}}$, i = 0, …, 7. Ako je set set_i selektovan signalom

elekcije $\mathbf{AS_i}$ se obezbeđuje da samo signali $\mathbf{ld_{i,0}}$ i $\mathbf{ld_{i,1}}$ mogu da budu aktivni. Korišćenjem



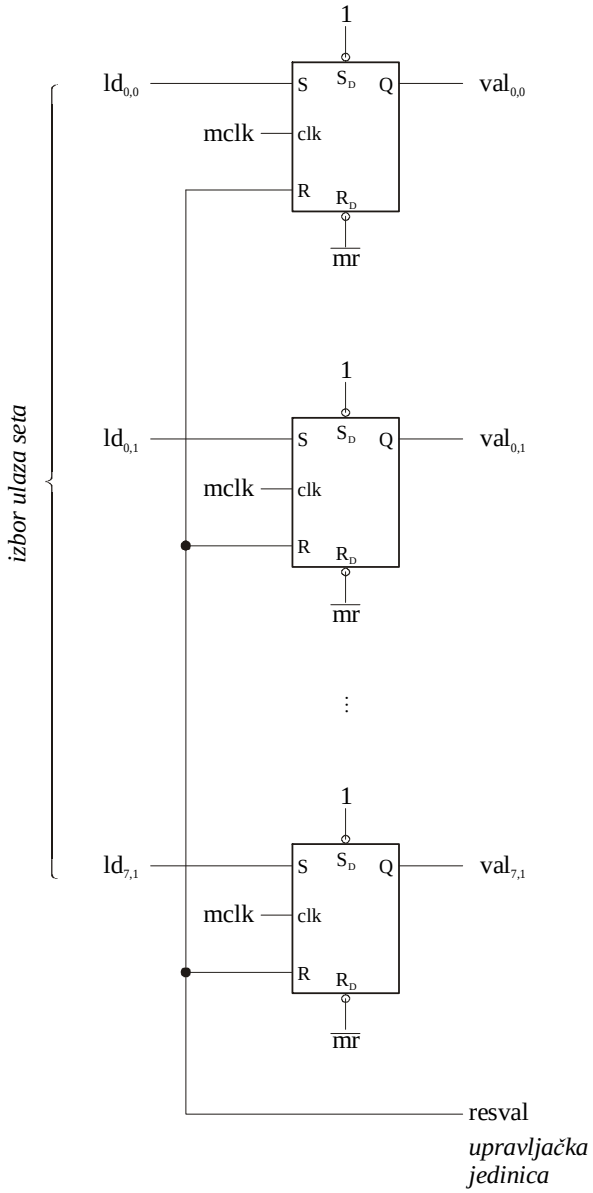
***Slika 5** Provera prava pristupa*

FIFO algoritma zamene unutar seta set_i bira se jedan od dva ulaza ovog seta za upis deskriptora i time obezbeđuje da samo jedan od signala $\mathbf{ld_{i,0}}$ i $\mathbf{ld_{i,1}}$ bude aktivan (slika 10). Signalom **resval** svi ovi flip-flopovi se postavljaju na vrednost 0 onda kada se svi ulazi set memorije SM proglašavaju za nevažeće. Jedna od situacija kada to treba učiniti je kada **jedinica SAJUP** generiše prekid Page Fault zbog toga što zahtevana stranica nije u memoriji. Tada operativni sistem treba da dovuče datu stranicu sa diska. Pri tome ako je operativna memorija puna, operativni sistem treba da izbaci neki blok iz operativne memorije da bi na njegovo mesto dovukao novu stranicu. Taj proces postaje blokiran, njemu se oduzima procesor i novi proces dobija procesor. Tom prilikom se menja i sadržaj registra SMTAR koji sada ukazuje na tabelu SMT procesa kome je dodeljen procesor. S obzirom da se u **jedinici SAJUP** nalaze deskriptori nekih stranica segmenata procesa kome je oduzet procesor, sve ulaze set memorije SM treba proglasiti za nevažeće. Zbog toga se svaki put kada se desi prekid Page Fault svi flip-flopovi val postavljaju na vrednost 0.

2.3.4.3.3. SAJUP/MEMORIJA interfejs

SAJUP/MEMORIJA interfejs se koristi kada se čita nulti ulaz tabele SMT, kada se dovlači deskriptor segmenta, i to i niža i viša reč, iz tabele SMT, kada se dovlači nulti ulaz tabele PMT, kada se dovlači deskriptor stranice iz tabele PMT, i to samo niža reč, i kada se ažurira I-bit nekog deskriptora u tabeli PMT.

Bloka **SAJUP/MEMORIJA interfejs** prikazan je na slici 7



Slika 6 *Flip-flopovi val*

U svim slučajevima na adresne linije memorije $\mathbf{A_{19...0}}$ se propušta sadržaj registra MAR. Postoje 4 karakteristična slučaja pristupa memoriji. Prvi slučaj je kada pristupamo nultom ulazu tabele SMT. Tada neaktivnom vrednošću signala mxADDER kroz multiplekser MP propuštamo signal $USMTAR_{19..0}$ na ulaz donjeg multipleksera MP kroz koji isti signal prolazi zahvaljujući aktivnoj vrednosti signala mxMAR. Na ulazu registra MAR tada se nalazi sadržaj registra USMTAR (Početna adresa SMT). Drugi slučaj je kada želimo da pristupimo deskriptoru tekućeg segmenta u tabeli SMT. U registru USR upisuje se broj segmenta + 1. Sadržaj ovog registra se zahvaljujući neaktivnom signalu mxADDER propušta kroz multiplekser na b ulaz sabirača ali pomeren za jedan bit ulevo, tako da se na b ulazu sabirača nalazi sadržaj 2*USR. Na a ulaz sabirača se isto zbog neaktivnog signala mxADDER propušta $USMTAR_{19..0}$. Sabirač obavlja funkciju a+b tako da se na njegovom izlazu dobija

adresa deskriptora tekućeg segmenta, koja se propušta kroz donji multiplekser zahvaljujući neaktivnom signalu mxMAR. U ovom slučaju na ulazu registra MAR nalazi se adresa deskriptora tekućeg segmenta u memoriji. Treći slučaj je kada pristupamo nultom ulazu tabele PMT. Tada neaktivnom vrednošću signala mxADDER kroz multiplekser MP propuštamo signal dobijen konkatenacijom signala $MDRL_{12..0}$ i $MDRH_{15..9}$ (početna adresa PMT) na ulaz donjeg multipleksera MP, kroz koji isti signal prolazi zahvaljujući aktivnom signalu mxMAR. Na ulazu registra MAR tada se nalazi početna adresa PMT-a. Četvrti slučaj je kada želimo da pristupimo deskriptoru stranice. Aktivnom vrednošću signala mxADDER na a ulaz sabirača dovodi se početna adresa PMT, a na b ulaz (broj stranice+1) * 2. Na izlazu sabirača biće adresa deskriptora stranice kojoj pristupamo i ona se propušta na ulaz MAR registra kroz multiplekser zahvaljujući neaktivnoj vrednosti signala mxMAR.

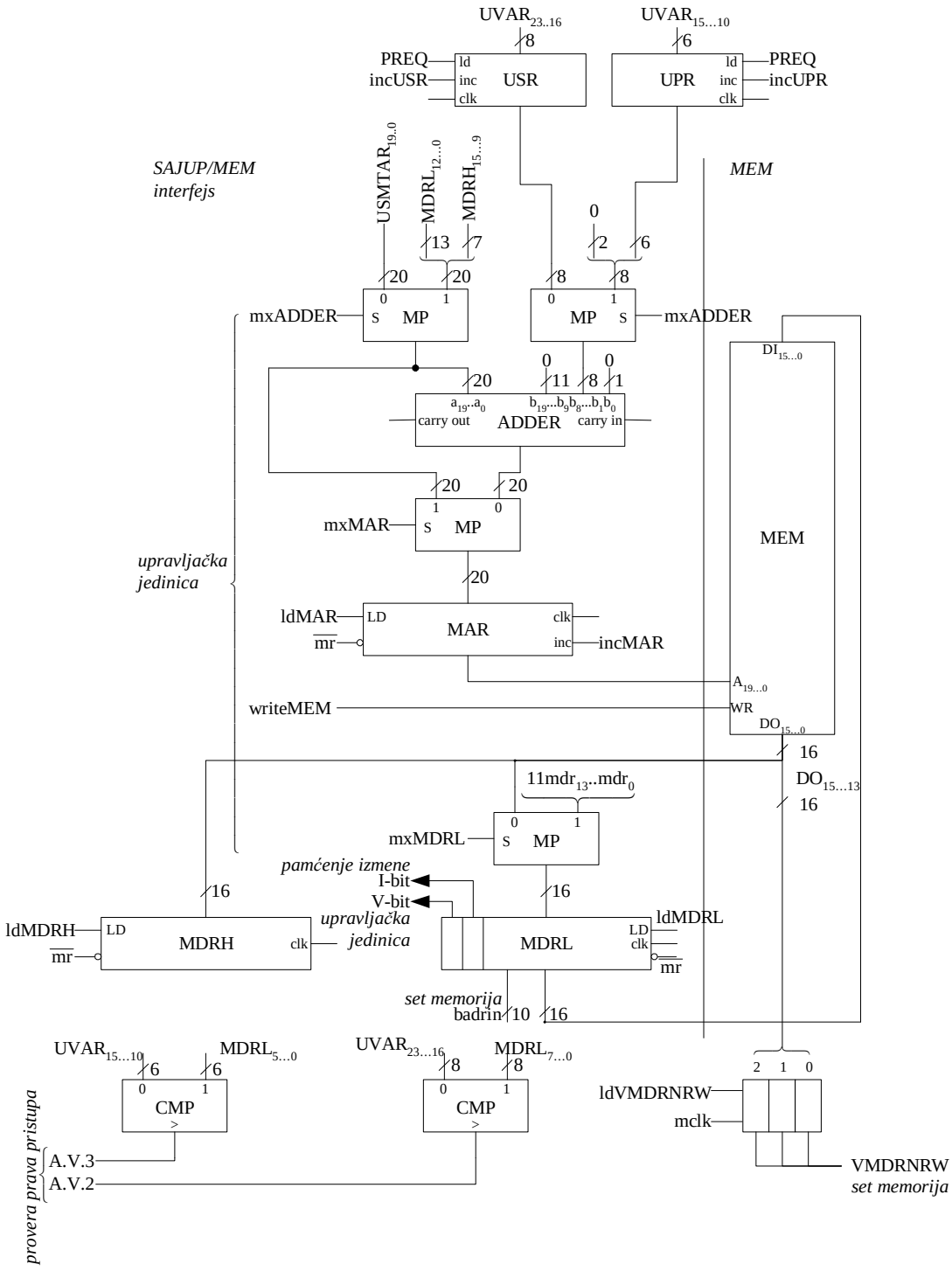
Deo **SAJUP/MEMORIJA interfejsa** za rad sa podacima sadrži 16-razredne prihvatne registre MDRL i MDRH i trorazredni registar VMDRNRW u koje se smeštaju sadržaji iz tabela SMT i PMT, dva komparatora za proveru da li je broj segmenta generisane virtuelne adrese unutar broja segmenata datog procesa i da li je broj stranice generisane virtuelne adrese unutar broja stranica datog segmenta. Reč koja se čita iz operativne memorije dolazi direktno u registre MDRL i MDRH, a u njih se upisuje upravljačkim signalima **ldMDRL** i **ldMDRH**, respektivno. Najviša tri bita reči se vode na ulaze registra VMDRNRW i u njega upisuje signalom **ldVMDRNRW**.

Postoje dva slučaja čitanja iz tabele SMT i to:

- ako je reč koja se čita donja reč nultog ulaza tabele SMT, onda se ona upisuje u registar MDRL, odakle 8 najmlađih razreda ide na poređenje u 8-razredni paralelni komparator sa bitovima $\mathbf{UVAR_{23...16}}$ koji predstavljaju broj segmenta virtuelne adrese da bi se utvrdilo da li je ispravan pristup segmentu i
- ako se vrši čitanje da bi se preneo deskriptor segmenta onda se čitaju i niža i viša reč i upisuju u registre MDRL i MDRH, respektivno, pri čemu se tri najstarija bita upisuju i u registar VMDRNRW i čuvaju za upis u set memoriju SM, da bi se zatim od razreda 12…0 registra MDRL i razreda 15…9 registra MDRH formirala početna adresa tabele stranica, dok razredi 8…0 registra VMDRH nisu od interesa za **jedinicu SAJUP**.

Postoje tri slučaja čitanja iz tabele PMT i to:

- ako je reč koja se čita donja reč nultog ulaza tabele PMT, onda ona ide na poređenje u 6-razredni paralelni komparator, gde se ispituje ispravnost generisane adrese stranice,
- ako se reč čita da bi se preneo deskriptor u set memoriju SM, ona se upisuje u registar MDRL i tek ako je V-bit dovučenog deskriptora jedinica, što proverava upravljačka jedinica, iz registra MDRL se izvlači broj bloka određen razredima 13…4 i vodi u set memoriju SM po linijama **badrin,** dok razredi 3…0 registra MDRL nisu od interesa za **jedinicu SAJUP**, i
- ako se deskriptor čita iz tabele PMT da bi se u njega utisnula jedinica na poziciji I-bita za slučaj prve izmene u okviru stranice.

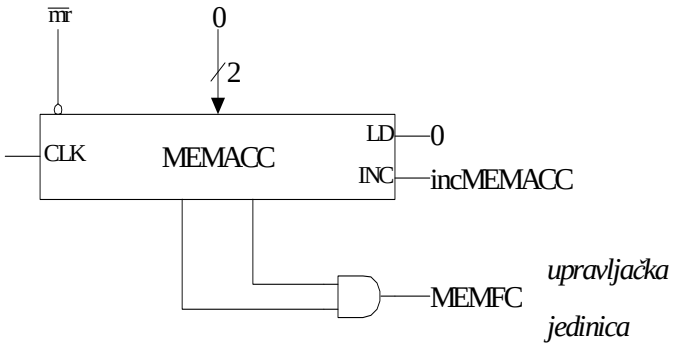


Slika 7 *SAJUP/MEMORIJA interfejs*

Postoji jedan slučaj upisa u tabelu PMT i to kada je niža reč deskriptora pročitana da bi se utisnula jedinica na poziciji I-bita. To se realizuje preko multipleksera na čijem jednom ulazu su najviša dva bita 11 a najnižih 14 su 14 najmlađih razreda registra MDRL. Taj ulaz se propušta kroz multiplekser kada je aktivan signal mxMDRL.

2.3.4.3.3.1. Brojač MEMACC

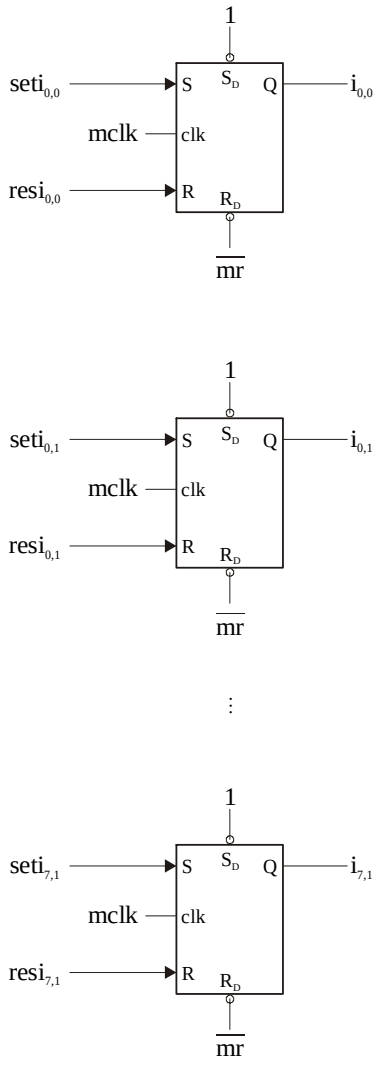
Brojač vremena pristupa operativnoj memoriji. MEMACC se koristi kod čitanja ili upisa u memoriju MEM da odbroji četiri signala takta CLK. Usvojeno je da toliko iznosi vreme pristupa memoriji. Signal MEMFC (*MEMory Function Completed*) postaje aktivan kada brojač MEMACC pređe u stanje tri i koristi se kao signal logičkog uslova da je pristup memoriji završen. Generisanje i korišćenje ovog signala prikazano je na slici 8. Uzeto je da se korak $step_i$ koristi za pristup memoriji. Ulaskom u ovaj korak kreće se sa odbrojavanjem četiri signala takta CLK i ostaje u koraku $step_i$. Posle trećeg signala takta CLK signal MEMFC postaje aktivan. Na četvrti signal takta CLK brojač MEMACC se vraća na stanje nula, signal MEMFC postaje neaktivan i prelazi se na korak $step_{i+1}$.



slika 8 *BROJAČ MEMACC*

2.3.4.3.4. Pamćenje izmene

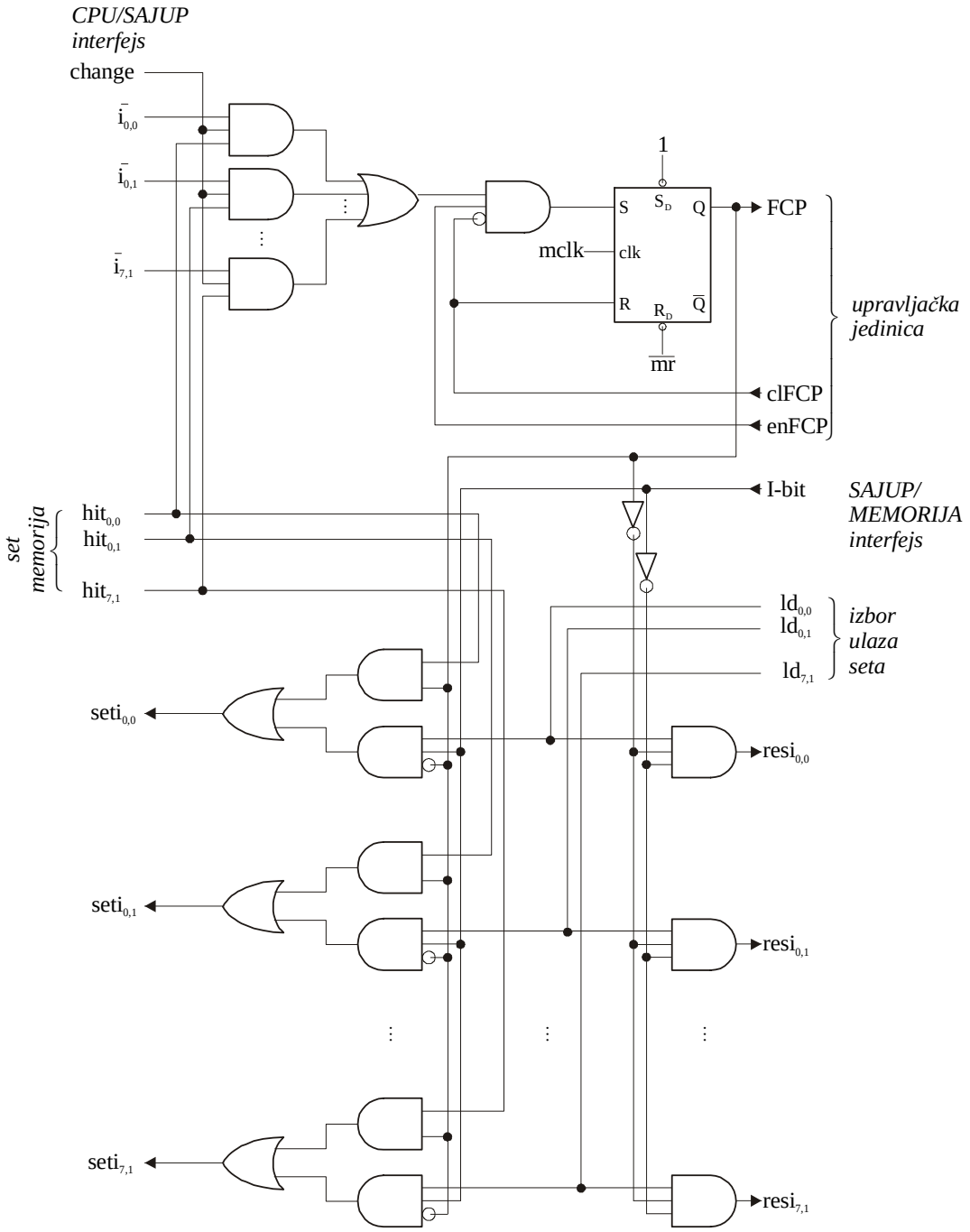
Blok pamćenje izmene se sastoji iz flip-flopova $i_{i,0}$, $i_{i,1}$, i = 0, …, 7, (slika 8) i logike za ažuriranje flip-flopova i (slika 9).



Slika 9 *Flip-flopovi I*

Flip-flopovi $i_{i,0}$, $i_{i,1}$, i = 0, …, 7, (slika 9) označavaju da je bilo upisa u stranicu čiji se deskriptor nalazi u odgovarajućem ulazu set memorije SM. Kada se deskriptor stranice dovlači iz tabele PMT u set memoriju SM postavlja se odgovarajući flip-flop i to na vrednost koja odgovara vrednosti I-bita koji se nalazi u deskriptoru u tabeli PMT. Ako je I-bit jedan, znači da je to obraćanje stranici koja je već menjana, a ako je I-bit nula, znači da je to obraćanje stranici koja prethodno nije menjana.

Kada se procesor obrati **jedinici SAJUP** sa namerom da u neku stranicu, čiji se deskriptor nalazi u set memoriji SM, nešto upiše, mogu da se jave dva slučaja u zavisnosti od toga koja je vrednost odgovarajućeg flip-flopa i. Ako je on na vrednosti 0, to znači da je ovo prvo obraćanje toj stranici sa zahtevom da se u nju nešto upiše. Tada se ne samo postavlja taj flip-flop I na jedan, već **jedinica SAJUP** obavlja postavljanje I-bita u odgovarajućem deskriptoru tabele PMT na vrednost 1. Ako je on na vrednosti 1, to znači da je ovo obraćanje stranici u koju je već nešto upisivano. Stoga je i u tabeli PMT odgovarajući I-bit na vrednosti 1, pa nema potrebe da **jedinica SAJUP** vrši njegovo ažuriranje.



Slika 10 *Logika za ažuriranja flip-flopova I*

Logika za ažuriranje flip-flopova i (slika 10) služi da postavi na vrednost 1 ili 0 odgovarajući flip-flop i. Tom prilikom mogu da se jave dva slučaja.

Prvi slučaj je kada je procesor generisao virtuelnu adresu na koju želi nešto da upiše, zbog čega je signal **change** na vrednosti 1, kada je pronađena saglasnost sa nekim ulazom u set memoriji SM, zbog čega je signal **hit** tog ulaza na jedinici, i kada je odgovarajući flip-flop i 0. To znači da je ovo prvo obraćanje toj stranici sa zahtevom za upis i da to je prva izmena u okviru stranice. Prva izmena u okviru adresirane stranice pamti se u flip-flopu FCP (*First*

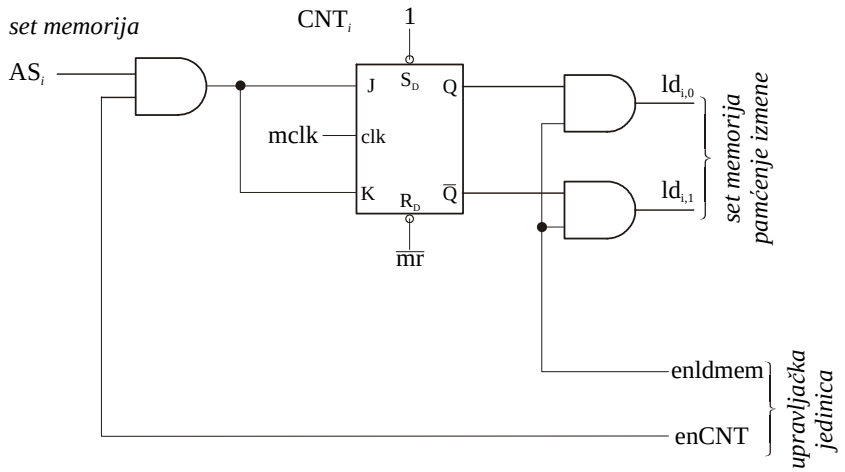
Change Page), kada upravljačka jedinica generiše aktivnu vrednost signala **enFCP** (gornja polovina slike 9). Signal **FCP** i odgovarajući signali $\mathbf{hit}_{i,0}$ ili $\mathbf{hit}_{i,1}$, i = 0, …, 7, generišu signale $\mathbf{set}_{i,0}$ i $\mathbf{set}_{i,1}$, i = 0, …, 7 (donja polovina slike 9), koji postavljaju odgovarajući flip-flop i. Nakon toga signalom **clFCP** flip-flop FCP se postavlja na vrednost 0.

Drugi slučaj je kada je procesor generisao neku virtuelnu adresu, kada nije pronađena saglasnost ni sa jednim ulazom set memorije SM i kada se stranica nalazi u operativnoj memoriji. Tada se deskriptor stranice iz tabele PMT dovlači u **jedinicu SAJUP** i na osnovu I-bita deskriptora postavlja se izabrani flip-flop i na vrednost 1 ili 0. Taj flip-flop i se bira na osnovu izabranog ulaza za upis deskriptora u set memoriju SM, a određen je aktivnom vrednošću odgovarajućeg signala $\mathbf{ld}_{i,0}$ ili $\mathbf{ld}_{i,1}$, i = 0, …, 7 (slika 9). Na osnovu signala $\mathbf{ld}_{i,0}$ ili $\mathbf{ld}_{i,1}$, i = 0, …, 7, generišu se odgovarajući signali $\mathbf{set}_{i,0}$ ili $\mathbf{set}_{i,1}$ ili $\mathbf{res}_{i,0}$ ili $\mathbf{res}_{i,1}$, i = 0, …, 7, (donja polovina slike 9).

2.3.4.3.5. Izbor ulaza seta

Blok izbor ulaza seta služi za izbor jednog od 8 setova i jednog od 2 ulaza u setu u koji će se upisati deskriptor, kao i za realizaciju algoritma zamene unutar svakog od 8 setova. Signali selekcije, i to po dva za svaki od 8 setova, za upis deskriptora su $\mathbf{ld}_{i,0}$, $\mathbf{ld}_{i,1}$, i = 0, …, 7. Međutim, ako je virtuelna adresa selektovanog seta set_i, samo je signal selekcije seta $\mathbf{AS}_i$ jedinica i samo jedan od dva signala $\mathbf{ld}_{i,0}$, $\mathbf{ld}_{i,1}$ može da bude jedinica.

Blok izbor ulaza seta se sastoji od 8 identičnih celina, a struktura za set set_i je data na slici 10. Unutar svakog od njih postoji brojač po modulu 2 označen sa CNT. U zavisnosti od njegove vrednosti zavisi koji će od signala $\mathbf{ld}_{i,0}$, $\mathbf{ld}_{i,1}$ seta set_i selektovanog signalom selekcije $\mathbf{AS}_i$ biti jedinica kada upravljačka jedninca generiše aktivnu vrednost signala **enldmem**. Ažuriranje brojača CNT se vrši pri generisanju aktivne vrednosti signala **enCNT** samo za set selektovan signalom selekcije seta $\mathbf{AS}_i$. Ažuriranje brojača se vrši samo pri upisu novog deskriptora, što se dešava kada je signal **enldmem** jedinica.



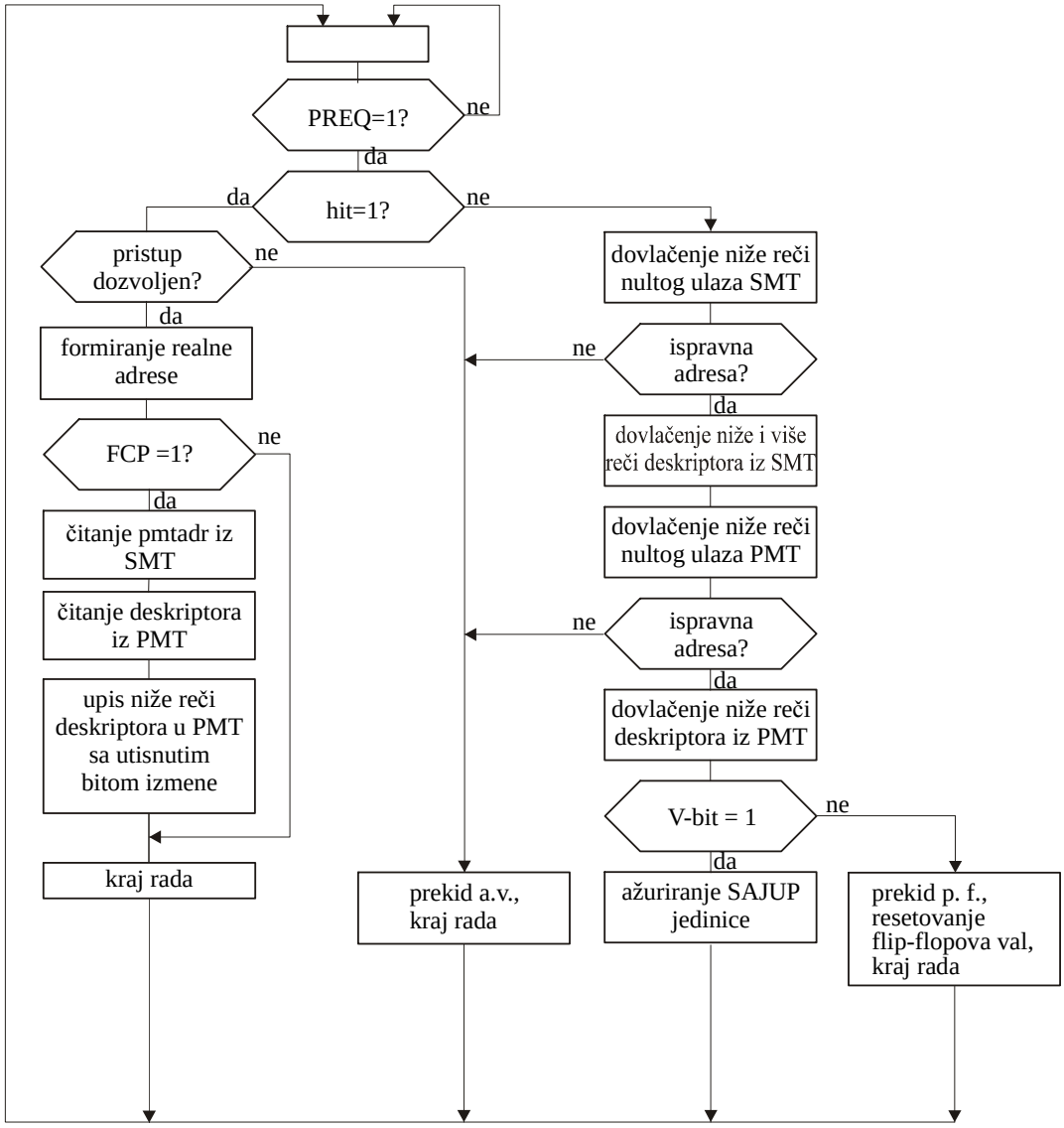
Slika 11 *Deo bloka izbora ulaza seta za set set_I*

2.3.4.4. UPRAVLJAČKA SAJUP JEDINICA

Upravljačka jedinica služi za generisanje upravljačkih signala prema algoritmu set-asocijativnog preslikavanja virtuelne adrese u realnu. U ovom poglavlju će biti prikazani i objašnjeni dijagram toka operacija, algoritam generisanja upravljačkih signala i struktura upravljačke jedinice.

2.3.4.4.1. Dijagram toka operacija

Dijagram toka operacija preslikavanja virtuelne adrese u realnu dat je na slici 12. On je kombinacija dijagrama toka operacija za **jedinicu ASOJUP** i **jedinicu DIRJUP** pa ovde neće biti detaljnije objašnjavan.



Slika 12 *Dijagram toka operacija*

2.3.4.4.2. ALGORITAM GENERISANJA UPRAVLJAČKIH SIGNALA

Na osnovu dijagrama toka operacija (slika 12) i strukture operacione jedinice SAJUP formiran je algoritam generisanja upravljačkih signala operacione jedinice. Za svaki korak je data simbolička oznaka samog koraka, spisak upravljačkih signala operacione jedinice koji se generišu bezuslovno i uslovno i korak na koji treba preći. Notacija koja se koristi je sledeća:

<korak>: <bezuslovni_signal> <uslovni_signal> <sledeći_korak>

pri čemu su:

<bezuslovni_signal> := <signal>, {<signal>,} | <prazno>

<uslovni_signal> := *if* (<uslov>, <signal>, {signal,}) | <prazno>

< sledeći_korak > := *br* <korak> |
br (*if* <uslov>*then* <korak> *else* <korak>) |
br (*if* <uslov>*then* <korak>*else* (*if* <uslov> *then* <korak> *else* <korak>))

<korak> := simboličke oznake za korake od 0 do 18 step0 do step18

<signal> := svi signali operacione jedinice koje generiše upravljačka jedinica.

<uslov> := izrazi formirani od signala logičkih uslova koje generiše upravljačka jedinica

<prazno> :=

Algoritam generisanja upravljačkih signala je dat u daljem tekstu.

step0: *br* (*if* **PREQ** *then step1 else step0)*

! U koraku step0 se čeka da iz procesora stigne signal zahteva za preslikavawe virtuelne u realnu adresu **PREQ.** Ako se pojavi signal **PREQ** na signal takta **CLK** virtuelna adresa se upisuje u registar UVAR, broj segmenta u registar USR, broj stranice u registar UPR, oznaka tekućeg korisnika se upisuje u registar UUSR, adresa početka tabele segmenata tekućeg korisnika se upisuje u registar USMTAR, tip operacije upisuje u flip-flop UR/WF i upisuje se podatak u flip-flop UE/FF. Ako se pojavi signal **PREQ** na signal takta **CLK** se prelazi na korak step1. U suprotnom slučaju se ostaje u koraku step0.

step1: *if* (**HIT*** $\overline{\textbf{a.v.1}}$, **ldURAR, enFCP**), **incUSR**, **incUPR**

if(☐,**ldMAR,mxMAR**)

*if(***HIT*a.v.1, AV)**

br (*if* ☐ *then step9 else (if* $\overline{\textbf{a.v.1}}$ *then step2 else step18*)

! U korak step1 može da se dođe ili iz koraka step0 ili iz koraka step17. Iz koraka step0 se dolazi po svakom novom zahtevu za preslikavanje adrese. Iz koraka step17 se dolazi po nekom zahtevu za preslikavanje adrese za koji je pri prethodnom prolasku kroz step1 otkriveno da nema saglasnosti, pa se išlo na dovlačenje deskriptora stranice iz operativne memorije u SAJUP jedinicu. U koraku step1 se vrši provera saglasnosti i na osnovu toga formira vrednost signala **HIT**. Ako se pojavi aktivna vrednost signala **HIT**, to znači da je otkrivena saglasnost. Osim toga vrši se provera prava pristupa. Ukoliko pristup nije dozvoljen generiše se aktivna vrednost signala **a.v.1.** U slučaju da je pristup dozvoljen (**a.v.1**=0**)** i da je otkrivena saglasnost (**hit**=1**)**, generiše se aktivna vrednost signala **ldURAR**, pa se na signal takta **CLK** realna adresa, koja se dobija konkatenacijom broja bloka iz set memorije SAJUP jedinice i adrese unutar bloka iz virtuelne adrese, upisuje u registar URAR. Za isti slučaj se generiše i signal **enFCP** koji treba da aktivira setovanje flip-flopa FCP na sledeći signal takta **CLK**, u slučaju da se radi o prvom upisu za tekući deskriptor. Ako signal **HIT** ima neaktivnu vrednost tj. ako nema saglasnosti, generišu se aktivne vrednosti signala **ldMAR** i **mxMAR.** Aktivnom vrednošću signala **mxMAR**, u bloku SAJUP/MEM interfejs, kroz multiplekser se propušta adresa nultog ulaza u tabelu segmenata. Aktivnom vrednošću signala **ldMAR** se na signal takta **CLK** u registar **MAR** upisuje adresa nultog ulaza tabele segmenata. U koraku step1

takođe se generišu i aktivne vrednosti signala **incUSR** i **incUPR** čime se na signal takta **CLK** za jedan uveća broj segmenta koji se nalazi u registru USR i broj stranice koji se nalazi u registru UPR. Ovo se radi zbog kasnijih određivanja tačne adrese. U slučaju da je otkrivena saglasnost, a da ne postoji pravo pristupa, generiše se signal **AV**, koji služi za sinhronizaciju između SAJUP jedinice i procesora i treba da ukaže procesoru da preslikavanje nije uspešno završeno i da je došlo do greške *Address Violation.* Ako se pojavi neaktivna vrednost signala **HIT**, na signal takta **CLK** se prelazi na korak step9. U suprotnom slučaju ako je vrednost signala **a.v.1** neaktivna prelazi se na korak step2 a za aktivnu vrednost signala **a.v.1** na korak step18.

Step2: *if*($\overline{\textbf{FCP}}$,**URP**,**clR/W,clE/F**),

if (**FCP**, **ldMAR**),

br (*if* **FCP** *then* step3 *else step0*)

! U korak step2 može da se dođe jedino iz koraka step1 i to samo onda kada je u koraku step2 otkriveno da je pristup dozvoljen i da je adresa ispravna. Ukoliko je aktivna vrednost signala **FCP** to znači da se radi o operaciji upisa, i to o prvoj takvoj operaciji na tekućoj stranici. Potrebno je iz operativne memorije učitati deskriptor stranice i u njega utisnuti bit izmene. Generiše se aktivna vrednost signala **ldMAR**. Neaktivnom vrednošću signala **mxMAR**, u bloku SAJUP/MEM interfejs, kroz multiplekser se propušta adresa ulaza u tabelu segmenata formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** ta adresa se na signal takta **CLK** upisuje u registar MAR. U slučaju da je signal **FCP** na neaktivnoj vrednosti, generišu se aktivne vrednosti signala **URP, clR/W** i **clE/F.** Aktivnom vrednošću signala **URP** SAJUP jedinica daje procesoru indikaciju o kraju rada, a aktivnim vrednostima signala **clR/W** i **clE/F** resetuju se flip-flopovi UR/WF i UE/FF. Na signal takta **CLK** se prelazi iz koraka step2 u korak step3 ako je signal **FCP** na aktivnoj vrednosti, odnosno u korak step0 ako je signal **FCP** na neaktivnoj vrednosti.

Step3: **incMEMACC**, *if* (**MEMFC**, **ldMDRL, incMAR**),

br (*if* **MEMFC** *then step4 else step3*)

! U korak step3 se dolazi samo iz koraka step2. U koraku step3 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRL**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRL upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju prva reč deskriptora tekućeg segmenta, i signala **incMAR**, čime se obezbeđuje da se na sledeći signal takta **CLK** inkrementira sadržaj registra MAR, tako da će u registru MAR biti adresa druge reči deskriptora tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step3 u korak step4. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step3.

Step4: **incMEMACC**, *if* (**MEMFC**, **ldMDRH**),

br (*if* **MEMFC** *then step5 else step4*)

! U korak step4 se dolazi samo iz koraka step3. U koraku step4 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRH**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRH upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju druga reč deskriptora tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step4 u korak step5. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step4.

Step5: **mxADDER, ldMAR,** *br step6*

! U korak step5 se dolazi samo iz koraka step4. U koraku step5 se bezuslovno generišu signali **mxADDER** i **ldMAR**. Aktivnom vrednošću signala **mxADDER** na ulaze sabirača se propušta početna adresa PMT-a s' jedne strane, odnosno sadržaj registra UPR (inkrementirana vrednost broja stranice) sa druge strane. Na taj način se na izlazu sabirača dobija adresa u memoriji deskriptora stranice, i ona se zahvaljujući neaktivnom signalu **mxMAR** propušta na ulaz registra MAR, u koji će zahvaljujući aktivnom signalu **ldMAR** biti upisana na signal takta **CLK**. Na signal takta **CLK** se uvek prelazi u korak step6.

Step6: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then step7 else step6*)

! U korak step6 se dolazi samo iz koraka step5. U koraku step6 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRL**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRL upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju prva reč deskriptora tekuće stranice. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step6 u korak step7. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step6.

Step7: **mxMDRL**, **ldMDRL**, *br step8*

! U korak step7 se dolazi samo iz koraka step6. U registru MDRL nalazi se učitan deskriptor stranice u koji je potrebno utisnuti bit izmene. Bezuslovno se generišu aktivne vednosti signala **mxMDRL** i **ldMDRL**. Aktivnom vrednošću signala **mxMDRL** kroz multiplekser u bloku SAJUP/MEM interfejs se propušta 16-bitna reč sastavljena od 14 nižih bitova registra MDRL i dve jedinice na dve najviše pozicije. Aktivna vrednost signala **ldMDRL** omogućuje da se na signal takta **CLK** ova vrednost upiše u registar MDRL čime će najviša dva bita (V i I biti u deskriptoru) biti postavljeni na jedan.. Na signal takta **CLK** se uvek prelazi iz koraka step7 u korak step8.

Step8: **writeMEM**, **incMEMACC**, *if* (**MEMFC**, **URP**, **clR/W, clE/F, clFCP**), *br* (*if* **MEMFC** *then step0 else step8*)

! U korak step8 se dolazi samo iz koraka step7. U koraku step8 se bezuslovno generišu aktivne vrednosti signala **writeMEM** i **incMEMACC**. Aktivnom vrednošću signala **writeMEM** se sadržaj registra MDRL upisuje u memoriju MEM na adresi određenoj sadržajem registra MAR. Aktivnom vrednošću signala **incMEMACC** se obezbeđuje da se pri pojavi signala takta **CLK** inkrementira sadržaj brojača MEMACC koji određuje vreme pristupa memoriji MEM. Aktivnom vrednošću signala **MEMFC** pri pojavi signala takta **CLK** završava se rad - generišu se aktivne vrednosti signala **URP** i **clR/W**. Aktivnom vrednošću signala **URP** daje indikacija o kraju rada, a aktivnom vrednošću signala **clR/W, clE/F** i **clFCP** resetuju se flip-flop-ovi UR/WF, UE/FF i FCP. Pri aktivnoj vrednosti signala **MEMFC** na signal takta **CLK** se prelazi na korak step0. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step8.

Step9: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then step10 else step9*)

! U korak step9 se dolazi iz koraka step1. Iz koraka step1 se dolazi kada se pri otkrivenoj nesaglasnosti odmah prelazi na dovlačenje deskriptora tekućeg segmenta iz operativne memorije u SAJUP jedinicu. U koraku step9 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRL**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRL upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju prva reč nultog ulaza tabele segmenata tekućeg korisnika. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step9 u korak step10. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step9.

Step10: *if* (**a.v.2**, **AV**),

if ($\overline{\textbf{a.v.2}}$, **ldMAR**),

br (*if* **a.v.2** *then step18 else step11*)

! U korak step10 se dolazi samo iz koraka step9. U koraku step10 se ispituje ispravnost virtuelne adrese tj. da li je broj segmenta u dozvoljenom opsegu za tekućeg korisnika. Signal **a.v.2** predstavqa izlaz iz komparatora u kojem se porede broj segmenta i niža reč nultog ulaza tabele segmenata gde je upisan ukupan broj segmenata korisnika. Ako je signal **a.v.2** na aktivnoj vrednosti to znači da je došlo do greške, pa se generiše aktivna vrednost signala **AV** kojom se postavlja odgovarajući indikator u procesoru. Ako je siganal **AV** na neaktivnoj vrednosti generiše se aktivna vrednost signala **ldMAR**. Neaktivnom vrednošću signala **mxMAR** se kroz odgovarajući multiplekser u bloku SAJUP/MEM interfejs propušta adresa ulaza u tabelu segmenata formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** se obezbeđuje da se na signal takta **CLK** ta adresa upiše u registar MAR. Na signal takta **CLK** se prelazi iz koraka step10 u korak step11 ukoliko je adresa ispravna tj. signal **a.v.2** je na neaktivnoj vrednosti. U suprotnom prelazi se u korak step18.

Step11: **incMEMACC**, *if* (**MEMFC**, **ldMDRL, incMAR**),

br (*if* **MEMFC** *then step12 else step11*)

! U korak step11 se dolazi samo iz koraka step10. U koraku step11 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRL**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRL upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju prva reč deskriptora tekućeg segmenta, i aktivna vrednost signala **incMAR**, čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj registra MAR, koji će tada da ukazuje na drugu reč deskriptora segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step11 u korak step12. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step11.

Step12: **incMEMACC**, *if* (**MEMFC**, **ldMDRH**),

br (*if* **MEMFC** *then step13 else step12*)

! U korak step12 se dolazi samo iz koraka step11. U koraku step12 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRH**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRH upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju druga reč deskriptora tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step12 u korak step13. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step12.

Step13: **ldMAR**, **mxMAR**, *br step14*

! U korak step13 se dolazi samo iz koraka step12. U ovom koraku bezuslovno se generišu aktivne vrednosti signala **ldMAR** i **mxMAR.** Aktivna vrednost signala **mxMAR** omogućava prolazak kroz odgovarajući multiplekser u SAJUP/MEM interfejsu, izlaz levog multipleksera, što je zahvaljujući neaktivnom signalu **mxADDER,** početna adresa PMT-a. Aktivna vrednost signala **ldMAR** omogućuje da se na signal takta **CLK** ova vrednost upiše u registar MAR. Na signal takta **CLK** se uvek prelazi iz koraka step13 u korak step14.

Step14: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then step15 else step14*)

! U korak step14 se dolazi samo iz koraka step13. U koraku step14 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRL**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRL upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju prva reč nultog ulaza tabele stranica tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal

takta **CLK** prelazi iz koraka step14 u korak step15. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step14.

Step15: *if* (**a.v.3**, **AV**),

if ($\overline{\textbf{a.v.3}}$, **mxADDER, ldMAR**),

br (*if* **a.v.3** *then step18 else step16*)

! U korak step15 se dolazi samo iz koraka step14. U koraku step15 se ispituje ispravnost virtuelne adrese tj. da li je broj stranice tekućeg segmenta u dozvoljenom opsegu za tekući segment. Signal **a.v.3** predstavlja izlaz iz komparatora u kojem se porede broj stranice i niža reč nultog ulaza tabele stranica gde je upisan ukupan broj stranica datog segmenta. Ako je signal **a.v.3** na aktivnoj vrednosti to znači da je došlo do greške, pa se generiše aktivna vrednost signala **AV** kojom se postavlja odgovarajući indikator u procesoru. Ako je signal **AV** na neaktivnoj vrednosti generiše se aktivna vrednost signala **ldMAR i mxADDER**. Neaktivnom vrednošću signala **mxMAR** se kroz odgovarajući multiplekser u bloku SAJUP/MEM interfejs propušta adresa ulaza u tabelu stranica formirana u sabiraču. Aktivnom vrednošću signala **ldMAR** se obezbeđuje da se na signal takta **CLK** ta adresa upiše u registar MAR. Signal **mxADDER** je aktivan kako bi se na ulaze sabirača propustili početna adresa PMT i sadržaj registra UPR (broj stranice+1). Na signal takta **CLK** se prelazi iz koraka step15 u korak step16 ukoliko je adresa ispravna tj. signal **a.v.3** je na neaktivnoj vrednosti. U suprotnom prelazi se u korak step18.

Step16: **incMEMACC**, *if* (**MEMFC**, **ldMDRL**),

br (*if* **MEMFC** *then step17 else step16*)

! U korak step16 se dolazi samo iz koraka step15. U koraku step16 se bezuslovno generiše aktivna vrednost signala **incMEMACC** čime se obezbeđuje da se na signal takta **CLK** inkrementira sadržaj brojača **MEMACC**, koji određuje vreme pristupa memoriji MEM. Pri aktivnoj vrednosti signala **MEMFC** generiše se aktivna vrednost signala **ldMDRL**, čime se obezbeđuje da se na signal takta **CLK** u registar MDRL upiše vrednost očitana iz memorije MEM sa adrese određene sadržajem registra MAR, što je u ovom slučaju prva reč deskriptora tekuće stranice tekućeg segmenta. Pri aktivnoj vrednosti signala **MEMFC** se na signal takta **CLK** prelazi iz koraka step16 u korak step17. Pri neaktivnoj vrednosti signala **MEMFC** se ostaje u koraku step16.

Step17: *if* (**V**, **endldmem, enCNT, decUPR, decUSR)**,

if ($\overline{\textbf{V}}$, **PF, resval**),

br (*if* **V** *then step1 else step18*)

! U korak step17 se dolazi samo iz koraka step16. U koraku step17 se ispituje da li se tekuća stranica nalazi u operativnoj memoriji. Signal **V** predstavlja vrednost V-bita deskriptora stranice. Ako ima aktivnu vrednost, stranica je u memoriji i generiše se aktivna vrednost signala **endldmem, enCNT, decUPR** i **decUSR**. Aktivna vrednost signala **enCNT** omogućuva izbor ulaza seta u koji će deskriptor biti dovučen a signal **endldmem** omogučava setovanje odgovarajućeg $\mathbf{ld_{i,j}}$ signala i

ažuriranje SAJUP jedinice. Aktivnom vrednošću signala decUPR i decUSR, dekrementiraju se sadržaji registara USR i UPR, kako bi se anuliralo drugo inkrementiranje tih registara u sledećem prolasku. Na signal takta **CLK** prelazi se iz koraka step17 u korak step1 ako je signal **V** na aktivnoj vrednosti, odnosno u korak step18 ako je signal **valid** na neaktivnoj vrednosti.

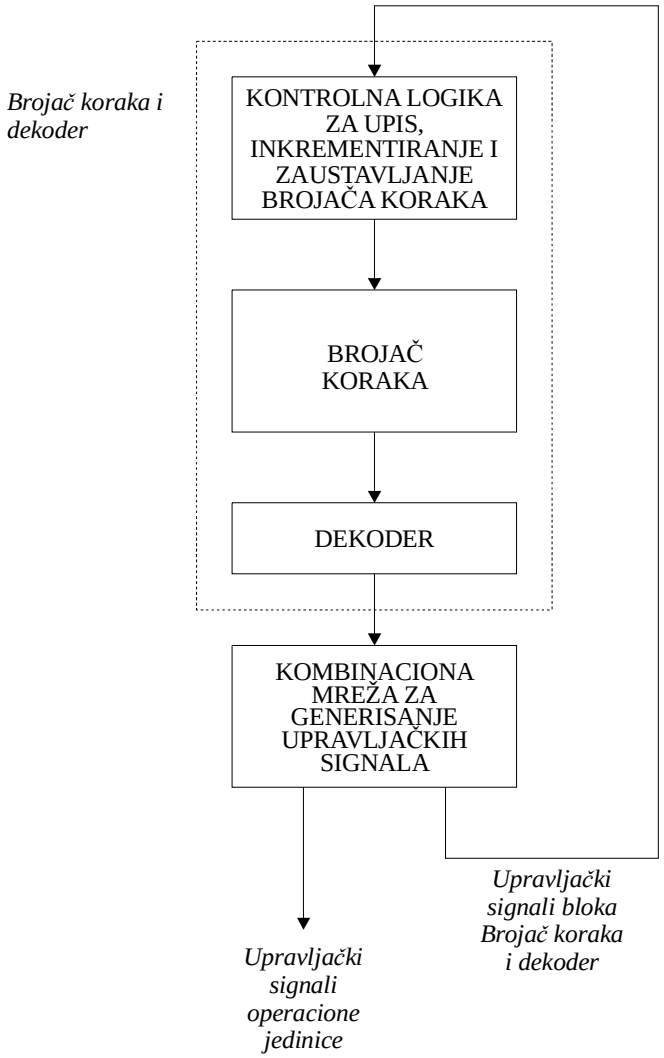
step18: **URP**, **clR/W**, **clE/F, clFCP** *br* step0

! U korak step18 se dolazi ili iz koraka step1,step 10,step15 ili iz koraka step17. U sva četiri slučaja došlo je do neuspešnog okončavanja preslikavanja. Indikatori koji govore o tipu problema su postavljeni u ranijim koracima, pa se u koraku step18 samo daje indikacija o kraju rada postavljanjem na aktivnu vrednost signala **URP**, i resetuju UR/WF,UE-FF i FCP flip-flop-ovi. postavljanjem na aktivnu vrednost signala **clUR/WF, clE/F** i **clFCP**. Na signal takta **CLK** uvek se prelazi iz koraka step18 u step0.

2.3.4.4.2.1. Struktura upravljačke jedinice

Upravljačka jedinica sastoji se iz dva bloka kao što je prikazano na slici 12 i to:

- **brojač koraka i dekoder** i
- **kombinaciona mreža za generisanje upravljačkih signala**.



Slika 13 *Principijelna šema upravljačke jedinice*

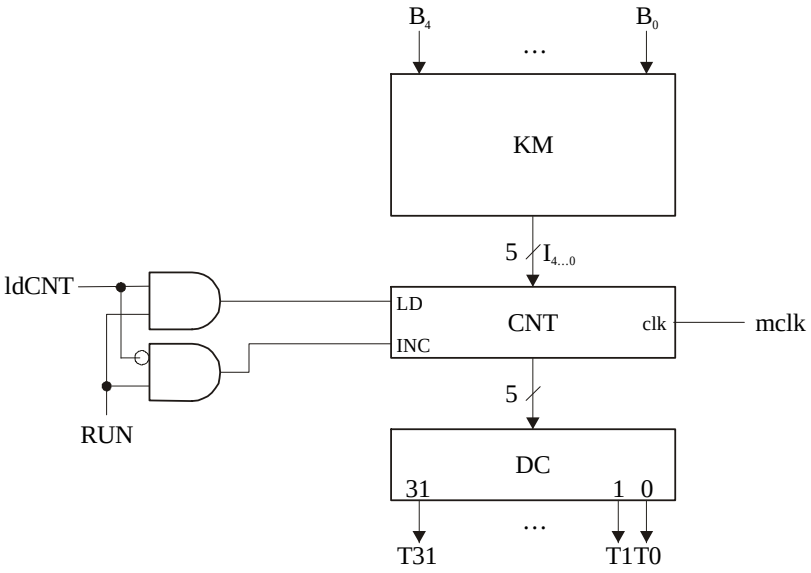
2.3.4.4.2.2. Brojač koraka i dekoder

Blok **brojač koraka i dekoder** je prikazan na slici 13 i ima istu strukturu i funkcioniše na identičan način kao odgovarajući blok jedinice sa asocijativnim preslikavanjem (odeljak **Error! Reference source not found.**). Blok KM je realizovan kao na slici 14 a vrednosti koje treba upisati u brojač CNT nalaze se na adresama EPROM-a kao što je dato na slici 15.

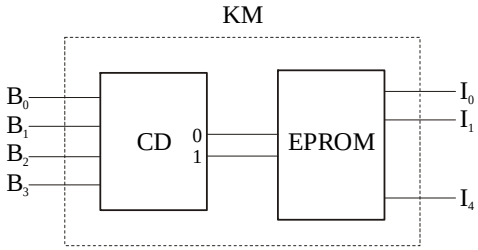
Dekoder DC na izlazima u svakom takt-intervalu generiše samo jedan od signala **T0**–**T18** pri čemu se signali **T19**–**T31** ne koriste. Ovi signali služe za generisanje upravljačkih signala u kombinacionoj mreži.

2.3.4.4.2.3. Kombinaciona mreža za generisanje upravljačkih signala

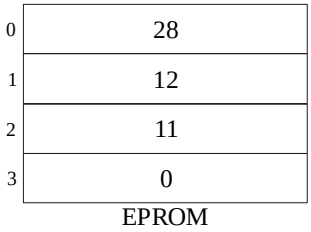
Svi upravljački signali su podeljeni u dve grupe u zavisnosti od toga da li se koriste u upravljačkoj ili operacionoj jedinici.



Slika 14 *Brojač koraka i dekoder*



Slika 15 *Blok KM*



Slika 16 *EPROM*

2.3.4.4.2.3.1. Upravljački signali upravljačke jedinice

Upravljački signali upravljačke jedinice su:

- **$B_{0...3}$** signali koji definišu adresu EPROM-a,
- **ldCNT** signal upisa u brojač CNT,
- **RUN** signal dozvole brojanja ili upisa brojača CNT,

Ovi signali se generišu na sledeći način:

B_0 = **T1 * hit * a.v.1 + T10 * a.v.2 + T15 * a.v.3**

B_1 = **T1 * $\overline{\text{hit}}$**

B_2 = **T17* V**

B_3 = **T8*MEMFC+T18** + **T2*** $\overline{\textbf{FCP}}$

ldCNT = $B_0 + B_1 + B_2 + B_3$

RUN = **T0*PREQ + T1 + T2 + (T3+ T4 + T6 + T8 + T9 + T11 + T12 + T14 + T16)*MEMFC + T5+ T7 + T10 + T13 + T15 + T17 + T18**

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz pojedinih blokova operacione jedinice i to:

hit - set memorija,

a.v.1 - set memorija,

a.v.2 - SAJUP/MEMORIJA interfejs,

a.v.3 - SAJUP/MEMORIJA interfejs,

FCP - Pamćenje izmene,

V-bit - SAJUP/MEMORIJA interfejs,

Kombinaciona mreža koja generiše upravljačke signale ima sličnu strukturu kao odgovarajuća kombinaciona mreža u upravljačkoj jedinici sa asocijativnim preslikavanjem (slika 14).

2.3.4.4.2.3.2. Upravljački signali operacione jedinice

Upravljački signali operacione jedinice podeljeni su u grupe koje odgovaraju blokovima operacione jedinice u kojima se koriste i to **CPU/SAJUP interfejs**, **set memorija**, **SAJUP/MEMORIJA interfejs**, **pamćenje izmene** i **izbor ulaza seta**. Kombinacione mreže za generisanje ovih signala su trivijalne, pa su stoga dati samo izrazi prema kojima se signali generišu.

2.3.4.4.2.3.2.1. Upravljački signali bloka CPU/SAJUP interfejs

Upravljački signali koji odgovaraju ovom bloku operacione jedinice su:

- **ClE/F –** resetovanje flip-flop-a UE/FF
- **ClR/W -** resetovanje flip-flop-a UR/WF,
- **AV -** prekid zbog nedozvoljenog pristupa i
- **PF -** prekid jer stranica nije u operativnoj memoriji.
- **URP –** signal procesoru da je preslikavanje završeno (uspešno ili ne)

Ovi upravljački signali se generišu na sledeći način:

ClE/F = **T2*** $\overline{\textbf{FCP}}$ + **T8*MEMFC** + **T18**

ClR/W = **T2*** $\overline{\textbf{FCP}}$ + **T8*MEMFC** + **T18**

AV = **T1*hit*a.v.1 + T10*a.v.2 + T15*a.v.3**

PF = **T17*** $\overline{\textbf{V}}$

URP = **T2*** $\overline{\textbf{FCP}}$ **+ T8*MEMFC + T18**

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz pojedinih blokova operacione jedinice i to:

hit - set memorija,

a.v.1 - set memorija,

a.v.2 - SAJUP/MEMORIJA interfejs,

a.v.3 - SAJUP/MEMORIJA interfejs,

FCP - Pamćenje izmene,

V-bit—SAJUP/MEMORIJA interfejs,

2.3.4.4.2.3.2.2. Upravljački signali bloka set memorija

Upravljački signali koji odgovaraju ovom bloku operacione jedinice je

- **resval -** reset V flip-flopova.

Ovaj upravljački signal se generiše na sledeći način:

resval = **T17*** $\overline{\textbf{V}}$

Pri njegovom generisanju koristi se:

V-bit—SAJUP/MEMORIJA interfejs,

2.3.4.4.2.3.2.3. Upravljački signali bloka SAJUP/MEMORIJA interfejs

Upravljački signali koji odgovaraju ovom bloku operacione jedinice su:

- **mxMAR -** računanje adrese za pristup nenultom ulazu tabele PMT ili SMT,
- **mxADDER -** pristup PMT-u(mxADDER=1) ili SMT-u(mxADDER=0),
- **incUSR -** inkrementiranje registra USR,
- **decUSR** - dekrementiranje registra USR,
- **incUPR -** inkrementiranje registra UPR,
- **decUPR** - dekrementiranje registra UPR,
- **ldMAR -** upis u registar MAR,
- **incMAR -** inkrementiranje registra MAR,
- **ldMDRL -** upis u registar MDRL,
- **ldMDRH -** upis u registar MDRH,
- **writeMEM -** zahtev za upis u operativnu memoriju,
- **incMEMACC -** inkrementiranje brojača MEMACC

Ovi upravljački signali se generišu na sledeći način:

mxMAR = **T1*** $\overline{\textbf{hit}}$ **+ T13**

mxADDER = **T13 + T15*** $\overline{\textbf{a.v.3}}$

incUSR = **T1**

decUSR = **T17*V**

incUPR = **T1**

decUPR = **T17*V**

ldMAR = **T1*** $\overline{\textbf{hit}}$ **+ T2*FCP + T5 + T10*** $\overline{\textbf{a.v.2}}$ **+ T13 + T15*** $\overline{\textbf{a.v.3}}$

ldMDRL = **T3*MEMFC + T6*MEMFC + T7 + T9*MEMFC + T11*MEMFC + T14*MEMFC +T16*MEMFC**

ldMDRH = **T4*MEMFC + T12*MEMFC**

writeMEM = **T8**

incMEMACC= T3 + T4 + T6 + T8 + T9 + T11 + T12 + T14 + T16

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz pojedinih blokova operacione jedinice i to:

hit - set memorija,

MEMFC - SAJUP/MEMORIJA interfejs,

a.v.2 - SAJUP/MEMORIJA interfejs,

a.v.3 - SAJUP/MEMORIJA interfejs,

V-bit—SAJUP/MEMORIJA interfejs,

2.3.4.4.2.3.2.4. **Upravljački signali bloka** pamćenje izmene

Upravljački signali koji odgovaraju ovom bloku operacione jedinice su:

- **clFCP**—resetovanje FCP flip-flopa i
- **enFCP**—setovanje FCP flip-flopa.

Ovi upravljački signali se generišu na sledeći način:

clFCP = **T8*MEMFC** + **T18**

enFCP = **T1** * **hit** * $\overline{\textbf{a.v.1.}}$

Pri njihovom generisanju koriste se sledeći signali logičkih uslova koji dolaze iz pojedinih blokova operacione jedinice i to:

hit - set memorija,

a.v.1 - set memorija,

MEMFC - SAJUP/MEMORIJA interfejs,

2.3.4.4.2.3.2.5. **Upravljački signali bloka** izbor ulaza seta

Upravljački signali koji odgovaraju ovom bloku operacione jedinice su:

- **enldmem**—dozvola za upisivanje u set memoriju i
- **enCNT**—dozvola brojačima da promene stanje.

Ovi upravljački signali se generišu na sledeći način:

enldmem = **T17*V**

enCNT = **T17*V**

Pri njihovom generisanju koristi se:

V-bit—**SAJUP/MEMORIJA interfejs**,